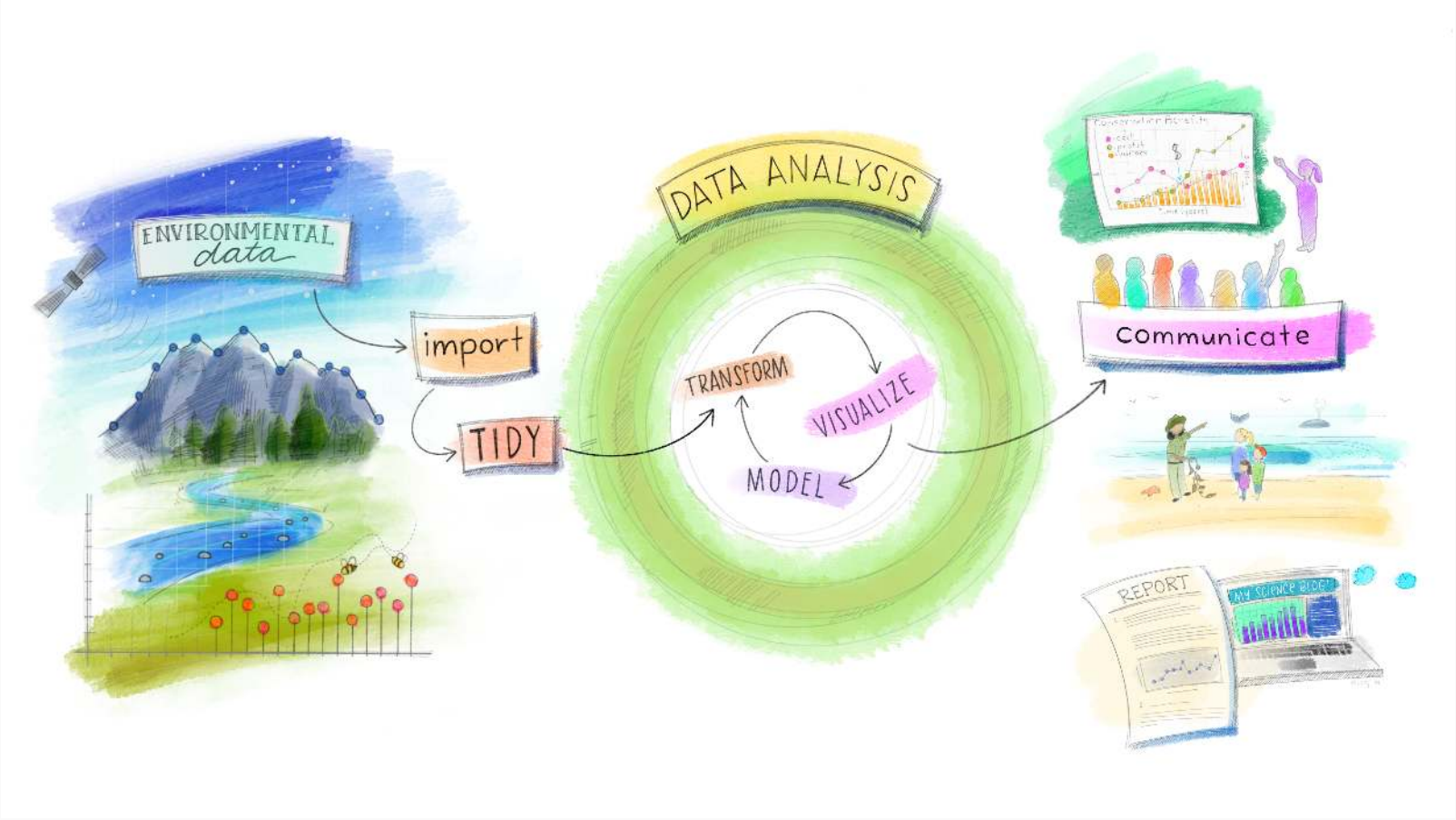
Introdução ao uso de dados geoespaciais no R

4 Introdução ao tidyverse

Maurício H. Vancine
Milton C. Ribeiro
UNESP - Rio Claro
Laboratório de Ecologia Espacial e Conservação (LEEC)
25/10/2021-05/11/2021



@allison_horst

4 Introdução ao tidyverse

Conteúdo

1. Contextualização
2. tidyverse
3. here
4. readr, readxl e writexl
5. tibble
6. magrittr (pipe - %>%)
7. tidyr
8. dplyr
9. stringr
10. forcats
11. lubridate
12. purrr

tidyverse

4 tidyverse

Script

`04_script_intro_geoespacial_r.R`

1. Contextualização

Descrição

O tidyverse é um **conjunto de pacotes** designados para **Data Science**

Todos os pacotes compartilham uma **filosofia** de design, gramática e estruturas de dados

É um **"dialeto"** novo para a linguagem R

tidy: organizado, arrumado, ordenado

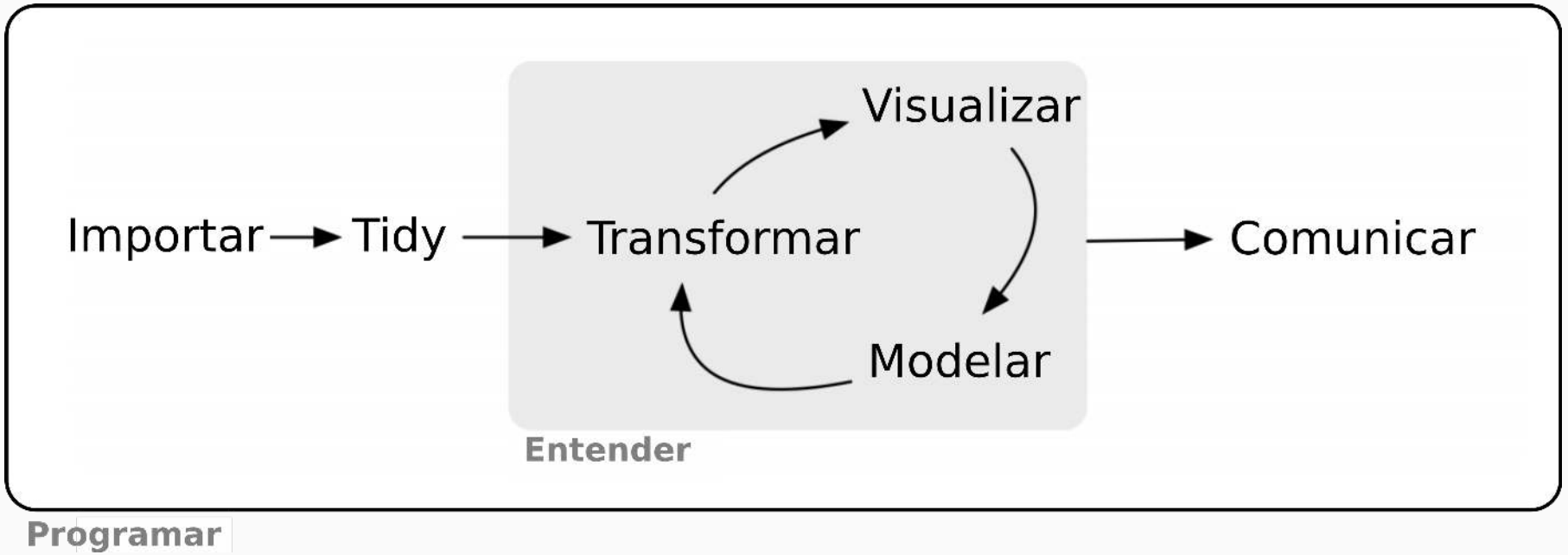
verse: universo

What is the tidyverse?

Iniciativa Vingadores do R

1. Contextualização

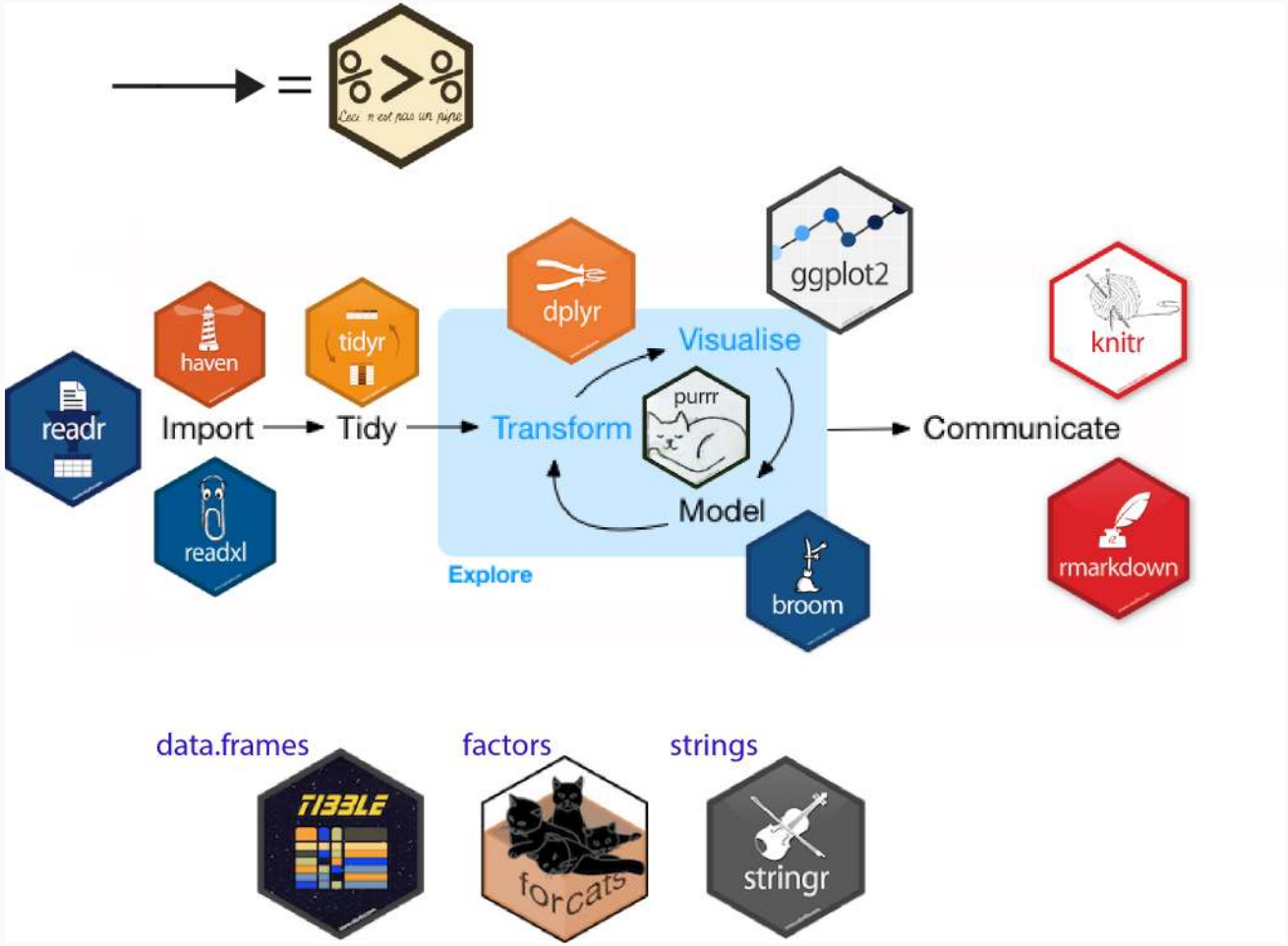
Fluxo de trabalho



Wickham & Grolemund (2017)

1. Contextualização

Pacotes



1. Contextualização

Pacotes e suas funções

1. Contextualização

O idealizador foi o **Hadley Wickham** e atualmente **muitas pessoas** têm contribuído para sua expansão

Hadley Wickham

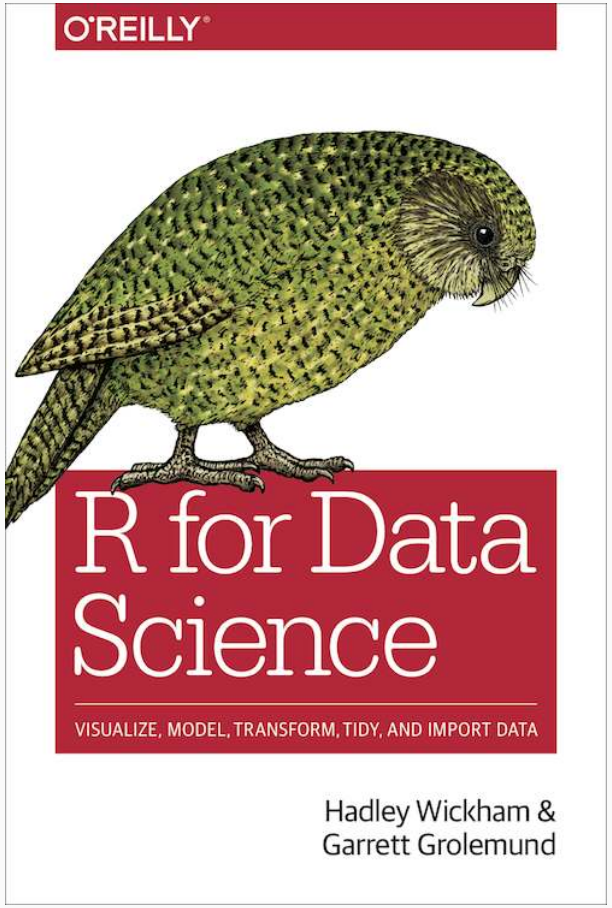
1. Contextualização

Artigos

> - Wickham, Hadley. "Tidy data." Journal of Statistical Software 59.10 (2014): 1-23.
> - Wickham, Hadley, et al. "Welcome to the Tidyverse." Journal of Open Source Software 4.43 (2019): 1686.

1. Contextualização

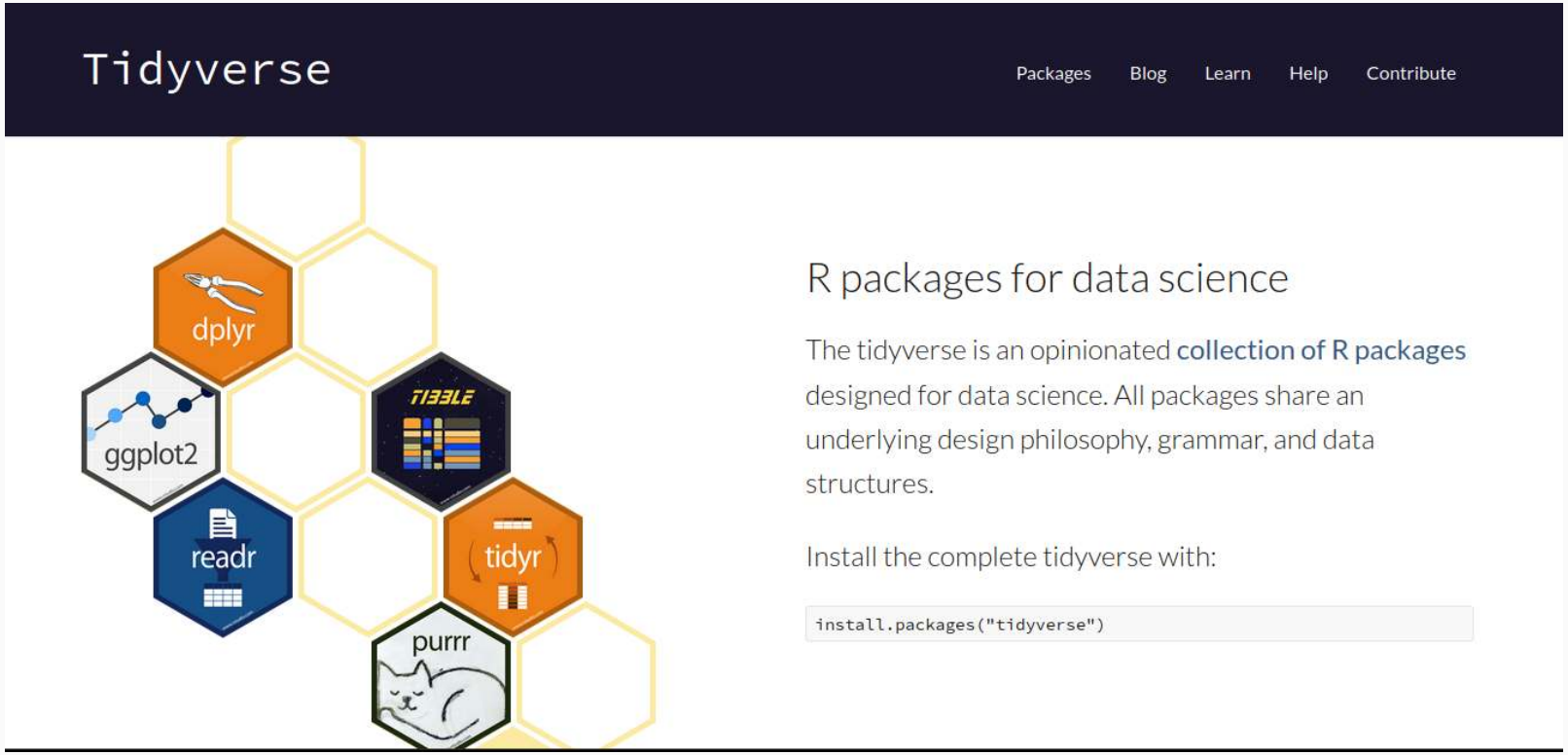
R for Data Science (2017)



Wickham & Grolemund (2017)

1. Contextualização

Sites



tidyverse

1. Contextualização

Sites

RPubs by RStudio — Sign in — Register

R para data science

Manipulação e Visualização de dados utilizando os pacotes do tidyverse()

Uma breve discussão sobre os procedimentos padrões para se trabalhar com dados

Para uma melhor compreensão de como é feito, em geral, o trabalho em data science, vamos definir os procedimentos ou passos a serem considerados na análise. Os procedimentos que apresentamos decorrem das ideias descritas no livro (Grolemund,2017) e podem ser observados na figura abaixo.

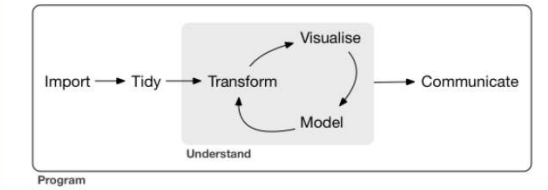


Diagrama de ferramentas para Data Science em (Grolemund,2017)

Pensando em um projeto com dados retangulares, devemos ter **as seguintes etapas**:

Passo 1: Formulação da pergunta de pesquisa

R para data science

tidyverse

2. tidyverse

Descrição

Para utilizar os pacotes do **tidyverse** é preciso instalar e carregar o pacote `tidyverse`

```
# instalar pacote
install.packages("tidyverse")
```

```
# carregar pacote
library(tidyverse)
```

2. tidyverse

Listar todos os pacotes do tidyverse

```
# listar todos os pacotes no tidyverse
tidyverse::tidyverse_packages(include_self = TRUE)
```

```
##  [1] "broom"         "cli"           "crayon"        "dbplyr"        "dplyr"         "dtplyr"        "forcats"
##  [9] "googlesheets4" "ggplot2"       "haven"         "hms"           "httr"          "jsonlite"      "lubridate"
## [17] "modelr"        "pillar"        "purrr"         "readr"         "readxl"        "reprex"        "rlang"
## [25] "rvest"         "stringr"       "tibble"        "tidyr"         "xml2"          "tidyverse"
```

2. tidyverse

Sintaxe

Todas as funções dos pacotes **tidyverse** usam fonte minúscula e usam `_` para separar os nomes internos das funções (snake_case)

`read_csv()`

`read_xlsx()`

`as_tibble()`

`left_join()`

`group_by()`

Sintaxe

É interessante **indicar de quais pacotes as funções** são utilizadas (`pacote::função`), para **evitar erros** com funções de outros pacotes

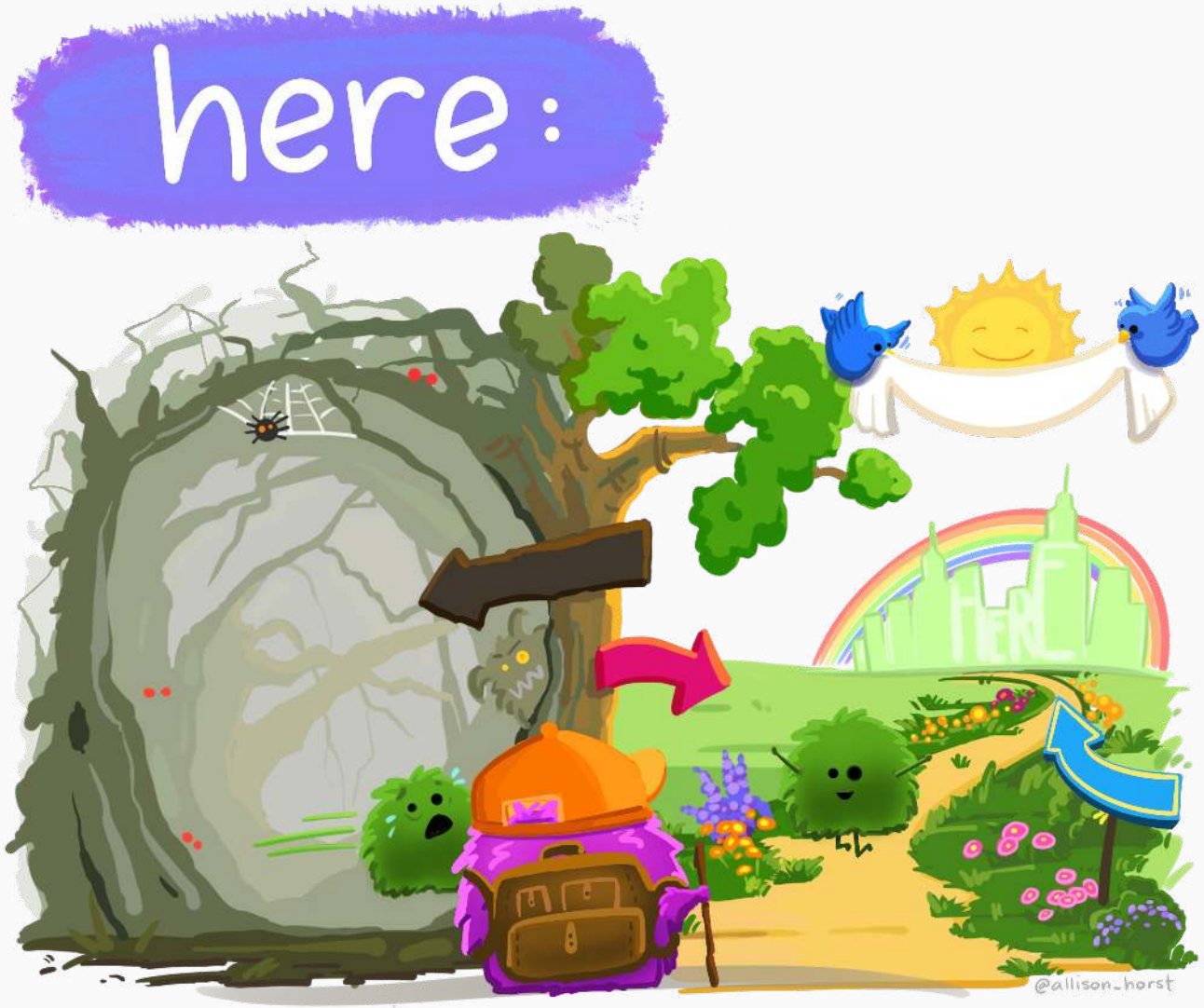
`readr::read_csv()`

`readxl::read_xlsx()`

`tibble::as_tibble()`

`dplyr::left_join()`

`dplyr::group_by()`



here @allison_horst

3. here

Descrição

Constrói caminhos para os arquivos do seu projeto

- **Mudar o diretório**: tende à ser chato demais (principalmente se é um código alheio)

- **Fragilidade**: conectado exatamente a um lugar e a um momento (baixa reprodutibilidade)

- **Subdiretórios**: facilita importar ou exportar para subpastas (Lei do Mínimo Esforço)

3. here

Pacote

```
# instalar
install.packages("here")

# carregar
library(here)
```

@allison_horst

3. here

Diversos critérios para encontrar o diretório raiz

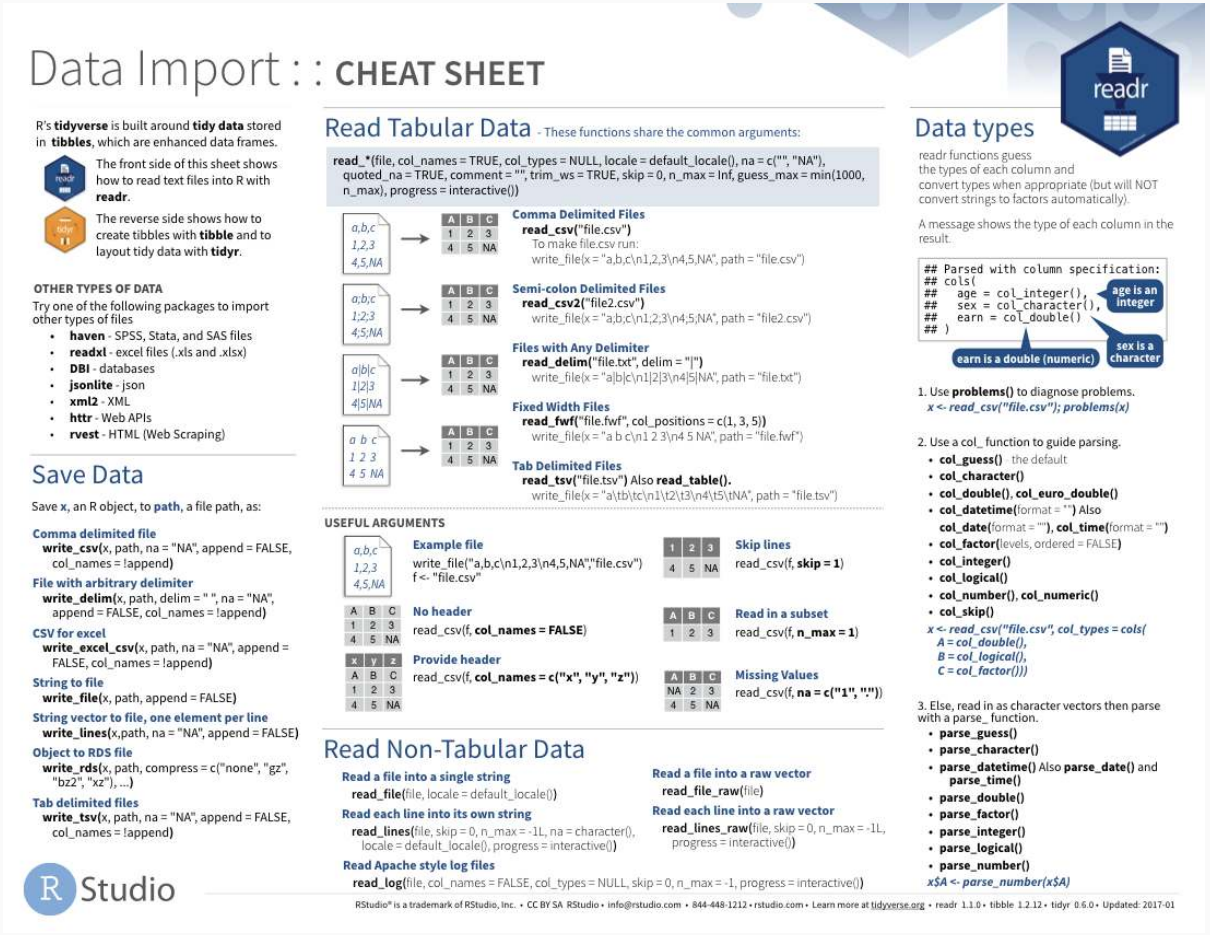
```
# conferir
here::here()

# criar um arquivo .here
here::set_here()
```

@allison_horst


readr
www.rstudio.com

4. readr, readxl e writexl

Data Import Cheatsheet



Data Import : : CHEAT SHEET

R's **tidyverse** is built around **tidy data** stored in **tibbles**, which are enhanced data frames.

The front side of this sheet shows how to read text files into R with **readr**.

The reverse side shows how to create tibbles with **tibble** and to layout tidy data with **tidyr**.

OTHER TYPES OF DATA

Try one of the following packages to import other types of files

- **haven** - SPSS, Stata, and SAS files
- **readxl** - excel files (.xls and .xlsx)
- **DBI** - databases
- **jsonlite** - json
- **xml2** - XML
- **httr** - Web APIs
- **rvest** - HTML (Web Scraping)

Save Data

Save **x**, an R object, to **path**, a file path, as:

Comma delimited file
write_csv(x, path, na = "NA", append = FALSE, col_names = !append)

File with arbitrary delimiter
write_delim(x, path, delim = " ", na = "NA", append = FALSE, col_names = !append)

CSV for excel
write_excel_csv(x, path, na = "NA", append = FALSE, col_names = !append)

String to file
write_file(x, path, append = FALSE)

String vector to file, one element per line
write_lines(x,path, na = "NA", append = FALSE)

Object to RDS file
write_rds(x, path, compress = c("none", "gz", "bz2", "xz"), ...)

Tab delimited files
write_tsv(x, path, na = "NA", append = FALSE, col_names = !append)

Read Tabular Data - These functions share the common arguments:

read_*(file, col_names = TRUE, col_types = NULL, locale = default_locale(), na = c("", "NA"), quoted_na = TRUE, comment = "", trim_ws = TRUE, skip = 0, n_max = Inf, guess_max = min(1000, n_max), progress = interactive())

Comma Delimited Files
read_csv("file.csv")
To make file.csv run:
write_file(x = "a,b,c\n1,2,3\n4,5,NA", path = "file.csv")

Semi-colon Delimited Files
read_csv2("file2.csv")
write_file(x = "a;b;c\n1;2;3\n4;5;NA", path = "file2.csv")

Files with Any Delimiter
read_delim("file.txt", delim = "|")
write_file(x = "a|b|c\n1|2|3\n4|5|NA", path = "file.txt")

Fixed Width Files
read_fwf("file.fwf", col_positions = c(1, 3, 5))
write_file(x = "a b c\n1 2 3\n4 5 NA", path = "file.fwf")

Tab Delimited Files
read_tsv("file.tsv") Also **read_table()**.
write_file(x = "a\tb\tc\n1\t2\t3\n4\t5\tNA", path = "file.tsv")

USEFUL ARGUMENTS

Example file
write_file("a,b,c\n1,2,3\n4,5,NA","file.csv")
f <- "file.csv"

No header
read_csv(f, **col_names = FALSE**)

Provide header
read_csv(f, **col_names = c("x", "y", "z")**)

Skip lines
read_csv(f, **skip = 1**)

Read in a subset
read_csv(f, **n_max = 1**)

Missing Values
read_csv(f, **na = c("1", ".")**)

Read Non-Tabular Data

Read a file into a single string
read_file(file, locale = default_locale())

Read each line into its own string
read_lines(file, skip = 0, n_max = -1L, na = character(), locale = default_locale(), progress = interactive())

Read Apache style log files
read_log(file, col_names = FALSE, col_types = NULL, skip = 0, n_max = -1, progress = interactive())

Read a file into a raw vector
read_file_raw(file)

Read each line into a raw vector
read_lines_raw(file, skip = 0, n_max = -1L, progress = interactive())

Data types

readr functions guess the types of each column and convert types when appropriate (but will NOT convert strings to factors automatically).

A message shows the type of each column in the result.

```
## Parsed with column specification:
## cols(
##   age = col_integer(),
##   sex = col_character(),
##   earn = col_double()
## )
```

age is an integer; sex is a character; earn is a double (numeric)

1. Use **problems()** to diagnose problems.
x <- read_csv("file.csv"); problems(x)

2. Use a col_ function to guide parsing.
- **col_guess()** - the default
- **col_character()**
- **col_double()**, **col_euro_double()**
- **col_datetime**(format = "") Also **col_date**(format = ""), **col_time**(format = "")
- **col_factor**(levels, ordered = FALSE)
- **col_integer()**
- **col_logical()**
- **col_number()**, **col_numeric()**
- **col_skip()**

x <- read_csv("file.csv", col_types = cols(
A = col_double(),
B = col_logical(),
C = col_factor()))

3. Else, read in as character vectors then parse with a parse_ function.
- **parse_guess()**
- **parse_character()**
- **parse_datetime()** Also **parse_date()** and **parse_time()**
- **parse_double()**
- **parse_factor()**
- **parse_integer()**
- **parse_logical()**
- **parse_number()**

x$A <- parse_number(x$A)

RStudio® is a trademark of RStudio, Inc. • CC BY SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more at tidyverse.org • readr 1.1.0 • tibble 1.2.12 • tidyr 0.6.0 • Updated: 2017-01

[Data Import Cheatsheet]

4. readr, readxl e writexl

readr - Descrição

Carrega e salva grandes arquivos de forma **mais rápida**

As funções **read.csv()** e **read.csv2()** são substituídas pelas funções **read_csv()** e **read_csv2()**

Essas funções fornecem **medidores de progresso**

E também **classificam** automaticamente o **modo** dos dados de cada coluna

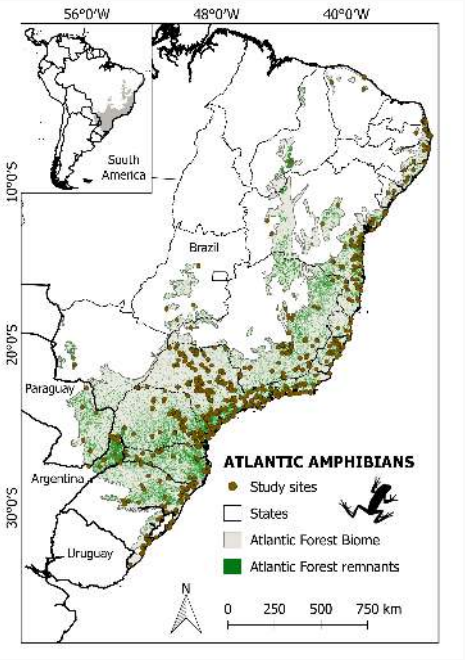
A classe do objeto atribuído é **tibble**

Para salvar arquivos no formato .csv: **write_csv()** e **write_csv2()**

4. readr, readxl e writexl

ATLANTIC AMPHIBIANS: a dataset of amphibian communities from the Atlantic Forests of South America

Eu mesmo et al. (2018)



Vancine et al. (2018)

4. readr, readxl e writexl

readr

Formato .csv

```
# importar sites com here
si ← readr::read_csv(here::here("03_dados", "tabelas", "ATLANTIC_AMPHIBIANS_sites.csv"),
                     locale = readr::locale(encoding = "latin1"))
si
```

```
## ¤[90m# A tibble: 1,163 × 25¤[39m
##    id      reference_number species_number record sampled_habitat active_methods passive_methods complementary_me…
##    ¤[3m¤[90m<chr>¤[39m¤[23m              ¤[3m¤[90m<dbl>¤[39m¤[23m          ¤[3m¤[90m<dbl>¤[39m¤[23m ¤[3m¤[90m<c
## ¤[90m 1¤[39m amp1001            ¤[4m1¤[24m001             19 ab     fo,ll           as              pt
## ¤[90m 2¤[39m amp1002            ¤[4m1¤[24m002             16 co     fo,la,ll        as              pt
## ¤[90m 3¤[39m amp1003            ¤[4m1¤[24m002             14 co     fo,la,ll        as              pt
## ¤[90m 4¤[39m amp1004            ¤[4m1¤[24m002             13 co     fo,la,ll        as              pt
## ¤[90m 5¤[39m amp1005            ¤[4m1¤[24m003             30 co     fo,ll,br        as              ¤[31mNA¤[39m
## ¤[90m 6¤[39m amp1006            ¤[4m1¤[24m004             42 co     tp,pp,la,ll,is  ¤[31mNA¤[39m            ¤[3
## ¤[90m 7¤[39m amp1007            ¤[4m1¤[24m005             23 co     sp              as              ¤[31mNA¤[39m
## ¤[90m 8¤[39m amp1008            ¤[4m1¤[24m005             19 co     sp,la,sw        as,sb,tr        ¤[31mNA¤[39m
## ¤[90m 9¤[39m amp1009            ¤[4m1¤[24m005             13 ab     fo              ¤[31mNA¤[39m            pt
## ¤[90m10¤[39m amp1010            ¤[4m1¤[24m006              1 ab     fo              ¤[31mNA¤[39m            pt
```

4. readr, readxl e writexl

readr

Formato .csv

```
# importar sites sem here
si ← readr::read_csv("./03_dados/tabelas/ATLANTIC_AMPHIBIANS_sites.csv",
                     locale = readr::locale(encoding = "latin1"))
si
```

```
## ¤[90m# A tibble: 1,163 × 25¤[39m
##    id      reference_number species_number record sampled_habitat active_methods passive_methods complementary_me…
##    ¤[3m¤[90m<chr>¤[39m¤[23m              ¤[3m¤[90m<dbl>¤[39m¤[23m          ¤[3m¤[90m<dbl>¤[39m¤[23m ¤[3m¤[90m<c
## ¤[90m 1¤[39m amp1001            ¤[4m1¤[24m001             19 ab     fo,ll           as              pt
## ¤[90m 2¤[39m amp1002            ¤[4m1¤[24m002             16 co     fo,la,ll        as              pt
## ¤[90m 3¤[39m amp1003            ¤[4m1¤[24m002             14 co     fo,la,ll        as              pt
## ¤[90m 4¤[39m amp1004            ¤[4m1¤[24m002             13 co     fo,la,ll        as              pt
## ¤[90m 5¤[39m amp1005            ¤[4m1¤[24m003             30 co     fo,ll,br        as              ¤[31mNA¤[39m
## ¤[90m 6¤[39m amp1006            ¤[4m1¤[24m004             42 co     tp,pp,la,ll,is  ¤[31mNA¤[39m            ¤[3
## ¤[90m 7¤[39m amp1007            ¤[4m1¤[24m005             23 co     sp              as              ¤[31mNA¤[39m
## ¤[90m 8¤[39m amp1008            ¤[4m1¤[24m005             19 co     sp,la,sw        as,sb,tr        ¤[31mNA¤[39m
## ¤[90m 9¤[39m amp1009            ¤[4m1¤[24m005             13 ab     fo              ¤[31mNA¤[39m            pt
## ¤[90m10¤[39m amp1010            ¤[4m1¤[24m006              1 ab     fo              ¤[31mNA¤[39m            pt
```

4. readr, readxl e writexl

readr

Formato .txt

```
# importar sites
si ← readr::read_tsv(here::here("03_dados", "tabelas", "ATLANTIC_AMPHIBIANS_sites.txt"))
si
```

```
## ¤[90m# A tibble: 1,163 × 25¤[39m
##    id      reference_number species_number record sampled_habitat active_methods passive_methods complementary_me…
##    ¤[3m¤[90m<chr>¤[39m¤[23m              ¤[3m¤[90m<dbl>¤[39m¤[23m          ¤[3m¤[90m<dbl>¤[39m¤[23m ¤[3m¤[90m<c
## ¤[90m 1¤[39m amp1001             ¤[4m1¤[24m001             19 ab     fo,ll           as              pt
## ¤[90m 2¤[39m amp1002             ¤[4m1¤[24m002             16 co     fo,la,ll        as              pt
## ¤[90m 3¤[39m amp1003             ¤[4m1¤[24m002             14 co     fo,la,ll        as              pt
## ¤[90m 4¤[39m amp1004             ¤[4m1¤[24m002             13 co     fo,la,ll        as              pt
## ¤[90m 5¤[39m amp1005             ¤[4m1¤[24m003             30 co     fo,ll,br        as              ¤[31mNA¤[39m
## ¤[90m 6¤[39m amp1006             ¤[4m1¤[24m004             42 co     tp,pp,la,ll,is  ¤[31mNA¤[39m            ¤[3
## ¤[90m 7¤[39m amp1007             ¤[4m1¤[24m005             23 co     sp              as              ¤[31mNA¤[39m
## ¤[90m 8¤[39m amp1008             ¤[4m1¤[24m005             19 co     sp,la,sw        as,sb,tr        ¤[31mNA¤[39m
## ¤[90m 9¤[39m amp1009             ¤[4m1¤[24m005             13 ab     fo              ¤[31mNA¤[39m            pt
## ¤[90m10¤[39m amp1010             ¤[4m1¤[24m006              1 ab     fo              ¤[31mNA¤[39m            pt
## ¤[90m# … with 1,153 more rows, and 14 more variables: month_finish <dbl>, year_finish <dbl>, effort_months <dbl>,
```


readxl
www.rstudio.com

readxl e writexl

Pacotes para importar e exportar planilhas no formato Excel®

```
# importar .xlsx
install.packages("readxl")
library("readxl")
```

```
# exportar .xlsx
install.packages("writexl")
library("writexl")
```

4. readr, readxl e writexl

readxl - Desccrição

Carrega e salva grandes arquivos de forma **mais rápida** no formato **.xlsx**

Funções **read_xlsx()** e **read_xlsx2()**

Essas funções fornecem **medidores de progresso**

E também **classificam** automaticamente o **modo** dos dados de cada coluna

A classe do objeto atribuído é **tibble**

Para salvar arquivos no formato .xlsx: **write_xlsx()** e **write_xlsx2()**

4. readr, readxl e writexl

Formato .xlsx

O argumento `sheet` define a aba a ser importada (número ou nome)

```
# importar sites
si ← readxl::read_xlsx(here::here("03_dados", "tabelas", "ATLANTIC_AMPHIBIANS_sites.xlsx"),
                       sheet = 1)
si
```

```
## ¤[90m# A tibble: 1,163 × 25¤[39m
##    id      reference_number species_number record sampled_habitat active_methods passive_methods complementary_me…
##    ¤[3m¤[90m<chr>¤[39m¤[23m              ¤[3m¤[90m<dbl>¤[39m¤[23m          ¤[3m¤[90m<dbl>¤[39m¤[23m ¤[3m¤[90m<c
## ¤[90m 1¤[39m amp1001            ¤[4m1¤[24m001             19 ab     fo,ll           as              pt
## ¤[90m 2¤[39m amp1002            ¤[4m1¤[24m002             16 co     fo,la,ll        as              pt
## ¤[90m 3¤[39m amp1003            ¤[4m1¤[24m002             14 co     fo,la,ll        as              pt
## ¤[90m 4¤[39m amp1004            ¤[4m1¤[24m002             13 co     fo,la,ll        as              pt
## ¤[90m 5¤[39m amp1005            ¤[4m1¤[24m003             30 co     fo,ll,br        as              NA
## ¤[90m 6¤[39m amp1006            ¤[4m1¤[24m004             42 co     tp,pp,la,ll,is  NA              NA
## ¤[90m 7¤[39m amp1007            ¤[4m1¤[24m005             23 co     sp              as              NA
## ¤[90m 8¤[39m amp1008            ¤[4m1¤[24m005             19 co     sp,la,sw        as,sb,tr        NA
## ¤[90m 9¤[39m amp1009            ¤[4m1¤[24m005             13 ab     fo              NA              pt
## ¤[90m10¤[39m amp1010            ¤[4m1¤[24m006              1 ab     fo              NA              pt
```

Importar os dados

Formato .csv

Dados das localidades

```
# importar sites
si ← readr::read_csv(here::here("03_dados", "tabelas", "ATLANTIC_AMPHIBIANS_sites.csv"),
                     locale = readr::locale(encoding = "latin1"))
si
```

Dados das espécies

```
# importar species
sp ← readr::read_csv(here::here("03_dados", "tabelas", "ATLANTIC_AMPHIBIANS_species.csv"),
                     locale = readr::locale(encoding = "latin1"))
sp
```

tibble

5. tibble

O tibble (classe *tbl_df*) é um **tipo especial de data frame**

É o **formato** aconselhado para que as funções do tidyverse **funcionem**

Converter **data frame** em **tibble** usa-se a função `as_tibble()`

Converter **tibble** em **data frame** usa-se a função `as_data_frame()`

Cada variável pode ser do modo *numbers(int, dbl)*, *character(chr)*, *logical(lgl)* ou *factor(fctr)*

5. tibble

Descrição dos modos das colunas através da função `glimpse()` - "espiar os dados"

```
# descricao dos dados de sites
tibble::glimpse(si)
```

```
## Rows: 1,163
## Columns: 25
## $ id                  ¤[3m¤[90m<chr>¤[39m¤[23m "amp1001", "amp1002", "amp1003", "amp1004", "amp1005", "amp1006"
## $ reference_number    ¤[3m¤[90m<dbl>¤[39m¤[23m 1001, 1002, 1002, 1002, 1003, 1004, 1005, 1005, 1005, 1006, 1006
## $ species_number      ¤[3m¤[90m<dbl>¤[39m¤[23m 19, 16, 14, 13, 30, 42, 23, 19, 13, 1, 1, 2, 4, 4, 6, 5, 8, 2, 5
## $ record              ¤[3m¤[90m<chr>¤[39m¤[23m "ab", "co", "co", "co", "co", "co", "co", "co", "ab", "ab", "ab"
## $ sampled_habitat     ¤[3m¤[90m<chr>¤[39m¤[23m "fo,ll", "fo,la,ll", "fo,la,ll", "fo,la,ll", "fo,ll,br", "tp,pp,
## $ active_methods      ¤[3m¤[90m<chr>¤[39m¤[23m "as", "as", "as", "as", "as", NA, "as", "as,sb,tr", NA, NA, NA,
## $ passive_methods     ¤[3m¤[90m<chr>¤[39m¤[23m "pt", "pt", "pt", "pt", NA, NA, NA, NA, "pt", "pt", "pt", "pt",
## $ complementary_methods ¤[3m¤[90m<chr>¤[39m¤[23m NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
## $ period              ¤[3m¤[90m<chr>¤[39m¤[23m "mo,da,tw,ni", "mo,da,tw,ni", "mo,da,tw,ni", "mo,da,tw,ni", "mo,
## $ month_start         ¤[3m¤[90m<dbl>¤[39m¤[23m 9, 12, 12, 12, 7, NA, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
## $ year_start          ¤[3m¤[90m<dbl>¤[39m¤[23m 2000, 2007, 2007, 2007, 1988, NA, 2007, 2007, 2007, 2011, 2011,
## $ month_finish        ¤[3m¤[90m<dbl>¤[39m¤[23m 1, 5, 5, 5, 8, NA, 4, 4, 4, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,
## $ year_finish         ¤[3m¤[90m<dbl>¤[39m¤[23m 2002, 2009, 2009, 2009, 2001, NA, 2009, 2009, 2009, 2011, 2011,
```

5. tibble

Descrição dos modos das colunas através da função `glimpse()` - "espiar os dados"

```
# descricao dos dados de especies
tibble::glimpse(sp)
```

```
## Rows: 17,618
## Columns: 10
## $ id          ¤[3m¤[90m<chr>¤[39m¤[23m "amp1001", "amp1001", "amp1009", "amp1009", "amp1009", "amp1009", "amp1009
## $ order       ¤[3m¤[90m<chr>¤[39m¤[23m "Anura", NA, "Anura", NA, "Anura", "Anura", "Anura", "Anura", "Anura", "An
## $ superfamily ¤[3m¤[90m<chr>¤[39m¤[23m NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA
## $ family      ¤[3m¤[90m<chr>¤[39m¤[23m "Hylidae", NA, "Hylidae", NA, "Odontophrynidae", "Phyllomedusidae", "Bufon
## $ subfamily   ¤[3m¤[90m<chr>¤[39m¤[23m "Dendropsophinae", NA, "Lophyohylinae", NA, NA, NA, NA, "Leiuperinae", "Le
## $ genus       ¤[3m¤[90m<chr>¤[39m¤[23m "Dendropsophus", NA, "Corythomantis", NA, "Odontophrynus", "Pithecopus", "
## $ species     ¤[3m¤[90m<chr>¤[39m¤[23m "Dendropsophus soaresi", "Scinax sp.", "Corythomantis greeningi", "Leptoda
## $ valid_name  ¤[3m¤[90m<chr>¤[39m¤[23m "Dendropsophus soaresi", NA, "Corythomantis greeningi", NA, "Odontophrynus
## $ individuals ¤[3m¤[90m<dbl>¤[39m¤[23m 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## $ endemic     ¤[3m¤[90m<dbl>¤[39m¤[23m 0, NA, NA, NA, 0, 0, 0, 0, 0, 0, NA, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, NA, 0,
```

5. tibble

tibble vs data.frame

1. Nunca converte um tipo **character como factor** (corrigido nas versões >R 4.0.x)

```
df ← data.frame(ch = c("a", "b"), nu = 1:2)
str(df)
```

```
## 'data.frame':    2 obs. of  2 variables:
##  $ ch: chr  "a" "b"
##  $ nu: int  1 2
```

```
tb ← tibble::tibble(ch = c("a", "b"), nu = 1:2)
tibble::glimpse(tb)
```

```
## Rows: 2
## Columns: 2
## $ ch ¤[3m¤[90m<chr>¤[39m¤[23m "a", "b"
## $ nu ¤[3m¤[90m<int>¤[39m¤[23m 1, 2
```

5. tibble

tibble vs data.frame

2. A indexação com colchetes retorna um **tibble**

```
df_ch ← df[, 1]
class(df_ch)
```

```
## [1] "character"
```

```
tb_ch ← tb[, 1]
class(tb_ch)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

```
# indexacao pelo nome devolve um vetor
tb_ch ← tb$ch
class(tb_ch)
```

```
## [1] "character"
```

5. tibble

tibble vs data.frame

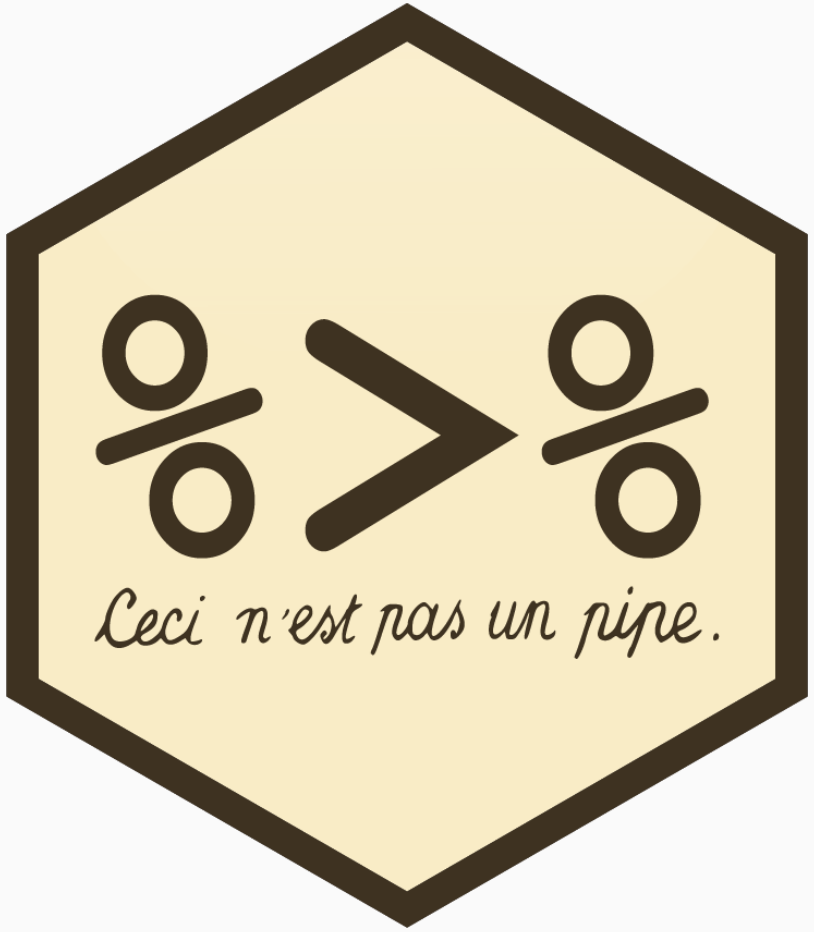
3. Não faz correspondência parcial, retorna **NULL** se a coluna não existe com o nome especificado

```
df$c
```

```
## [1] "a" "b"
```

```
tb$c
```

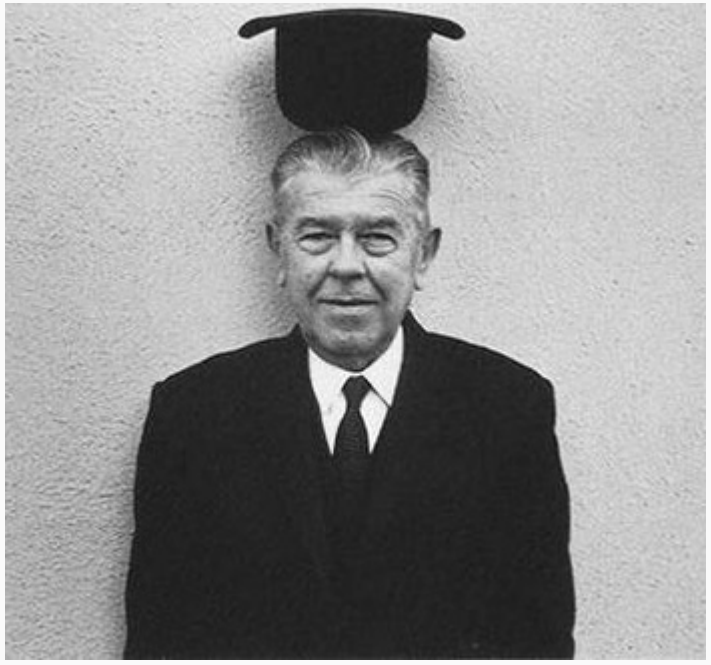
```
## NULL
```



magrittr

6. magrittr (pipe - %>%)

René Magritte (1898-1967)



6. magrittr (pipe - %>%)

Descrição

O operador pipe (%>%) permite o “encadeamento” de várias funções e **não é preciso de objetos** para armazenar resultados intermediários

Essa função torna os códigos em R **mais simples**, pois realizamos **múltiplas operações** em uma **única linha**

Ele captura o **resultado de uma declaração** e o **torna a entrada da próxima declaração**. Podemos pensar como *“EM SEGUIDA FAÇA”*

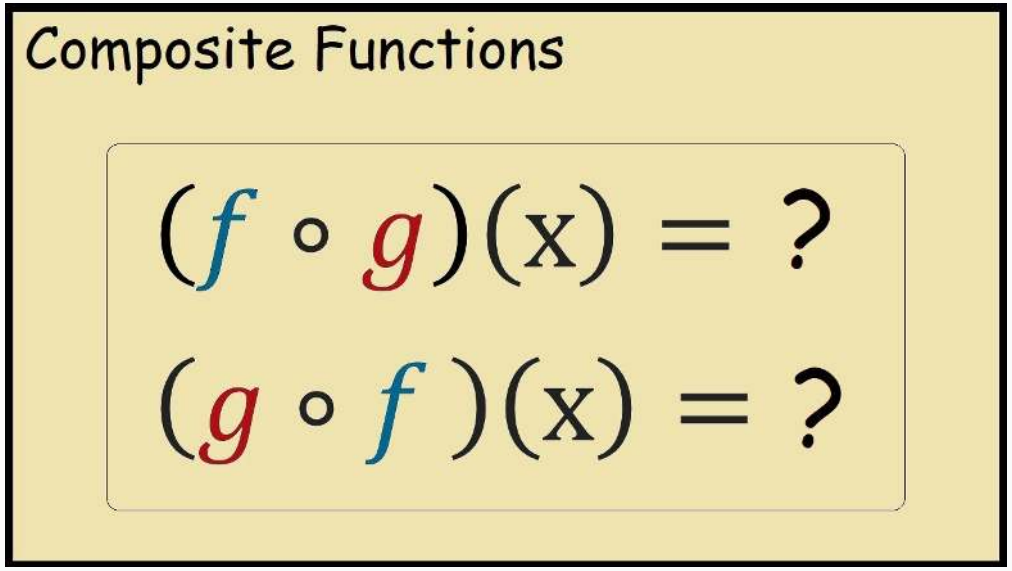
O operador pipe é **%>%**

6. magrittr (pipe - %>%)

Atalho: `Crtl + Shift + M`

```
# sem pipe
sqrt(sum(1:100))
```

```
## [1] 71.06335
```



6. magrittr (pipe - %>%)

Atalho: `Crtl + Shift + M`

```
# sem pipe
sqrt(sum(1:100))
```

```
## [1] 71.06335
```

6. magrittr (pipe - %>%)

Atalho: `Crtl + Shift + M`

```
# sem pipe
sqrt(sum(1:100))
```

```
## [1] 71.06335
```

```
# com pipe
1:100 %>%
  sum() %>%
  sqrt()
```

```
## [1] 71.06335
```

6. magrittr (pipe - %>%)

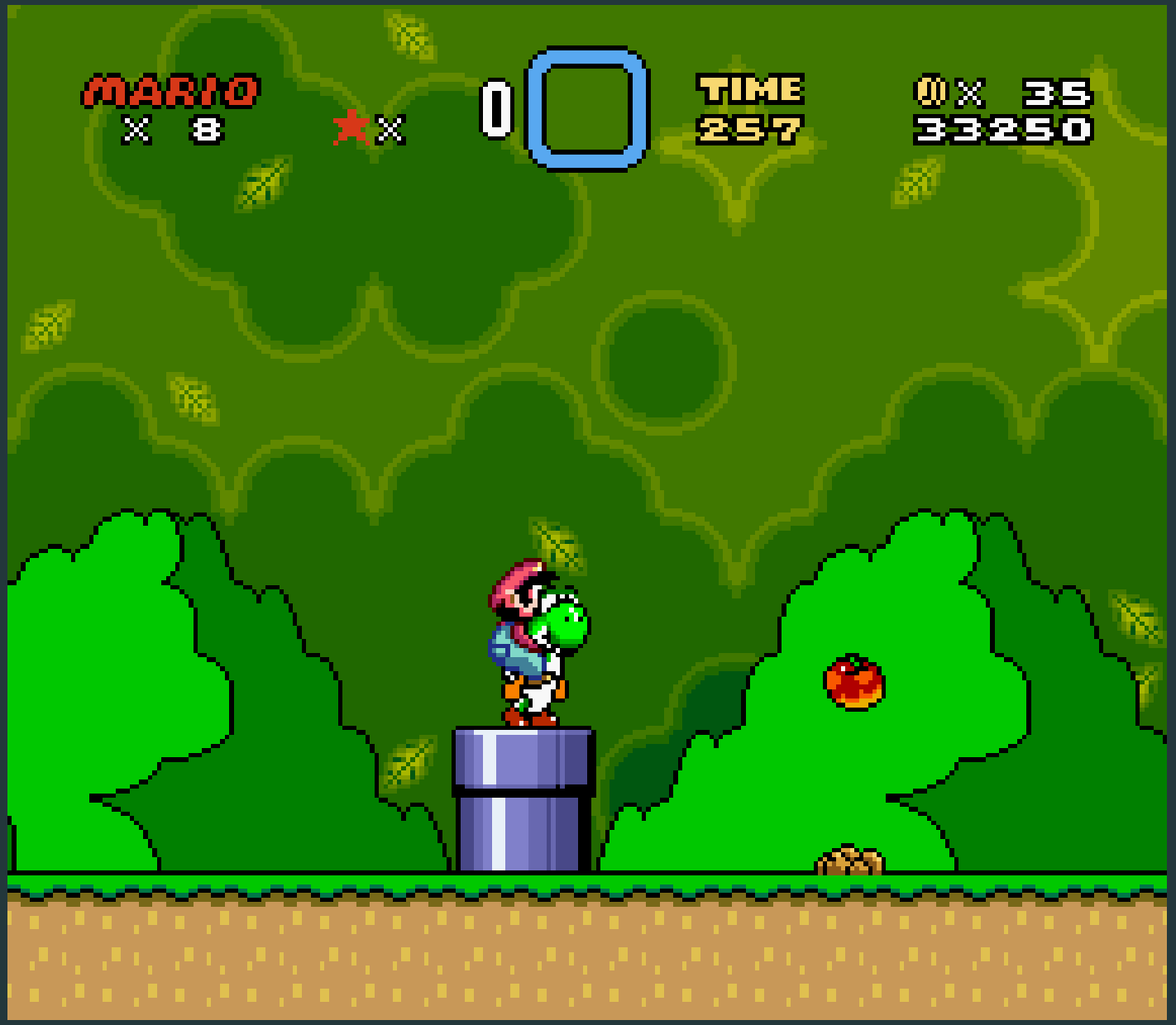
Atalho: `Crtl + Shift + M`

```
# fixar amostragem
set.seed(42)

# sem pipe
ve ← sum(sqrt(sort(log10(rpois(100, 10)))))
ve
```

```
## [1] 99.91426
```

```
# fixar amostragem
set.seed(42)

# com pipe
ve ← rpois(100, 10) %>%
  log10() %>%
  sort() %>%
  sqrt() %>%
  sum()
ve
```

```
## [1] 99.91426
```


MARIO
x 8
x
0
TIME
257
x 35
33250


JORBO

6. magrittr (pipe - %>%)

Outros pipes

maggritr

- `%T>%`: retorna o lado esquerdo em vez do lado direito
- `%$%`: “explode” as variáveis em um quadro de dados
- `%<>%`: permite atribuição usando pipes

Nativo do R:

A partir da versão do R 4.1+ (18/05/2021), o operador pipe se tornou nativo do R

> Tools > Global Options > Code > [x] Use native pipe operator, |> (requires R 4.1+)

- `▷`: pipe nativo do R

Exercícios

Exercício 09

Reescreva cada uma das operações utilizando pipes `%>%`:

```
log10(cumsum(1:100))
```

```
sum(sqrt(abs(rnorm(100))))
```

```
sum(sort(sample(1:10, 10000, rep = TRUE)))
```

05:00

Exercício 09

Solução

```r
# solucao
# 1.
log10(cumsum(1:100))

1:100 %>%
  cumsum %>%
  log10
```

Exercício 09

Solução

```r
# 2.
sum(sort(abs(rnorm(100))))

rnorm(100) %>%
  abs %>%
  sort %>%
  sum
```

Exercício 09

Solução

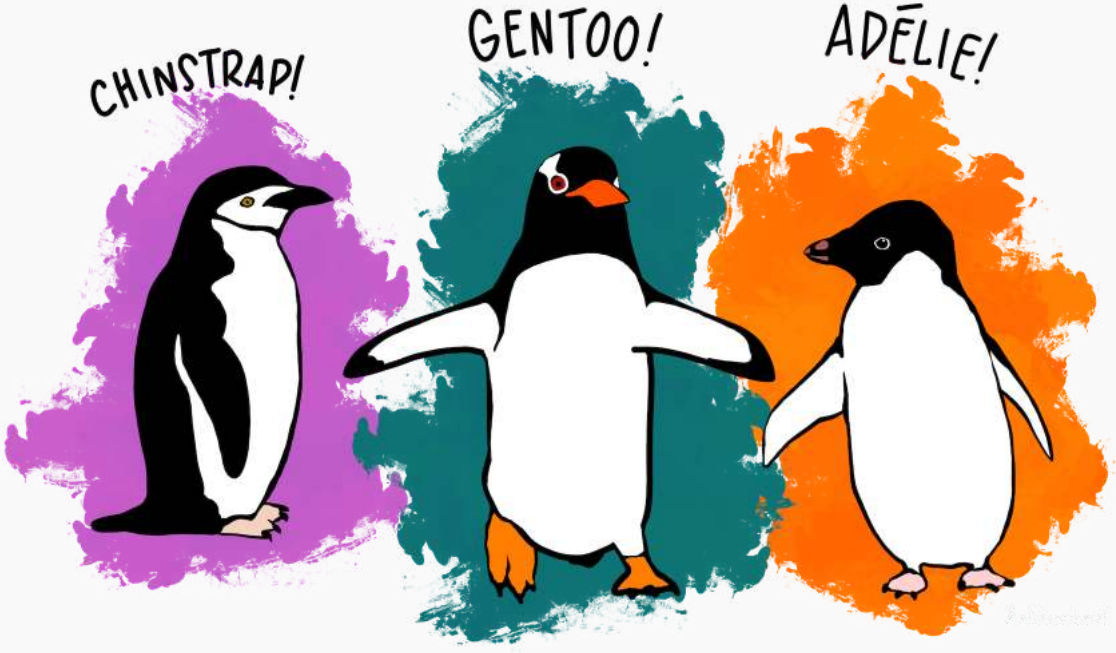
```r
# 3.
sum(sort(sample(1:10, 10000, rep = TRUE)))

1:10 %>%
  sample(10000, rep = TRUE) %>%
  sort %>%
  sum
```

palmerpenguins

Dados de medidas de pinguins chamados `palmerpenguins`

Esses dados foram coletados e disponibilizados pela Dra. Kristen Gorman e pela Palmer Station, Antarctica LTER, membro da Long Term Ecological Research Network

Dois conjuntos de dados: **penguins_raw** (dados brutos) e **penguins** (versão simplificada)

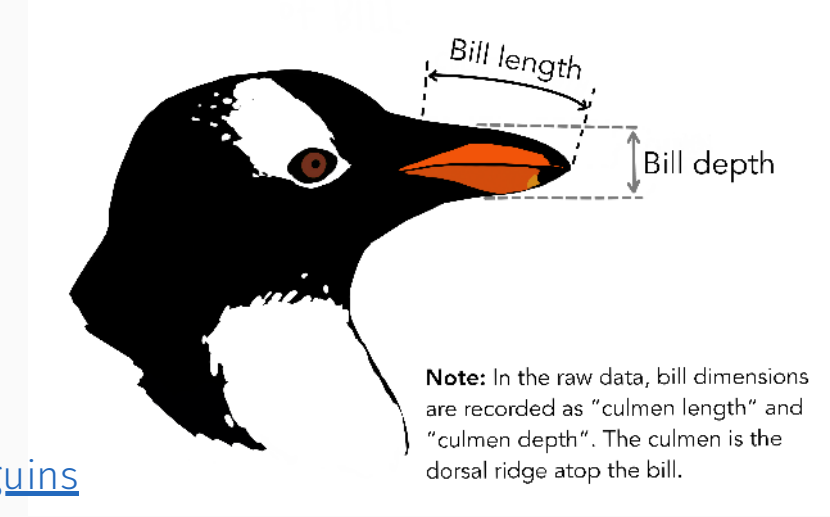


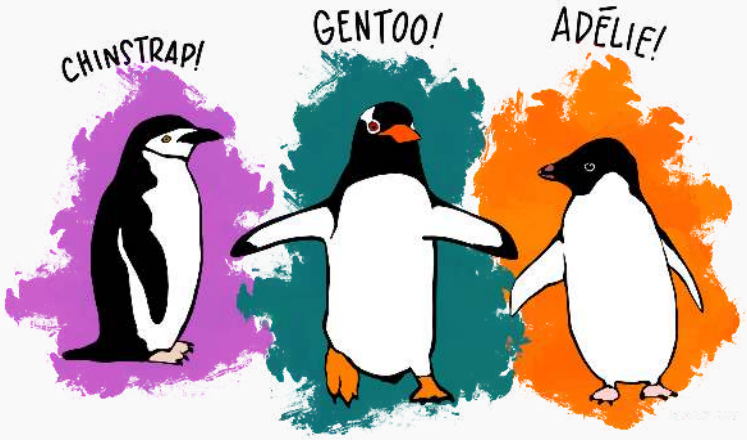
palmerpenguins

palmerpenguins

Dados de medidas de pinguins chamados `palmerpenguins`

```
# instalar
install.packages("palmerpenguins")

# carregar
library(palmerpenguins)

# ajuda dos dados
?penguins
?penguins_raw
```



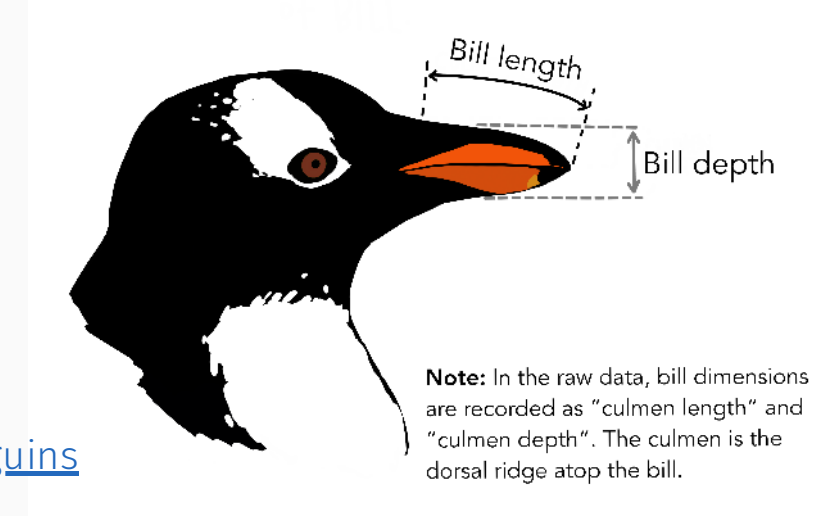


palmerpenguins

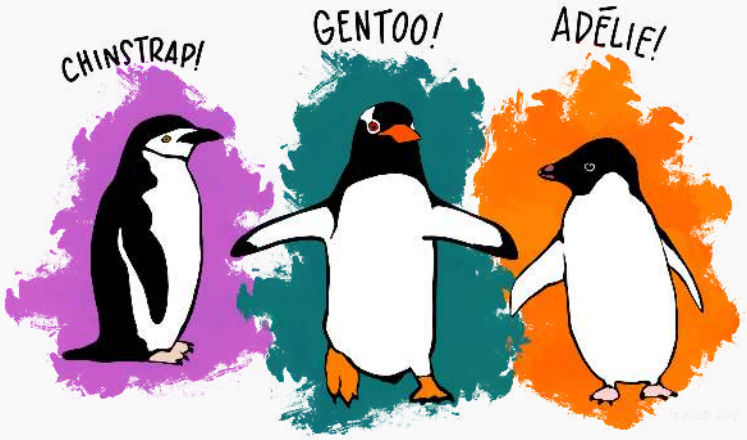
palmerpenguins

Dados de medidas de pinguins chamados `palmerpenguins`

```
# visualizar os dados
penguins
penguins_raw

# glimpse
tibble::glimpse(penguins)
tibble::glimpse(penguins_raw)
```



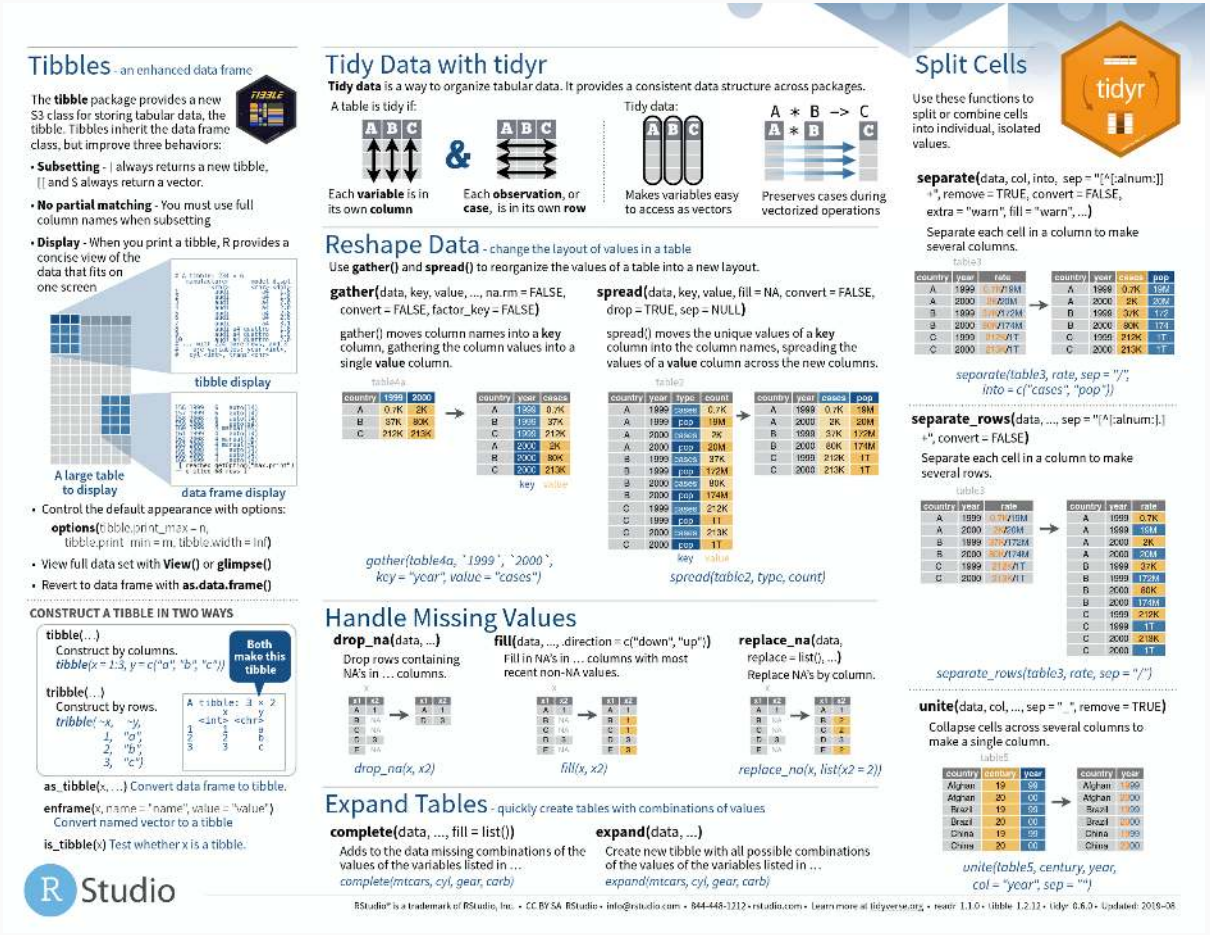
[palmerpenguins]



tidyr

7. tidyr

Data Import Cheatsheet



[Data Import Cheatsheet]

7. tidyr

Descrição

O pacote tidyr tem **função de tornar** um conjunto de dados **tidy** (organizados), sendo esses fáceis de manipular, modelar e visualizar

Um conjunto de dados está **arrumado ou não**, dependendo de como linhas, colunas e células são combinadas com observações, variáveis e valores

Nos dados tidy:

1. Cada variável em uma coluna

2. Cada observação em uma linha

3. Cada valor como uma célula

7. tidyr

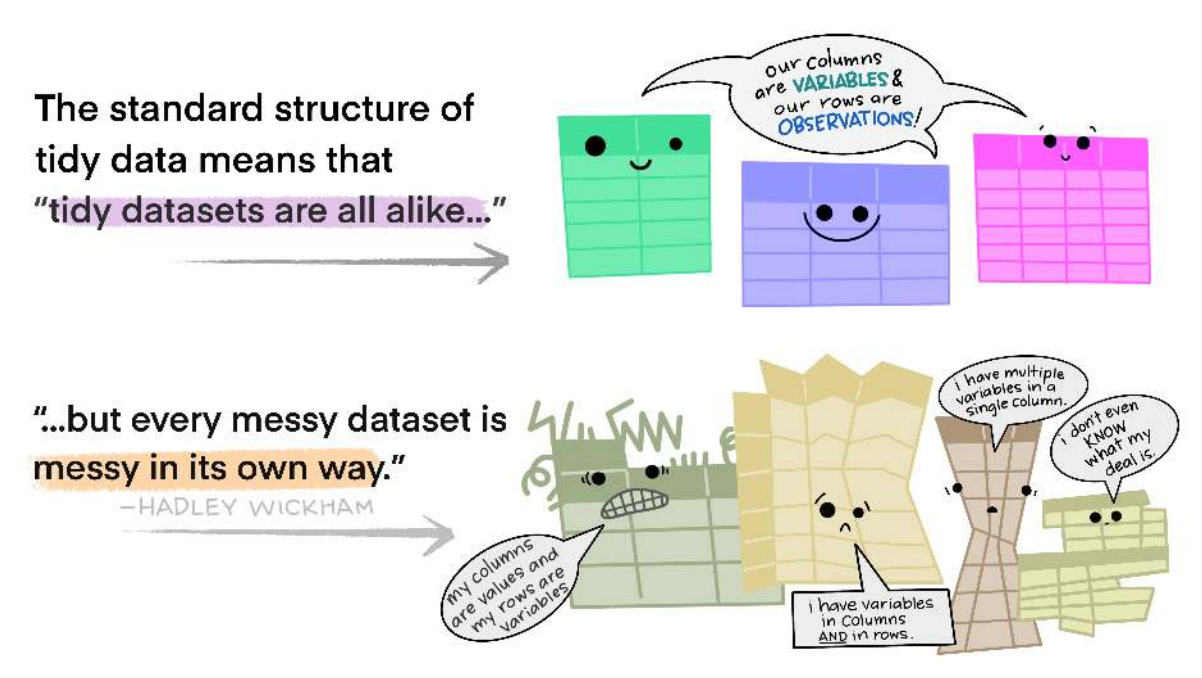
Conjunto de dados 'tidy'

[Wickham & Grolemund (2017)]

7. tidyr

Funções

1. **unite()**: junta dados de múltiplas colunas em uma
2. **separate()**: separa caracteres em múlplica colunas
3. **separate_rows()**: separa caracteres em múlplica colunas e linhas
4. **drop_na()**: retira linhas com NA
5. **replace_na()**: substitui NA
6. **pivot_wider()**: long para wide
7. **pivot_longer()**: wide para long



@allison_horst

7. tidyr

1. `unite()`

Unir as colunas "Region" e "Island" na nova coluna "region_island"

```
# unir colunas
penguins_raw_unir ← tidyr::unite(data = penguins_raw,
                                 col = "region_island",
                                 Region:Island,
                                 sep = ", ",
                                 remove = FALSE)
head(penguins_raw_unir[, c("Region", "Island", "region_island")])
```

```
## ¤[90m# A tibble: 6 × 3¤[39m
##   Region Island    region_island
##   ¤[3m¤[90m<chr>¤[39m¤[23m  ¤[3m¤[90m<chr>¤[39m¤[23m     ¤[3m¤[90m<chr>¤[39m¤[23m
## ¤[90m1¤[39m Anvers Torgersen Anvers, Torgersen
## ¤[90m2¤[39m Anvers Torgersen Anvers, Torgersen
## ¤[90m3¤[39m Anvers Torgersen Anvers, Torgersen
## ¤[90m4¤[39m Anvers Torgersen Anvers, Torgersen
## ¤[90m5¤[39m Anvers Torgersen Anvers, Torgersen
## ¤[90m6¤[39m Anvers Torgersen Anvers, Torgersen
```

7. tidyr

2. `separate()`

Separar a coluna “Stage” nas colunas “stage” e “egg_stage”

```
# separar colunas
penguins_raw_separar ← tidyr::separate(data = penguins_raw,
                                       col = Stage,
                                       into = c("stage", "egg_stage"),
                                       sep = ", ",
                                       remove = FALSE)
head(penguins_raw_separar[, c("Stage", "stage", "egg_stage")])
```

```
## ¤[90m# A tibble: 6 × 3¤[39m
##   Stage               stage egg_stage
##   ¤[3m¤[90m<chr>¤[39m¤[23m             ¤[3m¤[90m<chr>¤[39m¤[23m ¤[3m¤[90m<chr>¤[39m¤[23m
## ¤[90m1¤[39m Adult, 1 Egg Stage Adult 1 Egg Stage
## ¤[90m2¤[39m Adult, 1 Egg Stage Adult 1 Egg Stage
## ¤[90m3¤[39m Adult, 1 Egg Stage Adult 1 Egg Stage
## ¤[90m4¤[39m Adult, 1 Egg Stage Adult 1 Egg Stage
## ¤[90m5¤[39m Adult, 1 Egg Stage Adult 1 Egg Stage
## ¤[90m6¤[39m Adult, 1 Egg Stage Adult 1 Egg Stage
```

3. `separate_rows()`

Separar a coluna "Stage" nas colunas "stage" e "egg_stage" em novas linhas

```
# separar colunas em linhas
penguins_raw_separar_linhas ← tidyr::separate_rows(data = penguins_raw,
                                                   Stage,
                                                   sep = ", ")
head(penguins_raw_separar_linhas[, c("studyName", "Sample Number", "Species",
                                     "Region", "Island", "Stage")])
```

```
## ¤[90m# A tibble: 6 × 6¤[39m
##   studyName `Sample Number` Species                                Region Island    Stage
##   ¤[3m¤[90m<chr>¤[39m¤[23m             ¤[3m¤[90m<dbl>¤[39m¤[23m ¤[3m¤[90m<chr>¤[39m¤[23m
## ¤[90m1¤[39m PAL0708                 1 Adelie Penguin (Pygoscelis adeliae) Anvers Torgersen Adult
## ¤[90m2¤[39m PAL0708                 1 Adelie Penguin (Pygoscelis adeliae) Anvers Torgersen 1 Egg Stage
## ¤[90m3¤[39m PAL0708                 2 Adelie Penguin (Pygoscelis adeliae) Anvers Torgersen Adult
## ¤[90m4¤[39m PAL0708                 2 Adelie Penguin (Pygoscelis adeliae) Anvers Torgersen 1 Egg Stage
## ¤[90m5¤[39m PAL0708                 3 Adelie Penguin (Pygoscelis adeliae) Anvers Torgersen Adult
## ¤[90m6¤[39m PAL0708                 3 Adelie Penguin (Pygoscelis adeliae) Anvers Torgersen 1 Egg Stage
```

4. `drop_na()`

Remove as linhas com NA de todas as colunas

```
# remover linhas com na
penguins_raw_todas_na ← tidyr::drop_na(data = penguins_raw)
head(penguins_raw_todas_na)
```

```
## ¤[90m# A tibble: 6 × 17¤[39m
##   studyName `Sample Number` Species        Region Island  Stage    `Individual ID` `Clutch Completi… `Date Egg` `C
##   ¤[3m¤[90m<chr>¤[39m¤[23m             ¤[3m¤[90m<dbl>¤[39m¤[23m ¤[3m¤[90m<chr>¤[39m¤[23m          ¤[3m¤[90m<c
## ¤[90m1¤[39m PAL0708                 7 Adelie Pengui… Anvers Torger… Adult, … N4A1            No                200
## ¤[90m2¤[39m PAL0708                 8 Adelie Pengui… Anvers Torger… Adult, … N4A2            No                200
## ¤[90m3¤[39m PAL0708                29 Adelie Pengui… Anvers Biscoe  Adult, … N18A1           No                200
## ¤[90m4¤[39m PAL0708                30 Adelie Pengui… Anvers Biscoe  Adult, … N18A2           No                200
## ¤[90m5¤[39m PAL0708                39 Adelie Pengui… Anvers Dream   Adult, … N25A1           No                200
## ¤[90m6¤[39m PAL0809                69 Adelie Pengui… Anvers Torger… Adult, … N32A1           No                200
## ¤[90m# … with 6 more variables: Flipper Length (mm) <dbl>, Body Mass (g) <dbl>, Sex <chr>, Delta 15 N (o/oo) <dbl>
## ¤[90m#   Comments <chr>¤[39m
```

4. `drop_na()`

Remove as linhas com NA da coluna "Comments"

```
# remover linhas com na de uma coluna
penguins_raw_colunas_na ← tidyr::drop_na(data = penguins_raw,
                                         any_of("Comments"))
head(penguins_raw_colunas_na[, "Comments"])
```

```
## ¤[90m# A tibble: 6 × 1¤[39m
##   Comments
##   ¤[3m¤[90m<chr>¤[39m¤[23m
## ¤[90m1¤[39m Not enough blood for isotopes.
## ¤[90m2¤[39m Adult not sampled.
## ¤[90m3¤[39m Nest never observed with full clutch.
## ¤[90m4¤[39m Nest never observed with full clutch.
## ¤[90m5¤[39m No blood sample obtained.
## ¤[90m6¤[39m No blood sample obtained for sexing.
```

7. tidyr

5. `replace_na()`

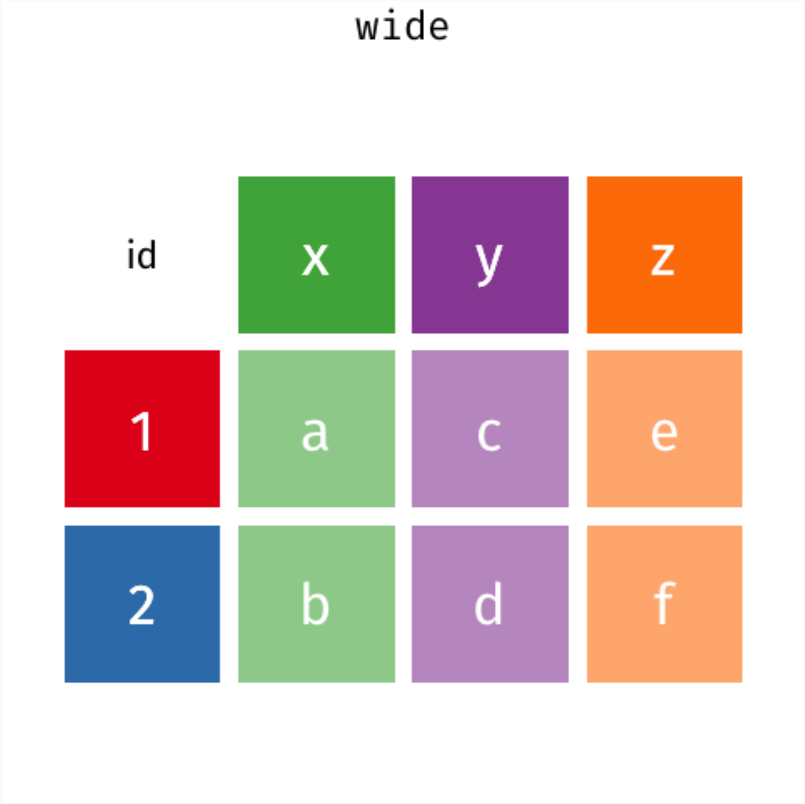
Substituir os NAs da coluna "Unknown" por 0

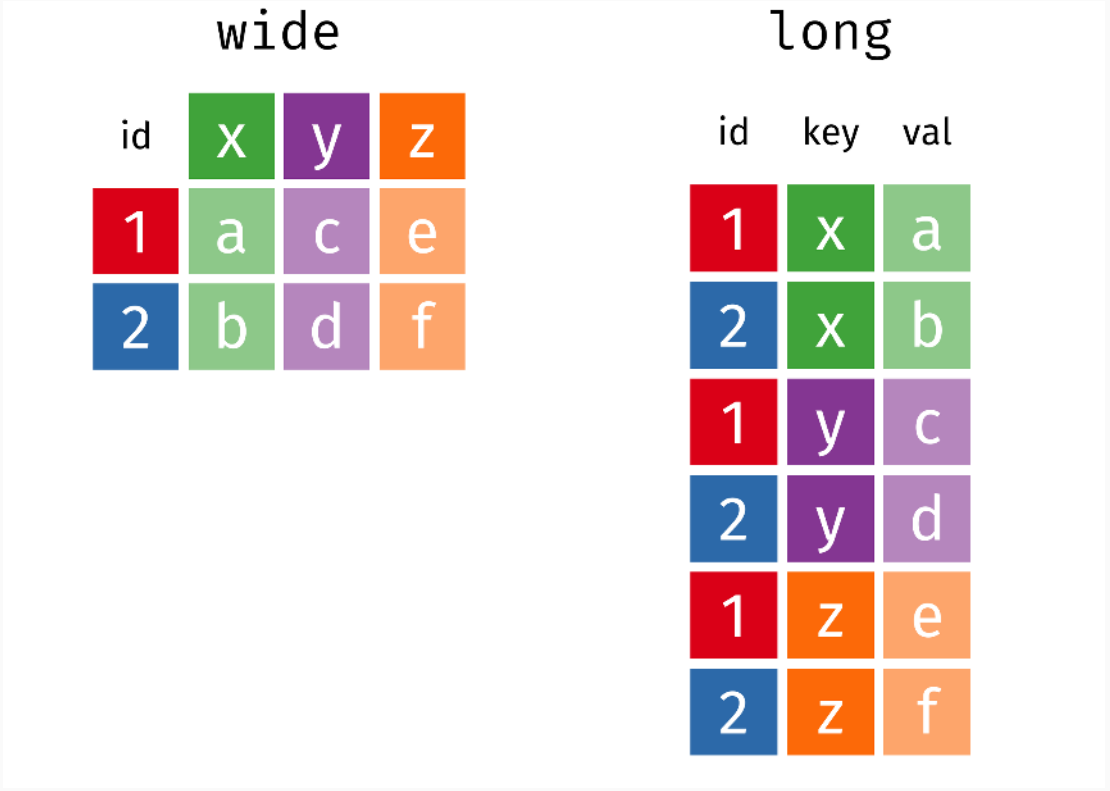
```
# substituir nas por 0 em uma coluna
penguins_raw_subs_na ← tidyr::replace_na(data = penguins_raw,
                                         list(Comments = "Unknown"))
head(penguins_raw_subs_na[, "Comments"])
```

```
## ¤[90m# A tibble: 6 × 1¤[39m
##   Comments
##   ¤[3m¤[90m<chr>¤[39m¤[23m
## ¤[90m1¤[39m Not enough blood for isotopes.
## ¤[90m2¤[39m Unknown
## ¤[90m3¤[39m Unknown
## ¤[90m4¤[39m Adult not sampled.
## ¤[90m5¤[39m Unknown
## ¤[90m6¤[39m Unknown
```

Pivotar os dados

`pivot_longer()` | `pivot_wider()`





7. tidyr

6. `pivot_wider()`

Long para wide

1. **names_from**: variável categórica que definirá os nomes das colunas
2. **values_from**: variável numérica que preencherá os dados
3. **values_fill**: valor para preencher os NAs

```
# long para wide
penguins_raw_pivot_wider ← tidyr::pivot_wider(data = penguins_raw[, c(2, 3, 13)],
                                              names_from = Species,
                                              values_from = `Body Mass (g)`)
head(penguins_raw_pivot_wider)
```

```
## ¤[90m# A tibble: 6 × 4¤[39m
##   `Sample Number` `Adelie Penguin (Pygoscelis adeliae)` `Gentoo penguin (Pygoscelis papua)` `Chinstrap penguin (Py
##             ¤[3m¤[90m<dbl>¤[39m¤[23m                                ¤[3m¤[90m<dbl>¤[39m¤[23m
## ¤[90m1¤[39m               1                              ¤[4m3¤[24m750                          ¤[4m4¤[
## ¤[90m2¤[39m               2                              ¤[4m3¤[24m800                          ¤[4m5¤[
## ¤[90m3¤[39m               3                              ¤[4m3¤[24m250                          ¤[4m4¤[
## ¤[90m4¤[39m               4                                ¤[31mNA¤[39m                         ¤[4m5¤
```

7. `pivot_longer()`

Wide para long

1. **names_to**: nome da coluna que receberá os nomes
2. **values_to**: nome da coluna que receberá os valores

```
# wide para long
penguins_raw_pivot_longer ← tidyr::pivot_longer(data = penguins_raw[, c(2, 3, 10:13)],
                                                cols = `Culmen Length (mm)`:`Body Mass (g)`,
                                                names_to = "medidas",
                                                values_to = "valores")
head(penguins_raw_pivot_longer)
```

```
## ¤[90m# A tibble: 6 × 4¤[39m
##   `Sample Number` Species                                  medidas            valores
##             ¤[3m¤[90m<dbl>¤[39m¤[23m ¤[3m¤[90m<chr>¤[39m¤[23m                                                ¤[3m¤[90m<chr>¤[39m¤
## ¤[90m1¤[39m               1 Adelie Penguin (Pygoscelis adeliae) Culmen Length (mm)     39.1
## ¤[90m2¤[39m               1 Adelie Penguin (Pygoscelis adeliae) Culmen Depth (mm)      18.7
## ¤[90m3¤[39m               1 Adelie Penguin (Pygoscelis adeliae) Flipper Length (mm)   181
## ¤[90m4¤[39m               1 Adelie Penguin (Pygoscelis adeliae) Body Mass (g)        ¤[4m3¤[24m750
```

Exercícios

Exercício 10

Separe os nomes comuns dos nomes científicos dos pinguins dos dados `penguins_raw`

03:00

Exercício 10

Solução

```
# exercicio 10
penguins_raw_sep ← penguins_raw %>%
  tidyr::separate(col = "Species",
                  into = c("common_names", "scientific_names"),
                  sep = "[(]",
                  remove = FALSE)
head(penguins_raw_sep)
```

```
## ¤[90m# A tibble: 6 × 19¤[39m
##   studyName `Sample Number` Species          common_names   scientific_names  Region Island  Stage    `Individual
##   ¤[3m¤[90m<chr>¤[39m¤[23m               ¤[3m¤[90m<dbl>¤[39m¤[23m ¤[3m¤[90m<chr>¤[39m¤[23m           ¤[3m¤[90m
## ¤[90m1¤[39m PAL0708                 1 Adelie Penguin … ¤[90m"¤[39mAdelie Pengu… Pygoscelis adeli… Anvers Torger… A
## ¤[90m2¤[39m PAL0708                 2 Adelie Penguin … ¤[90m"¤[39mAdelie Pengu… Pygoscelis adeli… Anvers Torger… A
## ¤[90m3¤[39m PAL0708                 3 Adelie Penguin … ¤[90m"¤[39mAdelie Pengu… Pygoscelis adeli… Anvers Torger… A
## ¤[90m4¤[39m PAL0708                 4 Adelie Penguin … ¤[90m"¤[39mAdelie Pengu… Pygoscelis adeli… Anvers Torger… A
## ¤[90m5¤[39m PAL0708                 5 Adelie Penguin … ¤[90m"¤[39mAdelie Pengu… Pygoscelis adeli… Anvers Torger… A
## ¤[90m6¤[39m PAL0708                 6 Adelie Penguin … ¤[90m"¤[39mAdelie Pengu… Pygoscelis adeli… Anvers Torger… A
## ¤[90m# … with 8 more variables: Culmen Length (mm) <dbl>, Culmen Depth (mm) <dbl>, Flipper Length (mm) <dbl>, Body
## ¤[90m#   Delta 15 N (o/oo) <dbl>, Delta 13 C (o/oo) <dbl>, Comments <chr>¤[39m
```

Exercício 11

Calcule a massa média dos indivíduos das espécies de penguins por ilha. Dicas: `tidyr::replace_na()` e `tidyr::pivot_wider()`

10:00

Exercício 11

Solução

```
# exercicio 11
penguins_raw_pivot_wider ← penguins_raw[, c("Island", "Species", "Body Mass (g)")] %>%
  tidyr::replace_na(list(`Body Mass (g)` = 0)) %>%
  tidyr::pivot_wider(names_from = c(Species),
                     values_from = `Body Mass (g)`,
                     values_fn = mean)
head(penguins_raw_pivot_wider)
```

```
## ¤[90m# A tibble: 3 × 4¤[39m
##   Island    `Adelie Penguin (Pygoscelis adeliae)` `Gentoo penguin (Pygoscelis papua)` `Chinstrap penguin (Pygoscel
##   ¤[3m¤[90m<chr>¤[39m¤[23m                                ¤[3m¤[90m<dbl>¤[39m¤[23m
## ¤[90m1¤[39m Torgersen                          ¤[4m3¤[24m635.                                  ¤[31mNA¤[39m
## ¤[90m2¤[39m Biscoe                             ¤[4m3¤[24m710.                                ¤[4m5¤[24m035.
## ¤[90m3¤[39m Dream                              ¤[4m3¤[24m688.                                  ¤[31mNA¤[39m
```

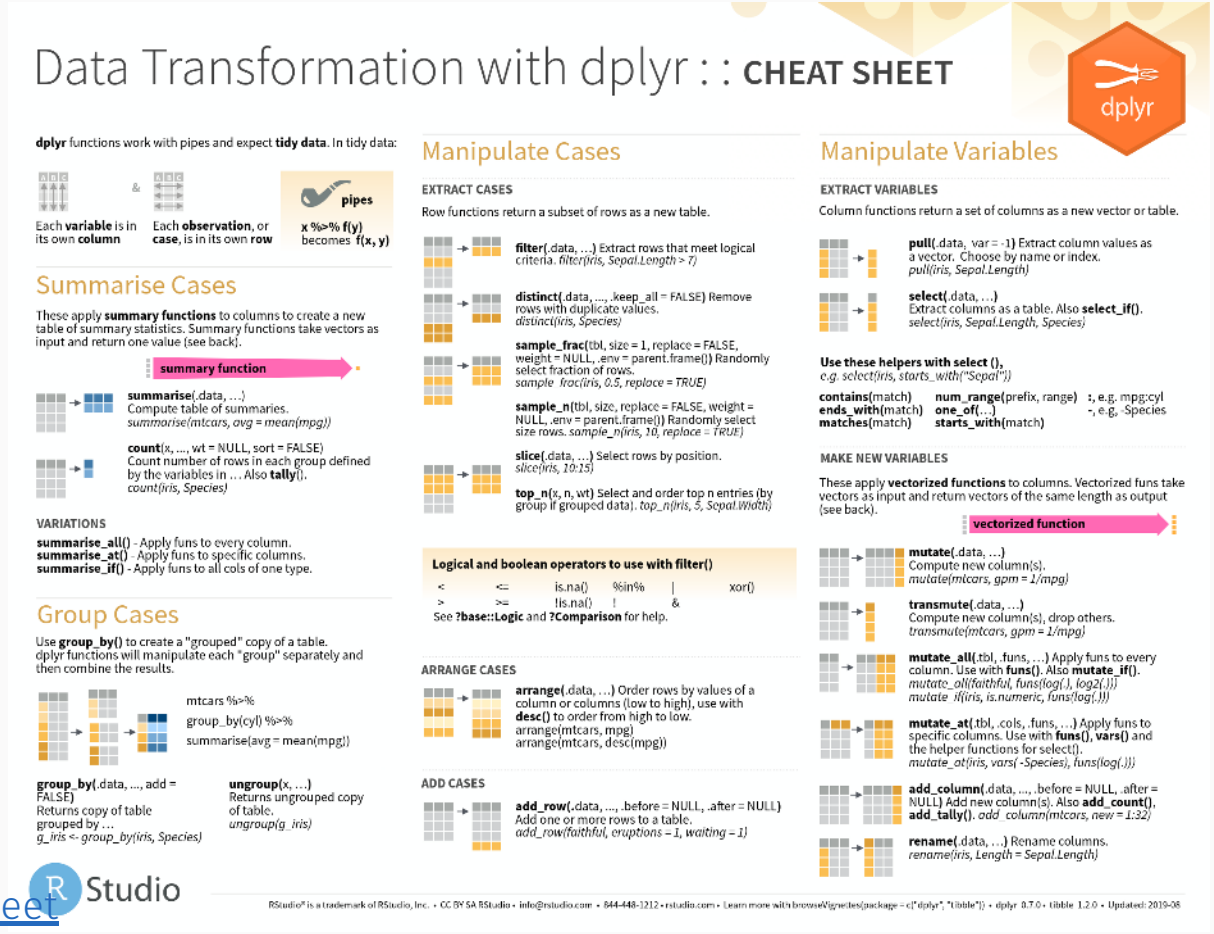
Dúvidas?

dplyr

@allison_horst

8. dplyr

Data Transformation Cheatsheet



Data Transformation with dplyr : : CHEAT SHEET

dplyr

dplyr functions work with pipes and expect **tidy data**. In tidy data:

Each **variable** is in its own **column** & Each **observation**, or **case**, is in its own **row**

pipes
x %>% f(y) becomes **f(x, y)**

Summarise Cases

These apply **summary functions** to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value (see back).

summary function

summarise(.data, …)
Compute table of summaries.
summarise(mtcars, avg = mean(mpg))

count(x, ..., wt = NULL, sort = FALSE)
Count number of rows in each group defined by the variables in … Also **tally**().
count(iris, Species)

VARIATIONS

summarise_all() - Apply funs to every column.
summarise_at() - Apply funs to specific columns.
summarise_if() - Apply funs to all cols of one type.

Group Cases

Use **group_by()** to create a "grouped" copy of a table. dplyr functions will manipulate each "group" separately and then combine the results.

mtcars %>%
group_by(cyl) %>%
summarise(avg = mean(mpg))

group_by(.data, ..., add = FALSE)
Returns copy of table grouped by …
g_iris <- group_by(iris, Species)

ungroup(x, …)
Returns ungrouped copy of table.
ungroup(g_iris)

Manipulate Cases

EXTRACT CASES

Row functions return a subset of rows as a new table.

filter(.data, …) Extract rows that meet logical criteria. *filter(iris, Sepal.Length > 7)*

distinct(.data, ..., .keep_all = FALSE) Remove rows with duplicate values.
distinct(iris, Species)

sample_frac(tbl, size = 1, replace = FALSE, weight = NULL, .env = parent.frame()) Randomly select fraction of rows.
sample_frac(iris, 0.5, replace = TRUE)

sample_n(tbl, size, replace = FALSE, weight = NULL, .env = parent.frame()) Randomly select size rows. *sample_n(iris, 10, replace = TRUE)*

slice(.data, …) Select rows by position.
slice(iris, 10:15)

top_n(x, n, wt) Select and order top n entries (by group if grouped data). *top_n(iris, 5, Sepal.Width)*

Logical and boolean operators to use with filter()

<	<=	is.na()	%in%	|	xor()
>	>=	!is.na()	!	&	

See **?base::Logic** and **?Comparison** for help.

ARRANGE CASES

arrange(.data, …) Order rows by values of a column or columns (low to high), use with **desc()** to order from high to low.
arrange(mtcars, mpg)
arrange(mtcars, desc(mpg))

ADD CASES

add_row(.data, ..., .before = NULL, .after = NULL) Add one or more rows to a table.
add_row(faithful, eruptions = 1, waiting = 1)

Manipulate Variables

EXTRACT VARIABLES

Column functions return a set of columns as a new vector or table.

pull(.data, var = -1) Extract column values as a vector. Choose by name or index.
pull(iris, Sepal.Length)

select(.data, …)
Extract columns as a table. Also **select_if()**.
select(iris, Sepal.Length, Species)

Use these helpers with select (),
e.g. select(iris, starts_with("Sepal"))

contains(match) **num_range**(prefix, range) :, e.g. mpg:cyl
ends_with(match) **one_of**(…) -, e.g, -Species
matches(match) **starts_with**(match)

MAKE NEW VARIABLES

These apply **vectorized functions** to columns. Vectorized funs take vectors as input and return vectors of the same length as output (see back).

vectorized function

mutate(.data, …)
Compute new column(s).
mutate(mtcars, gpm = 1/mpg)

transmute(.data, …)
Compute new column(s), drop others.
transmute(mtcars, gpm = 1/mpg)

mutate_all(.tbl, .funs, …) Apply funs to every column. Use with **funs()**. Also **mutate_if()**.
mutate_all(faithful, funs(log(.), log2(.)))
mutate_if(iris, is.numeric, funs(log(.)))

mutate_at(.tbl, .cols, .funs, …) Apply funs to specific columns. Use with **funs()**, **vars()** and the helper functions for select().
mutate_at(iris, vars(-Species), funs(log(.)))

add_column(.data, ..., .before = NULL, .after = NULL) Add new column(s). Also **add_count()**, **add_tally()**. *add_column(mtcars, new = 1:32)*

rename(.data, …) Rename columns.
rename(iris, Length = Sepal.Length)

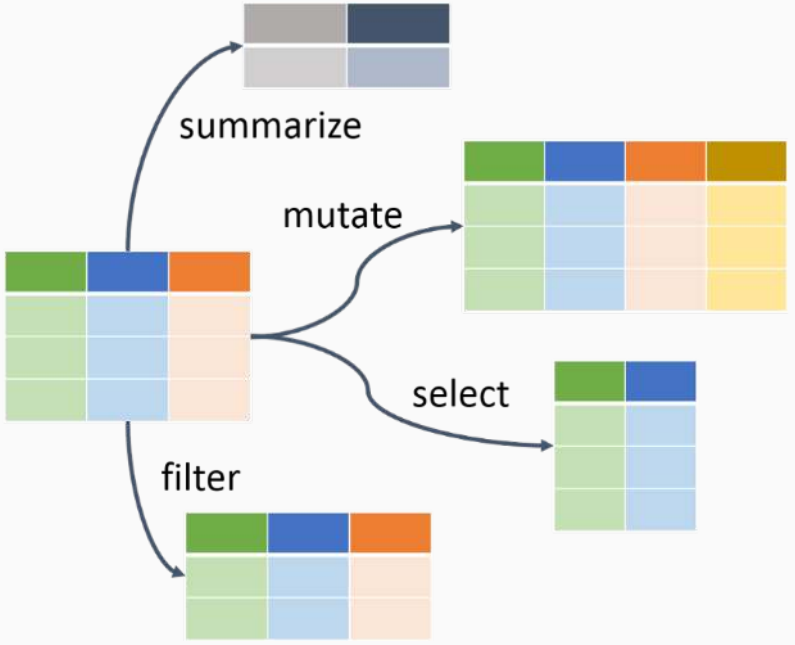
RStudio

Data Transformation Cheatsheet

8. dplyr

Descrição

O **dplyr** é um pacote que **facilita** o trabalho com dados, com uma **gramática de manipulação** de dados **simples e flexível** (filtragem, reordenamento, seleção, ...)



Funções

Sua gramática simples contém **funções verbais** para manipulação de dados

- Verbos: `mutate()`, `select()`, `filter()`, `arrange()`, `summarise()`, `slice()`, `rename()`, etc.
- Replicação: `across()`, `if_any()`, `if_all()`, `where()`, `starts_with()`, `ends_with()`, `contains()`, etc.
- Agrupamento: `group_by()` e `ungroup()`
- Junções: `inner_join()`, `full_join()`, `left_join()`, `right_join()`, etc.
- Combinações: `bind_rows()` e `bind_cols()`
- Resumos, contagem e seleção: `n()`, `n_distinct()`, `first()`, `last()`, `nth()`, etc.

8. dplyr

Funções

Exemplos

- `relocate()`: muda a ordem das colunas
- `rename()`: muda o nome das colunas
- `select()`: seleciona colunas pelo nome ou posição
- `pull()`: seleciona uma coluna como vetor
- `mutate()`: adiciona novas colunas ou resultados em colunas existentes
- `arrange()`: reordena as linhas com base nos valores de colunas
- `filter()`: seleciona linhas com base em valores de colunas
- `slice()`: seleciona linhas de diferente formas
- `distinct()`: remove linhas com valores repetidos com base nos valores de colunas
- `count()`: conta observações para um grupo
- `group_by()`: agrupa linhas pelos valores das colunas
- `summarise()`: resume os dados através de funções considerando valores das colunas
- `*_join()`: funções que juntam dados de duas tabelas através de uma coluna chave

8. dplyr

Sintaxe

O **tibble** é sempre o **primeiro argumento** das funções verbais

Todas seguem a mesma sintaxe:

1. tibble
2. operador pipe
3. nome da função verbal com os argumentos entre parênteses

As funções verbais **não modificam** o tibble original

```
# sintaxe
tb_dplyr ← tb %>%
  funcao_verbal1(argumento1, argumento2, ...) %>%
  funcao_verbal2(argumento1, argumento2, ...) %>%
  funcao_verbal3(argumento1, argumento2, ...)
```

8. dplyr

1. `relocate()`

Reordena as colunas por nome ou posição

```
# reordenar colunas - nome
penguins_relocate_col ← penguins %>%
  dplyr::relocate(sex, year, .after = island)
head(penguins_relocate_col)
```

```
## ¤[90m# A tibble: 6 × 8¤[39m
##   species island    sex     year bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##   ¤[3m¤[90m<fct>¤[39m¤[23m   ¤[3m¤[90m<fct>¤[39m¤[23m     ¤[3m¤[90m<fct>¤[39m¤[23m  ¤[3m¤[90m<int>¤[39m¤[23m
## ¤[90m1¤[39m Adelie  Torgersen male    ¤[4m2¤[24m007           39.1          18.7               181        ¤[4m3¤[
## ¤[90m2¤[39m Adelie  Torgersen female  ¤[4m2¤[24m007           39.5          17.4               186        ¤[4m3¤[
## ¤[90m3¤[39m Adelie  Torgersen female  ¤[4m2¤[24m007           40.3          18                 195        ¤[4m3¤[
## ¤[90m4¤[39m Adelie  Torgersen ¤[31mNA¤[39m      ¤[4m2¤[24m007           ¤[31mNA¤[39m            ¤[31mNA¤[39m
## ¤[90m5¤[39m Adelie  Torgersen female  ¤[4m2¤[24m007           36.7          19.3               193        ¤[4m3¤[
## ¤[90m6¤[39m Adelie  Torgersen male    ¤[4m2¤[24m007           39.3          20.6               190        ¤[4m3¤[
```

8. dplyr

1. `relocate()`

Reordena as colunas por nome ou posição

```
# reordenar colunas - posicao
penguins_relocate_ncol ← penguins %>%
  dplyr::relocate(sex, year, .after = 2)
head(penguins_relocate_ncol)
```

```
## ¤[90m# A tibble: 6 × 8¤[39m
##   species island    sex     year bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##   ¤[3m¤[90m<fct>¤[39m¤[23m   ¤[3m¤[90m<fct>¤[39m¤[23m     ¤[3m¤[90m<fct>¤[39m¤[23m  ¤[3m¤[90m<int>¤[39m¤[23m
## ¤[90m1¤[39m Adelie  Torgersen male    ¤[4m2¤[24m007          39.1          18.7               181        ¤[4m3¤[
## ¤[90m2¤[39m Adelie  Torgersen female  ¤[4m2¤[24m007          39.5          17.4               186        ¤[4m3¤[
## ¤[90m3¤[39m Adelie  Torgersen female  ¤[4m2¤[24m007          40.3          18                 195        ¤[4m3¤[
## ¤[90m4¤[39m Adelie  Torgersen ¤[31mNA¤[39m      ¤[4m2¤[24m007           ¤[31mNA¤[39m            ¤[31mNA¤[39m
## ¤[90m5¤[39m Adelie  Torgersen female  ¤[4m2¤[24m007          36.7          19.3               193        ¤[4m3¤[
## ¤[90m6¤[39m Adelie  Torgersen male    ¤[4m2¤[24m007          39.3          20.6               190        ¤[4m3¤[
```

2. `rename()`

Renomeia as colunas

```
# renomear colunas
penguins_rename ← penguins %>%
  dplyr::rename(bill_length = bill_length_mm,
                bill_depth = bill_depth_mm,
                flipper_length = flipper_length_mm,
                body_mass = body_mass_g)
head(penguins_rename)
```

```
## ¤[90m# A tibble: 6 × 8¤[39m
##   species island    bill_length bill_depth flipper_length body_mass sex     year
##   ¤[3m¤[90m<fct>¤[39m¤[23m   ¤[3m¤[90m<fct>¤[39m¤[23m           ¤[3m¤[90m<dbl>¤[39m¤[23m      ¤[3m¤[90m<dbl>¤[3
## ¤[90m1¤[39m Adelie  Torgersen        39.1       18.7            181      ¤[4m3¤[24m750 male    ¤[4m2¤[24m007
## ¤[90m2¤[39m Adelie  Torgersen        39.5       17.4            186      ¤[4m3¤[24m800 female  ¤[4m2¤[24m007
## ¤[90m3¤[39m Adelie  Torgersen        40.3       18              195      ¤[4m3¤[24m250 female  ¤[4m2¤[24m007
## ¤[90m4¤[39m Adelie  Torgersen        ¤[31mNA¤[39m         ¤[31mNA¤[39m                ¤[31mNA¤[39m        ¤[31mNA
## ¤[90m5¤[39m Adelie  Torgersen        36.7       19.3            193      ¤[4m3¤[24m450 female  ¤[4m2¤[24m007
## ¤[90m6¤[39m Adelie  Torgersen        39.3       20.6            190      ¤[4m3¤[24m650 male    ¤[4m2¤[24m007
```

2. `rename()`

Renomeia as várias colunas

```
# renomear todas as colunas
penguins_rename_with ← penguins %>%
  dplyr::rename_with(toupper)
head(penguins_rename_with)
```

```
## ¤[90m# A tibble: 6 × 8¤[39m
##   SPECIES ISLAND    BILL_LENGTH_MM BILL_DEPTH_MM FLIPPER_LENGTH_MM BODY_MASS_G SEX     YEAR
##   ¤[3m¤[90m<fct>¤[39m¤[23m   ¤[3m¤[90m<fct>¤[39m¤[23m              ¤[3m¤[90m<dbl>¤[39m¤[23m         ¤[3m¤[90m<d
## ¤[90m1¤[39m Adelie  Torgersen           39.1          18.7                181        ¤[4m3¤[24m750 male    ¤[4m2¤[
## ¤[90m2¤[39m Adelie  Torgersen           39.5          17.4                186        ¤[4m3¤[24m800 female  ¤[4m2¤[
## ¤[90m3¤[39m Adelie  Torgersen           40.3          18                  195        ¤[4m3¤[24m250 female  ¤[4m2¤[
## ¤[90m4¤[39m Adelie  Torgersen           ¤[31mNA¤[39m            ¤[31mNA¤[39m                ¤[31mNA¤[39m
## ¤[90m5¤[39m Adelie  Torgersen           36.7          19.3                193        ¤[4m3¤[24m450 female  ¤[4m2¤[
## ¤[90m6¤[39m Adelie  Torgersen           39.3          20.6                190        ¤[4m3¤[24m650 male    ¤[4m2¤[
```

3. `select()`

Seleciona colunas pela posição ou pelo nome

```
# selecionar colunas por posicao
penguins_select_position ← penguins %>%
  dplyr::select(3:6)
head(penguins_select_position)
```

```
## ¤[90m# A tibble: 6 × 4¤[39m
##   bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##            ¤[3m¤[90m<dbl>¤[39m¤[23m         ¤[3m¤[90m<dbl>¤[39m¤[23m             ¤[3m¤[90m<int>¤[39m¤[23m
## ¤[90m1¤[39m           39.1          18.7                181        ¤[4m3¤[24m750
## ¤[90m2¤[39m           39.5          17.4                186        ¤[4m3¤[24m800
## ¤[90m3¤[39m           40.3          18                  195        ¤[4m3¤[24m250
## ¤[90m4¤[39m           ¤[31mNA¤[39m            ¤[31mNA¤[39m                  ¤[31mNA¤[39m          ¤[31mNA¤[39m
## ¤[90m5¤[39m           36.7          19.3                193        ¤[4m3¤[24m450
## ¤[90m6¤[39m           39.3          20.6                190        ¤[4m3¤[24m650
```

3. `select()`

Seleciona colunas pela posição ou pelo nome

```
# selecionar colunas por nomes
penguins_select_names ← penguins %>%
  dplyr::select(bill_length_mm:body_mass_g)
head(penguins_select_names)
```

```
## ¤[90m# A tibble: 6 × 4¤[39m
##   bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##            ¤[3m¤[90m<dbl>¤[39m¤[23m         ¤[3m¤[90m<dbl>¤[39m¤[23m             ¤[3m¤[90m<int>¤[39m¤[23m
## ¤[90m1¤[39m           39.1          18.7                181        ¤[4m3¤[24m750
## ¤[90m2¤[39m           39.5          17.4                186        ¤[4m3¤[24m800
## ¤[90m3¤[39m           40.3          18                  195        ¤[4m3¤[24m250
## ¤[90m4¤[39m           ¤[31mNA¤[39m            ¤[31mNA¤[39m                  ¤[31mNA¤[39m          ¤[31mNA¤[39m
## ¤[90m5¤[39m           36.7          19.3                193        ¤[4m3¤[24m450
## ¤[90m6¤[39m           39.3          20.6                190        ¤[4m3¤[24m650
```

3. `select()`

Remove as colunas pela posição ou pelo nome

```
# remover colunas pelo nome
penguins_select_names_remove ← penguins %>%
  dplyr::select(-(bill_length_mm:body_mass_g))
head(penguins_select_names_remove)
```

```
## ¤[90m# A tibble: 6 × 4¤[39m
##   species island    sex     year
##   ¤[3m¤[90m<fct>¤[39m¤[23m   ¤[3m¤[90m<fct>¤[39m¤[23m     ¤[3m¤[90m<fct>¤[39m¤[23m  ¤[3m¤[90m<int>¤[39m¤[23m
## ¤[90m1¤[39m Adelie  Torgersen male    ¤[4m2¤[24m007
## ¤[90m2¤[39m Adelie  Torgersen female  ¤[4m2¤[24m007
## ¤[90m3¤[39m Adelie  Torgersen female  ¤[4m2¤[24m007
## ¤[90m4¤[39m Adelie  Torgersen ¤[31mNA¤[39m      ¤[4m2¤[24m007
## ¤[90m5¤[39m Adelie  Torgersen female  ¤[4m2¤[24m007
## ¤[90m6¤[39m Adelie  Torgersen male    ¤[4m2¤[24m007
```

3. `select()`

Seleciona colunas por um padrão

```
# selecionar colunas por padrao - starts_with(), ends_with() e contains()
penguins_select_contains ← penguins %>%
  dplyr::select(contains("_mm"))
head(penguins_select_contains)
```

```
## ¤[90m# A tibble: 6 × 3¤[39m
##   bill_length_mm bill_depth_mm flipper_length_mm
##            ¤[3m¤[90m<dbl>¤[39m¤[23m         ¤[3m¤[90m<dbl>¤[39m¤[23m             ¤[3m¤[90m<int>¤[39m¤[23m
## ¤[90m1¤[39m           39.1          18.7                181
## ¤[90m2¤[39m           39.5          17.4                186
## ¤[90m3¤[39m           40.3          18                  195
## ¤[90m4¤[39m           ¤[31mNA¤[39m            ¤[31mNA¤[39m                  ¤[31mNA¤[39m
## ¤[90m5¤[39m           36.7          19.3                193
## ¤[90m6¤[39m           39.3          20.6                190
```

4. `pull()`

Seleciona uma coluna como vetor

```
# coluna para vetor
penguins_select_pull ← penguins %>%
  dplyr::pull(bill_length_mm)
head(penguins_select_pull, 15)
```

```
##  [1] 39.1 39.5 40.3   NA 36.7 39.3 38.9 39.2 34.1 42.0 37.8 37.8 41.1 38.6 34.6
```

5. `mutate()`

Adiciona colunas novas ou vindas de operações

```
# adicionar colunas
penguins_mutate ← penguins %>%
  dplyr::mutate(body_mass_kg = body_mass_g/1e3, .before = sex)
head(penguins_mutate)
```

```
## ¤[90m# A tibble: 6 × 9¤[39m
##   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g body_mass_kg sex     year
##   ¤[3m¤[90m<fct>¤[39m¤[23m   ¤[3m¤[90m<fct>¤[39m¤[23m              ¤[3m¤[90m<dbl>¤[39m¤[23m         ¤[3m¤[90m<d
## ¤[90m1¤[39m Adelie  Torgersen           39.1          18.7                181        ¤[4m3¤[24m750         3.75 ma
## ¤[90m2¤[39m Adelie  Torgersen           39.5          17.4                186        ¤[4m3¤[24m800         3.8  fe
## ¤[90m3¤[39m Adelie  Torgersen           40.3          18                  195        ¤[4m3¤[24m250         3.25 fe
## ¤[90m4¤[39m Adelie  Torgersen           ¤[31mNA¤[39m            ¤[31mNA¤[39m                  ¤[31mNA¤[39m
## ¤[90m5¤[39m Adelie  Torgersen           36.7          19.3                193        ¤[4m3¤[24m450         3.45 fe
## ¤[90m6¤[39m Adelie  Torgersen           39.3          20.6                190        ¤[4m3¤[24m650         3.65 ma
```

5. `mutate()`

Adiciona colunas novas ou vindas de operações

```
## modificar varias colunas
penguins_mutate_across ← penguins %>%
  dplyr::mutate(across(where(is.factor), as.character))
head(penguins_mutate_across)
```

```
## ¤[90m# A tibble: 6 × 8¤[39m
##   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex     year
##   ¤[3m¤[90m<chr>¤[39m¤[23m   ¤[3m¤[90m<chr>¤[39m¤[23m              ¤[3m¤[90m<dbl>¤[39m¤[23m         ¤[3m¤[90m<d
## ¤[90m1¤[39m Adelie  Torgersen           39.1          18.7                181        ¤[4m3¤[24m750 male    ¤[4m2¤[
## ¤[90m2¤[39m Adelie  Torgersen           39.5          17.4                186        ¤[4m3¤[24m800 female  ¤[4m2¤[
## ¤[90m3¤[39m Adelie  Torgersen           40.3          18                  195        ¤[4m3¤[24m250 female  ¤[4m2¤[
## ¤[90m4¤[39m Adelie  Torgersen           ¤[31mNA¤[39m            ¤[31mNA¤[39m                  ¤[31mNA¤[39m
## ¤[90m5¤[39m Adelie  Torgersen           36.7          19.3                193        ¤[4m3¤[24m450 female  ¤[4m2¤[
## ¤[90m6¤[39m Adelie  Torgersen           39.3          20.6                190        ¤[4m3¤[24m650 male    ¤[4m2¤[
```

6. `arrange()`

Reordena de forma crescente pelos valores de uma coluna

```
# reordenar os valores por ordem crescente
penguins_arrange ← penguins %>%
  dplyr::arrange(body_mass_g)
head(penguins_arrange)
```

```
## ¤[90m# A tibble: 6 × 8¤[39m
##   species   island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex     year
##   ¤[3m¤[90m<fct>¤[39m¤[23m     ¤[3m¤[90m<fct>¤[39m¤[23m              ¤[3m¤[90m<dbl>¤[39m¤[23m         ¤[3m¤[90m
## ¤[90m1¤[39m Chinstrap Dream               46.9          16.6               192        ¤[4m2¤[24m700 female  ¤[4m2
## ¤[90m2¤[39m Adelie    Biscoe              36.5          16.6               181        ¤[4m2¤[24m850 female  ¤[4m2
## ¤[90m3¤[39m Adelie    Biscoe              36.4          17.1               184        ¤[4m2¤[24m850 female  ¤[4m2
## ¤[90m4¤[39m Adelie    Biscoe              34.5          18.1               187        ¤[4m2¤[24m900 female  ¤[4m2
## ¤[90m5¤[39m Adelie    Dream               33.1          16.1               178        ¤[4m2¤[24m900 female  ¤[4m2
## ¤[90m6¤[39m Adelie    Torgersen           38.6          17                 188        ¤[4m2¤[24m900 female  ¤[4m2
```

6. `arrange()`

Reordena de forma decrescente pelos valores de uma coluna

```
# reordenar os valores por ordem decrescente
penguins_arrange_desc ← penguins %>%
  dplyr::arrange(desc(body_mass_g))
head(penguins_arrange_desc)
```

```
## ¤[90m# A tibble: 6 × 8¤[39m
##   species island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex    year
##   ¤[3m¤[90m<fct>¤[39m¤[23m   ¤[3m¤[90m<fct>¤[39m¤[23m           ¤[3m¤[90m<dbl>¤[39m¤[23m         ¤[3m¤[90m<dbl>
## ¤[90m1¤[39m Gentoo  Biscoe           49.2          15.2               221        ¤[4m6¤[24m300 male   ¤[4m2¤[24m0
## ¤[90m2¤[39m Gentoo  Biscoe           59.6          17                 230        ¤[4m6¤[24m050 male   ¤[4m2¤[24m0
## ¤[90m3¤[39m Gentoo  Biscoe           51.1          16.3               220        ¤[4m6¤[24m000 male   ¤[4m2¤[24m0
## ¤[90m4¤[39m Gentoo  Biscoe           48.8          16.2               222        ¤[4m6¤[24m000 male   ¤[4m2¤[24m0
## ¤[90m5¤[39m Gentoo  Biscoe           45.2          16.4               223        ¤[4m5¤[24m950 male   ¤[4m2¤[24m0
## ¤[90m6¤[39m Gentoo  Biscoe           49.8          15.9               229        ¤[4m5¤[24m950 male   ¤[4m2¤[24m0
```

6. `arrange()`

Reordena de forma crescente ou decrescente pelos valores de uma coluna

```
# reordenar os valores por ordem decrescente
penguins_arrange_desc_m ← penguins %>%
  dplyr::arrange(-body_mass_g)
head(penguins_arrange_desc_m)
```

```
## ¤[90m# A tibble: 6 × 8¤[39m
##   species island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex    year
##   ¤[3m¤[90m<fct>¤[39m¤[23m   ¤[3m¤[90m<fct>¤[39m¤[23m           ¤[3m¤[90m<dbl>¤[39m¤[23m         ¤[3m¤[90m<dbl>
## ¤[90m1¤[39m Gentoo  Biscoe           49.2          15.2               221        ¤[4m6¤[24m300 male   ¤[4m2¤[24m0
## ¤[90m2¤[39m Gentoo  Biscoe           59.6          17                 230        ¤[4m6¤[24m050 male   ¤[4m2¤[24m0
## ¤[90m3¤[39m Gentoo  Biscoe           51.1          16.3               220        ¤[4m6¤[24m000 male   ¤[4m2¤[24m0
## ¤[90m4¤[39m Gentoo  Biscoe           48.8          16.2               222        ¤[4m6¤[24m000 male   ¤[4m2¤[24m0
## ¤[90m5¤[39m Gentoo  Biscoe           45.2          16.4               223        ¤[4m5¤[24m950 male   ¤[4m2¤[24m0
## ¤[90m6¤[39m Gentoo  Biscoe           49.8          15.9               229        ¤[4m5¤[24m950 male   ¤[4m2¤[24m0
```

6. `arrange()`

Reordena de forma crescente ou decrescente pelos valores de várias colunas

```
# reordenar os valores por ordem crescente de varias colunas
penguins_arrange_across ← penguins %>%
  dplyr::arrange(across(where(is.numeric)))
head(penguins_arrange_across)
```

```
## ¤[90m# A tibble: 6 × 8¤[39m
##   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex     year
##   ¤[3m¤[90m<fct>¤[39m¤[23m   ¤[3m¤[90m<fct>¤[39m¤[23m              ¤[3m¤[90m<dbl>¤[39m¤[23m         ¤[3m¤[90m<d
## ¤[90m1¤[39m Adelie  Dream               32.1          15.5               188        ¤[4m3¤[24m050 female  ¤[4m2¤[
## ¤[90m2¤[39m Adelie  Dream               33.1          16.1               178        ¤[4m2¤[24m900 female  ¤[4m2¤[
## ¤[90m3¤[39m Adelie  Torgersen           33.5          19                 190        ¤[4m3¤[24m600 female  ¤[4m2¤[
## ¤[90m4¤[39m Adelie  Dream               34            17.1               185        ¤[4m3¤[24m400 female  ¤[4m2¤[
## ¤[90m5¤[39m Adelie  Torgersen           34.1          18.1               193        ¤[4m3¤[24m475 ¤[31mNA¤[39m
## ¤[90m6¤[39m Adelie  Torgersen           34.4          18.4               184        ¤[4m3¤[24m325 female  ¤[4m2¤[
```

7. `filter()`

Filtro de linhas por valores de uma coluna

```
# filtrar linhas por valores de uma coluna
penguins_filter ← penguins %>%
  dplyr::filter(species == "Adelie")
head(penguins_filter)
```

7. `filter()`

Filtro de linhas por valores de uma coluna

```
# filtrar linhas por valores de duas colunas
penguins_filter_two ← penguins %>%
  dplyr::filter(species == "Adelie" & sex == "female")
head(penguins_filter_two)
```

7. `filter()`

Filtro de linhas por valores de uma coluna e vários valores

```
# filtrar linhas por mais de um valor e mais de uma coluna
penguins_filter_in ← penguins %>%
  dplyr::filter(species %in% c("Adelie", "Gentoo"),
                sex == "female")
head(penguins_filter_in)
```

7. `filter()`

Filtro por valores não-NA

```
# filtrar linhas por nas
penguins_filter_na ← penguins %>%
  dplyr::filter(!is.na(sex) == TRUE)
head(penguins_filter_na)
```

7. `filter()`

Filtro por intervalos

```
# filtrar linhas por valores em um intervalo
penguins_filter_between ← penguins %>%
  dplyr::filter(between(body_mass_g, 3000, 4000))
head(penguins_filter_between)
```

7. `filter()`

Filtro linhas por várias colunas simultaneamente

```
# filtrar linhas por valores de varias colunas
penguins_filter_if ← penguins %>%
  dplyr::filter(if_all(where(is.integer), ~ . > 200))
head(penguins_filter_if)
```

8. `slice()`

Seleção das linhas por intervalos, indicando quais linhas desejamos

```
# selecionar linhas por intervalos
penguins_slice ← penguins %>%
  dplyr::slice(n = c(1, 3, 300:n()))
head(penguins_slice)
```

```
## ¤[90m# A tibble: 6 × 8¤[39m
##   species   island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex     year
##   ¤[3m¤[90m<fct>¤[39m¤[23m     ¤[3m¤[90m<fct>¤[39m¤[23m              ¤[3m¤[90m<dbl>¤[39m¤[23m         ¤[3m¤[90m
## ¤[90m1¤[39m Adelie    Torgersen           39.1          18.7               181        ¤[4m3¤[24m750 male    ¤[4m2
## ¤[90m2¤[39m Adelie    Torgersen           40.3          18                 195        ¤[4m3¤[24m250 female  ¤[4m2
## ¤[90m3¤[39m Chinstrap Dream               50.6          19.4               193        ¤[4m3¤[24m800 male    ¤[4m2
## ¤[90m4¤[39m Chinstrap Dream               46.7          17.9               195        ¤[4m3¤[24m300 female  ¤[4m2
## ¤[90m5¤[39m Chinstrap Dream               52            19                 197        ¤[4m4¤[24m150 male    ¤[4m2
## ¤[90m6¤[39m Chinstrap Dream               50.5          18.4               200        ¤[4m3¤[24m400 female  ¤[4m2
```

8. `slice()`

Seleção das linhas iniciais

```
# selecionar linhas iniciais
penguins_slice_head ← penguins %>%
  dplyr::slice_head(n = 5)
head(penguins_slice_head)
```

```
## ¤[90m# A tibble: 5 × 8¤[39m
##   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex     year
##   ¤[3m¤[90m<fct>¤[39m¤[23m   ¤[3m¤[90m<fct>¤[39m¤[23m              ¤[3m¤[90m<dbl>¤[39m¤[23m         ¤[3m¤[90m<d
## ¤[90m1¤[39m Adelie  Torgersen           39.1          18.7                181        ¤[4m3¤[24m750 male    ¤[4m2¤[
## ¤[90m2¤[39m Adelie  Torgersen           39.5          17.4                186        ¤[4m3¤[24m800 female  ¤[4m2¤[
## ¤[90m3¤[39m Adelie  Torgersen           40.3          18                  195        ¤[4m3¤[24m250 female  ¤[4m2¤[
## ¤[90m4¤[39m Adelie  Torgersen           ¤[31mNA¤[39m            ¤[31mNA¤[39m                  ¤[31mNA¤[39m
## ¤[90m5¤[39m Adelie  Torgersen           36.7          19.3                193        ¤[4m3¤[24m450 female  ¤[4m2¤[
```

8. `slice()`

Seleção de linhas por valores de uma coluna

```
# selecionar linhas por valores de uma coluna
penguins_slice_max ← penguins %>%
  dplyr::slice_max(body_mass_g, n = 5)
head(penguins_slice_max)
```

```
## ¤[90m# A tibble: 6 × 8¤[39m
##   species island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex    year
##   ¤[3m¤[90m<fct>¤[39m¤[23m   ¤[3m¤[90m<fct>¤[39m¤[23m           ¤[3m¤[90m<dbl>¤[39m¤[23m         ¤[3m¤[90m<dbl>
## ¤[90m1¤[39m Gentoo  Biscoe           49.2          15.2               221        ¤[4m6¤[24m300 male   ¤[4m2¤[24m0
## ¤[90m2¤[39m Gentoo  Biscoe           59.6          17                 230        ¤[4m6¤[24m050 male   ¤[4m2¤[24m0
## ¤[90m3¤[39m Gentoo  Biscoe           51.1          16.3               220        ¤[4m6¤[24m000 male   ¤[4m2¤[24m0
## ¤[90m4¤[39m Gentoo  Biscoe           48.8          16.2               222        ¤[4m6¤[24m000 male   ¤[4m2¤[24m0
## ¤[90m5¤[39m Gentoo  Biscoe           45.2          16.4               223        ¤[4m5¤[24m950 male   ¤[4m2¤[24m0
## ¤[90m6¤[39m Gentoo  Biscoe           49.8          15.9               229        ¤[4m5¤[24m950 male   ¤[4m2¤[24m0
```

8. `slice()`

Seleção de linhas aleatoriamente

```
# selecionar linhas aleatoriamente
penguins_slice_sample ← penguins %>%
  dplyr::slice_sample(n = 30)
head(penguins_slice_sample)
```

```
## ¤[90m# A tibble: 6 × 8¤[39m
##   species island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex     year
##   ¤[3m¤[90m<fct>¤[39m¤[23m   ¤[3m¤[90m<fct>¤[39m¤[23m           ¤[3m¤[90m<dbl>¤[39m¤[23m         ¤[3m¤[90m<dbl>
## ¤[90m1¤[39m Gentoo  Biscoe           48.2          15.6               221        ¤[4m5¤[24m100 male    ¤[4m2¤[24m
## ¤[90m2¤[39m Adelie  Biscoe           35.7          16.9               185        ¤[4m3¤[24m150 female  ¤[4m2¤[24m
## ¤[90m3¤[39m Adelie  Biscoe           41            20                 203        ¤[4m4¤[24m725 male    ¤[4m2¤[24m
## ¤[90m4¤[39m Gentoo  Biscoe           46.2          14.5               209        ¤[4m4¤[24m800 female  ¤[4m2¤[24m
## ¤[90m5¤[39m Adelie  Dream            37.8          18.1               193        ¤[4m3¤[24m750 male    ¤[4m2¤[24m
## ¤[90m6¤[39m Gentoo  Biscoe           50            15.3               220        ¤[4m5¤[24m550 male    ¤[4m2¤[24m
```

9. `distinct()`

Retira linhas com valores duplicados com base nos valores de colunas

```
# retirar linhas com valores duplicados
penguins_distinct ← penguins %>%
  dplyr::distinct(body_mass_g)
head(penguins_distinct)
```

```
## ¤[90m# A tibble: 6 × 1¤[39m
##   body_mass_g
##         ¤[3m¤[90m<int>¤[39m¤[23m
## ¤[90m1¤[39m        ¤[4m3¤[24m750
## ¤[90m2¤[39m        ¤[4m3¤[24m800
## ¤[90m3¤[39m        ¤[4m3¤[24m250
## ¤[90m4¤[39m          ¤[31mNA¤[39m
## ¤[90m5¤[39m        ¤[4m3¤[24m450
## ¤[90m6¤[39m        ¤[4m3¤[24m650
```

9. `distinct()`

Retira linhas com valores duplicados com base nos valores de colunas, mas mantendo as colunas

```
# retirar linhas com valores duplicados e manter colunas
penguins_distinct_keep_all ← penguins %>%
  dplyr::distinct(body_mass_g, .keep_all = TRUE)
head(penguins_distinct_keep_all)
```

```
## ¤[90m# A tibble: 6 × 8¤[39m
##   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex     year
##   ¤[3m¤[90m<fct>¤[39m¤[23m   ¤[3m¤[90m<fct>¤[39m¤[23m              ¤[3m¤[90m<dbl>¤[39m¤[23m         ¤[3m¤[90m<d
## ¤[90m1¤[39m Adelie  Torgersen           39.1          18.7                181        ¤[4m3¤[24m750 male    ¤[4m2¤[
## ¤[90m2¤[39m Adelie  Torgersen           39.5          17.4                186        ¤[4m3¤[24m800 female  ¤[4m2¤[
## ¤[90m3¤[39m Adelie  Torgersen           40.3          18                  195        ¤[4m3¤[24m250 female  ¤[4m2¤[
## ¤[90m4¤[39m Adelie  Torgersen           ¤[31mNA¤[39m            ¤[31mNA¤[39m                  ¤[31mNA¤[39m
## ¤[90m5¤[39m Adelie  Torgersen           36.7          19.3                193        ¤[4m3¤[24m450 female  ¤[4m2¤[
## ¤[90m6¤[39m Adelie  Torgersen           39.3          20.6                190        ¤[4m3¤[24m650 male    ¤[4m2¤[
```

9. `distinct()`

Retira linhas com valores duplicados para várias colunas

```
# retirar linhas com valores duplicados para varias colunas
penguins_distinct_keep_all_across ← penguins %>%
  dplyr::distinct(across(where(is.integer)), .keep_all = TRUE)
head(penguins_distinct_keep_all_across)
```

```
## ¤[90m# A tibble: 6 × 8¤[39m
##   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex     year
##   ¤[3m¤[90m<fct>¤[39m¤[23m   ¤[3m¤[90m<fct>¤[39m¤[23m              ¤[3m¤[90m<dbl>¤[39m¤[23m         ¤[3m¤[90m<d
## ¤[90m1¤[39m Adelie  Torgersen           39.1          18.7                181        ¤[4m3¤[24m750 male    ¤[4m2¤[
## ¤[90m2¤[39m Adelie  Torgersen           39.5          17.4                186        ¤[4m3¤[24m800 female  ¤[4m2¤[
## ¤[90m3¤[39m Adelie  Torgersen           40.3          18                  195        ¤[4m3¤[24m250 female  ¤[4m2¤[
## ¤[90m4¤[39m Adelie  Torgersen           ¤[31mNA¤[39m            ¤[31mNA¤[39m                  ¤[31mNA¤[39m
## ¤[90m5¤[39m Adelie  Torgersen           36.7          19.3                193        ¤[4m3¤[24m450 female  ¤[4m2¤[
## ¤[90m6¤[39m Adelie  Torgersen           39.3          20.6                190        ¤[4m3¤[24m650 male    ¤[4m2¤[
```

10. `count()`

Conta valores de uma ou mais colunas, geralmente para variáveis categóricas

```
# contagens de valores para uma coluna
penguins_count ← penguins %>%
  dplyr::count(species)
penguins_count
```

```
## ¤[90m# A tibble: 3 × 2¤[39m
##   species       n
##   ¤[3m¤[90m<fct>¤[39m¤[23m     ¤[3m¤[90m<int>¤[39m¤[23m
## ¤[90m1¤[39m Adelie      152
## ¤[90m2¤[39m Chinstrap    68
## ¤[90m3¤[39m Gentoo      124
```

10. `count()`

Conta valores de uma ou mais colunas, geralmente para variáveis categóricas

```
# contagens de valores para mais de uma coluna
penguins_count_two ← penguins %>%
  dplyr::count(species, island)
penguins_count_two
```

```
## ¤[90m# A tibble: 5 × 3¤[39m
##   species   island        n
##   ¤[3m¤[90m<fct>¤[39m¤[23m     ¤[3m¤[90m<fct>¤[39m¤[23m     ¤[3m¤[90m<int>¤[39m¤[23m
## ¤[90m1¤[39m Adelie    Biscoe       44
## ¤[90m2¤[39m Adelie    Dream        56
## ¤[90m3¤[39m Adelie    Torgersen    52
## ¤[90m4¤[39m Chinstrap Dream        68
## ¤[90m5¤[39m Gentoo    Biscoe      124
```

11. `groun_by()`

Transforma um tibble em um tibble agrupado, onde as operações são realizadas "por grupo"

```
# agrupamento
penguins_group_by ← penguins %>%
  dplyr::group_by(species)
head(penguins_group_by)
```

```
## ¤[90m# A tibble: 6 × 8¤[39m
## ¤[90m# Groups:   species [1]¤[39m
##   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex     year
##   ¤[3m¤[90m<fct>¤[39m¤[23m   ¤[3m¤[90m<fct>¤[39m¤[23m              ¤[3m¤[90m<dbl>¤[39m¤[23m         ¤[3m¤[90m<d
## ¤[90m1¤[39m Adelie  Torgersen           39.1          18.7                181        ¤[4m3¤[24m750 male    ¤[4m2¤[
## ¤[90m2¤[39m Adelie  Torgersen           39.5          17.4                186        ¤[4m3¤[24m800 female  ¤[4m2¤[
## ¤[90m3¤[39m Adelie  Torgersen           40.3          18                  195        ¤[4m3¤[24m250 female  ¤[4m2¤[
## ¤[90m4¤[39m Adelie  Torgersen           ¤[31mNA¤[39m            ¤[31mNA¤[39m                  ¤[31mNA¤[39m
## ¤[90m5¤[39m Adelie  Torgersen           36.7          19.3                193        ¤[4m3¤[24m450 female  ¤[4m2¤[
## ¤[90m6¤[39m Adelie  Torgersen           39.3          20.6                190        ¤[4m3¤[24m650 male    ¤[4m2¤[
```

11. `groun_by()`

Transforma um tibble em um tibble agrupado, onde as operações são realizadas "por grupo"

```
# agrupamento de várias colunas
penguins_group_by_across ← penguins %>%
  dplyr::group_by(across(where(is.factor)))
head(penguins_group_by_across)
```

```
## ¤[90m# A tibble: 6 × 8¤[39m
## ¤[90m# Groups:   species, island, sex [3]¤[39m
##   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex     year
##   ¤[3m¤[90m<fct>¤[39m¤[23m   ¤[3m¤[90m<fct>¤[39m¤[23m              ¤[3m¤[90m<dbl>¤[39m¤[23m         ¤[3m¤[90m<d
## ¤[90m1¤[39m Adelie  Torgersen           39.1          18.7                181        ¤[4m3¤[24m750 male    ¤[4m2¤[
## ¤[90m2¤[39m Adelie  Torgersen           39.5          17.4                186        ¤[4m3¤[24m800 female  ¤[4m2¤[
## ¤[90m3¤[39m Adelie  Torgersen           40.3          18                  195        ¤[4m3¤[24m250 female  ¤[4m2¤[
## ¤[90m4¤[39m Adelie  Torgersen           ¤[31mNA¤[39m            ¤[31mNA¤[39m                  ¤[31mNA¤[39m
## ¤[90m5¤[39m Adelie  Torgersen           36.7          19.3                193        ¤[4m3¤[24m450 female  ¤[4m2¤[
## ¤[90m6¤[39m Adelie  Torgersen           39.3          20.6                190        ¤[4m3¤[24m650 male    ¤[4m2¤[
```

12. `summarise()`

Agregação ou resumo dos dados através de funções

```
# resumo
penguins_summarise ← penguins %>%
  dplyr::group_by(species) %>%
  dplyr::summarize(body_mass_g_mean = mean(body_mass_g, na.rm = TRUE),
                   body_mass_g_sd = sd(body_mass_g, na.rm = TRUE))
penguins_summarise
```

```
## ¤[90m# A tibble: 3 × 3¤[39m
##   species   body_mass_g_mean body_mass_g_sd
##   ¤[3m¤[90m<fct>¤[39m¤[23m                ¤[3m¤[90m<dbl>¤[39m¤[23m          ¤[3m¤[90m<dbl>¤[39m¤[23m
## ¤[90m1¤[39m Adelie               ¤[4m3¤[24m701.           459.
## ¤[90m2¤[39m Chinstrap            ¤[4m3¤[24m733.           384.
## ¤[90m3¤[39m Gentoo               ¤[4m5¤[24m076.           504.
```

12. `summarise()`

Agregação ou resumo dos dados através de funções

```
# resumo para várias colunas
penguins_summarise_across ← penguins %>%
  dplyr::group_by(species) %>%
  dplyr::summarize(across(where(is.numeric), ~ mean(.x, na.rm = TRUE)))
penguins_summarise_across
```

```
## ¤[90m# A tibble: 3 × 6¤[39m
##   species   bill_length_mm bill_depth_mm flipper_length_mm body_mass_g  year
##   ¤[3m¤[90m<fct>¤[39m¤[23m              ¤[3m¤[90m<dbl>¤[39m¤[23m         ¤[3m¤[90m<dbl>¤[39m¤[23m             ¤[
## ¤[90m1¤[39m Adelie              38.8          18.3               190.       ¤[4m3¤[24m701. ¤[4m2¤[24m008.
## ¤[90m2¤[39m Chinstrap           48.8          18.4               196.       ¤[4m3¤[24m733. ¤[4m2¤[24m008.
## ¤[90m3¤[39m Gentoo              47.5          15.0               217.       ¤[4m5¤[24m076. ¤[4m2¤[24m008.
```

13. `bind_rows()` e `bind_cols()`

Combina dados por linhas e por colunas

```
# selecionar as linhas para dois tibbles
penguins_01 ← dplyr::slice(penguins, 1:5) %>% dplyr::select(1:3)
penguins_02 ← dplyr::slice(penguins, 51:55) %>% dplyr::select(4:6)

# combinar as linhas
penguins_bind_rows ← dplyr::bind_rows(penguins_01, penguins_02, .id = "id")
head(penguins_bind_rows)
```

```
## ¤[90m# A tibble: 6 × 7¤[39m
##   id    species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##   ¤[3m¤[90m<chr>¤[39m¤[23m ¤[3m¤[90m<fct>¤[39m¤[23m   ¤[3m¤[90m<fct>¤[39m¤[23m             ¤[3m¤[90m<dbl>¤[39m
## ¤[90m1¤[39m 1     Adelie  Torgersen           39.1          ¤[31mNA¤[39m                ¤[31mNA¤[39m          ¤
## ¤[90m2¤[39m 1     Adelie  Torgersen           39.5          ¤[31mNA¤[39m                ¤[31mNA¤[39m          ¤
## ¤[90m3¤[39m 1     Adelie  Torgersen           40.3          ¤[31mNA¤[39m                ¤[31mNA¤[39m          ¤
## ¤[90m4¤[39m 1     Adelie  Torgersen           ¤[31mNA¤[39m            ¤[31mNA¤[39m                ¤[31mNA¤[39m
## ¤[90m5¤[39m 1     Adelie  Torgersen           36.7          ¤[31mNA¤[39m                ¤[31mNA¤[39m          ¤
## ¤[90m6¤[39m 2     ¤[31mNA¤[39m      ¤[31mNA¤[39m                  ¤[31mNA¤[39m           17.7                186
```

13. `bind_rows()` e `bind_cols()`

Combina dados por linhas e por colunas

```
# selecionar as linhas para dois tibbles
penguins_01 ← dplyr::slice(penguins, 1:5)
penguins_02 ← dplyr::slice(penguins, 51:55)

## combinar as colunas
penguins_bind_cols ← dplyr::bind_cols(penguins_01, penguins_02, .name_repair = "unique")
head(penguins_bind_cols)
```

```
## ¤[90m# A tibble: 5 × 16¤[39m
##   species ... 1 island ... 2 bill_length_mm ... 3 bill_depth_mm ... 4 flipper_length_mm ... 5 body_mass_g ... 6 sex ... 7 year ...
##   ¤[3m¤[90m<fct>¤[39m¤[23m       ¤[3m¤[90m<fct>¤[39m¤[23m                    ¤[3m¤[90m<dbl>¤[39m¤[23m
## ¤[90m1¤[39m Adelie      Torgersen                39.1              18.7                   181           ¤[4m3¤[24
## ¤[90m2¤[39m Adelie      Torgersen                39.5              17.4                   186           ¤[4m3¤[24
## ¤[90m3¤[39m Adelie      Torgersen                40.3              18                     195           ¤[4m3¤[24
## ¤[90m4¤[39m Adelie      Torgersen                ¤[31mNA¤[39m              ¤[31mNA¤[39m                        ¤[
## ¤[90m5¤[39m Adelie      Torgersen                36.7              19.3                   193           ¤[4m3¤[24
## ¤[90m# … with 6 more variables: bill_length_mm ... 11 <dbl>, bill_depth_mm ... 12 <dbl>, flipper_length_mm ... 13 <int>,
## ¤[90m#   sex ... 15 <fct>, year ... 16 <int>¤[39m
```

14. `*_join()`

Combinação de pares de dados tabulares por uma ou mais colunas chaves

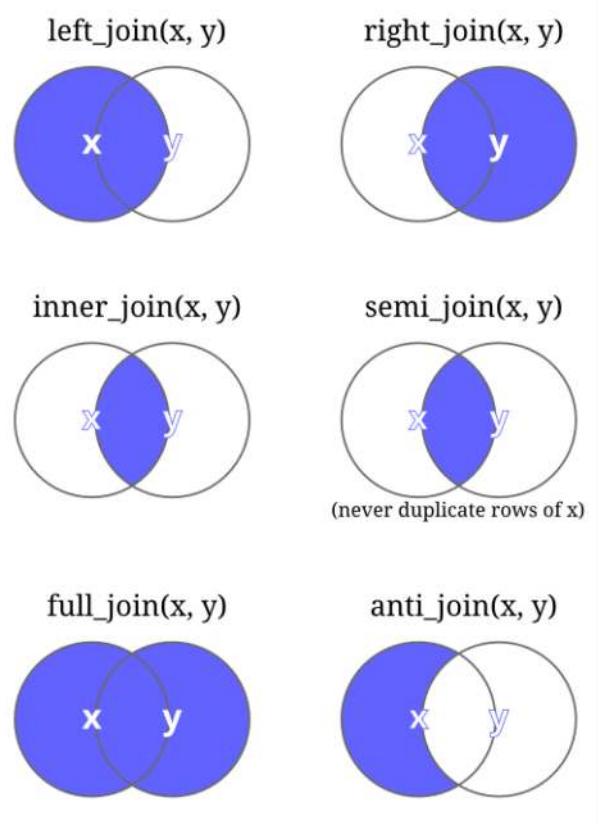
Junção de mutação

- `left_join(x, y)`: mantém as observações em x
- `right_join(x, y)`: mantém as observações em y
- `inner_join(x, y)`: mantém apenas as observações em x e em y
- `full_join(x, y)`: mantém todas as observações em x e em y

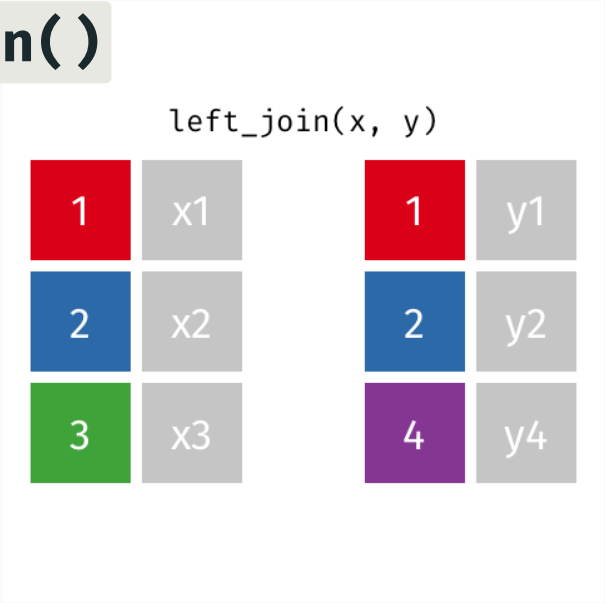
Junção de filtragem

- `semi_join(x, y)`: mantém as observações em x que têm uma correspondência em y
- `anti_join(x, y)`: elimina as observações em x que têm uma correspondência em y

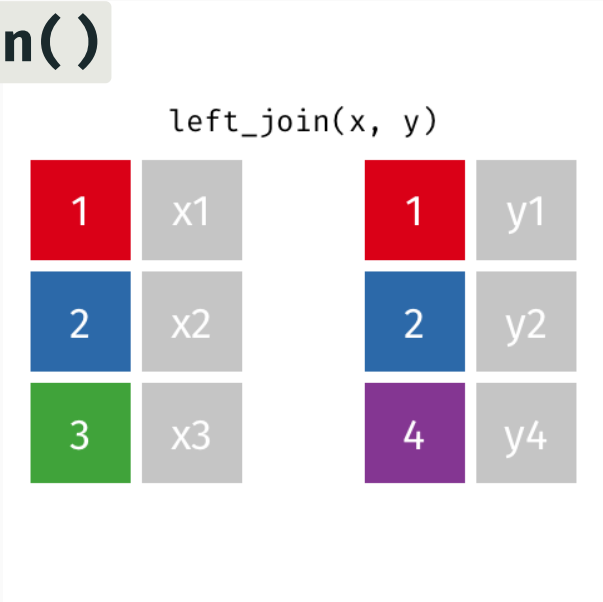
[Joining Data in R with dplyr]

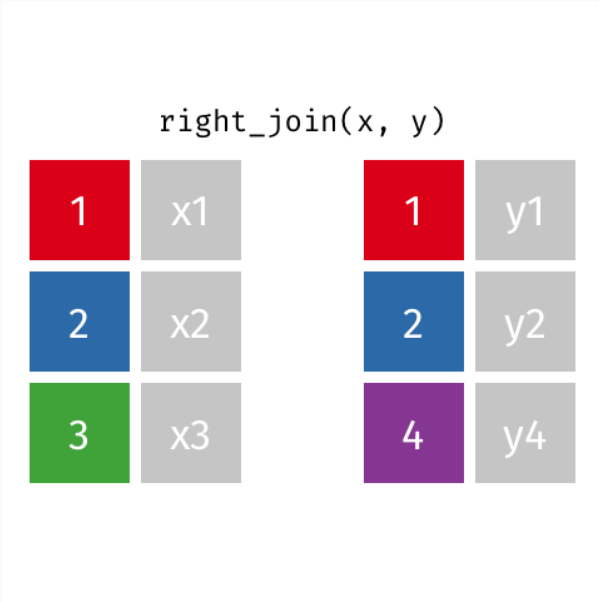


14. `*_join()`



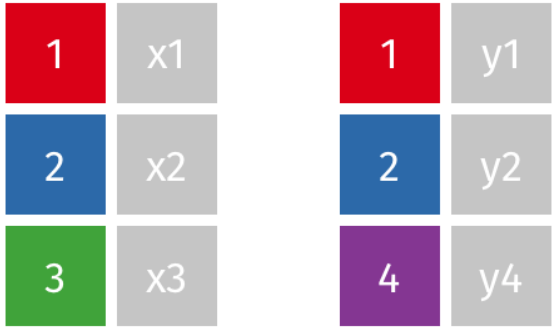
14. `*_join()`


left_join(x, y)
1
x1
2
x2
3
x3
1
y1
2
y2
4
y4

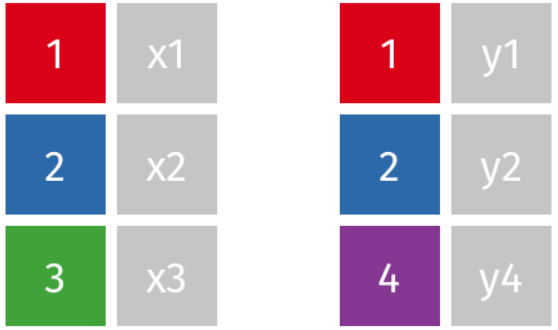

right_join(x, y)
1
x1
2
x2
3
x3
1
y1
2
y2
4
y4

14. `*_join()`

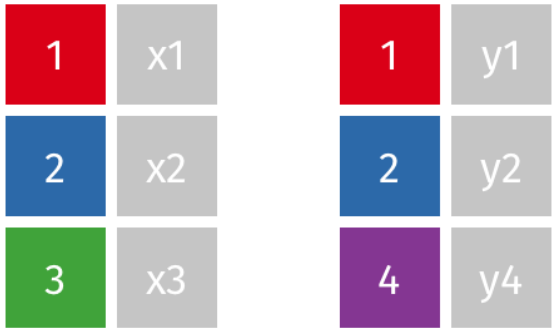
left_join(x, y)



right_join(x, y)

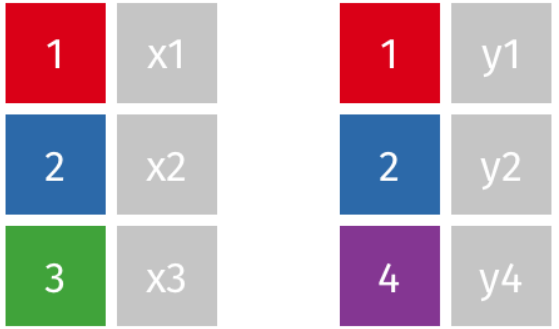


inner_join(x, y)

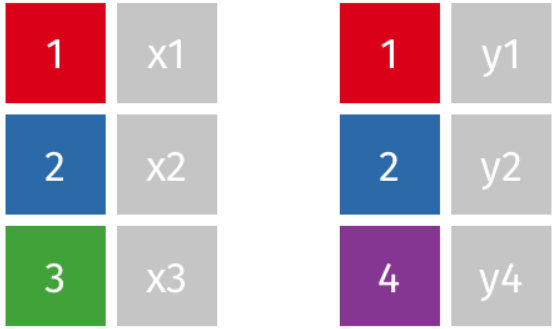


14. `*_join()`

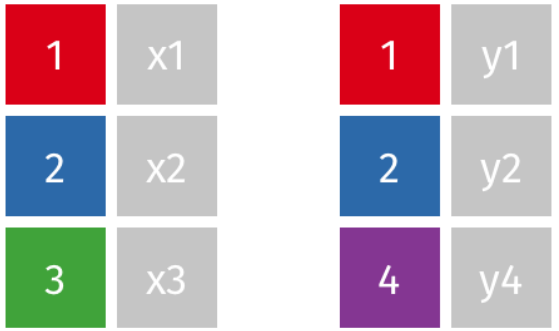
left_join(x, y)



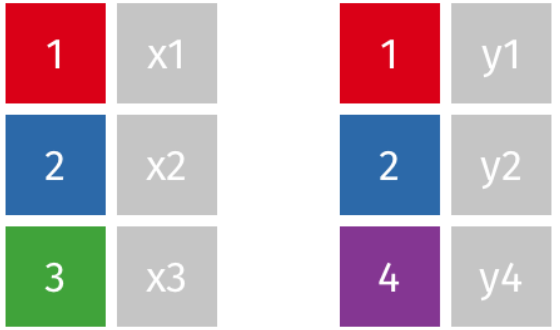
right_join(x, y)



inner_join(x, y)



full_join(x, y)



14. `*_join()`

Adicionar longitute e latitude para as ilhas amostradas

```
## coordenadas
penguin_islands ← tibble(
  island = c("Torgersen", "Biscoe", "Dream", "Alpha"),
  longitude = c(-64.083333, -63.775636, -64.233333, -63),
  latitude = c(-64.766667, -64.818569, -64.733333, -64.316667))
penguin_islands
```

```
## ¤[90m# A tibble: 4 × 3¤[39m
##   island    longitude latitude
##   ¤[3m¤[90m<chr>¤[39m¤[23m         ¤[3m¤[90m<dbl>¤[39m¤[23m    ¤[3m¤[90m<dbl>¤[39m¤[23m
## ¤[90m1¤[39m Torgersen     -¤[31m64¤[39m¤[31m.¤[39m¤[31m1¤[39m    -¤[31m64¤[39m¤[31m.¤[39m¤[31m8¤[39m
## ¤[90m2¤[39m Biscoe        -¤[31m63¤[39m¤[31m.¤[39m¤[31m8¤[39m    -¤[31m64¤[39m¤[31m.¤[39m¤[31m8¤[39m
## ¤[90m3¤[39m Dream         -¤[31m64¤[39m¤[31m.¤[39m¤[31m2¤[39m    -¤[31m64¤[39m¤[31m.¤[39m¤[31m7¤[39m
## ¤[90m4¤[39m Alpha         -¤[31m63¤[39m      -¤[31m64¤[39m¤[31m.¤[39m¤[31m3¤[39m
```

14. `*_join()`

Adicionar longitute e latitude para as ilhas amostradas

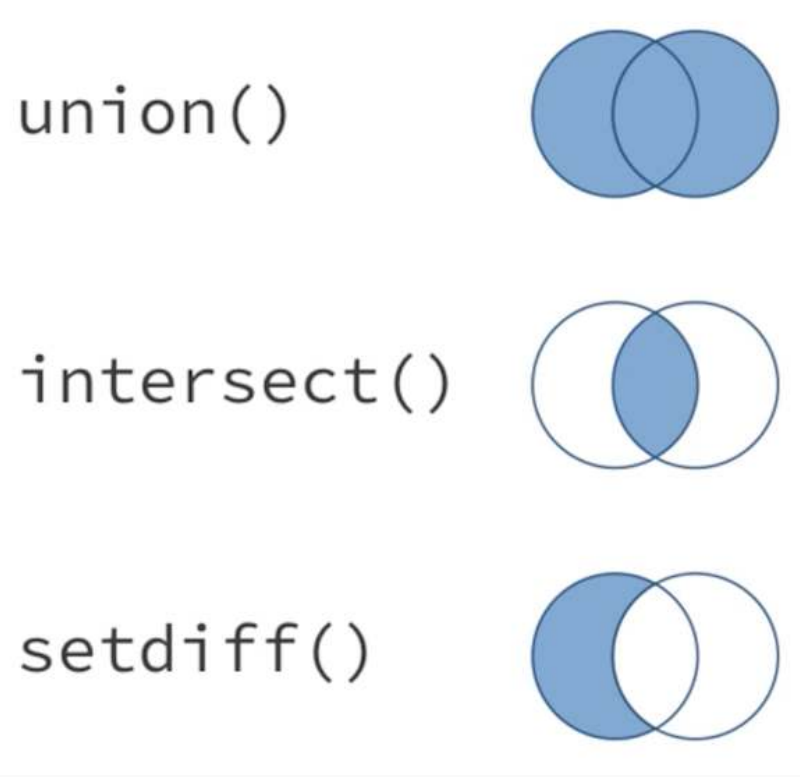
```
# juncao - left
penguins_left_join ← dplyr::left_join(penguins, penguin_islands, by = "island")
head(penguins_left_join)
```

```
## ¤[90m# A tibble: 6 × 10¤[39m
##   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex     year longitude latitude
##   ¤[3m¤[90m<fct>¤[39m¤[23m   ¤[3m¤[90m<chr>¤[39m¤[23m             ¤[3m¤[90m<dbl>¤[39m¤[23m         ¤[3m¤[90m<d
## ¤[90m1¤[39m Adelie  Torgersen           39.1          18.7               181        ¤[4m3¤[24m750 male    ¤[4m2¤[
## ¤[90m2¤[39m Adelie  Torgersen           39.5          17.4               186        ¤[4m3¤[24m800 female  ¤[4m2¤[
## ¤[90m3¤[39m Adelie  Torgersen           40.3          18                 195        ¤[4m3¤[24m250 female  ¤[4m2¤[
## ¤[90m4¤[39m Adelie  Torgersen           ¤[31mNA¤[39m            ¤[31mNA¤[39m                  ¤[31mNA¤[39m
## ¤[90m5¤[39m Adelie  Torgersen           36.7          19.3               193        ¤[4m3¤[24m450 female  ¤[4m2¤[
## ¤[90m6¤[39m Adelie  Torgersen           39.3          20.6               190        ¤[4m3¤[24m650 male    ¤[4m2¤[
```

15. Operações de conjuntos e comparação de dados

Operações para comparar dos dados

- `union(x, y)`: retorna todas as linhas que aparecem em x, y ou mais dos conjuntos de dados
- `interesect(x, y)`: retorna apenas as linhas que aparecem em x e em y
- `setdiff(x, y)`: retorna as linhas que aparecem x, mas não em y
- `setequal(x, y)`: retorna se x e y são iguais e quais suas diferenças



IMPORTANTE: ideia de **encadeamento das funções**

8. dplyr

Permite manipular os dados de forma simples

```
# selecionar colunas
penguins_dplyr ← penguins %>%
  dplyr::select(species, bill_length_mm, bill_depth_mm, flipper_length_mm, body_mass_g)
penguins_dplyr
```

```
## ¤[90m# A tibble: 344 × 5¤[39m
##    species bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##    ¤[3m¤[90m<fct>¤[39m¤[23m            ¤[3m¤[90m<dbl>¤[39m¤[23m         ¤[3m¤[90m<dbl>¤[39m¤[23m              ¤[3
## ¤[90m 1¤[39m Adelie            39.1          18.7               181        ¤[4m3¤[24m750
## ¤[90m 2¤[39m Adelie            39.5          17.4               186        ¤[4m3¤[24m800
## ¤[90m 3¤[39m Adelie            40.3          18                 195        ¤[4m3¤[24m250
## ¤[90m 4¤[39m Adelie            ¤[31mNA¤[39m            ¤[31mNA¤[39m                  ¤[31mNA¤[39m          ¤[31m
## ¤[90m 5¤[39m Adelie            36.7          19.3               193        ¤[4m3¤[24m450
## ¤[90m 6¤[39m Adelie            39.3          20.6               190        ¤[4m3¤[24m650
## ¤[90m 7¤[39m Adelie            38.9          17.8               181        ¤[4m3¤[24m625
## ¤[90m 8¤[39m Adelie            39.2          19.6               195        ¤[4m4¤[24m675
## ¤[90m 9¤[39m Adelie            34.1          18.1               193        ¤[4m3¤[24m475
## ¤[90m10¤[39m Adelie            42            20.2               190        ¤[4m4¤[24m250
## ¤[90m# … with 334 more rows¤[39m
```

8. dplyr

Permite manipular os dados de forma simples

```
# selecionar colunas e retirar linhas com nas
penguins_dplyr ← penguins %>%
  dplyr::select(species, bill_length_mm, bill_depth_mm, flipper_length_mm, body_mass_g) %>%
  dplyr::filter(if_all(where(is.numeric), ~ !is.na(.)))
penguins_dplyr
```

```
## ¤[90m# A tibble: 342 × 5¤[39m
##    species bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##    ¤[3m¤[90m<fct>¤[39m¤[23m            ¤[3m¤[90m<dbl>¤[39m¤[23m         ¤[3m¤[90m<dbl>¤[39m¤[23m              ¤[3
## ¤[90m 1¤[39m Adelie            39.1          18.7               181        ¤[4m3¤[24m750
## ¤[90m 2¤[39m Adelie            39.5          17.4               186        ¤[4m3¤[24m800
## ¤[90m 3¤[39m Adelie            40.3          18                 195        ¤[4m3¤[24m250
## ¤[90m 4¤[39m Adelie            36.7          19.3               193        ¤[4m3¤[24m450
## ¤[90m 5¤[39m Adelie            39.3          20.6               190        ¤[4m3¤[24m650
## ¤[90m 6¤[39m Adelie            38.9          17.8               181        ¤[4m3¤[24m625
## ¤[90m 7¤[39m Adelie            39.2          19.6               195        ¤[4m4¤[24m675
## ¤[90m 8¤[39m Adelie            34.1          18.1               193        ¤[4m3¤[24m475
## ¤[90m 9¤[39m Adelie            42            20.2               190        ¤[4m4¤[24m250
## ¤[90m10¤[39m Adelie            37.8          17.1               186        ¤[4m3¤[24m300
## ¤[90m# … with 332 more rows¤[39m
```

8. dplyr

Permite manipular os dados de forma simples

```
# selecionar colunas, retirar linhas com nas e calcular a media das colunas para as especies
penguins_dplyr ← penguins %>%
  dplyr::select(species, bill_length_mm, bill_depth_mm, flipper_length_mm, body_mass_g) %>%
  dplyr::filter(if_all(where(is.numeric), ~ !is.na(.))) %>%
  dplyr::group_by(species) %>%
  dplyr::summarize(across(where(is.numeric), ~ mean(.x, na.rm = TRUE)))
penguins_dplyr
```

```
## ¤[90m# A tibble: 3 × 5¤[39m
##   species   bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##   ¤[3m¤[90m<fct>¤[39m¤[23m              ¤[3m¤[90m<dbl>¤[39m¤[23m         ¤[3m¤[90m<dbl>¤[39m¤[23m              ¤[
## ¤[90m1¤[39m Adelie              38.8          18.3              190.       ¤[4m3¤[24m701.
## ¤[90m2¤[39m Chinstrap           48.8          18.4              196.       ¤[4m3¤[24m733.
## ¤[90m3¤[39m Gentoo              47.5          15.0              217.       ¤[4m5¤[24m076.
```

Exercícios

Exercício 12

Usando o conjunto de dados `penguins`, adicione essas novas colunas `bill_length_mm_log10, bill_depth_mm_log10, flipper_length_mm_log10, body_mass_g_log10`, que são a operação `log10` das colunas `bill_length_mm_log10, bill_depth_mm_log10, flipper_length_mm_log10, body_mass_g_log10` e atribua ao mesmo objeto `penguins_log` utilizando o formato tidyverse

04:00

Exercício 12

Solução

```
# exercicio 12
penguins_log ← penguins %>%
  dplyr::mutate(bill_length_mm_log10 = log10(bill_length_mm),
                bill_depth_mm_log10 = log10(bill_depth_mm),
                flipper_length_mm_log10 = log10(flipper_length_mm),
                body_mass_g_log10 = log10(body_mass_g))
penguins_log
```

```
## ¤[90m# A tibble: 344 × 12¤[39m
##    species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex     year bill_length_mm_l… bil
##    ¤[3m¤[90m<fct>¤[39m¤[23m   ¤[3m¤[90m<fct>¤[39m¤[23m              ¤[3m¤[90m<dbl>¤[39m¤[23m         ¤[3m¤[90m<
## ¤[90m 1¤[39m Adelie  Torgersen           39.1          18.7               181        ¤[4m3¤[24m750 male    ¤[4m2¤
## ¤[90m 2¤[39m Adelie  Torgersen           39.5          17.4               186        ¤[4m3¤[24m800 female  ¤[4m2¤
## ¤[90m 3¤[39m Adelie  Torgersen           40.3          18                 195        ¤[4m3¤[24m250 female  ¤[4m2¤
## ¤[90m 4¤[39m Adelie  Torgersen           ¤[31mNA¤[39m            ¤[31mNA¤[39m                  ¤[31mNA¤[39m
## ¤[90m 5¤[39m Adelie  Torgersen           36.7          19.3               193        ¤[4m3¤[24m450 female  ¤[4m2¤
## ¤[90m 6¤[39m Adelie  Torgersen           39.3          20.6               190        ¤[4m3¤[24m650 male    ¤[4m2¤
## ¤[90m 7¤[39m Adelie  Torgersen           38.9          17.8               181        ¤[4m3¤[24m625 female  ¤[4m2¤
## ¤[90m 8¤[39m Adelie  Torgersen           39.2          19.6               195        ¤[4m4¤[24m675 male    ¤[4m2¤
## ¤[90m 9¤[39m Adelie  Torgersen           34.1          18.1               193        ¤[4m3¤[24m475 ¤[31mNA¤[39m
```

Exercício 13

Usando o conjunto de dados `penguins`, ordene as linhas em forma decrescente pela coluna `body_mass_g`, atribuindo o resultado a um novo objeto `penguins_arrange` utilizando o formato tidyverse

04:00

Exercício 13

Solução

```
# exercicio 13
penguins_arrange ← penguins %>%
  dplyr::arrange(-body_mass_g)
penguins_arrange
```

```
## ¤[90m# A tibble: 344 × 8¤[39m
##    species island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex    year
##    ¤[3m¤[90m<fct>¤[39m¤[23m   ¤[3m¤[90m<fct>¤[39m¤[23m           ¤[3m¤[90m<dbl>¤[39m¤[23m         ¤[3m¤[90m<dbl
## ¤[90m 1¤[39m Gentoo  Biscoe           49.2          15.2               221        ¤[4m6¤[24m300 male   ¤[4m2¤[24m
## ¤[90m 2¤[39m Gentoo  Biscoe           59.6          17                 230        ¤[4m6¤[24m050 male   ¤[4m2¤[24m
## ¤[90m 3¤[39m Gentoo  Biscoe           51.1          16.3               220        ¤[4m6¤[24m000 male   ¤[4m2¤[24m
## ¤[90m 4¤[39m Gentoo  Biscoe           48.8          16.2               222        ¤[4m6¤[24m000 male   ¤[4m2¤[24m
## ¤[90m 5¤[39m Gentoo  Biscoe           45.2          16.4               223        ¤[4m5¤[24m950 male   ¤[4m2¤[24m
## ¤[90m 6¤[39m Gentoo  Biscoe           49.8          15.9               229        ¤[4m5¤[24m950 male   ¤[4m2¤[24m
## ¤[90m 7¤[39m Gentoo  Biscoe           48.4          14.6               213        ¤[4m5¤[24m850 male   ¤[4m2¤[24m
## ¤[90m 8¤[39m Gentoo  Biscoe           49.3          15.7               217        ¤[4m5¤[24m850 male   ¤[4m2¤[24m
## ¤[90m 9¤[39m Gentoo  Biscoe           55.1          16                 230        ¤[4m5¤[24m850 male   ¤[4m2¤[24m
## ¤[90m10¤[39m Gentoo  Biscoe           49.5          16.2               229        ¤[4m5¤[24m800 male   ¤[4m2¤[24m
## ¤[90m# … with 334 more rows¤[39m
```

Exercício 14

Usando o conjunto de dados `penguins`, filtre as linhas com `bill_length_mm` maior que 50 mm, `bill_depth_mm` menor que 20 mm, `flipper_length_mm` maior que 220 mm e `body_mass_g` menor que 5500 g, atribuindo o resultado a um novo objeto `penguins_filter` utilizando o formato tidyverse

04:00

Exercício 14

Solução

```
# exercicio 14
penguins_filter ← penguins %>%
  dplyr::filter(bill_length_mm > 50,
                bill_depth_mm < 20,
                flipper_length_mm > 220,
                body_mass_g < 5500)
penguins_filter
```

```
## ¤[90m# A tibble: 6 × 8¤[39m
##   species island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex    year
##   ¤[3m¤[90m<fct>¤[39m¤[23m   ¤[3m¤[90m<fct>¤[39m¤[23m           ¤[3m¤[90m<dbl>¤[39m¤[23m         ¤[3m¤[90m<dbl>¤
## ¤[90m1¤[39m Gentoo  Biscoe           50.5          15.9               225        ¤[4m5¤[24m400 male   ¤[4m2¤[24m0
## ¤[90m2¤[39m Gentoo  Biscoe           50.1          15                 225        ¤[4m5¤[24m000 male   ¤[4m2¤[24m0
## ¤[90m3¤[39m Gentoo  Biscoe           52.5          15.6               221        ¤[4m5¤[24m450 male   ¤[4m2¤[24m0
## ¤[90m4¤[39m Gentoo  Biscoe           52.2          17.1               228        ¤[4m5¤[24m400 male   ¤[4m2¤[24m0
## ¤[90m5¤[39m Gentoo  Biscoe           50.8          15.7               226        ¤[4m5¤[24m200 male   ¤[4m2¤[24m0
## ¤[90m6¤[39m Gentoo  Biscoe           51.1          16.5               225        ¤[4m5¤[24m250 male   ¤[4m2¤[24m0
```

Exercício 15

Usando o conjunto de dados `penguins`, amostre 70% das linhas aleatoriamente com `body_mass_g` menor que 5000 g, atribuindo o resultado a um novo objeto `penguins_sample` utilizando o formato tidyverse

01:30

Exercício 15

Solução

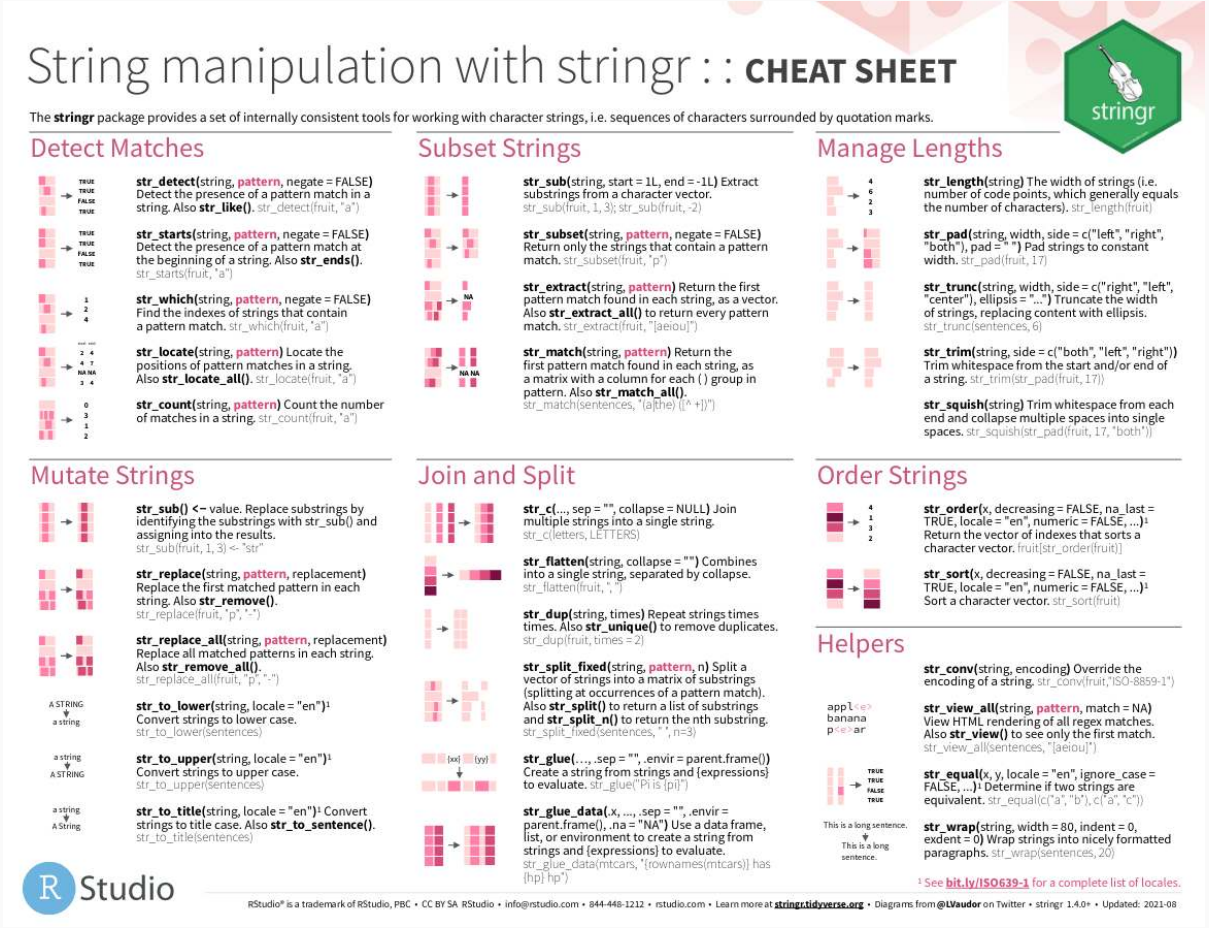
```
# exercicio 15
penguins_sample ← penguins %>%
  dplyr::filter(body_mass_g < 5000) %>%
  dplyr::slice_sample(prop = .7)
penguins_sample
```

```
## ¤[90m# A tibble: 192 × 8¤[39m
##    species   island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex     year
##    ¤[3m¤[90m<fct>¤[39m¤[23m     ¤[3m¤[90m<fct>¤[39m¤[23m           ¤[3m¤[90m<dbl>¤[39m¤[23m         ¤[3m¤[90m<d
## ¤[90m 1¤[39m Gentoo    Biscoe           47.2          13.7               214        ¤[4m4¤[24m925 female  ¤[4m2¤[
## ¤[90m 2¤[39m Adelie    Dream            39.6          18.1               186        ¤[4m4¤[24m450 male    ¤[4m2¤[
## ¤[90m 3¤[39m Gentoo    Biscoe           45.5          13.9               210        ¤[4m4¤[24m200 female  ¤[4m2¤[
## ¤[90m 4¤[39m Chinstrap Dream            50.8          18.5               201        ¤[4m4¤[24m450 male    ¤[4m2¤[
## ¤[90m 5¤[39m Adelie    Dream            41.1          17.5               190        ¤[4m3¤[24m900 male    ¤[4m2¤[
## ¤[90m 6¤[39m Chinstrap Dream            50.7          19.7               203        ¤[4m4¤[24m050 male    ¤[4m2¤[
## ¤[90m 7¤[39m Gentoo    Biscoe           45.8          14.6               210        ¤[4m4¤[24m200 female  ¤[4m2¤[
## ¤[90m 8¤[39m Adelie    Dream            40.6          17.2               187        ¤[4m3¤[24m475 male    ¤[4m2¤[
## ¤[90m 9¤[39m Adelie    Dream            39.5          16.7               178        ¤[4m3¤[24m250 female  ¤[4m2¤[
## ¤[90m10¤[39m Adelie    Biscoe           34.5          18.1               187        ¤[4m2¤[24m900 female  ¤[4m2¤[
## ¤[90m# … with 182 more rows¤[39m
```

stringr

9. stringr

Work with Strings Cheatsheet



String manipulation with stringr : : **CHEAT SHEET**

The **stringr** package provides a set of internally consistent tools for working with character strings, i.e. sequences of characters surrounded by quotation marks.

Detect Matches

str_detect(string, **pattern**, negate = FALSE) Detect the presence of a pattern match in a string. Also **str_like()**. str_detect(fruit, "a")

str_starts(string, **pattern**, negate = FALSE) Detect the presence of a pattern match at the beginning of a string. Also **str_ends()**. str_starts(fruit, "a")

str_which(string, **pattern**, negate = FALSE) Find the indexes of strings that contain a pattern match. str_which(fruit, "a")

str_locate(string, **pattern**) Locate the positions of pattern matches in a string. Also **str_locate_all()**. str_locate(fruit, "a")

str_count(string, **pattern**) Count the number of matches in a string. str_count(fruit, "a")

Subset Strings

str_sub(string, start = 1L, end = -1L) Extract substrings from a character vector. str_sub(fruit, 1, 3); str_sub(fruit, -2)

str_subset(string, **pattern**, negate = FALSE) Return only the strings that contain a pattern match. str_subset(fruit, "p")

str_extract(string, **pattern**) Return the first pattern match found in each string, as a vector. Also **str_extract_all()** to return every pattern match. str_extract(fruit, "[aeiou]")

str_match(string, **pattern**) Return the first pattern match found in each string, as a matrix with a column for each ( ) group in pattern. Also **str_match_all()**. str_match(sentences, "(a|the) ([^ +])")

Manage Lengths

str_length(string) The width of strings (i.e. number of code points, which generally equals the number of characters). str_length(fruit)

str_pad(string, width, side = c("left", "right", "both"), pad = " ") Pad strings to constant width. str_pad(fruit, 17)

str_trunc(string, width, side = c("right", "left", "center"), ellipsis = "...") Truncate the width of strings, replacing content with ellipsis. str_trunc(sentences, 6)

str_trim(string, side = c("both", "left", "right")) Trim whitespace from the start and/or end of a string. str_trim(str_pad(fruit, 17))

str_squish(string) Trim whitespace from each end and collapse multiple spaces into single spaces. str_squish(str_pad(fruit, 17, "both"))

Mutate Strings

str_sub() <– value. Replace substrings by identifying the substrings with str_sub() and assigning into the results. str_sub(fruit, 1, 3) <- "str"

str_replace(string, **pattern**, replacement) Replace the first matched pattern in each string. Also **str_remove()**. str_replace(fruit, "p", "-")

str_replace_all(string, **pattern**, replacement) Replace all matched patterns in each string. Also **str_remove_all()**. str_replace_all(fruit, "p", "-")

str_to_lower(string, locale = "en")[1] Convert strings to lower case. str_to_lower(sentences)

str_to_upper(string, locale = "en")[1] Convert strings to upper case. str_to_upper(sentences)

str_to_title(string, locale = "en")[1] Convert strings to title case. Also **str_to_sentence()**. str_to_title(sentences)

Join and Split

str_c(..., sep = "", collapse = NULL) Join multiple strings into a single string. str_c(letters, LETTERS)

str_flatten(string, collapse = "") Combines into a single string, separated by collapse. str_flatten(fruit, ", ")

str_dup(string, times) Repeat strings times times. Also **str_unique()** to remove duplicates. str_dup(fruit, times = 2)

str_split_fixed(string, **pattern**, n) Split a vector of strings into a matrix of substrings (splitting at occurrences of a pattern match). Also **str_split()** to return a list of substrings and **str_split_n()** to return the nth substring. str_split_fixed(sentences, " ", n=3)

str_glue(..., .sep = "", .envir = parent.frame()) Create a string from strings and {expressions} to evaluate. str_glue("Pi is {pi}")

str_glue_data(.x, ..., .sep = "", .envir = parent.frame(), .na = "NA") Use a data frame, list, or environment to create a string from strings and {expressions} to evaluate. str_glue_data(mtcars, "{rownames(mtcars)} has {hp} hp")

Order Strings

str_order(x, decreasing = FALSE, na_last = TRUE, locale = "en", numeric = FALSE, ...)[1] Return the vector of indexes that sorts a character vector. fruit[str_order(fruit)]

str_sort(x, decreasing = FALSE, na_last = TRUE, locale = "en", numeric = FALSE, ...)[1] Sort a character vector. str_sort(fruit)

Helpers

str_conv(string, encoding) Override the encoding of a string. str_conv(fruit,"ISO-8859-1")

str_view_all(string, **pattern**, match = NA) View HTML rendering of all regex matches. Also **str_view()** to see only the first match. str_view_all(sentences, "[aeiou]")

str_equal(x, y, locale = "en", ignore_case = FALSE, ...)[1] Determine if two strings are equivalent. str_equal(c("a", "b"), c("a", "c"))

str_wrap(string, width = 80, indent = 0, exdent = 0) Wrap strings into nicely formatted paragraphs. str_wrap(sentences, 20)

[1] See bit.ly/ISO639-1 for a complete list of locales.

RStudio® is a trademark of RStudio, PBC • CC BY SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more at stringr.tidyverse.org • Diagrams from @LVaudor on Twitter • stringr 1.4.0+ • Updated: 2021-08

[Work with Strings Cheatsheet]

9. stringr

Descrição

Pacote para a manipulação de strings ou caracteres

Exemplos: Correspondência de padrões, retirar e acrescentar espaços em branco, mudar maiúsculas e minúsculas

Pode ser utilizado em conjunto com o pacote `dplyr` para manejar colunas

Para funções mais específicas, recomenda-se usar o pacote stringi

9. stringr

Comprimento

```
# comprimento
stringr::str_length(string = "penguins")
```

```
## [1] 8
```

Substituir

```
# substituir
stringr::str_replace(string = "penguins", pattern = "i", replacement = "y")
```

```
## [1] "penguyns"
```

Separar

```
# separar
stringr::str_split(string = "p-e-n-g-u-i-n-s", pattern = "-", simplify = TRUE)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
```

9. stringr

Extrair pela posição

```
# extrair pela posicao
stringr::str_sub(string = "penguins", end = 3)
```

```
## [1] "pen"
```

Extrair por padrão

```
# extrair por padrao
stringr::str_extract(string = "penguins", pattern = "p")
```

```
## [1] "p"
```

9. stringr

Inserir espacos em branco

```
# inserir espaco em branco - esquerda
stringr::str_pad(string = "penguins", width = 10, side = "left")
```

```
## [1] "  penguins"
```

```
# inserir espaco em branco - direita
stringr::str_pad(string = "penguins", width = 10, side = "right")
```

```
## [1] "penguins  "
```

```
# inserir espaco em branco - ambos
stringr::str_pad(string = "penguins", width = 10, side = "both")
```

```
## [1] " penguins "
```

9. stringr

Remover espaços em branco

```
# remover espacos em branco - esquerda
stringr::str_trim(string = " penguins ", side = "left")
```

```
## [1] "penguins "
```

```
# remover espacos em branco - direta
stringr::str_trim(string = " penguins ", side = "right")
```

```
## [1] " penguins"
```

```
# remover espacos em branco - ambos
stringr::str_trim(string = " penguins ", side = "both")
```

```
## [1] "penguins"
```

9. stringr

Alterar minúsculas e maiúsculas

```
# minusculas
stringr::str_to_lower(string = "Penguins")
```

```
## [1] "penguins"
```

```
# maiusculas
stringr::str_to_upper(string = "penguins")
```

```
## [1] "PENGUINS"
```

```
# primeiro caracter maiusculo da primeira palavra
stringr::str_to_sentence(string = "palmer penGuins")
```

```
## [1] "Palmer penguins"
```

```
# primeiro caracter maiusculo de cada palavra
stringr::str_to_title(string = "palmer penGuins")
```

```
## [1] "Palmer Penguins"
```

9. stringr

Ordenar

```
# ordenar - crescente
stringr::str_sort(x = letters)
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s" "t" "u" "v" "w" "x" "y" "z"
```

```
# ordenar - decrescente
stringr::str_sort(x = letters, dec = TRUE)
```

```
##  [1] "z" "y" "x" "w" "v" "u" "t" "s" "r" "q" "p" "o" "n" "m" "l" "k" "j" "i" "h" "g" "f" "e" "d" "c" "b" "a"
```

9. stringr

Alterar valores das colunas

```
# alterar valores das colunas
penguins_stringr_valores ← penguins %>%
  dplyr::mutate(species = stringr::str_to_lower(species))
```

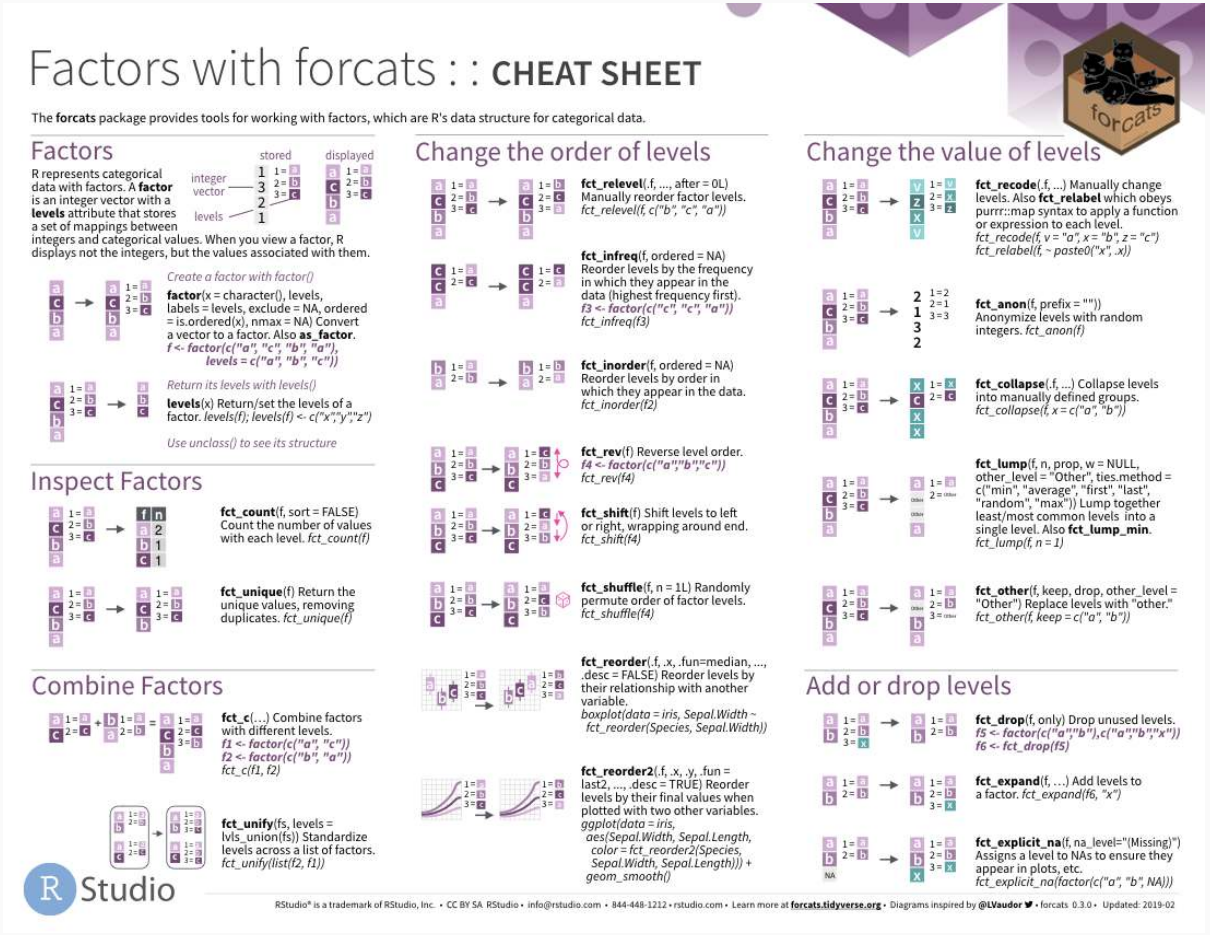
Alterar nome das colunas

```
# alterar nome das colunas
penguins_stringr_nomes ← penguins %>%
  dplyr::rename_with(stringr::str_to_title)
```

forcats

10. forcats

Factors with forcats Cheatsheet



[Factors with forcats Cheatsheet]

10. forcats

Descrição

Conjunto de ferramentas úteis para facilitar a manipulação de fatores e seus níveis

Exemplos: mudar a ordem dos níveis, mudar os valores dos níveis, adicionar e remover níveis, combinar múltiplos níveis

Pode ser utilizado em conjunto com o pacote `dplyr` para manejar colunas

10. forcats

Converter dados de string para fator

```
# converter dados de string para fator
forcats::as_factor(penguins_raw$Species) %>% head()
```

```
## [1] Adelie Penguin (Pygoscelis adeliae) Adelie Penguin (Pygoscelis adeliae) Adelie Penguin (Pygoscelis adeliae)
## [4] Adelie Penguin (Pygoscelis adeliae) Adelie Penguin (Pygoscelis adeliae) Adelie Penguin (Pygoscelis adeliae)
## Levels: Adelie Penguin (Pygoscelis adeliae) Gentoo penguin (Pygoscelis papua) Chinstrap penguin (Pygoscelis antarc
```

Mudar o nome dos níveis

```
# mudar o nome dos niveis
forcats::fct_recode(penguins$species, a = "Adelie", c = "Chinstrap", g = "Gentoo") %>% head()
```

```
## [1] a a a a a a
## Levels: a c g
```

10. forcats

Inverter os níveis

```
# inverter os niveis
forcats::fct_rev(penguins$species) %>% head()
```

```
## [1] Adelie Adelie Adelie Adelie Adelie Adelie
## Levels: Gentoo Chinstrap Adelie
```

Especificar a ordem dos níveis

```
# especificar a ordem dos niveis
forcats::fct_relevel(penguins$species, "Chinstrap", "Gentoo", "Adelie") %>% head()
```

```
## [1] Adelie Adelie Adelie Adelie Adelie Adelie
## Levels: Chinstrap Gentoo Adelie
```

10. forcats

Níveis pela ordem em que aparecem

```
# niveis pela ordem em que aparecem
forcats::fct_inorder(penguins$species) %>% head()
```

```
## [1] Adelie Adelie Adelie Adelie Adelie Adelie
## Levels: Adelie Gentoo Chinstrap
```

Ordem (decrescente) de frequência

```
# ordem (decrescente) de frequecia
forcats::fct_infreq(penguins$species) %>% head()
```

```
## [1] Adelie Adelie Adelie Adelie Adelie Adelie
## Levels: Adelie Gentoo Chinstrap
```

10. forcats

Agregação de níveis raros em um nível

```
# agregacao de niveis raros em um nivel
forcats::fct_lump(penguins$species) %>% head()
```

```
## [1] Adelie Adelie Adelie Adelie Adelie Adelie
## Levels: Adelie Gentoo Other
```

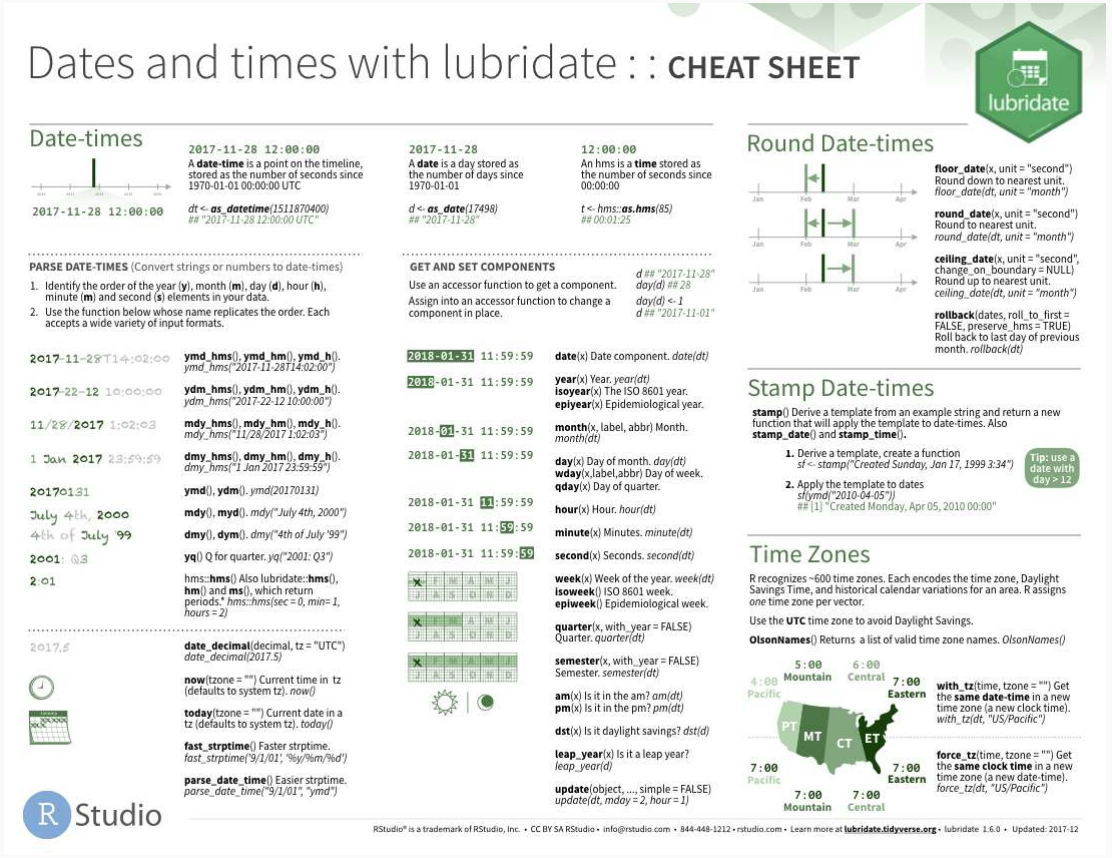
Transformar várias colunas em fator

```
# transformar varias colunas em fator
penguins_raw_multi_factor ← penguins_raw %>%
  dplyr::mutate(across(where(is.character), forcats::as_factor))
```

lubridate

11. lubridate

Dates and Times Cheatsheet



[Dates and Times Cheatsheet]

11. lubridate

Pacote à parte do tidyverse

Carregar sempre que for utilizar

```
# carregar
library(lubridate)
```

@allison_horst

11. lubridate

Descrição

Conjunto de funções para a manipulação de dados de data e horário

A manipulação desses dados não é intuitiva e muda dependendo do tipo de objeto de data e horário

O formato desses dados deve levar em consideração Fusos horários, anos bissextos, horários de verão, etc.

Exemplos: transformações de data/horário, componentes, arredondamentos, durações, períodos, intervalos, etc.

11. lubridate

Tipos de dados

- **Data**: tempo em dias, meses e anos `<date>`
- **Horário**: tempo dentro de um dia `<time>`
- **Data-horário**: tempo em um instante (data mais tempo) `<dttm>`

Para trabalhar exclusivamente com horários, podemos utilizar o pacote hms

Algumas letras possuem significados temporais, sendo abreviações de diferentes períodos em inglês:

- **y**ear (ano), **m**onth (mês)
- **w**eak (semana), **d**ay (dia)
- **h**our (hora), **m**inute (minuto) e **s**econd (segundo)

11. lubridate

Data e horários atuais

```
# extrair a data nesse instante
lubridate::today()
```

```
## [1] "2021-10-20"
```

```
# extrair a data e tempo nesse instante
lubridate::now()
```

```
## [1] "2021-10-20 17:04:52 -03"
```

11. lubridate

Existem três maneiras de criar um dado de data/horário

1. De um string ou caracter

2. De componentes individuais de data e horário

3. De um objeto de data/horário existente

11. lubridate

1. String ou caracter

Os dados de data/horário geralmente estão no formato de strings

Podemos transformar os dados especificando a ordem dos seus componentes usando as letras `y` (ano), `m` (mês) e `d` (dia)

```
# strings e numeros para datas
lubridate::dmy("03-03-2021")
```

```
## [1] "2021-03-03"
```

11. lubridate

1. String ou caracter

Essas funções também aceitam números sem aspas, além funcionarem em outros diversos formatos

```
# strings e numeros para datas
lubridate::dmy("03-Mar-2021")
lubridate::dmy(03032021)
lubridate::dmy("03032021")
lubridate::dmy("03/03/2021")
lubridate::dmy("03.03.2021")
```

11. lubridate

1. String ou caracter

Especificar horários e Fusos horários

```
# especificar horarios e fuso horario
lubridate::dmy_h("03-03-2021 13")
lubridate::dmy_hm("03-03-2021 13:32")
lubridate::dmy_hms("03-03-2021 13:32:01")
lubridate::dmy_hms("03-03-2021 13:32:01", tz = "America/Sao_Paulo")
```

11. lubridate

2. Componentes individuais de data e horário

```
# dados com componentes individuais
dados ← tibble::tibble(
  ano = c(2021, 2021, 2021),
  mes = c(1, 2, 3),
  dia = c(12, 20, 31),
  hora = c(2, 14, 18),
  minuto = c(2, 44, 55))
dados
```

```
## ¤[90m# A tibble: 3 × 5¤[39m
##     ano   mes   dia  hora minuto
##   ¤[3m¤[90m<dbl>¤[39m¤[23m ¤[3m¤[90m<dbl>¤[39m¤[23m ¤[3m¤[90m<dbl>¤[39m¤[23m ¤[3m¤[90m<dbl>¤[39m¤[23m  ¤[3m¤[90
## ¤[90m1¤[39m  ¤[4m2¤[24m021     1    12     2      2
## ¤[90m2¤[39m  ¤[4m2¤[24m021     2    20    14     44
## ¤[90m3¤[39m  ¤[4m2¤[24m021     3    31    18     55
```

11. lubridate

2. Componentes individuais de data e horário

```
# dados com componentes individuais
dados %>%
  dplyr::mutate(data = lubridate::make_datetime(ano, mes, dia, hora, minuto))
```

```
## ¤[90m# A tibble: 3 × 6¤[39m
##     ano   mes   dia  hora minuto data
##   ¤[3m¤[90m<dbl>¤[39m¤[23m ¤[3m¤[90m<dbl>¤[39m¤[23m ¤[3m¤[90m<dbl>¤[39m¤[23m ¤[3m¤[90m<dbl>¤[39m¤[23m ¤[3m¤[90m<dbl>¤[39m¤[23m  ¤[3m¤[90
## ¤[90m1¤[39m  ¤[4m2¤[24m021     1    12     2      2 2021-01-12 ¤[90m02:02:00¤[39m
## ¤[90m2¤[39m  ¤[4m2¤[24m021     2    20    14     44 2021-02-20 ¤[90m14:44:00¤[39m
## ¤[90m3¤[39m  ¤[4m2¤[24m021     3    31    18     55 2021-03-31 ¤[90m18:55:00¤[39m
```

3. Objeto de data/horário existente

```
# data para data-horario
lubridate::as_datetime(today())
```

```
## [1] "2021-10-20 UTC"
```

```
# data-horario para data
lubridate::as_date(now())
```

```
## [1] "2021-10-20"
```

11. lubridate

Funções para acessar e definir componentes individuais de datas e horários

- `year()`: acessa o ano
- `month()`: acessa o mês
- `day()`: acessa o dia
- `yday()`: acessa o dia do ano
- `mday()`: acessa o dia do mês
- `wday()`: acessa o dia da semana
- `hour()`: acessa as horas
- `minute()`: acessa os minutos
- `second()`: acessa os segundos

11. lubridate

Funções para acessar e definir componentes individuais de datas e horários

```
# extrair
lubridate::year(now())
lubridate::month(now())
lubridate::month(now(), label = TRUE)
lubridate::day(now())
lubridate::wday(now())
lubridate::wday(now(), label = TRUE)
lubridate::second(now())
```

11. lubridate

Inclusam de informações de datas e horários.

```
# data
data ← dmy_hms("04-03-2021 01:04:56")

# incluir
lubridate::year(data) ← 2020
lubridate::month(data) ← 01
lubridate::hour(data) ← 13
```

```
# incluir varios valores
update(data, year = 2020, month = 1, mday = 1, hour = 1)
```

```
## [1] "2020-01-01 01:04:56 UTC"
```

11. lubridate

Operações com datas

Muitas vezes precisamos fazer operações com datas, como subtração, adição e divisão. Para tanto, é preciso entender três classes importantes que representam intervalos de tempo

1. **Durações**: representam um número exato de segundos

2. **Períodos**: representam unidades humanas como semanas e meses

3. **Intervalos**: representam um ponto inicial e final

11. lubridate

1. Durações

Registram o intervalo de tempo em segundos, com alguma unidade de tempo maior entre parênteses

Funções

- `duration()`: cria data em duração
- `as.duration()`: converte datas em duração
- `dyears()`: duração de anos
- `dmonths()`: duração de meses
- `dweeks()`: duração de semanas
- `ddays()`: duração de dias
- `dhours()`: duração de horas
- `dminutes()`: duração de minutos
- `dseconds()`: duração de segundos

11. lubridate

1. Durações

Subtração de datas e conversão para duração

```
# subtracao de datas
tempo_estudando_r ← lubridate::today() - lubridate::dmy("30-11-2011")

# conversao para duracao
tempo_estudando_r_dur ← lubridate::as.duration(tempo_estudando_r)
```

11. lubridate

1. Durações

Criando durações

```
# criando duracoes
lubridate::duration(90, "seconds")
```

```
## [1] "90s (~1.5 minutes)"
```

```
lubridate::duration(1.5, "minutes")
```

```
## [1] "90s (~1.5 minutes)"
```

```
lubridate::duration(1, "days")
```

```
## [1] "86400s (~1 days)"
```

11. lubridate

1. Durações

Transformação das durações

```
# transformacao da duracao
lubridate::dseconds(100)
lubridate::dminutes(100)
lubridate::dhours(100)
lubridate::ddays(100)
lubridate::dweeks(100)
lubridate::dyears(100)
```

11. lubridate

1. Durações

Operações com durações

```
# somando duracoes a datas
lubridate::today() + lubridate::ddays(1)
```

```
## [1] "2021-10-21"
```

```
# subtraindo duracoes de datas
lubridate::today() - lubridate::dyears(1)
```

```
## [1] "2020-10-19 18:00:00 UTC"
```

```
# multiplicando duracoes
2 * dyears(2)
```

```
## [1] "126230400s (~4 years)"
```

11. lubridate

2. Períodos

São extensões de tempo não fixados em segundos como as durações, mas flexíveis, como dias, semanas, meses ou anos

Funções

- `period()`: cria data em período
- `as.period()`: converte datas em período
- `seconds()`: período em segundos
- `minutes()`: período em minutos
- `hours()`: período em horas
- `days()`: período em dias
- `weeks()`: período em semanas
- `months()`: período em meses
- `years()`: período em anos

2. Períodos

Criando períodos

```
# criando periodos
period(c(90, 5), c("second", "minute"))
```

```
## [1] "5M 90S"
```

```
period(c(3, 1, 2, 13, 1), c("second", "minute", "hour", "day", "week"))
```

```
## [1] "20d 2H 1M 3S"
```

2. Períodos

Transformações de períodos

```
# transformacao de periodos
lubridate::seconds(100)
lubridate::minutes(100)
lubridate::hours(100)
lubridate::days(100)
lubridate::weeks(100)
lubridate::years(100)
```

11. lubridate

2. Períodos

Operações com os períodos

```
# somando datas
lubridate::today() + lubridate::weeks(10)
```

```
## [1] "2021-12-29"
```

```
# subtraindo datas
lubridate::today() - lubridate::weeks(10)
```

```
## [1] "2021-08-11"
```

```
# criando datas recorrentes
lubridate::today() + lubridate::weeks(0:10)
```

```
##  [1] "2021-10-20" "2021-10-27" "2021-11-03" "2021-11-10" "2021-11-17" "2021-11-24" "2021-12-01" "2021-12-08" "2021
## [11] "2021-12-29"
```

11. lubridate

3. Intervalos

Períodos de tempo limitados por duas datas, possuindo uma duração com um ponto de partida, que o faz preciso para determinar uma duração

Funções

- `interval()`: cria data em intervalo
- `%--%: cria data em intervalo
- `as.interval()`: converte datas em intervalo
- `int_start()`: acessa ou atribui data inicial de um intervalo
- `int_end()`: acessa ou atribui data final de um intervalo
- `int_length()`: comprimento de um intervalo em segundos
- `int_flip()`: inverte a ordem da data de início e da data de término em um intervalo
- `int_shift()`: desloca as datas de início e término de um intervalo
- `int_aligns()`: testa se dois intervalos compartilham um ponto final
- `int_standardize()`: garante que todos os intervalos sejam positivos
- `int_diff()`: retorna os intervalos que ocorrem entre os elementos de data/horário

11. lubridate

3. Intervalos

Criando intervalos

```
# criando duas datas - inicio de estudos do R e nascimento do meu filho
r_inicio ← lubridate::dmy("30-11-2011")
filho_nascimento ← lubridate::dmy("26-09-2013")
r_hoje ← lubridate::today()

# criando intervalos - interval
r_intervalo ← lubridate::interval(r_inicio, r_hoje)

# criando intervalos - interval %--%
filho_intervalo ← filho_nascimento %--% lubridate::today()
```

3. Intervalos

Operaçõpes com intervalos

```
# operacooes com intervalos
lubridate::int_start(r_intervalo)
lubridate::int_end(r_intervalo)
lubridate::int_length(r_intervalo)
lubridate::int_flip(r_intervalo)
lubridate::int_shift(r_intervalo, duration(days = 30))
```

3. Intervalos

Sobreposição entre dois intervalos

```
# verificar sobreposicao - int_overlaps
lubridate::int_overlaps(r_intervalo, filho_intervalo)
```

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

```
# verificar se intervalo esta contido
r_intervalo %within% filho_intervalo
```

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

```
filho_intervalo %within% r_intervalo
```

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

3. Intervalos

Calcular quantos períodos existem dentro de um intervalo

```
# periodos dentro de um intervalo - anos
r_intervalo / lubridate::years()
```

```
## [1] 9.887671
```

```
r_intervalo %/% lubridate::years()
```

```
## [1] 9
```

```
# periodos dentro de um intervalo - dias e semandas
filho_intervalo / lubridate::days()
```

```
## [1] 2946
```

```
filho_intervalo / lubridate::weeks()
```

```
## [1] 420.8571
```

3. Intervalos

Transformações dos dados para períodos

```
# tempo total estudando R
lubridate::as.period(r_intervalo)
```

```
## [1] "9y 10m 20d 0H 0M 0S"
```

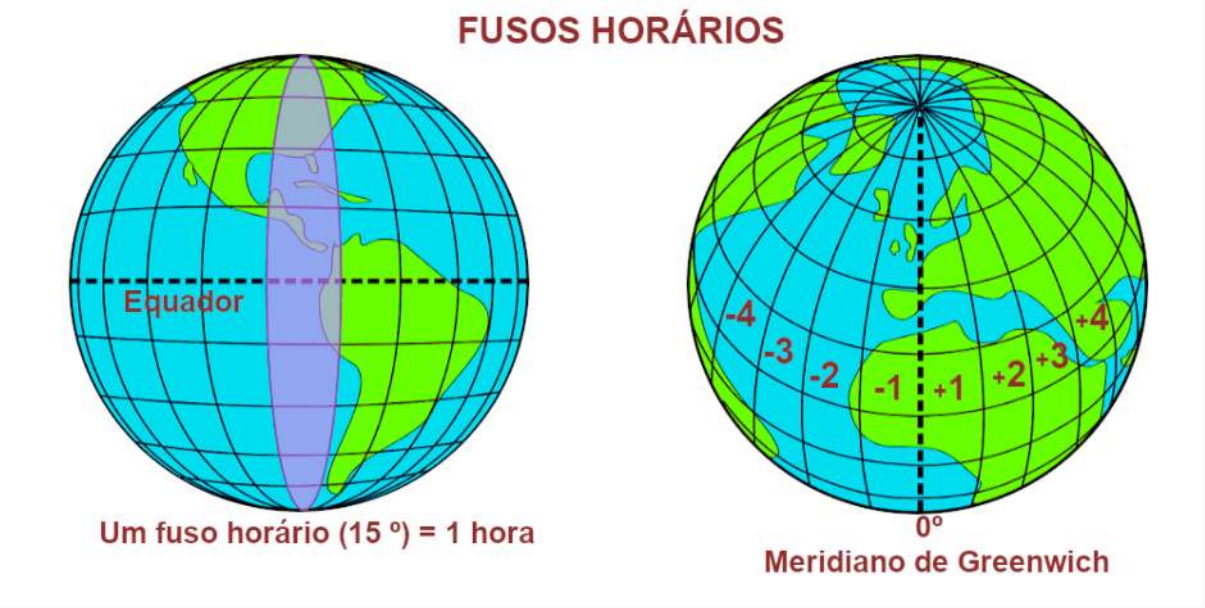
```
# idade do meu filho
lubridate::as.period(filho_intervalo)
```

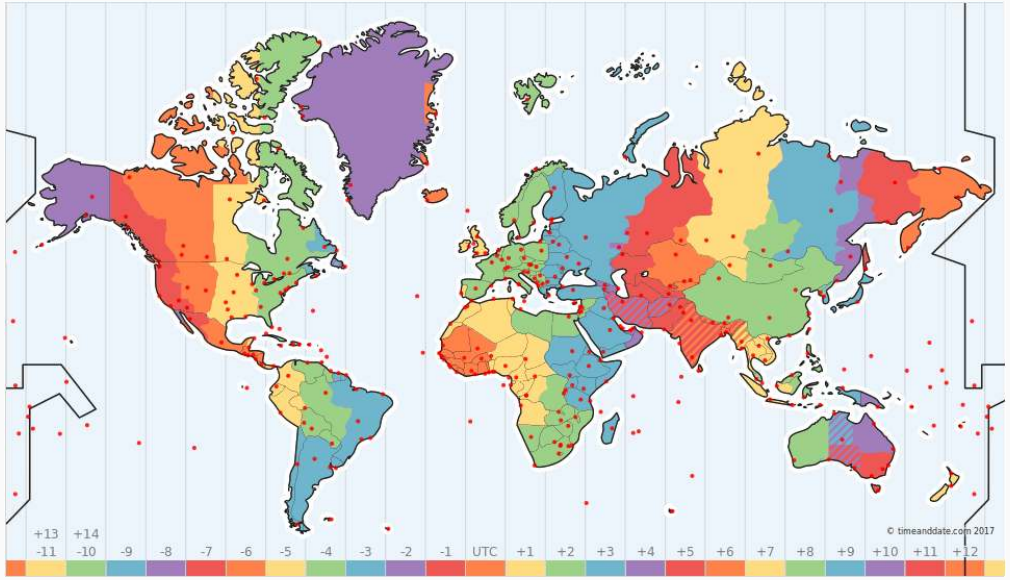
```
## [1] "8y 0m 24d 0H 0M 0S"
```

11. lubridate

Fusos horários

Tendem a ser um **fator complicador** quando precisamos analisar informações **instantâneas de tempo (horário)** de outras partes do planeta, ou mesmo fazer conversões dos horários





[Fuso horário, como funciona](), [Paradigma dos fusos horários e suas ocasionalidades]()

11. lubridate

Fusos horários

Descobrir o fuso horário que o R está considerando

```
# fuso horario no r
Sys.timezone()
```

```
## [1] "America/Sao_Paulo"
```

Listagem dos nomes dos fusos horários

```
# verificar os fusos horarios
length(OlsonNames())
```

```
## [1] 607
```

```
head(OlsonNames())
```

```
## [1] "Africa/Abidjan"     "Africa/Accra"       "Africa/Addis_Ababa" "Africa/Algiers"     "Africa/Asmara"      "Afri
```

11. lubridate

Fusos horários

Perguntar que horas são em outra parte do globo ou fazer as conversões

```
# que horas sao em ...
lubridate::with_tz(lubridate::now(), tzone = "America/Sao_Paulo")
```

```
## [1] "2021-10-20 17:04:52 -03"
```

```
lubridate::with_tz(lubridate::now(), tzone = "GMT")
```

```
## [1] "2021-10-20 20:04:52 GMT"
```

```
lubridate::with_tz(lubridate::now(), tzone = "Europe/Berlin")
```

```
## [1] "2021-10-20 22:04:52 CEST"
```

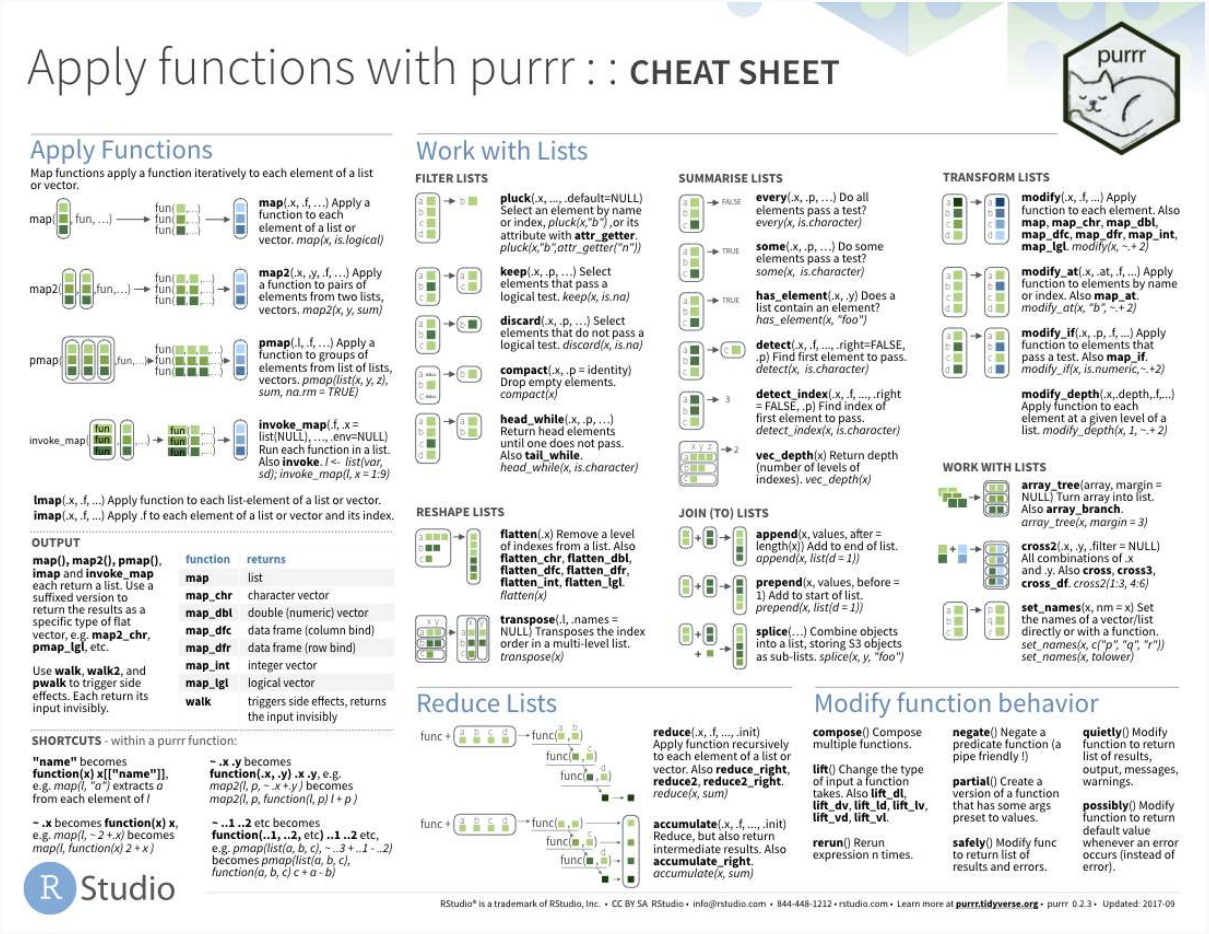
```
# alterar o fuso sem mudar a hora
lubridate::force_tz(lubridate::now(), tzone = "GMT")
```

```
## [1] "2021-10-20 17:04:52 GMT"
```

purrr

12. purrr

Apply functions with purrr Cheatsheet



[Apply functions with purrr Cheatsheet]

12. purrr

Descrição

Implementa a **Programação Funcional no R**, fornecendo um conjunto completo e consistente de ferramentas para trabalhar com funções e vetores

Permite substituir muitos *loops* for por um código mais sucinto e fácil de ler

12. purrr

Loop

Um loop for pode ser entendido como uma iteração: um bloco de códigos é repetido mudando um contador de uma lista de possibilidades

```
# loop for
for(i in 1:10){
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
```

12. purrr

Programação funcional

```
# loop for com map
purrr::map(.x = 1:10, .f = print)
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10

## [[1]]
## [1] 1
##
## [[2]]
## [1] 2
##
## [[3]]
```

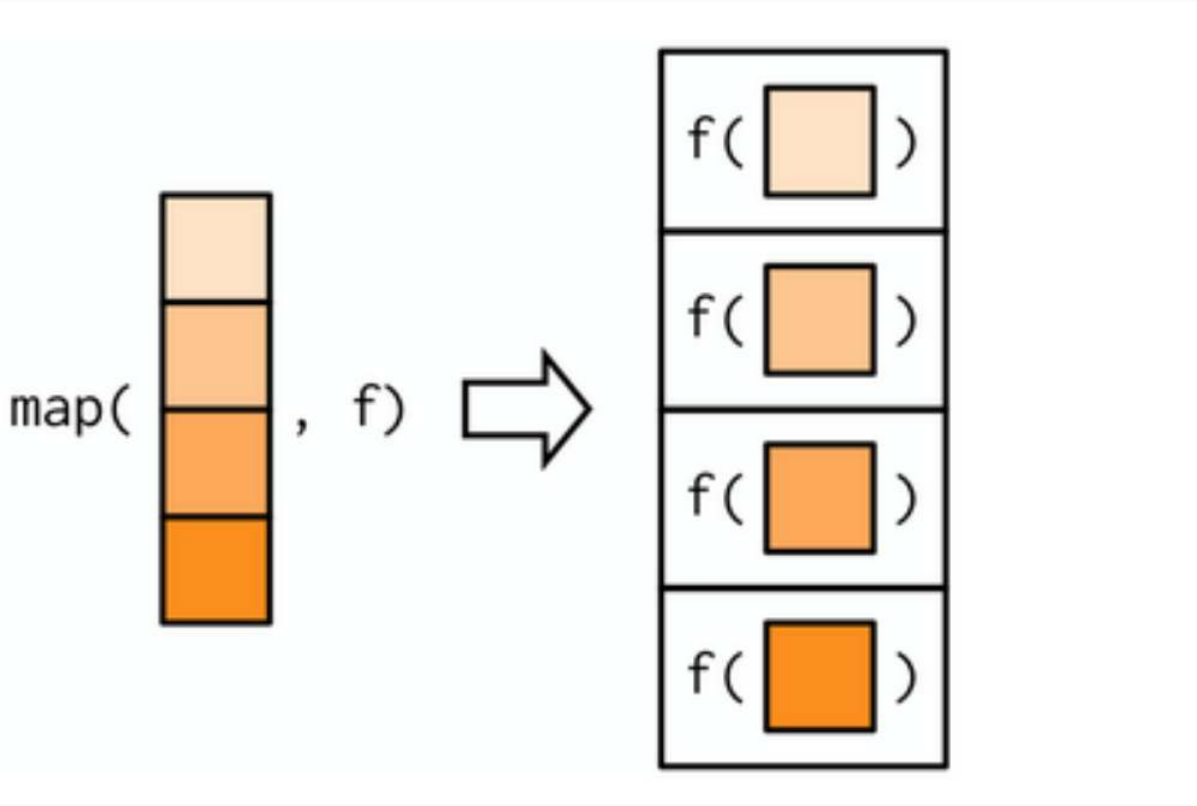
12. purrr

Estrutura

`map(.x, .f)`

`.x`: uma vetor, lista ou data frame

`.f`: uma função



12. purrr

Exemplo: aplicar a função `sum()` para somar os valores de vários elementos de uma lista

```
# lista
x ← list(1:5, c(4, 5, 7), c(1, 1, 1), c(2, 2, 2, 2, 2))

# funcao map
purrr::map(x, sum)
```

```
## [[1]]
## [1] 15
##
## [[2]]
## [1] 16
##
## [[3]]
## [1] 3
##
## [[4]]
## [1] 10
```

Tipos de retorno

Diferentes funções retornam diferentes estruturas de dados

- `map()`: retorna uma lista
- `map_chr()`: retorna um vetor de strings
- `map_dbl()`: retorna um vetor numérico (double)
- `map_int()`: retorna um vetor numérico (integer)
- `map_lgl()`: retorna um vetor lógico
- `map_dfr()`: retorna um data frame (por linhas)
- `map_dfc()`: retorna um data frame (por colunas)

12. purrr

Tipos de retorno

```
# variacoes da funcao map
purrr::map_dbl(x, sum)
```

```
## [1] 15 16  3 10
```

```
purrr::map_chr(x, paste, collapse = " ")
```

```
## [1] "1 2 3 4 5" "4 5 7"     "1 1 1"     "2 2 2 2 2"
```

Duas ou mais listas em paralelo

```
# listas
x ← list(3, 5, 0, 1)
y ← list(3, 5, 0, 1)
z ← list(3, 5, 0, 1)

# map2* duas litas
purrr::map2_dbl(x, y, prod)
```

```
## [1]  9 25  0  1
```

```
# pmap muitas listas
purrr::pmap_dbl(list(x, y, z), prod)
```

```
## [1]  27 125   0   1
```

12. purrr

Implementar rotinas de manipulação e análise de dados

```
# resumo dos dados
penguins %>%
  dplyr::select(where(is.numeric)) %>%
  tidyr::drop_na() %>%
  purrr::map_dbl(mean)
```

```
##    bill_length_mm     bill_depth_mm flipper_length_mm       body_mass_g
##          43.92193          17.15117         200.91520        4201.75439
##              year
##       2008.02924
```

Implementar rotinas de manipulação e análise de dados

```
# analise dos dados
penguins %>%
  dplyr::group_split(island, species) %>%
  purrr::map(~ lm(bill_depth_mm ~ bill_length_mm, data = .x)) %>%
  purrr::map(summary) %>%
  purrr::map("r.squared")
```

```
## [[1]]
## [1] 0.2192052
##
## [[2]]
## [1] 0.4139429
##
## [[3]]
## [1] 0.2579242
##
## [[4]]
## [1] 0.4271096
##
## [[5]]
```

12. purrr

Material

[Webinar de purrr avançado](#) - Curso-R

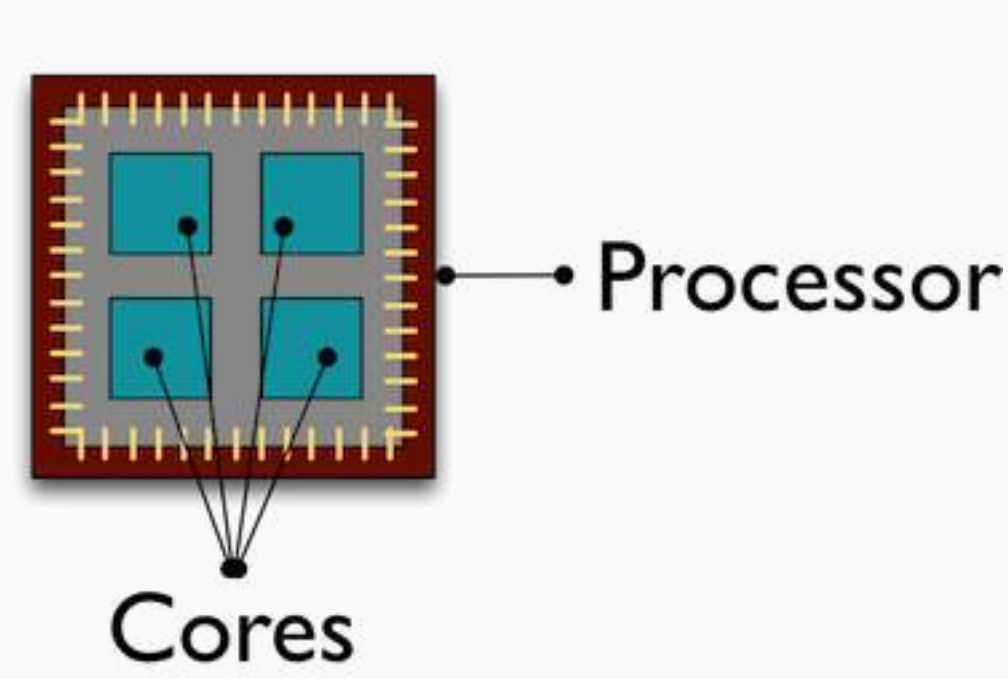
[purrr tutorial](#) - Jennifer (Jenny) Bryan

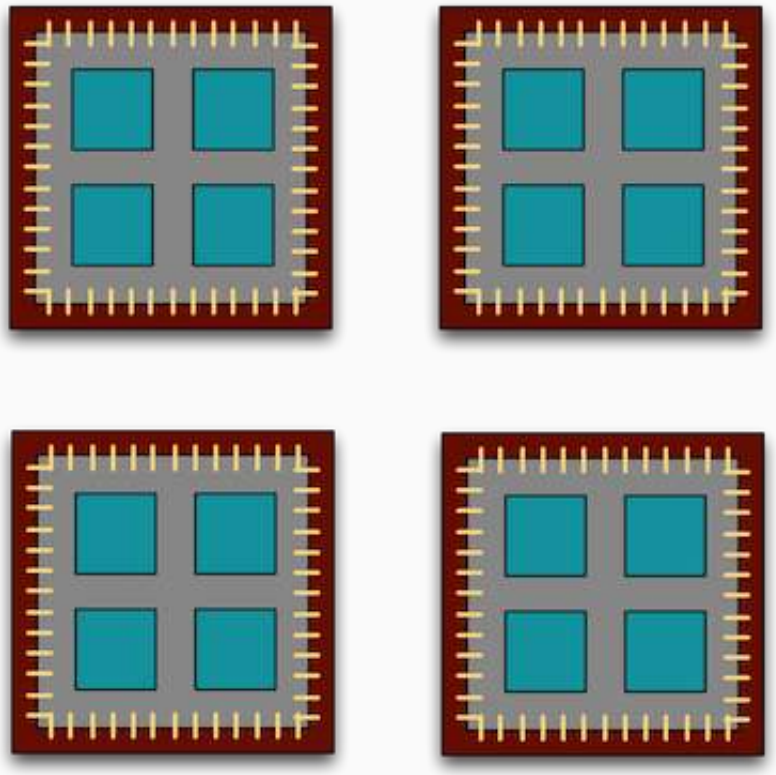
furrr

12. furrr

Programação funcional paralelizada

Faz o mapeamento em paralelo utilizando os *cores* do processador





[Quick Intro to Parallel Computing in R]

12. furrr

Programação funcional paralelizada

Número de *cores* do seu computador

```
# carregar
library(parallel)

# numero de cores
parallel::detectCores()
```

Dúvidas?

Maurício Vancine

Contatos:

mauricio.vancine@gmail.com

@mauriciovancine

mauriciovancine

mauriciovancine.github.io

Slides criados via pacote xaringan e tema Metropolis. Animação dos sapos por @probzz.