

# Your First Visualization

PSY 410: Data Science for Psychology

Dr. Sara Weston

2026-04-01

## Why visualize?

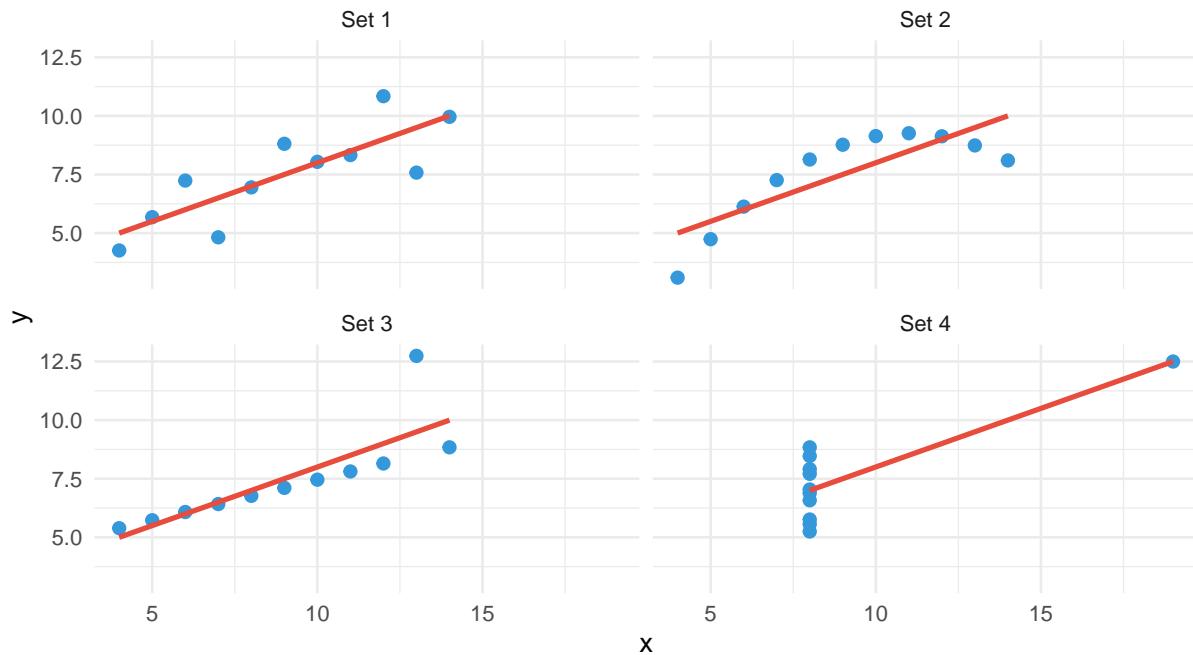
### Anscombe's Quartet

Four datasets with identical summary statistics:

set	mean_x	mean_y	sd_x	sd_y	cor
1	9	7.5	3.32	2.03	0.82
2	9	7.5	3.32	2.03	0.82
3	9	7.5	3.32	2.03	0.82
4	9	7.5	3.32	2.03	0.82

**But look at the plots!**

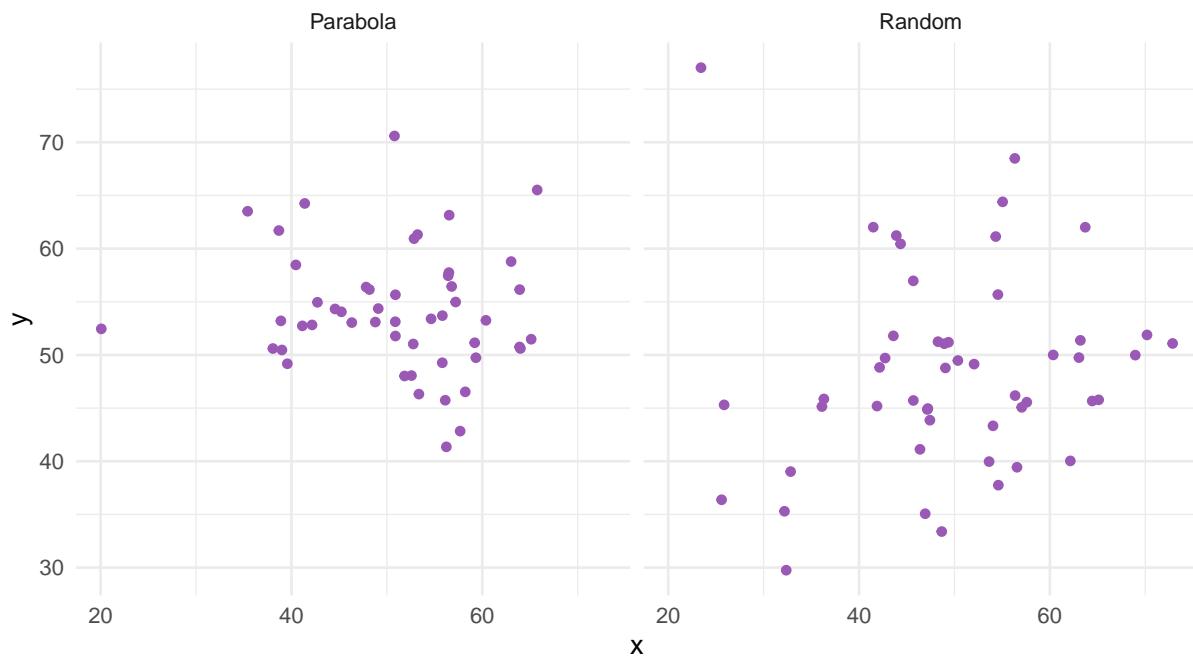
Same statistics, very different data!



Always visualize your data before running statistics.

## The datasaurus dozen

Different patterns can have similar summary stats



## Introduction to ggplot2

**ggplot2** builds plots in layers, like a grammar

- Created by Hadley Wickham (2005)
- Based on the **Grammar of Graphics** by Leland Wilkinson
- Most popular R visualization package
- Part of the tidyverse

The “gg” stands for “Grammar of Graphics”

## The grammar of graphics

Every ggplot has three essential components:

1. **Data** — what you want to visualize
2. **Aesthetics (aes)** — how variables map to visual properties

### 3. Geoms — what geometric shapes represent the data

...

```
# The basic template
ggplot(data = <DATA>, mapping = aes(<MAPPINGS>)) +
  <GEOM_FUNCTION>()
```

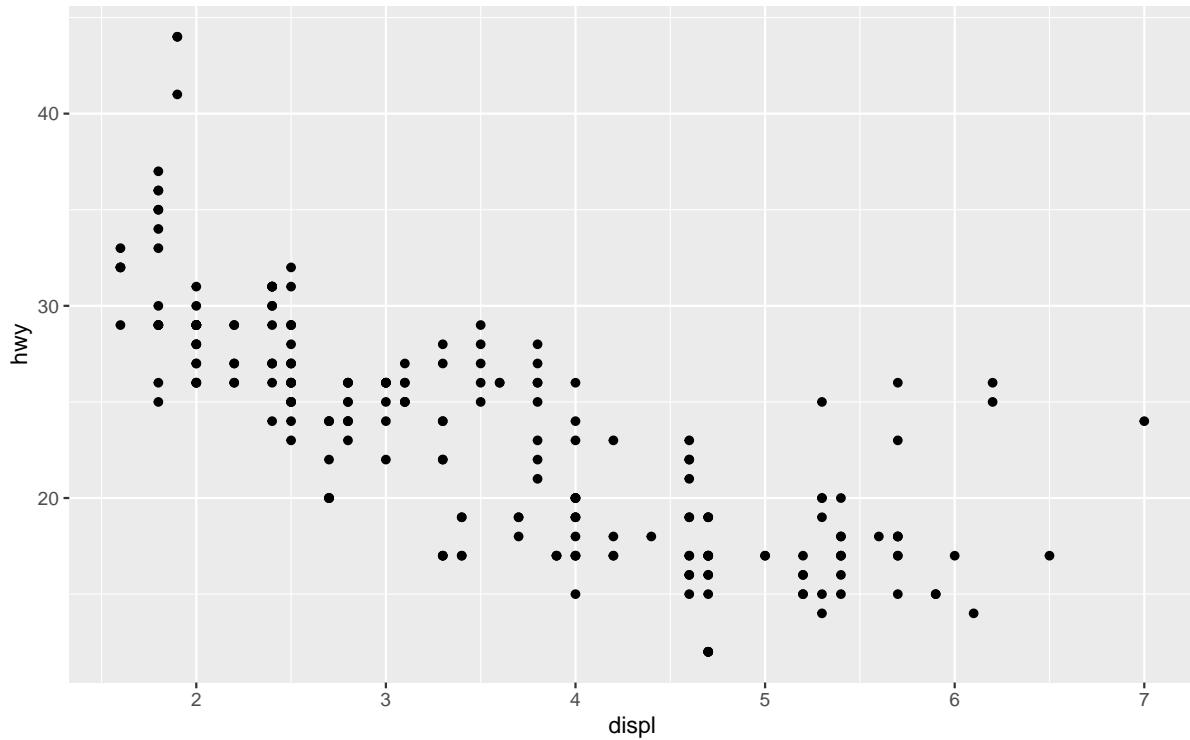
### Our dataset: mpg

```
# Fuel economy data for 234 cars
glimpse(mpg)
```

```
Rows: 234
Columns: 11
$ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi", "~
$ model         <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro", "~
$ displ          <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, 2.0, 2.~
$ year           <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, 200~
$ cyl            <int> 4, 4, 4, 4, 6, 6, 4, 4, 4, 6, 6, 6, 6, 6, 8, 8, ~
$ trans          <chr> "auto(15)", "manual(m5)", "manual(m6)", "auto(av)", "auto~
$ drv             <chr> "f", "f", "f", "f", "f", "f", "4", "4", "4", "4", "4", "4~
$ cty            <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17, 1~
$ hwy            <int> 29, 29, 31, 30, 26, 26, 27, 26, 25, 28, 27, 25, 25, 25, 2~
$ fl              <chr> "p", "p~
$ class          <chr> "compact", "compact", "compact", "compact", "compact", "c~
```

### Your first plot

```
# Relationship between engine size and highway mpg
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point()
```



## Breaking it down

```

1 ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
2   # ↑ Data and aesthetic mappings
3   geom_point()
4   # ↑ Geometric object (points = scatterplot)

```

- `data = mpg` — use the mpg dataset
- `aes(x = displ, y = hwy)` — map displacement to x, highway mpg to y
- `geom_point()` — represent data as points

## A cleaner way to write it

You can drop `data =` and `mapping =`:

```

# These are equivalent
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point()

```

```
ggplot(mpg, aes(x = displ, y = hwy)) +  
  geom_point()  
  
ggplot(mpg, aes(displ, hwy)) +  
  geom_point() # x and y are first args
```

I'll use the middle version — clear but not overly verbose.

## Aesthetic mappings

### What are aesthetics?

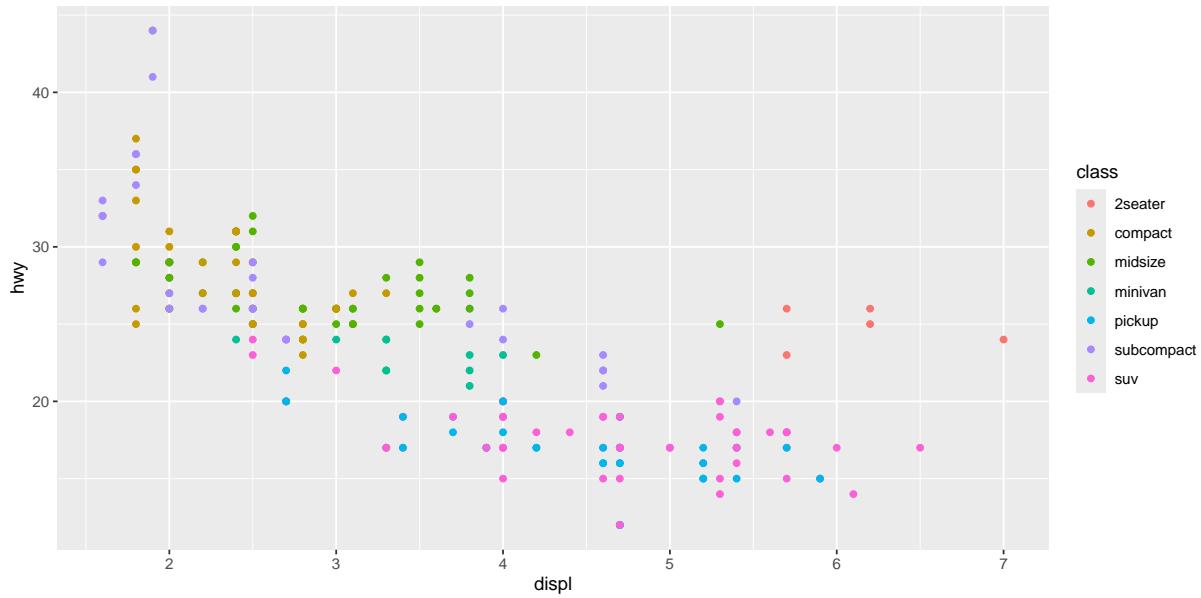
**Aesthetics** are visual properties of geoms:

- `x, y` — position
- `color` — outline color
- `fill` — interior color
- `size` — how big
- `shape` — what shape
- `alpha` — transparency

### Mapping color to a variable

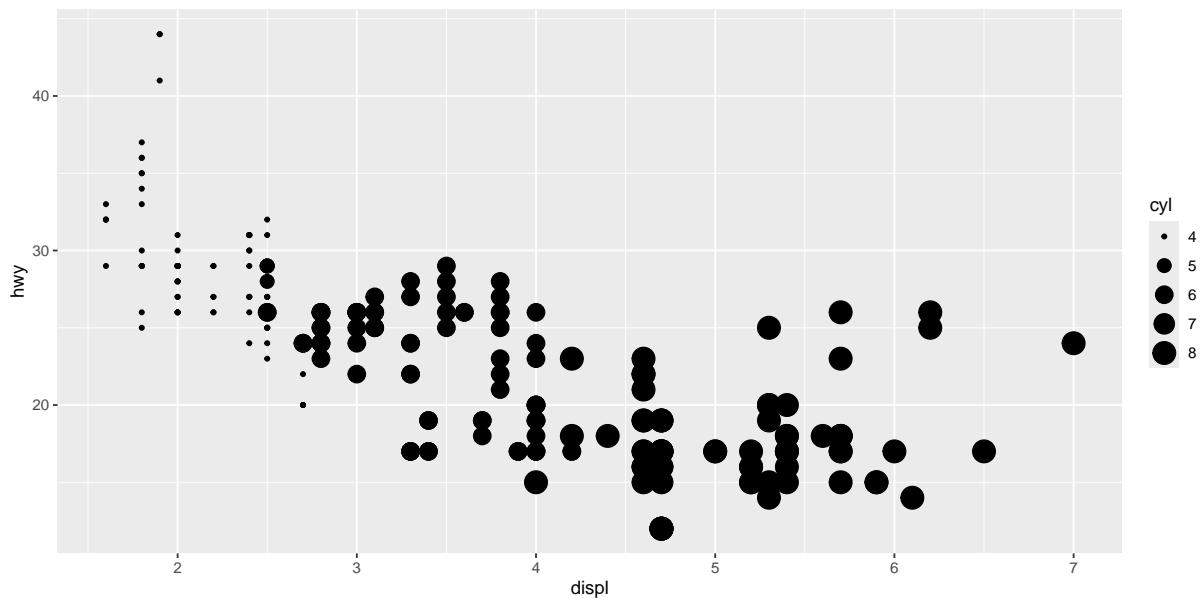
What if we want to see which points are which car class?

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



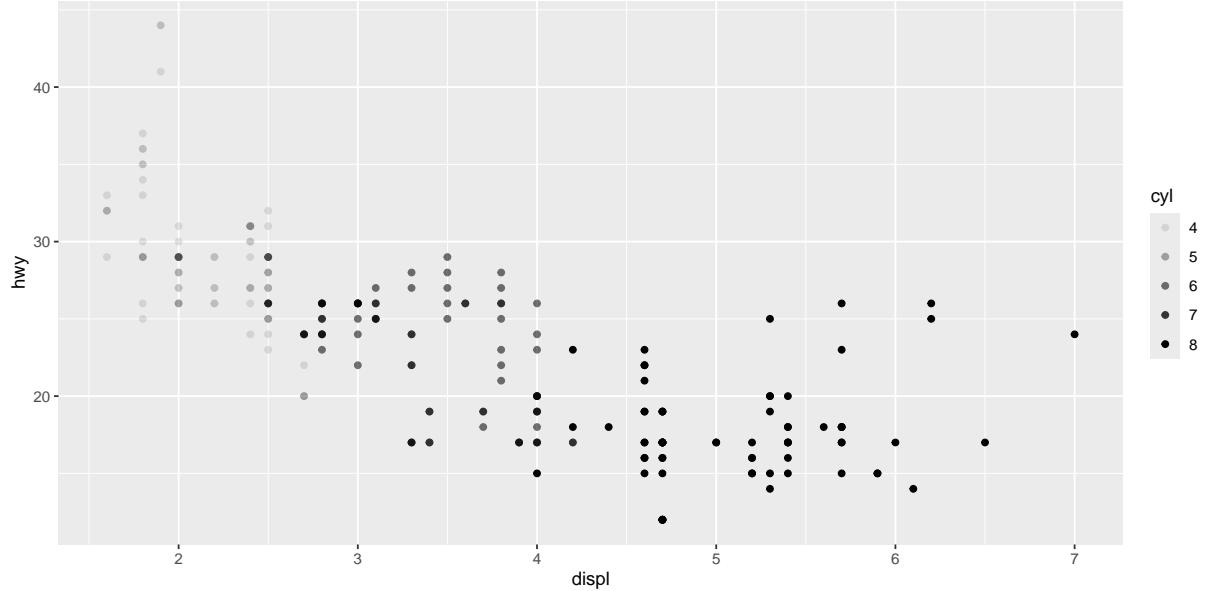
## Mapping size to a variable

```
# Size by number of cylinders
ggplot(mpg, aes(x = displ, y = hwy, size = cyl)) +
  geom_point()
```



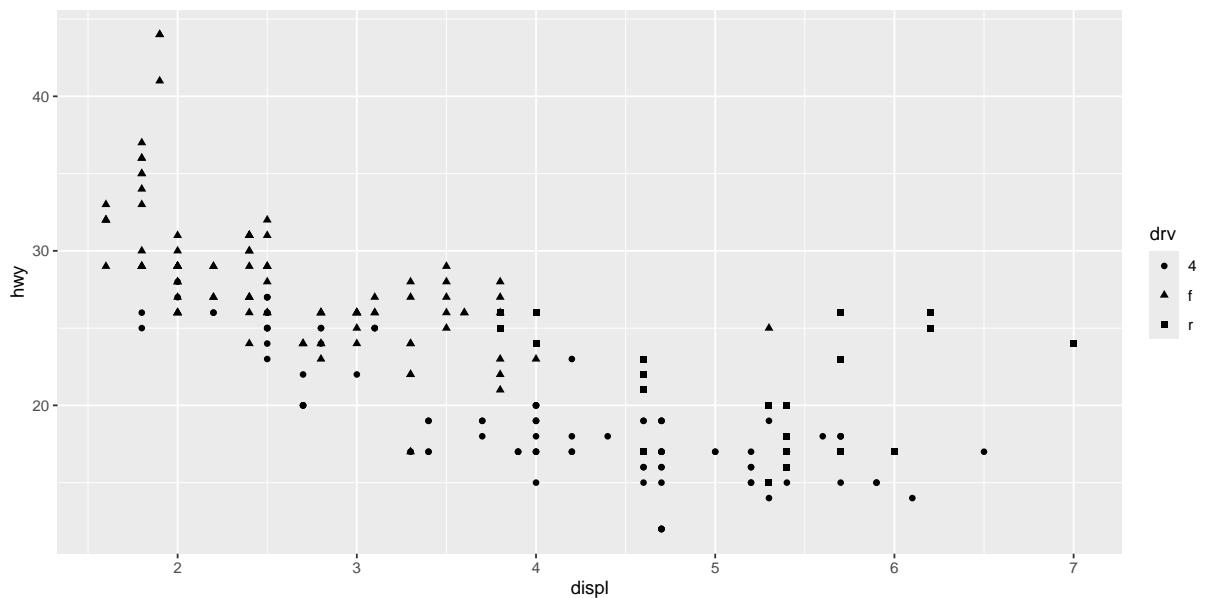
## Mapping alpha (transparency)

```
ggplot(mpg, aes(x = displ, y = hwy, alpha = cyl)) +  
  geom_point()
```



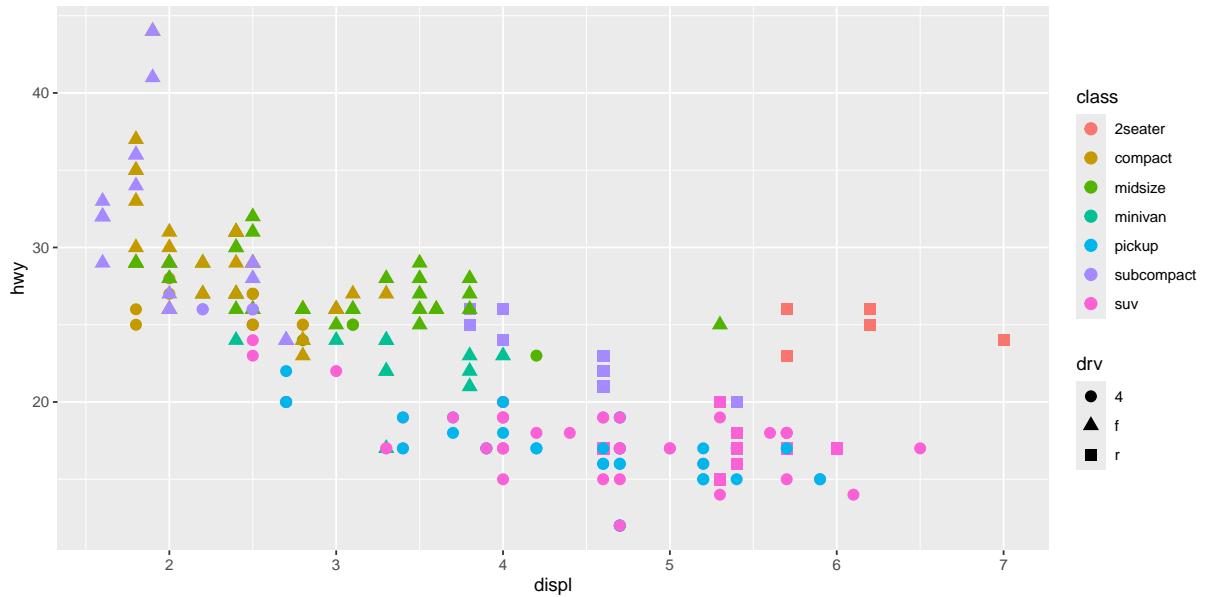
## Mapping shape

```
# Shape by drive type (front, rear, 4wd)  
ggplot(mpg, aes(x = displ, y = hwy, shape = drv)) +  
  geom_point()
```



## Combining multiple aesthetics

```
ggplot(mpg, aes(x = displ, y = hwy, color = class, shape = drv)) +
  geom_point(size = 3)
```



## Setting vs. mapping

Mapping — aesthetic varies with data (inside `aes()`)

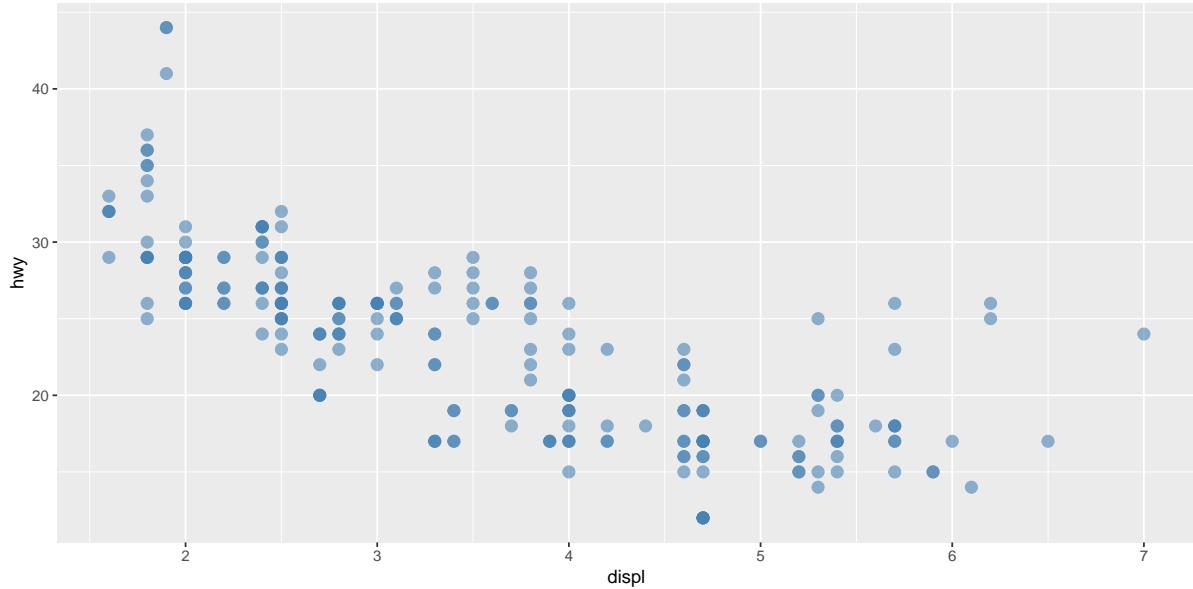
```
ggplot(mpg, aes(x = displ, y = hwy, color = class)) +  
  geom_point()
```

Setting — aesthetic is constant (outside `aes()`)

```
ggplot(mpg, aes(x = displ, y = hwy)) +  
  geom_point(color = "blue", size = 3)
```

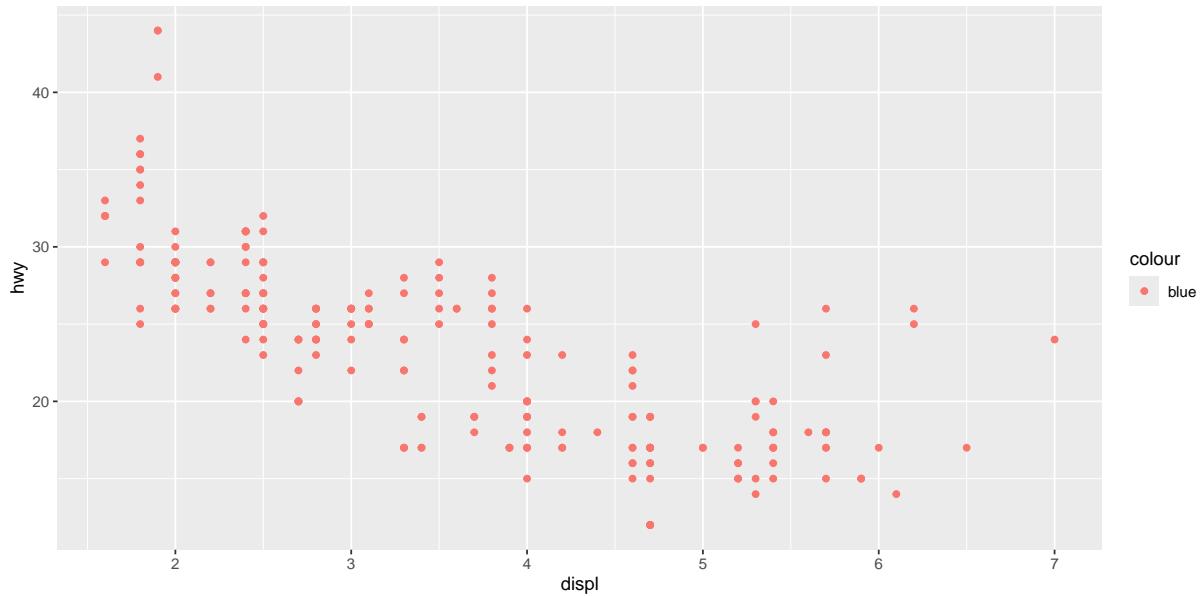
## Setting aesthetics manually

```
ggplot(mpg, aes(x = displ, y = hwy)) +  
  geom_point(color = "steelblue", size = 3, alpha = 0.6)
```



## Common mistake!

```
# What happens if you put a constant inside aes()?  
ggplot(mpg, aes(x = displ, y = hwy, color = "blue")) +  
  geom_point()
```



ggplot thinks “blue” is a category name!

## Pair coding break

**Your turn: 10 minutes**

With a partner, create a scatterplot using the `mpg` dataset:

1. Plot `cty` (x-axis) vs `hwy` (y-axis)
2. Color points by **fuel type** (`f1`)
3. Add a smooth trend line
4. Give it a title and axis labels
5. Who can make theirs look the best?

### 💡 Tip

You have everything you need from the last few slides. Start with the basic template and build from there.

## Before we move on

**Upload your code to Canvas** for participation credit. Paste what you have into today's in-class submission — it doesn't need to work perfectly.

## Different data types need different geoms

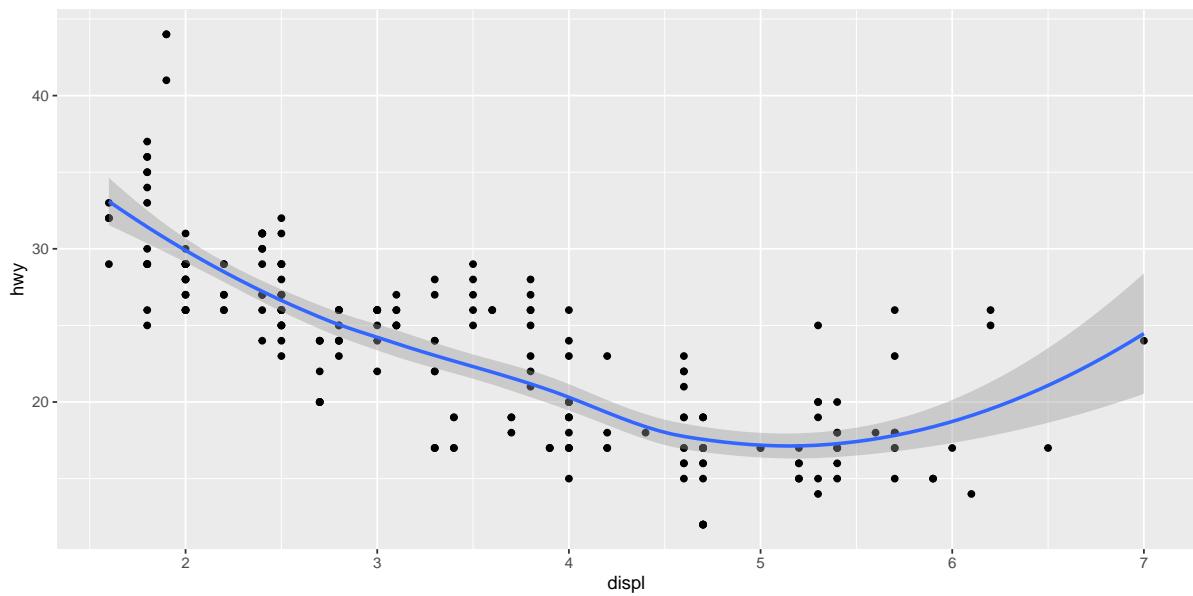
### Common geoms

Geom	What it makes
<code>geom_point()</code>	Scatterplot
<code>geom_line()</code>	Line graph
<code>geom_bar()</code>	Bar chart
<code>geom_histogram()</code>	Histogram
<code>geom_boxplot()</code>	Box plot
<code>geom_smooth()</code>	Smoothed line

### `geom_smooth()`

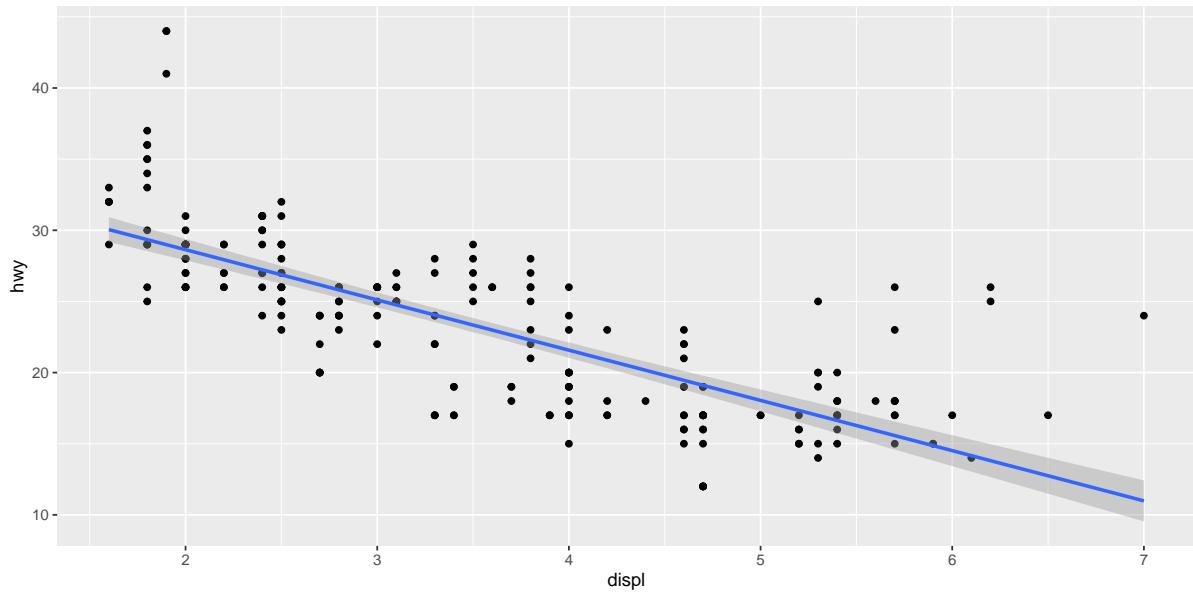
Add a trend line to your scatterplot:

```
ggplot(mpg, aes(x = displ, y = hwy)) +  
  geom_point() +  
  geom_smooth() # Adds a smoothed trend line
```



### Linear trend line

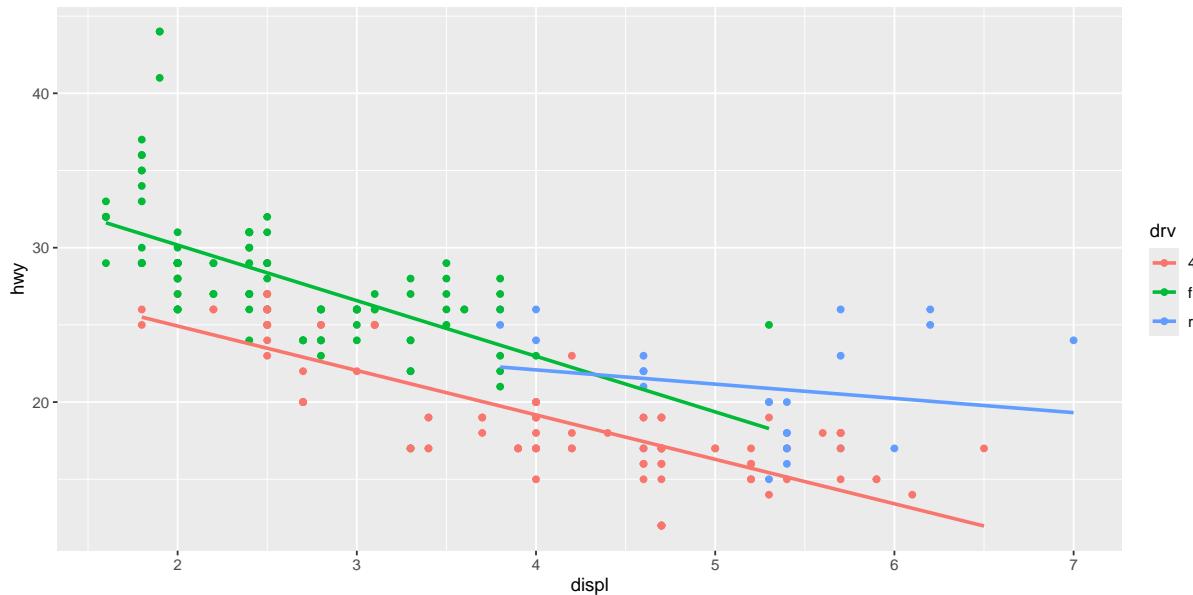
```
ggplot(mpg, aes(x = displ, y = hwy)) +  
  geom_point() +  
  geom_smooth(method = "lm") # lm = linear model
```



## Layering geoms

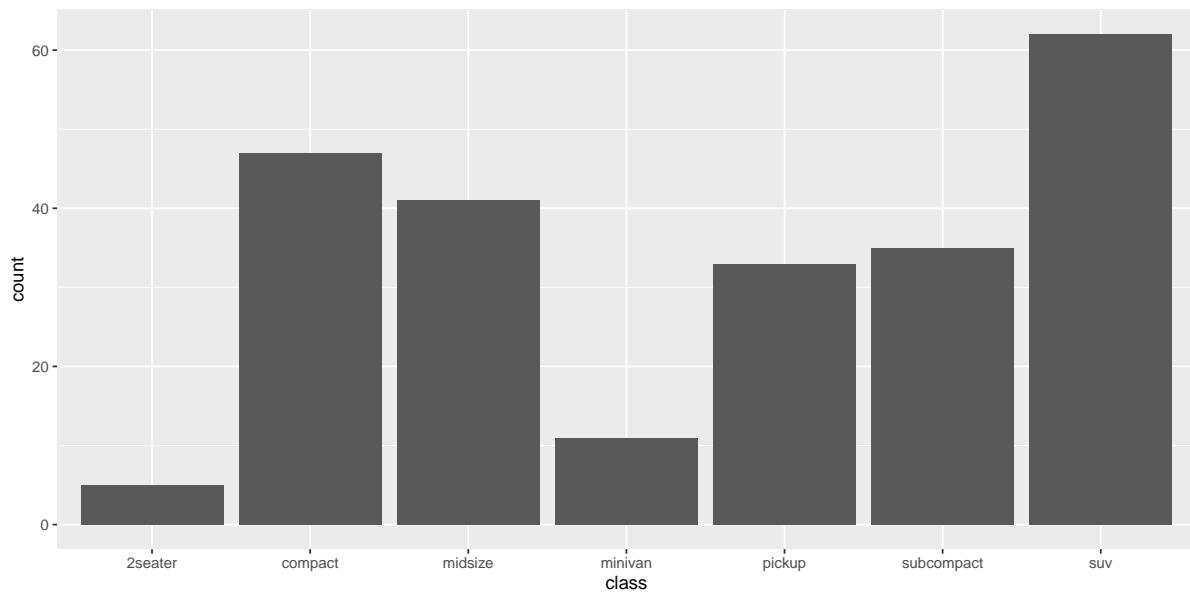
Each `+` adds a layer:

```
ggplot(mpg, aes(x = displ, y = hwy, color = drv)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) # se = FALSE removes confidence band
```



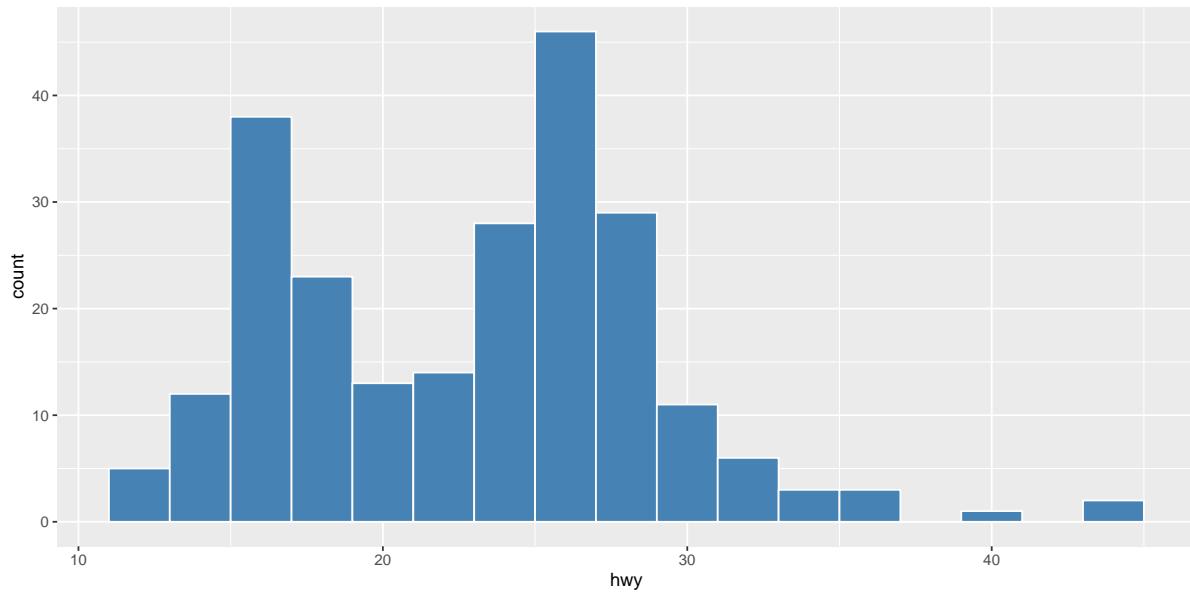
## geom\_bar() — categorical data

```
# Counts of each car class
ggplot(mpg, aes(x = class)) +
  geom_bar()
```



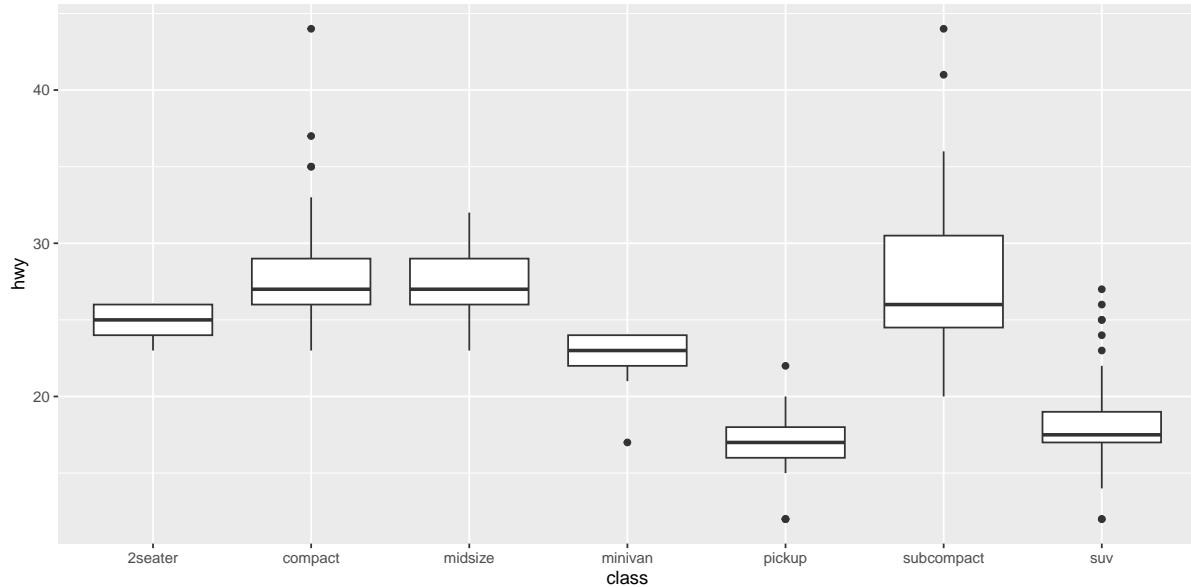
## geom\_histogram() — distributions

```
# Distribution of highway mpg
ggplot(mpg, aes(x = hwy)) +
  geom_histogram(binwidth = 2, fill = "steelblue", color = "white")
```



## `geom_boxplot()` — comparing groups

```
# Highway mpg by car class
ggplot(mpg, aes(x = class, y = hwy)) +
  geom_boxplot()
```

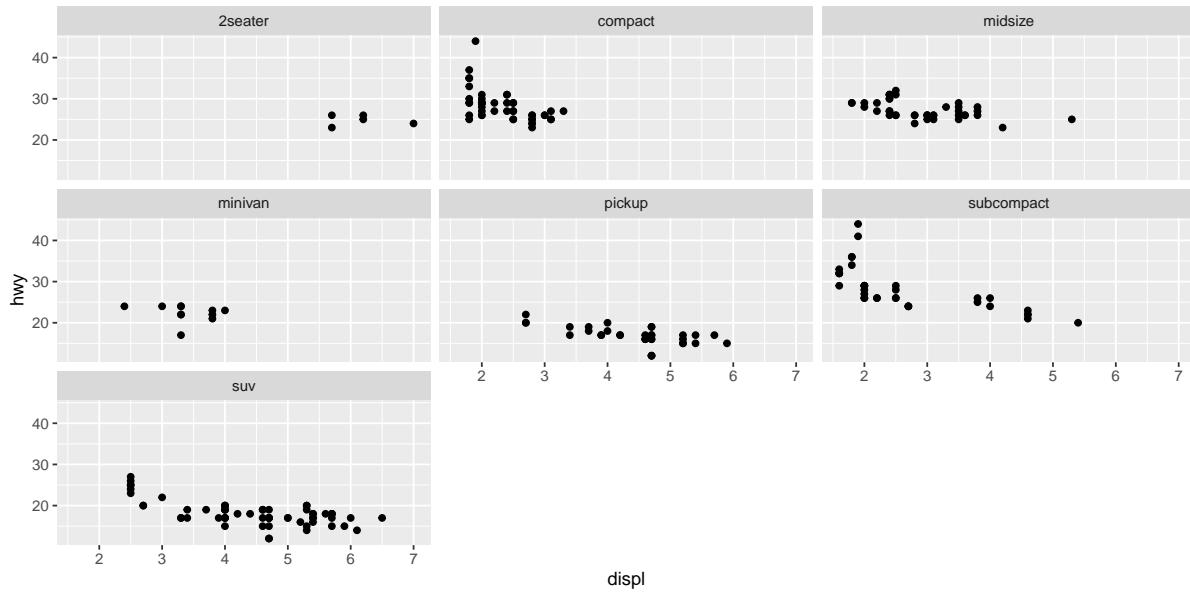


## Facets

### What are facets?

Facets split your plot into small multiples based on a variable.

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



### **facet\_wrap()**

Creates a ribbon of panels:

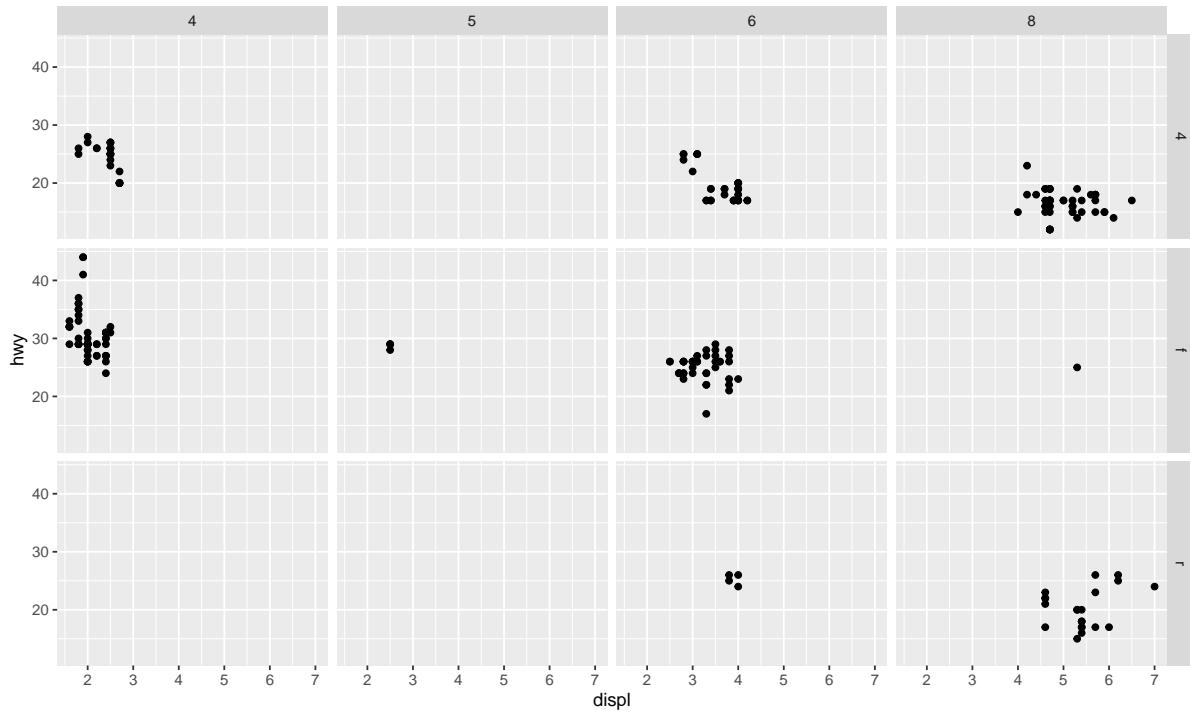
```
# Syntax: facet_wrap(~variable)
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_point() +
  facet_wrap(~class)
```

- Use `ncol` or `nrow` to control layout
- `scales = "free"` allows different axis ranges

### **facet\_grid()**

Creates a grid of panels with two variables:

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



## When to use facets

Facets are great when:

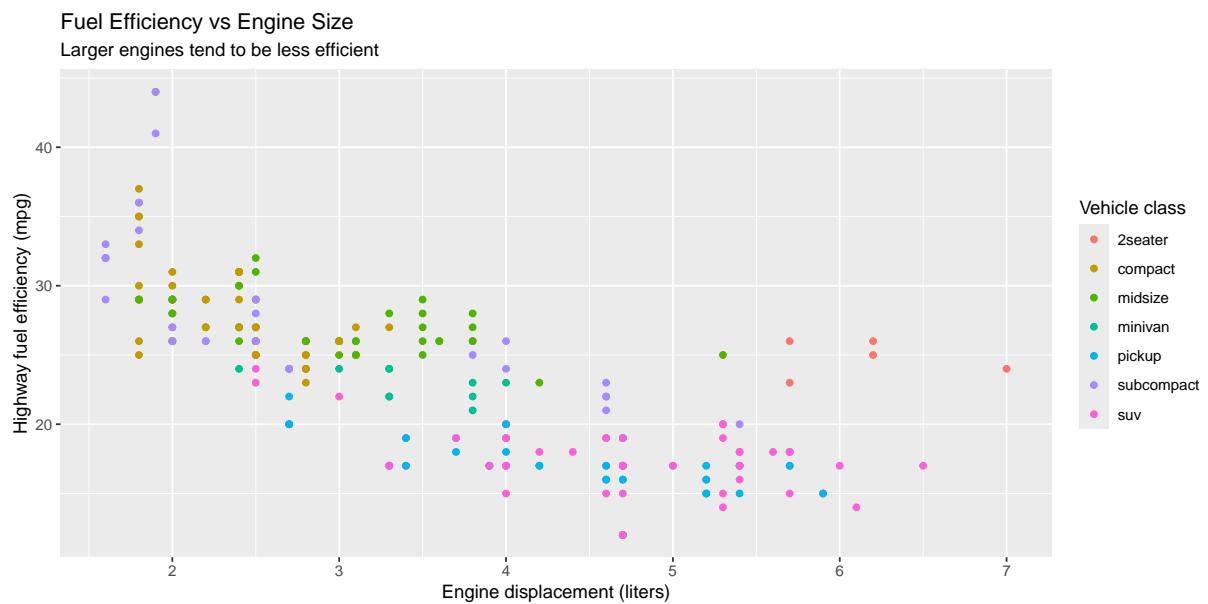
- You have too many colors/shapes to distinguish
- You want to compare patterns across groups
- You want each group to “stand alone”

## Making it look good

### Adding labels

```
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_point(aes(color = class)) +
  labs(
    title = "Fuel Efficiency vs Engine Size",
    subtitle = "Larger engines tend to be less efficient",
    x = "Engine displacement (liters)",
    y = "Highway fuel efficiency (mpg)",
```

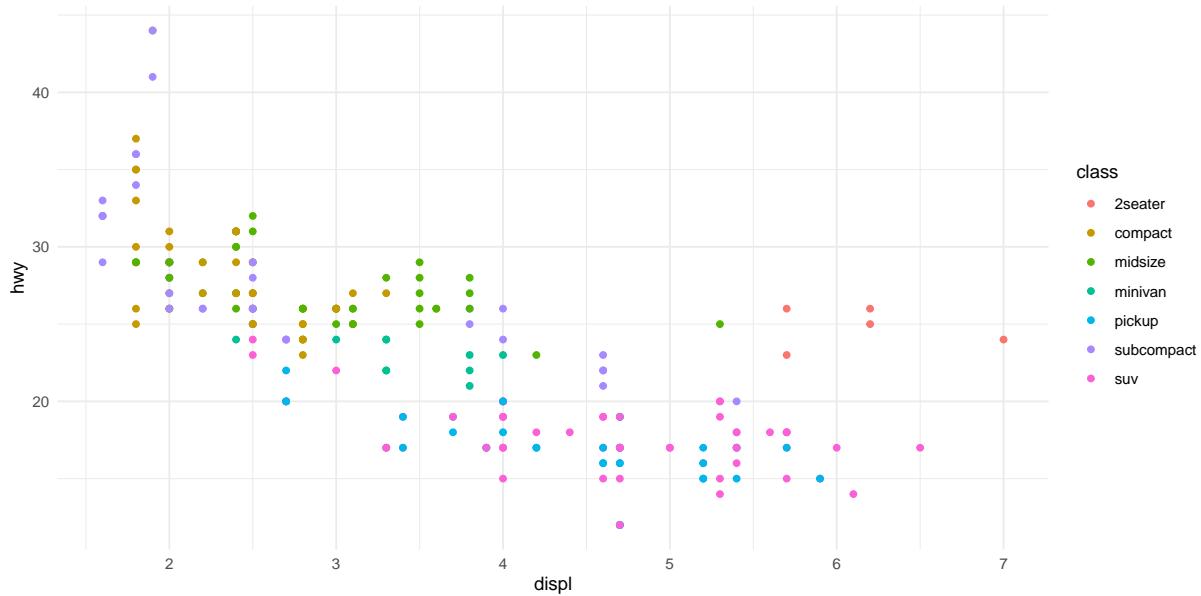
```
    color = "Vehicle class"  
)
```



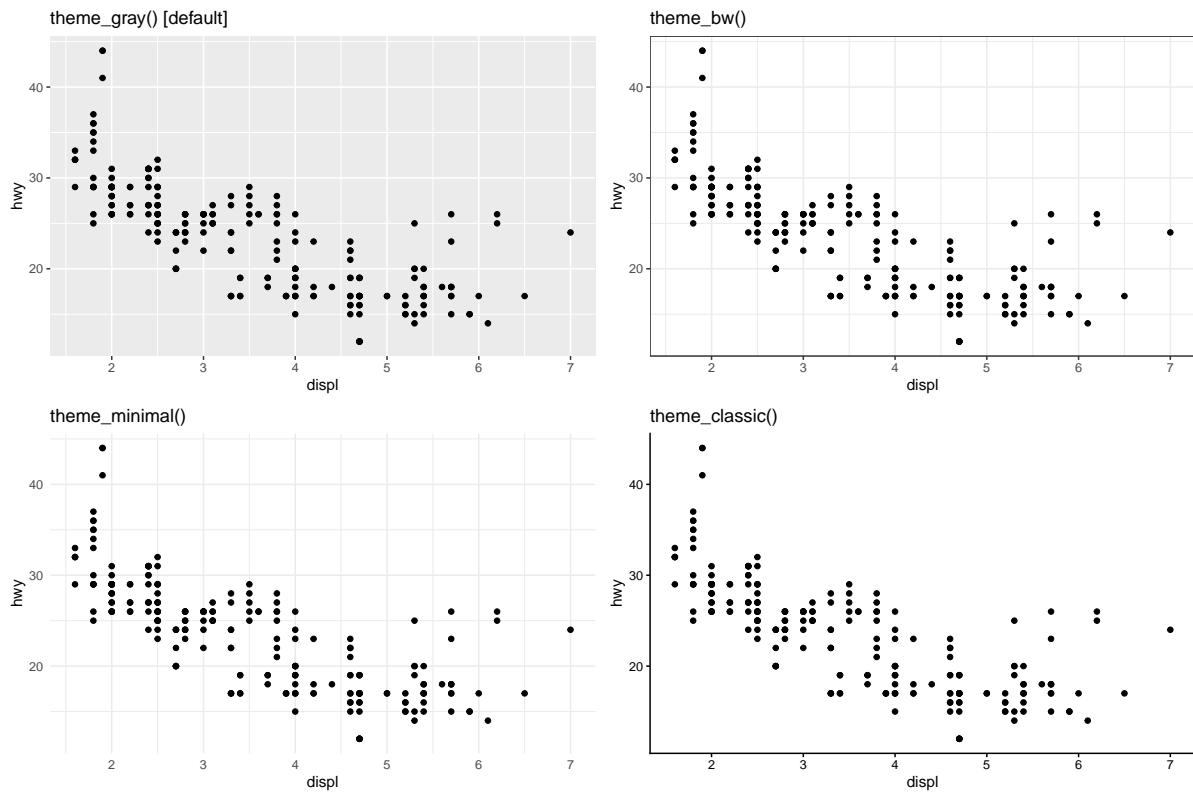
## Themes

Themes control the overall look:

```
ggplot(mpg, aes(x = displ, y = hwy)) +  
  geom_point(aes(color = class)) +  
  theme_minimal() # Clean, minimal theme
```



## Built-in themes



## Saving plots

Use `ggsave()` to save your plot:

```
# Create the plot
my_plot <- ggplot(mpg, aes(x = displ, y = hwy, color = class)) +
  geom_point() +
  theme_minimal()

# Save it
ggsave("my_plot.png", my_plot, width = 8, height = 6)
```

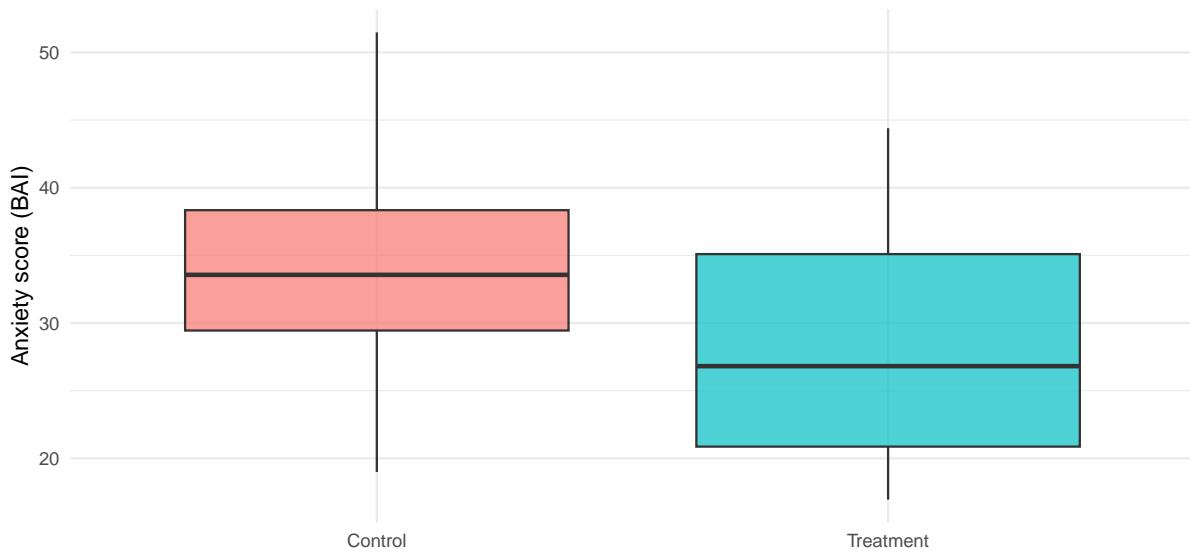
## Same template, psychology data

The `ggplot` template works the same way with any data:

```
# Simulated experiment: condition vs. anxiety score
psych_demo <- tibble(
  condition = rep(c("Control", "Treatment"), each = 30),
  anxiety = c(rnorm(30, mean = 35, sd = 8), rnorm(30, mean = 28, sd = 8))
)

ggplot(psych_demo, aes(x = condition, y = anxiety, fill = condition)) +
  geom_boxplot(alpha = 0.7, show.legend = FALSE) +
  labs(
    title = "Treatment Group Reports Lower Anxiety",
    x = NULL,
    y = "Anxiety score (BAI)"
  ) +
  theme_minimal(base_size = 14)
```

### Treatment Group Reports Lower Anxiety



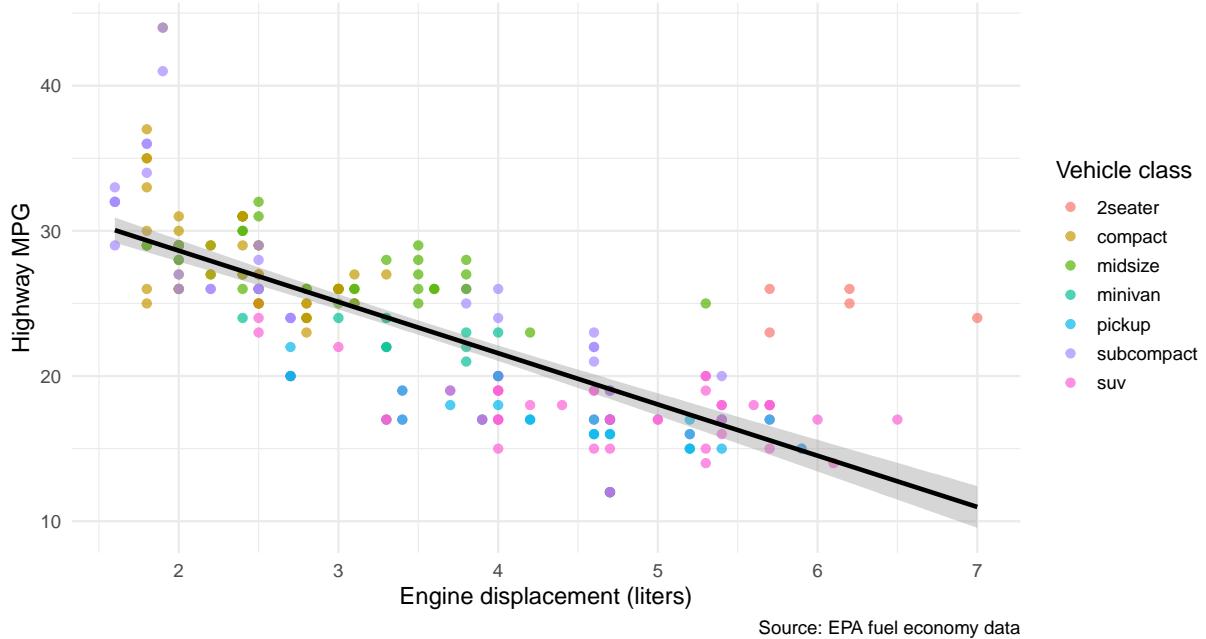
## Putting it together

### A complete example

```
ggplot(mpg, aes(x = displ, y = hwy)) +  
  geom_point(aes(color = class), size = 2, alpha = 0.7) +  
  geom_smooth(method = "lm", color = "black", se = TRUE) +  
  labs(  
    title = "Fuel Efficiency Decreases with Engine Size",  
    subtitle = "Data from 234 vehicles (1999-2008)",  
    x = "Engine displacement (liters)",  
    y = "Highway MPG",  
    color = "Vehicle class",  
    caption = "Source: EPA fuel economy data"  
) +  
  theme_minimal(base_size = 14)
```

## Fuel Efficiency Decreases with Engine Size

Data from 234 vehicles (1999–2008)



## Get a head start

### Assignment 1 preview

Open a new R script in your project. Try these on your own:

1. Make a scatterplot of `displ` vs `hwy` from `mpg`
2. Color it by `class`
3. Facet by `drv` (drive type)
4. Save it with `ggsave()`

This is the first part of Assignment 1 — finish it before next class.

## Solution

## Wrapping up

### The ggplot2 template

```
ggplot(<DATA>, aes(<MAPPINGS>)) +  
  <GEOM_FUNCTION>() +  
  <FACET_FUNCTION>() +  
  labs(<LABELS>) +  
  theme_<THEME>()
```

You can build almost any visualization with this template!

### Before next class

Read:

- [R4DS Ch 3: Data transformation](#) (sections 3.1–3.4)

Practice:

- Create 3 different plots with `mpg` or `diamonds`
- Try different geoms, aesthetics, and facets
- Save your favorite plot with `ggsave()`

### Key takeaways

1. Always visualize your data — summary stats can hide patterns
2. ggplot2 uses layers — data + aesthetics + geoms
3. Aesthetics can be mapped or set — `aes()` vs direct values
4. Facets are powerful — split plots by categories
5. Labels and themes matter — make your plots readable

### The one thing to remember

Never trust a summary statistic you haven't plotted.

Next time: Data Transformation with dplyr