# Một số khái niệm cơ bản trong R

Khoa Toán - Cơ - Tin học
Trường Đại học Khoa học Tự nhiên
Đại học Quốc gia Hà Nội

Ngày 13 tháng 9 năm 2023

# Nội dung

# Nội dung

# Các phép toán cơ bản trong R

Các phép toán số học trong R gồm có:

- Phép cộng: `123 + 234`
- Phép trừ: `3234 - 532`
- Phép nhân: `123 * 456`
- Phép chia: `100 / 25`
- Lũy thừa: `2 ^ 3`
- Chia lấy phần nguyên: `8 %/% 3`
- Chia lấy phần dư: `8 %% 3`

# Các phép toán so sánh

- Phép toán nhỏ hơn: `<`
- Phép toán lớn hơn: `>`
- Phép toán nhỏ hơn hoặc bằng: `<=`
- Phép toán lớn hơn hoặc bằng: `>=`
- Phép toán bằng: `==`

# Các phép toán cơ bản trong R

Các phép toán logic trong R:

- Toán tử và: `&` hoặc `&&`
- Toán tử hoặc `|` hoặc `||`
- Phép phủ định: `!`

```
> 3<5
[1] TRUE
> 3<5&5<10
[1] TRUE
> 6!=2
[1] TRUE
> !6!=2
[1] FALSE
> 3<=4
[1] TRUE
> 3<5|5>10
[1] TRUE
```

Lưu ý: Các toán tử này không chỉ áp dụng cho những phần tử đơn lẻ, mà trong R có tính "vectơ hóa", giúp chúng ta thực hiện các phép toán giữa 2 vector mà không cần dùng vòng lặp.

```
> x = c(1, 3, 5, 7, 9)
> y = c(2, 4, 6, 8, 10)
> #Cộng véc tơ với một số
> x + 2
[1]  3  5  7  9 11
> #Nhân một véc tơ với một số
> 2*x
[1]  2  6 10 14 18
> #Chia véc tơ cho một số
> x/5
[1] 0.2 0.6 1.0 1.4 1.8
> #Nâng lũy thừa một véc tơ
> x^2
[1]  1  9 25 49 81
```

```
> #Cộng hai véc tơ
> x + y
[1]  3  7 11 15 19
> #Nhân hai véc tơ với nhau
> x*y
[1]  2 12 30 56 90
> #Chia hai véc tơ cho nhau
> x/y
[1] 0.5000000 0.7500000
[3] 0.8333333 0.8750000
[5] 0.9000000
> #Lũy thừa các phần tử trong x với số mũ trong y
> x^y
[1]          1         81
[3]      15625    5764801
[5] 3486784401
> x<5
[1]  TRUE  TRUE FALSE FALSE
[5] FALSE
```

```
> !x<5
[1] FALSE FALSE  TRUE  TRUE
[5]  TRUE
> y==6
[1] FALSE FALSE  TRUE FALSE
[5] FALSE
> (x<5)&(y==6)
[1] FALSE FALSE FALSE FALSE
[5] FALSE
> (x<5)|(y==6)
[1]  TRUE  TRUE  TRUE FALSE
[5] FALSE
```

# Nội dung

# Biến trong R

Trong R, khi cần lưu lại kết quả của các phép tính hoặc các hàm phục vụ cho việc tính toán về sau, ta sẽ lưu các kết quả này vào các biến trong R Trong R, ta không cần khởi tạo biến, mà chỉ cần đặt tên biến và gán cho nó một giá trị khi cần thiết. Ví dụ

```
var_x = 34.5
```

# Nội dung

# Các kiểu dữ liệu trong R

Trong R có các kiểu dữ liệu cơ bản sau:

- character: Dạng chuỗi ký tự

```
fruit = "Apple"
```

- integer: Dạng số nguyên

```
integer_variable = 187
```

- numeric: Dạng số thực

```
weight = 63.5
height = 182
```

# Các kiểu dữ liệu trong R

- complex: Dạng số phức

```
complex_value = 3 + 2i
```

- logical: gồm 2 giá trị `TRUE` và `FALSE`. Ví dụ:

```
bool1 = TRUE
```

# Nội dung

# Các cấu trúc dữ liệu trong R

Trong R có các cấu trúc dữ liệu sau:

- vector
- list
- factor
- matrix
- data.frame

# Các cấu trúc dữ liệu trong R

## Cấu trúc: vector

Là một mảng dùng để lưu trữ các dữ liệu có cùng kiểu với nhau. Để tạo một vector, ta sử dụng hàm `c()`

```
# x là một vector
x = c(1, 3, 5, 7, 8)
```

Để lấy giá trị một phần tử trong vector, ta sử dụng cú pháp sau:

```
x[2] # [1] 3
x[-1] # [1] 3 5 7 8
x[c(TRUE, TRUE, FALSE, FALSE, TRUE)] # [1] 1 3 8
```

Lưu ý: Khác với nhiều ngôn ngữ khác (C++, Java, JS, . . . ), chỉ số các phần tử trong R được đánh số từ 1, thay vì từ 0.

Là một danh sách, có thể dùng để lưu các biến ở nhiều kiểu dữ liệu khác nhau. Để tạo 1 list, ta sử dụng hàm `list()`

```
list1 <- list(
    a = c(1, 2, 3),
    b = c("Apple", "Orange"),
    c = c(TRUE, FALSE)
)
```

Để lấy các phần tử trong 1 list, ta sử dụng dấu `$` như sau:

```
list1$a
# Kết quả: 1 2 3
```

Là một vector chuyên dùng để lưu trữ các biến định tính. Để tạo factor từ một vector, ta dùng hàm `factor()`

```
sex_vec <- c("female", "female", "female", "male",
↪  "female", "male", "male", "female")
sex_fac <- factor(sex_vec)
```

Ta có thể truy cập các phần tử của factor tương tự như vector

Là một mảng 2 chiều dùng để lưu trữ các dữ liệu có cùng kiểu với nhau. Để tạo một matrix, ta dùng hàm `matrix()`

```
matrix1 <- matrix(data=c(-3,2,893,0.17),ncol=2)
```

Để truy cập các phần tử trong matrix, ta làm tương tự như vector, nhưng lưu ý: matrix là mảng 2 chiều, nên sẽ cần 2 chỉ số

```
matrix[1, 1]
# Kết quả: -3
```

Là một bảng dữ liệu, với mỗi cột dữ liệu là một vector. Chúng ta sẽ tìm hiểu sâu hơn về data.frame trong các bài sau.

# Nội dung

# Hàm và cách sử dụng hàm trong R

Trong R, hàm là một tập hợp các câu lệnh được sắp xếp để thực hiện 1 công việc cụ thể. Trong thư viện chuẩn của R đã có sẵn rất nhiều hàm thông dụng trong tính toán xác suất thống kê.
Các hàm trong R có cấu trúc như sau:

```
<function_name>(arg1, arg2 = <default>, ...)
```

Trong đó:

- `<function_name>` là tên của hàm
- Các tham số `arg1`, `arg2` là các tham số của hàm, trong đó:
  - `arg1` được gọi là tham số bắt buộc. Để sử dụng hàm, ta bắt buộc phải truyền tham số này vào hàm.
  - `arg2` được gọi là tham số tùy chọn. Khi sử dụng hàm, nếu ta không truyền tham số này vào hàm, hàm sẽ sử dụng giá trị mặc định `<default>` để tính toán.

# Hàm và cách sử dụng hàm trong R

Để gọi hàm trong R, ta làm như sau:

```
# Nếu không cần lưu kết quả của hàm
<function_name>(<args>)
# Nếu cần lưu kết quả của hàm
result <- <function_name>(<args>)
```

Trong đó `<function_name>` là tên của hàm mà ta muốn gọi, sau đó ta truyền các tham số vào phần `<args>`, theo 3 cách:

# Hàm và cách sử dụng hàm trong R

- Truyền tham số theo tên: Ta có thể thực hiện khai báo này ở vị trí bất kỳ trong hàm:

```
# Tham số có tên <arg> sẽ nhận giá trị <value>
<function_name>(<arg> = <value>, ...)
```

- Truyền tham số theo vị trí: giá trị được liệt kê không đi kèm tên tham số thì giá trị này được gán cho tham số xác định theo thứ tự trong danh sách tham số của hàm (do R, người viết hàm định nghĩa):

```
# Tham số thứ nhất trong hàm sẽ nhận <value1>, tham số
↪  thứ hai nhận <value2>
<function_name>(<value1>, <value2>, ...)
```

# Hàm và cách sử dụng hàm trong R

- Sử dụng cả hai cách truyền tham số. Ta hoàn toàn có thể truyền tham số theo cả hai cách trên khi gọi hàm:

```
# Tham số thứ nhất trong hàm sẽ nhận <value1>, tham số tên <argument_n> nhận <value2>
<function_name>(<value1>, <argument_n> = <value2>, ...)
```

Ví dụ:

```
> diem = 1:10
> sd(x = diem, na.rm = FALSE)
[1] 3.02765
> sd(na.rm = FALSE, x = diem)
[1] 3.02765
> sd(diem, na.rm = FALSE)
[1] 3.02765
```

```
> sd(na.rm = FALSE,diem)
[1] 3.02765
> sd(diem,FALSE)
[1] 3.02765
> sd(FALSE,x=diem)
[1] 3.02765
> sd(FALSE,diem)
[1] NA
Warning message:
In if (na.rm) "na.or.complete" else "everything" :
the condition has length > 1 and only the first element
 ↪  will be used
```