R for Data Science - Solutions Manual

Jonathan Mendes de Almeida
jonathanalmd@gmail.com
jonathan@aluno.unb.br
@jonyddev (github)

Mar 13, 2018

Solutions to the exercises in **R for Data Science (Garrett Grolemund & Hadley Wickham)**. The individual R files are available in my github - **@jonyddev** - repository called **R4DataScience-Solutions**. *If you have any questions about my answers to these exercises do not hesitate to enter in contact with me.*

Useful links

- **RStudio Cheat Sheets**
- **ggplot2 Documentation**
- **ggplot2 Cheat Sheet**

Prerequisites

```
library(tidyverse)
```

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

```
## ✓ ggplot2 2.2.1     ✓ purrr   0.2.4
## ✓ tibble  1.4.2     ✓ dplyr   0.7.4
## ✓ tidyr   0.8.0     ✓ stringr 1.3.0
## ✓ readr   1.1.1     ✓ forcats 0.3.0
```

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

To install tidyverse library:

```
install.packages('tidyverse')
```

mpg data frame

```
mpg = ggplot2::mpg
mpg
```

```
## # A tibble: 234 x 11
##    manufacturer model    displ  year   cyl trans   drv     cty   hwy fl
##    <chr>        <chr>    <dbl> <int> <int> <chr>   <chr> <int> <int> <chr>
##  1 audi         a4        1.80  1999     4 auto(l~ f        18    29 p
##  2 audi         a4        1.80  1999     4 manual~ f        21    29 p
##  3 audi         a4        2.00  2008     4 manual~ f        20    31 p
```

```
##  4 audi         a4        2.00  2008     4 auto(a~ f        21    30 p
##  5 audi         a4        2.80  1999     6 auto(l~ f        16    26 p
##  6 audi         a4        2.80  1999     6 manual~ f        18    26 p
##  7 audi         a4        3.10  2008     6 auto(a~ f        18    27 p
##  8 audi         a4 quat~  1.80  1999     4 manual~ 4        18    26 p
##  9 audi         a4 quat~  1.80  1999     4 auto(l~ 4        16    25 p
## 10 audi         a4 quat~  2.00  2008     4 manual~ 4        20    28 p
## # ... with 224 more rows, and 1 more variable: class <chr>
```

Chapter 1

No exercises in this chapter.

Chapter 2

No exercises in this chapter.

Chapter 3

3.2.4 Exercises

Exercise 1

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

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

(Answer) An empty plot. To see some nice plots we should add some geom_function to map some points. Add a good caption for each axis is great to make your plot easier to read and understand!

Exercise 2

How many rows are in `mpg`? How many columns?

```
nrow(mpg)
```

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

```
ncol(mpg)
```

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

(Answer) 234 rows and 11 columns

Alternative method to check the number of rows and columns of a data frame:

```
glimpse(mpg)
```

```
## Observations: 234
## Variables: 11
## $ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi", "...
## $ model        <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4", "a4 qua...
## $ displ        <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0,...
## $ year         <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1...
## $ cyl          <int> 4, 4, 4, 4, 6, 6, 6, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6...
## $ trans        <chr> "auto(l5)", "manual(m5)", "manual(m6)", "auto(av)...
```

```
## $ drv          <chr> "f", "f", "f", "f", "f", "f", "f", "4", "4", "4",...
## $ cty          <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 1...
## $ hwy          <int> 29, 29, 31, 30, 26, 26, 27, 26, 25, 28, 27, 25, 2...
## $ fl           <chr> "p", "p", "p", "p", "p", "p", "p", "p", "p", "p",...
## $ class        <chr> "compact", "compact", "compact", "compact", "comp...
```

Exercise 3

What does the `drv` variable describe? Read the help for ?mpg to find out. *run ?mpg in RStudio console and check the 'help' tab)*

```
mpg['drv']
```

```
## # A tibble: 234 x 1
##    drv
##    <chr>
##  1 f
##  2 f
##  3 f
##  4 f
##  5 f
##  6 f
##  7 f
##  8 4
##  9 4
## 10 4
## # ... with 224 more rows
```

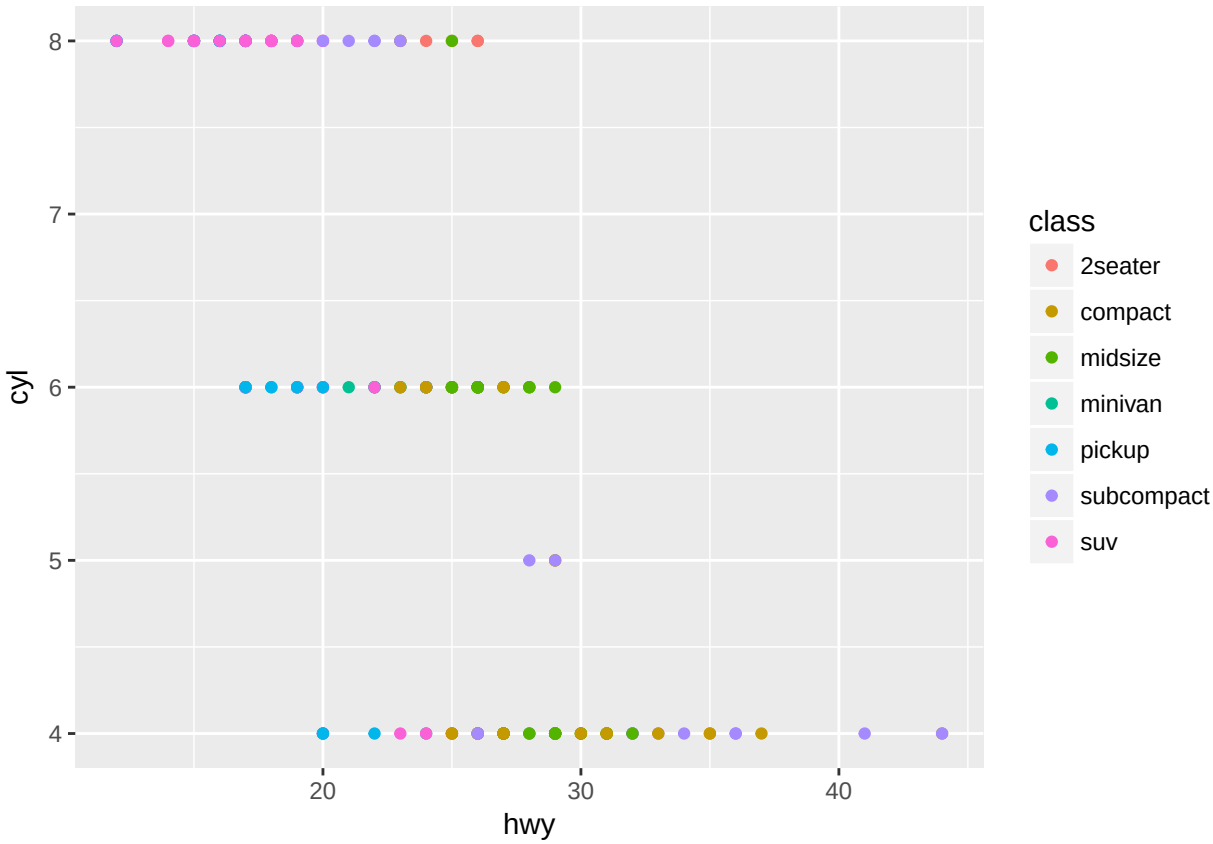
(Answer) The **drv** variable describes the traction control system. There are 3 possible values for `drv` variable (**variable** : *description*):

- **f**: *front-wheel drive*
- **r**: *rear wheel drive*
- **4**: *4wd*

Exercise 4

Make a scatterplot of `hwy` vs `cyl`

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

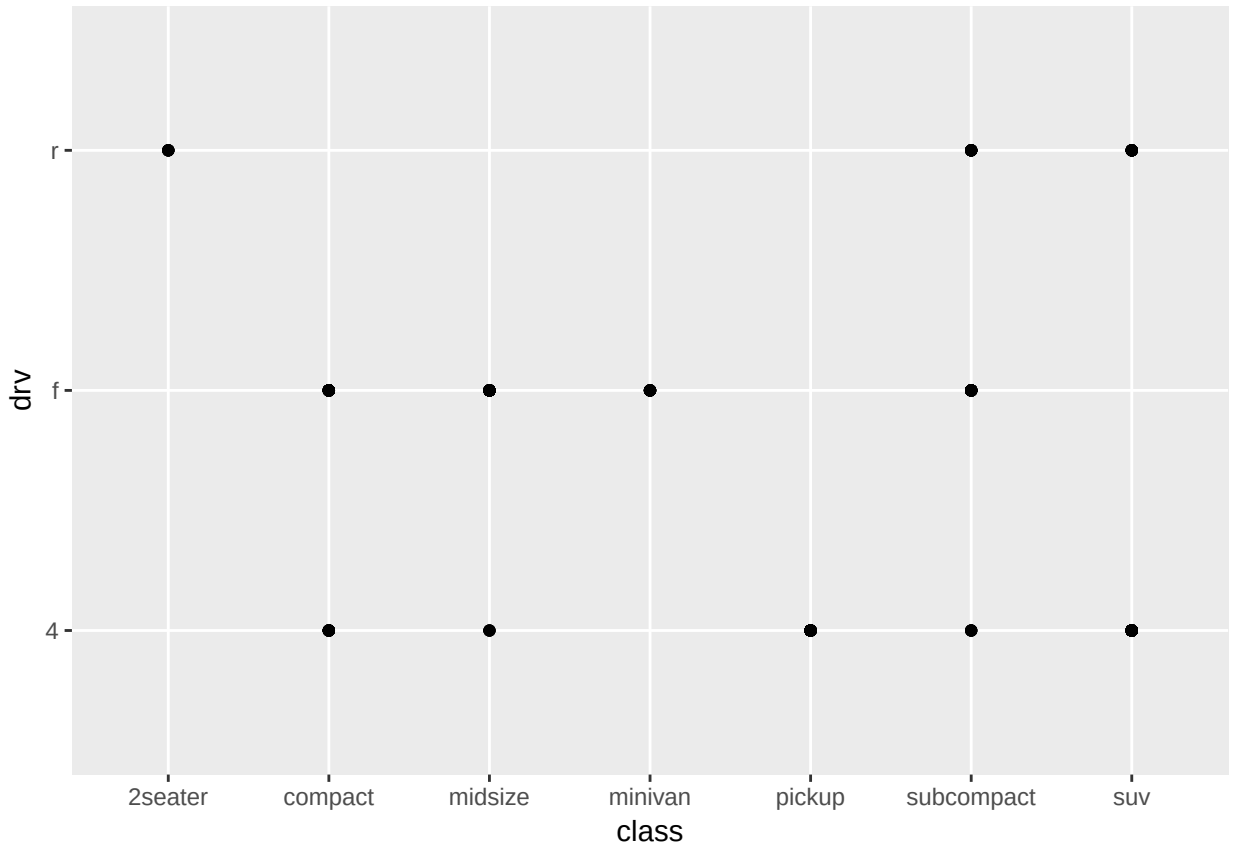


(Answer) We add a definition of the data used in x,y axis and add geom_point function mapping these points. Here we are using aes(colour = class) (to associate the name of the aesthetic with a variable to display) to plot using a different colour for each class present in our data frame (car type: 2seater, compact, midsize, minivan, pickup, subcompact, suv).

Exercise 5

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

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



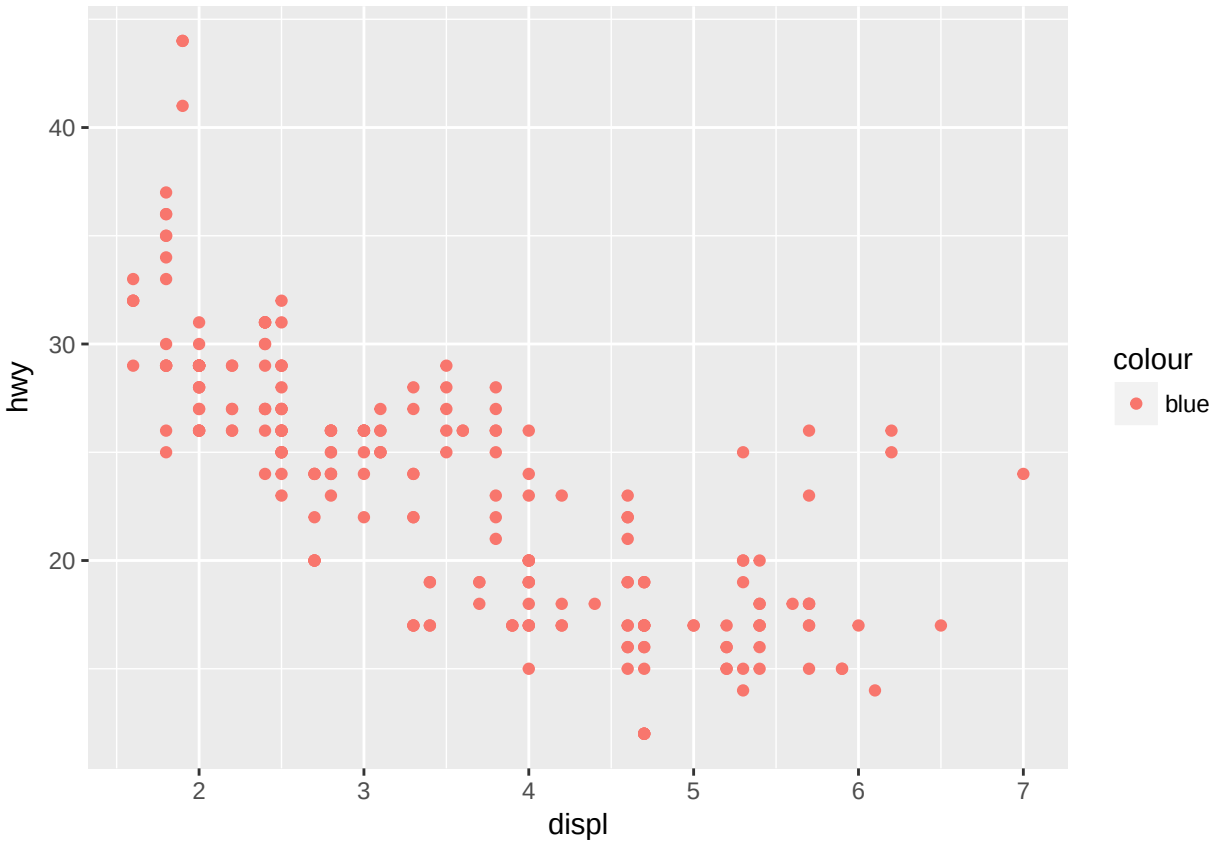
(Answer) This plot is not useful because `class` and `drv` are factor variables. Each possible value of these two variables is limited by a set (r, f and 4 are the possible values for `drv` and 2seater, compact, midsize, minivan, pickup, subcompact and suv are the possible values for `class`). This plot is pretty useless to perform a data analysis.

3.3.1 Exercises

Exercise 1

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

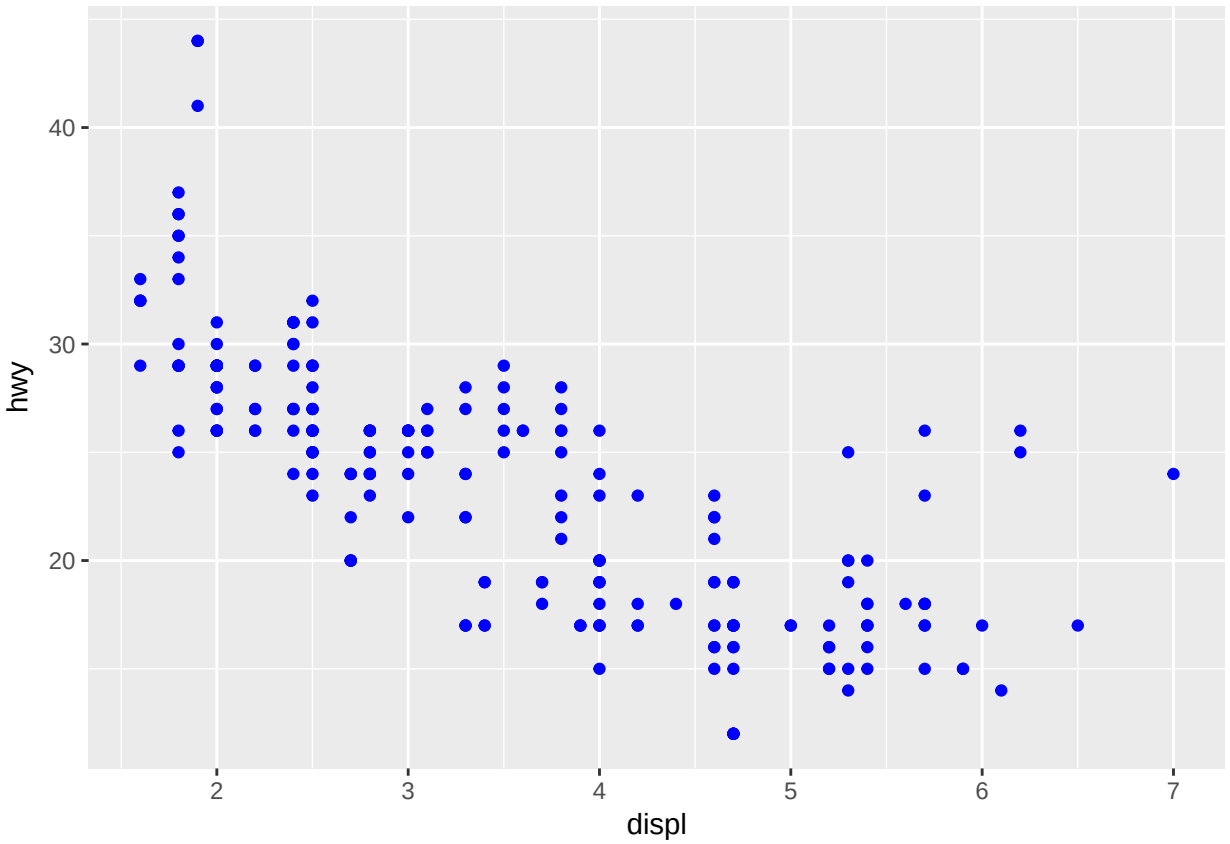
```
ggplot(data = mpg) + geom_point(mapping = aes(x = displ, y = hwy, color = "blue"))
```



(Answer) The `color` argument is not in the correct place. The `color` argument is included inside the `mapping` argument so it is treated as an aesthetic, which receives a variable (like we used `class` as argument in previous exercise). In this case, the `color` argument is interpreted as a variable with only one value (which is "blue" in this case).

If the goal is to plot all these points using blue, the correct code is:

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



Exercise 2

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

If you are not able to classify each variable as categorical or continuous by checking the description of each variable (by typing `?mpg`) you can print the data frame and R will answer this for you (another way to check this information is using the `glimpse()` function).

```
mpg
```

```
## # A tibble: 234 x 11
##    manufacturer model    displ  year   cyl trans   drv     cty   hwy fl
##    <chr>        <chr>    <dbl> <int> <int> <chr>   <chr> <int> <int> <chr>
##  1 audi         a4        1.80  1999     4 auto(l~ f        18    29 p
##  2 audi         a4        1.80  1999     4 manual~ f        21    29 p
##  3 audi         a4        2.00  2008     4 manual~ f        20    31 p
##  4 audi         a4        2.00  2008     4 auto(a~ f        21    30 p
##  5 audi         a4        2.80  1999     6 auto(l~ f        16    26 p
##  6 audi         a4        2.80  1999     6 manual~ f        18    26 p
##  7 audi         a4        3.10  2008     6 auto(a~ f        18    27 p
##  8 audi         a4 quat~  1.80  1999     4 manual~ 4        18    26 p
##  9 audi         a4 quat~  1.80  1999     4 auto(l~ 4        16    25 p
## 10 audi         a4 quat~  2.00  2008     4 manual~ 4        20    28 p
## # ... with 224 more rows, and 1 more variable: class <chr>
```

As you can see, the information is given at top of each column within '<>'. If the variable is categorical, it

will have a class of 'character' (represented as `<chr>`). So, once you know where to find this is information is easy to anwser which variable is categorical and which is continuous.

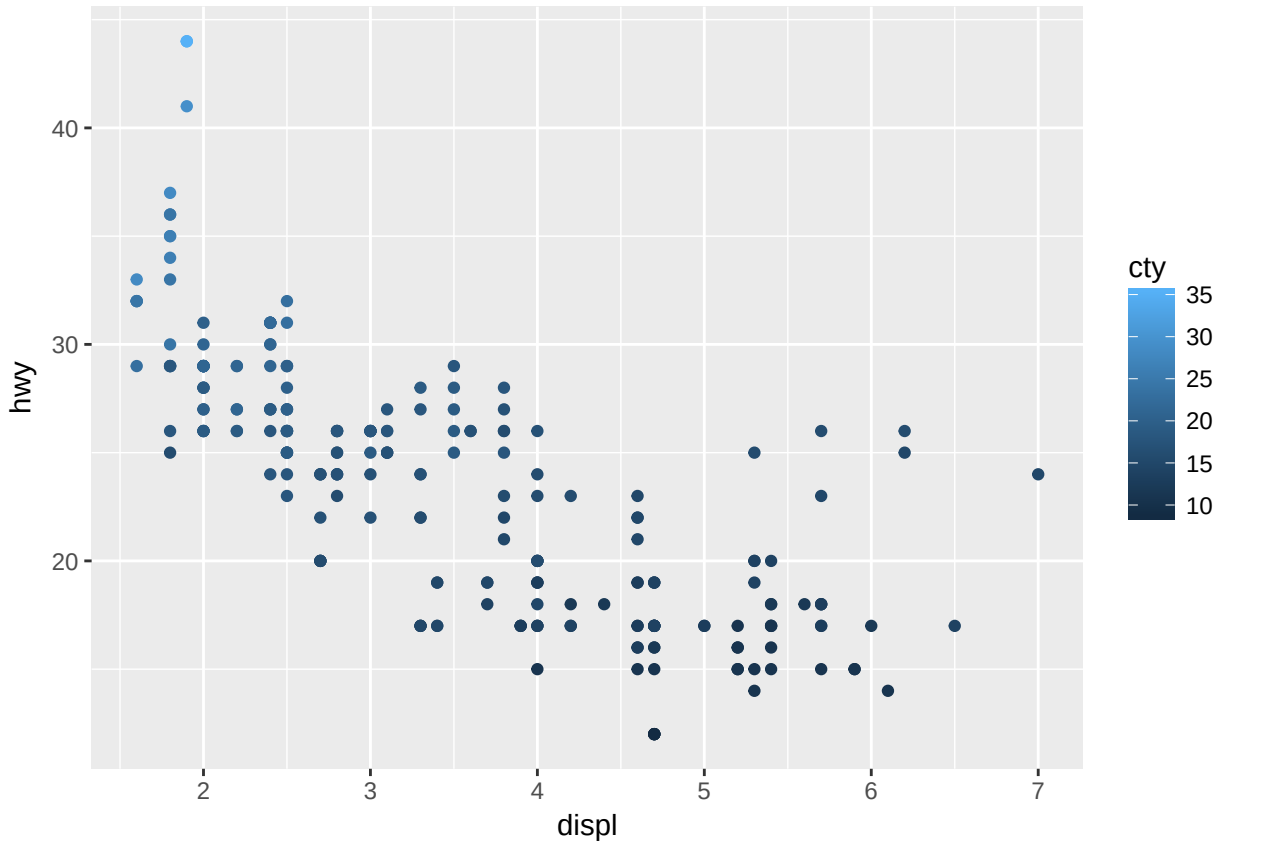
- **model**: categorical
- **displ**: continuous
- **year**: continuous
- **cyl**: continuous
- **trans**: categorical
- **drv**: categorical
- **cty**: continuous
- **hwy**: continuous
- **fl**: categorical
- **class**: categorical

Exercise 3

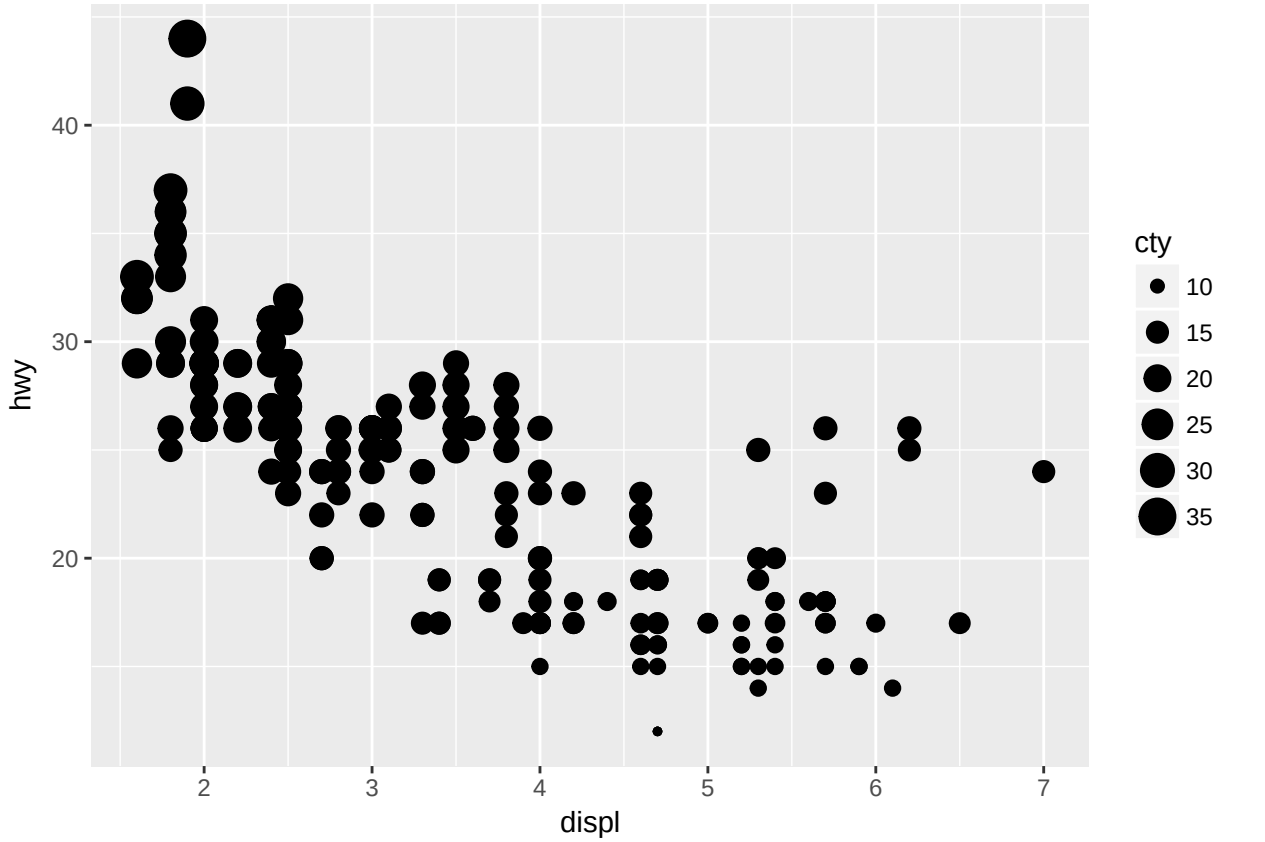
> Map a continuous variable to `color`, `size` and `shape`. How do these aesthetics behave differently for **categorical** *vs* **continuous** variables?

*Using the varible **cty** (city miles per gallon) - which is a continuous variable.*

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



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



(Answer) When mapped to colour: the continuous variable uses a scale that varies using tons of blue (light to dark).

When mapped to size: the continuous variable uses a scale that varies using different sizes.

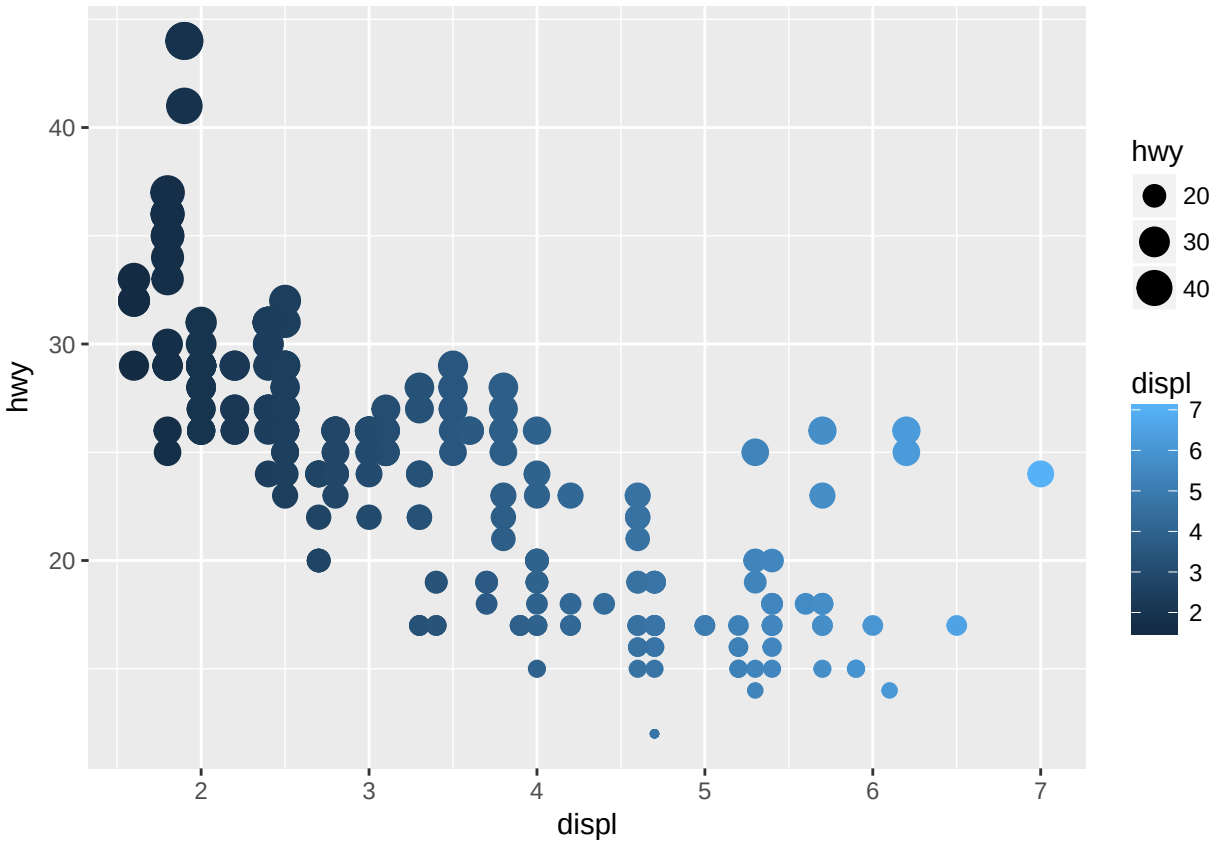
These information is easy to verify by checking these two previous plots.

However, when mapped to shape, R will give an error (a continuous variable can not be mapped to shape). This is because shapes does not have a natural order.

Exercise 4

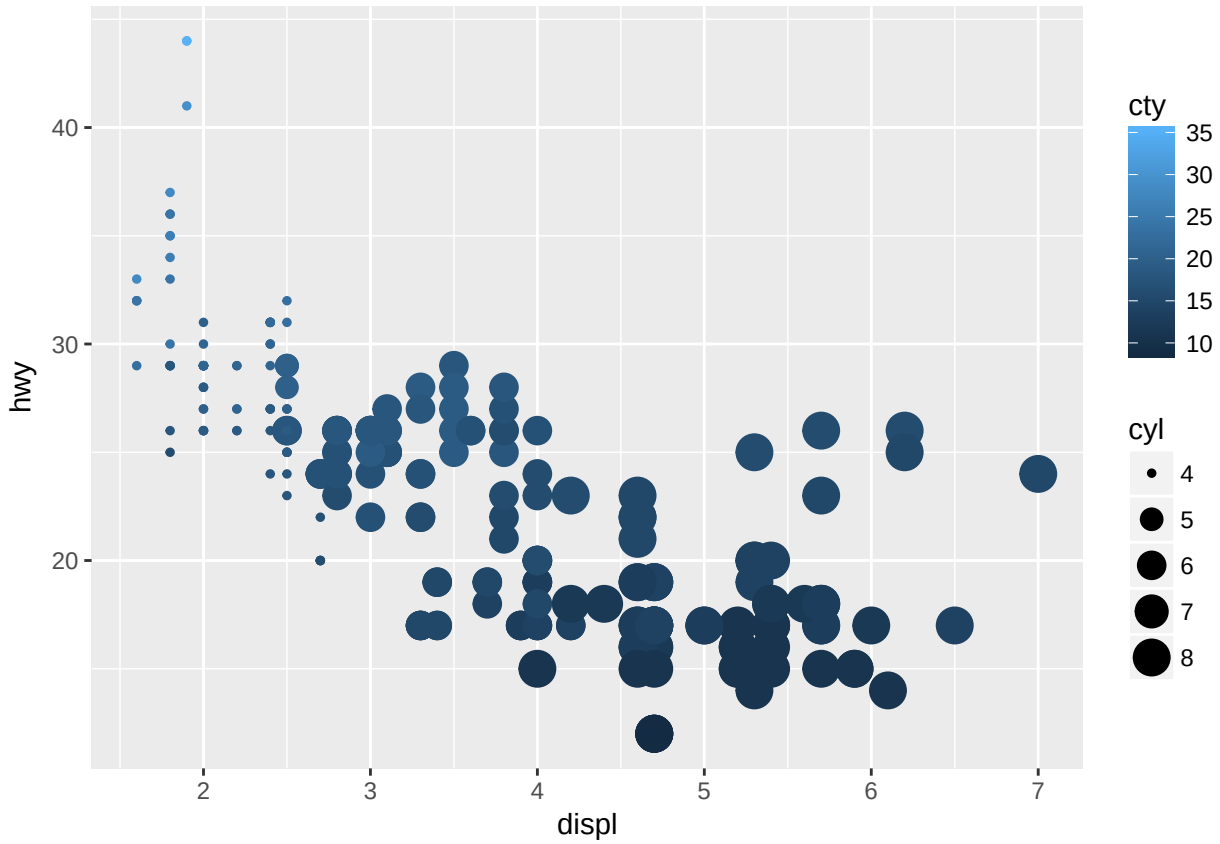
What happens if you map the same variable to multiple aesthetics?

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



Mapping one variable to multiple aesthetics is not a good idea because is redundant. Using different variables and the plot will show more information about your dataset. The next plot uses four different variables in aesthetics, which gives useful aditional information when compared to all the previous plots.

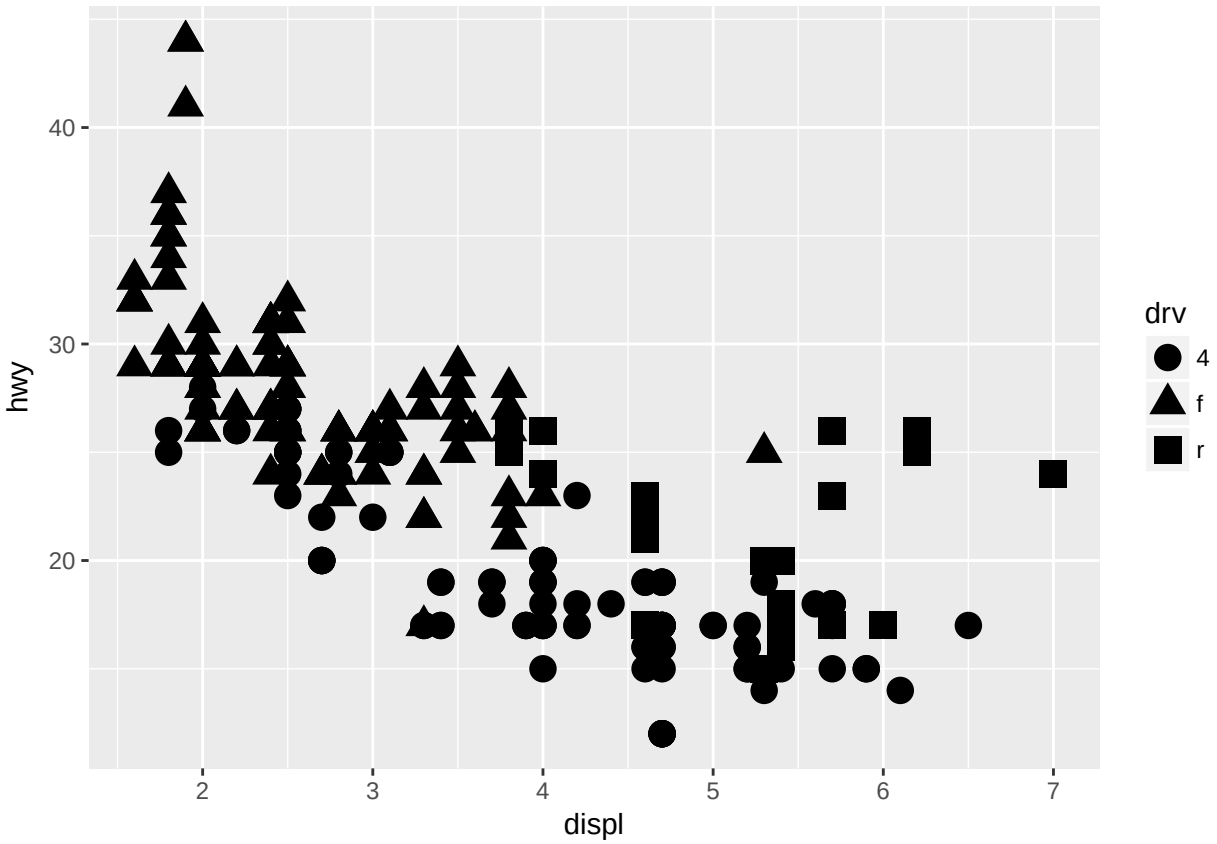
```
ggplot(data = mpg) + geom_point(mapping = aes(x = displ, y = hwy, colour = cty, size = cyl))
```



Exercise 5

What does the `stroke` aesthetic do? What shapes does it work with? (Hint: use `?geom_point` and check the 'help' tab)

```
ggplot(data = mpg) + geom_point(mapping = aes(x = displ, y = hwy, shape = drv), stroke = 5)
```



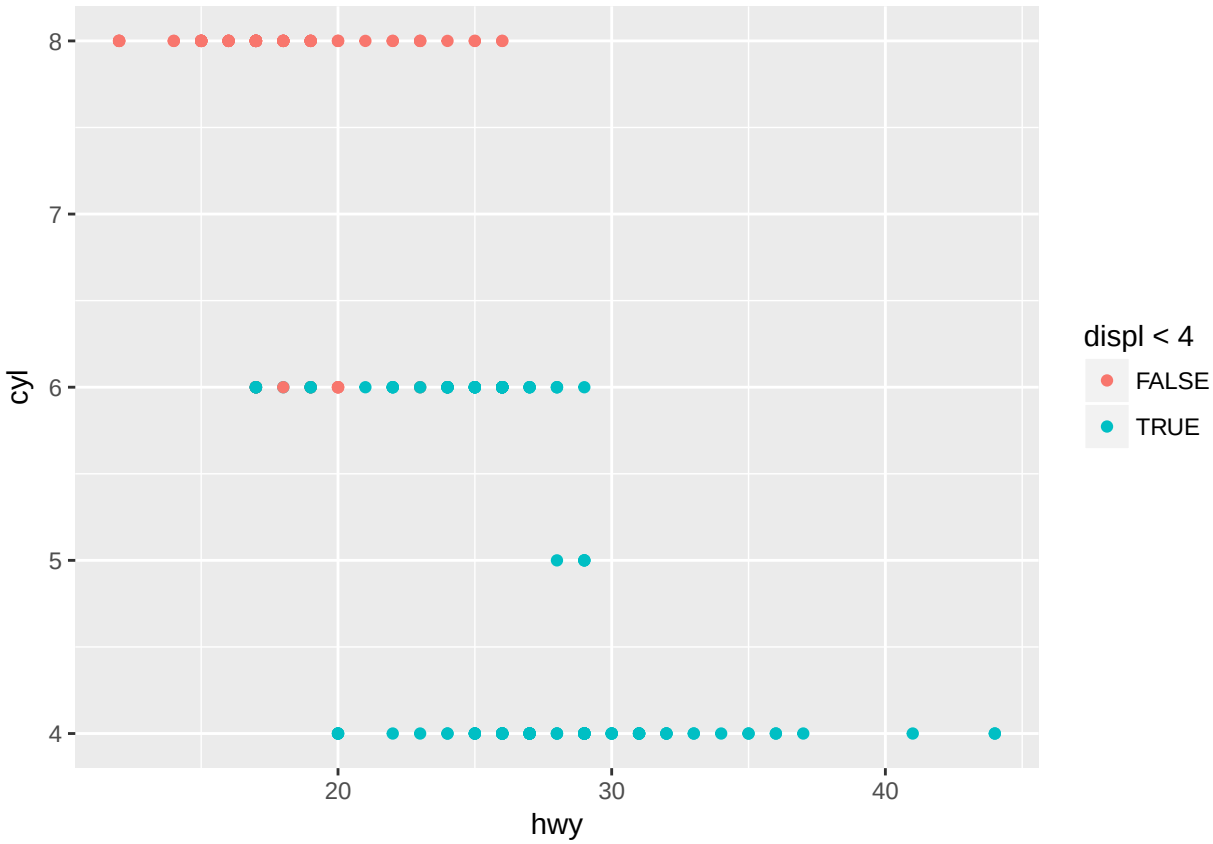
(Answer) The `stroke` aesthetic is used to modify the width of the border.

Exercise 6

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

(Answer) The colour indicates if each `displ` value is less than 4 or not. The ggpplot function will assign the result of this expression (displ < 5, which is going to be true or false) to a temporary variable and then will assign a colour for values > 5 and a different colour for values < 5). This is easy visualize by checking the results of this code:

```
ggplot(data = mpg) + geom_point(mapping = aes(x = hwy, y = cyl, colour = displ < 4))
```



3.5.1 Exercises

Exercise 1

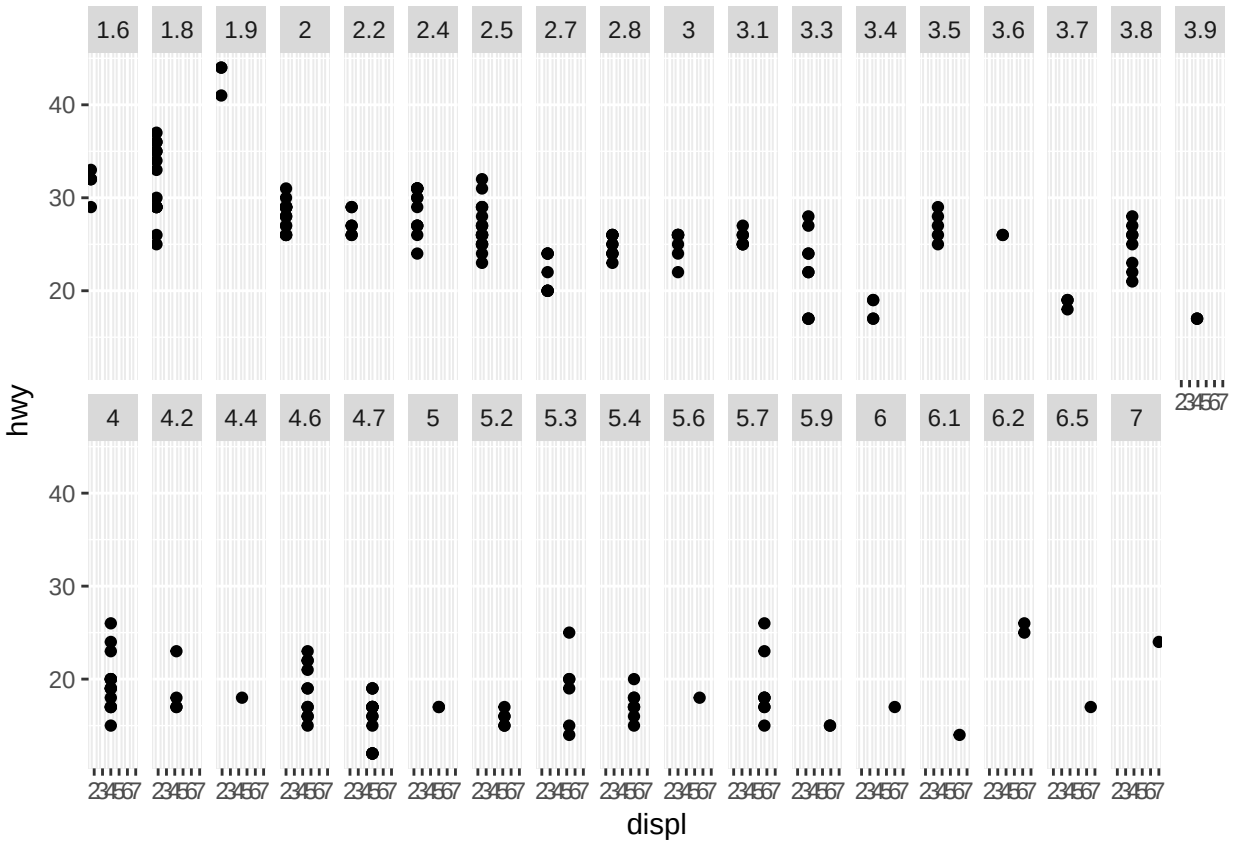
What happens if you facet on a continuous variable?

(Answer) To remember the variables classification:

Continuous	Categorical
displ	model
year	trans
cyl	drv
cty	fl
hwy	class

Let's plot and see what happens!

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

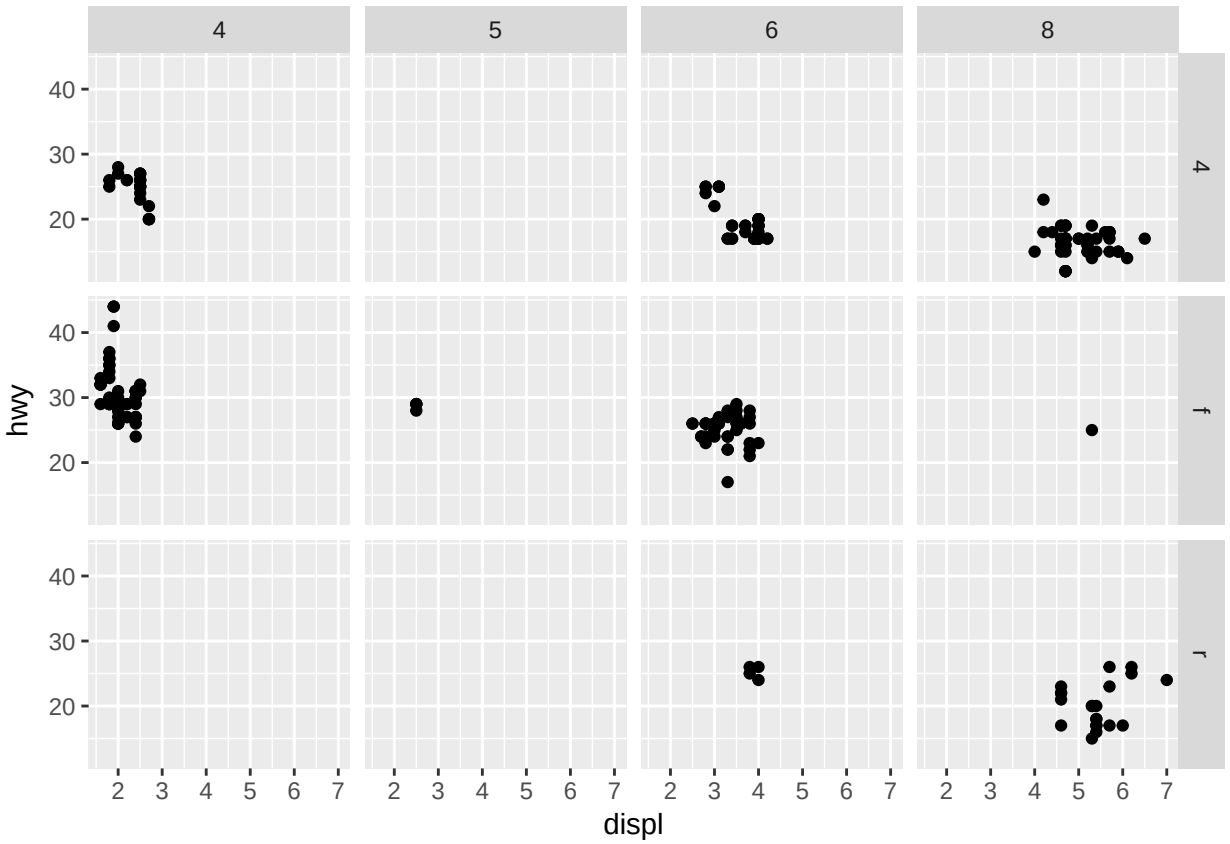


As you can see, it converts the continuous variable to a factor and then creates facets for all unique values of it. Facets is particularly useful for **categorical** variables.

Exercise 2

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

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



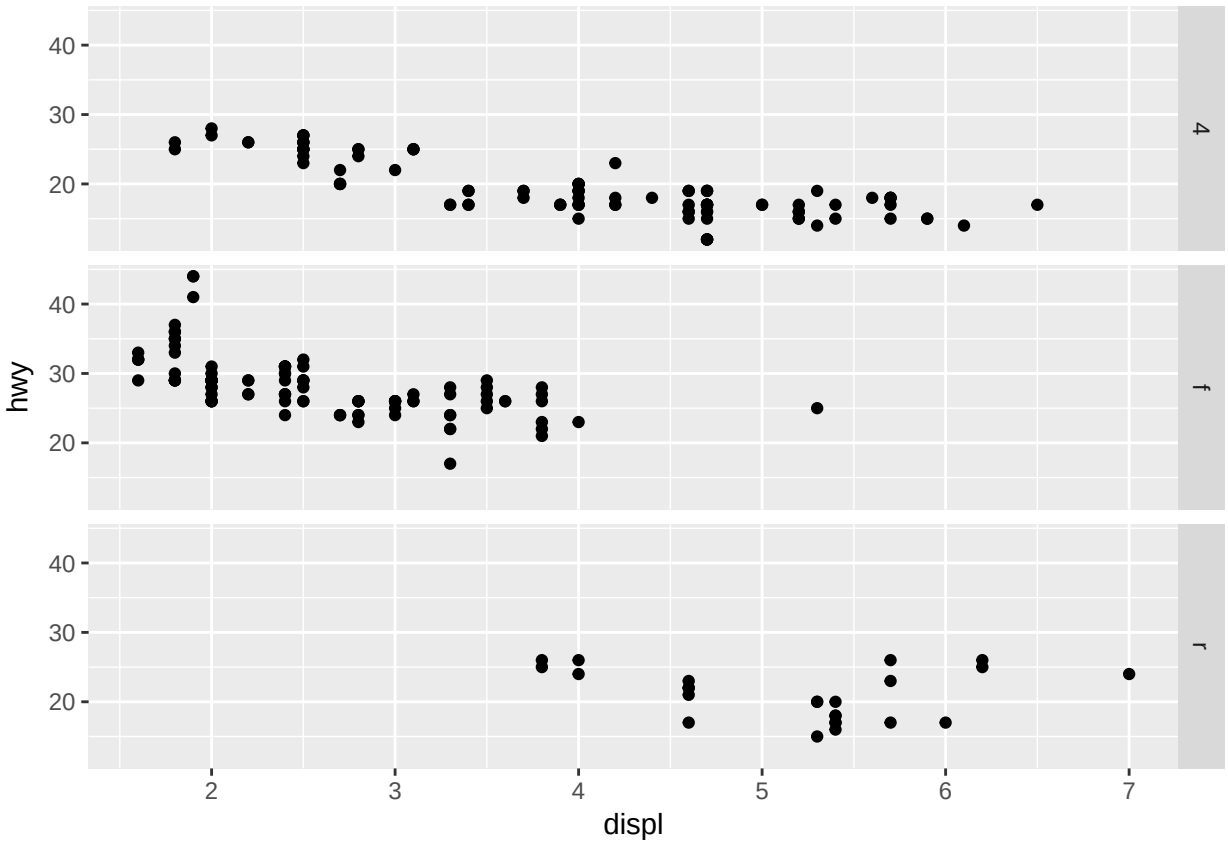
(Answer) The empty cells means that there are no values for the combination of `drv` and `cyl`. In this case, there are no cars which the traction control system is 4wd and the number of cylinders is 5, for example (you can check the same for the two others empty cells).

Exercise 3

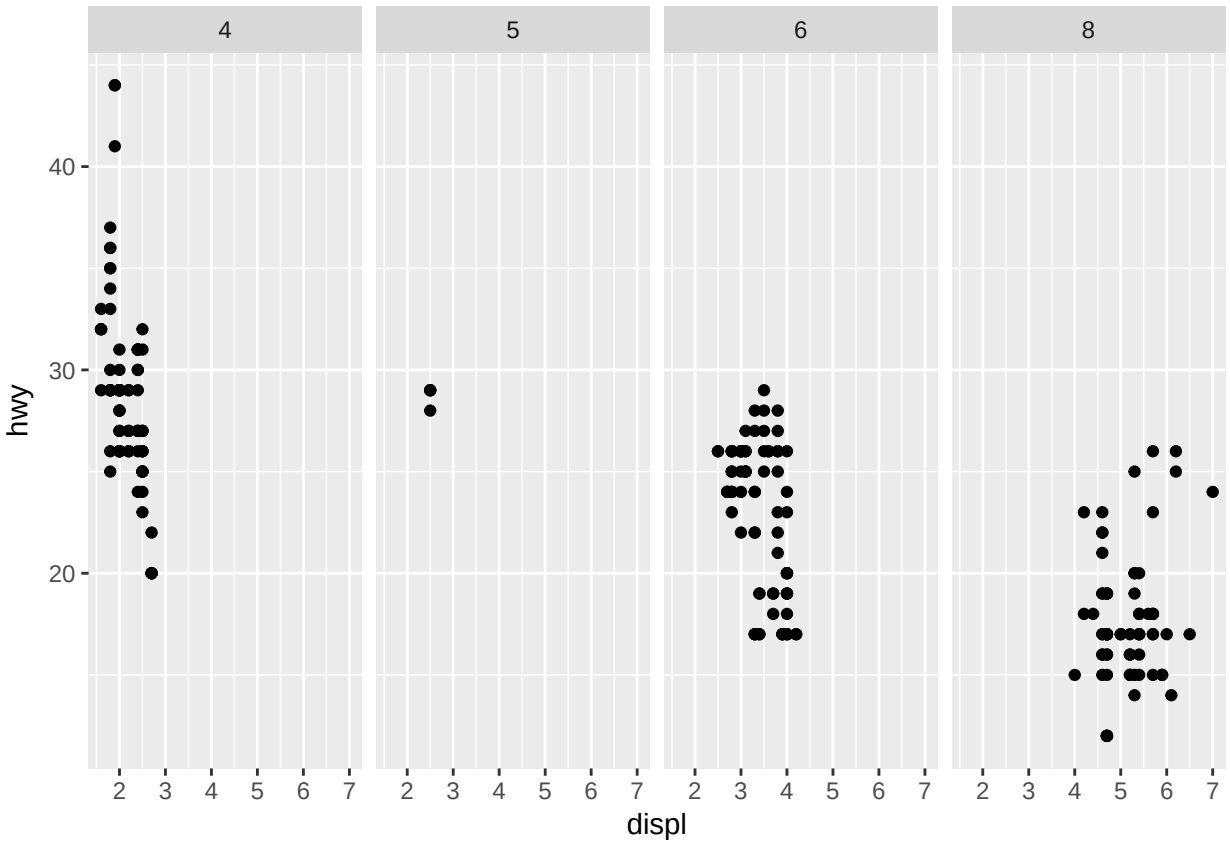
What plots does the following code make? What does `.` do?

Let's see!

```
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)
```

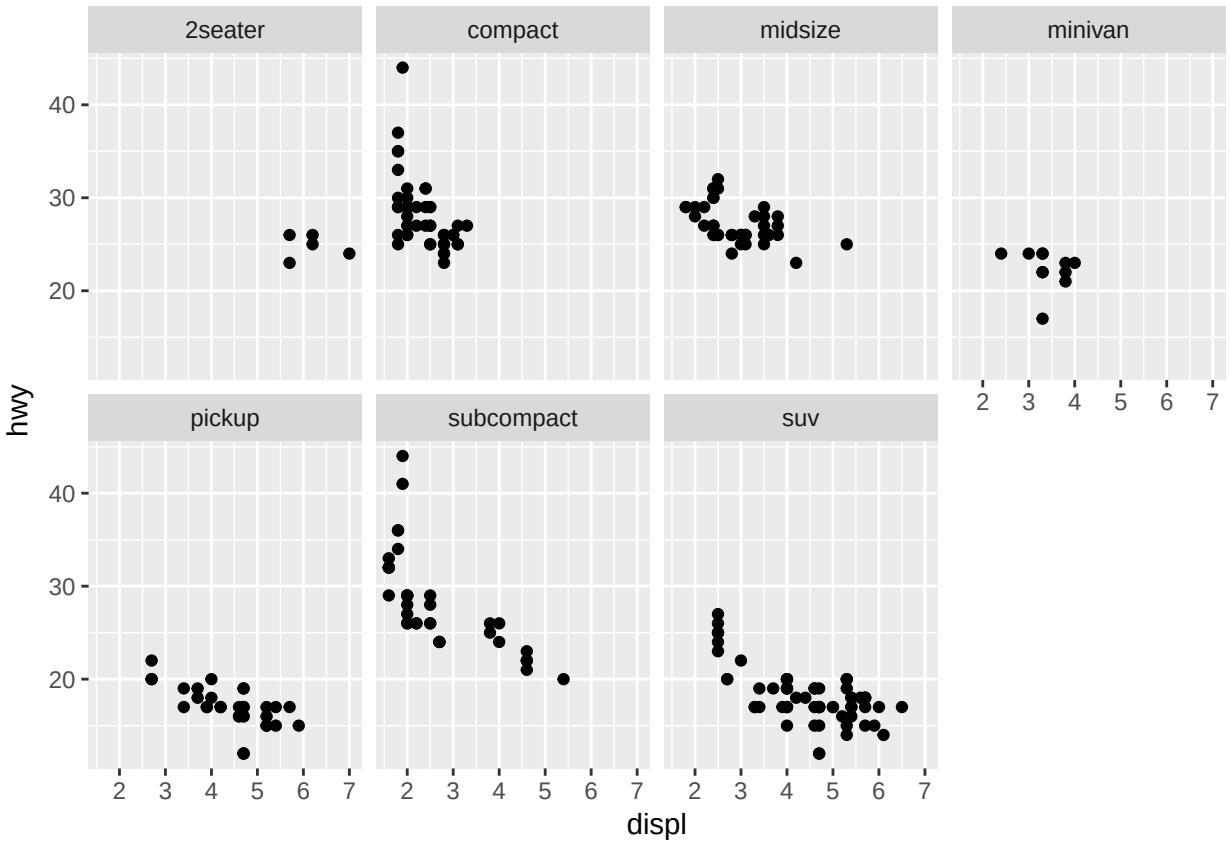


(Answer) As you can see in these two plots, `.` ignores a dimension for faceting (x or y axis).

Exercise 4

Take the first faceted plot in this section:

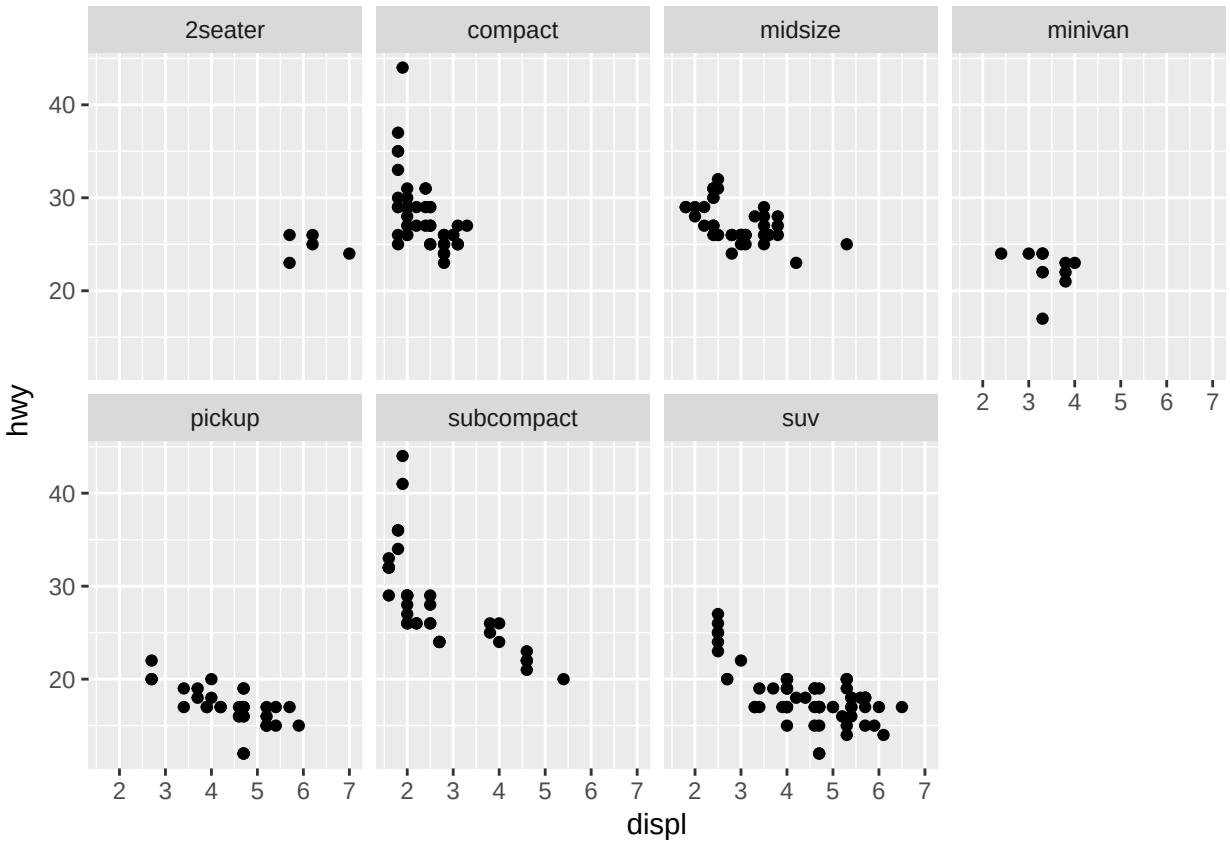
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_wrap(~ class, nrow = 2)
```



What are the advantages to using faceting instead of the colour aesthetic? What are the disadvantages? How might the balance change if you had a larger dataset?

Let's run this code (again):

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



(Answer) Using faceting could be a great option to visualize you data if the number of classes (categorical variable) is not so large because faceting permits to visualize each category separated and maybe this can show the information about your dataset better than when using colour aesthetic (if you want to see the results for each category or a set of categories, for example). However, for larger datasets we might face with a categorical variable with many possible results and for this situation is better to visualize the data using colour aesthetic. The function you use depends on your dataset.

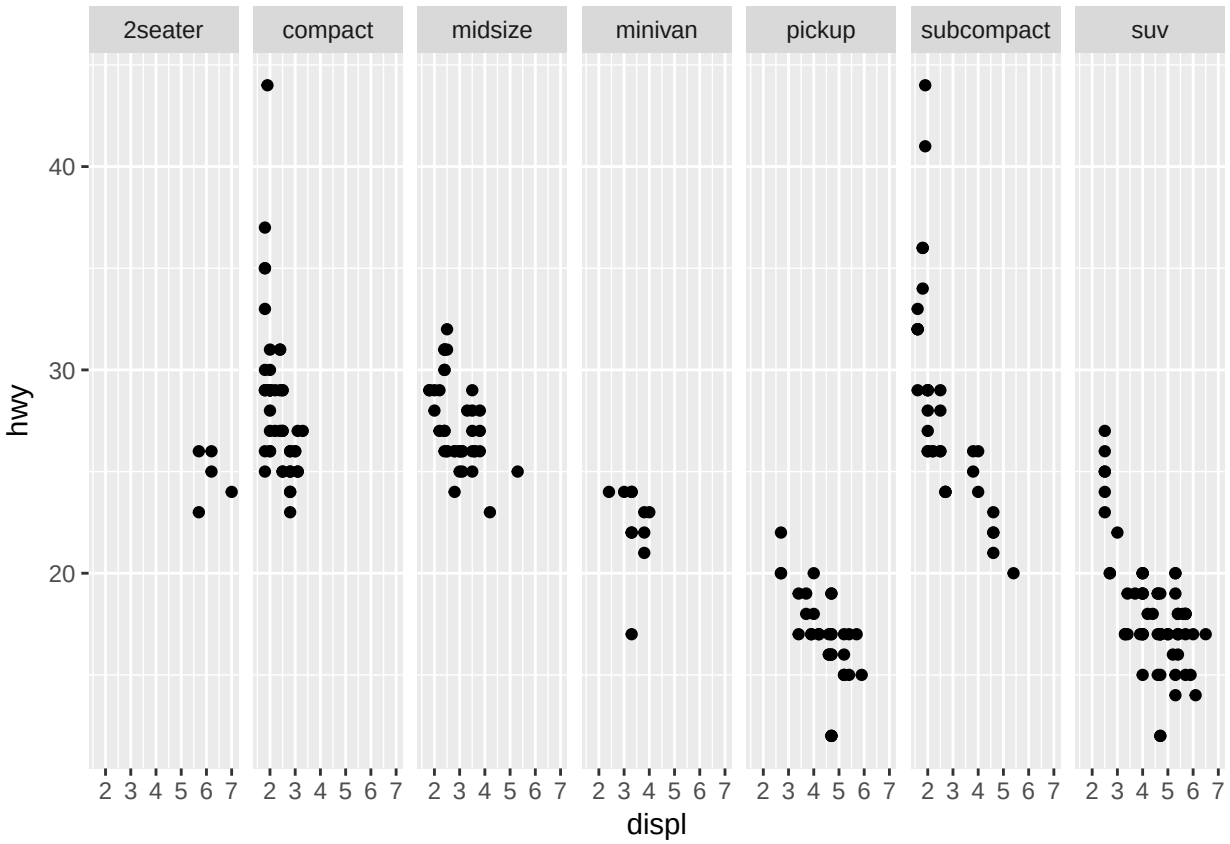
Exercise 5

> Read `?facet_wrap`. What does `nrow` do? What does `ncol` do? What other options control the layout of the individual panels? Why doesn't `facet_grid()` have `nrow` and `ncol` argument?

(Answer) `nrow` and `ncol` define the number of rows and columns and this is necessary since facet_wrap only facets on one variable. You also can change the layout of the individual panels with `scales`, `switch`, `as.table` or `dir`, for example.

Let's see what happens when we set `ncol = 7`, which is the number of different car classes (`class`).

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_wrap(~ class, ncol = 7)
```



Really cool, right?

On the other hand, `nrow` and `ncol` are not necessary in `face_grid()` because the number of rows and columns are already determined depending only on the variables that were chosen (number of unique values of the variables used).

Exercise 6

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

(Answer) This is usually used in this way just to be easier to visualize. Is better to see the plot larger horizontally than vertically. So, using the variable with more unique levels in the columns the plot will grow horizontally. On the other hand, if this variable is used in the rows, the plot will grow vertically and for humans, usually this is worse to visualize.

3.6.1 Exercises

Exercise 1

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

(Answer) Plot type - Geom you should use:

Plot type	Geom
Line chart	`geom_line()`

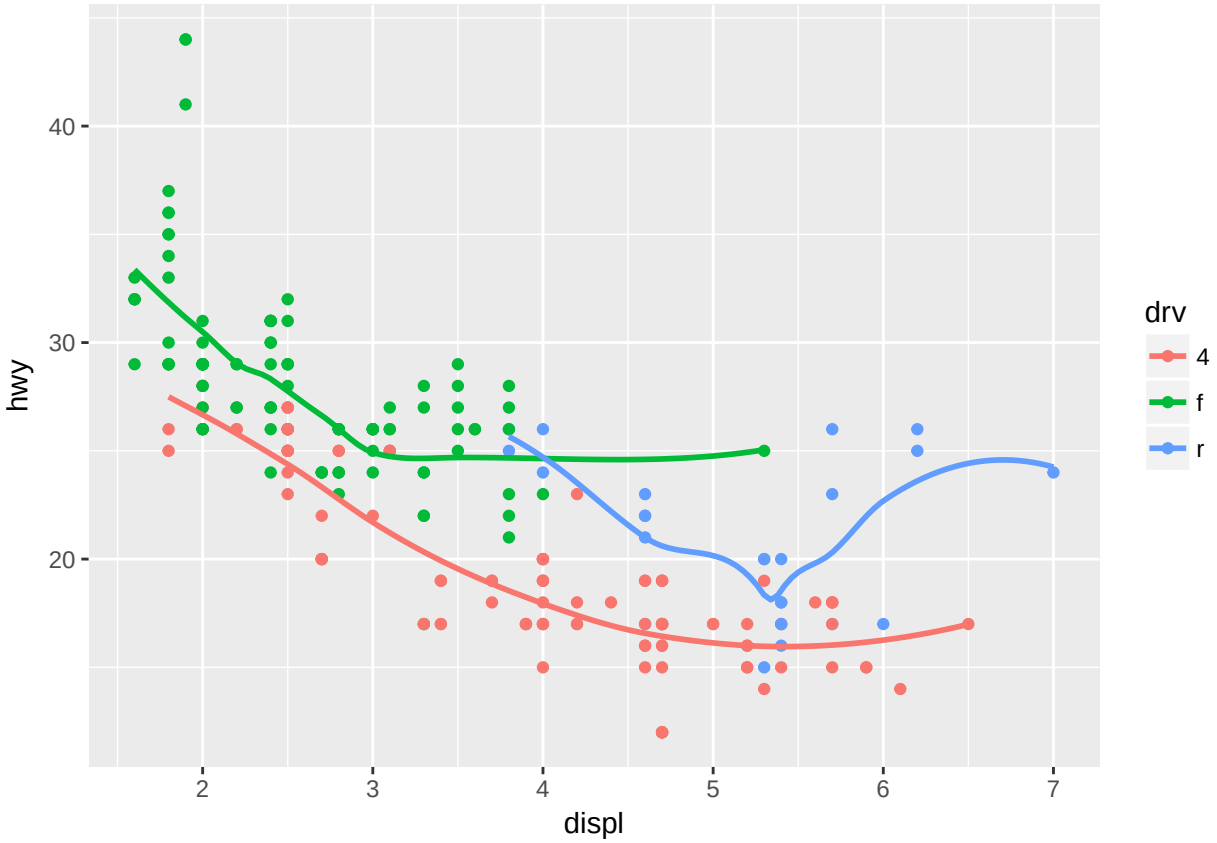
Plot type	Geom
Boxplot	`geom_boxplot()`
Histogram	`geom_hist()`
Area chart	`geom_area()`

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.

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

```
## `geom_smooth()` using method = 'loess'
```



(Answer) As you can see in the previous plot, this code produces a scatter plot with `displ` on the x axis and `hwy` on the y axis and the points are coloured according to the `drv` variable. Also, there is a smooth line created with `geom_smooth` with the standard errors setted to false (`se = FALSE`) and fitted according to `drv`.

Exercise 3

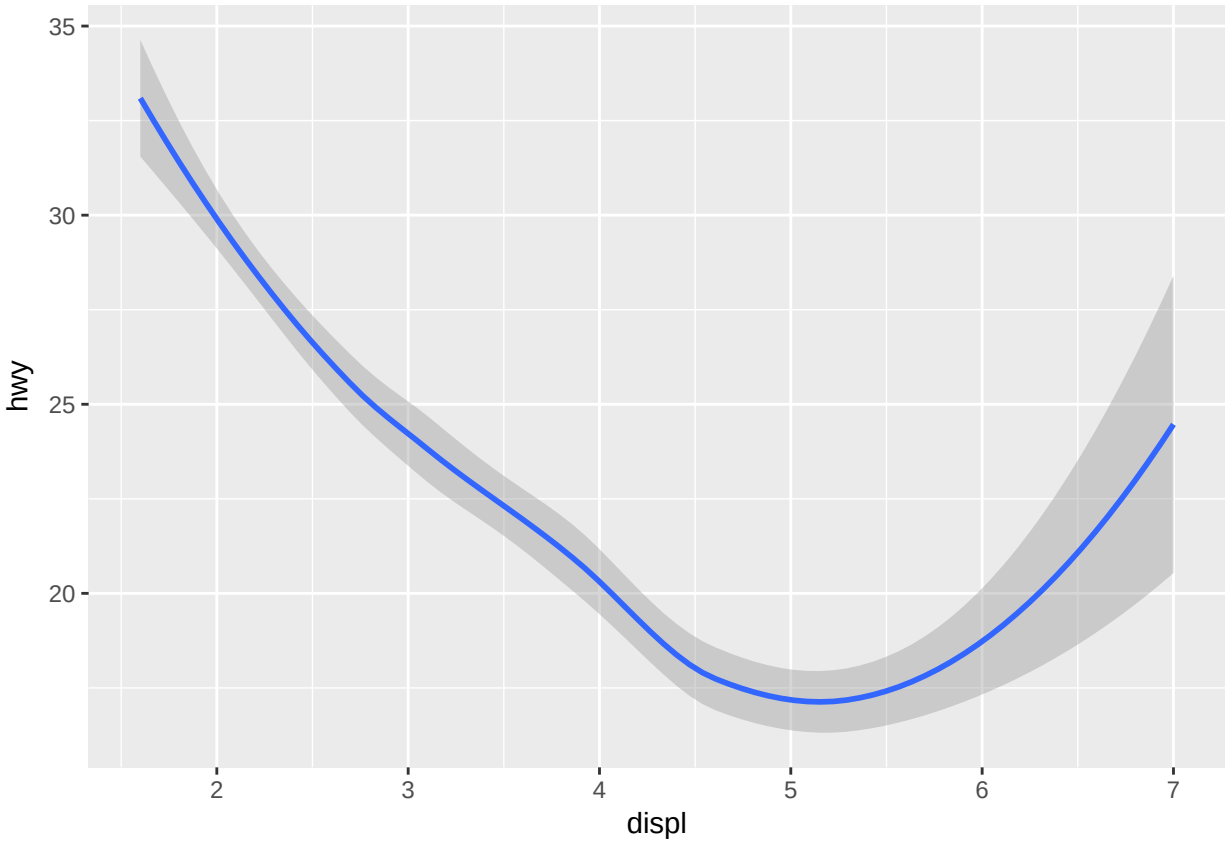
What does show.legend = FALSE do? What happens if you remove it? Why do you think I used it earlier in the chapter?

I am no sure if my answer is one hundred percent correct for the last question of this exercise.

(Answer) `show.legend = FALSE` hides the legend for the plot. If you do not specify this, the default value is going to be true (plot will show the legend box, if there is more than one category). The book used it earlier in this chapter to create these 3 plots:

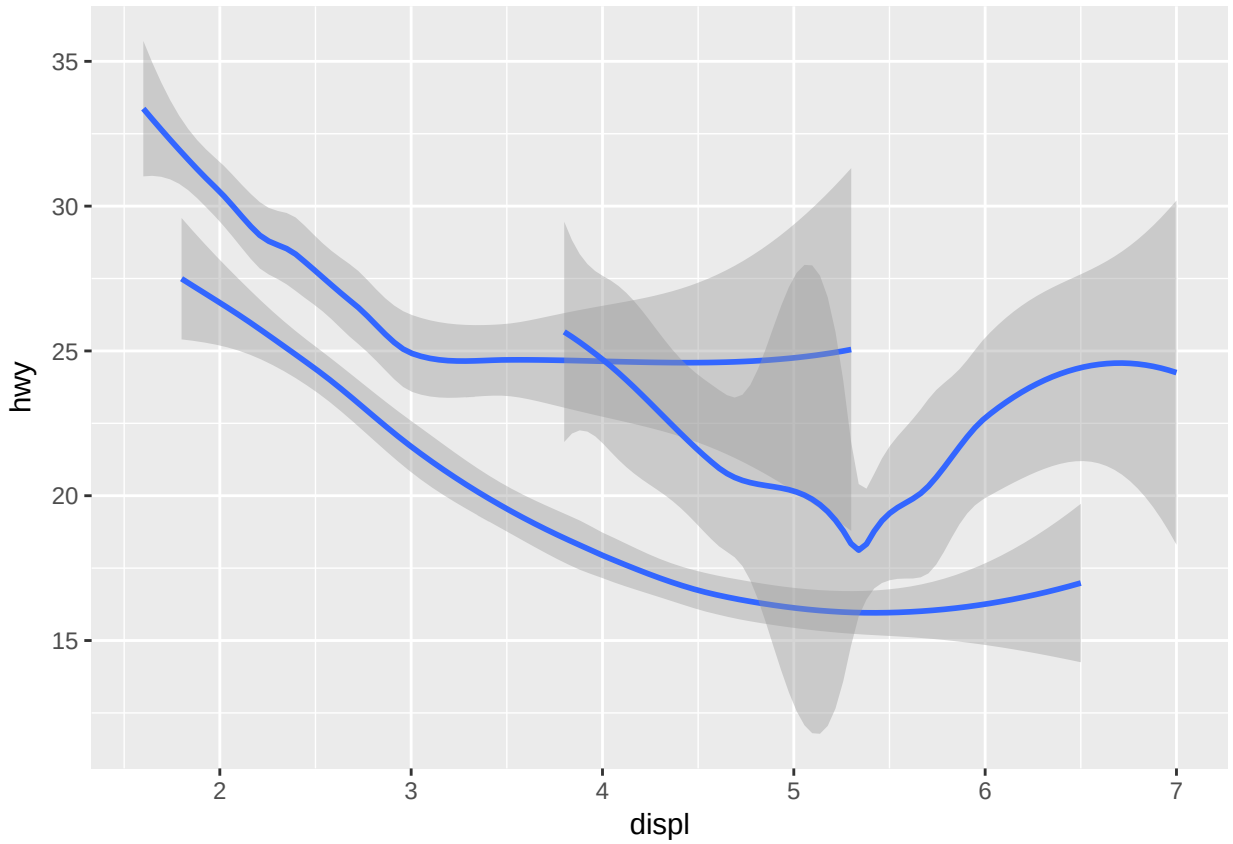
```
ggplot(data = mpg) +
  geom_smooth(mapping = aes(x = displ, y = hwy))
```

```
## `geom_smooth()` using method = 'loess'
```



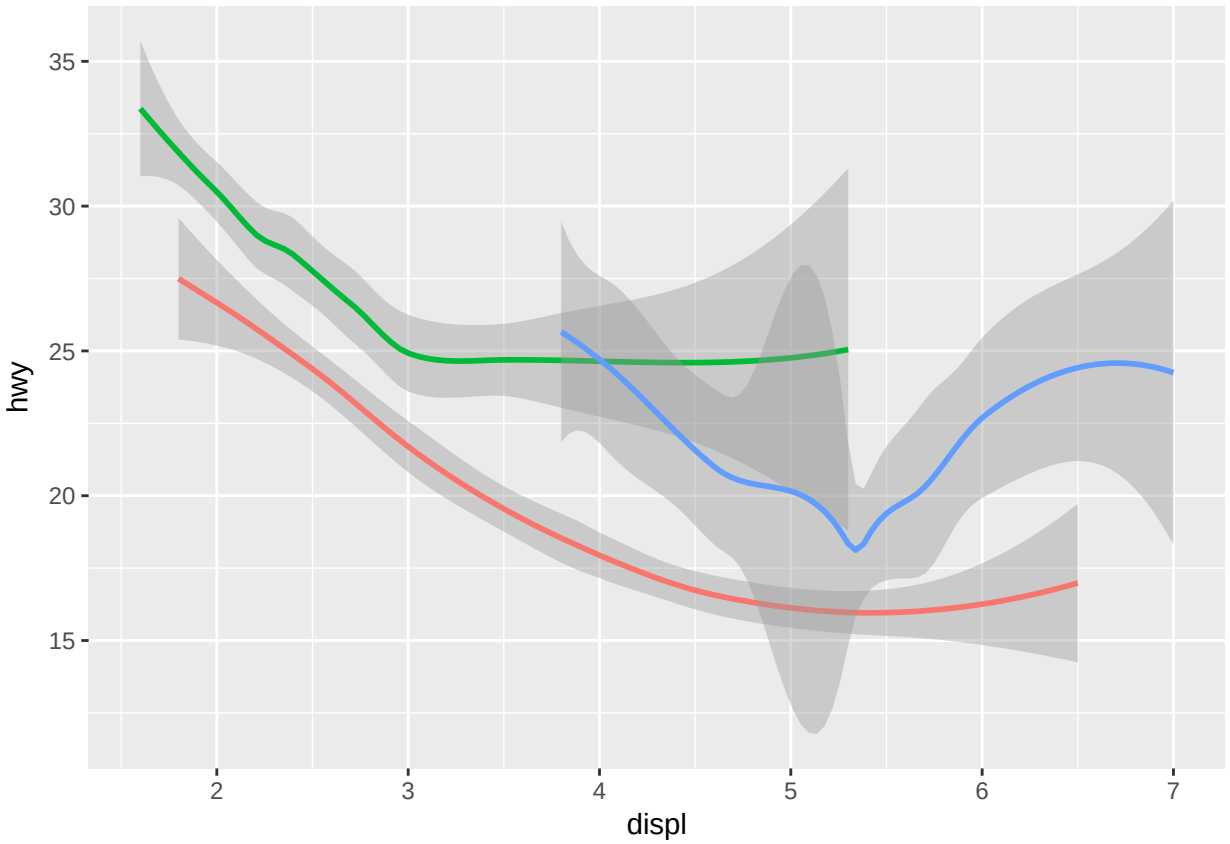

```
ggplot(data = mpg) +
  geom_smooth(mapping = aes(x = displ, y = hwy, group = drv))
```

```
## `geom_smooth()` using method = 'loess'
```




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

```
## `geom_smooth()` using method = 'loess'
```



In this case, a legend just in the last plot is not a good idea because in the two first plots there is no legend for the plot. The legend would make a irregular presentation and would show a irrelevant information (out of the scope of the goal that these 3 plots have).

Exercise 4

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

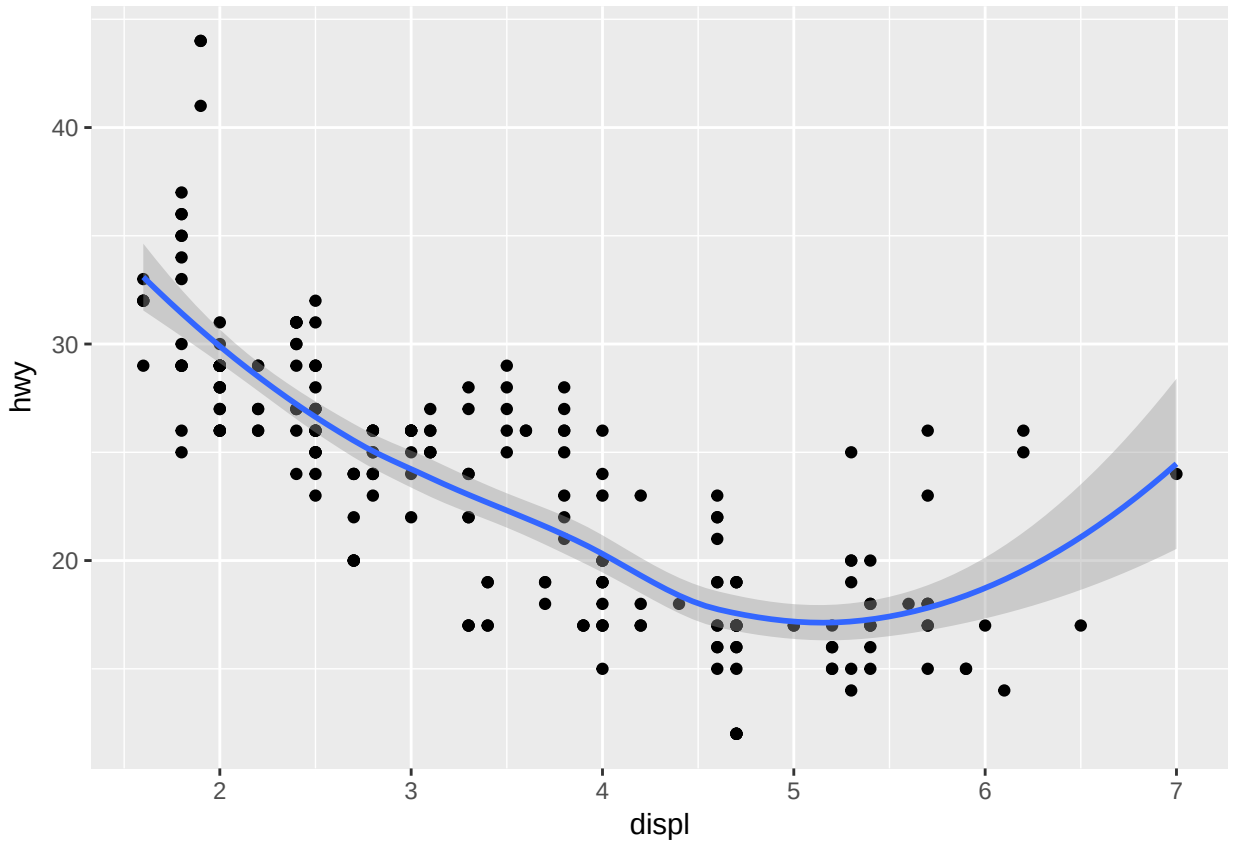
The answer for this question is inside the exercise 2 answer. The `se` argument used in `geom_smooth()` is used to specify if you want to plot with the standard errors (default or set `se = TRUE`) or not (`se = FALSE`). In the plot, the standard error is the 'grey shadow'.

Exercise 5

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

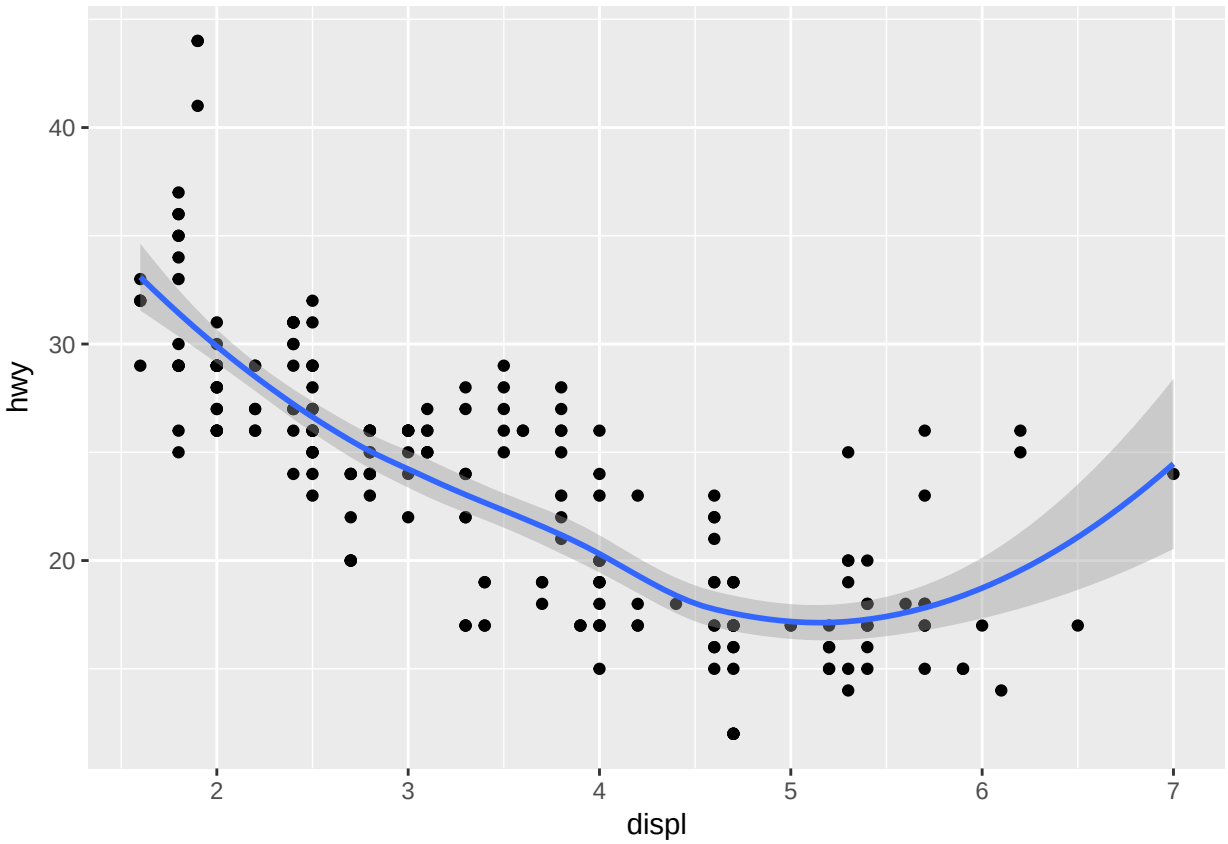
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth()
```

```
## `geom_smooth()` using method = 'loess'
```




```
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'
```



(Answer) As you can see, these two codes produce idendical plots. The first code specifies the data and mapping inside `ggplot()` function, which will automatically be used by geoms functions (in this case, `geom_point()` and `geom_smooth()`). In the second code, the data and mapping definition are specified in both geoms (duplicated code, which is bad even if works).

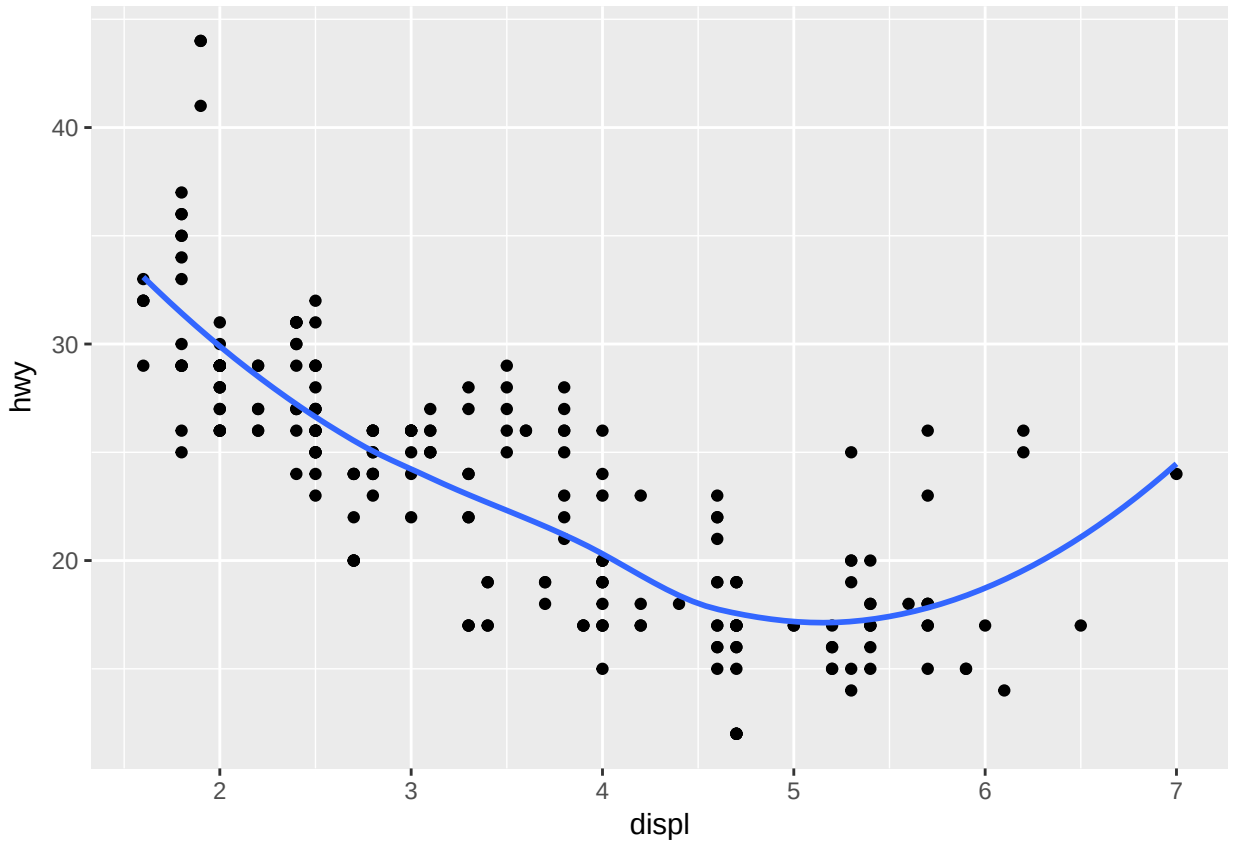
Exercise 6

Recreate the R code necessary to generate the following (6) graphs.

(Answer)

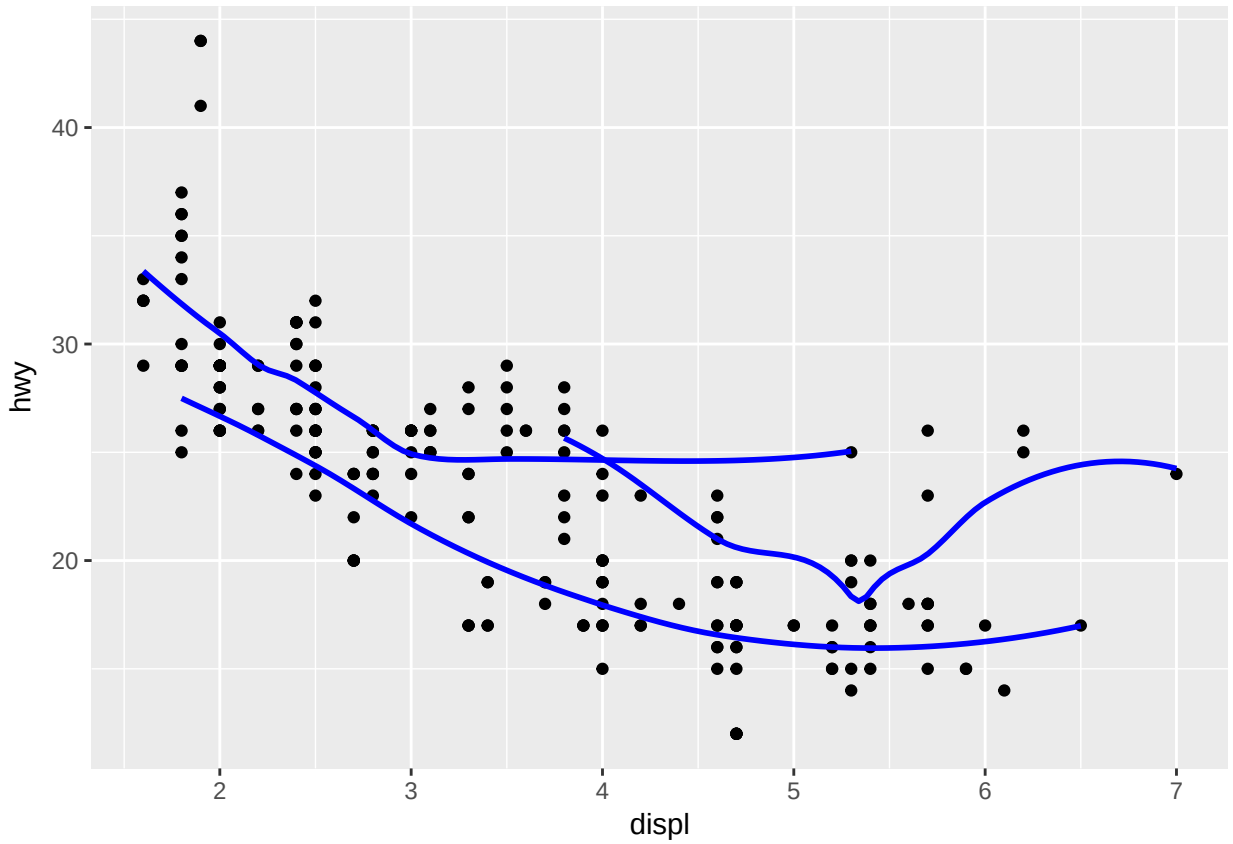
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth(se = FALSE)
```

```
## `geom_smooth()` using method = 'loess'
```



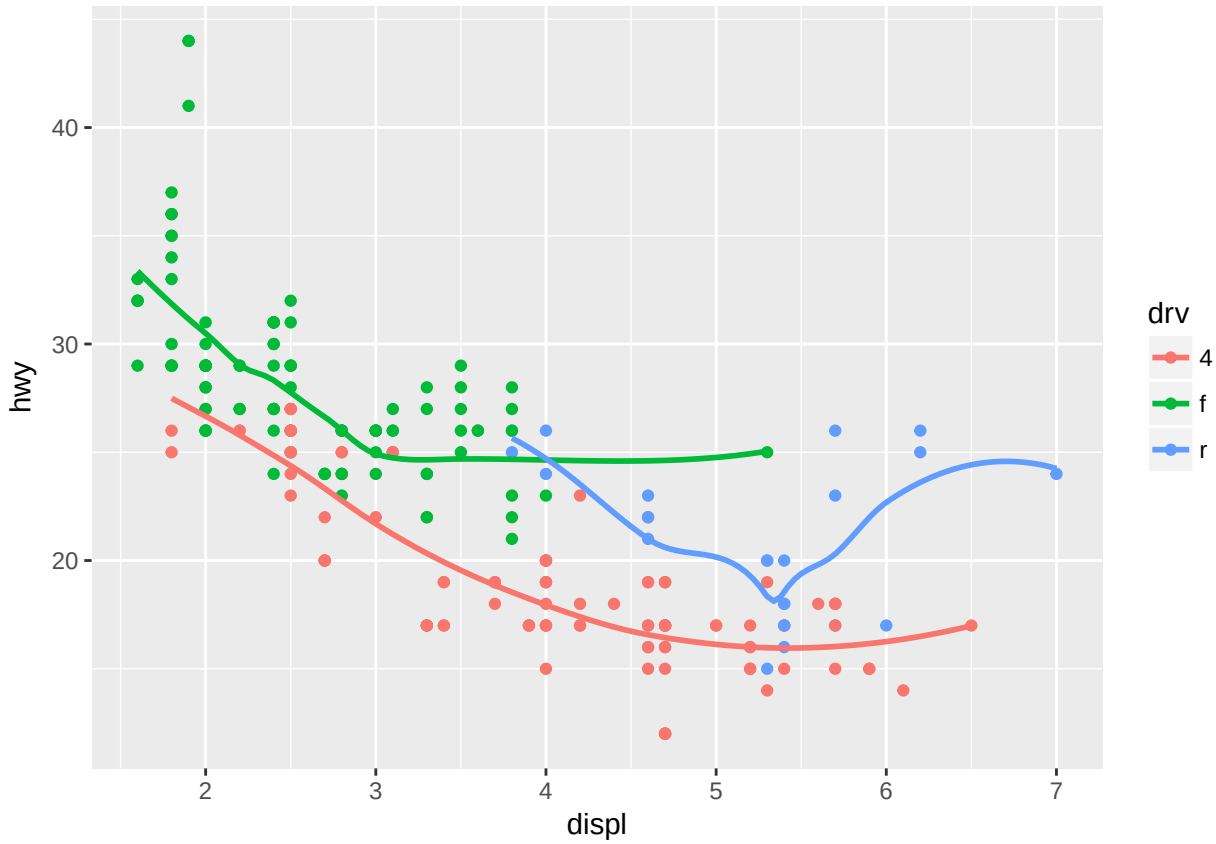

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth(mapping = aes(group = drv), color = 'blue', se = FALSE)
```

```
## `geom_smooth()` using method = 'loess'
```



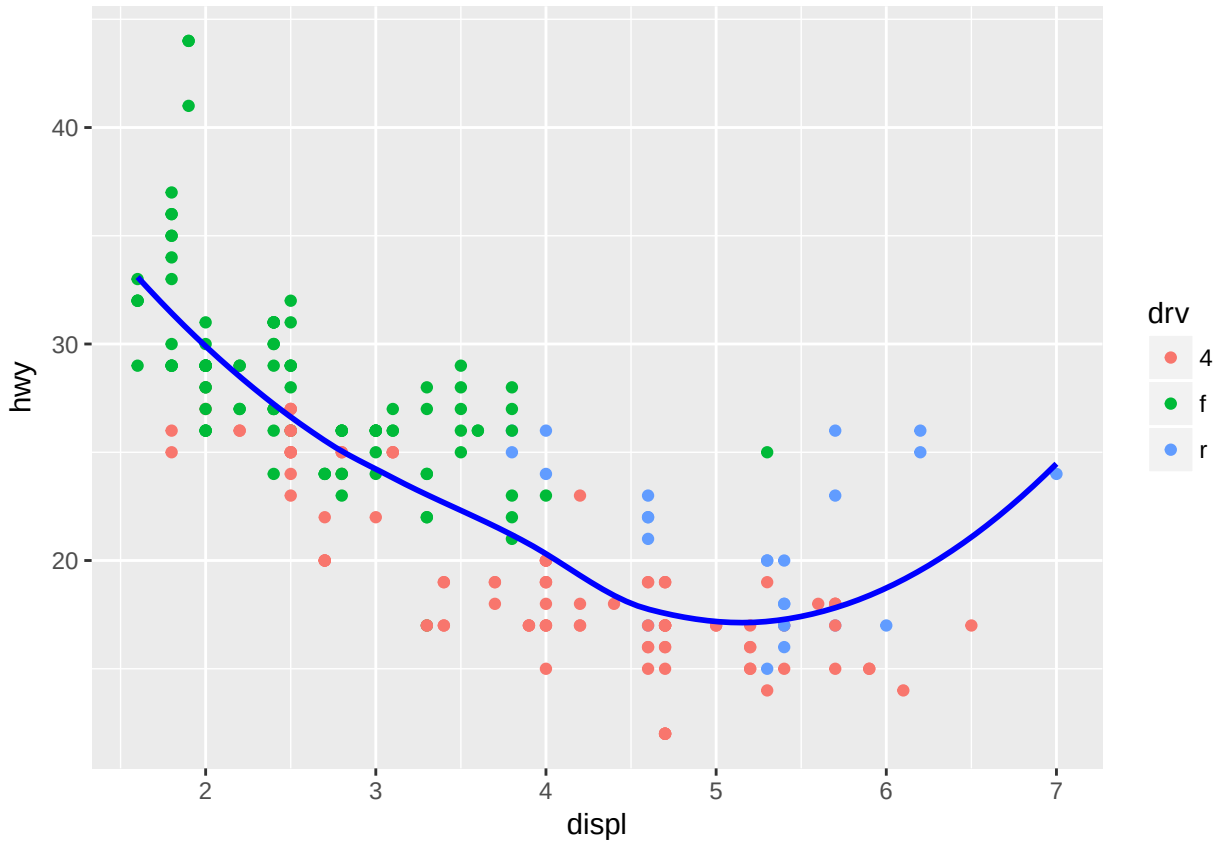

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(mapping = aes(colour = drv)) +
  geom_smooth(mapping = aes(group = drv, colour = drv), se = FALSE)
```

```
## `geom_smooth()` using method = 'loess'
```



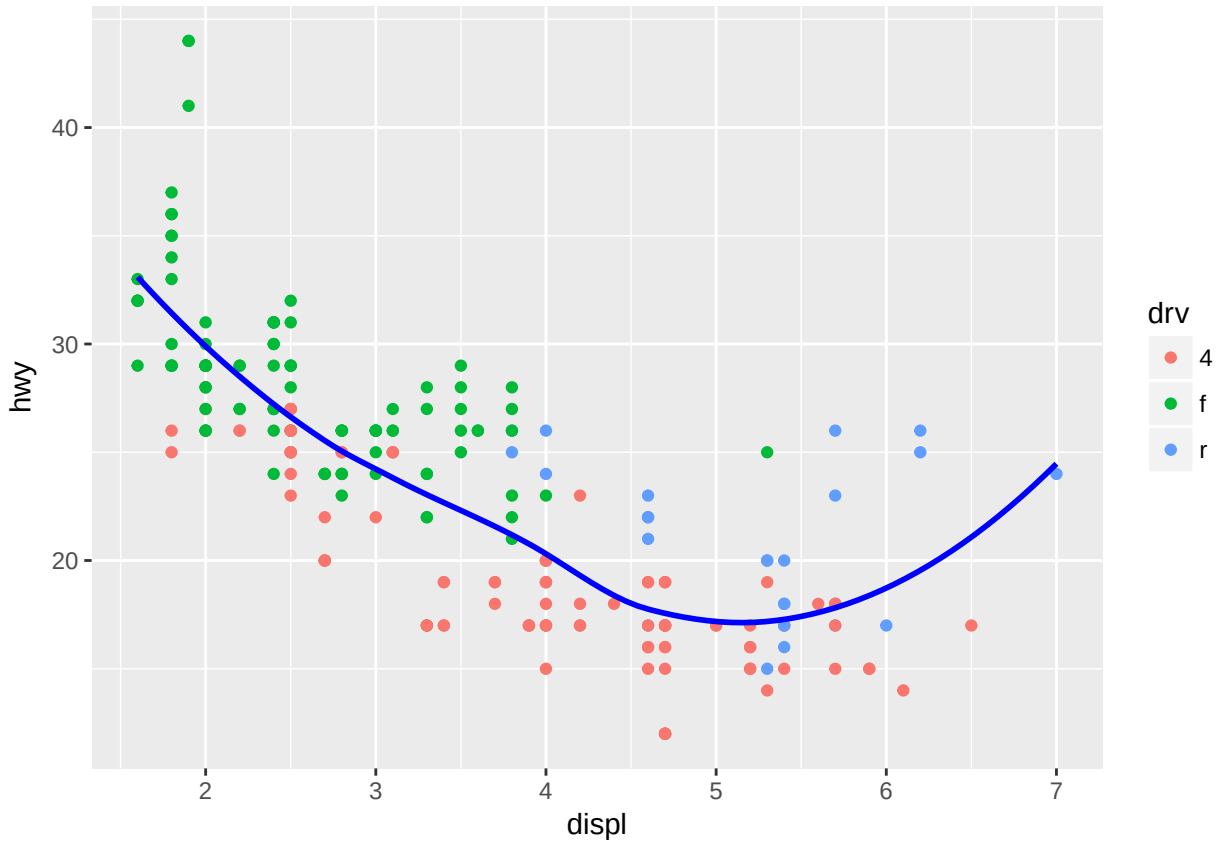

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(mapping = aes(colour = drv)) +
  geom_smooth(colour = 'blue', se = FALSE)
```

```
## `geom_smooth()` using method = 'loess'
```



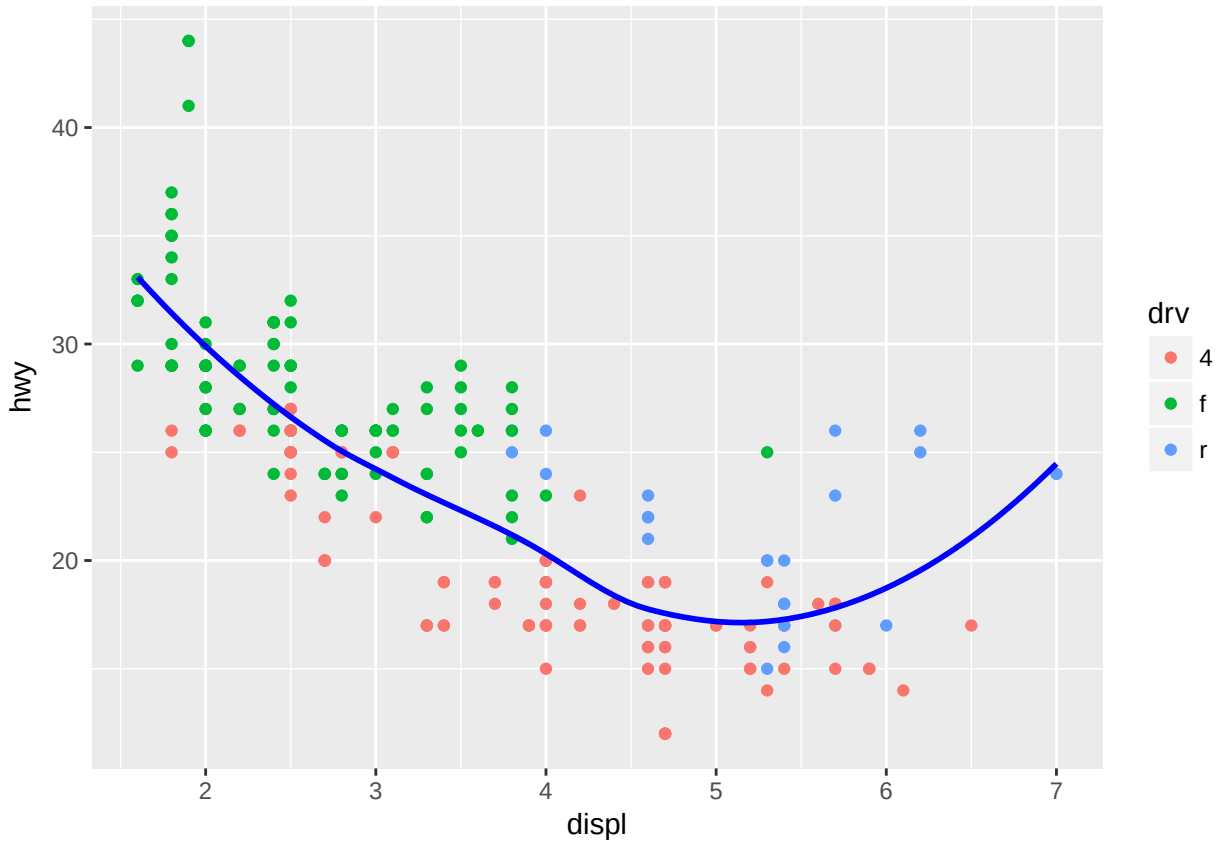

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(mapping = aes(colour = drv)) +
  geom_smooth(colour = 'blue', se = FALSE)
```

```
## `geom_smooth()` using method = 'loess'
```



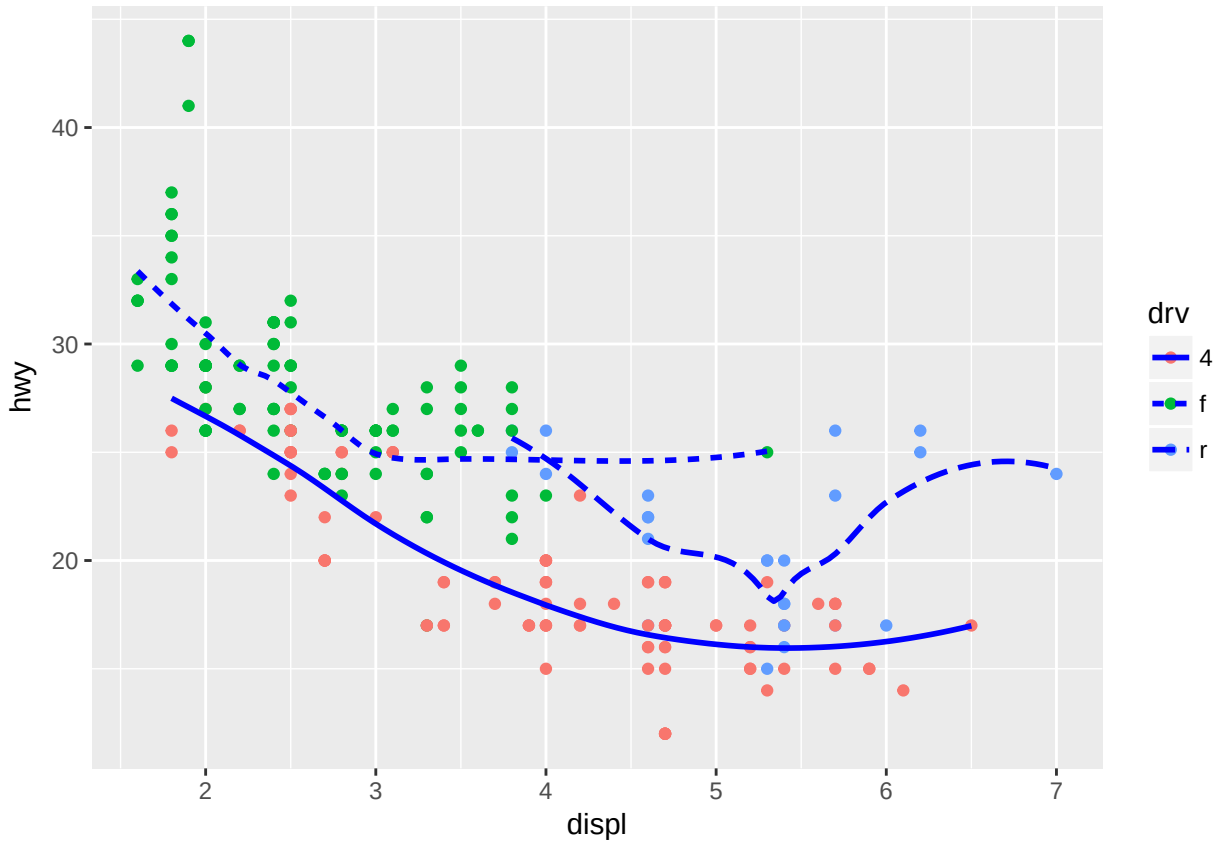

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(mapping = aes(colour = drv)) +
  geom_smooth(colour = 'blue', se = FALSE)
```

```
## `geom_smooth()` using method = 'loess'
```

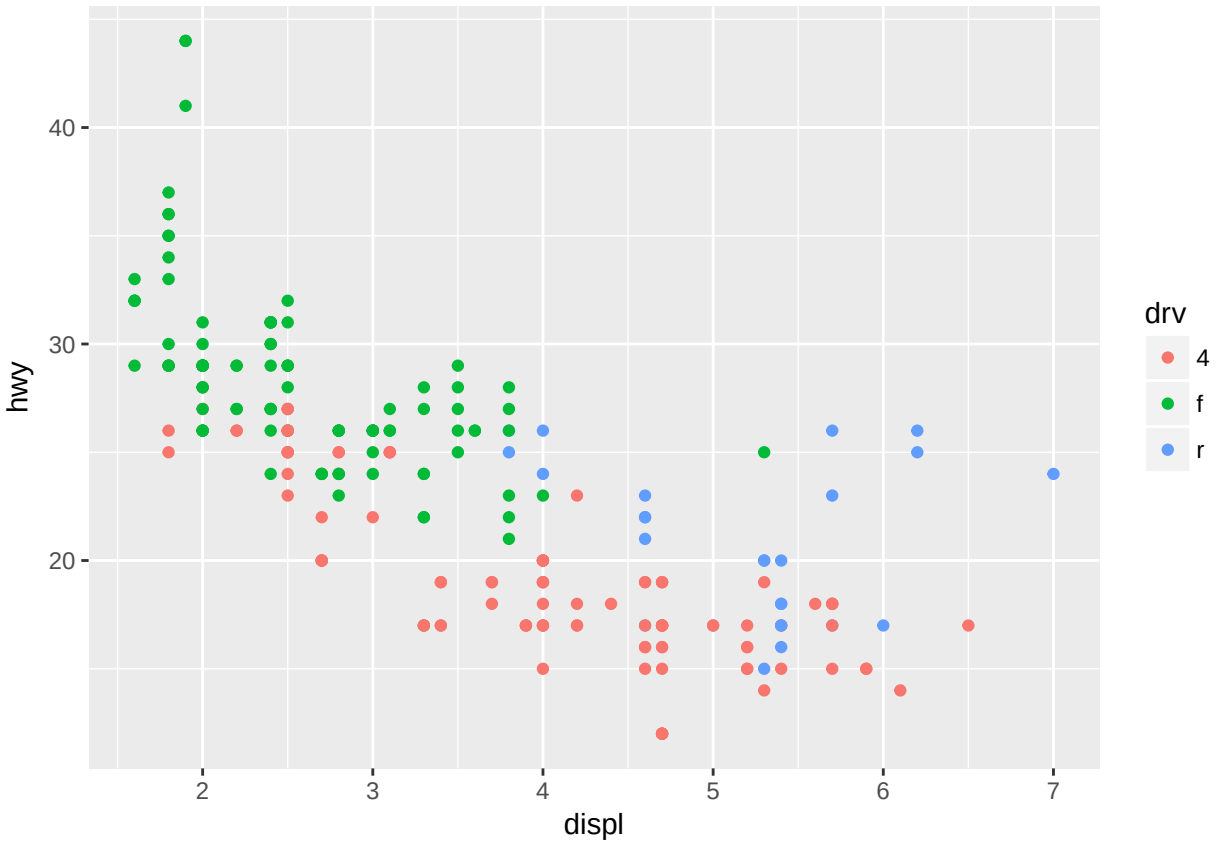



```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(mapping = aes(colour = drv)) +
  geom_smooth(mapping = aes(linetype = drv), colour = 'blue', se = FALSE)
```

```
## `geom_smooth()` using method = 'loess'
```




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



Easy peasy lemon squeezy.

3.7.1 Exercises

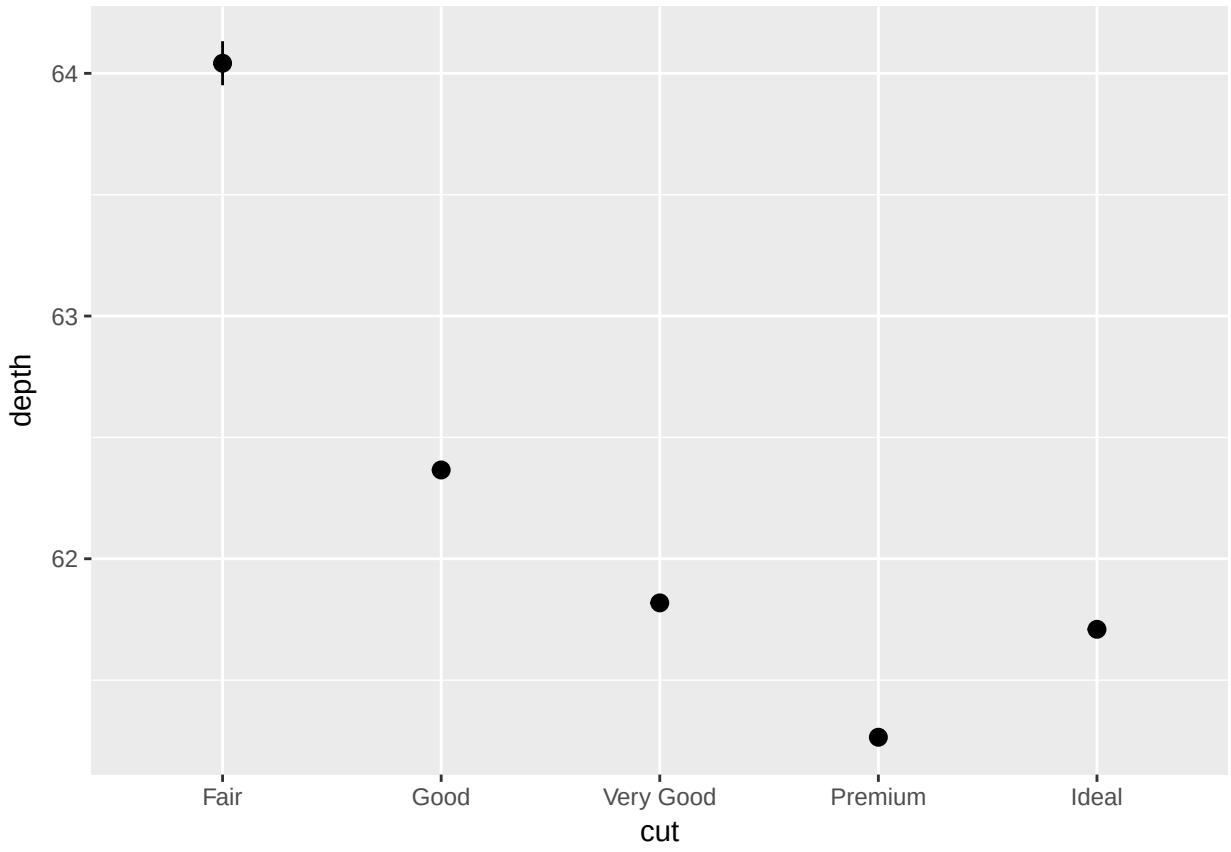
Exercise 1

> 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?

(Answer) By typing `?stat_summary()` you are able to see the documentation for this function. So, is easy to notice that the default `geom` associated with `stat_summary()` is the `geom_pointrange()` geom, which uses `identity` as the dafault `stat`. To use this `geom` to plot a `summary`, just override the default stat by using `stat = 'summary'` as follows:

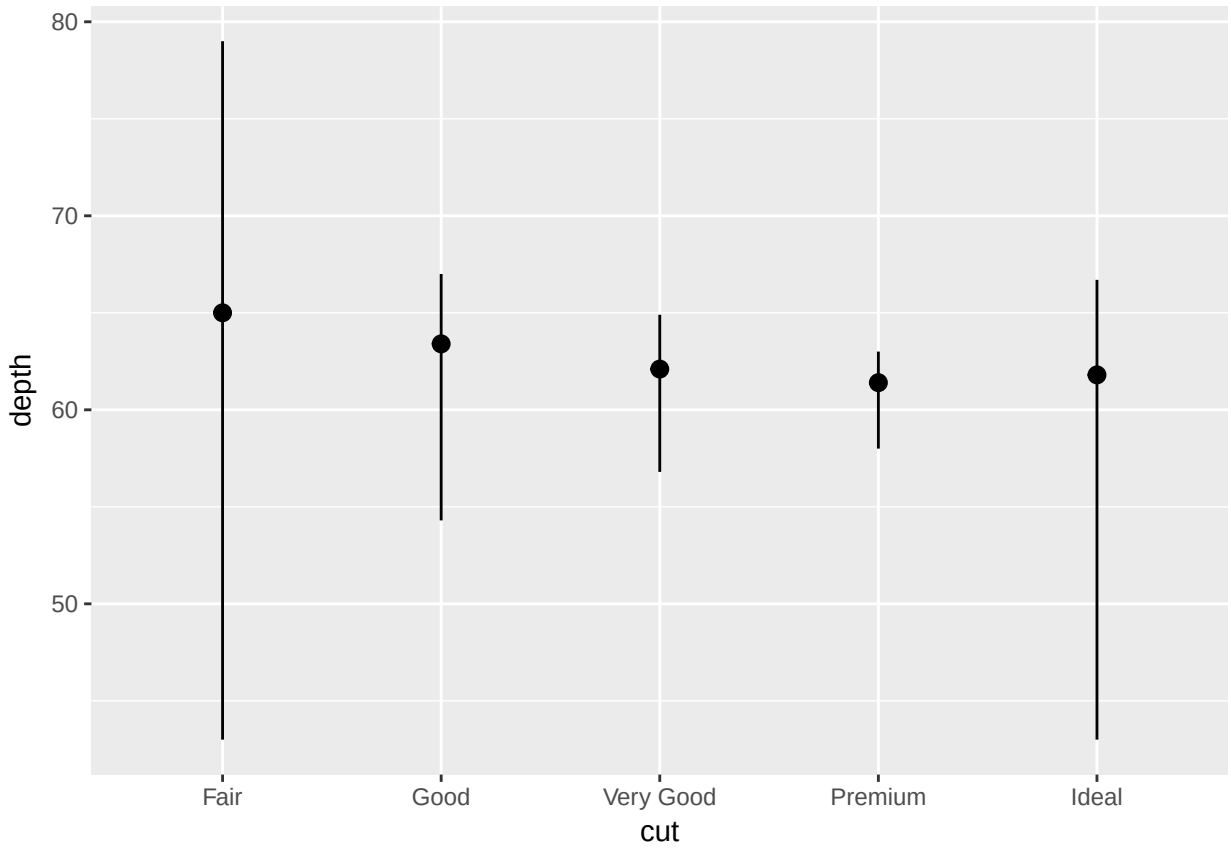
```
ggplot(data = diamonds) +
  geom_pointrange(
    mapping = aes(x = cut, y = depth),
    stat = 'summary'
  )
```

```
## No summary function supplied, defaulting to `mean_se()
```



Howerver, as you can notice that this last plot is a little bit different compared to the plot created with `stat_summary()`. It is because the default for `stat_summary()` is to use `mean` and `sd` to plt (the point and the range of the line). To fix this, just add the values used in the example (`fun.min = min`, `fun.max = max` and `fun.y = median`):

```
ggplot(data = diamonds) +
  geom_pointrange(
    mapping = aes(x = cut, y = depth),
    stat = 'summary',
    fun.ymin = min,
    fun.ymax = max,
    fun.y = median
  )
```



Voilà!

Exercise 2

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

(Answer) The answer for this question is inside `geom_col()` documentation. By typing `?geom_col()` - *I encourage you to always read the documentation for the function you want to use* - is possible to see there are two types of bar charts: `geom_bar()` makes the height of the bar proportional to the number of classes in each group (or if the `weight` aesthetic is supplied, the sum of the `weights`). If you want the heights of the bars to represent values in the data, use `geom_col()` instead. `geom_bar()` uses `stat_count` by default (it counts the number of cases at each `x` position). In other hand, `geom_col()` uses `stat_identity`, which leaves the data as is.

Exercise 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?

(Answer) The answer to this question is inside `ggplot2` documentation. I highly recommend to read the `ggplot2` documentation available **here**.

Exercise 4

What variables does `stat_smooth()` compute? What parameters control its behaviour?

(Answer) *This is the last time I am going to recommend you to always read the documentation for the functions you use.* The answer for this question is easy to find by checking `stat_smooth()` documentation. The variables computed by `stat_smooth()` are:

- `y`: predicted value
- `ymin`: lower pointwise confidence interval around the mean
- `ymax`: upper pointwise confidence interval around the mean
- `se`: standard error

And the arguments used to control its behaviour are:

- `mapping`
- `data`
- `position`
- `...`
- `method`
- `formula`
- `se`
- `na.rm`
- `show.legend`
- `inherit.aes`
- `geom, stat`
- `n`
- `span`
- `fullrange`
- `level`
- `method.args`

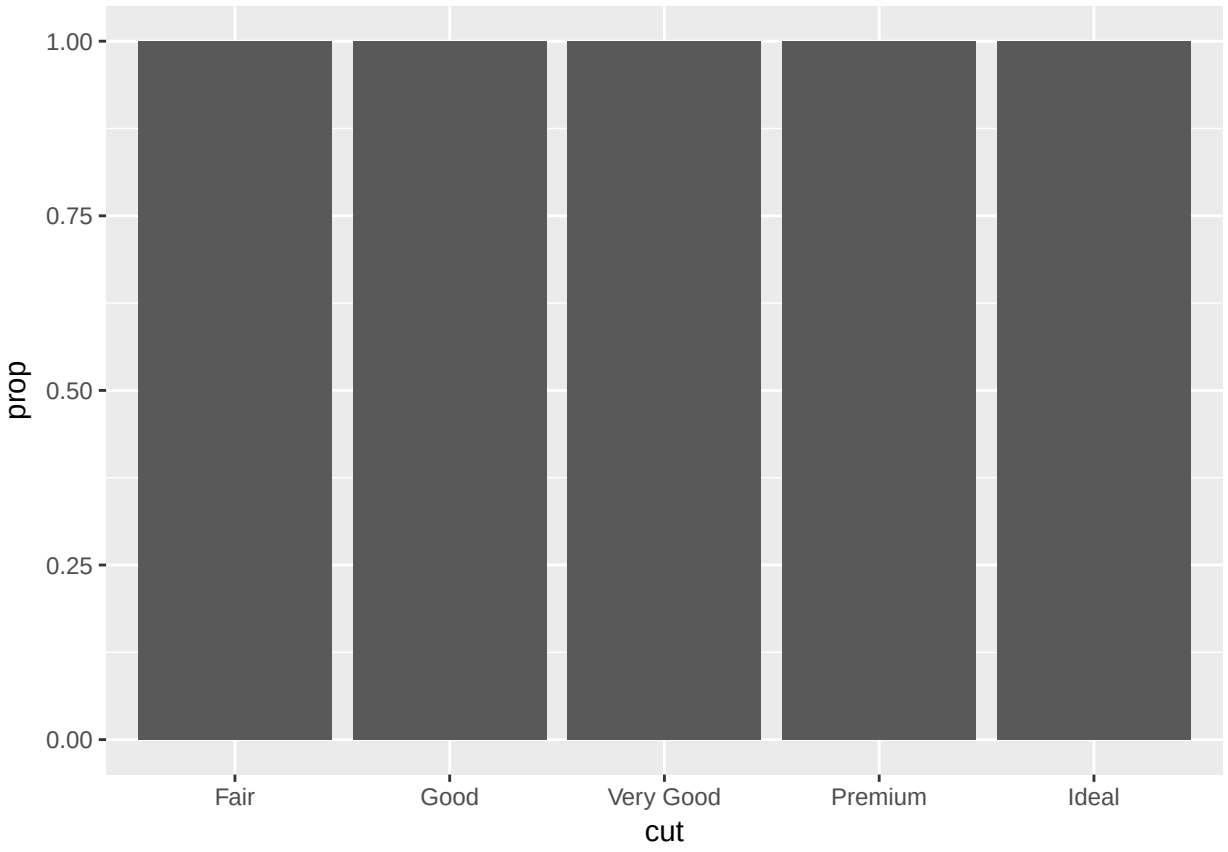
The most important argument is `mapping`, which determines which method will be used to calculate the predictions and confidence interval. To check the description for each argument, type `?stat_smooth()`.

Exercise 5

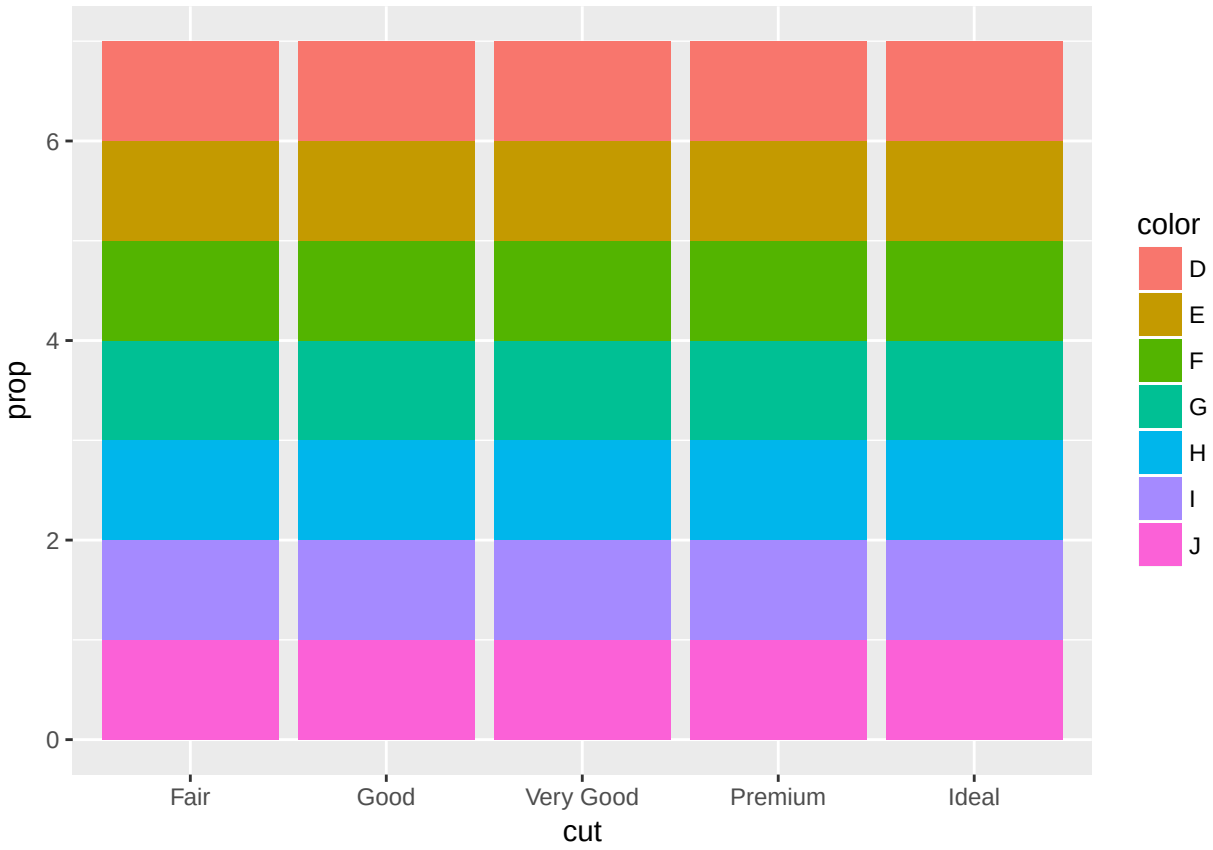
> In our proportion bar chart, we need to set `group = 1`. Why? In other words what is the problem with these two graphs?

I am no sure if my answer is one hundred percent correct for this exercise.

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, y = ..prop..))
```

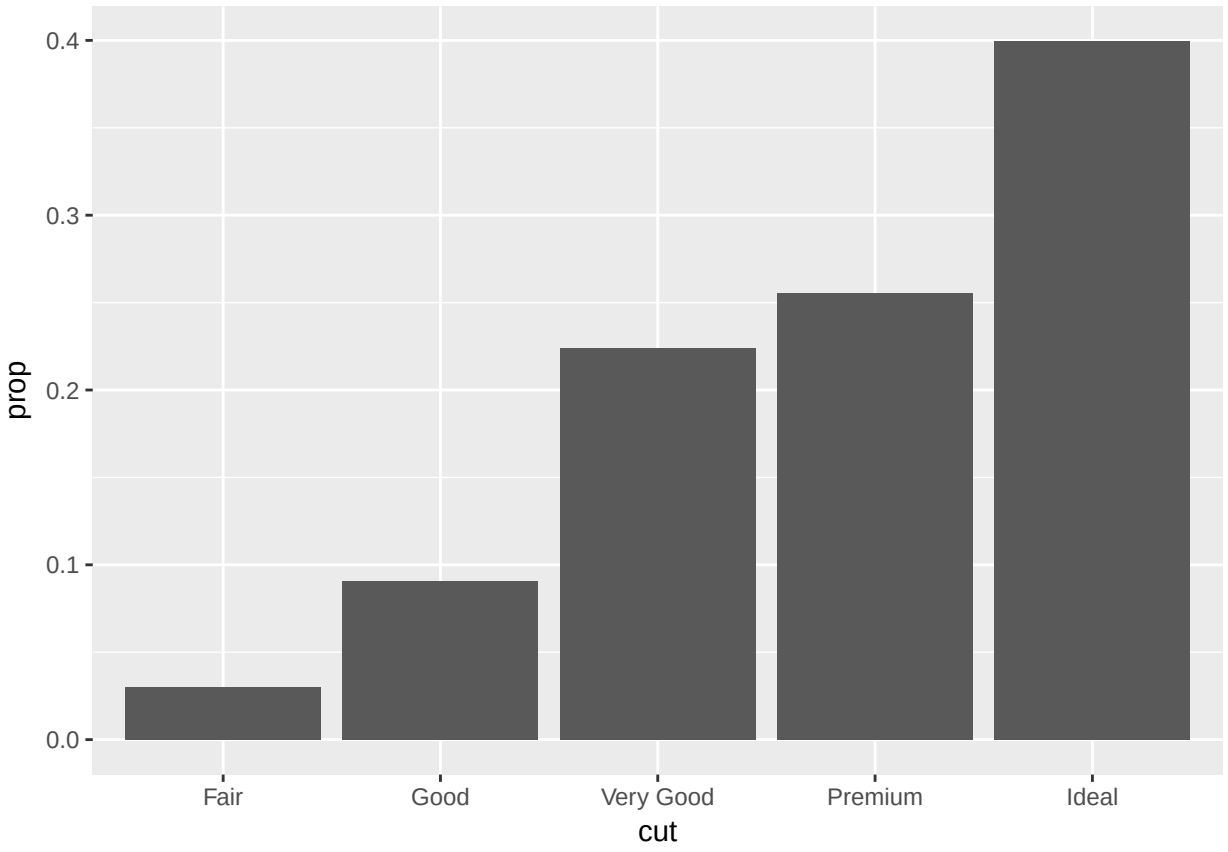



```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = color, y = ..prop..))
```

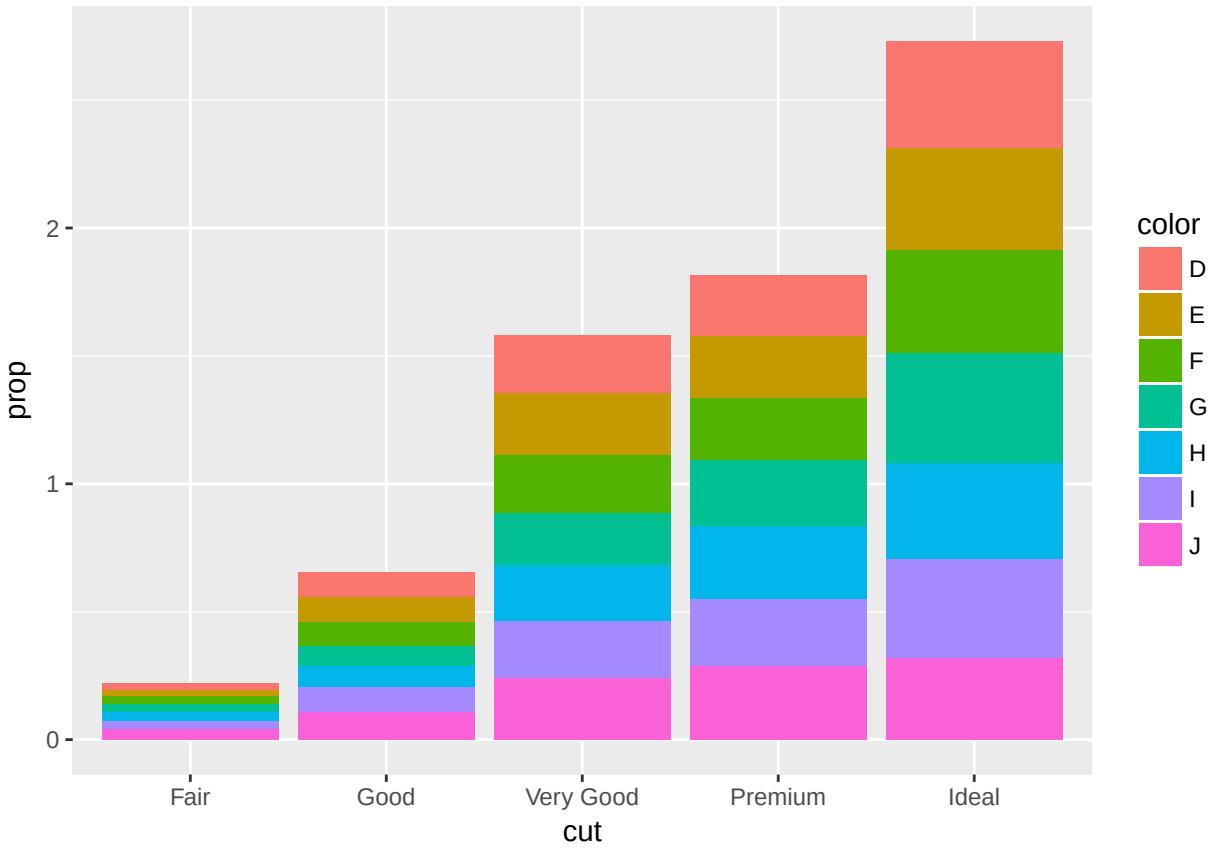


(Answer) `group = 1` is used to set the proportion (y axis) correctly. As you can see in these two plots above, the proportion for all diamonds are equals one (and this is not what we want). So the correct code would be something like this:

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, y = ..prop.., group = 1))
```




```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = color, y = ..prop.., group = color))
```

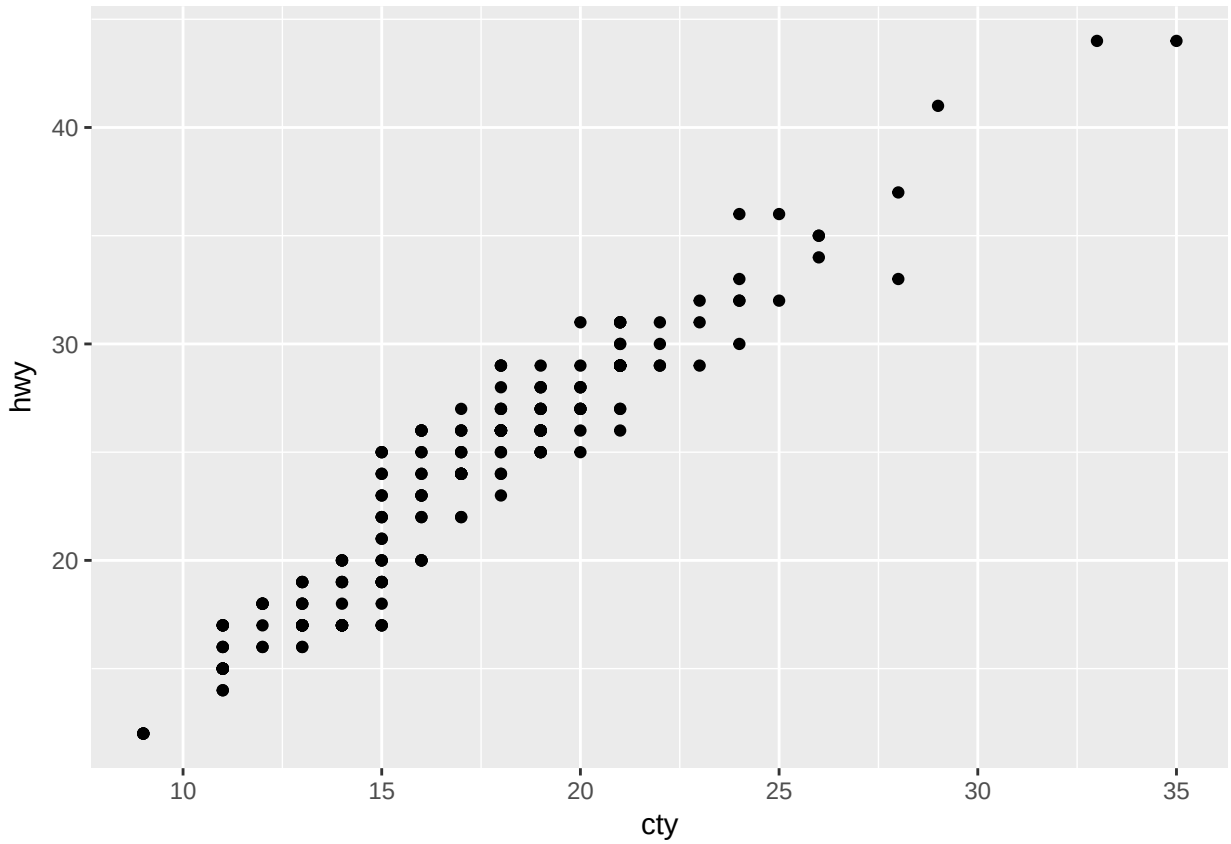


3.8.1 Exercises

Execise 1

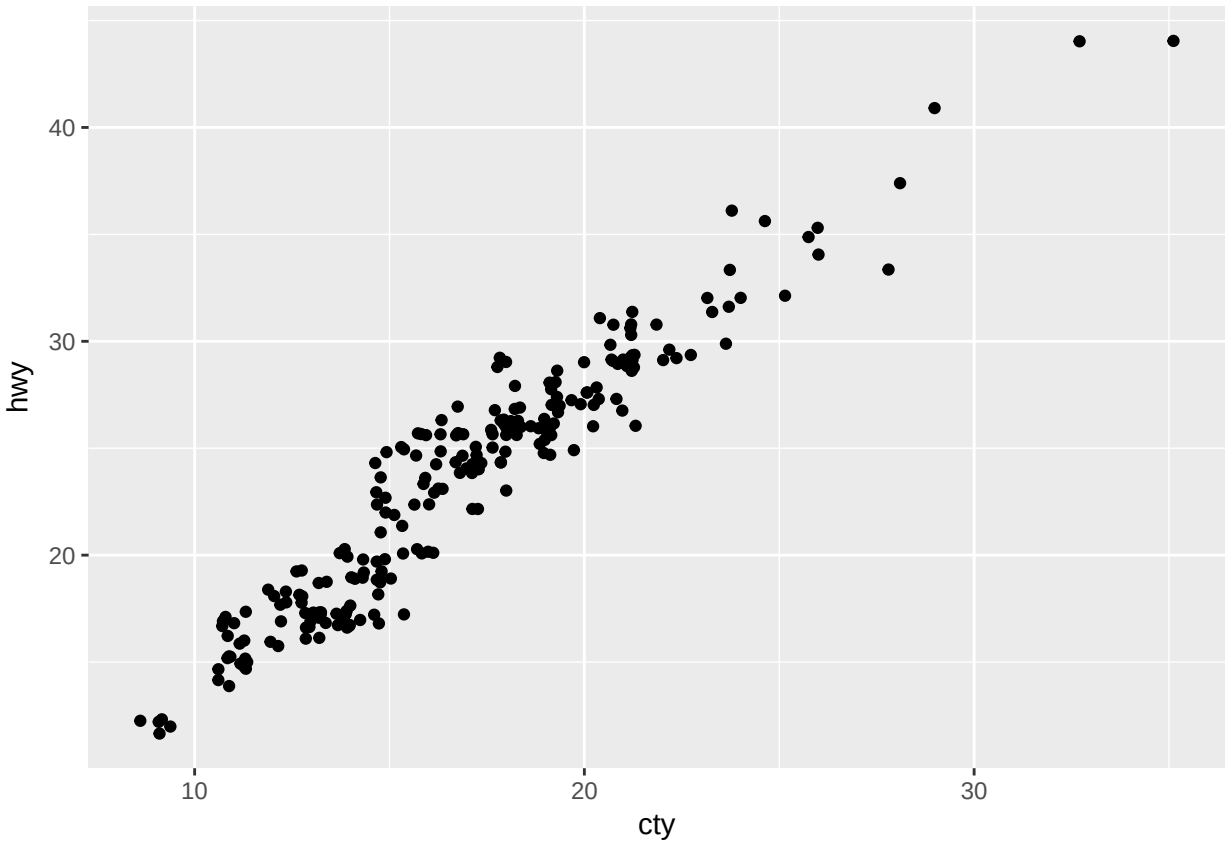
What is the problem with this plot? How could you improve it?

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



(Answer) Check this plot using same data:

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



There is a relevant difference between them, right? It is because there are a lot of observations for each combination of `cty` and `hwy`. So, for this situation `geom_jitter()` is a great option, as you can see in our last plot above.

Exercise 2

What parameters to geom_jitter() control the amount of jittering?

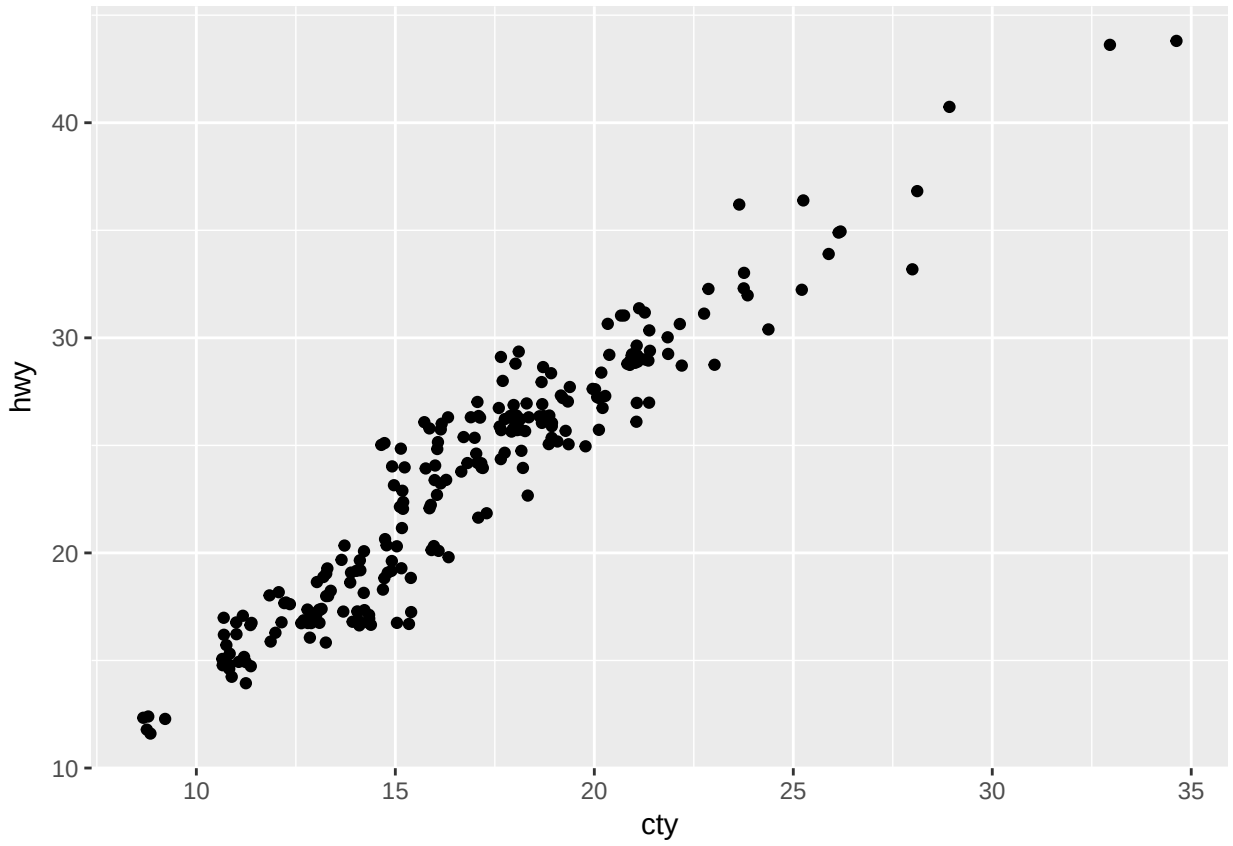
(Answer) As you can read in `position_jitter()` (or `geom_gitter()`) documentation, the parameters used to control the amount of jittering are: * **width:** amount of horizontal jitter * **height:** amount of vertical jitter The jitter is added in both positive and negative directions then the total spread is twice the value specified here. The default value is 40% of the resolution of the data. You can use with `geom_point(position = position_jitter(height, weight))` or with `geom_jitter(height, width)`.

Exercise 3

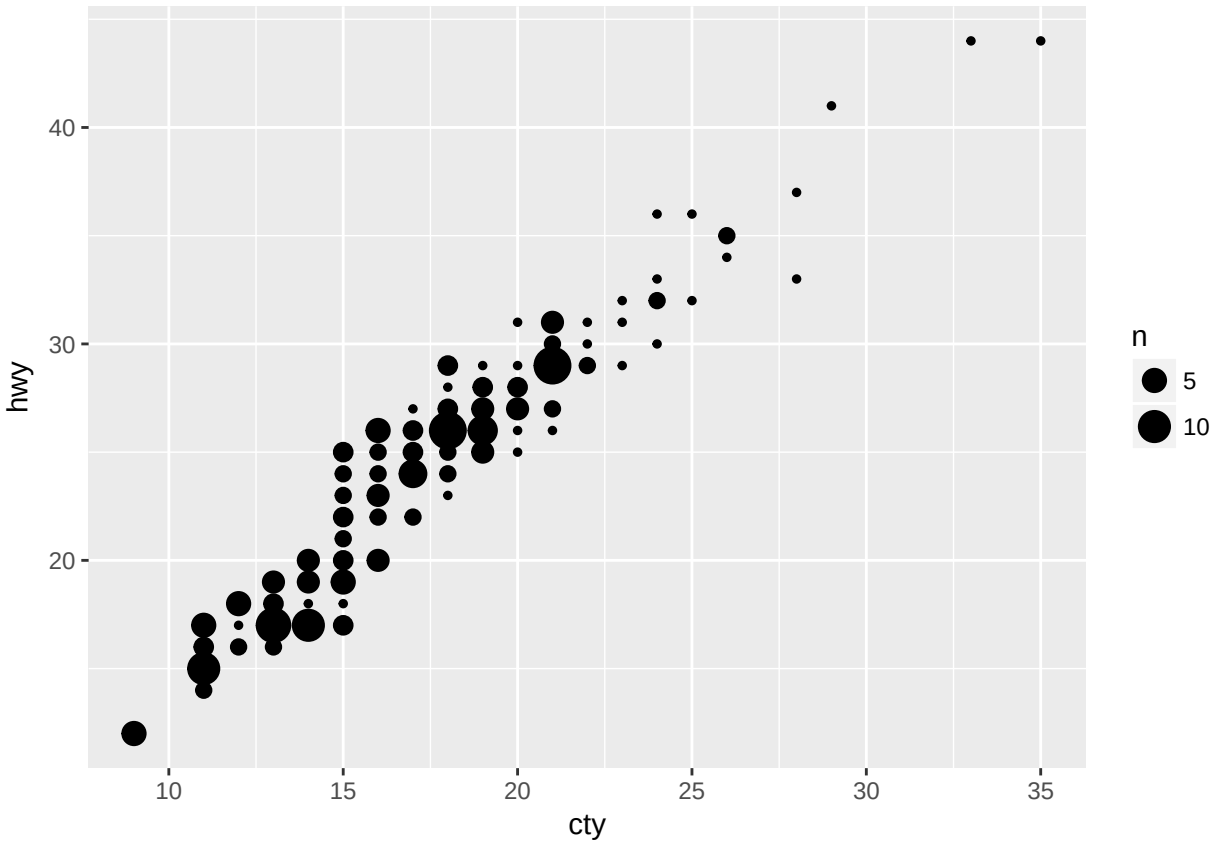
Compare and contrast geom_jitter() with geom_count().

(Answer) Let's plot two graphs, one using `geom_jitter()` and one using `geom_count()`:

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




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



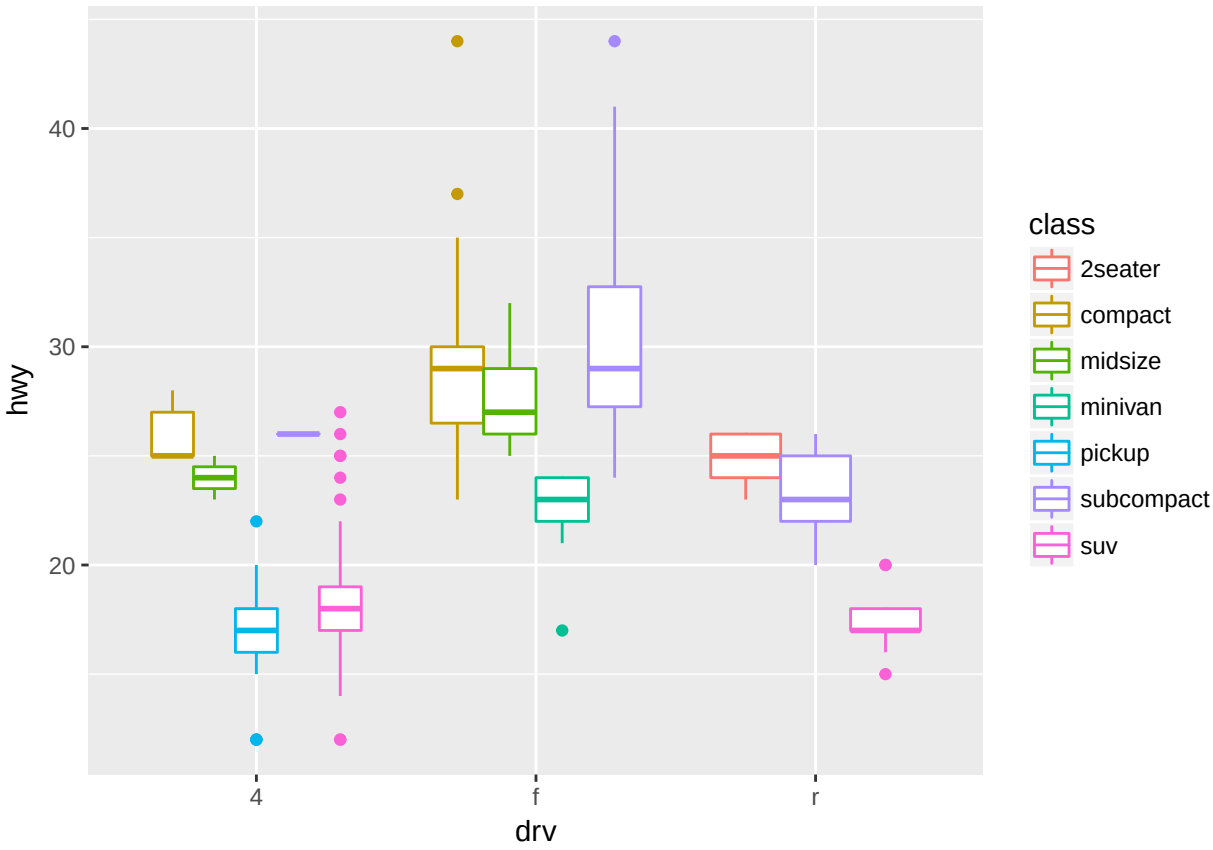
Is really easy to notice the difference between the two. In exercise 1 we verified that `geom_jitter()` adds 'noise' to our graph (both horizontally and vertically) and is easy to see this in the plot. As you can see in the last plot presented `geom_count()` makes agroupation of points and adds a legend to show the scale. In spite of the difference between the two functions, both are useful to understand better where are the concentrations of your dataset.

Exercise 4

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

(Answer) By checking the `geom_boxplot()` documentation you are able to verify that the default position for `geom_boxplot()` is `dodge`. Let's plot using `geom_boxplot()` without any custom argument:

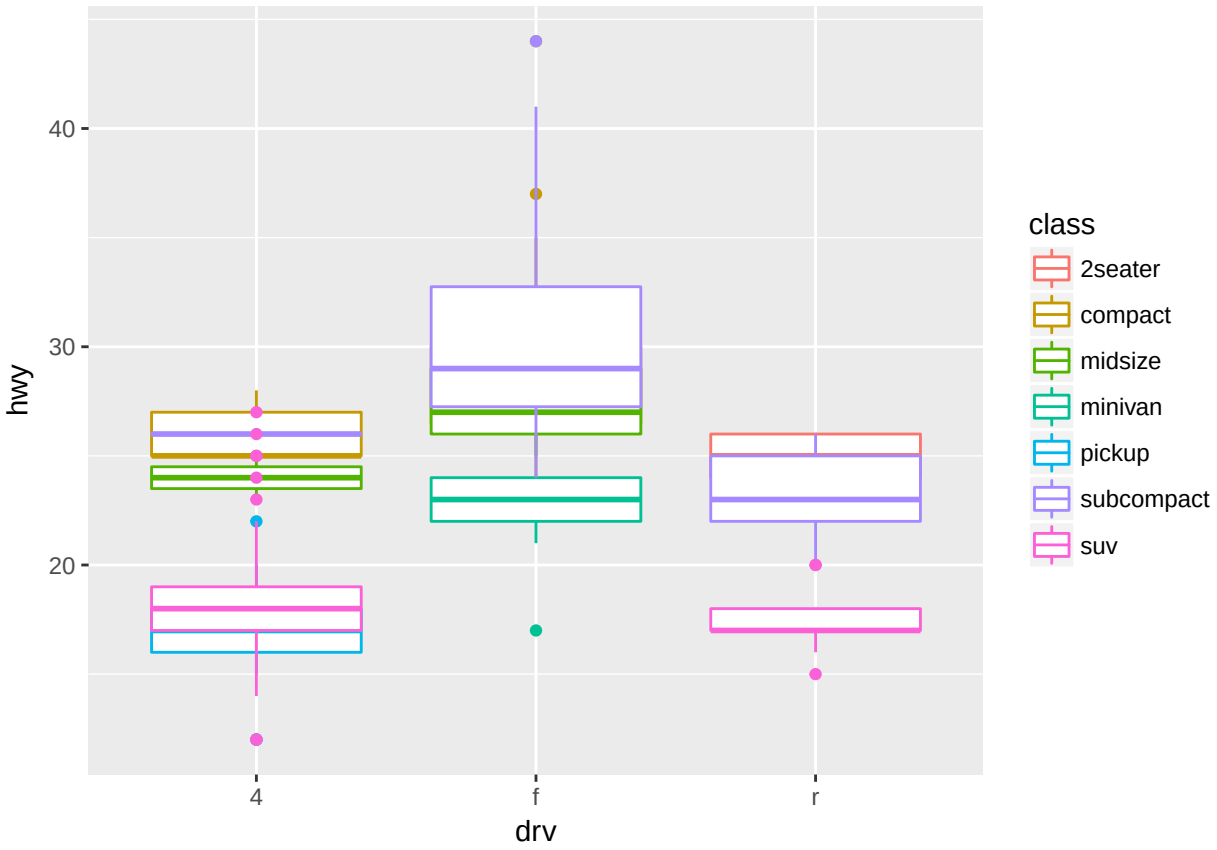
```
ggplot(data = mpg, aes(x = drv, y = hwy, color = class)) +
  geom_boxplot()
```



As you can see in this plot, the different classes from `drv` are side by side and it is because `geom_boxplot()` uses `dodge` as default position.

Now, let's plot overriding the default position adjustment:

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



In this last plot, the different classes from `drv` are not side by side anymore, they are overlapped! This is because now the `geom_boxplot()` is using `identity` as position adjustment insted of `dodge`.