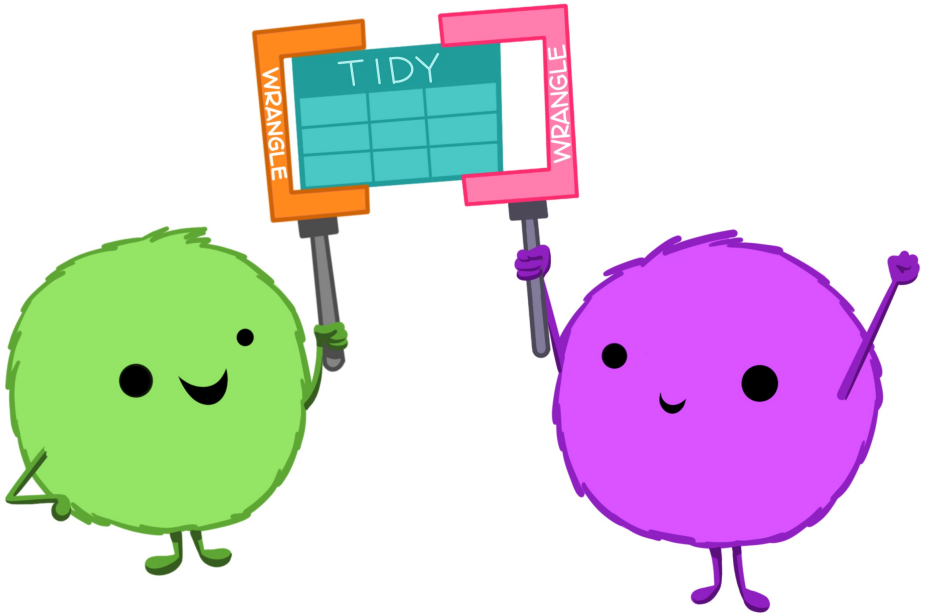
Deriving information from data

13 September 2020

Modern Research Methods



Artwork by @allison_horst

Extracting subset of rows using row order

- `head(df, n)` – extract first n rows of dataframe
- `tail(df, n)` – extract last n rows of dataframe
- `slice(df, index)` – extract rows based on indices of dataframe

Raw data

- Data is typically read into R from a local file
- Lots of different file formats (.csv, .txt, .tsv, .xlsx)
- We'll work mostly with a plain text format, called "comma separated value" (.csv)
- Each observation is separated by a comma
- Non-proprietory

lewis_2018_exp1.csv

```
exp,subids,trial_num,category,condition,proportion_basic_level_responses
1,1,9,vehicles,three_subordinate,0
1,2,9,animals,three_basic,1
1,3,9,animals,three_superordinate,1
1,4,9,vehicles,three_superordinate,1
1,5,9,animals,three_superordinate,1
1,6,9,vegetables,three_subordinate,0
1,7,9,vegetables,three_basic,0.5
1,8,9,animals,three_basic,1
```

```
> lf_data
# A tibble: 600 x 6
     exp subids trial_num category   condition           proportion_basic_level_responses
   <dbl>  <dbl>     <dbl> <chr>      <chr>                                          <dbl>
 1     1      1         9 vehicles   three_subordinate                                  0
 2     1      2         9 animals    three_basic                                        1
 3     1      3         9 animals    three_superordinate                                1
 4     1      4         9 vehicles   three_superordinate                                1
 5     1      5         9 animals    three_superordinate                                1
 6     1      6         9 vegetables three_subordinate                                  0
 7     1      7         9 vegetables three_basic                                        0.5
 8     1      8         9 animals    three_basic                                        1
 9     1      9         9 animals    three_superordinate                                1
10     1     10         9 animals    three_subordinate                                  0
# … with 590 more rows
```

Reading data into R

function	reads
read_csv()	**Comma separated values**
read_csv2()	Semi-colon separated values
read_delim()	General delimited files
read_fwf()	Fixed width files
read_table()	Space separated
read_tsv()	Tab delimited values

- Reads data frame into special tidyverse type of dataframe, “tibble”
- All of these have analogs for writing data, e.g., `write_csv()`

Missing data in R

- Missing data in R: NA ("not available")
- is.na(x) – Boolean (gives TRUE or FALSE), testing whether x has value NA
- Why might data be missing?
  - Participant didn't respond (e.g. got fussy, got bored, ended experiment)
  - Experimenter error
  - Not collected because variable not applicable
  - Lots of others…

Code style with the pipe

- Space before pipe and new line after
- Be indented two spaces after the first line where you name the dataframe

```
females_by_year<-babynames%>% arrange(year)%>%select(name, prop)%>%filter(sex == F)
```

```
females_by_year <- babynames %>%
  arrange(year) %>%
  select(name, prop) %>%
  filter(sex == F)
```

Working with tidy data

Isolating information

(in lab on Friday)

select() - extract **variables**

filter() - extract **cases**

arrange() - reorder **cases**

Deriving information

summarise() - summarise **variables**

group_by() - group **cases**

mutate() - create new **variables**

Data from an experiment where you randomly assigned 20 people to attend MRM at 6am and 20 people to attend MRM at 1pm. You measured:

(1) students' score on a quiz (out of 10), and

(2) students' subjective feeling of wakefulness (out of 7).

Student	Condition	Quiz score	Wakefulness
Charles	6am	4	3
Vali	6am	7	4
Juan	6am	3	1
Bruno	1pm	9	5
Margaret	1pm	6	6
Kwame	1pm	5	7
Josh	1pm	8	6

summarise()

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

n()

The number of rows in a dataset/group

```
babynames %>% summarise(n = 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

→

n
1858689

n_distinct()

The number of distinct values in a variable

```
babynames %>% summarise(n = n(), nname = n_distinct(name))
```

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

→

n	nname
1858689	95025

Summary functions

- Take a vector as input.
- Return a single value as output.

Summary Functions

to use with summarise()

summarise() applies summary functions to columns to create a new table. Summary functions take vectors as input and return single values as output.

Counts

dplyr::**n()** - number of values/rows
dplyr::**n_distinct()** - # of uniques
sum(!is.na()) - # of non-NA's

Location

mean() - mean, also **mean(is.na())**
median() - median

Logicals

mean() - Proportion of TRUE's
sum() - # of TRUE's

Position/Order

dplyr::**first()** - first value
dplyr::**last()** - last value
dplyr::**nth()** - value in nth location of vector

Rank

quantile() - nth quantile
min() - minimum value
max() - maximum value

Spread

IQR() - Inter-Quartile Range
mad() - mean absolute deviation
sd() - standard deviation
var() - variance

Grouping cases

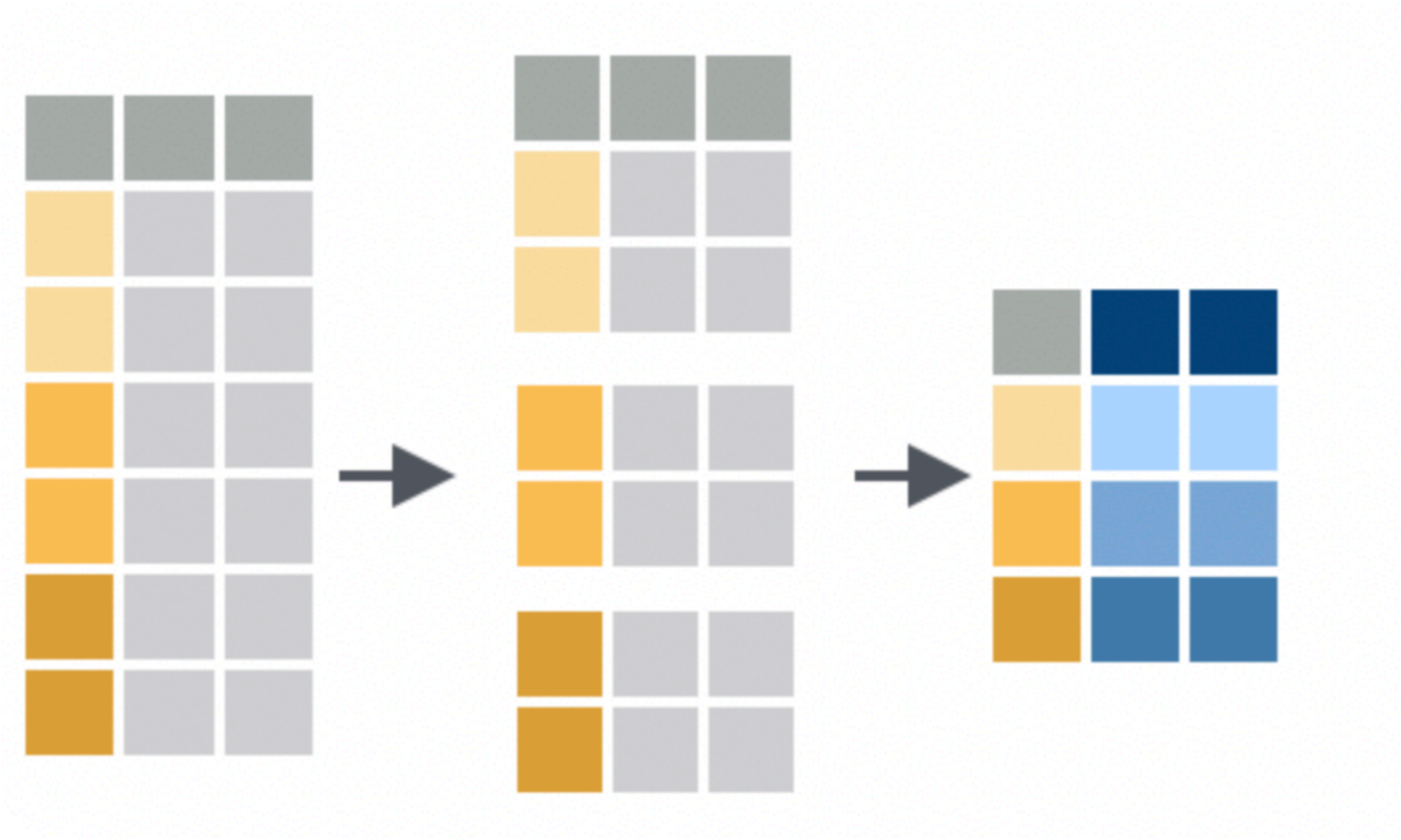
Data from an experiment where you randomly assigned 20 people to attend MRM at 6am and 20 people to attend MRM at 1pm. You measured:

(1) students' score on a quiz (out of 10), and

(2) students' subjective feeling of wakefulness (out of 7).

Student	Condition	Quiz score	Wakefulness
Charles	6am	4	3
Vali	6am	7	4
Juan	6am	3	1
Bruno	1pm	9	5
Margaret	1pm	6	6
Kwame	1pm	5	7
Josh	1pm	8	6

Grouping rows before summarizing



Adapted from: Rstudio Tidyverse cookbook

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

ungroup()

TWO grouping variables

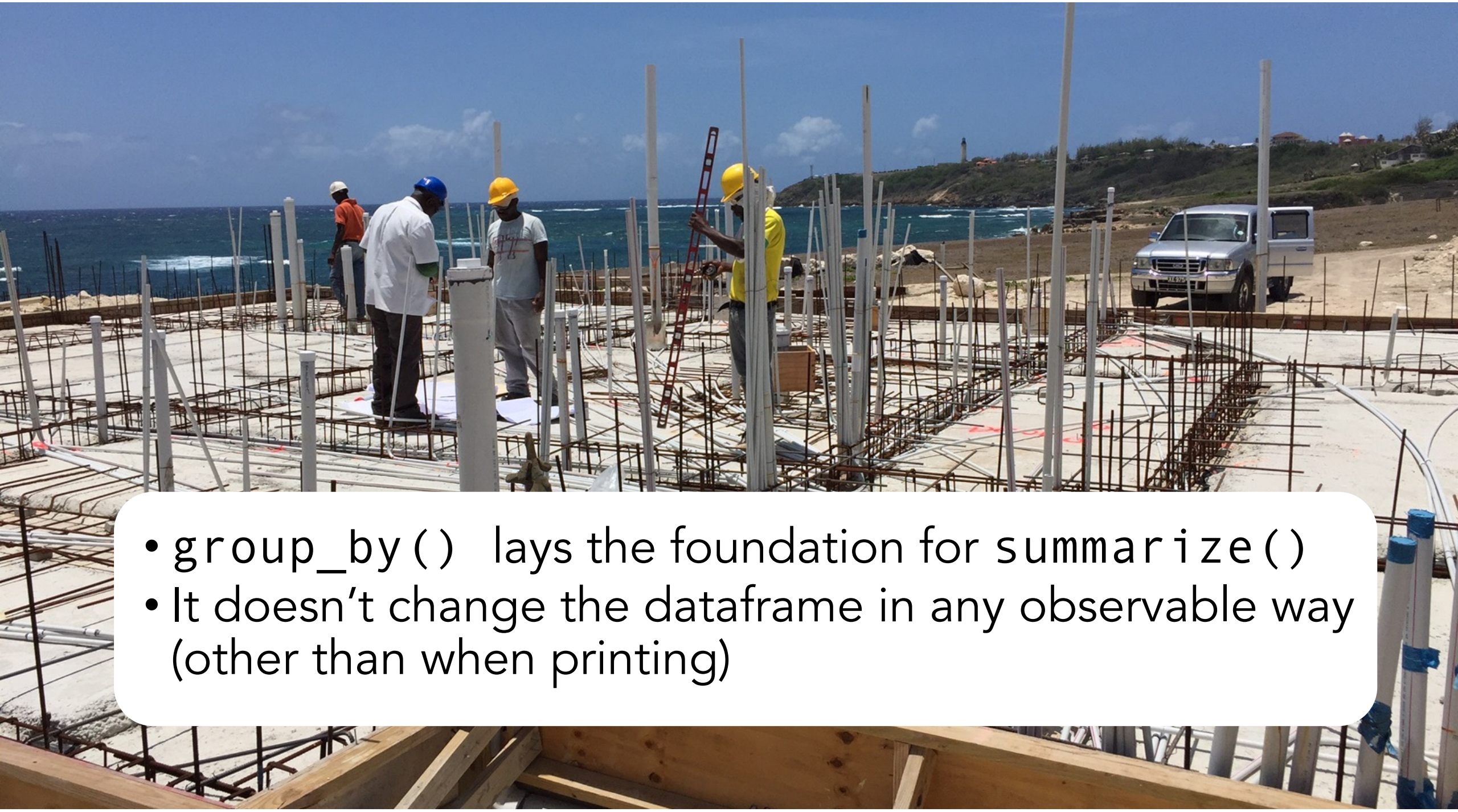
Removes grouping criteria from a data frame.

```
babynames %>%
  group_by(name, sex) %>%
  summarise(total = sum(n)) %>%
  arrange(desc(total))
#       name   sex   total
#  1   James     M 5120990
#  2    John     M 5095674
#  3  Robert     M 4803068
#  4 Michael     M 4323928
#  5    Mary     F 4118058
```

ungroup()

Removes grouping criteria from a data frame.

```
babynames %>%
  group_by(name, sex) %>%
  ungroup() %>%
  summarise(total = sum(n)) %>%
  arrange(desc(total))
#       total
# 1 340851912
```



- `group_by()` lays the foundation for `summarize()`
- It doesn't change the dataframe in any observable way (other than when printing)

mutate()

Artwork by
@allison_horst

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

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

Recap: Single table verbs

Extract variables with **select()**

Extract cases with **filter()**

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

Make tables of summaries with **summarise()**.

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

Sketch the data frame…

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

mean	sum	n
42	252	6

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

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

mean	sum	n
42	252	6

Sketch the data frame…

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

mean	sum	n
18.5	37	2
19.0	38	2
88.5	177	2

group_by() + summarise()

Sketch the data frame…

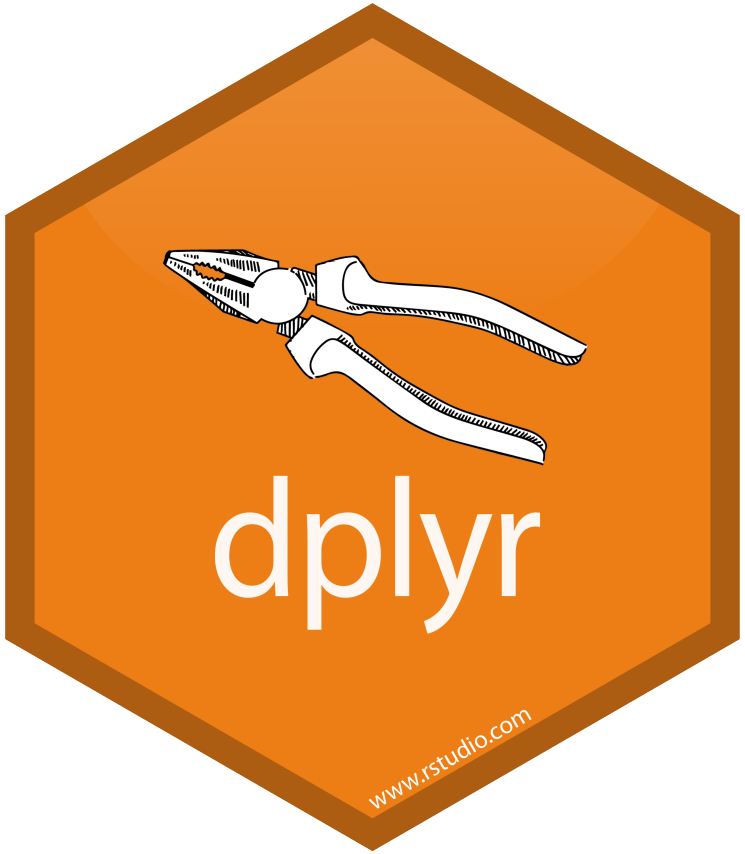
city	particle size	amount ($\mu g/m^3$)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

city	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	particle size	mean	sum	n
New York	large	23	23	1
New York	small	14	14	1
London	large	22	22	1
London	small	16	16	1
Beijing	large	121	121	1
Beijing	small	56	56	1

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

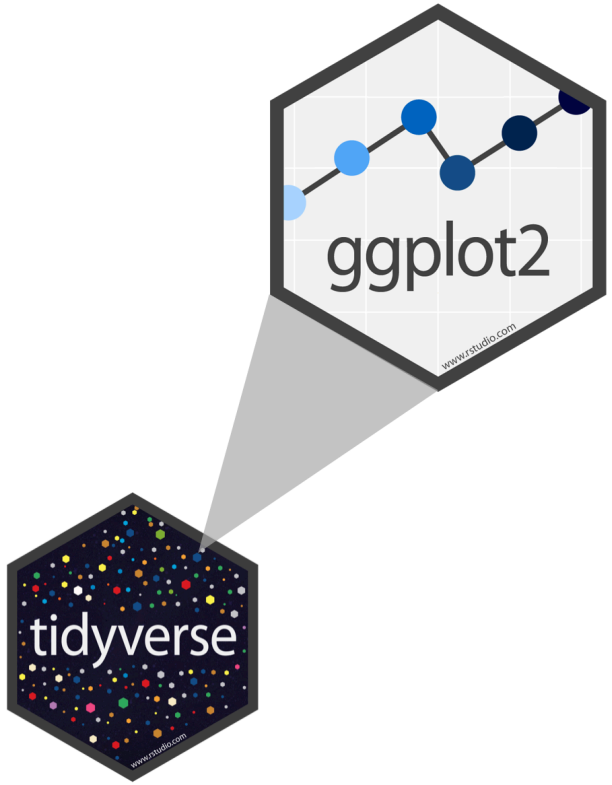
Things to know about dplyr verbs



- First argument is *always* a data frame
- Subsequent arguments say what to do with that data frame
- Always return a data frame
- Don't modify in place
- Link to more practice on website

Adapted from datasciencebox. by CC

Next Time: Plotting



Reading:

3 Make a plot

This Chapter will teach you how to use ggplot's core functions to produce a series of scatterplots. From one point of view, we will proceed slowly and carefully, taking our time to understand the logic behind the commands that you type. The reason for this is that the central activity of visualizing data with ggplot more or less *always* involves the same sequence of steps. So it is worth learning what they are.

Office Hours:

Roderick 3:30-5:30pm **today** (email)

Molly 2:45-4:45pm Wednesday

Acknowledgements

Slides 9-33 adapted from
https://github.com/rstudio/master-the-tidyverse by CC license