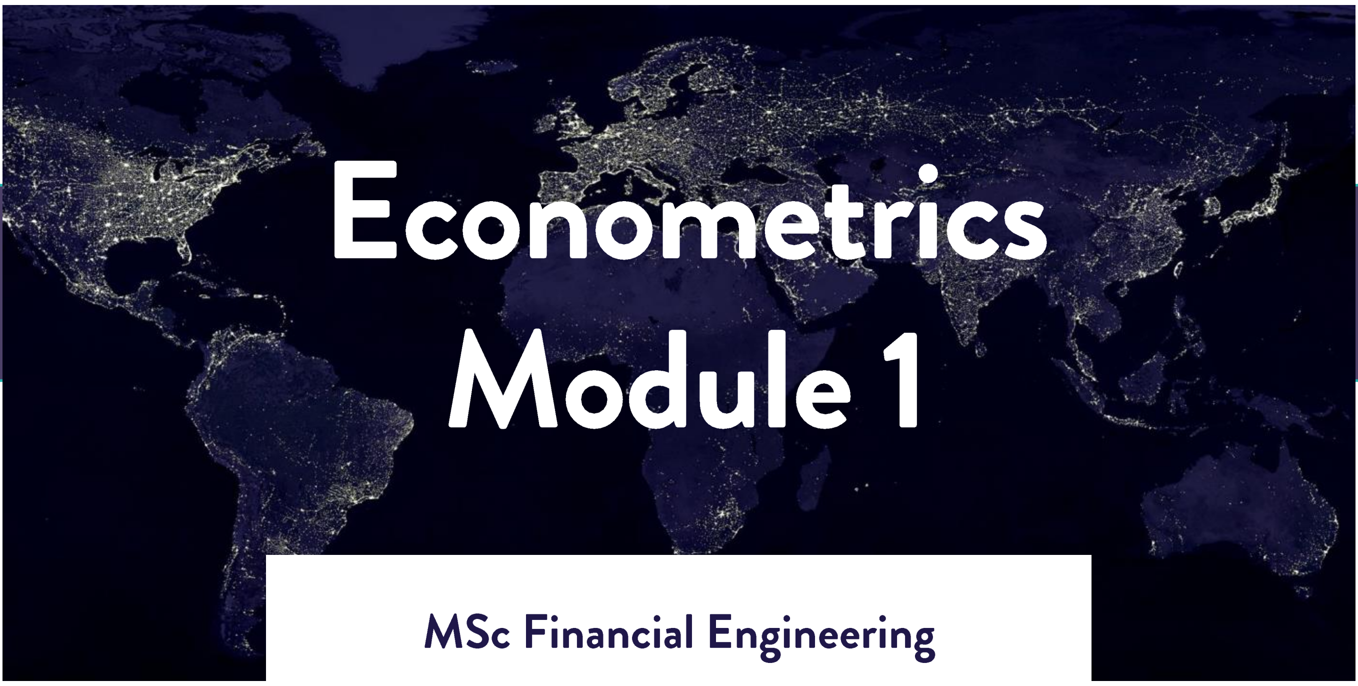
WORLDQUANT UNIVERSITY



Econometrics Module 1

MSc Financial Engineering

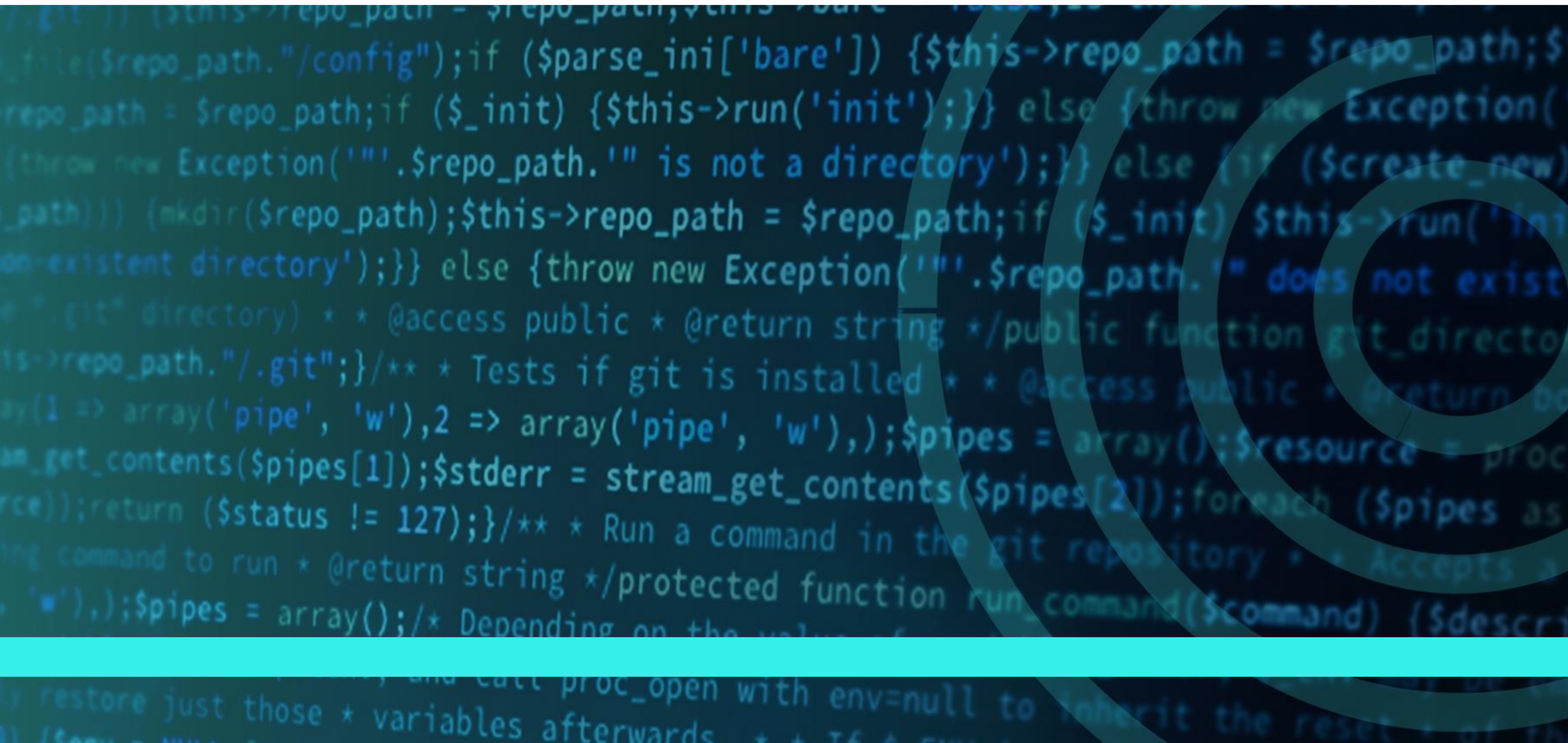


Table of Contents

1. Brief 2

2. Course Context 2

2.1 Course-level Learning Outcomes 3

2.2 Module Breakdown 4

3. Module 1: 5

Basic Statistics 5

3.1 Module-level Learning Outcomes 5

3.2 Transcripts and Notes 6

3.2.1 Notes: Introduction 6

3.2.2 Transcript: What is R? 8

3.2.3 Notes: Basics of R 10

3.2.4 Transcript: Basics of Using R 17

3.2.5 Notes: Using Scripts 24

3.2.7 Transcript: Quantile Plots 33

3 .2.8 Notes: Distributions 37

3.2.9 Notes: Types of Econometric Models 46

3.2.10 Notes: Finance Topics and Econometrics 53

3.3 Collaborative Review Task 67

Appendix 68

1. Brief

This document contains the core content for Module 1 of Econometrics, entitled Basic Statistics. It consists of three lecture video transcripts, seven sets of supplementary notes, and a collaborative review task.

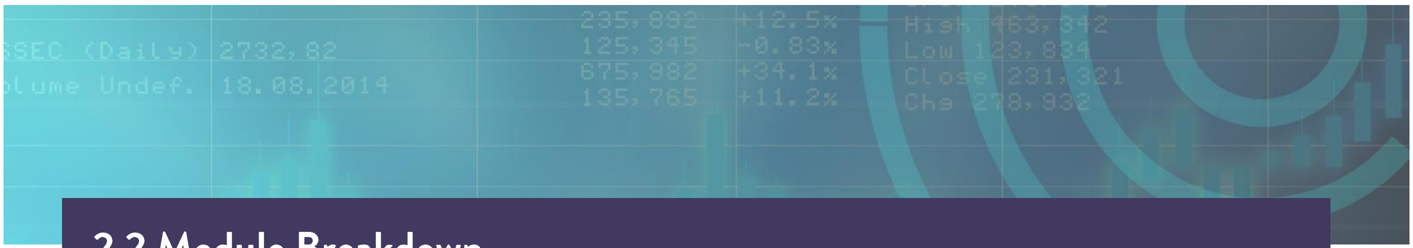
2. Course Context

Econometrics is the second course presented in the WorldQuant University (WQU) Master of Science in Financial Engineering (MScFE) program. In this course, you will apply statistical techniques to the analysis of econometric data. The course starts with an introduction to the R statistical programming languages that you will use to build econometric models, including multiple linear regression models, time series models, and stochastic volatility models. You will learn to develop programs using the R language, solve statistical problems, and understand value distributions in modeling extreme portfolio and basic algorithmic trading strategies. The course concludes with a review on applied econometrics in finance and algorithmic trading.

2.1 Course-level Learning Outcomes

Upon completion of the Econometrics course, you will be able to:

1. Write programs using the R language.
2. Use R packages to solve common statistical problems.
3. Formulate a generalized linear model and fit the model to data.
4. Use graphic techniques to visualize multidimensional data.
5. Apply multivariate statistical techniques (PCA, factor analysis, etc.) to analyze multidimensional data.
6. Fit a time series model to data.
7. Fit discrete-time volatility models.
8. Understand and apply filtering techniques to volatility modeling.
9. Understand the use of extreme value distribution in modeling.
10. Define some common risk measures like VaR and Expected Shortfall.
11. Define and use copulas in risk management.
12. Implement basic algorithmic trading strategies.



2.2 Module Breakdown

The Econometrics course consists of the following one-week modules:

1. Basic Statistics
2. Linear Models
3. Univariate Time Series Models
4. Univariate Volatility Modeling
5. Multivariate Time Series Analysis
6. Extreme Value Theory
7. Introduction to Risk Management

3. Module 1:

Basic Statistics

In this first module, we will explore the basics of coding and statistical analysis with R, the characteristics of univariate and multivariate distributions, and the important exploratory data techniques relevant to the financial engineer.

3.1 Module-level Learning Outcomes

After completing this module, you will be able to:

1. Write simple programs in R.
2. Use R packages to solve statistical problems.
3. Use R for visualization.

3.2 Transcripts and Notes

3.2.1 Notes: Introduction

Econometrics is the study of statistical methods that are able to extract useful information from economic data which, unlike most statistical data, is not experimental in origin. This makes econometrics different from classical statistics. We will delve into these differences in Module 2.

The first step of any econometric analysis is obtaining a strong understanding of the probabilistic properties of one's data. For this you need a strong statistical analysis package.

We will use the powerful and open source, community-based statistical programming language, R.

In this module, we will explore:

1. the basics of coding and statistical analysis with R;
2. the characteristics of univariate and multivariate distributions; and
3. the important exploratory data techniques relevant to the financial engineer.

In this course, we will not be exploring R in depth, but will rather introduce further features of R as we go along. You will find that this module is self-contained and includes everything you need to replicate each step performed in R.

For help with R installation and basic tutorials, there are a large set of online resources you can try, starting with the main sources and repositories:

Comprehensive R   Archive Network (https://cran.r-project.org/) and Quick R (https://www.statmethods.net/). If you need help on a package or module in R, the R documentation page is very helpful. It has a standard help-file type document for every module/method (https://www.rdocumentation.org/).

3.2.2 Transcript: What is R?

R is an open source project, which means that:

a) You can download the main package for free,
b) You have access to a vast array of R implementations written by the community that uses R, and
c) You can write and post your own contributions to the project.

In the context of the wonderful “open source revolution”, the R project is a prime example of the system working (almost) perfectly. You should keep in mind that this revolution is not flawless, however; there is always a trade-off. The fact that R is open source is both its biggest strength and weakness: anyone can use the project and anyone can contribute to it, which means quality and usefulness is not always guaranteed.

Not everyone who contributes packages to R are as meticulous about their contributions as the new user would like them to be. This means you should be careful and critical of any R package you use in your professional career. In particular, be wary of packages that have rarely been downloaded and view them with some rational conservatism. Popular and thoroughly tested packages will be more reliable, as they are likely to already have all their kinks sorted out.

R is so popular that, for the central parts of R that almost everyone uses, and every “often-used package”, a strong system of peer review and evaluation has organically evolved. This is why we can be sure the packages we use in this course are of good quality.

You should also be aware of the proprietary side of the statistical analysis world. If you work for a big player in the financial world, you will have access to an alternative group of programs that are often used for econometrics – for example, EViews, MATLAB, Mathematica, Stata, and a large number of alternatives. The relative benefit that these corporate (for-profit) programs have over the open source programs is that they employ professional coders that make sure that there are no bugs in any function or application. The downside is, of course, that these programs can be quite expensive.

You will have to work out which compromise is best for you, based on what you have access to. We will use R in this course as it is free, has a large network of users and evaluators and uses a syntax that is easy to learn.

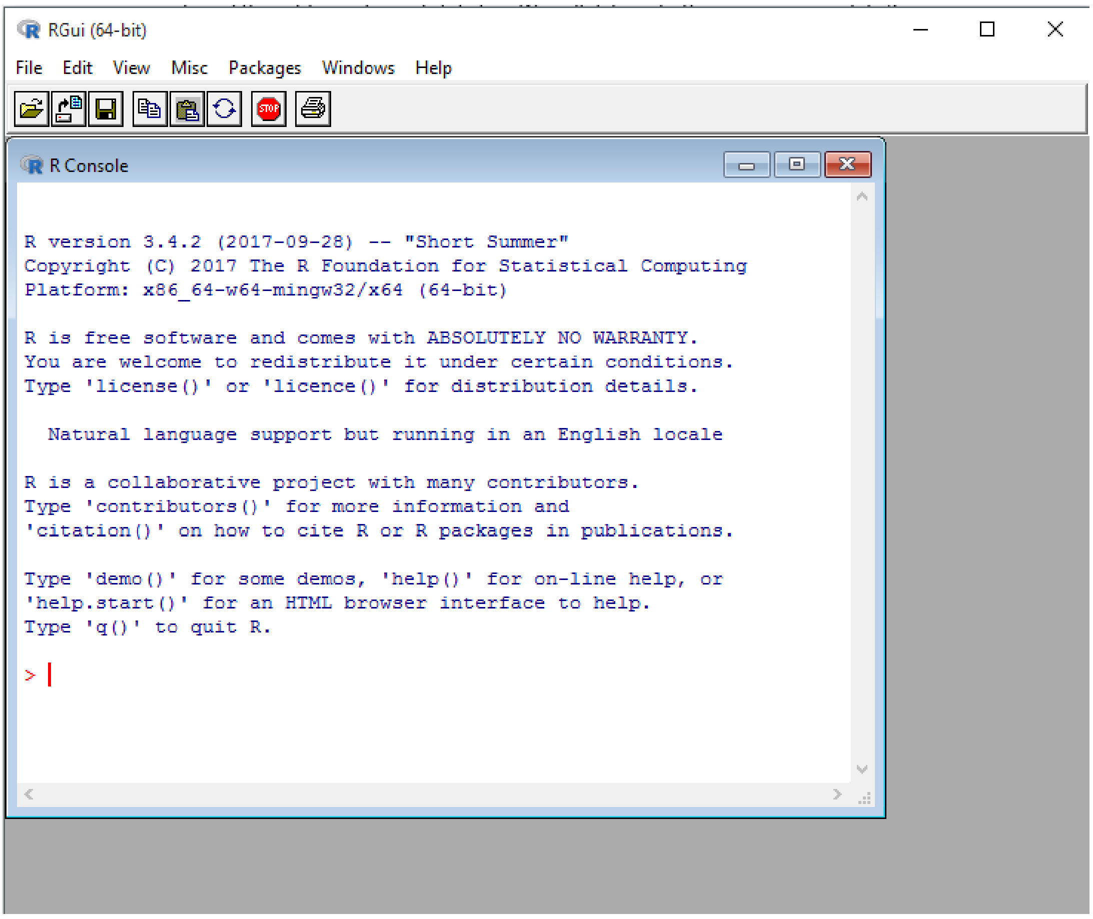
3.2.3 Notes: Basics of R

You can download the basic R program from CRAN (Comprehensive R Archive Network) available here: https://cran.r-project.org. CRAN also hosts all the packages from the network of contributors.

To get started, download and run through the basic installation on your system and then launch the graphical user interface (GUI).[1]

The GUI should look something like this:

[1] One can also use *R* from the command prompt of the operator system, which is useful for batch processing. In this module, we will stick to a GUI presentation as it is simpler and more intuitive, and thus faster to learn.



```
R version 3.4.2 (2017-09-28) -- "Short Summer"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> 
```

This basic GUI is not very informative, especially if one is new to R or used to programs like Eviews or MATLAB.

1. Obtaining packages

We will use the basic GUI for installing the necessary packages for this course.

The following convention will be followed in this module:

- Commands to be typed in R will be presented as `courier font in blue with beige background`.
- R output will be presented as `courier font in black with beige background`.
- Everything that follows a `#` is a R comment – i.e. R ignores this as text to explain something to the human user.

2. Choose a CRAN mirror

Choose a CRAN mirror site for the download. Type the following command in R:

```
chooseCRANmirror()
```

To make the download as fast as possible, choose the mirror nearest to you.

3. Installing packages

Next, we will install all the packages we will be using throughout this course, by running the following commands:

```
install.packages("tidyverse") # for all modules
install.packages("tseries") # for module 3
install.packages("forecast") # for most modules
install.packages("fGarch") # for module 4
install.packages("vars") # for module 5
install.packages("evir") # for module 6
install.packages("copula") # for module 7
```

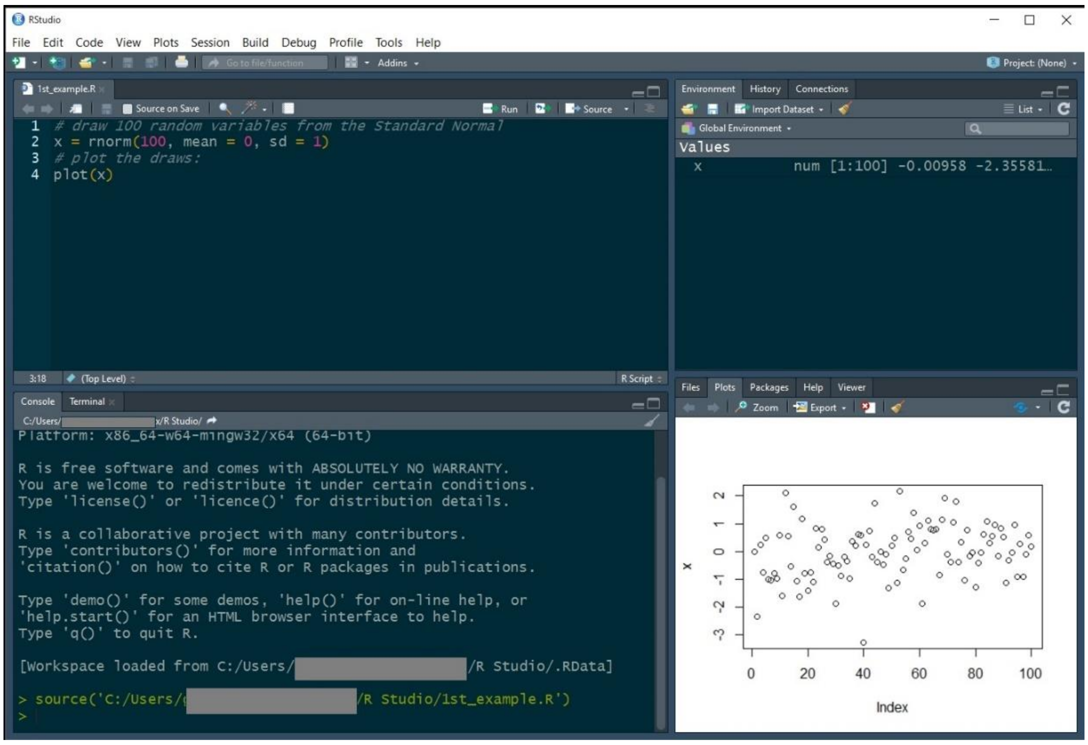
If you want to use a function or method in a package, you must place it in the current "global environment" so that R is able to call the functions in the package. To do this, use the command:

```
library(tidyverse) # loads the tidyverse package
```

Other GUI options:

There are some alternatives to the basic GUI available for you to use, such as RStudio. RStudio is a particularly creative suite of packages for R, with its own GUI that is very intuitive to use and makes interactivity easy. You can download the basic version for free at https://www.rstudio.com.

RStudio is also greatly customizable. Here is a screen capture of R studio set to dark grey (useful to lower eye-strain if you code for many hours at a time):



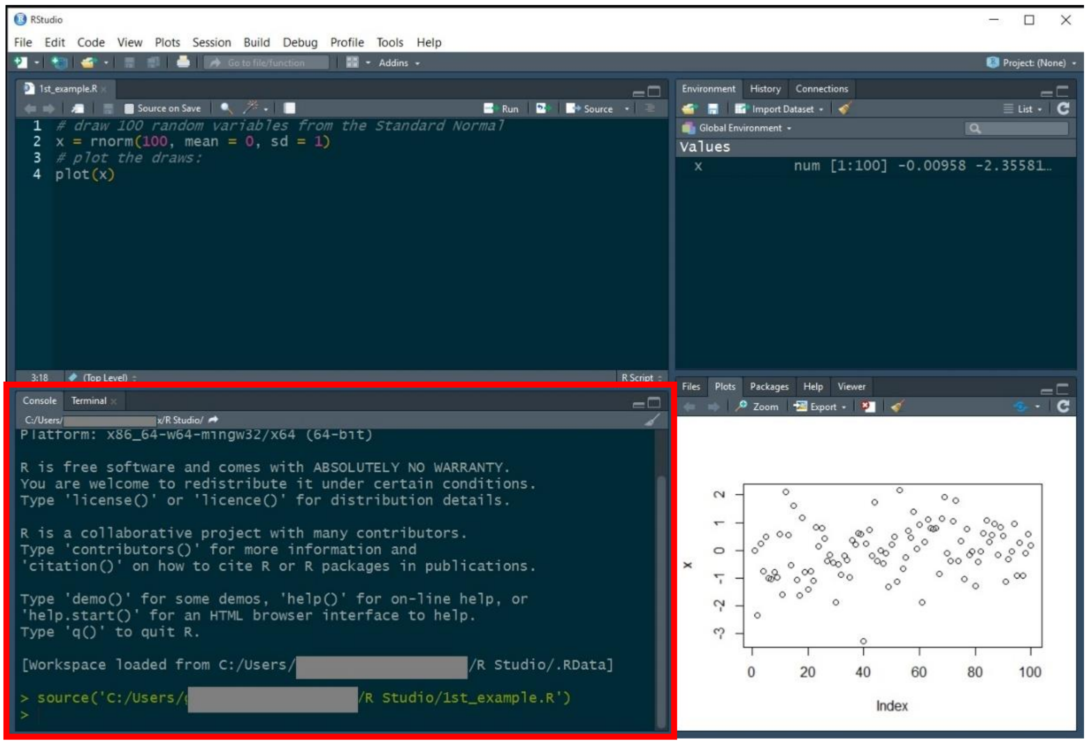
```
# draw 100 random variables from the Standard Normal
x = rnorm(100, mean = 0, sd = 1)
# plot the draws:
plot(x)
```

```
Values
  x              num [1:100] -0.00958 -2.35581…
```

```
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from C:/Users/[redacted]/R Studio/.RData]

> source('C:/Users/[redacted]/R Studio/1st_example.R')
> 
```

Figure 1

Figure 1 illustrates the main “windows” of Rstudio. First is the “terminal” where you can type in R commands directly (Figure 2):



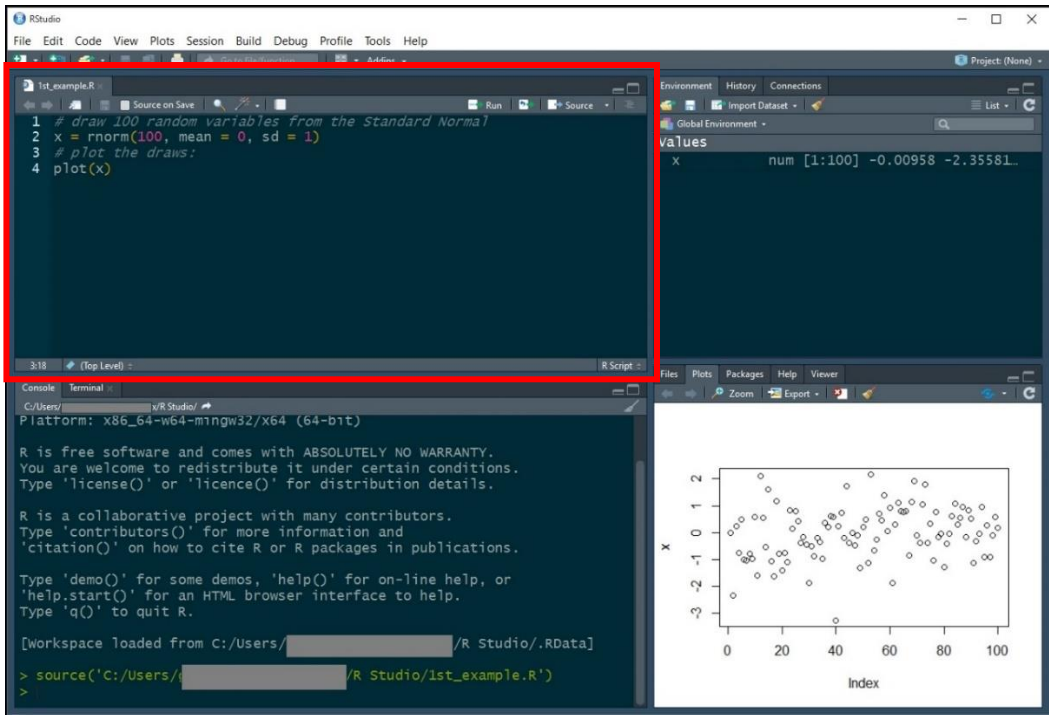
```r
# draw 100 random variables from the Standard Normal
x = rnorm(100, mean = 0, sd = 1)
# plot the draws:
plot(x)
```

```
Values
 x          num [1:100] -0.00958 -2.35581…
```

```
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from C:/Users/[redacted]/R Studio/.RData]

> source('C:/Users/[redacted]/R Studio/1st_example.R')
>
```

Figure 2

In Figure 3, the “script writer” where you can write a sequence of R commands, or, eventually, R programs, to be executed as a whole. In this example, the script 1st_example.R, when executed, draws 100 observations from a standard normal distribution and plots the draws against the index of the observation:

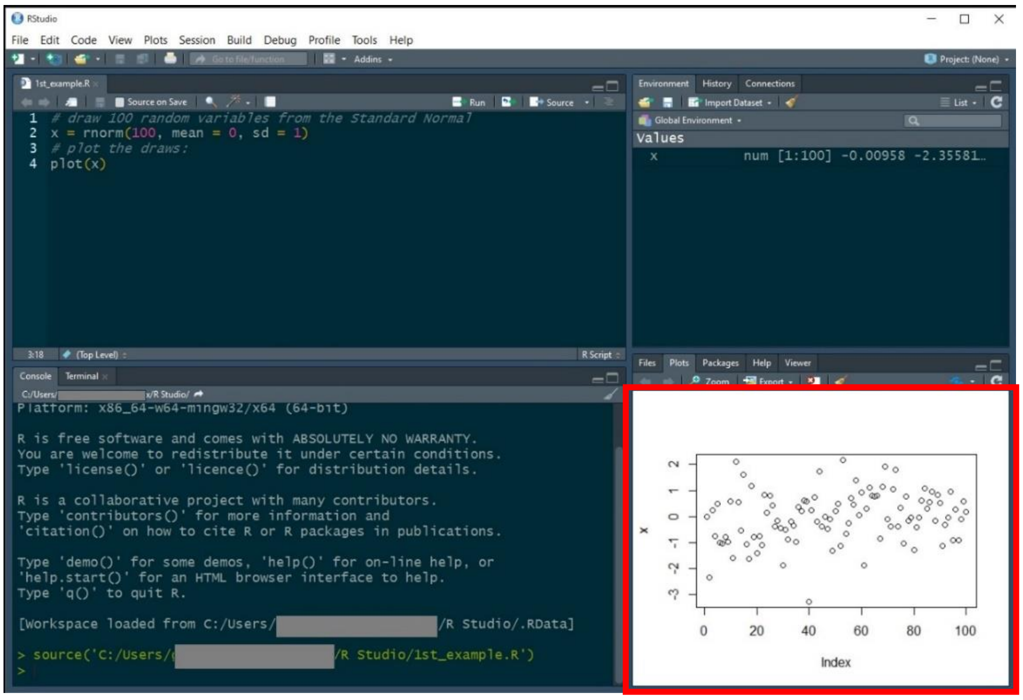

```
# draw 100 random variables from the Standard Normal
x = rnorm(100, mean = 0, sd = 1)
# plot the draws:
plot(x)
```

```
Values
  x          num [1:100] -0.00958 -2.35581…
```

```
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from C:/Users/[redacted]/R Studio/.RData]

> source('C:/Users/[redacted]/R Studio/1st_example.R')
>
```

Figure 3

In Figure 4, the plot output of these commands is displayed on the bottom right:



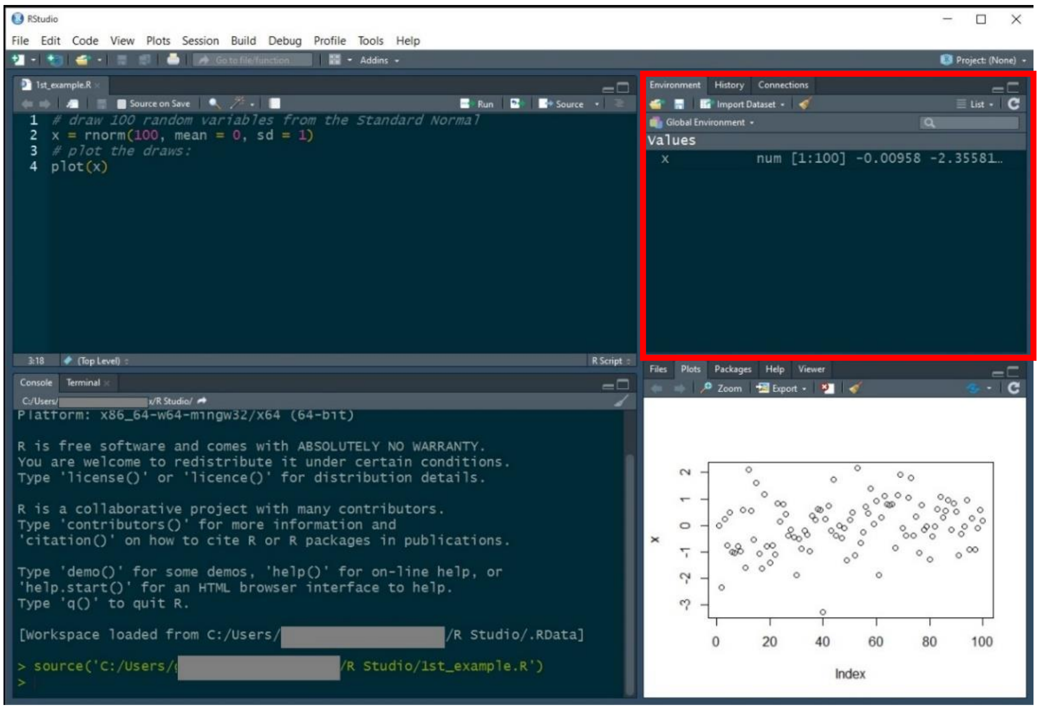
```
# draw 100 random variables from the Standard Normal
x = rnorm(100, mean = 0, sd = 1)
# plot the draws:
plot(x)
```

```
Values
  x          num [1:100] -0.00958 -2.35581…
```

```
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from C:/Users/[redacted]/R Studio/.RData]

> source('C:/Users/[redacted]/R Studio/1st_example.R')
>
```

Figure 4

Whatever variable or dataset you generate will be shown at the top right:



```
# draw 100 random variables from the Standard Normal
x = rnorm(100, mean = 0, sd = 1)
# plot the draws:
plot(x)
```

```
Values
  x          num [1:100] -0.00958 -2.35581…
```

```
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from C:/Users/[redacted]/R Studio/.RData]

> source('C:/Users/[redacted]/R Studio/1st_example.R')
>
```

Figure 5

In this example, we created 100 observations of a variable x that has a standard normal distribution. This means that each element of the 100-value vector is a random draw from a normal distribution with expected value 0 and standard deviation 1. In the rest of the course, we will consider various ways of visualizing and interpreting the meaning of each graph/ figure.

Getting used to the structure underlying the R environment will help you be efficient and productive with your analysis.

3.2.4 Transcript: Basics of Using R

Each statistical (or coding) language has its own *syntax*. This is the appropriate ways of calling functions and receiving their outputs. The R language is quite simple once you get used to it. It is therefore a critical part of your training to use R as frequently as possible so that it becomes “second nature” to convert any statistical problem into R code in your mind.

All statistical analyses eventually boil down to complicated combinations of basic operations: addition, subtraction, multiplication and division. Thus, it is unsurprising that you use it as a calculator:

```
23*35
[1] 805
```

Here you learn the first “storage type” that R encodes: unnamed numbers. Like in standard matrix algebra, we often refer to lists or vectors of numbers. R expects the same. The answer to our query 23*35 generated a scalar valued answer: 805.

In most instances, the financial engineer will work with many observations on several variables, the most common being vectors and matrices of numbers – e.g. returns on a group of assets.

Before we import data, let’s explore the standard matrix algebra computations that you can do with R.

Let’s assign a random vector to the name “a”:

```
library(stats)
a <- rnorm(3)
a
[1]  0.1481934 -0.3550795  0.8735789
```

and another to the vector to the name "b":

```
b <- rnorm(3).
```

Note the `<-` operator. This is R's "preferred" symbol to represent the operation "assign to". It is also possible to use the =, as we will demonstrate. You should take note of this, as it will make code developed by others accessible to you.

Next, we generate a matrix by using cbind (this binds arguments as columns, if conformable):

```
X = cbind(a,b)
X
              a          b
[1,]  0.1481934  0.7635945
[2,] -0.3550795 -0.1745421
[3,]  0.8735789  0.1133904
```

This now looks like a data matrix: three observations on two variables, with names a and b stored as column headings of the variables.

We can consider element-by-element multiplication:

```
X*X
              a          b
[1,] 0.02196129 0.58307659
[2,] 0.12608144 0.03046493
[3,] 0.76314009 0.01285737
```

We can also consider the inner product $\mathbf{X}'\mathbf{X}$ (where t`()` is the transpose function – its output is the matrix transpose of its input, and $\% * \%$ denotes matrix multiplication):

```
t(X)%*%X
          a         b
a 0.9111828 0.2741914
b 0.2741914 0.6263989
```

Or $\mathbf{XX}'$:

```
X%*%t(X)
           [,1]       [,2]       [,3]
[1,]  0.6050379 -0.1858998  0.2160429
[2,] -0.1858998  0.1565464 -0.3299813
[3,]  0.2160429 -0.3299813  0.7759975
```

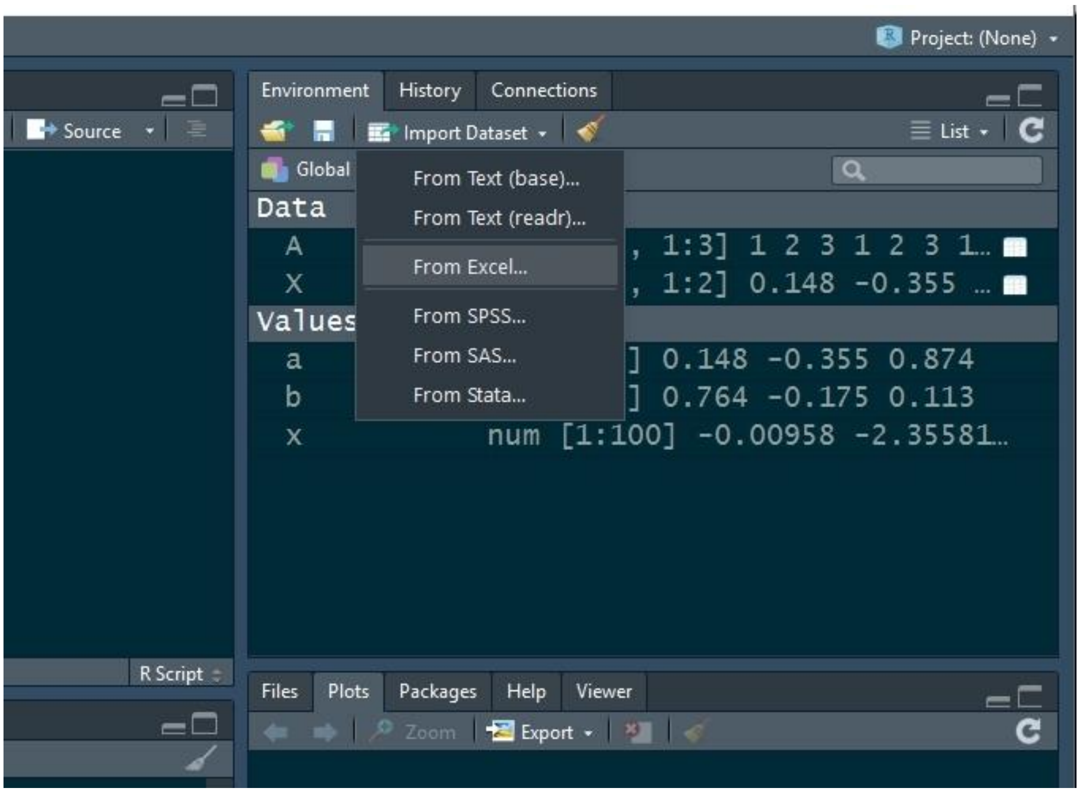
Note how R preserves the column names and row numbers for the two operations.

All statistical packages use something like these linear algebra results, but all pre-coded and applied to data types. A handy reference for the syntax of basic operation is available from Quick-R (https://www.statmethods.net/advstats/matrix.html).

Next, let's upload some data. We will be performing exploratory data analysis in this module in the form of a tutorial on R, and developing the content of exploratory data analysis and some features of the multivariate normal and *t* distributions.

The tidyverse packages which is part of the R-studio is a convenient and powerful data management option, which we will use to some extent in this course. It makes use of readxl, which is a simple way of importing Excel data into an R data structure.

In the Rstudio GUI, it is even simpler.
Under Global Environment, select "Import Dataset" and "From Excel" from the drop-down menu:

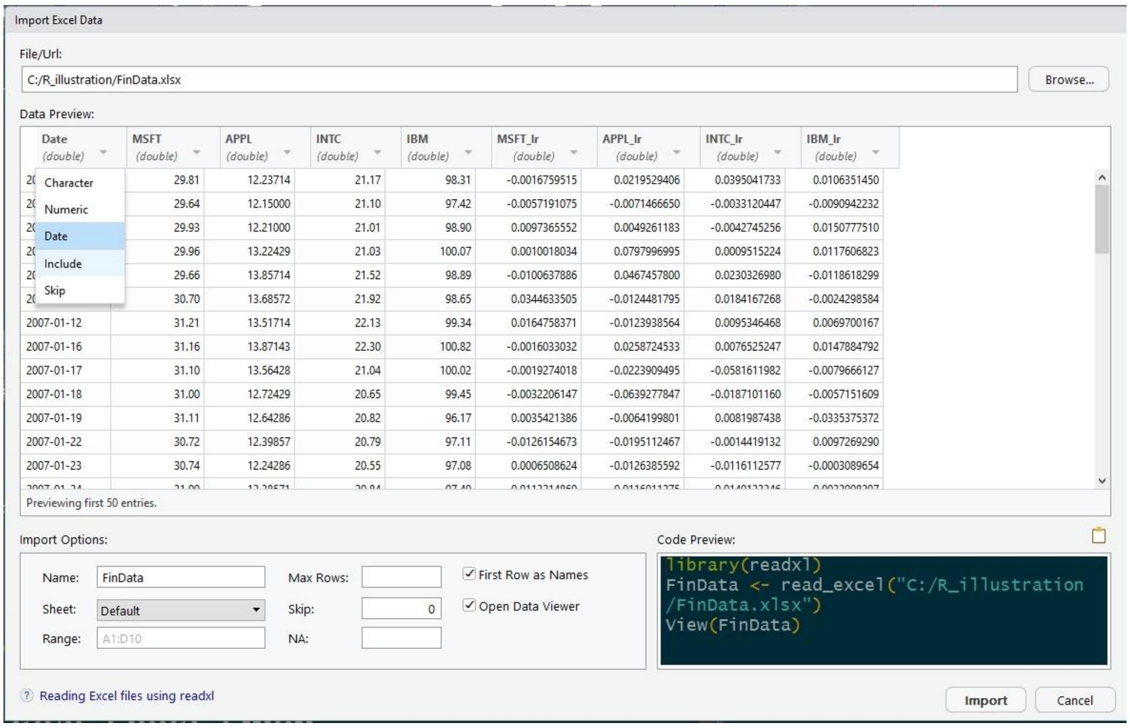


```
Data
  A            , 1:3] 1 2 3 1 2 3 1…
  X            , 1:2] 0.148 -0.355 …
Values
  a            ] 0.148 -0.355 0.874
  b            ] 0.764 -0.175 0.113
  x           num [1:100] -0.00958 -2.35581…
```

Browse to the location of your data and select the relevant dataset. You should see a preview of the dataset, assuming each column contains the same "arbitrary numerical data".

In this case the first column is the date of observation, which we want to use in our analyses and graphs.

Select the tab under the column label "Date" and select the Date data type.



File/Url: C:/R_illustration/FinData.xlsx

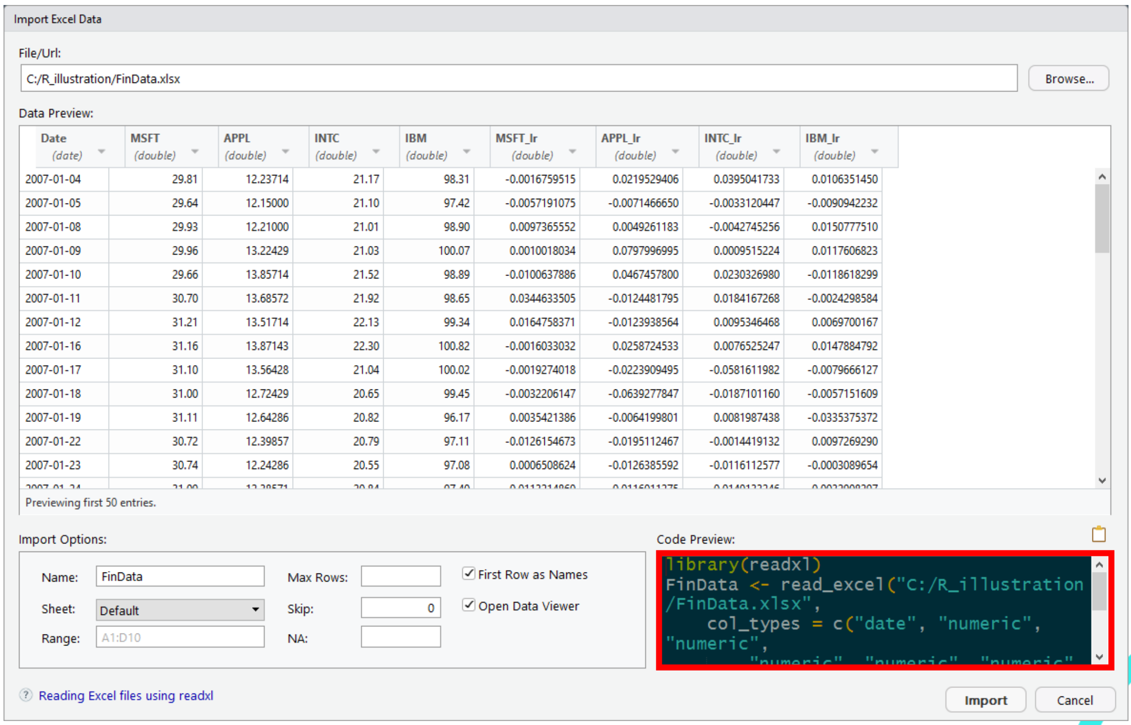
Date (double)	MSFT (double)	APPL (double)	INTC (double)	IBM (double)	MSFT_lr (double)	APPL_lr (double)	INTC_lr (double)	IBM_lr (double)
20	29.81	12.23714	21.17	98.31	-0.0016759515	0.0219529406	0.0395041733	0.0106351450
20	29.64	12.15000	21.10	97.42	-0.0057191075	-0.0071466650	-0.0033120447	-0.0090942232
20	29.93	12.21000	21.01	98.90	0.0097365552	0.0049261183	-0.0042745256	0.0150777510
20	29.96	13.22429	21.03	100.07	0.0010018034	0.0797996995	0.0009515224	0.0117606823
20	29.66	13.85714	21.52	98.89	-0.0100637886	0.0467457800	0.0230326980	-0.0118618299
20	30.70	13.68572	21.92	98.65	0.0344633505	-0.0124481795	0.0184167268	-0.0024298584
2007-01-12	31.21	13.51714	22.13	99.34	0.0164758371	-0.0123938564	0.0095346468	0.0069700167
2007-01-16	31.16	13.87143	22.30	100.82	-0.0016033032	0.0258724533	0.0076525247	0.0147884792
2007-01-17	31.10	13.56428	21.04	100.02	-0.0019274018	-0.0223909495	-0.0581611982	-0.0079666127
2007-01-18	31.00	12.72429	20.65	99.45	-0.0032206147	-0.0639277847	-0.0187101160	-0.0057151609
2007-01-19	31.11	12.64286	20.82	96.17	0.0035421386	-0.0064199801	0.0081987438	-0.0335375372
2007-01-22	30.72	12.39857	20.79	97.11	-0.0126154673	-0.0195112467	-0.0014419132	0.0097269290
2007-01-23	30.74	12.24286	20.55	97.08	0.0006508624	-0.0126385592	-0.0116112577	-0.0003089654

Previewing first 50 entries.

Name:	FinData	Max Rows:		☑ First Row as Names	
Sheet:	Default	Skip:	0	☑ Open Data Viewer	
Range:	A1:D10	NA:			

```
library(readxl)
FinData <- read_excel("C:/R_illustration
/FinData.xlsx")
View(FinData)
```

Now you can click on "Import", but before you do, note that the necessary code is provided for you. You can copy and paste this into a script so that you can repeat your analyses without having to manually import the dataset every time.



File/Url: C:/R_illustration/FinData.xlsx

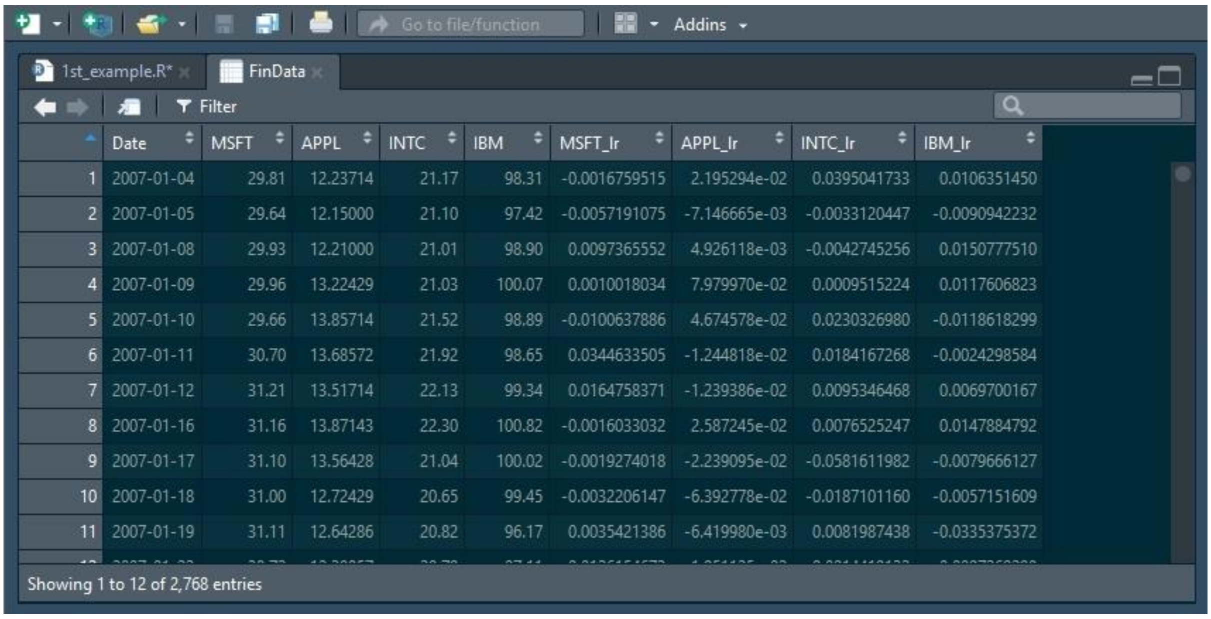
Date (date)	MSFT (double)	APPL (double)	INTC (double)	IBM (double)	MSFT_lr (double)	APPL_lr (double)	INTC_lr (double)	IBM_lr (double)
2007-01-04	29.81	12.23714	21.17	98.31	-0.0016759515	0.0219529406	0.0395041733	0.0106351450
2007-01-05	29.64	12.15000	21.10	97.42	-0.0057191075	-0.0071466650	-0.0033120447	-0.0090942232
2007-01-08	29.93	12.21000	21.01	98.90	0.0097365552	0.0049261183	-0.0042745256	0.0150777510
2007-01-09	29.96	13.22429	21.03	100.07	0.0010018034	0.0797996995	0.0009515224	0.0117606823
2007-01-10	29.66	13.85714	21.52	98.89	-0.0100637886	0.0467457800	0.0230326980	-0.0118618299
2007-01-11	30.70	13.68572	21.92	98.65	0.0344633505	-0.0124481795	0.0184167268	-0.0024298584
2007-01-12	31.21	13.51714	22.13	99.34	0.0164758371	-0.0123938564	0.0095346468	0.0069700167
2007-01-16	31.16	13.87143	22.30	100.82	-0.0016033032	0.0258724533	0.0076525247	0.0147884792
2007-01-17	31.10	13.56428	21.04	100.02	-0.0019274018	-0.0223909495	-0.0581611982	-0.0079666127
2007-01-18	31.00	12.72429	20.65	99.45	-0.0032206147	-0.0639277847	-0.0187101160	-0.0057151609
2007-01-19	31.11	12.64286	20.82	96.17	0.0035421386	-0.0064199801	0.0081987438	-0.0335375372
2007-01-22	30.72	12.39857	20.79	97.11	-0.0126154673	-0.0195112467	-0.0014419132	0.0097269290
2007-01-23	30.74	12.24286	20.55	97.08	0.0006508624	-0.0126385592	-0.0116112577	-0.0003089654

Previewing first 50 entries.

Name:	FinData	Max Rows:		☑ First Row as Names	
Sheet:	Default	Skip:	0	☑ Open Data Viewer	
Range:	A1:D10	NA:			

```
library(readxl)
FinData <- read_excel("C:/R_illustration
/FinData.xlsx",
    col_types = c("date", "numeric",
"numeric",
```

Now you should see your data open in the viewer at the top left:



	Date	MSFT	APPL	INTC	IBM	MSFT_lr	APPL_lr	INTC_lr	IBM_lr
1	2007-01-04	29.81	12.23714	21.17	98.31	-0.0016759515	2.195294e-02	0.0395041733	0.0106351450
2	2007-01-05	29.64	12.15000	21.10	97.42	-0.0057191075	-7.146665e-03	-0.0033120447	-0.0090942232
3	2007-01-08	29.93	12.21000	21.01	98.90	0.0097365552	4.926118e-03	-0.0042745256	0.0150777510
4	2007-01-09	29.96	13.22429	21.03	100.07	0.0010018034	7.979970e-02	0.0009515224	0.0117606823
5	2007-01-10	29.66	13.85714	21.52	98.89	-0.0100637886	4.674578e-02	0.0230326980	-0.0118618299
6	2007-01-11	30.70	13.68572	21.92	98.65	0.0344633505	-1.244818e-02	0.0184167268	-0.0024298584
7	2007-01-12	31.21	13.51714	22.13	99.34	0.0164758371	-1.239386e-02	0.0095346468	0.0069700167
8	2007-01-16	31.16	13.87143	22.30	100.82	-0.0016033032	2.587245e-02	0.0076525247	0.0147884792
9	2007-01-17	31.10	13.56428	21.04	100.02	-0.0019274018	-2.239095e-02	-0.0581611982	-0.0079666127
10	2007-01-18	31.00	12.72429	20.65	99.45	-0.0032206147	-6.392778e-02	-0.0187101160	-0.0057151609
11	2007-01-19	31.11	12.64286	20.82	96.17	0.0035421386	-6.419980e-03	0.0081987438	-0.0335375372

Showing 1 to 12 of 2,768 entries

In this dataset there are the daily closing prices of the following stocks: Microsoft (MSFT), Apple (APPL), Intel (INTC), and IBM (IBM), along with their implied log returns indicated with an _lr.

This data is stored in an object with the name, tibble, which is part of the tidyverse framework.

We will not work extensively with this form, but it is worth reading up on. For our purposes, all we need to know is how to work with objects.

An object can be thought of as a "folder" that stores various things. In our example, the FinData object contains "elements", each being a column with a specific name, data type and values.

To access/manipulate an element of the object, we use the $ operator (since this is an S3 object. There is also a different structure, an S4 object, where we use the operator

@ to access an element – beyond this distinction they work the same for our purposes).

Thus, to compute the mean of the series MSFT:

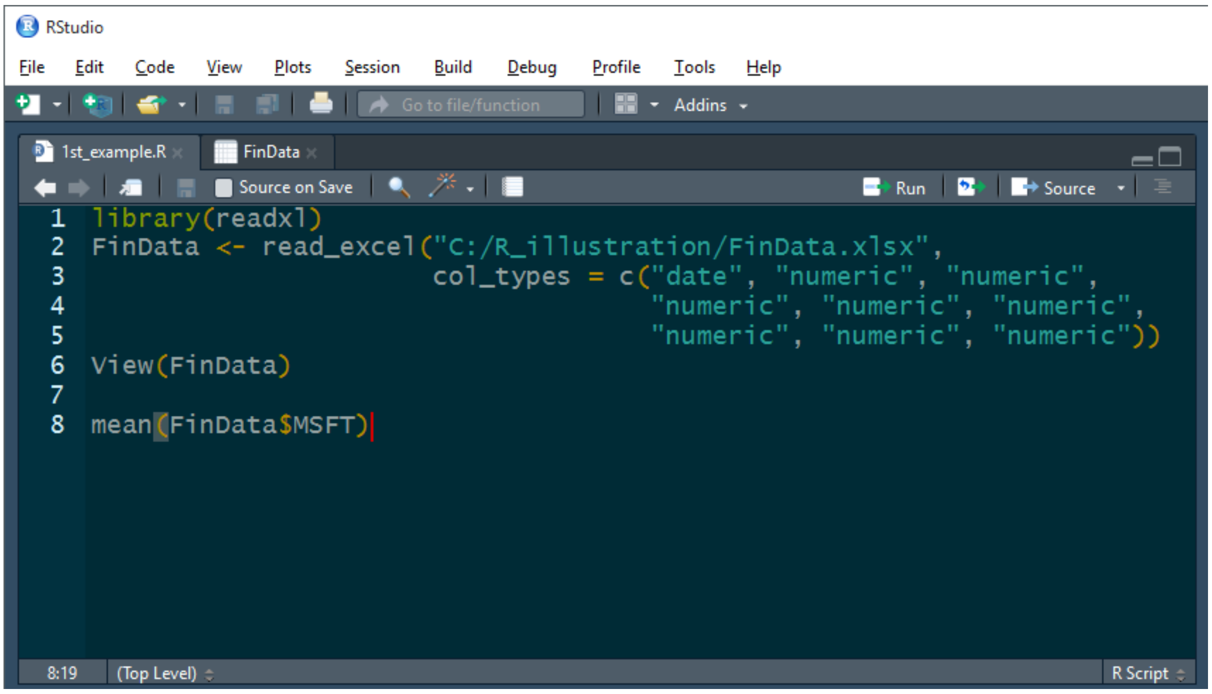
```
mean(FinData$MSFT)
[1] 37.44039
```

where the argument of the function mean is the element of the object `FinData` with name `MSFT`. An error would have occurred if we tried to reference an element not in `FinData`. You can thus think of objects as filing systems just like the systems on your computer.

3.2.5 Notes: Using Scripts

So far, we have been exercising the basics of R in the command window, but this is not a good way of doing complicated analyses that need to be replicable. In practice, you should always do your code-writing in a script, or a set of scripts and functions as you become more proficient with coding.

For instance, our data import step and mean calculation could be written up in a script, and all executed at once by clicking on "source" or executed line-by-line by using the key combination ctrl-enter.

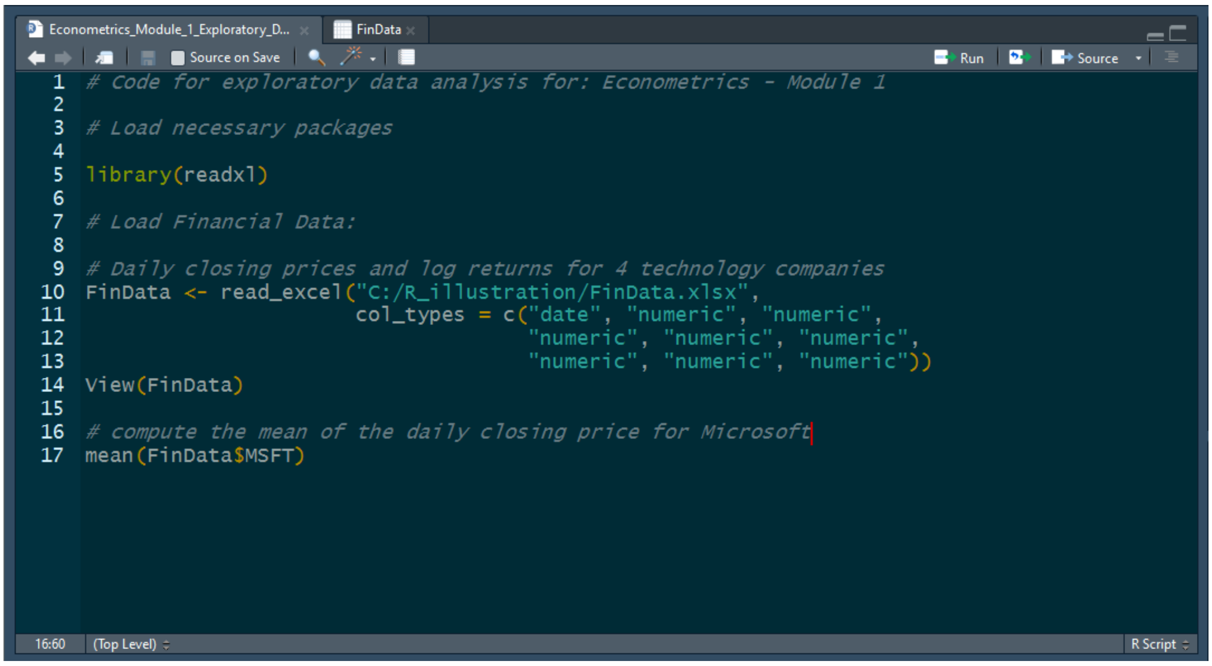


```r
library(readxl)
FinData <- read_excel("C:/R_illustration/FinData.xlsx",
                      col_types = c("date", "numeric", "numeric",
                                    "numeric", "numeric", "numeric",
                                    "numeric", "numeric", "numeric"))
View(FinData)

mean(FinData$MSFT)
```

You will be writing a lot of code for different applications in this course. Therefore, it is best practice to keep to a meticulous commenting regime. Every step needs an explanation of why it's done, otherwise you will waste time later in trying to reconstruct your thoughts. There are often many different ways of coding the same function, so keeping track of how you have coded functions is important.

In R you can use the # character to make comments. R ignores everything after # in a line of code when it runs blocks of code.

As an example, you could begin your work with something like this, allowing anyone to read your code as easily as you do.



```r
# Code for exploratory data analysis for: Econometrics - Module 1

# Load necessary packages

library(readxl)

# Load Financial Data:

# Daily closing prices and log returns for 4 technology companies
FinData <- read_excel("C:/R_illustration/FinData.xlsx",
                       col_types = c("date", "numeric", "numeric",
                                     "numeric", "numeric", "numeric",
                                     "numeric", "numeric", "numeric"))
View(FinData)

# compute the mean of the daily closing price for Microsoft
mean(FinData$MSFT)
```

Next, we will turn to some systematic exploratory data analysis to learn the characteristics of our data. One can get misleading results if you apply a method to data it is not created for. As we will be developing a large array of methods in this course, we need to understand the characteristics of our data to help us select the correct models.

Exploratory data analysis

The financial engineer will inevitably be interested in investing in a portfolio of assets to obtain his or her goals. You will learn a lot about the optimal construction of such a portfolio during this degree so to begin with we will just keep to the basics.

For you to know whether or not to invest in a portfolio of two assets, you would need to understand how they are likely to move individually, but also to understand how they are likely to have the same stochastic trend. If two stocks are anything less than perfectly correlated, a way to reduce portfolio risk is to lower than either of the component assets will always exist.

The financial engineer works with uncertain variables that evolve over time. For example, the returns on different banks may behave differently as a group during "normal times" to how they would behave during a housing boom or a banking sector crisis.

The properties we will impose on this time-varying behavior are captured statistically by the joint cumulative probability distribution, and it is the features of this mathematical object that we will study in this module. We will apply the joint cumulative probability distribution to some real-world data to extract some useful information. This information will tell us which of the models we develop in this course will be applicable to a certain set of data.

Time plots

The first step in exploratory data analysis is a visualization of the variable of interest over time. This is called a "time plot".

We will use the ggplot packages that form part of tidyverse/Rstudio package. The commands (taken from the R script attached to this module) are:

```
# Basic Time Plots:
# Load necessary packages
library(tidyverse)
library(ggplot2)
# view time plots of the closing price of Microsoft:
ggplot(data = FinData) +
       geom_point(mapping = aes(x = Date,y = MSFT),
       color = "darkblue") +
       labs(x = "year", y = "Microsoft - Daily Closing Price")
ggsave("C:/R_illustration/Microsoft_closing_price.png")
# generate plots for each of the log return graphs of the four assets
ggplot(data = FinData)
       + geom_line(mapping = aes(x = Date,y = APPL_lr),
                    color = "darkred") +
       labs(x = "year",y = "Apple - Log Daily Return")
ggsave("C:/R_illustration/Apple_log_returns.png")
ggplot(data = FinData)
       +geom_line(mapping = aes(x = Date,y = INTC_lr),
                   color ="darkgreen")
       +labs(x = "year",y = "Intel - Log Daily Return")
ggsave("C:/R_illustration/Intel_log_returns.png")
ggplot(data = FinData)
       + geom_line(mapping = aes(x = Date, y = IBM_lr),
       color = "darkcyan")
       +labs(x = "year",y = "IBM - Log Daily Return")
ggsave("C:/R_illustration/IBM_log_returns.png")
```

The ggplot method works as follows:

The main call establishes the dataset involved and can recognize dated data like ours without any further human input. This is one of the reasons why ggplot packages are great options.

For every additional element you wish to add to the graph, you add a `+ geom_XXX` call. In our example we will use:

`+ geom_line(…)`

This calls for a line graph to be created.

Its arguments mean the following:

`mapping = aes(x = Date,y = MSFT)`

Add an “aesthetic”, i.e., a visual graph, that maps the x variable date to the y variable MSFT. Since the calling function is “geom_line” the result will be a line plot. It would have been a scatter plot if the calling function had been “geom_point”.

`color = “darkblue”`

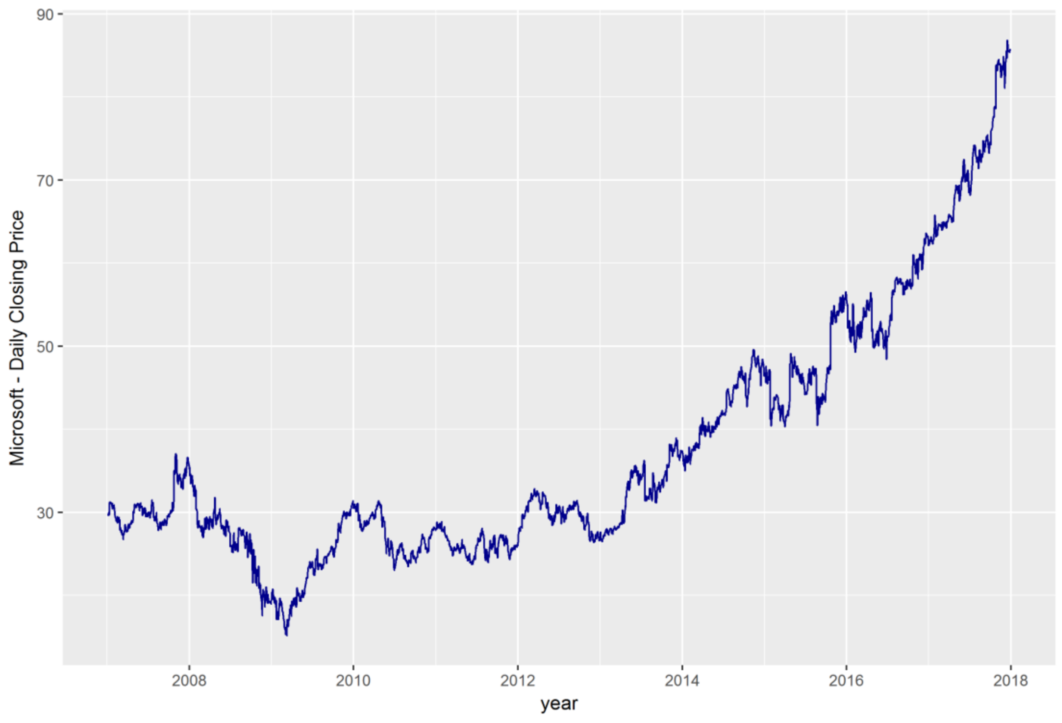
This sets the color of the line to a color in ggplot called “darkblue”.

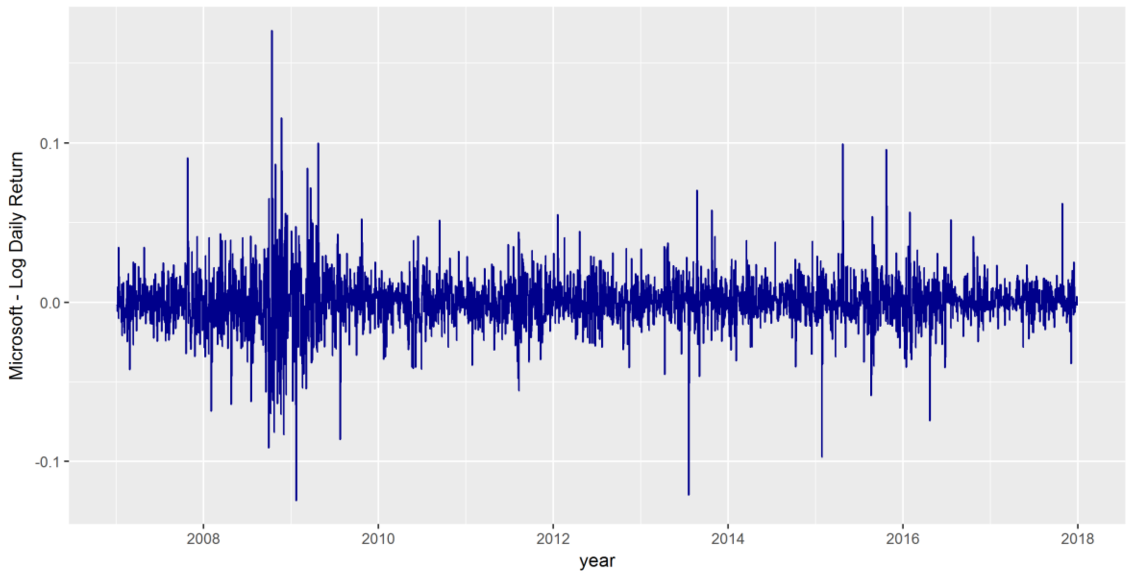
Lastly, we will add another component to the figure, the labels:

`+labs( x ="year",y = "Microsoft - Daily Closing Price")`

The `ggsave` command saves a png file of the graph you generated.

Compare the time plot of the price of the series to that of its implied log returns by viewing the two graphs you just generated:

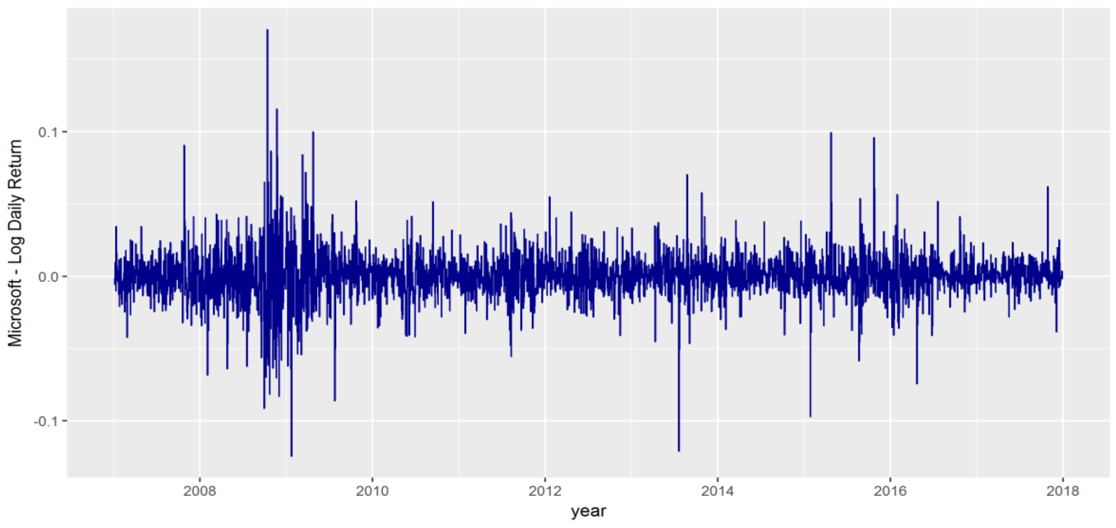


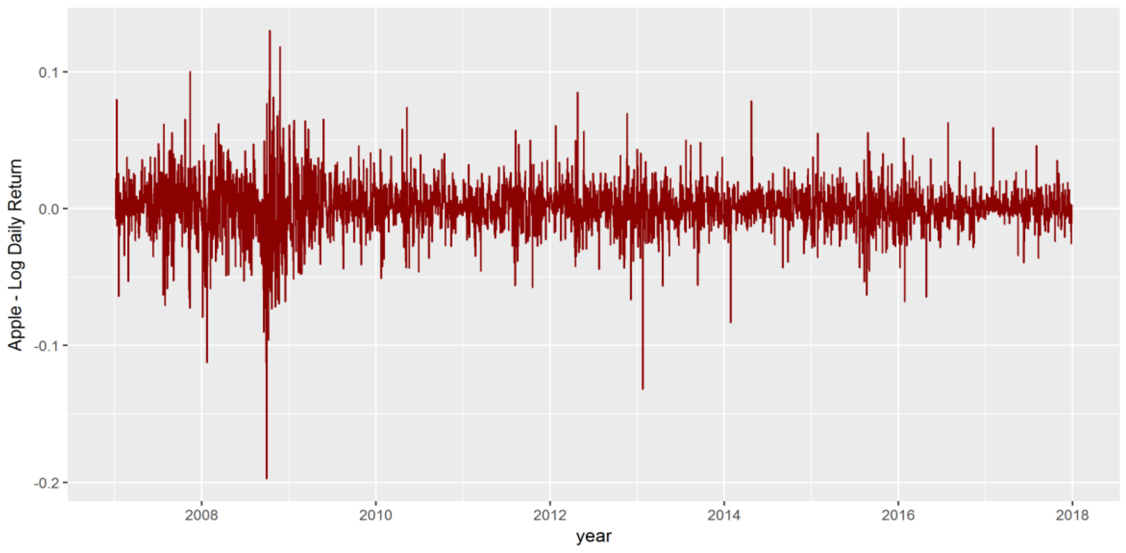


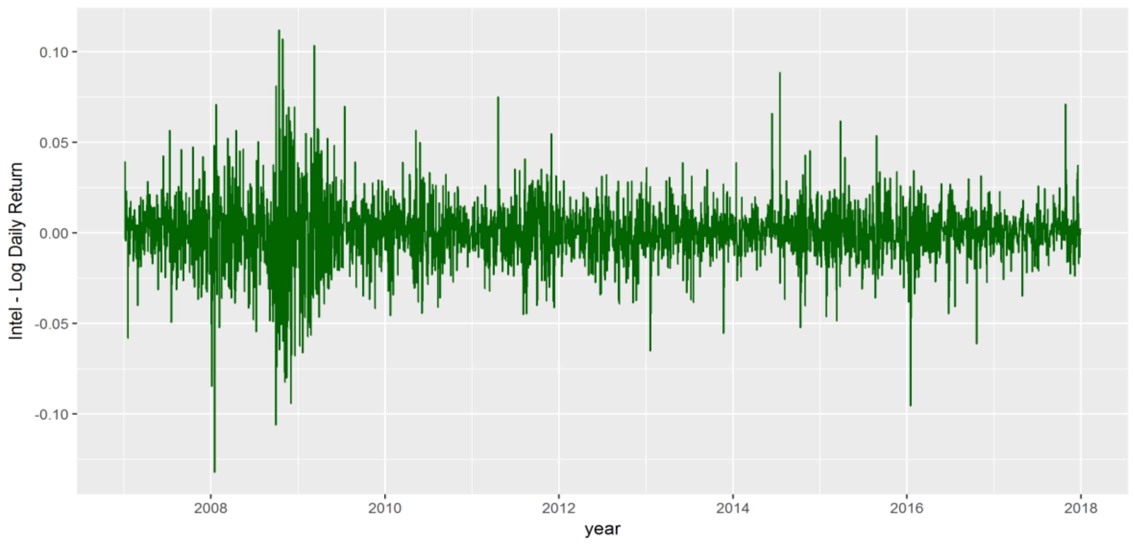
We immediately notice a stark difference: the closing price seems to meander upward while the log returns fluctuate around a visually constant mean. The closing price exhibits properties of non-stationarity, while we would call the log returns stationary. Module 3: Univariate Time Series Models will go into more detail about what the difference is. For now, we will focus on the stationary log return series to introduce the concepts of exploratory data analysis in the static setting. All of these methods have extensions to the non-stationary world that you should explore as necessary in your career.

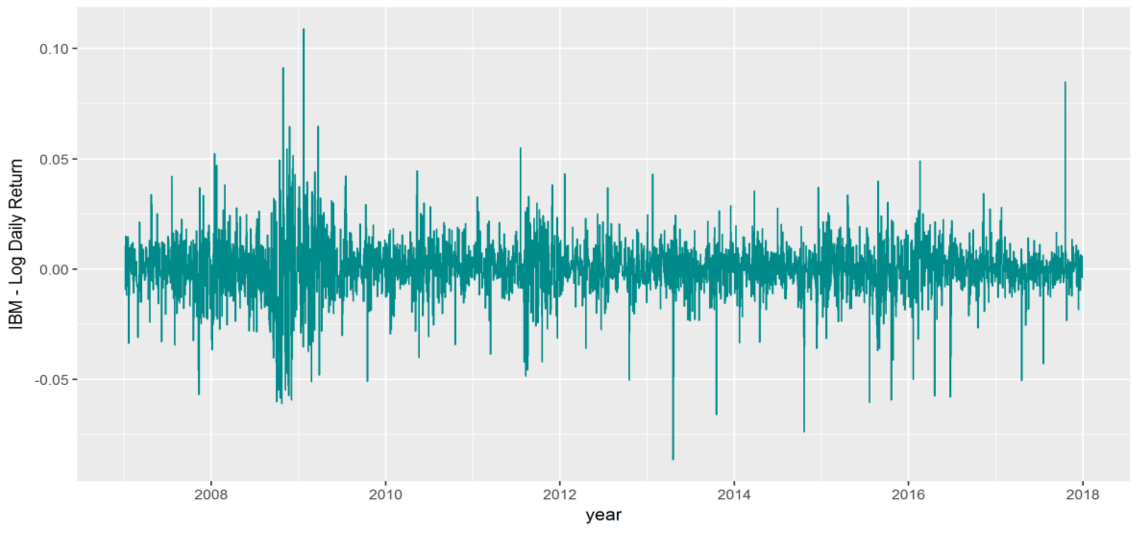
Of key interest is how two or more series move together over time. A first step we can take is to plot line graphs of each and compare obvious trends.

Let's consider the time plots of the four assets in the data set for this module:


Microsoft - Log Daily Return
0.1
0.0
-0.1
2008
2010
2012
2014
2016
2018
year


Apple - Log Daily Return
0.1
0.0
-0.1
-0.2
2008
2010
2012
2014
2016
2018
year


Intel - Log Daily Return
0.10
0.05
0.00
-0.05
-0.10
2008
2010
2012
2014
2016
2018
year



There are some similarities and some distinct aspects to these time plots. We observe that:

a) They all have seemingly constant means over the sample – there are clearly no long-term deviations from a mean just above zero.

b) The variance of each series shows periods of high volatility followed by periods of low volatility.

c) There are sometimes clusters of volatility across series, most notably the universal increase in volatility somewhere between 2008 and 2010, which corresponds to the global financial crisis.

Stylized facts of financial asset returns

With formal testing, which we study in this course, we can find a set of stylized facts that describes most returns series of financial assets. :

a) Almost no auto-correlation

Asset return levels tend to be uncorrelated with their own past, except at very low frequency data.

b) **Heavier tails than normal distribution (conditional heavy tails here as well)**

   There is a higher probability of extreme events in empirical distributions of asset returns than in the best fitting normal distribution.

c) **Asymmetric gain/loss behavior**

   We observe large negative returns more often than large positive returns.

d) **Aggregation to normality**

   The lower the frequency of observation (e.g. monthly vs. daily) the closer the return resembles a normally distributed variable.

e) **Volatility clustering**

   We observe periods of high volatility followed by periods of relative tranquility.

f) **Auto-correlated absolute values**

   The absolute value of a return series tends to be auto-correlated. This would not happen with a normally distributed variable.

g) **Leverage effect**

   There tends to be a negative correlation between the volatility and returns of an asset.

h) **Correlation between volume and volatility**

   Trading volumes tends to be correlated with any measure of volatility.

Before we consider co-movements (or correlations), we study individual marginal distributions. That is, we consider each returns series as a set of independent draws from a constant distribution.

3.2.7 Transcript: Quantile Plots

A useful graphical device to check coherence with a specific parametric distribution is a quantile-quantile plot.

This is the following construction for some sample: $x_1 < x_2 < \cdots < x_n$.

Consider a sample of n draws of the distribution of a random variable x. Order them from smallest to largest so that $x_1 < x_2 < \cdots < x_n$.

Standardize the sample distribution to have a zero mean and unit standard deviation. That is, construct:

$$z_i = \frac{x_i - mean(x_i)}{stddev(x_i)}.$$

Where $mean(x_i)$ is the sample mean of the variable and $stddev(x_i)$ is the sample standard deviation of the variable. This means that the variable z will have zero mean and standard deviation of 1.

Recall that the empirical distribution would assign probability:

$$F_{emp}(z_i) = \frac{i}{n+1}$$

to the i^{th} smallest element of the observations of the standardized quantiles z_i.

If the empirical distribution is very different from the normal distribution, the normal distribution would assign different quantile values to each of the values of the empirical distribution function.

Thus we construct the normal quantiles corresponding to the probability values of the empirical distribution.

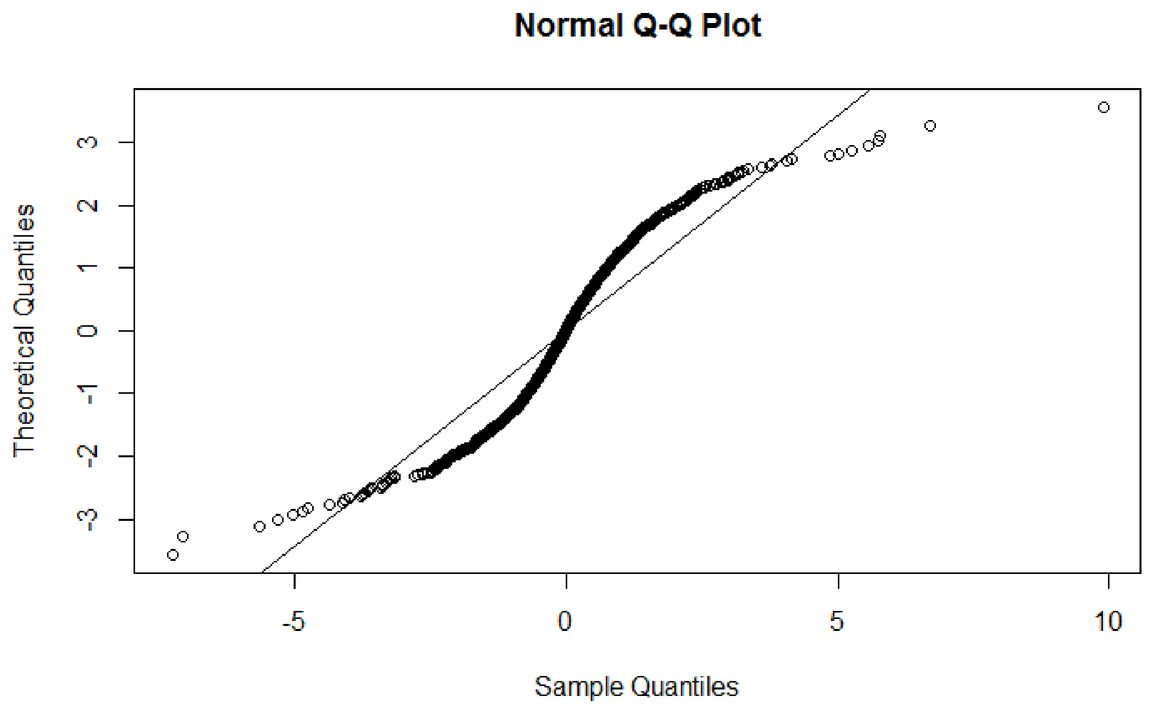
Let $F_{norm}(q)$ be the cumulative normal distribution. Then the normal quantiles $q_{norm,i}$ corresponding to the standardized empirical quantiles (z_i) are defined as:

$$q_{norm,i} = F_{norm}^{-1}\left(\frac{i}{n+1}\right).$$

The pattern of the plot of z_i: $q_{norm,i}$ gives strong diagnostic information about the nature of the deviation of the empirical distribution from the reference distribution.

If we apply it to our Microsoft data, we obtain the following result:

```
# QQ plots

# standardize microsoft return data to have zero mean and unit
# variance:

msft_std = (FinData$MSFT_lr - mean1)/sd1

# produce quantile-quantile scatter plot

qqnorm(msft_std,  main = "Normal Q-Q Plot",
       plot.it = TRUE, datax = TRUE)

# add a line that intersects the 25th and 75th quantile

qqline(msft_std, datax = FALSE, distribution = qnorm,
       probs = c(0.25, 0.75), qtype = 7)
```



This is typical of a situation where the empirical distribution has fatter tails than the normal distribution. For the same probability, the empirical distribution assigns more severe quantiles. This is because severe events are more likely in the empirical distribution than in the normal distribution.

Ruppert and Matteson (2015) give a simple illustration of what to conclude from any pattern you might observe in a qq plot.

We could conduct this kind of analysis for any reference distribution, and not just normal. Indeed, it is based on the closeness of the empirical and reference distribution that the "best distribution" is picked in empirical applications.

Also notice that you can observe that there are only few very extreme observations. Depending on your application, one might want to disregard these as "outliers"[2],

[2] One situation where this might be reasonable is when analyzing a data set captured by hand. Then a missing decimal may create a "zero information" mistake. In the type of data that the financial engineer will tend to use,

although one must have very strong reasons for doing so. The most sensible approach is that all well measured data is informative.

You can explore the approach called "Half-Normal plots" in Chapter 4 of Ruppert and Matteson (2015) as an additional outlier detection tool.

Next, we turn to joint distributions. The characteristics of a single asset is, alone, not of much use to a financial engineer. What a financial engineer cares about is a portfolio return which depends on the joint co-movements of two asset price returns. We will turn to this later.

this is extremely unlikely. Thus it is most reasonable to assume that all the data you have are correct and should form part of fitting the final model.

3 .2.8 Notes: Distributions

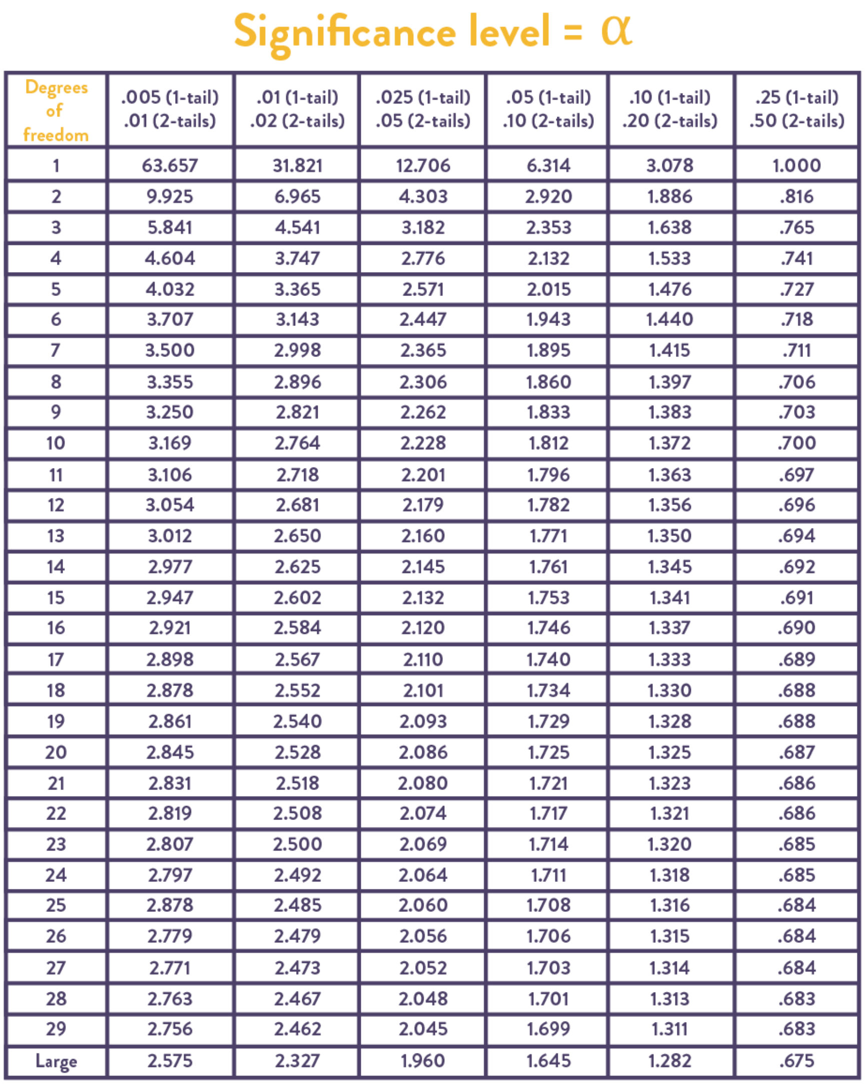
Normal distribution is a bell-shaped continuous probability density function (PDF) defined by parameters mean and standard deviation.

Log normal distribution, which is usually assumed to be the case for returns on many assets, is a continuous distribution in which the log – e.g. returns – of a variable has a normal distribution.

Uniform distribution is a PDF in which all possible events – e.g. returns – are equally likely to happen.

The following three distributions are especially important because they are used in hypothesis testing for several econometric procedures.

***t* distribution** is symmetrical, bell-shaped, and similar to the standard normal curve. The higher the degrees of freedom the closer the t will resemble the standard normal distribution with mean zero and standard deviation of one.



Significance level = α

Degrees of freedom	.005 (1-tail) .01 (2-tails)	.01 (1-tail) .02 (2-tails)	.025 (1-tail) .05 (2-tails)	.05 (1-tail) .10 (2-tails)	.10 (1-tail) .20 (2-tails)	.25 (1-tail) .50 (2-tails)
1	63.657	31.821	12.706	6.314	3.078	1.000
2	9.925	6.965	4.303	2.920	1.886	.816
3	5.841	4.541	3.182	2.353	1.638	.765
4	4.604	3.747	2.776	2.132	1.533	.741
5	4.032	3.365	2.571	2.015	1.476	.727
6	3.707	3.143	2.447	1.943	1.440	.718
7	3.500	2.998	2.365	1.895	1.415	.711
8	3.355	2.896	2.306	1.860	1.397	.706
9	3.250	2.821	2.262	1.833	1.383	.703
10	3.169	2.764	2.228	1.812	1.372	.700
11	3.106	2.718	2.201	1.796	1.363	.697
12	3.054	2.681	2.179	1.782	1.356	.696
13	3.012	2.650	2.160	1.771	1.350	.694
14	2.977	2.625	2.145	1.761	1.345	.692
15	2.947	2.602	2.132	1.753	1.341	.691
16	2.921	2.584	2.120	1.746	1.337	.690
17	2.898	2.567	2.110	1.740	1.333	.689
18	2.878	2.552	2.101	1.734	1.330	.688
19	2.861	2.540	2.093	1.729	1.328	.688
20	2.845	2.528	2.086	1.725	1.325	.687
21	2.831	2.518	2.080	1.721	1.323	.686
22	2.819	2.508	2.074	1.717	1.321	.686
23	2.807	2.500	2.069	1.714	1.320	.685
24	2.797	2.492	2.064	1.711	1.318	.685
25	2.878	2.485	2.060	1.708	1.316	.684
26	2.779	2.479	2.056	1.706	1.315	.684
27	2.771	2.473	2.052	1.703	1.314	.684
28	2.763	2.467	2.048	1.701	1.313	.683
29	2.756	2.462	2.045	1.699	1.311	.683
Large	2.575	2.327	1.960	1.645	1.282	.675

Chi-squared distribution is the distribution of the sum of squared standard normal deviates, where the degrees of freedom of the distribution is equal to the number of standard normal deviates being summed. The table is represented below.

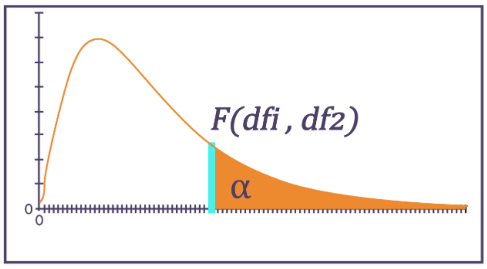
Percentage Points of the Chi-Square Distribution

Degrees of Freedom	Probability of a larger value of x²								
	0.99	0.95	0.90	0.75	0.50	0.25	0.10	0.05	0.01
1	0.000	0.004	0.016	0.102	0.455	1.32	2.71	3.84	6.63
2	0.020	0.103	0.211	0.575	1.386	2.77	4.61	5.99	9.21
3	0.115	0.352	0.584	1.212	2.366	4.11	6.25	7.81	11.34
4	0.297	0.711	1.064	1.923	3.357	5.39	7.78	9.49	13.28
5	0.554	1.145	1.610	2.675	4.351	6.63	9.24	11.07	15.09
6	0.872	1.635	2.204	3.455	5.348	7.84	10.64	12.59	16.81
7	1.239	2.167	2.833	4.255	6.346	9.04	12.02	14.07	18.48
8	1.647	2.733	3.490	5.071	7.344	10.22	13.36	15.51	20.09
9	2.088	3.325	4.168	5.899	8.343	11.39	14.68	16.92	21.67
10	2.558	3.940	4.865	6.737	9.342	12.55	15.99	18.31	23.21
11	3.053	4.575	5.578	7.584	10.341	13.70	17.28	19.68	24.72
12	3.571	5.226	6.304	8.438	11.340	14.85	18.55	21.03	26.22
13	4.107	5.892	7.042	9.299	12.340	15.98	19.81	22.36	27.69
14	4.660	6.571	7.790	10.165	13.339	17.12	21.06	23.68	29.14
15	5.229	7.261	8.547	11.037	14.339	18.25	22.31	25.00	30.58
16	5.812	7.962	9.312	11.912	15.338	19.37	23.54	26.30	32.00
17	6.408	8.672	10.085	12.792	16.338	20.49	24.77	27.59	33.41
18	7.015	9.390	10.865	13.675	17.338	21.60	25.99	28.87	34.80
19	7.633	10.117	11.651	14.562	18.338	22.72	27.20	30.14	36.19
20	8.260	10.851	12.443	15.452	19.337	23.83	28.41	31.41	37.57
22	9.542	12.338	14.041	17.240	21.337	26.04	30.81	33.92	40.29
24	10.856	13.848	15.659	19.037	23.337	28.24	33.20	36.42	42.98
26	12.198	15.379	17.292	20.843	25.336	30.43	35.56	38.89	45.64
28	13.565	16.928	18.939	22.657	27.336	32.62	37.92	41.34	48.28
30	14.953	18.493	20.599	24.478	29.336	34.80	40.26	43.77	50.89
40	22.164	26.509	29.051	33.660	39.335	45.62	51.80	55.76	63.69
50	27.707	34.764	37.689	42.942	49.335	56.33	63.17	67.50	76.15
60	37.485	43.188	46.459	52.294	59.335	66.98	74.40	79.08	88.38

F distribution

If U and V are independent chi-square random variables with r_1 and r_2 degrees of freedom respectively, then follow an F-distribution with r_1 numerator degrees of freedom and r_2 denominator degrees of freedom. A partial table is below:

F Table for α = 0.10

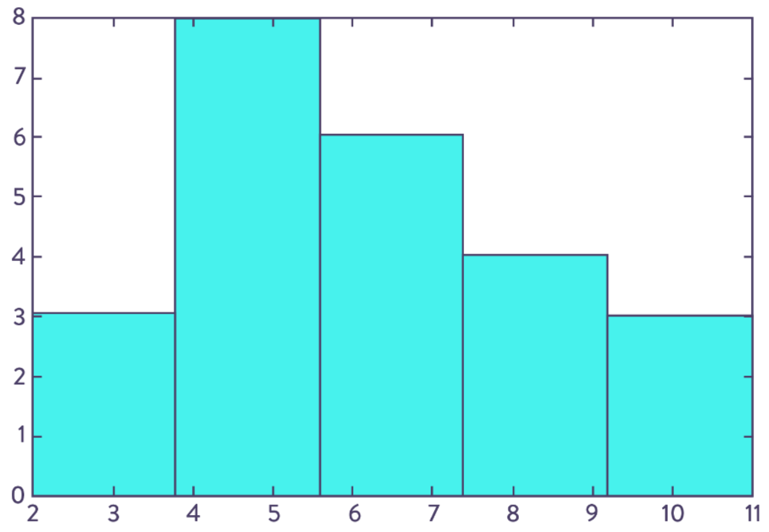


\	df_1=1	2	3	4	5	6	7	8	9
df_2=1	39.86346	49.50000	53.59324	55.83296	57.24008	58.20442	58.90595	59.43898	59.85759
2	8.52632	9.00000	9.16179	9.24342	9.29263	9.32553	9.34908	9.36677	9.38054
3	5.53832	5.46238	5.39077	5.34264	5.30916	5.28473	5.26619	5.25167	5.24000
4	4.54477	4.32456	4.19086	4.10725	4.05058	4.00975	3.97897	3.95494	3.93567
5	4.06042	3.77972	3.61948	3.52020	3.45298	3.40451	3.36790	3.33928	3.31628

Normality (graph, skewness)

Normality is an important aspect of many cases in econometrics. An important assumption often made in finance is that returns are normally distributed. It is also used in portfolio allocation models, pricing options, and Value-at-Risk (VaR) measurements. Frequently, analysts find that many real-world distributions have fat (heavy) tails and sometimes sharp peaks.

We want to explore skewness and symmetry. The example below is for the monthly simple returns of a med cap company. Do the returns normally distribute? This small size is for instructional purpose. Skewness is hard to judge in smaller samples.

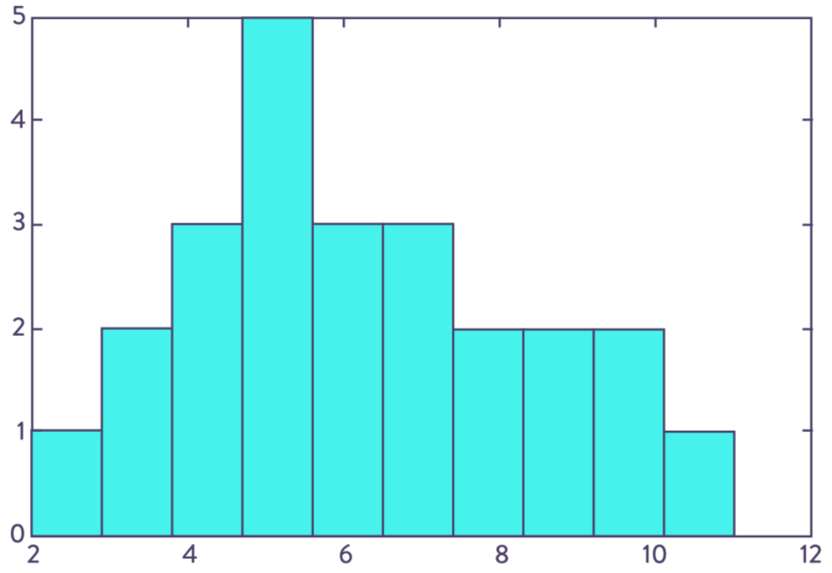


Skewness is an indicator used in distribution analysis as a sign of asymmetry and deviation from a normal distribution. The equation for skewness (called the third moment around the mean) is defined as:

$$\frac{n}{(n-1)(n-2)} \Sigma \left(\frac{x_i - \bar{x}}{s}\right)^3 .$$

Interpretation:

- Skewness > 0 – Right skewed distribution, most values are concentrated on left of the mean, with extreme values to the right.
- Skewness < 0 – Left skewed distribution, most values are concentrated on the right of the mean, with extreme values to the left.
- Skewness = 0 – mean = median, the distribution is symmetrical around the mean.



According to David P. Doane and Lori E. Seward (McGraw-Hill, 2011:155), Applied Statistics in Business and Economics, 3e: "If the sample size is 30 observations then 90% of the time the skewness stat will lie between -0.673 and 0.673 for normality to hold. If the sample size is 60 observations, then 90% of the time the skewness stat will lie between -0.496 and 0.496 for normality. If the sample size is 150 observations, then 90% of the time the skewness stat will lie between -0.322 and 0.322 to conclude normality". In the last example above, we reject normality.

Normality (kurtosis)

This is an indicator used as a sign of flatness or "peakedness" of a distribution. The formula for kurtosis is:

$$k = \frac{\frac{\sum_{i=n}^{n}\left(X_i - X_{avg}\right)^4}{n}}{\frac{\left(\sum_{i=n}^{n}\left(X_i - X_{avg}\right)^2\right)^2}{n^2}}$$

Interpretation:

Kurtosis > 3 – Leptokurtic distribution, sharper than a normal distribution, with values concentrated around the mean and thicker tails. This means high probability for extreme values.

- Kurtosis < 3 – Platykurtic distribution, flatter than a normal distribution with a wider peak. The probability for extreme values is less than for a normal distribution, and the values are wider spread around the mean.
- Kurtosis = 3– Mesokurtic distribution - normal distribution for example.

You should also examine a graph (histogram) of the data and consider performing other tests for normality, such as the Shapiro-Wilk test.

Periodic (or time consistent or continuously compounded) rate of return

Finance tends to use the "time-consistent" rate of return, also called the continuously compounded rate of return. This will be used in the econometric analysis for the duration of the course. The formula for the rate of return is

$$rt = ln(Pt/Pt-1),$$

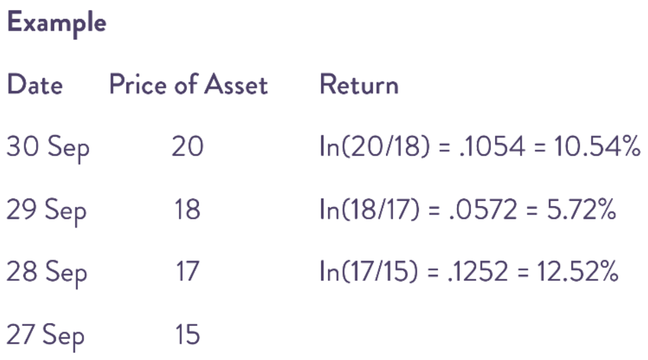
where:

P_t = Asset price at time t.

P_{t-1} = Asset price at time $t-1$

ln = Natural logarithm

r_t = Rate of return at time t.



Example

Date	Price of Asset	Return
30 Sep	20	ln(20/18) = .1054 = 10.54%
29 Sep	18	ln(18/17) = .0572 = 5.72%
28 Sep	17	ln(17/15) = .1252 = 12.52%
27 Sep	15	

These rates are often expressed as percentages, so the returns might be read as 10.54% for 30 Sep then 5.72% for 29 Sep and 12.52% for 28 Sep.

These returns are called time consistent because if the returns are summed, they equal the return from the start of the analysis to the end: Start at 27 Sep when the price was $15 and end at 30 Sep when the price was $20. The rate of return was

$$ln(20/15) = 28.77\%.$$

Instead, add all the returns from the table above= 10.54% + 5.72% + 12.52% = 28.77%

Exercise:

For the following prices go into Excel and find the returns as percentages.

Date	Price of Asset
16 Jan	40
15 Jan	36
14 Jan	38
13 Jan	35

Calculating simple volatility of returns for a series

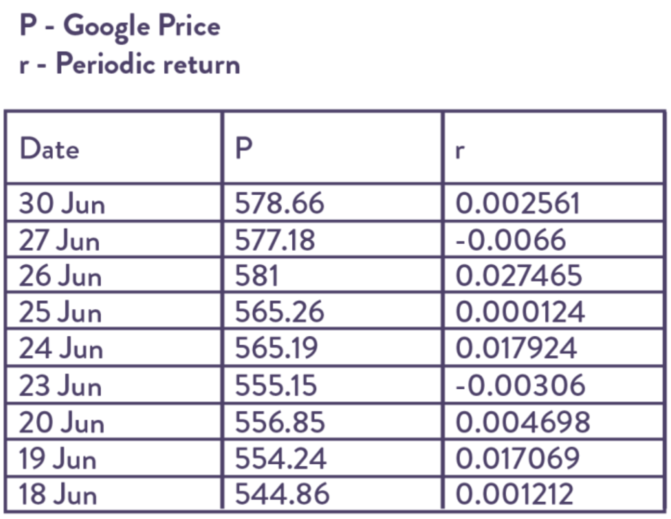
Given a series of prices, such as Google prices, it is possible to calculate the periodic return as above.

Step 1

P – Google Price

r – Periodic return

Where periodic return is $r_t = ln(Pt/Pt - 1)$.

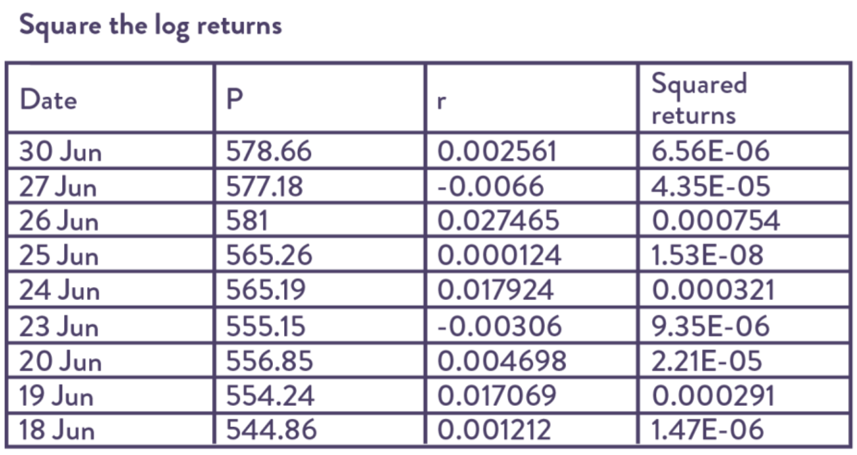


P - Google Price
r - Periodic return

Date	P	r
30 Jun	578.66	0.002561
27 Jun	577.18	-0.0066
26 Jun	581	0.027465
25 Jun	565.26	0.000124
24 Jun	565.19	0.017924
23 Jun	555.15	-0.00306
20 Jun	556.85	0.004698
19 Jun	554.24	0.017069
18 Jun	544.86	0.001212

Step 2

Square the log returns



Square the log returns

Date	P	r	Squared returns
30 Jun	578.66	0.002561	6.56E-06
27 Jun	577.18	-0.0066	4.35E-05
26 Jun	581	0.027465	0.000754
25 Jun	565.26	0.000124	1.53E-08
24 Jun	565.19	0.017924	0.000321
23 Jun	555.15	-0.00306	9.35E-06
20 Jun	556.85	0.004698	2.21E-05
19 Jun	554.24	0.017069	0.000291
18 Jun	544.86	0.001212	1.47E-06

Step 3

Use the following formula to get **simple volatility**

$$s = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N-1}}$$

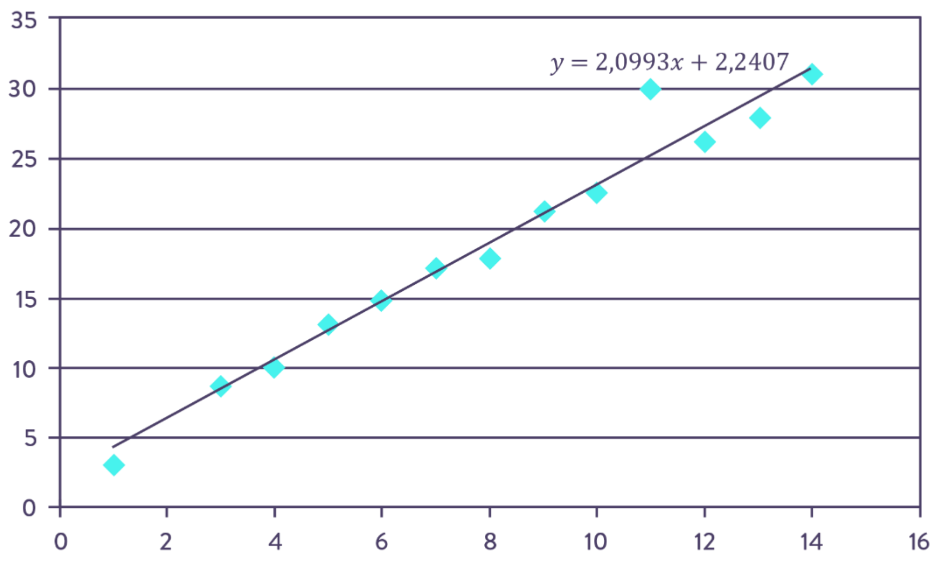
$$= 0.0126.$$

Note: This is just the average of the squared returns for the period under analysis.

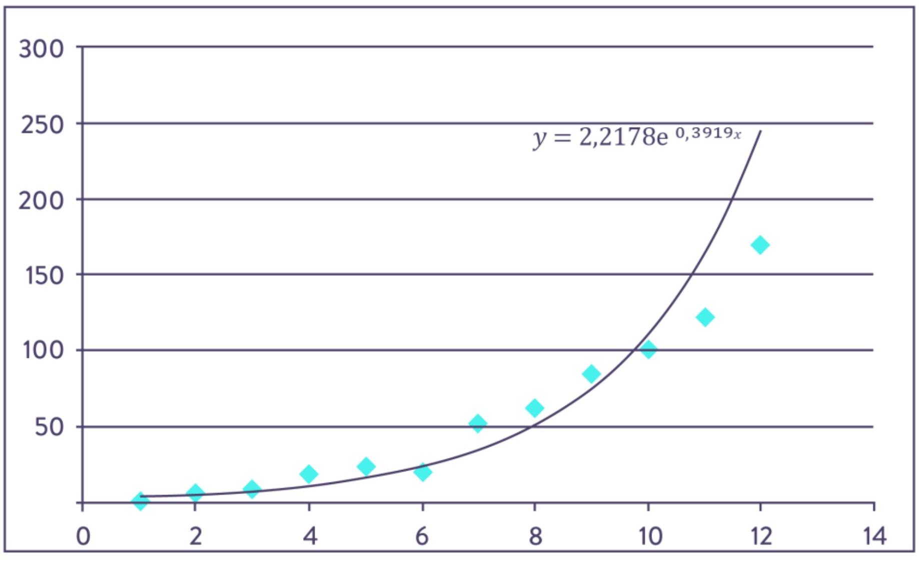
3.2.9 Notes: Types of Econometric Models

Plotting data to look for patterns to suggest functional form of regression

Linear relationship: The following plot of data (with a linear trend line/equation from Excel) suggests a linear model between x and y is appropriate.



In contrast, the following graph (again with a trend line) suggests a non-linear relationship between x and y (exponential relationship between x and y).



This can be transformed into a linear relationship as follows:

$$ln(y) = a + b(x).$$

Some other possible transformations are given in the table below.

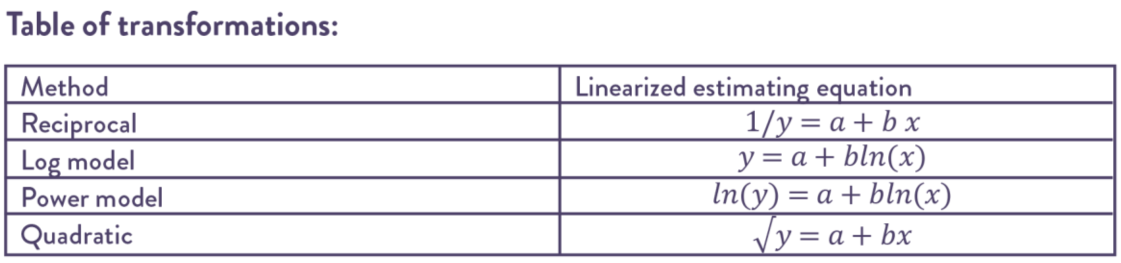


Table of transformations:

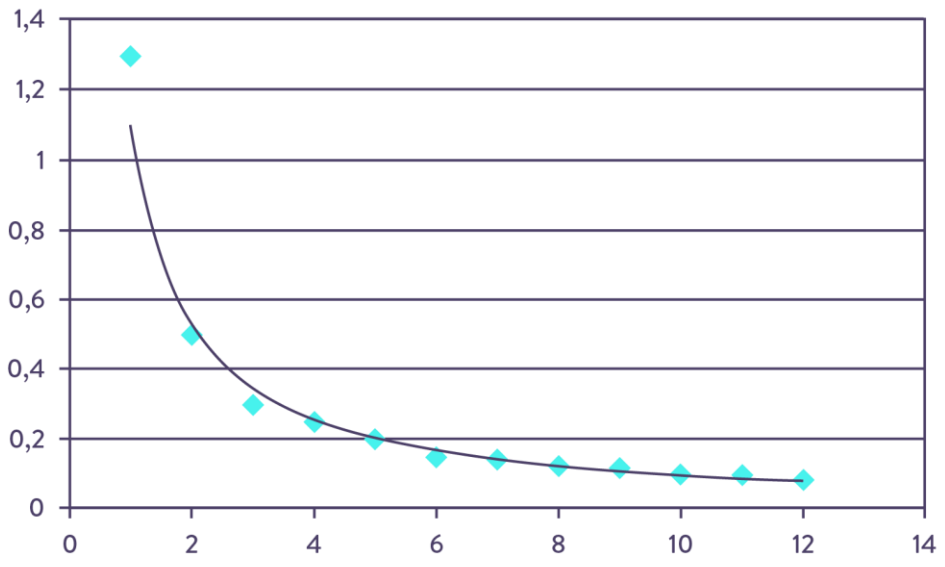
Method	Linearized estimating equation
Reciprocal	$1/y = a + b\,x$
Log model	$y = a + bln(x)$
Power model	$ln(y) = a + bln(x)$
Quadratic	$\sqrt{y} = a + bx$

Exercises:

1 Can the following equation be expressed linearly? If so how?

$$y = e^{\beta_0 + \beta_1 * X_1 + \beta_2 * X_2 + \varepsilon}$$

2 For the data below, what kind of functional relationship exists between x and y, and how this might be transformed using the transformations discussed in this section?



Interpretation of variables in equations

Dummy variables

A dummy variable (binary independent variable) assumes a value of 1 if some criterion is true, otherwise it will be zero. For example:

D=0 if respondent is male.

D=1 if respondent is female.

CEO earnings (in thousands of USD):

Y = 90 + 50x – 25 D [x is the number of years of experience]

If D=1 (female) the CEO can expect to make $25 000 less each year.

Categorical variables

Related to dummies are categorical variables. These noncontinuous variables can take on several values, some of which may be meaningful while others may not.

Variable C takes on several values.

CEO’s favorite color:

1 Yellow

2. Blue
3. Green
4. Red

Preferring yellow to blue does not demonstrate any intensity in the variable being measured. That is, the green is not 3 times as intense as yellow. Simply running this variable in a regression doesn't mean anything. This type of variable can be transformed into meaning by creating a dummy variable. For example:
D3 = 1 if yellow is favorite color
D3 = 0 otherwise.

The regression coefficient for D3 might be
CEO salary = ...+40 D3+.......
In this case a CEO whose favorite color is yellow earns $40 000 more per year than one whose favorite color is something else.

A categorical variable may express intensity and might be included in a regression as a categorical variable without resort to the use of a dummy. For example:
C is average household income

1. $0 to $20 000
2. $20 001 to $40 000
3. $40 001 to $60 000
4. $60 001 to $80 000
5. More than $80 000

In this case, intensity is represented by the data. The data represent a proxy for an unobservable variable: continuous income. For example, in a regression of:

Double log equation

$$ln(y) = a + bln(x),$$

b is the percentage change in y resulting from a one percent change in x. For example, if $b = 5$ then a one percent increase in x results in a 5% increase in y.

Exercises:

1 The following equation evaluating business majors' starting salary contains two dummy variables D1 and D2.

D1=1 if college major was accounting, zero otherwise.
D2=1 if college major was finance, otherwise zero.

Salary = 34 + 3.76D1 + 8.93 D2 [starting salary of business majors]

a. What is the starting salary of a business grad majoring in accounting?
b. What is the starting salary of a business grad majoring in finance?
c. What is the starting salary of other business majors?

Finance models

We are largely considering linear relationships in this course. As a consequence, we wish to have an independent variable, x, made a linear function.

Linearizing models

1 It may be you are trying to model a nonlinear relationship between y and independent variable x:

$Y = aX^b$ (where a and b are constant parameters)
This can be linearized by taking the natural log of both sides:
$In\ (Y) = ln(a) + bln(X)$ [log-log relationship]

This type of model can be estimated using linear regression, which we shall talk about later. It is important to note that if we change x, the change in y is as follows:

$dY = b(Y/X)\,dX$ [it depends on the values chosen for X and Y – e.g. mean values]

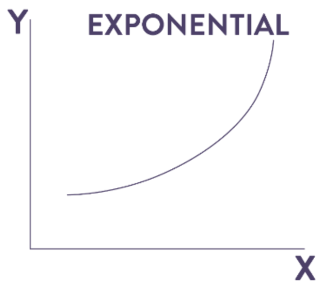
Suppose instead the relationship between x and y is

$$Y = e^{a+bx}.$$

This nonlinear relationship can be expressed as:

$$In\,Y = In\,[e^{a+bx}] = a + bX.$$

Bear in mind the relationship is not linear in X and Y but instead is illustrated by the graph below:



2 Another semi-log relationship is:

$$Y = a + b\,In\,(X).$$

3 Suppose that X is the independent variable and Y the dependent variable, but a strict linear relationship does not exist. Instead:

$$Y = a + bX + c\,X^2\ [polynomial]$$

This can still be estimated using the linear regression techniques of this course, but we treat X^2 as a new variable $Z = X^2$.

Exercise:

Transform the following equation into a linear relation:

$$Y = [a + bX]^{-1}.$$

Hint: What is the definition for the new dependent variable?

3.2.10 Notes: Finance Topics and Econometrics

Financial econometrics looks at functional relationships between variables in the financial environment, such as returns on financial assets or trends in variables such as bonds yields. Consequently, we estimate equations that are linear or nonlinear in nature, using the techniques developed in this course.

Capital asset pricing model (CAPM)

CAPM shows a linear relationship between risk and expected return and that is used in the pricing of risky securities. The security market line of the CAPM is defined as

$$SML: R_{it} - R_{ft} = a_i + \beta_i \left(R_{Mt} - R_{ft}\right) + \varepsilon_{it}.$$

The alpha coefficient (α_i) is the constant (intercept) in the security market line of the CAPM.

The alpha coefficient indicates how an investment has performed after accounting for the risk it involved.

If markets are efficient then $\alpha = 0$. If $\alpha < 0$ the security has earned too little for its risk and if $\alpha > 0$ the investment has a risk in excess of the return for the given risk.

Fama-French model (FFM)

The FFM is a factor model that expands on the capital asset pricing model (CAPM) by adding size and value factors along with the market risk factor in CAPM. They maintain that value and small cap stocks outperform markets on a consistent basis.

Fama and French started with the observation that two classes of stocks have tended to do better than the market as a whole: (i) small caps and (ii) stocks with a high book-value-to-price ratio (customarily called "value" stocks; their opposites are called "growth" stocks), where book value equals assets minus liabilities and preferred shares.

They then added two factors to CAPM to reflect a portfolio's exposure to these two classes:

$$r - R_f = beta_3 \times (K_m - R_f) + b_3 \times SMB + b_y \times HML + alpha,$$

where:
r is the portfolio's return rate,
R_f is the risk-free return rate (US T bills),
SMB (Small Minus Big) is the average return on the three small portfolios minus the average return on the three big portfolios:

$$SMB = 1/3\,(Small\,Value + Small\,Neutral + Small\,Growth) - 1/3\,(Big\,Value + Big\,Neutral + Big\,Growth),$$

HML (High Minus Low) is the average return on the two value portfolios minus the average return on the two growth portfolios:

$$HML = \frac{1}{2}(Small\,Value + Big\,Value) - \frac{1}{2}(Small\,Growth + Big\,Growth),$$

K_m is the return of the whole stock market.

Momentum is sometimes used as another factor.

The daily and weekly Fama-French factors can be found at the following:
http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html

Portfolio analysis

Like CAPM, the Fama-French model is used to explain the performance of portfolios via linear regression, only now the two extra factors give you two additional axes so, instead of a simple line, the regression is a big flat thing that lives in the fourth dimension.

Even though you can't visualize this regression, you can still solve for its coefficients in a spreadsheet. The result is typically a better fit to the data points than you get with CAPM, with an r-squared in the mid-ninety percent range instead of the mid-eighties.

Investing for the future

Analyzing the past is a job for academics; most people are more interested in investing intelligently for the future. Here the approach is to use software tools and/or professional advice to find the exposure to the three factors that's appropriate for you, and then to invest in special index funds that are designed to deliver that level of the factors. You can try this tool on another website. When you try it, you should note that it in fact collapses all risk to the single factor of volatility, which is typical, and which brings up the earlier question of whether the additional factors really are measures of risk. In this case, how would you ever help somebody figure out what their "value tolerance" was? As a matter of fact, what would that question even *mean*?

Arbitrage pricing theory (APT)

ABT states that the return of a financial asset is a linear function of a number of factors such as macroeconomic factors (e.g., the market rate of interest, economic growth, inflation) and market indices such as:

- short term interest rates;
- the difference in long-term and short-term interest rates;
- a diversified stock index, such as the S&P 500 or NYSE Composite;
- oil prices;
- gold and other precious metal prices; and
- currency exchange rates.

The APT produces a rate of return that can be checked against actual return to see if the asset is correctly priced.

Assumptions:

- Security returns are a function of a number of factors.
- There are sufficient securities for firms-specific (idiosyncratic) risk to be diversified away.
- Well-functioning security markets do not allow for persistent arbitrage opportunities.
- Risky asset returns are said to follow a *factor intensity structure* if they can be expressed as:

$$r_j = a_i + b_{i1}\, F_1 + b_{i2}\, F_2 + \cdots + b_{in}\, F_n + \varepsilon_i,$$

where:

a_i is a constant for asset i,

F_i is a systematic factor where n such factors exist in this model,

B is called the *factor loading* or the change in return resulting from a unit change in the factor,

ε_i is called the idiosyncratic random shock with zero mean.
Idiosyncratic shocks are assumed to be uncorrelated across assets and uncorrelated with the factors.

The APT states that if asset returns follow a factor structure then the following relation exists between expected returns and the factor sensitivities:

$$E\left(r_j\right) = r_f + b_{j1}RP_1 + b_{j2}RP_2 + \cdots + b_{jn}RP_n$$

where:
RP_k is the risk premium of the factor,
r_f is the risk-free rate.

Yield curve (term structure of interest rates)

Liquidity Preference Theory suggests market participants demand a premium for long-term investments in bonds (such as Treasuries, gilts, or French OATs) compared to short term loans to the government. Even in a situation where no interest rate change is expected, the curve will slightly slope upwards because of the risk/liquidity premium investors demand from buying long term. This is the usual case (normal interest curve), but inverse or inverted curves are possible.

The logic behind the econometrics associated with the yield curve can be understood by looking at investor expectations about what future yields will be.

The yield on a n-period bond is equal to the average of expectations for short-term bonds plus a risk/liquidity premium, P or

$$r_{n,t} = P + \frac{\left[r_{1,t+1} + E_t\left(r_{1,t+2}\right) + \cdots + E_t\left(r_{1,t+n-1}\right)\right]}{n}.$$

The expected one-year returns are all at time t. Assume the investor bases his/her expectation on the current one-year bond, $r_{1,t}$.The estimating equation for the n −year bond (long-term) is $r_{n,t} = a + b \quad r_{1,t}$ [estimating equation for yield curve].

Note that the theory (if it panned out) suggests that $b = 1$. This can be tested using the t-test from OLS.

Early warning systems

It is known that financial imbalances commonly lead to widespread financial stress which may cause the collapse of banks or other companies, financial crises, and recessions.

It is therefore of crucial importance to detect and to predict potential financial stress. There are a host of early warning systems regarding financial or credit crises. One is the MIMIC that consists of two sets of equations:

$$Y = bW + v,$$

and

$$W = cX + u,$$

where Y is a crisis indicator, X is an observable potential crisis cause, and W is a latent variable thought to represent the severity of the crisis. Disturbance terms u and v are explained later. Macroeconomic variables that can be used as crisis indicators include:

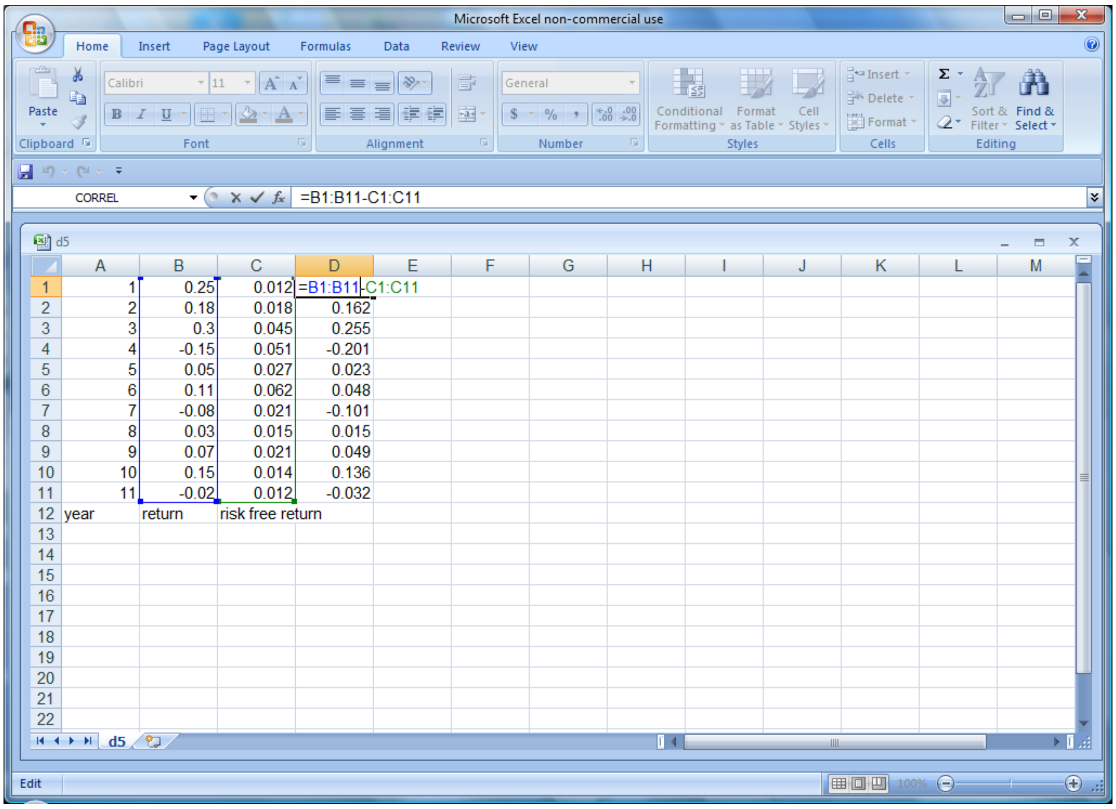
- Credit growth
- Property price growth
- Credit to GDP gap
- Equity price growth

Sharpe ratio

The Sharpe ratio is a common benchmark that describes how well an investment uses risk to get return. Given several investment choices, the Sharpe ratio can be used to quickly decide which one is a better use of your money. When comparing two assets, the one with a higher Sharpe ratio provides a superior return for the same risk, σ. Suppose you have invested in a mutual fund and want to find the Sharpe ratio for the past 11 years. You calculate the year-over-year returns from your investment as follows:

Year	Return	Risk-free return
1	0.25	0.012
2	0.18	0.018
3	0.3	0.045
4	-0.15	0.051
5	0.05	0.027
6	0.11	0.062
7	-0.08	0.021
8	0.03	0.015
9	0.07	0.021
10	0.15	0.014
11	-0.02	0.012

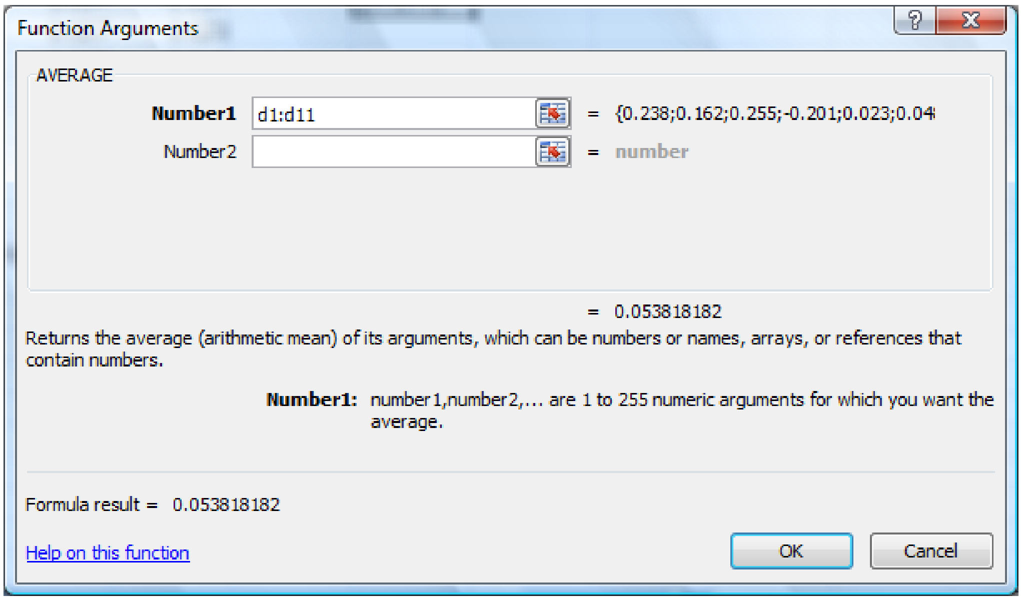
Subtract the risk-free return from the actual return by using the formula command in Excel. This is your Excess Return from d1:d11 in the chart below:



CORREL =B1:B11-C1:C11

	A	B	C	D
1	1	0.25	0.012	=B1:B11-C1:C11
2	2	0.18	0.018	0.162
3	3	0.3	0.045	0.255
4	4	-0.15	0.051	-0.201
5	5	0.05	0.027	0.023
6	6	0.11	0.062	0.048
7	7	-0.08	0.021	-0.101
8	8	0.03	0.015	0.015
9	9	0.07	0.021	0.049
10	10	0.15	0.014	0.136
11	11	-0.02	0.012	-0.032
12	year	return	risk free return	

Then use the AVERAGE command in statistical formulas of Excel to get the average excess return for the period. In this case 0.0538.



Function Arguments

AVERAGE

Number1 d1:d11 = {0.238;0.162;0.255;-0.201;0.023;0.04

Number2 = number

= 0.053818182

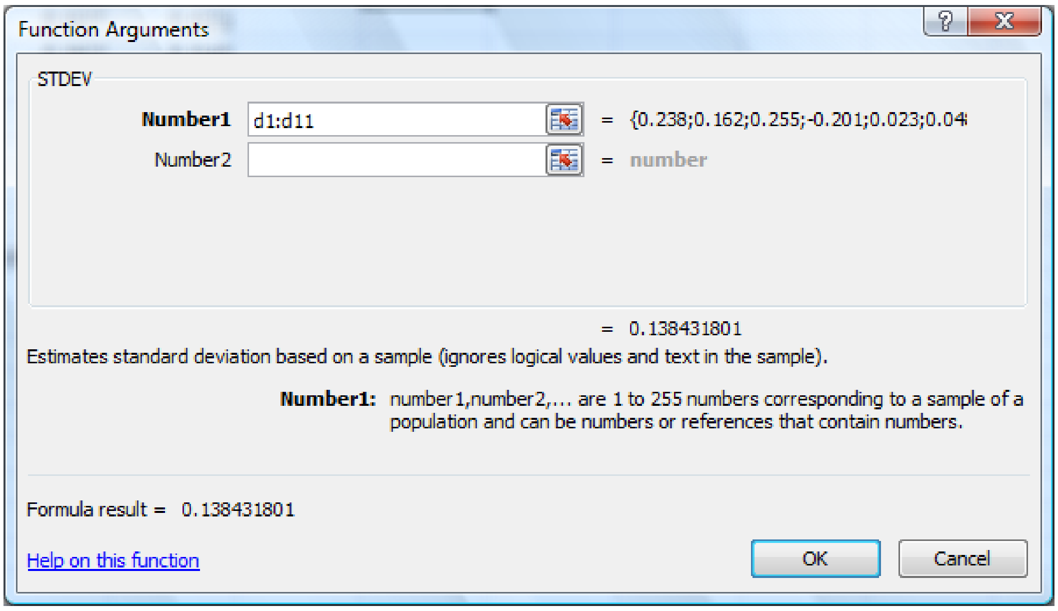
Returns the average (arithmetic mean) of its arguments, which can be numbers or names, arrays, or references that contain numbers.

Number1: number1,number2,... are 1 to 255 numeric arguments for which you want the average.

Formula result = 0.053818182

Help on this function

Next use the standard deviation command for the excess returns:


Function Arguments

STDEV

Number1 d1:d11 = {0.238;0.162;0.255;-0.201;0.023;0.04

Number2 = number

= 0.138431801

Estimates standard deviation based on a sample (ignores logical values and text in the sample).

Number1: number1,number2,... are 1 to 255 numbers corresponding to a sample of a population and can be numbers or references that contain numbers.

Formula result = 0.138431801

The Sharpe ratio = .0538 / .1384 = .3887.

Stock valuation (present value model)

The price, P, of a share of company stock at time t is equal to the expected discounted sum of all future dividends (cash payments):

$$P_t = E_t \{\frac{D_{t+1}}{(1+\ \delta)} + \frac{D_{t+2}}{(1+\ \delta)^2} + \ \ldots..\},$$

where:

D = dividend expected at some time in the future

δ = the constant discount factor.

If D is assumed to be a constant, the equation reduces to a perpetuity and the equation takes the form of:

$$P_t = \frac{D_t}{\delta}.$$

In log form this looks as follows:

$$In(P_t) = -In(\delta) + In(D_t),$$

which can be estimated by OLS with the following specification:

$$In\ (P_t = \beta_0 + \beta_1\, In(D_t).$$

OLS topic will be presented in detail in Module 2.

Parametric (analytical) Value-at-Risk

Value-at-Risk (VaR) as the name suggests is measure of risk of loss for a particular asset or portfolio. The probability level is one minus the probability of a VaR break (extremely risky scenario). For example if the confidence level is the extreme left-most 1%, the specified confidence level is 100%-1% or 99% confidence level.

Assumptions:

- The returns from the asset/portfolio are normally distributed. This allows the use of the two params from the normal mean and standard deviation.
- We choose a level of confidence. If the confidence is 95% sure the worst case is not going to happen, choose z-value = -1.645. If the confidence is 99% sure the worst case is not going to happen choose z-value = -2.33. The z-value represents the number of standard deviations away from the mean.
- The returns are assumed to be serially independent so no prior return should influence the current return.

Steps required to get VaR

1. Find (log) returns from financial data (keep things as percentages)
2. Calculate mean (percentage)
3. Calculate standard deviation of means (percentage)
4. Choose a confidence level (we assume z=1.645)
5. Calculate the dollar loss associated with your investment as a money manager. Assume you invested $10m in a security.

Example:

Choose a 95% confidence level, meaning we wish to have 5% of the observations in the left-hand tail of the normal distribution. That means that the observations in that area are 1.645 standard deviations away from the mean (assume to be zero for short period-of-time return data). The following are data for a security investment:

Amount of dollars invested: $10 million

Standard deviation: 1.99% or .0199

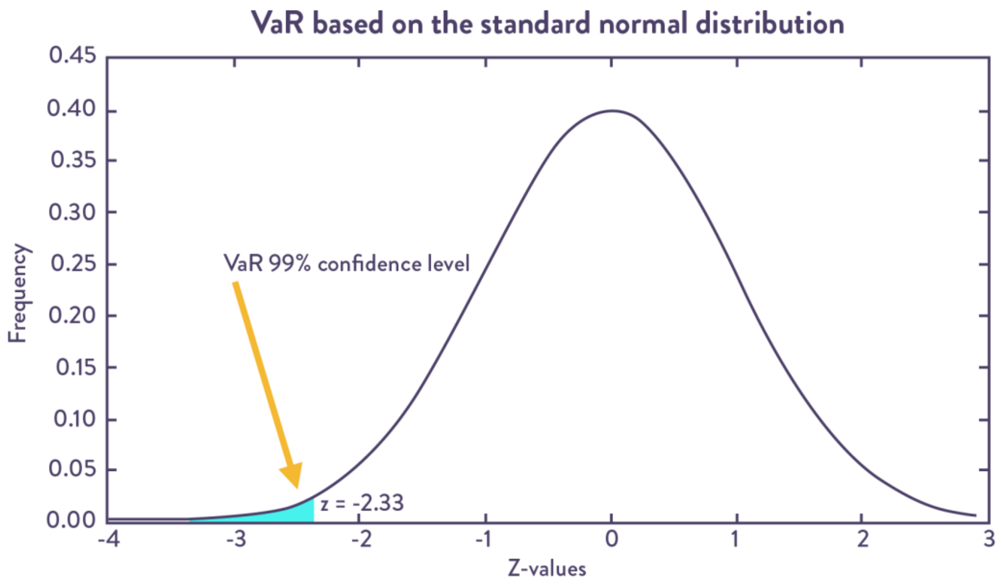
The VaR at the 95% confidence level is $1.645\ x\ 0.0199$ or 0.032736. The portfolio has a market value of £10 million, so the VaR of the portfolio is 0.032736 x 10 000 000 = $327 360.

$z = 2.33$ for a 99 % confidence level considering the normal distribution.

$z = 1.64$ for a 95 % confidence level considering the normal distribution.

Relationship of correlation and VaR

Covariance VaR changes linearly as the underlying model or factor variable instantaneous correlations change. Likewise, non-linear VaR will change non-linearly as the instantaneous correlations change. Dynamic hedging costs should properly reflect the VaR cost of cap.



Graph from "Python for Finance" by Yuxing Yan.

References

Berlinger, E. et al. Mastering R for Quantitative Finance. Packt Publishing.

Chan, E.P. Quantitative Trading. Wiley Trading.

Doan, D.P. and Seward, L. S. (2011) Applied Statistics in Business and Economics, 3rd edition McGraw-Hill p. 155.

Daroczi G. et al. Introduction to R for Quantitative Finance. Packt Publishing.

Greene, W. (2000). Econometric Analysis, Prentice-Hall, NY.

Gujarati, D. (2004). Basic Econometrics, McGraw-Hill.

Halls-Moore, M. L. (2017). Successful Algorithmic Trading.

Halls-Moore, M.L. (2017). Advanced Algorithmic Trading. Part III Time Series Analysis.

Jeet P. and Vats P. Learning Quantitative Finance with R. Packt Publishing.

McNeil, A. J. et al. Quantitative Risk Management. Princeton University Press.

Ojeda et al. Practical Data Science Cookbook, Packt Publishing.

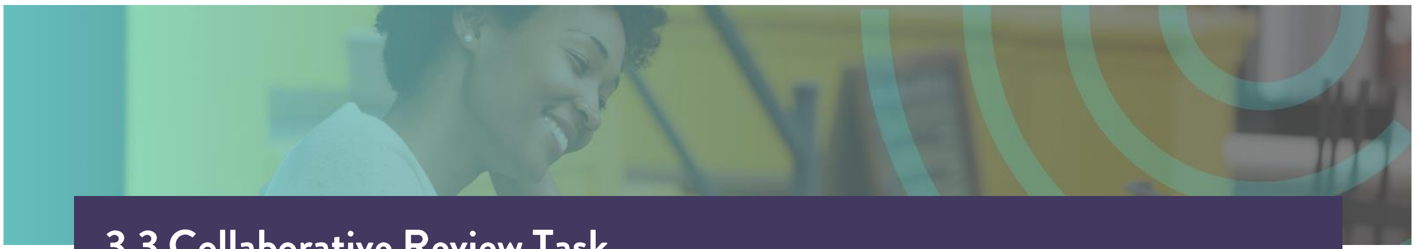
Scott, M. et al. (2013). Financial Risk Modelling and Portfolio Optimization with R. Wiley.

Yuxing, Y. (2017). Python for Finance. Packt Publishing.

Cont, R. (2001) 'Empirical properties of asset returns: Stylized facts and statistical issues', *Quantitative Finance*, 1(2), pp. 223–236. doi: 10.1080/713665670.

Ruppert, D. and Matteson, D. S. (2015) *Statistics and Data Analysis for Financial Engineering, with R examples*. Second. Ithaca, NY, USA: Springer Texts in Statistics.

Tsay, R. S. (2010) 'Analysis of Financial Time Series'. Wiley.



3.3 Collaborative Review Task

In this module, you are required to complete a collaborative review task, which is designed to test your ability to apply and analyze the knowledge you have learned during the week.

Question

CAPM and APT

Capital Asset Pricing Model (CAPM) and Arbitrage Pricing Theory (APT) are used on a large scale for asset pricing.

1. Describe the differences between CAPM and APT.
2. What model would you choose between CAPM and APT? Indicate research papers on this topic. Provide arguments to support your choice.

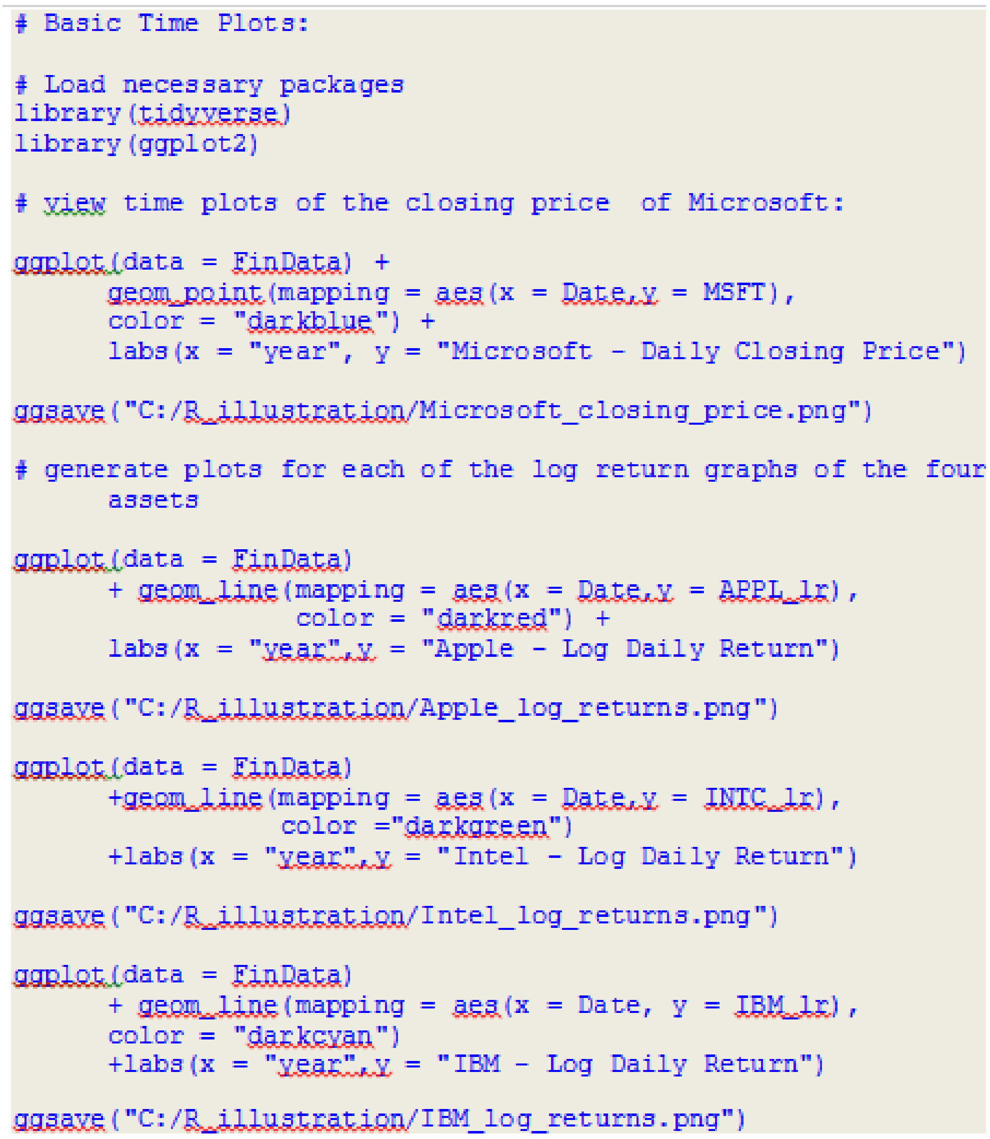
Appendix

Module 1 R Code

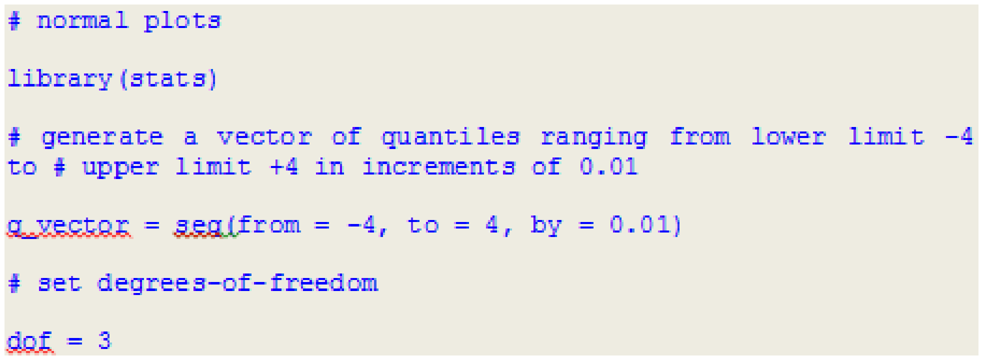
```
install.packages("tidyverse") # for all modules
install.packages("tseries") # for module 3
install.packages("forecast") # for most modules
install.packages("fGarch") # for module 4
install.packages("vars") # for module 5
install.packages("evir") # for module 6
install.packages("copula") # for module 7
```

If you want to use a function or method in a package, you must place it in the current "global environment" so that R code is able to call the functions in the package. To do this, use the command:

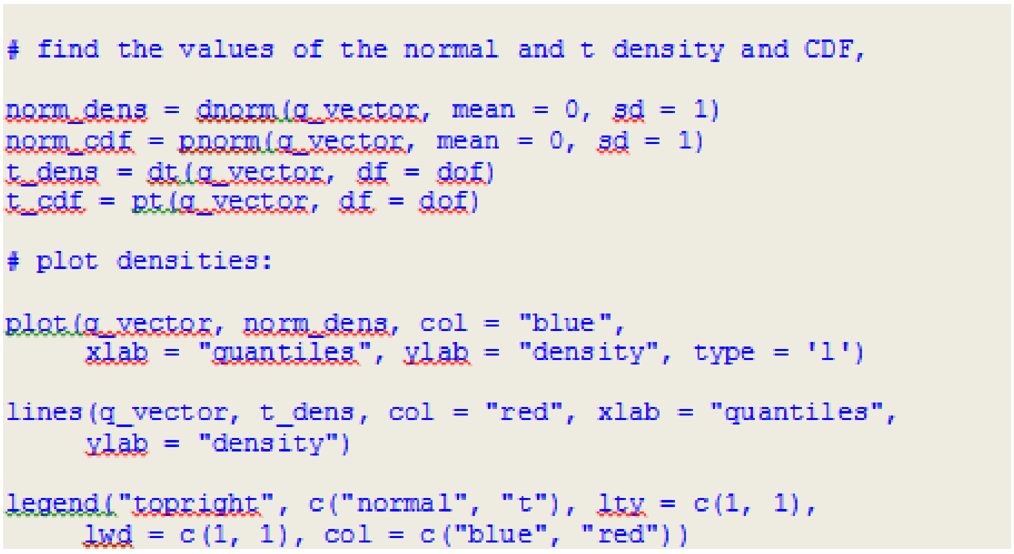
```
library(tidyverse) # loads the tidyverse package
```



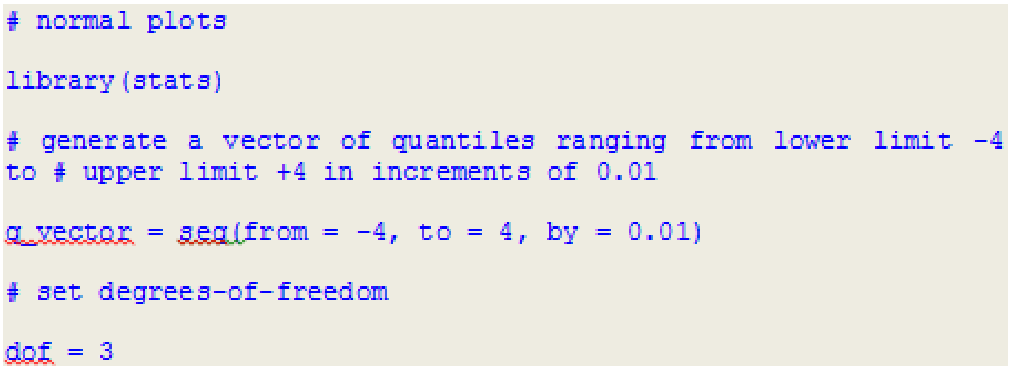
```
# Basic Time Plots:

# Load necessary packages
library(tidyverse)
library(ggplot2)

# view time plots of the closing price  of Microsoft:

ggplot(data = FinData) +
      geom_point(mapping = aes(x = Date,y = MSFT),
      color = "darkblue") +
      labs(x = "year", y = "Microsoft - Daily Closing Price")

ggsave("C:/R_illustration/Microsoft_closing_price.png")

# generate plots for each of the log return graphs of the four
      assets

ggplot(data = FinData)
      + geom_line(mapping = aes(x = Date,y = APPL_lr),
                  color = "darkred") +
      labs(x = "year",y = "Apple - Log Daily Return")

ggsave("C:/R_illustration/Apple_log_returns.png")

ggplot(data = FinData)
      +geom_line(mapping = aes(x = Date,y = INTC_lr),
                 color ="darkgreen")
      +labs(x = "year",y = "Intel - Log Daily Return")

ggsave("C:/R_illustration/Intel_log_returns.png")

ggplot(data = FinData)
      + geom_line(mapping = aes(x = Date, y = IBM_lr),
      color = "darkcyan")
      +labs(x = "year",y = "IBM - Log Daily Return")

ggsave("C:/R_illustration/IBM_log_returns.png")
```



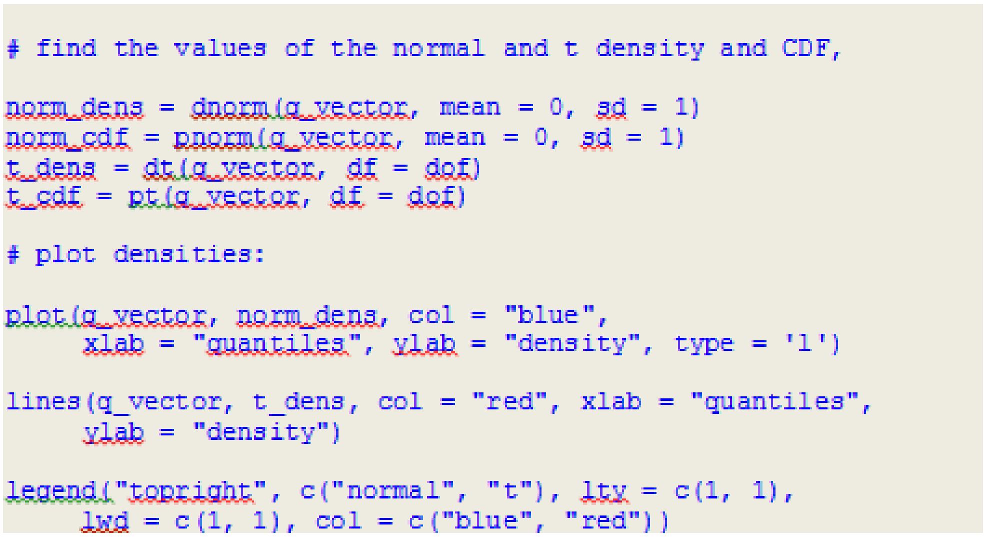
```
# normal plots

library(stats)

# generate a vector of quantiles ranging from lower limit -4
to # upper limit +4 in increments of 0.01

q_vector = seq(from = -4, to = 4, by = 0.01)

# set degrees-of-freedom

dof = 3
```



```
# find the values of the normal and t density and CDF,

norm_dens = dnorm(q_vector, mean = 0, sd = 1)
norm_cdf = pnorm(q_vector, mean = 0, sd = 1)
t_dens = dt(q_vector, df = dof)
t_cdf = pt(q_vector, df = dof)

# plot densities:

plot(q_vector, norm_dens, col = "blue",
     xlab = "quantiles", ylab = "density", type = 'l')

lines(q_vector, t_dens, col = "red", xlab = "quantiles",
     ylab = "density")

legend("topright", c("normal", "t"), lty = c(1, 1),
     lwd = c(1, 1), col = c("blue", "red"))
```



```
# normal plots

library(stats)

# generate a vector of quantiles ranging from lower limit -4
to # upper limit +4 in increments of 0.01

q_vector = seq(from = -4, to = 4, by = 0.01)

# set degrees-of-freedom

dof = 3
```



```
# find the values of the normal and t density and CDF,

norm_dens = dnorm(q_vector, mean = 0, sd = 1)
norm_cdf = pnorm(q_vector, mean = 0, sd = 1)
t_dens = dt(q_vector, df = dof)
t_cdf = pt(q_vector, df = dof)

# plot densities:

plot(q_vector, norm_dens, col = "blue",
     xlab = "quantiles", ylab = "density", type = 'l')

lines(q_vector, t_dens, col = "red", xlab = "quantiles",
     ylab = "density")

legend("topright", c("normal", "t"), lty = c(1, 1),
     lwd = c(1, 1), col = c("blue", "red"))
```

```
# plot distributions:

plot(q_vector, norm_cdf, col = "blue", xlab = "quantiles",
     ylab = "probability", type = 'l')

lines(q_vector,t_cdf, col = "red")

legend("bottomright", c("normal", "t"), lty = c(1, 1),
       lwd = c(1, 1), col = c("blue", "red"))
```

```
# Application 1 - Microsoft
library(kdensity)

# generate histogram with 128 breaks

hist(FinData$MSFT_lr, breaks = 64,
     freq = FALSE, main = "Microsoft",
     xlab = 'daily log returns')

# fit a kernal density estimate starting the search at a
normal distribution

kde_MSFT = kdensity(FinData$MSFT_lr, start = "normal")

# plot the kernel density estimate

lines(kde_MSFT, col = "blue")

# plot the closest normal distribution

lines(kde_MSFT, plot_start = TRUE, col = "red")
```

```
# fit empirical CDF to microsoft data

ecdf_msft = ecdf(FinData$MSFT_lr)

# plot the step function

plot(ecdf_msft, verticals = TRUE, do.p = FALSE, main = "EDF
and Normal CDF")

# compute some of the statistics of the empirical distribution

mean1 = mean(FinData$MSFT_lr)
sd1 = sd(FinData$MSFT_lr)
min1 = min(FinData$MSFT_lr)
max1 = max(FinData$MSFT_lr)

#  generate  the  closest  equivalent  CDF  for  the  normal
distribution
# i.e. with the mean and standard deviation of the empirical
distribution

q_vector = seq(from = min1, to = max1, by = 0.005)
norm_cdf = pnorm(q_vector, mean = mean1, sd = sd1)

# add the plot of the normal CDF

lines(q_vector,norm_cdf,col="red")

# add a legend

legend("bottomright",   c("empirical  distribution",   "normal
distribution"), lty = c(1, 1),
       lwd = c(1, 1), col = c("black", "red"))
```

```
# QQ plots

# standardize microsoft return data to have zero mean and unit
# variance:

msft_std = (FinData$MSFT_lr - mean1)/sd1

# produce quantile-quantile scatter plot

qqnorm(msft_std,  main = "Normal Q-Q Plot",
       plot.it = TRUE, datax = TRUE)

# add a line that intersects the 25th and 75th quantile

qqline(msft_std, datax = FALSE, distribution = qnorm,
       probs = c(0.25, 0.75), qtype = 7)
```

```
# scatterplot of Microsoft vs Intel log daily returns:
plot(FinData$MSFT_lr,FinData$INTC_lr, xlab = "Microsoft",
     ylab = "Intel")
```