Dados Relacionais

Nicholas A. C. Marino

nac.marino@gmail.com

github.com/nacmarino/compartilhaR

Estrutura da Aula

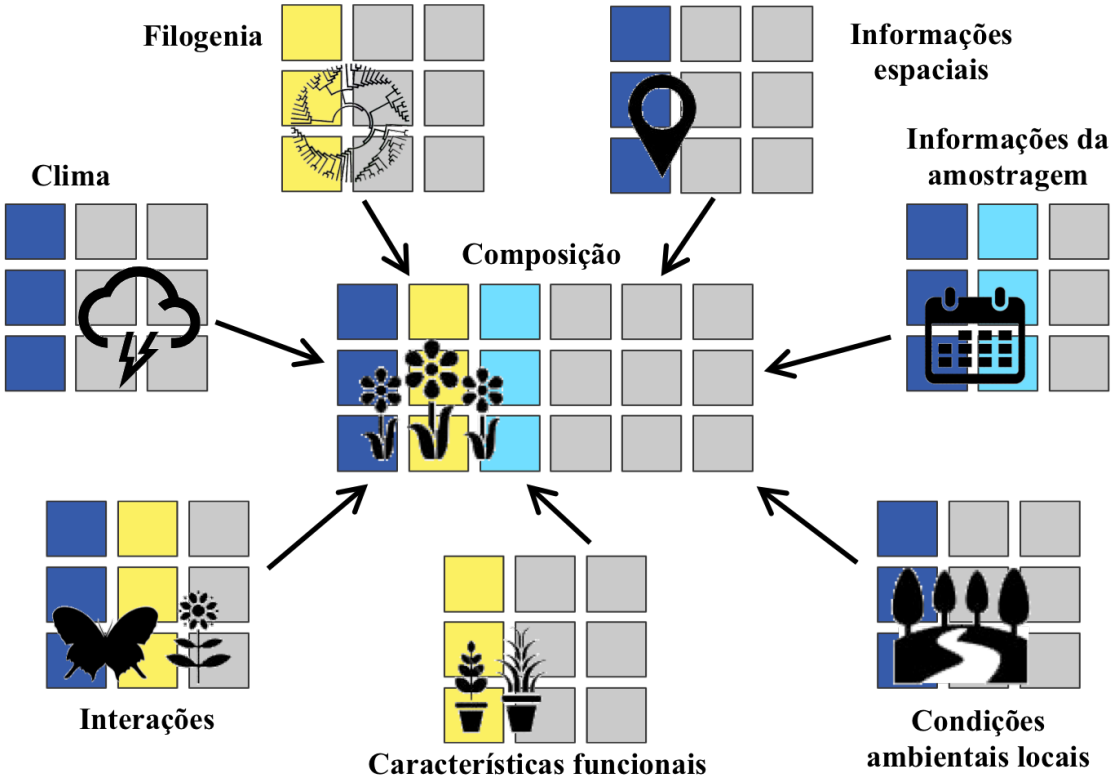
1. Dados relacionais
2. Funções `join` - mutate
3. Funções `join` - filter

Dados relacionais {.smaller}

- Quando estamos trabalhando com um conjunto de dados, normalmente pensamos em manter todas as informações juntas em um único arquivo.
- Apesar disso, nem todo dado pode ser incluído da mesma forma em uma tabela:
  - Dados das características das espécies *vs* dados da abundância das espécies;
  - Dados abióticos *vs* dados bióticos;
  - Dados das coordenadas amostradas *vs* dados dos arquivos de mapas;
  - Dados necessários para o cálculo de uma variável composta (e.g., vazão de um rio, emissão de $CO_{[2]}$, dados para o cálculo de alcalinidade da água);
  - …
- Além disso, é provável que ao tentar manter todos os dados em um único arquivo você acabe se repetindo bastante.

Dados relacionais

- Uma solução para isso é criar uma estrutura de base de **dados relacionais**.
- De acordo com este tipo de estrutura, os dados são armazenados em diferentes tabelas, que compartilham certos elementos que descrevem a inter-relação entre elas.



Utilizando a estrutura relacional no R

- Uma vez que tenhamos um conjunto de dados relacionais, podemos utilizar as funções `join` para realização a manipulação e processamento dos dados.
- Existem duas famílias de função `join`:
  - **Mutate join:** junta duas tabelas em uma nova tabela, mudando ou não o número de linhas e mantendo as colunas das duas tabelas.
    - `inner_join`
    - `left_join`
    - `right_join`
    - `full_join`
  - **Filter join:** junta duas tabelas, sempre mudando o número de linhas e mantendo as colunas de apenas uma delas.
    - `semi_join`
    - `anti_join`

Para seguir a demonstração

- Para trabalhar os exemplos a seguir, vamos criar duas tabelas com estrutura relacional (como usamos um `tibble`, precisamos carregar o tidyverse` também).

```
library(tidyverse)
df1 <-  tibble(id = c(1, 2, 3), x = c("x1", "x2", "x3"))
```

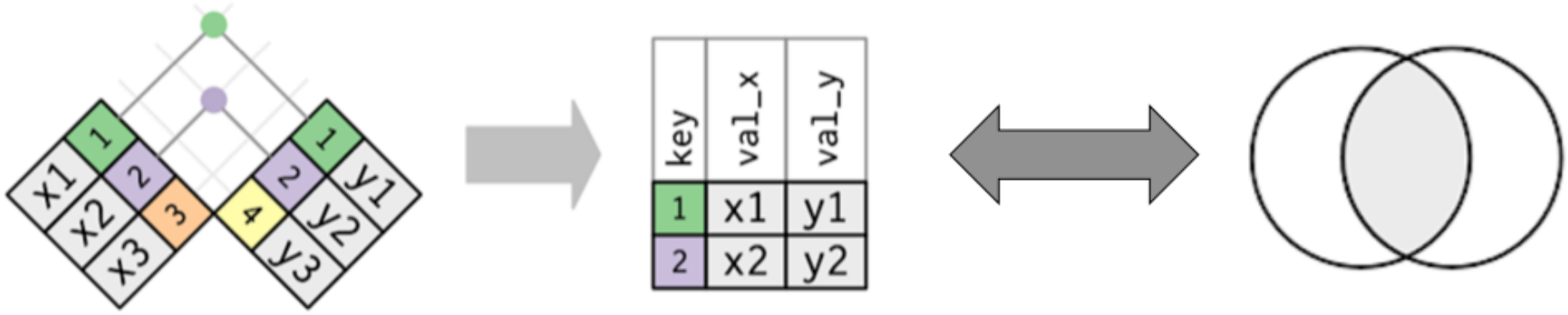
```
## # A tibble: 3 x 2
##      id x
##   <dbl> <chr>
## 1     1 x1
## 2     2 x2
## 3     3 x3
```

```
df2 <- tibble(id = c(1, 2, 4), y = c("y1", "y2", "y3"))
```

```
## # A tibble: 3 x 2
##      id y
##   <dbl> <chr>
## 1     1 y1
## 2     2 y2
## 3     4 y3
```

`inner_join`

- Retorna uma nova tabela com todas as linhas que `x` e `y` tem em comum, e todas as colunas de `x` e `y`.



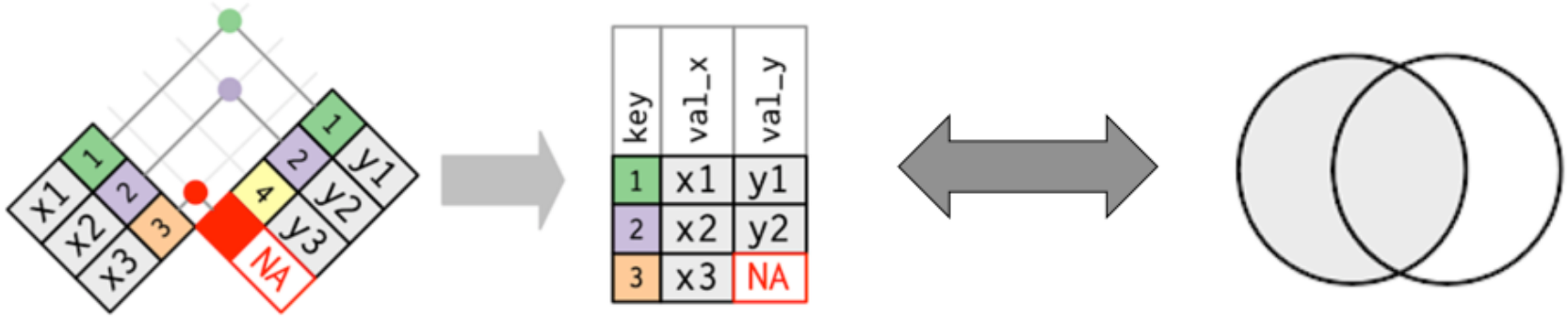
inner_join

```
inner_join(x = df1, y = df2, by = "id")
```

```
## # A tibble: 2 x 3
##      id x     y
##   <dbl> <chr> <chr>
## 1     1 x1    y1
## 2     2 x2    y2
```

`left_join`

- Retorna uma nova tabela com todas as linhas de `x` e todas as colunas de `x` e `y`. Um **`NA`** é retornado para os valores das colunas de `y` onde não houver correspondência com as linhas de `x`.



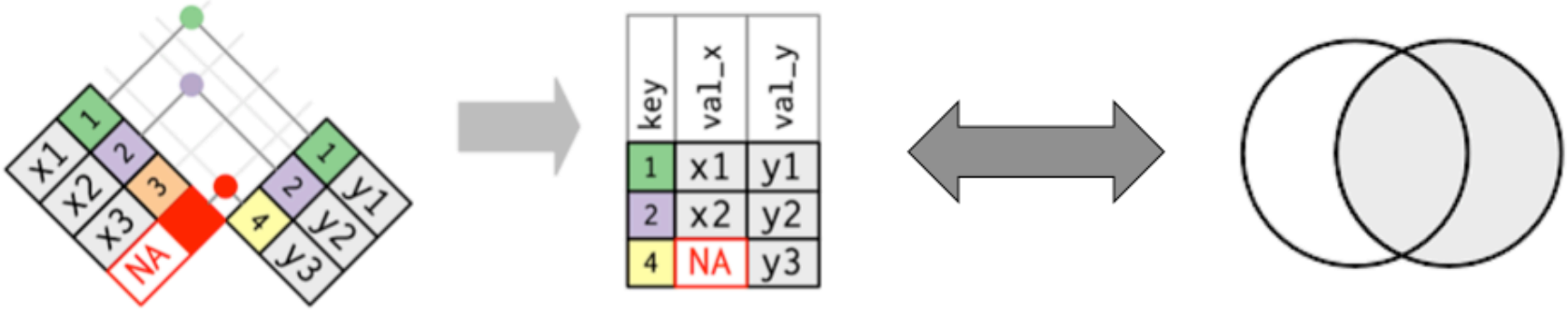
left_join

```
left_join(x = df1, y = df2, by = "id")
```

```
## # A tibble: 3 x 3
##      id x     y
##   <dbl> <chr> <chr>
## 1     1 x1    y1
## 2     2 x2    y2
## 3     3 x3    <NA>
```

`right_join`

- Retorna uma nova tabela com todas as linhas de `y` e todas as colunas de `x` e `y`. Um **`NA`** é retornado para os valores das colunas de `x` onde não houver correspondência com as linhas de `y`.



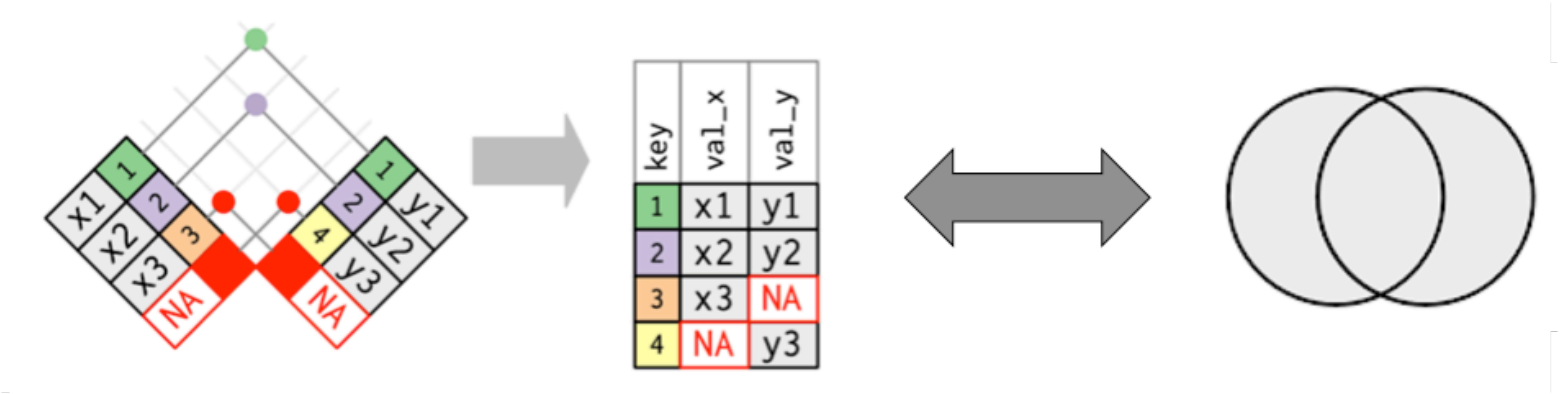
right_join

```
right_join(x = df1, y = df2, by = "id")
```

```
## # A tibble: 3 x 3
##      id x     y
##   <dbl> <chr> <chr>
## 1     1 x1    y1
## 2     2 x2    y2
## 3     4 <NA>  y3
```

`full_join`

- Retorna uma nova tabela com todas as linhas e colunas de `x` e `y`. Um **`NA`** é retornado para os valores das colunas de `y` onde não houver correspondência com as linhas de `x`, e vice-versa.



full_join

```
full_join(x = df1, y = df2, by = "id")
```

```
## # A tibble: 4 x 3
##      id x     y
##   <dbl> <chr> <chr>
## 1     1 x1    y1
## 2     2 x2    y2
## 3     3 x3    <NA>
## 4     4 <NA>  y3
```

Funções `join` com mais de um elemento relacional

- Nos exemplos anteriores, o argumento `by` sempre recebe um *vetor* que contém o elemento que descreve a relação entre as duas tabelas que se quer unir (`x` e `y`).
- Assim, podemos passar um vetor com tantos elementos quanto quisermos para descrever a relação entre duas tabelas.

```
df3 <- tibble(id1 = c(1, 1, 1, 1, 2, 2, 2, 2),
              id2 = c("a", "b", "a", "a","a", "b", "b", "b"),
              x = c("x1", "x2", "x3", "x4", "x5", "x6", "x7", "x8"))
df4 <- tibble(id1 = c(1, 1, 2, 2),
              id2 = c("a", "b", "a", "b"),
              y1 = c("y1", "y2", "y3", "y4"))
```

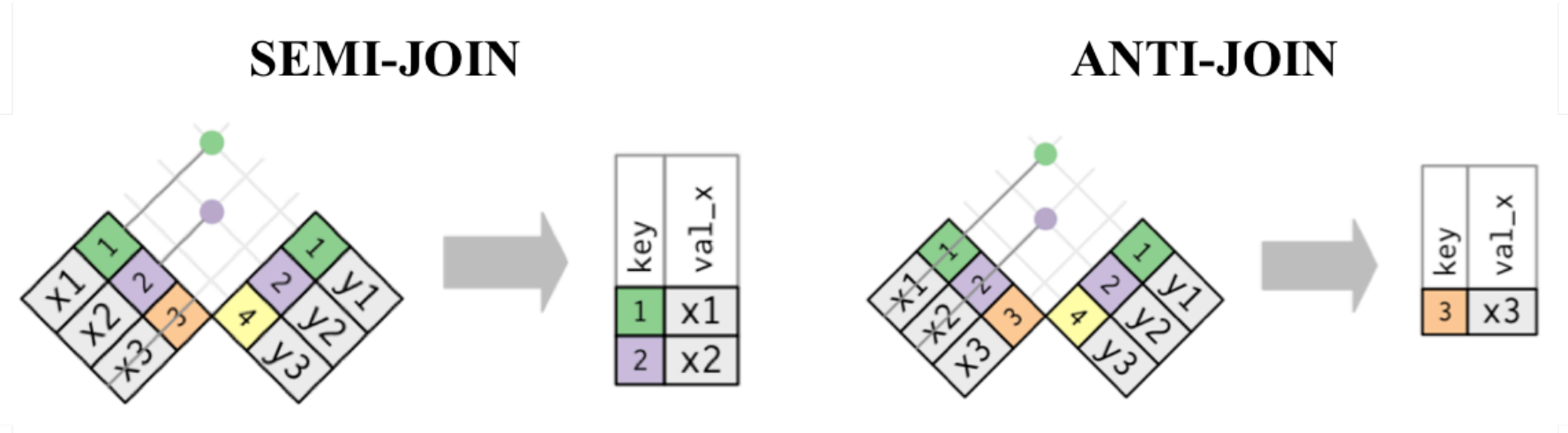
Funções `join` com mais de um elemento relacional

```
inner_join(x = df3, y = df4, by = c("id1", "id2"))
```

```
## # A tibble: 8 x 4
##     id1 id2   x     y1
##   <dbl> <chr> <chr> <chr>
## 1     1 a     x1    y1
## 2     1 b     x2    y2
## 3     1 a     x3    y1
## 4     1 a     x4    y1
## 5     2 a     x5    y3
## 6     2 b     x6    y4
## 7     2 b     x7    y4
## 8     2 b     x8    y4
```

`semi_join` e `anti_join`

- `semi_join`: retorna uma nova tabela com todas as linhas que `x` e `y` tem em comum, retendo apenas as colunas de `x`.
- `anti_join`: retorna uma nova tabela com as linhas que `x` que **não são compartilhadas** com `y` , retendo apenas as colunas de `x`.



`semi_join` e `anti_join`

- `semi_join`: retorna uma nova tabela com todas as linhas que `x` e `y` tem em comum, retendo apenas as colunas de `x`.

```
semi_join(x = df1, y = df2, by = "id")
```

```
## # A tibble: 2 x 2
##      id x
##   <dbl> <chr>
## 1     1 x1
## 2     2 x2
```

`semi_join` e `anti_join`

- `anti_join`: retorna uma nova tabela com as linhas que `x` que **não são compartilhadas** com `y` , retendo apenas as colunas de `x`.

```
anti_join(x = df1, y = df2, by = "id")
```

```
## # A tibble: 1 x 2
##      id x
##   <dbl> <chr>
## 1     3 x3
```

`Filter join` sem o uso das funções `join` correspondentes

- Uma das formas de determinar a similaridade entre dois vetores é através de um teste lógico.

```
df1$id %in% df2$id
```

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

- Como o resultado desse teste é um vetor lógico, poderíamos utilizar essa informação para realizar a indexação de uma das tabelas.

```
## resultado similar ao semi_join
df1[df1$id %in% df2$id,]
```

```
## # A tibble: 2 x 2
##      id x
##   <dbl> <chr>
## 1     1 x1
## 2     2 x2
```

`Filter join` sem o uso das funções `join` correspondentes

- Por outro lado, podemos usar a função `filter` para obter o mesmo resultado.

```
filter(.data = df1, id %in% df2$id)
```

```
## # A tibble: 2 x 2
##      id x
##   <dbl> <chr>
## 1     1 x1
## 2     2 x2
```

Exercício 1

1. Carregue todos os dados referentes ao financiamento de pesquisa do CNPq (**projetos**, **revistas** e **publicacoes**);
2. Determine qual(is) variável(is) descreve(m) a relação entre estas três tabelas de dados;
3. Calcule o número médio de publicacões e o número médio de citações dos trabalho publicados para cada projeto financiado;
4. Calcule o índice de impacto médio e máximo (i.e., SJR) das publicacoes de cada projeto;
5. Adicione estas informações à tabela que descreve as características gerais de cada projeto;
6. Importe estas informações, utilizando o formato `.csv`.