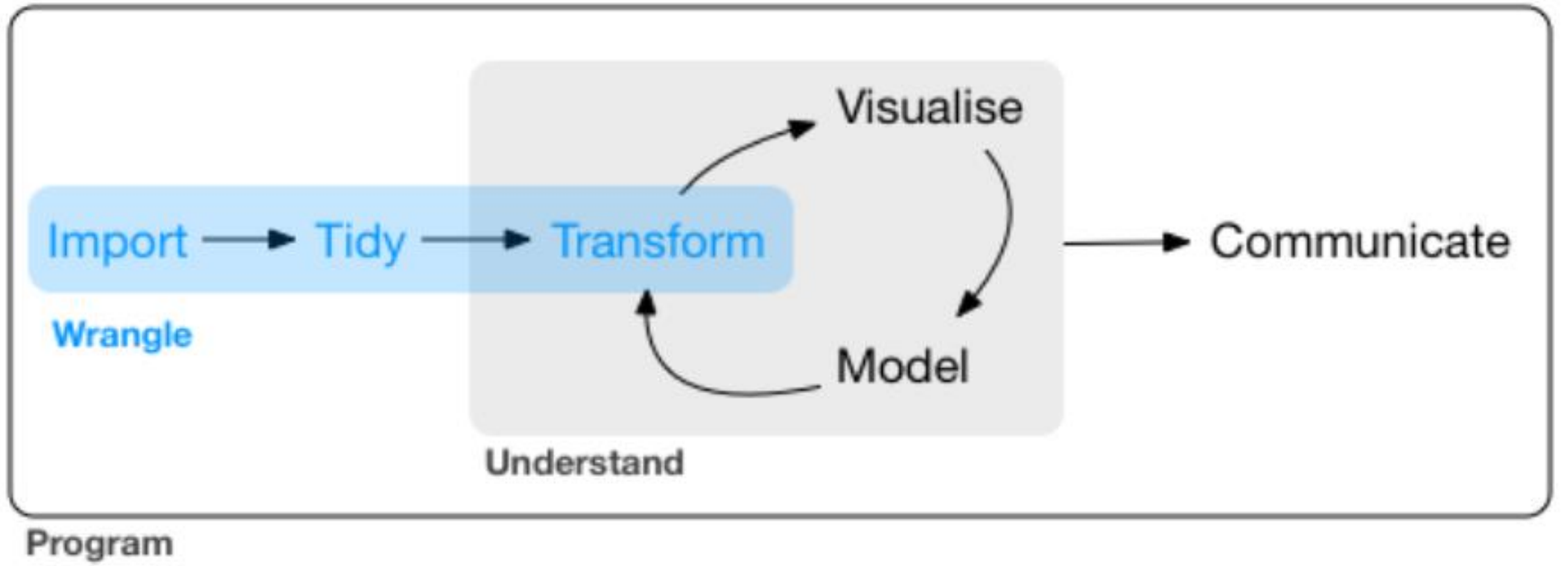
Data Wrangling
with tidyr & dplyr

2020/03/24

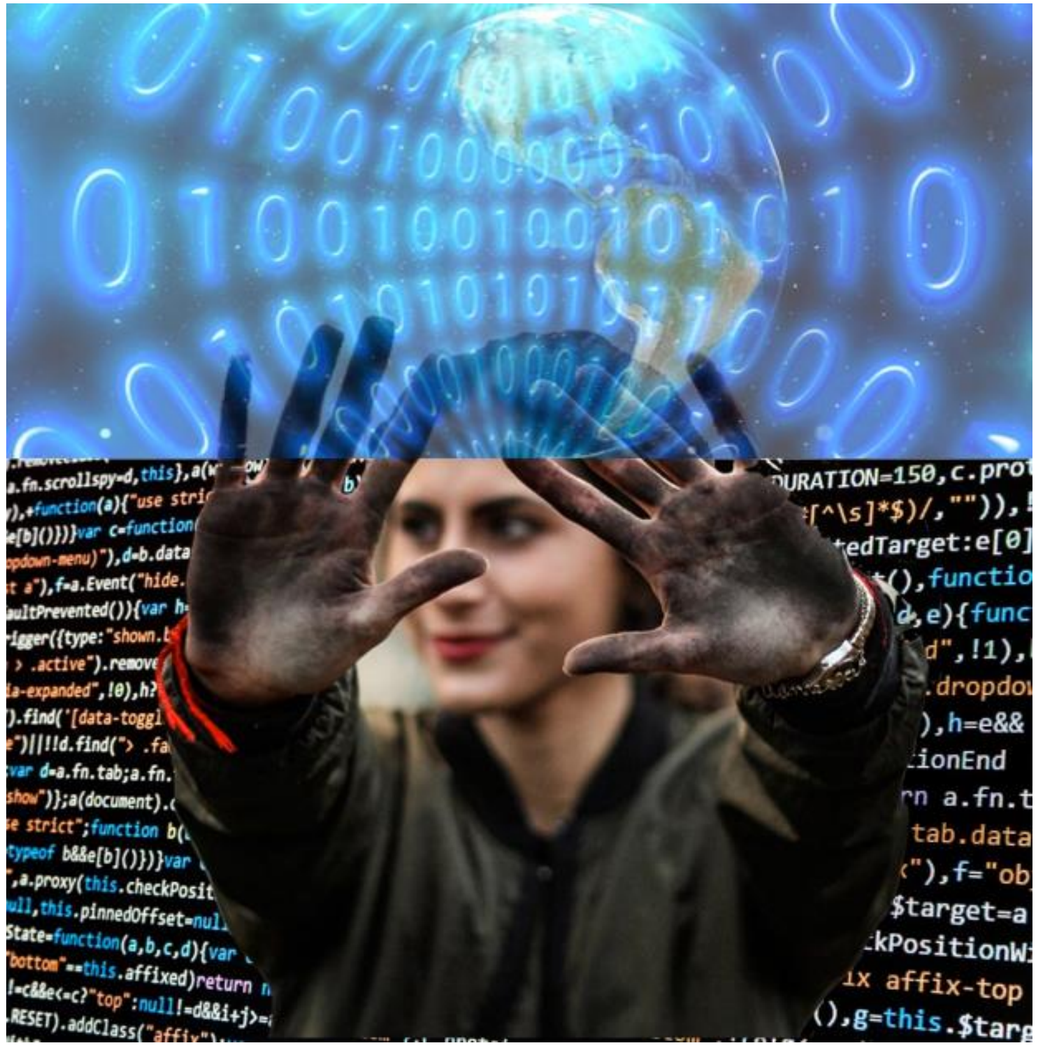
Overview



Reference

- R for Data Science: http://r4ds.had.co.nz/
- Data wrangling with R: http://goo.gl/cJJY47

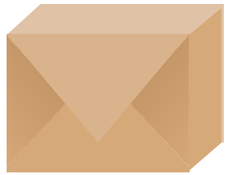
Getting your hands dirty with data!!



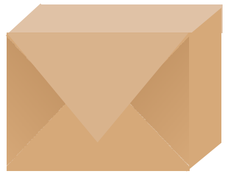
http://www.datascience.university/getting-your-hands-dirty-with-data-science/

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

Two packages to help you work with the structure of data.



tidyr



Dplyr

tidyverse

Data Wrangling with dplyr and tidyr Cheat Sheet

RStudio

Tidy Data - A foundation for wrangling in R

In a tidy data set:

Each **variable** is saved in its own **column**

&

Each **observation** is saved in its own **row**

Tidy data complements R's **vectorized operations**. R will automatically preserve observations as you manipulate variables. No other format works as intuitively with R.

Syntax - Helpful conventions for wrangling

dplyr::**tbl_df(iris)**

Converts data to tbl class. tbl's are easier to examine than data frames. R displays only the data that fits onscreen:

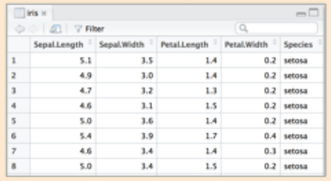
```
Source: local data frame [150 x 5]

   Sepal.Length Sepal.Width Petal.Length
1           5.1         3.5          1.4
2           4.9         3.0          1.4
3           4.7         3.2          1.3
4           4.6         3.1          1.5
5           5.0         3.6          1.4
..          ...         ...          ...
Variables not shown: Petal.Width (dbl),
  Species (fctr)
```

dplyr::**glimpse(iris)**

Information dense summary of tbl data.

utils::**View(iris)**

View data set in spreadsheet-like display (note capital V).



	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa

dplyr::**%>%**

Passes object on left hand side as first argument (or . argument) of function on righthand side.

```
x %>% f(y)  is the same as  f(x, y)
y %>% f(x, ., z)  is the same as  f(x, y, z )
```

"Piping" with %>% makes code more readable, e.g.

```
iris %>%
  group_by(Species) %>%
  summarise(avg = mean(Sepal.Width)) %>%
  arrange(avg)
```

Reshaping Data - Change the layout of a data set

tidyr::**gather(cases, "year", "n", 2:4)**

Gather columns into rows.

tidyr::**spread(pollution, size, amount)**

Spread rows into columns.

tidyr::**separate(storms, date, c("y", "m", "d"))**

Separate one column into several.

tidyr::**unite(data, col, ..., sep)**

Unite several columns into one.

dplyr::**data_frame(a = 1:3, b = 4:6)**

Combine vectors into data frame (optimized).

dplyr::**arrange(mtcars, mpg)**

Order rows by values of a column (low to high).

dplyr::**arrange(.mtcars, desc(mpg))**

Order rows by values of a column (high to low).

dplyr::**rename(tb, y = year)**

Rename the columns of a data frame.

Subset Observations (Rows)

dplyr::**filter(iris, Sepal.Length > 7)**

Extract rows that meet logical criteria.

dplyr::**distinct(iris)**

Remove duplicate rows.

dplyr::**sample_frac(iris, 0.5, replace = TRUE)**

Randomly select fraction of rows.

dplyr::**sample_n(iris, 10, replace = TRUE)**

Randomly select n rows.

dplyr::**slice(iris, 10:15)**

Select rows by position.

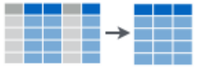
dplyr::**top_n(storms, 2, date)**

Select and order top n entries (by group if grouped data).

Logic in R - ?Comparison, ?base::Logic

<	Less than	!=	Not equal to
>	Greater than	%in%	Group membership
==	Equal to	is.na	Is NA
<=	Less than or equal to	!is.na	Is not NA
>=	Greater than or equal to	&, |, !, xor, any, all	Boolean operators

Subset Variables (Columns)



dplyr::**select(iris, Sepal.Width, Petal.Length, Species)**

Select columns by name or helper function.

Helper functions for select - ?select

select(iris, contains("."))
Select columns whose name contains a character string.

select(iris, ends_with("Length"))
Select columns whose name ends with a character string.

select(iris, everything())
Select every column.

select(iris, matches(".t."))
Select columns whose name matches a regular expression.

select(iris, num_range("x", 1:5))
Select columns named x1, x2, x3, x4, x5.

select(iris, one_of(c("Species", "Genus")))
Select columns whose names are in a group of names.

select(iris, starts_with("Sepal"))
Select columns whose name starts with a character string.

select(iris, Sepal.Length:Petal.Width)
Select all columns between Sepal.Length and Petal.Width (inclusive).

select(iris, -Species)
Select all columns except Species.

devtools::install_github("rstudio/EDAWR") for data sets

Learn more with browseVignettes(package = c("dplyr", "tidyr")) • dplyr 0.4.0 • tidyr 0.2.0 • Updated: 1/15

Outline

Toy Example

- Tidy data? View() & glimpse()
- tidyr: reshape data
  - gather() columns into rows
  - spread() rows into columns
  - unite() several columns into one
  - separate() one column into several
- dplyr: transform data
  - filter() observations in rows
  - select() variables in columns
  - arrange() observations in rows
  - mutate() from old var. to new one
  - summarise() the data by column
  - group_by() certain categories

Real Data

- Weather Forecasting Data

1. Tidy data?
   - Variables are stored in both rows and columns
   - Duplicate observations
2. tidyr & dplyr example
   - clean.data <- raw.data %>%
     mutate() %>%
     select() %>%
     distinct() %>%
     spread() %>%
     filter()
   - clean.data %>%
     group_by() %>%
     summarise()

Outline

Toy Example

- Tidy data? View() & glimpse()
- tidyr: reshape data
  - gather() columns into rows
  - spread() rows into columns
  - unite() several columns into one
  - separate() one column into several
- dplyr: transform data
  - filter() observations in rows
  - select() variables in columns
  - arrange() observations in rows
  - mutate() from old var. to new one
  - summarise() the data by column
  - group_by() certain categories

Real Data

- Weather Forecasting Data

1. Tidy data?
   - Variables are stored in both rows and columns
   - Duplicate observations
2. tidyr & dplyr example
   - clean.data <- raw.data %>%
     mutate() %>%
     select() %>%
     distinct() %>%
     spread() %>%
     filter()
   - clean.data %>%
     group_by() %>%
     summarise()

Outline

Toy Example

- **Tidy data? View() & glimpse()**
- tidyr: reshape data
  - gather() columns into rows
  - spread() rows into columns
  - unite() several columns into one
  - separate() one column into several
- dplyr: transform data
  - filter() observations in rows
  - select() variables in columns
  - arrange() observations in rows
  - mutate() from old var. to new one
  - summarise() the data by column
  - group_by() certain categories

Real Data

- Weather Forecasting Data
  1. Tidy data?
     - Variables are stored in both rows and columns
     - Duplicate observations
  2. tidyr & dplyr example
     - clean.data <- raw.data %>%
       mutate() %>%
       select() %>%
       distinct() %>%
       spread() %>%
       filter()
     - clean.data %>%
       group_by() %>%
       summarise()

Outline

Toy Example

- Tidy data? View() & glimpse()
- **tidyr: reshape data**
  - **gather() columns into rows**
  - **spread() rows into columns**
  - **unite() several columns into one**
  - **separate() one column into several**
- dplyr: transform data
  - filter() observations in rows
  - select() variables in columns
  - arrange() observations in rows
  - mutate() from old var. to new one
  - summarise() the data by column
  - group_by() certain categories

Real Data

- Weather Forecasting Data

1. Tidy data?
   - Variables are stored in both rows and columns
   - Duplicate observations
2. tidyr & dplyr example
   - clean.data <- raw.data %>%
     mutate() %>%
     select() %>%
     distinct() %>%
     spread() %>%
     filter()
   - clean.data %>%
     group_by() %>%
     summarise()

Outline

Toy Example

- Tidy data? View() & glimpse()
- tidyr: reshape data
  - gather() columns into rows
  - spread() rows into columns
  - unite() several columns into one
  - separate() one column into several
- **dplyr: transform data**
  - **filter() observations in rows**
  - **select() variables in columns**
  - **arrange() observations in rows**
  - **mutate() from old var. to new one**
  - **summarise() the data by column**
  - **group_by() certain categories**

Real Data

- Weather Forecasting Data

1. Tidy data?
   - Variables are stored in both rows and columns
   - Duplicate observations
2. tidyr & dplyr example
   - clean.data <- raw.data %>%
     mutate() %>%
     select() %>%
     distinct() %>%
     spread() %>%
     filter()
   - clean.data %>%
     group_by() %>%
     summarise()

Outline

Toy Example

- Tidy data? View() & glimpse()
- tidyr: reshape data
  - gather() columns into rows
  - spread() rows into columns
  - unite() several columns into one
  - separate() one column into several
- dplyr: transform data
  - filter() observations in rows
  - select() variables in columns
  - arrange() observations in rows
  - mutate() from old var. to new one
  - summarise() the data by column
  - group_by() certain categories

Real Data

- **Weather Forecasting Data**
  1. **Tidy data?**
     - **Variables are stored in both rows and columns**
     - **Duplicate observations**
  2. **tidyr & dplyr example**
     - **clean.data <- raw.data %>%**
       mutate() %>%
       select() %>%
       distinct() %>%
       spread() %>%
       filter()
     - **clean.data %>%**
       group_by() %>%
       summarise()

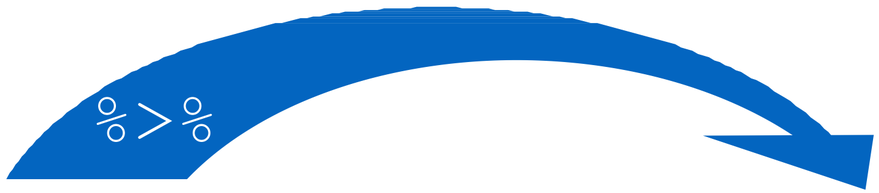
Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

The pipe operator %>%

```
library(dplyr)
select(tb, child:elderly)
tb %>% select(child:elderly)
```

%%

These do the same thing
Try it!




```
tb              select( ______, child:elderly)
```

Shortcut to type %>%

Cmd + Shift + M (Mac)

Ctrl + Shift + M (Windows)

Outline

Toy Example

- **Tidy data? View() & glimpse()**
- tidyr: reshape data
  - gather() columns into rows
  - spread() rows into columns
  - unite() several columns into one
  - separate() one column into several
- dplyr: transform data
  - filter() observations in rows
  - select() variables in columns
  - arrange() observations in rows
  - mutate() from old var. to new one
  - summarise() the data by column
  - group_by() certain categories

Real Data

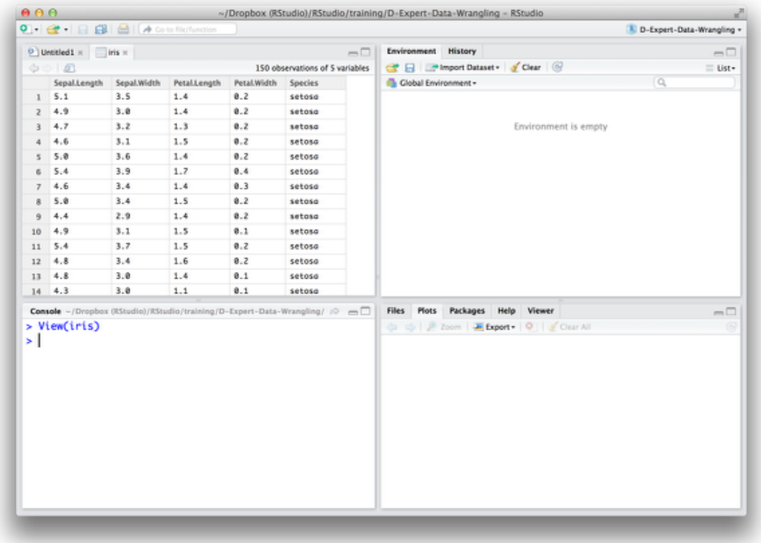
- Weather Forecasting Data
  1. Tidy data?
     - Variables are stored in both rows and columns
     - Duplicate observations
  2. tidyr & dplyr example
     - clean.data <- raw.data %>%
       mutate() %>%
       select() %>%
       distinct() %>%
       spread() %>%
       filter()
     - clean.data %>%
       group_by() %>%
       summarise()

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

View()

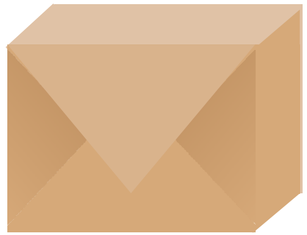
Examine any data set with the View() command (Capital V)

View(iris)
View(mtcars)
View(pressure)



Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

EDAWR



An R package with all of the data sets that we will use today.

```
# install.packages("devtools")
# evtools::install_github("rstudio/EDAWR")
library(EDAWR)
?storms                    ?pollution
?cases                     ?tb
```

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

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

cases

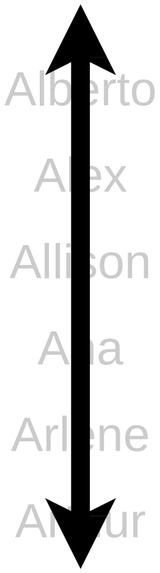
Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

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

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

storms



	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

cases

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

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

- Storm name

storms

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)

cases

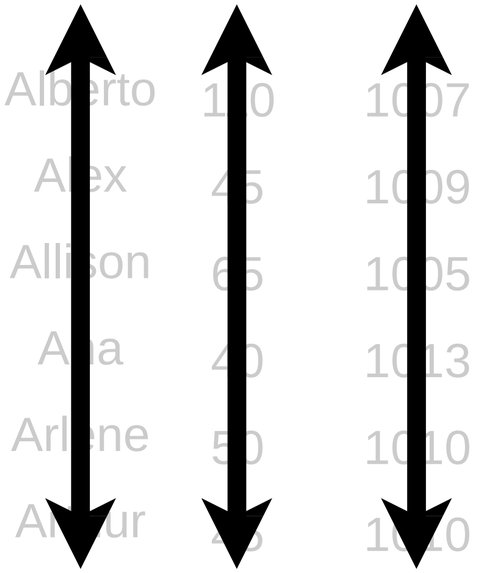
Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

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

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

storms



date
2000-08-12
1998-07-30
1995-06-04
1997-07-01
1999-06-13
1996-06-21

cases

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

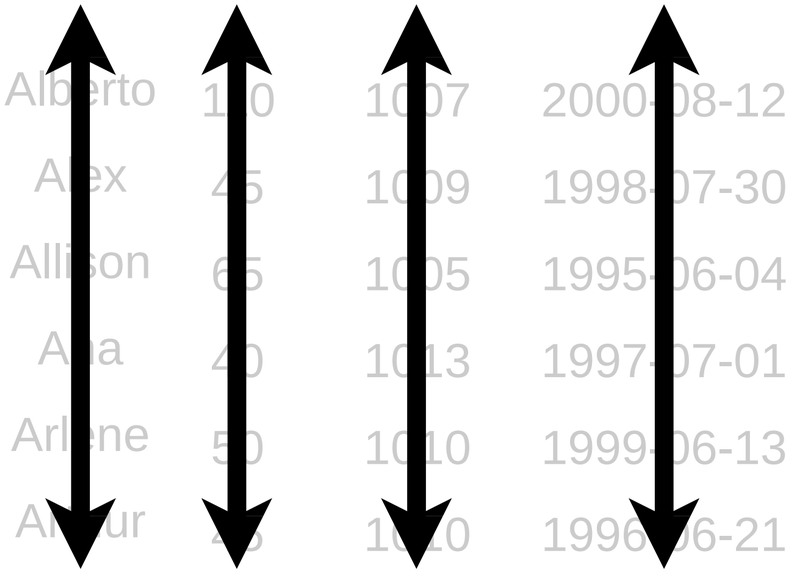
pollution

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

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

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

storms



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

cases

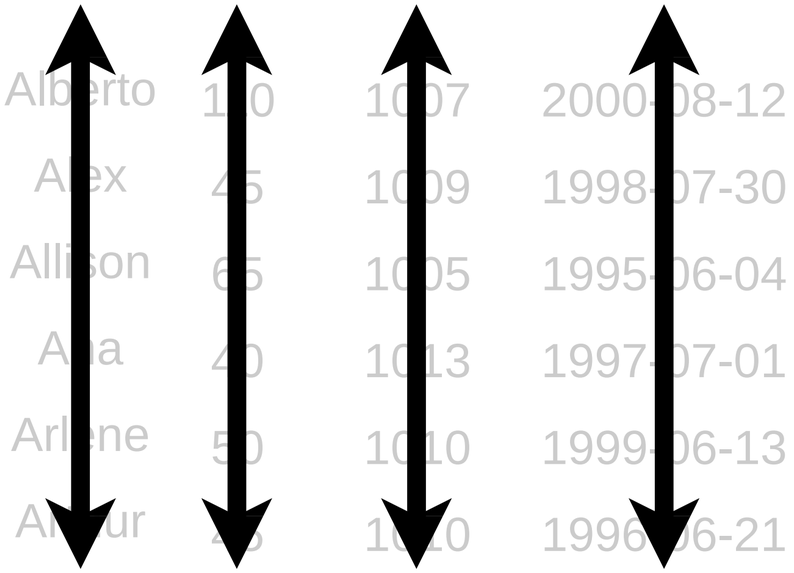
Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

pollution

city	particle size	amount (µg/m³)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

storms



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

cases

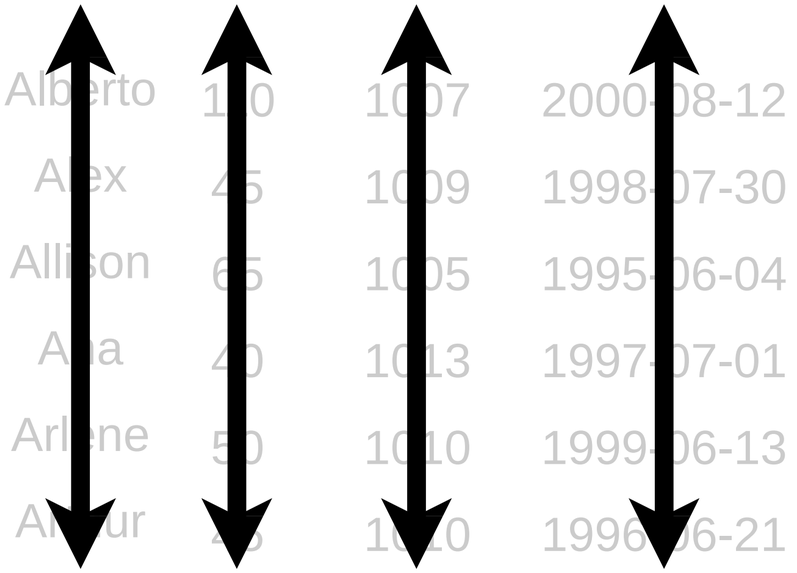
2011	2012	2013
7000	6900	7000
5800	6000	6200
15000	14000	13000

- Country

pollution

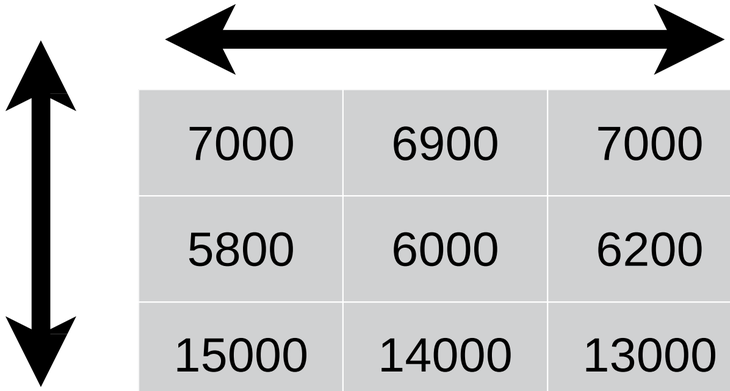
city	particle size	amount (µg/m³)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

storms



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

cases



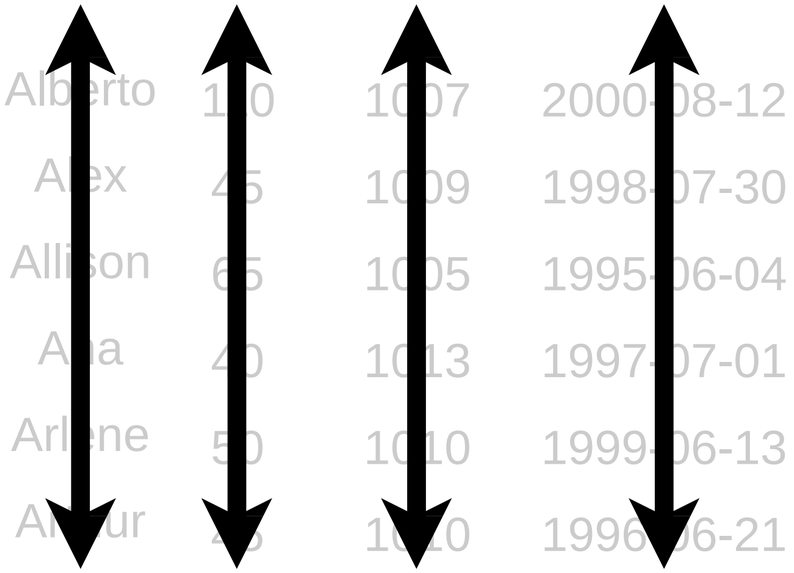
7000	6900	7000
5800	6000	6200
15000	14000	13000

- Country
- Year

pollution

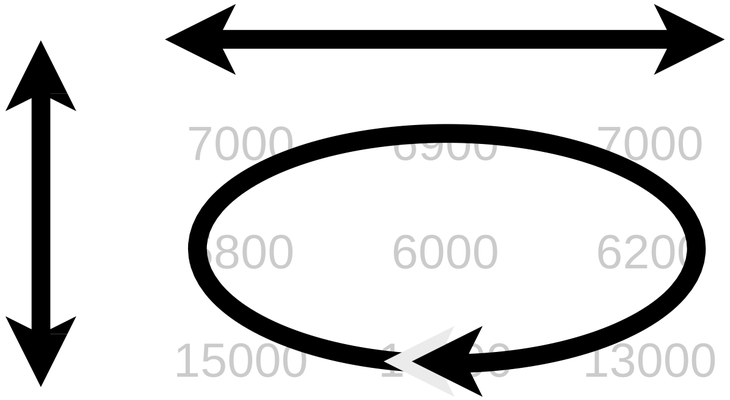
city	particle size	amount (µg/m³)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

storms



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

cases

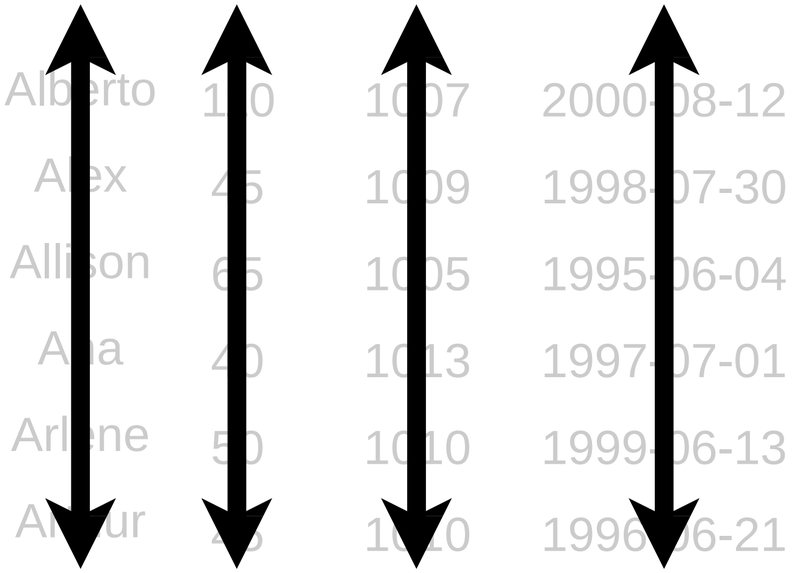


- Country
- Year
- Count

pollution

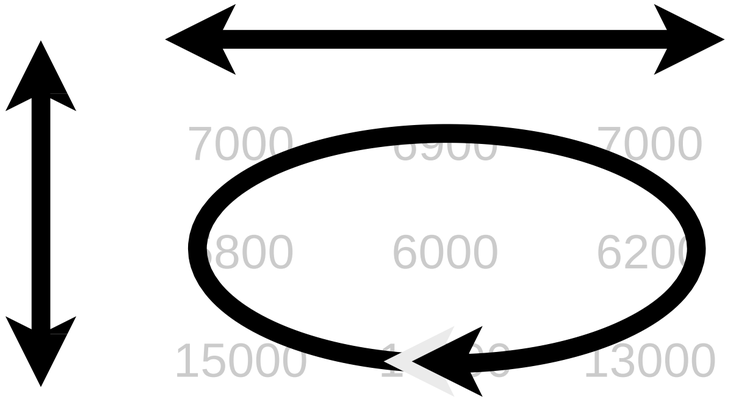
city	particle size	amount (µg/m³)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

storms



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

cases



- Country
- Year
- Count

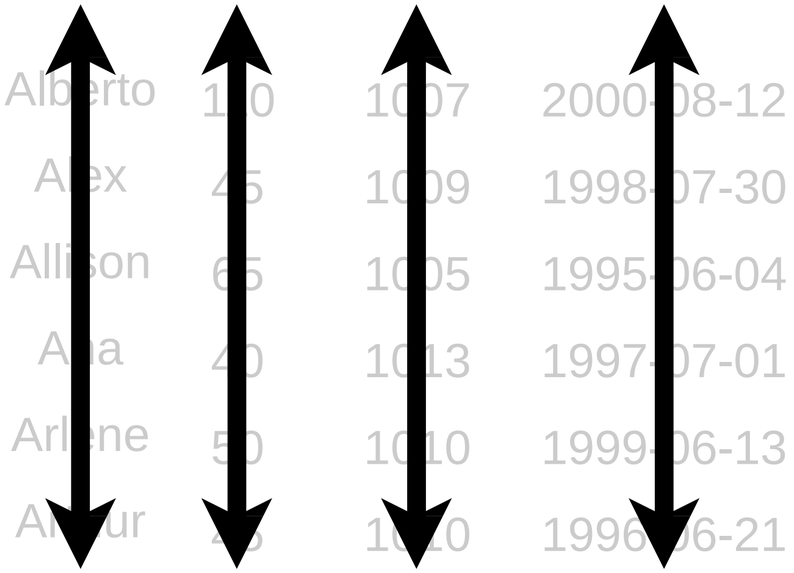
pollution

particle size	amount (µg/m³)
large	23
small	14
large	22
small	16
large	121
small	56

- City

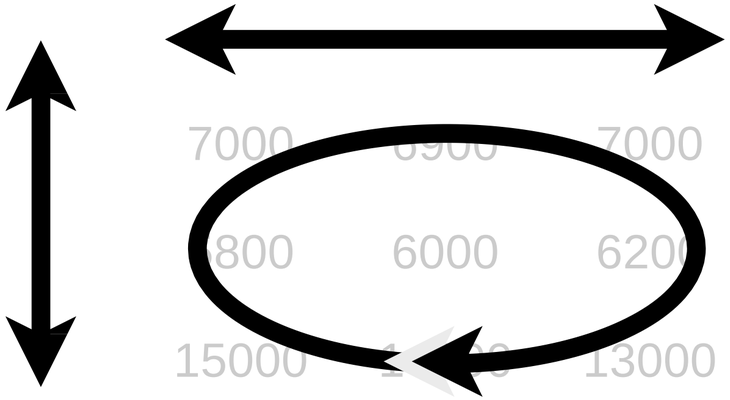
Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

storms



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

cases



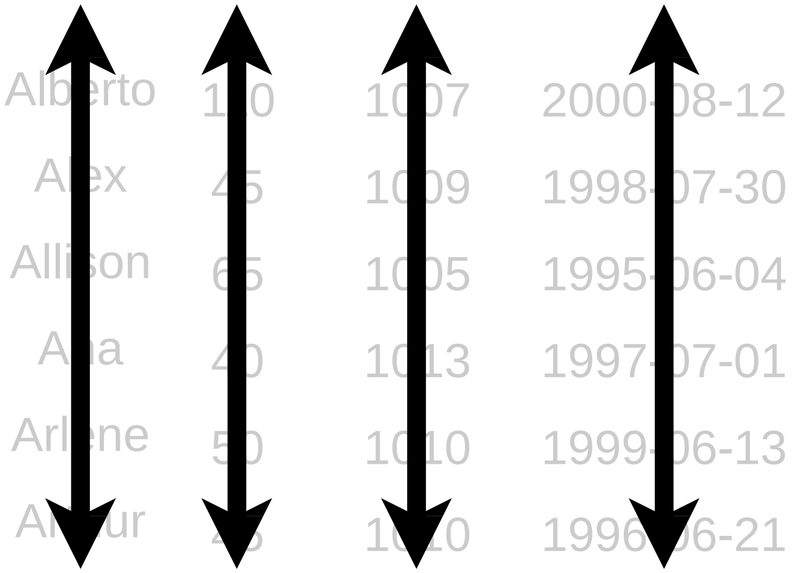
- Country
- Year
- Count

pollution

particle size	amount (µg/m³)
large	23
small	14
large	22
small	16
large	121
small	56

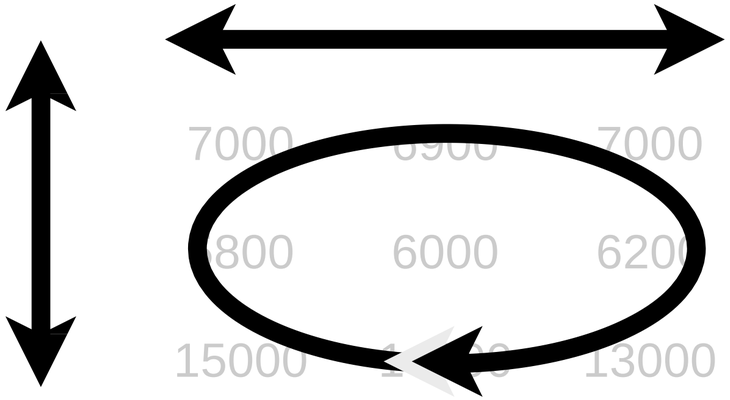
- City
- Amount of large particles

storms



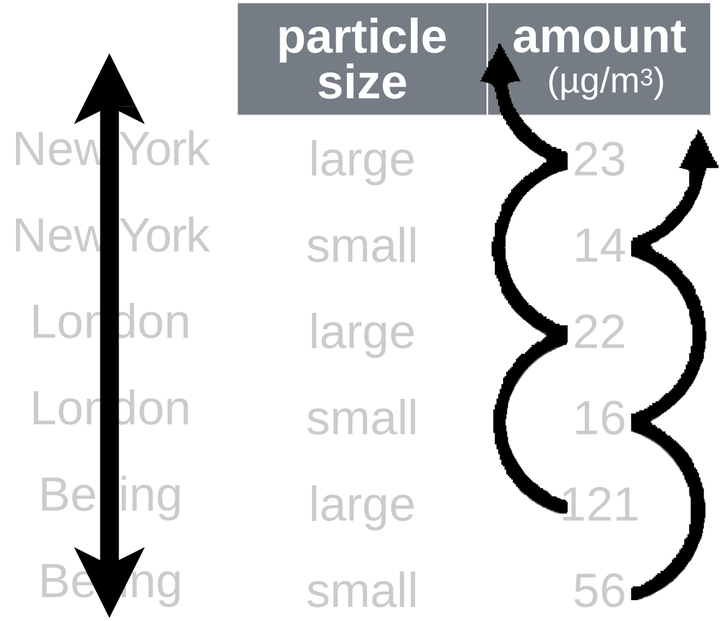
- Storm name
- Wind Speed (mph)
- Air Pressure
- Date

cases



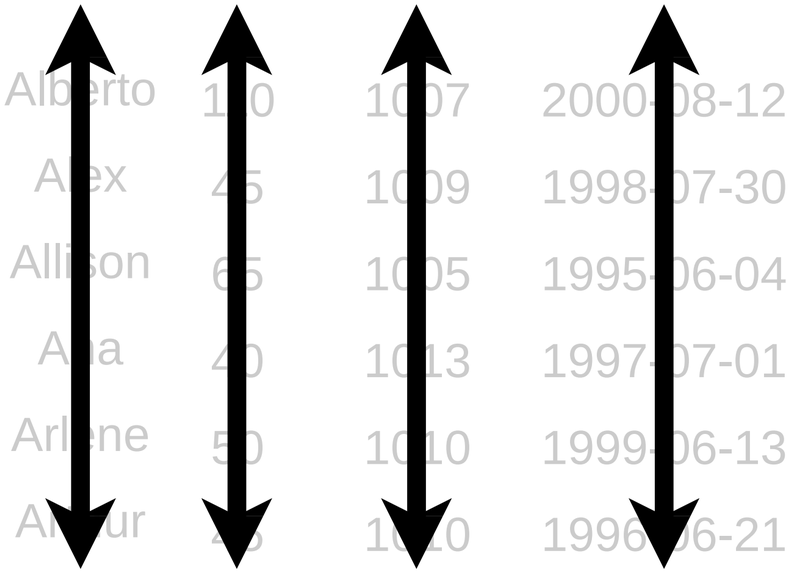
- Country
- Year
- Count

pollution



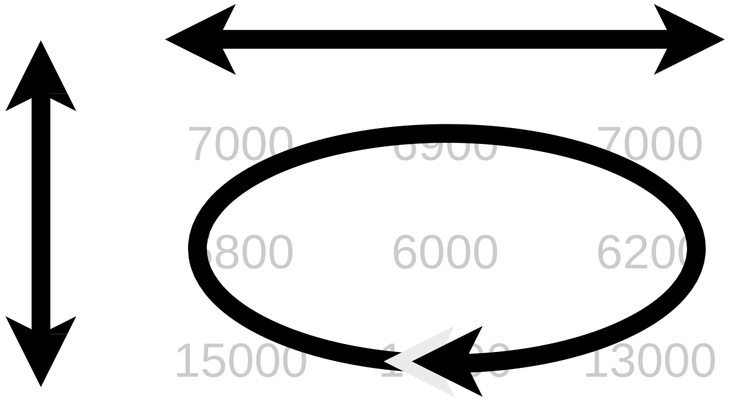
- City
- Amount of large particles
- Amount of small particles

storms



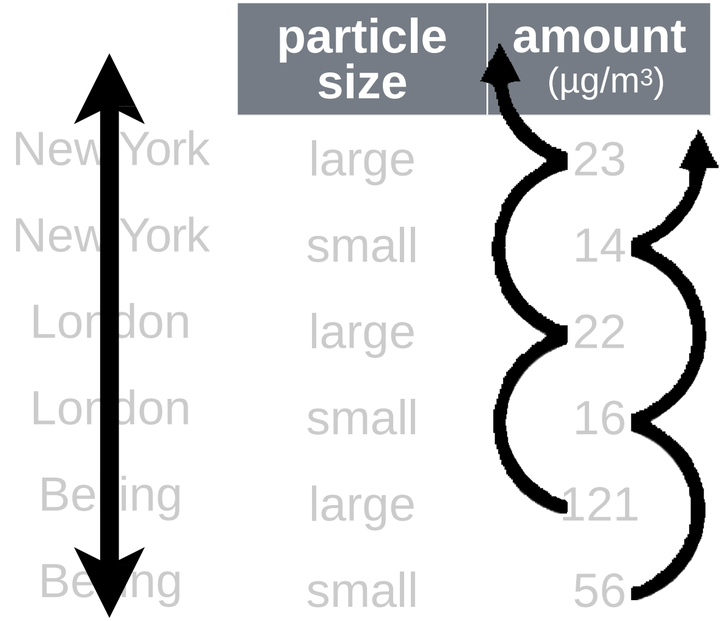
storms$storm
storms$wind
storms$pressure
storms$date

cases



cases$country
names(cases)[-1]
unlist(cases[1:3, 2:4])

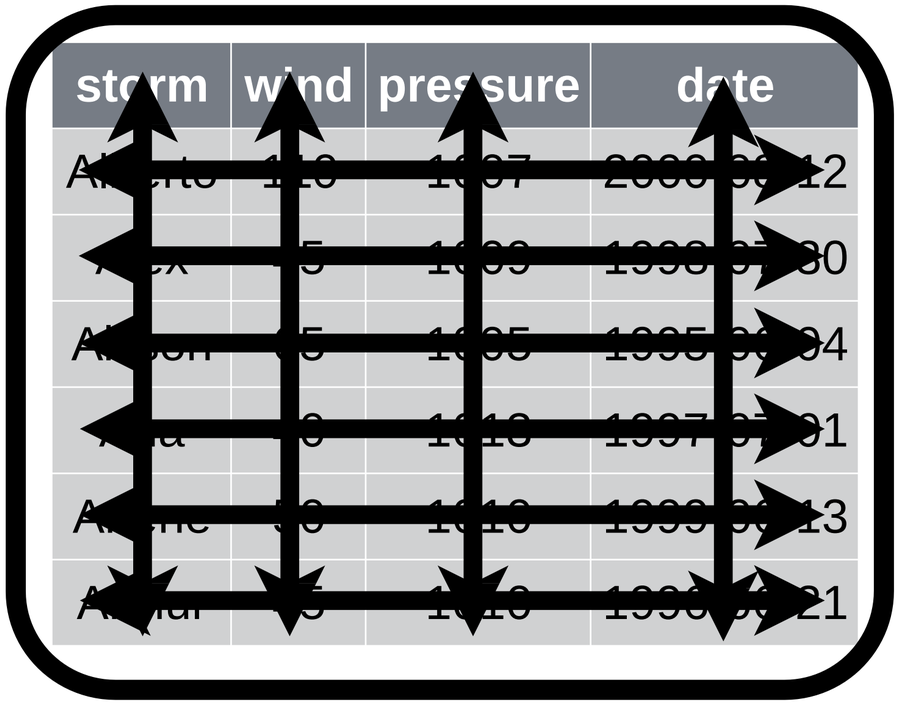
pollution



pollution$city[1,3,5]
pollution$amount[1,3,5]
pollution$amount[2,4,6]

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

1 Each **variable** is saved in its own **column**.

2 Each **observation** is saved in its own **row**.

3 Each "type" of observation stored in a **single table** (here, storms).

Outline

Toy Example

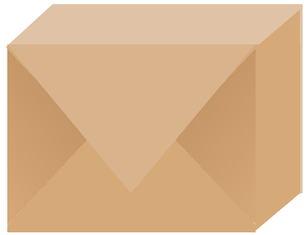
- Tidy data? View() & glimpse()
- **tidyr: reshape data**
  - **gather() columns into rows**
  - **spread() rows into columns**
  - **unite() several columns into one**
  - **separate() one column into several**
- dplyr: transform data
  - filter() observations in rows
  - select() variables in columns
  - arrange() observations in rows
  - mutate() from old var. to new one
  - summarise() the data by column
  - group_by() certain categories

Real Data

- Weather Forecasting Data

1. Tidy data?
   - Variables are stored in both rows and columns
   - Duplicate observations
2. tidyr & dplyr example
   - clean.data <- raw.data %>%
     mutate() %>%
     select() %>%
     distinct() %>%
     spread() %>%
     filter()
   - clean.data %>%
     group_by() %>%
     summarise()

tidyr



A package that reshapes the layout of tables.

Two main functions: **gather()** and **spread()**

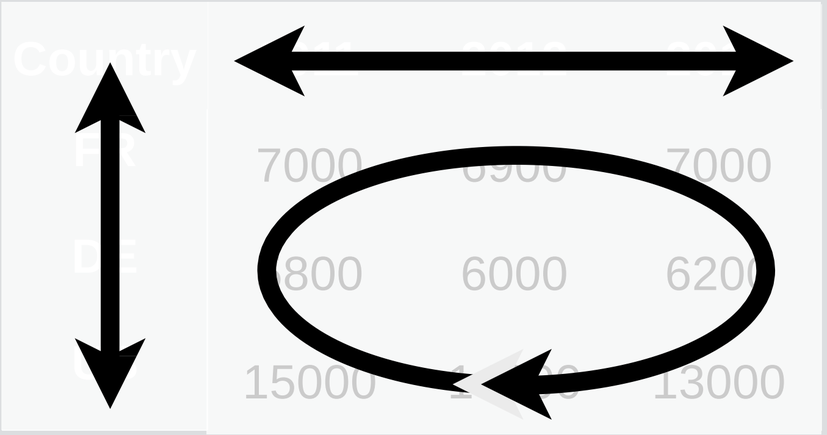
```
# install.packages("tidyr")
library(tidyr)
?gather
?spread
```

Your Turn

Imagine how this data would look if it were tidy with three variables: *country, year, n*

cases

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



slides at: bit.ly/wrangling-webinar

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

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

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

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

Country	Year	n

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

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

Country	Year	n
FR	2011	7000

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

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

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

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

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

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

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

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

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

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

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

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

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

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

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

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

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

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

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

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

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

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

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

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

Countr	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

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

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

gather()

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

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

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

key **value** (former cells)

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

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

gather()

Collapses multiple columns into two columns:

1. a **key** column that contains the former column names
2. a **value** column that contains the former column cells

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

gather()

Collapses multiple columns into two columns:

1. a **key** column that contains the former column names
2. a **value** column that contains the former column cells

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

data frame to reshape

gather()

Collapses multiple columns into two columns:

1. a **key** column that contains the former column names
2. a **value** column that contains the former column cells

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

data frame to reshape

name of the new key column (a character string)

gather()

Collapses multiple columns into two columns:

1. a **key** column that contains the former column names
2. a **value** column that contains the former column cells

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

data frame to reshape

name of the new key column (a character string)

name of the new value column (a character string)

gather()

Collapses multiple columns into two columns:

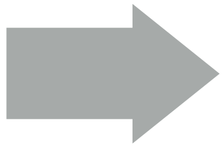
1. a **key** column that contains the former column names
2. a **value** column that contains the former column cells

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

- `cases`: data frame to reshape
- `"year"`: name of the new key column (a character string)
- `"n"`: name of the new value column (a character string)
- `2:4`: names or numeric indexes of columns to collapse

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

```
##   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
```

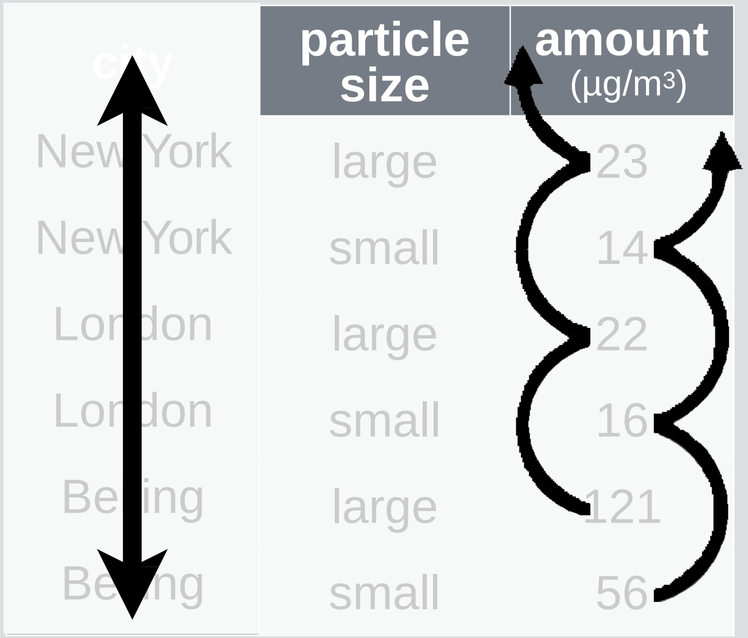
```
gather(cases, "year", "n", 2:4)
```

Your Turn

Imagine how the pollution data set would look tidy with three variables: *city, large, small*

pollution

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



slides at: bit.ly/wrangling-webinar

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

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

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

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

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

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	

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

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

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

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	

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

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

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

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

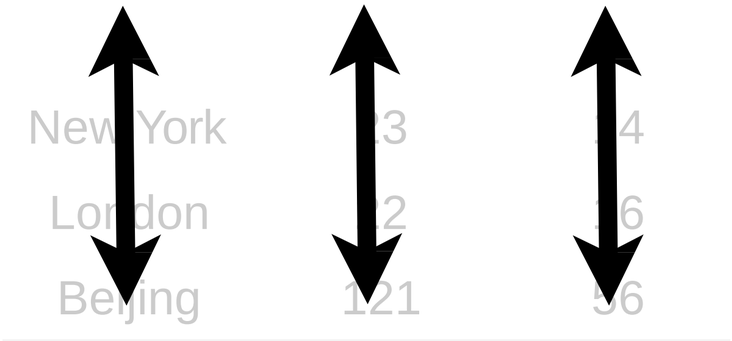
city	large	small
New York	23	14
London	22	16
Beijing	121	

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

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



Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

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

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

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

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

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

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

spread()

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

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

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

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

spread()

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

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

spread()

Generates multiple columns from two columns:

1. each unique value in the **key** column becomes a column name
2. each value in the **value** column becomes a cell in the new columns

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

spread()

Generates multiple columns from two columns:

1. each unique value in the **key** column becomes a column name
2. each value in the **value** column becomes a cell in the new columns

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

data frame to reshape

spread()

Generates multiple columns from two columns:

1. each unique value in the **key** column becomes a column name
2. each value in the **value** column becomes a cell in the new columns

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

data frame to reshape

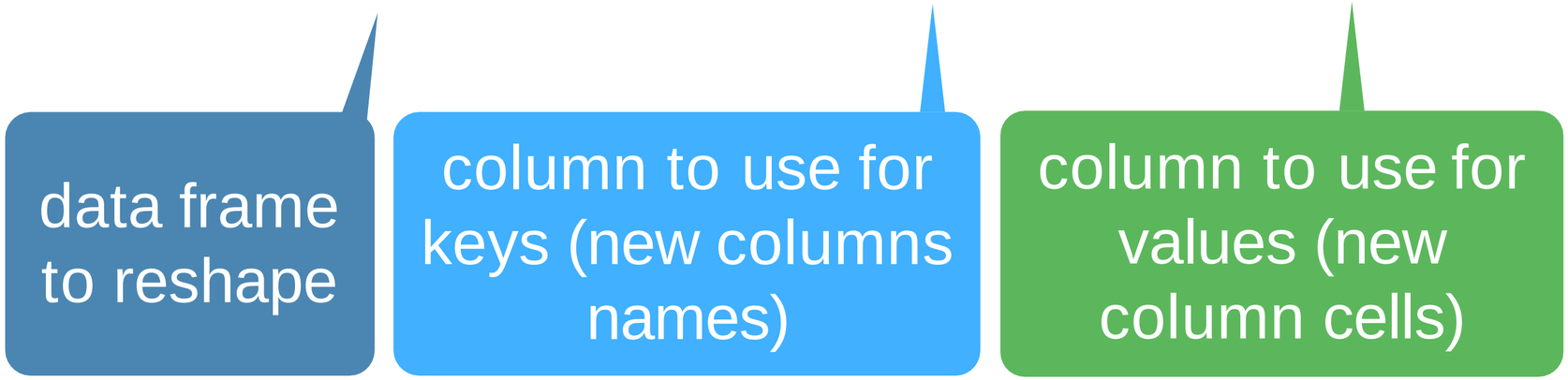
column to use for keys (new columns names)

spread()

Generates multiple columns from two columns:

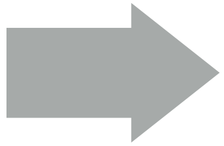
1. each unique value in the **key** column becomes a column name
2. each value in the **value** column becomes a cell in the new columns

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



Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

```
##       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
```

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

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

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

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

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

spread()

gather()

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

unite() and separate()

There are three more variables hidden in storms:

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

- Year
- Month
- Day

separate()

Separate splits a column by a character string separator.

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

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

storms2

storm	wind	pressure	year	month	day
Alberto	110	1007	2000	08	12
Alex	45	1009	1998	07	30
Allison	65	1005	1995	06	04
Ana	40	1013	1997	07	1
Arlene	50	1010	1999	06	13
Arthur	45	1010	1996	06	21

unite()

Unite unites columns into a single column.

```
unite(storms2, "date", year, month, day, sep = "-")
```

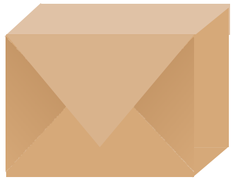
storms2

storm	wind	pressure	year	month	day
Alberto	110	1007	2000	08	12
Alex	45	1009	1998	07	30
Allison	65	1005	1995	06	04
Ana	40	1013	1997	07	1
Arlene	50	1010	1999	06	13
Arthur	45	1010	1996	06	21

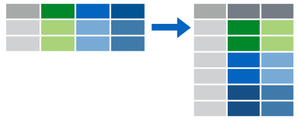
storms

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

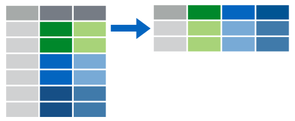
Recap: tidyr



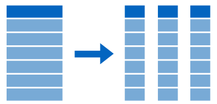
A package that reshapes the layout of data sets.



Make observations from variables with `gather()`



Make variables from observations with `spread()`



Split and merge columns with `unite()` and `separate()`

Outline

Toy Example

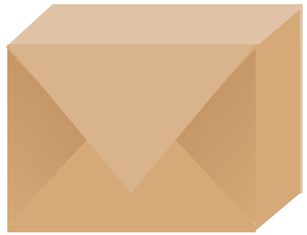
- Tidy data? View() & glimpse()
- tidyr: reshape data
  - gather() columns into rows
  - spread() rows into columns
  - unite() several columns into one
  - separate() one column into several
- **dplyr: transform data**
  - **filter() observations in rows**
  - **select() variables in columns**
  - **arrange() observations in rows**
  - **mutate() from old var. to new one**
  - **summarise() the data by column**
  - **group_by() certain categories**

Real Data

- Weather Forecasting Data

1. Tidy data?
   - Variables are stored in both rows and columns
   - Duplicate observations
2. tidyr & dplyr example
   - clean.data <- raw.data %>%
     mutate() %>%
     select() %>%
     distinct() %>%
     spread() %>%
     filter()
   - clean.data %>%
     group_by() %>%
     summarise()

dplyr



A package that helps transform tabular data.

```
# install.packages("dplyr")
library(dplyr)
```

```
?select
?filter
?arrange
?mutate
?summarise
?group_by
```

Ways to access information

1 **Extract** existing variables. **select()**

2 **Extract** existing observations. **filter()**

3 **Derive** new variables (from existing variables) **mutate()**

4 **Change** the unit of analysis **summarise()**

select()

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	pressure
Alberto	1007
Alex	1009
Allison	1005
Ana	1013
Arlene	1010
Arthur	1010

```
select(storms, storm, pressure)
```

select()

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

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

```
select(storms, -storm)
# see ?select for more
```

select()

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

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

```
select(storms, wind:date)
# see ?select for more
```

Useful select functions

* Blue functions come in dplyr

-	Select everything but
:	Select range
contains()	Select columns whose name contains a character string
ends_with()	Select columns whose name ends with a string
everything()	Select every column
matches()	Select columns whose name matches a regular expression
num_range()	Select columns named x1, x2, x3, x4, x5
one_of()	Select columns whose names are in a group of names
starts_with()	Select columns whose name starts with a character string

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

filter()

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	2000-08-12
Allison	65	1005	1995-06-04
Arlene	50	1010	1999-06-13

```
filter(storms, wind >= 50)
```

filter()

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	2000-08-12
Allison	65	1005	1995-06-04

```
filter(storms, wind >= 50,
  storm %in% c("Alberto", "Alex", "Allison"))
```

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

logical tests in R

`?Comparison`

<	Less than
>	Greater than
==	Equal to
<=	Less than or equal to
>=	Greater than or equal to
!=	Not equal to
%in%	Group membership
is.na	Is NA
!is.na	Is not NA

`?base::Logic`

&	boolean and
|	boolean or
xor	exactly or
!	not
any	any true
all	all true

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

arrange()

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

```
arrange(storms, wind)
```

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

arrange()

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

```
arrange(storms, wind)
```

arrange()

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

```
arrange(storms, desc(wind))
```

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

arrange()

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

```
arrange(storms, wind)
```

arrange()

storms

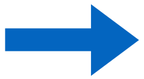
storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

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

```
arrange(storms, wind, date)
```

mutate()

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

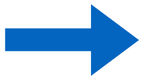


storm	wind	pressure	date	ratio
Alberto	110	1007	2000-08-12	9.15
Alex	45	1009	1998-07-30	22.42
Allison	65	1005	1995-06-04	15.46
Ana	40	1013	1997-07-01	25.32
Arlene	50	1010	1999-06-13	20.20
Arthur	45	1010	1996-06-21	22.44

```
mutate(storms, ratio = pressure / wind)
```

mutate()

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	ratio	inverse
Alberto	110	1007	2000-08-12	9.15	0.11
Alex	45	1009	1998-07-30	22.42	0.04
Allison	65	1005	1995-06-04	15.46	0.06
Ana	40	1013	1997-07-01	25.32	0.04
Arlene	50	1010	1999-06-13	20.20	0.05
Arthur	45	1010	1996-06-21	22.44	0.04

```
mutate(storms, ratio = pressure / wind, inverse = ratio^-1)
```

Useful mutate functions

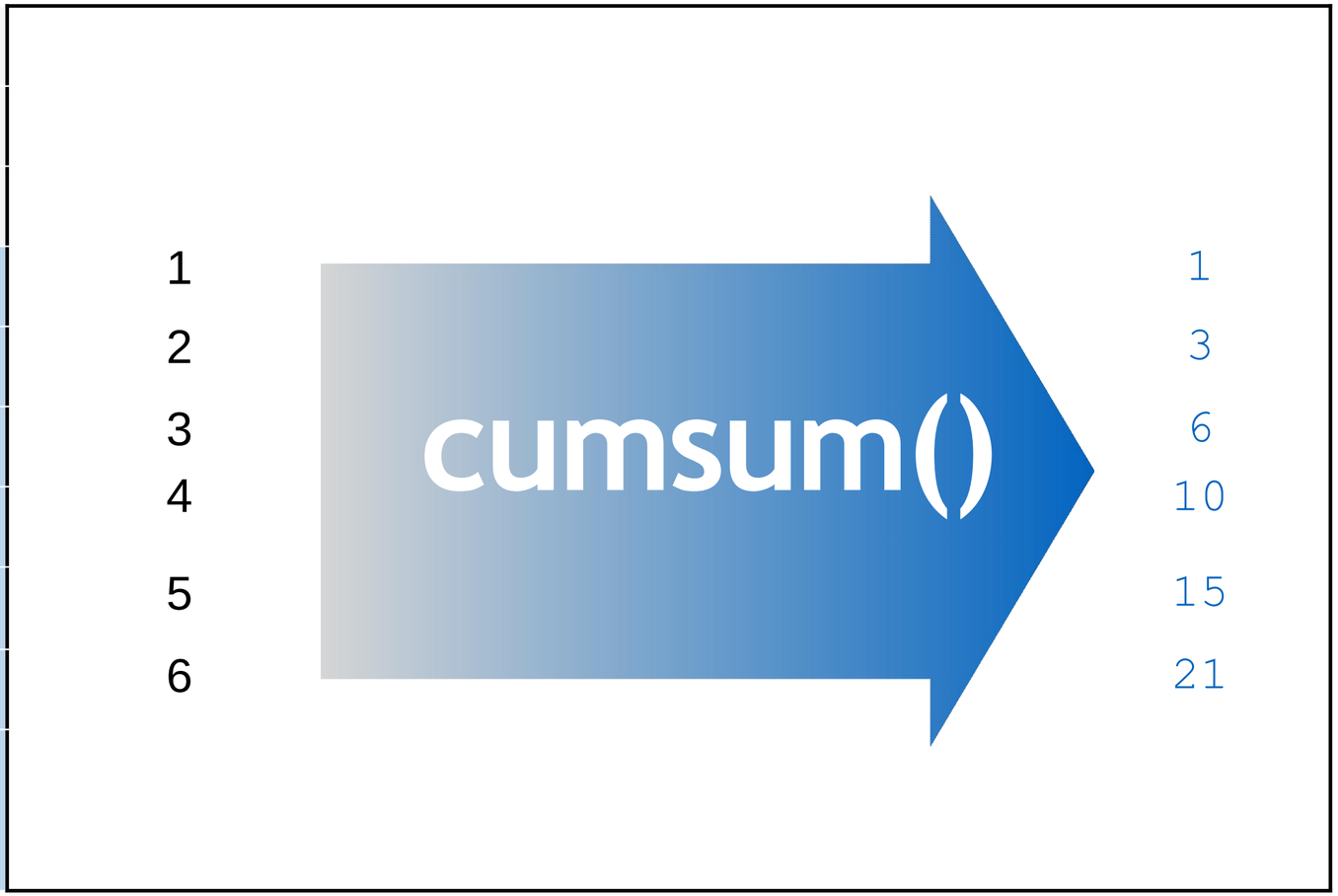
* All take a vector of values and return a vector of values
** Blue functions come in dplyr

pmin(), pmax()	Element-wise min and max
cummin(), cummax()	Cumulative min and max
cumsum(), cumprod()	Cumulative sum and product
between()	Are values between a and b?
cume_dist()	Cumulative distribution of values
cumall(), cumany()	Cumulative all and any
cummean()	Cumulative mean
lead(), lag()	Copy with values one position
ntile()	Bin vector into n buckets
dense_rank(), min_rank(), percent_rank(), row_number()	Various ranking methods

"Window" functions

* All take a vector of values and return a vector of values

pmin(), pmax()
cummin(), cummax()
cumsum(), cumprod()
between()
cume_dist()
cumall(), cumany()
cummean()
lead(), lag()
ntile()
dense_rank(), min_rank(), percent_rank(), row_number()



summarise()

city	particle size	amount (µg/m³)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

median	variance
22.5	1731.6

```
pollution %>% summarise(median = median(amount), variance = var(amount))
```

summarise()

city	particle size	amount (µg/m³)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

mean	sum	n
42	252	6

```
pollution %>% summarise(mean = mean(amount), sum = sum(amount), n = n())
```

Useful summary functions

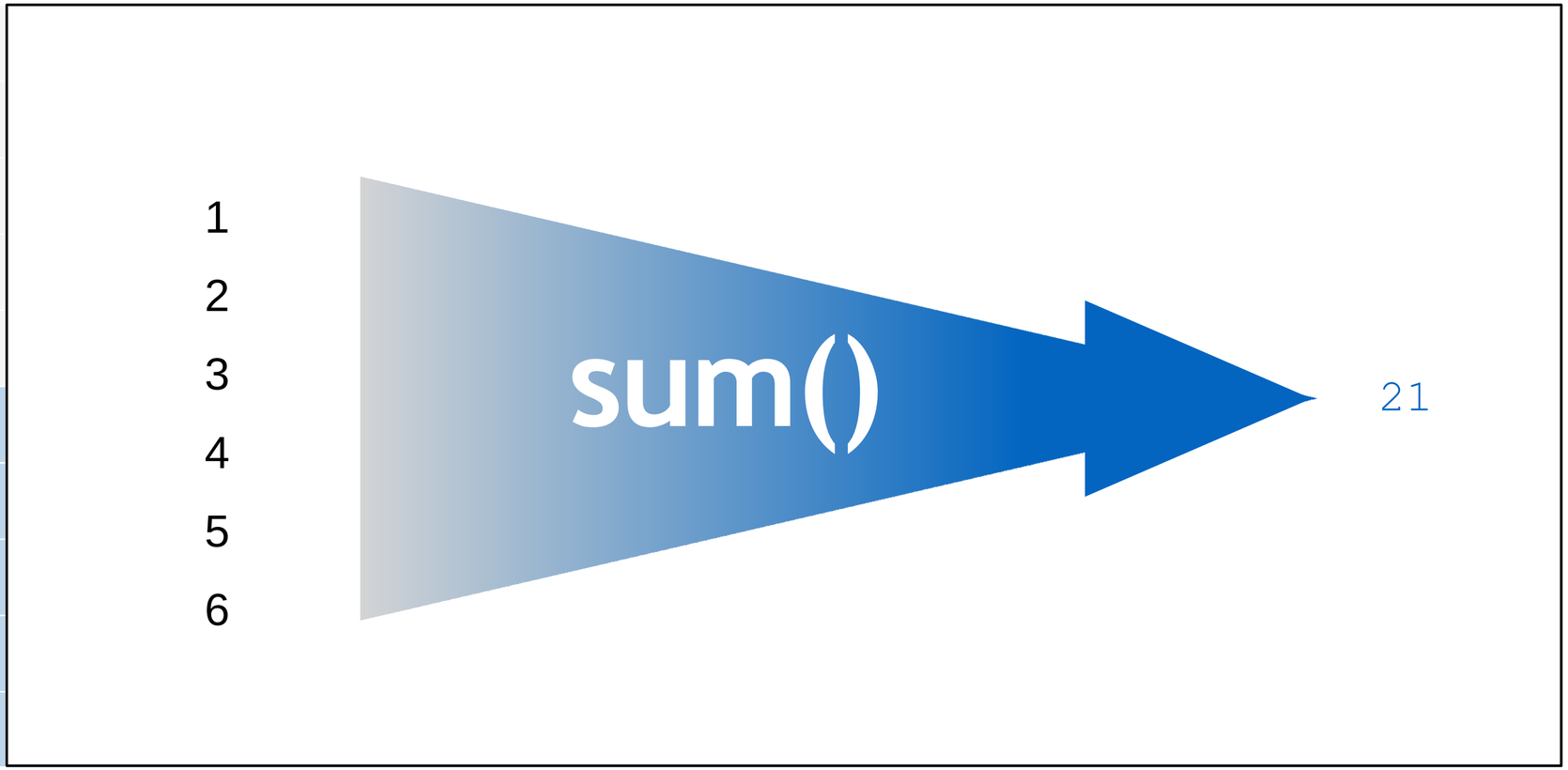
* All take a vector of values and return a single value
** Blue functions come in dplyr

min(), max()	Minimum and maximum values
mean()	Mean value
median()	Median value
sum()	Sum of values
var, sd()	Variance and standard deviation of a vector
first()	First value in a vector
last()	Last value in a vector
nth()	Nth value in a vector
n()	The number of values in a vector
n_distinct()	The number of distinct values in a vector

"Summary" functions

* All take a vector of values and return a single value

min(), max()
mean()
median()
sum()
var, sd()
first()
last()
nth()
n()
n_distinct()



Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

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	pressure
Alberto	1007
Allison	1005
Arlene	1010

```
storms %>%
  filter(wind >= 50) %>%
  select(storm, pressure)
```

mutate()

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


```
storms %>%
  mutate(ratio = pressure / wind) %>%
  select(storm, ratio)
```

mutate()

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	ratio
Alberto	9.15
Alex	22.42
Allison	15.46
Ana	25.32
Arlene	20.20
Arthur	22.44

```
storms %>%
  mutate(ratio = pressure / wind) %>%
  select(storm, ratio)
```

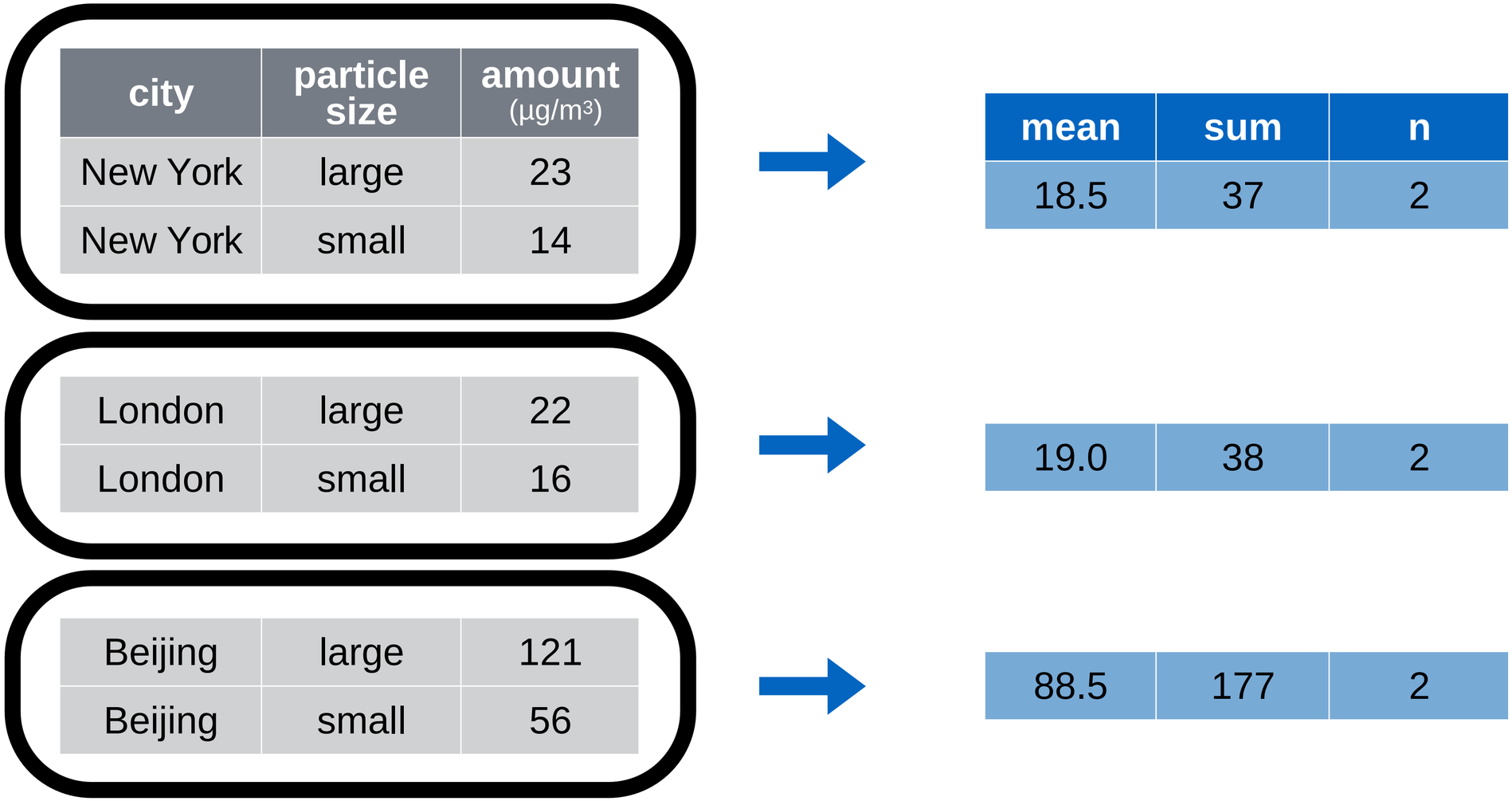
Unit of analysis

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

city	particle size	amount (µg/m³)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

mean	sum	n
42	252	6

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)



group_by() + summarise()

group_by()

city	particle size	amount (µg/m³)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

city	particle size	amount (µg/m³)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

```
pollution %>% group_by(city)
```

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

```
pollution %>% group_by(city)
## Source: local data frame [6 x 3]
## Groups: city
##
##       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
```

group_by() + summarise()

city	particle size	amount (µg/m³)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

```
pollution %>% group_by(city) %>%
  summarise(mean = mean(amount), sum = sum(amount), n = n())
```

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

city	particle size	amount (µg/m³)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

city	mean	sum	n
New York	18.5	37	2
London	19.0	38	2
Beijing	88.5	177	2

```
pollution %>% group_by(city) %>%
  summarise(mean = mean(amount), sum = sum(amount), n = n())
```

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

city	particle size	amount (µg/m³)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

city	mean	sum	n
New York	18.5	37	2
London	19.0	38	2
Beijing	88.5	177	2

```
pollution %>% group_by(city) %>%
  summarise(mean = mean(amount), sum = sum(amount), n = n())
```

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

city	particle size	amount (µg/m³)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

city	mean	sum	n
New York	18.5	37	2
London	19.0	38	2
Beijing	88.5	177	2


```
pollution %>% group_by(city) %>%
  summarise(mean = mean(amount), sum = sum(amount), n = n())
```

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

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	mean
New York	18.5
London	19.0
Beijing	88.5

```
pollution %>% group_by(city) %>%  summarise(mean = mean(amount))
```

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

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

size	mean
large	55.3
small	28.6

```
pollution %>% group_by(size) %>%  summarise(mean = mean(amount))
```

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

ungroup()

city	particle size	amount (µg/m³)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

city	particle size	amount (µg/m³)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

```
pollution %>% ungroup()
```

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

country	year	sex	cases
Afghanistan	1999	female	1
Afghanistan	1999	male	1
Afghanistan	2000	female	1
Afghanistan	2000	male	1
Brazil	1999	female	2
Brazil	1999	male	2
Brazil	2000	female	2
Brazil	2000	male	2
China	1999	female	3
China	1999	male	3
China	2000	female	3
China	2000	male	3

```
tb
```

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

country	year	sex	cases
Afghanistan	1999	female	1
Afghanistan	1999	male	1
Afghanistan	2000	female	1
Afghanistan	2000	male	1
Brazil	1999	female	2
Brazil	1999	male	2
Brazil	2000	female	2
Brazil	2000	male	2
China	1999	female	3
China	1999	male	3
China	2000	female	3
China	2000	male	3

country	year	sex	cases
Afghanistan	1999	female	1
Afghanistan	1999	male	1
Afghanistan	2000	female	1
Afghanistan	2000	male	1
Brazil	1999	female	2
Brazil	1999	male	2
Brazil	2000	female	2
Brazil	2000	male	2
China	1999	female	3
China	1999	male	3
China	2000	female	3
China	2000	male	3

```
tb %>%
  group_by(country, year)
```

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

country	year	sex	cases
Afghanistan	1999	female	1
Afghanistan	1999	male	1
Afghanistan	2000	female	1
Afghanistan	2000	male	1
Brazil	1999	female	2
Brazil	1999	male	2
Brazil	2000	female	2
Brazil	2000	male	2
China	1999	female	3
China	1999	male	3
China	2000	female	3
China	2000	male	3

country	year	sex	cases
Afghanistan	1999	female	1
Afghanistan	1999	male	1
Afghanistan	2000	female	1
Afghanistan	2000	male	1
Brazil	1999	female	2
Brazil	1999	male	2
Brazil	2000	female	2
Brazil	2000	male	2
China	1999	female	3
China	1999	male	3
China	2000	female	3
China	2000	male	3

country	year	cases
Afghanistan	1999	2
Afghanistan	2000	2
Brazil	1999	4
Brazil	2000	4
China	1999	6
China	1999	6

```
tb %>%
  group_by(country, year) %>%
  summarise(cases = sum(cases))
```

Retrieve from http://goo.gl/cJJY47 (Garrett Grolemund, 2015)

country	year	sex	cases
Afghanistan	1999	female	1
Afghanistan	1999	male	1
Afghanistan	2000	female	1
Afghanistan	2000	male	1
Brazil	1999	female	2
Brazil	1999	male	2
Brazil	2000	female	2
Brazil	2000	male	2
China	1999	female	3
China	1999	male	3
China	2000	female	3
China	2000	male	3

country	year	sex	cases
Afghanistan	1999	female	1
Afghanistan	1999	male	1
Afghanistan	2000	female	1
Afghanistan	2000	male	1
Brazil	1999	female	2
Brazil	1999	male	2
Brazil	2000	female	2
Brazil	2000	male	2
China	1999	female	3
China	1999	male	3
China	2000	female	3
China	2000	male	3

country	year	cases
Afghanistan	1999	2
Afghanistan	2000	2
Brazil	1999	4
Brazil	2000	4
China	1999	6
China	1999	6

country	cases
Afghanistan	4
Brazil	8
China	12

```
tb %>%
  group_by(country, year) %>%
  summarise(cases = sum(cases)) %>%
  summarise(cases = sum(cases))
```

Hierarchy of information

country	year	sex	cases
Afghanistan	1999	female	1
Afghanistan	1999	male	1
Afghanistan	2000	female	1
Afghanistan	2000	male	1
Brazil	1999	female	2
Brazil	1999	male	2
Brazil	2000	female	2
Brazil	2000	male	2
China	1999	female	3
China	1999	male	3
China	2000	female	3
China	2000	male	3

country	year	cases
Afghanistan	1999	2
Afghanistan	2000	2
Brazil	1999	4
Brazil	2000	4
China	1999	6
China	2000	6

country	cases
Afghanistan	4
Brazil	8
China	12

cases
24

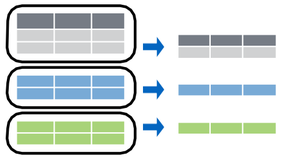
Larger units of analysis

Recap: Information

Extract variables and observations with **select()** and **filter()**

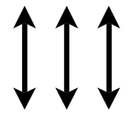
Arrange observations, with **arrange()**.

Make new variables, with **mutate()**.

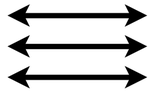


Make groupies observations with **group_by()** and **summarise()**.

Recap: Best format for analysis



Variables in columns



Observations in rows

Separate **all variables** *implied by law, formula or goal*

Unit of analysis matches the unit of analysis *implied by law, formula or goal*

Single table

Outline

Toy Example

- Tidy data? View() & glimpse()
- tidyr: reshape data
  - gather() columns into rows
  - spread() rows into columns
  - unite() several columns into one
  - separate() one column into several
- dplyr: transform data
  - filter() observations in rows
  - select() variables in columns
  - arrange() observations in rows
  - mutate() from old var. to new one
  - summarise() the data by column
  - group_by() certain categories

Real Data

- **Weather Forecasting Data**
  1. **Tidy data?**
     - **Variables are stored in both rows and columns**
     - **Duplicate observations**
  2. **tidyr & dplyr example**
     - **clean.data <- raw.data %>%**
       mutate() %>%
       select() %>%
       distinct() %>%
       spread() %>% …
     - **clean.data %>%**
       filter() %>%
       group_by() %>%
       summarise()