Chương 1

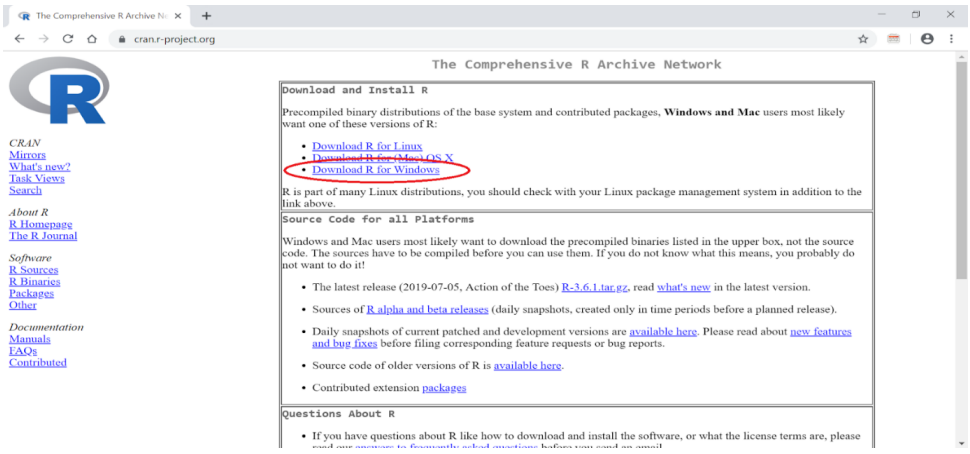
Giới thiệu về ngôn ngữ R

R là một ngôn ngữ lập trình và là một công cụ cho tính toán, máy học, thống kê và phân tích dữ liệu. R có thế mạnh về hiển thị trong thống kê và được sử dụng rộng rãi trong tài chính, y tế, sinh học cũng như cộng đồng các nhà khoa học, chuyên gia phân tích chính sách. R sử dụng cho bất kì hệ điều hành nào và được cài đặt miễn phí.

1.1 Cài đặt và các thao tác cơ bản

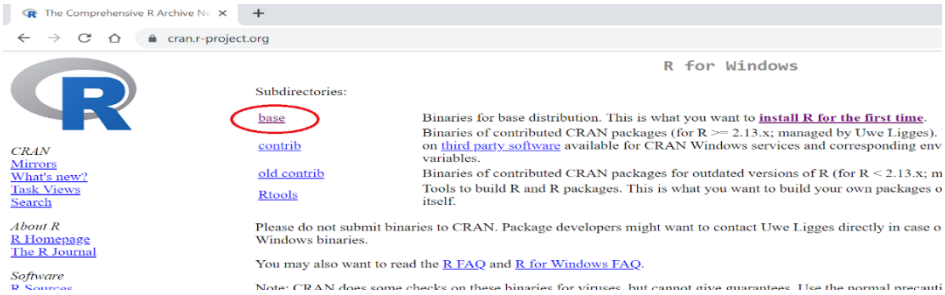
Các thao tác cài đặt trên Windows. Truy cập vào website `http://cran.R-project.org`:

Bước 1: Chọn *Download R for Windows*



Hình 1: Chọn *Download R for Windows*

Bước 2: Tiếp tục chọn *base*

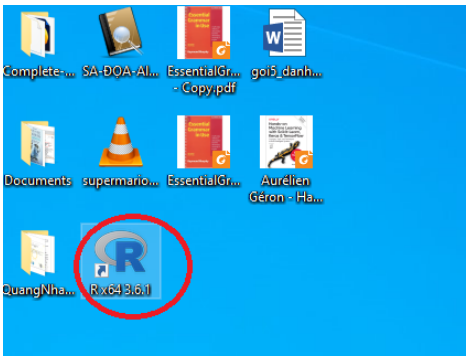


Hình 2: Tiếp tục chọn *base*

Bước 3: Chọn *Download R 3.6.1 for Windows* rồi cài đặt.

Hình 3: Chọn *Download R 3.6.1 for Windows* rồi cài đặt.

Bước 4: Sau khi hoàn tất việc cài đặt, một icon R sẽ xuất hiện trên desktop của máy tính.



Hình 4: Icon R xuất hiện trên desktop của máy tính.

1.2 Các phép toán cơ bản trong R

Ta có thể sử dụng R như một máy tính bỏ túi dựa trên các phép toán cơ bản sau:

- **Phép cộng:** +
- **Phép trừ:** -
- **Phép nhân:** *
- **Phép chia:** \\
- **Phép mũ:** ^
- **Phép chia lấy dư:** %%

Chẳng hạn, ta có đoạn code sau:

```
> 2^3+3^2
[1] 17
> 17%%4
[1] 1
```

Để gán một giá trị vào biến, ta sử dụng lệnh = hoặc <-. Chẳng hạn:

```
> a = "apple" #Gan gia tri "apple" vao bien a
> a
[1] "apple"

> b <- 1 #gan gia tri 1 vao bien b
> b
[1] 1
```

Ngoài ra, ta có thể sử dụng các hàm lượng giác như `sin`, `cos`, `tan`,... hoặc hàm lấy căn bậc hai `sqrt`, hàm trị tuyệt đối `abs`,...

Ví dụ

```
> sin(pi/2)
[1] 1
> cos(pi)
[1] -1
> sqrt(25)
[1] 5
> abs(-5)
[1] 5
```

1.3 Kiểu dữ liệu và định dạng biến

1.3.1 Kiểu dữ liệu trong R

Kiểu ma trận

Cú pháp:

```
mymatrix <- matrix(vector, nrow=r, ncol=c, byrow=FALSE, ...
   dimnames=list(char_vector_rownames, char_vector_colnames))
```

Trong đó, `nrow = r` là số dòng, `col = c` là số cột. Các tham số khác có thể tham khảo ở đây.

Ví dụ:

```
> A = matrix(
+     c(2,4,3,1,5,7), #Du lieu cac phan tu
+     nrow = 2, ncol = 3,
+     byrow = TRUE #Dien vao ma tran theo dong
+ )
> print(A)
     [,1] [,2] [,3]
[1,]    2    4    3
[2,]    1    5    7
```

Để truy cập đến các phần tử trong ma trận ta sử dụng dấu :, cụ thể như sau:

```
> print(A)
     [,1] [,2] [,3]
[1,]    2    4    3
[2,]    1    5    7

> A[,1] # trich xuat cot 1
[1] 2 1
> A[1,] # trich xuat dong 1
[1] 2 4 3
> A[2,2:3] # trich xuat phan tu thu 2 va 3 cua dong 2
[1] 5 7
```

Kiểu mảng (arrays)

Mảng là các đối tượng dữ liệu R, có thể được lưu trữ theo nhiều chiều. Ví dụ: Nếu chúng ta tạo một mảng có kích thước (2, 3, 4) thì nó tạo ra 4 ma trận, mỗi ma trận có kích thước 2×3.

Cú pháp:

```
array(vector,dim = c(nrow,ncolumn,nmatrix), dimnames = ...
    list(char_vector_rownames, char_vector_colnames, ...
    char_vector_matrixnames))
```

Chẳng hạn, ví dụ dưới đây nhằm tạo ra 2 ma trận, mỗi ma trận có kích thước 3×3, được tạo từ 2 vector có kích cỡ khác nhau:

```
# Tao 2 vecotr voi do dai khac nhau.
> vector1 <- c(5,9,3)
> vector2 <- c(10,11,12,13,14,15)
# Nhap 2 vector tren vao mang result
> result <- array(c(vector1,vector2),dim = c(3,3,2))
```

Ta thu được hai ma trận dưới đây:

```
> print(result)
, , 1

     [,1] [,2] [,3]
[1,]    5   10   13
[2,]    9   11   14
[3,]    3   12   15

, , 2

     [,1] [,2] [,3]
[1,]    5   10   13
[2,]    9   11   14
[3,]    3   12   15
```

Để trích xuất giá trị từ mảng ta sử dụng các lệnh sau:

```
> result[, , 1] # trich xuat ma tran 1
     [,1] [,2] [,3]
[1,]    5   10   13
[2,]    9   11   14
[3,]    3   12   15
> result[1, , 1] # trich xuat dong 1 ma tran 1
```

```
[1]  5 10 13
> result[, 1, 1] # trich xuat cot 1 ma tran 1
[1] 5 9 3
> result[2:3, , 2] # trich xuat dong 2, 3 cua ma tran 2
     [,1] [,2] [,3]
[1,]    9   11   14
[2,]    3   12   15
```

Data frame

Data frame là dữ liệu dạng bảng có mối liên hệ 2 chiều (hàng và cột). Mỗi chiều đại diện cho 1 trường có cùng kiểu dữ liệu và mỗi dòng đại diện cho một bản ghi hay quan sát.

Cú pháp:

```
data.frame(..., row.names = NULL, check.rows = FALSE,
check.names = TRUE, fix.empty.names = TRUE,
stringsAsFactors = default.stringsAsFactors())
```

Để hiểu thêm về `data.frame` ta có thể gõ lệnh `?data.frame` trong cửa sổ command của R. Chẳng hạn, để tạo một `data.frame` quản lý nhân viên ta dùng các lệnh dưới đây:

```
> # Tao data frame.
> emp.data <- data.frame(
+    emp_id = c (1:5), # chi so id nhan vien
+    emp_name = c("Rick","Dan","Michelle","Ryan","Gary"), # ...
   ten nhan vien
+    salary = c(623.3,515.2,611.0,729.0,843.25), # muc luong
+    start_date = as.Date(c("2012-01-01", "2013-09-23", ...
   "2014-11-15", "2014-05-11",
+                           "2015-03-27")), # Ngay bat dau lam ...
   viec
+    stringsAsFactors = FALSE)
> # Print the data frame.
> print(emp.data)
   emp_id emp_name salary start_date
1      1     Rick   623.30 2012-01-01
2      2      Dan   515.20 2013-09-23
3      3 Michelle   611.00 2014-11-15
4      4     Ryan   729.00 2014-05-11
5      5     Gary   843.25 2015-03-27
```

Để kiểm tra định dạng các biến trong một `data.frame` ta dùng hàm `str(dataframe_name)`. Chẳng hạn:

```
> str(emp.data)
'data.frame': 5 obs. of 4 variables:
$ emp_id   : int 1 2 3 4 5
$ emp_name : chr "Rick" "Dan" "Michelle" "Ryan" ...
$ salary   : num 623 515 611 729 843
$ start_date: Date, format: "2012-01-01" "2013-09-23" ...
```

Sử dụng hàm `summary` để có kết quả về giá trị trung bình, median, min, max, bách phân vị thứ 1 và 3. Chẳng hạn:

```
> summary(emp.data)
emp_id     emp_name           salary        start_date
Min.   :1  Length:5         Min.   :515.2  Min.   :2012-01-01
1st Qu.:2  Class :character 1st Qu.:611.0  1st Qu.:2013-09-23
Median :3  Mode  :character Median :623.3  Median :2014-05-11
Mean   :3                  Mean   :664.4  Mean   :2014-01-14
3rd Qu.:4                  3rd Qu.:729.0  3rd Qu.:2014-11-15
Max.   :5                  Max.   :843.2  Max.   :2015-03-27
```

List

List là một đối tượng trong R chứa nhiều phần tử có định dạng khác nhau như numeric, string, vector, và thậm chí là trong list có thể chứa các list khác. Một list có thể chứa cả hàm, data frame, array và matrix như là những phần tử của nó.

Để hiểu thêm về `list` ta có thể gõ lệnh `?list` trong cửa sổ command của R.

Ví dụ: Để tạo một list gồm chuỗi, số, vector và cả giá trị logic ta làm như sau:

```
> # Tao mot list chua kieu chuoi, so, vector, gia tri logic
> list_data <- list("Red", "Green", c(21,32,11), TRUE, ...
   51.23, 119.1)
> print(list_data)
[[1]]
[1] "Red"

[[2]]
[1] "Green"
```

```
[[3]]
[1] 21 32 11

[[4]]
[1] TRUE

[[5]]
[1] 51.23

[[6]]
[1] 119.1
```

1.3.2 Định dạng biến trong R

Trong R người ta thường sử dụng các biến có kiểu dữ liệu (hay định dạng) sau:

- Kiểu số nguyên (**integer**): `2, ...`
- Kiểu số thực (**numeric** hoặc **double**): `2.3; 4.678;` $\pi, \ldots$. Lưu ý: một số nguyên cũng là kiểu số thực.
- Kiểu logic: `TRUE` hoặc `FALSE`.
- Kiểu chuỗi (**character**): Nằm giữa cặp nháy đơn `'` hoặc `"`

Để xem một biến thuộc kiểu dữ liệu gì, ta dùng lệnh `typesof()` hoặc `class`. Chẳng hạn:

```
> typeof(1.5)
[1] "double"
> class("Chuong trinh R")
[1] "character"
```

1.4 Đọc và ghi dữ liệu

1.4.1 Nhập trực tiếp

Nhập dữ liệu qua lệnh `c()` và `data.frame`

Ví dụ: Nhập số liệu về độ tuổi và cân nặng của 10 người thông qua các biến là age và weight được cho bởi bảng dưới đây:

Age	Weight
10	24
20	39
35	46
67	60
54	45
80	50
15	30
47	52
66	55
29	67

Ta nhập như sau:

```
> age <- c(10, 20, 35, 67, 54, 80, 15, 47, 66, 29)
> weight <- c(24, 39, 46, 60, 45, 50, 30, 52, 55, 67)
> x <- data.frame(age, weight)
> print(x)
   age  weight
1   10     24
2   20     39
3   35     46
4   67     60
5   54     45
6   80     50
7   15     30
8   47     52
9   66     55
10  29     67
```

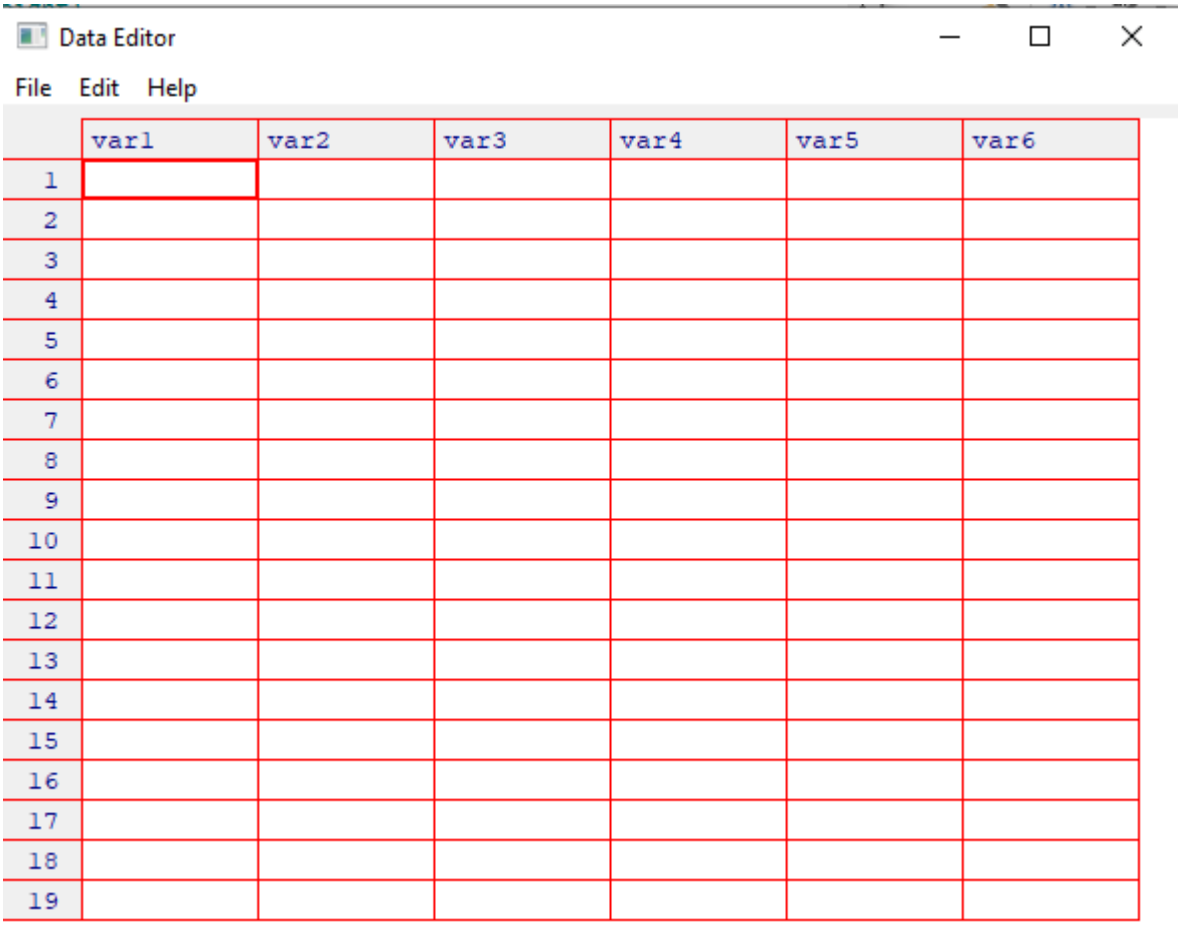
Nhập dữ liệu trực tiếp thông qua lệnh `edit(data.frame())`

Nhập `edit(data.frame())` vào cửa sổ Command trong R, ta thu được một bảng để nhập giá trị như sau:

1.4.2 Nhập từ file .txt

Để nhập dữ liệu vào R chúng ta sử dụng lệnh `read.table` như sau:

```
> setwd("Noi luu tru du lieu")
> data1 <- read.table ("data1.txt", header=TRUE)
```



Hình 5: Bảng nhập giá trị trực tiếp

Ví dụ: Chúng ta thu thập dữ liệu về độ tuổi và lượng đường trong máu từ những bệnh nhân mắc bệnh tiểu đường và nó được lưu trong file có tên là `data2.txt` tại đường dẫn `C:\user\sugar`. Ta làm như sau:

```
> setwd("C:\users\sugar")
> data2 <- read.table ("data2.txt", header=TRUE)
```

Để kiểm tra lại, ta có thể sử dụng các lệnh sau:

```
> data2
> names(data2) # liet ke ten cac cot cua du lieu
> save (data2, file="data2.txt") # luu lai file
```

1.4.3 Nhập từ file Excel

Ta sử dụng lệnh `read.csv` để nhập dữ liệu vào R, tương tự như lệnh `read.table` ở trên. Cụ thể:

```
> setwd("C:/users/sugar")
> x <- read.csv(excel.txt, header=TRUE)
> save (x, file= "x.rda")
```

Ngoài ra, để nhập dữ liệu từ file `.xls` và `.xlsx` ta sử dụng thư viện `readxl` và lệnh `read_excel`. Cụ thể như sau:

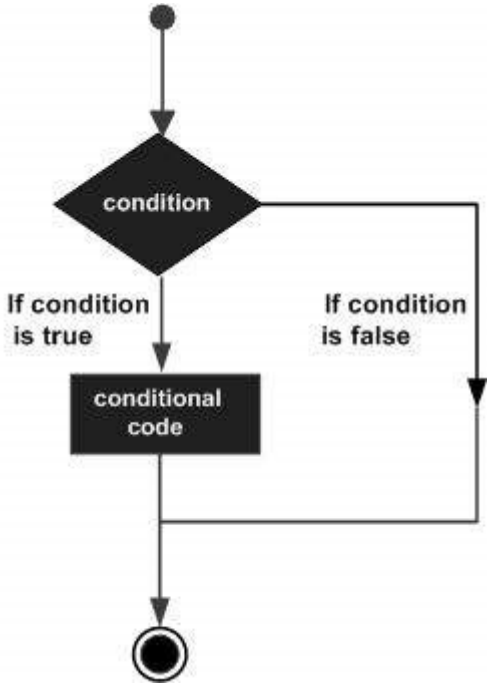
```
library("readxl")
# xls files
myData <- read_excel("my_file.xls")
# xlsx files
myData <- read_excel("my_file.xlsx")
```

1.5 Các cấu trúc điều khiển

1.5.1 Cấu trúc `if - else`

Cấu trúc `if - else` được mô tả như hình 6, cú pháp được cho dưới đây:

```
if(Dieu kien) {
# Cau lenh duoc thuc hien khi thoa dieu kien
} else {
# Cau lenh duoc thuc hien khi dieu kien sai.
}
```



Hình 6: Lược đồ Cấu trúc `if - else`

Chẳng hạn với đoạn code sau:

```
x <- c("what","is","truth")
if("Truth" %in% x) {
print("Truth is found")
} else {
print("Truth is not found")
}
```

Ta thu được kết quả sau:

```
[1] "Truth is not found"
```

Ngoài ra, ta có thể sử dụng cấu trúc `if - else if - else` như sau:

```
if(boolean_expression 1) {
   // Executes when the boolean expression 1 is true.
} else if( boolean_expression 2) {
   // Executes when the boolean expression 2 is true.
} else if( boolean_expression 3) {
   // Executes when the boolean expression 3 is true.
```

```
} else {
   // executes when none of the above condition is true.
}
```

Chẳng hạn:

```
x <- c("what","is","truth")

if("Truth" %in% x) {
print("Truth is found the first time")
} else if ("truth" %in% x) {
print("truth is found the second time")
} else {
print("No truth found")
}
```

ta thu được kết quả sau:

```
[1] "truth is found the second time"
```

1.5.2 Vòng lặp `for`

Cấu trúc `for` được mô tả như hình 7, cú pháp được cho dưới đây:

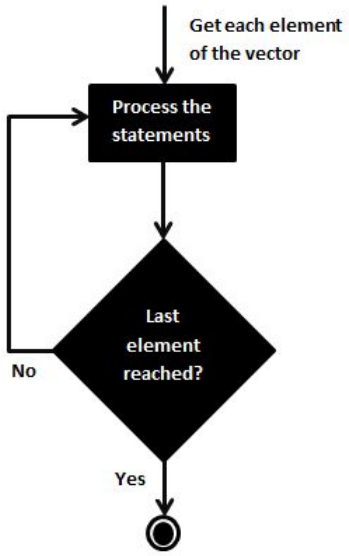
```
for (variable in vector){
# cau lenh
}
```

Chẳng hạn với đoạn code sau:

```
v <- LETTERS[1:4]
for ( i in v) {
print(i)
}
```

Ta thu được kết quả sau:

```
[1] "A"
[1] "B"
[1] "C"
[1] "D"
```



Hình 7: Lược đồ Cấu trúc `for`

1.5.3 Vòng lặp `while`

Tương tự như vòng lặp `for`, cấu trúc `while` được mô tả như hình 8, cú pháp được cho dưới đây:

```
while (Dieu kien) {
# Cau lenh
}
```

Chẳng hạn với đoạn code sau:

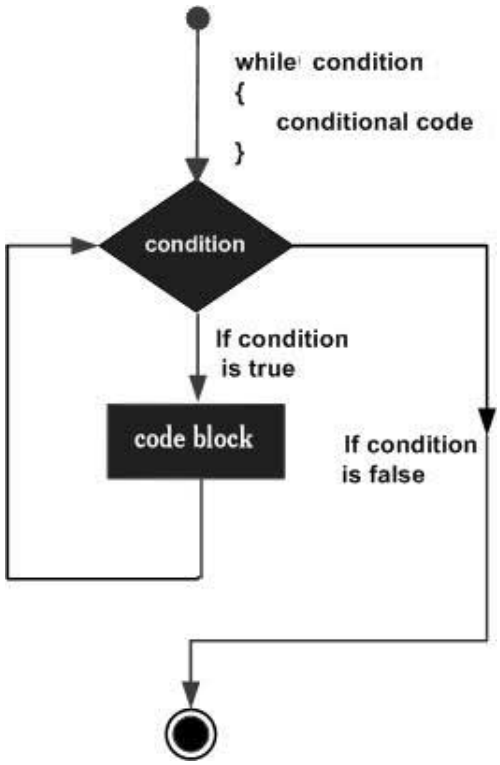
```
v <- c("Hello","while loop")
cnt <- 2

while (cnt < 7) {
print(v)
cnt = cnt + 1
}
```

Ta thu được kết quả sau:

```
[1] "Hello" "while loop"
[1] "Hello" "while loop"
[1] "Hello" "while loop"
```

```
[1] "Hello" "while loop"
[1] "Hello" "while loop"
```



Hình 8: Lược đồ Cấu trúc `while`