Manipulação de dados no R

Tidyverse, Data Table a alternativas

Prof. Dr. Walmes Marques Zeviani (DEST)
Linguagens de Programação para Ciência de Dados
Especialização em Data Science & Big Data · UFPR

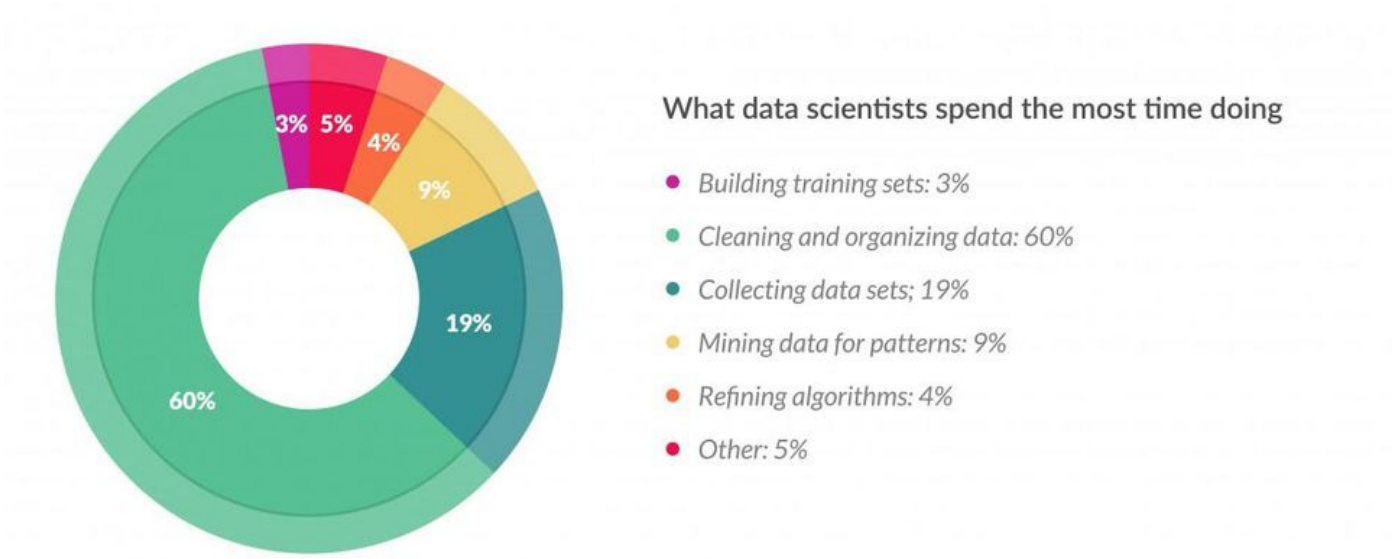
03 de junho de 2019

Motivação

Preparar. Apontar. Fogo!

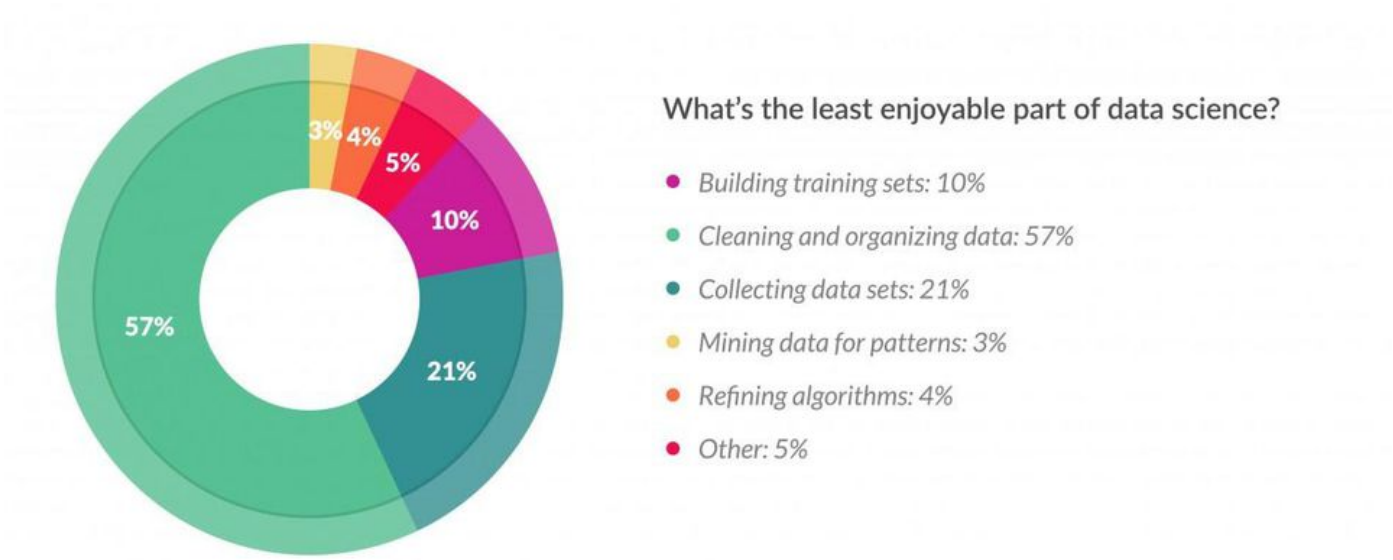
- Manipular e visualizar dados (MVD) são atividades obrigatórias em Data Science (DS).
- A MVD determina o sucesso de uma série de etapas.
  - Entendimento dos dados.
  - Limpeza e conciliação de dados.
  - Engenharia de características.
  - Especificação de modelos.
  - Comunicação de resultados, etc.
- 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.

O tempo gasto



Tempo gasto nas atividades de Data Science. Fonte: https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says.

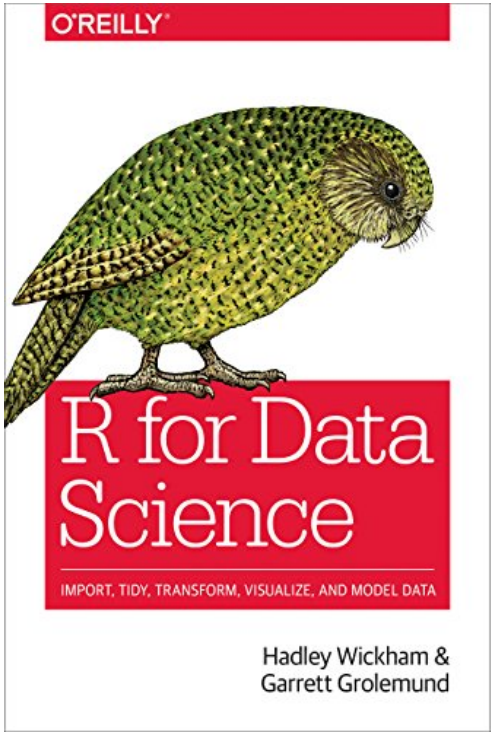
O quanto (não) é divertido



Quão enfadonhas são as atividades em Data Science. Fonte: https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says.

R para Ciência de Dados

R4DS · A principal referência



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

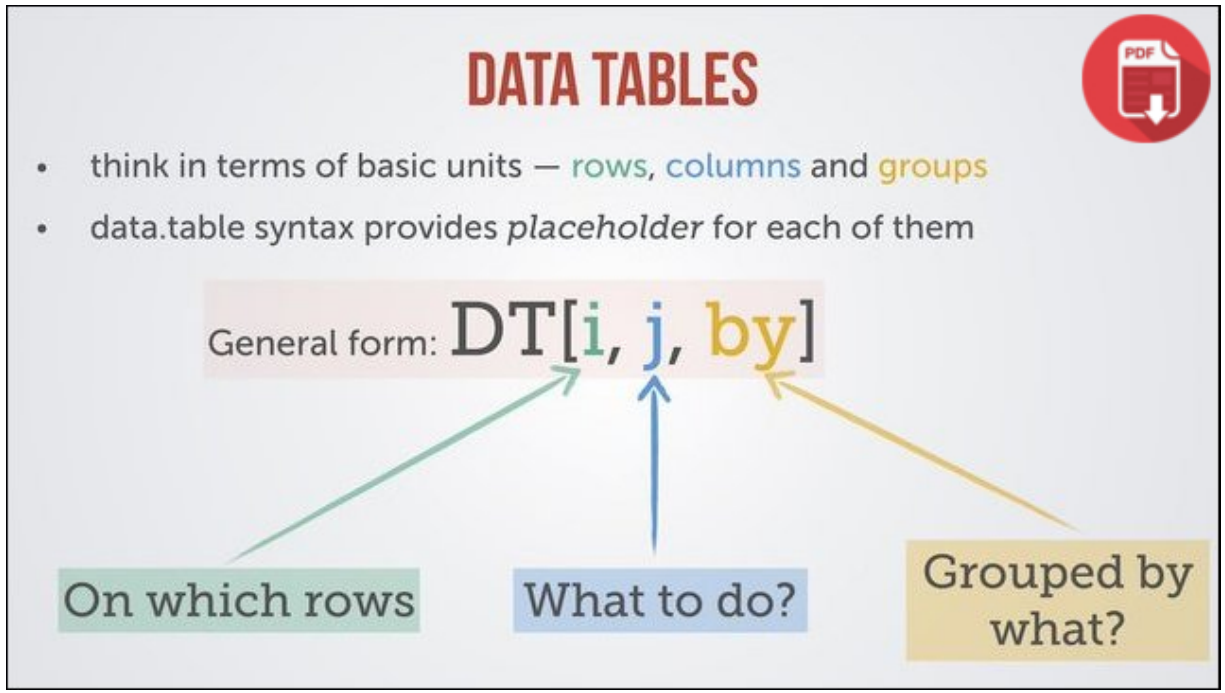
Principais bibliotecas

- Tidyverse
  - `tidyverse` = {`tibble`, `readr`, `tidyr`, `dplyr`, `ggplot2`, `stringr`, `forcats`, `purrr`}.
  - Hadleyverse = `tidyverse` + {`lubridate`, `rvest`, `xml2`, `httr`, etc}.
- Alternativas: Data table e SQLDF.
- Bancos de dados: Database task view.
- Machine learning: MLR, Caret, etc.
- Visualização de dados: Plotly, DataExplorer, Esquisse, Leaflet, igraph, networkD3, etc.
- Comunicação: Shiny, Rmarkdown, Knitr, etc.
- Web scraping: rvest, xml2, httr, Rcrawler, RSelenium, etc.
- Modelagem estatística: inúmeros, são mais 14 mil pacotes.

Reduzindo escopo: manipulação de dados

- R básico.
  - Classe `data.frame`.
  - Métodos disponíveis no pacote `base`, `reshape`, `reshape2`, etc.
  - Resolve muitos problemas mas tem espaços para melhorias.
- data.table: número 1 em performance para operações de manipulação de dados.
- sqldf: manipulação de data.frames com instruções SQL.
- tidyverse:
  - É uma coleção de pacotes.
  - Principais: `tibble`, `readr`, `dplyr`, `tidyr`.
  - Acessórios: `ggplot2`, `purrr`, `stringr`, `forcats`.

data.table



Sintaxe do `data.table`. Para uma visão geral, visite a wiki do projeto: https://github.com/Rdatatable/data.table/wiki.

```
library(data.table)

iris_dt <- as.data.table(iris)
iris_dt[,
        list(SPmean = mean(Sepal.Length),
             PLmean = mean(Petal.Length)),
        by = Species]
```

Species <fctr>	SPmean <dbl>	PLmean <dbl>
setosa	5.006	1.462
versicolor	5.936	4.260
virginica	6.588	5.552

3 rows

sqldf

- Permite uso de expressões SQL para operar em `data.frames`.
- Cria um banco de dados e faz as operações lá, trazendo os resultados como `data.frames`.
- Suporta vários backends: `RSQLite`, `RH2`, `RMySQL` and `RPostgreSQL`.
- Tutoriais rápidos:
  - https://jasminedaly.com/tech-short-papers/sqldf_tutorial.html.
  - http://dept.stat.lsa.umich.edu/~jerrick/courses/stat701/notes/sql.html.
  - https://anythingbutrbitrary.blogspot.com/2012/08/manipulating-data-frames-using-sqldf.html.
- Recomendado para:
  - Conhecedores de SQL.
  - Situações em que exigem maior desempenho.

```
library(sqldf)

sqldf(paste("SELECT Species,",
            "AVG(`Sepal.Length`) AS SLmean,",
            "AVG(`Petal.Length`) AS PLmean",
            "FROM iris",
            "GROUP BY Species"))
```

Species <fctr>	SLmean <dbl>	PLmean <dbl>
setosa	5.006	1.462
versicolor	5.936	4.260
virginica	6.588	5.552

3 rows

Manipulação de dados com Tidyverse

O `tidyverse`

- Oferece uma reimplementação e extensão das funcionalidades para manipulação e visualização.
- É uma coleção 8 de pacotes R que operam em harmonia.
- Eles foram planejados e construídos para trabalhar em conjunto.
- Possuem gramática, organização, filosofia e estruturas de dados mais clara.
- 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.

O que o `tidyverse` contém

```
library(tidyverse)
ls("package:tidyverse")
```

```
## [1] "tidyverse_conflicts" "tidyverse_deps"      "tidyverse_logo"
## [4] "tidyverse_packages"  "tidyverse_update"
```

```
tidyverse_packages()
```

```
##  [1] "broom"       "cli"         "crayon"      "dplyr"       "dbplyr"
##  [6] "forcats"     "ggplot2"     "haven"       "hms"         "httr"
## [11] "jsonlite"    "lubridate"   "magrittr"    "modelr"      "purrr"
## [16] "readr"       "readxl\n(>=" "reprex"      "rlang"       "rstudioapi"
## [21] "rvest"       "stringr"     "tibble"      "tidyr"       "xml2"
## [26] "tidyverse"
```

Os pacotes do `tidyverse`

Pacotes que fazer parte do `tidyverse`.

Mas na realidade... ;)

Mas em uma realidade pararela.

A anatomia do `tidyverse`

`tibble`

- O `data.frame` é a estrutura nativa (primitiva) para representar tabelas de dados.
- O `tibble` é uma reimplementação da estrutura com melhorias.
  - Método print mais enxuto e informativo.
  - Mais consistente para seleção e modificação de conteúdo.
  - Mais fácil conversão de outros formatos para `tibble`.
  - Colunas/cédulas podem representar objetos mais complexos.
- Documentação:
  - https://tibble.tidyverse.org/.
  - https://r4ds.had.co.nz/tibbles.html.
  - https://cran.r-project.org/package=tibble

readr

- O `readr` tem recursos para importação de dados retangulares na forma de texto pleno.
- 10x mais rápido que R básico.
- 1.2-2x mais lento de `data.table`.
- Leitura/escrita de dados tabulares: csv, tsv, fwf.
  - Funções de importação: `read_*()`.
  - Funções de escrita: `write_*()`.
  - Funções de *parsing*: `parse_*`.
- Recursos "inteligentes" que determinam tipo de variável.
- Ex: importar campos de datas como datas!
- Muitas opções de controle de importação:
  - Encoding, delimitador, decimal, aspas, comentários, etc.
- Documentação:
  - https://readr.tidyverse.org/.
  - https://r4ds.had.co.nz/data-import.html.
  - https://cran.r-project.org/package=readr

`tidyr`

- Suporte a criar de dados no formato `tidy` (tabular).
- Formato `tidy`:
  - Cada variável está em uma coluna.
  - Cada observação é uma linha.
  - Cada valor é uma cédula.
- Principais recursos:
  - Mudar disposição dos dados: long/empilhar $\rightleftharpoons$ wide/esparramar.
  - Lidar com valores ausentes.
  - Partir/concatenar variáveis.
  - Aninhar/desaninhar listas.
- Documentação:
  - https://tidyr.tidyverse.org/.
  - https://r4ds.had.co.nz/tidy-data.html.
  - https://cran.r-project.org/package=tidyr

`dplyr`

- O `dplyr` é a gramática para manipulação de dados.
- Tem um conjunto consistente de verbos para atuar sobre tabelas.
  - Verbos: `mutate()`, `select()`, `filter()`, `arrange()`, `summarise()`, `slice()`, `rename()`, etc.
  - Sufixos: `_at()`, `_if()`, `_all()`, etc.
  - Extratificação: `group_by()` e `ungroup()`.
  - Junções: `inner_join()`, `full_join()`, `left_join()` e `right_join()`.
- Destaque são as operações de *split-apply-combine*.
- Documentação:
  - https://dplyr.tidyverse.org/.
  - https://r4ds.had.co.nz/relational-data.html.
  - https://cran.r-project.org/package=dplyr.

ggplot2

- Criação de gráficos baseado no *The Grammar of Graphics* (WILKINSON et al., 2013).
- Claro mapeamento das variáveis em variáveis visuais e construção baseada em camadas.
- Referências:
  - WICKHAM (2016): `ggplot2` - Elegant Graphics for Data Analysis.
  - TEUTONICO (2015): `ggplot2` Essentials.
- Documentação: https://ggplot2.tidyverse.org/.

purrr

- O `purrr` fornece um conjunto completo e consistente para programação funcional.
- São uma sofisticação da *família* `apply`.
- Funções que aplicam funções em lote varrendo objetos: vetores, listas, etc.
- Várias função do tipo `map()` para cada tipo de input/output.
- Percorrem vetores, listas, colunas, linhas, etc.
- Permitem filtar, concatenar, parear listas, etc.
- Além disso, permite:
  - Chamar funções de forma não tradicional.
  - Aplicar funções para tratamento de excessões.
  - Operar de forma a acumular e reduzir recursivamente.
  - Aninhar e aplanar objetos.
- Documentação: https://purrr.tidyverse.org/.

`stringr`

- Recursos coesos construídos para manipulação de *strings*.
- Feito sobre o pacote `stringi`.
- Praticamente tudo que envolva aplicação de expressões regulares.
  - Detectar.
  - Contar.
  - Partir.
  - Extrair.
  - Substituir.
  - etc.
- Documentação:
  - https://stringr.tidyverse.org/.

forcats

- Para manipulação de variáveis categóricas/fatores.
- As principais operação são:
  - Renomenar: manualmente, programaticamente (truncar, abreviar, etc.).
  - Reordenar: manualmente, por frequência, por alguma variável.
  - Aglutinar: combinar níveis menos frequentes, etc.
- Documentação:
  - https://forcats.tidyverse.org/.
  - https://peerj.com/preprints/3163/.

Harmonizam bem com o `tidyverse`

Pacotes adicionais

- `magrittr`: operadores *pipe* → %>%.
- `rvest`: *web scraping*.
- `httr`: requisições HTTP e afins.
- `xml2`: manipulação de XML.
- `lubridate` e `hms`: manipulação de dados cronológicos.
- Os pacotes que melhor se acoplam são àqueles do Hadleyverse.

`lubridate` e `hms`

- Recursos para manipulação de dados *date*, *time* e *date-time*.
- Fácil decomposição de datas: dia, mês, semana, dia da semana, etc.
- Lida com fusos horários, horários de verão, etc.
- Extende para outras classes de dados baseados em *date-time*: duração, período, intervalos.
- Mas não é carregado junto com o `tidyverse`.

magrittr

- O operador permite expressar de forma mais direta as operações.
- É uma ideia inspirada no Shell e usada em várias linguagens.
- A lógica é bem simples:
  - `x %>% f` é o mesmo que `f(x)`.
  - `x %>% f(y)` é o mesmo que `f(x, y)`.
  - `x %>% f %>% g %>% h` é o mesmo que `h(g(f(x)))`.
- Existem outros operadores *pipe* para situações específicas.

Instalar o `tidyverse`

```
# Do CRAN (recomendado).
install.packages("tidyverse")

# Do GitHub.
# install.packages("devtools")
devtools::install_github("hadley/tidyverse")

# Atualizar caso já tenha instalado.
tidyverse_update()
```

Principais operações com o Tidyverse

Operações típicas de manipulação

1. Importar e/ou acessar dados.
2. Ordenar os registros da tabela.
3. Selecionar e fatiar nos índices/eixos.
4. Filtrar registros por predicado.
5. Renomear os índices/eixos.
6. Modificar a disposição do conteúdo.
7. Modificar/transformar o conteúdo.
8. Aplicar funções/calcular medidas resumo.
9. Agregar por categorias e aplicar.
10. Concatenar tabelas.
11. Juntar ou conciliar tabelas.

Importação do `tidyverse`

```
library(tidyverse)

u <- sessionInfo()
names(u$otherPkgs)

##  [1] "forcats"    "stringr"    "dplyr"      "purrr"      "readr"
##  [6] "tidyr"      "tibble"     "ggplot2"    "tidyverse"  "sqldf"
## [11] "RSQLite"    "gsubfn"     "proto"      "data.table" "rmarkdown"
## [16] "knitr"
```

Leitura de arquivos de dados

- As funções de leitura de arquivos do `readr` começam com `read_`.
- O argumento obrigatório é o caminho para o arquivo.
- Os demais são opcionais.

```
ls("package:readr") %>%
    str_subset("^read_")
```

```
[1] "forcats"   "stringr"   "dplyr"     "purrr"     "readr"     "tidyr"
[7] "tibble"    "ggplot2"   "tidyverse"
```

Exemplo de importação

- O caminho para o arquivo pode ser uma URL.
- A função faz a conexão e download do arquivo.

```
# Endereço web do arquivo, mas poderia ser local.
url <- "http://leg.ufpr.br/~walmes/data/euro_football_players.txt"

# Importa a tabela de dados.
tb <- read_tsv(file = url,
               comment = "#")
head(tb, n = 6)
```

Atributos do objeto

- A função `class()` retorna a classe do objeto.
- A função `str()` exibe a estrutura de um objeto.
- A função `attributes()` retorna atributos.
- A função `methods()` exibe os métodos de uma classe.

```
attributes(tb)
attr(tb, "spec") <- NULL

class(tb)
methods(class = "tbl_df")
str(tb)
```

Criação de um `data.frame`

	matrícula	nome	curso	prova1	prova2	prova3	faltas
1	256	João	Mat	80	90	80	4
2	487	Vanessa	Mat	75	75	75	4
3	965	Tiago	Est	95	80	75	0
4	125	Luana	Est	70	85	50	8
5	458	Gisele	Est	45	50		16
6	874	Pedro	Mat	55	75	90	0
7	963	André	Est	30		30	20

Uma tabela com dados fictícios.

Criação de um `data.frame`

Criação por colunas

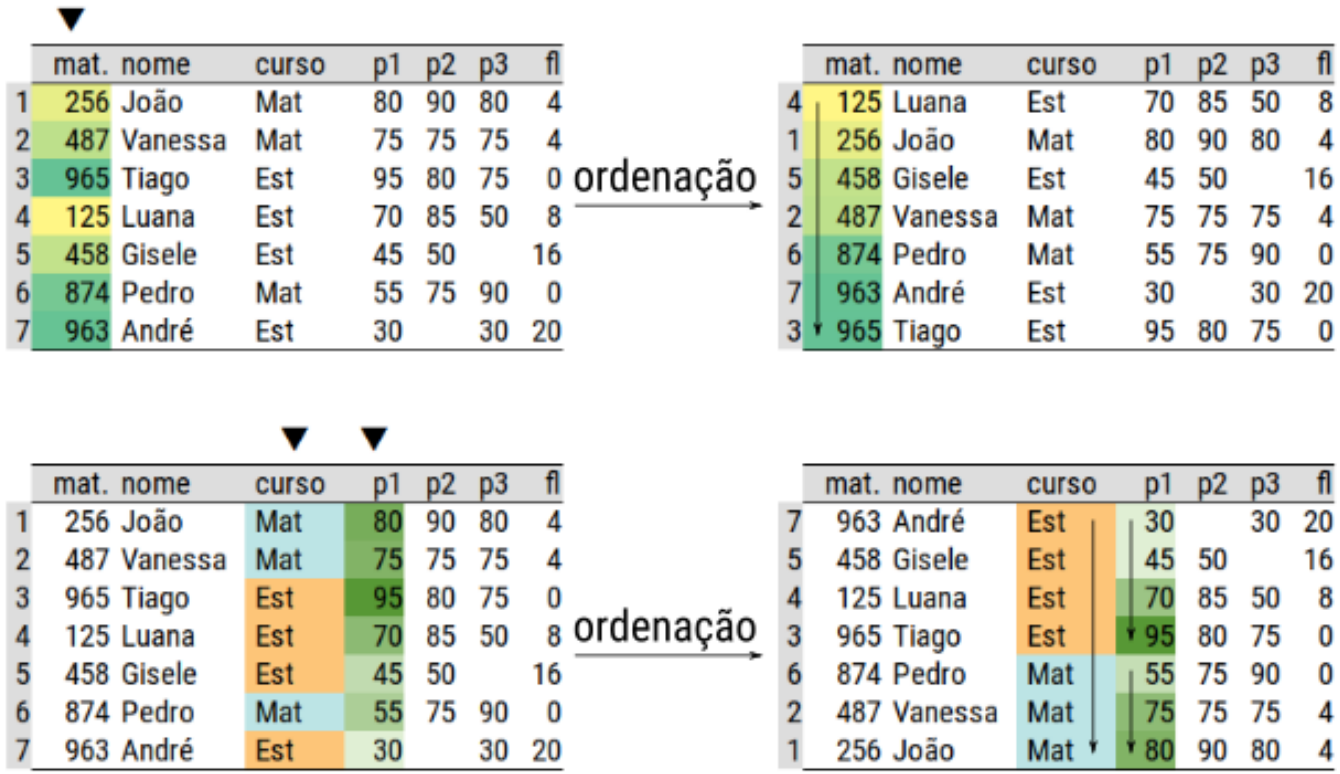
```
# Cria um tibble data.frame a partir de vetores.
df1 <- tibble(matricula = c(256, 487, 965, 125, 458, 874, 963),
              nome = c("João", "Vanessa", "Tiago", "Luana", "Gisele",
                       "Pedro", "André"),
              curso = c("Mat", "Mat", "Est", "Est", "Est", "Mat", "Est"),
              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))
df2 <- tibble(matricula = c(505, 658, 713),
              nome = c("Bia", "Carlos", "Cris"),
              curso = c("Eng", "Eng", "Eng"),
              prova1 = c(65, 75, 75),
              prova2 = c(85, 80, 90),
              faltas = c(0, 0, 2))
```

Criação de um `data.frame`

Criação por linhas

```
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",
)
```

Ordenação



Ordenação dos registros de uma tabela.

Ordenação

Por uma variável

```
df1 %>%
    arrange(matricula)
```

Ordenação

Por mais de uma variável

```
df1 %>% arrange(curso, desc(prova1))
```

`desc()`: ordenação de descente da variável.

Seleção e fatiamento

Nomes

```
names(df1)
colnames(df1)
rownames(df1)
```

Tamanho

```
dim(df1)
nrow(df1)
ncol(df1)
```

Seleção das variáveis (colunas)

Seleção com lista de índices

```
df1[, c("nome")]
df1[, c("nome", "prova1", "prova2", "prova3")]

df1 %>% select(c("nome", "prova1", "prova2", "prova3"))
df1 %>% select(nome, prova1, prova2, prova3)
df1 %>% select(-nome, -faltas)
df1 %>% select(prova1:prova3)
```

Seleção pela posição

```
df1[, 1:3]
df1[, c(1, 4)]
df1[, c(-1, -4)]
df1 %>% select(1:3)
df1 %>% select(1, 4)
df1 %>% select(-1, -4)
```

Questão!

1. Como selecionar apenas as posições pares?
2. Como selecionar um conjunto de intervalos: 1 a 3 e 5 a 8?
3. Como selecionar todas exceto as variáveis de uma lista?
4. Como selecionar de acordo com o tipo de valor: apenas a variáveis de tipo numérico?
5. Como selecionar variáveis baseado em padrões de caracteres (regex)?

Solução

```
df1[, seq(1, ncol(df1), by = 2)]

df1[, c(1:3, 5:8)]

v <- c("prova1", "prova2", "prova3")
df1 %>% select(-v)

df1 %>% select_if(is.numeric)

df1 %>% select(matches("^prova"))
df1 %>% select(matches("\\d$"))
df1 %>% select(matches("^.{6}$"))
```

Seleção de observações (linhas)

```
df1[1, ]
df1[3:5, ]
df1[-(3:5), ]
df1[c(3:4, 1:2), ]

tail(df1, n = 2)
head(df1, n = 2)

df1 %>% slice(1)
df1 %>% slice(3:5)
df1 %>% slice(-(3:5))
df1 %>% slice(c(3:4, 1:2))
```

No `tibble` e `data.frame` não é recomendado ter nome para as linhas. Veja explicação em https://tibble.tidyverse.org/reference/rownames.html.

Seleção de observações (linhas)

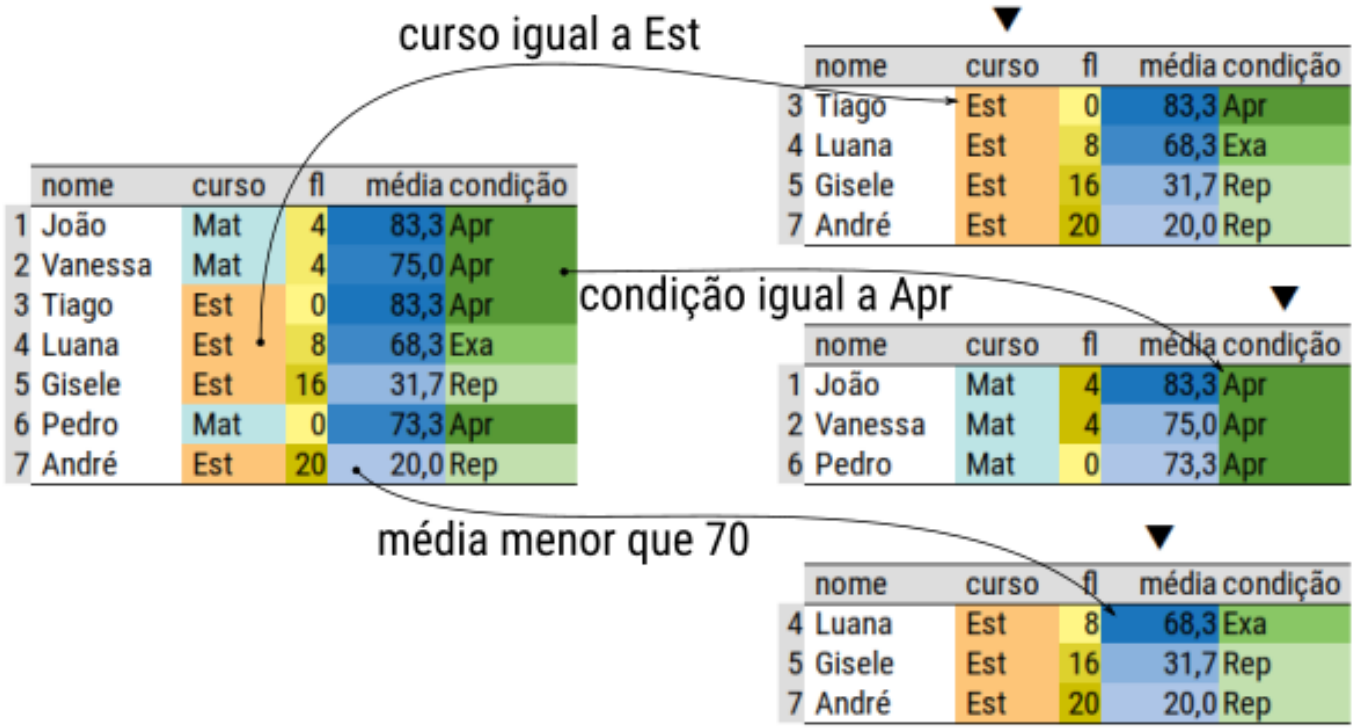
Por máscara lógica

```
i <- sample(c(TRUE, FALSE), size = nrow(df1),
            prob = c(0.4, 0.6), replace = TRUE)
df1[i, ]

df1[df1$prova1 < 50, ]

# Amostra aleatória das linhas.
df1 %>% sample_n(size = 3, replace = FALSE)
df1 %>% sample_frac(size = 0.5, replace = FALSE)
```

Seleção de registros

```
df1[2, "nome"]
```

Filtros



Filtro dos registros de uma tabela.

Filtros

Um vetor lógico

```
df1[df1$curso == "Est", ]
df1[df1$faltas == 0, ]
df1[df1$faltas != 0, ]
df1[df1$faltas %in% c("Aline", "Vanessa"), ]

df1 %>% filter(curso == "Est")
df1 %>% filter(faltas == 0)
df1 %>% filter(faltas != 0)
df1 %>% filter(faltas %in% c("Aline", "Vanessa"))
```

Renomeação

	matrícula	nome	curso	prova1	prova2	prova3	faltas
1	256	João	Mat	80	90	80	4
2	487	Vanessa	Mat	75	75	75	4
3	965	Tiago	Est	95	80	75	0
4	125	Luana	Est	70	85	50	8
5	458	Gisele	Est	45	50		16
6	874	Pedro	Mat	55	75	90	0
7	963	André	Est	30		30	20

encurtar substituir abreviar

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

Formas de renomear as colunas de uma tabela.

Renomeação

Via pares de substituição

```
# Renomeia nomes de colunas (variáveis).
df1 %>% rename("mat." = "matricula", "fl" = "faltas")
```

Função de transformação de *strings*

```
names(df1) <- names(df1) %>% str_to_upper()
names(df1) <- names(df1) %>% str_sub(start = 1, stop = 3)
```

Transformação

As operações podem modificar a tabela com a:

1. Criação de novas variáveis.
2. Remoção de variáveis.
3. Transformação de variáveis.

As operações de criação/transformação podem ser:

1. Matemáticas: aritméticas, potência, logarítmicas, trigonométricas, etc.
2. Compartimentação (*binning*): agrupar em classes.
3. Conversão de tipo de valor: i.e. de `int` $\rightarrow$ `str`.
4. Substituição: i.e. preencher um valor ausente.

Transformação

As transformações podem ser:

1. Uma $\rightarrow$ uma:

$$y = \log(x).$$

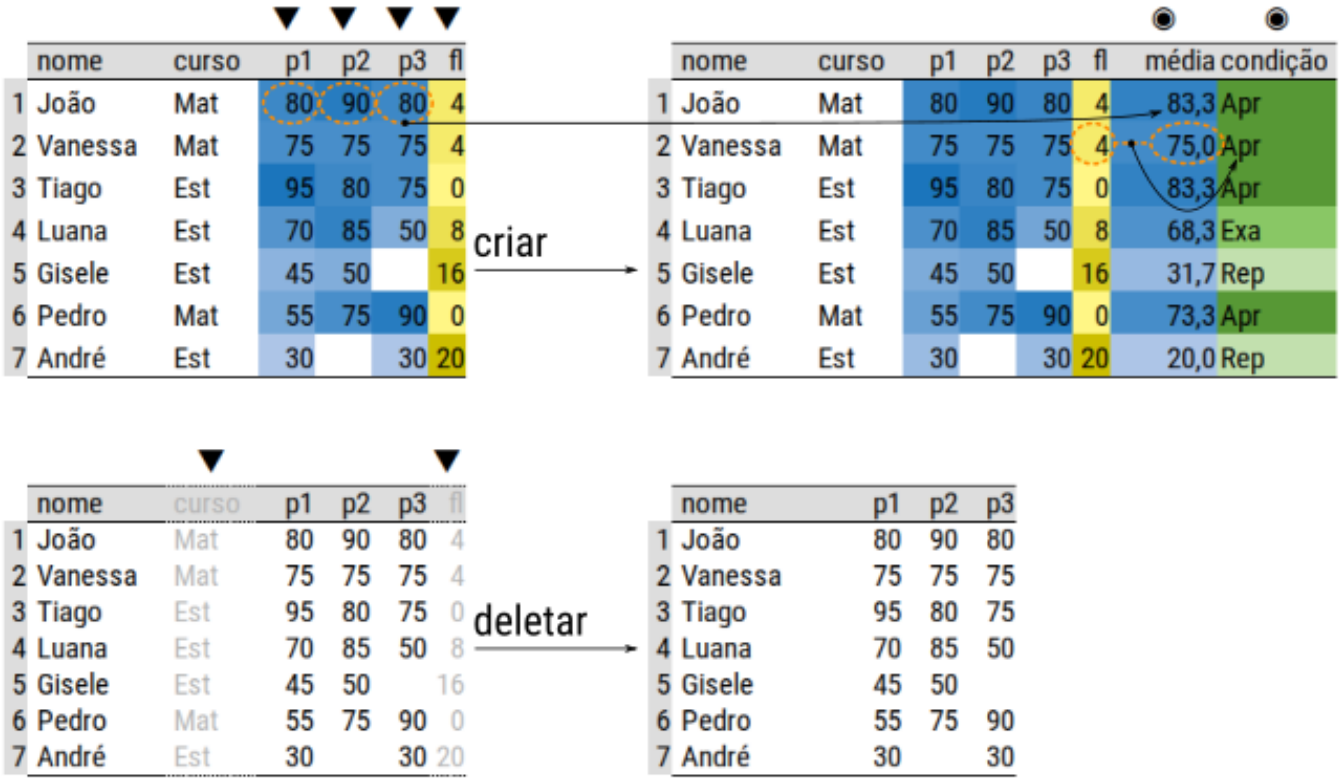
2. Várias $\rightarrow$ uma:

$$z = x/y^2.$$

3. Várias $\rightarrow$ várias:

$$y_1, \cdots, y_k = f_1(x_1, \cdots, x_m), \cdots, f_k(x_1, \cdots, x_m).$$

Transformações



Criação e deleção de variáveis em uma tabela.

Transformações

Transformações matemáticas

```
df1$media <- with(df1, (prova1 + prova2 + prova3/3))
df1$media <- df1 %>% select(prova1:prova3) %>% apply(MARGIN = 1, mean)
```

Transformações

Transformações matemáticas homogêneas

```
df1 %>% mutate_if(is.numeric, sqrt)
df1 %>% mutate_if(is.numeric, log)
df1 %>% mutate_if(is.character, str_to_upper)
```

Transformações

Compartimentação

```
# Intervalos para corte e rótulos.
inter <- c(0, 4, 7, Inf)
condi <- c("reprovado", "exame", "aprovado")

# Cria a variável que é a condição.
df1[["condicao"]] <- cut(df1[["media"]],
                         breaks = inter,
                         labels = condi,
                         right = FALSE,
                         include.lowest = TRUE)
```

Transformações

Conversão ou coerção

```
df1 %>% mutate_if(is.character, as.factor)
df1 %>% mutate_if(is.numeric, as.integer)
```

Transformações

Substituição

```
df1$prova2 %>% is.na()
df1$prova3 %>% negate(is.na)()
df1$prova3 %>% is.na() %>% `!`()

df1$prova2 %>% replace_na(replace = 0)
df1 %>% replace_na(replace = list(prova2 = 0, prova3 = 0, faltas = 60))
```

Transformações

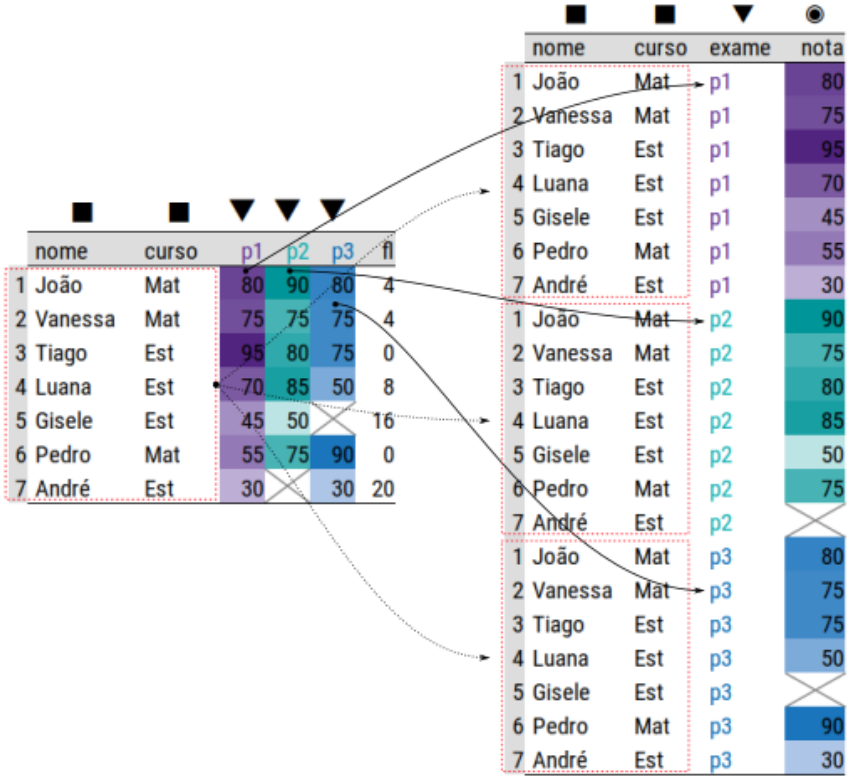
Remoção

```
df1$media <- NULL
df1$condicao <- NULL
```

Rearranjo

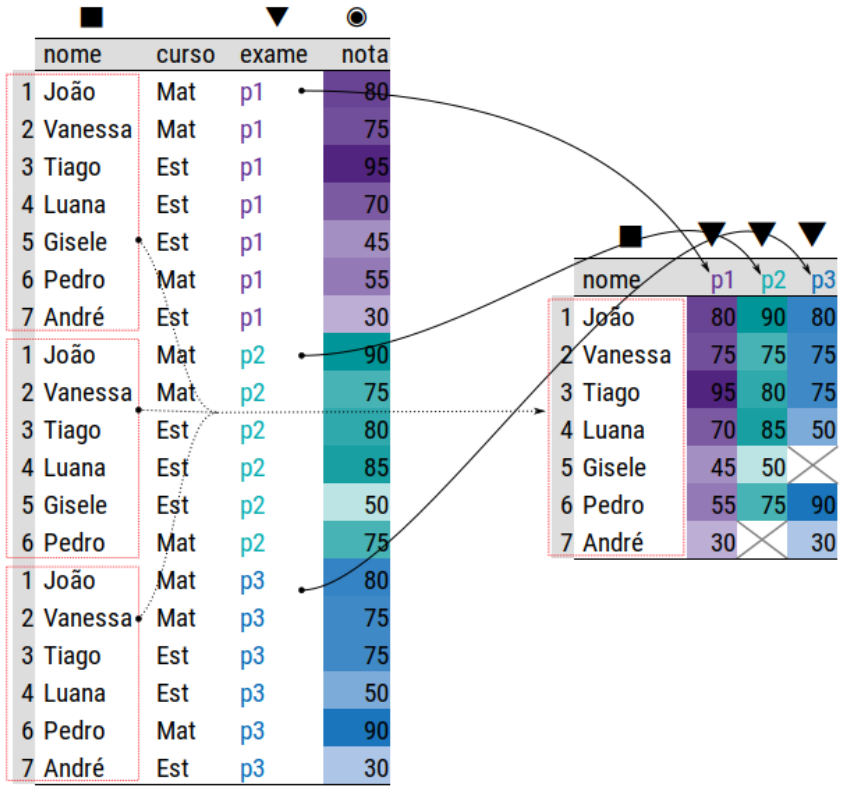
- São operações de *reshaping* da tabela.
- Modificam a disposição dos registros.
  - Empilhar ou amontoar um conjunto de variáveis.
  - Desempilhar ou esparramar os níveis de uma variável.

Rearranjo · Empilhar



Modificação da disposição com empilhamento.

Rearranjo · Esparramar



Modificação da disposição com desempilhamento.

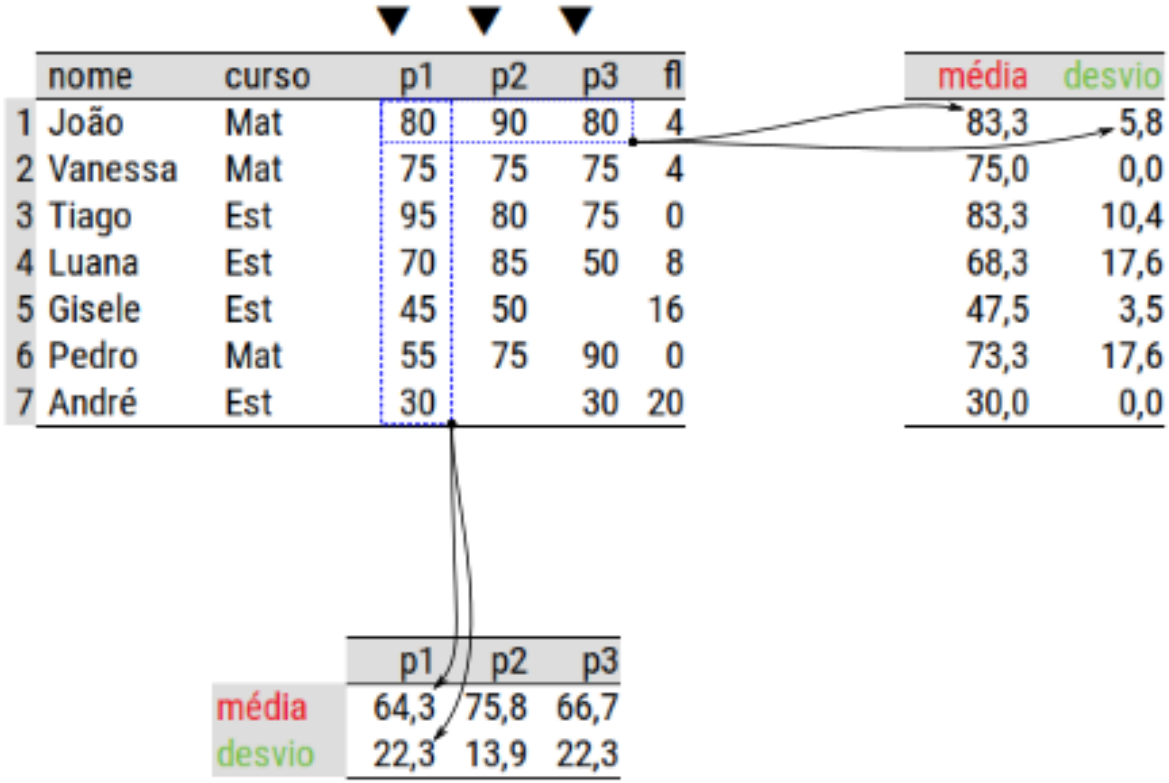
Rearranjo

```
# Gather = amontoar.
u <- df1 %>%
    gather(key = "exame", value = "nota", prova1:prova3)

# Spread = esparramar.
v <- u %>%
    spread(key = "exame", value = "nota")
```

Medidas resumo

- Operações para determinar estatísticas descritivas.
  - Soma, média, mediana, quartis, quantis, etc.
  - Variância, desvio-padrão, amplitude, desvio absoluto da mediana, coeficiente de variação, etc.
  - Número de níveis distintos, frequências absolutas/relativas, etc.
- Elas podem ser marginais ou considerar a estratificação conforme uma ou mais variáveis categóricas.
- Podem ser aplicadas em todas as variáveis de um mesmo tipo (homegêneo).

Medidas resumo



Cálculo de medidas resumo.

Medidas resumo

Uma variável ou uma estatística

```
with(df1, c(sum(prova1), mean(prova1), max(prova1),
            min(prova1), median(prova1), sd(prova1),
            var(prova1), length(prova1)))

df1 %>% summarise(sum(prova1), mean(prova1), max(prova1),
                  min(prova1), median(prova1), sd(prova1),
                  var(prova1), length(prova1)) %>% t()

quantile(df1$prova1, probs = c(0.25, 0.75))
table(df1$prova1)
```

Medidas resumo

Estatísticas definidas pelo usuário

```
df1 %>% summarise(CV = 100 * sd(prova1)/mean(prova1))
```

```
CV <- function(x, ...) 100 * sd(x, ...)/mean(x, ...)
```

```
df1 %>%
    summarise_if(is.numeric, CV, na.rm = TRUE)
```

Medidas resumo

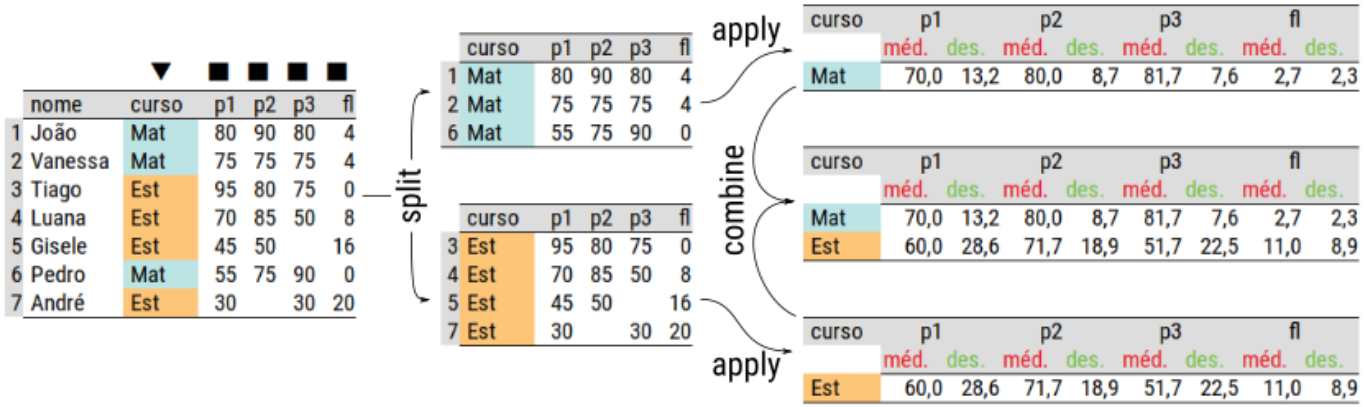
Várias estatísticas para várias variáveis

```
df1 %>%
    summarise_at(vars(prova1:prova3, faltas),
                 c("mean", "max", "min", "CV"),
                 na.rm = TRUE)
```

Agregação

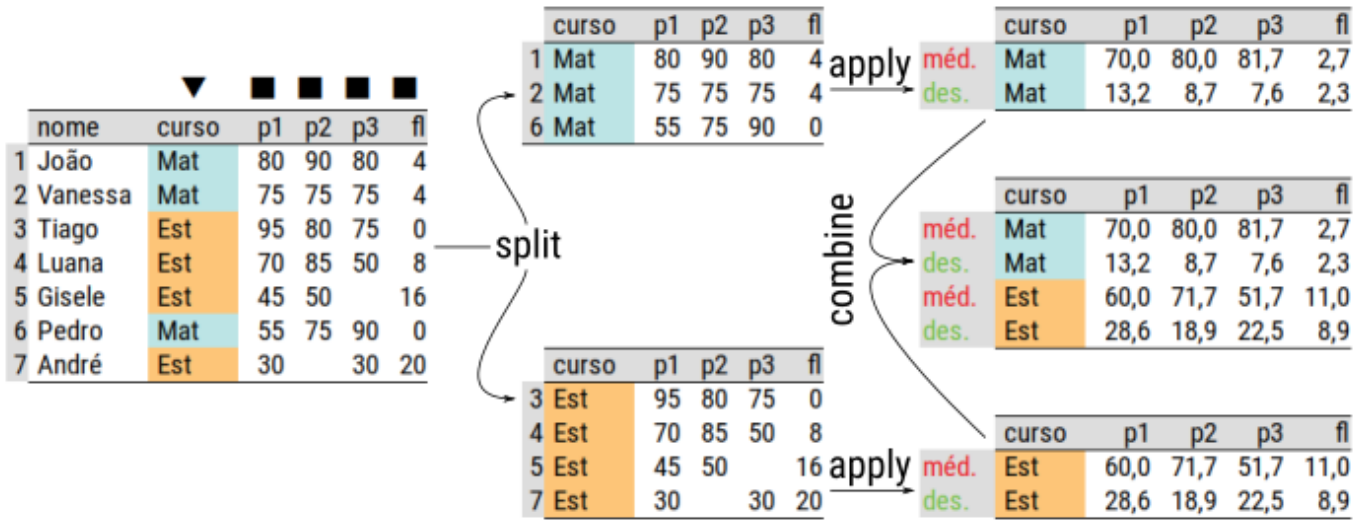
- Consiste em aplicar estatísticas em variáveis fazendo a extratificação por outras variáveis.
- São tarefas conhecidas como *split-apply-combine*.
- Ou também chamadas de *GROUP BY*.

Agregação



Agregação de uma tabela.

Agregação



	nome	curso	p1	p2	p3	fl
1	João	Mat	80	90	80	4
2	Vanessa	Mat	75	75	75	4
3	Tiago	Est	95	80	75	0
4	Luana	Est	70	85	50	8
5	Gisele	Est	45	50		16
6	Pedro	Mat	55	75	90	0
7	André	Est	30		30	20

	curso	p1	p2	p3	fl
1	Mat	80	90	80	4
2	Mat	75	75	75	4
6	Mat	55	75	90	0

	curso	p1	p2	p3	fl
3	Est	95	80	75	0
4	Est	70	85	50	8
5	Est	45	50		16
7	Est	30		30	20

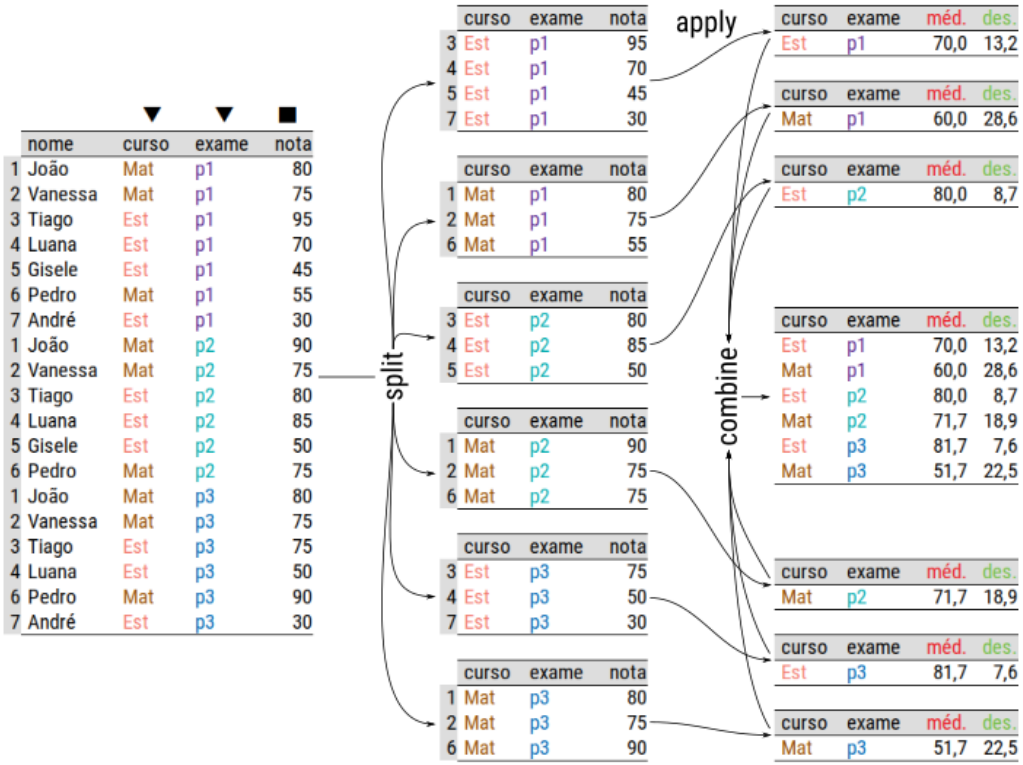
	curso	p1	p2	p3	fl
méd.	Mat	70,0	80,0	81,7	2,7
des.	Mat	13,2	8,7	7,6	2,3

	curso	p1	p2	p3	fl
méd.	Est	60,0	71,7	51,7	11,0
des.	Est	28,6	18,9	22,5	8,9

	curso	p1	p2	p3	fl
méd.	Mat	70,0	80,0	81,7	2,7
des.	Mat	13,2	8,7	7,6	2,3
méd.	Est	60,0	71,7	51,7	11,0
des.	Est	28,6	18,9	22,5	8,9

Agregação de uma tabela.

Agregação



	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

Agregação de uma tabela.

Agregação

```
df1 %>%
    group_by(curso) %>%
    summarise_at(vars(prova1:prova3, faltas),
                 c("mean", "sd"),
                 na.rm = TRUE)

df1 %>%
    select(curso, prova1:prova3, faltas) %>%
    gather(key = "variavel", value = "valor", -curso) %>%
    group_by(curso, variavel) %>%
    summarise_at("valor",
                 c("mean", "sd"),
                 na.rm = TRUE)
```

Agregação

- Uma vez que uma tabela é agrupada, várias informações e métodos estão disponíveis.

```
u <- df1 %>%
    group_by(curso)

class(u)
methods(class = "grouped_df")

n_groups(u)
group_vars(u)
group_size(u)
group_indices(u)
```

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.

Concatenação

	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.

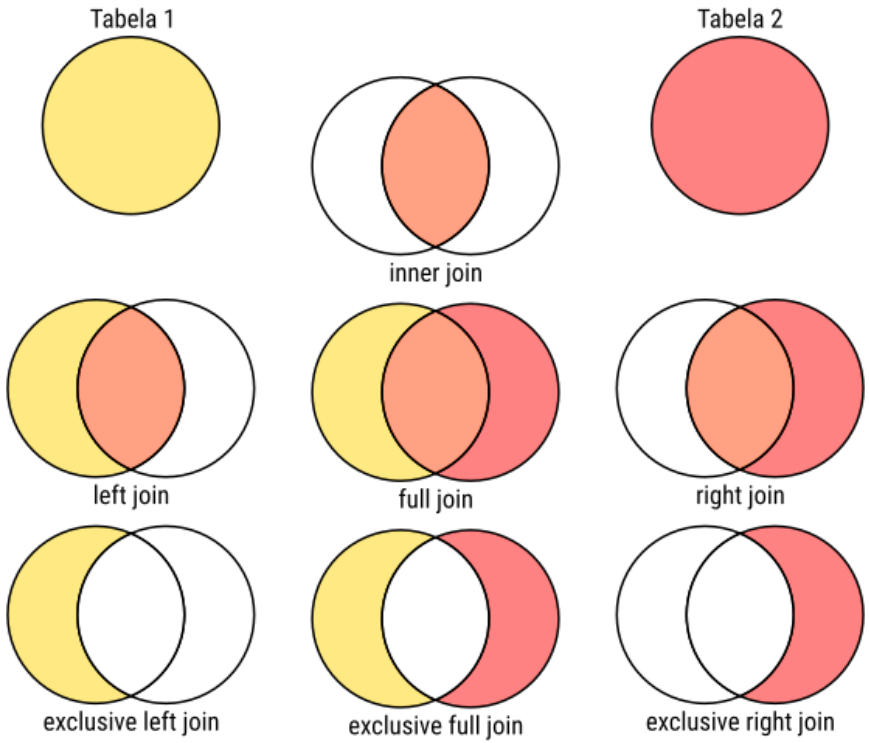
Concatenação

```
bind_rows(df1[1:3, c(1, 3, 5)],
          df1[5:7, c(1, 3, 5, 4)],
          df1[4,   c(1,    5, 4)])

bind_cols(df1[, c(1:3)],
          df1[, c(6:7)])
```

Junções

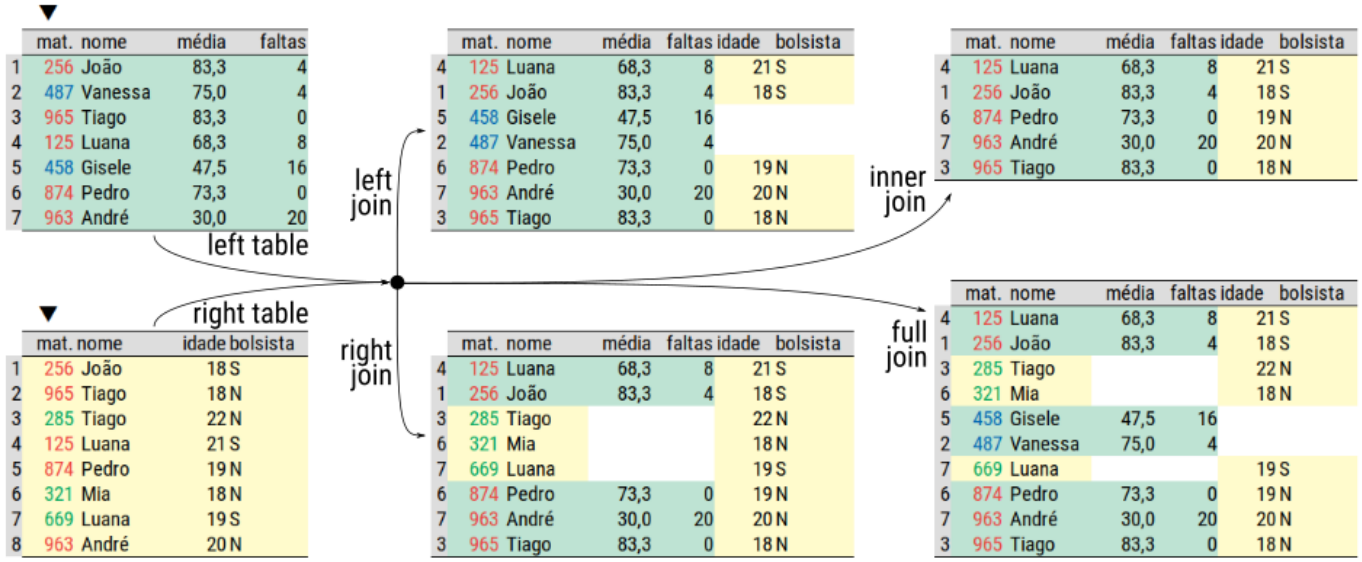
- Junções permitem parear dados de tabelas separadas quando elas possuem uma chave (ou chave primária).
- As operações de junção podem ser inicialmente de 4 tipos:
  - Junção por interseção (*inner join*).
  - Junção por união (*full join*).
  - Junção à esquerda (*left join*).
  - Junção à direita (*right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

Junções



Tipos de junções de tabelas ilustrado com diagramas de Veen.

Junções



Junções de tabelas do tipo inclusivas.

Junções

```
inner_join(df1, df_extra, by = c("matricula" = "mat", "nome"))
full_join(df1, df_extra, by = c("matricula" = "mat", "nome"))
left_join(df1, df_extra, by = c("matricula" = "mat", "nome"))
right_join(df1, df_extra, by = c("matricula" = "mat", "nome"))
```

Considerações

- Foram vistas as principais operações com dados tabulares.
- Para mais detalhes consulte a documentação e livros dedicados.
- Exercícios serão aplicados para fixação do conteúdo.
- Links para guias de referência e tutoriais serão dados no final.
- Conexão com banco de dados será visto em detalhe em outra disciplina.

Gráficos

Gráficos no R

- Gráficos serão abordados na próxima aula.

Exercícios

```
url <- "http://leg.ufpr.br/~walmes/data/jogadores-brasileirao-2018.txt"
bra <- read_tsv(url)
attr(bra, "spec") <- NULL
str(bra)
```

Referências

TEUTONICO, D. ggplot2 essentials. Packt Publishing, 2015.

WICKHAM, H. ggplot2: Elegant graphics for data analysis. Springer International Publishing, 2016.

WILKINSON, L.; WILLS, D.; ROPE, D.; NORTON, A.; DUBBS, R. The grammar of graphics. Springer New York, 2013.