XXX Semana de Estudos da Ecologia

Introdução à linguagem R: manipulação e visualização de dados

4 Introdução ao tidyverse

Maurício Vancine
Helena Oliveira
Lucas Almeida

07/11/2019


tidyverse
www.rstudio.com

4 Introdução ao tidyverse

Tópicos

4.1 tidyverse

4.2 magrittr (pipe - %>%)

4.3 readr

4.4 readxl e writexl

4.5 tibble

4.6 tidyr

4.7 dplyr

4 Introdução ao tidyverse

Script

`script_aula_04.R`


tidyverse
www.rstudio.com

4.1 tidyverse

O tidyverse é um **pacote** com a função de **instalar** e **carregar** outros pacotes

O **conjunto** desses pacotes forma o **tidyverse**

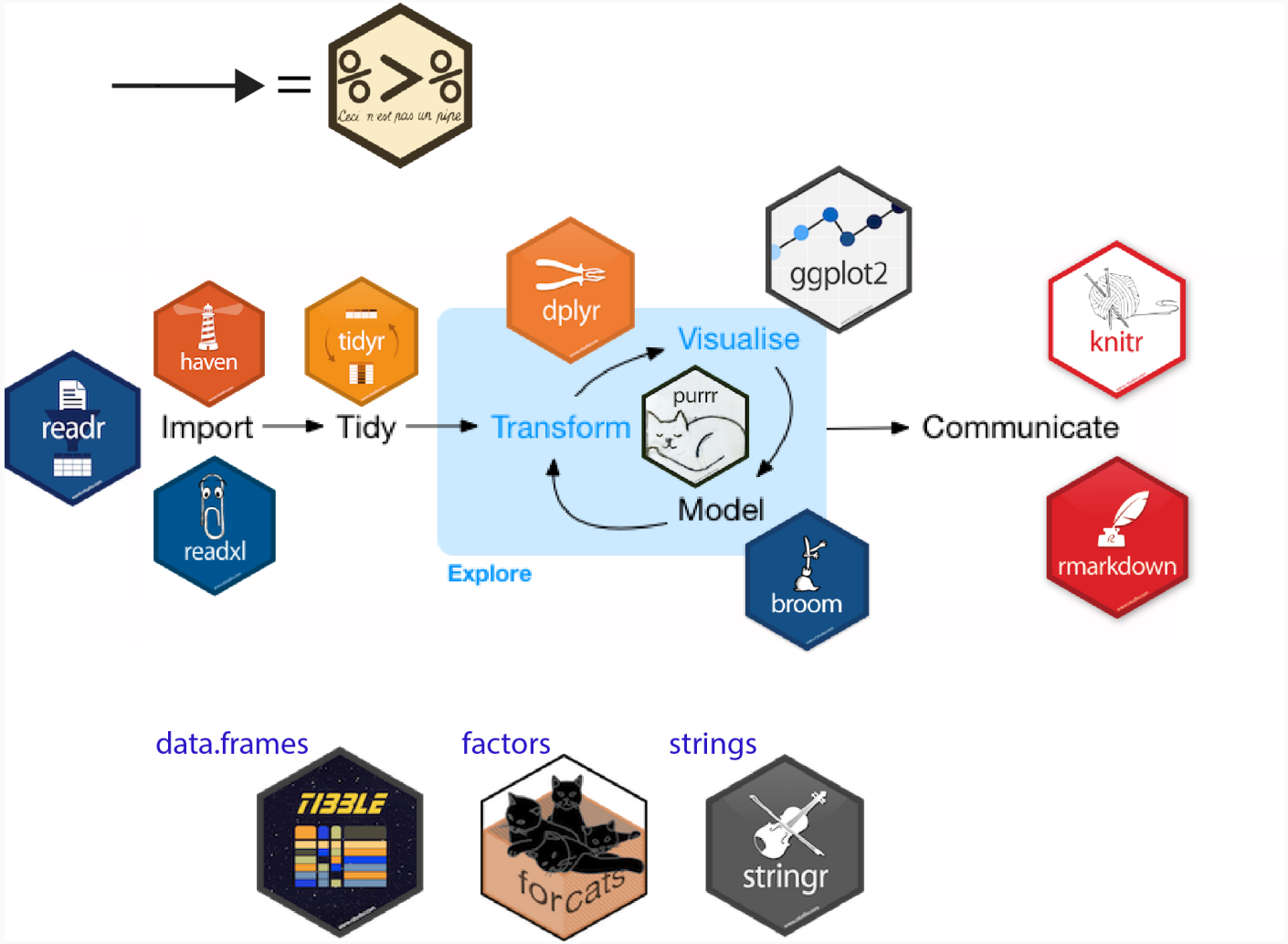
É considerado um “universo” à parte do R, pois todas as suas **ferramentas** possuem formas de uso consistentes e **funcionam** muito bem em conjunto

Seu uso é mais voltado para a **Ciência de Dados**

E depois que vocês **aprenderem**, nunca mais usaram o R de outra forma...

Iniciativa Vingadores do R

4.1 tidyverse



4.1 tidyverse

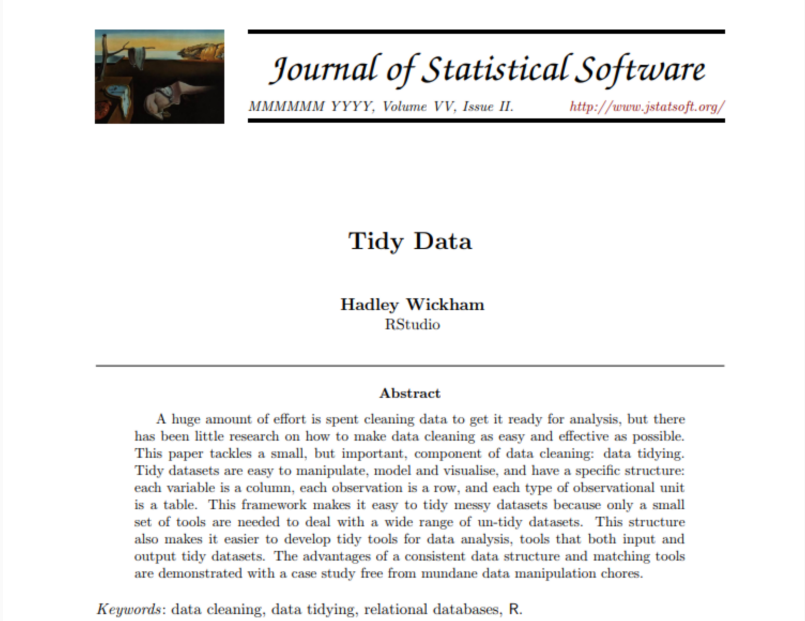
O idealizador foi o **Hadley Wickham** e atualmente **muitas pessoas** têm contribuído para sua expansão

[*] http://hadley.nz/

4.1 tidyverse

Artigo: **Tidy Data (2014)** - *Journal of Statistics Software*

Hadley Wickham



Journal of Statistical Software
MMMMMM YYYY, Volume VV, Issue II. *http://www.jstatsoft.org/*

Tidy Data

Hadley Wickham
RStudio

Abstract

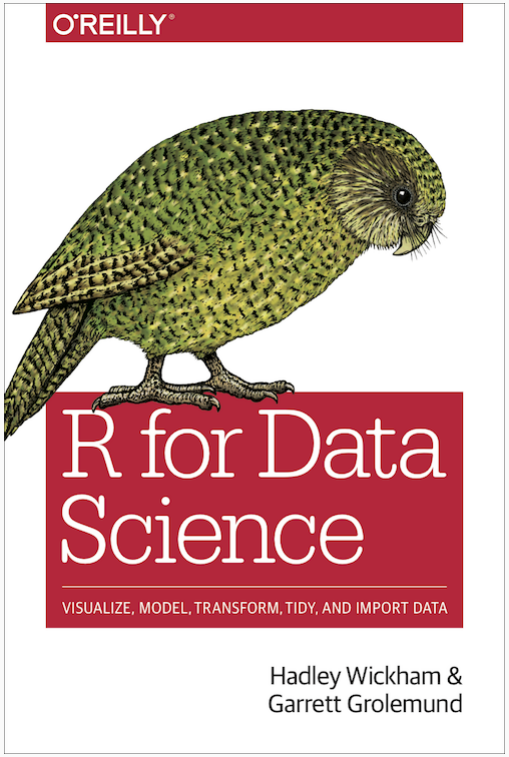
A huge amount of effort is spent cleaning data to get it ready for analysis, but there has been little research on how to make data cleaning as easy and effective as possible. This paper tackles a small, but important, component of data cleaning: data tidying. Tidy datasets are easy to manipulate, model and visualise, and have a specific structure: each variable is a column, each observation is a row, and each type of observational unit is a table. This framework makes it easy to tidy messy datasets because only a small set of tools are needed to deal with a wide range of un-tidy datasets. This structure also makes it easier to develop tidy tools for data analysis, tools that both input and output tidy datasets. The advantages of a consistent data structure and matching tools are demonstrated with a case study free from mundane data manipulation chores.

Keywords: data cleaning, data tidying, relational databases, R.

[*] http://vita.had.co.nz/papers/tidy-data.pdf

4.1 tidyverse

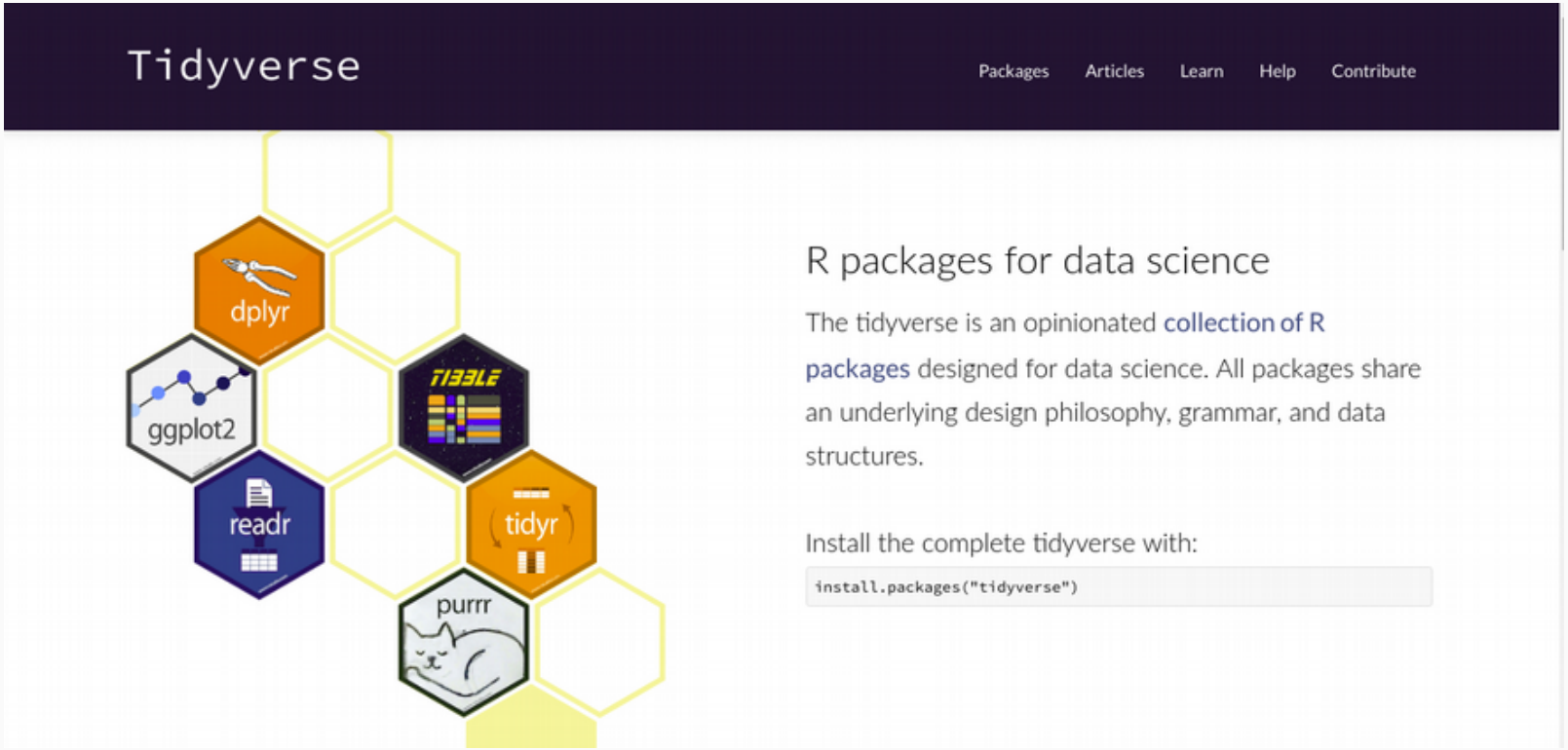
Livro: **R for Data Science (2017)**



[*] https://r4ds.had.co.nz/

4.1 tidyverse

Sites



[*] https://www.tidyverse.org/

4.1 tidyverse

Sites

RPubs brought to you by RStudio

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.

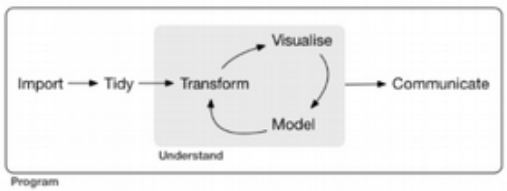


Diagrama de ferramentas para Data Science em (Grolemund,2017)

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

Passo 1: Formulação da pergunta de pesquisa

[*] https://rpubs.com/modelthinkingbr/dados

4.1 tidyverse

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

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

4.1 tidyverse

Notem a saída do carregamento do pacote

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

4.1 tidyverse

IMPORTANTE

Todas as funções dos pacotes atrelados ao **tidyverse** usam `_` para separar os nomes internos das funções (snack_code)

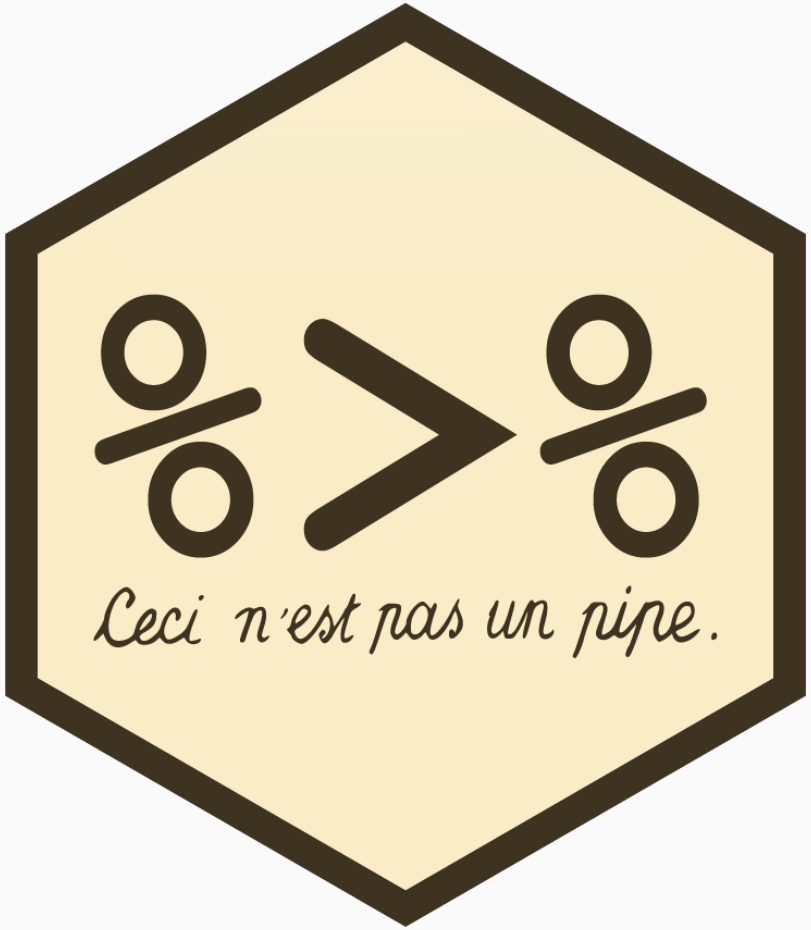
`read_csv`

`read_xlsx`

`as_tibble`

`left_join`

`group_by`


%>%
Ceci n'est pas un pipe.

4.2 magrittr (pipe - %>%)

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”*

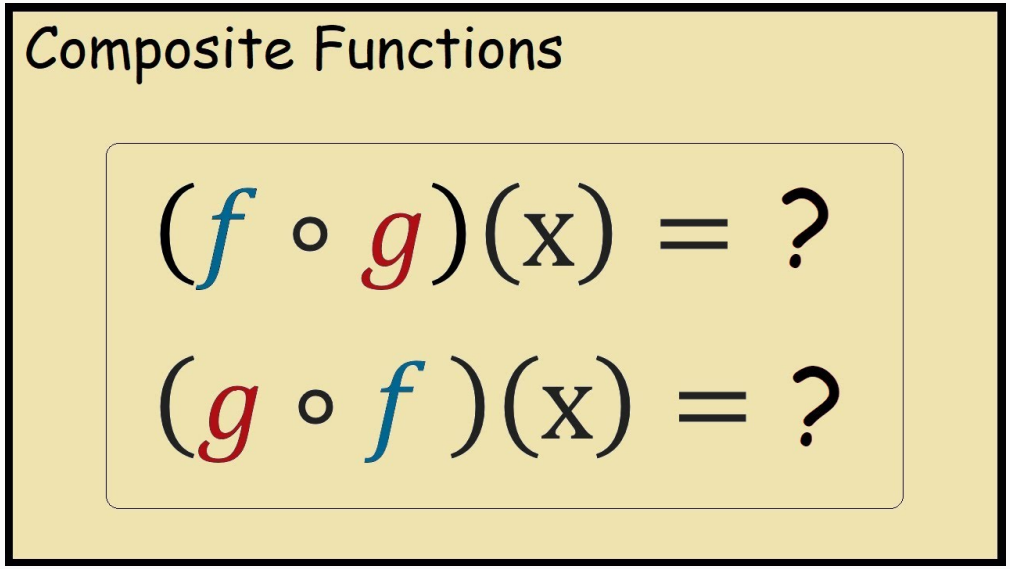
O operador pipe é o **%>%** (atalho: `crtl + shift + M`)

4.2 magrittr (pipe - %>%)

Atalho: `crtl + shift + M`

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

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



4.2 magrittr (pipe - %>%)

Atalho: `crtl + shift + M`

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

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

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

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

4.2 magrittr (pipe - %>%)

Atalho: `crtl + shift + M`

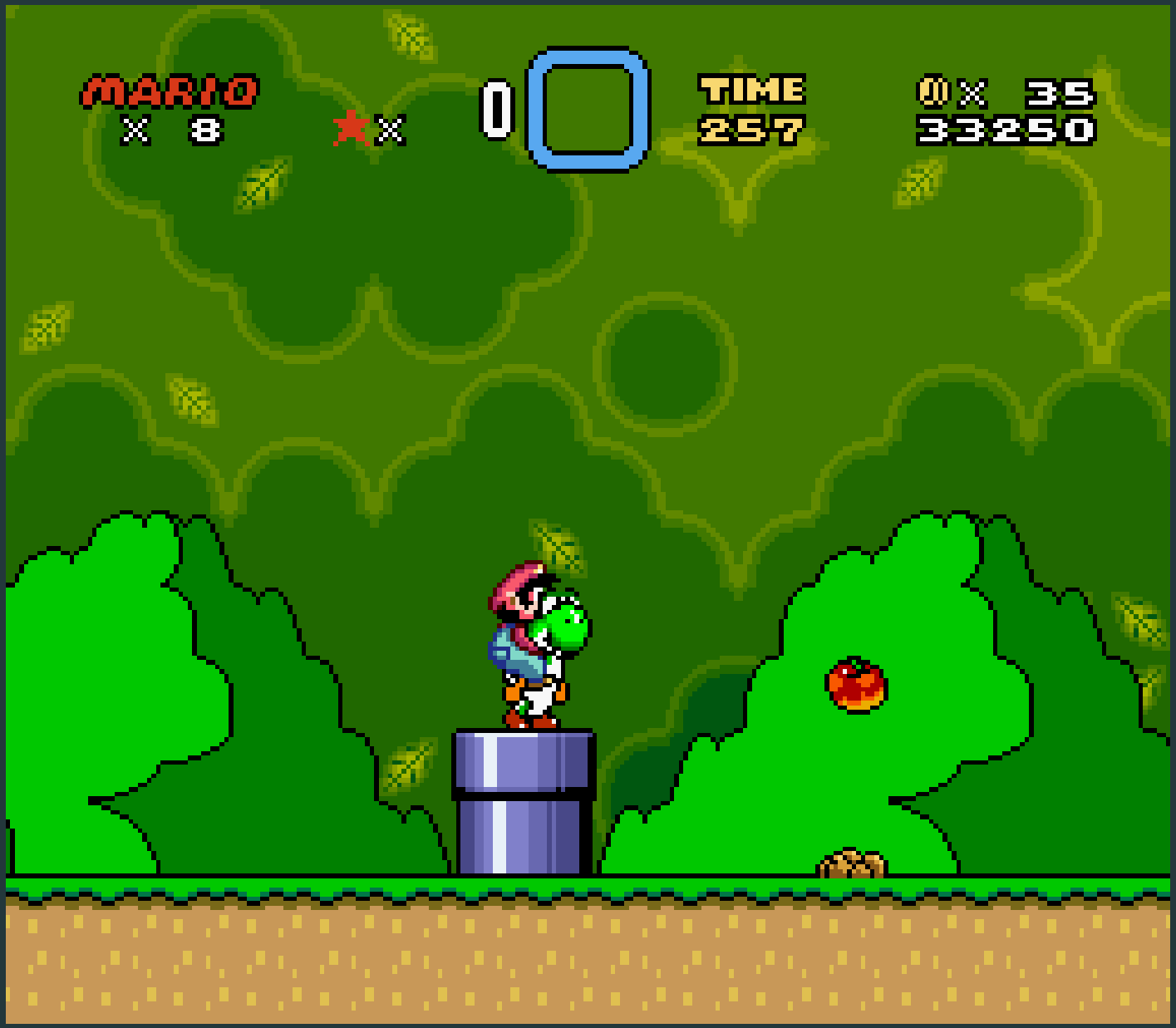
```
# fixar amostragem
set.seed(42)

# 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
```


MARIO
X 8
0
TIME
257
0X 35
33250

Exercícios

Exercício 12

Reescreva as operações utilizando pipes `%>%`:

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

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

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

Exercício 12

Solução

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

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

Exercício 12

Solução

```r
# 2.
sqrt(abs(rnorm(100)))

100 %>%
  rnorm %>%
  abs %>%
  sqrt
```

Exercício 12

Solução

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

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

Dúvidas?


readr
www.rstudio.com

4.3 readr

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

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

Essas funções fornecem **medidores de progresso** (barra do tempo de leitura dos dados)

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

A classe do objeto atribuído é **tibble** (data frame lv2)

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

Download de dados do GitHub

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

Eu mesmo et al. (2018)

Formato .csv

```
# diretorio
setwd("/home/mude/data/github/minicurso-r-sebio-2019/03_dados")
```

```
# import sites
si <- readr::read_csv("ATLANTIC_AMPHIBIANS_sites.csv")
si
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   reference_number = col_double(),
##   species_number = col_double(),
##   month_start = col_double(),
##   year_start = col_double(),
##   month_finish = col_double(),
##   year_finish = col_double(),
##   effort_months = col_double(),
##   latitude = col_double(),
```

4.3 readr

Formato .txt

```
# diretorio
setwd("/home/mude/data/github/minicurso-r-sebio-2019/03_dados")
```

```
# import sites
si <- readr::read_tsv("ATLANTIC_AMPHIBIANS_sites.txt")
si
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   reference_number = col_double(),
##   species_number = col_double(),
##   month_start = col_double(),
##   year_start = col_double(),
##   month_finish = col_double(),
##   year_finish = col_double(),
##   effort_months = col_double(),
##   latitude = col_double(),
```


readxl
www.rstudio.com

4.4 readxl e writexl

São pacotes à parte do **tidyverse**

```
install.packages("readxl")
library("readxl")
```

```
install.packages("writexl")
library("writexl")
```

4.4 readxl e writexl

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

As funções **read.xlsx()** e **read.xlsx2()** são substituídas pelas funções **read_xlsx()** e **read_xlsx2()**

Essas funções fornecem **medidores de progresso** (barra do tempo de leitura dos dados)

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

A classe do objeto atribuído é **tibble** (data frame lv2)

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

4.4 readxl e writexl

Formato .xlsx

```
# diretorio
setwd("/home/mude/data/github/minicurso-r-sebio-2019/03_dados")
```

```
# import sites
si <- readxl::read_xlsx("ATLANTIC_AMPHIBIANS_sites.xlsx")
si
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   reference_number = col_double(),
##   species_number = col_double(),
##   month_start = col_double(),
##   year_start = col_double(),
##   month_finish = col_double(),
##   year_finish = col_double(),
##   effort_months = col_double(),
##   latitude = col_double(),
```

Dúvidas?


TIBBLE
www.rstudio.com

4.5 tibble

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

É o **formato** necessário 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 tipo *numbers(int, dbl)*, *character(chr)*, *logical(lgl)* ou *factor(fctr)*

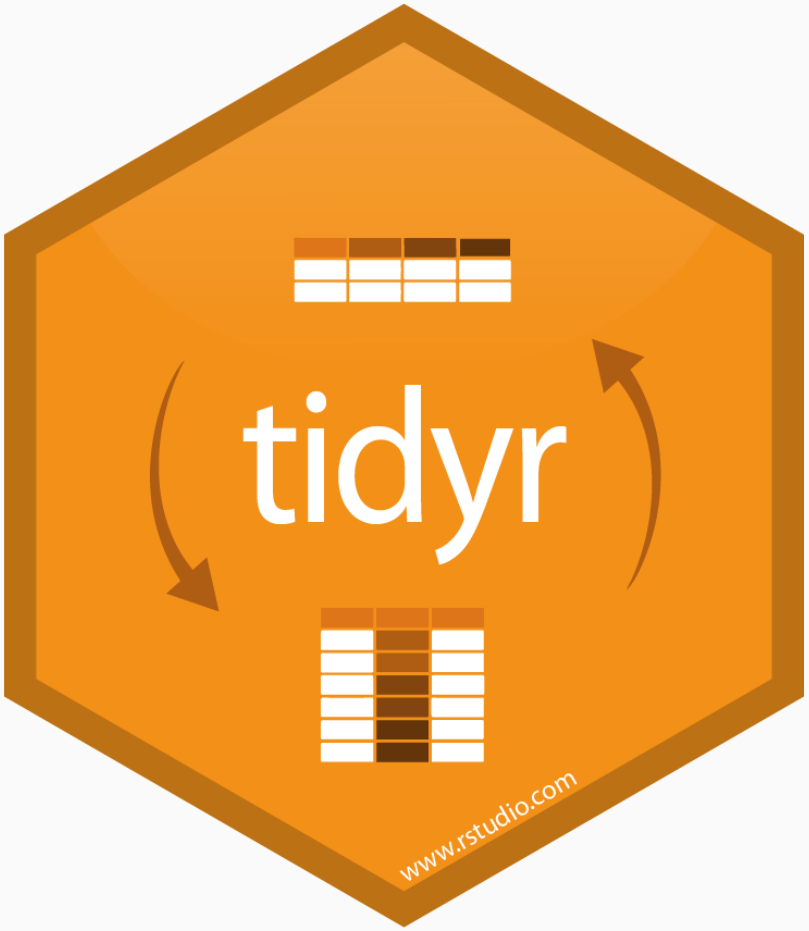
4.5 tibble

Descrição dos modos das colunas através da função `glimpse()`

```
tibble::glimpse(si)
```

```
## Observations: 1,163
## Variables: 25
## $ id                   <chr> "amp1001", "amp1002", "amp1003", "amp1004"…
## $ reference_number     <dbl> 1001, 1002, 1002, 1002, 1003, 1004, 1005, …
## $ species_number       <dbl> 19, 16, 14, 13, 30, 42, 23, 19, 13, 1, 1, …
## $ record               <chr> "ab", "co", "co", "co", "co", "co", "co", …
## $ sampled_habitat      <chr> "fo,ll", "fo,la,ll", "fo,la,ll", "fo,la,ll…
## $ active_methods       <chr> "as", "as", "as", "as", "as", NA, "as", "a…
## $ passive_methods      <chr> "pt", "pt", "pt", "pt", NA, NA, NA, NA, "p…
## $ complementary_methods <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…
## $ period               <chr> "mo,da,tw,ni", "mo,da,tw,ni", "mo,da,tw,ni…
## $ month_start          <dbl> 9, 12, 12, 12, 7, NA, 4, 4, 4, 5, 5, 5, 5,…
## $ year_start           <dbl> 2000, 2007, 2007, 2007, 1988, NA, 2007, 20…
## $ month_finish         <dbl> 1, 5, 5, 5, 8, NA, 4, 4, 4, 7, 7, 7, 7, 7,…
```

Dúvidas?


tidyr
www.rstudio.com

4.6 tidyr

Os conjuntos de dados **tidy** (organizados) são 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

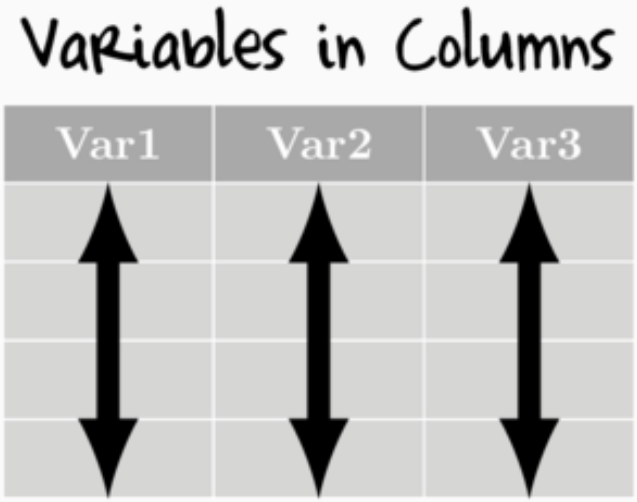
Nos dados tidy:

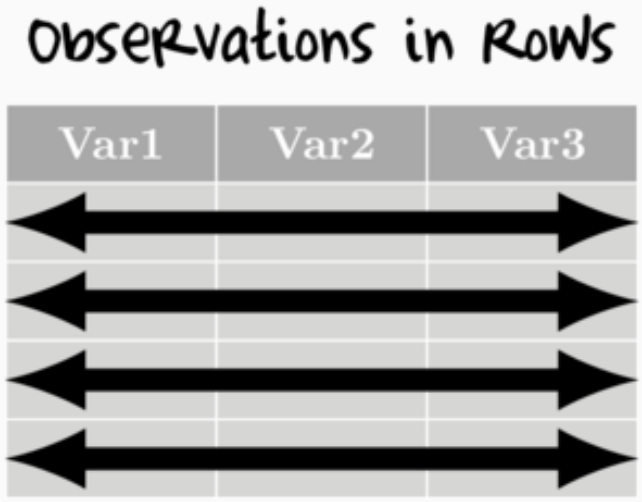
1 Cada variável em uma coluna

2 Cada observação em uma linha

3 Cada valor como uma célula

4.6 tidyr


Variables in Columns
Var1
Var2
Var3


Observations in Rows
Var1
Var2
Var3

4.6 tidyr

Funções

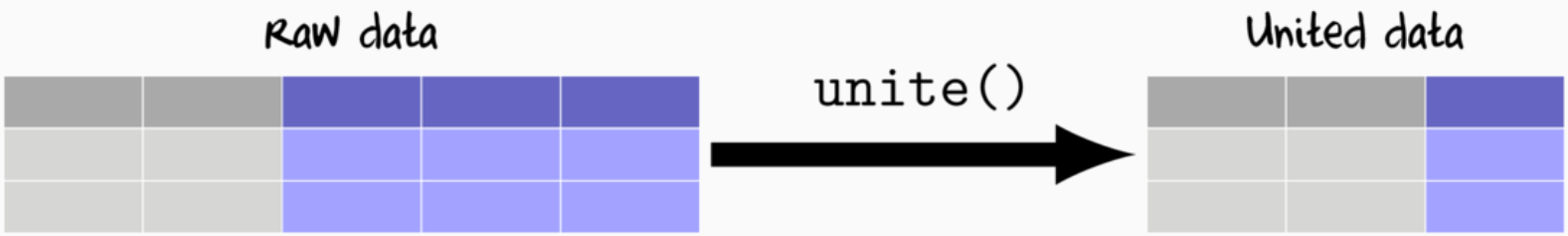
1 unite(): junta dados de múltiplas colunas em uma

2 separate(): separa caracteres em múlplica colunas

3 drop_na(): retira linhas com NA

4 replace_na(): substitui NA

4.6 tidyr



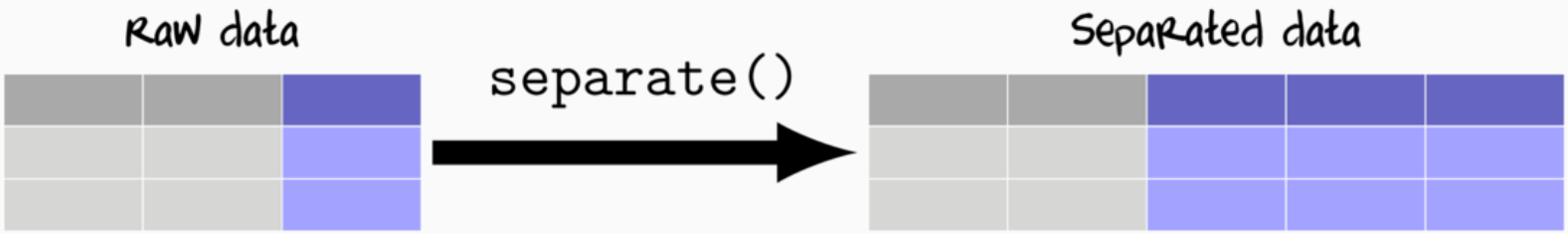
1 unite

unir as colunas latirude e longitude separadas por uma vírgula

```
si_unite <- si %>%
  tidyr::unite("lat_lon", latitude:longitude, sep = ",")
si_unite$lat_lon
```

```
##    [1] "-8.68,-43.42194444"        "-3.545527,-38.857833"
##    [3] "-3.574194,-38.888694"      "-3.51525,-38.918805"
##    [5] "-4.280555556,-38.91083333" "-9.229166667,-36.42805556"
##    [7] "-3.846111111,-40.89444444" "-3.825833333,-40.91944444"
##    [9] "-3.8375,-40.91027778"      "-6.136944444,-35.22944444"
##   [11] "-6.173888889,-35.22027778" "-6.185833333,-35.19944444"
##   [13] "-6.212222222,-35.21027778" "-6.234166667,-35.3125"
##   [15] "-6.261666667,-35.29194444" "-6.248333333,-35.25694444"
##   [17] "-6.256944444,-35.17222222" "-6.271944444,-35.17027778"
##   [19] "-6.253333333,-35.14"       "-6.340833333,-35.21416667"
##   [21] "-6.349722222,-35.21555556" "-6.3425,-35.19194444"
##   [23] "-6.370277778,-35.20388889" "-6.392777778,-35.18388889"
```

4.6 tidyr



2 separate

separar os dados de "period" em quatro colunas dos seus valores

```
si_separate <- si %>%
  tidyr::separate("period", c("mo", "da", "tw", "ni"), remove = FALSE)
si_separate[, c(1, 9:13)]
```

```
## # A tibble: 1,163 x 6
##    id      period      mo    da    tw    ni
##    <chr>   <chr>       <chr> <chr> <chr> <chr>
##  1 amp1001 mo,da,tw,ni mo    da    tw    ni
##  2 amp1002 mo,da,tw,ni mo    da    tw    ni
##  3 amp1003 mo,da,tw,ni mo    da    tw    ni
##  4 amp1004 mo,da,tw,ni mo    da    tw    ni
##  5 amp1005 mo,da,ni    mo    da    ni    <NA>
##  6 amp1006 <NA>        <NA>  <NA>  <NA>  <NA>
##  7 amp1007 <NA>        <NA>  <NA>  <NA>  <NA>
##  8 amp1008 tw,ni       tw    ni    <NA>  <NA>
##  9 amp1009 mo,da,tw,ni mo    da    tw    ni
```

3 drop_na()

remove as linhas com NA de todas as colunas

```
si_drop_na <- si %>%
  tidyr::drop_na()
si_drop_na
```

```
## # A tibble: 40 x 25
##    id    reference_number species_number record sampled_habitat
##    <chr>            <dbl>          <dbl> <chr>  <chr>
##  1 amp1…             1011             14 co     fo,tp,ll,is
##  2 amp1…             1028             29 co     fo
##  3 amp1…             1031             33 co     fo,sw
##  4 amp1…             1077             29 co     fo,pp,la,sw,is
##  5 amp1…             1086              9 co     fo,la,is
##  6 amp1…             1086             18 co     fo,la,is
##  7 amp1…             1086             20 co     fo,la,is
##  8 amp1…             1086             18 co     fo,la,is
##  9 amp1…             1087             49 co     fo,tp,la,sw,is
```

3 drop_na()

remove as linhas com NA da coluna "year_start"

```
si_drop_na <- si %>%
  tidyr::drop_na(year_start)
si_drop_na
```

```
## # A tibble: 1,107 x 25
##    id    reference_number species_number record sampled_habitat
##    <chr>            <dbl>          <dbl> <chr>  <chr>
##  1 amp1…             1001             19 ab     fo,ll
##  2 amp1…             1002             16 co     fo,la,ll
##  3 amp1…             1002             14 co     fo,la,ll
##  4 amp1…             1002             13 co     fo,la,ll
##  5 amp1…             1003             30 co     fo,ll,br
##  6 amp1…             1005             23 co     sp
##  7 amp1…             1005             19 co     sp,la,sw
##  8 amp1…             1005             13 ab     fo
##  9 amp1…             1006              1 ab     fo
```

4 replace_na()

substituir os NAs da coluna "year_start" por 0

```
si_replace_na <- si %>%
  tidyr::replace_na(list(year_start = 0))
si_replace_na
```

```
## # A tibble: 1,163 x 25
##    id    reference_number species_number record sampled_habitat
##    <chr>            <dbl>          <dbl> <chr>  <chr>
##  1 amp1…             1001             19 ab     fo,ll
##  2 amp1…             1002             16 co     fo,la,ll
##  3 amp1…             1002             14 co     fo,la,ll
##  4 amp1…             1002             13 co     fo,la,ll
##  5 amp1…             1003             30 co     fo,ll,br
##  6 amp1…             1004             42 co     tp,pp,la,ll,is
##  7 amp1…             1005             23 co     sp
##  8 amp1…             1005             19 co     sp,la,sw
##  9 amp1…             1005             13 ab     fo
```

Exercícios

Exercício 13

Combine as colunas `country`, `state`, `state_abbreviation`, `municipality`, `site`, em uma coluna chamada `local_total` separadas por `, (vírgula + espaço)`, atribuindo o resultado a um novo objeto, utilizando o formato tidyverse

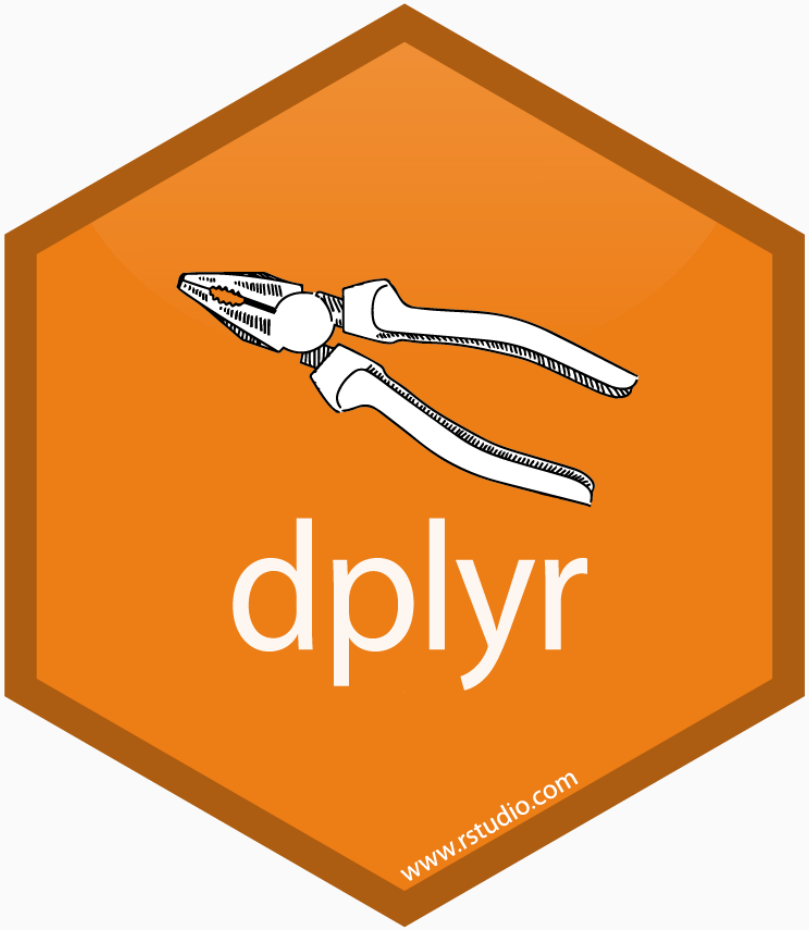
Exercício 13

Solução

```
si_local <- si %>%
  tidyr::unite("local_total", c(country, state, state_abbreviation,
                          municipality, site), sep = ", ")
si_local$local_total
```

```
##    [1] "Brazil, Piauí, BR-PI, Canto do Buriti, Parque Nacional Serra das Confu
##    [2] "Brazil, Ceará, BR-CE, São Gonçalo do Amarante, Dunas"
##    [3] "Brazil, Ceará, BR-CE, São Gonçalo do Amarante, Jardim Botânico Mu
##    [4] "Brazil, Ceará, BR-CE, São Gonçalo do Amarante, Taíba"
##    [5] "Brazil, Ceará, BR-CE, Baturité, Serra de Baturité"
##    [6] "Brazil, Ceará, BR-CE, Quebrangulo, Reserva Biológica de Pedra Talhada'
##    [7] "Brazil, Ceará, BR-CE, Ubajara, Planalto da Ibiapaba"
##    [8] "Brazil, Ceará, BR-CE, Ubajara, Planalto da Ibiapaba"
##    [9] "Brazil, Ceará, BR-CE, Ubajara, Planalto da Ibiapaba"
##   [10] "Brazil, Rio Grande do Norte, BR-RN, São José de Mipibu, Patch A"
##   [11] "Brazil, Rio Grande do Norte, BR-RN, Arês, Patch B"
##   [12] "Brazil, Rio Grande do Norte, BR-RN, Arês, Patch C"
##   [13] "Brazil, Rio Grande do Norte, BR-RN, Arês, Patch D"
```

Dúvidas?


dplyr
www.rstudio.com

4.7 dplyr

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

Ele foi construído com o intuito de obter uma forma **mais rápida** e **expressiva** de tratar os dados

O **tibble** é a **versão de data frame** mais **conveniente** para **se usar** com dplyr

4.7 dplyr

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

Funções

1 select(): seleciona colunas pelo nome gerando um tibble
2 mutate(): adiciona novas colunas ou adiciona resultados em colunas existentes
3 arrange(): reordenar as linhas com base nos valores de colunas
4 filter(): seleciona linhas com base em valores
5 summarise(): agrega ou resume os dados através de funções, podendo considerar valores das colunas

4.7 dplyr

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

Todas seguem a mesma sintaxe:

1. tibble

2. operador pipe

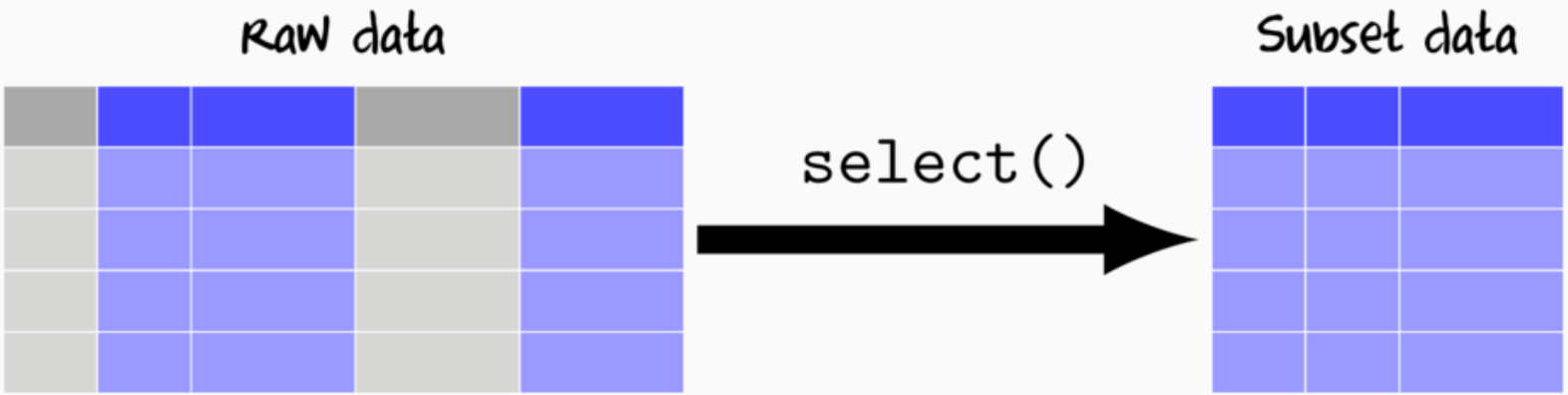
3. nome da função verbal com os argumentos entre parênteses

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

```
sp_dplyr <- sp %>%
  funcao_verbal(argumento1, argumento2)
```

4.7 dplyr

1 select



1 select

Seleciona colunas pelo nome

```
si_select <- si %>%
  dplyr::select(id, longitude, latitude)
si_select
```

```
## # A tibble: 1,163 x 3
##    id      longitude latitude
##    <chr>       <dbl>    <dbl>
##  1 amp1001     -43.4    -8.68
##  2 amp1002     -38.9    -3.55
##  3 amp1003     -38.9    -3.57
##  4 amp1004     -38.9    -3.52
##  5 amp1005     -38.9    -4.28
##  6 amp1006     -36.4    -9.23
##  7 amp1007     -40.9    -3.85
##  8 amp1008     -40.9    -3.83
##  9 amp1009     -40.9    -3.84
```

1 select

Remove as colunas pelo nome

```
si_select <- si %>%
  dplyr::select(-c(id, longitude, latitude))
si_select
```

```
## # A tibble: 1,163 x 22
##    reference_number species_number record sampled_habitat active_methods
##               <dbl>          <dbl> <chr>  <chr>           <chr>
##  1             1001             19 ab     fo,ll           as
##  2             1002             16 co     fo,la,ll        as
##  3             1002             14 co     fo,la,ll        as
##  4             1002             13 co     fo,la,ll        as
##  5             1003             30 co     fo,ll,br        as
##  6             1004             42 co     tp,pp,la,ll,is  <NA>
##  7             1005             23 co     sp              as
##  8             1005             19 co     sp,la,sw        as,sb,tr
##  9             1005             13 ab     fo              <NA>
```

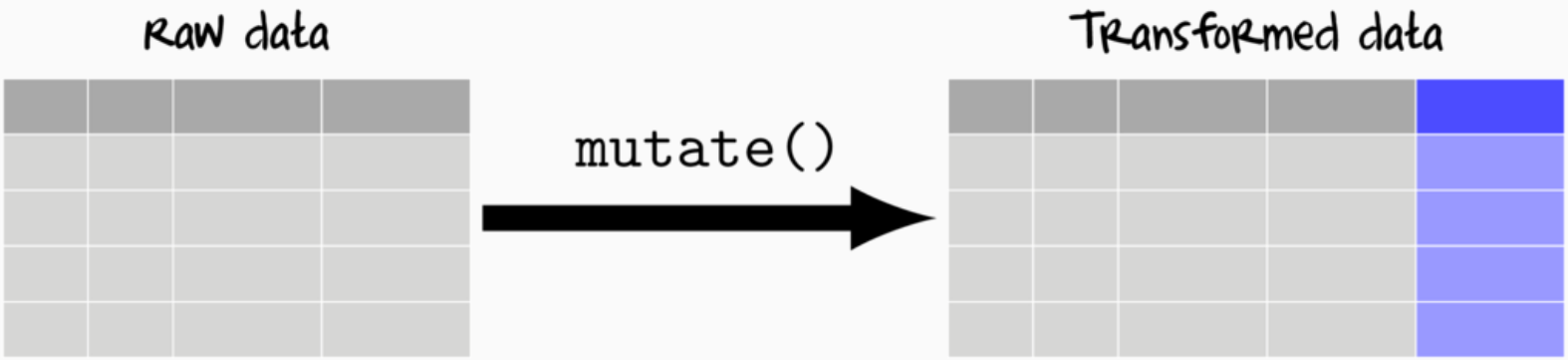
1 select

Seleciona colunas por um padrão

```
#  starts_with(), ends_with() e contains()
si_select <- si %>%
  dplyr::select(contains("sp"))
si_select
```

```
## # A tibble: 1,163 x 1
##    species_number
##             <dbl>
##  1             19
##  2             16
##  3             14
##  4             13
##  5             30
##  6             42
##  7             23
##  8             19
```

2 mutate



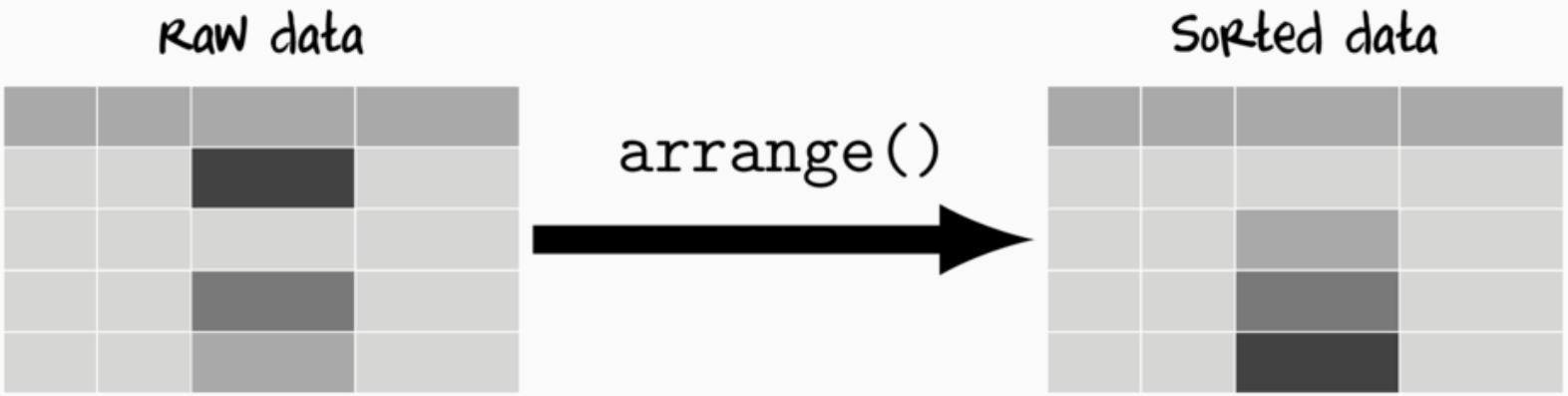
2 mutate

Adiciona colunas novas ou vindas de operações

```
# add colum record as factor
si_mutate <- si %>%
  dplyr::mutate(record_factor = as.factor(record))
si_mutate
```

```
## # A tibble: 1,163 x 26
##    id    reference_number species_number record sampled_habitat
##    <chr>            <dbl>          <dbl> <chr>  <chr>
##  1 amp1…             1001             19 ab     fo,ll
##  2 amp1…             1002             16 co     fo,la,ll
##  3 amp1…             1002             14 co     fo,la,ll
##  4 amp1…             1002             13 co     fo,la,ll
##  5 amp1…             1003             30 co     fo,ll,br
##  6 amp1…             1004             42 co     tp,pp,la,ll,is
##  7 amp1…             1005             23 co     sp
##  8 amp1…             1005             19 co     sp,la,sw
```

3 arrange



3 arrange

Ordenar de forma crescente pela coluna **species_number**

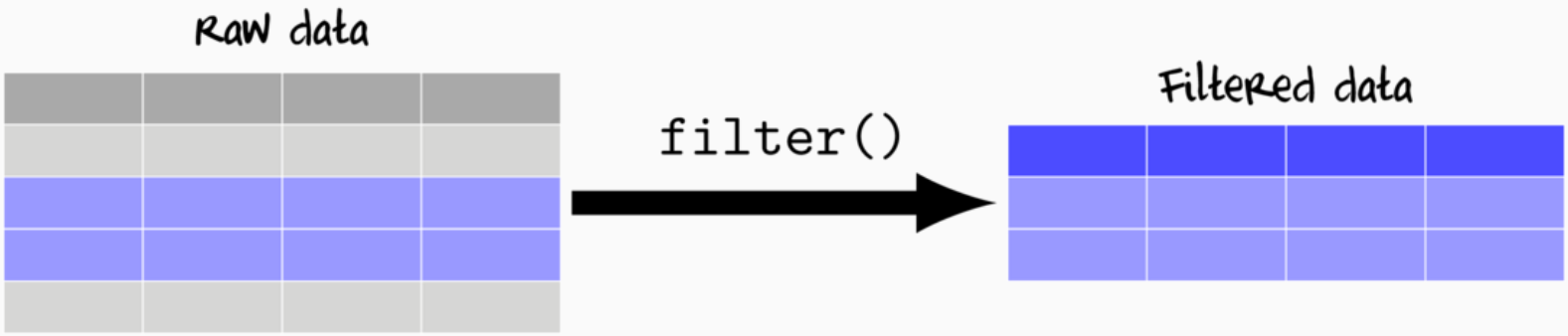
```
si_arrange <- si %>%
  dplyr::arrange(species_number)
si_arrange
```

Ordenar de forma decrescente pela coluna **species_number**

```
si_arrange <- si %>%
  dplyr::arrange(desc(species_number))
si_arrange
```

```
si_arrange <- si %>%
  dplyr::arrange(-species_number)
si_arrange
```

4 filter



4 filter

operadores: >, >=, <, <=, ==, !=, is.na, !is.na, %in%

Filtrar para locais com mais de 5 espécies

```
si_filter <- si %>%
  dplyr::filter(species_number > 5)
si_filter
```

Filtrar para locais centre 1 e 5 espécies

```
si_filter <- si %>%
  dplyr::filter(between(species_number, 1, 5))
si_filter
```

4 filter

Filtrar para locais sem NA no métodos passivos

```
si_filter <- si %>%
  dplyr::filter(is.na(passive_methods))
si_filter
```

Filtrar para locais sem NA no métodos ativos **E** passivos

```
si_filter <- si %>%
  dplyr::filter(is.na(active_methods) & is.na(passive_methods))
si_filter
```

4 filter

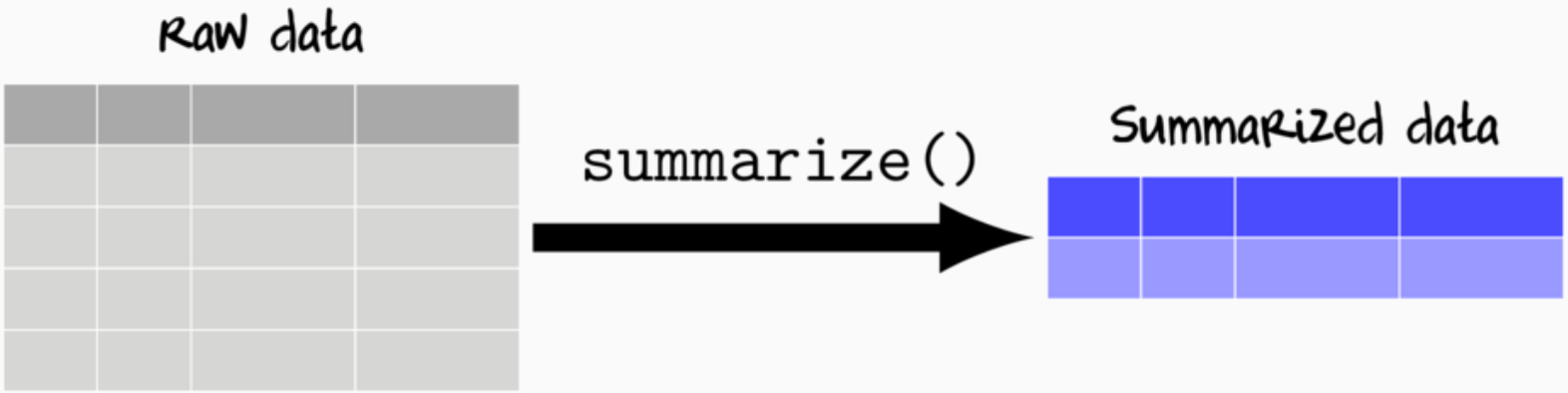
Filtrar para locais amostrados com mais de 5 espécies **E** em Pernambuco

```
si_filter <- si %>%
  dplyr::filter(species_number > 5 & state_abbreviation == "BR-PE")
si_filter
```

Filtrar para locais amostrados com mais de 5 espécies **OU** em Pernambuco

```
si_filter <- si %>%
  dplyr::filter(species_number > 5 | state_abbreviation == "BR-PE")
si_filter
```

5 summarise



5 summarise

Média e desvio padrão do número de espécies total

```
si_summarise <- si %>%
  dplyr::summarise(mean_sp = mean(species_number), sd_sp = sd(species_number
si_summarise
```

```
## # A tibble: 1 x 2
##   mean_sp sd_sp
##     <dbl> <dbl>
## 1    15.2  11.2
```

5 summarise

Média e desvio padrão do número de espécies por país

```
si_summarise_group <- si %>%
  dplyr::group_by(country) %>%
  dplyr::summarise(mean_sp = mean(species_number), sd_sp = sd(species_number
si_summarise_group
```

```
## # A tibble: 3 x 3
##   country   mean_sp sd_sp
##   <chr>       <dbl> <dbl>
## 1 Argentina    8.36  7.73
## 2 Brazil      15.2  11.3
## 3 Paraguay    28     2.92
```

4.7 dplyr

Permite manipular os dados de forma mais simples

```
da <- si %>%
  dplyr::select(id, state_abbreviation, species_number)
da
```

```
## # A tibble: 1,163 x 3
##    id      state_abbreviation species_number
##    <chr>   <chr>                       <dbl>
##  1 amp1001 BR-PI                          19
##  2 amp1002 BR-CE                          16
##  3 amp1003 BR-CE                          14
##  4 amp1004 BR-CE                          13
##  5 amp1005 BR-CE                          30
##  6 amp1006 BR-CE                          42
##  7 amp1007 BR-CE                          23
##  8 amp1008 BR-CE                          19
##  9 amp1009 BR-CE                          13
## 10 amp1010 BR-RN                           1
## # … with 1,153 more rows
```

4.7 dplyr

Permite manipular os dados de forma mais simples

```
da <- si %>%
  dplyr::select(id, state_abbreviation, species_number) %>%
  dplyr::filter(species_number > 5)
da
```

```
## # A tibble: 934 x 3
##    id      state_abbreviation species_number
##    <chr>   <chr>                       <dbl>
##  1 amp1001 BR-PI                          19
##  2 amp1002 BR-CE                          16
##  3 amp1003 BR-CE                          14
##  4 amp1004 BR-CE                          13
##  5 amp1005 BR-CE                          30
##  6 amp1006 BR-CE                          42
##  7 amp1007 BR-CE                          23
##  8 amp1008 BR-CE                          19
##  9 amp1009 BR-CE                          13
## 10 amp1015 BR-RN                           6
```

4.7 dplyr

Permite manipular os dados de forma mais simples

```
da <- si %>%
  dplyr::select(id, state_abbreviation, species_number) %>%
  dplyr::filter(species_number > 5) %>%
  dplyr::group_by(state_abbreviation) %>%
  dplyr::summarise(nsp_state_abb = n())
da
```

```
## # A tibble: 23 x 2
##    state_abbreviation nsp_state_abb
##    <chr>                      <int>
##  1 AR-N                           7
##  2 BR-AL                          3
##  3 BR-BA                         63
##  4 BR-CE                          8
##  5 BR-ES                         33
##  6 BR-GO                          4
##  7 BR-MG                         91
##  8 BR-MS                          1
```

4.7 dplyr

Permite manipular os dados de forma mais simples

```
da <- si %>%
  dplyr::select(id, state_abbreviation, species_number) %>%
  dplyr::filter(species_number > 5) %>%
  dplyr::group_by(state_abbreviation) %>%
  dplyr::summarise(nsp_state_abb = n()) %>%
  dplyr::arrange(nsp_state_abb)
da
```

```
## # A tibble: 23 x 2
##    state_abbreviation nsp_state_abb
##    <chr>                      <int>
##  1 BR-MS                          1
##  2 BR-PI                          1
##  3 PY-13                          1
##  4 PY-14                          1
##  5 PY-2                           1
##  6 PY-4                           1
##  7 PY-7                           1
```

Exercícios

Exercício 14

Adicione essas novas colunas `alt_log, tem_log e pre_log`, que são a operação `log10` das colunas `altitude, temperature e precipitation` e atribua ao mesmo objeto `si` utilizando o formato tidyverse

Exercício 14

Solução

```
# solucao
si <- si %>%
  dplyr::mutate(alt_log = log10(altitude),
                tem_log = log10(temperature),
                pre_log = log10(precipitation))
si[, 25:28]
```

```
## # A tibble: 1,163 x 4
##    precipitation alt_log tem_log pre_log
##            <dbl>   <dbl>   <dbl>   <dbl>
##  1           853    2.73    1.40    2.93
##  2          1318    1.18    1.42    3.12
##  3          1248    1.46    1.42    3.10
##  4          1376    1.40    1.42    3.14
##  5          1689    2.88    1.33    3.23
##  6          1249    2.87    1.31    3.10
##  7          1520    2.94    1.33    3.18
##  8          1474    2.94    1.33    3.17
```

Exercício 15

Ordene os dados em forma decrescente pela coluna `altitude`, atribuindo o resultado a um novo objeto utilizando o formato tidyverse

Exercício 15

Solução

```
# solucao
si_ar <- si %>%
  dplyr::arrange(-altitude)
si_ar
```

```
## # A tibble: 1,163 x 28
##    id    reference_number species_number record sampled_habitat
##    <chr>            <dbl>          <dbl> <chr>  <chr>
##  1 amp1…             1102             25 co     fo,sw,ll,is,br
##  2 amp1…             1142             14 co     fo,ll,is,br
##  3 amp1…             1090             18 co     la
##  4 amp1…             1120             31 co     fo,tp,pp,sw,is…
##  5 amp1…             1273             19 co     tp,pp,sw,ll,is…
##  6 amp1…             1068             15 co     tp
##  7 amp1…             1059             43 co     sp,sw,is
##  8 amp1…             1219             46 co     fo,pp,sw,ll,is…
##  9 amp1…             1109             40 co     fo,pp,la,is,br
## 10 amp1…             1283             15 co     fo,tp,pp,ll,is
```

Exercício 16

Filtre as linhas com `altitude` maior que 1000 mm, `temperature` menor que 15 ºC e `precipitation` maior que 1000 mm, atribuindo o resultado a um novo objeto utilizando o formato tidyverse

Exercício 16

Solução

```
# solucao
si_fi <- si %>%
  dplyr::filter(altitude > 1000,
                temperature < 15,
                precipitation > 1000)
si_fi
```

```
## # A tibble: 25 x 28
##    id    reference_number species_number record sampled_habitat
##    <chr>            <dbl>          <dbl> <chr>  <chr>
##  1 amp1…             1090             18 co     la
##  2 amp1…             1120             31 co     fo,tp,pp,sw,is…
##  3 amp1…             1102             25 co     fo,sw,ll,is,br
##  4 amp1…             1142             14 co     fo,ll,is,br
##  5 amp1…             1167              9 ab     <NA>
##  6 amp1…             1219             46 co     fo,pp,sw,ll,is…
##  7 amp1…             1224             17 ab     tp,la,is
##  8 amp1…             1273             19 co     tp,pp,sw,ll,is…
```

Dúvidas?

Maurício Vancine

Contatos:

- mauricio.vancine@gmail.com
- mauriciovancine.netlify.com
- @mauriciovancine
- @mauriciovancine
- @mauriciovancine

Slides criados via pacote xaringan e tema Metropolis