R: Introduction

Sakol Suethanapornkul

28 October 2020

Quiz

Use the following vectors to answer the questions below:

```
scores <- c(3, 12, 8, 2, 4, 11, 15, 19, 3, 7, 6, 9)

answer <- c("Yes", "No", "Yes", "Yes", "No", "Yes",
            "No", "No", "No")
```

[1] Check if the values of `scores` exceed 10:

[2] Calculate the mean of `scores`:

[3] Convert the values in `answer` to numbers (Y = 1; N = 0):

Quiz

Use the following dataframe to answer the following questions:

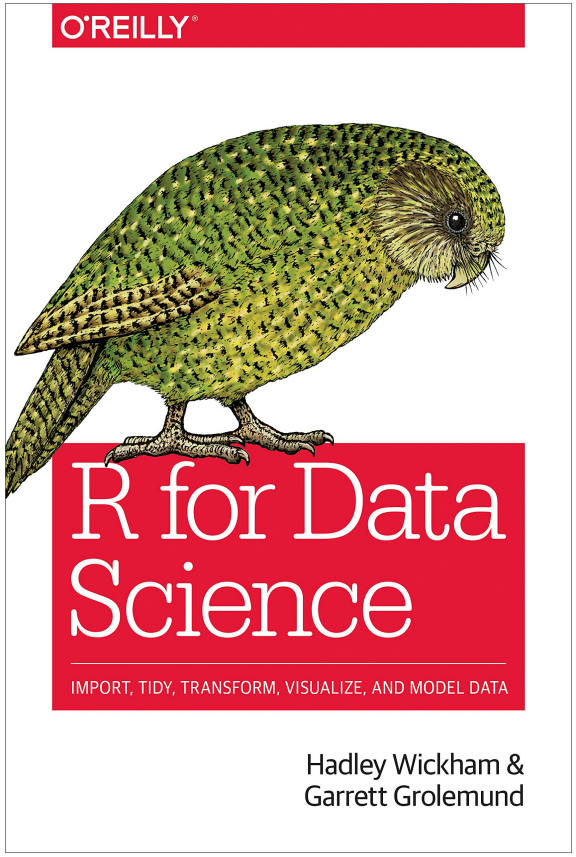
```
##   Score Gender Class
## 1    23   male     B
## 2    23   male     B
## 3    30 female     B
## 4    17 female     A
## 5    24 female     B
## 6    37   male     C
## 7    26 female     C
## 8    29 female     B
```

[4] Find scores from female students

[5] Find students from class C only

Introduction

We will use quite a lot of materials from this book:



(Image credit: Amazon.com)

Introduction

- What is R?
  - R is a language and environment for **statistical computing and graphics** (r-project.org)
  - R is a **programming language** and free software environment for statistical computing and graphics (Wikipedia)

Introduction

- What is R?
  - R is a language and environment for **statistical computing and graphics** (r-project.org)
  - R is a **programming language** and free software environment for statistical computing and graphics (Wikipedia)
- Why should I learn R?
  - With R, we can turn raw data into understanding, insight, and knowledge (R for Data Science)
  - R is an open-source software, which means it is absolutely free
  - R works across different platforms (Windows, Mac, Linux)

Installing R

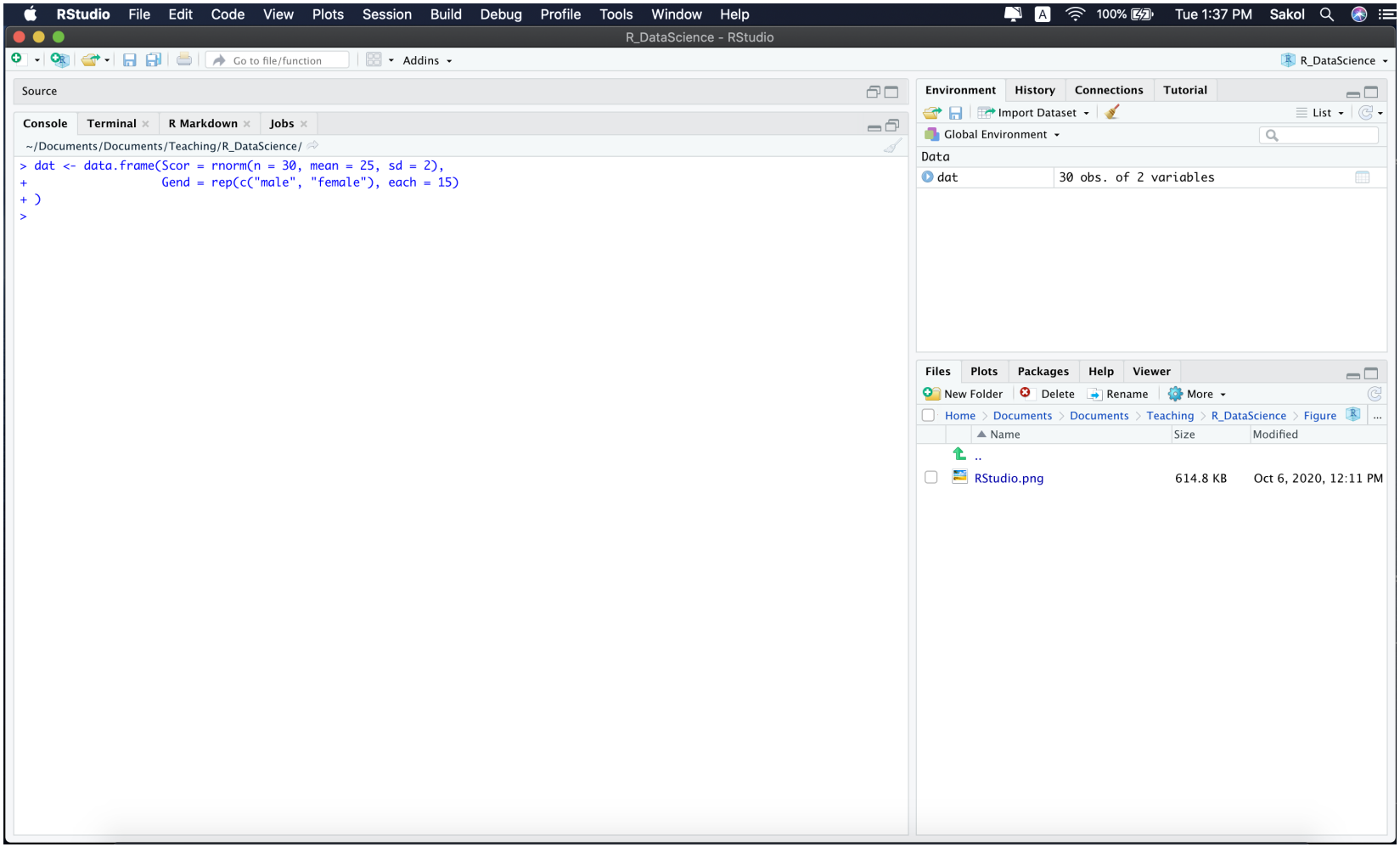
- Visit Comprehensive R Archive Network (CRAN) to download R
- We must have R installed for R Studio to work

What is RStudio?

- RStudio is an integrated development environment (IDE) for R
- RStudio is user-friendly, making it easier for us to interact with R with things like drop-down menus, etc.
- RStudio can be downloaded from this site

RStudio

When we start RStudio, this is what we see:



Part I: Basic Math

Four basic arithmetic operators:

- addition (+), subtraction (-), division (/), and multiplication (*)

```
5 + 4 - 2
5 - 4 + 2

5 + 4 * 2
5 / 4 + 2
```

Part I: Basic Math

The order of operations is ***PEMDAS***: Use parenthesis & space to make clear the order

```
(5 + 4) * 2
(5 + 4) * (2 + 5)

(5 + (4 / 2)) + (2 * 3)
```

EXERCISE: What are these two symbols for: ^ and %%?

```
2 ^ 3
3 ^ 4


5 %% 2
8 %% 3
```

ANSWER: ^ (raise x to the power of y) and %% (returns remainder of division)

```
2 ^ 3     # 2 * 2 * 2
3 ^ 4     # 3 * 3 * 3 * 3

5 %% 2    # 5 / 2 -->  4 + 1 / 2
8 %% 3
```

Part I: More math

There are a few other functions, which we will return to later on:

```
sum()
mean()
sd()

min()
max()

log()
sqrt()
exp()
```

Part II: Vector & Assignment

Vectors are the most basic data objects in R:

```
numbers <- c(1, 3, 5, 8, 4, 9, 2)
names   <- c("Jack", "James", "Jill", "Alix")
answers <- c(TRUE, FALSE, TRUE, TRUE, TRUE, FALSE)
```

NOTE:

- `<-` is an assignment symbol; shortcut = **Option + minus**
- `c()` is short for combine

When we create a vector, we tell R that the name `numbers` refers to an object, which is `c(1, 3, 5, 8, 4, 9, 2)`

Part II: Vector & Assignment

We can use a colon operator (:) to create sequences

```
num1  <- 1:10
num2  <- c(1:5, c(12, 16, 19), 30:40)
```

Part II: Numeric Vectors

For now, we will focus on numeric vectors (numbers are doubles by default[1]).

```
num1  <- c(1, 4, 12, 8, 6, 5)
num2  <- c(2, 5, 9, 11, 2, 3)
```

EXERCISE: What will happen when `num1 + num2`?

[1] Doubles are floating-point numbers and thus are approximations

ANSWER: Arithmetic operations in R are vectorized (i.e., operating on a vector of elements):

```
num1
```

```
## [1]  1  4 12  8  6  5
```

```
num2
```

```
## [1]  2  5  9 11  2  3
```

```
num1 + num2
```

```
## [1]  3  9 21 19  8  8
```

Part II: Numeric Vectors

EXERCISE: What will happen if we sum these two vectors together?

```
num1  <- c(11, 17, 23, 25, 26, 19)
num2  <- c(6, 13)
```

ANSWER: R uses a recycling rule: a shorter vector is expanded to be of the same length as a longer one

```
num1
```

```
## [1] 11 17 23 25 26 19
```

```
num2
```

```
## [1]  6 13
```

```
num1 + num2
```

```
## [1] 17 30 29 38 32 32
```

EXERCISE: What will happen in this case?

```
num1  <- c(11, 17, 23, 25, 26, 19, 21)
num2  <- c(6, 13)

num1 + num2
```

Part II: Numeric Vectors

Two key properties of vectors: *type* and *length*

```
nums  <- c(3, 5, 11, 13)

typeof(nums)
length(nums)
```

We will get to use one of the built-in functions, `length()`, in many different scenarios.

Part II: Numeric Vectors

EXERCISE: How many kinds of values can numeric vectors take?

```
nums  <- c(-2, 0, 2, 4, NA)

nums/2

nums/0
```

Part II: Numeric Vectors

Why are we wasting our time on numeric vectors? Well, because they are useful if we want to talk about **comparison operators** in R:

- == (equal), != (not equal), >, >=, <, <=

We can check if elements inside a vector meet certain conditions

```
nums  <- c(2, 4, 7, 9, 11, NA)

nums  == 11
nums  != 11
```

EXERCISE: What do you think are the answers to the following codes:

```
nums  <  9
nums  <= 9
```

We can apply these operators to solve some interesting problems:

```
(nums %% 2) == 0
(nums %% 2) != 0
```

This whole comparison operations will be of limited use if we have to conduct each "test" one by one. But luckily:

```
nums >  5  &  nums <  10
nums <= 7  |  nums >  10
nums <= 7  |  nums != 11
nums <= 10 & !(nums %% 2 == 0)
```

EXERCISE: What do we get from performing these operations? What does R return?

```
nums <- c(11, 6, 18, 21, 24, 18, 5, 9, 14)
nums >= 10
```

Well, you may find this a bit more familiar:

```
test <- nums >= 10
typeof(test)
```