1. Giới thiệu về R và RStudio

R là gì ?

- R là một môi trường thống kê và ngôn ngữ lập trình toàn diện để phân tích và trực quan dữ liệu một cách khoa học và chuyên nghiệp.
- R là sự kết hợp giữa ngôn ngữ lập trình và các gói lệnh thống kê.
- R có nguồn gốc từ ngôn ngữ lập trình S được phát triển từ năm 1988.

Các toán tử trong R:

Toán tử số học		Toán tử logic	
KÝ HIỆU TOÁN TỬ	MÔ TẢ	KÝ HIỆU TOÁN TỬ	MÔ TẢ
+	Phép cộng	>	Lớn hơn
-	Phép trừ	<	Nhỏ hơn
*	Phép nhân	>=	Lớn hơn hoặc bằng
/	Phép chia	==	Chính xác bằng
^ hoặc **	Luỹ thừa	!=	Không bằng

Các hàm toán học trong R:

log(x)	Log	**sum(x)**	Tổng
exp(x)	Hàm exp	**mean(x)**	Giá trị trung bình.
max(x)	Giá trị lớn nhất	**median(x)**	Trung vị
min(x)	Giá trị nhỏ nhất	**quantile(x)**	Phân vị x
round(x, n)	Làm tròn đến n chữ số.	**rank(x)**	Rank các giá trị của x
signif(x, n)	Làm tròn đến n chữ số dạng e	**var(x)**	Phương sai
cor(x, y)	Hệ số tương quan	**sd(x)**	Độ lệch tiêu chuẩn

RStudio là gì ?

- RStudio là một IDE (Interface Development Environment) của R. Chúng ta có thể chạy R thông qua RStudio (mà ko cần phải chạy R), nó cung cấp

các chức năng rất thuận tiện để chạy R. *(Hiểu nôm na là giống như C++ muốn code, build rồi run code thì cần một IDE cho nó ví dụ như Codeblock hay Visual Studio hoặc với Java thì là Eclipse thì R sẽ cần RStudio)*

2. Hướng dẫn cài đặt R và R Studio

Cài đặt R:

- Truy cập link: https://posit.co/download/rstudio-desktop/
- Chọn "Download and Install R"
- Chọn phiên bản phù hợp với hệ điều hành của máy tính
- Cài đặt R theo hướng dẫn *(Khuyến khích xài chung 1 phiên bản R-4.3.3)*

Cài đặt RStudio:

- Truy cập link: https://posit.co/download/rstudio-desktop/
- Chọn "Download RStudio Desktop for Windows" *(Nếu dùng hệ điều hành khác kéo xuống dưới chọn bản thích hợp với hệ điều hành đó)*
- Cài đặt RStudio *(Khuyến khích chung bản RStudio-2023.12.1-402.exe)*

3. Một số thao tác cơ bản trong RStudio:

- Link một số package phổ biến: https://support.posit.co/hc/en-us/articles/201057987-Quick-list-of-useful-R-packages
- Link Cheatsheet cho R: https://iqss.github.io/dss-workshops/R/Rintro/base-r-cheat-sheet.pdf

Cài đặt và load các packages:

- Tải Rtools: https://cran.r-project.org/bin/windows/Rtools/
- Nhấp chọn phiên bản phù hợp với phiên bản R hiện tại và cài đặt *(RTools 4.3 for R versions 4.3.x nếu ở trên cài bản R-4.3.3)*
- Cài đặt package: **install.packages("package_name")**
- Cài đặt nhiều packages: **install.packages(c("name_1", "name_2" , "..."))**
- Load packages: **library(package_name)**

Lệnh cơ bản:

- **setwd('path')**
- **getwd()**
- **list.files()**, **dir()**: liệt kê tất cả file trong thư mục
- **save.image('name.rda')**: lưu workspace
- **save(x, file='name.rda')**: lưu biến x

- **load('name.rda')**: load file .rda
- **rm(x)**: xóa biến x
- **rm(list=ls())**: xóa tất cả biến
- **ls()**: liệt kê tất cả biến
- **str(x)**: xem thông tin biến x

Thao tác cơ bản trên dataframe:

- Các hàm:

Filter: chọn hàng
Arrange: sắp thứ tự dòng
Select: chọn cột
Rename: đổi tên
Distinct: tìm các hàng riêng biệt
Mutate: thêm các biến mới
Summaries: tóm tắt trên một tập dữ liệu
Sample_n: lấy mẫu từ tập dữ liệu

Viết hàm:

```
# Function template
        my_function = function(input1, input2, … , inputN) # Define inputs
        {
            # Define 'output' using input1, input2, … , inputN
            return(output) # Return this output
}
```

Ví dụ:

```
        add_number = function(x, y) # Define inputs
        {
            z = x + y
            return(z)
}
```

Tạo và truy cập phần tử trong vector:

Trong R, bạn có thể sử dụng hàm c() để tạo một vector:

v <- c(1, 2, 3, 4, 5) # Tạo một vector với các giá trị từ 1 đến 5

Bạn có thể sử dụng chỉ số để truy cập vào các phần tử của vector:

v[1] # Truy cập phần tử đầu tiên của vector

v[2:4] # Truy cập phần tử từ thứ 2 đến thứ 4 của vector

Đọc dữ liệu:

- Đọc file ".csv" vào data frame (df):

  df <- read.csv(file, header, sep)

  file: đường dẫn tới file
  header: nếu TRUE, dòng đầu tiên được dùng như tên biến
  sep: ký tự phân cách các trường dữ liệu *(ít dùng)*

Vẽ biểu đồ: (dùng package ggplot2)

- Tạo một biểu đồ và định nghĩa các biến cần ánh xạ đến đồ thị: **ggplot(data = data_frame, aes(x = variable_1, y = variable_2))**
- Gắn đường thẳng: **ggplot(data = data_frame, aes(x = variable_1, y = variable_2)) + geom_line()**
- Gắn tựa đề biểu đồ và nhãn các trục: **ggplot(data = data_frame, aes(x = variable_1, y = variable_2)) + geom_line() + labs(title = "Title of Graph", x = "new x label", y = "new y label")**

Tham khảo vài đồ thị sẽ dùng:

1. Biểu đồ Histogram

hist(x) # x là vector chứa dữ liệu

2. Biểu đồ Scatter (biểu đồ phân tán)

plot(x, y) # x và y là hai vector chứa dữ liệu

3. Biểu đồ Line (biểu đồ đường)

plot(x, type = "l") # x là vector chứa dữ liệu

4. Biểu đồ Bar (biểu đồ cột)

barplot(x) # x là vector chứa dữ liệu

5. Biểu đồ Pie (biểu đồ tròn)

pie(x) # x là vector chứa dữ liệu

Ước lượng:

- **mean(data)** # Tính trung bình
- **median(data)** # Tính trung vị
- **sd(data)** # Tính độ lệch chuẩn
- **length(data)** # Cỡ mẫu
- **qnorm(1 – alpha/2)** # Tìm phân vị trong phân phối chuẩn.
- **pt(t, n – 1, lower.tail = FALSE)** # Tìm phân vị P(t(n – 1) > t) trong phân phối Student, lower.tail mặc định là TRUE. Nếu là TRUE, hàm sẽ tìm phân vị P(t(n – 1) < t).

Kiểm định một mẫu:

- **t.test(x, alternative = "…", mu, conf.level)**: kiểm định kỳ vọng, x là vector dữ liệu, alternative = "two.sides" là kiểm định 2 phía, "less" và "greater" lần lượt là H1 nhỏ hơn và H1 lớn hơn, nếu alternative trống thì mặc định là "two.sides", mu = mu0 là giá trị cần kiểm định, conf.level là độ tin cậy (nếu để trống mặc định 0.95).

- **result$statistic, result[['statistic']], unname(result[['statistic']]))**: cả 3 dùng để rút ra thống kê kiểm định với ***result = t.test(…)***.
- **prop.test(y, n, p = p0, alternative = "two.sides", conf.level = 0.95)**: y là số phần tử thỏa, n là cỡ mẫu, p0 là giá trị cần kiểm định.

Kiểm định hai mẫu:

- **var.test(x1, x2)**: kiểm định phương sai, nếu p-value < alpha thì phương sai khác nhau.
- **t.test(x1, x2, alternative = "two.sides", var.equal, conf.level = 0.95)**: kiểm định kỳ vọng cho hai mẫu độc lập, x1 x2 là hai mẫu dữ liệu, var.equal = FALSE nếu phương sai hai mẫu khác nhau, TRUE nếu như nhau, nếu để trống thì mặc định là FALSE.
- **t.test(data ~ group, alternative = "two.sides", var.equal, conf.level = 0.95)**: tương tự như trên nhưng trong trường hợp này, các cặp giá trị được chứa trong cùng 1 biến data và được phân biệt bởi biến group, thay vì x1 x2.
- **t.test(x1, x2, alternative = "two.sides", paired = TRUE, var.equal, conf.level = 0.95)** hoặc **t.test(data ~ group, alternative = "two.sides", paired = TRUE, var.equal, conf.level = 0.95)**: kiểm định kỳ vọng hai mẫu không độc lập.
- **prop.test(y = c(y1, y2), n = c(n1, n2), alternative = "…", conf.level, correct = TRUE)**: kiểm định tỷ lệ hai mẫu, y và n phải là các vector, y1 y2 là số phần tử thỏa ở mẫu 1 và 2, n1 n2 là cỡ mẫu ở mẫu 1 và 2.

LƯU Ý: khi dùng các hàm t.test, prop.test, ta thường so sánh p-value với alpha để kết luận, nếu p-value < alpha thì bác bỏ H0.

Hồi quy:

- **anova(modelName_1, modelName_2)** # So sánh 2 mô hình hồi quy
- **abline(lm(y~x)** #Vẽ đường thẳng hồi quy.
- **lm(y~x)** #Các giá trị cơ bản của phân tích hồi quy.

######### LƯU Ý: Trong quá trình làm BTL, có thể các lệnh trên sẽ không đủ để dùng hoặc thiếu các thông số cần truyền vào nên các bạn nên kiểm tra kĩ. ############

VÍ DỤ SỬ DỤNG CÁC LỆNH TRONG BTL CŨ: (THAM KHẢO)

Tiền xử lí dữ liệu

1 . Import file

All_GPUs <- read.csv("E:/Nam 2 DHBK/BTL XSTK/All_GPUs.csv", header=FALSE)

Gán đối All_GPUs cho hàm đọc file csv dùng để import dữ liệu. Sử dụng đối tượng All_GPUs lúc này như là toàn bộ dữ liệu trong file được import

2. Xem dữ liệu

View(All_GPUs)

Thể hiện dữ liệu All_GPUs dưới dạng bảng đầy đủ.

head(All_GPUs, số_dòng)

Thể hiển số_dòng đầu tiên của bảng All_GPUs, nếu không có tham số số_dòng thì mặc định số dòng là 6. Lệnh head được sử dụng thường xuyên hơn View do thông tin được thể hiện trực tiếp dưới console thay vì màn hình làm việc mới như View.

str(All_GPUs)

Thể hiện thông tin bảng dưới dạng liệt kê số chiều, số biến, kiểu dữ liệu của các biến trong bảng.

3. Làm sạch dữ liệu

- Trích ra 1 tệp tin con các biến sẽ sử dụng.

All_GPUs[3, 2]

Dữ liệu tại ô (3, 2) trong bảng All_GPUs

All_GPUs[,1:5]

Dữ liệu tại cột 1 tới 5

All_GPUs[,c("V1", "V2", "V3", "V4", "V5")]

Dữ liệu tại các cột V1, V2, V3, V4, V5

subs_data <- All_GPUs[,c("V1", "V2", "V3", "V4", "V5")]

Gán một tập dữ liệu con

- Kiểm tra dữ liệu khuyết và đề xuất cách xử lí khuyết.

is.na(All_GPUs)

Kiểm tra từng ô dữ liệu xem có ô nào bị đánh dữ liệu khuyết hay không ("N/A"), nếu không ô đó có giá trị FALSE, ngược lại là TRUE.

apply(is.na(All_GPUs), 2, which) hoặc which(is.na(All_GPUs)) hoặc sum(is.na(All_GPUs))

Kiểm tra có bao nhiêu ô có dữ liệu khuyết. Kết quả xuất ra số ô có dữ liệu khuyết.

apply(is.na(All_GPUs), 2, sum)

Kiểm tra số ô có khuyết dữ liệu trên từng cột. Output các cột và số ô bị khuyết dữ liệu.

- Kiểm tra định dạng biến

V1 <- as.character(subs_data$V1)

Gán V1 nhận array giá trị của cột V1 trong tập tin con

is.numeric(V1)

Kiểm tra biến V1 có phải dạng số không

- Xử lí ngoại lai (phụ)
- Tạo thêm biến

Mục 2. Thống kê và mô tả

summary(subs_data)

Tính thông kê đơn giản

```
hist(subs_data$TMUs, xlab = "Ten cot x", ylab = "Ten cot y", main = "Ten do thi", col = heat.colors(5), labels = T, ylim = c(0, 1500))
```

Vẽ đồ thị tần số với labels = T để hiển thị số cho mỗi cột, ylim để tăng giới hạn cột y