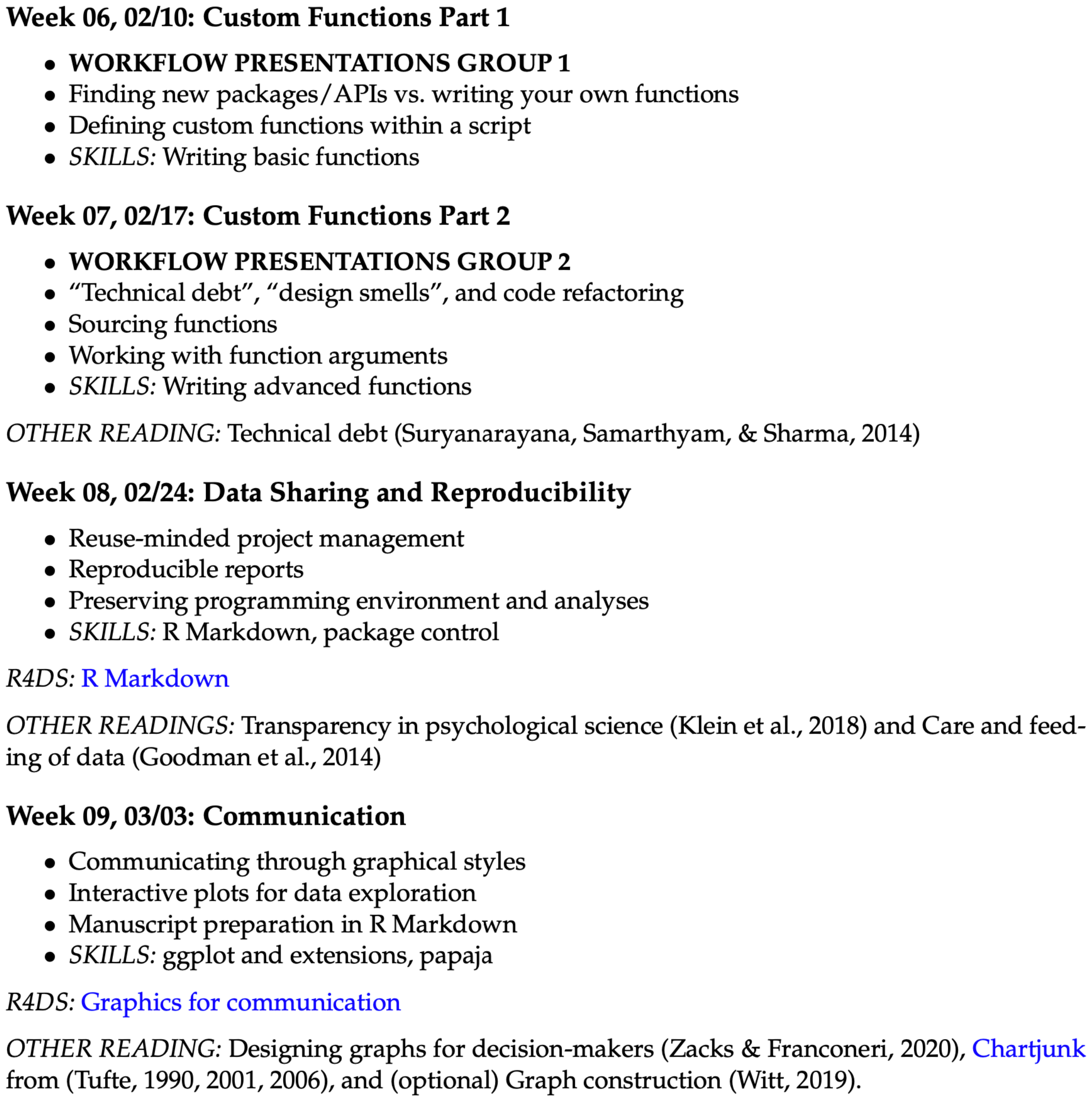
Today

1. Go over automation homework
2. Exploratory data analysis lecture
3. Visualization tutorial
4. Break
5. Group exercise – wrangling & EDA working in small groups
6. Reconvene to go over exercise

Workflow assignment reminders

- Everyone: Self-critique paper turned in before class on Canvas
- Next week's presenters
  - Tina, Michael, Ziyi, Antonio, Hea Jung, Michelle, Jun
  - Email PDF of slides to me before class
  - For presentation, you will share screen and have 8 minutes max to present
  - I will warn you at 7 minutes to wrap it up

Syllabus updates



Week 06, 02/10: Custom Functions Part 1

- **WORKFLOW PRESENTATIONS GROUP 1**
- Finding new packages/APIs vs. writing your own functions
- Defining custom functions within a script
- *SKILLS:* Writing basic functions

Week 07, 02/17: Custom Functions Part 2

- **WORKFLOW PRESENTATIONS GROUP 2**
- “Technical debt”, “design smells”, and code refactoring
- Sourcing functions
- Working with function arguments
- *SKILLS:* Writing advanced functions

OTHER READING: Technical debt (Suryanarayana, Samarthyam, & Sharma, 2014)

Week 08, 02/24: Data Sharing and Reproducibility

- Reuse-minded project management
- Reproducible reports
- Preserving programming environment and analyses
- *SKILLS:* R Markdown, package control

R4DS: R Markdown

OTHER READINGS: Transparency in psychological science (Klein et al., 2018) and Care and feeding of data (Goodman et al., 2014)

Week 09, 03/03: Communication

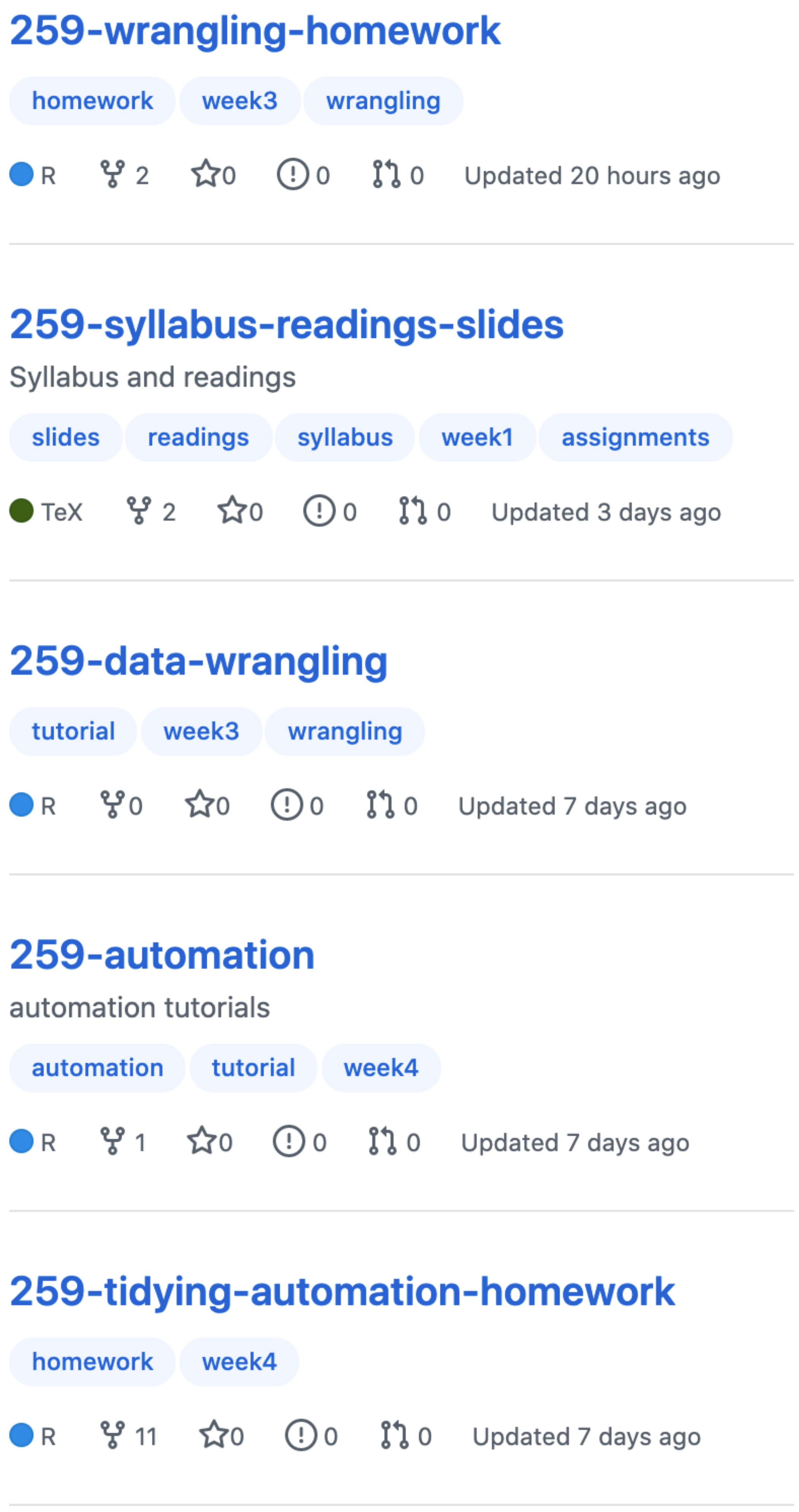
- Communicating through graphical styles
- Interactive plots for data exploration
- Manuscript preparation in R Markdown
- *SKILLS:* ggplot and extensions, papaja

R4DS: Graphics for communication

OTHER READING: Designing graphs for decision-makers (Zacks & Franconeri, 2020), Chartjunk from (Tufte, 1990, 2001, 2006), and (optional) Graph construction (Witt, 2019).

GitHub updates

- Adding any code Jake writes when going over homework
- Tagging repos so that they're a bit easier to find



259-wrangling-homework

homework · week3 · wrangling

● R ⑂ 2 ☆ 0 ⓘ 0 ⇅ 0 Updated 20 hours ago

259-syllabus-readings-slides

Syllabus and readings

slides · readings · syllabus · week1 · assignments

● TeX ⑂ 2 ☆ 0 ⓘ 0 ⇅ 0 Updated 3 days ago

259-data-wrangling

tutorial · week3 · wrangling

● R ⑂ 0 ☆ 0 ⓘ 0 ⇅ 0 Updated 7 days ago

259-automation

automation tutorials

automation · tutorial · week4

● R ⑂ 1 ☆ 0 ⓘ 0 ⇅ 0 Updated 7 days ago

259-tidying-automation-homework

homework · week4

● R ⑂ 11 ☆ 0 ⓘ 0 ⇅ 0 Updated 7 days ago

PSYC 259: Principles of Data Science

Week 5: Exploratory Data Analysis
Error Checking & Visualization

Exploratory Data Analysis

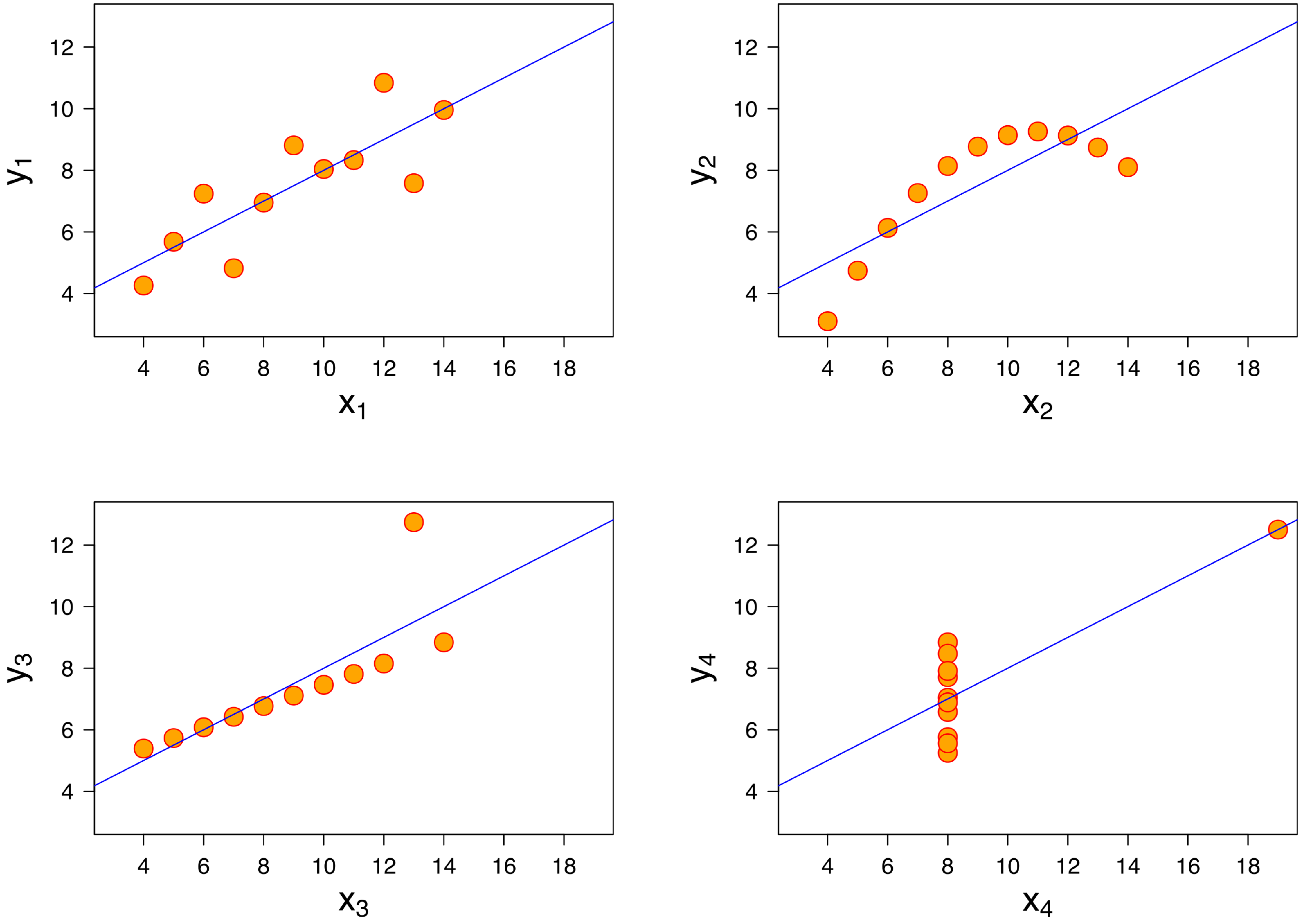
CDA vs. EDA

- Goals of Confirmatory Data Analysis
  - Hypothesis testing, probabilistic modeling, inference
- Goals of Exploratory Data Analysis (Tukey)
  - Understanding the patterns in the data
  - Generating hypotheses
  - Checking your assumptions about data quality
  - “To find the unexpected, to avoid being fooled, and to develop rich descriptions” (Behrens & Yu, 2003)

Why do we need EDA? (Behrens & Yu, 2003)

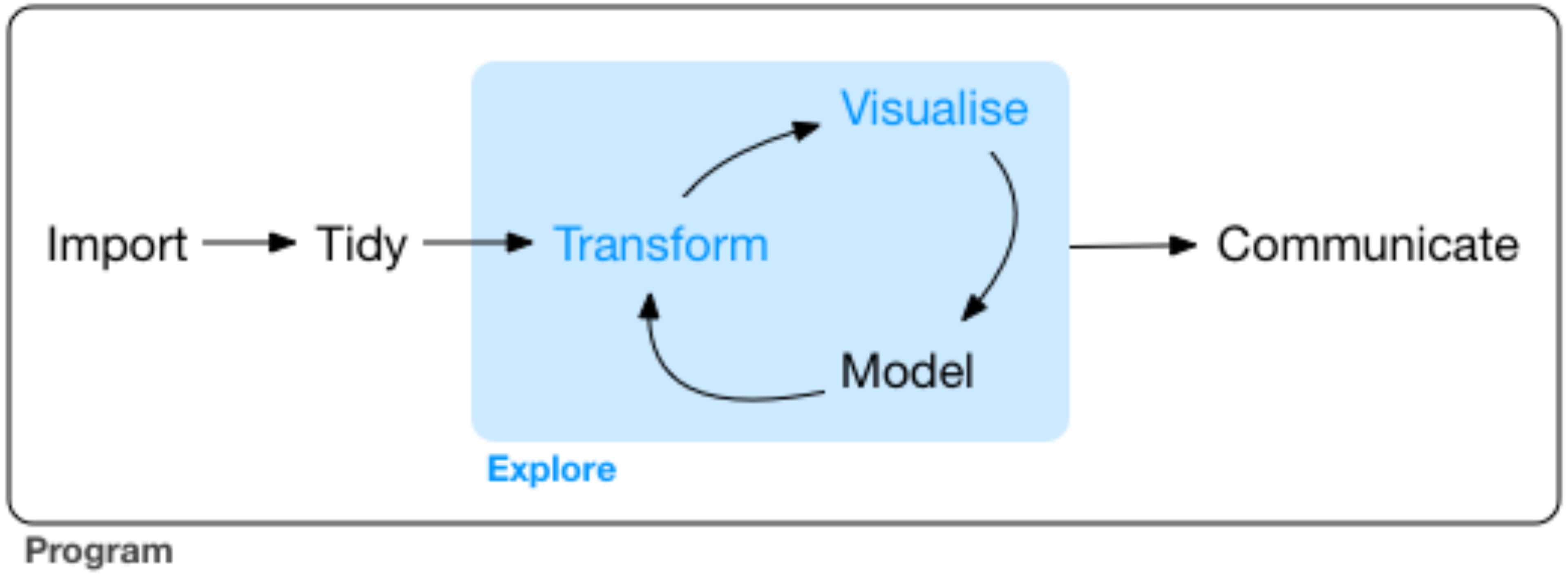
- Summarization = a loss of information
  - If you first look at summarized data (across trials, across participants, etc.), you may miss important patterns that exist at the raw data level
- Statistics lie, so you need graphics
  - Correlations without looking at the scatterplot
  - Means without examining outliers/distribution
  - Statistical tests without examining *n*

Anscombe’s quartet



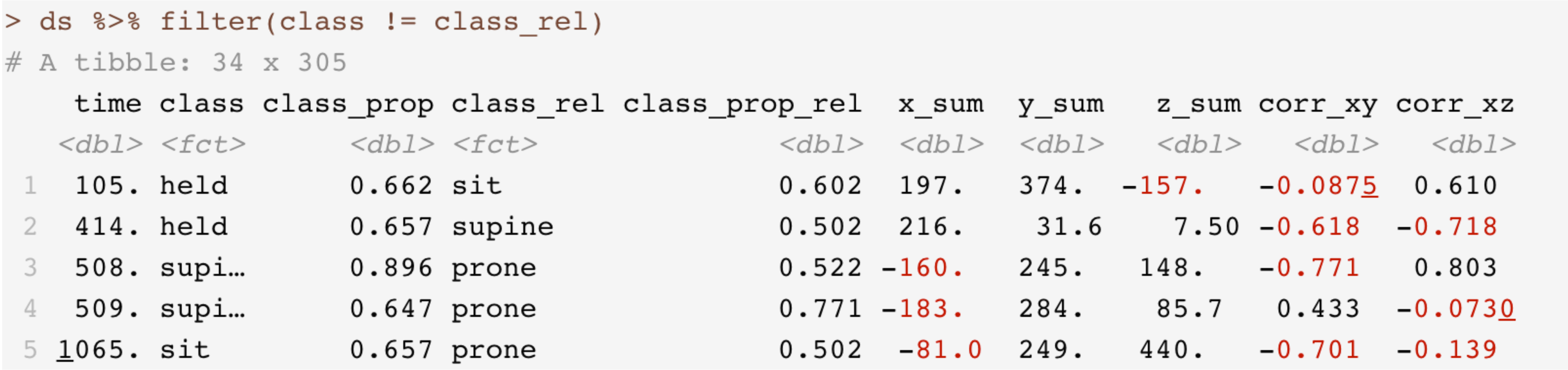
Where does wrangling stop and EDA begin?

- Data need to be minimally read in, appropriately labelled, and tidied to check and visualize
- EDA will reveal errors or redundancies that will require new data wrangling steps



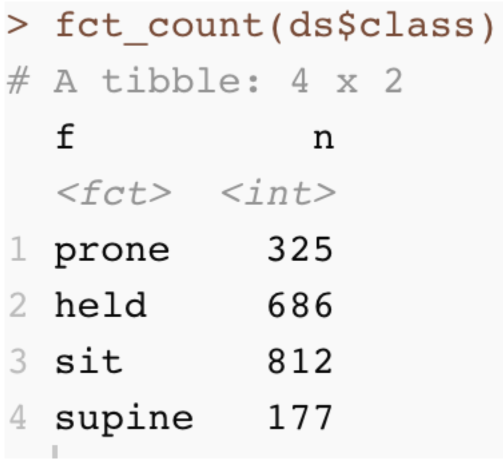
Tools for data checking that we have already covered

- filter with logical statements



```
> ds %>% filter(class != class_rel)
# A tibble: 34 x 305
   time class class_prop class_rel class_prop_rel  x_sum  y_sum  z_sum corr_xy corr_xz
  <dbl> <fct>      <dbl> <fct>              <dbl>  <dbl>  <dbl>  <dbl>   <dbl>   <dbl>
1  105. held       0.662 sit                0.602  197.   374.  -157.  -0.0875   0.610
2  414. held       0.657 supine             0.502  216.    31.6    7.50 -0.618  -0.718
3  508. supi…      0.896 prone              0.522 -160.   245.   148.  -0.771    0.803
4  509. supi…      0.647 prone              0.771 -183.   284.    85.7   0.433  -0.0730
5 1065. sit        0.657 prone              0.502  -81.0  249.   440.  -0.701   -0.139
```

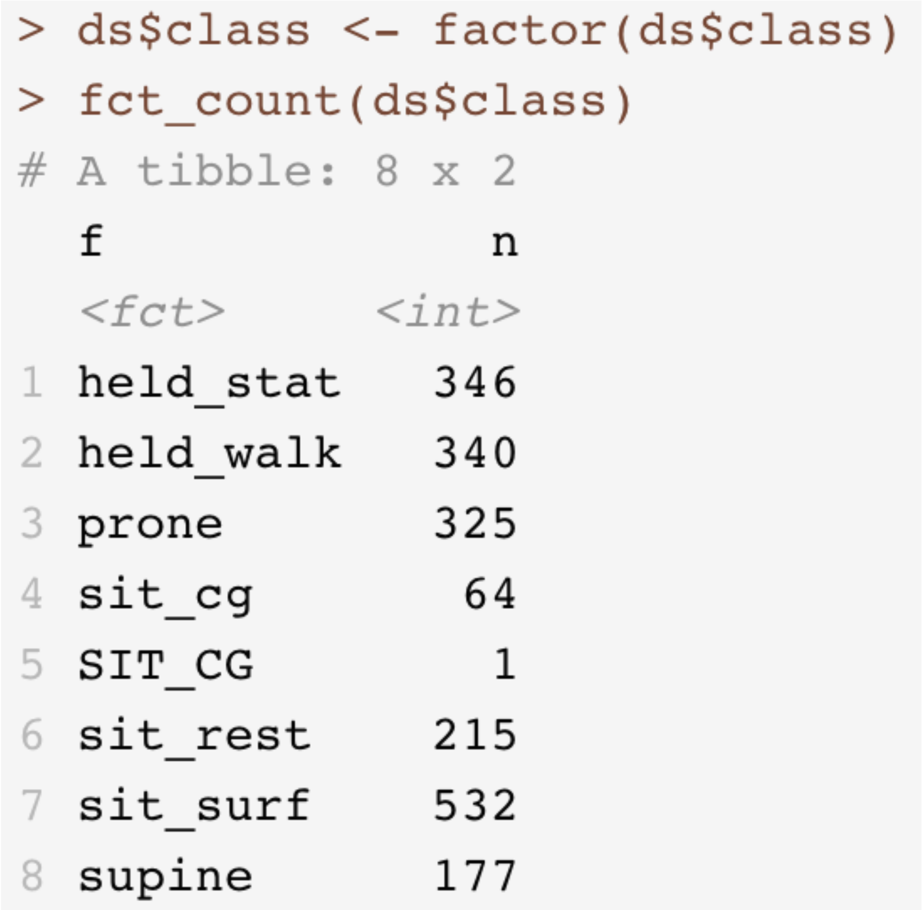
- fct_count to check factor frequencies



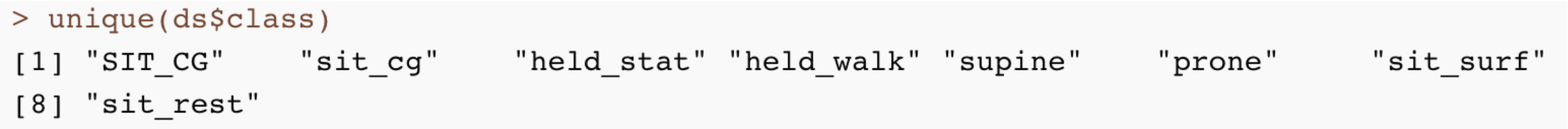
```
> fct_count(ds$class)
# A tibble: 4 x 2
  f          n
  <fct>  <int>
1 prone    325
2 held     686
3 sit      812
4 supine   177
```

Quick aside about factors

- When defining factors, all data that do not match a level will be coded as NA
- Checking levels before coding factors (or making a factor without pre-specified levels) should be a first step



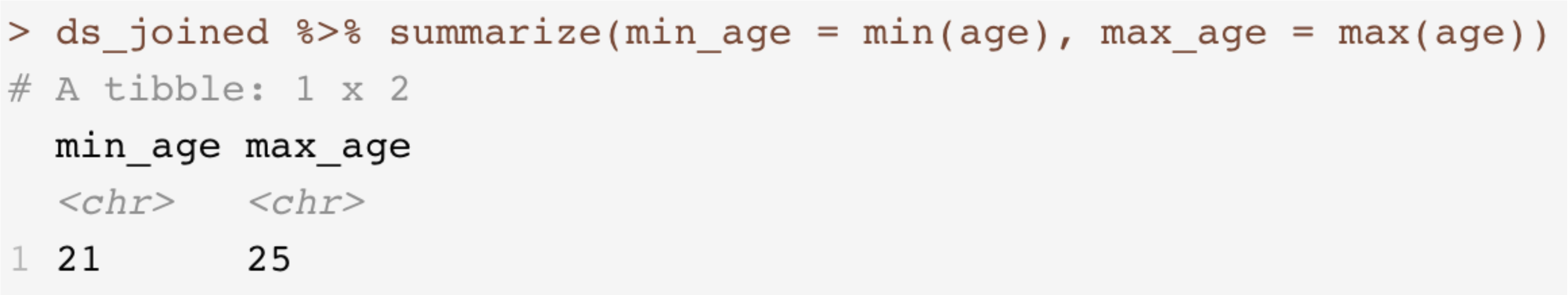
```
> ds$class <- factor(ds$class)
> fct_count(ds$class)
# A tibble: 8 x 2
  f             n
  <fct>     <int>
1 held_stat   346
2 held_walk   340
3 prone       325
4 sit_cg       64
5 SIT_CG        1
6 sit_rest    215
7 sit_surf    532
8 supine      177
```



```
> unique(ds$class)
[1] "SIT_CG"    "sit_cg"    "held_stat" "held_walk" "supine"    "prone"     "sit_surf" 
[8] "sit_rest"
```

Tools for data checking that we have already covered

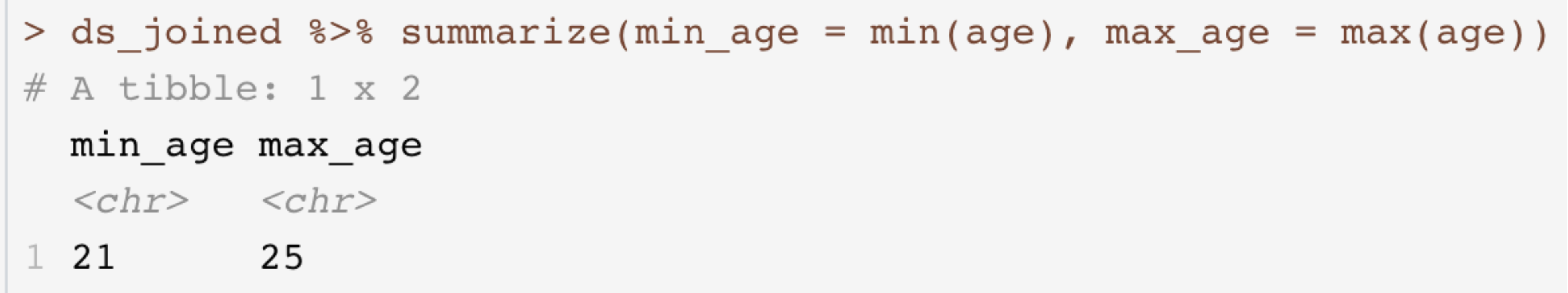
- summaries (with the right statistics/groupings)



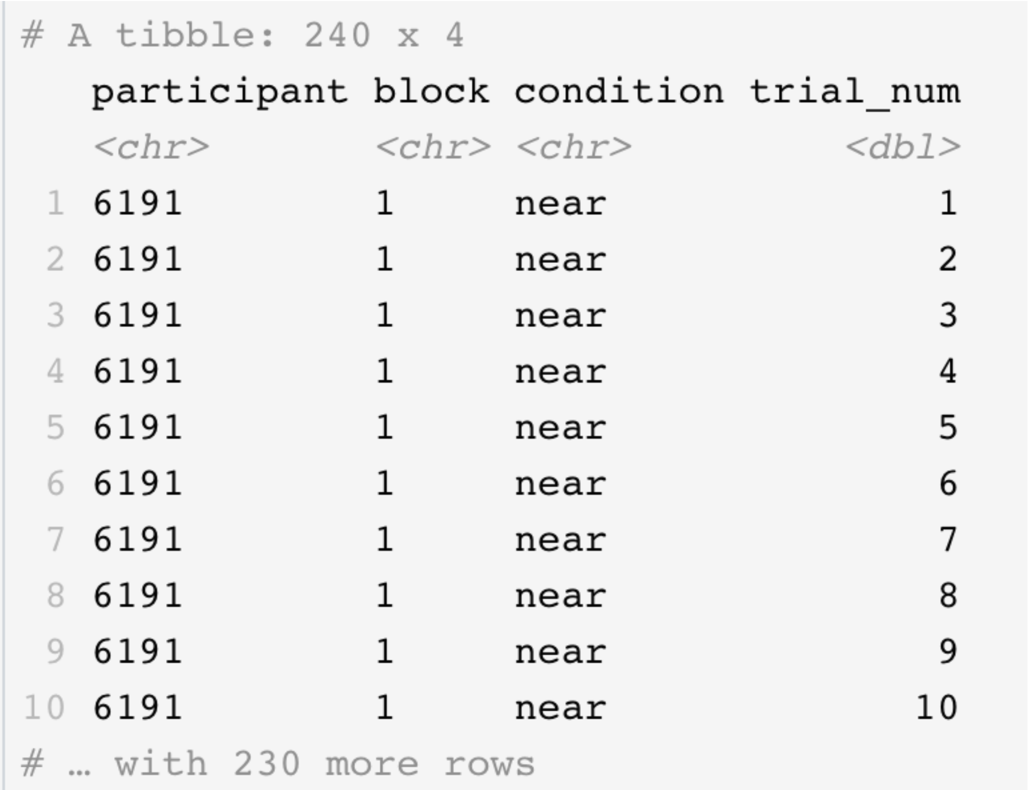
```
> ds_joined %>% summarize(min_age = min(age), max_age = max(age))
# A tibble: 1 x 2
  min_age max_age
  <chr>   <chr>  
1 21      25     
```

Tools for data checking that we have already covered

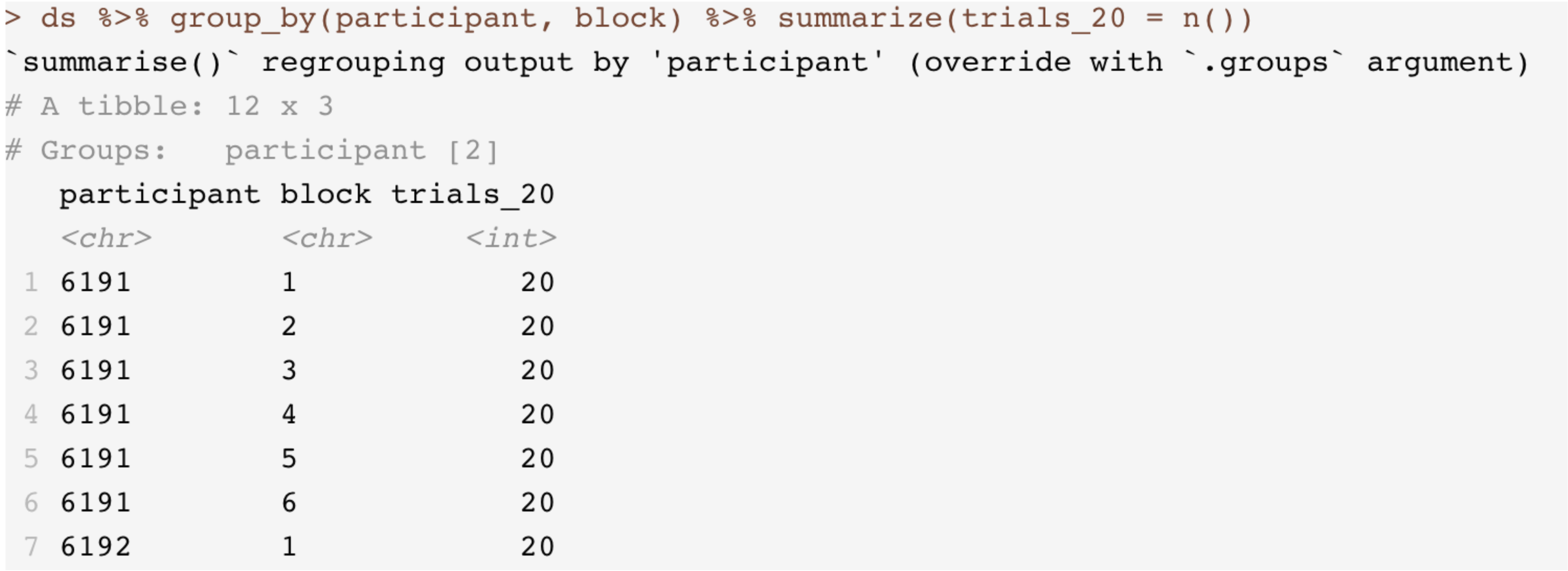
- summaries (with the right statistics/groupings)



```
> ds_joined %>% summarize(min_age = min(age), max_age = max(age))
# A tibble: 1 x 2
  min_age max_age
  <chr>   <chr>
1 21      25
```



```
# A tibble: 240 x 4
   participant block condition trial_num
   <chr>       <chr> <chr>         <dbl>
 1 6191        1     near              1
 2 6191        1     near              2
 3 6191        1     near              3
 4 6191        1     near              4
 5 6191        1     near              5
 6 6191        1     near              6
 7 6191        1     near              7
 8 6191        1     near              8
 9 6191        1     near              9
10 6191        1     near             10
# … with 230 more rows
```



```
> ds %>% group_by(participant, block) %>% summarize(trials_20 = n())
`summarise()` regrouping output by 'participant' (override with `.groups` argument)
# A tibble: 12 x 3
# Groups:   participant [2]
  participant block trials_20
  <chr>       <chr>     <int>
1 6191        1            20
2 6191        2            20
3 6191        3            20
4 6191        4            20
5 6191        5            20
6 6191        6            20
7 6192        1            20
```

Tools for data checking that we have already covered

- Automation
  - EDA means taking a detailed approach to look at data on different levels (participant/condition/wave/etc.)
  - Running multiple filters/checks, plotting multiple figures, etc. can get overwhelming without automation

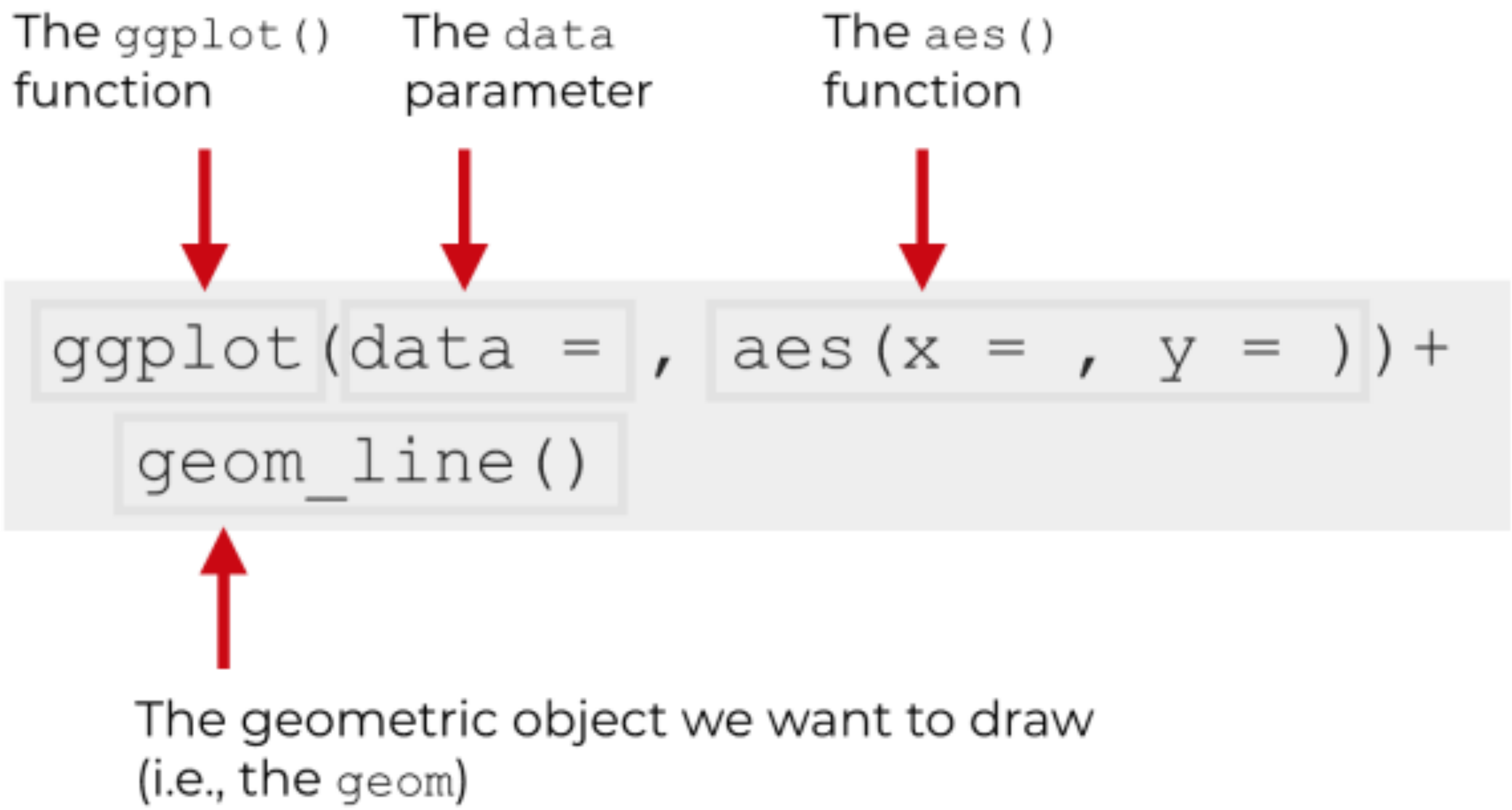
New tools for EDA - Visualizations

- DataExplorer package
  - Brute force, first glance methods
  - plot_histogram() of every continuous variable
  - plot_bar() counts of every categorical variable
- VisDat package
  - vis_miss() to plot missing values
  - vis_expect() to plot conditionals

New tools for EDA - Visualizations

- ggplot2 package (part of tidyverse)
  - Create any type of graph
  - Today we'll talk about making quicker plots for eda using geom_histogram, geom_point, geom_boxplot, and a few others
  - Week 9 we'll talk about making publication-ready plots to communicate effects

Anatomy of a ggplot call



```
ggplot(data = , aes(x = , y = )) +
  geom_line()
```

The ggplot() function

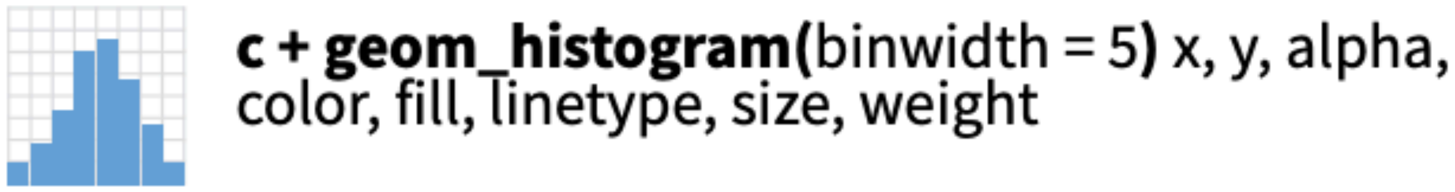
The data parameter

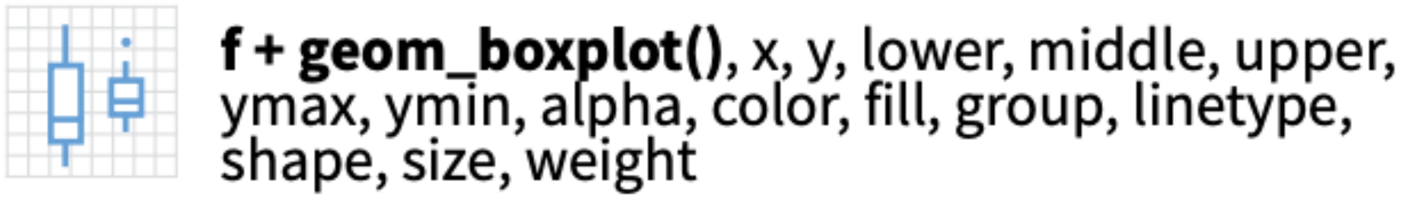
The aes() function

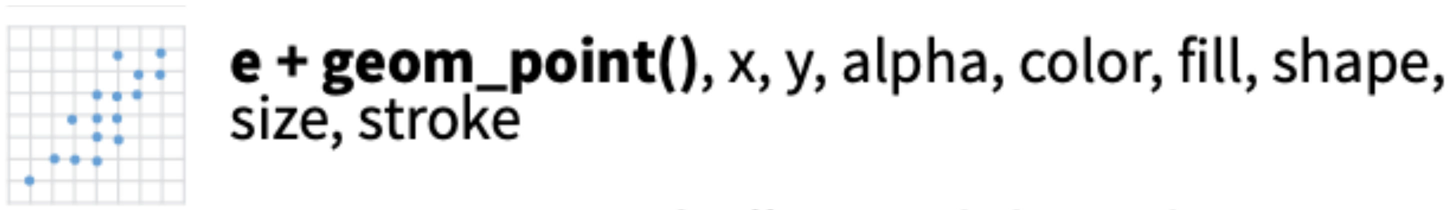
The geometric object we want to draw (i.e., the geom)

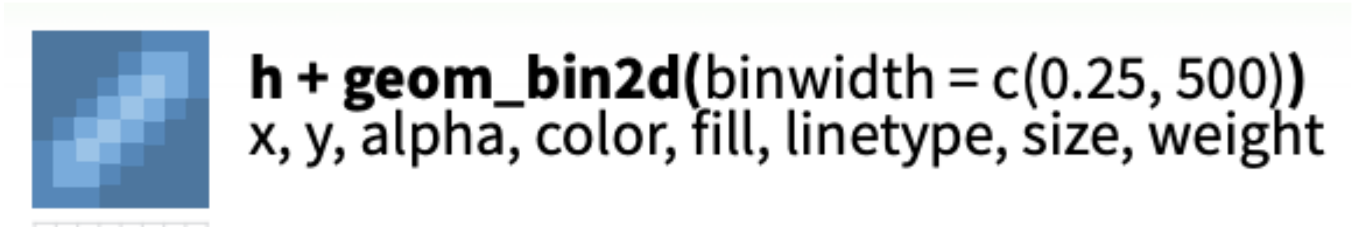
- define the dataset we are using (long format)
- define the mapping of variables to *aes*thestics
- Add (+) geoms, graphical elements like histograms, lines, points, bars, boxplots, and many others
- Optional arguments to change the overall look

Each type of geom has different aesthetics that can be mapped


c + geom_histogram(binwidth = 5**)** x, y, alpha, color, fill, linetype, size, weight

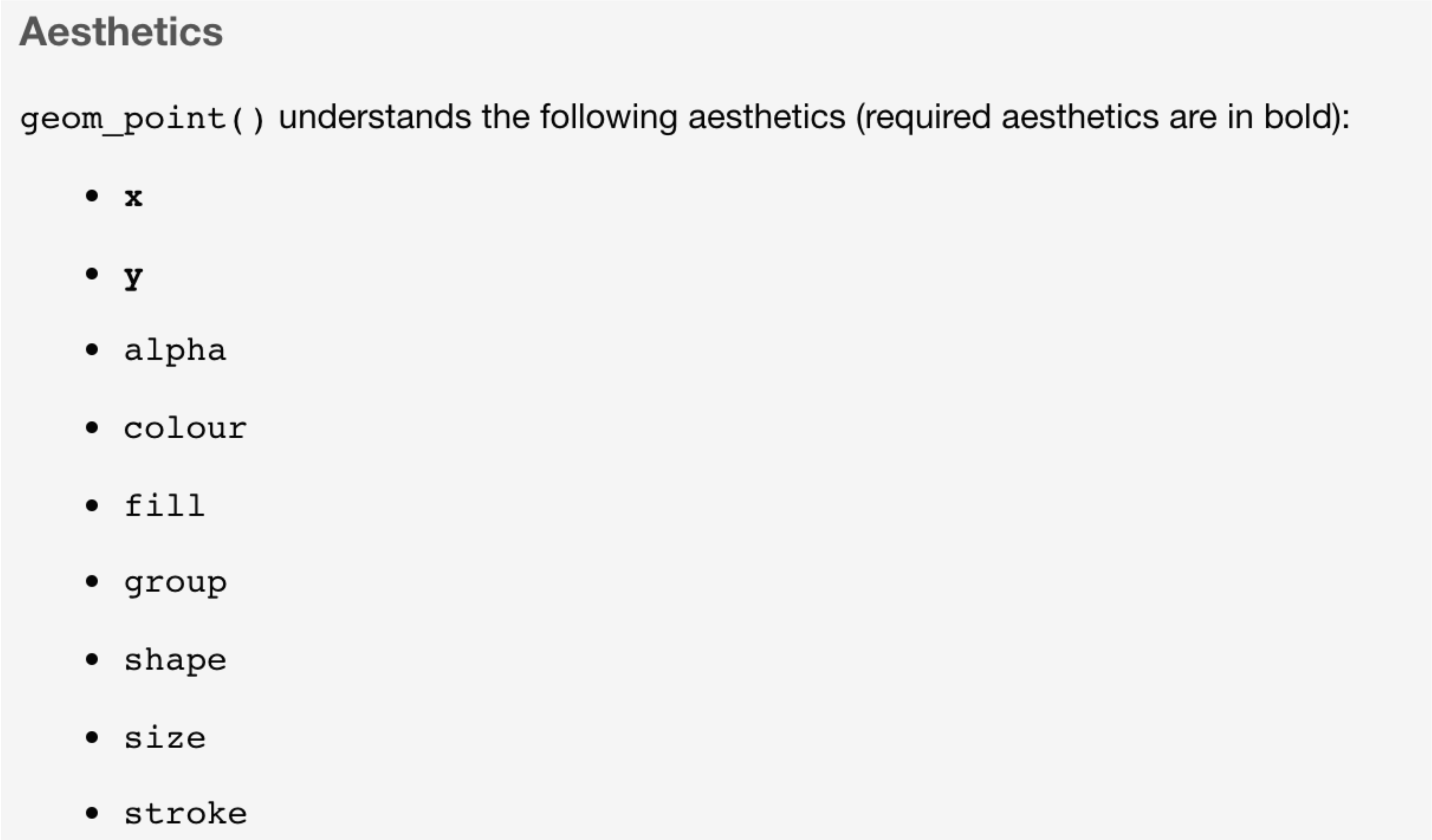

f + geom_boxplot(), x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight


e + geom_point(), x, y, alpha, color, fill, shape, size, stroke


h + geom_bin2d(binwidth = c(0.25, 500)**)** x, y, alpha, color, fill, linetype, size, weight

What aes values are required for each geom?

- Check the help page to see required mappings in bold



Aesthetics

geom_point() understands the following aesthetics (required aesthetics are in bold):

- **x**
- **y**
- alpha
- colour
- fill
- group
- shape
- size
- stroke

Adding elements to graphs

- ggplot() + geomX() +….
- Add (+) other modifications to the plot
    - xlim(lower_bound, upper_bound) or ylim
    - hline(yintercept = X) or vline
    - xlab(“x label”)
    - titles, custom scales, other geoms
- Make sure that plus is on the previous line, lines that start with + will throw an error

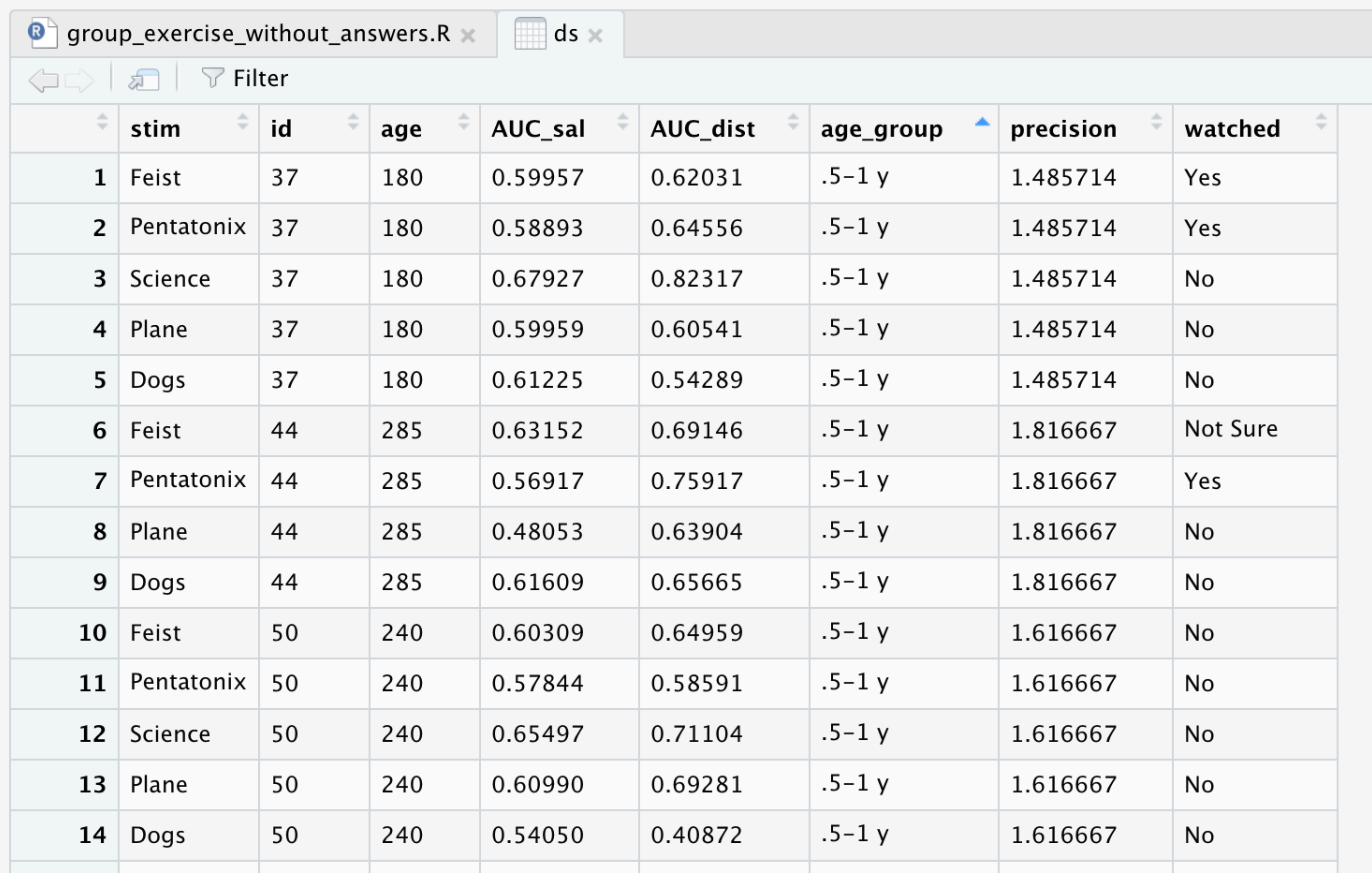
EDA visualization tutorial

Group work:
Exploratory Data Analysis

“259-group-wrangling-checking”

Study setup

- Participants (6 months - 10 years and adults) watched 5 different stimulus videos (“Feist”, “Pentatonix”, “Science”, “Plane”, “Dogs”) while their eyes were tracked
- The file includes their age in days and their age group (e.g., 0.5-1 year, 1-1.5 years, etc.)
- The DVs are AUC_sal and AUC_dist, which measure how well a saliency vs. a distance model predicted whether participants look (0-1, where 0.5 = chance)
- Precision is a measure of eye tracking data quality, where larger = worse (and over 2.5 is concerning)
- Watched asks whether participants had seen the videos prior to the study



	stim	id	age	AUC_sal	AUC_dist	age_group	precision	watched
1	Feist	37	180	0.59957	0.62031	.5-1 y	1.485714	Yes
2	Pentatonix	37	180	0.58893	0.64556	.5-1 y	1.485714	Yes
3	Science	37	180	0.67927	0.82317	.5-1 y	1.485714	No
4	Plane	37	180	0.59959	0.60541	.5-1 y	1.485714	No
5	Dogs	37	180	0.61225	0.54289	.5-1 y	1.485714	No
6	Feist	44	285	0.63152	0.69146	.5-1 y	1.816667	Not Sure
7	Pentatonix	44	285	0.56917	0.75917	.5-1 y	1.816667	Yes
8	Plane	44	285	0.48053	0.63904	.5-1 y	1.816667	No
9	Dogs	44	285	0.61609	0.65665	.5-1 y	1.816667	No
10	Feist	50	240	0.60309	0.64959	.5-1 y	1.616667	No
11	Pentatonix	50	240	0.57844	0.58591	.5-1 y	1.616667	No
12	Science	50	240	0.65497	0.71104	.5-1 y	1.616667	No
13	Plane	50	240	0.60990	0.69281	.5-1 y	1.616667	No
14	Dogs	50	240	0.54050	0.40872	.5-1 y	1.616667	No

Work in groups

- We’ll randomly assign you to groups of 3
- Work with your group in RStudio Cloud, and Jake and I will filter in to check on groups
- Do as much as you can, but don’t worry if you don’t get through all of them
- We will reconvene to go over it together