transform and tidy

Copy the masters

- You are (eventually) going to reproduce a plot made by the folks at FiveThirtyEight

- This is a hard assignment for two reasons

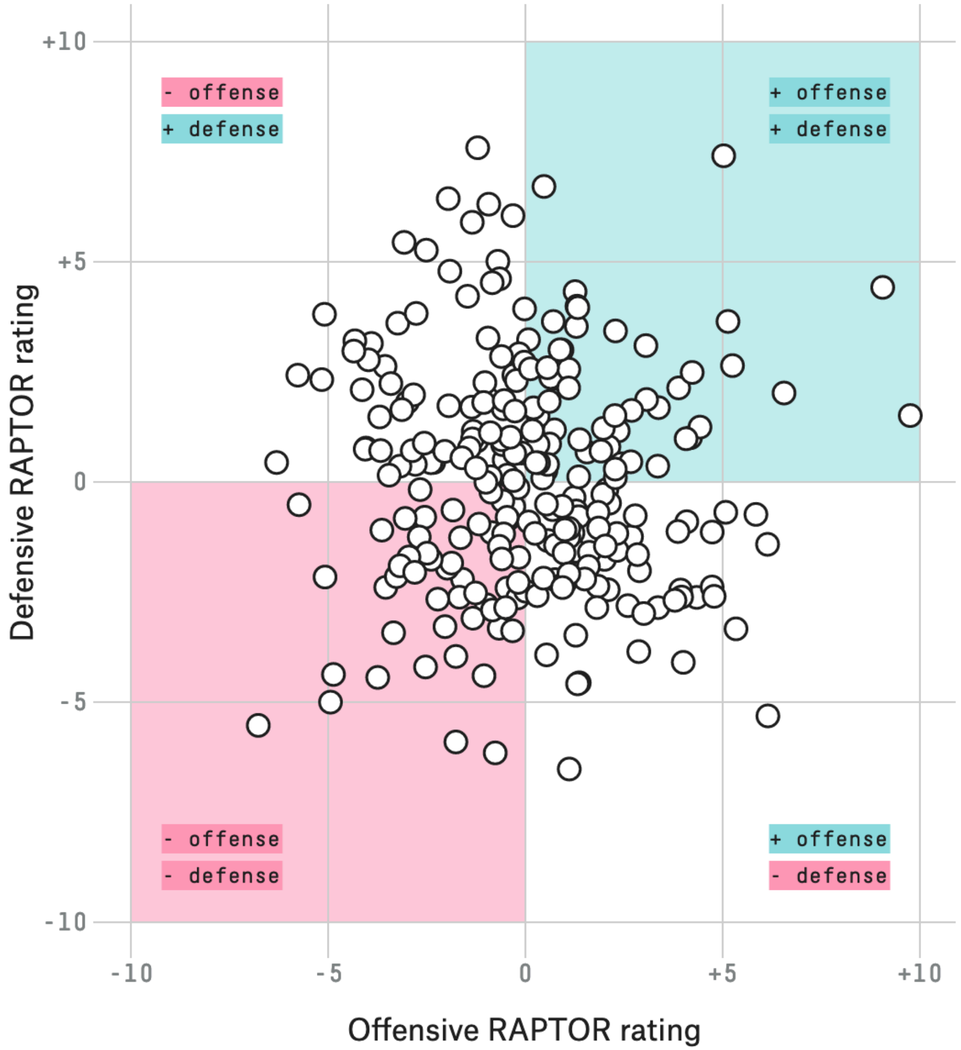
  1. There are lots of finicky settings for all sorts of plot details

  2. Data needs to be in the right format before you can easily plot it

"Once you have the data in the right format, the visualization is trivial"
- (paraphrasing) Hadley Wickham

Brainstorm:
what does the dataset that produced this graph look like?

- From “The Best NBA Players, According To RAPTOR”
- https://projects.fivethirtyeight.com/nba-player-ratings/



What does the dataset actually look like?

player_name	player_id	season	poss	mp	raptor_box_offense	raptor_box_defense
Alex Abrines	abrinal01	2017	2387	1135	0.745505030	-0.37293782
Alex Abrines	abrinal01	2018	2546	1244	0.317549034	-1.72532525
Alex Abrines	abrinal01	2019	1279	588	-3.215683211	1.07839855
Precious Achiuwa	achiupr01	2021	1581	749	-4.122965543	1.35927803
Quincy Acy	acyqu01	2014	1716	847	-1.716079145	0.13311503
Quincy Acy	acyqu01	2015	2517	1287	-2.014956419	-1.26843986
Quincy Acy	acyqu01	2016	1852	876	-0.008327617	0.34078337
Quincy Acy	acyqu01	2017	1169	558	-0.129001837	0.44433867
Quincy Acy	acyqu01	2018	2856	1359	-2.620031644	-0.80626910
Quincy Acy	acyqu01	2019	278	123	-5.741316630	3.25336528
Jaylen Adams	adamsja01	2019	952	428	-2.679919718	-4.59200168
Jaylen Adams	adamsja01	2020	43	21	2.023140328	4.19800473
Jaylen Adams	adamsja01	2021	41	18	-13.078112065	-3.29802113

Where every player in the NBA stands, according to RAPTOR

Offensive and defensive ratings for NBA players. Filter or search to find players.

CHOOSE SEASON(S)

☐ '13-'14 ☐ '14-'15 ☐ '15-'16 ☐ '16-'17 ☐ '17-'18
☐ '18-'19 ☐ '19-'20 ☐ '20-'21 ☑ '21-'22

CHOOSE A SEASON TYPE: Full season

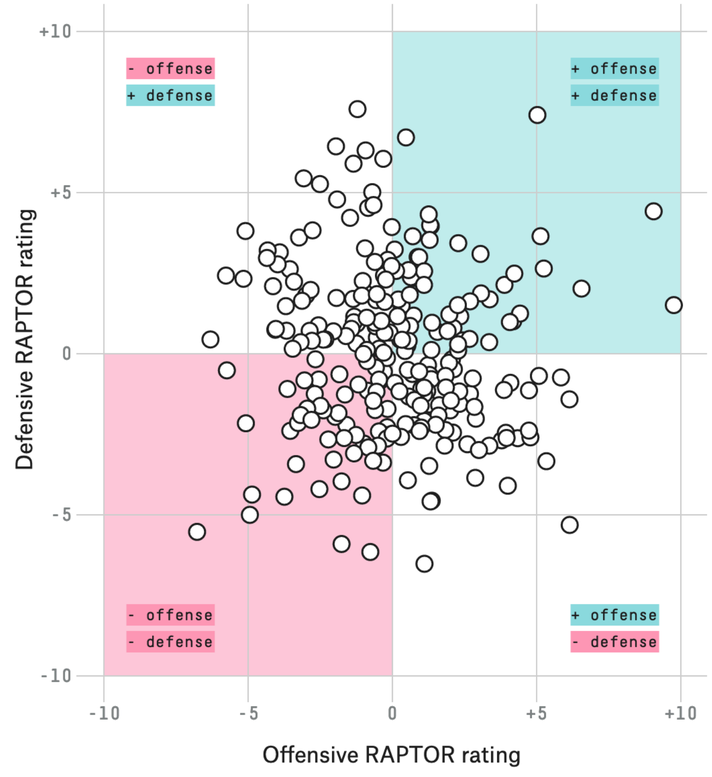
MINIMUM MINUTES PLAYED: 196

CHOOSE A TEAM: All teams

CHOOSE A POSITION: All positions

YEARS OF EXPERIENCE: Any number

SEARCH FOR A PLAYER



What's different? What needs to be changed before we can visualize this data?

Brainstorm:
what does the dataset that produced this graph look like?

- From “What Do Men Think It Means To Be A Man?”
- https://fivethirtyeight.com/features/what-do-men-think-it-means-to-be-a-man/

How often do you try to be the one who pays when on a date?

	AGE GROUP		
ANSWER	18 – 34	35 – 64	65 AND UP
Always	36%	55%	53%
Often	22	26	27
Sometimes	24	13	10
Rarely	3	1	2
Never	12	4	5

Among 1,615 adult men surveyed May 10-22, 2018. Two percent of respondents did not answer this question.

SOURCE: FIVETHIRTYEIGHT/DEATH, SEX & MONEY/SURVEYMONKEY

What does the dataset actually look like?

question	response	overall	age_18_34	age_35_64	age_65_over	white_yes
AMONG EMPLOYED WHO'VE HEARD OF METOO: As a …	Yes	0.34	NA	0.32	0.23	0.33
AMONG EMPLOYED WHO'VE HEARD OF METOO: As a …	No	0.65	NA	0.67	0.77	0.66
AMONG EMPLOYED WHO'VE HEARD OF METOO: As a …	No answer	0.01	NA	0.01	0.00	0.01
Do you typically feel as though you're expected to ma…	Yes	0.61	0.62	0.63	0.54	0.63
Do you typically feel as though you're expected to ma…	No	0.38	0.38	0.35	0.44	0.36
Do you typically feel as though you're expected to ma…	No answer	0.01	0.01	0.01	0.02	0.02
How often do you try to be the one who pays when o…	Always	0.49	0.36	0.55	0.53	0.52
How often do you try to be the one who pays when o…	Often	0.25	0.22	0.26	0.27	0.25
How often do you try to be the one who pays when o…	Sometimes	0.16	0.24	0.13	0.10	0.13
How often do you try to be the one who pays when o…	Rarely	0.02	0.03	0.01	0.02	0.01
How often do you try to be the one who pays when o…	Never	0.07	0.12	0.04	0.05	0.07
How often do you try to be the one who pays when o…	No answer	0.02	0.02	0.01	0.03	0.02
AMONG THOSE WHO TRY TO PAY ON FIRST DATE: Whi…	It's the right thing to do	0.66	NA	0.68	0.69	0.66
AMONG THOSE WHO TRY TO PAY ON FIRST DATE: Whi…	You asked the person out, so you feel obligated to pay	0.49	NA	0.49	0.52	0.49

What's different? What needs to be changed before we can visualize this data?

How often do you try to be the one who pays when on a date?

	AGE GROUP		
ANSWER	18 – 34	35 – 64	65 AND UP
Always	36%	55%	53%
Often	22	26	27
Sometimes	24	13	10
Rarely	3	1	2
Never	12	4	5

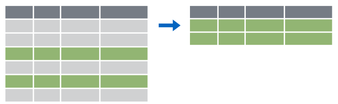
dplyr

dplyr

A package that transforms data.

dplyr implements a *grammar* for transforming tabular data.

Isolating data

	verb	direction	order	number
	**select()**	columns	✅	✅
	**filter()**	rows		✅
	**arrange()**	rows	✅	

select()

select()

Extract columns by name.

```
select(babynames, name, prop)
```

babynames

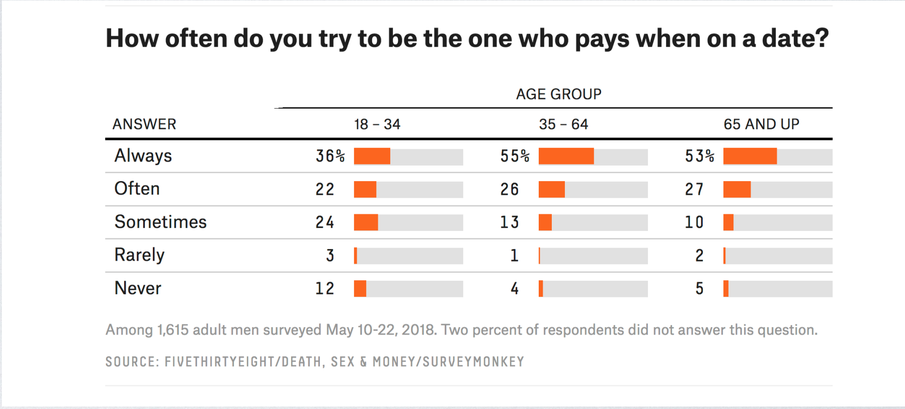
year	sex	name	n	prop
1880	M	John	9655	0.0815
1880	M	William	9532	0.0805
1880	M	James	5927	0.0501
1880	M	Charles	5348	0.0451
1880	M	Garrett	13	0.0001
1881	M	John	8769	0.081

name	prop
John	0.0815
William	0.0805
James	0.0501
Charles	0.0451
Garrett	0.0001
John	0.081

Let's do it!

recall

```
select(babynames, name, prop)
```



How often do you try to be the one who pays when on a date?

ANSWER	AGE GROUP: 18 – 34	35 – 64	65 AND UP
Always	36%	55%	53%
Often	22	26	27
Sometimes	24	13	10
Rarely	3	1	2
Never	12	4	5

Among 1,615 adult men surveyed May 10-22, 2018. Two percent of respondents did not answer this question.

SOURCE: FIVETHIRTYEIGHT/DEATH, SEX & MONEY/SURVEYMONKEY

Step 1:

Make a new RMarkdown document, and delete the sample material

Step 2:

Add the relevant packages to the startup chunk

Step 3:

Load the appropriate data

Step 4:

Think about what variables we want to select from the dataset, then select them!

filter()

filter()

Extract rows that meet logical criteria.

```
filter(.data, … )
```

data frame to transform

one or more logical tests (filter returns each row for which the test is TRUE)

filter()

Extract rows that meet logical criteria.

```
filter(babynames, name == "Amelia")
```

babynames

year	sex	name	n	prop
1880	F	Mary	7065	0.0724
1880	F	Anna	2604	0.0267
1880	F	Emma	2003	0.0205
1880	F	Elizabeth	1939	0.0199
1880	F	Amelia	221	0.00226
1880	F	Bertha	1320	0.0135

year	sex	name	n	prop
1880	F	Amelia	221	0.00226
1881	F	Amelia	225	0.00238
…	…	Amelia	…	…

filter()

Extract rows that meet logical criteria.

```
filter(babynames, name == "Amelia")
```

babynames

year	sex	name	n	prop
1880	F	Mary	7065	0.0724
1880	F	Anna	2604	0.0267
1880	F	Emma	2003	0.0205
1880	F	Elizabeth	1939	0.0199
1880	F	Amelia	221	0.00226
1880	F	Bertha	1320	0.0135

= sets
(returns nothing)

== tests if equal
(returns TRUE or FALSE)

Logical tests

?Comparison

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

Two common mistakes

1. Using `=` instead of `==`

```
filter(babynames, name = "Sea")
filter(babynames, name == "Sea")
```

2. Forgetting quotes

```
filter(babynames, name == Sea)
filter(babynames, name == "Sea")
```

filter()

Extract rows that meet *every* logical criteria.

```
filter(babynames, name == "Amelia", year == 1880)
```

babynames

year	sex	name	n	prop
1880	F	Mary	7065	0.0724
1880	F	Anna	2604	0.0267
1880	F	Emma	2003	0.0205
1880	F	Elizabeth	1939	0.0199
1880	F	Amelia	221	0.00226
1880	F	Bertha	1320	0.0135

year	sex	name	n	prop
1880	F	Amelia	221	0.00226

Boolean operators

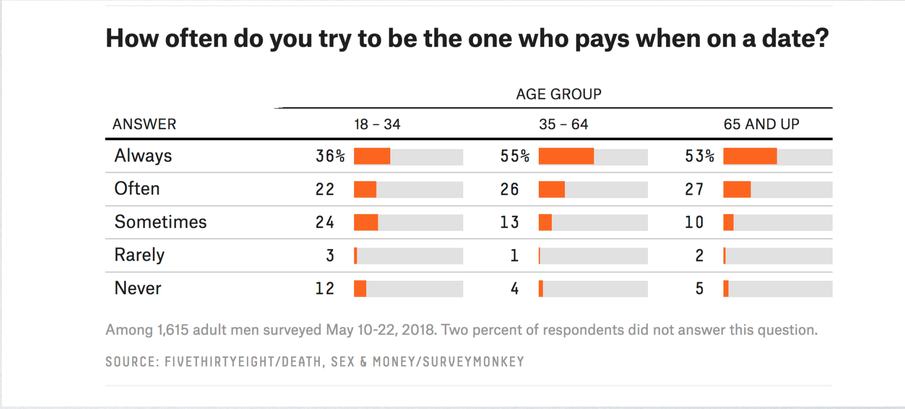
?base::Logic

`a & b`	and
`a | b`	or
`xor(a,b)`	exactly or
`!a`	not
`a %in% c(a, b)`	one of (in)

Let’s do it!

recall

```
filter(babynames, name == "Amelia")
```



How often do you try to be the one who pays when on a date?

ANSWER	AGE GROUP 18 – 34	35 – 64	65 AND UP
Always	36%	55%	53%
Often	22	26	27
Sometimes	24	13	10
Rarely	3	1	2
Never	12	4	5

Among 1,615 adult men surveyed May 10-22, 2018. Two percent of respondents did not answer this question.

SOURCE: FIVETHIRTYEIGHT/DEATH, SEX & MONEY/SURVEYMONKEY

What rows do we want to filter from the dataset?

Filter them!

arrange()

arrange()

Order rows from smallest to largest values.

```
arrange(.data, …)
```

data frame to transform

one or more columns to order by (additional columns will be used as tie breakers)

arrange()

Order rows from smallest to largest values.

```
arrange(babynames, n)
```

babynames

year	sex	name	n	prop
1880	F	Mary	7065	0.0724
1880	F	Anna	2604	0.0267
1880	F	Emma	2003	0.0205
1880	F	Elizabeth	1939	0.0199
1880	F	Amelia	221	0.00226
1880	F	Bertha	1320	0.0135

year	sex	name	n	prop
1880	F	Amelia	221	0.00226
1880	F	Bertha	1320	0.0135
1880	F	Elizabeth	1939	0.0199
1880	F	Emma	2003	0.0205
1880	F	Anna	2604	0.0267
1880	F	Mary	7065	0.0724

desc()

Changes ordering to largest to smallest.

```
arrange(babynames, desc(n))
```

babynames

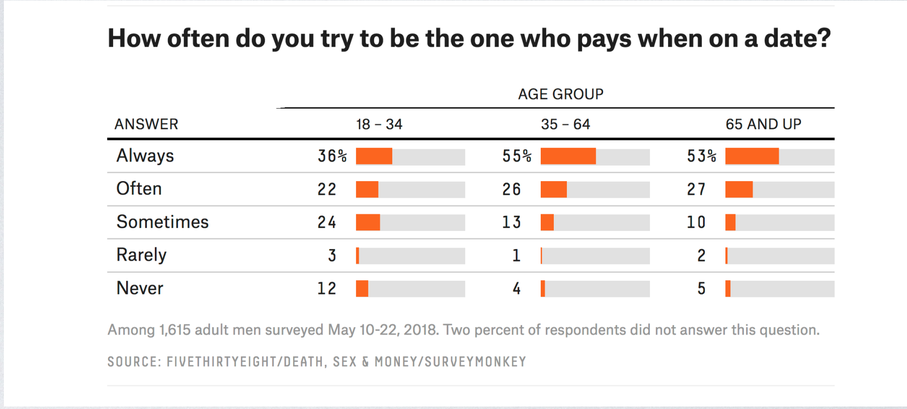
year	sex	name	n	prop
1880	F	Mary	7065	0.0724
1880	F	Anna	2604	0.0267
1880	F	Emma	2003	0.0205
1880	F	Elizabeth	1939	0.0199
1880	F	Amelia	221	0.00226
1880	F	Bertha	1320	0.0135

year	sex	name	n	prop
1880	F	Mary	7065	0.0724
1880	F	Anna	2604	0.0267
1880	F	Emma	2003	0.0205
1880	F	Elizabeth	1939	0.0199
1880	F	Bertha	1320	0.0135
1880	F	Amelia	221	0.00226

Let's do it!

recall

```
arrange(babynames, desc(n))
```



Not super relevant for this dataset, but try an arrange!

%>%

Steps

```
boys_2015 <- filter(babynames, year == 2015, sex == "M")
boys_2015 <- select(boys_2015, name, n)
boys_2015 <- arrange(boys_2015, desc(n))
boys_2015
```

1. Filter babynames to just boys born in 2015
2. Select the name and n columns from the result
3. Arrange those columns so that the most popular names appear near the top.

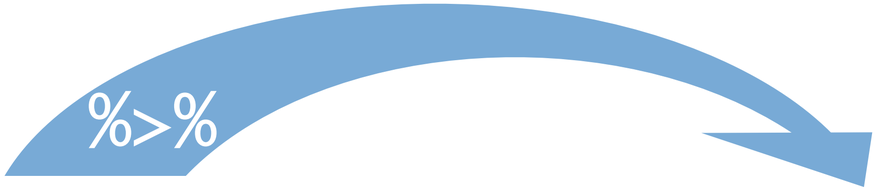
Steps

```
boys_2015 <- filter(babynames, year == 2015, sex == "M")
boys_2015 <- select(boys_2015, name, n)
boys_2015 <- arrange(boys_2015, desc(n))
boys_2015
```

Steps

```
arrange(select(filter(babynames, year == 2015,
  sex == "M"), name, n), desc(n))
```

The pipe operator %>%




```
babynames          filter(_____, n == 99680)
```

Passes result on left into first argument of function on right.
So, for example, these do the same thing.

```
filter(babynames, n == 99680)
babynames %>% filter(n == 99680)
```

Pipes

```
babynames
boys_2015 <- filter(babynames, year == 2015, sex == "M")
boys_2015 <- select(boys_2015, name, n)
boys_2015 <- arrange(boys_2015, desc(n))
boys_2015
```

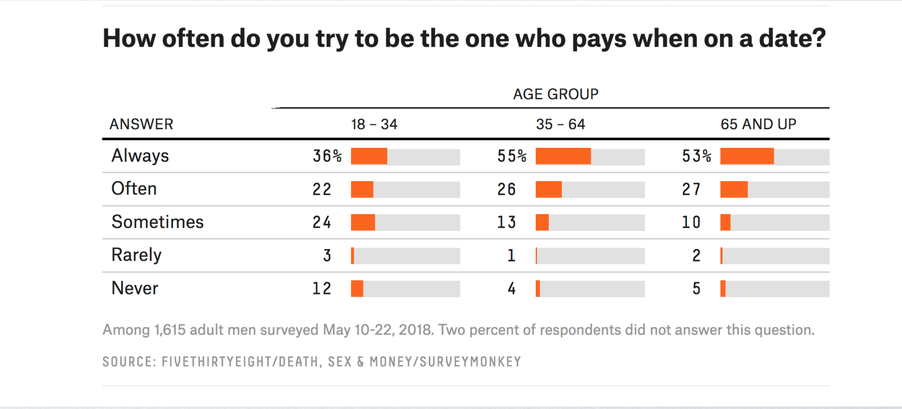
```
babynames %>%
  filter(year == 2015, sex == "M") %>%
  select(name, n) %>%
  arrange(desc(n))
```

Say "then"

Let’s do it!

recall

```
babynames %>%
  filter(year == 2015, sex == "M") %>%
  select(name, n) %>%
  arrange(desc(n))
```



How often do you try to be the one who pays when on a date?

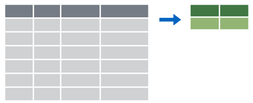
ANSWER	AGE GROUP 18 – 34	35 – 64	65 AND UP
Always	36%	55%	53%
Often	22	26	27
Sometimes	24	13	10
Rarely	3	1	2
Never	12	4	5

Among 1,615 adult men surveyed May 10-22, 2018. Two percent of respondents did not answer this question.

SOURCE: FIVETHIRTYEIGHT/DEATH, SEX & MONEY/SURVEYMONKEY

Now, put the steps you’ve done together into one long data pipeline

Deriving information

	verb	rows	existing columns
	**summarise()**	reduced	dropped
	**mutate()**	unchanged	kept

summarise()

/

summarize()

British and American spellings recognised/recognized!
Apologies for my (inevitable) inconsistency

summarise()

Compute table of summaries.

```
babynames %>% summarise(total = sum(n), max = max(n))
```

babynames

year	sex	name	n	prop
1880	M	John	9655	0.0815
1880	M	William	9532	0.0805
1880	M	James	5927	0.0501
1880	M	Charles	5348	0.0451
1880	M	Garrett	13	0.0001
1881	M	John	8769	0.081

total	max
127538	99680

Grouping cases

group_by()

Groups cases by common values of one or more columns.

```
babynames %>%
  group_by(sex)
```

```
Source: local data frame [1,825,433 x 5]
Groups: sex [2]

   year   sex       name     n        prop
  <dbl> <chr>      <chr> <int>       <dbl>
1  1880     F       Mary  7065 0.07238359
```

group_by()

Groups cases by common values.

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

sex	total
F	167070477
M	170064949

mutate()

mutate()

Create new columns.

```
babynames %>%
  mutate(percent = round(prop*100, 2))
```

babynames

year	sex	name	n	prop
1880	M	John	9655	0.0815
1880	M	William	9532	0.0805
1880	M	James	5927	0.0501
1880	M	Charles	5348	0.0451
1880	M	Garrett	13	0.0001
1881	M	John	8769	0.081

year	sex	name	n	prop	percent
1880	M	John	9655	0.0815	8.15
1880	M	William	9532	0.0805	8.05
1880	M	James	5927	0.0501	5.01
1880	M	Charles	5348	0.0451	4.51
1880	M	Garrett	13	0.0001	0.01
1881	M	John	8769	0.081	8.1

mutate()

Create new columns.

```
babynames %>%
  mutate(percent = round(prop*100, 2), nper = round(percent))
```

babynames

year	sex	name	n	prop
1880	M	John	9655	0.0815
1880	M	William	9532	0.0805
1880	M	James	5927	0.0501
1880	M	Charles	5348	0.0451
1880	M	Garrett	13	0.0001
1881	M	John	8769	0.081

year	sex	name	n	prop	percent	nper
1880	M	John	9655	0.0815	8.15	8
1880	M	William	9532	0.0805	8.05	8
1880	M	James	5927	0.0501	5.01	5
1880	M	Charles	5348	0.0451	4.51	5
1880	M	Garrett	13	0.0001	0.01	0
1881	M	John	8769	0.081	8.1	8

group_by()

Groups cases by common values of one or more columns.

```
babynames %>%
  group_by(sex)
```

```
Source: local data frame [1,825,433 x 5]
Groups: sex [2]

   year   sex       name     n        prop
  <dbl> <chr>      <chr> <int>       <dbl>
1  1880     F       Mary  7065 0.07238359
```

group_by()

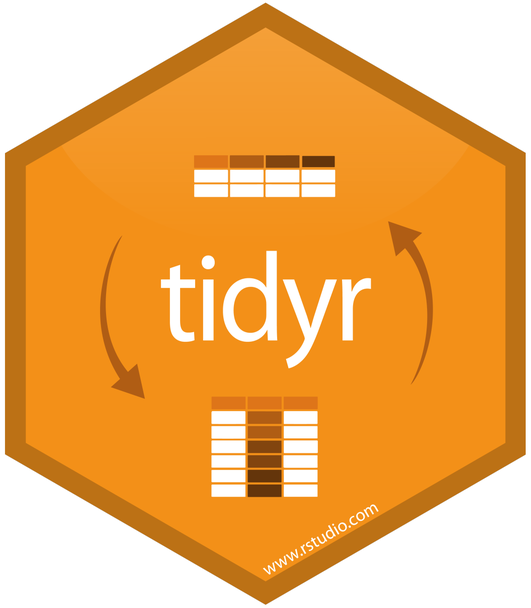
Groups cases by common values.

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

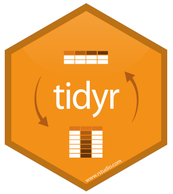
sex	total
F	167070477
M	170064949

tidyr

tidyr



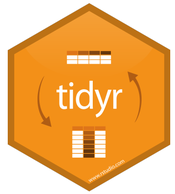
A package that reshapes the layout of tabular data.



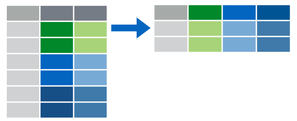
Tidy data

A data set is **tidy** iff:

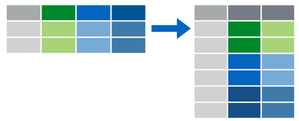
1. Each **variable** is in its own **column**
2. Each **case** is in its own **row**
3. Each **value** is in its own **cell**



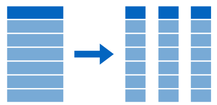
tidyr verbs



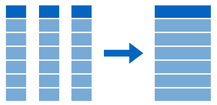
Make data wider with **pivot_wider()**



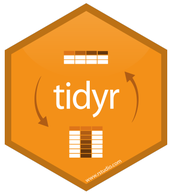
Make data longer with **pivot_longer()**



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



Unite columns with **unite()**



pivot_longer()

Consider

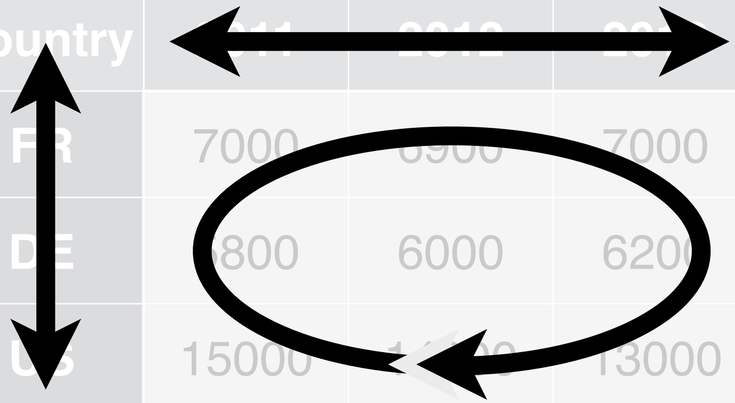
What are the variables in cases?

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

Consider

What are the variables in cases?

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



- Country
- Year
- Count

Your Turn

On a sheet of paper, draw how the cases data set would look if it had the same values grouped into three columns: *country*, *year*, *n*

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

pivot_longer()

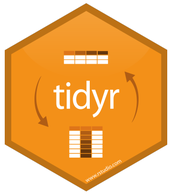
```
cases %>%
    pivot_longer(2:4, names_to = "year", values_to = "n")
```

data frame to reshape

numeric indexes of columns to collapse (or names)

name of the new variable created from column names (a character string)

name of the new variable created from cell values (a character string)

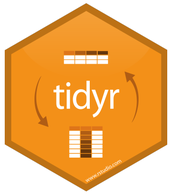


pivot_longer()

```
cases %>%
    pivot_longer(2:4, names_to = "year", values_to = "count")
```

numeric indexes

	2	3	4
Country <chr>	**2011** <dbl>	**2012** <dbl>	**2013** <dbl>
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

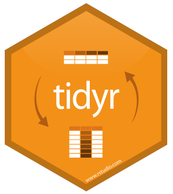


pivot_longer()

names

```
cases %>%
    pivot_longer(cols=c("2011","2012","2013"),
                 names_to = "year", values_to = "count")
```

	2011	2012	2013
Country	**2011**	**2012**	**2013**
<chr>	<dbl>	<dbl>	<dbl>
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

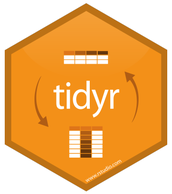


pivot_longer()

```
cases %>%
    pivot_longer(-Country, names_to = "year", values_to = "count")
```

Everything except…

	Not Country	Not Country	Not Country
Country <chr>	**2011** <dbl>	**2012** <dbl>	**2013** <dbl>
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000



pivot_wider()

Consider

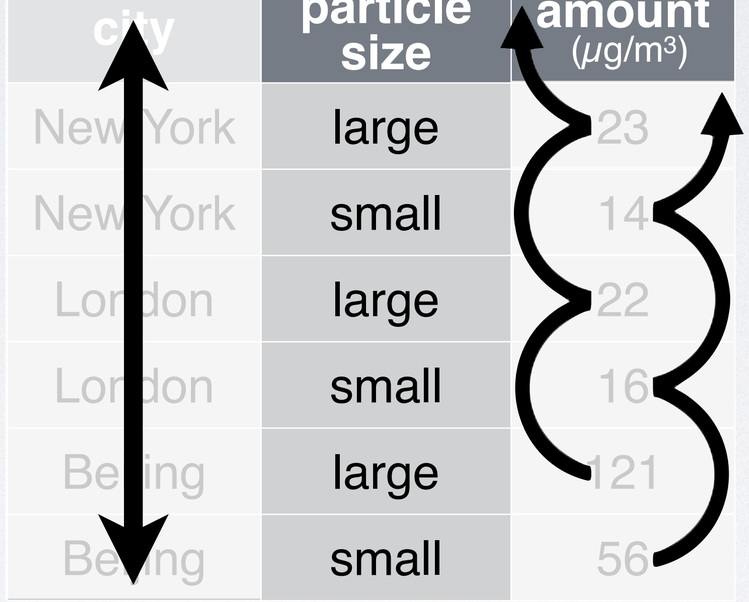
What are the variables in pollution?

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

Consider

What are the variables in pollution?

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



- City
- Amount of large particulate
- Amount of small particulate

Your Turn

On a sheet of paper, draw how this data set would look if it had the same values grouped into three columns: *city*, *large*, *small*

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

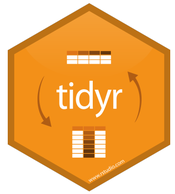
pivot_wider()

```
pollution %>% pivot_wider(names_from = size, values_from = amount)
```

data frame to reshape

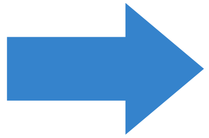
new columns from names in this variable

values for these columns from this variable

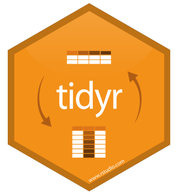


```
pollution %>% pivot_wider(names_from = size,
                          values_from = amount)
```

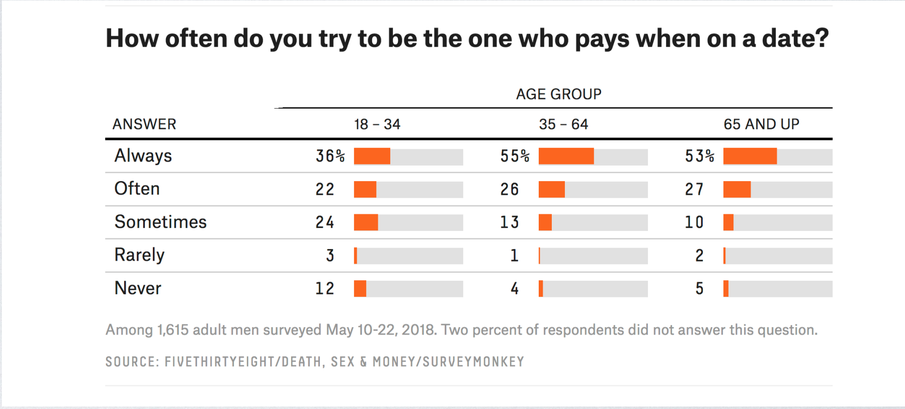
```
      city  size amount
1 New York large     23
2 New York small     14
3   London large     22
4   London small     16
5  Beijing large    121
6  Beijing small     56
```



```
     city large small
1  Beijing   121    56
2   London    22    16
3 New York    23    14
```



Let’s do it!



How often do you try to be the one who pays when on a date?

ANSWER	AGE GROUP 18 – 34	35 – 64	65 AND UP
Always	36%	55%	53%
Often	22	26	27
Sometimes	24	13	10
Rarely	3	1	2
Never	12	4	5

Among 1,615 adult men surveyed May 10-22, 2018. Two percent of respondents did not answer this question.

SOURCE: FIVETHIRTYEIGHT/DEATH, SEX & MONEY/SURVEYMONKEY

recall

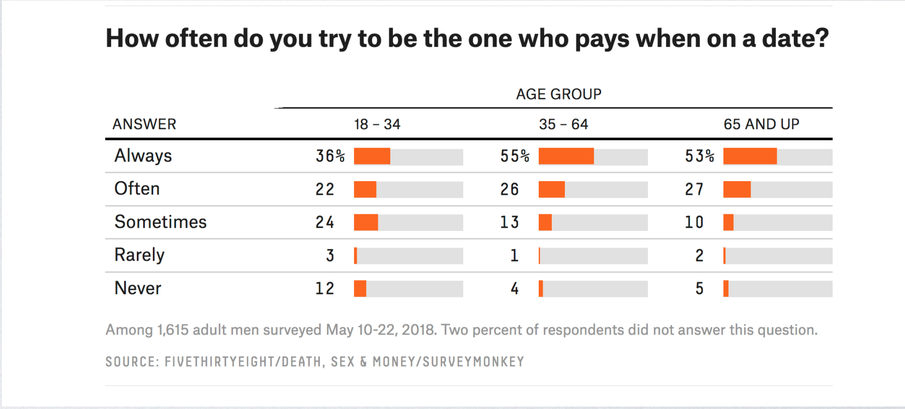
```
cases %>%
    pivot_longer(-Country, names_to = "year", values_to = "count")
pollution %>%
   pivot_wider(names_from = size, values_from = amount)
```

Imagine the shape of the data we want

Pivot your data into the appropriate shape

Let’s do it!

… Make the plot!



How often do you try to be the one who pays when on a date?

ANSWER	AGE GROUP: 18 – 34	35 – 64	65 AND UP
Always	36%	55%	53%
Often	22	26	27
Sometimes	24	13	10
Rarely	3	1	2
Never	12	4	5

Among 1,615 adult men surveyed May 10-22, 2018. Two percent of respondents did not answer this question.

SOURCE: FIVETHIRTYEIGHT/DEATH, SEX & MONEY/SURVEYMONKEY