Metodologias Informacionais com R

Módulo I: Introdução à Linguagem R

Telmo dos Santos Klipp (telmo.klipp@inpe.br)

Informações Gerais sobre o Curso

- Materiais disponibilizados via Classroom;
- O aprendizado requer a prática que será constante nas aulas;

Bibliografia Básica:

- Kennedy, R., & Waggoner, P. D. (2021). Introduction to r for social scientists: a tidy programming approach. CRC Press.

Bibliografia Complementar:

- Wickham, H., Çetinkaya-Rundel, M., & Grolemund, G. (2023). R for data science (2e): import, tidy, transform, visualize, and model data. "O'Reilly Media, Inc.". Disponível em: https://r4ds.hadley.nz/. Acesso em: 14 de junho, 2023. (Online)
- Damiani, A. et. al., (2022). Ciência de Dados em R. Curso-R. Disponível em: https://livro.curso-r.com. Acesso em: 12 de maio, 2023. (Online)
- de Aquino, J. A. (2014). R para cientistas sociais. Editora da UESC (editus). Disponível em: http://www.uesc.br/editora/. Acesso em: 12 de maio, 2023.
- de Oliveira, P. F., Guerra, S., McDonnell, R. (2018). Ciência de Dados com R: Introdução. Editora IBPAD. Disponível em: https://cdr.ibpad.com.br/index.html. Acesso em: 12 de maio, 2023. (Online)

Nas últimas aulas vimos:

- Definição de uma sessão do R.
- Como nos manter organizados usando R project.
- Diretórios (caminhos absolutos e relativos), pastas e tipos de arquivos.
- Pacotes no R - instalar, carregar e obter fontes de informações como manuais, *vignettes* e *Cheatsheets*.
- Como salvar e carregar dados (ex: arquivos csv).
- Conceitos iniciais de dados brutos e tratados (arrumados/organizados).
- Onde ficaremos no ciclo da ciência dos dados?

> **Agora focaremos em *Data Wrangling*** com o pacote **dplyr** do **tidyverse**.

Tidyverse

Coleção de pacotes que possuem filosofia de design, gramática e estrutura de dados em comum e, permitem trabalho conjunto, clareza de código, reprodutibilidade, dentre outros benefícios.

Manipulação de dados

Dados tratados no formato tidy permitem:

- uma série de operações e manipulações (*Data Wrangling*).
- prosseguir com outras etapas da ciência de dados como exploração, modelagem e análise.

Nesse momento, é comum usar o pacote **dplyr**. Este provém uma lista intuitiva de verbos que permitem executar as tarefas mais comuns da manipulação de dados. O **dplyr** possui um site próprio, onde podem ser encontradas informações desse pacote, bem como, alguns *vignettes* e *cheat sheets.*

Manipulação de dados

As principais funções do **dplyr** são:

- select() - seleciona colunas usando os nomes
- filter() - filtra linhas verificando condições lógicas sobre os valores
- slice() - seleciona linhas usando as posições (índices) das mesmas
- arrange() - ordena (reorganiza) as linhas
- mutate() - cria/modifica colunas
- summarise() - sumarização de valores
- group_by() - agrupamento de dados

O primeiro argumento de qualquer uma das funções acima é o **dataframe** ou **tibble** de dados. Ex:

```
dplyr::slice(mtcars, 1:2)
```

```
##               mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4      21   6  160 110  3.9 2.620 16.46  0  1    4    4
## Mazda RX4 Wag  21   6  160 110  3.9 2.875 17.02  0  1    4    4
```

Manipulação de ⊞ dados

Vejamos exemplos de uso das funções do **dplyr** através do dataframe **mtcars**. Informações sobre este conjunto de dados ficam disponíveis com **?mtcars**. O **mtcars** possui a classe **data.frame**, mas iremos convertê-lo para a classe **tibble** que é a estrutura de dados comum aos pacotes do **tidyverse**.

```
mtcars <- tibble::as_tibble(mtcars, rownames = 'model')
dplyr::glimpse(mtcars) # glimpse mostra as colunas e parte dos valores de forma transposta
```

```
## Rows: 32
## Columns: 12
## $ model <chr> "Mazda RX4", "Mazda RX4 Wag", "Datsun 710", "Hornet 4 Drive", "H…
## $ mpg   <dbl> 21.0, 21.0, 22.8, 21.4, 18.7, 18.1, 14.3, 24.4, 22.8, 19.2, 17.8…
## $ cyl   <dbl> 6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8, 8, 8, 4, 4, 4, 4, 8…
## $ disp  <dbl> 160.0, 160.0, 108.0, 258.0, 360.0, 225.0, 360.0, 146.7, 140.8, 1…
## $ hp    <dbl> 110, 110, 93, 110, 175, 105, 245, 62, 95, 123, 123, 180, 180, 18…
## $ drat  <dbl> 3.90, 3.90, 3.85, 3.08, 3.15, 2.76, 3.21, 3.69, 3.92, 3.92, 3.92…
## $ wt    <dbl> 2.620, 2.875, 2.320, 3.215, 3.440, 3.460, 3.570, 3.190, 3.150, 3…
## $ qsec  <dbl> 16.46, 17.02, 18.61, 19.44, 17.02, 20.22, 15.84, 20.00, 22.90, 1…
## $ vs    <dbl> 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0…
## $ am    <dbl> 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0…
## $ gear  <dbl> 4, 4, 4, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 4, 4, 4, 3, 3…
## $ carb  <dbl> 4, 4, 1, 1, 2, 1, 4, 2, 2, 4, 4, 3, 3, 3, 4, 4, 4, 1, 2, 1, 1, 2…
```

Manipulação de ⊞ dados - seleção

Para selecionar colunas usamos a função select().

```
library(dplyr) # Importação para uso direto das funções, ou seja, sem indição do pacote
```

```
select(mtcars, gear)
```

```
## # A tibble: 32 × 1
##     gear
##    <dbl>
##  1     4
##  2     4
##  3     4
##  4     3
##  5     3
##  6     3
##  7     3
##  8     4
##  9     4
## 10     4
## # ℹ 22 more rows
```

Manipulação de dados - seleção

Como selecionar multiplas colunas?

```
# select(mtcars, c(mpg, cyl, gear))
select(mtcars, mpg, cyl, gear)
```

```
## # A tibble: 32 × 3
##      mpg   cyl  gear
##    <dbl> <dbl> <dbl>
##  1  21       6     4
##  2  21       6     4
##  3  22.8     4     4
##  4  21.4     6     3
##  5  18.7     8     3
##  6  18.1     6     3
##  7  14.3     8     3
##  8  24.4     4     4
##  9  22.8     4     4
## 10  19.2     6     4
## # ℹ 22 more rows
```

Manipulação de ⊞ dados - seleção

Como excluir colunas da seleção?

```
# select(mtcars, -mpg, -cyl, -gear)
select(mtcars, -c(mpg, cyl, gear))
```

```
## # A tibble: 32 × 9
##    model              disp    hp  drat    wt  qsec    vs    am  carb
##    <chr>             <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 Mazda RX4           160   110  3.9   2.62  16.5     0     1     4
##  2 Mazda RX4 Wag       160   110  3.9   2.88  17.0     0     1     4
##  3 Datsun 710          108    93  3.85  2.32  18.6     1     1     1
##  4 Hornet 4 Drive      258   110  3.08  3.22  19.4     1     0     1
##  5 Hornet Sportabout   360   175  3.15  3.44  17.0     0     0     2
##  6 Valiant             225   105  2.76  3.46  20.2     1     0     1
##  7 Duster 360          360   245  3.21  3.57  15.8     0     0     4
##  8 Merc 240D           147.   62  3.69  3.19  20       1     0     2
##  9 Merc 230            141.   95  3.92  3.15  22.9     1     0     2
## 10 Merc 280            168.  123  3.92  3.44  18.3     1     0     4
## # ℹ 22 more rows
```

Manipulação de ⊞ dados - seleção

Selecionar colunas consecutivas? Use o operador : (*Colon*).

```
select(mtcars, model:hp) # ou select(mtcars, 1:5)
```

```
## # A tibble: 32 × 5
##    model               mpg   cyl  disp    hp
##    <chr>             <dbl> <dbl> <dbl> <dbl>
##  1 Mazda RX4          21       6  160    110
##  2 Mazda RX4 Wag      21       6  160    110
##  3 Datsun 710         22.8     4  108     93
##  4 Hornet 4 Drive     21.4     6  258    110
##  5 Hornet Sportabout  18.7     8  360    175
##  6 Valiant            18.1     6  225    105
##  7 Duster 360         14.3     8  360    245
##  8 Merc 240D          24.4     4  147.    62
##  9 Merc 230           22.8     4  141.    95
## 10 Merc 280           19.2     6  168.   123
## # ℹ 22 more rows
```

Manipulação de dados - seleção

Mais informações?

```
?select
```

Leia mais sobre em Curso R.

Exercícios

1. Crie uma tabela somente com as colunas **model** e **hp** (cavalos de potência) atribuindo-a a um objeto de nome **modelo_potencia**.

2. Exclua as colunas que possuem valores iguais a zero (obs: não precisa atribuir a um novo objeto).

Manipulação de dados - filtragem

Para filtrar linhas usamos a função filter().

```
filter(mtcars, gear > 4)
```

```
## # A tibble: 5 × 12
##   model          mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
##   <chr>        <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Porsche 914…  26       4 120.     91  4.43  2.14  16.7     0     1     5     2
## 2 Lotus Europa  30.4     4  95.1   113  3.77  1.51  16.9     1     1     5     2
## 3 Ford Panter…  15.8     8 351     264  4.22  3.17  14.5     0     1     5     4
## 4 Ferrari Dino  19.7     6 145     175  3.62  2.77  15.5     0     1     5     6
## 5 Maserati Bo…  15       8 301     335  3.54  3.57  14.6     0     1     5     8
```

```
filter(mtcars, gear > 4, hp < 100) # ou filter(mtcars, gear > 4 & hp < 100)
```

```
## # A tibble: 1 × 12
##   model          mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
##   <chr>        <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Porsche 914…    26     4  120.    91  4.43  2.14  16.7     0     1     5     2
```

Manipulação de ⊞ dados - filtragem

Podemos filtrar linhas categóricas, conforme:

```
filter(mtcars, model %in% c('Mazda RX4', 'Toyota Corona', 'Pontiac Firebird'))
```

```
## # A tibble: 3 × 12
##   model          mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
##   <chr>        <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Mazda RX4     21       6  160    110  3.9   2.62  16.5     0     1     4     4
## 2 Toyota Coro…  21.5     4  120.    97  3.7   2.46  20.0     1     0     3     1
## 3 Pontiac Fir…  19.2     8  400    175  3.08  3.84  17.0     0     0     3     2
```

Manipulação de ⊞ dados - filtragem

Podemos combinar as funções filter() e stringr::str_detect() para realizar uma filtragem. Nesse caso, stringr::str_detect() retornará os índices das linhas cujos textos possuam correspondências parciais, ou seja, possuem o texto que se deseja encontrar. A função filter(), por sua vez, fará a filtragem das linhas baseado nesses índices, conforme:

```
filter(mtcars, stringr::str_detect(model, 'Mazda|Hornet'))
```

```
## # A tibble: 4 × 12
##   model          mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
##   <chr>        <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Mazda RX4     21       6   160   110  3.9   2.62  16.5     0     1     4     4
## 2 Mazda RX4 W…  21       6   160   110  3.9   2.88  17.0     0     1     4     4
## 3 Hornet 4 Dr…  21.4     6   258   110  3.08  3.22  19.4     1     0     3     1
## 4 Hornet Spor…  18.7     8   360   175  3.15  3.44  17.0     0     0     3     2
```

Exercícios

1. Quais modelos de veículos possuem 8 cilindros (coluna **cyl**) ?
2. Quais modelos de veículos possuem potência maior que 300 cavalos (coluna **hp**)?
3. Quais modelos de veículos possuem menos de 100 cavalos de potência?
4. Retorne todos os veículos do tipo 'Merc' (abreviação de Mercedes Benz) que possuem 180 cavalos de potência.
5. Quais modelos de veículos possuem entre 200 e 300 cavalos de potência e pesam (coluna **wt**) mais de 5000 libras?

Operador pipe

- O operador pipe serve para encadear operações:
  - a saída da operação à esquerda serve de entrada para a operação à direita.
- São opções básicas para o uso do pipe:
  - `|>` proveniente do pacote **rbase**;
  - `%>%` proveniente do pacote **magrittr**.
- Vantagens:
  - organização de código e fluxo de operações;
  - facilitar entendimento de código;
  - reprodutibilidade.

Leia mais sobre na boa descrição em Curso R.

Manipulação de ⊞ dados - ordenação

Podemos ordenar as linhas com a função arrange().

```
mtcars |> arrange(desc(hp)) # para ordenar de forma crescente basta não usar desc()
```

```
## # A tibble: 32 × 12
##    model          mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
##    <chr>        <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 Maserati B…  15        8  301    335  3.54  3.57  14.6     0     1     5     8
##  2 Ford Pante…  15.8      8  351    264  4.22  3.17  14.5     0     1     5     4
##  3 Duster 360   14.3      8  360    245  3.21  3.57  15.8     0     0     3     4
##  4 Camaro Z28   13.3      8  350    245  3.73  3.84  15.4     0     0     3     4
##  5 Chrysler I…  14.7      8  440    230  3.23  5.34  17.4     0     0     3     4
##  6 Lincoln Co…  10.4      8  460    215  3     5.42  17.8     0     0     3     4
##  7 Cadillac F…  10.4      8  472    205  2.93  5.25  18.0     0     0     3     4
##  8 Merc 450SE   16.4      8  276.   180  3.07  4.07  17.4     0     0     3     3
##  9 Merc 450SL   17.3      8  276.   180  3.07  3.73  17.6     0     0     3     3
## 10 Merc 450SLC  15.2      8  276.   180  3.07  3.78  18       0     0     3     3
## # ℹ 22 more rows
```

Manipulação de dados - ordenação

Podem ser usadas duas ou mais colunas na ordenação.

```
mtcars |>
  arrange(cyl, desc(hp))
```

```
## # A tibble: 32 × 12
##    model          mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
##    <chr>        <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 Lotus Euro…  30.4      4  95.1   113  3.77  1.51  16.9     1     1     5     2
##  2 Volvo 142E   21.4      4 121     109  4.11  2.78  18.6     1     1     4     2
##  3 Toyota Cor…  21.5      4 120.     97  3.7   2.46  20.0     1     0     3     1
##  4 Merc 230     22.8      4 141.     95  3.92  3.15  22.9     1     0     4     2
##  5 Datsun 710   22.8      4 108      93  3.85  2.32  18.6     1     1     4     1
##  6 Porsche 91…  26        4 120.     91  4.43  2.14  16.7     0     1     5     2
##  7 Fiat 128     32.4      4  78.7    66  4.08  2.2   19.5     1     1     4     1
##  8 Fiat X1-9    27.3      4  79      66  4.08  1.94  18.9     1     1     4     1
##  9 Toyota Cor…  33.9      4  71.1    65  4.22  1.84  19.9     1     1     4     1
## 10 Merc 240D    24.4      4 147.     62  3.69  3.19  20       1     0     4     2
## # ℹ 22 more rows
```

Exercícios

- Retorne os modelos da marca da Mercedes Benz e ordene-os de forma decrescente segundo seu peso.
- Retorne os veículos que possuem menos de 100 cavalos de potência e ordene-os de forma crescente segundo sua potência.

Manipulação de ⊞ dados - criação/modificação

Para modificar colunas existentes ou criar colunas novas usamos a função mutate(), conforme:

```
mtcars |>
  mutate(wt =  wt * 1000)
```

```
## # A tibble: 32 × 12
##    model          mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
##    <chr>        <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 Mazda RX4     21       6  160    110  3.9   2620  16.5     0     1     4     4
##  2 Mazda RX4 …   21       6  160    110  3.9   2875  17.0     0     1     4     4
##  3 Datsun 710    22.8     4  108     93  3.85  2320  18.6     1     1     4     1
##  4 Hornet 4 D…   21.4     6  258    110  3.08  3215  19.4     1     0     3     1
##  5 Hornet Spo…   18.7     8  360    175  3.15  3440  17.0     0     0     3     2
##  6 Valiant       18.1     6  225    105  2.76  3460  20.2     1     0     3     1
##  7 Duster 360    14.3     8  360    245  3.21  3570  15.8     0     0     3     4
##  8 Merc 240D     24.4     4  147.    62  3.69  3190  20       1     0     4     2
##  9 Merc 230      22.8     4  141.    95  3.92  3150  22.9     1     0     4     2
## 10 Merc 280      19.2     6  168.   123  3.92  3440  18.3     1     0     4     4
## # ℹ 22 more rows
```

Manipulação de ⊞ dados - criação/modificação

Nesse exemplo, transformaremos as colunas **vs** e **am** de numéricas para categóricas. Enquanto **vs** indica o tipo de motor (0 = V-shaped, 1 = straight), **am** indica o tipo de transmissão (0 = automatic, 1 = manual).

```
mtcars |>
  mutate(vs = factor(vs, labels = c("V", "S")),
         am = factor(am, labels = c("automatic", "manual")))
```

```
## # A tibble: 32 × 12
##    model          mpg   cyl  disp    hp  drat    wt  qsec vs    am     gear  carb
##    <chr>        <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <fct> <fct> <dbl> <dbl>
##  1 Mazda RX4     21       6  160    110  3.9   2.62  16.5 V     manu…     4     4
##  2 Mazda RX4 …   21       6  160    110  3.9   2.88  17.0 V     manu…     4     4
##  3 Datsun 710    22.8     4  108     93  3.85  2.32  18.6 S     manu…     4     1
##  4 Hornet 4 D…   21.4     6  258    110  3.08  3.22  19.4 S     auto…     3     1
##  5 Hornet Spo…   18.7     8  360    175  3.15  3.44  17.0 V     auto…     3     2
##  6 Valiant       18.1     6  225    105  2.76  3.46  20.2 S     auto…     3     1
##  7 Duster 360    14.3     8  360    245  3.21  3.57  15.8 V     auto…     3     4
##  8 Merc 240D     24.4     4  147.    62  3.69  3.19  20   S     auto…     4     2
##  9 Merc 230      22.8     4  141.    95  3.92  3.15  22.9 S     auto…     4     2
## 10 Merc 280      19.2     6  168.   123  3.92  3.44  18.3 S     auto…     4     4
## # ℹ 22 more rows
```

Manipulação de ⊞ dados - criação/modificação

Aqui criaremos uma nova coluna (**power_to_weight**), dada a razão entre a potência em cavalos e o peso do veículo, conforme:

```
mtcars |>
  select( -c(vs, am)) |> # excluindo algumas colunas para melhorar a visualização abaixo
  mutate(power_to_weight =  hp / (wt * 1000))
```

```
## # A tibble: 32 × 11
##    model      mpg   cyl  disp    hp  drat    wt  qsec  gear  carb power_to_weight
##    <chr>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>           <dbl>
##  1 Mazda …  21       6  160    110  3.9   2.62  16.5     4     4          0.0420
##  2 Mazda …  21       6  160    110  3.9   2.88  17.0     4     4          0.0383
##  3 Datsun…  22.8     4  108     93  3.85  2.32  18.6     4     1          0.0401
##  4 Hornet…  21.4     6  258    110  3.08  3.22  19.4     3     1          0.0342
##  5 Hornet…  18.7     8  360    175  3.15  3.44  17.0     3     2          0.0509
##  6 Valiant  18.1     6  225    105  2.76  3.46  20.2     3     1          0.0303
##  7 Duster…  14.3     8  360    245  3.21  3.57  15.8     3     4          0.0686
##  8 Merc 2…  24.4     4  147.    62  3.69  3.19  20       4     2          0.0194
##  9 Merc 2…  22.8     4  141.    95  3.92  3.15  22.9     4     2          0.0302
## 10 Merc 2…  19.2     6  168.   123  3.92  3.44  18.3     4     4          0.0358
## # ℹ 22 more rows
```

Manipulação de ⊞ dados - criação/modificação

Podemos criar colunas categóricas considerando valores de outras colunas com if_else() ou case_when(). A título apenas de demonstração (sem abordagem técnica ou ciêntifica) segue um exemplo:

```
mtcars |> select(-c(qsec, vs, am)) |>
  mutate(wt =  wt * 1000, wt_class = case_when(wt < 2581 ~ "lightweight",
                      between(wt, 2581, 3610)  ~ "middleweight", wt > 3610 ~ "heavyweight" ))
```

```
## # A tibble: 32 × 10
##    model              mpg   cyl  disp    hp  drat    wt  gear  carb wt_class
##    <chr>            <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>
##  1 Mazda RX4         21       6  160    110  3.9   2620     4     4 middleweig…
##  2 Mazda RX4 Wag     21       6  160    110  3.9   2875     4     4 middleweig…
##  3 Datsun 710        22.8     4  108     93  3.85  2320     4     1 lightweight
##  4 Hornet 4 Drive    21.4     6  258    110  3.08  3215     3     1 middleweig…
##  5 Hornet Sportabout 18.7     8  360    175  3.15  3440     3     2 middleweig…
##  6 Valiant           18.1     6  225    105  2.76  3460     3     1 middleweig…
##  7 Duster 360        14.3     8  360    245  3.21  3570     3     4 middleweig…
##  8 Merc 240D         24.4     4  147.    62  3.69  3190     4     2 middleweig…
##  9 Merc 230          22.8     4  141.    95  3.92  3150     4     2 middleweig…
## 10 Merc 280          19.2     6  168.   123  3.92  3440     4     4 middleweig…
## # ℹ 22 more rows
```

Manipulação de dados - criação/modificação

Por fim todas as operações anteriores serão persistidas em um novo conjunto de dados.

```
mtcars_new <- mtcars |>
  mutate(
    wt =  wt * 1000,
    vs = if_else(vs == 0, "V", "S"),
    am = if_else(am == 0, "automatic", "manual"),
    # variáveis novas
    power_to_weight =  hp / wt,
    wt_class = case_when(
      wt < 2581 ~ "lightweight",
      between(wt, 2581, 3610)  ~ "middleweight",
      wt > 3610 ~ "heavyweight")
  ) |>
  mutate(
    # o pacote forcats oferece funcionalidades para trabalhar com dados categóricos
    vs = forcats::as_factor(vs),
    am = forcats::as_factor(am),
    wt_class = forcats::as_factor(wt_class)
  )
```

Manipulação de ⊞ dados - criação/modificação

Como ficou o novo conjunto de dados?

```
#summary(mtcars_new) # sumarização de dados do rbase. Teste essa opção!
glimpse(mtcars_new)
```

```
## Rows: 32
## Columns: 14
## $ model           <chr> "Mazda RX4", "Mazda RX4 Wag", "Datsun 710", "Hornet 4 …
## $ mpg             <dbl> 21.0, 21.0, 22.8, 21.4, 18.7, 18.1, 14.3, 24.4, 22.8, …
## $ cyl             <dbl> 6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8, 8, 8, 4, …
## $ disp            <dbl> 160.0, 160.0, 108.0, 258.0, 360.0, 225.0, 360.0, 146.7…
## $ hp              <dbl> 110, 110, 93, 110, 175, 105, 245, 62, 95, 123, 123, 18…
## $ drat            <dbl> 3.90, 3.90, 3.85, 3.08, 3.15, 2.76, 3.21, 3.69, 3.92, …
## $ wt              <dbl> 2620, 2875, 2320, 3215, 3440, 3460, 3570, 3190, 3150, …
## $ qsec            <dbl> 16.46, 17.02, 18.61, 19.44, 17.02, 20.22, 15.84, 20.00…
## $ vs              <fct> V, V, S, S, V, S, V, S, S, S, S, V, V, V, V, V, V, S, …
## $ am              <fct> manual, manual, manual, automatic, automatic, automati…
## $ gear            <dbl> 4, 4, 4, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 4, …
## $ carb            <dbl> 4, 4, 1, 1, 2, 1, 4, 2, 2, 4, 4, 3, 3, 3, 4, 4, 4, 1, …
## $ power_to_weight <dbl> 0.04198473, 0.03826087, 0.04008621, 0.03421462, 0.0508…
## $ wt_class        <fct> middleweight, middleweight, lightweight, middleweight,…
```

Manipulação de dados - sumarização

A sumarização de valores é obtida com a função summarize(). Sumarizar significa usar alguma métrica (média, variância, désvio padrão, mediana, moda, mínimo, máximo, soma total, número de ocorrências, ...) para obter um único valor representativo sobre os valores de um conjunto ou subconjuntos - estes, por sua vez, podem ser obtidos por agrupamento. As métricas fornecerão informações sobre uma variável. Vejamos no seguinte exemplo:

```
mtcars_new |>
  summarize(min_hp = min(hp), mean_hp = mean(hp), max_hp = max(hp), number_of_models = n())
```

```
## # A tibble: 1 × 4
##   min_hp mean_hp max_hp number_of_models
##    <dbl>   <dbl>  <dbl>            <int>
## 1     52    147.    335               32
```

As funções min(), max(), e mean() usadas acima são do pacote **rbase**. Já a função n() é do pacote **dplyr** e retorna o número de elementos de um conjunto ou subconjuntos (grupos). Além disso, essa função pode ser usada somente dentro de summarize() e mutate().

Manipulação de dados - sumarização

Podemos fazer agrupamentos considerando valores de uma ou mais colunas usando o parâmetro .by, vide:

```
mtcars_new |>
  summarize(
    min_hp = min(hp),
    mean_hp = mean(hp),
    max_hp = max(hp),
    number_of_models = n(),
    .by = cyl
  )
```

```
## # A tibble: 3 × 5
##     cyl min_hp mean_hp max_hp number_of_models
##   <dbl>  <dbl>   <dbl>  <dbl>            <int>
## 1     6    105   122.     175                7
## 2     4     52    82.6    113               11
## 3     8    150   209.     335               14
```

Manipulação de ⊞ dados - sumarização

Podemos fazer agrupamentos considerando valores de uma ou mais colunas usando o parâmetro .by, vide:

```
mtcars_new |>
  summarize(
    min_hp = min(hp),
    mean_hp = mean(hp),
    max_hp = max(hp),
    number_of_models = n(),
    .by = c(cyl, vs, am)
  )
```

```
## # A tibble: 7 × 7
##     cyl vs    am        min_hp mean_hp max_hp number_of_models
##   <dbl> <fct> <fct>      <dbl>   <dbl>  <dbl>             <int>
## 1     6 V     manual       110   132.     175                 3
## 2     4 S     manual        52    80.6    113                 7
## 3     6 S     automatic    105   115.     123                 4
## 4     8 V     automatic    150   194.     245                12
## 5     4 S     automatic     62    84.7     97                 3
## 6     4 V     manual        91    91       91                 1
## 7     8 V     manual       264   300.     335                 2
```

Manipulação de ⊞ dados - agrupamento

A função group_by() permite agrupamentos baseados nos valores de uma ou mais colunas. Os grupos são persistidos no conjunto de dados, mas podem ser desfeitos com ungroup(). Todas as funções que vimos podem ser usadas em combinação com group_by(). Vejamos o exemplo:

```
mtcars_new |>
  group_by(wt_class) |>
  summarize(median_wt = median(wt), mean_wt = mean(wt), sd_wt = sd(wt),
            number_of_models = n())
```

```
## # A tibble: 3 × 5
##   wt_class     median_wt mean_wt sd_wt number_of_models
##   <fct>            <dbl>   <dbl> <dbl>            <int>
## 1 middleweight     3325    3228.  313.               16
## 2 lightweight      2038.   2003.  337.                8
## 3 heavyweight      3958.   4410.  777.                8
```

Obs: O parâmetro .by que vimos anteriormente é uma alternativa a group_by(), porém opera internamente em algumas funções como filter(), mutate() e summarize().

Exercícios

- Encontre a mediana, valor mínimo, média, valor máximo e desvio padrão das cilindradas de um motor (indica quanto de combustível e ar cada pistão de um motor pode deslocar em um cilindro - coluna **disp**), agrupando essas métricas por quantidade de cilindros (coluna **cyl**).
- Obtenha as mesmas métricas anteriores, porém, agrupando por quantidade de cilindros (coluna **cyl**) e a classificação segundo o peso (coluna **wt_class**).
- Encontre a mediana, valor mínimo, média, valor máximo e desvio padrão para o consumo de combustível (milhas por galão - coluna **mpg**). Teste os mesmos agrupamentos: por **cyl**, **wt_class** e ambas.

Manipulação de ⊞ dados - contagem

Podemos obter o número de elementos de um conjunto ou subconjuntos (grupos) com a função count().

```
mtcars_new |>
  count(cyl, vs, wt_class)
```

```
## # A tibble: 7 × 4
##     cyl vs    wt_class         n
##   <dbl> <fct> <fct>        <int>
## 1     4 V     lightweight      1
## 2     4 S     middleweight     3
## 3     4 S     lightweight      7
## 4     6 V     middleweight     3
## 5     6 S     middleweight     4
## 6     8 V     middleweight     6
## 7     8 V     heavyweight      8
```

Manipulação de ⊞ dados - contagem

A mesma contagem pode ser obtida combinando group_by(), summarize() e n().

```
mtcars_new |>
  group_by(cyl, vs, wt_class) |>
  summarize(number_of_model = n())
```

```
## # A tibble: 7 × 4
## # Groups:   cyl, vs [5]
##     cyl vs    wt_class     number_of_model
##   <dbl> <fct> <fct>                  <int>
## 1     4 V     lightweight                1
## 2     4 S     middleweight               3
## 3     4 S     lightweight                7
## 4     6 V     middleweight               3
## 5     6 S     middleweight               4
## 6     8 V     middleweight               6
## 7     8 V     heavyweight                8
```

Manipulação de dados

Vimos que o **dplyr** possui uma série de funções (verbos principais) mais comulmente usadas. Porém, o **dplyr** possui uma lista maior de funções que geralmente servem de forma auxiliar. Dentre estas, vimos: between(), if_else(), case_when(). Também existem funções auxiliares que funcionam internamente as funções principais como n() e os *selection helpers* – starts_with(), ends_with(), contains().

Não se preocupe! Tanto as funções principais, como as auxiliares, vão aos poucos fazendo parte de nosso vocabulário conforme nossas necessidades de operar sobre dados aparecem. Não obstante, não é necessário lembrar de tudo, já que o material de consulta é vasto.

Metotologias Informacionais com 

Muito Obrigado pela Atenção!