Your First Visualization

PSY 410: Data Science for Psychology

Dr. Sara Weston

2026-04-01

Why visualize?

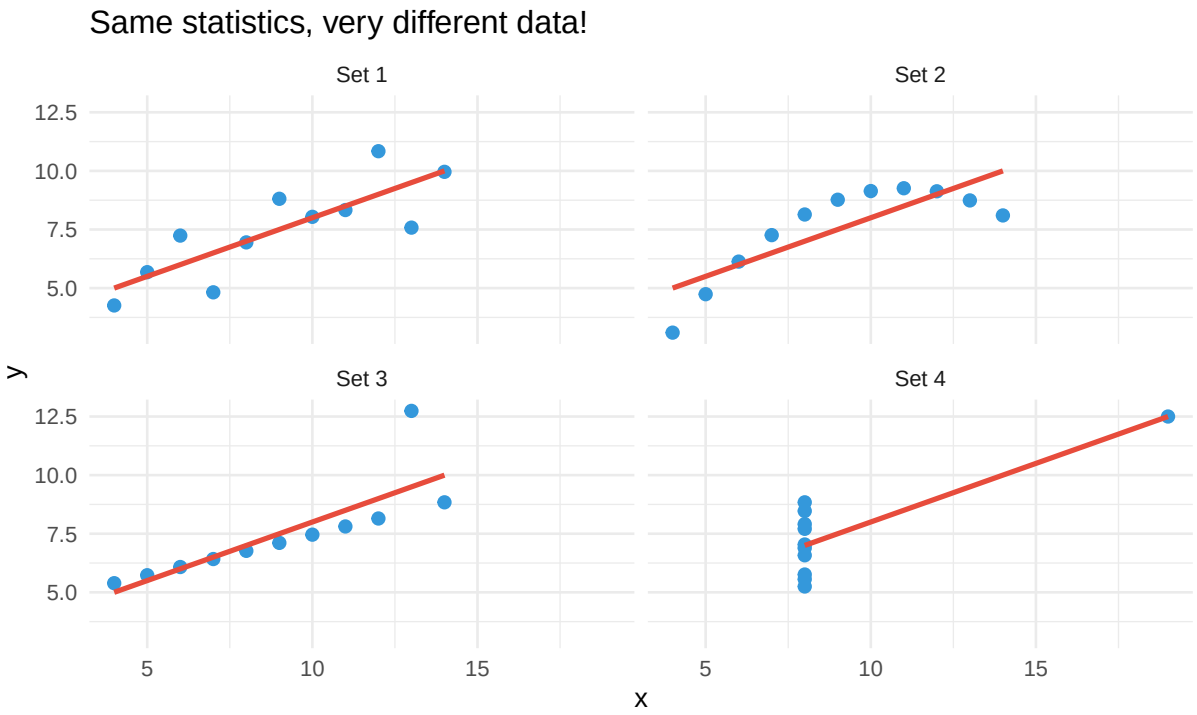
Anscombe's Quartet

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

Spend ~2 minutes here. Let students look at the table and ask them: "If you saw these summary statistics, would you think these datasets look the same?" Get a few responses before advancing. The goal is to build suspense for the reveal on the next slide.

But look at the plots!



Always visualize your data before running statistics.

This is the payoff. Let it land. Point out that Set 1 is a nice linear relationship, Set 2 is a curve, Set 3 has an outlier pulling the stats, and Set 4 is clustered with one extreme point. Emphasize: "This is why we visualize before we analyze." This connects back to the replication crisis discussion from Session 1 — researchers who only look at numbers miss things.

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"

Don't dwell on history. The key point is that ggplot2 is not a random collection of plotting commands — it has a coherent logic. The "grammar" metaphor means that just like you can combine words following grammatical rules to make any sentence, you can combine data, aesthetics, and geoms to make any plot. Students don't need to remember who Hadley Wickham is for the quiz.

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>()
```

Spend a moment on this template — it will be their mental model for the rest of the course. Emphasize that every ggplot follows this same structure. Walk through each piece slowly: "First you tell it what data, then how variables map to visual things, then what shape to draw." Students often ask why we use `+` instead of `|>` — the short answer is that ggplot2 predates the pipe and uses `+` to add layers. They'll get used to it.

Our dataset: mpg

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

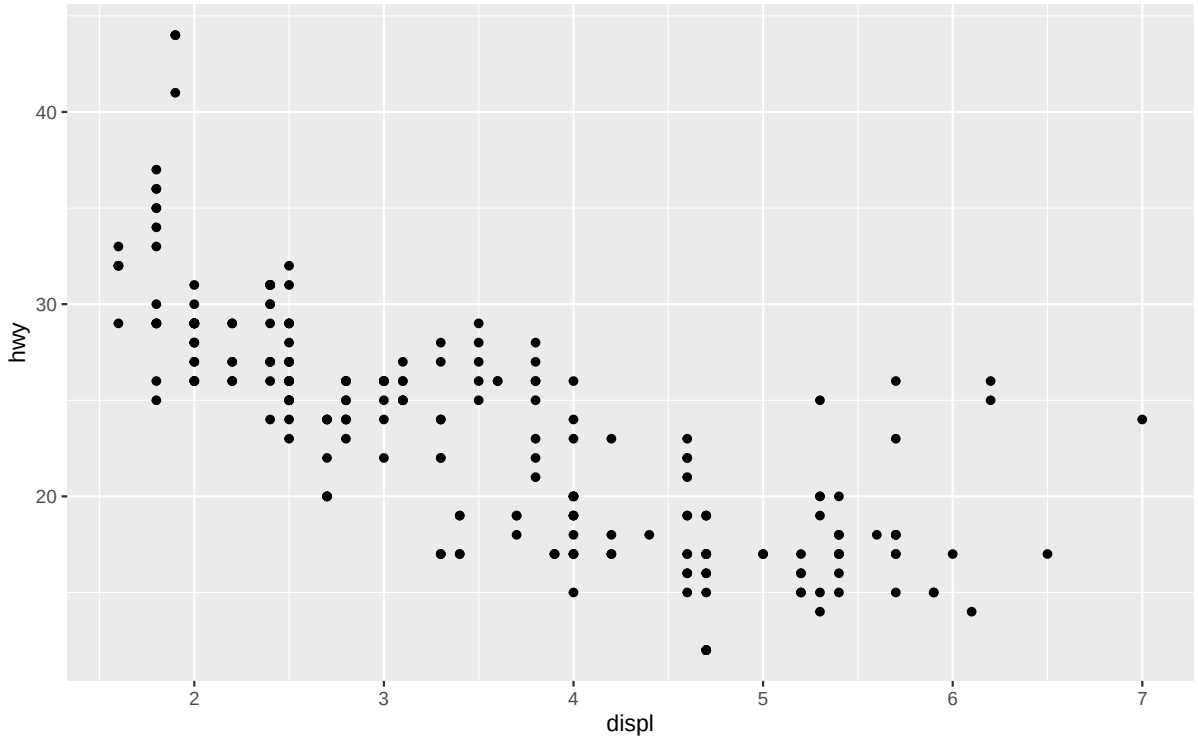
```
Rows: 234
Columns: 11
$ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi", "audi", "~
$ model        <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro", "~
$ displ        <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, 2.0, 2.~
$ year         <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, 200~
$ cyl          <int> 4, 4, 4, 4, 6, 6, 6, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 8, 8, ~
$ trans        <chr> "auto(l5)", "manual(m5)", "manual(m6)", "auto(av)", "auto~
$ drv          <chr> "f", "f", "f", "f", "f", "f", "f", "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", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p~
$ class        <chr> "compact", "compact", "compact", "compact", "compact", "c~
```

Walk through the output of `glimpse()` briefly: 234 rows means 234 cars, each column is a variable. Point out `displ` (engine displacement in liters) and `hwy` (highway miles per gallon) since those are what we'll plot first. Don't explain every column — they'll explore more in Assignment 1. If students ask what a tibble is, say it's just a modern version of a data frame.

Your first plot

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



This is a big moment — their first real plot from code. Let students react to seeing it appear. Ask: "What does this plot tell us about the relationship between engine size and fuel efficiency?" (Bigger engines, worse mileage.) Type the code live in RStudio if possible so students see the

workflow of writing code and seeing the output. Expect questions about the warning message from overplotting.

Breaking it down

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  # ↑ Data and aesthetic mappings
  geom_point()
  # ↑ 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

. . .

You can also write it without the argument names — these are equivalent:

```
ggplot(mpg, aes(x = displ, y = hwy)) +   # what I'll use in slides
  geom_point()
```

Walk through each piece slowly: "First you tell it what data, then how variables map to visual things, then what shape to draw." When mentioning the shorter form, reassure students that both versions are equivalent — you're dropping `data =` and `mapping =` since they're the first arguments. Mention they'll see even shorter versions online (`ggplot(mpg, aes(displ, hwy))`). Don't spend more than 1 minute on the shorthand.

Aesthetic mappings

What are aesthetics?

Aesthetics are visual properties of geoms:

- `x`, `y` — position
- `color` — outline or line color
- `fill` — interior color (bars, boxes)
- `size`, `shape`, `alpha` — size, shape, and transparency

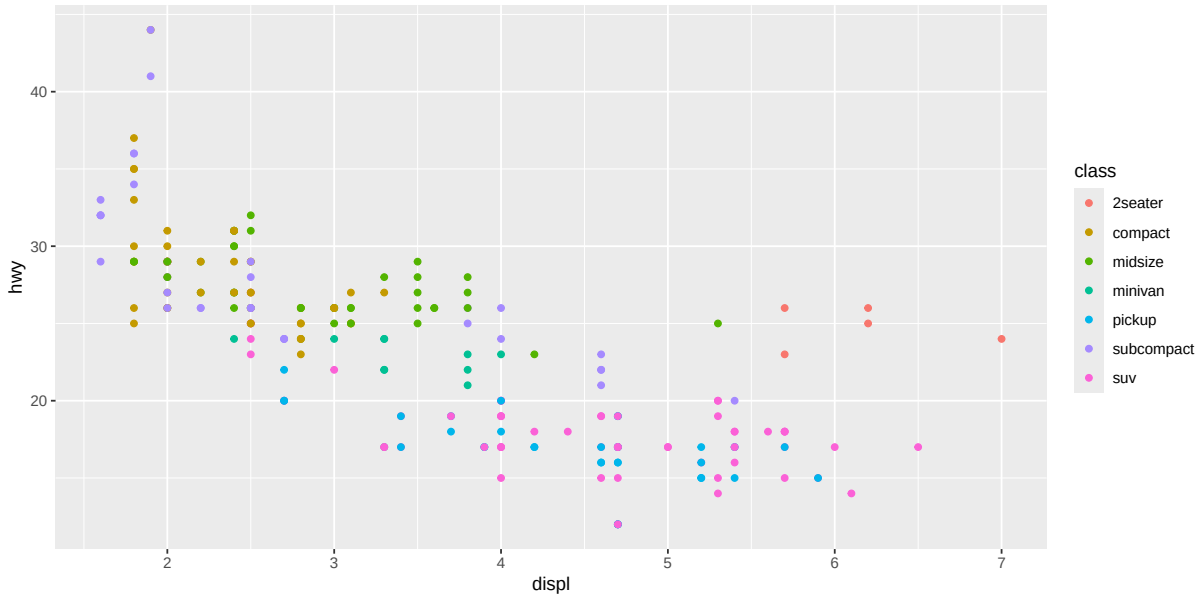
We'll focus on `color` today — you'll explore the others in assignments.

Don't demo size, shape, and alpha individually — just mention that they exist and work the same way as color (put them inside `aes()` to map a variable, or set them directly in the geom). Students will encounter them naturally in Assignment 1 and later sessions.

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()
```



This is the first time students see a third variable encoded visually. Point out how the legend appeared automatically — ggplot handles that for you. Ask: "Which car class gets the best highway mileage? Which has the biggest engines?" (Compact/subcompact do well; SUVs and pickups have big engines.) This is a good moment to connect to psych: "Imagine color = experimental condition."

Setting vs. mapping

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

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

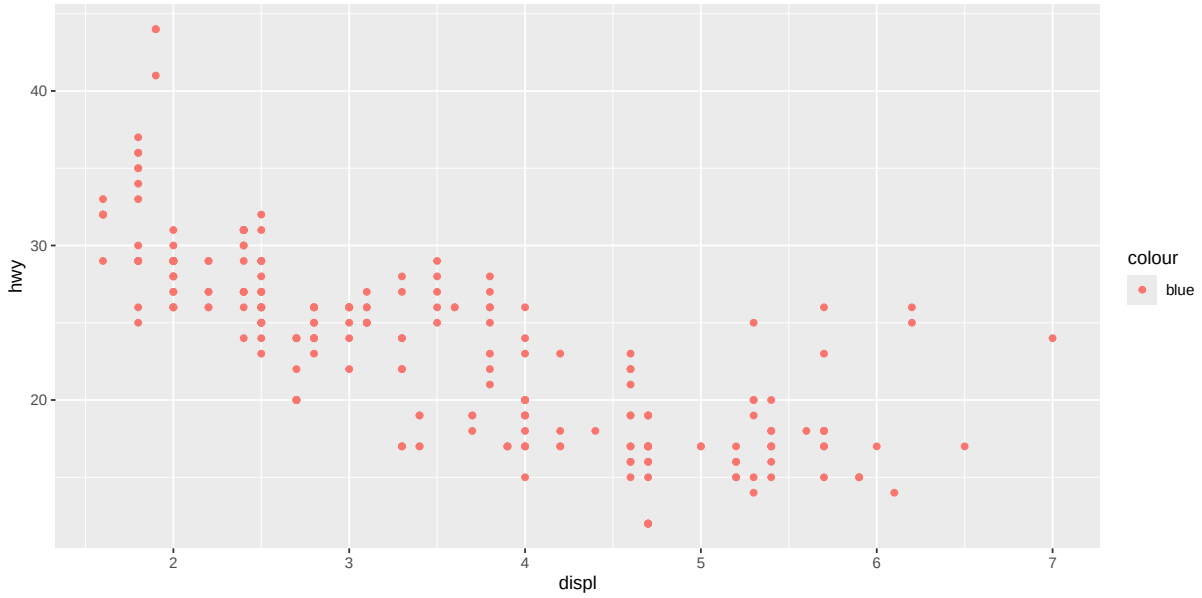
Setting — aesthetic is constant (outside `aes()`)

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

This is one of the most common sources of confusion for beginners. Emphasize the rule: if the aesthetic should vary with the data, it goes inside `aes()`. If you want every point the same color/size, it goes outside `aes()` as a direct argument to the geom. You'll see the consequence of getting this wrong on the next slide with the "common mistake" example. Spend ~2 minutes here making sure the distinction is clear before moving on.

Common mistake!

```r
# 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!

Let students laugh at this one. It's a very common mistake and worth burning into memory now. Explain: when you put `"blue"` inside `aes()`, ggplot treats it as a categorical variable with one level called "blue" and assigns it its default color (which happens to be salmon/red). The fix is simple — move it outside `aes()`. If students encounter this later, they'll hopefully remember this slide. This is a natural stopping point for the first half of class (~25 minutes in).

Different data types need different geoms

Common geoms

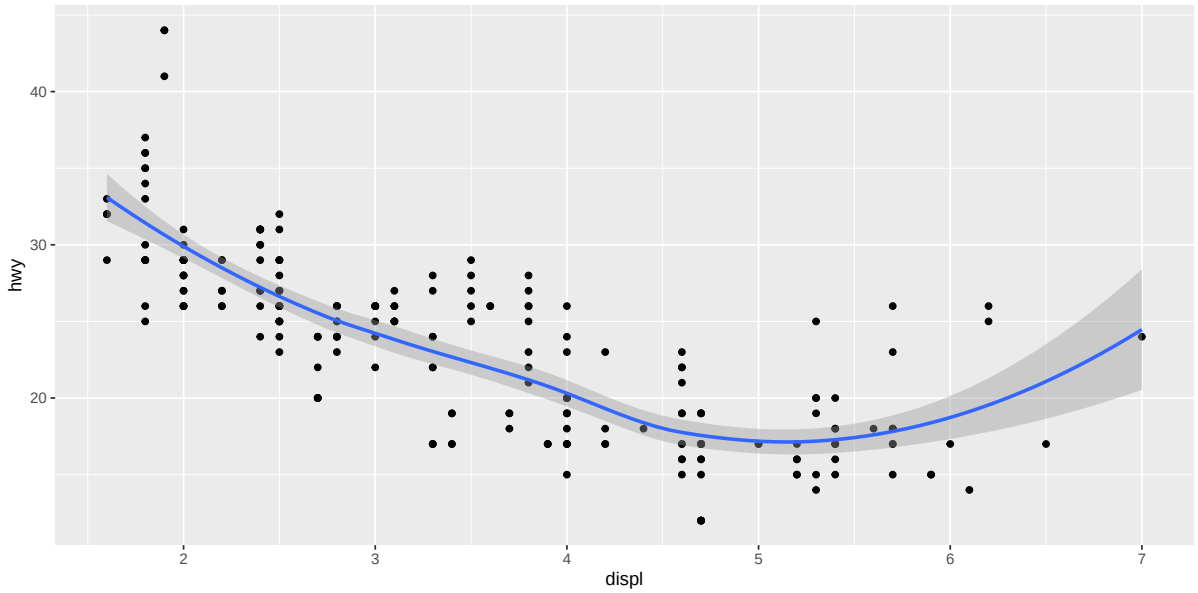
Geom	What it makes
`geom_point()`	Scatterplot
`geom_line()`	Line graph
`geom_bar()`	Bar chart
`geom_histogram()`	Histogram
`geom_boxplot()`	Box plot
`geom_smooth()`	Smoothed line

Don't try to cover every geom in detail — this is a reference slide they can come back to. Just emphasize that ggplot2 has many geoms and the choice depends on your data type. The next few slides will demo the most common ones. Students will use `geom_point()` and `geom_smooth()` most in the first few weeks; bar charts and histograms come up more in EDA sessions later.

geom_smooth()

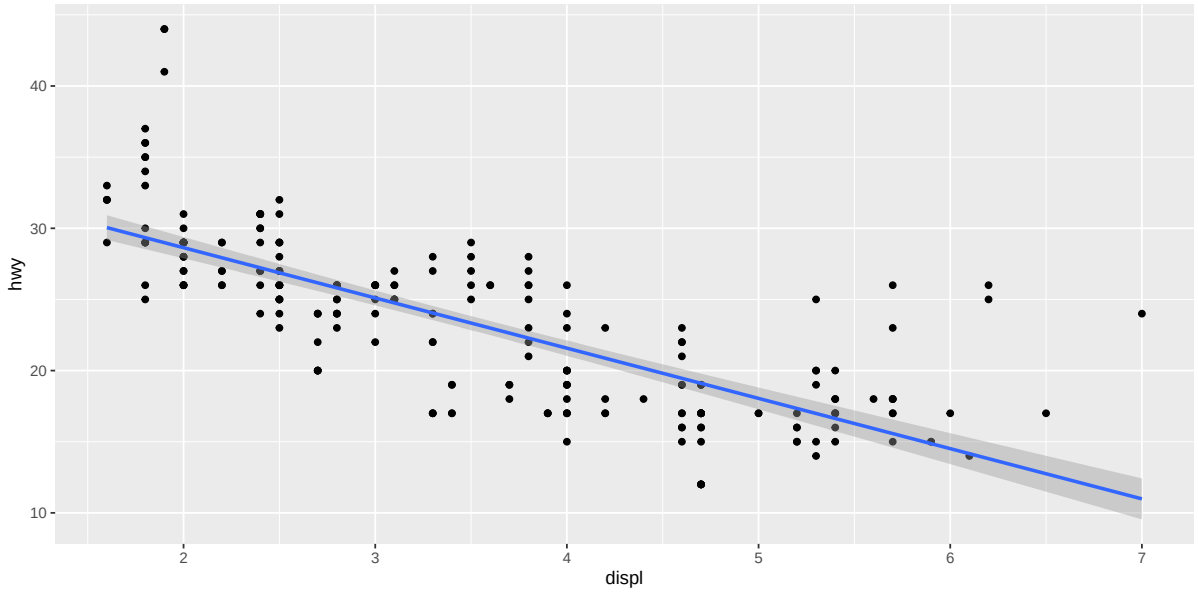
Add a trend line to your scatterplot:

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



Linear trend line

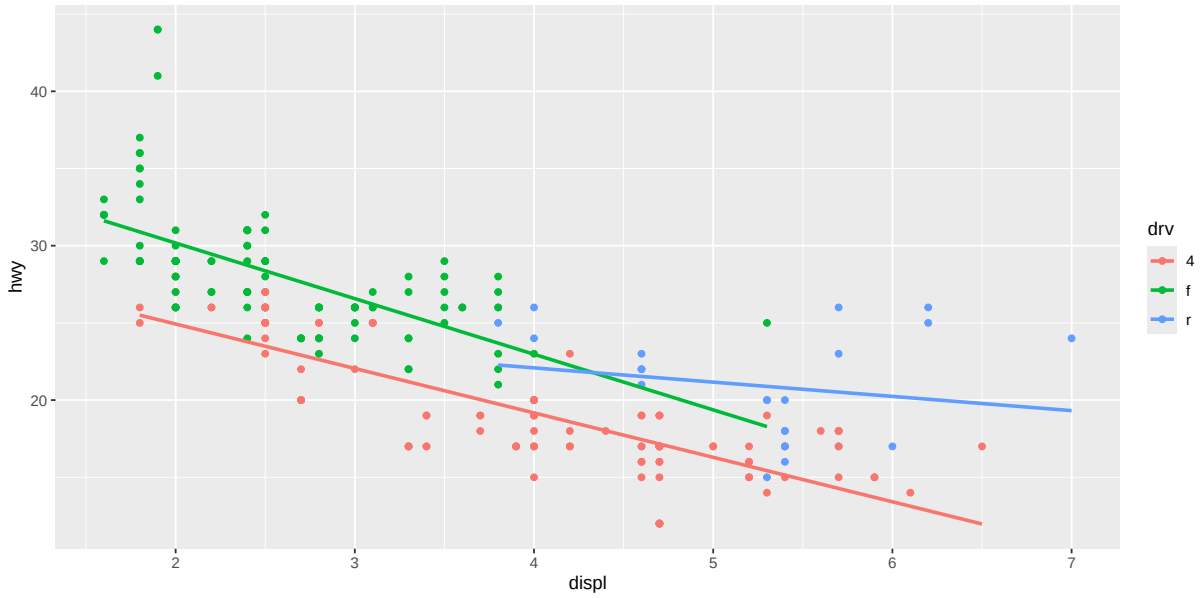
```
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth(method = "lm")  # lm = linear model
```



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
```



Point out what happened here: because `color = drv` is in the `aes()` of the main `ggplot()` call, both `geom_point()` and `geom_smooth()` inherit it, so you get separate trend lines per group. This is a preview of how layering and aesthetic inheritance work — they'll explore this more in Session 8 (Layers & Aesthetics). Students often ask what the gray band is on the previous slide — it's a confidence interval. The `se = FALSE` argument removes it.

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** (`fl`)
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.

Set a visible timer for 10 minutes. Circulate the room and help pairs who are stuck — the most common issues will be: forgetting the `+` between layers and putting `color` outside `aes()`. Encourage students to experiment beyond the requirements (try different colors, themes, alpha). For the "who can make it look the nicest" prompt, you might do a quick informal show-and-tell at the end. They haven't learned `labs()` or themes yet, so don't penalize anyone who doesn't add them — but some students will figure it out from autocomplete.

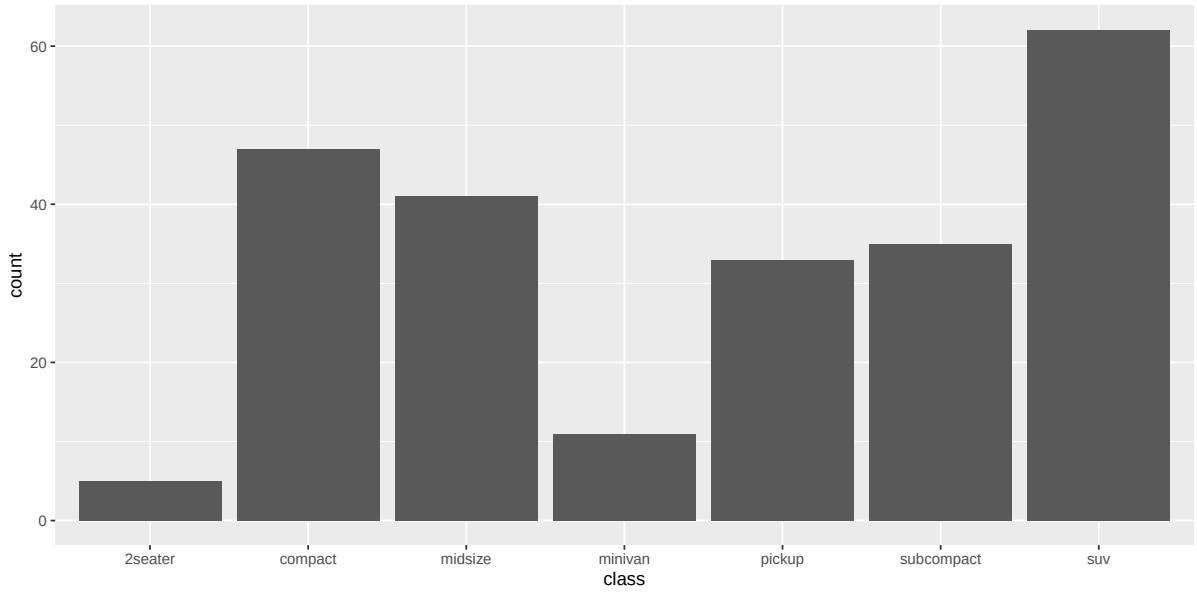
Before we move on

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.

Give students 1 minute to submit. Emphasize that it doesn't need to be perfect or even run — partial code is fine. This is about participation, not correctness. Transition: "Now let's look at some other geom types for different kinds of data."

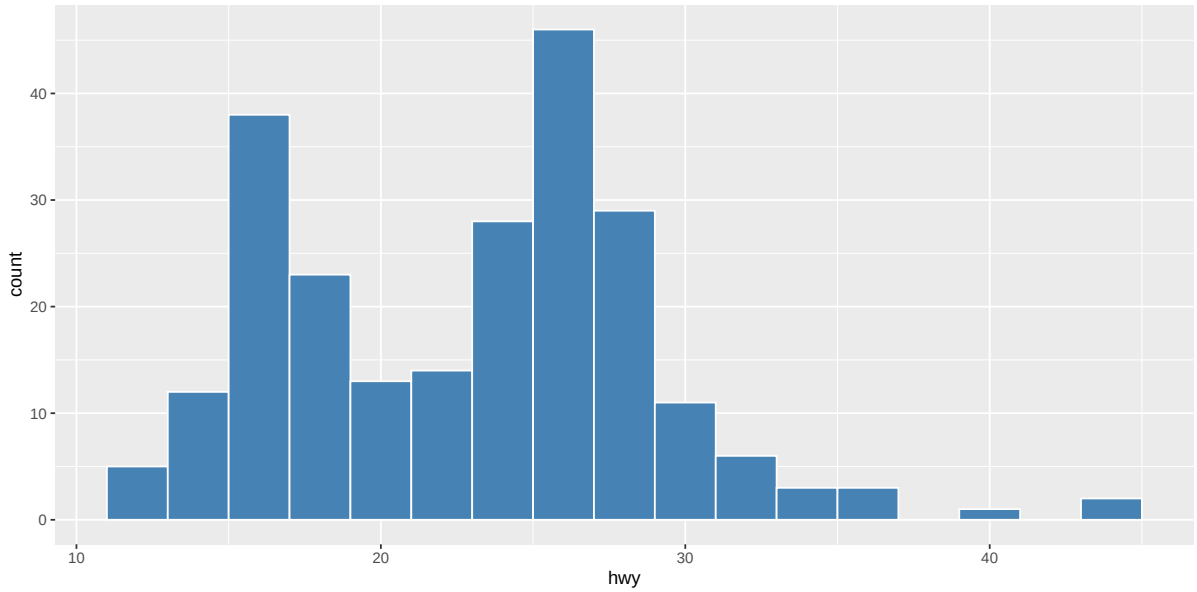
geom_bar() — categorical data

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



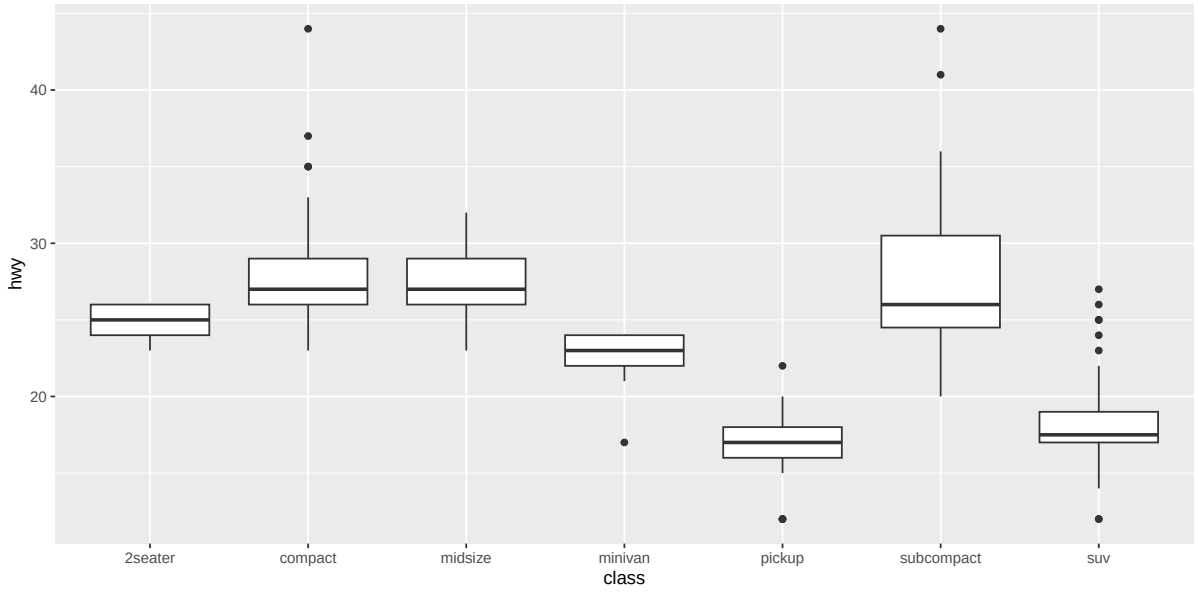
geom_histogram() — distributions

```
# 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()
```



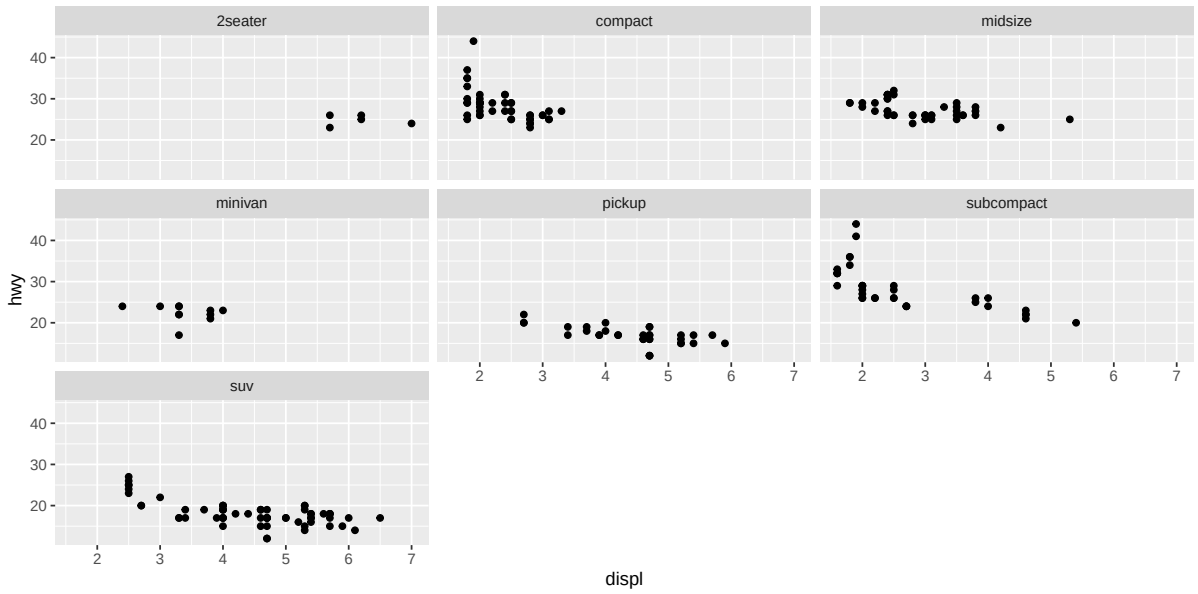
Briefly connect boxplots to psych: "This is exactly the kind of plot you'd use to show group differences — treatment vs. control, different age groups, etc." Students may not know how to read boxplots; if anyone looks confused, do a 30-second refresher on median, IQR, and whiskers. Don't belabor it — they'll get more practice in EDA sessions.

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)
```

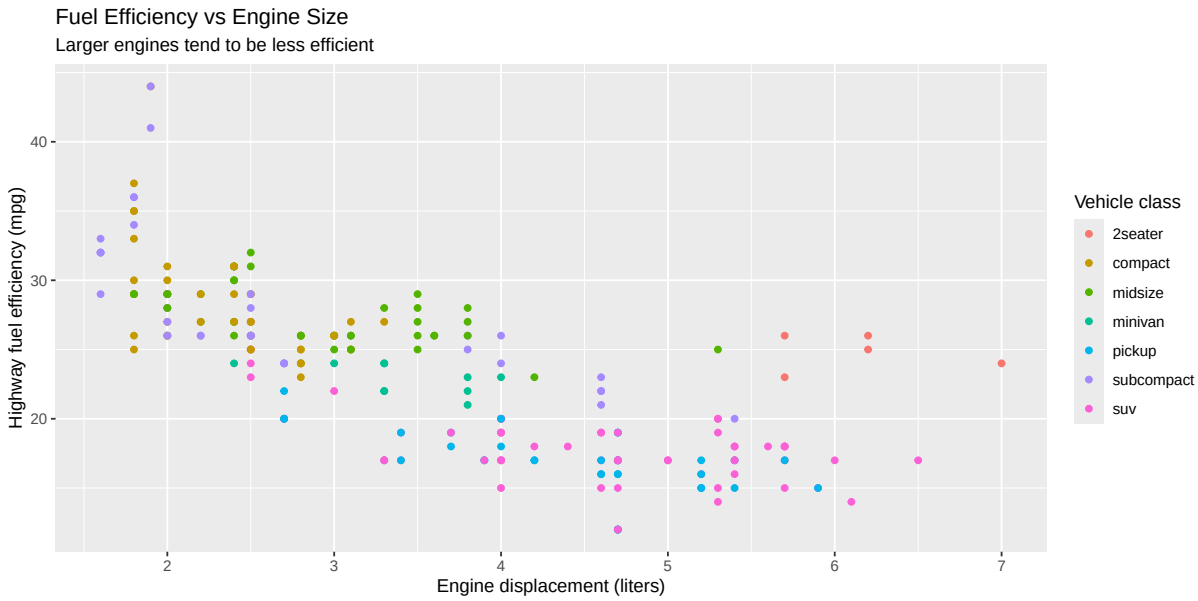


Facets are often the first "wow" moment for students. Point out how much clearer the patterns are when each class gets its own panel vs. the earlier color-coded version. The `~` (tilde) syntax may confuse students — just say "read it as 'by'" for now (facet by class). Mention that `facet_wrap()` is the one they'll use most often — there's also `facet_grid()` for crossing two variables, but they don't need it today. Facets are great when you have too many colors to distinguish or 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"
  )
```

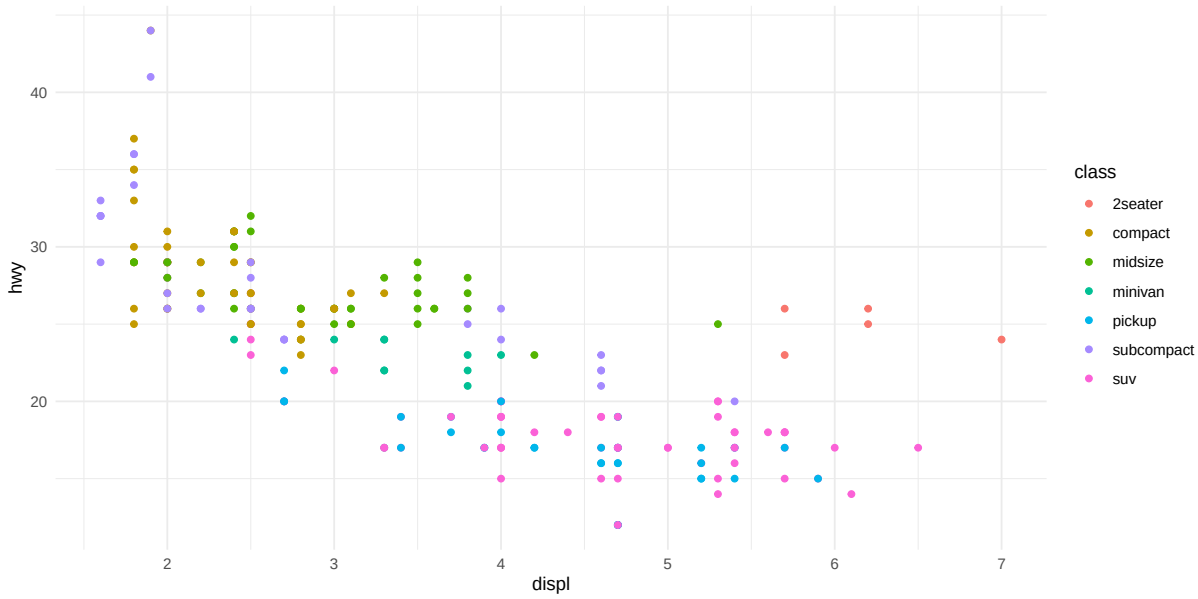


Emphasize that `labs()` is not optional decoration — it's what makes a plot interpretable. Point out the subtitle as a place for the main finding ("Larger engines tend to be less efficient") and the axis labels as a place for units. Relate to psych: "When you present findings in a paper or poster, your reader needs to understand the axes without reading your methods section." Students often forget to relabel the legend — show that `color = "Vehicle class"` changes the legend title.

Themes

Themes control the overall look:

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



Try different ones: `theme_bw()`, `theme_classic()`, `theme_gray()` (default)

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)
```

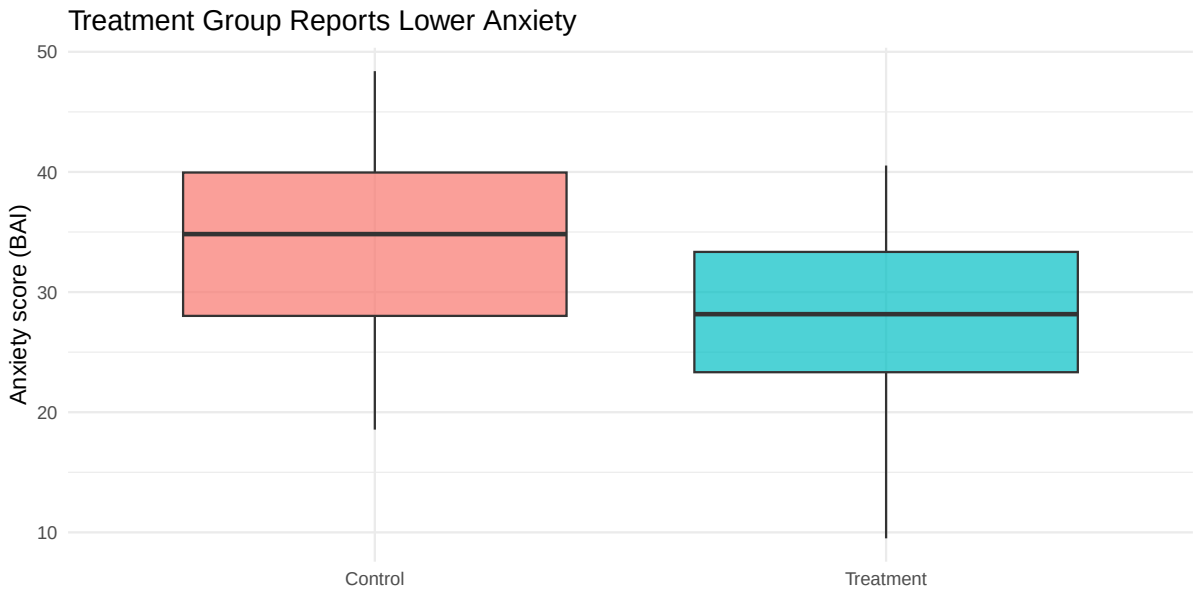
Two things to emphasize: (1) assigning a plot to a variable (`my_plot <-`) so you can save it, and (2) specifying width and height in inches for consistent output. Students will need `ggsave()` for Assignment 1. Common question: "Where does it save?" Answer: in your current working directory (the project folder). They can also give a full path. Mention that `ggsave()` auto-detects the file type from the extension (.png, .pdf, .jpg).

Same template, psychology data

The ggplot template works the same way with any data:

```r
set.seed(410)  # Reproducible results
# 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)
```

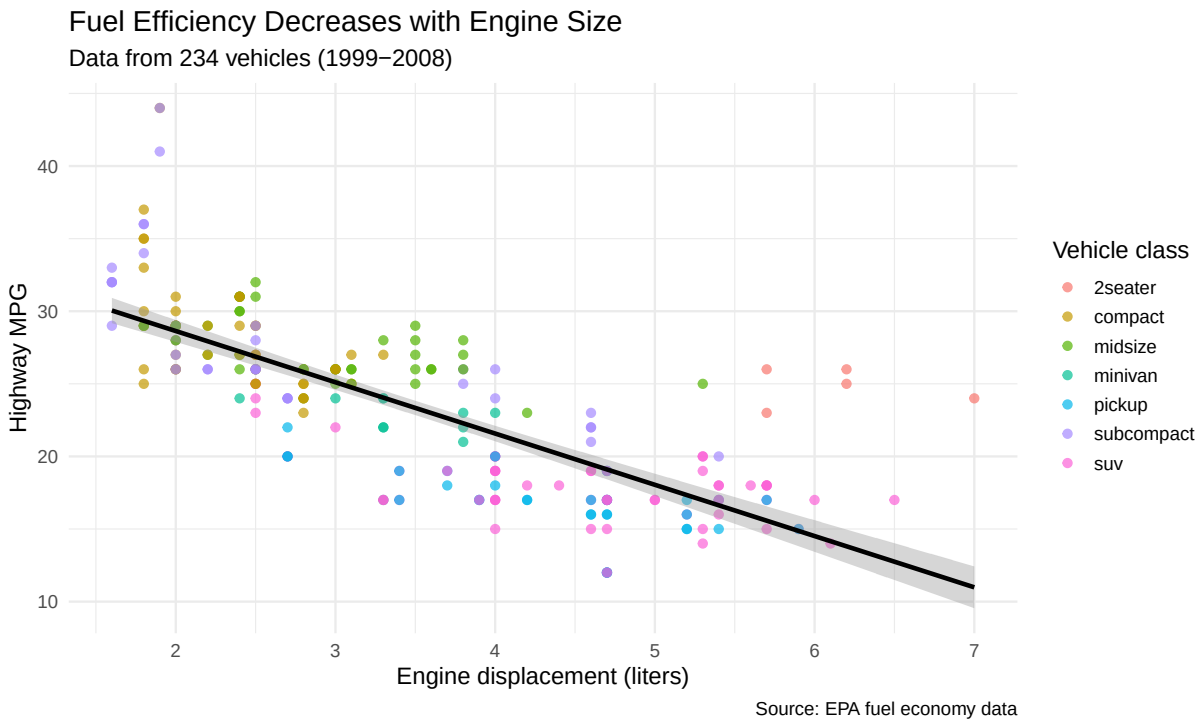


This is the "so what" bridge to psychology. Emphasize that the exact same template they just learned with car data works with any dataset — including their own research data. The simulated anxiety data uses realistic values (BAI scores typically range from 0–63). Ask: "What does this plot tell us about the treatment?" Make the point that even with a small simulated dataset, a boxplot immediately communicates the group difference.

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)
```



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.

Give students ~10 minutes of independent work time here. Walk around and help. This exercise combines everything from the session: the basic template, color mapping, faceting, and saving. Students who finish early can experiment with themes, labels, or different geoms. Reveal the solution only if most students are stuck. Remind them that Assignment 1 is due Sunday at 11:59 PM and includes additional tasks beyond this exercise.

Solution

Wrapping up

Before next class

Read: R4DS Ch 3: Data transformation (sections 3.1–3.4)

Assignment 1 is due Sunday at 11:59 PM

Mention that the reading for next time is about dplyr and data transformation — a different topic from today. They don't need to understand it perfectly before class, but reading it will make the lecture click faster. Remind them that Assignment 1 includes additional tasks beyond the in-class exercise and is due Sunday.

Never trust a summary statistic you haven't plotted

Today's template — you'll use it all quarter:

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

Next time: Data Transformation with dplyr