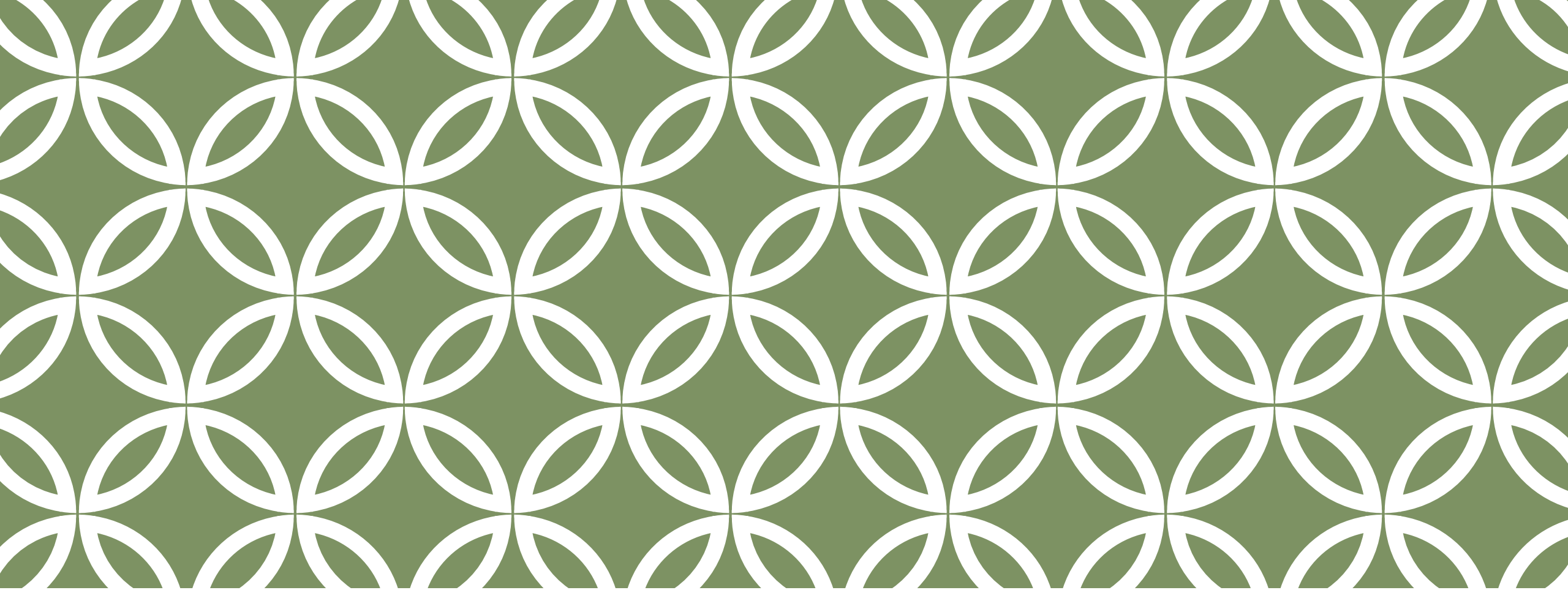
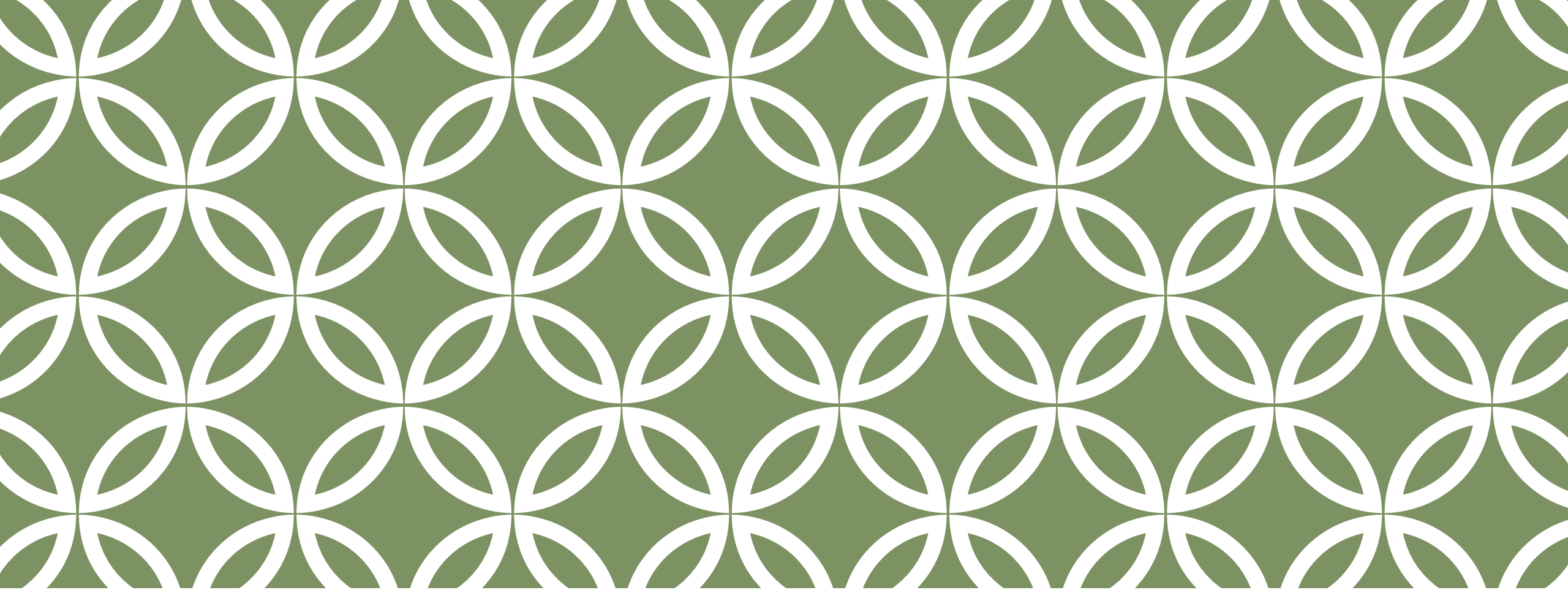


LING 0700 LAB: WEEK 2

TA: Wesley Mark Lincoln

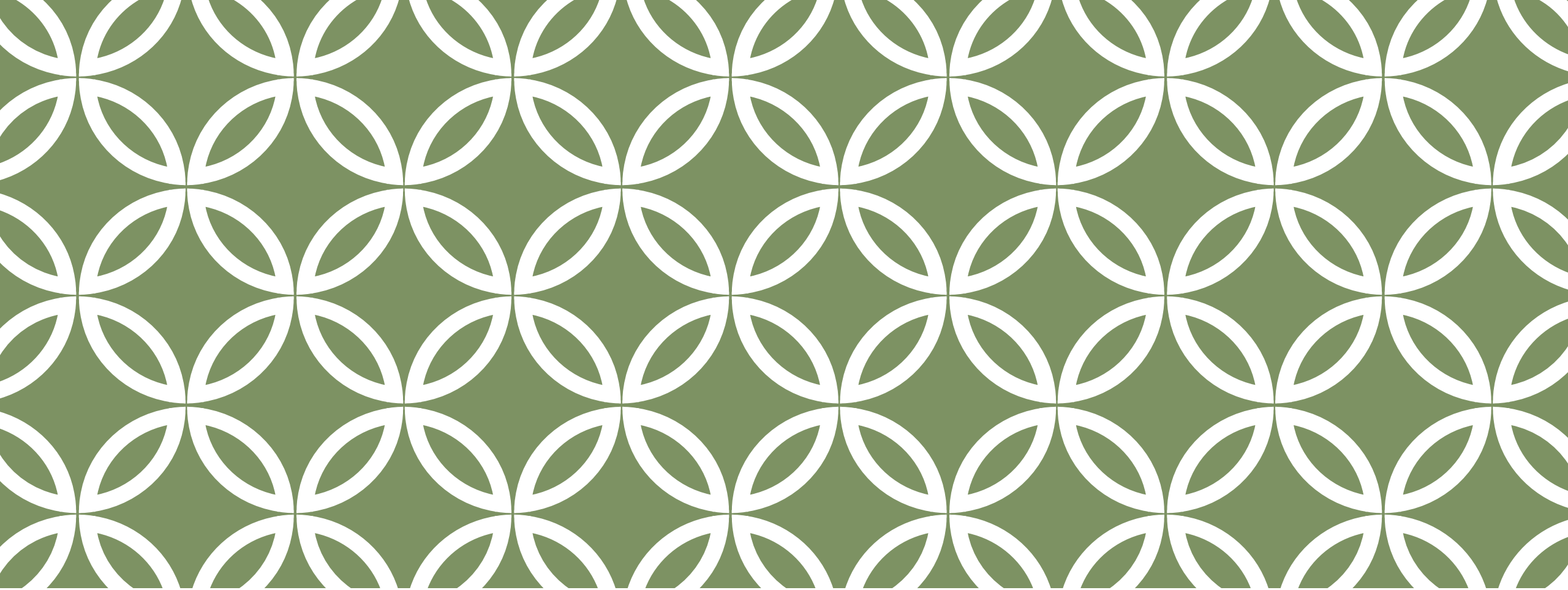


COURSE ADMIN

REMINDERS

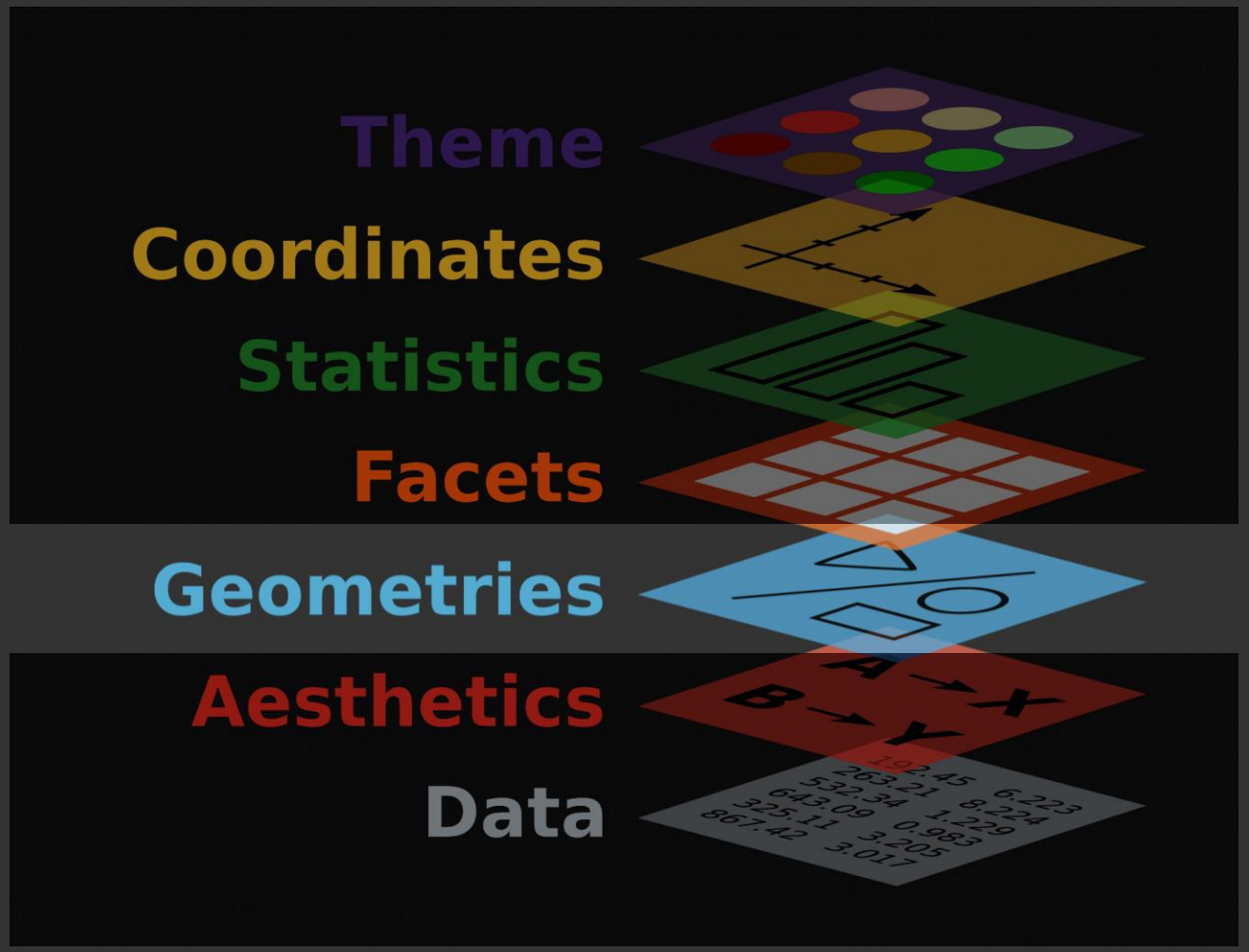
- Reminder: PSet 1 due Monday, September 9, 12pm
  - How to submit PSets?
  - **Administrative** questions about PSet 1?

- Automatic extension until Thursday, September 12, 12pm
  - e.g. to attend my office hour (Monday 2pm-3pm)



LECTURE REVIEW

GGPLOT LAYERS

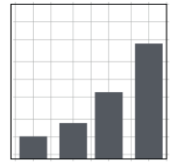


Geoms:

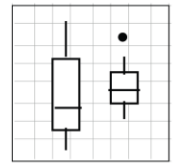
- Geometric objects that represent data
- e.g. lines, dots, boxes, bars, curves, and so on.

Different Geoms (Plot Type) in ggplot2

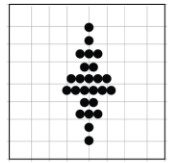
Two Variables (X,Y)
- Discrete X, continuous Y
- Visualise distribution of Y with respect to X



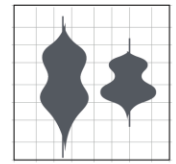
geom_col()
- heights of bars represent values



geom_boxplot()
- summarise distribution using median, hinges and whiskers

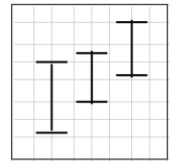


geom_jitter()
- adds jitter to prevent overplotting

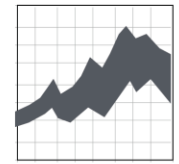


geom_violin()
- mirrored density plot (smoothed distribution)

Visualising Errors and Uncertainties

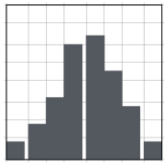


geom_errorbar()
- uncertainty in continuous Y against discrete X

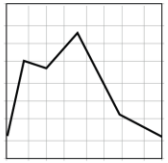


geom_ribbon()
- uncertainty in continuous Y against continuous X

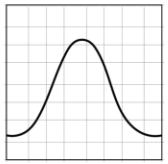
One Variable (X)
- Continuous X
- Visualise distribution of X



geom_histogram()
- divide X into bins and count no. observation

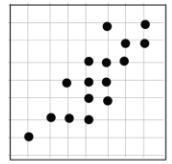


geom_freqpoly()
- display counts with lines
- able to overlay multiple distributions

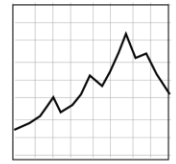


geom_density()
- smoothed version of the histogram

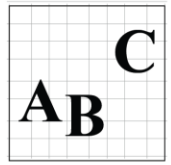
Two Variables (X,Y)
- Continuous X, continuous Y
- Visualise relationship between X and Y



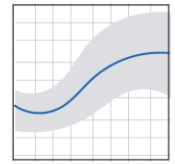
geom_point()
- scatterplot of X vs Y



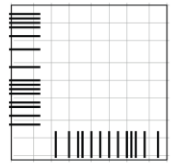
geom_line()
- connect data points, ordered by X
- alt: geom_path()



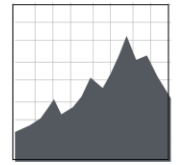
geom_text()
- labelling data points



geom_smooth()
- add smoothed curve
- helps to see trends

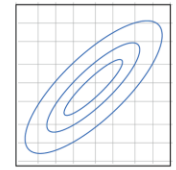


geom_rug()
- supplement 2D plot with 1D distribution along X and Y

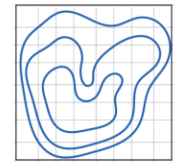


geom_area()
- can be stacked to see cumulative contribution

Contour Plots
- Representing a third dimension using contours

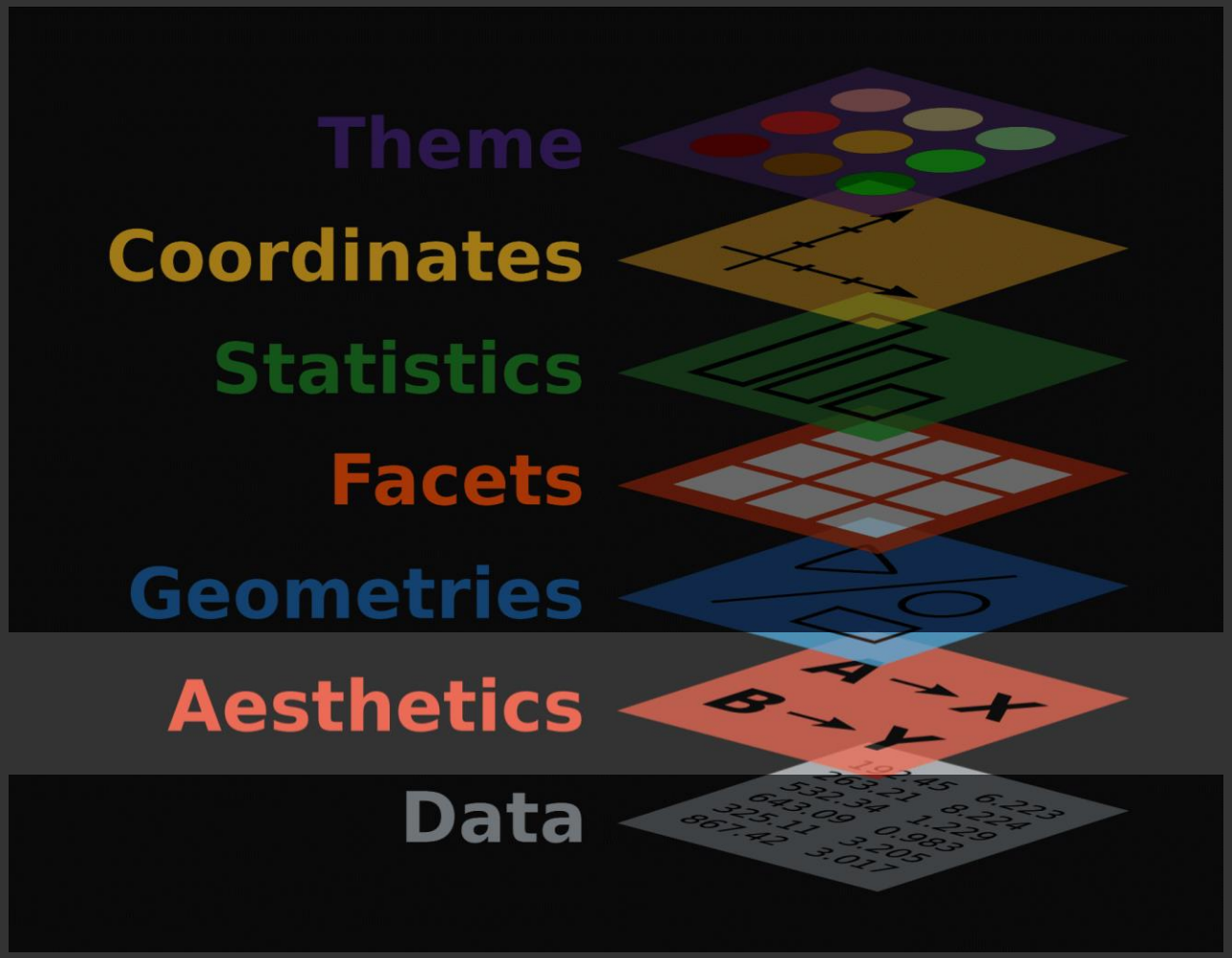


geom_density2d()
- contour represents 2D density of data points



geom_contour()
- contour represents z-axis value / height

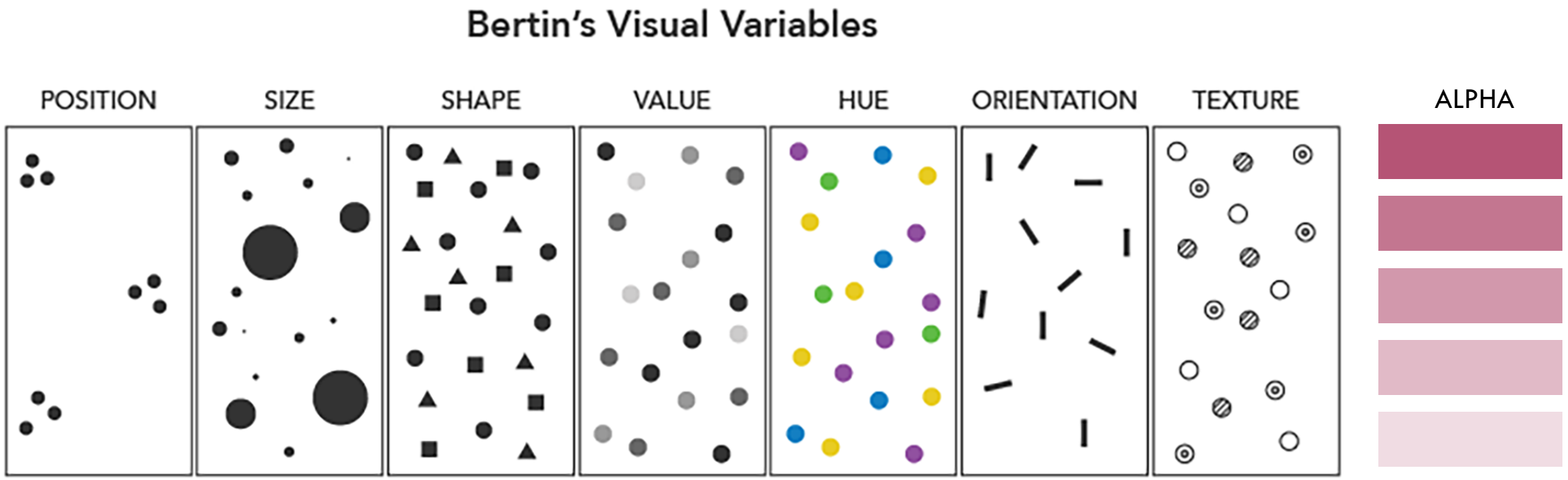
GGPLOT LAYERS



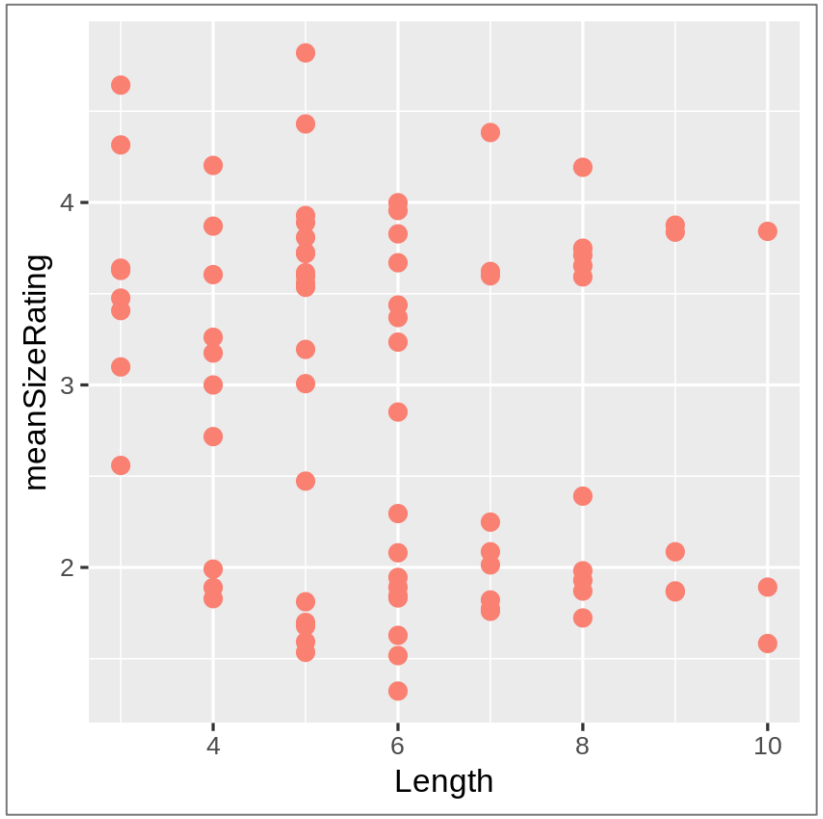
Aesthetics:

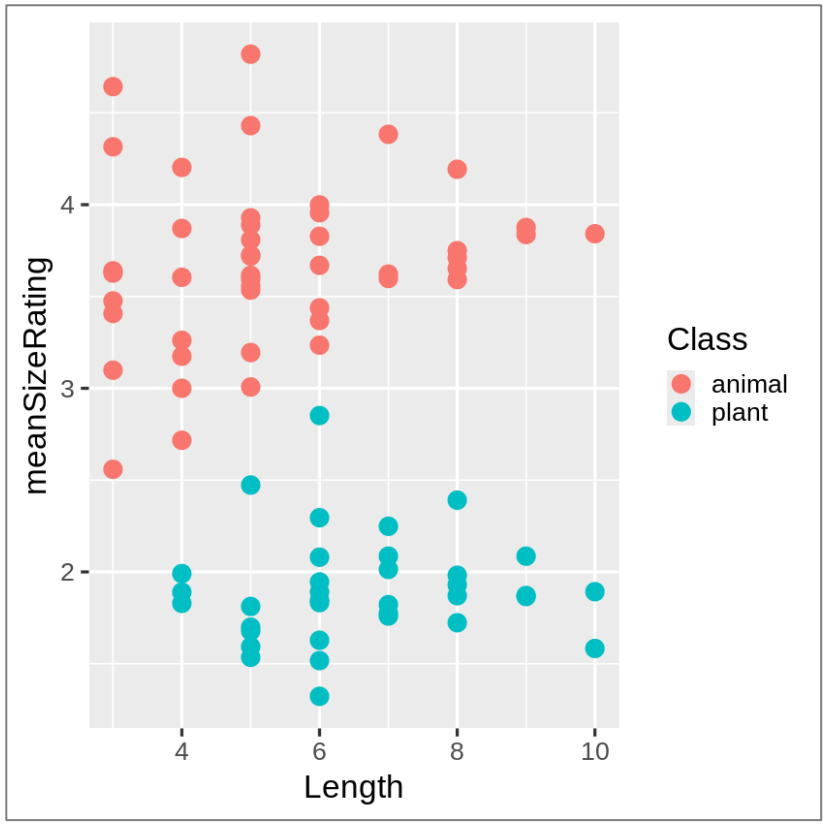
- Visual properties of geoms
- E.g. colour, shape, size, and transparency.

AESTHETIC: VISUAL PROPERTIES



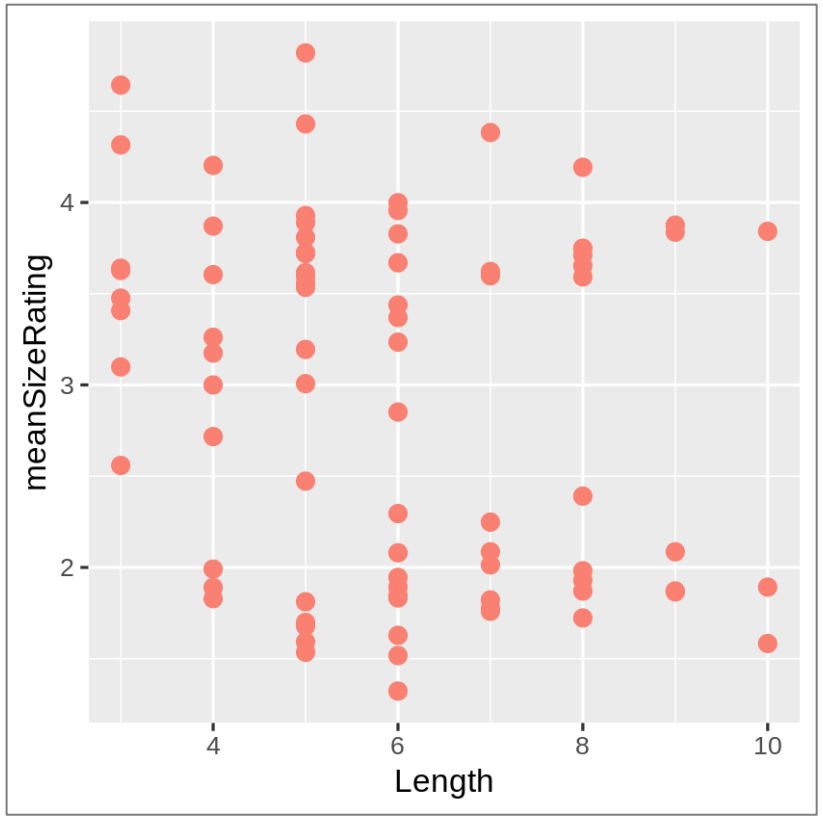
AESTHETICS: VISUAL PROPERTIES



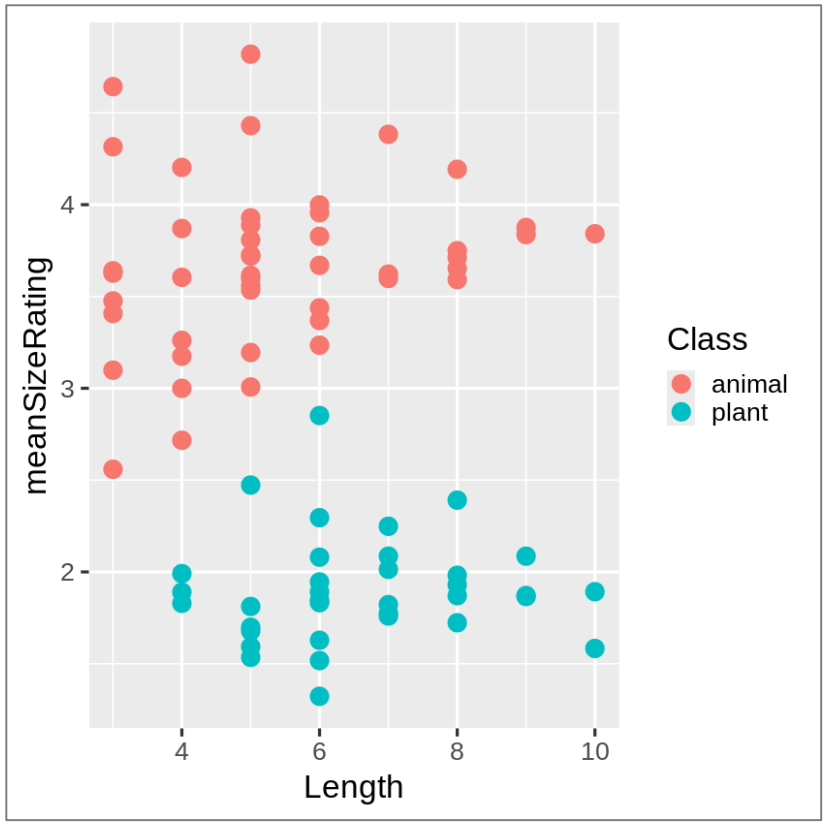


AESTHETICS: VISUAL PROPERTIES

Colour aesthetic **set** to `"salmon"`



Colour aesthetic **mapped** to `Class`



AESTHETICS: VISUAL PROPERTIES

Colour aesthetic **set** to `"salmon"`

When we **set** an aesthetic:

- We give R a **set** value to use across the board
- Colour doesn't represent anything about the data

Colour aesthetic **mapped** to `Class`

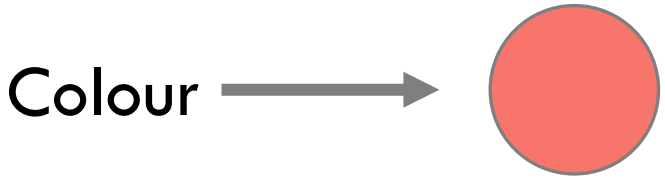
When we **map** an aesthetic:

- We give R a **variable** to encode using the aesthetic
- Colour represents/corresponds to/encodes different values of the variable

AESTHETICS: VISUAL PROPERTIES

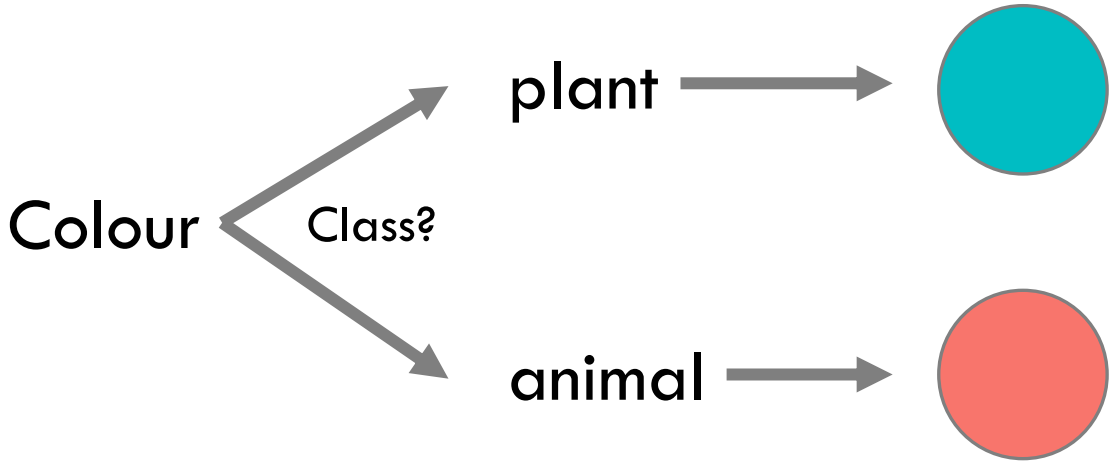
Colour aesthetic **set** to `"salmon"`

When we **set** an aesthetic:



Colour aesthetic **mapped** to `Class`

When we **map** an aesthetic:



AESTHETICS: VISUAL PROPERTIES

Colour aesthetic **set** to `"salmon"`

When we **set** an aesthetic:

- We don't use the `aes()` function

Colour aesthetic **mapped** to `Class`

When we **map** an aesthetic:

STRUCTURE OF GGPLOT2 CODE

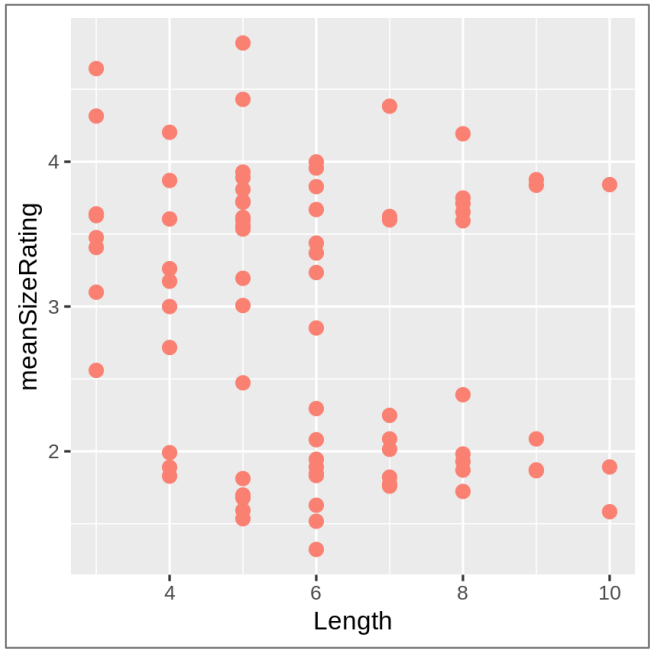
```
ggplot(
      data = ratings,
      mapping = aes(x=Length, y=meanSizeRating)

)+

   geom_point(colour="salmon")
```

data

aesthetic mappings

geom

aesthetic setting



AESTHETICS: VISUAL PROPERTIES

Colour aesthetic **set** to `"salmon"`

When we **set** an aesthetic:

- We don't use the `aes()` function

Colour aesthetic **mapped** to `Class`

When we **map** an aesthetic:

- We **do** use the `aes()` function

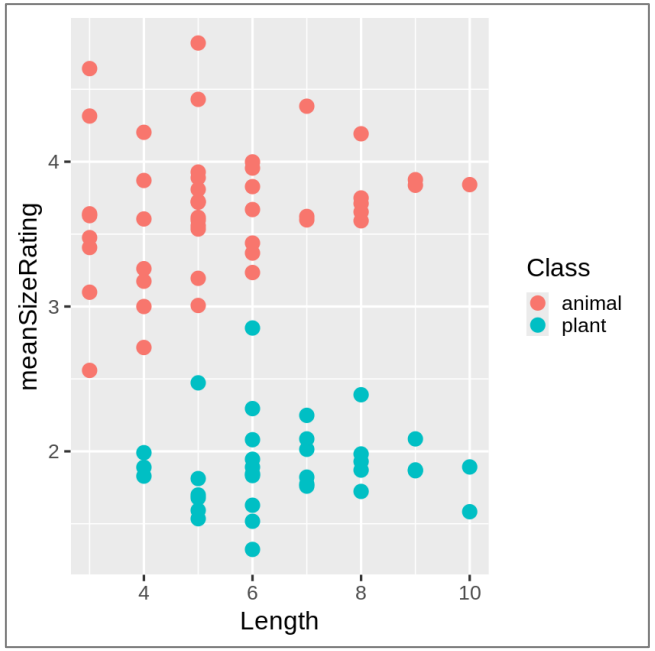
MAPPING VS. SETTING AESTHETICS

```
ggplot(
      data = ratings,
      mapping = aes(x=Length, y=meanSizeRating)
)+
   geom_point(aes(colour=Class))
```

data

aesthetic mappings

geom



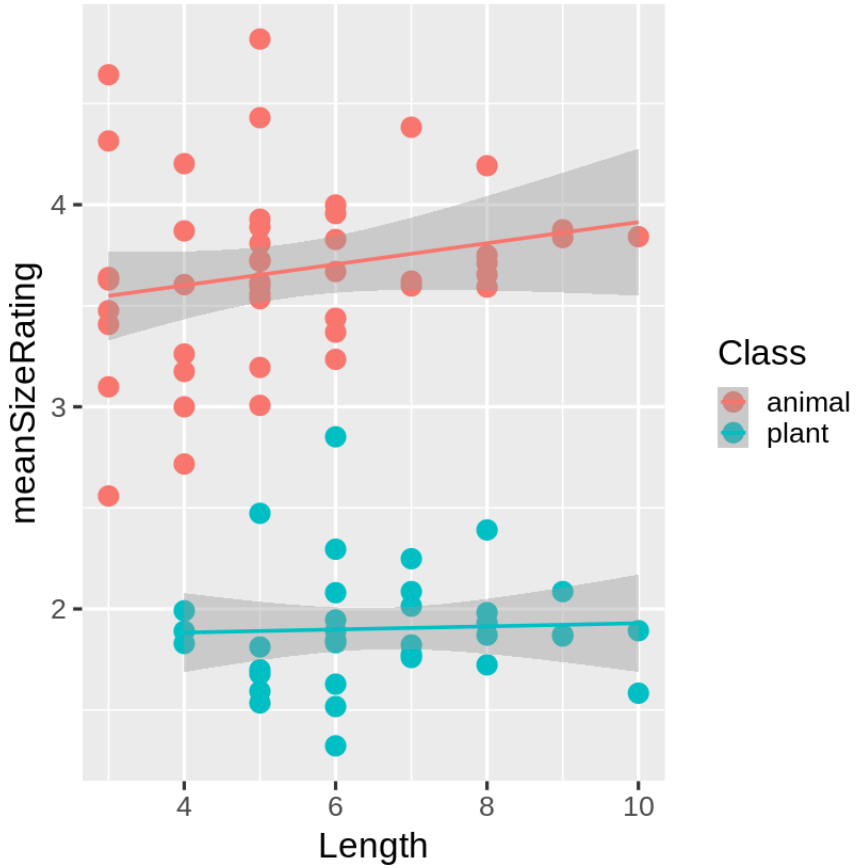
AESTHETICS: LOCAL VS. GLOBAL

- A further distinction for aesthetics is **local** vs. **global**
- **Global aesthetics** are shared by all geoms
- **Local aesthetics** are specific to one geom

MAPPING AESTHETICS: LOCAL VS. GLOBAL

```
ggplot(
  ratings,
  aes(x=Length, y=meanSizeRating, colour=Class)
)+
  geom_point(size=5)+
  geom_smooth(method="lm")
```

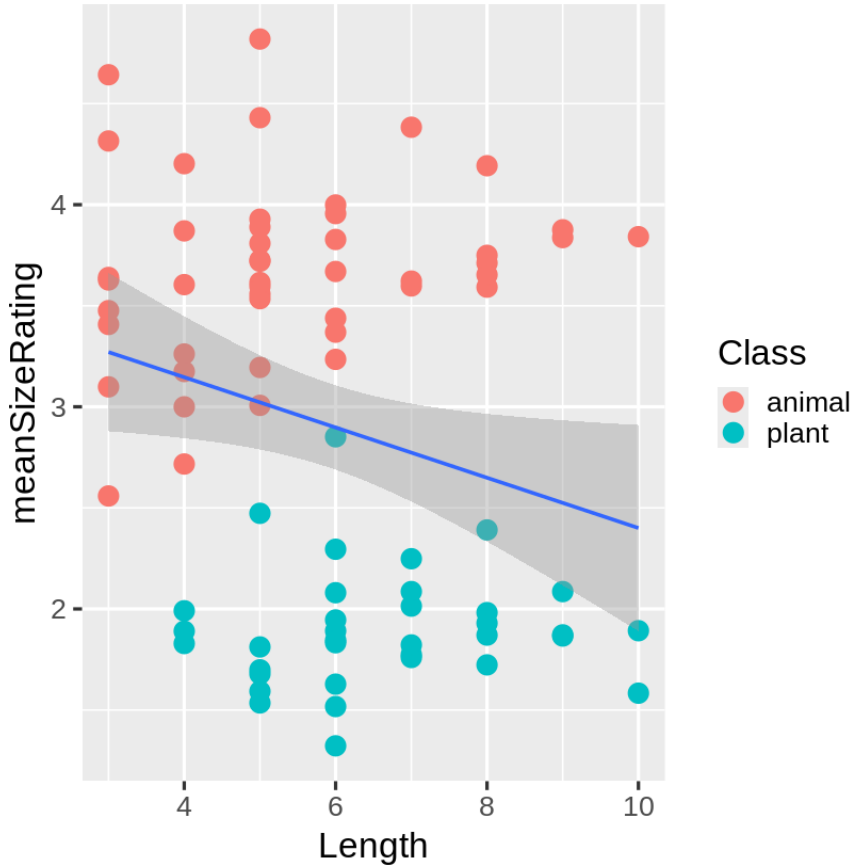
`colour=Class` is a **global** mapping – it applies to all geoms (`geom_smooth` and `geom_point`)



MAPPING AESTHETICS: LOCAL VS. GLOBAL

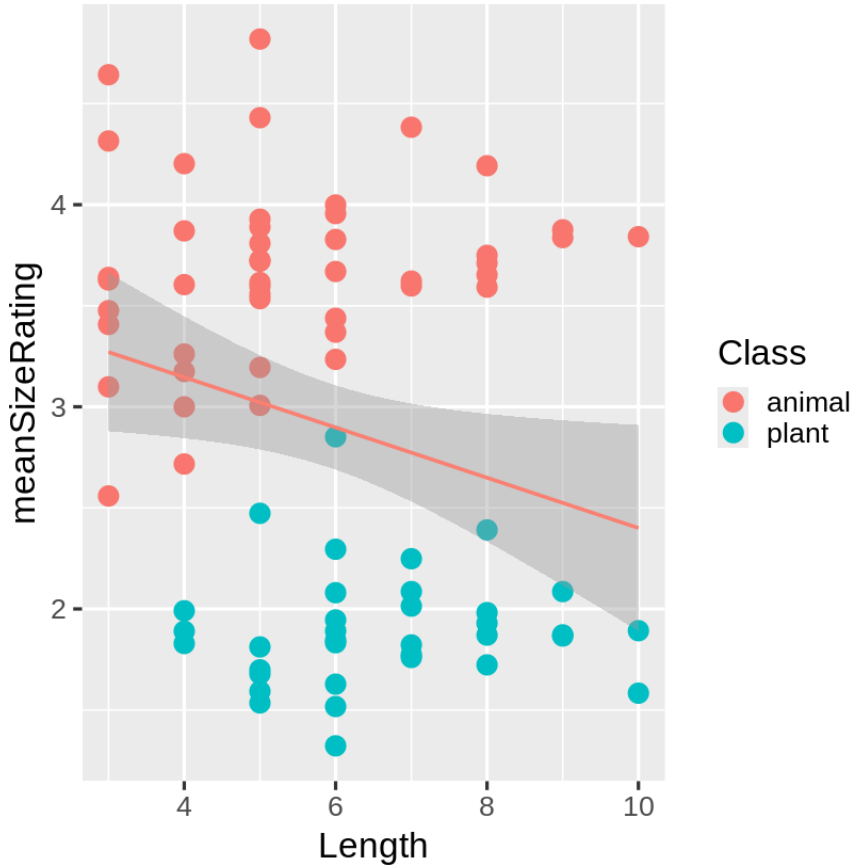
```
ggplot(
  ratings,
  aes(x=Length, y=meanSizeRating)
)+
  geom_point(aes(colour=Class), size=5)+
  geom_smooth(method="lm")
```

`colour=Class` is a **local** mapping – it applies only to the geom `geom_point`



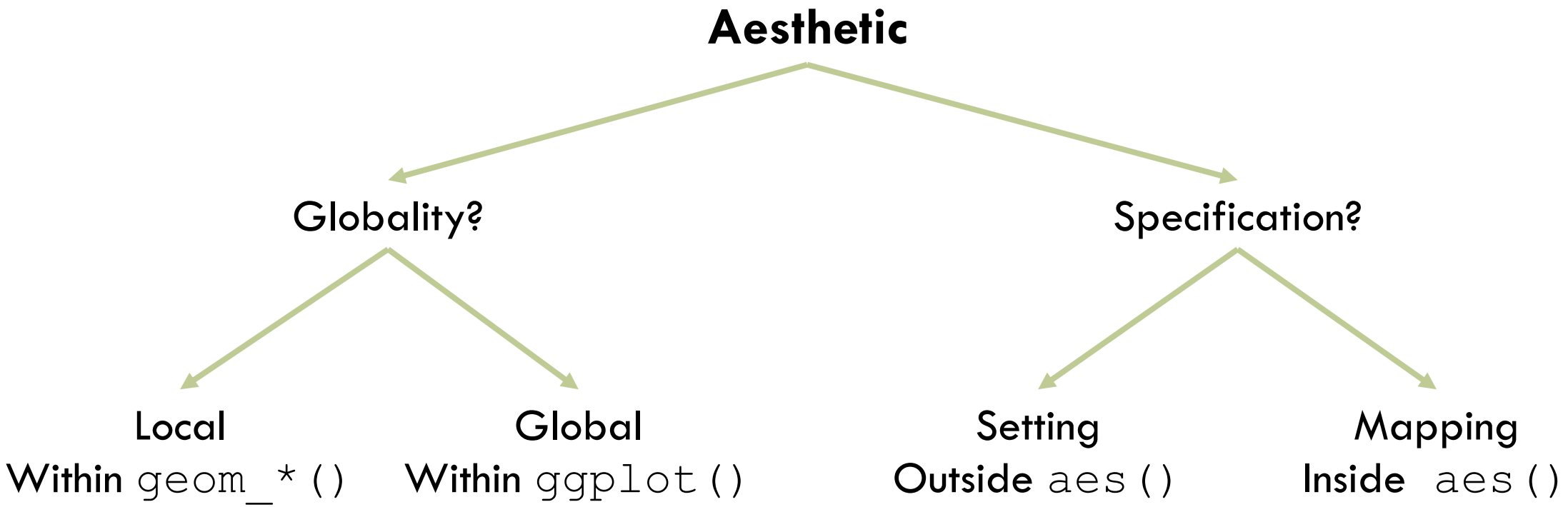
MAPPING AESTHETICS: LOCAL VS. GLOBAL

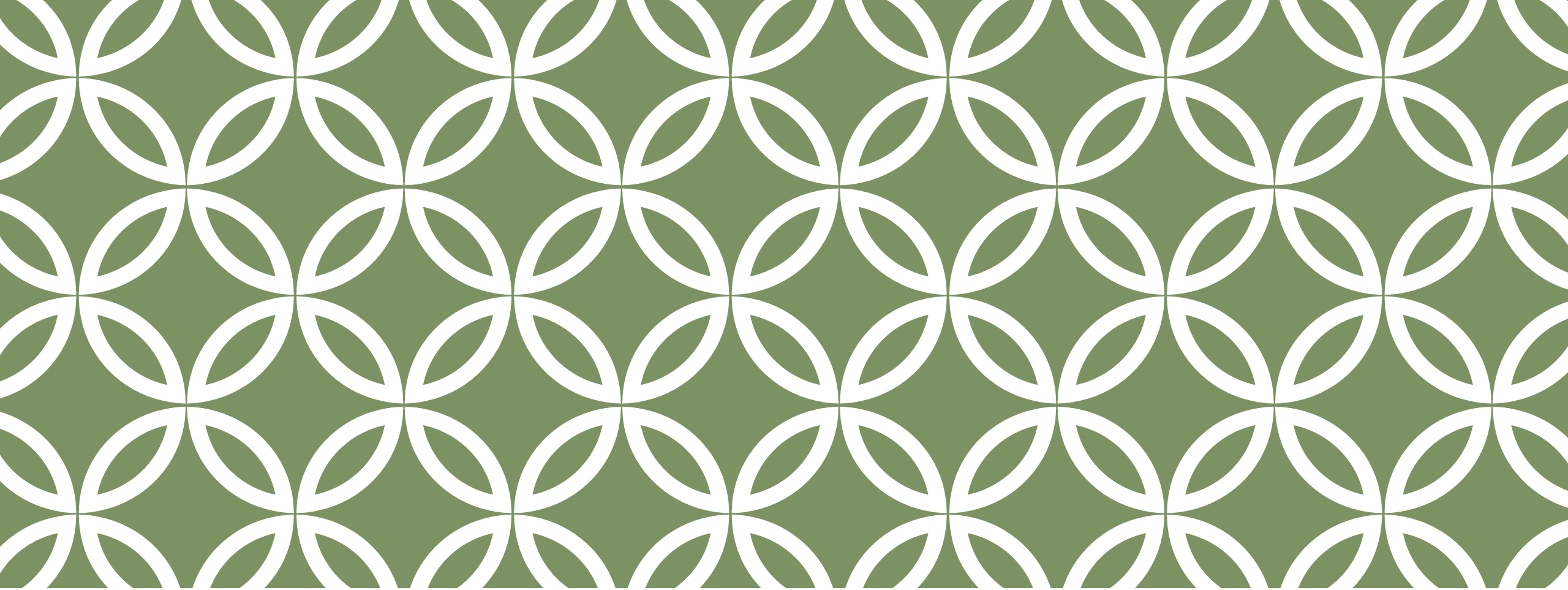
```
ggplot(
  ratings,
  aes(x=Length, y=meanSizeRating, colour=Class)
)+
  geom_point(size=5)+
  geom_smooth(method="lm", colour="salmon")
```



- `colour=Class` is a **global** mapping – it should apply to all geoms.
- However, `colour="salmon"`, a local setting, overrides the global mapping.
- So `colour=Class` only applies to `geom_point`.

AESTHETICS: SUMMARY





LAB EXERCISES

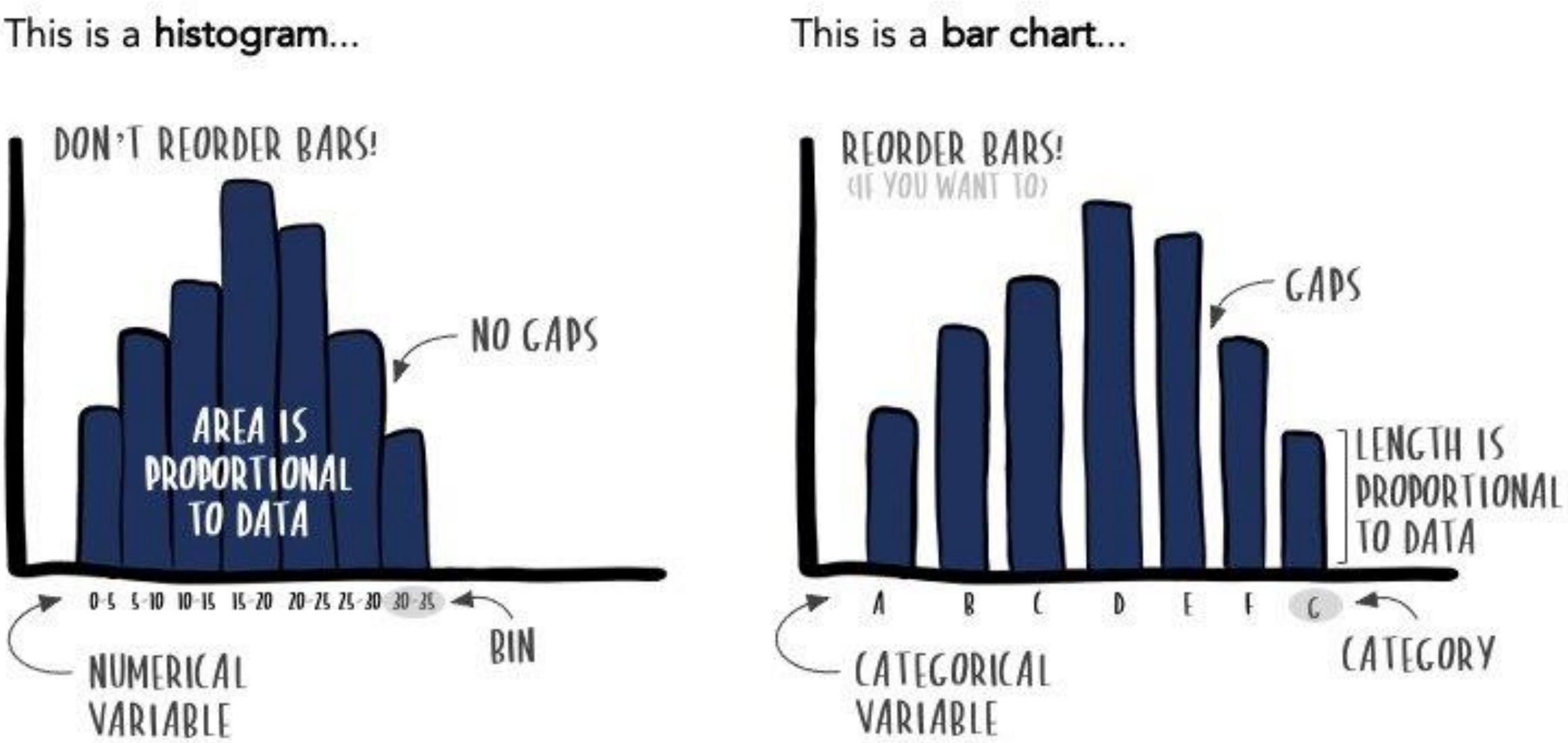
Q2.3

```
ggplot(
    data = ratings,
    mapping = aes(x = Frequency, y = meanFamiliarity)
) +
    geom_point(mapping = aes(color = Class)) +
    geom_smooth(method = "lm") +
    theme_classic(base_size=20)
```

Global mapping: all geoms will inherit these mappings

Local mapping: only geom_point will have this mapping

Q2.6, Q2.7



Source: https://www.storytellingwithdata.com/blog/2021/1/28/histograms-and-bar-charts

FACET_WRAP AND FACET_GRID

- **Faceting:** splitting a plot into smaller panels called *facets*

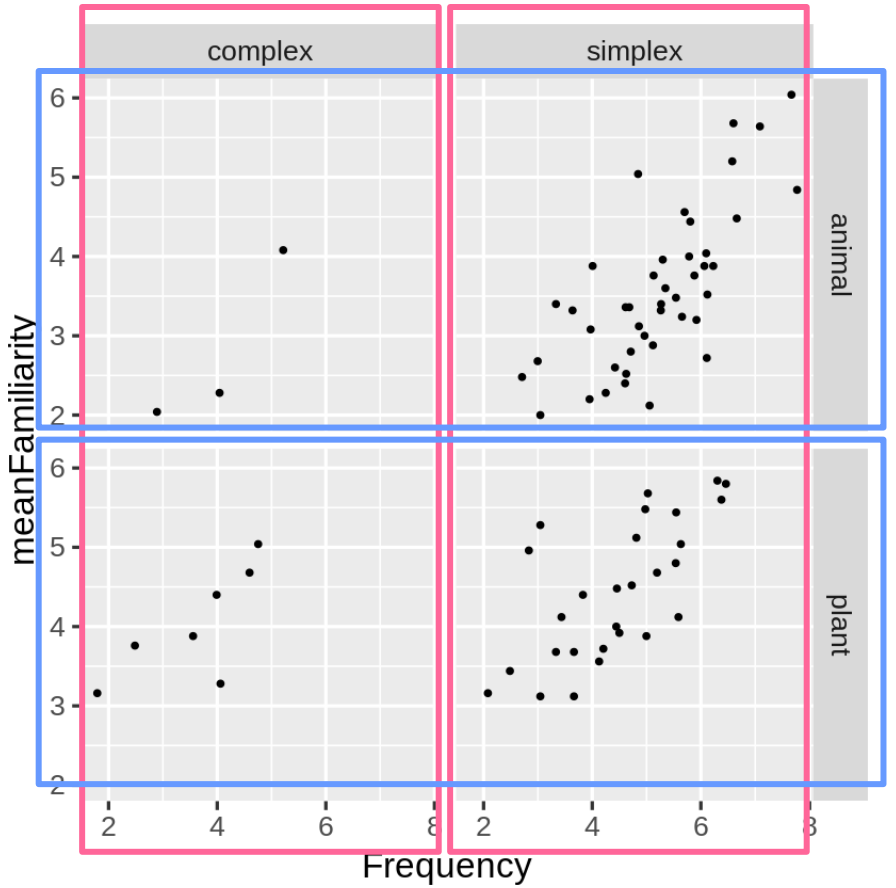
- `facet_wrap()` **and** `facet_grid()` **are two ways to achieve this**
  - Both take arguments indicating *how* to split the plot
  - But they differ in how they lay out the facets

facet_grid()

```
ggplot(ratings, aes(x=Frequency, y=meanFamiliarity))+
  geom_point()+
  facet_grid(Class~Complex)+
  theme_gray(base_size=20)
```

- `facet_grid()` splits the plot into facets for each combination of values of Class and Complex

- The facets are arranged in such a way that the columns (pink) represent values of Complex, and rows (blue) represent values of Class

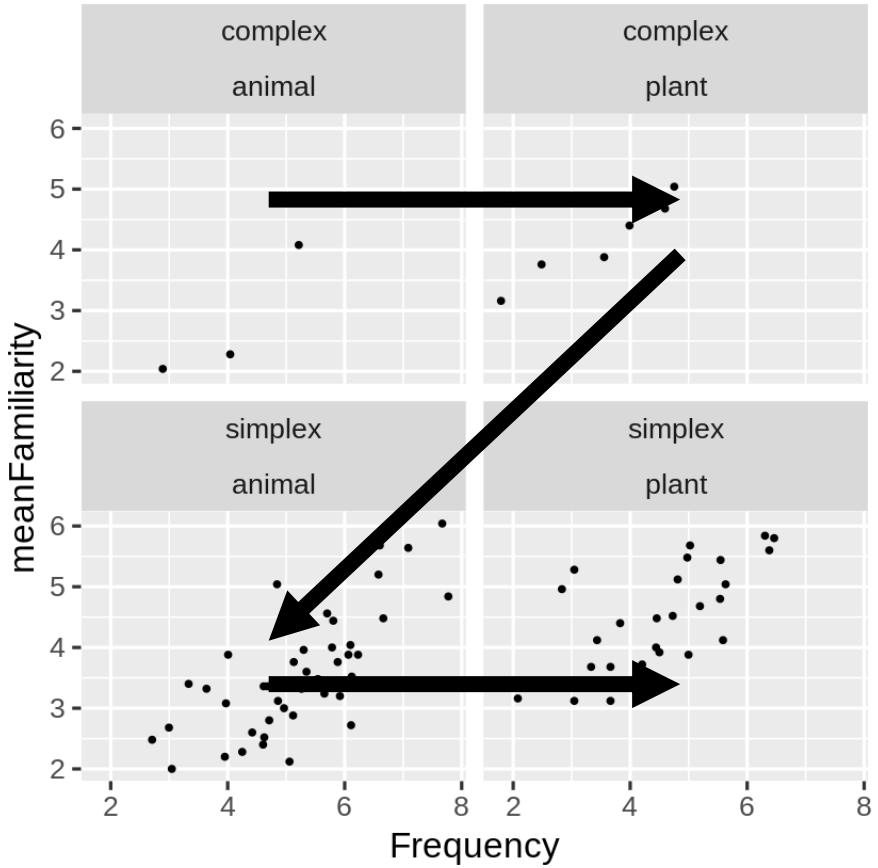


facet_wrap()

```
ggplot(ratings, aes(x=Frequency, y=meanFamiliarity))+
  geom_point()+
  facet_grid(Class~Complex)+
  theme_gray(base_size=20)
```

- `facet_wrap()` behaves similarly in the way it splits the plot, but it arranges the facets in a different way

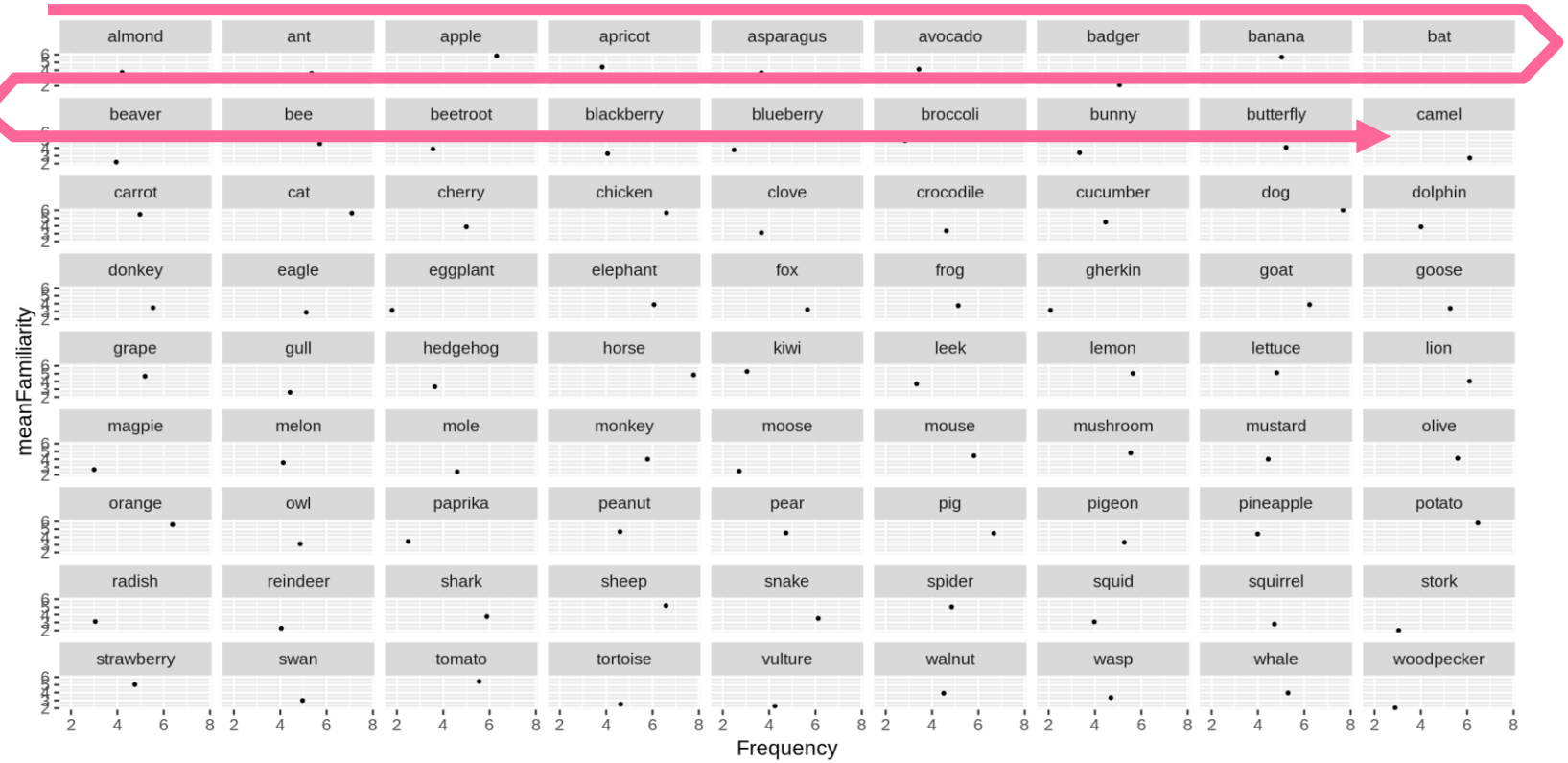
- Here, we can imagine the facets lined up next to each other in a single file, which is then made to fit into the available space by "wrapping" it onto subsequent rows.



facet_wrap()

```
options(repr.plot.width = 20, repr.plot.height = 10)
ggplot(ratings, aes(x=Frequency, y=meanFamiliarity))+
  geom_point()+
  facet_wrap(~ Word)+
  theme_gray(base_size = 20)
```

- The behaviour of `facet_wrap()` is more appreciable when the faceting variable has many levels:

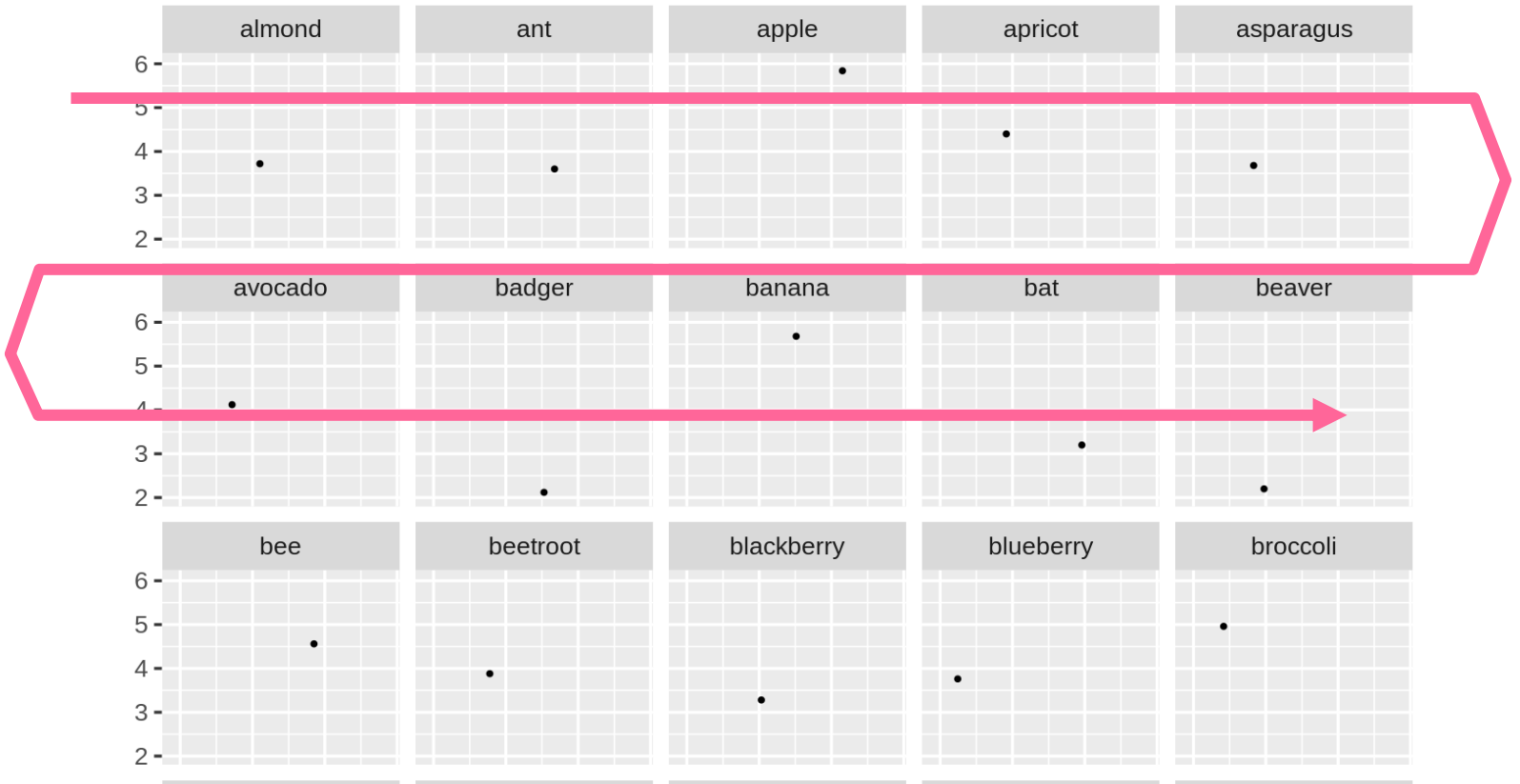


`ggplot()` fills the horizontal space with as many facets as possible before wrapping to the next line

facet_wrap()

```
options(repr.plot.width = 12, repr.plot.height = 40)
ggplot(ratings, aes(x=Frequency, y=meanFamiliarity))+
  geom_point()+
  facet_wrap(~ Word, ncol=5)+
  theme_gray(base_size = 20)
```

- The behaviour of `facet_wrap()` is more appreciable when the faceting variable has many levels:



You can also control the behaviour of `facet_wrap()` by specifying the number of columns (`ncol=`) or rows (`nrow=`) you want.

Here, `ncol=5`, so wrapping occurs after every 5th facet.