manipulação de dados

Sara Mortara, Andrea Sánchez-Tapia, Diogo Rocha

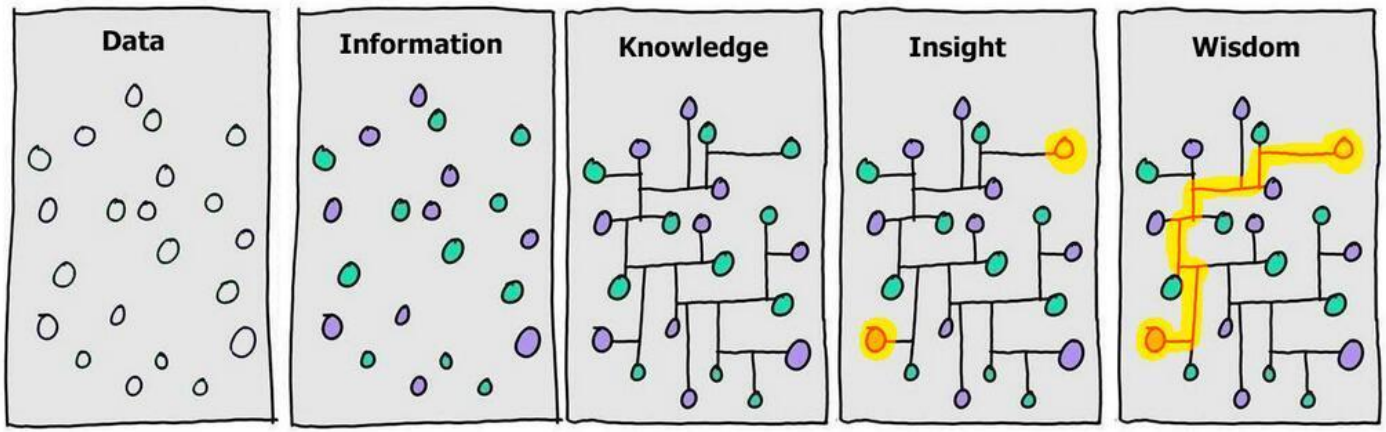
11 fev 2020

sobre a aula

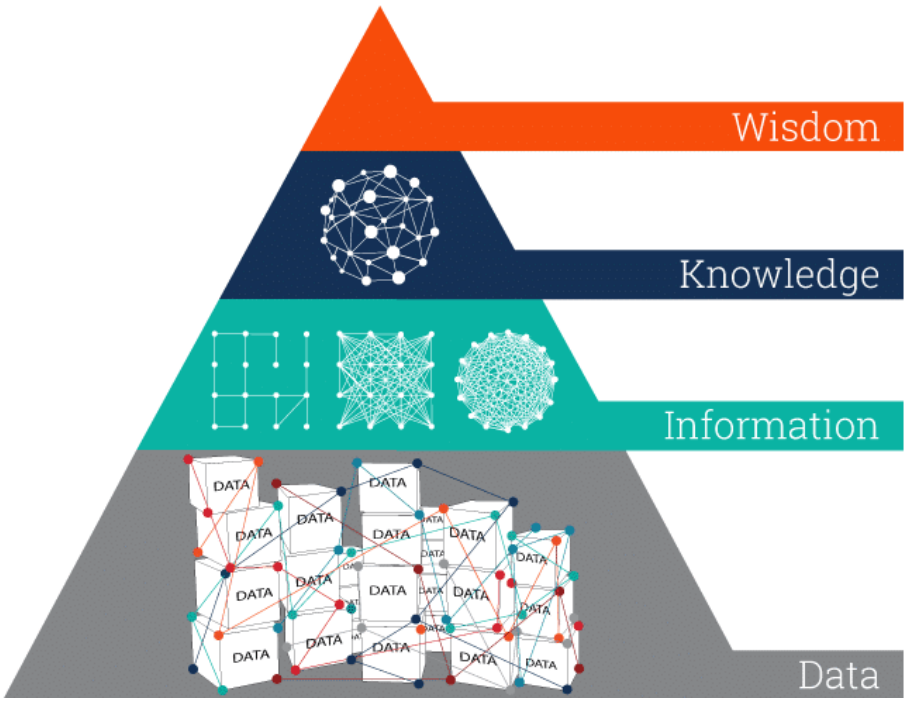
1. dados de biodiversidade
2. bases de dados relacionais
3. manipulação de dados em R

1. dados de biodiversidade

dados & conhecimento



dados & conhecimento



Each step up
the pyramid
answers
questions
about and
adds value
to the initial data.

dados de biodiversidade

museus & herbários

dados de biodiversidade

amostragem

componentes dos dados de biodiversidade

- espécies (informação taxonômica)

componentes dos dados de biodiversidade

- espécies (informação taxonômica)
- atributos das espécies

componentes dos dados de biodiversidade

- espécies (informação taxonômica)
- atributos das espécies
- localidades

componentes dos dados de biodiversidade

- espécies (informação taxonômica)
- atributos das espécies
- localidades
- ocorrências

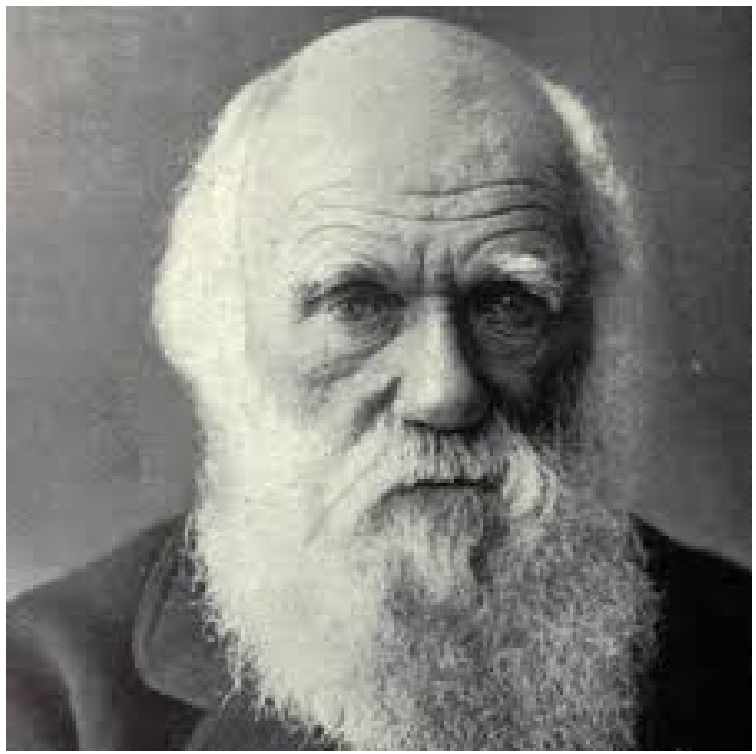
componentes dos dados de biodiversidade

- espécies (informação taxonômica)
- atributos das espécies
- localidades
- ocorrências
- variáveis: altura, peso…

padronização de dados em biodiversidade

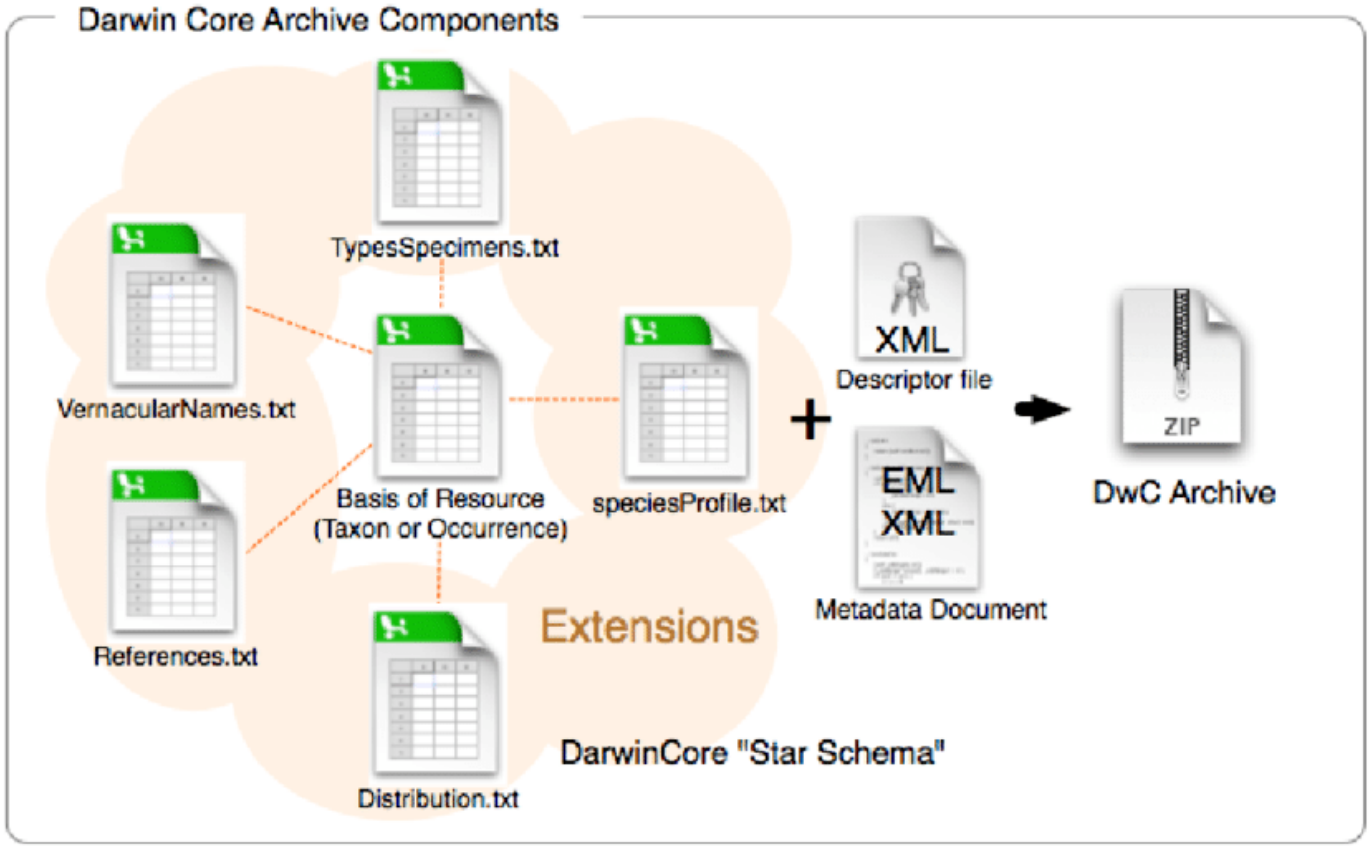
Darwin Core

facilitar o compartilhamento da informação sobre diversidade biológica



padronização de dados em biodiversidade

Darwin Core



2. bases de dados relacionais

como funcionam as bases de dados relacionais

- diferentes dados são organizados em diferentes tabelas

como funcionam as bases de dados relacionais

- diferentes dados são organizados em diferentes tabelas
- tabelas são integradas

como funcionam as bases de dados relacionais

- diferentes dados são organizados em diferentes tabelas
- tabelas são integradas
- identificador comum para cada tabela

como funcionam as bases de dados relacionais

- diferentes dados são organizados em diferentes tabelas
- tabelas são integradas
- identificador comum para cada tabela
- em geral organizadas em SQL (*Structured Query Language*)

como lidamos com dados relacionais

- tipicamente análises de dados em diferentes tabelas

como lidamos com dados relacionais

- tipicamente análises de dados em diferentes tabelas
- relações entre as tabelas são feitas aos pares

como lidamos com dados relacionais

- tipicamente análises de dados em diferentes tabelas
- relações entre as tabelas são feitas aos pares
- relações de três ou mais tabelas são sempre propriedades das relações de cada par

como lidamos com dados relacionais

- tipicamente análises de dados em diferentes tabelas
- relações entre as tabelas são feitas aos pares
- relações de três ou mais tabelas são sempre propriedades das relações de cada par
- usamos verbos para trabalhar com pares de tabelas

tipos de verbos

- mutating

tipos de verbos

- mutating
- filtering

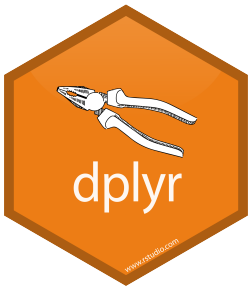
tipos de verbos

- mutating
- filtering
- set operations

tipos de verbos

- mutating
- filtering
- set operations
- no R manipulação de bases pode ser feita dentro dos pacotes **base** e **dplyr**

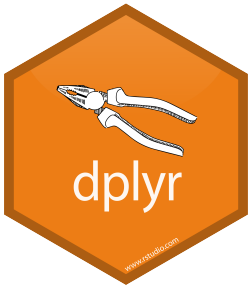
pacote **dplyr**



construído a partir da lógica de bases relacionais

- mais fácil que SQL porque é voltado para análise de dados

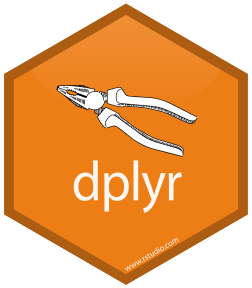
pacote **dplyr**



construído a partir da lógica de bases relacionais

- mais fácil que SQL porque é voltado para análise de dados
- gramática para manipulação de dados

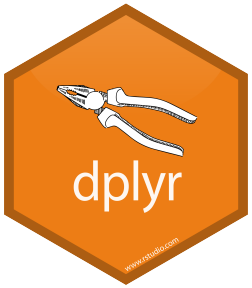
pacote **dplyr**



construído a partir da lógica de bases relacionais

- mais fácil que SQL porque é voltado para análise de dados
- gramática para manipulação de dados
- `mutate()`

pacote **dplyr**

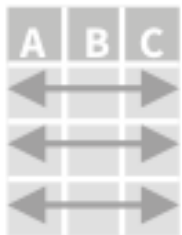


construído a partir da lógica de bases relacionais

- mais fácil que SQL porque é voltado para análise de dados
- gramática para manipulação de dados
- `mutate()`
- `filter()`

estrutura de dados

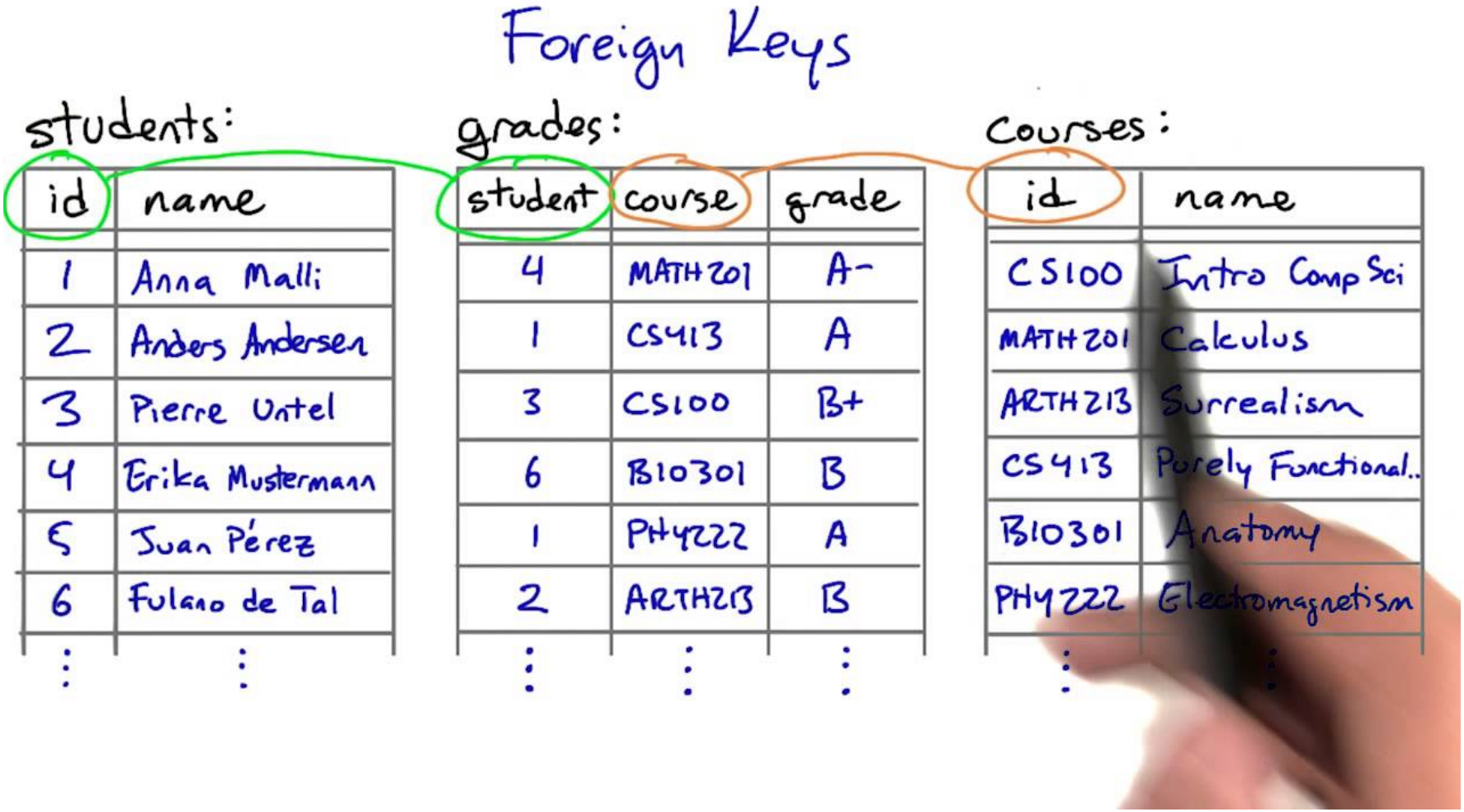
&



Each **variable** is in its own **column**

Each **observation**, or **case**, is in its own **row**

chaves


Foreign Keys
students:
id
name
1
Anna Malli
2
Anders Andersen
3
Pierre Untel
4
Erika Mustermann
5
Juan Pérez
6
Fulano de Tal
grades:
student
course
grade
4
MATH201
A-
1
CS413
A
3
CS100
B+
6
BIO301
B
1
PHY222
A
2
ARTH213
B
Courses:
id
name
CS100
Intro Comp Sci
MATH201
Calculus
ARTH213
Surrealism
CS413
Purely Functional..
BIO301
Anatomy
PHY222
Electromagnetism

chaves

- chave primária $\longrightarrow$ identifica uma observação em sua própria tabela

chaves

- chave primária $\longrightarrow$ identifica uma observação em sua própria tabela
- chave estrangeira $\longrightarrow$ identifica uma observação em uma outra tabela

chaves

- chave primária $\longrightarrow$ identifica uma observação em sua própria tabela
- chave estrangeira $\longrightarrow$ identifica uma observação em uma outra tabela
- uma variável pode ser uma chave primária em uma tabela e uma chave estrangeira em outra

chaves

- chave primária $\longrightarrow$ identifica uma observação em sua própria tabela
- chave estrangeira $\longrightarrow$ identifica uma observação em uma outra tabela
- uma variável pode ser uma chave primária em uma tabela e uma chave estrangeira em outra
- toda chave primária deve conter uma informação única!

chaves

- chave primária $\longrightarrow$ identifica uma observação em sua própria tabela
- chave estrangeira $\longrightarrow$ identifica uma observação em uma outra tabela
- uma variável pode ser uma chave primária em uma tabela e uma chave estrangeira em outra
- toda chave primária deve conter uma informação única!
- toda tabela deve ter uma chave primária

relação

é dada pela chave primária e sua respectiva chave estrangeira

- 1-para-muitas

relação

é dada pela chave primária e sua respectiva chave estrangeira

- 1-para-muitas
- 1-para-1

relação

é dada pela chave primária e sua respectiva chave estrangeira

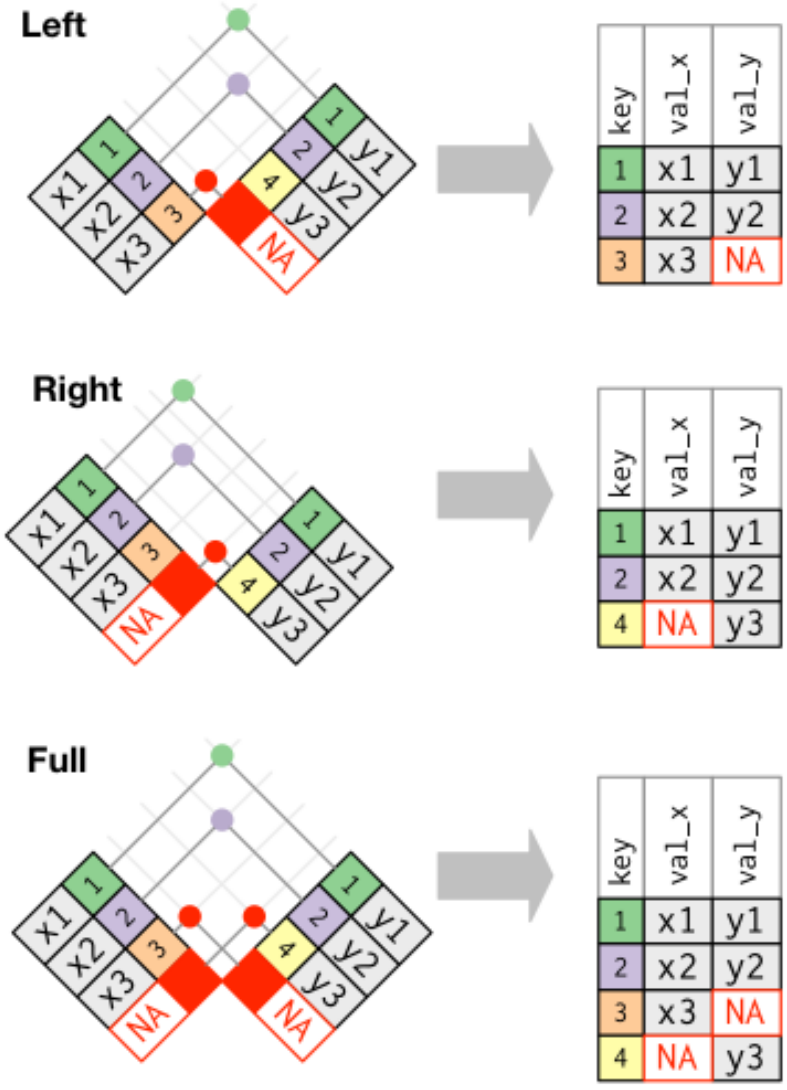
- 1-para-muitas
- 1-para-1
- usamos as chaves para combinar tabelas

relação

é dada pela chave primária e sua respectiva chave estrangeira

- 1-para-muitas
- 1-para-1
- usamos as chaves para combinar tabelas
- usamos `merge()` (**base**) ou `join()` (**dplyr**)

usando relações para juntar dados

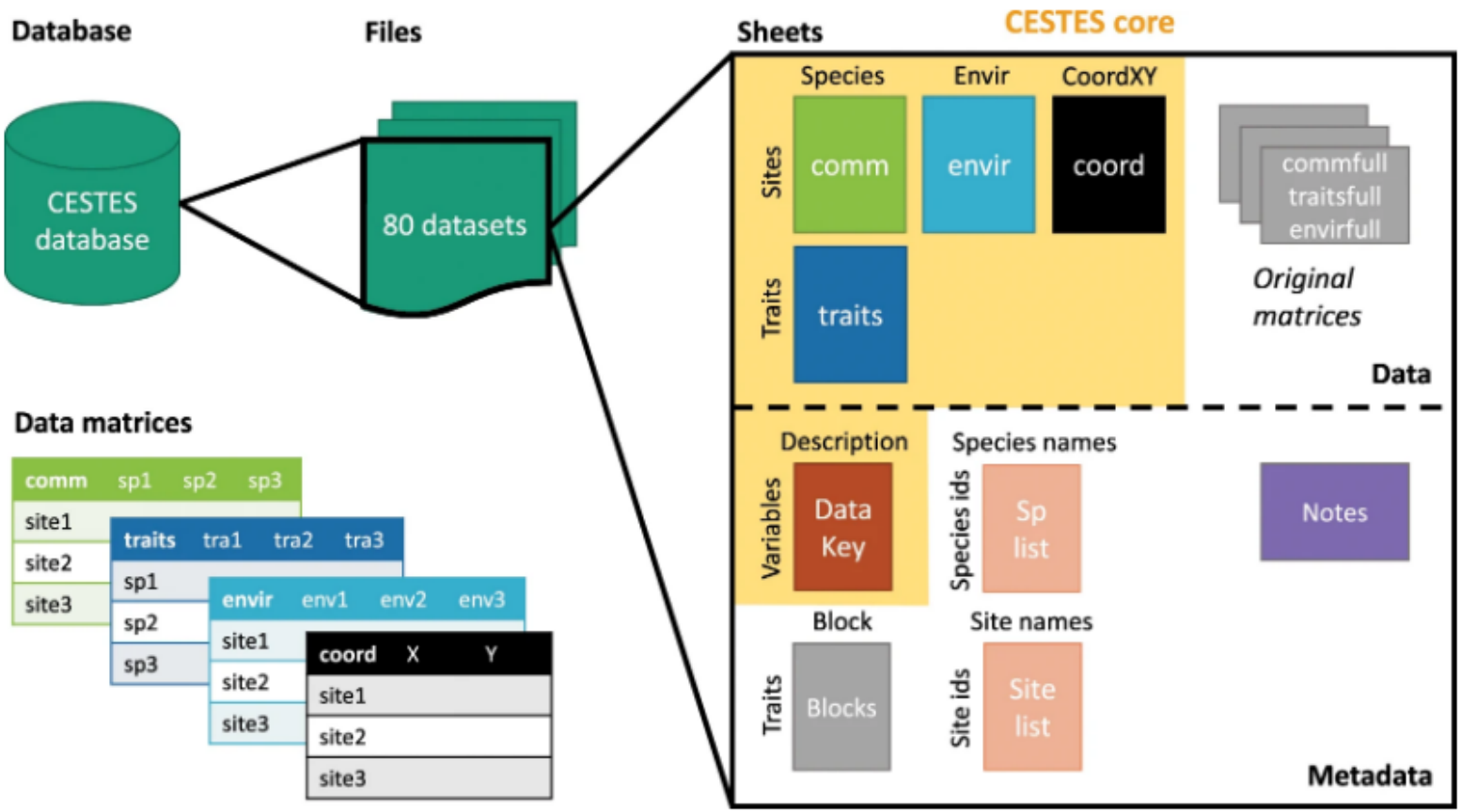


equivalências entre **dplyr** e **base**

dplyr	merge
`inner_join(x, y)`	`merge(x, y)`
`left_join(x, y)`	`merge(x, y, all.x = TRUE)`
`right_join(x, y)`	`merge(x, y, all.y = TRUE)` ,
`full_join(x, y)`	`merge(x, y, all.x = TRUE, all.y = TRUE)`

3. manipulação de dados em R

base de dados CESTES



Jeliazkov et al 2020 Sci Data

dados e código aberto

```
#########################################################################
####
############### DATA PREPARATION FOR THE META-STUDY ###############
####
############### Date: 09/03/2018 | Revision: 26/07/2019
############### Author: Alienor Jeliazkov
#########################################################################
####
## Associated publication:
## Jeliazkov A., [78 co-authors] & J. Chase. A global database
for metacommunity ecology:
## species, traits, environment and space. Nature's Scientific
Data.
```

dados e código aberto

```
############ Load packages #####
library(doBy)
library(readxl)
library(plyr)
library(dplyr)
library(gdata)
library(cellranger)


############ Set working directory and load packages #####
setwd("Path_to_CESTES_data_dir") # directory where ONLY the DATA
Excel files are stored
setwd("C:/AJ/POSTDOC_iDIV/PROJECTS/DB_CESTES_REV/xCESTES")
datfiles <- list.files() # list the files in the directory
```

dados e código aberto

https://www.tidyverse.org/blog/2017/12/workflow-vs-script/

> *If the first line of your R script is*
>
> `setwd("C:\Users\jenny\path\that\only\I\have")`
>
> *I will come into your office and SET YOUR COMPUTER ON FIRE 🔥.*

> *If the first line of your R script is*
>
> `rm(list = ls())`
>
> *I will come into your office and SET YOUR COMPUTER ON FIRE 🔥.*

um fluxo de trabalho no R

começamos carregando os pacotes necessários

```
library("reshape2")
library("tidyverse")
```

explorando os dados

species vs. sites

```
files_path <- list.files("data",
                         pattern = ".csv",
                         full.names = TRUE)

files_path
```

```
## [1] "data/comm.csv"   "data/coord.csv"  "data/envir.csv"  "da
## [5] "data/traits.csv"
```

```
files_path[3]
```

```
## [1] "data/envir.csv"
```

lendo o dado no R

criando objetos 'data.frame'

```
comm <- read.csv(files_path[1])
coord <- read.csv(files_path[2])
envir <- read.csv(files_path[3])
splist <- read.csv(files_path[4])
traits <- read.csv(files_path[5])
```

outra opção de leitura do dado

criando objetos 'list'

```
data <- sapply(files_path, read.csv)

length(data)
```

```
## [1] 5
```

```
head(data[[4]])
```

```
##             TaxonName Taxon TaxCode
## 1   Arisarum vulgare  Arvu     sp1
## 2    Alisma plantago  Alpl     sp2
## 3  Damasonium alisma  Daal     sp3
## 4 Asphodelus aetivus  Asae     sp4
## 5  Narcissus tazetta  Nata     sp5
## 6  Narcissus elegans  Nael     sp6
```

ainda outra opção usando um ciclo

criando objetos 'list'

```
length(files_path)

data <- list()

for (i in 1:length(files_path)) {
  data[[i]] <- read.csv(files_path[i])
}
```

inspecionando os dados da comunidade

```
comm[1:6, 1:10]
```

```
##   Sites sp1 sp2 sp3 sp4 sp5 sp6 sp7 sp8 sp9
## 1     1   0   0   0   0   0   0   0   1   0
## 2     2   0   1   0   0   0   0   0   0   0
## 3     3   0   1   0   0   0   0   0   0   2
## 4     4   0   0   0   0   0   0   0   0   0
## 5     5   0   0   0   0   0   0   0   0   0
## 6     6   0   0   0   0   0   0   0   0   0
```

inspecionando os dados de atributos

```
traits[1:9, 1:7]
```

```
##    Sp Anemo Auto Entomo Annual Biennial Perennial
## 1 sp1     0    0      1      0         0         1
## 2 sp2     0    0      1      0         0         1
## 3 sp3     0    0      1      1         1         1
## 4 sp4     0    0      1      0         0         1
## 5 sp5     0    0      1      0         0         1
## 6 sp6     0    0      1      0         0         1
## 7 sp7     0    0      1      0         0         1
## 8 sp8     0    0      1      0         0         1
## 9 sp9     1    0      0      0         0         1
```

inspecionando os dados ambientais

```
head(envir)
```

```
##   Sites Clay Silt Sand K2O   Mg Na100g   K Elev
## 1     1 0.73 0.24 0.03 1.3  9.2    4.2 1.2    6
## 2     2 0.75 0.24 0.02 0.8 10.7   10.4 1.4    2
## 3     3 0.74 0.24 0.02 1.7  8.6   10.8 1.9    2
## 4     4 0.23 0.26 0.49 0.3  2.0    1.2 0.3    6
## 5     5 0.73 0.24 0.03 1.3  9.2    4.2 1.2    6
## 6     6 0.72 0.22 0.03 1.7  6.0   10.7 1.3    4
```

inspecionando os dados de coordenadas

```
head(coord)
```

```
##   Sites      X      Y
## 1     1 365.80 181.20
## 2     2 349.60 185.00
## 3     3 333.40 185.40
## 4     4 373.86 161.43
## 5     5 380.25 179.50
## 6     6 354.40 168.60
```

inspecionando os dados da lista de espécies

```
head(splist)
```

```
##              TaxonName Taxon TaxCode
## 1   Arisarum vulgare  Arvu     sp1
## 2    Alisma plantago  Alpl     sp2
## 3  Damasonium alisma  Daal     sp3
## 4 Asphodelus aetivus  Asae     sp4
## 5  Narcissus tazetta  Nata     sp5
## 6  Narcissus elegans  Nael     sp6
```

```
# quantas especies?
nrow(splist)
```

```
## [1] 56
```

adicionando as coordenadas na planilha dos sites

```
# info sobre coord
names(coord)
```

```
## [1] "Sites" "X"     "Y"
```

```
dim(coord)
```

```
## [1] 97  3
```

```
# info sobre envir
names(envir)
```

```
## [1] "Sites"  "Clay"   "Silt"   "Sand"   "K2O"    "Mg"     "Na
## [9] "Elev"
```

```
dim(envir)
```

```
## [1] 97  9
```

usando `merge` e uma coluna comum

coluna comum é chave primária e estrangeira

```
envir.coord <- merge(x = envir,
                     y = coord,
                     by = "Sites")

dim(envir.coord)
```

```
## [1] 97 11
```

checando o merge

```
names(envir.coord)
```

```
##  [1] "Sites"  "Clay"   "Silt"   "Sand"   "K2O"    "Mg"     "N
##  [8] "K"      "Elev"   "X"      "Y"
```

transformando a matriz espécie vs. area em tabela de dados

```
comm[1:6, 1:10]
```

```
##   Sites sp1 sp2 sp3 sp4 sp5 sp6 sp7 sp8 sp9
## 1     1   0   0   0   0   0   0   0   1   0
## 2     2   0   1   0   0   0   0   0   0   0
## 3     3   0   1   0   0   0   0   0   0   2
## 4     4   0   0   0   0   0   0   0   0   0
## 5     5   0   0   0   0   0   0   0   0   0
## 6     6   0   0   0   0   0   0   0   0   0
```

usando o pacote **reshape**

```
# transformando matriz em tabela de dado
comm.df <- reshape2::melt(comm[, -1])
```

```
## No id variables; using all as measure variables
```

```
# checando se funcionou
head(comm.df)
```

```
##   variable value
## 1      sp1     0
## 2      sp1     0
## 3      sp1     0
## 4      sp1     0
## 5      sp1     0
## 6      sp1     0
```

usando o pacote **reshape**

```
# quantas vezes a especie se repete
table(comm.df$variable)
```

```
##
##  sp1  sp2  sp3  sp4  sp5  sp6  sp7  sp8  sp9 sp10 sp11 sp12 s
##   97   97   97   97   97   97   97   97   97   97   97   97
## sp16 sp17 sp18 sp19 sp20 sp21 sp22 sp23 sp24 sp25 sp26 sp27 s
##   97   97   97   97   97   97   97   97   97   97   97   97
## sp31 sp32 sp33 sp34 sp35 sp36 sp37 sp38 sp39 sp40 sp41 sp42 s
##   97   97   97   97   97   97   97   97   97   97   97   97
## sp46 sp47 sp48 sp49 sp50 sp51 sp52 sp53 sp54 sp55 sp56
##   97   97   97   97   97   97   97   97   97   97   97
```

criando a variável “Sites”

```
# quantas especies?
n.sp <- nrow(splist)
n.sp
```

```
## [1] 56
```

```
# vetor contendo todos os Sites
Sites <- envir$Sites
length(Sites)
```

```
## [1] 97
```

```
comm.df$Sites <- rep(Sites, each = n.sp)
```

checando se a primeira coluna foi criada

```
head(comm.df)
```

```
##   variable value Sites
## 1      sp1     0     1
## 2      sp1     0     1
## 3      sp1     0     1
## 4      sp1     0     1
## 5      sp1     0     1
## 6      sp1     0     1
```

alterando o nome das colunas

```
names(comm.df)
```

```
## [1] "variable" "value"    "Sites"
```

```
names(comm.df)[1:2] <- c("TaxCode", "Abundance")

head(comm.df)
```

```
##   TaxCode Abundance Sites
## 1     sp1         0     1
## 2     sp1         0     1
## 3     sp1         0     1
## 4     sp1         0     1
## 5     sp1         0     1
## 6     sp1         0     1
```

usando os pacote **dplyr** e **tidyr**

vamos adicionar uma coluna com o nome das espécies ao nosso objeto `comm.df`

```
comm.sp <- merge(x = comm.df,
                 y = splist[, c(1, 3)],
                 by = "TaxCode")
```

usando os pacote **dplyr** e **tidyr**

vamos adicionar uma coluna com o nome das espécies ao nosso objeto `comm.df`

```
comm.sp <- inner_join(x = comm.df,
                      y = splist[, c(1, 3)])
```

```
## Joining, by = "TaxCode"
```

```
## Warning: Column `TaxCode` joining factors with different leve
## to character vector
```

checando a nossa nova coluna

```
head(comm.sp)
```

```
##   TaxCode Abundance Sites         TaxonName
## 1     sp1         0     1 Arisarum vulgare
## 2     sp1         0     1 Arisarum vulgare
## 3     sp1         0     1 Arisarum vulgare
## 4     sp1         0     1 Arisarum vulgare
## 5     sp1         0     1 Arisarum vulgare
## 6     sp1         0     1 Arisarum vulgare
```

usando o pacote **tidyr**

```
comm.tidy <- tidyr::gather(comm[, -1])

head(comm.tidy)
```

```
##   key value
## 1 sp1     0
## 2 sp1     0
## 3 sp1     0
## 4 sp1     0
## 5 sp1     0
## 6 sp1     0
```

```
dim(comm.tidy)
```

```
## [1] 5432    2
```

juntando todas as variáveis em uma única tabela

juntando 'comm.sp', com 'envir.coord'

```
comm.envir <- inner_join(x = comm.sp,
                         y = envir.coord,
                         by = "Sites")

dim(comm.sp)
```

```
## [1] 5432    4
```

```
dim(envir.coord)
```

```
## [1] 97 11
```

```
dim(comm.envir)
```

```
## [1] 5432   14
```