R 语言编程：基于 tidyverse

第 03 讲　数据结构 I: 向量, 矩阵, 多维数组

张敬信

2022 年 2 月 10 日

哈尔滨商业大学

- **数据结构**是为了便于存储不同类型的数据而设计的**数据容器**。
- 学习数据结构，就是要把各个数据容器的特点、适合存取什么样的数据理解透彻，只有这样才能在实际中选择最佳的数据容器。数据容器选择的合适与否，直接关系到代码是否高效简洁，甚至能否解决问题。
- R 常用数据结构：
    - 存放同类型数据：向量、矩阵、多维数组
    - 存放不同类型数据：列表、数据框
    - 分类变量数据：因子
    - 其它数据：字符串、日期时间数据、时间序列数据、空间地理数据等

- 另一种广义向量角度的区分：
  - **原子向量**：各个值都是同类型（`logical`、`integer`、`double`、`character`、`complex`、`raw`）[1]
  - **列表**：各个值可以不同类型
- 广义向量添加额外属性的扩展类型：
  - 基于整数型向量构建的因子
  - 基于数值型向量构建的日期和日期时间
  - 基于数值型向量构建的时间序列
  - 基于列表构建的数据框和 tibble

[1] `integer` 和 `double` 统称为 `numeric`.

一. 向量（一维数据）

- 向量是由一组相同类型的原子值构成的序列，可以是一组数值、一组逻辑值、一组字符串等。
- 常用向量有：数值向量、逻辑向量、字符向量。

1. 数值向量

- 数值向量就是由数值组成的向量，单个数值是长度为 1 的数值向量

```
x = 1.5
x
#> [1] 1.5
numeric(10)    # 长度为 10 的全 0 向量
#>  [1] 0 0 0 0 0 0 0 0 0 0
```

- 函数 c() 实现将多个对象合并到一起

```
c(1, 2, 3, 4, 5)
#> [1] 1 2 3 4 5
c(1, 2, c(3, 4, 5))   # 将多个数值向量合并成一个数值向量
#> [1] 1 2 3 4 5
```

- 创建等差的数值向量

```
1:5                          # 同 seq(5) 或 seq(1,5)
#> [1] 1 2 3 4 5
seq(1, 10, 2)                # 从 1 开始, 到 10 结束, 步长为 2
#> [1] 1 3 5 7 9
seq(3, length.out = 10)
#>  [1]  3  4  5  6  7  8  9 10 11 12
```

附 seq() 基本语法:

```
seq(from, to, by, length.out, along.with, ...)
```

- 创建重复的数值向量

```
x = 1:3
rep(x, 2)
#> [1] 1 2 3 1 2 3
rep(x, each = 2)
#> [1] 1 1 2 2 3 3
rep(x, c(2, 1, 2))                    # 按照规则重复序列中的各元素
#> [1] 1 1 2 3 3
rep(x, each = 2, length.out = 4)
#> [1] 1 1 2 2
rep(x, times = 3, each = 2)
#>  [1] 1 1 2 2 3 3 1 1 2 2 3 3 1 1 2 2 3 3
```

附 rep() 的基本语法:

```
rep(x, times, length.out, each, ...)
```

- 向量可以做 `+`、`-`、`*`、`/`四则运算，即对应元素分别做运算的向量化运算。注意，R 中两个不同长度的向量做运算，短的会自动循环补齐以配合长的：

```
2:3 + 1:5
#> [1] 3 5 5 7 7
```

2. 逻辑向量

- 逻辑向量，是由逻辑值（TRUE 或 FALSE, 或简写为 T 或 F）组成的向量。
- 向量做逻辑运算，得到的结果是逻辑向量：

```
c(1, 2) > c(2, 1)           # 等价于 c(1 > 2, 2 > 1)
#> [1] FALSE  TRUE
c(2, 3) > c(1, 2, -1, 3)  # 等价于 c(2 > 1, 3 > 2, 2 > -1, 3
#> [1]  TRUE  TRUE  TRUE FALSE
```

- 元素属于运算符%in%：判断元素是否属于集合

```
c(1, 4) %in% c(1, 2, 3)   # 左边向量每一个元素是否属于右边集合
#> [1]  TRUE FALSE
```

3. 字符向量

- 字符（串）向量，是一组字符串组成的向量，单引号和双引号都可以用来生成字符串。

```
"hello, world!"
#> [1] "hello, world!"
c("Hello", "World")
#> [1] "Hello" "World"
c("Hello", "World") == "Hello, World"
#> [1] FALSE FALSE
```

- 若字符串本身包含单引号或双引号，可以错开使用，或者用转义符\做转义，

```
'Is "You" a Chinese name?'
#> [1] "Is \"You\" a Chinese name?"
```

- writeLines()函数输出纯字符串内容：

```
writeLines("Is \"You\" a Chinese name?")
#> Is "You" a Chinese name?
```

4. 访问向量子集

即访问向量的一些特定元素或者某个子集。注意，R 中的索引是从 1 开始的。

- 使用元素的位置来访问：

```
v1 = c(1, 2, 3, 4)
v1[2]             # 第 2 个元素
v1[2:4]           # 第 2-4 个元素
v1[-3]            # 除了第 3 个之外的元素
v1[3:6]           # 可以访问不存在的位置，返回`NA`
```

- 位置向量作为索引，注意不能既放正数又放负数：

```
v1[c(1,3)]
v1[c(1, 2, -3)]      # 报错
```

- 使用逻辑向量来访问，输入与向量相同长度的逻辑向量，以此决定每一个元素是否要被获取：

```
v1[c(TRUE, FALSE, TRUE, FALSE)]
```

- 这可以引申为“根据条件访问向量子集”：

```
v1[v1 <= 2]   # 同 v1[which(v1 <= 2)] 或 subset(v1, v1<=2)
v1[v1 ^ 2 - v1 >= 2]
which.max(v1)     # 返回向量 v1 中最大值所在的位置
which.min(v1)     # 返回向量 v1 中最小值所在的位置
```

5. 对向量子集赋值，替换相应元素

对向量子集赋值，就是先访问到向量子集，再赋值。

```
v1[2] = 0
v1[2:4] = c(0, 1, 3)
v1[c(TRUE, FALSE, TRUE, FALSE)] = c(3, 2)
v1[v1 <= 2] = 0
v1[10] = 8      # 若对不存在的位置赋值，前面将用`NA`补齐
v1
```

6. 对向量元素命名

- 可以在创建向量的同时对其每个元素命名

```
x = c(a = 1, b = 2, c = 3)
x
#> a b c
#> 1 2 3
```

- 命名后，就可以通过名字来访问向量元素：

```
x[c("a", "c")]
x[c("a", "a", "c")]     # 重复访问也是可以的
x["d"]                  # 访问不存在的名字
```

- 访问或修改向量名字

```r
names(x)                          # 获取向量元素的名字
#> [1] "a" "b" "c"
names(x) = c("x", "y", "z")       # 更改向量元素的名字
names(x) = NULL                   # 移除向量元素的名字
```

重要: 区分 [ ] 与 [[ ]]

- [ ] 是提取对象的子集，[[ ]] 是提取对象内的元素。

二者的区别：将一个向量比作 10 盒糖果，你可以使用 [ ] 获取其中的 3 盒糖果，使用 [[ ]] 打开盒子并从中取出一颗糖果。

- 对于未命名向量，[ ] 和 [[ ]] 结果相同，而对命名向量：

```
x = c(a = 1, b = 2, c = 3)
x["a"]            # 取出标签为"a" 的糖果盒
#> a
#> 1
x[["a"]]          # 取出标签为"a" 的糖果盒里的糖果
#> [1] 1
```

- 注意，[[ ]] 只能放一个、不能放：多个索引、负索引、不存在索引

7. 对向量排序

- `sort()`: 对向量排序，默认 `decreasing = FALSE` 表示升序
- `order()`: 返回元素排好序的索引，以其结果作为索引访问元素，正好是排好序的向量
- `rank()`: 返回该向量中各个元素的 “排名”，参数 `method` 设置相同值的处理方法
- `rev()`: 将向量进行反转

```r
x = c(1,5,8,2,9,7,4)
sort(x)
#> [1] 1 2 4 5 7 8 9
order(x)      # 默认升序，排名第 2 的元素在原向量的第 4 个位置
#> [1] 1 4 7 2 6 3 5
x[order(x)]  # 同 sort(x)
#> [1] 1 2 4 5 7 8 9
rank(x)       # 默认升序，第 2 个元素排名第 4 位
#> [1] 1 4 6 2 7 5 3
```

二. 矩阵（二维数据）

- **矩阵**是一个用两个维度表示和访问的向量。故适用于向量的方法大多也适用于矩阵。
- 矩阵也要求元素是同一类型，数值矩阵、逻辑矩阵等。

1. **创建矩阵**
   - `matrix()` 将一个向量创建为矩阵：

```
matrix(x, nrow, ncol, byrow, dimnames, ...)
```

```
matrix(c(1, 2, 3,
         4, 5, 6,
         7, 8, 9), nrow = 3, byrow = FALSE)
#>      [,1] [,2] [,3]
#> [1,]    1    4    7
#> [2,]    2    5    8
#> [3,]    3    6    9
matrix(c(1, 2, 3,
         4, 5, 6,
         7, 8, 9), nrow = 3, byrow = TRUE)
#>      [,1] [,2] [,3]
#> [1,]    1    2    3
#> [2,]    4    5    6
#> [3,]    7    8    9
```

- 对矩阵的行列命名：

```
matrix(1:9, nrow = 3, byrow = TRUE,
       dimnames = list(c("r1","r2","r3"), c("c1","c2","c3")))
#>    c1 c2 c3
#> r1  1  2  3
#> r2  4  5  6
#> r3  7  8  9
```

- 也可以创建后再命名：

```
m1 = matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), ncol = 3)
rownames(m1) = c("r1", "r2", "r3")
colnames(m1) = c("c1", "c2", "c3")
```

- 特殊矩阵：

```
diag(1:4, nrow = 4)       # 对角矩阵
#>      [,1] [,2] [,3] [,4]
#> [1,]    1    0    0    0
#> [2,]    0    2    0    0
#> [3,]    0    0    3    0
#> [4,]    0    0    0    4
```

注：函数 `as.vector()`，可将矩阵转化为向量，元素按列读取。

2. 访问矩阵子集

- 矩阵是用两个维度表示和访问的向量，可以用一个二维存取器 [,] 来访问。
- 方括号中的第 1 个参数是行索引，第 2 个参数是列索引。与构建向量子集一样，可以在两个维度中使用数值向量、逻辑向量和字符向量。

```
m1[1,2]                  # 提取第 1 行，第 2 列的单个元素
m1[1:2, 2:4]             # 提取第 1 至 2 行，第 2 至 4 列的元素
# 提取行名为 r1 和 r3，列名为 c1 和 c3 的元素
m1[c("r1","r3"), c("c1","c3")]
```

- 若一个维度空缺，则选出该维度的所有元素，负索引仍是删除：

```
m1[1,]                  # 提取第 1 行，所有列元素
m1[,2:4]                # 提取所有行，第 2 至 4 列的元素
m1[-1,]                 # 提取除了第 1 行之外的所有元素
m1[,-c(2,4)]            # 提取除了第 2 和 4 列之外的所有元素
```

注意，矩阵本质上仍然是一个向量，故也可以使用一维索引：

```
m1[3:7]
#> [1] 3 4 5 6 7
```

- 若输入一个不等式，则返回同样大小的逻辑矩阵：

```
m1 > 3
#>       c1    c2    c3
#> r1 FALSE  TRUE  TRUE
#> r2 FALSE  TRUE  TRUE
#> r3 FALSE  TRUE  TRUE
```

- 根据它就可以选择矩阵元素或赋值：

```
m1[m1 > 3]    # 注意选出来的结果是向量
#> [1] 4 5 6 7 8 9
```

- **矩阵运算**
    - `A+B`，`A-B`，`A*B`，`A/B`：矩阵四则运算，要求矩阵同型，类似 Matlab 中的点运算，分别将对应位置的元素做四则运行；
    - `A %*% B`：矩阵乘法，要求 A 的列数 = B 的行数。

三. 多维数组（多维数据）

- 向量/矩阵向更高维度的自然推广，就是多维数组，也要求元素是同一类型。

1. **创建多维数组**
   - `array()` 将一个向量创建为多维数组：

```
array(x, dim, dimnames, ...)
```

```r
a1 = array(1:24, dim = c(3, 4, 2))
a1
#> , , 1
#>
#>      [,1] [,2] [,3] [,4]
#> [1,]    1    4    7   10
#> [2,]    2    5    8   11
#> [3,]    3    6    9   12
#>
#> , , 2
#>
#>      [,1] [,2] [,3] [,4]
#> [1,]   13   16   19   22
#> [2,]   14   17   20   23
#> [3,]   15   18   21   24
```

- 在创建数组时对每个维度进行命名：

```
a1 = array(1:24, dim = c(3, 4, 2),
           dimnames = list(c("r1","r2","r3"),
                           c("c1","c2","c3","c4"), c("k1","k2")))
```

- 创建之后再命名

```
a1 = array(1:24, dim = c(3, 4, 2))
dimnames(a1) = list(c("r1","r2","r3"),
                    c("c1","c2","c3","c4"), c("k1","k2"))
```

2. 访问多维数组子集

- 第 3 个维度姑且称为 “页”

```
a1[2,4,2]              # 提取第 2 行，第 4 列，第 2 页的元素
a1["r2","c4","k2"]     # 提取第 r2 行，第 c4 列，第 k2 页的元素
a1[1,2:4,1:2]          # 提取第 1 行，第 2 至 4 列，第 1 至 2 页
a1[,,2]                # 提取第 2 页的所有元素
dim(a1)                # 返回多维数组 a1 的各维度的维数
```

注： 在想象多维数组时，为了便于形象地理解，可以将其维度依次想象为与 “书” 相关的概念：行、列、页、本、层、架、室……

本篇主要参阅 (张敬信, 2022), (Hadley Wickham, 2017), (任坤, 2017), 模板感谢 (黄湘云, 2021), (谢益辉, 2021).

参考文献

Hadley Wickham, G. G. (2017). *R for Data Science*. O' Reilly, 1 edition. ISBN 978-1491910399.

任坤 (2017). *R* 语言编程指南. 人民邮电出版社, 北京. ISBN 978-7115462640.

张敬信 (2022). *R* 语言编程：基于 *tidyverse*. 人民邮电出版社, 北京.

谢益辉 (2021). *rmarkdown: Dynamic Documents for R*.

黄湘云 (2021). *Github: R-Markdown-Template*.