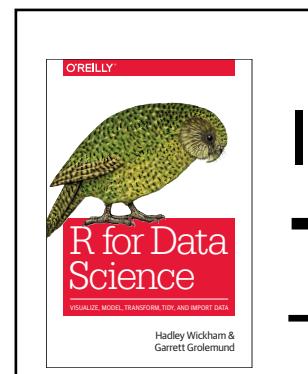
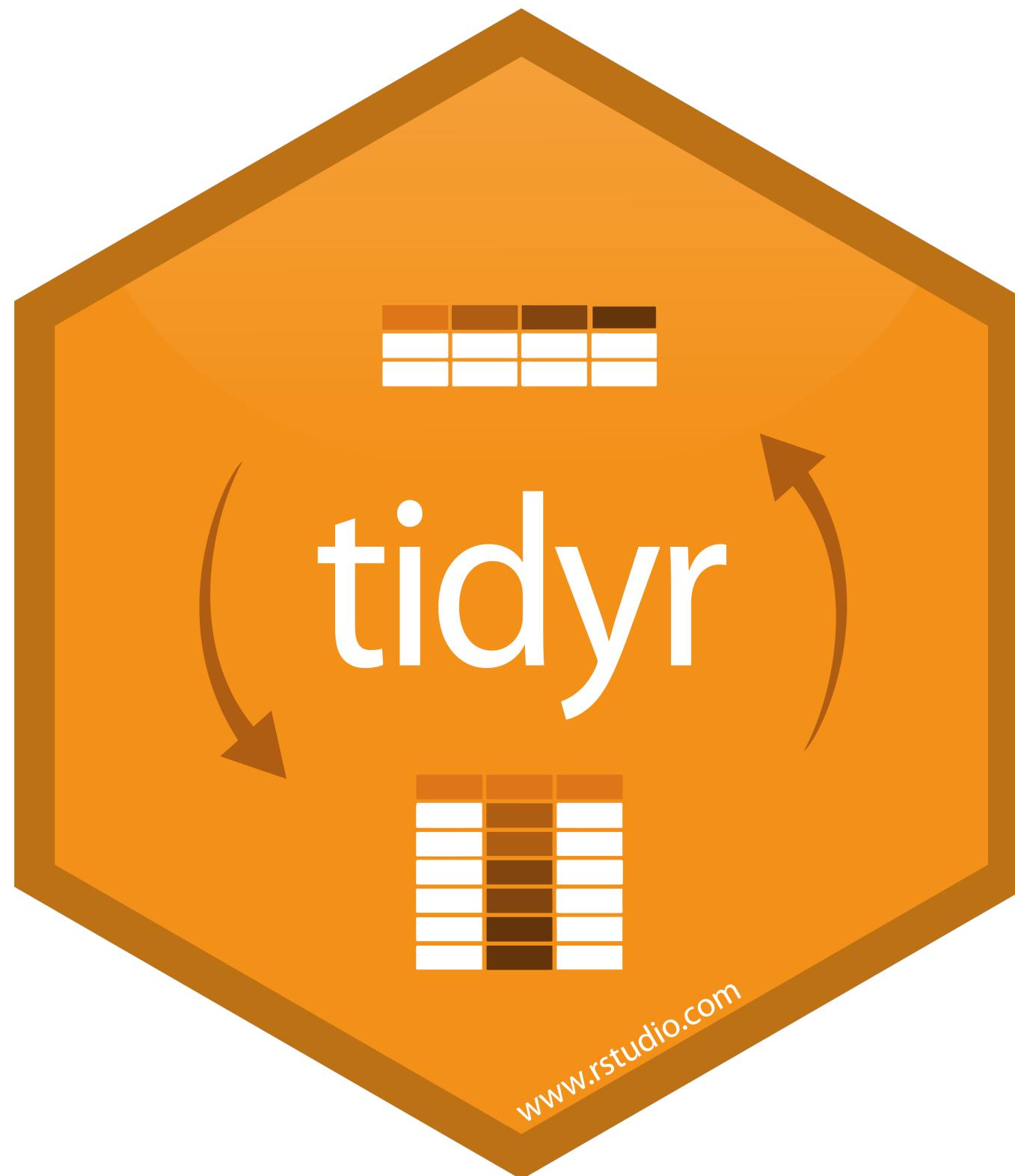


Tidy Data with



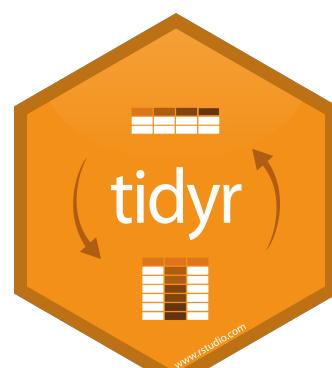
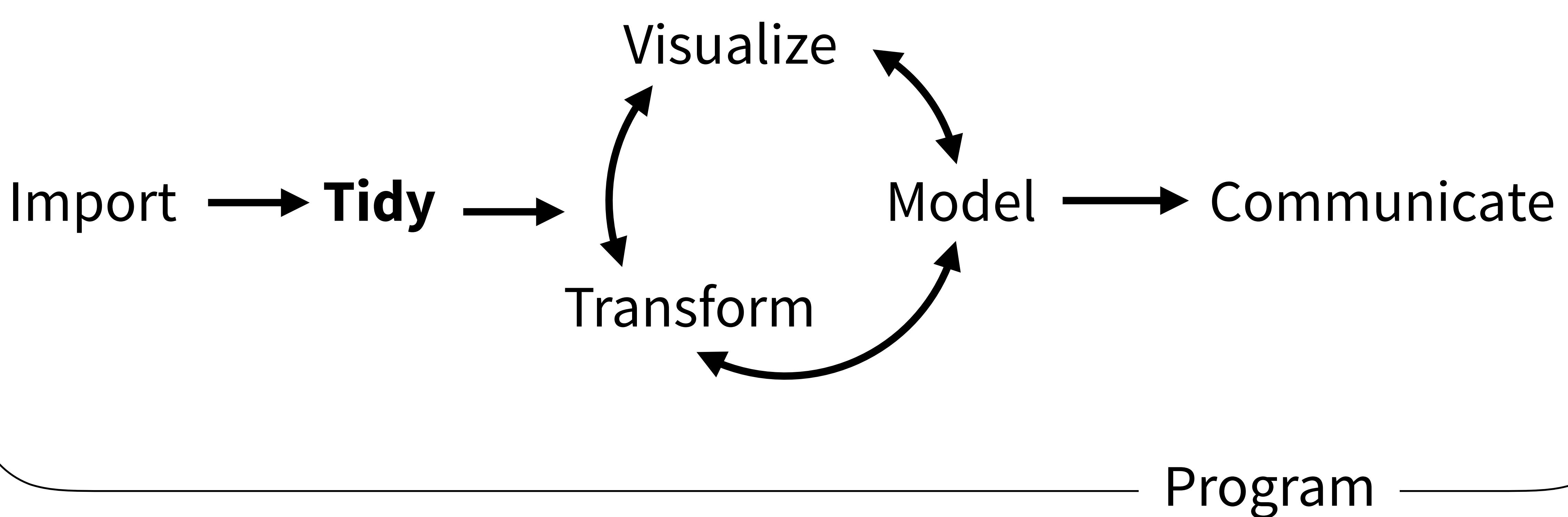
In R4DS
Tidy Data

Your Turn 0

Open 03-Tidy.Rmd

Run the setup chunk

(Applied) Data Science



Tidy tools



Tidy tools

Functions are easiest to use when they are:

1. **Simple** - They do one thing, and they do it well
2. **Composable** - They can be combined with other functions for multi-step operations
3. **Smart** - They can use R objects as input.

Tidy functions do these things in a specific way.



Tidy tools

Functions are easiest to use when they are:

1. **Simple** - They do one thing, and they do it well
2. **Composable** - They can be combined with other functions for multi-step operations
3. **Smart** - They can use R objects as input.

Tidy functions do these things in a specific way.



1. Simple - They do one thing, and they do it well

filter() - extract **cases**

arrange() - reorder **cases**

group_by() - group **cases**

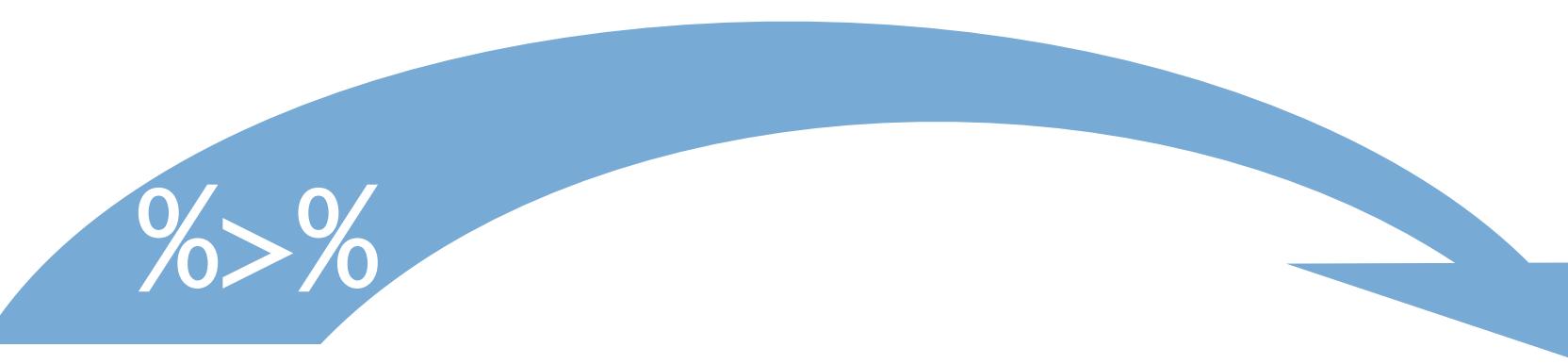
select() - extract **variables**

mutate() - create new **variables**

summarise() - summarise **variables** / create **cases**



2. Composable - They can be combined with other functions for multi-step operations



`babynames` `mutate(_____, percent = prop * 100)`

Each dplyr function takes a data frame as its first argument and returns a data frame. As a result, you can directly pipe the output of one function into the next.



"Data are not just numbers,
they are numbers with a context."

- George Cobb and David Moore (1997)

Consider

What are the variables in this data set?

table1

country <chr>	year <int>	cases <int>	population <int>
Afghanistan	1999	745	19937071
Afghanistan	2000	2666	20505360
Brazil	1999	3737	172006362
Brazil	2000	8088	174504898
China	1999	21258	127295272
China	2000	21366	128048583

6 rows

Consider

What are the variables in this data set?

table2

country	year	type	count
Afghanistan	1999	cases	745
Afghanistan	1999	population	1998701
Afghanistan	2000	cases	2666
Afghanistan	2000	population	2059530
Brazil	1999	cases	7737
Brazil	1999	population	17200632
Brazil	2000	cases	9488
Brazil	2000	population	17450408
China	1999	cases	2258
China	1999	population	127201522

table2 isn't tidy

contains two variables

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

"Data comes in many formats, but R
prefers just one: tidy data."

- Garrett Grolemund

Tidy data

country	year	cases	pop
Afghanistan	1990	745	1012771
Afghanistan	2000	666	2012570
Afghanistan	2010	112	2125222
Afghanistan	2014	113	2135333
Afghanistan	2015	153	2135333
Afghanistan	2016	153	2135333
Afghanistan	2017	128	2135363
Afghanistan	2018	128	2135363
Afghanistan	2019	128	2135363
Afghanistan	2020	43700	2135363
Albania	2014	172	271272
Albania	2015	172	271272
Albania	2016	172	271272
Albania	2017	172	271272
Albania	2018	172	271272
Albania	2019	172	271272
Albania	2020	172	271272
Algeria	2014	112	371272
Algeria	2015	112	371272
Algeria	2016	112	371272
Algeria	2017	112	371272
Algeria	2018	112	371272
Algeria	2019	112	371272
Algeria	2020	112	371272
Angola	2014	112	11272
Angola	2015	112	11272
Angola	2016	112	11272
Angola	2017	112	11272
Angola	2018	112	11272
Angola	2019	112	11272
Angola	2020	112	11272
Antigua and Barbuda	2014	112	11272
Antigua and Barbuda	2015	112	11272
Antigua and Barbuda	2016	112	11272
Antigua and Barbuda	2017	112	11272
Antigua and Barbuda	2018	112	11272
Antigua and Barbuda	2019	112	11272
Antigua and Barbuda	2020	112	11272
Argentina	2014	112	11272
Argentina	2015	112	11272
Argentina	2016	112	11272
Argentina	2017	112	11272
Argentina	2018	112	11272
Argentina	2019	112	11272
Argentina	2020	112	11272
Armenia	2014	112	11272
Armenia	2015	112	11272
Armenia	2016	112	11272
Armenia	2017	112	11272
Armenia	2018	112	11272
Armenia	2019	112	11272
Armenia	2020	112	11272
Aruba	2014	112	11272
Aruba	2015	112	11272
Aruba	2016	112	11272
Aruba	2017	112	11272
Aruba	2018	112	11272
Aruba	2019	112	11272
Aruba	2020	112	11272
Barbados	2014	112	11272
Barbados	2015	112	11272
Barbados	2016	112	11272
Barbados	2017	112	11272
Barbados	2018	112	11272
Barbados	2019	112	11272
Barbados	2020	112	11272
Bolivia	2014	112	11272
Bolivia	2015	112	11272
Bolivia	2016	112	11272
Bolivia	2017	112	11272
Bolivia	2018	112	11272
Bolivia	2019	112	11272
Bolivia	2020	112	11272
Bosnia and Herzegovina	2014	112	11272
Bosnia and Herzegovina	2015	112	11272
Bosnia and Herzegovina	2016	112	11272
Bosnia and Herzegovina	2017	112	11272
Bosnia and Herzegovina	2018	112	11272
Bosnia and Herzegovina	2019	112	11272
Bosnia and Herzegovina	2020	112	11272
Burkina Faso	2014	112	11272
Burkina Faso	2015	112	11272
Burkina Faso	2016	112	11272
Burkina Faso	2017	112	11272
Burkina Faso	2018	112	11272
Burkina Faso	2019	112	11272
Burkina Faso	2020	112	11272
Burundi	2014	112	11272
Burundi	2015	112	11272
Burundi	2016	112	11272
Burundi	2017	112	11272
Burundi	2018	112	11272
Burundi	2019	112	11272
Burundi	2020	112	11272
Cambodia	2014	112	11272
Cambodia	2015	112	11272
Cambodia	2016	112	11272
Cambodia	2017	112	11272
Cambodia	2018	112	11272
Cambodia	2019	112	11272
Cambodia	2020	112	11272
Cameroon	2014	112	11272
Cameroon	2015	112	11272
Cameroon	2016	112	11272
Cameroon	2017	112	11272
Cameroon	2018	112	11272
Cameroon	2019	112	11272
Cameroon	2020	112	11272
Central African Republic	2014	112	11272
Central African Republic	2015	112	11272
Central African Republic	2016	112	11272
Central African Republic	2017	112	11272
Central African Republic	2018	112	11272
Central African Republic	2019	112	11272
Central African Republic	2020	112	11272
Chad	2014	112	11272
Chad	2015	112	11272
Chad	2016	112	11272
Chad	2017	112	11272
Chad	2018	112	11272
Chad	2019	112	11272
Chad	2020	112	11272
Comoros	2014	112	11272
Comoros	2015	112	11272
Comoros	2016	112	11272
Comoros	2017	112	11272
Comoros	2018	112	11272
Comoros	2019	112	11272
Comoros	2020	112	11272
Congo	2014	112	11272
Congo	2015	112	11272
Congo	2016	112	11272
Congo	2017	112	11272
Congo	2018	112	11272
Congo	2019	112	11272
Congo	2020	112	11272
Cote d'Ivoire	2014	112	11272
Cote d'Ivoire	2015	112	11272
Cote d'Ivoire	2016	112	11272
Cote d'Ivoire	2017	112	11272
Cote d'Ivoire	2018	112	11272
Cote d'Ivoire	2019	112	11272
Cote d'Ivoire	2020	112	11272
Dominican Republic	2014	112	11272
Dominican Republic	2015	112	11272
Dominican Republic	2016	112	11272
Dominican Republic	2017	112	11272
Dominican Republic	2018	112	11272
Dominican Republic	2019	112	11272
Dominican Republic	2020	112	11272
Ecuador	2014	112	11272
Ecuador	2015	112	11272
Ecuador	2016	112	11272
Ecuador	2017	112	11272
Ecuador	2018	112	11272
Ecuador	2019	112	11272
Ecuador	2020	112	11272
Egypt	2014	112	11272
Egypt	2015	112	11272
Egypt	2016	112	11272
Egypt	2017	112	11272
Egypt	2018	112	11272
Egypt	2019	112	11272
Egypt	2020	112	11272
El Salvador	2014	112	11272
El Salvador	2015	112	11272
El Salvador	2016	112	11272
El Salvador	2017	112	11272
El Salvador	2018	112	11272
El Salvador	2019	112	11272
El Salvador	2020	112	11272
Greece	2014	112	11272
Greece	2015	112	11272
Greece	2016	112	11272
Greece	2017	112	11272
Greece	2018	112	11272
Greece	2019	112	11272
Greece	2020	112	11272
Honduras	2014	112	11272
Honduras	2015	112	11272
Honduras	2016	112	11272
Honduras	2017	112	11272
Honduras	2018	112	11272
Honduras	2019	112	11272
Honduras	2020	112	11272
Iceland	2014	112	11272
Iceland	2015	112	11272
Iceland	2016	112	11272
Iceland	2017	112	11272
Iceland	2018	112	11272
Iceland	2019	112	11272

Your Turn 1

Is bp_systolic tidy? What are the variables?



Your Turn 1

Is bp_systolic tidy? What are the variables?

subject_id <dbl>	time_1 <dbl>	time_2 <dbl>	time_3 <dbl>
1	120	118	121
2	125	131	NA
3	141	NA	NA

3 rows

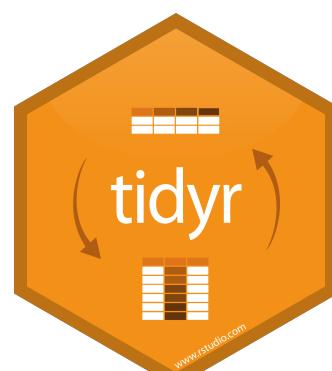
Variables:

- subject
- time
- systolic blood pressure

bp_systolic2 is tidy

subject_id <dbl>	time <dbl>	systolic <dbl>
1	1	120
1	2	118
1	3	121
2	1	125
2	2	131
3	1	141

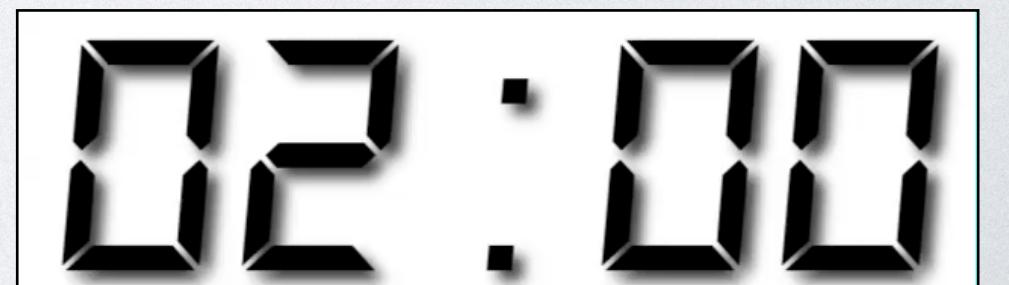
6 rows



Your Turn 2

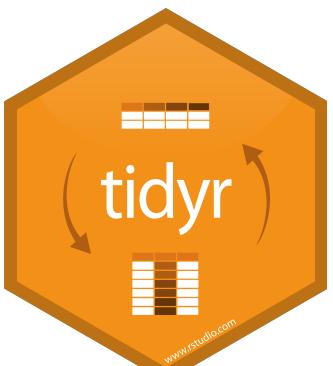
Using `bp_systolic2` with `group_by()` and `summarise()`:

- Find the average systolic blood pressure for each subject
- Find the last time each subject was measured



```
bp_systolic2 %>%  
  group_by(subject_id) %>%  
  summarise(avg_sys = mean(systolic),  
            last_measurement = max(time))
```

```
# A tibble: 3 x 3  
  subject_id avg_sys last_measurement  
        <dbl>     <dbl>             <dbl>  
1           1    120.                 3  
2           2    128                  2  
3           3    141                  1
```



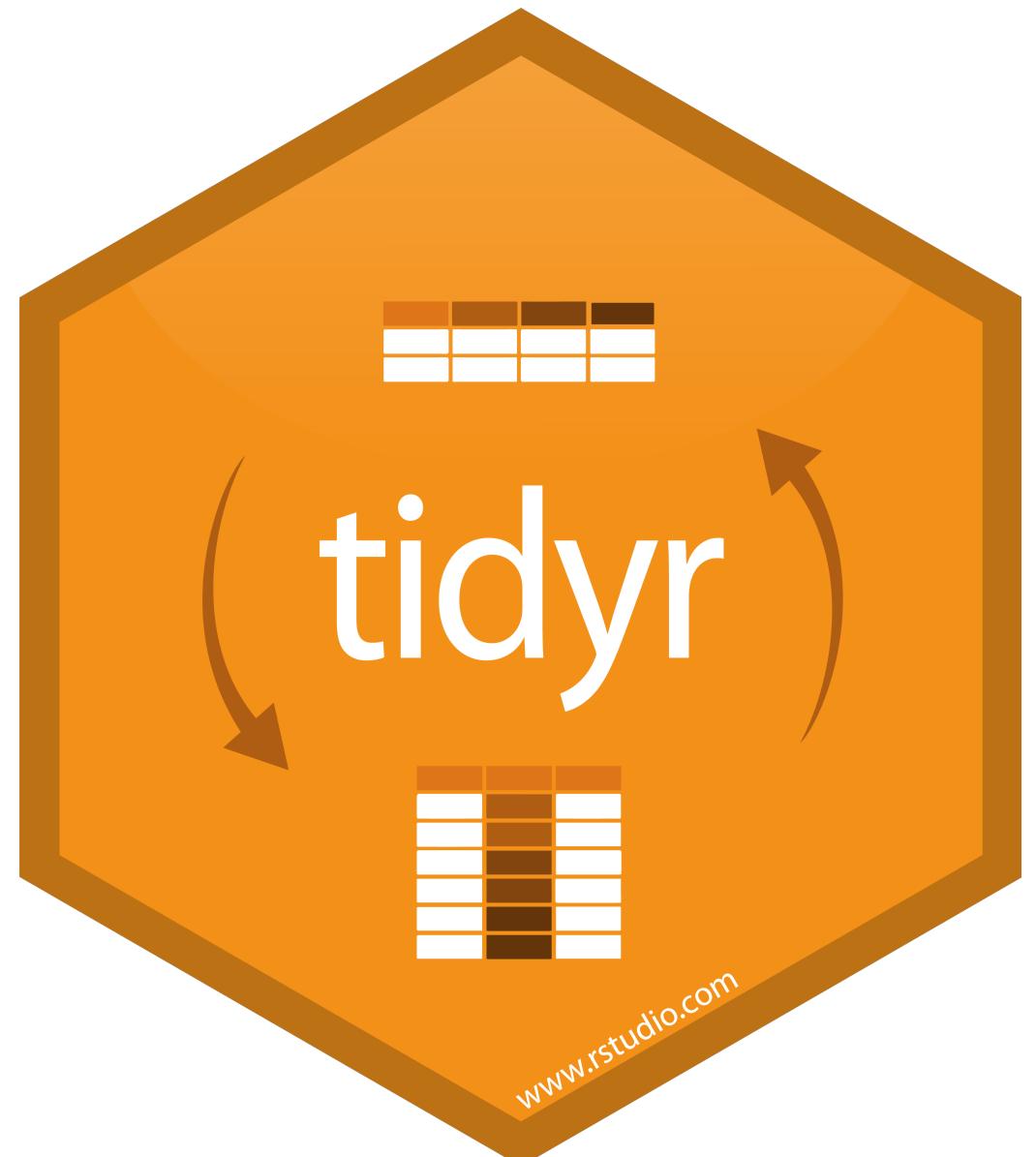
"Tidy data sets are all alike; but
every messy data set is messy in its
own way."

- Hadley Wickham

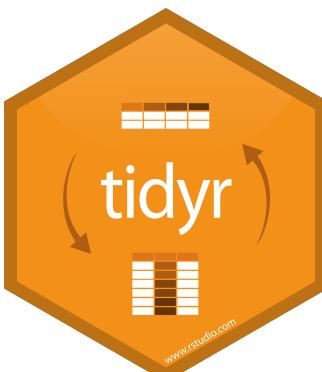
tidyR



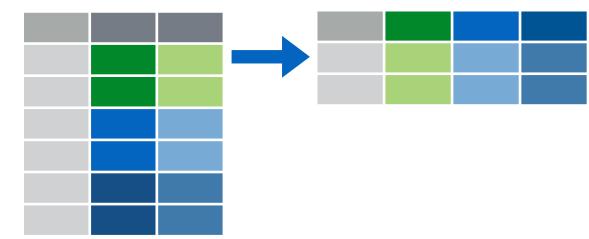
tidyr



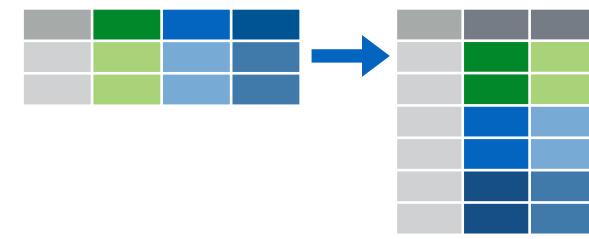
A package that reshapes the layout of tabular data.



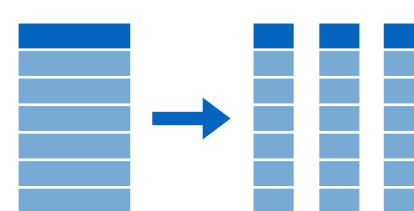
tidyr verbs



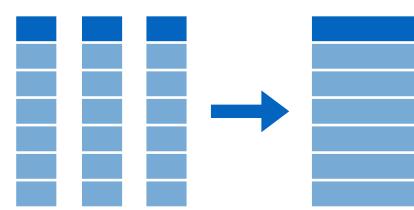
Make data wider with **pivot_wider()**



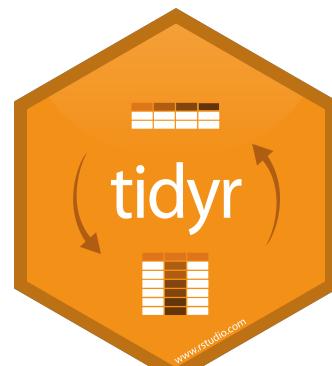
Make data longer with **pivot_longer()**



Split a column with **separate()** or
separate_rows()



Unite columns with **unite()**



pivot_longer()

A faint watermark of the R logo is visible in the bottom right corner, consisting of a circular arrow with the letter 'R' inside.

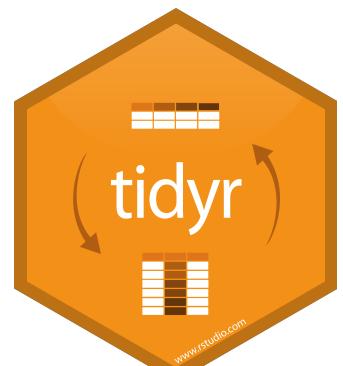
Toy data

The screenshot shows an RStudio interface with an R Markdown file titled "03-Tidy-Data.Rmd". The code editor contains the following R code:

```
1 ---  
2 title: "Tidy Data"  
3 output: html_notebook  
4 ---  
5  
6 ```{r setup}  
7 library(tidyverse)  
8 library(babynames)  
9  
10 # Toy data  
11 cases <- tribble(  
12   ~Country, ~"2011", ~"2012", ~"2013",  
13   "FR",      7000,     6900,     7000,  
14   "DE",      5800,     6000,     6200,  
15   "US",     15000,    14000,    13000  
16 )  
17  
18 pollution <- tribble(  
19   ~city, ~size, ~amount,  
20   "New York", "large",    23,  
21   "New York", "small",    14,  
22   "London",   "large",    22,  
23   "London",   "small",    16,  
24   "Beijing",  "large",   121,  
25   "Beijing",  "small",   121  
26 )  
27  
28 x <- tribble(  
29   ~x1, ~x2,  
30   "A",    1,  
31   "B",    NA,  
32   "C",    NA,  
33   "D",    3,  
34   "E",    NA  
35 )
```

The output pane displays the generated R code for creating the "cases" data frame:

```
cases <- tribble(  
  ~Country, ~"2011", ~"2012", ~"2013",  
  "FR",      7000,     6900,     7000,  
  "DE",      5800,     6000,     6200,  
  "US",     15000,    14000,    13000
```



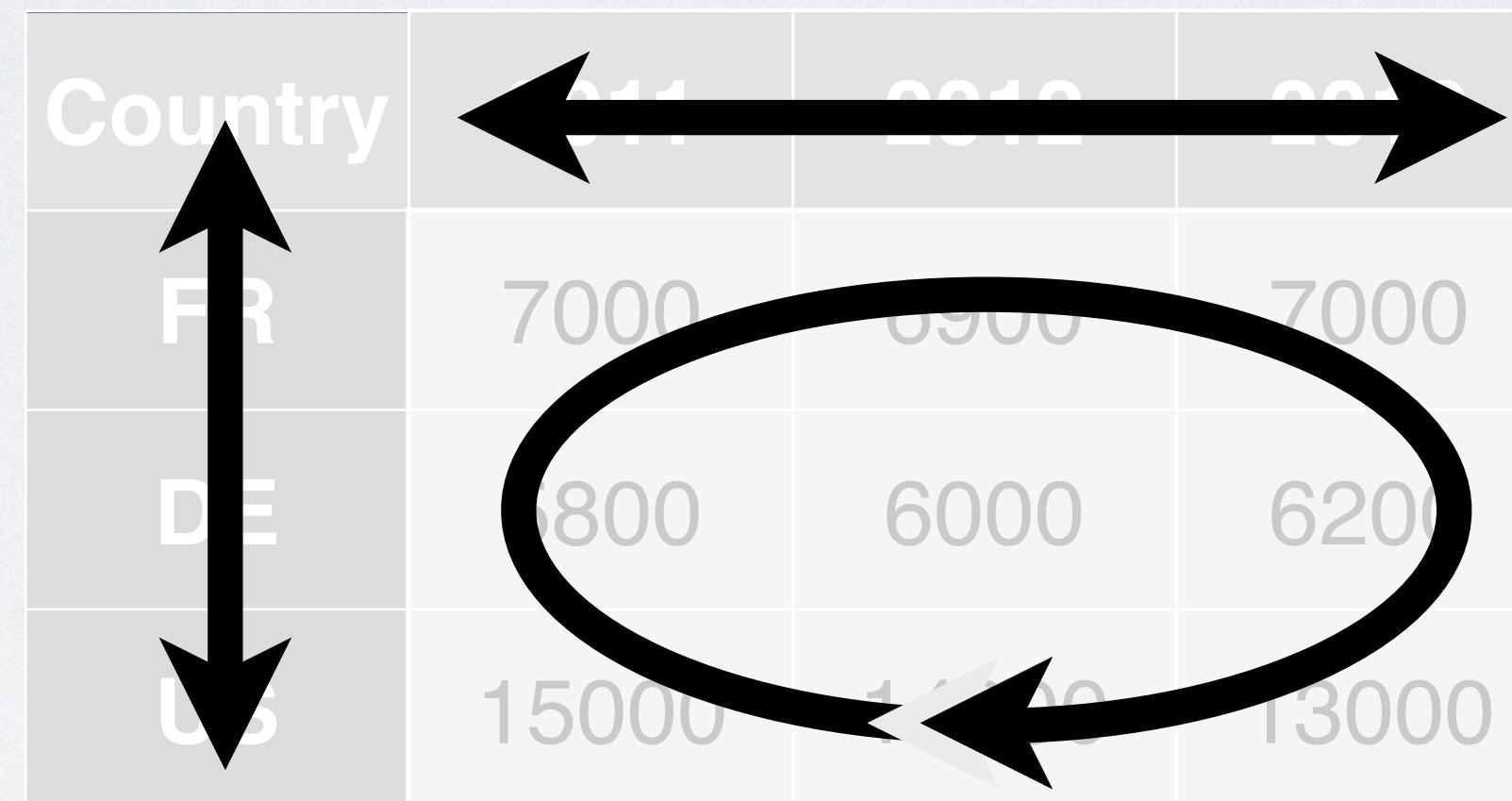
Consider

What are the variables in cases?

Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

Consider

What are the variables in cases?



- Country
- Year
- Count

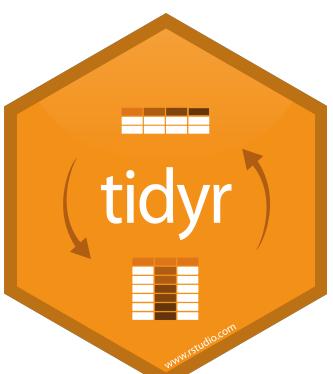
Your Turn 3

On a sheet of paper, draw how the cases data set would look if it had the same values grouped into three columns: *country, year, n*

Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

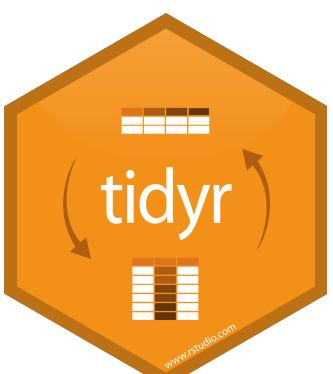


Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000



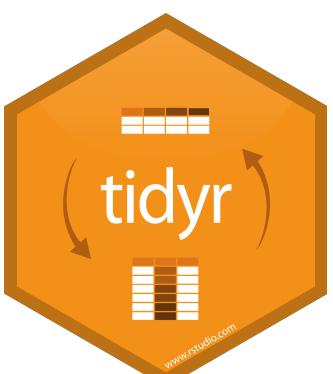
Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

Country	Year	n
---------	------	---



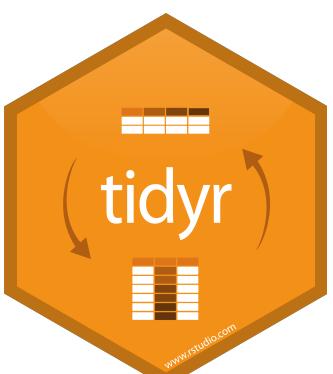
Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

Country	Year	n
FR	2011	7000



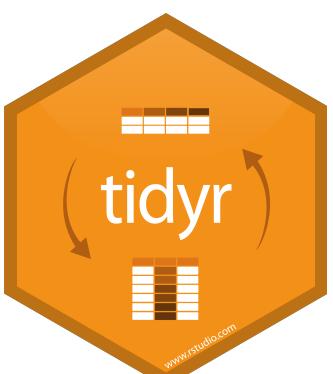
Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

Country	Year	n
FR	2011	7000
DE	2011	5800



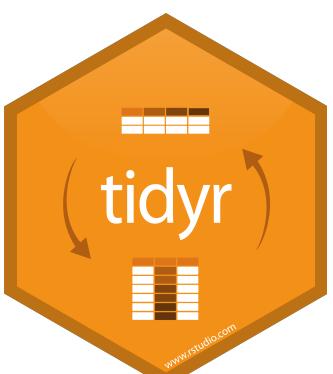
Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

Country	Year	n
FR	2011	7000
DE	2011	5800
US	2011	15000



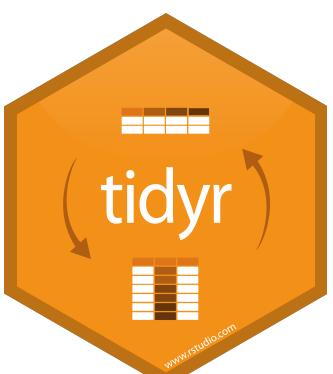
Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

Country	Year	n
FR	2011	7000
DE	2011	5800
US	2011	15000
FR	2012	6900



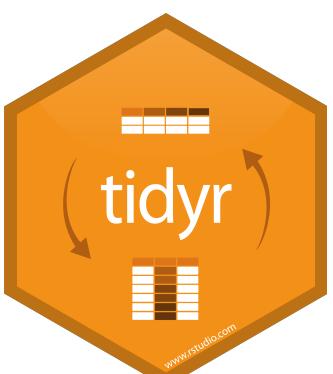
Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

Country	Year	n
FR	2011	7000
DE	2011	5800
US	2011	15000
FR	2012	6900
DE	2012	6000



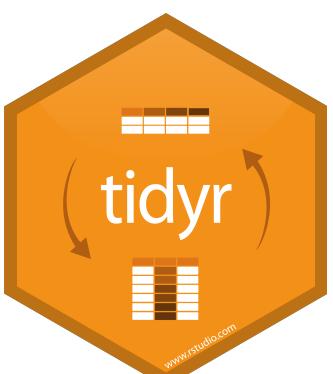
Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

Country	Year	n
FR	2011	7000
DE	2011	5800
US	2011	15000
FR	2012	6900
DE	2012	6000
US	2012	14000



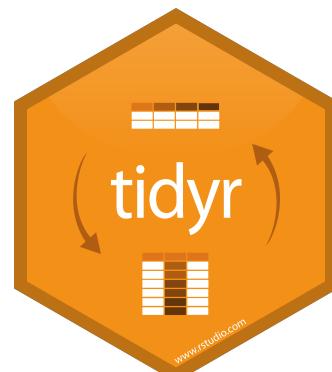
Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

Country	Year	n
FR	2011	7000
DE	2011	5800
US	2011	15000
FR	2012	6900
DE	2012	6000
US	2012	14000
FR	2013	7000



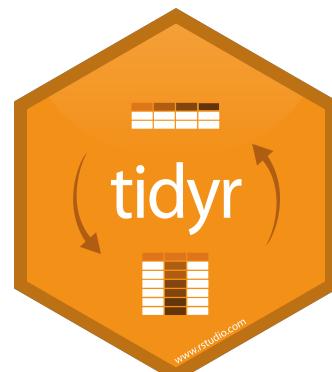
Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

Country	Year	n
FR	2011	7000
DE	2011	5800
US	2011	15000
FR	2012	6900
DE	2012	6000
US	2012	14000
FR	2013	7000
DE	2013	6200

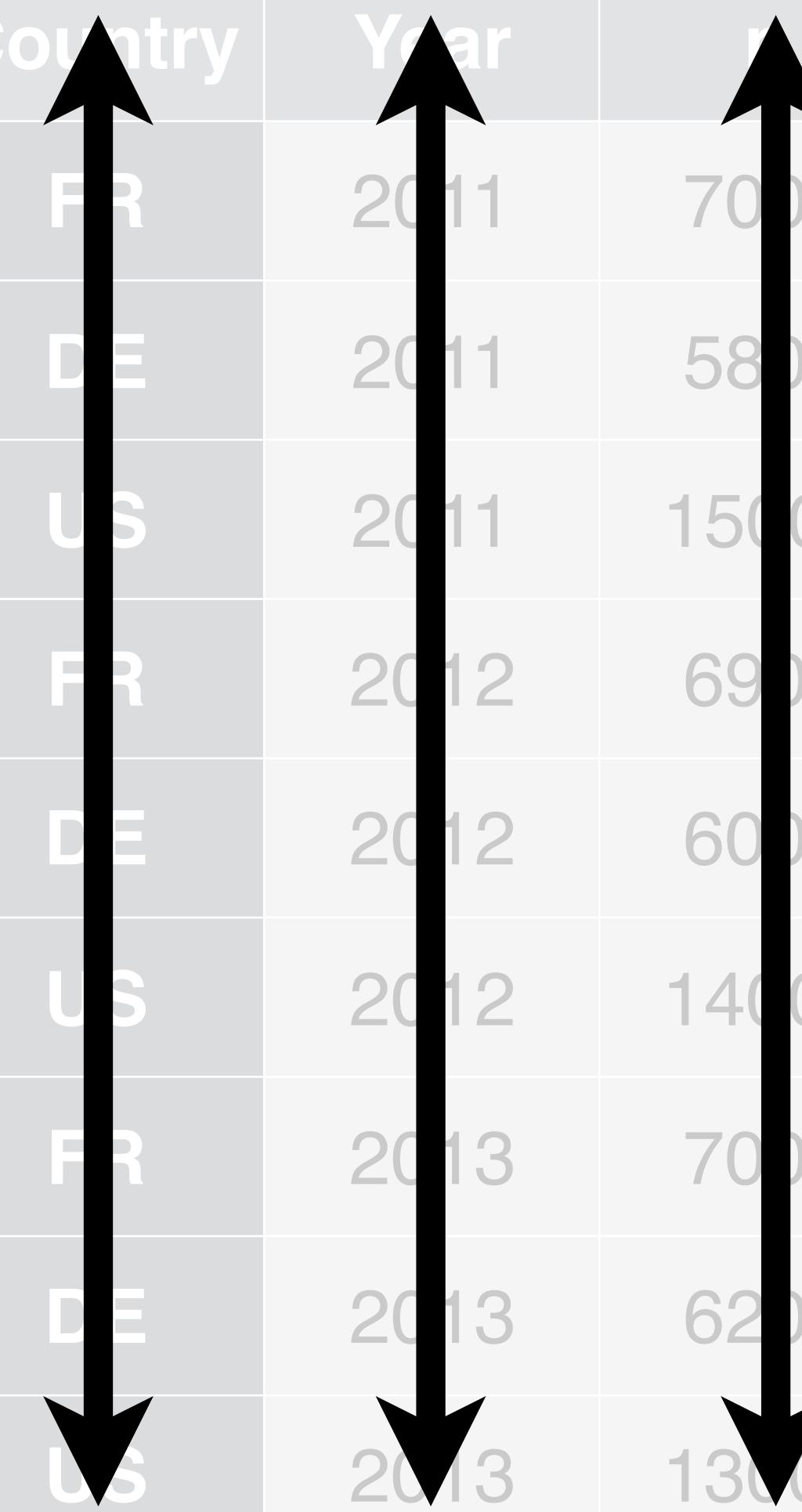


Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

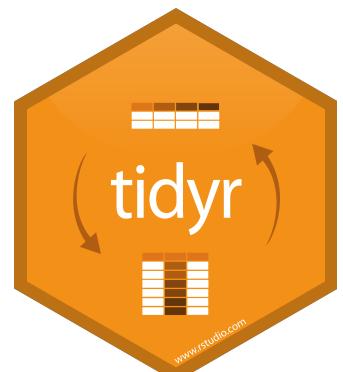
Country	Year	n
FR	2011	7000
DE	2011	5800
US	2011	15000
FR	2012	6900
DE	2012	6000
US	2012	14000
FR	2013	7000
DE	2013	6200
US	2013	13000



Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000



Country	Year	Value
FR	2011	7000
DE	2011	5800
US	2011	15000
FR	2012	6900
DE	2012	6000
US	2012	14000
FR	2013	7000
DE	2013	6200
US	2013	13000

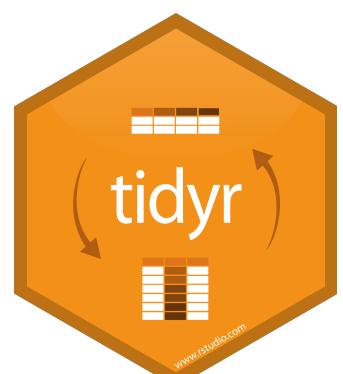


Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000



gather()

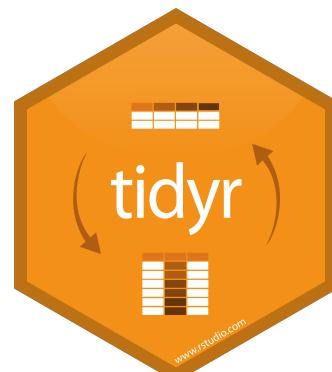
Country	Year	n
FR	2011	7000
DE	2011	5800
US	2011	15000
FR	2012	6900
DE	2012	6000
US	2012	14000
FR	2013	7000
DE	2013	6200
US	2013	13000



1 2

Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

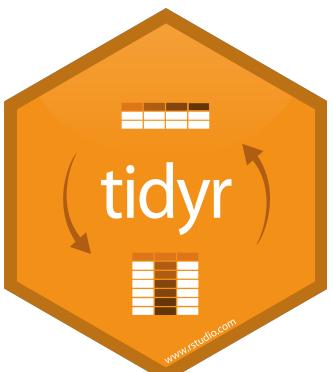
Country	Year	n
FR	2011	7000
DE	2011	5800
US	2011	15000
FR	2012	6900
DE	2012	6000
US	2012	14000
FR	2013	7000
DE	2013	6200
US	2013	13000



key (former column names)

Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

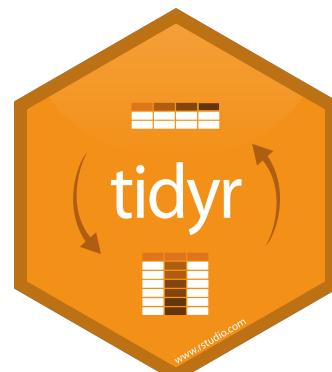
Country	Year	n
FR	2011	7000
DE	2011	5800
US	2011	15000
FR	2012	6900
DE	2012	6000
US	2012	14000
FR	2013	7000
DE	2013	6200
US	2013	13000



key value (former cells)

Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

Country	Year	n
FR	2011	7000
DE	2011	5800
US	2011	15000
FR	2012	6900
DE	2012	6000
US	2012	14000
FR	2013	7000
DE	2013	6200
US	2013	13000



pivot_longer()

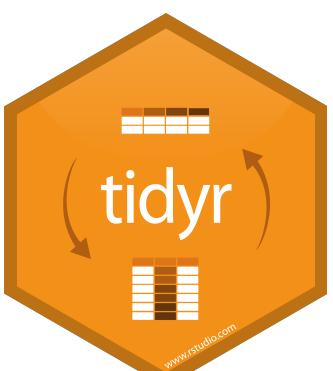
```
cases %>%  
  pivot_longer(2:4, names_to = "year", values_to = "n")
```

**data frame to
reshape**

**numeric
indexes of
columns to
collapse
(or names)**

**name of the
new variable
created from
column names
(a character
string)**

**name of the new
variable created
from cell values
(a character string)**

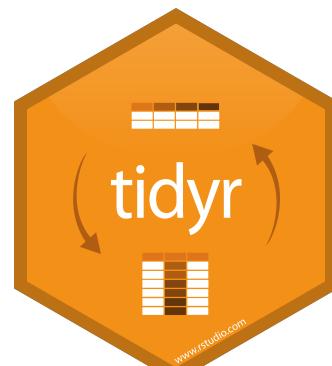


pivot_longer()

```
cases %>%  
  pivot_longer(2:4, names_to = "year", values_to = "count")
```

numeric
indexes

Country	2	3	4
	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

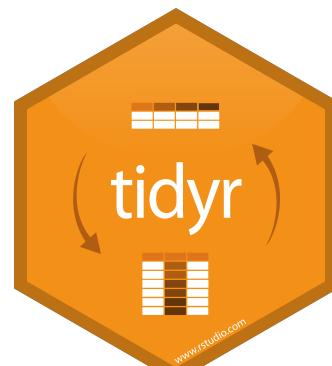


pivot_longer()

names

```
cases %>%  
  pivot_longer(cols=c("2011","2012","2013"),  
               names_to = "year", values_to = "count")
```

Country	2011	2012	2013
	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000



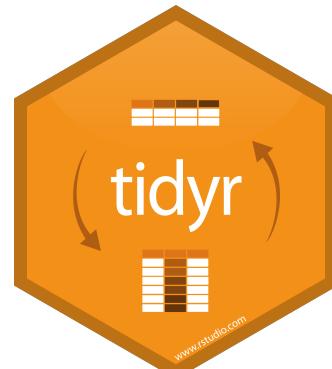
pivot_longer()

```
cases %>%  
  pivot_longer(-Country, names_to = "year", values_to = "count")
```

Everything
except...

Not Country Not Country Not Country

Country <code><chr></code>	2011 <code><dbl></code>	2012 <code><dbl></code>	2013 <code><dbl></code>
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000



Your Turn 4

recall

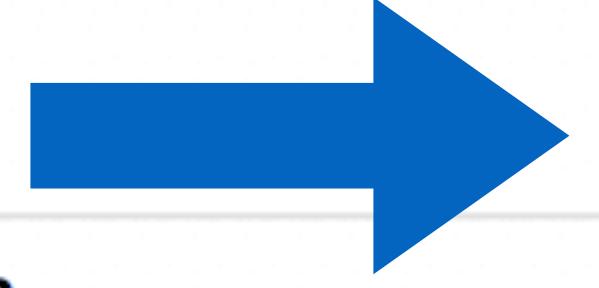
```
cases %>%  
  pivot_longer(-Country, names_to = "year", values_to = "count")
```

Use **pivot_longer()** to reorganize **table4a** into three columns: *country*, *year*, and *cases*.

	country <chr>	1999 <int>	2000 <int>
1	Afghanistan	745	2666
2	Brazil	37737	80488
3	China	212258	213766
3 rows			

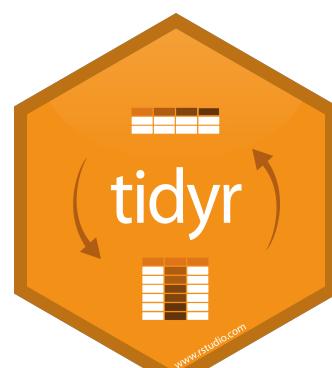


```
table4a %>%  
  gather(key = "year", value = "n", 2:3)
```



country	year	n
Afghanistan	1999	745
Brazil	1999	37737
China	1999	212258
Afghanistan	2000	2666
Brazil	2000	80488
China	2000	213766

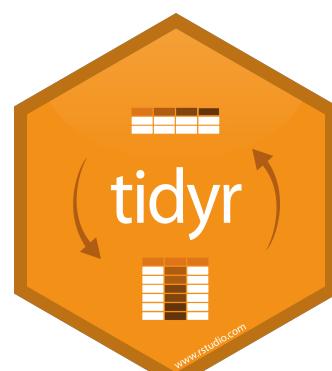
6 rows



```
table4a %>%  
  gather(key = "year", value = "n", 2:3, convert = TRUE)
```

country	year	n
Afghanistan	1999	745
Brazil	1999	37737
China	1999	212258
Afghanistan	2000	2666
Brazil	2000	80488
China	2000	213766

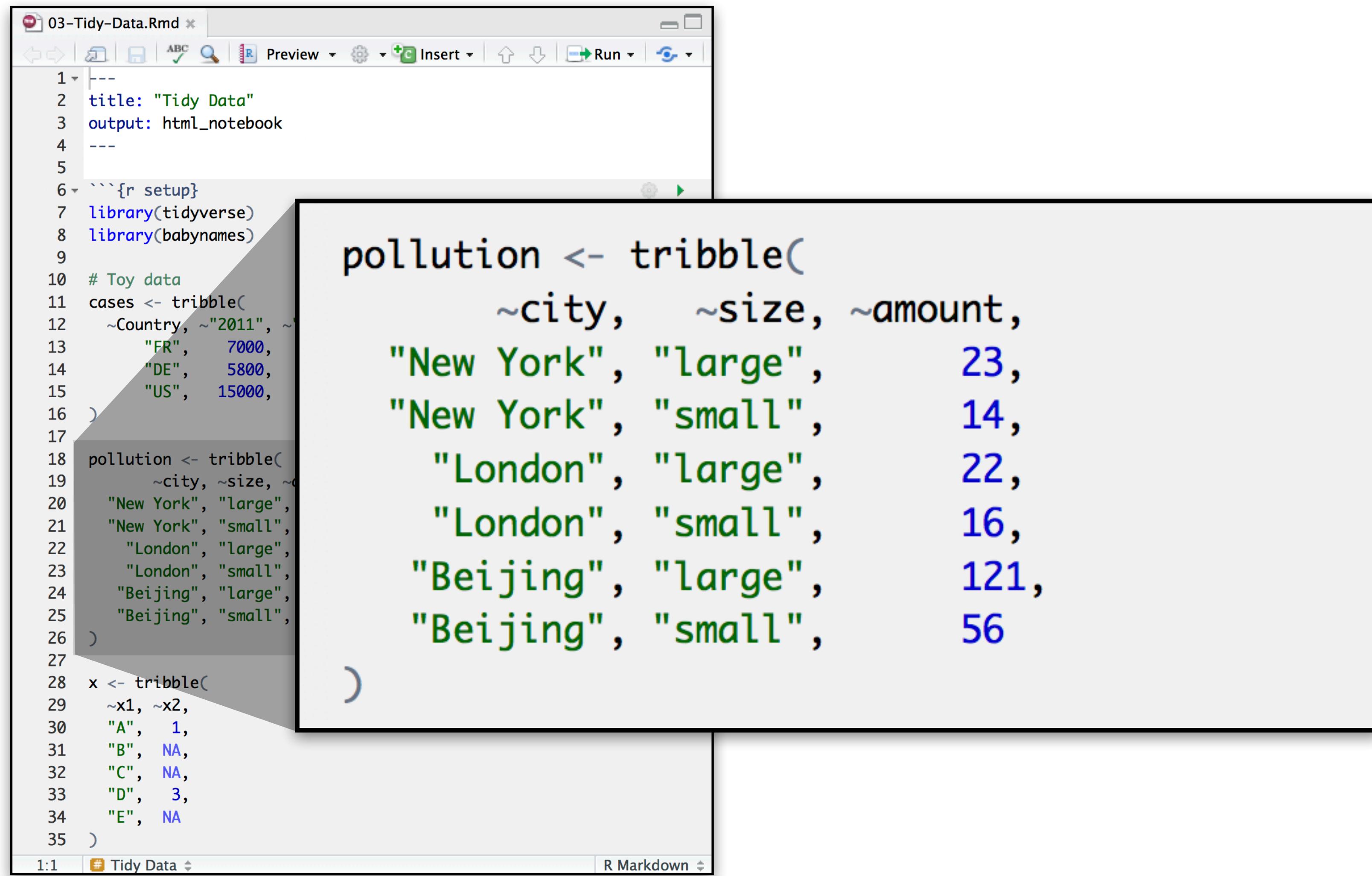
6 rows



pivot_wider()

A faint watermark of the R logo is visible in the bottom right corner, consisting of a circular arrow with the letter 'R' inside.

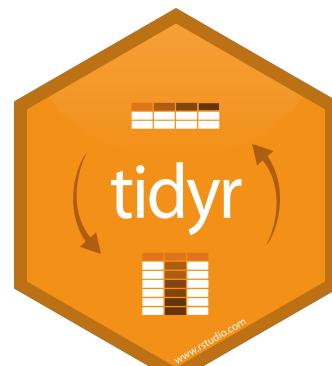
Toy data



The screenshot shows an RStudio interface with an R Markdown file titled "03-Tidy-Data.Rmd". The code uses the tidyverse and babynames packages to generate toy data. A callout box highlights the creation of a 'pollution' tibble.

```
1 ---  
2 title: "Tidy Data"  
3 output: html_notebook  
4 ---  
5  
6 ```{r setup}  
7 library(tidyverse)  
8 library(babynames)  
9  
10 # Toy data  
11 cases <- tribble(  
12   ~Country, ~"2011", ~  
13   "FR",    7000,  
14   "DE",    5800,  
15   "US",   15000,  
16 )  
17  
18 pollution <- tribble(  
19   ~city,   ~size, ~amount,  
20   "New York", "large", 23,  
21   "New York", "small", 14,  
22   "London",  "large", 22,  
23   "London",  "small", 16,  
24   "Beijing", "large", 121,  
25   "Beijing", "small", 56  
26 )  
27  
28 x <- tribble(  
29   ~x1, ~x2,  
30   "A",  1,  
31   "B",  NA,  
32   "C",  NA,  
33   "D",  3,  
34   "E",  NA  
35 )
```

pollution <- tribble(
 ~city, ~size, ~amount,
 "New York", "large", 23,
 "New York", "small", 14,
 "London", "large", 22,
 "London", "small", 16,
 "Beijing", "large", 121,
 "Beijing", "small", 56
)



Consider

What are the variables in pollution?

city	particle size	amount ($\mu\text{g}/\text{m}^3$)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

Consider

What are the variables in pollution?

city	particle size	amount ($\mu\text{g}/\text{m}^3$)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

- City
- Amount of large particulate
- Amount of small particulate

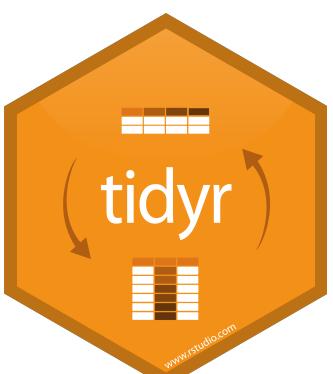
Your Turn 5

On a sheet of paper, draw how this data set would look if it had the same values grouped into three columns: *city, large, small*

city	size	amount
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

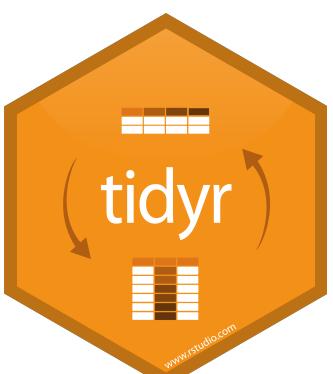


city	size	amount
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56



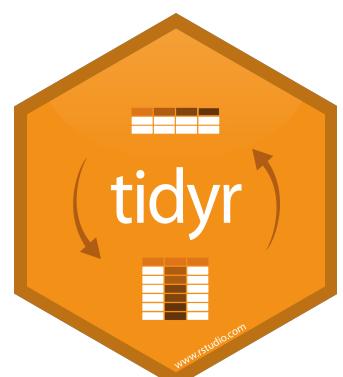
city	size	amount
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

city	large	small
New York	23	14



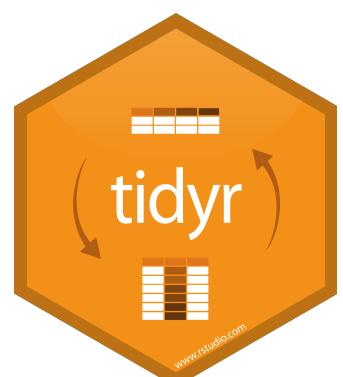
city	size	amount
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

city	large	small
New York	23	



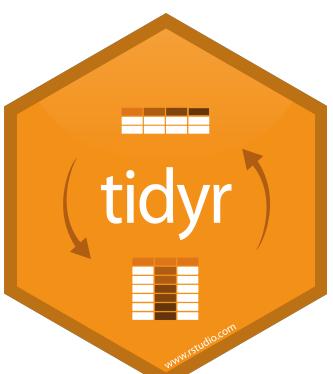
city	size	amount
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

city	large	small
New York	23	14



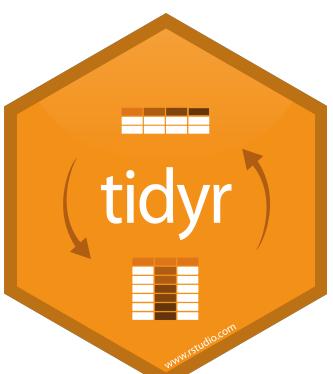
city	size	amount
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

city	large	small
New York	23	14
London	22	



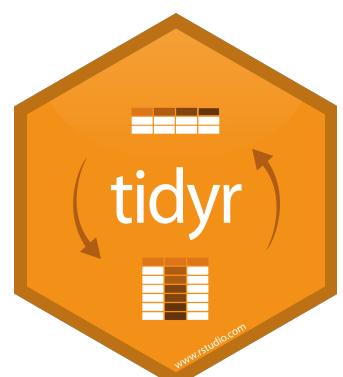
city	size	amount
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

city	large	small
New York	23	14
London	22	16



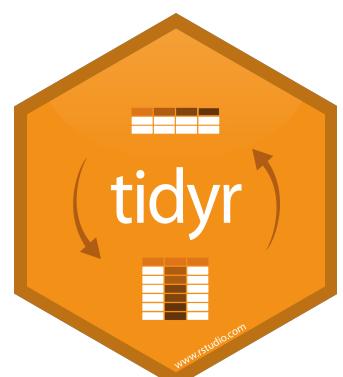
city	size	amount
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

city	large	small
New York	23	14
London	22	16
Beijing	121	



city	size	amount
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

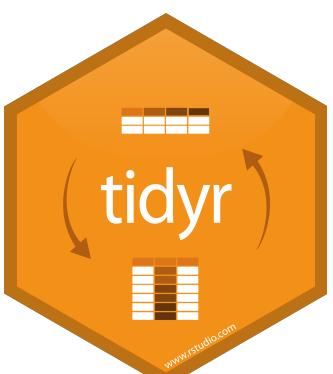
city	large	small
New York	23	14
London	22	16
Beijing	121	56



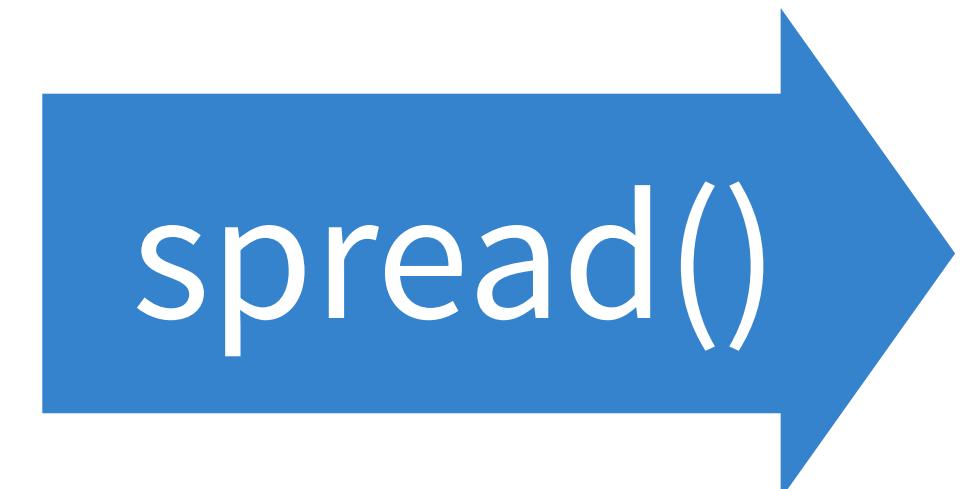
city	size	amount
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56



city	large	small
New York	23	14
London	22	16
Beijing	121	56

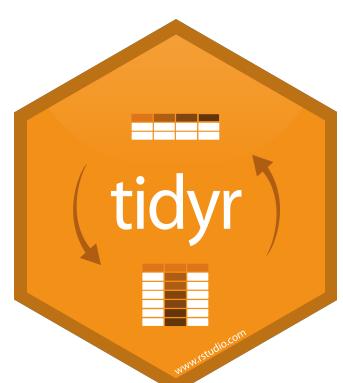


city	size	amount
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56



spread()

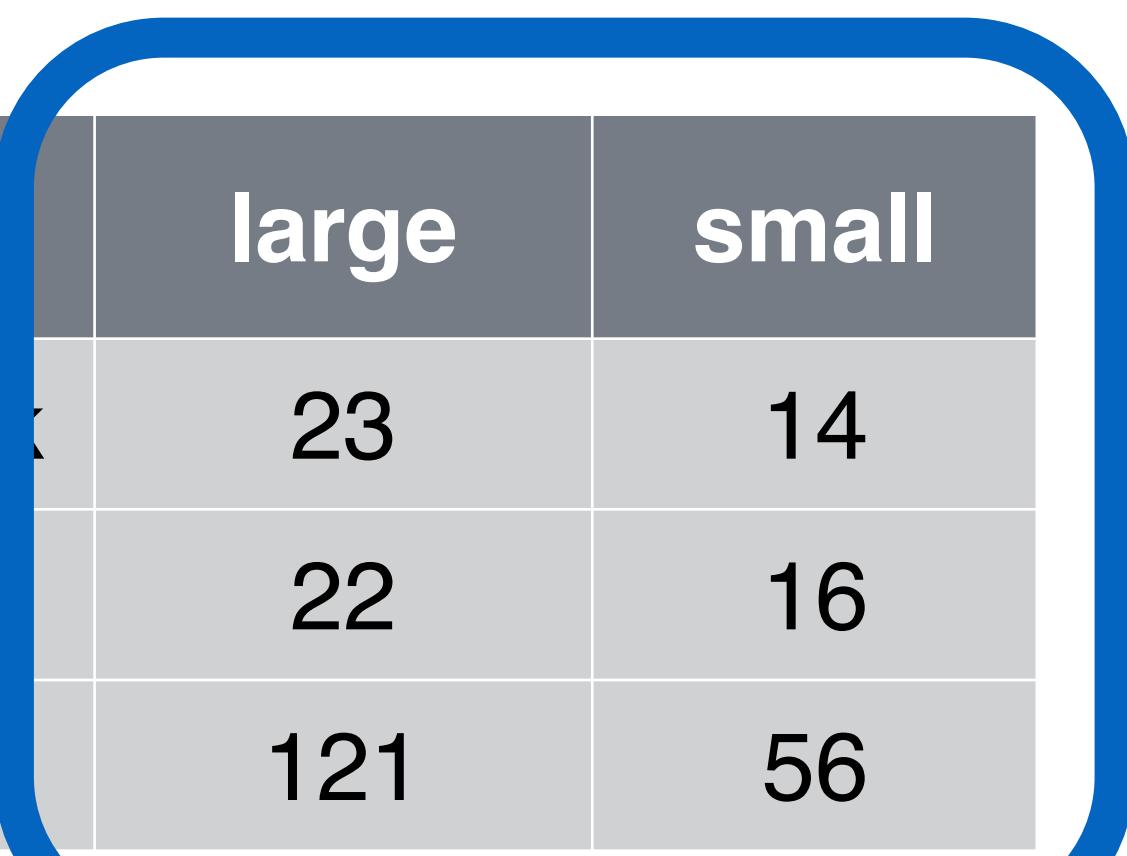
city	large	small
New York	23	14
London	22	16
Beijing	121	56



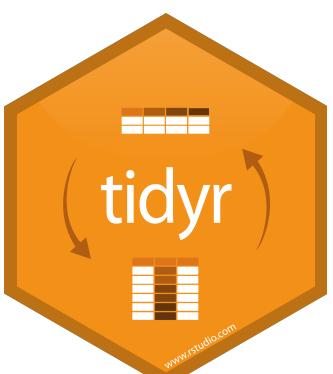
1

2

city	size	amount
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56



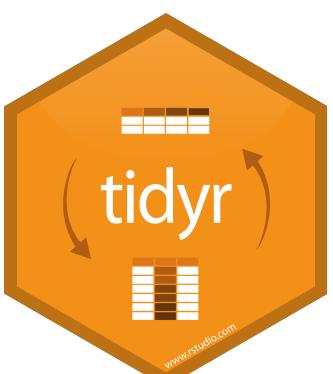
city	large	small
New York	23	14
London	22	16
Beijing	121	56



key (new column names)

city	size	amount
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

city	large	small
New York	23	14
London	22	16
Beijing	121	56

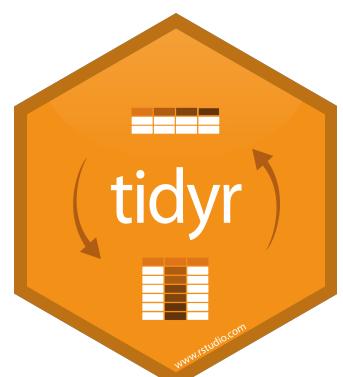


key

value (new cells)

city	size	amount
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

city	large	small
New York	23	14
London	22	16
Beijing	121	56



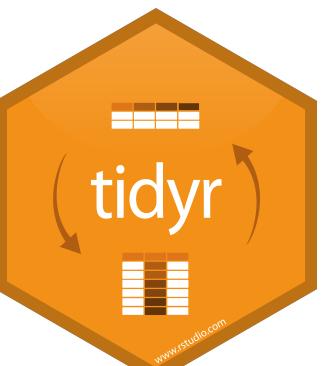
`pivot_wider()`

```
pollution %>% pivot_wider(names_from = size, values_from = amount)
```

**data frame to
reshape**

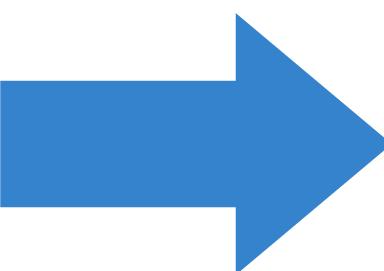
**new columns from
names in this variable**

**values for these columns
from this variable**

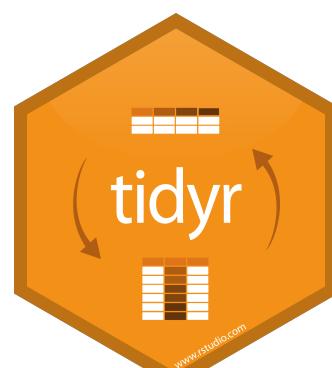


```
pollution %>% pivot_wider(names_from = size,  
                           values_from = amount)
```

	city	size	amount
1	New York	large	23
2	New York	small	14
3	London	large	22
4	London	small	16
5	Beijing	large	121
6	Beijing	small	56



	city	large	small
1	Beijing	121	56
2	London	22	16
3	New York	23	14



Your Turn 6

```
pollution %>% pivot_wider(names_from = size,  
                           values_from = amount)
```

Use **pivot_wider()** to reorganize **table2** into four columns: *country*, *year*, *cases*, and *population*.

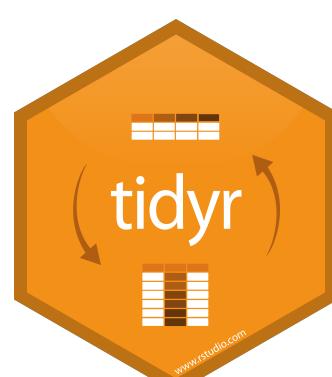
country	year	type	count
<chr>	<int>	<chr>	<int>
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362



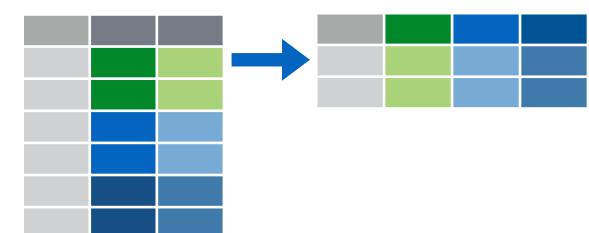
```
table2 %>%  
  pivot_wider(names_from = type, values_from = count)
```

	country	year	cases	population
	<chr>	<int>	<int>	<int>
1	Afghanistan	1999	745	19987071
2	Afghanistan	2000	2666	20595360
3	Brazil	1999	37737	172006362
4	Brazil	2000	80488	174504898
5	China	1999	212258	1272915272
6	China	2000	213766	1280428583

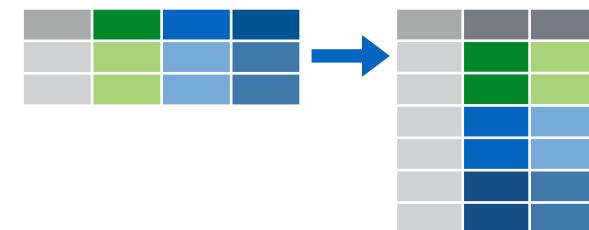
6 rows



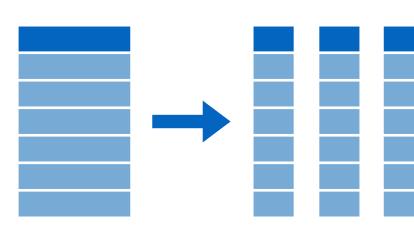
tidyverse verbs



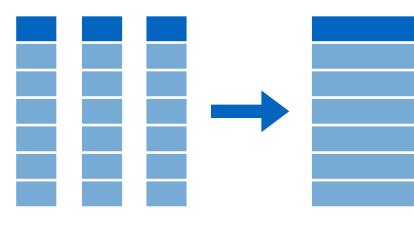
Make data wider with **pivot_wider()**



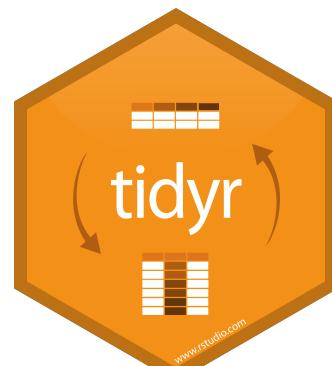
Make data longer with **pivot_longer()**



Split a column with **separate()** or
separate_rows()



Unite columns with **unite()**



Tidy Data with

