Video 19: pivoting tables with tidyr

Stats 102A

Miles Chen, PhD

Pivoting Data

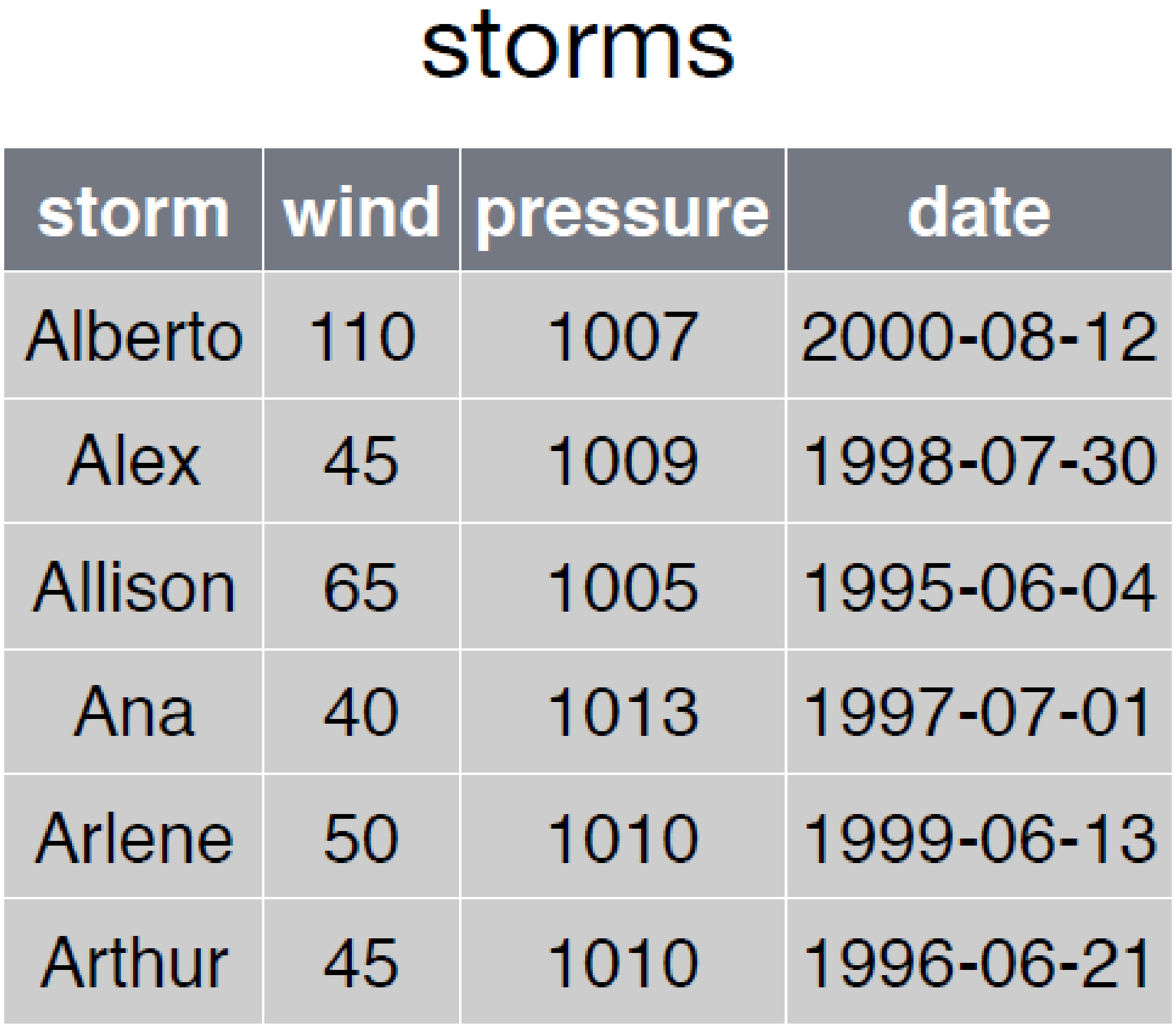
Tidy Data

The Philosophy of the Tidyverse

There are three rules which make a data set tidy:

- Every column is variable.
- Every row is an observation.
- Every cell is a single value.

Tidy Data

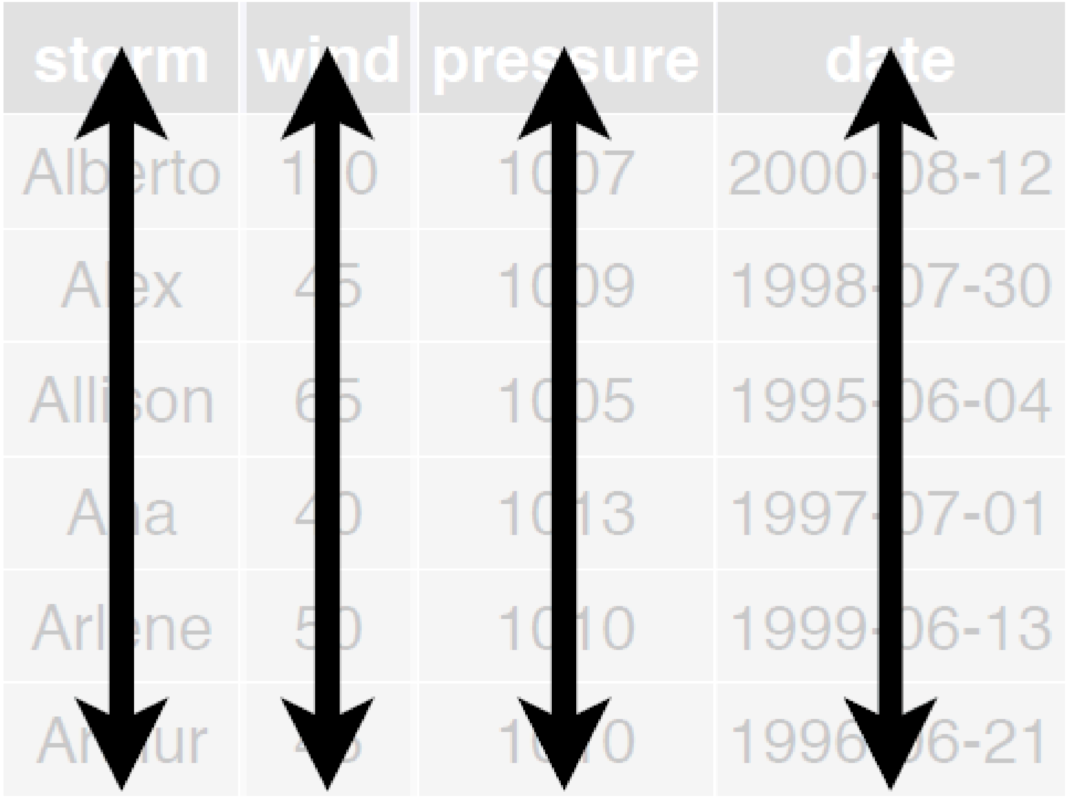


storms

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

Tidy Data

storms

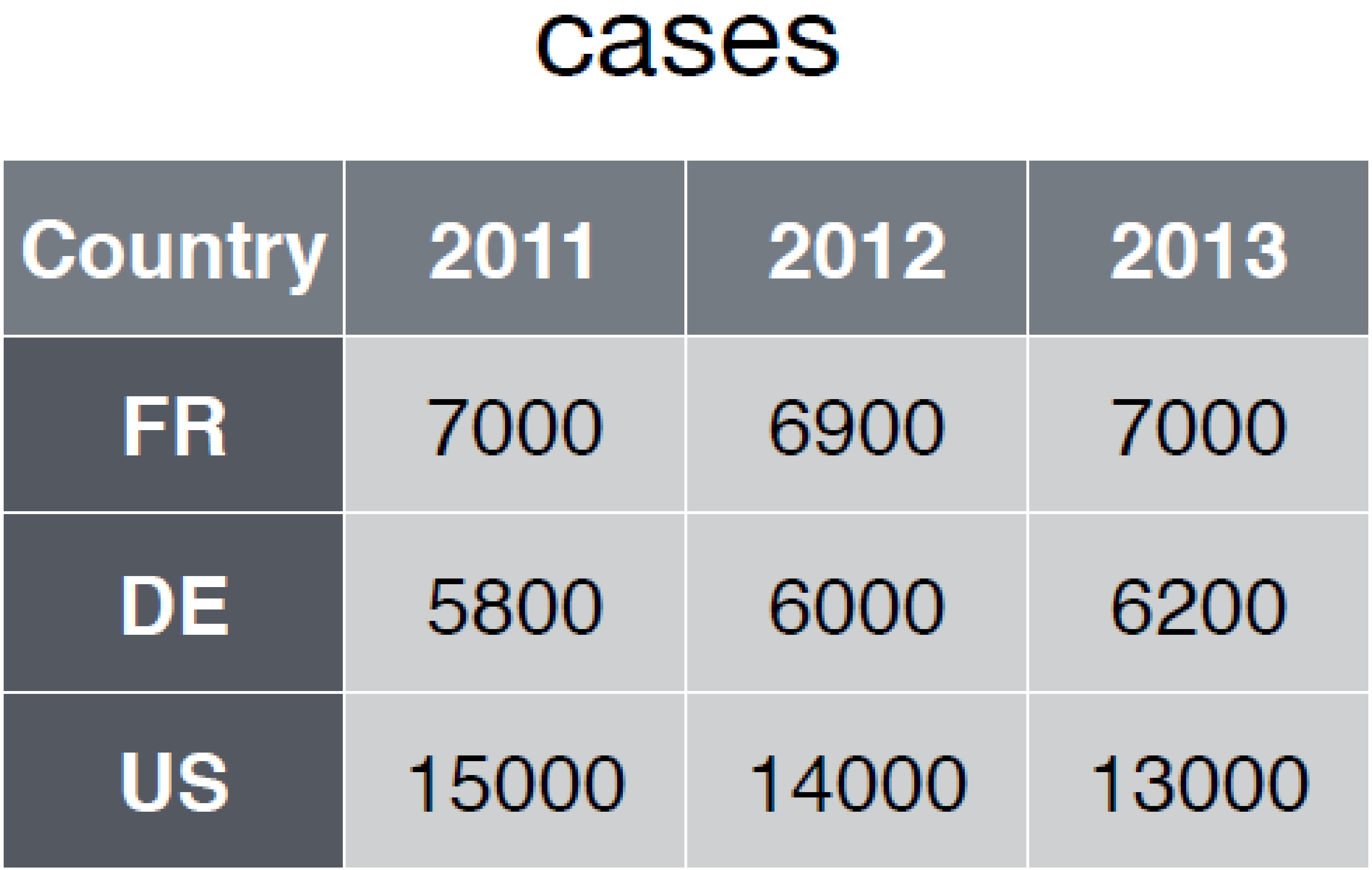


storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

- Storm name
- Wind Speed (mph)
- Air Pressure
- Date

We have one column for each variable

Not Tidy Data

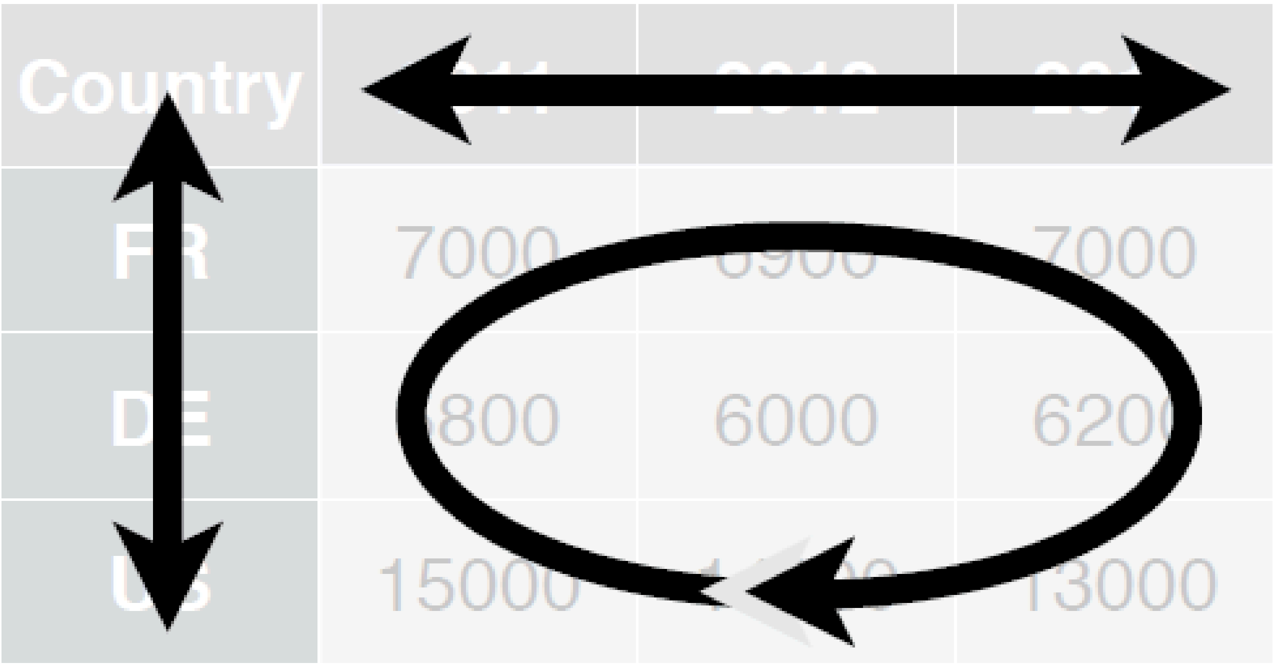


cases

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

Not Tidy Data

cases

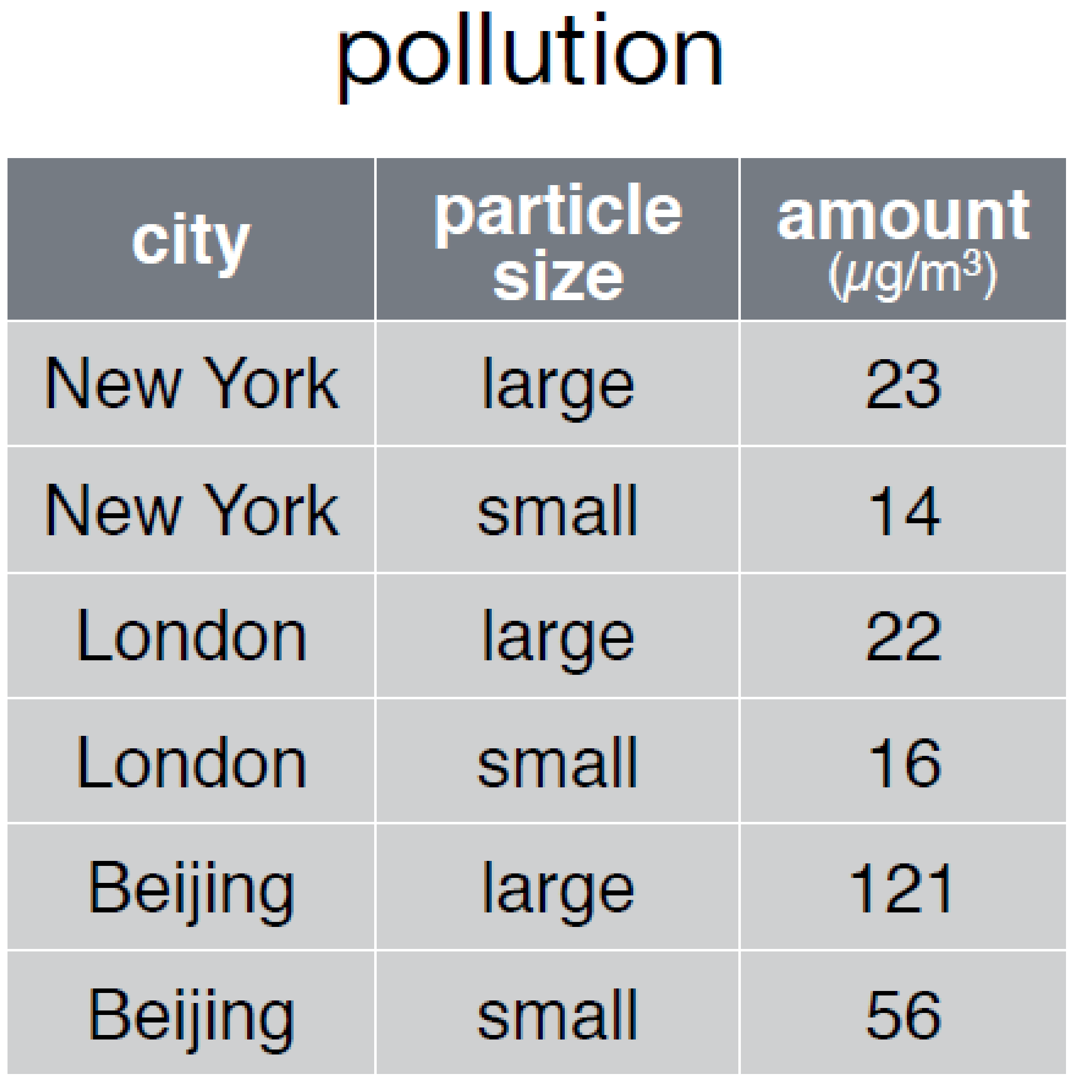


Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

- Country
- Year
- Count

One variable forms column headings, and the values are spread out across columns.

Not Tidy Data

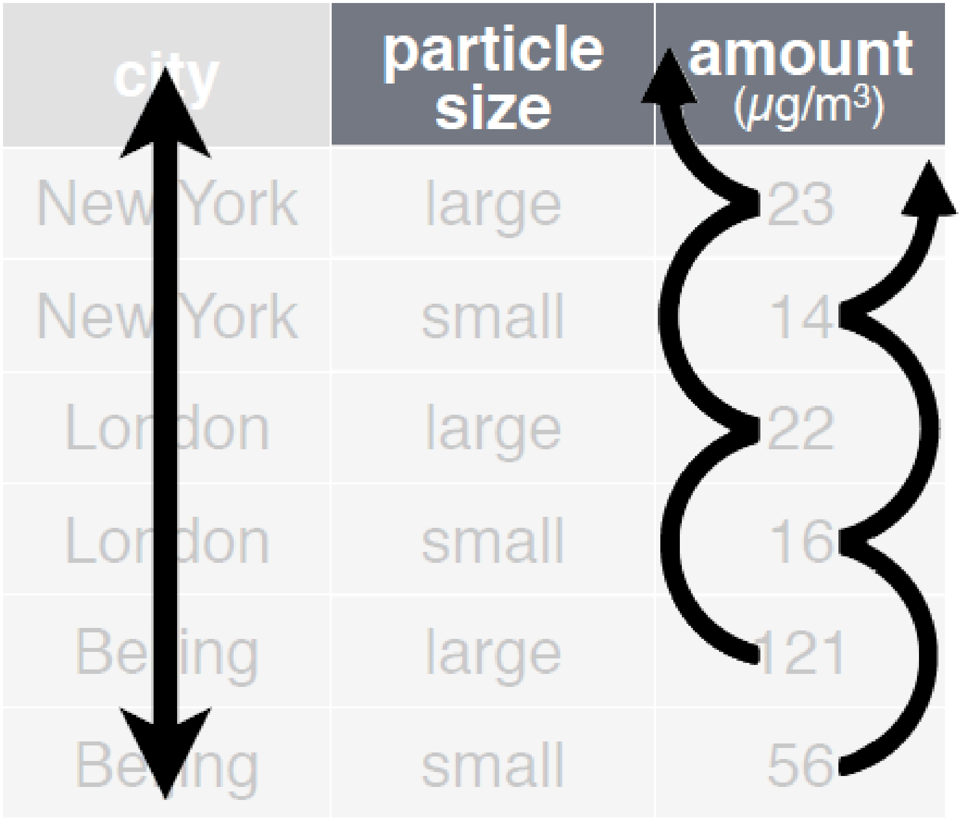


pollution

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

Not Tidy Data

pollution



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

- City
- Amount of large particles
- Amount of small particles

The values of two different variables are stored in one column.

Reading in the data

```r
storms <- read_csv("https://raw.githubusercontent.com/rstudio/EDAWR
cases <- read_csv("https://raw.githubusercontent.com/rstudio/EDAWR/
pollution <- read_csv("https://raw.githubusercontent.com/rstudio/ED
```

Vectorized operations work for tidy data

```r
storms$ratio <- storms$pressure / storms$wind
# 1007 / 110 = 9.15, 1009 / 45 = 22.4, etc.
storms
```

```
# A tibble: 6 × 5
  storm     wind pressure date        ratio
  <chr>    <dbl>    <dbl> <date>      <dbl>
1 Alberto    110     1007 2000-08-03   9.15
2 Alex        45     1009 1998-07-27  22.4
3 Allison     65     1005 1995-06-03  15.5
4 Ana         40     1013 1997-06-30  25.3
5 Arlene      50     1010 1999-06-11  20.2
6 Arthur      45     1010 1996-06-17  22.4
```

The Cases table

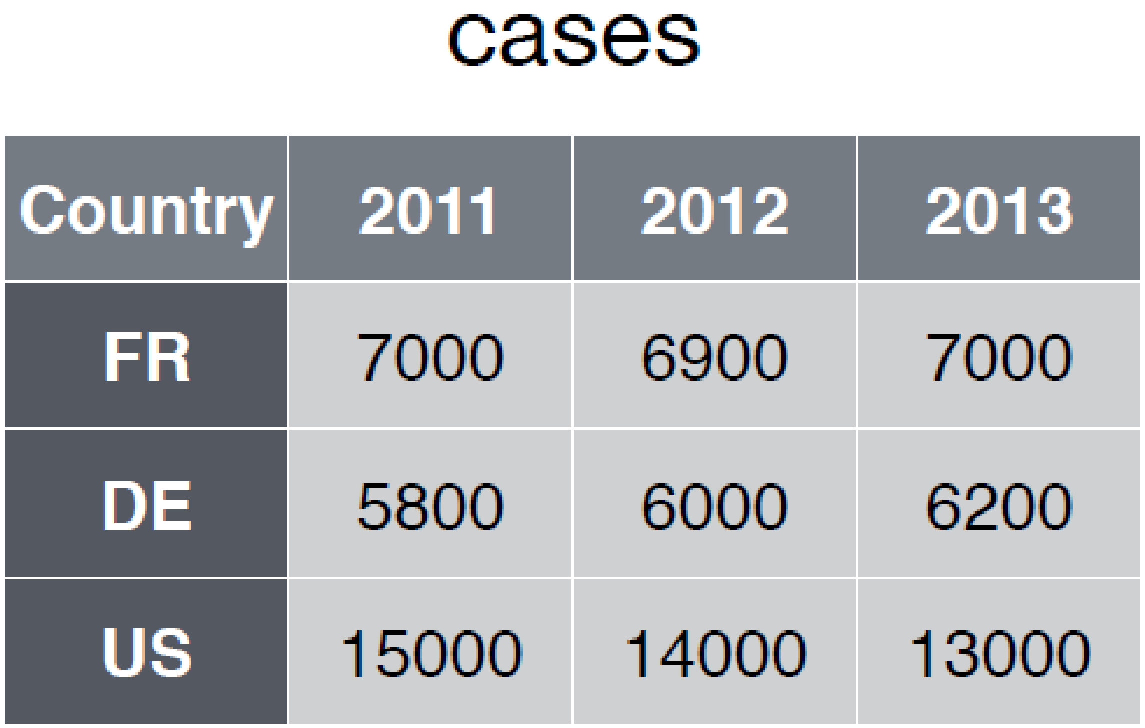
```
cases
```

```
# A tibble: 3 × 4
  country `2011` `2012` `2013`
  <chr>    <dbl>  <dbl>  <dbl>
1 FR        7000   6900   7000
2 DE        5800   6000   6200
3 US       15000  14000  13000
```

Getting things tidy

The variables are: country, year, count

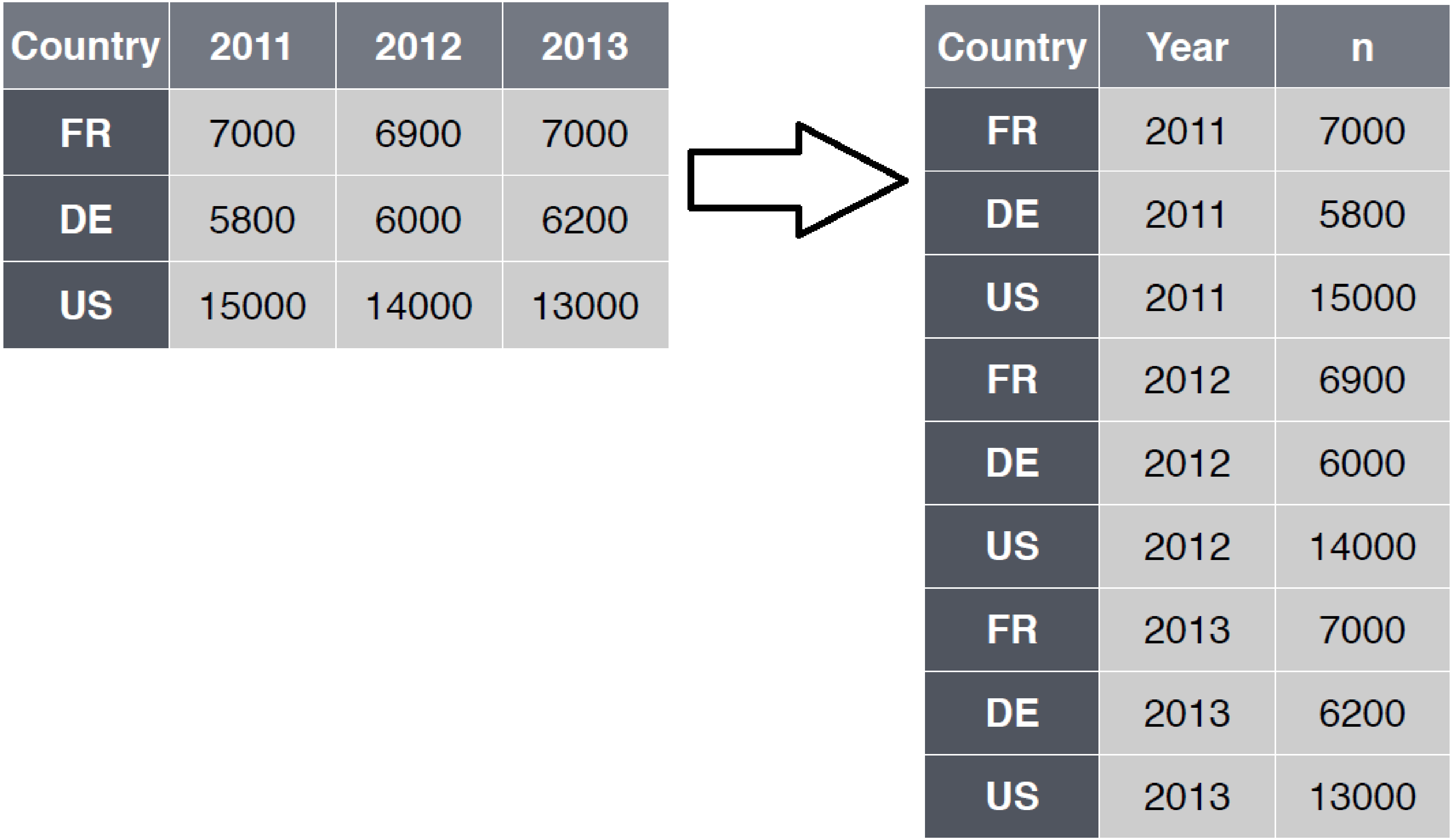
If we want to make this tidy, what will be the dimensions of the resulting data?



cases

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

Result: a 9 x 3 tibble



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

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

pivot_longer()

To achieve the desired result, we use the function `pivot_longer()` because we want the resulting data set to be longer than the original data. (older versions called this `gather()`)

```r
pivot_longer(cases,
             cols = "2011":"2013",
             names_to = "year",
             values_to = "cases")
```

```
# A tibble: 9 × 3
  country year  cases
  <chr>   <chr> <dbl>
1 FR      2011   7000
2 FR      2012   6900
3 FR      2013   7000
4 DE      2011   5800
5 DE      2012   6000
6 DE      2013   6200
7 US      2011  15000
8 US      2012  14000
9 US      2013  13000
```

The `pivot_longer()` function

```r
pivot_longer(data = cases,
             cols = "2011":"2013",
             names_to = "year",
             values_to = "cases")
```

The `pivot_longer()` function takes in a few arguments:

- `data` is the name of the data.frame or tibble that we will pivot
- `cols` are the names of the columns that will be pivoted. In this case, we want the columns named “2011” through “2013”. With `tidyr`, you can specify a range of column names with the `:` operator. Otherwise, you can provide a vector of column names
- `names_to` is a character string with what you want to call the resulting column of names. The former column names will be put into this column.
- `values_to` is a character string with what you want to call the resulting column of values. The former cell values will be put into this column.

The names are arbitrary

```r
pivot_longer(cases,
             cols = "2011":"2013",
             names_to = "when it happened",
             values_to = "how many")
```

```
# A tibble: 9 × 3
  country `when it happened` `how many`
  <chr>   <chr>                   <dbl>
1 FR      2011                     7000
2 FR      2012                     6900
3 FR      2013                     7000
4 DE      2011                     5800
5 DE      2012                     6000
6 DE      2013                     6200
7 US      2011                    15000
8 US      2012                    14000
9 US      2013                    13000
```

What happens if?

```
# A tibble: 3 × 4
  country `2011` `2012` `2013`
  <chr>    <dbl>  <dbl>  <dbl>
1 FR        7000   6900   7000
2 DE        5800   6000   6200
3 US       15000  14000  13000
```

What happens if the columns I pivot are only “2012” and “2013”?

```r
pivot_longer(cases,
             cols = "2012":"2013",
             names_to = "year",
             values_to = "cases")
```

Answer

What happens if the columns I pivot are only “2012” and “2013”?

The columns that are not pivoted are duplicated for the new rows created.

```r
pivot_longer(cases,
             cols = "2012":"2013",
             names_to = "year",
             values_to = "cases")
```

```
# A tibble: 6 × 4
  country `2011` year  cases
  <chr>    <dbl> <chr> <dbl>
1 FR        7000 2012   6900
2 FR        7000 2013   7000
3 DE        5800 2012   6000
4 DE        5800 2013   6200
5 US       15000 2012  14000
6 US       15000 2013  13000
```

What happens if?

```
# A tibble: 3 × 4
  country `2011` `2012` `2013`
  <chr>    <dbl>  <dbl>  <dbl>
1 FR        7000   6900   7000
2 DE        5800   6000   6200
3 US       15000  14000  13000
```

What happens if I include “country” in the columns I pivot?

```r
pivot_longer(cases,
             cols = "country":"2013",
             names_to = "name",
             values_to = "value")
```

Answer

What happens if I include “country” in the columns I pivot?

```r
pivot_longer(cases,
             cols = "country":"2013",
             names_to = "name",
             values_to = "value")
```

```
Error in `pivot_longer()`:
! Can't combine `country` <character> and `2011` <double>.
```

Answer

What happens if I include “country” in the columns I pivot? (I’ve converted everything to character.)

```r
cases2 <- data.frame(lapply(cases[,1:4], as.character))
names(cases2) <- c("country", "2011", "2012", "2013")
pivot_longer(cases2,
             cols = "country":"2013",
             names_to = "name",
             values_to = "value")
```

```
# A tibble: 12 × 2
   name    value
   <chr>   <chr>
 1 country FR
 2 2011    7000
 3 2012    6900
 4 2013    7000
 5 country DE
 6 2011    5800
 7 2012    6000
 8 2013    6200
 9 country US
10 2011    15000
```

```
11 2012     14000
12 2013     13000
```

The pollution table

```r
pollution
```

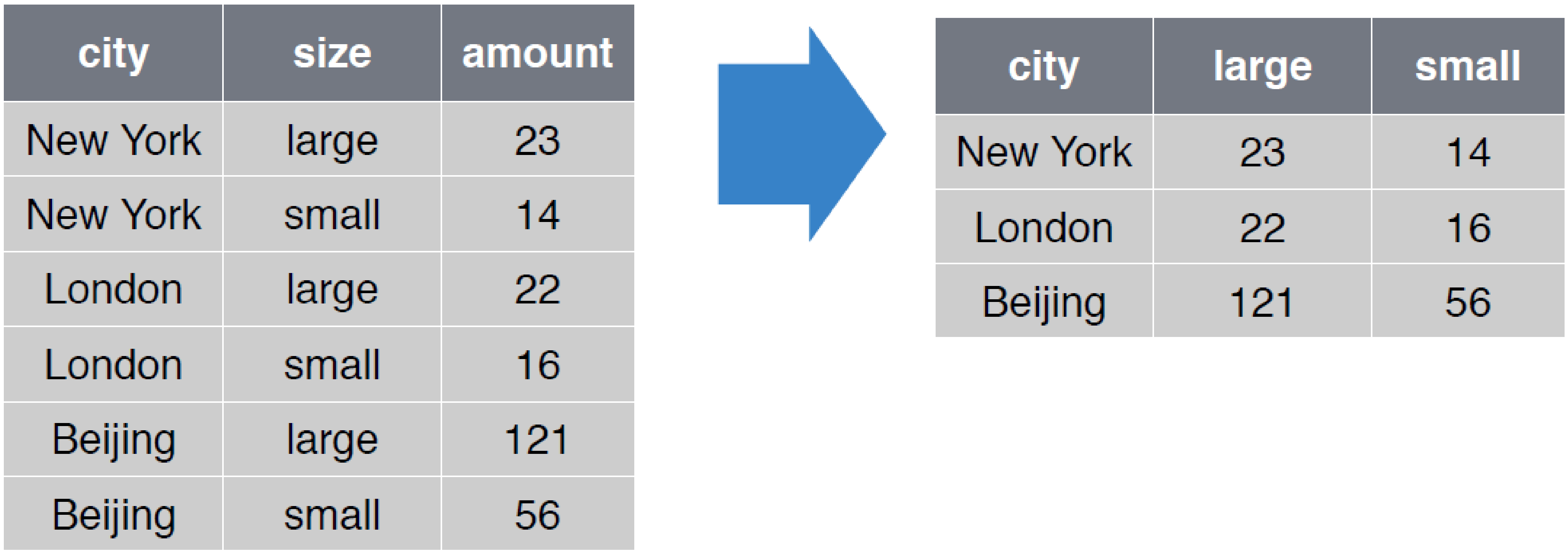
```
# A tibble: 6 × 3
  city     size  amount
  <chr>    <chr>  <dbl>
1 New York large     23
2 New York small     14
3 London   large     22
4 London   small     16
5 Beijing  large    121
6 Beijing  small     56
```

Getting things tidy

The variables are: city, large particle amount, small particle amount

If we want to make this tidy, what will be the dimensions of the resulting data?

Result: a 3 x 3 tibble



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

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

pivot_wider()

To achieve the desired result, we use the function pivot_wider() because we want the resulting dataset to be wider than the original data. (older versions called this spread())

```
pivot_wider(pollution,
             names_from = "size",
             values_from = "amount")
```

```
# A tibble: 3 × 3
  city      large small
  <chr>     <dbl> <dbl>
1 New York     23    14
2 London       22    16
3 Beijing     121    56
```

pivot_wider()

```r
pivot_wider(pollution,
             names_from = "size",
             values_from = "amount")
```

The pivot_wider() function takes in a few arguments:

- data is the name of the data.frame or tibble that we will pivot
- names_from is the name of the column that has the names that will become column headers
- values_from is the name of the column that has the values

`pivot_wider()` is sensitive to spelling differences

What will happen?

```r
pollution2 <- pollution
pollution2[1,1] <- "NYC"
pollution2
```

```
# A tibble: 6 × 3
  city     size  amount
  <chr>    <chr>  <dbl>
1 NYC      large     23
2 New York small     14
3 London   large     22
4 London   small     16
5 Beijing  large    121
6 Beijing  small     56
```

```r
pivot_wider(pollution2,
             names_from = "size",
             values_from = "amount")
```

Result

```r
pivot_wider(pollution2,
             names_from = "size",
             values_from = "amount")
```

```
# A tibble: 4 × 3
  city     large small
  <chr>    <dbl> <dbl>
1 NYC         23    NA
2 New York    NA    14
3 London      22    16
4 Beijing    121    56
```

Result

Sometimes you truly do have a scenario where you want to pivot wider and some entries do not exist. If you don’t want NAs to show, you can specify a fill value.

```r
pivot_wider(pollution2,
             names_from = "size",
             values_from = "amount",
             values_fill = 0)
```

```
# A tibble: 4 × 3
  city     large small
  <chr>    <dbl> <dbl>
1 NYC         23     0
2 New York     0    14
3 London      22    16
4 Beijing    121    56
```

Another example

What will happen?

```
pollution2 <- pollution
pollution2[1,2] <- "LARGE"
pollution2
```

```
# A tibble: 6 × 3
  city     size  amount
  <chr>    <chr>  <dbl>
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
```

```
pivot_wider(pollution2,
            names_from = "size",
            values_from = "amount")
```

Result

```r
pivot_wider(pollution2,
             names_from = "size",
             values_from = "amount")
```

```
# A tibble: 3 × 4
  city     LARGE small large
  <chr>    <dbl> <dbl> <dbl>
1 New York    23    14    NA
2 London      NA    16    22
3 Beijing     NA    56   121
```

pivot_longer() and pivot_wider() are inverse operations

```r
pollution
```

```
# A tibble: 6 × 3
  city     size  amount
  <chr>    <chr>  <dbl>
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
```

```r
w <- pivot_wider(pollution, names_from = "size", values_from = "amount")
w
```

```
# A tibble: 3 × 3
  city     large small
  <chr>    <dbl> <dbl>
1 New York    23    14
2 London      22    16
3 Beijing    121    56
```

pivot_longer() and pivot_wider() are inverse operations

```r
w
```

```
# A tibble: 3 × 3
  city     large small
  <chr>    <dbl> <dbl>
1 New York    23    14
2 London      22    16
3 Beijing    121    56
```

```r
pivot_longer(w, cols = "large":"small", names_to = "size", values_to = "amount")
```

```
# A tibble: 6 × 3
  city     size  amount
  <chr>    <chr>  <dbl>
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
```

pivot_longer() and pivot_wider() are inverse operations

```
cases
```

```
# A tibble: 3 × 4
  country `2011` `2012` `2013`
  <chr>    <dbl>  <dbl>  <dbl>
1 FR        7000   6900   7000
2 DE        5800   6000   6200
3 US       15000  14000  13000
```

```
l <- pivot_longer(cases, cols = "2011":"2013", names_to = "year", values_to = "count")
l
```

```
# A tibble: 9 × 3
  country year  count
  <chr>   <chr> <dbl>
1 FR      2011   7000
2 FR      2012   6900
3 FR      2013   7000
4 DE      2011   5800
5 DE      2012   6000
6 DE      2013   6200
7 US      2011  15000
8 US      2012  14000
9 US      2013  13000
```

`pivot_longer()` and `pivot_wider()` are inverse operations

```r
l
```

```
# A tibble: 9 × 3
  country year  count
  <chr>   <chr> <dbl>
1 FR      2011   7000
2 FR      2012   6900
3 FR      2013   7000
4 DE      2011   5800
5 DE      2012   6000
6 DE      2013   6200
7 US      2011  15000
8 US      2012  14000
9 US      2013  13000
```

```r
pivot_wider(l, names_from = "year", values_from = "count")
```

```
# A tibble: 3 × 4
  country `2011` `2012` `2013`
  <chr>    <dbl>  <dbl>  <dbl>
1 FR        7000   6900   7000
2 DE        5800   6000   6200
3 US       15000  14000  13000
```