Manipulação de Dados em R

Soraia Pereira

2025-11-06

Objetivos da aula

- Entender o **pipeline** de manipulação com `%>%` ou `|>`
- Dominar os **verbos principais do dplyr**: `select()`, `filter()`, `arrange()`, `mutate()`, `summarise()`, `group_by()`
- Reestruturar dados com **tidyr**: `pivot_longer()`, `pivot_wider()`, `separate()`, `unite()`
- Realizar **junções** (joins) entre tabelas
- **exercícios** finais

Filosofia tidyverse

- Tabelas como **tibbles**
- Cada passo faz **uma coisa clara**
- Leitura "de cima para baixo" com **pipes**

```
tibble::tibble(modelo = rownames(mtcars),
               mtcars) |>
  dplyr::glimpse()
```

```
Rows: 32
Columns: 12
$ modelo <chr> "Mazda RX4", "Mazda RX4 Wag", "Datsun 710", "Hornet 4 Drive", "~
$ mpg    <dbl> 21.0, 21.0, 22.8, 21.4, 18.7, 18.1, 14.3, 24.4, 22.8, 19.2, 17.~
$ cyl    <dbl> 6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8, 8, 8, 4, 4, 4, 4, ~
$ disp   <dbl> 160.0, 160.0, 108.0, 258.0, 360.0, 225.0, 360.0, 146.7, 140.8, ~
$ hp     <dbl> 110, 110, 93, 110, 175, 105, 245, 62, 95, 123, 123, 180, 180, 1~
```

```
$ drat   <dbl> 3.90, 3.90, 3.85, 3.08, 3.15, 2.76, 3.21, 3.69, 3.92, 3.92, 3.9~
$ wt     <dbl> 2.620, 2.875, 2.320, 3.215, 3.440, 3.460, 3.570, 3.190, 3.150, ~
$ qsec   <dbl> 16.46, 17.02, 18.61, 19.44, 17.02, 20.22, 15.84, 20.00, 22.90, ~
$ vs     <dbl> 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, ~
$ am     <dbl> 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, ~
$ gear   <dbl> 4, 4, 4, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 4, 4, 4, 3, ~
$ carb   <dbl> 4, 4, 1, 1, 2, 1, 4, 2, 2, 4, 4, 3, 3, 3, 4, 4, 4, 1, 2, 1, 1, ~
```

Pipes: %>% vs |>

- %>% (magrittr) e |> (pipe nativo do R) encadeiam passos de forma legível.

```r
mtcars |>
  as_tibble(rownames = "modelo") |>
  select(modelo, mpg, cyl, hp) |>
  arrange(desc(mpg)) |>
  head(5)
```

```
# A tibble: 5 x 4
  modelo           mpg   cyl    hp
  <chr>          <dbl> <dbl> <dbl>
1 Toyota Corolla  33.9     4    65
2 Fiat 128        32.4     4    66
3 Honda Civic     30.4     4    52
4 Lotus Europa    30.4     4   113
5 Fiat X1-9       27.3     4    66
```

Seleção de colunas — `select()`

- Selecionar, **renomear**, **reordenar** colunas
- *Helpers*: `starts_with()`, `ends_with()`, `contains()`, `matches()`

```r
starwars |>
  select(name, height, mass, starts_with("home")) |>
  head(8)
```

```
# A tibble: 8 x 4
  name               height  mass homeworld
  <chr>               <int> <dbl> <chr>
1 Luke Skywalker        172    77 Tatooine
2 C-3PO                 167    75 Tatooine
3 R2-D2                  96    32 Naboo
4 Darth Vader           202   136 Tatooine
5 Leia Organa           150    49 Alderaan
6 Owen Lars             178   120 Tatooine
7 Beru Whitesun Lars    165    75 Tatooine
8 R5-D4                  97    32 Tatooine
```

Filtrar linhas — `filter()`

- Mantém casos que satisfazem condições lógicas

```
starwars |>
  filter(species == "Human", height >= 180) |>
  select(name, height, mass) |>
  arrange(desc(height))
```

```
# A tibble: 17 x 3
   name                height  mass
   <chr>                <int> <dbl>
 1 Darth Vader            202 136
 2 Qui-Gon Jinn           193  89
 3 Dooku                  193  80
 4 Bail Prestor Organa    191  NA
 5 Anakin Skywalker       188  84
 6 Mace Windu             188  84
 7 Raymus Antilles        188  79
 8 Padmé Amidala          185  45
 9 Biggs Darklighter      183  84
10 Boba Fett              183  78.2
11 Ric Olié               183  NA
12 Quarsh Panaka          183  NA
13 Cliegg Lars            183  NA
14 Jango Fett             183  79
15 Obi-Wan Kenobi         182  77
```

```
16 Wilhuff Tarkin          180  NA
17 Han Solo                180  80
```

Ordenação — `arrange()`

- Ordena por colunas (crescente/decrescente)

```
mtcars |>
  as_tibble(rownames = "modelo") |>
  arrange(desc(hp), mpg) |>
  select(modelo, hp, mpg) |>
  head(10)
```

```
# A tibble: 10 x 3
   modelo                hp   mpg
   <chr>              <dbl> <dbl>
 1 Maserati Bora        335  15
 2 Ford Pantera L       264  15.8
 3 Camaro Z28           245  13.3
 4 Duster 360           245  14.3
 5 Chrysler Imperial    230  14.7
 6 Lincoln Continental  215  10.4
 7 Cadillac Fleetwood   205  10.4
 8 Merc 450SLC          180  15.2
 9 Merc 450SE           180  16.4
10 Merc 450SL           180  17.3
```

Criar/transformar variáveis — `mutate()`

- Cria novas colunas a partir das existentes
- Funções úteis: `if_else()`, `case_when()`, `rowSums()`, etc.

```r
mtcars |>
  as_tibble(rownames = "modelo") |>
  mutate(kmpl = mpg * 0.4251,
         potencia_cat = case_when(
           hp < 100 ~ "baixa",
           hp < 150 ~ "média",
           TRUE     ~ "alta"
         )) |>
  select(modelo, mpg, kmpl, hp, potencia_cat) |>
  head(8)
```

```
# A tibble: 8 x 5
  modelo              mpg  kmpl    hp potencia_cat
  <chr>             <dbl> <dbl> <dbl> <chr>
1 Mazda RX4          21    8.93   110 média
2 Mazda RX4 Wag      21    8.93   110 média
3 Datsun 710         22.8  9.69    93 baixa
4 Hornet 4 Drive     21.4  9.10   110 média
5 Hornet Sportabout  18.7  7.95   175 alta
6 Valiant            18.1  7.69   105 média
7 Duster 360         14.3  6.08   245 alta
8 Merc 240D          24.4 10.4     62 baixa
```

Resumos — `summarise()` + `group_by()`

- Agregação por grupos com estatísticas: `mean()`, `sd()`, `n()`, `median()`, `quantile()`, …

```r
nycflights13::flights |>
  group_by(carrier) |>
  summarise(
    n_voos = n(),
    atraso_medio_partida = mean(dep_delay, na.rm = TRUE),
    atraso_medio_chegada = mean(arr_delay, na.rm = TRUE)
  ) |>
  arrange(desc(n_voos)) |>
  head(10)
```

```
# A tibble: 10 x 4
   carrier n_voos atraso_medio_partida atraso_medio_chegada
   <chr>    <int>                <dbl>                <dbl>
 1 UA       58665                12.1                 3.56
 2 B6       54635                13.0                 9.46
 3 EV       54173                20.0                15.8
 4 DL       48110                 9.26                1.64
 5 AA       32729                 8.59                0.364
 6 MQ       26397                10.6                10.8
 7 US       20536                 3.78                2.13
 8 9E       18460                16.7                 7.38
 9 WN       12275                17.7                 9.65
10 VX        5162                12.9                 1.76
```

Várias colunas de uma vez — `across()`

- Aplicar a mesma função a múltiplas colunas

```
mtcars |>
  as_tibble() |>
  summarise(across(c(mpg, hp, wt), list(media = mean, sd = sd)))
```

```
# A tibble: 1 x 6
  mpg_media mpg_sd hp_media hp_sd wt_media wt_sd
      <dbl>  <dbl>    <dbl> <dbl>    <dbl> <dbl>
1      20.1   6.03     147.  68.6     3.22 0.978
```

Reestruturação — `pivot_longer()` / `pivot_wider()`

- Dados "longos" vs "largos"; **cada variável numa coluna**

```
tbl <- tibble(id = 1:3, Jan = c(10, 12, 9), Fev = c(11, 8, 13))

# Largo -> Longo
long <- tbl |>
  pivot_longer(cols = Jan:Fev, names_to = "mes", values_to = "valor")
long
```

```
# A tibble: 6 x 3
     id mes   valor
  <int> <chr> <dbl>
1     1 Jan      10
2     1 Fev      11
3     2 Jan      12
4     2 Fev       8
5     3 Jan       9
6     3 Fev      13
```

```r
# Longo -> Largo
long |>
  pivot_wider(names_from = mes, values_from = valor)
```

```
# A tibble: 3 x 3
     id   Jan   Fev
  <int> <dbl> <dbl>
1     1    10    11
2     2    12     8
3     3     9    13
```

Separar e unir colunas — `separate()` / `unite()`

```r
nomes <- tibble(nome_completo = c("Ana Silva","Bruno Costa","C. Rocha"))

nomes |>
  separate(nome_completo, into = c("primeiro","apelido"), sep = " ") |>
  unite("nome_compacto", primeiro, apelido, sep = "_")
```

```
# A tibble: 3 x 1
  nome_compacto
  <chr>
1 Ana_Silva
2 Bruno_Costa
3 C._Rocha
```

Junções (joins)

- **left_join** (mantém esquerda), **inner_join** (interseção), **full_join** (união), **right_join** (mantém direita)

```
clientes <- tibble(id=1:5, nome=c("Ana","Bruno","Carla","Duarte","Eva"))
compras  <- tibble(id=c(1,1,3,6), total=c(25,15,40,10))

left_join(clientes, compras, by = "id")#se tiverem nomes diferentes usar left_join(clientes,
```

```
# A tibble: 6 x 3
     id nome   total
  <dbl> <chr>  <dbl>
1     1 Ana       25
2     1 Ana       15
3     2 Bruno     NA
4     3 Carla     40
5     4 Duarte    NA
6     5 Eva       NA
```

Tratamento de NAs

- `is.na()`, `replace_na()`, `coalesce()`

```
tibble(a = c(1,NA,3), b = c(10,20,30)) |>
  mutate(a_sem_na = replace_na(a, 0),
         a_fallback= coalesce(a, b))
```

```
# A tibble: 3 x 4
      a     b a_sem_na a_fallback
  <dbl> <dbl>    <dbl>      <dbl>
1     1    10        1          1
2    NA    20        0         20
3     3    30        3          3
```

Mini-exercícios

1. Usando `starwars`:
   - selecione `name`, `species`, `height`, `mass`
   - filtre apenas *Humans* com `height > 170`
   - crie `imc = mass / (height/100)^2`
   - resuma IMC médio por `sex`
2. Usando `nycflights13::flights`:
   - compute o atraso médio de chegada por `dest`
   - junte (`left_join`) com `airports` para obter `name`
   - ordene do maior para o menor atraso
3. Crie uma pequena tabela larga e converta para **longo** com `pivot_longer()`; depois reconverta para largo.

Soluções (exemplo 1)

```
starwars |>
  select(name, species, height, mass, sex) |>
  filter(species == "Human", height > 170) |>
  mutate(imc = mass / (height/100)^2) |>
  group_by(sex) |>
  summarise(n = n(),
            imc_medio = mean(imc, na.rm = TRUE)) |>
  arrange(desc(imc_medio))
```

```
# A tibble: 2 x 3
  sex        n imc_medio
  <chr>  <int>     <dbl>
1 male      20      25.6
2 female     1      13.1
```

Soluções (exemplo 2)

```r
atraso_dest <- flights |>
  group_by(dest) |>
  summarise(atraso_med = mean(arr_delay, na.rm = TRUE))

left_join(atraso_dest, nycflights13::airports, by = c("dest" = "faa")) |>
  select(dest, name, atraso_med) |>
  arrange(desc(atraso_med)) |>
  head(10)
```

```
# A tibble: 10 x 3
   dest  name                           atraso_med
   <chr> <chr>                               <dbl>
 1 CAE   Columbia Metropolitan                41.8
 2 TUL   Tulsa Intl                          33.7
 3 OKC   Will Rogers World                   30.6
 4 JAC   Jackson Hole Airport                28.1
 5 TYS   Mc Ghee Tyson                       24.1
 6 MSN   Dane Co Rgnl Truax Fld              20.2
 7 RIC   Richmond Intl                       20.1
 8 CAK   Akron Canton Regional Airport       19.7
 9 DSM   Des Moines Intl                     19.0
10 GRR   Gerald R Ford Intl                  18.2
```

Junção de tabelas — conceitos

- **Chaves (keys)**: variáveis que identificam registos para ligar tabelas.
  - Chave simples (ex.: `id`) ou composta (ex.: `year`, `month`, `day`, `hour`, `origin`).

- **Nomes diferentes**: use `by = c("colA" = "colB")`.

```r
# Tabelas do nycflights13 para exemplos
library(nycflights13)
list(
  flights = dim(flights),
  airlines = dim(airlines),
```

```
  airports = dim(airports),
  planes = dim(planes),
  weather = dim(weather)
)
```

```
$flights
[1] 336776     19

$airlines
[1] 16  2

$airports
[1] 1458    8

$planes
[1] 3322    9

$weather
[1] 26115    15
```

---

left_join (o mais usado)

- Mantém **todas** as linhas da tabela da esquerda.
- Junta colunas da direita quando há correspondência.

```
# Adicionar o nome da companhia (carrier) aos voos
flights |>
  select(year, month, day, carrier, flight, origin, dest) |>
  left_join(airlines, by = "carrier") |>
  select(year, month, day, carrier, name, flight, origin, dest) |>
  head(6)
```

```
# A tibble: 6 x 8
   year month   day carrier name                  flight origin dest
  <int> <int> <int> <chr>   <chr>                  <int> <chr>  <chr>
1  2013     1     1 UA      United Air Lines Inc.   1545 EWR    IAH
2  2013     1     1 UA      United Air Lines Inc.   1714 LGA    IAH
3  2013     1     1 AA      American Airlines Inc.  1141 JFK    MIA
```

```
4  2013     1     1 B6      JetBlue Airways           725 JFK    BQN
5  2013     1     1 DL      Delta Air Lines Inc.      461 LGA    ATL
6  2013     1     1 UA      United Air Lines Inc.    1696 EWR    ORD
```

---

inner_join, right_join, full_join

- `inner_join`: mantém **apenas** correspondências (interseção).
- `right_join`: mantém todas as linhas da **direita**.
- `full_join`: **união** (tudo de ambos).

```r
# Ex.: destinos (dest) com info de aeroportos (airports)
base <- flights |>
  count(dest, name = "n_voos")

inner_join(base, airports, by = c("dest" = "faa")) |>   # só destinos com aeroporto conhecido
  select(dest, n_voos, name, lat, lon) |>
  arrange(desc(n_voos)) |>
  head(5)
```

```
# A tibble: 5 x 5
  dest  n_voos name                                lat    lon
  <chr>  <int> <chr>                               <dbl>  <dbl>
1 ORD    17283 Chicago Ohare Intl                   42.0  -87.9
2 ATL    17215 Hartsfield Jackson Atlanta Intl     33.6  -84.4
3 LAX    16174 Los Angeles Intl                    33.9 -118.
4 BOS    15508 General Edward Lawrence Logan Intl  42.4  -71.0
5 MCO    14082 Orlando Intl                        28.4  -81.3
```

```r
# full_join para ver destinos sem metadados (ou metadados sem voos)
full_join(base, airports, by = c("dest" = "faa")) |>
  summarise(
    sem_meta = sum(is.na(name)),
    sem_voos = sum(is.na(n_voos))
  )
```

```
# A tibble: 1 x 2
  sem_meta sem_voos
     <int>    <int>
1        4     1357
```