

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



**tidyr**



**dplyr**

## Data Wrangling with dplyr and tidyr Cheat Sheet

R Studio

### 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).

**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)
```

### 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.

**Reshaping Data - Change the layout of a data set**

**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).

### Subset Variables (Columns)

**dplyr::select(iris, Sepal.Width, Petal.Length, Species)**  
Select columns by name or helper function.

#### Helper functions for select - ?select

<b>select(iris, contains("."))</b>
Select columns whose name contains a character string.
<b>select(iris, ends_with("Length"))</b>
Select columns whose name ends with a character string.
<b>select(iris, everything())</b>
Select every column.
<b>select(iris, matches("t.*"))</b>
Select columns whose name matches a regular expression.
<b>select(iris, num_range("x", 1:5))</b>
Select columns named x1, x2, x3, x4, x5.
<b>select(iris, one_of(c("Species", "Genus")))</b>
Select columns whose names are in a group of names.
<b>select(iris, starts_with("Sepa"))</b>
Select columns whose name starts with a character string.
<b>select(iris, Sepal.Length:Petal.Width)</b>
Select all columns between Sepal.Length and Petal.Width (inclusive).
<b>select(iris, -Species)</b>
Select all columns except Species.

<http://www.rstudio.com/resources/cheatsheets/>

© 2014 RStudio, Inc. All rights reserved.

RStudio® is a trademark of RStudio, Inc. • All rights reserved • [info@rstudio.com](mailto:info@rstudio.com) • 844-448-1212 • [rstudio.com](http://rstudio.com) devtools::install\_github("rstudio/EDAWR") for data sets Learn more with browseVignettes(package = c("dplyr", "tidyverse")) • dplyr 0.4.0 • tidyverse 0.2.0 • Updated: 1/15

# **Ground rules**

# tbl's

Just like data frames, but play better with the console window.

Source: local data frame [53,940 x 10]

	carat	cut	color	clarity	depth	table
1	0.23	Ideal	E	SI2	61.5	55
2	0.21	Premium	E	SI1	59.8	61
3	0.23	Good	E	VS1	56.9	65
4	0.29	Premium	I	VS2	62.4	58
5	0.31	Good	J	SI2	63.3	58
6	0.24	Very Good	J	VVS2	62.8	57
7	0.24	Very Good	I	VVS1	62.3	57
8	0.26	Very Good	H	SI1	61.9	55
9	0.22	Fair	E	VS2	65.1	61
10	0.23	Very Good	H	VS1	59.4	61
..	...	...	...	...	...	...

Variables not shown: price (int), x (dbl), y (dbl), z (dbl)

tbl

977	59.0	2893	6.09	6.06	3.64
978	57.0	2894	5.91	5.99	3.71
979	57.0	2894	5.96	6.00	3.72
980	56.0	2894	5.88	5.92	3.62
981	56.0	2895	5.75	5.78	3.51
982	59.0	2895	5.66	5.76	3.53
983	53.0	2895	5.71	5.75	3.56

986	63.0	2896	6.00	6.05	3.51
987	56.0	2896	5.18	5.24	3.21
988	56.0	2896	5.91	5.96	3.65
989	55.0	2896	5.82	5.86	3.59
990	56.0	2896	5.83	5.89	3.64
991	58.0	2896	5.94	5.88	3.60
992	57.0	2896	6.39	6.35	4.02
993	57.0	2896	6.46	6.45	3.97
994	57.0	2897	5.48	5.51	3.33
995	58.0	2897	5.91	5.85	3.59
996	52.0	2897	5.30	5.34	3.26
997	55.0	2897	5.69	5.74	3.57
998	61.0	2897	5.82	5.89	3.48
999	58.0	2897	5.81	5.77	3.58
1000	59.0	2898	6.68	6.61	4.03

[ reached getOption("max.print") --  
omitted 52940 rows ]

data.frame

# View()

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

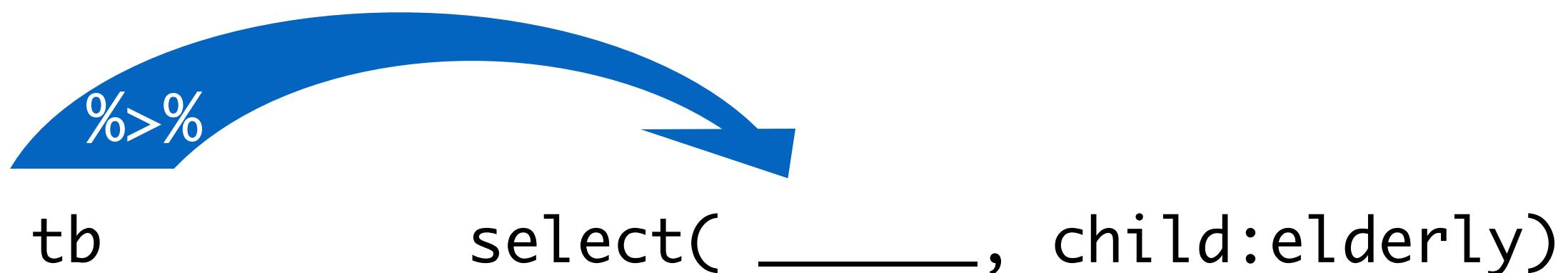
View(iris)  
View(mtcars)  
View(pressure)



	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
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa
11	5.4	3.7	1.5	0.2	setosa
12	4.8	3.4	1.6	0.2	setosa
13	4.8	3.0	1.4	0.1	setosa
14	4.3	3.0	1.1	0.1	setosa

# The pipe %>% operator

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



# Data Wrangling

# Wrangling Munging Janitor Work Manipulation Transformation

**50-80%**  
of your time?

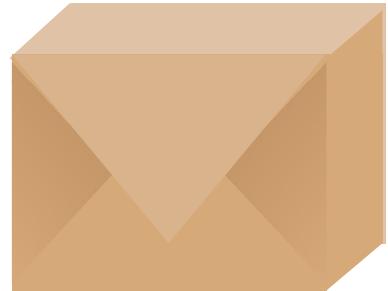
# Two goals

- 1** Make data suitable to use with  
a particular piece of software
- 2** Reveal information

**Data sets  
come in many  
formats**

**...but R prefers just one.**

# EDAWR



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

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

```
# devtools::install_github("rstudio/EDAWR")
library(EDAWR)
```

**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 ( $\mu\text{g}/\text{m}^3$ )
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

```
# devtools::install_github("rstudio/EDAWR")
library(EDAWR)
```

**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 ( $\mu\text{g}/\text{m}^3$ )
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

- Storm name

```
# devtools::install_github("rstudio/EDAWR")
library(EDAWR)
```

**storms**

storm	wind	pressure	date
Alberto	10	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 ( $\mu\text{g}/\text{m}^3$ )
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)

```
# devtools::install_github("rstudio/EDAWR")
library(EDAWR)
```

**storms**

storm	wind	pressure	date
Alberto	100	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**

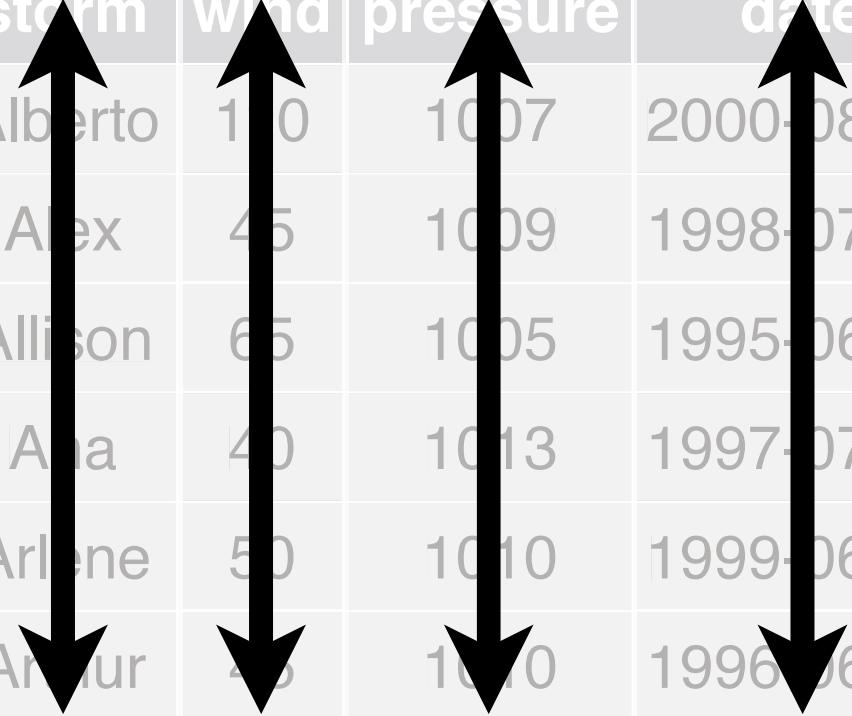
city	particle size	amount ( $\mu\text{g}/\text{m}^3$ )
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

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

```
# devtools::install_github("rstudio/EDAWR")
library(EDAWR)
```

**storms**

storm	wind	pressure	date
Alberto	10	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**

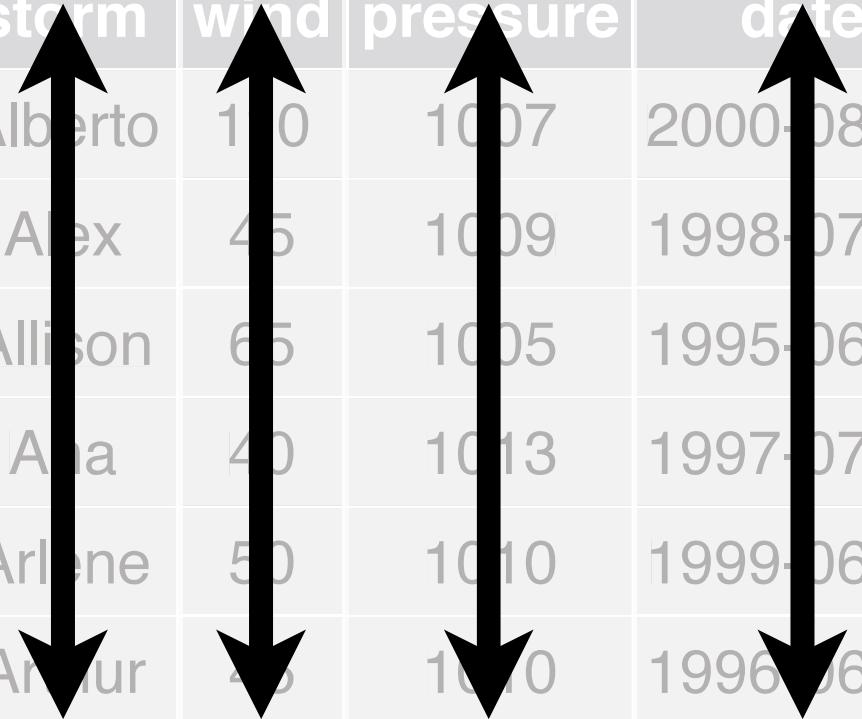
city	particle size	amount ( $\mu\text{g}/\text{m}^3$ )
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

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

```
# devtools::install_github("rstudio/EDAWR")
library(EDAWR)
```

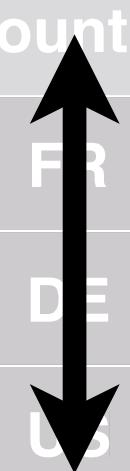
storms

storm	wind	pressure	date
Alberto	10	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 ( $\mu\text{g}/\text{m}^3$ )
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

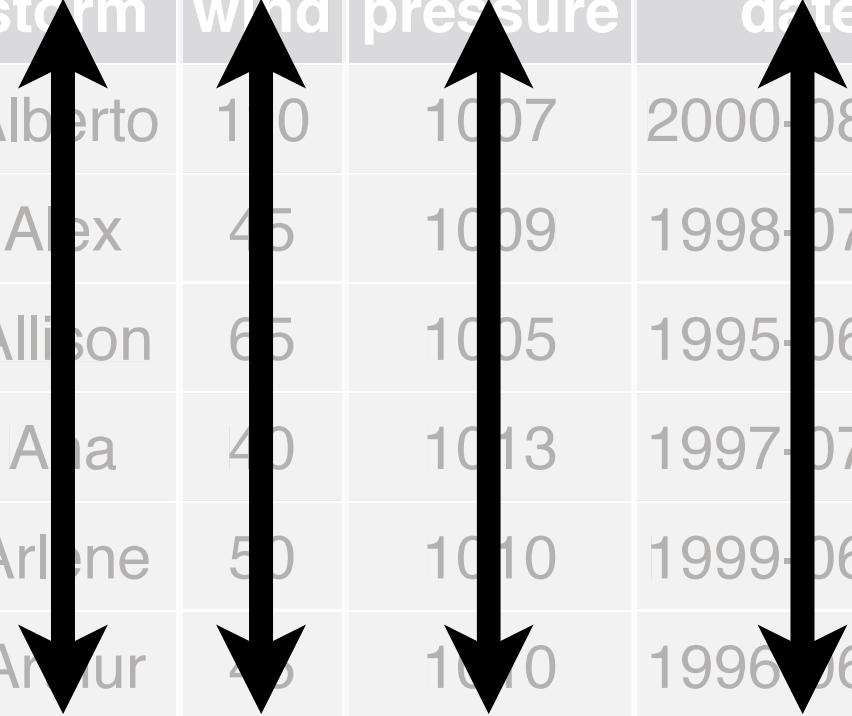
- Storm name
- Wind Speed (mph)
- Air Pressure
- Date

- Country

```
# devtools::install_github("rstudio/EDAWR")
library(EDAWR)
```

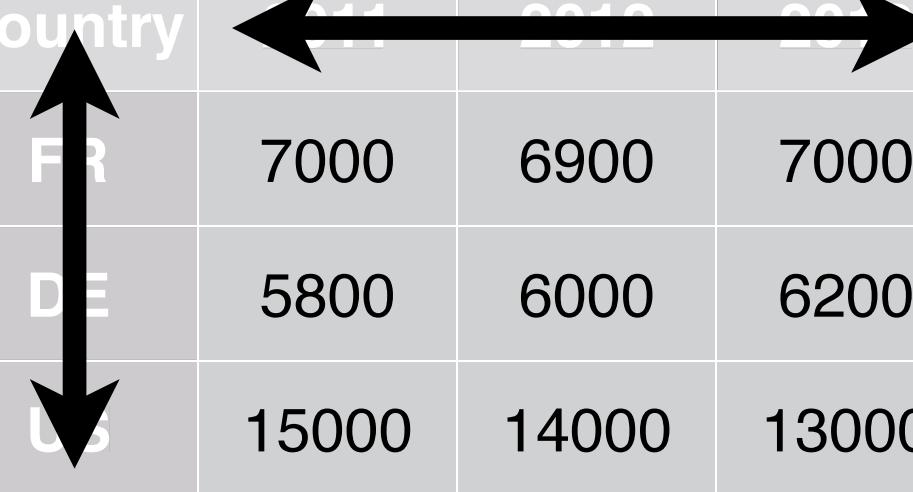
storms

storm	wind	pressure	date
Alberto	10	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	1994	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000



pollution

city	particle size	amount ( $\mu\text{g}/\text{m}^3$ )
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
- Date

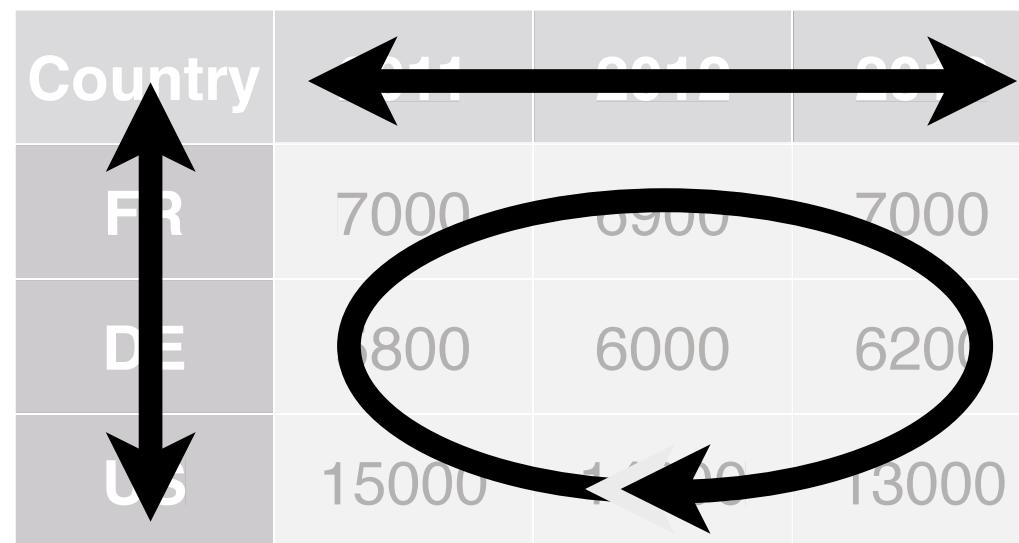
- Country
- Year

```
# devtools::install_github("rstudio/EDAWR")
library(EDAWR)
```

storms

storm	wind	pressure	date
Alberto	10	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



pollution

city	particle size	amount ( $\mu\text{g}/\text{m}^3$ )
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
- Date

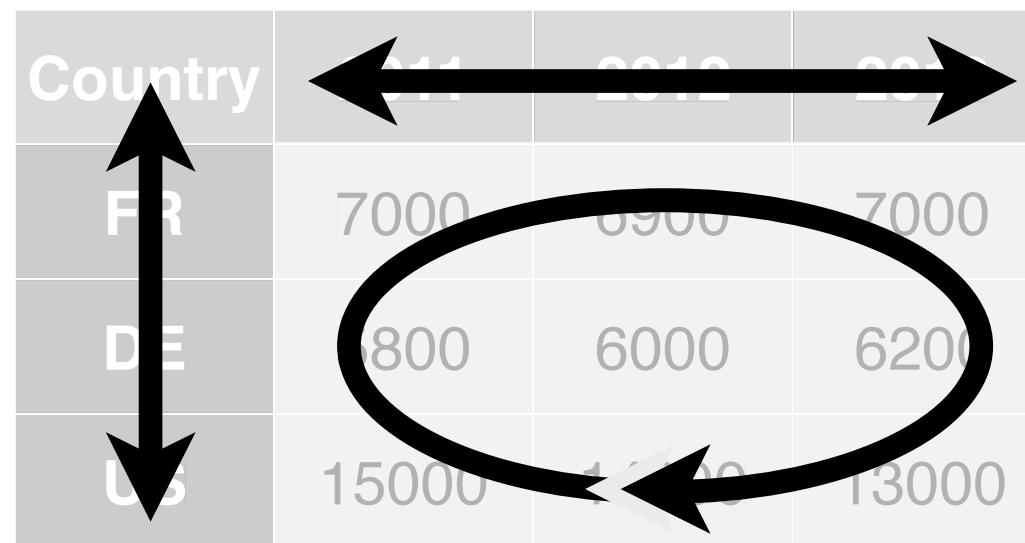
- Country
- Year
- Count

```
# devtools::install_github("rstudio/EDAWR")  
library(EDAWR)
```

storms

storm	wind	pressure	date
Alberto	10	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



pollution

city	particle size	amount ( $\mu\text{g}/\text{m}^3$ )
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
- Date

- Country
- Year
- Count

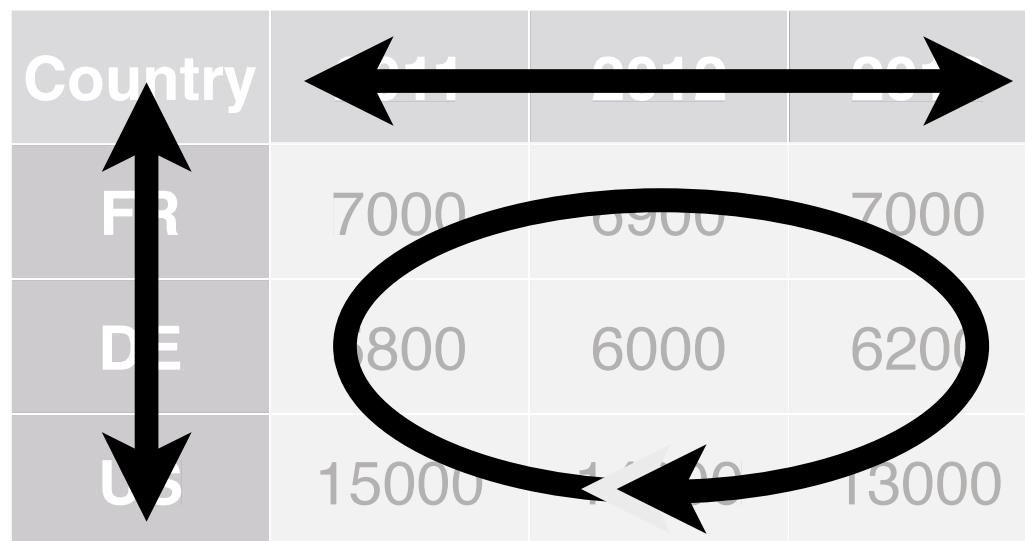
- City

```
# devtools::install_github("rstudio/EDAWR")
library(EDAWR)
```

storms

storm	wind	pressure	date
Alberto	10	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



pollution

city	particle size	amount ( $\mu\text{g}/\text{m}^3$ )
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
- Date

- Country
- Year
- Count

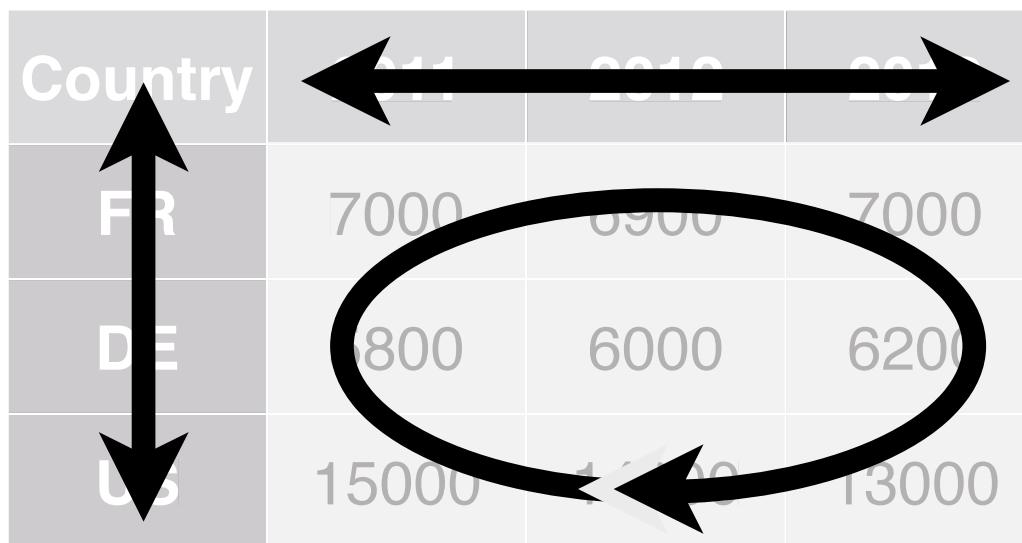
- City
- Amount of large particles

```
# devtools::install_github("rstudio/EDAWR")
library(EDAWR)
```

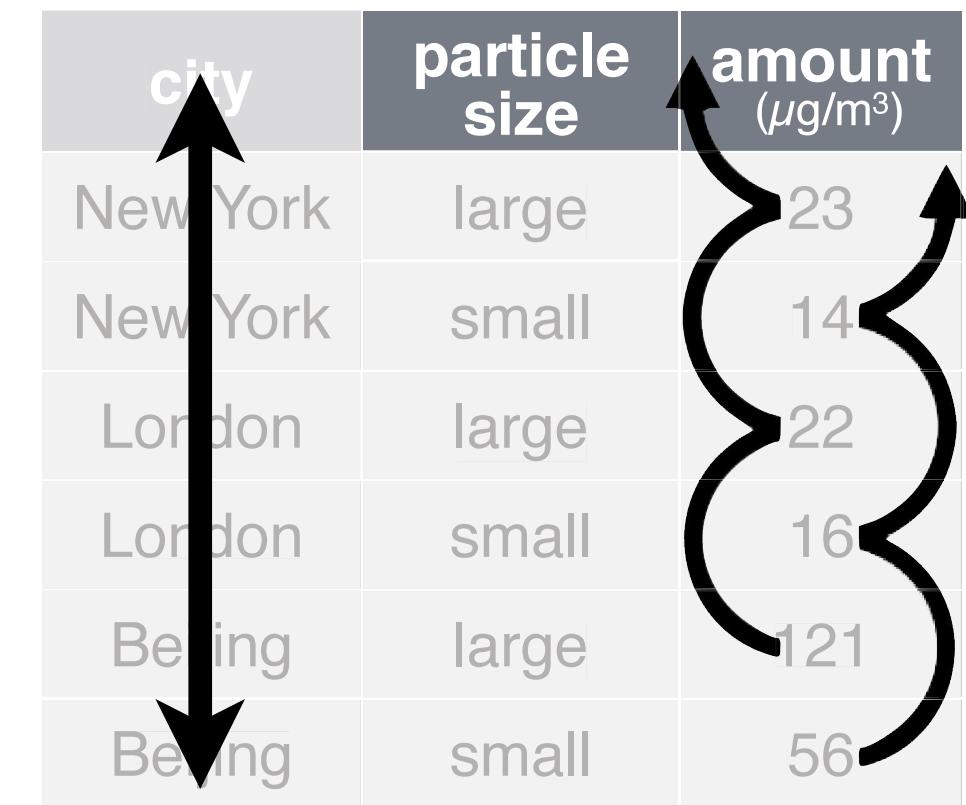
storms

storm	wind	pressure	date
Alberto	10	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



pollution



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

- Country
- Year
- Count

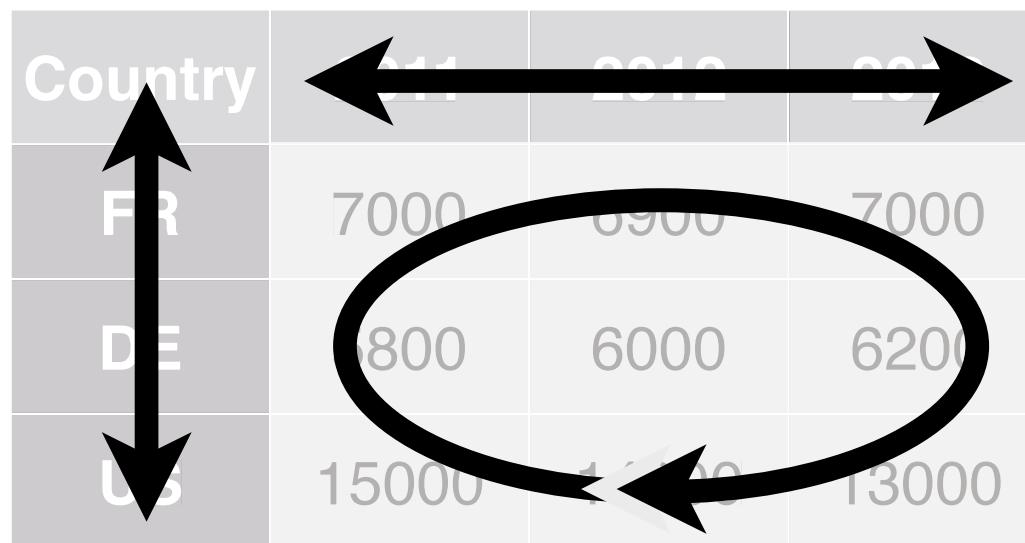
- City
- Amount of large particles
- Amount of small particles

```
# devtools::install_github("rstudio/EDAWR")
library(EDAWR)
```

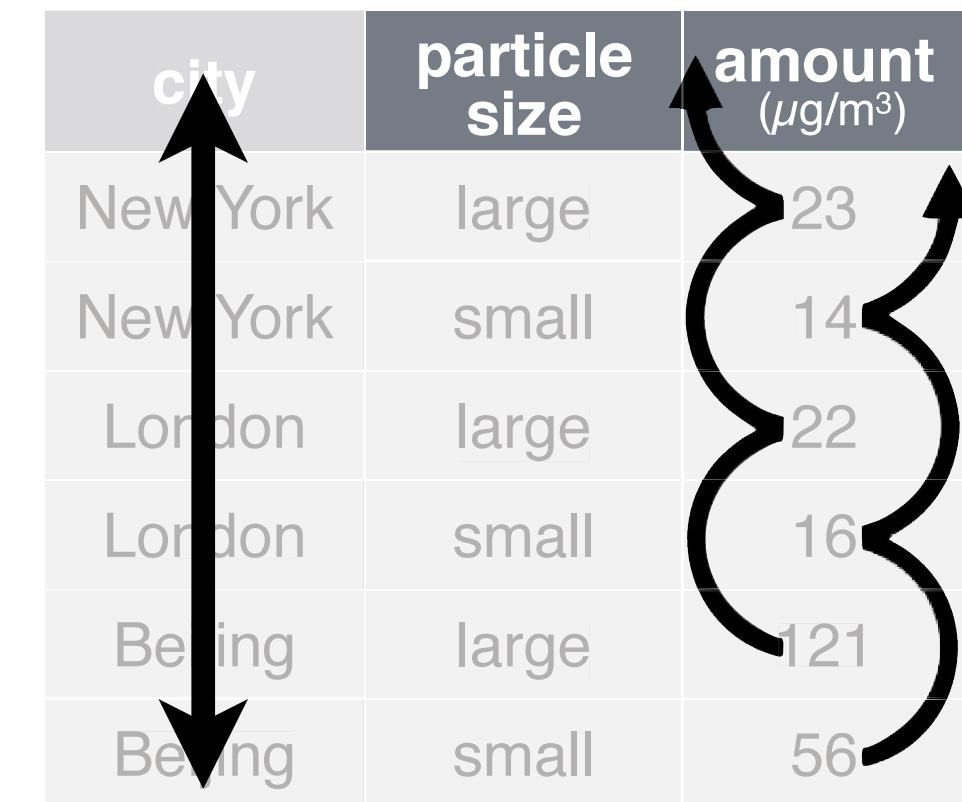
storms

	storm	wind	pressure	date
Alberto	10	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



pollution



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

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

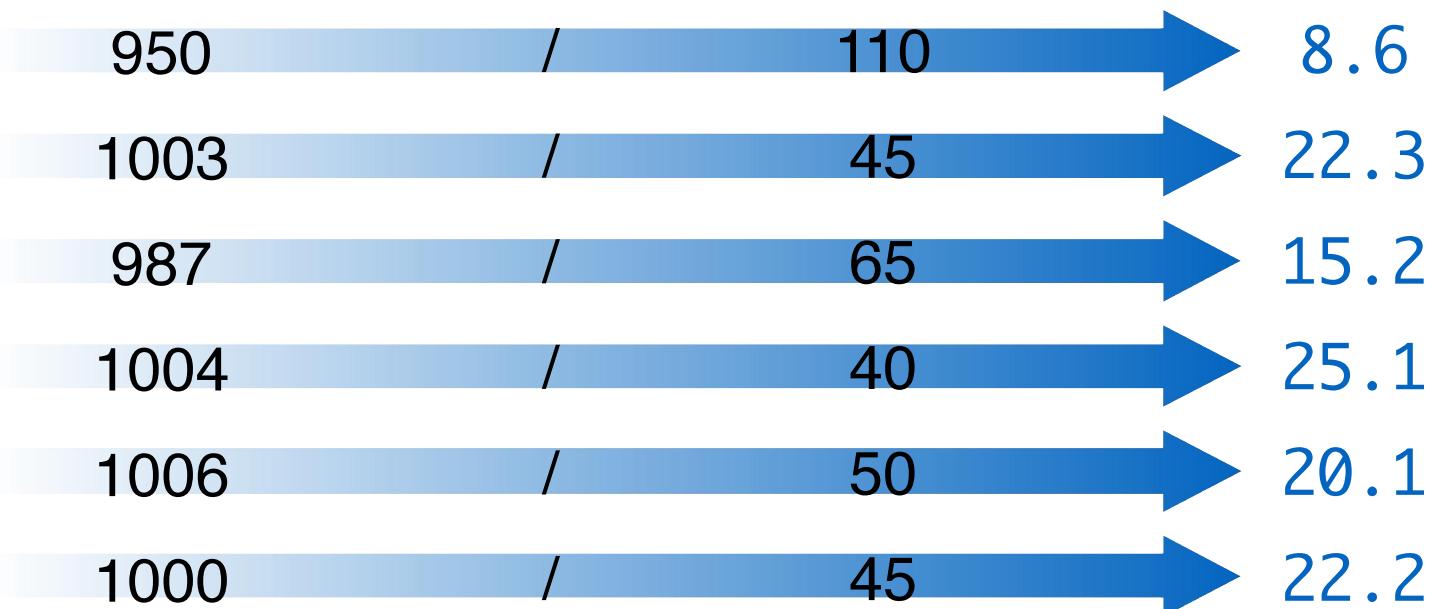
pollution\$city[1,3,5]  
pollution\$amount[1,3,5]  
pollution\$amount[2,4,6]

ratio =  $\frac{\text{pressure}}{\text{wind}}$

## 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

`storms$pressure / storms$wind`



# Tidy data

storms

	storm	wind	pressure	date
Alberto	10	1007	2000-07-12	
Alex	45	1009	1998-07-30	
Amber	65	1005	1995-07-04	
Ana	40	1013	1997-07-01	
Anne	30	1010	1999-07-13	
Anton	45	1010	1996-07-21	

- 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).

# Recap: Tidy data

123

Variables in columns, observations in rows,  
each type in a table



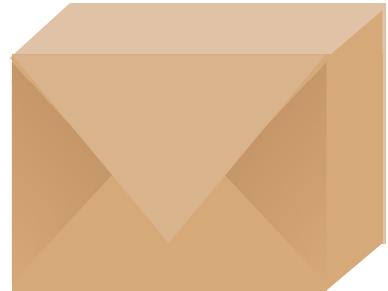
Easy to access variables



Automatically preserves observations

**tidyr**

# 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

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

slides at: [bit.ly/wrangling-webinar](http://bit.ly/wrangling-webinar)

© 2014 RStudio, Inc. All rights reserved.

00 : 30

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

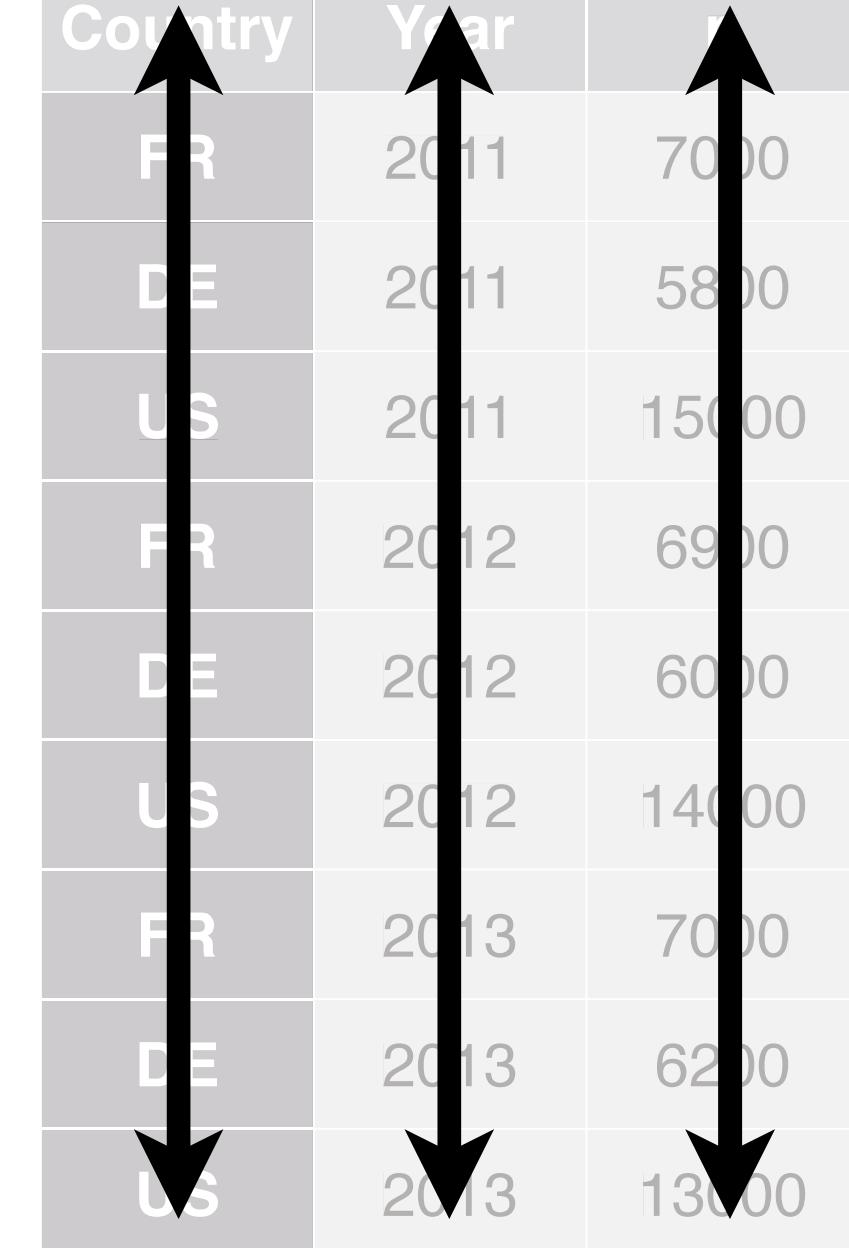
Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

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	R
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

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

## key (former column names)

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

**key**   **value** (former cells)

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()

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)
```

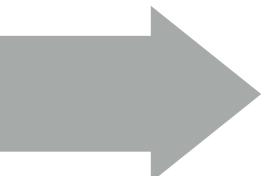
data frame  
to reshape

name of the new  
key column  
(a character string)

name of the new  
value column  
(a character string)

names or numeric  
indexes of columns  
to collapse

```
##   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

city	particle size	amount ( $\mu\text{g}/\text{m}^3$ )
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](http://bit.ly/wrangling-webinar)

© 2014 RStudio, Inc. All rights reserved.

00 : 30

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

## 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:

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

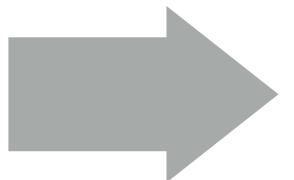
`spread(pollution, size, amount)`

data frame  
to reshape

column to use for  
keys (new columns  
names)

column to use for  
values (new  
column cells)

```
##          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)`

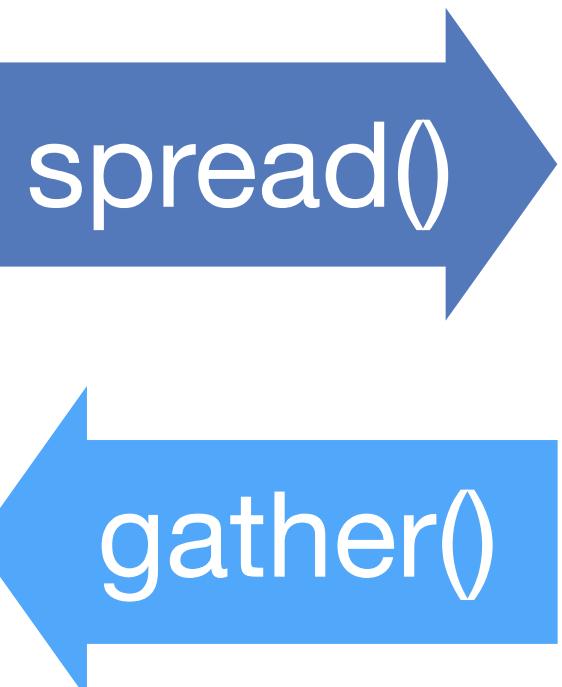
city	size	amount
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56



spread()

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

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

# 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 = "-")
```

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	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.

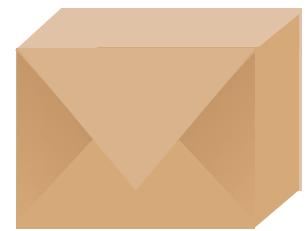
```
unite(storms2, "date", year, month, day, sep = "-")
```

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

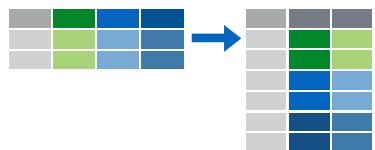


storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

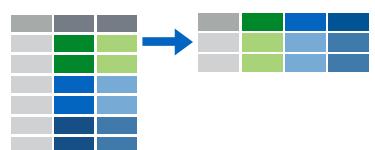
# Recap: `tidyverse`



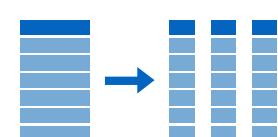
A package that reshapes the layout of data sets.



Make observations from variables with `gather()`



Make variables from observations with `spread()`



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

**Data sets contain  
more information  
than they display**

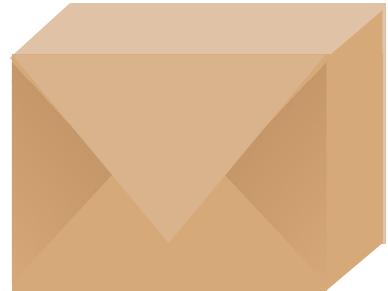
# dplyr



A package that helps transform  
tabular data.

```
# install.packages("dplyr")
library(dplyr)
?select
?filter
?arrange
?mutate
?summarise
?group_by
```

# nycflights13



Data sets related to flights that departed from NYC in 2013

```
# install.packages("nycflights13")
library(nycflights13)
?airlines          ?planes
?airports         ?weather
?flights
```

# 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

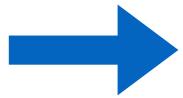
\* Blue functions come in dplyr

-	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

# 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"))
```

# 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

# 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

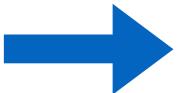


storm	wind	pressure	date	ratio
Alberto	110	1007	2000-08-12	9.15
Alex	45	1009	1998-07-30	22.42
Allison	65	1005	1995-06-04	15.46
Ana	40	1013	1997-07-01	25.32
Arlene	50	1010	1999-06-13	20.20
Arthur	45	1010	1996-06-21	22.44

```
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

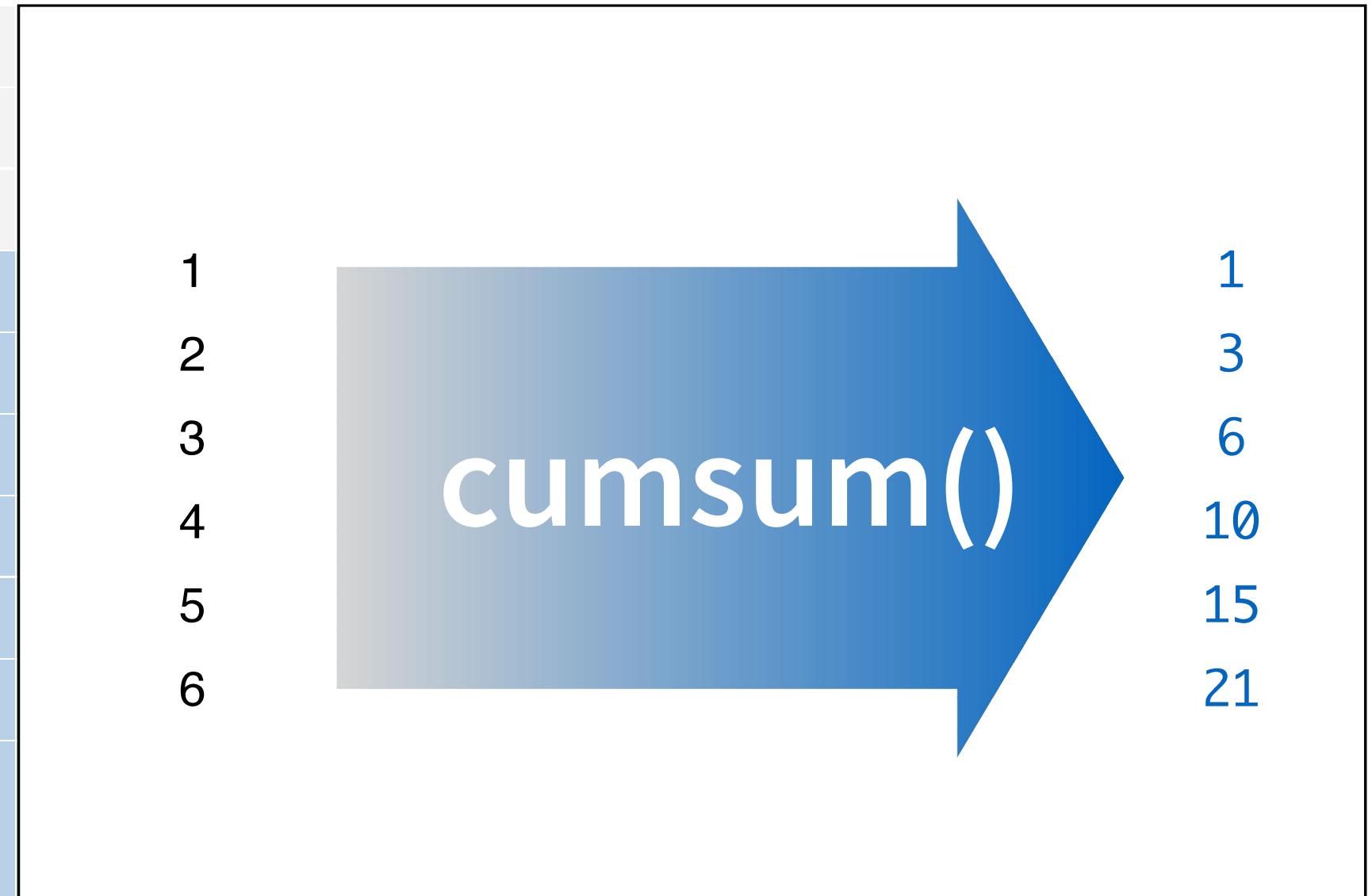
- \* All take a vector of values and return a vector of values
- \*\* Blue functions come in dplyr

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 ( $\mu\text{g}/\text{m}^3$ )
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 ( $\mu\text{g}/\text{m}^3$ )
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()

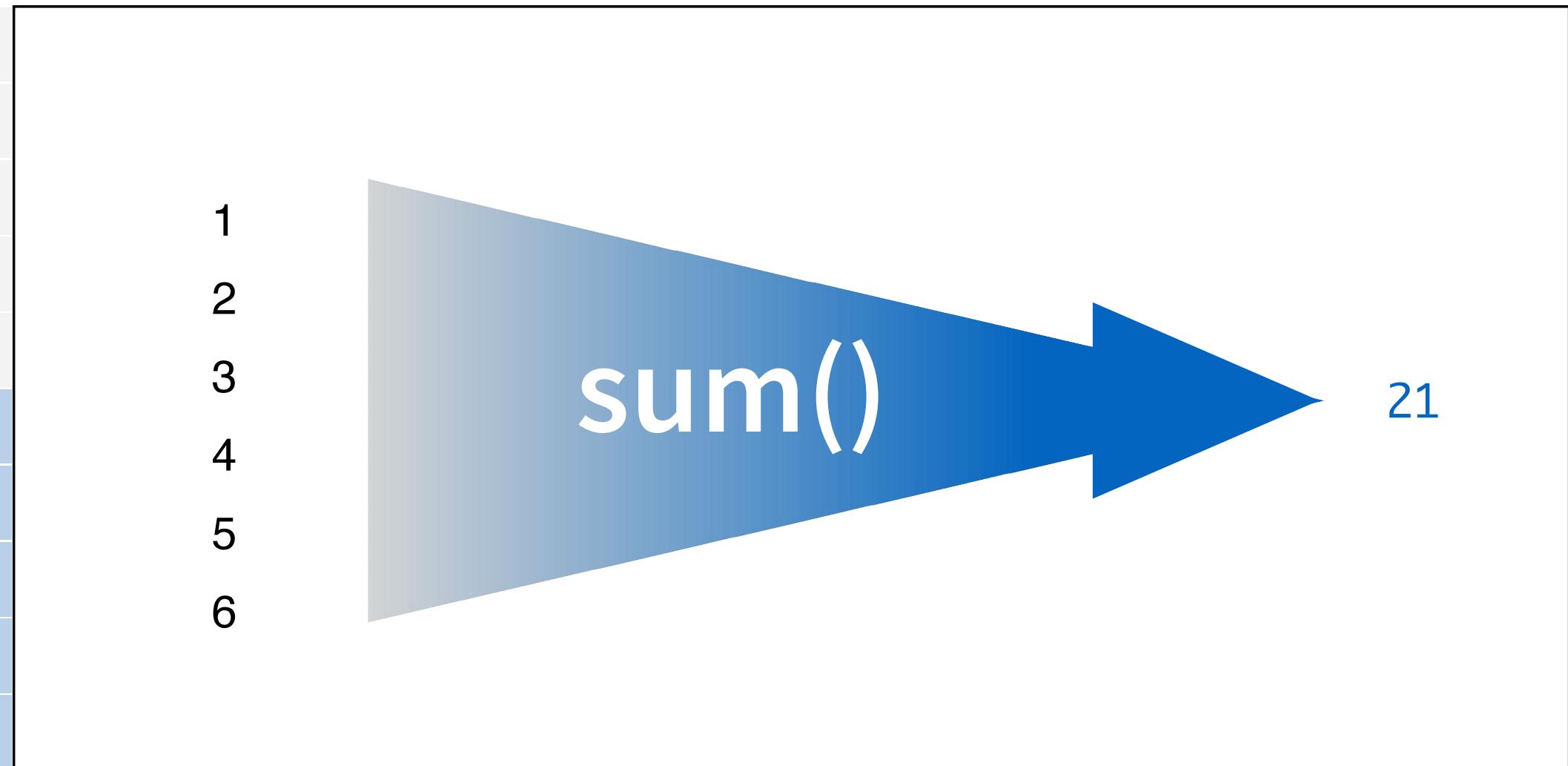
first()

last()

nth()

n()

n\_distinct()



# 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
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))`

# 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



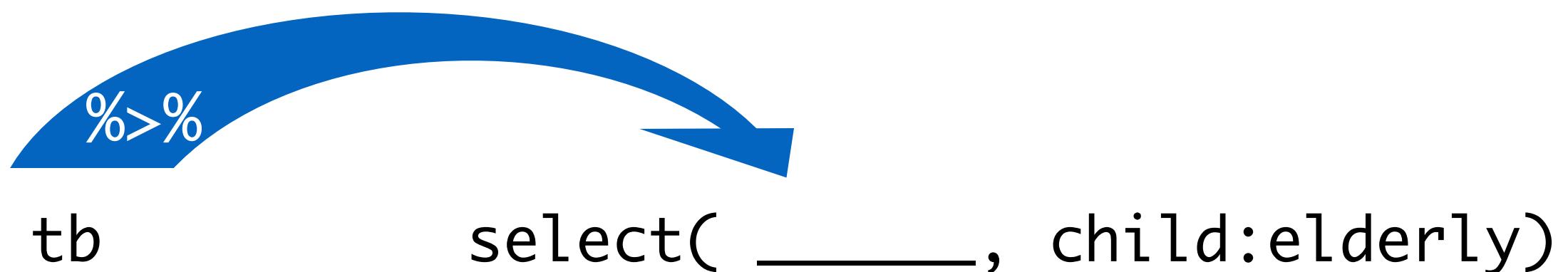
storm	wind	pressure	date
Ana	40	1013	1997-07-01
Arthur	45	1010	1996-06-21
Alex	45	1009	1998-07-30
Arlene	50	1010	1999-06-13
Allison	65	1005	1995-06-04
Alberto	110	1007	2000-08-12



`arrange(storms, wind, date)`

# The pipe %>% operator

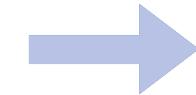
```
library(dplyr)  
select(tb, child:elderly)  
tb %>% select(child:elderly)
```



# 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



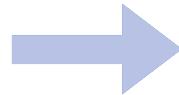
storm	pressure
Alberto	1007
Alex	1009
Allison	1005
Ana	1013
Arlene	1010
Arthur	1010

`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



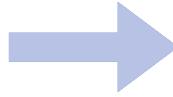
storm	pressure
Alberto	1007
Alex	1009
Allison	1005
Ana	1013
Arlene	1010
Arthur	1010

storms %>% select(storm, pressure)

# 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



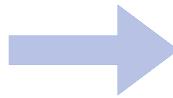
storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Allison	65	1005	1995-06-04
Arlene	50	1010	1999-06-13

`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
Arlene	50	1010	1999-06-13

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

## 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 %>%

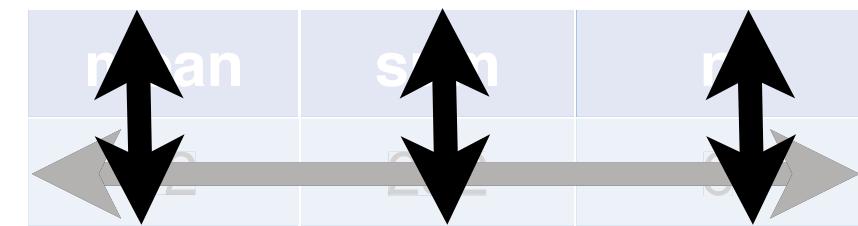
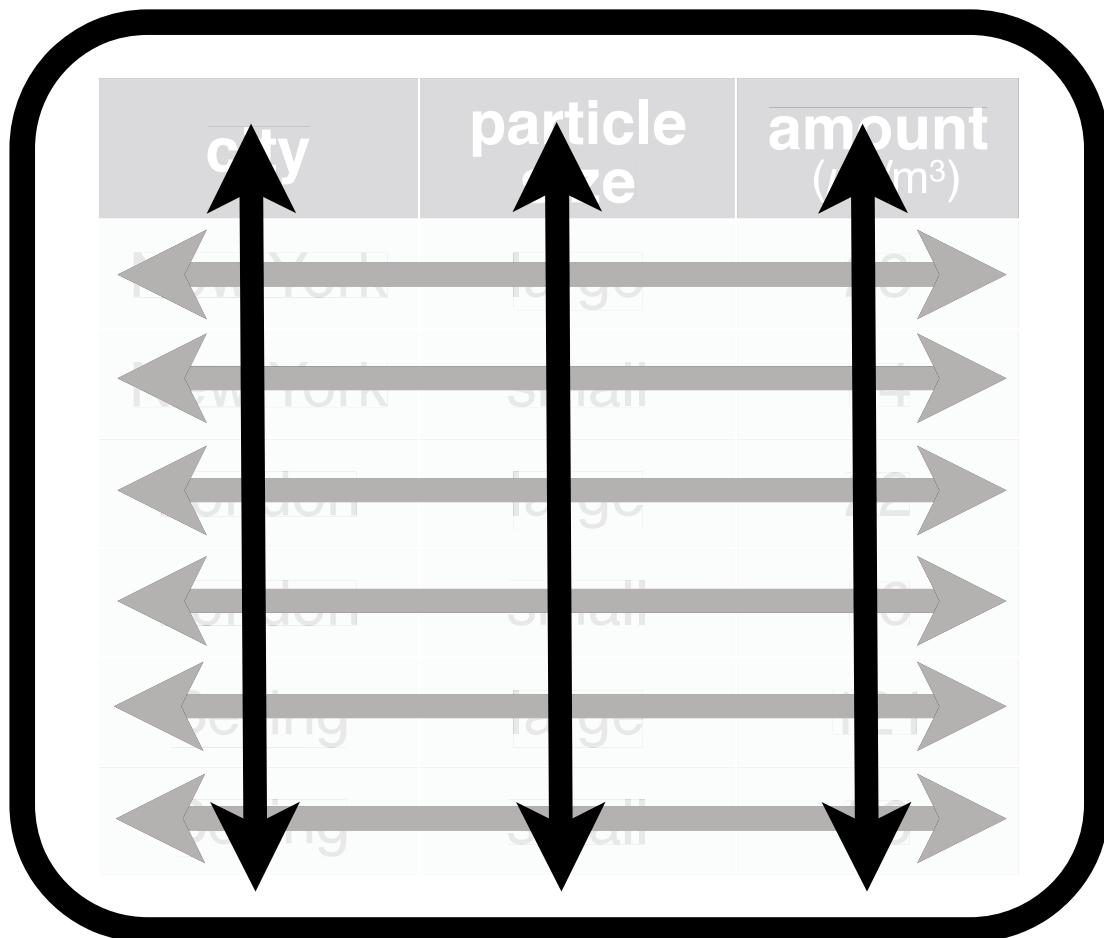
```
  mutate(ratio = pressure / wind) %>%  
  select(storm, ratio)
```

# Shortcut to type %>%

Cmd + Shift + M (Mac)

Ctrl + Shift + M (Windows)

**Unit of  
analysis**



city	particle size	amount ( $\mu\text{g}/\text{m}^3$ )
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

city	particle size	amount ( $\mu\text{g}/\text{m}^3$ )
New York	large	23
New York	small	14



mean	sum	n
18.5	37	2

London	large	22
London	small	16



19.0	38	2
------	----	---

Beijing	large	121
Beijing	small	56



88.5	177	2
------	-----	---

group\_by() + summarise()

# group\_by()

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



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

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

```
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 ( $\mu\text{g}/\text{m}^3$ )
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())
```

city	particle size	amount ( $\mu\text{g}/\text{m}^3$ )
New York	large	23
New York	small	14



city	mean	sum	n
New York	18.5	37	2

London	large	22
London	small	16



London	19.0	38	2
--------	------	----	---

Beijing	large	121
Beijing	small	56



Beijing	88.5	177	2
---------	------	-----	---

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

city	particle size	amount ( $\mu\text{g}/\text{m}^3$ )
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

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

Beijing	88.5	177	2
---------	------	-----	---

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

city	particle size	amount ( $\mu\text{g}/\text{m}^3$ )
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())
```

city	particle size	amount ( $\mu\text{g}/\text{m}^3$ )
New York	large	23
New York	small	14

London	large	22
London	small	16

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

Beijing	large	121
Beijing	small	56

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

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))
```

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))
```

# ungroup()

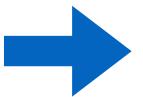
city	particle size	amount ( $\mu\text{g}/\text{m}^3$ )
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56



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

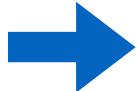
pollution %>% ungroup()

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

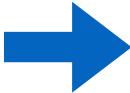
country	year	sex	cases
Afghanistan	1999	female	1
Afghanistan	1999	male	1
Afghanistan	2000	female	1
Afghanistan	2000	male	1
Brazil	1999	female	2
Brazil	1999	male	2
Brazil	2000	female	2
Brazil	2000	male	2
China	1999	female	3
China	1999	male	3
China	2000	female	3
China	2000	male	3



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)
```

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



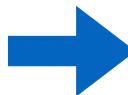
country	year	sex	cases
Afghanistan	1999	female	1
Afghanistan	1999	male	1
Afghanistan	2000	female	1
Afghanistan	2000	male	1
Brazil	1999	female	2
Brazil	1999	male	2
Brazil	2000	female	2
Brazil	2000	male	2
China	1999	female	3
China	1999	male	3
China	2000	female	3
China	2000	male	3



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))
```

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



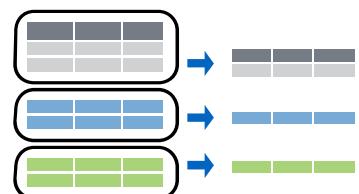
Extract variables and observations with  
**select()** and **filter()**



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



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



Make groupies observations with **group\_by()** and  
**summarise()**.

# Joining data

# dplyr::bind\_cols()

y	
x1	x2
A	1
B	2
C	3

1

z	
x1	x2
B	2
C	3
D	4

1

x1	x2	x1	x2
A	1	B	2
B	2	C	3
C	3	D	4

`bind_cols(y, z)`

# dplyr::bind\_rows()

y		z			
x1	x2	x1	x2	x1	x2
A	1	B	2	A	1
B	2	C	3	B	2
C	3	D	4	C	3

+

x1	x2
B	2
C	3
D	4

=

x1	x2
A	1
B	2
C	3
D	4

bind\_rows(y, z)

# dplyr::union()

y		z			
x1	x2	x1	x2	x1	x2
A	1	B	2	A	1
B	2	C	3	B	2
C	3	D	4	C	3
				D	4

+

=

`union(y, z)`

# dplyr::intersect()

y	
x1	x2
A	1
B	2
C	3

1

x1	x2
B	2
C	3
D	4

2

x1	x2
B	2
C	3

**intersect(y, z)**

# dplyr::setdiff()

y		z			
x1	x2	x1	x2	x1	x2
A	1	B	2	A	1
B	2	C	3	D	4
C	3	D	4		

+

=

`setdiff(y, z)`

# dplyr::left\_join()

songs		artists				
song	name	name	plays	song	name	plays
Across the Universe	John	George	sitar	Across the Universe	John	guitar
Come Together	John	John	guitar	Come Together	John	guitar
Hello, Goodbye	Paul	Paul	bass	Hello, Goodbye	Paul	bass
Peggy Sue	Buddy	Ringo	drums	Peggy Sue	Buddy	<NA>

`left_join(songs, artists, by = "name")`

# dplyr::left\_join()

songs		artists				
song	name	name	plays	song	name	plays
Across the Universe	John	George	sitar	Across the Universe	John	guitar
Come Together	John	John	guitar	Come Together	John	guitar
Hello, Goodbye	Paul	Paul	bass	Hello, Goodbye	Paul	bass
Peggy Sue	Buddy	Ringo	drums	Peggy Sue	Buddy	<NA>

`left_join(songs, artists, by = "name")`

# dplyr::left\_join()

songs2

song	first	last
Across the Universe	John	Lennon
Come Together	John	Lennon
Hello, Goodbye	Paul	McCartney
Peggy Sue	Buddy	Holly

+

artists2

first	last	plays
George	Harrison	sitar
John	Lennon	guitar
Paul	McCartney	bass
Ringo	Starr	drums
Paul	Simon	guitar
John	Coltranee	sax

=

song	first	last	plays
Across the Universe	John	Lennon	guitar
Come Together	John	Lennon	guitar
Hello, Goodbye	Paul	McCartney	bass
Peggy Sue	Buddy	Holly	<NA>

```
left_join(songs2, artists2, by = c("first", "last"))
```

# dplyr::left\_join()

songs2

song	first	last
Across the Universe	John	Lennon
Come Together	John	Lennon
Hello, Goodbye	Paul	McCartney
Peggy Sue	Buddy	Holly

+

artists2

first	last	plays
George	Harrison	sitar
John	Lennon	guitar
Paul	McCartney	bass
Ringo	Starr	drums
Paul	Simon	guitar
John	Coltrane	sax

song	first	last	plays
Across the Universe	John	Lennon	guitar
Come Together	John	Lennon	guitar
Hello, Goodbye	Paul	McCartney	bass
Peggy Sue	Buddy	Holly	<NA>

```
left_join(songs2, artists2, by = c("first", "last"))
```



# left\_join()

songs

song	name
Across the Universe	John
Come Together	John
Hello, Goodbye	Paul
Peggy Sue	Buddy

artists

name	plays
George	sitar
John	guitar
Paul	bass
Ringo	drums

+

=

song	name	plays
Across the Universe	John	guitar
Come Together	John	guitar
Hello, Goodbye	Paul	bass
Peggy Sue	Buddy	<NA>

`left_join(songs, artists, by = "name")`

inner\_join()

songs

song	name
Across the Universe	John
Come Together	John
Hello, Goodbye	Paul
Peggy Sue	Buddy

+

artists

name	plays
George	sitar
John	guitar
Paul	bass
Ringo	drums

=

song	name	plays
Across the Universe	John	guitar
Come Together	John	guitar
Hello, Goodbye	Paul	bass

```
inner_join(songs, artists, by = "name")
```

# semi\_join()

songs

song	name
Across the Universe	John
Come Together	John
Hello, Goodbye	Paul
Peggy Sue	Buddy

+

artists

name	plays
George	sitar
John	guitar
Paul	bass
Ringo	drums

=

song	name
Across the Universe	John
Come Together	John
Hello, Goodbye	Paul

```
semi_join(songs, artists, by = "name")
```

# anti\_join()

songs

song	name
Across the Universe	John
Come Together	John
Hello, Goodbye	Paul
Peggy Sue	Buddy

+

artists

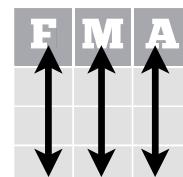
name	plays
George	sitar
John	guitar
Paul	bass
Ringo	drums

=

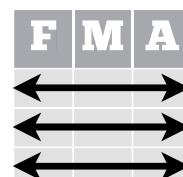
song	name
Peggy Sue	Buddy

```
anti_join(songs, artists, by = "name")
```

# 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

# **How to learn more**

## Data Wrangling with dplyr and tidyr Cheat Sheet

R Studio

### 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).

**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)
```

### 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.

**Reshaping Data - Change the layout of a data set**

**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).

### Subset Variables (Columns)

**dplyr::select(iris, Sepal.Width, Petal.Length, Species)**  
Select columns by name or helper function.

#### Helper functions for select - ?select

<b>select(iris, contains("."))</b>
Select columns whose name contains a character string.
<b>select(iris, ends_with("Length"))</b>
Select columns whose name ends with a character string.
<b>select(iris, everything())</b>
Select every column.
<b>select(iris, matches("t.*"))</b>
Select columns whose name matches a regular expression.
<b>select(iris, num_range("x", 1:5))</b>
Select columns named x1, x2, x3, x4, x5.
<b>select(iris, one_of(c("Species", "Genus")))</b>
Select columns whose names are in a group of names.
<b>select(iris, starts_with("Sepa"))</b>
Select columns whose name starts with a character string.
<b>select(iris, Sepal.Length:Petal.Width)</b>
Select all columns between Sepal.Length and Petal.Width (inclusive).
<b>select(iris, -Species)</b>
Select all columns except Species.

RStudio® is a trademark of RStudio, Inc. • All rights reserved • [info@rstudio.com](mailto:info@rstudio.com) • 844-448-1212 • [rstudio.com](http://rstudio.com)    [devtools::install\\_github\("rstudio/EDAWR"\) for data sets](http://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

<http://www.rstudio.com/resources/cheatsheets/>