Marketing Analytics

Wei Miao

2021-09-01

Contents

1 Preface **5**

2 Syllabus: MSIN0094 Marketing Analytics **7**

2.1 Instructor Information 7

2.2 Teaching Assistants 7

2.3 Course Objective 7

2.4 Module Structure Overview 8

2.5 Office Hours 8

2.6 Programming Language 9

2.7 Classroom Etiquette 9

2.8 Assignments and Grading policy 10

2.9 How to Make the Best Use of Case Studies 11

2.10 Copyright Issues 11

3 Detailed Class Descriptions **13**

3.1 Week 1 (Thu, 07 Oct) 13

3.2 Week 2 (Thu, 14 Oct) 14

3.3 Week 3 (Thu, 21 Oct) 15

3.4 Week 4 (Thu, 28 Oct) 16

3.5 Week 5 (Thu, 04 Nov) 17

3.6 Week 6 (Thu, 11 Nov) 18

3.7 Week 7 (Thu, 18 Nov) 19

3.8 Week 8 (Thu, 25 Nov) 20

3.9 Week 9 (Thu, 02 Dec) 20

3.10 Week 10 (Thu, 09 Dec) 22

4 R Tutorials **23**

4.1 Introduction to R History 23

4.2 R Basics 28

4.3 Programming Basics 45

5 Lecture Notes **49**

Chapter 1

Preface

This is the online supplement to the MSIN0094 Marketing Analytics Module at UCL School of Management.

Chapter 2

Syllabus: MSIN0094 Marketing Analytics

(Last update on 01 September, 2021)

2.1 Instructor Information

Module Leader: Dr. Wei Miao[1] (wei.miao@ucl.ac.uk)

Office: S3, Level 38, One Canada Square

Office hour:

- Appointment link
- Please refer to the office hour session for how to make office hour appointments. More details to be confirmed in Week 1's class.

2.2 Teaching Assistants

- Christopher Ogbunuzor (christopher.ogbunuzor.14@ucl.ac.uk)
- Henry Bellhouse (henry.bellhouse.18@ucl.ac.uk)

2.3 Course Objective

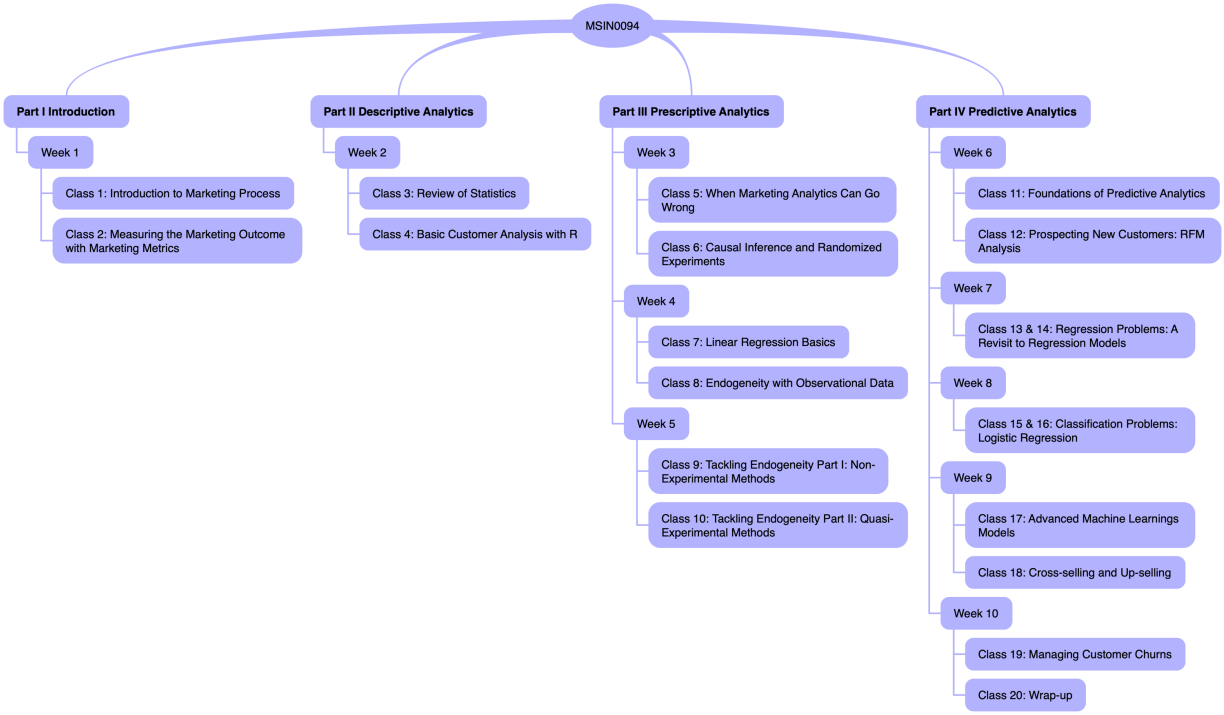
Marketing analytics addresses how to utilize analytics tools to better carry out marketing tactics and strategies. The digital age has fundamentally altered the way we collect, process, analyze, and disseminate data intelligence. Driven by advances in computing power, software, and other information technologies, the very nature of marketing analytics has evolved. In this module,

[1]Wei is pronounced as "way", which stands for jade in Mandarin; Miao is my family name, which is exactly pronounced as "meow", the sound of a kitty.

students will learn the scientific approach to real-world marketing problems with hands-on use of a combination of econometrics, statistics, machine learning, and programming, to collect, analyze, and act on customer intelligence.

Throughout the module, student will learn how to guide a company's marketing decisions along three dimensions: descriptive analytics, prescriptive analytics, and predictive analytics. Specifically, descriptive analytics involves using visualization tools and statistical tests to provide model-free insights into the customer intelligence; prescriptive analytics focuses on drawing causal inference via experimental, quasi-experimental, and non-experimental methods; predictive analytics shed light on how to prospect new customers, up-sell and cross-sell high value products, and mitigate customer churns, using appropriate machine learning models. At the end of the module, students will have the full capacity to carry out independent marketing research for their dissertation projects as well as lead a marketing analytics team in their future jobs.

2.4 Module Structure Overview



2.5 Office Hours

It is a good practice to utilize the Moodle forum as an interactive place to promote peer-to-peer learning. For any question, please first check the Moodle forum and see if the questions are already posted and answered by any of your fellow classmate there. If not, please post your questions in the Moodle forum. For any forum questions, we will first leave for your fellow classmates to provide potential answers. The teaching team will also monitor the forum and clarify any unanswered questions.

If your question is still unsolved, please make appointments for office hours. I am available to answer any questions you may have about past lectures, case studies, and any other queries related to marketing analytics during my office hours. I will confirm the detailed time slots and the link in the first class.

Please make sure of the following before/when you make your appointment:

- Please be careful when operating on the Excel form and kindly avoid overwriting/modifying other students' appointments.
- Please use the "comment" function in Excel form to briefly describe the questions you have so that other students sharing similar questions can drop in as well.
- Each session lasts for 15 minutes. If you expect your questions may take a longer time to solve, you can book two consecutive sessions. Each student is entitled to at most two sessions per week to ensure equal opportunities for all students.

2.6 Programming Language

We will use an open-source programming language R throughout the course. R is one of the most commonly used programming languages by data scientists, economists, and statisticians, and is sometimes called the "golden child" of data science. R is cross-platform compatible; with over 16,000 packages ready for use in CRAN (R's open-source repository), it has one of the richest ecosystems to perform data analytics tasks. More importantly, it's free of charge compared with other commercial software (it is good to have cost-effectiveness in mind as you are now business school students)!

In this module, I will cover the basics of R and then focus on how to perform data-cleaning tasks efficiently with R. If you have little prior exposure to R, **it is highly recommended that you start learning some R basics before Week 2 to avoid falling behind**. UCL library has rich digital resources of R textbooks, which can be downloaded free of charge using your UCL credentials.

During (and even after) this module, whenever your run into any problems with R, Google (especially Stack Overflow) is always your best place to seek answers for most debugging issues; if you have any specific tasks in mind, the R Task Views page provides comprehensive guidance of which packages on CRAN are relevant for tasks related to a certain topic.

2.7 Classroom Etiquette

- **Punctuality**. Classes will start on time. Please be on time.
- **Class participation**. Please keep your mobile phones and laptops muted at all times unless otherwise instructed.

2.8 Assignments and Grading policy

Much of the learning during the course will happen with the help of individual assignments. Each assignment is carefully designed to examine the learning outcome of the topics taught during the previous weeks. Your final grade in this module will be based on 3 individual assignments with their associated weights as follows:

- 1st assignment, 30% weight, 1500 words, due on Oct 29
- 2nd assignment, 40% weight, 2000 words, due on Nov 26
- 3rd assignment, 30% weight, 1500 words, due on Dec 17

Detailed assignment descriptions will be given in due course.

2.8.1 Code of Conduct and Plagiarism

As all assignments in this module are **individual** assignments, you should **NOT** work with other students and the write-up should reflect your own work only. Please refrain from consulting classmates, friends, or seniors who have taken the course already. In the context of this module, it is acceptable to refer to concepts, frameworks, and analytical tools from the readings or class lectures with appropriate citation.

However, do not directly copy or paraphrase anything from other sources outside our classroom and present it as your own. It will be a violation of academic integrity if you base your assignments on solutions which you have found on the Internet or which you have obtained from others. You may risk failing this module if such a violation is found and investigated by the University. For detecting plagiarism, the School uses Turnitin. Turnitin checks your submission against 24+ billion web pages, 300+ million student papers, and 110,000+ publications and provides a similarity index. A particularly high similarity index will draw attention to the submission.

2.8.2 Word count

Please note, according to UCL's policy, you should **follow the word limit**. Otherwise, your mark may be reduced. Therefore, try to keep your answer concise and to the point.

2.8.3 Marks and Feedback

Your marks and feedback for each assignment will be usually released no later than 4 weeks after the submission date. Each assignment will be carefully marked according to the mark scheme by our teaching assistants, and there will be adequate second-marking procedures in the marking process to ensure the marking quality, as required by UCL. As a result, it is the university policy that students shouldn't question the academic judgement of markers, and **please refrain from emailing teaching assistants for any re-marking**.

2.9 How to Make the Best Use of Case Studies

To develop and test your skills in solving real-life marketing analytics problems, we will use many relevant case studies along the learning journey. Most case situations have been developed after careful research on actual situations faced by real companies and managers. The case authors have attempted to describe enough of the background and details of the situation in order to provide an adequate basis for class discussion.

Merely reading the case is not enough. To maximize the learning outcome, please follow the following advice for case preparations:

- After an initial reading to get a broad overview, go back and study the case thoroughly.
- Make notes wherever necessary and mark up the case, especially key information such as important numbers, to facilitate your understanding of the case background.
- Carefully think about the key problems. Always try to come up with alternative solutions to the problem and think about the pros and cons of each.

2.10 Copyright Issues

Chapter 3

Detailed Class Descriptions

Students are expected to complete the required preparation work for each week's class, including reading case studies, preparing for case questions, etc. Detailed guidelines on how to prepare for each week are described below. Please follow the guidelines unless otherwise informed.

Note that the class schedules are tentative and are **subject to changes based on our progress**, which means this online supplement may be updated regularly. Therefore, It's a good practice to come back and check if there are any updates to the guideline each Wednesday before Thursday's class.

Some explanations:

- `(before class)` means the reading material should be read and prepared before that week's class.
- `(after class)` means the reading material serves as a supplementary after-class reading to enhance your understanding of the week's lecture contents.
- `(optional)` means the reading material is technically challenging but can be handy when you need them. You can selectively read them based on your time availability.

3.1 Week 1 (Thu, 07 Oct)

3.1.1 Class 1: Introduction to Marketing Process

Class objectives:

- To provide an overview of the course topics and requirements
- To understand the typical marketing process in real-life examples

Readings:

- Supplementary reading: "Marketing Process" (after-class)

3.1.2 Class 2: Marketing Metrics: Customer Lifetime Value

Class objectives:

- Understand common marketing metrics
- Know how to compute customer lifetime value and how to use CLV to guide marketing decisions
- Apply CLV calculation in real-life business cases: Tesco & Vodafone

Readings:

- Assignment 1 Part I: Vodafone: Using CLV for Better Marketing Decisions (before-class, 15 min prep time)
- Supplementary reading: "Customer Lifetime Social Value" (after-class; optional)

Preparations:

- Important: Please read the Vodafone case carefully before Week 1's class. Highlight any information (such as figures or key information) you feel may be relevant to solving the case. We will discuss the Vodafone case in class and solve the "in-class" questions which are listed at the end of the case.
- If you haven't learned linear algebra before and if your time allows, I recommend you start reading "Review of basic linear algebra", which is a prerequisite for Week 4.

Assignment 1:

- Assignment 1 examines your learning outcome from Week 1 to Week 4, including CLV; basic customer analysis; and causal inference. You now have access to Part I of Assignment 1. I suggest you start working on the part I now so that you leave yourselves enough time for the remaining parts.

3.2 Week 2 (Thu, 14 Oct)

3.2.1 Class 3: Review of Statistics

Class objectives:

- Understand the type of data and variables commonly used in marketing analytics
- Review of statistics
  - Statistical inference & hypothesis testing

  - Statistical tests commonly used in marketing

Readings:

- "Tips for using statistics in marketing analytics" (after class)

Preparations:

- You should have installed R and RStudio on your laptops **before this class**

3.2.2 Class 4: Basic Customer Analysis with R

Class objectives:

- Understand the usage and syntax of R data.table packages for data manipulation
- Able to apply basic statistics for basic customer analysis in business cases

Readings:

- Case study: "Basic Customer Analysis: Amazon Prime" (before class, 10 min prep time)
- Supplementary reading: data.table Cheat Sheet (after class)
- Assignment 1 Part II: "Basic Customer Analysis with Amazon Sales Data" (after class)

Preparations:

- Please read the case "Basic Customer Analysis: Amazon Prime". There are quite a few questions at the end of the case. We will devote most of this class to hands-on exercise with R and data.table, answering the questions listed at the end of the Amazon Prime case.
  - We will use the dataset "Amazon Prime.csv" from Moodle. Remember to download the dataset to your hard disk before class.
- You should be able to solve the part II of assignment 1 after this week's class. Get started now and remember not to procrastinate until the deadline!

3.3 Week 3 (Thu, 21 Oct)

3.3.1 Class 5: When Marketing Analytics Can Go Wrong

Class objectives:

- Learn how to tell good analytics from bad ones

- Understand how to avoid interpreting statistical results incorrectly

Readings:

- Case study: "When Marketing Analytics Can Go Wrong: Case Studies" (before class, 20 min prep time)

Preparations:

- Read all case studies carefully before class
- Prepare and think over the questions at the end of each case for class discussion

3.3.2 Class 6: Causal Inference and Randomized Experiments

Class objectives:

- Understand the Rubin causal inference framework
- Know the keys steps to run experiments
- Understand the pitfalls when running experiments

Readings:

- Case study: "Superb Trucks Lp: How to Conduct a Field Experiment" (before class, 20 min prep time)

Preparations:

- Read the case "Superb Trucks Lp" before class. We will demonstrate how to use field experiments to solve marketing problems with this

3.4 Week 4 (Thu, 28 Oct)

3.4.1 Class 7: Linear Regression Basics

Class objectives:

- Understand the concept of "data generating process" and a "model"
- Understand linear regression models from different perspectives
- Understand how to interpret the regression coefficients

  - Continuous variables and standardization
  - Log transformation
  - Categorical variables
  - Interactions

Readings:

- Prerequisite reading: "Review of basic linear algebra" (before class, 60 min prep time)
- Supplementary reading: "Practical Regression: Regression Basics" (before and after class)

Preparations:

- This week, we will learn the OLS linear regression model. This is the simplest yet most powerful model for data analytics, and it is the cornerstone of the remaining contents of this module. Therefore, it is important that you are well-prepared and stay focused for the class.
- *Important:* Please finish the "Review of basic linear algebra" before class. This is important, as during the lecture, I will show you how to run regression using linear algebra.

3.4.2 Class 8: Endogeneity with Observational Data

Class objectives:

- Understand the concept of selection bias
- Understand the common causes of endogeneity from observational data

Readings:

- Supplementary reading: "Practical Regression: Introduction to Endogeneity: Omitted Variable Bias" (before and after class)

3.5 Week 5 (Thu, 04 Nov)

3.5.1 Class 9: Tackling Endogeneity Part I: Non-Experimental Methods

Class objectives:

- Understand the intuition behind the instrumental variable methods
  - Able to find appropriate instrumental variables to solve endogeneity using 2SLS method.
- Understand the intuition behind the fixed effects

Readings:

- Supplementary reading: "Practical Regression: Causality and Instrumental Variables" (before and after class)

Preparations:

3.5.2 Class 10: Tackling Endogeneity Part II: Quasi-Experimental Methods

Class objectives:

- Understand the intuition and how to use difference-in-differences method
- Understand the intuition and how to use regression discontinuity method
- Learn real-life applications of quasi-experimental methods

3.6 Week 6 (Thu, 11 Nov)

3.6.1 Class 11: Foundations of Predictive Analytics

Class objectives:

- Understand the difference between prescriptive analytics and predictive analytics
- Understand the bias-variance tradeoff in predictive analytics
- Understand the difference between regression problems and classification problems

Readings:

- Supplementary reading: "Assessing Prediction Accuracy of Machine Learning Models" (after class)
- Supplementary reading: Varian, Hal R. 2014. 'Big Data: New Tricks for Econometrics'. Journal of Economic Perspectives 28 (2): 3–28. (after class)
  - page 6-7: General Considerations for Prediction
  - page 21-24: Econometrics and Machine Learning

3.6.2 Class 12: Prospecting New Customers: RFM Analysis

Class objectives:

- Understand how to use RFM analysis to prospect new customers and improve the effectiveness of marketing activities
  - The concept of RFM
  - The steps to do RFM
  - Break-even analysis of RFM
  - Important variants of RFM
- Understand the concept of "lift" and "gain", and how to use the concepts to evaluate predictive models

Readings:

- Case study: "UberEat: Mobile App Push Messaging" (before class, 10 min prep. time)
- Assignment 2: "Amazon Prime: Using Predictive Analytics to Improve Marketing ROI" (after class)

Preparations:

- Read and prepare "UberEat: Mobile App Push Messaging" for class discussion; the dataset "UberEat.csv" is on Moodle. Note that, for you to better understand and be able to compare different predictive models, we will be using the same UberEat case study and dataset in the next few weeks. Each week, we will learn a new predictive model, and you will learn how to apply different models on the UberEat dataset, and compare their performance.
- It's time to catch up with our old friend again in the assignment. After this class, you should be able to solve the Part I of Assignment 2 "Amazon Prime: Using Predictive Analytics to Improve Marketing ROI". It's good to start early as the second assignment accounts for 40% of the total grade.

3.7 Week 7 (Thu, 18 Nov)

3.7.1 Class 13 & 14: Regression Problems: A Revisit to Regression Models

Class objectives:

- Review of simple linear regression models
- Learn the definition and application of step-wise and backward stepwise regression models.
- Learn the concept of L1 and L2 regularization in the context of linear regression models

- Know the pros and cons of Lasso and Ridge regressions; Understand when and how to apply the two regularized regression models

Readings:

- Supplementary reading: Varian, Hal R. 2014. 'Big Data: New Tricks for Econometrics'. Journal of Economic Perspectives 28 (2): 3–28. (after class)
  - page 15: Variable Selection

3.8 Week 8 (Thu, 25 Nov)

3.8.1 Class 15 & 16: Classification Problems: Logistic Regression

Class objectives:

- Understand the mathematics and intuition behind the logistic regression
- Understand how to interpret logistic regression results
  - probability, odds, odds ratio, and log odds ratio
- Understand how to score customers using logistic regression
- Compare the predictive power of logistic regression against RFM analysis

Preparations:

- This week, I will demonstrate how to use logistic regression to solve the UberEat case. We will also compare the performance of logistic regression with that of RFM model.
- At this point we have covered everything you need to know to work on the second assignment.

3.9 Week 9 (Thu, 02 Dec)

3.9.1 Class 17: Advanced Machine Learnings Models

Class objectives:

- Learn the intuition behind advanced machine learning models and their application
  - Decision tree
  - Bagging and boosting models
  - Deep learning

- Learn how to use automatic machine learning tools to make our life easier

Readings:

- Supplementary reading: R-blog, ["https://www.r-bloggers.com/2020/04/automl-frameworks-in-r-python/"] (before and after class)
- Supplementary reading: Varian, Hal R. 2014. 'Big Data: New Tricks for Econometrics'. Journal of Economic Perspectives 28 (2): 3–28. (after class)
  - page 7-15: Classification and Regression Trees

Preparations:

- Follow the instructions in https://docs.h2o.ai/h2o/latest-stable/h2o-docs/downloading.html and install h2o package on your laptop before the class.

3.9.2 Class 18: Cross-selling and Up-selling

Class objectives:

- Understand the benefits of using customer purchase data for customer development
- Understand how to cross-sell and up-sell

Readings:

- Case study: "Cross-Selling Insurance Products Using Predictive Analytics" (before class)
- Supplementary reading: Knott, Hayes, and Neslin "Next-Product-to-Buy Models for Cross-Selling Applications," Journal of Interactive Marketing 16(3) (2002) (after class)

Preparations:

- Please read the "Cross-Selling Insurance Products Using Predictive Analytics" before class. Since we have already learned all the tools in the previous weeks. Please try if you can develop you own predictive model using RFM and logistic regression, which we have covered in previous weeks.
- In the class, I will demonstrate how to use more sophisticated machine learning techniques to solve the case.

3.10 Week 10 (Thu, 09 Dec)

3.10.1 Class 19: Managing Customer Churns

Class objectives:

- Understand how to use attrition prediction for managing customer churn management

Readings:

- Case study: "Customer Churn Management Using Predictive Analytics" (before class)

Preparations:

- Read the case "Customer Churn Management Using Predictive Analytics" and prepare the questions for class discussion
- Since we have covered all the necessary predictive models in the previous weeks, please explore the dataset and develop at least one predictive model for managing customer churn before class. I will invite some of you to share your solutions in class.

3.10.2 Class 20: Module Wrap-up

Class objectives:

- To review and synthesize the module

Chapter 4

R Tutorials

This is a supplementary teaching material for the MSc Business Analytics program at UCL School of Management. This tutorial aims to provide a comprehensive introduction to the basics of R programming language.

4.1 Introduction to R History

4.1.1 Class Objective

Learning objective:

- Be able to setup R and Rstudio
- Understand how use quickly install and load packages in R
- Understand the key concepts of R

4.1.2 What is R

- R is both a programming language and software environment for statistical computing, which is free and open-source (https://www.r-project.org/about.html).
- The ***R Project*** was initiated by Robert Gentleman and Ross Ihaka (University of Auckland) in the early 1990s as a different implementation of the S language, which was developed at Bell Laboratories.
- Since 1997, R has been developed by the ***R Development Core Team***.

4.1.3 What is R used for?

R can do almost anything you can think of! Below is a (incomplete) list of what I normally use R for on a daily basis.

- Data analytics (statistics, data manipulation, machine learning, econometric analysis, etc.)
- Building your own homepage. I build and maintain my personal website with the `blogdown` package completely in R
- Prepare your presentations using R `markdown`
- Write books (such as this whole online supplement) using R `bookdown`

You can even scrawl webpages and send emails with R once you install the right packages.

4.1.4 Popularity of R

R has been one of the most popular programming language used by statisticians, economists, and data analysts.

Below is the popularity index of different languages by tiobe.com. R reached its historic highest place in August 2020, ranking the 8th most popular language among all.

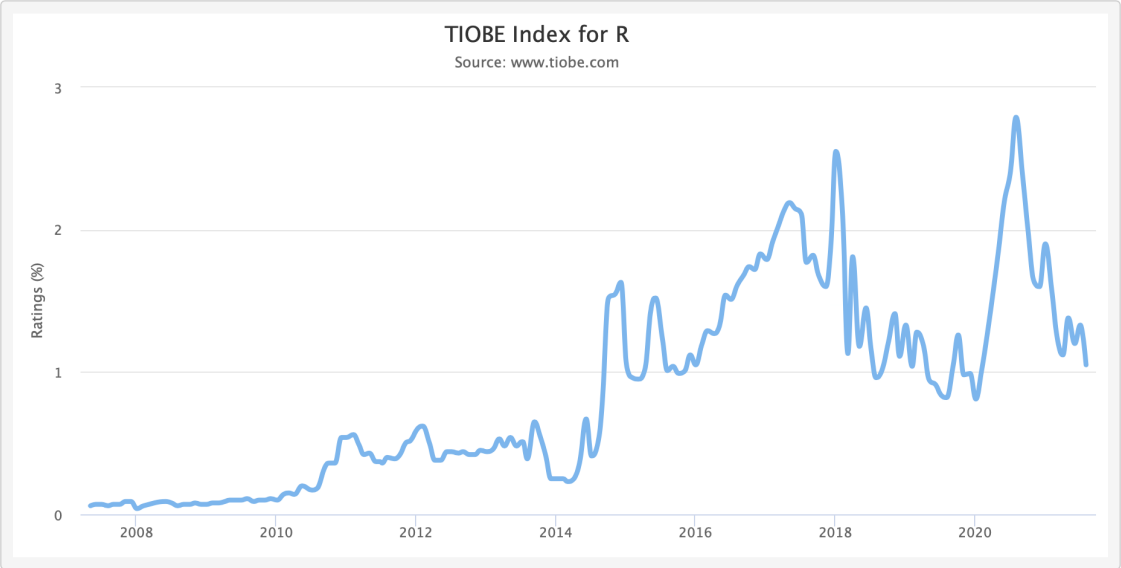


Figure 4.1: R's Ranking

4.1.5 Install and Setup R on your computers: Step 1

There are alternative ways to install and use R on your computers. Below is the most recommended and hassle free way to install and setup R on your computer.

Step 1: Download R from the official website CRAN.

Tips:
- Note that R version is updated on a regular basis, to fix bugs or add new features. It's good to keep your R up to date with the official website.
- However, sometimes across major versions, some R's base functions may be depreciated and cause

breaks in your code, so it's important to read the R News after each update, so that you understand what's been updated in the latest version and if there will be any breaks in your previous code. (For a deeper understanding of this issue, please refer to this article regarding reproducible research with R.)

4.1.6 Screenshot of RConsole

R's default user interface:

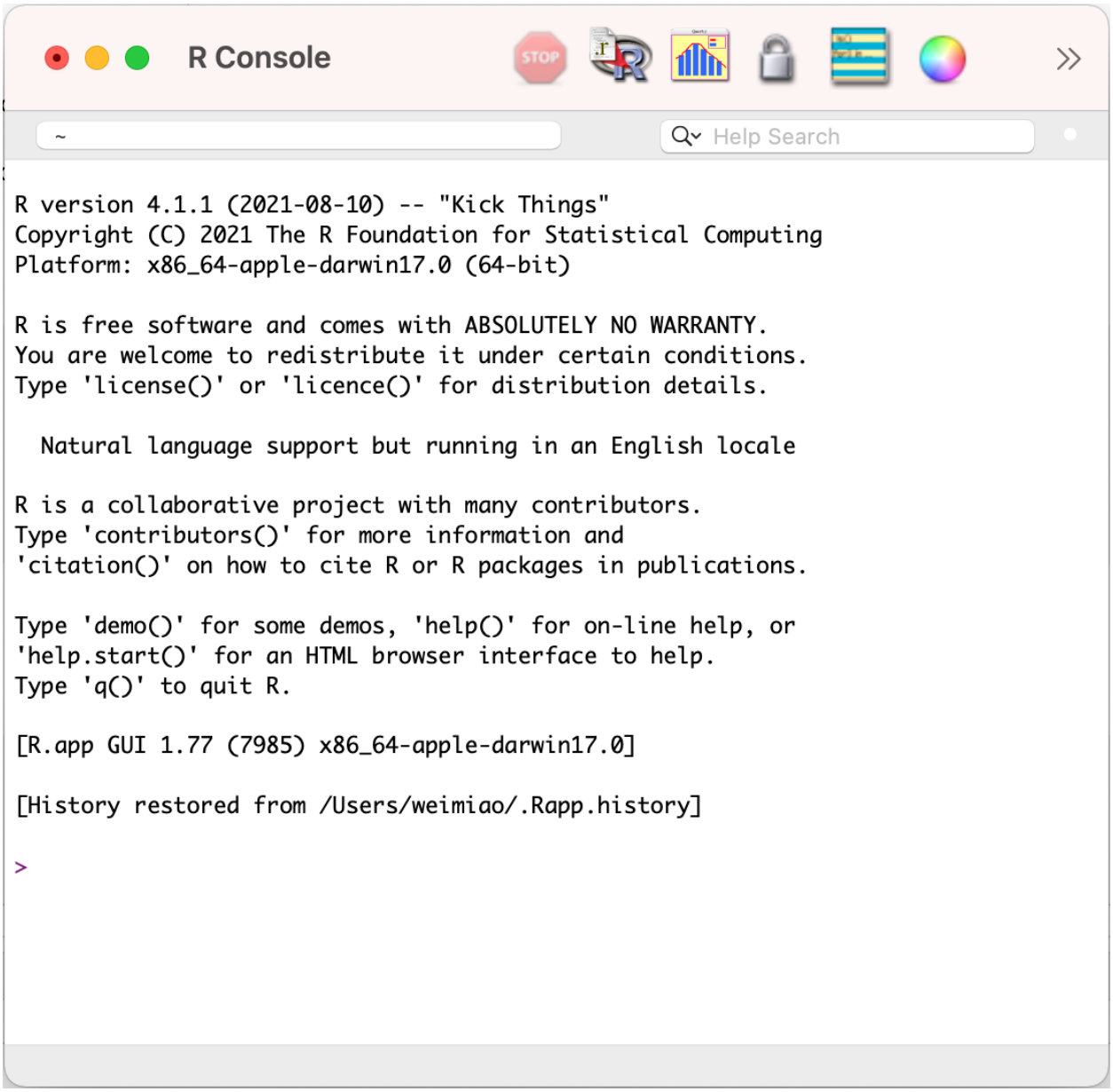


Figure 4.2: R Console

4.1.7 Install and Setup R on your computers: Step 2

After step 1, you can already open R's own GUI and run R codes. However, we still need an integrated development environment (IDE) to help us better write R codes. Rstudio is currently the best IDE for R.

Step 2: Download and install Rstudio from Rstudio's website.

For a introduction to the Rstudio IDE, refer to this cheat sheet.

4.1.8 Screenshot of RStudio

Our preferred R editor: Rstudio

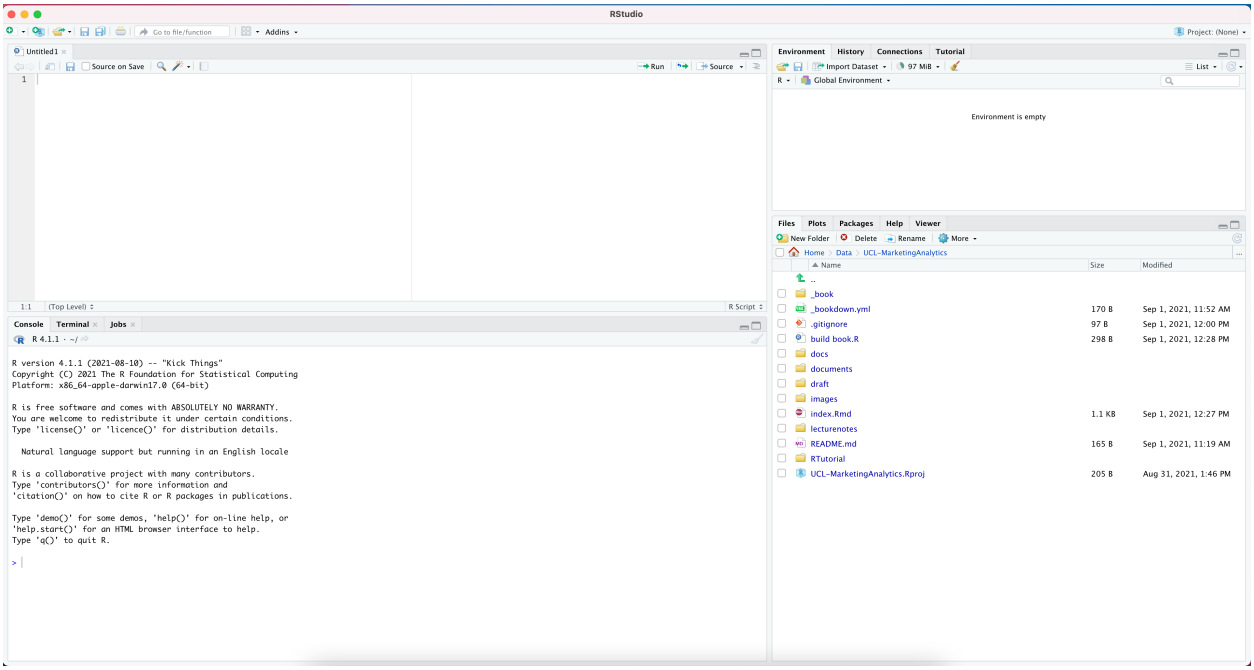


Figure 4.3: Rstudio

4.1.9 Glossary

- ***script***: (top left) where you write codes a text-readable file (E.g., 'test.R', 'test.Rmd', etc.), each separated by a new line.
- ***console***: (bottom left) where you can type commands interactively with R and get the results immediately
- ***environment***: (top right) a list of named variables/objects that we have generated; can be checked on the 'environment' panel.
- ***history***: (top right) the list of past commands thaty we have used
- ***help***: (bottom right) a documentation of all the functions available in R (the user manual)

- ***package***: (bottom right) a collection of additional functions and dataset

4.1.10 Use Help

Whenever you don't know how to use a function, you can either

- use `help`. Enter the command `help(log)` in R console

```
?log
help(log)
```

- or you can directly search for the function in RStudio help panel.

If you cannot get an answer:

- (1st) Google your error message or search/ask for help in Stack Overflow
- (2nd) Ask in Moodle Forum
- (3rd) Use office hours

4.1.11 Libraries and packages management: the coventional way

To achieve certain tasks, for instance, to run a regression analysis, we can definitely write our own code from scratch. However, it may take much time if we would like to build everything from zero. That's when the packages/libraries can help us to improve our efficiency by a large margin.

One of the best things about R is that it has tens of thousands of extremely useful packages/libraries. And some are even exclusive to R, for instance, the `grf` package that runs the causal forest model, which I have used for my research.

```
# install.packages('data.table') # install the package
library('data.table') # load the package
```

4.1.12 Libraries and packages management: the simplest way

In R, there is a package called `pacman`, which stands for package management and is probably the most convenient way to manage R packages.

Task: Load `ggplot2` and `data.table` packages; if they are not installed, install them and then load them.

```
# install.packages('pacman')

pacman::p_load(data.table,ggplot2)
```

4.1.13 Comment your codes

You can put a "#" before any line of code, to indicate that this line is your comment, and this line will turn green and not run by R.

```
# print("David is handsome")
```

4.2 R Basics

4.2.1 Use R as a calculator

We can do the basic mathematical calculation in R as follows:

```
8+8
```

```
## [1] 16
```

```
8-8
```

```
## [1] 0
```

```
8*8
```

```
## [1] 64
```

```
2/2
```

```
## [1] 1
```

Common mathematical operations can also be used in R.

```
log(1)
```

```
## [1] 0
```

```
exp(1)
```

```
## [1] 2.718282
```

```
log(exp(1))
```

```
## [1] 1
```

```
sqrt(25)
```

```
## [1] 5
```

4.2.2 Assign value

In R, we use the left arrow to assign a value to an object `<-`

```
x <- log(2)
x
```

```
## [1] 0.6931472
```

You can also use equal sign `=` without issues, but it's recommended to stick with a language's tradition.

4.2.3 Data types

- Numeric/Double (e.g. 2.5, 1/5, 1.0, etc.)
- Integer (e.g. 1, 2, 3, etc.)
- Complex (e.g. 1 + 2i, etc.)
- Logical (e.g. TRUE, FALSE or NA)
- Character (e.g. "Wei", "UCL", "1 + 1 = 3", "TRUE", etc.)
- Factor/Categorical ("male", "female", etc.)
- Special type: `NA` which stands for missing values, a very tricky type.

```
# Character Vector
str1 <- c("R","Python","Java","Scala","Julia")

# Numeric Vector
num2 <- c(1, 2, 3.4, 5, 5.9)

# Integer Vector
int3 <- (1:10)

# Logical Vector
log4 <- c(TRUE, FALSE, FALSE, T, F)

# 5. Complex
com5 <- c(2+3i, 3+5i, 4+5i)
```

```
missing <- c(1,NA,2,NA)

mean(missing)
```

```
## [1] NA
```

```
mean(missing,na.rm = T) ##### important!!!, do not forget na.rm = T
```

```
## [1] 1.5
```

```
# check missing values
is.na(missing)
```

```
## [1] FALSE  TRUE FALSE  TRUE
```

```
anyNA(missing)
```

```
## [1] TRUE
```

```
sum(is.na(missing))
```

```
## [1] 2
```

4.2.4 Data type: other special values

- `Inf` is infinity. You can have either positive or negative infinity.
- `NaN` means Not a Number. It's an undefined value.

```
a <- 1/0 #Inf
a
```

```
## [1] Inf
```

```
is.infinite(a)
```

```
## [1] TRUE
```

```
is.na(a)
```

```
## [1] FALSE
```

```
b <- 0/0 #NaN
b
```

```
## [1] NaN
```

```
is.nan(b)
```

```
## [1] TRUE
```

```
is.na(b)
```

```
## [1] TRUE
```

4.2.5 Data type: check data type using class()

We can use `class()` to check the type of an object in R.

```
a <- '1+1=2'

a
```

```
## [1] "1+1=2"
```

```
class(a)
```

```
## [1] "character"
```

4.2.6 Data type: conversion

```
a <- '1'
class(a)
```

```
## [1] "character"
```

```
b <- as.numeric(a)
class(b)
```

```
## [1] "numeric"
```

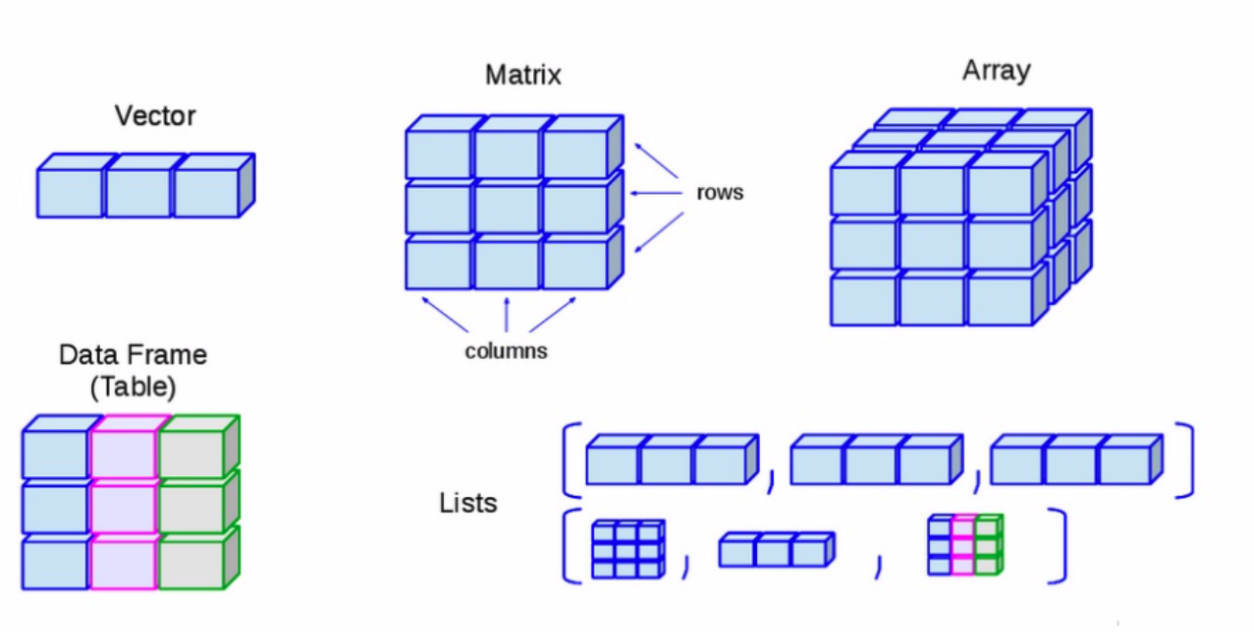


Figure 4.4: Visualization of data structures

4.2.7 Data structures

4.2.8 Vectors: creating vectors

4.2.8.1 Creating vectors: c()

Vector are created using the command `c()`.

```
x <- c(1, 3, 5, 10)
x
```

```
## [1]  1  3  5 10
```

```
class(x)
```

```
## [1] "numeric"
```

Vectors must contain elements of the same data type. Otherwise, it will convert elements into the same type.

```
x <- c(1, "intro", TRUE)

class(x)
```

```
## [1] "character"
```

4.2.8.2 Creating vectors: length()

You can measure the length of a **vector** using the command `length()`

```
x <- c('R',' is', ' fun')
length(x)
```

```
## [1] 3
```

```
y <- c(NULL)
length(y)
```

```
## [1] 0
```

4.2.8.3 Creating vectors: seq() and rep()

It is also possible to easily create sequences

```
# use colon ':'
1:10
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
# use seq()
seq(from = 1, to = 2, by = 0.1)
```

```
##  [1] 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0
```

```
# for date object
seq.Date(from = as.Date("2021-09-27"),
         to = as.Date("2021-10-01"),
         by = '1 days')
```

```
## [1] "2021-09-27" "2021-09-28" "2021-09-29" "2021-09-30" "2021-10-01"
```

```
# replication using rep()
rep("A", times = 5)
```

```
## [1] "A" "A" "A" "A" "A"
```

To understand all usages of a function, use `help()`

4.2.8.4 Creating vectors: combine vectors

You can use `c` to combine different vectors; this is very commonly used when concatenating vectors.

```
x <- 1:3 # from 1 to 3
y <- c(10, 15) # 10 and 15
z <- c(x,y) # x first and then y
z
```

```
## [1]  1  2  3 10 15
```

4.2.9 Vectors: subsetting

```
x <- c(1,3,8,7)
x[2] # which element is in the second position?
```

```
## [1] 3
```

```
x[1:2] # which elements are in the first 2 positions?
```

```
## [1] 1 3
```

```
x[c(1,3,4)] # which elements are in positions 1, 3 and 4?
```

```
## [1] 1 8 7
```

4.2.10 Vectors: relational operation

```
x <- c(1,3,8,7)
x < 6 # is each element lower than 6?
```

```
## [1]  TRUE  TRUE FALSE FALSE
```

```
x == 10 # is the element equal to 10?
```

```
## [1] FALSE FALSE FALSE FALSE
```

```
which(x == 8) # which element equals 8
```

```
## [1] 3
```

```
max(x) # what is the max element
```

```
## [1] 8
```

```
which.max(x) # which is the max element
```

```
## [1] 3
```

```
x[(which.min(x))] #  = min(x)
```

```
## [1] 1
```

```
## logical operation
T & F # and
```

```
## [1] FALSE
```

```
T | F # or
```

```
## [1] TRUE
```

```
!T # not
```

```
## [1] FALSE
```

A special relational operation is `%in%` in R, which tests whether an element exists in the object.

```
3 %in% x
```

```
## [1] TRUE
```

```
4 %in% x
```

```
## [1] FALSE
```

```
3 %in% x & 4 %in% x
```

```
## [1] FALSE
```

4.2.11 Vectors: elementwise operations

Similar to other matrix languages (Matlab), R is a vectorized language, meaning by default it will do vector operation internally.

```
x <- c(1,3,8,7)
x+2
```

```
## [1]  3  5 10  9
```

```
x-2
```

```
## [1] -1  1  6  5
```

```
x*2
```

```
## [1]  2  6 16 14
```

```
x^2
```

```
## [1]  1  9 64 49
```

4.2.11.1 Caveat with vector elementwise operation

Caveat When the length of vectors do not match, R will still do it for your without reporting error but a warning message. As you can see, even if the length of vectors does not match, R can still return an output.

It's important to check the warning messages when there is any!!

```
x <- c(1,3,8,7)

y <- c(1,3,4) # careful!!! does not report error
x + y
```

```
## Warning in x + y: longer object length is not a multiple of shorter object length
```

```
## [1]  2  6 12  8
```

```
z <- c(10,12,13,14)
x + z
```

```
## [1] 11 15 21 21
```

4.2.12 Matrices: creating matrices

4.2.12.1 Creating matrices: matrix()

A matrix can be created using the command `matrix()`
- the first argument is the vector to be converted into matrix
- the second argument is the number of rows
- the last argument is the number of cols (optional)

```
matrix(1:9, nrow = 3, ncol = 3)
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```
matrix(1:9, nrow = 3, ncol = 3)
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```
matrix(1:9, nrow = 3, ncol = 2)
```

```
## Warning in matrix(1:9, nrow = 3, ncol = 2): data length [9] is not a sub-multiple or multipl
```

```
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
## [3,]    3    6
```

Note: R automatically inserts elements by columns

4.2.12.2 Creating matrices: inserting by row

However, we can ask R to insert by rows by setting the 'byrow' argument.

```
matrix(1:9, nrow = 3, ncol = 3, byrow = TRUE)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
```

4.2.12.3 Creating matrices: concatenation of matrices cbind() and rbind()

We can use `cbind()` and `rbind()` to concatenate vectors and matrices into new matrices.

```
x <- cbind(1:3, 4:6) # column bind
x
```

```
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
## [3,]    3    6
```

```
y <- rbind(7:9, 10:12) # row bind
y
```

```
##      [,1] [,2] [,3]
## [1,]    7    8    9
## [2,]   10   11   12
```

```
cbind(x,x)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    1    4
## [2,]    2    5    2    5
## [3,]    3    6    3    6
```

4.2.13 Matrices: indexing and subsetting

```
x
```

```
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
## [3,]    3    6
```

```
x[1,2] # the element in the 1st row, 2nd column
```

```
## [1] 4
```

```
x[1,] # all elements in the first row
```

```
## [1] 1 4
```

```
x[,2] # all elements in the second columns
```

```
## [1] 4 5 6
```

```
x[c(1,2),]
```

```
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
```

4.2.14 Matrices: operations

Let's use 3 matrices `x`, `y`, and `z`:

```
x
```

```
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
## [3,]    3    6
```

```
y
```

```
##      [,1] [,2] [,3]
## [1,]    7    8    9
## [2,]   10   11   12
```

```
z <- x^2
z
```

```
##      [,1] [,2]
## [1,]    1   16
## [2,]    4   25
## [3,]    9   36
```

4.2.14.1 Matrices' operations: matrix addition and multiplication

```
x+z   # elementwise addition
```

```
##      [,1] [,2]
## [1,]    2   20
## [2,]    6   30
## [3,]   12   42
```

```
x*z   # elementwise multiplication
```

```
##      [,1] [,2]
## [1,]    1   64
## [2,]    8  125
## [3,]   27  216
```

```
X%*%Y # matrix multiplication
```

```
##      [,1] [,2]
## [1,]   18   13
## [2,]   46   32
```

4.2.14.2 Matrices' operations: inverse and transpose

```
t(x) # transpose
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
```

```
t(x)%*%x
```

```
##      [,1] [,2]
## [1,]   14   32
## [2,]   32   77
```

```
solve(t(x)%*%x) # inverse; must be on a square matrix
```

```
##            [,1]       [,2]
## [1,]  1.4259259 -0.5925926
## [2,] -0.5925926  0.2592593
```

4.2.15 Arrays

```
x <- 1:4
x <- array(data = x, dim = c(2,3,2))
x
```

```
## , , 1
##
##      [,1] [,2] [,3]
## [1,]    1    3    1
## [2,]    2    4    2
##
## , , 2
##
##      [,1] [,2] [,3]
## [1,]    3    1    3
## [2,]    4    2    4
```

- Just like vectors and matrices, arrays can include only data types of the same kind.
- A 3D array is basically a combination of matrices each laid on top of other

4.2.16 Lists

A list is an R object that can contain anything.

```
x <- 1:2
y <- c("a", "b")
L <- list( numbers = x, letters = y)
```

4.2.17 Lists: indexing and subsetting

There are many ways to extract a certain element from a list.
- by index
- by the name of the element
- by dollar sign $

```
L[[1]] # extract the first element
```

```
## [1] 1 2
```

```
L[['numbers']] # based on element name
```

```
## [1] 1 2
```

```
L$numbers # extract the element called numbers
```

```
## [1] 1 2
```

After extracting the element, we can work on the element further:

```
L$numbers[1:3] > 2
```

```
## [1] FALSE FALSE    NA
```

4.2.18 Data Frames: creating dataframe

4.2.18.1 Data Frames: create dataframe using data.frame()

Data Frame is the R object that we will deal with most of the time in the MSc program. You can think of `data.frame` as a spreadsheet in excel.

```
df <-  data.frame(id = 1:4,
  name = c("David", "Yongdong", "Anil", "Wei"),
  wage = rnorm(n=4, mean = 10^5, sd = 10^3),
  male = c(T, T, T, T)
  )
df
```

```
##   id     name      wage male
## 1  1    David 102172.61 TRUE
## 2  2 Yongdong 100475.51 TRUE
## 3  3     Anil  99290.05 TRUE
## 4  4      Wei 100610.73 TRUE
```

4.2.18.2 Data Frames: create dataframe using read.csv()

```
#read.csv('data.csv')
```

4.2.18.3 Use R's own dataset

```
data("mtcars")
```

4.2.19 Data Frames: property

Each row stands for an observation; each column stands for a variable.

Each column should have a **unique** name.

Each column must contain the same data type, but the different columns can store different data types.
- dataframe is a special type of matrix, but allows different types across columns.

Each column must be of same length, because rows have the same length across variables.

You can verify the size of the `data.frame` using the command `dim()`; or `nrow()` and `ncol()`

```
dim(df)
```

```
## [1] 4 4
```

```
nrow(df)
```

```
## [1] 4
```

```
ncol(df)
```

```
## [1] 4
```

You can get the `data type` info using the command `str()`

```
class(df)
```

```
## [1] "data.frame"
```

```
str(df)
```

```
## 'data.frame':    4 obs. of  4 variables:
##  $ id  : int  1 2 3 4
##  $ name: chr  "David" "Yongdong" "Anil" "Wei"
##  $ wage: num  102173 100476 99290 100611
##  $ male: logi  TRUE TRUE TRUE TRUE
```

```
lapply(df, class)
```

```
## $id
## [1] "integer"
##
## $name
## [1] "character"
##
## $wage
## [1] "numeric"
##
## $male
## [1] "logical"
```

4.2.20 Data Frames: inspection

```
head(df, n=3) # first n observations
```

```
##   id     name      wage male
## 1  1    David 102172.61 TRUE
## 2  2 Yongdong 100475.51 TRUE
## 3  3     Anil  99290.05 TRUE
```

```
tail(df, n=3) # last n observations
```

```
##   id     name      wage male
## 2  2 Yongdong 100475.51 TRUE
## 3  3     Anil  99290.05 TRUE
## 4  4      Wei 100610.73 TRUE
```

```
names(df)
```

```
## [1] "id"   "name" "wage" "male"
```

```
summary(df)
```

```
##        id           name                wage          male
##  Min.   :1.00   Length:4           Min.   : 99290   Mode:logical
##  1st Qu.:1.75   Class :character   1st Qu.:100179   TRUE:4
##  Median :2.50   Mode  :character   Median :100543
##  Mean   :2.50                      Mean   :100637
##  3rd Qu.:3.25                      3rd Qu.:101001
##  Max.   :4.00                      Max.   :102173
```

```
str(df)
```

```
## 'data.frame':    4 obs. of  4 variables:
##  $ id  : int  1 2 3 4
##  $ name: chr  "David" "Yongdong" "Anil" "Wei"
##  $ wage: num  102173 100476 99290 100611
##  $ male: logi  TRUE TRUE TRUE TRUE
```

4.2.21 Data Frames: subsetting

Since a dataframe is essentially a matrix, all the subsetting syntax with matrices can be applied here.

```
df$name # subset a column
```

```
## [1] "David"    "Yongdong" "Anil"     "Wei"
```

```
df[,c(2,3)] # can also subset like a matrix
```

```
##       name      wage
## 1    David 102172.61
## 2 Yongdong 100475.51
## 3     Anil  99290.05
## 4      Wei 100610.73
```

We are interesting in the cylinders and the weights of inefficient cars (lower than 15 miles per gallon).

```
poll_cars <- mtcars[mtcars$mpg<15, c("cyl", "wt")] # remember to assign the generated datafram
poll_cars
```

```
##                     cyl    wt
## Duster 360            8 3.570
## Cadillac Fleetwood    8 5.250
## Lincoln Continental   8 5.424
## Chrysler Imperial     8 5.345
## Camaro Z28            8 3.840
```

4.3 Programming Basics

4.3.1 Variables

In programming, a variable denotes an object, i.e., a variable is a name that refers to an object in the memory. Variables in R programming can be used to store any R object discussed previously, including numbers, characters, matrices, and data frames.

R is a ***dynamically programmed language**, which means that unlike other programming languages (such as C++, which is a pain to debug), we do not have to declare the data type of a variable before we can use it in our program.

For a variable to be valid, it should follow these rules

- It should contain letters, numbers, and only dot or underscore characters.
- It should not start with a number (eg:- 2iota)
- It should not start with a dot followed by a number (eg:- .2iota)
- It should not start with an underscore (eg:- _iota)
- It should not be a reserved keyword.

4.3.2 if/else

Sometimes, you want to run your code based on different conditions. For instance, if the observation is a missing value, then use the population average to impute the missing value. This is where `if/else` kicks in.

```
if (condition == TRUE) {
  action 1
} else if (condition == TRUE ){
  action 2
} else {
  action 3
}
```

Example 1:

```
a <- 15

if (a > 10) {
larger_than_10 <- TRUE
} else {
  larger_than_10 <- FALSE
}

larger_than_10
```

```
## [1] TRUE
```

Example 2:

```
x <- -5
if(x > 0){
  print("x is a non-negative number")
} else {
  print("x is a negative number")
}
```

```
## [1] "x is a negative number"
```

4.3.3 Loops

As the name suggests, in a loop the program repeats a set of instructions many times, until the stopping criteria is met.

Loop is very useful for repetitive jobs.

```
for (i in 1:10){ # i is the iterator
  # loop body: gets executed each time
  # the value of i changes with each iteration
}
```

4.3.4 Nested loops

We can also nest loops into other loops.

```
x <- cbind(1:3, 4:6) # column bind
x
```

```
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
## [3,]    3    6
```

```
y <- cbind(7:9, 10:12) # row bind
y
```

```
##      [,1] [,2]
## [1,]    7   10
## [2,]    8   11
## [3,]    9   12
```

```
z <- x

for (i in 1:nrow(x)) {
  for (j in 1:ncol(x)){
    z[i,j] <- x[i,j] + y[i,j]
  }
}

z
```

```
##      [,1] [,2]
## [1,]    8   14
## [2,]   10   16
## [3,]   12   18
```

4.3.5 Functions

A function takes the argument as input, run some specified actions, and then return the result to us.

Functions are very useful. When we would like to test different ideas, we can combine functions with loops: We can write a function which takes different parameters as input, and we can use a loop to go through all the possible combinations of parameters.

4.3.5.1 User-defined function syntax

Here is how to define a function in general:

```
function_name <- function(arg1 ,arg2 = default_value){
  # write the actions to be done with arg1 and arg2
  # you can have any number of arguments, with or without defaults
  return() # the last line is to return some value
}
```

Example:

```
magic <- function( x, y){
  return(x^2 + y)
}

magic(1,3)
```

```
## [1] 4
```

4.3.6 A comprehensive example

Task: write a function, which takes a vector as input, and returns the max value of the vector

```
get_max <- function(input){
  max_value <- input[1]
  for (i in 2:length(input) ) {
    if (input[i] > max_value) {
      max <- input[i]
    }
  }

  return(max)
}

get_max(c(-1,3,2))
```

```
## [1] 2
```

Exercise:

Write your own version of `which.max()` function

Chapter 5

Lecture Notes

The lecture notes will be released one week before each class. Please do not circulate without permission from the module leader. To use the lecture notes outside MSIN0094, please contact the module leader (wei.miao@ucl.ac.uk)