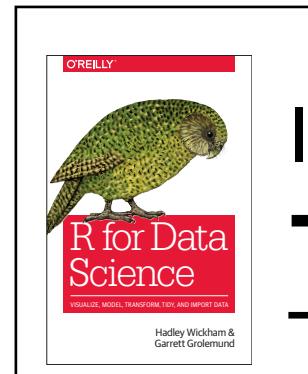
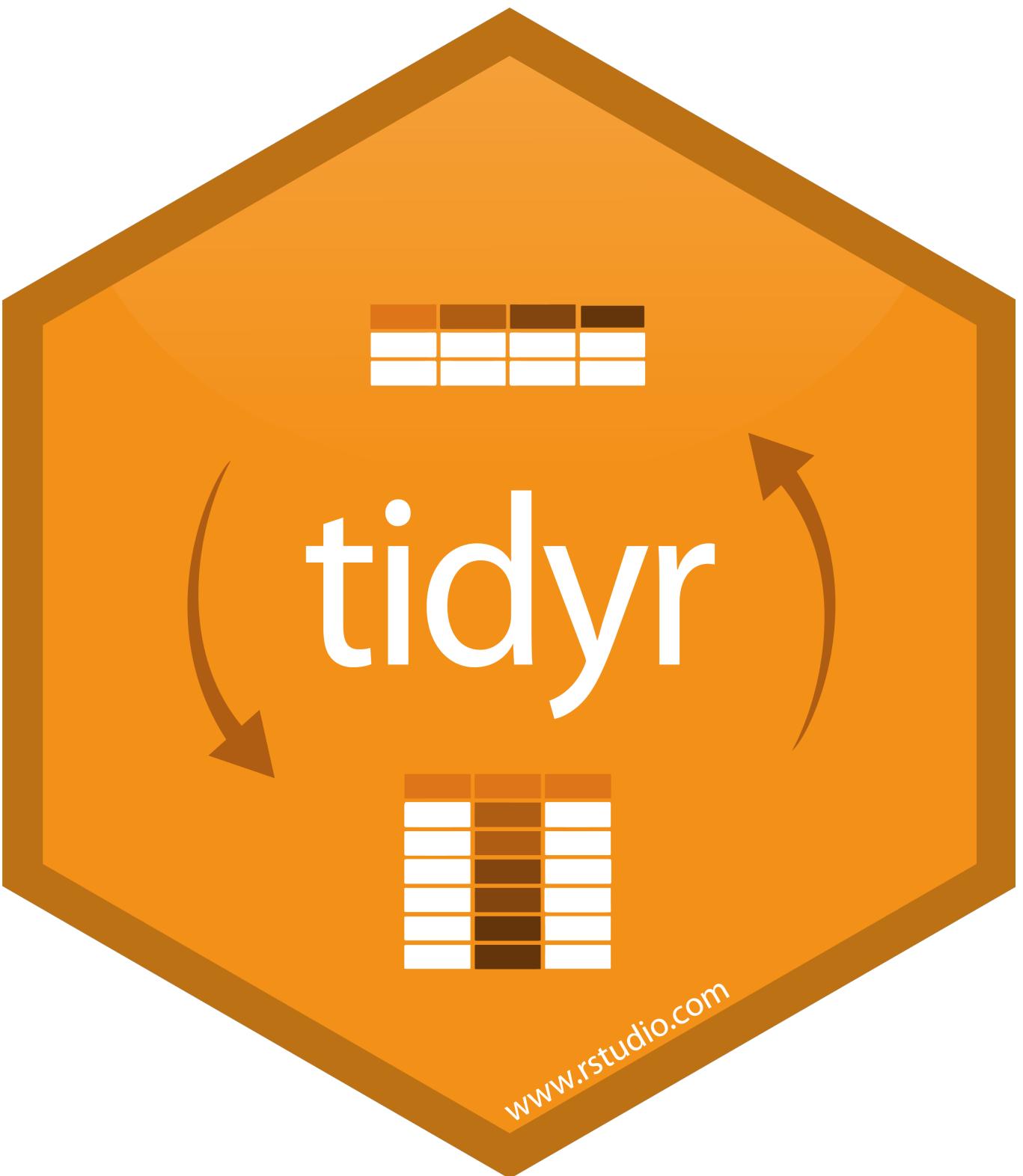
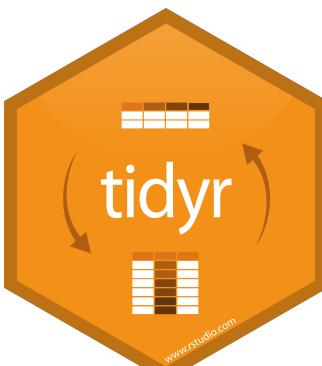
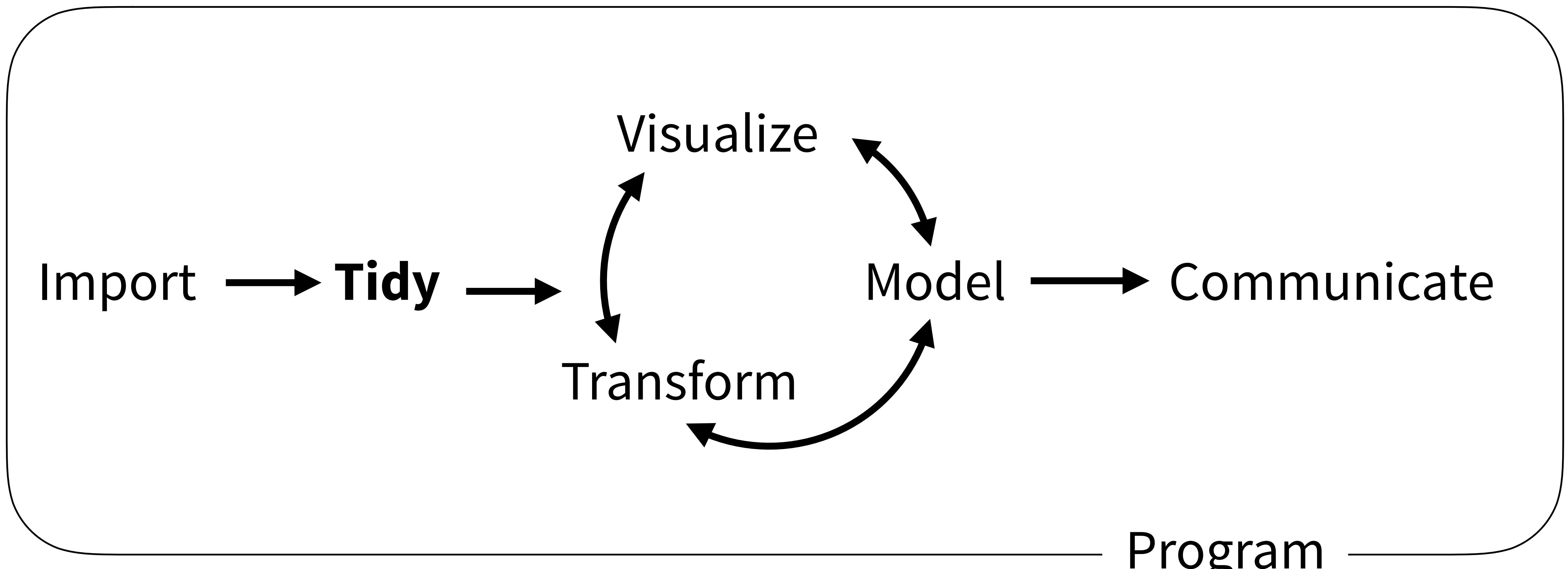


Tidy Data with



In R4DS
Tidy Data

(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**

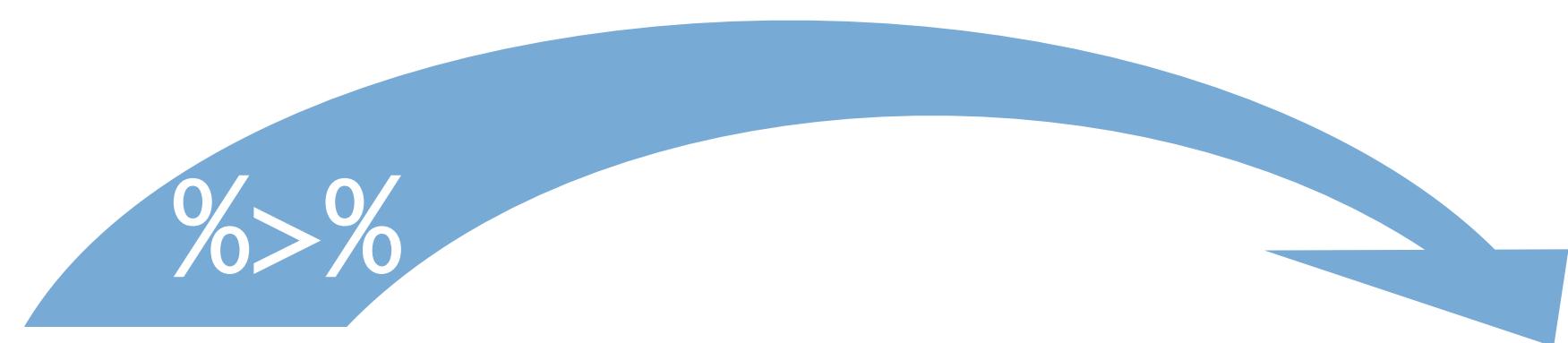
`select()` - extract **variables**

`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 <small><chr></small>	year <small><int></small>	cases <small><int></small>	population <small><int></small>
Afghanistan	1999	745	19937071
Afghanistan	2000	2666	20505360
Brazil	1999	3737	172096362
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	3488
Brazil	2000	population	17450408
China	1999	cases	22258
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

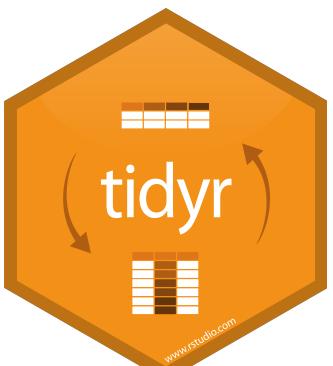
Tidy data

A treemap visualization showing the relationship between four variables: country, year, cases, and pop. The variables are represented by columns in a grid. The width of each column corresponds to the value of that variable for each row. The rows represent different entities, likely countries, though the labels are partially visible.

country	year	cases	pop
Afghanistan	1000	745	1908771
Afghanistan	2000	6666	2012520
Algeria	1000	4707	41700022
Algeria	2000	4707	41700022
Angola	1000	3700	17100000
Angola	2000	3700	17100000
Bangladesh	1000	2250	127491272
Bangladesh	2000	2250	127491272
China	2000	6700	12692883

A data set is **tidy** iff:

1. Each **variable** is in its own **column**
 2. Each **case** is in its own **row**
 3. Each **value** is in its own **cell**



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

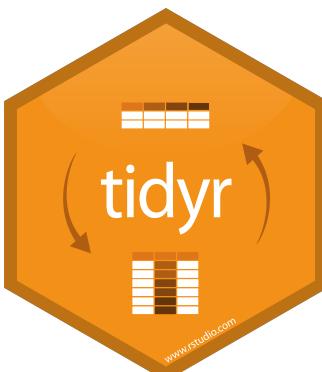
Variables:

- 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

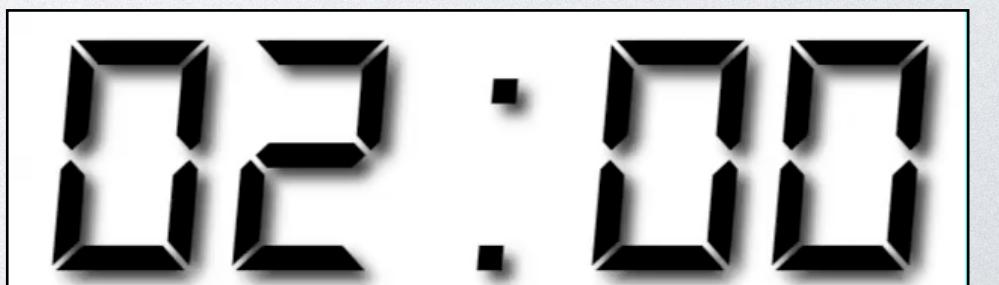
6 rows



Your Turn 2

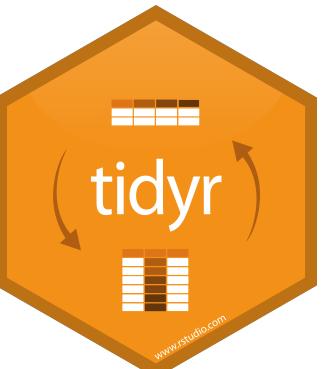
Using `bp_systolic2` with `group_by()` and `summarise()`:

- 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
```



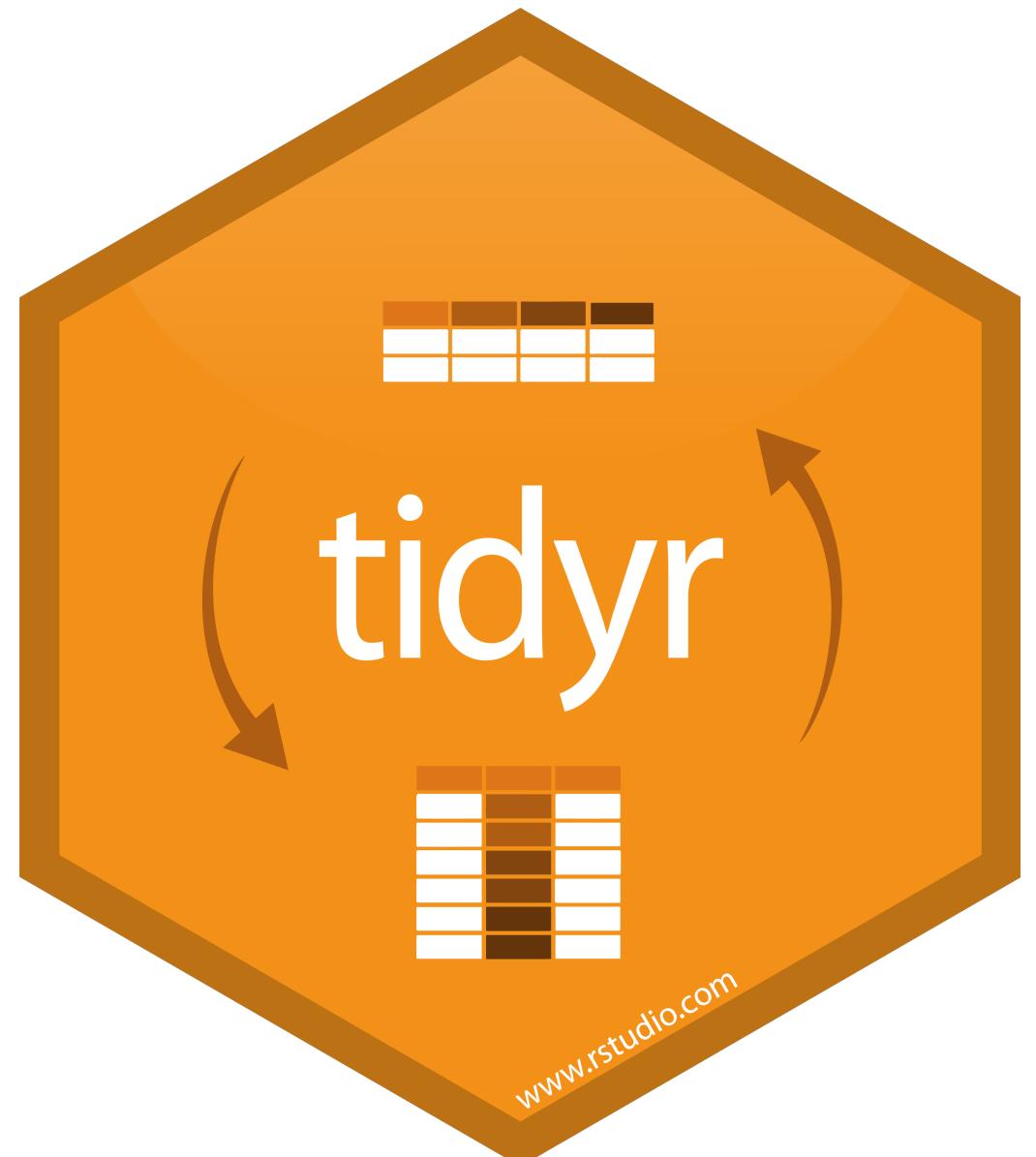
"Tidy data sets are all alike; but
every messy data set is messy in its
own way."

- Hadley Wickham

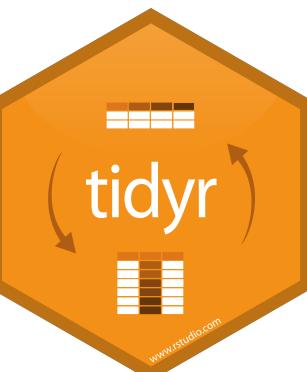
tidyR



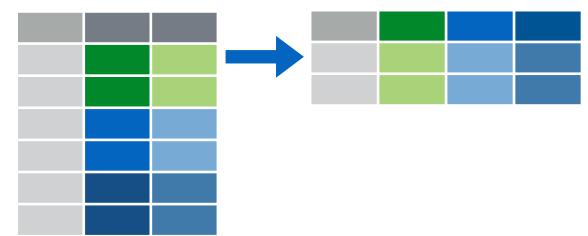
tidyr



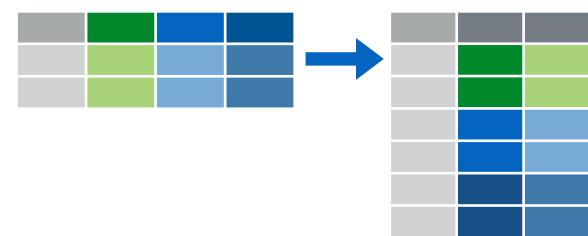
A package that reshapes the layout of tabular data.



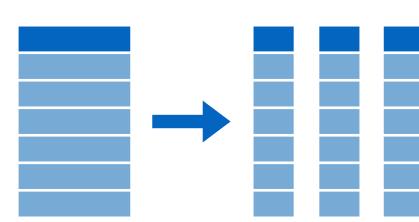
tidyr verbs



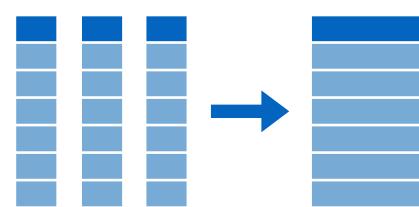
Move values into column names with **spread()**



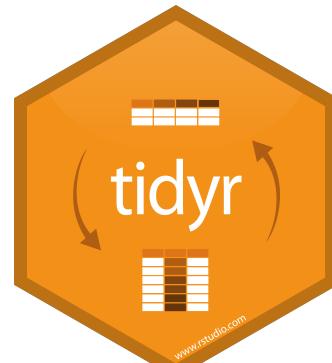
Move column names into values with **gather()**



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



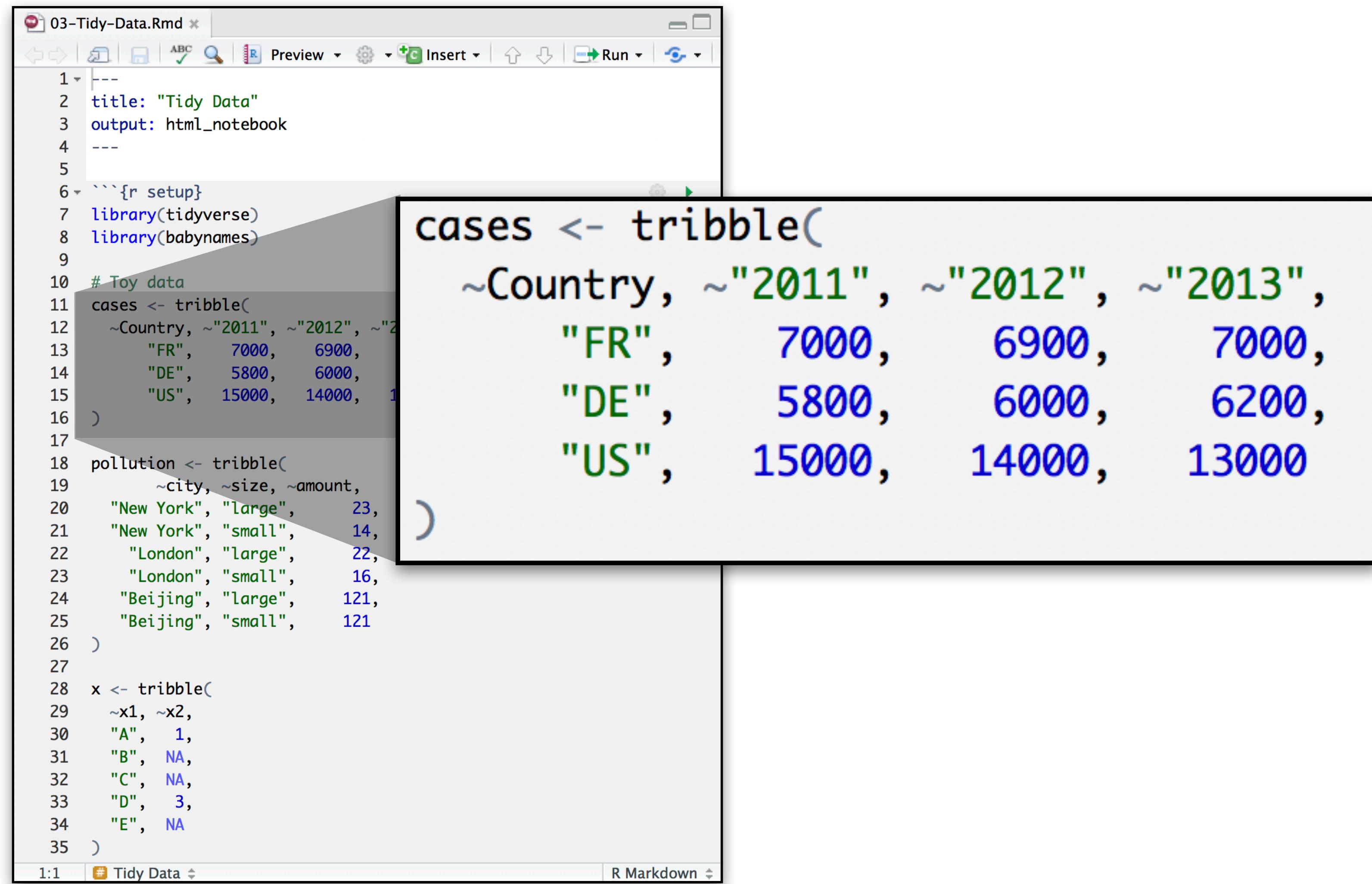
Unite columns with **unite()**



gather()



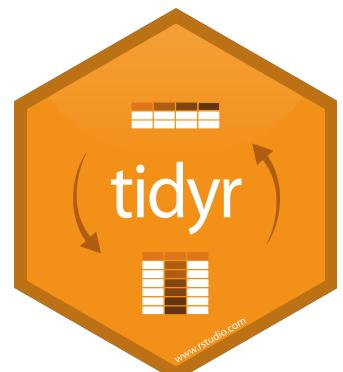
Toy data



The image shows a screenshot of RStudio. On the left, the code editor window displays an R Markdown file named "03-Tidy-Data.Rmd". The code includes setup code for tidyverse and babynames packages, followed by three tribble() calls to create datasets for cases, pollution, and x. The preview pane on the right shows the resulting data frame for the 'cases' dataset, which has columns for Country (~Country), Year (~"2011"/~"2012"/~"2013"), and Value (7000/6900/7000, 5800/6000/6200, 15000/14000/13000).

```
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,  
14   "DE", 5800, 6000,  
15   "US", 15000, 14000,  
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 )
```

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



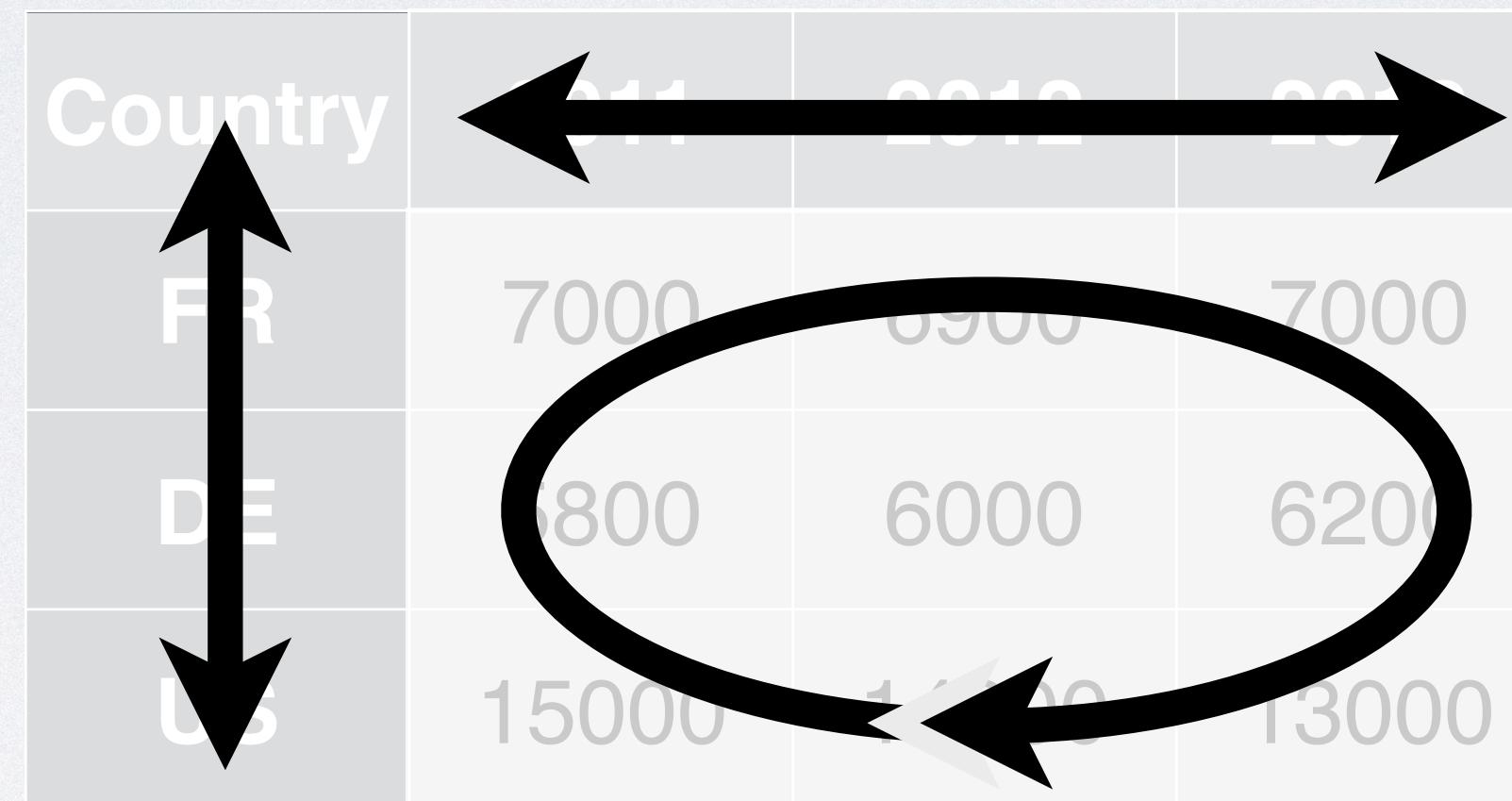
Consider

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

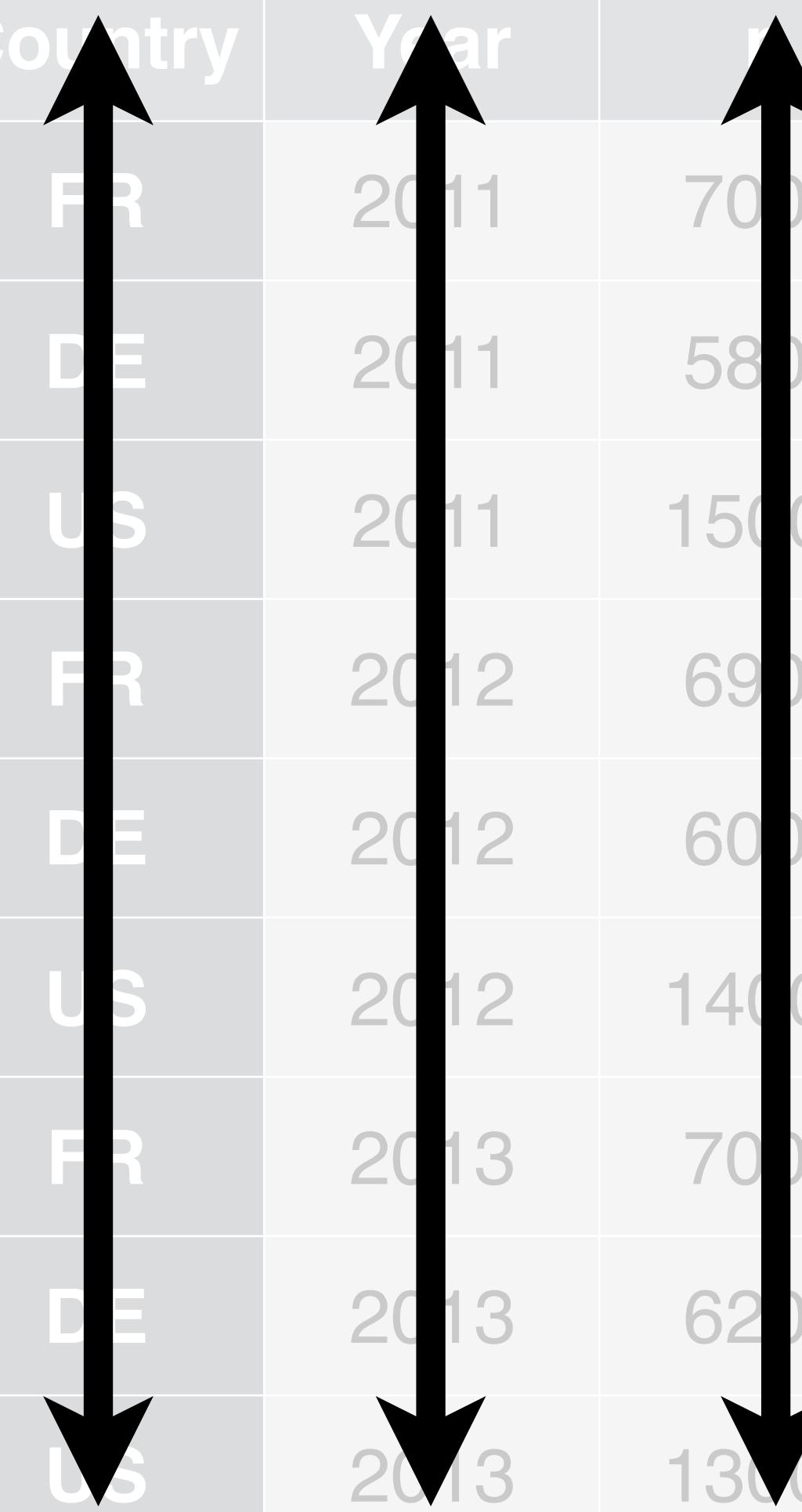
Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

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

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

gather()

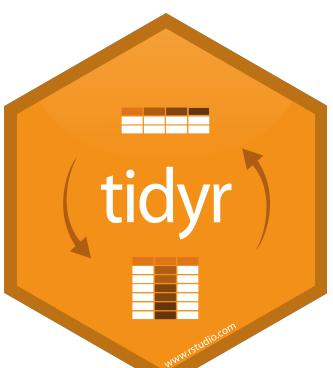
```
cases %>% gather(key = "year", value = "n", 2:4)
```

**data frame to
reshape**

**name of the
new key
column**
(a character
string)

**name of the
new value
column**
(a character
string)

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

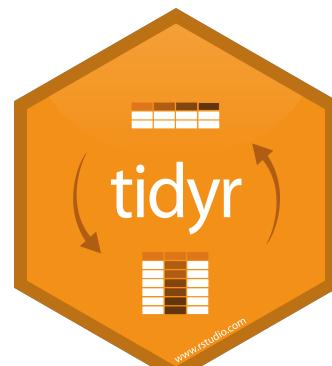


gather()

```
cases %>% gather("year", "n", 2:4)
```

numeric
indexes

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

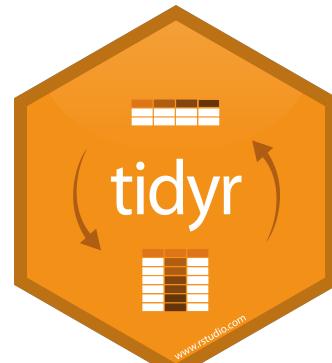


gather()

```
cases %>% gather("year", "n", "2011", "2012", "2013")
```

names

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

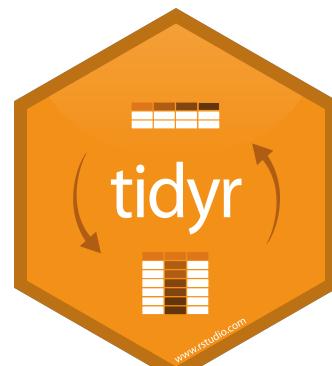


gather()

```
cases %>% gather("year", "n", -Country)
```

Everything
except...

Country	Not Country	Not Country	Not Country
	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000



Your Turn 4

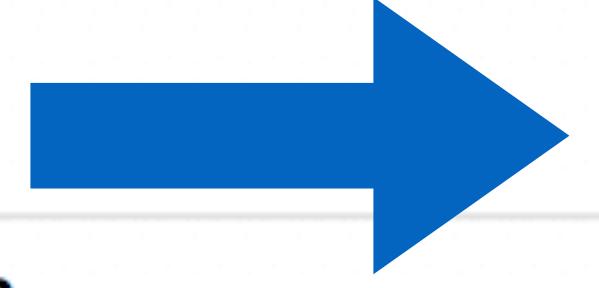
Use `gather()` to reorganize `table4a` into three columns: *country*, *year*, and *cases*.

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

3 rows

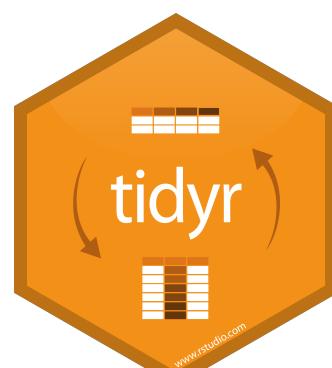


```
table4a %>%  
  gather(key = "year", value = "n", 2:3)
```



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

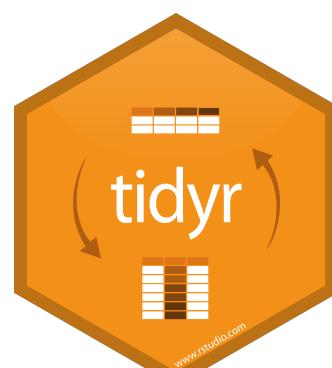
6 rows



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

country	year	n
<chr>	<int>	<int>
Afghanistan	1999	745
Brazil	1999	37737
China	1999	212258
Afghanistan	2000	2666
Brazil	2000	80488
China	2000	213766

6 rows



spread()

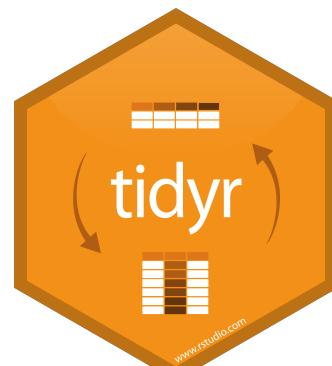


Toy data

The screenshot shows an RStudio interface with an R Markdown file titled "03-Tidy-Data.Rmd". The code in the editor illustrates the creation of a "toy data" set using the `tribble` function from the `tidyverse` package. 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

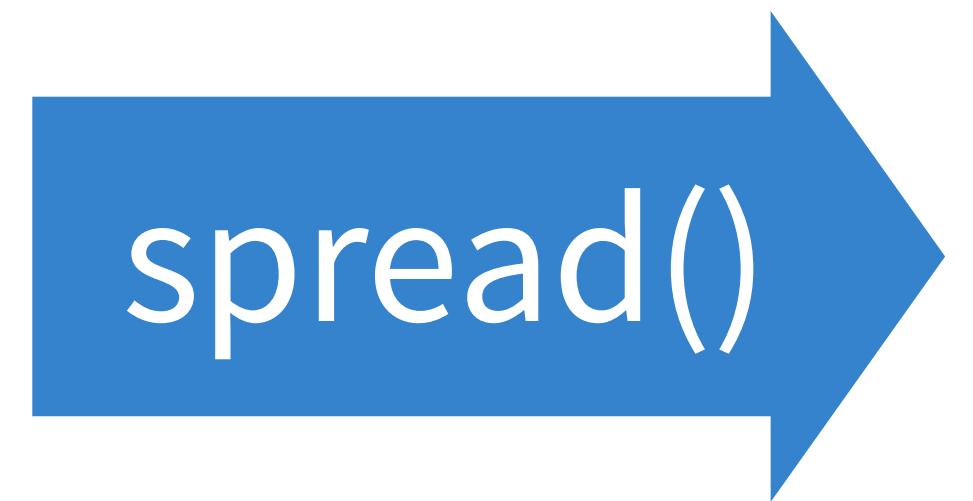
city	large	small
New York	23	14
London	22	16
Beijing	121	56

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

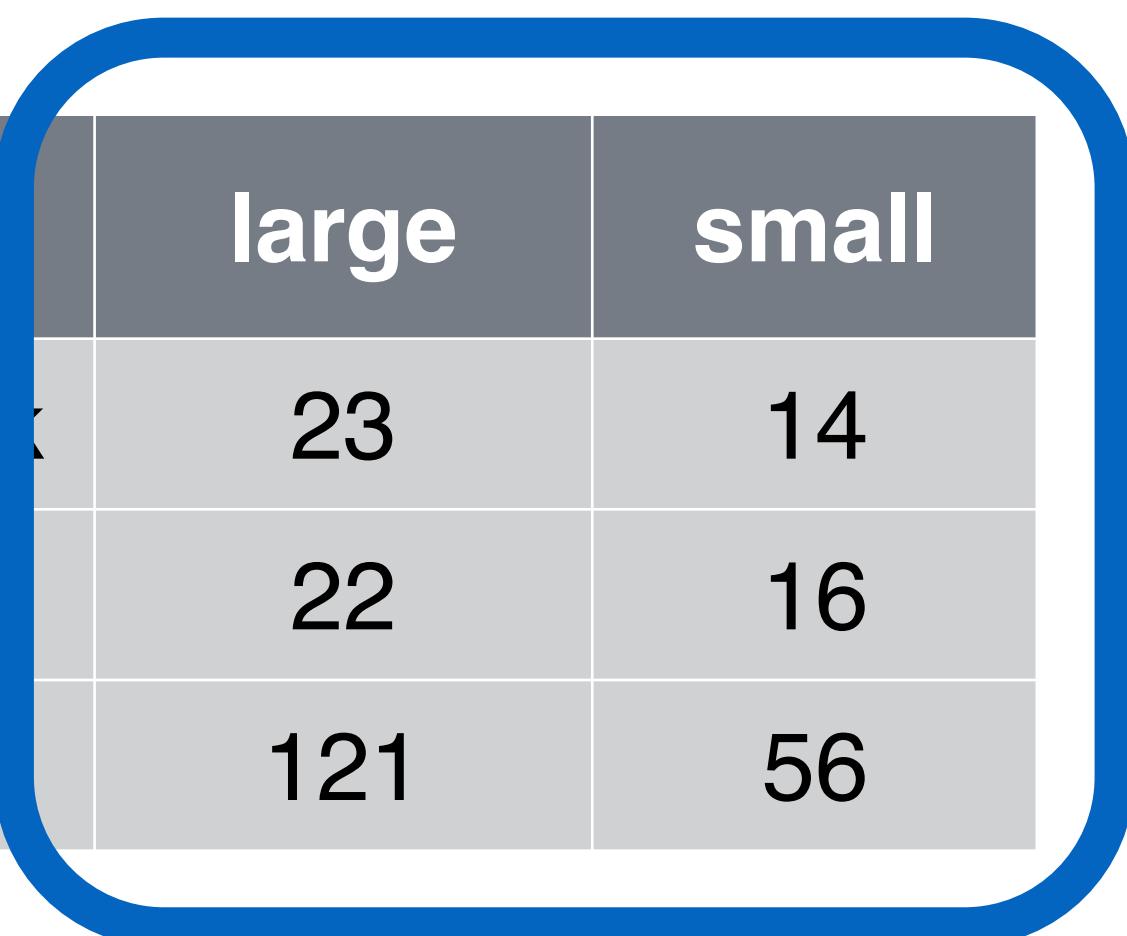


spread()

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

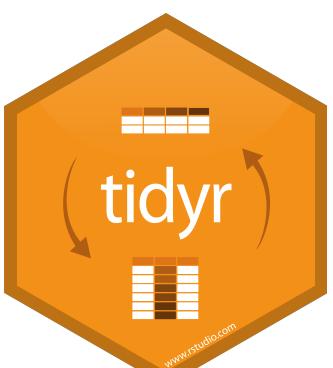
spread()

```
pollution %>% spread(key = size, value = amount)
```

**data frame to
reshape**

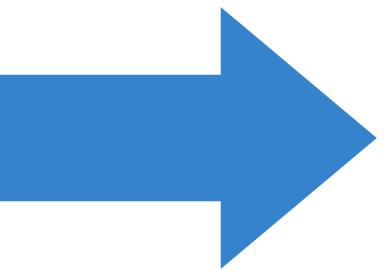
column to use for keys
(becomes new
column names)

column to use for values
(becomes new
column cells)

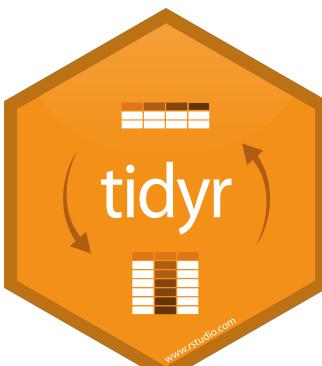


```
pollution %>% spread(size, amount)
```

	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

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

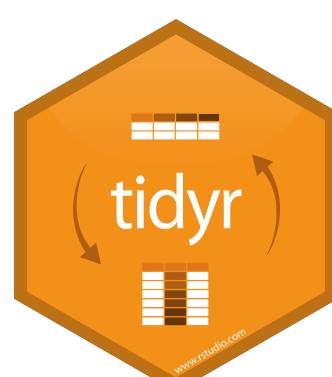
country	year	type	count
<chr>	<int>	<chr>	<int>
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362



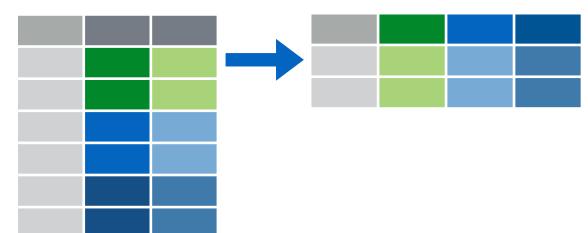
```
table2 %>%  
  spread(key = type, value = count)
```

	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

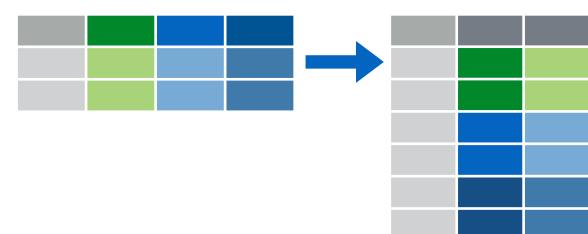
6 rows



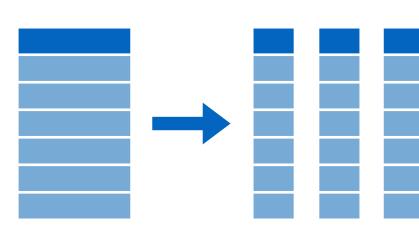
tidyr verbs



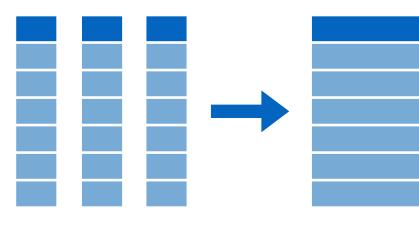
Move values into column names with **spread()**



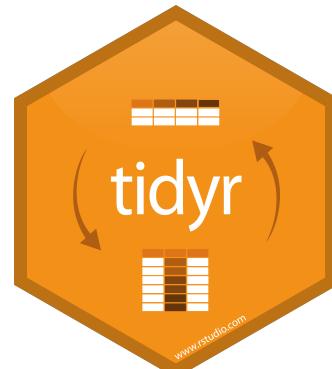
Move column names into values with **gather()**



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



Unite columns with **unite()**



Tidy Data with

