

R para Ciência de Dados I

Manipulando bases de dados



Manipulando bases de dados

0 pacote dplyr

O `dplyr` é o pacote mais útil para realizar manipulação de dados, pois possui funções para fazer virtualmente qualquer tipo de transformação nas linhas e colunas da base.

As principais funções do `dplyr` são:

- `filter()`: filtra linhas
- `select()`: seleciona colunas
- `arrange()`: ordena as linhas conforme os valores de uma coluna
- `mutate()`: modifica ou cria novas colunas
- `group_by()`: agrupa a base conforme uma coluna
- `summarise()`: sumariza colunas

Todas essas funções seguem as seguintes características:

- A função sempre recebe uma `tibble` e sempre devolve uma `tibble`.
- Colocamos o `tibble` no primeiro argumento e o que queremos fazer nos demais argumentos.

Os exemplos apresentados aqui continuarão a usar a base IMDB. Não se esqueça de carregar o pacote `tidyverse` e carregar a base de dados.

```
library(tidyverse)

imdb <- read_rds("dados/imdb.rds")
```

Selecionando colunas

Para selecionar colunas, utilizamos a função `select()`.

O primeiro argumento da função é a base de dados e os demais argumentos são os nomes das colunas que você gostaria de selecionar. Repare que você não precisa colocar o nome da coluna entre aspas.

```
select(imdb, titulo)
```

```
## # A tibble: 3,713 x 1
##   titulo
##   <chr>
## 1 Avatar
## 2 Pirates of the Caribbean: At World's End
## 3 The Dark Knight Rises
## 4 John Carter
## 5 Spider-Man 3
## 6 Tangled
## 7 Avengers: Age of Ultron
## 8 Batman v Superman: Dawn of Justice
## 9 Superman Returns
## 10 Pirates of the Caribbean: Dead Man's Chest
## # ... with 3,703 more rows
```

Você também pode selecionar várias colunas.

```
select(imdb, titulo, ano, orcamento)
```

```
## # A tibble: 3,713 x 3
##   titulo                                ano orcamento
##   <chr>                                <int>      <int>
## 1 Avatar                                2009 2370000000
## 2 Pirates of the Caribbean: At World's End 2007 3000000000
## 3 The Dark Knight Rises                  2012 2500000000
## 4 John Carter                           2012 2637000000
## 5 Spider-Man 3                           2007 2580000000
## 6 Tangled                               2010 2600000000
## 7 Avengers: Age of Ultron                 2015 2500000000
## 8 Batman v Superman: Dawn of Justice       2016 2500000000
## 9 Superman Returns                       2006 2090000000
## 10 Pirates of the Caribbean: Dead Man's Chest 2006 2250000000
## # ... with 3,703 more rows
```

Um operador `:` é muito útil para selecionar colunas consecutivas.

```
select(imdb, titulo:cor)
```

```
## # A tibble: 3,713 x 5
##   titulo                                ano diretor          duracao cor
##   <chr>                                <int> <chr>          <int> <chr>
## 1 Avatar                                2009 James Cameron      178 Color
## 2 Pirates of the Caribbean: At World's End 2007 Gore Verbinski      169 Color
## 3 The Dark Knight Rises                  2012 Christopher Nol~    164 Color
## 4 John Carter                           2012 Andrew Stanton      132 Color
## 5 Spider-Man 3                           2007 Sam Raimi           156 Color
## 6 Tangled                               2010 Nathan Greno        100 Color
## 7 Avengers: Age of Ultron                 2015 Joss Whedon          141 Color
## 8 Batman v Superman: Dawn of Justice       2016 Zack Snyder          183 Color
## 9 Superman Returns                       2006 Bryan Singer         169 Color
## 10 Pirates of the Caribbean: Dead Man's Ch~ 2006 Gore Verbinski      151 Color
## # ... with 3,703 more rows
```

O `dplyr` possui o conjunto de funções auxiliares muito úteis para seleção de colunas. As principais são:

- `starts_with()`: para colunas que começam com um texto padrão
- `ends_with()`: para colunas que terminam com um texto padrão
- `contains()`: para colunas que contêm um texto padrão

Selecionamos a seguir todas as colunas que começam com o texto "ator".

```
select(imdb, starts_with("ator"))
```

```
## # A tibble: 3,713 x 3
##   ator_1      ator_2      ator_3
##   <chr>      <chr>      <chr>
## 1 CCH Pounder  Joel David Moore Wes Studi
## 2 Johnny Depp  Orlando Bloom  Jack Davenport
## 3 Tom Hardy    Christian Bale  Joseph Gordon-Levitt
## 4 Daryl Sabara Samantha Morton  Polly Walker
## 5 J.K. Simmons James Franco    Kirsten Dunst
## 6 Brad Garrett Donna Murphy    M.C. Gainey
## 7 Chris Hemsworth Robert Downey Jr. Scarlett Johansson
## 8 Henry Cavill  Lauren Cohan    Alan D. Purwin
## 9 Kevin Spacey Marlon Brando    Frank Langella
## 10 Johnny Depp  Orlando Bloom    Jack Davenport
## # ... with 3,703 more rows
```


Ordenando linhas

Para ordenar linhas, utilizamos a função `arrange()`. O primeiro argumento é a base de dados. Os demais argumentos são as colunas pelas quais queremos ordenar as linhas. No exemplo a seguir, ordenamos as linhas da base por ordem crescente de orçamento.

```
arrange(imdb, orcamento)
```

```
## # A tibble: 3,713 x 15
##   titulo    ano diretor duracao cor   generos pais classificacao orcamento
##   <chr>  <int> <chr>      <int> <chr> <chr>   <chr> <chr>          <int>
## 1 Tarna~   2003 Jonath~    88 Color Biogra~ USA   Outros           218
## 2 My Da~   2004 Jon Gu~    90 Color Docume~ USA   Livre            1100
## 3 A Pla~   2013 Benjam~    76 Color Drama|~ USA   Outros            1400
## 4 The M~   2005 Anthon~    84 Color Crime|~ USA   A partir de ~    3250
## 5 Primer   2004 Shane ~    77 Color Drama|~ USA   A partir de ~    7000
## 6 El Ma~   1992 Robert~    81 Color Action~ USA   A partir de ~    7000
## 7 Newly~   2011 Edward~    95 Color Comedy~ USA   Outros            9000
## 8 Pink ~   1972 John W~   108 Color Comedy~ USA   A partir de ~   10000
## 9 The T~   2007 Jane C~     7 Color Romanc~ USA   Outros           13000
## 10 Paran~  2007 Oren P~    84 Color Horror  USA   A partir de ~   15000
## # ... with 3,703 more rows, and 6 more variables: receita <int>,
## #   nota_imdb <dbl>, likes_facebook <int>, ator_1 <chr>, ator_2 <chr>,
## #   ator_3 <chr>
```

Também podemos ordenar de forma decrescente usando a função `desc()`.

```
arrange(imdb, desc(orcamento))
```

```
## # A tibble: 3,713 x 15
##   titulo    ano diretor duracao cor   generos pais classificacao orcamento
##   <chr>    <int> <chr>      <int> <chr> <chr>   <chr> <chr>          <int>
## 1 Pirat~   2007 Gore V~    169 Color Action~ USA   A partir de ~ 3000000000
## 2 John ~   2012 Andrew~    132 Color Action~ USA   A partir de ~ 2637000000
## 3 Tangl~   2010 Nathan~    100 Color Advent~ USA   Livre          2600000000
## 4 Spide~   2007 Sam Ra~    156 Color Action~ USA   A partir de ~ 2580000000
## 5 The D~   2012 Christ~    164 Color Action~ USA   A partir de ~ 2500000000
## 6 Aveng~   2015 Joss W~    141 Color Action~ USA   A partir de ~ 2500000000
## 7 Batma~   2016 Zack S~    183 Color Action~ USA   A partir de ~ 2500000000
## 8 Pirat~   2011 Rob Ma~    136 Color Action~ USA   A partir de ~ 2500000000
## 9 Capta~   2016 Anthon~    147 Color Action~ USA   A partir de ~ 2500000000
## 10 Avatar  2009 James ~    178 Color Action~ USA   A partir de ~ 2370000000
## # ... with 3,703 more rows, and 6 more variables: receita <int>,
## #   nota_imdb <dbl>, likes_facebook <int>, ator_1 <chr>, ator_2 <chr>,
## #   ator_3 <chr>
```

E claro, ordenar segundo duas ou mais colunas.

```
arrange(imdb, desc(ano), desc(orcamento))
```

```
## # A tibble: 3,713 x 15
##   titulo    ano diretor duracao cor   generos pais classificacao orcamento
##   <chr>  <int> <chr>      <int> <chr> <chr>  <chr> <chr>          <int>
## 1 Batma~   2016 Zack S~    183 Color Action~ USA   A partir de ~ 2500000000
## 2 Capta~   2016 Anthon~    147 Color Action~ USA   A partir de ~ 2500000000
## 3 Star ~   2016 Justin~    122 Color Action~ USA   A partir de ~ 1850000000
## 4 The L~   2016 David ~    110 Color Action~ USA   A partir de ~ 1800000000
## 5 X-Men~   2016 Bryan ~    144 Color Action~ USA   A partir de ~ 1780000000
## 6 Suici~   2016 David ~    123 Color Action~ USA   A partir de ~ 1750000000
## 7 Alice~   2016 James ~    113 Color Advent~ USA   Livre          1700000000
## 8 Indep~   2016 Roland~    120 Color Action~ USA   A partir de ~ 1650000000
## 9 Warcr~   2016 Duncan~    123 Color Action~ USA   A partir de ~ 1600000000
## 10 Kung ~   2016 Alessa~     95 Color Action~ USA   Livre          1450000000
## # ... with 3,703 more rows, and 6 more variables: receita <int>,
## #   nota_imdb <dbl>, likes_facebook <int>, ator_1 <chr>, ator_2 <chr>,
## #   ator_3 <chr>
```

Aplicando mais de uma operação

Na grande maioria dos casos, vamos aplicar mais de uma função de manipulação em uma base para obtermos a tabela que desejamos. Poderíamos, por exemplo, querer uma tabela apenas com o título e ano dos filmes, ordenada de forma crescente de lançamento. Para fazer isso, poderíamos aninhar as funções

```
arrange(select(imdb, titulo, ano), ano)
```

ou criar um objeto intermediário

```
tab_titulo_ano <- select(imdb, titulo, ano)
arrange(tab_titulo_ano, ano)
```

Os dois códigos funcionam e levam ao mesmo resultado, mas não são eficientes.

A primeira alternativa é ruim de escrever, já que precisamos escrever primeiro a função que roda por último, e de ler, pois é difícil identificar qual argumento pertence a qual função.

A segunda alternativa é ruim pois exige a criação de objetos auxiliares. Se quiséssimos aplicar 10 operações na base, precisaríamos criar 9 objetos intermediários.

A solução para aplicar diversas operações de manipulação em uma base de dados é aplicar o operador pipe: `%>%`.

O operador pipe %>%

A ideia do operador pipe é a seguinte: ele vai aplicar a função do lado direito ao objeto do lado esquerdo.

No exemplo a seguir, estamos aplicando a função `sum()` (lado direito) no objeto `vetor` (lado esquerdo).

```
vetor <- c(1, 2, 3)
vetor %>% sum()
```

```
## [1] 6
```

O código acima é equivalente a:

```
sum(vetor)
```

```
## [1] 6
```

Na prática, o pipe coloca o objeto do lado esquerdo no primeiro argumento da função no lado direito. Se precisarmos passar mais argumentos para a função, podemos fazer isso normalmente. É como se estivéssemos escrevendo a função, omitindo o primeiro argumento.

```
vetor <- c(1, 2, 3, NA)

vetor %>% sum(na.rm = TRUE)
```

```
## [1] 6
```

O código acima é equivalente a

```
sum(vetor, na.rm = TRUE)
```

```
## [1] 6
```

Quando estamos aplicando apenas uma função, o pipe não parece trazer vantagens. Mas vamos ver como fica o nosso exemplo do imdb utilizando esse operador:

```
# Sem pipe
arrange(select(imdb, titulo, ano), ano)

# Com pipe
imdb %>%
  select(titulo, ano) %>%
  arrange(ano)
```

O que está sendo feito no código com pipe? Da primeira para a segunda linha, estamos aplicando a função `select()` à base `imdb`. Da segunda para a terceira, estamos aplicando a função `arrange()` à base resultante da função `select()`.

O resultado desse código é idêntico às tentativas sem pipe, com a vantagem de termos escrito o código na ordem em que as funções são aplicadas, de termos um código muito mais legível e de não precisarmos utilizar objetos intermediários.

Filtrando linhas

Para filtrar valores de uma coluna da base, utilizamos a função `filter()`.

```
imdb %>% filter(nota_imdb > 9)
```

```
## # A tibble: 3 x 15
##   titulo    ano diretor duracao cor   generos pais classificacao orcamento
##   <chr>  <int> <chr>    <int> <chr> <chr>  <chr> <chr>          <int>
## 1 The S~  1994 Frank ~    142 Color Crime|~ USA   A partir de ~ 25000000
## 2 The G~  1972 Franci~    175 Color Crime|~ USA   A partir de ~  6000000
## 3 Kickb~  2016 John S~     90 <NA> Action  USA   Outros          17000000
## # ... with 6 more variables: receita <int>, nota_imdb <dbl>,
## #   likes_facebook <int>, ator_1 <chr>, ator_2 <chr>, ator_3 <chr>
```

Podemos seleccionar apenas as colunas título e nota para visualizarmos as notas:

```
imdb %>%  
  filter(nota_imdb > 9) %>%  
  select(titulo, nota_imdb)
```

```
## # A tibble: 3 x 2  
##   titulo          nota_imdb  
##   <chr>          <dbl>  
## 1 The Shawshank Redemption    9.3  
## 2 The Godfather              9.2  
## 3 Kickboxer: Vengeance      9.1
```

Podemos estender o filtro para duas ou mais colunas. Para isso, separamos cada operação por uma vírgula.

```
imdb %>% filter(ano > 2010, nota_imdb > 8.5)
```

```
## # A tibble: 5 x 15
##   titulo    ano diretor duracao cor   generos pais classificacao orcamento
##   <chr>   <int> <chr>    <int> <chr> <chr>   <chr> <chr>          <int>
## 1 Inter~   2014 Christ~    169 Color Advent~ USA   A partir de ~ 165000000
## 2 Runni~   2015 Mike M~     88 Color Family USA   Outros          5000000
## 3 A Beg~   2016 Mitche~    87 Color Comedy~ USA   Outros           NA
## 4 Kickb~   2016 John S~     90 <NA> Action  USA   Outros        17000000
## 5 Butte~   2014 Cary B~     78 Color Docume~ USA   Outros          180000
## # ... with 6 more variables: receita <int>, nota_imdb <dbl>,
## #   likes_facebook <int>, ator_1 <chr>, ator_2 <chr>, ator_3 <chr>
```

Também podemos fazer operações com as colunas da base dentro da função filter. O código abaixo devolve uma tabela apenas com os filmes que lucraram.

```
imdb %>% filter(receita - orcamento > 0)
```

```
## # A tibble: 1,710 x 15
##   titulo    ano diretor duracao cor   generos pais classificacao orcamento
##   <chr>    <int> <chr>      <int> <chr> <chr>   <chr> <chr>          <int>
## 1 Avatar    2009 James ~    178 Color Action~ USA   A partir de ~ 2370000000
## 2 Pirat~    2007 Gore V~    169 Color Action~ USA   A partir de ~ 3000000000
## 3 The D~    2012 Christ~    164 Color Action~ USA   A partir de ~ 2500000000
## 4 Spide~    2007 Sam Ra~    156 Color Action~ USA   A partir de ~ 2580000000
## 5 Aveng~    2015 Joss W~    141 Color Action~ USA   A partir de ~ 2500000000
## 6 Batma~    2016 Zack S~    183 Color Action~ USA   A partir de ~ 2500000000
## 7 Pirat~    2006 Gore V~    151 Color Action~ USA   A partir de ~ 2250000000
## 8 Man o~    2013 Zack S~    143 Color Action~ USA   A partir de ~ 2250000000
## 9 The A~    2012 Joss W~    173 Color Action~ USA   A partir de ~ 2200000000
## 10 The A~    2012 Marc W~    153 Color Action~ USA   A partir de ~ 2300000000
## # ... with 1,700 more rows, and 6 more variables: receita <int>,
## #   nota_imdb <dbl>, likes_facebook <int>, ator_1 <chr>, ator_2 <chr>,
## #   ator_3 <chr>
```

Naturalmente, podemos filtrar colunas categóricas. O exemplo abaixo retorna uma tabela apenas com os filmes com a Angelina Jolie Pitt ou o Brad Pitt no papel principal.

```
imdb %>%  
  filter(ator_1 %in% c('Angelina Jolie Pitt', 'Brad Pitt'))
```

```
## # A tibble: 29 x 15  
##   titulo    ano diretor duracao cor   generos pais classificacao orcamento  
##   <chr>  <int> <chr>    <int> <chr> <chr>  <chr> <chr>          <int>  
## 1 Malef~  2014 Robert~    97 Color Action~ USA   Livre          1800000000  
## 2 The C~  2008 David ~   166 Color Drama|~ USA   A partir de ~ 1500000000  
## 3 Kung ~  2011 Jennif~    90 Color Action~ USA   Livre          1500000000  
## 4 Troy    2004 Wolfga~   196 Color Advent~ USA   A partir de ~ 1750000000  
## 5 Kung ~  2008 Mark O~    92 Color Action~ USA   Livre          1300000000  
## 6 Salt    2010 Philli~   101 Color Action~ USA   A partir de ~ 1100000000  
## 7 Ocean~  2004 Steven~   125 Color Crime|~ USA   A partir de ~ 1100000000  
## 8 Mr. &~  2005 Doug L~   126 Color Action~ USA   A partir de ~ 1200000000  
## 9 Lara ~  2001 Simon ~   100 Color Action~ USA   A partir de ~ 1150000000  
## 10 Ocean~ 2001 Steven~   116 Color Crime|~ USA   A partir de ~ 850000000  
## # ... with 19 more rows, and 6 more variables: receita <int>, nota_imdb <dbl>,  
## #   likes_facebook <int>, ator_1 <chr>, ator_2 <chr>, ator_3 <chr>
```

Para filtrar textos sem correspondência exata, podemos utilizar a função auxiliar `str_detect()`. Ela serve para verificar se cada string de um vetor contém um determinado padrão de texto.

```
str_detect(  
  string = c("a", "aa", "abc", "bc", "A", NA),  
  pattern = "a"  
)
```

```
## [1] TRUE TRUE TRUE FALSE FALSE NA
```

Podemos utilizá-la para filtrar apenas os filmes que contêm o gênero ação.

```
# A coluna gêneros apresenta todos os gêneros dos filmes concatenados  
imdb$generos[1:6]
```

```
## [1] "Action|Adventure|Fantasy|Sci-Fi"  
## [2] "Action|Adventure|Fantasy"  
## [3] "Action|Thriller"  
## [4] "Action|Adventure|Sci-Fi"  
## [5] "Action|Adventure|Romance"  
## [6] "Adventure|Animation|Comedy|Family|Fantasy|Musical|Romance"
```

```
# Podemos detectar se o gênero Action aparece na string
str_detect(
  string = imdb$generos[1:6],
  pattern = "Action"
)
```

```
## [1] TRUE TRUE TRUE TRUE TRUE FALSE
```

```
# Aplicamos essa lógica dentro da função filter, para a coluna completa
imdb %>% filter(str_detect(generos, "Action"))
```

```
## # A tibble: 832 x 15
##   titulo    ano diretor duracao cor   generos pais classificacao orcamento
##   <chr>    <int> <chr>      <int> <chr> <chr>   <chr> <chr>          <int>
## 1 Avatar    2009 James ~    178 Color Action~ USA    A partir de ~ 2370000000
## 2 Pirat~    2007 Gore V~    169 Color Action~ USA    A partir de ~ 3000000000
## 3 The D~    2012 Christ~    164 Color Action~ USA    A partir de ~ 2500000000
## 4 John ~    2012 Andrew~    132 Color Action~ USA    A partir de ~ 2637000000
## 5 Spide~    2007 Sam Ra~    156 Color Action~ USA    A partir de ~ 2580000000
## 6 Aveng~    2015 Joss W~    141 Color Action~ USA    A partir de ~ 2500000000
## 7 Batma~    2016 Zack S~    183 Color Action~ USA    A partir de ~ 2500000000
## 8 Super~    2006 Bryan ~    169 Color Action~ USA    A partir de ~ 2090000000
## 9 Pirat~    2006 Gore V~    151 Color Action~ USA    A partir de ~ 2250000000
## 10 The L~    2013 Gore V~    150 Color Action~ USA    A partir de ~ 2150000000
## # ... with 822 more rows, and 6 more variables: receita <int>, nota_imdb <dbl>,
## #   likes_facebook <int>, ator_1 <chr>, ator_2 <chr>, ator_3 <chr>
```

Modificando e criando novas colunas

Para modificar uma coluna existente ou criar uma nova coluna, utilizamos a função `mutate()`. O código abaixo divide os valores da coluna duração por 60, mudando a unidade de medida dessa variável de minutos para horas.

```
imdb %>% mutate(duracao = duracao/60)
```

```
## # A tibble: 3,713 x 15
##   titulo    ano diretor duracao cor   generos pais classificacao orcamento
##   <chr>    <int> <chr>    <dbl> <chr> <chr>   <chr> <chr>          <int>
## 1 Avatar    2009 James ~    2.97 Color Action~ USA   A partir de ~ 2370000000
## 2 Pirat~    2007 Gore V~    2.82 Color Action~ USA   A partir de ~ 3000000000
## 3 The D~    2012 Christ~    2.73 Color Action~ USA   A partir de ~ 2500000000
## 4 John ~    2012 Andrew~    2.2  Color Action~ USA   A partir de ~ 2637000000
## 5 Spide~    2007 Sam Ra~    2.6  Color Action~ USA   A partir de ~ 2580000000
## 6 Tangl~    2010 Nathan~    1.67 Color Advent~ USA   Livre          2600000000
## 7 Aveng~    2015 Joss W~    2.35 Color Action~ USA   A partir de ~ 2500000000
## 8 Batma~    2016 Zack S~    3.05 Color Action~ USA   A partir de ~ 2500000000
## 9 Super~    2006 Bryan ~    2.82 Color Action~ USA   A partir de ~ 2090000000
## 10 Pirat~    2006 Gore V~    2.52 Color Action~ USA   A partir de ~ 2250000000
## # ... with 3,703 more rows, and 6 more variables: receita <int>,
## #   nota_imdb <dbl>, likes_facebook <int>, ator_1 <chr>, ator_2 <chr>,
## #   ator_3 <chr>
```


Também poderíamos ter criado essa variável em uma nova coluna. Repare que a nova coluna `duracao_horas` é colocada no final da tabela.

```
imdb %>% mutate(duracao_horas = duracao/60)
```

```
## # A tibble: 3,713 x 16
##   titulo    ano diretor duracao cor   generos pais classificacao orcamento
##   <chr>    <int> <chr>    <int> <chr> <chr>   <chr> <chr>          <int>
## 1 Avatar    2009 James ~    178 Color Action~ USA   A partir de ~ 2370000000
## 2 Pirat~    2007 Gore V~    169 Color Action~ USA   A partir de ~ 3000000000
## 3 The D~    2012 Christ~    164 Color Action~ USA   A partir de ~ 2500000000
## 4 John ~    2012 Andrew~    132 Color Action~ USA   A partir de ~ 2637000000
## 5 Spide~    2007 Sam Ra~    156 Color Action~ USA   A partir de ~ 2580000000
## 6 Tangl~    2010 Nathan~    100 Color Advent~ USA   Livre          2600000000
## 7 Aveng~    2015 Joss W~    141 Color Action~ USA   A partir de ~ 2500000000
## 8 Batma~    2016 Zack S~    183 Color Action~ USA   A partir de ~ 2500000000
## 9 Super~    2006 Bryan ~    169 Color Action~ USA   A partir de ~ 2090000000
## 10 Pirat~    2006 Gore V~    151 Color Action~ USA   A partir de ~ 2250000000
## # ... with 3,703 more rows, and 7 more variables: receita <int>,
## #   nota_imdb <dbl>, likes_facebook <int>, ator_1 <chr>, ator_2 <chr>,
## #   ator_3 <chr>, duracao_horas <dbl>
```

Podemos fazer qualquer operação com uma ou mais colunas. A única regra é que o resultado da operação retorne um vetor com comprimento igual ao número de linhas da base (ou com comprimento 1 para distribuir um mesmo valor em todas as linhas). Você também pode criar/modificar quantas colunas quiser dentro de um mesmo `mutate`.

```
imdb %>%  
  mutate(lucro = receita - orcamento, pais = "Estados Unidos") %>%  
  select(titulo, lucro, pais)
```

```
## # A tibble: 3,713 x 3  
##   titulo                                lucro pais  
##   <chr>                                <int> <chr>  
## 1 Avatar                               523505847 Estados Unidos  
## 2 Pirates of the Caribbean: At World's End    9404152 Estados Unidos  
## 3 The Dark Knight Rises                      198130642 Estados Unidos  
## 4 John Carter                             -190641321 Estados Unidos  
## 5 Spider-Man 3                             78530303 Estados Unidos  
## 6 Tangled                                -59192738 Estados Unidos  
## 7 Avengers: Age of Ultron                  208991599 Estados Unidos  
## 8 Batman v Superman: Dawn of Justice         80249062 Estados Unidos  
## 9 Superman Returns                        -8930592 Estados Unidos  
## 10 Pirates of the Caribbean: Dead Man's Chest 198032628 Estados Unidos  
## # ... with 3,703 more rows
```

Sumarizando colunas

Sumarização é a técnica de se resumir um conjunto de dados utilizando alguma métrica de interesse. A média, a mediana, a variância, a frequência, a proporção, por exemplo, são tipos de sumarização que trazem diferentes informações sobre uma variável.

Para sumarizar uma coluna da base, utilizamos a função `summarize()`. O código abaixo resume a coluna orçamento pela sua média.

```
imdb %>% summarize(media_orcamento = mean(orçamento, na.rm = TRUE))
```

```
## # A tibble: 1 x 1
##   media_orcamento
##               <dbl>
## 1           35237114.
```

Repare que a saída da função continua sendo uma tibble.

Podemos calcular diversas sumarizações diferentes em um mesmo `summarize`. Cada sumarização será uma coluna da nova base.

```
imdb %>% summarise(  
  media_orcamento = mean(orcamento, na.rm = TRUE),  
  mediana_orcamento = median(orcamento, na.rm = TRUE),  
  variancia_orcamento = var(orcamento, na.rm = TRUE)  
)
```

```
## # A tibble: 1 x 3  
##   media_orcamento mediana_orcamento variancia_orcamento  
##           <dbl>           <dbl>           <dbl>  
## 1      35237114.      20000000      1769050050704312.
```

E também sumarizar diversas colunas.

```
imdb %>% summarize(  
  media_orcamento = mean(orcamento, na.rm = TRUE),  
  media_receita = mean(receita, na.rm = TRUE),  
  media_lucro = mean(receita - orcamento, na.rm = TRUE)  
)
```

```
## # A tibble: 1 x 3  
##   media_orcamento media_receita media_lucro  
##           <dbl>         <dbl>         <dbl>  
## 1      35237114.      54530550.      17161639.
```

Sumarizando colunas agrupadas

Muitas vezes queremos sumarizar uma coluna agrupada pelas categorias de uma segunda coluna. Para isso, além do `summarize`, utilizamos também a função `group_by()`.

O código a seguir calcula a receita média dos filmes para cada categoria da coluna "cor".

```
imdb %>%  
  group_by(cor) %>%  
  summarise(receita_media = mean(receita, na.rm = TRUE))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 3 x 2  
##   cor          receita_media  
##   <chr>          <dbl>  
## 1 Black and White 36124154.  
## 2 Color          55094030.  
## 3 <NA>          80014842
```

A única alteração que a função `group_by()` faz na base é a marcação de que a base está agrupada.

```
imdb %>% group_by(cor)
```

```
## # A tibble: 3,713 x 15
## # Groups:   cor [3]
##   titulo    ano diretor duracao cor   generos pais classificacao orcamento
##   <chr>    <int> <chr>      <int> <chr> <chr>   <chr> <chr>          <int>
## 1 Avatar   2009 James ~    178 Color Action~ USA   A partir de ~ 2370000000
## 2 Pirat~   2007 Gore V~    169 Color Action~ USA   A partir de ~ 3000000000
## 3 The D~   2012 Christ~    164 Color Action~ USA   A partir de ~ 2500000000
## 4 John ~   2012 Andrew~    132 Color Action~ USA   A partir de ~ 2637000000
## 5 Spide~   2007 Sam Ra~    156 Color Action~ USA   A partir de ~ 2580000000
## 6 Tangl~   2010 Nathan~    100 Color Advent~ USA   Livre          2600000000
## 7 Aveng~   2015 Joss W~    141 Color Action~ USA   A partir de ~ 2500000000
## 8 Batma~   2016 Zack S~    183 Color Action~ USA   A partir de ~ 2500000000
## 9 Super~   2006 Bryan ~    169 Color Action~ USA   A partir de ~ 2090000000
## 10 Pirat~   2006 Gore V~    151 Color Action~ USA   A partir de ~ 2250000000
## # ... with 3,703 more rows, and 6 more variables: receita <int>,
## #   nota_imdb <dbl>, likes_facebook <int>, ator_1 <chr>, ator_2 <chr>,
## #   ator_3 <chr>
```

Juntando bases a partir de uma chave

Podemos juntar duas tabelas a partir de uma (coluna) chave utilizando a função `left_join()`. Como exemplo, vamos inicialmente calcular o lucro médio dos filmes de cada diretor e salvar no objeto `tab_lucro_diretor`.

```
tab_lucro_diretor <- imdb %>%  
  group_by(diretor) %>%  
  summarise(lucro_medio = mean(receita - orcamento, na.rm = TRUE))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
tab_lucro_diretor
```

```
## # A tibble: 1,813 x 2  
##   diretor      lucro_medio  
##   <chr>         <dbl>  
## 1 A. Raven Cruz      NaN  
## 2 Aaron Hann         NaN  
## 3 Aaron Schneider   1676553  
## 4 Aaron Seltzer     28546578  
## 5 Abel Ferrara     -11272676  
## 6 Adam Carolla     -1394057  
## 7 Adam Goldberg    -1647420  
## 8 Adam Green        NaN  
## 9 Adam Jay Epstein   NaN  
## 10 Adam Marcus     13435068  
## #> with 1,803 more rows
```


E se quisermos colocar essa informação na base original? Basta usar a função `left_join()` utilizando a coluna `diretor` como chave. Observe que a coluna `lucro_medio` aparece agora no fim da tabela.

```
imdb_com_lucro_medio <- left_join(imdb, tab_lucro_diretor, by = "diretor")  
imdb_com_lucro_medio
```

```
## # A tibble: 3,713 x 16  
##   titulo    ano diretor duracao cor   generos pais classificacao orcamento  
##   <chr>   <int> <chr>      <int> <chr> <chr>   <chr> <chr>          <int>  
## 1 Avatar   2009 James ~    178 Color Action~ USA   A partir de ~ 2370000000  
## 2 Pirat~   2007 Gore V~    169 Color Action~ USA   A partir de ~ 3000000000  
## 3 The D~   2012 Christ~    164 Color Action~ USA   A partir de ~ 2500000000  
## 4 John ~   2012 Andrew~    132 Color Action~ USA   A partir de ~ 2637000000  
## 5 Spide~   2007 Sam Ra~    156 Color Action~ USA   A partir de ~ 2580000000  
## 6 Tangl~   2010 Nathan~    100 Color Advent~ USA   Livre          2600000000  
## 7 Aveng~   2015 Joss W~    141 Color Action~ USA   A partir de ~ 2500000000  
## 8 Batma~   2016 Zack S~    183 Color Action~ USA   A partir de ~ 2500000000  
## 9 Super~   2006 Bryan ~    169 Color Action~ USA   A partir de ~ 2090000000  
## 10 Pirat~  2006 Gore V~    151 Color Action~ USA   A partir de ~ 2250000000  
## # ... with 3,703 more rows, and 7 more variables: receita <int>,  
## #   nota_imdb <dbl>, likes_facebook <int>, ator_1 <chr>, ator_2 <chr>,  
## #   ator_3 <chr>, lucro_medio <dbl>
```

Na tabela `imdb_com_lucro_medio`, como na tabela `imdb`, cada linha continua a representar um filme diferente, mas agora temos também a informação do lucro médio do diretor de cada filme.

A primeira linha, por exemplo, traz as informações do filme Avatar. O valor do `lucro_medio` nessa linha representa o lucro médio de todos os filmes do James Cameron, que é o diretor de Avatar. Com essa informação, podemos calcular o quanto o lucro do Avatar se afasta do lucro médio do James Cameron.

```
imdb_com_lucro_medio %>%
  mutate(
    lucro = receita - orcamento,
    lucro_relativo = (lucro - lucro_medio)/lucro_medio,
    lucro_relativo = scales::percent(lucro_relativo)
  ) %>%
  select(titulo, diretor, lucro, lucro_medio, lucro_relativo)
```

```
## # A tibble: 3,713 x 5
##   titulo                diretor      lucro lucro_medio lucro_relativo
##   <chr>                 <chr>      <int>      <dbl> <chr>
## 1 Avatar                James Cameron  5.24e8  194620985  168.987359%
## 2 Pirates of the Caribbean: ~ Gore Verbins~  9.40e6   36942999. -74.544157%
## 3 The Dark Knight Rises  Christopher ~  1.98e8  101028447  96.113716%
## 4 John Carter            Andrew Stant~ -1.91e8   46668146 -508.504167%
## 5 Spider-Man 3           Sam Raimi      7.85e7   66820940.  17.523494%
## 6 Tangled                Nathan Greno -5.92e7  -59192738  0.000000%
## 7 Avengers: Age of Ultron Joss Whedon    2.09e8  199202360.  4.914218%
## 8 Batman v Superman: Dawn of~ Zack Snyder    8.02e7   33186745. 141.810582%
## 9 Superman Returns       Bryan Singer  -8.93e6  -2887024.  209.335596%
## 10 Pirates of the Caribbean: ~ Gore Verbins~  1.98e8   36942999. 436.049135%
## # ... with 3,703 more rows
```

Observamos então que o Avatar obteve um lucro aproximadamente 169% maior que a média dos filmes do James Cameron.