The United Nations Voting Dataset

CASE STUDY: EXPLORATORY DATA ANALYSIS IN R

Dave Robinson
Chief Data Scientist, DataCamp

UN Voting Dataset

rcid	session	vote	ccode
46	2	1	2
46	2	1	20
46	2	9	31
46	2	1	40
46	2	1	41
46	2	1	42
46	2	1	51
46	2	9	52
46	2	9	53

[1] Erik Voeten, "Data and Analyses of Voting in the UN General Assembly"

UN Voting Dataset

rcid	session	vote	ccode
46	2	1	2
46	2	1	20
46	2	9	31
46	2	1	40
46	2	1	41
46	2	1	42
46	2	1	51
46	2	9	52
46	2	9	53

Each row has a country-vote pair

[1] Erik Voeten, "Data and Analyses of Voting in the UN General Assembly"

UN Voting Dataset

rcid	session	vote	ccode
46	2	1	2
46	2	1	20
46	2	9	31
46	2	1	40
46	2	1	41
46	2	1	42
46	2	1	51
46	2	9	52
46	2	9	53

Each row has a country-vote pair

rcid = "Roll call ID"

[1] Erik Voeten, "Data and Analyses of Voting in the UN General Assembly"

UN Voting Dataset

rcid	session	vote	ccode	
46	2	1	2	Each row has a country-vote pair
46	2	1	20	
46	2	9	31	**rcid** = Roll call ID
46	2	1	40	
46	2	1	41	**session** = Session year
46	2	1	42	
46	2	1	51	
46	2	9	52	
46	2	9	53	

[1] Erik Voeten, "Data and Analyses of Voting in the UN General Assembly"

UN Voting Dataset

rcid	session	vote	ccode
46	2	1	2
46	2	1	20
46	2	9	31
46	2	1	40
46	2	1	41
46	2	1	42
46	2	1	51
46	2	9	52
46	2	9	53

Each row has a country-vote pair

rcid = Roll call ID

session = Session year

vote = Vote code

[1] Erik Voeten, "Data and Analyses of Voting in the UN General Assembly"

UN Voting Dataset

rcid	session	vote	ccode	
46	2	1	2	Each row has a country-vote pair
46	2	1	20	
46	2	9	31	**rcid** = Roll call ID
46	2	1	40	
46	2	1	41	**session** = Session year
46	2	1	42	
46	2	1	51	**vote** = Vote code
46	2	9	52	
46	2	9	53	**ccode** = Country code

[1] Erik Voeten, "Data and Analyses of Voting in the UN General Assembly"

Votes in dplyr

```r
# Load dplyr package
library(dplyr)
votes
```

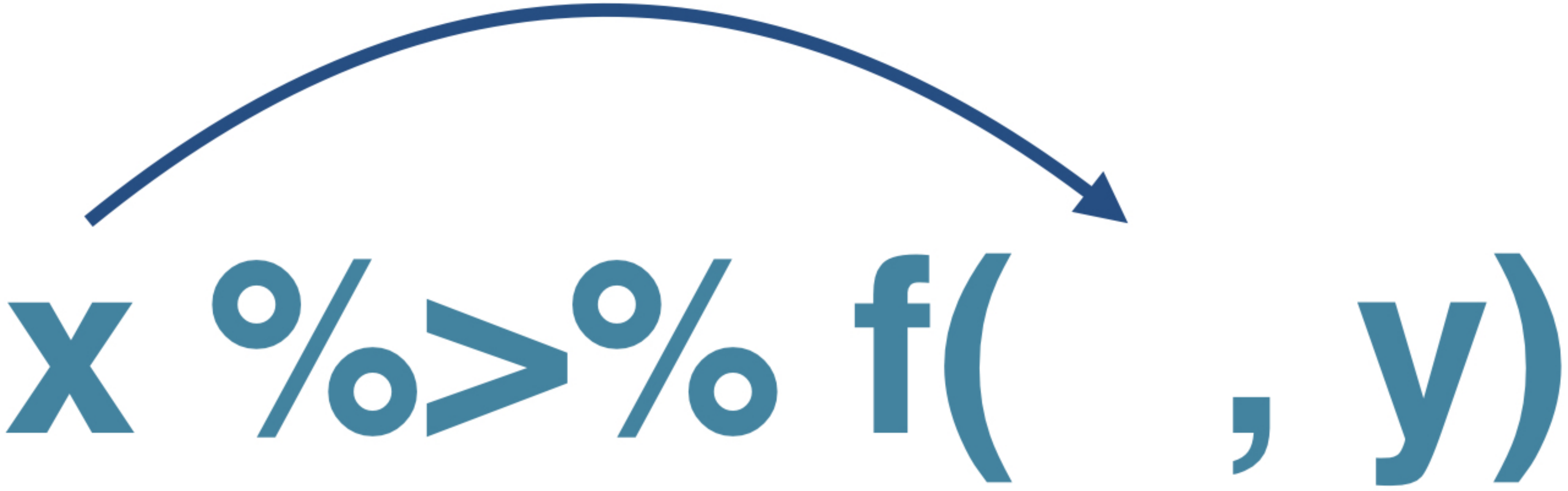
```
# A tibble: 508,929 × 4
    rcid session  vote ccode
   <dbl>   <dbl> <dbl> <int>
1     46       2     1     2
2     46       2     1    20
3     46       2     9    31
4     46       2     1    40
5     46       2     1    41
6     46       2     1    42
7     46       2     9    51
8     46       2     9    52
9     46       2     9    53
10    46       2     9    54
# ... with 508,919 more rows
```

Variable names

The pipe operator

```
%>%
```

The pipe operator

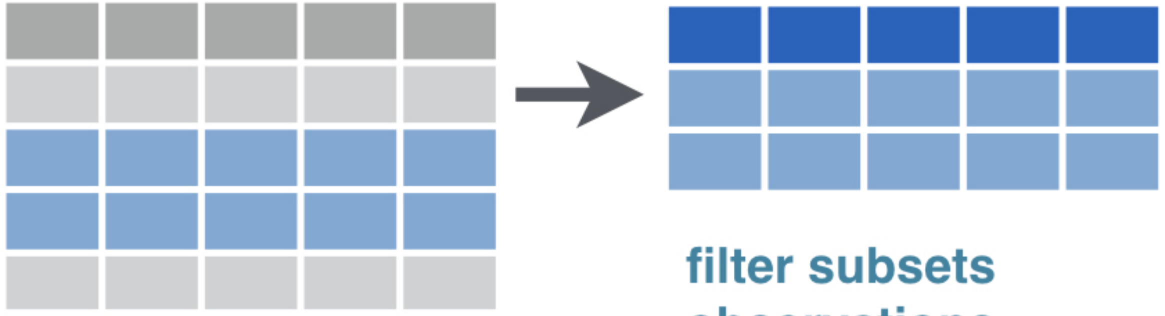


x %>% f(, y)

f(x, y)

dplyr verbs

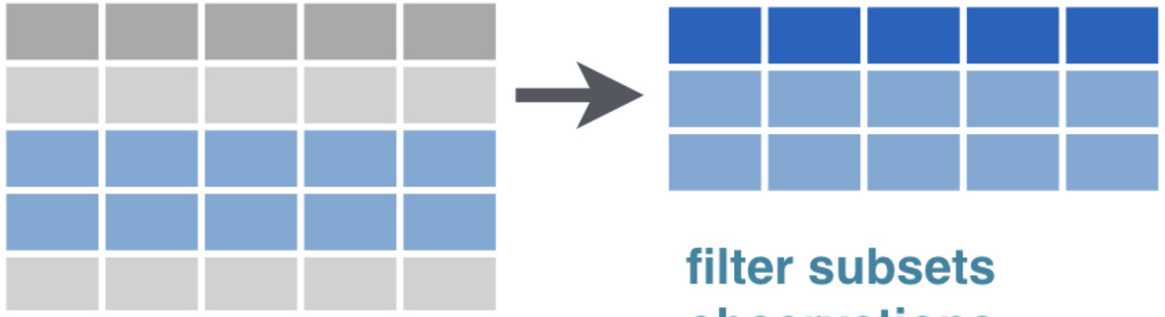
filter()



filter subsets observations

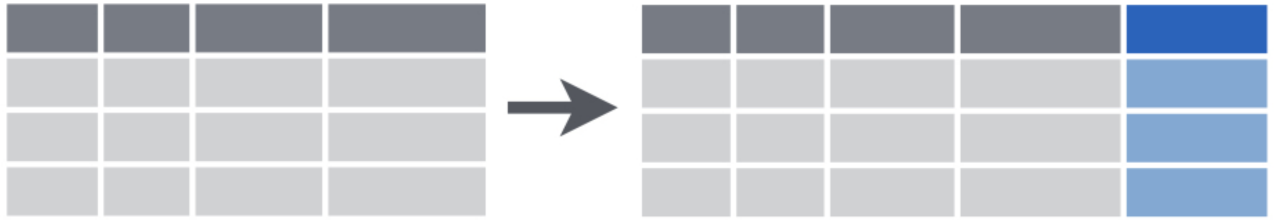
dplyr verbs

filter()



filter subsets observations

mutate()



mutate adds or changes variables

Original data

```
votes
```

```
# A tibble: 508,929 × 4
    rcid session  vote ccode
   <dbl>   <dbl> <dbl> <int>
1     46       2     1     2
2     46       2     1    20
3     46       2     9    31
4     46       2     1    40
5     46       2     1    41
6     46       2     1    42
7     46       2     9    51
8     46       2     9    52
9     46       2     9    53
10    46       2     9    54
# ... with 508,919 more rows
```

```
1 = Yes
2 = Abstain
3 = No
8 = Not present
9 = Not a member
```

dplyr verbs: filter

`filter` keeps observations based on a condition

```r
votes %>%
  filter(vote <= 3)
```

```
# A tibble: 353,547 × 4
    rcid session  vote ccode
   <dbl>   <dbl> <dbl> <int>
1     46       2     1     2
2     46       2     1    20
3     46       2     1    40
4     46       2     1    41
5     46       2     1    42
6     46       2     1    70
7     46       2     1    90
8     46       2     1    91
9     46       2     1    92
10    46       2     1    93
# ... with 508,919 more rows
```

dplyr verbs: mutate

`mutate` adds an additional variable

```
votes %>%
  mutate(year = session + 1945)
```

```
# A tibble: 508,929 × 5
    rcid session  vote ccode  year
   <dbl>   <dbl> <dbl> <int> <dbl>
1     46       2     1     2  1947
2     46       2     1    20  1947
3     46       2     9    31  1947
4     46       2     1    40  1947
5     46       2     1    41  1947
```

Chaining operations in data cleaning

```
data %>%
  filter(...) %>%
  mutate(...)
```

Let's practice!

CASE STUDY: EXPLORATORY DATA ANALYSIS IN R

Grouping and summarizing

CASE STUDY: EXPLORATORY DATA ANALYSIS IN R

Dave Robinson
Chief Data Scientist, DataCamp

Processed votes

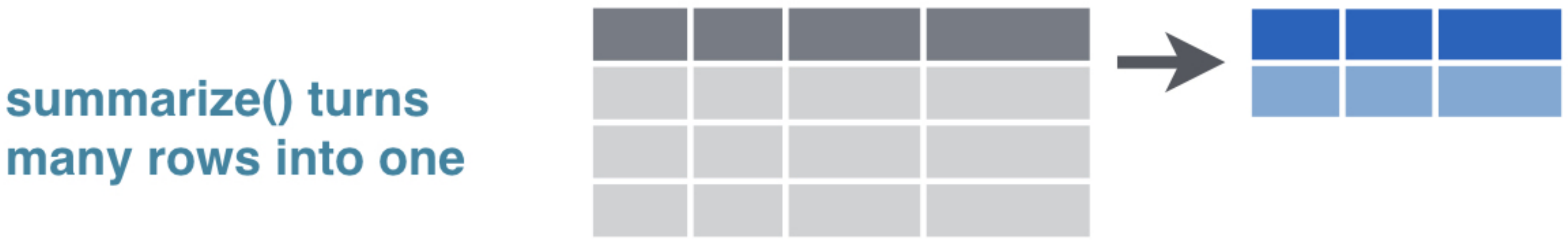
```
votes_processed
```

```
# A tibble: 353,547 × 6
    rcid session  vote ccode  year             country
   <dbl>   <dbl> <dbl> <int> <dbl>              <chr>
1     46       2     1     2  1947      United States
2     46       2     1    20  1947             Canada
3     46       2     1    40  1947               Cuba
4     46       2     1    41  1947              Haiti
5     46       2     1    42  1947 Dominican Republic
6     46       2     1    70  1947             Mexico
7     46       2     1    90  1947          Guatemala
8     46       2     1    91  1947           Honduras
9     46       2     1    92  1947        El Salvador
10    46       2     1    93  1947          Nicaragua
# ... with 353,537 more rows
```

Using “% of Yes votes” as a summary

dplyr verb: summarize

`summarize()` turns many rows into one



summarize() turns many rows into one

dplyr verbs: summarize

```
votes_processed %>%
  summarize(total = n())
```

```
# A tibble: 1 × 1
   total
   <int>
1 353547
```

dplyr verbs: summarize

```
votes_processed %>%
  summarize(total = n(),
            percent_yes = mean(vote == 1))
```

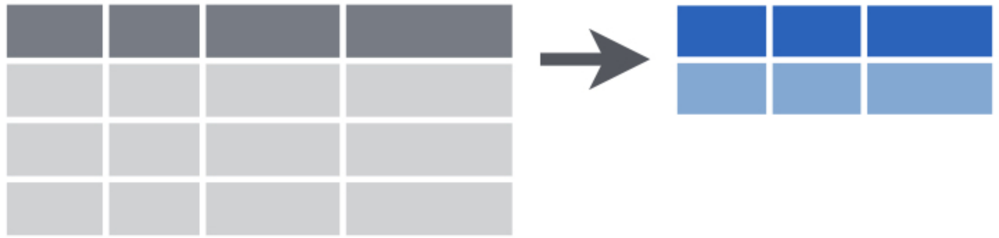
```
# A tibble: 1 × 2
   total percent_yes
   <int>       <dbl>
1 353547   0.7999248
```

- `mean(vote == 1)` is a way of calculating “percent of vote equal to 1”

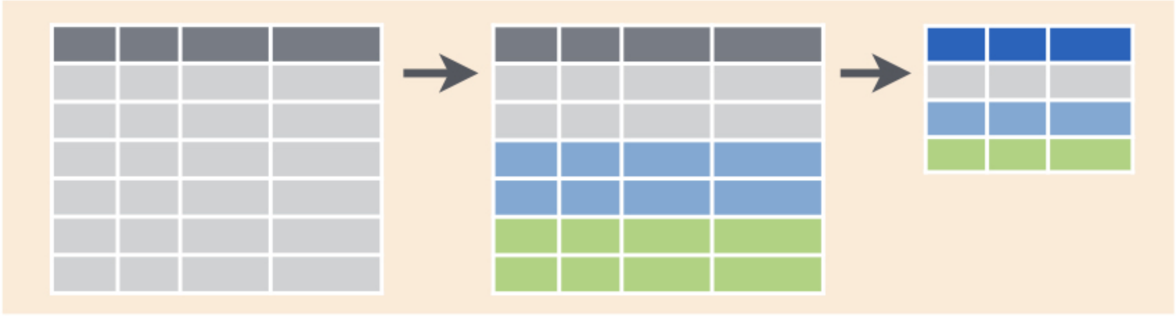
dplyr verb: group_by

- `summarize()` turns many rows into one
- `group_by()` before `summarize()` turns groups into one row each

summarize() turns many rows into one



group_by() before summarize() turns groups into one row each



dplyr verbs: group_by

```
votes_processed %>%
  group_by(year) %>%
  summarize(total = n(),
            percent_yes = mean(vote == 1))
```

```
# A tibble: 34 × 3
    year total percent_yes
   <dbl> <int>       <dbl>
1   1947  2039   0.5693968
2   1949  3469   0.4375901
3   1951  1434   0.5850767
4   1953  1537   0.6317502
5   1955  2169   0.6947902
6   1957  2708   0.6085672
7   1959  4326   0.5880721
8   1961  7482   0.5729751
9   1963  3308   0.7294438
10  1965  4382   0.7078959
# ... with 24 more rows
```

Let's practice!

CASE STUDY: EXPLORATORY DATA ANALYSIS IN R

Sorting and filtering summarized data

CASE STUDY: EXPLORATORY DATA ANALYSIS IN R

Dave Robinson

Chief Data Scientist, DataCamp

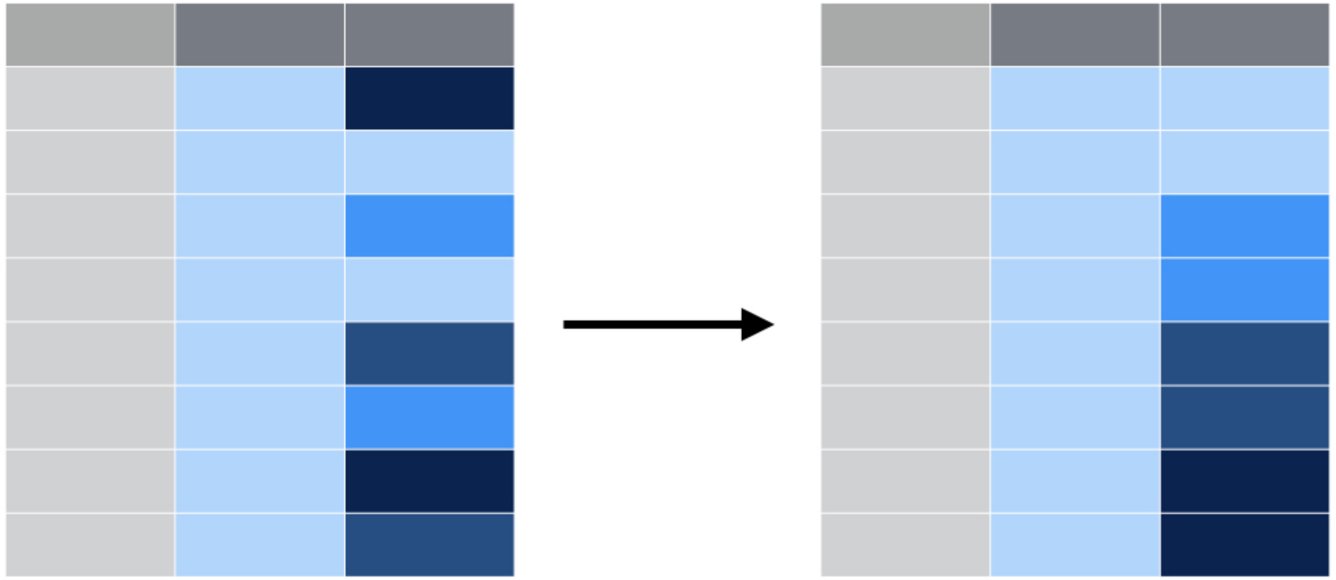
by_country dataset

```
by_country
```

```
# A tibble: 200 × 3
               country total percent_yes
                 <chr> <int>       <dbl>
1          Afghanistan  2373   0.8592499
2              Albania  1695   0.7174041
3              Algeria  2213   0.8992318
4              Andorra   719   0.6383866
5               Angola  1431   0.9238295
6  Antigua and Barbuda  1302   0.9124424
7            Argentina  2553   0.7677242
8              Armenia   758   0.7467018
9            Australia  2575   0.5565049
10             Austria  2389   0.6224362
# ... with 190 more rows
```

dplyr verb: arrange()

arrange() sorts a table based on a variable

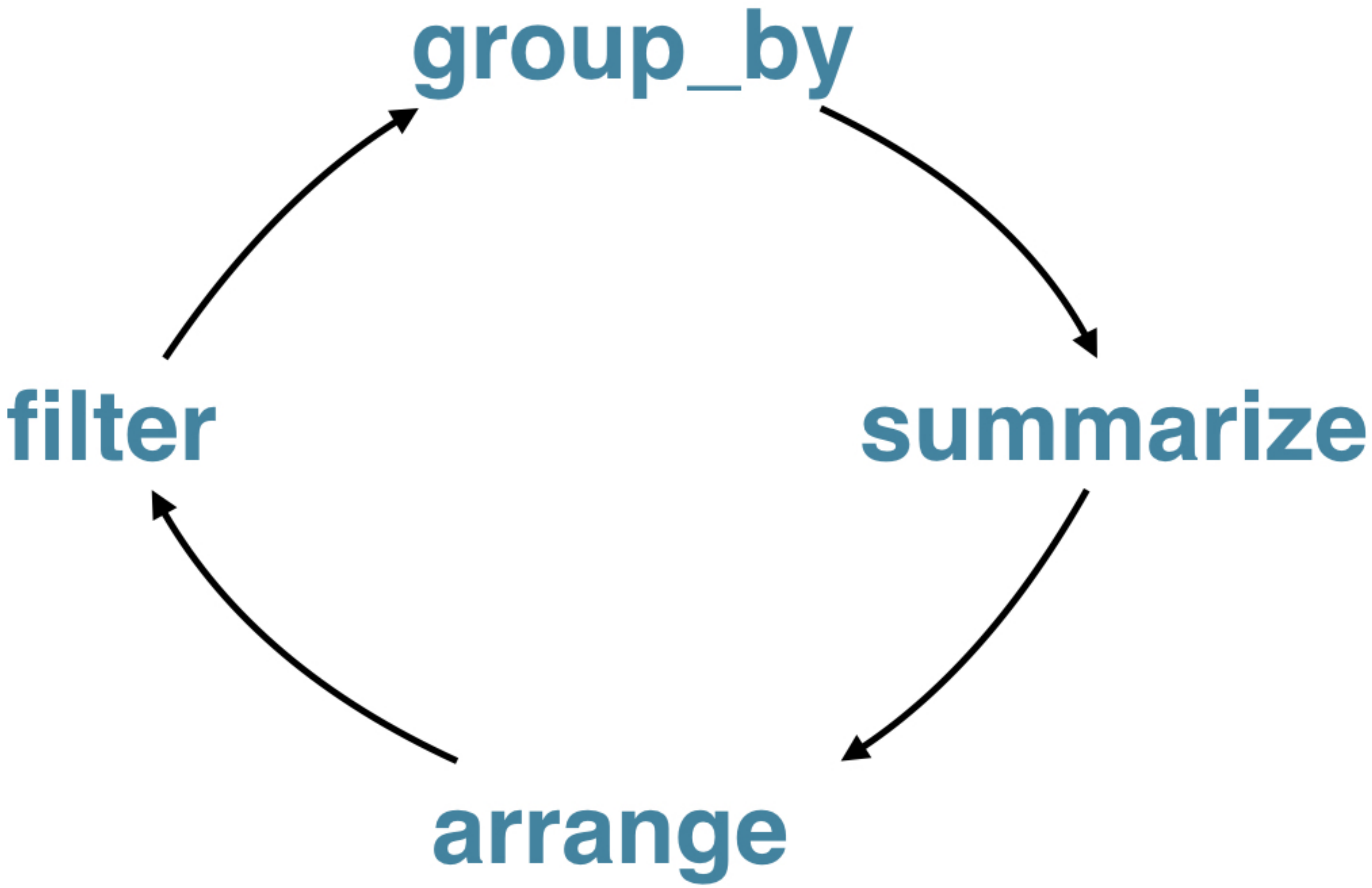


arrange()

```
by_country %>%
  arrange(percent_yes)
```

```
# A tibble: 200 × 3
                           country total percent_yes
                             <chr> <int>       <dbl>
1                         Zanzibar     2   0.0000000
2                    United States  2568   0.2694704
3                            Palau   369   0.3387534
4                           Israel  2380   0.3407563
5       Federal Republic of Germany  1075   0.3972093
6                   United Kingdom  2558   0.4167318
7                           France  2527   0.4265928
8   Micronesia, Federated States of   724   0.4419890
9                 Marshall Islands   757   0.4914135
10                         Belgium  2568   0.4922118
# ... with 190 more rows
```

Transforming tidy data



Let's practice!

CASE STUDY: EXPLORATORY DATA ANALYSIS IN R