

Tidyverse 学习笔记

Tidyverse

Qingyao Zhang

2025-11-10

目录

1	%>% versus >	1
1.1	> 的用法	2
1.2	%>% 的用法	4
2	c_across()	7

```
1 library(dplyr)
2 ##
3 ## Attaching package: 'dplyr'
4 ## The following objects are masked from 'package:stats':
5 ##
6 ##     filter, lag
7 ## The following objects are masked from 'package:base':
8 ##
9 ##     intersect, setdiff, setequal, union
10 library(tibble)
11 # 使用 Keng 中的 depress 数据进行演示
12 data("depress", package = "Keng")
```

1 %>% versus |>

对于 R 语言 4.1 之前的版本, tidyverse 依赖 magrittr 中的管道操作符 %>%. 对于 R 语言 4.1 及以后的版本, R base 中包含自己的管道运算符 |>。然而, %>% 与 |> 的功能不同。

1.1 |> 的用法

R base 中的 |> 的功能比较简单，运行?'|>'查看 |> 的文档。|> 的用法有以下几种：

1. 常规用法，将 |> 左侧对象传递给右侧函数的第一个参数。

```
1 # simple uses:
2 # same as pull(depress, gender)
3 depress |>
4   pull(gender)
5 ## [1] 0 1 1 0 1 0 1 0 1 1 0 0 1 0 1 1 1 0 1 0 1 0 0 1 1 1 1 0 1 1 0 1 0 0 1 0 1 0
6 ## [39] 1 0 0 1 0 1 0 1 1 1 1 1 1 0 0 0 1 1 1 0 1 1 1 1 0 1 1 0 0 0 0 1 0 1 1 1 1 1 0
7 ## [77] 1 0 1 0 0 1 0 1 0 1 0 1 0 0 1 1 1 0
8 # same as nrow(subset(depress, gender == 0))
9 depress |>
10   subset(gender == 0) |>
11   nrow()
12 ## [1] 41
```

2. 将 |> 左侧对象传递给右侧函数的其他参数。

- a. 使用 _ 占位符指代 |> 左侧对象

_ 占位符的使用规则

1. _ 默认内隐地传递给 |> 右侧函数的第一个参数；当 _ 以外显的方式传递时，_ 的默认传递将被取消
2. 只能使用参数名传递 _
3. 只能使用一个 _
4. 不能在嵌套函数中使用 _

```
1 # valid use
2 depress |>
3   subset(class == 3) |>
4   t.test(anx ~ gender, data = _)
5 ##
6 ## Welch Two Sample t-test
7 ##
8 ## data:  anx by gender
9 ## t = 1.3118, df = 19.977, p-value = 0.2045
10 ## alternative hypothesis: true difference in means between group 0 and group 1 is not equal
11 ## 95 percent confidence interval:
12 ## -0.2922643 1.2824604
```

```

13 ## sample estimates:
14 ## mean in group 0 mean in group 1
15 ##      2.083333      1.588235

```

下面的用法不可行:

```

1 depress |>
2   summary(select(.data = _, gender))

```

b. 使用匿名函数

```

1 depress |>
2   subset(class == 3) |>
3   (function(d) t.test(dm1 ~ gender, data = d))()
4 ##
5 ## Welch Two Sample t-test
6 ##
7 ## data: dm1 by gender
8 ## t = -0.89829, df = 26.676, p-value = 0.3771
9 ## alternative hypothesis: true difference in means between group 0 and group 1 is not equal
10 ## 95 percent confidence interval:
11 ## -0.4163176 0.1628862
12 ## sample estimates:
13 ## mean in group 0 mean in group 1
14 ##      1.729167      1.855882
15 depress |>
16   subset(class == 9) |>
17   (\(d) t.test(dm1 ~ gender, data = d))()
18 ##
19 ## Welch Two Sample t-test
20 ##
21 ## data: dm1 by gender
22 ## t = 0.89103, df = 12.351, p-value = 0.3899
23 ## alternative hypothesis: true difference in means between group 0 and group 1 is not equal
24 ## 95 percent confidence interval:
25 ## -0.2704022 0.6465927
26 ## sample estimates:
27 ## mean in group 0 mean in group 1
28 ##      2.171429      1.983333

```

c. |> 右侧函数使用参数名传递参数值, 从而跳过目标参数之前的参数

```

1 depress |>
2   subset(class == 3) |>
3   t.test(formula = dm1 ~ gender)
4 ##
5 ## One Sample t-test
6 ##
7 ## data: subset(depress, class == 3)
8 ## t = 5.5043, df = 6785, p-value = 3.841e-08
9 ## alternative hypothesis: true mean is not equal to 0
10 ## 95 percent confidence interval:
11 ## 84.56869 178.12434
12 ## sample estimates:
13 ## mean of x
14 ## 131.3465

```

3. |> 结合使用 _ 占位符提取数据子集

```

1 depress |>
2   _$gender
3 ## [1] 0 1 1 0 1 0 1 0 1 1 0 0 1 0 1 1 1 0 1 0 1 0 0 1 1 1 1 0 1 1 0 1 0 0 1 0 1 0
4 ## [39] 1 0 0 1 0 1 0 1 1 1 1 1 1 0 0 0 1 1 1 0 1 1 1 1 0 1 1 0 0 0 0 1 0 1 1 1 1 0
5 ## [77] 1 0 1 0 0 1 0 1 0 1 0 1 0 0 1 1 1 0
6 depress |>
7   _[["gender"]]
8 ## [1] 0 1 1 0 1 0 1 0 1 1 0 0 1 0 1 1 1 0 1 0 1 0 0 1 1 1 1 0 1 1 0 1 0 0 1 0 1 0
9 ## [39] 1 0 0 1 0 1 0 1 1 1 1 1 1 0 0 0 1 1 1 0 1 1 1 1 0 1 1 0 0 0 0 1 0 1 1 1 1 0
10 ## [77] 1 0 1 0 0 1 0 1 0 1 0 1 0 0 1 1 1 0

```

4. 使用 |> 转换代码

```

1 quote(depress |> subset(class == 3) |> nrow())
2 ## nrow(subset(depress, class == 3))

```

1.2 %>% 的用法

1. 常规用法，将 %>% 左侧对象传递给右侧函数的第一个参数。

```

1 # 同 pull(depress, gender)
2 depress %>%
3   pull(gender)
4 ## [1] 0 1 1 0 1 0 1 0 1 1 0 0 1 0 1 1 1 0 1 0 1 0 0 1 1 1 1 0 1 1 0 1 0 0 1 0 1 0

```

```

5 ## [39] 1 0 0 1 0 1 0 1 1 1 1 1 1 0 0 0 1 1 1 0 1 1 1 1 0 1 1 0 0 0 0 1 0 1 1 1 1 0
6 ## [77] 1 0 1 0 0 1 0 1 0 1 0 1 0 0 1 1 1 0
7 # 同 nrow(subset(depress, gender == 0))
8 depress %>%
9   subset(gender == 0) %>%
10   nrow()
11 ## [1] 41

```

2. 将%>% 左侧对象传递给右侧函数的其他参数。

a. 使用 . 占位符指代%>% 左侧对象

i . 占位符的使用规则

1. . 总是默认以内隐的方式传递给%>% 右侧函数的第一个参数，这一行为可以用 {} 取消
2. 可以使用位置传递。
3. %>% 右侧函数可以使用多个 .
4. 可以在嵌套函数中使用。

```

1 # 有效用法
2 depress %>%
3   mutate(.,
4     depr1_mean = rowMeans(select(., starts_with("depr1")))) %>%
5   pull(depr1_mean)
6 ## [1] 1.55 1.55 1.90 1.35 1.30 1.95 1.60 1.45 2.70 1.75 2.10 1.90 1.75 1.85 1.65
7 ## [16] 1.90 1.50 1.55 1.80 1.70 1.85 1.35 2.45 2.10 2.15 2.95 1.50 1.55 1.60 1.75
8 ## [31] 1.50 1.80 1.80 1.50 1.95 2.45 1.50 1.85 1.45 2.30 2.70 1.45 2.25 2.15 2.10
9 ## [46] 2.00 2.20 2.05 1.60 1.75 2.00 2.20 1.35 2.40 1.95 1.20 2.20 2.35 1.95 2.15
10 ## [61] 1.60 2.05 1.55 1.65 2.85 2.95 2.40 1.85 2.05 2.00 2.05 2.50 1.65 2.00 1.15
11 ## [76] 1.95 1.90 1.65 1.90 1.80 2.20 2.30 2.45 2.25 2.00 2.15 1.45 2.35 1.80 1.75
12 ## [91] 1.85 1.95 1.70 1.45
13 # `.` 总会传递给%>% 右侧函数第一个参数
14 depress %>%
15   slice_head(n = 5) %>%
16   pull(dm1) %>%
17   c(length(.))
18 ## [1] 1.55 1.55 1.90 1.35 1.30 5.00
19 # 在`{}`内, `.` 不会传递给第一个参数
20 depress %>%
21   slice_head(n = 5) %>%
22   pull(dm1) %>%
23   {c(length(.))}

```

```
24 ## [1] 5
```

b. 使用匿名函数

```
1 depress %>%
2   subset(class == 3) %>%
3   (function(d) t.test(dm1 ~ gender, data = d))
4 ##
5 ## Welch Two Sample t-test
6 ##
7 ## data: dm1 by gender
8 ## t = -0.89829, df = 26.676, p-value = 0.3771
9 ## alternative hypothesis: true difference in means between group 0 and group 1 is not equal
10 ## 95 percent confidence interval:
11 ## -0.4163176 0.1628862
12 ## sample estimates:
13 ## mean in group 0 mean in group 1
14 ## 1.729167 1.855882
15 depress %>%
16   subset(class == 5) %>%
17   t.test(dm1 ~ gender, data = .)
18 ##
19 ## Welch Two Sample t-test
20 ##
21 ## data: dm1 by gender
22 ## t = 1.4493, df = 17.276, p-value = 0.1652
23 ## alternative hypothesis: true difference in means between group 0 and group 1 is not equal
24 ## 95 percent confidence interval:
25 ## -0.09295787 0.50248168
26 ## sample estimates:
27 ## mean in group 0 mean in group 1
28 ## 2.033333 1.828571
29 # 当`%>%`右侧有多个语句时, 使用`{}`将这些语句括起来,
30 # 在`{}`中`.`不会传递给第一个参数
31 depress %>%
32   subset(class == 9) %>%
33   {
34     ncol(.)
35     nrow(.)
36   }
37 ## [1] 19
```

3. %>% 结合使用. 占位符提取数据子集

```

1 depress %>%
2   .$gender
3 ## [1] 0 1 1 0 1 0 1 0 1 1 0 0 1 0 1 1 1 0 1 0 1 0 0 1 1 1 1 0 1 1 0 1 0 0 1 0 1 0
4 ## [39] 1 0 0 1 0 1 0 1 1 1 1 1 1 0 0 0 1 1 1 0 1 1 1 1 0 1 1 0 0 0 0 1 0 1 1 1 1 0
5 ## [77] 1 0 1 0 0 1 0 1 0 1 0 1 0 0 1 1 1 0
6 depress %>%
7   .[["gender"]]
8 ## [1] 0 1 1 0 1 0 1 0 1 1 0 0 1 0 1 1 1 0 1 0 1 0 0 1 1 1 1 0 1 1 0 1 0 0 1 0 1 0
9 ## [39] 1 0 0 1 0 1 0 1 1 1 1 1 1 0 0 0 1 1 1 0 1 1 1 1 0 1 1 0 0 0 0 1 0 1 1 1 1 0
10 ## [77] 1 0 1 0 0 1 0 1 0 1 0 1 0 0 1 1 1 0

```

4. 使用%>% 与 . 创建函数

```

1 quote(depress %>% subset(class == 3) %>% nrow())
2 ## depress %>% subset(class == 3) %>% nrow()
3 # 当`.`出现在`%>%`左侧，它们将创建一个函数
4 subset_nrow <- . %>% subset(class == 3) %>% nrow()
5 subset_nrow(depress)
6 ## [1] 29

```

2 c_across()

dplyr 文档中介绍到，与 `c()` 相比，(1) `c_across()` 使用 tidy select，更加便捷；(2) `c_across()` 使用 `vecr::vec_c()`，给出的输出更加安全。

tidy select 只有 15 种（见 `help("select")`）：`:`，`!`，`&`，`c()`，`everything()`，`last_col()`，`group_cols()`，`starts_with()`，`ends_with()`，`contains()`，`matches()`，`num_range()`，`all_of()`，`any_of()`，`where()`。

下面的例子在 `data.frame` 中对 `c()` 与 `c_across()` 进行了比较，代码中注释了结果正确与否。

```

1 set.seed(20250927)
2 toy_dat <- data.frame(x1 = rnorm(10), x2 = rnorm(10), x3 = rnorm(10))
3 head(toy_dat)
4 ##           x1           x2           x3
5 ## 1 -2.7734224 -0.6661558  1.0882648
6 ## 2 -1.1618233 -0.9076756  0.1483481
7 ## 3 -0.3773254  0.1182323  1.0507643
8 ## 4  0.2743375 -0.1521034  0.7412800
9 ## 5  0.6070295  0.7295347 -0.5049492
10 ## 6 -0.3429219 -0.1650072  2.3721901

```

```

11 toy_dat |>
12   rowwise() |>
13   mutate(
14     # correct
15     y1 = mean(c(x1, x2, x3)),
16     # incorrect
17     y2 = mean(x1:x3),
18     # incorrect
19     y3 = mean(c(x1:x3)),
20     # correct
21     y4 = mean(c_across(x1:x3))
22   ) |>
23   head()
24 ## # A tibble: 6 x 7
25 ## # Rowwise:
26 ##       x1      x2      x3      y1      y2      y3      y4
27 ##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
28 ## 1 -2.77 -0.666 1.09 -0.784 -1.27 -1.27 -0.784
29 ## 2 -1.16 -0.908 0.148 -0.640 -0.662 -0.662 -0.640
30 ## 3 -0.377 0.118 1.05  0.264 0.123 0.123 0.264
31 ## 4  0.274 -0.152 0.741 0.288 0.274 0.274 0.288
32 ## 5  0.607 0.730 -0.505 0.277 0.107 0.107 0.277
33 ## 6 -0.343 -0.165 2.37  0.621 0.657 0.657 0.621

```

下面的例子在 tibble 中对 `c()` 与 `c_across()` 进行了比较，代码中注释了结果正确与否。

```

1 toy_dat <- as_tibble(toy_dat)
2 head(toy_dat)
3 ## # A tibble: 6 x 3
4 ##       x1      x2      x3
5 ##   <dbl> <dbl> <dbl>
6 ## 1 -2.77 -0.666 1.09
7 ## 2 -1.16 -0.908 0.148
8 ## 3 -0.377 0.118 1.05
9 ## 4  0.274 -0.152 0.741
10 ## 5  0.607 0.730 -0.505
11 ## 6 -0.343 -0.165 2.37
12 toy_dat |>
13   rowwise() |>
14   mutate(
15     # correct
16     y1 = mean(c(x1, x2, x3)),

```

```

17   # incorrect
18   y2 = mean(x1:x3),
19   # incorrect
20   y3 = mean(c(x1:x3)),
21   # correct
22   y4 = mean(c_across(x1:x3))
23 ) |>
24 head()
25 ## # A tibble: 6 x 7
26 ## # Rowwise:
27 ##       x1      x2      x3      y1      y2      y3      y4
28 ##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
29 ## 1 -2.77  -0.666  1.09  -0.784 -1.27  -1.27  -0.784
30 ## 2 -1.16  -0.908  0.148 -0.640 -0.662 -0.662 -0.640
31 ## 3 -0.377  0.118  1.05   0.264  0.123  0.123  0.264
32 ## 4  0.274  -0.152  0.741  0.288  0.274  0.274  0.288
33 ## 5  0.607  0.730 -0.505  0.277  0.107  0.107  0.277
34 ## 6 -0.343 -0.165  2.37   0.621  0.657  0.657  0.621

```

可见，在使用 `rowwise()` 时，必须将 tidy select 语法与 `c_across()` 结合使用，才能得出正确的计算结果。