

R 语言编程：基于 tidyverse

第 11 讲 数据操作 I: 选择列修改列

张敬信

2022 年 3 月 25 日

哈尔滨商业大学

用 `dplyr` 包实现各种数据操作，通常的数据操作无论多么复杂，往往都可以分解为 5 种基本数据操作的组合：

- `select()` —— 选择列
- `filter()/slice()` —— 筛选行
- `arrange()` —— 对行排序
- `mutate()` —— 修改列/创建新列
- `summarize()` —— 汇总

它们都可以与

- `group_by()` —— 分组

连用，以改变数据操作的作用域：

作用在整个数据框，还是分别作用在数据框的每个分组

这些函数组合使用就足以完成各种数据操作，它们的相同之处是：

- 第 1 个参数是数据框，方便管道操作
- 根据列名访问数据框的列，且列名不用加引号
- 返回结果是一个新数据框，不改变原数据框

从而，可以方便地实现：

将多个简单操作，依次用管道连接，实现复杂的数据操作

- 若要同时对所选择的多列应用函数，还有强大的 `across()` 函数，它支持各种**选择列语法**，搭配 `mutate()` 和 `summarise()` 使用，产生非常强大同时修改/汇总多列的效果；
- 类似地，`if_any()`，`if_all()` 函数，搭配 `filter()` 使用，产生强大的根据多列的值筛选行的效果。

一. 选择列

- 选择列，包括对数据框做**选择列**、**调整列序**、**重命名列**。
- 以虚拟的学生成绩数据来演示，包含随机生成的 20 个 NA：

```
library(readxl)
df = read_xlsx("datas/ExamDatas_NAs.xlsx")
df
#> # A tibble: 50 x 8
#>   class name    sex  chinese  math english moral science
#>   <chr> <chr>  <chr>    <dbl> <dbl>   <dbl> <dbl>   <dbl>
#> 1 六 1 班 何娜    女         87    92     79      9    10
#> 2 六 1 班 黄才菊  女         95    77     75     NA      9
#> 3 六 1 班 陈芳妹  女         79    87     66      9    10
#> # ... with 47 more rows
```

1. 选择列语法

(1) 用列名或索引选择列

```
df %>%  
  select(name, sex, math)  # 或者 select(2, 3, 5)  
#> # A tibble: 50 x 3  
#>   name    sex    math  
#>   <chr>  <chr> <dbl>  
#> 1 何娜    女      92  
#> 2 黄才菊  女      77  
#> 3 陈芳妹  女      87  
#> # ... with 47 more rows
```

(2) 借助运算符选择列

- 用: 选择连续的若干列
- 用! 选择变量集合的余集（反选）
- & 和 | 选择变量集合的交或并
- c() 合并多个选择

(3) 借助选择助手函数

- **选择指定列：**
 - everything(): 选择所有列
 - last_col(): 选择最后一列，可以带参数，如 last_col(5) 选择倒数第 6 列

- **选择列名匹配的列:**

- `starts_with()`: 以某前缀开头的列名
- `ends_with()`: 以某后缀结尾的列名
- `contains()`: 包含某字符串的列名
- `matches()`: 匹配正则表达式的列名
- `num_range()`: 匹配数值范围的列名, 如 `num_range("x", 1:3)` 匹配 `x1`, `x2`, `x3`

- **结合函数选择列:**

- `where()`: 应用一个函数到所有列, 选择返回结果为 `TRUE` 的列, 比如与 `is.numeric` 等函数连用

2. 一些选择列的示例

```
df %>%  
  select(starts_with("m"))  
#> # A tibble: 50 x 2  
#>   math moral  
#>   <dbl> <dbl>  
#> 1     92     9  
#> 2     77    NA  
#> 3     87     9  
#> # ... with 47 more rows
```

```
df %>%  
  select(ends_with("e"))  
#> # A tibble: 50 x 3  
#>   name      chinese science  
#>   <chr>      <dbl>    <dbl>  
#> 1 何娜          87        10  
#> 2 黄才菊          95         9  
#> 3 陈芳妹          79        10  
#> # ... with 47 more rows
```

```
df %>%  
  select(contains("a"))  
#> # A tibble: 50 x 4  
#>   class name    math moral  
#>   <chr> <chr>  <dbl> <dbl>  
#> 1 六 1 班 何娜      92      9  
#> 2 六 1 班 黄才菊      77     NA  
#> 3 六 1 班 陈芳妹      87      9  
#> # ... with 47 more rows
```

- 根据正则表达式匹配选择列:

```
df %>%  
  select(matches("m.*a"))  
#> # A tibble: 50 x 2  
#>   math moral  
#>   <dbl> <dbl>  
#> 1     92     9  
#> 2     77    NA  
#> 3     87     9  
#> # ... with 47 more rows
```

- 根据条件（逻辑判断）选择列，例如选择所有数值型的列：

```
df %>%  
  select(where(is.numeric))  
#> # A tibble: 50 x 5  
#>   chinese  math english moral science  
#>   <dbl> <dbl>   <dbl> <dbl>   <dbl>  
#> 1      87    92      79     9      10  
#> 2      95    77      75    NA       9  
#> 3      79    87      66     9      10  
#> # ... with 47 more rows
```

- 自定义返回 TRUE 或 FALSE 的判断函数, 支持 purrr 风格公式

```
df[, 4:8] %>% # 选择列和 > 3000 的列
  select(where(~ sum(.x, na.rm = TRUE) > 3000))
#> # A tibble: 50 x 2
#>   chinese math
#>   <dbl> <dbl>
#> 1     87    92
#> 2     95    77
#> 3     79    87
#> # ... with 47 more rows
```

```
df %>%      # 选择唯一值数目 < 10 的列
  select(where(~ n_distinct(.x) < 10))

#> # A tibble: 50 x 4
#>   class sex    moral science
#>   <chr> <chr> <dbl>    <dbl>
#> 1 六 1 班 女      9      10
#> 2 六 1 班 女     NA      9
#> 3 六 1 班 女      9      10
#> # ... with 47 more rows
```

3. 用“-“删除列

```
df %>%  
  select(-c(name, chinese, science))  
#> # A tibble: 50 x 5  
#>   class sex    math english moral  
#>   <chr> <chr> <dbl>   <dbl> <dbl>  
#> 1 六 1 班 女      92      79      9  
#> 2 六 1 班 女      77      75     NA  
#> 3 六 1 班 女      87      66      9  
#> # ... with 47 more rows  
# 或者 select(-ends_with("e"))
```



```
df %>%  
  select(math, everything(), -ends_with("e"))  
#> # A tibble: 50 x 5  
#>   math class sex   english moral  
#>   <dbl> <chr> <chr>   <dbl> <dbl>  
#> 1    92 六 1 班 女           79      9  
#> 2    77 六 1 班 女           75     NA  
#> 3    87 六 1 班 女           66      9  
#> # ... with 47 more rows
```

注意： `-ends_with()` 要放在 `everything()` 后面，否则删除的列就全回来了。

4. 调整列的顺序

- 列是根据被选择的顺序排列:

```
df %>%  
  select(ends_with("e"), math, name, class, sex)  
#> # A tibble: 50 x 6  
#>   name    chinese science  math class sex  
#>   <chr>    <dbl>    <dbl> <dbl> <chr> <chr>  
#> 1 何娜          87         10     92 六 1 班 女  
#> 2 黄才菊          95          9     77 六 1 班 女  
#> 3 陈芳妹          79         10     87 六 1 班 女  
#> # ... with 47 more rows
```

- `everything()` 返回未被选择的所有列，将某一列移到第一列时很方便：

```
df %>%  
  select(math, everything())  
#> # A tibble: 50 x 8  
#>   math class name    sex  chinese english moral science  
#>   <dbl> <chr> <chr>  <chr>    <dbl>    <dbl> <dbl>    <dbl>  
#> 1    92 六 1 班 何娜    女      87      79      9      10  
#> 2    77 六 1 班 黄才菊 女      95      75     NA      9  
#> 3    87 六 1 班 陈芳妹 女      79      66      9     10  
#> # ... with 47 more rows
```

- `relocate(.data, ..., .before, .after)`: 将选择的列移到某列之前或之后

```
df %>%      # 将数值列移到 name 列的后面
  relocate(where(is.numeric), .after = name)
#> # A tibble: 50 x 8
#>   class name    chinese  math english moral science sex
#>   <chr> <chr>      <dbl> <dbl>    <dbl> <dbl>    <dbl> <chr>
#> 1 六 1 班 何娜          87    92      79      9      10 女
#> 2 六 1 班 黄才菊        95    77      75     NA      9 女
#> 3 六 1 班 陈芳妹        79    87      66      9     10 女
#> # ... with 47 more rows
```

5. 重命名列

- `set_names()`: 为所有列设置新列名

```
df %>%
```

```
  set_names(" 班级", " 姓名", " 性别", " 语文",  
            " 数学", " 英语", " 品德", " 科学")
```

```
#> # A tibble: 50 x 8
```

```
#>   班级  姓名  性别  语文  数学  英语  品德  科学
```

```
#>   <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl>
```

```
#> 1 六 1 班 何娜 女      87     92     79      9     10
```

```
#> 2 六 1 班 黄才菊 女      95     77     75     NA      9
```

```
#> 3 六 1 班 陈芳妹 女      79     87     66      9     10
```

```
#> # ... with 47 more rows
```

- `rename()`: 只修改部分列名, 格式为: 新名 = 旧名

```
df %>%
```

```
  rename(数学 = math, 科学 = science)
```

```
#> # A tibble: 50 x 8
```

```
#>   class name    sex  chinese  数学 english moral  科学
```

```
#>   <chr> <chr>  <chr>    <dbl> <dbl>    <dbl> <dbl> <dbl>
```

```
#> 1 六 1 班 何娜    女          87     92          79      9     10
```

```
#> 2 六 1 班 黄才菊 女          95     77          75     NA      9
```

```
#> 3 六 1 班 陈芳妹 女          79     87          66      9     10
```

```
#> # ... with 47 more rows
```

- `rename_with(.data, .fn, .cols)`: 用函数`.fn` 变化选中列的列名

```
df %>%      # 为所选列名增加前缀
```

```
  rename_with(~ paste0("new_", .x), matches("m"))
```

```
#> # A tibble: 50 x 8
```

```
#>   class new_name sex   chinese new_math english new_moral
```

```
#>   <chr> <chr>     <chr>    <dbl>    <dbl>    <dbl>    <dbl>
```

```
#> 1 六 1 班 何娜      女          87          92          79
```

```
#> 2 六 1 班 黄才菊    女          95          77          75
```

```
#> 3 六 1 班 陈芳妹    女          79          87          66
```

```
#> # ... with 47 more rows
```

6. 强大的 across() 函数

- across() 人如其名，让零个/一个/多个函数**穿过**所选择的列，即**同时对所选择的多列应用若干函数**，基本格式为：

```
across(.cols = everything(), .fns = NULL, ..., .names)
```

- .cols 根据**选择列语法**选定的列；
- .fns 为应用到选定列上的函数¹：
 - NULL：不对列作变换；
 - 一个函数，如 mean；
 - 一个 purrr 风格的匿名函数，如 `~ .X * 10`
 - 多个函数或匿名函数构成的列表
- .names 设置输出列的列名样式，默认为 `{col}_{fn}`

¹这些函数内部可以使用 `cur_column()` 和 `cur_group()` 以访问当前列和分组键值。

- `across()` 支持各种选择列语法, 与 `mutate()` 和 `summarise()` 连用, 产生非常强大的同时修改/ (多种) 汇总多列效果;
- `across()` 也能与 `group_by()`, `count()` 和 `distinct()` 连用, 此时 `.fns` 为 `NULL`, 只起选择列的作用。
- `across()` 函数的引入, 使得可以弃用那些限定列范围的后缀:
 - `across(everything(), .fns)`: 在所有列范围内, 代替后缀 `_all`
 - `across(where(), .fns)`: 在满足条件的列范围内, 代替后缀 `_if`
 - `across(.cols, .fns)`: 在给定的列范围内, 代替后缀 `_at`

注: f 将长度为 n 的向量, 映射为长度为 n 的向量

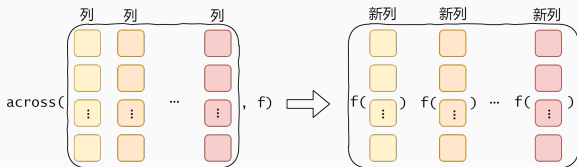


图 1: `across` 函数示意图

- `across()` 包含了**分解思维**: 想要同时修改多列, 只需要选出多列, 把对一列做的事情写成函数, 剩下的交给 `across()` 就行了。

二. 修改列

修改列，即修改数据框的列，计算新列。

1. 创建新列

- `mutate()` 创建或修改列²，返回原数据框并增加新列，默认加在最后一列，参数 `.before`，`.after` 可以设置新列的位置。

```
df %>% # 只给新列 1 个值，循环使用
  mutate(new_col = 5, .before = chinese)
#> # A tibble: 50 x 9
#>   class name    sex  new_col chinese  math english moral s
#>   <chr> <chr>  <chr>    <dbl>    <dbl> <dbl>    <dbl> <dbl>
#> 1 六 1 班 何娜    女          5      87     92      79     9
#> 2 六 1 班 黄才菊 女          5      95     77      75    NA
#> 3 六 1 班 陈芳妹 女          5      79     87      66     9
#> # ... with 47 more rows
```

²`transmute()` 只返回增加的新列。

- 正常是以长度等于行数的向量赋值:

```
df %>%  
  mutate(new_col = 1:n())  
#> # A tibble: 50 x 9  
#>   class name    sex  chinese  math english moral science r  
#>   <chr> <chr>  <chr>    <dbl> <dbl>   <dbl> <dbl>   <dbl>  
#> 1 六 1 班 何娜    女          87    92      79      9      10  
#> 2 六 1 班 黄才菊 女          95    77      75     NA      9  
#> 3 六 1 班 陈芳妹 女          79    87      66      9     10  
#> # ... with 47 more rows
```

注: `n()` 返回当前分组的样本数, 未分组则为总行数。

2. 计算新列

- 用数据框的现有列计算新列，若修改当前列，只需要赋值给原列名

```
df %>%
```

```
  mutate(total = chinese + math + english + moral + science)
```

```
#> # A tibble: 50 x 9
```

```
#>   class name    sex  chinese  math english moral science t
```

```
#>   <chr> <chr>  <chr>    <dbl> <dbl>   <dbl> <dbl>   <dbl> <
```

```
#> 1 六 1 班 何娜    女          87    92        79     9    10
```

```
#> 2 六 1 班 黄才菊 女          95    77        75    NA     9
```

```
#> 3 六 1 班 陈芳妹 女          79    87        66     9    10
```

```
#> # ... with 47 more rows
```

注意：不能用 `sum()`，它会将整个列的内容都加起来，类似的还有 `mean()`。

- 在同一个 `mutate()` 中可以同时创建或计算多个列，它们是从前往后依次计算，所以可以使用前面新创建的列

```
df %>%
```

```
  mutate(med = median(math, na.rm = TRUE),  
         label = math > med,  
         label = as.numeric(label))
```

```
#> # A tibble: 50 x 10
```

```
#>   class name    sex  chinese  math english moral science
```

```
#>   <chr> <chr>  <chr>    <dbl> <dbl>   <dbl> <dbl>   <dbl> <
```

```
#> 1 六 1 班 何娜    女          87    92      79      9      10
```

```
#> 2 六 1 班 黄才菊 女          95    77      75     NA      9
```

```
#> 3 六 1 班 陈芳妹 女          79    87      66      9     10
```

```
#> # ... with 47 more rows
```

3. 修改多列

结合 `across()` 和**选择列语法**可以应用函数到多列，从而实现同时修改多列。

(1) 应用函数到所有列

- 将所有列转化为字符型

```
df %>%  
  mutate(across(everything(), as.character))  
#> # A tibble: 50 x 8  
#>   class name    sex  chinese math  english moral science  
#>   <chr> <chr>  <chr> <chr>   <chr> <chr>   <chr> <chr>  
#> 1 六 1 班 何娜    女    87      92      79        9      10  
#> 2 六 1 班 黄才菊 女    95      77      75       <NA>    9  
#> 3 六 1 班 陈芳妹 女    79      87      66        9      10  
#> # ... with 47 more rows
```

(2) 应用函数到满足条件的列

- 对所有数值列做归一化

```
df %>%
```

```
  mutate(across(where(is.numeric), Rescale))
```

```
#> # A tibble: 50 x 8
```

```
#>   class name    sex  chinese  math english  moral science  
#>   <chr> <chr>  <chr>    <dbl> <dbl>   <dbl>  <dbl>   <dbl>
```

```
#> 1 六 1 班 何娜    女      0.843 0.974    1      0.875    1
```

```
#> 2 六 1 班 黄才菊 女      1      0.776    0.926 NA      0.83
```

```
#> 3 六 1 班 陈芳妹 女      0.686 0.908    0.759 0.875    1
```

```
#> # ... with 47 more rows
```


(3) 应用函数到指定的列

- 将 iris 中的列名包含 length 和 width 的列的测量单位从厘米变成毫米:

```
as_tibble(iris) %>%  
  mutate(across(contains("Length") | contains("Width"),  
                ~ .x * 10))
```

```
#> # A tibble: 150 x 5
```

```
#>   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
```

```
#>           <dbl>         <dbl>         <dbl>         <dbl> <fct>
```

```
#> 1           51           35           14           2 setosa
```

```
#> 2           49           30           14           2 setosa
```

```
#> 3           47           32           13           2 setosa
```

```
#> # ... with 147 more rows
```

4. 替换 NA

(1) `replace_na()`

实现用某个值替换一系列中的所有 NA 值，该函数接受一个命名列表，其成分为列名 = 替换值。

- 替换具体的列的缺失值

```
starwars %>%  
  replace_na(list(hair_color = "UNKNOWN",  
                  height = round(mean(.$height, na.rm = TRUE))  
#> # A tibble: 87 x 14  
#>   name      height  mass hair_color skin_color eye_color b  
#>   <chr>      <int> <dbl> <chr>      <chr>      <chr>  
#> 1 Luke Sk~    172    77 blond      fair       blue  
#> 2 C-3PO      167    75 UNKNOWN   gold       yellow  
#> 3 R2-D2       96    32 UNKNOWN   white, blue red  
#> # ... with 84 more rows, and 5 more variables: homeworld <
```

- 所有浮点列的缺失值用其均值替换:

```
starwars %>%  
  mutate(across(where(is.double),  
                 ~ replace_na(.x, mean(.x, na.rm = TRUE))))  
#> # A tibble: 87 x 14  
#>   name      height  mass hair_color skin_color eye_color  
#>   <chr>      <int> <dbl> <chr>      <chr>      <chr>  
#> 1 Luke Sk~    172    77 blond      fair       blue  
#> 2 C-3PO      167    75 <NA>      gold       yellow  
#> 3 R2-D2       96    32 <NA>      white, blue red  
#> # ... with 84 more rows, and 5 more variables: homeworld <chr>,  
#> #   films <list>, vehicles <list>, starships <list>
```

(2) fill()

- 用前一个 (或后一个) 非缺失值填充 NA

```
load("datas/gap_data.rda")
```

```
knitr::kable(gap_data, align="c")
```

site	species	sample_num	bees_present
Bilpin	A. longiforlia	1	TRUE
NA	NA	2	TRUE
NA	NA	3	TRUE
NA	A. elongata	1	TRUE
NA	NA	2	FALSE
NA	NA	3	TRUE
Grose Vale	A. terminalis	1	FALSE
NA	NA	2	FALSE
NA	NA	2	TRUE

```
gap_data %>% # 默认用上一个值填充
  fill(site, species)
#> # A tibble: 9 x 4
#>   site species sample_num bees_present
#>   <chr>  <chr>         <dbl> <lgl>
#> 1 Bilpin A. longiforlia         1 TRUE
#> 2 Bilpin A. longiforlia         2 TRUE
#> 3 Bilpin A. longiforlia         3 TRUE
#> # ... with 6 more rows
```

5. 重新编码

(1) 两类别情形: `if_else()`: 做二分支判断进而重新编码

```
df %>%
```

```
  mutate(sex = if_else(sex == " 男", "M", "F"))
```

```
#> # A tibble: 50 x 8
```

```
#>   class name    sex  chinese  math english moral science
```

```
#>   <chr> <chr>  <chr>    <dbl> <dbl>   <dbl> <dbl>   <dbl>
```

```
#> 1 六 1 班 何娜    F          87    92      79      9      10
```

```
#> 2 六 1 班 黄才菊 F          95    77      75     NA      9
```

```
#> 3 六 1 班 陈芳妹 F          79    87      66      9     10
```

```
#> # ... with 47 more rows
```

(2) 多类别情形: `case_when()`: 做多分支判断进而重新编码, 避免使用很多 `if_else()` 嵌套

```
df %>%
```

```
  mutate(math = case_when(math >= 75 ~ "High",  
                           math >= 60 ~ "Middle",  
                           TRUE      ~ "Low"))
```

```
#> # A tibble: 50 x 8
```

```
#>   class name    sex  chinese math  english moral science
```

```
#>   <chr> <chr>   <chr>    <dbl> <chr>    <dbl> <dbl>    <dbl>
```

```
#> 1 六 1 班 何娜    女          87 High          79      9      10
```

```
#> 2 六 1 班 黄才菊 女          95 High          75     NA      9
```

```
#> 3 六 1 班 陈芳妹 女          79 High          66      9     10
```

```
#> # ... with 47 more rows
```

case_when() 中用的是公式形式,

- 左边是返回 TRUE 或 FALSE 的表达式或函数
- 右边是若左边表达式为 TRUE, 则重新编码的值, 也可以是表达式或函数
- 每个分支条件将从上到下的计算, 并接受第一个 TRUE 条件
- 最后一个分支直接用 TRUE 表示若其他条件都不为 TRUE 时怎么做

(3) 更强大的重新编码函数

sjmisc 包提供了 `rec(x, rec, append, ...)`, 基本实现了 SPSS 重新编码的功能 - `x`: 为数据框 (或向量);

- `append`: 默认为 `TRUE`, 则返回包含重编码新列的数据框, `FALSE` 则只返回重编码的新列;
- `rec`: 设置重编码模式:
 - 重编码对: 每个重编码对用 “;” 隔开, 例如 `rec="1=1; 2=4; 3=2; 4=3"`
 - 多值: 多个旧值 (逗号分隔) 重编码为一个新值, 例如 `rec="1,2=1; 3,4=2"`
 - 值范围: 用冒号表示值范围, 例如 `rec="1:4=1; 5:8=2"`
 - 数值型值范围: 带小数部分的数值向量, 值范围内的所有值将被重新编码, 例如 `rec="1:2.5=1; 2.6:3=2"`³

³注意 2.55 因未包含在值范围将不被重新编码.

- “min” 和 “max”：最小值和最大值分别用 min 和 max 表示，例如 `rec = "min:4=1; 5:max=2"` (min 和 max 也可以作为新值，如 `5:7=max`，表示将 5~7 编码为 `max(x)`)
- “else”：所有未设定的其他值，用 else 表示，例如 `rec="3=1; 1=2; else=3"`
- “copy”：else 可以结合 copy 一起使用，表示所有未设定的其他值保持原样 (从原数值 copy)，例如 `rec="3=1; 1=2; else=copy"`
- NAs：NA 既可以作为旧值，也可以作为新值，例如 `rec="NA=1; 3:5=NA"`
- “rev”：设置反转值顺序
- 非捕获值：不匹配的值将设置为 NA, 除非使用 else 和 copy.

```

library(sjmisc)
rec(df, math,
    rec = "min:59= 不及格; 60:74= 中; 75:85= 良; 85:max= 优",
    append = FALSE) %>%
    frq()                                # 频率表
#>
#> math_r <character>
#> # total N=50  valid N=50  mean=3.28  sd=1.26
#>
#> Value | N | Raw % | Valid % | Cum. %
#> -----
#> -Inf | 3 | 6.00 | 6.00 | 6
#> 不及格 | 14 | 28.00 | 28.00 | 34
#> 良 | 10 | 20.00 | 20.00 | 54
#> 优 | 12 | 24.00 | 24.00 | 78
#> 中 | 11 | 22.00 | 22.00 | 100
#> <NA> | 0 | 0.00 | <NA> | <NA>

```

本篇主要参阅 (张敬信, 2022), (Hadley Wickham, 2017), (Desi Quintans, 2019), 以及包文档, 模板感谢 (黄湘云, 2021), (谢益辉, 2021).

参考文献

Desi Quintans, J. P. (2019). *Working in the Tidyverse*. HIE Advanced R workshop.

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

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*.