Tidyverse 学习笔记

Tidyverse

Qingyao Zhang

2025-11-10

目录


```
library(dplyr)
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
library(tibble)
# 使用 Keng 中的 depress 数据进行演示
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。

```
# simple uses:
# same as pull(depress, gender)
depress |>
  pull(gender)
##  [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
## [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
## [77] 1 0 1 0 0 1 0 1 0 1 0 1 0 0 1 1 1 0
# same as nrow(subset(depress, gender == 0))
depress |>
  subset(gender == 0) |>
  nrow()
## [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 _

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

```
## sample estimates:
## mean in group 0 mean in group 1
##        2.083333        1.588235
```

下面的用法不可行:

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

b. 使用匿名函数

```
depress |>
  subset(class == 3) |>
  (function(d) t.test(dm1 ~ gender, data = d))()
##
##  Welch Two Sample t-test
##
## data:  dm1 by gender
## t = -0.89829, df = 26.676, p-value = 0.3771
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal
## 95 percent confidence interval:
##  -0.4163176  0.1628862
## sample estimates:
## mean in group 0 mean in group 1
##        1.729167        1.855882
depress |>
  subset(class == 9) |>
  (\(d) t.test(dm1 ~ gender, data = d))()
##
##  Welch Two Sample t-test
##
## data:  dm1 by gender
## t = 0.89103, df = 12.351, p-value = 0.3899
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal
## 95 percent confidence interval:
##  -0.2704022  0.6465927
## sample estimates:
## mean in group 0 mean in group 1
##        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

```r
depress |>
  subset(class == 3) |>
  t.test(formula = dm1 ~ gender)
##
##  One Sample t-test
##
## data:  subset(depress, class == 3)
## t = 5.5043, df = 6785, p-value = 3.841e-08
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##   84.56869 178.12434
## sample estimates:
## mean of x
##  131.3465
```

3. |> 结合使用 _ 占位符提取数据子集

```r
depress |>
  _$gender
##  [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
## [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
## [77] 1 0 1 0 0 1 0 1 0 1 0 1 0 0 1 1 1 0
depress |>
  _[["gender"]]
##  [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
## [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
## [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

```r
quote(depress |> subset(class == 3) |> nrow())
## 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。

```r
# 同 pull(depress, gender)
depress %>%
  pull(gender)
##  [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
```

```
## [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
## [77] 1 0 1 0 0 1 0 1 0 1 0 1 0 0 1 1 1 0
# 同 nrow(subset(depress, gender == 0))
depress %>%
  subset(gender == 0) %>%
  nrow()
## [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 .

```
# 有效用法
depress %>%
  mutate(.,
         depr1_mean = rowMeans(select(., starts_with("depr1")))) %>%
  pull(depr1_mean)
##  [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
## [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
## [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
## [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
## [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
## [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
## [91] 1.85 1.95 1.70 1.45
# `.`总会传递给%>% 右侧函数第一个参数
depress %>%
  slice_head(n = 5) %>%
  pull(dm1) %>%
  c(length(.))
## [1] 1.55 1.55 1.90 1.35 1.30 5.00
# 在`{}`内，`.`不会传递给第一个参数
depress %>%
  slice_head(n = 5) %>%
  pull(dm1) %>%
  {c(length(.))}
```

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

b. 使用匿名函数

```
depress %>%
  subset(class == 3) %>%
  (function(d) t.test(dm1 ~ gender, data = d))
##
##  Welch Two Sample t-test
##
## data:  dm1 by gender
## t = -0.89829, df = 26.676, p-value = 0.3771
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal
## 95 percent confidence interval:
##  -0.4163176  0.1628862
## sample estimates:
## mean in group 0 mean in group 1
##        1.729167        1.855882
depress %>%
  subset(class == 5) %>%
  t.test(dm1 ~ gender, data = .)
##
##  Welch Two Sample t-test
##
## data:  dm1 by gender
## t = 1.4493, df = 17.276, p-value = 0.1652
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal
## 95 percent confidence interval:
##  -0.09295787  0.50248168
## sample estimates:
## mean in group 0 mean in group 1
##        2.033333        1.828571
# 当`%>%`右侧有多个语句时，使用`{}`将这些语句括起来，
# 在`{}`中`.`不会传递给第一个参数
depress %>%
  subset(class == 9) %>%
  {
    ncol(.)
    nrow(.)
  }
## [1] 19
```

3. %>% 结合使用 . 占位符提取数据子集

```r
depress %>%
  .$gender
##  [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
## [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
## [77] 1 0 1 0 0 1 0 1 0 1 0 1 0 0 1 1 1 0
depress %>%
  .[["gender"]]
##  [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
## [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
## [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

```r
quote(depress %>% subset(class == 3) %>% nrow())
## depress %>% subset(class == 3) %>% nrow()
# 当`.`出现在`%>%`左侧，它们将创建一个函数
subset_nrow <- . %>% subset(class == 3) %>% nrow()
subset_nrow(depress)
## [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 vctrs::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。

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

```r
toy_dat |>
  rowwise() |>
  mutate(
    # correct
    y1 = mean(c(x1, x2, x3)),
    # incorrect
    y2 = mean(x1:x3),
    # incorrect
    y3 = mean(c(x1:x3)),
    # correct
    y4 = mean(c_across(x1:x3))
  ) |>
  head()
## # A tibble: 6 x 7
## # Rowwise:
##       x1     x2     x3     y1     y2     y3     y4
##    <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1 -2.77  -0.666  1.09  -0.784 -1.27  -1.27  -0.784
## 2 -1.16  -0.908  0.148 -0.640 -0.662 -0.662 -0.640
## 3 -0.377  0.118  1.05   0.264  0.123  0.123  0.264
## 4  0.274 -0.152  0.741  0.288  0.274  0.274  0.288
## 5  0.607  0.730 -0.505  0.277  0.107  0.107  0.277
## 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。

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

```
    # incorrect
    y2 = mean(x1:x3),
    # incorrect
    y3 = mean(c(x1:x3)),
    # correct
    y4 = mean(c_across(x1:x3))
  ) |>
  head()
## # A tibble: 6 x 7
## # Rowwise:
##       x1     x2     x3     y1     y2     y3     y4
##    <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1 -2.77  -0.666  1.09  -0.784 -1.27  -1.27  -0.784
## 2 -1.16  -0.908  0.148 -0.640 -0.662 -0.662 -0.640
## 3 -0.377  0.118  1.05   0.264  0.123  0.123  0.264
## 4  0.274 -0.152  0.741  0.288  0.274  0.274  0.288
## 5  0.607  0.730 -0.505  0.277  0.107  0.107  0.277
## 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。