

Tidy Data

Jake Thompson

 wjakethompson.com

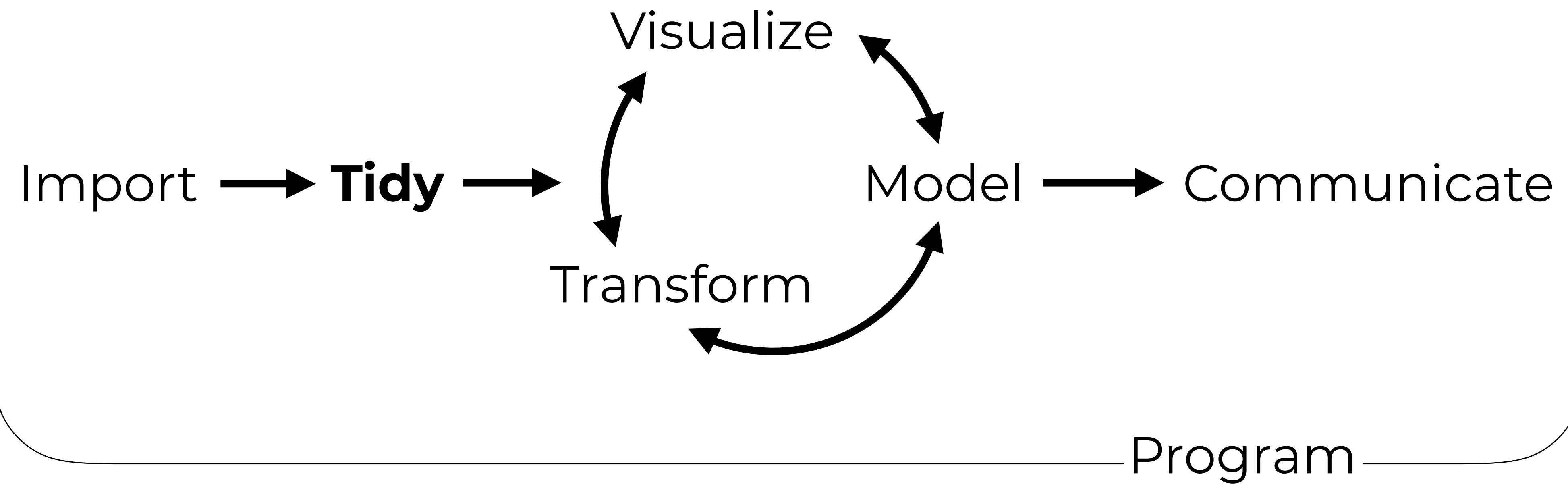
 /  @wjakethompson



Your Turn 0

- Open **05-Tidy.Rmd**
- Run the setup chunk





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.



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**

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



Composable

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



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.

Tidy Data

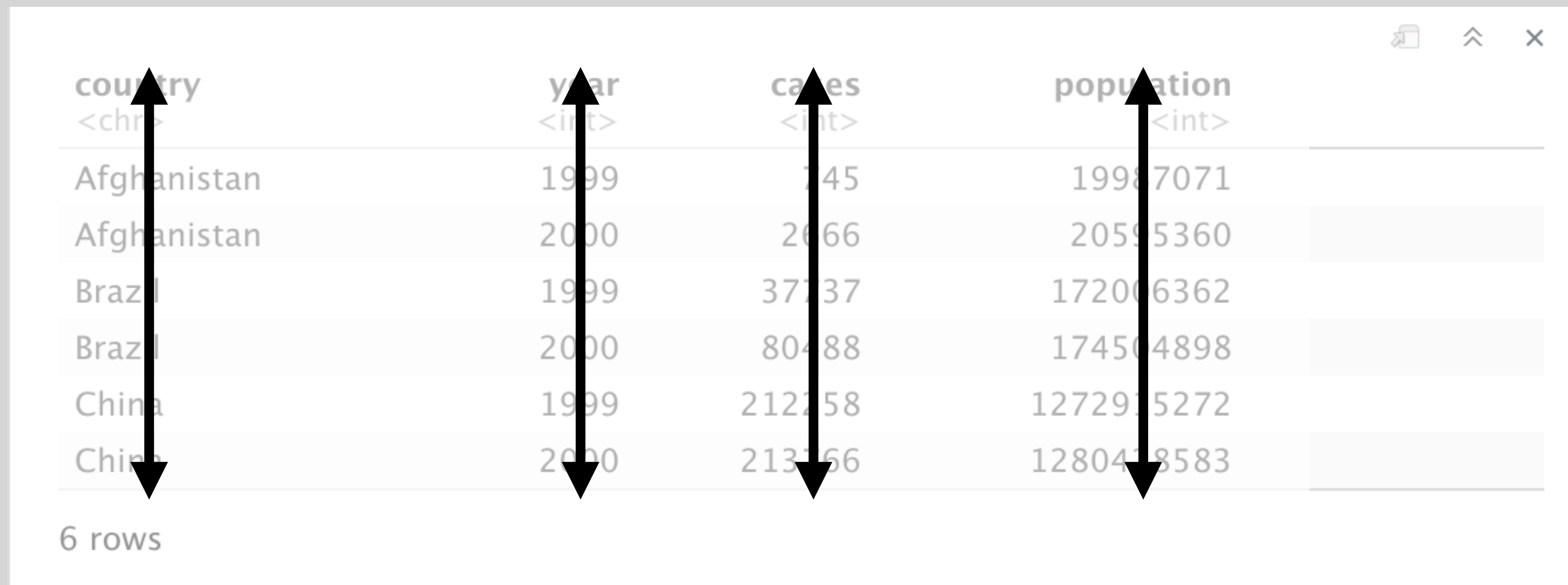


"Data are not just number,
they are numbers with a context."

–George Cobb and David Moore (1997)

Consider

What are the variables in this data set?

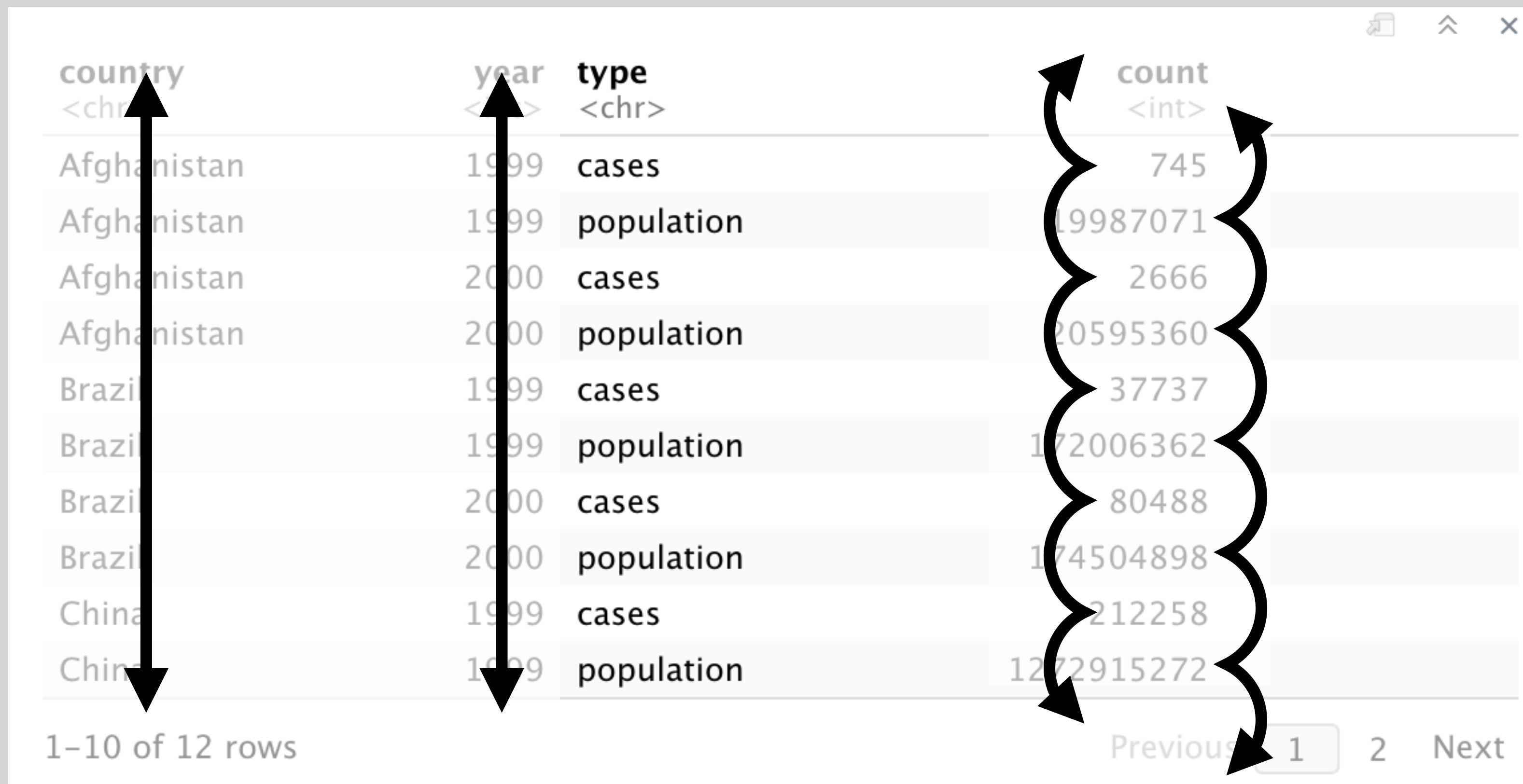


country <chr>	year <int>	cases <int>	population <int>
Afghanistan	1999	745	19987071
Afghanistan	2000	2066	20595360
Brazil	1999	37137	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213166	1280418583

6 rows

Consider

What are the variables in this data set?



A screenshot of a data table interface. The table has four columns: 'country' (type <chr>), 'year' (type <chr>), 'type' (type <chr>), and 'count' (type <int>). The rows show data for Afghanistan, Brazil, and China across the years 1999 and 2000, categorized by 'cases' and 'population'. The 'count' column contains numerical values. A vertical double-headed arrow is positioned to the left of the 'country' column, another to the left of the 'year' column, and a large wavy double-headed arrow is positioned to the left of the 'count' column. The interface includes a 'Previous' button, a page number '1', and a 'Next' button at the bottom right. The text '1-10 of 12 rows' is visible at the bottom left of the table area.

country <chr>	year <chr>	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

Previous 1 2 Next

This isn't tidy

What are the variables in this data set?

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

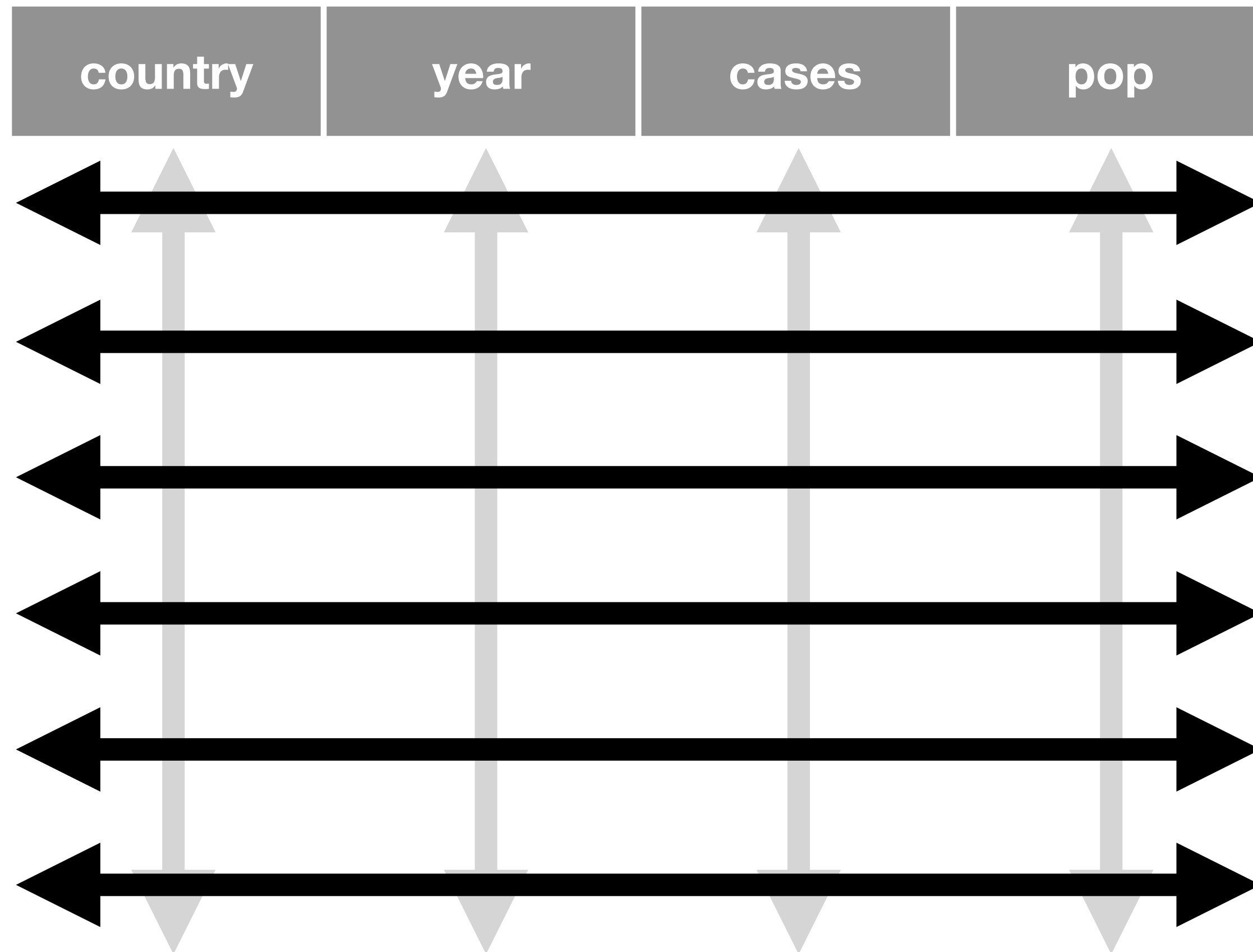
Previous 1 2 Next

Contains two variables

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

–Garrett Grolemund

Tidy data



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**

Your Turn 1

Is **bp_systolic** tidy? What are the variables?

01:00




```
bp_systolic
```

```
# A tibble: 3 x 4
```

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

Time

Subject

Systolic blood
pressure

bp_systolic2

```
bp_systolic2
# A tibble: 6 x 3
  subject_id time systolic
    <dbl> <dbl>    <dbl>
1         1     1     120
2         1     2     118
3         1     3     121
4         2     1     125
5         2     2     131
6         3     1     141
```

Your Turn 2

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

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

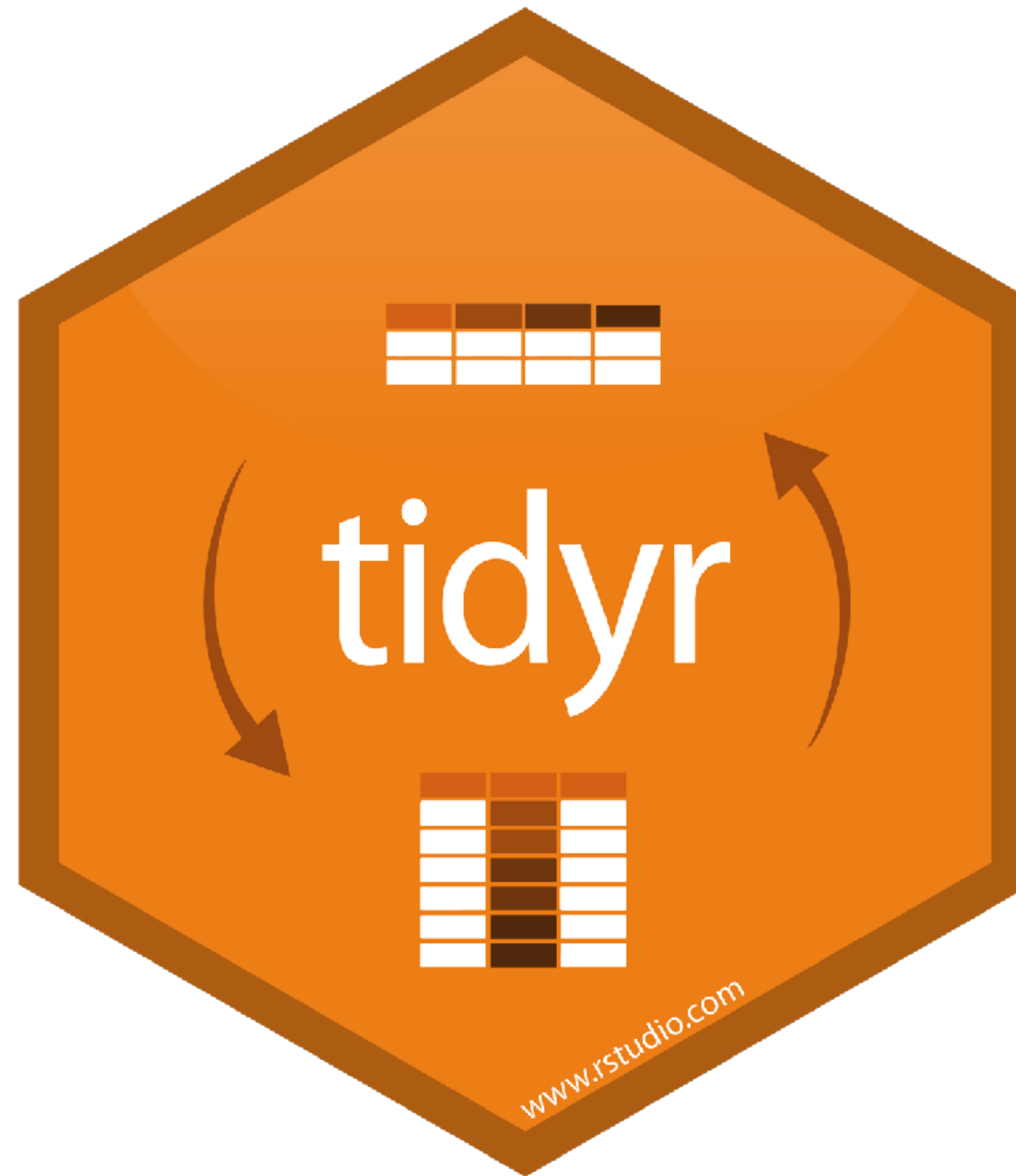
02 : 00




```
bp_systolic2 %>%  
  group_by(subject_id) %>%  
  summarize(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



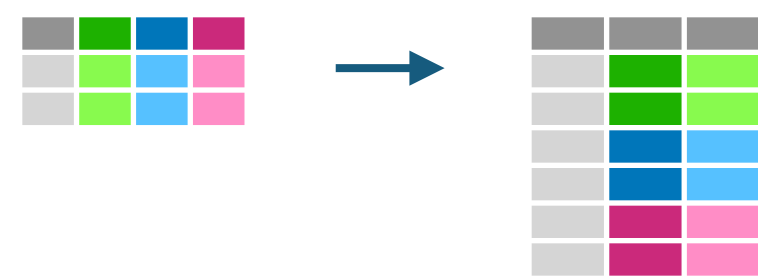


In R4DS
Tidy Data

tidyr verbs



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



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

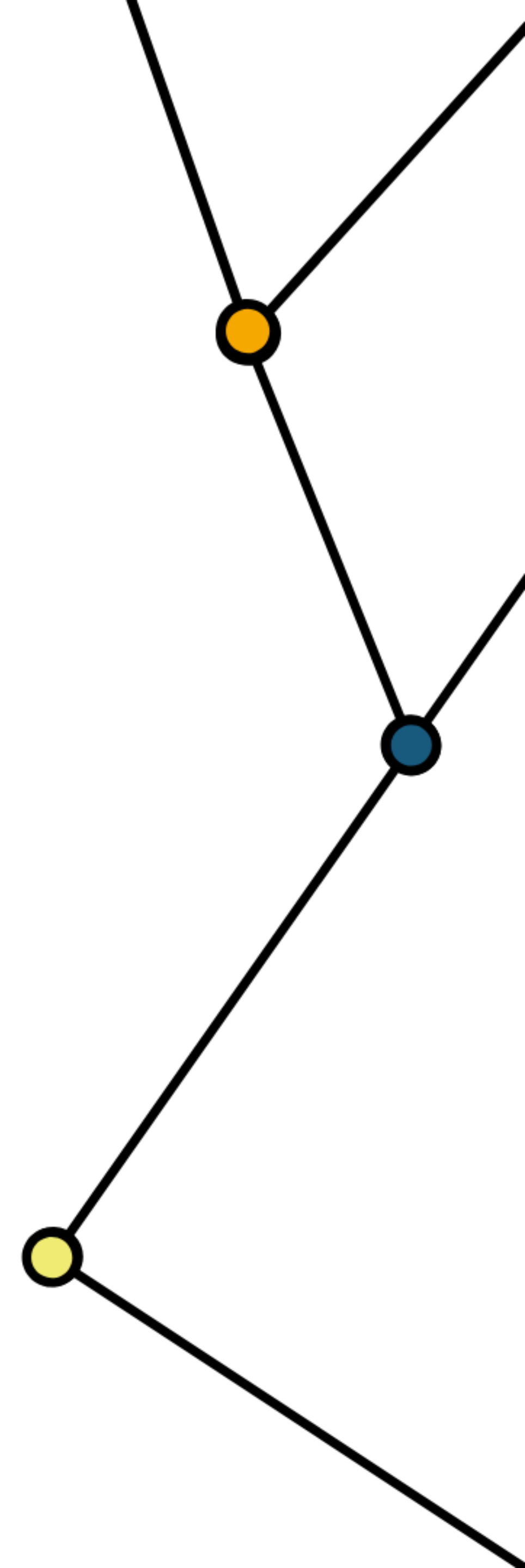
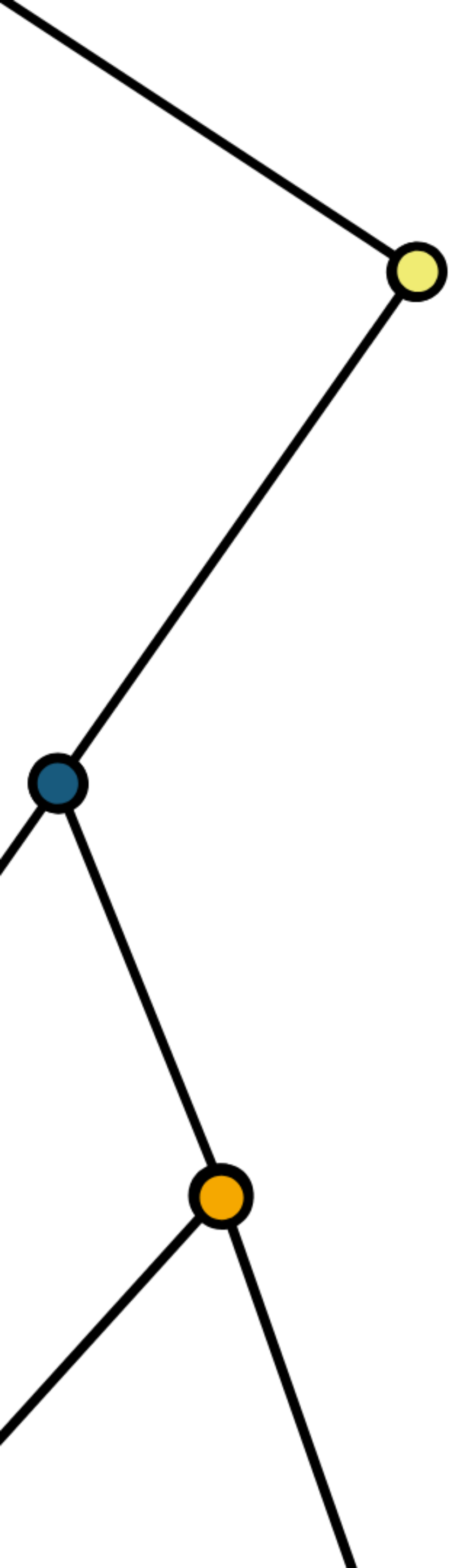


Split a column with **separate()**



Unite columns with **unite()**

pivot_longer()



Toy data for practice

```
05-Tidy.Rmd
1 ---
2 title: "Tidy Data"
3 output: html_notebook
4 editor_options:
5   chunk_output_type: inline
6 ---
7
8 <!-- This file by Jake Thompson is licensed under a Creative Commons Attribution
9 4.0 International License, adapted from the original work at
10 https://github.com/rstudio/master-the-tidyverse by RStudio. -->
11
12 {r setup}
13 library(tidyverse)
14 library(babynames)
15
16 # Toy data
17 cases <- tribble(
18   ~Country, ~"2011", ~"2012", ~"2013",
19   "FR",      7000,    6900,    7000,
20   "DE",      5800,    6000,    6200,
21   "US",     15000,   14000,   13000
22 )
23
24 pollution <- tribble(
25   ~city, ~size, ~amount,
26   "New York", "large",    23,
27   "New York", "small",    14,
28   "London", "large",     22,
29   "London", "small",     16,
30   "Beijing", "large",    121,
31   "Beijing", "small",     56
32 )
33
34 bp_systolic <- tribble(
35   ~subject_id, ~time_1, ~time_2, ~time_3,
36   "S1", 120, 110, 100,
37   "S2", 110, 100, 90,
38   "S3", 100, 90, 80,
39   "S4", 90, 80, 70,
40   "S5", 80, 70, 60,
41   "S6", 70, 60, 50,
42   "S7", 60, 50, 40,
43   "S8", 50, 40, 30,
44   "S9", 40, 30, 20,
45   "S10", 30, 20, 10
46 )
```

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

Consider

What are the variables in this **cases**?

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

Consider

What are the variables in this **cases**?

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

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*, and *n*.

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

04 : 00

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	
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

`pivot_longer()`

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

	1	2
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

former
column
names

names_to

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

names_to		values_to
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

former
cells

pivot_longer()

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

**Data frame to
reshape**

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

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

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

Your Turn 4

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

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

03 : 00

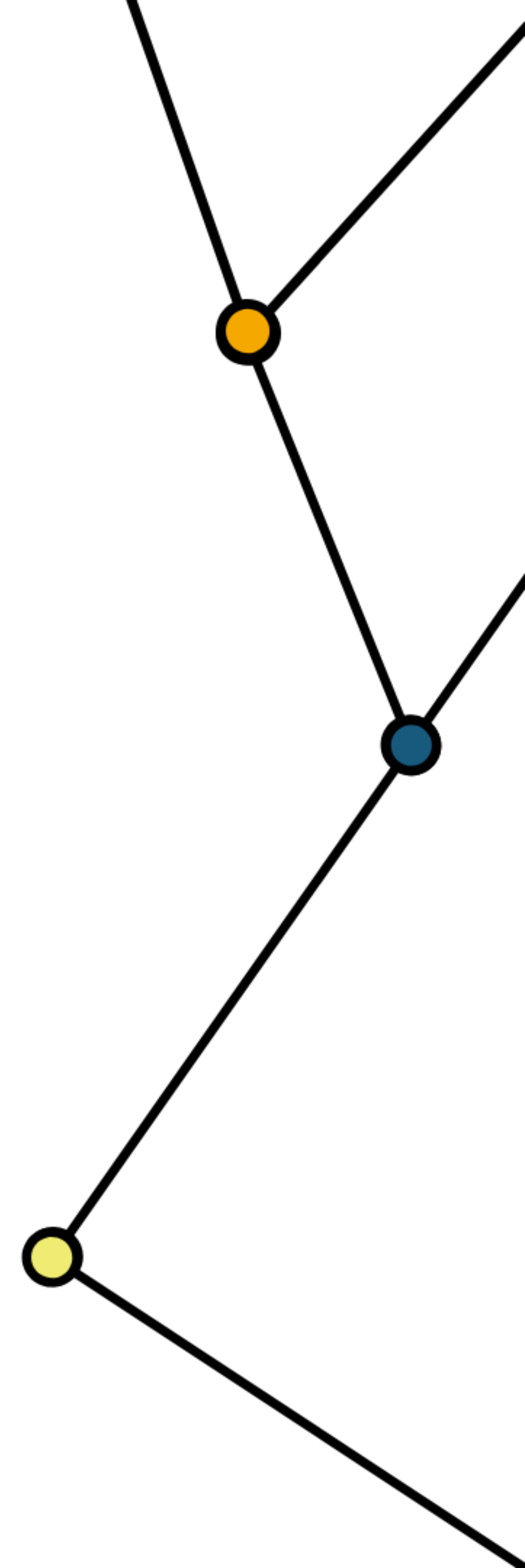
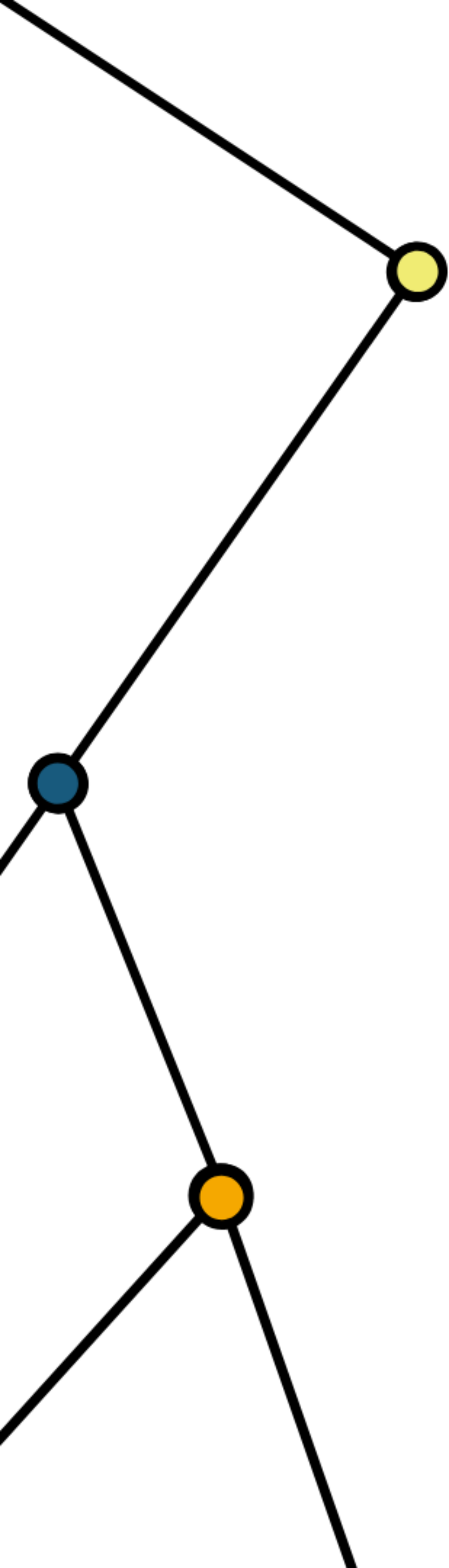
```
table4a %>%  
  pivot_longer(-country, names_to = "year", values_to = "n")  
# A tibble: 6 x 3  
  country      year      n  
  <chr>      <chr> <int>  
1 Afghanistan 1999      745  
2 Afghanistan 2000     2666  
3 Brazil      1999    37737  
4 Brazil      2000    80488  
5 China       1999   212258  
6 China       2000   213766
```

```
table4a %>%  
  pivot_longer(-country, names_to = "year", values_to = "n",  
               col_ptypes = list(year = integer()))
```

```
# A tibble: 6 x 3
```

	country	year	n
	<chr>	<int>	<int>
1	Afghanistan	1999	745
2	Afghanistan	2000	2666
3	Brazil	1999	37737
4	Brazil	2000	80488
5	China	1999	212258
6	China	2000	213766

pivot_wider()



Toy data for practice

```
05-Tidy.Rmd x
1 ---
2 title: "Tidy Data"
3 output: html_notebook
4 editor_options:
5   chunk_output_type: inline
6 ---
7
8 <!-- This file by Jake Thompson is licensed under a Creative Commons Attribution
9 4.0 International License, adapted from the original work at
10 https://github.com/rstudio/master-the-tidyverse by RStudio. -->
11
12 {r setup}
13 library(tidyverse)
14 library(babynames)
15
16 # Toy data
17 cases <- tribble(
18   ~Country, ~"2011", ~"2012", ~"2013",
19   "FR", 7000, 6900, 7000,
20   "DE", 5800, 6000, 6200,
21   "US", 15000, 14000, 13000
22 )
23
24 pollution <- tribble(
25   ~city, ~size, ~amount,
26   "New York", "large", 23,
27   "New York", "small", 14,
28   "London", "large", 22,
29   "London", "small", 16,
30   "Beijing", "large", 121,
31   "Beijing", "small", 56
32 )
33
34 bp_systolic <- tribble(
35   ~subject_id, ~time_1, ~time_2, ~time_3,
36   "S1", 120, 125, 130,
37   "S2", 110, 115, 120,
38   "S3", 100, 105, 110,
39   "S4", 90, 95, 100,
40   "S5", 80, 85, 90
41 )
42
43 # Your Turn 3
```

```
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 this **pollution**?

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 this **pollution**?

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 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*, and *small*.

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

03 : 00

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_wider()`

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

names_from

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

new column
names

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

names_from values_from

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

new cells

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**

column to use for keys
(becomes new column
names)

column to use for values
(becomes new column
cells)

Your Turn 6

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

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

Previous 1 2 Next

03 : 00

```
table2 %>%  
  pivot_wider(names_from = "type", values_from = "count")  
# A tibble: 6 x 4  
  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
```



separate()* and *unite()

Toy data for practice

```
05-Tidy.Rmd
9
10 {r setup}
11 library(tidyverse)
12 library(babynames)
13
14 # Toy data
15 cases <- tribble(
16   ~country, ~"2011", ~"2012", ~"2013",
17   "FR", 7000, 6900, 7000,
18   "DE", 5800, 6000, 6200,
19   "US", 15000, 14000, 13000
20 )
21
22 pollution <- tribble(
23   ~city, ~size, ~amount,
24   "New York", "large", 23,
25   "New York", "small", 14,
26   "London", "large", 22,
27   "London", "small", 16,
28   "Beijing", "large", 121,
29   "Beijing", "small", 56
30 )
31
32 scores <- tribble(
33   ~name, ~house, ~score,
34   "Ronald Weasley", "Gryffindor", 78,
35   "Harry Potter", "Gryffindor", 85,
36   "Hermione Granger", "Gryffindor", 100,
37   "Justin Finch-Fletchley", "Hufflepuff", 87,
38   "Hannah Abbot", "Hufflepuff", 92,
39   "Susan Bones", "Hufflepuff", 93,
40   "Anthony Goldstein", "Ravenclaw", 84,
41   "Michael Corner", "Ravenclaw", 93,
42   "Padma Patil", "Ravenclaw", 97,
43   "Vincent Crabbe", "Slytherin", 61,
44   "Gregory Goyle", "Slytherin", 61,
45   "Draco Malfoy", "Slytherin", 92
46 )
47
48
49
```

```
scores <- tribble(
  ~name, ~house, ~score,
  "Ronald Weasley", "Gryffindor", 78,
  "Harry Potter", "Gryffindor", 85,
  "Hermione Granger", "Gryffindor", 100,
  "Justin Finch-Fletchley", "Hufflepuff", 87,
  "Hannah Abbot", "Hufflepuff", 92,
  "Susan Bones", "Hufflepuff", 93,
  "Anthony Goldstein", "Ravenclaw", 84,
  "Michael Corner", "Ravenclaw", 93,
  "Padma Patil", "Ravenclaw", 97,
  "Vincent Crabbe", "Slytherin", 61,
  "Gregory Goyle", "Slytherin", 61,
  "Draco Malfoy", "Slytherin", 92
)
```

Consider

What are the variables in this **scores**?

name	code	score
Ronald Weasley	Gryffindor	78
Harry Potter	Gryffindor	85
Hermione Granger	Gryffindor	100
Justin Finch-Fletchley	Hufflepuff	87
Hannah Abbot	Hufflepuff	92
Susan Bones	Hufflepuff	93
Anthony Goldstein	Ravenclaw	84
Michael Corner	Ravenclaw	93
Padma Patil	Ravenclaw	97
Vincent Crabbe	Slytherin	61
Gregory Goyle	Slytherin	61
Draco Malfoy	Slytherin	92

Consider

What are the variables in this **scores**?

name	house	score
Ronald Weasley	Gryffindor	78
Harry Potter	Gryffindor	85
Hermione Granger	Gryffindor	100
Justin Finch-Fletchley	Hufflepuff	87
Hannah Abbot	Hufflepuff	92
Susan Bones	Hufflepuff	93
Anthony Goldstein	Ravenclaw	84
Michael Corner	Ravenclaw	93
Padma Patil	Ravenclaw	97
Vincent Crabbe	Slytherin	61
Gregory Goyle	Slytherin	61
Draco Malfoy	Slytherin	92

1 variable? or 2?

separate()

```
separate(data, col, into, sep = "[^:alnum:]]+", ...)
```

**Data frame to
tidy**

**column to
separate**

**new columns to
be created**

**what divides the
pieces of
information**

```
scores %>%
  separate(name, into = c("first", "last"))
# A tibble: 12 x 4
   first    last    house    score
  <chr>   <chr>   <chr>   <dbl>
1 Ronald  Weasley Gryffindor    78
2 Harry   Potter  Gryffindor    85
3 Hermione Granger  Gryffindor   100
4 Justin   Finch   Hufflepuff    87
5 Hannah  Abbot   Hufflepuff    92
6 Susan    Bones   Hufflepuff    93
7 Anthony Goldstein Ravenclaw    84
8 Michael  Corner  Ravenclaw    93
9 Padma    Patil   Ravenclaw    97
10 Vincent Crabbe  Slytherin    61
11 Gregory Goyle   Slytherin    61
12 Draco   Malfoy  Slytherin    92
Warning message:
Expected 2 pieces. Additional pieces discarded in 1 rows [4].
```

```
scores %>%
  separate(name, into = c("first", "last"))
# A tibble: 12 x 4
   first    last    house    score
  <chr>   <chr>   <chr>   <dbl>
1 Ronald  Weasley Gryffindor    78
2 Harry   Potter  Gryffindor    85
3 Hermione Granger  Gryffindor   100
4 Justin  Finch   Hufflepuff    87
5 Hannah  Abbot   Hufflepuff    92
6 Susan   Bones   Hufflepuff    93
7 Anthony Goldstein Ravenclaw    84
8 Michael Corner   Ravenclaw    93
9 Padma   Patil    Ravenclaw    97
10 Vincent Crabbe   Slytherin    61
11 Gregory Goyle    Slytherin    61
12 Draco   Malfoy   Slytherin    92
```

Warning message:
Expected 2 pieces. Additional pieces discarded in 1 rows [4].

```
scores %>%
  separate(name, into = c("first", "last"), sep = " ")
# A tibble: 12 x 4
   first    last      house    score
  <chr>   <chr>   <chr>   <dbl>
1 Ronald  Weasley Gryffindor    78
2 Harry   Potter  Gryffindor    85
3 Hermione Granger  Gryffindor   100
4 Justin  Finch-Fletchley Hufflepuff    87
5 Hannah  Abbot    Hufflepuff    92
6 Susan   Bones    Hufflepuff    93
7 Anthony Goldstein Ravenclaw     84
8 Michael Corner   Ravenclaw     93
9 Padma   Patil     Ravenclaw     97
10 Vincent Crabbe  Slytherin     61
11 Gregory Goyle    Slytherin     61
12 Draco  Malfoy    Slytherin     92
```

separate()

```
separate(data, col, into, sep = "[^:alnum:]]+", ...,  
         extra = "warn", fill = "warn")
```

What to do with too many pieces:

- **warn** - emit warning, drop extra values
- **drop** - drop extra values without warning
- **merge** - splint at most `length(into)` times

What to do with not enough pieces:

- **warn** - emit warning, fill with NA from right
- **right** - fill with missing values from the right
- **left** - fill with missing values from the left

unite()

Opposite of **separate()** - merges columns together

```
unite(data, col, ..., sep = "_")
```

Data frame to
tidy

name of
new
column

columns to be
united

what divides the
pieces of
information

```

sep_scores <- scores %>%
  separate(name, into = c("first", "last"))
sep_scores %>%
  unite("full_name", first, last)
# A tibble: 12 x 3
  full_name      house      score
  <chr>          <chr>    <dbl>
1 Ronald_Weasley Gryffindor    78
2 Harry_Potter   Gryffindor    85
3 Hermione_Granger Gryffindor  100
4 Justin_Finch   Hufflepuff    87
5 Hannah_Abbot   Hufflepuff    92
6 Susan_Bones    Hufflepuff    93
7 Anthony_Goldstein Ravenclaw    84
8 Michael_Corner  Ravenclaw    93
9 Padma_Patil     Ravenclaw    97
10 Vincent_Crabbe Slytherin     61
11 Gregory_Goyle   Slytherin     61
12 Draco_Malfoy    Slytherin     92

```



```

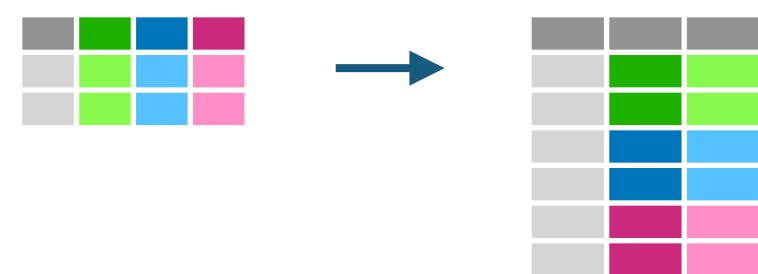
sep_scores <- scores %>%
  separate(name, into = c("first", "last"))
sep_scores %>%
  unite("full_name", first, last, sep = " ")
# A tibble: 12 x 3
  full_name      house      score
  <chr>          <chr>    <dbl>
1 Ronald Weasley Gryffindor    78
2 Harry Potter   Gryffindor    85
3 Hermione Granger Gryffindor  100
4 Justin Finch   Hufflepuff    87
5 Hannah Abbot   Hufflepuff    92
6 Susan Bones    Hufflepuff    93
7 Anthony Goldstein Ravenclaw    84
8 Michael Corner Ravenclaw    93
9 Padma Patil    Ravenclaw    97
10 Vincent Crabbe Slytherin    61
11 Gregory Goyle  Slytherin    61
12 Draco Malfoy   Slytherin    92

```

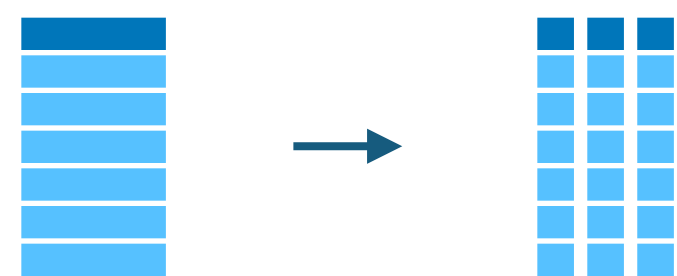
Recap: tidyr verbs



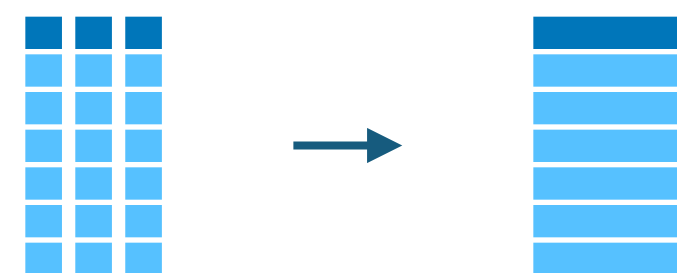
Move values into column names with **`pivot_wider()`**



Move column names into values with **`pivot_longer()`**



Split a column with **`separate()`**



Unite columns with **`unite()`**

Tidy Data

