Lidando com dados · Uma abordagem baseada no {tidyverse}

Prof. Dr. Wagner Hugo Bonat

Estrutura e objetivos do módulo

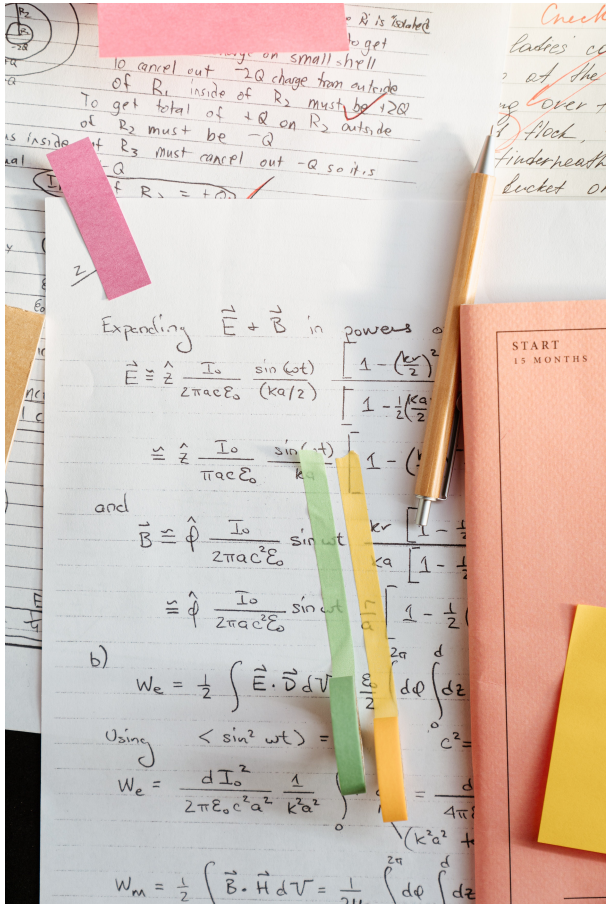
Estrutura do módulo

- Framework {tidyverse}.
- Os princípios do {tidyverse}.
- Etapas do processo de análise de dados.
  - Importação de dados.
  - Arrumação de dados.
  - Transformação e manipulação de dados.
  - Combinação de dados.
  - Exportação de dados.
- Projeto prático.

https://www.pexels.com/photo/top-view-of-people-at-the-meeting-3184287/

Objetivos do módulo

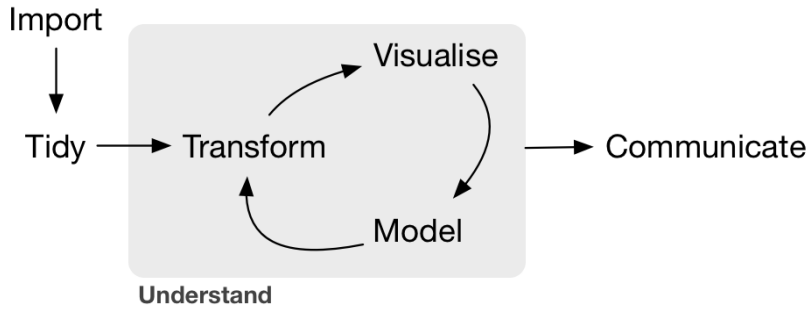
- Importar dados de diferentes fontes.
- Compreender as principais técnicas de arrumação de dados.
- Dominar os principais verbos para manipulação de dados.
- Compreender as diferentes formas de combinação de dados.
- Exportar dados em diferentes formatos.
- Dominar a gramática da manipulação de dados.



Lidando com dados

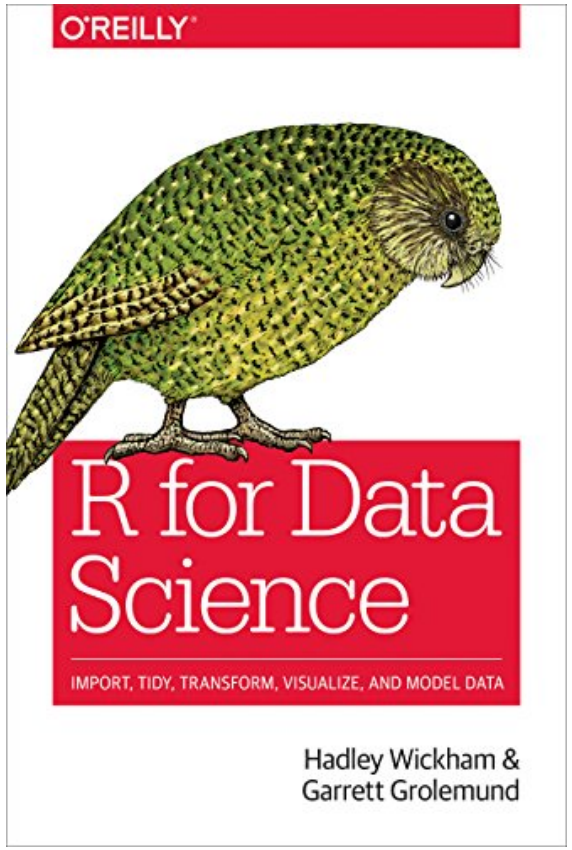
Manipulação e visualização de dados

- Manipular e visualizar dados (MVD) são atividades **obrigatórias** em qualquer atividade científica.
- A MVD **determina o sucesso** de uma série de etapas.
  - Entendimento dos dados.
  - Limpeza e conciliação de dados.
  - Engenharia de características.
  - Especificação de modelos.
  - Comunicação de resultados, etc.
- Fazer MVD de forma **eficiente** requer:
  - Conhecer o processo e suas etapas.
  - Dominar a tecnologia para execução.
- **Linguagens de programação** oferecem uma série de vantagens: reproduzível, extensível, escalonável, integrável, portável, etc.

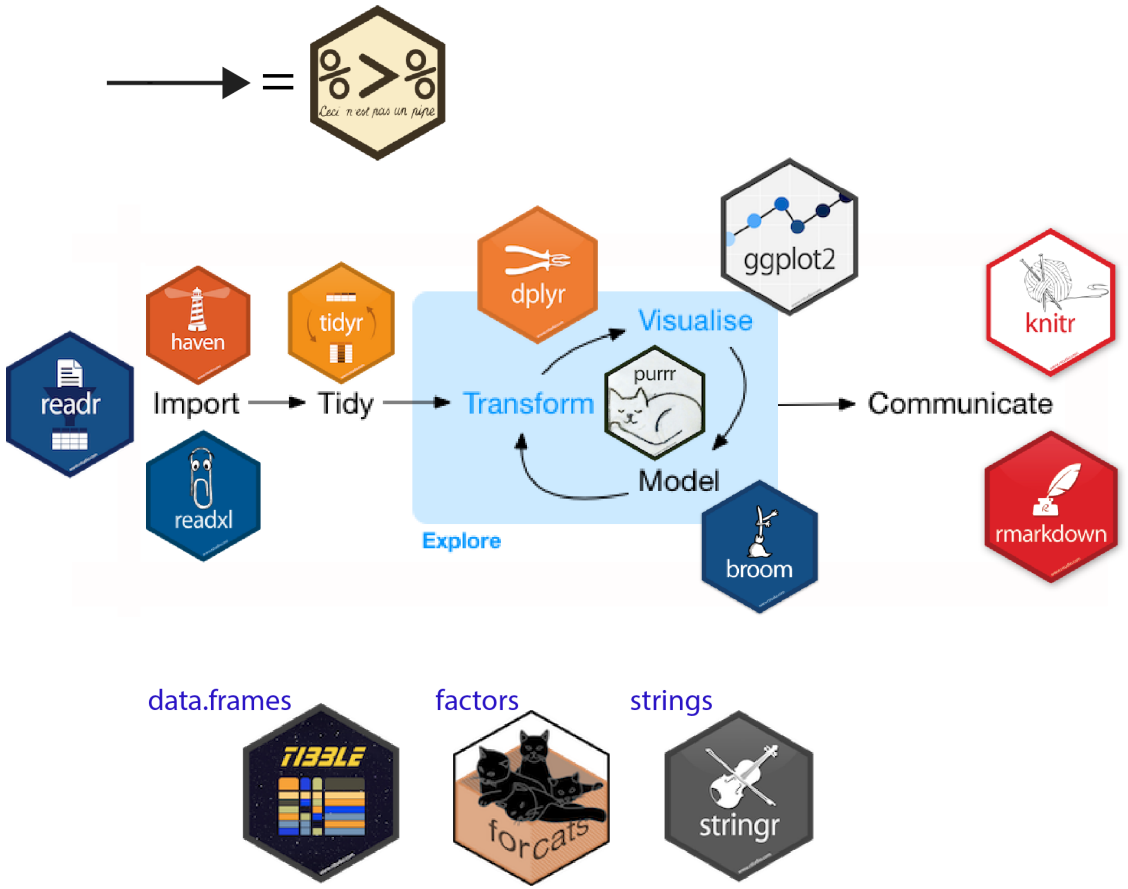


Ciclo de vida da ciência de dados. Fonte da imagem: https://bookdown.org/fjmcgrade/ismaykim/#intro-for-students

R for Data Science



R for Data Science, a principal referência sobre o emprego da linguagem R em ciência de dados.



Workflow de ciência de dados com o {tidyverse}. Fonte: https://oliviergimenez.github.io/intro_tidyverse/#7

O framework {tidyverse}

O {tidyverse}

- Oferece uma **reimplementação e extensão** das funcionalidades do `R` para manipulação e visualização de dados.
- É uma coleção de **8 pacotes** que operam em harmonia.
- Foram planejados e construídos para trabalhar em conjunto.
- Possuem gramática, organização, filosofia e estruturas de dados mais claras.
- Maior facilidade de desenvolvimento de código e portabilidade.
- Outros pacotes acoplam muito bem com o {tidyverse}.
- Pacotes: https://www.tidyverse.org/packages/.
- **R4DS**: https://r4ds.had.co.nz/.
- Cookbook: https://rstudio-education.github.io/tidyverse-cookbook/program.html.

```
library(tidyverse)
tidyverse_packages()
```

```
##  [1] "broom"          "cli"
##  [3] "crayon"         "dbplyr"
##  [5] "dplyr"          "dtplyr"
##  [7] "forcats"        "googledrive"
##  [9] "googlesheets4"  "ggplot2"
## [11] "haven"          "hms"
## [13] "httr"           "jsonlite"
## [15] "lubridate"      "magrittr"
## [17] "modelr"         "pillar"
## [19] "purrr"          "readr"
## [21] "readxl"         "reprex"
## [23] "rlang"          "rstudioapi"
## [25] "rvest"          "stringr"
## [27] "tibble"         "tidyr"
## [29] "xml2"           "tidyverse"
```

Os pacotes do {tidyverse}

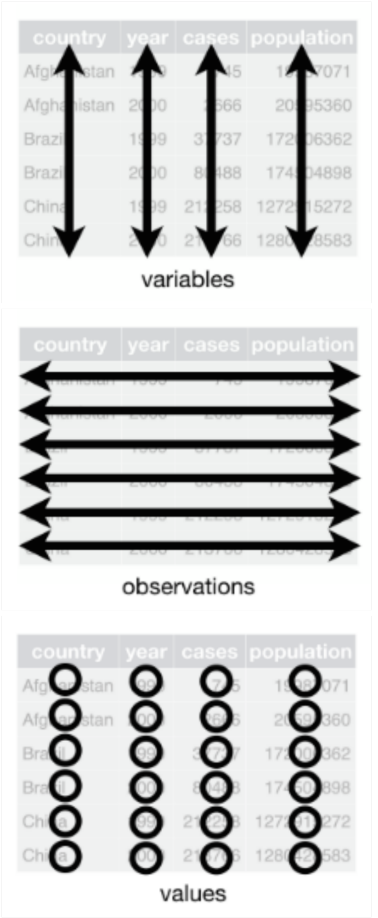
Pacotes que fazem parte do {tidyverse}.

Princípios dos dados organizados

- Cada variável está em uma coluna.
- Cada observação está em uma linha.
- Cada tipo de unidade observacional está em uma célula.

Tarefas comuns ao lidar com dados

- Importação.
- Arrumação.
- Manipulação.
- Combinação.
- Exportação.



Representação dos três princípios do tidy data.

Importação de dados

O pacote {readr}

- O processo de análise de dados começa com a importação dos dados para o ambiente de manipulação.
- Existem vários meios para armazenar dados.
  - Arquivos de texto pleno (tsv, txt, csv, etc).
  - Planilhas eletrônicas.
  - Bancos de dados relacionais.
  - Etc.
- O `readr` tem recursos para importação de dados retangulares na forma de texto pleno.
- Documentação:
  - https://readr.tidyverse.org/.
  - https://r4ds.had.co.nz/data-import.html.
  - https://cran.r-project.org/package=readr.

Importando arquivos de texto pleno

- Importando dados do tipo .txt.

```
library(readr)
url <- "http://leg.ufpr.br/~wagner/scientificR/anovareg.txt"
dados <- read_tsv(url, col_names = TRUE)
```

```
## Rows: 72 Columns: 4
## ── Column specification ────────────────────
## Delimiter: "\t"
## chr (2): cultivar, bloco
## dbl (2): dose, indice
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(dados)
```

```
## # A tibble: 6 × 4
##   cultivar  dose bloco indice
##   <chr>    <dbl> <chr>  <dbl>
## 1 Ag-1002      0 I         46
## 2 Ag-1002      0 II        48
## 3 Ag-1002      0 III       44
## 4 Ag-1002      0 IV        46
## 5 Ag-1002     60 I         48
## 6 Ag-1002     60 II        47
```

Importando arquivos de texto pleno

- Importando dados do tipo .csv.

```
library(readr)
url <- "http://leg.ufpr.br/~wagner/scientificR/reglinear.csv"
dados <- read_table(url, col_names = TRUE)
```

```
##
## ── Column specification ────────────────────
## cols(
##   `"y"` = col_double(),
##   `"x"` = col_double()
## )
```

```
head(dados)
```

```
## # A tibble: 6 × 2
##     `"y"` `"x"`
##     <dbl> <dbl>
## 1 207318.    55
## 2 250846.    69
## 3 165755.    46
## 4 219817.    61
## 5 268582.    73
## 6 229060.    63
```

Importando planilhas eletrônicas

- Importando uma planilha eletrônica

```
library(readxl)
library(httr)
url <- "http://leg.ufpr.br/~wagner/scientificR/meus_dados.xlsx"
GET(url, write_disk(tf <- tempfile(fileext = ".xlsx")))
```

```
## Response [http://leg.ufpr.br/~wagner/scientificR/meus_dados.xlsx]
##   Date: 2023-03-09 12:18
##   Status: 200
##   Content-Type: application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
##   Size: 10.7 kB
## <ON DISK>  /tmp/RtmpXjPWSU/file1aa439eb5dc5.xlsx
```

```
tb <- read_excel(tf, sheet = "mtcars")
head(tb[,1:4])
```

```
## # A tibble: 6 × 4
##     mpg   cyl  disp    hp
##   <dbl> <dbl> <dbl> <dbl>
## 1  21       6   160   110
## 2  21       6   160   110
## 3  22.8     4   108    93
## 4  21.4     6   258   110
## 5  18.7     8   360   175
## 6  18.1     6   225   105
```

Conexão com bancos de dados relacionais

- Conectando e importando tabelas de bancos relacionais - `MySQL`.

```
library(DBI)
library(RMySQL)

# Criando a conexão.
db <- dbConnect(
  RMySQL::MySQL(),
  user = "rfamro", password = "",
  port = 4497, dbname = "Rfam",
  host = "mysql-rfam-public.ebi.ac.uk")

# Lista as tabelas do BD.
dbListTables(db)

# Listas as colunas em uma tabela.
dbListFields(db, "keywords")

# Importanto a tabela.
tb <- RMySQL::dbFetch(
  RMySQL::dbSendQuery(
    db, "SELECT * FROM keywords"))
str(tb)
# Desconecta
dbDisconnect(db)
```

Cartão de referência de importação de dados com o {readr}

Clique no texto para abrir o arquivo

Arrumação de dados

Arrumando dados · Situações típicas

- Variáveis nas colunas.

```
tb1 <- data.frame("city" =
                    c("C1", "C2", "C3"),
                  '2011' = c(5, 7, 3),
                  '2012' = c(6, 2, 5),
                  '2013' = c(6, 9, 7),
                  check.names = FALSE)

tb1
```

```
##   city 2011 2012 2013
## 1   C1    5    6    6
## 2   C2    7    2    9
## 3   C3    3    5    7
```

- Versão longa.

```
tb1_long <- tb1 %>%
  pivot_longer(names_to = 'ano',
               values_to = 'resposta',
               cols = -city)

tb1_long
```

```
## # A tibble: 9 × 3
##   city  ano   resposta
##   <chr> <chr>    <dbl>
## 1 C1    2011         5
## 2 C1    2012         6
## 3 C1    2013         6
## 4 C2    2011         7
## 5 C2    2012         2
## 6 C2    2013         9
## 7 C3    2011         3
## 8 C3    2012         5
## 9 C3    2013         7
```

Arrumando dados · Situações típicas

- Pode ser necessário variáveis nas colunas.

```
tb1_long %>%
  pivot_wider(names_from =  'ano',
              values_from = 'resposta')
```

```
## # A tibble: 3 × 4
##   city  `2011` `2012` `2013`
##   <chr>  <dbl>  <dbl>  <dbl>
## 1 C1         5      6      6
## 2 C2         7      2      9
## 3 C3         3      5      7
```

Arrumando dados · Situações típicas

- Separando variáveis.

```
tb <- data.frame(US = c("US1", "US2", "US3"),
                 cidade_ano = c("Curitiba/2012", "Santos/2012", "Viçosa/2016"),
                 local = c("Curitiba-PR", "Santos-SP", "Viçosa-MG"))

tb
```

```
##    US    cidade_ano       local
## 1 US1 Curitiba/2012 Curitiba-PR
## 2 US2   Santos/2012   Santos-SP
## 3 US3   Viçosa/2016   Viçosa-MG
```

```
tb_nova1 <- tb %>% separate(col = cidade_ano,
                            into = c('Cidade', 'Ano'),
                            sep = '/')
tb_nova1
```

```
##    US   Cidade  Ano       local
## 1 US1 Curitiba 2012 Curitiba-PR
## 2 US2   Santos 2012   Santos-SP
## 3 US3   Viçosa 2016   Viçosa-MG
```

- Exercício: Separe a coluna `local` em duas novas colunas.

Arrumando dados · Situações típicas

- Unindo variáveis.

```
tb <- data.frame(dia = c(1, 5, 23, 16),
                 mes = c(3, 6, 2, 9),
                 ano = 2018)
tb
```

```
##   dia mes  ano
## 1   1   3 2018
## 2   5   6 2018
## 3  23   2 2018
## 4  16   9 2018
```

```
tb <- tb %>% unite(col = 'data',
                   sep = '/',
                   c('dia', 'mes', 'ano'),
                   remove = FALSE)

tb
```

```
##        data dia mes  ano
## 1  1/3/2018   1   3 2018
## 2  5/6/2018   5   6 2018
## 3 23/2/2018  23   2 2018
## 4 16/9/2018  16   9 2018
```

Arrumando dados · Situações típicas

- Dados faltantes.

```
tb <- data.frame(Paciente = 1:5,
                 N_con = c(0, 1, 3, 1, 2),
                 N_trat = c(NA, 0, 0, 2, 1),
                 N_rem = c(NA, 1, 1, 0, 0))
tb
```

```
##   Paciente N_con N_trat N_rem
## 1        1     0     NA    NA
## 2        2     1      0     1
## 3        3     3      0     1
## 4        4     1      2     0
## 5        5     2      1     0
```

- Substituindo NA por 0.

```
# drop_na(tb) Remove todas as linhas com NA
tb %>% replace_na(list(N_trat = 0,
                       N_rem = 0))
```

```
##   Paciente N_con N_trat N_rem
## 1        1     0      0     0
## 2        2     1      0     1
## 3        3     3      0     1
## 4        4     1      2     0
## 5        5     2      1     0
```

Cartão de referência de arrumação de dados com o {tidyr}

Clique no texto para abrir o arquivo

Manipulação de dados

O pacote {dplyr}

- Depois dos dados arrumados, é a hora de começar a conhecê-los!
- Começa a fase da **análise exploratória de dados** (AED).
- Os dados são explorados para:
  - Conhecer as propriedades das variáveis.
  - Determinar medidas descritivas.
  - Comparar grupos.
  - Quantificar relações entre variáveis.
  - Extrair padrões.
  - Detectar erros e corrigir problemas.
  - AED envolve inúmeras operações.
  - É preciso conhecê-las e ser criativo para aplicar da melhor forma.

Detalhes do `dplyr`

- O `dplyr` é a **gramática** para manipulação de dados.
- Tem um conjunto **consistente** de verbos para atuar sobre tabelas.
  - Verbos: `arrange()`, `select()`, `mutate()`, `slice()`, `rename()`, `filter()`, `summarise()`, etc.
  - Sufixos: `_at()`, `_if()`, `_all()`, etc.
  - Agrupamento: `group_by()` e `ungroup()`.
  - Junções: `inner_join()`, `full_join()`, `left_join()` e `right_join()`.
  - Funções resumo: `n()`, `n_distinct()`, `first()`, `last()`, `nth()`, etc.
  - E muito mais no cartão de referência: https://github.com/rstudio/cheatsheets/raw/master/data-transformation.pdf.
- Documentação:
  - https://dplyr.tidyverse.org/.
  - https://r4ds.had.co.nz/relational-data.html.
  - https://cran.r-project.org/package=dplyr

Criação de um `tibble`

Criação por colunas

```
library(tidyverse)

# Tabela com alunos do curso de
# Matemática e de Estatística.
df1 <- tibble(
    matricula = c(256, 487, 965,
                  125, 458, 874, 963),
    nome = c("João", "Vanessa", "Tiago",
             "Luana", "Gisele", "Pedro",
             "André"),
    curso = c("Mat", "Mat", "Est", "Est",
              "Est", "Mat", "Est"),
    prova1 = c(80, 75, 95, 70, 45, 55, 30),
    prova2 = c(90, 75, 80, 85, 50, 75, NA),
    prova3 = c(80, 75, 75, 50, NA, 90, 30),
    faltas = c(4, 4, 0, 8, 16, 0, 20))
```

```
df1
```

```
## # A tibble: 7 × 7
##   matricula nome    curso prova1 prova2
##       <dbl> <chr>   <chr>  <dbl>  <dbl>
## 1       256 João    Mat       80     90
## 2       487 Vanessa Mat       75     75
## 3       965 Tiago   Est       95     80
## 4       125 Luana   Est       70     85
## 5       458 Gisele  Est       45     50
## 6       874 Pedro   Mat       55     75
## 7       963 André   Est       30     NA
## # … with 2 more variables:
## #   prova3 <dbl>, faltas <dbl>
```

Criação de um `tibble`

Criação por linhas

```
# Informações de cadastro dos alunos
# em outra base de dados.
df_extra <- tribble(
    ~mat,     ~nome, ~idade, ~bolsista,
     256,  'João'  ,     18,        "S",
     965,  'Tiago' ,     18,        "N",
     285,  'Tiago' ,     22,        "N",
     125,  'Luana' ,     21,        "S",
     874,  'Pedro' ,     19,        "N",
     321,  'Mia'   ,     18,        "N",
     669,  'Luana' ,     19,        "S",
     967,  'André' ,     20,        "N",
)
```

```
df_extra
```

```
## # A tibble: 8 × 4
##     mat nome  idade bolsista
##   <dbl> <chr> <dbl> <chr>
## 1   256 João     18 S
## 2   965 Tiago    18 N
## 3   285 Tiago    22 N
## 4   125 Luana    21 S
## 5   874 Pedro    19 N
## 6   321 Mia      18 N
## 7   669 Luana    19 S
## 8   967 André    20 N
```

Ordenação

Ordenação por uma variável

```
df1 %>% arrange(prova1)
```

```
## # A tibble: 7 × 7
##   matricula nome    curso prova1 prova2
##       <dbl> <chr>   <chr>  <dbl>  <dbl>
## 1       963 André   Est       30     NA
## 2       458 Gisele  Est       45     50
## 3       874 Pedro   Mat       55     75
## 4       125 Luana   Est       70     85
## 5       487 Vanessa Mat       75     75
## 6       256 João    Mat       80     90
## 7       965 Tiago   Est       95     80
## # … with 2 more variables:
## #   prova3 <dbl>, faltas <dbl>
```

Ordenação por duas variáveis

```
df1 %>%
  arrange(prova1, desc(faltas)) %>%
  select(nome, prova1, faltas)
```

```
## # A tibble: 7 × 3
##   nome    prova1 faltas
##   <chr>    <dbl>  <dbl>
## 1 André       30     20
## 2 Gisele      45     16
## 3 Pedro       55      0
## 4 Luana       70      8
## 5 Vanessa     75      4
## 6 João        80      4
## 7 Tiago       95      0
```

Seleção

Seleção pelo nome das colunas

```
df1 %>%
  select("nome", "prova1", "faltas")
```

```
## # A tibble: 7 × 3
##   nome    prova1 faltas
##   <chr>    <dbl>  <dbl>
## 1 João        80      4
## 2 Vanessa     75      4
## 3 Tiago       95      0
## 4 Luana       70      8
## 5 Gisele      45     16
## 6 Pedro       55      0
## 7 André       30     20
```

Seleção pela posição das colunas

```
df1 %>%
  select(c(2, 4, 7))
```

```
## # A tibble: 7 × 3
##   nome    prova1 faltas
##   <chr>    <dbl>  <dbl>
## 1 João        80      4
## 2 Vanessa     75      4
## 3 Tiago       95      0
## 4 Luana       70      8
## 5 Gisele      45     16
## 6 Pedro       55      0
## 7 André       30     20
```

Seleção por fatias

Fatiando pelo número das linhas

```
df1 %>%
  slice(3:5)
```

```
## # A tibble: 3 × 7
##   matricula nome   curso prova1 prova2
##       <dbl> <chr>  <chr>  <dbl>  <dbl>
## 1       965 Tiago  Est       95     80
## 2       125 Luana  Est       70     85
## 3       458 Gisele Est       45     50
## # … with 2 more variables:
## #   prova3 <dbl>, faltas <dbl>
```

Cabeça (head) e cauda (tail)

```
df1 %>% slice_head(n = 3)
```

```
## # A tibble: 3 × 7
##   matricula nome    curso prova1 prova2
##       <dbl> <chr>   <chr>  <dbl>  <dbl>
## 1       256 João    Mat       80     90
## 2       487 Vanessa Mat       75     75
## 3       965 Tiago   Est       95     80
## # … with 2 more variables:
## #   prova3 <dbl>, faltas <dbl>
```

```
df1 %>% slice_tail(n = 3)
```

```
## # A tibble: 3 × 7
##   matricula nome   curso prova1 prova2
##       <dbl> <chr>  <chr>  <dbl>  <dbl>
## 1       458 Gisele Est       45     50
## 2       874 Pedro  Mat       55     75
## 3       963 André  Est       30     NA
## # … with 2 more variables:
## #   prova3 <dbl>, faltas <dbl>
```

Seleção das variáveis

Seleção de variáveis por condição

```
df1 %>%
    select_if(is.numeric)
```

```
## # A tibble: 7 × 5
##   matricula prova1 prova2 prova3 faltas
##       <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       256     80     90     80      4
## 2       487     75     75     75      4
## 3       965     95     80     75      0
## 4       125     70     85     50      8
## 5       458     45     50     NA     16
## 6       874     55     75     90      0
## 7       963     30     NA     30     20
```

```
df1 %>%
    select_if(negate(is.numeric))
```

Filtros

Filtro por valor de uma variável

```
df1 %>% filter(prova1 > 90 )
```

```
## # A tibble: 1 × 7
##   matricula nome  curso prova1 prova2
##       <dbl> <chr> <chr>  <dbl>  <dbl>
## 1       965 Tiago Est       95     80
## # … with 2 more variables:
## #   prova3 <dbl>, faltas <dbl>
```

Filtro pelo valor de duas ou mais

```
df1 %>%
  filter((prova1 + prova2 + prova3)/3 > 80)
```

```
## # A tibble: 2 × 7
##   matricula nome  curso prova1 prova2
##       <dbl> <chr> <chr>  <dbl>  <dbl>
## 1       256 João  Mat       80     90
## 2       965 Tiago Est       95     80
## # … with 2 more variables:
## #   prova3 <dbl>, faltas <dbl>
```

Valor da unidade observacional

```
df1 %>% filter(curso == "Mat")
```

```
## # A tibble: 3 × 7
##   matricula nome    curso prova1 prova2
##       <dbl> <chr>   <chr>  <dbl>  <dbl>
## 1       256 João    Mat       80     90
## 2       487 Vanessa Mat       75     75
## 3       874 Pedro   Mat       55     75
## # … with 2 more variables:
## #   prova3 <dbl>, faltas <dbl>
```

Combinando filtros

```
df1 %>% filter(curso == "Est" & prova2 > 80)
```

```
## # A tibble: 1 × 7
##   matricula nome  curso prova1 prova2
##       <dbl> <chr> <chr>  <dbl>  <dbl>
## 1       125 Luana Est       70     85
## # … with 2 more variables:
## #   prova3 <dbl>, faltas <dbl>
```

Filtros

Apenas observações com NA

```
df1 %>% filter(is.na(prova2))
```

```
## # A tibble: 1 × 7
##   matricula nome  curso prova1 prova2
##       <dbl> <chr> <chr>  <dbl>  <dbl>
## 1       963 André Est       30     NA
## # … with 2 more variables:
## #   prova3 <dbl>, faltas <dbl>
```

Por grupos

```
df1 %>% filter(nome %in% c("João", "Pedro"))
```

```
## # A tibble: 2 × 7
##   matricula nome  curso prova1 prova2
##       <dbl> <chr> <chr>  <dbl>  <dbl>
## 1       256 João  Mat       80     90
## 2       874 Pedro Mat       55     75
## # … with 2 more variables:
## #   prova3 <dbl>, faltas <dbl>
```

Observações sem NAs em determinadas colunas

```
df1 %>%
  filter(!is.na(prova2) & !is.na(prova3))
```

```
## # A tibble: 5 × 7
##   matricula nome    curso prova1 prova2
##       <dbl> <chr>   <chr>  <dbl>  <dbl>
## 1       256 João    Mat       80     90
## 2       487 Vanessa Mat       75     75
## 3       965 Tiago   Est       95     80
## 4       125 Luana   Est       70     85
## 5       874 Pedro   Mat       55     75
## # … with 2 more variables:
## #   prova3 <dbl>, faltas <dbl>
```

Renomear

```
df1 <- df1 %>% rename(mat = 'matricula',
                      nome = 'nome',
                      curso = 'curso',
                      p1 = 'prova1',
                      p2 = 'prova2',
                      p3 = 'prova3',
                      fal = 'faltas')
```

```
df1
```

```
## # A tibble: 7 × 7
##     mat nome    curso    p1    p2    p3
##   <dbl> <chr>   <chr> <dbl> <dbl> <dbl>
## 1   256 João    Mat      80    90    80
## 2   487 Vanessa Mat      75    75    75
## 3   965 Tiago   Est      95    80    75
## 4   125 Luana   Est      70    85    50
## 5   458 Gisele  Est      45    50    NA
## 6   874 Pedro   Mat      55    75    90
## 7   963 André   Est      30    NA    30
## # … with 1 more variable: fal <dbl>
```

Realocação

Realocação pelos nomes

```
df1 %>%
    relocate(p1:p3, fal)
```

```
## # A tibble: 7 × 7
##      p1    p2    p3   fal   mat nome
##   <dbl> <dbl> <dbl> <dbl> <dbl> <chr>
## 1    80    90    80     4   256 João
## 2    75    75    75     4   487 Vanessa
## 3    95    80    75     0   965 Tiago
## 4    70    85    50     8   125 Luana
## 5    45    50    NA    16   458 Gisele
## 6    55    75    90     0   874 Pedro
## 7    30    NA    30    20   963 André
## # … with 1 more variable: curso <chr>
```

```
df1 %>%
    relocate(mat, nome,
             .after = last_col())
```

```
## # A tibble: 7 × 7
##   curso    p1    p2    p3   fal   mat
##   <chr> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Mat      80    90    80     4   256
## 2 Mat      75    75    75     4   487
## 3 Est      95    80    75     0   965
## 4 Est      70    85    50     8   125
## 5 Est      45    50    NA    16   458
## 6 Mat      55    75    90     0   874
## 7 Est      30    NA    30    20   963
## # … with 1 more variable: nome <chr>
```

Transformação

Criando uma nova variável

```
df1 <- df1 %>%
  replace_na(list(p1 = 0, p2 = 0, p3 = 0))
df1 <- df1 %>%
  mutate(media = (p1 + p2 + p3)/3)
df1
```

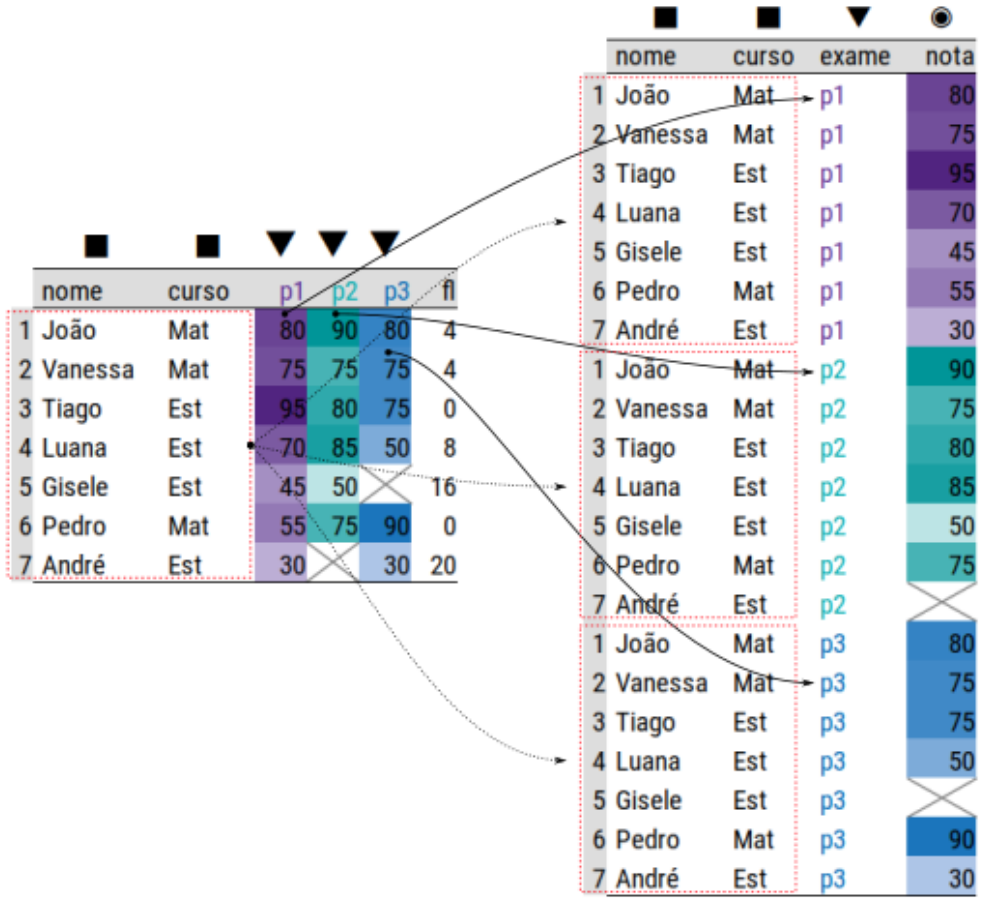
```
## # A tibble: 7 × 8
##     mat nome    curso    p1    p2    p3
##   <dbl> <chr>   <chr> <dbl> <dbl> <dbl>
## 1   256 João    Mat      80    90    80
## 2   487 Vanessa Mat      75    75    75
## 3   965 Tiago   Est      95    80    75
## 4   125 Luana   Est      70    85    50
## 5   458 Gisele  Est      45    50     0
## 6   874 Pedro   Mat      55    75    90
## 7   963 André   Est      30     0    30
## # … with 2 more variables: fal <dbl>,
## #   media <dbl>
```

Classificando as notas

```
breaks <- c(0, 40, 70, 100)
df1$classificacao <- cut(df1$media,
                         breaks = breaks,
                         labels = c("Baixa",
                                    "Média",
                                    "Alta"))
df1 %>% select(media, classificacao)
```

```
## # A tibble: 7 × 2
##   media classificacao
##   <dbl> <fct>
## 1  83.3 Alta
## 2  75   Alta
## 3  83.3 Alta
## 4  68.3 Média
## 5  31.7 Baixa
## 6  73.3 Alta
## 7  20   Baixa
```

Resumo e sumarização



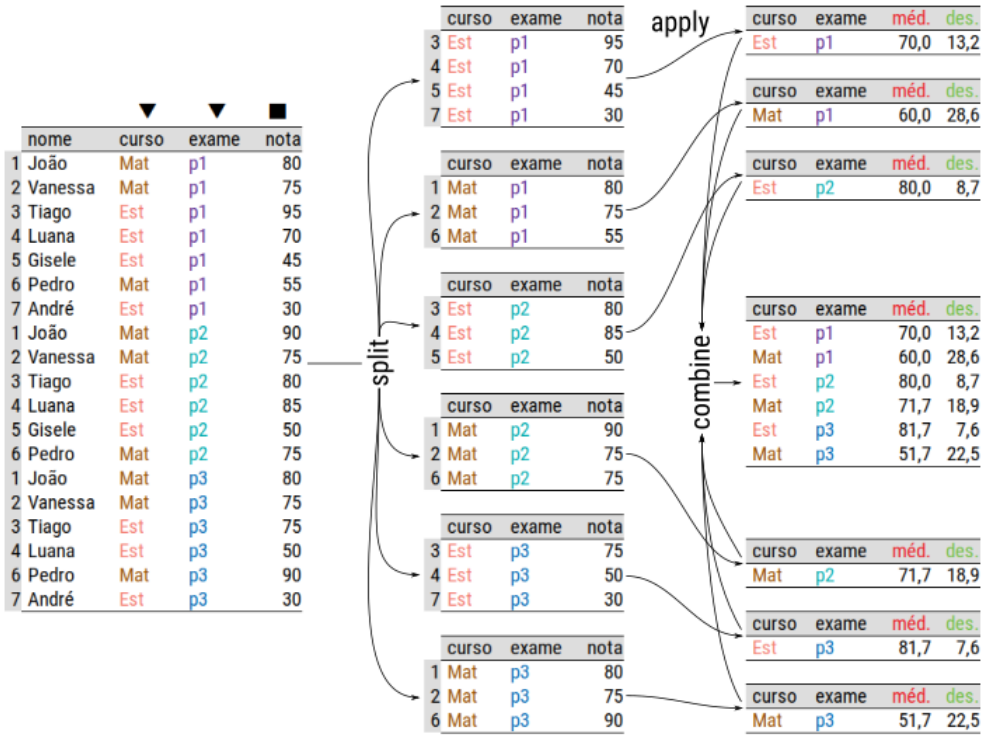
Empilhamento.

Resumo e sumarização

```
df1_temp <- df1 %>% select(-c(mat, media, classificacao, fal))
df1_long <- df1_temp %>% pivot_longer(names_to = 'prova',
                                      values_to = 'valor',
                                      cols = -c(nome, curso))
df1_long
```

```
## # A tibble: 21 × 4
##    nome    curso prova valor
##    <chr>   <chr> <chr> <dbl>
##  1 João    Mat   p1       80
##  2 João    Mat   p2       90
##  3 João    Mat   p3       80
##  4 Vanessa Mat   p1       75
##  5 Vanessa Mat   p2       75
##  6 Vanessa Mat   p3       75
##  7 Tiago   Est   p1       95
##  8 Tiago   Est   p2       80
##  9 Tiago   Est   p3       75
## 10 Luana   Est   p1       70
## # … with 11 more rows
```

Resumo e sumarização



	nome	curso	exame	nota
1	João	Mat	p1	80
2	Vanessa	Mat	p1	75
3	Tiago	Est	p1	95
4	Luana	Est	p1	70
5	Gisele	Est	p1	45
6	Pedro	Mat	p1	55
7	André	Est	p1	30
1	João	Mat	p2	90
2	Vanessa	Mat	p2	75
3	Tiago	Est	p2	80
4	Luana	Est	p2	85
5	Gisele	Est	p2	50
6	Pedro	Mat	p2	75
1	João	Mat	p3	80
2	Vanessa	Mat	p3	75
3	Tiago	Est	p3	75
4	Luana	Est	p3	50
6	Pedro	Mat	p3	90
7	André	Est	p3	30

	curso	exame	nota
3	Est	p1	95
4	Est	p1	70
5	Est	p1	45
7	Est	p1	30

	curso	exame	nota
1	Mat	p1	80
2	Mat	p1	75
6	Mat	p1	55

	curso	exame	nota
3	Est	p2	80
4	Est	p2	85
5	Est	p2	50

	curso	exame	nota
1	Mat	p2	90
2	Mat	p2	75
6	Mat	p2	75

	curso	exame	nota
3	Est	p3	75
4	Est	p3	50
7	Est	p3	30

	curso	exame	nota
1	Mat	p3	80
2	Mat	p3	75
6	Mat	p3	90

curso	exame	méd.	des.
Est	p1	70,0	13,2

curso	exame	méd.	des.
Mat	p1	60,0	28,6

curso	exame	méd.	des.
Est	p2	80,0	8,7

curso	exame	méd.	des.
Est	p1	70,0	13,2
Mat	p1	60,0	28,6
Est	p2	80,0	8,7
Mat	p2	71,7	18,9
Est	p3	81,7	7,6
Mat	p3	51,7	22,5

curso	exame	méd.	des.
Mat	p2	71,7	18,9

curso	exame	méd.	des.
Est	p3	81,7	7,6

curso	exame	méd.	des.
Mat	p3	51,7	22,5

Empilhamento.

Resumo e sumarização

Agrupando por uma variável

```
df1_long %>%
    group_by(curso) %>%
    summarise(media = mean(valor))
```

```
## # A tibble: 2 × 2
##   curso media
##   <chr> <dbl>
## 1 Est    50.8
## 2 Mat    77.2
```

Agrupando por mais de uma variável

```
df1_long %>%
    group_by(curso, prova) %>%
    summarise(media = mean(valor))
```

```
## `summarise()` has grouped output by
## 'curso'. You can override using the
## `.groups` argument.

## # A tibble: 6 × 3
## # Groups:   curso [2]
##   curso prova media
##   <chr> <chr> <dbl>
## 1 Est   p1     60
## 2 Est   p2     53.8
## 3 Est   p3     38.8
## 4 Mat   p1     70
## 5 Mat   p2     80
## 6 Mat   p3     81.7
```

Resumo e sumarização

Calculando mais que uma estatística

```
df1_long %>%
    group_by(curso) %>%
    summarise(media = mean(valor),
              desvio_padrao = sd(valor))
```

```
## # A tibble: 2 × 3
##   curso media desvio_padrao
##   <chr> <dbl>         <dbl>
## 1 Est    50.8          31.6
## 2 Mat    77.2          10.3
```

Combinações de estatísticas

```
df1_long %>%
    group_by(curso) %>%
    summarise(media = mean(valor),
              desvio = sd(valor),
              CV = desvio/media)
```

```
## # A tibble: 2 × 4
##   curso media desvio    CV
##   <chr> <dbl>  <dbl> <dbl>
## 1 Est    50.8   31.6 0.622
## 2 Mat    77.2   10.3 0.134
```

Resumo e sumarização

Tabela de frequências

```
df1 %>%
    group_by(curso) %>%
    summarise("N_alunos" = n())
```

```
## # A tibble: 2 × 2
##   curso N_alunos
##   <chr>    <int>
## 1 Est          4
## 2 Mat          3
```

Outra opção

```
df1 %>%
  count(curso)
```

```
## # A tibble: 2 × 2
##   curso     n
##   <chr> <int>
## 1 Est       4
## 2 Mat       3
```

Frequência por grupos

```
df1 %>%
  group_by(curso) %>%
  count(classificacao)
```

```
## # A tibble: 4 × 3
## # Groups:   curso [2]
##   curso classificacao     n
##   <chr> <fct>         <int>
## 1 Est   Baixa             2
## 2 Est   Média             1
## 3 Est   Alta              1
## 4 Mat   Alta              3
```

Cartão de referência de manipulação de dados com o {dplyr}

Clique no texto para abrir o arquivo

Combinação de dados

Concatenação

- A concatenação permite adicionar novas observações a uma tabela ou novas variáveis.
- Seja por linha ou colunas, entradas com `NA` são criadas para os índices que não foram especificados.

	mat.	nome	p1	p2	p3	fl
1	256	João	80	90	80	4
2	487	Vanessa	75	75	75	4
3	965	Tiago	95	80	75	0
4	125	Luana	70	85	50	8
5	458	Gisele	45	50		16
6	874	Pedro	55	75	90	0
7	963	André	30		30	20

	mat.	nome	p1	p2	fl
1	505	Bia	65	85	0
2	658	Carlos	75	80	2
3	713	Cris	75	90	2

	mat.	nome	p1	p2	p3	fl
1	256	João	80	90	80	4
2	487	Vanessa	75	75	75	4
3	965	Tiago	95	80	75	0
4	125	Luana	70	85	50	8
5	458	Gisele	45	50		16
6	874	Pedro	55	75	90	0
7	963	André	30		30	20
8	505	Bia	65	85		0
9	658	Carlos	75	80		2
10	713	Cris	75	90		2

Concatenação de duas tabelas.

Concatenação

De linhas (vertical)

```
# Concatenação na vertical (pilha).
bind_rows(df1[1:3, c(1, 3, 5)],
          df1[5:7, c(1, 3, 5, 4)],
          df1[4,   c(1,    5, 4)])
```

```
## # A tibble: 7 × 4
##     mat curso    p2    p1
##   <dbl> <chr> <dbl> <dbl>
## 1   256 Mat      90    NA
## 2   487 Mat      75    NA
## 3   965 Est      80    NA
## 4   458 Est      50    45
## 5   874 Mat      75    55
## 6   963 Est       0    30
## 7   125 <NA>     85    70
```

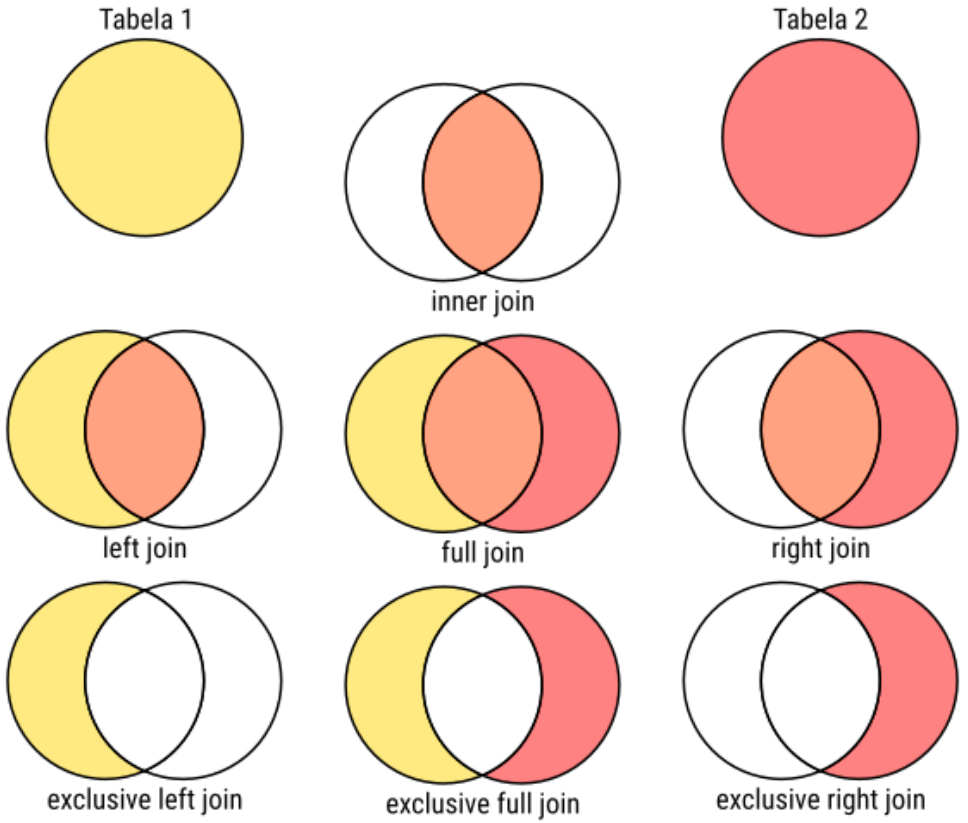
De colunas (horizontal)

```
# Concatenação na horizontal (fila).
bind_cols(df1[, c(1:3)],
          df1[, c(6:7)])
```

```
## # A tibble: 7 × 5
##     mat nome    curso    p3   fal
##   <dbl> <chr>   <chr> <dbl> <dbl>
## 1   256 João    Mat      80     4
## 2   487 Vanessa Mat      75     4
## 3   965 Tiago   Est      75     0
## 4   125 Luana   Est      50     8
## 5   458 Gisele  Est       0    16
## 6   874 Pedro   Mat      90     0
## 7   963 André   Est      30    20
```

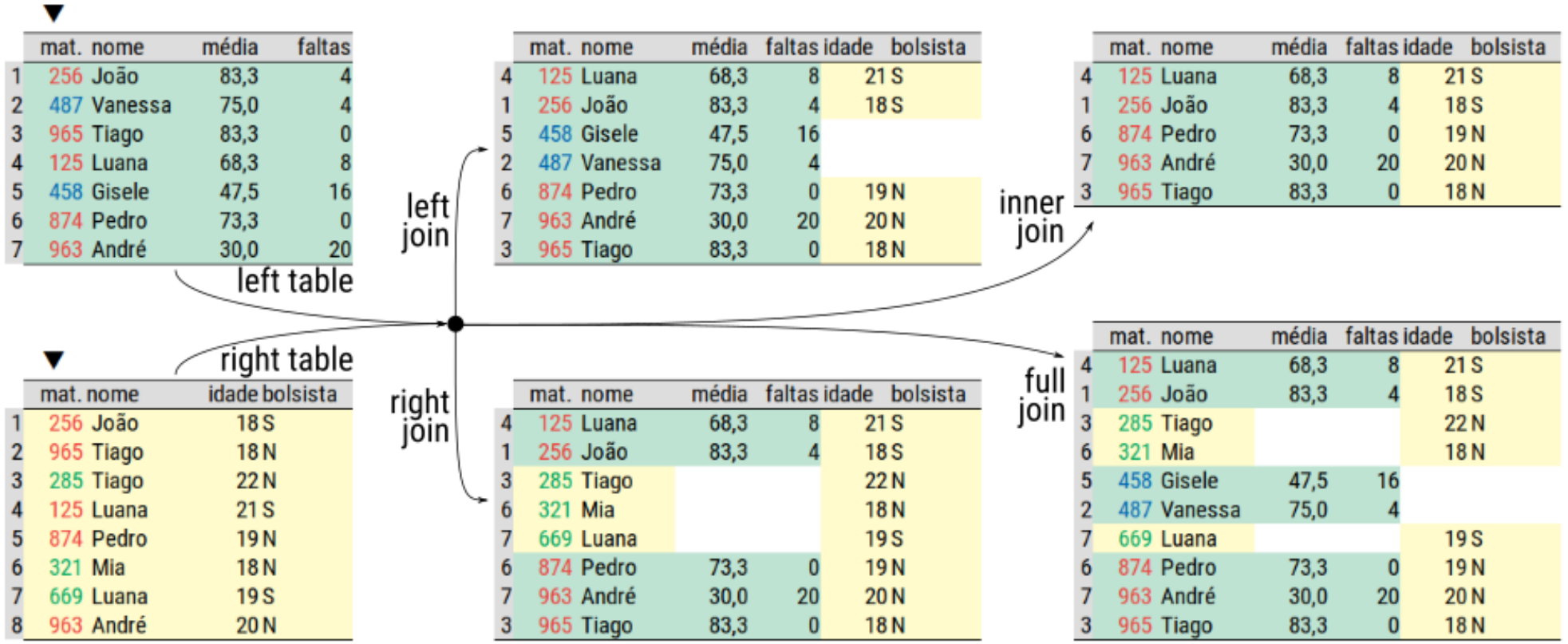
Junções

- Junções permitem parear dados de tabelas separadas quando elas possuem uma chave (ou chave primária).
- As operações de junção podem ser inicialmente de 4 tipos:
  - Junção por interseção (*inner join*).
  - Junção por união (*full join*).
  - Junção à esquerda (*left join*).
  - Junção à direita (*right join*).
  - Existe também os *exclusive joins*.



Tipos de junções de tabelas ilustrado com diagramas de Veen.

Junções



	mat.	nome	média	faltas
1	256	João	83,3	4
2	487	Vanessa	75,0	4
3	965	Tiago	83,3	0
4	125	Luana	68,3	8
5	458	Gisele	47,5	16
6	874	Pedro	73,3	0
7	963	André	30,0	20

	mat.	nome	idade	bolsista
1	256	João	18	S
2	965	Tiago	18	N
3	285	Tiago	22	N
4	125	Luana	21	S
5	874	Pedro	19	N
6	321	Mia	18	N
7	669	Luana	19	S
8	963	André	20	N

	mat.	nome	média	faltas	idade	bolsista
4	125	Luana	68,3	8	21	S
1	256	João	83,3	4	18	S
5	458	Gisele	47,5	16		
2	487	Vanessa	75,0	4		
6	874	Pedro	73,3	0	19	N
7	963	André	30,0	20	20	N
3	965	Tiago	83,3	0	18	N

	mat.	nome	média	faltas	idade	bolsista
4	125	Luana	68,3	8	21	S
1	256	João	83,3	4	18	S
6	874	Pedro	73,3	0	19	N
7	963	André	30,0	20	20	N
3	965	Tiago	83,3	0	18	N

	mat.	nome	média	faltas	idade	bolsista
4	125	Luana	68,3	8	21	S
1	256	João	83,3	4	18	S
3	285	Tiago			22	N
6	321	Mia			18	N
7	669	Luana			19	S
6	874	Pedro	73,3	0	19	N
7	963	André	30,0	20	20	N
3	965	Tiago	83,3	0	18	N

	mat.	nome	média	faltas	idade	bolsista
4	125	Luana	68,3	8	21	S
1	256	João	83,3	4	18	S
3	285	Tiago			22	N
6	321	Mia			18	N
5	458	Gisele	47,5	16		
2	487	Vanessa	75,0	4		
7	669	Luana			19	S
6	874	Pedro	73,3	0	19	N
7	963	André	30,0	20	20	N
3	965	Tiago	83,3	0	18	N

Junções de tabelas do tipo inclusivas.

Junções

```
# Full join = união.
full_join(df1, df_extra,
          by = c("mat" = "mat", "nome"))
```

```
# Inner join = intersecção.
inner_join(df1,
           df_extra,
           by = c("mat" = "mat",
                  "nome"))
```

```
## # A tibble: 4 × 11
##     mat nome  curso    p1    p2    p3
##   <dbl> <chr> <chr> <dbl> <dbl> <dbl>
## 1   256 João  Mat      80    90    80
## 2   965 Tiago Est      95    80    75
## 3   125 Luana Est      70    85    50
## 4   874 Pedro Mat      55    75    90
## # … with 5 more variables: fal <dbl>,
## #   media <dbl>, classificacao <fct>,
## #   idade <dbl>, bolsista <chr>
```

```
# Todos os que estão na 1º tabela
left_join(df1, df_extra,
          by = c("mat" = "mat",
                 "nome"))

# Todos os que estão na 2º tabela
right_join(df1, df_extra,
           by = c("mat" = "mat",
                  "nome"))
```

```
# Os da 2º que não aparecem na 1º.
anti_join(df1, df_extra,
           by = c("mat" = "mat",
                  "nome"))
```

```
## # A tibble: 3 × 9
##     mat nome    curso    p1    p2    p3
##   <dbl> <chr>   <chr> <dbl> <dbl> <dbl>
## 1   487 Vanessa Mat      75    75    75
## 2   458 Gisele  Est      45    50     0
## 3   963 André   Est      30     0    30
## # … with 3 more variables: fal <dbl>,
## #   media <dbl>, classificacao <fct>
```

Exportação de dados

Exportação de dados

Exportando arquivos em texto pleno

```
write_csv(df1,
          file = "Nome_do_arquivo.csv")
```

Arquivo binário do R

```
save(df1,
     file = "Nome_do_arquivo.RData")
## Carregando arquivo .RData
load("Nome_do_arquivo.RData")
```

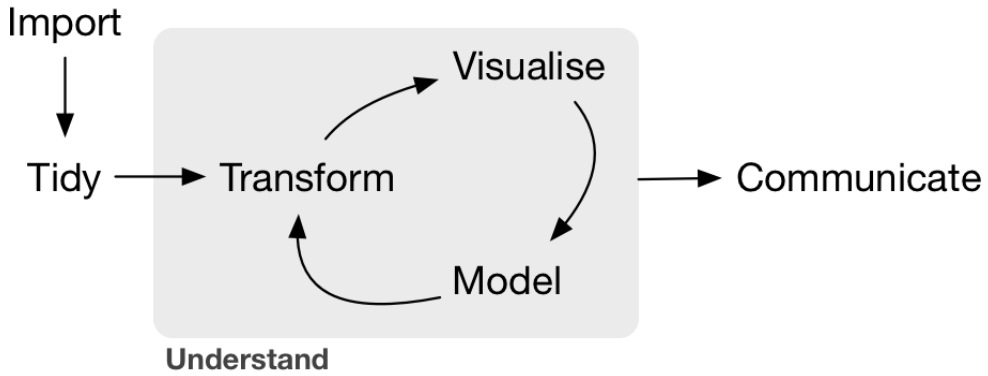
Criando planilha eletronica

```
library(writexl)
write_xlsx(df1, "Nome_do_arquivo.xlsx")
```

Considerações finais

Lidando com dados

- Principais etapas envolvidas ao lidar com dados.
  - Importação de dados.
  - Arrumação de dados.
  - Transformação e manipulação de dados.
  - Combinação de dados.
  - Exportação de dados.
- Gramática para manipulação de dados.
- Abordagem baseada no {tidyverse}.



Ciclo de vida da ciência de dados. Fonte da imagem: https://bookdown.org/fjmcgrade/ismaykim/#intro-for-students