## A code template

> "The simple graph has brought more information to the data analyst's mind than any other device." — John Tukey

Let's begin with a question to explore.

**What do you think: Do cars with bigger engines use more fuel than cars with smaller engines?**

[✓ Cars with bigger engines use *more* fuel. ✓]

Cars with bigger engines use *less* fuel. ✓

Great!

In other words, there is a positive relationship between engine size and fuel efficiency. Now let's test your hypothesis against data.

### ✓ `mpg`

You can test your hypothesis with the `mpg` dataset that comes in the `ggplot2` package. `mpg` contains observations collected on 38 models of cars by the US Environmental Protection Agency.

To see the `mpg` data frame, type `mpg` in the code block below and click "Submit Answer".

Code | Start Over | Solution | Run Code | Submit Answer

```
mpg
```

| manufacturer <chr> | model <chr> | displ <dbl> | year <int> | cyl <int> | trans <chr> | drv <chr> | cty <int> | hwy <int> |
|---|---|---|---|---|---|---|---|---|
| audi | a4 | 1.8 | 1999 | 4 | auto(l5) | f | 18 | 29 |
| audi | a4 | 1.8 | 1999 | 4 | manual(m5) | f | 21 | 29 |
| audi | a4 | 2.0 | 2008 | 4 | manual(m6) | f | 20 | 31 |
| audi | a4 | 2.0 | 2008 | 4 | auto(av) | f | 21 | 30 |
| audi | a4 | 2.8 | 1999 | 6 | auto(l5) | f | 16 | 26 |
| audi | a4 | 2.8 | 1999 | 6 | manual(m5) | f | 18 | 26 |
| audi | a4 | 3.1 | 2008 | 6 | auto(av) | f | 18 | 27 |
| audi | a4 quattro | 1.8 | 1999 | 4 | manual(m5) | 4 | 18 | 26 |
| audi | a4 quattro | 1.8 | 1999 | 4 | auto(l5) | 4 | 16 | 25 |
| audi | a4 quattro | 2.0 | 2008 | 4 | manual(m6) | 4 | 20 | 28 |

1-10 of 234 rows | 1-9 of 11 columns    Previous 1 2 3 4 5 6 ... 24 Next

"Good job! We'll use interactive code chunks like this throughout these tutorials. Whenever you encounter one, you can click Submit Answer to run (or re-run) the code in the chunk. If there is a Solution button, you can click it to see the answer."

You can use the black triangle that appears at the top right of the table to scroll through all of the columns in `mpg`.

Among the variables in `mpg` are:

1. `displ`, a car's engine size, in liters.
2. `hwy`, a car's fuel efficiency on the highway, in miles per gallon (mpg). A car with a low mpg consumes more fuel than a car with a high mpg when they travel the same distance.

Now let's use this data to make our first graph.

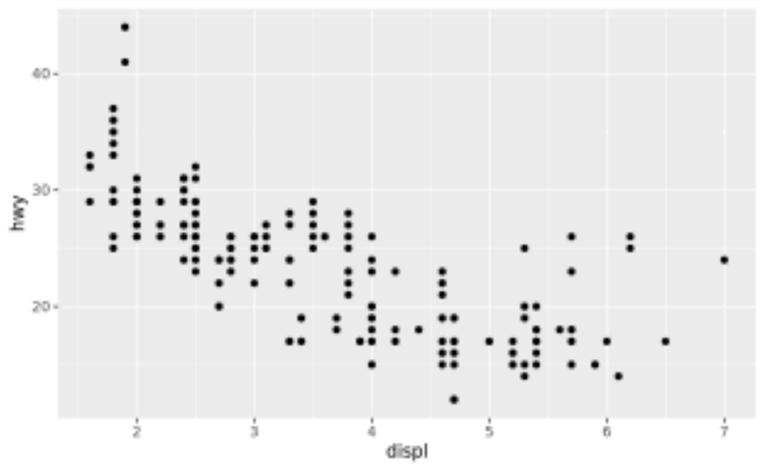
## A plot

The code below uses functions from the **ggplot2** package to plot the relationship between `displ` and `hwy`.

To see the plot, click "Run Code."

Code | Start Over | ▶ Run Code

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))

```



Can you spot the relationship?

## And the answer is...

The plot shows a negative relationship between engine size (`displ`) and fuel efficiency (`hwy`). Points that have a large value of `displ` have a small value of `hwy` and vice versa.

In other words, cars with big engines use *more* fuel. If that was your hypothesis, you were right!

Now let's look at how we made the plot.

## `ggplot()`

Here's the code that we used to make the plot. Notice that it contains three functions: `ggplot()`, `geom_point()`, and `aes()`.

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```

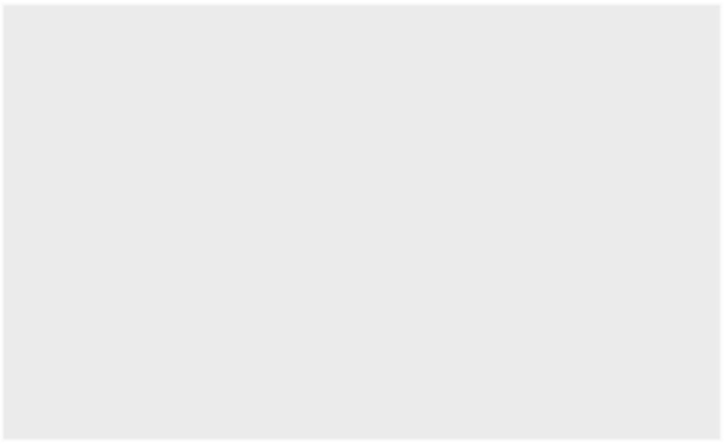
In R, a function is a name followed by a set of parentheses. Many functions require special information to do their jobs, and you write this information between the parentheses.

## ✓ `ggplot`

The first function, `ggplot()`, creates a coordinate system that you can add layers to. The first argument of `ggplot()` is the dataset to use in the graph.

By itself, `ggplot(data = mpg)` creates an empty graph, which looks like this.

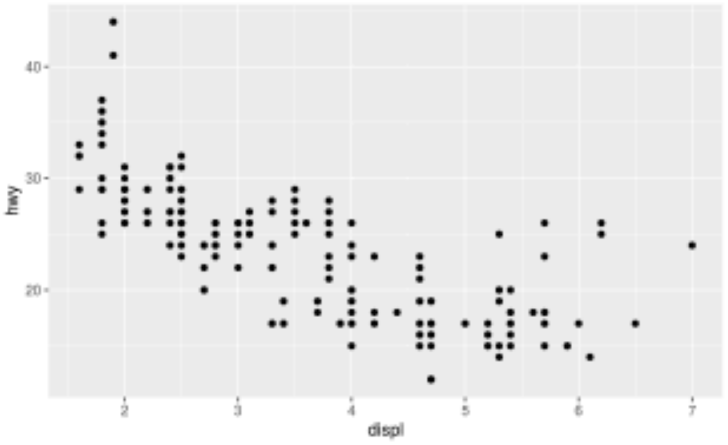
```
ggplot(data = mpg)
```



## ✓ geom_point()

`geom_point()` adds a layer of points to the empty plot created by `ggplot()`. This gives us a scatterplot.

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```



## ✓ mapping = aes()

`geom_point()` takes a `mapping` argument that defines which variables in your dataset are mapped to which axes in your graph. The `mapping` argument is always paired with the function `aes()`, which you use to gather together all of the mappings that you want to create.

Here, we want to map the `displ` variable to the x axis and the `hwy` variable to the y axis, so we add `x = displ` and `y = hwy` inside of `aes()` (and we separate them with a comma).

Where will ggplot2 look for these mapped variables? In the data frame that we passed to the `data` argument, in this case, `mpg`.

## ✓ A graphing workflow

Our code follows the common workflow for making graphs with ggplot2. To make a graph, you:

1. Start the graph with `ggplot()`
2. Add elements to the graph with a `geom_` function
3. Select variables with the `mapping = aes()` argument

## ✓ A graphing template

In fact, you can turn our code into a reusable template for making graphs. To make a graph, replace the bracketed sections in the code below with a data set, a `geom_` function, or a collection of mappings.

Give it a try! Replace the bracketed sections with `mpg`, `geom_boxplot`, and `x = class, y = hwy` to make a slightly different graph. Be sure to delete the `#` symbols before you run the code.

Code   Start Over   Solution   Run Code   Submit Answer

```
# ggplot(data = <DATA>) +
#   <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))

```

"Good job! This plot uses boxplots to compare the fuel efficiencies of different types of cars. ggplot2 comes with many geom functions that each add a different type of layer to a plot. You'll learn more about boxplots and other geoms in the tutorials that follow."

## ✓ Common problems

As you start to run R code, you're likely to run into problems. Don't worry — it happens to everyone. I have been writing R code for years, and every day I still write code that doesn't work!

Start by carefully comparing the code that you're running to the code in the examples. R is extremely picky, and a misplaced character can make all the difference. Make sure that every `(` is matched with a `)` and every `"` is paired with another `"`. Also pay attention to capitalization; R is case sensitive.

## ✓ + location

One common problem when creating ggplot2 graphics is to put the `+` in the wrong place: it has to come at the end of a line, not the start. In other words, make sure you haven't accidentally written code like this:

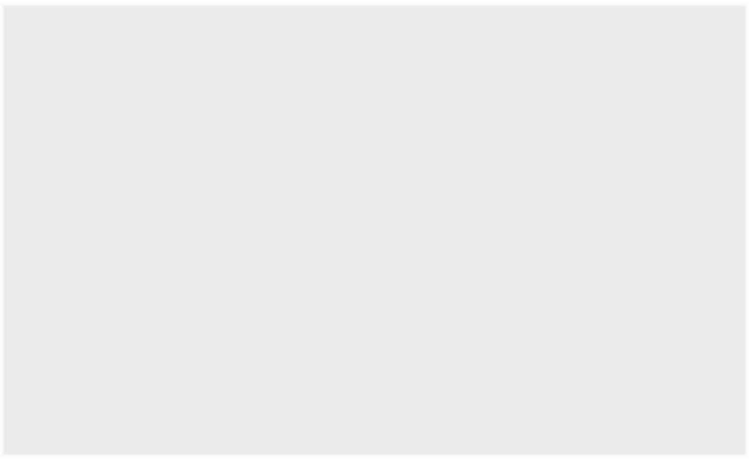
```
ggplot(data = mpg)
+ geom_point(mapping = aes(x = displ, y = hwy))
```

## ✓ Exercise 1

Run `ggplot(data = mpg)` what do you see?

Code | Start Over | Run Code | Submit Answer

```
ggplot(data=mpg)
```



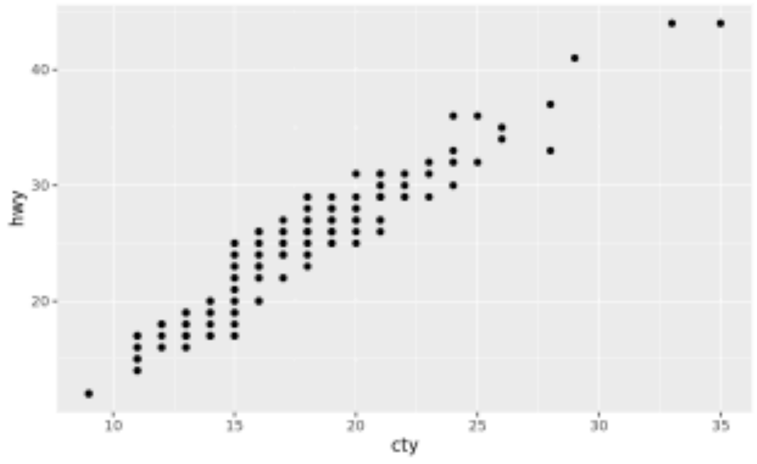
"Good job! A ggplot that has no layers looks blank. To finish the graph, add a geom function."

## ✓ Exercise 2

Make a scatterplot of `cty` vs `hwy`.

Code | Start Over | Hint | Run Code | Submit Answer

```
ggplot(data = mpg, aes(x=cty, y=hwy)) + geom_point()
```



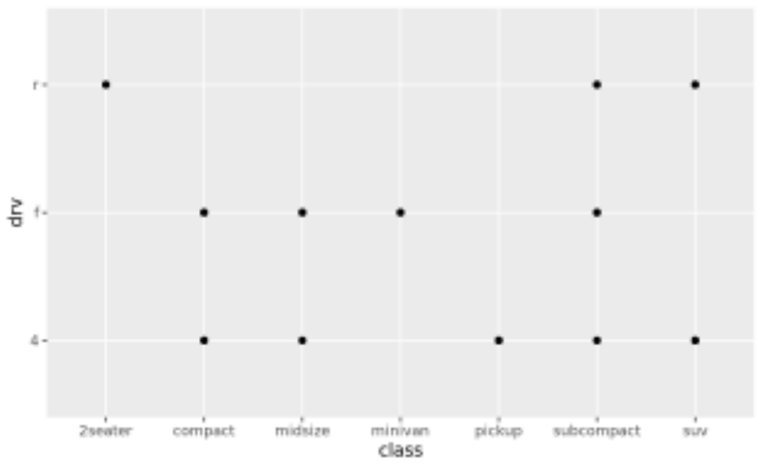
"Excellent work!"

## Exercise 3

What happens if you make a scatterplot of `class` vs `drv`. Try it. Why is the plot not useful?

Code  Start Over  Run Code  Submit Answer

```
ggplot(data = mpg, aes(x = class, y = drv)) + geom_point()
```



"Nice job! `class` and `drv` are both categorical variables. As a result, points can only appear at certain values, where many points overlap each other. You have no idea how many points fall on top of each other at each location. Experiment with geom_count() to find a better solution."

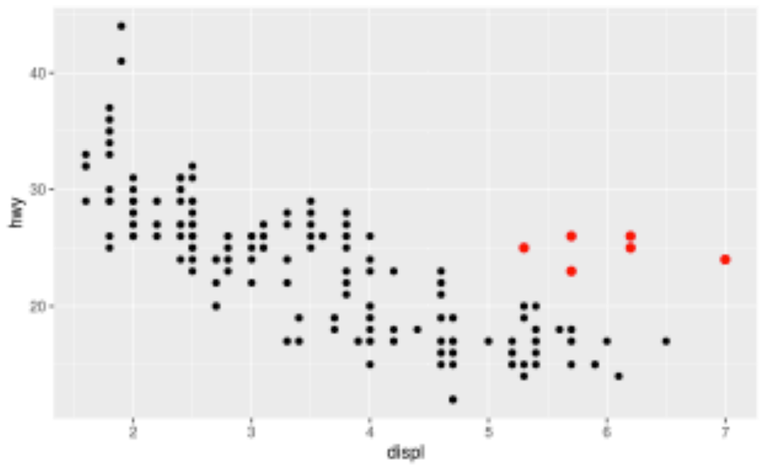
Previous Topic  Next Topic

## Aesthetic mappings

"The greatest value of a picture is when it forces us to notice what we never expected to see." — John Tukey

### ✓ A closer look

In the plot below, one group of points (highlighted in red) seems to fall outside of the linear trend between engine size and gas mileage. These cars have a higher mileage than you might expect. How can you explain these cars?



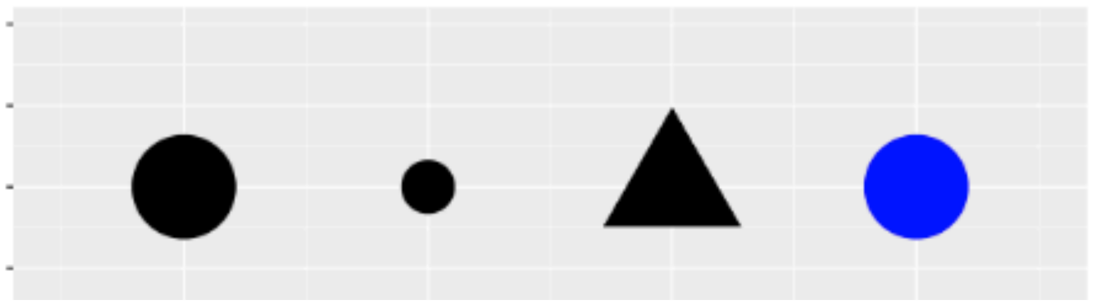
### ✓ A hypothesis

Let's hypothesize that the cars are hybrids. One way to test this hypothesis is to look at the `class` value for each car. The `class` variable of the `mpg` dataset classifies cars into groups such as compact, midsize, and SUV. If the outlying points are hybrids, they should be classified as compact cars or, perhaps, subcompact cars (keep in mind that this data was collected before hybrid trucks and SUVs became popular). To check this, we need to add the `class` variable to the plot.

### ✓ Aesthetics

You can add a third variable, like `class`, to a two dimensional scatterplot by mapping it to a new **aesthetic**. An aesthetic is a visual property of the objects in your plot. Aesthetics include things like the size, the shape, or the color of your points.

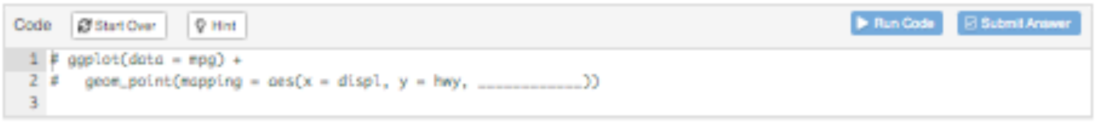
You can display a point (like the one below) in different ways by changing the values of its aesthetic properties. Since we already use the word "value" to describe data, let's use the word "level" to describe aesthetic properties. Here we change the levels of a point's size, shape, and color to make the point small, triangular, or blue:



## ✓ A strategy

We can add the `class` variable to the plot by mapping the levels of an aesthetic (like color) to the values of `class`. For example, we can color a point green if it belongs to the compact class, blue if it belongs to the midsize class, and so on.

Let's give this a try. Fill in the blank piece of code below with `color = class`. What happens? Delete the commenting symbols (`#`) before running your code. (If you prefer British English, you can use `colour` instead of `color`.)



```
# ggplot(data = mpg) +
#   geom_point(mapping = aes(x = displ, y = hwy, ____________))

```

"Great Job! You can now tell which class of car each point represents by examining the color of the point."

## ✓ And the answer is...

The colors reveal that many of the unusual points in `mpg` are two-seater cars. These cars don't seem like hybrids, and are, in fact, sports cars! Sports cars have large engines like SUVs and pickup trucks, but small bodies like midsize and compact cars, which improves their gas mileage. In hindsight, these cars were unlikely to be hybrids since they have large engines.

This isn't the only insight we've gleaned; you've also learned how to add new aesthetics to your graph. Let's review the process.

## ✓ Aesthetic mappings

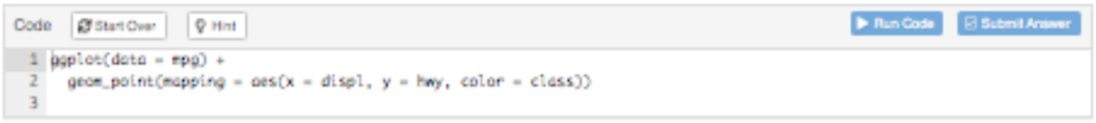
To map an aesthetic to a variable, set the name of the aesthetic equal to the name of the variable, and do this inside `mapping = aes()`. ggplot2 will automatically assign a unique level of the aesthetic (here a unique color) to each unique value of the variable. ggplot2 will also add a legend that explains which levels correspond to which values.

This insight gives us a new way to think about the mapping argument. Mappings tell ggplot2 more than which variables to put on which axes, they tell ggplot2 which variables to map to which visual properties. The x and y locations of each point are just two of the many visual properties displayed by a point.

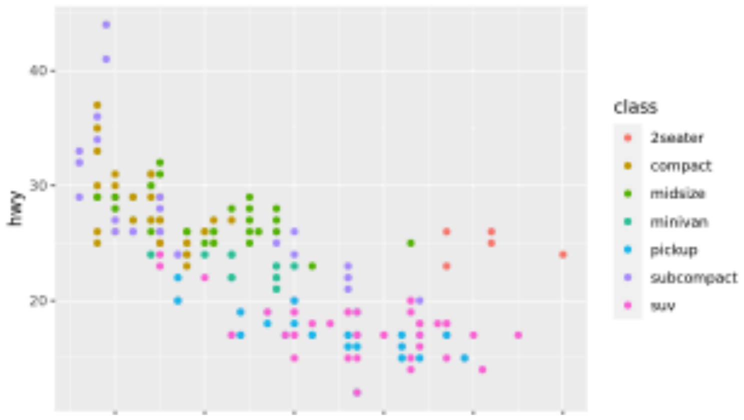
## ✓ Other aesthetics

In the above example, we mapped color to `class`, but we could have mapped size to `class` in the same way.

Change the code below to map `size` to `class`. What happens?



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

```



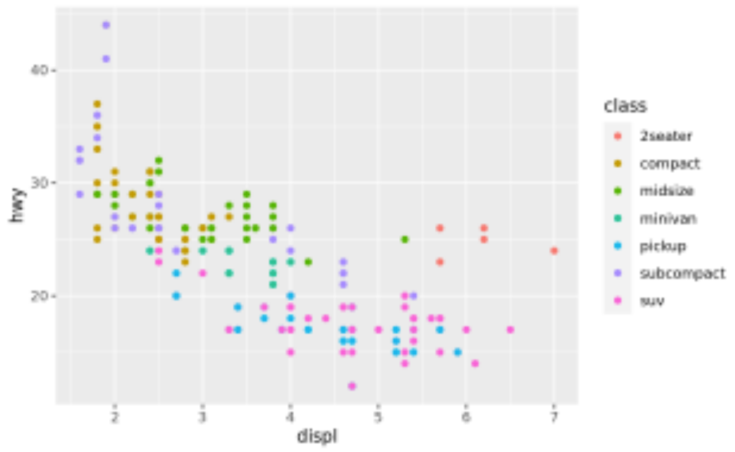
## ✓ Other aesthetics

In the above example, we mapped color to `class`, but we could have mapped size to `class` in the same way.

Change the code below to map `size` to `class`. What happens?

Code | Start Over | Hint | ▶ Run Code | Submit Answer

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

```



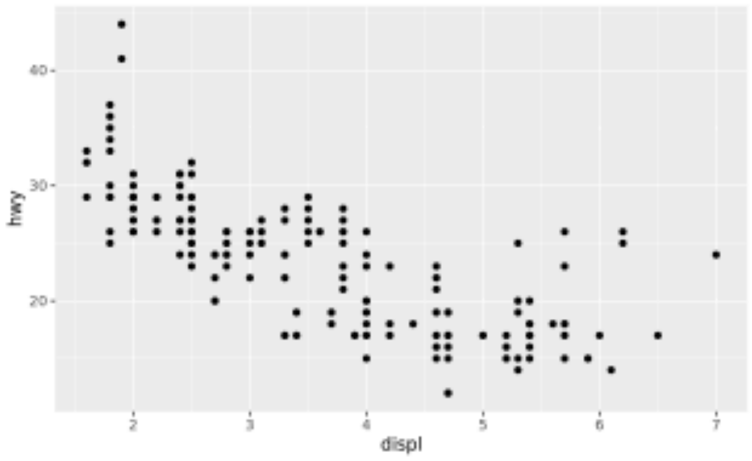
"Great Job! Now the size of a point represents its class. Did you notice the warning message? ggplot2 gives us a warning here because mapping an unordered variable (class) to an ordered aesthetic (size) is not a good idea."

## ✓ alpha

You can also map `class` to the `alpha` aesthetic, which controls the transparency of the points. Try it below.

Code | Start Over | Hint | Run Code | Submit Answer

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))

```



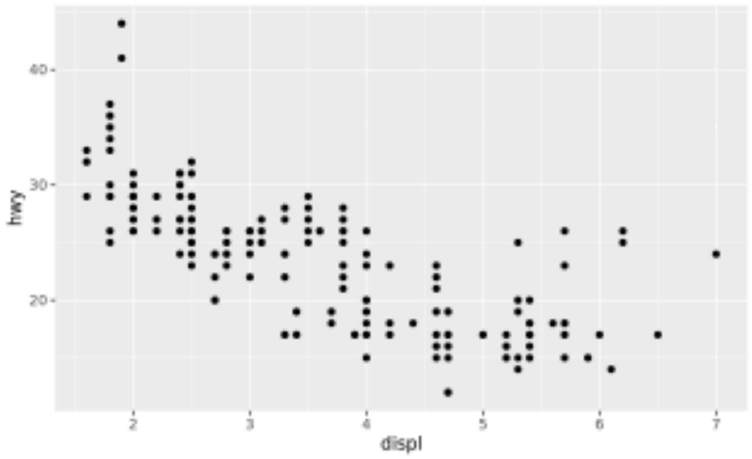
"Great Job! If you look closely, you can spot something subtle: many locations contain multiple points stacked on top of each other (alpha is additive so multiple transparent points will appear opaque)."

## ✓ Shape

Let's try one more aesthetic. This time map the class of the points to `shape`, then look for the SUVs. What happened?

Code | Start Over | Hint | Run Code | Submit Answer

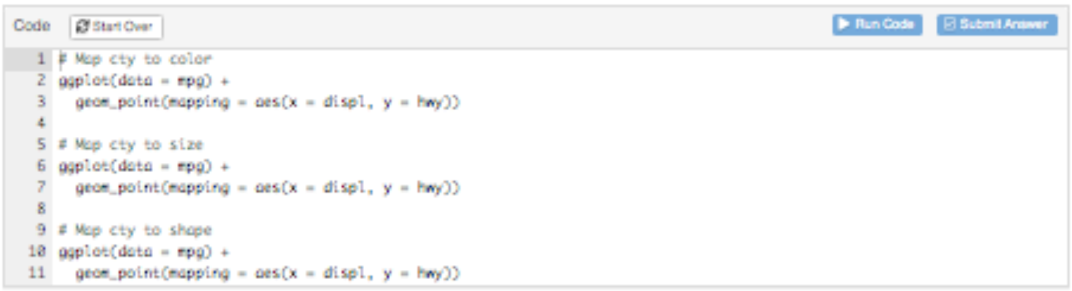
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))

```



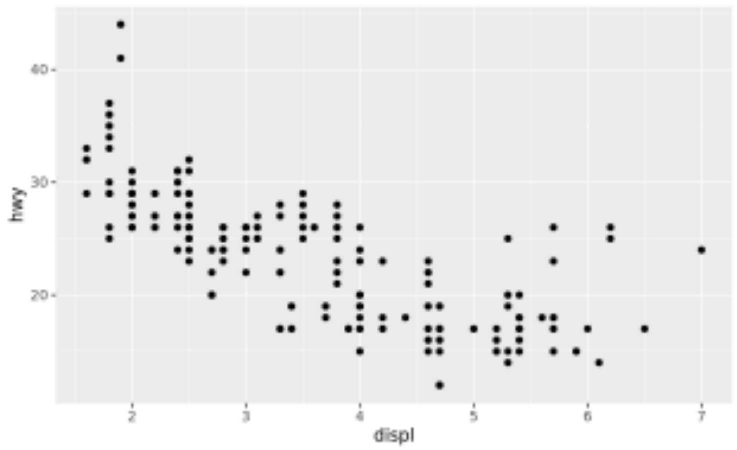
"Good work! What happened to the SUVs? ggplot2 will only use six shapes at a time. By default, additional groups will go unplotted when you use the shape aesthetic. So only use it when you have fewer than seven groups."

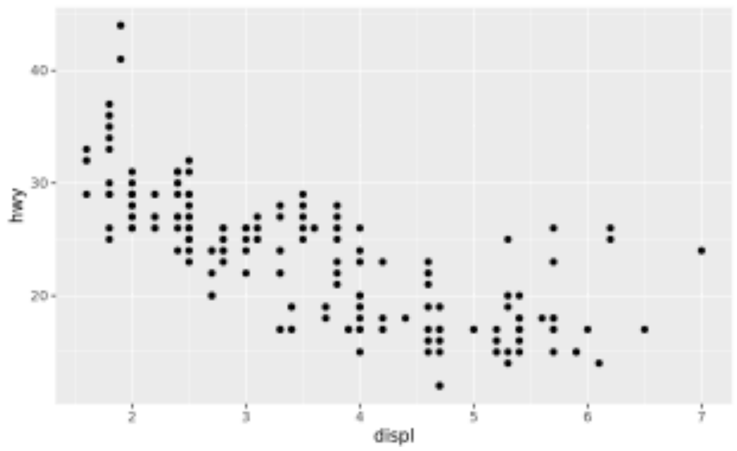
## ✓ Exercise 1

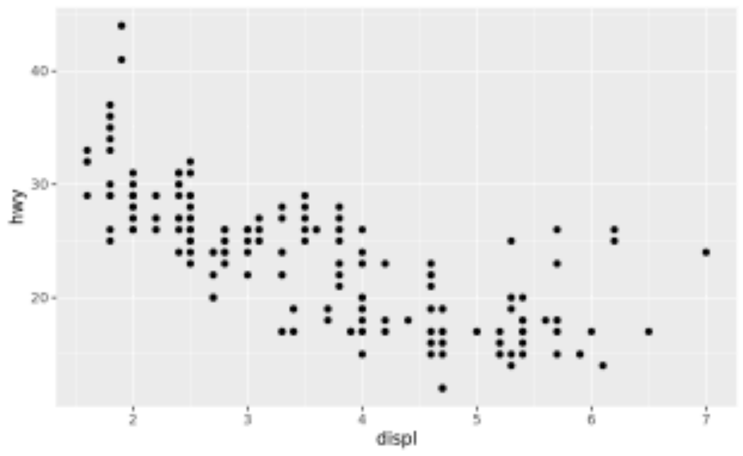
In the code below, map `cty`, which is a continuous variable, to `color`, `size`, and `shape`. How do these aesthetics behave differently for continuous variables, like `cty`, vs. categorical variables, like `class`?



```r
# Map cty to color
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))

# Map cty to size
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))

# Map cty to shape
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```





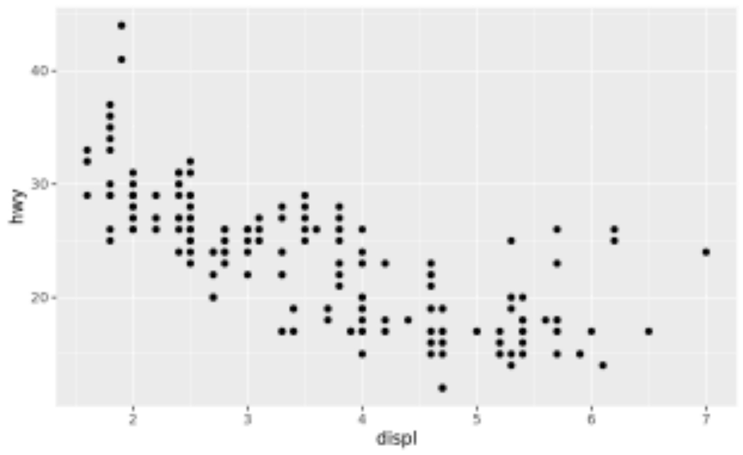


"Very nice! ggplot2 treats continuous and categorical variables differently. Noteably, ggplot2 supplies a blue gradient when you map a continuous variable to color, and ggplot2 will not map continuous variables to shape."

## ✓ Exercise 2

Map `class` to `color`, `size`, and `shape` all in the same plot. Does it work?

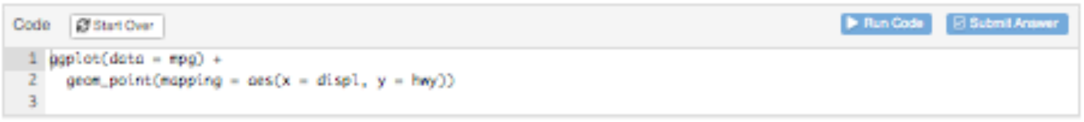
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```



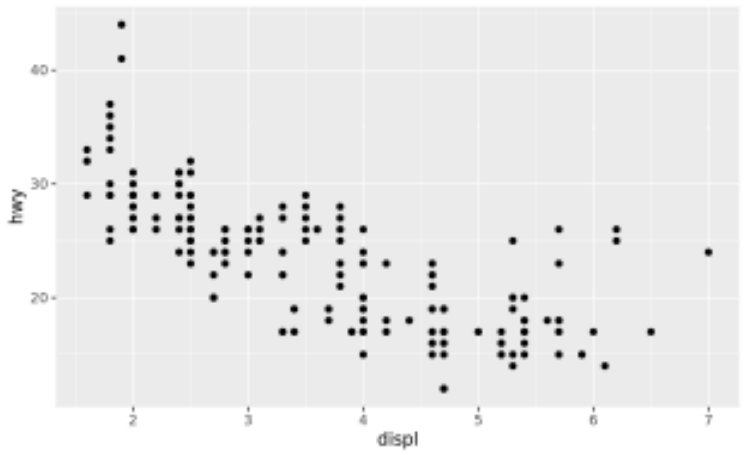
"Very nice! ggplot2 can map the same variable to multiple aesthetics."

## ✓ Exercise 3

What happens if you map an aesthetic to something other than a variable name, like `aes(colour = displ < 5)`? Try it.



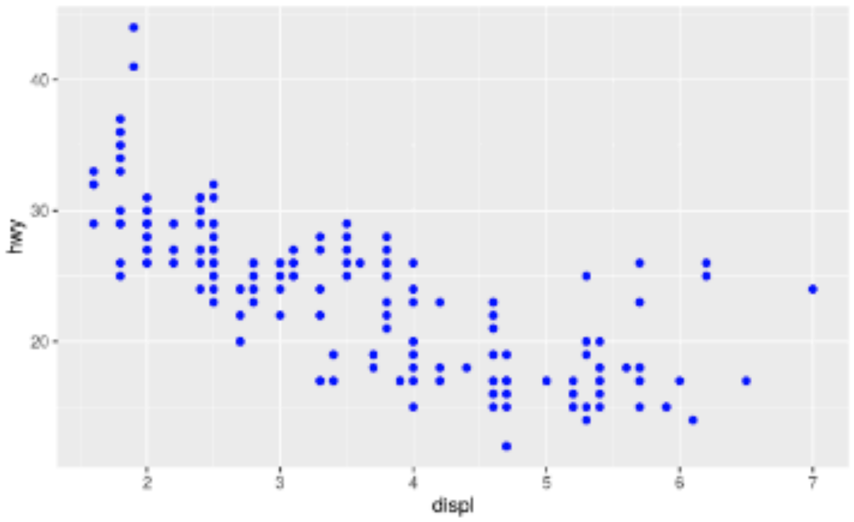
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))

```



"Good job! ggplot2 will map the aesthetic to the results of the expression. Here, ggplot2 mapped the color of each point to TRUE or FALSE based on whther the point's `displ` value was less than five."

## ✓ Setting aesthetics

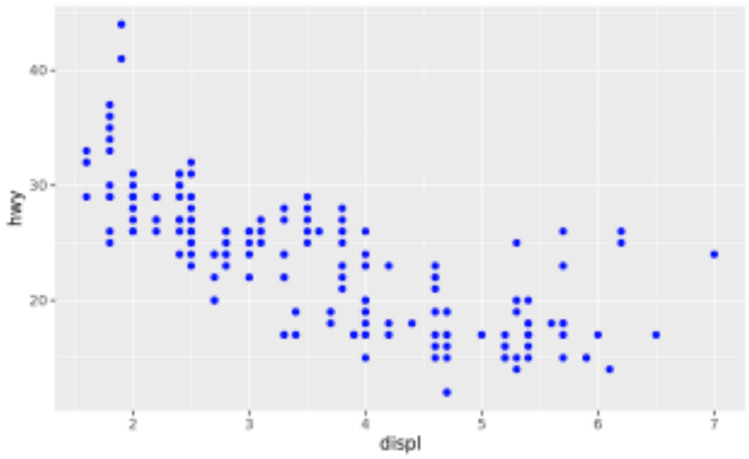
What if you just want to make all of the points in your plot blue, like in the plot below?



You can do this by setting the color aesthetic *outside* of the `aes()` function, like this

Code Start Over Run Code

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

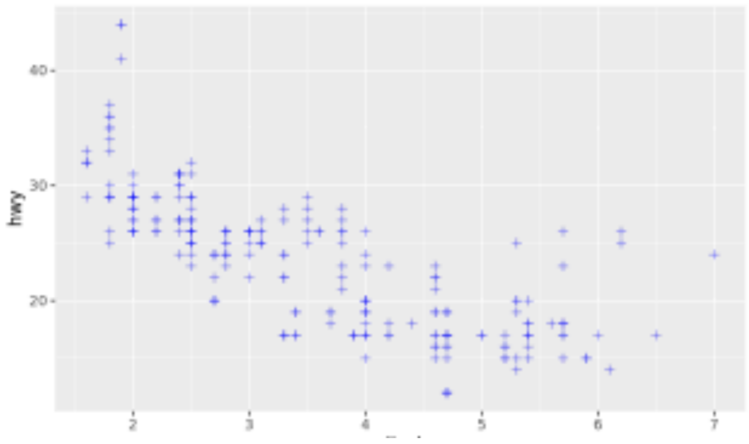


## ✓ Setting vs. Mapping

Setting works for every aesthetic in ggplot2. If you want to manually set the aesthetic to a **value** in the visual space, set the aesthetic outside of `aes()`.

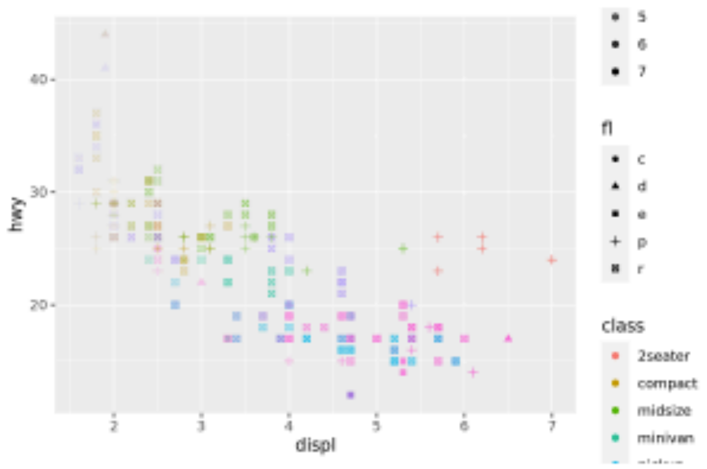
Code Start Over Run Code

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy), color = "blue", shape = 3, alpha = 0.5)
```



If you want to map the aesthetic to a **variable** in the data space, map the aesthetic inside `aes()`.

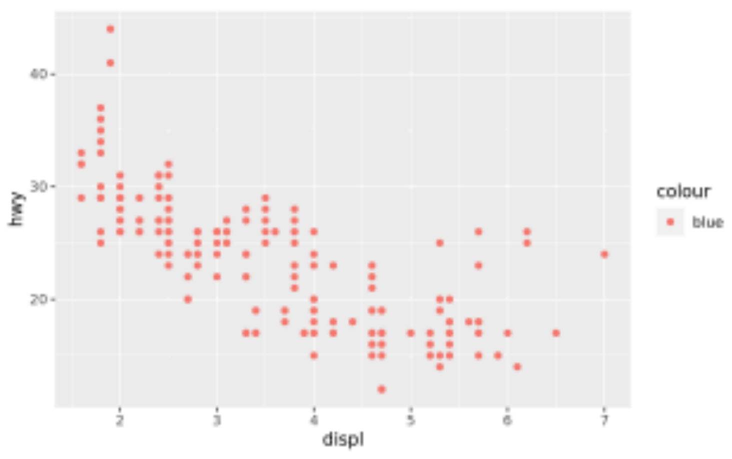
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = class, shape = fl, alpha = displ))
```



## ✓ Exercise 4

What do you think went wrong in the code below? Fix the code so it does something sensible.

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

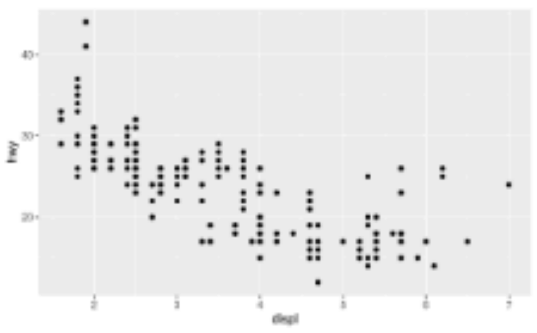


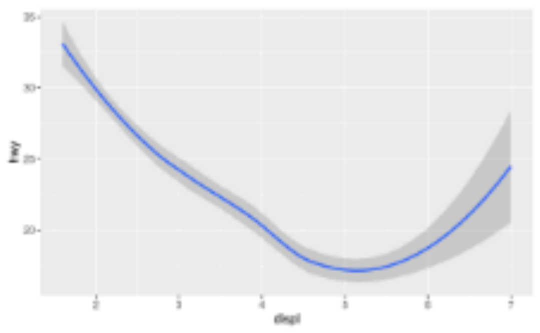
"Good job! Putting an aesthetic in the wrong location is one of the most common graphing errors. Sometimes it helps to think of legends. If you will need a legend to understand what the color/shape/etc. means, then you should probably put the aesthetic inside `aes()` --- ggplot2 will build a legend for every aesthetic mapped here. If the aesthetic has no meaning and is just... well, aesthetic, then set it outside of `aes()`."

## Geometric objects

### ✓ Geoms

How are these two plots similar?





Both plots contain the same x variable, the same y variable, and both describe the same data. But the plots are not identical. Each plot uses a different visual object to represent the data. In ggplot2 syntax, we say that they use different **geoms**.

A **geom** is the geometrical object that a plot uses to represent observations. People often describe plots by the type of geom that the plot uses. For example, bar charts use bar geoms, line charts use line geoms, boxplots use boxplot geoms, and so on. Scatterplots break the trend; they use the point geom.

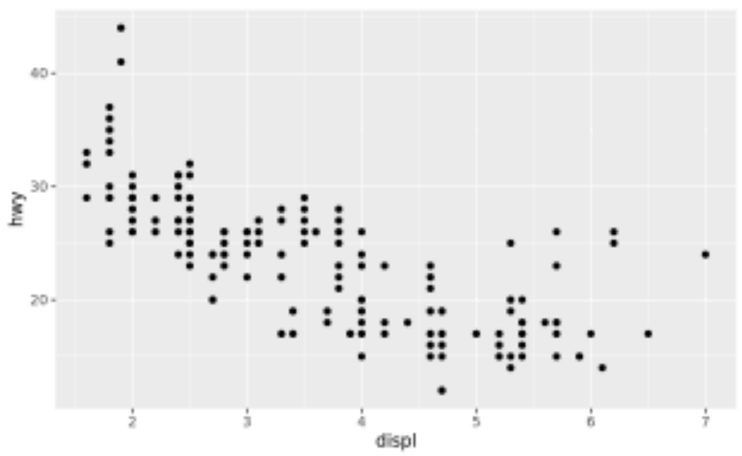
As we see above, you can use different geoms to plot the same data. The plot on the left uses the point geom, and the plot on the right uses the smooth geom, a smooth line fitted to the data.

### ✓ Geom functions

To change the geom in your plot, change the geom function that you add to `ggplot()`. For example, take this code which makes the plot on the left (above), and change `geom_point()` to `geom_smooth()`. What do you get?

Code | Start Over | Solution | Run Code | Submit Answer

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))

```



"Good job! You get the plot on the right (above)."

## More about geoms

ggplot2 provides over 30 geom functions that you can use to make plots, and extension packages provide even more (see https://exts.ggplot2.tidyverse.org/gallery/ for a sampling). You'll learn how to use these geoms to explore data in the Visualize Data primer.

Until then, the best way to get a comprehensive overview of the available geoms is with the ggplot2 cheatsheet. To learn more about any single geom, look at its help page, e.g. `?geom_smooth`.

## Exercise 1

What geom would you use to draw a line chart? A boxplot? A histogram? An area chart?

## Exercise 2

**What does the `se` argument to `geom_smooth()` do?**

- Nothing. `se` is not an argument of `geom_smooth()` ✗
- chooses a method for calculating the smooth line ✗
- controls whether or not to show errors ✗
- [✓ Adds or removes a standard error ribbon around the smooth line ✓]

Correct!