Sprint 03

Tags	R Programming
Status	Done

😎 R CODE

▼ R101 / Getting Started with R

Download

Core - Base R : https://cran.r-project.org/bin/windows/base/

IDE - RStudio Desktop : https://posit.co/download/rstudio-desktop/

RStudio Cloud : https://posit.cloud/content/yours?sort=name_asc

Why R?

R คือภาษาที่เกิดมาสำหรับงาน data science โดยเฉพาะ เราเรียก R ว่า "**Fast data crunching tool**" คือ ดาวน์โหลดมาปุ๊ป เปิดขึ้นมาเขียน ก็ทำงานกับ data ได้เลย (native support) และที่สำคัญที่สุดคือ free ด้วย

🌵 **Core Building Blocks** ของ Programming Languages ทุกภาษาจะแบ่งได้เป็น 5 เรื่อง

- Variables
- Data Types
- Data Structures
- Control Flow [if, for, while]
- Functions

ไม่ว่าโปรแกรมที่เขียนจะซับซ้อนแค่ไหน สุดท้ายคือ 5 เรื่องนี้ประกอบร่างรวมกัน ถ้าเข้าใจ ก็เขียนได้ทุกภาษาเลย พอเราเขียนภาษาแรกได้แล้ว ภาษาต่อๆไปก็จะง่ายขึ้น

Comparison Operators

เราสามารถเขียน comparison operators ต่อไปนี้เพื่อเปรียบเทียบสองฝั่งของสมการได้ใน R

- `>`

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

🌵 Note - เราใช้เครื่องหมาย **double** equal signs `==` เพื่อเทียบสองฝั่งของสมการ ถ้าใช้ **single** equal sign เช่น `x = 5` จะเป็นการประกาศตัวแปร เพราะใน R เราสามารถสร้างตัวแปรได้สองแบบคือ `<-` หรือ `=`

Data Types in R

ความรู้พื้นฐานเกี่ยวกับ data types คือสำคัญขั้นสุดตอนเขียนโปรแกรม ภาษาคอมพิวเตอร์หลายๆภาษาจะ strict เรื่อง **data types** มากๆ ใน R ก็เช่นเดียวกัน โดย common data types ที่ data analyst เราต้องใช้งานเป็นประจำจะมีอยู่ 5 ประเภทคือ

- `numeric`
- `character`
- `logical`
- `factor` อันนี้คือตัวแปร categorical ในทางสถิติ
- `date`

Data Structure

- Vectoc
- Matrix
- List
- DataFrame

Data Frame

The Importance of Data Frame

🌵 ความรู้เรื่อง **Data Frame** คือพื้นฐานสำคัญของการทำงานเป็น **Data Analyst** งานของเรามากกว่า 80% ทำงานกับ structured data เช่น ข้อมูลใน Excel/ Google Sheets และที่เราดึงออกมาจาก SQL databases

Note : Data Frame คือ การเก็บข้อมูลให้อยู่ในรูปของ Data structure ที่คล้ายกับ Excel , Database

DataFrame เป็นโครงสร้างข้อมูลที่จัดระเบียบข้อมูลลงในตารางแถวและคอลัมน์แบบ 2 มิติ เหมือนกับสเปรดชีต

```r
friends <- c("Wan", "Ink", "Aan",
             "Bee", "Top")

ages <- c(26, 27, 32, 31, 28)

locations <- c("New York", "London",
               "London", "Tokyo",
               "Manchester")

movie_lover <- c(TRUE, TRUE, FALSE,
                 TRUE, TRUE)

## create dataframe from list
my_list <- list(friends = friends,
                ages = ages,
                locations = locations,
                movie = movie_lover)

data.frame(my_list)
```

```
> data.frame(my_list)
  friends ages  locations movie
1     Wan   26   New York  TRUE
2     Ink   27     London  TRUE
3     Aan   32     London FALSE
4     Bee   31      Tokyo  TRUE
5     Top   28 Manchester  TRUE
>
```

Subset

Subset ด้วย position

```r
friends <- c("Folk", "Mameaw", "Soobin", "Yeonjun", "Beomgyu")
```

```
> friends[1]
[1] "Folk"
> friends[2]
[1] "Mameaw"
> friends[1:3]
[1] "Folk"   "Mameaw" "Soobin"
> friends[3:4]
[1] "Soobin"  "Yeonjun"
> friends[c(2,4,5)]
[1] "Mameaw"  "Yeonjun" "Beomgyu"
```

```
> df[2,4]
[1] TRUE
> df[1,3]
[1] "Bangkok"
>
> df[1:2,2]
[1] 22 23
> df[2:3,1:2]
  friends ages
2  Mameaw   23
3  Soobin   23
>
> df[,"friends"]
[1] "Folk"    "Mameaw"  "Soobin"  "Yeonjun" "Beomgyu"
> df[,c("friends", "ages")]
  friends ages
1    Folk   22
2  Mameaw   23
3  Soobin   23
4 Yeonjun   24
5 Beomgyu   22
```

Subset ด้วย conditions

```
ages <- c(22, 23, 23, 24, 22)
```

```
> ages[ages <= 23]
[1] 22 23 23 22
> ages[ages >23]
[1] 24
> ages[ages >= 23]
[1] 23 23 24
```

```
> df[ df$anime_lover == TRUE, ]
  friends ages  locations anime_lover
1    Folk   22    Bangkok        TRUE
2  Mameaw   23     Phuket        TRUE
3  Soobin   23 Sangnok-gu        TRUE
>
> df[ df$ages < 24, ]
  friends ages  locations anime_lover
1    Folk   22    Bangkok        TRUE
2  Mameaw   23     Phuket        TRUE
3  Soobin   23 Sangnok-gu        TRUE
5 Beomgyu   22      Daegu       FALSE
>
> df[ df$friends == "Yeonjun"]
  anime_lover
1        TRUE
2        TRUE
3        TRUE
4       FALSE
5       FALSE
> df[df$friends == "Yeonjun", 3]
[1] "Seoul"
> df[ df$friends == "Yeonjun", 1:3]
  friends ages locations
4 Yeonjun   24     Seoul
```

Note : df[name_df $ name_col (condition), from_col)

Subset ด้วย name

```
> ages
[1] 22 23 23 24 22
> names(ages) <- friends
> ages
   Folk  Mameaw  Soobin Yeonjun Beomgyu
     22      23      23      24      22
>
> ages["Folk"]
Folk
  22
> ages["Folk", "Mameaw"]
Error in ages["Folk", "Mameaw"] : incorrect number of dimensions
> ages[c("Folk", "Soobin", "Beomgyu")]
   Folk  Soobin Beomgyu
     22      23      22
```

▼ R102 / Control Flow and Function

Common Errors

1. เขียน program ยังไม่จบ เห็นเป็นเครื่องหมาย `+` ใน console

```
> print("Rattapol Pet-in"
+
+
+
+ )
[1] "Rattapol Pet-in"
```

Note : ปัญหานี้แก้ได้ง่ายๆด้วยการกด `ESC` หรือไล่ดูโค้ดว่าเราลืมปิด `)` หรือ `}` หรือเปล่า (กด ESC คือง่ายสุด เริ่มเขียนใหม้ได้เลย)

2. Object 'xxx' not found

เราเรียกใช้ตัวแปร variable ที่ยังไม่ได้ประกาศขึ้นมา หรือเขียนชื่อตัวแปรผิด ในตัวอย่างนี้แอดลองเรียกตัวแปร `income` แต่เจอ error `income` not found ให้ลองเช็คใน environment หรือประกาศค่าตัวแปรใหม่ก่อน

```
> income
Error: object 'income' not found
>
> income <- 50000
> income
[1] 50000
```

3. Could not find function "xxx"

คล้ายๆกับปัญหาที่สอง คือ R หาชื่อ `function` นี้ไม่เจอ

วิธีแก้ให้ลองดูชื่อ function อีกทีว่าเราเขียนถูกหรือเปล่า หรือเรียกใช้ `library()`

```
> hello()
Error in hello() : could not find function "hello"
```

Note : 🌵 **library** คือ code ที่ developers คนอื่นพัฒนาไว้แล้ว และเราขอยืมมาใช้ในโปรแกรมที่เราเขียน ถ้าเราต้องการใช้ function ใน **library** นั้นๆเช่น dplyr, ggplot2, tidyverse เราต้องเรียก library นั้นขึ้นมาก่อนทุกครั้งที่เปิด RStudio

What is control flow?

Control Flow คือหนึ่งใน building blocks ที่สำคัญของการเขียนโปรแกรม (ความรู้ในบทเรียนนี้ใช้ได้กับทุกภาษาเลยนะครับ) ใน R เรามี control flow สำคัญอยู่สามตัวคือ

1. `if`
2. `for`
3. `while`

Note - เวลาเราพิมพ์ keyword `if` `for` `while` ใน script จะมี highlight syntax ให้เราด้วย

🦊 หน้าที่ของ **Control Flow** คือการควบคุมพฤติกรรมของโปรแกรมที่เราเขียน ตัวอย่างเช่น

```
score <- 85
if (score >= 80) { print("passed") } else { print("failed") }
```

ถ้าคะแนนสอบมากกว่าหรือเท่ากับ 80 คะแนน สอบผ่าน "passed" แต่ถ้าคะแนนไม่ถึงเกณฑ์คือสอบตก "failed"

Function Parameter & Argument

🌵 คำศัพท์ที่เราใช้ในการเขียน `function` มีอยู่สองคำ (concept นี้ใช้ได้กับทุกภาษาเลย)

- `parameter`
- `argument`

Parameter คือชื่อ named input ที่เราใส่ใน function ส่วน **Argument** คือ actual value ที่เราใส่ไปใน parameter นั้นๆตอนเรารัน function

ตัวอย่างการเขียน parameter และ argument ใน R

```
greeting() <- function( name = "Folk" ) { print(paste("Hi!", name)) }
```

- name คือ parameter
- "Folk" คือ argument

apply() coolest function

🌵 ใน R มีฟังก์ชัน `apply()` ที่เราใช้แทนการเขียน `loop` ได้ทั้ง columns, rows สะดวกมาก

อ่านเพิ่มเติมเกี่ยว
กับ `apply()` ได้ที่ https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/apply

🐼Hands-On Programming with R (Full version)

Hands-On Programming R (Distill Version)

▼ R103 / Working with Data

Working with Data in R

- R คือภาษาที่เราเรียกว่า fast data crunching ที่ถูกพัฒนาสำหรับงาน data โดยเฉพาะ คือเปิดโปรแกรมมาก็เริ่มทำงานได้เลย

📥 How to install R packages

ติดตั้ง packages ได้สองวิธีใน RStudio (Desktop/ Cloud ใช้งานเหมือนกัน)

1. เขียนฟังก์ชัน `install.packages()` ติดตั้งเอง
2. ไปที่ tab `Packages` ที่หน้าจอขวาล่างของ RStudio แล้วกดปุ่ม `install`

Note - คำว่า package/ library สองคำนี้ความหมายเดียวกัน

Note - package/ library คือ code หรือ program ที่ programmer คนอื่นเขียนเอาไว้แล้วๆ เรายืมมาใช้

⭐ ดูรายชื่อ packages ทั้งหมดใน **CRAN R** ได้ที่ลิ้งนี้

Join function in R

สามารถเขียน `join` ได้หลายแบบเลย มากกว่าที่มีใน standard SQL

- `inner_join`
- `left_join`
- `right_join`
- `full_join`
- `anti_join`
- `semi_join`
- cross join (ด้วยฟังก์ชัน `tidyr::crossing()`)

SQLite

How to read data from SQLite database

ใน R เราใช้ library `RSQLite` เพื่อจัดการกับข้อมูลใน sqlite .db file ขั้นตอนการทำงานกับ database จะแบ่งเป็น 3 steps ง่ายๆ

1. connect to database i.e. open connection
2. get data (with SQL)
3. disconnect from database i.e. close connection

🌵 อ่านเพิ่มเติมเรื่อง database ใน R ได้ที่ https://dbi.r-dbi.org/

Save Data in R

🌵 เราสามารถ save data ใน R ได้สองแบบ และสามารถโหลดข้อมูลกลับมาใช้ได้อีกครั้งในอนาคต

- `save.image()` ใช้ save objects ทั้งหมดที่อยู่ใน environment (i.e. workspace) ของเราเข้าไปที่ไฟล์ `.RData`
- `saveRDS()` ใช้ save single object แค่ไฟล์เดียวที่ไฟล์ `.rds`

ถ้าต้องการโหลด .RData ให้ใช้ฟังก์ชัน `load()` หรือถ้าอยากจะโหลด .rds ให้ใช้ฟังก์ชัน `readRDS()`