R CODE

▼ R101 / Getting Started with R

Create and Remove variables

```
## create variables
income <- 50000
expense <- 30000
saving <- income - expense

## remove variables
rm(saving)
```

Compare Values

```
## compare values
1 + 1 == 2
2 * 2 <= 4
5 >= 10
```

```
5 - 3 != 3
10 < 9
20 > 9

## compare texts/ character
"Hello" == "hello"
"Folk" == "Folk"
'Mameaw' == 'Mameaw'
```

Data Type

```
## Data Types

## numeric
age <- 22
print(age)
class(age) # check data type

## character
my_name <- "Folk"
my_university <- 'Kmitl'
print(my_name)
print(my_university)
class(my_name); class(my_university)

## logical
result <- 1 + 1 == 2
print(result)
class(result)

## factor
animals <- c("dog", "cat", "panda", "pig", "panda", "elephant
print(animals)
class(animals)
```

```
animals <- factor(animals)
print(animals)
class(animals)

## date/time
time_now <- Sys.time()
class(time_now)
```

Convert Data Types

```
## Convert Data Types

## main functions
## as.numeric()
## as.character()
## as.logical()

x <- 100
class(x)

char_x <- as.character(x)
char_x
num_x <- as.numeric(char_x)
num_x

## logical: TRUE(1)/FALSE(0)
as.logical(0)
as.logical(1)
as.numeric(TRUE)
as.numeric(FALSE)
```

Vector

```
## Vector -- การสร้างชุดข้อมูลที่เรียงต่อกัน และเก็บข้อมูลได้แค่ 1 ประเภทเท่านั้น
1:10
20:3

## sequence generation
seq(from = 1, to = 100, by = 5)

## help("name functions")
help("seq")

## function c
friends <- c("Folk", "Panda", "Elephant")
age <- c(22, 23, 24)
is_male <- c(TRUE, FALSE, TRUE)
```

Matrix

```
## Matrix
x <- 1:25
length(x)
dim(x) <- c(5,5) # การสร้าง matrix จาก dim

m1 <- matrix(1:25, ncol = 5) # การสร้าง matrix จาก matrix funct
m2 <- matrix(1:6, ncol = 3, nrow = 2, byrow = T)

## element wise computation
m2 + 100
m2 * 4
```

Note : ncol คือ จำนวน column, nrow คือ จำนวน row, byrow คือ การเรียงลำดับจาก row

List

```
## List -เก็บข้อมูลได้หลายชนิด
my_name <- "Folk"
my_pet <- c("cat", "dog", "panda")
m1 <- matrix(1:20, ncol = 5)
mameaw_is_cute <- TRUE

my_list <- list(item1 = my_name,
                item2 = my_pet,
                item3 = m1,
                item4 = mameaw_is_cute)
my_list$item1
my_list$item2
```

Note : เครื่องหมาย “$” การบอกว่าเราจะดึงข้อมูลชุดไหนจาก list ที่เราเลือก

Data Frame

```
## Data Frame

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

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

locations <- c("Bangkok", "Phuket", "Sangnok-gu", "Seoul", "D

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

## 1 create data frame from list
my_list <- list(friends = friends,
                ages = ages,
                locations = locations,
                anime = anime_lover)
data.frame(my_list)
```

```
## 2 create data frame from data.frame
df <- data.frame(friends,
                 ages,
                 locations,
                 anime_lover)

View(df)
```

Note : View เป็นฟังก์ชันที่แสดงผลข้อมูลออกมาในอีกหน้าต่างนึง

Subset

```
scores <- c(
    Folk = 90,
    Mameaw = 100,
    Soobin = 20,
    Yeonjun = 59,
    Jisu = 69
)
scores
scores["Soobin"]
scores[[3]]
```

Note : Subset เป็นการดึงข้อมูลออกมาจาก data structure

▼ R102 / Control Flow and Function

IF

```
## IF
## =IF() in google sheet

score <- 49

if (score >= 90){
```

```
  print("Passed")
} else {
  print("Failed")
}

if (score >= 90){
  print("Passed")
} else if (score >= 50){
  print("OK")
} else {
  print("Enroll again!!")
}
```

IFELSE

```
## IFELSE
ifelse(score >= 80, "Passed", "Failed")

ifelse(score >= 80, "Passed", ifelse(
      score >= 50, "OK", "Enroll again!"
))
```

Note :

🌵 `ifelse()` syntax เขียนเหมือนกับ `=IF()` ใน Google Sheets/ Excel

`ifelse(condition, TRUE, FALSE)`

FOR

```
## FOR
friends <- c("Soobin", "Beomgyu", "Mameaw", "Jisu")

for (friend in friends){
  print(paste("Hi!",friend)) # paste เป็นการเชื่อมข้อความ
}
```

```
for (num in nums){
  print(num - 2)
}

# Vectorization
paste("Hi", friends)

(nums <- c(4, 5, 9, 19))
(nums <- nums + 2)
 # การใส่วงเล็บเพิ่มจะทำให้หน้าต่าง console แสดงผลค่าออกมาเลยโดยไม่ต้องเขียนเพิ่ม
```

WHILE

```
## While loop
count <- 0

while (count < 5){
  print("Hi!")
  count <- count + 1
}
```

Create Our First Function

🌵 R ใช้ keyword `function` ในการประกาศ function ใหม่

```
# create first function

greeting <- function(){
  print("Hello World!")
}

greeting_name <- function(name) {
  print( paste("Hello!", name))
```

```
}

func <- function() {
  greeting()
  greeting_name("Folk")
}
```

Practice Practice Practice

🌵 มาลองฝึกเขียน 3 functions ใน R ด้วยกันครับ

- `add_two_nums()`
- `cube()`
- `count_ball()`

```
# add_two_nums() function
add_two_nums <- function(val1, val2){
  return(val1 + val2) # return is optional
}

# cube() function
cube <- function(base, power = 3){
  return(base ** power)
}

# count_ball() function
balls <- c("green","red","blue","blue","red","blue","purple")

count_ball <- function(balls, color){
  sum(balls == color)
}
```

Looping over a dataframe

```r
# loop over a dataframe
data() # ข้อมูล build in ที่มีอยู่ใน R

nrow(USArrests) # nrow : นับจำนวน row ของ data
ncol(USArrests) # ncol : นับจำนวน column ของ data
head(USArrests) # head : เป็น preview หน้าตาของ dataframe ออกมาแ

for (i in 1:ncol(USArrests)) {
  print( names(USArrests)[i])
  print( mean(USArrests[[i]]))
}

class(USArrests[1]) # data type : data.frame
class(USArrests[[1]]) # data type : numeric
# Note : [[]] จะได้ data type จริงๆของ column นั้นเลย
# data$col_name = data[["col_name"]] = data[[5]]

################################################
cal_mean_by_col <- function(df) {
  for (i in 1:ncol(df)) {
    print( names(df)[i])
    print( mean(df[[i]]))
  }
}
```

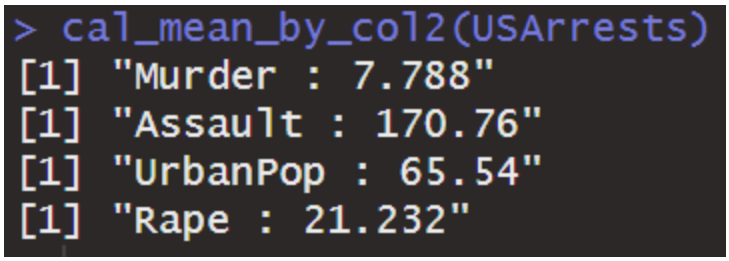
```
> cal_mean_by_col(USArrests)
[1] "Murder"
[1] 7.788
[1] "Assault"
[1] 170.76
[1] "UrbanPop"
[1] 65.54
[1] "Rape"
[1] 21.232
```

Refactoring your code

```
# refactor : การปรับให้ code เราอ่านง่ายขึ้น แสดงผลได้ดีขึ้น รันเร็วขึ้น แต่ผลรับ

cal_mean_by_col2 <- function(df) {
  col_name <- names(df)

  for (i in 1:ncol(df)) {
    avg_col <- mean(df[[i]])
    print(paste(col_name[i], ":", avg_col))
  }
}
```



Apply() coolest function

```
# apply function
avg_by_row_mtcars <- apply(mtcars, MARGIN = 1, mean)
avg_by_col_mtcars <- apply(mtcars, MARGIN = 2, mean)

apply(mtcars, MARGIN = 2, sum)
apply(mtcars, MARGIN = 2, sd)
apply(mtcars, MARGIN = 2, median)
apply(mtcars, MARGIN = 2, mean)
```

▼ R103 / Working with Data

How to install R packages

```
# install packages
install.packages(c("readr",
                   "readxl",
                   "googlesheets4",
                   "jsonlite",
                   "dplyr",
                   "sqldf",
                   "RSQLite"))

# load library
library(readr)
library(readxl)
library(googlesheets4)
library(jsonlite)
library(dplyr)
library(sqldf)
library(RSQLite)
```

Text File

```
student <- read_table("student.txt")
```

Note : เป็นฟังก์ชันที่จะอ่านไฟล์ข้อมูลที่ถูกแยกกันด้วย whitespace ให้กลายเป็น data frame

CSV

```
student2 <- read_csv("student.csv")
```

🌵 `read_csv()` ใช้อ่านไฟล์ .csv comma separated values เป็น common data format ที่ data analyst เราใช้กันเป็นประจำ

Excel File

```
result <- list()

for (i in 1:3) {
  result[[i]] <- read_excel("students.xlsx", sheet=i)
}

result[[1]]
result[[2]]
result[[3]]
```

🌵 `read_excel()` ใช้อ่านไฟล์ Excel .xlsx อ่านทีละ sheet ได้เลย

Google Sheets

🌵 `read_sheet()` ใช้อ่าน data จาก Google Sheets ถ้าเป็น public link ก่อนใช้คำสั่ง read_sheet() ให้เรารันคำสั่ง `gs4_deauth()` ก่อนนะครับ (แปลว่า ลิ้งที่เราจะอ่านไฟล์เป็น public link ไม่ต้องมีการ login)

อ่านเพิ่มเติมเรื่องการทำ authentication ของ Google Sheets ได้ที่ https://googlesheets4.tidyverse.org/articles/auth.html

```
gs4_deauth()
url <- "https://docs.google.com/spreadsheets/d/1Sg0ZjO6NRdjkq
df <- read_sheet(url, sheet="students")
```

JSON

🌵 `fromJSON()` ใช้อ่านไฟล์ .json เข้ามาเป็น list ใน R แล้วค่อยใช้ฟังก์ชัน `data.frame()` เพื่อเปลี่ยน list → data frame ได้

> JSON = JavaScript Object Notation

`Key` ใน JSON ต้องเขียนใน double quote เท่านั้นนะครับ ถ้าใช้ single quote จะ error

อ่านเพิ่มเติมเรื่อง jsonlite ได้ที่ https://cran.r-project.org/web/packages/jsonlite/index.html

```
library(jsonlite)

txt <- data.frame(fromJSON("tomorrowbytogether.json"))

View(txt)
```

Note : data type → logical ของ json ต้องเป็นตัวพิมพ์เล็ก เช่น true, false

Bind Rows (UNION ALL)

```
library(dplyr)
library(readxl)

# read excel file
econ <- read_excel("students.xlsx", sheet=1)
business <- read_excel("students.xlsx", sheet=2)
data <- read_excel("students.xlsx", sheet=3)

# bind_rows == SQL UNION ALL
list_df <- list(econ, business, data)
full_df <- bind_rows(list_df)

# loop
list_df2 <- list()

for (i in 1:3) {
  list_df2[[i]] <- read_excel("students.xlsx",sheet=i)
}
full_df2 <- bind_rows(list_df2)
```

🌵 `bind_rows()` เทียบเท่ากับการเขียน `UNION ALL` ใน SQL

Bind Cols (!= JOIN)

🌵 `bind_cols()` ไม่เทียบเท่ากับการเขียน `join` เพราะ bind columns ไม่จำเป็นต้องใช้ `key` ใดๆ แค่เอา data frame สองตัวมาวางต่อกัน ซ้าย-ขวา

```
# bind_cols() != join

df1 <- data.frame(
  id = 1:5,
  name = c("Yeonjun", "Soobin", "Beomgyu", "Yaehyun", "Kai")
)

df2 <- data.frame(
  id = 1:5,
  city = c( rep("BKK",3), rep("LONDON",2)),
  country = c( rep("TH",2), rep("US",3))
)

bind_cols(df1, df2)
left_join(df1, df2, by="id")
```

SQL

```
# load library sqldf
library(sqldf)
library(readr)

school <- read_csv("school.csv")

sqldf("select * from school;")

sqldf("select avg(student), sum(student) from school;")

sqldf("select school_id, school_name, country from school;")
```

```
sql_query <- "select * from school where country = 'USA';"
usa_school <- sqldf(sql_query)
```

🌵 `sqldf()` ใช้เขียน SQL เพื่อจัดการกับ dataframe ที่อยู่ใน R

SQLite

```
# load library
library(RSQLite)

# connect to SQLite database (.db file)
# 1. open connection
conn <- dbConnect(SQLite(), "chinook.db")

# 2. get data
dbListTables(conn)
dbListFields(conn, "customers")

df <- dbGetQuery(conn, "select * from customers where country
df2 <- dbGetQuery(conn, "select * from customers where countr

# 3. close connection
dbDisconnect(conn)
```

Save data in R

```
save.image(file = "data.RData")
load(file = "data.RData")

saveRDS(business, file = "business.RDS")
business <- readRDS("business.RDS")
```