

R 语言编程：基于 tidyverse

第 12 讲 数据操作 II: 筛选行分组汇总

张敬信

2022 年 2 月 12 日

哈尔滨商业大学

三. 筛选行

- 筛选行，即按行选择数据子集，包括**过滤行**、**对行切片**、**删除行**。

```
library(readxl)
df = read_xlsx("datas/ExamDatas_NAs.xlsx")
set.seed(123)
df_dup = df %>%      # 创建一个包含重复行的数据框
  slice_sample(n = 60, replace = TRUE)
```

1. 用 filter() 根据条件筛选行

- 筛选条件可以是长度同行数的逻辑向量，更一般的是基于能返回这样逻辑向量的列表表达式

```
df_dup %>%
```

```
  filter(sex == "男", math > 80)
```

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

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

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

```
#> 1 六 2 班 陈华健 男          92    84      70      9      10
```

```
#> 2 六 2 班 陈华健 男          92    84      70      9      10
```

```
#> 3 六 4 班 <NA>   男          84    85      52      9      8
```

```
#> # ... with 5 more rows
```

注：多个条件之间用“`&`”隔开，相当于 `and`。

```
df_dup %>%
  filter(sex == " 女", (is.na(english) | math > 80))
```

#> # A tibble: 11 x 8

#>	class	name	sex	chinese	math	english	moral	science
#>	<chr>	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
#> 1	六 4 班	周婵	女	92	94	77	10	9
#> 2	六 1 班	陈芳妹	女	79	87	66	9	10
#> 3	六 5 班	陆曼	女	88	84	69	8	10

#> # ... with 8 more rows

```
df_dup %>%
```

```
  filter(between(math, 70, 80))      # 闭区间
```

```
#> # A tibble: 15 x 8
```

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

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

```
#> 1 六 2 班 杨远芸 女          93    80      68    9      10
```

```
#> 2 六 5 班 容唐    女          83    71      56    9      7
```

```
#> 3 六 4 班 关小孟 男          84    78      49    8      5
```

```
#> # ... with 12 more rows
```

2. 在限定列范围内根据条件筛选行

- `if_any()` 和 `if_all()`, 其操作逻辑类似 `across()`, 只是返回的是关于行的逻辑向量 (长度同行数), 用于根据多列的值筛选行



图 1: `ifany, ifall` 函数筛选行示意图

- 在 `.cols` 所选择的列范围内, 分别对每一行做逻辑判断, 前者返回各行是否存在某列满足函数 `fns` 所表示的判断条件; 后者返回各行是否所有列都满足函数 `fns` 所表示的判断条件。

(1) 限定列范围内，筛选“所有值都满足某条件的行”

用 `if_all()`，其操作逻辑是，在所选列范围内根据条件做判断，得到多列逻辑值，再借助 `all`（所有）语义，合成一个逻辑值向量，用于 `filter()` 筛选行。

- 选出第 4-6 列范围内，所有值都 > 75 的行：

```
df %>%
```

```
  filter(if_all(4:6, ~ .x > 75))
```

```
#> # A tibble: 3 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 4 班 周婵    女          92    94      77     10      9
```

```
#> 3 六 5 班 符苡榕  女          85    89      76      9     NA
```

- 选出所有列范围内，所有值都不是 NA 的行

```
df_dup %>%
```

```
  filter(if_all(everything(), ~ !is.na(.x)))
```

```
#> # A tibble: 38 x 8
```

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

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

```
#> 1 六 4 班 周婵 女          92    94      77    10      9
```

```
#> 2 六 2 班 杨远芸 女          93    80      68     9     10
```

```
#> 3 六 2 班 陈华健 男          92    84      70     9     10
```

```
#> # ... with 35 more rows
```


(2) 限定列范围内，筛选“存在值满足某条件的行”

用 `if_any()`，其操作逻辑是，在所选列范围内，根据条件做判断，得到多列逻辑值，再借助 `any`（存在）语义，合成一个逻辑值向量，用于 `filter()` 筛选行。

- 选出所有列范围内，存在值包含“bl”的行

```
starwars %>%
```

```
  filter(if_any(everything(), ~ str_detect(.x, "bl")))
```

```
#> # A tibble: 47 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 R2-D2       96    32 <NA>      white, bl~  red
```

```
#> 3 Owen La~   178   120 brown, grey light      blue
```

```
#> # ... with 44 more rows, and 5 more variables: homeworld <
```

```
#> #   films <list>, vehicles <list>, starships <list>
```

- 选出数值列范围内, 存在值 > 90 的行

```
df %>%
```

```
  filter(if_any(where(is.numeric), ~ .x > 90))
```

```
#> # A tibble: 8 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 2 班 黄祖娜 女          94    88        75    10        10
```

```
#> # ... with 5 more rows
```

- 从字符列范围内，选择包含（存在）NA 的行

```
df_dup %>%
```

```
  filter(if_any(where(is.character), is.na))
```

```
#> # A tibble: 3 x 8
```

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

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

```
#> 1 六 4 班 <NA>   男           84     85         52     9         8
```

```
#> 2 <NA> 徐达政 男           90     86         72     9        10
```

```
#> 3 六 5 班 符芳盈 <NA>           58     85         48     9        10
```

3. 对行切片: slice_*()

slice 就是对行切片的意思, 该系列函数的共同参数:

- n: 用来指定要选择的行数
- prop: 用来指定选择的行比例

```
slice(df, 3:7)                # 选择 3-7 行
slice_head(df, n, prop)       # 从前面开始选择若干行
slice_tail(df, n, prop)       # 从后面开始选择若干行
# 根据 order_by 选择最小的若干行
slice_min(df, order_by, n, prop)
# 根据 order_by 选择最大的若干行
slice_max(df, order_by, n, prop)
slice_sample(df, n, prop)      # 随机选择若干行
```

- 选择 math 列值中前 5 大的行

```
df %>%
```

```
  slice_max(math, n = 5)
```

```
#> # A tibble: 5 x 8
```

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

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

```
#> 1 六 4 班 周婵    女          92    94        77    10          9
```

```
#> 2 六 4 班 陈丽丽 女          87    93        NA     8          6
```

```
#> 3 六 1 班 何娜    女          87    92        79     9         10
```

```
#> # ... with 2 more rows
```

4. 删除行

(1) `distinct()`: 删除重复行

- 根据所有列判定重复, 只保留第 1 个, 删除其余

```
df_dup %>%
```

```
  distinct()
```

```
#> # A tibble: 35 x 8
```

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

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

```
#> 1 六 4 班 周婵    女          92    94      77    10      9
```

```
#> 2 六 2 班 杨远芸 女          93    80      68     9     10
```

```
#> 3 六 2 班 陈华健 男          92    84      70     9     10
```

```
#> # ... with 32 more rows
```

- 也可以只根据某些列判定重复

```
df_dup %>%      # 只根据 sex 和 math 判定重复
```

```
  distinct(sex, math, .keep_all = TRUE)
```

```
#> # A tibble: 32 x 8
```

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

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

```
#> 1 六 4 班 周婵    女          92    94        77    10        9
```

```
#> 2 六 2 班 杨远芸 女          93    80        68     9       10
```

```
#> 3 六 2 班 陈华健 男          92    84        70     9       10
```

```
#> # ... with 29 more rows
```

注：默认只返回选择的列，要返回所有列，需要设置参数`.keep_all = TRUE`。

(2) drop_na(): 删除包含 NA 的行

- 删除所有包含 NA 的行

```
df_dup %>%
```

```
  drop_na()
```

```
#> # A tibble: 38 x 8
```

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

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

```
#> 1 六 4 班 周婵    女          92    94      77    10      9
```

```
#> 2 六 2 班 杨远芸  女          93    80      68     9     10
```

```
#> 3 六 2 班 陈华健  男          92    84      70     9     10
```

```
#> # ... with 35 more rows
```


- 也可以只删除某些列包含 NA 的行:

```
df_dup %>%
```

```
  drop_na(sex:math)
```

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

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

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

```
#> 1 六 4 班 周婵    女          92    94        77    10        9
```

```
#> 2 六 2 班 杨远芸 女          93    80        68     9       10
```

```
#> 3 六 2 班 陈华健 男          92    84        70     9       10
```

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

- 若要删除某些列都是 NA 的行，借助 `if_all()` 也很容易实现：

```
df_dup %>%
```

```
  filter(!if_all(where(is.numeric), is.na))
```

```
#> # A tibble: 60 x 8
```

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

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

```
#> 1 六 4 班 周婵    女          92    94      77    10      9
```

```
#> 2 六 2 班 杨远芸 女          93    80      68     9     10
```

```
#> 3 六 2 班 陈华健 男          92    84      70     9     10
```

```
#> # ... with 57 more rows
```

四. 对行排序

- `arrange()`: 对行排序, 默认是递增

```
df_dup %>%
```

```
  arrange(math, sex)
```

```
#> # A tibble: 60 x 8
```

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

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

```
#> 1 六 3 班 邹嘉伟 男           67    18        62     8        NA
```

```
#> 2 六 3 班 刘虹均 男           72    23        74     3         6
```

```
#> 3 六 3 班 刘虹均 男           72    23        74     3         6
```

```
#> # ... with 57 more rows
```

- 若要递减排序，套一个 desc() 或变量名前加-

```
df_dup %>%
```

```
  arrange(-math)    # 同 desc(math), 递减排序
```

```
#> # A tibble: 60 x 8
```

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

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

```
#> 1 六 4 班 周婵 女          92    94      77    10      9
```

```
#> 2 六 4 班 陈丽丽 女          87    93      NA     8      6
```

```
#> 3 六 5 班 符苡榕 女          85    89      76     9     NA
```

```
#> # ... with 57 more rows
```

五. 分组汇总 (透视表)

对未分组的数据框，一些操作如 `mutate()` 是在所有行上执行——或者说，整个数据框是一个分组，所有行都属于它。

若数据框被分组，则这些操作是分别在每个分组上独立执行。可以认为是，将数据框拆分为更小的多个数据框。在每个更小的数据框上执行操作，最后再将结果合并回来。

1. 创建分组

- 用 `group_by()` 创建分组，只是对数据框增加了分组信息，并不是真的将数据分割为多个数据框

```
df_grp = df %>%  
  group_by(sex)  
df_grp  
#> # A tibble: 50 x 8  
#> # Groups:   sex [3]  
#>   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
```

- 访问或查看分组情况

<code>group_keys(df_grp)</code>	# 分组键值（唯一识别分组）
<code>group_indices(df_grp)</code>	# 查看每一行属于哪一分组
<code>group_rows(df_grp)</code>	# 查看每一组包含哪些行
<code>ungroup(df_grp)</code>	# 解除分组

其他分组函数

- `group_split()`: 真正将数据框分割为多个分组, 返回列表, 每个成分是一个分组数据框
- `group_nest()`: 将数据框分组 (`group_by`), 再做嵌套 (`nest`), 生成嵌套数据框

```
iris %>%  
  group_nest(Species)  
#> # A tibble: 3 x 2  
#>   Species          data  
#>   <fct>      <list<tibble[,4]>>  
#> 1 setosa      [50 x 4]  
#> 2 versicolor [50 x 4]  
#> 3 virginica  [50 x 4]
```


- `purrr` 风格的分组迭代：将函数 `.f` 依次应用到分组数据框 `.data` 的每个分组上
 - `group_map(.data, .f, ...)`: 返回列表
 - `group_walk(.data, .f, ...)`: 不返回，只关心副作用
 - `group_modify(.data, .f, ...)`: 返回修改后的分组数据框

```
iris %>%  
  group_by(Species) %>%  
  group_map(~ head(.x, 2))      # 提取每组的前两个观测
```

```

#> [[1]]
#> # A tibble: 2 x 4
#>   Sepal.Length Sepal.Width Petal.Length Petal.Width
#>   <dbl>         <dbl>         <dbl>         <dbl>
#> 1         5.1         3.5         1.4         0.2
#> 2         4.9         3         1.4         0.2
#>
#> [[2]]
#> # A tibble: 2 x 4
#>   Sepal.Length Sepal.Width Petal.Length Petal.Width
#>   <dbl>         <dbl>         <dbl>         <dbl>
#> 1          7         3.2         4.7         1.4
#> 2         6.4         3.2         4.5         1.5
#>
#> [[3]]
#> # A tibble: 2 x 4
#>   Sepal.Length Sepal.Width Petal.Length Petal.Width

```

2. 分组修改

- **分组**分组是一种强大的数据思维，当你想分组并分别对每组数据做操作时，应该优先采用 `group_by + mutate`，而不是分割数据 + 循环迭代。
- 这里仍是数据分解的思维：一旦对数据框分组，你只需要考虑对一个分组（子数据框）做的操作怎么实现，剩下的事情：分组 + 合并结果，`group_by + mutate` 会帮你完成。

例如，对如下的股票数据，分别计算每支股票的收盘价与前一天的差价。

- 例如，股票数据分别计算每支股票的收盘价与前一天的差价

```
load("datas/stocks.rda")
stocks
#> # A tibble: 753 x 3
#>   Date      Stock  Close
#>   <date>    <chr> <dbl>
#> 1 2017-01-03 Google  786.
#> 2 2017-01-03 Amazon  754.
#> 3 2017-01-03 Apple   116.
#> # ... with 750 more rows
```

- 只要对 Stock 分组，对一支股票怎么计算收盘价与前一天的差价，就怎么写代码

```
stocks %>%
  group_by(Stock) %>%
  mutate(delta = Close - lag(Close))
#> # A tibble: 753 x 4
#> # Groups:   Stock [3]
#>   Date      Stock  Close delta
#>   <date>    <chr>  <dbl> <dbl>
#> 1 2017-01-03 Google  786.    NA
#> 2 2017-01-03 Amazon  754.    NA
#> 3 2017-01-03 Apple   116.    NA
#> # ... with 750 more rows
```

3. 分组汇总

- `group_by + summarise`: 对数据框做分组, 更常与汇总连用。汇总就是以某种方式组合行。
 - `group_by + summarise`: 分组汇总, 结果是“**有几个分组就有几个样本**”
 - `group_by + mutate`: 分组修改, 结果是“**原来几个样本还是几个样本**”

(1) `summarise()`

- 与很多自带或自定义的汇总函数连用:
 - `n()`: 观测数
 - `n_distinct(var)`: 变量 `var` 的唯一值数目
 - `sum(var), max(var), min(var), ...`
 - `mean(var), median(var), sd(var), IQR(var), ...`

```

df %>%
  group_by(sex) %>%
  summarise(n = n(),
            math_avg = mean(math, na.rm = TRUE),
            math_med = median(math))
#> # A tibble: 3 x 4
#>   sex      n math_avg math_med
#>   <chr> <int>   <dbl>   <dbl>
#> 1 男      24    64.6      NA
#> 2 女      25    70.8      NA
#> 3 <NA>     1    85      85

```

- summarise(), 配合 across() 可以对所选择的列做一种或多种汇总

(2) 对某些列做汇总

```
df %>%  
  group_by(class, sex) %>%  
  summarise(across(contains("h"), mean, na.rm = TRUE))  
#> # A tibble: 12 x 5  
#> # Groups:   class [6]  
#>   class sex   chinese  math english  
#>   <chr> <chr>    <dbl> <dbl>   <dbl>  
#> 1 六 1 班 男      57    79.7    64.7  
#> 2 六 1 班 女     80.7   77.2    67.4  
#> 3 六 2 班 男     75.4   68.8    42.6  
#> # ... with 9 more rows
```


(3) 对所有列做汇总

```
df %>%  
  select(-name) %>%  
  group_by(class, sex) %>%  
  summarise(across(everything(), mean, na.rm = TRUE))  
#> # A tibble: 12 x 7  
#> # Groups:   class [6]  
#>   class sex   chinese  math english moral science  
#>   <chr> <chr>    <dbl> <dbl>    <dbl> <dbl>    <dbl>  
#> 1 六 1 班 男      57    79.7    64.7  8.67    9.33  
#> 2 六 1 班 女     80.7   77.2    67.4  8.33    9.57  
#> 3 六 2 班 男     75.4   68.8    42.6  8.8     9.25  
#> # ... with 9 more rows
```

(4) 对满足条件的列做多种汇总

```
df_grp = df %>%
  group_by(class) %>%
  summarise(across(where(is.numeric),
                    list(sum=sum, mean=mean, min=min),
                    na.rm = TRUE))

df_grp
#> # A tibble: 6 x 16
#>   class chinese_sum chinese_mean chinese_min math_sum math_min
#>   <chr>         <dbl>         <dbl>         <dbl>     <dbl>
#> 1 六 1 班           622           77.8           57       702
#> 2 六 2 班           746           82.9           66       570
#> 3 六 3 班           606           67.3           44       349
#> # ... with 3 more rows, and 9 more variables: english_sum <dbl>,
#> #   english_mean <dbl>, english_min <dbl>, moral_sum <dbl>,
#> #   moral_min <dbl>, science_sum <dbl>, science_mean <dbl>, science_min <dbl>
```

- 可读性不好，再来个宽变长：

```
df_grp %>%  
  pivot_longer(-class, names_to = c("Vars", ".value"),  
               names_sep = "_")  
#> # A tibble: 30 x 5  
#>   class Vars      sum mean  min  
#>   <chr> <chr>   <dbl> <dbl> <dbl>  
#> 1 六 1 班 chinese    622  77.8    57  
#> 2 六 1 班 math      702   78     55  
#> 3 六 1 班 english   666  66.6    54  
#> # ... with 27 more rows
```

(5) 支持多返回值的汇总函数

- summarise() 也支持多返回值（返回向量值、甚至是数据框）的汇总函数，如 range(), quantile() 等

```
qs = c(0.25, 0.5, 0.75)
df_q = df %>%
  group_by(sex) %>%
  summarise(math_qs = quantile(math, qs, na.rm = TRUE),
            q = qs)
```

```
df_q
```

```
#> # A tibble: 9 x 3
#> # Groups:   sex [3]
#>   sex    math_qs      q
#>   <chr>    <dbl> <dbl>
#> 1 男         57.5  0.25
#> 2 男         69    0.5
#> 3 男         80    0.75
```

- 可读性不好，再来个长变宽：

```
df_q %>%  
  pivot_wider(names_from = q, values_from = math_qs,  
              names_prefix = "q_")  
  
#> # A tibble: 3 x 4  
#> # Groups:   sex [3]  
#>   sex    q_0.25 q_0.5 q_0.75  
#>   <chr>   <dbl> <dbl> <dbl>  
#> 1 男      57.5    69    80  
#> 2 女      55     73   86.5  
#> 3 <NA>    85     85    85
```

3. 分组计数

- 用 `count()` 按分类变量 `class` 和 `sex` 分组, 并按分组大小排序

```
df %>%
```

```
  count(class, sex, sort = TRUE)
```

```
#> # A tibble: 12 x 3
```

```
#>   class sex      n
```

```
#>   <chr> <chr> <int>
```

```
#> 1 六 1 班 女      7
```

```
#> 2 六 4 班 男      6
```

```
#> 3 六 2 班 男      5
```

```
#> # ... with 9 more rows
```

- 对已分组的数据框，用 `tally()` 计数

```
df %>%  
  group_by(math_level = cut(math,  
                             breaks = c(0, 60, 75, 80, 100),  
                             right = FALSE)) %>%  
  tally()  
#> # A tibble: 5 x 2  
#>   math_level      n  
#>   <fct>        <int>  
#> 1 [0,60)         14  
#> 2 [60,75)        11  
#> 3 [75,80)         5  
#> # ... with 2 more rows
```

注：`count()` 和 `tally()` 都有参数 `wt` 设置加权计数。

- 用 `add_count()` 和 `add_tally()` 可为数据集增加一列按分组变量分组的计数

```
df %>%  
  add_count(class, sex)  
#> # A tibble: 50 x 9  
#>   class name    sex  chinese  math english moral science  
#>   <chr> <chr>  <chr>    <dbl> <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
```


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*.