ANT 6973: DATA VISUALIZATION AND EXPLORATION

RESHAPING DATA

LAST TIME

- Pipes
- File paths (`here`)
- Reading from flat files (`readr`)
- Reading from spreadsheets (`readxl`)

TODAY'S TOPICS

- Tidy data
- Reshaping from wide to long
- Reshaping from long to wide

WRANGLING

RESHAPING

MANIPULATION

MUNGING

TRANSFORMATION

MAYBE >50% OF YOUR TIME

TYPICAL GOALS

- Clean and error-check the data
- Make the data suitable to use with particular software
  - Plotting
  - Statistical tests
- Reveal information

PACKAGES FOR WORKING WITH DATA

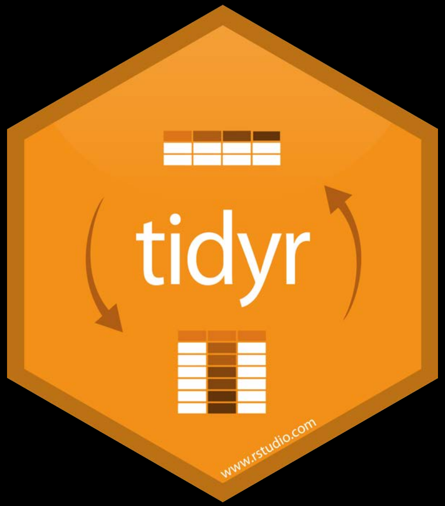
tidyr

dplyr

Both are
part of core

```
library("tidyverse")
```

PACKAGES FOR WORKING WITH DATA



`tidyr`

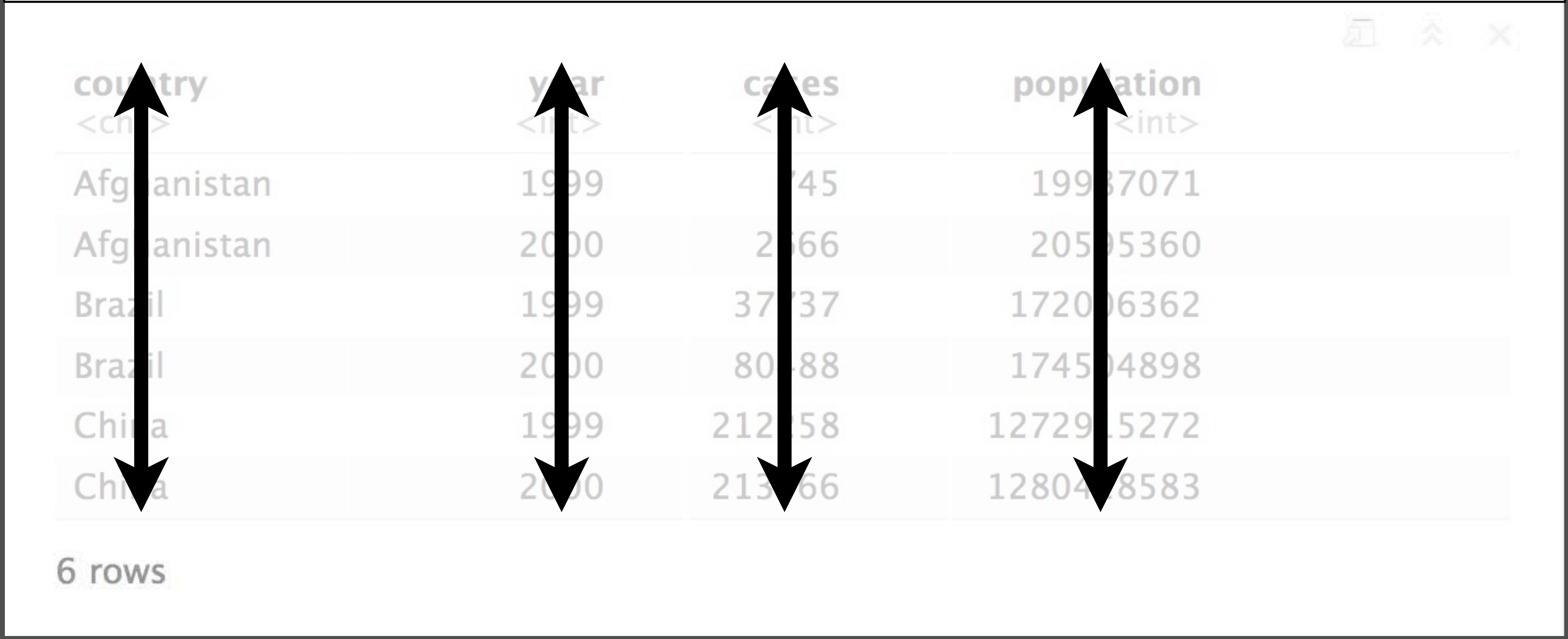
Create tidy data by reshaping

`dplyr`

Manipulate and summarize data

WHAT ARE THE VARIABLES IN THIS DATA SET?

```
tidyr::table1
```

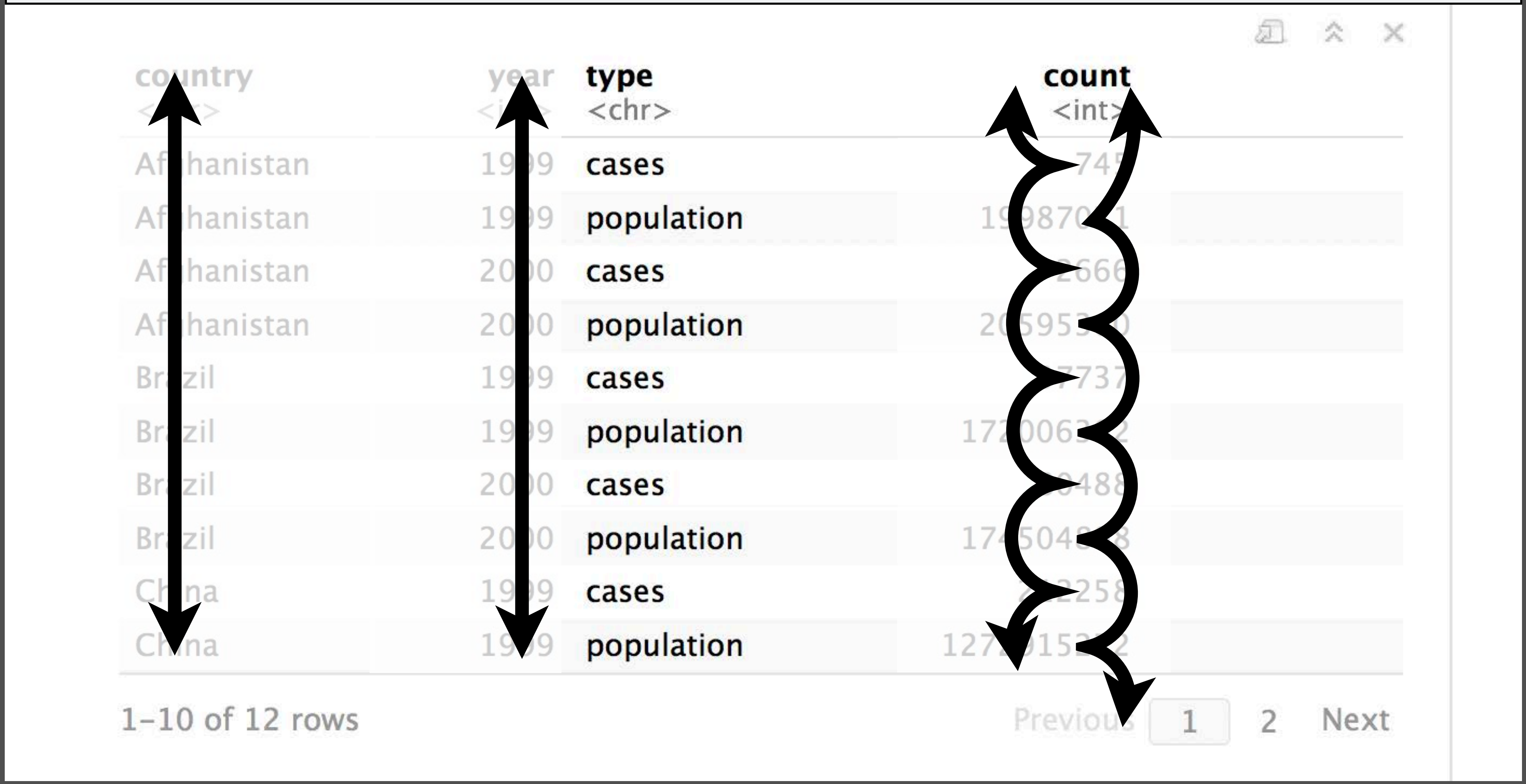


country <chr>	year <int>	cases <int>	population <int>
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

6 rows

WHAT ARE THE VARIABLES IN THIS DATA SET?

```
tidyr::table2
```

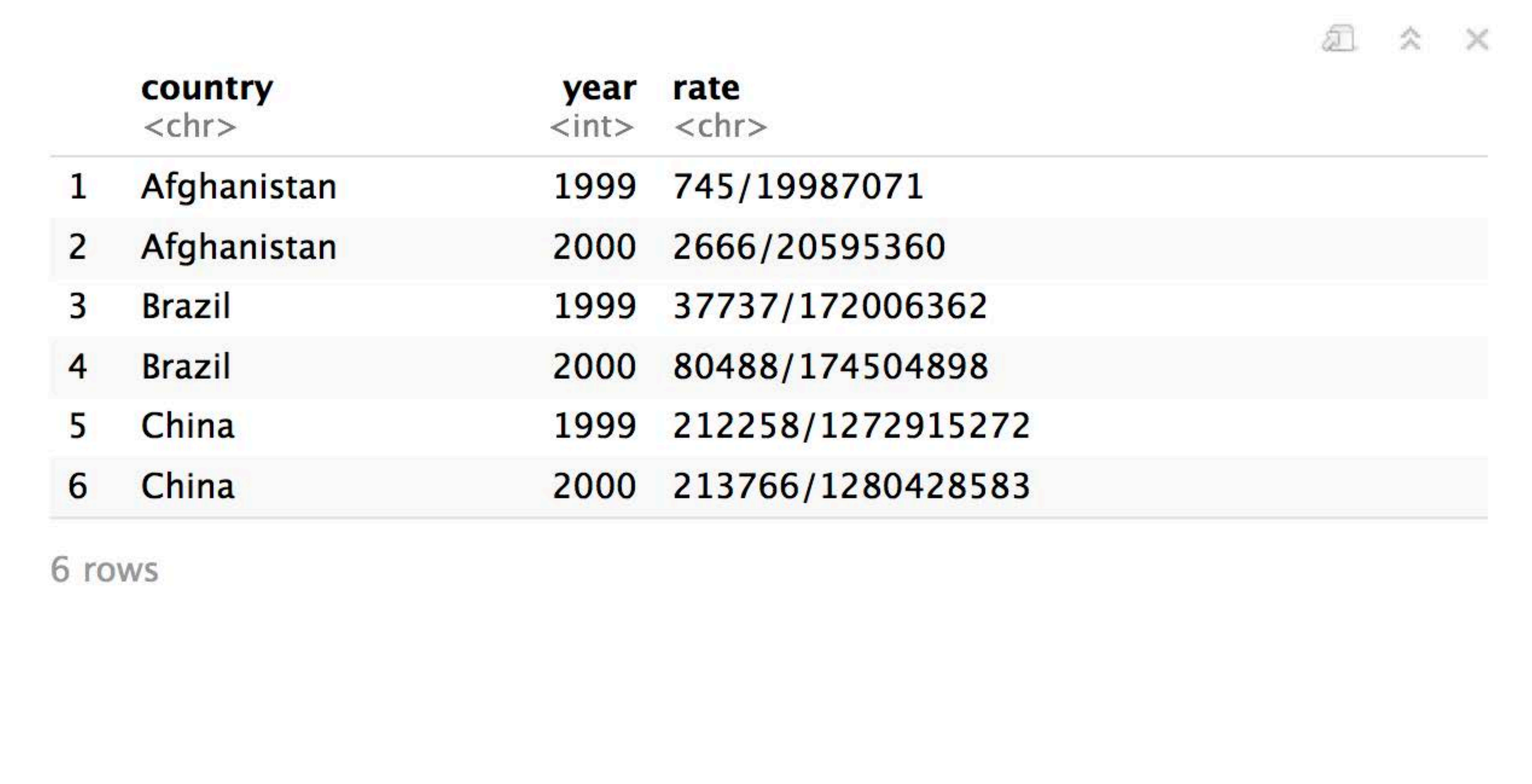


country <chr>	year <int>	type <chr>	count <int>
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999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

1–10 of 12 rows

OTHER (BAD) IDEAS

```
tidyr::table3
```

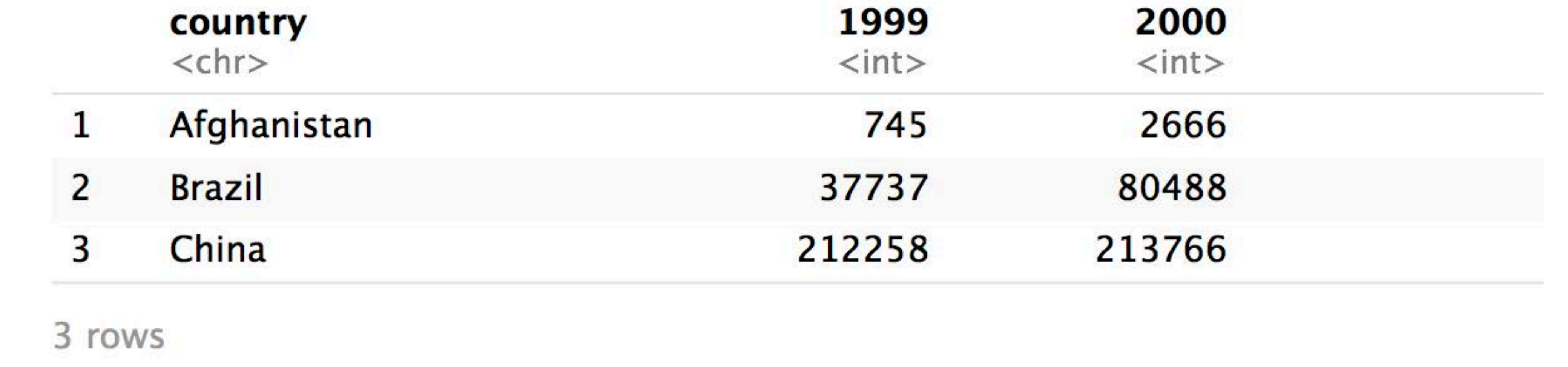


	country <chr>	year <int>	rate <chr>
1	Afghanistan	1999	745/19987071
2	Afghanistan	2000	2666/20595360
3	Brazil	1999	37737/172006362
4	Brazil	2000	80488/174504898
5	China	1999	212258/1272915272
6	China	2000	213766/1280428583

6 rows

OTHER (BAD) IDEAS

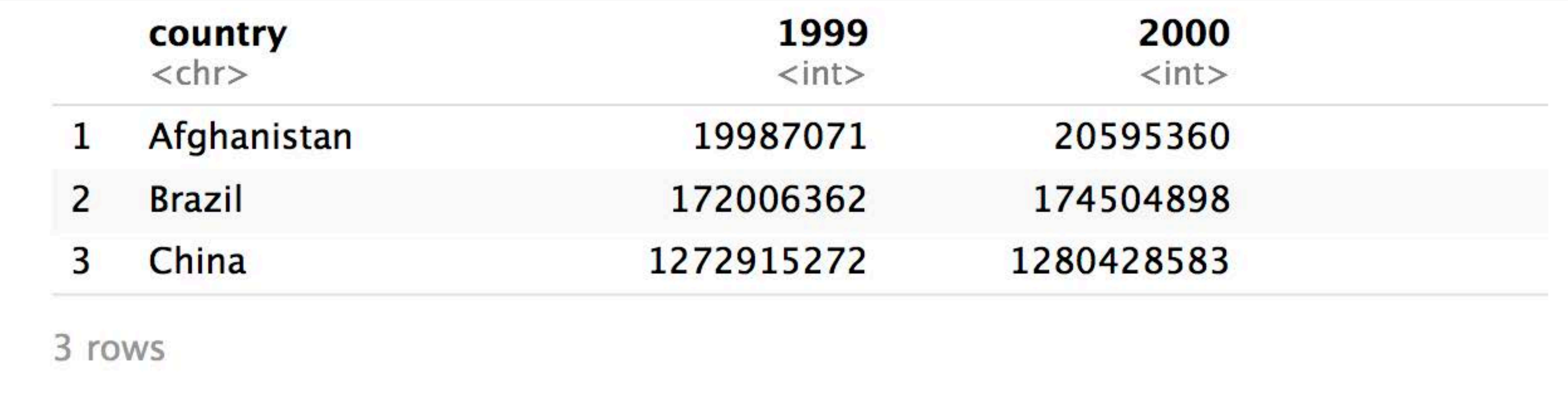
```
tidyr::table4a
```



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

3 rows

```
tidyr::table4b
```

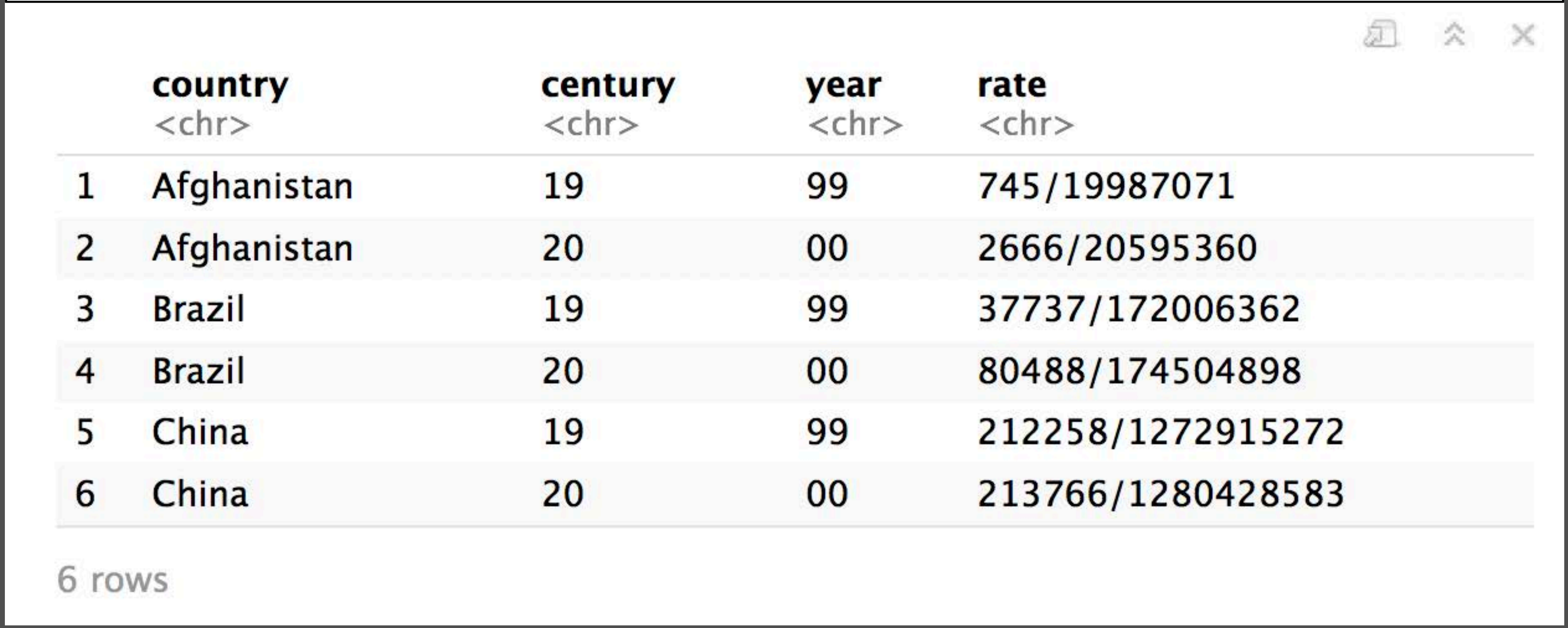


	country	1999	2000
	<chr>	<int>	<int>
1	Afghanistan	19987071	20595360
2	Brazil	172006362	174504898
3	China	1272915272	1280428583

3 rows

OTHER (BAD) IDEAS

```
tidyr::table5
```



	country	century	year	rate
	<chr>	<chr>	<chr>	<chr>
1	Afghanistan	19	99	745/19987071
2	Afghanistan	20	00	2666/20595360
3	Brazil	19	99	37737/172006362
4	Brazil	20	00	80488/174504898
5	China	19	99	212258/1272915272
6	China	20	00	213766/1280428583

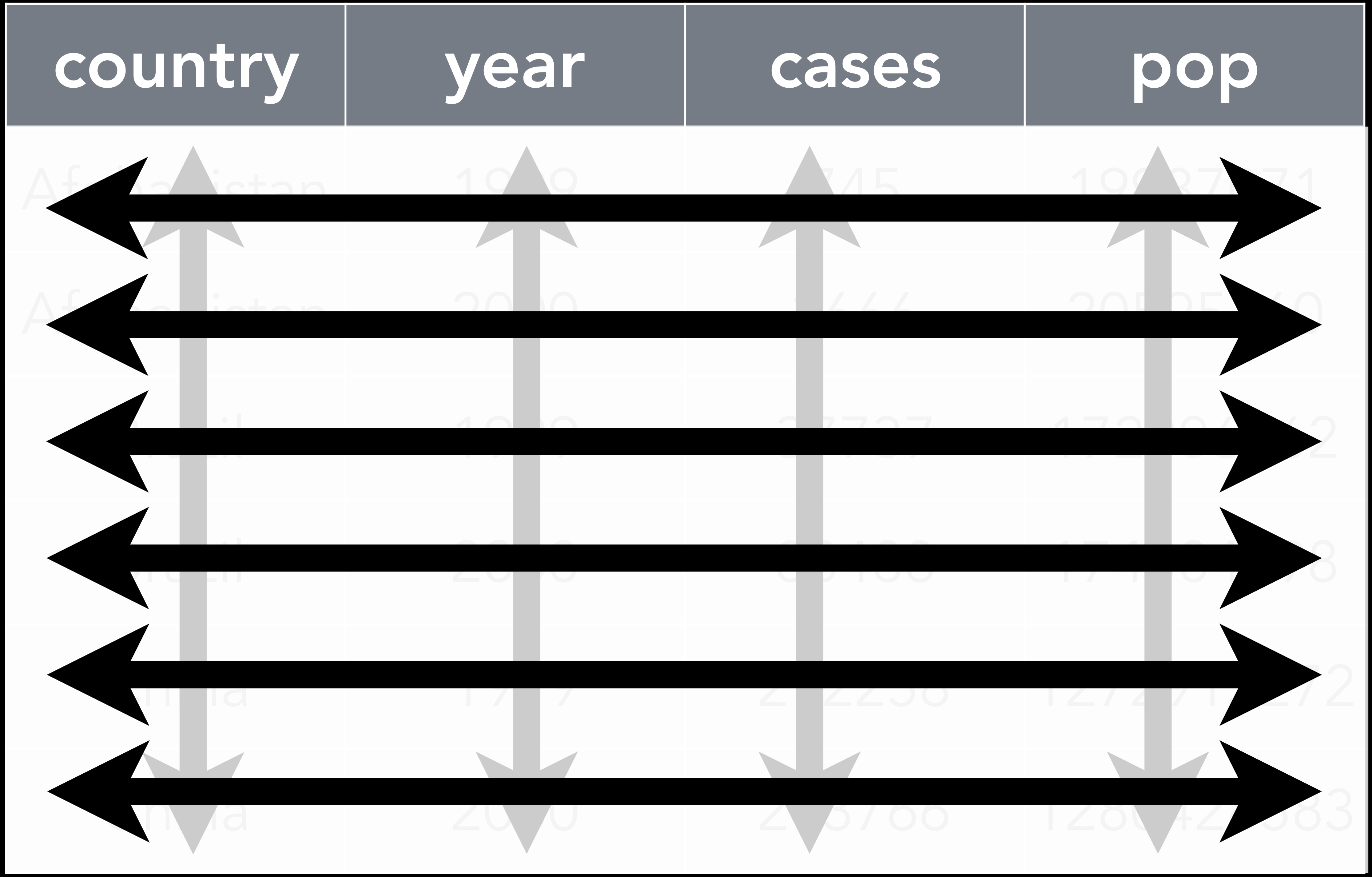
6 rows

TIDY DATA

- Data sets come in **many** different formats.
- Often data are in a format that facilitates data *entry* rather than data *analysis*.
- Most software for scientific computing (including R and SPSS) prefers just **one** format.

A data set is **tidy** if:

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



country	year	cases	pop

EXAMPLE: CONTINGENCY TABLE

	Survived	Died
Drug	15	3
Placebo	4	12

Is this tidy?

REORGANIZE TO MAKE IT TIDY

	Survived	Died
Drug	15	3
Placebo	4	12

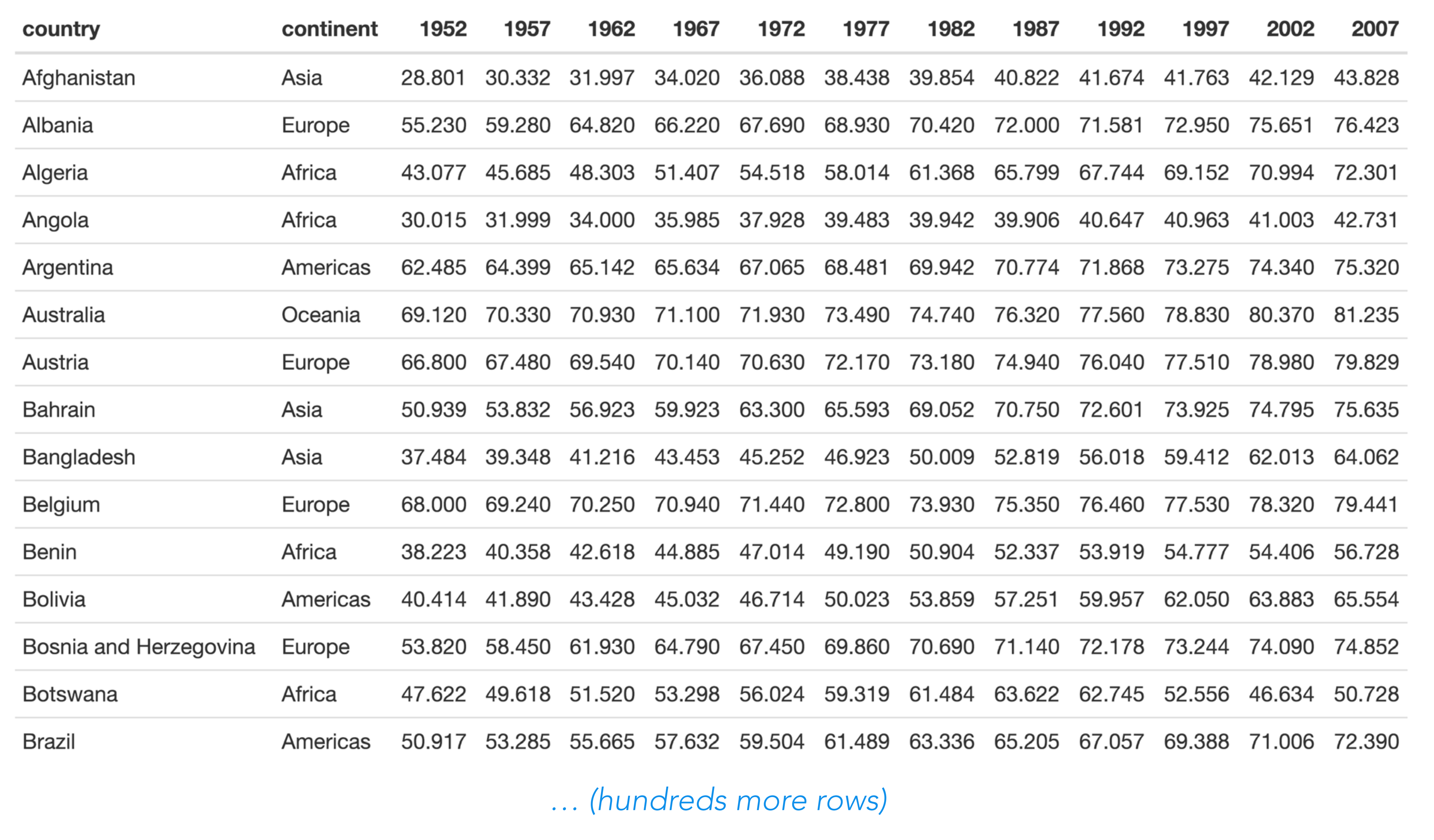
Not tidy

Treatment	Outcome	Count
Drug	Survived	15
Drug	Died	3
Placebo	Survived	4
Placebo	Died	12

Tidy

RESHAPING BY HAND IS NOT ALWAYS SO SIMPLE…

A wide version of gapminder life expectancy data

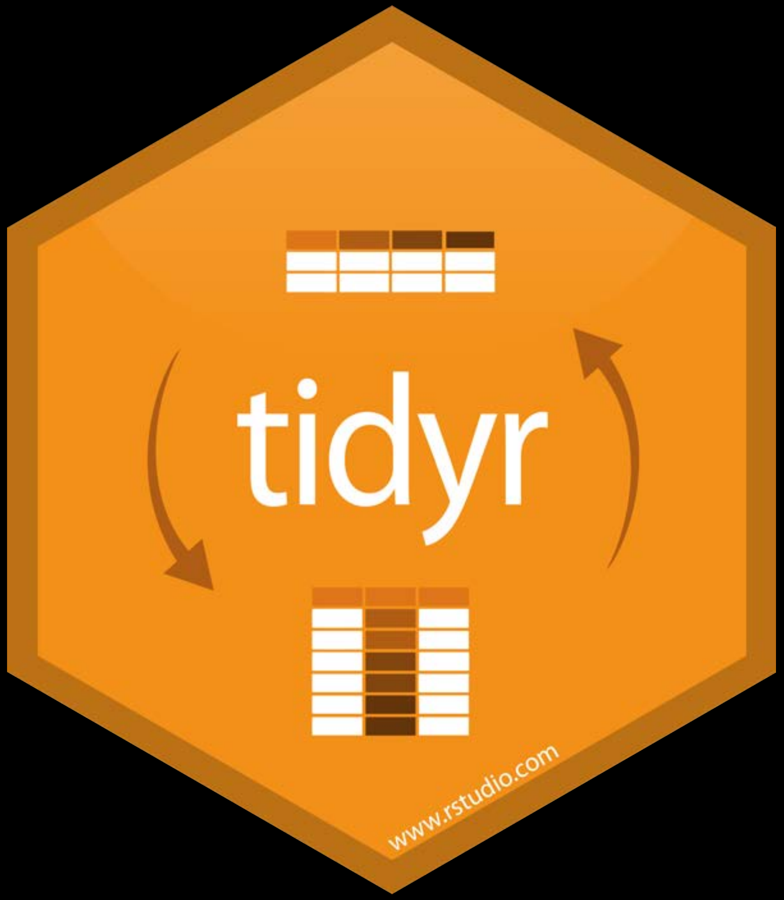


country	continent	1952	1957	1962	1967	1972	1977	1982	1987	1992	1997	2002	2007
Afghanistan	Asia	28.801	30.332	31.997	34.020	36.088	38.438	39.854	40.822	41.674	41.763	42.129	43.828
Albania	Europe	55.230	59.280	64.820	66.220	67.690	68.930	70.420	72.000	71.581	72.950	75.651	76.423
Algeria	Africa	43.077	45.685	48.303	51.407	54.518	58.014	61.368	65.799	67.744	69.152	70.994	72.301
Angola	Africa	30.015	31.999	34.000	35.985	37.928	39.483	39.942	39.906	40.647	40.963	41.003	42.731
Argentina	Americas	62.485	64.399	65.142	65.634	67.065	68.481	69.942	70.774	71.868	73.275	74.340	75.320
Australia	Oceania	69.120	70.330	70.930	71.100	71.930	73.490	74.740	76.320	77.560	78.830	80.370	81.235
Austria	Europe	66.800	67.480	69.540	70.140	70.630	72.170	73.180	74.940	76.040	77.510	78.980	79.829
Bahrain	Asia	50.939	53.832	56.923	59.923	63.300	65.593	69.052	70.750	72.601	73.925	74.795	75.635
Bangladesh	Asia	37.484	39.348	41.216	43.453	45.252	46.923	50.009	52.819	56.018	59.412	62.013	64.062
Belgium	Europe	68.000	69.240	70.250	70.940	71.440	72.800	73.930	75.350	76.460	77.530	78.320	79.441
Benin	Africa	38.223	40.358	42.618	44.885	47.014	49.190	50.904	52.337	53.919	54.777	54.406	56.728
Bolivia	Americas	40.414	41.890	43.428	45.032	46.714	50.023	53.859	57.251	59.957	62.050	63.883	65.554
Bosnia and Herzegovina	Europe	53.820	58.450	61.930	64.790	67.450	69.860	70.690	71.140	72.178	73.244	74.090	74.852
Botswana	Africa	47.622	49.618	51.520	53.298	56.024	59.319	61.484	63.622	62.745	52.556	46.634	50.728
Brazil	Americas	50.917	53.285	55.665	57.632	59.504	61.489	63.336	65.205	67.057	69.388	71.006	72.390

… (hundreds more rows)

HOW CAN WE CONVERT BETWEEN WIDE AND LONG FORMATS?

RESHAPE DATA



RESHAPING "VERBS"

- `gather()`*: reshape from wide to long
- `spread()`*: reshape from long to wide
- `separate()`: split a column with multiple values
- `unite()`: combine multiple columns into one

* Soon to be replaced / deprecated!

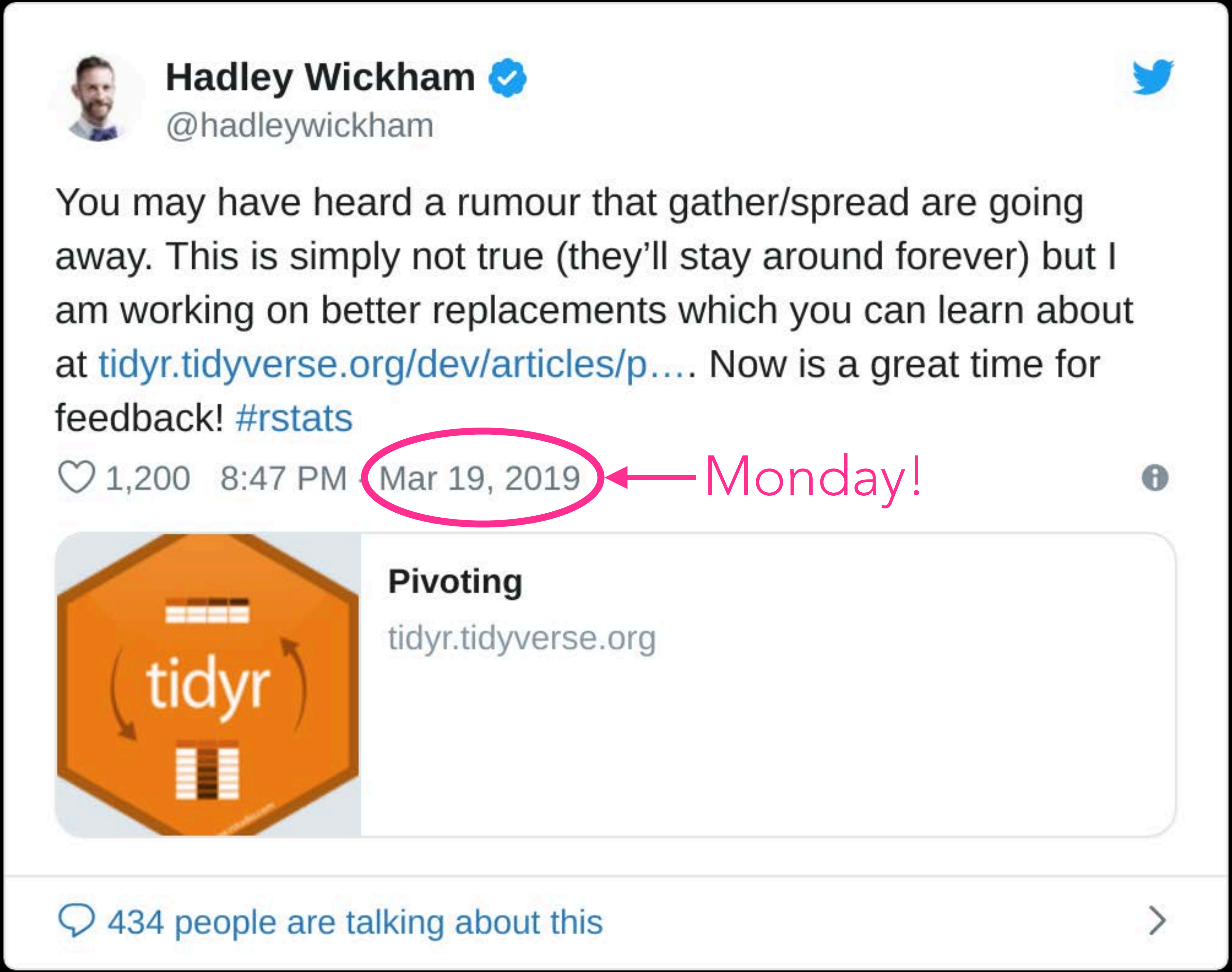
RESHAPING "VERBS"

- `pivot_long()`*: reshape from wide to long
- `pivot_wide()`*: reshape from long to wide
- `separate()`: split a column with multiple values
- `unite()`: combine multiple columns into one

* New & better alternatives; still in flux and not in main package!

A DILEMA

A DILEMA



Hadley Wickham
@hadleywickham

You may have heard a rumour that gather/spread are going away. This is simply not true (they'll stay around forever) but I am working on better replacements which you can learn about at tidyr.tidyverse.org/dev/articles/p…. Now is a great time for feedback! #rstats

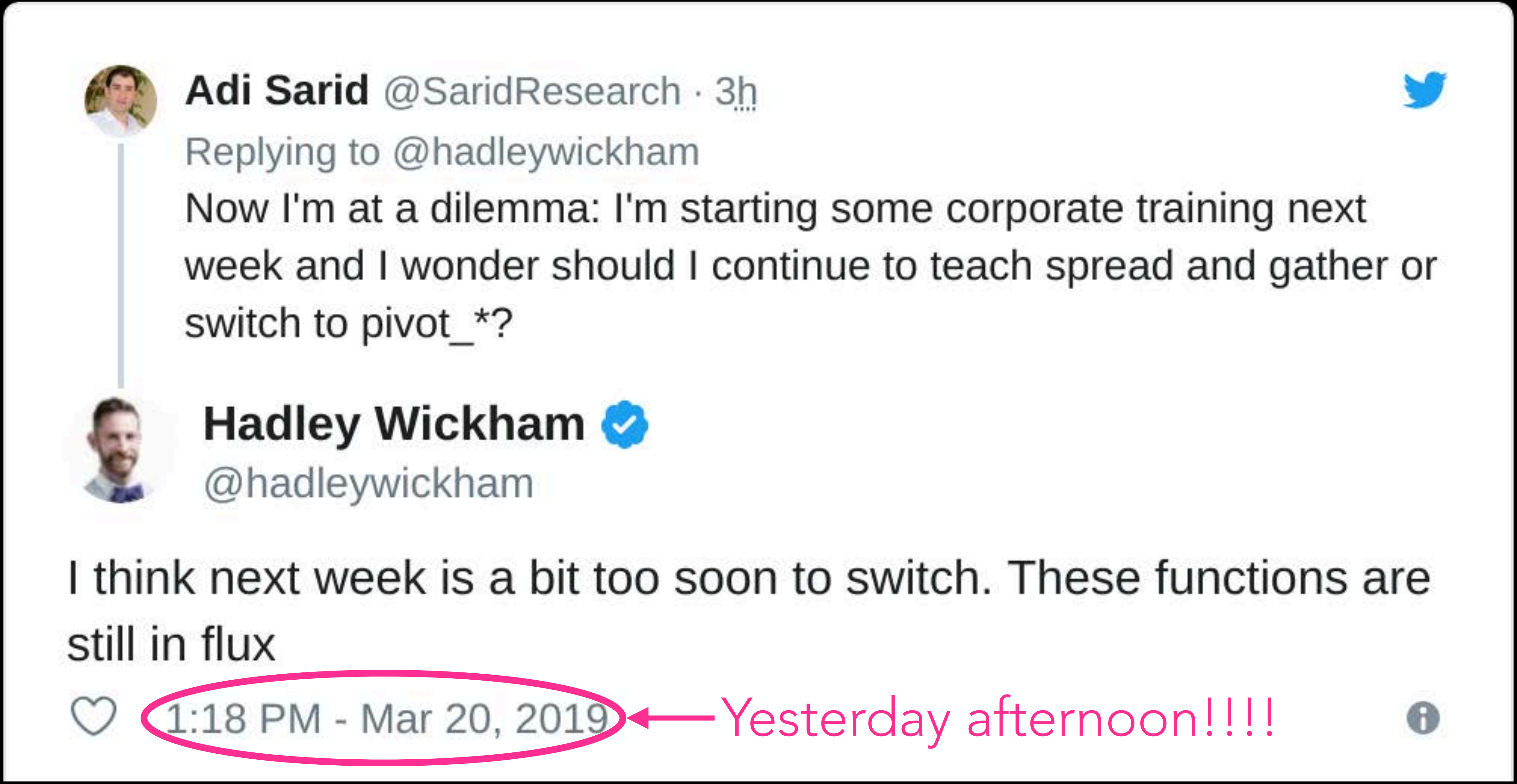
1,200 8:47 PM - Mar 19, 2019

Monday!

Pivoting
tidyr.tidyverse.org

434 people are talking about this

A DILEMA



Adi Sarid @SaridResearch · 3h

Replying to @hadleywickham

Now I'm at a dilemma: I'm starting some corporate training next week and I wonder should I continue to teach spread and gather or switch to pivot_*?

Hadley Wickham

@hadleywickham

I think next week is a bit too soon to switch. These functions are still in flux

1:18 PM - Mar 20, 2019

Yesterday afternoon!!!!

PREPPING TODAY'S CLASS

PREPPING TODAY'S CLASS

USING EXPERIMENTAL PACKAGES

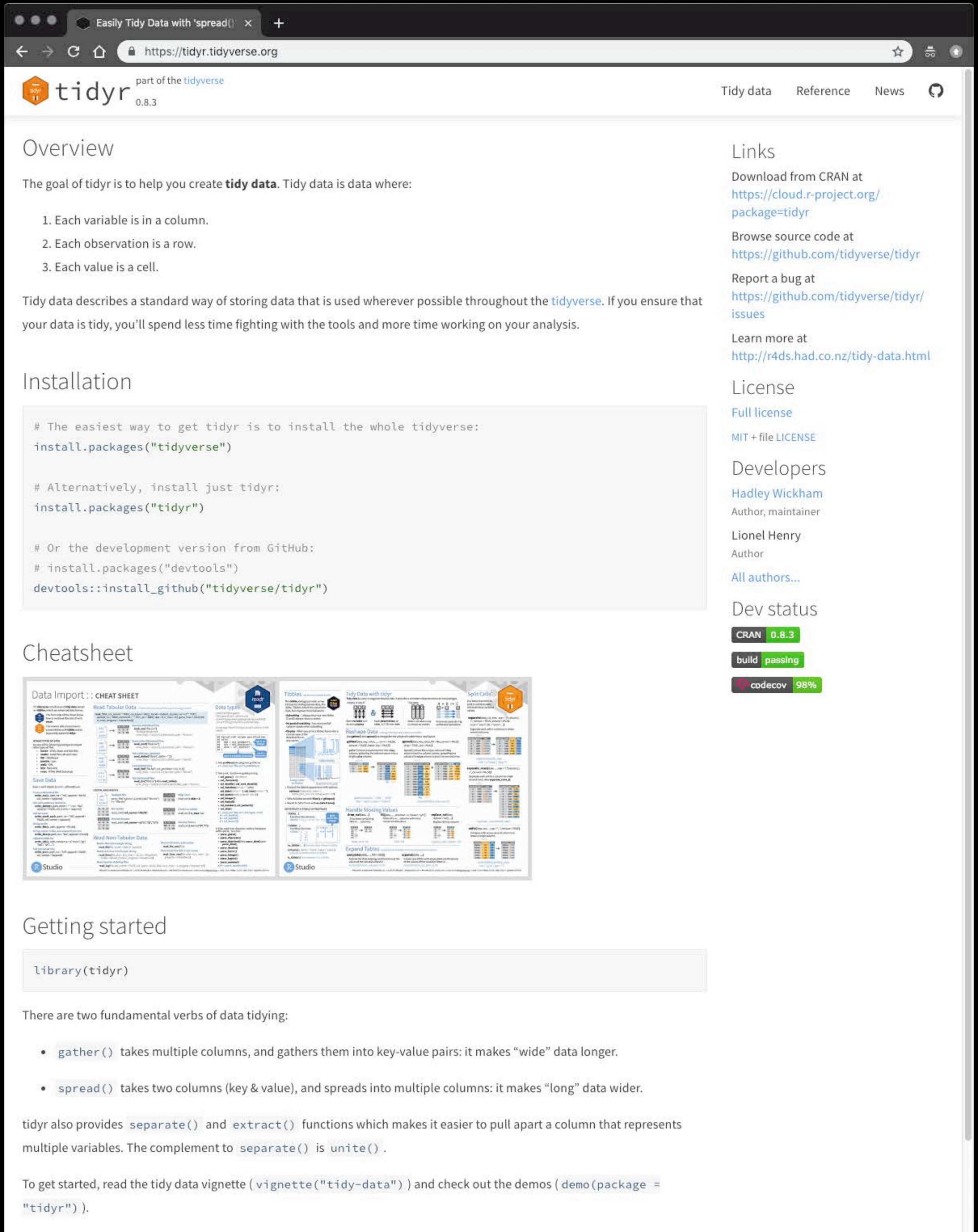
- We will learn the new functions because:
  - They are better for learning how to *reason* about the data transformation.
  - Their syntax is more intuitive (in my opinion), whereas the old functions were confusing to many.
  - They (or very similar variants) will be the way forward in the future.

This class →

USING EXPERIMENTAL PACKAGES

- Possible dangers ahead:
  - Some details (e.g., function or argument names) might change before final version!
  - You will still see `gather()` / `spread()` "out in the wild" for a long time.
  - If it suddenly stops working, check the documentation.

USING EXPERIMENTAL PACKAGES



https://tidyr.tidyverse.org

tidyr part of the tidyverse
0.8.3

Overview

The goal of tidyr is to help you create **tidy data**. Tidy data is data where:

1. Each variable is in a column.
2. Each observation is a row.
3. Each value is a cell.

Tidy data describes a standard way of storing data that is used wherever possible throughout the tidyverse. If you ensure that your data is tidy, you'll spend less time fighting with the tools and more time working on your analysis.

Installation

```
# The easiest way to get tidyr is to install the whole tidyverse:
install.packages("tidyverse")

# Alternatively, install just tidyr:
install.packages("tidyr")

# Or the development version from GitHub:
# install.packages("devtools")
devtools::install_github("tidyverse/tidyr")
```

Cheatsheet

Getting started

```
library(tidyr)
```

There are two fundamental verbs of data tidying:

- `gather()` takes multiple columns, and gathers them into key-value pairs: it makes "wide" data longer.
- `spread()` takes two columns (key & value), and spreads into multiple columns: it makes "long" data wider.

tidyr also provides `separate()` and `extract()` functions which makes it easier to pull apart a column that represents multiple variables. The complement to `separate()` is `unite()`.

To get started, read the tidy data vignette (`vignette("tidy-data")`) and check out the demos (`demo(package = "tidyr")`).

Links

Download from CRAN at https://cloud.r-project.org/package=tidyr

Browse source code at https://github.com/tidyverse/tidyr

Report a bug at https://github.com/tidyverse/tidyr/issues

Learn more at http://r4ds.had.co.nz/tidy-data.html

License

Full license

MIT + file LICENSE

Developers

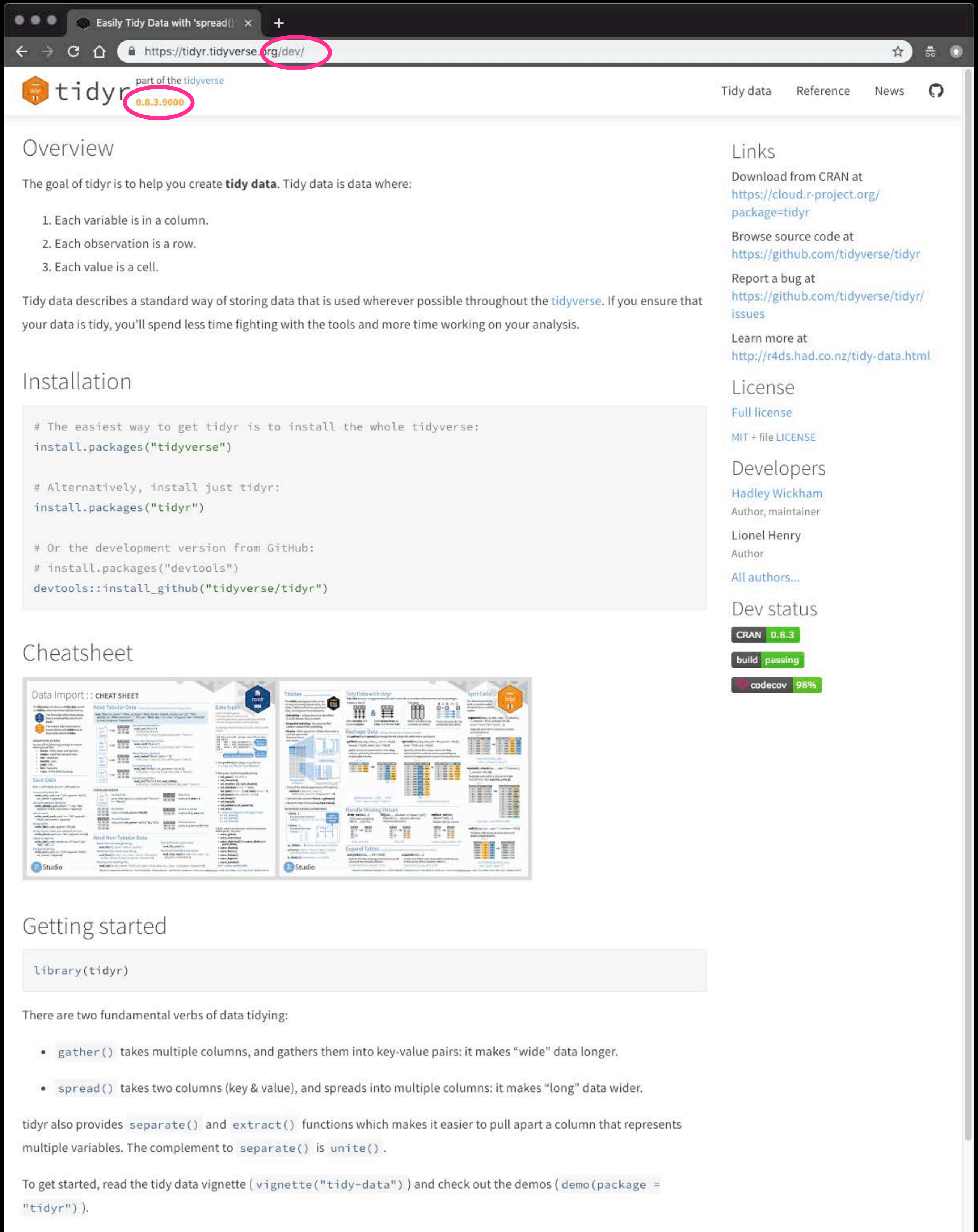
Hadley Wickham
Author, maintainer

Lionel Henry
Author

All authors...

Dev status

CRAN 0.8.3
build passing
codecov 98%



https://tidyr.tidyverse.org/dev/

tidyr part of the tidyverse
0.8.3.9000

Overview

The goal of tidyr is to help you create **tidy data**. Tidy data is data where:

1. Each variable is in a column.
2. Each observation is a row.
3. Each value is a cell.

Tidy data describes a standard way of storing data that is used wherever possible throughout the tidyverse. If you ensure that your data is tidy, you'll spend less time fighting with the tools and more time working on your analysis.

Installation

```
# The easiest way to get tidyr is to install the whole tidyverse:
install.packages("tidyverse")

# Alternatively, install just tidyr:
install.packages("tidyr")

# Or the development version from GitHub:
# install.packages("devtools")
devtools::install_github("tidyverse/tidyr")
```

Cheatsheet

Getting started

```
library(tidyr)
```

There are two fundamental verbs of data tidying:

- `gather()` takes multiple columns, and gathers them into key-value pairs: it makes "wide" data longer.
- `spread()` takes two columns (key & value), and spreads into multiple columns: it makes "long" data wider.

tidyr also provides `separate()` and `extract()` functions which makes it easier to pull apart a column that represents multiple variables. The complement to `separate()` is `unite()`.

To get started, read the tidy data vignette (`vignette("tidy-data")`) and check out the demos (`demo(package = "tidyr")`).

Links

Download from CRAN at https://cloud.r-project.org/package=tidyr

Browse source code at https://github.com/tidyverse/tidyr

Report a bug at https://github.com/tidyverse/tidyr/issues

Learn more at http://r4ds.had.co.nz/tidy-data.html

License

Full license

MIT + file LICENSE

Developers

Hadley Wickham
Author, maintainer

Lionel Henry
Author

All authors...

Dev status

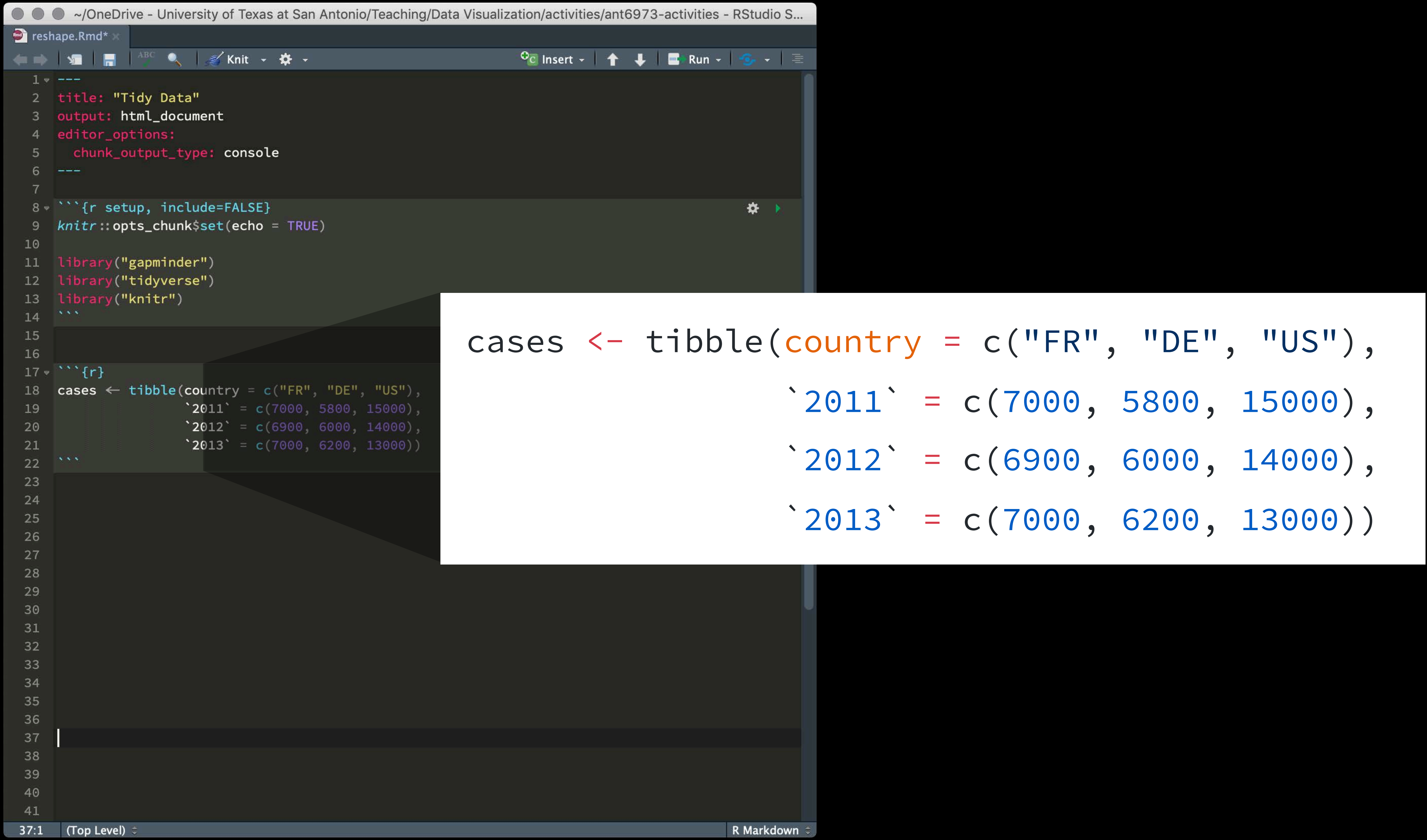
CRAN 0.8.3
build passing
codecov 98%

USING EXPERIMENTAL PACKAGES

```r
devtools::install_github("tidyverse/tidyr")
```

```
pivot_long()
```

PRACTICE DATA


~/OneDrive - University of Texas at San Antonio/Teaching/Data Visualization/activities/ant6973-activities - RStudio S...
reshape.Rmd*
Knit
Insert
Run
37:1
(Top Level)
R Markdown

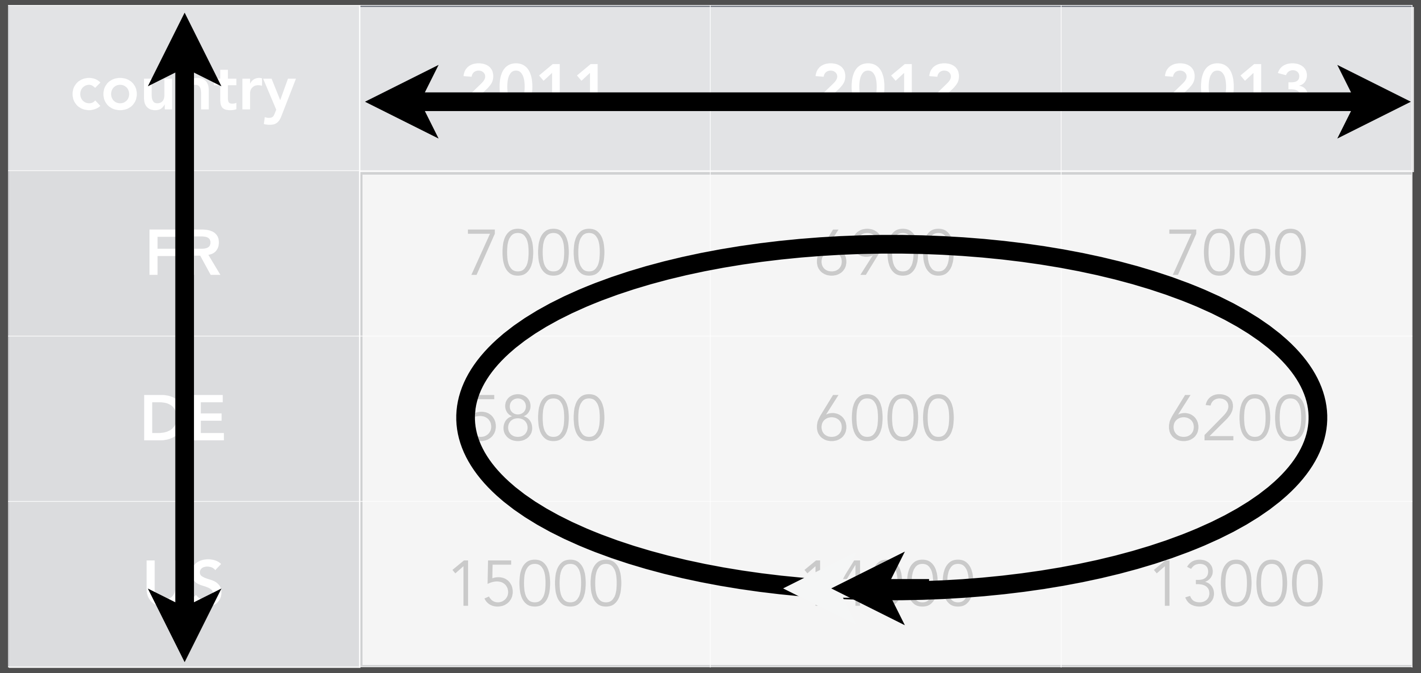
~~~
---
title: "Tidy Data"
output: html_document
editor_options:
  chunk_output_type: console
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)

library("gapminder")
library("tidyverse")
library("knitr")
```


```{r}
cases <- tibble(country = c("FR", "DE", "US"),
                `2011` = c(7000, 5800, 15000),
                `2012` = c(6900, 6000, 14000),
                `2013` = c(7000, 6200, 13000))
```
~~~

```
cases <- tibble(country = c("FR", "DE", "US"),
                `2011` = c(7000, 5800, 15000),
                `2012` = c(6900, 6000, 14000),
                `2013` = c(7000, 6200, 13000))
```

WHAT ARE THE VARIABLES?

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

WHAT ARE THE VARIABLES?



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

- Country
- Year
- Count

ACTIVITY 1

- Plan (e.g., draw on paper) how the data would look if it were organized in 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
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
FR	2012	6900
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
FR	2012	6900
FR	2013	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
FR	2012	6900
FR	2013	7000
DE	2011	5800
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
FR	2012	6900
FR	2013	7000
DE	2011	5800
DE	2012	6000
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
FR	2012	6900
FR	2013	7000
DE	2011	5800
DE	2012	6000
DE	2013	6200
US	2011	15000

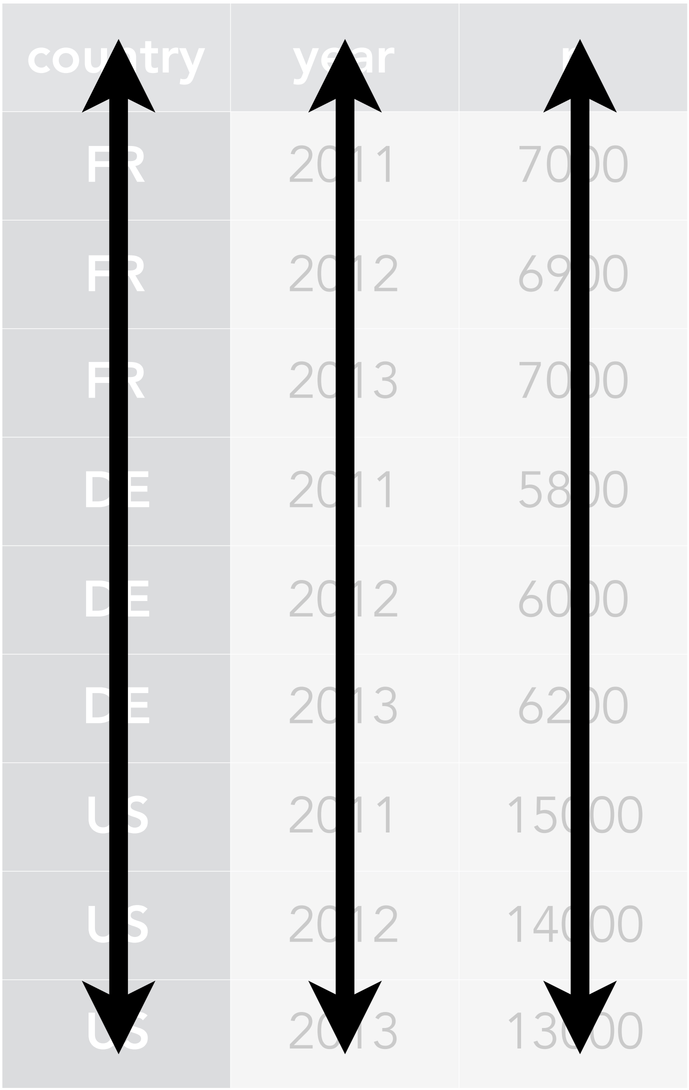
country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

country	year	n
FR	2011	7000
FR	2012	6900
FR	2013	7000
DE	2011	5800
DE	2012	6000
DE	2013	6200
US	2011	15000
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
FR	2012	6900
FR	2013	7000
DE	2011	5800
DE	2012	6000
DE	2013	6200
US	2011	15000
US	2012	14000
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
FR	2012	6900
FR	2013	7000
DE	2011	5800
DE	2012	6000
DE	2013	6200
US	2011	15000
US	2012	14000
US	2013	13000

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

`pivot_long()`

country	year	n
FR	2011	7000
FR	2012	6900
FR	2013	7000
DE	2011	5800
DE	2012	6000
DE	2013	6200
US	2011	15000
US	2012	14000
US	2013	13000

Column names

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

New variable "year"

(former column names)

country	year	n
FR	2011	7000
FR	2012	6900
FR	2013	7000
DE	2011	5800
DE	2012	6000
DE	2013	6200
US	2011	15000
US	2012	14000
US	2013	13000

Transformation logic: column names **TO** new variable

Cell values

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

New variable "n"

(former cell values)

country	year	n
FR	2011	7000
FR	2012	6900
FR	2013	7000
DE	2011	5800
DE	2012	6000
DE	2013	6200
US	2011	15000
US	2012	14000
US	2013	13000

Transformation logic: cell values **TO** new variable

pivot_long()

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

pivot_long()

Data to reshape

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

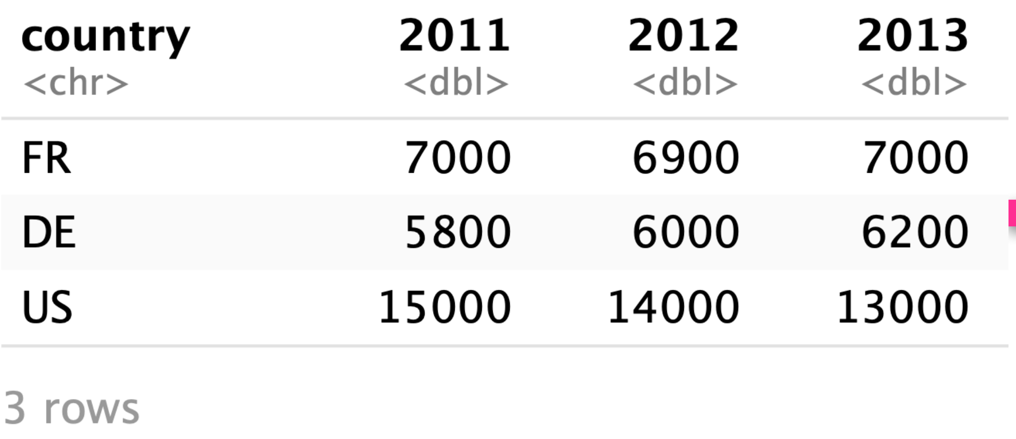
Columns to pivot *(more later)*

Name of new variable for data stored in column names

Name of new variable for data stored in cells values

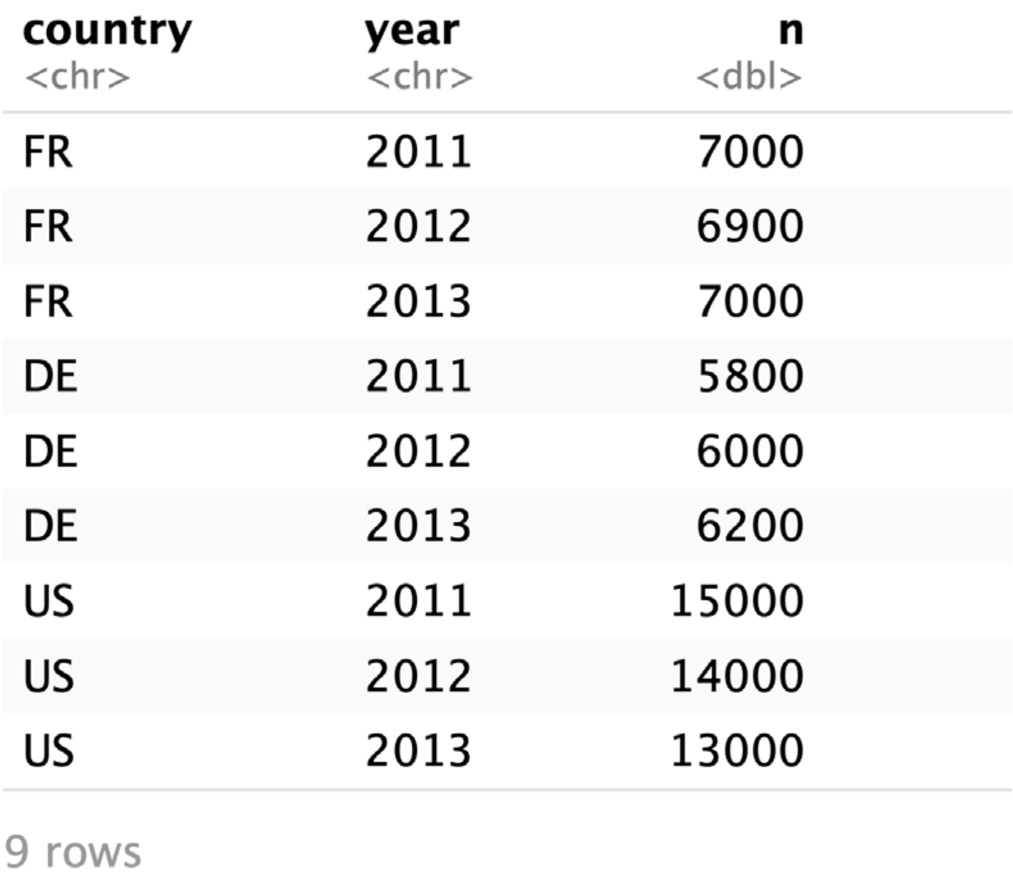
pivot_long()

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



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

3 rows



country	year	n
<chr>	<chr>	<dbl>
FR	2011	7000
FR	2012	6900
FR	2013	7000
DE	2011	5800
DE	2012	6000
DE	2013	6200
US	2011	15000
US	2012	14000
US	2013	13000

9 rows

pivot_long()

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

Numeric indices

	2	3	4
country	**2011**	**2012**	**2013**
<chr>	<dbl>	<dbl>	<dbl>
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

3 rows

pivot_long()

Actual column names

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

	"2011"	"2012"	"2013"
country	2011	2012	2013
<chr>	<dbl>	<dbl>	<dbl>
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

3 rows

pivot_long()

```
cases %>%
  pivot_long(cols = -country,
             names_to = "year",
             values_to = "n")
```

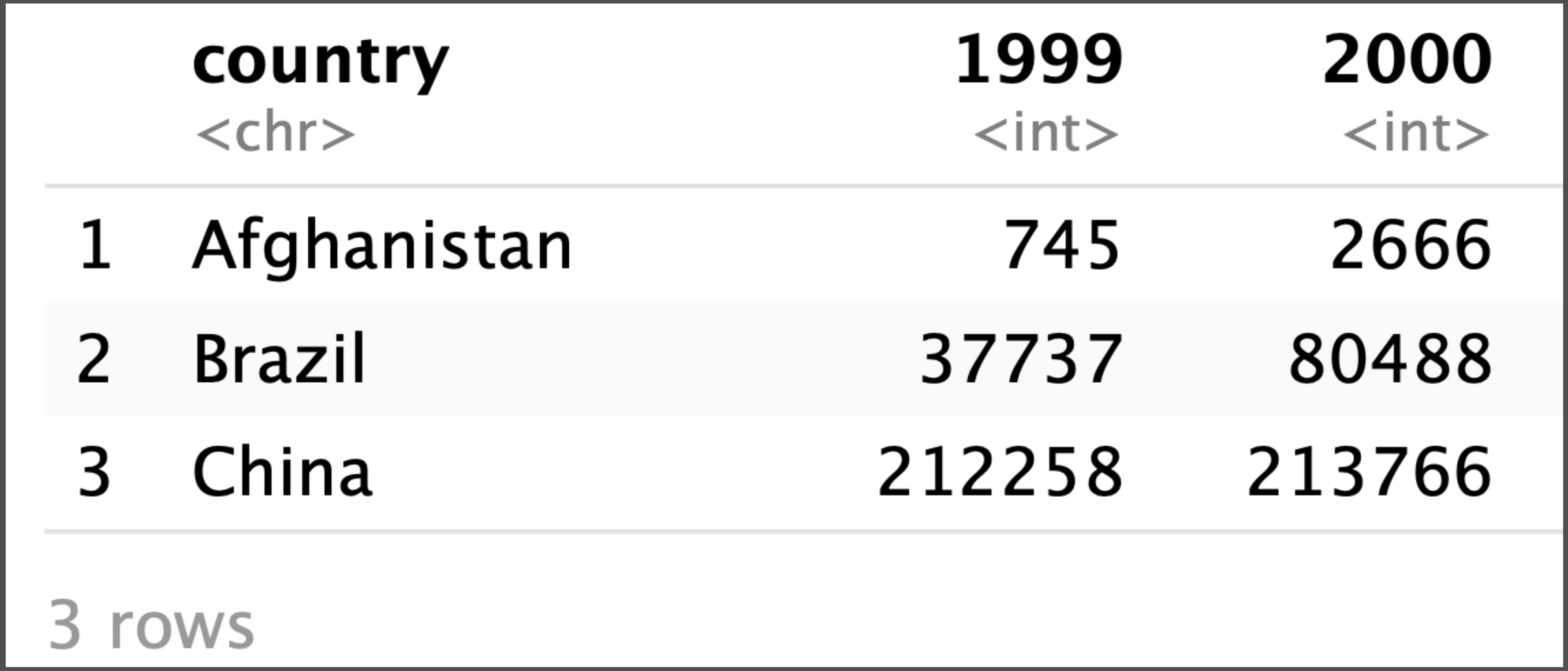
Everything except...

	Not country	Not country	Not country
country	**2011**	**2012**	**2013**
<chr>	<dbl>	<dbl>	<dbl>
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

3 rows

ACTIVITY 2

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

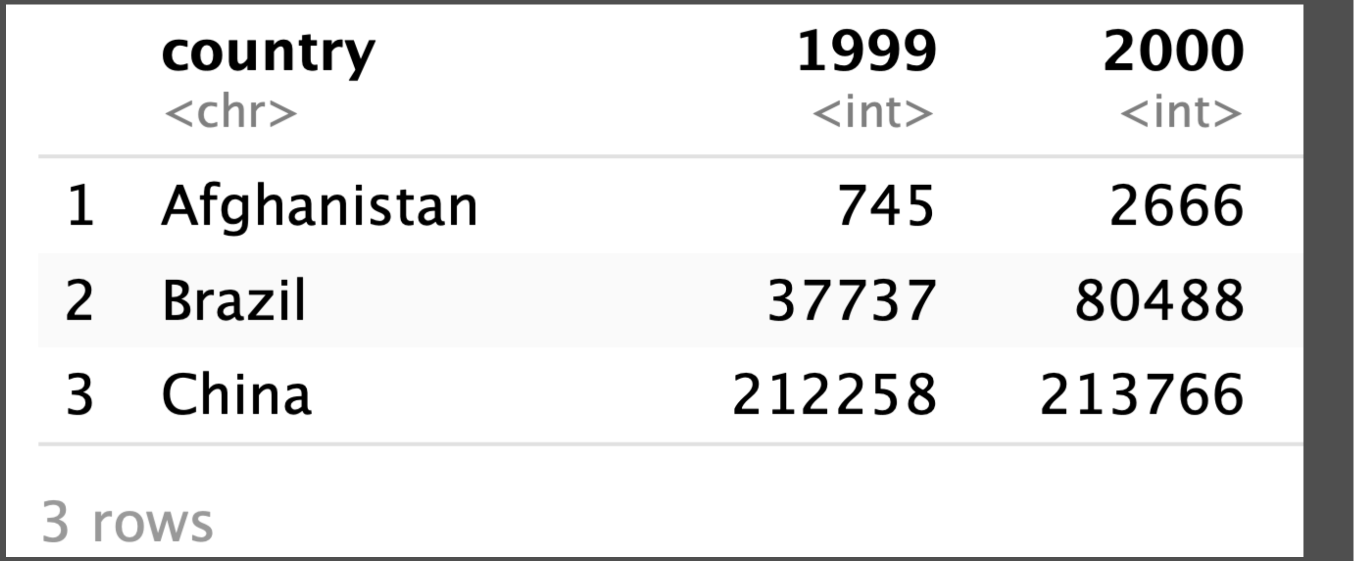


	country <chr>	1999 <int>	2000 <int>
1	Afghanistan	745	2666
2	Brazil	37737	80488
3	China	212258	213766

3 rows

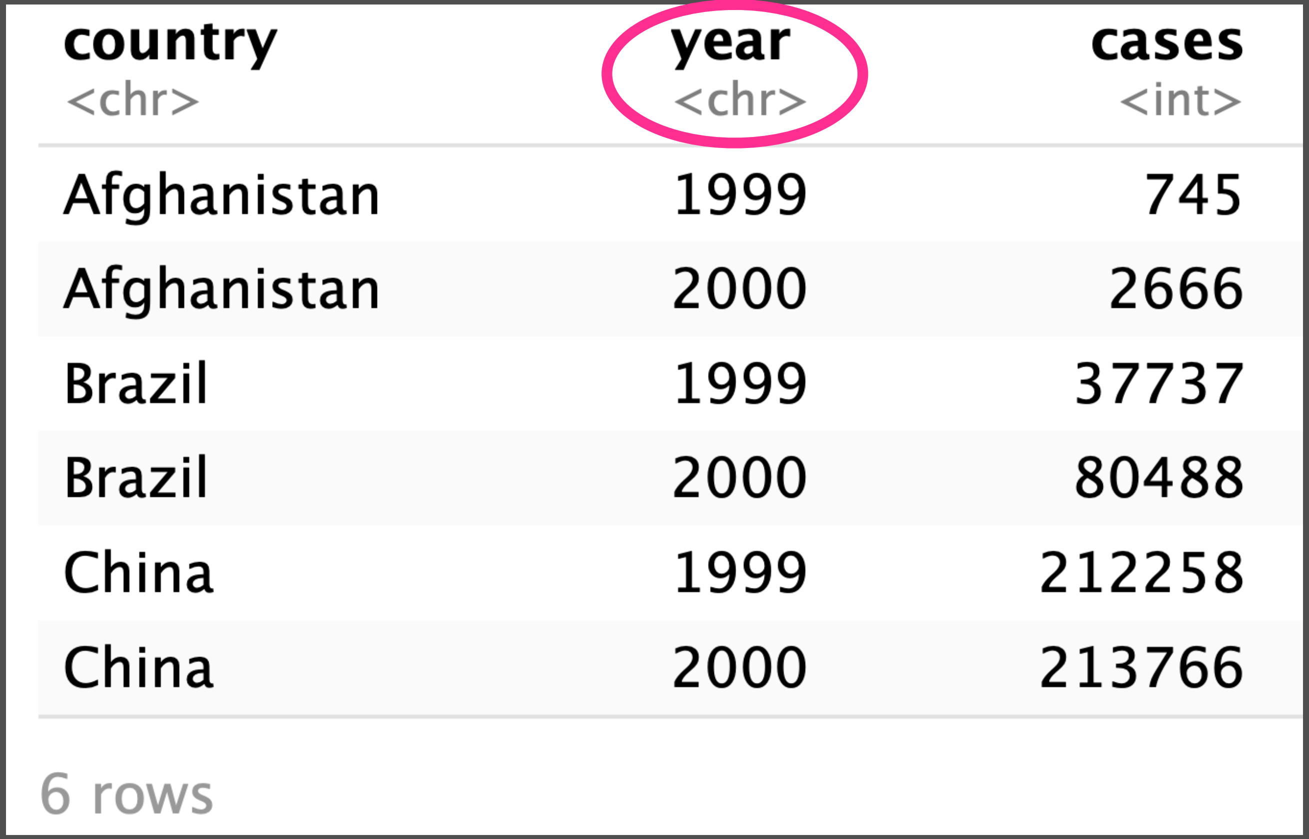
SOLUTION

```
table4a %>%
  pivot_long(cols = 2:3, names_to = "year", values_to = "cases")
```



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

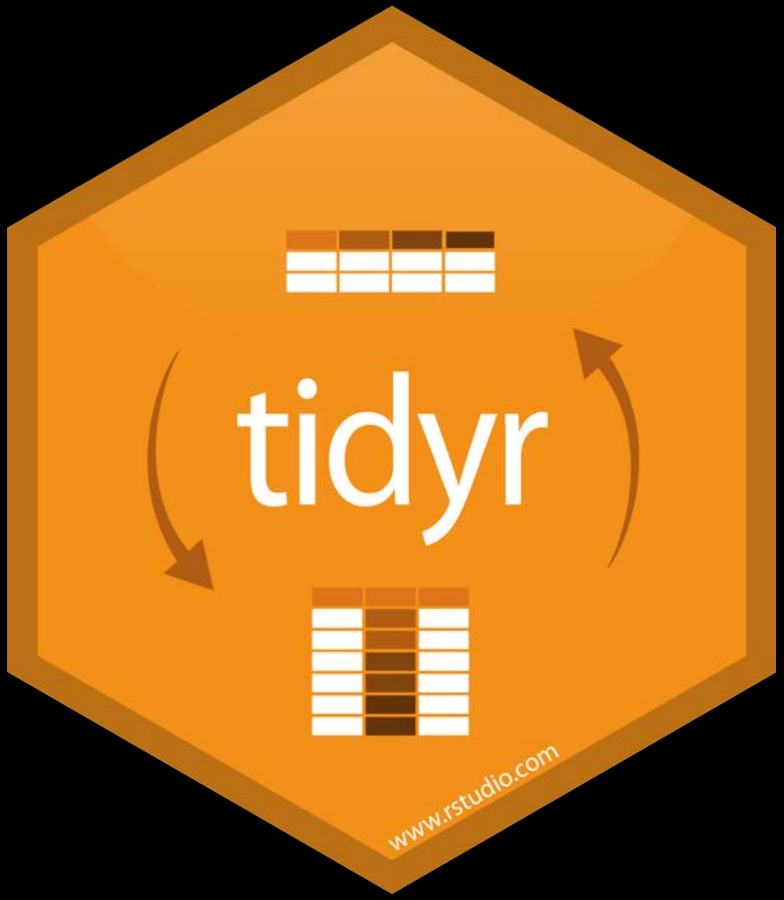
3 rows



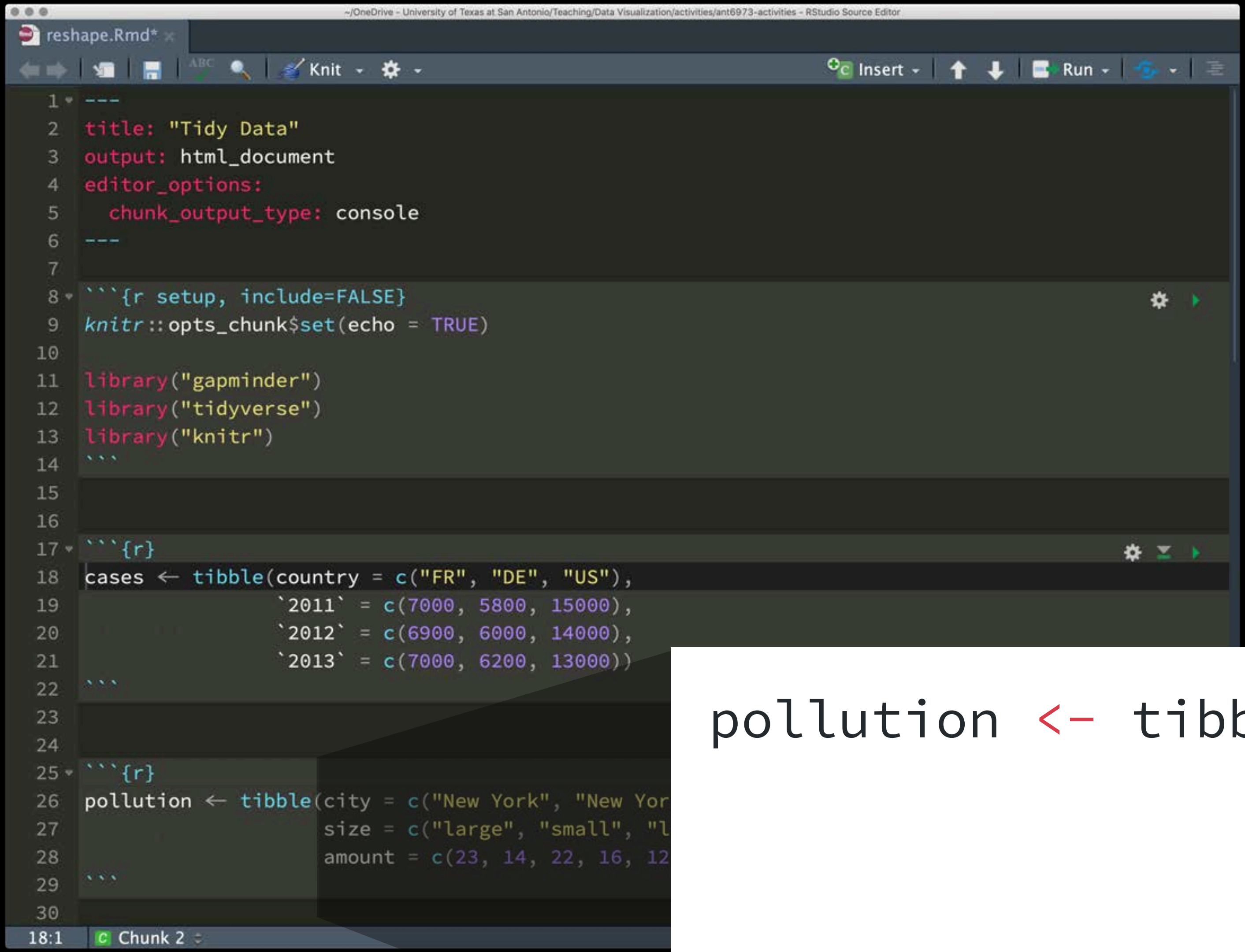
country	year	cases
<chr>	<chr>	<int>
Afghanistan	1999	745
Afghanistan	2000	2666
Brazil	1999	37737
Brazil	2000	80488
China	1999	212258
China	2000	213766

6 rows

pivot_wide()



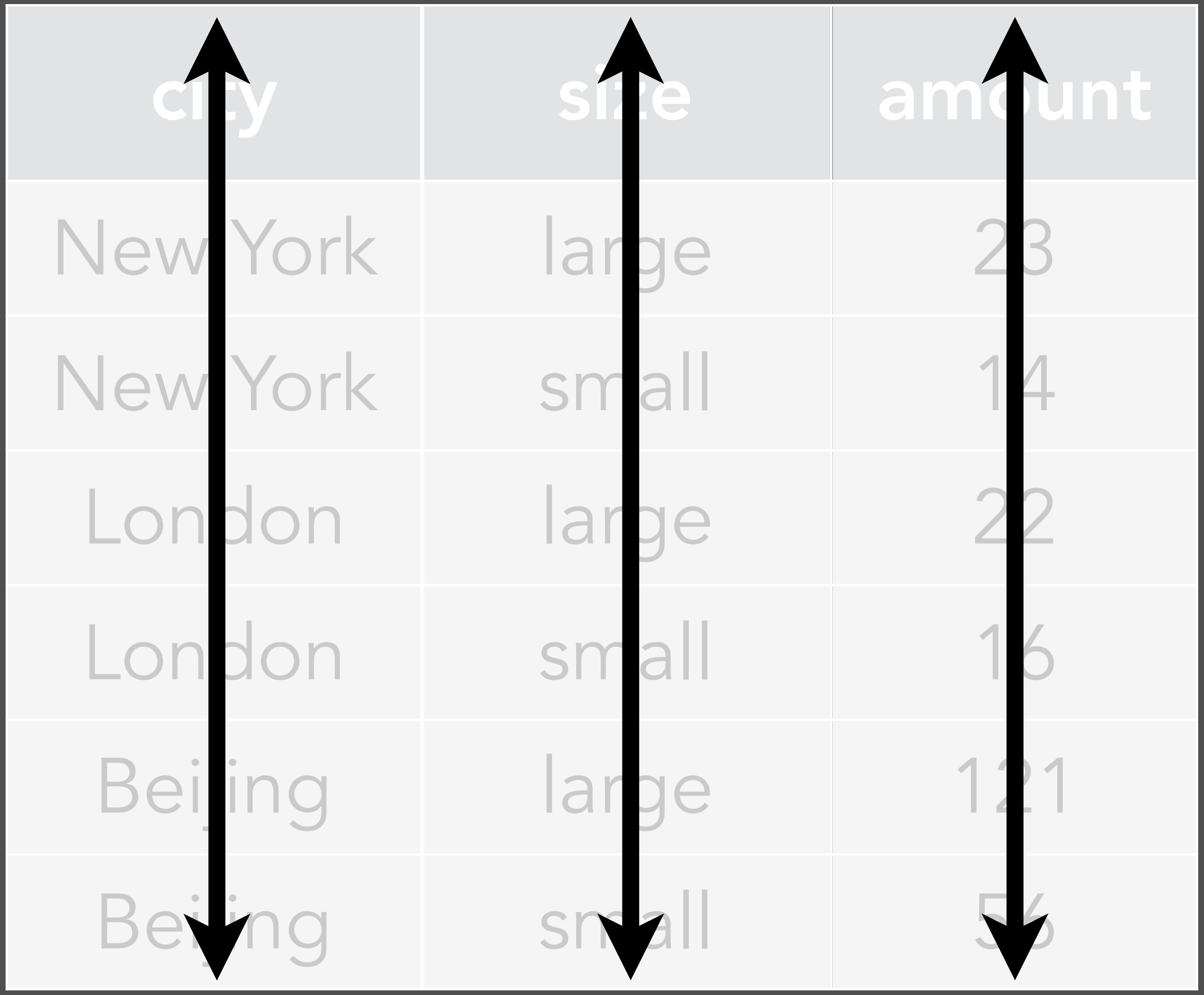
PRACTICE DATA


~/OneDrive - University of Texas at San Antonio/Teaching/Data Visualization/activities/ant6973-activities - RStudio Source Editor
reshape.Rmd*
Knit
Insert
Run
18:1
Chunk 2

~~~
---
title: "Tidy Data"
output: html_document
editor_options:
  chunk_output_type: console
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)

library("gapminder")
library("tidyverse")
library("knitr")
```


```{r}
cases <- tibble(country = c("FR", "DE", "US"),
                `2011` = c(7000, 5800, 15000),
                `2012` = c(6900, 6000, 14000),
                `2013` = c(7000, 6200, 13000))
```


```{r}
pollution <- tibble(city = c("New York", "New Yor
                    size = c("large", "small", "l
                    amount = c(23, 14, 22, 16, 12
```

~~~

```
pollution <- tibble(city = ...,
                    size = ...,
                    amount = ...)
```

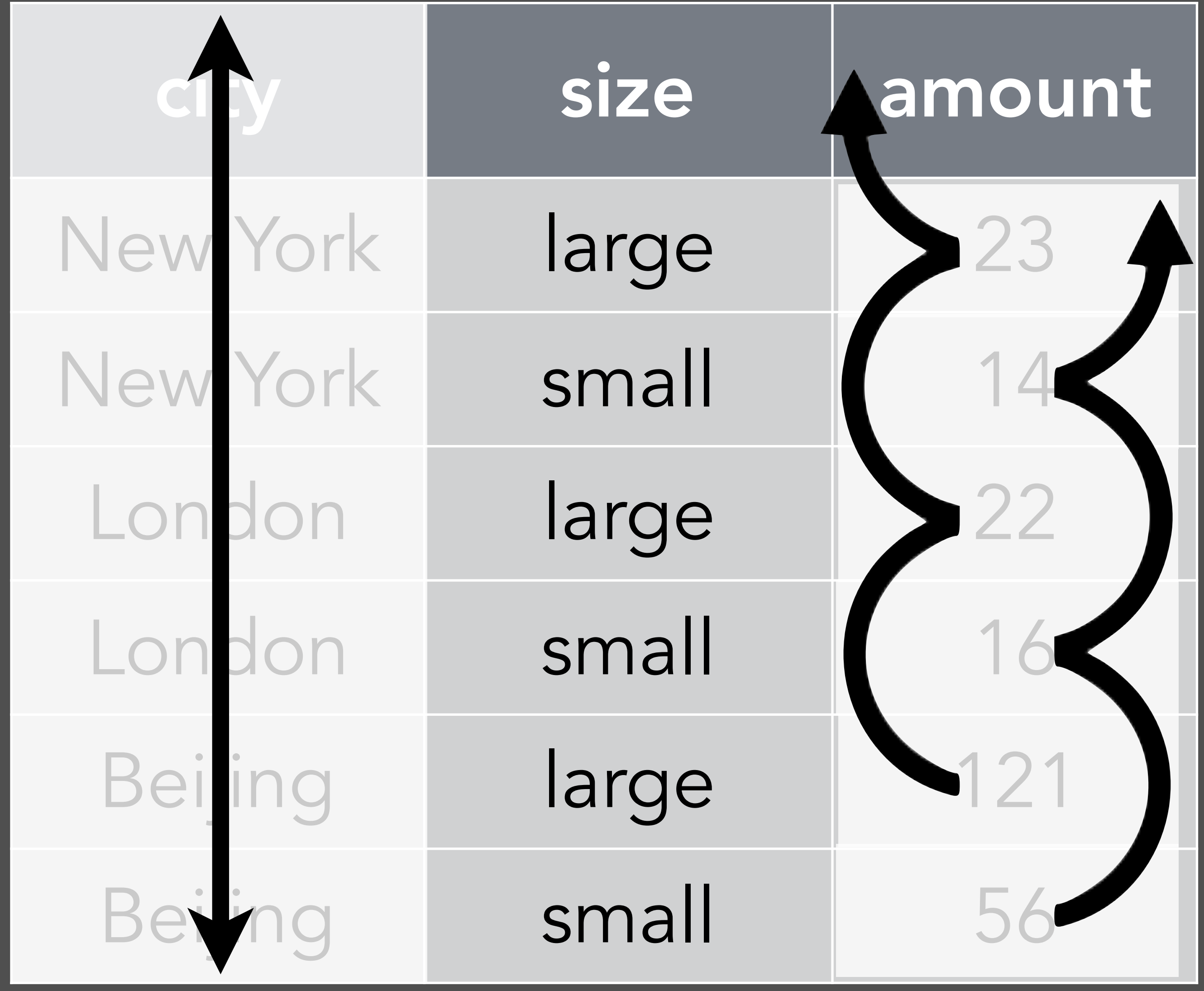
WHAT ARE THE VARIABLES?



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

- City
- Particle size
- Amount of particulate

TO MAKE A SCATTER PLOT OF OF LARGE VS. SMALL PARTICLE AMOUNTS, WHAT COLUMNS WOULD WE NEED?



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

- City
- Amount of large
- Amount of small

What is a variable and an observation may depend on your immediate goal.!

ACTIVITY 3

- Plan (e.g., draw on paper) 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

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

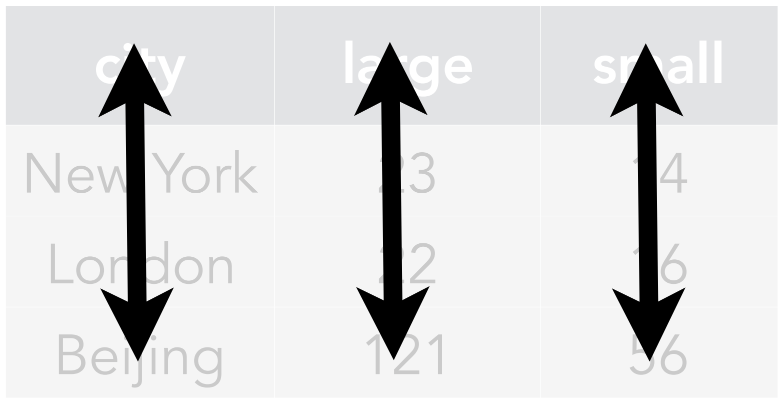
city	size	amount
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

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

`pivot_wide()`

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

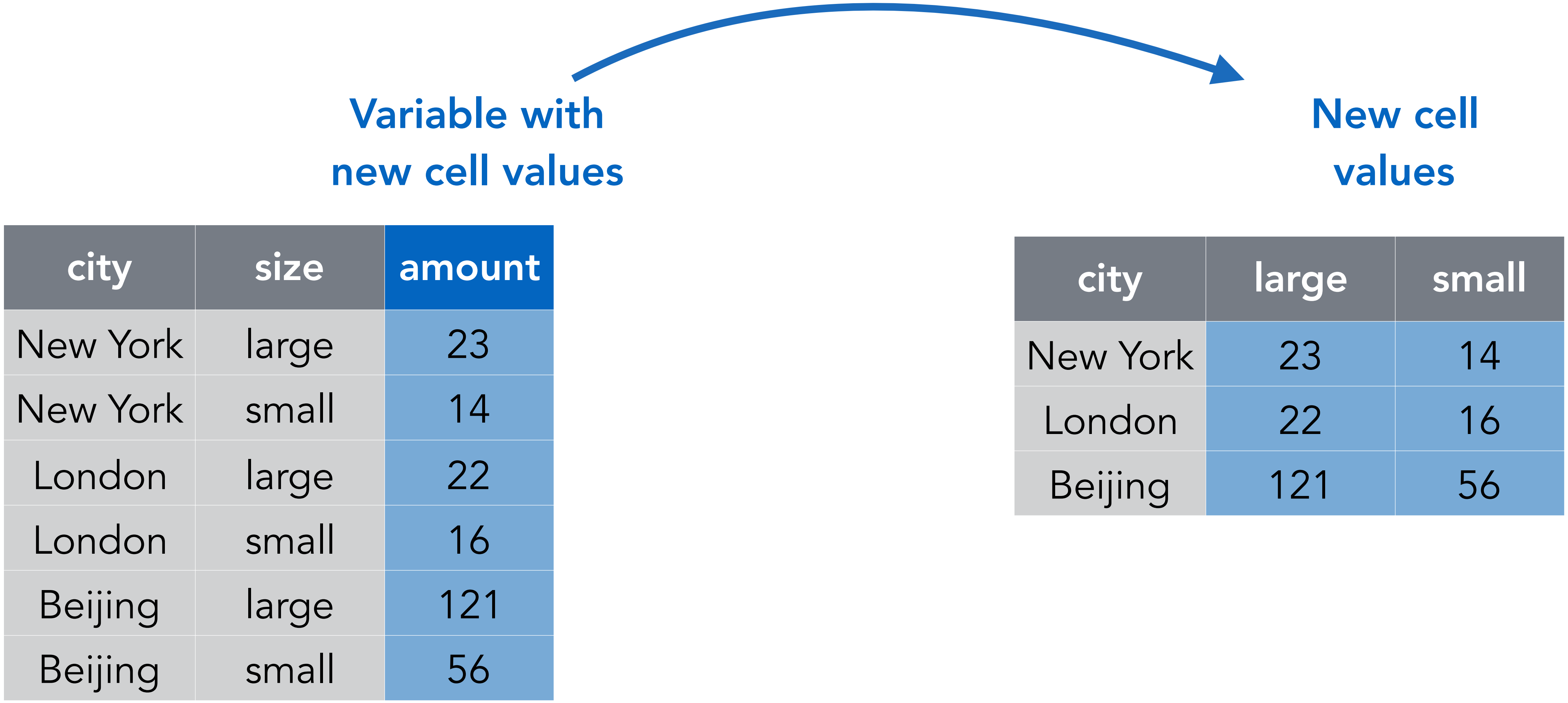
Variable with new column names

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

New columns

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

Transformation logic: column names **FROM** old variable



Variable with
new cell values

New cell
values

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

Transformation logic: cell values **FROM** old variable

pivot_wide()

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

pivot_wide()

Data to reshape

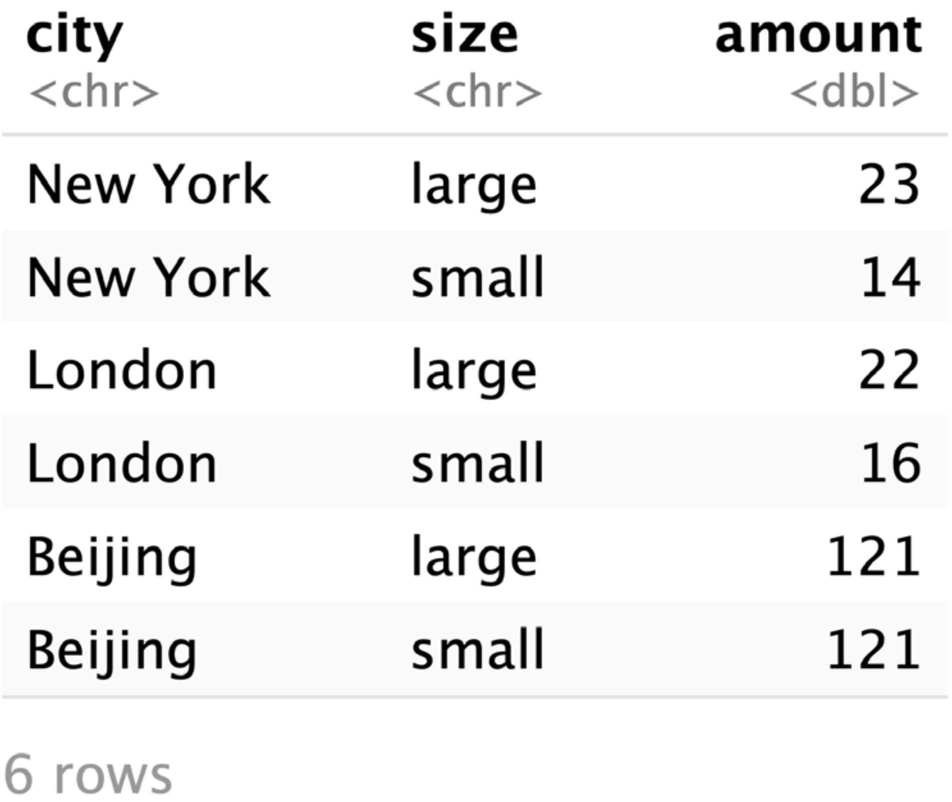
```
pollution %>%
  pivot_wide(names_from = size,
             values_from = amount)
```

Old variable that becomes new column names

Old variable that becomes new cell values

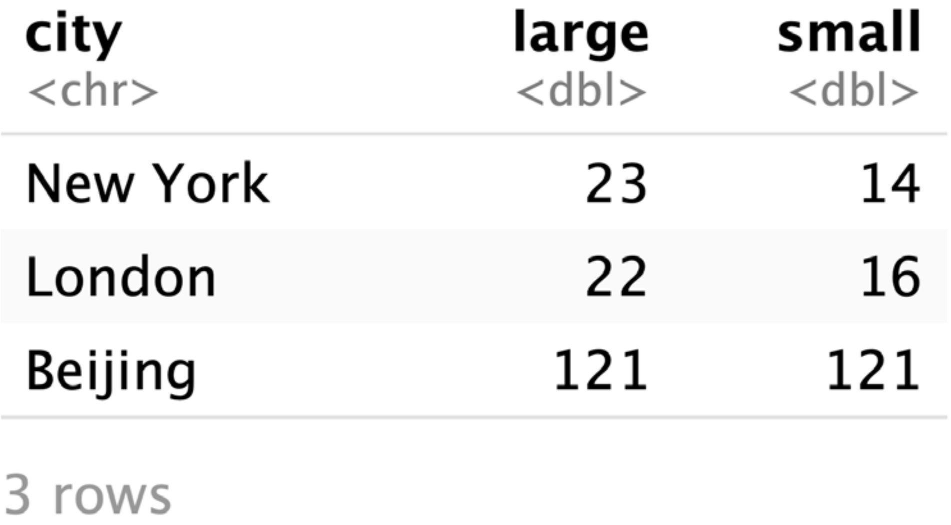
pivot_wide()

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



city	size	amount
<chr>	<chr>	<dbl>
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	121

6 rows

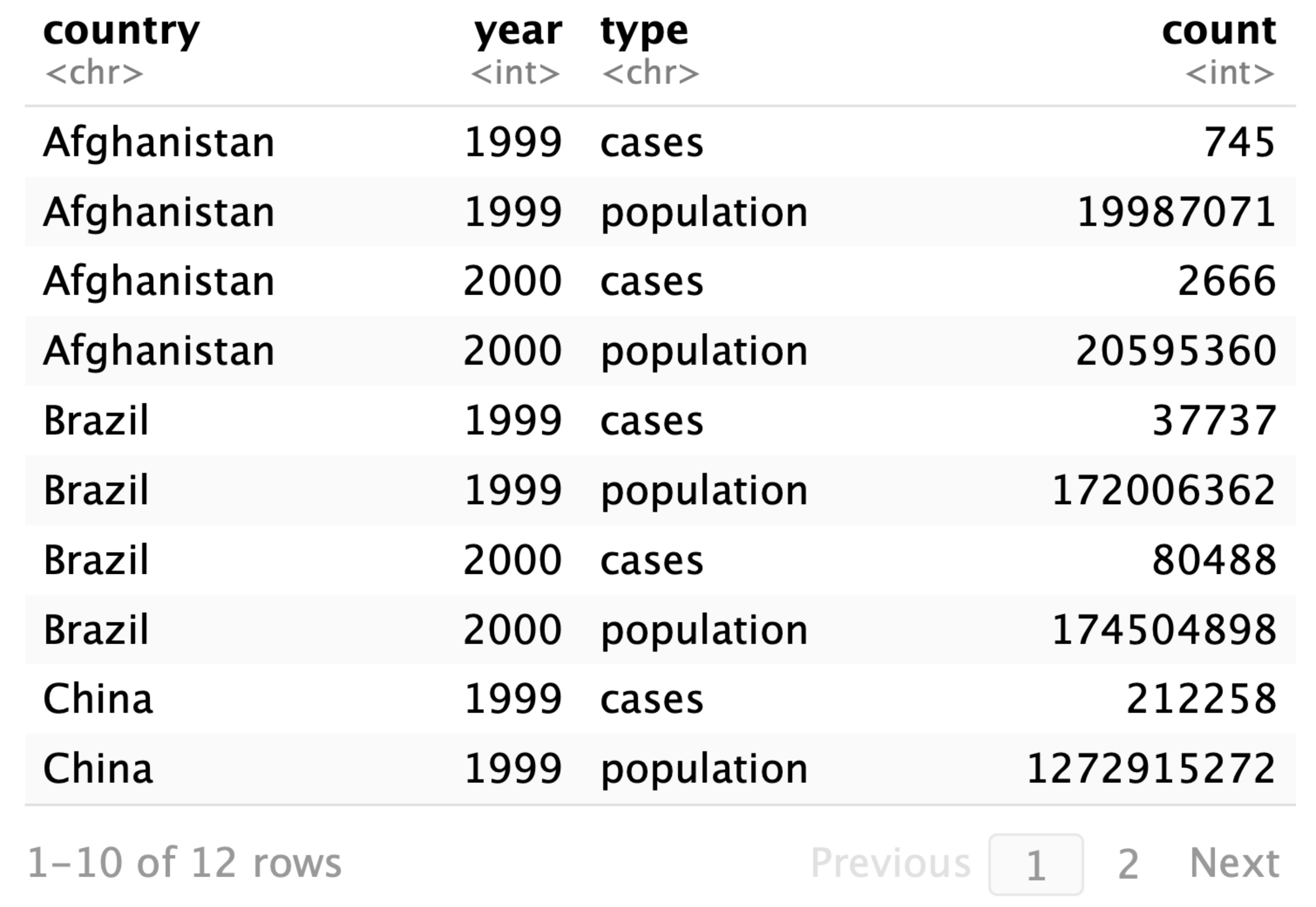


city	large	small
<chr>	<dbl>	<dbl>
New York	23	14
London	22	16
Beijing	121	121

3 rows

ACTIVITY 4

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

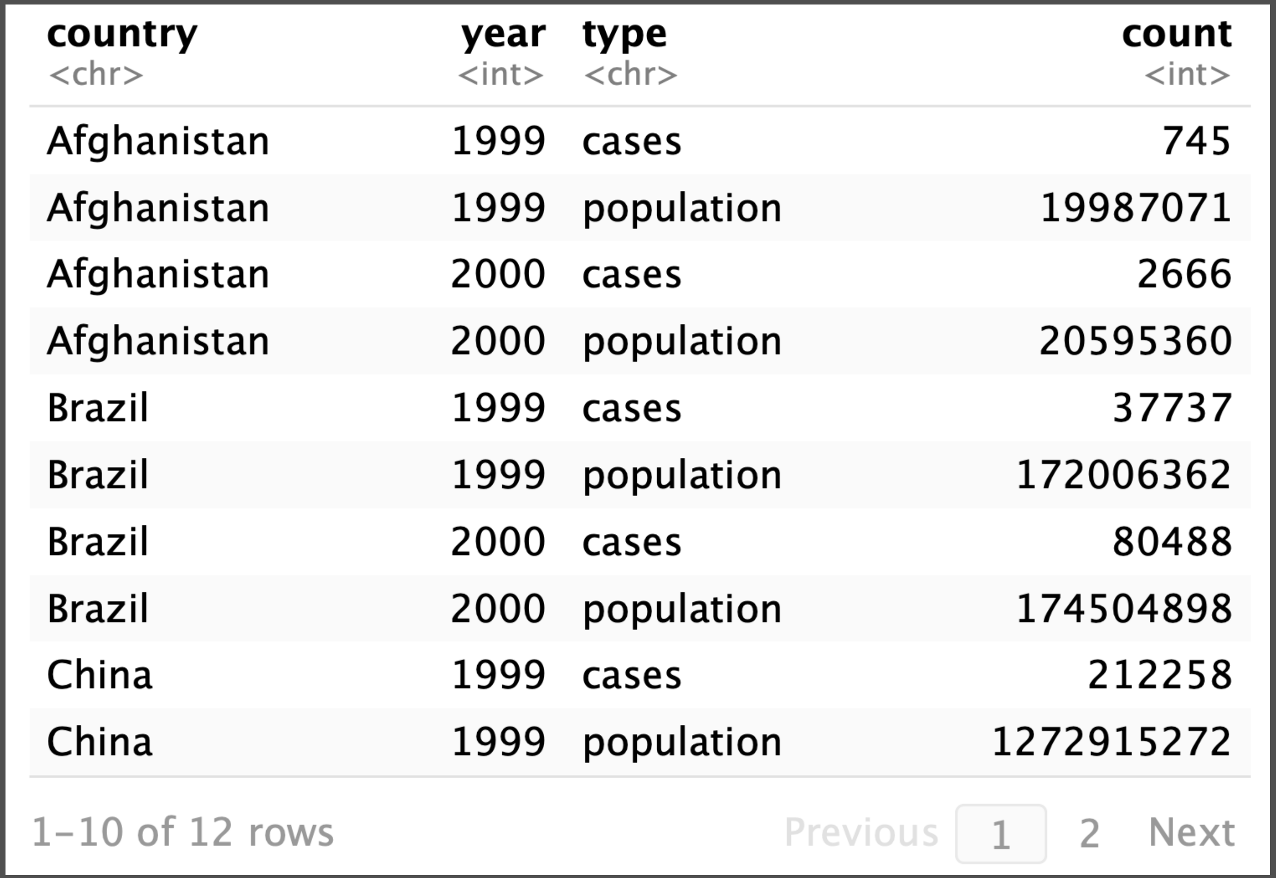


country	year	type	count
<chr>	<int>	<chr>	<int>
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999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

1–10 of 12 rows

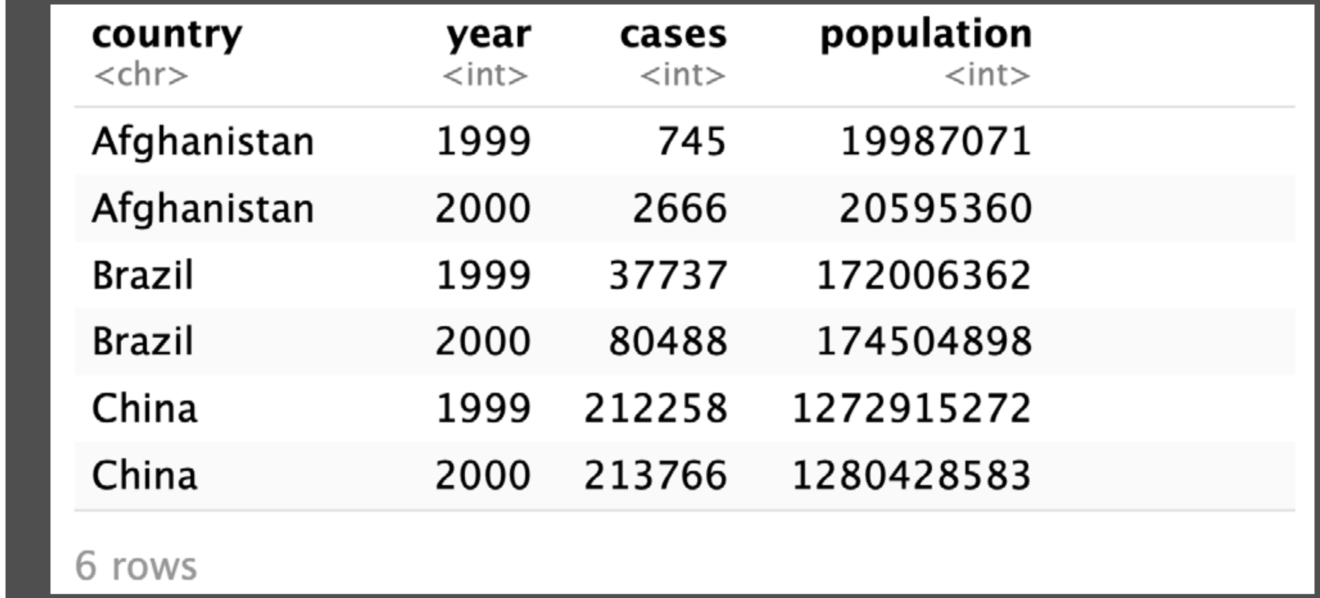
SOLUTION

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



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
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272

1–10 of 12 rows Previous 1 2 Next



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

6 rows

ACTIVITY 5: TIDY DATA

- Finish the last activity in tidy-data.Rmd
- Send me the html report

ACKNOWLEDGEMENTS

- Some ideas, examples, and figures from RStudio webinars, which are licensed CC by SA.