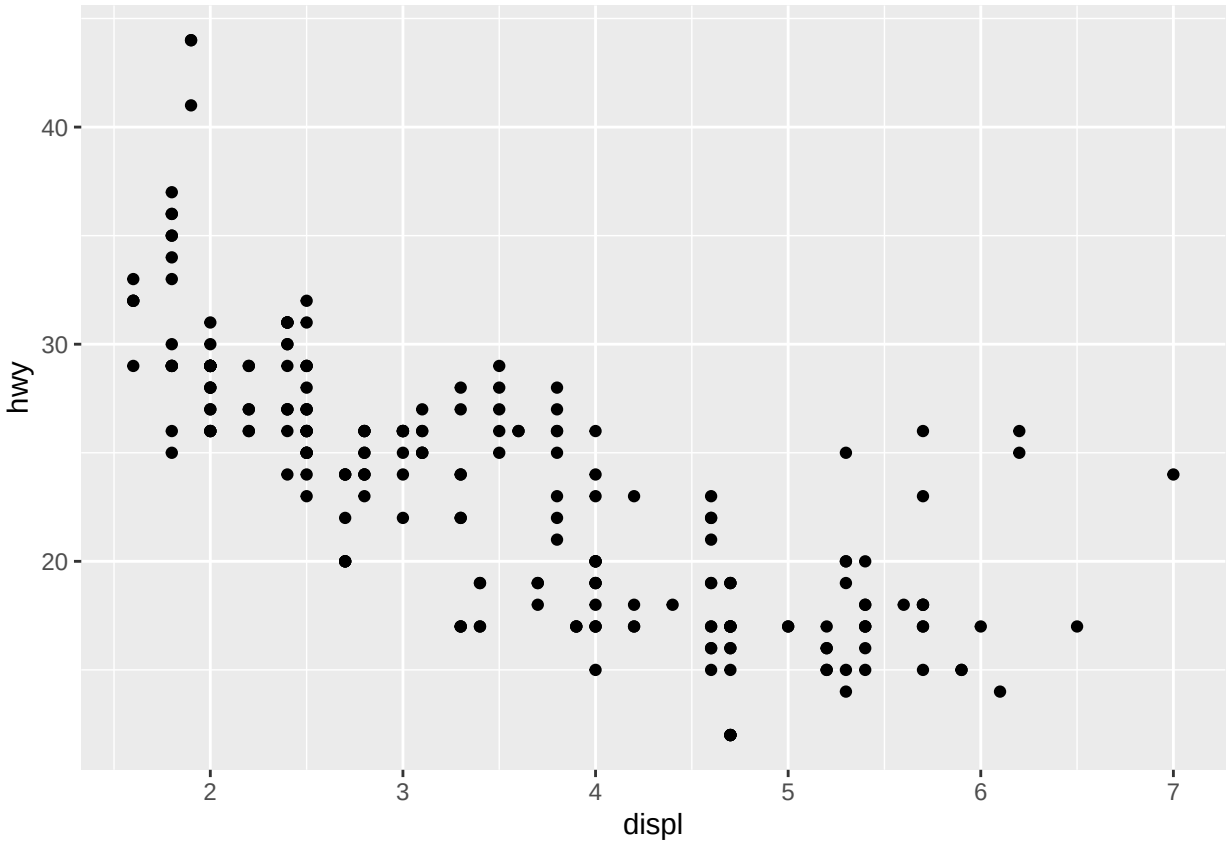
Chapter 3

Icaro Franco Hernandes

2022/09/23

Aesthetic Mapping

```
library(ggplot2)
ggplot2::ggplot(data=ggplot2::mpg)+
  geom_point(aes(x=displ, y=hwy))
```



Placing the *aes* inside the ggplot we do not need to restate the variables in the geometry parts. All the aesthetic changes will be made considering the declared dataset. If instead you place the *aes* in the geometry segment, you'll have to change accordly to what type of graph you want.

Exercises 3.2.4

Exercise 1

Run $ggplot(data = mpg)$. What do you see?

```
ggplot2::ggplot(data=mpg)
```


We see nothing. Adding just the *ggplot* creates a canvas to which we could add layers upon with further codes, like *geom_point*. —

Exercise 2

How many rows are in mpg? How many columns?

```
nrow(mpg)
```

```
## [1] 234
```

```
ncol(mpg)
```

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

Exercise 3

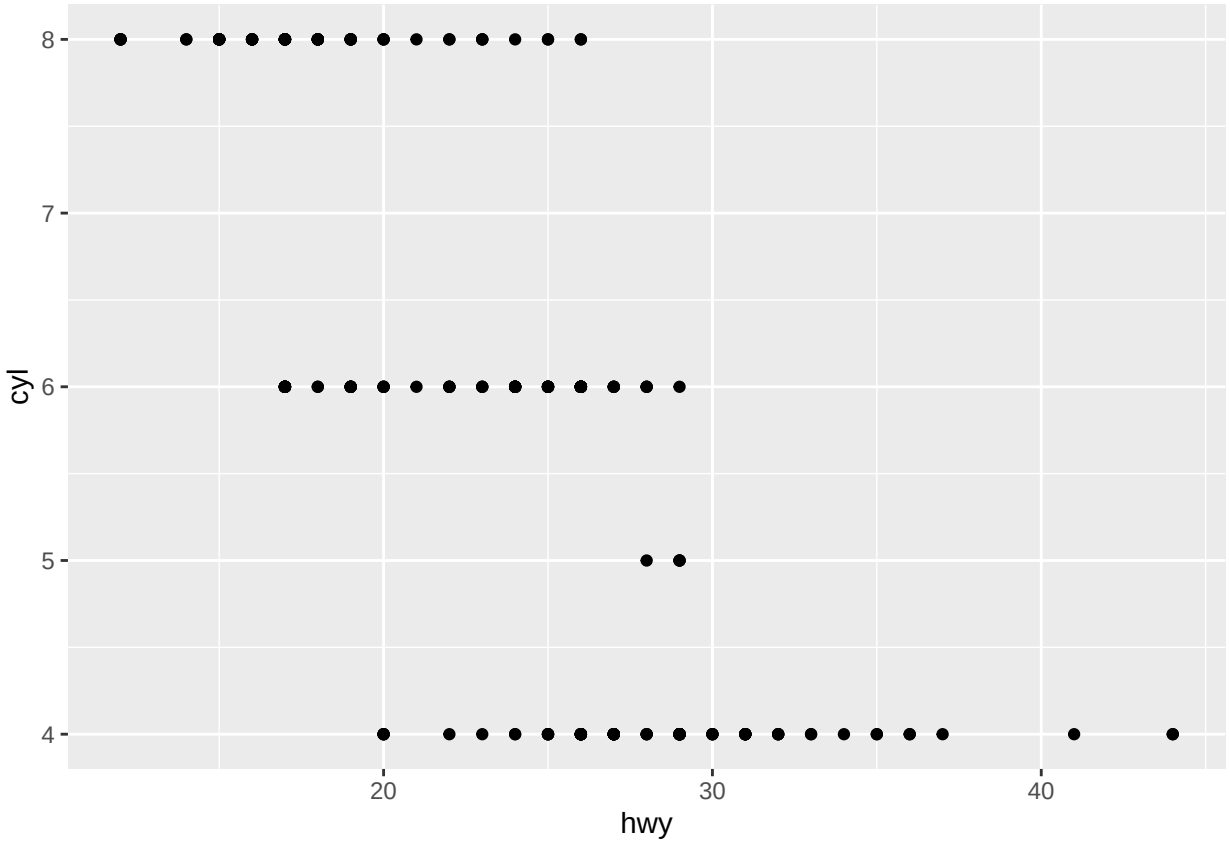
What does the *drv* variable describe? Read the help for *?mpg* to find out.

Adding the question mark before any command takes us to the Help tab. According to the documentation of *mpg* the *drv* variable gives us what type of drive train the car uses, wether is front, rear ow four-wheel drive.

Exercise 4

Make a scatterplot of *hwy* vs *cyl*.

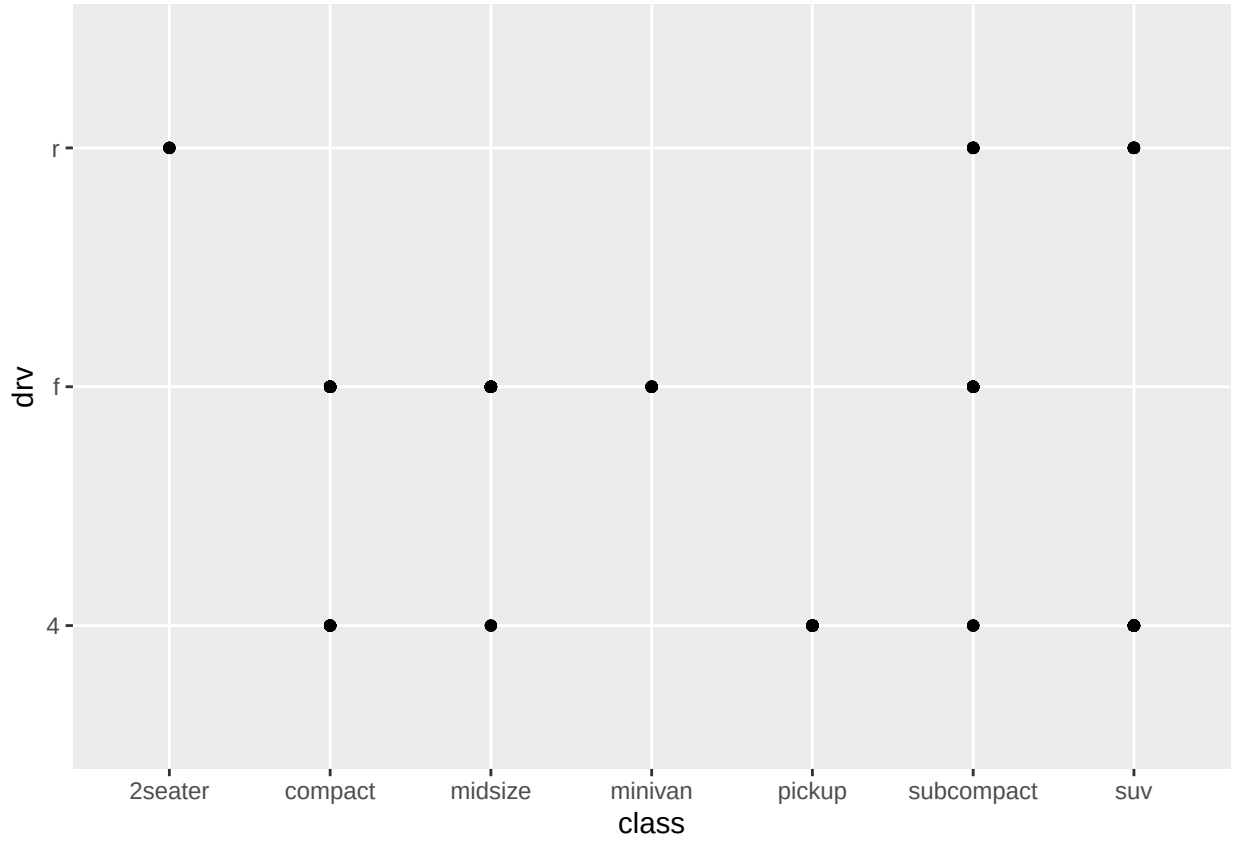
```
ggplot2::ggplot(data=mpg, aes(x=hwy, y=cyl))+
  geom_point()
```



Exercise 5

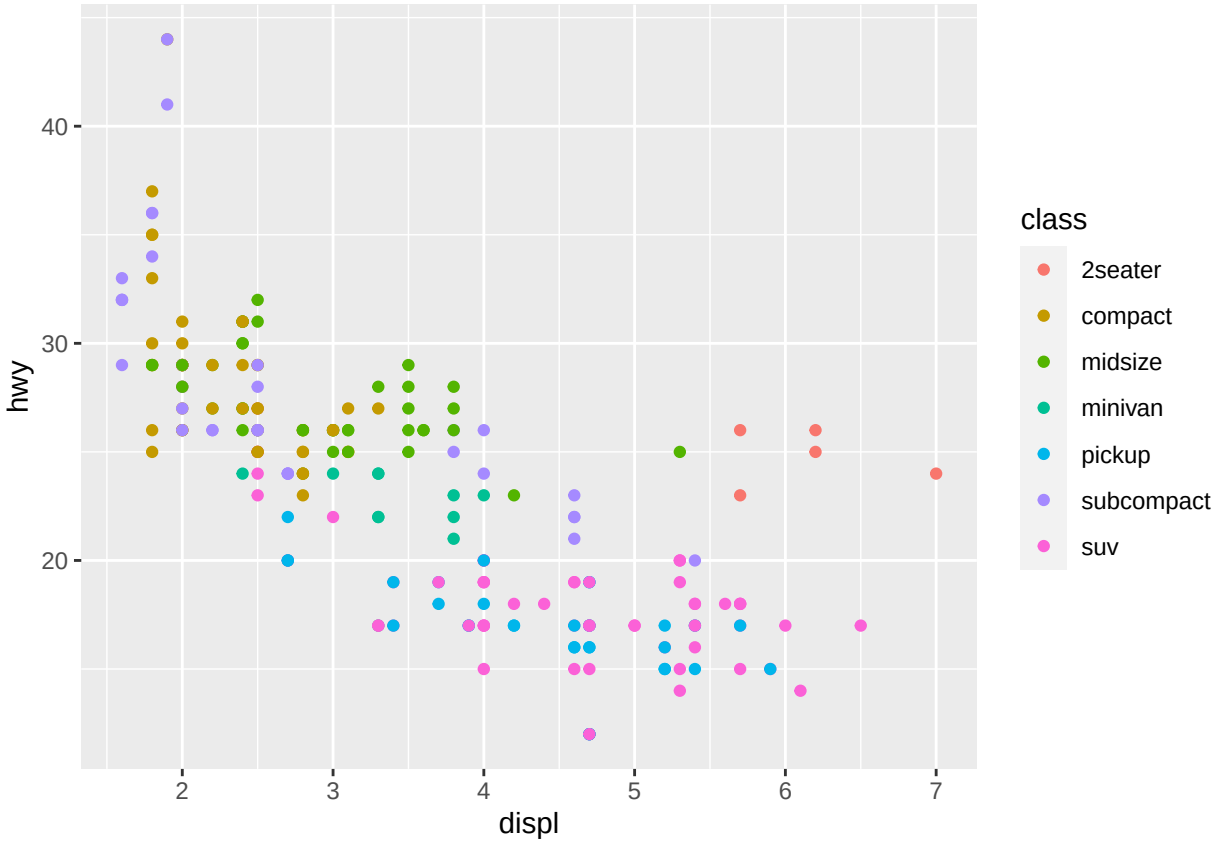
What happens if you make a scatterplot of *class* vs *drv*? Why is the plot not useful?

```
ggplot2::ggplot(data=mpg,aes(x=class, y=drv))+
  geom_point()
```



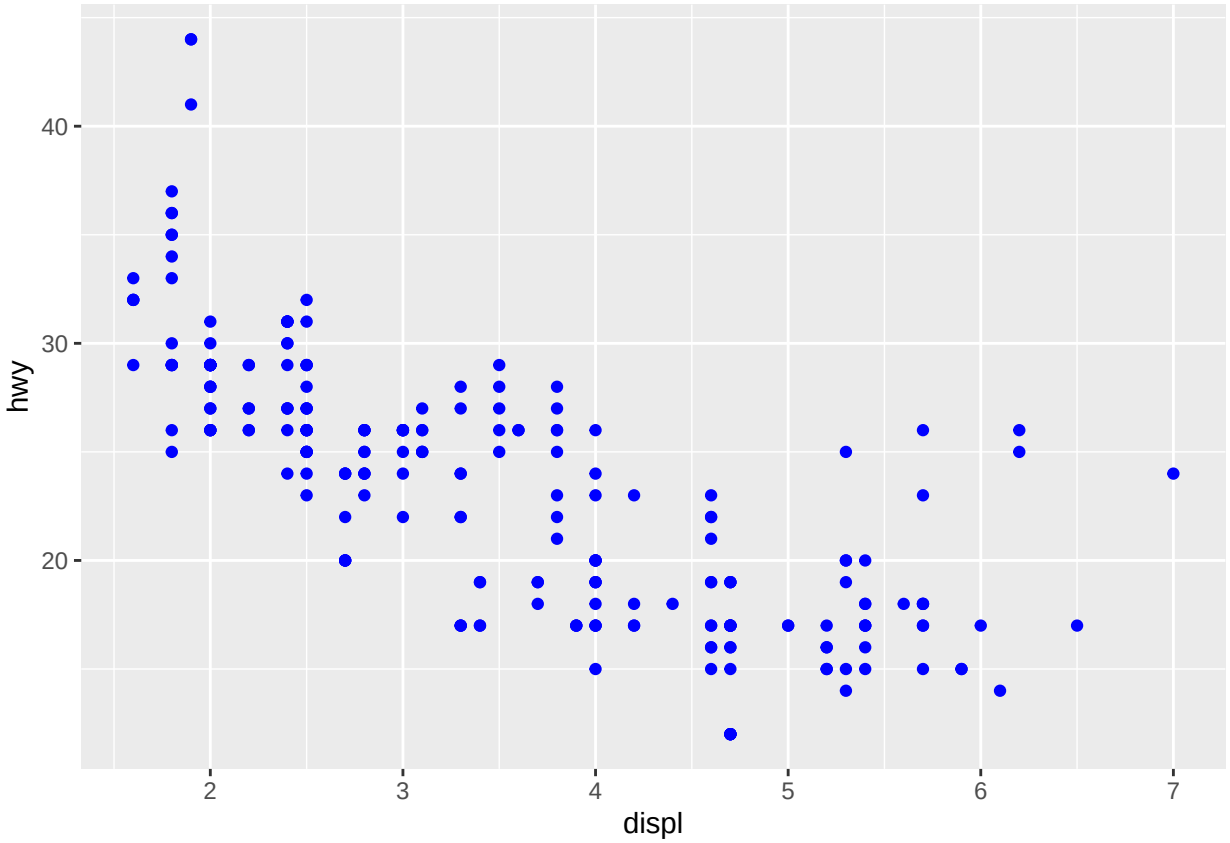
This scatter plot is not useful since it does not indicate any pattern. No inference can be made with this graph.

```
ggplot2::ggplot(data=mpg)+
  geom_point(aes(x=displ,y=hwy, color=class))
```



It is not a good practice to map an unordered variable to a ordered aesthetic (like in the example $size = class$). $alpha = variable$ is for transparency of the points and $shape = variable$ is for each variable an observation has a different shape in the graph.

```
ggplot2::ggplot(data=mpg)+
  geom_point(aes(x=displ,y=hwy), color="blue")
```



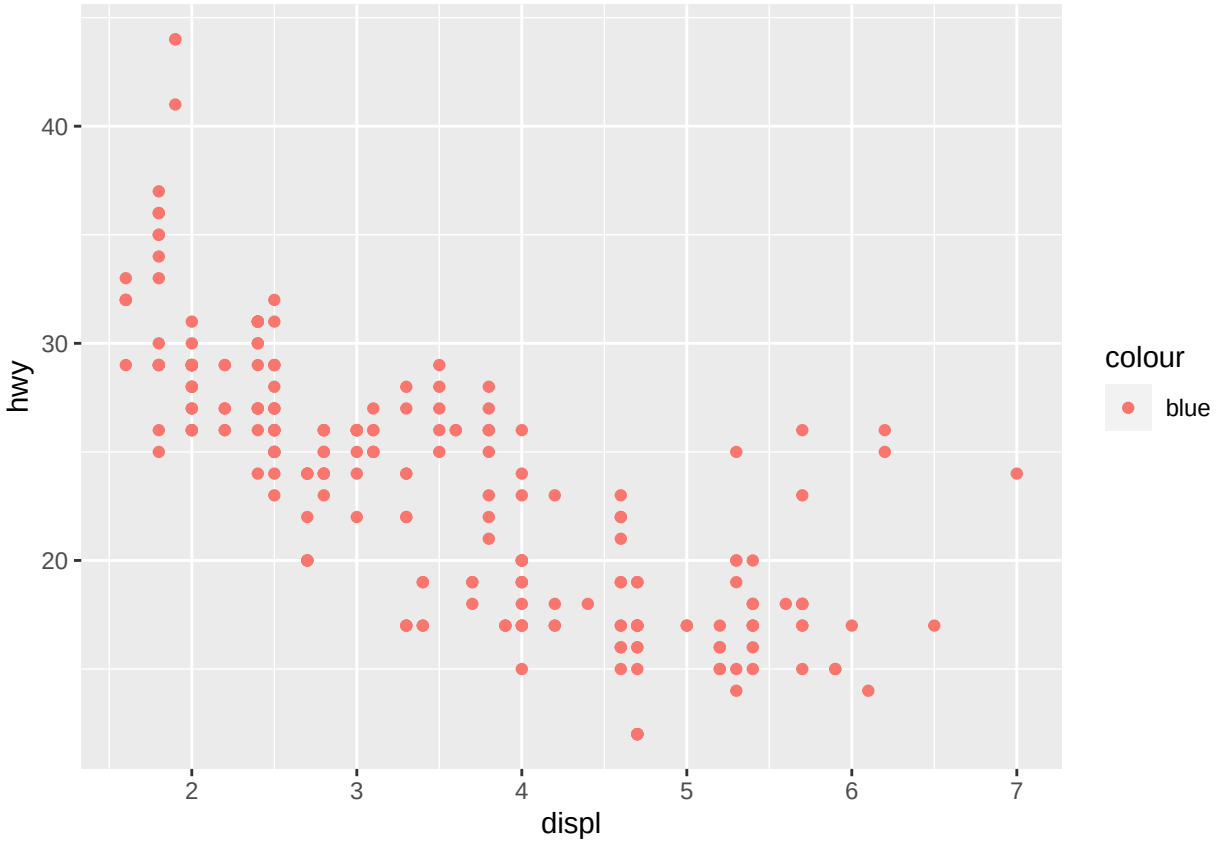
This way, all the points will be blue. Since the command is outside the *aes*() "the color does not convey information about a variable".

Exercises 3.3.1

Exercise 1

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

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



The points are not blue since the color command is inside the *aes*() function. That would be corret if we wanted to segment the data by some variable. If we just want to paint all the points blue we'd just have to put the command outside *aes*()

Exercise 2

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

```
head(mpg)
```

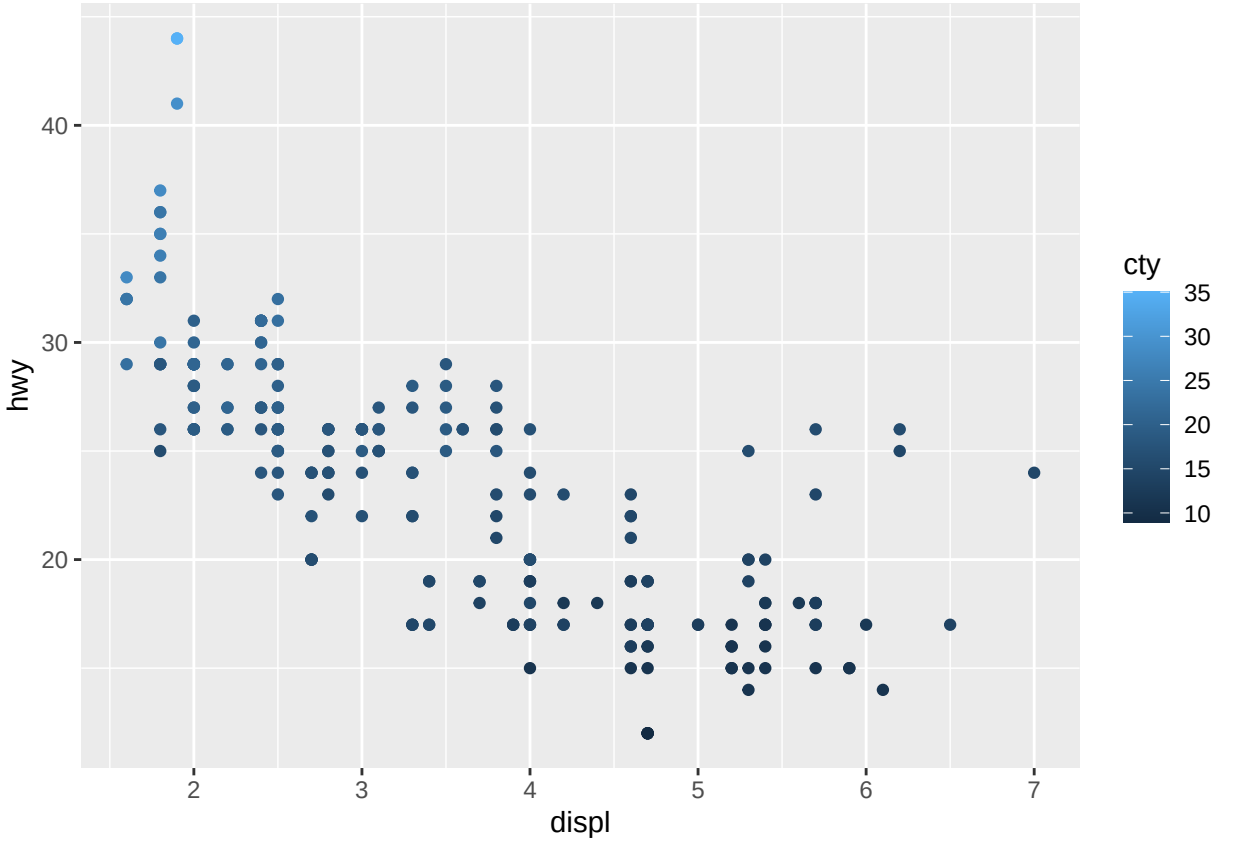
```
## # A tibble: 6 x 11
##   manufacturer model displ  year   cyl trans      drv     cty   hwy fl    class
##   <chr>        <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr> <chr>
## 1 audi         a4      1.8  1999     4 auto(l5)   f        18    29 p     compa~
## 2 audi         a4      1.8  1999     4 manual(m5) f        21    29 p     compa~
## 3 audi         a4      2    2008     4 manual(m6) f        20    31 p     compa~
## 4 audi         a4      2    2008     4 auto(av)   f        21    30 p     compa~
## 5 audi         a4      2.8  1999     6 auto(l5)   f        16    26 p     compa~
## 6 audi         a4      2.8  1999     6 manual(m5) f        18    26 p     compa~
```

The only continuous variables are *cty* and *hwy*. These variables regard the fuel consumption in the city and on the highway.

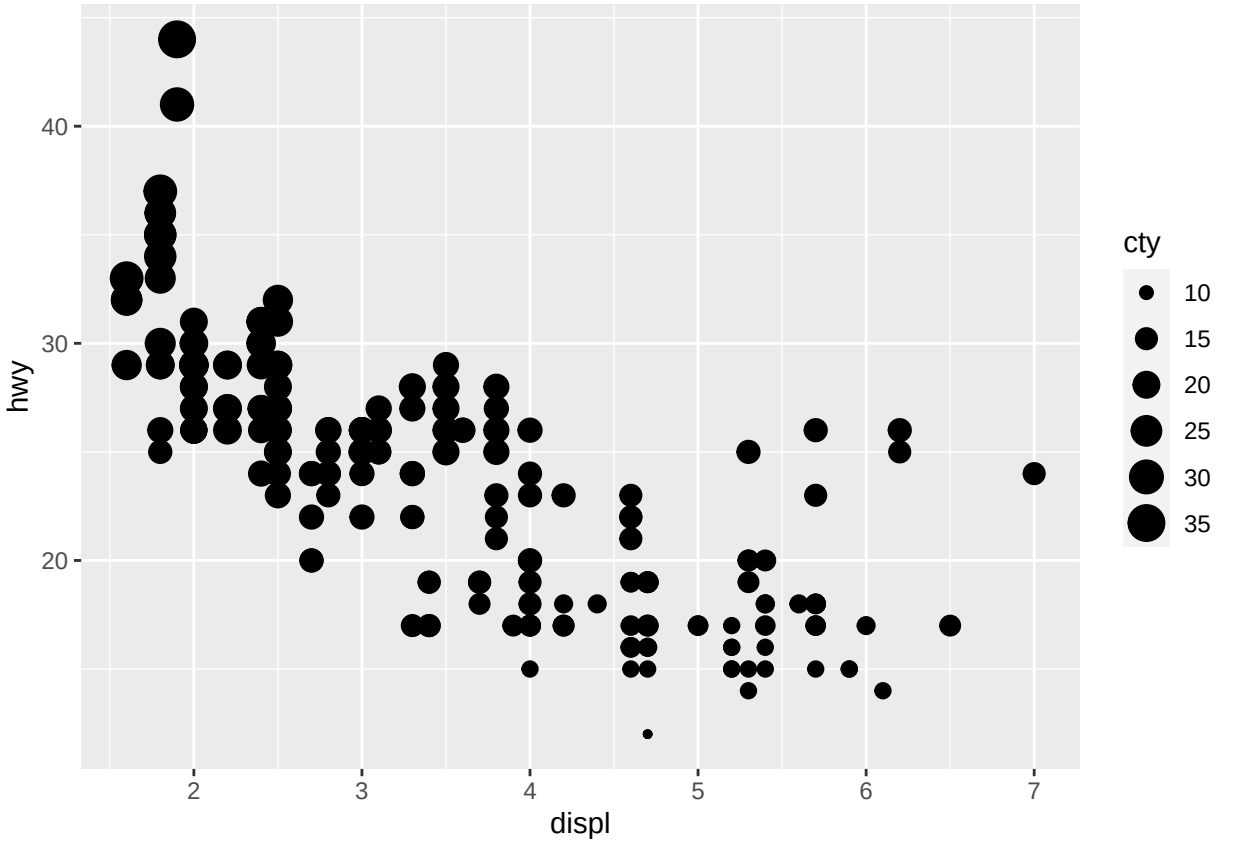
Exercise 3

Map a continuous variable to *color*, *size*, and *shape*. How do these aesthetics behave differently for categorical vs. continuous variables?

```
ggplot2::ggplot(data=mpg)+
  geom_point(aes(x=displ, y=hwy, color=cty))
```




```
ggplot2::ggplot(data=mpg)+
  geom_point(aes(x=displ, y=hwy, size=cty))
```

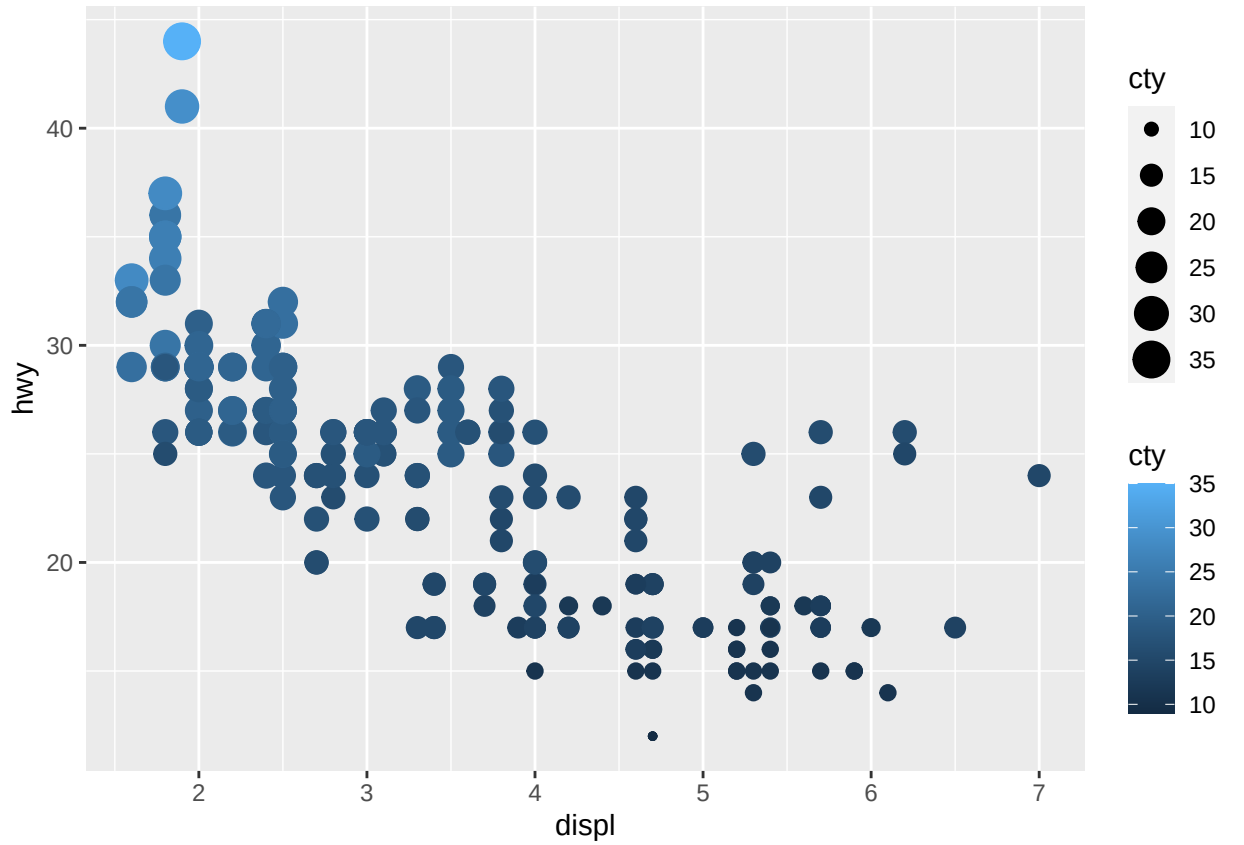


The last graph proposed cannot be plotted since "a continuous variable cannot be mapped to shape". The way they behave differently in the way that a continuous variable will give a gradient for colors and sizes while a discrete variable will have a different color for each class.

Exercise 4

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

```
ggplot2::ggplot(data=mpg)+
  geom_point(aes(x=displ, y=hwy, color=cty, size=cty))
```

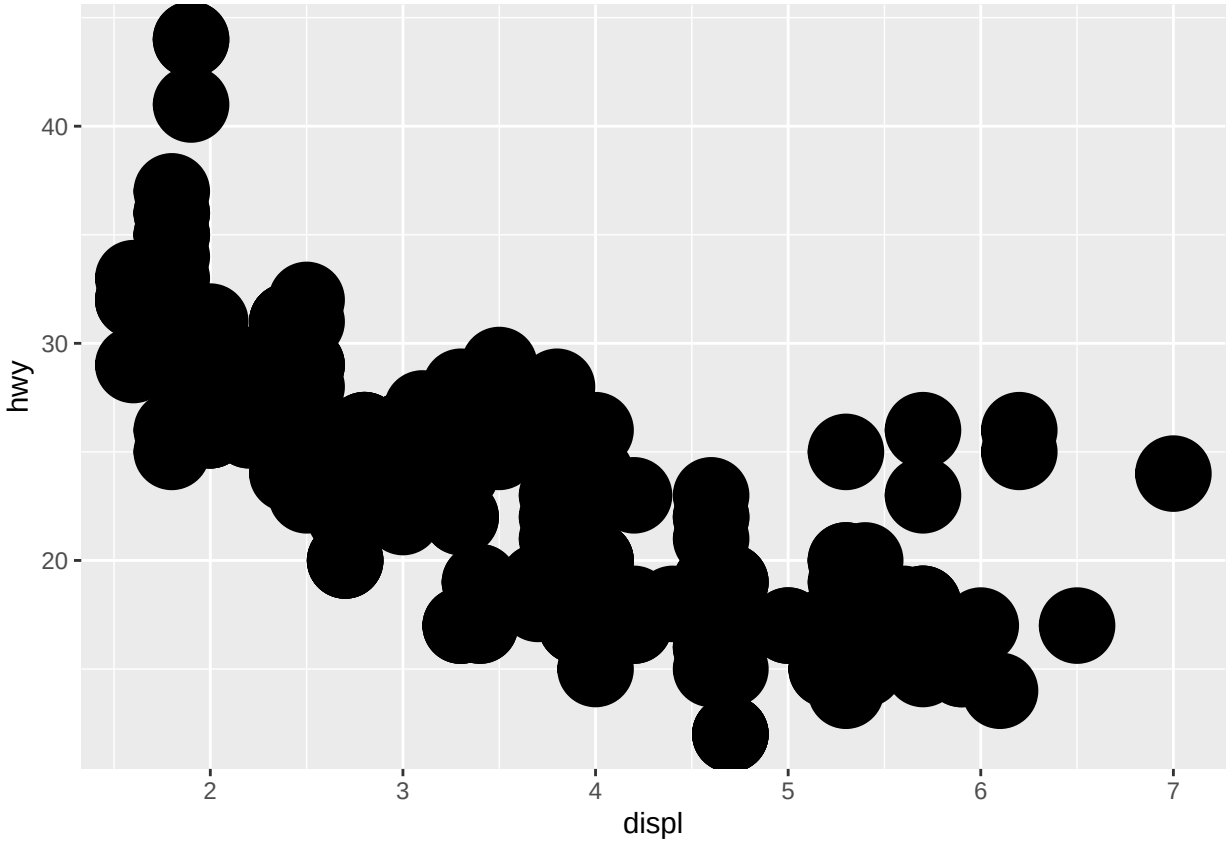


They get combined (is this the answer they were expecting?)

Exercise 5

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

```
ggplot2::ggplot(data=mpg)+
  geom_point(aes(x=displ, y=hwy, stroke=9))
```

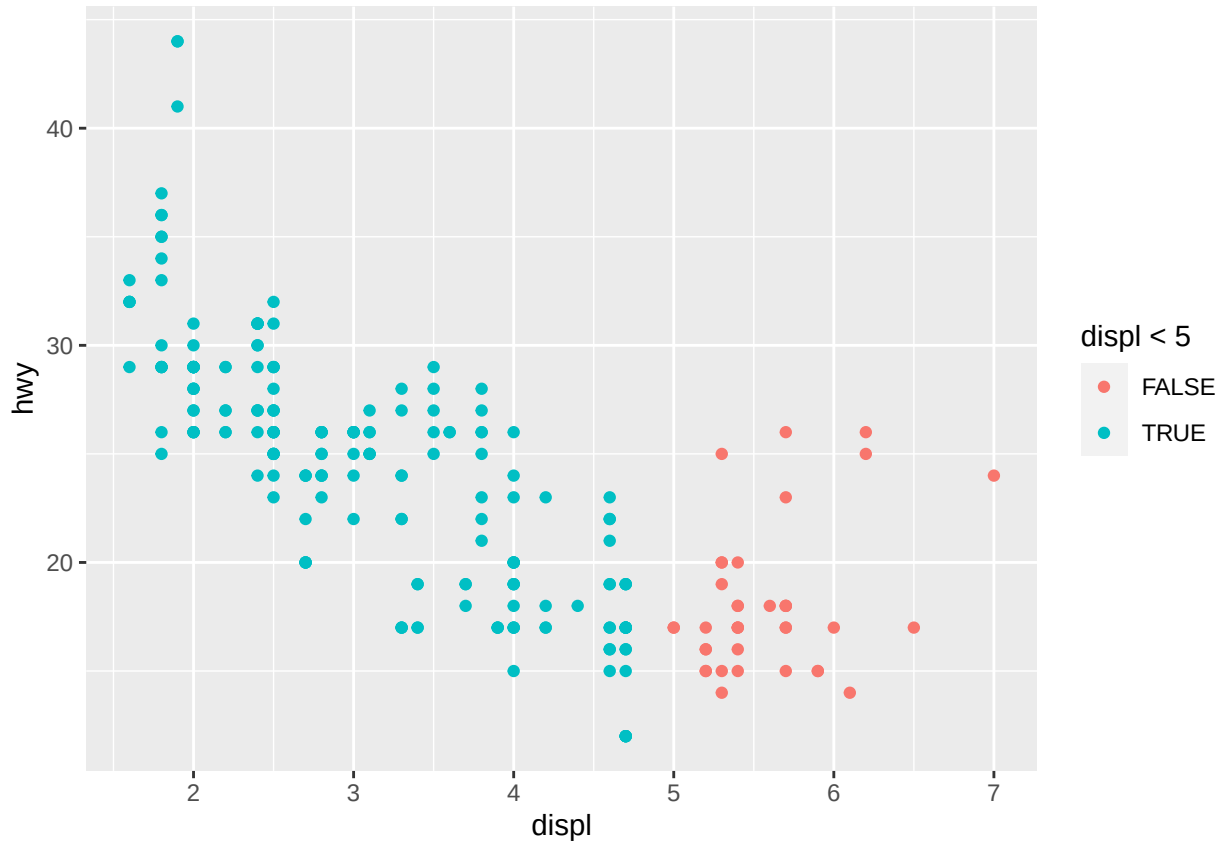


The stroke changes the width of the border for each observation.

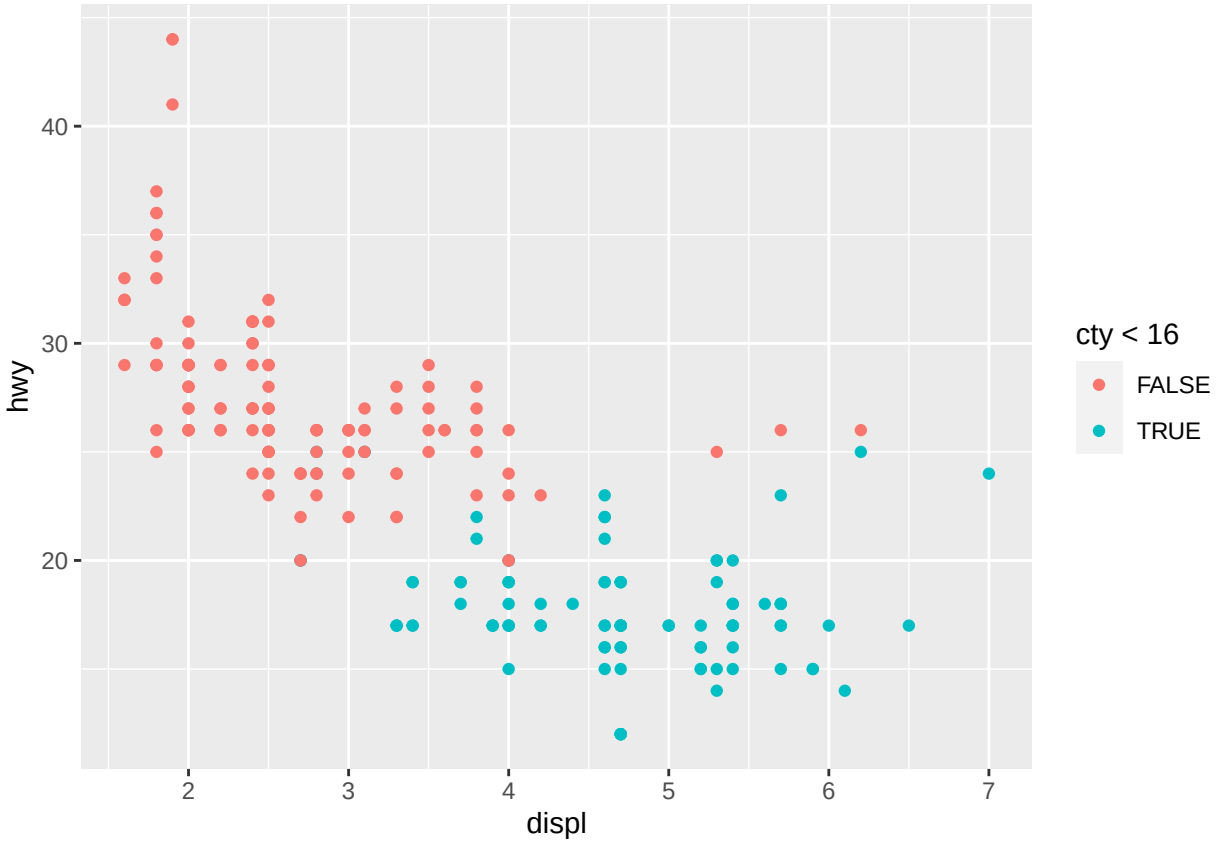
Exercise 6

What happens if you map an aesthetic to something other than a variable name, like $aes(color = displ < 5)$? Note, you'll also need to specify x and y.

```
ggplot2::ggplot(data=mpg)+
  geom_point(aes(x=displ,y=hwy, color=displ<5))
```



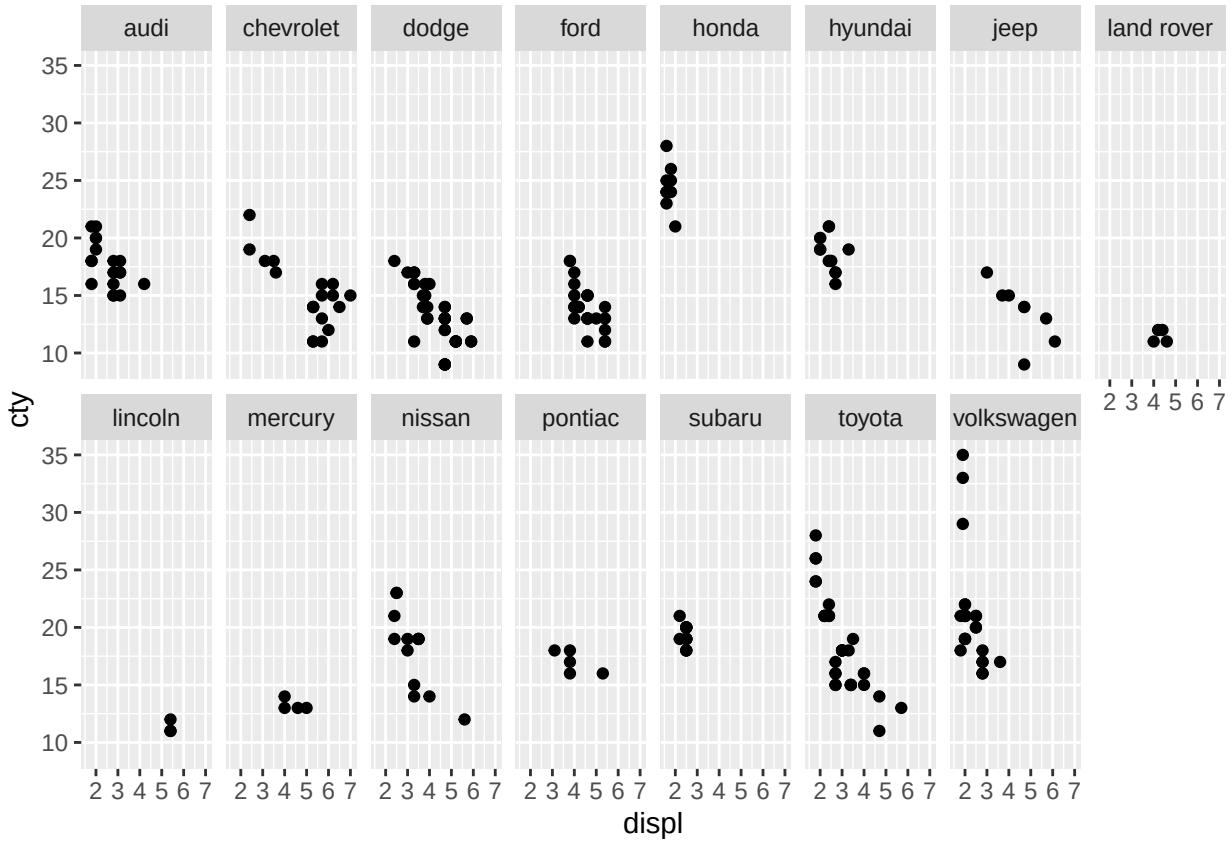

```
ggplot2::ggplot(data=mpg)+
  geom_point(aes(x=displ,y=hwy, color=cty<16))
```



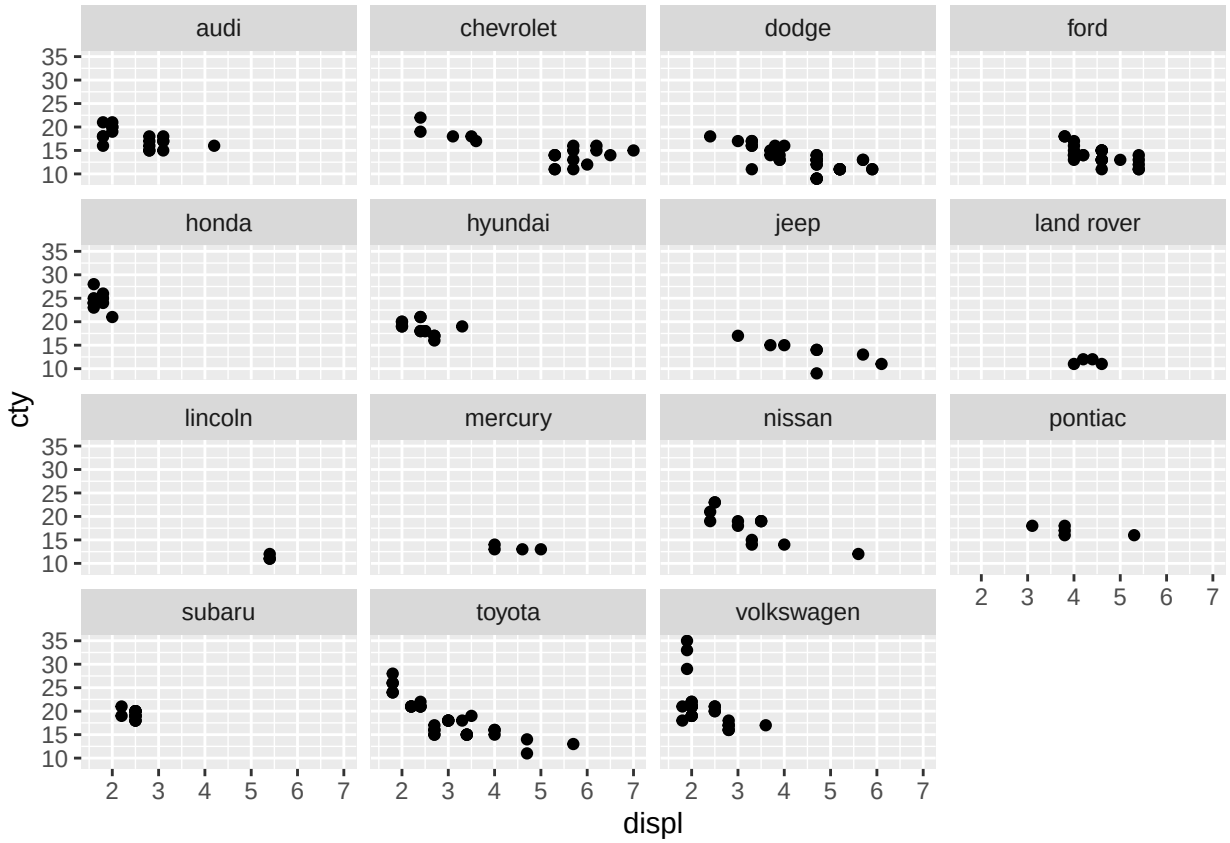
This way we will "create" two classes to be colored. Those cars with less than 5 cylinders will have a color, and the cars with more than 5 cylinders will have another.

Facets

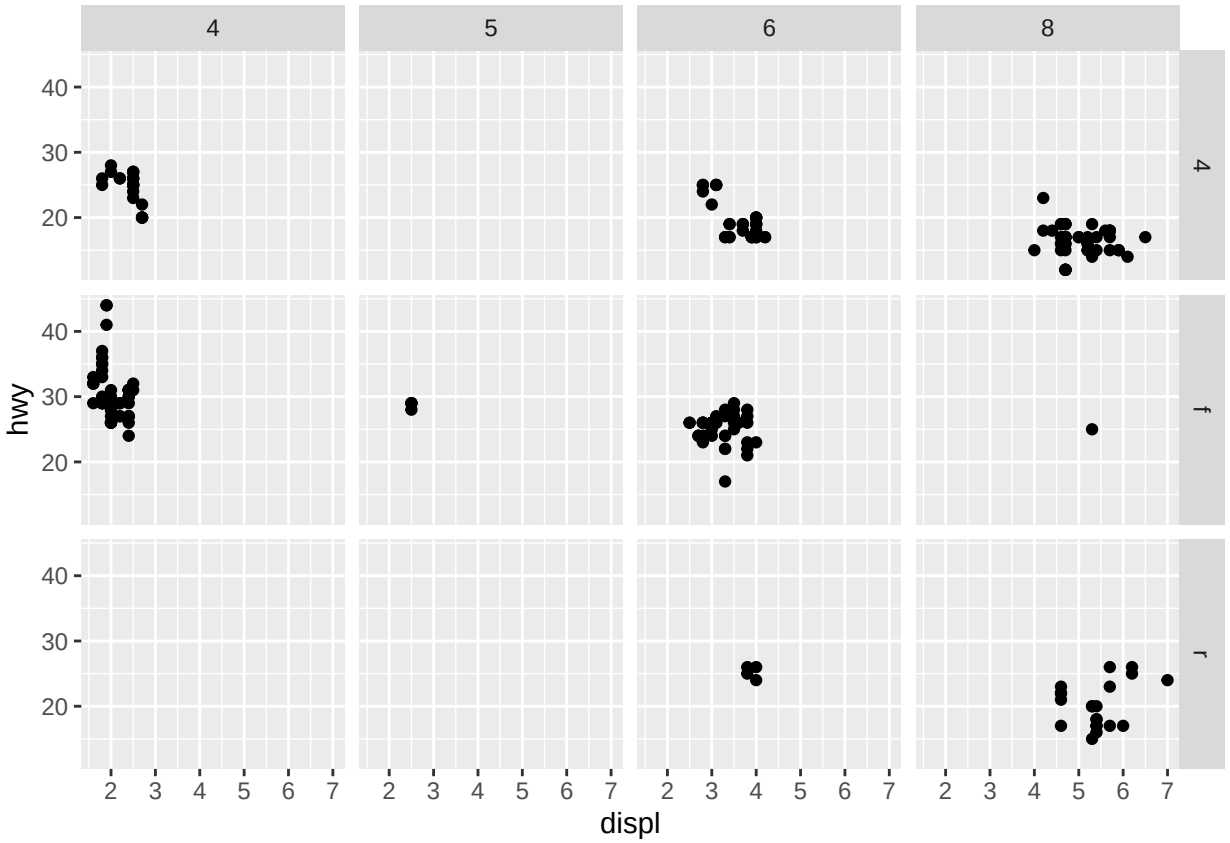
```
ggplot2::ggplot(data=mpg)+
  geom_point(aes(x=displ, y=cty))+
  facet_wrap(~manufacturer, nrow=2)
```




```
ggplot2::ggplot(data=mpg)+
  geom_point(aes(x=displ, y=cty))+
  facet_wrap(~manufacturer)
```

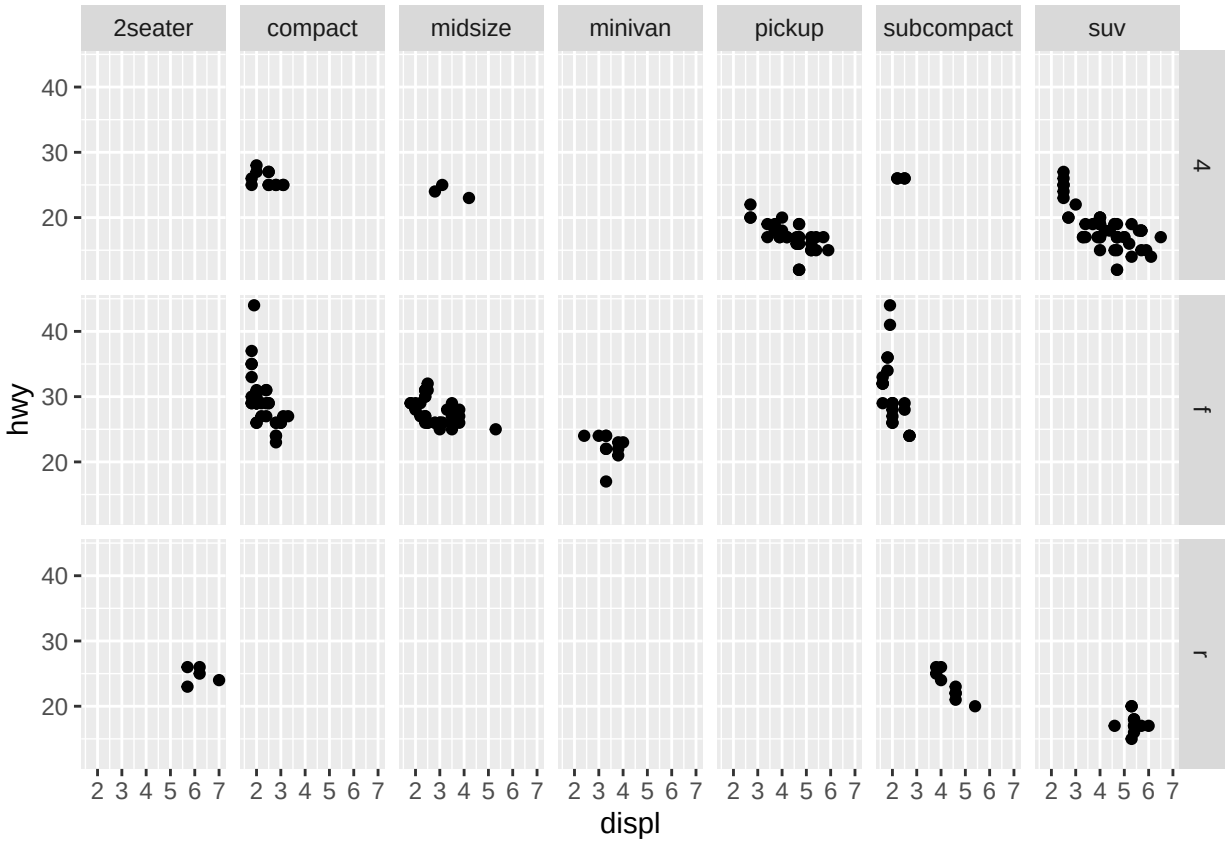



```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_grid(drv ~ cyl)
```



The *nrow* command within the facet is not necessary, only if you want to custom how all the graphs will be displayed. For this dataset I personally prefer leaving the default settings. Facet gridding for more variables:

```
ggplot2::ggplot(data=mpg)+
  geom_point(aes(x=displ, y=hwy))+
  facet_grid(drv~class)
```



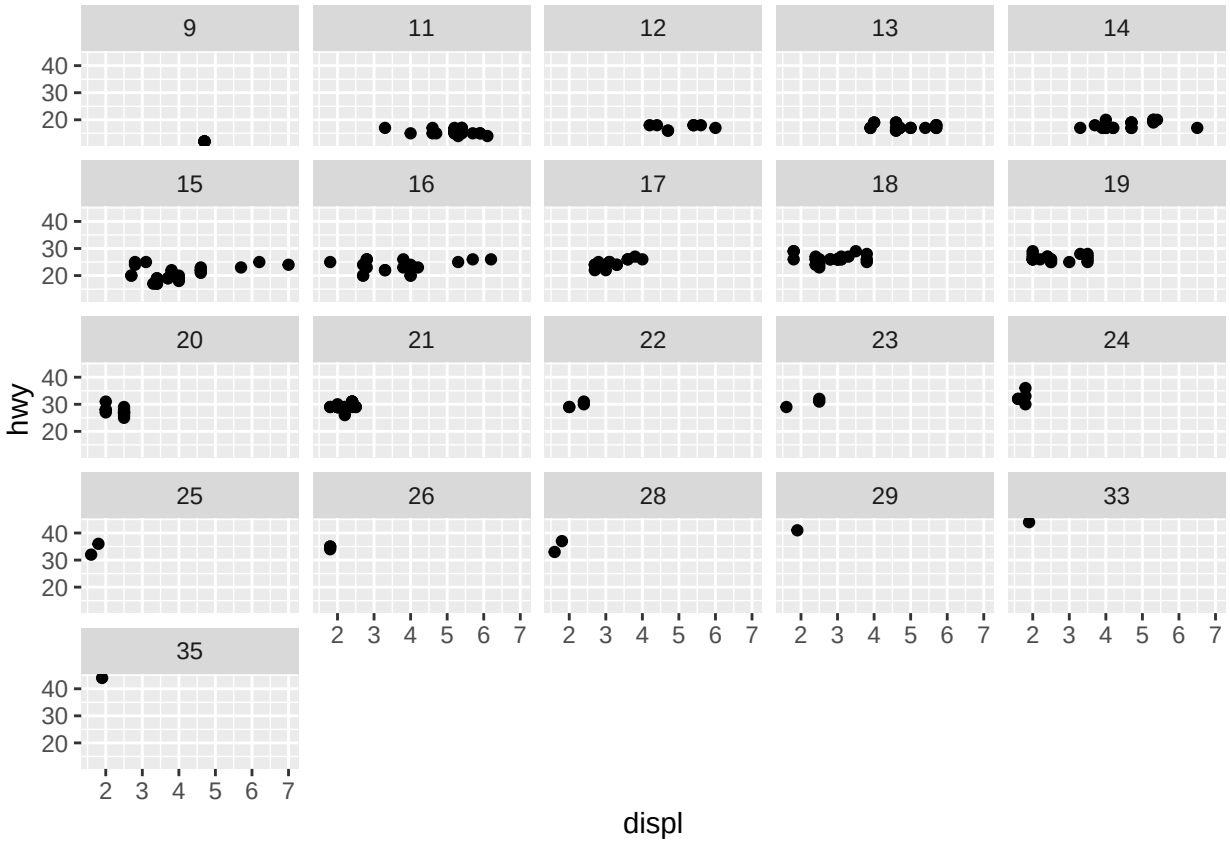
Personally, facet wrapping for more variables seems a little messy for visualization.

Exercises 3.5.1

Exercise 1

What happens if you facet on a continuous variable?

```
ggplot2::ggplot(data=mpg)+
  geom_point(aes(x=displ,y=hwy))+
  facet_wrap(~cty)
```

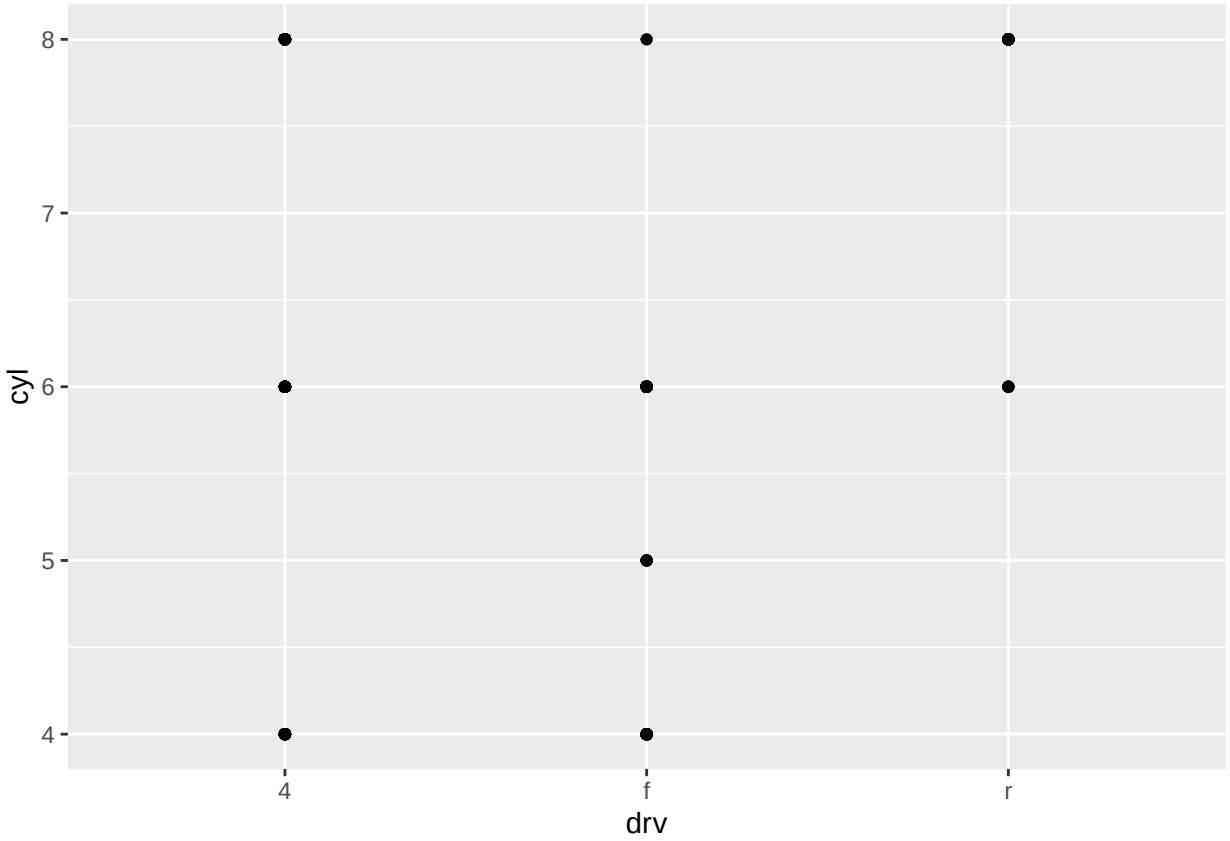


I this case i facet wrapped by the city mileage variable. What happens is that it creates too many graphs to account for the many values of a continuous variable. This leads to an worse visualization design.

Exercise 2

What do the empty cells in plot with $facet\_grid(drv \sim cyl)$ mean? How to they relate to this plot?

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

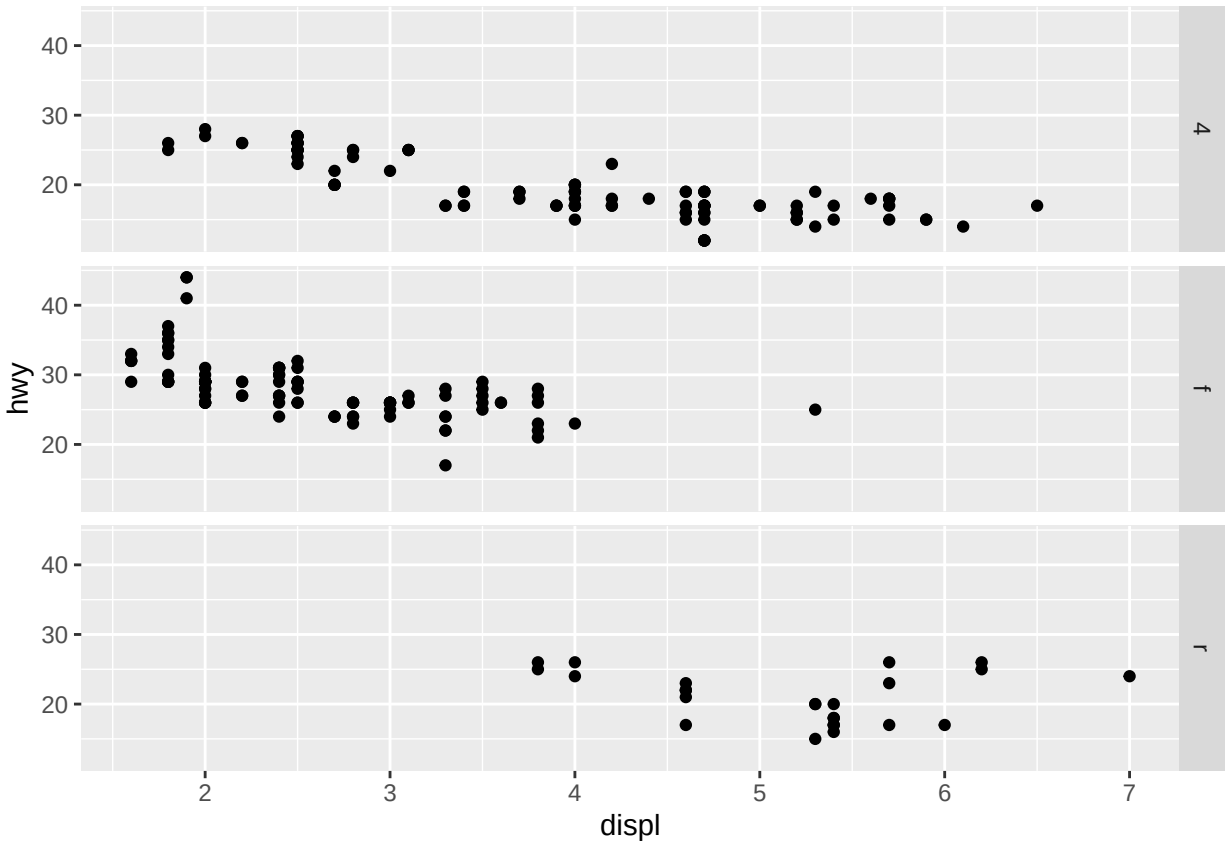


The empty cells tells us that they are a combination of characteristics of a car where we do not have any observation. In this example, it is the points where we do not "observe" an intersection between the axis.

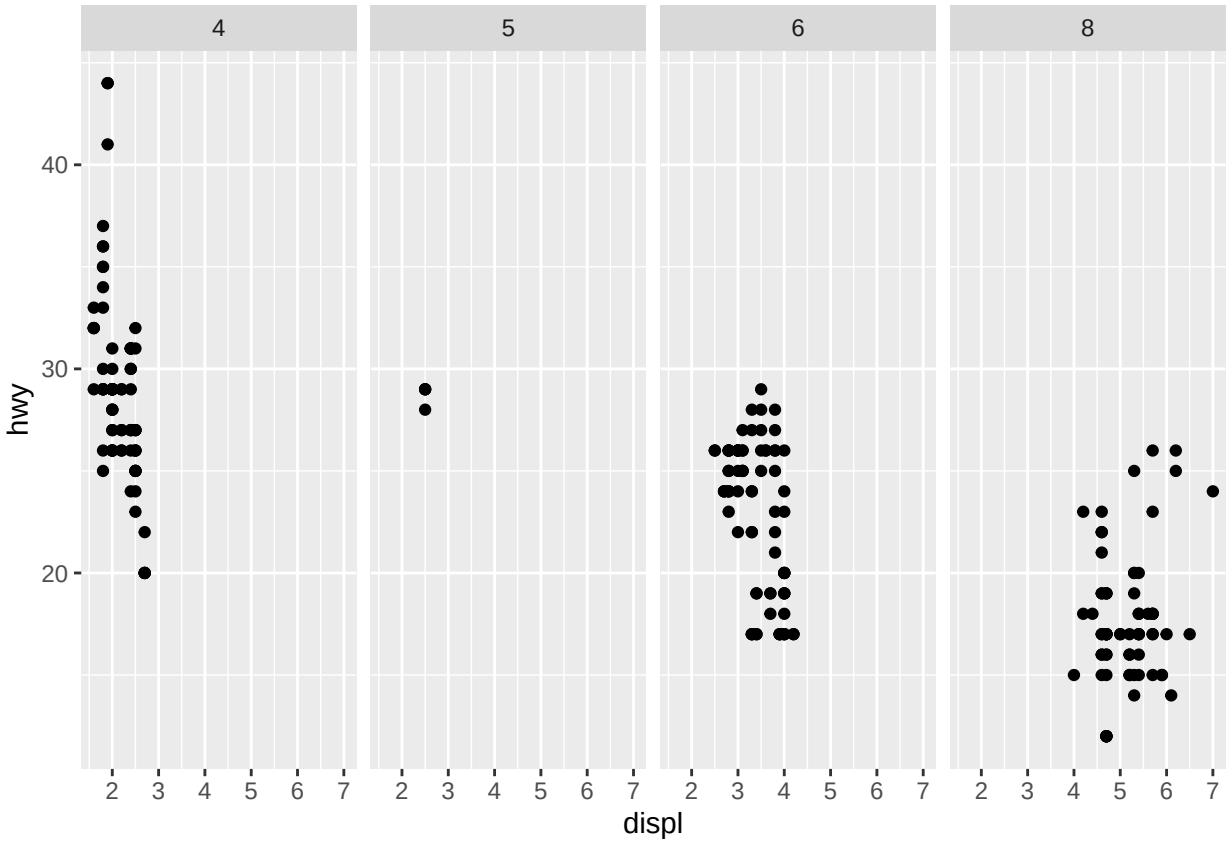
Exercise 3

What plot does the following code make? What does · do?

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




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



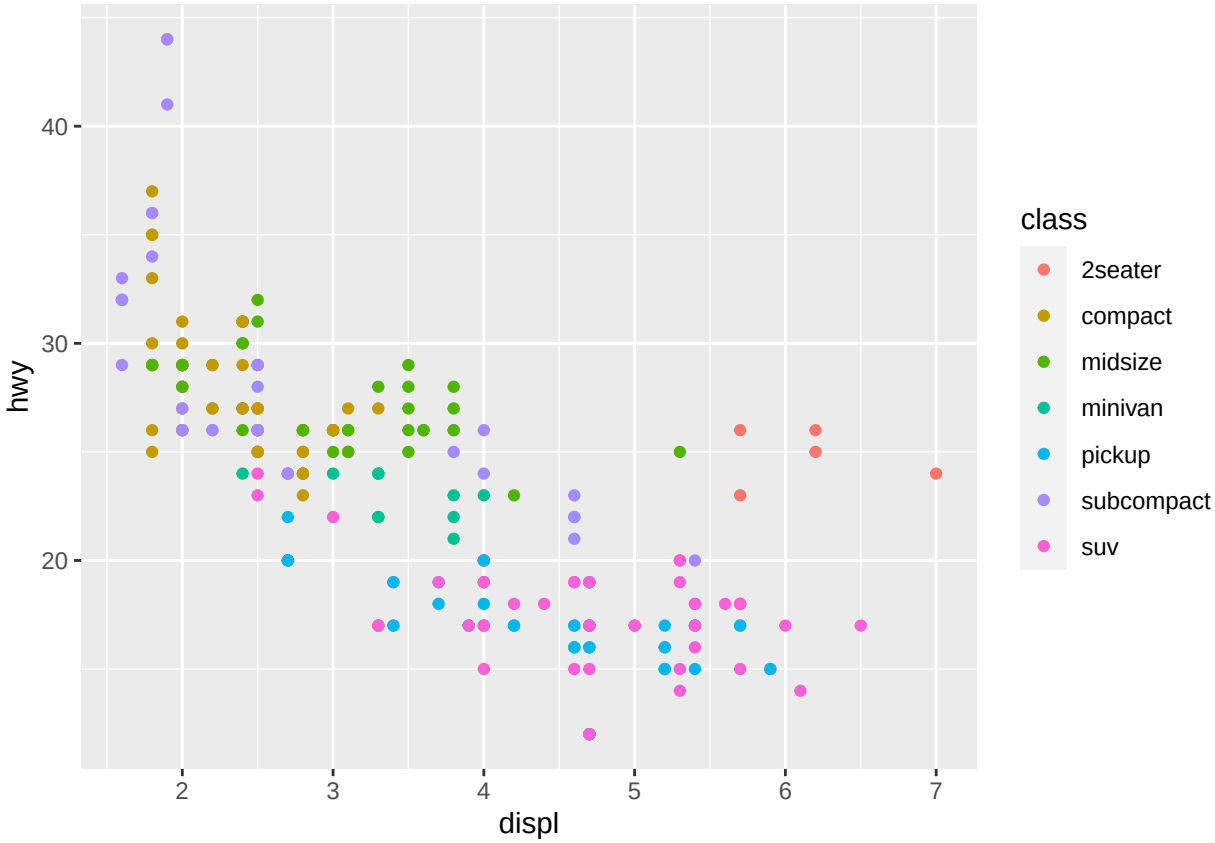
The point let us choose which axis we want to facet. In the expression $a. \sim b$, a is a function of b.

Exercise 4

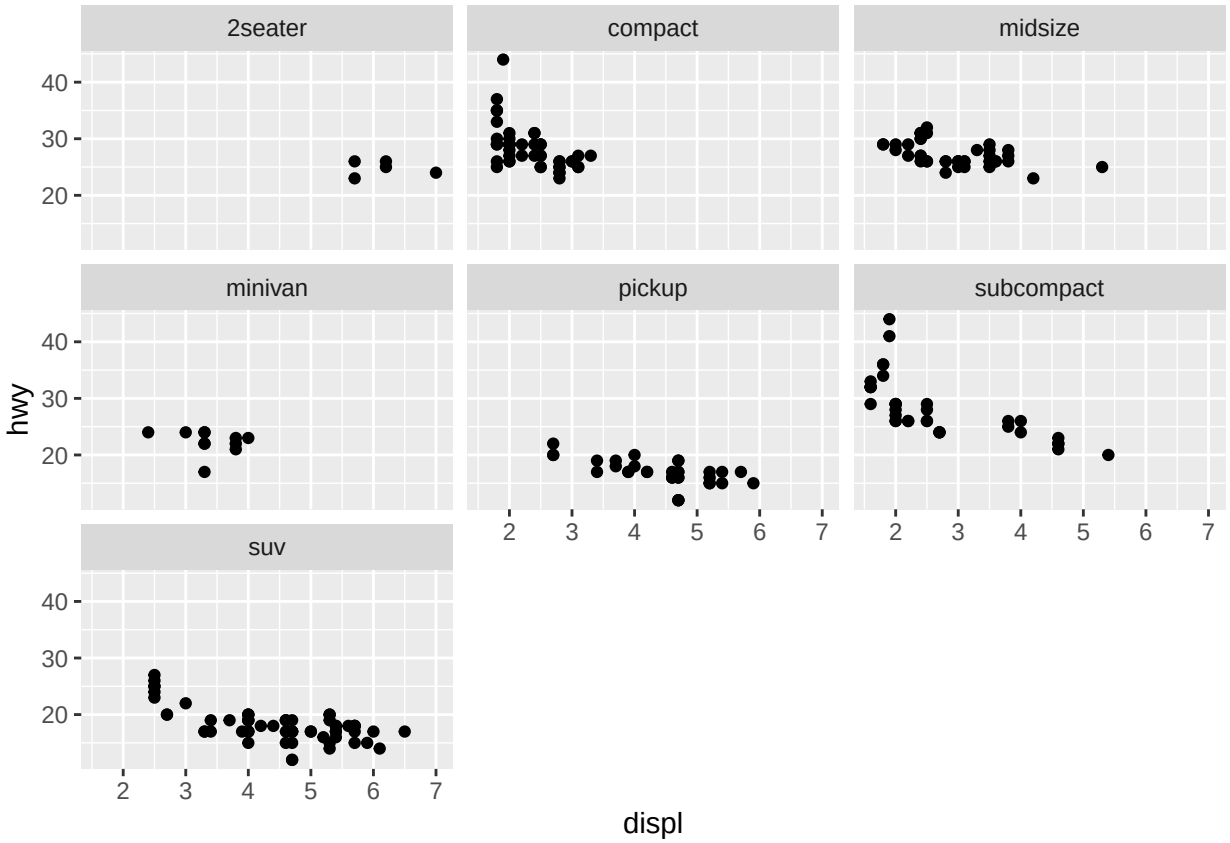
Take the first faceted plot in this section. What are the advantages to using the faceting instead of the color aesthetic? What are the disadvantages? How might the balance change if you had a larger data set?

Let me plot both graphs stated in the exercise:

```
ggplot2::ggplot(data=mpg)+
  geom_point(aes(x=displ, y=hwy, color=class))
```




```
ggplot2::ggplot(data=mpg)+
  geom_point(aes(x=displ, y=hwy))+
  facet_wrap(~class)
```



The advantage of facet wrapping is that we can infer relationships more easily within each class. In this case we can see how each class of cars behave related to its fuel consumption. The first disadvantage is that we lose some of the global behavior for the observations. The second is that when facet wrapping regarding to a variable with too many classes we end up with too many graphs, which may not helpful.

Exercise 5

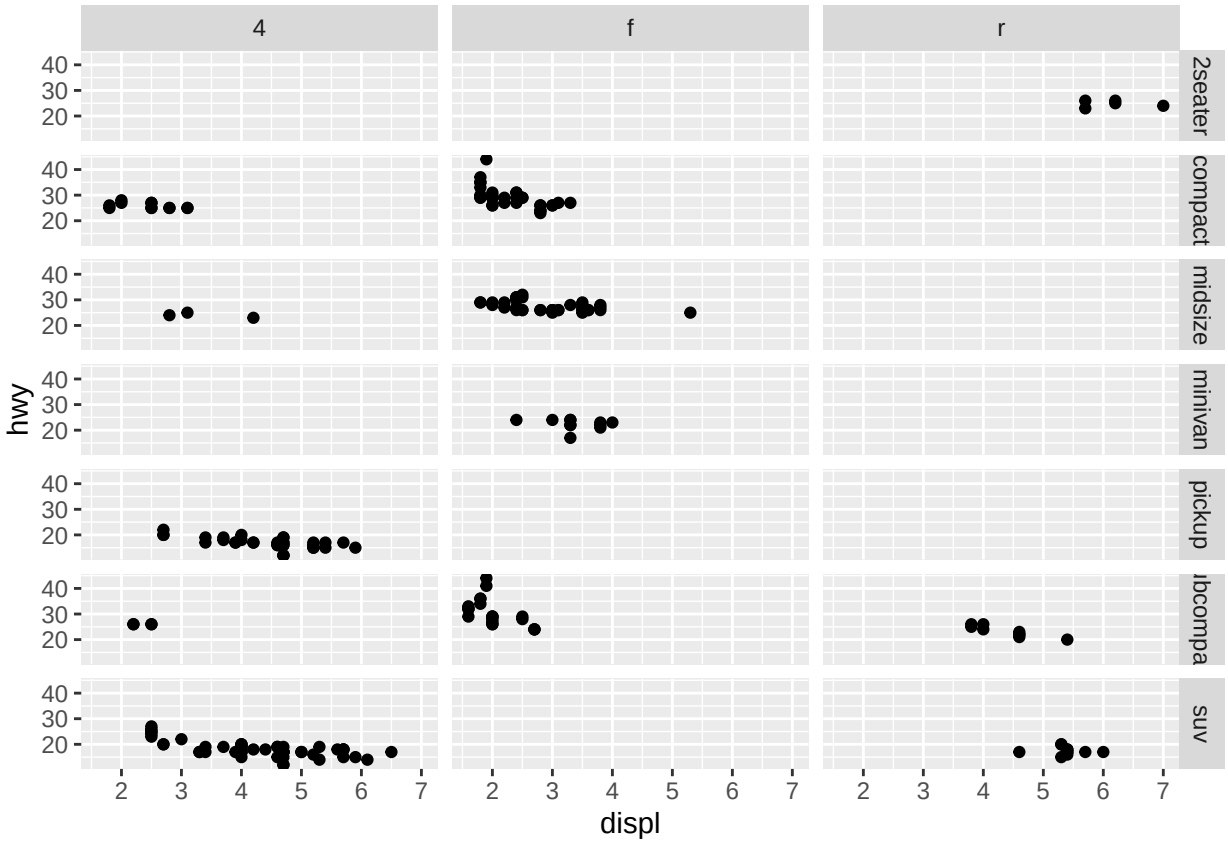
Read ?*facet_wrap*. What does *nrow* do? What does *ncol* do? What other options control the layout of the individual panels? Why doesn't *face_grid*() have *nrow* and *ncol* arguments?

These commands change the number of rows and columns which the cells will be displayed. Other options for changing individual panels are *scales* and *shrink*. The *facet_grid* does not have those commands since they can already be stated within the command, which will create rows and columns according to you variables.

Exercise 6

When using *facet_grid*() you should usually put the variable with more unique levels in the columns. Why?

```
ggplot2::ggplot(data = mpg) +
  geom_point(aes(x = displ, y = hwy)) +
  facet_grid(class~drv)
```



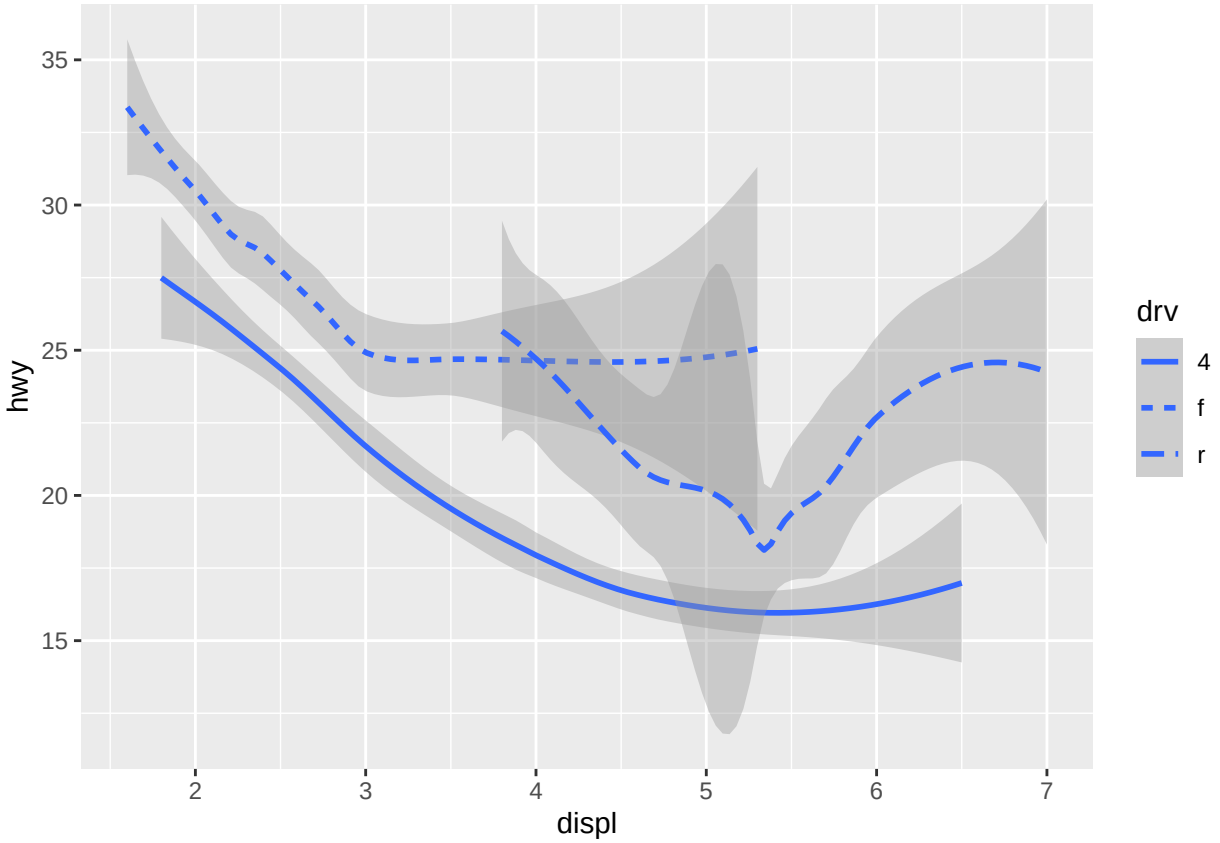
I think it leads to a better format for the graphs.

Geometric Objects

A scatter plot and a regression line represents different *geoms*

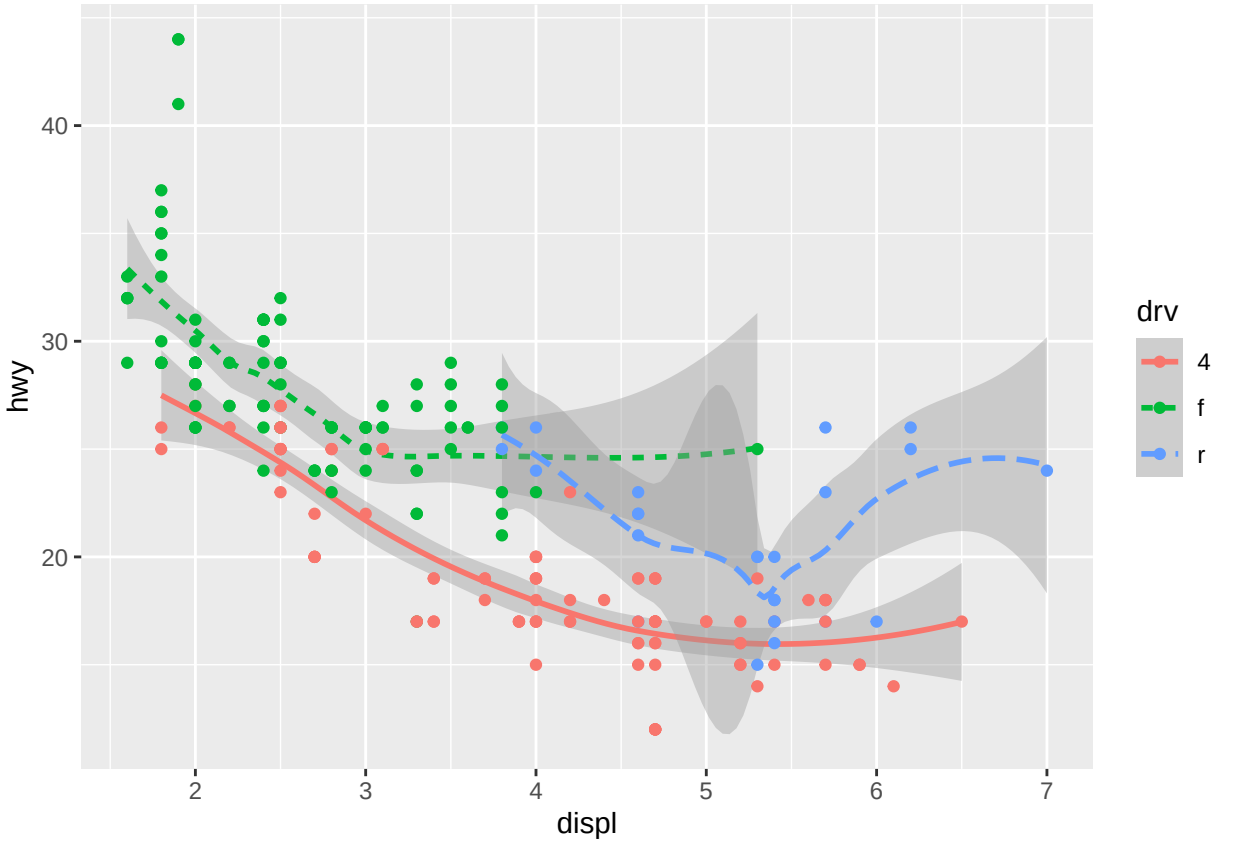
```
ggplot2::ggplot(data=mpg)+
  geom_smooth(aes(x=displ, y=hwy, linetype=drv))
```

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




```
ggplot2::ggplot(data=mpg, aes(x=displ, y=hwy))+
  geom_smooth(aes(color=drv, linetype=drv))+
  geom_point(aes(color=drv))
```

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




```
#this last graph is the one in the book
```

Putting the mapping into ggplot we basically set them globally for any geometry following. With that we can customize the color and types for each geometry individually without changing the map more than one time.

If we want to filter the graph, we filter in the geometry command. For example:

(pretty cool, heh).

Exercises 3.6.1

Exercise 1

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

Depends on the data. For frequency and distribution of a certain variable (like test grades for example) a histogram would be a good choice.

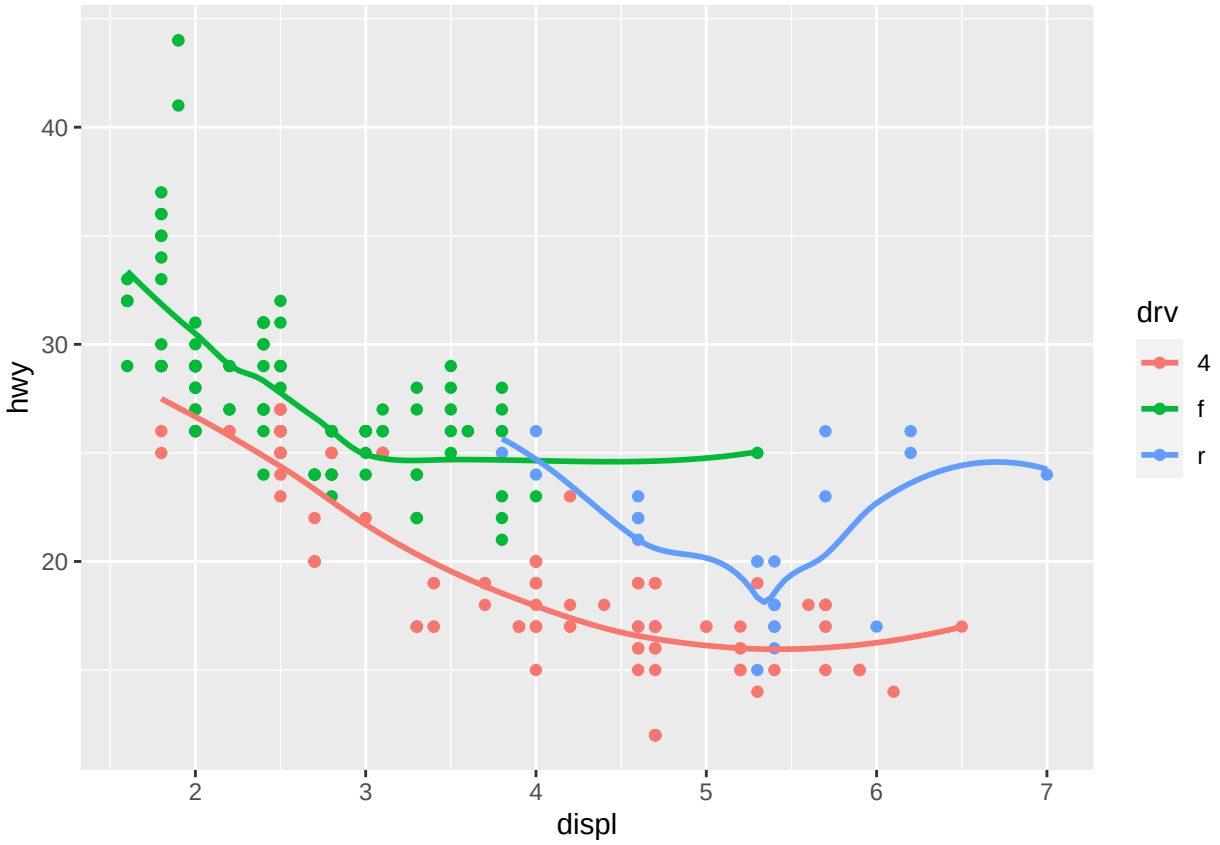
Exercise 2

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

This code will create a graph with three different regression lines, each one with a different color regarding the drive train group but with no shaded region indicating the standard error for the regression. Also a scatter plot will be created segmented by color also for each drive train.

```
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'
```

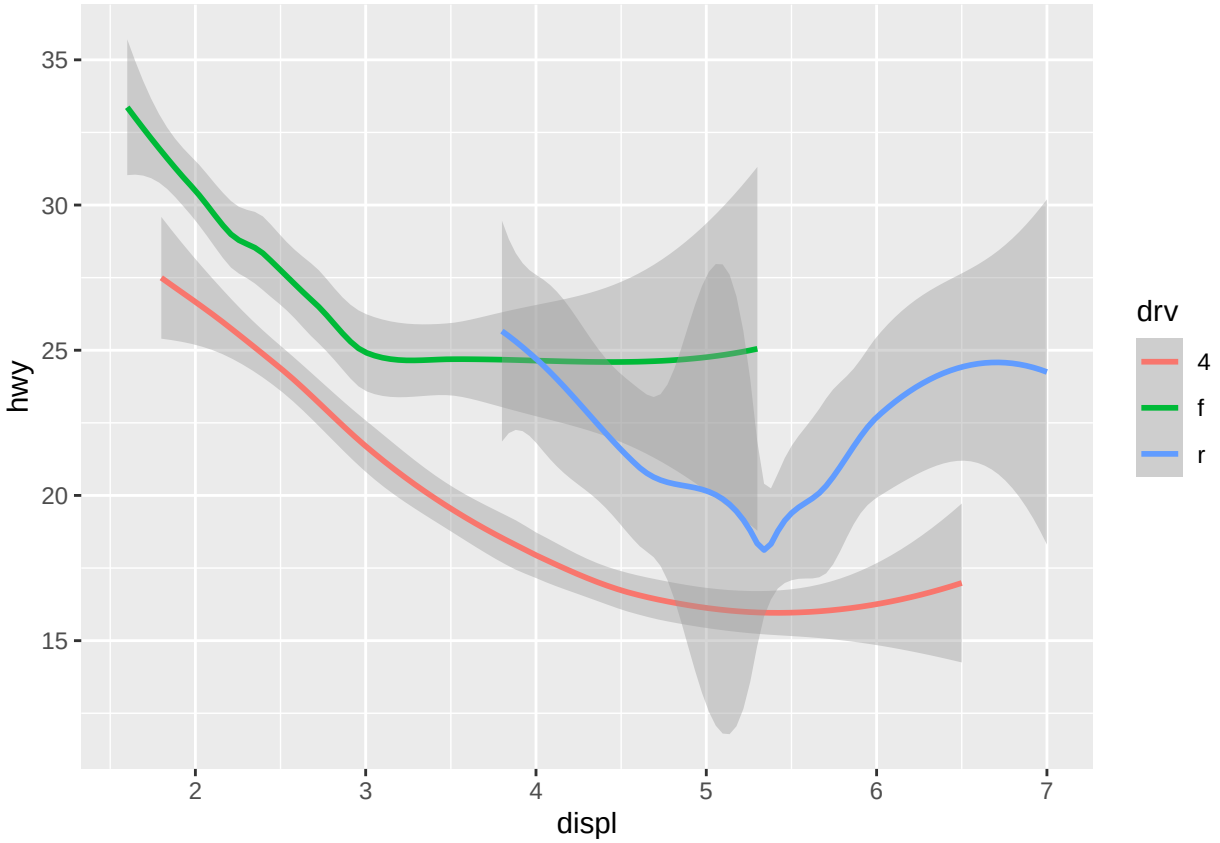


Exercise 3

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

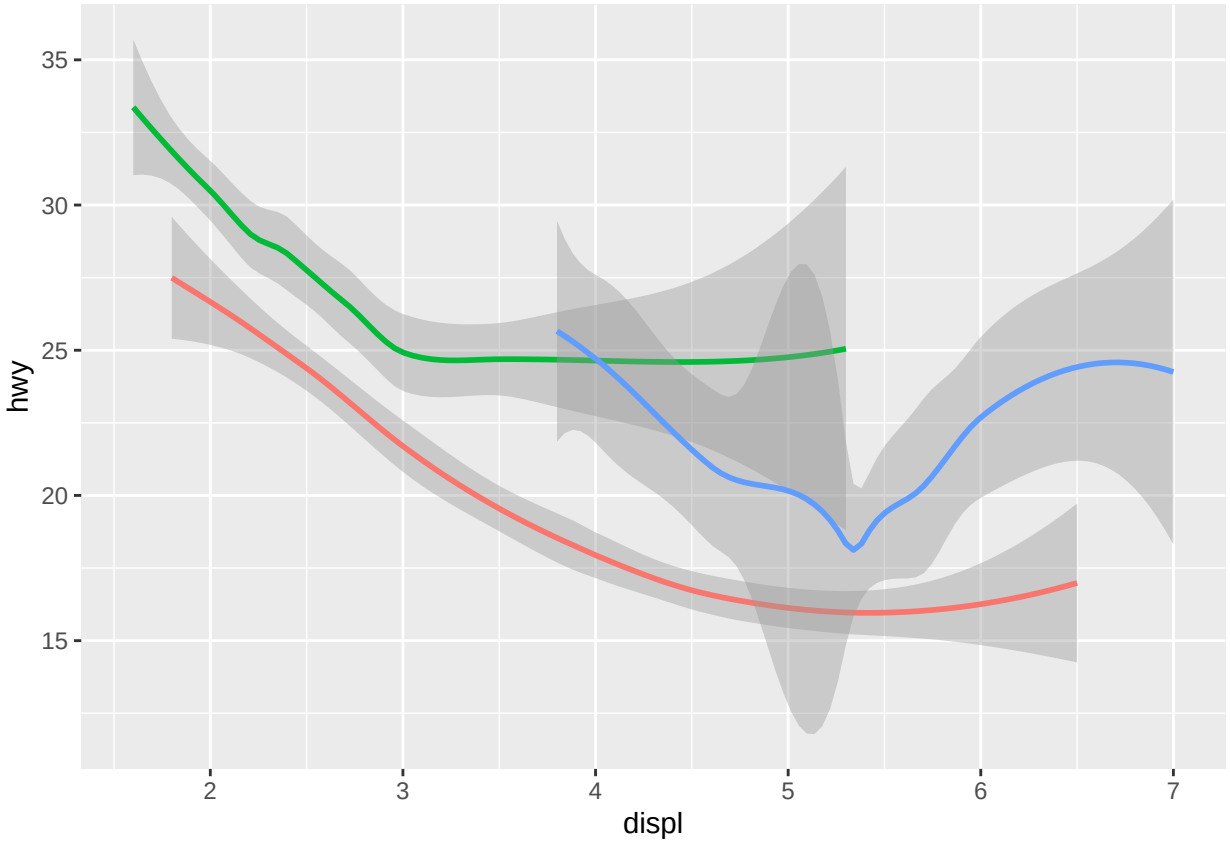
```
ggplot2::ggplot(data = mpg) +
  geom_smooth(
    aes(x = displ, y = hwy, color = drv),
  )
```

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




```
ggplot2::ggplot(data = mpg) +
  geom_smooth(
    aes(x = displ, y = hwy, color = drv),
    show.legend = FALSE
  )
```

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



This command just omits the different classes in the variable which was "segmented". In the book it was used to make the graph a little simpler, since this information was implied in the text.

Exercise 4

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

se stands for standard deviation. This argument plots the region within the error interval in the regression.

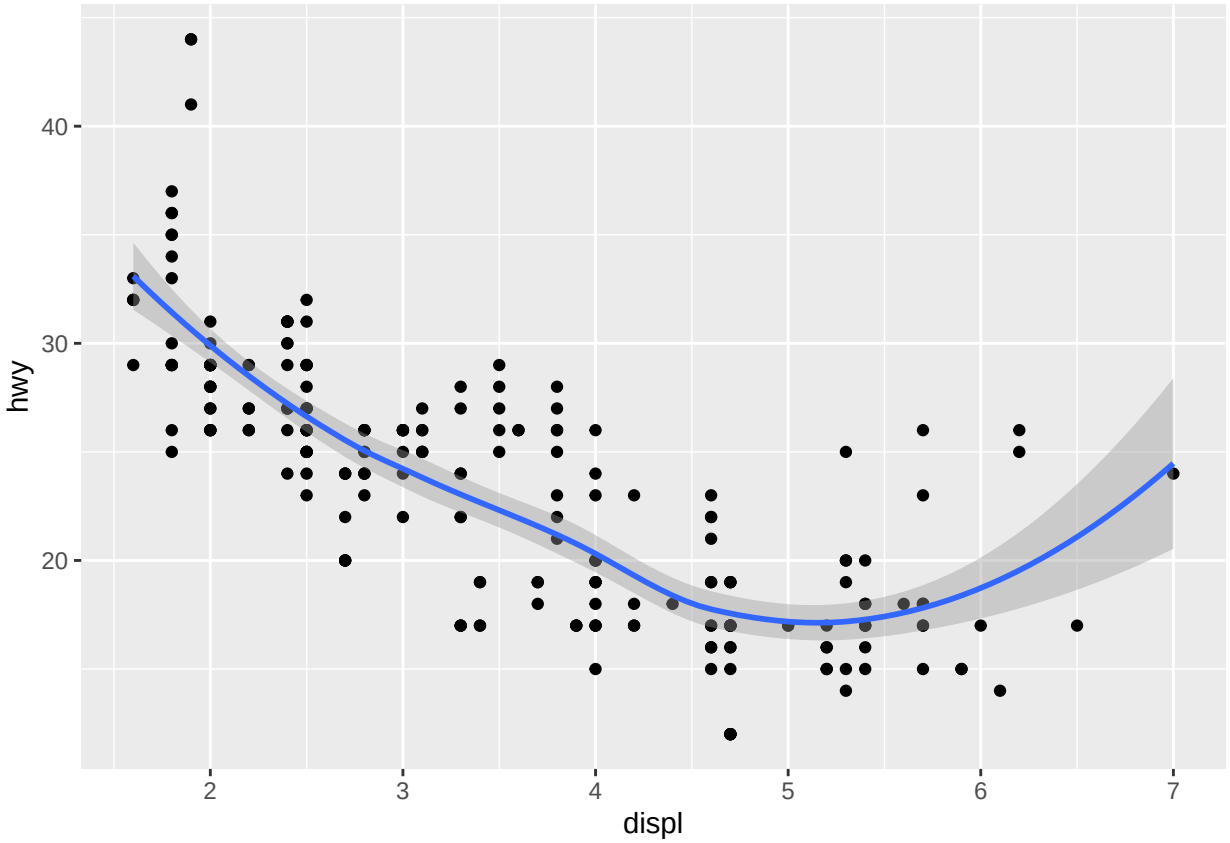
Exercise 5

Will these two graphs look different? Why/why not?

Both codes will lead to the same graph. However the first one is a little simpler since the variables are being defined in a global way (inside the *ggplot* command) while the second one we must state the variables at each new geometry added.

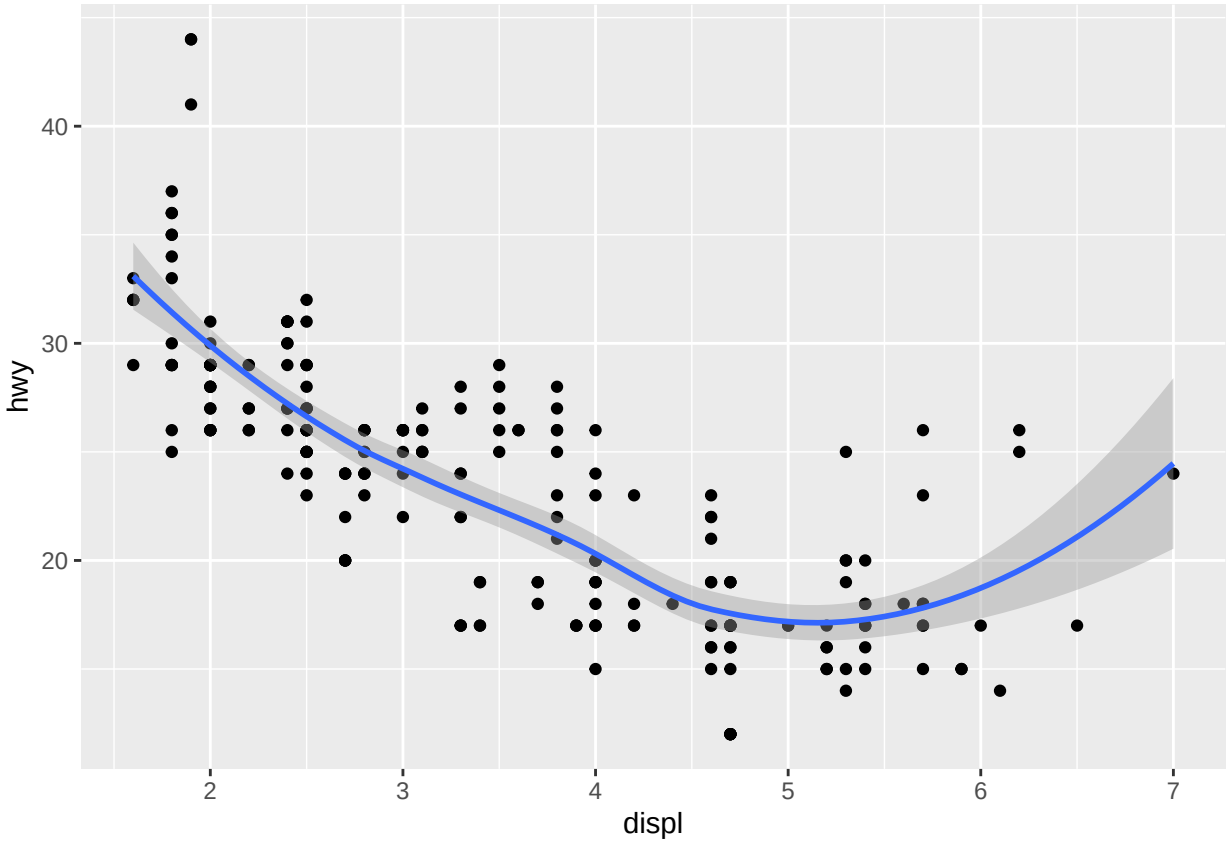
```
ggplot2::ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth()
```

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




```
ggplot2::ggplot() +
  geom_point(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_smooth(data = mpg, mapping = aes(x = displ, y = hwy))
```

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

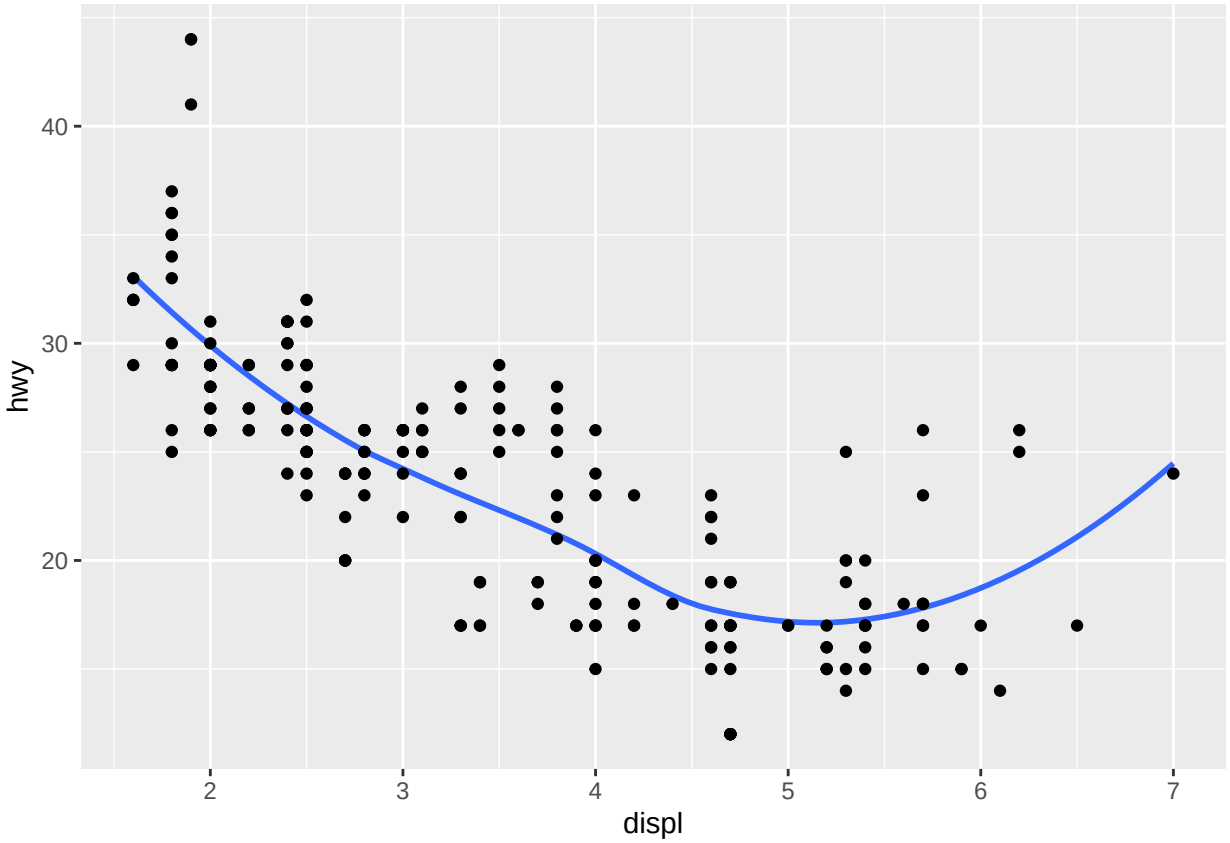


Exercise 6

Recreate the R code necessary to generate the following graphs.

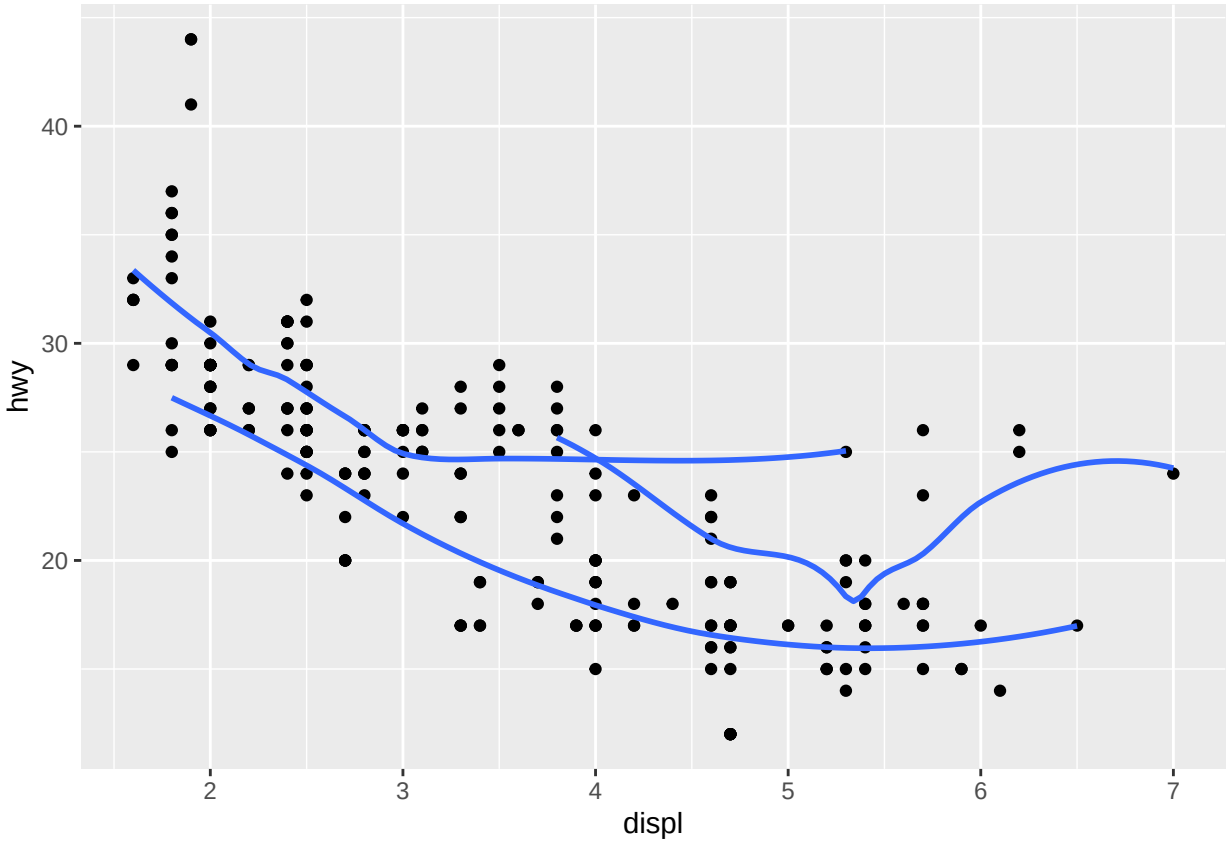
```
ggplot2::ggplot(data=mpg, aes(x=displ, y=hwy))+
  geom_smooth(se=FALSE)+
  geom_point()
```

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



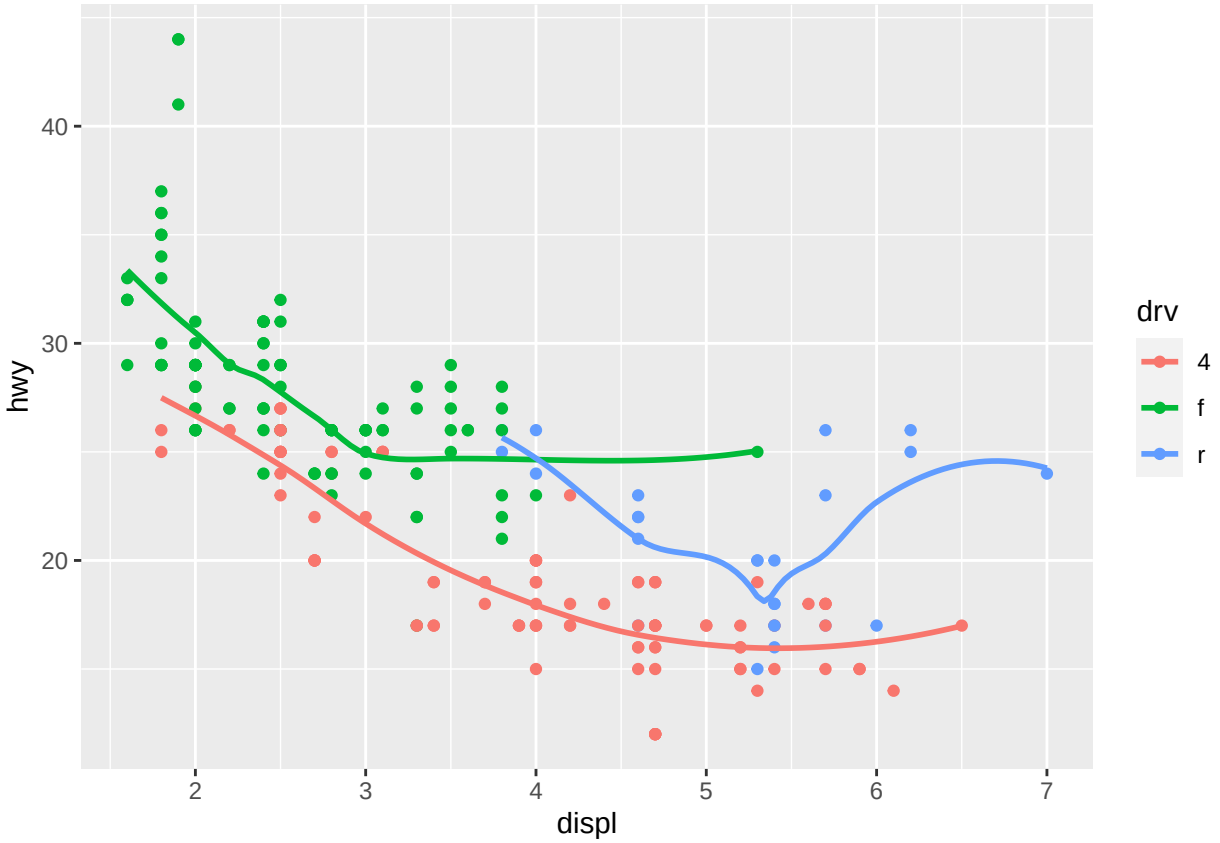

```
ggplot2::ggplot(data=mpg, aes(x=displ, y=hwy))+
  geom_point()+
  geom_smooth(aes(group=drv), se=FALSE)
```

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



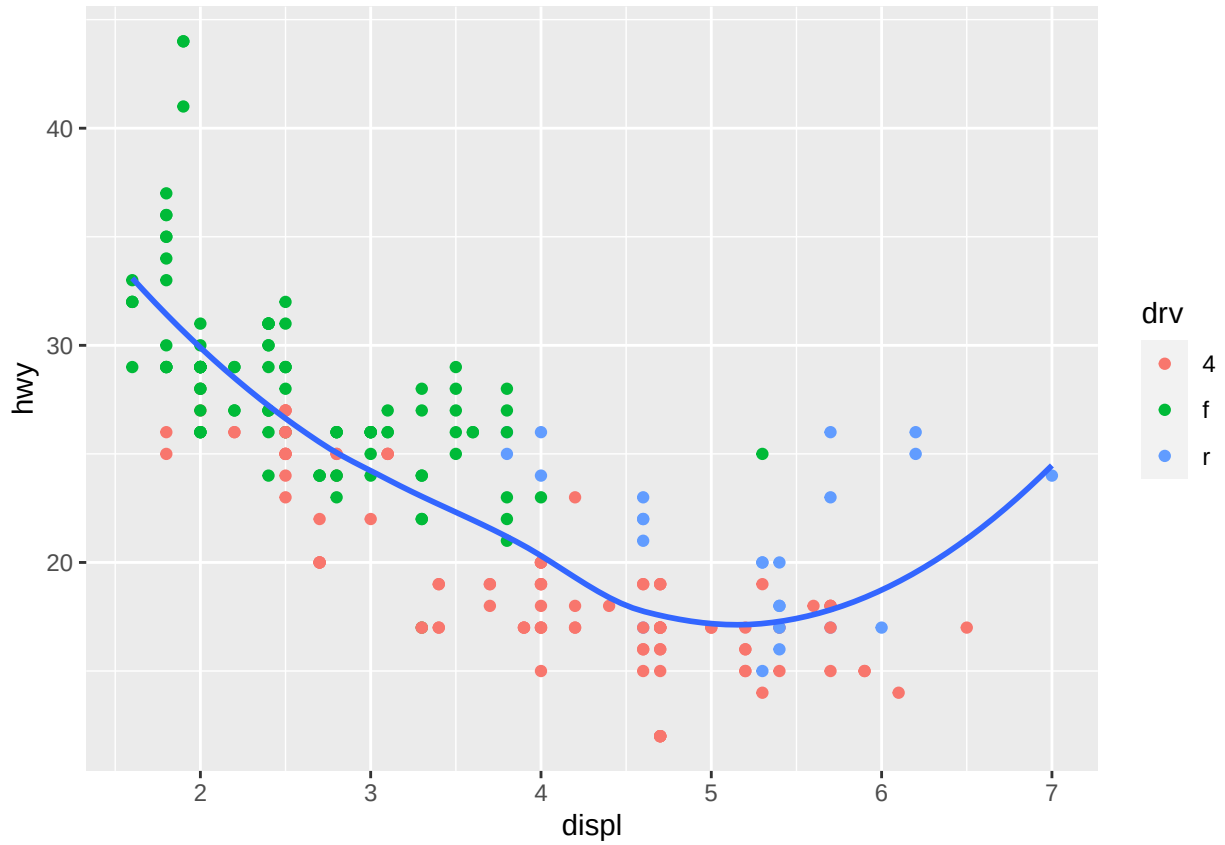

```
ggplot2::ggplot(data=mpg, aes(x=displ, y=hwy))+
  geom_point(aes(color=drv))+
  geom_smooth(aes(color=drv), se=FALSE)
```

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



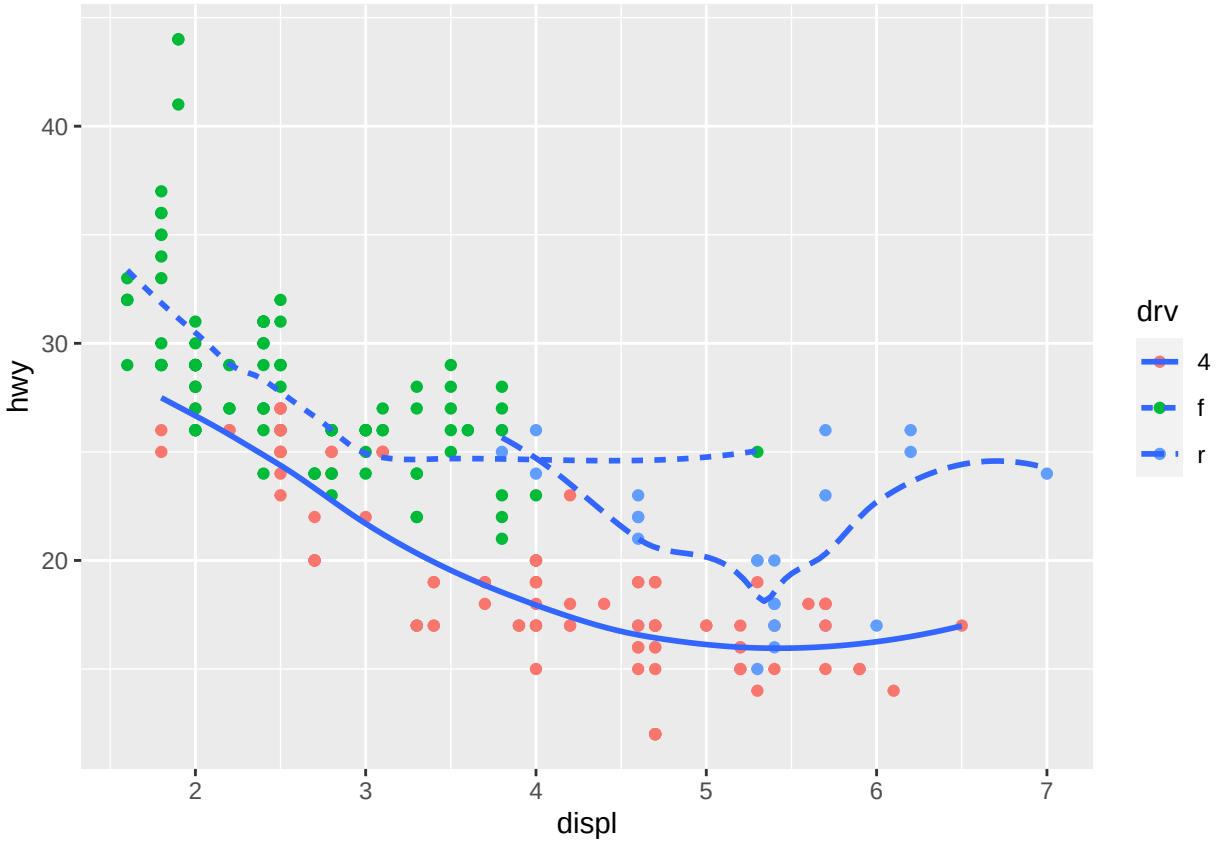

```
ggplot2::ggplot(data = mpg, aes(x=displ, y=hwy))+
  geom_point(aes(color=drv))+
  geom_smooth(se=FALSE)
```

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

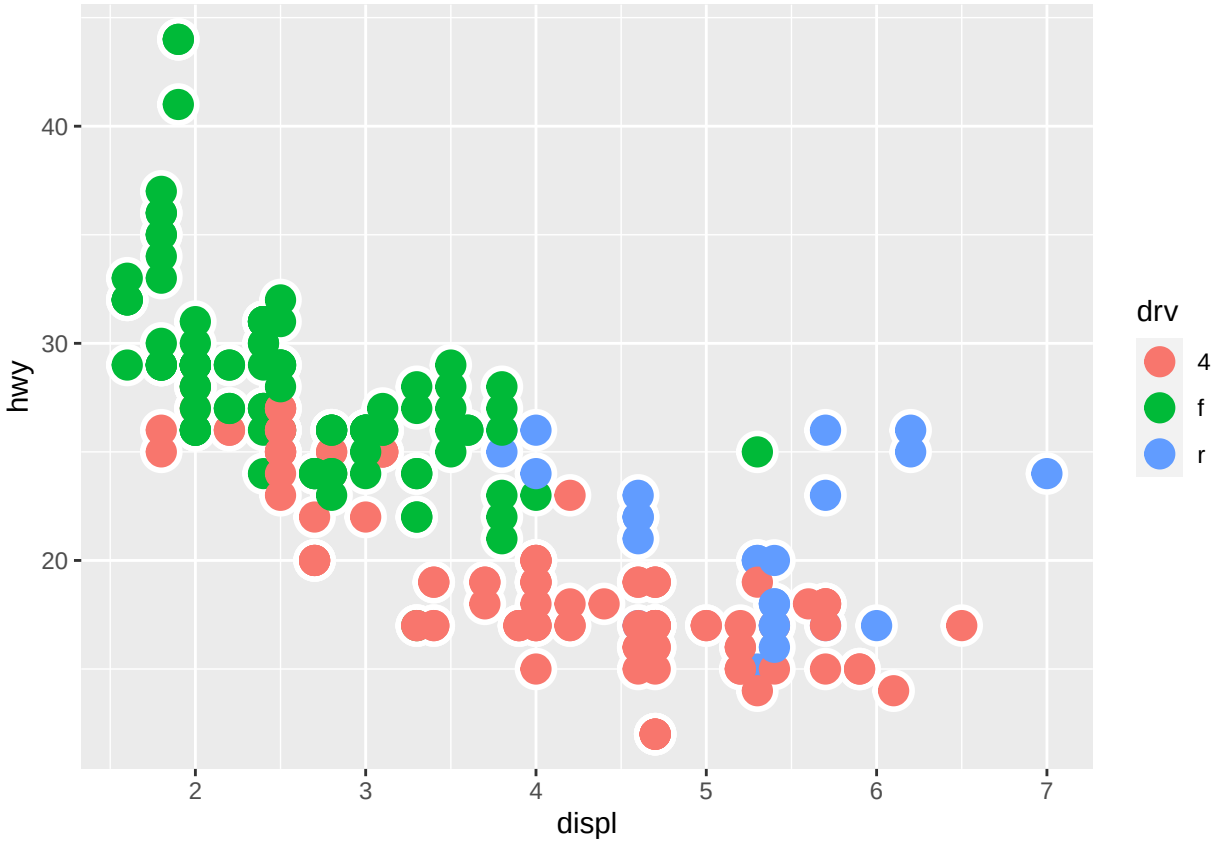



```
ggplot2::ggplot(mpg, aes(x=displ, y=hwy))+
  geom_point(aes(color=drv))+
  geom_smooth(aes(linetype=drv), se=FALSE)
```

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



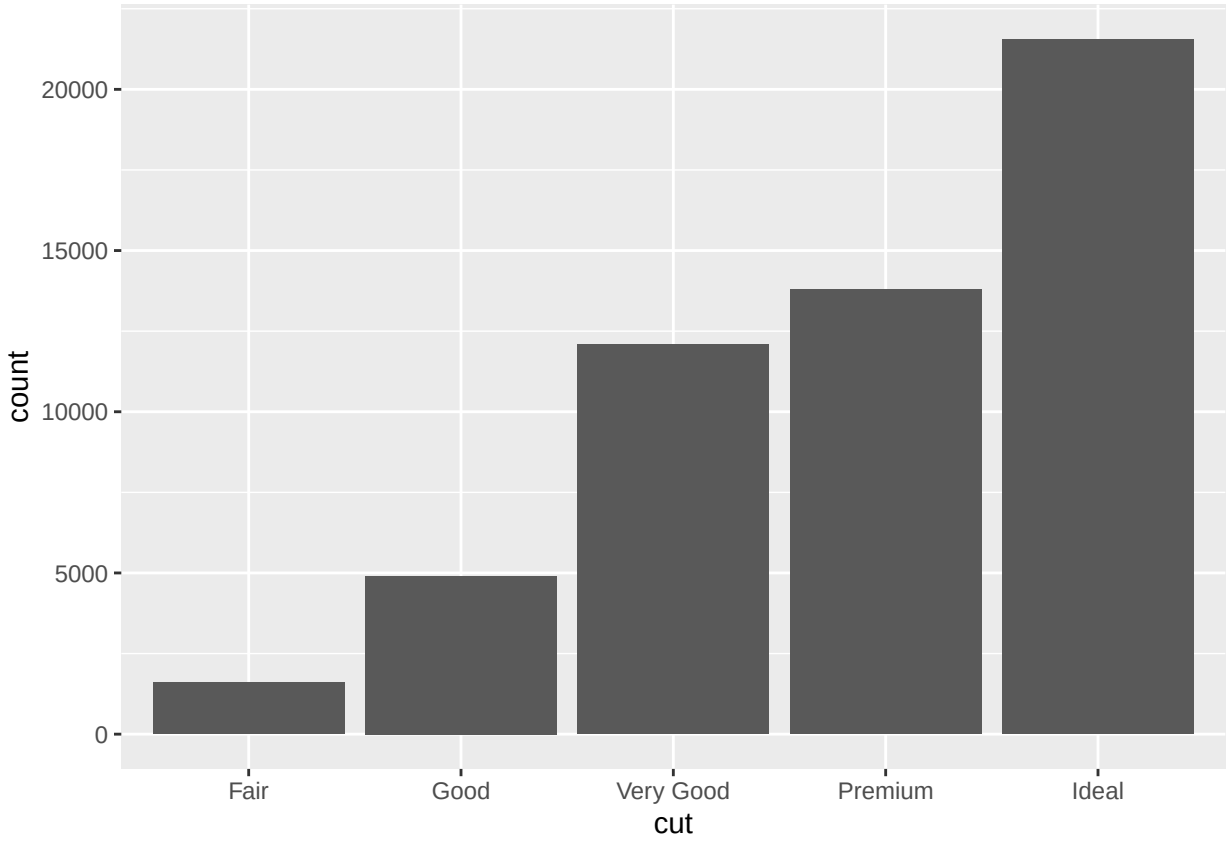

```
ggplot2::ggplot(mpg, aes(x=displ, y=hwy))+
  geom_point(stroke = 4.5, color="white")+
  geom_point(aes(color=drv), stroke=3)
```



For the last one we just over plot the points. But remember that geometry works as layers, so keep in mind what do you want to plot. (maybe this is not the optimal way to make what the books wants, but that's life I guess.)

Statistical Transformations

```
ggplot2::ggplot(data=diamonds)+
  geom_bar(
    aes(x=cut)
    )
```

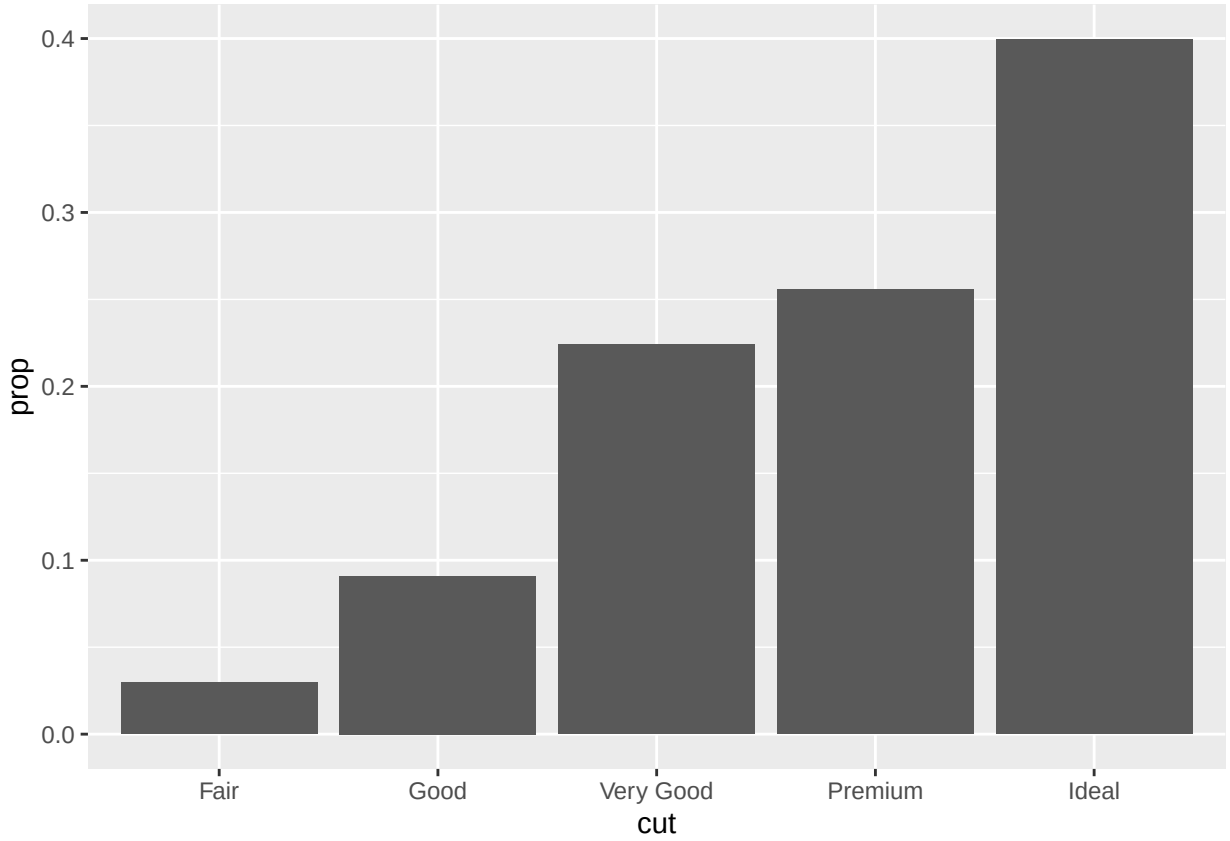


Notice that the geomtry adds a new value, which is the count number for each class. The *stat* is responsible for calculation underlying these geometries. There is a "middle" data set involved in the plot. We can check the *stat* by looking at the documentation for each geometry, within the computed variables section.

Every geometry has a default stat count!

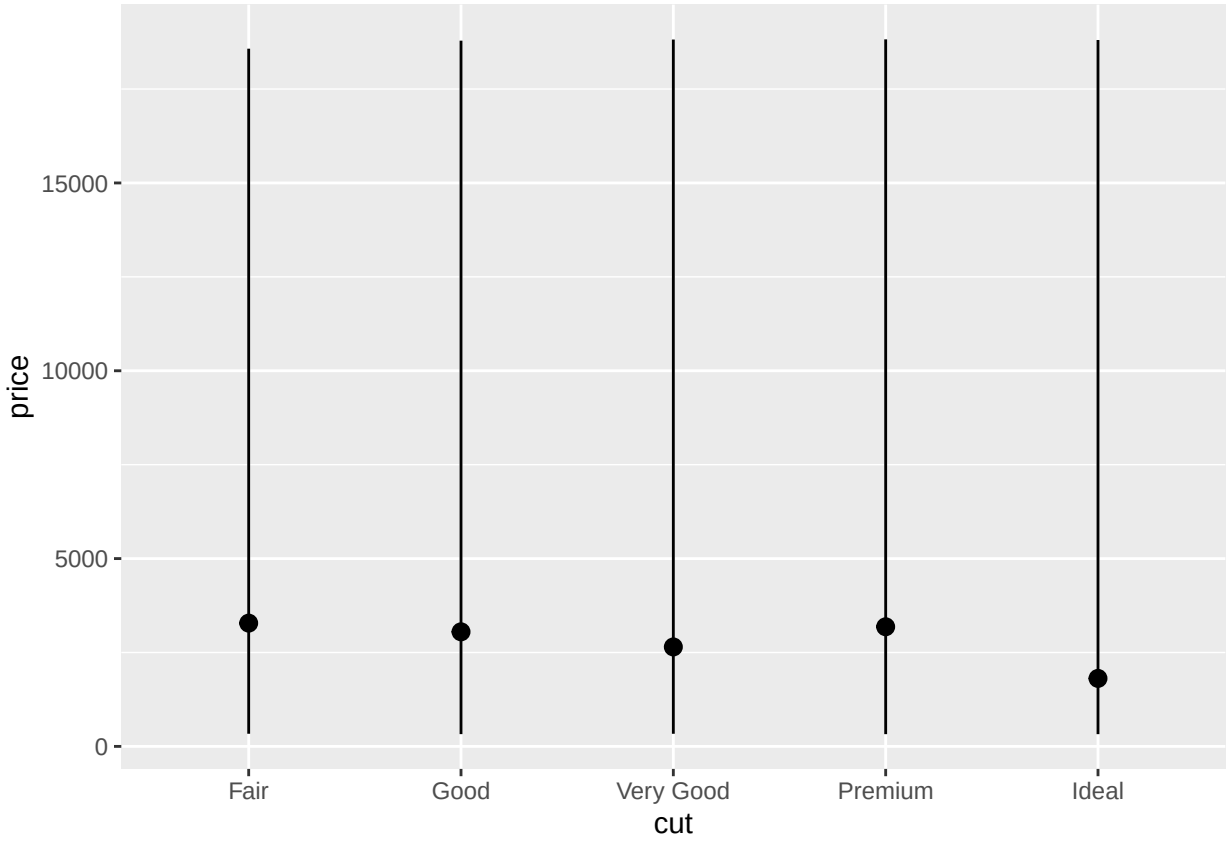
In the following code, we override the count with the proportion:

```
ggplot2::ggplot(data=diamonds)+
  geom_bar(
    aes(x=cut, y=stat(prop), group=1)
    )
```



Using the summary stat to see other information:

```
ggplot2::ggplot(data=diamonds)+
  stat_summary(
    aes(x=cut,y=price),
    fun.min = min,
    fun.max = max,
    fun = median
  )
```



Exercises 3.7.1

1

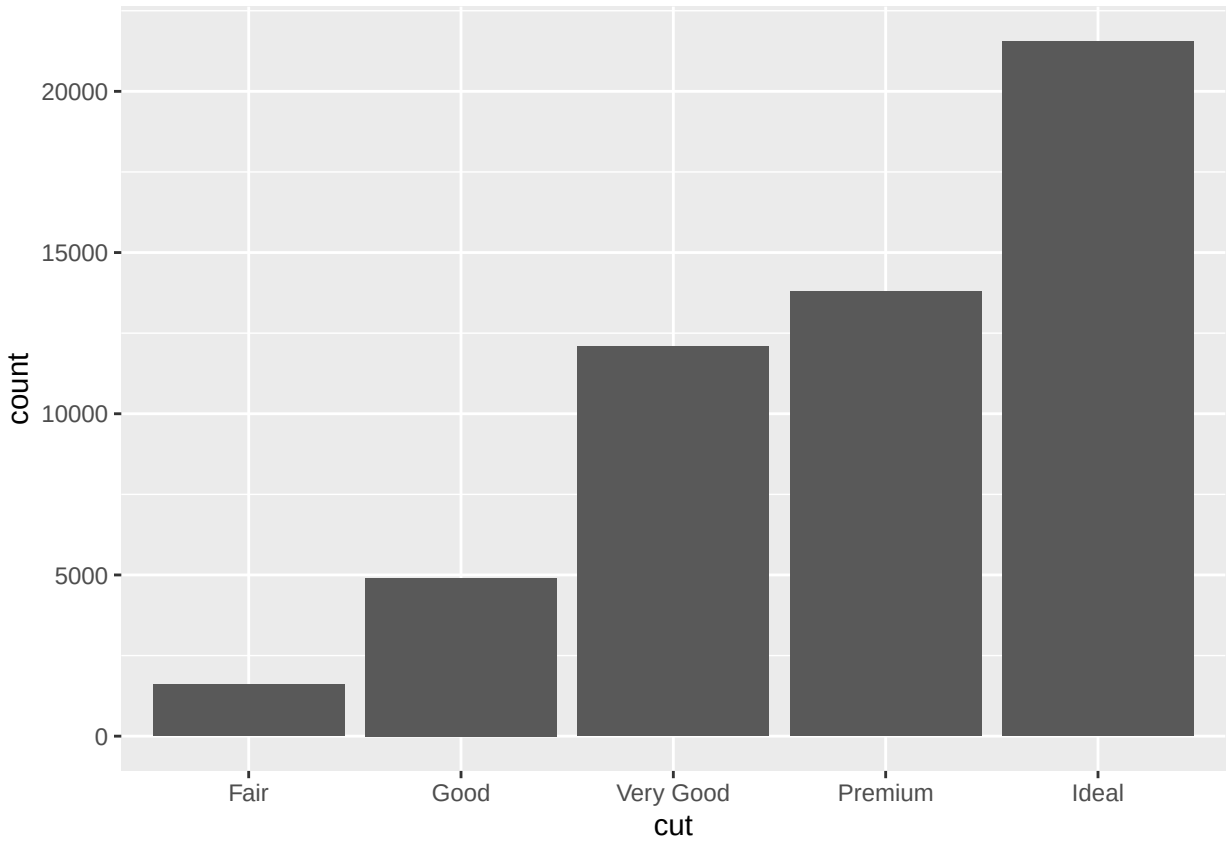
What is the default geom associated with `stat_summary()`? How could you rewrite the previous plot to use that geom function instead of the stat function?

The default geometry associated with `stat_summary` is "pointrange". Lets recreate using the geometry:

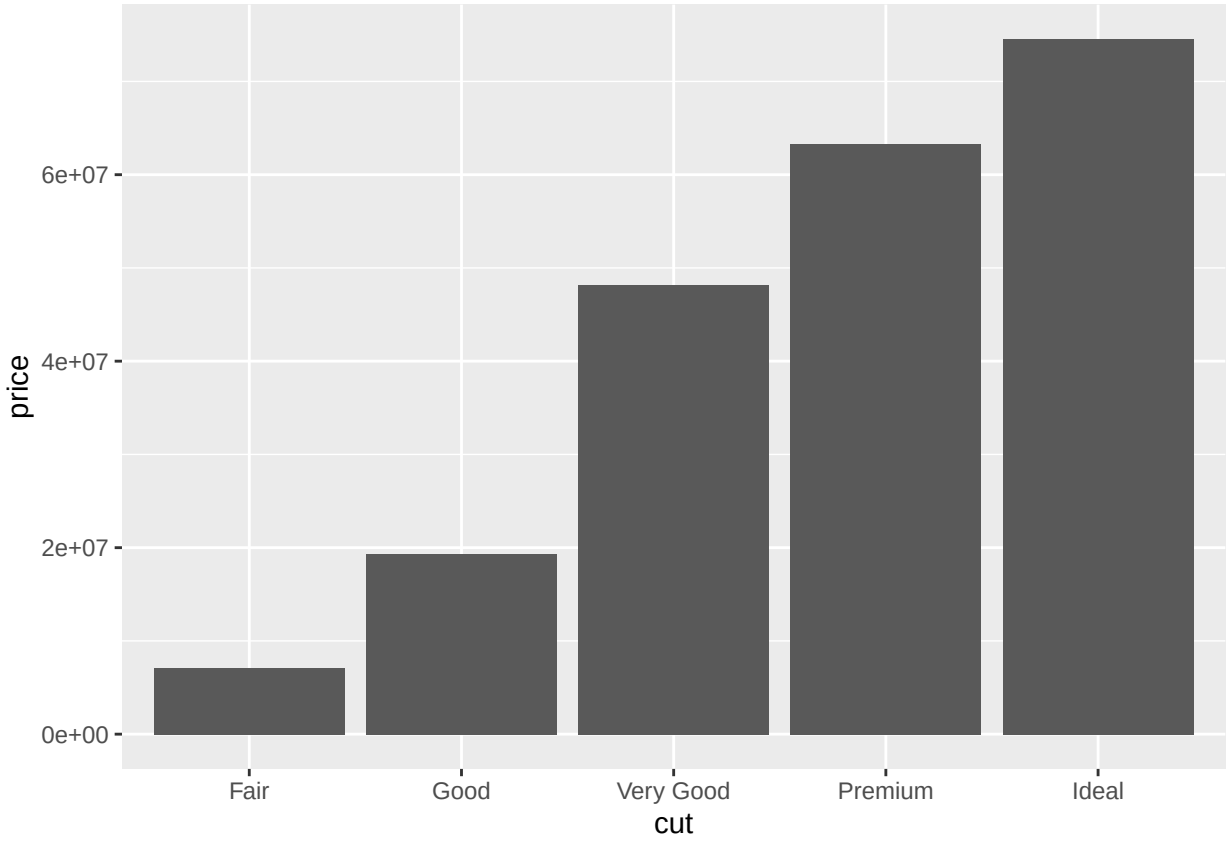
2

What does `geom_col()` do? How is it different to `geom_bar()`?

```
ggplot2::ggplot(data= diamonds, aes(x=cut))+
geom_bar()
```




```
ggplot2::ggplot(data=diamonds)+
geom_col(aes(x=cut, y=price))
```



The main difference is that `geom_col` is used to bake each bar the height proportional do the number of cases of each class in the data set. While `geom_bar` the height of each bar represents the values, not the proportion. In the documentation we can see that the `geom_bar$` already has a default statistical transformation (which is count).

3

Most geoms and stats come in pairs that are almost always used in concert. Read through the documentation and make a list of all the pairs. What do they have in common?

Geometry	Statistical Transformation
geom_bar	count
geom_col	count
geom_histogram	bin
geom_density	density
geom_boxplot	boxplot
geom_smooth	smooth

They have in common the fact that each geometry has a statistical transformations almost exactly the same as the geometry itself. The statistical transformation is used when we want to customize the default settings from the geometry.

4

What Variables does `stat_smooth()` compute? What parameters control its behavior?

```
?stat_smooth
```

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

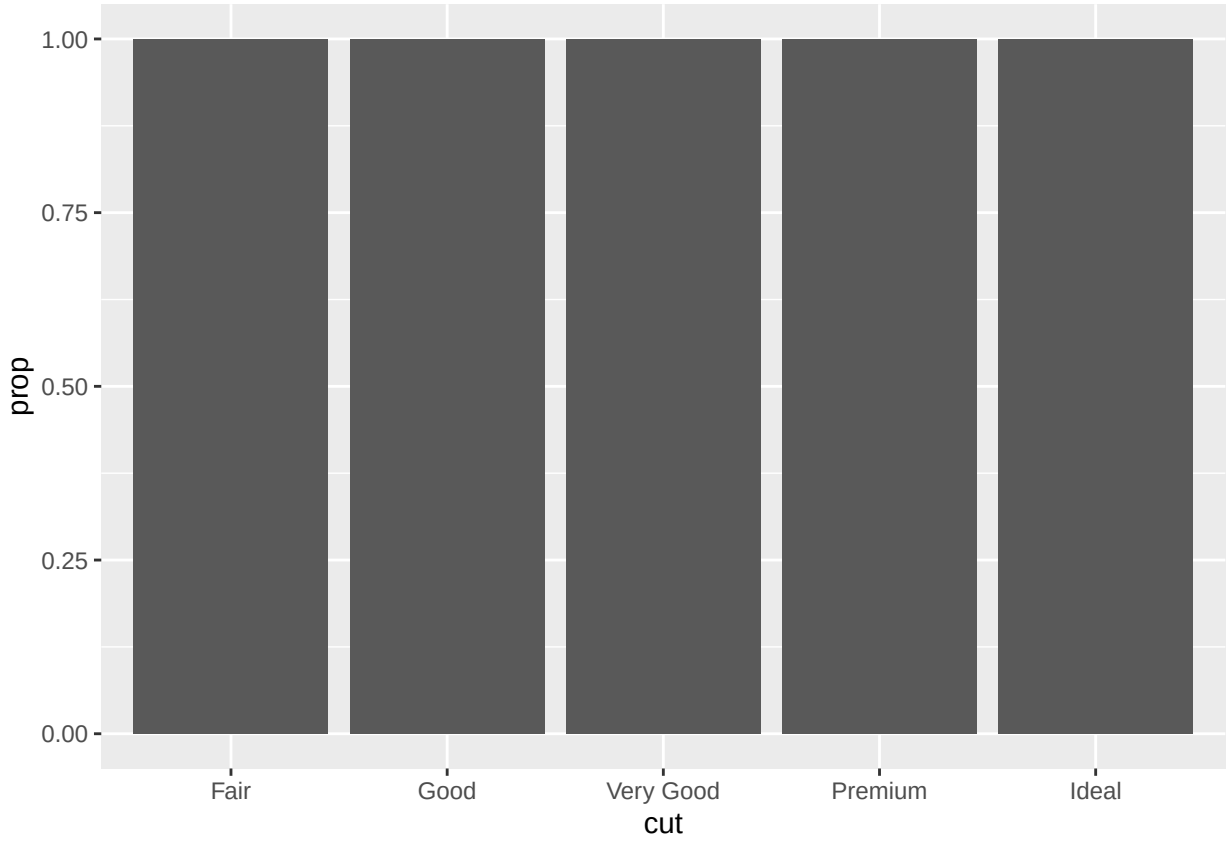
According to the documentation the `stat_smooth` computes the following variables:

- y **or** x
- ymin **or** xmin
- ymax **or** xmax
- se

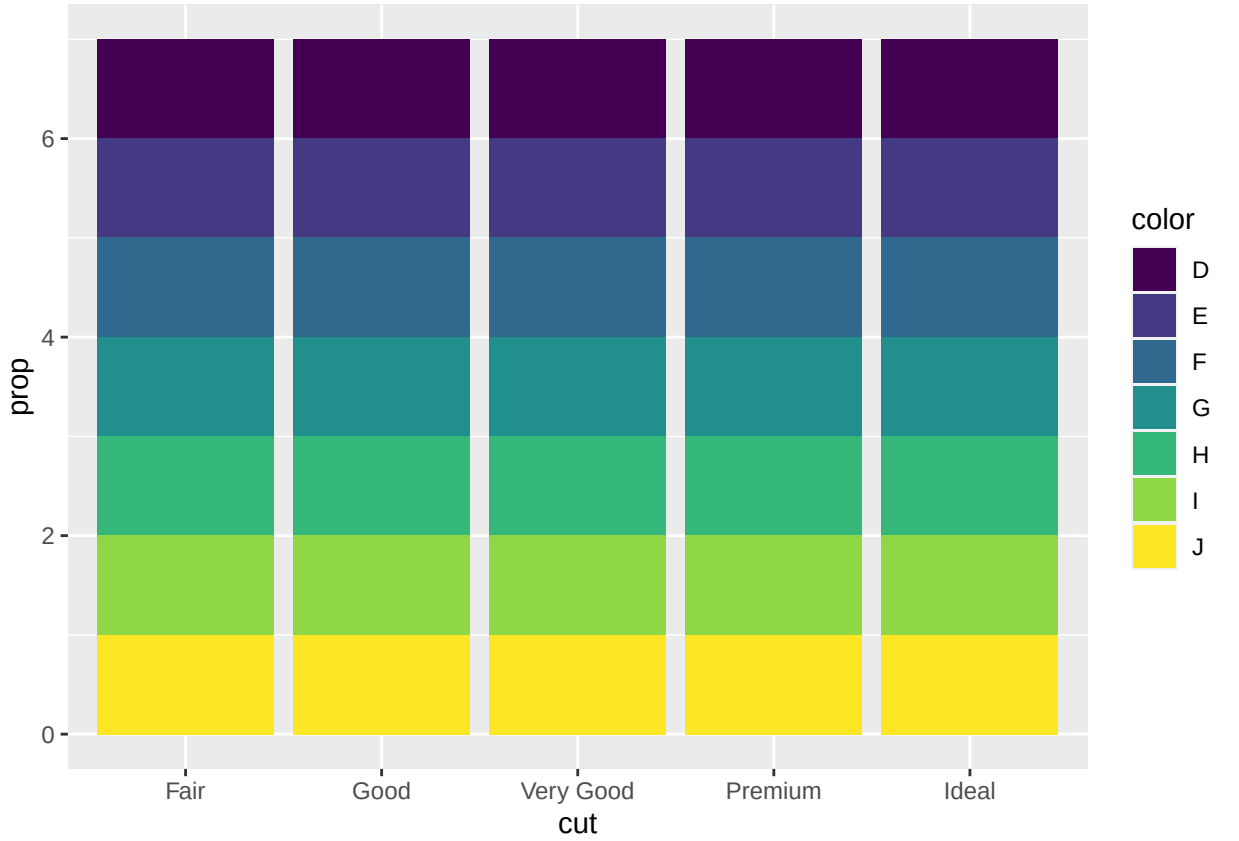
5

In our Proportion bar chart, we need to set `group = 1`. Why? In other words what is the problem with these two graphs?

```
diamonds%>%
ggplot2::ggplot() +
  geom_bar(mapping = aes(x = cut, y = after_stat(prop)))
```




```
diamonds%>%
ggplot2::ggplot() +
  geom_bar(mapping = aes(x = cut, fill = color, y = after_stat(prop)))
```



If we do not set `group=1` the statistical transformation does not divide the data into classes. What we have, then, is that each class has 100% of each type, so the proportion is always 1 in every bar.