R 语言编程：基于 tidyverse

第 04 讲　数据结构 II: 列表, 数据框

张敬信

2022 年 2 月 10 日

哈尔滨商业大学

四. 列表（list）

- 列表，可以包含不同类型的对象，甚至可以包括其他列表。列表的灵活性使得它非常有用。
- 列表最大的好处就是，它能够将多个不同类型的对象打包到一起，使得可以根据位置和名字访问它们。
- 常用场景：将函数的多个返回值打包到一起作为一个对象返回。

1. 创建列表

- 用 `list()` 将多个不同类型的对象创建为列表：

```
l0 = list(1, c(TRUE, FALSE), c("a", "b", "c"))
l0
#> [[1]]
#> [1] 1
#>
#> [[2]]
#> [1]  TRUE FALSE
#>
#> [[3]]
#> [1] "a" "b" "c"
```

- 在创建列表时，为列表的每个成分指定名字：

```
l1 = list(A = 1, B = c(TRUE, FALSE), C = c("a", "b", "c"))
l1
#> $A
#> [1] 1
#>
#> $B
#> [1]  TRUE FALSE
#>
#> $C
#> [1] "a" "b" "c"
```

- 创建列表后再对列表成分命名或修改名字：

```
names(l1) = NULL                    # 移除列表成分的名字
names(l1) = c("x","y","z")
```

再来区分：[[ ]] 与 [ ]

- [[ ]] 始终是提取一个元素的内容，列表某一个成分的内容（下一级元素）
- [ ] 始终是提取子集，列表的子集是包含若干成分的子列表（仍是同类型对象）

2. 从列表中提取成分的内容

- 最常用的方法是用 $，通过成分名字来提取该成分下的内容：

```
l1$y
l1$m                    # 访问不存在的成分 m，将会返回 NULL
```

- 用 [[n]] 来提取列表第 n 个成分的内容，n 也可以换成成分的名字：

```
l1[[2]]                 # 同 l1[["y"]]
```

注：用 [[ ]] 提取列表中某个成分的内容更加灵活，可用在函数调用中，通过参数来传递成分索引或名字：

```
p = "y"                 # 想要提取其内容的成分名字
l1[[p]]
```

3. 提取列表子集

- 用 [ ]，可以取出列表中的一些成分，作为一个新的（子）列表。
- [ ] 中可以用字符向量表示成分名字，用数值向量表示成分位置，或用逻辑向量指定是否选择，来取出列表成分。

```
l1["x"]           # 同 l1[1]
l1[c("x", "z")]   # 同 l1[c(1, 3)], l1[c(TRUE, FALSE, TRUE)]
```

4. 对列表的成分赋值

- 即先访问（提取）到列表的成分，再赋以相适应的值[1]

```
l1$x = 0    # 将列表的成分 x 赋值为 0
```

- 同时给多个列表成分赋值：

```
l1[c("x", "y")] = list(x = "new value for y", y = c(3, 1))
```

- 移除列表中的某些成分，只需赋值为 NULL：

```
l1[c("z", "m")] = NULL
```

[1]若给一个不存在的成分赋值，列表会自动地增加一个新成分.

5. 列表函数

- as.list() 将向量转换成列表:

```
l2 = as.list(c(a = 1, b = 2))
l2
#> $a
#> [1] 1
#>
#> $b
#> [1] 2
```

- `unlist()` 将一个列表打破成分界线，强制转换成一个向量[2]：

```
unlist(l2)
#> a b
#> 1 2
```

[2]若列表的成分具有不同类型，则自动向下兼容到统一类型.

tidyverse 系列中的 purrr 包为方便操作列表，提供了一系列操作列表的函数：

- pluck(): 同 [[提取列表中的元素
- keep(): 保留满足条件的元素
- discard(): 删除满足条件的元素
- compact(): 删除列表中的空元素
- append(): 在列表末尾增加元素
- flatten(): 摊平列表（只摊平一层）

五. 数据框（数据表）

- R 语言中做统计分析的样本数据，都是按数据框类型操作的。
- 数据框是指有若干行和列的数据集，它与矩阵类似，但并不要求所有列都是相同的类型；
- 本质上讲，数据框就是一个列表，它的每个成分都是一个向量，并且长度相同，以表格的形式展现。总之，**数据框是由列向量组成、有着矩阵形式的列表。**

- 数据框与最常见的数据表是一致的：每一列代表一个变量属性，每一行代表一条样本数据：

表 1: 数据表示例

Name	Gender	Age	Major
Ken	Male	24	Finance
Ashley	Female	25	Statistics
Jennifer	Female	23	Computer Science

tibble 与 data.frame

- R 自带的数据框是 data.frame，建议改用更现代的数据框：tibble[3]，tidyverse 包都是基于 tibble 数据框。
- tibble 对比 data.frame 的优势：
  - tibble() 比 data.frame() 做的更少：不改变输入变量的类型（R 4.0.0 **之前默认将字符串转化为因子！**），不会改变变量名，不会创建行名；
  - tibble 对象的列名可以是 R 中的“非法名”：非字母开头、包含空格，但定义和使用变量时都需要用倒引号 ‘括起来；
  - tibble 在输出时不自动显示所有行，避免大数据框时显示很多内容；
  - 用 [] 选取列子集时，即使只选取一列，返回结果仍是 tibble，而不自动简化为向量。

[3]读者若习惯用 data.frame，只需要换个名字，将 tibble 改为 data.frame 即可.

1. 创建数据框

- 用 tibble() 根据若干列向量创建 tibble:

```
library(tidyverse)          # 或 library(tibble)
persons = tibble(
  Name = c("Ken", "Ashley", "Jennifer"),
  Gender = c("Male", "Female", "Female"),
  Age = c(24, 25, 23),
  Major = c("Finance", "Statistics", "Computer Science"))
persons
#> # A tibble: 3 x 4
#>   Name     Gender   Age Major
#>   <chr>    <chr>  <dbl> <chr>
#> 1 Ken      Male      24 Finance
#> 2 Ashley   Female    25 Statistics
#> 3 Jennifer Female    23 Computer Science
```

- 用 tribble() 按行录入数据式创建 tibble:

```
tribble(
  ~Name, ~Gender, ~Age, ~Major,
  "Ken", "Male", 24, "Finance",
  "Ashley", "Female", 25, "Statistics",
  "Jennifer", "Female", 23, "Computer Science")
```

- 用 as_tibble() 将 data.frame, matrix, 各成分等长度的 list, 转换为 tibble。

- 对数据框的各列重命名：

```
df = tibble(id = 1:4,
            level = c(0, 2, 1, -1),
            score = c(0.5, 0.2, 0.1, 0.5))
names(df) = c("id", "x", "y")
df
#> # A tibble: 4 x 3
#>       id     x     y
#>    <int> <dbl> <dbl>
#> 1     1     0   0.5
#> 2     2     2   0.2
#> 3     3     1   0.1
#> 4     4    -1   0.5
```

2. 提取数据框的元素、子集

数据框是由列向量组成、有着矩阵形式的列表，所以可以用两种操作方式来访问数据框的元素和子集。

(1) 以列表方式提取数据框的元素、子集

- 用 $ 按列名来提取某一列的值，或者用 [[]] 按照位置或列名提取

```
df$x                    # 同 df[["x"]], df[[2]]
#> [1]  0  2  1 -1
```

- 用 [ ] 提取数据框的一列或多列，得到子数据框，其内可以是数值向量（列位置）、字符向量（列名）、逻辑向量（是否选择各列）。

```
df[1]    # 提取第 1 列, 同 df["id"]
#> # A tibble: 4 x 1
#>       id
#>    <int>
#> 1     1
#> 2     2
#> 3     3
#> 4     4
```

```
df[1:2]  # 同 df[c("id","x")], df[c(TRUE,TRUE,FALSE)]
#> # A tibble: 4 x 2
#>       id     x
#>    <int> <dbl>
#> 1     1     0
#> 2     2     2
#> 3     3     1
#> 4     4    -1
```

(2) 以矩阵方式提取数据框的元素、子集

- 以列表形式操作并不支持行选择。以矩阵形式操作更加灵活，同时支持列选择和行选择。即用 `[i, j]` 指定行或列来提取数据框子集，`[ , ]` 其内可以是数值向量、字符向量或者逻辑向量。

- 若行选择器为空，则只选择列（所有行）：

```
df[, "x"]
#> # A tibble: 4 x 1
#>       x
#>   <dbl>
#> 1     0
#> 2     2
#> 3     1
#> 4    -1
```

```
df[, c("x","y")]   # 同 df[,2:3]
#> # A tibble: 4 x 2
#>       x     y
#>   <dbl> <dbl>
#> 1     0   0.5
#> 2     2   0.2
#> 3     1   0.1
#> 4    -1   0.5
```

- 若列选择器为空，则只选择行（所有列）：

```
df[c(1,3),]
#> # A tibble: 2 x 3
#>       id     x     y
#>    <int> <dbl> <dbl>
#> 1     1     0   0.5
#> 2     3     1   0.1
```

- 同时选择行和列：

```
df[1:3, c("id","y")]
#> # A tibble: 3 x 2
#>       id     y
#>    <int> <dbl>
#> 1     1   0.5
#> 2     2   0.2
#> 3     3   0.1
```

- 根据条件筛选行或列：

```
df[df$y >= 0.5, c("id","y")]
#> # A tibble: 2 x 2
#>      id     y
#>   <int> <dbl>
#> 1     1   0.5
#> 2     4   0.5
ind = names(df) %in% c("x","y","w")
df[1:2, ind]
#> # A tibble: 2 x 2
#>      x     y
#>   <dbl> <dbl>
#> 1     0   0.5
#> 2     2   0.2
```

3. 给数据框赋值

给数据框赋值，就是选择要赋值的位置，再准备好同样大小且格式匹配的数据，赋值给那些位置即可，所以同样有列表方式和矩阵方式。

(1) 以列表方式给数据框赋值

- 用 $ 或 [[ ]] 对数据框的 1 列赋值

```
df$y = c(0.6,0.3,0.2,0.4)
# 同 df[["y"]] = c(0.6,0.3,0.2,0.4)
```

- 利用现有列，创建（计算）新列：

```
df$z = df$x + df$y
df
#> # A tibble: 4 x 4
#>      id     x     y     z
#>   <int> <dbl> <dbl> <dbl>
#> 1     1     0   0.5   0.5
#> 2     2     2   0.2   2.2
#> 3     3     1   0.1   1.1
#> 4     4    -1   0.5  -0.5
```

```r
df$z = as.character(df$z)   # 转换列的类型
df
#> # A tibble: 4 x 4
#>      id     x     y z
#>   <int> <dbl> <dbl> <chr>
#> 1     1     0   0.5 0.5
#> 2     2     2   0.2 2.2
#> 3     3     1   0.1 1.1
#> 4     4    -1   0.5 -0.5
```

- 用 [] 可以对数据框的 1 列或多列进行赋值：

```
df["y"] = c(0.8,0.5,0.2,0.4)
df[c("x", "y")] = list(level = c(1,2,1,0),
                       score = c(0.1,0.2,0.3,0.4))
```

(2) 以矩阵方式给数据框赋值

以列表方式对数据框进行赋值时，也是只能访问列。若需要更加灵活地进行赋值操作，可以以矩阵方式进行。

```
df[1:3,"y"] = c(-1,0,1)
df[1:2,c("x","y")] = list(level = c(0,0),
                          score = c(0.9,1.0))
```

4. 一些有用函数

- str() 或 glimpse() 作用在 R 对象上，显示该对象的结构：

```
str(persons)
#> tibble [3 x 4] (S3: tbl_df/tbl/data.frame)
#>  $ Name  : chr [1:3] "Ken" "Ashley" "Jennifer"
#>  $ Gender: chr [1:3] "Male" "Female" "Female"
#>  $ Age   : num [1:3] 24 25 23
#>  $ Major : chr [1:3] "Finance" "Statistics" "Computer Scie
```

- summary() 作用在数据框/列表上，将生成各列/成分的汇总信息：

```
summary(persons)
#>      Name               Gender                Age             M
#>  Length:3           Length:3           Min.   :23.0   Leng
#>  Class :character   Class :character   1st Qu.:23.5   Clas
#>  Mode  :character   Mode  :character   Median :24.0   Mode
#>                                        Mean   :24.0
#>                                        3rd Qu.:24.5
#>                                        Max.   :25.0
```

- 经常需要将多个数据框（或矩阵）按行或按列进行合并：
  - rbind()，增加行（样本数据），要求宽度（列数）相同；
  - cbind()，增加列（属性变量），要求高度（行数）相同。
- 向 persons 数据框中添加一个人的新记录：

```
rbind(persons,
      tibble(Name = "John", Gender = "Male",
             Age = 25, Major = "Statistics"))
#> # A tibble: 4 x 4
#>   Name     Gender   Age Major
#>   <chr>    <chr>  <dbl> <chr>
#> 1 Ken      Male      24 Finance
#> 2 Ashley   Female    25 Statistics
#> 3 Jennifer Female    23 Computer Science
#> 4 John     Male      25 Statistics
```

- 向 persons 数据框中添加两个新列表示每个人是否已注册和其手头的项目数量:

```
cbind(persons, Registered = c(TRUE, TRUE, FALSE),
                 Projects = c(3, 2, 3))
#>       Name Gender Age              Major Registered Projects
#> 1      Ken   Male  24            Finance       TRUE        3
#> 2   Ashley Female  25         Statistics       TRUE        2
#> 3 Jennifer Female  23 Computer Science      FALSE        3
```

注: 更建议用 dplyr 包提供的 bind_rows(), bind_cols(). rbind() 和 cbind() .

- expand.grid( ) 可生成多个属性水平值所有组合（笛卡儿积）的数据框：

```
expand.grid(type = c("A","B"), class = c("M","L","XL"))
#>   type class
#> 1    A     M
#> 2    B     M
#> 3    A     L
#> 4    B     L
#> 5    A    XL
#> 6    B    XL
```

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*.