

01-012

Alex Cookson

2020-07-10

```
library(tidyverse)
```

Question 1

What's gone wrong with this code? Why are the points not blue?

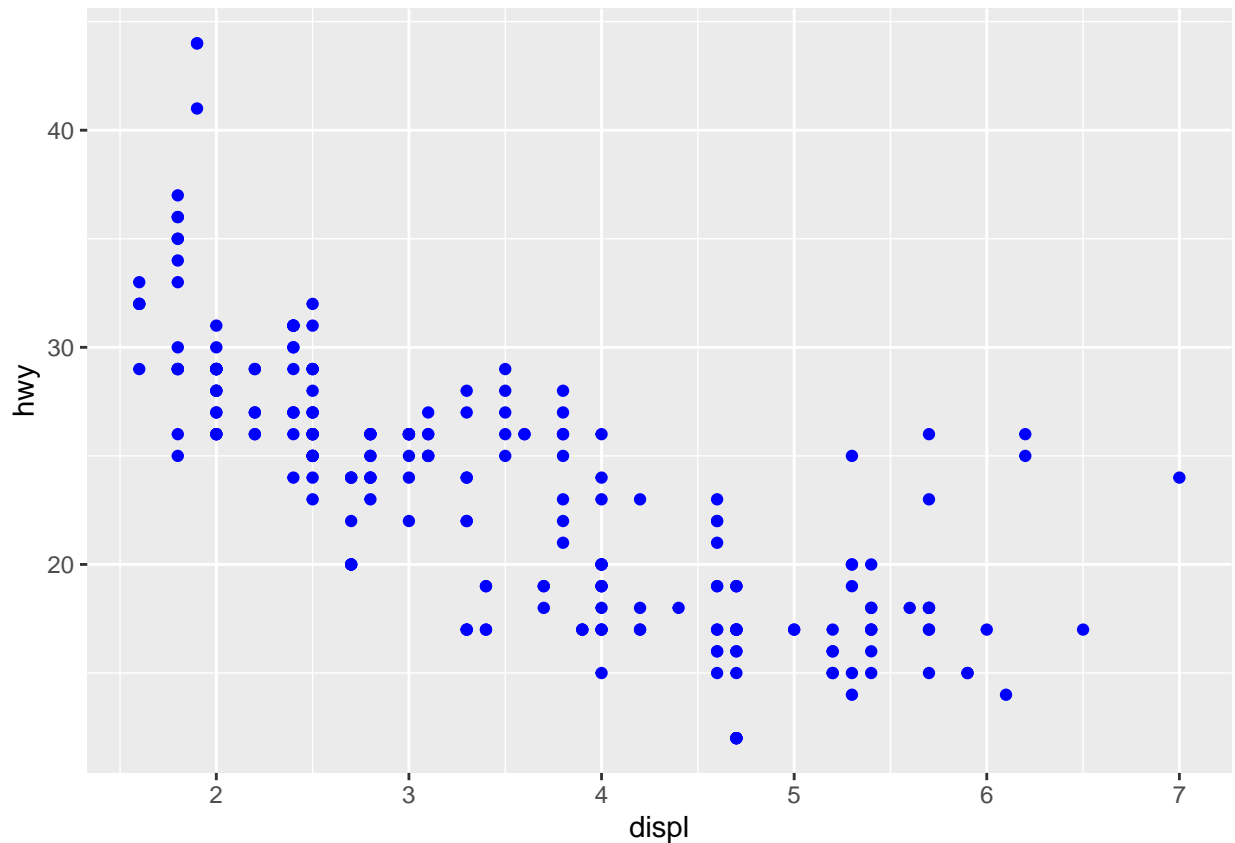
```
ggplot(data = mpg) +  
  geom_point(  
    mapping = aes(x = displ, y = hwy, colour = "blue")  
  )
```

First, we should deduce what we are trying to do with this code: create a scatterplot of `displ` vs. `hwy`, with points manually set to “blue”.

The points are blue, though, because `colour = "blue"` is *inside* the `aes()` call. Being inside `aes()` sets the colour aesthetic to a *character* value of “blue”, which applies to all data. **ggplot** doesn't interpret this as setting a colour, so it uses the default red colour.

The correct code would set `colour = "blue"` *outside* the `aes()`, which we need to do when manually specifying an aesthetic:

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



Question 2

Which variables in `mpg` are categorical? Which variables are continuous? (Hint: type `?mpg` to read the documentation for the dataset.) How can you see this information when you run `mpg`?

Reading the documentation by running `?mpg` gives us enough information to determine which variables are categorical and continuous:

```
?mpg
```

```
## starting httpd help server ... done
```

- `manufacturer`: categorical
- `model`: categorical
- `displ`: continuous
- `year`: continuous
- `cyl`: continuous
- `trans`: categorical
- `drv`: categorical
- `cty`: continuous
- `hwy`: continuous
- `fl`: categorical

- `class`: categorical

We could also run `mpg` and look at the class of the variable – character, double, integer, etc. Between the class of the variable and the first 10 observations displayed, you should be able to figure out which variables are categorical or continuous

`mpg`

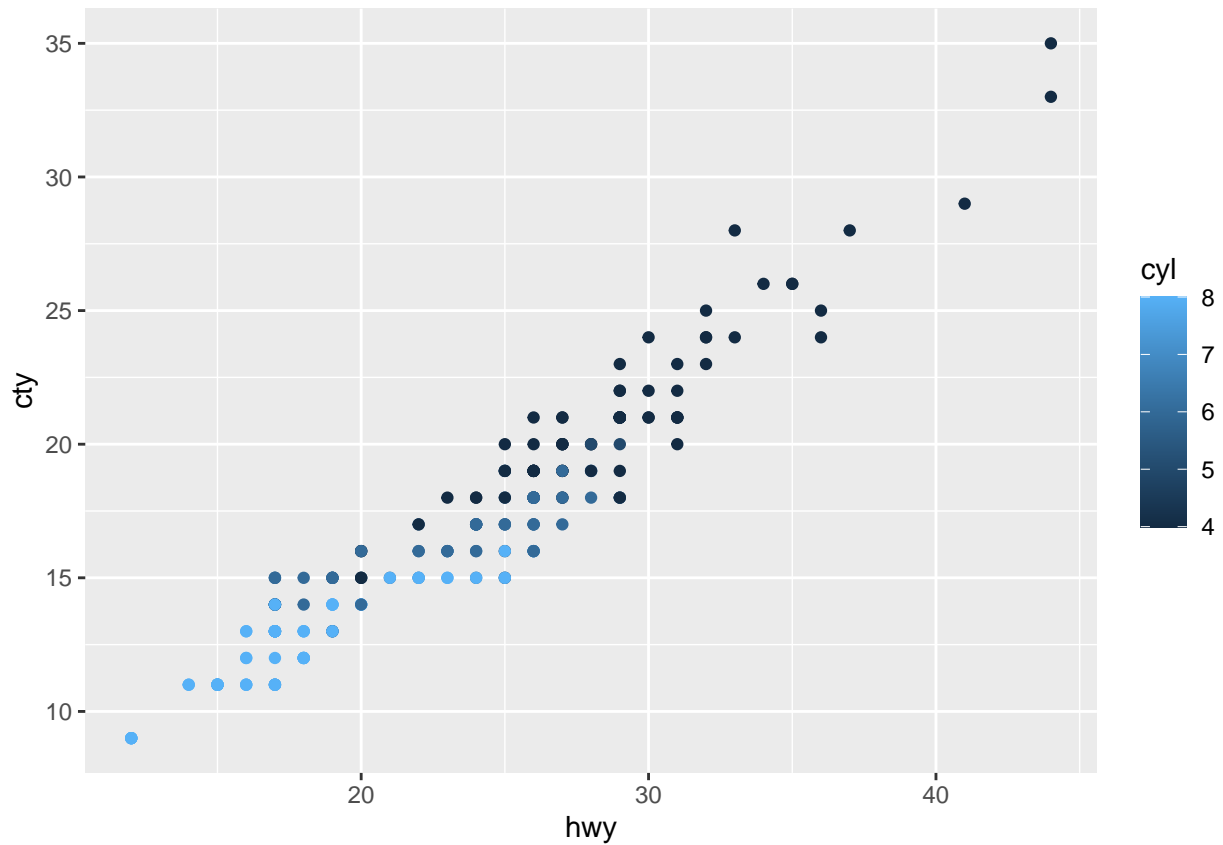
```
## # A tibble: 234 x 11
##   manufacturer model   displ  year   cyl trans  drv      cty   hwy fl      class
##   <chr>          <chr>   <dbl> <int> <int> <chr>  <chr> <int> <int> <chr> <chr>
## 1 audi          a4       1.8  1999     4 auto(l~ f      18    29 p      comp~
## 2 audi          a4       1.8  1999     4 manual~ f      21    29 p      comp~
## 3 audi          a4       2    2008     4 manual~ f      20    31 p      comp~
## 4 audi          a4       2    2008     4 auto(a~ f      21    30 p      comp~
## 5 audi          a4       2.8  1999     6 auto(l~ f      16    26 p      comp~
## 6 audi          a4       2.8  1999     6 manual~ f      18    26 p      comp~
## 7 audi          a4       3.1  2008     6 auto(a~ f      18    27 p      comp~
## 8 audi          a4 quat~ 1.8  1999     4 manual~ 4      18    26 p      comp~
## 9 audi          a4 quat~ 1.8  1999     4 auto(l~ 4      16    25 p      comp~
## 10 audi         a4 quat~ 2    2008     4 manual~ 4      20    28 p      comp~
## # ... with 224 more rows
```

Question 3

Map a continuous variable to `colour`, `size`, and `shape`. How do these aesthetics behave differently for categorical versus continuous variables?

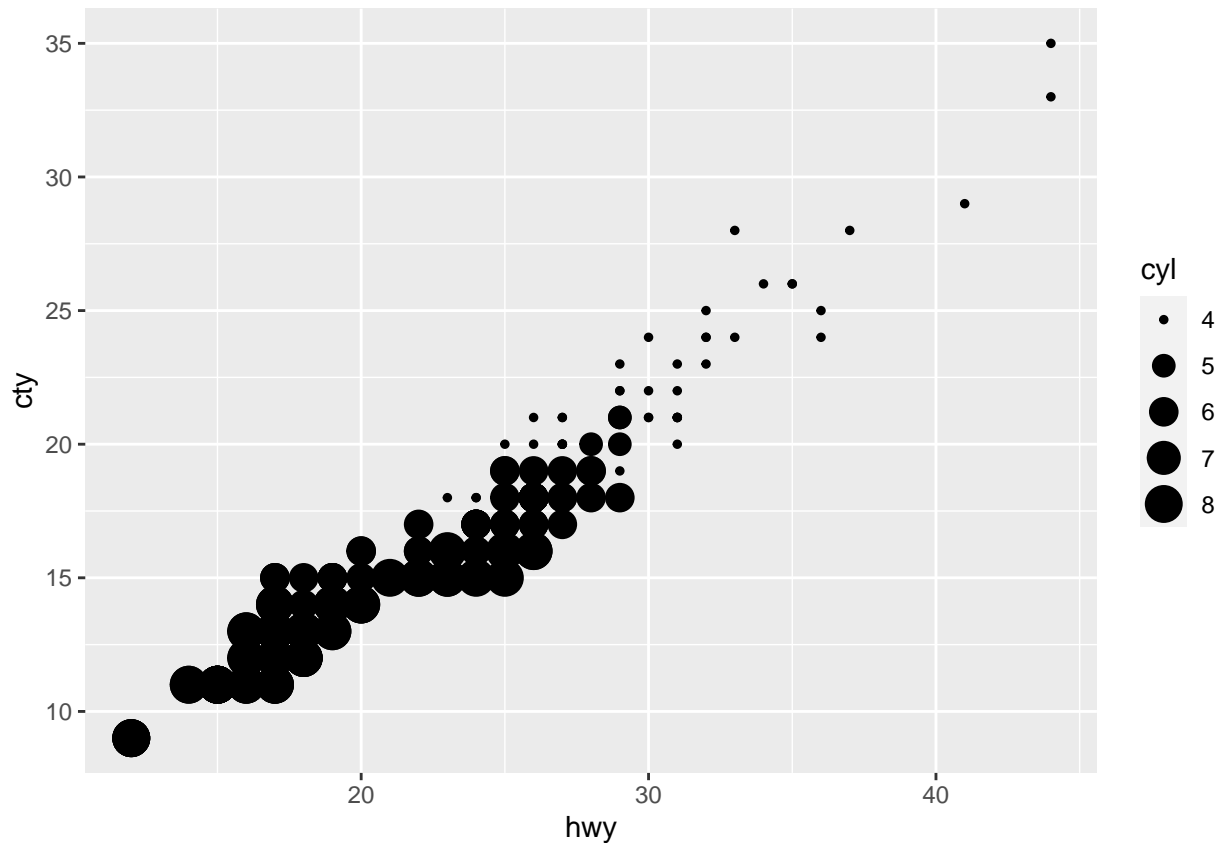
Let's create a scatterplot of `hwy` vs. `cty` mileage, with `cyl`, numbers of cylinders, mapped to the aesthetics in the question.

```
# colour
ggplot(data = mpg) +
  geom_point(
    mapping = aes(x = hwy, y = cty, colour = cyl)
  )
```



When mapped to colour, `cyl` values are on a gradient, with high values as light blue and low values as dark blue.

```
# size
ggplot(data = mpg) +
  geom_point(
    mapping = aes(x = hwy, y = cty, size = cyl)
  )
```



When mapped to **size**, high **cyl** values have large points and low **cyl** values have small points.

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

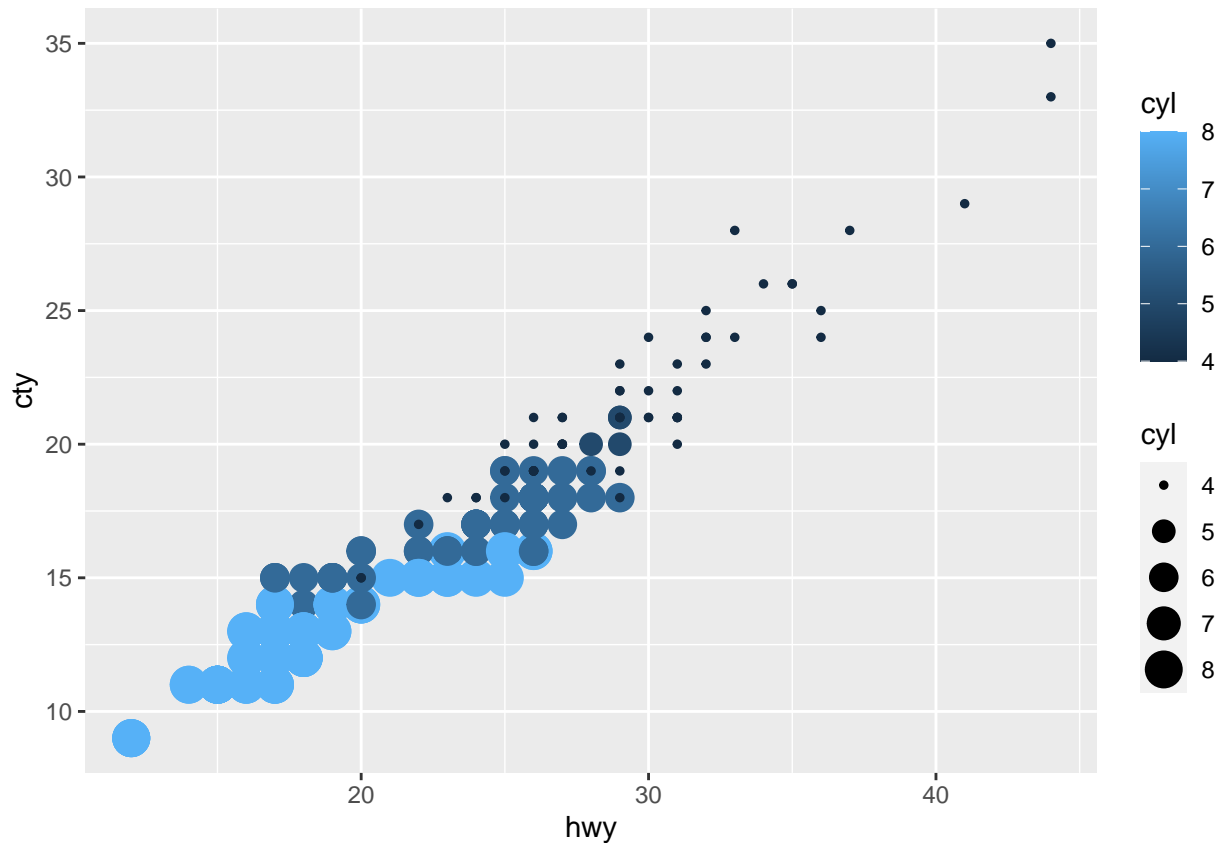
When mapped to **shape**, we get an error message and nothing is graphed. Shapes don't have a natural ordering to them in the same way **colour** and **size** do, so **ggplot** doesn't know what to do with it.

Question 4

What happens if you map the same variable to multiple aesthetics?

Let's find out. We'll use the previous question as inspiration and plot **hwy** vs. **cty**, but this time we will map **cyl** to the **colour** and **size** aesthetics.

```
ggplot(data = mpg) +
  geom_point(
    mapping = aes(x = hwy, y = cty, colour = cyl, size = cyl)
  )
```



cyl has taken on elements of each aesthetic: cars with many cylinders have large (**size** aesthetic), light blue (**colour** aesthetic) points, while cars with few cylinders have small, dark blue points

Question 5

What does the **stroke** aesthetic do? What shapes does it work with? (Hint: use `?geom_point.`)

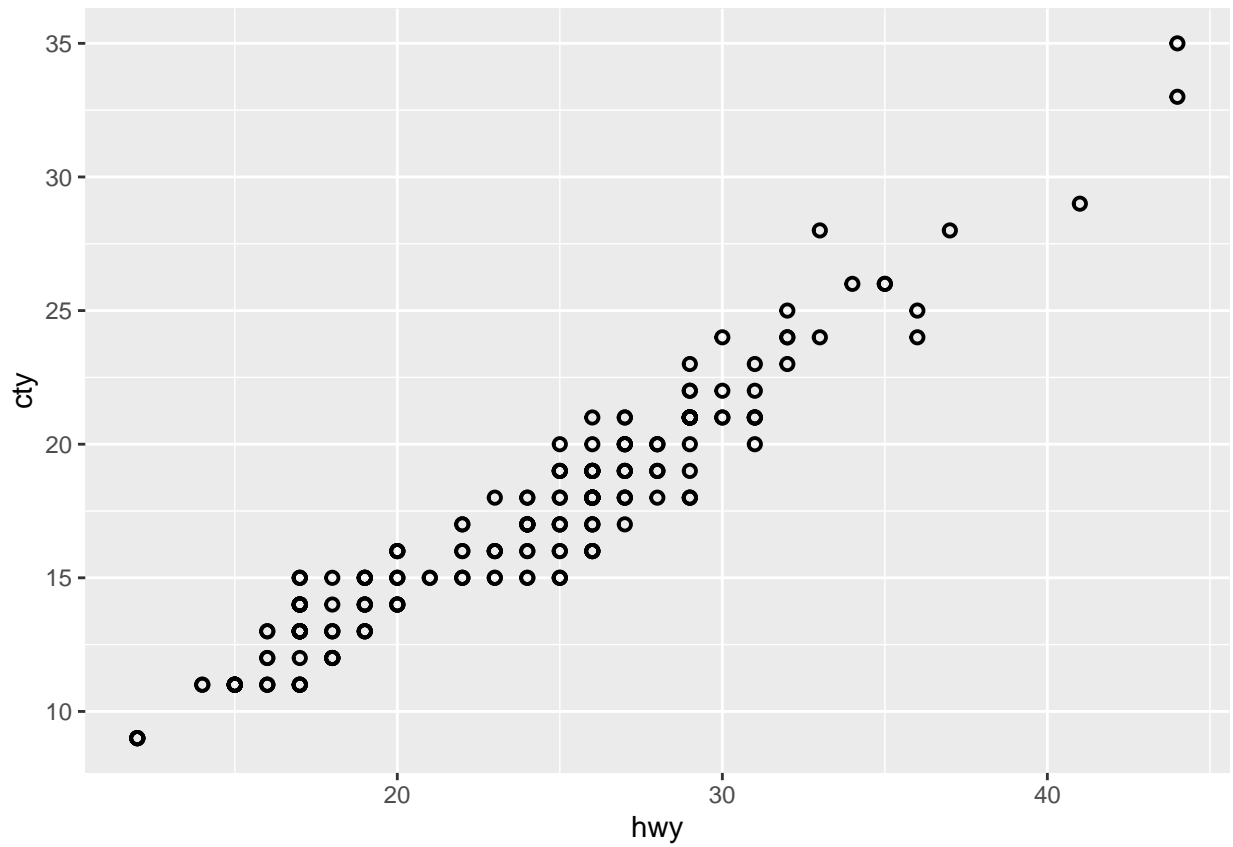
```
?geom_point
```

In one of the examples, we see the following comment:

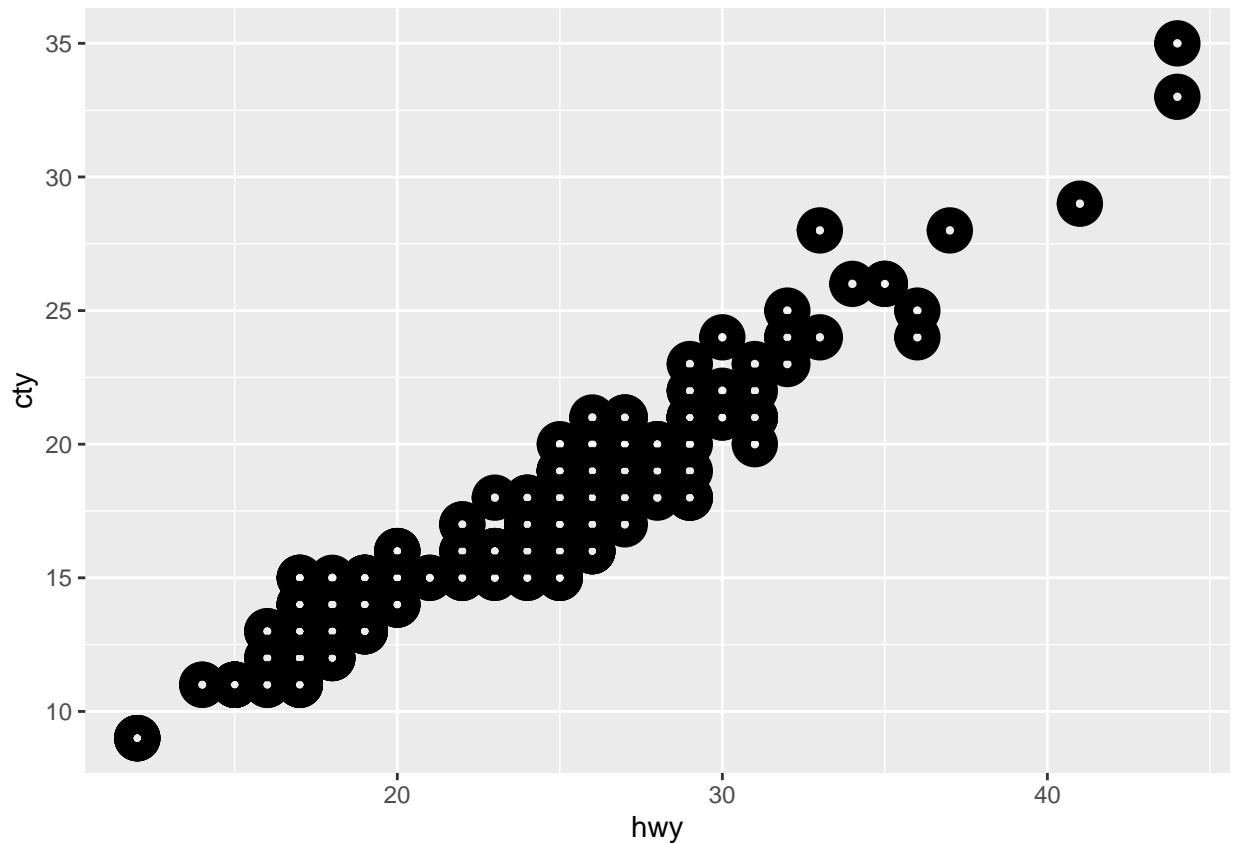
For shapes that have a border (like 21), you can colour the inside and outside separately. Use the **stroke** aesthetic to modify the width of the border

So **stroke** changes the width of the border of a shape's border. Let's look at two examples with different **stroke** values to illustrate what **stroke** does:

```
# stroke = 1
ggplot(data = mpg) +
  geom_point(
    mapping = aes(x = hwy, y = cty),
    shape = 21,
    stroke = 1
  )
```



```
# stroke = 5
ggplot(data = mpg) +
  geom_point(
    mapping = aes(x = hwy, y = cty),
    shape = 21,
    stroke = 5
  )
```



Question 6

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

ggplot will evaluate expressions that aren't variable names. In the case of `aes(colour = displ < 5)`, **ggplot** will treat it as a logical test and return `TRUE` or `FALSE` then graph those results. We end up with points coloured based on whether they have `displ < 5` (`TRUE`) or `NOT displ < 5` (`FALSE`):

```
ggplot(data = mpg) +
  geom_point(
    mapping = aes(x = hwy, y = cty, colour = displ < 5)
  )
```