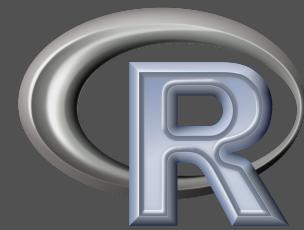




ME115 - Linguagem R

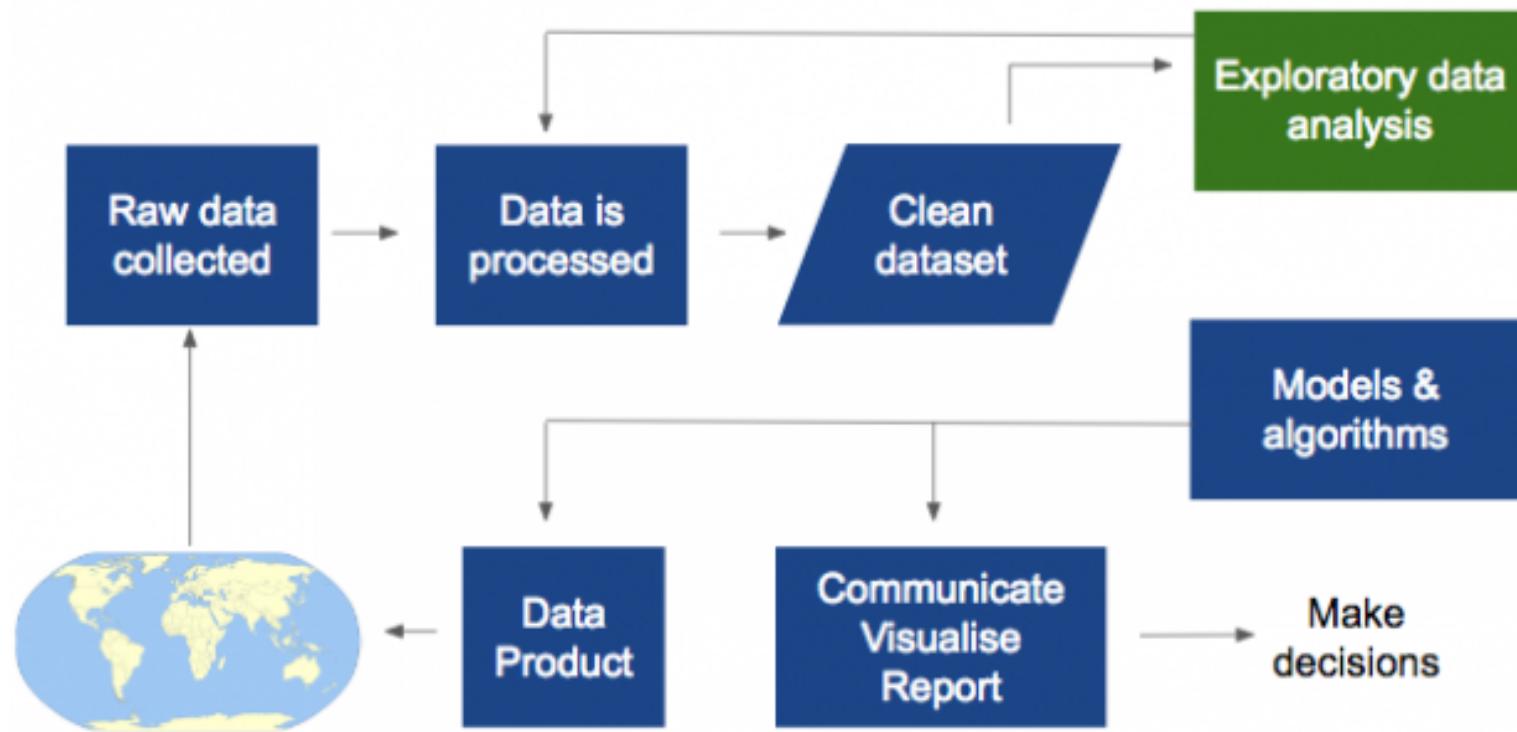
Parte 07

1º semestre de 2023



Limpeza, Manipulação e Transformação de Dados no R

Data Science Process



Por que se preocupar com a limpeza dos dados?

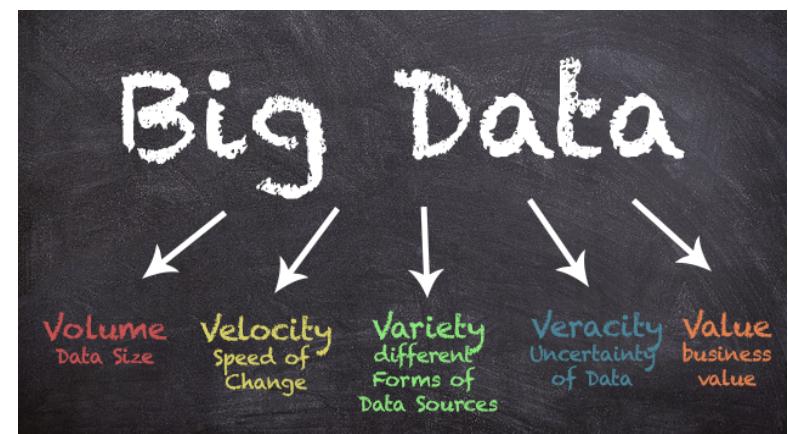
No mundo real, os dados não chegam até o cientista de dados prontos para análise, como vocês vêem em exemplos dos livros.

Na era do Big Data, limpeza de dados é um processo fundamental!!!

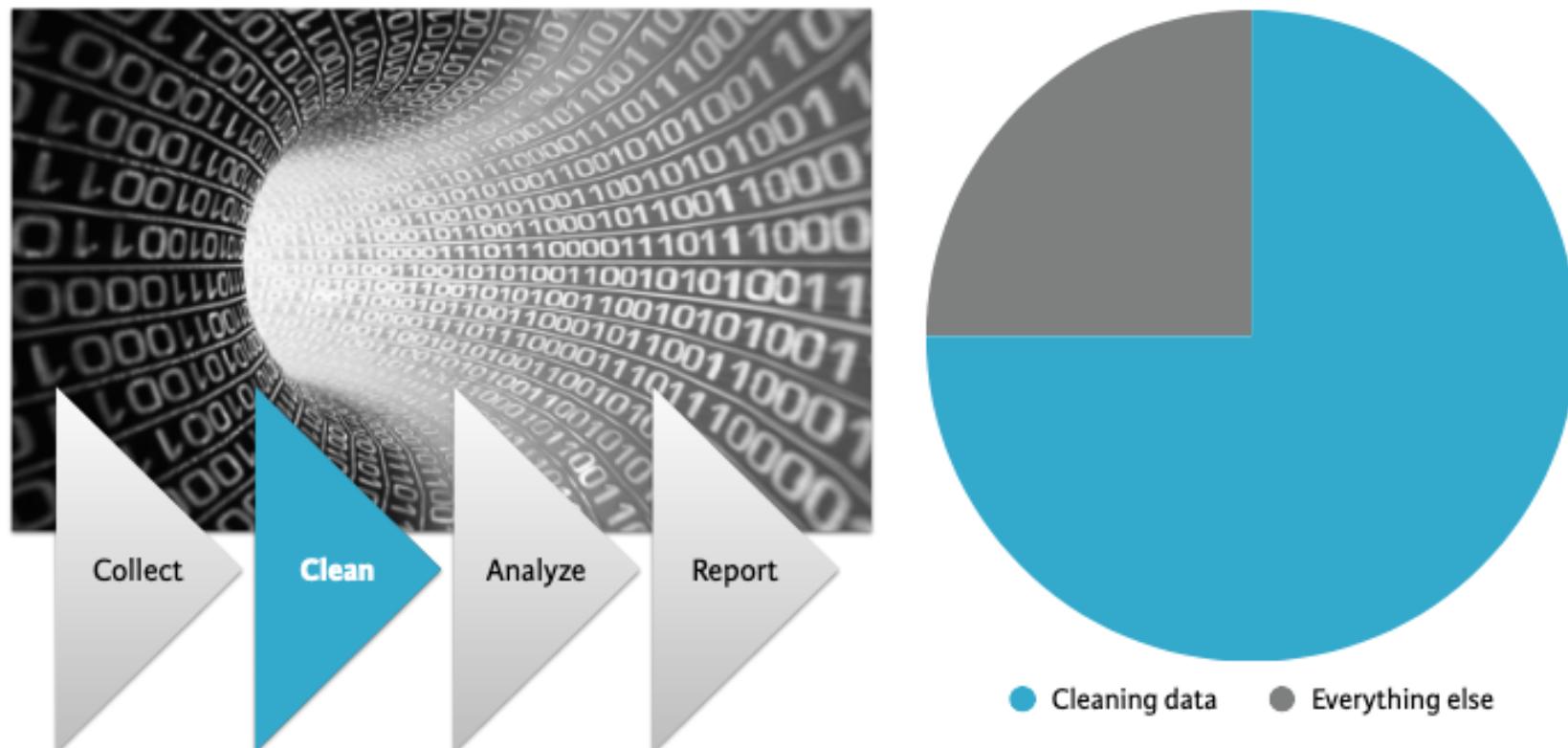
Quantidades gigantescas de dados nas mais diversas áreas: Indústria, Finanças, Saúde, Educação, etc.

Quanto maior o banco de dados, maior é a chance de que algo esteja errado também!

Imperfeições tornam-se difíceis de serem encontradas quando você não consegue visualizar seus dados em uma planilha de Excel.



Limpeza dos dados



Limpeza dos dados

For Big-Data Scientists, ‘Janitor Work’ Is Key Hurdle to Insights



Monica Rogati, Jawbone's vice president for data science, with Brian Wilt, a senior data scientist.
Peter DaSilva for The New York Times

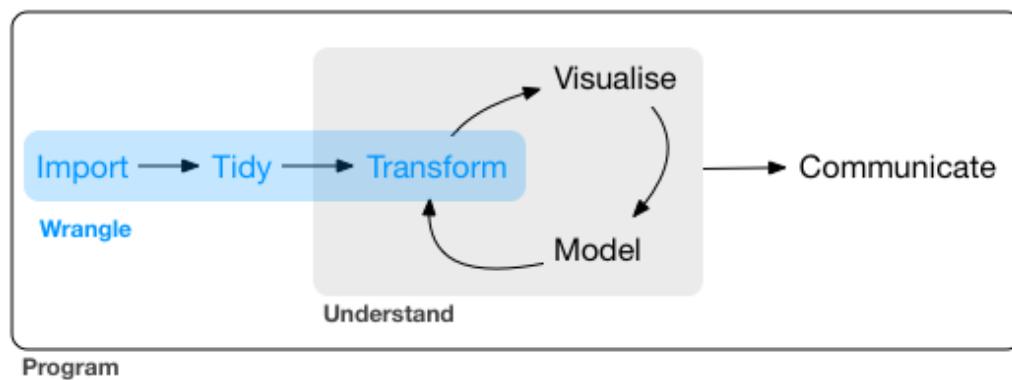
Fonte: [The New York Times, Aug. 17, 2014](#)

Data Wrangling

Data Wrangling: termo bastante utilizado atualmente que significa preparação dos dados.

Diz respeito ao ato de coletar, limpar, normalizar, combinar, estruturar e organizar os dados que serão analisados.

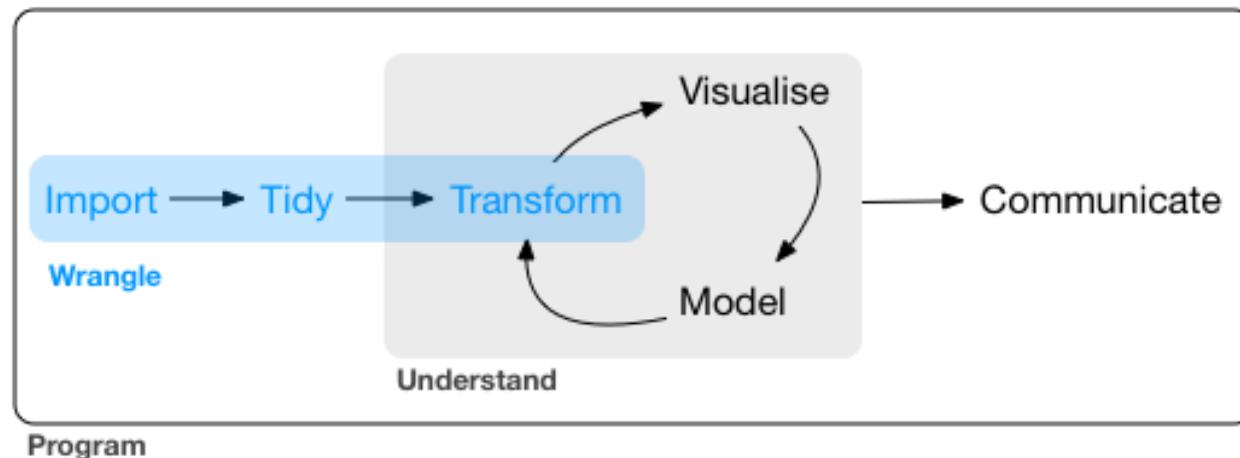
Existem três partes principais, ilustradas na figura abaixo:



Apesar de ser uma tarefa que toma bastante tempo, ela é indispensável, pois permite que você consiga trabalhar com os dados.

"Data Wrangling: the art of getting your data into R in a useful form for visualisation and modelling." [R for Data Science](#)

Preparação dos Dados



Importação: já vimos essa parte em aulas anteriores.

Tidy data: um método consistente de armazenar os dados que tornam transformação, visualização e modelagem mais fácil.

Manipulação/Transformação: aprenderemos várias ferramentas para nos auxiliar nesse processo.

Formato *Tidy*

- Observações são linhas
- Variáveis (atributos) são colunas
- Um tipo de unidade observacional por tabela

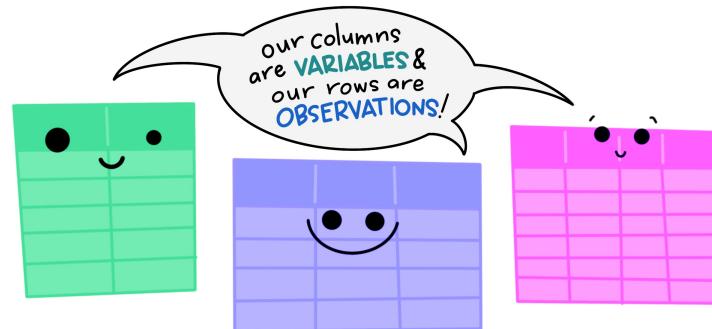
name	age	eye_color	height	Observation
Jake	34	Other	6'1"	
Alice	55	Blue	5'9"	
Tim	76	Brown	5'7"	
Denise	19	Other	5'1"	

Variable or Attribute

Leitura: Hadley Wickham. [Tidy data](#). *The Journal of Statistical Software*, vol. 59, 2014.

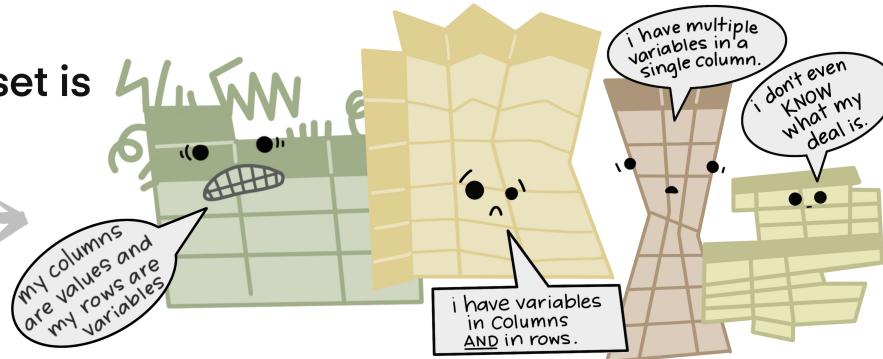
Formato *Tidy*

The standard structure of
tidy data means that
“tidy datasets are all alike...”



“...but every messy dataset is
messy in its own way.”

—HADLEY WICKHAM



Fonte: <https://cfss.uchicago.edu/notes/tidy-data/>

Formato *Tidy*

- Pode não ser o formato mais compacto, mas é o mais versátil e o melhor para análises estatísticas;
- *Tidy data* funciona bem no R, pois é uma linguagem de programação vetorizada: estruturas de dados são construídas de vetores e operações são otimizadas para vetores;
- Converter seus dados para esse formato pode dar um certo trabalho, mas o tempo gasto aqui é compensado no restante da análise.

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

table1

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

variables

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

observations

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

values

Fonte:

<https://garrettgman.github.io/tidying/>

Example de Dados *Tidy*

Nesse exemplo, temos os dados no formato *tidy*?

name	age	brown	blue	other	height
Jake	34	0	0	1	6'1"
Alice	55	0	1	0	5'9"
Tim	76	1	0	0	5'7"
Denise	19	0	0	1	5'1"



- Algumas colunas são valores e não nomes de variáveis!!!

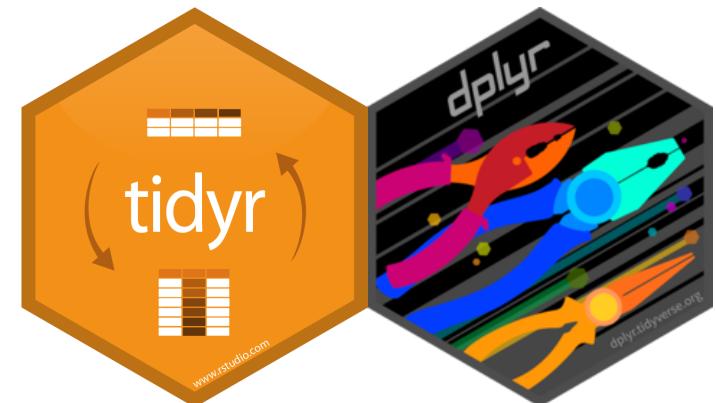
Ferramentas para Data Wrangling

Hadley Wickham e seu time no RStudio está na liderança do desenvolvimento de pacotes que funcionam de maneira bem eficiente e conversam bem entre si.

Dois pacotes muito utilizados são o `tidyverse` e `dplyr`.

Muitas tarefas de processamento de dados estão sendo colocadas em pacotes de maneiras mais consistentes, o que gera:

- códigos mais eficientes
- sintaxes mais fáceis de serem lembradas
- sintaxes mais fáceis de serem lidas



Tibbles

Tibble é um tipo especial de *data frame*. Podemos dizer que é que um *data frame* melhorado.

Como o pacote `tibble`, você pode criar seu próprio *tibble* usando:

```
library(tibble)  
tibble(x = 1:5, y = 1, z = x^2 + y)
```

```
## # A tibble: 5 × 3  
##       x     y     z  
##   <int> <dbl> <dbl>  
## 1     1     1     2  
## 2     2     1     5  
## 3     3     1    10  
## 4     4     1    17  
## 5     5     1    26
```



Tibbles

Você pode criar um *tibble* a partir de um objeto (*data frames*, listas, matrizes ou tabelas) já existente usando a função `as_tibble()`:

```
as_tibble(iris)
```

```
## # A tibble: 150 × 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##       <dbl>      <dbl>      <dbl>      <dbl> <fct>
## 1         5.1        3.5        1.4        0.2 setosa
## 2         4.9        3.0        1.4        0.2 setosa
## 3         4.7        3.2        1.3        0.2 setosa
## 4         4.6        3.1        1.5        0.2 setosa
## 5         5.0        3.6        1.4        0.2 setosa
## 6         5.4        3.9        1.7        0.4 setosa
## 7         4.6        3.4        1.4        0.3 setosa
## 8         5.0        3.4        1.5        0.2 setosa
## 9         4.4        2.9        1.4        0.2 setosa
## 10        4.9        3.1        1.5        0.1 setosa
## # i 140 more rows
```



Tibbles

Essa classe tem sido muito utilizada e ela é melhor que um objeto da classe `data.frame` por alguns motivos:

- **Subconjuntos:** “[” sempre retorna um novo tibble, enquanto que “[[” e “\$” sempre retornam um vetor.
- **Subconjuntos de *tibbles* são *tibbles*:** isso pode não acontecer com um data frame.
- **Nomes das colunas:** trabalhar sempre com o nome completo das colunas e não apenas parte do nome.
- **Visualização:** quando você imprime um *tibble* no R, você visualiza uma versão concisa dos dados que se ajusta à sua tela.
- Nunca mudam os tipos das entradas, ou seja, não convertem strings para fatores, nunca mudam os nomes das variáveis e não usam nomes das linhas.

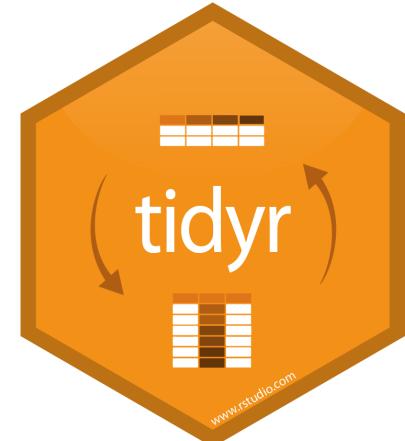


Introdução ao `tidyverse`

O pacote `tidyverse`, parte do `tidyverse`, foi criado especificamente para criar/transformar dados para o formato *tidy*.

Principais funções do pacote `tidyverse`:

- `pivot_longer()`: transforma formato *wide* no formato *tidy* (longo). Chamada anteriormente de `gather()`.
- `pivot_wider()`: transforma formato *longo* no formato *wide*. Chamada anteriormente de `spread()`.
- `separate()`: separa uma coluna em múltiplas colunas.
- `separate_rows()`: separa uma linha em múltiplas linhas.
- `unite()`: junta várias colunas em uma única coluna.



Formato *wide* para longo (*tidy*)

`pivot_longer()`: converte dados no formato *wide* para o formato *tidy* (longo).

table4a

country	1999	2000
A	0.7K	2K
B	37K	80K
C	212K	213K



country	year	cases
A	1999	0.7K
B	1999	37K
C	1999	212K
A	2000	2K
B	2000	80K
C	2000	213K

```
pivot_longer(table4a, cols = 2:3, names_to = "year", values_to = "cases")
```

Usando a função mais antiga `gather()`:

```
gather(table4a, key = "year", values = "cases", 2:3)
```

Formato longo para *wide* (*tidy*)

`pivot_wider()`: converte dados no formato longo para o formato *wide*.

table2

country	year	type	count
A	1999	cases	0.7K
A	1999		19M
A	2000	cases	2K
A	2000		20M
B	1999	cases	37K
B	1999		172M
B	2000	cases	80K
B	2000		174M
C	1999	cases	212K
C	1999		1T

→

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172M
B	2000	80K	174M
C	1999	212K	1T
C	2000	NA	NA

```
pivot_wider(table2, names_from = type, values_from = count)
```

Usando a função mais antiga `spread()`:

```
spread(table2, key = "type", value = "count")
```

Separar uma coluna em várias

`separate()`: separa os dados de uma coluna para múltiplas colunas, especificando o separador.

table3

The diagram illustrates the transformation of a data frame named 'table3'. On the left, the original data is shown in a wide format with three columns: 'country', 'year', and 'rate'. The 'rate' column contains values like '0.7K/19M' and '213K/1T', which are separated by a slash. An arrow points from this table to the right, indicating the transformation process. On the right, the resulting data is shown in a tidy format with four columns: 'country', 'year', 'cases', and 'pop'. The 'cases' column contains values like '0.7K' and '213K', and the 'pop' column contains values like '19M' and '1T'. This transformation separates the original 'rate' column into two distinct variables.

country	year	rate
A	1999	0.7K/19M
A	2000	2K/20M
B	1999	37K/172M
B	2000	80K/174M
C	1999	212K/1T
C	2000	213K/1T

→

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172
B	2000	80K	174
C	1999	212K	1T
C	2000	213K	1T

```
separate(table3, rate, sep = "/", into = c("cases", "pop"))
```

Vejam que o data frame resultante está no formato *tidy*.

Separar uma linha em várias

`separate_rows()`: separa os dados de uma para múltiplas linhas, especificando o separador.

table3

country	year	rate
A	1999	0.7K/19M
A	2000	2K/20M
B	1999	37K/172M
B	2000	80K/174M
C	1999	212K/1T
C	2000	213K/1T



country	year	rate
A	1999	0.7K
A	1999	19M
A	2000	2K
A	2000	20M
B	1999	37K
B	1999	172M
B	2000	80K
B	2000	174M
C	1999	212K
C	1999	1T
C	2000	213K
C	2000	1T

Vejam que o data frame resultante não está no formato *tidy*. Por que?

```
separate_rows(table3, rate, sep = "/")
```

Combinando várias colunas em uma só

`unite()`: combinar os valores de várias colunas, formando uma única coluna.

table5

country	century	year		country	year
Afghan	19	99	→	Afghan	1999
Afghan	20	00		Afghan	2000
Brazil	19	99		Brazil	1999
Brazil	20	00		Brazil	2000
China	19	99		China	1999
China	20	00		China	2000

```
unite(table5, century, year, col = "year", sep = "")
```

Outros pacotes e funções

Outros pacotes que também podem ser utilizados para mudar o formato dos dados são: `reshape2`, `spreadsheets` e `databases`.

tidyverse 1.0.0	pivot longer	pivot wider
tidyverse < 1.0.0	gather	spread
reshape(2)	melt	cast
spreadsheets	unpivot	pivot
databases	fold	unfold

Exemplo

Em um experimento, foram dados dois medicamentos diferentes a três pessoas e registraram seus batimentos cardíacos:

```
library(tibble)
df_wide <- tibble(name = c("Wilbur", "Petunia", "Gregory"),
                    a = c(67, 80, 64),
                    b = c(56, 90, 50))
df_wide
```

```
## # A tibble: 3 × 3
##   name     a     b
##   <chr> <dbl> <dbl>
## 1 Wilbur    67    56
## 2 Petunia   80    90
## 3 Gregory   64    50
```

Fonte: <https://blog.rstudio.com/2014/07/22/introducing-tidyr/>

Exemplo (cont.)

Usar a função `pivot_longer()` para colocar as colunas `a` e `b` em uma única coluna que representa o tipo de droga:

```
df_tidy <- pivot_longer(df_wide, cols = a:b, names_to = "drug", values_to = "heartrate")
```

```
## # A tibble: 6 × 3
##   name   drug  heartrate
##   <chr>  <chr>    <dbl>
## 1 Wilbur  a        67
## 2 Wilbur  b        56
## 3 Petunia a        80
## 4 Petunia b        90
## 5 Gregory a        64
## 6 Gregory b        50
```

O resultado seria equivalente se usássemos
`gather(df_wide, drug, heartrate, a:b)`

Exemplo (cont.)

Se quisermos transformar o formato longo (*tidy*) de volta em *wide*, usamos a função `pivot_wider()`:

```
pivot_wider(df_tidy, names_from = drug, values_from = heartrate)
```

```
## # A tibble: 3 × 3
##   name      a     b
##   <chr>  <dbl> <dbl>
## 1 Wilbur    67    56
## 2 Petunia   80    90
## 3 Gregory   64    50
```

E você obtém o mesmo resultado usando a função `spread()`:

```
spread(df_tidy, drug, heartrate)
```

Quanto tempo você fica ao telefone?

Nesse exemplo, temos algumas medidas de quanto tempo as pessoas ficam no telefone, tanto em casa quanto no trabalho, em duas ocasiões diferentes.

Cada pessoa foi aleatoriamente assinalada ao **Tratamento** ou **Controle**.

Os dados foram tabulados em um *data frame* diretamente no R e armazenados em um objeto chamado `messy`:

```
set.seed(10)
messy <- data.frame(id = 1:4,
                      trt = sample(rep(c('control', 'treatment'), each = 2)),
                      work.T1 = runif(4),
                      home.T1 = runif(4),
                      work.T2 = runif(4),
                      home.T2 = runif(4))
```

Quanto tempo você fica ao telefone?

O que precisamos fazer para deixar esses dados no formato *tidy*?

messy

```
##   id      trt work.T1 home.T1 work.T2 home.T2
## 1  1 treatment  0.0851   0.616   0.114  0.0519
## 2  2 control    0.2254   0.430   0.596  0.2642
## 3  3 control    0.2745   0.652   0.358  0.3988
## 4  4 treatment   0.2723   0.568   0.429  0.8361
```

1. Agrupar as colunas `work.T1`, `home.T1`, `work.T2` and `home.T2` em uma única coluna, que chamaremos de `local`, e os valores em uma coluna chamada `time`.
2. Separar a coluna `local` em duas colunas que chamaremos de `location` e `occasion`.

Quanto tempo você fica ao telefone?

Agrupar as colunas `work.T1`, `home.T1`, `work.T2` and `home.T2` em uma única coluna, que chamaremos de `local`, e os valores em uma coluna chamada `time`.

```
tidier <- gather(messy, local, time, -id, -trt)
head(tidier, 5)
```

```
##   id      trt    local    time
## 1 1 treatment work.T1 0.0851
## 2 2 control   work.T1 0.2254
## 3 3 control   work.T1 0.2745
## 4 4 treatment work.T1 0.2723
## 5 1 treatment home.T1 0.6158
```

Usando a função mais atual `pivot_longer()`:

```
tidier <- pivot_longer(messy, work.T1:home.T2, names_to = "local", values_to = "time")
```

Quanto tempo você fica ao telefone?

Separar a coluna `local` em duas colunas que chamaremos de `location` e `occasion`.

```
tidy <- separate(tidier, local, into = c("location", "occasion"), sep = "\\.")  
head(tidy, 8)
```

```
##   id      trt location occasion    time  
## 1  1 treatment    work       T1 0.0851  
## 2  2   control    work       T1 0.2254  
## 3  3   control    work       T1 0.2745  
## 4  4 treatment    work       T1 0.2723  
## 5  1 treatment   home       T1 0.6158  
## 6  2   control   home       T1 0.4297  
## 7  3   control   home       T1 0.6517  
## 8  4 treatment   home       T1 0.5677
```

Quanto tempo você fica ao telefone?

A função complementar de `separate()` é a `unite()`. Portanto, se quisermos juntar as colunas `location` e `occasion` em um única coluna chamada de `local`, basta fazer:

```
tidier <- unite(tidy, "local", c("location", "occasion"))  
head(tidier, 5)
```

```
##   id      trt    local    time  
## 1 1 treatment work_T1 0.0851  
## 2 2 control  work_T1 0.2254  
## 3 3 control  work_T1 0.2745  
## 4 4 treatment work_T1 0.2723  
## 5 1 treatment home_T1 0.6158
```

Veja que o separador usado é por default “_”, mas pode ser usado o que você quiser, especificando-o no argumento `sep`.

Exemplo `iris`

Voltando ao exemplo dos dados `iris` disponível no R.

```
head(iris, 8)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1         5.1       3.5        1.4       0.2  setosa
## 2         4.9       3.0        1.4       0.2  setosa
## 3         4.7       3.2        1.3       0.2  setosa
## 4         4.6       3.1        1.5       0.2  setosa
## 5         5.0       3.6        1.4       0.2  setosa
## 6         5.4       3.9        1.7       0.4  setosa
## 7         4.6       3.4        1.4       0.3  setosa
## 8         5.0       3.4        1.5       0.2  setosa
```

Esse data frame está no formato *tidy*? Por que?

Exemplo `iris`

Os dados `iris` não estão no formato *tidy*.

As colunas `Sepal.Length`, `Sepal.Width`, `Petal.Length` e `Petal.Width` são valores e não apenas nomes de variáveis.

Para ter o data frame `iris` no formato *tidy*:

1. Usar a função `pivot_longer()` para agrupar as colunas 1 a 4 em uma única coluna, com os valores em uma coluna chamada `Medida`;
2. Usar a função `separate()` para separar a parte da planta e a dimensão em duas colunas.

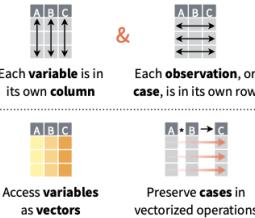
Faremos esse exemplo na aula prática de 5a-feira!

Tidyr Cheat Sheet

Data tidying with tidyr :: CHEAT SHEET

Tidy data is a way to organize tabular data in a consistent data structure across packages.

A table is tidy if:



Tibbles

AN ENHANCED DATA FRAME

Tibbles are a table format provided by the `tibble` package. They inherit the data frame class, but have improved behaviors:

- `Subset` a new tibble with `[]`, a vector with `[[]]` and `$`.
- `No partial matching` when subsetting columns.
- `Display` concise views of the data on one screen.

`options(tibble.print_max = n, tibble.print_min = m, tibble.width = Inf)` Control default display settings.

`View()` or `glimpse()` View the entire data set.

CONSTRUCT A TIBBLE

`tibble(...)` Construct by columns.

`tibble(x = 1:3, y = c("a", "b", "c"))`

`tibble(...)` Construct by rows.

`tibble(x ~ y,`

```
  1, "a";
  2, "b";
  3, "c")
```

Both make this tibble

`as_tibble(x, ...)` Convert a data frame to a tibble.

`enframe(x, name = "name", value = "value")`

Convert a named vector to a tibble. Also `deframe()`.

`is_tibble(x)` Test whether x is a tibble.



Reshape Data - Pivot data to reorganize values into a new layout.

table4a

country	1999	2000
A	0.7K	2K
B	37K	80K
C	212K	213K

table4b

country	year	cases
A	1999	0.7K
B	2000	37K
C	1999	212K
A	2000	2K
B	2000	80K
C	2000	213K

table2

country	year	type	count
A	1999	cases	0.7K
A	1999	pop	19M
A	2000	cases	2K
A	2000	pop	20M
B	1999	cases	37K
B	1999	pop	172M
B	2000	cases	80K
B	2000	pop	174M
C	1999	cases	212K
C	1999	pop	1T
C	2000	cases	213K
C	2000	pop	1T

table3

country	year	rate
A	1999	0.7K/1M
A	2000	2K/2M
B	1999	37K/172M
B	2000	80K/174M

table3

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172M
B	2000	80K	174M

table4

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172M
B	2000	80K	174M

`pivot_longer(data, cols, names_to = "name", values_to = "value", values_drop_na = FALSE)`
"Lengthen" data by collapsing several columns into two. Column names move to a new names_to column and values to a new values_to column.

`pivot_longer(table4a, cols = 2:3, names_to = "year", values_to = "cases")`

`pivot_wider(data, names_from = "name", values_from = "value")`
The inverse of `pivot_longer()`. "Widen" data by expanding two columns into several. One column provides the new column names, the other the values.

`pivot_wider(table2, names_from = type, values_from = count)`

Expand Tables



Create new combinations of variables or identify implicit missing values (combinations of variables not present in the data).

`x` `expand(data, ...)` Create a new tibble with all possible combinations of the values of the variables listed in ...
Drop other variables.
`expand(mtcars, cyl, gear, carb)`

`x` `complete(data, ..., fill = list())` Add missing possible combinations of values of variables listed in ... Fill remaining variables with NA.
`complete(mtcars, cyl, gear, carb)`

Handle Missing Values

Drop or replace explicit missing values (NA).

`x` `drop_na(data, ...)` Drop rows containing NA's in ... columns.
`drop_na(x, x2)`

`x` `fill(data, ..., direction = "down")` Fill in NA's in ... columns using the next or previous value.
`fill(x, x2)`

`x` `replace_na(data, replace)` Specify a value to replace NA in selected columns.
`replace_na(x, list(x2 = 2))`

Fonte: [Tidyr Cheat Sheet](#)

Referências

Algumas referências utilizadas para a construção desse material:

- [Introdução à Ciência de Dados - Capítulo 4](#)
- [R for Data Science - Chapter 9](#)
- [Ciência de Dados em R - Capítulo 7](#)
- [Tidyr Cheat Sheet](#)



Slides produzidos pela profa. Tatiana Benaglia.