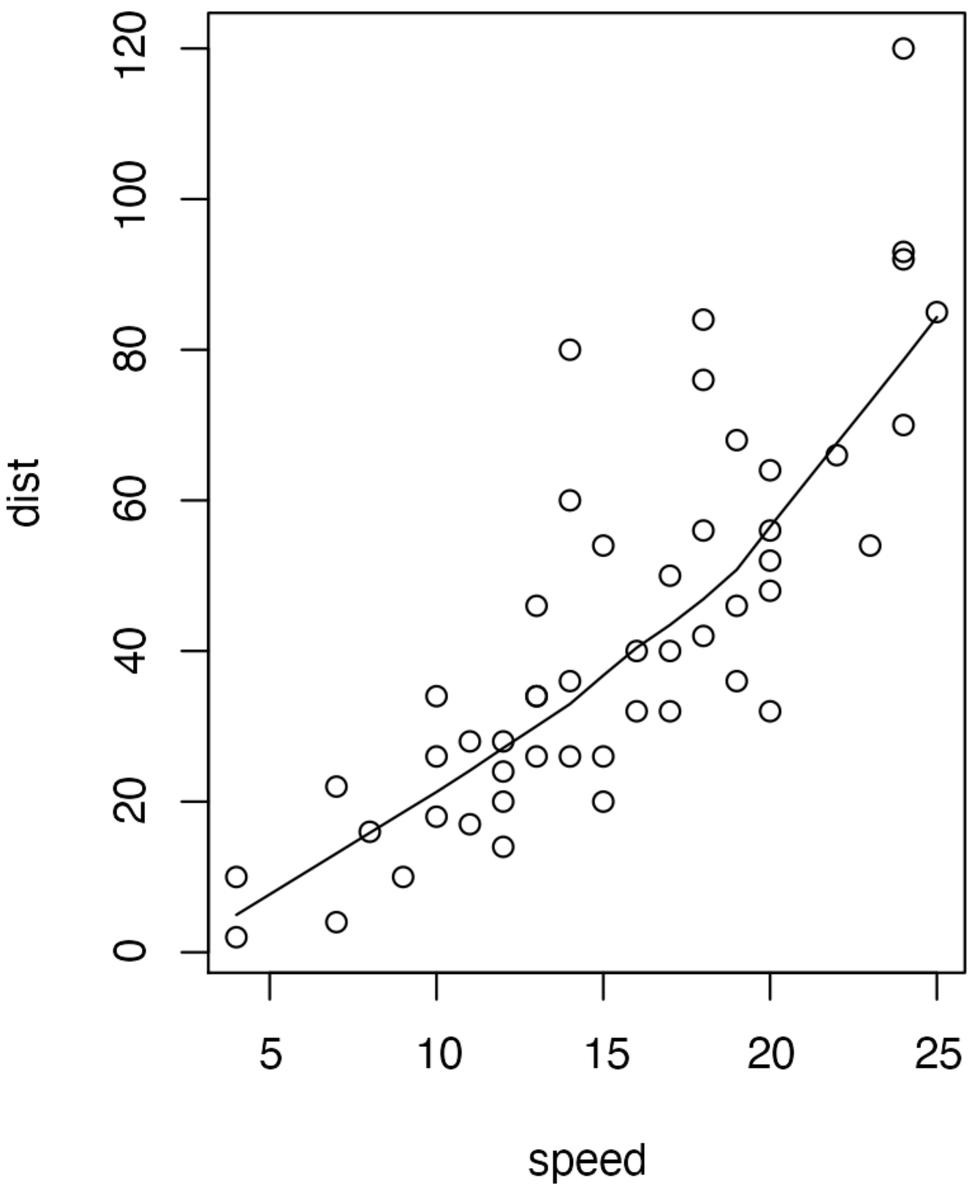
Introduction to GGPlot2

- Slides are summarized from excellent presentation of Thomas Lin Pedersen
- See his github repo: https://github.com/thomasp85/ggplot2_workshop
- See his video

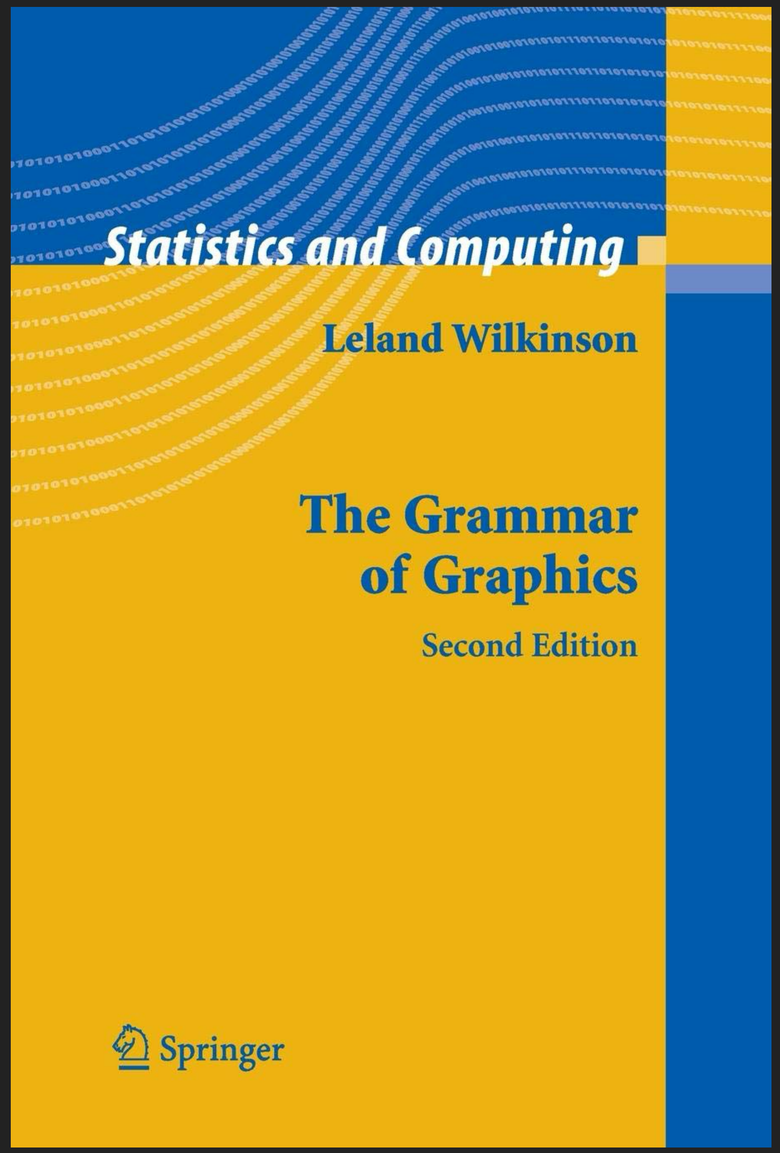


PART 1:

THE GRAMMAR OF GRAPHICS

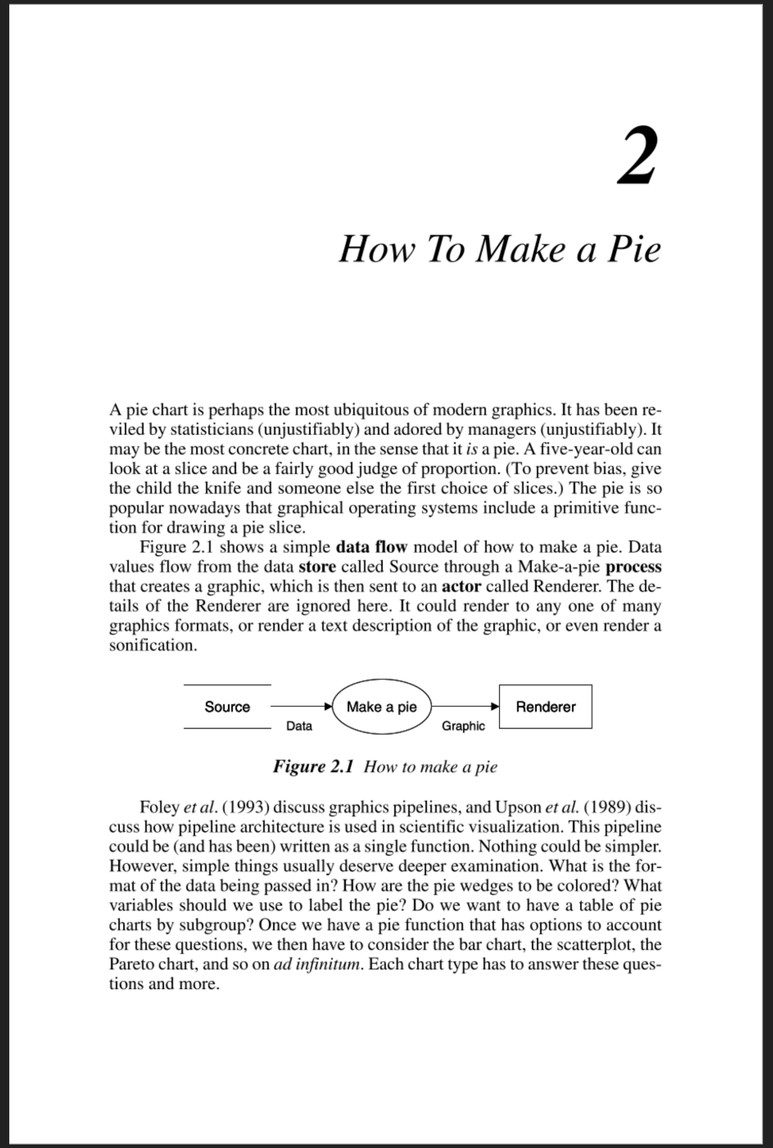
THE BOOK

- 1st edition in 1999
- A theoretical deconstruction of data graphics
- Foundation for many graphic applications
  - ggplot2
  - Polaris (→ Tableau)
  - Vega-Lite



THE BOOK

- Not interested in why
  - *why choose this or that chart type?*
- Not interested in beauty
  - *how do you make attractive charts?*
- Not interested in algorithms
  - *how do you calculate the correct positions of the data?*
- Very interested in how to design the system that allows all that



2

How To Make a Pie

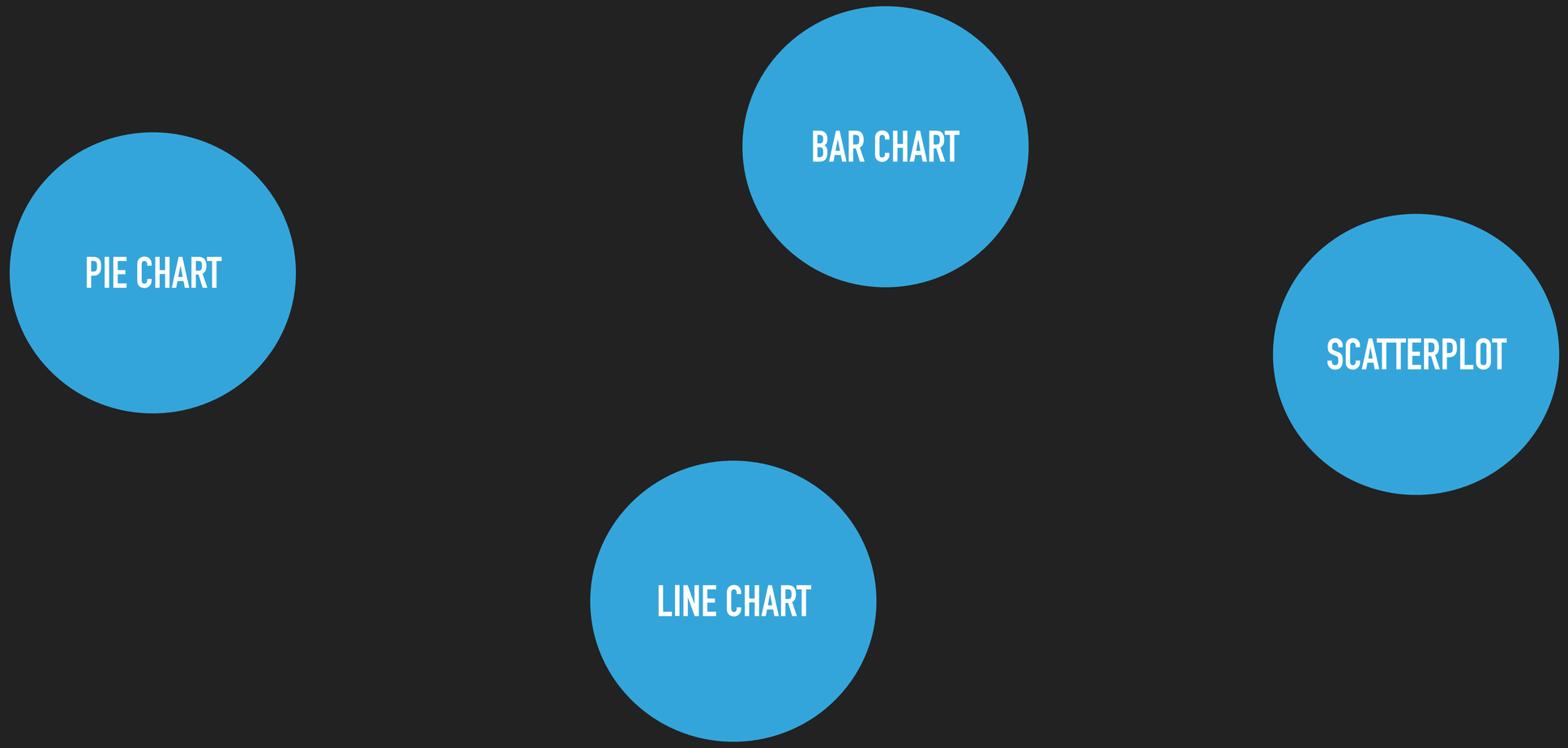
A pie chart is perhaps the most ubiquitous of modern graphics. It has been reviled by statisticians (unjustifiably) and adored by managers (unjustifiably). It may be the most concrete chart, in the sense that it *is* a pie. A five-year-old can look at a slice and be a fairly good judge of proportion. (To prevent bias, give the child the knife and someone else the first choice of slices.) The pie is so popular nowadays that graphical operating systems include a primitive function for drawing a pie slice.

Figure 2.1 shows a simple **data flow** model of how to make a pie. Data values flow from the data **store** called Source through a Make-a-pie **process** that creates a graphic, which is then sent to an **actor** called Renderer. The details of the Renderer are ignored here. It could render to any one of many graphics formats, or render a text description of the graphic, or even render a sonification.

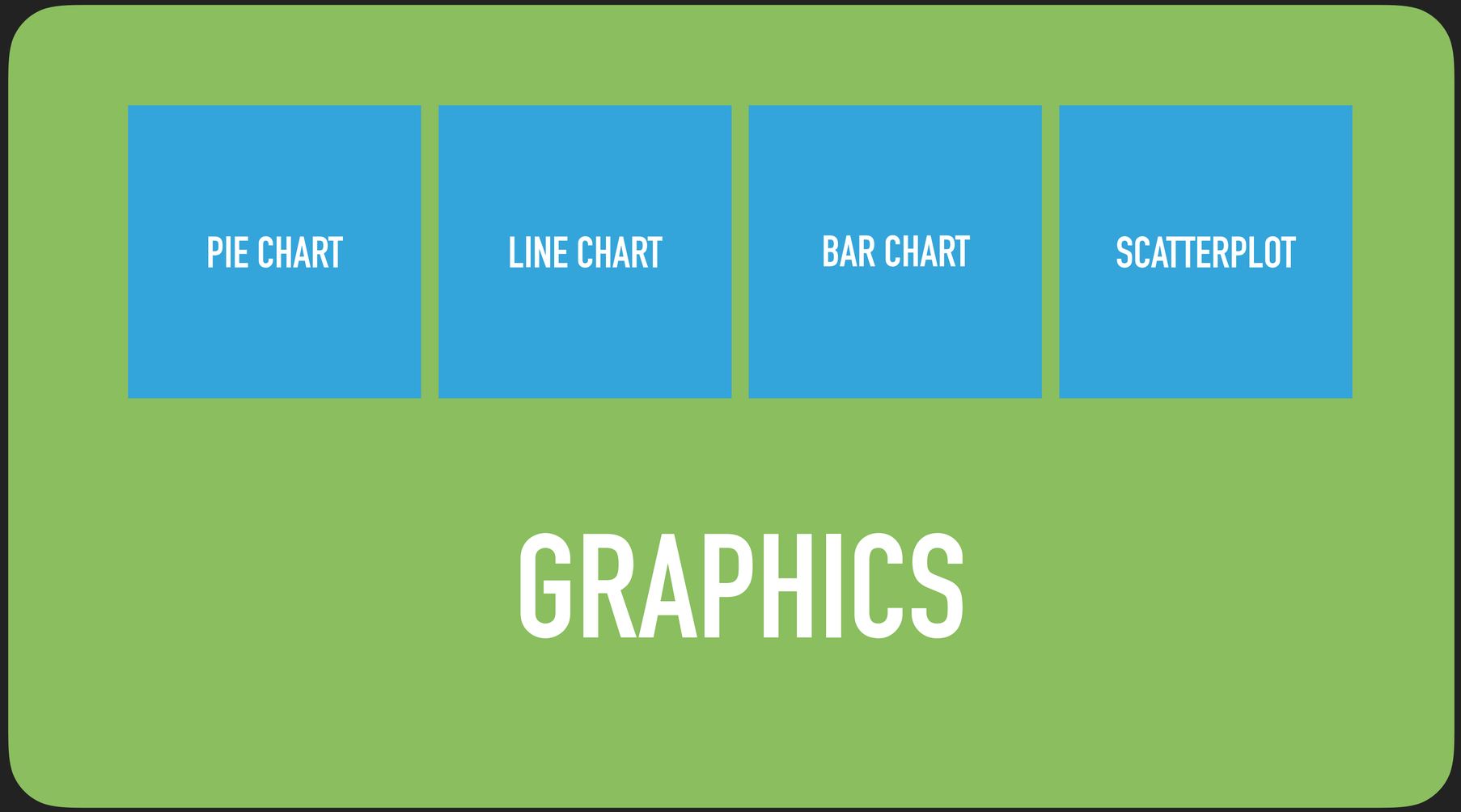
Figure 2.1 *How to make a pie*

Foley *et al.* (1993) discuss graphics pipelines, and Upson *et al.* (1989) discuss how pipeline architecture is used in scientific visualization. This pipeline could be (and has been) written as a single function. Nothing could be simpler. However, simple things usually deserve deeper examination. What is the format of the data being passed in? How are the pie wedges to be colored? What variables should we use to label the pie? Do we want to have a table of pie charts by subgroup? Once we have a pie function that has options to account for these questions, we then have to consider the bar chart, the scatterplot, the Pareto chart, and so on *ad infinitum*. Each chart type has to answer these questions and more.

THE IDEA



THE IDEA



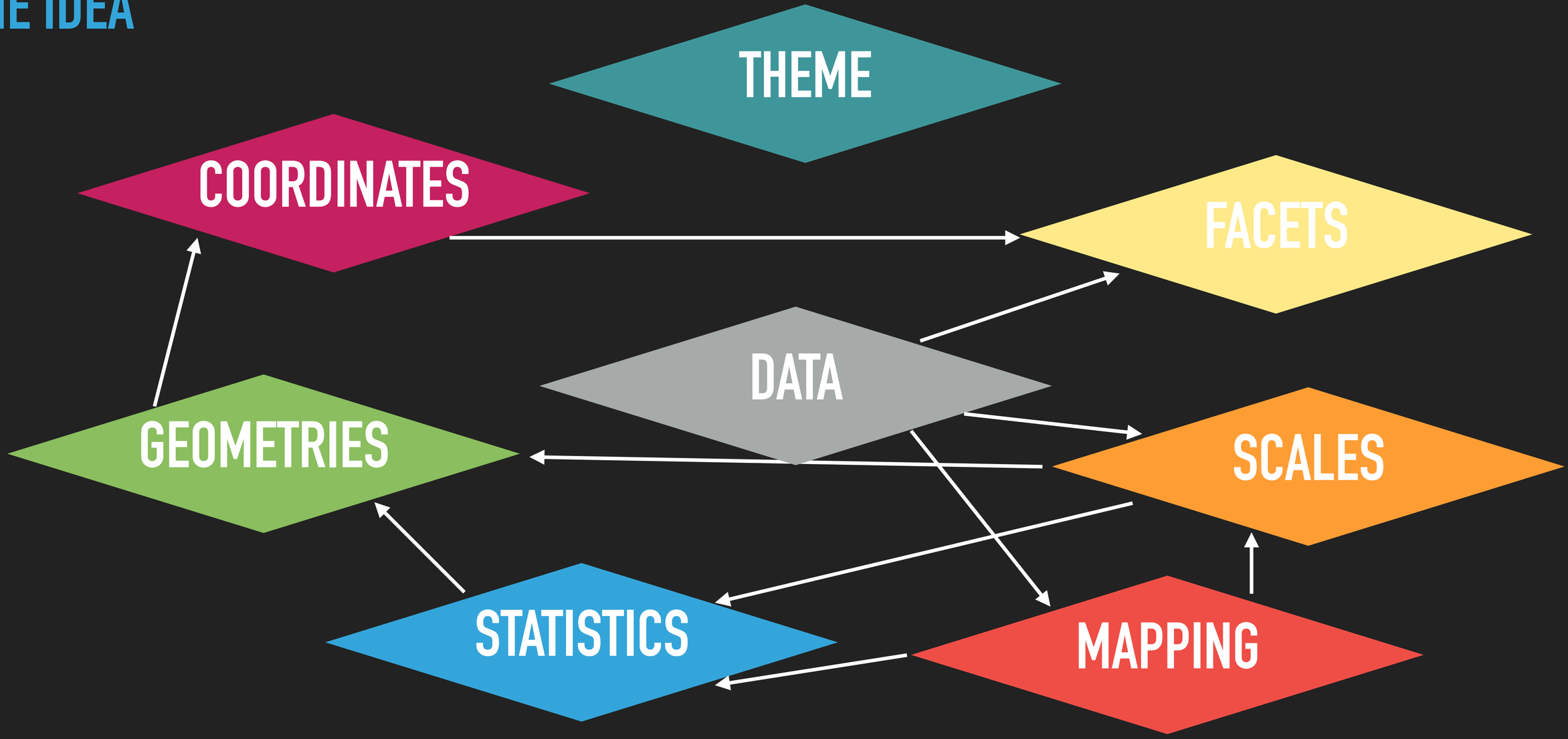
THE IDEA

WHAT
IS
GRAPHICS?

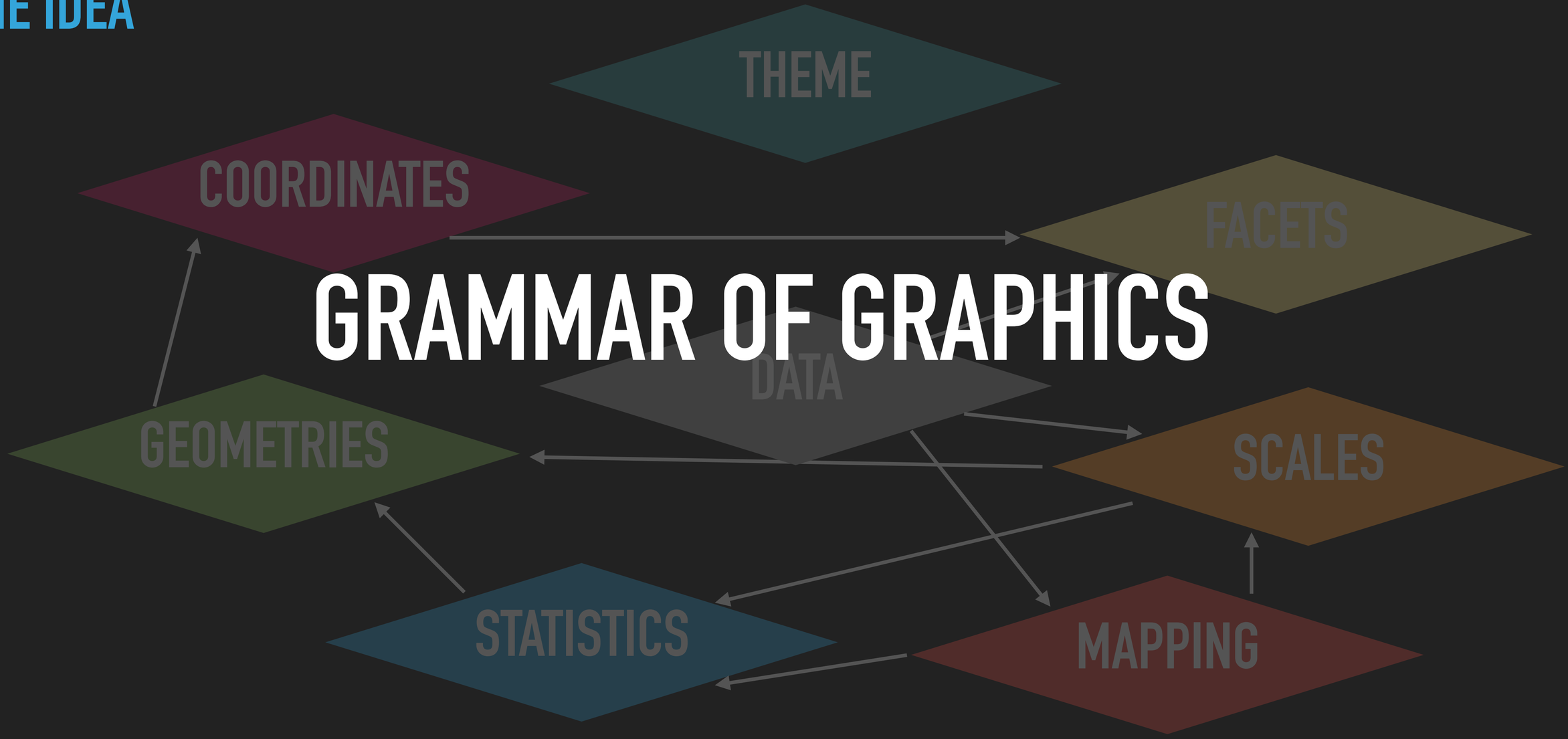
THE IDEA

DECOMPOSE GRAPHICS INTO ITS CONSTITUENTS

THE IDEA


THEME
COORDINATES
FACETS
DATA
GEOMETRIES
SCALES
STATISTICS
MAPPING

THE IDEA


THEME
COORDINATES
FACETS
GRAMMAR OF GRAPHICS
DATA
GEOMETRIES
SCALES
STATISTICS
MAPPING

THE IDEA

PIE CHART

BAR CHART

LINE CHART

SCATTERPLOT

VS

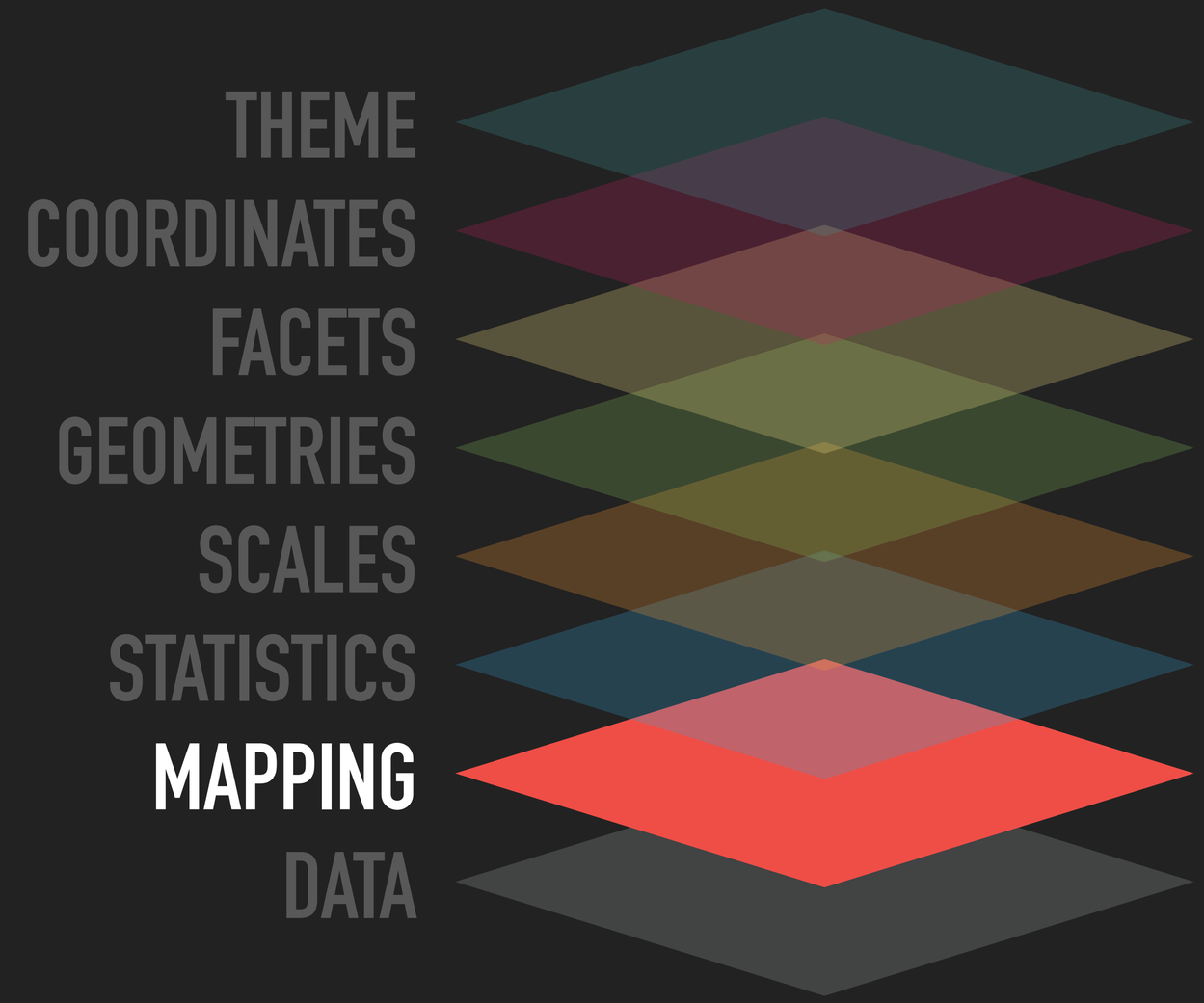
THEME
COORDINATES
FACETS
GEOMETRIES
SCALES
STATISTICS
MAPPING
DATA

THE GRAMMAR

- Data is not just data
- Representation defines what can be done with it
- Grammar requires a tidy format (though it precedes the notion)
- We will not talk more about it but it is very important 😬

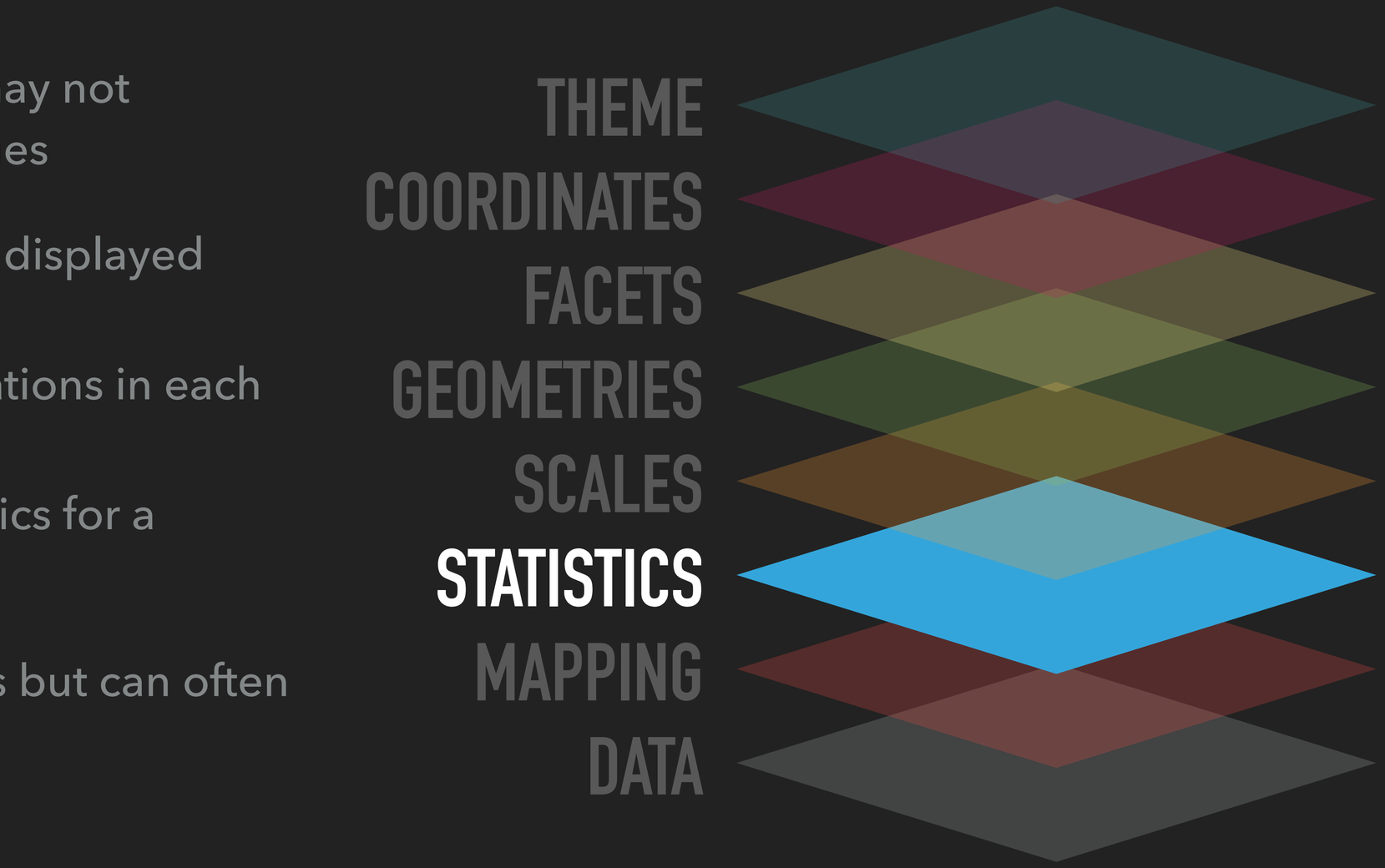
THE GRAMMAR

- Allow generic datasets to be understood by the graphic system.
- *Aesthetic* mapping: Link variables in data to graphical properties in the geometry.
- *Facet* mapping: Link variables in the data to panels in the facet layout.



THE GRAMMAR

- Even though data is tidy it may not represent the displayed values
- Transform input variables to displayed values:
  - Count number of observations in each category for a bar chart
  - Calculate summary statistics for a boxplot.
- Is implicit in many plot-types but can often be done prior to plotting

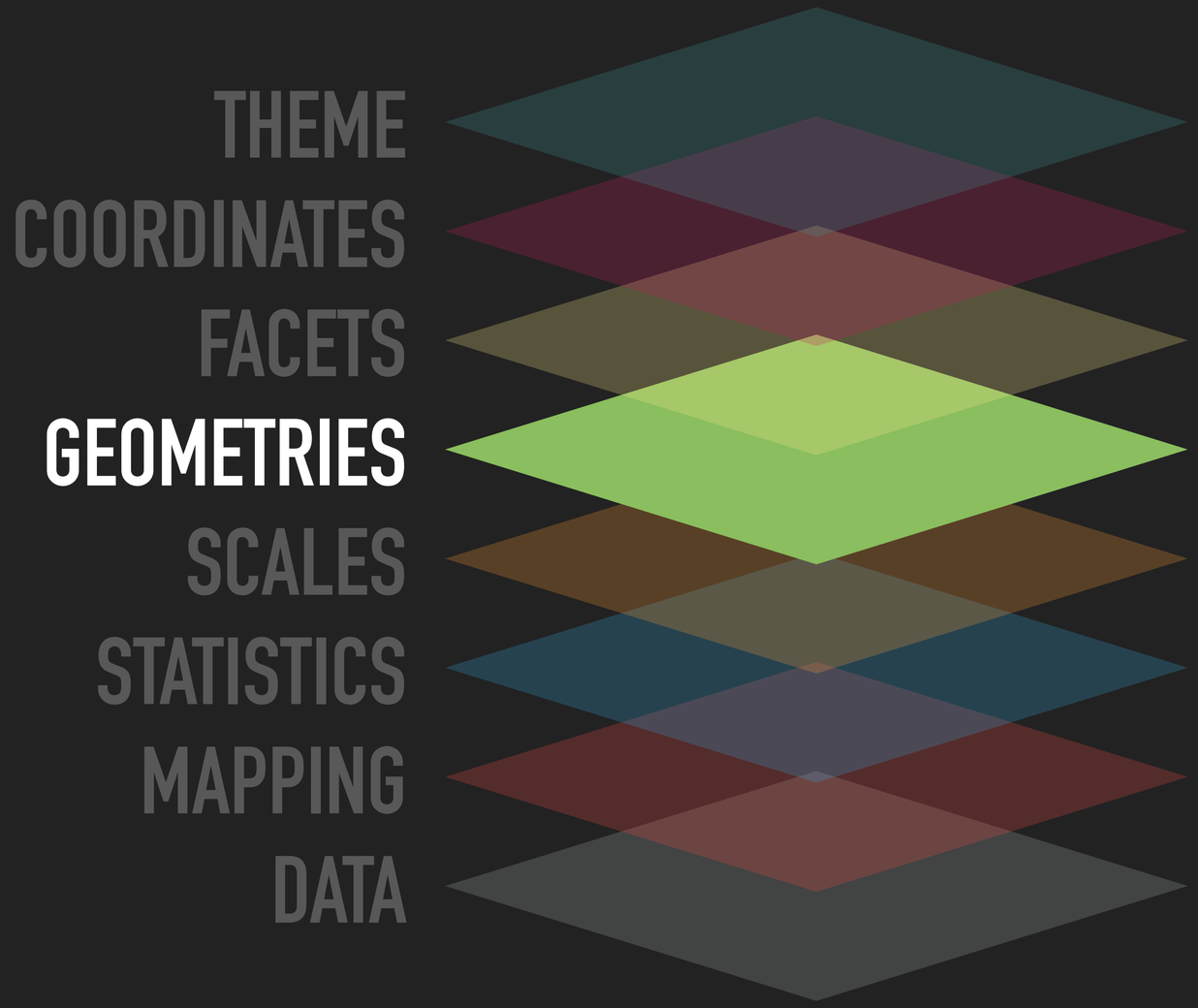


THE GRAMMAR

- Most data does not directly represent graphical properties.
- A scale translate back and forth between variable ranges and property ranges
  - Categories → Colour
  - Numbers → Position
  - …
- Imply a specific interpretation of values; discrete, continuous, etc.

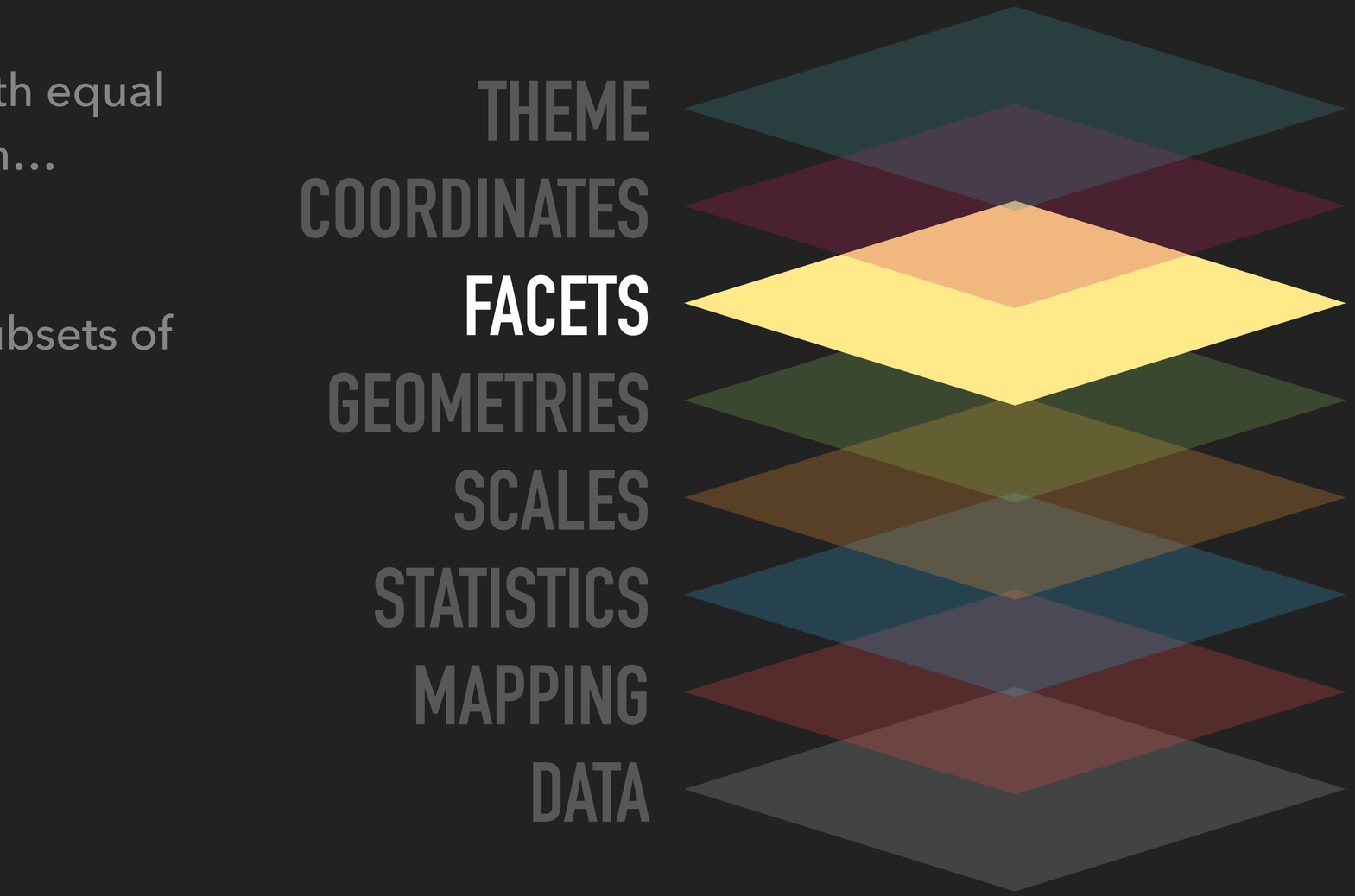
THE GRAMMAR

- How to interpret aesthetics as graphical representations
- Is a progression of positional aesthetics a number of points, a line, a single polygon, or something else entirely?
- To a high degree the determinator of your plot type



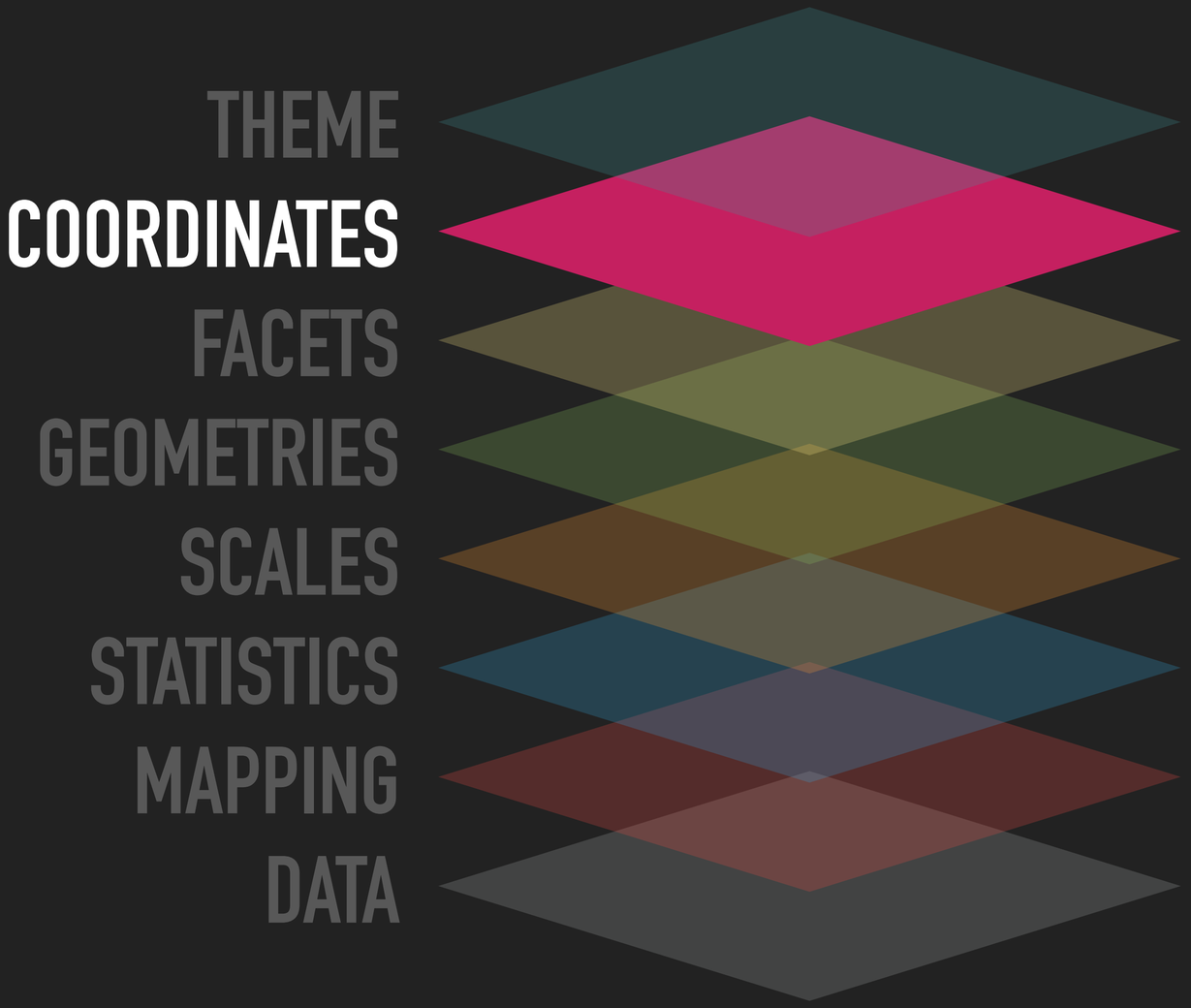
THE GRAMMAR

- Define the number of panels with equal logic and split data among them…
- Small multiples
  - Allows you to look at small subsets of your data in separate plots
- Panel layout may carry meaning



THE GRAMMAR

- Positional aesthetics are special.
  1. Variables are mapped, scaled, applied to a geometry
  2. But in the end, the position values are interpreted by a coordinate system
- Defines the physical mapping of the aesthetics to the paper
- Vaguely similar to colour profile mapping for colour aesthetics (though you don't have control over it)

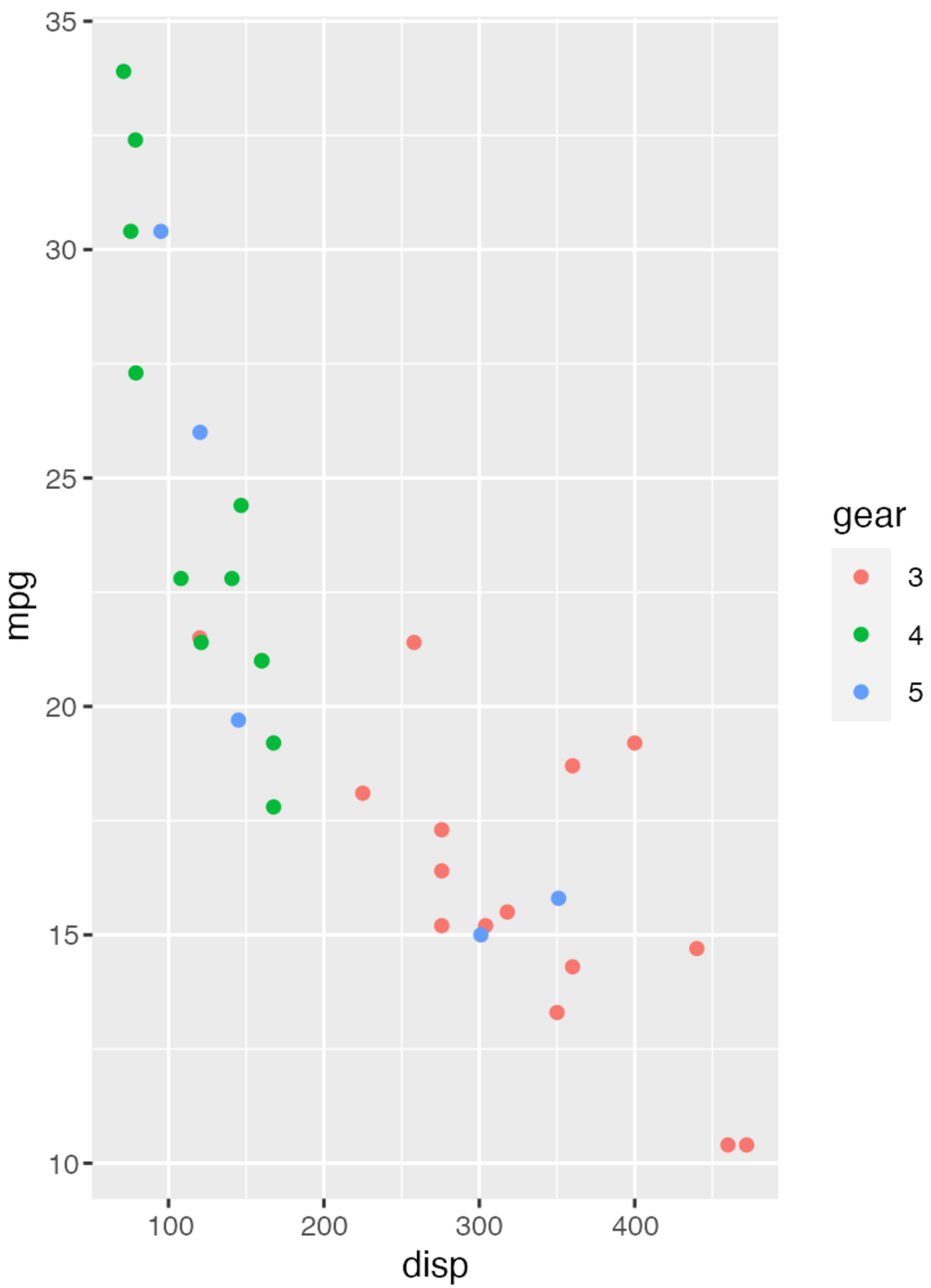


THE GRAMMAR

- None of the priors talked about the visual look of the plot.
- Theming spans every part of the graphic that is not linked to data

BUT I JUST WANTED TO MAKE A PIE CHART . . .

You

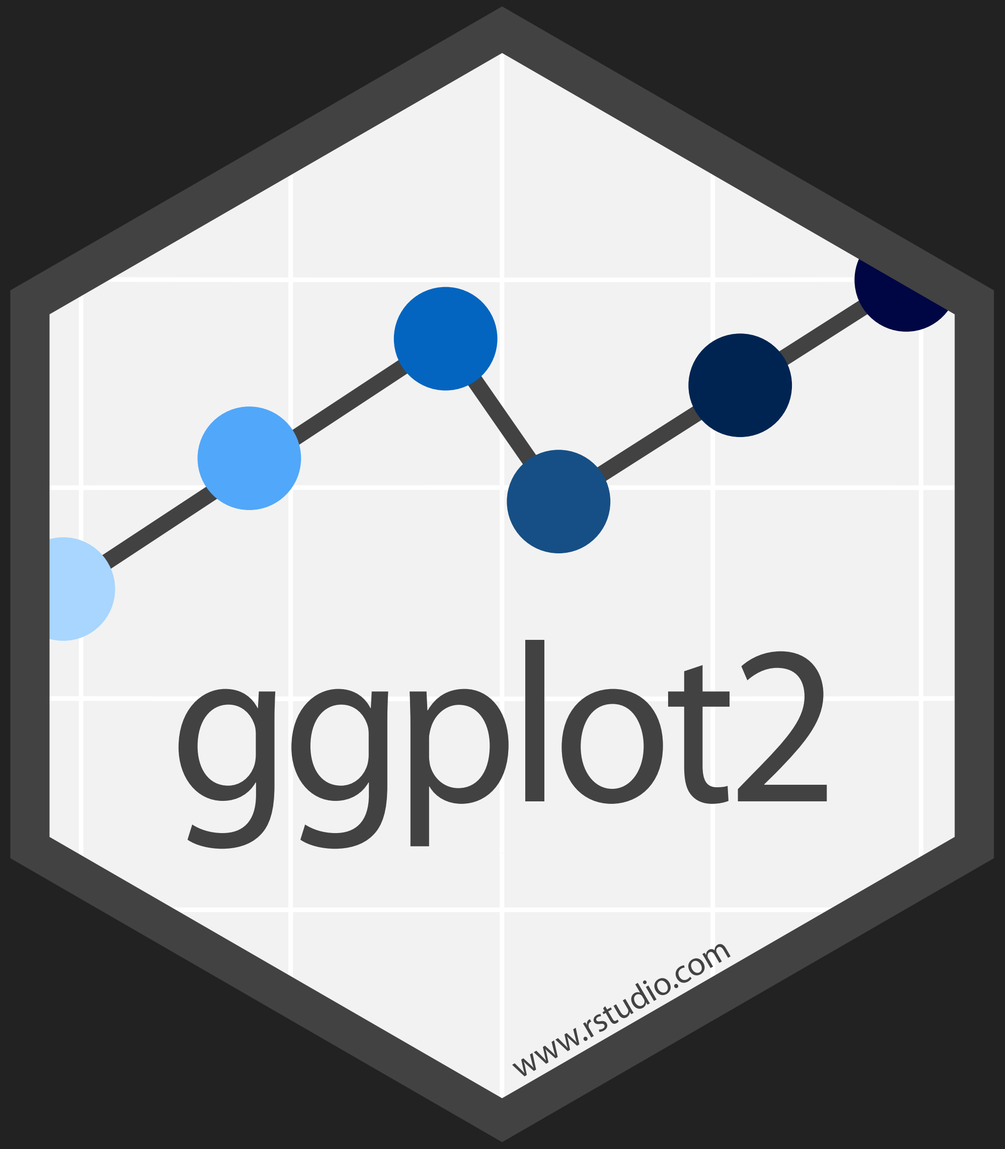


PART 2:

THE GGPLOT2 API

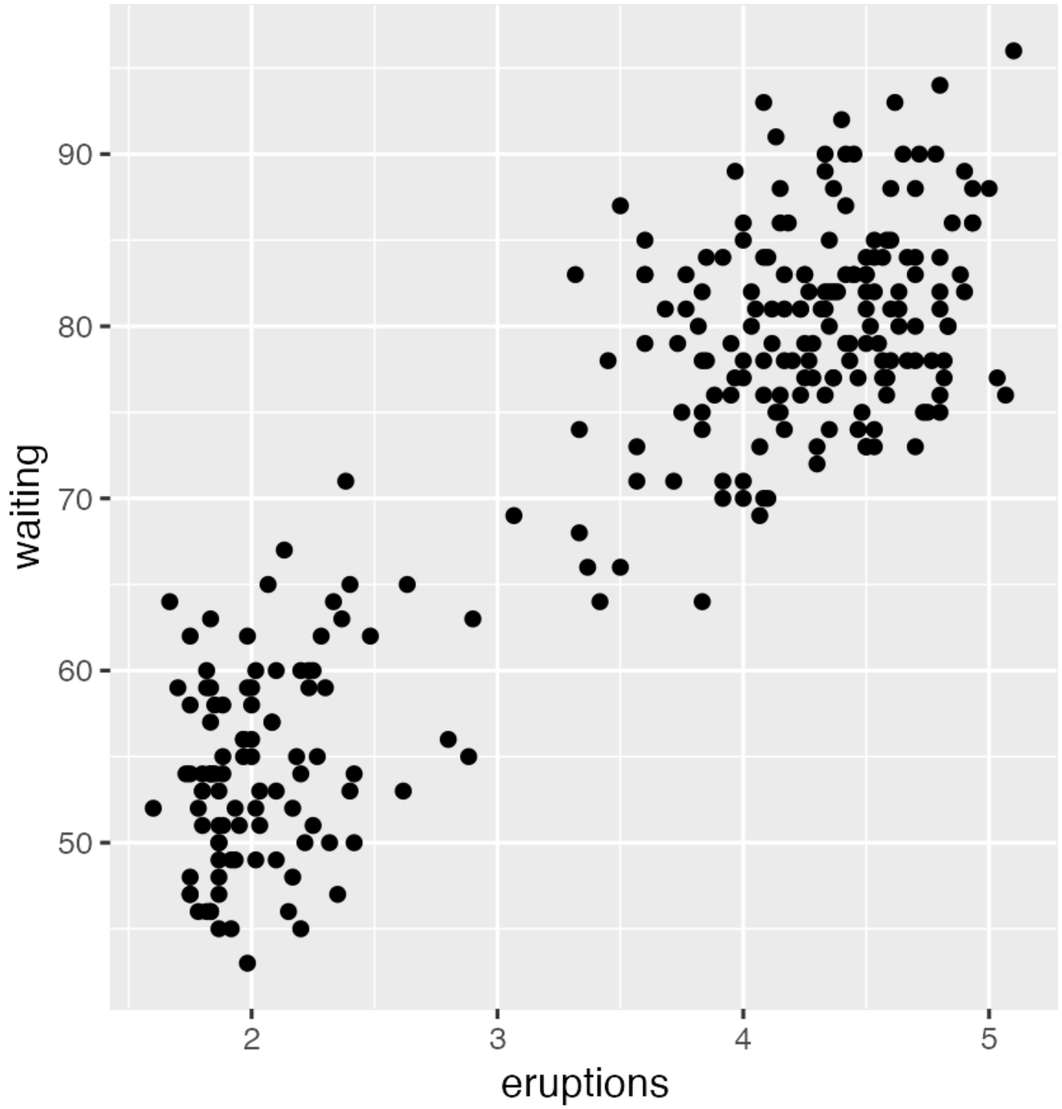
GGPLOT2

- Grammar of Graphics is a design system
- ggplot2 is an implementation
- Hadleys grammar != Lelands grammar
  - Sometimes real life gets in the way
- The spirit lives on
- Other implementations:
  - Tableau
  - Vega-Lite



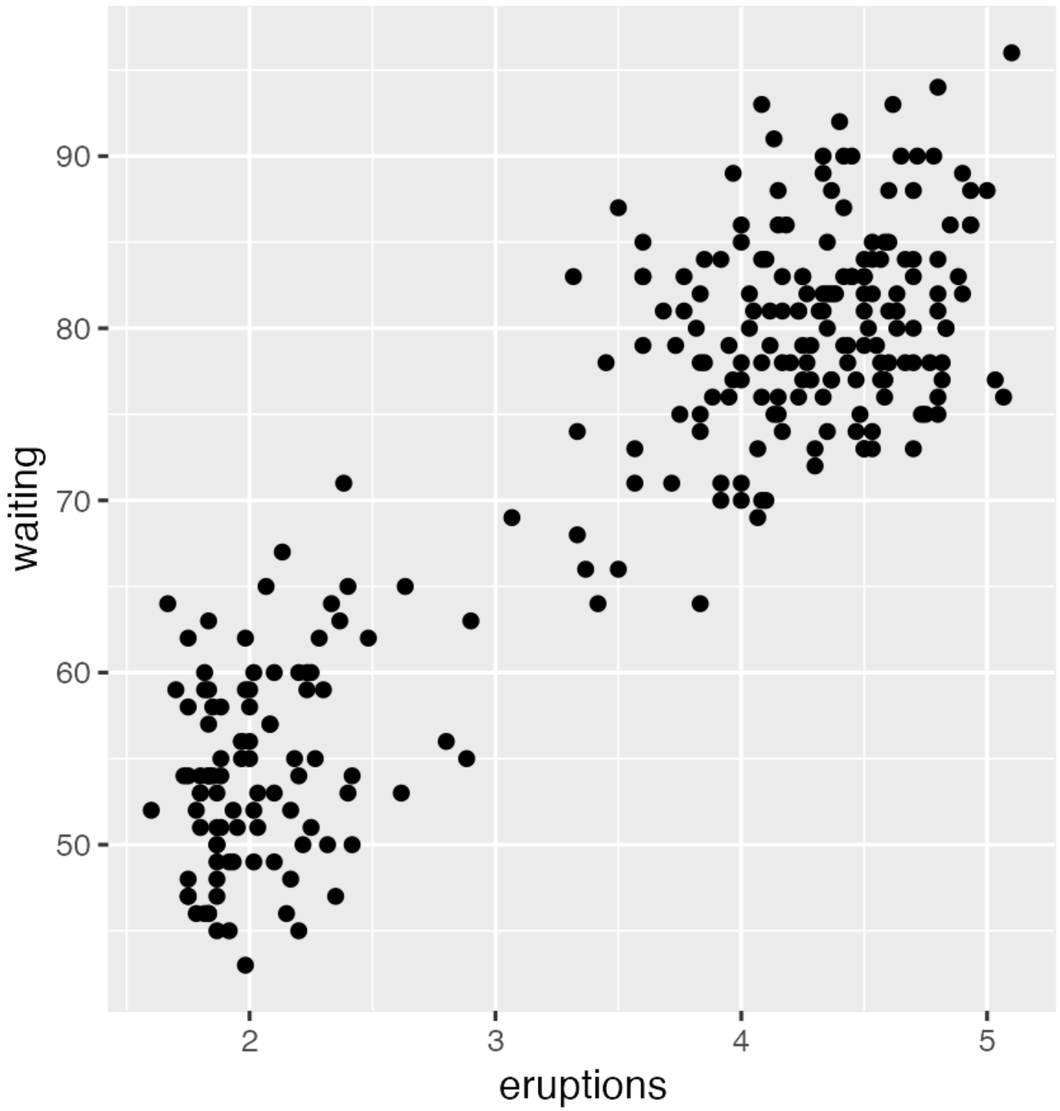
THE GGPLOT2 API

```
ggplot(data = faithful,
       mapping = aes(x = eruptions,
                     y = waiting)) +
  geom_point()
```



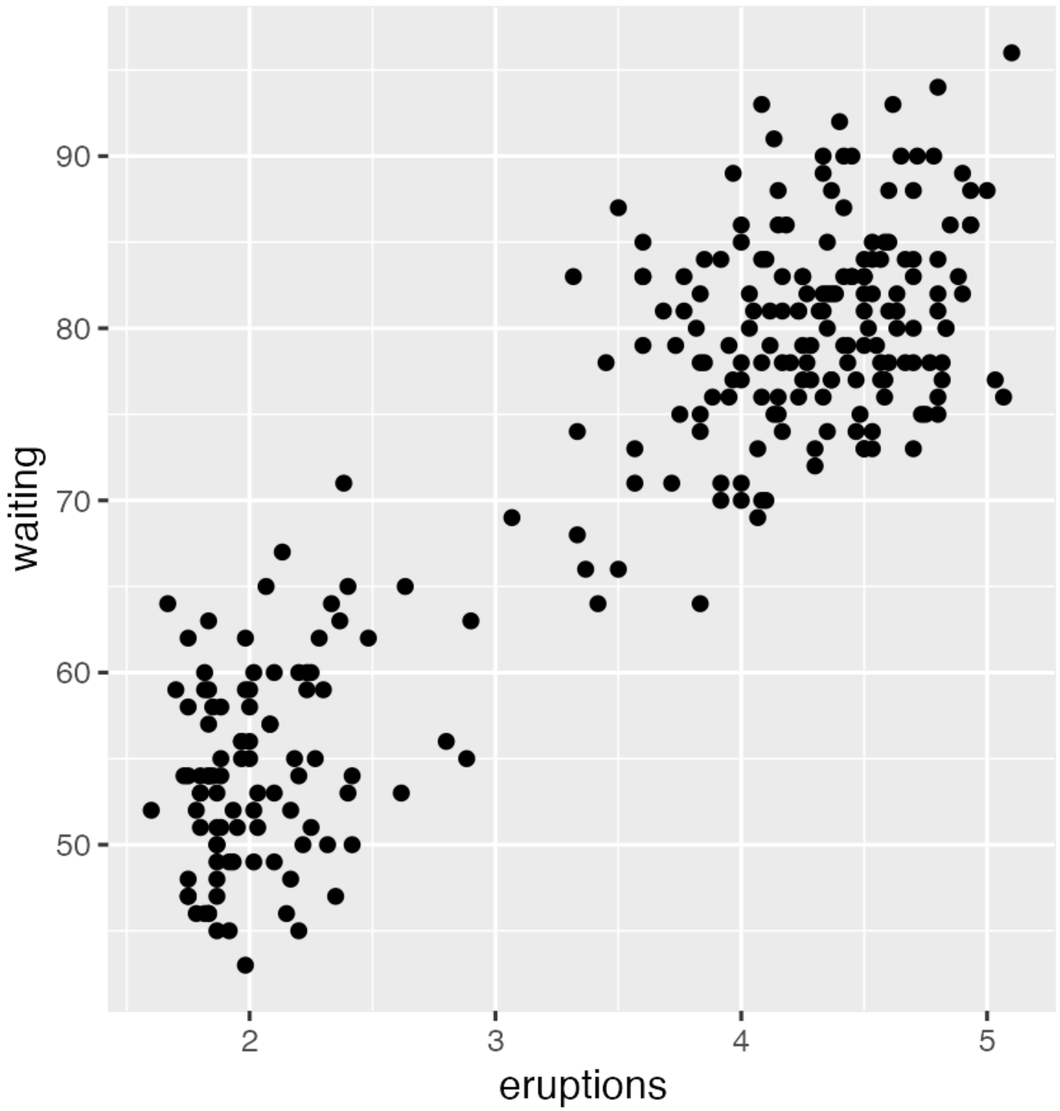
```
ggplot(data = faithful,
       mapping = aes(x = eruptions,
                     y = waiting)) +
  geom_point()
```

Which dataset to plot



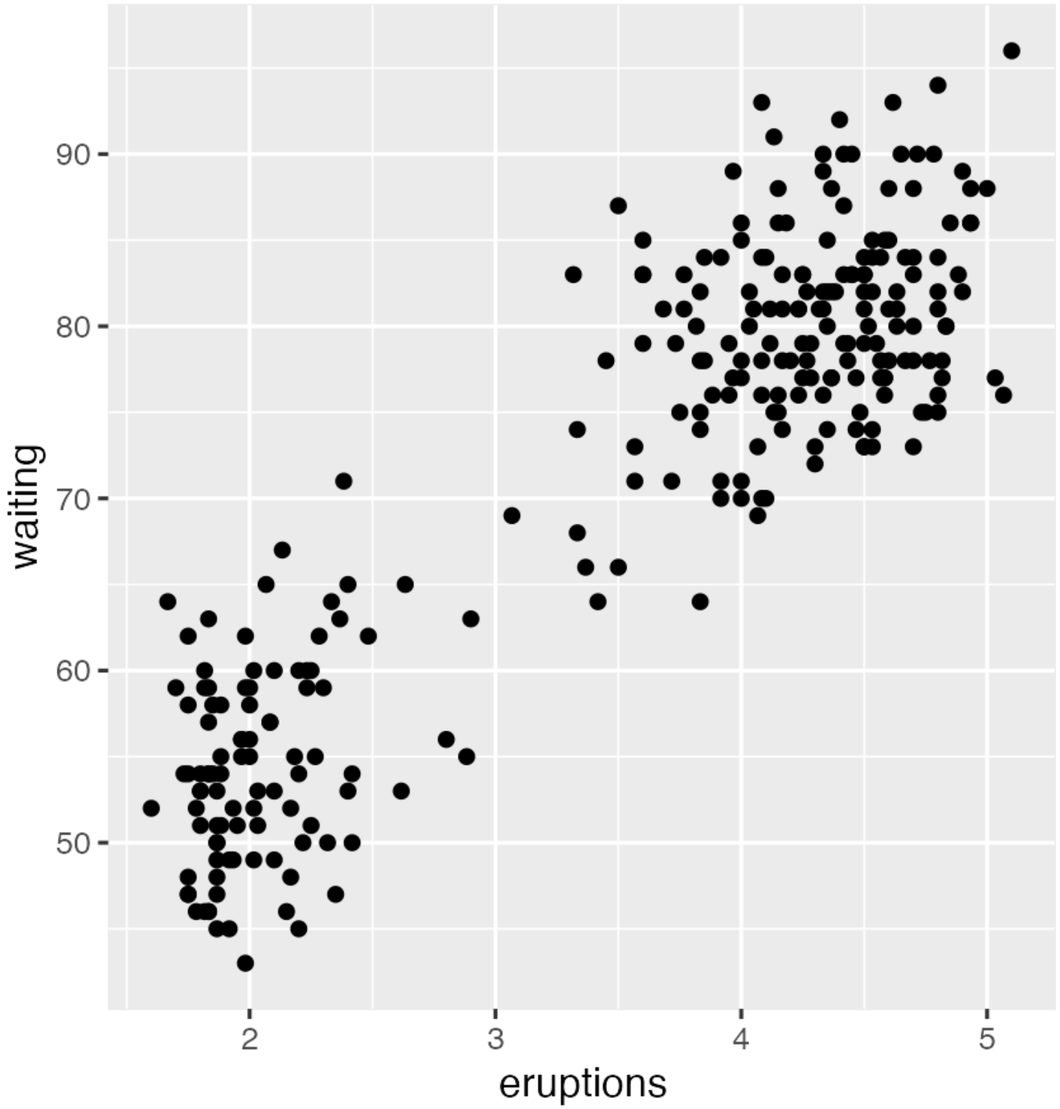
```
ggplot(data = faithful,
       mapping = aes(x = eruptions,
                     y = waiting)) +
  geom_point()
```

Which columns to use for x and y



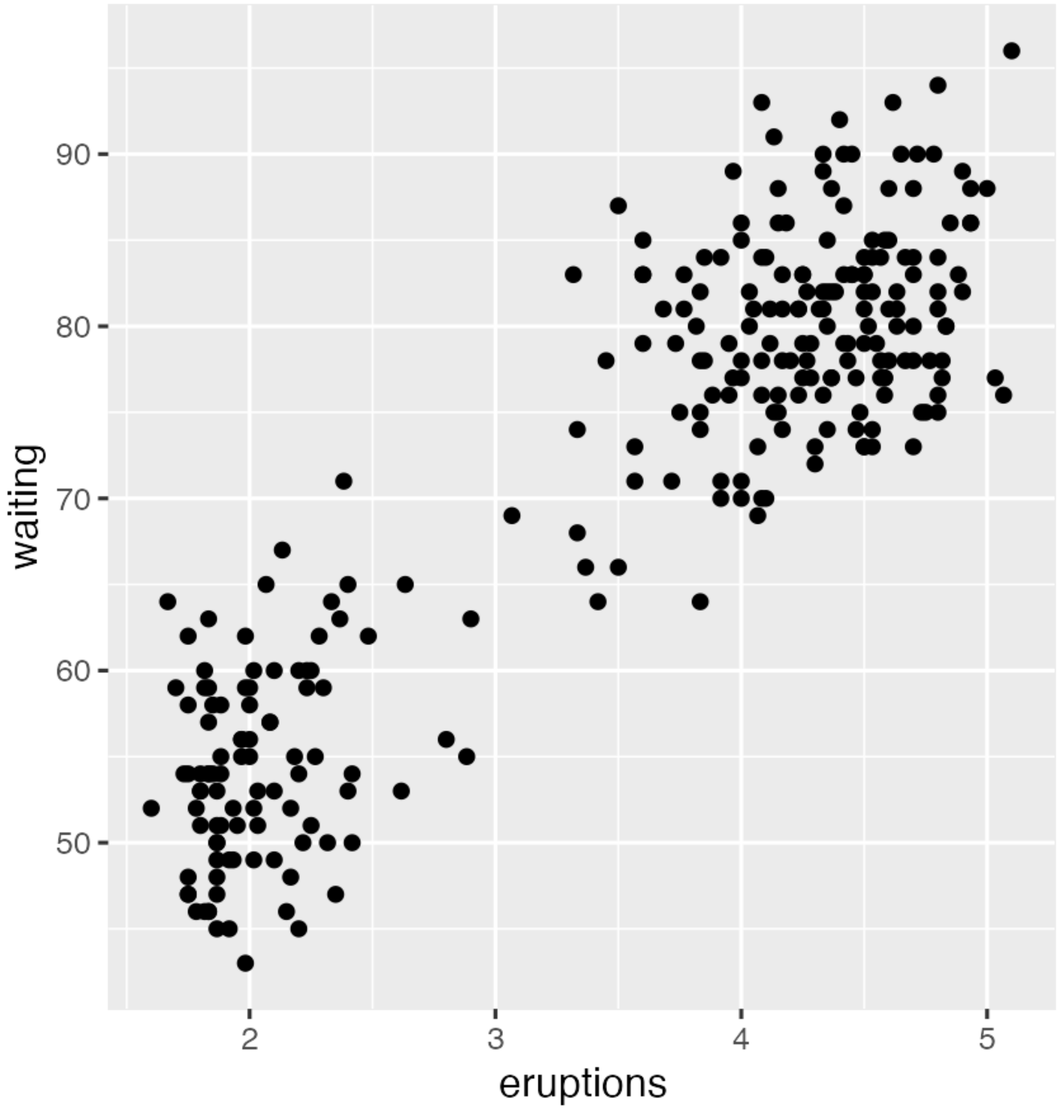
```
ggplot(data = faithful,
       mapping = aes(x = eruptions,
                     y = waiting)) +
  geom_point()
```

How to draw the plot

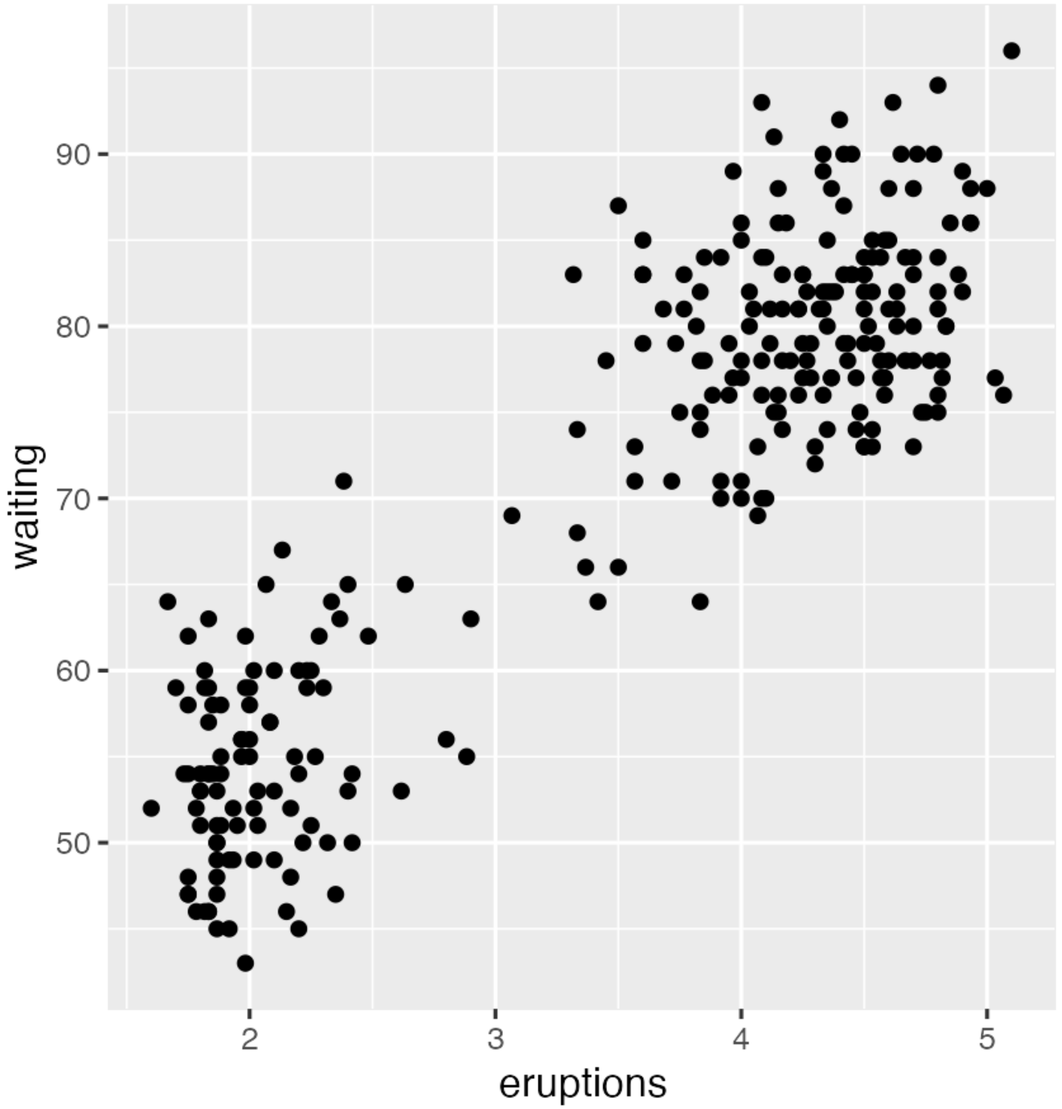


```
ggplot(data = faithful,
       mapping = aes(x = eruptions,
                     y = waiting)) +
  geom_point()
```

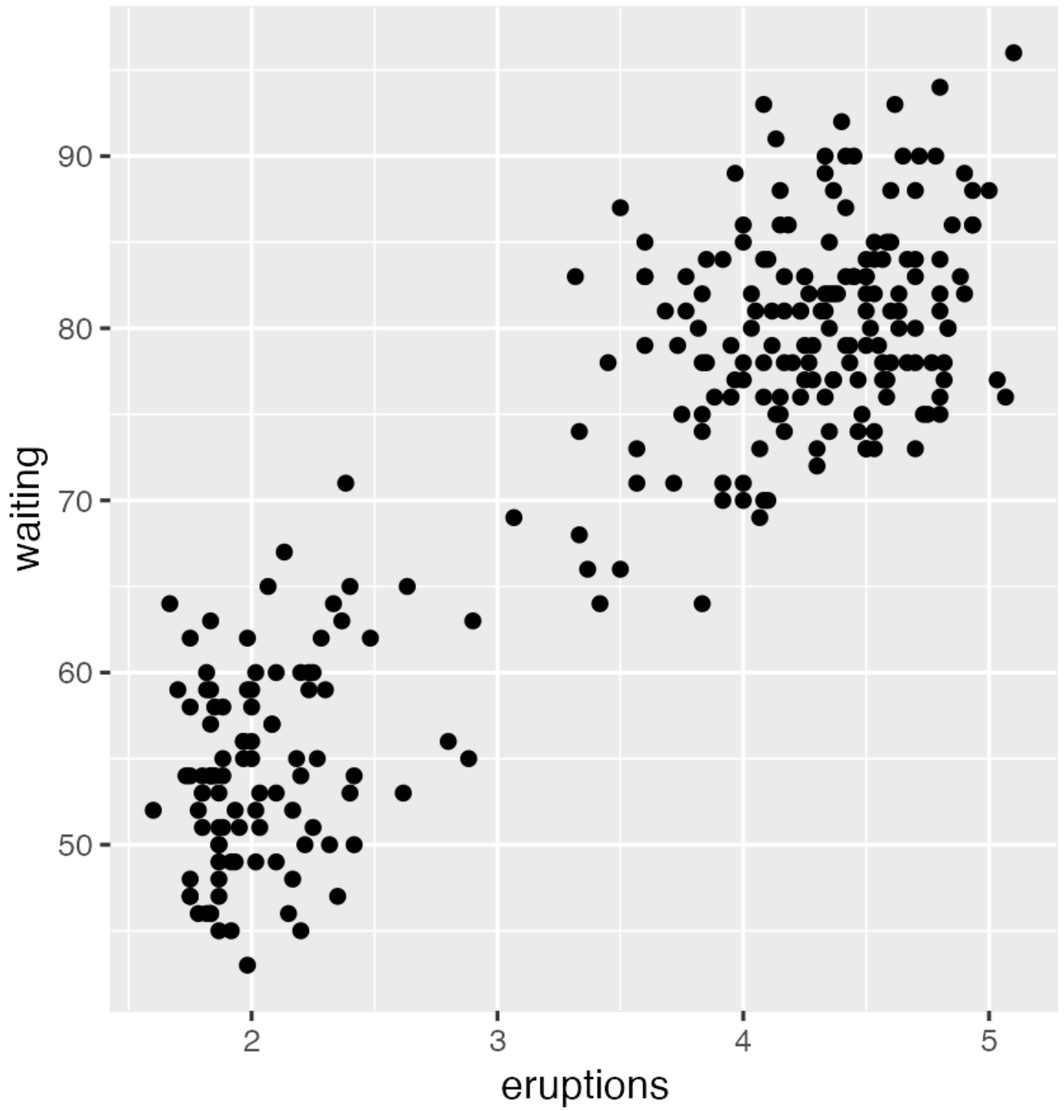
‘+’ is used to combine ggplot2 elements



```
ggplot(data = faithful) +
  geom_point(mapping = aes(x = eruptions,
                           y = waiting))
```

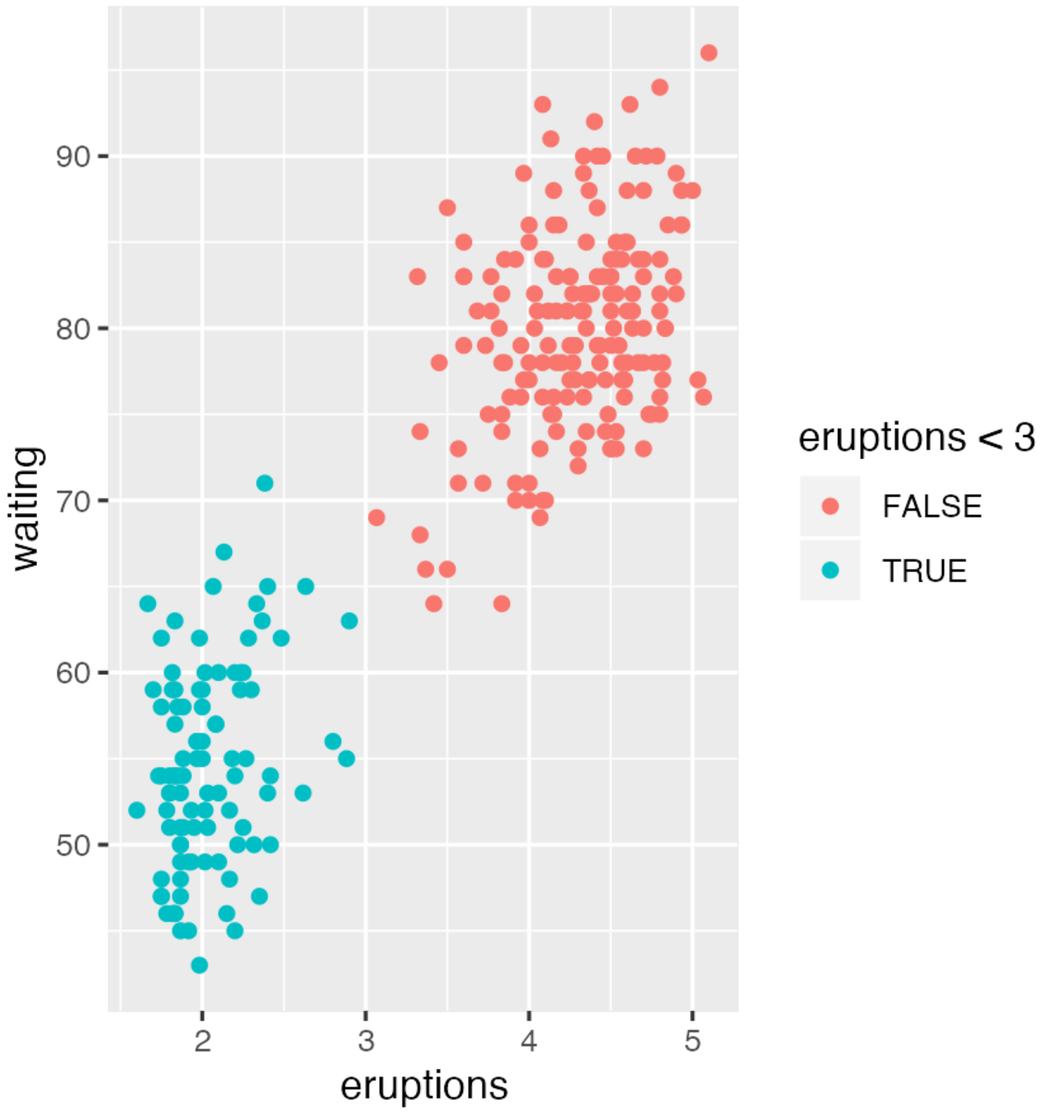


```
ggplot() +
  geom_point(mapping = aes(x = eruptions,
                           y = waiting),
             data = faithful)
```



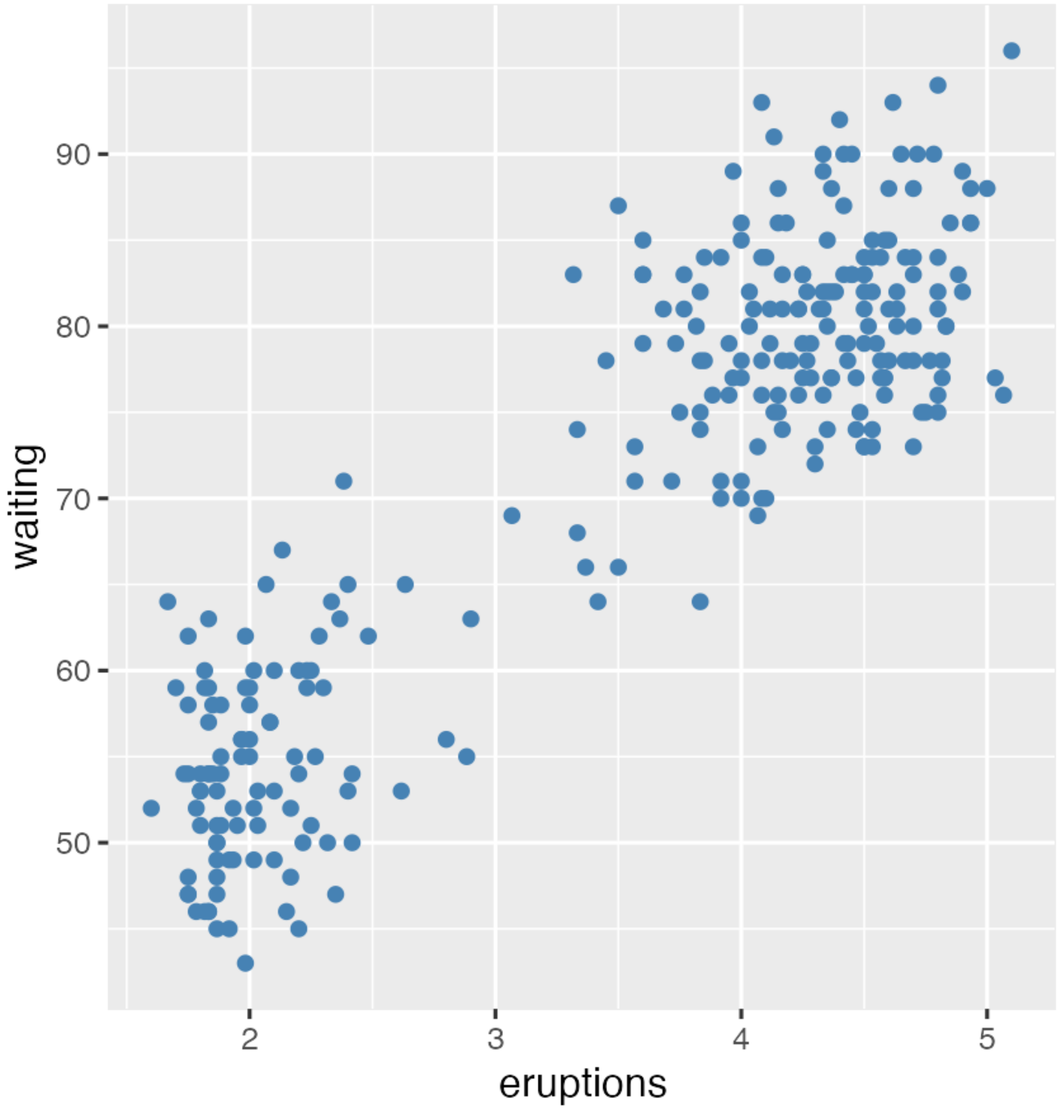
```
ggplot(faithful) +
  geom_point(aes(x = eruptions,
                 y = waiting,
                 colour = eruptions < 3))
```

Mapping colour



```
ggplot(faithful) +
  geom_point(aes(x = eruptions,
                 y = waiting),
             colour = 'steelblue')
```

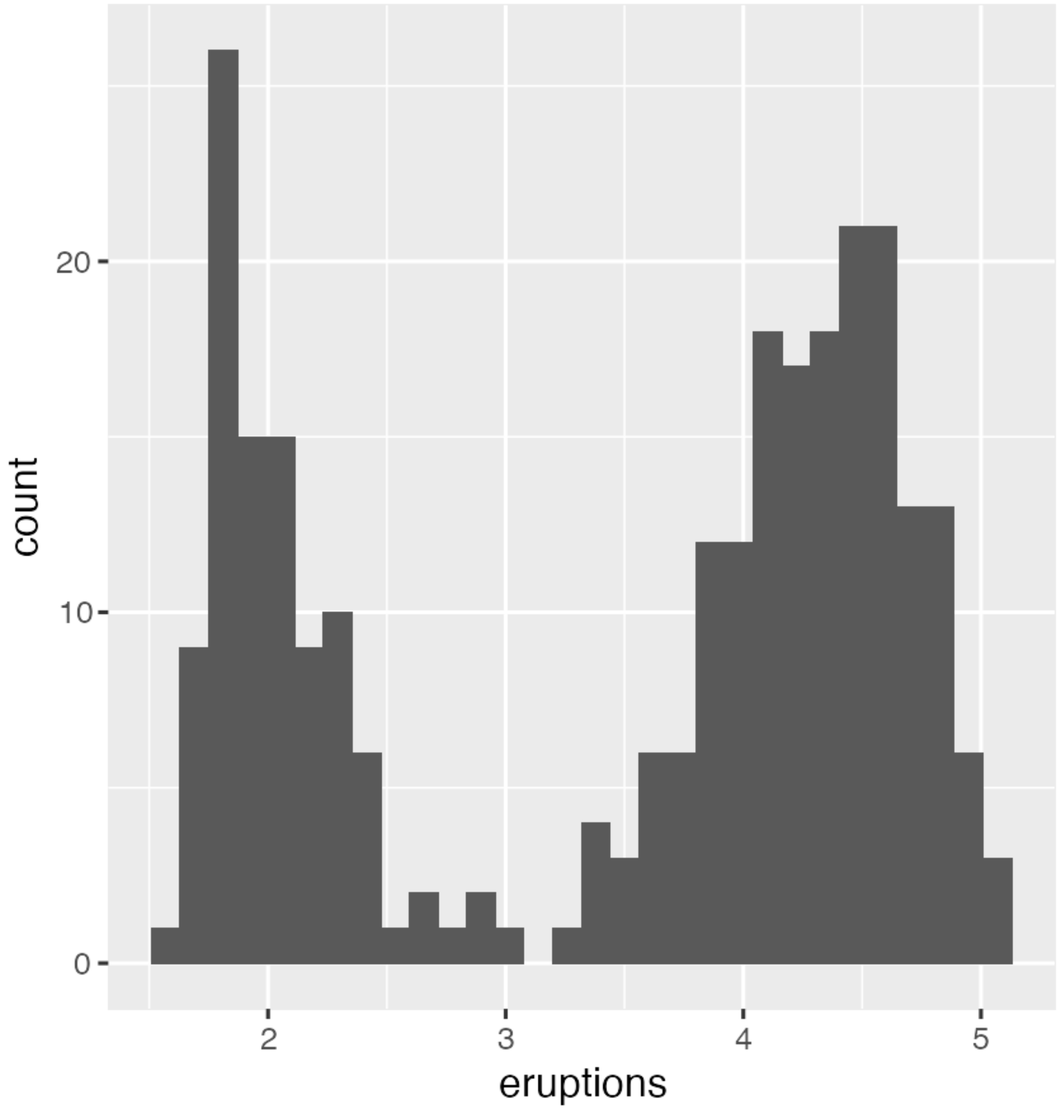
Setting colour



```
ggplot(faithful) +
  geom_histogram(aes(x = eruptions))
```

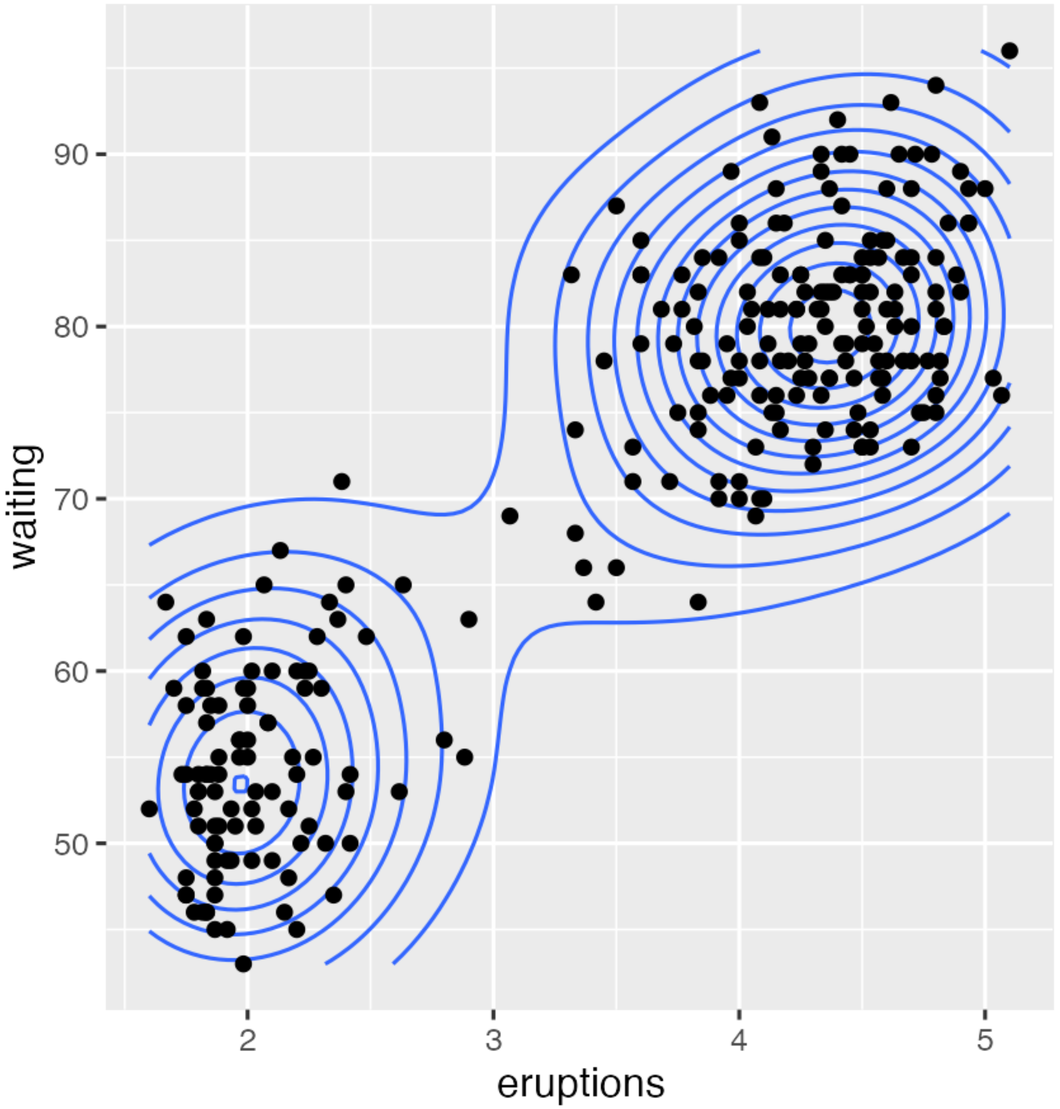
There are many types of geoms

Their mapping requirements differ



```
ggplot(faithful,
       aes(x = eruptions, y = waiting)) +
  geom_density_2d() +
  geom_point()
```

Layers are stacked in the order of code appearance



WHAT DID WE NEED?

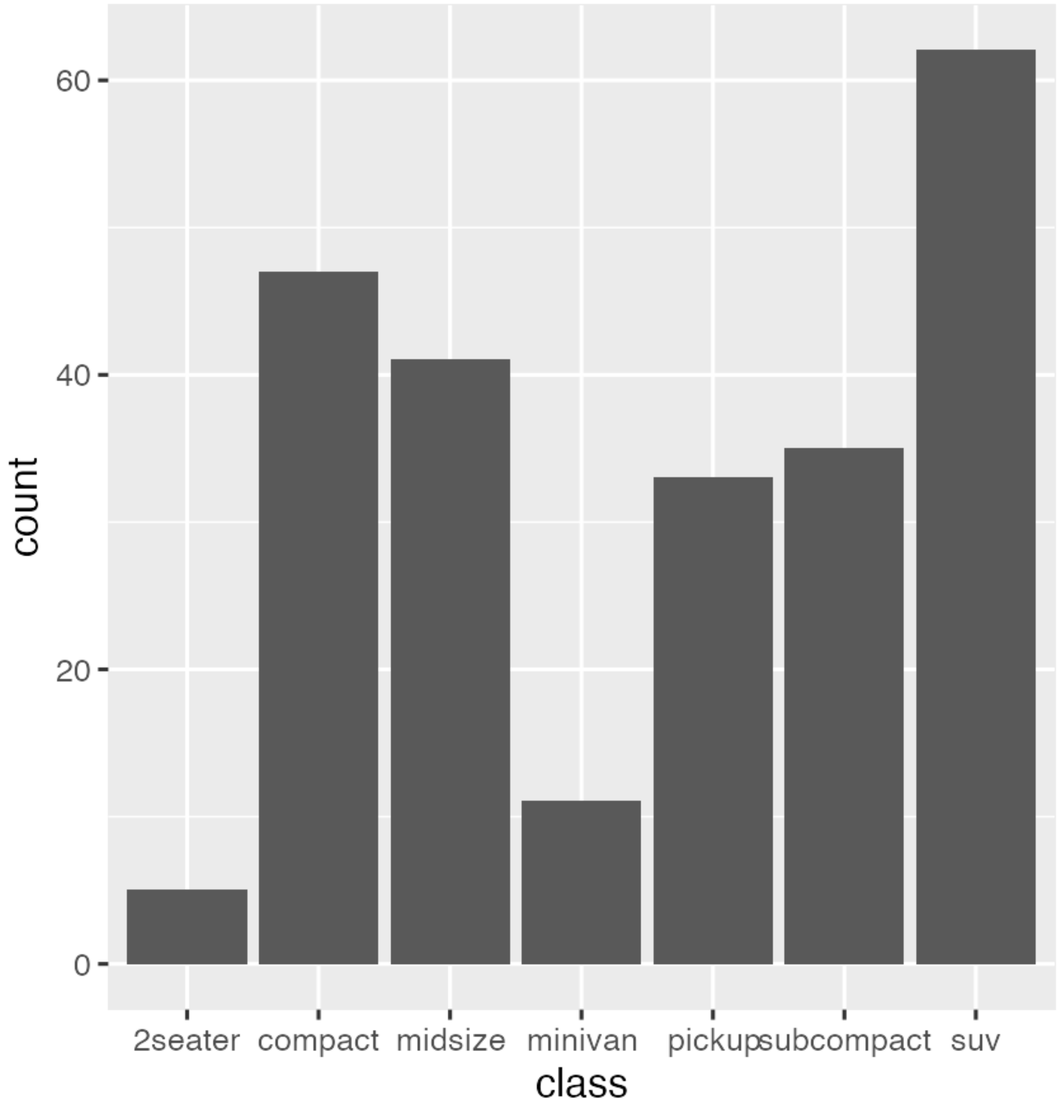
EVERYTHING ELSE HAS SENSIBLE DEFAULTS

STATISTICS

- Linked to geometries
- Every geom has a default stat(istic)
- A layer can be created with a call to stat_*() or geom_*(), but community has coalesced around geom_*()

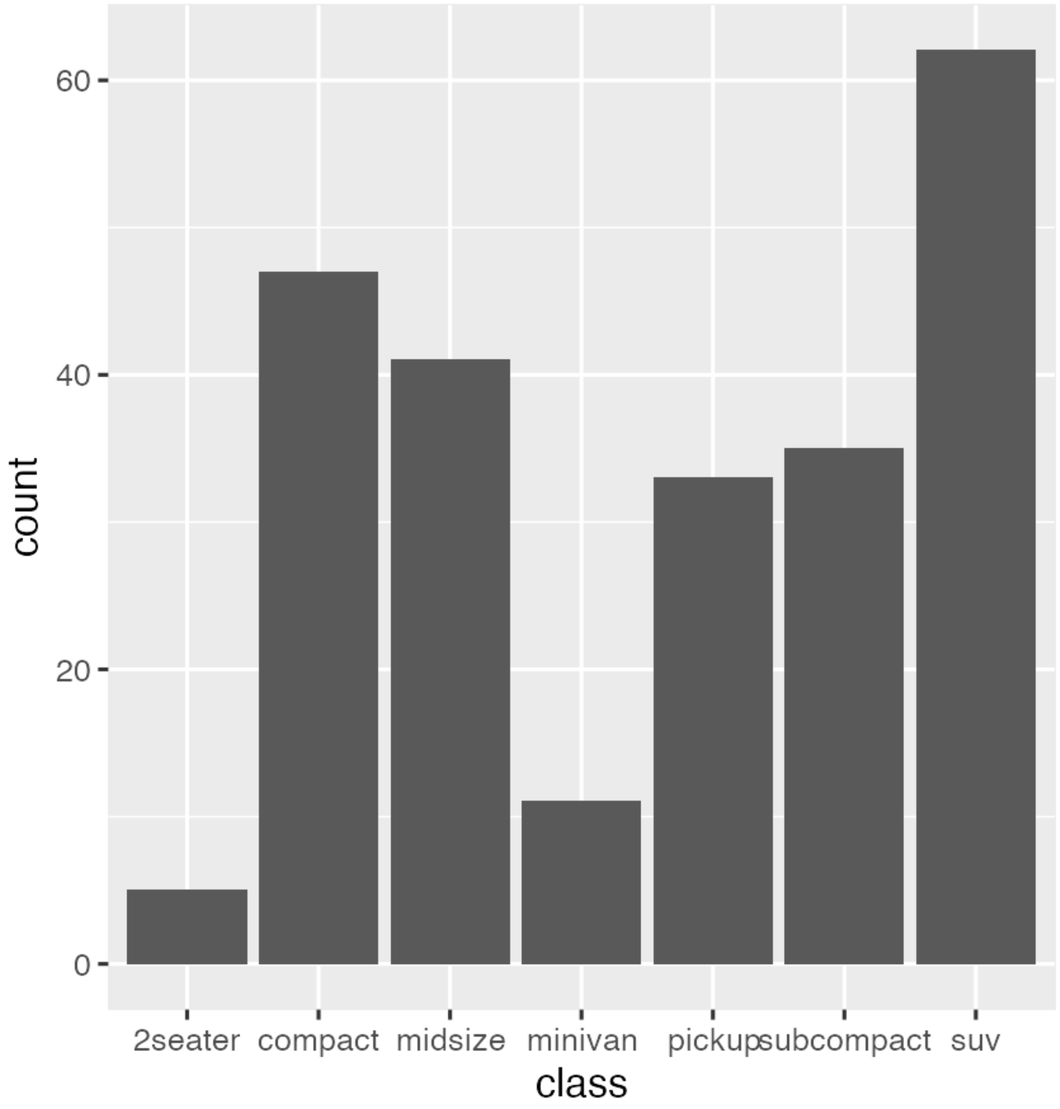
```
ggplot(mpg) +
  geom_bar(aes(x = class))
```

geom_bar() uses stat_count() by default



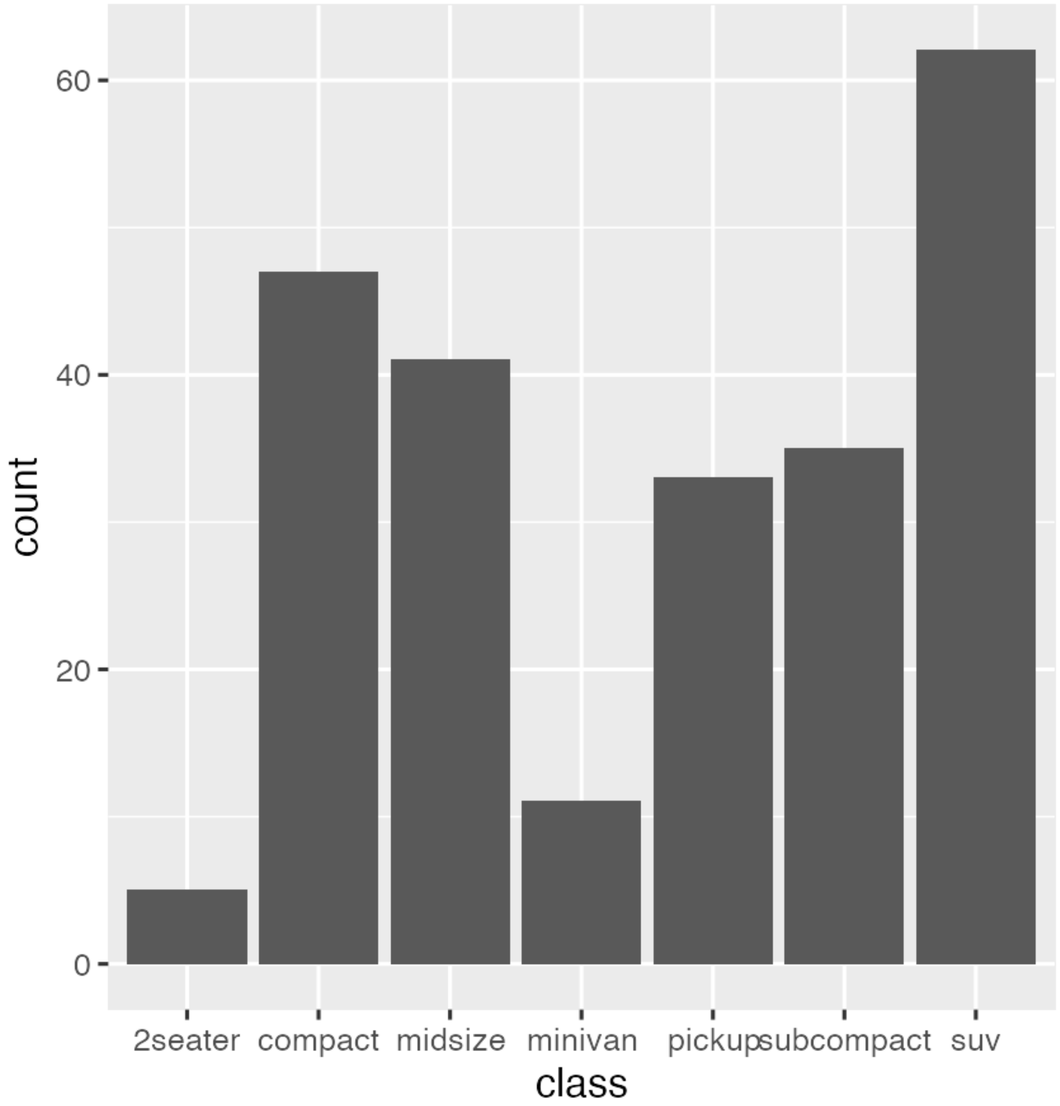
```
mpg_counted ← mpg %>%
  count(class, name = 'count')

ggplot(mpg_counted) +
  geom_bar(aes(x = class, y = count),
           stat = 'identity')
```

If you have precomputed data
use identity stat



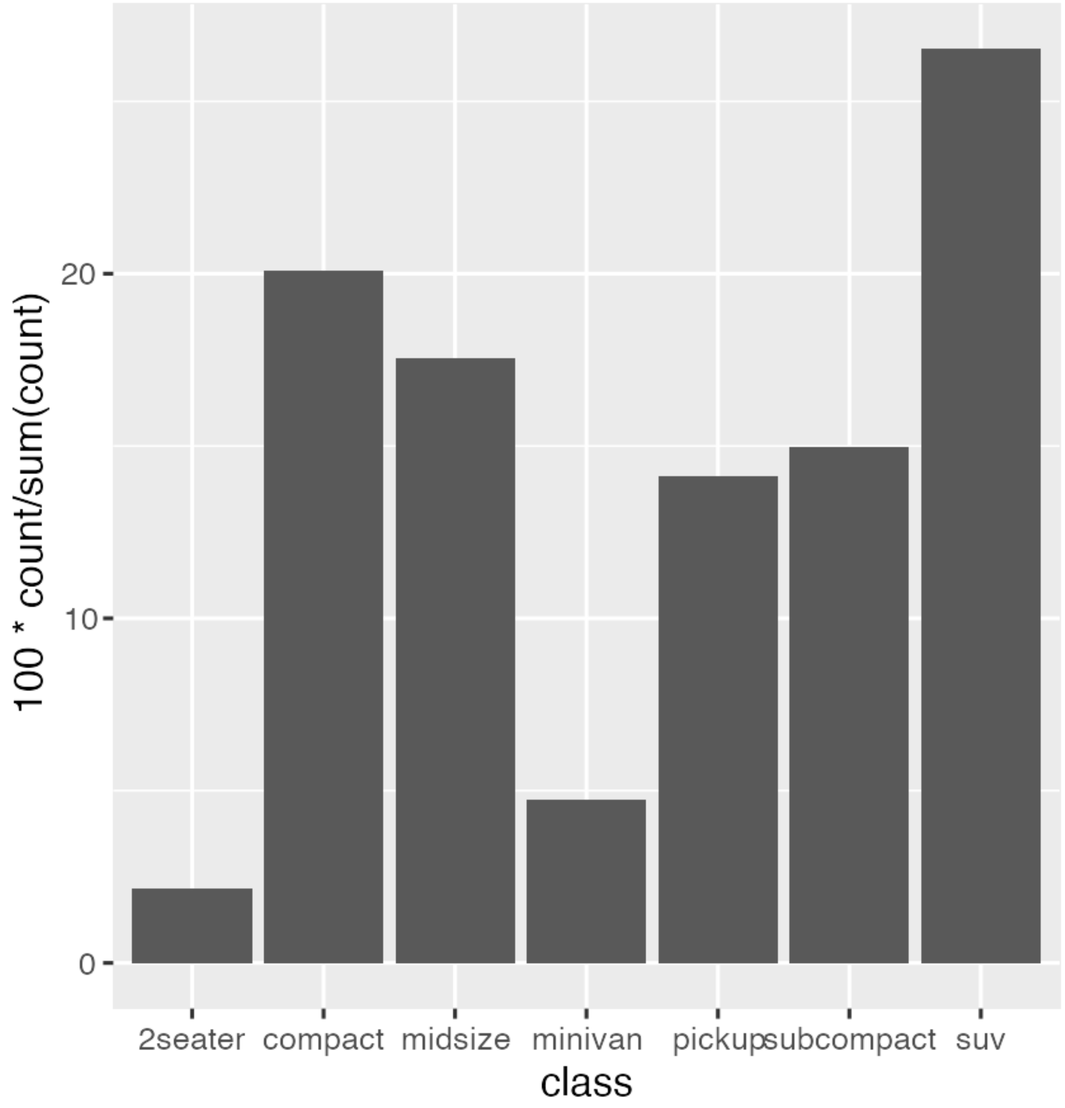
```
mpg_counted ← mpg %>%
  count(class, name = 'count')

ggplot(mpg_counted) +
  geom_col(aes(x = class, y = count))
```

… or use the geom_col() shortcut



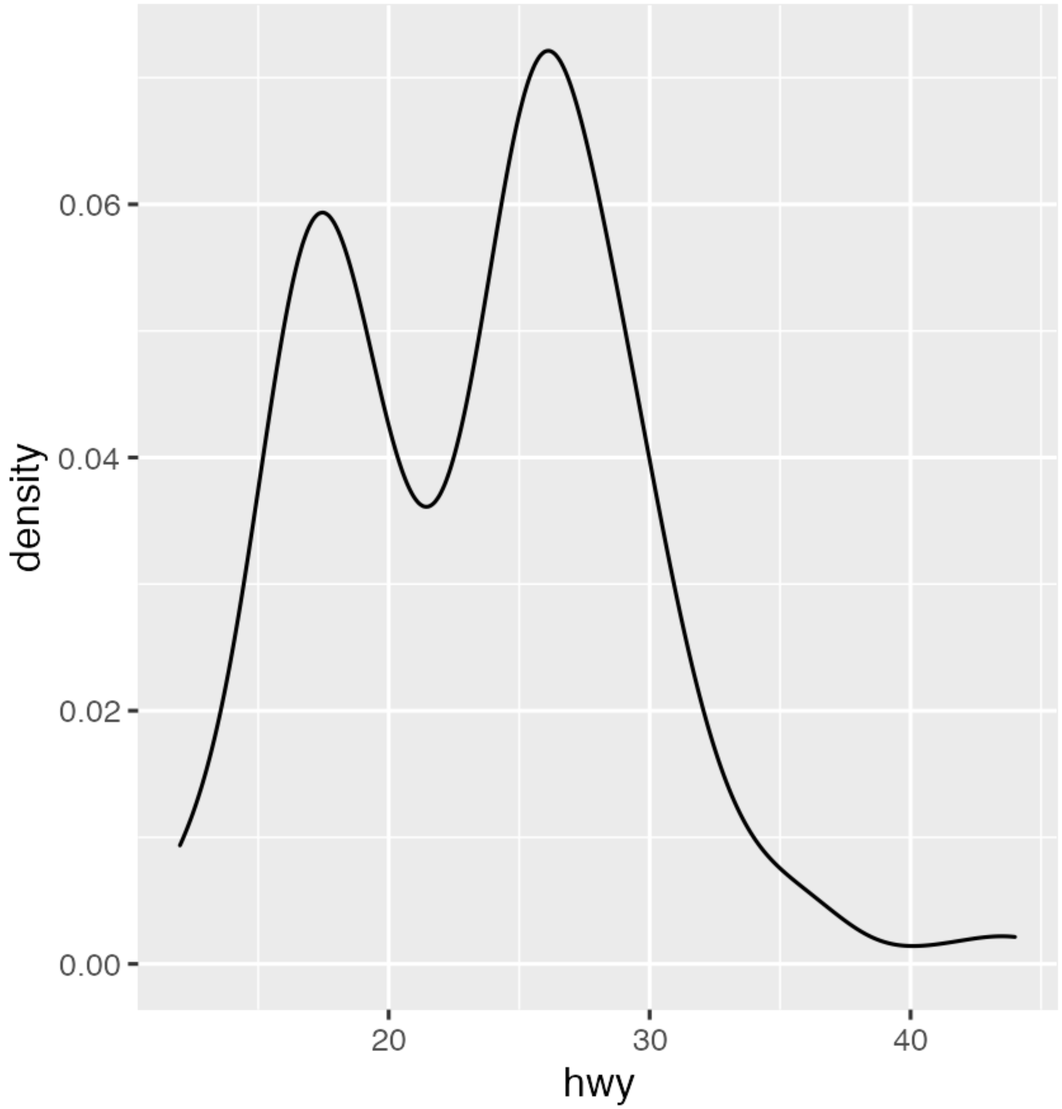
```
ggplot(mpg) +
  geom_bar(
    aes(
      x = class,
      y = after_stat(100 * count / sum(count))
    )
  )
```

Use after_stat() to modify mapping from stats (here we calculate %)



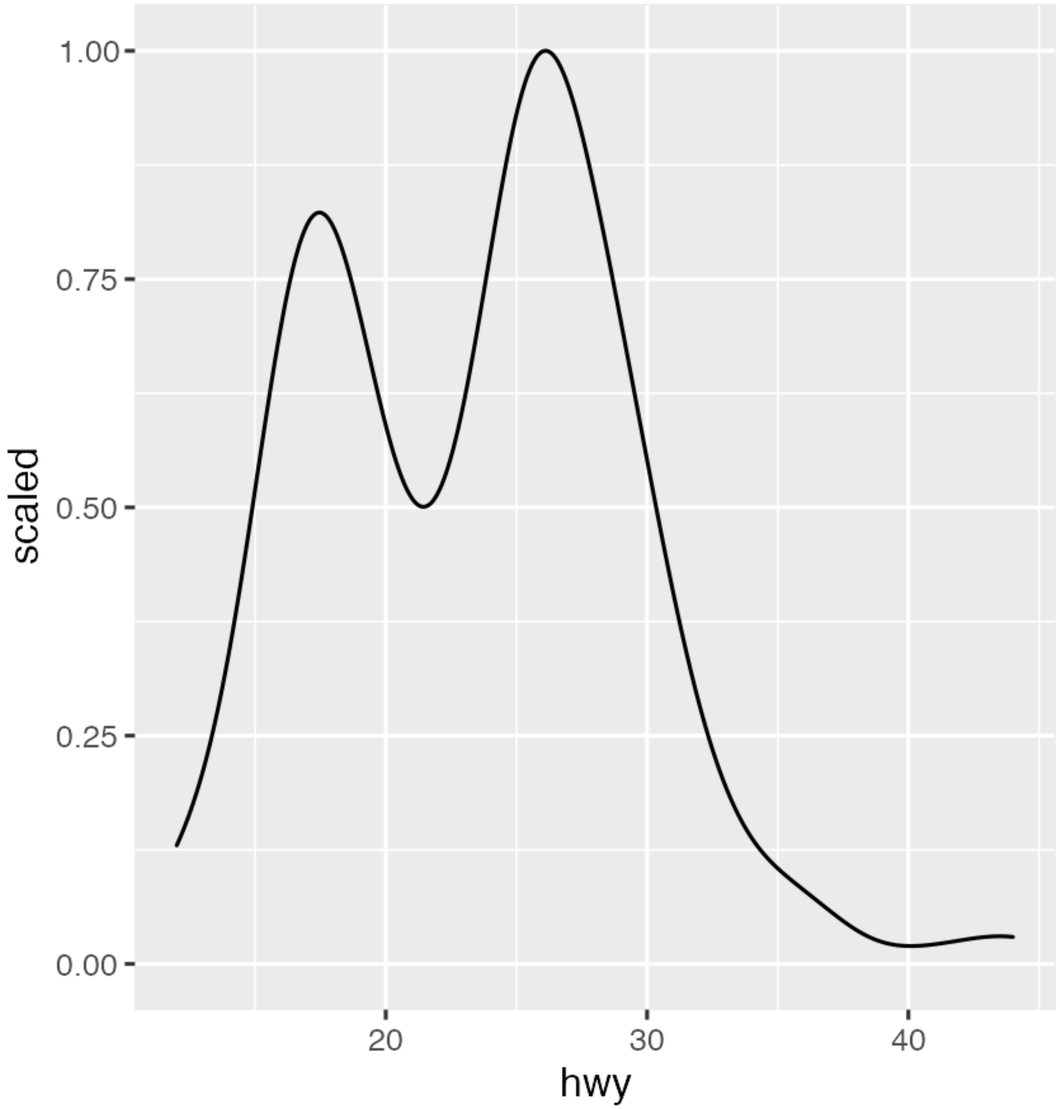
```
ggplot(mpg) +
  geom_density(aes(x = hwy))
```

Many stats provide multiple calculated values and use one by default (here ‘density’)



```
ggplot(mpg) +
  geom_density(aes(x = hwy,
                   y = after_stat(scaled)))
```

As before, these can be accessed with the after_stat() function

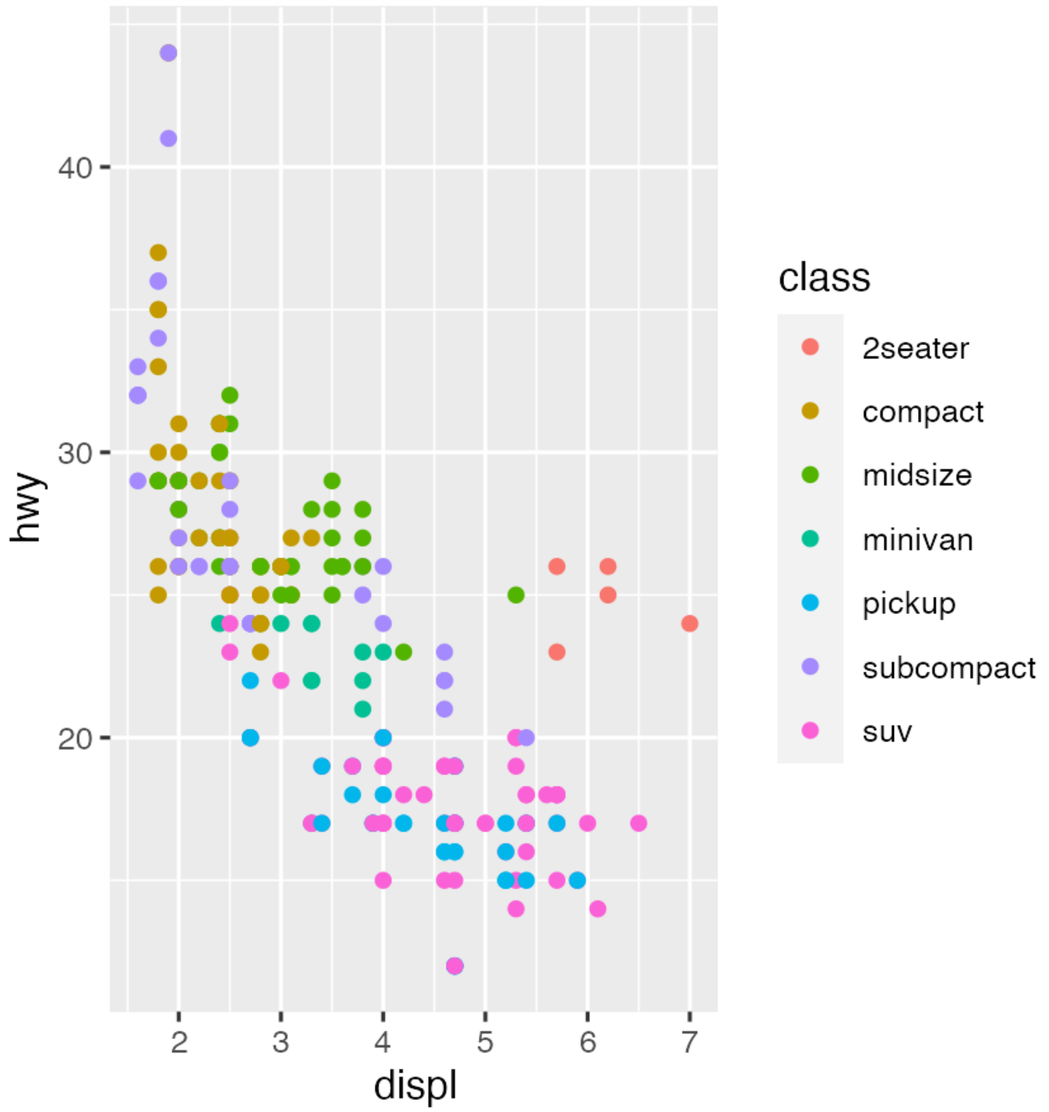


SCALES

- Everything inside aes() will have a scale
  - If none is provided it will get a default
- Scales follow a predictable naming scheme: scale_<aesthetic>_<type>()
- <type> can either be a generic (continuous, discrete, or binned) or specific (e.g. area, for scaling size to circle area)

```
ggplot(mpg) +
  geom_point(
    aes(x = displ, y = hwy, colour = class)
  )
```

based on the vector type of class, a discrete colour scale is picked



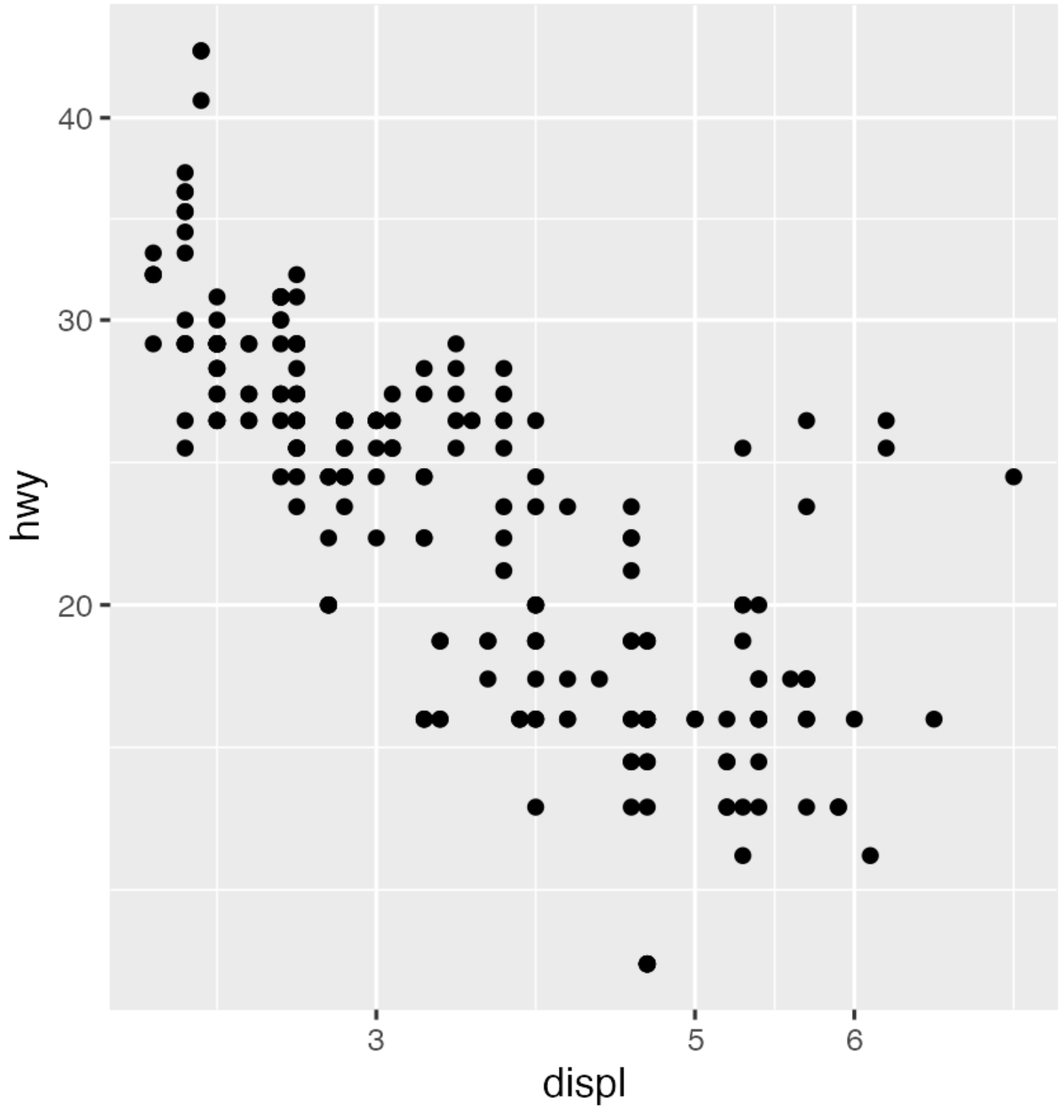
```
ggplot(mpg) +
  geom_point(
    aes(x = displ, y = hwy, colour = class)
  ) +
  scale_colour_brewer(type = 'qual')
```

We take control by adding
our own explicitly

While function name is predictable,
arguments are not

```
ggplot(mpg) +
  geom_point(aes(x = displ, y = hwy)) +
  scale_x_continuous(breaks = c(3, 5, 6)) +
  scale_y_continuous(trans = 'log10')
```

x and y are also controlled with scales

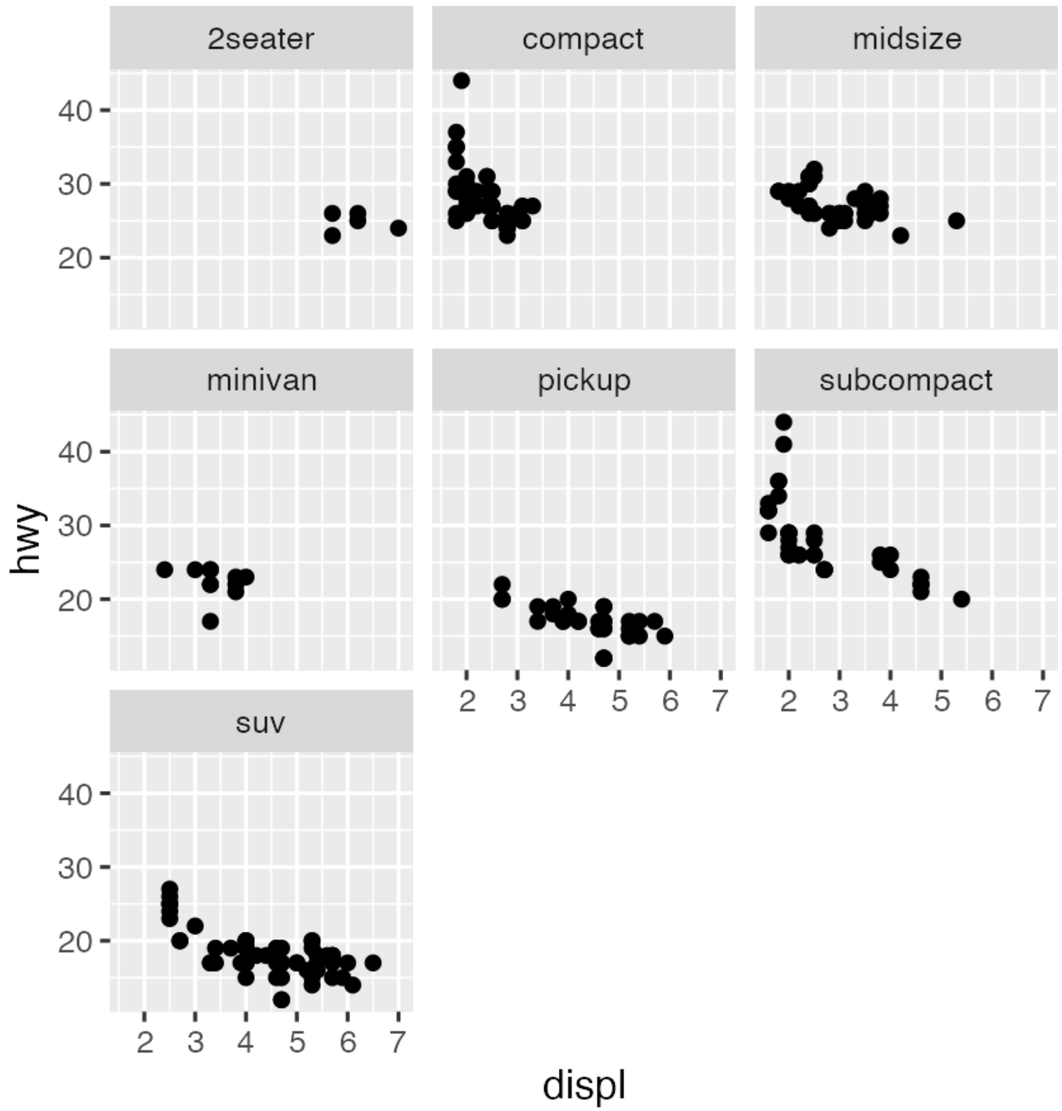


FACETS

- Split data into multiple panels
- Each panel is a representation of the same underlying logic
- Should not be used to combine multiple separate plots
- ggplot2 provide two facets for splitting data by categories

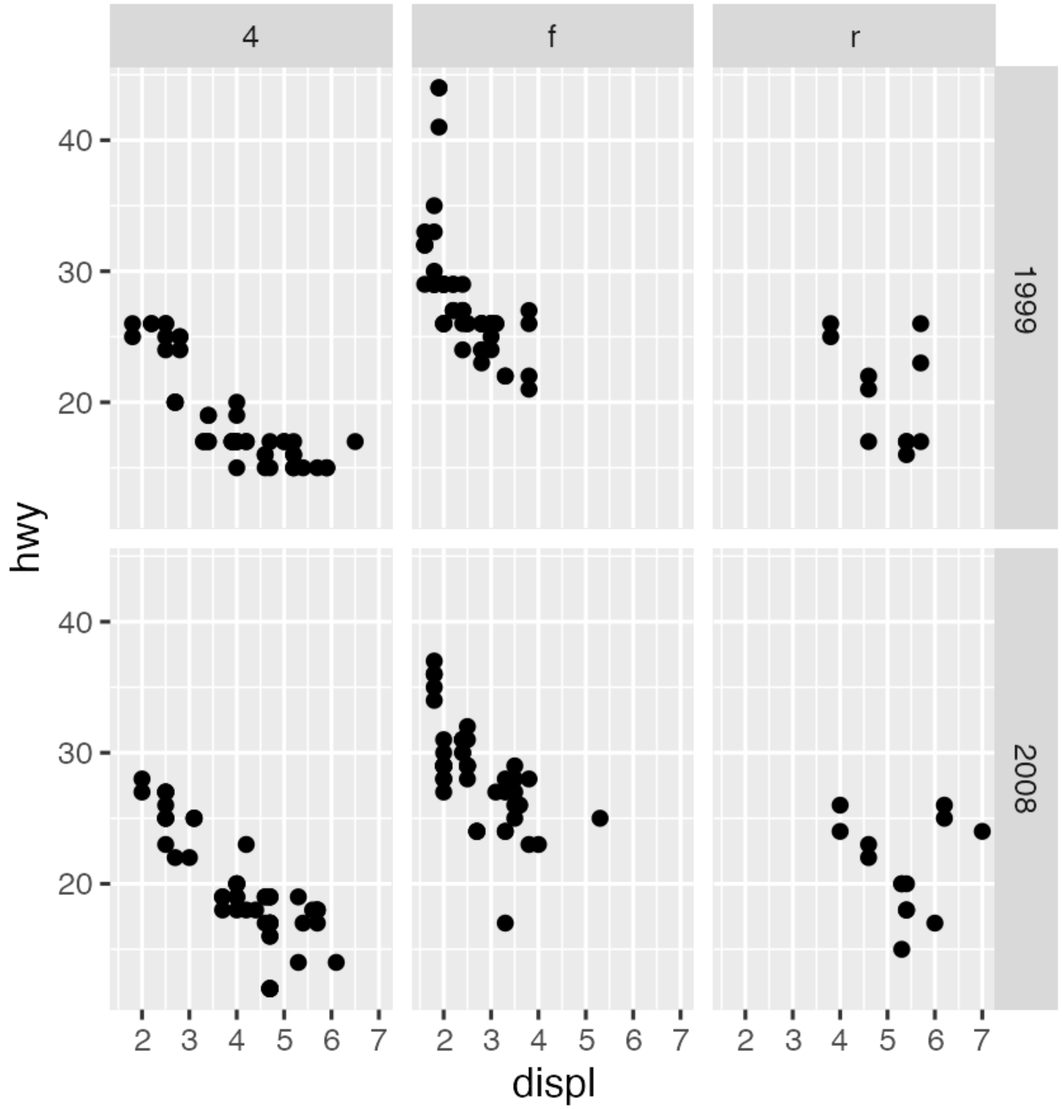
```
ggplot(mpg) +
  geom_point(aes(x = displ, y = hwy)) +
  facet_wrap(~ class)
```

Faceting is often the best way to avoid overplotting



```
ggplot(mpg) +
  geom_point(aes(x = displ, y = hwy)) +
  facet_grid(year ~ drv)
```

facet_grid() provides a way of doing graphic pivots

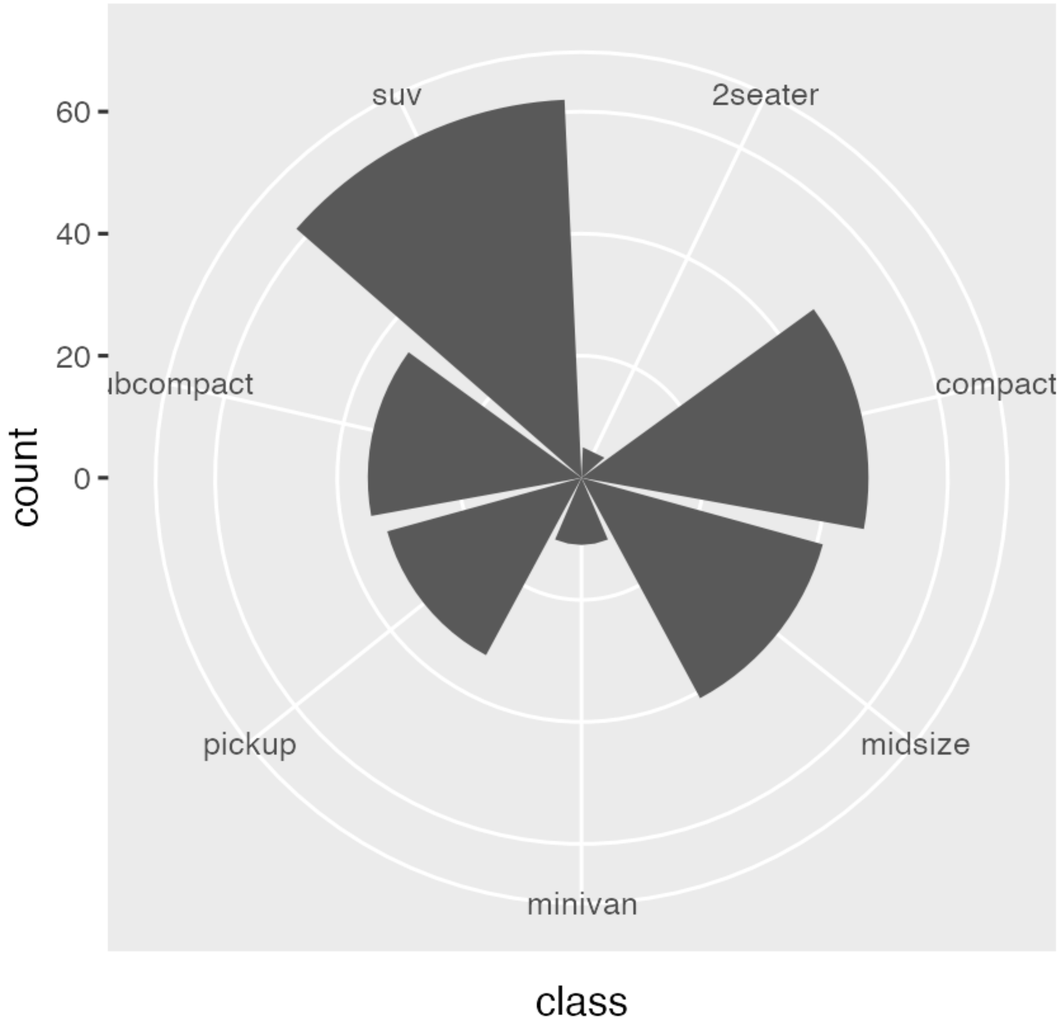


COORDINATES

- What kind of canvas should the final data be drawn on?
  - i.e. how should x and y be interpreted.
- Limits and transformation can be applied in scale or in coord
  - scale will apply it in the beginning
  - coord will apply it in the end
  - you usually want coord
- Extremely useful in cartography (map projections)

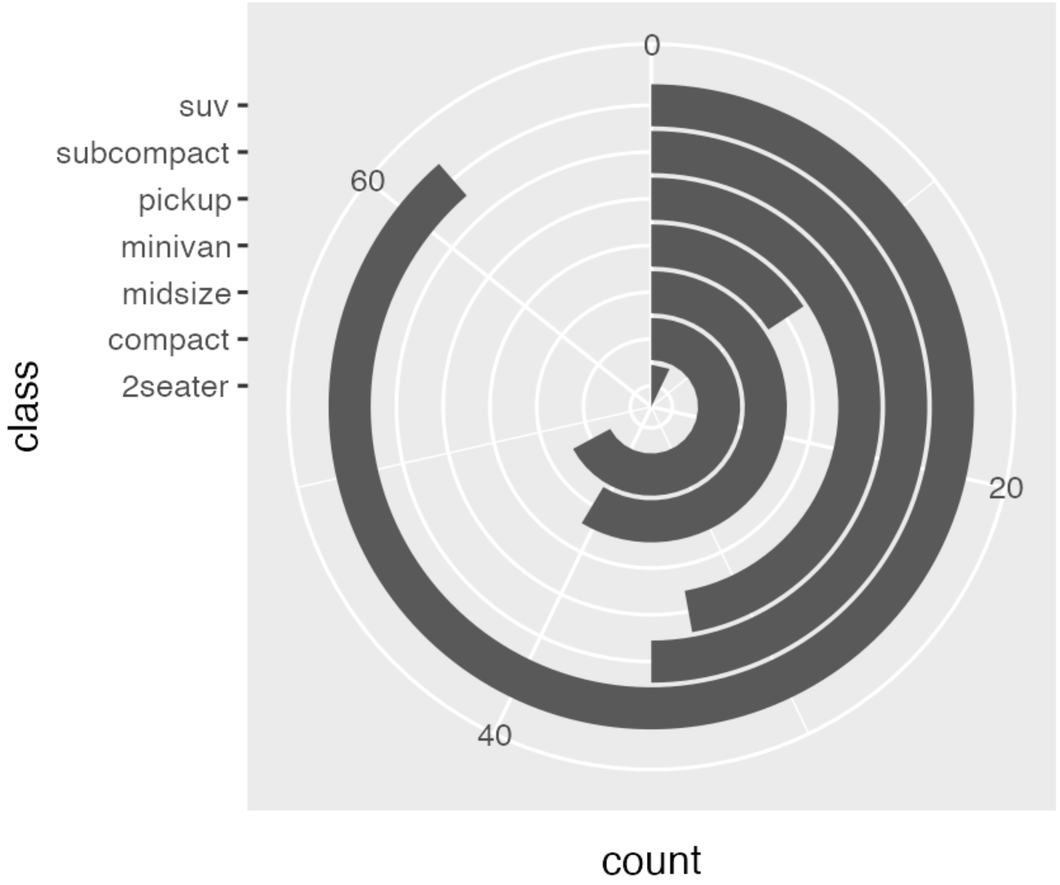
```
ggplot(mpg) +
  geom_bar(aes(x = class)) +
  coord_polar()
```

A polar coordinate system interprets x and y as radius and angle



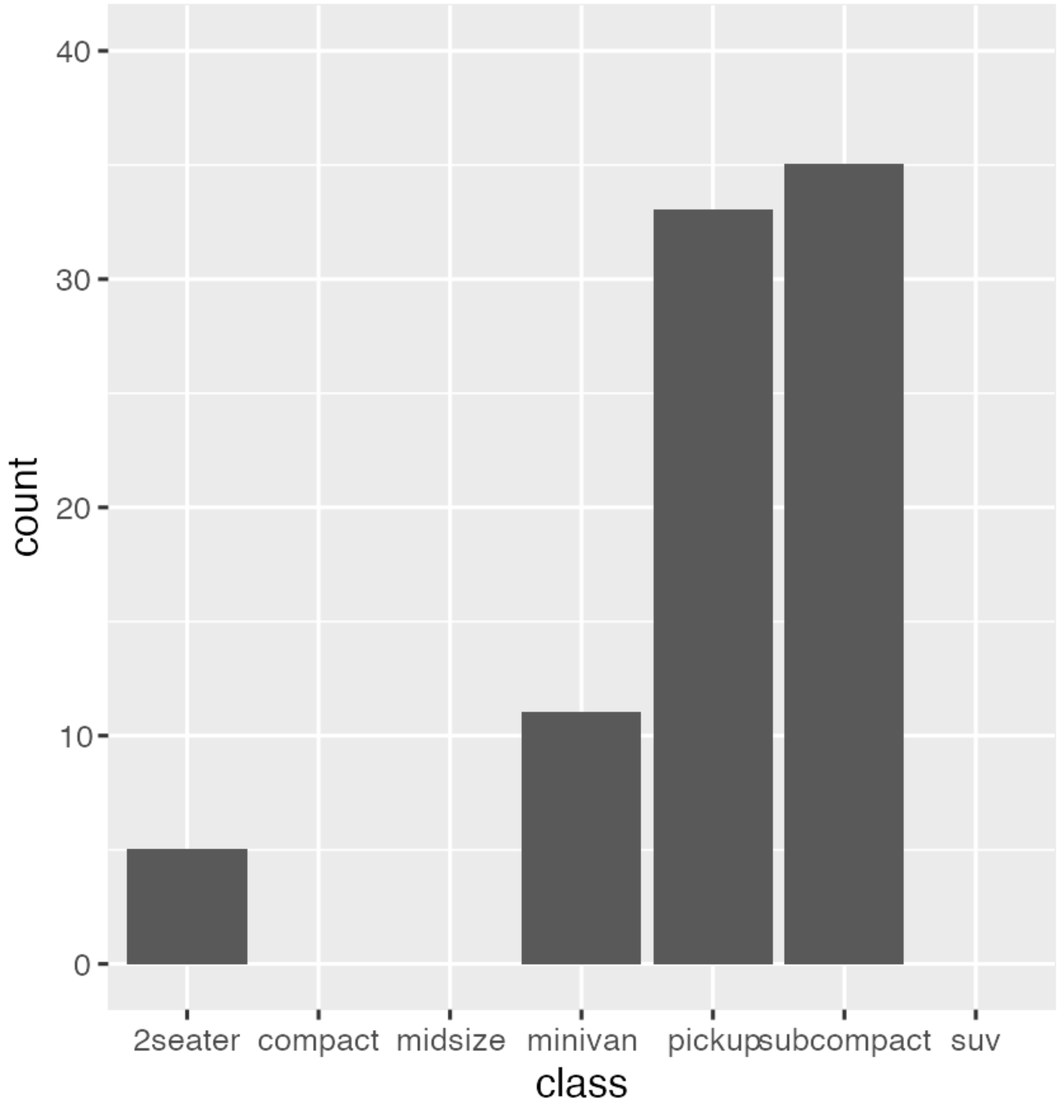
```
ggplot(mpg) +
  geom_bar(aes(x = class)) +
  coord_polar(theta = 'y') +
  expand_limits(y = 70)
```

Changing what is mapped to angle gives a very different plot



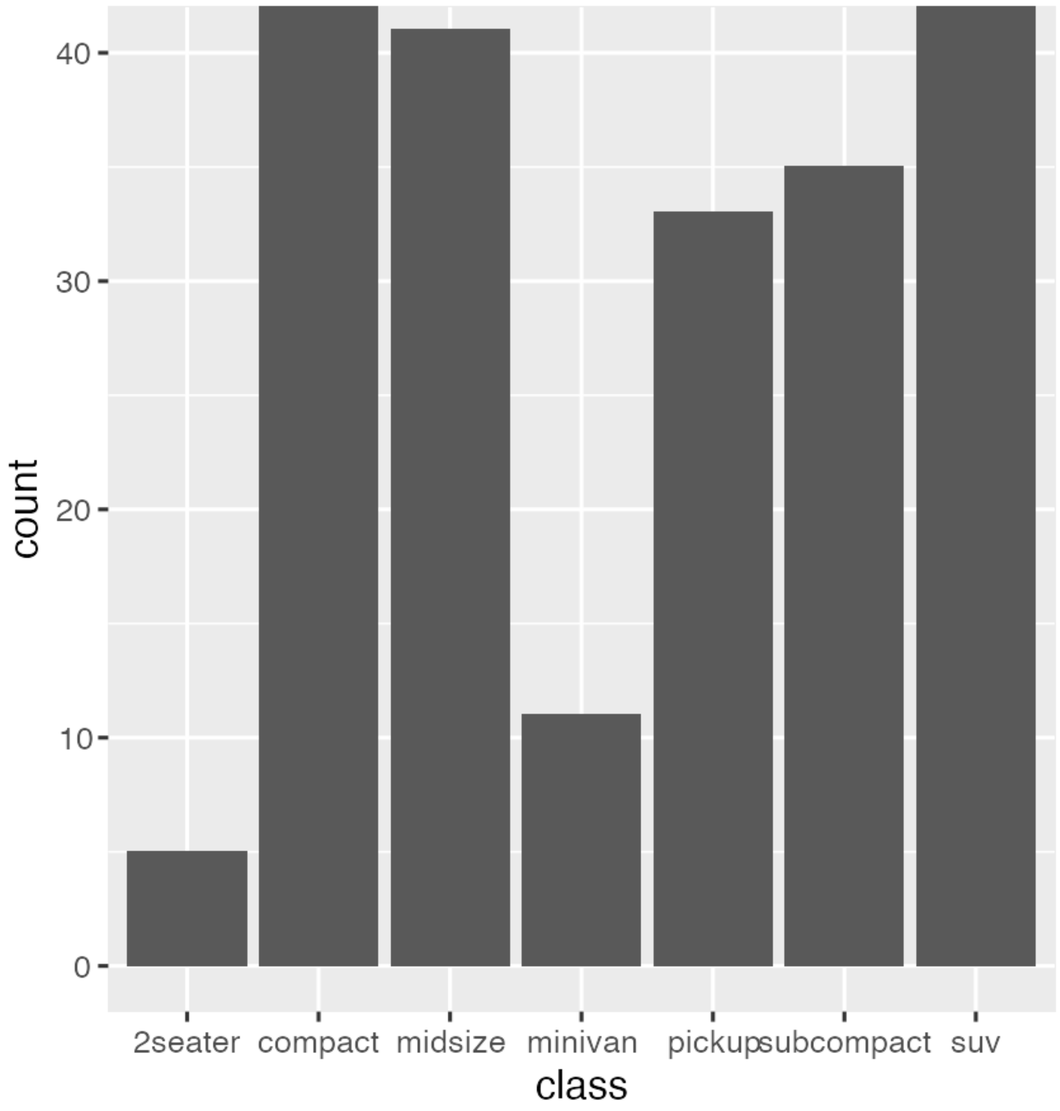
```
ggplot(mpg) +
  geom_bar(aes(x = class)) +
  scale_y_continuous(limits = c(0, 40))
#> Warning message:
#> Removed 3 rows containing missing values
(geom_bar).
```

Zooming with scale removes data outside limits



```
ggplot(mpg) +
  geom_bar(aes(x = class)) +
  coord_cartesian(ylim = c(0, 40))
```

Zooming with coord creates a proper zoom

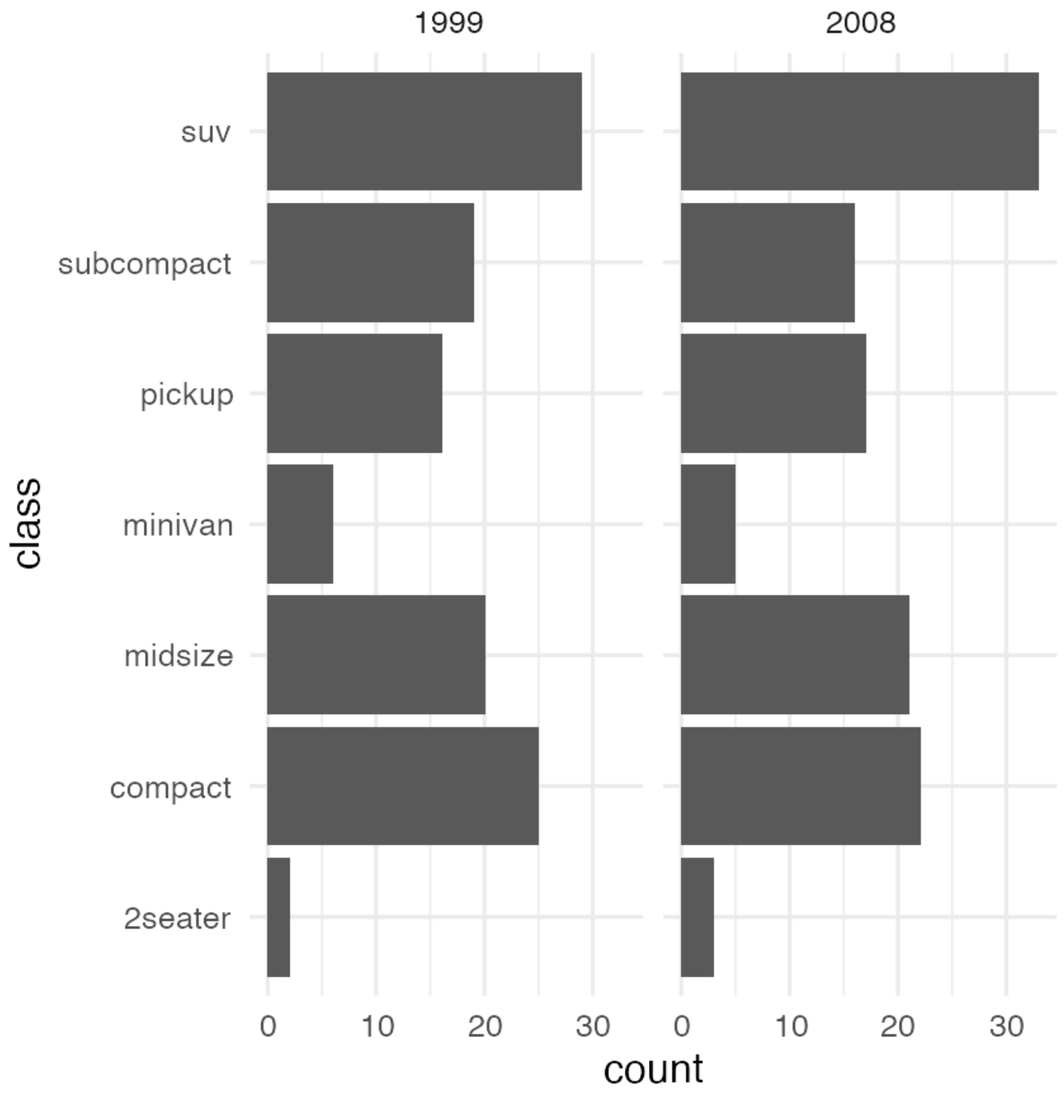


THEME

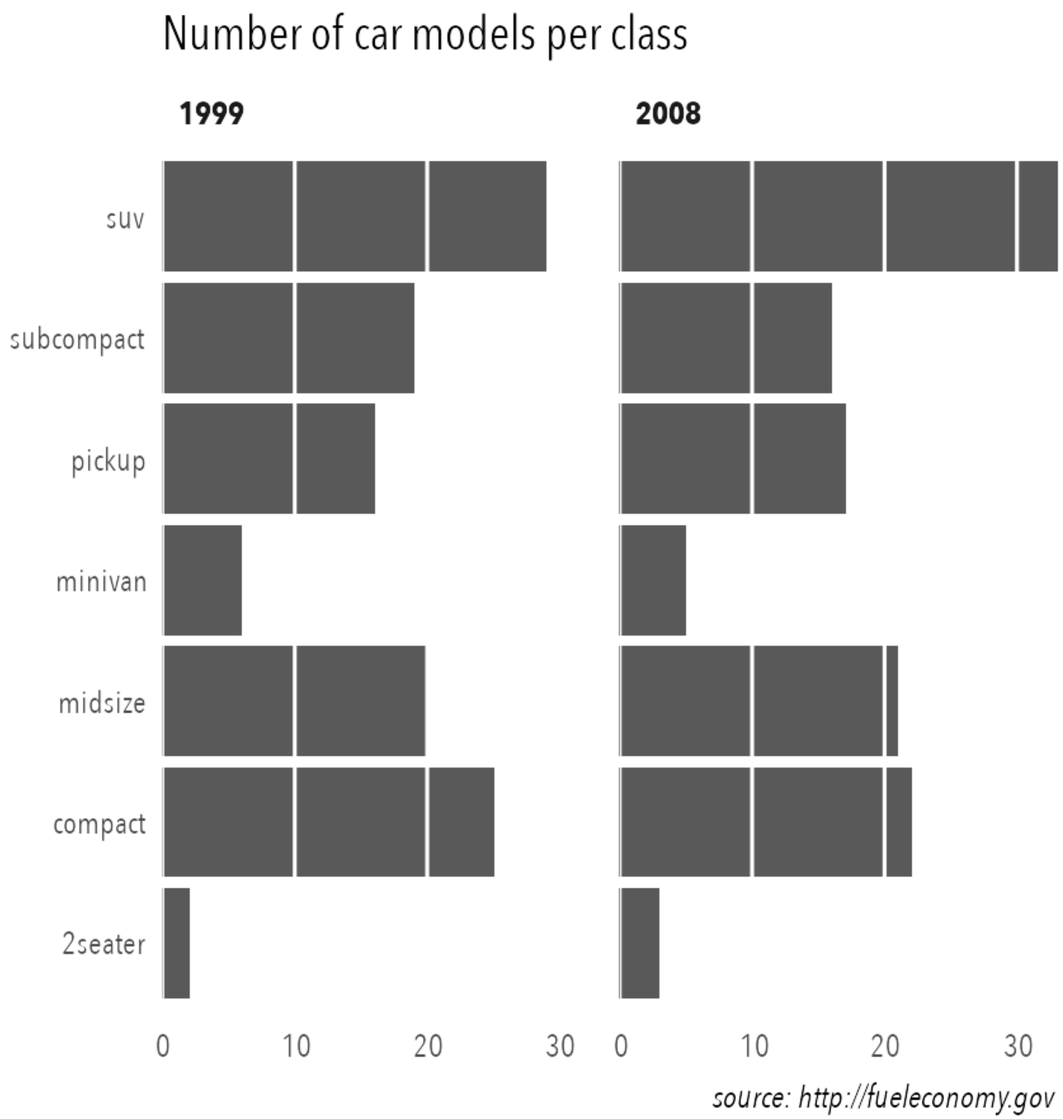
- Stylistic changes to the plot not related to data
- Can both apply complete themes or modify elements directly
- Theming is hierarchical

```
ggplot(mpg) +
  geom_bar(aes(y = class)) +
  facet_wrap(~year) +
  theme_minimal()
```

It is quick to change the overall look with a build-in theme...

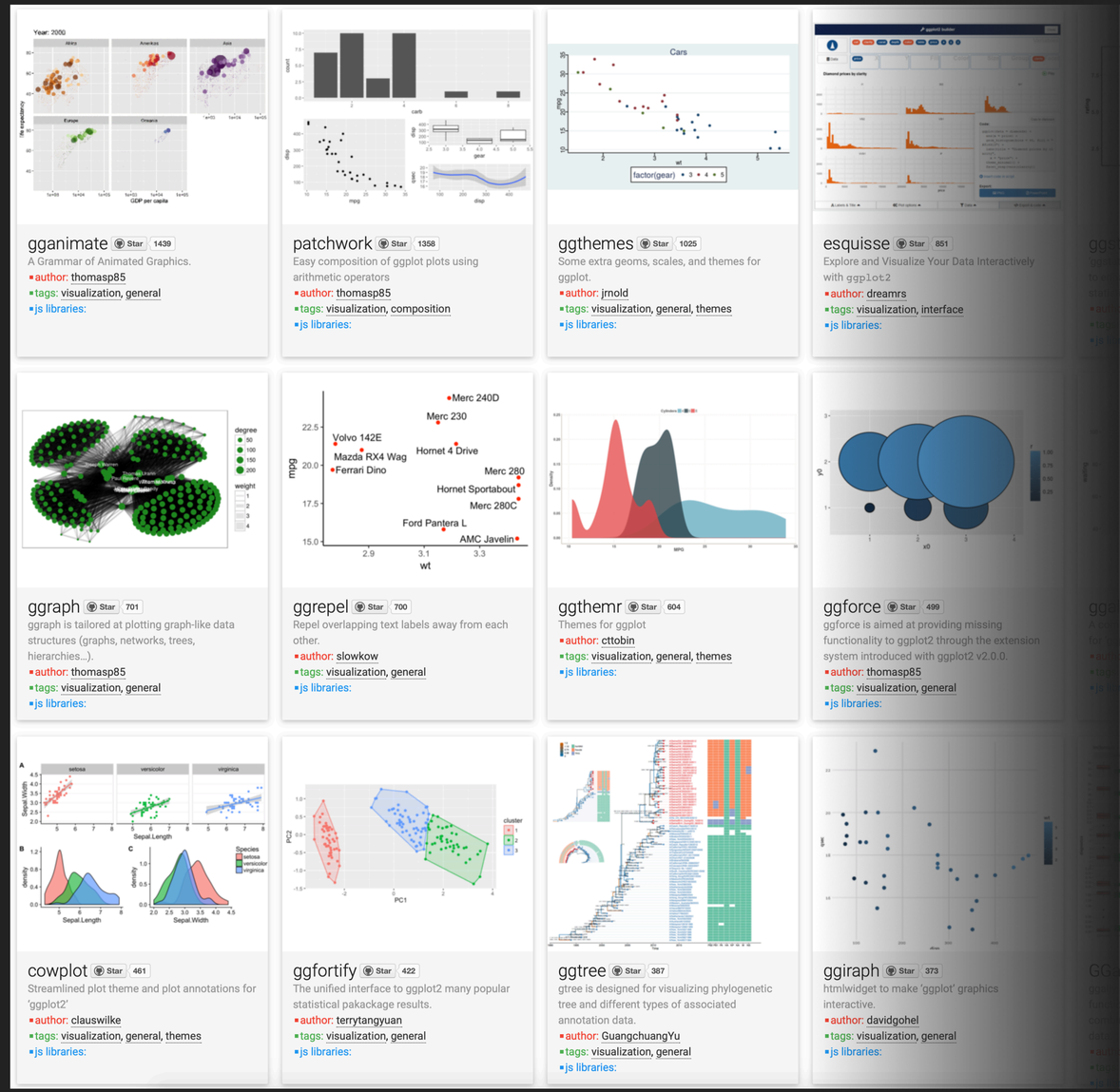


```
ggplot(mpg) +
  geom_bar(aes(y = class)) +
  facet_wrap(~year) +
  labs(title = "Number of car models per class",
       caption = "source: http://fueleconomy.gov",
       x = NULL,
       y = NULL) +
  scale_x_continuous(expand = c(0, NA)) +
  theme_minimal() +
  theme(
    text = element_text('Avenir Next Condensed'),
    strip.text = element_text(face = 'bold',
                              hjust = 0),
    plot.caption = element_text(face = 'italic'),
    panel.grid.major = element_line('white',
                                    size = 0.5),
    panel.grid.minor = element_blank(),
    panel.grid.major.y = element_blank(),
    panel.ontop = TRUE
  )
```



WHY EXTENSIONS

- ggplot2 is huge! Maintenance is already a team effort
  - 47 geoms
  - 25 stats
  - 62 scales
- Many extensions are very niche specific and better developed by experts in the field
- It is easier to promote focused packages
- www.ggplot2-exts.org

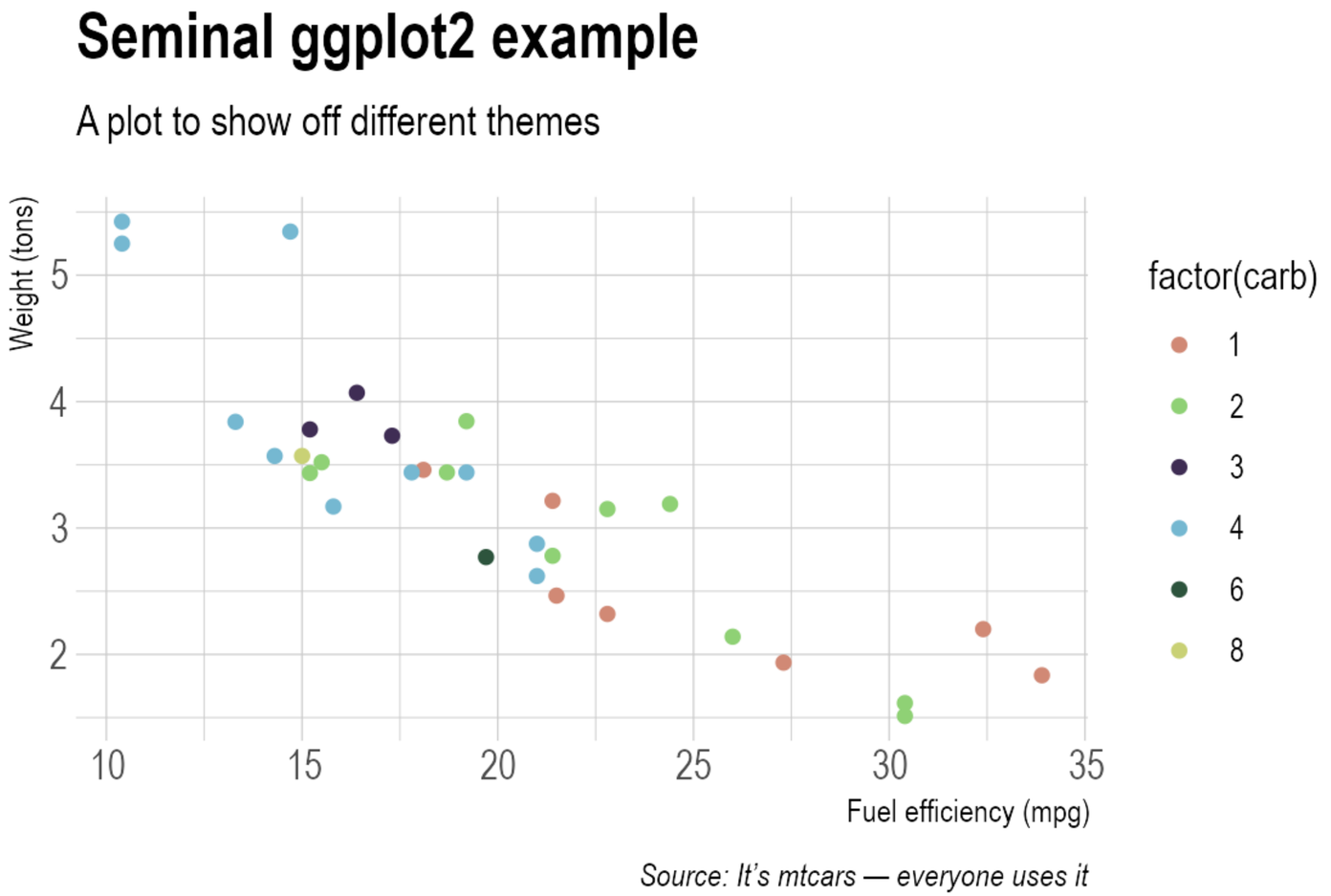


LOOKS

- Theming is one of the (programmatically) easiest things to extend
- Lots of options:
  - ggthemes
  - tvthemes
  - ggtech
  - ggthemr
  - hrbrthemes
- Often coupled with colour scales
- Don’t go too crazy unless warranted

```
p ← ggplot(mtcars, aes(mpg, wt)) +
  geom_point(aes(color = factor(carb))) +
  labs(
    x = 'Fuel efficiency (mpg)',
    y = 'Weight (tons)',
    title = 'Seminal ggplot2 example',
    subtitle = 'A plot to show off
different themes',
    caption = 'Source: It’s mtcars —
everyone uses it'
  )
```

```
library(hrbrthemes)

p +
  scale_colour_ipsum() +
  theme_ipsum()
```



```
library(ggthemes)

p +
  scale_colour_excel() +
  theme_excel()
```