# GIỚI THIỆU NGÔN NGỮ R

Phân tích và xử lý số liệu là một trong những thao tác cần thiết và quan trọng đối với các nhà nghiên cứu trong nhiều ngành, như sinh học, địa lý, toán học,…Trước đây, các công ty phần mềm đã phát triển các phần mềm chuyên nghiệp như SPSS, Excel, Stata,… cho việc phân tích số liệu. Tuy nhiên, các phần mềm này đều là các phần mềm thương mại, có giá từ vài trăm đến vài nghìn USD, không phải trường đại học hay trung tâm nghiên cứu nào cũng có thể mua được. Do đó, trong khoảng mười năm lại đây, các nhà nghiên cứu thống kê trên thế giới đã tập hợp nhau lại và phát triển một công cụ theo hướng mã nguồn mở sao cho tất cả mọi người đều có thể sử dụng và hoàn toàn miễn phí. Công cụ này có tên là ngôn ngữ R, một trong những ngôn ngữ được giới nghiên cứu sử dụng nhiều nhất hiện nay.

Ở Việt Nam, việc sử dụng ngôn ngữ R vẫn còn mới mẻ, vì nhiều lý do. Trong tài liệu này, chúng tôi muốn cung cấp một cách nhìn tổng quan về ngôn ngữ R. Các nội dung chuyên sâu hơn sẽ được cung cấp trong thời gian tới.

## 1. Tổng quan về ngôn ngữ R

Nói một cách ngắn gọn, R là một phần mềm sử dụng cho phân tích thống kê và đồ thị. Thật ra về bản chất, R là ngôn ngữ máy tính đa năng, có thể sử dụng cho nhiều mục tiêu khác nhau, từ tính toán đơn giản, toán học giả trí, tính toán ma trận, đến các phân tích thống kê phức tạp. Vì là một ngôn ngữ cho nên người ta có thể sử dụng R để phát triển các thành phần mềm chuyên môn cho một vẫn đề tính toán cá biệt.

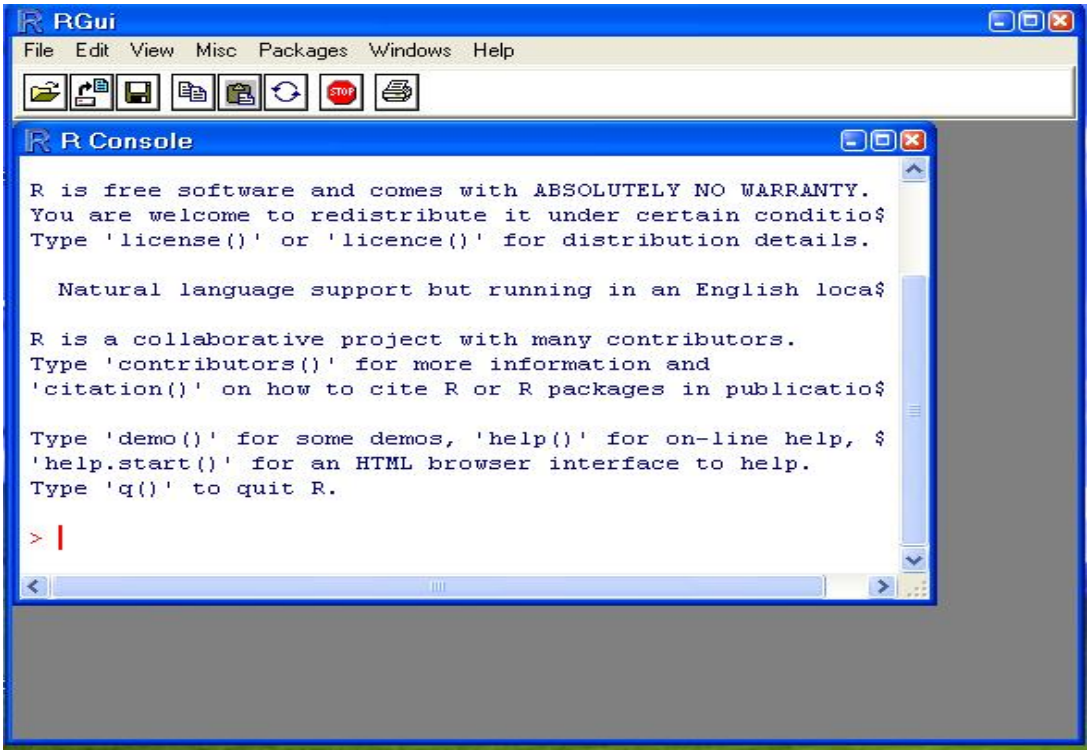
## 2. Cài đặt và chạy R

Để sử dụng R việc đầu tiên chúng ta cần làm là cài đặt R trong máy tính của mình. Để làm điều này chúng ta truy cập vào website. http://cran.R-project.org và tải R xuống.

Khi đã tải R xuống máy tính, bước kế tiếp là cài đặt vào máy tính. Để làm việc này, chúng ta chỉ đơn giản nhấn chuột vào tài liều trên và làm theo hướng dẫn cách cài đặt trên màn hình. Đây là một bước rất đơn giản chỉ cần 1 phút là việc cài đặt sẽ hoàn tất.

Sau khi hoàn tất việc cài đặt một icon


sẽ xuất hiện trên desktop của máy tính. Đến đây thì chúng ta đã sắn sàng sử dụng R có thể nhấp chuột vào icon này và chúng ta sẽ có một window như sau :




### 3. Tính toán dòng lệnh trong R

R thường sử dụng dưới dạng "command line" có nghĩa là chúng ta phải gõ trực tiếp các lệnh vào prompt mầu đỏ trên hình. Các lệnh phải tuân thử nghiêm ngặt các luật của ngôn ngữ R. Một câu lệnh sẽ được thực thi ngay sau khi nhấn phím Enter

R phân biệt chữ hoa và chữ thường vd: library khác với Library. Một văn phạm khác nữa là khi có hai chữ rồi nhau, R thường dùng dấu chấm để thay khoảng trống, chẳng hạn như ***data.frame, t.test, read.table*** ... Điều này rất là quan trọng nếu không để ý sẽ làm mất thì giờ của ngườ sử dụng.

Nếu lệnh gõ ra đúng "Văn phạm " thì R sẽ cho chúng ta một cái prompt khác hay cho ra kết quả nào đó (tùy theo lệnh); nếu lệnh không đúng "Văn Phạm" thì R sẽ đưa ra một thông báo ngắn là không đúng hay không hiểu. Ví dụ : khi chúng ta gõ.

```
> x <- rnorm(20)

>
```

thì R sẽ hiểu và cho chúng ta một cái prompt khác. Nhưng nếu chúng ta gõ lệnh sau :

> R is great

R sẽ không hiểu và đưa ra một thông báo lỗi.

> Error: syntax error

Khi muốn rời khỏi R, chúng ta sẽ đơn giản nhấn nút (x) trên góc trái window hay gõ lệnh q().

**3.1 "Văn phạm" ngôn ngữ R**

Văn phạm chung của R là một lệnh (command) hay function. Mà đã là hàm thì phải có tham số; cho nên theo sau hàm là những tham số mà chúng ta phải cung cấp. chẳng hạn như:

> reg <- lm(y~x)

Để biết một hàm có những tham số nào, chúng ta dùng lệnh args(x), mà trong đó x là hàm mà chúng ta cần biết:

R là một ngôn ngữ "đối tượng ". Điều này có nghĩa là các dữ liệu trong R được chứa trong object. Định hướng này cũng có vài ảnh hưởng đến cách viết cảu R. Chẳng hạn như thay vì viết x = 5 như thông thường chúng ta vẫn viết, thì R yêu cầu viết x == 5.

Đối với R phép gán x = 5 tương đương với x <- 5. Cách viết sau (dùng kí hiệu <- ) được khuyến khích hơn là cách viết trước (=).

Một số kí hiệu hay dùng trong R :

| | |
|---|---|
| x == y | x bằng y |
| x != y | x không bằng y |
| y < x | y nhở hơn x |
| x > y | x lớn hơn y |
| x <= y | x nhở hơn hoặc bằng y |
| x >= y | x lớn hơn hoặc bằng y |
| is.na(x) | có phải x là biến số missing |
| A & B | A và B |

| | |
|---|---|
| A \| B | A hoặc B |
| ! | không là |

Với R thì tất cả các câu chữ hay lệnh sau kí hiệu # đều không có hiệu ứng, vì # là kí hiệu dành cho người sử dụng thêm vào các ghi chú.

### 3.2 Cách đặt tên trong R

Đặt tên một đối tượng hay một biến số trong R khá linh hoạt, vì R không có nhiều giới hạn như trong các phần mền khác. Tên một đối tượng phải được viết liền nhau. Chẳng hạn như R chấp nhận ***myobject*** nhưng không chấp nhận ***my object*** .

*> myobject <- rnorm(10)*

*> my object <- rnorm(10)*

*Error: syntax error in "my object"*

Nhưng đôi khi tên myobject khó đọc cho nên chúng ta nên tách rời bằng "." Như my.object.

*> my.object <- rnorm(10)*

Một điều quan trọng cần lưu ý là R phân biệt mẫu kí tự viết hoa và viết thường. Cho nên ***My.object*** khác so với ***my.object***. Ví dụ:

> My.object.u <- 15

> my.object.L <- 5

> My.object.u + my.object.L

> [1] 20

Một vài điều cần lưu ý khi đặt tên trong R.

- Không nên đặt tên một biến số bằng kí hiệu "_" như ***my_object*** hay ***my-object.***
- Không nên đặt tên một object giống như một biến số trong một dữ liệu. Ví dụ: nếu chúng ta có một data.frame với biến số ***age*** trong đó, thì chúng ta có một đối tượng trùng tên với ***age***, tức là không nên viết ***age <- age***. Tuy nhiên, nếu ***data.frame*** tên là data thì chúng ta

có thể đề cập đến biến số ***age*** với mọt kí tự ***$*** như sau: ***data$age***. (tức là biến số age trong ***data.frame*** *data*), và trong trường hợp đó, age <- data$age có thể chấp nhận được.

### 2.2 Trợ giúp trong R

Ngoài lệnh ***args()*** R còn cung cấp lệnh ***help()*** để người sử dụng có thể hiểu "Văn phạm" của từng hàm. Chẳng hạn như muốn biết hàm ***lm*** có những tham số gì chúng ta chỉ cần gõ lệnh:

*> help()*

hay

*> ?lm*

một cửa sổ sẽ hiện ra bên ngoài của màn hình chỉ rõ cách sử dụng ra sao và thậm chí có cả ví du.

Sử dụng lệnh *help.start()* một cửa sổ sẽ xuất hiện chỉ dẫn toàn bộ hệ thống R.

Hàm ***apropos*** cũng rất có ích vì nó cung cấp cho chúng ta tất cả các hàm trong R bắt đầu bằng kí tự mà chúng ta muốn tìm. Chẳng hạn như chúng ta muốn biết hàm nào trong R có kí tự "lm" thì chỉ gõ lệnh:

*> apropos(lm)* .

## 4. Làm việc với dữ liệu trong R :

### 4.1 Nhập dữ liệu :

Muốn làm phân tích dữ liệu bằng R, chúng ta phải có sẵn dữ liệu ở dạng mà R có thể hiểu được để xử lí. Dữ liệu mà R hiểu được phải là dữ liệu trong một data.frame. Có nhiều cách để nhập số liệu vào một data.frame trong R, từ nhập trực tiếp đến nhập từ các nguồn khác nhau. Sau đây là những cách thông dụng nhất:

#### 4.1.1 Nhập số liệu bằng dòng lệnh :

Để nhập số liệu trực tiếp chúng ta sử dụng function c( ). Lệnh này cho phép chúng ta tạo ra một cột dữ liệu . Cú pháp của hàm này :

>Tên_biến_lưu_dữ_liệu <- c(phần_tử_thứ_1, phần_tử_thứ_2,…… phần_tử_thứ_n).

Ví dụ 1:

a <-c(1,2,3,4,5,6,7,8,9,10,11,12 )

b <-c(12,11,10,9,8,7,6,5,4,3,2,1)

2 Câu lệnh trên cho thấy chúng ta muốn tạo một cột dữ liệu gồm các phần tử từ 1 đến 12 , và cột dữ liệu này được lưu trong một biến có tên là a và một cột dữ liệu bao gồm các phần tử từ 12 trở về 1 và đc lưu trong một biến có tên là b.

R là một ngôn ngữ hướng đối tượng hai biến a và b ở trên là 2 đối tượng riêng lẻ, ta có thể kết hợp chúng để tạo thành môt khung số liệu , để R có thể xử lý chúng sau này . Để làm được điều này , chúng ta dùng phương thức data.frame. Cú pháp của phương thức này như sau :

Tên_biến _lưu_trữ <-data.frame(tham_số_1, tham_số_2, ……. tham_số_n)

Các tham số ở đây là các cột dữ liệu đc khởi tạo bằng function c(). Ví dụ :

ab <- data.frame(a,b) (với a và b là các biến đc khởi tạo trong ví dụ 1)

câu lệnh này cho R biết rằng chúng ta muốn kết hợp hai cột riêng lẻ a và b thành một khung số liệu và được lưu trữ trong biến có tên là ab. Để xem thông tin được lưu trữ trong biến ab vừa tạo ra , ta chỉ cần gõ lệnh :

>ab

Sau đó R sẽ hiển thị :

| | a | b |
|---|---|---|
| 1 | 1 | 12 |
| 2 | 2 | 11 |
| 3 | 3 | 10 |
| 4 | 4 | 9 |

| | | |
|---|---|---|
| 5 | 5 | 8 |
| 6 | 6 | 7 |
| 7 | 7 | 6 |
| 8 | 8 | 5 |
| 9 | 9 | 4 |
| 10 | 10 | 3 |
| 11 | 11 | 2 |
| 12 | 12 | 1 |

Để lưu các số liệu này trong một file có định dang phù hợp với R nhằm mục đích sử dụng cho các lần sau, chúng ta dùng lệnh save với cú pháp như sau :

save(tên_biến , tênfile.rda)

Khi đó biến sẽ được lưu vào ổ đĩa với vị trí được xác định bởi lệnh setwd(). Cú pháp của lệnh này :

setwd("tên_đường _dẫn_mà_ta muốn_lưu_file")

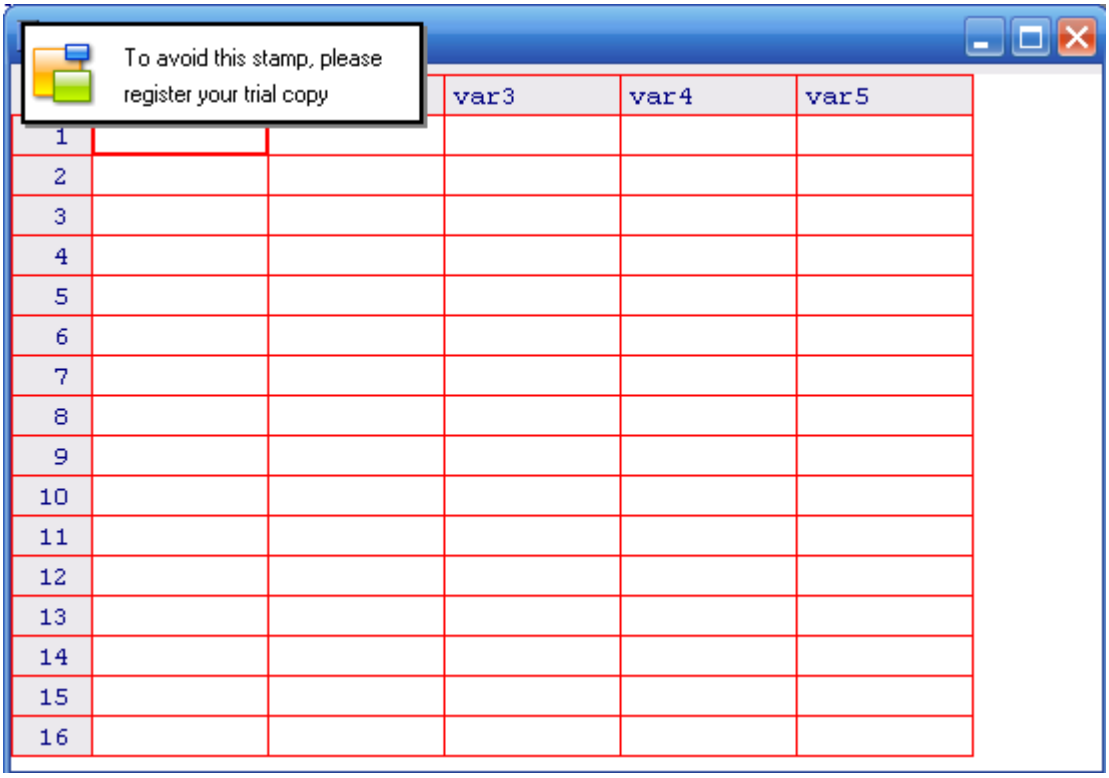
**4.1.2 Nhập số liệu trực tiếp :**

Chúng ta có thể nhập số liệu về độ tuổi và insulin cho 10 bệnh nhân bằng một function rất có ích, đó là: edit(data.frame()).

Với function này, R sẽ cung cấp cho chúng ta một window mới với một dãy cột và dòng giống như Excel, và chúng ta có thể nhập số liệu trong bảng đó. Ví dụ:

> ins <- edit(data.frame())

Chúng ta sẽ có một cửa sổ như sau:




Ở đây, R không biết chúng ta có biến số nào, cho nên R liệt kê các biến số var1,var2, v.v… Nhấp chuột vào cột var1 và thay đổi bằng cách gõ vào đó a. Nhấp chuột vào cột var2 và thay đổi bằng cách gõ vào đó b. Sau đó gõ số liệu cho từng cột. Sau khi xong, click vào biểu tượng close ở góc phải của spreadsheet, chúng ta sẽ có một data.frame tên ins với hai biến số a và b.

### 4.1.3 Nhập dữ liệu từ file text :

Chúng ta thu thập số liệu về độ tuổi và cholesterol từ một nghiên cứu ở 50 bệnh nhân mắc bệnh cao huyết áp. Các số liệu này được lưu trong một text file có tên là chol.txt tại directory c:\works\stats. Số liệu này như sau: cột 1 là mã số của bệnh nhân, cột 2 là giới tính, cột 3 là body mass index (bmi), cột 4 là HDL cholesterol (viết tắt là hdl), kế đến là LDL cholesterol, total cholesterol (tc) và triglycerides (tg).

| id | sex | age | bmi | hdl | ldl | tc | tg |
|---|---|---|---|---|---|---|---|
| 1 | Nam | 57 | 17 | 5.000 | 2.0 | 4.0 | 1.1 |
| 2 | Nu | 64 | 18 | 4.380 | 3.0 | 3.5 | 2.1 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | 18 | 3.360 | 3.0 | 4.7 | 0.8 |
| | | | 18 | 5.920 | 4.0 | 7.7 | 1.1 |
| | | | 18 | 6.250 | 2.1 | 5.0 | 2.1 |
| | | | 18 | 4.150 | 3.0 | 4.2 | 1.5 |
| 7 | Nam | 76 | 19 | 0.737 | 3.0 | 5.9 | 2.6 |
| 8 | Nam | 61 | 19 | 7.170 | 3.0 | 6.1 | 1.5 |
| 9 | Nam | 59 | 19 | 6.942 | 3.0 | 5.9 | 5.4 |
| 10 | Nu | 57 | 19 | 5.000 | 2.0 | 4.0 | 1.9 |
| 11 | Nu | 63 | 20 | 4.217 | 5.0 | 6.2 | 1.7 |
| 12 | Nam | 51 | 20 | 4.823 | 1.3 | 4.1 | 1.0 |
| 13 | Nu | 60 | 20 | 3.750 | 1.2 | 3.0 | 1.6 |
| 14 | Nam | 42 | 20 | 1.904 | 0.7 | 4.0 | 1.1 |
| 15 | Nam | 64 | 20 | 6.900 | 4.0 | 6.9 | 1.5 |
| 16 | Nu | 49 | 20 | 0.633 | 4.1 | 5.7 | 1.0 |
| 17 | Nu | 44 | 21 | 5.530 | 4.3 | 5.7 | 2.7 |
| 18 | Nu | 45 | 21 | 6.625 | 4.0 | 5.3 | 3.9 |
| 19 | Nu | 80 | 21 | 5.960 | 4.3 | 7.1 | 3.0 |
| 20 | Nu | 48 | 21 | 3.800 | 4.0 | 3.8 | 3.1 |
| 21 | Nu | 61 | 21 | 5.375 | 3.1 | 4.3 | 2.2 |
| 22 | Nu | 45 | 21 | 3.360 | 3.0 | 4.8 | 2.7 |
| 23 | Nu | 70 | 21 | 5.000 | 1.7 | 4.0 | 1.1 |
| 24 | Nu | 51 | 21 | 2.608 | 2.0 | 3.0 | 0.7 |
| 25 | Nam | 63 | 22 | 4.130 | 2.1 | 3.1 | 1.0 |
| 26 | Nam | 54 | 22 | 5.000 | 4.0 | 5.3 | 1.7 |
| 27 | Nu | 57 | 22 | 6.235 | 4.1 | 5.3 | 2.9 |
| 28 | Nam | 70 | 22 | 3.600 | 4.0 | 5.4 | 2.5 |
| 29 | Nu | 47 | 22 | 5.625 | 4.2 | 4.5 | 6.2 |
| 30 | Nu | 60 | 22 | 5.360 | 4.2 | 5.9 | 1.3 |
| 31 | Nu | 60 | 22 | 6.580 | 4.4 | 5.6 | 3.3 |
| 32 | Nam | 50 | 22 | 7.545 | 4.3 | 8.3 | 3.0 |
| 33 | Nam | 60 | 22 | 6.440 | 2.3 | 5.8 | 1.0 |
| 34 | Nu | 55 | 22 | 6.170 | 6.0 | 7.6 | 1.4 |
| 35 | Nu | 74 | 23 | 5.270 | 3.0 | 5.8 | 2.5 |
| 36 | Nam | 48 | 23 | 3.220 | 3.0 | 3.1 | 0.7 |
| 37 | Nu | 46 | 23 | 5.400 | 2.6 | 5.4 | 2.4 |
| 38 | Nam | 49 | 23 | 6.300 | 4.4 | 6.3 | 2.4 |

Chúng ta muốn nhập các dữ liệu này vào R để tiện việc phân tích sau này. Chúng ta sẽ sử dụng lệnh read.table như sau:

> setwd(“c:/works/stats”)

> chol <- read.table("chol.txt", header=TRUE)

Lệnh thứ nhất chúng ta muốn đảm bảo R truy nhập đúng directory mà số liệu đang được lưu giữ. Lệnh thứ hai yêu cầu R nhập số liệu từ file có tên là “chol.txt”(trong directory c:\works\stats) và cho vào đối tượng chol. Trong lệnh này, header=TRUE có nghĩa là yêu cầu

> chol

Hay

> names(chol)

R sẽ cho biết có các cột như sau trong dữ liệu (name là lệnh hỏi trong dữ liệu có những cột nào và tên gì):

[1] "id" "sex" "age" "bmi" "hdl" "ldl" "tc" "tg"

Bây giờ chúng ta có thể lưu dữ liệu dưới dạng R để xử lí sau này bằng cách ra lệnh:

> save(chol, file="chol.rda")

**4.1.4 Nhập dữ liệu từ file xls (Excel) :**

R cũng cho phép chúng ta có thể nhập dữ liệu từ một file định dạng xls của Excel một cách đơn giản chỉ với vài thao tác. Trước tiên chúng ta lưu lại file xls dưới định dạng *.csv để R có thể xử lý được. Sau đó sử dụng lệnh read.csv() để xử lý. Cú pháp của lệnh này như sau :

Tên_biến_lưu_liệu<-read.csv("đường_dẫn_đến_file_csv",HEADER=true)

Tham số HEADER = true cho R biết chúng ta muốn chọn dòng đầu tiên của file xls làm tên của các cột. Sau khi thực hiện lệnh này chúng ta đã có một đối tượng chuẩn của R để lưu trữ dữ liệu của file xls ban đầu. Chúng ta có thể lưu lại đối tượng này cho các lần làm việc sau băng lện save() đã được giới thiệu ở trên.

**4.2 Xử lý dữ liệu :**

Biên tập số liệu ở đây không có nghĩa là thay đổi số liệu gốc (vì đó là một tội lớn, một sự gian dối trong khoa học không thể chấp nhận được), mà chỉ có nghĩa tổ chức số liệu sao cho R có thể phân tích một cách hữu hiệu. ẩ hiều khi trong phân tích thống kê, chúng ta cần phải tập trung số liệu thành một nhóm, hay tách rời thành từng nhóm, hay thay thế từ kí tự (characters) sang số (numeric) cho tiện việc tính toán. Trong chương này, tôi sẽ bàn qua một số lệnh căn bản cho việc biên tập số liệu. Chúng ta sẽ quay lại với dữ liệu chol trong ví dụ 1.

> setwd(“c:/works/stats”)

> chol <- read.table(“chol.txt”, header=TRUE)

> attach(chol)

### 4.2.1 Kiểm tra số liệu trống không (missing value)

Trong nghiên cứu, vì nhiều lí do số liệu không thể thu thập được cho tất cả đối tượng, hay không thể đo lường tất cả biến số cho một đối tượng. Trong trường hợp đó, số liệu trống được xem là “missing value” . R xem các số liệu trống không là ả A. Có một số kiểm định thống kê đòi hỏi các số liệu trống không phải được loại ra (vì không thể tính toán được) trước khi phân tích. R có một lệnh rất có ích cho việc này: na.omit, và cách sử dụng như sau:

> chol.new <- na.omit(chol)

Trong lệnh trên, chúng ta yêu cầu R loại bỏ các số liệu trống không trong data.frame chol và đưa các số liệu không trống vào data.frame mới tên là chol.new. Chú ý lệnh trên chỉ là ví dụ, vì trong dữ liệu chol không có số liệu trống không.

### 4.2.2 Tách rời dữ liệu: subset

ả ếu chúng ta, vì một lí do nào đó, chỉ muốn phân tích riêng cho nam giới, chúng ta có thể tách chol ra thành hai data.frame, tạm gọi là nam và nu. Để làm chuyện này,chúng ta dùng lệnh subset(data, cond), trong đó data là data.frame mà chúng ta muốn tách rời, và cond là điều kiện. Ví dụ:

> nam <- subset(chol, sex==”ả am”)

> nu <- subset(chol, sex==”ả u”)

Sau khi ra hai lệnh này, chúng ta đã có 2 dữ liệu (hai data.frame) mới tên là nam và nu. Chú ý điều kiện sex == “ả am” và sex == “ả u” chúng ta dùng == thay vì = để chỉ điều kiện chính xác. Tất nhiên, chúng ta cũng có thể tách dữ liệu thành nhiều data.frame khác nhau với những điều kiện dựa vào các biến số khác. Chẳng hạn như lệnh sau đây tạo ra một data.frame mới tên là old với những bệnh nhân trên 60 tuổi:

> old <- subset(chol, age>=60)

```
> dim(old)

[1] 25 8
```

Hay một data.frame mới với những bệnh nhân trên 60 tuổi và nam giới:

```
> n60 <- subset(chol, age>=60 & sex==”ẩ am”)

> dim(n60)

[1] 9 8
```

**4.2.3 Chiết số liệu từ một data .frame**

Trong chol có 8 biến số. Chúng ta có thể chiết dữ liệu chol và chỉ giữ lại những biến số cần thiết như mã số (id), độ tuổi (age) và total cholestrol (tc). Để ý từ lệnh names(chol) rằng biến số id là cột số 1, age là cột số 3, và biến số tc là cột số 7. Chúng ta có thể dùng lệnh sau đây:

```
> data2 <- chol[, c(1,3,7)]
```

Ở đây, chúng ta lệnh cho R biết rằng chúng ta muốn chọn cột số 1, 3 và 7, và đưa tất cả số liệu của hai cột này vào data.frame mới có tên là data2. Chú ý chúng ta sử dụng ngoặc kép vuông [] chứ không phải ngoặc kép vòng (), vì chol không phải làm một function. Dấu phẩy phía trước c, có nghĩa là chúng ta chọn tất cả các dòng số liệu trong data.frame chol. ẩ hưng nếu chúng ta chỉ muốn chọn 10 dòng số liệu đầu tiên, thì lệnh sẽ là:

```
> data3 <- chol[1:10, c(1,3,7)]

> print(data3)

id sex tc

1 1 ẩ am 4.0

2 2 ẩ u 3.5

3 3 ẩ u 4.7

4 4 ẩ am 7.7

5 5 ẩ am 5.0
```

6 6 ả u 4.2

7 7 ả am 5.9

8 8 ả am 6.1

9 9 ả am 5.9

10 10 ả u 4.0

Chú ý lệnh print(arg) đơn giản liệt kê tất cả số liệu trong data.frame arg. Thật ra, chúng ta chỉ cần đơn giản gõ data3, kết quả cũng giống y như print(data3).

### 4.2.4 Nhập hai data.frame thành một: merge

Giả dụ như chúng ta có dữ liệu chứa trong hai data.frame. Dữ liệu thứ nhất tên là d1 gồm 3 cột: id, sex, tc như sau:

id sex tc

1 ả am 4.0

2 ả u 3.5

3 ả u 4.7

4 ả am 7.7

5 ả am 5.0

6 ả u 4.2

7 ả am 5.9

8 ả am 6.1

9 ả am 5.9

10 ả u 4.0

Dữ liệu thứ hai tên là d2 gồm 3 cột: id, sex, tg như sau:

id sex tg

1 ả am 1.1

2 ả u 2.1

3 ả u 0.8

4 ả am 1.1

5 ả am 2.1

6 ả u 1.5

7 ả am 2.6

8 ả am 1.5

9 ả am 5.4

10 ả u 1.9

11 ả u 1.7

Hai dữ liệu này có chung hai biến số id và sex. ả hưng dữ liệu d1 có 10 dòng, còn dữ liệu d2 có 11 dòng. Chúng ta có thể nhập hai dữ liệu thành một data.frame bằng cách dùng lệnh merge như sau:

> d <- merge(d1, d2, by="id", all=TRUE)

> d

id sex.x tc sex.y tg

1 1 ả am 4.0 ả am 1.1

2 2 ả u 3.5 ả u 2.1

3 3 ả u 4.7 ả u 0.8

4 4 ả am 7.7 ả am 1.1

5 5 ả am 5.0 ả am 2.1

6 6 ả u 4.2 ả u 1.5

7 7 ả am 5.9 ả am 2.6

8 8 ả am 6.1 ả am 1.5

9 9 ả am 5.9 ả am 5.4

10 10 ả u 4.0 ả u 1.9

11 11 <ả A> ả A ả u 1.7

Trong lệnh merge, chúng ta yêu cầu R nhập 2 dữ liệu d1 và d2 thành một và đưa vào data.frame mới tên là d, và dùng biến số id làm chuẩn. Chúng ta để ý thấy bệnh nhân số 11 không có số liệu cho tc, cho nên R cho là ả A (một dạng "not available").

### 4.2.5 Mã hóa số liệu (data coding)

Trong việc xử lí số liệu dịch tễ học, nhiều khi chúng ta cần phải biến đổi số liệu từ biến liên tục sang biến mang tính cách phân loại. Chẳng hạn như trong chẩn đoán loãng xương, những phụ nữ có chỉ số T của mật độ chất khoáng trong xương (bone mineral density hay BMD) bằng hay thấp hơn -2.5 được xem là "loãng xương", những ai có BMD giữa -2.5 và -1.0 là "xốp xương" (osteopenia), và trên -1.0 là "bình thường". Ví dụ, chúng ta có số liệu BMD từ 10 bệnh nhân như sau:

-0.92, 0.21, 0.17, -3.21, -1.80, -2.60, -2.00, 1.71, 2.12, -2.11

Để nhập các số liệu này vào R chúng ta có thể sử dụng *function* c như sau:

bmd <- c(-0.92,0.21,0.17,-3.21,-1.80,-2.60,-2.00,1.71,2.12,-2.11)

Để phân loại 3 nhóm loãng xương, xốp xương, và bình thường, chúng ta có thể dùng mã số 1, 2 và 3. ả ói cách khác, chúng ta muốn tạo nên một biến số khác (hãy gọi là diagnosis) gồm 3 giá trị trên dựa vào giá trị của bmd. Để làm việc này, chúng ta sử dụng lệnh:

# tạm thời cho biến số diagnosis bằng bmd

> diagnosis <- bmd

# biến đổi bmd thành diagnosis

> diagnosis[bmd <= -2.5] <- 1

> diagnosis[bmd > -2.5 & bmd <= 1.0] <- 2

```
> diagnosis[bmd > -1.0] <- 3

# tạo thành một data frame

> data <- data.frame(bmd, diagnosis)

# liệt kê để kiểm tra xem lệnh có hiệu quả không

> data

bmd diagnosis

1 -0.92 3

2 0.21 3

3 0.17 3

4 -3.21 1

5 -1.80 2

6 -2.60 1

7 -2.00 2

8 1.71 3

9 2.12 3

10 -2.11 2
```

#### 4.2.5.1 Biến đổi số liệu bằng cách dùng *replace*

Một cách biến đổi số liệu khác là dùng replace, dù cách này có vẻ rườm rà chút ít. Tiếp tục ví dụ trên, chúng ta biến đổi từ bmd sang diagnosis như sau:

```
> diagnosis <- bmd

> diagnosis <- replace(diagnosis, bmd <= -2.5, 1)

> diagnosis <- replace(diagnosis, bmd > -2.5 & bmd <= 1.0, 2)
```

> diagnosis <- replace(diagnosis, bmd > -1.0, 3)

#### 4.2.5.2 Biến đổi thành yếu tố (*factor*)

Trong phân tích thống kê, chúng ta phân biệt một biến số mang tính *yếu tố* (factor) và biến số liên tục bình thường. Biến số yếu tố không thể dùng để tính toán như cộng trừ nhân chia, nhưng biến số số học có thể sử dụng để tính toán. Chẳng hạn như trong ví dụ bmd và diagnosis trên, diagnosis là yếu tố vì giá trị trung bình giữa 1 và 2 chẳng có ý nghĩa thực tế gì cả; còn bmd là biến số số học. ả hưng hiện nay, diagnosis được xem là một biến số số học. Để biến thành biến số yếu tố, chúng ta cần sử dụng *function* factor như sau:

> diag <- factor(diagnosis)

> diag

[1] 3 3 3 1 2 1 2 3 3 2

Levels: 1 2 3

Chú ý R bây giờ thông báo cho chúng ta biết diag có 3 bậc: 1, 2 và 3. ả ếu chúng ta yêu cầu R tính số trung bình của diag, R sẽ không làm theo yêu cầu này, vì đó không phải là một biến số số học:

> mean(diag)

[1] ả A

Warning message:

argument is not numeric or logical: returning ả A in: mean.default(diag)

Dĩ nhiên, chúng ta có thể tính giá trị trung bình của diagnosis:

> mean(diagnosis)

[1] 2.3

nhưng kết quả 2.3 này không có ý nghĩa gì trong thực tế cả.

### 4.2.6 Chia nhóm bằng *cut*

Với một biến liên tục, chúng ta có thể chia thành nhiều nhóm bằng hàm cut. Ví dụ, chúng ta có biến age như sau:

```
> age <- c(17,19,22,43,14,8,12,19,20,51,8,12,27,31,44)
```

Độ tuổi thấp nhất là 8 và cao nhất là 51. ả ếu chúng ta muốn chia thành 2 nhóm tuổi:

```
> cut(age, 2)

[1] (7.96,29.5] (7.96,29.5] (7.96,29.5] (29.5,51] (7.96,29.5] (7.96,29.5]

(7.96,29.5] (7.96,29.5]

[9] (7.96,29.5] (29.5,51] (7.96,29.5] (7.96,29.5] (7.96,29.5] (29.5,51]

(29.5,51]

Levels: (7.96,29.5] (29.5,51]
```

cut chia biến age thành 2 nhóm: nhóm 1 tuổi từ 7.96 đến 29.5; nhóm 2 từ 29.5 đến 51. Chúng ta có thể đếm số đối tượng trong từng nhóm tuổi bằng hàm table như sau:

```
> table(cut(age, 2))

(7.96,29.5] (29.5,51]

11 4

> ageg <- cut(age, 3, labels=c("low", "medium", "high"))

[1] low low low high low low low low low high

low low medium medium

[15] high

Levels: low medium high

> ageg <- cut(age, 3, labels=c("low", "medium", "high"))

> table(ageg)

ageg
```

low medium high

10 2 3

Tất nhiên, chúng ta cũng có thể chia age thành 4 nhóm (quartiles) bằng cách cho những thông số 0, 0.25, 0.50 và 0.75 như sau:

```
cut(age,
breaks=quantiles(age, c(0, 0.25, 0.50, 0.75, 1)),
labels=c("q1", "q2", "q3", "q4"),
include.lowest=TRUE)
cut(age,
breaks=quantiles(c(0, 0.25, 0.50, 0.75, 1)),
labels=c("q1", "q2", "q3", "q4"),
include.lowest=TRUE)
```

### 4.7. Tập hợp số liệu bằng *cut2* (Hmisc)

Hàm cut trên chia biến số theo giá trị của biến, chứ không dựa vào số mẫu, cho nên số lượng mẫu trong từng nhóm không bằng nhau. Tuy nhiên, trong phân tích thống kê, có khi chúng ta cần phải phân chia một biến số liên tục thành nhiều nhóm dựa vào phân phối của biến số nhưng số mẫu bằng hay tương đương nhau. Chẳng hạn như đối với biến số bmd chúng ta có thể "cắt" dãy số thành 3 nhóm với số mẫu tương đương nhau bằng cách dùng function cut2 (trong thư viện Hmisc) như sau:

```
> # nhập thư viện Hmisc để có thể dùng function cut2
> library(Hmisc)
> bmd <- c(-0.92,0.21,0.17,-3.21,-1.80,-2.60,-2.00,1.71,2.12,-2.11)
> # chia biến số bmd thành 2 nhóm và để trong đối tượng group
> group <- cut2(bmd, g=2)
> table(group)
```

group

[-3.21,-0.92) [-0.92, 2.12]

5 5

ả hư thấy qua ví dụ trên, g = 2 có nghĩa là chia thành 2 nhóm (g = group). R tự động chia thành nhóm 1 gồm giá trị bmd từ -3.21 đến -0.92, và nhóm 2 từ -0.92 đến 2.12. Mỗi nhóm gồm có 5 số.

Tất nhiên, chúng ta cũng có thể chia thành 3 nhóm bằng lệnh:

> group <- cut2(bmd, g=3)

Và với lệnh table chúng ta sẽ biết có 3 nhóm, nhóm 1 gồm 4 số, nhóm 2 và 3 mỗi nhóm có 3 số:

> table(group)

group

[-3.21,-1.80) [-1.80, 0.21) [ 0.21, 2.12]

4 3 3

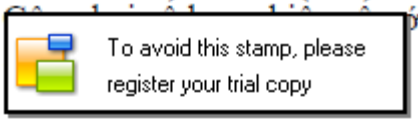
**5. Tính toán dòng lệnh trong R :**

R cung cấp rất nhiều các phép toán và các hàm (function) đa dạng để phuc vụ cho việc tính toán , hầu hết các hám số thông dụng đều được hỗ trợ bởi R. ả goài ra còn rất nhiều các hàm phục vụ cho cho các công việc tính toán phức tạp và nâng cao cũng được cung cấp bởi rất nhiều các gói mở rộng dành cho R. Chúng ta có thể tải các gói mở rông đó tại địa chỉ sau:

http://www.cran.r-project.org/

Chọn mục packages sau đó bạn có thể tải các gói cần thiết một cách dễ dàng . Các gói này là hoàn toàn miễn phí.

Sau đây là một số lệnh đơn giản dung để tính toán của R :

<table>
<tr><td> 

ới nhau:</td><td>Cộng và trừ:<br>
<pre>> 15+2997-9768
[1] -6756</pre></td></tr>
<tr><td>Nhân và chia<br>
<pre>> -27*12/21
[1] -15.42857</pre></td><td>Số lũy thừa: $(25-5)^3$<br>
<pre>> (25 - 5)^3
[1] 8000</pre></td></tr>
<tr><td>Căn số bậc hai: $\sqrt{10}$<br>
<pre>> sqrt(10)
[1] 3.162278</pre></td><td>Số pi ($\pi$)<br>
<pre>> pi
[1] 3.141593
> 2+3*pi
[1] 11.42478</pre></td></tr>
<tr><td>Logarit: $\log_e$<br>
<pre>> log(10)
[1] 2.302585</pre></td><td>Logarit: $\log_{10}$<br>
<pre>> log10(100)
[1] 2</pre></td></tr>
<tr><td>Số mũ: $e^{2.7689}$<br>
<pre>> exp(2.7689)
[1] 15.94109

> log10(2+3*pi)
[1] 1.057848</pre></td><td>Hàm số lượng giác<br>
<pre>> cos(pi)
[1] -1</pre></td></tr>
<tr><td>Vector<br>
<pre>> x <- c(2,3,1,5,4,6,7,6,8)
> x
[1] 2 3 1 5 4 6 7 6 8

> sum(x)
[1] 42

> x*2</pre></td><td><pre>> exp(x/10)
[1] 1.221403 1.349859 1.105171 1.648
1.491825 1.822119 2.013753 1.822119
[9] 2.225541

> exp(cos(x/10))
[1] 2.664634 2.599545 2.704736 2.405
2.511954 2.282647 2.148655 2.282647
[9] 2.007132</pre></td></tr>
</table>

| | |
|---|---|
| [1] 4 6 8 10 12 14 12 16 | |
| n of squares): $1^2 + 2^2 + 3^2 + 4^2 + 5^2 = ?$<br>`> x <- c(1,2,3,4,5)`<br>`> sum(x^2)`<br>`[1] 55` | Tính tổng bình phương điều chỉnh (adjusted sum of squares): $\sum_{i=1}^{n}(x_i - \bar{x})^2 = ?$<br>`> x <- c(1,2,3,4,5)`<br>`> sum((x-mean(x))^2)`<br>`[1] 10`<br><br>Trong công thức trên `mean(x)` là số trung bình của vector `x`. |
| Tính sai số bình phương (mean square): $\sum_{i=1}^{n}(x_i - \bar{x})^2 / n = ?$<br>`> x <- c(1,2,3,4,5)`<br>`> sum((x-mean(x))^2)/length(x)`<br>`[1] 2`<br><br>Trong công thức trên, `length(x)` có nghĩa là tổng số phần tử (elements) trong vector `x`. | Tính phương sai (variance) và độ lệch chuẩn (standard deviation):<br>Phương sai: $s^2 = \sum_{i=1}^{n}(x_i - \bar{x})^2 / (n-1) = ?$<br>`> x <- c(1,2,3,4,5)`<br>`> var(x)`<br>`[1] 2.5`<br>Độ lệch chuẩn: $\sqrt{s^2}$ :<br>`> sd(x)`<br>`[1] 1.581139` |

**6. Lập trình với ngôn ngữ R :**

**6.1 Tổng quan về lập trình với R :**

ả gôn ngữ R có rất nhiều ưu điểm so với các ngôn ngữ lập trình bậc cao như C , C++ , Java….

- R có khả năng điều khiển dữ liệu và lưu trữ số liệu, R còn có tính nguyên bản.
- R cho phép sử dụng ma trận đại số.
- Có thể sử dụng bảng băm và các biểu thức chính quy
- R cũng hỗ trợ lập trình hướng đối tượng.
- Khả năng biểu diễn đồ họa phong phú.
- ả gôn ngữ R cũng cung cấp các cấu trúc điều khiển cơ bản như các ngôn ngữ lập trình bậc cao khác. Ví dụ như :

  If…else…;while….;for……v..v..

R cũng có một số nhược điểm hay có thể gọi là thiếu sót , tuy nhiên các nhược điểm này có thể được khắc phục dễ dàng bởi chính R :

- R không phải là một cơ sở dữ liệu nhưng lại có thể kết nối với các hệ quán trị cơ sở dữ liệu (DBMS)
- R không có giao diện đồ họa người dùng, nhưng nó co thể kết nối với Java, TclTk.
- Việc diễn giải ngôn ngữ R có thể rất chậm, nhưng có thể cho phép gọi tới các mã C hoặc C++.
- R không có các bảng tính quan sát dữ liệu, nhưng nó có thể kết nối với Excel/MSOffice.
- Mỗi câu lệnh của R kết thúc bằng phím Enter, điều này gây ra sự bất tiện trong khi lập trình, đặt biệt là khi xây dựng một hàm, chỉ cần sai một dòng lệnh, ta sẽ phải làm lại từ đầu.
- Một nhược điểm khác của R là nó không chuyên nghiệp và không hỗ trợ thương mại .

**6.2 Các kiểu dữ liệu cơ bản sử dụng trong lập trình với R :**

ả gôn ngữ R không bắt buộc phải khai báo kiểu dữ liệu ngay khi khai báo 1 biến , kiểu dữ liệu của một biến sẽ được xác định khi ta gán một giá trị cụ thể cho nó.

R gồm có 4 kiểu dữ liệu cơ bản :

- numeric (kiểu số):

- character

- string

- logical

Khác với các ngôn ngữ lập trình bậc cao, các giá trị số trong R không chia thành kiểu thực(real) và kiểu nguyên(integer). Dữ liệu kiểu số trong R chỉ có 1 kiểu duy nhất. Ví dụ :

```
>a<-10

>b<-1.5
```

Dấu “<-”tương tự với phép gán “=” trong các ngôn ngữ lập trình bậc cao, chúng ta cũng có thể thay dấu “<-” bởi dấu “=” nhưng được khuyến cáo là nên sử dụng dấu “<-”.

Kiểu character và string trong R cũng tương tự như trong các ngôn ngữ lập trình bậc cao. Các giá trị char và string được đặt trong cặp dấu “”. Ví dụ :

>a<-“nguyen trung kien”

>b<-“a”

Kiểu logical tương tự như kiểu bool (giá trị logic ) trong các ngôn ngữ lập trình bậc cao. Kiểu dữ liệu này chỉ gồm 2 giá trị đúng hoặc sai. Ví dụ :

>a<-((1+3)==4)

>a

[1]TRUE

**6.3 Các kiểu dữ liệu tuyến tính trong R :**

Ở đây ta sẽ tìm hiểu về 2 kiểu dữ liệu tuyến tính trong R là array và list.

Một mảng (array) trong R được khai báo theo cú pháp sau :

>a<- c(1,2,3,4)

>a[3]

[1]3

Khác với mảng, list trong R cũng là một dãy các phần tử nhưng chúng được gọi theo tên của phần tử được đặt khi khai báo, ví dụ khai báo một list :

>kien<-list(name=”kien”,class=”k57b”,age=21)

>kien$name

[1]”kien”

>kien$class

[1]”k57b”

>kien$age

[1]”21”

**6.4 Các cấu trúc điều khiển cơ bản trong R :**

ả gôn ngữ R cung cấp cho chúng ta các cấu trúc điều khiển cơ bản như trong đa số các ngôn ngữ lập trình bậc cao khác

**6.4.1 Cấu trúc rẽ nhánh :**

- Cấu trúc if…else..:

Cú pháp :

if(biểu_thức_logic){khối_lệnh_1}else{khối_lệnh_2}

khối_lênh_1 được thực hiện nếu biểu thức logic trong dấu ngoặc trả về kết quả true (đúng), trong trường hợp ngược lại thì khối_lênh_2 sẽ đc thực hiện.

Ví dụ :

if(1==0){print(1)}else{print}

- Cấu trúc ifelse:

  Cú pháp : **ifelse(Biểu thức logic, khối lệnh 1, khối lệnh 2)**

  **Khối lệnh 1** được thực hiện khi **biểu thức logic** trả về **TRUE**, ngược lại thì sẽ thực hiện **khối lệnh 2**.

Ví dụ:

```
x <- 1:10

ifelse(x<5 | x>8, x, 0)
```

**6.4.2 Cấu trúc lặp :**

- Cấu trúc lặp với số lần lặp đã xác định for :

Cú pháp :

```
for(biến_điều_khiển in khoảng_giá_trị) {
   đoạn_lệnh_cần_lặp
}
```

Ví dụ :

```
x <- 1:10; z <- NULL
for(i in seq(along=x)) {
    if (x[i]<5) { z <- c(z,x[i]-1)  } else { z <- c(z,x[i]/x[i])  }
}
```

- Cấu trúc lặp với số lần lặp không biết trước :

**>while(điều_kiện_lặp){đoạn_lệnh_cần_lặp}**

Trong cấu trúc lặp này , đoạn lệnh cần lặp sẽ được thực hiện trong khi biểu_thức_điều_kiện còn đúng . Ví dụ :

```
>i=0
>while(i<5){print(i)
i=i+1
}
```

Trong R một đoạn lệnh được đóng gói bởi một cặp dấu "{}", vì vậy khi bắt đầu thực hiện một đoạn lệnh bởi dấu "{" các câu lệnh trong đoạn lệnh sẽ không được thực thi ngay sau khi nhấn Enter, lúc đó con trỏ dòng lệnh sẽ chuyển sang dấu "+" , khi chương trình gặp dấu "}" thì đoạn lệnh trong cặp dấu "{}" mới được thực thi.

**6.5 Xây dựng hàm trong R :**

Cú pháp định nghĩa một hàm trong R :

**>myfct <- function(arg1, arg2, ...) { function_body  }**

Giá trị được trả về bởi một hàm là giá trị được tạo bởi thân hàm đó , giá trị này thường được trả về trong dòng lệnh cuối của thân hàm , ví dụ ta có thể dùng câu lệnh return() để trả về giá trị của hàm .

Cú pháp khi gọi một hàm đã được định nghĩa :

**myfct(arg1=..., arg2=...)**

Các quy tắc cú pháp :

- Thông thường :

Các chức năng được định nghĩa bằng việc gán bởi từ khóa "function". Phần khai báo các tham số được đặt trong cặp dấu () , các tham số được ngăn cách bởi dấu ",". Các câu lệnh thực hiện chức năng của hàm nằm trong phần thân hàm giữa hai dấu "{}" , cần phải gán tên cho hàm để có thể gọi lại sau này.

- Cách đặt tên hàm :

Tên hàm gần như có thể đặt bằng bất cứ cách nào, tuy nhiên cần tránh đặt tên hàm trùng các hàm sẵn có trong R.

- Chức năng của phần thân hàm :

Tại đây các câu lệnh điều khiển và thực hiện chức năng của hàm được khai báo. Các câu lệnh riêng biệt được ngăn cách nhau bởi dấu ";"

- Phạm vi của biến :

Một biến được khai báo trong một hàm sẽ chỉ tồn tại trong thời gian hoạt động của hàm đó. Vì vậy, chúng ta không thể gọi tới 1 biến được khai báo bên trong một hàm từ bên ngoài hàm đó . Chúng ta có thể làm điều này bằng cách khai báo biến là một biến toàn cục với toán tử "<<-" thay vì toán tử gán "<-" thông thường.

Ví Dụ: viết hàm so sánh 2 số a và b

```
sosanh <- function(a,b)
{
if(a >= b)
```

```
{return (TRUE)}else{return (FALSE)}
}
```

Ví dụ: giải phương trình bậc 2:

```
giai <- function(a,b,c)
{if(a==0)
{ifelse(b==0,result <- c("pt vo nghiem"),result <- -c/b)}
else
{delta <- b*b-4*a*c
if(delta > 0)
{
re1 <- (-b+sqrt(delta))/2*a
re2 <- (-b-sqrt(delta))/2*a
result <- c(re1,re2)
}
else{if(delta == 0){result <- -b/2*a}else{
result <- c("pt vo nghiem")}
}
}
return (result)
}
```