

Preparado para:



# REFORM/SC2022/126 DELIVERABLE 4 **MÓDULO 2** **GESTÃO E TRATAMENTO DE DADOS EM R**

DESIGNING A NEW VALUATION MODEL  
FOR RURAL PROPERTIES IN PORTUGAL

## Parte I

Formador: Luís Teles Morais | Nova SBE  
*Lisboa, 5 maio 2023*



This project is carried out with funding by the European Union via the Structural Reform Support Programme and in cooperation with the Directorate General for Structural Reform Support of the European Commission

AARC

**NOVA**  
NOVA SCHOOL OF  
BUSINESS & ECONOMICS

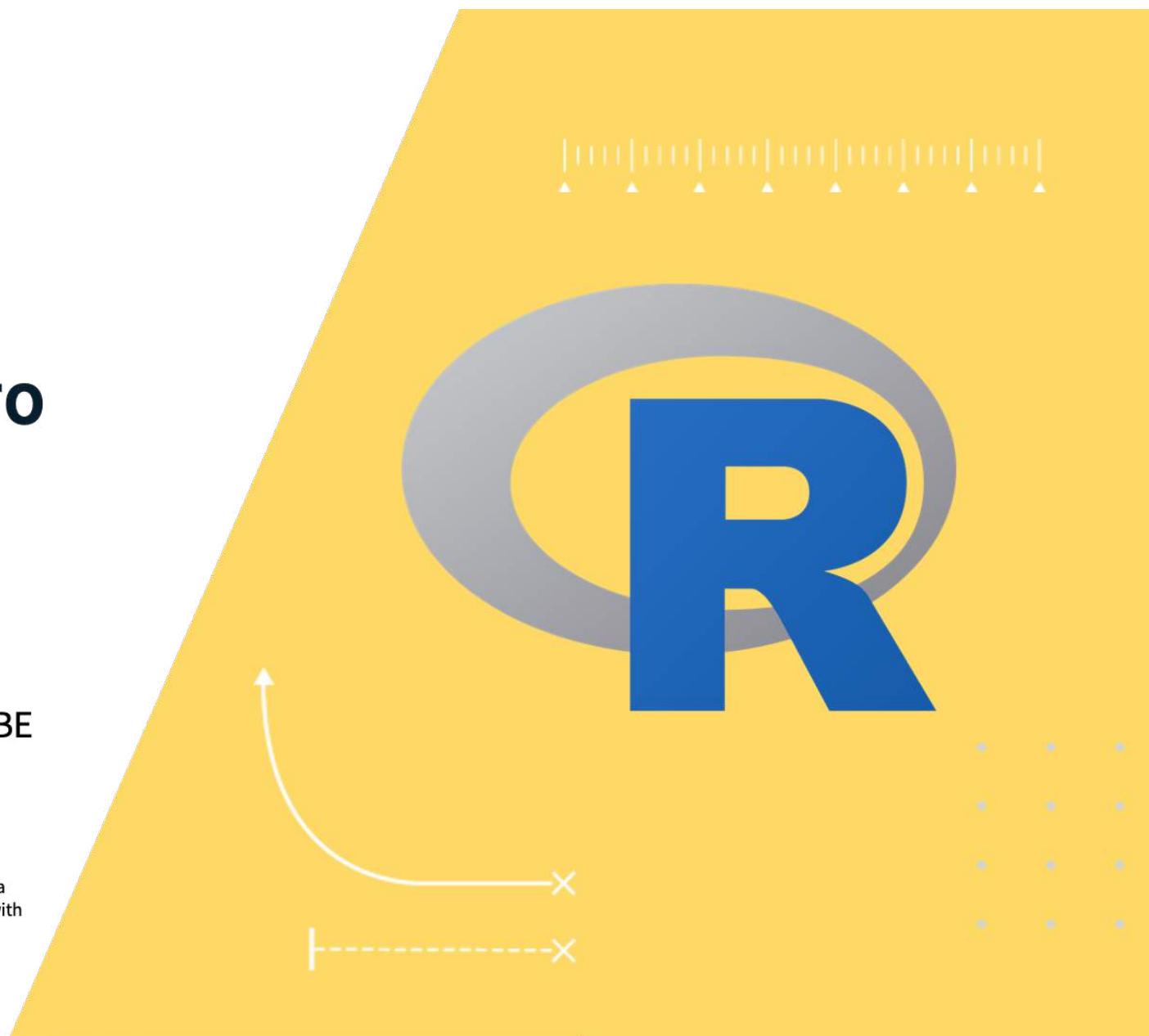
LOBO VASQUES



**esri**



INNERLANDS



# Programa

MÓDULOS	DURAÇÃO
<b>Módulo 1 – Introdução ao R:</b> <ul style="list-style-type: none"> <li>- O que é o R?</li> <li>- Como instalar e configurar o R.</li> <li>- Sintaxe básica e comandos.</li> <li>- Tipos de dados, objetos e classes.</li> </ul>	4 Horas
<b>Módulo 2 – Gestão e tratamento de dados em R:</b> <ul style="list-style-type: none"> <li>- Carregar dados no R.</li> <li>- Perceber as estruturas de dados e <i>subsetting</i>.</li> <li>- Limpeza de dados: <i>missing values</i>, <i>outliers</i> e transformações</li> <li>- Juntar bases de dados</li> </ul>	8 Horas
<b>Módulo 3 – Estatística básica em R:</b> <ul style="list-style-type: none"> <li>- Estatísticas descritivas: medidas de dispersão central e variação.</li> <li>- Distribuições probabilísticas: variáveis discretas e contínuas.</li> <li>- Testes de hipóteses.</li> </ul>	8 Horas

MÓDULOS	DURAÇÃO
<b>Módulo 4 – Regressão Linear:</b> <ul style="list-style-type: none"> <li>- O modelo classico linear.</li> <li>- Estimação de parametros segundo o MMQ.</li> <li>- Testes de hipóteses: significância estatística e ajuste do modelo.</li> <li>- Modelo de regressão múltipla.</li> <li>- Testar as premissas: multicolinearidade, heteroscedasticidade e normalidade dos resíduos.</li> <li>- Critérios de seleção dos modelos.</li> </ul>	12 Horas
<b>Módulo 5 – O modelo:</b> <ul style="list-style-type: none"> <li>- Estrutura do modelo e premissas – Perceber o modelo (4 Hours).</li> <li>- Uso e tratamento dos dados (4 Hours).</li> <li>- Descrição do modelo (4 Hours).</li> <li>- Aplicação do modelo a cada piloto (12 Hours).</li> <li>- Aplicação autónoma do modelo a uma região (8 Hours).</li> </ul>	32 Horas



# Vamos a isso

Aceda a este link para começar já

<https://posit.cloud/content/5906356>



# Quiz 1

**slido.com**

**#3915775**

# **Importar dados no R**

# Ciência de dados

**Importar**

Organizar  
(*Tidy*)

Visualizar  
Transformar

Modelizar

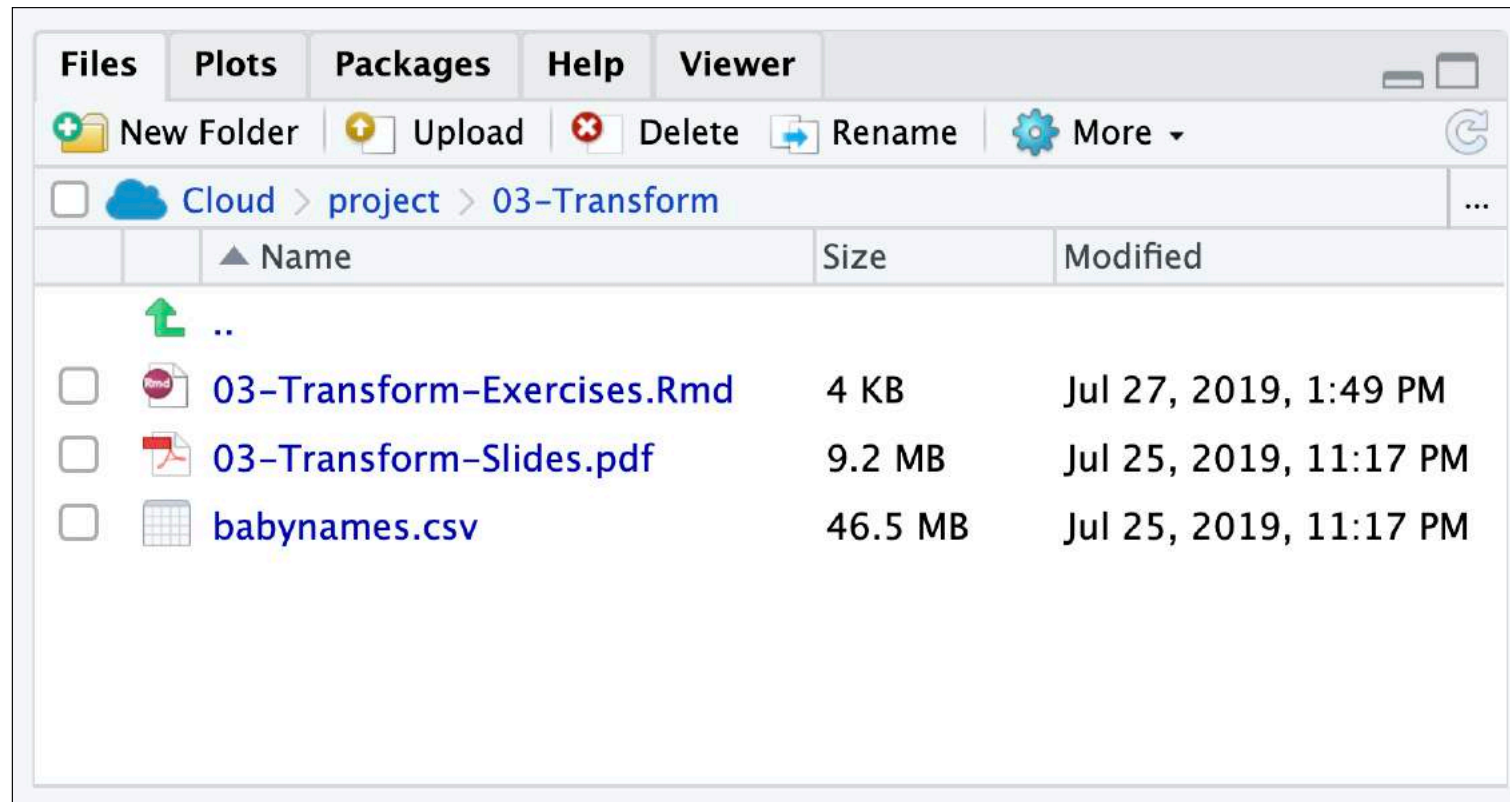
Comunicar

**Programar**

```
graph LR; Importar[Importar] --> Organizar[Organizar (Tidy)]; Organizar --> Visualizar[Visualizar]; Organizar --> Transformar[Transformar]; Visualizar --> Modelizar[Modelizar]; Transformar --> Modelizar; Modelizar --> Comunicar[Comunicar];
```

# babynames.csv

Nomes e sexo dos bebês nascidos nos EUA de 1880 a 2017. 1.9M de observações.





# babynames.csv

```
year,sex,name,n,prop
1880,F,Mary,7065,0.07238359
1880,F,Anna,2604,0.02667896
1880,F,Emma,2003,0.02052149
1880,F,Elizabeth,1939,0.01986579
1880,F,Minnie,1746,0.01788843
1880,F,Margaret,1578,0.0161672
1880,F,Ida,1472,0.01508119
1880,F,Alice,1414,0.01448696
```



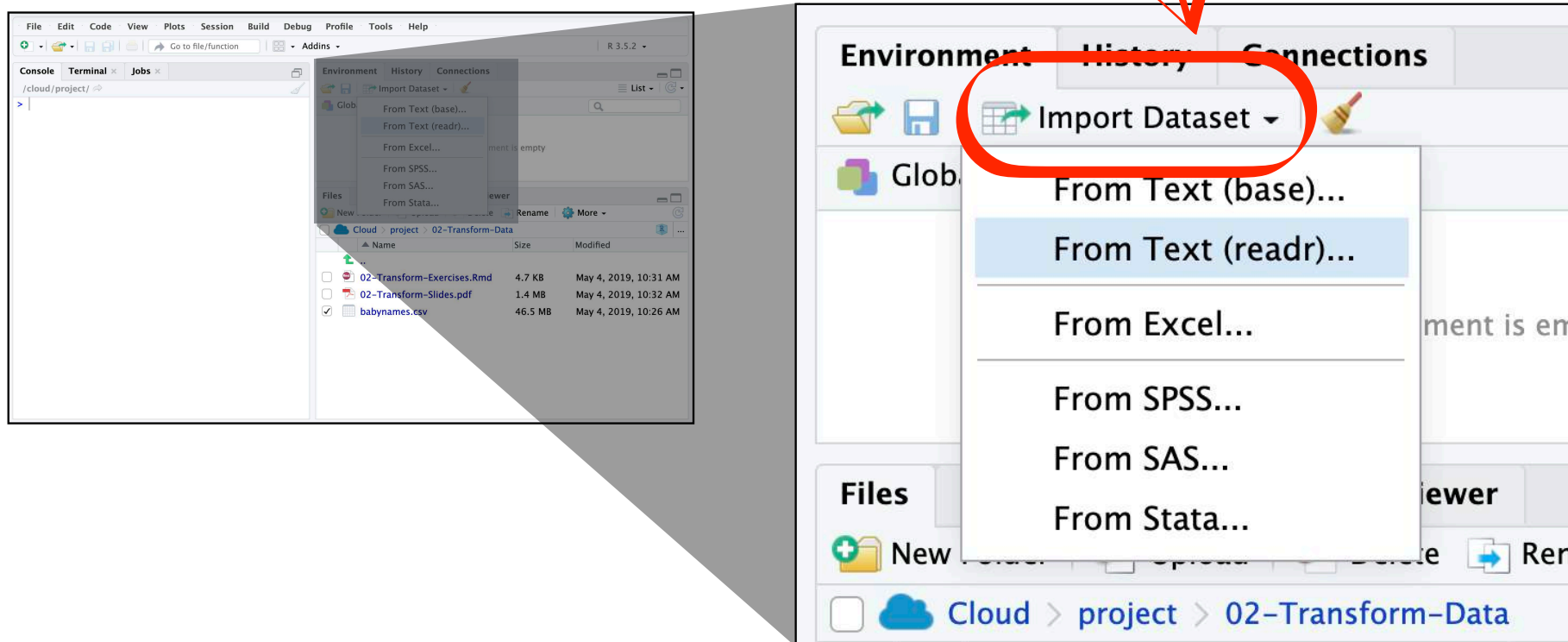
# babynames.csv

```
year,sex,name,n,prop
1880,F,Mary,7065,0.07238359
1880,F,Anna,2604,0.02667896
1880,F,Emma,2003,0.02052149
1880,F,Elizabeth,1939,0.01986579
1880,F,Minnie,1746,0.01788843
1880,F,Margaret,1578,0.0161672
1880,F,Ida,1472,0.01508119
1880,F,Alice,1414,0.01448696
```



# Importar

Import Dataset From Text (readr)...





Import Text Data

File/URL:

/cloud/project/02-Transform-Data/babynames.csv

Browse...

Data Preview:

year (double)	sex (logical)	name (character)	n (double)	prop (double)
1880	FALSE	Mary	7065	0.07238359
1880	FALSE	Anna	2604	0.02667896
1880	FALSE	Emma	2003	0.02052149
1880	FALSE	Elizabeth	1939	0.01986579

Previewing first 50 entries.

Import Options:

Name: babynames

Skip: 0

☒ First Row as Names

☒ Trim Spaces

☒ Open Data Viewer

Delimiter: Comma

Quotes: Default

Locale: Configure...

Escape: None

Comment: Default

NA: Default

Code Preview:

```

library(readr)
babynames <- read_csv("02-Transform-Data/babynames.csv")
View(babynames)

```

? Reading rectangular data using readr

Import

Cancel



## babynames

<b>year</b> <dbl>	<b>sex</b> <chr>	<b>name</b> <chr>	<b>n</b> <dbl>	<b>prop</b> <dbl>
1880	F	Mary	7065	0.07238359
1880	F	Anna	2604	0.02667896
1880	F	Emma	2003	0.02052149
1880	F	Elizabeth	1939	0.01986579
1880	F	Minnie	1746	0.01788843
1880	F	Margaret	1578	0.01616720
1880	F	Ida	1472	0.01508119
1880	F	Alice	1414	0.01448696
1880	F	Bertha	1320	0.01352390
1880	F	Sarah	1288	0.01319605

1-10 of 1,924,665 rows

Previous 1 2 3 4 5 6 ... 100 Next

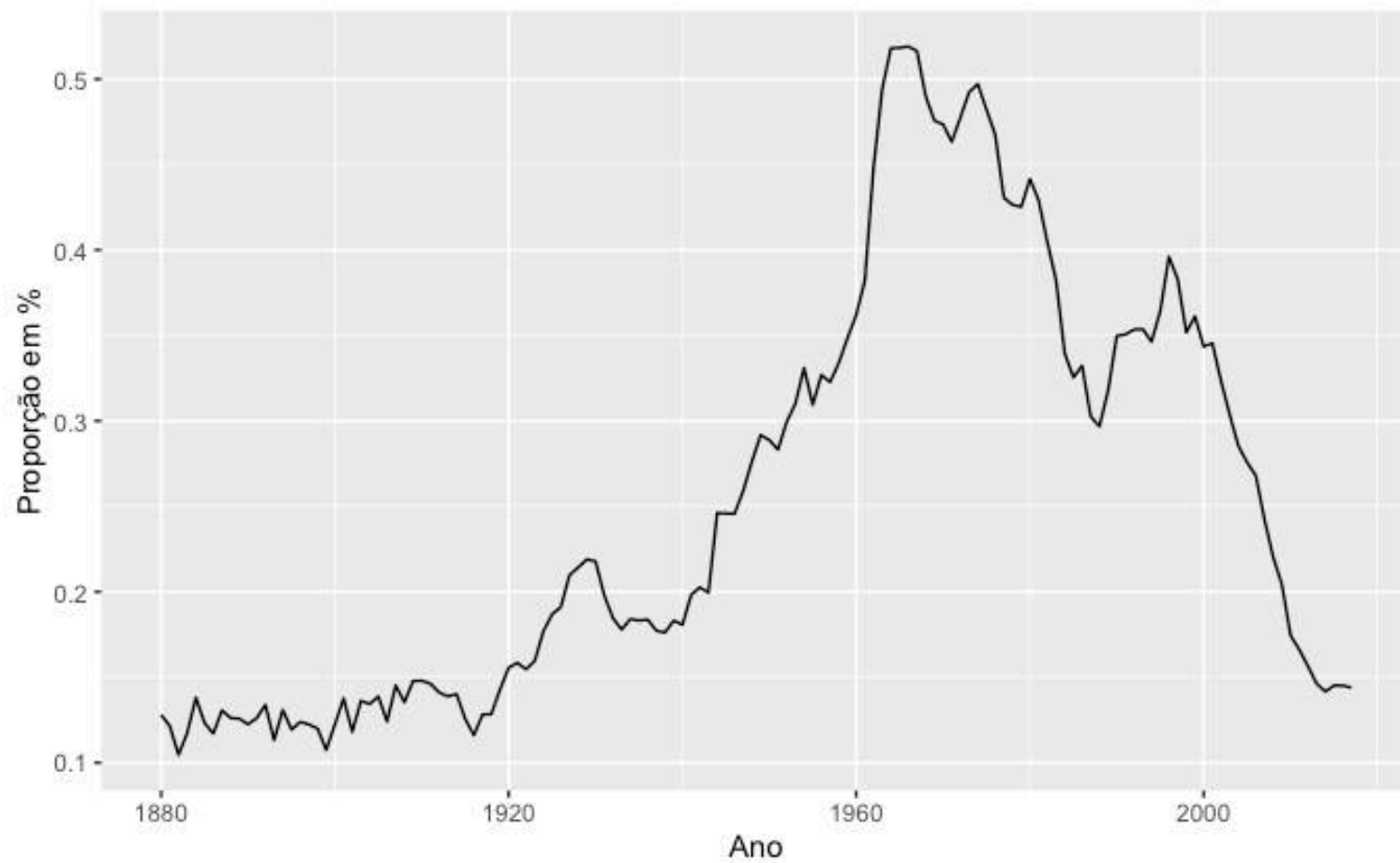


# **Transformação de dados (II)**

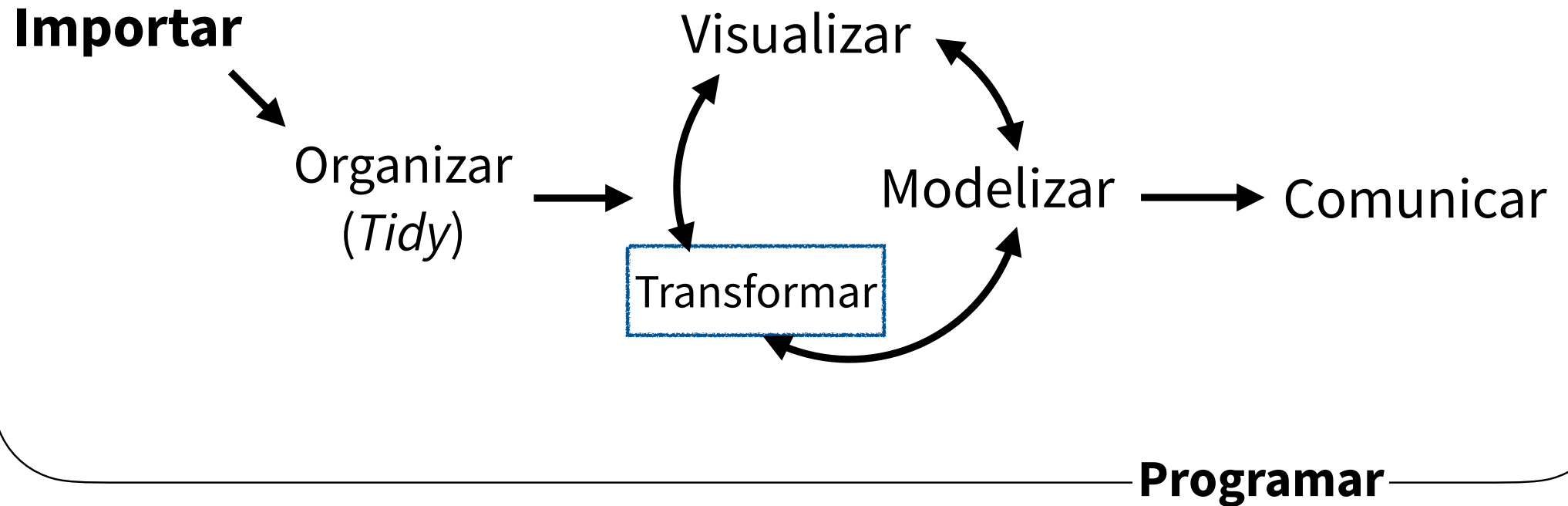
Estruturar e filtrar

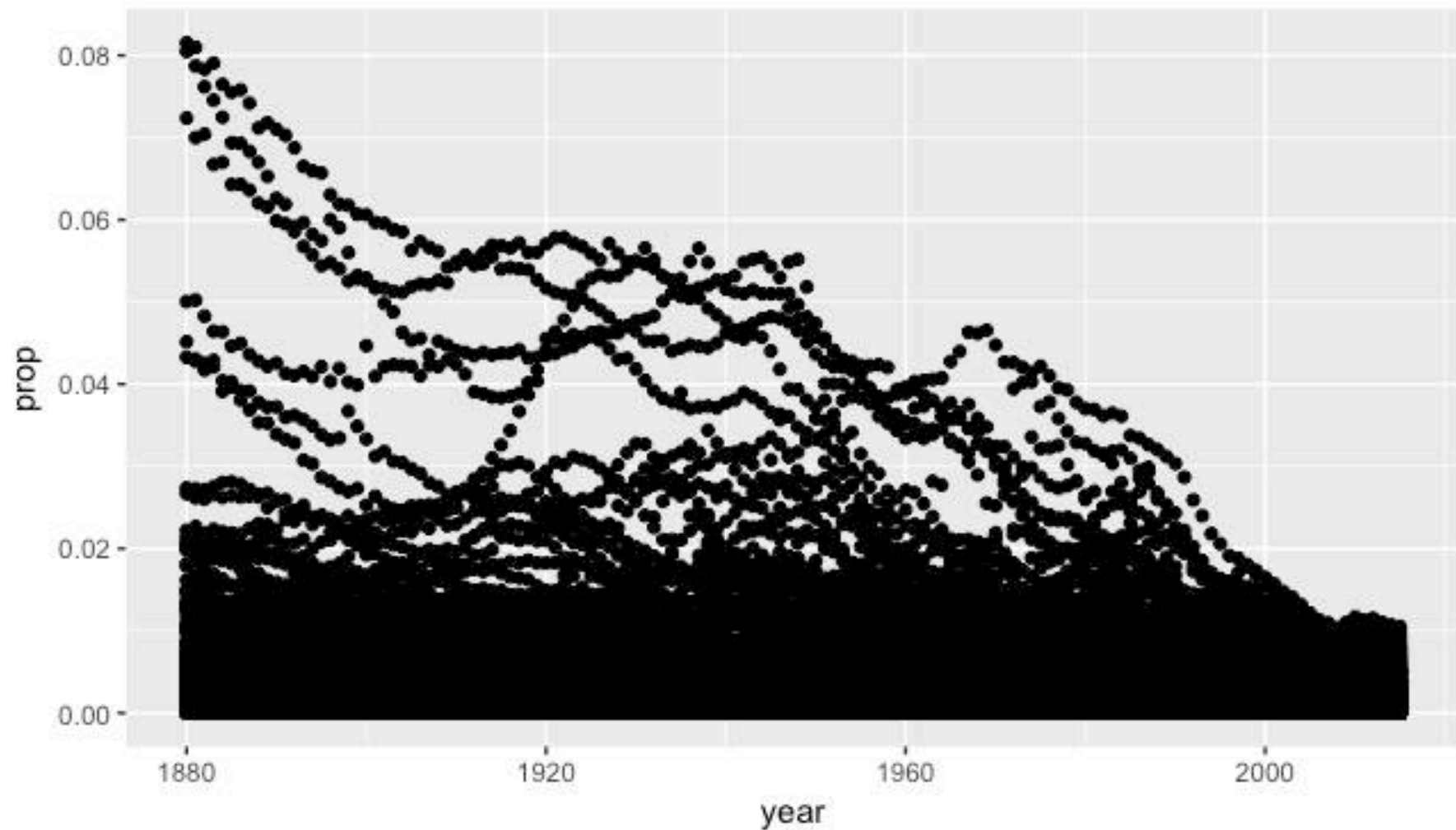


## Percentagem de recém-nascidas com o nome Maria 🇵🇹 nos EUA 🇺🇸



# Ciência de dados

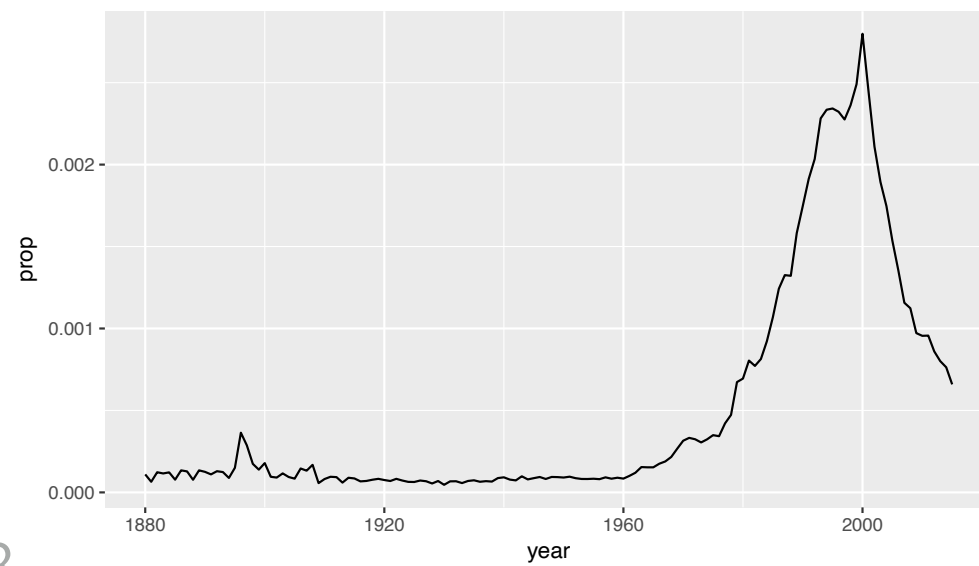
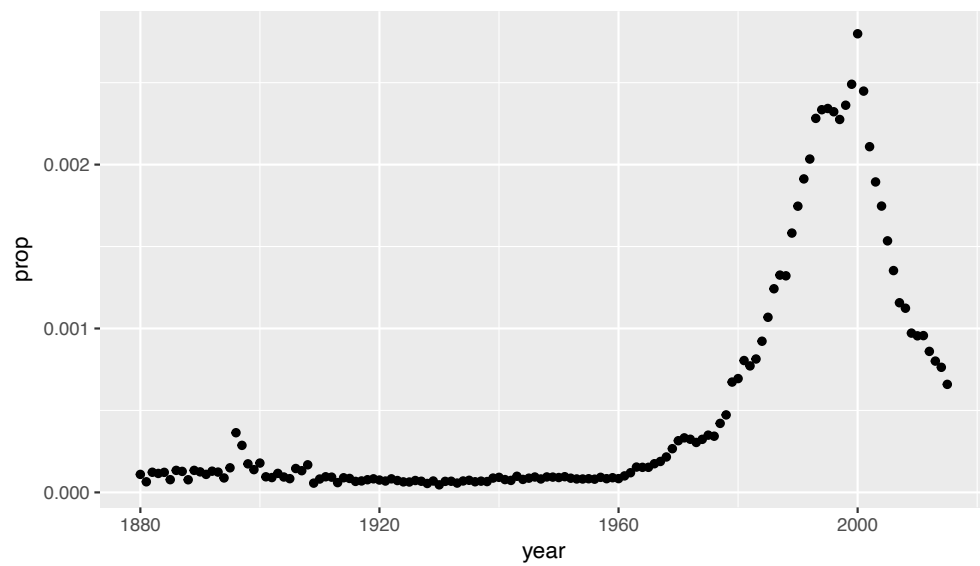
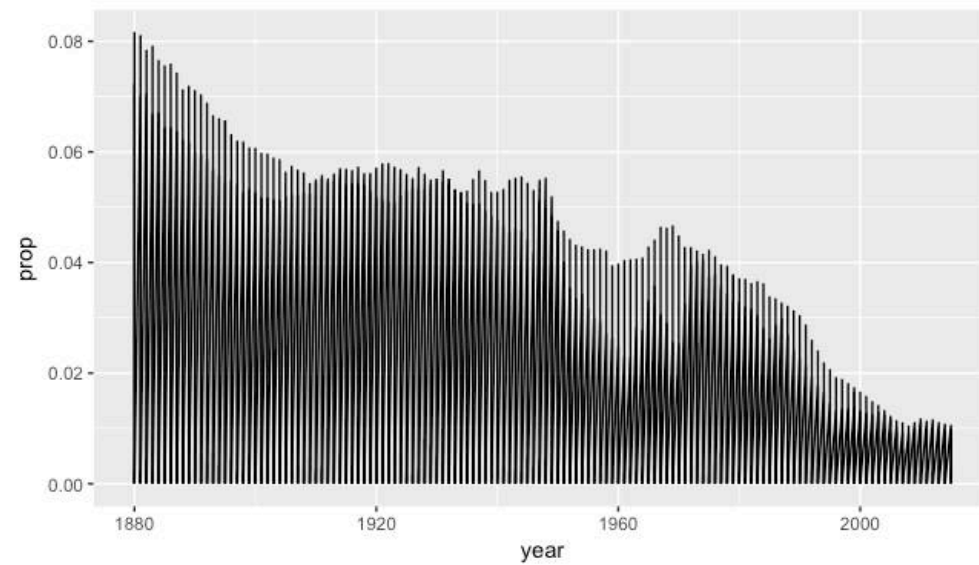
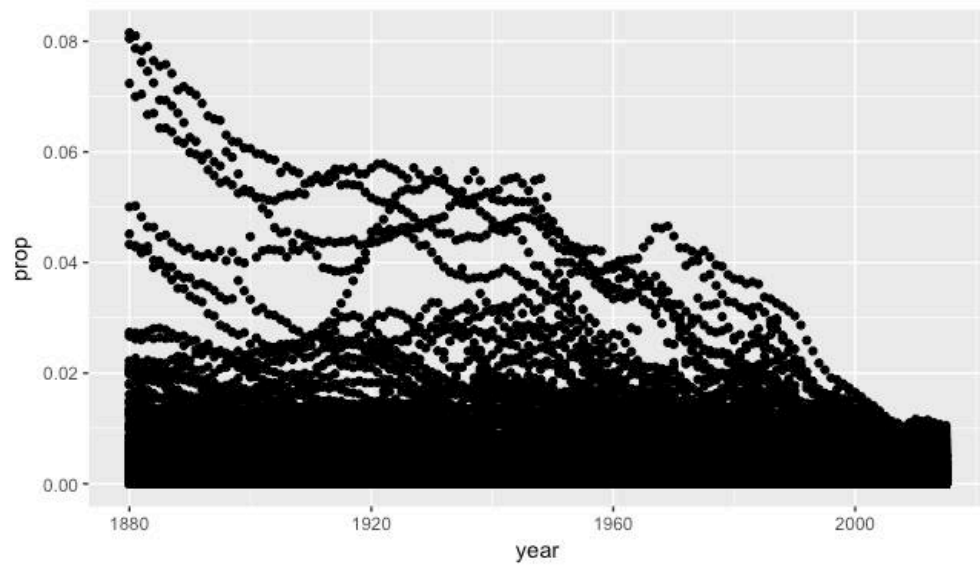




```
ggplot(data = babynames) +  
  geom_point(mapping = aes(x = year, y = prop))
```







## Como isolar as observações de interesse?

	year	sex	name	n	prop
1	1880	F	Mary	7065	0.0724
2	1880	F	Margar	1578	0.0162
3	1880	F	Martha	1040	0.0107
4	1880	F	Marie	471	0.00483
5	1880	F	Maria	125	0.00128
1	1881	F	Mary	6919	0.0700
2	1881	F	Margar	1658	0.0168
3	1881	F	Martha	1044	0.0106
4	1881	F	Marie	499	0.00505
5	1881	F	Maria	120	0.00121
	1881	M	Gideon	7	0.0001



year	sex	name	n	prop
1880	F	Maria	125	0.00128
1881	F	Maria	120	0.00121
...	...	Maria	...	...

## *Subsetting* - subconjuntos de interesse

**select()** - extrair **variáveis**

**filter()** - extrair **observações**

**arrange()** - reordenar **observações**



# select()

Extrair determinadas variáveis (colunas)

```
select(.data, ...)
```

**conjunto de dados  
a transformar**  
(data.frame/tibble)

**nome das variáveis a extrair**  
(ou função que selecione  
variáveis)

# select()

Extrair determinadas variáveis (colunas)

```
select(babynames, name, prop)
```

babynames

year	sex	name	n	prop
1880	M	John	9655	0.0815
1880	M	William	9532	0.0805
1880	M	James	5927	0.0501
1880	M	Charles	5348	0.0451
1880	M	Garrett	13	0.0001
1881	M	John	8769	0.081



name	prop
John	0.0815
William	0.0805
James	0.0501
Charles	0.0451
Garrett	0.0001
John	0.081



Como seleccionar a columna **n**?

Como seleccionar a columna **n**?

```
select(babynames, n)
```

```
select(babynames, n)
```

```
#      n  
#  <int>  
# 1  7065  
# 2  2604  
# 3  2003  
# 4  1939  
# 5  1746  
# ...  ...
```

Que tipo de objeto é este?



# select()

Extrair determinadas variáveis (colunas)

```
select(babynames, n)
```

babynames

year	sex	name	n	prop
1880	M	John	9655	0.0815
1880	M	William	9532	0.0805
1880	M	James	5927	0.0501
1880	M	Charles	5348	0.0451
1880	M	Garrett	13	0.0001
1881	M	John	8769	0.081



n
9655
9532
5927
5348
13
8769

**TIBBLE**



\$

Extrair *conteúdo* de uma variável (como um vector)

```
babynames$n
```

babynames

year	sex	name	n	prop
1880	M	John	9655	0.0815
1880	M	William	9532	0.0805
1880	M	James	5927	0.0501
1880	M	Charles	5348	0.0451
1880	M	Garrett	13	0.0001
1881	M	John	8769	0.081

→ 9655 9532 5927 5348 ...



\$

Extraire *conteúdo* de uma variável (como um vector)

```
babynames$n
```

**data  
frame**

\$

**nome da variável  
(sem plicas)**

# select() helpers - seleção de variáveis

- Selecionar intervalo de várias colunas

```
select(mpg, cty:class)
```

- - Todas as colunas exceto

```
select(mpg, -c(cty, hwy))
```

**starts\_with()** - Colunas cujo nome começa por...


```
select(mpg, starts_with("c"))
```

**ends\_with()** - Colunas cujo nome acaba em...

```
select(mpg, ends_with("y"))
```

etc.

## Data Transformation with dplyr : : CHEAT SHEET



**dplyr** functions work with pipes and expect **tidy data**. In tidy data:

- Each **variable** is in its own **column**
- Each **observation**, or **case**, is in its own **row**
- $x \%>\% f(y)$  becomes  $f(x, y)$

**pipes**

These apply **summary functions** to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value (see back).

**summary function**

**summarise(data, ...)**  
Compute table of summaries.  
`summarise(mpg, avg = mean(mpg))`

**count(x, ...)** (`wt = NULL`, `sort = FALSE`)  
Count number of rows in each group defined by the variables in ... Also **tally()**.  
`count(iris, Species)`

**VARIATIONS**

- summarise\_all()** - Apply funs to every column.
- summarise\_at()** - Apply funs to specific columns.
- summarise\_if()** - Apply funs to all cols of one type.

**Group Cases**

Use **group\_by()** to create a "grouped" copy of a table. dplyr functions will manipulate each "group" separately and then combine the results.

`mtcars %>%  
group_by(cyl) %>%  
summarise(avg = mean(mpg))`

**group\_by(data, ...)**, `add = FALSE`  
Returns copy of table grouped by ...  
`g_cyls = group_by(iris, Species)`

**ungroup(x, ...)**  
Returns ungrouped copy of table.  
`ungroup(g_iris)`

**Logical and boolean operators to use with filter()**

See **base::logic** and **?Comparison** for help.

**ARRANGE CASES**

**arrange(data, ...)** Order rows by values of a column or columns (low to high). Use with **desc()** to order from high to low.  
`arrange(mtcars, mpg)`  
`arrange(mtcars, desc(mpg))`

**ADD CASES**

**add\_row(data, ...)**, `before = NULL`, `after = NULL`  
Add one or more rows to a table.  
`add_row(mtcars, displations = 1, waiting = 1)`

**EXTRACT CASES**

Row functions return a subset of rows as a new table.

**filter(data, ...)** Extract rows that meet logical criteria. `filter(iris, Sepal.Length > 7)`

**distinct(data, ...)**, `keep_all = FALSE` Remove rows with duplicate values.  
`distinct(iris, Species)`

**sample\_frac(frac, size = 1, replace = FALSE, weight = NULL, n = parent.frame())** Randomly select fraction of rows.  
`sample_frac(iris, 0.5, replace = TRUE)`

**sample\_n(n, size, replace = FALSE, weight = NULL, n = parent.frame())** Randomly select size rows.  
`sample_n(iris, 10, replace = FALSE)`

**slice(data, ...)** Select rows by position.  
`slice(iris, 10:20)`

**top\_n(x, n, wt)** Select and order top n entries (by group if grouped data).  
`top_n(mtcars, 5, displations)`

**EXTRACT VARIABLES**

Column functions return a set of columns as a new vector or table.

**pull(data, var = 1)** Extract column values as a vector. Choose by name or index.  
`pull(iris, Sepal.Length)`

**select(data, ...)**  
Extract columns as a table. Also **select\_if()**.  
`select(iris, Sepal.Length, Species)`

Use these helpers with **select()**.  
e.g. `select(iris, starts_with("Sepal"))`

**contains(match)** `num_range(prefix, range)` e.g. `mpg:cyl`  
**ends\_with(match)** `one_of(...)` -, e.g. `-Species`  
**matches(match)** **starts\_with(match)**

**MAKE NEW VARIABLES**

These apply vectors as input (see back).

**R Studio**

RStudio is a trademark of RStudio, Inc. • CC BY-SA/RStudio • info@rstudio.com • 844-445-1332 • rstudio.com • Learn more with R in your graphics package • © dplyr, "tidyverse" • dplyr 0.7.0 • 9999 • Updated 2017-10





Qual destas não seleciona as colunas **name** e **n**?

`select(babynames, -c(year, sex, prop))`

`select(babynames, name:n)`

`select(babynames, starts_with("n"))`

`select(babynames, ends_with("n"))`

Qual destas não seleciona as colunas **name** e **n**?

`select(babynames, -c(year, sex, prop))`

`select(babynames, name:n)`

`select(babynames, starts_with("n"))`

`select(babynames, ends_with("n"))`

## Como isolar as observações de interesse?

	year	sex	name	n	prop
1	1880	F	Mary	7065	0.0724
2	1880	F	Margar	1578	0.0162
3	1880	F	Martha	1040	0.0107
4	1880	F	Marie	471	0.00483
5	1880	F	Maria	125	0.00128
1	1881	F	Mary	6919	0.0700
2	1881	F	Margar	1658	0.0168
3	1881	F	Martha	1044	0.0106
4	1881	F	Marie	499	0.00505
5	1881	F	Maria	120	0.00121
	1881	M	Gideon	7	0.0001



year	sex	name	n	prop
1880	F	Maria	125	0.00128
1881	F	Maria	120	0.00121
...	...	Maria	...	...

# filter()

Extrair observações (linhas) de acordo com critérios lógicos

```
filter(.data, ... )
```

**tabela de dados a  
transformar**

**um ou mais testes de  
lógica** (que serão  
avaliados linha a linha)

# filter()

Extrair observações (linhas) de acordo com critérios lógicos

```
filter(babynames, name == "Maria")
```

babynames

year	sex	name	n	prop
1880	F	Mary	7065	0.0724
1880	F	Margaret	1578	0.0162
1880	F	Martha	1040	0.0107
1880	F	Marie	471	0.00483
1880	F	Maria	125	0.00128
1881	F	Mary	6919	0.0700



year	sex	name	n	prop
1880	F	Maria	125	0.00128
1881	F	Maria	120	0.00121
...	...	Maria	...	...

35



# filter()

Extrair observações (linhas) de acordo com critérios lógicos

```
filter(babynames, name == "Maria")
```

babynames

year	sex	name	n	prop
1880	F	Mary	7065	0.0724
1880	F	Margaret	1578	0.0162
1880	F	Martha	1040	0.0107
1880	F	Marie	471	0.00483
1880	F	Maria	125	0.00128
1881	F	Mary	6919	0.0700

= serve para definir opções etc.

**== para testes de igualdade**  
(TRUE ou FALSE)



# Testes de lógica

<code>x &lt; y</code>	Menor que
<code>x &gt; y</code>	Maior que
<code>x == y</code>	Igual a
<code>x &lt;= y</code>	Menor ou igual a
<code>x &gt;= y</code>	Maior ou igual a
<code>x != y</code>	Não é igual a
<code>x %in% y</code>	Pertence a

O que significa NA?

1

"1"

"one"

NA

Qual é o resultado de:

**1 == 1**

Qual é o resultado de:

**1 == 1**

**TRUE**

Qual é o resultado de:

**1 == NA**

Qual é o resultado de:

**1 == NA**

**NA**



Qual é o resultado de:

**NA == NA**

Qual é o resultado de:

**NA == NA**

**NA**

Qual é o resultado de:

`is.na(NA)`

**TRUE**

# Testes de lógica

## ?Comparison

<code>x &lt; y</code>	Menor que
<code>x &gt; y</code>	Maior que
<code>x == y</code>	Igual a
<code>x &lt;= y</code>	Menor ou igual a
<code>x &gt;= y</code>	Maior ou igual a
<code>x != y</code>	Não é igual a
<code>x %in% y</code>	Pertence a
<code>is.na(x)</code>	Valor não disponível
<code>!is.na(x)</code>	Valor existe

Aplique a função *filter* aos dados *babynames* para extrair:

- As observações onde **prop** é maior ou igual a 0.08
- As observações de crianças chamadas José

```
filter(babynames, prop >= 0.08)
```

#	year	sex	name	n	prop
# 1	1880	M	John	9655	0.08154630
# 2	1880	M	William	9531	0.08049899
# 3	1881	M	John	8769	0.08098299

```
filter(babynames, name == "Jose")
```

	year	sex	name	n	prop
	<dbl>	<chr>	<chr>	<dbl>	<dbl>
1	1880	M	Jose	84	0.000709
2	1881	M	Jose	76	0.000702
3	1882	M	Jose	95	0.000778
4	1883	M	Jose	74	0.000658

1. = em vez de ==

```
filter(babynames, name = "Jose")  
filter(babynames, name == "Jose")
```

2. Esquecer aspas

```
filter(babynames, name == Jose)  
filter(babynames, name == "Jose")
```

3. Caracteres especiais

```
filter(babynames, name == "José")  
filter(babynames, name == "Jose")
```



# filter()

Extrai observações que cumpram todos os critérios fornecidos

```
filter(babynames, name == "Maria", year == 1880)
```

babynames

year	sex	name	n	prop
1880	F	Mary	7065	0.0724
1880	F	Margaret	1578	0.0162
1880	F	Martha	1040	0.0107
1880	F	Marie	471	0.00483
1880	F	Maria	125	0.00128
1881	F	Mary	6919	0.0700



year	sex	name	n	prop
1880	F	Maria	125	0.00128

50



# filter(!is.na())

Remover observações com dados em falta

```
filter(x, !is.na(x2))
```

<sup>x</sup>  
X

x1	x2
A	1
B	NA
C	NA
D	3
E	NA

→

x1	x2
A	1
D	3



# filter()

Extrai observações que cumpram todos os critérios fornecidos

```
filter(babynames, name == "Maria" & year == 1880)
```

babynames

year	sex	name	n	prop
1880	F	Mary	7065	0.0724
1880	F	Margaret	1578	0.0162
1880	F	Martha	1040	0.0107
1880	F	Marie	471	0.00483
1880	F	Maria	125	0.00128
1881	F	Mary	6919	0.0700



year	sex	name	n	prop
1880	F	Maria	125	0.00128

52



# Operadores lógicos

?base::Logic

<code>a &amp; b</code>	e
<code>a   b</code>	ou
<code>xor(a,b)</code>	ou ... ou
<code>!a</code>	não é
<code>( )</code>	agrupar testes

$x \geq 2 \ \& \ x < 3$



TRUE & TRUE



TRUE

# Atenção

3. Testes lógicos de dois lados (intervalos) não funcionam

```
filter(babynames, 10 < n < 20)  
filter(babynames, 10 < n, n < 20)
```

4. Juntar vários testes (em vez de usar %in%)

```
filter(babynames, n == 5 | n == 6 | n == 7 | n == 8)  
filter(babynames, n %in% c(5, 6, 7, 8))
```

# arrange()

Ordenar do mais pequeno para o maior

```
arrange(.data, ...)
```

**variáveis a usar para ordenação**  
(por ordem de prioridade)

# arrange()

Ordenar do mais pequeno para o maior

```
arrange(babynames, n)
```

babynames

year	sex	name	n	prop
1880	M	John	9655	0.0815
1880	M	William	9532	0.0805
1880	M	James	5927	0.0501
1880	M	Charles	5348	0.0451
1880	M	Garrett	13	0.0001
1881	M	John	8769	0.081



57

year	sex	name	n	prop
1880	M	Garrett	13	0.0001
1880	M	Charles	5348	0.0451
1880	M	James	5927	0.0501
1881	M	John	8769	0.081
1880	M	William	9532	0.0805
1880	M	John	9655	0.0815





# desc()

Ordenar do maior para o mais pequeno

```
arrange(babynames, desc(n))
```

babynames

year	sex	name	n	prop
1880	M	John	9655	0.0815
1880	M	William	9532	0.0805
1880	M	James	5927	0.0501
1880	M	Charles	5348	0.0451
1880	M	Garrett	13	0.0001
1881	M	John	8769	0.081



58

year	sex	name	n	prop
1880	M	John	9655	0.0815
1880	M	William	9532	0.0805
1881	M	John	8769	0.081
1880	M	James	5927	0.0501
1880	M	Charles	5348	0.0451
1880	M	Garrett	13	0.0001



Qual é o nome mais raro?  
E o mais frequente?

```
arrange(babynames, n, prop)
```

```
#   year  sex  name  n  prop
# 1 2007   M  Aaban  5 2.259872e-06
# 2 2007   M  Aareon  5 2.259872e-06
# 3 2007   M  Aaris  5 2.259872e-06
# 4 2007   M   Abd  5 2.259872e-06
# 5 2007   M Abdulazeez  5 2.259872e-06
# 6 2007   M  Abdulhadi  5 2.259872e-06
# 7 2007   M  Abdulhamid  5 2.259872e-06
# 8 2007   M  Abdulkadir  5 2.259872e-06
# 9 2007   M Abdulraheem  5 2.259872e-06
# 10 2007   M  Abdulrahim  5 2.259872e-06
# ... with 1,924,655 more rows
```

```
arrange(babynames, desc(n))
```

```
#   year sex  name    n    prop
# 1 1947  F   Linda 99680 0.05483609
# 2 1948  F   Linda 96211 0.05521159
# 3 1947  M   James 94763 0.05102057
# 4 1957  M Michael 92726 0.04238659
# 5 1947  M   Robert 91646 0.04934237
# 6 1949  F   Linda 91010 0.05184281
# 7 1956  M Michael 90623 0.04225479
# 8 1958  M Michael 90517 0.04203881
# 9 1948  M   James 88588 0.04969679
#10 1954  M Michael 88493 0.04279403
# ... with 1,924,655 more rows
```

# Mini-teste

Qual o nome de menino mais popular em 2015?

```
boys_2015 <- filter(babynames, year == 2015, sex == "M")  
boys_2015 <- select(boys_2015, name, n)  
boys_2015 <- arrange(boys_2015, desc(n))  
boys_2015
```

```
boys_2015 <- filter(babynames, year == 2015, sex == "M")  
boys_2015 <- select(boys_2015, name, n)  
boys_2015 <- arrange(boys_2015, desc(n))  
boys_2015
```

```
arrange(select(filter(babynames, year == 2015,  
  sex == "M"), name, n), desc(n))
```



# sintaxe dplyr de manipulação de dados

Todas as funções têm como primeiro argumento uma tabela de dados e devolvem outra como resultado.

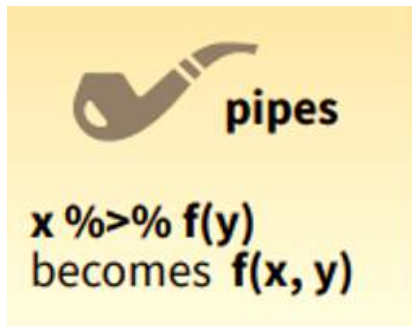
```
filter(.data, ... )
```

**função dplyr**

**data frame a  
transformar**

**outros argumentos  
da função**

# Operador pipe %>%



$\%>\%$

`babynames %>% filter(_____, n == 99680)`

Passa resultado da esquerda à função da direita, para o seu primeiro argumento. Experimente:

```
filter(babynames, n == 99680)  
babynames %>% filter(n == 99680)
```

# Pipes

```
babynames  
boys_2015 <- filter(babynames, year == 2015, sex == "M")  
boys_2015 <- select(boys_2015, name, n)  
boys_2015 <- arrange(boys_2015, desc(n))  
boys_2015
```

```
babynames %>%  
  filter(year == 2015, sex == "M") %>%  
  select(name, n) %>%  
  arrange(desc(n))
```

# Atalho de teclado para %>%

**Ctrl** + **Shift** + **M** (Windows)

e para <-

**Alt** + **-** (Windows)

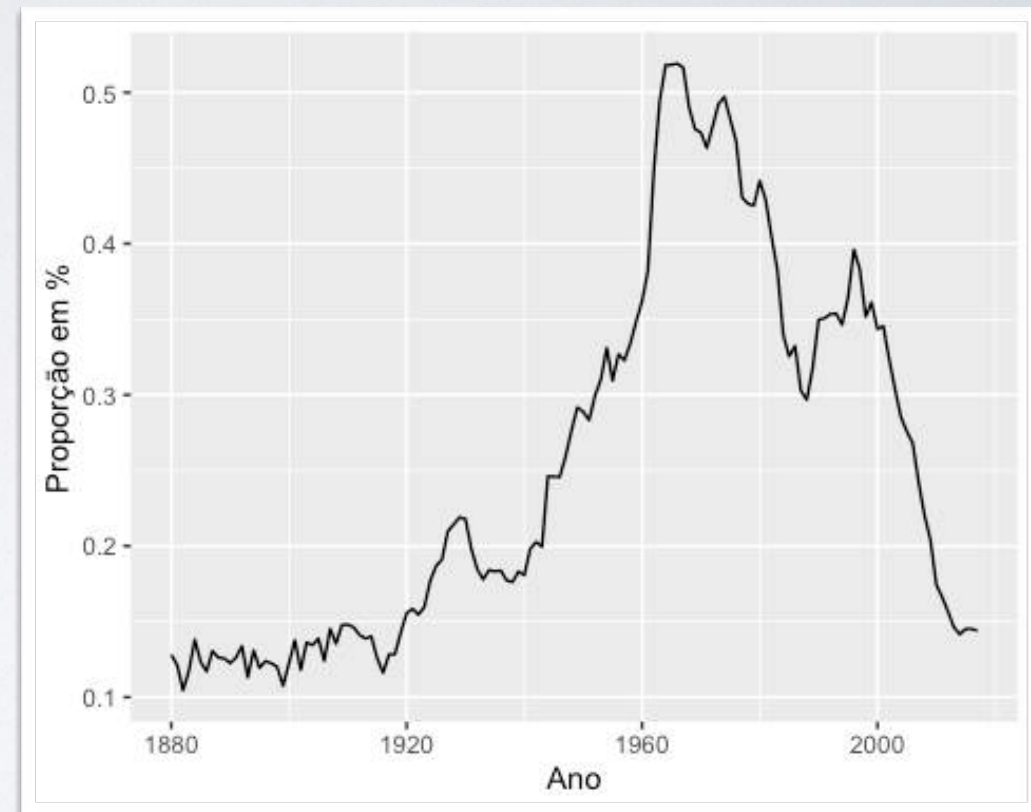
Qual o nome de menina mais popular em 2015?

```
babynames %>%  
  filter(year == 2017, sex == "F") %>%  
  select(name, n) %>%  
  arrange(desc(n))
```

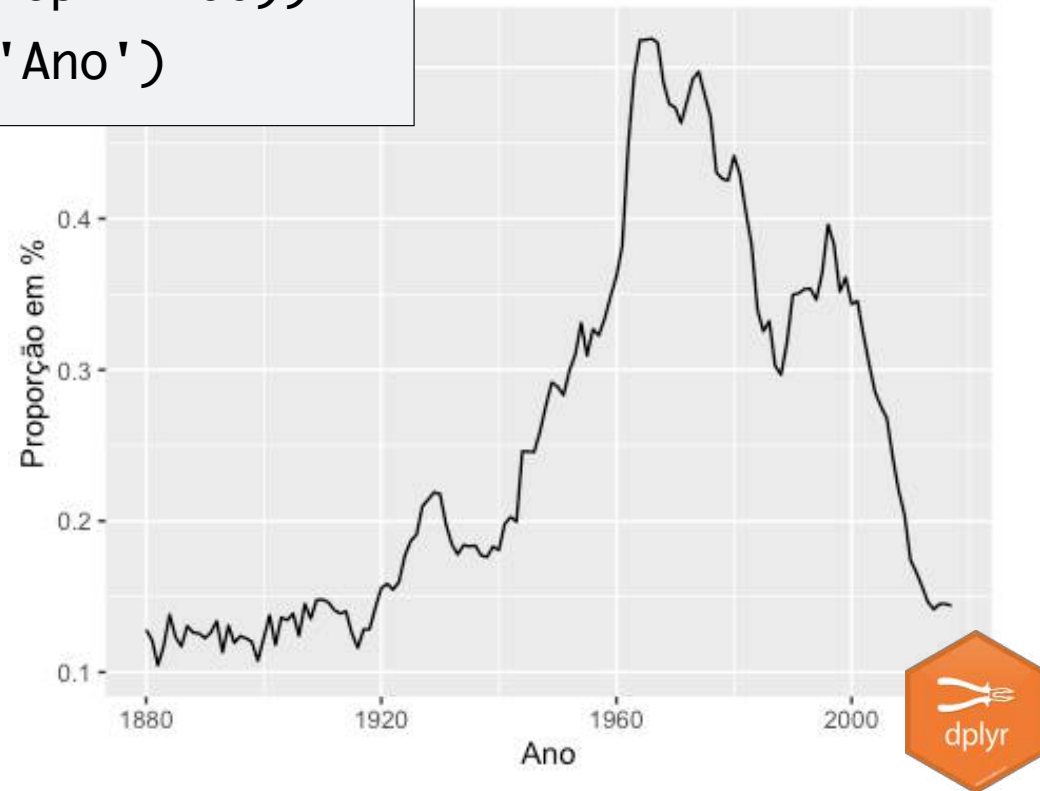
```
#   name      n  
# 1 Emma    19738  
# 2 Olivia  18632  
# 3 Ava     15902  
# 4 Isabella 15100  
# 5 Sophia  14831  
# 6 Mia     13437  
# 7 Charlotte 12893  
# 8 Amelia  11800  
# 9 Evelyn  10675  
## ... with 20,170 more rows
```

# Reproduzir o gráfico

1. Filtrar dados para conter apenas as “Marias”
2. Selecionar apenas as colunas necessárias para o gráfico
3. Construir gráfico de linhas com **year** no eixo do x and **prop** no eixo dos y



```
babynames %>%  
  filter(name == "Maria", sex == "F") %>%  
  select(year, prop) %>%  
  ggplot() +  
    geom_line(mapping = aes(year, prop * 100)) +  
    labs(y = 'Proporção em %', x = 'Ano')
```





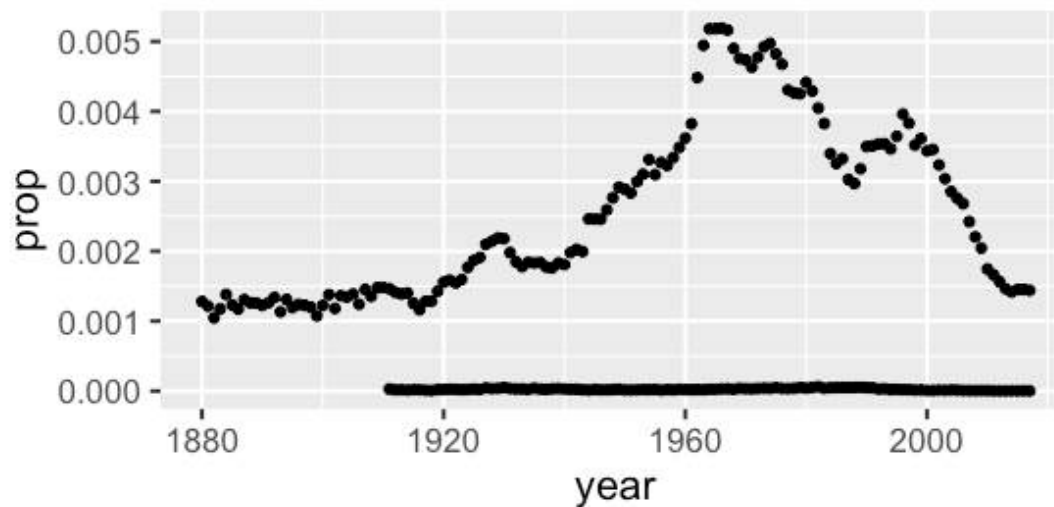
```
babynames %>%
```

```
  filter(name == "Maria") %>%
```

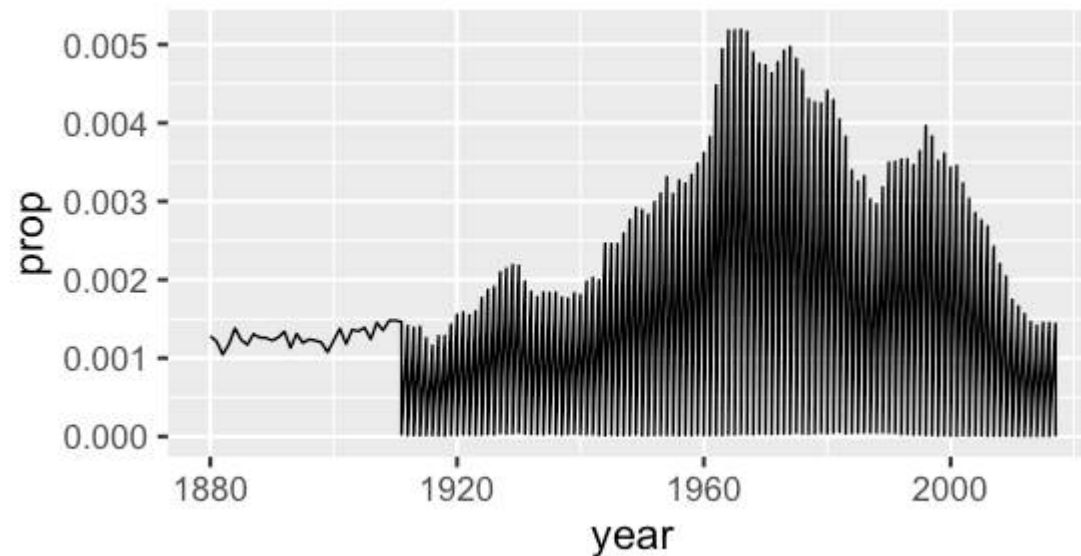
```
  ggplot() +
```

```
    geom_point(mapping = aes(year, prop))
```

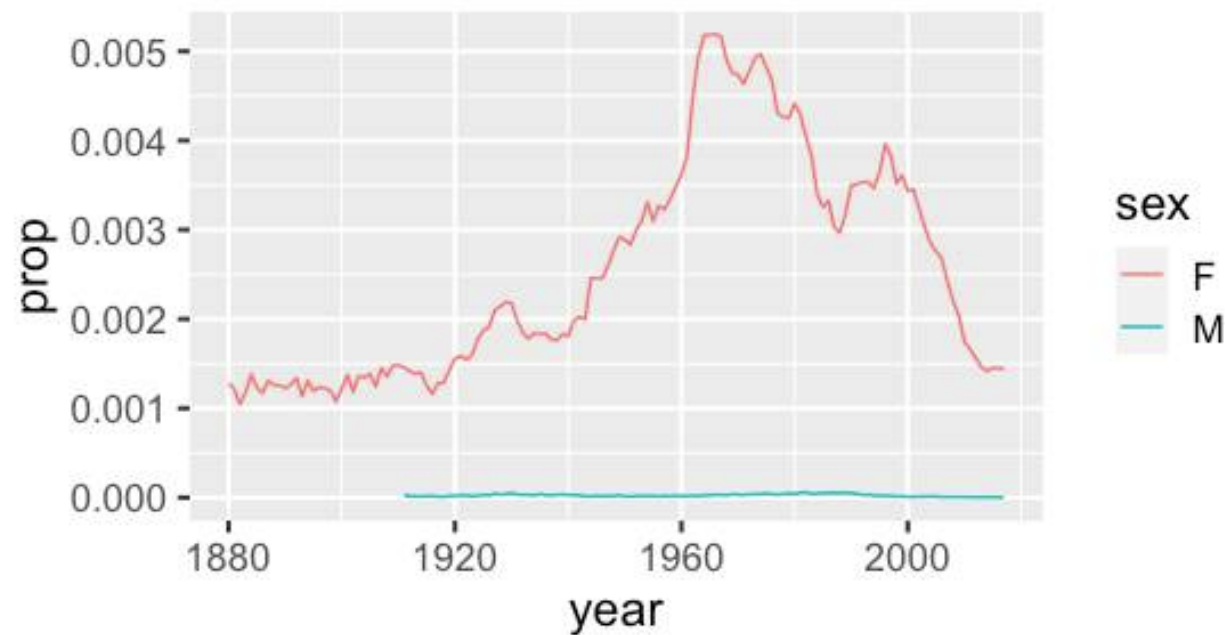
Sem filtrar na  
variável SEX



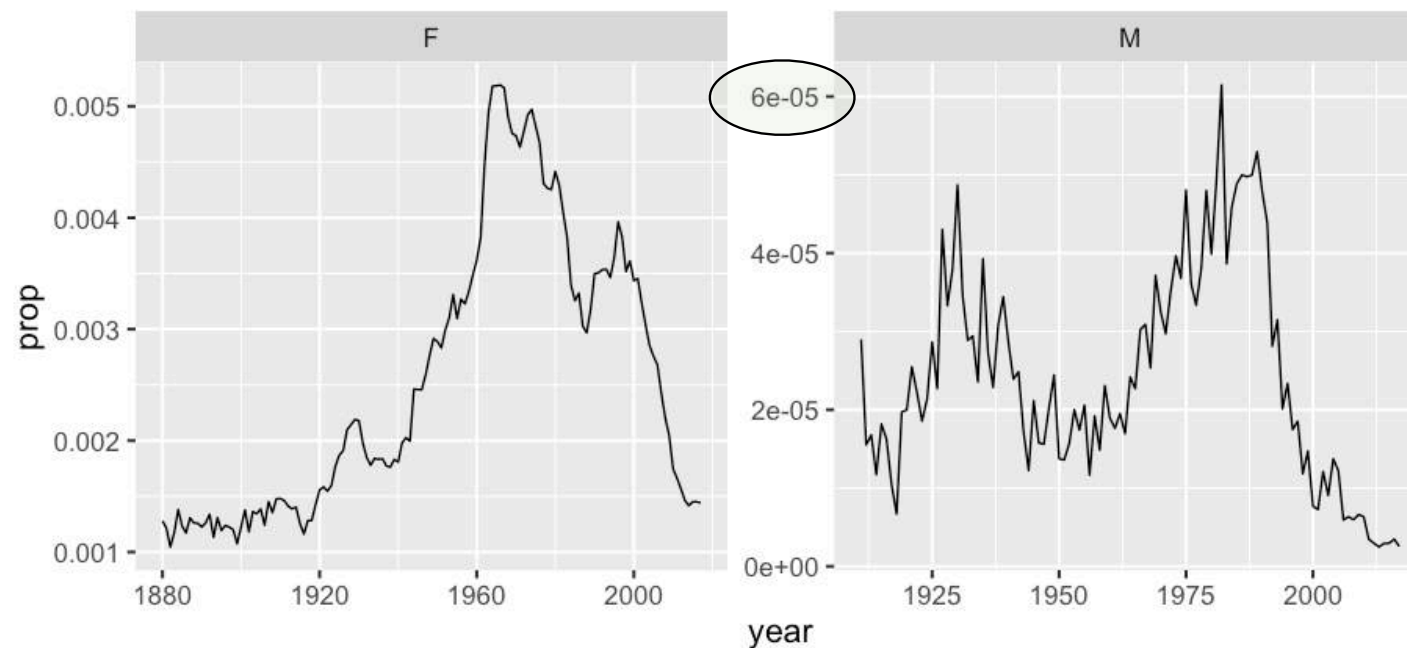
```
babynames %>%  
  filter(name == "Maria") %>%  
  ggplot() +  
    geom_line(mapping = aes(year, prop))
```



```
babynames %>%  
  filter(name == "Maria") %>%  
  ggplot() +  
    geom_line(mapping = aes(year, prop, color = sex))
```



```
babynames %>% filter(name == "Michael") %>%  
  ggplot() +  
    geom_line(mapping = aes(year, prop)) +  
    facet_wrap(~ sex)
```



# Transformação de dados (I)

Arrumar - *tidy*

# sales.xlsx

```
clientes <- readxl::read_xlsx("sales_data/sales.xlsx", sheet = 'clientes')
clientes
```

```
## # A tibble: 2 × 4
##   id_cliente item_1 item_2          item_3
##   <dbl> <chr> <chr> <chr>
## 1      1      1 pao      leite      banana
## 2      2      2 leite    papel higienico <NA>
```

```
precos <- readxl::read_xlsx("sales_data/sales.xlsx", sheet = 'precos')
precos
```

```
## # A tibble: 5 × 2
##   item      price
##   <chr>    <dbl>
## 1 abacate      2
## 2 banana     0.5
## 3 pao         1.5
## 4 leite        1
## 5 papel higienico 3
```



# clientes

## Temos...

```
## # A tibble: 2 × 4
##   id_cliente item_1 item_2 item_3
##   <dbl> <chr> <chr> <chr>
## 1         1 pao    leite banana
## 2         2 leite papel higienico <NA>
```

## Queremos...

```
## # A tibble: 6 × 3
##   id_cliente item_no item
##   <dbl> <chr> <chr>
## 1         1 item_1 pao
## 2         1 item_2 leite
## 3         1 item_3 banana
## 4         2 item_1 leite
## 5         2 item_2 papel higienico
## 6         2 item_3 <NA>
```



# 0 objetivo

wide			
id	x	y	z
1	a	c	e
2	b	d	f





# Wide vs. long

## wide

mais colunas / variáveis

```
## # A tibble: 2 × 4
##   id_cliente item_1 item_2 item_3
##   <dbl> <chr> <chr> <chr>
## 1         1 pao    leite banana
## 2         2 leite papel higienico <NA>
```

## long

mais linhas / observações

```
## # A tibble: 6 × 3
##   id_cliente item_no item
##   <dbl> <chr> <chr>
## 1         1 item_1 pao
## 2         1 item_2 leite
## 3         1 item_3 banana
## 4         2 item_1 leite
## 5         2 item_2 papel higienico
## 6         2 item_3 <NA>
```



# `pivot_longer()`

- data

```
pivot_longer(  
  data,  
  cols,  
  names_to = "name",  
  values_to = "value"  
)
```



# `pivot_longer()`

- `data`
- `cols`: columnas a transpor para formato long

```
pivot_longer(  
  data,  
  cols,  
  names_to = "name",  
  values_to = "value"  
)
```



# `pivot_longer()`

- `data`
- `cols`: colunas a transpor para formato long
- `names_to`: nome da variável que vai receber os nomes das colunas a transpor para long, como valores

```
pivot_longer(  
  data,  
  cols,  
  names_to = "name",  
  values_to = "value"  
)
```



# pivot\_longer()

- data
- cols: colunas a transpor para formato long
- names\_to: nome da variável que vai receber os nomes das colunas a transpor para long, como valores
- values\_to: nome da variável que vai receber os valores atualmente dispersos por várias colunas (string)

```
pivot_longer(  
  data,  
  cols,  
  names_to = "name",  
  values_to = "value"  
)
```



# clientes → compras

```
compras <- clientes %>%  
  pivot_longer(  
    cols = item_1:item_3, # variables item_1 to item_3  
    names_to = "item_no", # column names -> new column called item_no  
    values_to = "item"     # values in columns -> new column called item  
  )
```

compras

```
## # A tibble: 6 × 3  
##   id_cliente item_no item  
##       <dbl> <chr>   <chr>  
## 1         1 item_1   pao  
## 2         1 item_2   leite  
## 3         1 item_3   banana  
## 4         2 item_1   leite  
## 5         2 item_2   papel higienico  
## 6         2 item_3   <NA>
```



# Exemplo da importância de dados tidy

Várias operações de transformação requerem-no (e.g. *join* - prox. aula)

```
precos
```

```
## # A tibble: 5 × 2
##   item           price
##   <chr>         <dbl>
## 1 abacate         2
## 2 banana         0.5
## 3 pao             1.5
## 4 leite           1
## 5 papel higienico 3
```

```
compras %>%
  left_join(precos)
```

```
## # A tibble: 6 × 4
##   id_cliente item_no item           price
##   <dbl> <chr> <chr>         <dbl>
## 1         1 item_1 pao             1.5
## 2         1 item_2 leite           1
## 3         1 item_3 banana         0.5
## 4         2 item_1 leite           1
## 5         2 item_2 papel higienico 3
## 6         2 item_3 <NA>          NA
```



# compras → clientes

- data
- names\_from: variável em formato long a transpor para nomes de novas colunas
- values\_from: variável em formato long que contém valores a dispersar por várias colunas no formato wide

```
compras %>%  
  pivot_wider(  
    names_from = item_no,  
    values_from = item  
  )
```

```
## # A tibble: 2 × 4  
##   id_cliente item_1 item_2 item_3  
##   <dbl> <chr> <chr> <chr>  
## 1         1 pao    leite banana  
## 2         2 leite papel higienico <NA>
```





**Obrigado  
e bom fim-de-semana!**

luis.morais@novasbe.pt