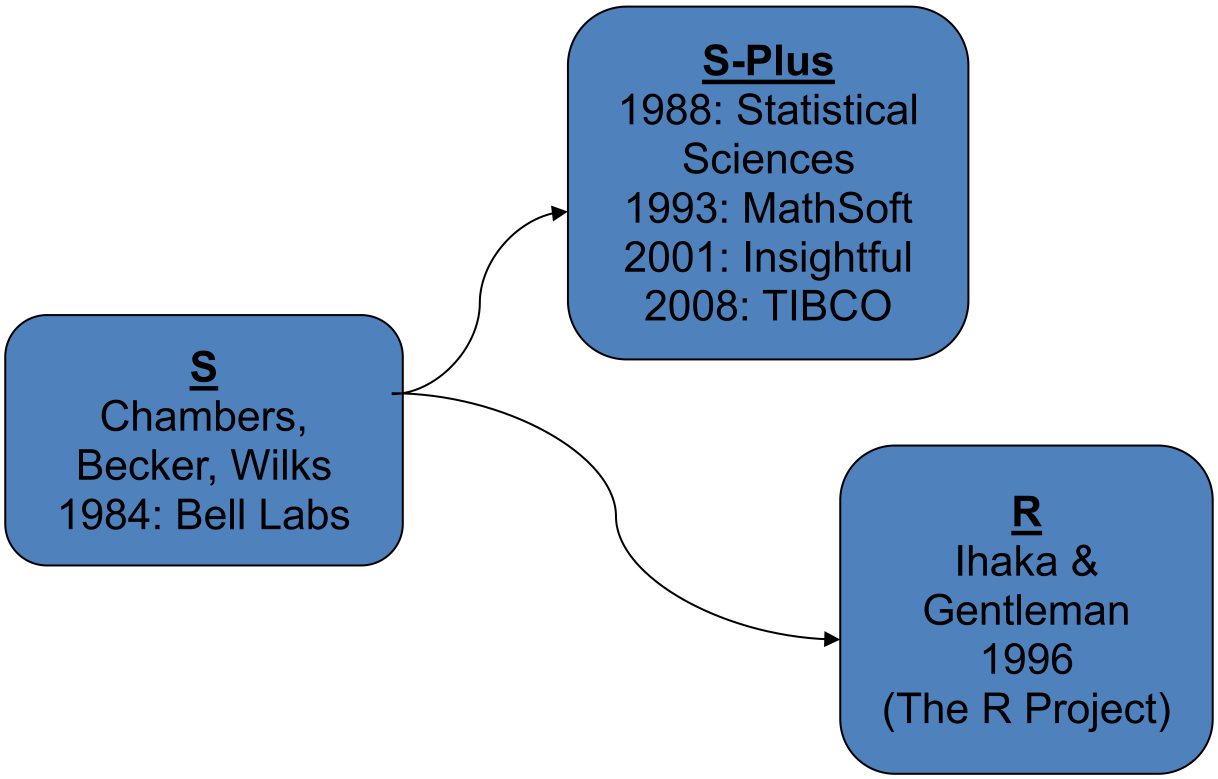
Lab 1: Giới thiệu về R

1 Giới thiệu chung

1.1 Lịch sử ra đời và phát triển

R là một ngôn ngữ lập trình và môi trường phần mềm cho phân tích thống kê, trình bày đồ thị và viết báo cáo. R được tạo ra bởi Ross Ihaka và Robert Gentleman tại Đại Học Auckland, New Zealand, và hiện tại được phát triển bởi "*R Development Core Team*". R xuất hiện lần đầu vào năm 1993. Khi đó một nhóm lớn các cá nhân đã đóng góp vào R bằng việc gửi code và báo cáo lỗi. Từ giữa 1997, có một nhóm trụ cột ("*R Development Core Team*" là những người có thể chỉnh sửa trên kho lưu trữ mã nguồn R .

Ngôn ngữ này được đặt tên "R", dựa trên ký tự đầu của tên của hai tác giả "R" (Robert Gentleman và Ross Ihaka), và một phần theo cách đặt tên của Ngôn ngữ S của Bell Labs.

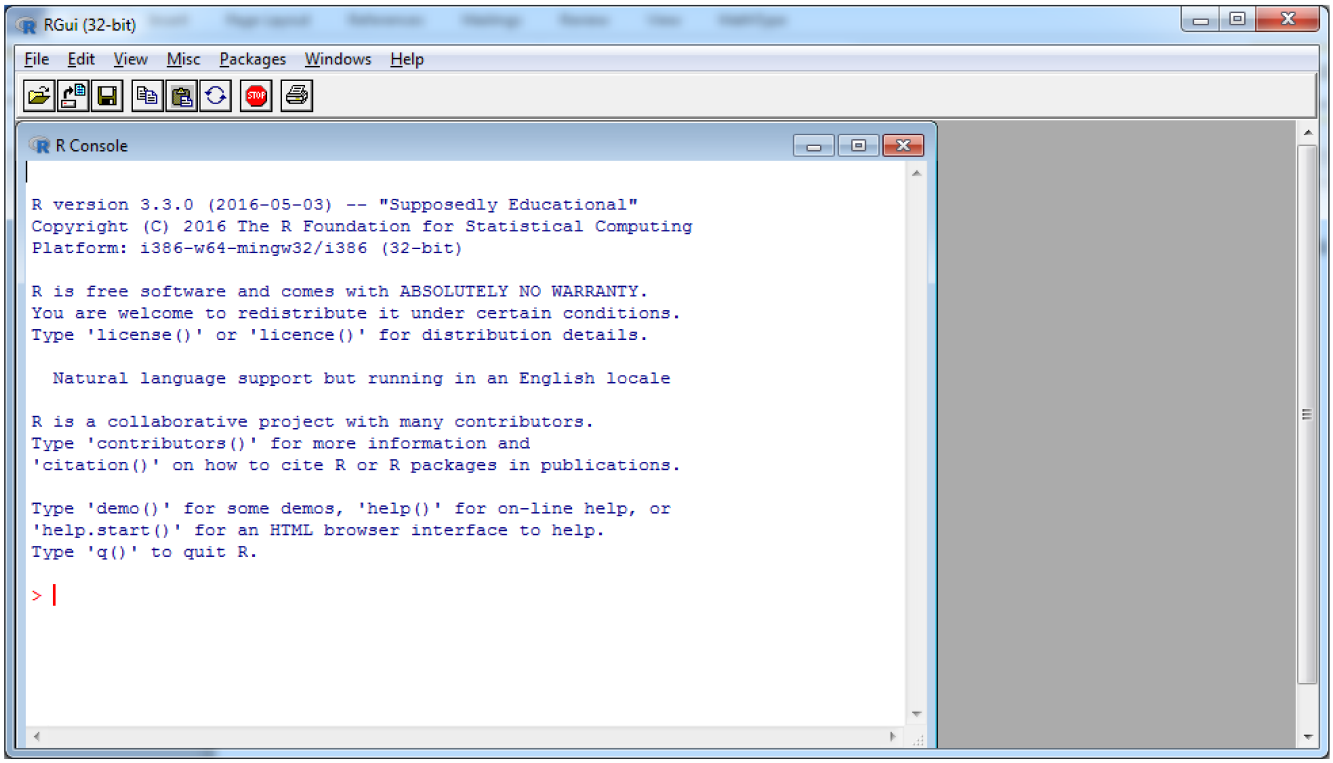


1.2 Cài đặt

Để cài đặt R trên máy tính, vào trang chủ của R : https://cran.r-project.org/bin/windows/base/ và (áp dụng cho hệ điều hành Windows) chọn "Download R 4.3.3. for Windows" (phiên bản mới nhất).

1.3 Giao diện làm việc

Khi mở R ta thấy xuất hiện cửa sổ chính là `RGUI (32-bit)` và một cửa sổ con trong đó là *R Console* (Bảng điều khiển lệnh). Các lệnh trong R được thực hiện qua cửa sổ *R Console* này.



1.4 Trợ giúp trong R

Lệnh `help(tên hàm)` hoặc `?tên hàm` sẽ trả ra trang thông tin mô tả về `tên hàm`.

Thí dụ: chạy thử lần lượt từng dòng lệnh sau

```
> help(matrix)
##
> ?(matrix)
```

2 Cấu trúc dữ liệu cơ bản trong R

Các kiểu đối tượng

`integer`: các số tự nhiên (tức là các số $0, 1, 2, 3, ...$);

`numeric`: các số thực;

`character`: chuỗi ký tự (thí dụ: "Hello World", "HCMUS", "A", ...);

`logical`: TRUE/FALSE (hoặc T/F);

Các lớp đối tượng

`vector`:	có thể chứa các giá trị thuộc kiểu `logical`, `integer`, `numeric`, `complex` hay `character` ;
`factor`:	các số thực;
`array`:	mảng nhiều chiều;
`matrix`:	ma trận (mảng hai chiều);
`list`:	vector các thành phần;
`dataframe`:	a data frame is a list of variables of the same number of rows with unique row names, given class "data.frame". If no variables are included, the row names determine the number of rows. A data frame, a matrix-like structure whose columns may be of differing types (numeric, logical, factor and character and so on);

Các giá trị đặc biệt

`NULL`:	đối tượng có chiều dài 0, có thể kiểm tra đối tượng `x` có phải `NULL` hay không với lệnh `is.null(x)`;
`NA`:	Not Available, giá trị bị khuyết (missing), kiểm tra với lệnh `is.na(x)`;
`NaN`:	Not a Number, kiểm tra với lệnh `is.nan(x)` (thí dụ $0/0, log(-1), ...$);
`Inf, -Inf`:	Positive/Negative Infinity, kiểm tra với lệnh `is.infinite(x)` (thí dụ $1/0$);

Một số hàm để kiểm tra thông tin của đối tượng

`str(x)`:	hiển thị cấu trúc của đối tượng `x`;
`class(x)`:	truy xuất lớp của đối tượng `x`;
`is.<mode>(x)`:	kiểm tra đối tượng `x` có phải kiểm *mode* hay không (thí dụ: `is.numeric(x)`, `is.logical(x)`, ...);
`attr(x, which)`:	nhận hoặc thiết lập thuộc tính của đối tượng `x`;

Thí dụ:

```
# create a 2 by 5 matrix
x <- 1:10
attr(x,"dim") <- c(2, 5)
```

3 Tạo `vector, matrix, data frame, list`

3.1 Tạo `vector`

Ta có thể tạo một vector bằng cách dùng lệnh `c()` để nối các phần tử lại với nhau.

```
> c(1,2,3,4,5)
[1] 1 2 3 4 5
> c("a","b","c","d","e")
[1] "a" "b" "c" "d" "e"
> c(T,F,T,F)
[1] TRUE FALSE TRUE FALSE
```

Dòng lệnh `1:5` sẽ tạo ra một dãy các số tự nhiên giữa 1 và 5.

```
> 1:5
[1] 1 2 3 4 5
> 5:1
[1] 5 4 3 2 1
```

■ Ta có thể tạo một dãy số cách đều nhau bằng hàm `seq()`. Hàm `seq()` cho phép ta lựa chọn khoảng cách giữa các số liên tiếp.

```
> seq(1,5)
[1] 1 2 3 4 5
> seq(1,5,by=.5)
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

■ Ta cũng có thể tạo một dãy số trong đó lặp lại một hoặc một số giá trị bằng cách dùng lệnh `rep()`.

```
> rep(1,5)
[1] 1 1 1 1 1
> rep(1:2,5)
[1] 1 2 1 2 1 2 1 2 1 2
```

■ Ta cũng có thể tạo một vector số gồm các giá trị trống bằng cách dùng `numeric()` và một vector `logical` chứa các giá trị `FALSE` bằng hàm `logical()`.

```
> numeric(5)
[1] 0 0 0 0 0

> logical(5)
[1] FALSE FALSE FALSE FALSE FALSE
```

■ Hàm `length(v)` trả về chiều dài của vector `v`.

```
> x <- seq(1,5,by=.5) # Create a sequence of integer numbers from 1 to 5
                      # and then assign it  to x
> x                     # Print values of x
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
> length(x)           # Find the length of vector x
[1] 9
```

3.2 Tạo `factor`

Hàm `factor()` chuyển đổi một `vector` thành một `factor`. Một `factor` cũng có thể được sắp thứ tự với tùy chọn `ordered = T` hoặc lệnh `ordered()`. Hàm `levels()` trả về các mức của một `factor`.
Hàm `gl()` sinh ra các `factor`.

Với `n` là số mức, `k` là số lần lặp lại của mỗi `factor` và `length` là tổng chiều dài của `factor`. Ngoài ra, `labels` là tùy chọn và cho các nhãn của mỗi mức. Các `factor` có thể được xem như các biến danh mục.
Một hàm quan trọng cho phân tích `factor` là hàm `table()`, xuất ra một kết quả tổng hợp thông tin của `factor`). Khi xét các loại dữ liệu thống kê (danh mục, thứ bậc, khoảng và tỷ lệ), các `factor` có thể là danh mục, thứ bậc, hoặc khoảng. Các `factor` danh mục là các tên danh mục, ví dụ trong số đó có thể là tên các quốc gia kết hợp với một số thông tin khác.
Một ví dụ về `factor` thức bậc là tập các lần đua của một vận động viên nào đó kèm theo vị trí hoàn thành của vận động viên này (nhất, nhì, ...).
Ngoài ra, một ví dụ về các `factor` mức khoảng sẽ là các khoảng tuổi như "20 - 29", "30 - 39", v.v.
Nói chung, R có thể thứ tự một cách tự động các số được lưu một cách phù hợp nhưng ta có thể sử dụng cùng các kỹ thuật với loại dữ liệu này để đặt thứ bậc theo kiểu phù hợp nhất với áp dụng.
Xem thêm các hàm `is.factor()` , `as.factor()` , `is.ordered()` và `as.ordered()` .

```
> factor(c("yes","no","yes","maybe","maybe","no","maybe","no","no"))
[1] yes no yes maybe maybe no maybe no no
Levels: maybe no yes
##
> factor(c("yes","no","yes","maybe","maybe","no","maybe","no","no"),
ordered = T)
[1] yes no yes maybe maybe no maybe no no
Levels: maybe < no < yes
##
> ordered(c("yes","no","yes","maybe","maybe","no","maybe","no","no"))
[1] yes no yes maybe maybe no maybe no no
Levels: maybe < no < yes
##
> ordered(as.factor(c("First","Third","Second","Fifth","First","First"
,"Third")), levels = c("First","Second","Third","Fourth","Fifth"))
[1] First Third Second Fifth First First Third
Levels: First < Second < Third < Fourth < Fifth
##
> gl(n=2, k=2, length=10, labels = c("Male", "Female")) # generate
factor levels
[1] Male Male Female Female Male Male Female Female Male
Male
Levels: Male Female
##
```

3.3 Tạo matrix

Nếu muốn tạo một ma trận ta sử dụng hàm `matrix()`. Khi đó, ta phải nhập một vector dữ liệu, số hàng (ứng với tham số `nrow =`) và/hoặc cột (ứng với tham số `ncol =`) và cuối cùng ta có thể xác định xem ta muốn R đọc vector dữ liệu này theo hàng (`byrow = T`) hay theo cột (`byrow = F`). Ta xét hai ví dụ sau

```
> matrix(data = NA, nrow = 5, ncol = 5, byrow = T)
[,1] [,2] [,3] [,4] [,5]
[1,] NA NA NA NA NA
[2,] NA NA NA NA NA
[3,] NA NA NA NA NA
[4,] NA NA NA NA NA
[5,]  NA NA NA NA NA
##
> matrix(data = 1:15, nrow = 5, ncol = 5, byrow = T)
[,1] [,2] [,3] [,4] [,5]
[1,] 1 2 3 4 5
[2,] 6 7 8 9 10
[3,] 11 12 13 14 15
[4,] 1 2 3 4 5
[5,] 6 7 8 9 10
```

■ Ngoài ra, cũng có thể tạo ma trận bằng các hàm `cbind()` hoặc `rbind()` để kết hợp các `vector` (cùng độ dài) thành ma trận theo cột hoặc theo hàng:

```
> v1 <- 1:5
> v2 <- 5:1
> v2
[1] 5 4 3 2 1
##
> cbind(v1,v2)
     v1 v2
[1,]  1  5
[2,]  2  4
```

```
[3,]   3   3
[4,]   4   2
[5,]   5   1
##
> rbind(v1,v2)
    [,1] [,2] [,3] [,4] [,5]
v1     1    2    3    4    5
v2     5    4    3    2    1
```

■ Số chiều (dimension) của một ma trận có thể được xác định bằng hàm `dim()`. Một cách khác là dùng hàm `nrow()` và `ncol()` tương ứng sẽ trả về số hàng và cột của ma trận:

```
> X <- matrix(data = 1:15, nrow = 5, ncol = 5, byrow = T)
> dim(X)
[1] 5 5
> nrow(X)
[1] 5
> ncol(X)
[1] 5
```

■ Hàm `t()` thực hiện chuyển vị một ma trận:

```
> t(X)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    6   11    1    6
[2,]    2    7   12    2    7
[3,]    3    8   13    3    8
[4,]    4    9   14    4    9
[5,]    5   10   15    5   10
```

Remark: Không giống data frame, ma trận phải có dạng số hoặc ký tự:

```
> a = matrix(2,2,2)
> a
     [,1] [,2]
[1,]    2    2
[2,]    2    2
##
> a = rbind(a,c("A","A"))
> a
      [,1] [,2]
[1,]  "2"  "2"
[2,]  "2"  "2"
[3,]  "A"  "A"
```

3.4 Tạo array (mảng)

Một mảng gồm n chiều với mỗi chiều là một vector về các đối tượng R cùng loại. Một mảng một chiều một phần tử có thể được xây dựng như sau bằng hàm `array()`.

```
> x = array(c(T,F),dim=c(1))
> print(x)
[1] TRUE
```

Mảng `x` được tạo với một chiều (ứng với tham số `dim=c(1)`) được rút từ vector các giá trị có thể `c(T,F)`. Một cách tương tự, y, có thể được tạo với một chiều và hai giá trị.

```
> y = array(c(T,F),dim=c(2))
> print(y)
[1] TRUE FALSE
```

Một mảng ba chiều $(3 \times 3 \times 3)$ có thể được tạo như sau.

```
> z = array(1:27,dim=c(3,3,3))
> dim(z)
[1] 3 3 3
##
> print(z)
, , 1
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

, , 2
     [,1] [,2] [,3]
[1,]   10   13   16
[2,]   11   14   17
[3,]   12   15   18

, , 3
     [,1] [,2] [,3]
[1,]   19   22   25
[2,]   20   23   26
[3,]   21   24   27
```

Các mảng trong R được truy xuất theo cách tương tự các mảng trong các ngôn ngữ khác: tức là, theo chỉ số nguyên, bắt đầu từ `1` (không phải `0`). Ví dụ sau đây cho ta thấy cách mà chiều thứ ba của mảng 3x3x3 có thể được truy xuất như thế nào.

```
> z[,,3]
     [,1] [,2] [,3]
[1,]   19   22   25
[2,]   20   23   26
[3,]   21   24   27
```

Xác định cụ thể hai trong ba chiều sẽ trả về một mảng một chiều.

```
> z[,3,3]
[1] 25 26 27
```

Xác định cụ thể ba trong ba chiều sẽ trả về một phần tử của mảng 3x3x3.

```
> z[3,3,3]
[1] 27
```

Ta cũng có thể phân hoạch mảng phức tạp hơn.

```
> z[,c(2,3),c(2,3)]
, , 1
     [,1] [,2]
[1,]   13   16
[2,]   14   17
[3,]   15   18

, , 2
     [,1] [,2]
[1,]   22   25
[2,]   23   26
[3,]   24   27
```

■ Các mảng không cần đối xứng theo mọi chiều. Đoạn code sau tạo một cặp các mảng 3x3.

```
> w = array(1:18,dim=c(3,3,2))
> print(w)
, , 1
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
, , 2
     [,1] [,2] [,3]
[1,]   10   13   16
[2,]   11   14   17
[3,]   12   15   18
```

■ Các đối tượng của các vector tạo nên mảng phải cùng loại, nhưng chúng không cần phải là số.

```
> u = array(c(T,F),dim=c(3,3,2))
> print(u)
, , 1
      [,1]  [,2]  [,3]
[1,]  TRUE FALSE  TRUE
[2,] FALSE  TRUE FALSE
[3,]  TRUE FALSE  TRUE

, , 2
      [,1]  [,2]  [,3]
[1,] FALSE  TRUE FALSE
[2,]  TRUE FALSE  TRUE
[3,] FALSE  TRUE FALSE
```

3.5 Tạo `list`

Một `list` là một tập các đối tượng trong R . Để tạo một `list` ta dùng hàm `list()`.
Hàm `unlist()` chuyển đổi một `list` thành một `vector`.

Remark: các đối tượng trong một danh sách KHÔNG nhất thiết phải cùng loại hoặc chiều dài.

```
> x <- c(1:4)
> y <- FALSE
> z <- matrix(c(1:4),nrow=2,ncol=2)
> myList <- list(x,y,z)
> myList
 [[1]]
[1] 1 2 3 4

 [[2]]
[1] FALSE

 [[3]]
     [,1] [,2]
[1,]    1    2
[2,]    3    4
```

■ Các `list` có một số phương pháp linh hoạt để tham chiếu. Tham chiếu theo chỉ số:

```
> a = list()
> a
list()
##
> a[[1]] = "A"
> a
[[1]]
[1] "A"
##
> a[[2]]="B"
> a
$$[[1]]
[1] "A"

[[2]]
[1] "B"
```

■ Tham chiếu theo tên:

```
> a
list()
> a$fruit = "Apple"
> a
$fruit
[1] "Apple"
##
> a$color = "green"
> a
$fruit
[1] "Apple"

$color
[1] "green"
```

■ Tham chiếu bằng đệ quy và kết hợp

```
> a = list()
> a[[1]] = "house"
> a$park = "green's park"
> a
[[1]]
[1] "house"
$park
[1] "green's park"
##
>a$park = "green's park"
> a[[1]]$address = "1 main st."

> a
[[1]]
[[1]][[1]]
[1] "house"
[[1]]$address
[1] "1 main st."
$park
[1] "green's park"
```

■ Sử dụng các quy tắc phạm vi trong R người ta cũng có thể đặt tên động và tạo các phần tử danh sách

```
> a = list()
> n = 1:3
> fruit = paste("number of coconuts in bin",n)
> my.number = paste("I have",3:1,"coconuts")
> a[fruit[1]] = my.number[1]
> a[fruit[2]] = my.number[2]
> a[fruit[3]] = my.number[3]
##
a$'number of coconuts in bin 3'
[1] "I have 1 coconut"
```

3.6 Tạo `data frame`

Một **`data frame`** có thể được xem như "một danh sách các biến/vector có cùng chiều dài".

Ta dùng lệnh `data.frame()` để tạo một **`data frame`**.

Trong ví dụ sau, một **`data frame`** có hai **`vector`** được tạo, mỗi **`vector`** có năm phần tử. Trong đó, **`vector`** đầu, `v1`, là một dãy các số nguyên từ 1 đến 5. Và **`vector`** thứ hai, `v2`, là 5 giá trị **`logical`** được rút từ loại `T` và `F`.
Sau đó, **`data frame`** được tạo từ các **`vector`** này. Các cột của **`data frame`** có thể được truy xuất bằng cách dùng chỉ số (index) nguyên hoặc tên cột và ký hiệu `$`.

```
> v1 = 1:5
> v2 = c(T,T,F,F,T)
> df = data.frame(v1,v2)
##
> print(df)
  v1    v2
1  1  TRUE
2  2  TRUE
3  3 FALSE
4  4 FALSE
5  5  TRUE
df[,1]
 [1] 1 2 3 4 5
df$v2
 [1] TRUE TRUE FALSE FALSE TRUE
```

■ Bên cạnh đó, **`data frame`** cũng có thể được tạo một cách trực tiếp. Trong đoạn code sau, data frame được tạo và đặt tên mỗi **`vector`** tạo nên **`data frame`** như là một phần của danh sách đối biến.

```
> df = data.frame(foo=1:5,bar=c(T,T,F,F,T))
> print(df)
  foo   bar
1   1  TRUE
2   2  TRUE
3   3 FALSE
4   4 FALSE
5   5  TRUE
```

4 Bài tập thực hành

Ta thấy nếu làm việc trên cửa sổ R Console, thì ta chỉ có thể thực hiện từng câu lệnh, và không thể lưu lại một cách thuận tiện các câu lệnh đã thực hiện.

$\Rightarrow$ dùng R script: trong R , vào File, chọn *New script.*

- Bây giờ, ta có thể viết nhiều câu lệnh trên file R script, và sau đó có thể lưu(*save*) file R script vào máy tính.
- Để chạy (*run*) các câu lệnh trong file R script:
  - ▷ Cách 1: nếu chạy một dòng lệnh: đặt con trỏ ở dòng lệnh muốn chạy và nhấn *Ctr + R*;
  - ▷ Cách 2: tô đen đoạn code muốn chạy và nhấn *Ctr + R*;

(Xem thêm trong tab Edit phần *Run line or selection*
và *Run all* (chạy tất cả câu lệnh trong file R script))

Bài 1. Tạo các `vector` `x` $= (4, 2, 6)$ và `y` $= (1, 0, -1)$. Dự đoán kết quả của các câu lệnh sau:

(a) `length(x);` (b) `sum(x) ;` (c) `x^2;` (d) `sum(x^2);`

(e) `x + y;` (f) `x*y ;` (g) `x-2;`

Dùng R để kiểm tra kết quả.

Bài 2. Dự đoán kết quả của các câu lệnh dưới đây, sau đó dùng R để kiểm tra kết quả:

(a) `10:16;` (b) `seq(2,9) ;` (c) `seq(4,10,by=2);`

(d) `seq(3,30,length=10);` (e) `seq(6,-4,by=-2) ;`

Bài 3. Dự đoán kết quả của các câu lệnh dưới đây, sau đó dùng R để kiểm tra kết quả:

(a) `rep(2,4);` (b) `rep(c(1,2), 4) ;` (c) `rep(c(1,2), c(4,4));`

(d) `rep(1:4, 4);` (e) `rep(1:4, rep(3,4)) ;`

Bài 4. Dùng hàm `rep()` để tạo các `vector` sau:

(a) `(6, 6, 6, 6, 6, 6);` (b) `(5, 8, 5, 8, 5, 8, 5, 8);` (c) `(5, 5, 5, 5, 8, 8, 8, 8);`

Bài 5. Cho `x <- c(5,9,2,3,4,6,7,0,8,12,2,9)`, xác định kết quả các câu lệnh dưới đây, sau đó dùng R để kiểm tra:

(a) `x[2];` (b) `x[2:4];` (c) `x[c(2,3,6)];`

(d) `x[c(1:5, 10:12)] ;` (e) `x[-(10:12)];`

Bài 6. Cho dữ liệu `y <- c(33,44,29,16,25,45,33,19,54,22,21,49,11,24,56)` chứa doanh thu bán sữa theo lít trong 5 ngày tại ba cửa hàng khác nhau (ba giá trị đầu là cho cửa hàng 1, 2 và 3 vào các ngày Thứ hai, Thứ ba, v.v.).

Tạo một tóm tắt thống kê về doanh thu cho từng ngày trong tuần. Sau đó, tạo một tóm tắt thống kê về doanh thu cho mỗi cửa hàng.

Hint: dùng lệnh `summary(y[1:3])` , ...

Bài 7. Tạo ma trận đơn vị có số chiều 3×3. Sau đó, tạo các ma trận

$$A = \begin{pmatrix} 3 & 2 \\ -1 & 1 \end{pmatrix} \quad \text{và} \quad B = \begin{pmatrix} 1 & 4 & 0 \\ 0 & 1 & -1 \end{pmatrix}$$

Tính các biểu thức dưới đây:

(a) `2*A`; (b) `A*A`; (c) `A%*%A`; (d) `A%*%B`;

(e) `t(B)`; (f) `solve(A)`; (Hint: dùng lệnh `help(solve)` để xem mô tả về hàm `solve()`)

Bài 8. Với ma trận A và B trong bài 7, thực hiện các yêu cầu sau:

(a) trích ra hàng thứ 1 của A; (b) trích ra cột thứ 2 của A;

(c) trích ra phần tử $B_{1;2}$ của B; (d) trích ra đồng thời cột thứ 2 và cột thứ 3 của B;

Bài 9. Dùng câu lệnh `load(tên dataset)` để load dữ liệu từ dataset có tên `mtcars` (có sẵn trong thư viện của R)

(a) Dùng lệnh `help(mtcars)` hãy cho biết bộ dữ liệu này mô tả dữ liệu gì?

(b) Tính trọng lượng trung bình và mức tiêu thụ nhiên liệu trung bình cho các xe trong bộ dữ liêu này.

Tài liệu tham khảo

[1] The R manuals, https://cran.r-project.org/manuals.html.

[2] Introduction to R, GS. Nguyễn Văn Tuấn
https://cran.r-project.org/doc/contrib/Intro_to_R_Vietnamese.pdf.