

PUBH732 Weeks 6-9

R Programming

Emma Wei Chen, Ph.D.
(wei.chen@ku.ac.ae)

2025-02-17

Lecture 1

Intorduction to R

Outline

Topics:

- Why use R for data analysis?
- Setting up the environment
- Basic syntax and functionality of R
- Loading datasets
- Running simple scripts

Outline

Class Activities:

- Running basic commands in R
- Create a data frame
- Loading datasets
- Writing and executing a small script

What tools did you use before?

What tools did you use before?

List of Other Statistical Tools

- SPSS
- MATLAB
- STATA

Why use R?

- free of cost;
- open access;
- a large number of up-to-date packages;

Textbook

Huynh, YW (2019) R for Graduate Students.

https://bookdown.org/yih_huynh/Guide-to-R-Book/

- Other Materials

Batra, Neale, et al. The Epidemiologist R Handbook. 2021.

DOI [10.5281/zenodo.4752646](https://doi.org/10.5281/zenodo.4752646)

<https://www.epirhandbook.com/en/> (Advanced with a focus on epidemiology)

Setting up the environment

Components

- R
- RStudio

Installation

<https://posit.co/download/rstudio-desktop/>

Setting up the environment

- Creating a New Project (.Rproj)
- Creating a Script (.R)
- Layout of RStudio Interface
 - Environment
 - Script
 - Console
 - Package
 - Working Directory

Installing and loading packages

- What are packages?
 - A collection of free R tools that an R User wrote.
 - Packages are free but must be installed to your computer first. After installation, packages must be loaded into your RStudio library each time RStudio is opened/launched.
 - `install.packages()`
 - `library()`

Installing and loading packages

- Different packages will provide different sets of tools, though some tools may have overlapping functions. In this case, the later loaded function will mask the earlier one.
 - e.g. `select()` is in both *MASS* and *dplyr*. Use `dplyr::select()` to specify.
- How do you know which package/function to use

Useful packages

- Useful packages:
 - “*tidyverse*”:
 - This package is actually comprised of multiple packages: graphing (*ggplot2*) and user-friendly formatting (*dplyr*).
 - “*writexl*”: To produce Excel files from R data.
 - “*readxl*”: To load my Excel files into R.
 - “*utils*”: To read or write CSV files

Basic syntax

- R has all of the basic arithmetic operations available (+, -, /, *, ^) and can function as a calculator. However, R is also a powerful tool for managing our data.
- Frequently used operational symbols:
 - ==, !=, >, >=, <, <=
 - %in% (similar to == but much more useful)
 - |, &
- Missing Value: NA

Executing code

```
1 ## Example 1  
2 1 + 2 + 3 + 4
```

[1] 10

```
1 ## Example 2  
2 1 + 2 + 3 + # What happens if only select this line  
3 4
```

[1] 10

```
1 ## Example 3  
2 2 %in% c(1, 2, 3, 4)
```

[1] TRUE

```
2 1 + 2 + 3 + NA
```

```
[1] NA
```

Executing code

- Note the behavior of logic operations when NA is involved.

```
1 ## Example 5  
2 TRUE | FALSE
```

```
[1] TRUE
```

```
1 ## Example 6  
2 TRUE | NA
```

```
[1] TRUE
```

```
1 ## Example 7  
2 FALSE | NA
```


[1] NA

Executing code

- Note the behavior of logic operations when NA is involved.

```
1 ## Example 8  
2 TRUE & NA
```

```
[1] NA
```

```
1 ## Example 9  
2 FALSE & NA
```

```
[1] FALSE
```

Using objects

- Assigning values to objects (<-)
- What's the difference between <- and = in R?
 - <- is used for general assignments to object
 - = is used for parameters in functions

Using objects

```
1 ## Example for using <-  
2 x <- c(1, 2, 3, 4)  
3 print(x)
```

```
[1] 1 2 3 4
```

```
1 y <- c(1, 2, 3, 4, NA)  
2 ## Example for using =  
3 mean(y, na.rm = FALSE)
```

```
[1] NA
```

```
1 mean(y, na.rm = TRUE)
```

```
[1] 2.5
```

Using objects

- Why you want to use objects
 - save information in an object so you don't need to type them in again
 - to apply functions on the object
- Most commonly used object type in R: **data frame**
 - `data.frame()` is the function to create a new data frame

Data frame

```
1 ## Example for create a data frame
2 df <- data.frame(
3   Height = c(150, 60, 145, 187, 175),
4   Weight = c(55.5, 60.2, 62.3, 50.0, 48.1)
5 )
6
7 print(df)
```

	Height	Weight
1	150	55.5
2	60	60.2
3	145	62.3
4	187	50.0
5	175	48.1

Data frame

- Similar as a table.
- Each column can have a different data type
 - Commonly used data types: logical, integer, numeric, character, factor

```
1 ## Example for logical data type
2 Smoker <- c(TRUE, FALSE, TRUE, FALSE, FALSE)
3
4 ## Example for integer data type
5 ID <- c(1, 2, 3, 4, 5, 6, 7, 8)
6
7 ## Example for numeric data type
8 df$Height
```

```
[1] 150  60 145 187 175
```

Data frame

- Note the difference between character and factor

```
1 ## Add another column to df
2 df$City = c("Abu Dhabi", "Dubai", "Dubai", "Abu Dhabi", "Al Ain")
3 df$City_f = factor(df$City, levels = c("Abu Dhabi", "Dubai", "Al Ain"))
4 summary(df)
```

Height	Weight	City	City_f
Min. : 60.0	Min. :48.10	Length:5	Abu Dhabi:2
1st Qu.:145.0	1st Qu.:50.00	Class :character	Dubai :2
Median :150.0	Median :55.50	Mode :character	Al Ain :1
Mean :143.4	Mean :55.22		Sharjah :0
3rd Qu.:175.0	3rd Qu.:60.20		
Max. :187.0	Max. :62.30		

Data frame

- Each row represents a single observation across all variables.
- Columns have names (variable names), and rows can optionally have names.
- You can access elements using `df[row, column]` notation.

Data frame

```
1 # ## Add another column with less observations
2 # df$Gender = c("F", "M", "M")
3 #
4 # ## Add another column with more observations
5 # df$Gender = c("F", "M", "M", "F", "F", "M")
6
7 ## Add another column with the same number of observations
8 df$Gender = c("F", "M", "M", "F", "F")
9
10 ## Add row names
11 rownames(df) <- c(1, 2, 3, 4, 5)
12 print(df)
```

	Height	Weight	City	City_f	Gender
1	150	55.5	Abu Dhabi	Abu Dhabi	F
2	60	60.2	Dubai	Dubai	M
3	145	62.3	Dubai	Dubai	M

4	187	50.0	Abu Dhabi	Abu Dhabi	F
5	175	48.1	Al Ain	Al Ain	F

Datasets

- Build-in Datasets
 - `data()`
- Datasets from R Packages. Below are PH&E related packages offers practice datasets:
 - *pubh* [link](#)
 - *epi* [link](#)
- External data files
 - `read_xlsx()` from *readxl* package
 - `read.csv()` from *utils* package

Writing a simple script

- Header and Comment Organization
- Naming (Note: R is case-sensitive!)
 - Choose Descriptive and Concise Names
 - Do Not Start with a Number or Symbol
 - Avoid using punctuation other than a period (.), dash (-), or underscore (_)
 - No Spaces
 - Consistency
 - Avoid Names Already in Use

Writing a simple script

- A few naming styles
 - Snake case (snake_case), e.g., group_mean
 - Camel case (camelCase), e.g., groupMean
 - Dot-separated (dot.case), e.g., group.mean
 - This is not recommended because dot is used in some R functions, e.g. data.frame()
 - Uppercase (UPPER_CASE), e.g., GROUP_MEAN

Writing a simple script

- My personal style
 - I name key objects using Title Snake Case, e.g., Total_Count
 - I name temporary objects using snake case, e.g., city_tbl

End of the current lecture

Any questions/feedback?

Class activities:

- Practice basic commands in R
- Create a data frame that include at least 3 different types of data: logical, integer, factor
 - apply function `summary()` for your data frame to see what happens
- Loading datasets
 - try different ways to load datasets
- Writing and executing a small script
 - `source()`

Before Next Class

- Read “II Tidyverse” on the [R for Graduate Students](#)