Introdução à Linguagem R - Workshop

Encontro 2

Davi Moreira

22 de Outubro, 2018

Sumário

1 Encontro 2

1.1 Dúvidas e revisão do conteúdo do encontro prévio

- 15 minutos serão reservados para dúvidas e revisão do conteúdo do encontro prévio.

1.2 Estrutura do encontro 2

2. PRÉ-PROCESSAMENTO DE DADOS

- Tidyverse tools: Pacote dplyr
- Filtro, ordenação, agregação, sumarização, merging, reshaping, criação de novas variáveis;
- Manipulação de strings e datas;

Até o final deste encontro o aluno deverá ser capaz de:

- Criar novas variáveis
- Filtrar, formatar e ajustar base de dados
- Unir bases de dados distintas

2 Dados para este encontro

2.1 Preparando dados

```
library(ffbase) # carregando o pacote

# definindo diretório
setwd("./dados/dados_encontro_1_ufpe/")

# Os seguintes passos serão dados:
# 1. Carregando bases grandes e salvando no formato ffdf
# 2. Filtrando bases para selecionar apenas dados do Estado de Pernambuco

# 1. Carregando bases grandes e salvando no formato ffdf ----

# DOCENTES ----
# carregando base de dados
docentes_ne <- read.csv2.ffdf(file = "DOCENTES_NORDESTE.csv", sep = "|", first.rows=100000)

# definindo diretório
setwd('..')  # move wd para nível anterior
setwd("./dados_encontro_1_ufpe/")

# salvando base no formato ffdf
save.ffdf(docentes_ne, dir = "./docentes_ne", overwrite = TRUE)
rm(list = ls()) # limpando ambiente de trabalho

# MATRICULA ----
# definindo diretório
setwd('..')  # move wd para nível anterior
setwd("./dados_encontro_1_ufpe/")
```

```r
# carregando base de dados
matricula_ne <- read.csv2.ffdf(file = "MATRICULA_NORDESTE.csv", sep = "|", first.rows=100000)

# definindo diretório
setwd('..')  # move wd para nível anterior
setwd("./dados_encontro_1_ufpe/")

# salvando base no formato ffdf
save.ffdf(matricula_ne, dir = "./matricula_ne", overwrite = TRUE)
rm(list = ls()) # limpando ambiente de trabalho

# TURMAS ----
# definindo diretório
setwd('..')  # move wd para nível anterior
setwd("./dados_encontro_1_ufpe/")

# carregando base de dados
turmas_ne <- read.csv2.ffdf(file = "TURMAS.csv", sep = "|", first.rows=100000)

# definindo diretório
setwd('..')  # move wd para nível anterior
setwd("./dados_encontro_1_ufpe/")

# salvando base no formato ffdf
save.ffdf(turmas_ne, dir = "./turmas_ne", overwrite = TRUE)
rm(list = ls()) # limpando ambiente de trabalho

# 2. Filtrando bases para selecionar apenas dados do Estado de Pernambuco ----
# MATRICULA ----
# definindo diretório
setwd('..')  # move wd para nível anterior
setwd("./dados_encontro_1_ufpe/")

# carregando base de dados
load.ffdf(dir="./matricula_ne")

# verificando estrutura da base de dados
dim(matricula_ne)

# Selecionando PE
matricula_pe <- subset(matricula_ne, CO_UF == 26)
dim(matricula_pe)

# transformando em data.frame
matricula_pe <- as.data.frame(matricula_pe) # definindo diretorio

# definindo diretório
setwd('..')  # move wd para nível anterior
setwd("./dados_encontro_2_ufpe/")

# salvando arquivo em formato RData
save(matricula_pe, file = "matricula_pe_censo_escolar_2016.RData")
```

```r
# DOCENTES ----
# definindo diretório
setwd('..')  # move wd para nível anterior
setwd("./dados_encontro_1_ufpe/")

# carregando base de dados
load.ffdf(dir="./docentes_ne")

# verificando estrutura da base de dados
dim(docentes_ne)

# Selecionando PE
docentes_pe <- subset(docentes_ne, CO_UF == 26)
dim(docentes_pe)

# transformando em data.frame
docentes_pe <- as.data.frame(docentes_pe)

# definindo diretorio
setwd('..')  # move wd para nível anterior
setwd("./dados_encontro_2_ufpe/")

# salvando arquivo em formato RData
save(docentes_pe, file = "docentes_pe_censo_escolar_2016.RData")

# TURMAS ----

# carregando base de dados
# definindo diretorio
setwd('..')  # move wd para nível anterior
setwd("./dados_encontro_1_ufpe/")

# carregando base de dados
load.ffdf(dir="./turmas_ne/")

# Selecionando PE
turmas_pe <- subset(turmas_ne, CO_UF == 26)
dim(turmas_pe)

# transformando em data.frame
turmas_pe <- as.data.frame(turmas_pe) # definindo diretorio

# definindo diretorio
setwd('..')  # move wd para nível anterior
setwd("./dados_encontro_2_ufpe/")

# salvando arquivo em formato RData
save(turmas_pe, file = "turmas_pe_censo_escolar_2016.RData")

# ESCOLAS ----
# definindo diretorio
setwd('..')  # move wd para nível anterior
setwd("./dados_encontro_1_ufpe/")
```

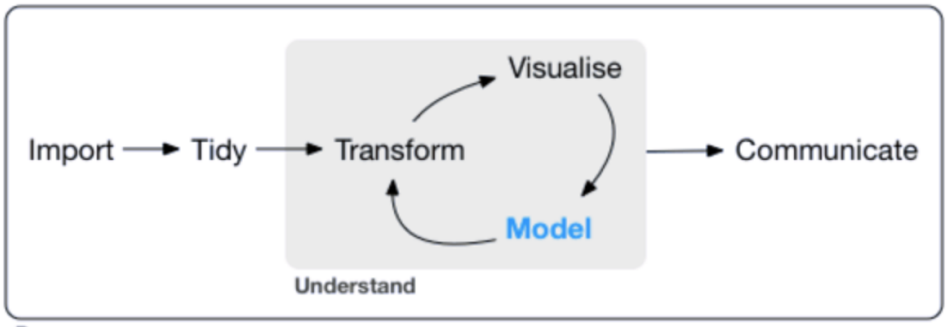
```r
# Carregando base de dados
escolas <- read.csv2("ESCOLAS.csv", sep = "|")

# Selecionando PE
escolas_pe <- subset(escolas, CO_UF == 26)
dim(escolas_pe)

# transformando em data.frame
escolas_pe <- as.data.frame(escolas_pe) # definindo diretorio

# definindo diretorio
setwd('..')
setwd("./dados_encontro_2_ufpe/")

# salvando arquivo em formato RData
save(escolas_pe, file = "escolas_pe_censo_escolar_2016.RData")
rm(list = ls()) # removendo todos os objetos do ambiente de trabalho
```

2.2 Carregando dados

```r
# definindo diretório
setwd("./dados_encontro_2_ufpe/")

# carregando dados
load("matricula_pe_censo_escolar_2016.RData")
load("docentes_pe_censo_escolar_2016.RData")
load("turmas_pe_censo_escolar_2016.RData")
load("escolas_pe_censo_escolar_2016.RData")
```

3 Tidyverse tools

O Tidyverse é um conjunto de pacotes que funcionam em harmonia porque compartilham representações de dados comuns. O pacote tidyverse foi projetado para facilitar a instalação e o carregamento de pacotes em um único comando.



Pacotes Tidyverse:

- **ggplot2**, para visualização de dados.
- **dplyr**, para manipulação de dados.

- **tidyr**, para ajustes de dados.
- **stringr**, para strings.
- forcats, para factors.
- readr, para importação de dados.
- **purrr**, para programação funcional.
- tibble, para tibbles.

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

3.1 Novo operador: %>%

O operador %>% torna a leitura de códigos no R mais lógica e fácil. Para utilizá-lo, vamos instalar e carregar o pacote magrittr^(O conteúdo dessa seção foi fortemente influenciado pelo conteúdo do seguinte curso: http://material.curso-r.com/pipe/).

```
install.packages("magrittr")  # instalando o pacote
library(magrittr)  # carregando o pacote
```

Sua interpretação é bem simples e pode ser entendida como: "use o objeto do lado esquerdo como primeiro argumento da função do lado direito".

```
library(magrittr)

# definindo vetor
x <- c(1, 2, 3, 4)

# As duas linhas abaixo são equivalentes.
sqrt(sum(x))  # sem o pipe
x %>% sum %>% sqrt # agora com o pipe.
```

O código `x %>% sum %>% sqrt` pode ser lido do seguinte modo: envie o objeto x como argumento da função sum() e, em seguida, envie a saída da expressão sum(x) como argumento da função sqrt().

Para que o resultado ou objeto do lado esquerdo vá para outro argumento do lado direito que não o primeiro, usa-se um `.` como marcador.

```
reg <- airquality %>%
  na.omit %>%
  lm(Ozone ~ Wind + Temp + Solar.R, data = .)

summary(reg)
```

Para mais informações sobre o operador %>% (pipe), acesse Ceci n'est pas un pipe. Outra excelente referência pode ser encontrada no livro R for Data Science.

3.1.0.1 Ativdade prática

```
# Reescreva as expressões abaixo utilizando o %>%.
# Use a função magrittr::divide_by() para divisão caso necessário.

x <- seq(0, 100, 5)

# 1.  sqrt(mean(x))
# 2. round(mean(c(1:10)/2), digits = 1)
```

```r
# Reescreva o codigo abaixo sem utilizar o %>%.
reg <- airquality %>%
  na.omit %>%
  lm(Ozone ~ Wind + Temp + Solar.R, data = .)

# Use summary(reg) para verificar o resultado do modelo
```

3.2 Stringr

As "strings" (cadeias de caracteres) desempenham um papel importante em muitas tarefas de limpeza e preparação de dados. O pacote `stringr` fornece um conjunto coeso de funções projetadas para tornar o trabalho com strings o mais fácil possível. Uma ótima referência para trabalhar com strings pode ser encontrada no livro R para Data Science. Outra ótima referência pode ser encontrada no stringr::cheat sheet.

```r
library(stringr)

string1 <- "Esta é uma string"
string1
writeLines(string1)

string2 <- 'para incluir "aspas" numa string, deve se usar aspas simples na string'
string2
writeLines(string2)

string2 <- "ou ao invés de aspas simples, pode-se usar \\ por exemplo: \"aspas\". "
string2
writeLines(string2)

# tamanho de uma string
str_length(c("a", "R para data science", NA))

# concatenar
str_c("x", "y", "z")

name <- "Fulano"
time_of_day <- "dia"
birthday <- "2018-10-23"
birthday_bin <- if(birthday == Sys.Date()){
  TRUE
  }else{
    FALSE}

str_c(
  "Bom ", time_of_day, " ", name,
  if (birthday_bin) " e Feliz Aniversário",
  "."
)

# Subsetting strings
x <- c("Maça", "Banana", "Pera")
str_sub(x, 1, 3)

# caixa alta e caixa baixa
```

```r
str_to_lower(x)
str_to_upper("fulano")

# Detect matches e count
x <- c("Maça", "Banana", "Pera")
str_detect(x, "e")

x <- c("Maça", "Banana", "Pera")
str_count(x, "a")

# aplicacao num data frame
df <- tibble(
  word = words,
  i = seq_along(word)
)
df %>%
  filter(str_detect(words, "x$"))  # '$' significa terminada em

# Replacing matches
str_replace_all(x, "[aeiou]", "-") # [aeiou] todas as vogais
str_replace_all(x, "[MB]", "-") # [MB] quaisquer caracteres

# Spliting
string_vec <- c(string1, string2)

string_vec_split <- string_vec %>%
                        str_split(" ")
string_vec_split

# Outros usos: Regular Expressions
?regex

sentenca <- c(string_vec, "suponha que tenha números 12345 numa sentença")
sentenca
str_detect(sentenca, "[:digit:]")
```

3.3 Datas

```r
library(tidyverse)
library(lubridate)

# criando data
matricula_pe_nasc <- matricula_pe %>%
      select(CO_PESSOA_FISICA, NU_DIA, NU_MES, NU_ANO) %>%
      mutate(nascimento = make_date(NU_ANO, NU_MES, NU_DIA))

head(matricula_pe_nasc)
is.Date(matricula_pe_nasc$nascimento)

# outra forma
matricula_pe_nasc$nascimento2 <- as.Date(paste(matricula_pe$NU_ANO,
                                               matricula_pe$NU_MES,
```

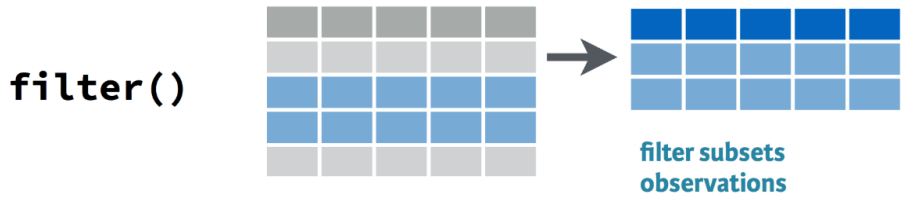
```
                                          matricula_pe$NU_DIA, sep = "-"),
                                    "%Y-%m-%d")

head(matricula_pe_nasc)
is.Date(matricula_pe_nasc$nascimento2)

# obtendo ano
matricula_pe_nasc$ano <- year(matricula_pe_nasc$nascimento)
head(matricula_pe_nasc)
```

3.4 Pacote dplyr

O dplyr é uma gramática de manipulação de dados, fornecendo um conjunto consistente de verbos que ajudam a resolver os desafios mais comuns de manipulação de dados:

- **filter()**: seleciona casos com base em seus valores.
- **arrange()**: altera a ordenação das linhas.
- **mutate()**: adiciona novas variáveis que são funções de variáveis existentes
- **summarise() e group_by**: reduz vários valores para um único resumo por grupo desejável.
- select(): escolhe variáveis com base em seus nomes.

3.4.1 Filter



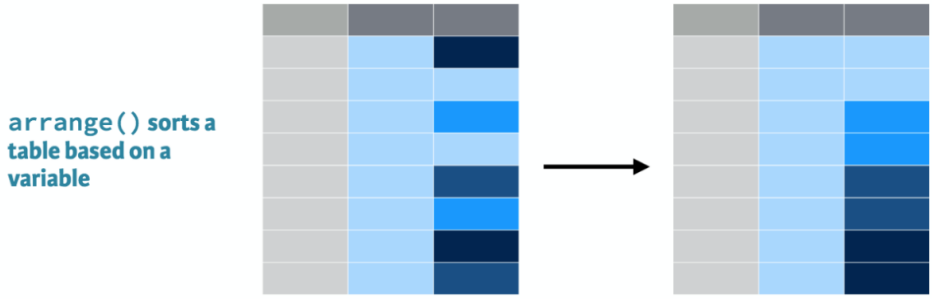

```
library(tidyverse)
setwd("./dados/")
load("matricula_pe_censo_escolar_2016.RData")
names(matricula_pe)
matricula_pe_selecao <- matricula_pe %>% filter(NU_IDADE > 10, TP_SEXO == 2)
dim(matricula_pe_selecao)
head(matricula_pe_selecao)
```

3.4.1.1 Atividade prática:

```
# faça o filtro abaixo usando a função subset
matricula_pe_selecao <- matricula_pe %>% filter(NU_IDADE > 10, TP_SEXO == 2)
dim(matricula_pe_selecao)
head(matricula_pe_selecao)

# Selecione outras variáveis de interesse e repita a operação na base de matricula_pe

# Na base turmas_pe, selecione apenas aquelas que têm disciplinas de língua indigena
```

```
# Quantas são?
# Quantos municípios diferentes?
```

3.4.2 Arrange

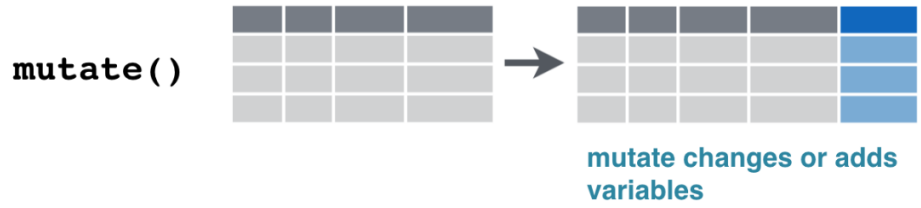



```
library(tidyverse)
setwd("./dados/")
load("matricula_pe_censo_escolar_2016.RData")
matricula_pe_selecao <- matricula_pe %>% arrange(desc(NU_IDADE))
head(matricula_pe_selecao)
summary(matricula_pe_selecao$NU_IDADE)

matricula_pe_selecao <- matricula_pe %>%
  filter(NU_IDADE > 10, TP_SEXO == 2) %>%
    arrange(desc(NU_IDADE))

head(matricula_pe_selecao)
summary(matricula_pe_selecao$NU_IDADE)
```

3.4.2.1 Atividade prática:

```
# Faça o mesmo filtro na base de matricula, mas ordene pelo número de matrículas em ordem crescente

# Filtre a base docentes_pe selecionando professores negros (Pretos e Pardos) e ordene a base pela idad
# Quantos são? Qual a proporção no Estado?
```

3.4.3 Mutate




```
library(tidyverse)
setwd("./dados/")
load("matricula_pe_censo_escolar_2016.RData")
```

```r
matricula_pe_selecao <- matricula_pe %>%
                      mutate(FX_IDADE = ifelse(NU_IDADE <= 10, "até 10", "acima de 10"))
head(matricula_pe_selecao)
summary(factor(matricula_pe_selecao$FX_IDADE))
```

3.4.3.1 Atividade prática:

```r
# Com a base de escolas, use o codigo de Recife no IBGE para criar uma variável binária
# que indique se a escola está na capital do Estado ou não.
# Qual a proporção de escolas na capital?
```

3.4.4 Summarise e Group_by

```r
library(tidyverse)
setwd("./dados/")
load("matricula_pe_censo_escolar_2016.RData")

matricula_pe_selecao <- matricula_pe %>% group_by(CO_MUNICIPIO_END, TP_SEXO) %>%
  summarise(n_alunos = n(), media_idade = mean(NU_IDADE))

matricula_pe_selecao
```

3.4.4.1 Atividade prática:

```r
# Com a base de turmas, obtenha nova matriz de dados que apresente a média de matriculas e
# o número de turmas por municipio
```

3.5 Pacote tidyr

O principal objetivo do `tidyr` é auxiliar o analista na reestruturação da base de dados para propósitos específicos. Duas são suas principais funções:

- `gather()`
- `spread()`

3.5.1 Gather

Transforma a base de dados no formato "wide" ("amplo") para "long" ("longo").

```r
nutri <- tibble(
  dia = as.Date('2016-01-01') + 0:29,
  X = rnorm(30, 0, 1),
  Y = rnorm(30, 0, 2),
  Z = rnorm(30, 0, 4)
)
dim(nutri)

nutri_long <- gather(nutri, item, valor, -dia)
dim(nutri_long)
```

3.5.1.1 Atividade prática:

```
# Utilize o codigo abaixo para criar uma base de dados fictícia que tenha as notas
# médias de 100 alunos de três escolas do Estado de Pernambuco por ano.
install.packages("lubridate")
library(lubridate)

notas <- tibble(
  ano = year(as.Date('1990-01-01')) + 0:27,
  A = sample.int(10, 28, replace = T),
  B = sample.int(10, 28, replace = T),
  C = sample.int(10, 28, replace = T)
)

# Obtenha as estatísticas descritivas das notas por escola
# Transforme a base do formato wide para long (notas_long)
# Obtenha as estatísticas descritivas das notas por ano
```

3.5.2 Spread

Transforma a base de dados no formato "long" ("longo") para "wide" ("amplo").

```
dim(nutri_long)
nutri_wide <- nutri_long %>% spread(item, valor)
dim(nutri_wide)
dim(nutri)
```

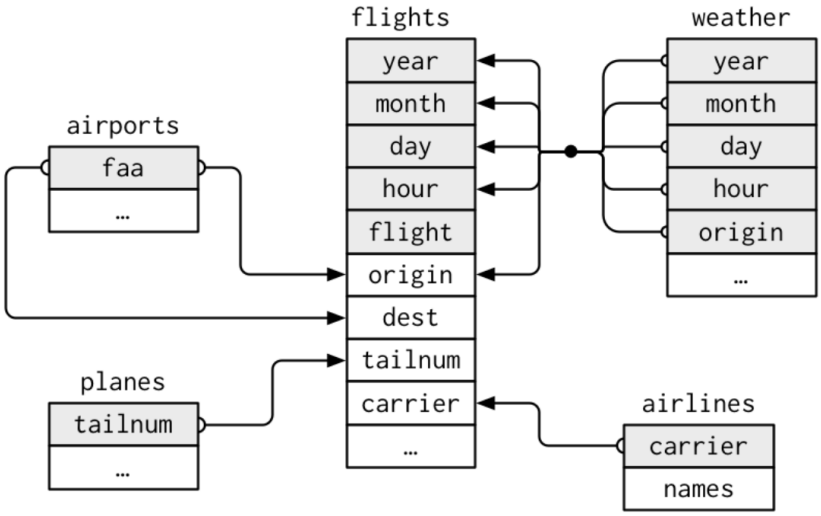
3.5.2.1 Atividade prática:

```
# Coloque a base da atividade anterior no formato wide
```

3.6 Join ou Merge

Não é raro que uma análise de dados envolva mais de uma base de dados. É comum ter que combinar diferentes bases para responder às perguntas de interesse. Bases de dados que possuem alguma conexão substantiva e que podem ser mescladas constituem o que conhecemos como bases de dados relacionais.

As relações são definidas entre pares de bases, de modo que as relações de três ou mais bases são sempre uma propriedade das relações entre cada par. Vejamos no exemplo abaixo:



*flights se conecta a planes por: tailnum.

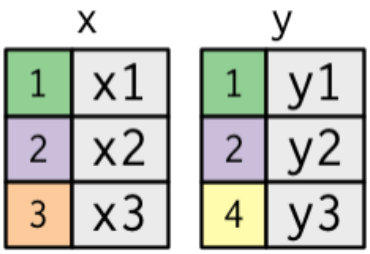
*flights se conecta a airlines por: carrier.

*flights se conecta a airports por:: origin e dest.

*flights se conecta a weather por: origin, year, month, day e hour.

Para exemplificar como podemos unir diferentes bases de dados, vamos utilizar os exemplos apresentados no livro R para Data Science. Daremos ênfase às seguintes funções:

- inner_join()
- left_join()
- right_join()
- full_join()

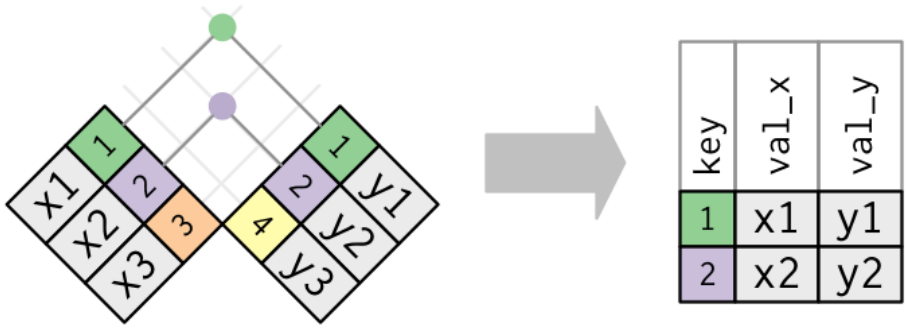



```
# Contruindo bases para exemplos:
library(tidyverse)
x <- tribble(
  ~key, ~val_x,
     1, "x1",
     2, "x2",
     3, "x3"
)
y <- tribble(
  ~key, ~val_y,
     1, "y1",
     2, "y2",
```

```
    4, "y3"
)
```

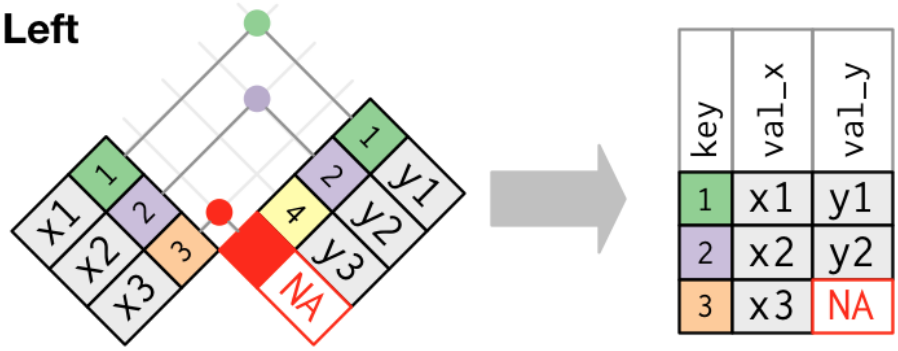
3.6.1 inner_join()

Combina pares de observações sempre que suas chaves são iguais:



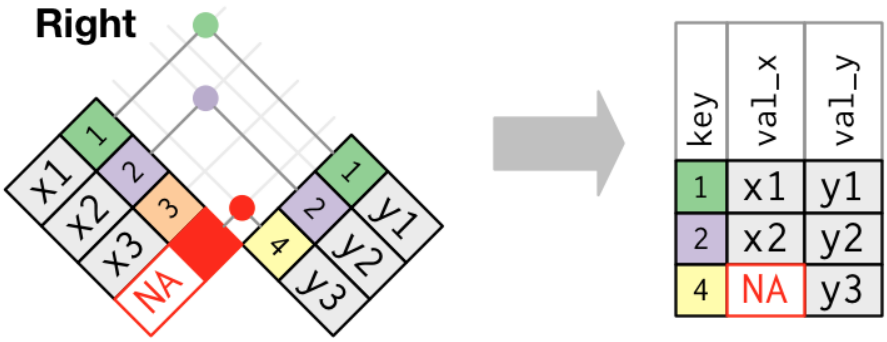

```
x %>%
  inner_join(y, by = "key")
```

3.6.2 left_join()



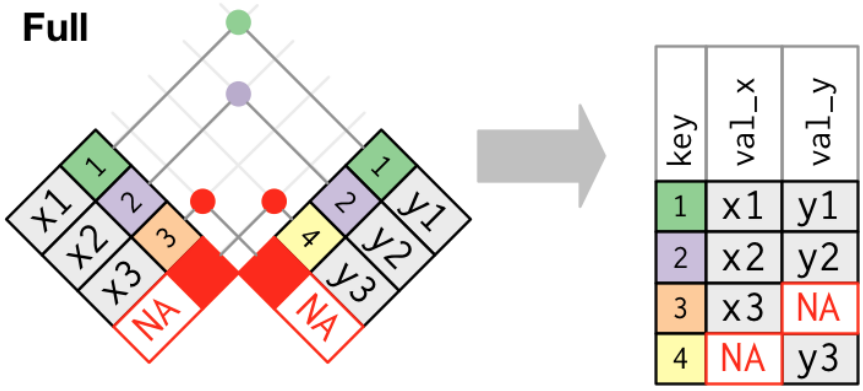

```
x %>%
  left_join(y, by = "key")
```

3.6.3 right_join()




```
x %>%
  right_join(y, by = "key")
```

3.6.4 full_join()




```
x %>%
  full_join(y, by = "key")
```

3.6.5 Merge

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)`

3.6.5.1 Atividade prática:

```
# Com os dados do censo escolar, obtenha o número de docentes e o número de
# matriculas por município do Estado de Pernambuco. Com essas duas novas bases,
# crie uma terceira base de dados que tenha quatro colunas: codigo do municipio,
# número de docentes, número de matrículas, número de matrículas por docente.
```

4 Atividade Prática

- Não tivemos a chance de conhecer todos os pacotes do `tydiverse`. Selecione um dos pacotes abaixo e desenvolva alguma aplicação sobre as bases de dados do Censo Escolar de 2016.
- `forcats`, para factors.
- `readr`, para importação de dados.
- `purrr`, para programação funcional.
- `tibble`, para tibbles.

5 Links úteis para o próximo encontro

- Web Scraping
- rvest Tutorial