# R for Data Science PSTAT 131

JEFFREY CHAN

12/13/2020

please use this command to install the library `install.packages("tidyverse")`

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------------------------------------------

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

## -- Conflicts ---------------------------------------------------------------------------------- tid
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

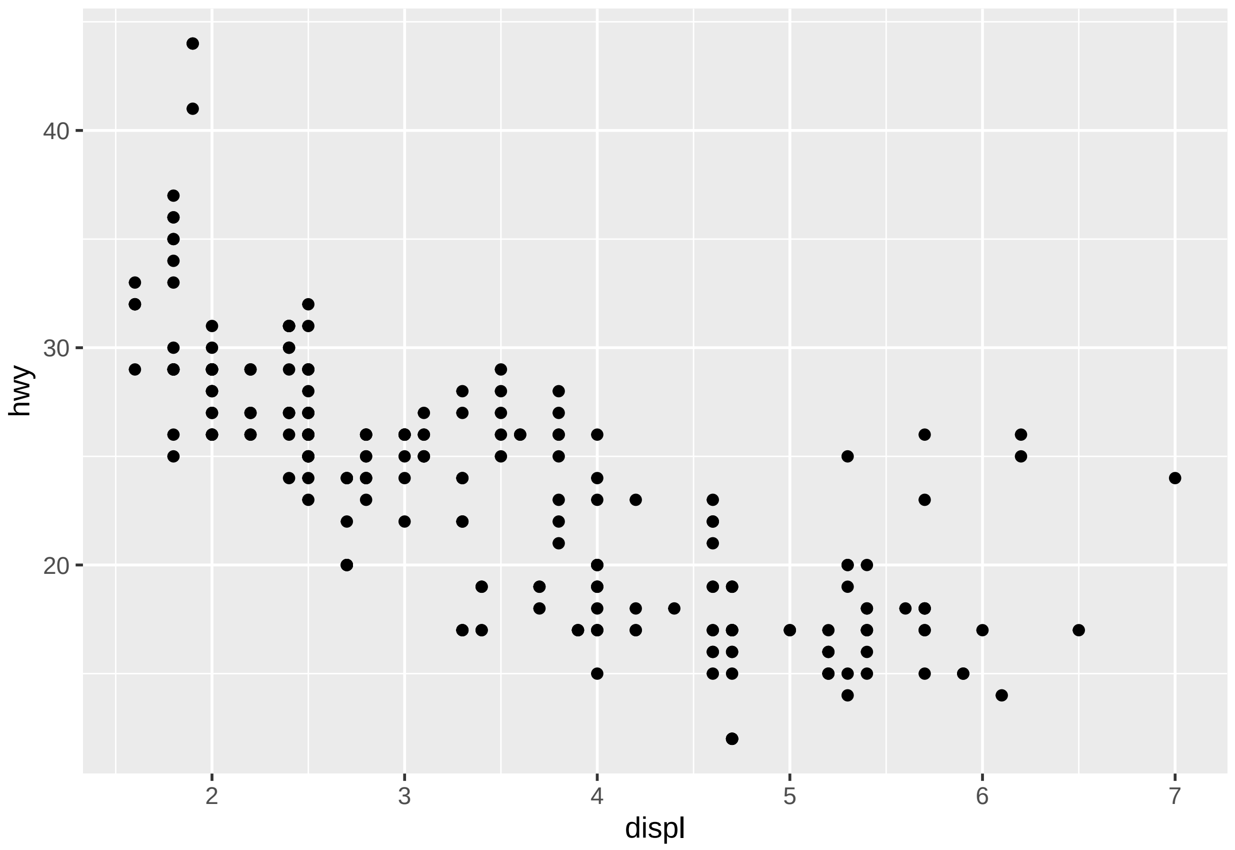
ˆ if you can see, there is some conflict from tidyverse library and dplyr library. So when we are using those functions, either from stats or dplyr, we need to specify the library name.

build in data frame / data called mpg Lets find out if do big engines use more fuel

```r
str(mpg)
```

```
## tibble [234 x 11] (S3: tbl_df/tbl/data.frame)
##  $ manufacturer: chr [1:234] "audi" "audi" "audi" "audi" ...
##  $ model       : chr [1:234] "a4" "a4" "a4" "a4" ...
##  $ displ       : num [1:234] 1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
##  $ year        : int [1:234] 1999 1999 2008 2008 1999 1999 2008 1999 1999 2008 ...
##  $ cyl         : int [1:234] 4 4 4 4 6 6 6 4 4 4 ...
##  $ trans       : chr [1:234] "auto(l5)" "manual(m5)" "manual(m6)" "auto(av)" ...
##  $ drv         : chr [1:234] "f" "f" "f" "f" ...
##  $ cty         : int [1:234] 18 21 20 21 16 18 18 18 16 20 ...
##  $ hwy         : int [1:234] 29 29 31 30 26 26 27 26 25 28 ...
##  $ fl          : chr [1:234] "p" "p" "p" "p" ...
##  $ class       : chr [1:234] "compact" "compact" "compact" "compact" ...
```

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

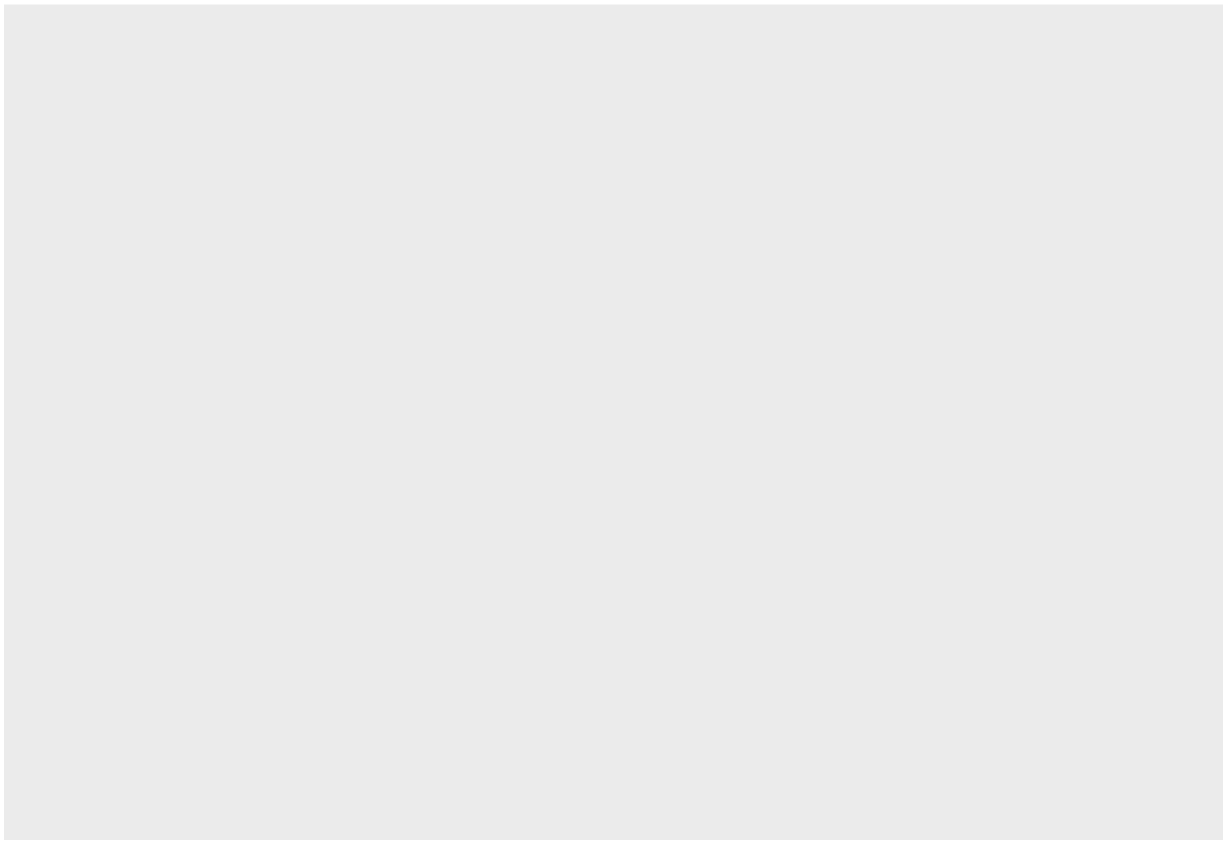


In ggplot2, ggplot() creates the coordinate system, and then i can add layers on top of the coordinate. in deed, geom_point is my second layer. Also, ggplot will take the data set as parameter / input. geom_point creates the scatter dots. this goes in pair in side the geom_point(mappint = aes(x = , y = )) this will tell the graph what data on the x and y axies

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

exercises

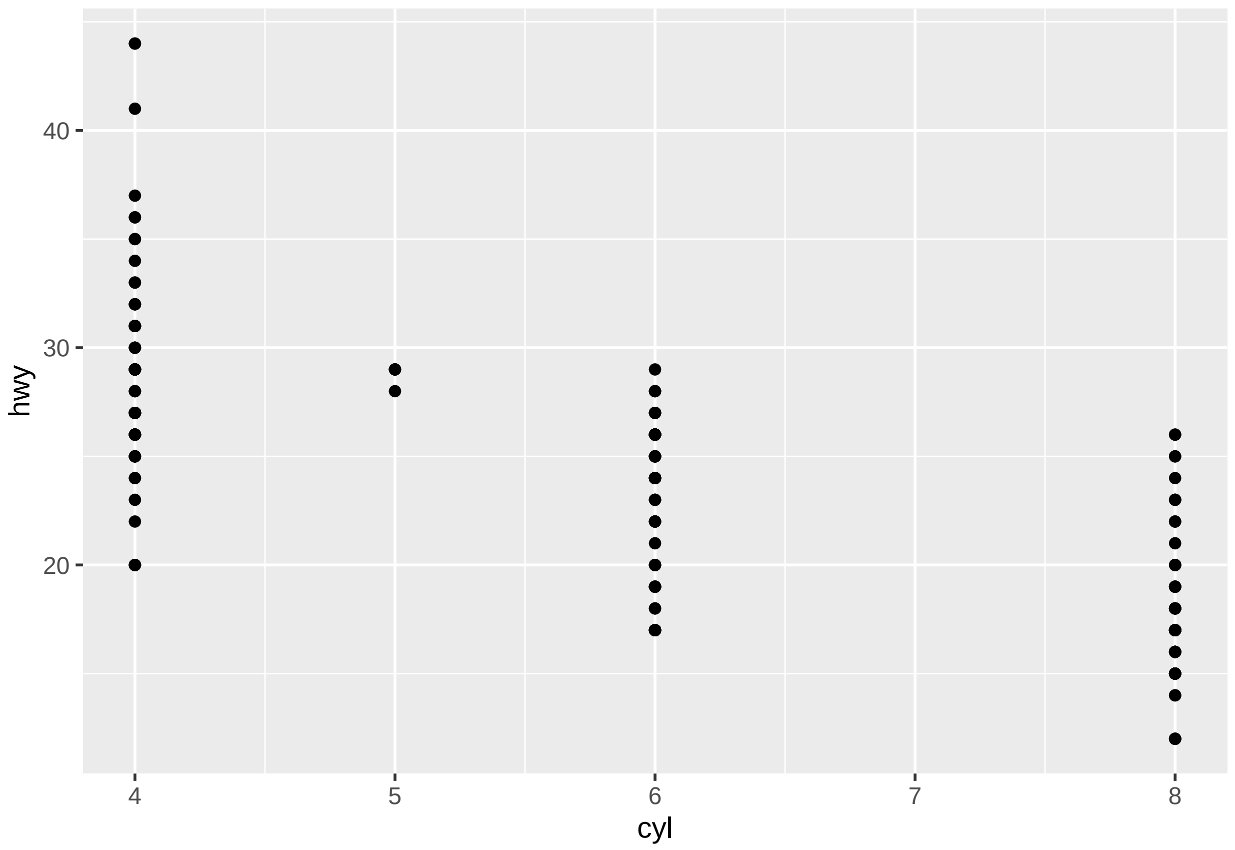
```
ggplot(data = mpg)
```



```r
dim(mtcars)
```

```
## [1] 32 11
```

```r
?mpg
ggplot(data = mpg) +
  geom_point(mapping = aes(x = cyl, y = hwy))
```



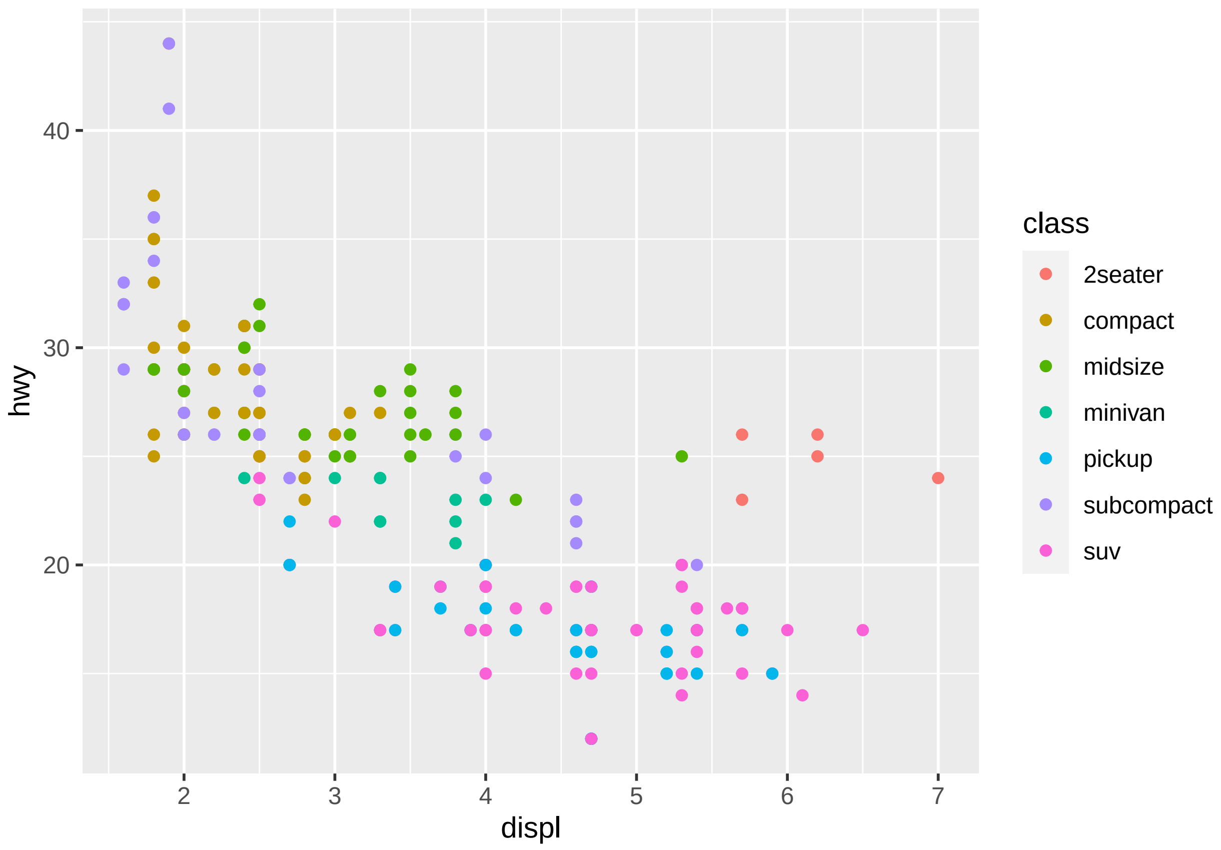
let jump back to mpg plot. we know that we plot hwy ~ displ or y = hwy, x = displ, but what if we want to know more insight like what type of vehicles are those? we can identify it with color Lets check what variables do they have in mpg

```r
str(mpg)
```

```
## tibble [234 x 11] (S3: tbl_df/tbl/data.frame)
##  $ manufacturer: chr [1:234] "audi" "audi" "audi" "audi" ...
##  $ model       : chr [1:234] "a4" "a4" "a4" "a4" ...
##  $ displ       : num [1:234] 1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
##  $ year        : int [1:234] 1999 1999 2008 2008 1999 1999 2008 1999 1999 2008 ...
##  $ cyl         : int [1:234] 4 4 4 4 6 6 6 4 4 4 ...
##  $ trans       : chr [1:234] "auto(l5)" "manual(m5)" "manual(m6)" "auto(av)" ...
##  $ drv         : chr [1:234] "f" "f" "f" "f" ...
##  $ cty         : int [1:234] 18 21 20 21 16 18 18 18 16 20 ...
##  $ hwy         : int [1:234] 29 29 31 30 26 26 27 26 25 28 ...
##  $ fl          : chr [1:234] "p" "p" "p" "p" ...
##  $ class       : chr [1:234] "compact" "compact" "compact" "compact" ...
```

^ as we see we have class that illustrate what type of vehicle is it so we will use that info to plot a better graph

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



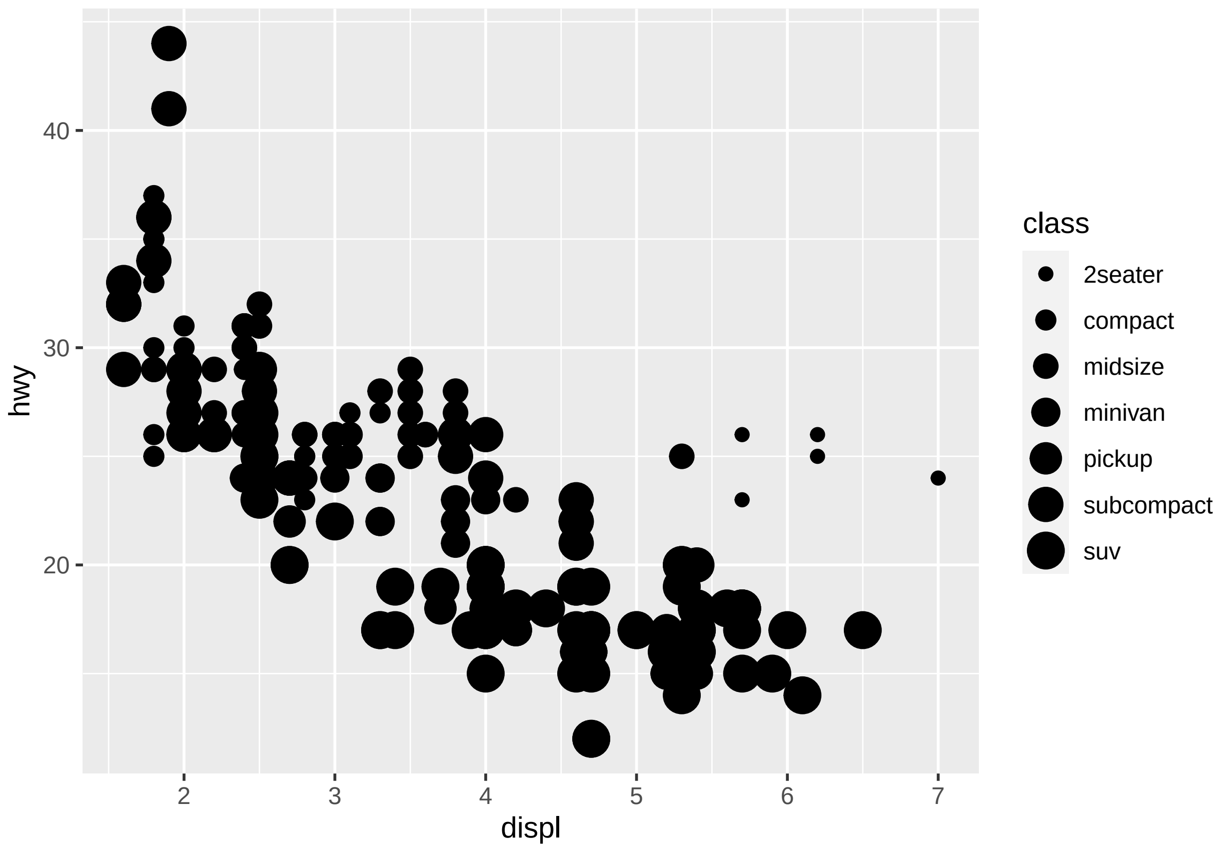
^

now we set different color by it's own type of vehicle type it is more information from and we can more a better conclusion. color function comes with automatically labelling the vehicle types at the side.

now we can plot the graph with the following

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

```
## Warning: Using size for a discrete variable is not advised.
```



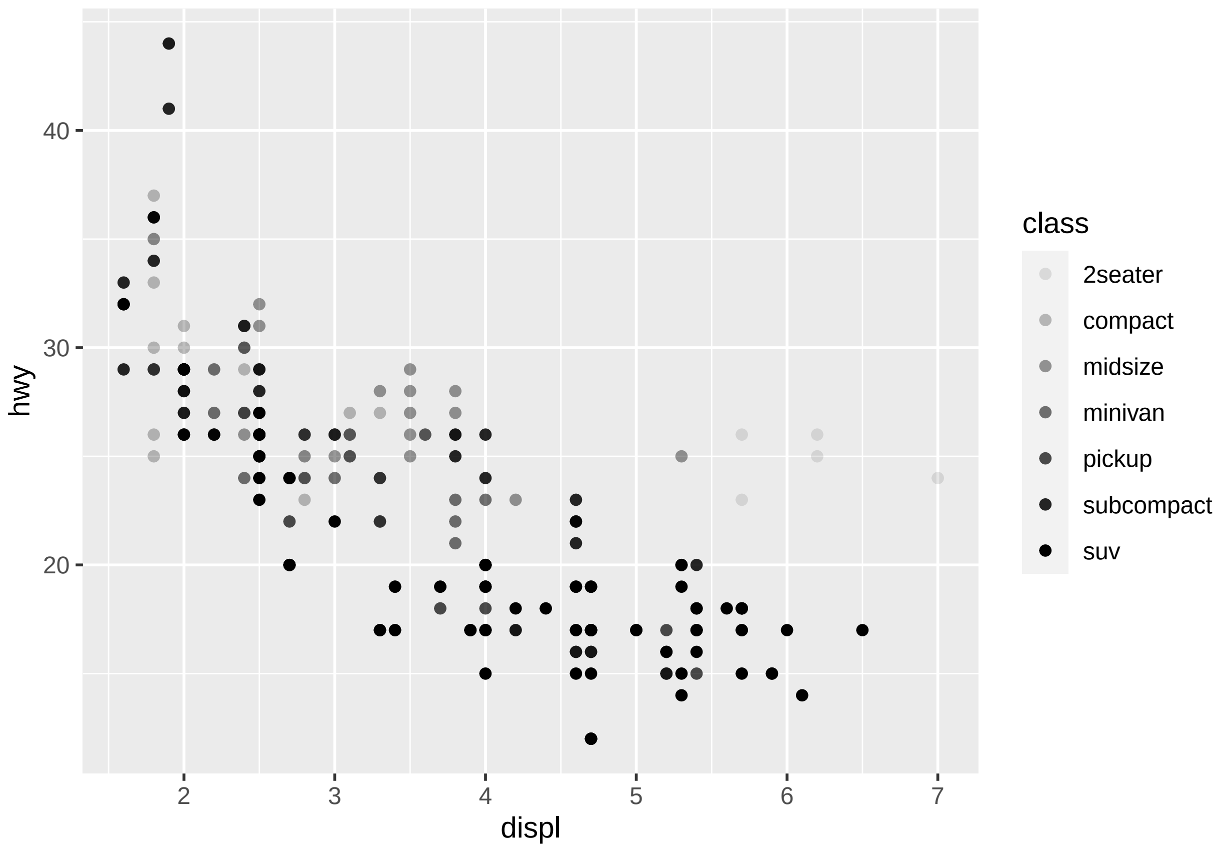
^

as you can see, we can a warning "using size for a discrete varaible is not advised." because mapping an unordered variable (class) to an ordered aesthetic (size) is not a good idea.

Alternative

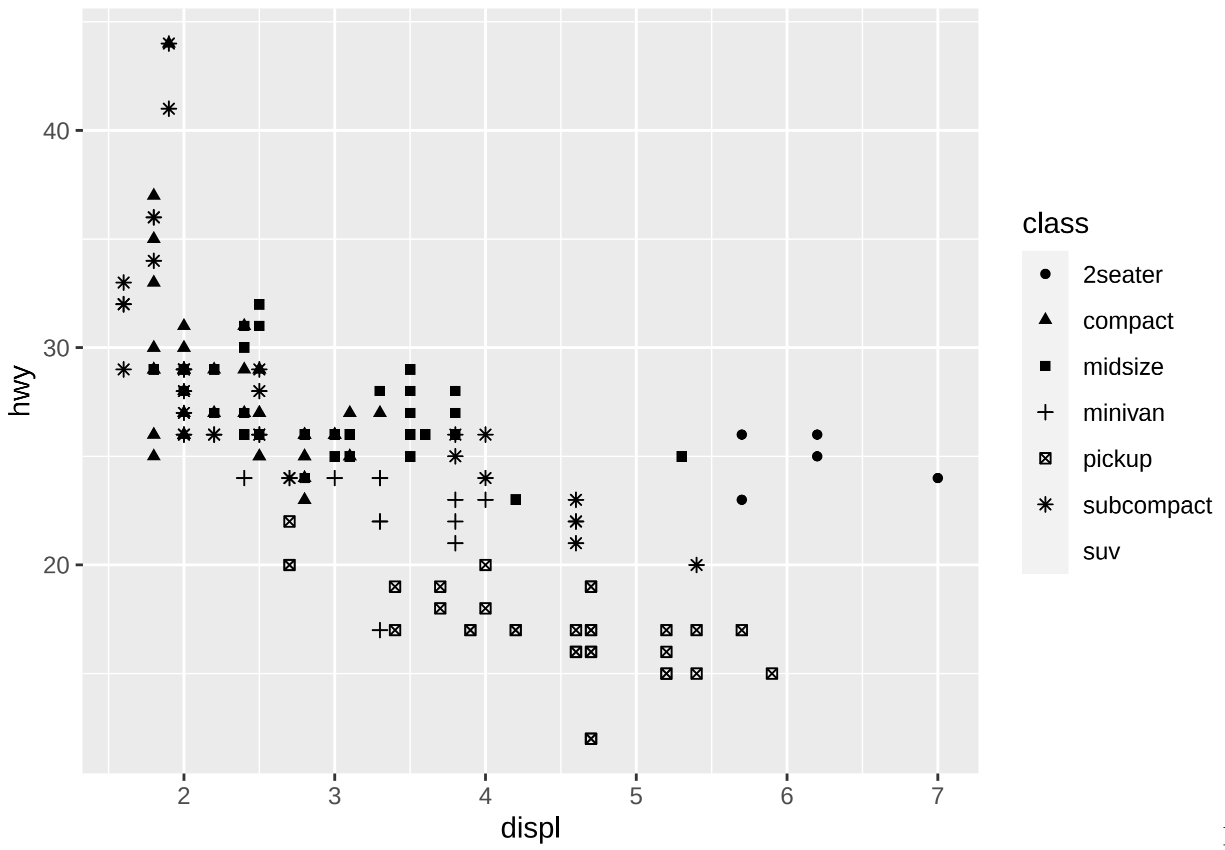
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, alpha = class))
```

```
## Warning: Using alpha for a discrete variable is not advised.
```



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

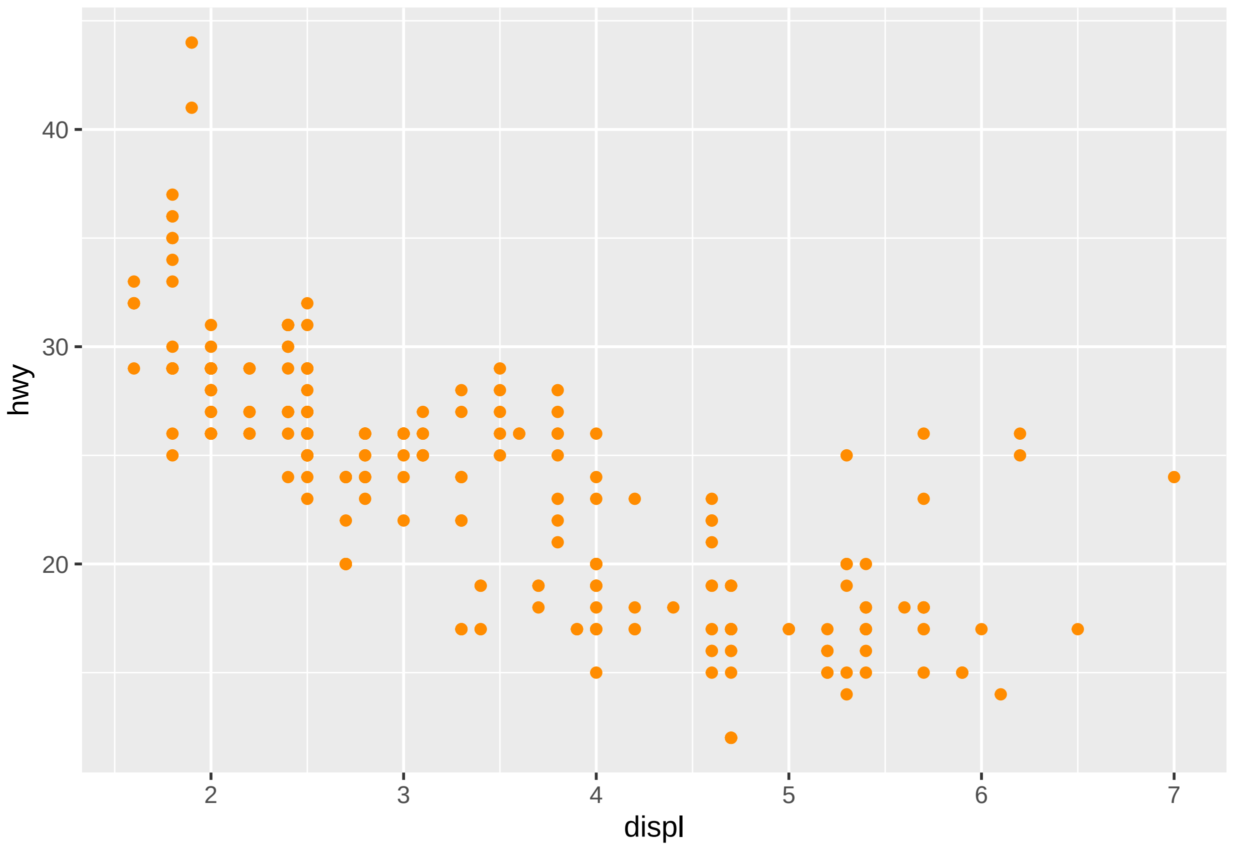
```
## Warning: The shape palette can deal with a maximum of 6 discrete values because
## more than 6 becomes difficult to discriminate; you have 7. Consider
## specifying shapes manually if you must have them.

## Warning: Removed 62 rows containing missing values (geom_point).
```



By now you should see the warning if you use rmd to read this file. shape = class this function right here, in ggplot2, it only reserves 6 different symbols only.

we can also set the color manually like the following

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



I

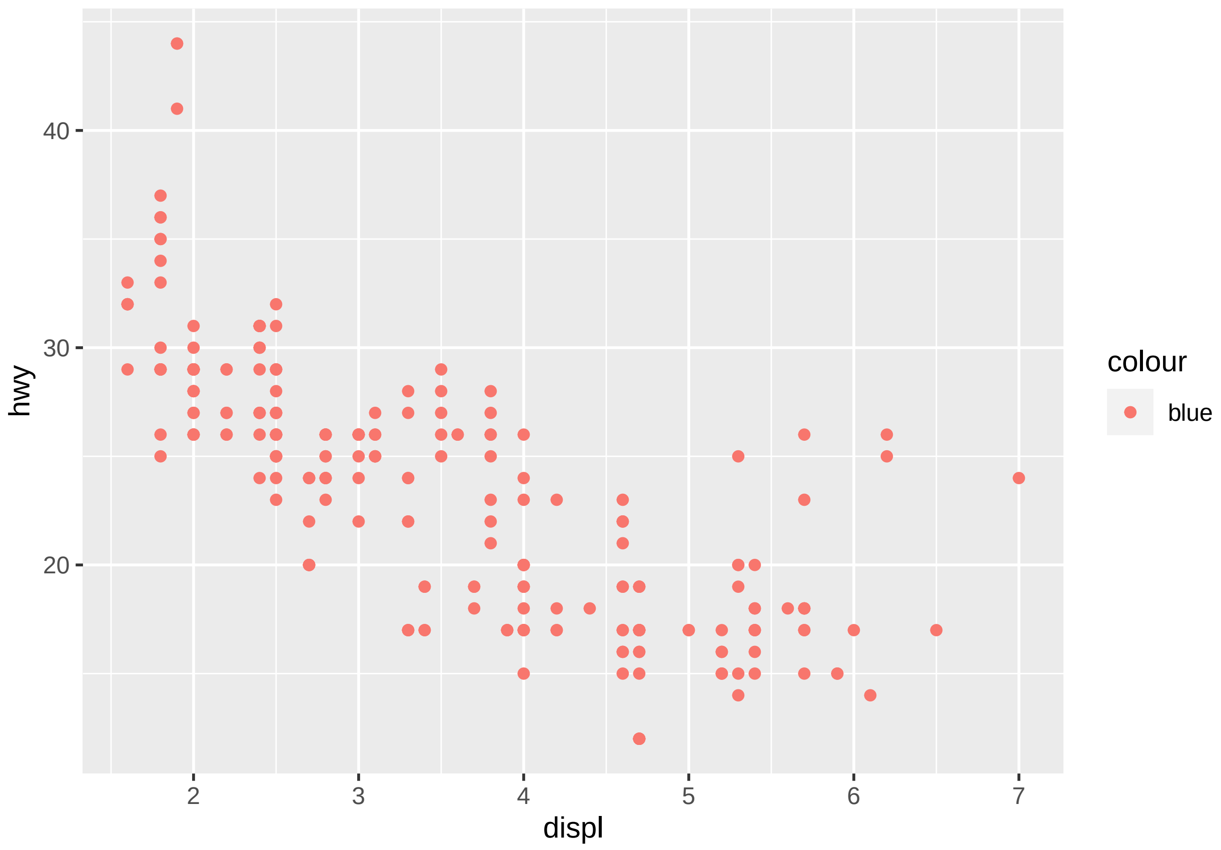
hope you are paying attention to the code itself. As you can see, if we want to define the color, the argument is outside of aes. `ggplot(mpg) +     geom_point(aes(x = displ, y = hwy), color = "darkorange")`

Compare with the previous one `ggplot(data = mpg) +   geom_point(mapping = aes(x = displ, y = hwy, shape = class))   ggplot(data = mpg) +   geom_point(mapping = aes(x = displ, y = hwy, alpha = class))   ggplot(mpg) +    geom_point(aes(x = displ, y = hwy, size = class)) ggplot(mpg) +    geom_point(aes(x = displ, y = hwy, color = class))`

All the color function is INSIDE the aes function. Please take note for this step If you refere to the table / the book. 0-14 hollow shapes can be defined with color 15-19 solid shapes are filled with color 21-24 filled shapes and it can be defined a new boarder color

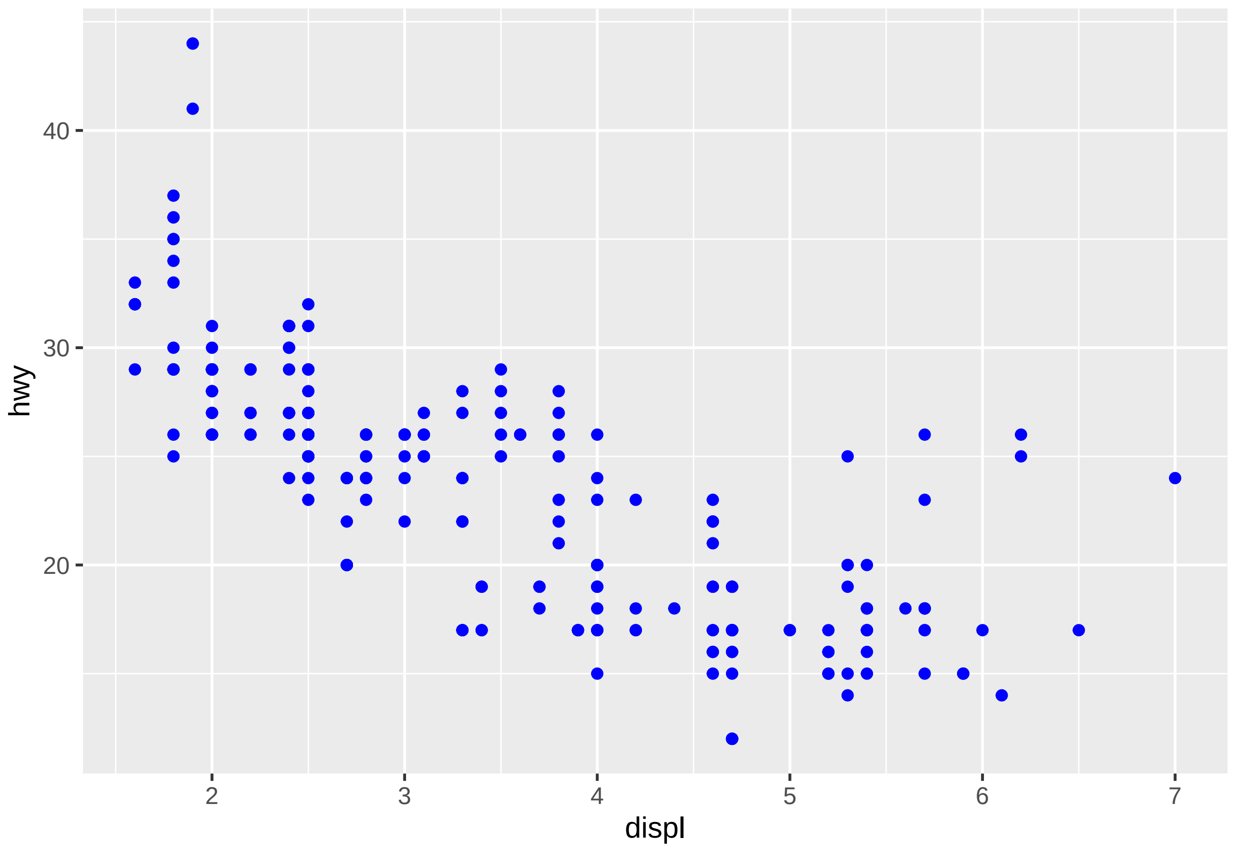
Q1 Why it is not blue?

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



because color = “blue” is inside the aes, move it outside of aes will work

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



Q2 Map a continuous variable to color, size, and shape. How do these aesthetics behave differently for categorical versus contin- uous variables?

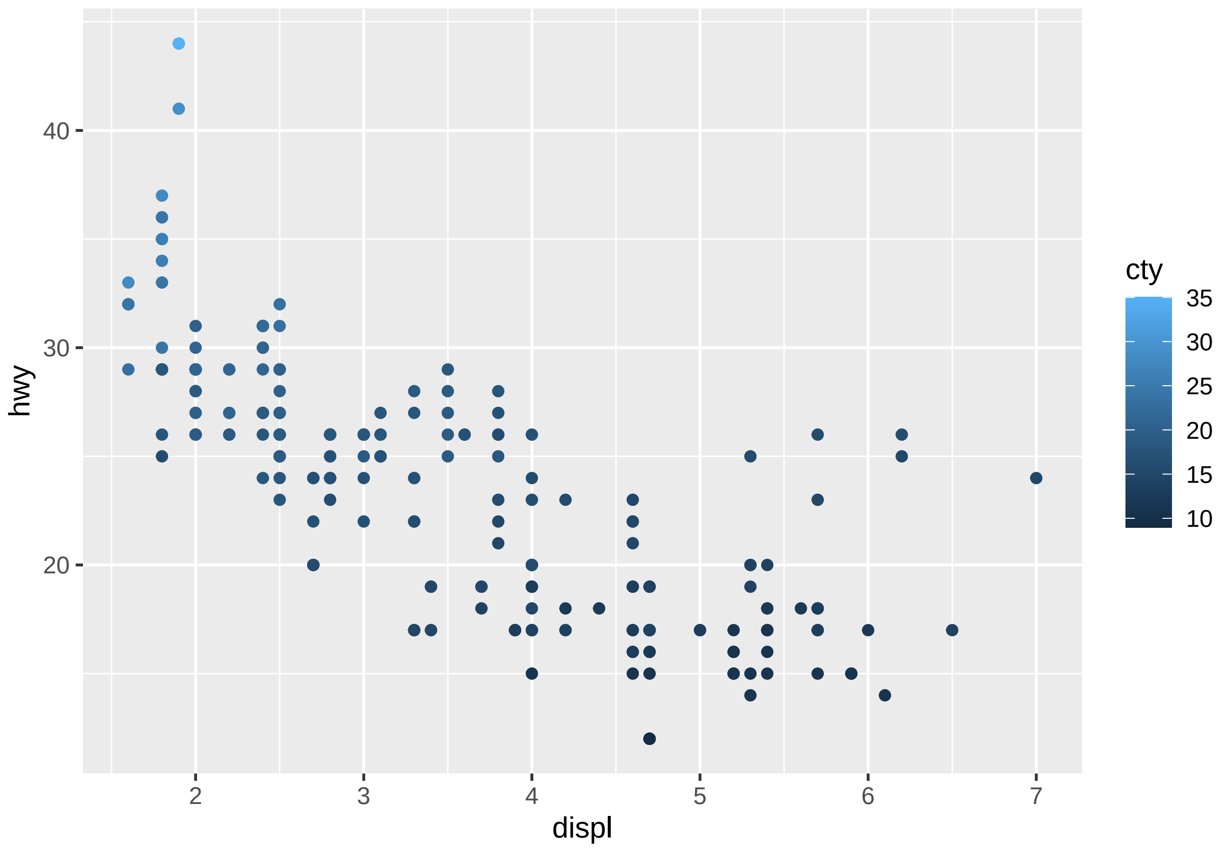
```
mpg
```

```
## # 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
```

ˆ when you type mpg, you will have this table. quick important note, when you see under the variable name, then it is categorical variable.

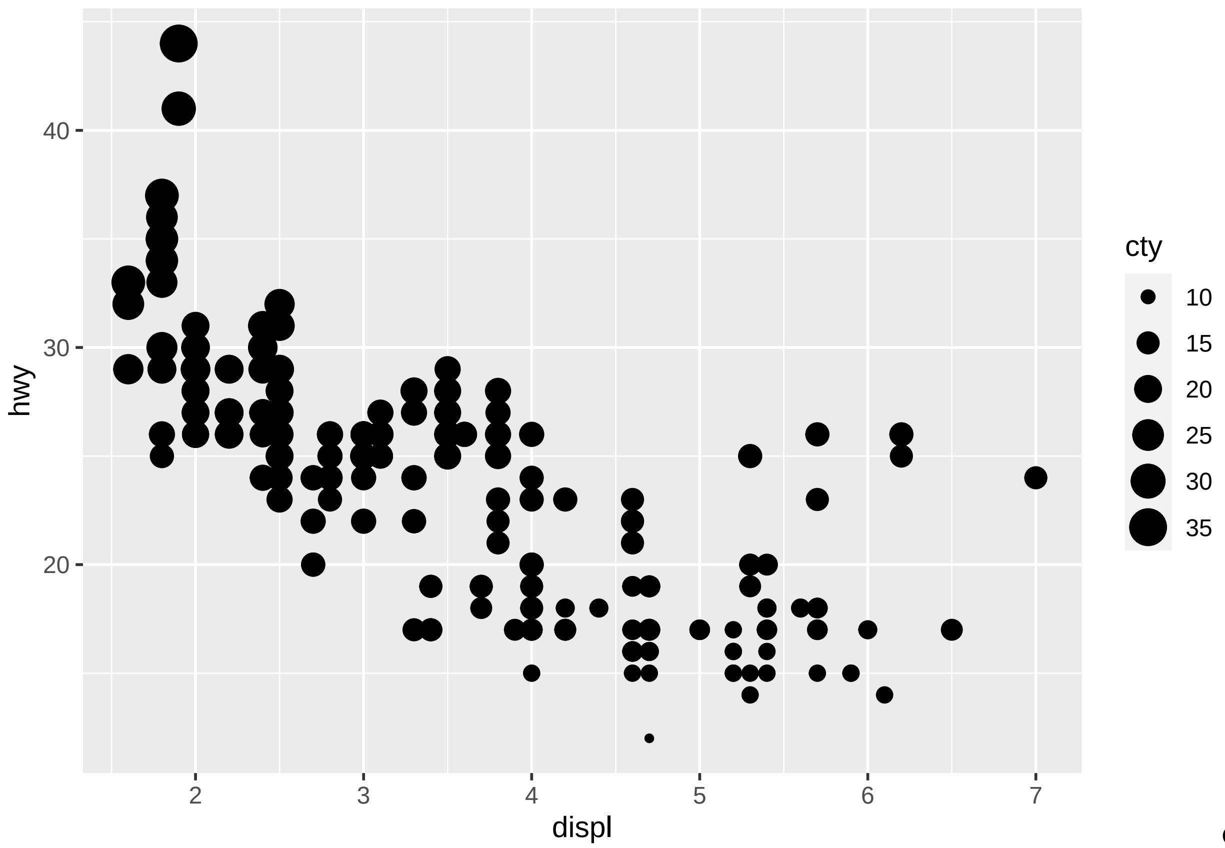
Q3 Map a continuous variable to color, size, and shape. How do these aesthetics behave differently for categorical versus contin- uous variables?

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



instead of using discrete colors, continuous variable uses a scale that varies from light to dark.

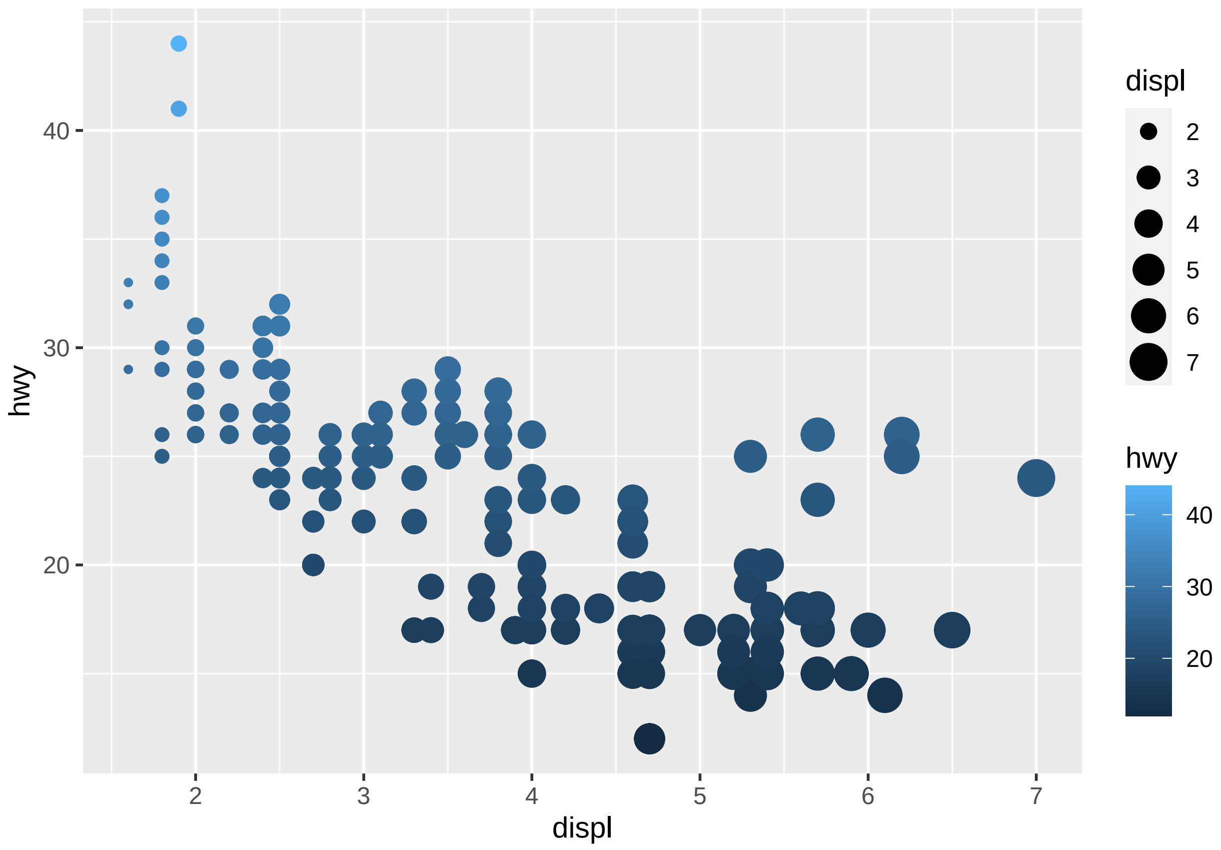
```
ggplot(mpg, aes(x=displ, y = hwy, size = cty)) +
  geom_point()
```



Q4

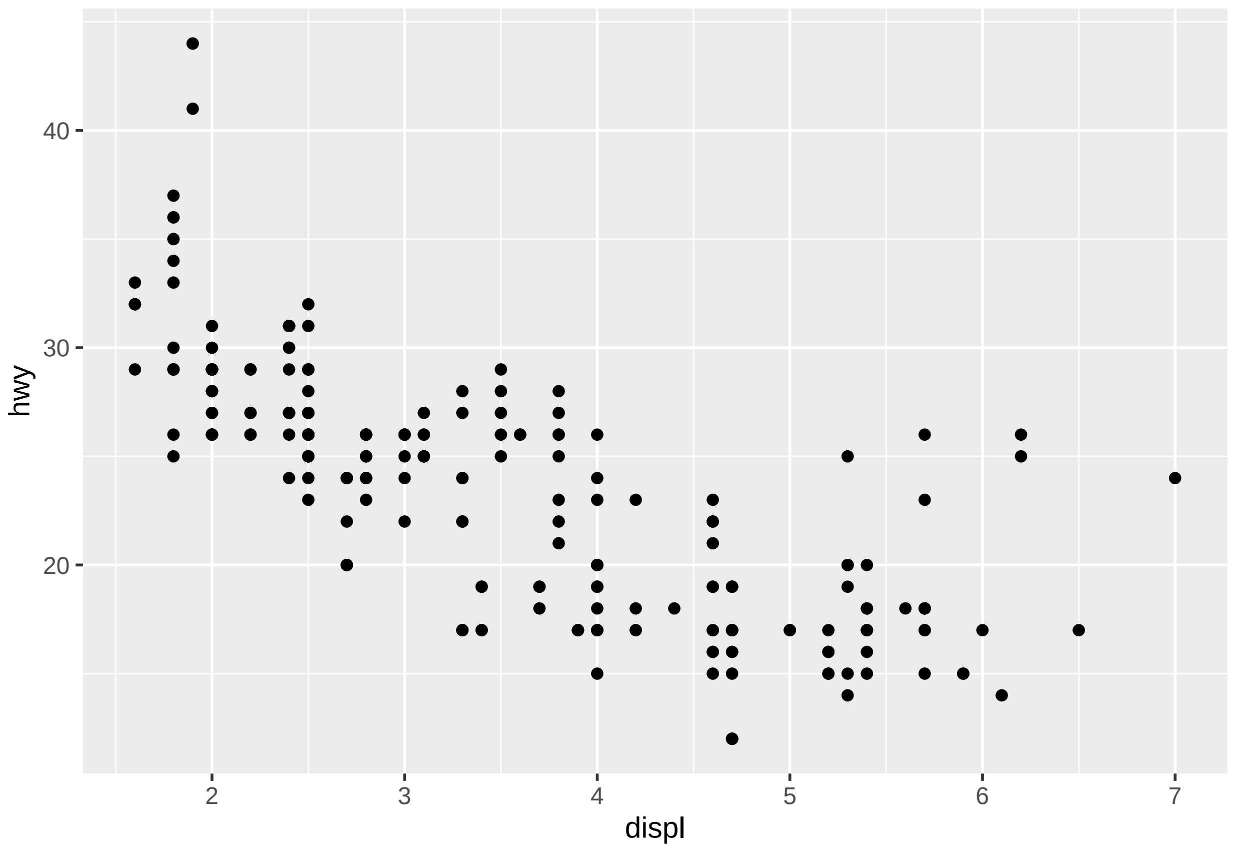
What happens if you map the same variable to multiple aesthetics? it just looks bad, because it is redundant

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



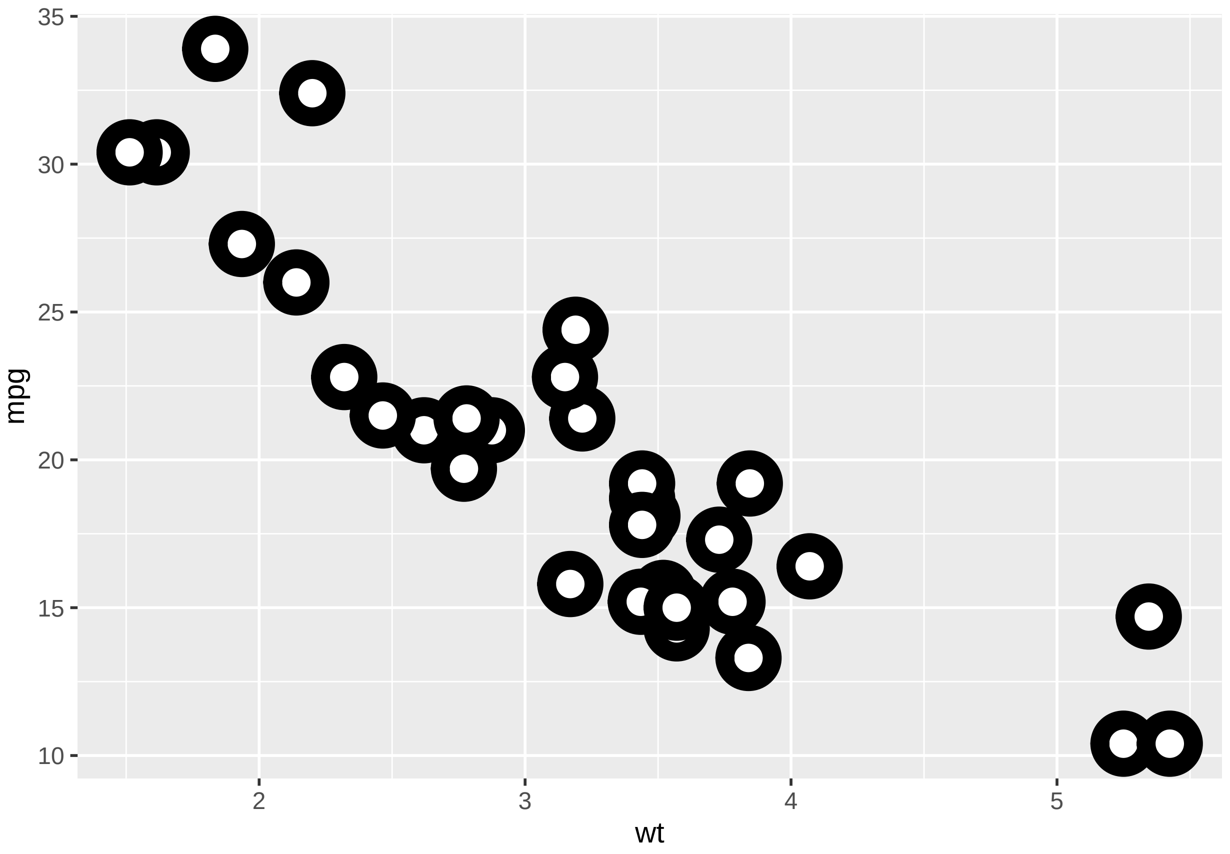
another way of plotting the ggplot

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

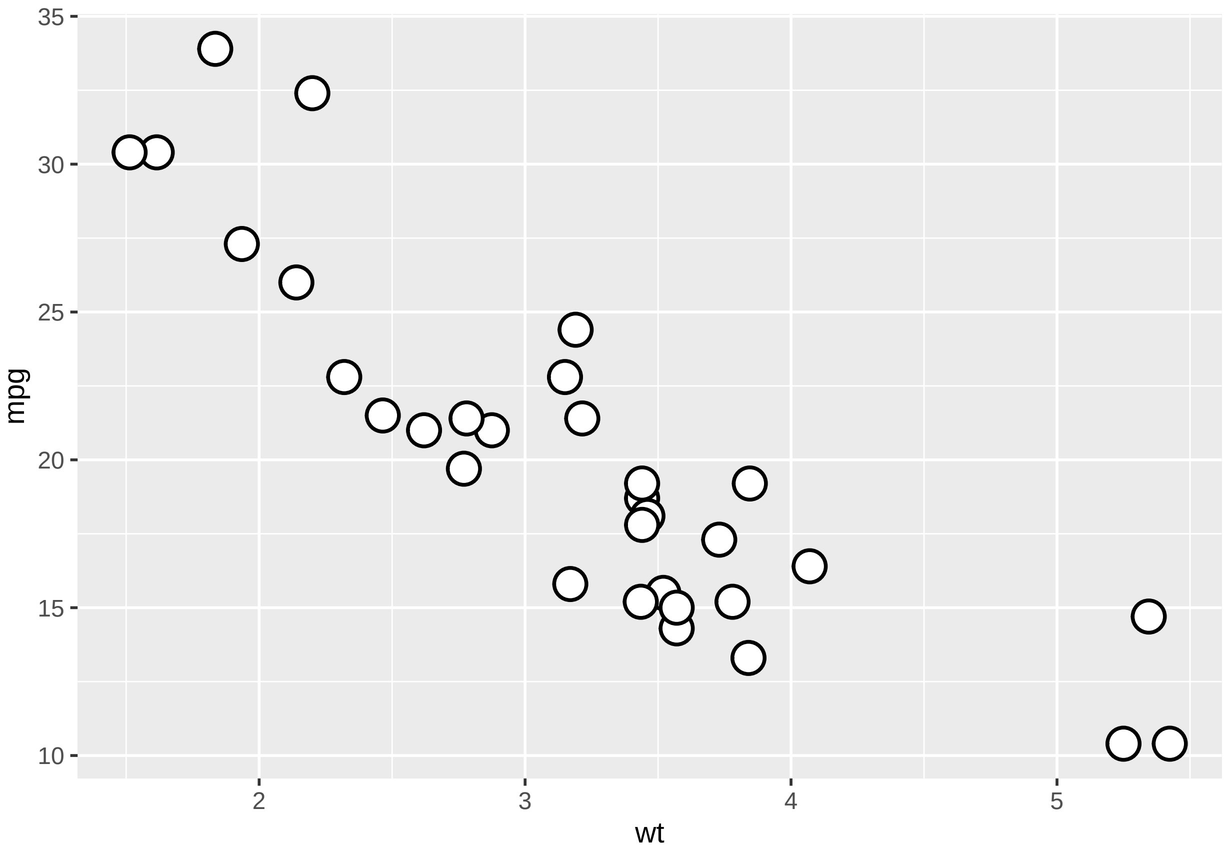


Q5 What does the stroke aesthetic do? What shapes does it work with? (Hint: use ?geom_point.)

```
ggplot(mtcars, aes(wt,mpg)) +
  geom_point(shape = 21, color = "black", fill = "white", size = 5,
             stroke = 5)
```



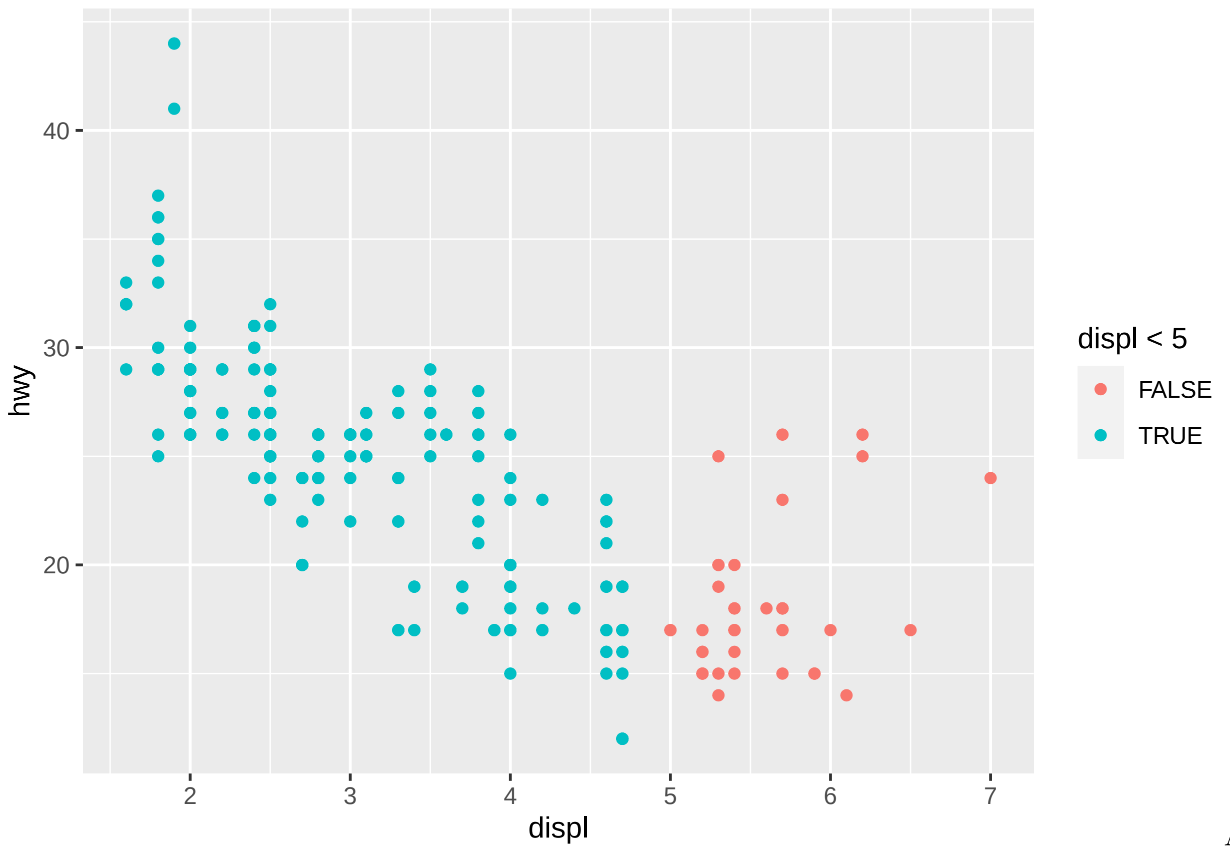
```
ggplot(mtcars, aes(wt,mpg)) +
  geom_point(shape = 21, color = "black", fill = "white", size = 5,
             stroke = 1)
```



As you can see stroke control the border size and stroke will only work for the shapes (21-25)

Q6 What happens if you map an aesthetic to something other than a variable name, like aes(color = displ < 5)?

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



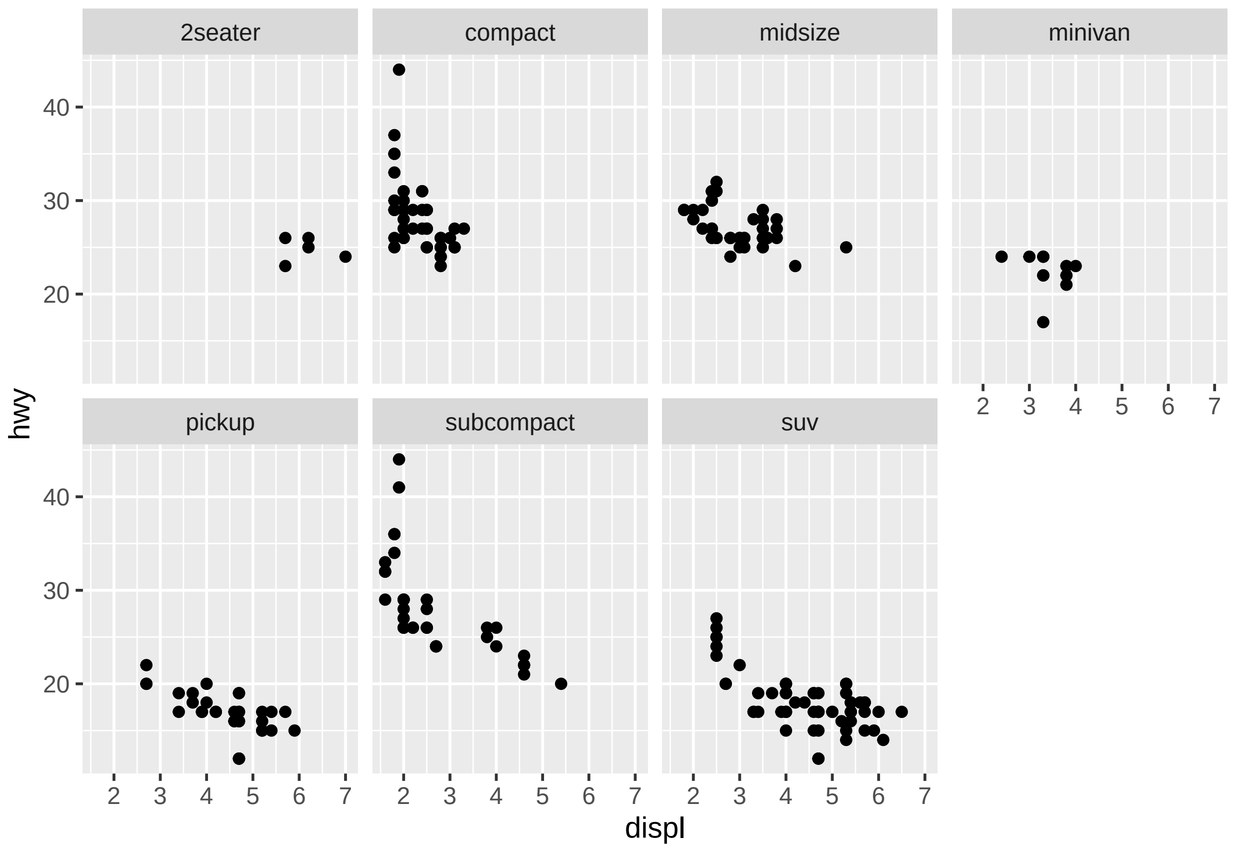
As you can see, if the displacement is >= 5 it will switch color from green to red.

```r
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
```

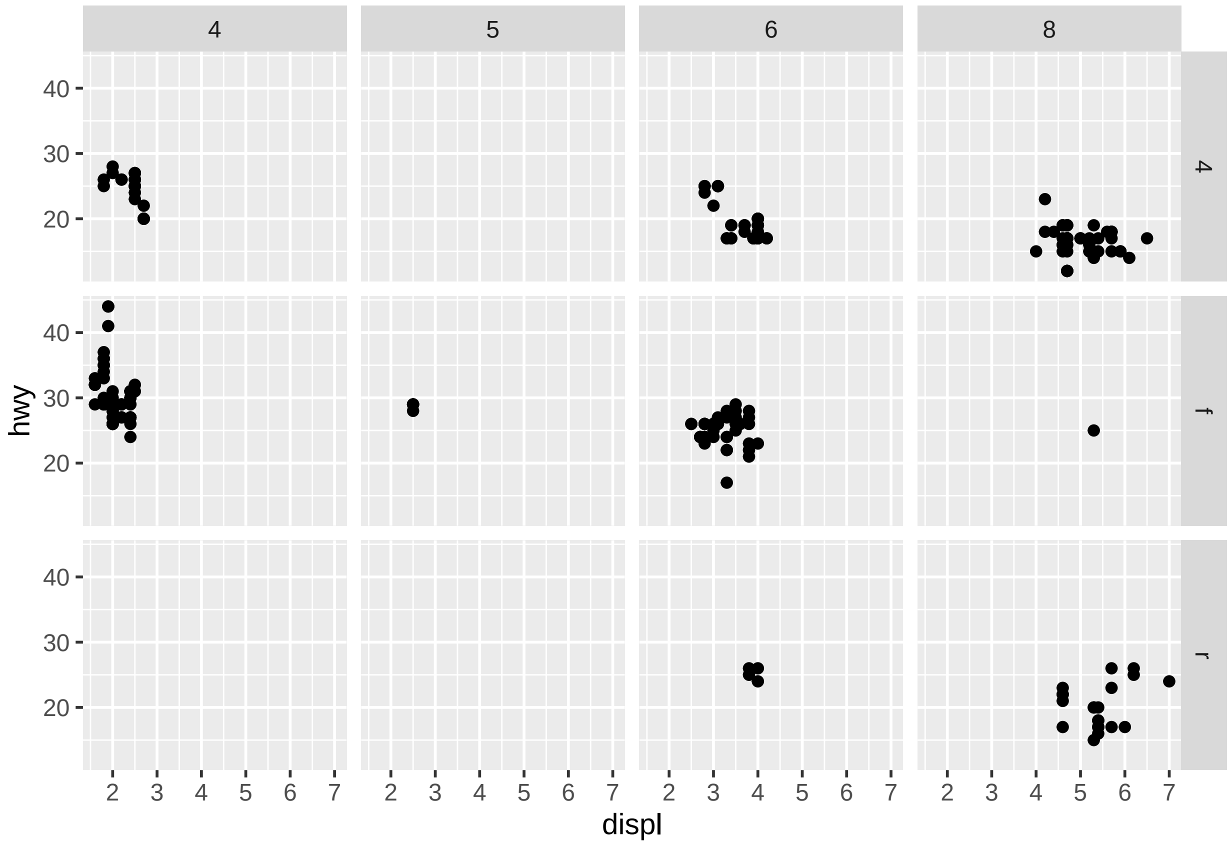
Facets this will plot the graph by it's own kind code: facet_wrap(y ~ x variable_name, nrow = #?) i can do it by itself ( ~ class)

```r
ggplot(mpg) +
  geom_point(aes(x = displ, hwy)) +
  facet_wrap(~ class, nrow = 2)
```



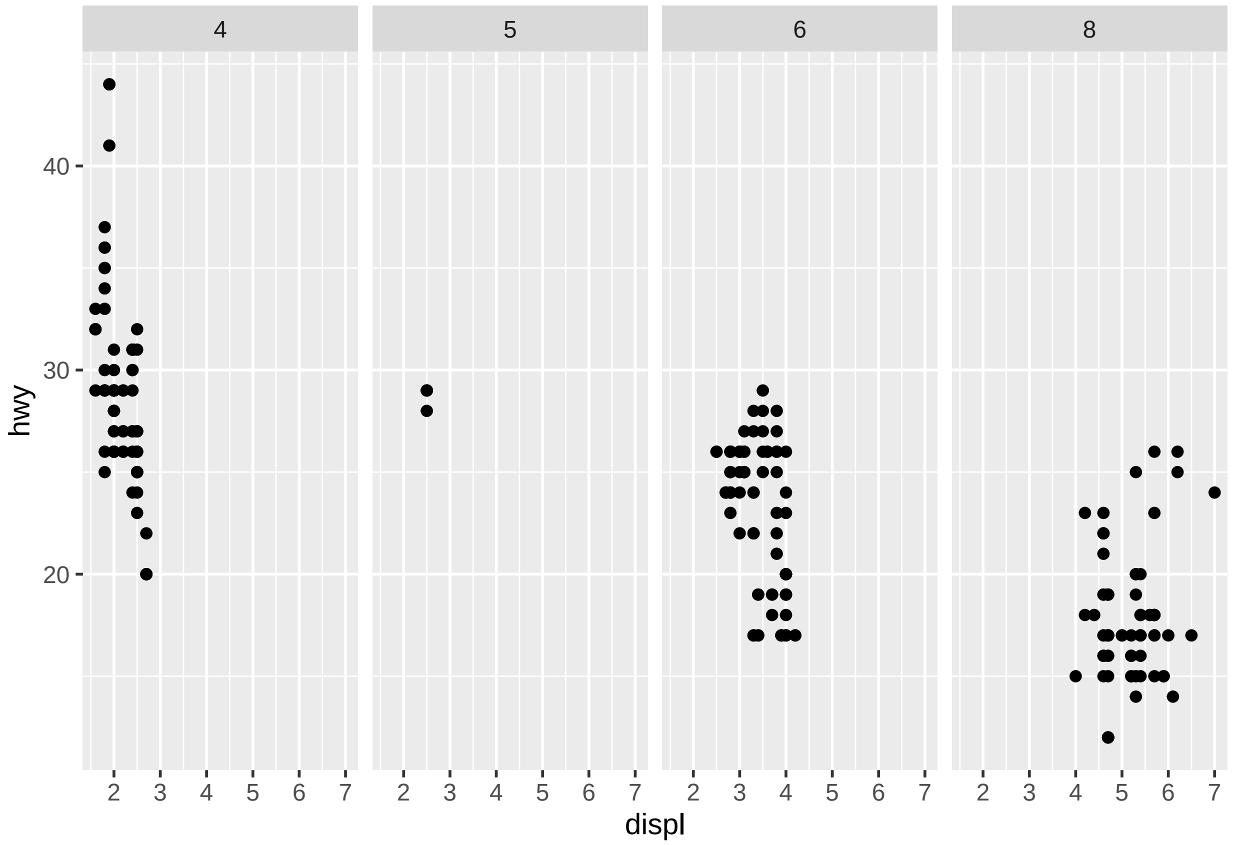
let's plot drv ~ cyl

```r
ggplot(mpg) +
  geom_point(aes(displ,hwy)) +
  facet_grid(drv ~ cyl)
```



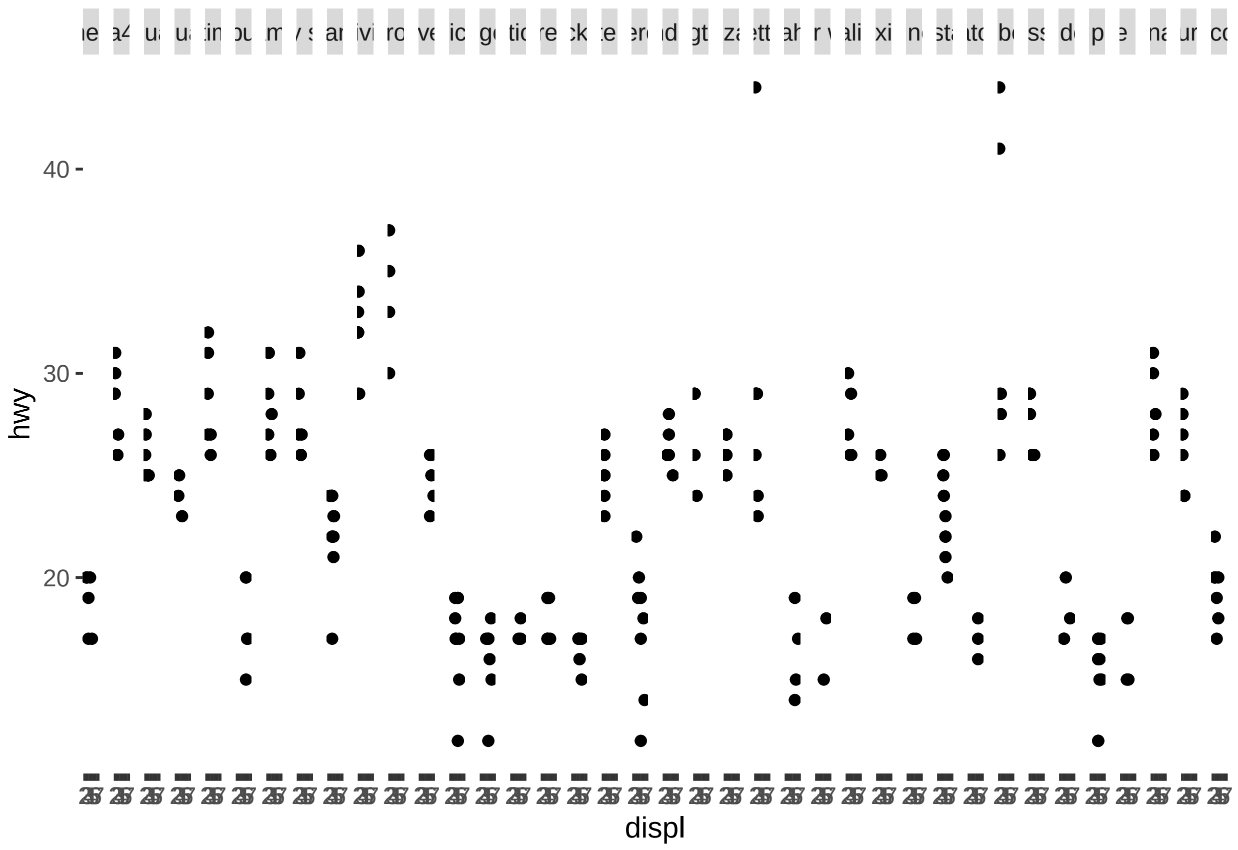
grid vs wrap is i have 2 more variables at the side. 4 5 6 8 on top is the cylinder, and 4 f r is the drive train. try to figure out how to read the variables from the data set with help function.

```r
ggplot(mpg) +
  geom_point(aes(displ,hwy)) +
  facet_grid(. ~ cyl)
```



Q1 What happens if you facet on a continuous variable?

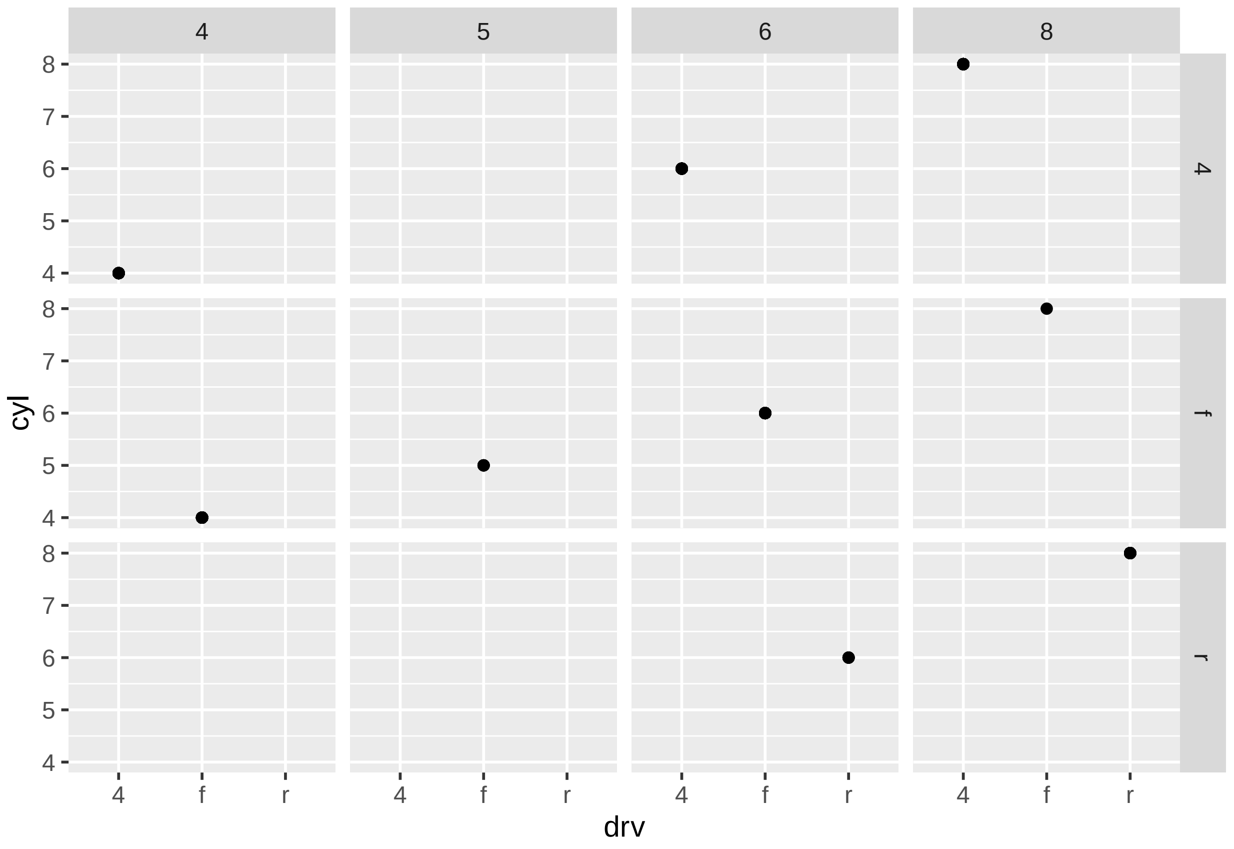
```r
ggplot(mpg) +
  geom_point(aes(displ,hwy)) +
  facet_grid(. ~ model)
```



As you can see the graph above, if i facet a continuous variable, the plot will become unreadable.

Q2 What do the empty cells in a plot with facet_grid(drv ~ cyl) mean? How do they relate to this plot?

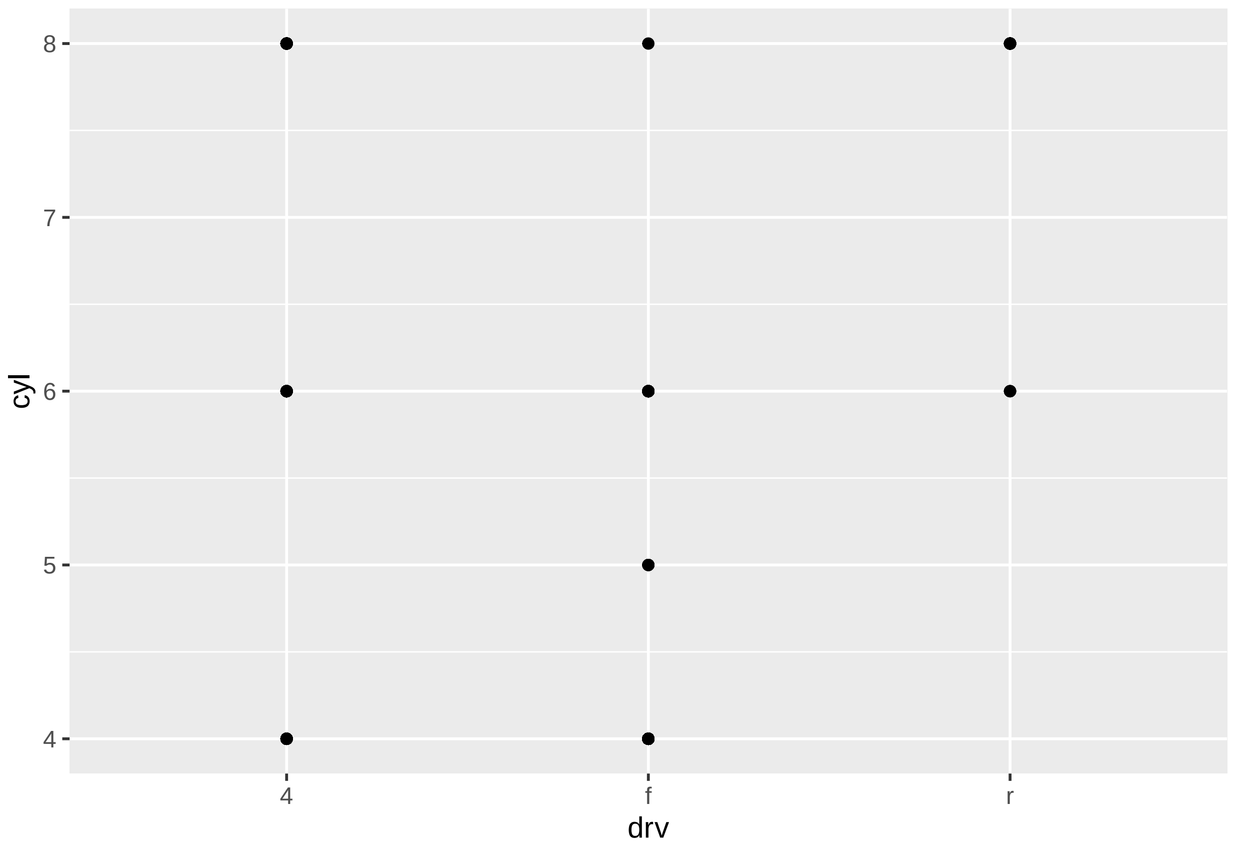
```r
ggplot(data = mpg) +
  geom_point(mapping = aes(x = drv, y = cyl)) +
  facet_grid(drv ~ cyl)
```



I

have two empty plots at the lower left corners

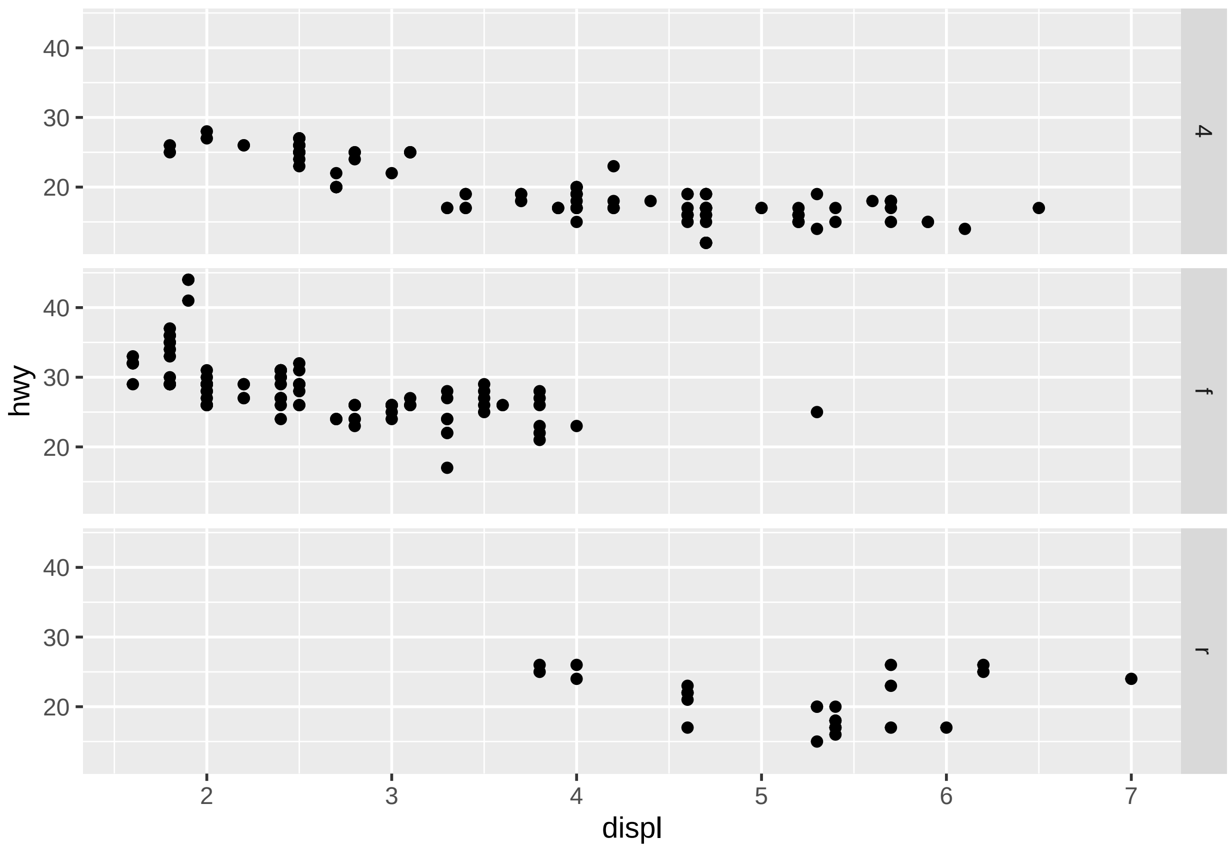
```
ggplot(mpg) +
  geom_point(aes(drv,cyl))
```



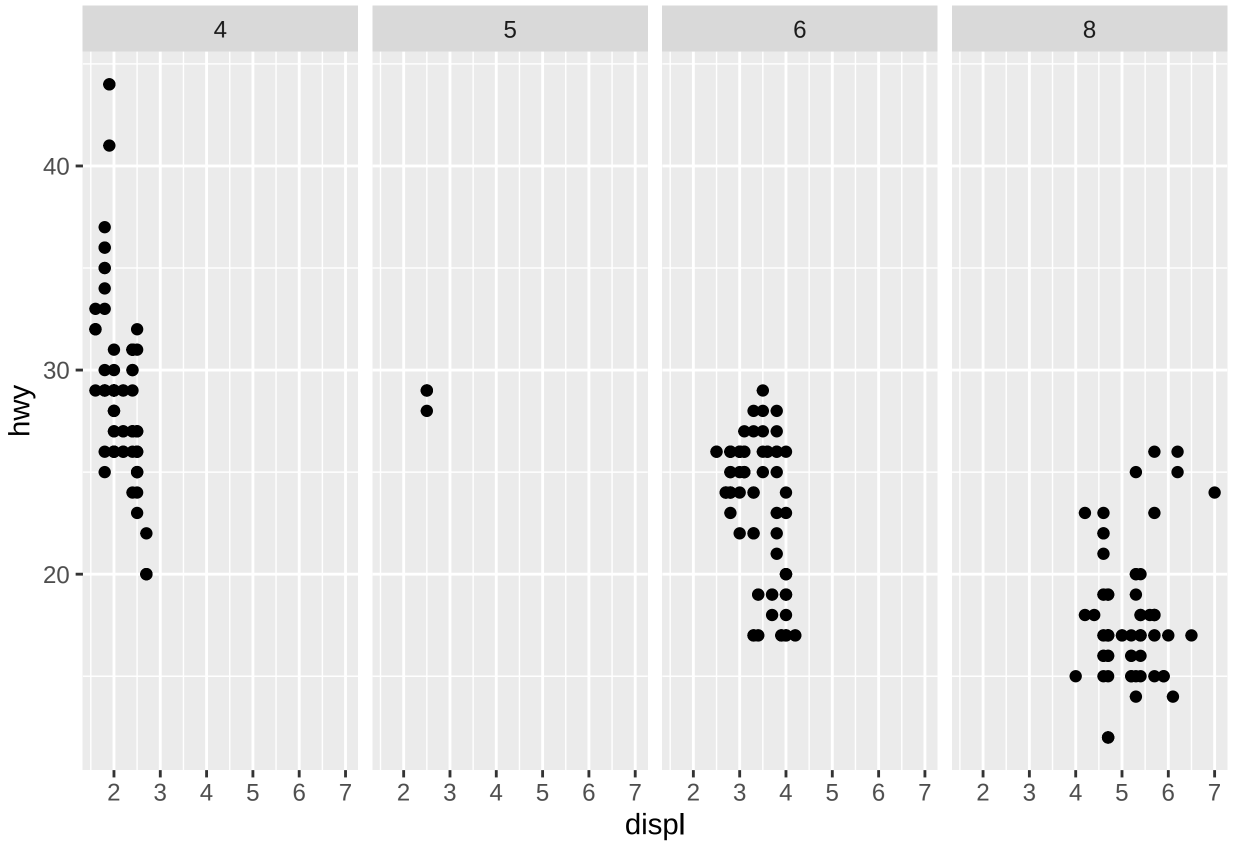
The empty cells (facets) in this plot are combinations of drv and cyl that have no observations. These are the same locations in the scatter plot of drv and cyl that have no points. (from the book but i still dont get it)

Q3 What plots does the following code make? What does . do?

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



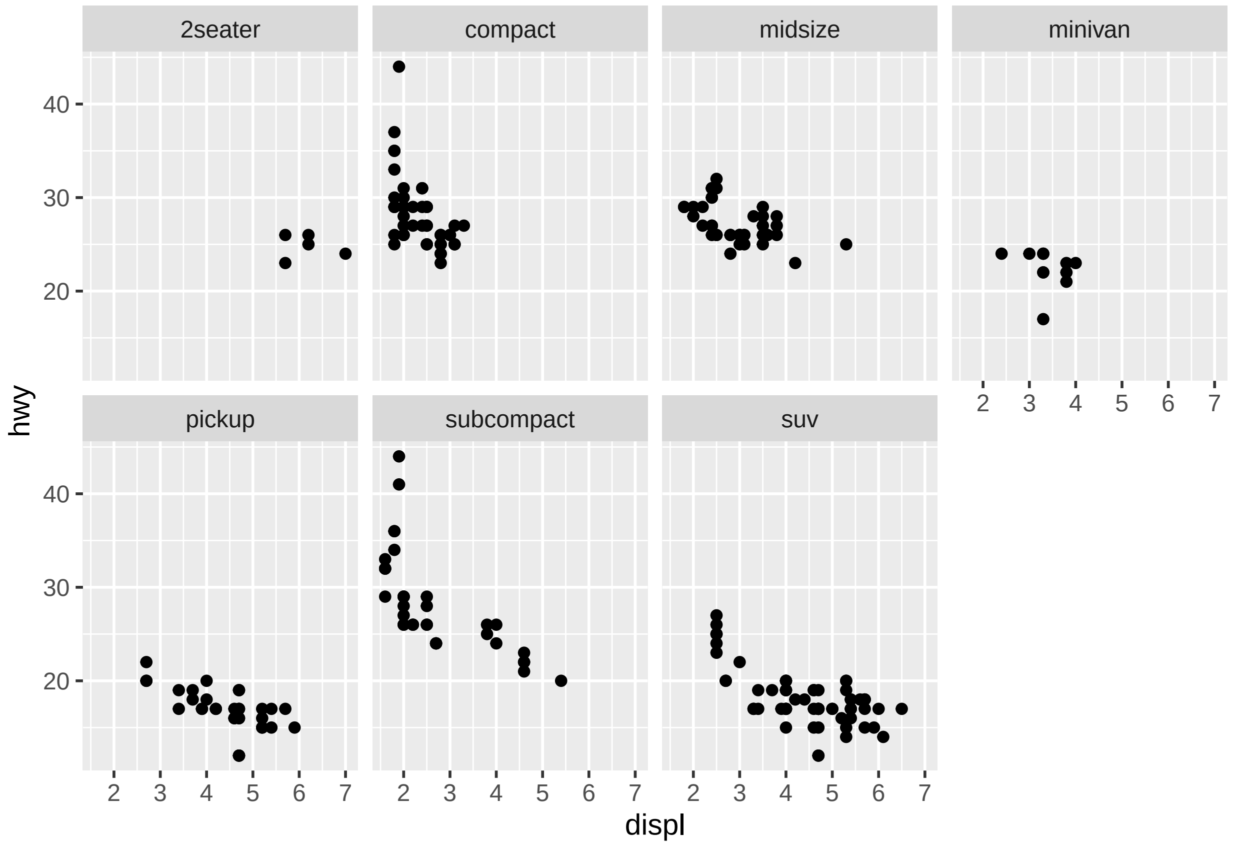
```r
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_grid(. ~ cyl)
```



Well, by the book the dot ignores the dimension when faceting. just remember 1 thing, facet_grid( y ~ x)

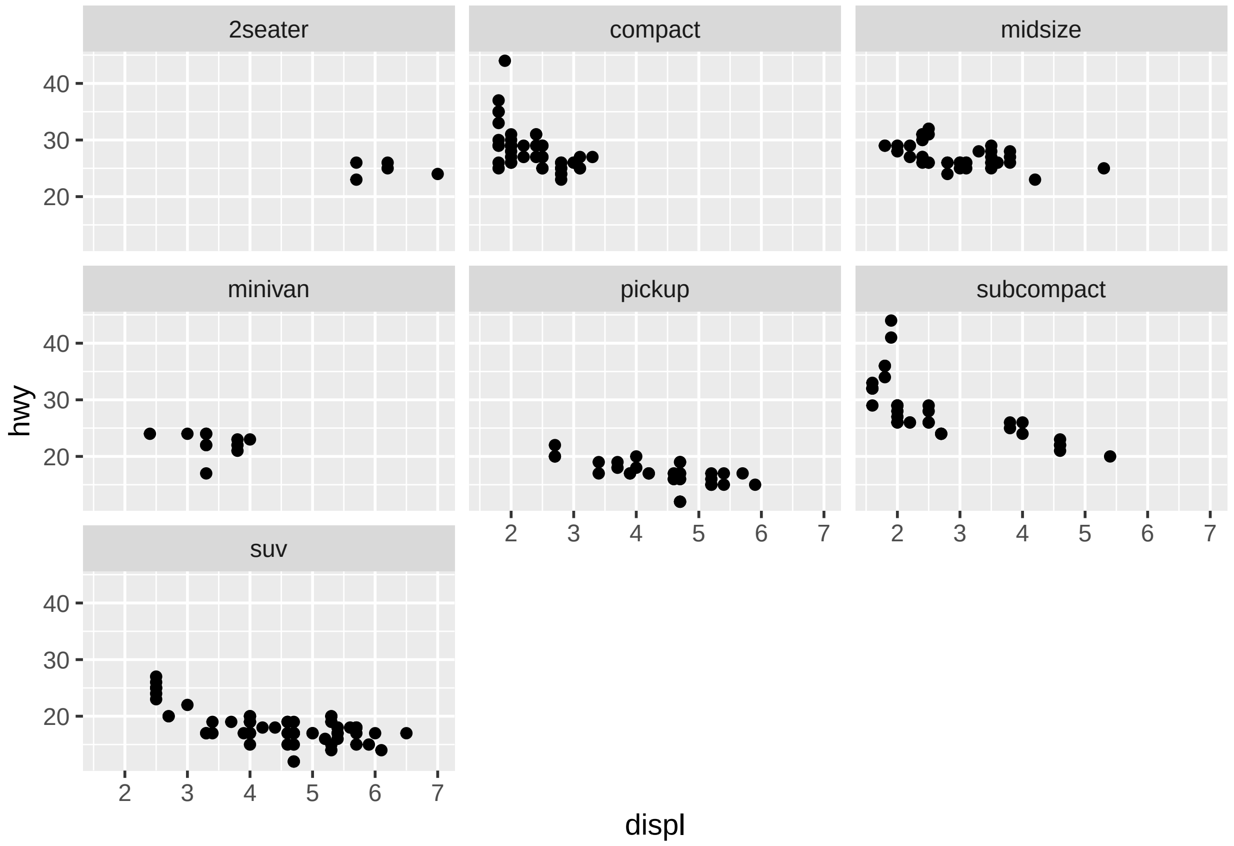
Q4 Take the first faceted plot in this section: ggplot(data = mpg) + geom_point(mapping = aes(x = displ, y = hwy)) + facet_wrap(~ class, nrow = 2) What are the advantages to using faceting instead of the color aesthetic? What are the disadvantages? How might the balance change if you had a larger dataset?

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_wrap(~ class, nrow = 2)
```



obviously, by using facet_wrap it is more readable.

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_wrap(~ class, nrow = 3, ncol = 3)
```



We can define how many rows and cols, also for facet_grid, we do not have to identify how many cols and rows.

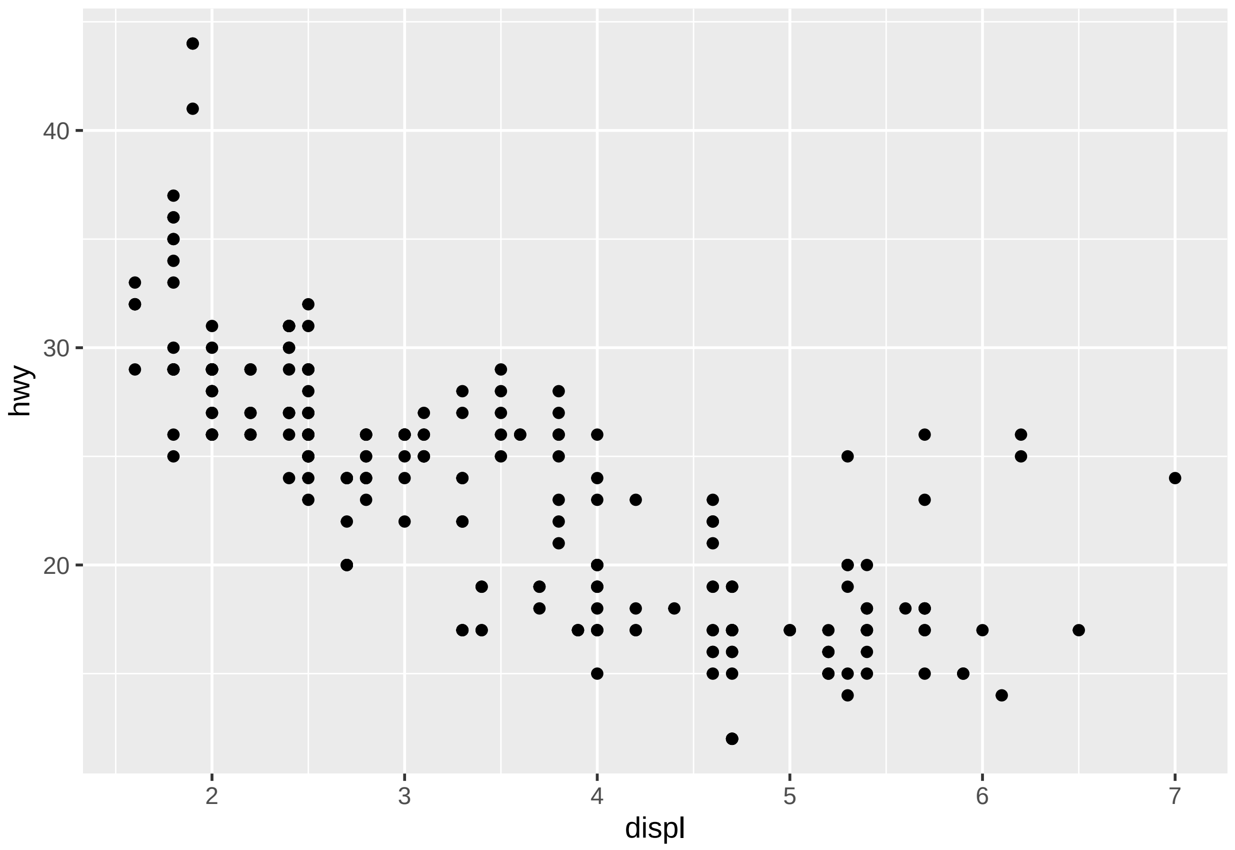
Q6 When using facet_grid() you should usually put the variable with more unique levels in the columns. Why? Since we usually have more spacing for cols than rows

Geometric Objects

there are different types of geom. point, smooth, line, boxplot and so on…

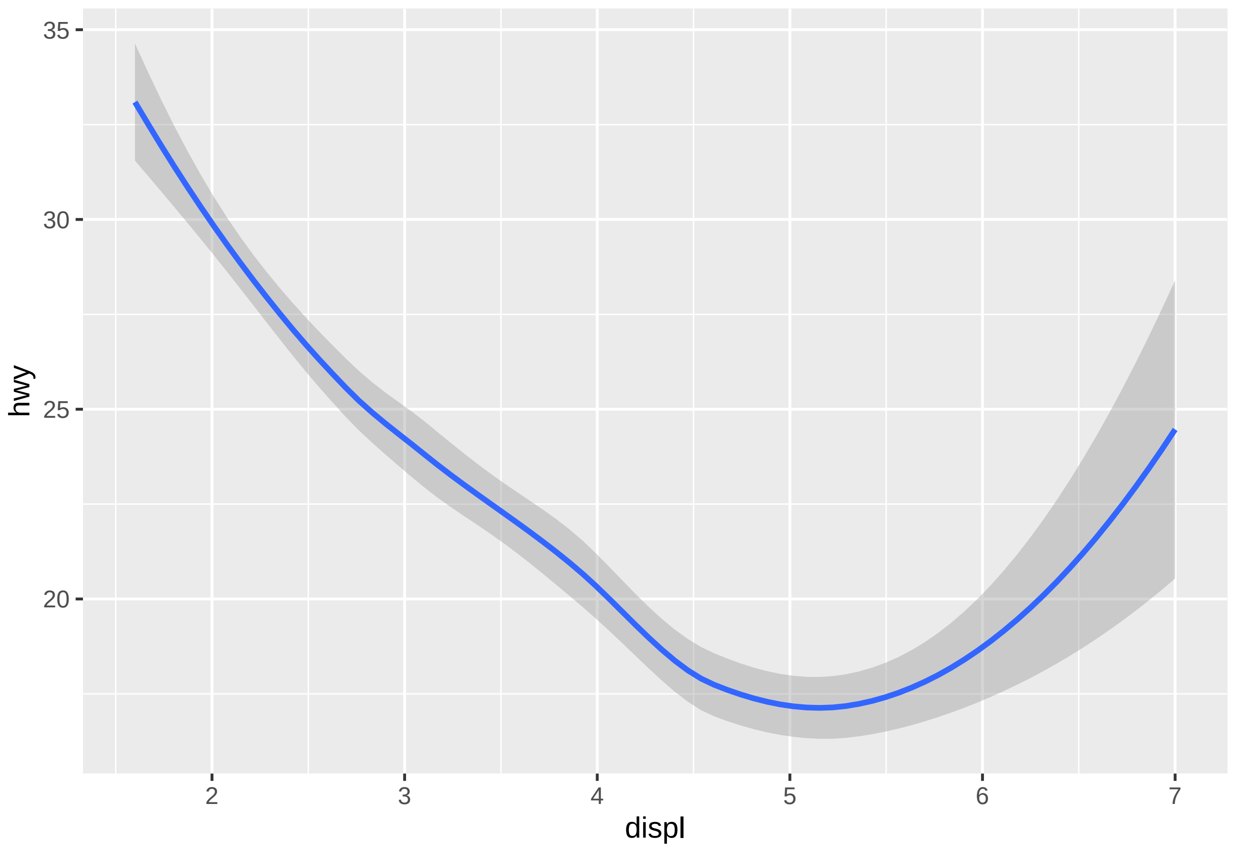
let display the basic

```r
ggplot(mpg) +
  geom_point(aes(displ, hwy))
```



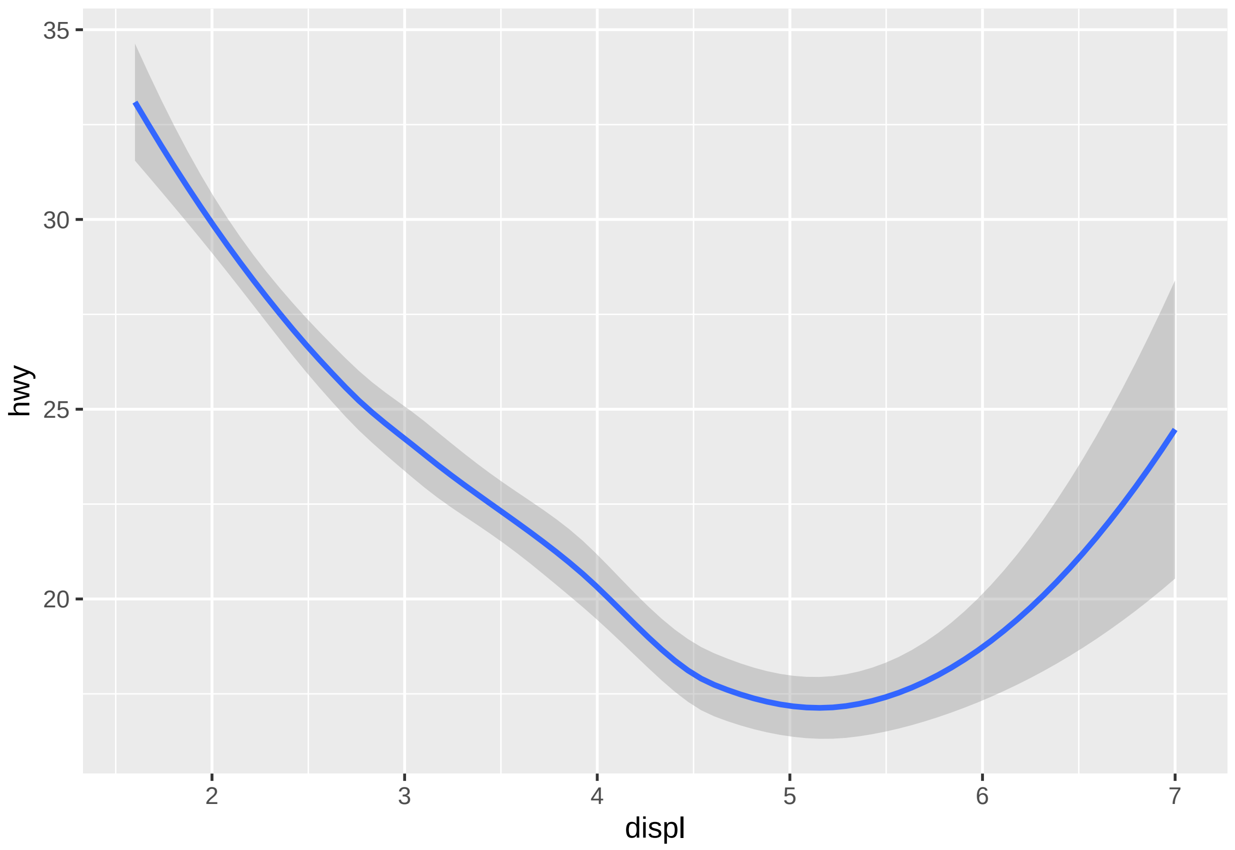
```
ggplot(mpg) +
  geom_smooth(aes(displ,hwy))
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



```r
ggplot(mpg) +
  geom_smooth(aes(displ,hwy))
```

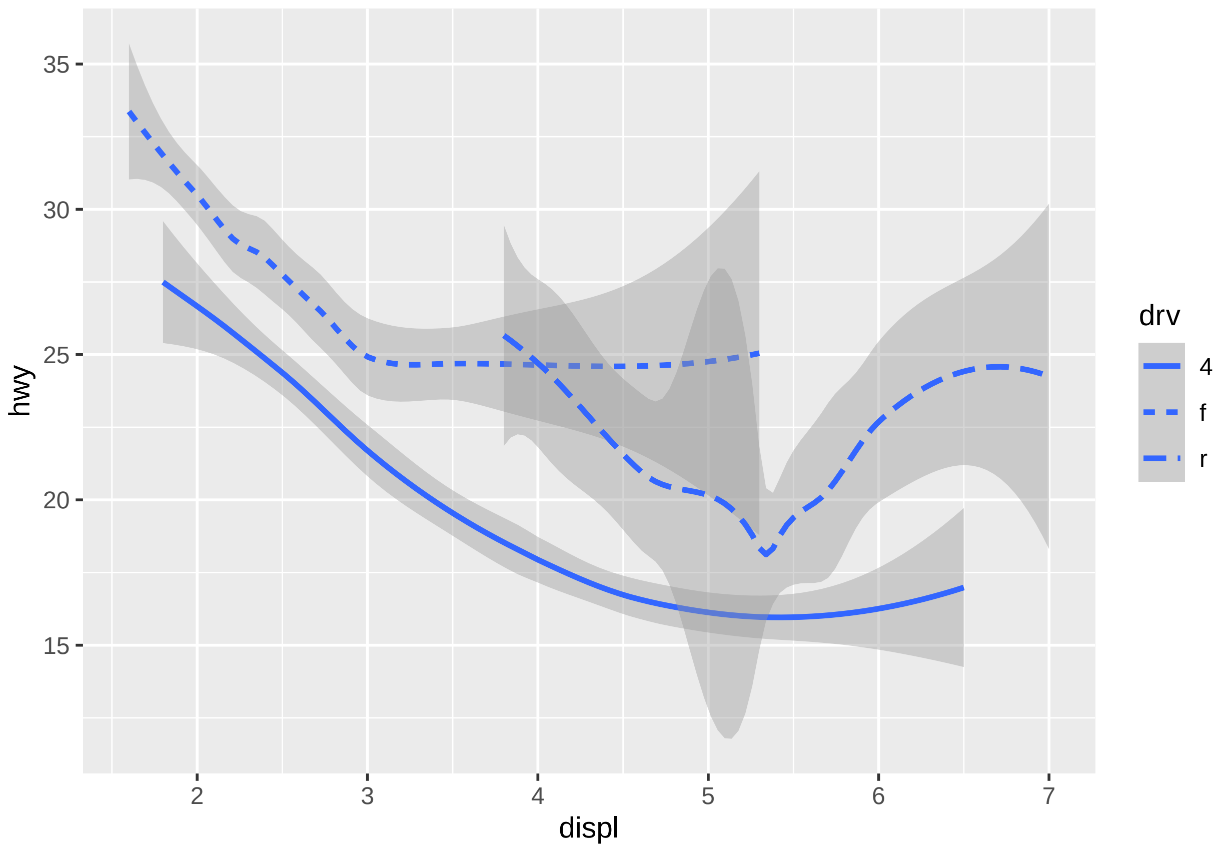
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Here, geom_smooth() separates the cars into three lines based on drv, 4, f, r.

```
ggplot(mpg) +
  geom_smooth(aes(displ, hwy, linetype = drv))
```

```
## ‘geom_smooth()‘ using method = ’loess’ and formula ’y ~ x’
```

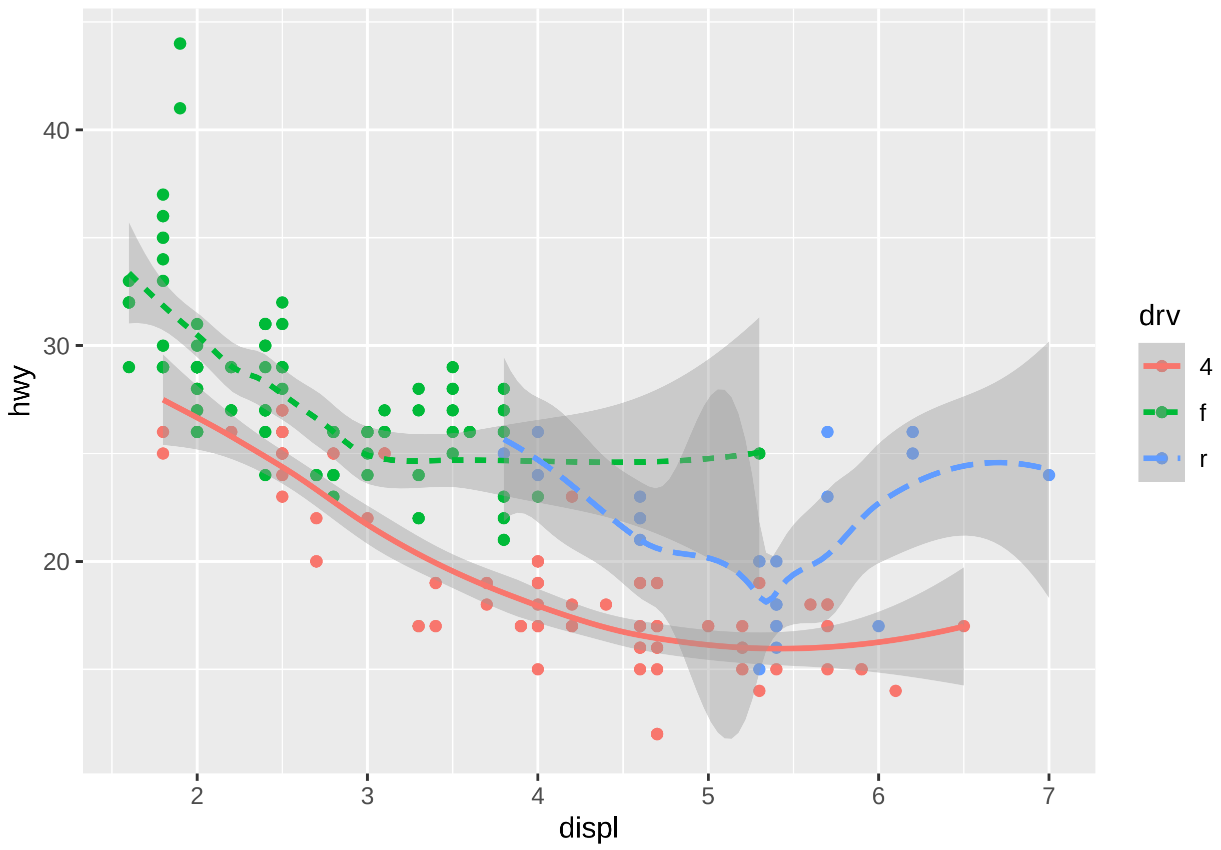


I

can also use the same X and Y values and plot the points and smooth.

```r
ggplot(mpg) +
  geom_point(aes(displ, hwy, color = drv)) +
  geom_smooth(aes(displ, hwy, linetype = drv, color = drv))
```

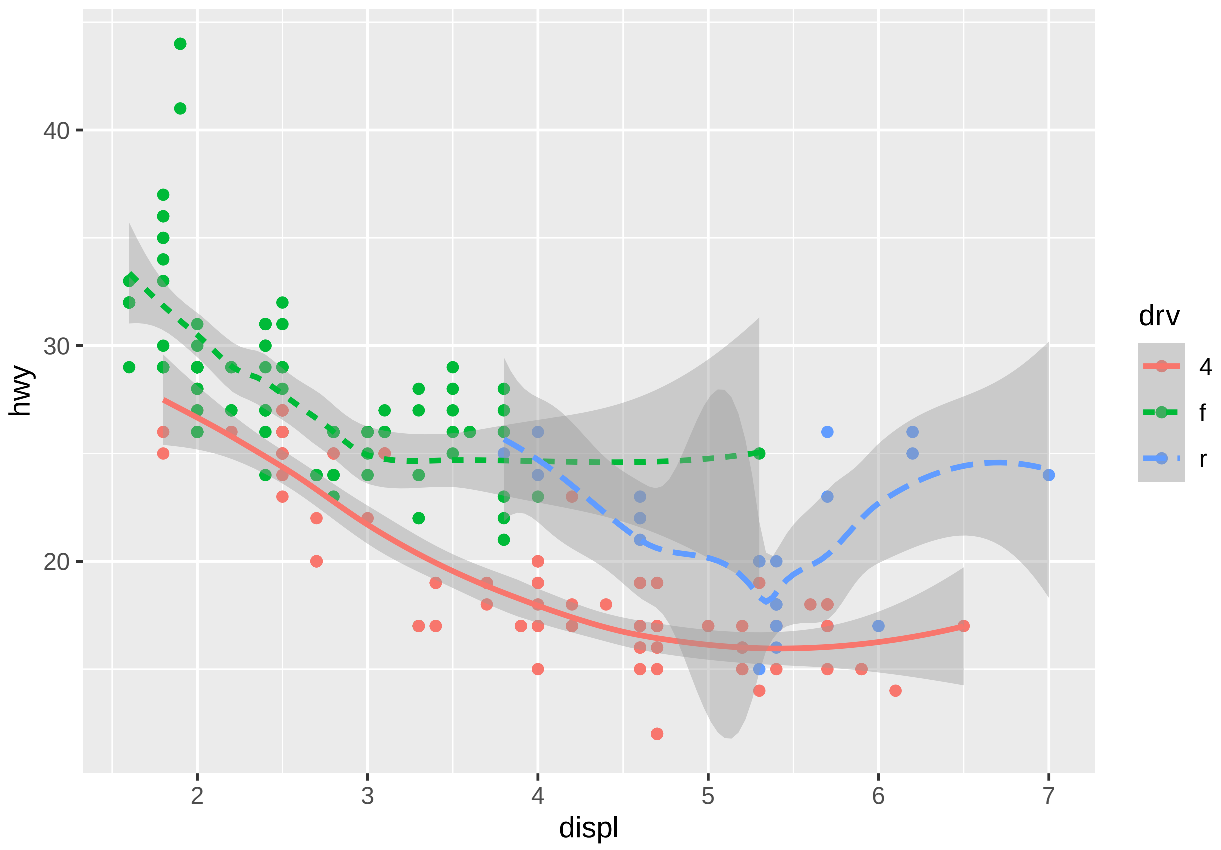
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



From the above code you can see that i have duplicated code. displ, hwy @ two places. so we can imporve it by the following method

```
ggplot(data = mpg, aes(displ, hwy, color = drv)) +
  geom_point() +
  geom_smooth(aes(linetype = drv))
```

```
## ‘geom_smooth()‘ using method = ’loess’ and formula ’y ~ x’
```



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

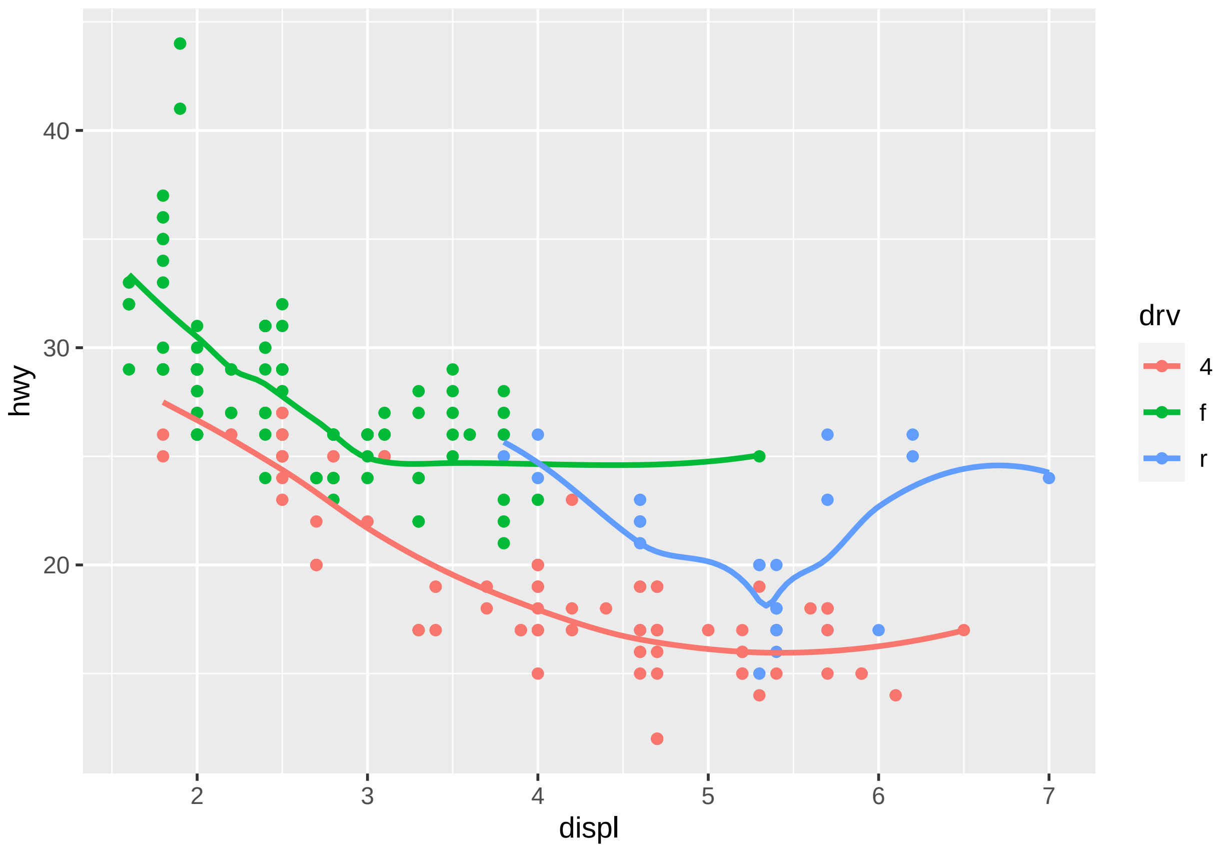
geom_line() geom_boxplot() geom_histogram() geom_area()

2) Run this code in your head and predict what the output will look like. Then, run the code in R and check your predictions:

color points and line without confident interval

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = drv)) +
  geom_point() +
  geom_smooth(se = FALSE)
```

```
## ‘geom_smooth()‘ using method = ’loess’ and formula ’y ~ x’
```



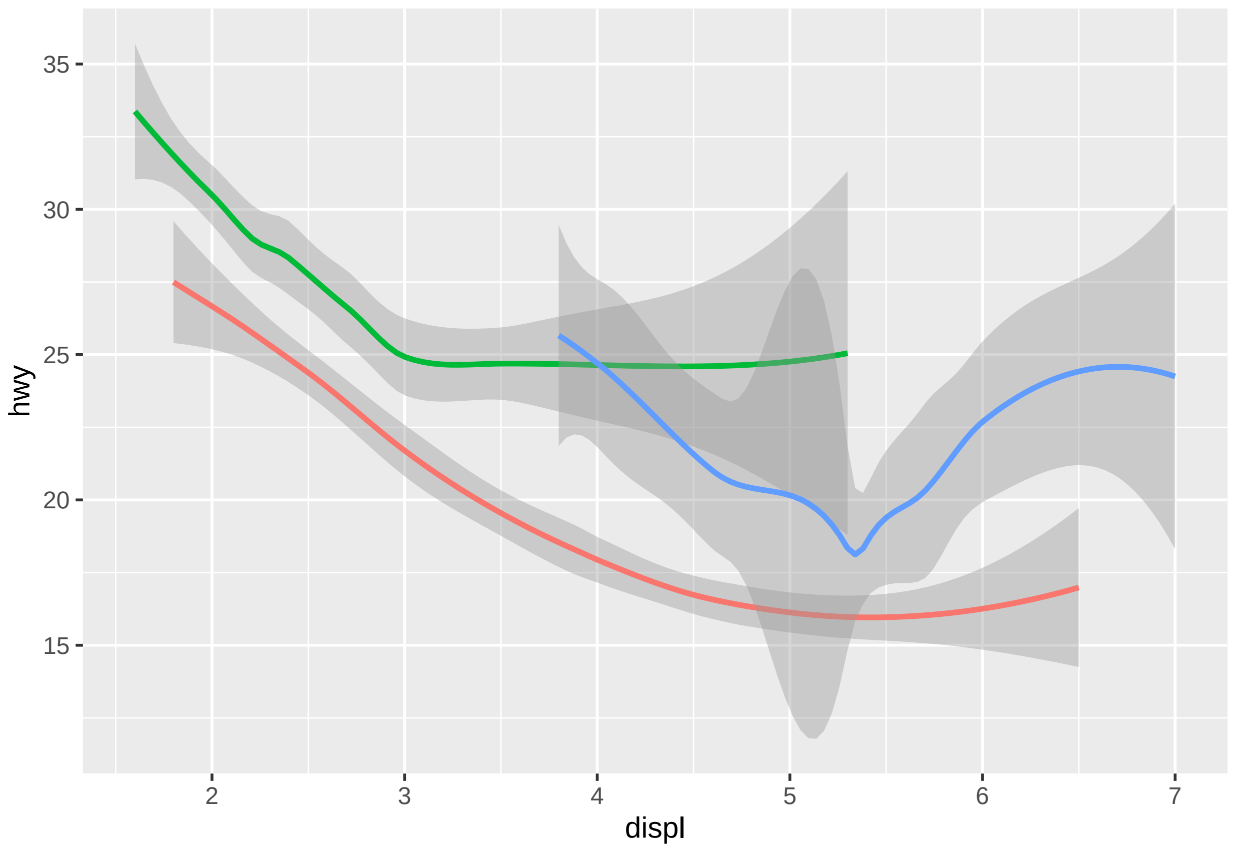
3)

What does show.legend = FALSE do? What happens if you remove it? Why do you think I used it earlier in the chapter?

show.legend = F will remove the DRV info box from the plot and the graph is actually slightly larger.

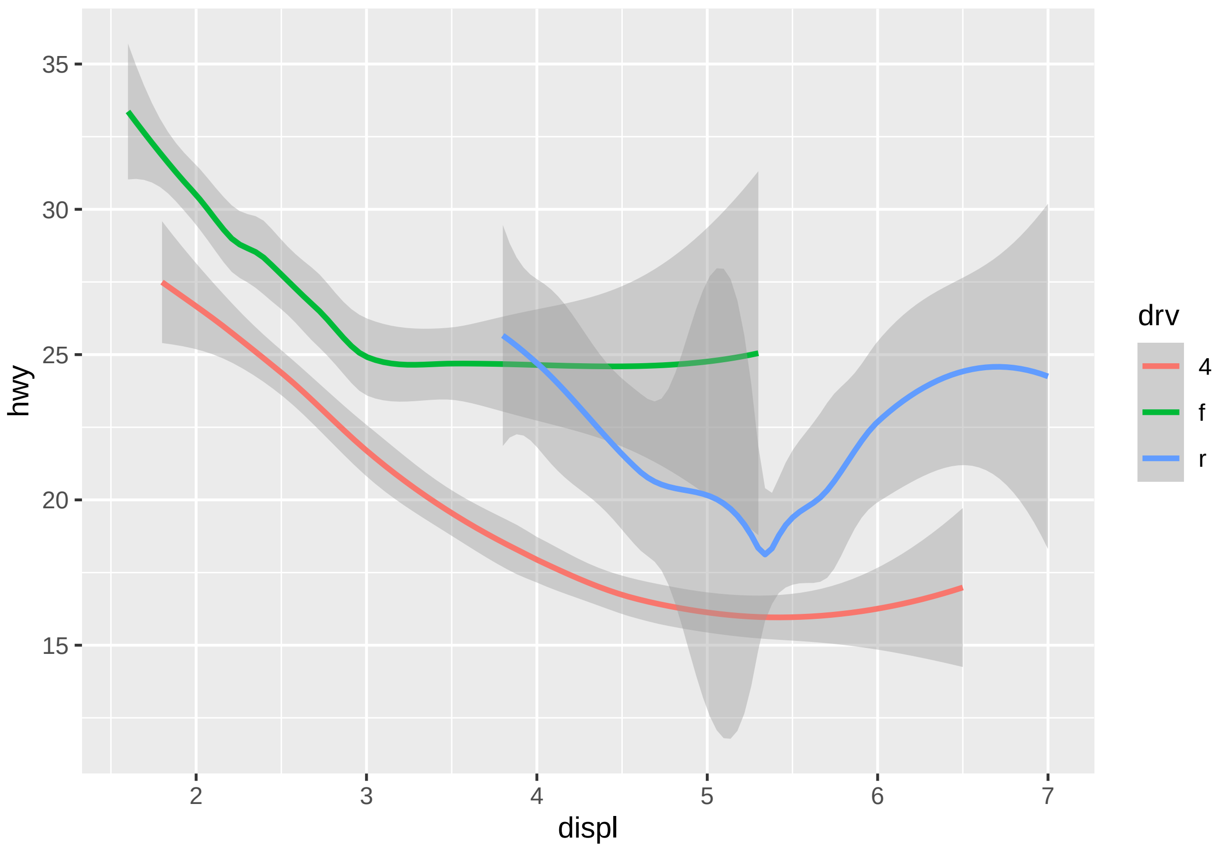
```
ggplot(data = mpg) +
  geom_smooth(
    mapping = aes(x = displ, y = hwy, colour = drv),
    show.legend = FALSE)
```

```
## ‘geom_smooth()‘ using method = ’loess’ and formula ’y ~ x’
```



```r
ggplot(data = mpg) +
  geom_smooth(
    mapping = aes(x = displ, y = hwy, colour = drv),
    show.legend = T)
```

```
## ‘geom_smooth()‘ using method = ’loess’ and formula ’y ~ x’
```



4. What does the se argument to geom_smooth() do? se from geom_smooth() is to add the standard error bands

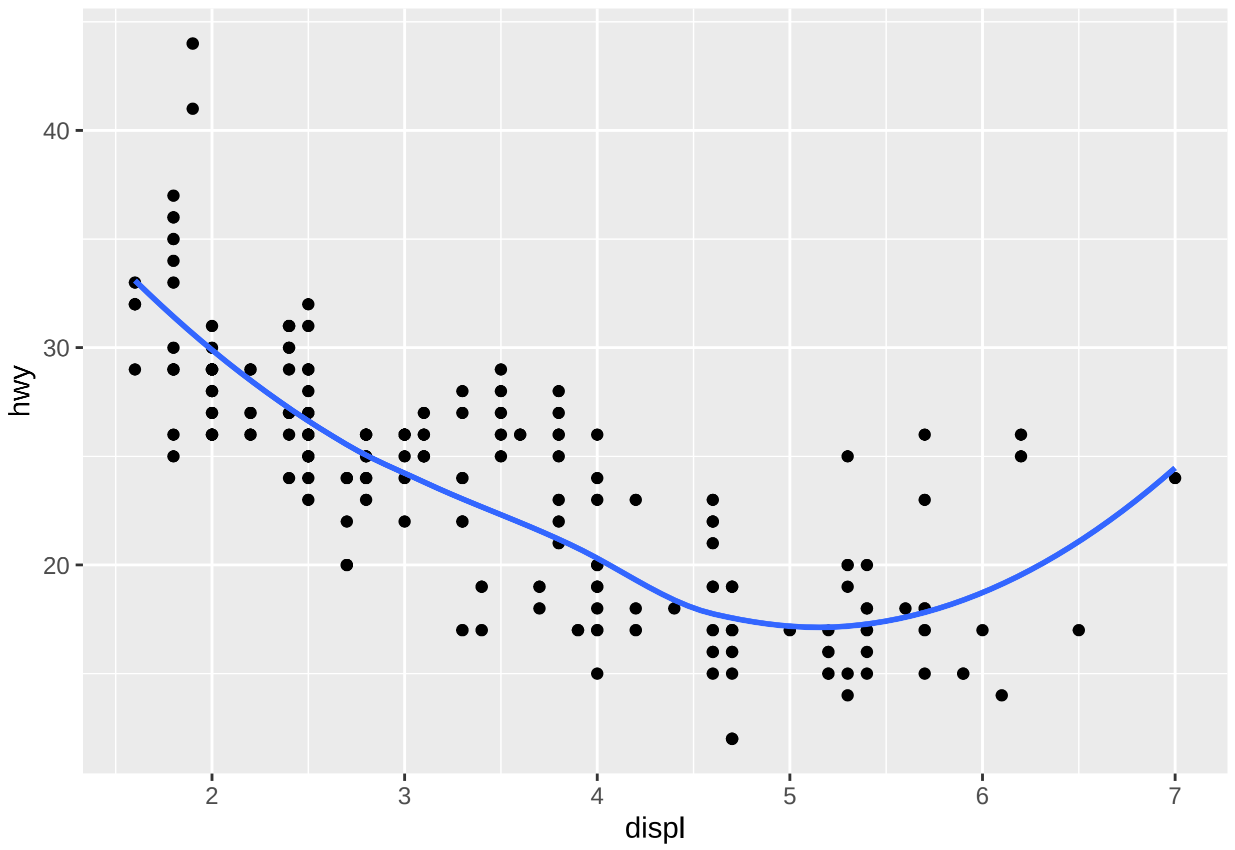
5. Will these two graphs look different? Why/why not? ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) + geom_point() + geom_smooth() ggplot() + geom_point( data = mpg, mapping = aes(x = displ, y = hwy) )+ geom_smooth( data = mpg, mapping = aes(x = displ, y = hwy) )

no difference.

6. Re-create the R code necessary to generate the following graphs. plot (1,1)

```r
ggplot(mpg, aes(displ,hwy)) +
  geom_point() +
  geom_smooth(se = F)
```

```
## ‘geom_smooth()‘ using method = ’loess’ and formula ’y ~ x’
```

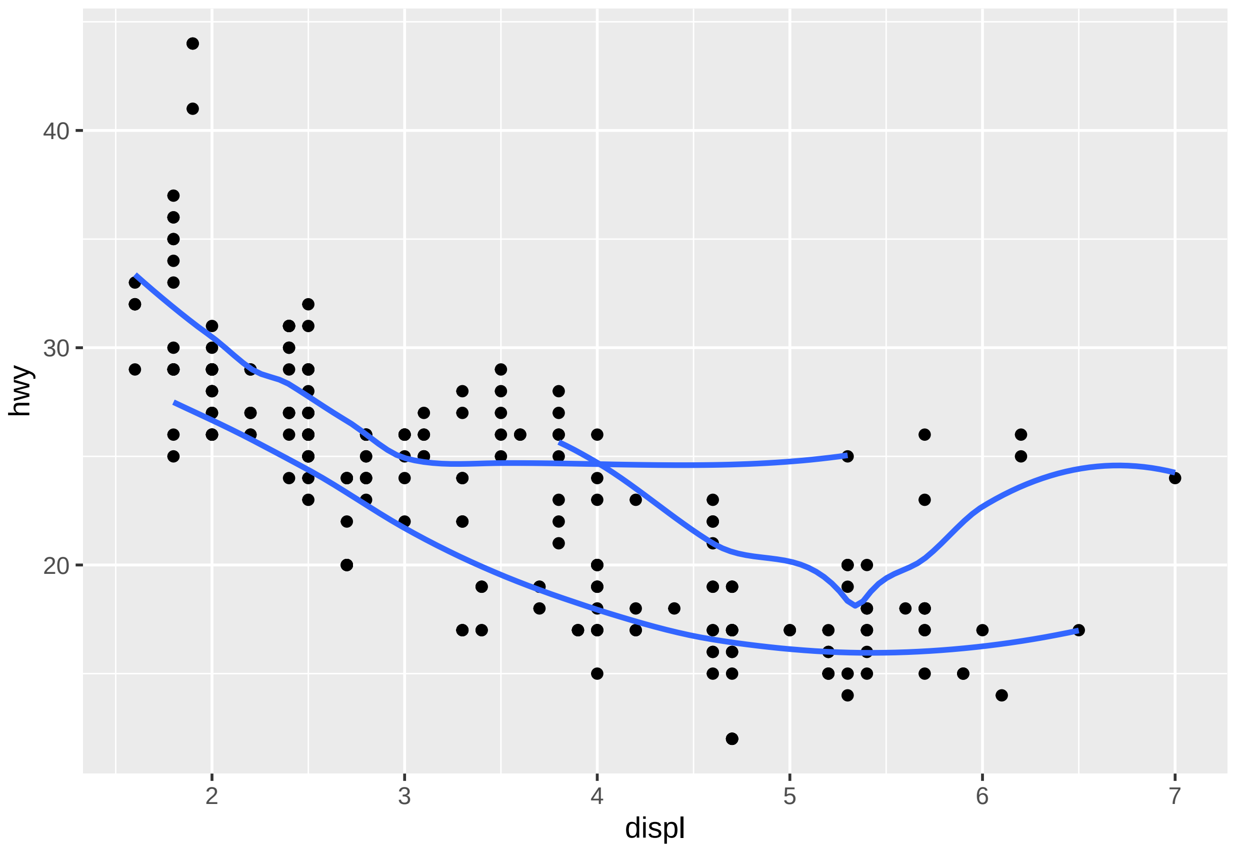


plot (1,2)

```
ggplot(mpg, aes(displ,hwy)) +
  geom_point() +
  geom_smooth(se = F, aes(line = drv))
```

```
## Warning: Ignoring unknown aesthetics: line
```

```
## ‘geom_smooth()‘ using method = ’loess’ and formula ’y ~ x’
```

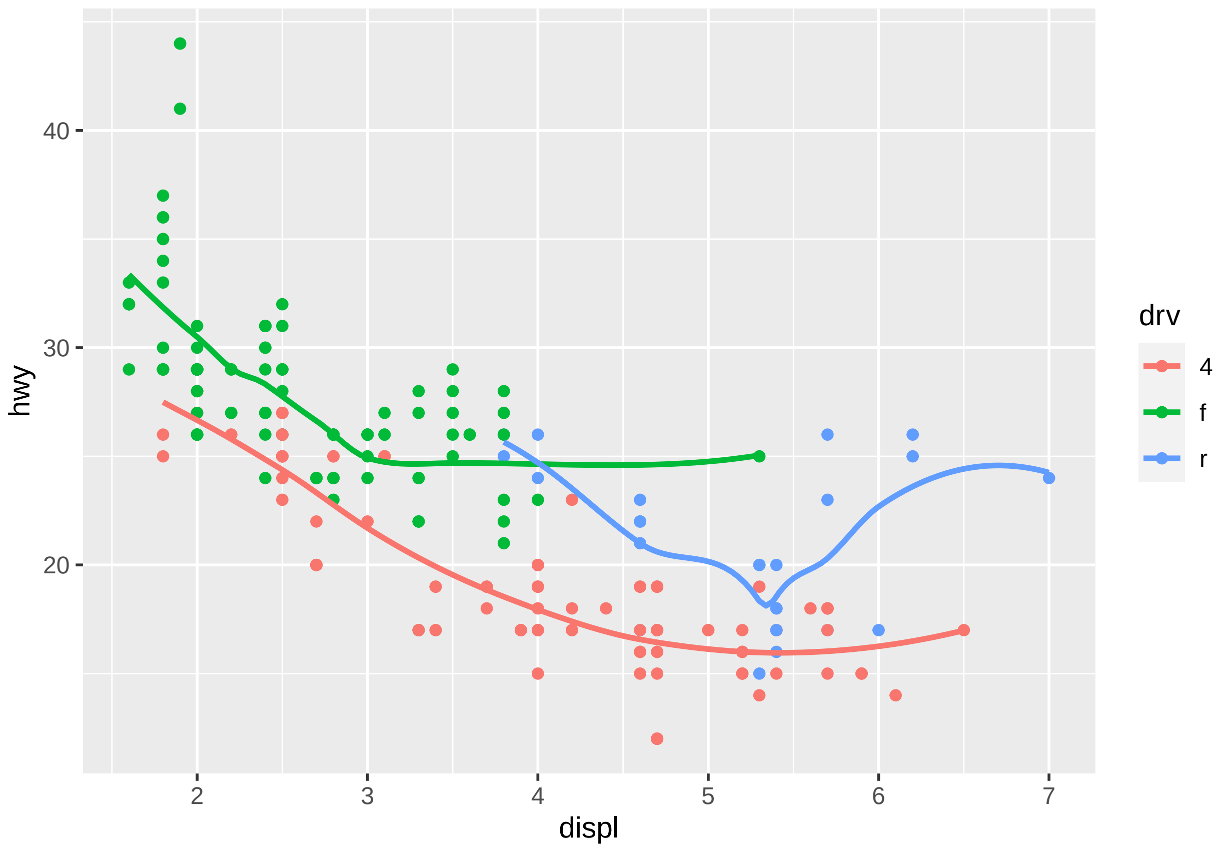


plot(2,1)

```r
ggplot(mpg, aes(displ,hwy, color = drv)) +
  geom_point() +
  geom_smooth(se = F, aes(line = drv))
```

```
## Warning: Ignoring unknown aesthetics: line
```

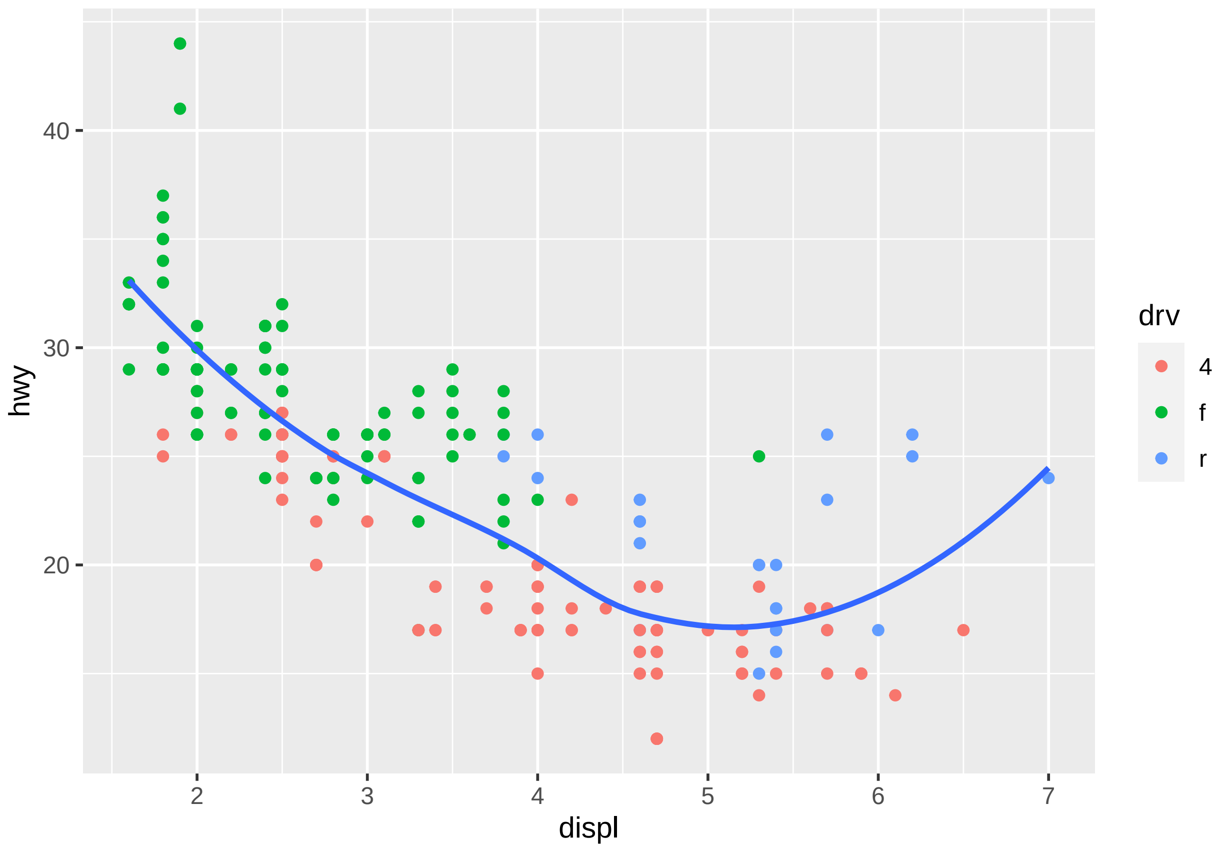
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



plot(2,2)

```r
ggplot(mpg, aes(displ, hwy)) +
  geom_point(aes(color = drv)) +
  geom_smooth(se = F)
```

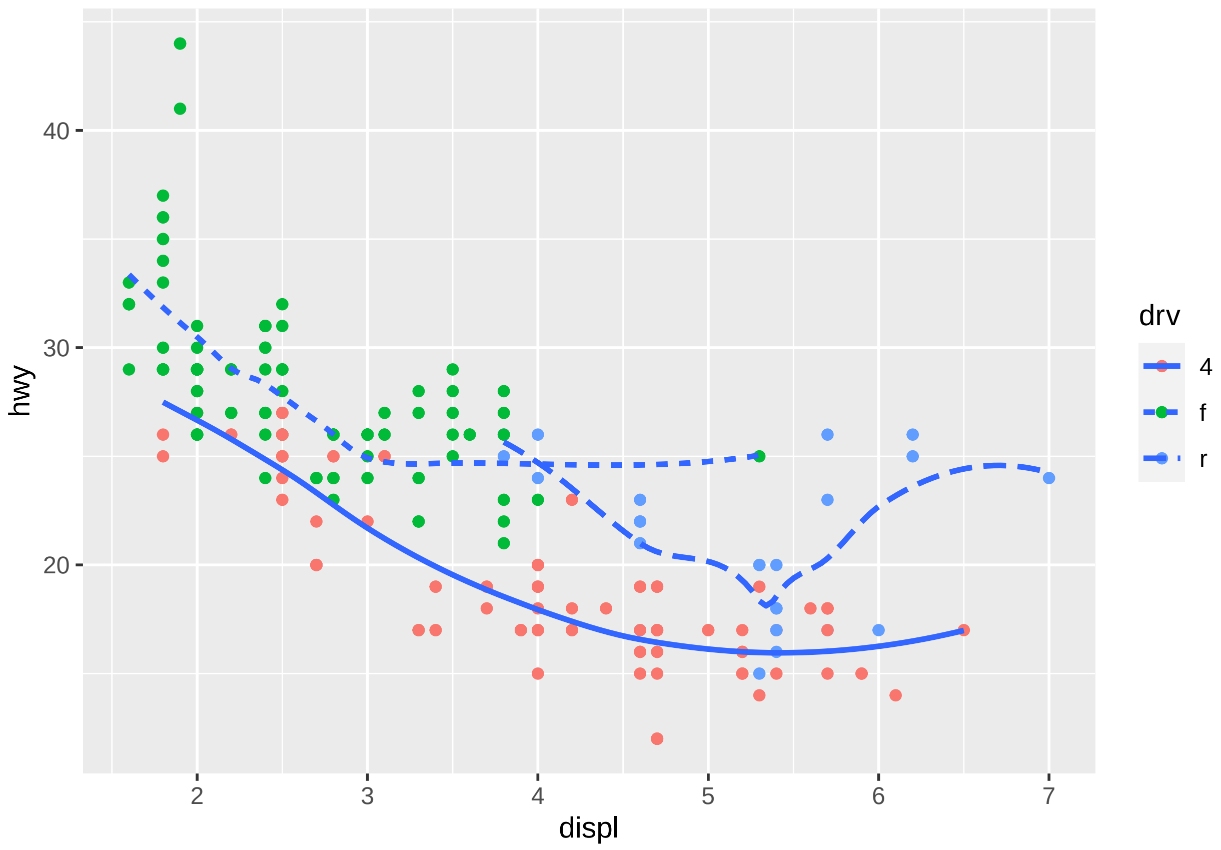
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



plot(3,1)

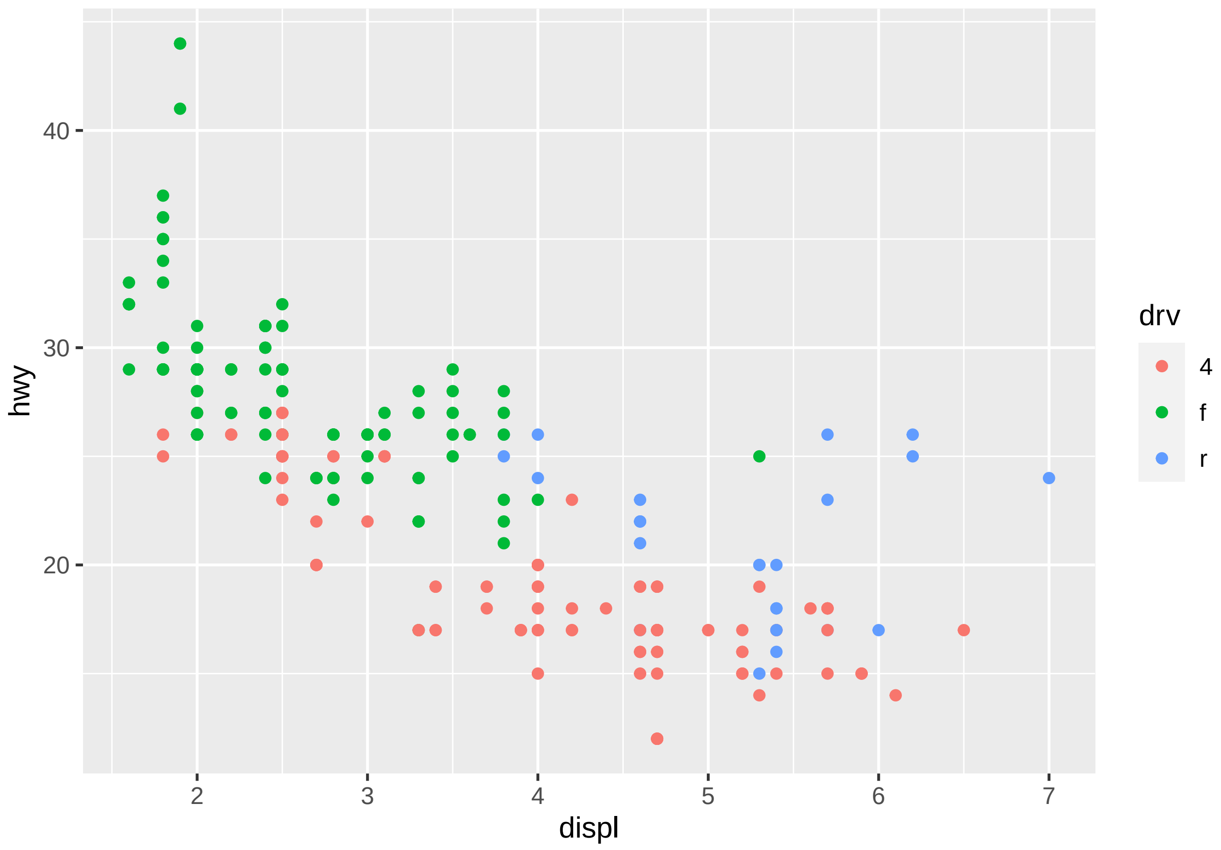
```r
ggplot(mpg, aes(displ, hwy)) +
  geom_point(aes(color = drv))+
  geom_smooth(se = F, aes(linetype = drv))
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



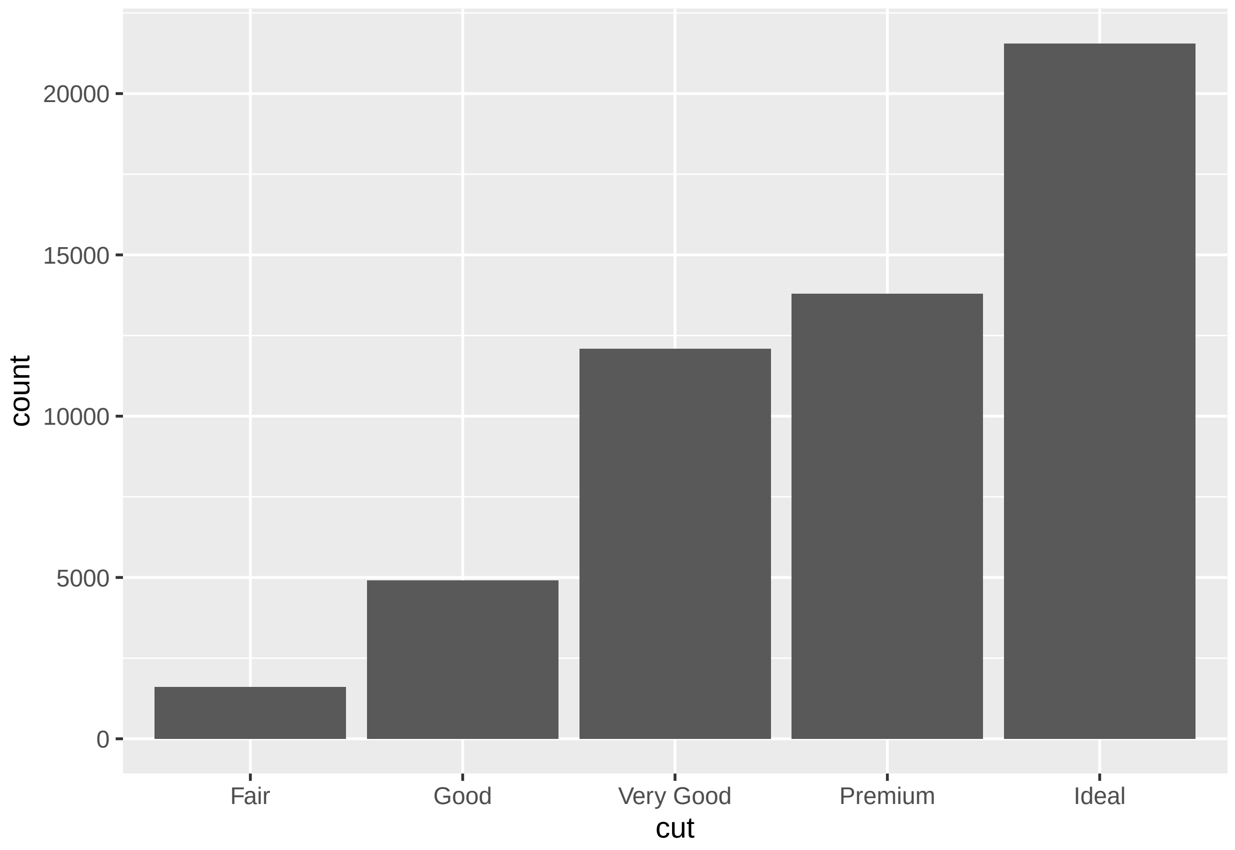
plot(3,2)

```r
ggplot(mpg, aes(displ, hwy, color = drv)) +
  geom_point()
```



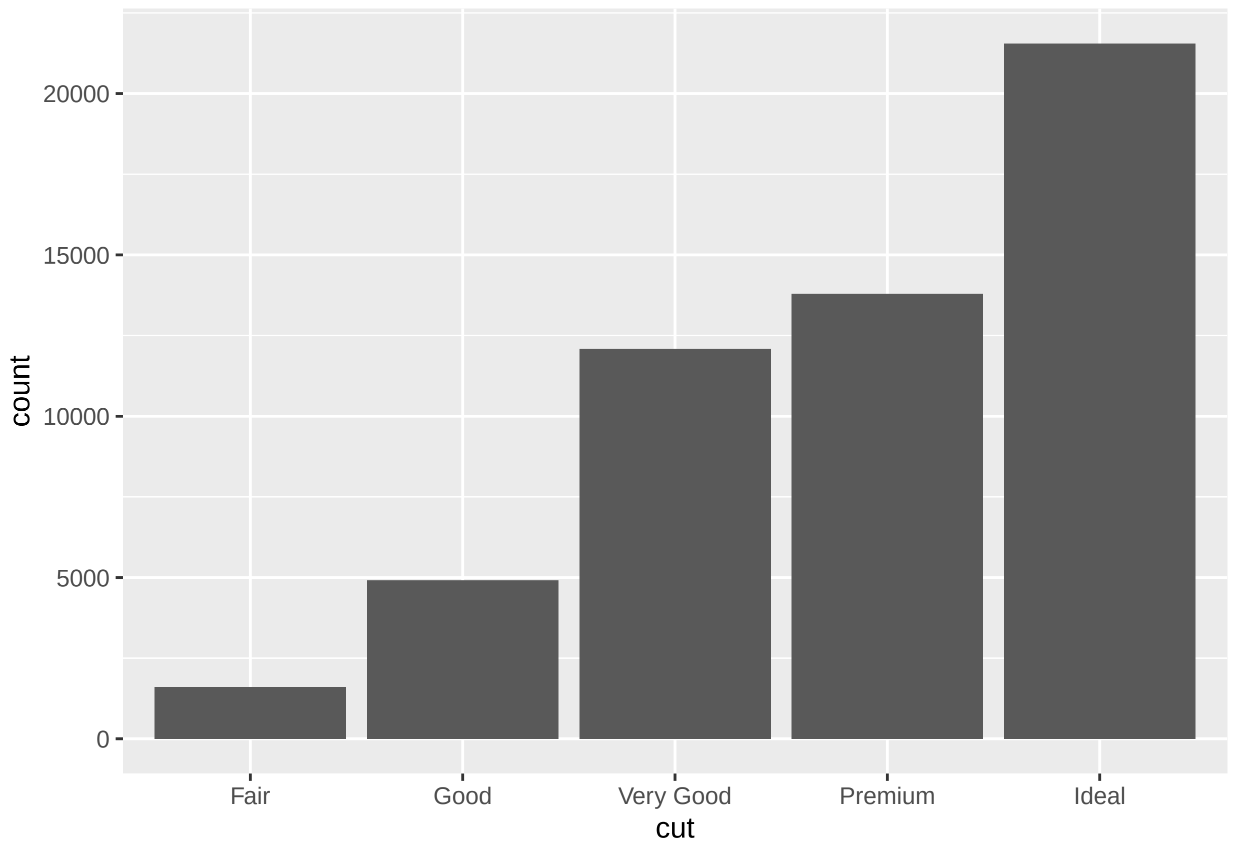
Statistical Transformations

```r
ggplot(diamonds, aes(cut)) +
  geom_bar()
```



i can reproduce this graph with stat_count

```
ggplot(diamonds, aes(cut)) +
  stat_count()
```

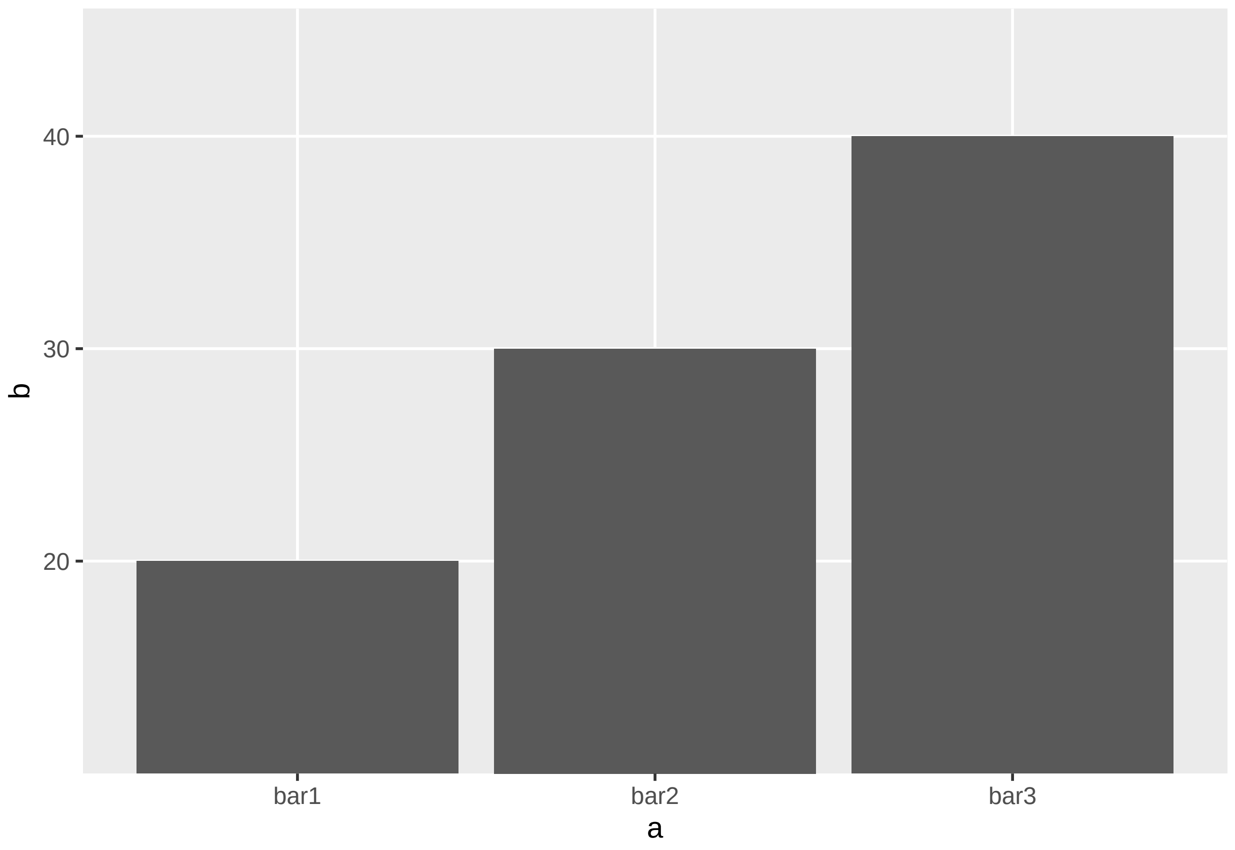


Let get it from data frame

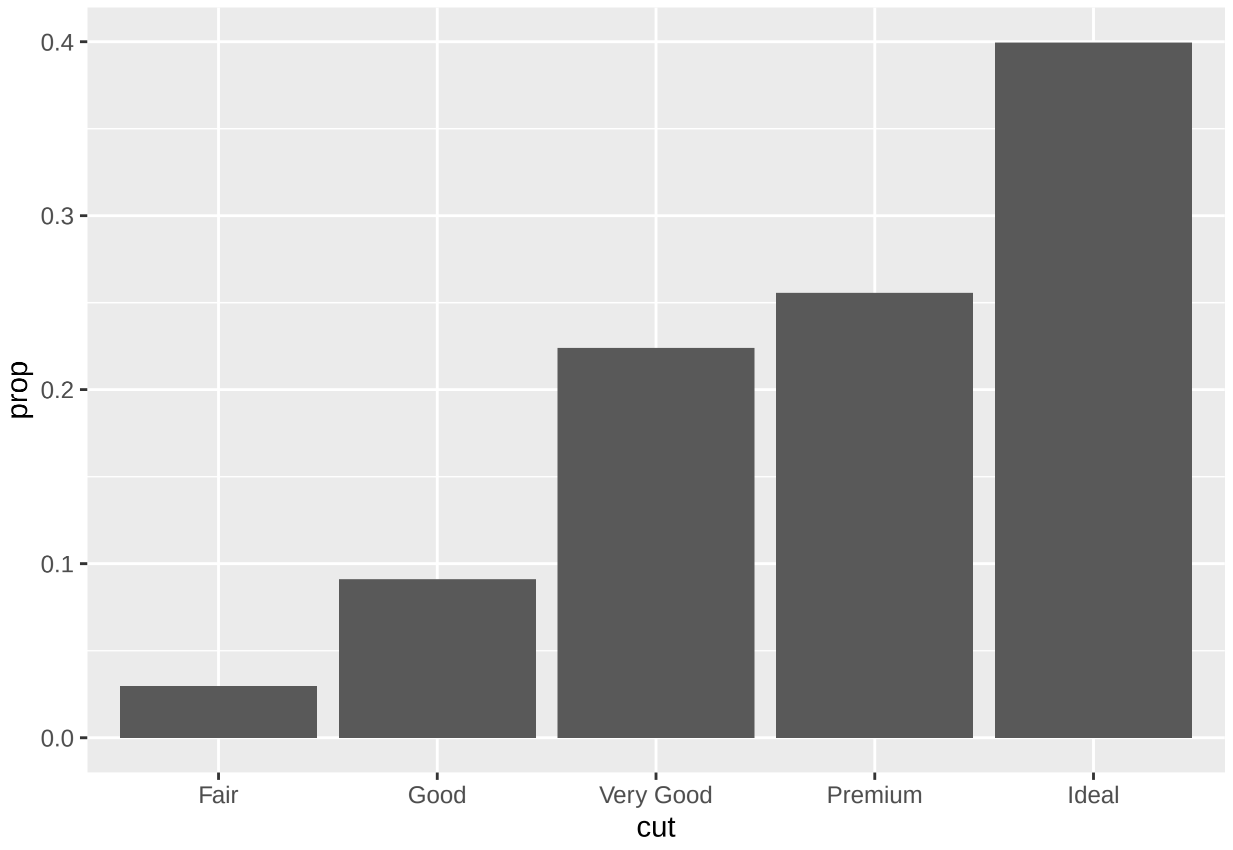
```
a <- c("bar1","bar2","bar3")
b <- c(20,30,40)
d <- cbind(a,b)
demo <- as.data.frame(d)
demo
```

```
##       a  b
## 1 bar1 20
## 2 bar2 30
## 3 bar3 40
```

```
ggplot(demo) +
  geom_bar(aes(x = `a`, y= `b`), stat = "identity")
```



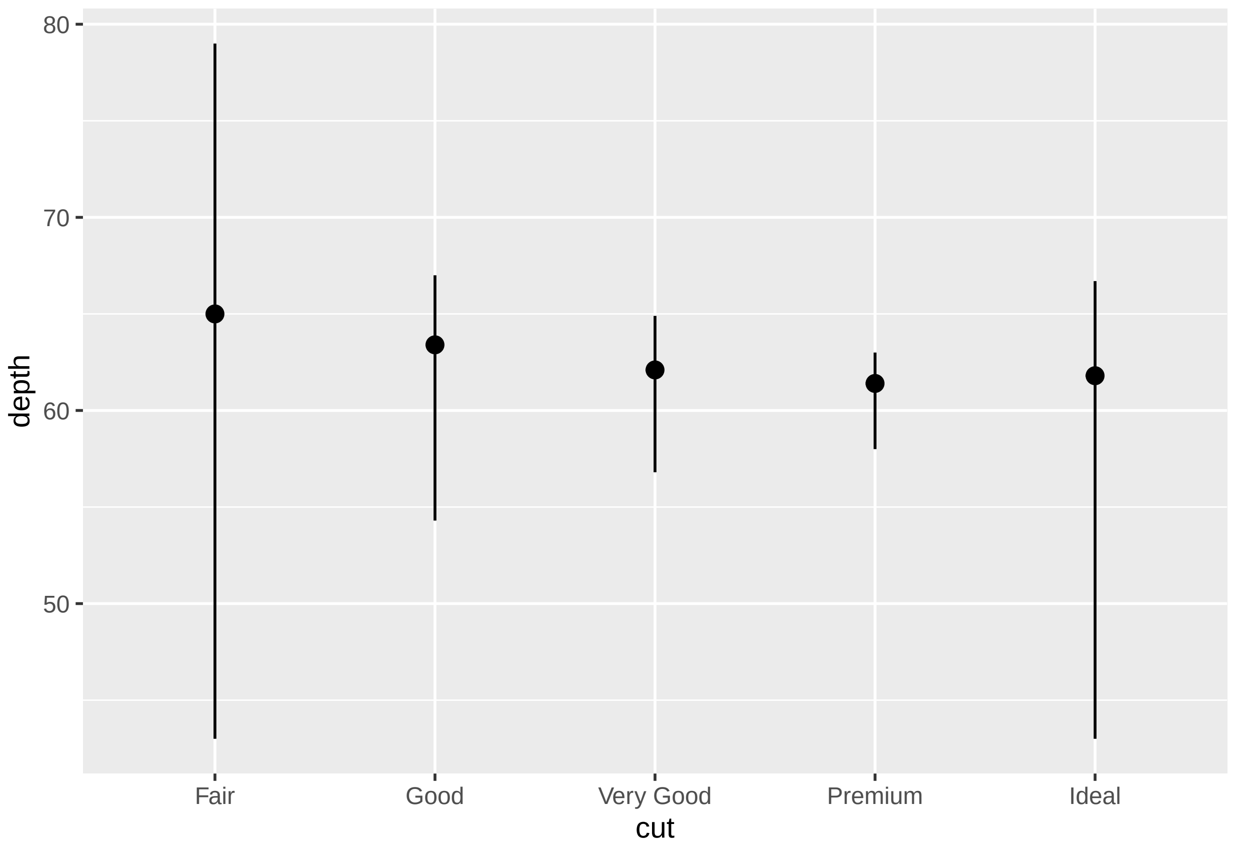
```r
ggplot(diamonds) +
  geom_bar(aes(cut, ..prop.., group = 1))
```



stat_summary will summarize y values for each unique x value by looking at the parameters, i believe you can guess what is going on. ymin = min, ymax = max, and y =median. this parameters are similar with the 5 number of summary like q1 and q3 with median and min with max.

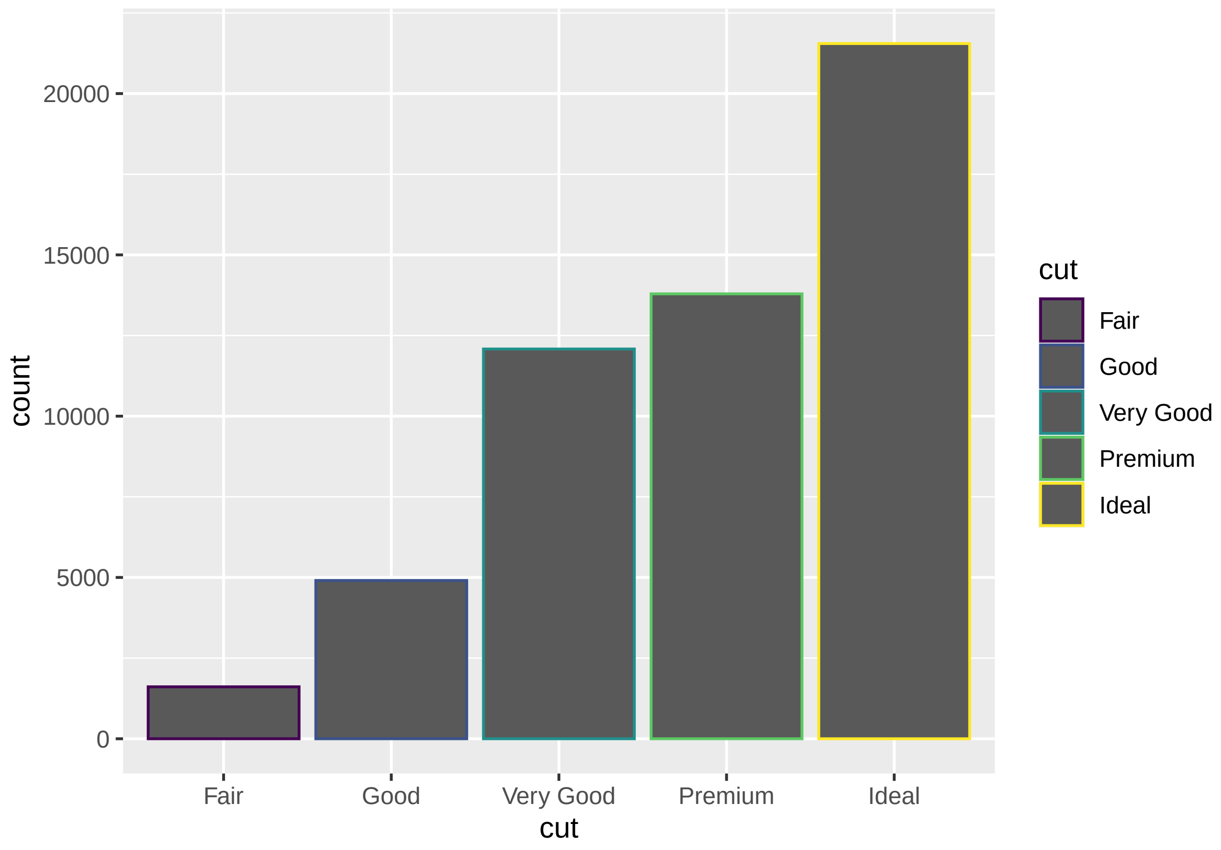
```r
ggplot(diamonds) +
  stat_summary(
    aes(cut, depth),
    fun.ymin = min,
    fun.ymax = max,
    fun.y = median
  )
```

```
## Warning: ‘fun.y‘ is deprecated. Use ‘fun‘ instead.

## Warning: ‘fun.ymin‘ is deprecated. Use ‘fun.min‘ instead.

## Warning: ‘fun.ymax‘ is deprecated. Use ‘fun.max‘ instead.
```



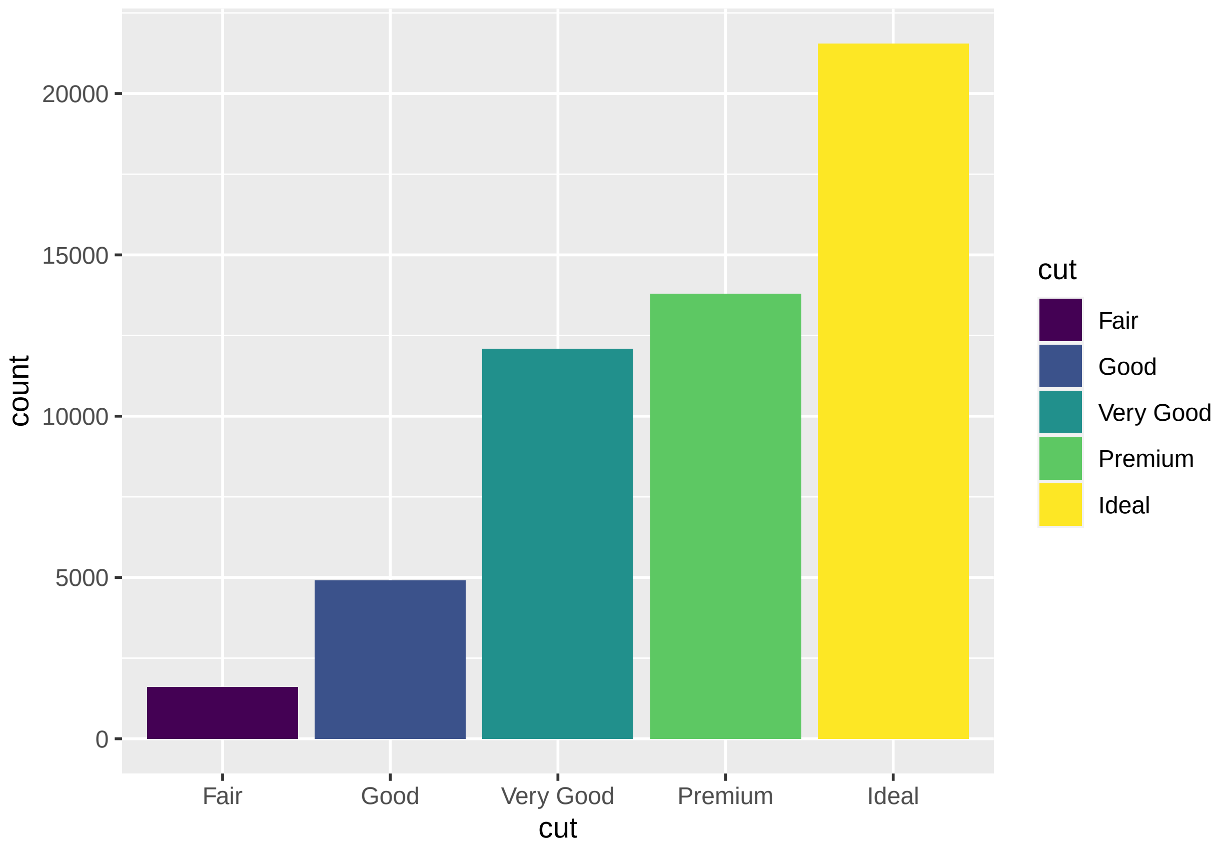
Position Adjustment

```r
ggplot(diamonds) +
  geom_bar(aes(cut, color = cut))
```



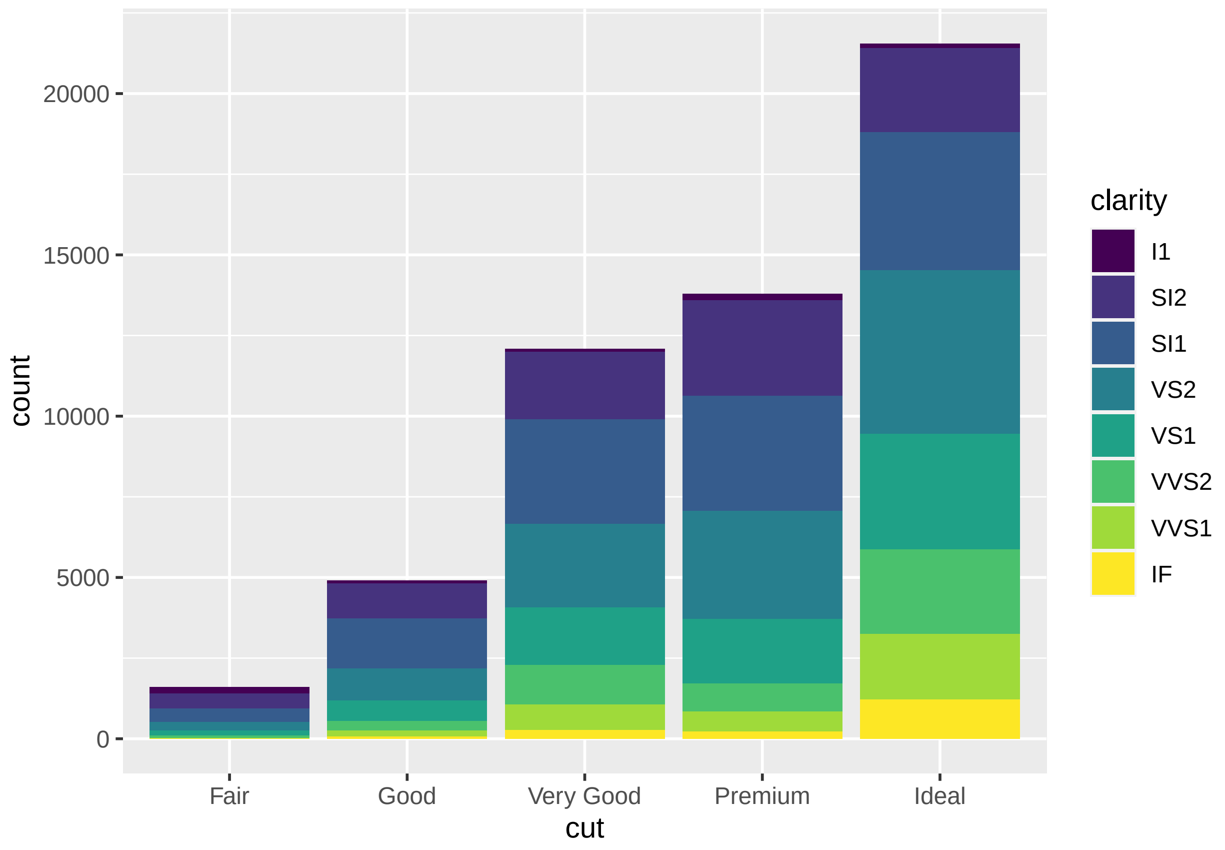
As you can see the color parameter is specifically for the border of the bar unlike the geom_point, the color will represent the actual dot

```
ggplot(diamonds) +
  geom_bar(aes(cut, fill = cut))
```



If you would like the entire bar with color, all we have to do is to use the fill function What about if we map the fill with different variable?

```
ggplot(diamonds) +
  geom_bar(aes(cut, fill = clarity))
```

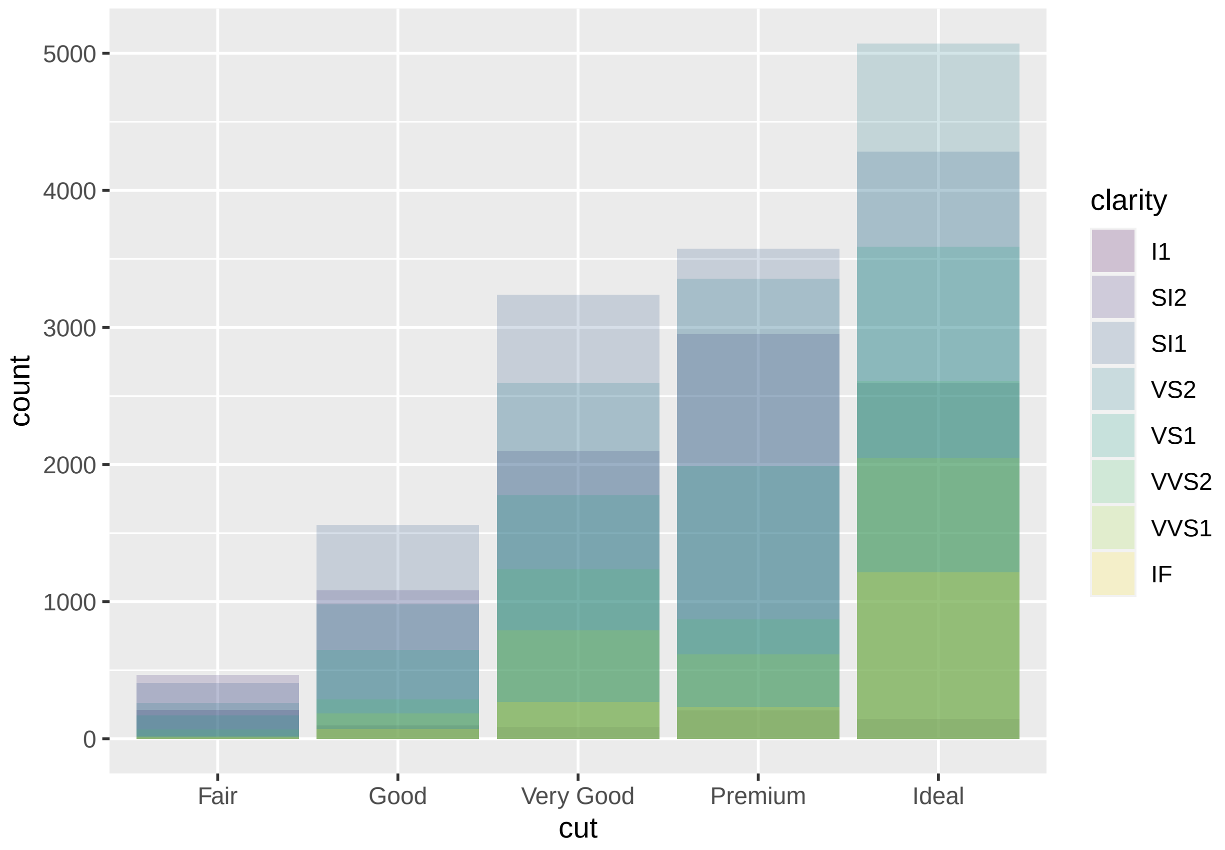


You will able to see there are different layers on a single bar. Because, from the data frame, clarity, we have multiple clarity with different cut, that's why a fair cut can have a high clarity which creates those layers in a bar.

There are three types of position we can use identity, fill, and dodge.

Let's try identity

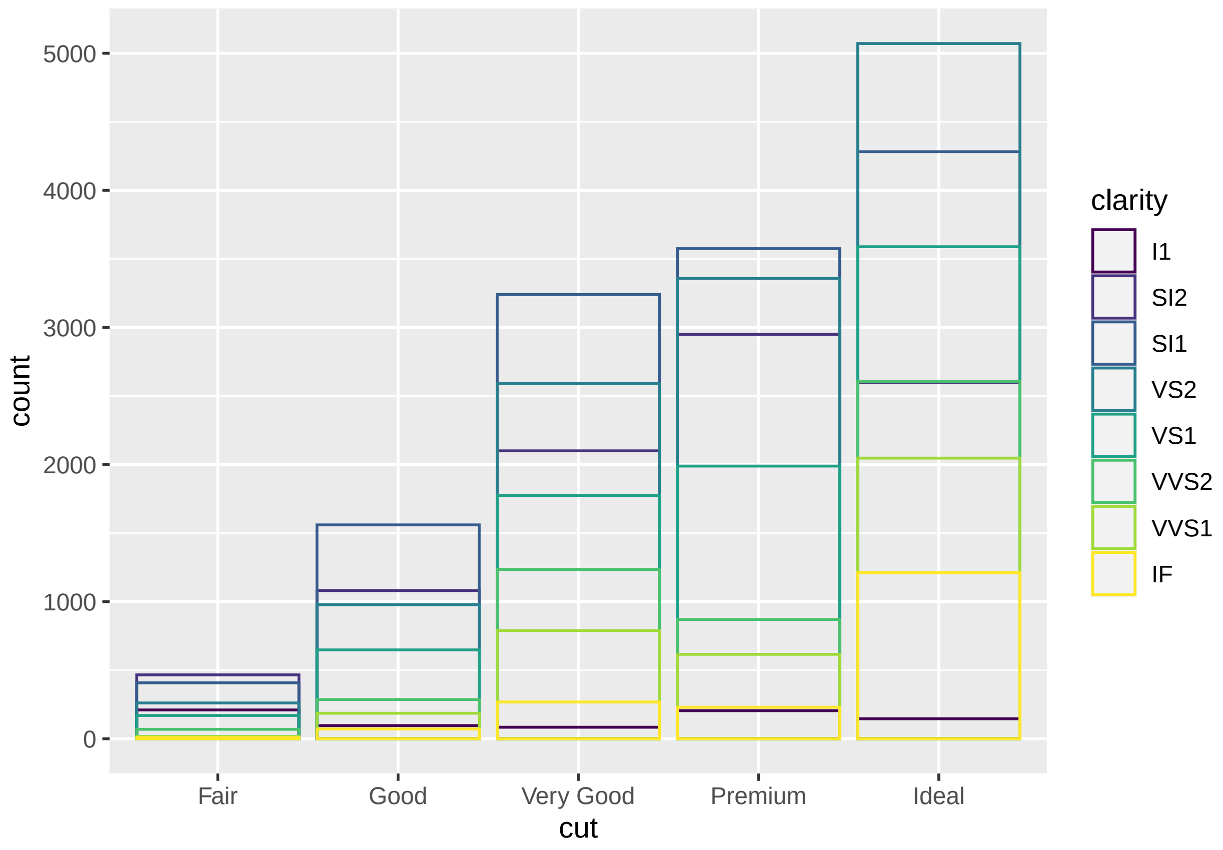
```
ggplot(diamonds, aes(cut, fill = clarity)) +
  geom_bar(alpha = 1/5, position = "identity")
```



alpha will control the transparency, it has the the effect as fill, fill = stack over each other.

Let's look at fill = NA without setting the alpha.

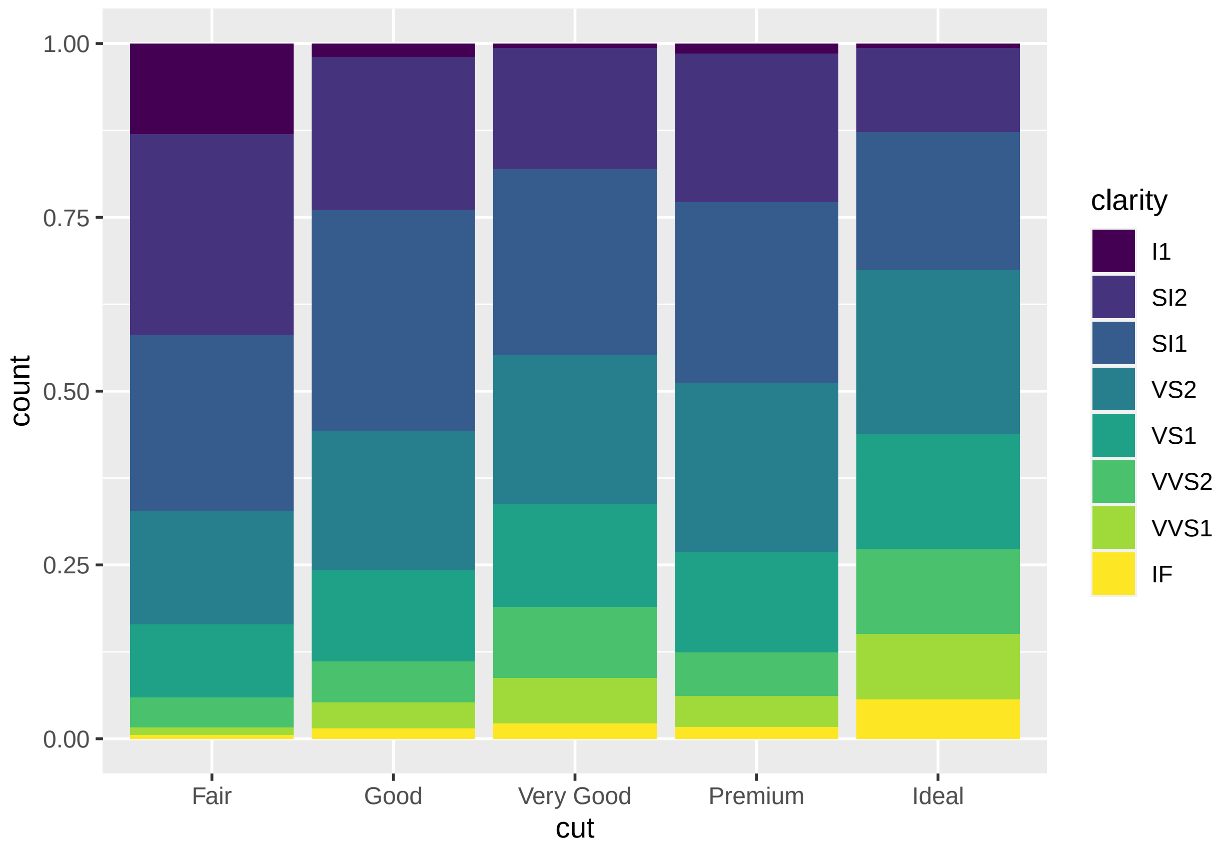
```r
ggplot(diamonds, aes(cut ,color = clarity)) +
  geom_bar(fill = NA, position = "identity")
```



This setting is simply unreadable.

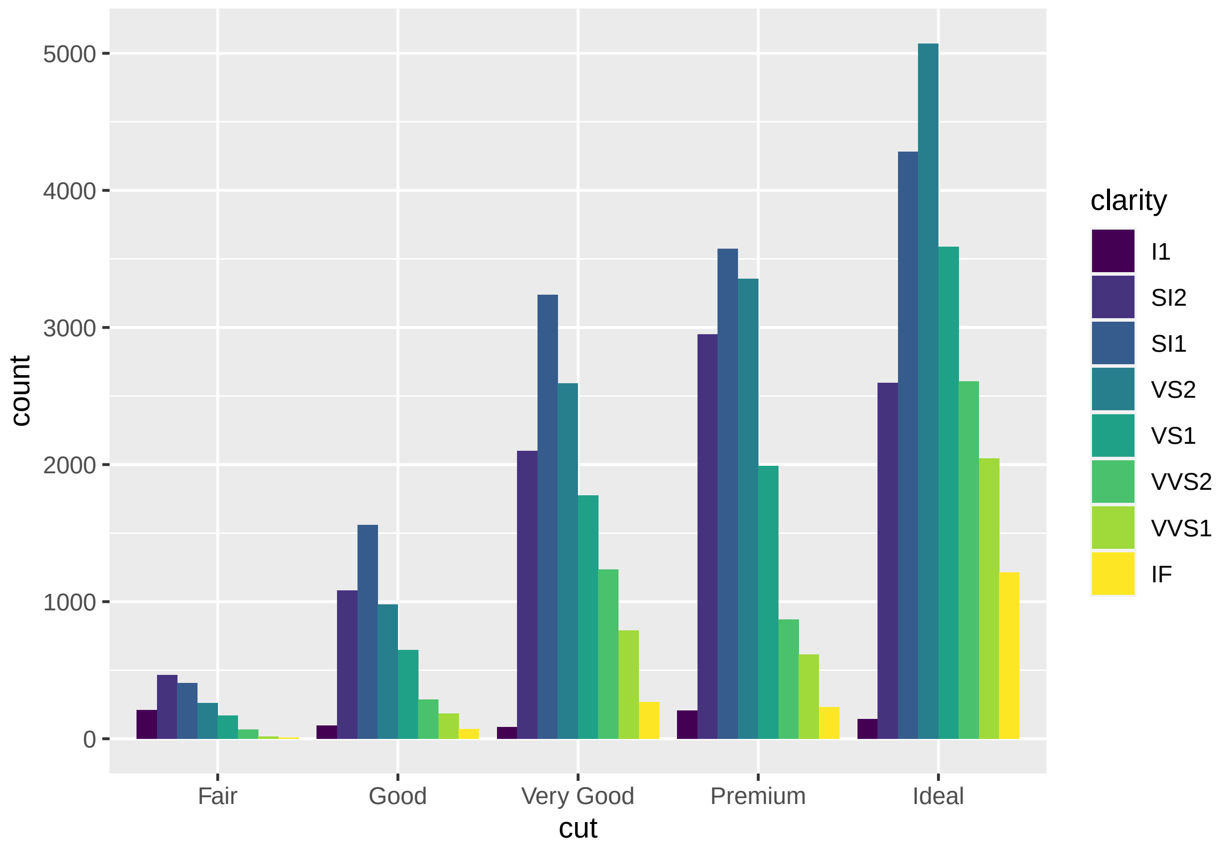
let's look at position = fill

```
ggplot(diamonds) +
  geom_bar( aes(cut, fill=clarity), position = "fill")
```



Lets take a look at position = dodge

```r
ggplot(diamonds) +
  geom_bar(aes(cut, fill =clarity), position = "dodge")
```



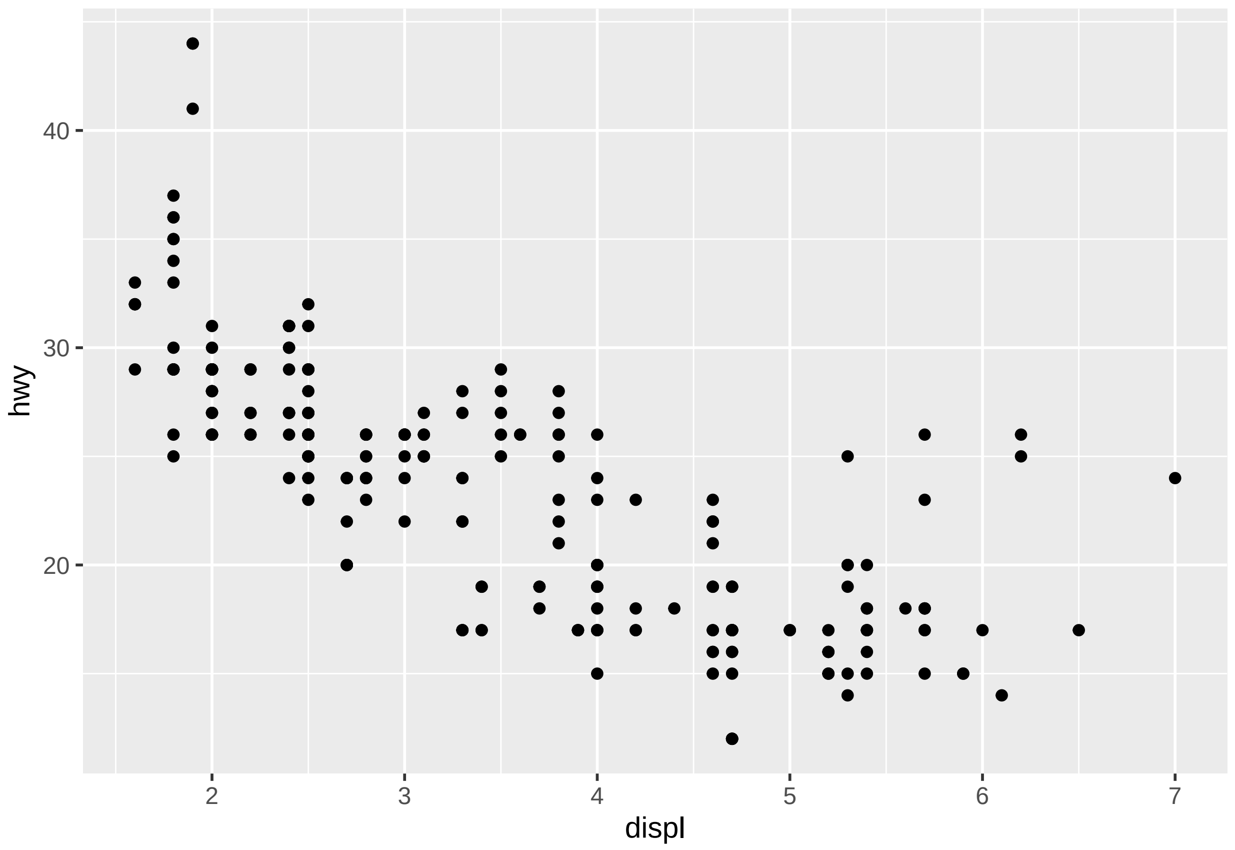
dodge is simply mean put those bars side by sideit is not too bad compare with stack. dodge is more useful imo.

There is another adjustment is useful when we using the geom_point. recall

```r
str(mpg)
```

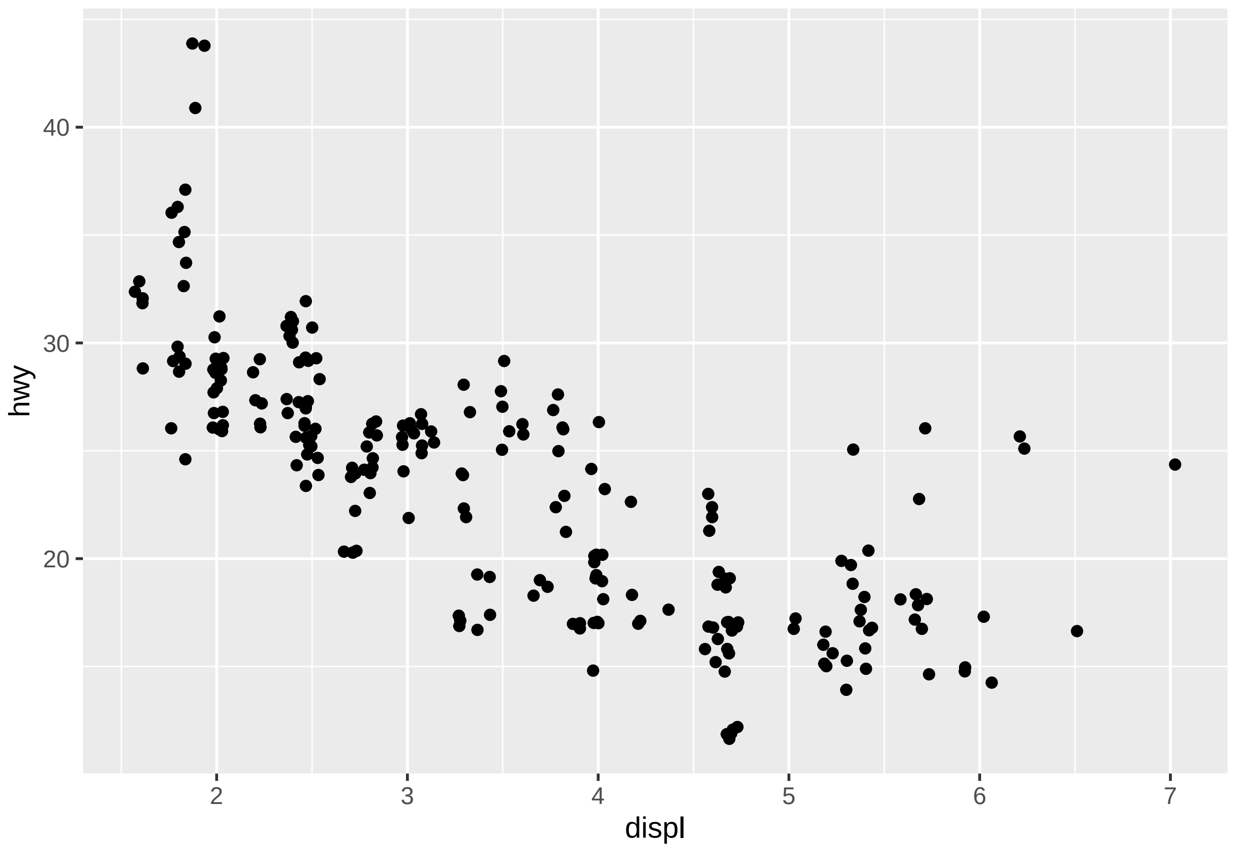
```
## tibble [234 x 11] (S3: tbl_df/tbl/data.frame)
##  $ manufacturer: chr [1:234] "audi" "audi" "audi" "audi" ...
##  $ model       : chr [1:234] "a4" "a4" "a4" "a4" ...
##  $ displ       : num [1:234] 1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
##  $ year        : int [1:234] 1999 1999 2008 2008 1999 1999 2008 1999 1999 2008 ...
##  $ cyl         : int [1:234] 4 4 4 4 6 6 6 4 4 4 ...
##  $ trans       : chr [1:234] "auto(l5)" "manual(m5)" "manual(m6)" "auto(av)" ...
##  $ drv         : chr [1:234] "f" "f" "f" "f" ...
##  $ cty         : int [1:234] 18 21 20 21 16 18 18 18 16 20 ...
##  $ hwy         : int [1:234] 29 29 31 30 26 26 27 26 25 28 ...
##  $ fl          : chr [1:234] "p" "p" "p" "p" ...
##  $ class       : chr [1:234] "compact" "compact" "compact" "compact" ...
```

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



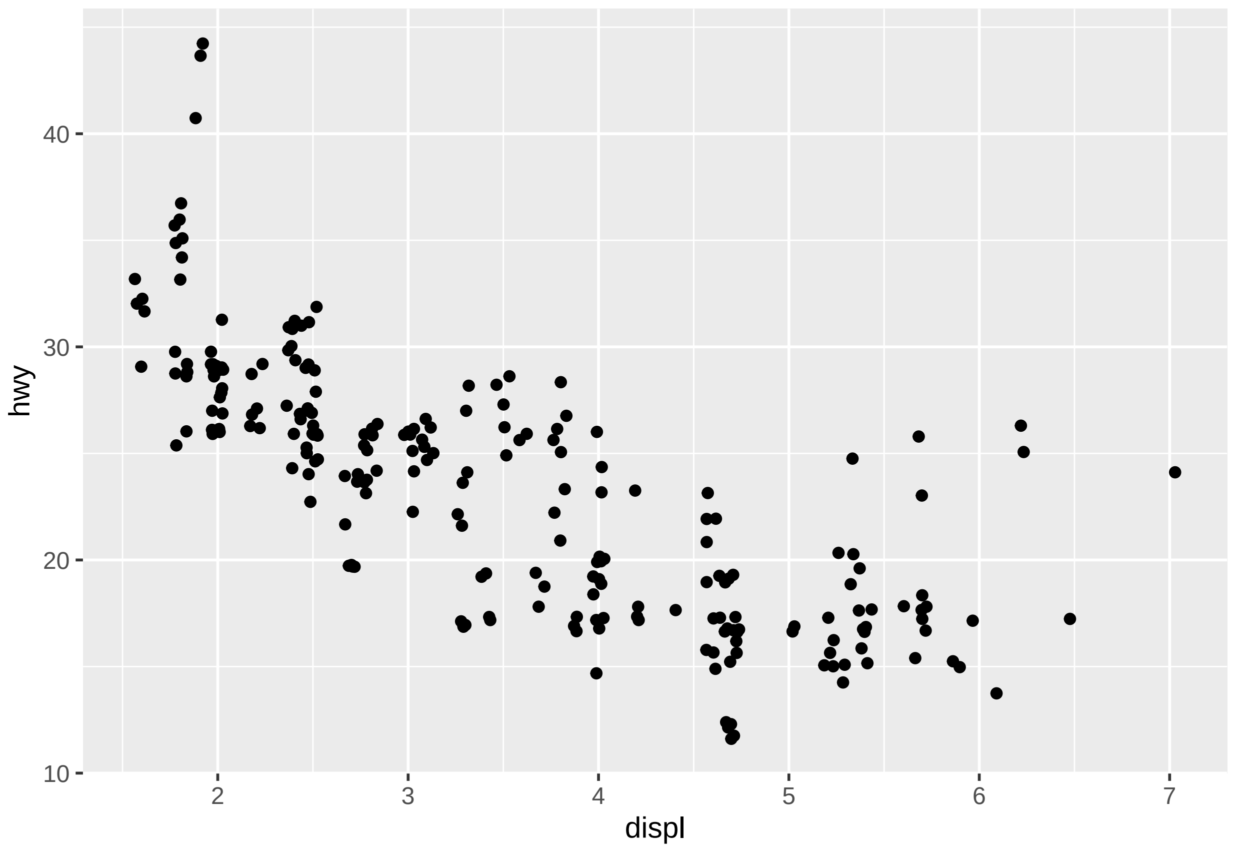
This data set has 234 observation, but obviously the plot does not have all of those. Well, because there are values of hwy and displ are rounded, so the points appear on the plot are actually overlapping each other. This problem is known as overplotting.

```
ggplot(mpg) +
  geom_point(aes(displ,hwy), position ="jitter")
```



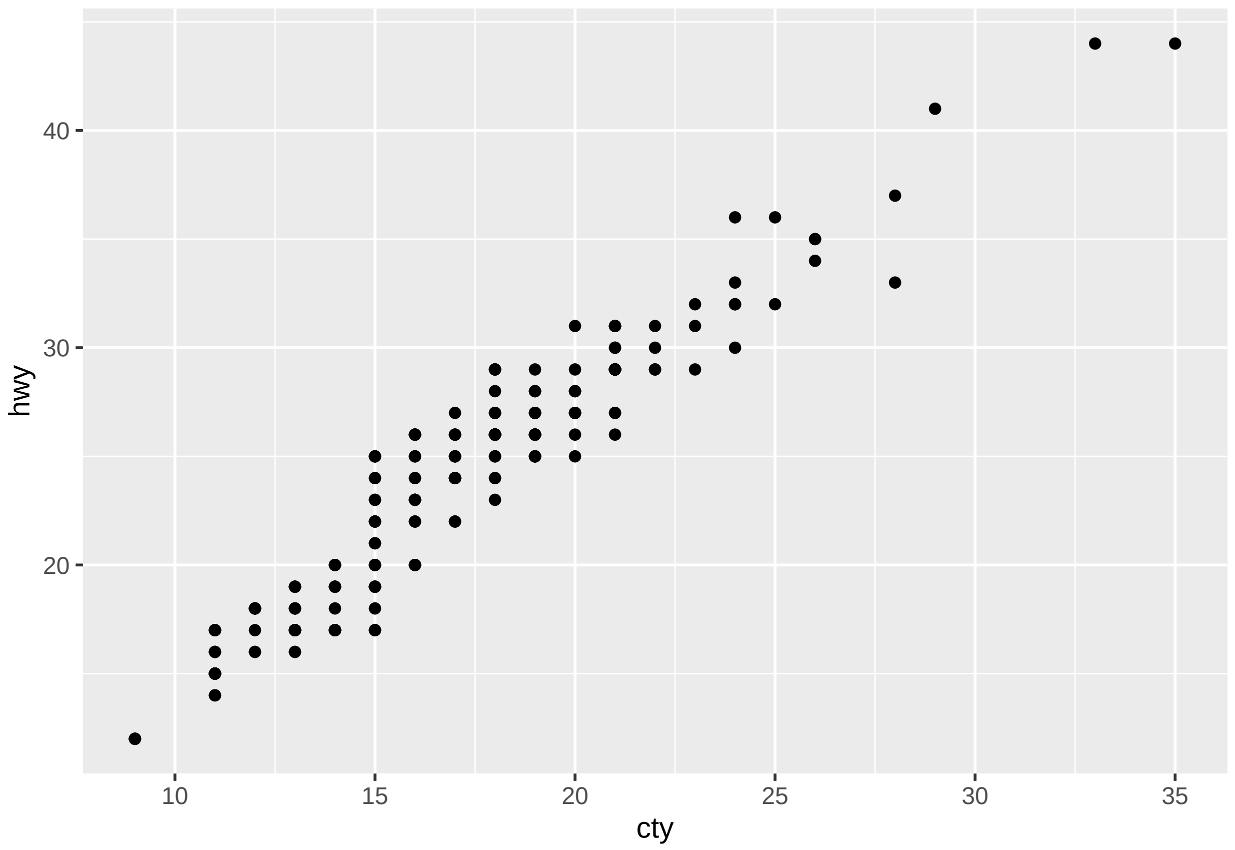
The jitter will add small amount of random noise to each point. which means the scatter point will be drifted slightly. Short handed version

```
ggplot(mpg) +
  geom_jitter(aes(displ,hwy))
```



Q1 What is the problem with this plot? How could you improve it?

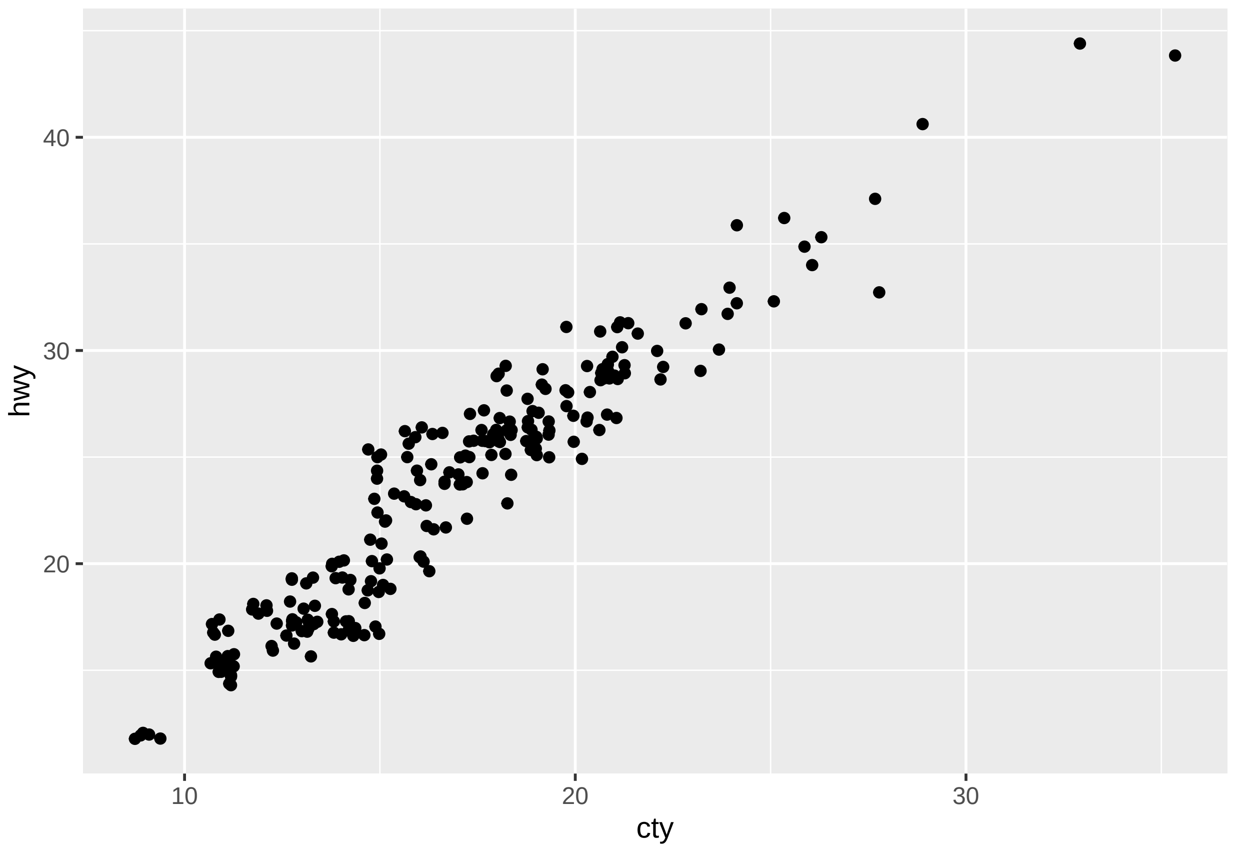
```r
ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +
  geom_point()
```



as you can see, for example x = 15, i can see a stack of dots, which indicates that there is a over plotting problem.

We can simply use jitter to solve the problem

```r
ggplot(mpg, aes(cty,hwy)) +
  geom_jitter()
```

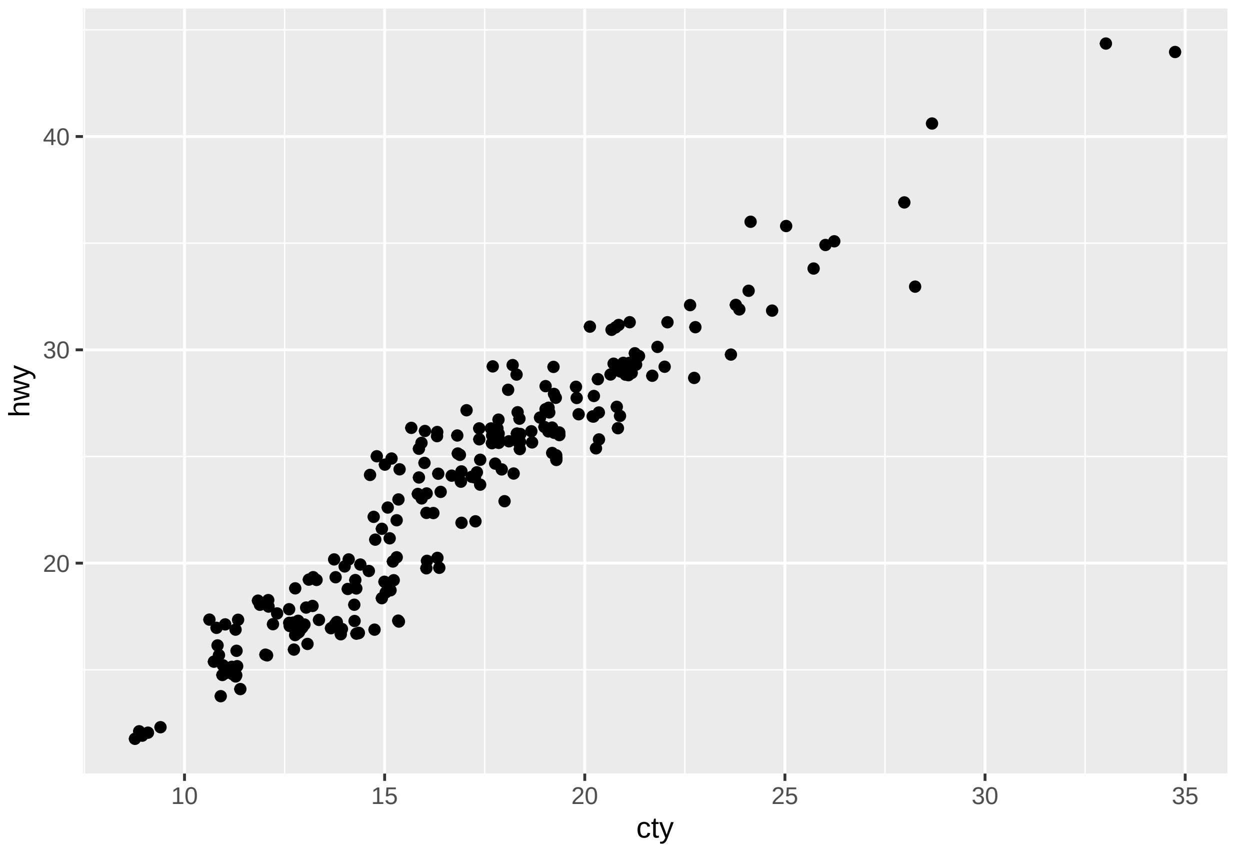


Q2 2. What parameters to geom_jitter() control the amount of jit- tering? According to the documentation, geom_jitter() will take width and height. so if you setthe width and hieght, it will either spread out wider or taller.

Q3 Compare and contrast geom_jitter() with geom_count().

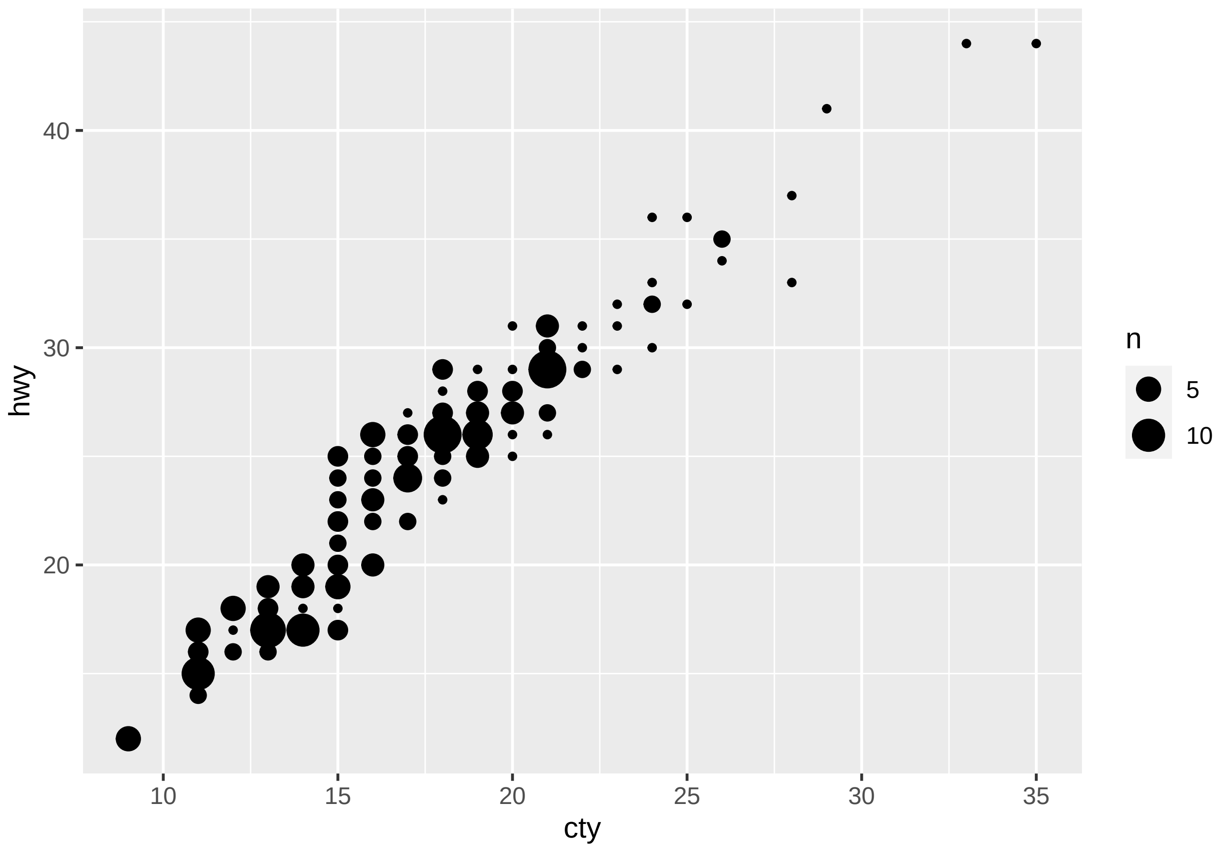
As we have covered, geom_jitter() will create a random noise on the points. Thats is how the point wont overlap each other.

```
ggplot(mpg, aes(cty,hwy)) +
  geom_jitter()
```

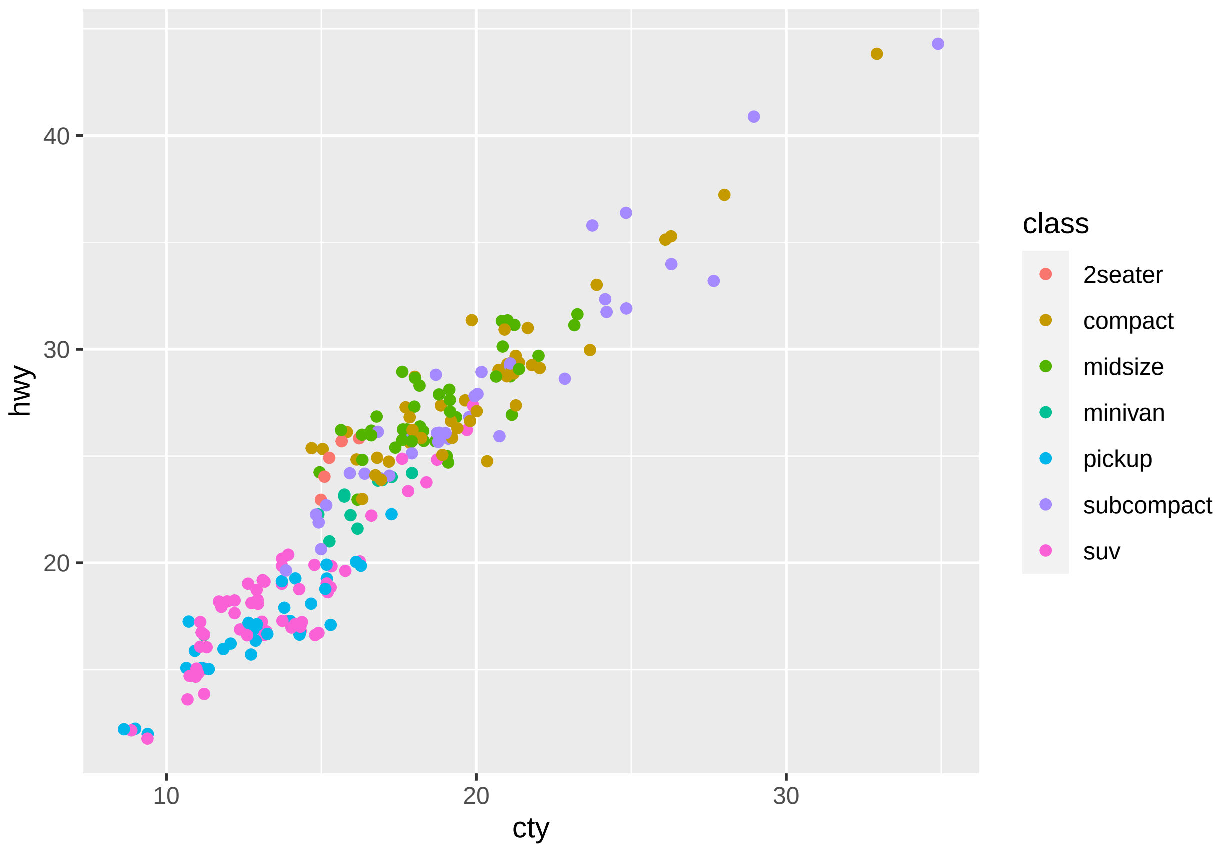


But geom_count, it will size up the point if they are overlapping.

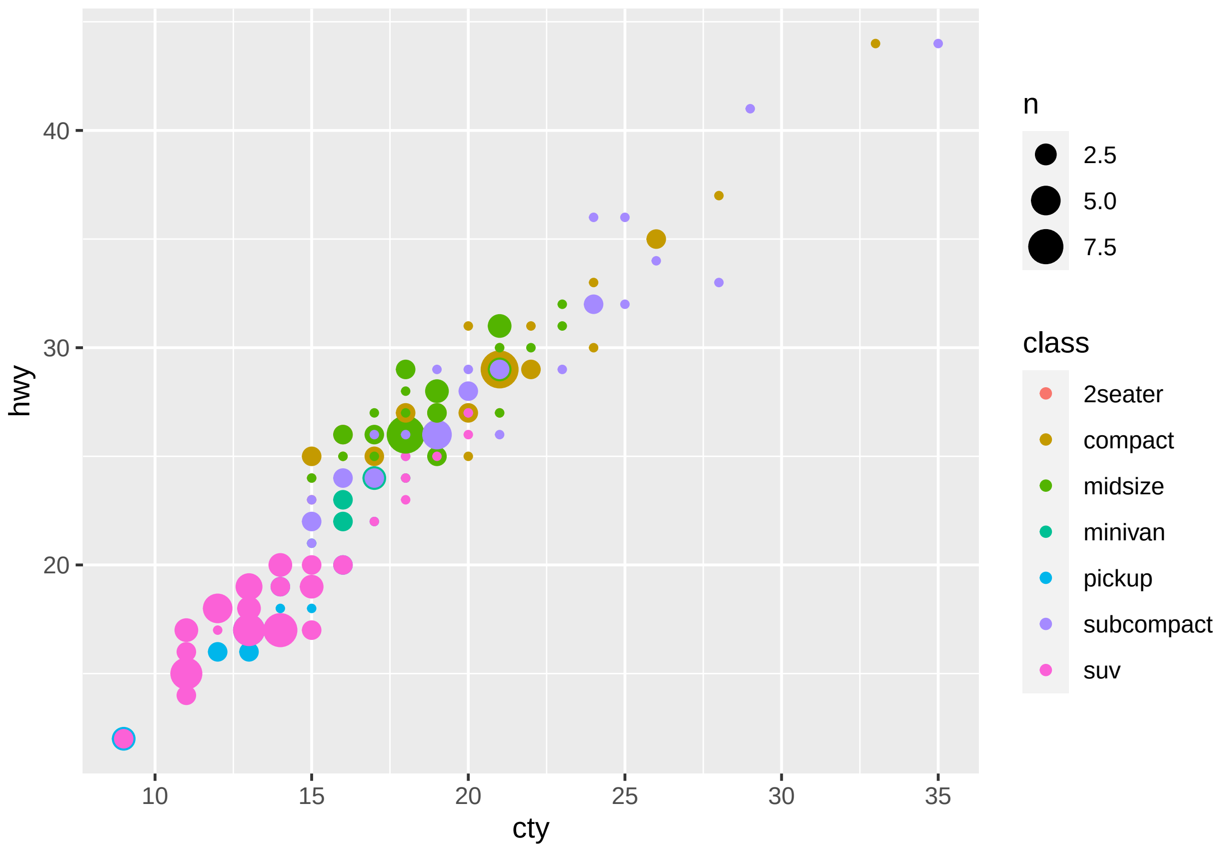
```r
ggplot(mpg, aes(cty,hwy)) +
  geom_count()
```



```r
ggplot(mpg, aes(cty, hwy, color = class)) +
  geom_jitter()
```

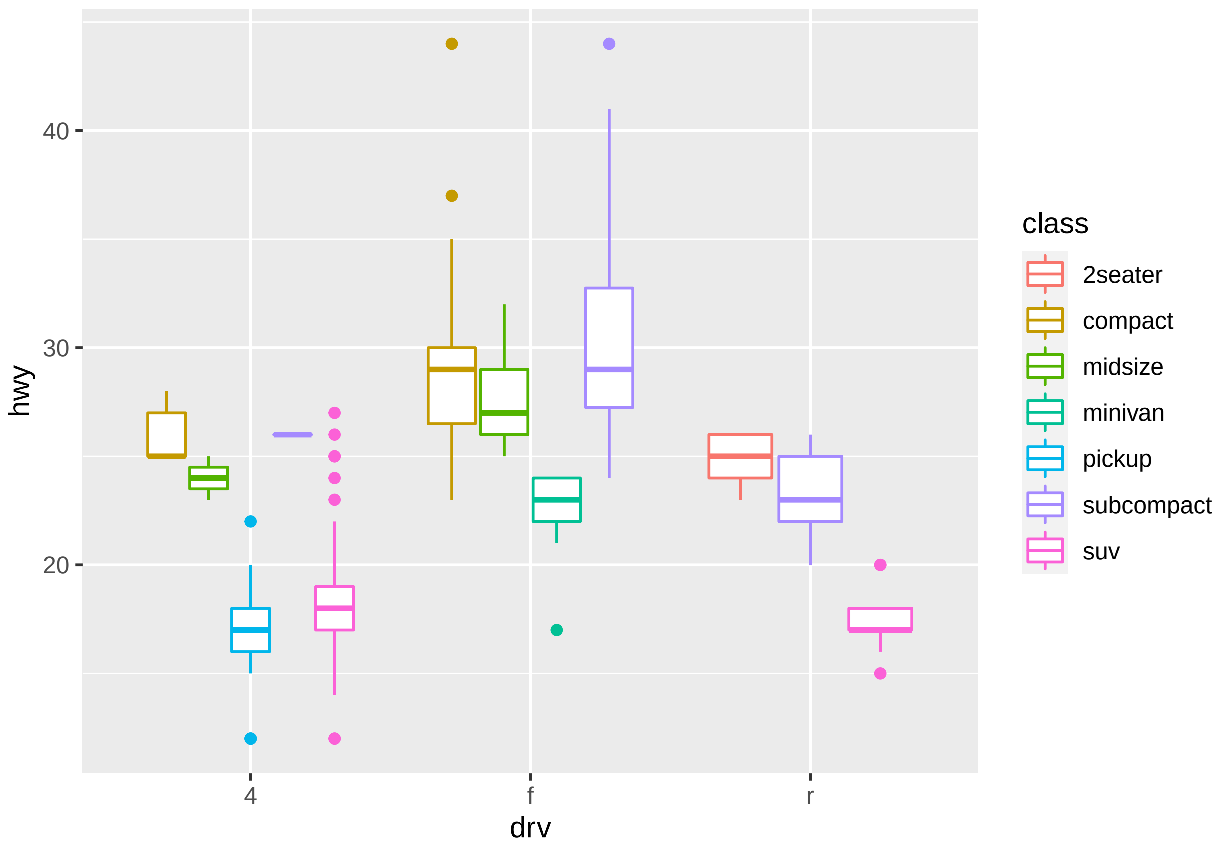


```
ggplot(mpg, aes(cty,hwy, color = class)) +
  geom_count()
```



Q4 What's the default position adjustment for geom_boxplot()? Create a visualization of the mpg dataset that demonstrates it.

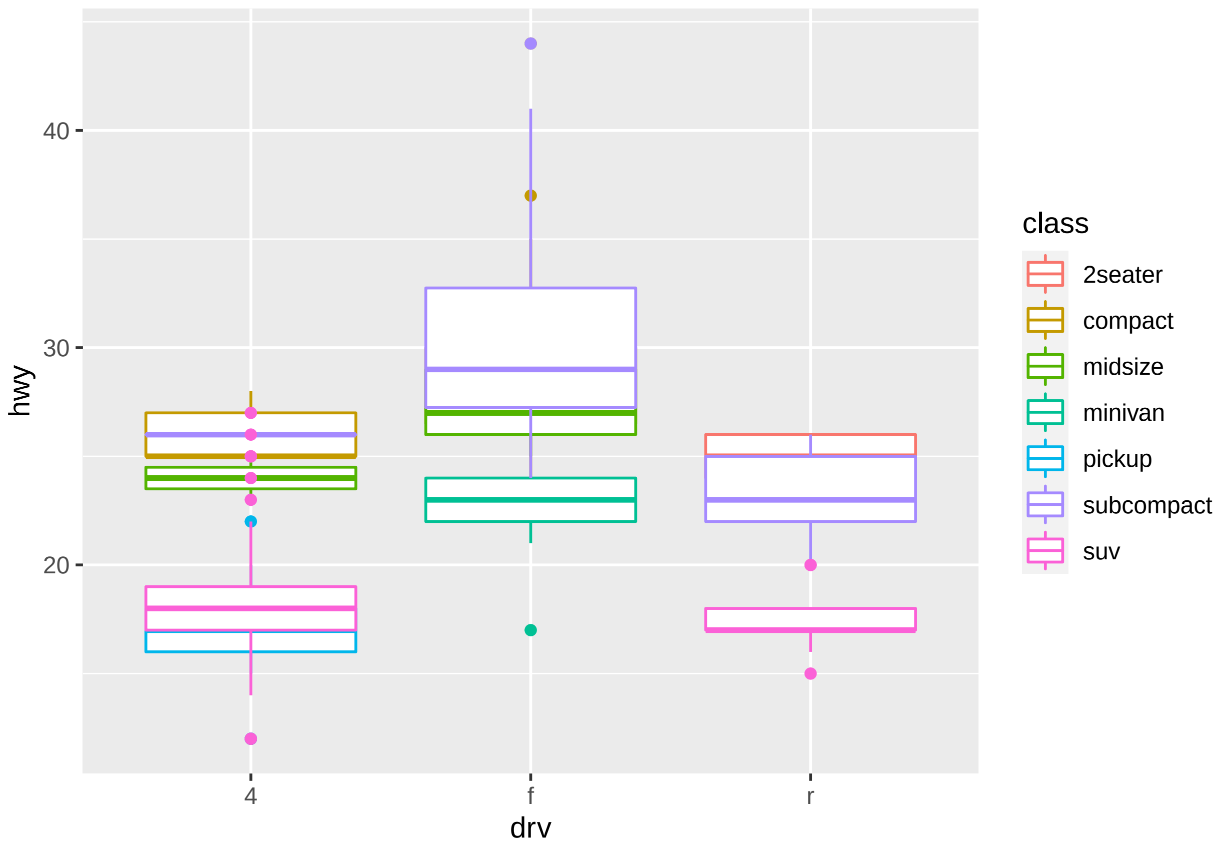
```r
ggplot(mpg, aes(drv,hwy, color = class)) +
  geom_boxplot()
```



A

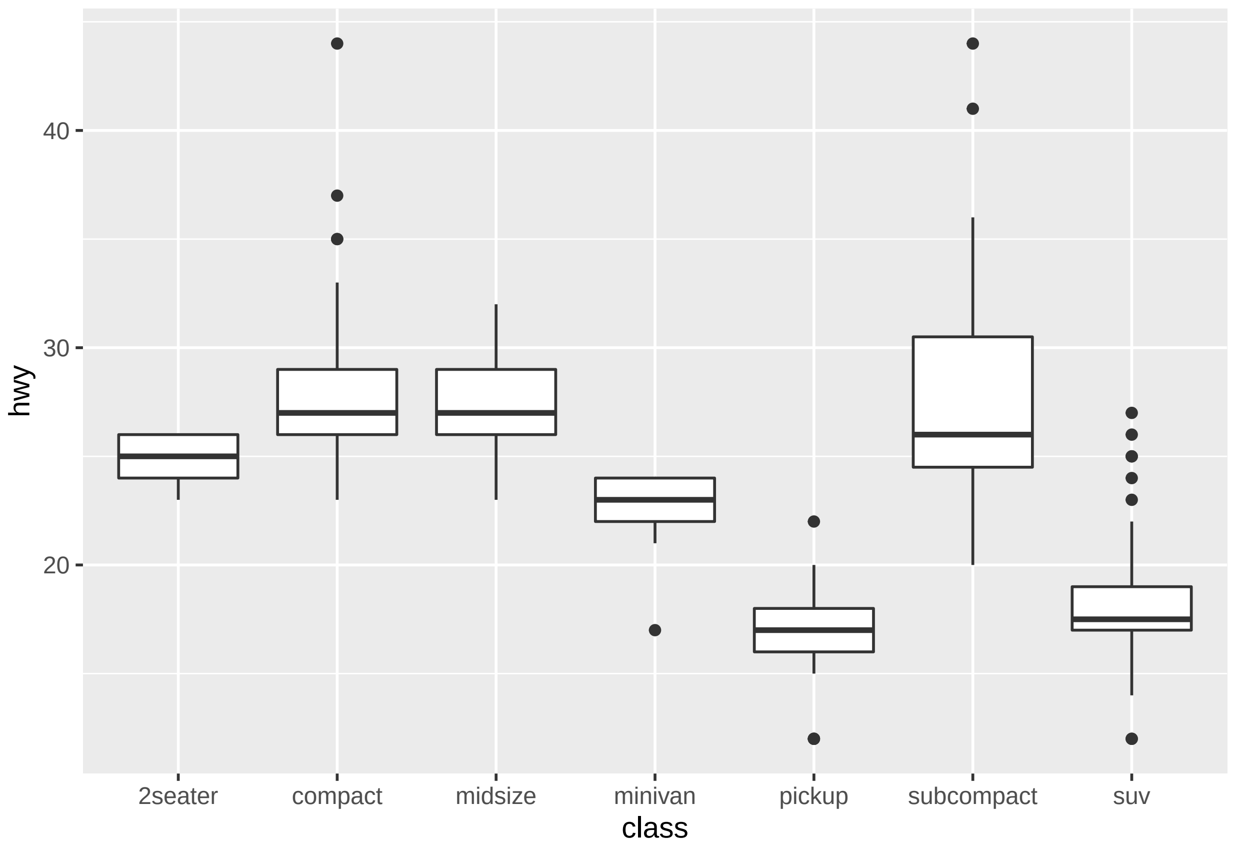
quick refresh, if the color is = a varaible, then it has to be in side the aes. Othwerise, it must be outside of aes. now i can position those box plot overlapping each other.

```r
ggplot(mpg, aes(drv, hwy, color = class)) +
  geom_boxplot(position = "identity")
```



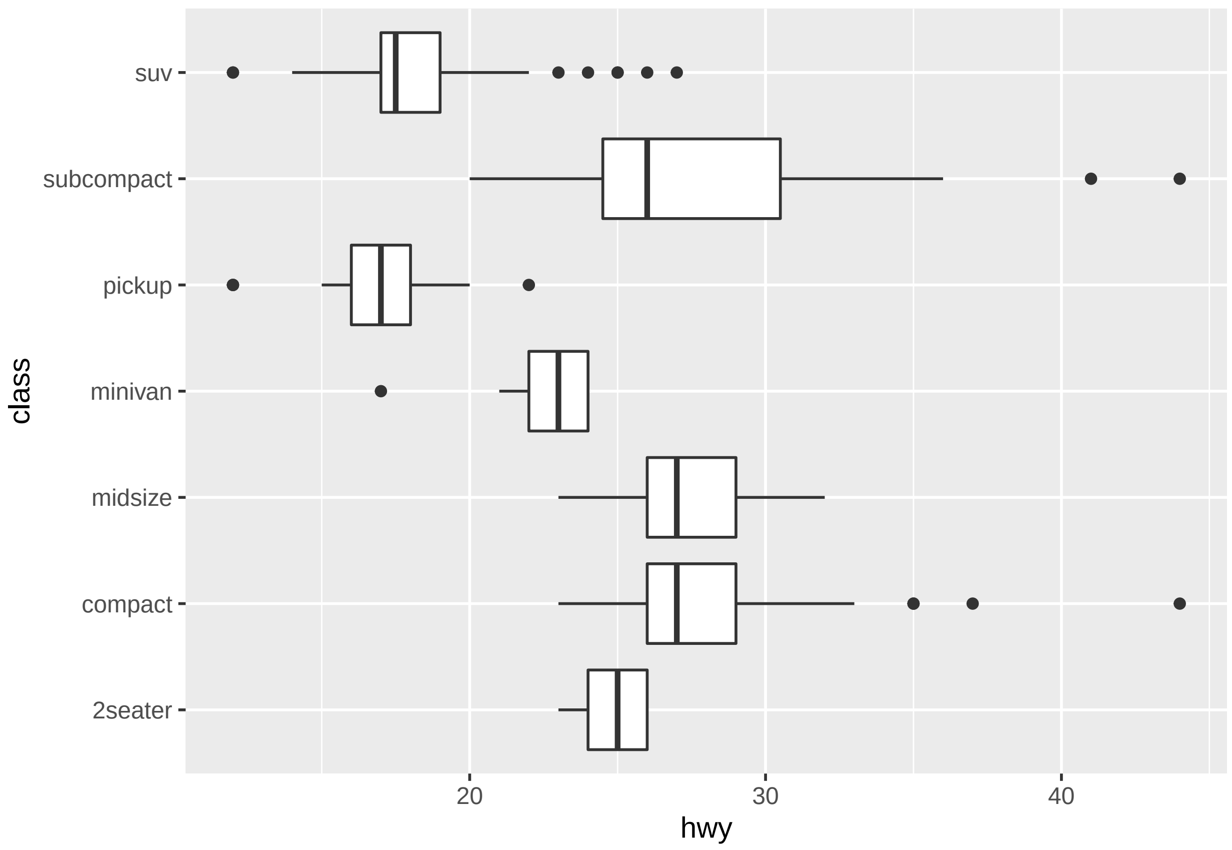
Coordinate systems.

```r
ggplot(mpg, aes(x = class,y = hwy)) +
  geom_boxplot()
```



What if we would like to flip the co ordinate around?

```
ggplot(mpg, aes(class, hwy)) +
  geom_boxplot() +
  coord_flip()
```

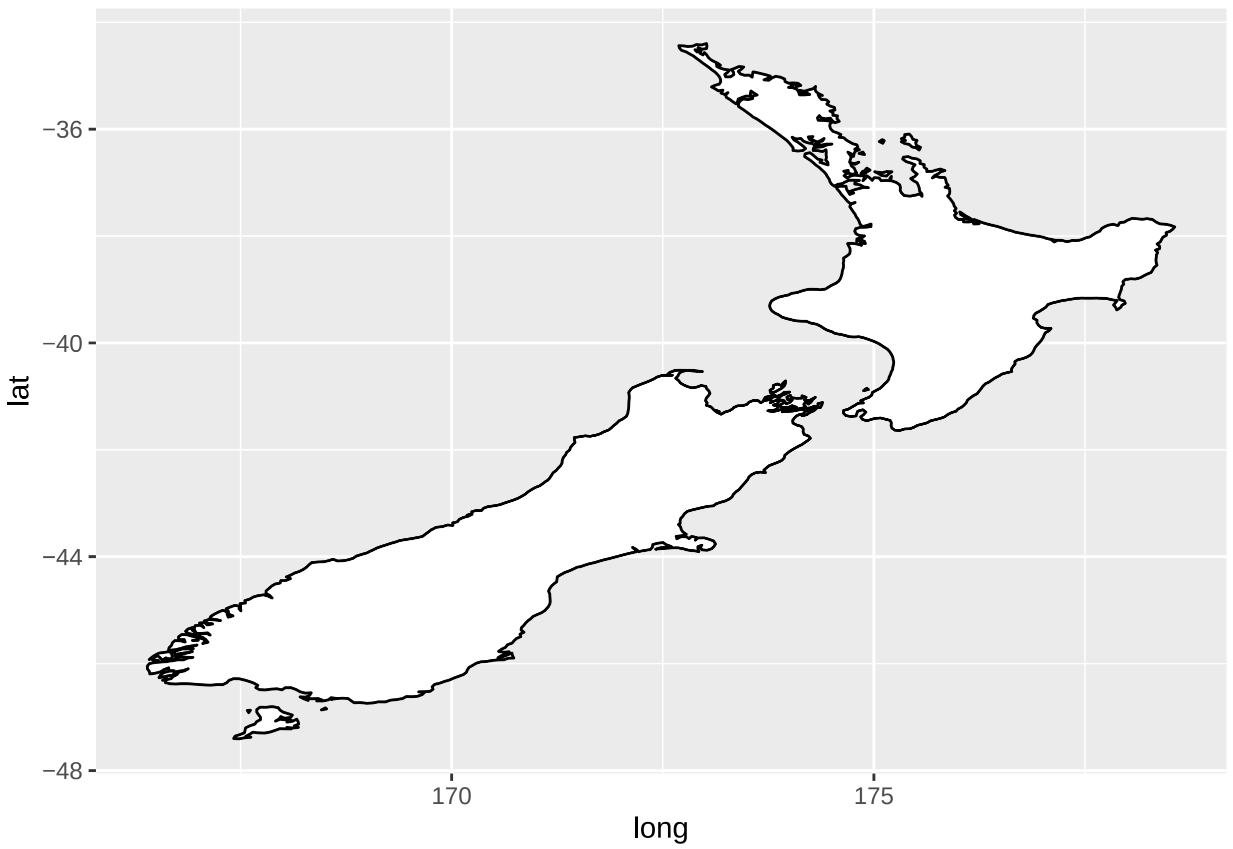


the coord_flip will do

```r
library(ggplot2)
require(maps)
```

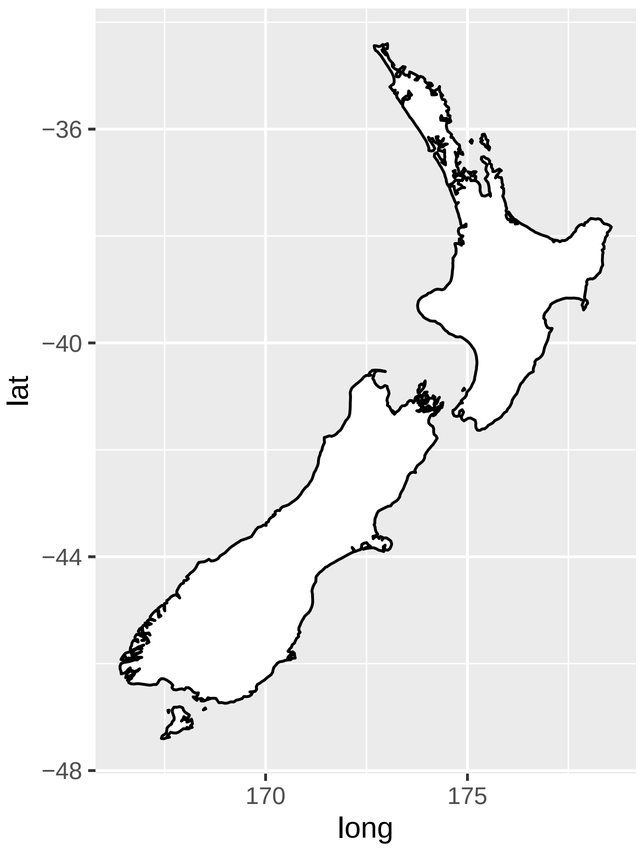
```
## Loading required package: maps

##
## Attaching package: 'maps'

## The following object is masked from 'package:purrr':
##
##     map
```

```r
nz <- map_data("nz")
ggplot(nz, aes(long,lat, group = group)) +
  geom_polygon(fill = "white", color = "black")
```



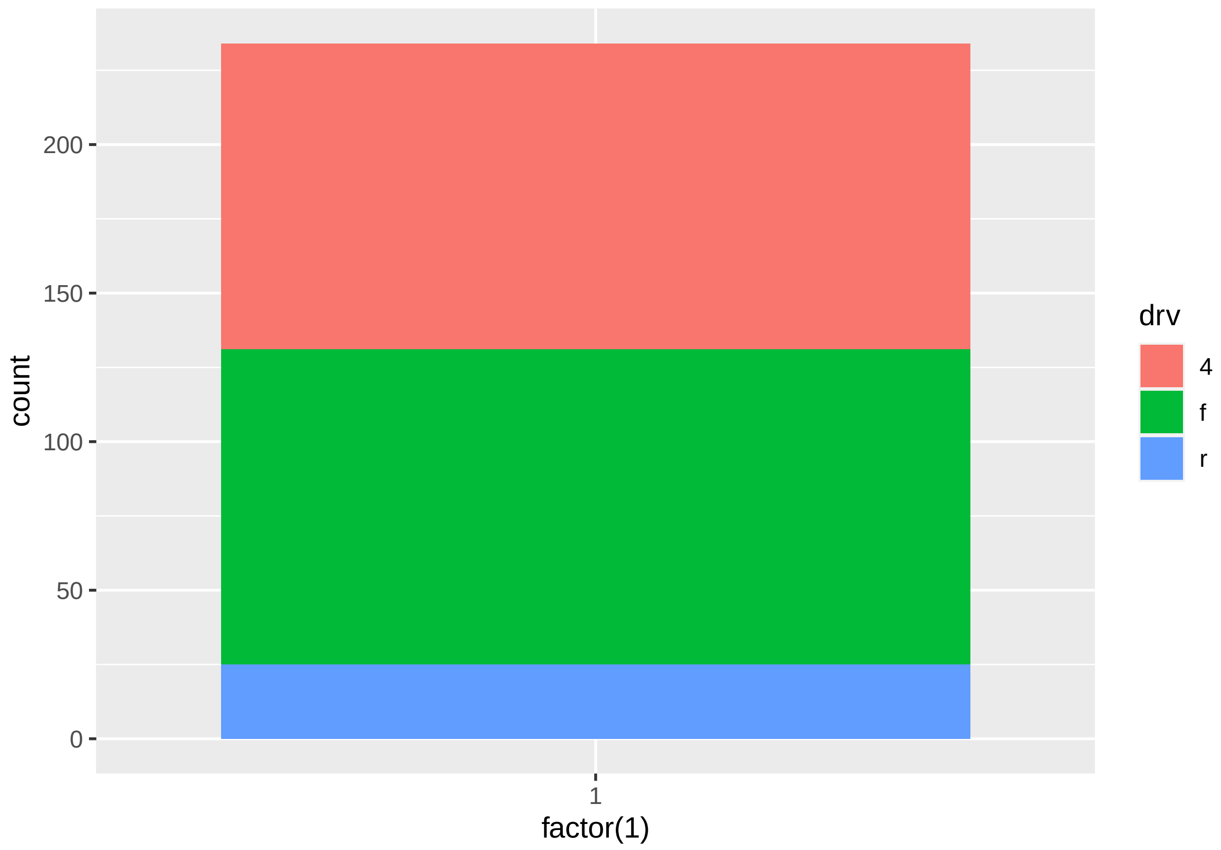
coord_quickmap() will set the aspect ratio correctly for maps. It is super important if you are plotting a lot of data

```
ggplot(nz, aes(long,lat,group = group)) +
  geom_polygon(fill = "white", color = "black") +
  coord_quickmap()
```



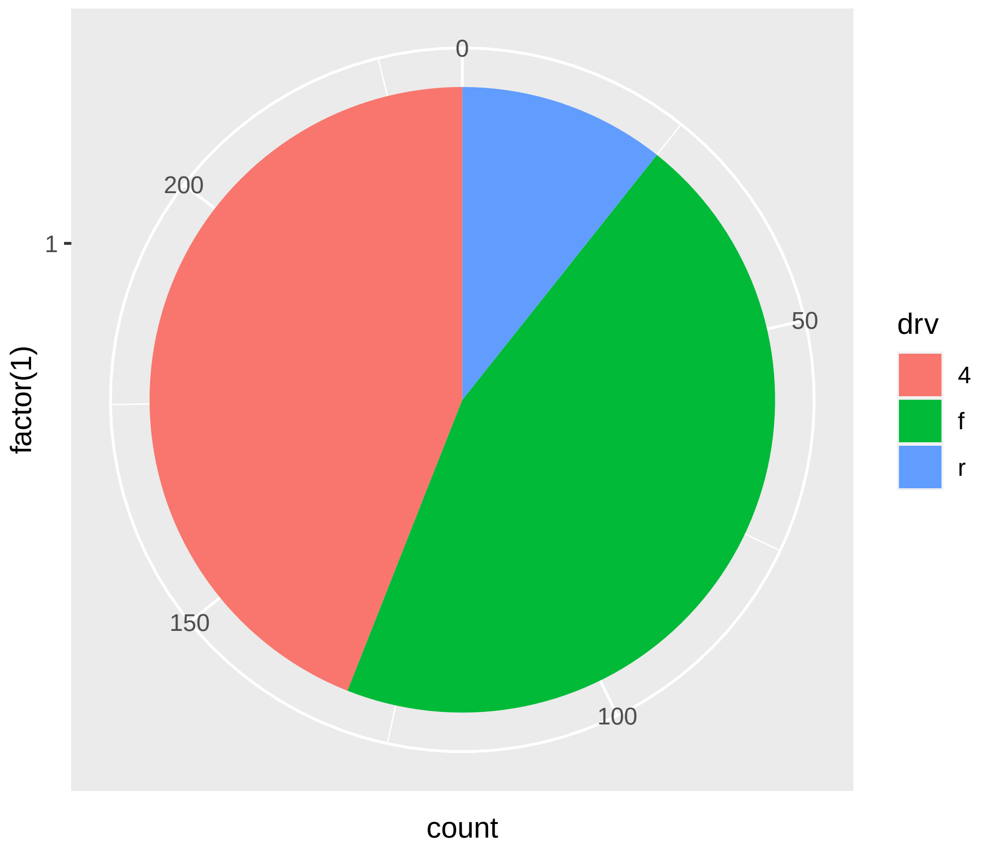
1. Turn a stacked bar chart into a pie chart using coord_polar(). From

```r
ggplot(mpg, aes(x = factor(1), fill = drv)) +
  geom_bar()
```



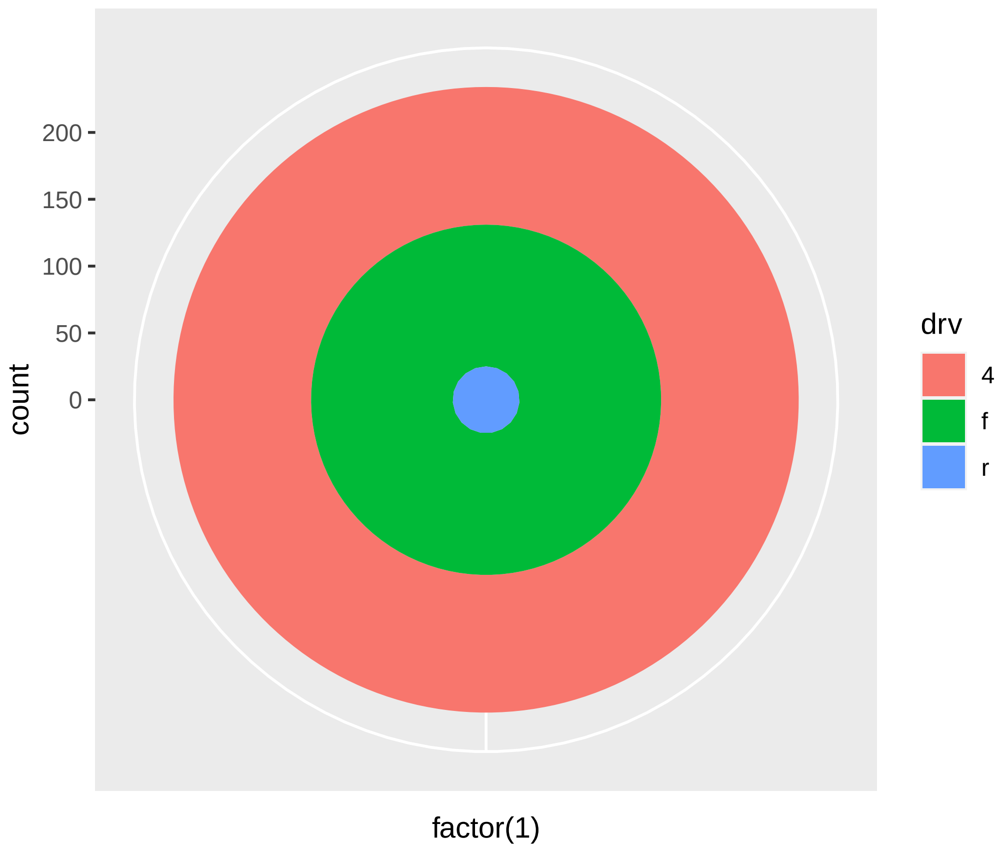
To

```
ggplot(mpg, aes(factor(1), fill = drv)) +
  geom_bar(width =1) +
  coord_polar(theta = "y")
```



The theta is map y as an angle. If the coord_polar() does not specify the the theta, I will have an bull eye effect

```
ggplot(mpg,aes(factor(1), fill = drv)) +
  geom_bar(width = 1) +
  coord_polar()
```



2. What does labs() do? Read the documentation.

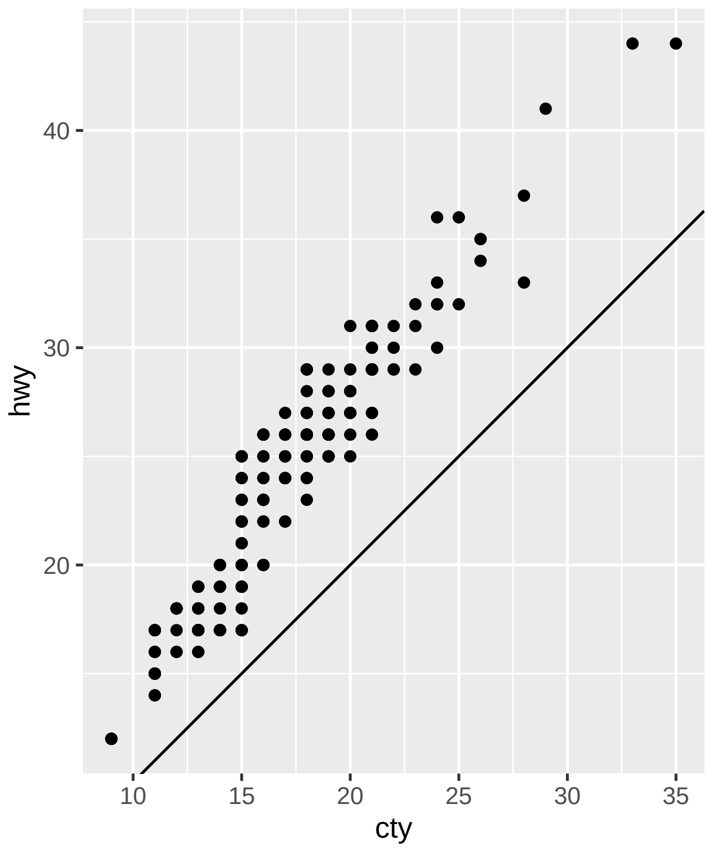
It has multiple purposes, it will add the label for x y and title axies on the plot

3.What's the difference between coord_quickmap() and coord_map()? The coord_map() function uses map projections to project the three-dimensional Earth onto a two-dimensional plane. By default, coord_map() uses the Mercator projection. This projection is applied to all the geoms in the plot. The coord_quickmap() function uses an approximate but faster map projection. This approximation ignores the curvature of Earth and adjusts the map for the latitude/longitude ratio. The coord_quickmap() project is faster than coord_map() both because the projection is computationally easier, and unlike coord_map(), the coordinates of the individual geoms do not need to be transformed.

4. What does the following plot tell you about the relationship between city and highway mpg? Why is coord_fixed() important? What does geom_abline() do?

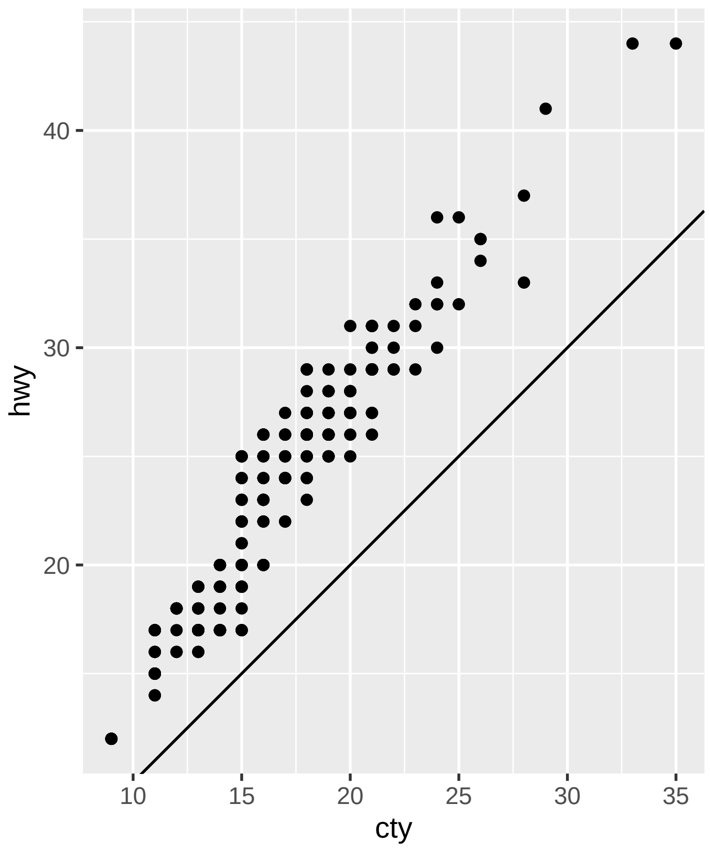
```
p <- ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +
          geom_point() +
          geom_abline() +
          coord_fixed()
p + coord_fixed()
```

```
## Coordinate system already present. Adding new coordinate system, which will replace the existing one
```



without coord_fixed(), the book claim that the line would no longer be 45 degree which i dont see a big difference

```r
q <- ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +
          geom_point() +
          geom_abline() +
          coord_fixed()
q
```


40
30
20
hwy
10
15
20
25
30
35
cty