Manipulação de Dados

Nicholas A. C. Marino

nac.marino@gmail.com

github.com/nacmarino/compartilhaR

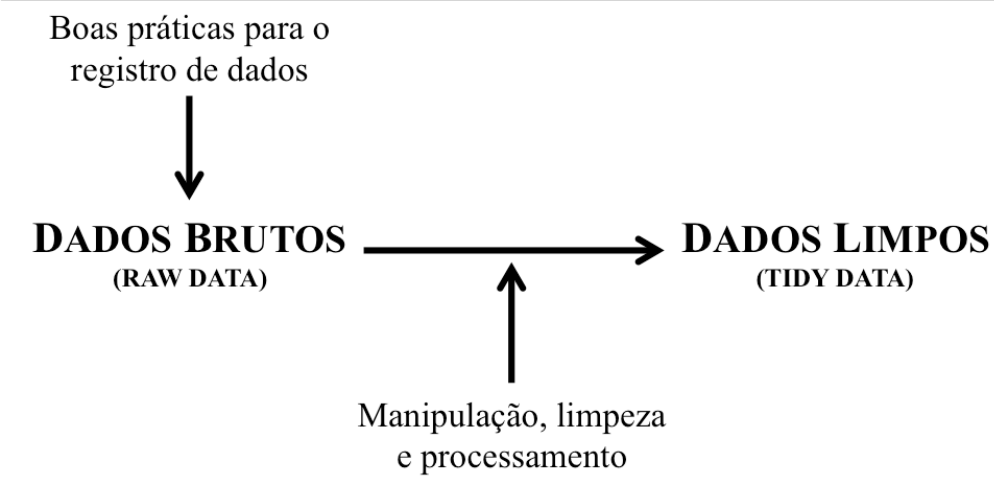
Elementos da Aula

1. A natureza dos dados
2. Métodos orientados ao conteúdo das colunas
3. Métodos orientados à tabela de dados
4. De largo para longo, e de volta outra vez

A natureza dos dados

Um dado pode estar em duas diferentes 'fases de maturidade':

1. **Dados brutos (*raw data*):** são os dados em sua forma mais bruta, recém tabelados, com todos os erros de digitação, de unidade, e etc…
2. **Dados limpos (*tidy data*):** são os dados em uma forma limpa. Aqui, os dados brutos foram checados e corrigidos, erros de digitação desfeitos, unidades transformadas e etc. Todo e qualquer nova variável que pode ser gerada com os dados brutos está aqui.

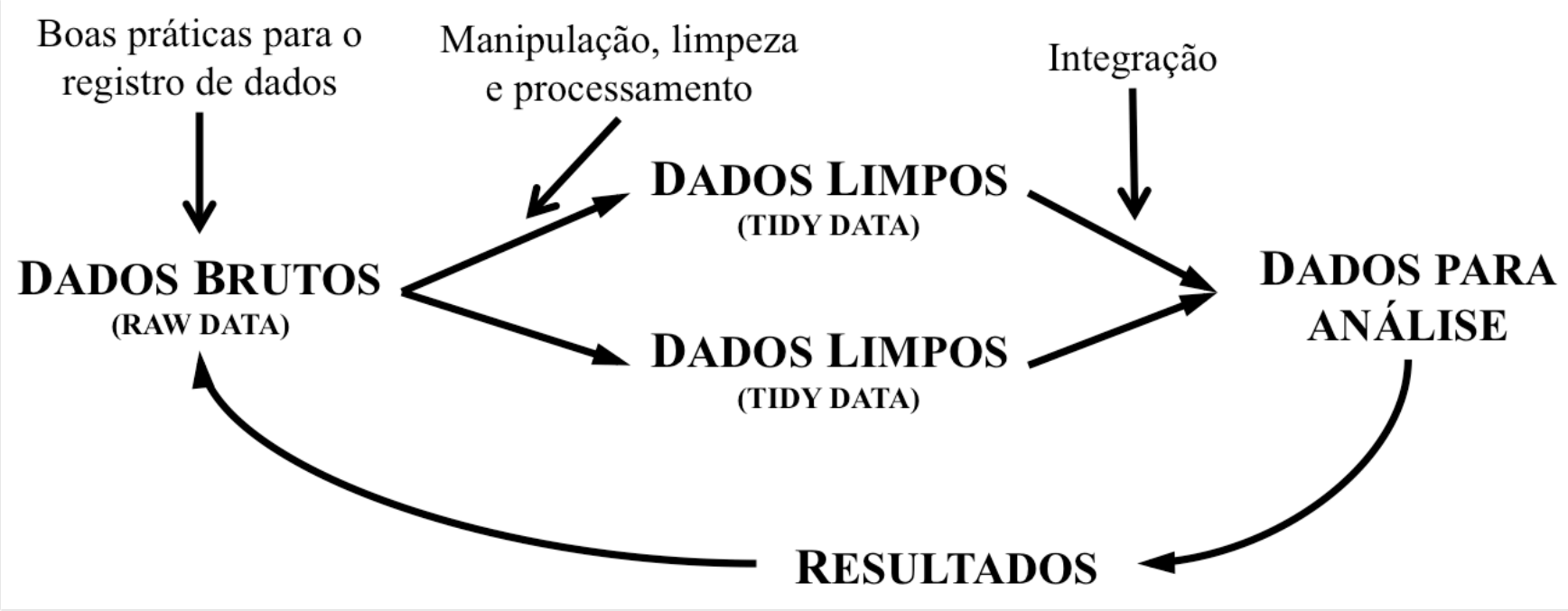


A natureza dos dados

Além disso, mesmo os dados limpos podem não estar prontos para o uso ou, ainda, existirem dados derivados que serão o enfoque do seu trabalho.

1. **Dados para análise:** normalmente você não precisa de todos os dados que você limpou e/ou alguns dos dados úteis para a análise podem estar em outras tabelas. Assim, ao invés de começar toda a análise de dados removendo àquelas informações que não são úteis e buscando àquelas outras que são, você também pode criar recortes de dados que serão específicos para certas tarefas.
2. **Dados dos resultados das análises:** após rodar uma análise você pode exportar os resultados para fora do R. Ao fazer isso, estes dados retornam para a etapa número 1 - você precisa ajeitar os nomes das colunas, casas decimais,…

A natureza dos dados



A natureza dos dados

Outro ponto importante é que algumas análises exigem que os dados sejam apresentados de uma forma específica, o que também leva à duas formas de apresentar um mesmo dado.

<table>
<tr><th></th><th>Largo</th><th>Longo</th></tr>
<tr><td>Abundância</td><td>
<table>
<tr><th>site</th><th>sp1</th><th>sp2</th><th>sp3</th></tr>
<tr><td>site1</td><td>20</td><td>10</td><td>0</td></tr>
<tr><td>site2</td><td>5</td><td>0</td><td>10</td></tr>
<tr><td>site3</td><td>0</td><td>15</td><td>0</td></tr>
</table>
</td><td>
<table>
<tr><th>site</th><th>especie</th><th>abundancia</th></tr>
<tr><td>site1</td><td>sp1</td><td>20</td></tr>
<tr><td>site1</td><td>sp2</td><td>10</td></tr>
<tr><td>site2</td><td>sp1</td><td>5</td></tr>
<tr><td>site2</td><td>sp3</td><td>10</td></tr>
<tr><td>site3</td><td>sp2</td><td>15</td></tr>
</table>
</td></tr>
<tr><td>P/A</td><td>
<table>
<tr><th>site</th><th>sp1</th><th>sp2</th><th>sp3</th></tr>
<tr><td>site1</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>site2</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>site3</td><td>0</td><td>1</td><td>0</td></tr>
</table>
</td><td>
<table>
<tr><th>site</th><th>especie</th></tr>
<tr><td>site1</td><td>sp1</td></tr>
<tr><td>site1</td><td>sp2</td></tr>
<tr><td>site2</td><td>sp1</td></tr>
<tr><td>site2</td><td>sp3</td></tr>
<tr><td>site3</td><td>sp2</td></tr>
</table>
</td></tr>
</table>

A natureza dos dados

- Uma parte comum e bastante importante em todas essas fases é a manipulação, limpeza e processamento de dados (*tidying data*).
- É aqui que vamos preparar os dados para o uso em uma análise, para a criação de uma tabela com os resultados que encontramos e, também, para a confecção de figuras.
- Mas também é onde:
  - Você normalmente faz tudo de forma manual;
  - Você não mantém registro escrito do que está fazendo;
  - Você vai criar múltiplas versões de uma mesma planilha, pois não sabe se as coisas que você está manipulando, mexendo e editando fazem sentido ou estão corretas;
  - Você perde tempo da forma mais repetitiva possível - a não ser que você use uma linguagem de programação! =]

Objetivos da manipulação, limpeza e processamento de dados

1. Criar e/ou eliminar novas variáveis (normalmente, nas colunas);
2. Substituir valores que foram digitados errados;
3. Substituir palavras e expressões que estejam má digitadas ou onde hajam nomes melhores;
4. Modificar os nomes das variáveis (normalmente, as colunas);
5. Modificar os nomes dos níveis das variáveis (normalmente, os valores das linhas de uma determinada coluna);
6. Separar a informação de uma coluna em duas ou mais;
7. Rearranjar a ordem das colunas;
8. Selecionar as colunas que vão compor os dados que serão analisados;
9. Passar os dados de um formato longo para um formato largo (e vice-versa);
10. Juntar dados que estão separados em planilhas diferentes;
11. Remover `NA`s;
12. Selecionar sub-conjuntos dos dados para destinações diferentes (e.g.,

`tidyverse`

- Existem muitas funções na `base` do R que podem ser utilizadas para a manipulação, limpeza e processamento de dados.
- No entanto, muitos dos pacotes mais úteis para estas tarefas estão organizados dentro de um pacote 'guarda-chuva', chamado `tidyverse`.

```
library(tidyverse)
tidyverse_packages()
```

```
##  [1] "broom"       "cli"         "crayon"      "dplyr"       "dbplyr"
##  [6] "forcats"     "ggplot2"     "haven"       "hms"         "httr"
## [11] "jsonlite"    "lubridate"   "magrittr"    "modelr"      "purrr"
## [16] "readr"       "readxl\n(>=" "reprex"      "rlang"       "rstudioapi"
## [21] "rvest"       "stringr"     "tibble"      "tidyr"       "xml2"
## [26] "tidyverse"
```

Exercício 1

- Vamos utilizar todas as seis tabelas de dados abaixo nas tarefas e exercícios a seguir.

1. Importe para o R as seguintes tabelas:
   - **dados dos projetos.csv**, e atribua este arquivo ao objeto **projetos**;
   - **publicacoes.xls**, e atribua este arquivo ao objeto **publicacoes**;
   - **revistas.xlsx**, e atribua este arquivo ao objeto **revistas**.
2. Também, carregue dois conjuntos de dados que estão disponíveis dentro de pacotes:
   - **varechem** e **varespec**, disponíveis no pacote `vegan`;
   - **gapminder** disponível no pacote `gapminder`.

Métodos orientados ao conteúdo das colunas

- É muito comum que cometamos erros de digitação ao preenchermos uma tabela e uma vez que estes erros sejam detectados, normalmente consertamos eles manualmente na tabela de dados brutos.
- No entanto, é bastante preferível que estes consertos sejam realizados através da própria linguagem de programação, a fim de que toda e qualquer alteração e editoração que você tenha feito a um conjunto de dados fique registrado e você não se esqueça no futuro.
- Outra vantagem disso é que qualquer alteração futura que precise ser feita será muito mais fácil, uma vez que apenas será necessário mudar uma única linha de comando - ao invés de repetir manualmente todas as etapas da manipulação, limpeza e processamento de dados.
- Este processo também é útil quando queremos criar novas variáveis baseado nos valores daquelas que já existem.

Substituição de valores

- Duas funções bastante úteis para modificar valores são `sub` e `gsub`.

```
sub(pattern = "é", replacement = "e", x = "América do Norte")
```

```
## [1] "America do Norte"
```

```
sub(pattern = "E", replacement = "e", x = "AmErica do NortE")
```

```
## [1] "America do NortE"
```

```
gsub(pattern = "E", replacement = "e", x = "AmErica do NortE")
```

```
## [1] "America do Norte"
```

Substituição de valores

- Estas funções também podem ser utilizadas para remover espaços e outros caracteres (como `.`, `,`, `-`, e etc).

```
gsub(pattern = "  ", replacement = " ", x = "Rio  de  Janeiro")
```

```
## [1] "Rio de Janeiro"
```

```
sub(pattern = "/", replacement = " ", x = "PPGE/UFRJ")
```

```
## [1] "PPGE UFRJ"
```

```
sub(pattern = "-", replacement = " ", x = c("Pé-de-moleque", "PPGE-UFRJ"))
```

```
## [1] "Pé de-moleque" "PPGE UFRJ"
```

Substituição de valores

- Também podemos empregar estas funções em vetores e colunas.
- No exemplo abaixo, utilizamos a função `unique` para termos uma noção de quais são os valores únicos que aparecem dentro da coluna **Chamada** do objeto **projetos** - você consegue encontrar algum erro em algum dos elementos?

```
unique(x = projetos$Chamada)
```

```
##  [1] "Universal - Faixa A"             "BJT"
##  [3] "Universal - Faixa C"             "Incubadoras"
##  [5] "PPBio - Rede Mata Atletica"      "Universal - Faixa B"
##  [7] "PVE"                             "Linha 1 - Faixa - B"
##  [9] "PPBio - Rede Amazonia Ocidental" "Linha 1: Universidades"
## [11] "Pesca"                           "PELD"
## [13] "PPBio - Rede Cerrado"            "PPBio - Rede Campos Sulinos"
## [15] "Alemanha DFG"                    "GEOMA - Rede GEOMA"
## [17] "Ilhas Oceanicas"                 "PV"
## [19] "Argentina - CNPq/CONICET"        "Belgica"
## [21] "CsF"
```

Exercício 2

1. Substitua o erro de digitação que você encontrou pela grafia certa da palavra.
2. Avalie se esta substiuição corrigiu esta entrada na coluna **Chamada** do objeto **projetos**.
3. Caso não tenha sido corrigida, o que você acha que aconteceu? Como podemos realizar essa correção?

Edição de valores em um vetor

- Valores e nomes de variáveis com espaço, muito longos, com caracteres especiais, em caixa alta e etc, podem causar erros durante a indexação e operação de algumas funções. Portanto, é sempre desejável que simplifiquemos estes nomes e tornemos eles consistentes, para evitar possíveis dores de cabeça. Para isso, quatro funções podem ser bastante úteis: `tolower`, `toupper`, `make.names` e `abbreviate`.

```
## vamos criar um vetor com as 10 primeiras publicacoes que aparecem na coluna Publicacao do objeto revistas
exemplo <- revistas$Publicacao[1:10]
exemplo
```

```
##  [1] "Acta Amazonica"
##  [2] "Acta Biologica Colombiana"
##  [3] "Acta Botanica Brasilica"
##  [4] "Acta Limnologica Brasiliensia"
##  [5] "Acta Oecologica"
##  [6] "Acta Scientiarum - Biological Sciences"
##  [7] "Acta Tropica"
##  [8] "African Journal of Agricultural Research"
```

Edição de valores em um vetor

```
# tolower faz com que todos os caracteres fiquem em caixa baixa
tolower(exemplo)
# toupper faz com que todos os caracteres fiquem em caixa alta
toupper(exemplo)
# make.names faz com que os nomes das colunas mudem para um formato mais amigável a um computador
make.names(exemplo)
# e, se você achar que os nomes estão muito longos, podemos usar a função abbreviate
abbreviate(exemplo)
```

Exercício 3

- Observe que os nomes das colunas do objeto **projetos** não são de todo consistentes. Você conseguiria modificar o nome dessas colunas, fazendo com que todos os caracteres ficassem em caixa baixa? Dica: utilize `names(projetos)` ou `colnames(projetos)` para visualizar o nome das colunas desse `data.frame`.

Combinando vetores

- Você também pode unir informações presentes em dois ou mais vetores (ou colunas) em um único elemento, utilizando as funções `paste` e `paste0`. Como exemplo, vamos unir a sigla da Unidade da Federação com o nome da Cidade que estão no objeto **projetos**.

```
## compare as formas abaixo
paste(projetos$cidade, projetos$uf)
paste(projetos$cidade, projetos$uf, sep = "/")
paste0(projetos$cidade, projetos$uf)
paste0(projetos$cidade, "/", projetos$uf)
```

Exercício 4

- De que forma podemos criar o `data.frame` abaixo? (apenas as primeiras linhas são apresentadas aqui por conta do tamanho)

```
##    id_coordenador        localidade
## 1               1     Rio Grande/RS
## 2               2     Sao Carlos/SP
## 3               3       Curitiba/PR
## 4               4         Recife/PE
## 5               5    Joao Pessoa/PB
## 6               6         Manaus/AM
## 7               7         Manaus/AM
## 8               8        Goiania/GO
## 9               9 Rio de Janeiro/RJ
## 10             10 Rio de Janeiro/RJ
## 11             11       Salvador/BA
## 12             12 Belo Horizonte/MG
```

Edição de vetores baseado em lógica

- Finalmente, também podemos editar e alterar os valores de uma coluna baseado em testes e argumentos lógicos.
- Uma função importante neste sentido é o `ifelse`.

```
## um vetor contendo o sexo de 10 pessoas
sexo <- rep(x = c("M", "F"), each = 5)
sexo
```

```
##  [1] "M" "M" "M" "M" "M" "F" "F" "F" "F" "F"
```

```
## substituindo
ifelse(test = sexo == "M", yes = "masculino", no = "feminino")
```

```
##  [1] "masculino" "masculino" "masculino" "masculino" "masculino"
##  [6] "feminino"  "feminino"  "feminino"  "feminino"  "feminino"
```

Exercício 5

- Adicione uma coluna ao objeto **revistas** que especifique se a revista em questão é nacional ou internacional.

Edição de vetores baseado em lógica

- Mas o que acontece se tivéssemos três categorias diferentes de sexo?

```
## um vetor contendo o sexo de 15 pessoas
sexo <- rep(x = c("M", "F", "ND"), each = 5)
sexo
```

```
##  [1] "M"  "M"  "M"  "M"  "M"  "F"  "F"  "F"  "F"  "F"  "ND" "ND" "ND" "ND"
## [15] "ND"
```

```
## substituindo
ifelse(test = sexo == "M", yes = "masculino", no = "feminino")
```

Edição de vetores baseado em lógica

- Cada elemento de um vetor que não passa no teste do `ifelse` recebe o valor que determinamos. Portanto, quando temos um vetor com múltiplos valores e queremos substituir apenas um deles devemos fazer essa operação em cadeia.

```
# se o resultado for verdadeiro substitua por 'masculino', caso contrario, substitua pelo valor de sexo naquela posicao
(sexo <- ifelse(test = sexo == "M", yes = "masculino", no = sexo))

# se o resultado for verdadeiro substitua por 'feminino', caso contrario, substitua pelo valor de sexo naquela posicao
(sexo <- ifelse(test = sexo == "F", yes = "feminino", no = sexo))

# se o resultado for verdadeiro substitua por 'feminino', caso contrario, substitua pelo valor de sexo naquela posicao
(sexo <- ifelse(test = sexo == "ND", yes = "não determinado", no = sexo))
```

Exercício 6

- Você consegue realizar a mesma operação que acabamos demonstrar sem criar um objeto a cada etapa?

```
## um vetor contendo o sexo de 15 pessoas
sexo <- rep(x = c("M", "F", "ND"), each = 5)
sexo
```

```
##  [1] "M"  "M"  "M"  "M"  "M"  "F"  "F"  "F"  "F"  "F"  "ND" "ND" "ND" "ND"
## [15] "ND"
```

Editando todo o conteúdo das colunas de uma tabela

- O `ifelse` também pode ser aplicado à toda a tabela de dados, o que pode nos ajudar a transformar uma matriz de abundância em uma de presenca/ausência.

```
ifelse(test = varespec > 0, yes = 1, no = 0)
```

```
##    Callvulg Empenigr Rhodtome Vaccmyrt Vaccviti Pinusylv Descflex Betupube
## 18        1        1        0        0        1        1        0        0
## 15        1        1        0        1        1        1        0        0
## 24        1        1        0        0        1        1        0        0
## 27        0        1        1        1        1        0        1        0
## 23        0        1        0        0        1        1        0        0
## 19        0        1        0        1        1        1        1        0
## 22        1        1        1        1        1        1        1        1
## 16        1        1        0        1        1        1        0        0
## 28        0        1        1        1        1        1        1        0
## 13        1        1        1        1        1        1        0        0
```

Exercício 7

- Uma tarefa muito comum para quem trabalha com ecologia de comunidades é quantificar a abundância total e a riqueza de espécies em uma dada comunidade. Nesse sentido, você poderia criar um `data.frame` com estas duas quantidades para cada comunidade apresentada no conjunto de dados `varespec`?
    - Dica: veja o arquivo de ajuda das funções `colSums` e `rowSums`.

Métodos orientados à tabela de dados {#anchor3}

- As funções apresentadas anteriormente são úteis para realizar algumas manipulações básicas e mais comuns do conteúdo de um tabela. No entanto, existe muitas outras coisas que precisamos fazer durante a manipulação, limpeza e processamento de dados, onde muitas delas são feitas de forma complexa ou pouco intuitiva através das funções da `base` do R.
- Algumas dessas tarefas são:
  - Renomear o nome de uma coluna específica (na `base`, indexação);
  - Ordenar uma tabela de acordo com uma ou mais colunas (na `base`, indexação + `sort` e/ou `order`);
  - Selecionar algumas colunas específicas ou mudar a ordem delas (na `base`, indexação ou escrever o nome de cada uma - entre aspas);
  - Filtrar uma tabela de dados de acordo com critérios lógicos (na `base`, indexação por lógica ou `subset`);
  - Adicionar uma nova coluna a um conjunto de dados (na `base`, indexação);
  - Realizar uma operação para cada nível de uma categoria que defina as observações (na `base`, loop ou funções loop (`by`, `apply`, `lapply`,...));
  - Separar as informações de uma coluna em múltiplas colunas (na `base`, `strsplit` + `unlist` + `rbind.data.frame`);
  - Transformar uma tabela de dados do formato largo para o formato longo e vice-versa.

Métodos orientados à tabela de dados

- Note que todas essas ações podem ser definidas por **verbos**, que foram implementas através dos pacotes `dplyr` e `tidyr` no `tidyverse`.
  - `rename`: renomeia as colunas da tabela;
  - `arrange`: ordena as linhas de uma tabela de acordo com uma condição;
  - `select`: seleciona uma ou mais colunas de acordo com seu nome ou com um padrão;
  - `filter`: filtra as linhas de acordo com uma ou mais condições;
  - `mutate`: adiciona novas variáveis à tabela;
  - `group_by`: agrupa observações antes de realizar opereações;
  - `summarise`: sumariza múltiplos valores para um único;
  - `separate`: separa uma coluna em múltiplas colunas;
  - `spread`: transforma uma tabela do **formato longo para o formato largo**;
  - `gather`: transforma uma tabela do **formato largo para o formato longo**.

Funcionamento geral do verbos do `tidyverse`

- Unidade primária de manipulação é o `data.frame` e ou `tibble`, e as colunas presentes neles;
- Todo o verbo é interpretado através de *lazy evaluation*: a primeira coisa que você fornece para a função é o conjunto de dados; a partir daí, a própria função entende que tudo o que você for fazer é em referência às informações que estão ali - elimina a necessidade de indexação.
- A idéia geral aqui é encurtar o espaço entre o que você quer fazer e o resultado (ou seja, entre a pergunta e a resposta), sem importar o tamanho do conjunto de dados.
- Para provar estes pontos, vamos utilizar o conjunto de dados `gapminder` que carregamos no início da aula.

gapminder

```
gapminder
```

```
## # A tibble: 1,704 x 6
##    country     continent  year lifeExp      pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.
##  2 Afghanistan Asia       1957    30.3  9240934      821.
##  3 Afghanistan Asia       1962    32.0 10267083      853.
##  4 Afghanistan Asia       1967    34.0 11537966      836.
##  5 Afghanistan Asia       1972    36.1 13079460      740.
##  6 Afghanistan Asia       1977    38.4 14880372      786.
##  7 Afghanistan Asia       1982    39.9 12881816      978.
##  8 Afghanistan Asia       1987    40.8 13867957      852.
##  9 Afghanistan Asia       1992    41.7 16317921      649.
## 10 Afghanistan Asia       1997    41.8 22227415      635.
## # ... with 1,694 more rows
```

rename

Utilizada para renomear colunas específicas, contornando a necessidade de indexação.

```
rename(.data = gapminder, pais = country, continente = continent, ano = year)
```

```
## # A tibble: 1,704 x 6
##    pais        continente   ano lifeExp      pop gdpPercap
##    <fct>       <fct>      <int>   <dbl>    <int>     <dbl>
##  1 Afghanistan Asia        1952    28.8  8425333      779.
##  2 Afghanistan Asia        1957    30.3  9240934      821.
##  3 Afghanistan Asia        1962    32.0 10267083      853.
##  4 Afghanistan Asia        1967    34.0 11537966      836.
##  5 Afghanistan Asia        1972    36.1 13079460      740.
##  6 Afghanistan Asia        1977    38.4 14880372      786.
##  7 Afghanistan Asia        1982    39.9 12881816      978.
##  8 Afghanistan Asia        1987    40.8 13867957      852.
##  9 Afghanistan Asia        1992    41.7 16317921      649.
## 10 Afghanistan Asia        1997    41.8 22227415      635.
## # ... with 1,694 more rows
```

arrange

Utilizada para ordernar as linhas de uma tabela em ordem crescente ou decrescente.

```
arrange(.data = gapminder, year)
```

```
## Warning: package 'bindrcpp' was built under R version 3.4.4
```

```
## # A tibble: 1,704 x 6
##    country     continent  year lifeExp      pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.
##  2 Albania     Europe     1952    55.2  1282697     1601.
##  3 Algeria     Africa     1952    43.1  9279525     2449.
##  4 Angola      Africa     1952    30.0  4232095     3521.
##  5 Argentina   Americas   1952    62.5 17876956     5911.
##  6 Australia   Oceania    1952    69.1  8691212    10040.
##  7 Austria     Europe     1952    66.8  6927772     6137.
##  8 Bahrain     Asia       1952    50.9   120447     9867.
##  9 Bangladesh  Asia       1952    37.5 46886859      684.
```

arrange

Utilizada para ordernar as linhas de uma tabela em ordem crescente ou decrescente.

```
arrange(.data = gapminder, desc(year))
```

```
## # A tibble: 1,704 x 6
##    country     continent  year lifeExp       pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>     <int>     <dbl>
##  1 Afghanistan Asia       2007    43.8  31889923      975.
##  2 Albania     Europe     2007    76.4   3600523     5937.
##  3 Algeria     Africa     2007    72.3  33333216     6223.
##  4 Angola      Africa     2007    42.7  12420476     4797.
##  5 Argentina   Americas   2007    75.3  40301927    12779.
##  6 Australia   Oceania    2007    81.2  20434176    34435.
##  7 Austria     Europe     2007    79.8   8199783    36126.
##  8 Bahrain     Asia       2007    75.6    708573    29796.
##  9 Bangladesh  Asia       2007    64.1 150448339     1391.
## 10 Belgium     Europe     2007    79.4  10392226    33693.
## # ... with 1,694 more rows
```

arrange

Pode comportar tantas colunas quantas aquelas que você desejar.

```
arrange(.data = gapminder, desc(year), lifeExp)
```

```
## # A tibble: 1,704 x 6
##    country                 continent  year lifeExp      pop gdpPercap
##    <fct>                   <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Swaziland               Africa     2007    39.6  1133066     4513.
##  2 Mozambique              Africa     2007    42.1 19951656      824.
##  3 Zambia                  Africa     2007    42.4 11746035     1271.
##  4 Sierra Leone            Africa     2007    42.6  6144562      863.
##  5 Lesotho                 Africa     2007    42.6  2012649     1569.
##  6 Angola                  Africa     2007    42.7 12420476     4797.
##  7 Zimbabwe                Africa     2007    43.5 12311143      470.
##  8 Afghanistan             Asia       2007    43.8 31889923      975.
##  9 Central African Republic Africa    2007    44.7  4369038      706.
## 10 Liberia                 Africa     2007    45.7  3193942      415.
## # ... with 1,694 more rows
```

Exercício 8

1. Qual foi o projeto que mais gastou recursos dentre aqueles financiados pelo CNPq?
2. Qual foi o coordenador que terminou o doutorado há mais tempo?
3. Qual o coordenador que tem o maior índice H e não fez pós-doutorado?

select

Utilizada para selecionar uma ou mais colunas de acordo com seu nome.

```
# somente o pais, ano e gdp per capita
select(.data = gapminder, country, year, gdpPercap)
```

```
## # A tibble: 1,704 x 3
##    country      year gdpPercap
##    <fct>       <int>     <dbl>
##  1 Afghanistan  1952      779.
##  2 Afghanistan  1957      821.
##  3 Afghanistan  1962      853.
##  4 Afghanistan  1967      836.
##  5 Afghanistan  1972      740.
##  6 Afghanistan  1977      786.
##  7 Afghanistan  1982      978.
##  8 Afghanistan  1987      852.
##  9 Afghanistan  1992      649.
## 10 Afghanistan  1997      635.
## # ... with 1,694 more rows
```

select

Utilizada para selecionar uma ou mais colunas de acordo com seu nome.

```
# apenas algumas variáveis e reordenando elas
select(.data = gapminder, gdpPercap, country:lifeExp)
```

```
## # A tibble: 1,704 x 5
##    gdpPercap country     continent  year lifeExp
##        <dbl> <fct>       <fct>     <int>   <dbl>
##  1      779. Afghanistan Asia       1952    28.8
##  2      821. Afghanistan Asia       1957    30.3
##  3      853. Afghanistan Asia       1962    32.0
##  4      836. Afghanistan Asia       1967    34.0
##  5      740. Afghanistan Asia       1972    36.1
##  6      786. Afghanistan Asia       1977    38.4
##  7      978. Afghanistan Asia       1982    39.9
##  8      852. Afghanistan Asia       1987    40.8
##  9      649. Afghanistan Asia       1992    41.7
## 10      635. Afghanistan Asia       1997    41.8
## # ... with 1,694 more rows
```

`select`

Também pode ser utilizada para selecionar colunas baseado em um padrão específico, utilizando os argumentos auxiliares:

- `starts_with()`: seleciona colunas que comecem com um certo padrão em seu nome;
- `ends_with()`: seleciona colunas que terminem com um certo padrão em seu nome;
- `contains()`: seleciona colunas que contenham um certo padrão em seu nome.

Exercício 9

- Do conjunto de dados do `gapminder`:
  - Remova apenas a coluna `country` de `gapminder`;
  - Selecione as colunas `continent`, `year` e `pop`;
  - Com o resultado da última operação, ordene as linhas de acordo com a ordem decrescente dos anos;
  - Além de ordenar as linhas pela ordem crescente dos continentes, ordene agora também o tamanho da população.
- Do conjunto de dados dos projetos, selecione a coluna com o ID do coordenador do projeto e todas as colunas que representam de recursos gastos em cada projeto.

filter

Utilizada para filtrar uma tabela de acordo com as condições que você determina.

```
# apenas os dados do Brasil
filter(.data = gapminder, country == "Brazil")
```

```
## # A tibble: 12 x 6
##    country continent  year lifeExp       pop gdpPercap
##    <fct>   <fct>     <int>   <dbl>     <int>     <dbl>
##  1 Brazil  Americas   1952    50.9  56602560     2109.
##  2 Brazil  Americas   1957    53.3  65551171     2487.
##  3 Brazil  Americas   1962    55.7  76039390     3337.
##  4 Brazil  Americas   1967    57.6  88049823     3430.
##  5 Brazil  Americas   1972    59.5 100840058     4986.
##  6 Brazil  Americas   1977    61.5 114313951     6660.
##  7 Brazil  Americas   1982    63.3 128962939     7031.
##  8 Brazil  Americas   1987    65.2 142938076     7807.
##  9 Brazil  Americas   1992    67.1 155975974     6950.
## 10 Brazil  Americas   1997    69.4 168546719     7958.
```

filter

Utilizada para filtrar uma tabela de acordo com as condições que você determina.

```
# dados das Americas, apenas os 20 últimos anos
filter(.data = gapminder, continent == "Americas", year > 1996)
```

```
## # A tibble: 75 x 6
##    country   continent  year lifeExp       pop gdpPercap
##    <fct>     <fct>     <int>   <dbl>     <int>     <dbl>
##  1 Argentina Americas   1997    73.3  36203463    10967.
##  2 Argentina Americas   2002    74.3  38331121     8798.
##  3 Argentina Americas   2007    75.3  40301927    12779.
##  4 Bolivia   Americas   1997    62.0   7693188     3326.
##  5 Bolivia   Americas   2002    63.9   8445134     3413.
##  6 Bolivia   Americas   2007    65.6   9119152     3822.
##  7 Brazil    Americas   1997    69.4 168546719     7958.
##  8 Brazil    Americas   2002    71.0 179914212     8131.
##  9 Brazil    Americas   2007    72.4 190010647     9066.
## 10 Canada    Americas   1997    78.6  30305843    28955.
```

filter

Utilizada para filtrar uma tabela de acordo com as condições que você determina.

```
# dados das Americas e Europa, apenas os 20 últimos anos
filter(.data = gapminder, continent == "Americas" | continent == "Europe", year > 1996)
```

```
## # A tibble: 165 x 6
##    country   continent  year lifeExp      pop gdpPercap
##    <fct>     <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Albania   Europe     1997    73.0  3428038     3193.
##  2 Albania   Europe     2002    75.7  3508512     4604.
##  3 Albania   Europe     2007    76.4  3600523     5937.
##  4 Argentina Americas   1997    73.3 36203463    10967.
##  5 Argentina Americas   2002    74.3 38331121     8798.
##  6 Argentina Americas   2007    75.3 40301927    12779.
##  7 Austria   Europe     1997    77.5  8069876    29096.
##  8 Austria   Europe     2002    79.0  8148312    32418.
##  9 Austria   Europe     2007    79.8  8199783    36126.
## 10 Belgium   Europe     1997    77.5 10199787    27561.
```

filter

Podemos utilizar o argumento lógico `%in%` para selecionar múltiplos elementos de uma mesma coluna.

```
# dados das Americas Europa e Oceania, apenas os 20 últimos anos
filter(.data = gapminder, continent %in% c("Americas", "Europe", "Oceania"), year > 1996)
```

```
## # A tibble: 171 x 6
##    country   continent  year lifeExp      pop gdpPercap
##    <fct>     <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Albania   Europe     1997    73.0  3428038     3193.
##  2 Albania   Europe     2002    75.7  3508512     4604.
##  3 Albania   Europe     2007    76.4  3600523     5937.
##  4 Argentina Americas   1997    73.3 36203463    10967.
##  5 Argentina Americas   2002    74.3 38331121     8798.
##  6 Argentina Americas   2007    75.3 40301927    12779.
##  7 Australia Oceania    1997    78.8 18565243    26998.
##  8 Australia Oceania    2002    80.4 19546792    30688.
##  9 Australia Oceania    2007    81.2 20434176    34435.
## 10 Austria   Europe     1997    77.5  8069876    29096.
```

filter

E podemos criar uma função para fazer o inverso do `%in%`!

```
## criando a função
`%nin%` <- Negate(f = `%in%`)

## removendo os dados das Americas, Europa e Oceania
filter(.data = gapminder, continent %nin% c("Americas", "Europe", "Oceania"), year > 1996)
```

```
## # A tibble: 255 x 6
##    country     continent  year lifeExp      pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Afghanistan Asia       1997    41.8 22227415      635.
##  2 Afghanistan Asia       2002    42.1 25268405      727.
##  3 Afghanistan Asia       2007    43.8 31889923      975.
##  4 Algeria     Africa     1997    69.2 29072015     4797.
##  5 Algeria     Africa     2002    71.0 31287142     5288.
##  6 Algeria     Africa     2007    72.3 33333216     6223.
##  7 Angola      Africa     1997    41.0  9875024     2277.
##  8 Angola      Africa     2002    41.0 10866106     2773.
```

Exercício 10

- Quais foram os projetos financiados no estado do Rio de Janeiro?
- Quais os projetos financiados no estado do Rio de Janeiro foram coordenados por mulheres?
- Onde estão localizados os bolsistas de produtividade 1C ou 1D que mais gastaram recursos?

mutate

Cria uma nova coluna na tabela de dados, inclusive usando as próprias colunas que estão sendo criadas dentro da função naquele momento.

```
mutate(.data = gapminder, log_gdp = log(gdpPercap), exp_gdp = exp(log_gdp))
```

```
## # A tibble: 1,704 x 8
##    country     continent  year lifeExp      pop gdpPercap log_gdp exp_gdp
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>   <dbl>   <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.    6.66    779.
##  2 Afghanistan Asia       1957    30.3  9240934      821.    6.71    821.
##  3 Afghanistan Asia       1962    32.0 10267083      853.    6.75    853.
##  4 Afghanistan Asia       1967    34.0 11537966      836.    6.73    836.
##  5 Afghanistan Asia       1972    36.1 13079460      740.    6.61    740.
##  6 Afghanistan Asia       1977    38.4 14880372      786.    6.67    786.
##  7 Afghanistan Asia       1982    39.9 12881816      978.    6.89    978.
##  8 Afghanistan Asia       1987    40.8 13867957      852.    6.75    852.
##  9 Afghanistan Asia       1992    41.7 16317921      649.    6.48    649.
## 10 Afghanistan Asia       1997    41.8 22227415      635.    6.45    635.
## # ... with 1,694 more rows
```

`transmute`

Similar ao `mutate`, mas elimina todas as outras colunas ao retornar o resultado

```
transmute(.data = gapminder, log_gdp = log(gdpPercap), exp_gdp = exp(log_gdp))
```

```
## # A tibble: 1,704 x 2
##    log_gdp exp_gdp
##      <dbl>   <dbl>
##  1    6.66    779.
##  2    6.71    821.
##  3    6.75    853.
##  4    6.73    836.
##  5    6.61    740.
##  6    6.67    786.
##  7    6.89    978.
##  8    6.75    852.
##  9    6.48    649.
## 10    6.45    635.
## # ... with 1,694 more rows
```

Exercício 11

- Crie uma nova coluna na tabelas **processos** que seja a combinação das colunas `cidade` e `uf`.

group_by

Agrupa as observações de acordo com os níveis de uma ou mais variáveis presentes nas colunas. É excelente para ser combinado com outras funções.

```
group_by(.data = gapminder, continent)
```

```
## # A tibble: 1,704 x 6
## # Groups:   continent [5]
##    country     continent  year lifeExp      pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.
##  2 Afghanistan Asia       1957    30.3  9240934      821.
##  3 Afghanistan Asia       1962    32.0 10267083      853.
##  4 Afghanistan Asia       1967    34.0 11537966      836.
##  5 Afghanistan Asia       1972    36.1 13079460      740.
##  6 Afghanistan Asia       1977    38.4 14880372      786.
##  7 Afghanistan Asia       1982    39.9 12881816      978.
##  8 Afghanistan Asia       1987    40.8 13867957      852.
##  9 Afghanistan Asia       1992    41.7 16317921      649.
## 10 Afghanistan Asia       1997    41.8 22227415      635.
```

group_by + filter

```
## quais sao os paises em cada continente que tiveram menor expectativa de vida em toda a serie
filter(.data = group_by(.data = gapminder, continent), lifeExp == min(lifeExp))
```

```
## # A tibble: 5 x 6
## # Groups:   continent [5]
##   country     continent  year lifeExp      pop gdpPercap
##   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
## 1 Afghanistan Asia       1952    28.8  8425333      779.
## 2 Australia   Oceania    1952    69.1  8691212    10040.
## 3 Haiti       Americas   1952    37.6  3201488     1840.
## 4 Rwanda      Africa     1992    23.6  7290203      737.
## 5 Turkey      Europe     1952    43.6 22235677     1969.
```

Exercício 12

- Quais são os coordenadores estrangeiros e brasileiros que tem indice H menor do que 15 e se terminaram o doutorado antes de 1990?
- Quais são os coordenadores que mais gastaram recursos por classe de bolsa de produtividade?

`group_by` + `summarise`

Uma das grandes vantagens do `group_by` é observada quando combinamos ele com a função `summarise`, que aplica uma mesma função para cada nível da variável agrupadora e retorna uma tabela com o sumário estatístico.

```
## expectativade vida media por continente
summarise(.data = group_by(.data = gapminder, continent), expectativa_media = mean(lifeExp))
```

```
## # A tibble: 5 x 2
##   continent expectativa_media
##   <fct>                 <dbl>
## 1 Africa                 48.9
## 2 Americas               64.7
## 3 Asia                   60.1
## 4 Europe                 71.9
## 5 Oceania                74.3
```

`group_by` + `summarise`

- Podemos agrupar os dados de acordo com várias colunas.
- Para quebrar o agrupamento basta utilizarmos a função `ungroup`.

```
## expectativamente de vida media por continente por ano
summarise(.data = group_by(.data = gapminder, continent, year), expectativa_media = mean(lifeExp))
```

```
## # A tibble: 60 x 3
## # Groups:   continent [?]
##    continent  year expectativa_media
##    <fct>     <int>             <dbl>
##  1 Africa     1952              39.1
##  2 Africa     1957              41.3
##  3 Africa     1962              43.3
##  4 Africa     1967              45.3
##  5 Africa     1972              47.5
##  6 Africa     1977              49.6
##  7 Africa     1982              51.6
##  8 Africa     1987              53.3
##  9 Africa     1992              53.6
```

Exercício 13

- Utilizando o objeto **revistas**, adicione uma coluna indicando se cada revista é nacional ou estrangeira e calcule o índice SJR médio destas duas categorias.
- Calcule o índice H médio de coordenadores de projetos brasileiros e extrangeiros de acordo com o tipo de bolsa de produtividade recebido, e considerando apenas os coordenadores que terminaram o doutorado após o ano 2000.

O operador pipe: `%>%`

- Em toda língua, todo texto fica difícil de compreender quando emendarmos frases sem adicionar uma pontuação.
- Acabamos de ver isso acontecendo também quando utilizamos a linguagem R, ao utilizarmos resultado de um verbo diretamente no processamento de outro verbo do `tidyverse`.
- Isso faz com que todo o código que escrevamos rapidamente fique complexo demais de se ler ou, ainda, exija a criação de diversas etapas intermediárias.

```
## muito complexo
summarise(.data = group_by(.data = select(.data = gapminder, country, continent, gdpPercap), continent, country),
          media_gdp = mean(gdpPercap))

## muito enfadonho
passo1 <- select(.data = gapminder, country, continent, gdpPercap)
passo2 <- group_by(.data = passo1, continent, country)
passo3 <- summarise(.data = passo2, media_gdp = mean(gdpPercap))
```

O operador pipe: `%>%`

- A fim de descomplicar a escrita do código e fazer com que ele fique mais claro, o operador `pipe` é implementado no ambiente de trabalho quando você carrega o `tidyverse`.
- O operador `pipe`, representado pelo símbolo `%>%`, é implementado especificamente através do pacote `dplyr`.
- O atalho do teclado para o `pipe` é **Control + Shift + M** (no Windows) ou **Command + Shift + M** (no MAC).
- Um exemplo do uso do `pipe` no mesmo contexto apresentado no slide anterior:

```
gapminder %>%
  select(country, continent, gdpPercap) %>%
  group_by(continent, country) %>%
  summarise(media_gdp = mean(gdpPercap))
```

O operador pipe: `%>%`

- O `pipe` funciona potencializando o *lazy evaluation* nos verbos do `tidyverse`.
- Ele passa o `data.frame`/`tibble` de uma linha de comando ou resultante do processamento de um verbo para o argumento `.data` do verbo que o segue - assim como a pontuação e adjuntos conectam frases no português.

```
gapminder %>%
  select(country, continent, gdpPercap) %>%
  group_by(continent, country) %>%
  summarise(media_gdp = mean(gdpPercap))
```

O operador pipe: `%>%`

- Como veremos nas outras aulas, o `pipe` também pode ser empregado no processamento de outras funções, incluindo a extração e processamento de resultados de análises.

```
gapminder %>%
  group_by(country) %>%
  summarise(expectativa = mean(lifeExp), gdp = mean(gdpPercap)) %>%
  lm(gdp ~ expectativa, data = .) %>%
  summary(.)
```

Exercício 14

- Determine o número médio, mínimo e máximo do número de citações recebidas em cada uma das revistas onde os artigos científicos foram publicados (dados presentes no objeto __publicacoes).
- Repita o procedimento acima, mas calcule também o desvio padrão e o número total de artigos publicados em cada revista (utilize a função `n()` para tal).

drop_na

- Uma das formas de remover as linhas contendo `NA` na `base` do R é através da indexação por lógica, utilizando o `is.na`.
- Você pode fazer a mesma coisa no `tidyverse`, utilizando a função `drop_na` - especificando inclusive de qual coluna você quer que os `NA` sejam removidas.

```
drop_na(data = publicacoes, citacoes)
```

```
## # A tibble: 548 x 6
##       Processo    id titulo                    journal      ISSN  citacoes
##          <dbl> <dbl> <chr>                     <chr>        <chr>    <dbl>
##  1 40060020137     1 Larval Biology of Anthop… Journal of … 1536…        4
##  2 40060020137     2 Seasonal variation in di… Biotropica   0106…        0
##  3 40060020137     3 Biology of the immature … Revista Bra… 0085…        3
##  4 40060020137     4 Immature Stages and Ecol… Journal of … 0024…        0
##  5 40060020137     5 Importance of Habitat He… Environment… 0046…        2
##  6 40060020137     6 Sexual Dimorphism and Al… Journal of … 1536…        0
##  7 40060020137     7 Species composition and … Zoologia (C… 1984…        3
##  8 40060020137     8 Temporal Dynamics of Fru… Florida Ent… 0015…        0
```

separate

Utilizado para separar as informações de uma coluna em várias colunas diferentes.

```
projetos %>%
  separate(col = inicio, into = c("dia", "mes", "ano"), sep = "/")
```

```
## # A tibble: 119 x 28
##     processo chamada        id_coordenador sexo  dia   mes   ano   termino
##        <dbl> <chr>                   <int> <chr> <chr> <chr> <chr> <chr>
##  1   4.86e10 Universal - F…              1 M     06    11    12    05/11/…
##  2   4.01e10 BJT                         2 F     12    09    12    11/09/…
##  3   4.82e10 Universal - F…              3 M     08    11    13    30/11/…
##  4   4.72e10 Universal - F…              4 F     14    12    12    31/12/…
##  5   4.76e10 Universal - F…              5 M     30    10    13    31/10/…
##  6   4.82e10 Universal - F…              6 M     07    05    13    05/05/…
##  7   4.80e10 Universal - F…              7 M     19    11    12    18/11/…
##  8   4.46e10 Incubadoras                 8 M     02    08    16    31/12/…
##  9   4.58e10 PPBio - Rede …              9 M     12    12    12    11/12/…
## 10   4.72e10 Universal - F…             10 M     28    11    12    30/11/…
```

`rownames_to_column`

Por padrão, um `tibble` não comporta nomes nas linhas, o que pode ser particularmente problemático quando convertemos um `data.frame` para àquela classe de objeto. No entanto, podemos usar a função `rownames_to_column` para adicionar uma coluna que contenha o nome de cada linha.

```
varespec <- varespec %>%
  rownames_to_column(var = "site")
varespec
```

```
##    site Callvulg Empenigr Rhodtome Vaccmyrt Vaccviti Pinusylv Descflex
## 1    18     0.55    11.13     0.00     0.00    17.80     0.07     0.00
## 2    15     0.67     0.17     0.00     0.35    12.13     0.12     0.00
## 3    24     0.10     1.55     0.00     0.00    13.47     0.25     0.00
## 4    27     0.00    15.13     2.42     5.92    15.97     0.00     3.70
## 5    23     0.00    12.68     0.00     0.00    23.73     0.03     0.00
## 6    19     0.00     8.92     0.00     2.42    10.28     0.12     0.02
## 7    22     4.73     5.12     1.55     6.05    12.40     0.10     0.78
## 8    16     4.47     7.33     0.00     2.15     4.33     0.10     0.00
```

De largo para longo, e de volta outra vez

- Uma tarefa que normalmente precisamos fazer é também converter uma tabela do formato largo para o formato longo e vice-versa.
- Além disso, as vezes é mais fácil converter uma tabela para um desses formatos para realizar rapidamente um processamento ou manipulação de dados (e.g., aplicar uma mesma transformação apenas às colunas que contenham números).
- Existem duas funções que podem nos ajudar nesse sentido:
  - `gather`, para juntar as informações de múltiplas colunas em uma única coluna;
  - `spread`, para espalhar as informações de uma única coluna para múltiplas colunas.

gather

```
formato_longo <- gather(data = varespec, key = "especie", value = "densidade", Callvulg:Cladphyl)
formato_longo
```

```
##      site  especie densidade
## 1      18 Callvulg      0.55
## 2      15 Callvulg      0.67
## 3      24 Callvulg      0.10
## 4      27 Callvulg      0.00
## 5      23 Callvulg      0.00
## 6      19 Callvulg      0.00
## 7      22 Callvulg      4.73
## 8      16 Callvulg      4.47
## 9      28 Callvulg      0.00
## 10     13 Callvulg     24.13
## 11     14 Callvulg      3.75
## 12     20 Callvulg      0.02
## 13     25 Callvulg      0.00
## 14      7 Callvulg      0.00
## 15      5 Callvulg      0.00
```

spread

```
formato_largo <- spread(data = formato_longo, key = especie, value = densidade, fill = 0)
formato_largo
```

```
## # A tibble: 24 x 45
##    site  Barbhatc Betupube Callvulg Cetreric Cetrisla Cladamau Cladarbu
##    <chr>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
##  1 10        0           0     0.25     0.25     0.25     0         1.3
##  2 11        0           0     2.37     0        0.25     0         9.67
##  3 12        0           0     0.25     0        0.25     0         3.6
##  4 13        0.07        0    24.1      0.18     0.02     0        23.1
##  5 14        0           0     3.75     0.68     0.02     0        17.4
##  6 15        0           0     0.67     0.15     0.03     0        12.0
##  7 16        0           0     4.47     0.18     0.08     0         7.13
##  8 18        0           0     0.55     0.02     0        0.08    21.7
##  9 19        0.02        0     0        0        0        0         7.23
## 10 2         0           0     0.05     0        0        0         0.48
## # ... with 14 more rows, and 37 more variables: Cladbotr <dbl>,
## #   Cladcerv <dbl>, Cladchlo <dbl>, Cladcocc <dbl>, Cladcorn <dbl>,
## #   Cladcris <dbl>, Claddefo <dbl>, Cladfimb <dbl>, Cladgrac <dbl>,
```

Exercício 15

- A partir do objeto **formato_longo**, calcule a riqueza de espécies e a abundância total de espécies em cada site.