**HƯỚNG DẪN SỬ DỤNG R CĂN BẢN**

Họ và tên sinh viên:

Mã sinh viên:

**1.Cài đặt R và R-Studio:**

**Cài đặt R**:

* Truy cập trang web: [https://cran.r-project.org](https://cran.r-project.org/)
* Lựa chọn Download R for Windows
* Lựa chọn base
* Download phiên bản mới nhất Download R 3.5.0 for Windows
* Cài đặt R như phần mềm thông thường

**Cài đặt Rstudio**:

* Truy cập trang web: [https://rstudio.com](https://rstudio.com/)
* Lựa chọn Product >> Rstudio
* Lựa chọn Rstudio desktop >> Download RStudio Desktop
* Download phiên bản RStudio mới nhất
* Cài đặt RStudio như phần mềm thông thường

**2.Cửa sổ làm việc trong phần mềm R:**

****

*H 5: RStudio có 4 ô (panel), trên trái là Source (mã nguồn), trên phải là Workspace và History, dưới trái là Console, dưới phải là Files, Plots, Packages và Help*

Khác với bản R gốc với RStudio bạn có nhiều panel và mỗi panel có mục đích khác nhau.

Bạn có thể tắt mở panel bằng nút minimize và maximize ở góc trên phải của nó.

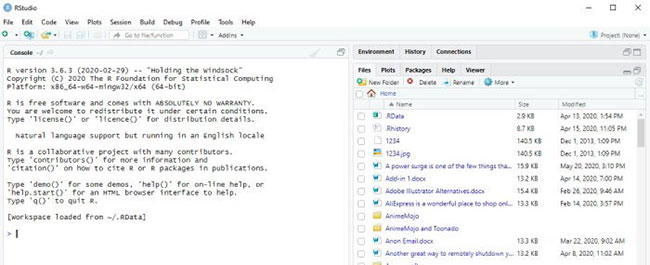


*H 6: Nút thu nhỏ và phóng to ở góc mỗi panel.*

Bạn có thể thay đổi kích thước panel bằng cách đưa chuột đến biên của panel, khi con trỏ chuột trở thành hình mũi tên bốn góc, bạn kéo chuột (drag).

**Làm việc với R console**

Chính là nơi ta gõ dòng lệnh với R (bình thường nếu chạy R độc lập thì ta chỉ có R console, nhưng trong RStudio nó là một thành phần)



**3.Khởi động R, các phép tính cơ bản:**

* Khởi động: nháy đúp vào biểu tượng.
* Phép tính cơ bản:

|  |  |
| --- | --- |
| Cộng hai số hay nhiều số với nhau:  > 15+2997  [1] 3012 | Cộng và trừ:  > 15+2997-9768  [1] -6756 |
| Nhân và chia  > -27\*12/21  [1] -15.42857 | Số lũy thừa: (25 – 5)3  > (25 - 5)^3  [1] 8000 |
| Căn số bậc hai: Your browser may not support display of this image.  > sqrt(10)  [1] 3.162278 | Số pi (p)  > pi  [1] 3.141593  > 2+3\*pi  [1] 11.42478 |
| Logarit: loge  > log(10)  [1] 2.302585 | Logarit: log10  > log10(100)  [1] 2 |
| Số mũ: *e*2.7689  > exp(2.7689)  [1] 15.94109    > log10(2+3\*pi)  [1] 1.057848 | Hàm số lượng giác  > cos(pi)  [1] -1 |
| Vector  > 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  [1] 4 6 2 10 8 12 14 12 16 | > exp(x/10)  [1] 1.221403 1.349859 1.105171 1.648721 1.491825 1.822119 2.013753 1.822119  [9] 2.225541    > exp(cos(x/10))  [1] 2.664634 2.599545 2.704736 2.405079 2.511954 2.282647 2.148655 2.282647  [9] 2.007132 |
| Tính tổng bình phương (sum of squares): 12 + 22 + 32 + 42 + 52 = ?  > x <- c(1,2,3,4,5)  > sum(x^2)  [1] 55 | Tính tổng bình phương điều chỉnh (adjusted sum of squares): Your browser may not support display of this image.= ?  > x <- c(1,2,3,4,5)  > sum((x-mean(x))^2)  [1] 10  Trong công thức trên mean(x) là số trung bình của vector x. |
| Tính sai số bình phương (mean square): Your browser may not support display of this image.= ?  > x <- c(1,2,3,4,5)  > sum((x-mean(x))^2)/length(x)  [1] 2    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):  Phương sai: Your browser may not support display of this image.= ?  > x <- c(1,2,3,4,5)  > var(x)  [1] 2.5  Độ lệch chuẩn: Your browser may not support display of this image.:  > sd(x)  [1] 1.581139 |

**4. Cấu trúc vector, matrix, list:**

* Vector:

vector là một những kiểu dữ liệu cơ bản của R.

Để hiểu rõ vector của R các bạn có thể đọc tài liệu chi tiết về vector bằng lệnh:

?vector

Copy

## **Tạo một vector:**

Để tạo một vector như bài trước chúng ta đã thảo luận, có thể dùng hàm c() với các phần tử của vector:

s <- c("r", "hello", "stat", "software") s

Copy

[1] "r" "hello" "stat" "software"

Copy

Nếu nhu cầu của chúng ta là tạo một vector gồm các số liền kề, R cung cấp toán tử : rất hữu ích.

num <- -3:8 num

Copy

[1] -3 -2 -1 0 1 2 3 4 5 6 7 8

Copy

Lưu ý a:b trả về một vector với phần tử đầu là a các phần tử tiếp theo liền kề nhau một đơn vị trong đoạn [a:b]:

num <- 5.4:9 num

Copy

[1] 5.4 6.4 7.4 8.4

Copy

Ngoài ra bạn có thể tạo một chuổi liên tiếp với “bước nhảy” tùy ý với hàm seq(...) để hiểu rõ hơn hàm seq các bạn có thể đọc thêm tài liệu R với ?seq. Hai chức năng cơ bản mà mình muốn giới thiệu bao gồm:

* Tạo một vector tăng dần từ from đến to với bước nhảy by:

seq(from=1, to=5, by=0.5) # Viết gọn hơn seq(1,5,0.5)

Copy

[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0

Copy

* Tạo một vector tăng dần từ from đến to với kích thước length.out:

seq(from=1, to=11, length.out=5) # Viết gọn hơn seq(1, 11, length.out=5)

Copy

[1] 1.0 3.5 6.0 8.5 11.0

Copy

## **Xem độ dài của một vector:**

Để xem độ dài của một vector chúng ta có thể sử dụng hàm length() của R:

x <- c(0,1,2,3,9,8,7,4,5,6) length(x)

Copy

[1] 10

Copy

## **Trích phần tử từ vector:**

Để trích ra một phần tử của vector chúng ta có thể dùng toán tử [], chúng ta có thể trích duy nhất một phần tử hoặc một vector phần tử theo thứ tự:

x <- c(0,1,2,3,9,8,7,4,5,6) # Lấy ra phần tử thứ 4 x[4] # Lấy phần từ 2 3 4 5 x[2:5] # Lấy vector phần tử 1 3 5 x[c(1,3,5)]

Copy

[1] 3

[1] 1 2 3 9

[1] 0 2 9

Copy

## **Sửa phần tử trong vector:**

vector trong R chúng ta có thể gán lại giá trị phần tử trong vector như sau:

x[5] <- 256 x

Copy

[1] 0 1 2 3 256 8 7 4 5 6

Copy

Hay thay bằng một vector phần tử:

x[c(1,5,7)] <- c(-3,-2,-1) x

Copy

[1] -3 1 2 3 -2 8 -1 4 5 6

Copy

## **Thêm, xóa phần tử trong vector:**

vector trong R không cho phép thêm và xóa một phần tử bất kì. Vì vậy để thêm hoặc xóa phần tử, chúng ta chỉ có cách duy nhất là tạo một vector mới từ vector ban đầu:

a <- c(1,2,3,4,5,6) # Xoá phần tử thứ 3 bằng cách tạo vector mới từ 1:2 và 4:6 a <- c(a[1:2], a[4:6]) a

Copy

[1] 1 2 4 5 6

Copy

Tương tự với thêm phần tử.

## **Toán tử vector hóa:**

vector trong R cung cấp nhiều toán tử hữu ích giữa hai vector, trong nhiều trường hợp để tăng hiệu năng tính toán thì chúng ta nên dùng những toán tử này, vì chúng được tối ưu bởi các thư viện đại số ở bên dưới khá kĩ. Không nên truy xuất từng phần tử trong vector hay matrix để tính toán:

Một số các ví dụ về toán tử được vector hóa:

u <- c(1,2,3) v <- c(2,3,1) u > v

Copy

[1] FALSE FALSE TRUE

Copy

Thực hiện cộng với một scalar:

u + 10

Copy

[1] 11 12 13

Copy

Làm tròn số:

g <- c(3.4, 4.5, 5.6, 10.1) round(g)

Copy

[1] 3 4 6 10

Copy

Hoặc thậm chí cả tính căn bậc hai:

sqrt(g)

Copy

[1] 1.843909 2.121320 2.366432 3.178050

* Matrix:

matrix là một những kiểu dữ liệu cơ bản của R.

Để hiểu rõ matrix của R các bạn có thể đọc tài liệu chi tiết về matrix bằng lệnh:

?matrix

Copy

## **Tạo một ma trận matrix:**

Ma trận trong R có thể tạo với hàm matrix().

Chiều của ma trận có thể tự “suy diễn” bởi giá trị nrow và ncol truyền vào, việc truyền cả hai tham số vào đôi khi không cần thiết. Ví dụ sử dụng nrow:

m1 <- matrix(1:6, nrow=3) m1

Copy

[,1] [,2]

[1,] 1 4

[2,] 2 5

[3,] 3 6

Copy

Hay sử dụng ncol:

m2 <- matrix(1:6, ncol=3) m2

Copy

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

[1,] 1 3 5

[2,] 2 4 6

Copy

Một điều nếu bạn quan sát kĩ sẽ thấy, R khi dùng hàm matrix nó tự động lấy các giá trị của vector và phủ theo cột (từ trên xuống). Nếu bạn muốn phủ theo hàng thì cần sửa tham số mặc đinh byrow=FALSE về thành TRUE. Ở đây mình sẽ ví dụ cả hai trường hợp cùng lúc cho bạn đọc dể quan sát:

# Trường hợp mặc định byrow = FALSE (không cần khai báo gì thêm) m3 <- matrix(1:6, nrow=3) m3

Copy

[,1] [,2]

[1,] 1 4

[2,] 2 5

[3,] 3 6

Copy

# Trường hợp byrow = TRUE lúc này matrix sẻ phủ theo dòng m4 <- matrix(1:6, nrow=3, byrow=TRUE) m4

Copy

[,1] [,2]

[1,] 1 2

[2,] 3 4

[3,] 5 6

Copy

Một cách khác nữa để tạo ma trận là sử dụng hàm cbind() và hàm rbind() dùng để tạo các ma trận từ các vector cột hoặc vector hàng:

a.vec <- c(1, 1, 1) b.vec <- c(2, 2, 2) c.vec <- c(3, 3, 3) cbind(a.vec, b.vec, c.vec)

Copy

a.vec b.vec c.vec

[1,] "a" "b" "c"

[2,] "a" "b" "c"

[3,] "a" "b" "c"

Copy

Tương tự với hàm rbind() bạn đọc tự kiểm tra.

Lưu ý là matrix trong R cho phép đặt tên cho hàng và cột, như kết quả thu được ở trên, các cột tạo bởi hàm cbind tự động đặt tên là a.vec, b.vec, c.vec. Tuy nhiên đối với mình thì đây là một chức năng không cần thiết, nếu bạn đọc muốn tìm hiểu thêm có thể tìm hiểu hàm colnames() và rownames() trong R để thực hiện việc này.

## **Xem chiều của một matrix:**

Để xem chiều của một matrix trong R chúng ta có thể sử dụng hàm dim().

Giá trị đầu tiên trả về là số dòng nrow, giá trị tiếp theo là số cột ncol.

dim(m1)

Copy

[1] 3 2

Copy

## **Trích phần tử từ matrix:**

Để trích các phần tử từ ma trận chúng ta có thể sử dụng toán tử [vec.dim.m, vec.dim.n]. Ở đây mình lưu ý rằng mỗi chiều có thể lấy theo chỉ số index đánh thứ tự, nếu bỏ trống mặc định sẽ lấy hết toàn bộ:

* Lấy vector cột

# Lấy vector cột của m1 m1[,2]

Copy

[1] 4 5 6

Copy

* Lấy vector dòng

# Lấy vector dòng của m1 m1[3,]

Copy

[1] 3 6

Copy

* Lấy theo vector với dãy index:

# Lấy vector cột của m1 m1[c(1,3), ]

Copy

[,1] [,2]

[1,] 1 4

[2,] 3 6

Copy

## **Kiểm tra đối tượng có phải là matrix:**

Để kiểm tra một đối tượng có phải là matrix hay không, các bạn có thể dùng hàm is.matrix.

d <- warpbreaks # Dataframe từ dữ liệu warpbreaks str(d) # in ra thông tin đối tượng is.matrix(d) # Kiểm tra xem đối tượng có phải là một matrix không

Copy

'data.frame': 54 obs. of 3 variables:

$ breaks : num 26 30 54 25 70 52 51 26 67 18 ...

$ wool : Factor w/ 2 levels "A","B": 1 1 1 1 1 1 1 1 1 1 ...

$ tension: Factor w/ 3 levels "L","M","H": 1 1 1 1 1 1 1 1 1 2 ...

[1] FALSE

Copy

## **Ép kiểu đối tượng thành matrix:**

Một trong những hàm hữu ích của matrix trong R là hàm as.matrix, hàm này cho phép ép một số đối tượng về ma trận.

matrix.warpbreaks <- as.matrix(d) # Ép kiểu của data.frame bước trên is.matrix(matrix.warpbreaks)

Copy

[1] TRUE

Copy

Hoặc thậm chí có thể ép một vector về dạng matrix (n rows x 1 col) như sau:

col.vector <- as.matrix(1:6) col.vector

Copy

[,1]

[1,] 1

[2,] 2

[3,] 3

[4,] 4

[5,] 5

[6,] 6

Copy

## **Tính toán đại số với matrix:**

Đây là những phép tính quan trọng :D đặc biệt là phép nhân, nếu các bạn nào chưa quen thì cần cẩn thận với phép nhân ma trận, xác định phép nhân mà mình đang định dùng:

Gọi mat.A và mat.B là hai ma trận, vec.x và vec.y là hai vector, k là một số nguyên.

|  |  |
| --- | --- |
| **Toán tử hoặc hàm** | **Chú thích** |
| mat.A \* mat.B | Nhân từng phần từ |
| mat.A %\*% mat.B | Nhân ma trận |
| t(mat.A) | Ma trận chuyển vị |
| diag(vec.x) | Tạo ma trận đường chéo từ vector |
| diag(mat.A) | Lấy vector đường chéo từ ma trận |
| diag(k) | Tạo một ma trận đơn vị k x k |
| solve(mat.A, vec.y) | Trả về vector x nghiệm của hệ y = Ax  (với y là vec.y, và A là mat.A) |
| solve(mat.A) | Lấy ma trận nghịch đảo của A |
| res <- eigen(mat.A) | Lấy trị riêng vector riêng ma trận  - res$val là trị riêng của mat.A  - res$vec là vector riêng của mat.A |
| cbind(mat.A,mat.B,…) | Kết hợp nhiều ma trận lại theo chiều ngang |
| rbind(mat.A,mat.B,…) | Kết hợp nhiều ma trận lại theo chiều dọc |
| rowMeans(mat.A) | Tính trung bình theo dòng |
| rowSums(mat.A) | Tính tổng theo dòng |
| colMeans(mat.A) | Tính trung bình theo cột |
| colSums(mat.A) | Tính tổng theo cột |

* A\*B (element-wise multiplication): phép nhân từng phần tử của hai ma trận cùng chiều.

mat1 <- matrix(1:6, nrow=2) mat2 <- matrix(11:16, nrow=2) mat1 mat2 mat1\*mat2

Copy

# Kết quả ma trận 1

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

[1,] 1 3 5

[2,] 2 4 6

# Kết quả ma trận 2

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

[1,] 11 13 15

[2,] 12 14 16

# Kết quả nhân hai ma trận A\*B

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

[1,] 11 39 75

[2,] 24 56 96

Copy

* A %\*% B (matrix multiplication): nhân ma trận đại số.

mat3 <- matrix(1:6, nrow=2) mat4 <- matrix(2:4, nrow=3) mat3 %\*% mat4

Copy

[,1]

[1,] 31

[2,] 40

* List:

## *Tạo danh sách:*

Sau đây là một ví dụ để tạo một danh sách chứa chuỗi, số, vectơ và giá trị logic.

[Bản thử trực tiếp](http://tpcg.io/Vfm45u)

# Create a list containing strings, numbers, vectors and a logical

# values.

list\_data <- list("Red", "Green", c(21,32,11), TRUE, 51.23, 119.1)

print(list\_data)

Khi chúng tôi thực thi đoạn mã trên, nó tạo ra kết quả sau:

[[1]]

[1] "Red"

[[2]]

[1] "Green"

[[3]]

[1] 21 32 11

[[4]]

[1] TRUE

[[5]]

[1] 51.23

[[6]]

[1] 119.1

## *Đặt tên phần tử danh sách:*

Các phần tử danh sách có thể được đặt tên và chúng có thể được truy cập bằng các tên này.

[Bản thử trực tiếp](http://tpcg.io/PlyiUw)

# Create a list containing a vector, a matrix and a list.

list\_data <- list(c("Jan","Feb","Mar"), matrix(c(3,9,5,1,-2,8), nrow = 2),

list("green",12.3))

# Give names to the elements in the list.

names(list\_data) <- c("1st Quarter", "A\_Matrix", "A Inner list")

# Show the list.

print(list\_data)

Khi chúng tôi thực thi đoạn mã trên, nó tạo ra kết quả sau:

$`1st\_Quarter`

[1] "Jan" "Feb" "Mar"

$A\_Matrix

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

[1,] 3 5 -2

[2,] 9 1 8

$A\_Inner\_list

$A\_Inner\_list[[1]]

[1] "green"

$A\_Inner\_list[[2]]

[1] 12.3

## *Truy cập các phần tử danh sách:*

Các phần tử của danh sách có thể được truy cập bằng chỉ mục của phần tử trong danh sách. Trong trường hợp danh sách được đặt tên, nó cũng có thể được truy cập bằng cách sử dụng tên.

Chúng tôi tiếp tục sử dụng danh sách trong ví dụ trên -

[Bản thử trực tiếp](http://tpcg.io/2JMmWL)

# Create a list containing a vector, a matrix and a list.

list\_data <- list(c("Jan","Feb","Mar"), matrix(c(3,9,5,1,-2,8), nrow = 2),

list("green",12.3))

# Give names to the elements in the list.

names(list\_data) <- c("1st Quarter", "A\_Matrix", "A Inner list")

# Access the first element of the list.

print(list\_data[1])

# Access the thrid element. As it is also a list, all its elements will be printed.

print(list\_data[3])

# Access the list element using the name of the element.

print(list\_data$A\_Matrix)

Khi chúng tôi thực thi đoạn mã trên, nó tạo ra kết quả sau:

$`1st\_Quarter`

[1] "Jan" "Feb" "Mar"

$A\_Inner\_list

$A\_Inner\_list[[1]]

[1] "green"

$A\_Inner\_list[[2]]

[1] 12.3

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

[1,] 3 5 -2

[2,] 9 1 8

## *Thao tác các phần tử danh sách:*

Chúng ta có thể thêm, xóa và cập nhật các phần tử danh sách như hình dưới đây. Chúng tôi chỉ có thể thêm và xóa các phần tử ở cuối danh sách. Nhưng chúng tôi có thể cập nhật bất kỳ phần tử nào.

[Bản thử trực tiếp](http://tpcg.io/Weegyj)

# Create a list containing a vector, a matrix and a list.

list\_data <- list(c("Jan","Feb","Mar"), matrix(c(3,9,5,1,-2,8), nrow = 2),

list("green",12.3))

# Give names to the elements in the list.

names(list\_data) <- c("1st Quarter", "A\_Matrix", "A Inner list")

# Add element at the end of the list.

list\_data[4] <- "New element"

print(list\_data[4])

# Remove the last element.

list\_data[4] <- NULL

# Print the 4th Element.

print(list\_data[4])

# Update the 3rd Element.

list\_data[3] <- "updated element"

print(list\_data[3])

Khi chúng tôi thực thi đoạn mã trên, nó tạo ra kết quả sau:

[[1]]

[1] "New element"

$<NA>

NULL

$`A Inner list`

[1] "updated element"

## *Hợp nhất danh sách:*

Bạn có thể hợp nhất nhiều danh sách thành một danh sách bằng cách đặt tất cả các danh sách bên trong một hàm list ().

[Bản thử trực tiếp](http://tpcg.io/NMMVz7)

# Create two lists.

list1 <- list(1,2,3)

list2 <- list("Sun","Mon","Tue")

# Merge the two lists.

merged.list <- c(list1,list2)

# Print the merged list.

print(merged.list)

Khi chúng tôi thực thi đoạn mã trên, nó tạo ra kết quả sau:

[[1]]

[1] 1

[[2]]

[1] 2

[[3]]

[1] 3

[[4]]

[1] "Sun"

[[5]]

[1] "Mon"

[[6]]

[1] "Tue"

**4. Cấu trúc dataframe:**

##### **Khung Dữ Liệu**

Hàm data.frame()tạo ra các khung dữ liệu, các tập hợp các biến được kết hợp chặt chẽ với nhau chia sẻ nhiều thuộc tính của ma trận và danh sách, được sử dụng làm cấu trúc dữ liệu cơ bản bởi hầu hết các phần mềm mô hình của R.

**Từ khóa**

[các lớp](https://www.rdocumentation.org/search/keywords/classes) , [phương thức](https://www.rdocumentation.org/search/keywords/methods)

##### **Sử Dụng**

data.frame(…, row.names = NULL, check.rows = FALSE,

check.names = TRUE, fix.empty.names = TRUE,

stringsAsFactors = default.stringsAsFactors())

default.stringsAsFactors()

##### **Tranh Luận**

**…**

các đối số này có dạng valuehoặc tag = value. Tên thành phần được tạo dựa trên thẻ (nếu có) hoặc chính đối số đã rút gọn.

**row.names**

NULL hoặc một số nguyên hoặc một chuỗi ký tự chỉ định một cột được sử dụng làm tên hàng hoặc một ký tự hoặc vectơ số nguyên cung cấp tên hàng cho khung dữ liệu.

**check.rows**

nếu TRUEsau đó các hàng được kiểm tra về tính nhất quán của độ dài và tên.

**check.names**

hợp lý. Nếu TRUEsau đó tên của các biến trong khung dữ liệu được kiểm tra để đảm bảo rằng chúng là tên biến hợp lệ về mặt cú pháp và không bị trùng lặp. Nếu cần, chúng được điều chỉnh (bằng cách [make.names](https://www.rdocumentation.org/link/make.names?package=base&version=3.6.2)) để chúng như vậy.

**fix.empty.names**

lôgic cho biết nếu các đối số “không được đặt tên” (theo nghĩa không được gọi chính thức là someName = arg) nhận được một tên được xây dựng tự động hoặc đúng hơn là tên "". Cần được đặt thành FALSEngay cả khi check.namessai nếu ""tên nên được giữ nguyên.

**stringAsFactors**

logic: vectơ ký tự có nên được chuyển đổi thành thừa số không? Mặc định 'mới ban đầu' là TRUE, nhưng điều này có thể được thay đổi bằng cách cài đặt [options](https://www.rdocumentation.org/link/options?package=base&version=3.6.2)(stringsAsFactors = FALSE).

##### **Chi Tiết**

Khung dữ liệu là danh sách các biến có cùng số hàng với tên hàng duy nhất, lớp đã cho "data.frame". Nếu không có biến nào được đưa vào, tên hàng sẽ xác định số hàng.

Tên cột không được để trống và việc cố gắng sử dụng tên trống sẽ có kết quả không được hỗ trợ. Tên cột trùng lặp được phép, nhưng bạn cần sử dụng check.names = FALSEfor data.frame để tạo khung dữ liệu như vậy. Tuy nhiên, không phải tất cả các thao tác trên khung dữ liệu sẽ bảo toàn các tên cột bị trùng lặp: ví dụ tập hợp con giống ma trận sẽ buộc các tên cột trong kết quả là duy nhất.

data.framechuyển đổi từng đối số của nó thành khung dữ liệu bằng cách gọi [as.data.frame](https://www.rdocumentation.org/link/as.data.frame?package=base&version=3.6.2)(optional = TRUE). Vì đó là một hàm chung, các phương thức có thể được viết để thay đổi hành vi của các đối số theo các lớp của chúng: R đi kèm với nhiều phương thức như vậy. Các biến ký tự được chuyển đến data.frameđược chuyển đổi thành các cột nhân tố trừ khi được bảo vệ bởi [I](https://www.rdocumentation.org/link/I?package=base&version=3.6.2)hoặc đối số stringsAsFactorslà sai. Nếu một danh sách hoặc khung dữ liệu hoặc ma trận được chuyển đến data.framenó thì giống như thể mỗi thành phần hoặc cột đã được truyền dưới dạng một đối số riêng biệt (ngoại trừ ma trận được bảo vệ bởi [I](https://www.rdocumentation.org/link/I?package=base&version=3.6.2)).

Các đối tượng được chuyển đến data.framephải có cùng số hàng, nhưng vectơ nguyên tử (xem [is.vector](https://www.rdocumentation.org/link/is.vector?package=base&version=3.6.2)), các yếu tố và vectơ ký tự được bảo vệ bởi [I](https://www.rdocumentation.org/link/I?package=base&version=3.6.2)sẽ được tái chế nhiều lần nếu cần thiết (bao gồm cả các phần tử của đối số danh sách).

Nếu tên hàng không được cung cấp trong lệnh gọi tới data.frame, thì tên hàng được lấy từ thành phần đầu tiên có tên phù hợp, ví dụ: vectơ được đặt tên hoặc ma trận có tên hàng hoặc khung dữ liệu. (Nếu thành phần đó sau đó được tái chế, các tên sẽ bị loại bỏ kèm theo cảnh báo.) Nếu row.namesđược cung cấp NULLhoặc không tìm thấy thành phần phù hợp, tên hàng là dãy số nguyên bắt đầu từ một (và các tên hàng như vậy được coi là 'tự động', và không được bảo quản bởi [as.matrix](https://www.rdocumentation.org/link/as.matrix?package=base&version=3.6.2)).

Nếu tên hàng được cung cấp có độ dài là một và khung dữ liệu có một hàng duy nhất, thì giá trị row.namesđược lấy để chỉ định tên hàng chứ không phải cột (theo tên hoặc số).

Tên bị xóa khỏi đầu vào vectơ không được bảo vệ bởi [I](https://www.rdocumentation.org/link/I?package=base&version=3.6.2).

default.stringsAsFactorslà một tiện ích nhận [getOption](https://www.rdocumentation.org/link/getOption?package=base&version=3.6.2)("stringsAsFactors")và đảm bảo kết quả là TRUEhoặc FALSE(hoặc tạo ra một lỗi nếu giá trị không phải là NULL).

##### **Giá Trị**

Một khung dữ liệu, một cấu trúc giống như ma trận mà các cột có thể thuộc các kiểu khác nhau (số, lôgic, yếu tố và ký tự, v.v.).

Cách tạo tên của khung dữ liệu rất phức tạp và phần còn lại của đoạn này chỉ là câu chuyện cơ bản. Nếu các đối số là tất cả các đối tượng được đặt tên và đơn giản (không phải danh sách, ma trận của khung dữ liệu) thì tên đối số cung cấp tên cột. Đối với một đối số đơn giản không được đặt tên, phiên bản mô tả của đối số được sử dụng làm tên (vớiI(...)đã loại bỏ). Đối với đối số ma trận / danh sách / khung dữ liệu được đặt tên có nhiều hơn một cột được đặt tên, tên của các cột là tên của đối số theo sau là dấu chấm và tên cột bên trong đối số: nếu đối số không có tên, cột của đối số là tên được sử dụng. Đối với đối số ma trận / danh sách / khung dữ liệu có tên hoặc chưa đặt tên chứa một cột duy nhất, tên cột trong kết quả là tên cột trong đối số. Cuối cùng, các tên được điều chỉnh để trở thành duy nhất và hợp lệ về mặt cú pháp trừ khi check.names = FALSE.

##### **Ghi Chú**

Trong các phiên bản của R trước 2.4.0 row.namesphải là ký tự: để đảm bảo tính tương thích với các phiên bản R như vậy , hãy cung cấp một vectơ ký tự làm row.namesđối số.

##### **Người Giới Thiệu**

Chambers, JM (1992) *Dữ liệu cho các mô hình.* Chương 3 của *Mô hình Thống kê trong S xuất bản* JM Chambers và TJ Hastie, Wadsworth & Brooks / Cole.

[I](https://www.rdocumentation.org/link/I?package=base&version=3.6.2), [plot.data.frame](https://www.rdocumentation.org/link/plot.data.frame?package=base&version=3.6.2), [print.data.frame](https://www.rdocumentation.org/link/print.data.frame?package=base&version=3.6.2), [row.names](https://www.rdocumentation.org/link/row.names?package=base&version=3.6.2), [names](https://www.rdocumentation.org/link/names?package=base&version=3.6.2)(Đối với các tên cột), [[.data.frame](https://www.rdocumentation.org/link/%5B.data.frame?package=base&version=3.6.2)cho Subsetting phương pháp và I(matrix(..))ví dụ; [Math.data.frame](https://www.rdocumentation.org/link/Math.data.frame?package=base&version=3.6.2)vv, về các phương thức *Nhóm* cho data.frames; [read.table](https://www.rdocumentation.org/link/read.table?package=base&version=3.6.2), [make.names](https://www.rdocumentation.org/link/make.names?package=base&version=3.6.2).

**5. Import text file, csv, excel vào R:**

# **Excel to R**

Để import file Excel vào R, chúng ta có thể sử dụng package readxl.

library(readxl)

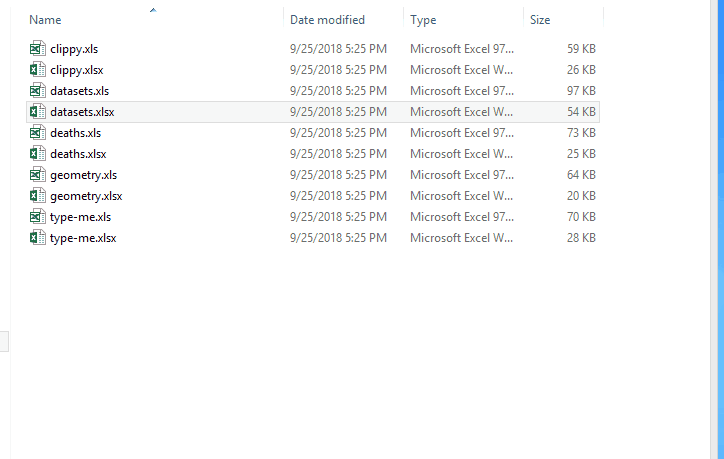
library(dplyr)

Giả sử, chúng ta đang có 1 file Excel datasets.xlsx có sẵn trong thư mục package readxl trong đường dẫn sau:

path <- readxl\_example("datasets.xlsx")

path

*## [1] "E:/Rlibrary/readxl/extdata/datasets.xlsx"*



File Excel datasets.xlsx có 4 sheets: iris, mtcars, chickwts, quakes.

Bây giờ để import dữ liệu iris (sheet đầu tiên trong file Excel này) vào R, chúng ta sẽ sử dụng hàm read\_excel() với câu lệnh như sau:

iris <- read\_excel(path, *# đường dẫn*

sheet = 1 *# số thứ tự sheet*

)

iris %>%

as.data.frame %>%

head

## Sepal.Length Sepal.Width Petal.Length Petal.Width Species

## 1 5.1 3.5 1.4 0.2 setosa

## 2 4.9 3.0 1.4 0.2 setosa

## 3 4.7 3.2 1.3 0.2 setosa

## 4 4.6 3.1 1.5 0.2 setosa

## 5 5.0 3.6 1.4 0.2 setosa

## 6 5.4 3.9 1.7 0.4 setosa

Tương tự như vậy, nếu chúng ta muốn import dữ liệu mtcars (sheet thứ 2 trong file Excel) vào R, chúng ta chỉ cần chọn option sheet = 2.

mtcars <- read\_excel(path, *# đường dẫn*

sheet = 2 *# số thứ tự sheet*

)

mtcars %>%

as.data.frame %>%

head

## mpg cyl disp hp drat wt qsec vs am gear carb

## 1 21.0 6 160 110 3.90 2.620 16.46 0 1 4 4

## 2 21.0 6 160 110 3.90 2.875 17.02 0 1 4 4

## 3 22.8 4 108 93 3.85 2.320 18.61 1 1 4 1

## 4 21.4 6 258 110 3.08 3.215 19.44 1 0 3 1

## 5 18.7 8 360 175 3.15 3.440 17.02 0 0 3 2

## 6 18.1 6 225 105 2.76 3.460 20.22 1 0 3 1

Như vậy, để import dữ liệu của tất cả các sheet trong file Excel vào R, chúng ta có thể import từng sheet trong file Excel đó vào R bằng việc sử dụng câu lệnh ở trên. Tuy nhiên, việc làm “manual” đó chỉ thích hợp với trường hợp file Excel chỉ gồm 1 vài sheet, còn đối với những trường hợp file Excel bao gồm cả hàng chục hoặc hàng trăm sheet thì chúng ta cần một giải pháp khác thông minh hơn để xử lý vấn đề trên.

Để giải quyết vấn đề trên, chúng ta có thể dùng cách sau:

library(openxlsx) *# package cần dùng*

wb\_obj <- loadWorkbook(path) *# thống kê những sheets có trong file excel*

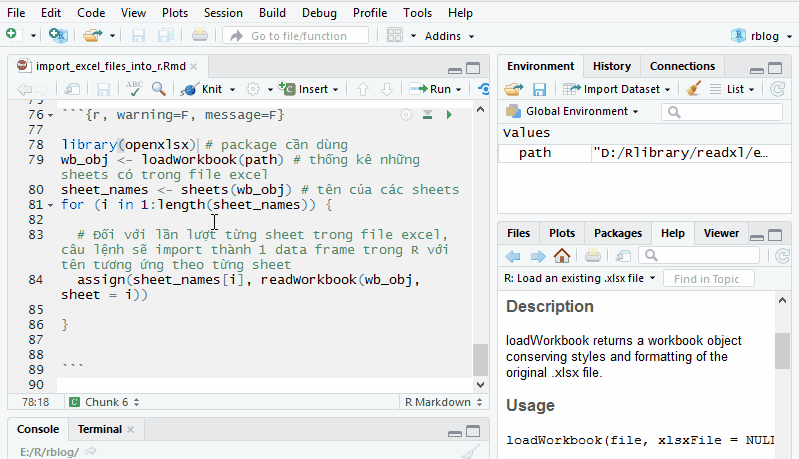
sheet\_names <- sheets(wb\_obj) *# tên của các sheets*

for (i in 1:length(sheet\_names)) {

*# Đối với lần lượt từng sheet trong file excel, câu lệnh sẽ import thành 1 data frame trong R với tên tương ứng theo từng sheet*

assign(sheet\_names[i], readWorkbook(wb\_obj, sheet = i))

}



Kết quả: Tất cả các sheet trong file Excel sẽ được import vào R thành các data frame tương ứng (trong Global Environment).

**6. Export số liệu từ R ra text file, csv, excel:**

# **R to Excel**

Như vậy, chúng ta đã biết làm thế nào để import dữ liệu từ file Excel vào R. Câu hỏi đặt ra lúc này là: Vậy làm thế nào để export dữ liệu từ R ra file Excel?

Câu trả lời là: Dùng hàm write.xlsx() trong package xlsx.

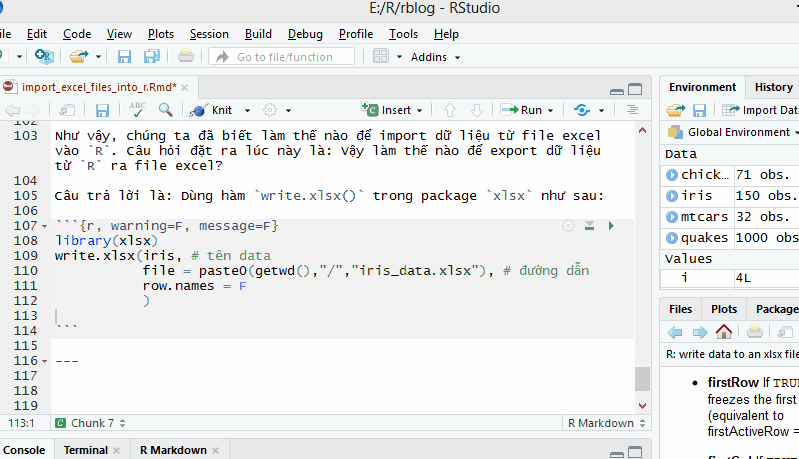
library(xlsx)

write.xlsx(iris, *# tên data*

file = paste0(getwd(),"/","iris\_data.xlsx"), *# đường dẫn*

row.names = F

)



Tuy nhiên, với cách sử dụng xlsx, máy tính đòi hỏi phải cài đặt Java. Do đó, hiện này có một cách thuận tiện hơn, đó là sử dụng package writexl

library(writexl)

writexl::write\_xlsx(iris, path = "./iris\_data.xlsx")

Kết quả: Dữ liệu iris đã được export từ R sang Excel (file iris\_data.xlsx).

Như vậy, chúng ta đã vừa được học cách import và export dữ liệu từ Excel vào R. Chúc các bạn học tập và làm việc hiệu quả với Ranalytics.vn!