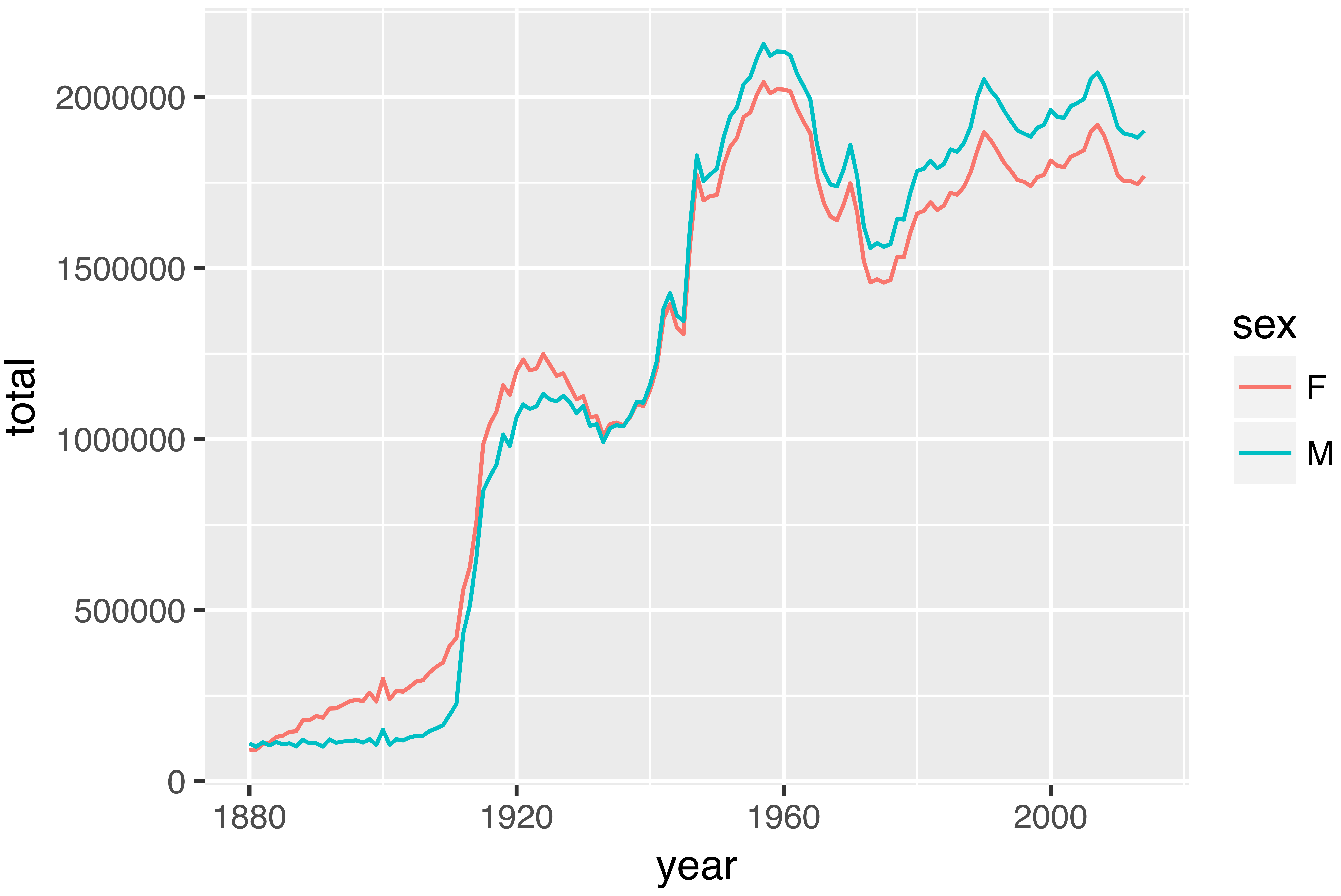
Tidy Data with

tidyr

www.rstudio.com

Number of children by year and gender



Can we calculate the ratio of boys to girls?

```
babynames %>%
  group_by(year, sex) %>%
  summarise(n = sum(n))
```

```
    year   sex      n
   <dbl> <chr>  <int>
1   1880     F  90993
2   1880     M 110491
3   1881     F  91954
4   1881     M 100745
5   1882     F 107850
6   1882     M 113688
```

Can we calculate the ratio of boys to girls?

```
babynames %>%
  group_by(year, sex) %>%
  summarise(n = sum(n))
```

```
   year   sex      n
  <dbl> <chr>  <int>
1  1880     F  90993
2  1880     M 110491
3  1881     F  91954
4  1881     M 100745
5  1882     F 107850
6  1882     M 113688
```

Now what?

Can we calculate the ratio of boys to girls?

```
better_layout %>%
  mutate(percent_male = M / (M + F) * 100)
```

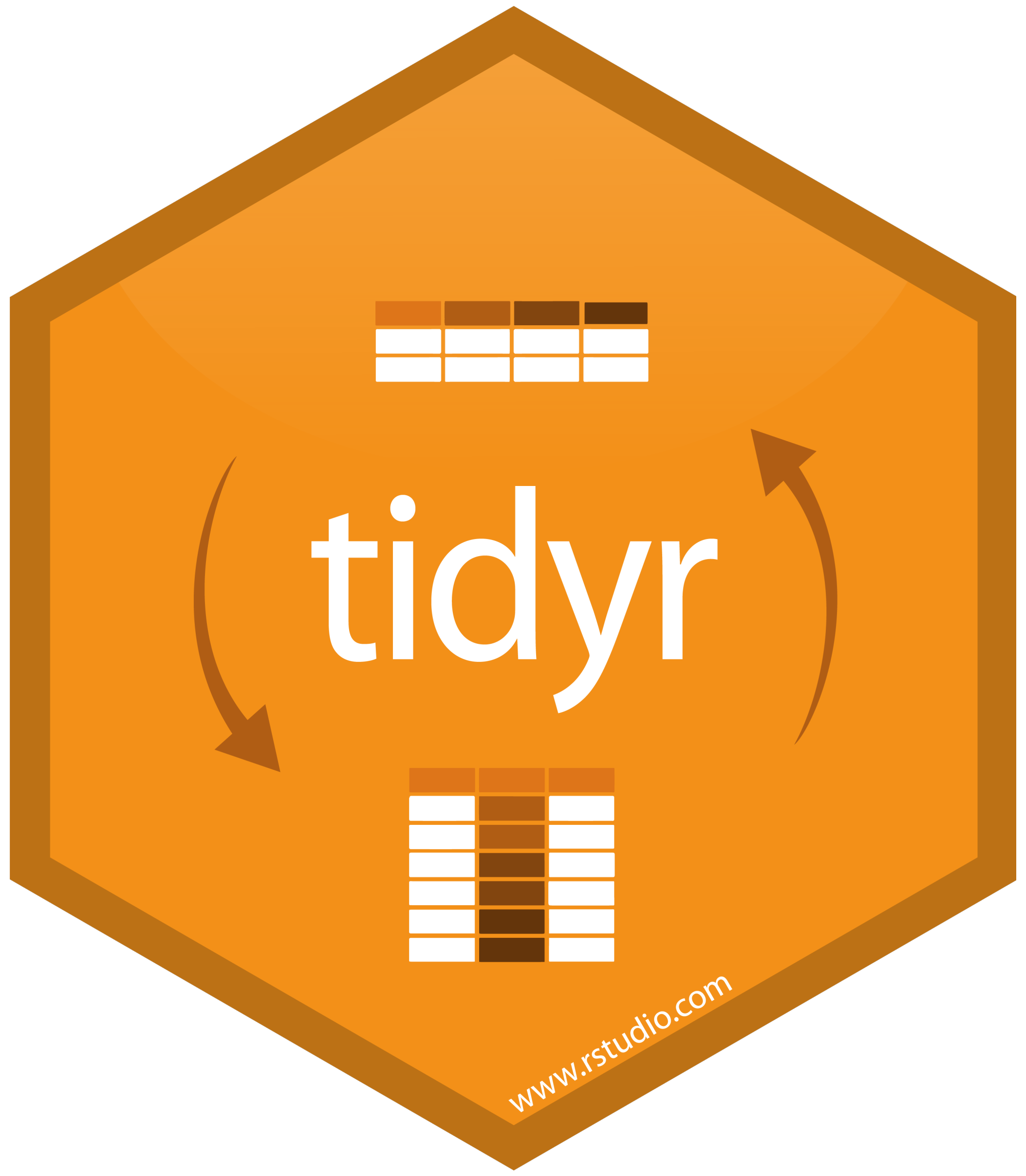
```
    year   sex       n
   <dbl> <chr>   <int>
1   1880     F   90993
2   1880     M  110491
3   1881     F   91954
4   1881     M  100745
5   1882     F  107850
6   1882     M  113688
```

?

```
      year      F      M
*    <dbl>  <int>  <int>
1     1880  90993 110491
2     1881  91954 100745
3     1882 107850 113688
4     1883 112321 104629
5     1884 129022 114445
6     1885 133055 107800
```

tidyr

tidyr



A package that reshapes the layout of tabular data.

spread()

Toy data

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

pollution

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

Your Turn

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

05:00

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

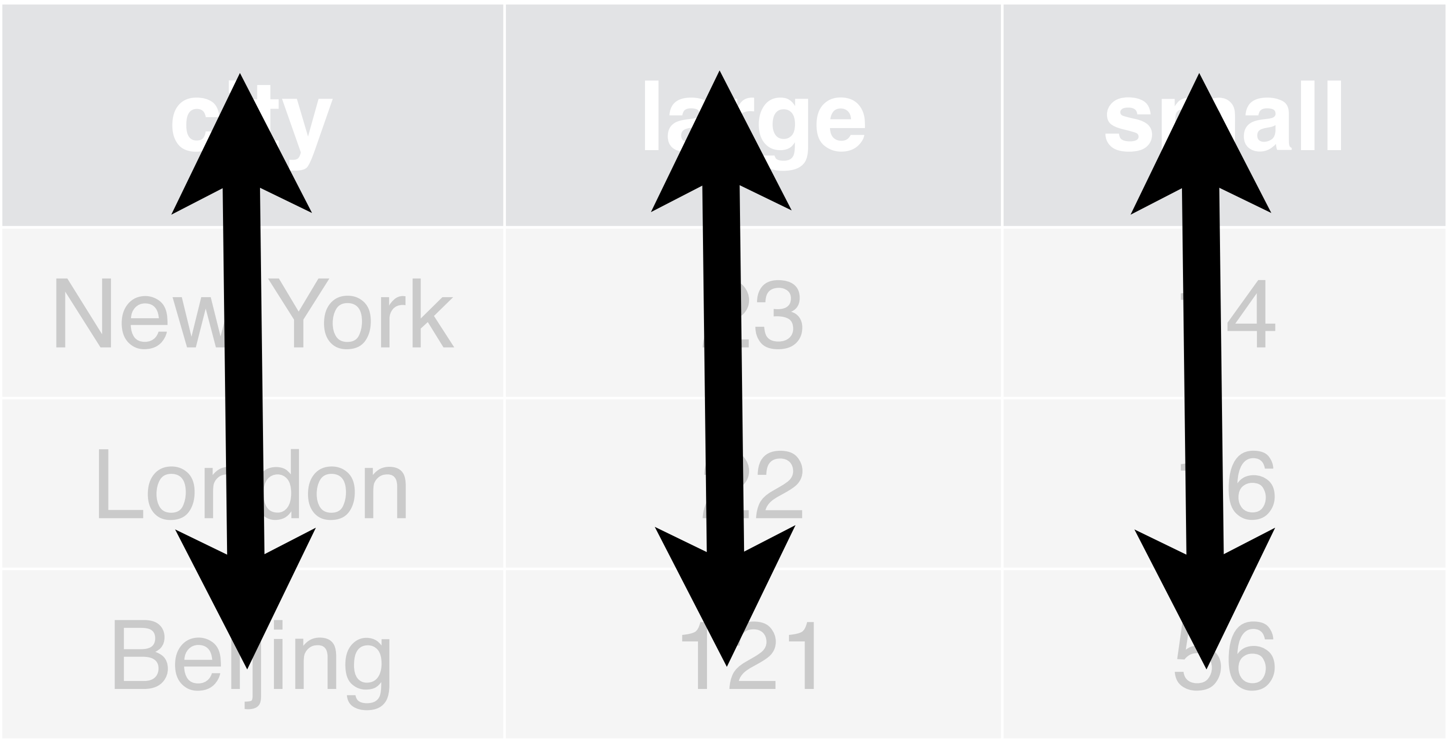
city	size	amount
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

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

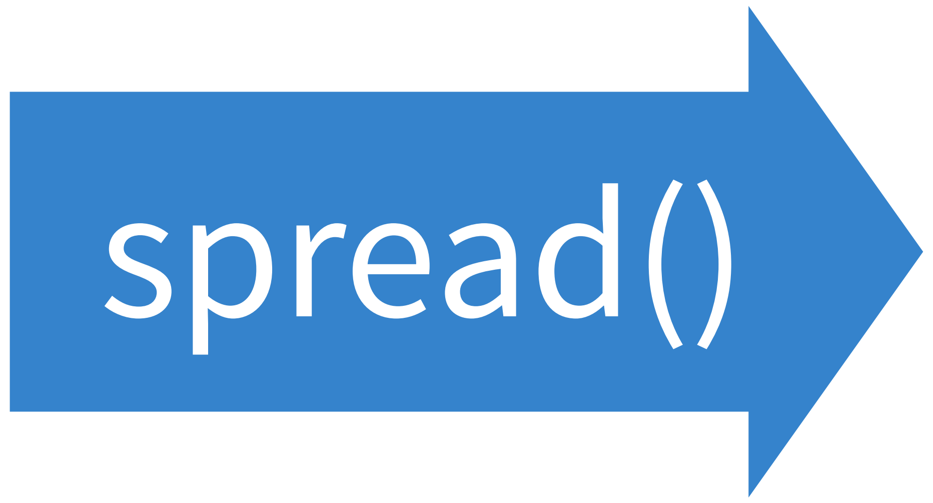
city	large	small
New York	23	14
London	22	16
Beijing	121	56

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



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

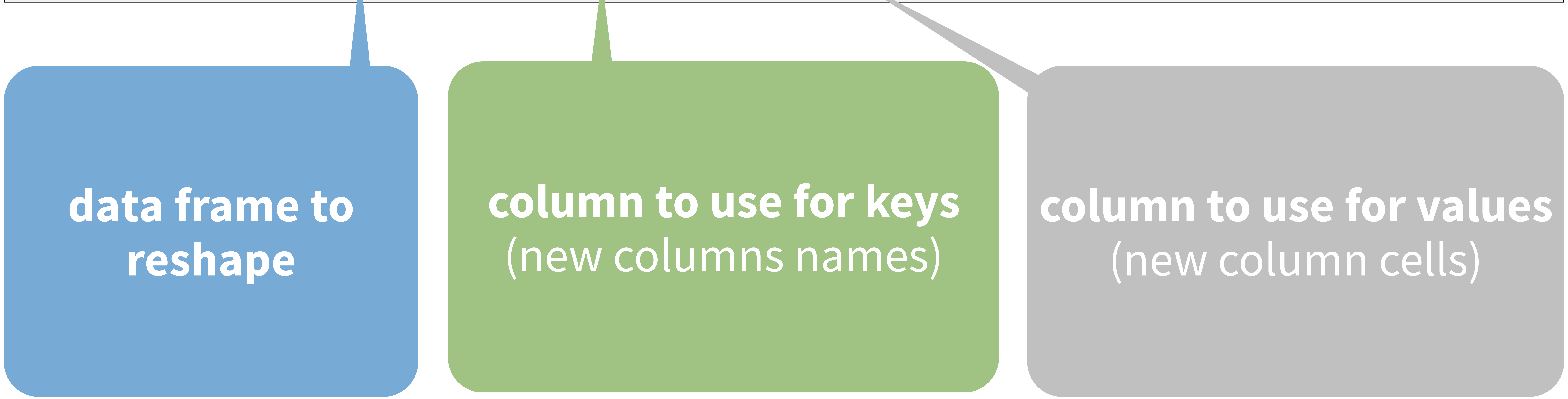
city	large	small
New York	23	14
London	22	16
Beijing	121	56

spread()

Generates multiple columns from two columns:

- unique values in the **key** column become **column names**
- values in the **value** column become **cells** in the new columns

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



data frame to reshape

column to use for keys
(new columns names)

column to use for values
(new column cells)

```
spread(pollution, 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
```

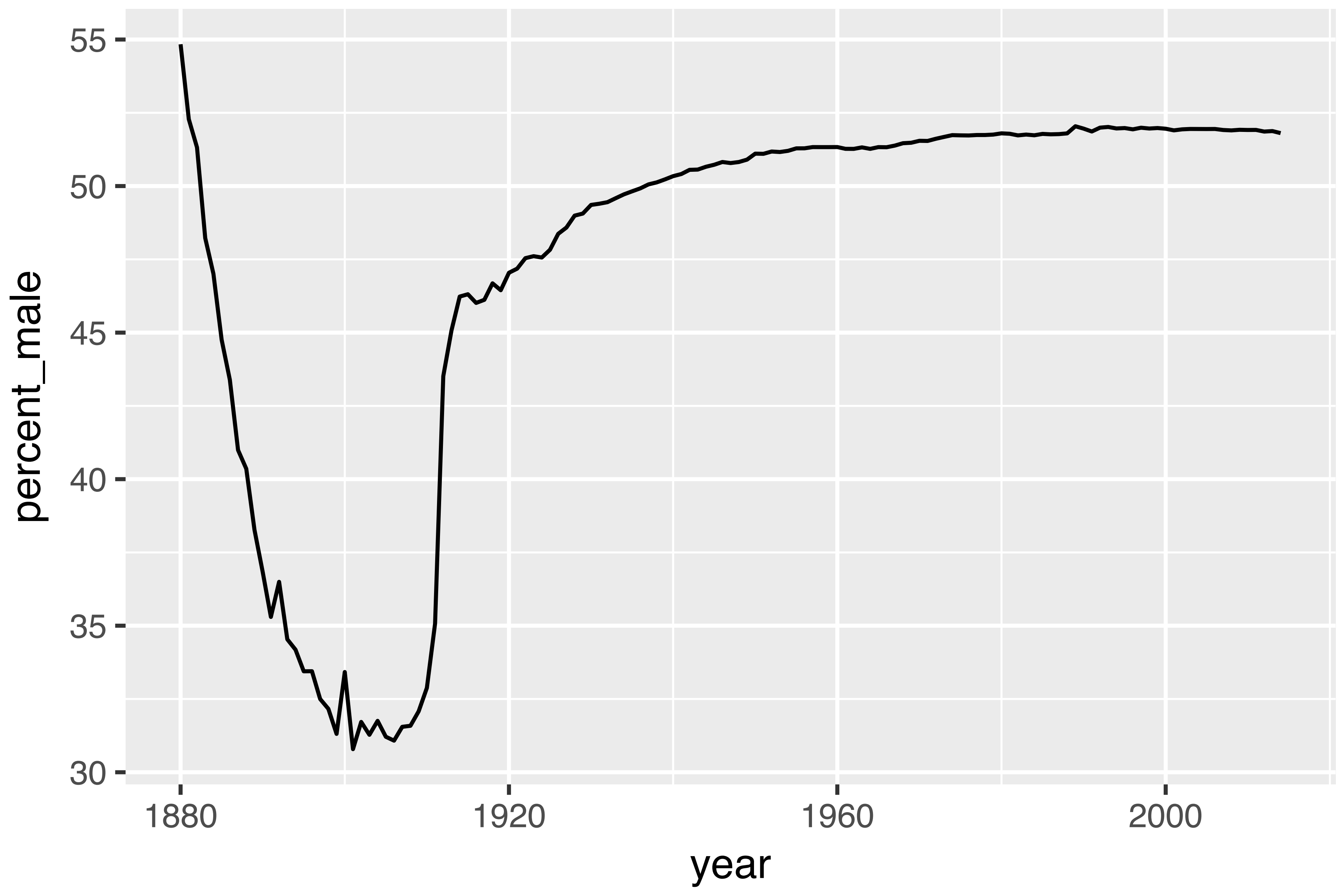
Your Turn

Reshape the layout of this data. Calculate the percent of male children by year. And then plot the percent over time.

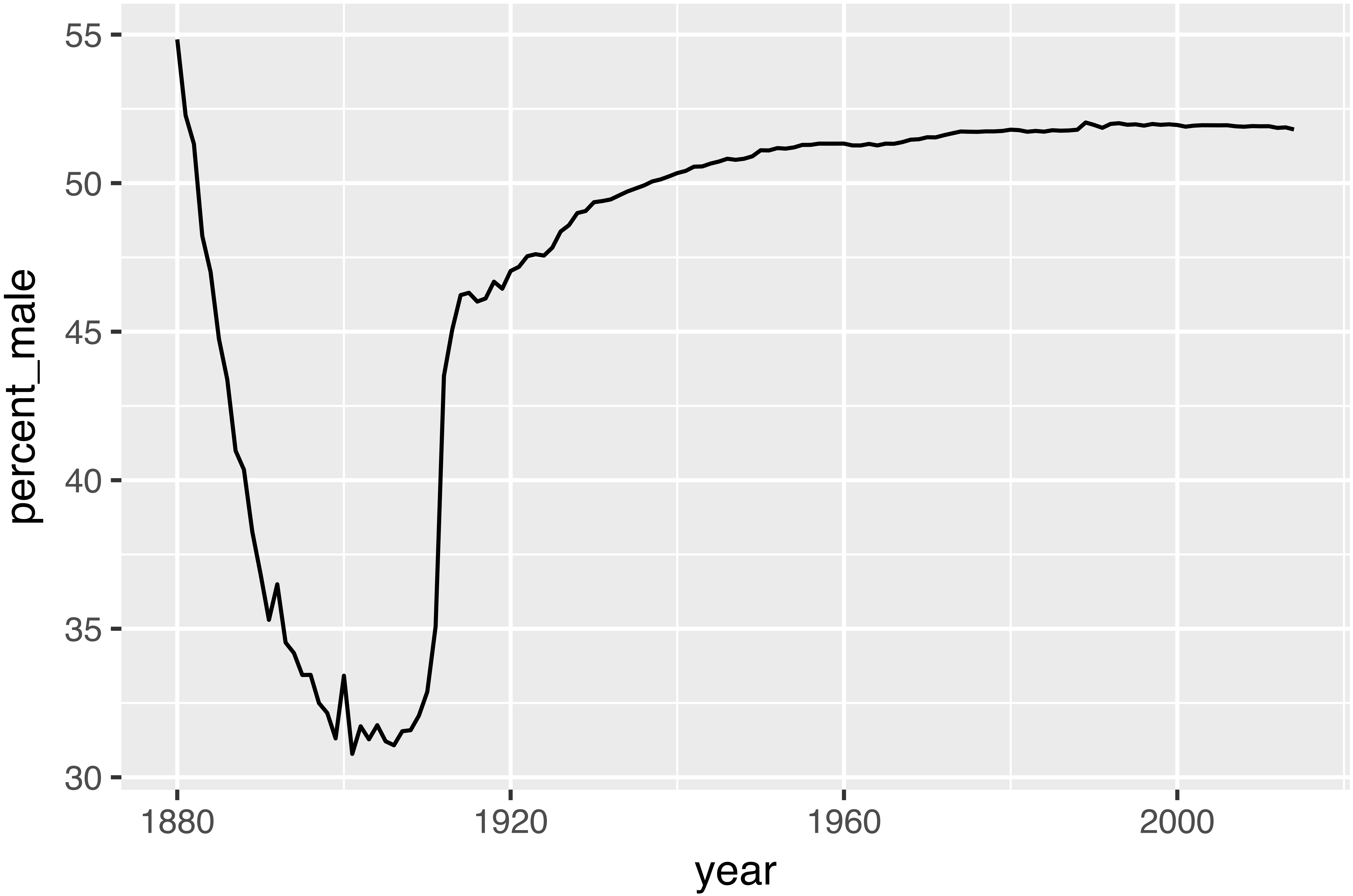
```
babynames %>%
  group_by(year, sex) %>%
  summarise(n = sum(n))
```

05:00

```
babynames %>%
  group_by(year, sex) %>%
  summarise(n = sum(n)) %>%
  spread(sex, n) %>%
  mutate(percent_male = M / (M + F) * 100) %>%
  ggplot(aes(year, percent_male)) + geom_line()
```



Percent of children that are male by year



Reshaping tables

spread() is one of a family of functions for reshaping tables.

- **spread()** - move values into column names
- **gather()** - move column names into values
- **separate()** - separate variables that share a column
- **unite()** - unite a variable that is split across several columns

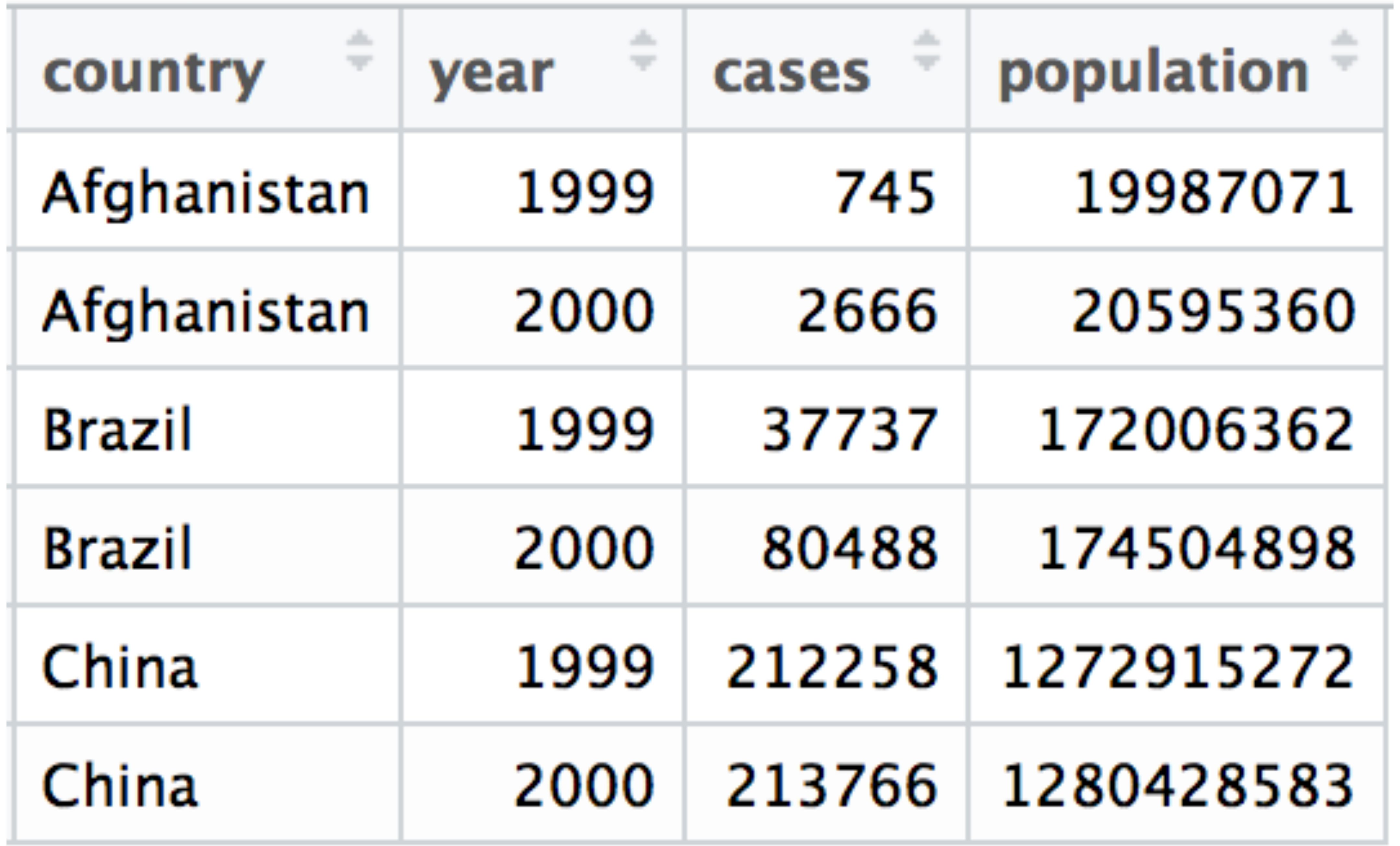
Tidy Data

Tidy data

Tidy functions all expect and return the same data structure, known as **tidy data**:

1. A **data frame** that contains
2. **variables** in the **columns** and
3. **cases** in the **rows**.

```
View(table1)
```

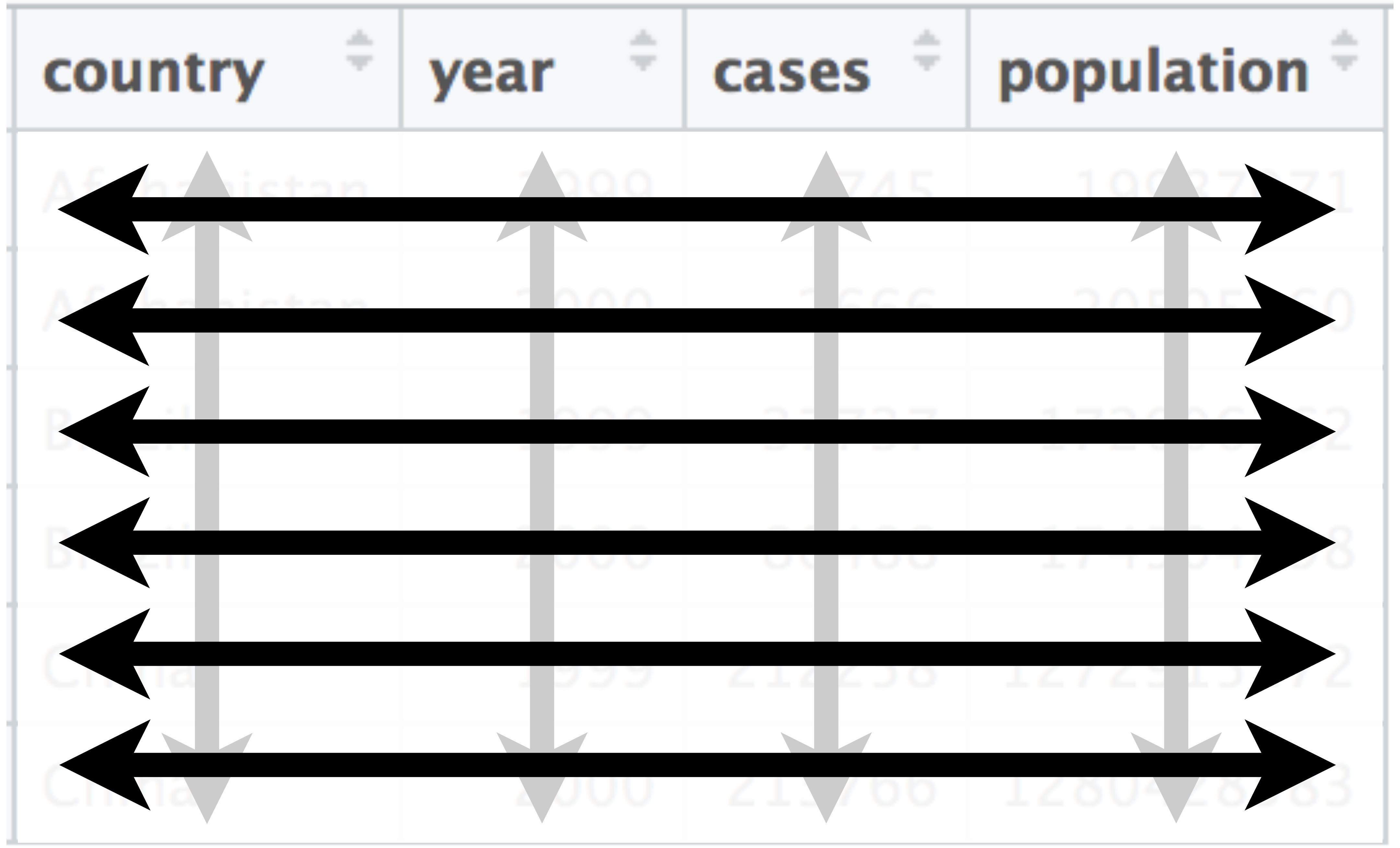


country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

columns

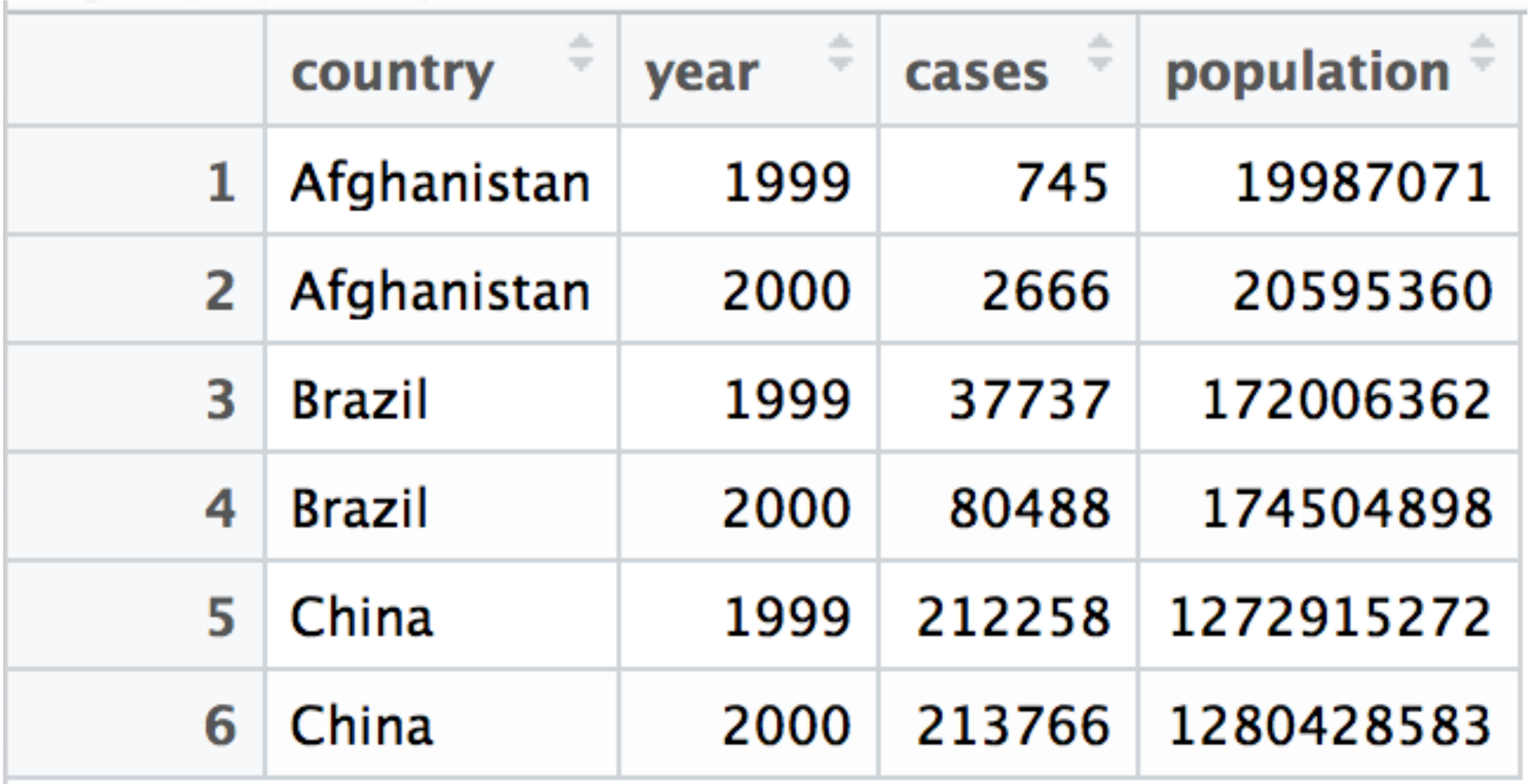
rows

Tidy data



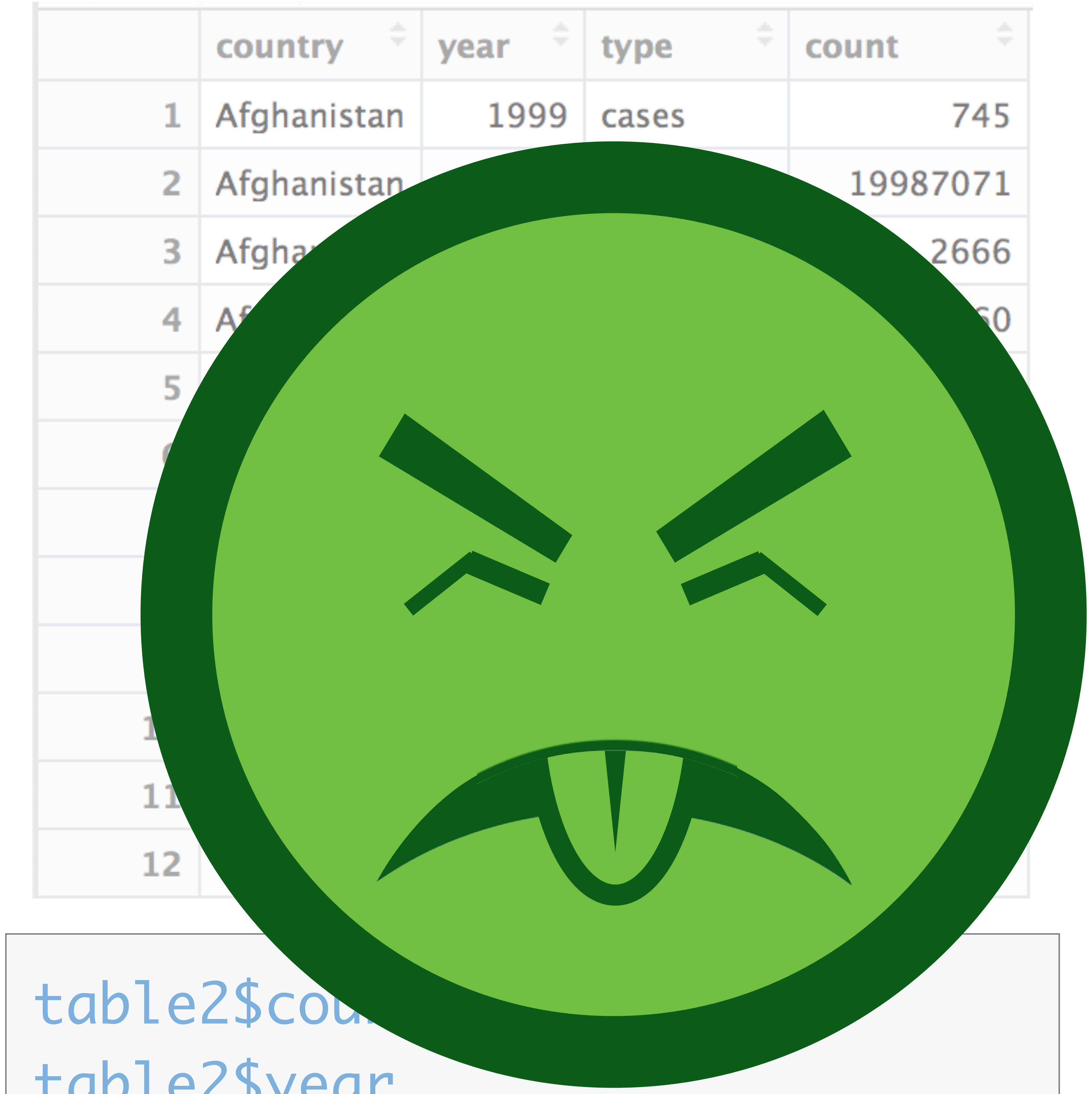
A data set is **tidy** iff:

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



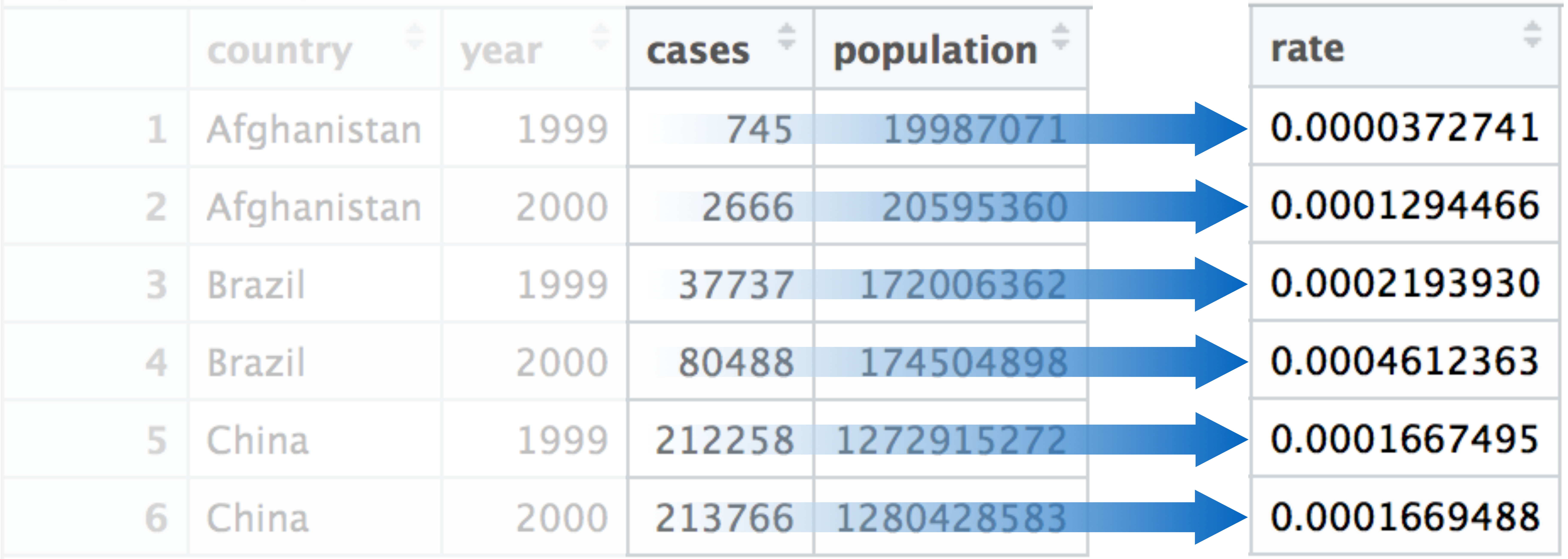
	country	year	cases	population
1	Afghanistan	1999	745	19987071
2	Afghanistan	2000	2666	20595360
3	Brazil	1999	37737	172006362
4	Brazil	2000	80488	174504898
5	China	1999	212258	1272915272
6	China	2000	213766	1280428583

```
table1$country
table1$year
table1$cases
table1$population
```



	country	year	type	count
1	Afghanistan	1999	cases	745
2	Afghanistan			19987071
3	Afgha			2666
4	Af			60
5				
1				
11				
12				

```
table2$cou
table2$year
table2$count[c(1,3,5,7,9,11)]
table2$count[c(2,4,6,8,10,12)]
```



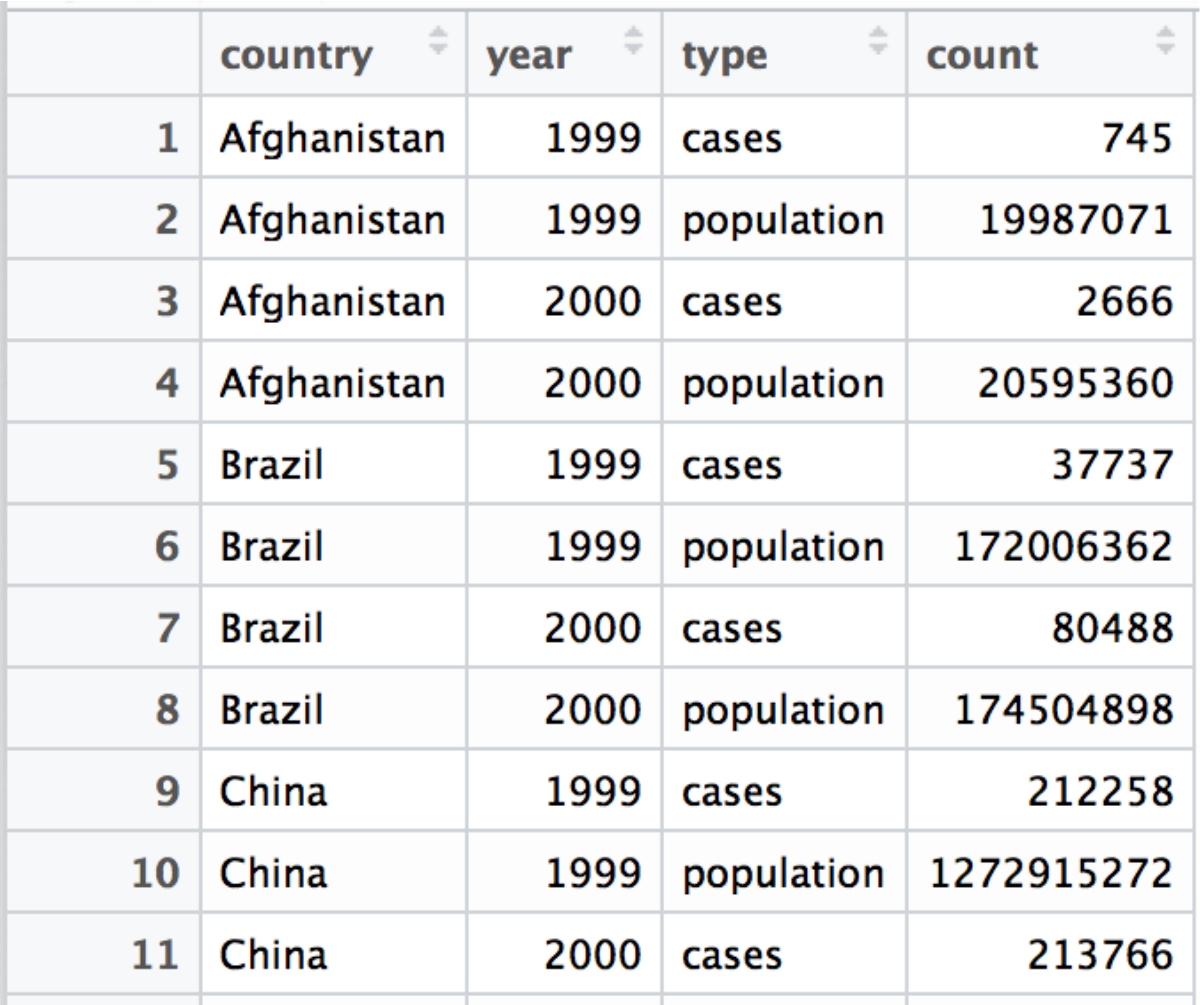
	country	year	cases	population	rate
1	Afghanistan	1999	745	19987071	0.0000372741
2	Afghanistan	2000	2666	20595360	0.0001294466
3	Brazil	1999	37737	172006362	0.0002193930
4	Brazil	2000	80488	174504898	0.0004612363
5	China	1999	212258	1272915272	0.0001667495
6	China	2000	213766	1280428583	0.0001669488

```
table1$cases / table1$population -> table1$rate
```

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

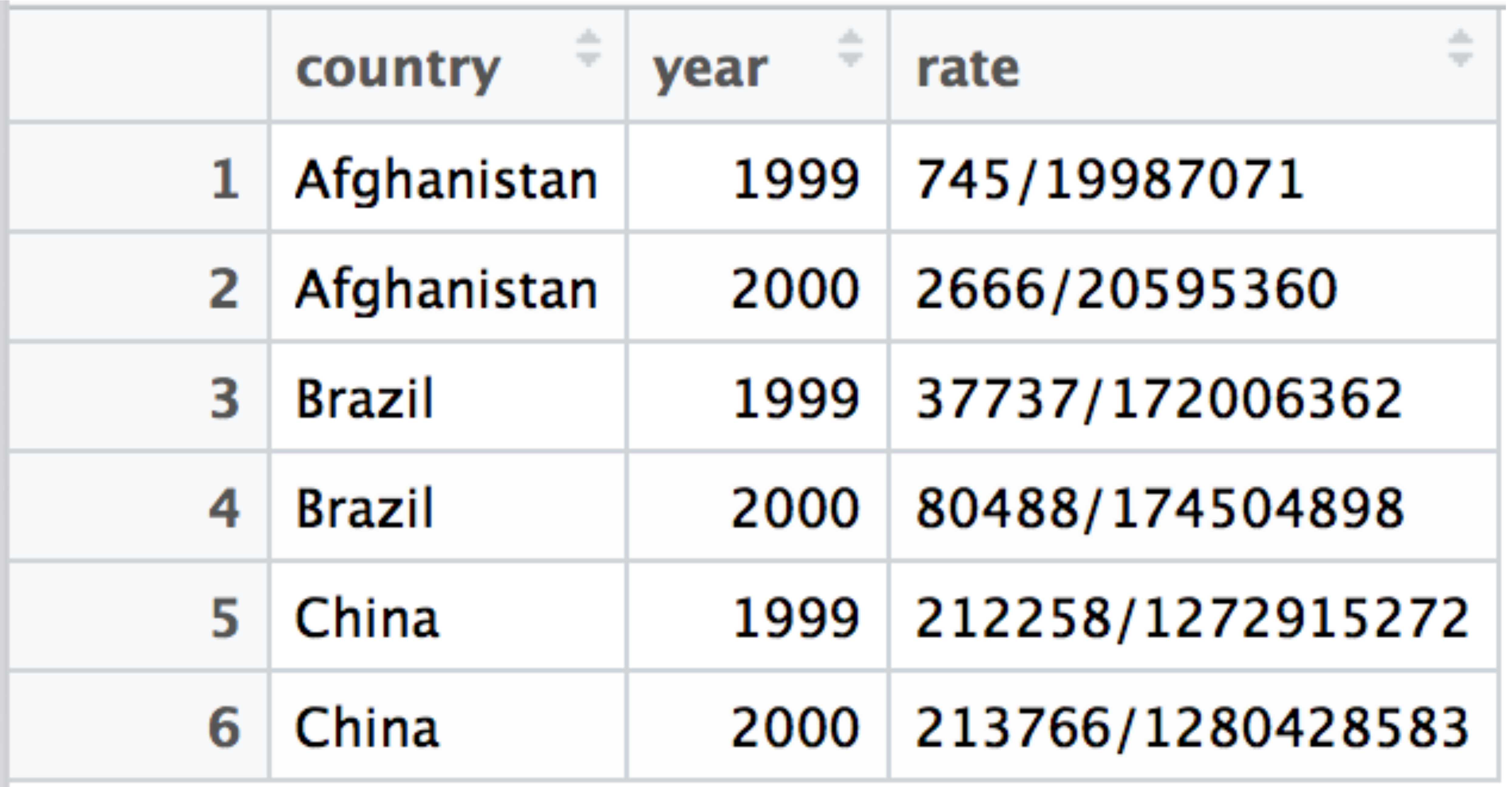
- Garrett Grolemund

table2



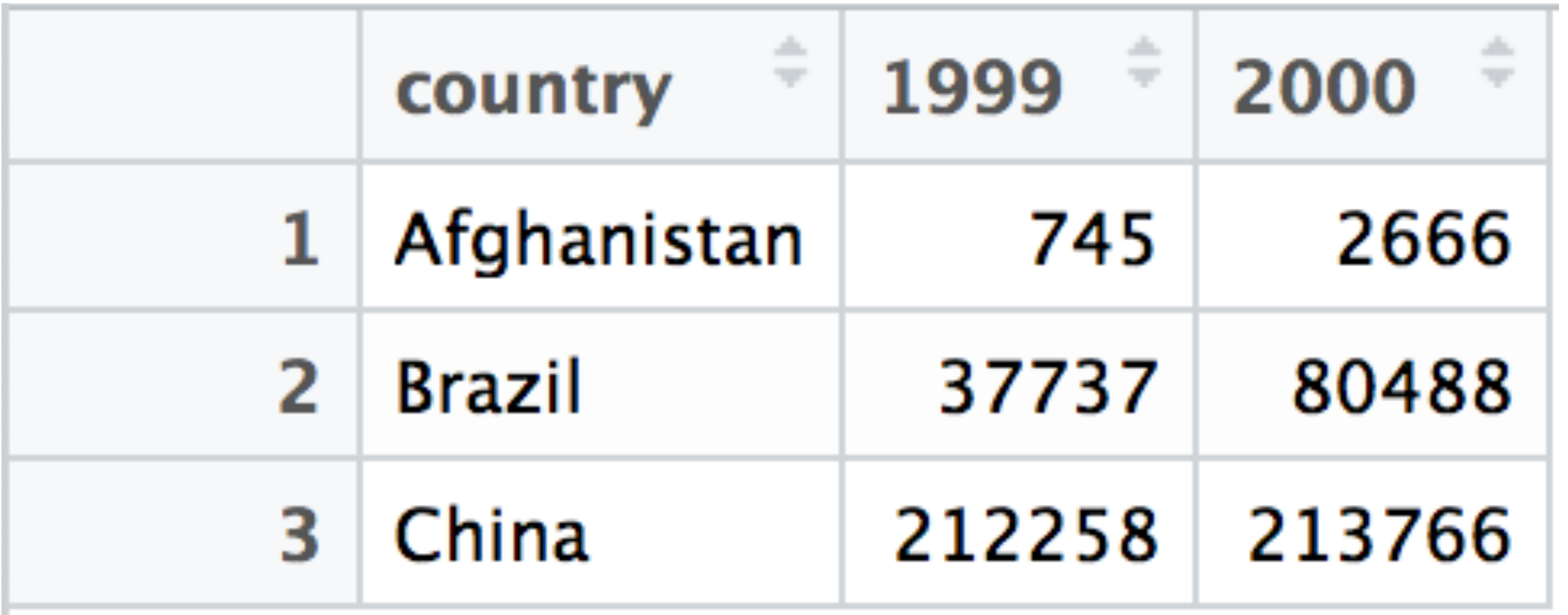
	country	year	type	count
1	Afghanistan	1999	cases	745
2	Afghanistan	1999	population	19987071
3	Afghanistan	2000	cases	2666
4	Afghanistan	2000	population	20595360
5	Brazil	1999	cases	37737
6	Brazil	1999	population	172006362
7	Brazil	2000	cases	80488
8	Brazil	2000	population	174504898
9	China	1999	cases	212258
10	China	1999	population	1272915272
11	China	2000	cases	213766

table3

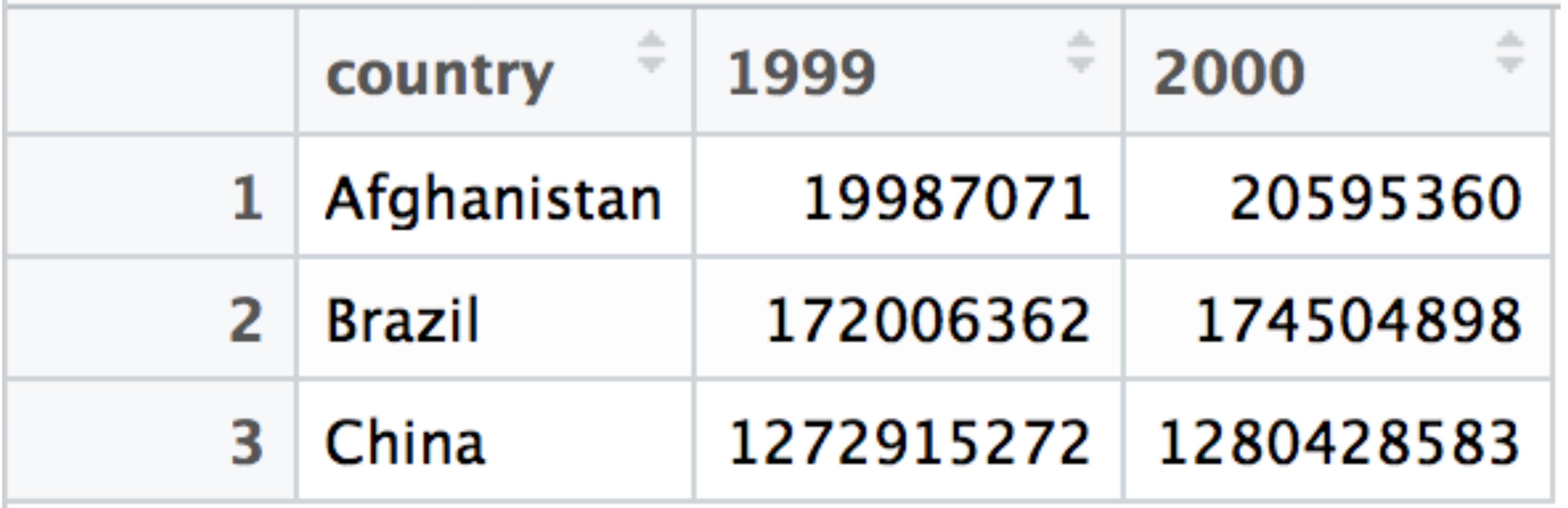


	country	year	rate
1	Afghanistan	1999	745/19987071
2	Afghanistan	2000	2666/20595360
3	Brazil	1999	37737/172006362
4	Brazil	2000	80488/174504898
5	China	1999	212258/1272915272
6	China	2000	213766/1280428583

table4a and table4b

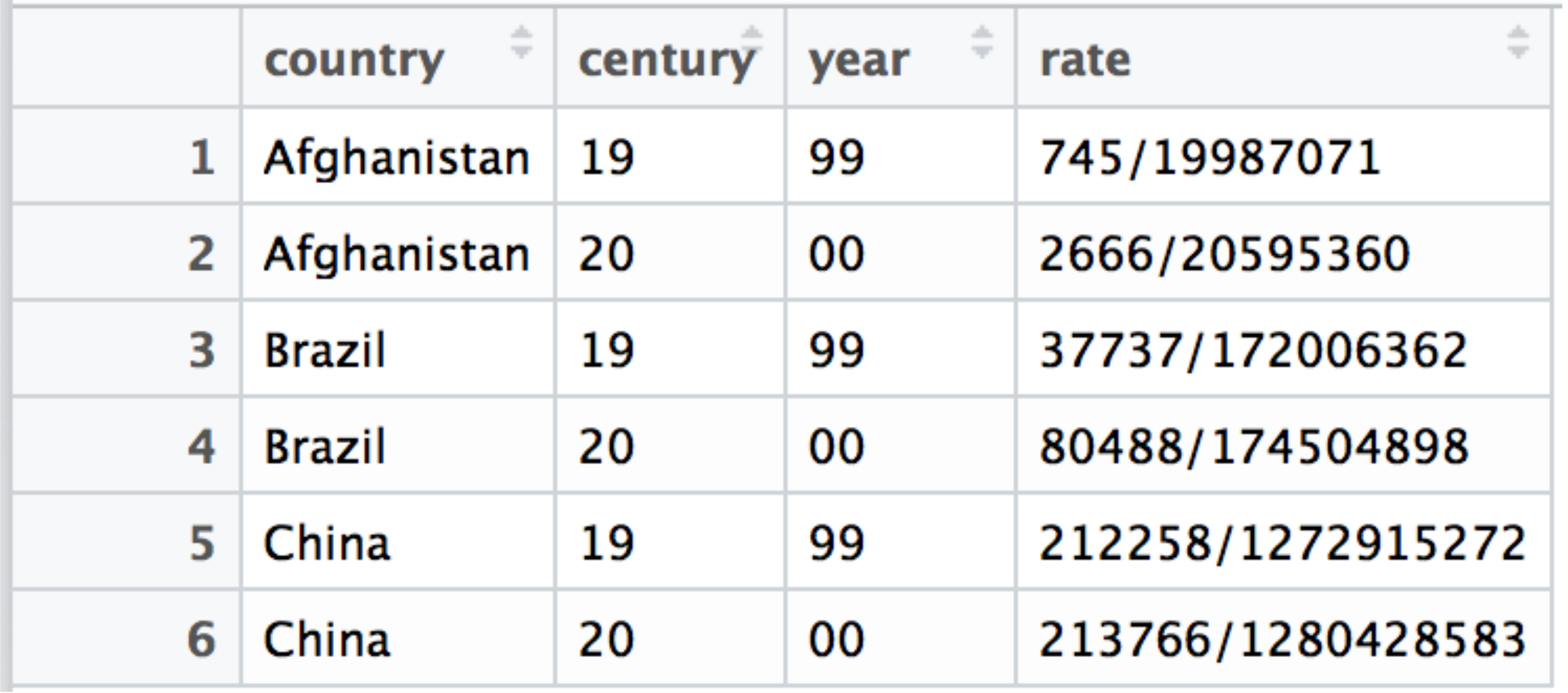


	country	1999	2000
1	Afghanistan	745	2666
2	Brazil	37737	80488
3	China	212258	213766



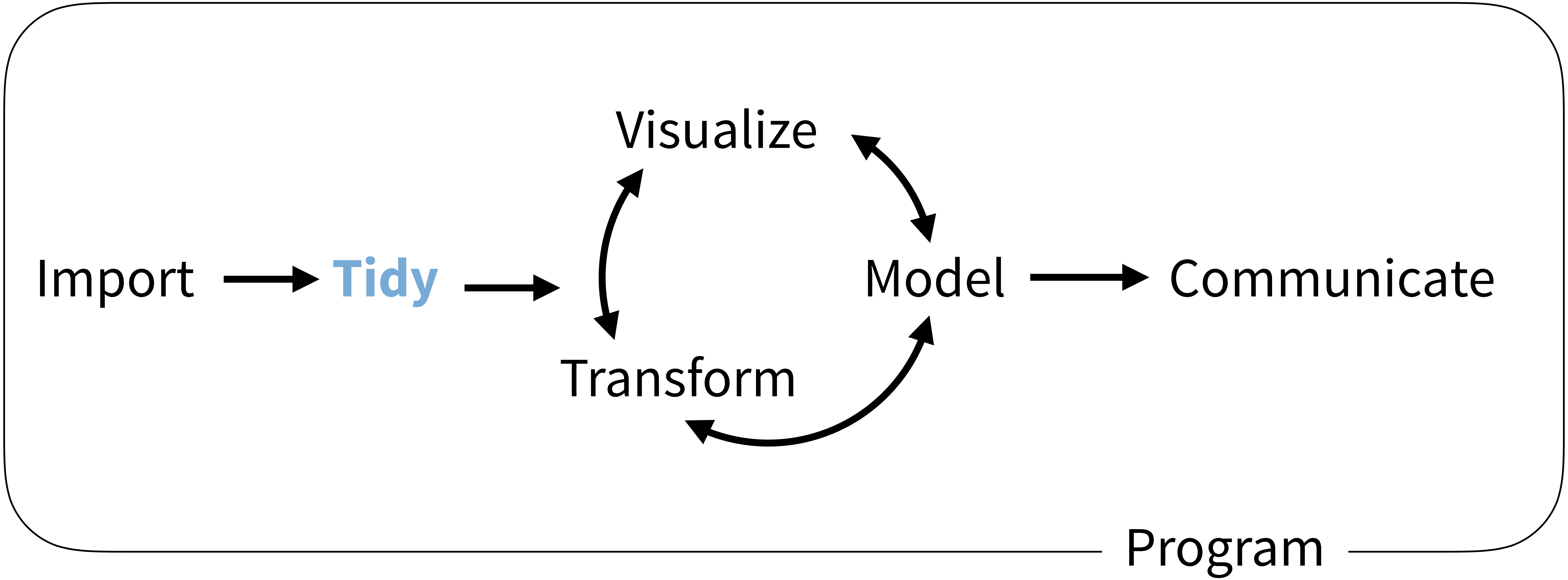
	country	1999	2000
1	Afghanistan	19987071	20595360
2	Brazil	172006362	174504898
3	China	1272915272	1280428583

table5



	country	century	year	rate
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

(Applied) Data Science



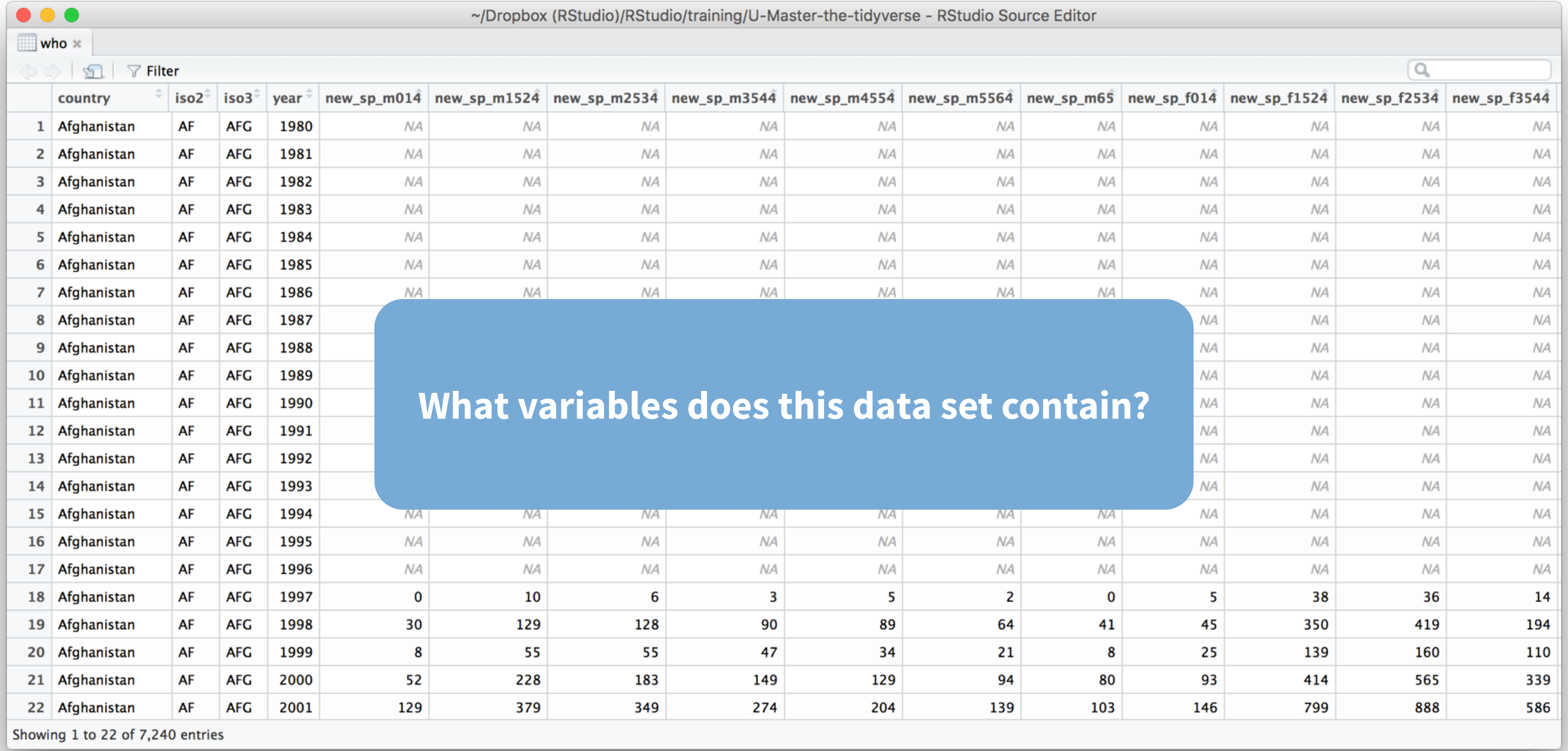
who

(Untidy Data)

who

Tuberculosis (TB) cases broken down by year, country, age, gender, and diagnosis method from the *2014 World Health Organization Global Tuberculosis Report*

```
View(who)
```

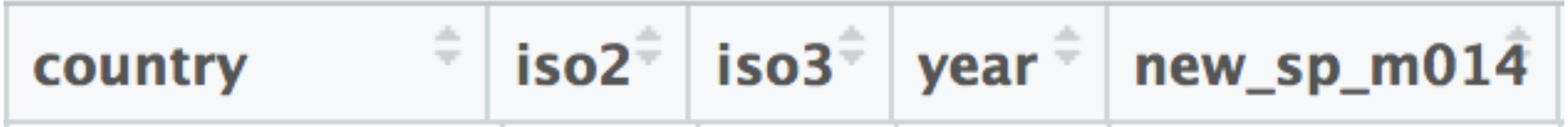

~/Dropbox (RStudio)/RStudio/training/U-Master-the-tidyverse - RStudio Source Editor
who
Filter

	country	iso2	iso3	year	new_sp_m014	new_sp_m1524	new_sp_m2534	new_sp_m3544	new_sp_m4554	new_sp_m5564	new_sp_m65	new_sp_f014	new_sp_f1524	new_sp_f2534	new_sp_f3544
1	Afghanistan	AF	AFG	1980	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2	Afghanistan	AF	AFG	1981	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
3	Afghanistan	AF	AFG	1982	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
4	Afghanistan	AF	AFG	1983	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
5	Afghanistan	AF	AFG	1984	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
6	Afghanistan	AF	AFG	1985	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
7	Afghanistan	AF	AFG	1986	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
8	Afghanistan	AF	AFG	1987								NA	NA	NA	NA
9	Afghanistan	AF	AFG	1988								NA	NA	NA	NA
10	Afghanistan	AF	AFG	1989								NA	NA	NA	NA
11	Afghanistan	AF	AFG	1990								NA	NA	NA	NA
12	Afghanistan	AF	AFG	1991								NA	NA	NA	NA
13	Afghanistan	AF	AFG	1992								NA	NA	NA	NA
14	Afghanistan	AF	AFG	1993								NA	NA	NA	NA
15	Afghanistan	AF	AFG	1994	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
16	Afghanistan	AF	AFG	1995	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
17	Afghanistan	AF	AFG	1996	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
18	Afghanistan	AF	AFG	1997	0	10	6	3	5	2	0	5	38	36	14
19	Afghanistan	AF	AFG	1998	30	129	128	90	89	64	41	45	350	419	194
20	Afghanistan	AF	AFG	1999	8	55	55	47	34	21	8	25	139	160	110
21	Afghanistan	AF	AFG	2000	52	228	183	149	129	94	80	93	414	565	339
22	Afghanistan	AF	AFG	2001	129	379	349	274	204	139	103	146	799	888	586

Showing 1 to 22 of 7,240 entries

What variables does this data set contain?

who variables



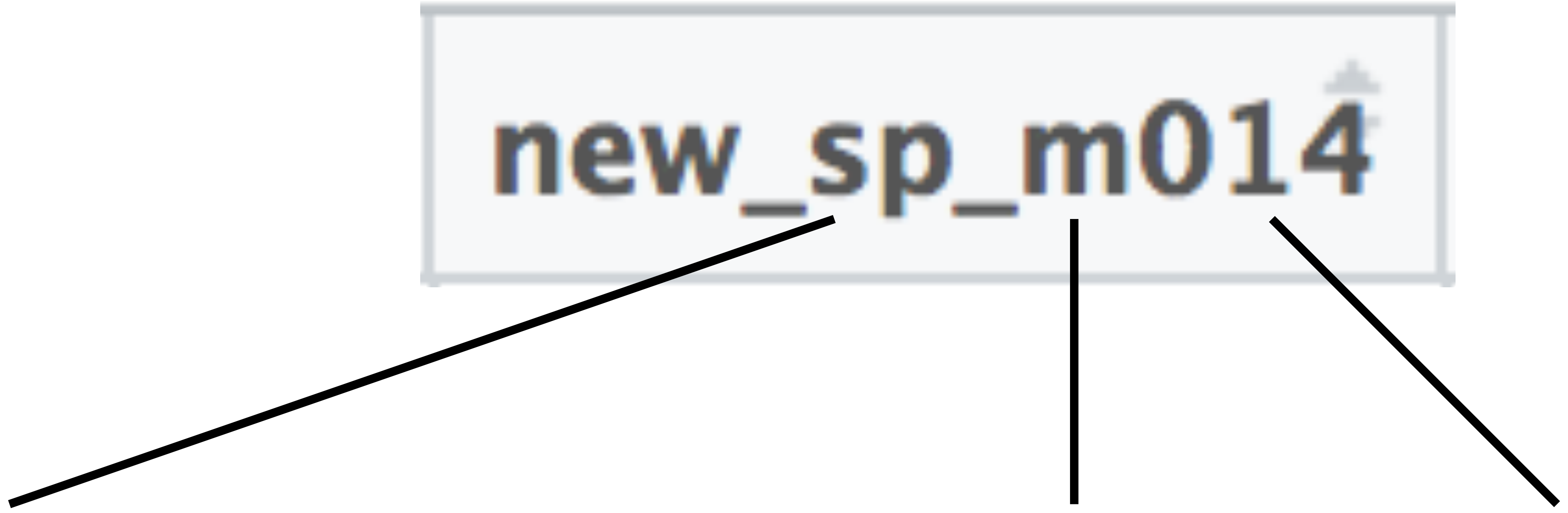
country	iso2	iso3	year	new_sp_m014

country, **iso2**, **iso3** - country identifiers

year - year

other columns names - encode **type** of TB case, **sex**, and **age**

who codes



new_sp_m014

Type of TB case

- **rel** - relapse
- **ep** - extra-pulmonary
- **sn**- pulmonary, smear negative
- **sp** -pulmonary, smear positive

Gender

- **m** - male
- **f** - female

Age group

- **014** - 0 to 14 years old
- **1524** - 15 to 24 years old
- **2534** - 25 to 34 years old
- **3544** - 35 to 44 years old
- **4554** - 45 to 54 years old
- **5564** - 55 to 64 years old
- **65** - 65 and older

~/Dropbox (RStudio)/RStudio/training/U-Master-the-tidyverse - RStudio Source Editor
who
Filter

	country	iso2	iso3	year	new_sp_m014	new_sp_m1524	new_sp_m2534	new_sp_m3544	new_sp_m4554	new_sp_m5564	new_sp_m65	new_sp_f014	new_sp_f1524	new_sp_f2534	new_sp_f3544
1	Afghanistan	AF	AFG	1980	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2	Afghanistan	AF	AFG	1981	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
3	Afghanistan	AF	AFG	1982	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
4	Afghanistan	AF	AFG	1983	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
5	Afghanistan	AF	AFG	1984	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
6	Afghanistan	AF	AFG	1985	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
7	Afghanistan	AF	AFG	1986	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
8	Afghanistan	AF	AFG	1987	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
9	Afghanistan	AF	AFG	1988	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
10	Afghanistan	AF	AFG	1989	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
11	Afghanistan	AF	AFG	1990	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
12	Afghanistan	AF	AFG	1991	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
13	Afghanistan	AF	AFG	1992	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
14	Afghanistan	AF	AFG	1993	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
15	Afghanistan	AF	AFG	1994	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
16	Afghanistan	AF	AFG	1995	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
17	Afghanistan	AF	AFG	1996	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
18	Afghanistan	AF	AFG	1997	0	10	6	3	5	2	0	5	38	36	14
19	Afghanistan	AF	AFG	1998	30	129	128	90	89	64	41	45	350	419	194
20	Afghanistan	AF	AFG	1999	8	55	55	47	34	21	8	25	139	160	110
21	Afghanistan	AF	AFG	2000	52	228	183	149	129	94	80	93	414	565	339
22	Afghanistan	AF	AFG	2001	129	379	349	274	204	139	103	146	799	888	586

Showing 1 to 22 of 7,240 entries

gather()

Toy data

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

cases

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

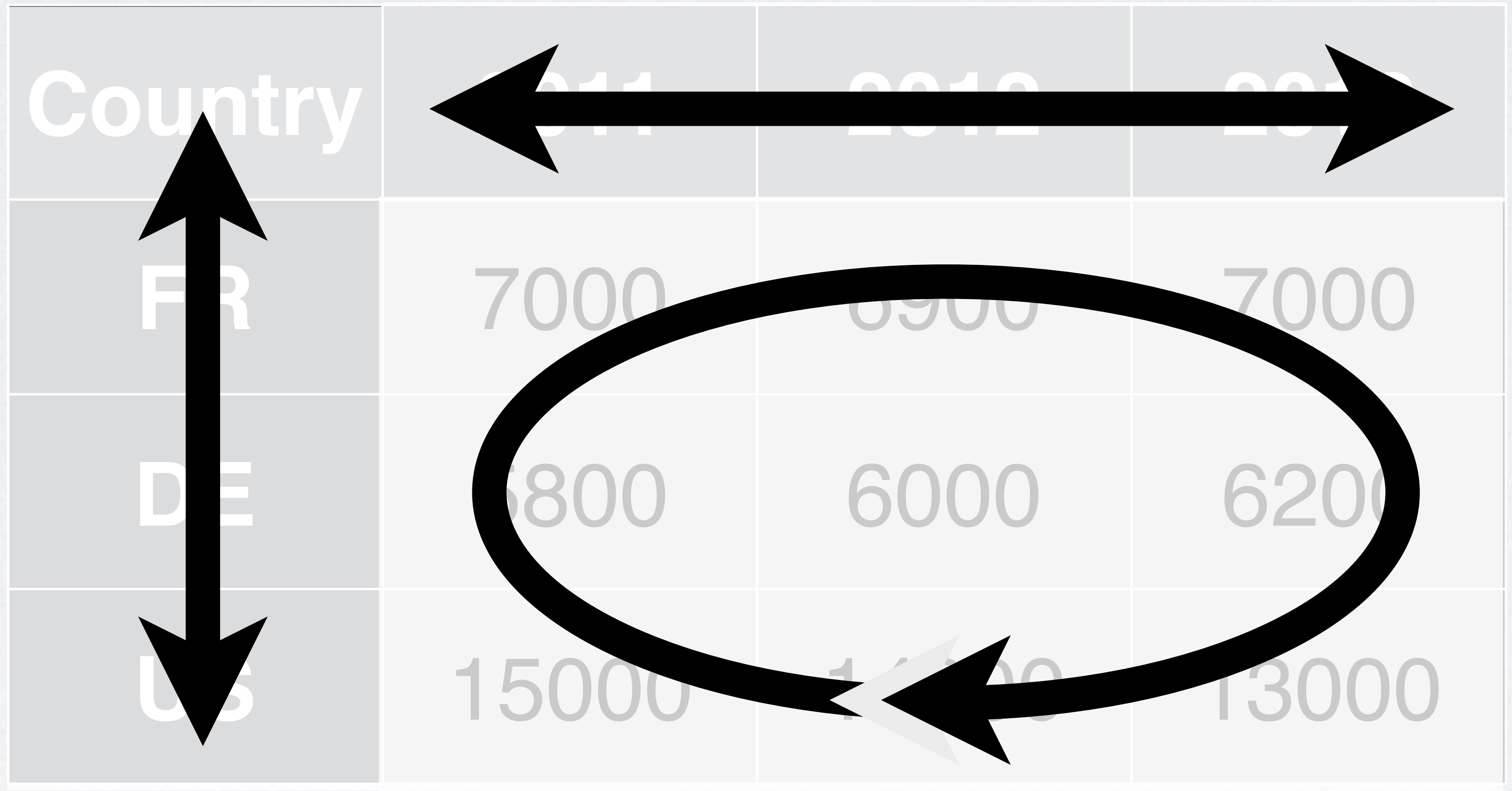
Quiz

What are the variables in this data set?

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

Quiz

What are the variables in this data set?



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

- Country
- Year
- Count

Your Turn

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

05:00

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

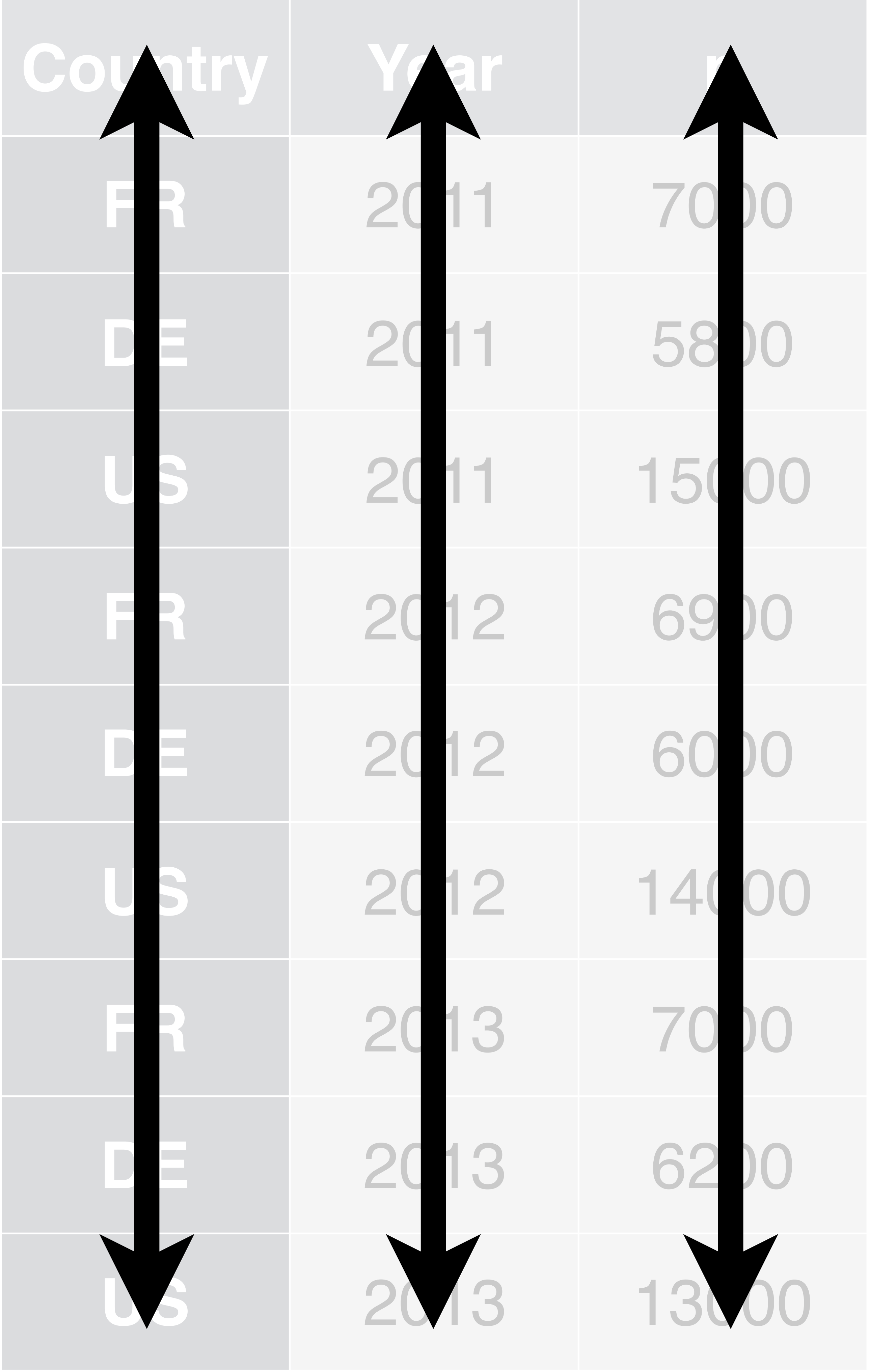
Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

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	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

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

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

key (former column names)

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

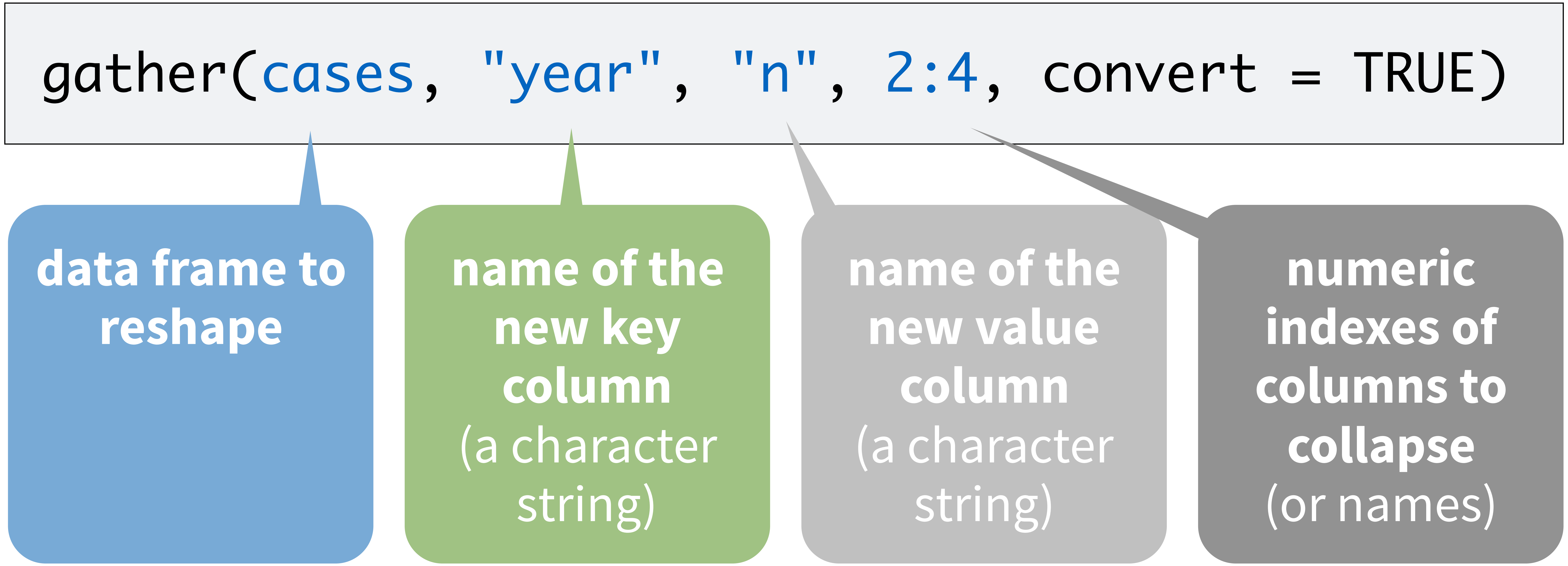
key value (former cells)

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()

Collapses multiple columns into two columns:

- a **key** column that contains the former **column names**
- a **value** column that contains the former **column cells**



```
gather(cases, "year", "n", 2:4, convert = TRUE)
```

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)

```r
gather(cases, "year", "n", 2:4, convert = TRUE)
```

```
  country  2011  2012  2013
1      FR  7000  6900  7000
2      DE  5800  6000  6200
3      US 15000 14000 13000
```

```
  country year     n
1      FR 2011  7000
2      DE 2011  5800
3      US 2011 15000
4      FR 2012  6900
5      DE 2012  6000
6      US 2012 14000
7      FR 2013  7000
8      DE 2013  6200
9      US 2013 13000
```

```
gather(cases, "year", "n", 2:4, convert = TRUE)
```

```
  country  2011  2012  2013
1      FR  7000  6900  7000
2      DE  5800  6000  6200
3      US 15000 14000 13000
```

```
  country year     n
1      FR 2011  7000
2      DE 2011  5800
3      US 2011 15000
4      FR 2012  6900
5      DE 2012  6000
6      US 2012 14000
7      FR 2013  7000
8      DE 2013  6200
9      US 2013 13000
```

Converts numeric column names to numbers

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

```
  country  2011  2012  2013
1      FR  7000  6900  7000
2      DE  5800  6000  6200
3      US 15000 14000 13000
```

Converts numeric column names to numbers

```
  country year      n
1      FR "2011"  7000
2      DE "2011"  5800
3      US "2011" 15000
4      FR "2012"  6900
5      DE "2012"  6000
6      US "2012" 14000
7      FR "2013"  7000
8      DE "2013"  6200
9      US "2013" 13000
```

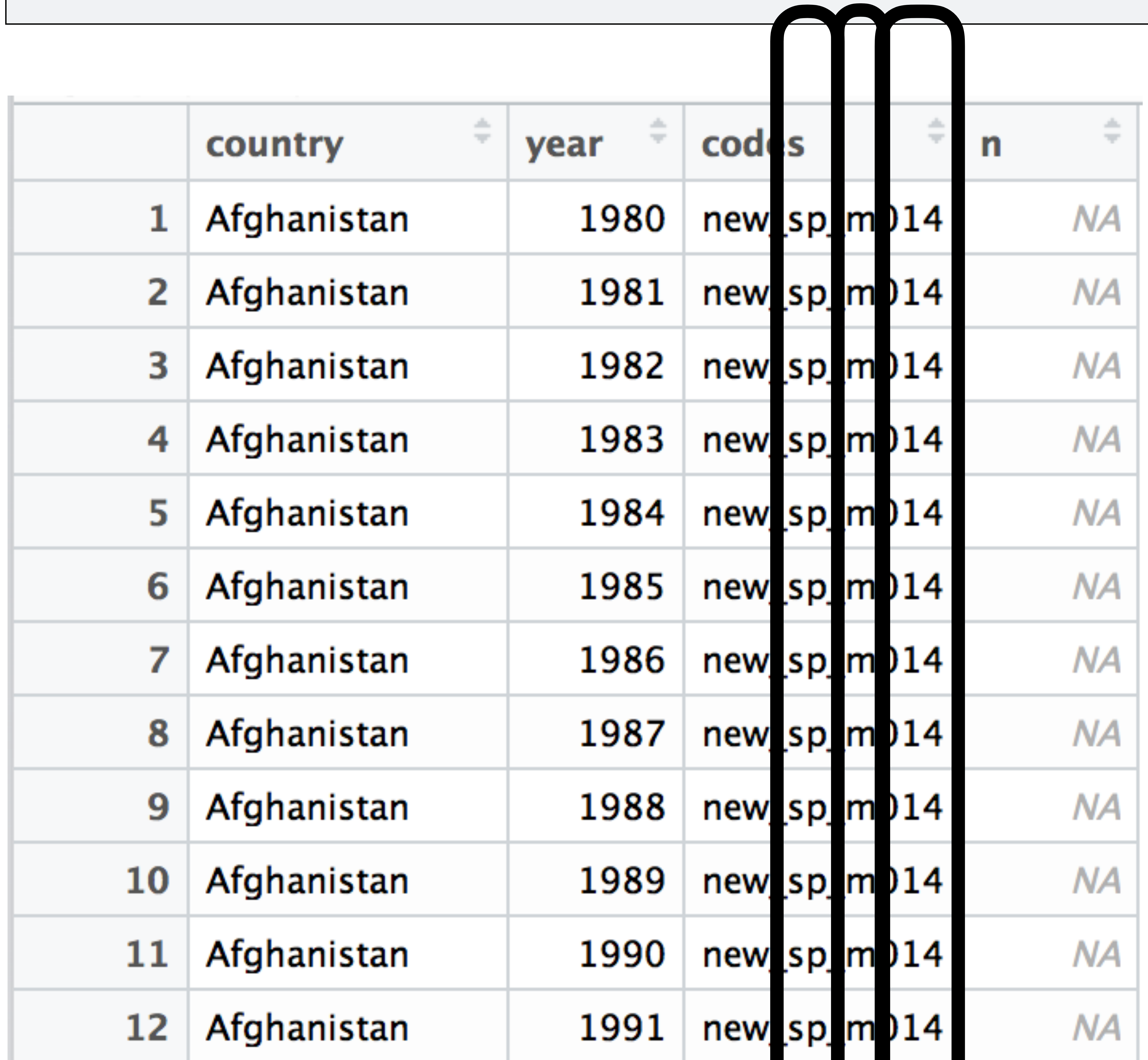
Your Turn

Gather the 5th through 60th columns of who into a key column: value column pair named codes and n.

Then select just the county, year, codes and n variables.

05:00

```
who %>%
  gather("codes", "n", 5:60) %>%
  select(-iso2, -iso3)
```



	country	year	codes	n
1	Afghanistan	1980	new_sp_m014	NA
2	Afghanistan	1981	new_sp_m014	NA
3	Afghanistan	1982	new_sp_m014	NA
4	Afghanistan	1983	new_sp_m014	NA
5	Afghanistan	1984	new_sp_m014	NA
6	Afghanistan	1985	new_sp_m014	NA
7	Afghanistan	1986	new_sp_m014	NA
8	Afghanistan	1987	new_sp_m014	NA
9	Afghanistan	1988	new_sp_m014	NA
10	Afghanistan	1989	new_sp_m014	NA
11	Afghanistan	1990	new_sp_m014	NA
12	Afghanistan	1991	new_sp_m014	NA

separate()

Quiz

What variables are "hidden" here in plain sight?

storms

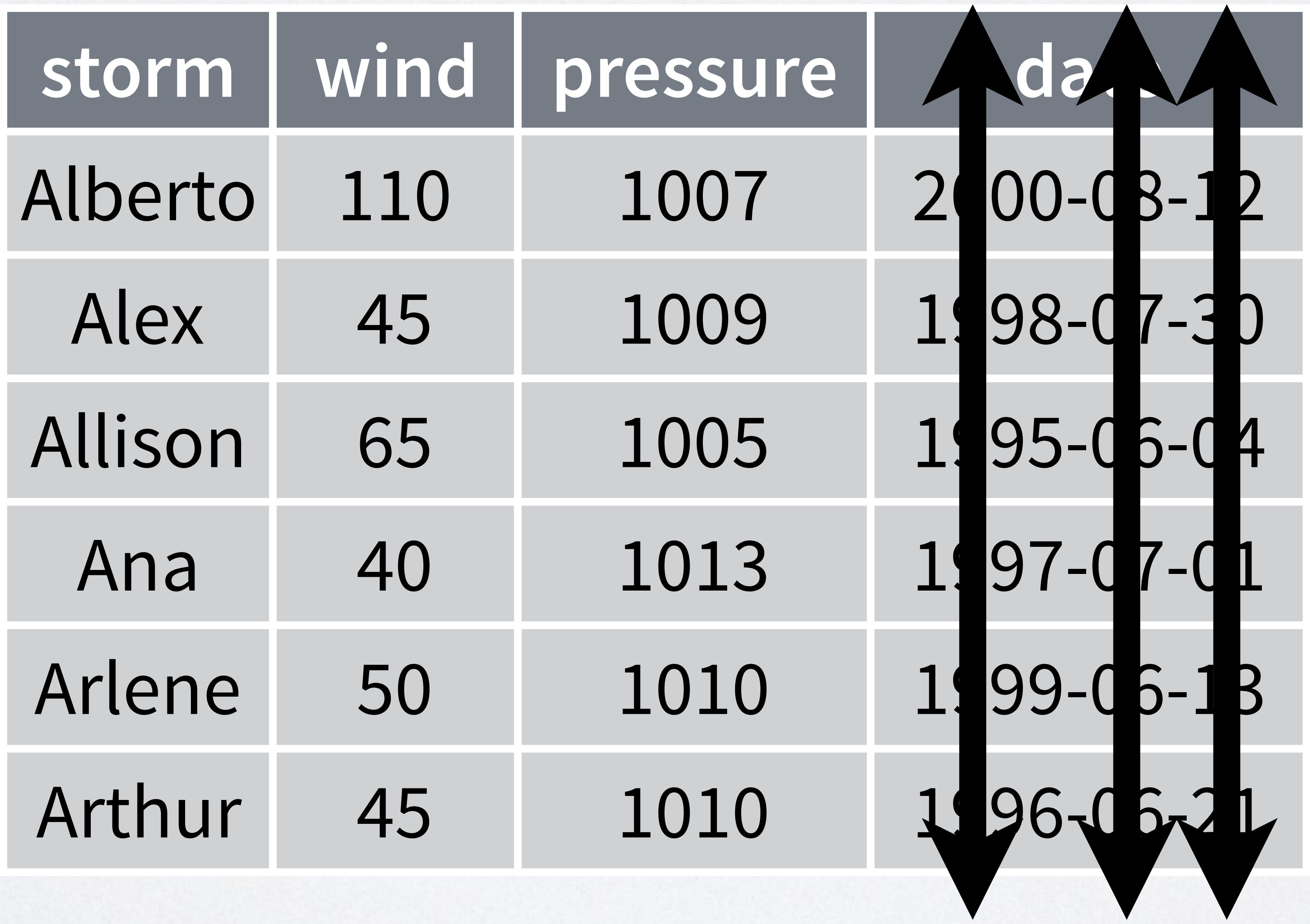
storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

Quiz

What variables are "hidden" here in plain sight?

- year
- month
- day

storms

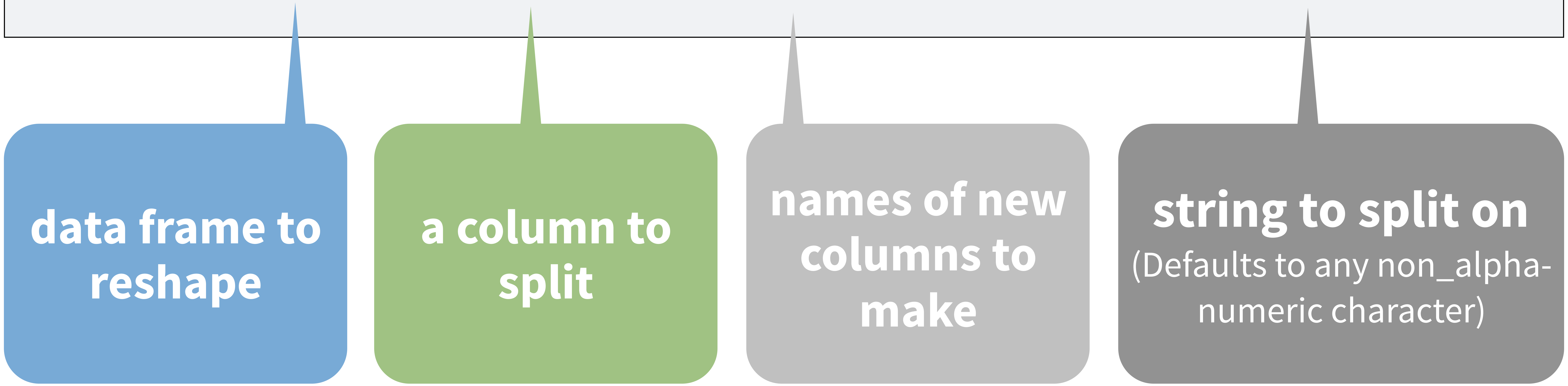


storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

separate()

Splits a column by dividing values at a specific character.

```
separate(storms, date, c("year","month","day"), sep = "-")
```



data frame to reshape

a column to split

names of new columns to make

string to split on
(Defaults to any non_alpha-numeric character)

```
separate(storms, date, c("year","month","day"), sep = "-")
```

```
    storm wind pressure       date
1 Alberto  110     1007 2000-08-03
2    Alex   45     1009 1998-07-27
3 Allison   65     1005 1995-06-03
4     Ana   40     1013 1997-06-30
5  Arlene   50     1010 1999-06-11
6  Arthur   45     1010 1996-06-17
```

```
    storm wind pressure year month day
1 Alberto  110     1007 2000    08  03
2    Alex   45     1009 1998    07  27
3 Allison   65     1005 1995    06  03
4     Ana   40     1013 1997    06  30
5  Arlene   50     1010 1999    06  11
6  Arthur   45     1010 1996    06  17
```

```
separate(storms, date, c("year","month","day"), sep = "-")
```

```
    storm wind pressure       date
1 Alberto  110     1007 2000-08-03
2    Alex   45     1009 1998-07-27
3 Allison   65     1005 1995-06-03
4     Ana   40     1013 1997-06-30
5  Arlene   50     1010 1999-06-11
6  Arthur   45     1010 1996-06-17
```

```
    storm wind pressure   year month day
1 Alberto  110     1007 "2000" "08" "03"
2    Alex   45     1009 "1998" "07" "27"
3 Allison   65     1005 "1995" "06" "03"
4     Ana   40     1013 "1997" "06" "30"
5  Arlene   50     1010 "1999" "06" "11"
6  Arthur   45     1010 "1996" "06" "17"
```

```r
separate(storms, date, c("year","month","day"), sep = "-",
   convert = TRUE)
```

```
    storm wind pressure       date
1 Alberto  110     1007 2000-08-03
2    Alex   45     1009 1998-07-27
3 Allison   65     1005 1995-06-03
4     Ana   40     1013 1997-06-30
5  Arlene   50     1010 1999-06-11
6  Arthur   45     1010 1996-06-17
```

```
    storm wind pressure year month day
1 Alberto  110     1007 2000    08  03
2    Alex   45     1009 1998    07  27
3 Allison   65     1005 1995    06  03
4     Ana   40     1013 1997    06  30
5  Arlene   50     1010 1999    06  11
6  Arthur   45     1010 1996    06  17
```

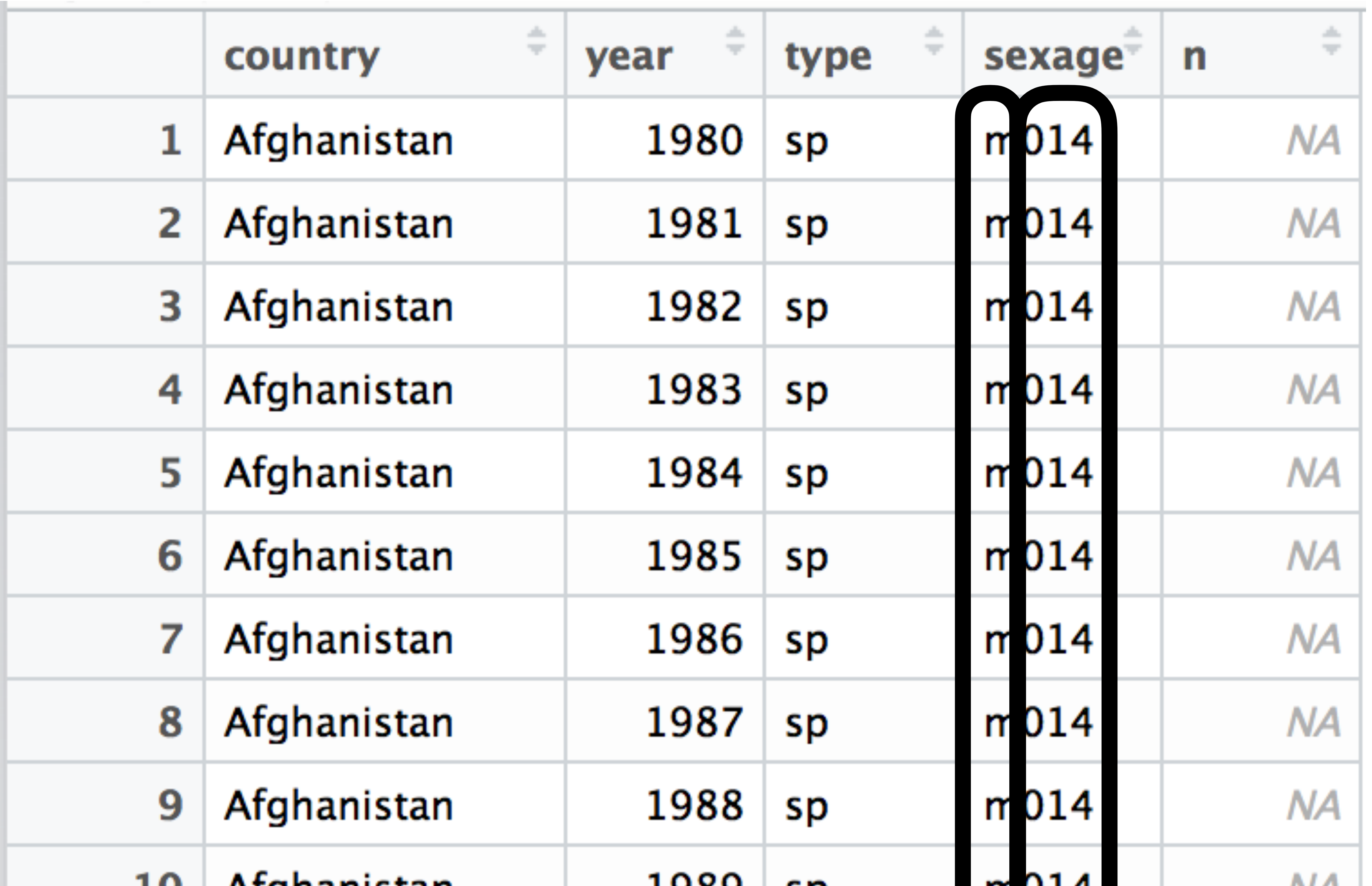
Converts numeric column names to numbers

Your Turn

Separate the codes column into three columns at the underscores. Use the column names "new", "type", "sexage". Then select everything but the "new" column.

02:00

```
who %>%
  gather("codes", "n", 5:60) %>%
  select(-iso2, -iso3) %>%
  separate(codes, c("new", "type", "sexage"), sep = "_") %>%
  select(-new)
```



	country	year	type	sexage	n
1	Afghanistan	1980	sp	m014	NA
2	Afghanistan	1981	sp	m014	NA
3	Afghanistan	1982	sp	m014	NA
4	Afghanistan	1983	sp	m014	NA
5	Afghanistan	1984	sp	m014	NA
6	Afghanistan	1985	sp	m014	NA
7	Afghanistan	1986	sp	m014	NA
8	Afghanistan	1987	sp	m014	NA
9	Afghanistan	1988	sp	m014	NA
10	Afghanistan	1989	sp	m014	NA

separate()

Splits a column by dividing values at a specific character.

```
separate(storms, date, c("year","rest"), sep = c(4,8))
```

locations to split at

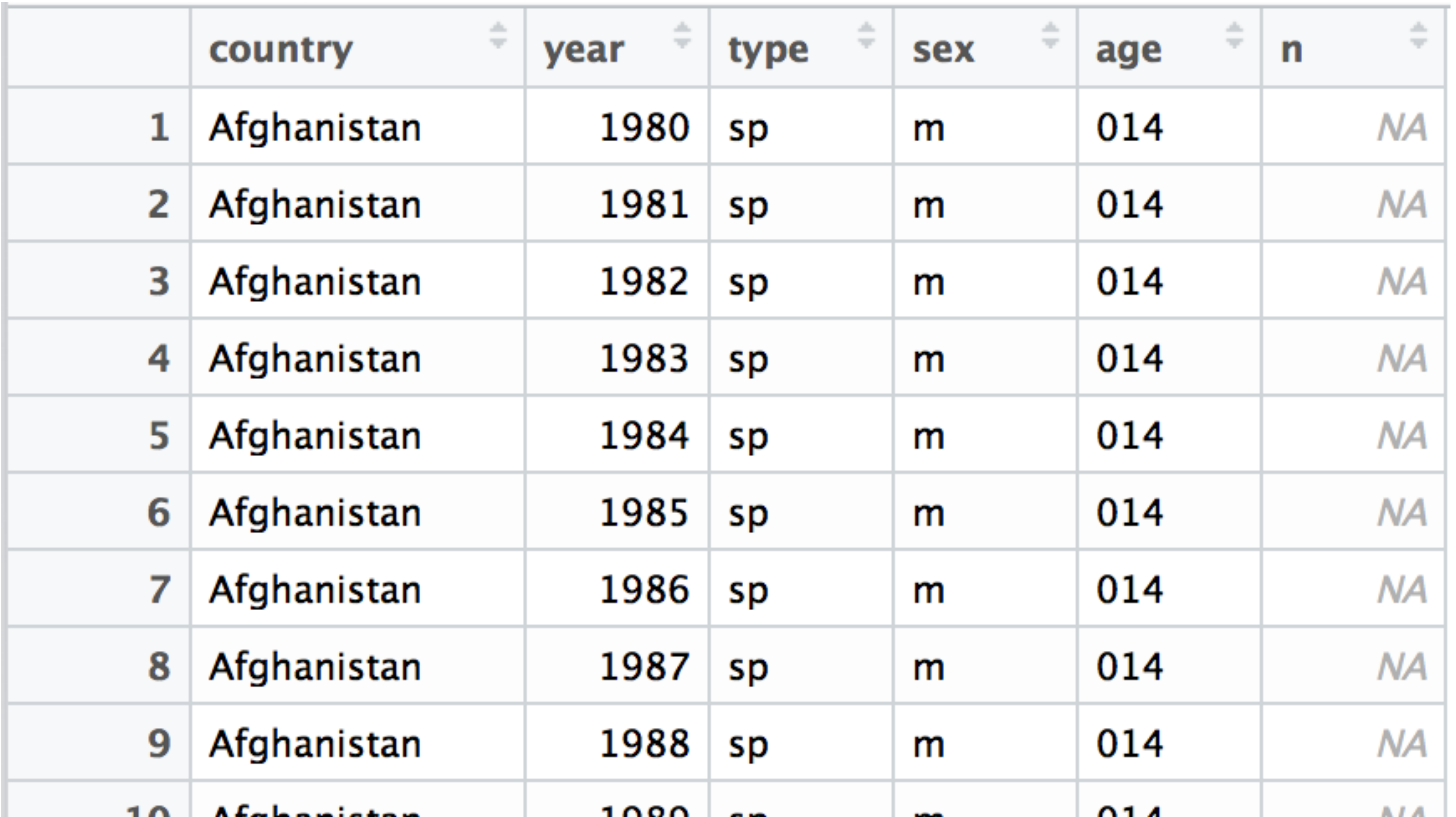
(Split after 4th and 8th characters)

Your Turn

Separate the sexage column into sex and age columns.

01:00

```
who %>%
  gather("codes", "n", 5:60) %>%
  select(-iso2, -iso3) %>%
  separate(codes, c("new", "type", "sexage"), sep = "_") %>%
  select(-new) %>%
  separate(sexage, c("sex", "age"), sep = 2)
```



	country	year	type	sex	age	n
1	Afghanistan	1980	sp	m	014	NA
2	Afghanistan	1981	sp	m	014	NA
3	Afghanistan	1982	sp	m	014	NA
4	Afghanistan	1983	sp	m	014	NA
5	Afghanistan	1984	sp	m	014	NA
6	Afghanistan	1985	sp	m	014	NA
7	Afghanistan	1986	sp	m	014	NA
8	Afghanistan	1987	sp	m	014	NA
9	Afghanistan	1988	sp	m	014	NA
10	Afghanistan	1989	sp	m	014	NA

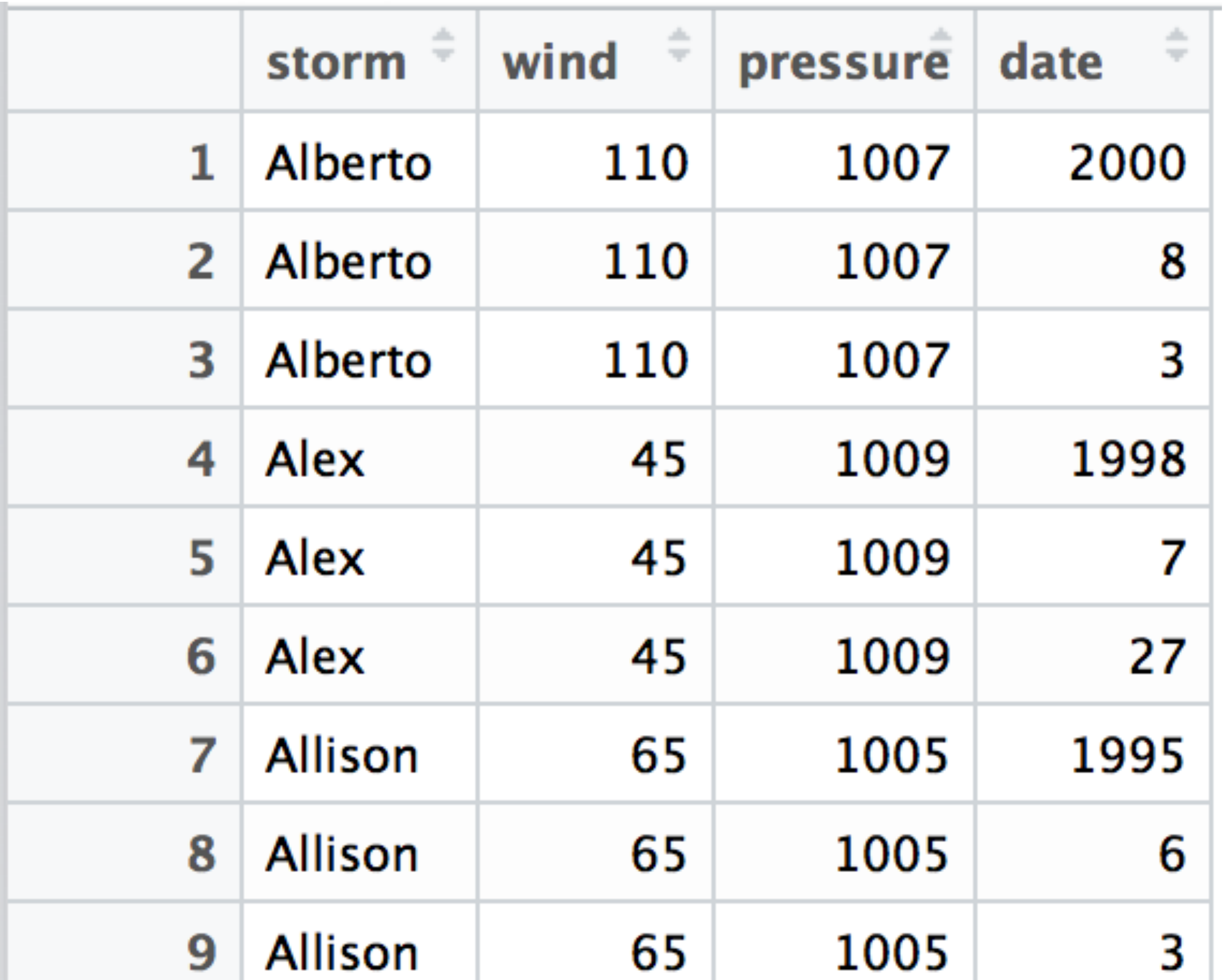
separate_rows()

Splits a column. Creates a new row for each result.

```
separate_rows(storms, date, sep = "-", convert = TRUE)
```

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

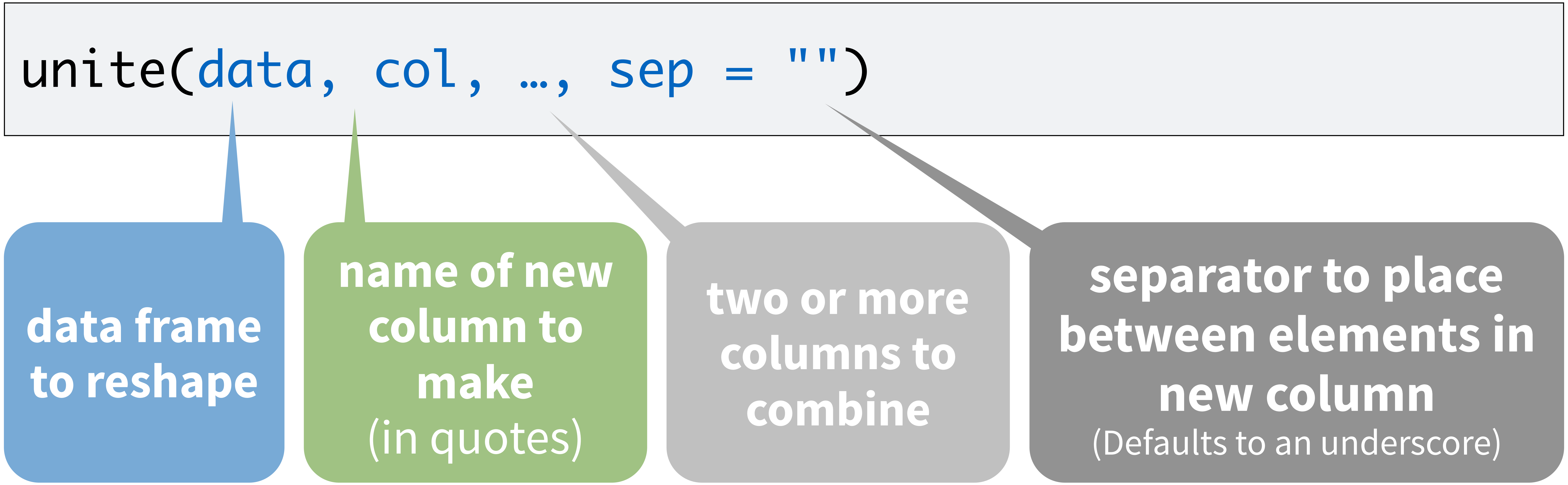


	storm	wind	pressure	date
1	Alberto	110	1007	2000
2	Alberto	110	1007	8
3	Alberto	110	1007	3
4	Alex	45	1009	1998
5	Alex	45	1009	7
6	Alex	45	1009	27
7	Allison	65	1005	1995
8	Allison	65	1005	6
9	Allison	65	1005	3

unite()

unite()

Unites columns into single column by combining cells.



```
unite(data, col, …, sep = "")
```

data frame to reshape

name of new column to make
(in quotes)

two or more columns to combine

separator to place between elements in new column
(Defaults to an underscore)

Your Turn

Use separate() and then unite() to change how storms codes date, as below.

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

→

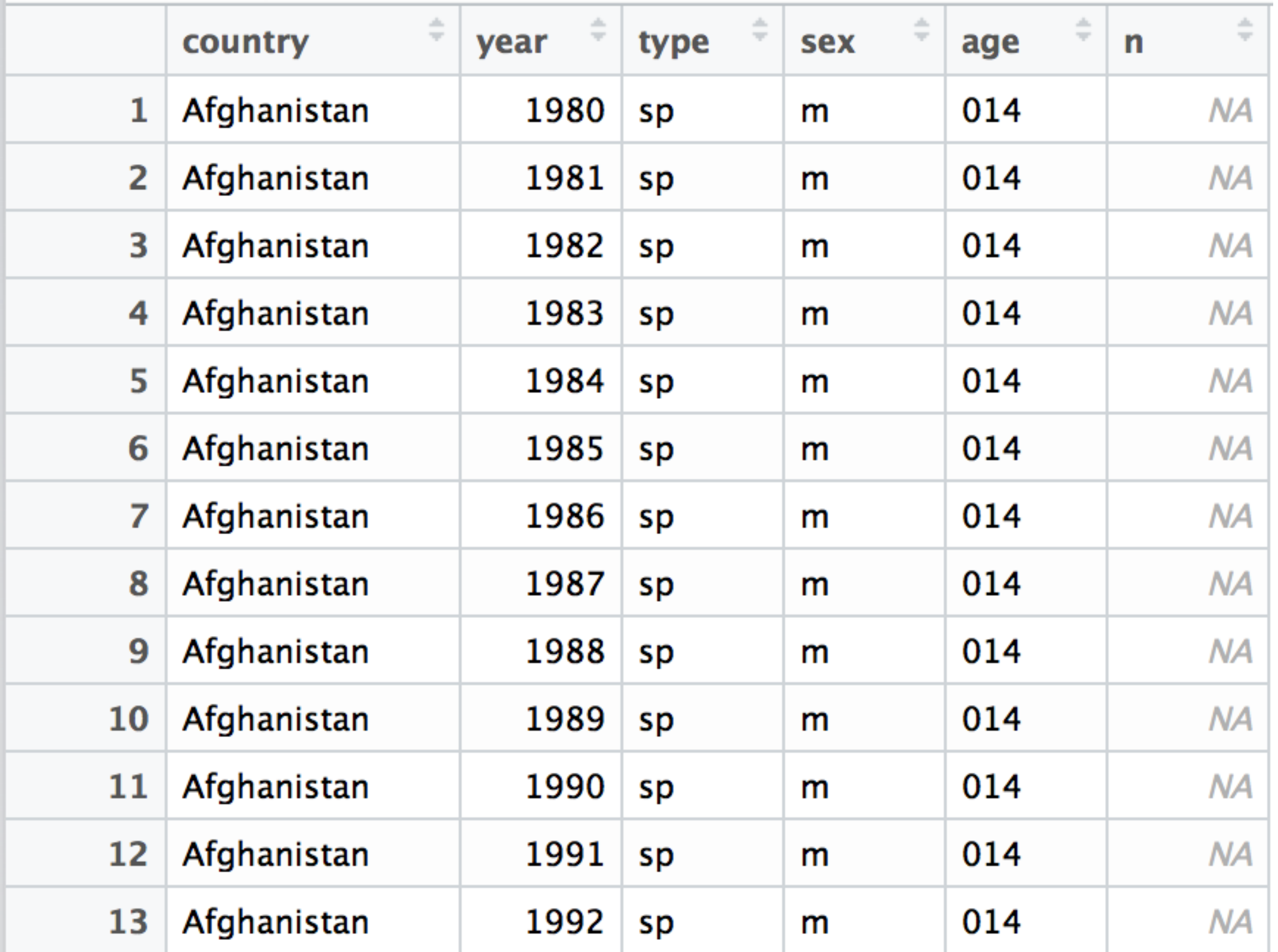
storm	wind	pressure	date
Alberto	110	1007	08/12/2000
Alex	45	1009	07/30/1998
Allison	65	1005	06/04/1995
Ana	40	1013	07/01/1997
Arlene	50	1010	06/13/1999
Arthur	45	1010	06/21/1996

03:00

```
storms %>%
  separate(date, c("year", "month", "day"), sep = "-") %>%
  unite("date", month, day, year, sep = "/")
```

Missing Values

Can we clean up the missing values?



	country	year	type	sex	age	n
1	Afghanistan	1980	sp	m	014	NA
2	Afghanistan	1981	sp	m	014	NA
3	Afghanistan	1982	sp	m	014	NA
4	Afghanistan	1983	sp	m	014	NA
5	Afghanistan	1984	sp	m	014	NA
6	Afghanistan	1985	sp	m	014	NA
7	Afghanistan	1986	sp	m	014	NA
8	Afghanistan	1987	sp	m	014	NA
9	Afghanistan	1988	sp	m	014	NA
10	Afghanistan	1989	sp	m	014	NA
11	Afghanistan	1990	sp	m	014	NA
12	Afghanistan	1991	sp	m	014	NA
13	Afghanistan	1992	sp	m	014	NA

Toy data

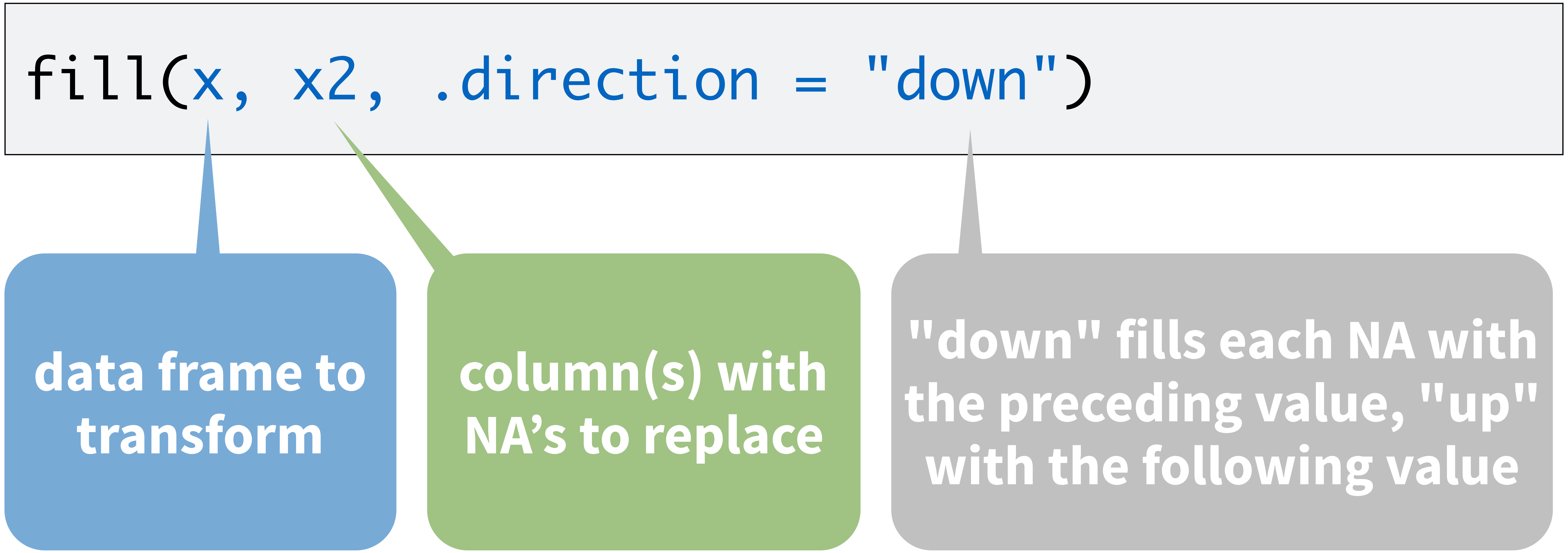
```
x <- tribble(
  ~x1, ~x2,
  "A",   1,
  "B",  NA,
  "C",  NA,
  "D",   3,
  "E",  NA
)
```

x

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

fill()

Fills in NA’s with preceding (or following) values.



```
fill(x, x2, .direction = "down")
```

data frame to transform

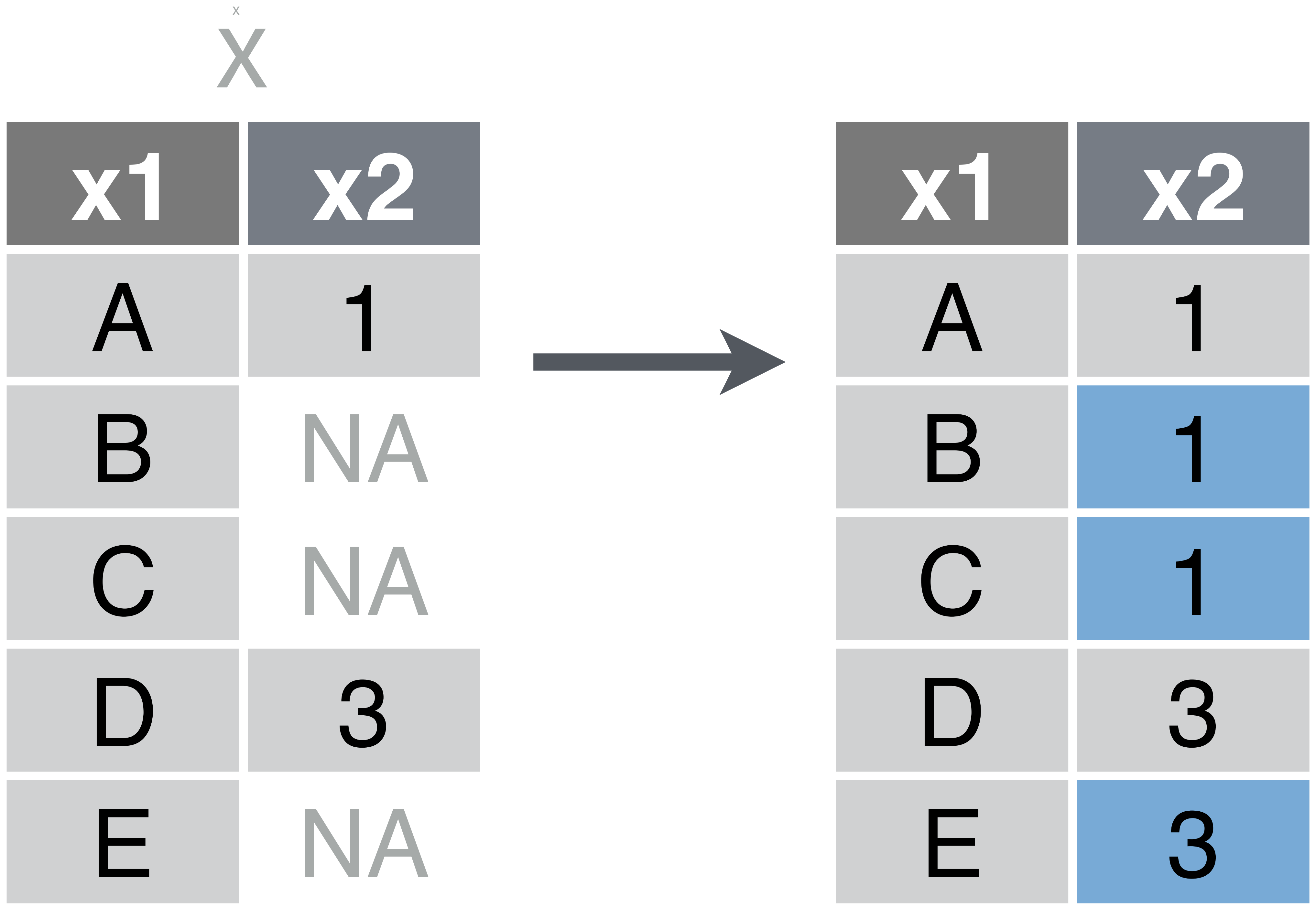
column(s) with NA’s to replace

"down" fills each NA with the preceding value, "up" with the following value

fill()

Fills in NA’s with preceding (or following) values.

```
fill(x, x2, .direction = "down")
```



x

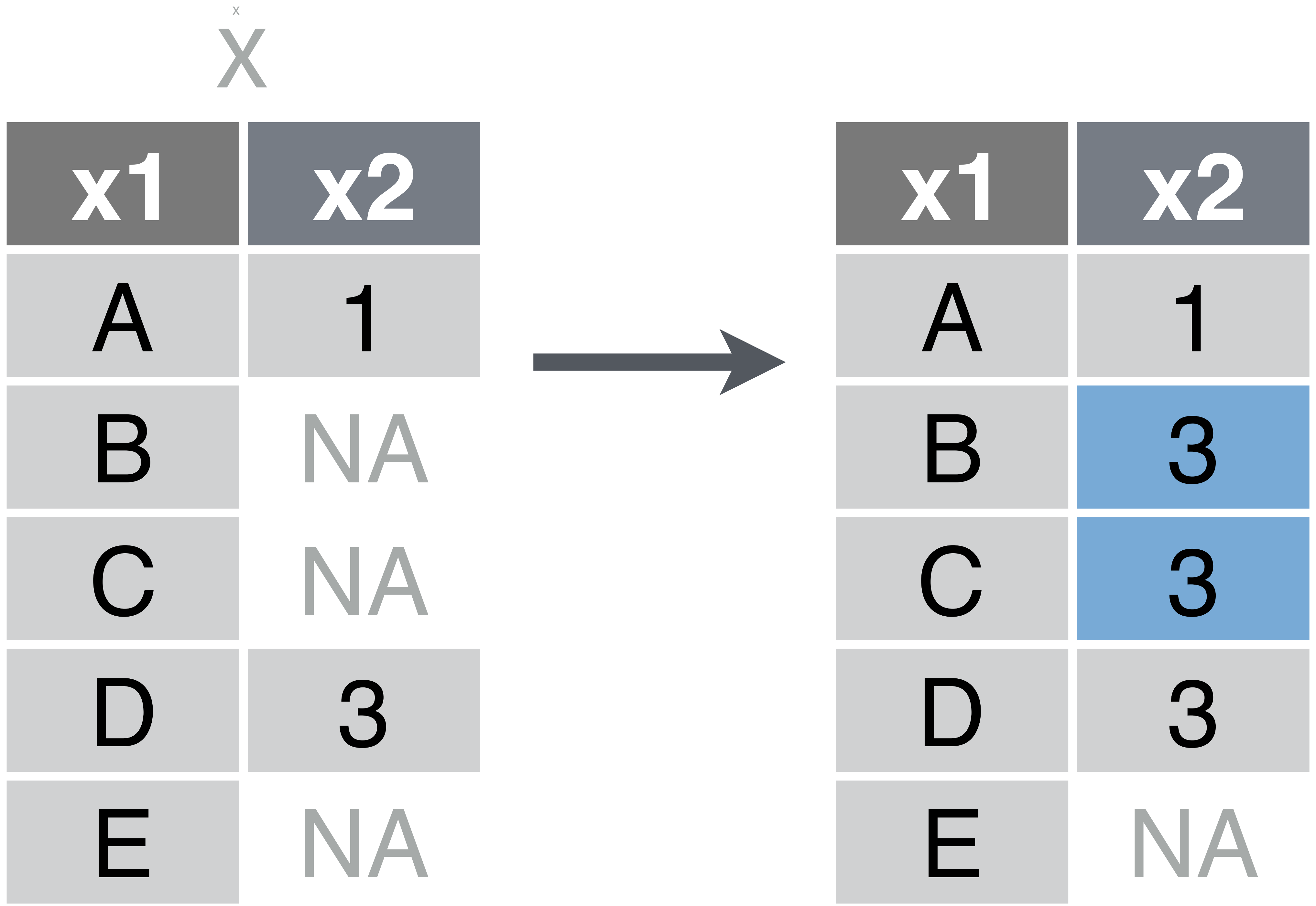
x1	x2
A	1
B	NA
C	NA
D	3
E	NA

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

fill()

Fills in NA’s with preceding (or following) values.

```
fill(x, x2, .direction = "up")
```



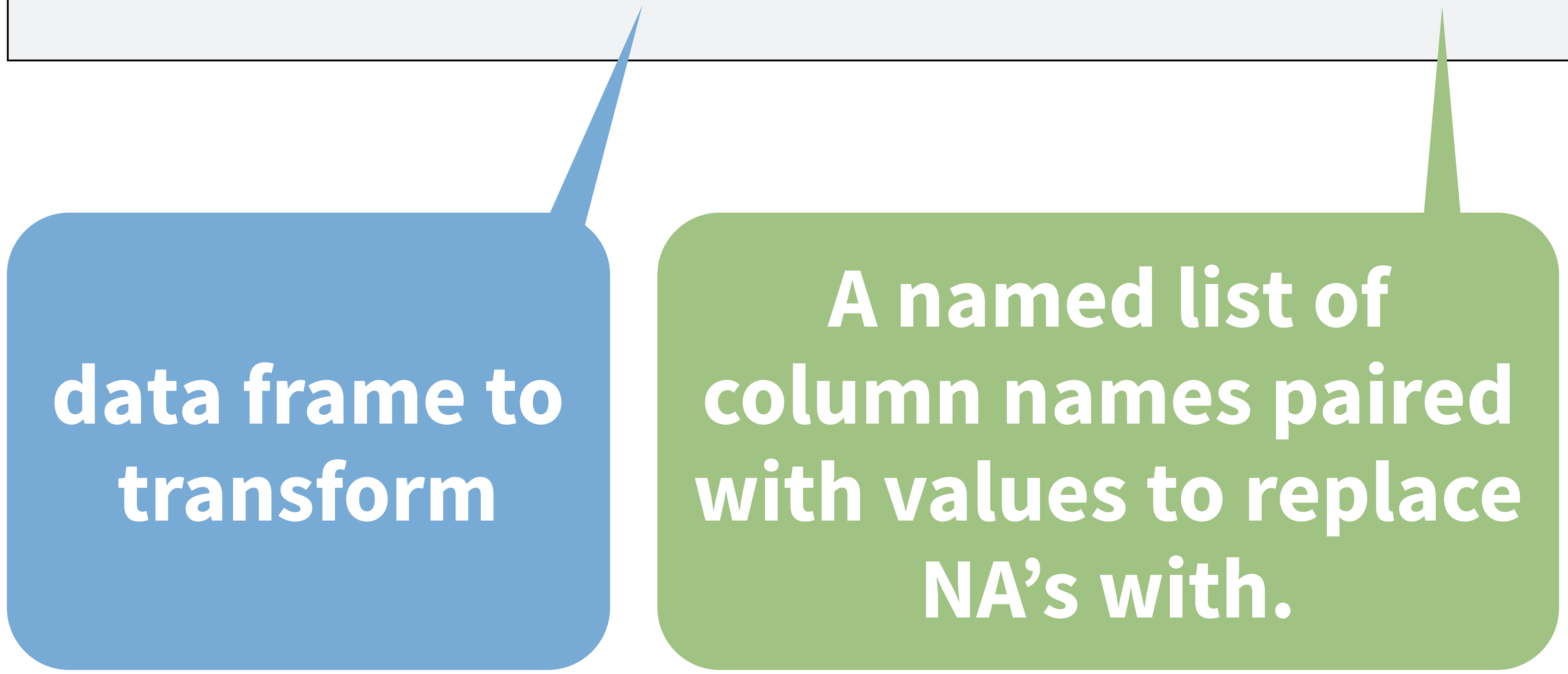
x1	x2
A	1
B	NA
C	NA
D	3
E	NA

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

replace_na()

Replace NA’s by column.

```
replace_na(x, replace = list(x2 = 2))
```



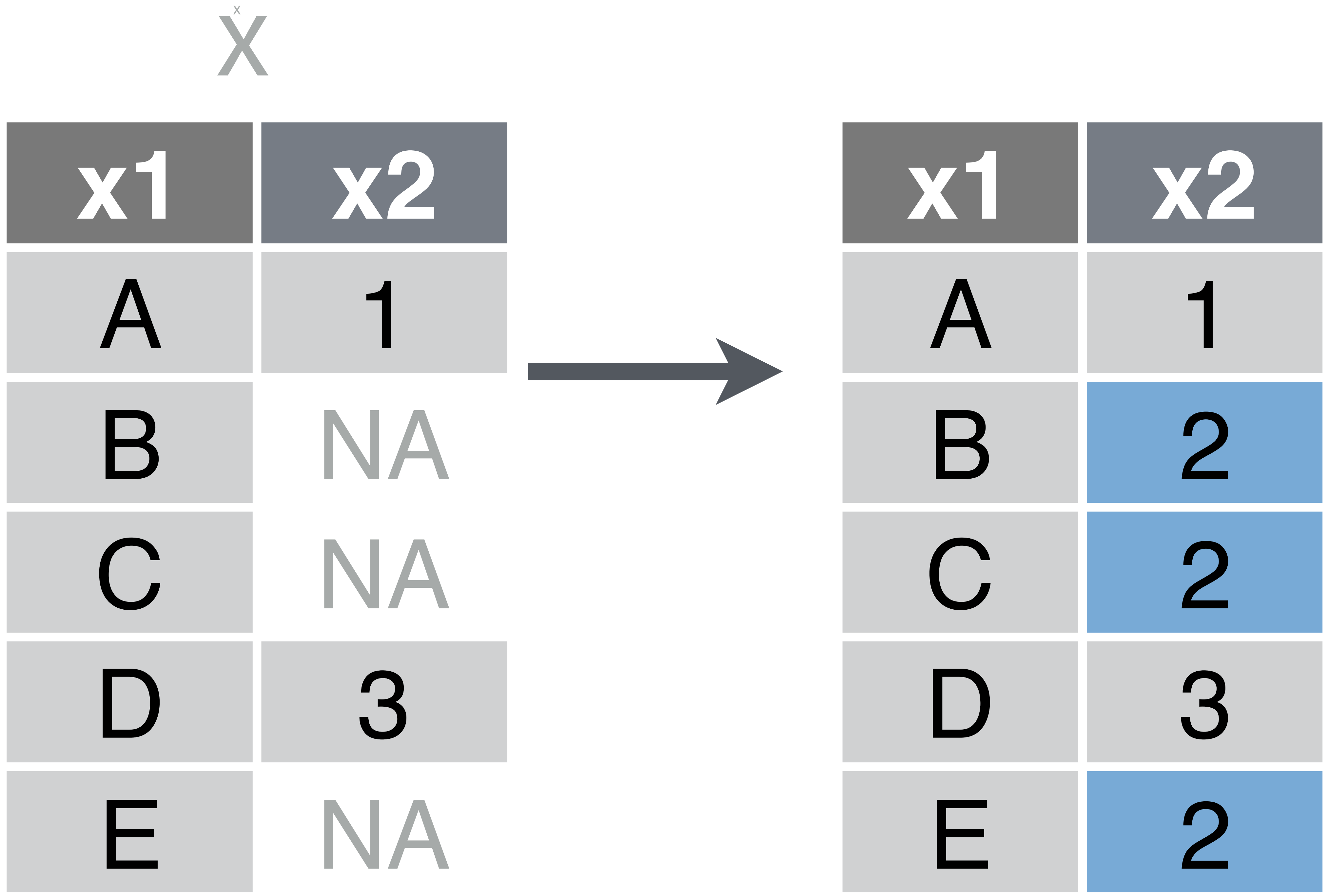
data frame to transform

A named list of column names paired with values to replace NA’s with.

replace_na()

Replace NA's by column.

```
replace_na(x, replace = list(x2 = 2))
```



x

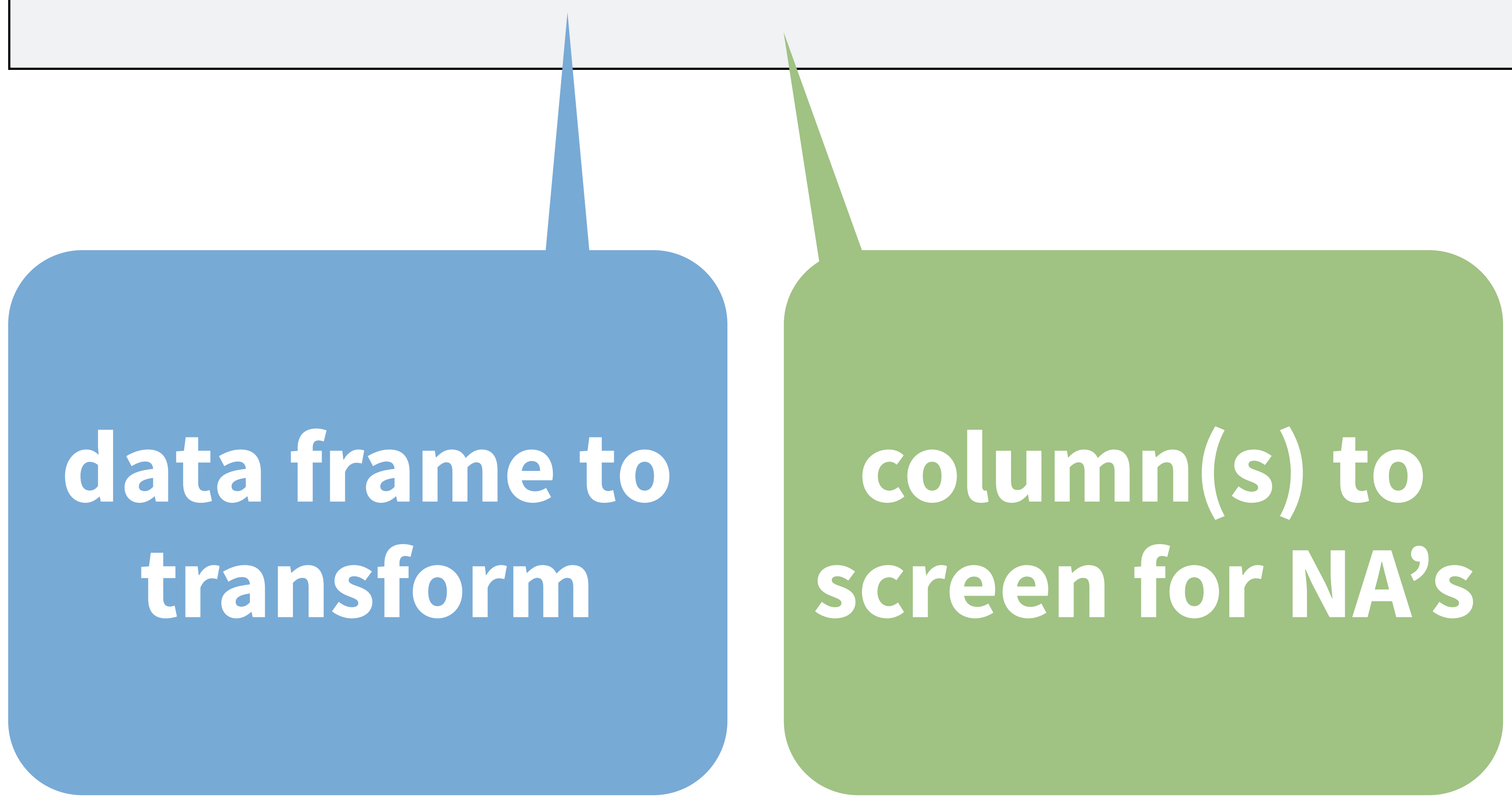
x1	x2
A	1
B	NA
C	NA
D	3
E	NA

x1	x2
A	1
B	2
C	2
D	3
E	2

drop()

Drops rows that contain NA’s in the specified columns.

```
drop_na(x, x2)
```



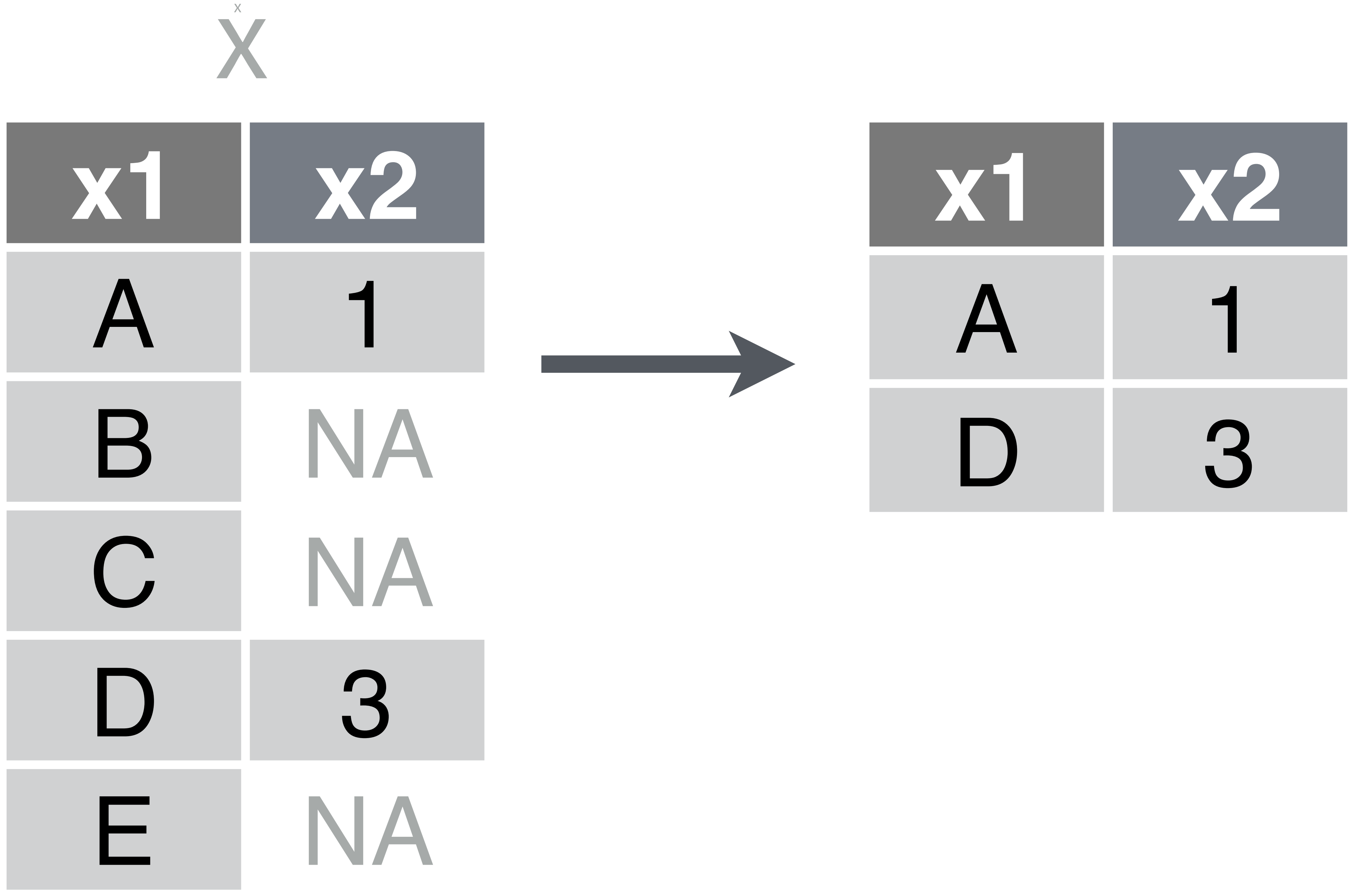
data frame to transform

column(s) to screen for NA’s

drop_na()

Drops rows that contain NA’s in the specified columns.

```
drop_na(x, x2)
```

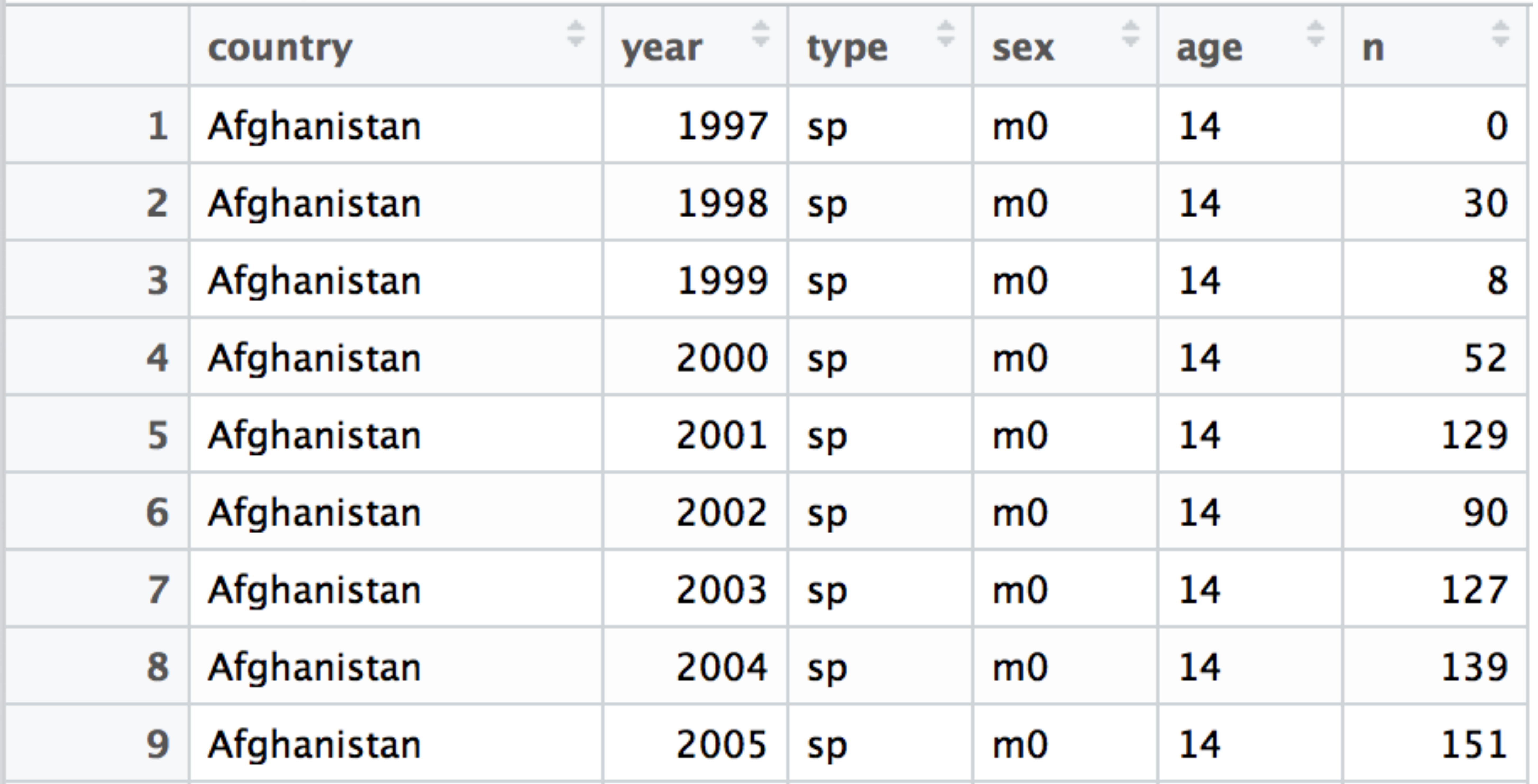


x

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

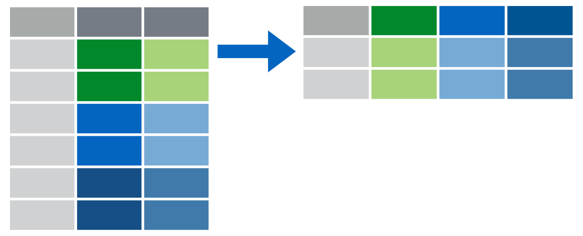
x1	x2
A	1
D	3

```
who %>%
  gather("codes", "n", 5:60) %>%
  separate(codes, c("new", "type", "sexage"), sep = "_") %>%
  select(-new, -iso2, -iso3) %>%
  separate(sexage, c("sex", "age"), sep = 2) %>%
  drop_na(n)
```

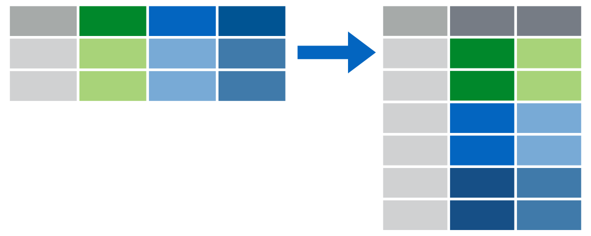


	country	year	type	sex	age	n
1	Afghanistan	1997	sp	m0	14	0
2	Afghanistan	1998	sp	m0	14	30
3	Afghanistan	1999	sp	m0	14	8
4	Afghanistan	2000	sp	m0	14	52
5	Afghanistan	2001	sp	m0	14	129
6	Afghanistan	2002	sp	m0	14	90
7	Afghanistan	2003	sp	m0	14	127
8	Afghanistan	2004	sp	m0	14	139
9	Afghanistan	2005	sp	m0	14	151

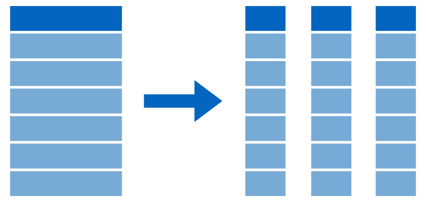
Recap



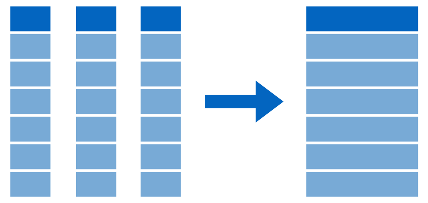
Move values into column names with **spread()**



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



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



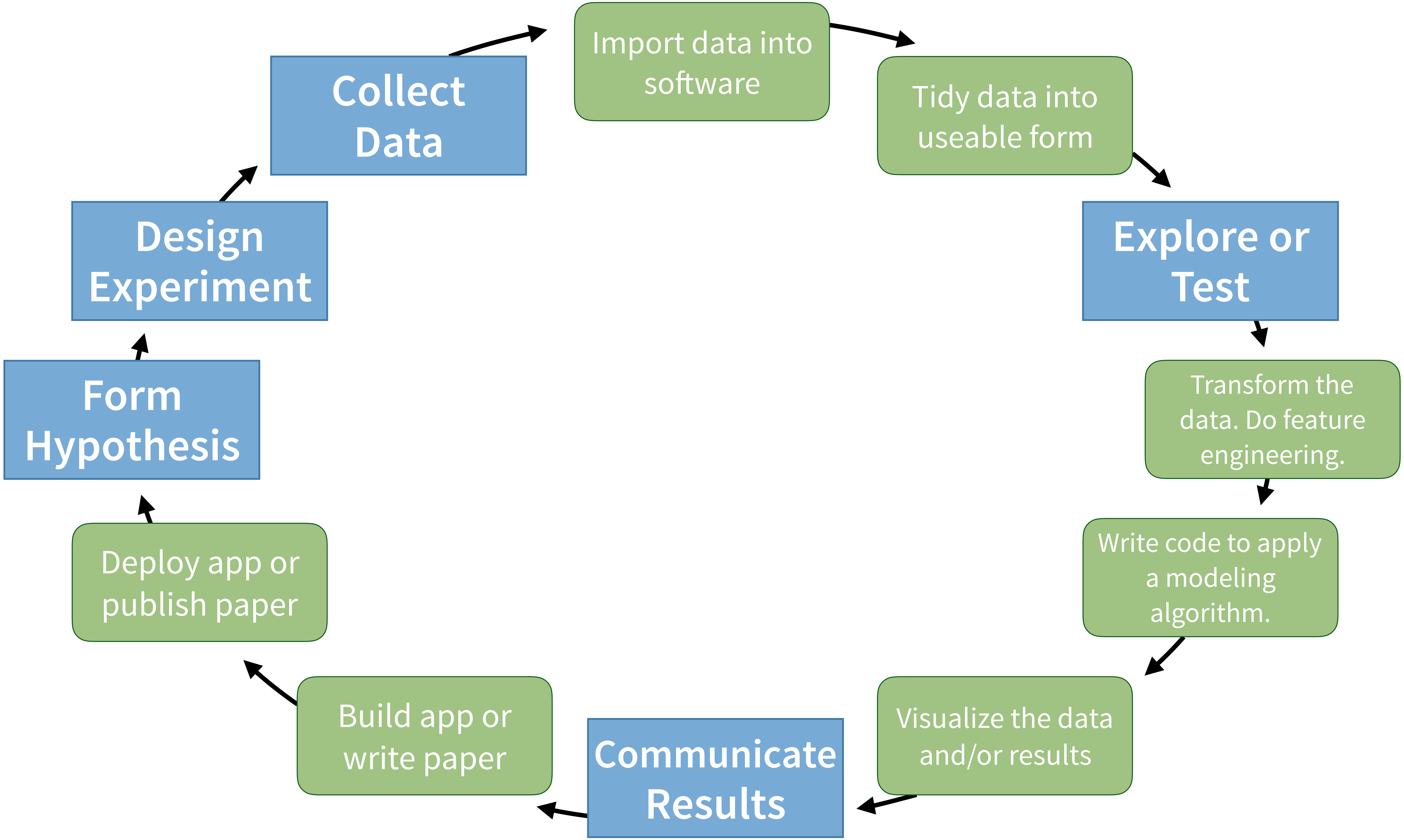
Unite columns with **unite()**

The role of tidy data in the tidyverse

Tidy data

Tidy functions **all expect and return the same data structure**, known as tidy data:

1. A **data frame** that contains
2. **variables** in the **columns** and
3. **cases** in the **rows**.



Form Hypothesis

Design Experiment

Collect Data

Import data into software

Tidy data into useable form

Explore or Test

Transform the data. Do feature engineering.

Write code to apply a modeling algorithm.

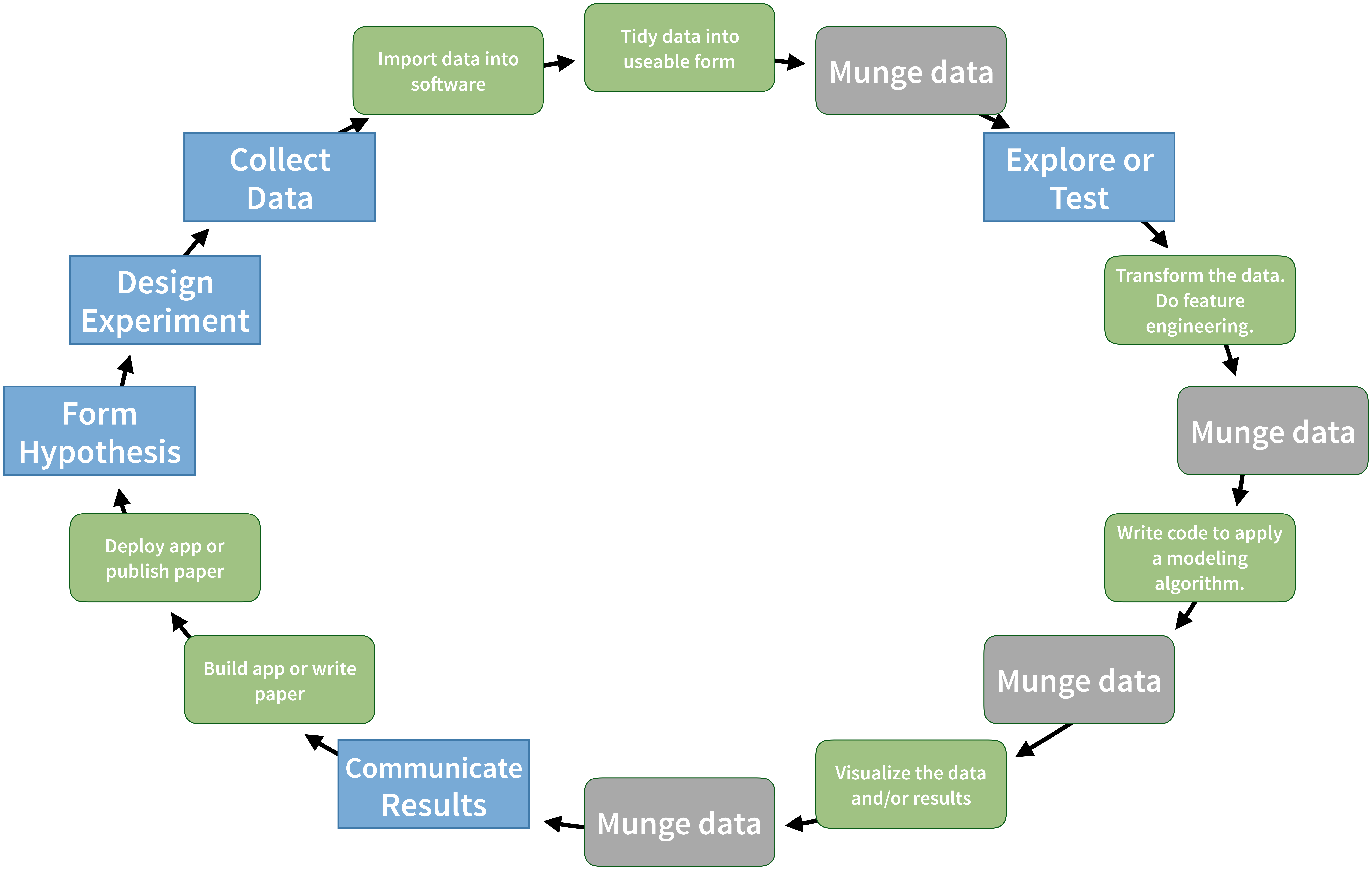
Visualize the data and/or results

Communicate Results

Build app or write paper

Deploy app or publish paper

tidyr



Form Hypothesis

Design Experiment

Collect Data

Import data into software

Tidy data into useable form

Munge data

Explore or Test

Transform the data. Do feature engineering.

Munge data

Write code to apply a modeling algorithm.

Munge data

Visualize the data and/or results

Munge data

Communicate Results

Build app or write paper

Deploy app or publish paper

Tidy data

Tidy functions all expect and return the same data structure, known as **tidy data**:

1. A **data frame** that contains
2. **variables** in the **columns** and
3. **cases** in the **rows**.

Which is tidy? What is a variable?

```
layout1 %>%
  ggplot(aes(year, n)) +
  geom_line(aes(color = sex))
```

```
   year   sex      n
  <dbl> <chr>  <int>
1  1880     F  90993
2  1880     M 110491
3  1881     F  91954
4  1881     M 100745
5  1882     F 107850
6  1882     M 113688
```

```
layout2 %>%
  mutate(pmale = M / (M + F))
```

```
   year      F      M
*  <dbl>  <int>  <int>
1  1880  90993 110491
2  1881  91954 100745
3  1882 107850 113688
4  1883 112321 104629
5  1884 129022 114445
6  1885 133055 107800
```

General advice

Describe what you want to do in an **equation**. Each variable in the equation should correspond to a variable in your data:

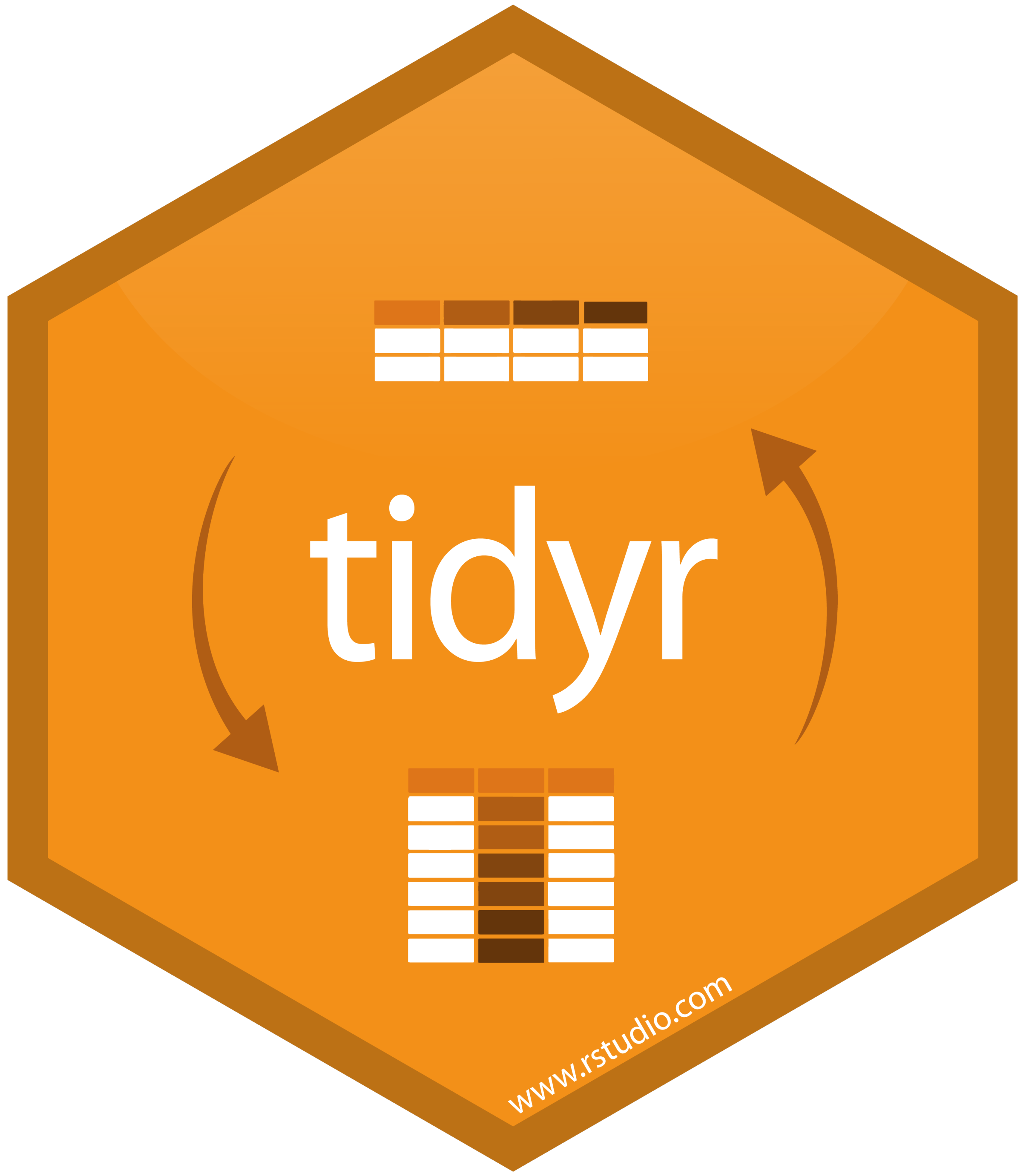
- "color by sex"

  color = sex

- "calculate the proportion of males"

  prop male = number of males / number of females + number of males

Tidy Data with



Reshaping Final Exam

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

- Hadley Wickham

Your Turn

In your grous, use **spread()**, **gather()**, **separate()**, and/or **unite()** to tidy each of the following tables:

- table2
- table3
- table4a - does not contain population
- table4b - does not contain cases
- table5

Unless otherwise specified, each contains a country, year, cases, and population variable.

```
table2 %>%
  spread(type, count)
```

```
table3 %>%
  separate(rate, c("cases", "population"), sep = "/",
    convert = TRUE)
```

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

```
table4b %>%
  gather("year", "population", 2:3, convert = TRUE)
```

```
table5 %>%
  unite("year", century, year, sep = "") %>%
  separate(rate, c("cases", "population"), sep = "/",
    convert = TRUE)
```