R for Data Science

Bài 6: *Vector và Factor*

Phòng LT & Mạng

https://csc.edu.vn/lap-trinh-va-csdl/R-Programming-Language-for-Data-Science_150

2020

Nội dung

1. Vector
2. Factor

Vector

❑ Giới thiệu

- Vector là đối tượng dữ liệu cơ bản nhất của R. Có 6 loại vector: logical, integer, double, complex, character và raw.

- Trong một vector có thể chứa một phần tử giá trị trong R (Single Element Vector). Vector này có chiều dài = 1 và thuộc một trong 6 loại vector nêu trên. Và vector cũng có thể chứa nhiều phần tử (Multiple Elements Vector)

Vector

❑ Single Element Vector

```
# Atomic vector of type character.
print("abc")

# Atomic vector of type double.
print(12.5)

# Atomic vector of type integer.
print(63L)

# Atomic vector of type logical.
print(TRUE)

# Atomic vector of type complex.
print(2+3i)

# Atomic vector of type raw.
print(charToRaw('hello'))
```

```
> source('~/R/chuong7/demo_vector.R')
[1] "abc"
[1] 12.5
[1] 63
[1] TRUE
[1] 2+3i
[1] 68 65 6c 6c 6f
```

Vector

Data Science

❑Multiple Elements Vector

- Cách 1: Sử dụng dấu : khi là dữ liệu kiểu số
  - Ví dụ:

```
v <- 1:10
print(v)

v <- 1.6:5.6
print(v)

# If the final element specified does not belong to the sequence then it is discarded.
v <- 5.8:11.4
print(v)
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
[1] 1.6 2.6 3.6 4.6 5.6
[1]  5.8  6.8  7.8  8.8  9.8 10.8
```

Vector

Data Science

❑Multiple Elements Vector

- Cách 2: Sử dụng seq(start, end, by = incrementing)
  - Ví dụ:

```
# Create vector with elements from 2 to 10 incrementing by 0.5
v <- seq(2, 10, by = 0.5)
print(v)
```

```
[1]  2.0  2.5  3.0  3.5  4.0  4.5  5.0  5.5  6.0  6.5  7.0  7.5  8.0  8.5  9.0  9.5 10.0
```

Vector

Data Science

❑ Multiple Elements Vector

- Cách 3: Sử dụng c(element1, element2, ...)
  - Ví dụ:

```
colors <- c("red", "green", "blue", "white")
print(colors)
```

```
[1] "red"   "green" "blue"  "white"
```

Vector

Data Science

❑ Truy xuất phần tử trong vector

- Phần tử trong vector truy xuất thông qua chỉ mục
- Sử dụng: [index] với index bắt đầu từ 1
- Có thể sử dụng **index âm để xóa phần tử**
- Ví dụ:

```
colors <- c("red", "green", "blue", "black", "white", "yellow", "orange")
print(colors)
# get items 1, 3, 5
c1 <- colors[c(1, 3, 5)]
print(c1)
# get items 2, 4
c2 <- colors[c(-2, -4)]
print(c2)
```

```
[1] "red"    "green"  "blue"   "black"  "white"  "yellow" "orange"
[1] "red"   "blue"  "white"
[1] "red"    "blue"   "white"  "yellow" "orange"
```

Data Science

❑Tính toán các vector

- Trường hợp 1: Hai vector **có cùng chiều dài** và có **dữ liệu kiểu số** có thể cộng, trừ, nhân, chia => tạo thành vector mới.
- Ví dụ:

```
> # Create two vectors.
> v1 <- c(5, 7, 4, 8, 0, 11)
> v2 <- c(4, 10,2, 8, 1, 2)
>
> # Vector addition.
> add.result <- v1+v2
> print(add.result)
[1]  9 17  6 16  1 13
>
> # Vector substraction.
> sub.result <- v1-v2
> print(sub.result)
[1]  1 -3  2  0 -1  9
>
> # Vector multiplication.
> multi.result <- v1*v2
> print(multi.result)
[1] 20 70  8 64  0 22
>
> # Vector division.
> divi.result <- v1/v2
> print(divi.result)
[1] 1.25 0.70 2.00 1.00 0.00 5.50
```

Data Science

- Trường hợp 2: Hai vector có chiều dài khác nhau và có dữ liệu kiểu số thì vector nào có chiều dài ngắn hơn sẽ quay vòng tuần tự khi cộng, trừ, nhân, chia => tạo thành vector mới.
- Ví dụ:

```
> # Create two vectors.
> v1 <- c(5, 7, 4, 8, 0, 11)
> v2 <- c(4, 10)
> print(v1)
[1]  5  7  4  8  0 11
> print(v2)
[1]  4 10
> add.result <- v1+v2
> print(add.result)
[1]  9 17  8 18  4 21
```

Data Science

❑Sắp xếp các phần tử trong vector

- Sử dụng phương thức sort()
- Ví dụ

```
v <- c(1, 2, 5, -10, 7, -6, 9)
v
```

```
1  2  5 -10  7 -6  9
```

```
# asc
print(sort(v))
```

```
[1] -10  -6   1   2   5   7   9
```

```
# desc
print(sort(v, decreasing = TRUE))
```

```
[1]  9  7  5  2  1 -6 -10
```

```
colors
```

```
'red'  'green'  'blue'  'black'  'white'  'yellow'  'orange'
```

```
# asc
print(sort(colors))
```

```
[1] "black"  "blue"   "green"  "orange" "red"    "white"  "yellow"
```

```
# desc
print(sort(colors, decreasing = TRUE))
```

```
[1] "yellow" "white"  "red"    "orange" "green"  "blue"   "black"
```


Nội dung

Data Science

1. Vector
2. Factor

Factor

Data Science

- Là đối tượng dữ liệu được sử dụng để phân loại dữ liệu và lưu trữ nó dưới dạng các level.
- Factor có thể lưu trữ cả string và integer.
- Factor rất hữu ích trong column với số lượng giới hạn các giá trị duy nhất, ví dụ như "Male, "Female" hoặc True, False...
- Factor cũng rất hữu ích trong phân tích dữ liệu thống kê.

Factor

Data Science

- Sử dụng factor(vector) để tạo factor từ vector
  - Ví dụ

```
gender <- c("Male", "Female", "Female", "Male", "Female", "Male")

is.factor(gender)

FALSE

factor_gender <- factor(gender)
is.factor(factor_gender)

TRUE

print(factor_gender)

[1] Male   Female Female Male   Female Male
Levels: Female Male
```

❑Thay đổi trật tự của level trong factor

```
new_order_factor <- factor(factor_gender, levels = c("Male", "Female"))
print(new_order_factor)
```

```
[1] Male   Female Female Male   Female Male
Levels: Male Female
```

❑Tạo factor level sử dụng gl(n,k,labels)

- n: số level, k: số lần lặp, labels: vector chứa các lables

```
colors_factor <- gl(3, 5, labels = c("Red", "Green", "Blue"))
print(colors_factor)
```

```
 [1] Red   Red   Red   Red   Red   Green Green Green Green Green Blue  Blue
[13] Blue  Blue  Blue
Levels: Red Green Blue
```


Chapter 6: Vector - Factor

Exercise 1: Vector Friends

- Tạo 1 vector trong đó chứa tên 3 người bạn của bạn
- Lần lượt thêm vào vector các người bạn cho đến khi không muốn thêm nữa
- Cho biết trong vector có bao nhiêu bạn. In tên của các bạn trong vector
- Cho biết tên 2 bạn đầu vector
- Cho biết tên bạn thứ 2 và bạn thứ 3 trong vector
- Cho biết tên 2 bạn cuối cùng trong vector
- Sắp xếp danh sách tăng dần
- Sắp xếp danh sách giảm dần

Exercise 2: Vector Numbers

- Tạo vector vec1 chứa các số: 2, 5, 8, 12, 16
- Tạo vector vec2 chứa các số từ 5 đến 9
- Tạo vector vec3 bằng cách vec1 – vec2
- Tạo vector 4 có 100 phần tử, phần từ đầu tiên = 2, mỗi phần tử sau lớn hơn phần tử trước 2 đơn vị (gợi ý: dùng seq())
- In giá trị của các phần tử ở vị trí 5, 10, 15, 20 của vector 4
- In giá trị của các phần tử ở vị trí từ 10 tới 30 của vector 4

Exercise 3: Vector Alphabets

- Tạo 1 vector chứa bảng chữ cái tiếng anh (alphabets). Gợi ý: dùng letters[]
- Tạo 1 vector chứa các nguyên âm
- Tạo 1 vector chứa các phụ âm. Gợi ý: những phần tử trong alphabets không có nguyên âm
- Tạo 1 vector chứa các phần tử, mỗi phần tử là một chuỗi gồm có một phụ âm và một nguyên âm

Exercise 4: Factor Numbers

- Tạo 1 vector có nội dung như sau: x <- c(1, 2, 3, 3, 5, 3, 2, 4, NA, 11, 22, 47, 47, 11, 47, 11)
- Tạo 1 factor từ vector trên
- In ra các levels trong factor
- In ra số lượng levels trong factor

Exercise 5: Factor Strings

- Tạo 1 vector có nội dung như sau: v1<- c("low", "high", "medium", "high", "low", "medium", "high")
- Tạo 1 factor từ vector trên
- In ra các levels trong factor
- In ra số lượng levels trong factor
- Thêm mới một level 'medium high' vào factor
- In ra các levels trong factor sau khi thêm mới
- In ra số lượng levels trong factor sau khi thêm mới
- Thống kê số lượng phần tử theo levels

Gợi ý:

Exercise 1: Vector Friends

In [1]:
```
names.vector <- c("Minh", "Thanh", "Xuan")
i<-1
repeat{
  len <-length(names.vector)
  names.vector[len+1] <- readline(prompt = "Input your friend's name: ")
  i <- as.integer(readline(prompt = "Continue: 1, Stop: !=1 "))
  if(i!=1){
    break
  }
}
```

```
Input your friend's name: Tuan
Continue: 1, Stop: !=1 1
Input your friend's name: Hoang
Continue: 1, Stop: !=1 1
Input your friend's name: Nguyen
Continue: 1, Stop: !=1 0
```

In [2]:
```
print(paste("There are", length(names.vector), "friends in vector:",
            toString(names.vector)))
print(paste("Two first friends:", toString(names.vector[1:2])))
print(paste("Second and third friends:", toString(names.vector[c(2,3)])))
print(paste("Two last friends:",
            toString(names.vector[c(length(names.vector),
                                    length(names.vector)-1)])))

print(paste("Sort of my firends:", toString(sort(names.vector))))
print(paste("Descending Sort of my firends:",
            toString(sort(names.vector, decreasing = TRUE))))
```

```
[1] "There are 6 friends in vector: Minh, Thanh, Xuan, Tuan, Hoang, Nguyen"
[1] "Two first friends: Minh, Thanh"
[1] "Second and third friends: Thanh, Xuan"
[1] "Two last friends: Nguyen, Hoang"
[1] "Sort of my firends: Hoang, Minh, Nguyen, Thanh, Tuan, Xuan"
[1] "Descending Sort of my firends: Xuan, Tuan, Thanh, Nguyen, Minh, Hoang"
```

Exercise 2: Vector Numbers

In [4]:
```
vec1 <- c(2, 5, 8, 12, 16)
vec2 <- c(5:9)
print(paste("vec1:", toString(vec1)))
print(paste("vec2:", toString(vec2)))
print(paste("vec subtract:", toString(vec1-vec2)))
# tao vector 100 phan tu voi phan tu dau tien la 2,
# tang 2 don vi cho moi phan tu
vec3 <-seq(from=2, by=2, length.out = 100)
print(paste("vec3:", toString(vec3)))
print(paste("values at indexs 5, 10, 15, 20:",
            toString(vec3[c(5, 10, 15, 20)])))
print(paste("values from indexs 10 to 30:",
            toString(vec3[10:30])))
```

```
[1] "vec1: 2, 5, 8, 12, 16"
[1] "vec2: 5, 6, 7, 8, 9"
[1] "vec subtract: -3, -1, 1, 4, 7"
[1] "vec3: 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36,
38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76,
78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98, 100, 102, 104, 106, 108, 110, 112,
114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144,
146, 148, 150, 152, 154, 156, 158, 160, 162, 164, 166, 168, 170, 172, 174, 176,
178, 180, 182, 184, 186, 188, 190, 192, 194, 196, 198, 200"
[1] "values at indexs 5, 10, 15, 20: 10, 20, 30, 40"
[1] "values from indexs 10 to 30: 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 4
2, 44, 46, 48, 50, 52, 54, 56, 58, 60"
```

Exercise 3: Vector Alphabets

In [5]:
```
alphabets <- letters[1:26]
vowels <-c('a', 'e', 'i','o','u')
consonants <- alphabets[!(alphabets %in% vowels)]
print(paste("alphabets:", toString(alphabets)))
print(paste("vowels:", toString(vowels)))
print(paste("consonants:", toString(consonants)))

consonants_vowels <- sub(" ", "", paste(consonants,vowels))
print(paste("consonants:", toString(consonants_vowels)))
```

```
[1] "alphabets: a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u,
v, w, x, y, z"
[1] "vowels: a, e, i, o, u"
[1] "consonants: b, c, d, f, g, h, j, k, l, m, n, p, q, r, s, t, v, w, x, y, z"
[1] "consonants: ba, ce, di, fo, gu, ha, je, ki, lo, mu, na, pe, qi, ro, su, t
a, ve, wi, xo, yu, za"
```

Exercise 4: Factor Numbers

In [6]:
```
x <- c(1, 2, 3, 3, 5, 3, 2, 4, NA, 11, 22, 47, 47, 11, 47, 11)
print(paste("Vector", toString(x)))
# tao factor
ft <- factor(x)
#print(ft)
print(paste("Levels in factor:", toString(levels(ft))))
print(paste("Number of levels in factor:", nlevels(ft) ))
```

```
[1] "Vector 1, 2, 3, 3, 5, 3, 2, 4, NA, 11, 22, 47, 47, 11, 47, 11"
[1] "Levels in factor: 1, 2, 3, 4, 5, 11, 22, 47"
[1] "Number of levels in factor: 8"
```

Exercise 5: Factor Strings

In [9]:
```
v1<- c("low", "high", "medium", "high", "low", "medium", "high")
print(paste("Vector:", toString(v1)))
food <- factor(v1)
levels_food <- levels(food)
nums_levels <- nlevels(food)

print(paste("Levels:", toString(levels(food))))
print(paste("Number of levels:", nlevels(food)))

# add level medium high
levels_food[nums_levels+1] <- "medium high"
food <- factor(v1, levels = levels_food)
print(paste("Levels after having 'medium high':", toString(levels(food))))
print(paste("Number of levels after having 'medium high':", nlevels(food)))
```

```
[1] "Vector: low, high, medium, high, low, medium, high"
[1] "Levels: high, low, medium"
[1] "Number of levels: 3"
[1] "Levels after having 'medium high': high, low, medium, medium high"
[1] "Number of levels after having 'medium high': 4"
```

In [10]:
```
print("Summary:")
for(lv in levels(food)){
  count = 0
  for(e in v1){
    if(lv == e){
      count = count + 1
    }
  }
  print(paste(lv,":", count,"element"))
}
```

```
[1] "Summary:"
[1] "high : 3 element"
[1] "low : 2 element"
[1] "medium : 2 element"
[1] "medium high : 0 element"
```