L4: Tidyverse

Nan He, School of Basic Medical Sciences, Southern Medical University

2026-01-19

目录

Tidyverse 数据处理 **1**
1. 引言 1
2. Tidyverse 概述 1
3. dplyr 包：数据操作 3
4. tidyr 包：数据整理 22
5. dplyr 与 tidyr 结合使用 31
6. 实战练习 34
总结 36

Tidyverse 数据处理

1. 引言

在上节内容中，我们学习了 Base R 中的数据结构操作方法。虽然 Base R 功能强大，但在处理复杂数据时，代码往往不够简洁和直观。`tidyverse` 是一个专为数据科学设计的 R 包集合，它提供了一套统一、优雅的语法来进行数据处理和可视化。

本节目标：掌握 `tidyverse` 生态系统中两个最核心的包——`dplyr`（数据操作）和 `tidyr`（数据整理）的基本用法。

2. Tidyverse 概述

2.1 什么是 Tidyverse

`tidyverse` 是由 Hadley Wickham 主导开发的一系列 R 包的集合，包括：

- **dplyr**：数据操作（筛选、排序、汇总等）
- **tidyr**：数据整理（长宽格式转换等）
- **ggplot2**：数据可视化
- **readr**：数据读取

- **tibble**：增强型数据框
- **stringr**：字符串处理
- **purrr**：函数式编程
- **forcats**：因子处理

这些包共享统一的设计理念和语法风格，使得数据处理流程更加流畅。

2.2 安装与加载

```
# 安装整个 tidyverse
install.packages("tidyverse")

# 或单独安装
install.packages("dplyr")
install.packages("tidyr")
```

加载 tidyverse 会同时加载核心包：

```
{if (!requireNamespace("tidyverse", quietly = TRUE))
    install.packages("tidyverse")}
```

2.3 Tibble：增强型数据框

Tibble 是 tidyverse 中的数据框形式，相比传统 `data.frame` 有以下优势：

- 打印时更简洁，只显示前几行和适配屏幕的列
- 不自动将字符串转换为因子 (默认 `stringsAsFactors==FALSE`)
- 不会自动修改列名
- 子集操作更一致

```
# 创建 tibble
library(tidyverse)
tb <- tibble(
  sample_id = paste0("S", 1:6),
  group = c("Ctrl", "Ctrl", "Treat", "Treat", "Treat", "Ctrl"),
  age = c(45, 52, 60, 58, NA, 49),
  score = c(85.5, 92.3, 78.1, 88.6, 91.2, 76.8)
)

# 查看 tibble
tb
```

```
## # A tibble: 6 x 4
##   sample_id group   age score
```

```
##   <chr>     <chr> <dbl> <dbl>
## 1 S1        Ctrl     45  85.5
## 2 S2        Ctrl     52  92.3
## 3 S3        Treat    60  78.1
## 4 S4        Treat    58  88.6
## 5 S5        Treat    NA  91.2
## 6 S6        Ctrl     49  76.8
```

```
class(tb)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

注意表头下方用尖括号标注了数据类型：- `<chr>` 表示字符型 - `<dbl>` 表示双精度浮点数 - `<int>` 表示整数型

2.4 管道符 |>

管道符是 tidyverse 代码的核心特征，它将前一步的结果作为下一个函数的第一个参数，使代码更具可读性。

R 4.1.0 之后，Base R 提供了原生管道符 `|>`。tidyverse 早期使用的是 `magrittr` 包的 `%>%`，两者在大多数情况下可以互换。

```
# 传统嵌套写法（不简洁）
round(mean(c(1, 2, 3, 4, 5)), 2)
```

```
## [1] 3
```

```
# 管道符写法（清晰直观）
c(1, 2, 3, 4, 5) |>
  mean() |>
  round(2)
```

```
## [1] 3
```

3. dplyr 包：数据操作

`dplyr` 提供了一组”动词”函数来进行数据操作。这些函数有以下共同特点：

- 第一个参数是数据框
- 后续参数使用不带引号的变量名
- 输出结果是一个新数据框

3.1 准备示例数据

我们使用 `nycflights13` 包中的航班数据作为示例：

```r
# install.packages("nycflights13")
library(nycflights13)

flights
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      517            515         2      830            819
##  2  2013     1     1      533            529         4      850            830
##  3  2013     1     1      542            540         2      923            850
##  4  2013     1     1      544            545        -1     1004           1022
##  5  2013     1     1      554            600        -6      812            837
##  6  2013     1     1      554            558        -4      740            728
##  7  2013     1     1      555            600        -5      913            854
##  8  2013     1     1      557            600        -3      709            723
##  9  2013     1     1      557            600        -3      838            846
## 10  2013     1     1      558            600        -2      753            745
## # i 336,766 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

该数据集包含 2013 年从纽约市出发的所有 336,776 个航班记录。

3.2 行操作

3.2.1 filter() - 筛选行

`filter()` 根据条件筛选数据框中的行：

```r
# 筛选起飞延误超过 120 分钟的航班
flights |>
  filter(dep_delay > 120)
```

```
## # A tibble: 9,723 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      848           1835       853     1001           1950
##  2  2013     1     1      957            733       144     1056            853
##  3  2013     1     1     1114            900       134     1447           1222
##  4  2013     1     1     1540           1338       122     2020           1825
##  5  2013     1     1     1815           1325       290     2120           1542
##  6  2013     1     1     1842           1422       260     1958           1535
```

```
##  7  2013     1     1     1856           1645       131     2212           2005
##  8  2013     1     1     1934           1725       129     2126           1855
##  9  2013     1     1     1938           1703       155     2109           1823
## 10  2013     1     1     1942           1705       157     2124           1830
## # i 9,713 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

可以使用的比较运算符：- > (大于)、>= (大于等于) - < (小于)、<= (小于等于) - == (等于)、!= (不等于)

多条件筛选：

```
# 使用 & 表示" 与"
flights |>
  filter(month == 1 & day == 1)
```

```
## # A tibble: 842 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      517            515         2      830            819
##  2  2013     1     1      533            529         4      850            830
##  3  2013     1     1      542            540         2      923            850
##  4  2013     1     1      544            545        -1     1004           1022
##  5  2013     1     1      554            600        -6      812            837
##  6  2013     1     1      554            558        -4      740            728
##  7  2013     1     1      555            600        -5      913            854
##  8  2013     1     1      557            600        -3      709            723
##  9  2013     1     1      557            600        -3      838            846
## 10  2013     1     1      558            600        -2      753            745
## # i 832 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

```
# 使用 | 表示" 或"
flights |>
  filter(month == 1 | month == 2)
```

```
## # A tibble: 51,955 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      517            515         2      830            819
```

```
##  2  2013     1     1      533            529         4      850            830
##  3  2013     1     1      542            540         2      923            850
##  4  2013     1     1      544            545        -1     1004           1022
##  5  2013     1     1      554            600        -6      812            837
##  6  2013     1     1      554            558        -4      740            728
##  7  2013     1     1      555            600        -5      913            854
##  8  2013     1     1      557            600        -3      709            723
##  9  2013     1     1      557            600        -3      838            846
## 10  2013     1     1      558            600        -2      753            745
## # i 51,945 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

```
# 使用 %in% 简化多值匹配
flights |>
  filter(month %in% c(1, 2)) # 保留在 1 月或 2 月的航班
```

```
## # A tibble: 51,955 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      517            515         2      830            819
##  2  2013     1     1      533            529         4      850            830
##  3  2013     1     1      542            540         2      923            850
##  4  2013     1     1      544            545        -1     1004           1022
##  5  2013     1     1      554            600        -6      812            837
##  6  2013     1     1      554            558        -4      740            728
##  7  2013     1     1      555            600        -5      913            854
##  8  2013     1     1      557            600        -3      709            723
##  9  2013     1     1      557            600        -3      838            846
## 10  2013     1     1      558            600        -2      753            745
## # i 51,945 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

常见错误：- 用 = 判断相等，而非 == - 写成 `month == 1 | 2` 而非 `month == 1 | month == 2`

3.2.2 arrange() - 排序

arrange() 根据列值对行进行排序：

```
# 按年、月、日、起飞时间排序
flights |>
```

```
  arrange(year, month, day, dep_time) # 默认从小到大排列
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      517            515         2      830            819
##  2  2013     1     1      533            529         4      850            830
##  3  2013     1     1      542            540         2      923            850
##  4  2013     1     1      544            545        -1     1004           1022
##  5  2013     1     1      554            600        -6      812            837
##  6  2013     1     1      554            558        -4      740            728
##  7  2013     1     1      555            600        -5      913            854
##  8  2013     1     1      557            600        -3      709            723
##  9  2013     1     1      557            600        -3      838            846
## 10  2013     1     1      558            600        -2      753            745
## # i 336,766 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

```
# 使用 desc() 进行降序排列
flights |>
  arrange(desc(dep_delay))
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     9      641            900      1301     1242           1530
##  2  2013     6    15     1432           1935      1137     1607           2120
##  3  2013     1    10     1121           1635      1126     1239           1810
##  4  2013     9    20     1139           1845      1014     1457           2210
##  5  2013     7    22      845           1600      1005     1044           1815
##  6  2013     4    10     1100           1900       960     1342           2211
##  7  2013     3    17     2321            810       911      135           1020
##  8  2013     6    27      959           1900       899     1236           2226
##  9  2013     7    22     2257            759       898      121           1026
## 10  2013    12     5      756           1700       896     1058           2020
## # i 336,766 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
```

```
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

3.2.3 distinct() - 去重 distinct() 查找唯一行:

```
# 删除完全重复的行
flights |>
  distinct()
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      517            515         2      830            819
##  2  2013     1     1      533            529         4      850            830
##  3  2013     1     1      542            540         2      923            850
##  4  2013     1     1      544            545        -1     1004           1022
##  5  2013     1     1      554            600        -6      812            837
##  6  2013     1     1      554            558        -4      740            728
##  7  2013     1     1      555            600        -5      913            854
##  8  2013     1     1      557            600        -3      709            723
##  9  2013     1     1      557            600        -3      838            846
## 10  2013     1     1      558            600        -2      753            745
## # i 336,766 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

```
# 获取所有起点和终点的唯一组合
flights |>
  distinct(origin, dest)
```

```
## # A tibble: 224 x 2
##    origin dest
##    <chr>  <chr>
##  1 EWR    IAH
##  2 LGA    IAH
##  3 JFK    MIA
##  4 JFK    BQN
##  5 LGA    ATL
##  6 EWR    ORD
##  7 EWR    FLL
##  8 LGA    IAD
##  9 JFK    MCO
```

```
## 10 LGA    ORD
## # i 214 more rows
```

```
# 保留其他列信息
flights |>
  distinct(origin, dest, .keep_all = TRUE)
```

```
## # A tibble: 224 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      517            515         2      830            819
##  2  2013     1     1      533            529         4      850            830
##  3  2013     1     1      542            540         2      923            850
##  4  2013     1     1      544            545        -1     1004           1022
##  5  2013     1     1      554            600        -6      812            837
##  6  2013     1     1      554            558        -4      740            728
##  7  2013     1     1      555            600        -5      913            854
##  8  2013     1     1      557            600        -3      709            723
##  9  2013     1     1      557            600        -3      838            846
## 10  2013     1     1      558            600        -2      753            745
## # i 214 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

使用 count() 统计各组合出现次数:

```
flights |>
  count(origin, dest, sort = TRUE)
```

```
## # A tibble: 224 x 3
##    origin dest      n
##    <chr>  <chr> <int>
##  1 JFK    LAX   11262
##  2 LGA    ATL   10263
##  3 LGA    ORD    8857
##  4 JFK    SFO    8204
##  5 LGA    CLT    6168
##  6 EWR    ORD    6100
##  7 JFK    BOS    5898
##  8 LGA    MIA    5781
##  9 JFK    MCO    5464
```

```
## 10 EWR    BOS    5327
## # i 214 more rows
```

3.3 列操作

3.3.1 select() - 选择列

`select()` 用于选取需要的列：

```
# 指定列名
flights |>
  select(year, month, day)
```

```
## # A tibble: 336,776 x 3
##     year month   day
##    <int> <int> <int>
##  1  2013     1     1
##  2  2013     1     1
##  3  2013     1     1
##  4  2013     1     1
##  5  2013     1     1
##  6  2013     1     1
##  7  2013     1     1
##  8  2013     1     1
##  9  2013     1     1
## 10  2013     1     1
## # i 336,766 more rows
```

```
# 选择连续区间
flights |>
  select(year:day)
```

```
## # A tibble: 336,776 x 3
##     year month   day
##    <int> <int> <int>
##  1  2013     1     1
##  2  2013     1     1
##  3  2013     1     1
##  4  2013     1     1
##  5  2013     1     1
##  6  2013     1     1
##  7  2013     1     1
##  8  2013     1     1
##  9  2013     1     1
```

```
## 10  2013     1     1
## # i 336,766 more rows
```

```
# 排除某些列
flights |>
  select(!year:day)
```

```
## # A tibble: 336,776 x 16
##    dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier
##       <int>          <int>     <dbl>    <int>          <int>     <dbl> <chr>
##  1      517            515         2      830            819        11 UA
##  2      533            529         4      850            830        20 UA
##  3      542            540         2      923            850        33 AA
##  4      544            545        -1     1004           1022       -18 B6
##  5      554            600        -6      812            837       -25 DL
##  6      554            558        -4      740            728        12 UA
##  7      555            600        -5      913            854        19 B6
##  8      557            600        -3      709            723       -14 EV
##  9      557            600        -3      838            846        -8 B6
## 10      558            600        -2      753            745         8 AA
## # i 336,766 more rows
## # i 9 more variables: flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

```
# 选择字符型列
flights |>
  select(where(is.character))
```

```
## # A tibble: 336,776 x 4
##    carrier tailnum origin dest
##    <chr>   <chr>   <chr>  <chr>
##  1 UA      N14228  EWR    IAH
##  2 UA      N24211  LGA    IAH
##  3 AA      N619AA  JFK    MIA
##  4 B6      N804JB  JFK    BQN
##  5 DL      N668DN  LGA    ATL
##  6 UA      N39463  EWR    ORD
##  7 B6      N516JB  EWR    FLL
##  8 EV      N829AS  LGA    IAD
##  9 B6      N593JB  JFK    MCO
## 10 AA      N3ALAA  LGA    ORD
```

```
## # i 336,766 more rows
```

辅助函数进行模式匹配：

```
# 以 "dep" 开头的列
flights |>
  select(starts_with("dep"))
```

```
## # A tibble: 336,776 x 2
##    dep_time dep_delay
##       <int>     <dbl>
##  1      517         2
##  2      533         4
##  3      542         2
##  4      544        -1
##  5      554        -6
##  6      554        -4
##  7      555        -5
##  8      557        -3
##  9      557        -3
## 10      558        -2
## # i 336,766 more rows
```

```
# 以 "time" 结尾的列
flights |>
  select(ends_with("time"))
```

```
## # A tibble: 336,776 x 5
##    dep_time sched_dep_time arr_time sched_arr_time air_time
##       <int>          <int>    <int>          <int>    <dbl>
##  1      517            515      830            819      227
##  2      533            529      850            830      227
##  3      542            540      923            850      160
##  4      544            545     1004           1022      183
##  5      554            600      812            837      116
##  6      554            558      740            728      150
##  7      555            600      913            854      158
##  8      557            600      709            723       53
##  9      557            600      838            846      140
## 10      558            600      753            745      138
## # i 336,766 more rows
```

```r
# 包含 "arr" 的列
flights |>
  select(contains("arr"))
```

```
## # A tibble: 336,776 x 4
##    arr_time sched_arr_time arr_delay carrier
##       <int>          <int>     <dbl> <chr>
##  1      830            819        11 UA
##  2      850            830        20 UA
##  3      923            850        33 AA
##  4     1004           1022       -18 B6
##  5      812            837       -25 DL
##  6      740            728        12 UA
##  7      913            854        19 B6
##  8      709            723       -14 EV
##  9      838            846        -8 B6
## 10      753            745         8 AA
## # i 336,766 more rows
```

3.3.2 mutate() - 创建新列

mutate() 基于现有列创建新列：

```r
flights |>
  mutate(
    gain = dep_delay - arr_delay,        # 延误恢复时间
    speed = distance / air_time * 60     # 飞行速度（英里/小时）
  ) |>
  select(flight, gain, speed)
```

```
## # A tibble: 336,776 x 3
##    flight  gain speed
##     <int> <dbl> <dbl>
##  1   1545    -9  370.
##  2   1714   -16  374.
##  3   1141   -31  408.
##  4    725    17  517.
##  5    461    19  394.
##  6   1696   -16  288.
##  7    507   -24  404.
##  8   5708    11  259.
##  9     79     5  405.
## 10    301   -10  319.
```

```
## # i 336,766 more rows
```

控制新列位置：

```
# 将新列放在最前面
flights |>
  mutate(
    speed = distance / air_time * 60,
    .before = 1 # .before 控制新列位置
  ) |>
  select(1:5)
```

```
## # A tibble: 336,776 x 5
##    speed  year month   day dep_time
##    <dbl> <int> <int> <int>    <int>
##  1  370.  2013     1     1      517
##  2  374.  2013     1     1      533
##  3  408.  2013     1     1      542
##  4  517.  2013     1     1      544
##  5  394.  2013     1     1      554
##  6  288.  2013     1     1      554
##  7  404.  2013     1     1      555
##  8  259.  2013     1     1      557
##  9  405.  2013     1     1      557
## 10  319.  2013     1     1      558
## # i 336,766 more rows
```

```
# 只保留参与计算的列
flights |>
  mutate(
    gain = dep_delay - arr_delay,
    hours = air_time / 60,
    gain_per_hour = gain / hours,
    .keep = "used"
  )
```

```
## # A tibble: 336,776 x 6
##    dep_delay arr_delay air_time  gain hours gain_per_hour
##        <dbl>     <dbl>    <dbl> <dbl> <dbl>         <dbl>
##  1         2        11      227    -9 3.78          -2.38
##  2         4        20      227   -16 3.78          -4.23
##  3         2        33      160   -31 2.67         -11.6
```

```
##  4        -1       -18      183    17 3.05           5.57
##  5        -6       -25      116    19 1.93           9.83
##  6        -4        12      150   -16 2.5           -6.4
##  7        -5        19      158   -24 2.63          -9.11
##  8        -3       -14       53    11 0.883         12.5
##  9        -3        -8      140     5 2.33           2.14
## 10        -2         8      138   -10 2.3           -4.35
## # i 336,766 more rows
```

```
# 将 tailnum 重命名为 tail_num
flights |>
  rename(tail_num = tailnum) |>
  select(tail_num)
```

3.3.3 rename() - 重命名列

```
## # A tibble: 336,776 x 1
##    tail_num
##    <chr>
##  1 N14228
##  2 N24211
##  3 N619AA
##  4 N804JB
##  5 N668DN
##  6 N39463
##  7 N516JB
##  8 N829AS
##  9 N593JB
## 10 N3ALAA
## # i 336,766 more rows
```

```
# 将 time_hour 和 air_time 移到最前面
flights |>
  relocate(time_hour, air_time)
```

3.3.4 relocate() - 调整列位置

```
## # A tibble: 336,776 x 19
##    time_hour           air_time  year month   day dep_time sched_dep_time
##    <dttm>                 <dbl> <int> <int> <int>    <int>          <int>
```

```
##  1 2013-01-01 05:00:00      227  2013     1     1      517            515
##  2 2013-01-01 05:00:00      227  2013     1     1      533            529
##  3 2013-01-01 05:00:00      160  2013     1     1      542            540
##  4 2013-01-01 05:00:00      183  2013     1     1      544            545
##  5 2013-01-01 06:00:00      116  2013     1     1      554            600
##  6 2013-01-01 05:00:00      150  2013     1     1      554            558
##  7 2013-01-01 06:00:00      158  2013     1     1      555            600
##  8 2013-01-01 06:00:00       53  2013     1     1      557            600
##  9 2013-01-01 06:00:00      140  2013     1     1      557            600
## 10 2013-01-01 06:00:00      138  2013     1     1      558            600
## # i 336,766 more rows
## # i 12 more variables: dep_delay <dbl>, arr_time <int>, sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>, origin <chr>,
## #   dest <chr>, distance <dbl>, hour <dbl>, minute <dbl>
```

```
# 将 "arr" 开头的列移到 dep_time 之前
flights |>
  relocate(starts_with("arr"), .before = dep_time)
```

```
## # A tibble: 336,776 x 19
##     year month   day arr_time arr_delay dep_time sched_dep_time dep_delay
##    <int> <int> <int>    <int>     <dbl>    <int>          <int>     <dbl>
##  1  2013     1     1      830        11      517            515         2
##  2  2013     1     1      850        20      533            529         4
##  3  2013     1     1      923        33      542            540         2
##  4  2013     1     1     1004       -18      544            545        -1
##  5  2013     1     1      812       -25      554            600        -6
##  6  2013     1     1      740        12      554            558        -4
##  7  2013     1     1      913        19      555            600        -5
##  8  2013     1     1      709       -14      557            600        -3
##  9  2013     1     1      838        -8      557            600        -3
## 10  2013     1     1      753         8      558            600        -2
## # i 336,766 more rows
## # i 11 more variables: sched_arr_time <int>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

3.4 分组与汇总

3.4.1 group_by() - 分组

group_by() 将数据按变量分组，后续操作将以组为单位进行：

```
flights |>
  group_by(month)
```

```
## # A tibble: 336,776 x 19
## # Groups:   month [12]
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      517            515         2      830            819
##  2  2013     1     1      533            529         4      850            830
##  3  2013     1     1      542            540         2      923            850
##  4  2013     1     1      544            545        -1     1004           1022
##  5  2013     1     1      554            600        -6      812            837
##  6  2013     1     1      554            558        -4      740            728
##  7  2013     1     1      555            600        -5      913            854
##  8  2013     1     1      557            600        -3      709            723
##  9  2013     1     1      557            600        -3      838            846
## 10  2013     1     1      558            600        -2      753            745
## # i 336,766 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

3.4.2 summarize() - 汇总统计

summarize() 计算每组的统计量:

```
flights |>
  group_by(month) |>
  summarize(
    avg_delay = mean(dep_delay, na.rm = TRUE),
    flight_count = n()
  )
```

```
## # A tibble: 12 x 3
##    month avg_delay flight_count
##    <int>     <dbl>        <int>
##  1     1      10.0         27004
##  2     2      10.8         24951
##  3     3      13.2         28834
##  4     4      13.9         28330
##  5     5      13.0         28796
##  6     6      20.8         28243
##  7     7      21.7         29425
```

```
##  8     8     12.6          29327
##  9     9      6.72         27574
## 10    10      6.24         28889
## 11    11      5.44         27268
## 12    12     16.6          28135
```

- `na.rm = TRUE` 用于忽略缺失值
- `n()` 返回当前分组的行数

3.4.3 多重分组 可以按多个变量分组:

```
flights |>
  group_by(year, month, day) |>
  summarize(
    avg_delay = mean(dep_delay, na.rm = TRUE),
    .groups = "drop"  # 取消分组
  )
```

```
## # A tibble: 365 x 4
##      year month   day avg_delay
##    <int> <int> <int>     <dbl>
##  1  2013     1     1     11.5
##  2  2013     1     2     13.9
##  3  2013     1     3     11.0
##  4  2013     1     4      8.95
##  5  2013     1     5      5.73
##  6  2013     1     6      7.15
##  7  2013     1     7      5.42
##  8  2013     1     8      2.55
##  9  2013     1     9      2.28
## 10  2013     1    10      2.84
## # i 355 more rows
```

`.groups` 参数控制输出的分组状态: - `"drop_last"`: 保留上层分组（默认）- `"drop"`: 全部取消分组 - `"keep"`: 保留所有分组

```
flights |>
  group_by(month) |>
  summarize(n = n()) |>
  ungroup() |>
  summarize(total = sum(n))
```

3.4.4 ungroup() - 移除分组

```
## # A tibble: 1 x 1
##    total
##    <int>
## 1 336776
```

3.4.5 .by 参数（dplyr 1.1.0+）

.by 参数提供了更简洁的分组语法，分组仅在当前操作中生效：

```
# 传统写法
flights |>
  group_by(month) |>
  summarize(delay = mean(dep_delay, na.rm = TRUE)) |>
  ungroup()
```

```
## # A tibble: 12 x 2
##    month delay
##    <int> <dbl>
##  1     1 10.0
##  2     2 10.8
##  3     3 13.2
##  4     4 13.9
##  5     5 13.0
##  6     6 20.8
##  7     7 21.7
##  8     8 12.6
##  9     9  6.72
## 10    10  6.24
## 11    11  5.44
## 12    12 16.6
```

```
# .by 简化写法
flights |>
  summarize(
    delay = mean(dep_delay, na.rm = TRUE),
    .by = month
  )
```

```
## # A tibble: 12 x 2
##    month delay
##    <int> <dbl>
##  1     1 10.0
```

```
##  2    10  6.24
##  3    11  5.44
##  4    12 16.6
##  5     2 10.8
##  6     3 13.2
##  7     4 13.9
##  8     5 13.0
##  9     6 20.8
## 10     7 21.7
## 11     8 12.6
## 12     9  6.72
```

```
# 多变量分组
flights |>
  summarize(
    delay = mean(dep_delay, na.rm = TRUE),
    n = n(),
    .by = c(origin, dest)
  )
```

```
## # A tibble: 224 x 4
##    origin dest  delay     n
##    <chr>  <chr> <dbl> <int>
##  1 EWR    IAH   11.8   3973
##  2 LGA    IAH    9.06  2951
##  3 JFK    MIA    9.34  3314
##  4 JFK    BQN    6.67   599
##  5 LGA    ATL   11.4  10263
##  6 EWR    ORD   14.6   6100
##  7 EWR    FLL   13.5   3793
##  8 LGA    IAD   16.7   1803
##  9 JFK    MCO   10.6   5464
## 10 LGA    ORD   10.7   8857
## # i 214 more rows
```

3.4.6 slice_*() 系列函数 用于提取组内特定行：

```
# 每个目的地到达延误最长的航班
flights |>
  group_by(dest) |>
  slice_max(arr_delay, n = 1) |>
```

```
  relocate(dest, arr_delay)
```

```
## # A tibble: 108 x 19
## # Groups:   dest [105]
##    dest  arr_delay  year month   day dep_time sched_dep_time dep_delay arr_time
##    <chr>     <dbl> <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1 ABQ         153  2013     7    22     2145           2007        98      132
##  2 ACK         221  2013     7    23     1139            800       219     1250
##  3 ALB         328  2013     1    25      123           2000       323      229
##  4 ANC          39  2013     8    17     1740           1625        75     2042
##  5 ATL         895  2013     7    22     2257            759       898      121
##  6 AUS         349  2013     7    10     2056           1505       351     2347
##  7 AVL         228  2013     8    13     1156            832       204     1417
##  8 BDL         266  2013     2    21     1728           1316       252     1839
##  9 BGR         238  2013    12     1     1504           1056       248     1628
## 10 BHM         291  2013     4    10       25           1900       325      136
## # i 98 more rows
## # i 10 more variables: sched_arr_time <int>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

常用函数：- `slice_head(n = 1)`：每组取最前一行 - `slice_tail(n = 1)`：每组取最后一行 - `slice_max(order_by, n = 1)`：每组取最大值 - `slice_min(order_by, n = 1)`：每组取最小值 - `slice_sample(n = 1)`：每组随机取一行

3.5 综合示例

找出飞往 IAH 的航班中速度最快的几架飞机：

```
flights |>
  filter(dest == "IAH") |>
  mutate(speed = distance / air_time * 60) |>
  select(year:day, dep_time, carrier, flight, speed) |>
  arrange(desc(speed)) |>
  head(10)
```

```
## # A tibble: 10 x 7
##     year month   day dep_time carrier flight speed
##    <int> <int> <int>    <int> <chr>    <int> <dbl>
##  1  2013     7     9      707 UA         226  522.
##  2  2013     8    27     1850 UA        1128  521.
##  3  2013     8    28      902 UA        1711  519.
```

```
##  4  2013     8    28     2122 UA        1022  519.
##  5  2013     6    11     1628 UA        1178  515.
##  6  2013     8    27     1017 UA         333  515.
##  7  2013     8    27     1205 UA        1421  515.
##  8  2013     8    27     1758 UA         302  515.
##  9  2013     9    27      521 UA         252  515.
## 10  2013     8    28      625 UA         559  515.
```

这段代码的逻辑是：1. 筛选目的地为 IAH 的航班 2. 计算飞行速度 3. 选择需要的列 4. 按速度降序排列 5. 取前 10 行

4. tidyr 包：数据整理

tidyr 专门用于数据的”整理”，即将数据转换为”整洁数据”（tidy data）格式。

4.1 整洁数据的定义

整洁数据满足以下三个条件：1. 每个变量占一列 2. 每个观测占一行 3. 每个值占一个单元格

4.2 长宽格式转换

在实际分析中，数据常常需要在”宽格式”和”长格式”之间转换。

4.2.1 pivot_longer() - 宽转长　将多列合并为一列，增加行数：

```
# 创建宽格式数据
df_wide <- tibble(
  gene = c("GeneA", "GeneB", "GeneC"),
  sample1 = c(10, 20, 15),
  sample2 = c(12, 18, 22),
  sample3 = c(8, 25, 19)
)

df_wide
```

```
## # A tibble: 3 x 4
##   gene  sample1 sample2 sample3
##   <chr>   <dbl>   <dbl>   <dbl>
## 1 GeneA      10      12       8
## 2 GeneB      20      18      25
## 3 GeneC      15      22      19
```

```r
# 转换为长格式
df_long <- df_wide |>
  pivot_longer(
    cols = starts_with("sample"),  # 要转换的列
    names_to = "sample",           # 列名存入的新列
    values_to = "expression"       # 值存入的新列
  )

df_long
```

```
## # A tibble: 9 x 3
##   gene  sample  expression
##   <chr> <chr>        <dbl>
## 1 GeneA sample1         10
## 2 GeneA sample2         12
## 3 GeneA sample3          8
## 4 GeneB sample1         20
## 5 GeneB sample2         18
## 6 GeneB sample3         25
## 7 GeneC sample1         15
## 8 GeneC sample2         22
## 9 GeneC sample3         19
```

更复杂的例子:

```r
# 列名包含多个信息
df_complex <- tibble(
  id = 1:3,
  ctrl_day1 = c(1, 2, 3),
  ctrl_day2 = c(4, 5, 6),
  treat_day1 = c(7, 8, 9),
  treat_day2 = c(10, 11, 12)
)

df_complex |>
  pivot_longer(
    cols = -id,
    names_to = c("group", "time"),
    names_sep = "_",
    values_to = "value"
  )
```

```
## # A tibble: 12 x 4
##       id group time  value
##    <int> <chr> <chr> <dbl>
##  1     1 ctrl  day1      1
##  2     1 ctrl  day2      4
##  3     1 treat day1      7
##  4     1 treat day2     10
##  5     2 ctrl  day1      2
##  6     2 ctrl  day2      5
##  7     2 treat day1      8
##  8     2 treat day2     11
##  9     3 ctrl  day1      3
## 10     3 ctrl  day2      6
## 11     3 treat day1      9
## 12     3 treat day2     12
```

4.2.2 pivot_wider() - 长转宽

将一列拆分为多列，减少行数：

```
# 从长格式转回宽格式
df_long |>
  pivot_wider(
    names_from = sample,       # 从哪列取新的列名
    values_from = expression   # 从哪列取值
  )
```

```
## # A tibble: 3 x 4
##   gene  sample1 sample2 sample3
##   <chr>   <dbl>   <dbl>   <dbl>
## 1 GeneA      10      12       8
## 2 GeneB      20      18      25
## 3 GeneC      15      22      19
```

实际应用示例：

```
# 汇总数据通常是长格式
summary_data <- flights |>
  group_by(carrier, month) |>
  summarize(
    avg_delay = mean(dep_delay, na.rm = TRUE),
    .groups = "drop"
```

```
  ) |>
  filter(month <= 3)

summary_data
```

```
## # A tibble: 46 x 3
##    carrier month avg_delay
##    <chr>   <int>     <dbl>
##  1 9E          1    16.9
##  2 9E          2    16.5
##  3 9E          3    13.4
##  4 AA          1     6.93
##  5 AA          2     8.28
##  6 AA          3     8.70
##  7 AS          1     7.35
##  8 AS          2     0.722
##  9 AS          3     8.42
## 10 B6          1     9.49
## # i 36 more rows
```

```
# 转换为宽格式便于查看
summary_data |>
  pivot_wider(
    names_from = month,
    values_from = avg_delay,
    names_prefix = "month_"
  )
```

```
## # A tibble: 16 x 4
##    carrier month_1 month_2 month_3
##    <chr>     <dbl>   <dbl>   <dbl>
##  1 9E        16.9   16.5     13.4
##  2 AA         6.93   8.28     8.70
##  3 AS         7.35   0.722    8.42
##  4 B6         9.49  13.8     14.2
##  5 DL         3.85   5.54     9.93
##  6 EV        24.2   21.5     26.2
##  7 F9        10     29.8     16.8
##  8 FL         1.97   5.18    17.3
##  9 HA        54.4   17.4      1.16
## 10 MQ         6.49   8.09     7.19
```

```
## 11 OO         67     NA       NA
## 12 UA          8.33   7.71    11.7
## 13 US          1.82   0.980    2.72
## 14 VX          1.06   6.61     9.68
## 15 WN          9.14  11.8     15.2
## 16 YV         15.8   10.7     31.9
```

4.3 分离与合并

4.3.1 separate() / separate_wider_delim() - 拆分列 将一列拆分为多列：

```
# 创建示例数据
df_sep <- tibble(
  id = 1:3,
  date = c("2024-01-15", "2024-02-20", "2024-03-25")
)

# 使用 separate_wider_delim() 拆分
df_sep |>
  separate_wider_delim(
    cols = date,
    delim = "-",
    names = c("year", "month", "day")
  )
```

```
## # A tibble: 3 x 4
##      id year  month day
##   <int> <chr> <chr> <chr>
## 1     1 2024  01    15
## 2     2 2024  02    20
## 3     3 2024  03    25
```

按位置拆分：

```
df_pos <- tibble(
  id = 1:3,
  code = c("AB123", "CD456", "EF789")
)

df_pos |>
  separate_wider_position(
    cols = code,
```

```
    widths = c(letters = 2, numbers = 3)
  )
```

```
## # A tibble: 3 x 3
##      id letters numbers
##   <int> <chr>   <chr>
## 1     1 AB      123
## 2     2 CD      456
## 3     3 EF      789
```

4.3.2 unite() - 合并列　将多列合并为一列：

```
df_unite <- tibble(
  year = c(2024, 2024, 2024),
  month = c(1, 2, 3),
  day = c(15, 20, 25)
)

df_unite |>
  unite(
    col = "date",
    year, month, day,
    sep = "-"
  )
```

```
## # A tibble: 3 x 1
##   date
##   <chr>
## 1 2024-1-15
## 2 2024-2-20
## 3 2024-3-25
```

4.4 缺失值处理

```
df_na <- tibble(
  x = c(1, 2, NA, 4),
  y = c("a", NA, "c", "d")
)

# 删除任何列有 NA 的行
df_na |>
```

```
  drop_na()
```

4.4.1 drop_na() - 删除含缺失值的行

```
## # A tibble: 2 x 2
##       x y
##   <dbl> <chr>
## 1     1 a
## 2     4 d
```

```
# 只针对特定列
df_na |>
  drop_na(x)
```

```
## # A tibble: 3 x 2
##       x y
##   <dbl> <chr>
## 1     1 a
## 2     2 <NA>
## 3     4 d
```

4.4.2 fill() - 填充缺失值

用前一个或后一个值填充 NA：

```
df_fill <- tibble(
  group = c("A", NA, NA, "B", NA, "C"),
  value = 1:6
)

# 向下填充
df_fill |>
  fill(group, .direction = "down")
```

```
## # A tibble: 6 x 2
##   group value
##   <chr> <int>
## 1 A         1
## 2 A         2
## 3 A         3
## 4 B         4
## 5 B         5
## 6 C         6
```

```r
# 向上填充
df_fill |>
  fill(group, .direction = "up")
```

```
## # A tibble: 6 x 2
##   group value
##   <chr> <int>
## 1 A         1
## 2 B         2
## 3 B         3
## 4 B         4
## 5 C         5
## 6 C         6
```

```r

df_na |>
  replace_na(list(x = 0, y = "missing"))
```

4.4.3 replace_na() - 替换缺失值

```
## # A tibble: 4 x 2
##       x y
##   <dbl> <chr>
## 1     1 a
## 2     2 missing
## 3     0 c
## 4     4 d
```

4.5 嵌套与展开

4.5.1 nest() - 嵌套数据　将数据按组嵌套为列表列：

```r
flights_nested <- flights |>
  select(carrier, flight, origin, dest, air_time) |>
  group_by(carrier) |>
  nest()

flights_nested
```

```
## # A tibble: 16 x 2
## # Groups:   carrier [16]
##    carrier data
```

```
##    <chr>   <list>
##  1 UA      <tibble [58,665 x 4]>
##  2 AA      <tibble [32,729 x 4]>
##  3 B6      <tibble [54,635 x 4]>
##  4 DL      <tibble [48,110 x 4]>
##  5 EV      <tibble [54,173 x 4]>
##  6 MQ      <tibble [26,397 x 4]>
##  7 US      <tibble [20,536 x 4]>
##  8 WN      <tibble [12,275 x 4]>
##  9 VX      <tibble [5,162 x 4]>
## 10 FL      <tibble [3,260 x 4]>
## 11 AS      <tibble [714 x 4]>
## 12 9E      <tibble [18,460 x 4]>
## 13 F9      <tibble [685 x 4]>
## 14 HA      <tibble [342 x 4]>
## 15 YV      <tibble [601 x 4]>
## 16 OO      <tibble [32 x 4]>
```

```
flights_nested |>
  unnest(data) |>
  head(10)
```

4.5.2 unnest() - 展开嵌套

```
## # A tibble: 10 x 5
## # Groups:   carrier [1]
##    carrier flight origin dest  air_time
##    <chr>    <int> <chr>  <chr>    <dbl>
##  1 UA        1545 EWR    IAH        227
##  2 UA        1714 LGA    IAH        227
##  3 UA        1696 EWR    ORD        150
##  4 UA         194 JFK    LAX        345
##  5 UA        1124 EWR    SFO        361
##  6 UA        1187 EWR    LAS        337
##  7 UA        1077 EWR    MIA        157
##  8 UA         303 JFK    SFO        366
##  9 UA         496 LGA    IAH        229
## 10 UA        1665 EWR    LAX        366
```

5. dplyr 与 tidyr 结合使用

5.1 数据清洗完整流程

假设我们有一个基因表达数据集：

```
# 创建模拟数据
gene_expr <- tibble(
  gene_id = paste0("Gene", 1:5),
  ctrl_rep1 = c(100, 200, 150, NA, 300),
  ctrl_rep2 = c(110, 190, 160, 180, 290),
  treat_rep1 = c(150, 180, 200, 220, 350),
  treat_rep2 = c(140, 175, 210, 215, 340)
)

gene_expr
```

```
## # A tibble: 5 x 5
##   gene_id ctrl_rep1 ctrl_rep2 treat_rep1 treat_rep2
##   <chr>       <dbl>     <dbl>      <dbl>      <dbl>
## 1 Gene1         100       110        150        140
## 2 Gene2         200       190        180        175
## 3 Gene3         150       160        200        210
## 4 Gene4          NA       180        220        215
## 5 Gene5         300       290        350        340
```

完整的清洗和分析流程：

```
gene_expr |>
  # 1. 宽格式转长格式
  pivot_longer(
    cols = -gene_id,
    names_to = "sample",
    values_to = "expression"
  ) |>
  # 2. 拆分样本信息
  separate_wider_delim(
    cols = sample,
    delim = "_",
    names = c("group", "replicate")
  ) |>
  # 3. 删除缺失值
  drop_na() |>
```

```r
  # 4. 分组汇总
  group_by(gene_id, group) |>
  summarize(
    mean_expr = mean(expression),
    sd_expr = sd(expression),
    .groups = "drop"
  ) |>
  # 5. 计算 fold change
  pivot_wider(
    names_from = group,
    values_from = c(mean_expr, sd_expr)
  ) |>
  mutate(
    fold_change = mean_expr_treat / mean_expr_ctrl,
    log2FC = log2(fold_change)
  ) |>
  # 6. 按 fold change 排序
  arrange(desc(abs(log2FC)))
```

```
## # A tibble: 5 x 7
##   gene_id mean_expr_ctrl mean_expr_treat sd_expr_ctrl sd_expr_treat fold_change
##   <chr>            <dbl>           <dbl>        <dbl>         <dbl>       <dbl>
## 1 Gene1              105            145          7.07          7.07       1.38
## 2 Gene3              155            205          7.07          7.07       1.32
## 3 Gene4              180            218.        NA             3.54       1.21
## 4 Gene5              295            345          7.07          7.07       1.17
## 5 Gene2              195            178.         7.07          3.54       0.910
## # i 1 more variable: log2FC <dbl>
```

5.2 表格连接

dplyr 提供了多种表格连接函数，类似于 SQL 的 JOIN 操作：

```r
# 创建示例表
df1 <- tibble(
  id = c(1, 2, 3, 4),
  value1 = c("a", "b", "c", "d")
)

df2 <- tibble(
  id = c(2, 3, 4, 5),
```

```
  value2 = c("x", "y", "z", "w")
)
```

```
# 内连接：只保留两表都有的记录
inner_join(df1, df2, by = "id")
```

```
## # A tibble: 3 x 3
##      id value1 value2
##   <dbl> <chr>  <chr>
## 1     2 b      x
## 2     3 c      y
## 3     4 d      z
```

```
# 左连接：保留左表所有记录
left_join(df1, df2, by = "id")
```

```
## # A tibble: 4 x 3
##      id value1 value2
##   <dbl> <chr>  <chr>
## 1     1 a      <NA>
## 2     2 b      x
## 3     3 c      y
## 4     4 d      z
```

```
# 右连接：保留右表所有记录
right_join(df1, df2, by = "id")
```

```
## # A tibble: 4 x 3
##      id value1 value2
##   <dbl> <chr>  <chr>
## 1     2 b      x
## 2     3 c      y
## 3     4 d      z
## 4     5 <NA>   w
```

```
# 全连接：保留所有记录
full_join(df1, df2, by = "id")
```

```
## # A tibble: 5 x 3
##      id value1 value2
##   <dbl> <chr>  <chr>
## 1     1 a      <NA>
## 2     2 b      x
```

```
## 3     3 c       y
## 4     4 d       z
## 5     5 <NA>    w
```

当连接键名称不同时：

```
df3 <- tibble(
  sample_id = c(2, 3, 4, 5),
  value3 = c("p", "q", "r", "s")
)

left_join(df1, df3, by = c("id" = "sample_id"))
```

```
## # A tibble: 4 x 3
##      id value1 value3
##   <dbl> <chr>  <chr>
## 1     1 a      <NA>
## 2     2 b      p
## 3     3 c      q
## 4     4 d      r
```

6. 实战练习

使用 `flights` 数据集完成以下任务：

练习 1：数据筛选与排序

找出 2013 年 7 月从 JFK 机场起飞、延误超过 60 分钟的航班，并按延误时间降序排列：

```
flights |>
  filter(
    month == 7,
    origin == "JFK",
    dep_delay > 60
  ) |>
  arrange(desc(dep_delay)) |>
  select(month, day, carrier, flight, dep_delay, dest)
```

```
## # A tibble: 1,396 x 6
##    month   day carrier flight dep_delay dest
##    <int> <int> <chr>    <int>     <dbl> <chr>
##  1     7    22 MQ        3075      1005 CVG
##  2     7    10 VX         411       634 LAX
##  3     7     7 VX          23       629 SFO
```

```
##  4     7     6 DL         141       589 SFO
##  5     7    27 EV        5716       536 IAD
##  6     7    28 MQ        3075       486 CVG
##  7     7    10 DL        1643       471 SEA
##  8     7    10 B6         415       453 SFO
##  9     7     7 DL         503       452 SAN
## 10     7    10 VX          27       432 SFO
## # i 1,386 more rows
```

练习 2：分组汇总

计算每个航空公司的平均延误时间和航班总数，并按平均延误时间排序：

```
flights |>
  group_by(carrier) |>
  summarize(
    avg_delay = mean(dep_delay, na.rm = TRUE),
    total_flights = n(),
    pct_delayed = mean(dep_delay > 0, na.rm = TRUE) * 100
  ) |>
  arrange(desc(avg_delay))
```

```
## # A tibble: 16 x 4
##    carrier avg_delay total_flights pct_delayed
##    <chr>       <dbl>         <int>       <dbl>
##  1 F9          20.2            685        50
##  2 EV          20.0          54173        45.1
##  3 YV          19.0            601        42.8
##  4 FL          18.7           3260        51.9
##  5 WN          17.7          12275        54.3
##  6 9E          16.7          18460        40.6
##  7 B6          13.0          54635        39.6
##  8 VX          12.9           5162        43.4
##  9 OO          12.6             32        31.0
## 10 UA          12.1          58665        47.0
## 11 MQ          10.6          26397        31.9
## 12 DL           9.26         48110        31.9
## 13 AA           8.59         32729        31.7
## 14 AS           5.80           714        31.7
## 15 HA           4.90           342        20.2
## 16 US           3.78         20536        24.0
```

练习 3：数据转换

创建一个按月份和航空公司汇总的宽格式表格：

```
flights |>
  group_by(month, carrier) |>
  summarize(n = n(), .groups = "drop") |>
  pivot_wider(
    names_from = carrier,
    values_from = n,
    values_fill = 0
  )
```

```
## # A tibble: 12 x 17
##    month  `9E`    AA    AS    B6    DL    EV    F9    FL    HA    MQ    OO    UA
##    <int> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>
##  1     1  1573  2794    62  4427  3690  4171    59   328    31  2271     1  4637
##  2     2  1459  2517    56  4103  3444  3827    49   296    28  2044     0  4346
##  3     3  1627  2787    62  4772  4189  4726    57   316    31  2256     0  4971
##  4     4  1511  2722    60  4517  4092  4561    57   311    30  2211     0  5047
##  5     5  1462  2803    62  4576  4082  4817    58   325    31  2284     0  4960
##  6     6  1437  2757    60  4622  4126  4456    55   252    30  2178     2  4975
##  7     7  1494  2882    62  4984  4251  4641    58   263    31  2261     0  5066
##  8     8  1456  2856    62  4952  4318  4563    55   263    31  2263     4  5124
##  9     9  1540  2614    60  4291  3883  4725    58   255    25  2206    20  4694
## 10    10  1673  2715    62  4361  4093  4908    57   236    21  2228     0  5060
## 11    11  1595  2577    52  4289  3849  4471    61   202    25  2056     5  4854
## 12    12  1633  2705    54  4741  4093  4307    61   213    28  2139     0  4931
## # i 4 more variables: US <int>, VX <int>, WN <int>, YV <int>
```

总结

本节介绍了 tidyverse 生态系统中两个最重要的包：

1. **dplyr**：提供了一套直观的”动词”函数进行数据操作
   - 行操作：`filter()`、`arrange()`、`distinct()`
   - 列操作：`select()`、`mutate()`、`rename()`、`relocate()`
   - 分组汇总：`group_by()`、`summarize()`、`slice_*()`
   - 表格连接：`*_join()` 系列函数
2. **tidyr**：专注于数据整理和格式转换
   - 长宽转换：`pivot_longer()`、`pivot_wider()`

- 列的分离合并：`separate_*()`、`unite()`
- 缺失值处理：`drop_na()`、`fill()`、`replace_na()`
- 嵌套操作：`nest()`、`unnest()`