

Working with Tidy Data

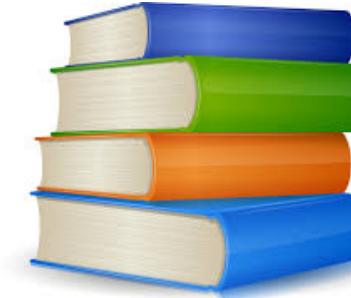
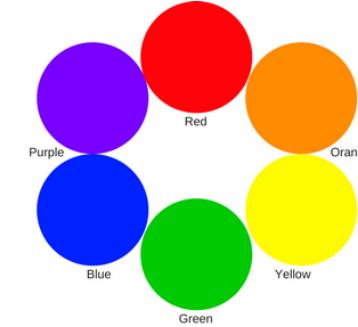
27 January 2020

Modern Research Methods

Course Website: <https://cumulativescience.netlify.com/>

Last Time: Variable types

- **Qualitative** – describe quality (no intrinsic ordering)
- **Quantitative** – describe quantity
 - **Binary** – 1 or 0 (or, TRUE or FALSE)
 - **Integers** – whole numbers
 - **Real numbers** – have fractional/decimal part



What is the metaphysical status of variable types?



A screenshot of a Twitter thread. The first tweet is from Solomon Messing (@SolomonMg) 19 hours ago, stating: "Hi. Email was invented 50 years ago. Can we all stop saying "nice to e-meet you" now thanks". It has 6 replies, 2 retweets, and 37 likes. The second tweet is from Lukasz Dębowski (@LukaszJDebowski) 22 hours ago, linking to an essay about GPT-2. The third tweet is from Gary Marcus (@GaryMarcus) 22 hours ago, discussing AI challenges. Both the second and third tweets have 1 reply, 2 retweets, and 12 likes.

Variable types are not inherent; they're determined by the analyst so you can do the analysis you want.



V1	V2	V3	V4



KEY ANALYSIS

e.g., Are there more x than y?

Is x greater than y?

Class attendance



How many students showed up to class today?
(binary)

Student	Attendance
Kyla	TRUE
Kara	FALSE
Zara	TRUE

What was the most common type of attention of students in class today?
(qualitative)

Student	Attendance
Kyla	“quiet”
Kara	NA
Zara	“inquisitive”

How many times did each student comment in class today? *(integer)*

Student	Attendance
Kyla	1
Kara	NA
Zara	3

How many seconds early was each student today?
(real number)

Student	Attendance
Kyla	120.457
Kara	NA
Zara	125.332

Variables, Observations, and Values

- **Variable** – unique measurement or quantity
 - e.g., temperature, mood, attendance, # of books owned, reaction time, color
- **Observation**– Smallest unit you have data about (person, trial in an experiment, city, school)
- **Value** – Quantity/quality associated with a particular variable and observation

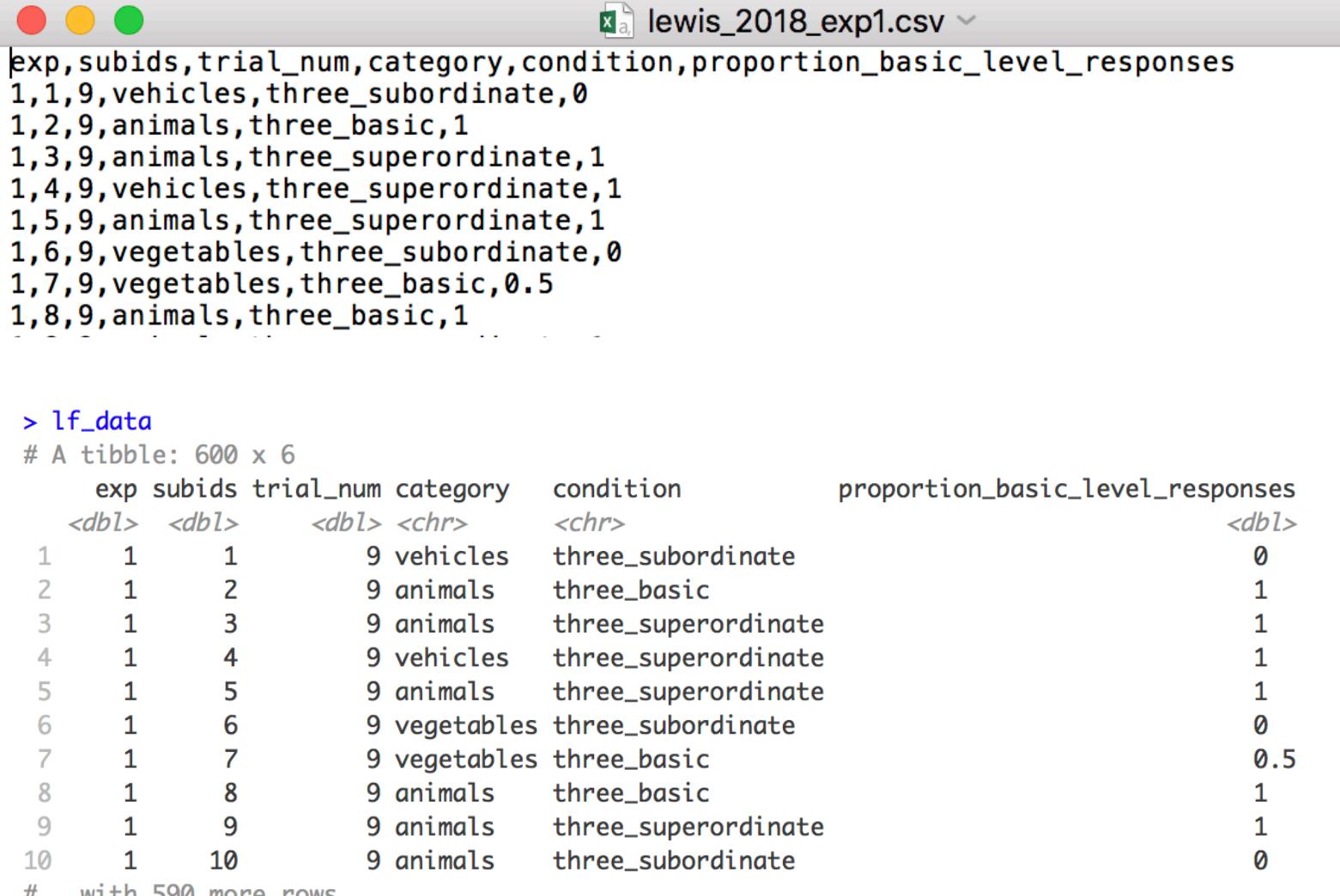
Variable – Jan. high temp. in PGH
Observation – day
Value – 38 degrees

Variable – age of students in MRM
Observation – student
Value – 19.3 years

Variable – Condition
Observation – person
Value – tennis class or chess class

Working with data in R

- 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



The screenshot shows the RStudio interface. At the top, there's a toolbar with three colored circles (red, yellow, green) and a tab labeled "lewis_2018_exp1.csv". Below the toolbar, the CSV file content is displayed:

```
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
```

Below the file content, the R console output shows the creation of a tibble:

```
> 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
```

At the bottom, it says "# ... with 590 more rows".

readr functions

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

read_csv()

readr functions share a common syntax

```
df <- read_csv("path/to/file.csv", ...)
```

object to save
output into

path from working
directory to file

```
lf_data <- read_csv("data/lewis_2018_exp1.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...



Tidy data

- When analyzing data, we want the data to be *tidy* ("long format")
- In a tidy dataset:

Each **variable** is its own column

V1	V2	V3	V4

Each **observation** is its own row

V1	V2	V3	V4

Each **value** is its own cell

V1	V2	V3	V4
O	O	O	O
O	O	O	O
O	O	O	O

* Allows R to use vectorized observations

Variable – Jan. high temp in PGH
Observation – day
Value – 38 degrees

Tidy data of Pittsburgh January 2020 Temperature

Date	High Temp.	Low Temp.
1/1/2020	38	28
1/2/2020	48	28
1/3/2020	51	45

Observation →

↑
Variable

Sketch the tidy data...

Of the data from an experiment where you randomly assigned 20 people to attend MRM at 6am and 20 people to attend MRM at 2pm. You measured:

- (1) students' score on a quiz (out of 10), and
- (2) students' subjective feeling of wakefulness (out of 7).

Tidy data of MRM class time experiment

Observation→

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

↑
Variable

**Airplanes on Hand in the AAF, By Major Type:
Jul 1939 to Aug 1945**

End of Month	Total	Very Heavy Bombers	Heavy Bombers	Medium Bombers	Light Bombers	Fighters	Reconnaissance	Transports	Trainers	Communications
1939										
Jul	2,402	-	16	400	276	494	356	118	735	7
Aug	2,440	-	18	414	276	492	359	129	745	7
[Germany invades Poland, 1 Sep 1939]										
Sep	2,473	-	22	428	278	489	359	136	754	7
Oct	2,507	-	27	446	277	490	365	137	758	7
Nov	2,536	-	32	458	275	498	375	136	755	7
Dec	2,546	-	39	464	274	492	378	131	761	7
1940										
Jan	2,588	-	45	466	271	464	409	128	798	7
Feb	2,658	-	49	470	271	458	415	128	860	7
Mar	2,709	-	54	468	267	453	415	125	920	7
Apr	2,806	-	54	468	263	451	416	125	1,022	7
May	2,906	-	54	470	259	459	410	124	1,123	7
Jun	2,966	-	54	478	166	477	414	127	1,243	7
[France surrenders to Germany, 25 Jun 1940] [Battle of Britain begins, 10 July 1940]										
Jul	3,102	-	56	483	161	500	410	128	1,357	7
Aug	3,295	-	65	485	158	539	407	128	1,506	7

Source: Army Air Forces Statistical Digest, WW II

	A	AA	AB	AC	AD	AE	AF	AG	AH
1	Estimated HIV Prevalence% - (Ages 15-49)	2004	2005	2006	2007	2008	2009	2010	2011
2	Abkhazia								
3	Afghanistan						0.06	0.06	0.06
4	Akrotiri and Dhekelia								
5	Albania								
6	Algeria	0.1	0.1	0.1	0.1	0.1			
7	American Samoa								
8	Andorra								
9	Angola	1.9	1.9	1.9	1.9	2.1	2.1	2.1	2.1
10	Anguilla								
11	Antigua and Barbuda								
12	Argentina	0.4	0.4	0.4	0.4	0.5	0.4	0.4	0.4
13	Armenia	0.1	0.1	0.1	0.1	0.1	0.2	0.2	0.2
14	Aruba								
15	Australia	0.1	0.1	0.1	0.1	0.1	0.2	0.2	0.2
16	Austria	0.2	0.2	0.2	0.3	0.3	0.3	0.4	0.4
17	Azerbaijan	0.06	0.06	0.06	0.1	0.1	0.1	0.1	0.1
18	Bahamas	3	3	3	3.1	3.1	2.9	2.8	2.8

Source: [Gapminder](#), Estimated HIV prevalence among 15-49 year olds

Working with tidy data in R

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**

Note: All functions in the tidyverse should be used with tidy data!

summarise()

A faint, large watermark of the R logo is positioned in the bottom right corner of the slide. The logo consists of a circular emblem with a horizontal line through it, containing the letters 'R'.

summarise()

Compute table of summaries.

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()

summary function

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

Vectorized Functions

to use with mutate()

Offsets

dplyr::lag() - Offset elements by ...
dplyr::lead() - Offset elements by ...

Cumulative Aggregates

dplyr::cumall() - Cumulative all()
dplyr::cumany() - Cumulative any()
cummax() - Cumulative max()
dplyr::cummean() - Cumulative mean()
cummin() - Cumulative min()
cumprod() - Cumulative prod()
cumsum() - Cumulative sum()

Rankings

dplyr::cume_dist() - Cumulative dist values
dplyr::dense_rank() - rank with ties = 1.0
dplyr::min_rank() - rank with ties = min
dplyr::ntile() - k-way rankings
dplyr::percent_rank() - min + rank scaled to [0,1]
dplyr::row_number() - rank with ties = first

Math

+ - * ^ %/% %>% - arithmetic ops
log(), log2(), log10(), loge()
<-, >, >=, >>, >= - logical comparisons

Misc

dplyr::between() - x > n < y < z
dplyr::case_when() - init case, else()
dplyr::coalesce() - fastest NA values by element across a set of vectors
if_else() - element-wise if/else()
dplyr::nz_wt() - replace specific values with NAs
pnas() - element-wise nans()
pnmin() - element-wise min()
dplyr::recode() - re-coded w/ new values
dplyr::recode_factor() - Value can switch for factors

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.

Combine Tables

Combine Variables

Combine Cases

bind_cols()

Use bind_col() to paste tables beside each other as they are.

bind_rows()

Pastes tables placed side-by-side as a single table. BE SURE THAT ROWS ALIGN.

left_join(x, y, by = "id")

Left Joining table x to table y on column id. If there are multiple columns named id, dplyr will automatically choose the first one.

right_join(x, y, by = NULL, copy = FALSE, suffix = c("x", "y"))

Right Joining table y to table x on column id. If there are multiple columns named id, dplyr will automatically choose the first one.

inner_join(x, y, by = NULL, copy = FALSE, suffix = c("x", "y"))

Join data from both tables on their common columns.

full_join(x, y, by = NULL, copy = FALSE, suffix = c("x", "y"))

Join data from both tables on all rows.

intersect(x, y)

Returns the unique values that appear in both datasets.

setdiff(x, y)

Returns the unique values that appear in the first dataset but not the second.

union(x, y)

How to union two datasets. union_all() eliminates duplicates.

select(x, y)

Used to keep either two data sets contain the exact same rows, or they do not.

Extract Rows

filter(x, y = "A")

Use a filtering variable to filter a table against the rows in a vector.

semi_join(x, y, by = NULL, ...)

Retains rows of x that have a matching row in y. USEFUL TO JOIN ON A KEY.

anti_join(x, y, by = NULL, ...)

Retains rows of x that do not have a match in y. USEFUL TO JOIN ON A KEY.



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



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
```



mutate()

A faint watermark of the R logo is visible in the bottom right corner, consisting of a circular emblem with the letters "R" and "D" inside.

mutate()

Create new columns.

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

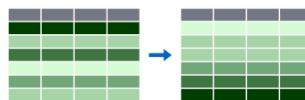
Recap: Single table verbs



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

mean	sum	n
42	252	6



Sketch the data frame...

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



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()

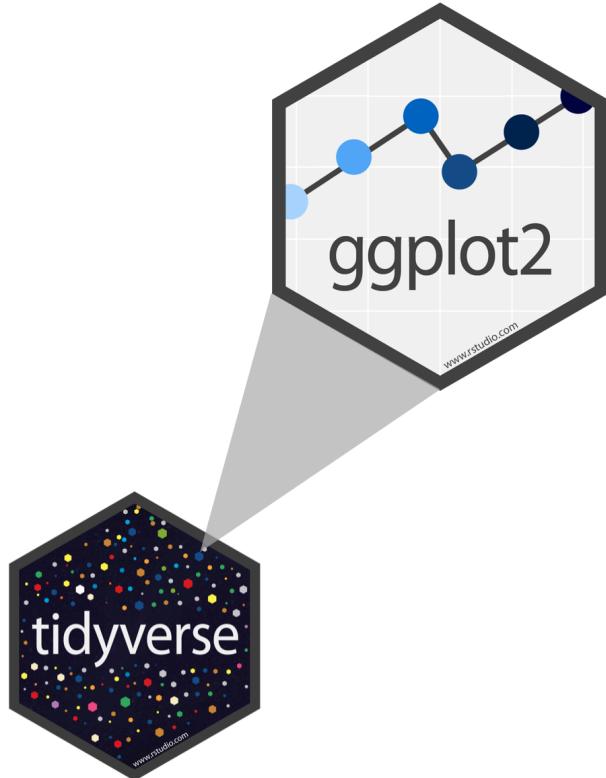


Sketch the data frame...

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

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:

Jaeah 11:30-1:30pm **today only**;
Molly 4:30-6:30pm Wednesday (Porter 223A)

Acknowledgements

Slides 14-15 adapted from

<https://datasciencebox.org/> by CC license

Slides 16-35 adapted from

<https://github.com/rstudio/master-the-tidyverse> by CC license