Solutions to G. Grolemund & H. Wickhams's R for Data Science, Chapter 3

Krista DeStasio

7/11/2017

Contents

A Brief Introduction to This File

This R file walks through G. Grolemund & H. Wickhams's online text, "R for Data Science." Much of the code is sourced directly from the book and credit belongs to the authors. Here, some sections of code are heavily commented so that the beginning R programmer can read through and understand what each line of code does and compare it to their own as they work through the text. Throughout, the book provides the primary and most thorough explanation. **For the greatest learning benefit, I suggest you attempt each exercise on your own before looking at the code or write-ups provided here.** Of course, there is more than one way to write code and you may find a more elegant solution that you prefer.

For those new to R and RStudio, it may be of additional benefit to knit the document and examine how the code in the Rmd file is visually expressed in the resultant knitted document. For example, see how the `["R for Data Science."](http://r4ds.had.co.nz/index.html)` is expressed as a hyperlink in the preceeding paragraph where it was not surrounded by tick-marks and compare that to how the same text is expressed in this paragraph when surrounded by ticks. See also the difference in appearance when knitting to different document types (HTML, PDF, Word).

Tip: *If you are using RStudio, click the text next to the orange # box at the bottom of the editor window to*

easily navigate the code chunks.

Tip: *Use the* `?` *before any command to view the documentation on that function. Do this often. For example, type* `?setwd` *to see a description, usage, arguments, and more for the function* `setwd()`.

Tip: Find RStudio Cheatsheets at https://www.rstudio.com/resources/cheatsheets/

Chapter 3, Data Visualisation

To really understand ggplot2, I highly recommend reading "The Layered Grammar of Graphics" as suggested at the beginning of Chapter 3.

The `mpg` data frame

```
str(mpg) # Look at the structure of the mpg data frame
```

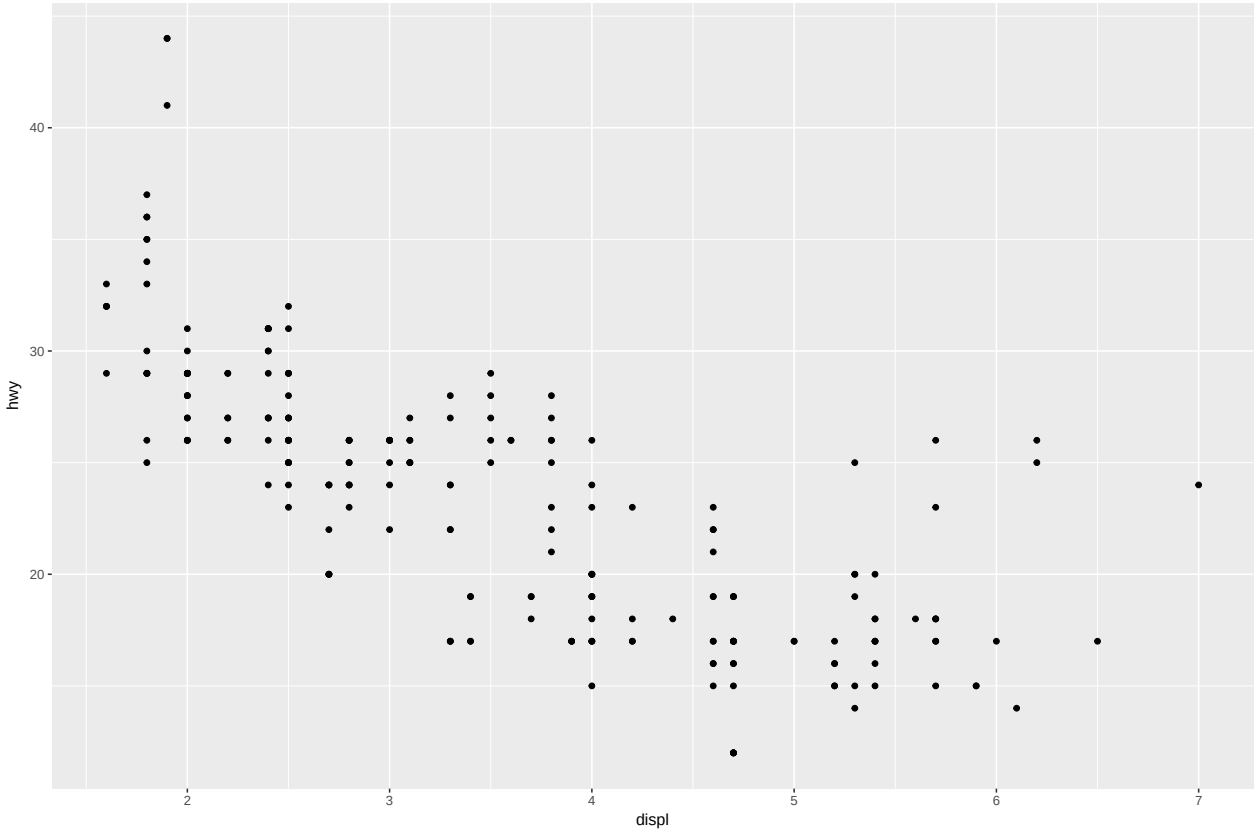
```
## Classes 'tbl_df', 'tbl' and 'data.frame':    234 obs. of  11 variables:
##  $ manufacturer: chr  "audi" "audi" "audi" "audi" ...
##  $ model       : chr  "a4" "a4" "a4" "a4" ...
##  $ displ       : num  1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
##  $ year        : int  1999 1999 2008 2008 1999 1999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

```
mpg # Look at the first 10 rows of the mpg data frame
```

```
## # A tibble: 234 x 11
##    manufacturer      model displ  year   cyl      trans   drv   cty   hwy
##           <chr>      <chr> <dbl> <int> <int>      <chr> <chr> <int> <int>
##  1         audi         a4   1.8  1999     4   auto(l5)     f    18    29
##  2         audi         a4   1.8  1999     4 manual(m5)     f    21    29
##  3         audi         a4   2.0  2008     4 manual(m6)     f    20    31
##  4         audi         a4   2.0  2008     4   auto(av)     f    21    30
##  5         audi         a4   2.8  1999     6   auto(l5)     f    16    26
##  6         audi         a4   2.8  1999     6 manual(m5)     f    18    26
##  7         audi         a4   3.1  2008     6   auto(av)     f    18    27
##  8         audi a4 quattro   1.8  1999     4 manual(m5)     4    18    26
##  9         audi a4 quattro   1.8  1999     4   auto(l5)     4    16    25
## 10         audi a4 quattro   2.0  2008     4 manual(m6)     4    20    28
## # ... with 224 more rows, and 2 more variables: fl <chr>, class <chr>
```

Hypothesis: There is a negative linear relationship between engine size and fuel efficiency, such that as engine size increases fuel efficiency decreases.

```
ggplot(data=mpg) + # specify data frame
    geom_point(mapping = aes(x = displ, y = hwy)) # specify that plot is a scatterplot with displ on th
```



The plot confirms the hypothesis that there is a negative relationship between engine size and fuel efficiency.

Making graphs with ggplot2

Template:

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

Exercises 3.2.4

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

There are no visible results from the code below.

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

2. How many rows are in mpg? How many columns?

Based on the output from `str(mpg)`, we see that there are 234 rows and 11 columns in the mpg data frame.

```
# Alternative means of finding number of rows and columns
nrow(mpg) # Pring the number of rows
```

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

```
ncol(mpg)
```

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

There are 234 rows and 11 columns in the mpg data frame.

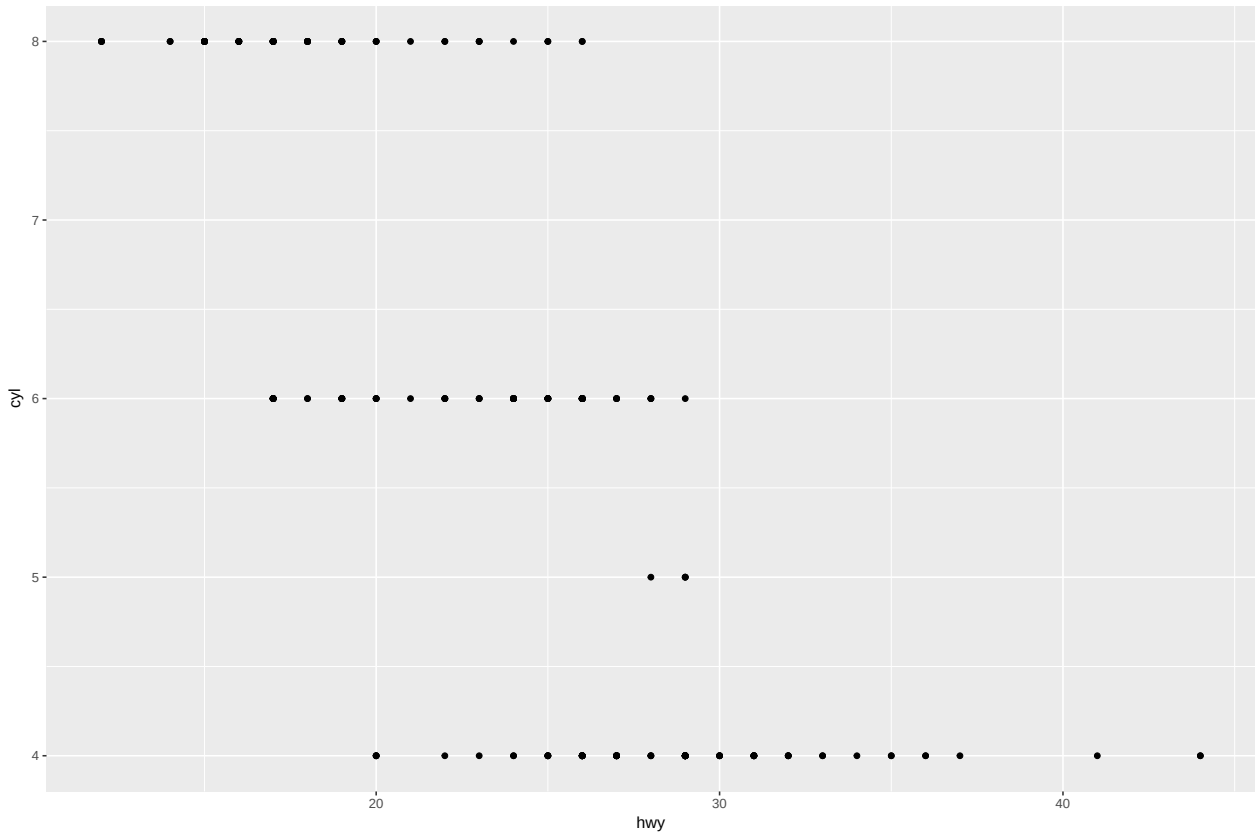
3. What does the drv variable describe? Read the help for ?mpg to find out.

The `drv` variable describes whether the vehicle is front, rear, or 4-wheel drive.

```
?mpg
```

4. Make a scatterplot of hwy vs cyl.

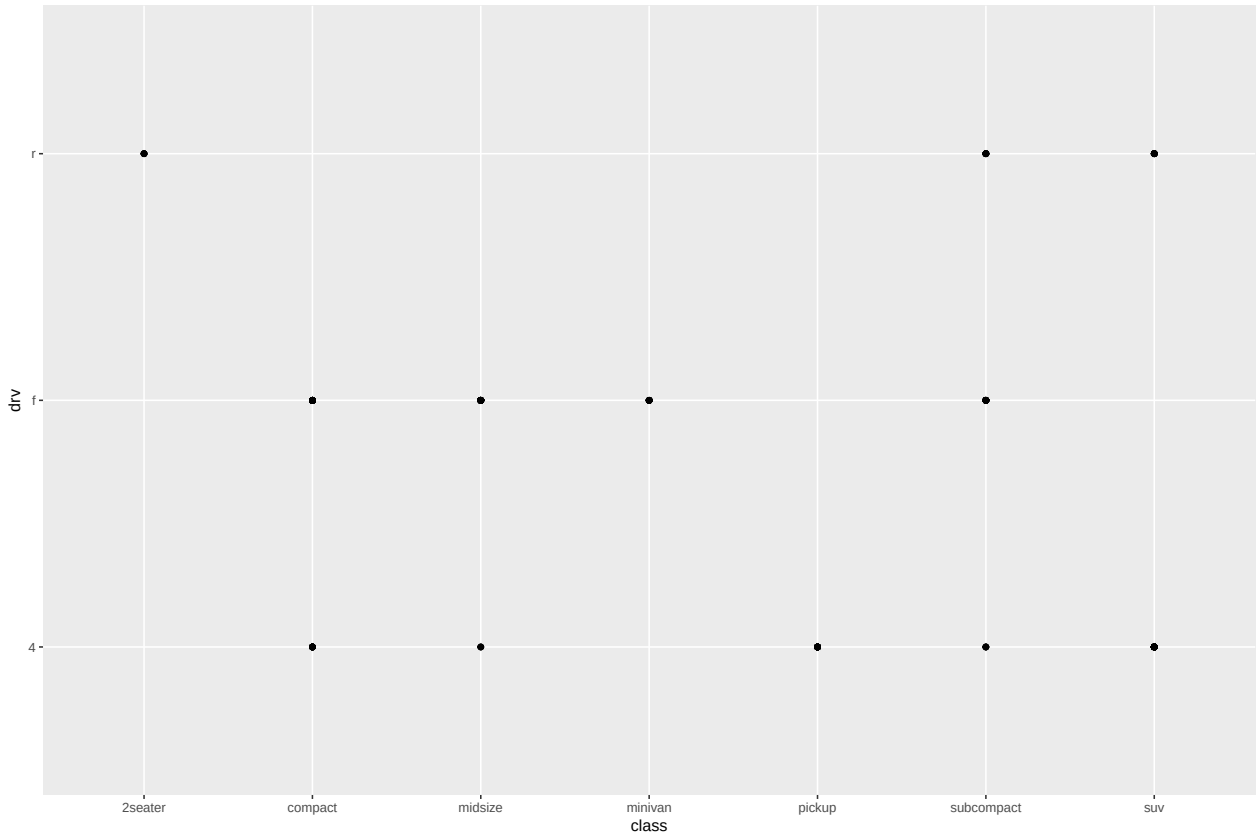
```
ggplot(data=mpg) +
    geom_point(mapping = aes(x=hwy, y=cyl))
```



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

The plot is not useful because the variables are categorical and multiple points are plotted atop one another. We are unable to determine from this plot how many observations there are of each class-drive combination.

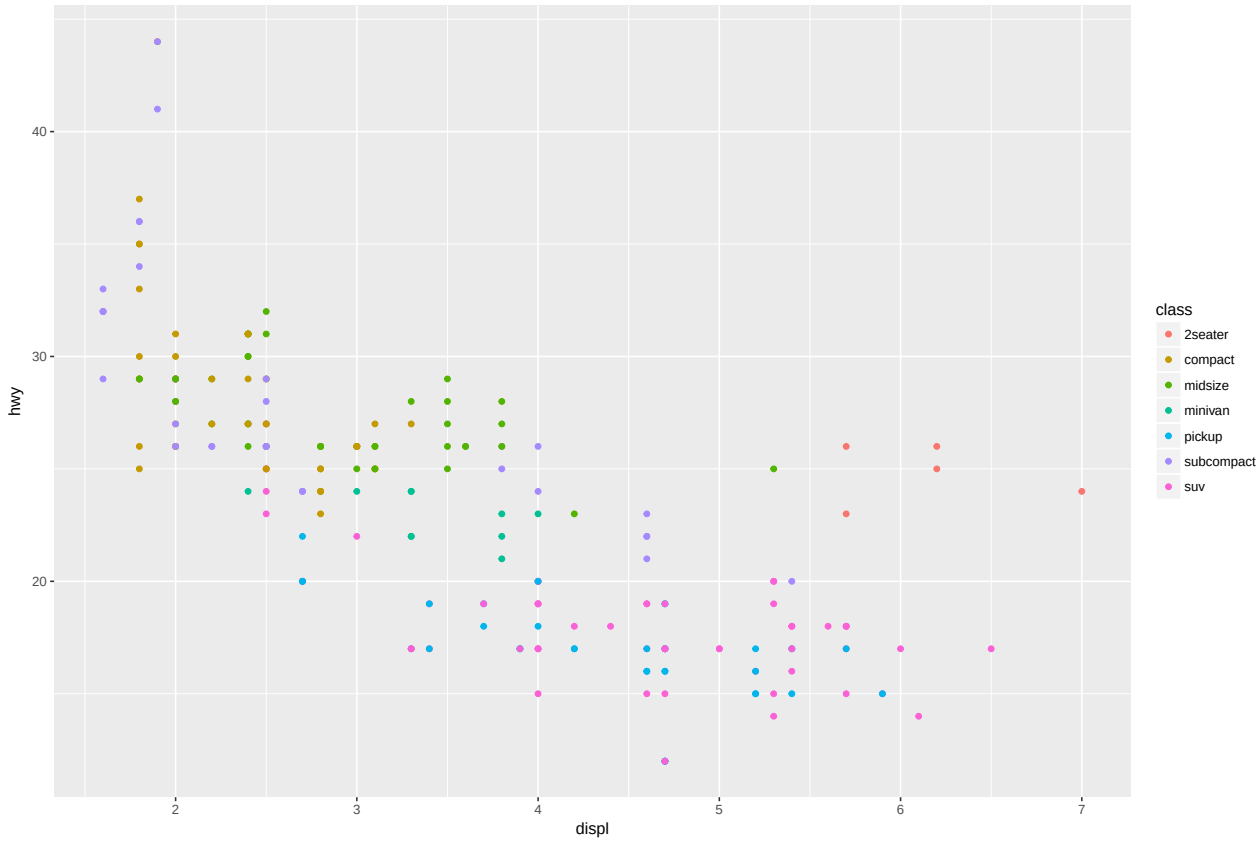
```
ggplot(data = mpg) +
    geom_point(mapping = aes(x=class, y=drv))
```



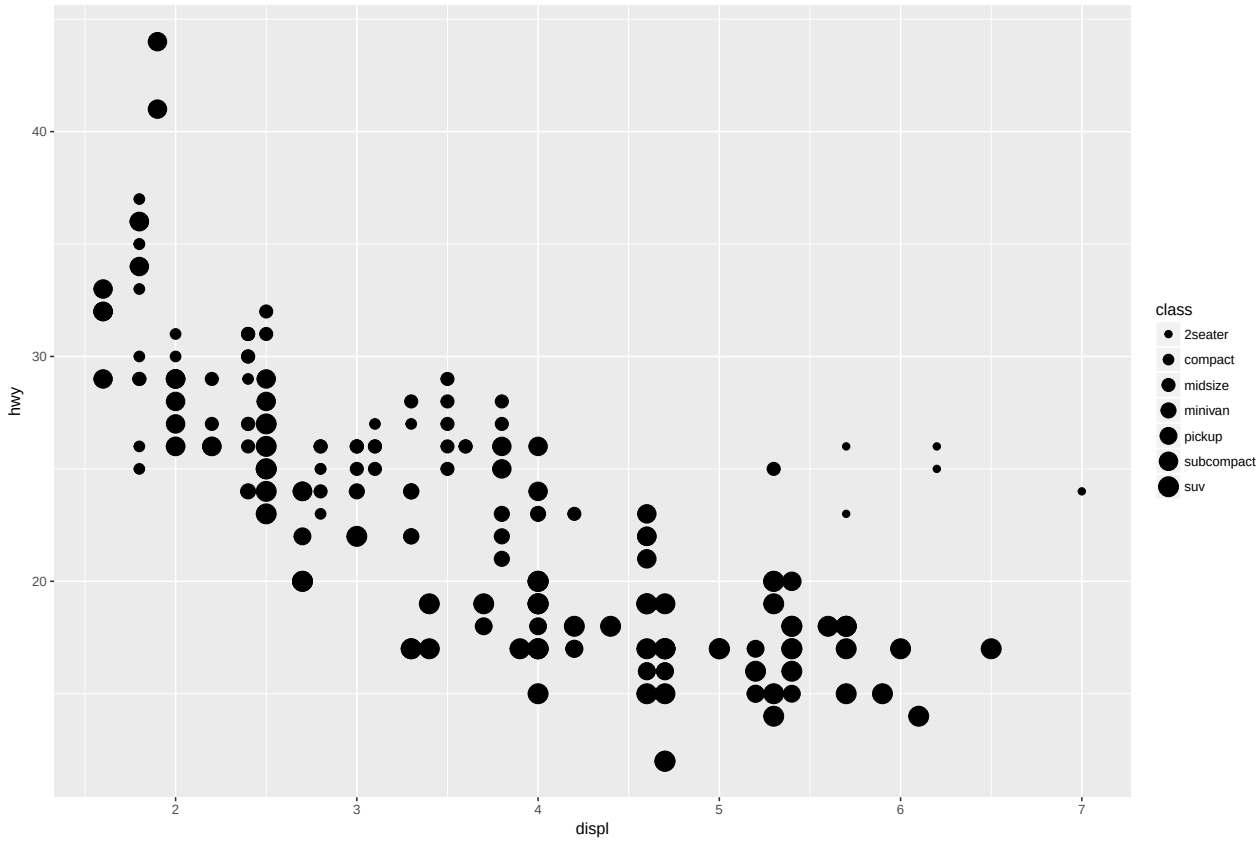
Aesthetic mappings

Test the hypothesis that the cars highlighted in red are hybrids by mapping car class to an aesthetic.

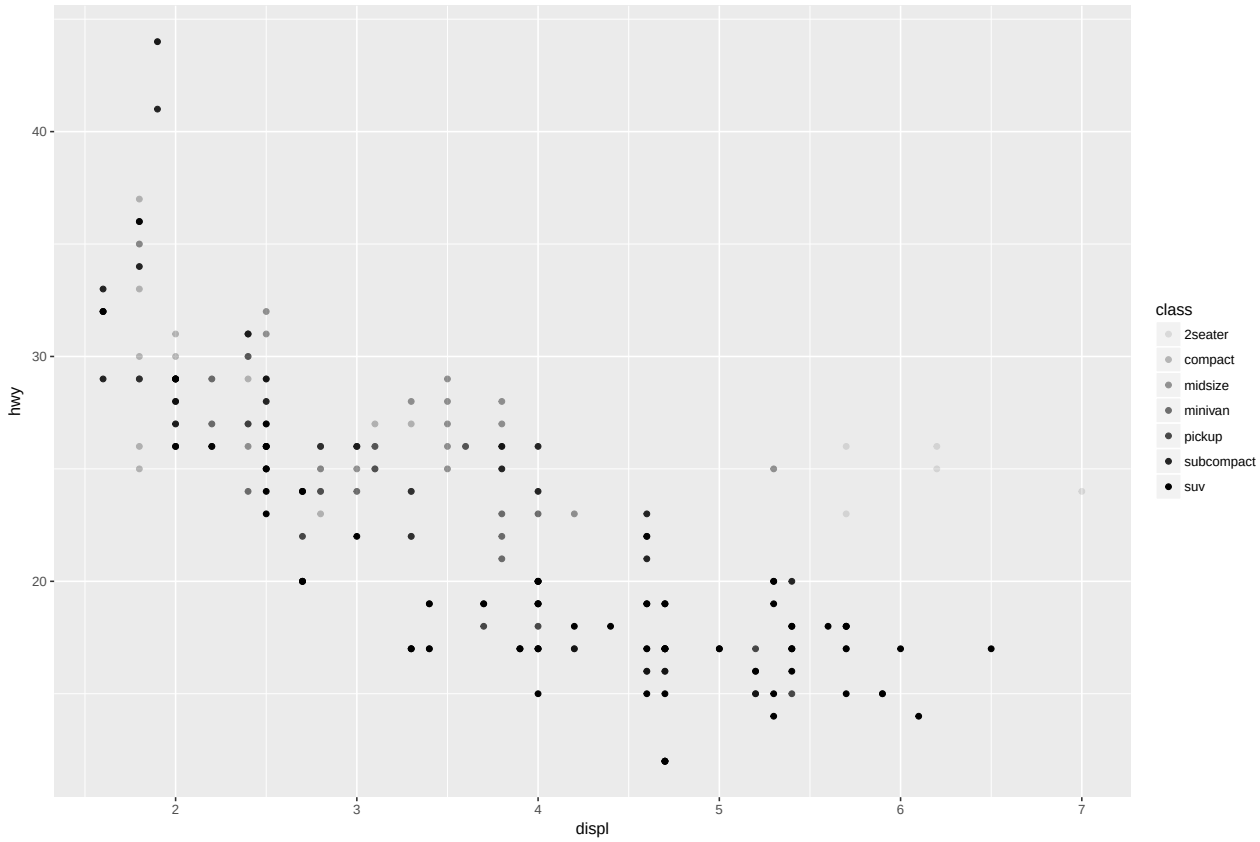
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = class)) # map class to the color aesthetic so th
```



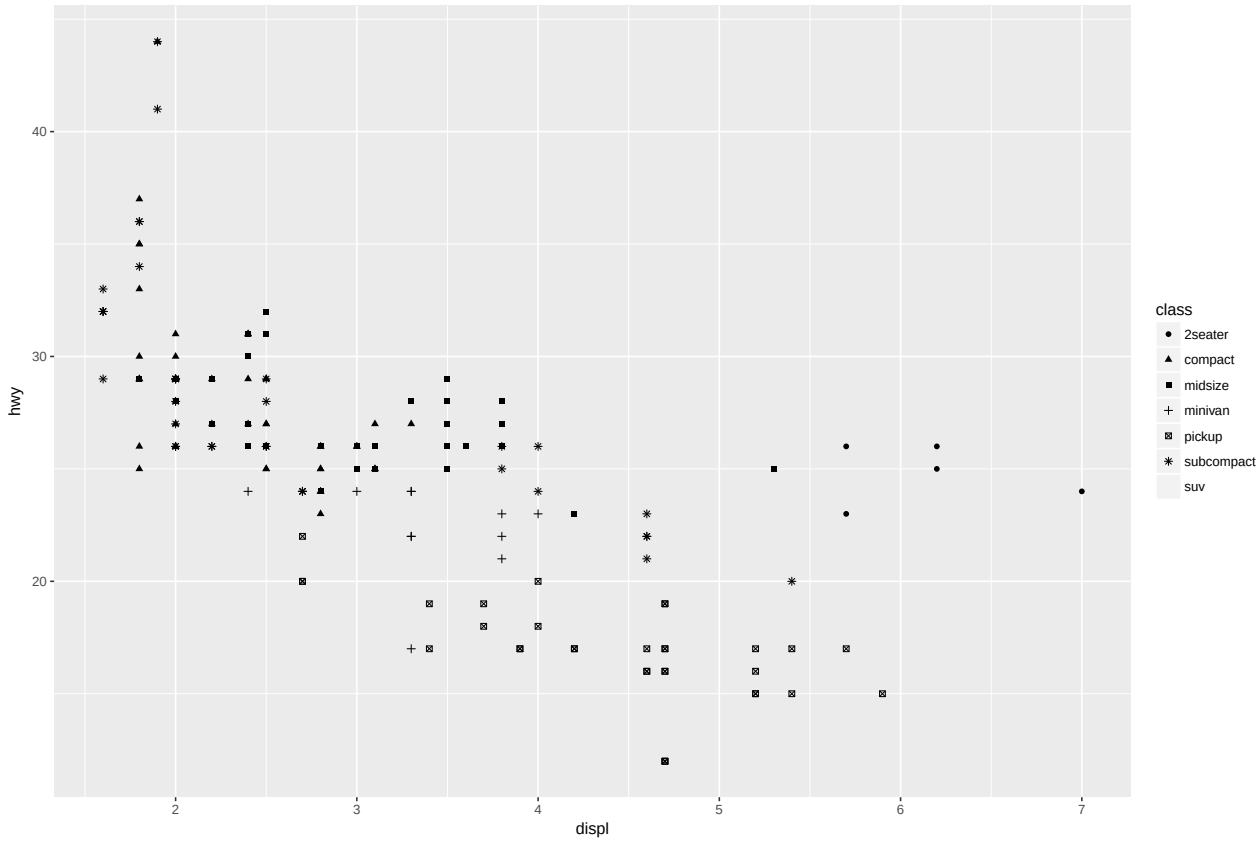

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, size = class))
```



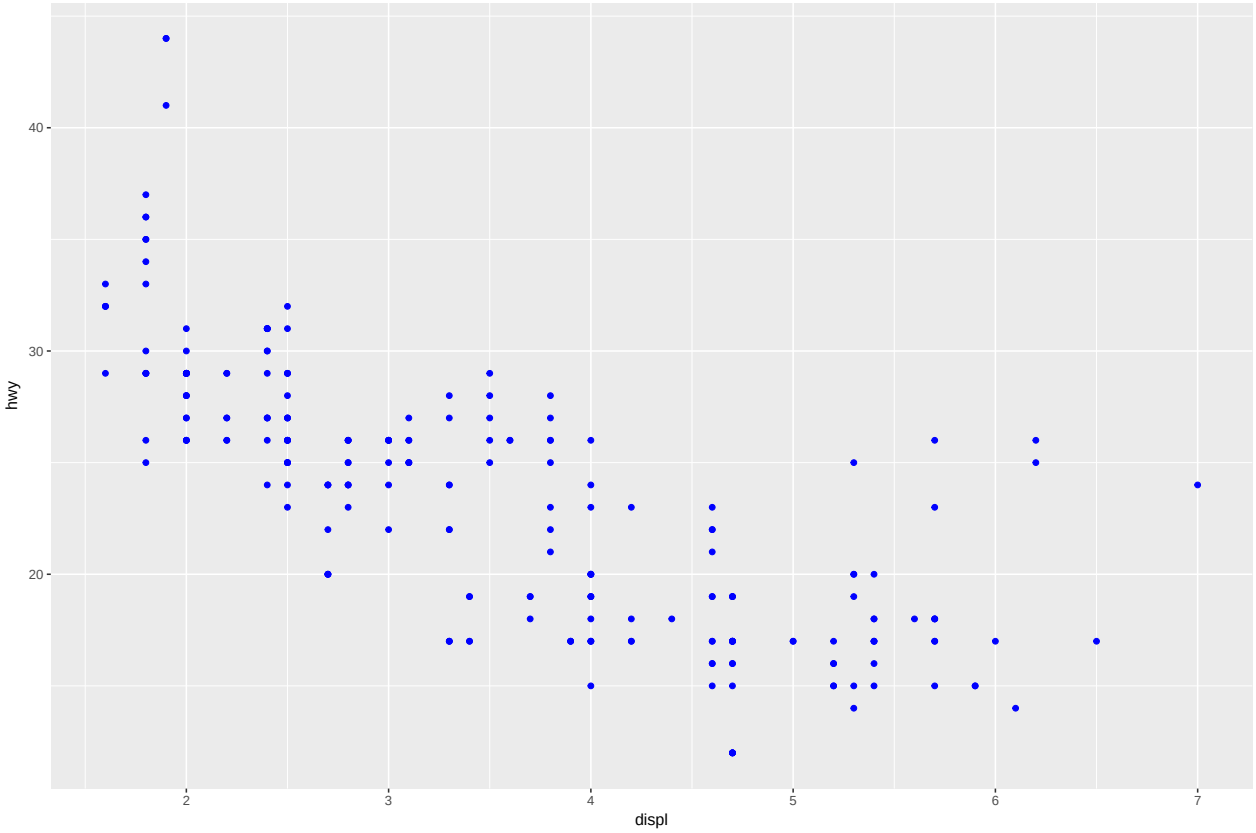

```
# Left
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, alpha = class))
```




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




```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy), color = "blue") # Set the aesthetic outside of aes() to
```



Aesthetic shapes:

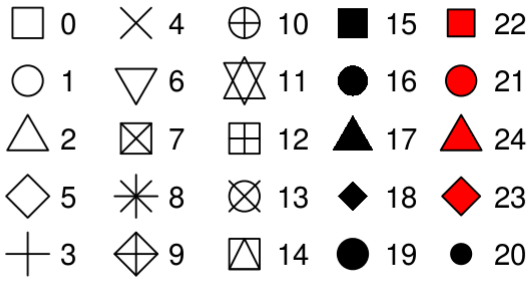


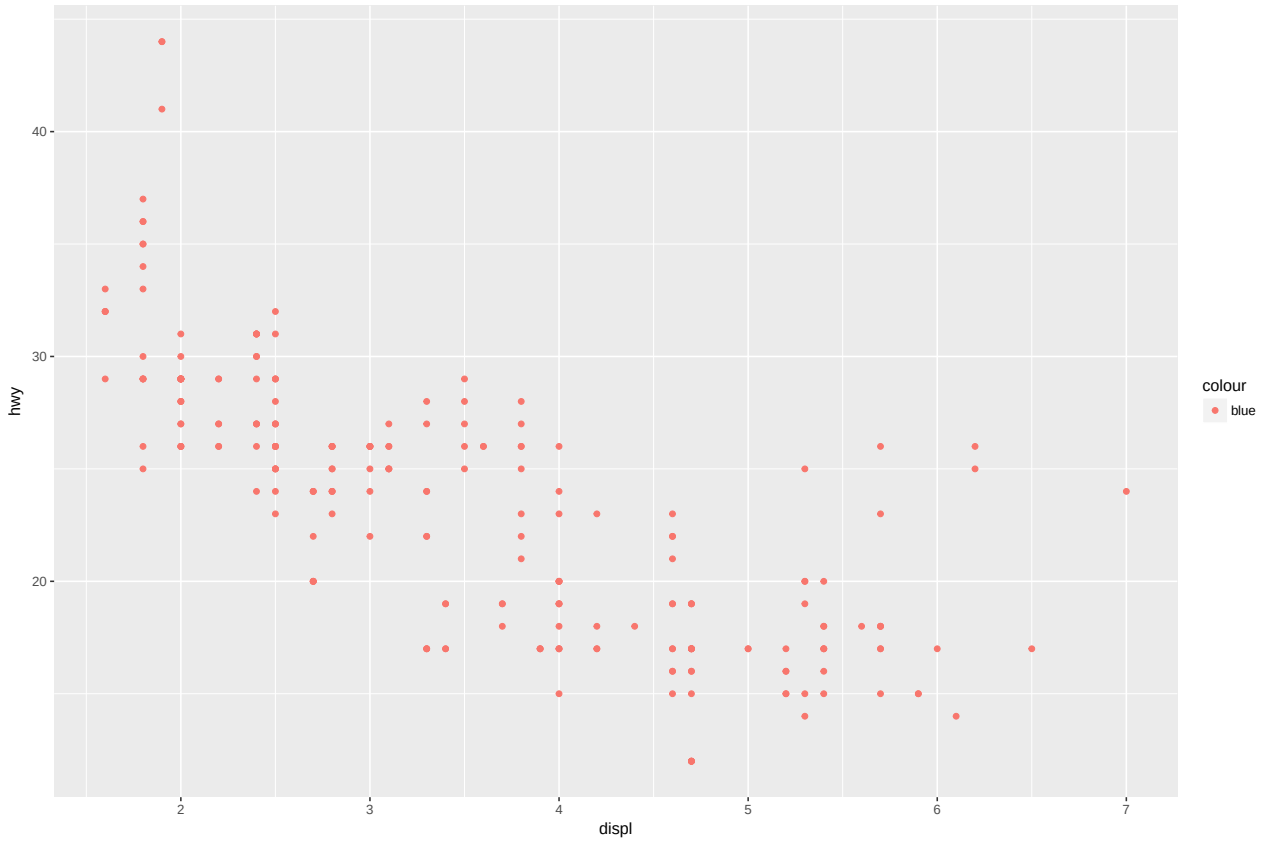
Figure 1:

Exercises 3.3.1

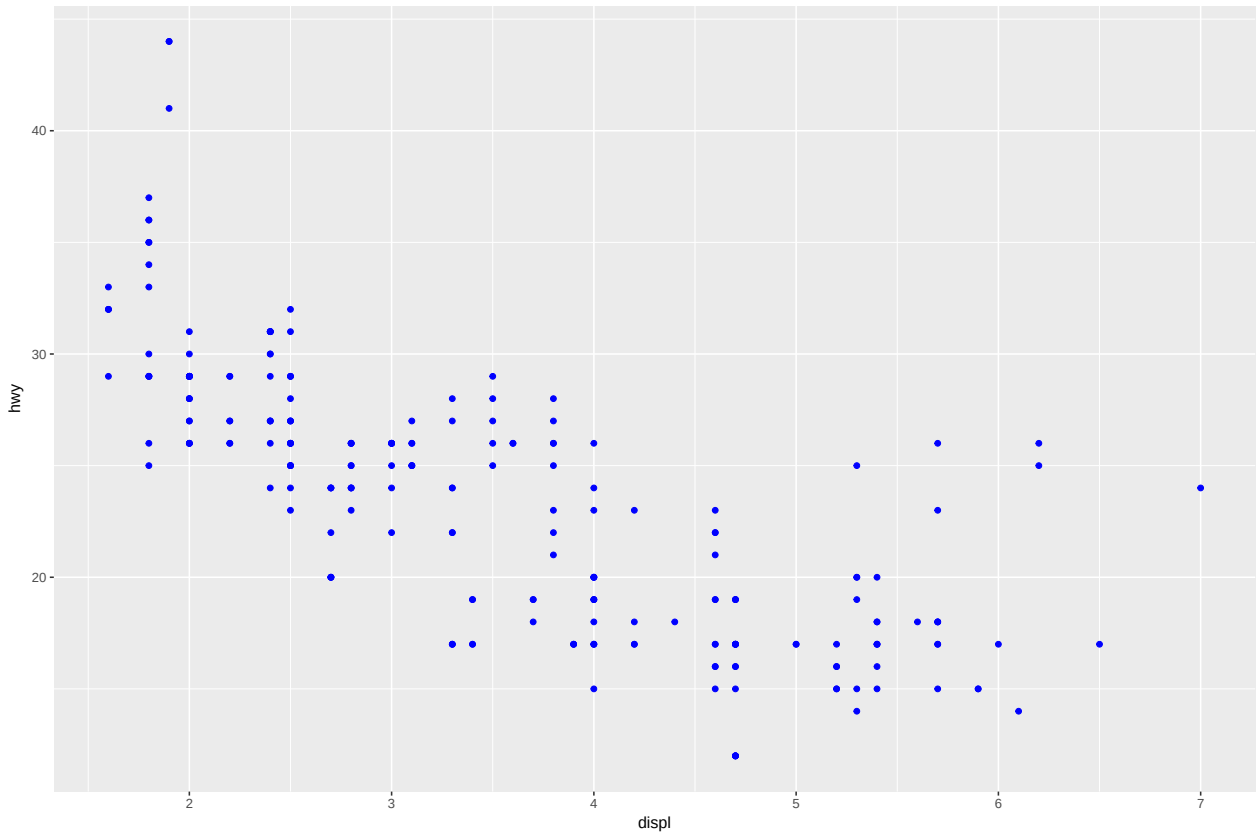
1. What's gone wrong with this code? Why are the points not blue?

The points are not blue because the color aesthetic is set inside `aes()`.

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




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



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

To determine what the categorical and continuous variables are, one can either view the tibble by typing `mpg` or by viewing the documentation `?mpg`. One may decide whether a variable is categorical or continuous by checking whether it is stored as a character, integer, or double (floating point integer) value. However, this can lead to miscategorization in some cases. For example, while year is an integer, it is typically considered a whole number, a discrete variable without a meaningful 0 value anchor, and therefore not continuous.

The categorical variables are:

- model
- year (discrete, rather than categorical)
- trans
- drv
- fl
- class

The continuous variables are:

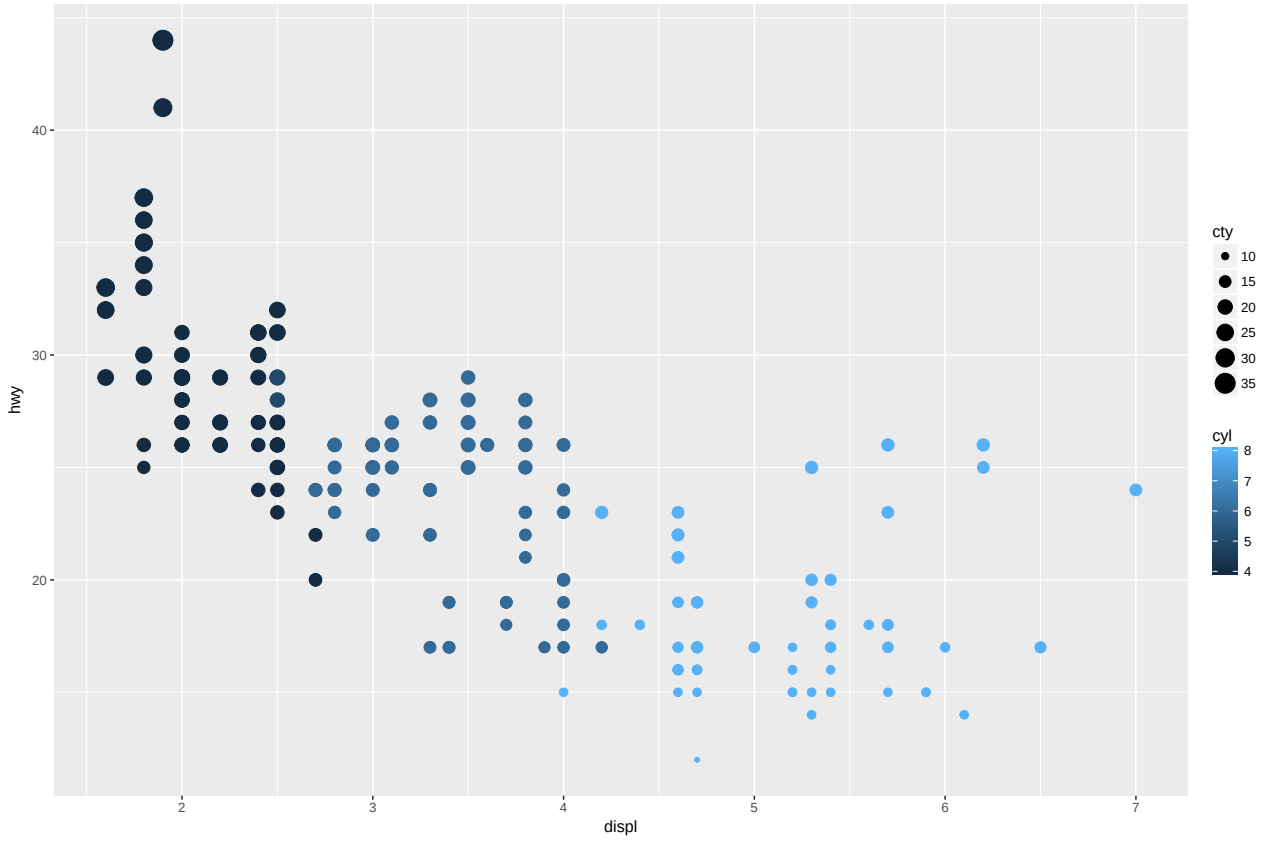
- displ
- cyl
- cty
- hwy
- year (in this data set, year is treated as an integer variable. Better to consider this "quantitative", rather than "continuous")

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

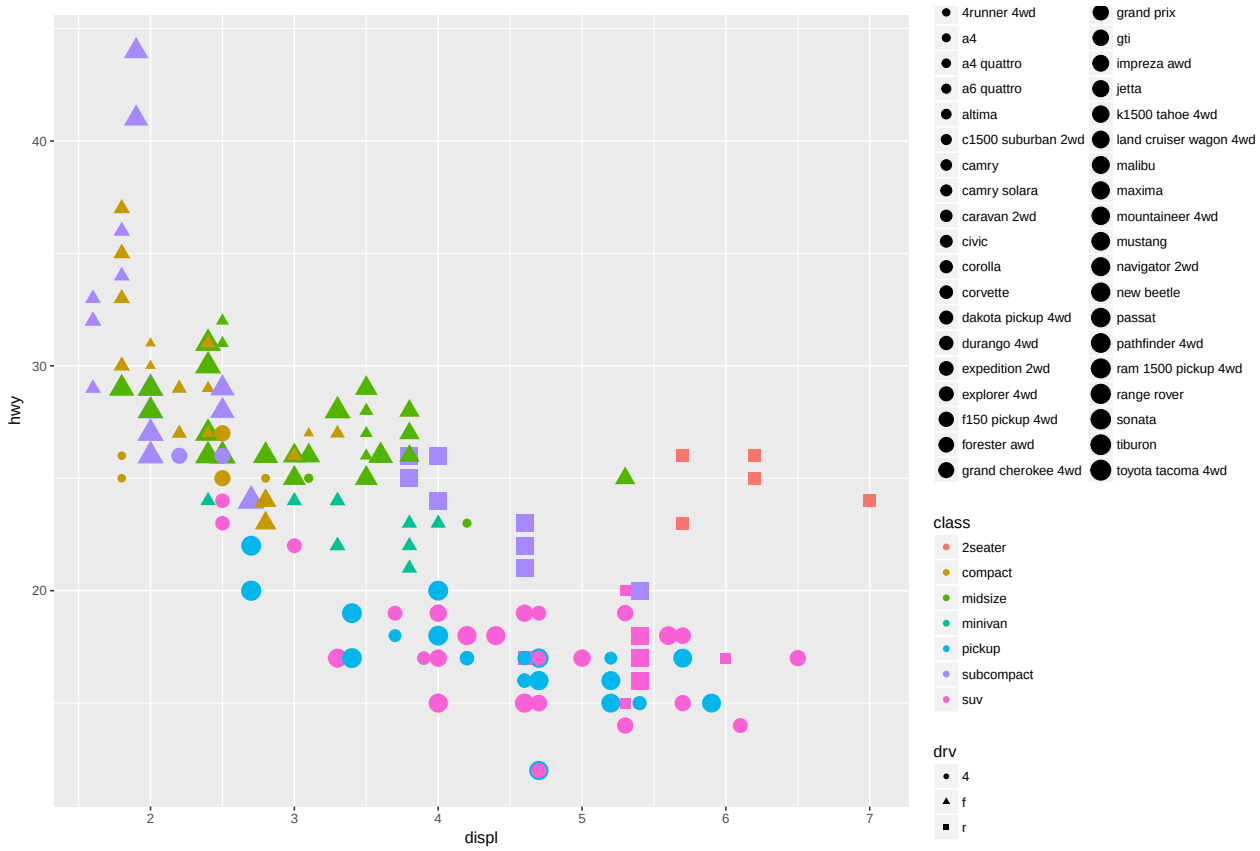
A continuous variable cannot be mapped to shape. When mapped to size or color, the continuous variable is binned by equal intervals (in this case, intervals of 5 mpg). When mapped to the size aesthetic, points scale by the intervals. Continuous variables when mapped to a color aesthetic are mapped along a gradient scale.

```
# Mapping a continuous variable to the shape aesthetic
ggplot(data=mpg) +
    geom_point(mapping = aes(x = displ, y = hwy, shape = cty))
```

```
# Mapping continuous variables to the color and size aesthetics
ggplot(data=mpg) +
    geom_point(mapping = aes(x = displ, y = hwy, color = cyl, size = cty))
```



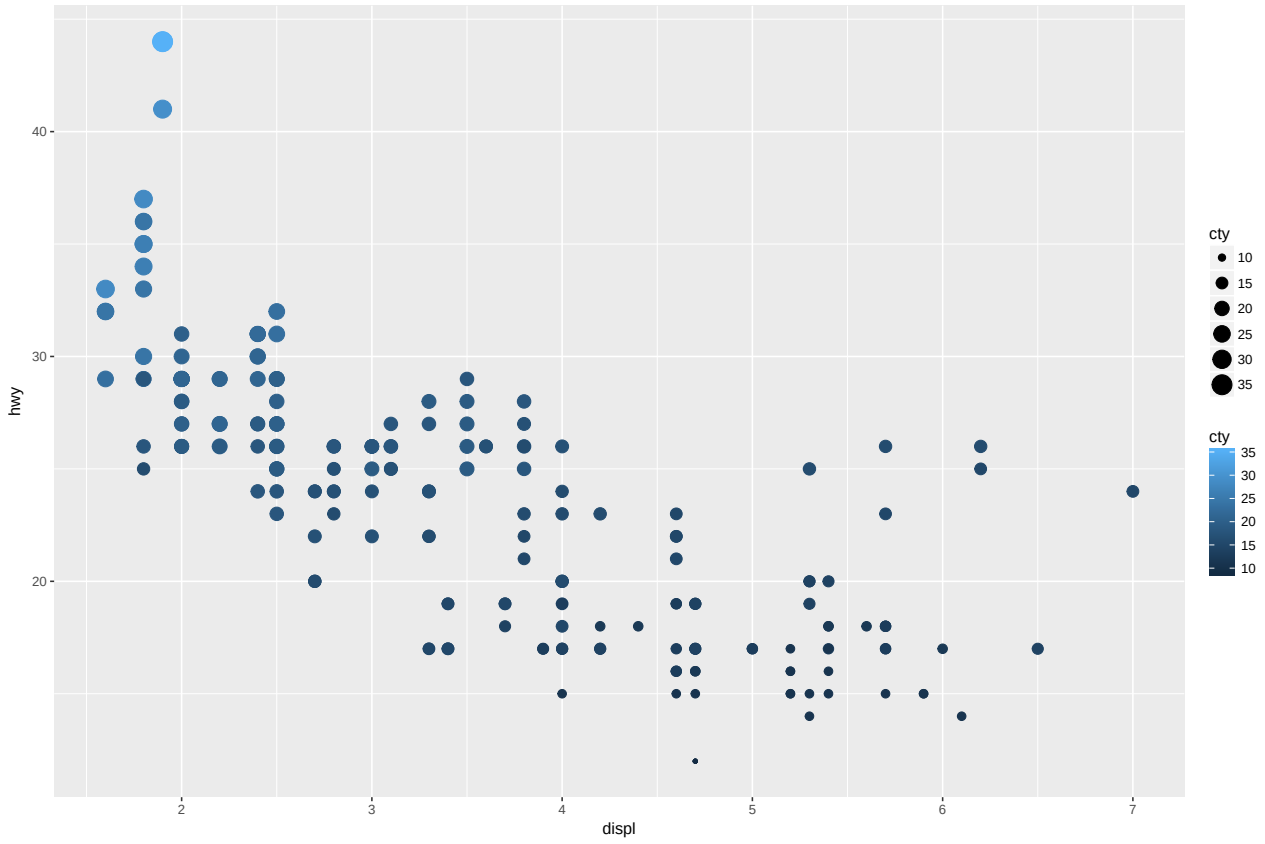

```
# Mapping categorical variables to size, color, and shape
ggplot(data=mpg) +
    geom_point(mapping = aes(x = displ, y = hwy, size = model, color = class, shape = drv))
```



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

When the same variable is mapped to multiple aesthetics, it is represented by those aesthetics.

```
# Mapping the same variable to multiple aesthetics
ggplot(data=mpg) +
    geom_point(mapping = aes(x = displ, y = hwy, color = cty, size = cty)) # Here, city is mapped to th
```

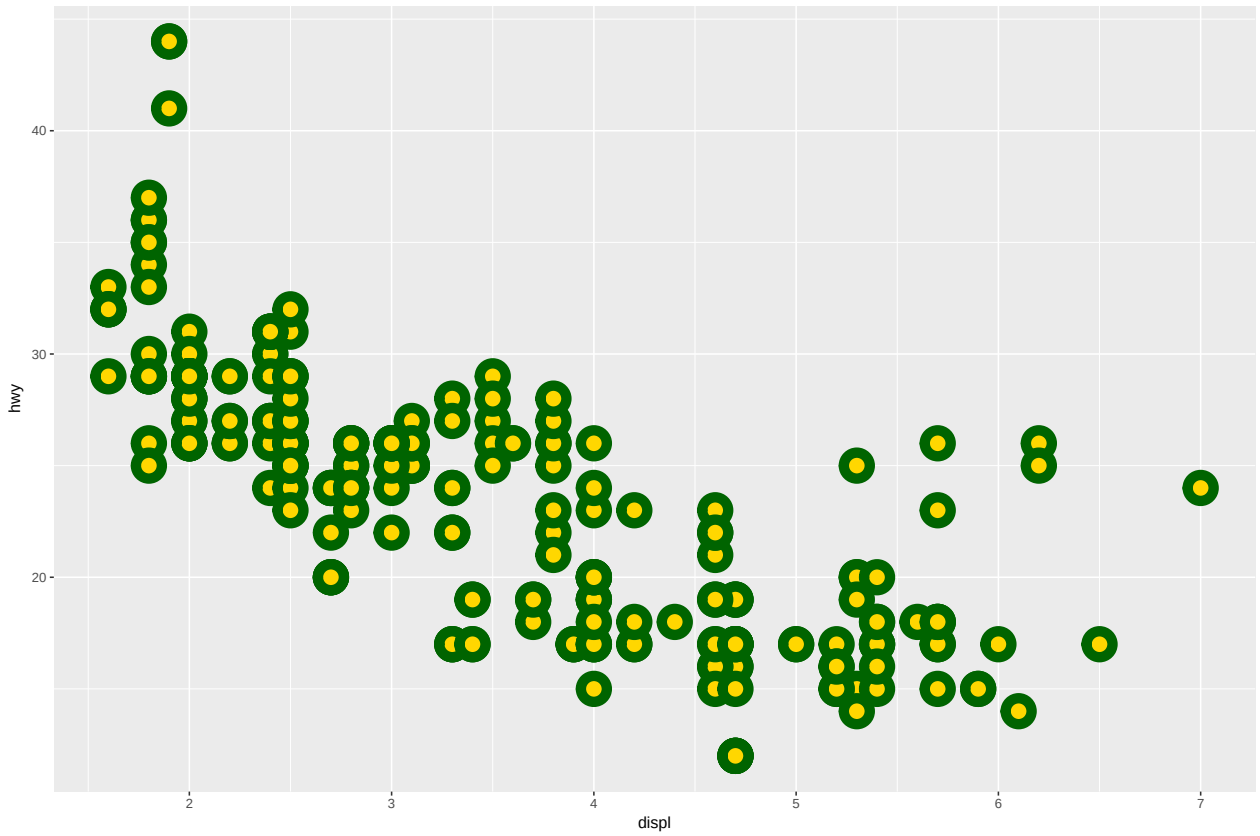


5. What does the stroke aesthetic do? What shapes does it work with? (Hint: use ?geom_point)

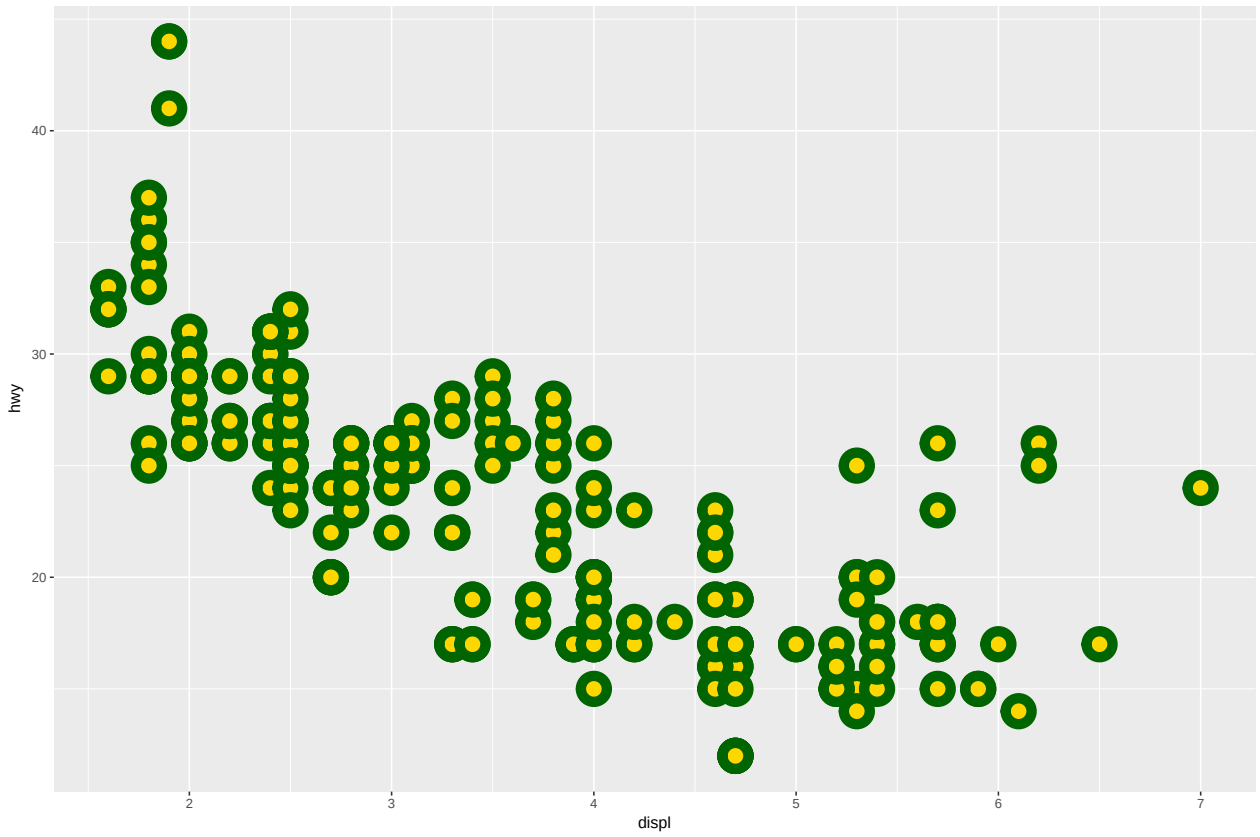
According to the R documentation:
"For shapes that have a border (like 21), you can colour the inside and outside separately. Use the **stroke** aesthetic to modify the width of the border."

Tip: You can find documentation of available colors here.

```
?geom_point

# Example using `stroke`
ggplot(data=mpg)+
    geom_point(mapping = aes(x=displ, y=hwy), shape = 21, colour = "darkgreen", fill = "gold", size = 5
```



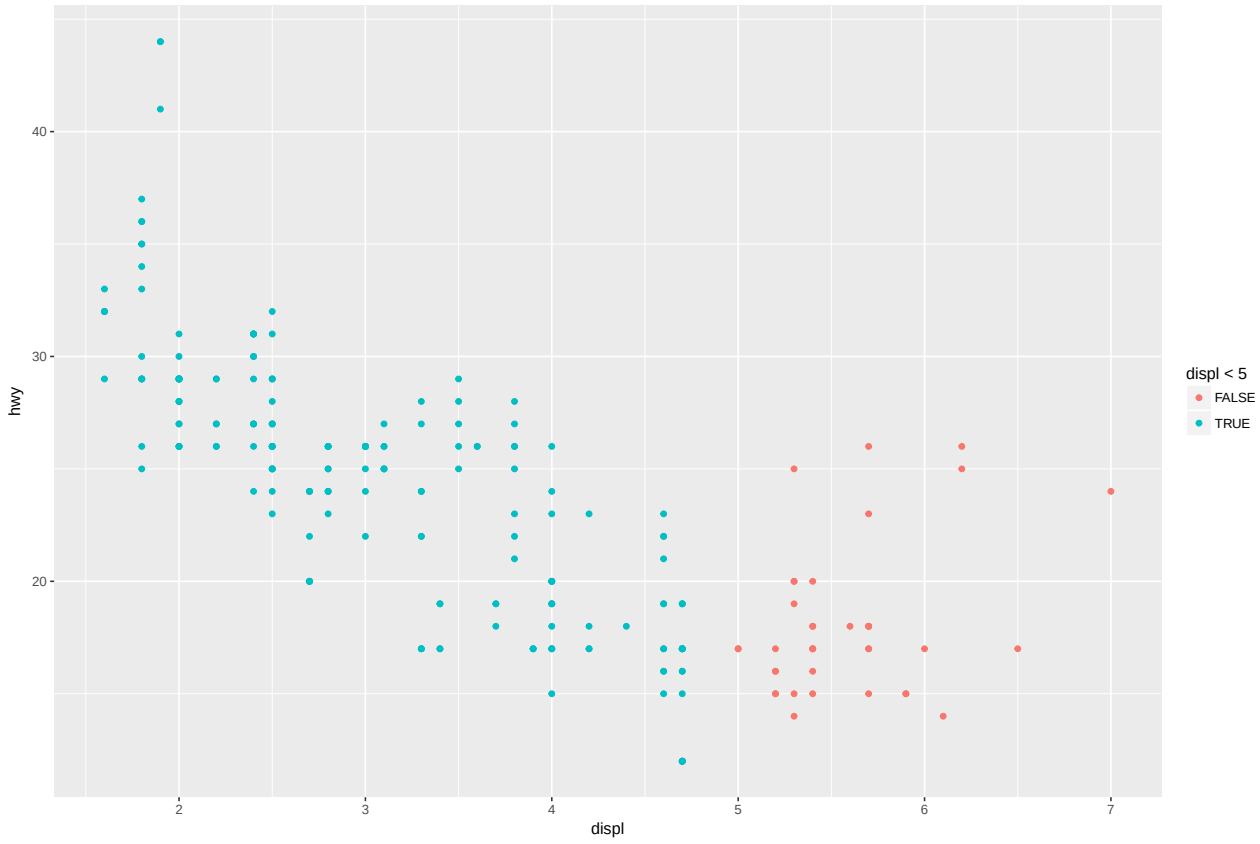

```
# Just for fun, let's write short-hand code make the same plot
ggplot(mpg, aes(displ, hwy)) +
  geom_point(shape = 21, colour = "darkgreen", fill = "gold", size = 5, stroke = 5)
```



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

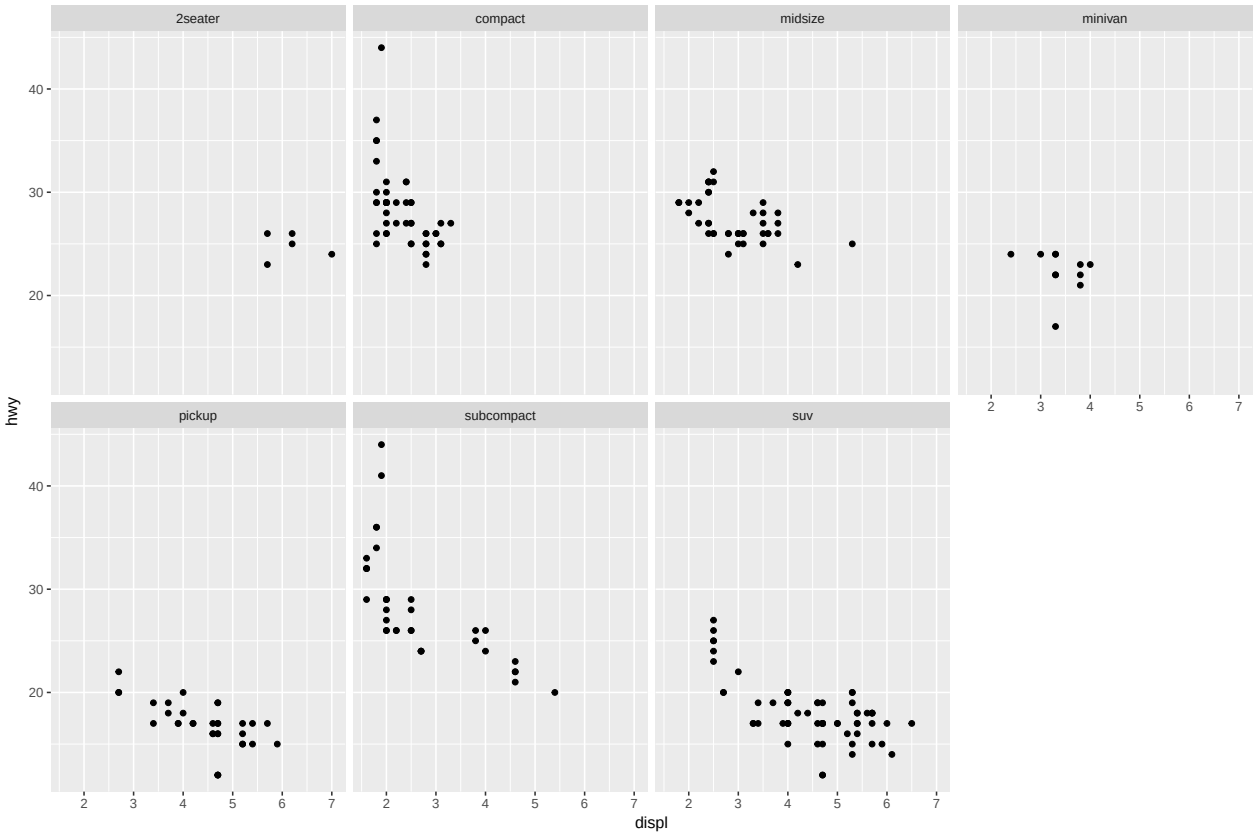
Setting the color aesthetic to `displ < 5` will assign one color to all x-axis (hwy) values < 5 and a different color to x-axis values ≥ 5. Since the color palette is not specified, default colors are used.

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

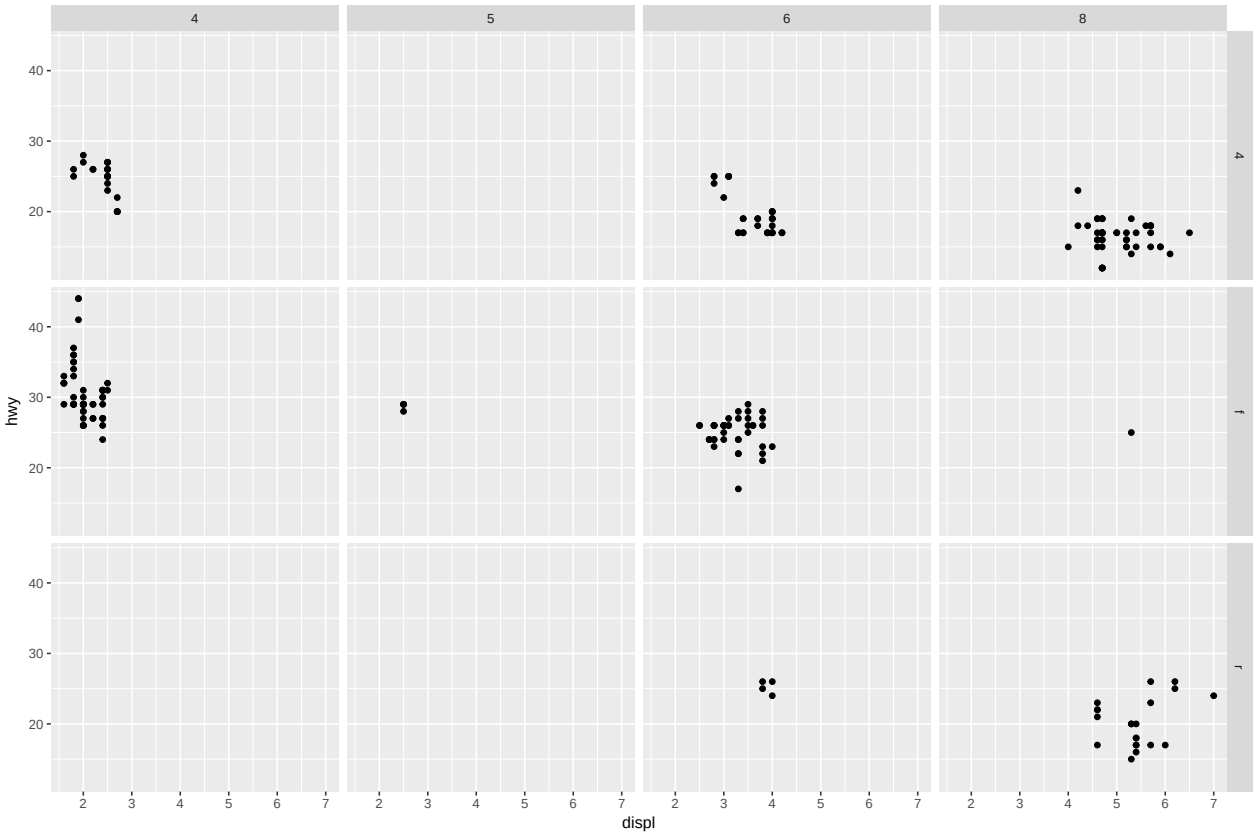


Facets

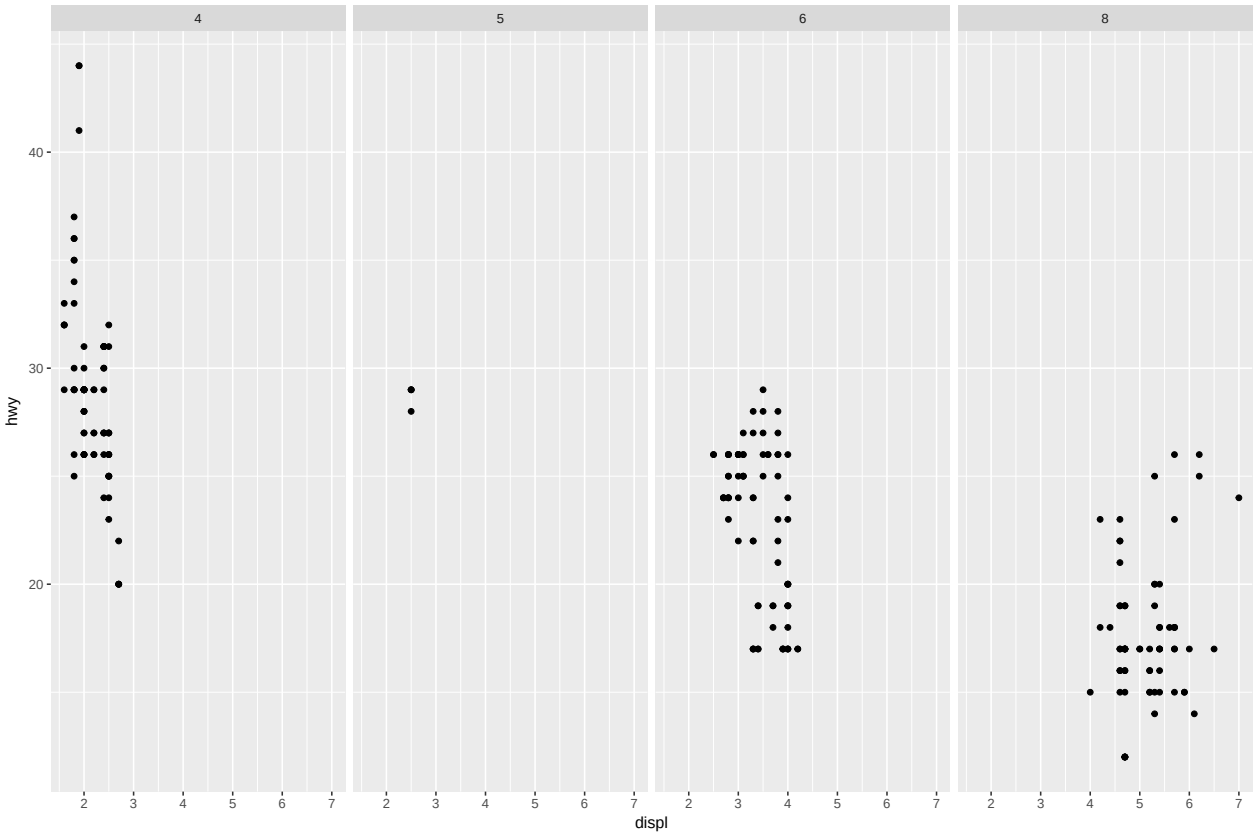
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_wrap(~ class, nrow = 2) # This will create a separate plot for each class of vehicle and will f
```




```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_grid(drv ~ cyl) # This will create a grid of plots with one plot for each combination of drv an
```




```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
    facet_grid(. ~ cyl) # Use the . to create plots for each level of cylinder (cyl) in the columns dim
```

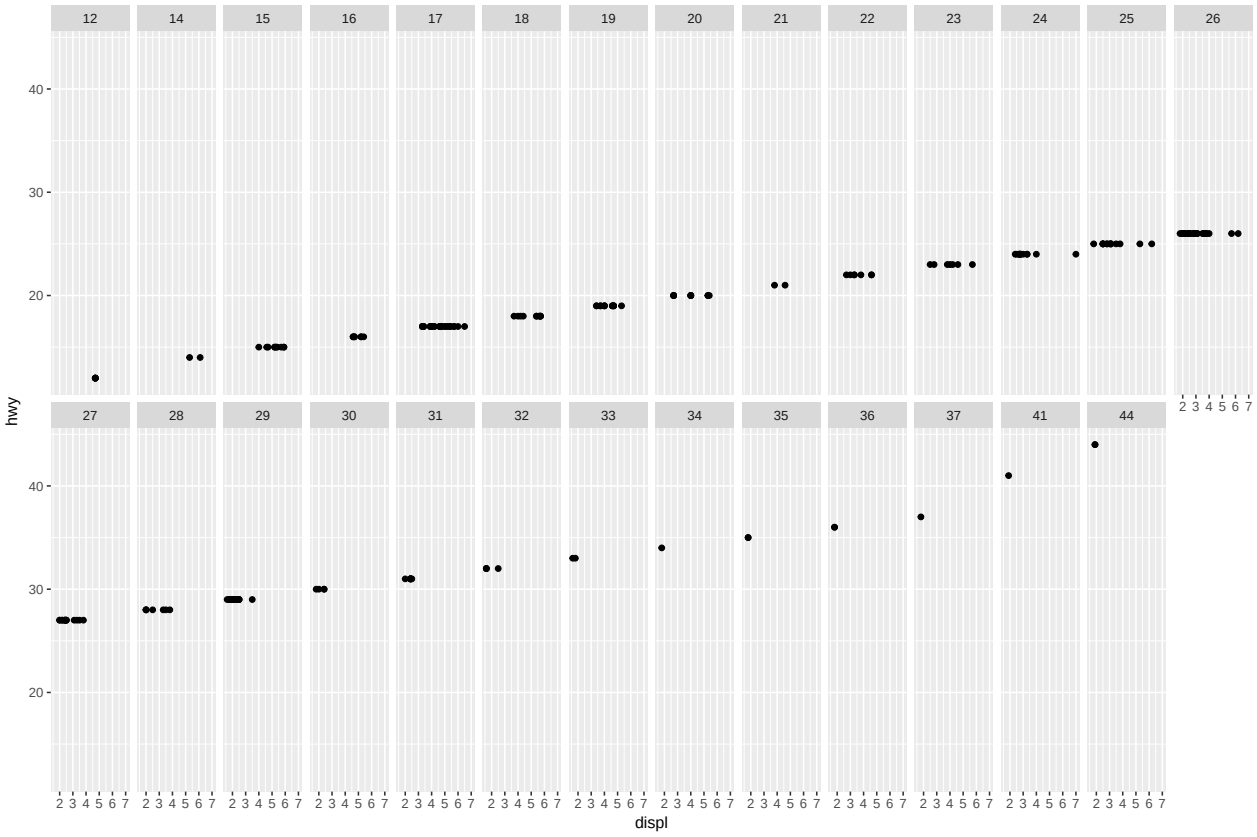


Exercises 3.5.1

1. What happens if you facet on a continuous variable?

If faceting is done with a continuous variable, a plot is created for each value for which there is at least one observation.

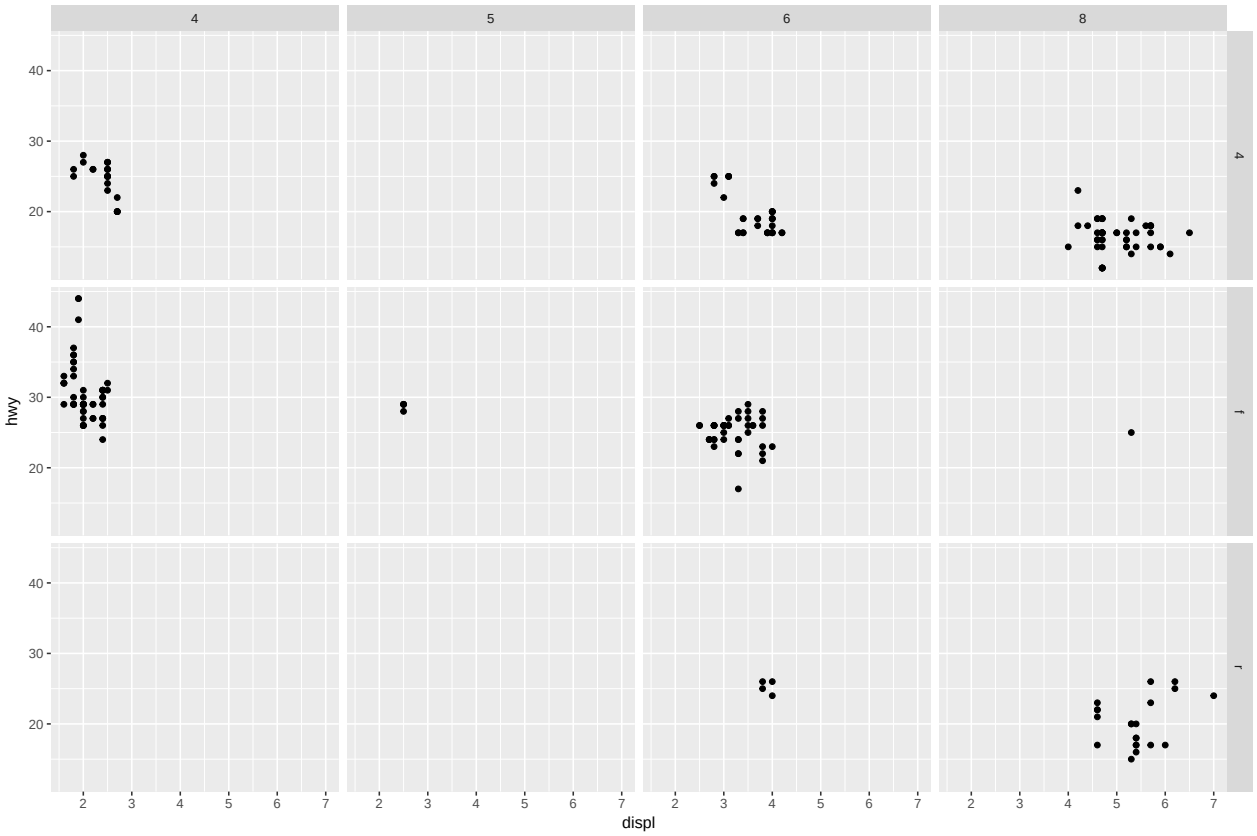
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_wrap(~ hwy, nrow = 2)
```



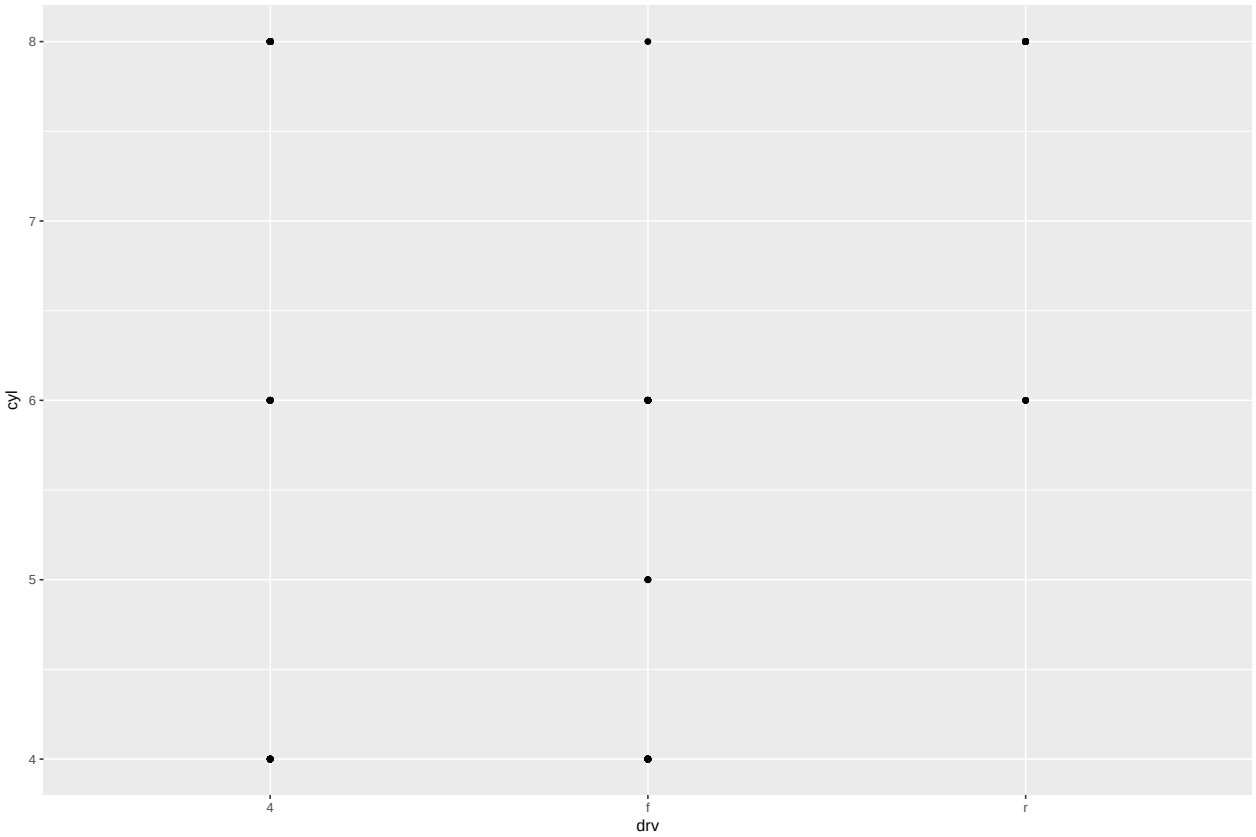
2. What do the empty cells in plot with facet_grid(drv ~ cyl) mean? How do they relate to this plot?

The empty cells in the plot with `facet_grid(drv ~ cyl)` indicate that there are no cars with at the intersection of that number of cylinders and that type of drivetrain (e.g. no cars with 5 cylinders and 4-wheel drive). The absence of vehicles corresponding to specific cylinde r-drive combinations is also evident in the second plot. Those intersections in the second plot without a point correspond to the empty cells in the first plot (see again cars with 5 cylinders on the y-axis and 4-wheel drive on the x-axis).

```
# First plot, with drivetrain and cylinder faceted
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_grid(drv ~ cyl)
```



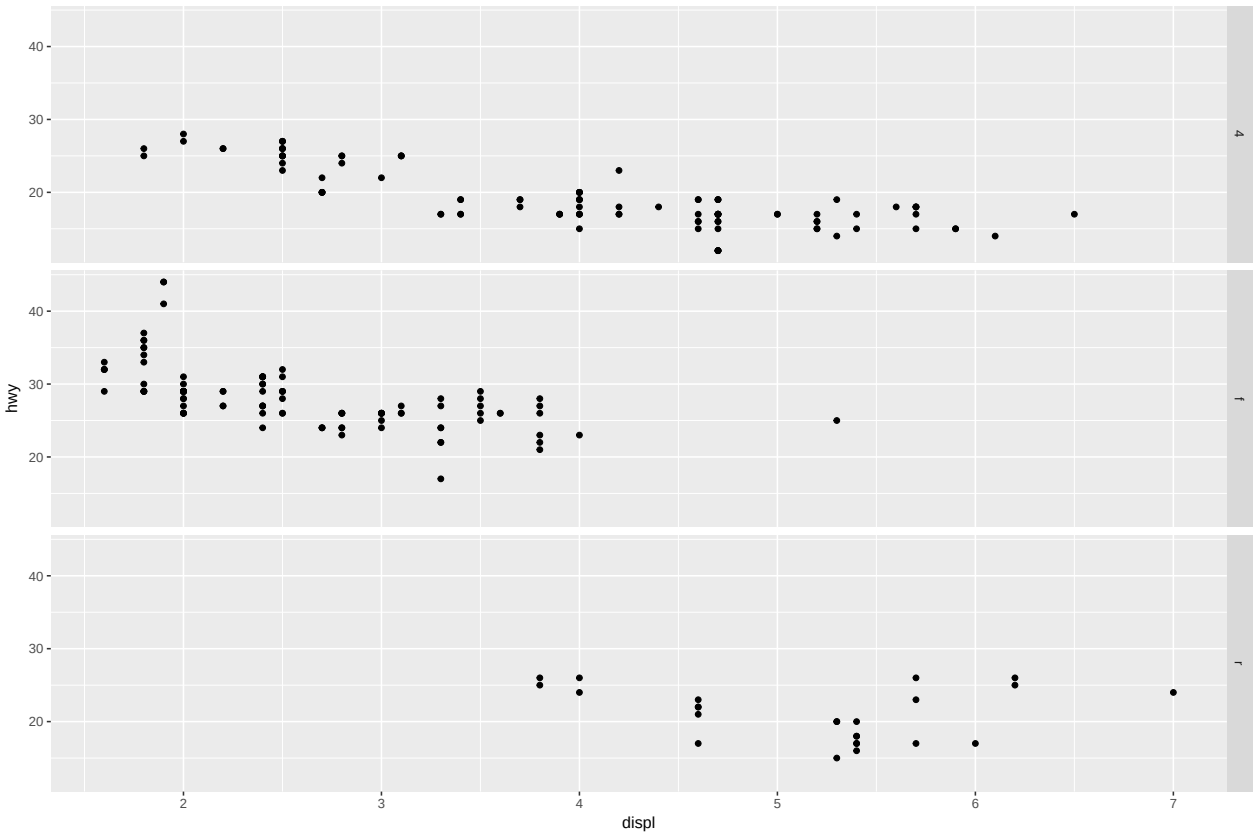

```
# Second plot, with drivetrain and cylinder represented on the axes of a single plot
ggplot(data = mpg) +
  geom_point(mapping = aes(x = drv, y = cyl))
```



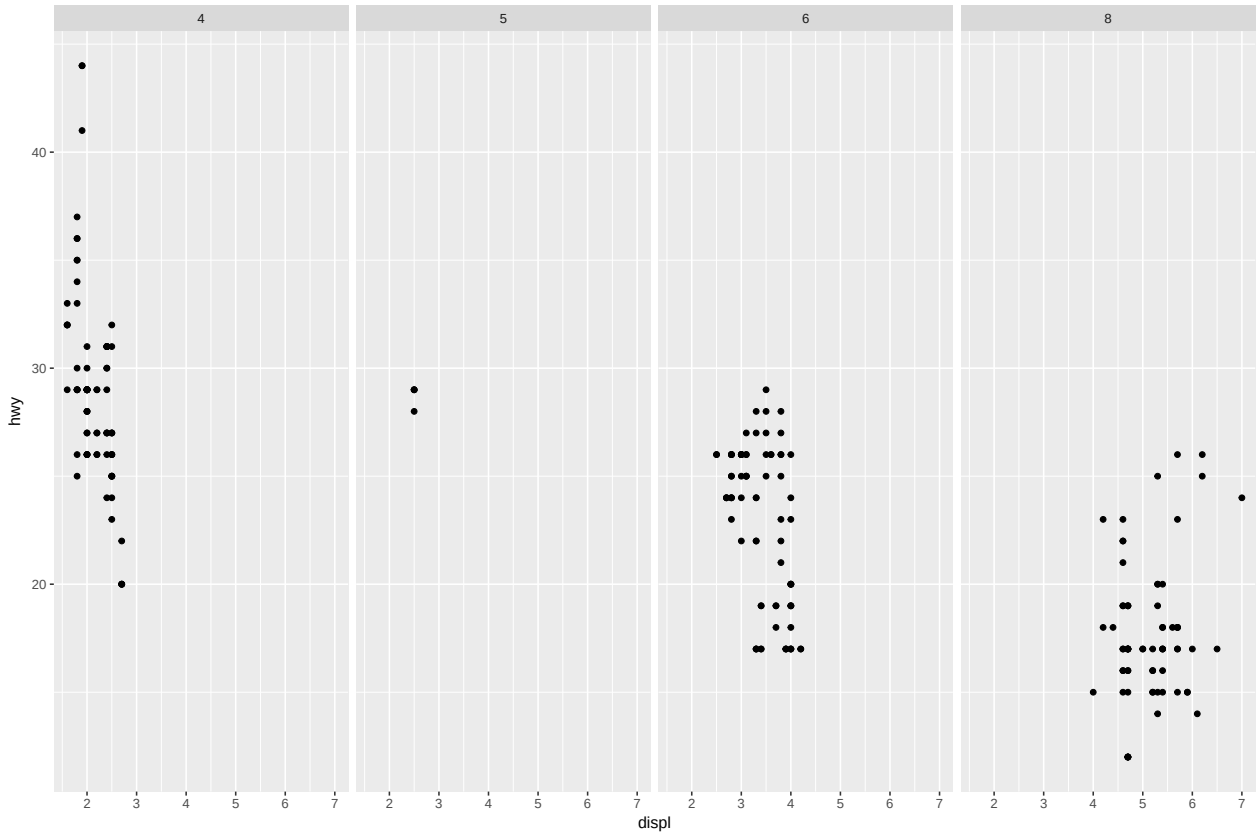
3. What plots does the following code make? What does . do?

The first plot shows highway miles per gallon and engine displacement faceted by drivetrain type. The . in the second position specifies that drivetrain type should be displayed in rows. The second plot shows highway miles per gallon and engine displacement faceted by number of cylinders. The . in the first position specifies that number of cylinders should be displayed in columns.

```
# Plot of highway mpg and engine displacement faceted by drivetrain type
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_grid(drv ~ .)
```




```
# The above is the same as the following except that the drivetrain labels shift from right to top alig
#ggplot(data = mpg) +
#  geom_point(mapping = aes(x = displ, y = hwy)) +
#  facet_wrap(~ drv, nrow = 3)

# Plot of highway mpg and engine displacement faceted by number of cylinders
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_grid(. ~ cyl)
```

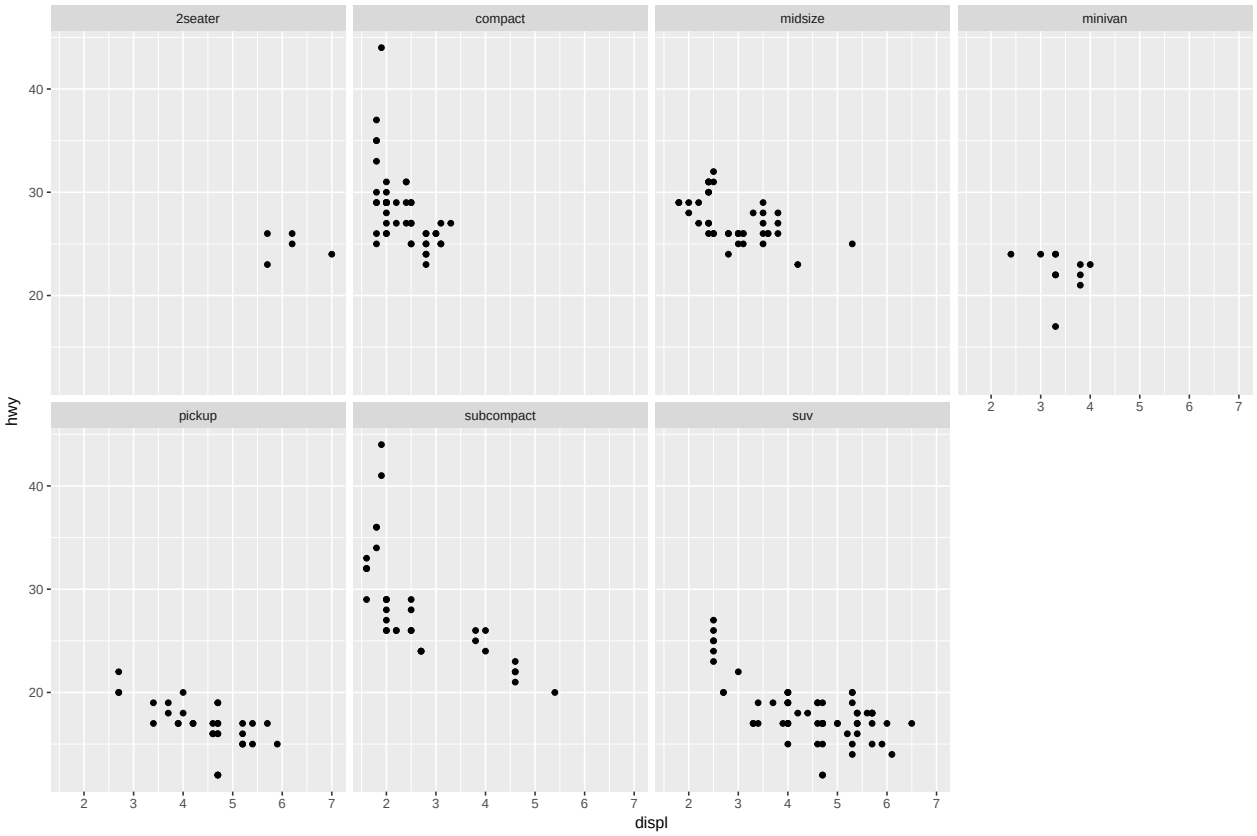



```
# The above is the same as the following. Uncomment and run the code to see.
#ggplot(data = mpg) +
#    geom_point(mapping = aes(x = displ, y = hwy)) +
#    facet_wrap(~ cyl, nrow = 1)
```

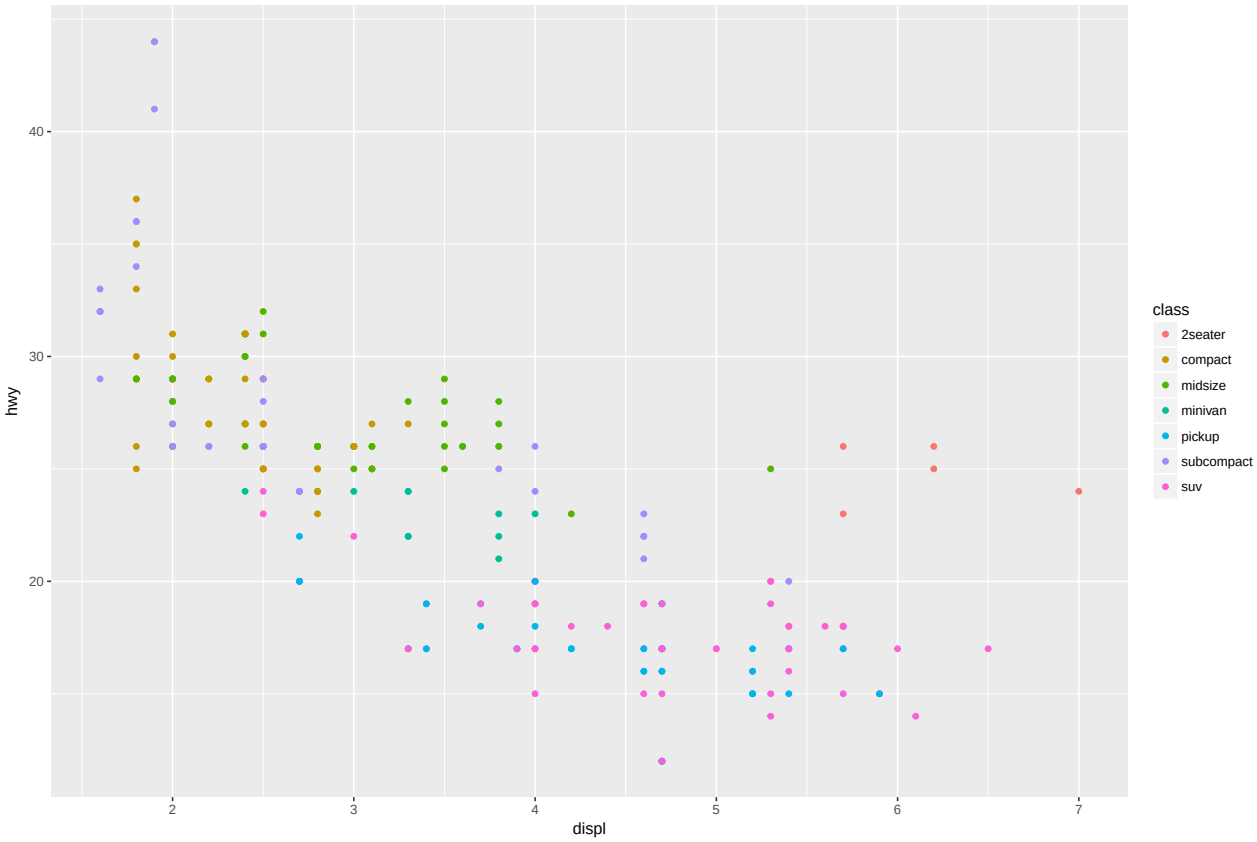
4. Take the first faceted plot in this section. What are the advantages to using faceting instead of the colour aesthetic? What are the disadvantages? How might the balance change if you had a larger dataset?

The advantage of using faceting rather than the color aesthetic is that with separate plots it is easier to see the shape and spread of the data points for each level of the variable. A disadvantage is that it's difficult to see the overall shape and spread of the observations across levels of the faceted variable. While using the color aesthetic works well with the mpg dataset, with a larger dataset, the likelihood of overlapping data points increases and with enough overlapping observations jittering may be insufficient. It may therefore be preferable to use faceting with large datasets.

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




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



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?

`nrow` - specifies the number of rows into which the faceted plots are fitted.
`ncol` - specifies the number of columns into which the faceted plots are fitted.
`facet_grid()` does not have `nrow` or `ncol` arguments because the number of rows and columns is determined by the number of levels of the row and column facetting variables.

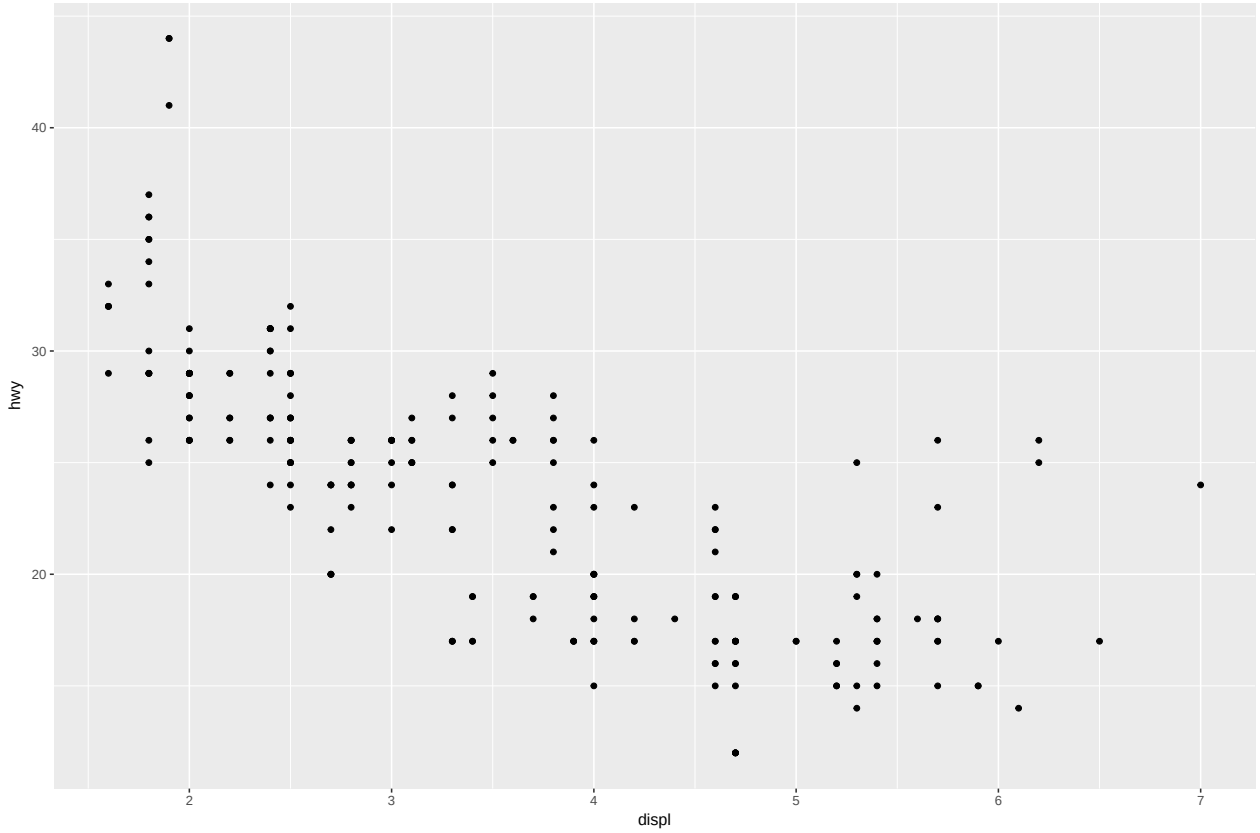
```
?facet_wrap
```

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

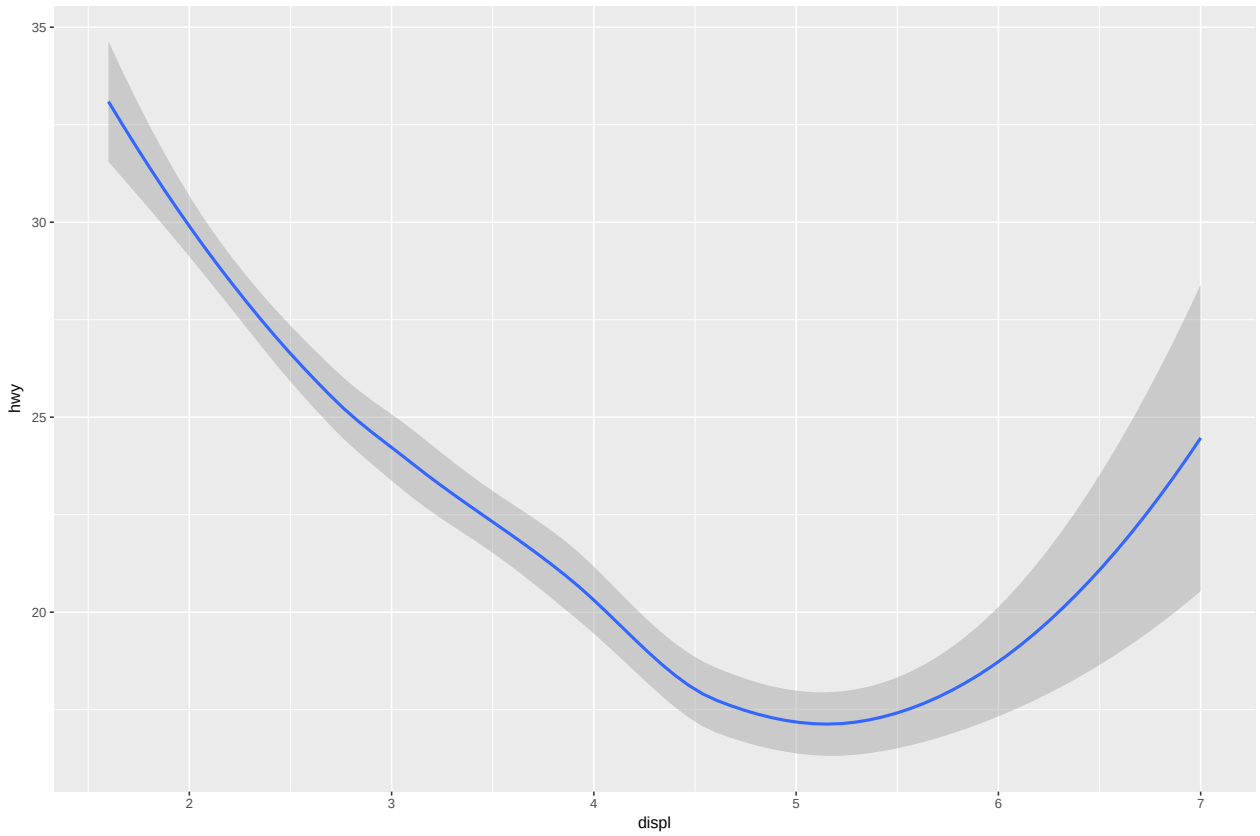
One should put the variable with more unique levels in the columns so the plots can extend vertically where there is more space. The horizontal space is limited by the page width and adding more plots compresses them, making them difficult to read.

Geometric objects

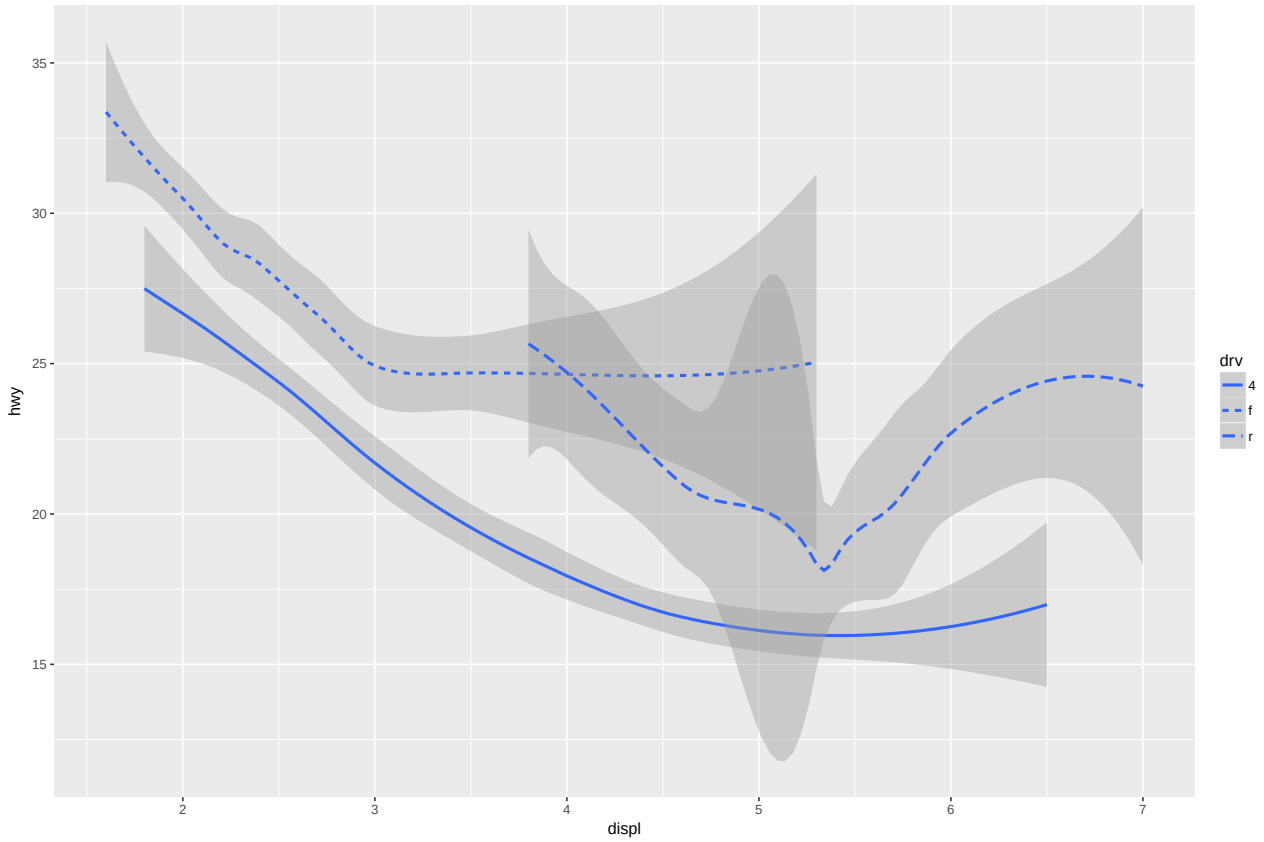
```
# Scatterplot
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```



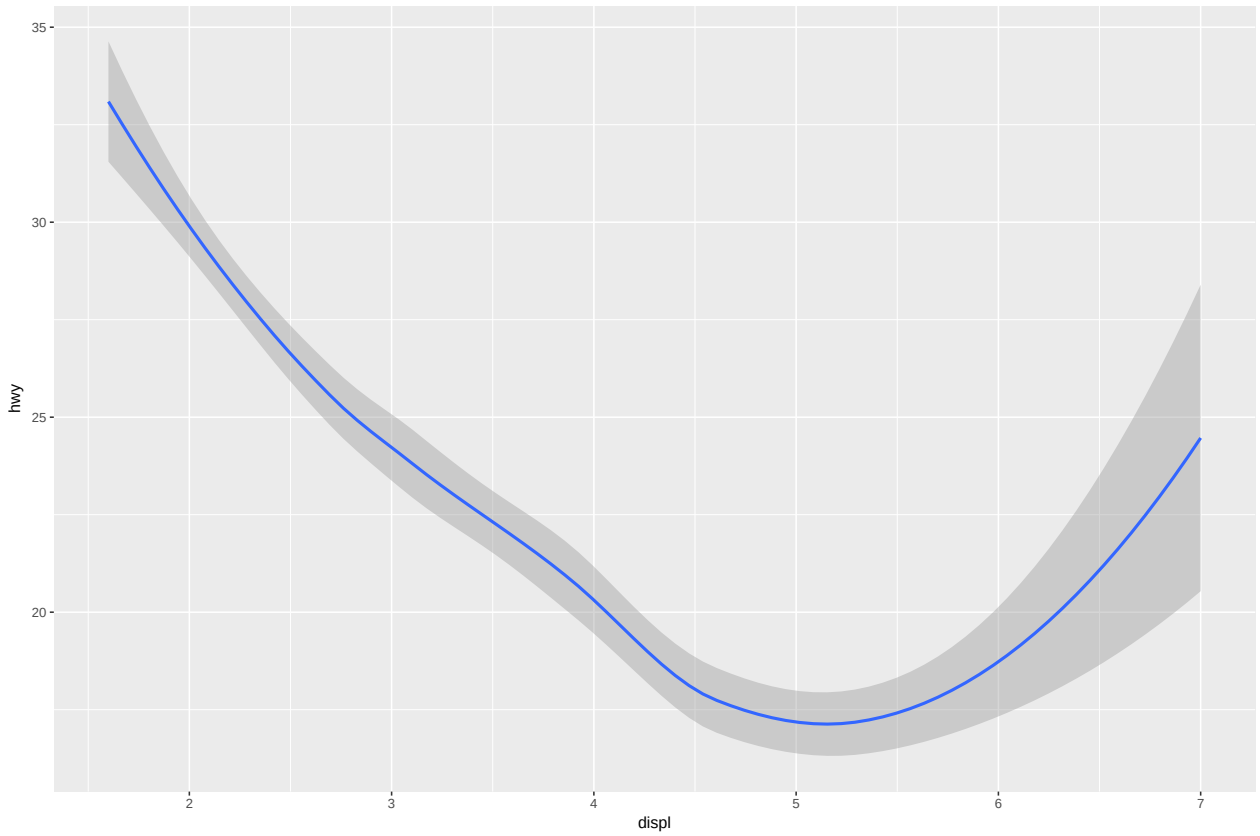

```
# Smooth geom plot
ggplot(data = mpg) +
  geom_smooth(mapping = aes(x = displ, y = hwy))
```



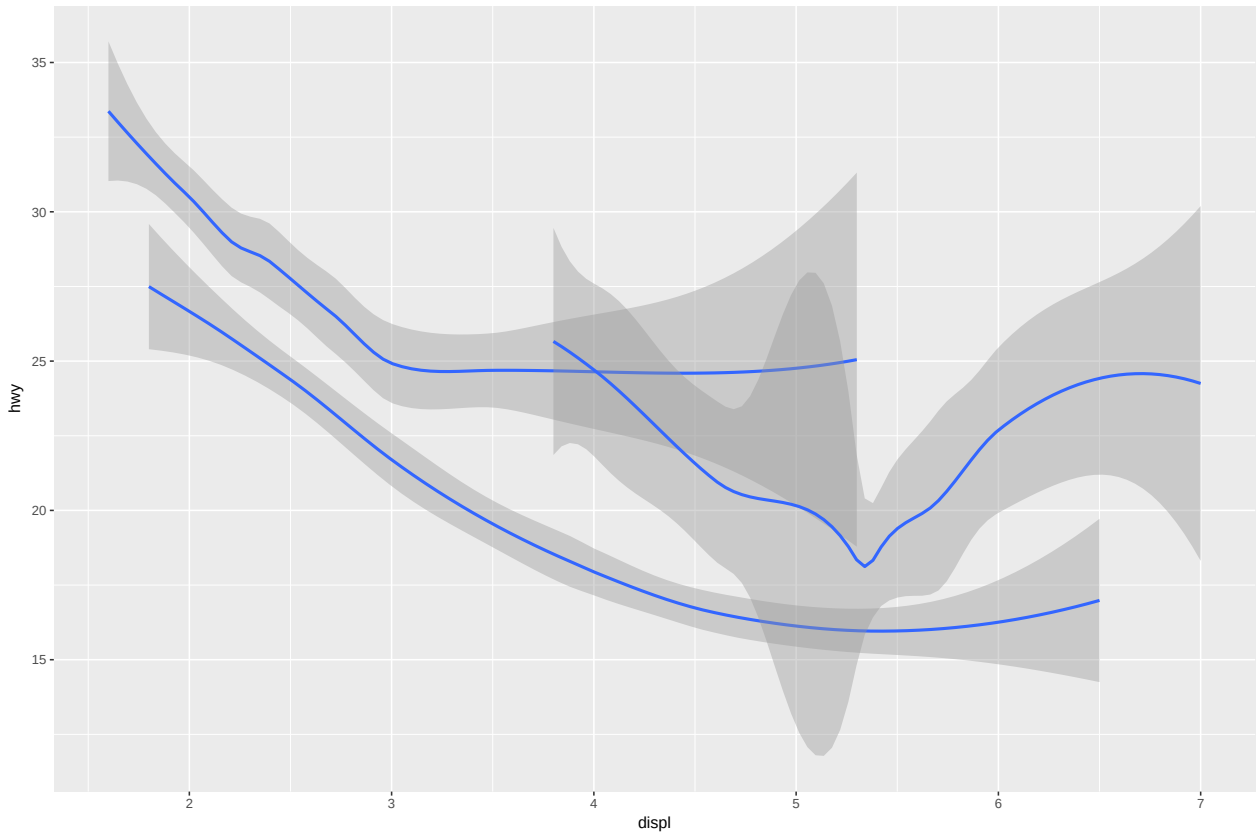

```
# Use a different linetype for each unique value of drv
ggplot(data = mpg) +
  geom_smooth(mapping = aes(x = displ, y = hwy, linetype = drv))
```



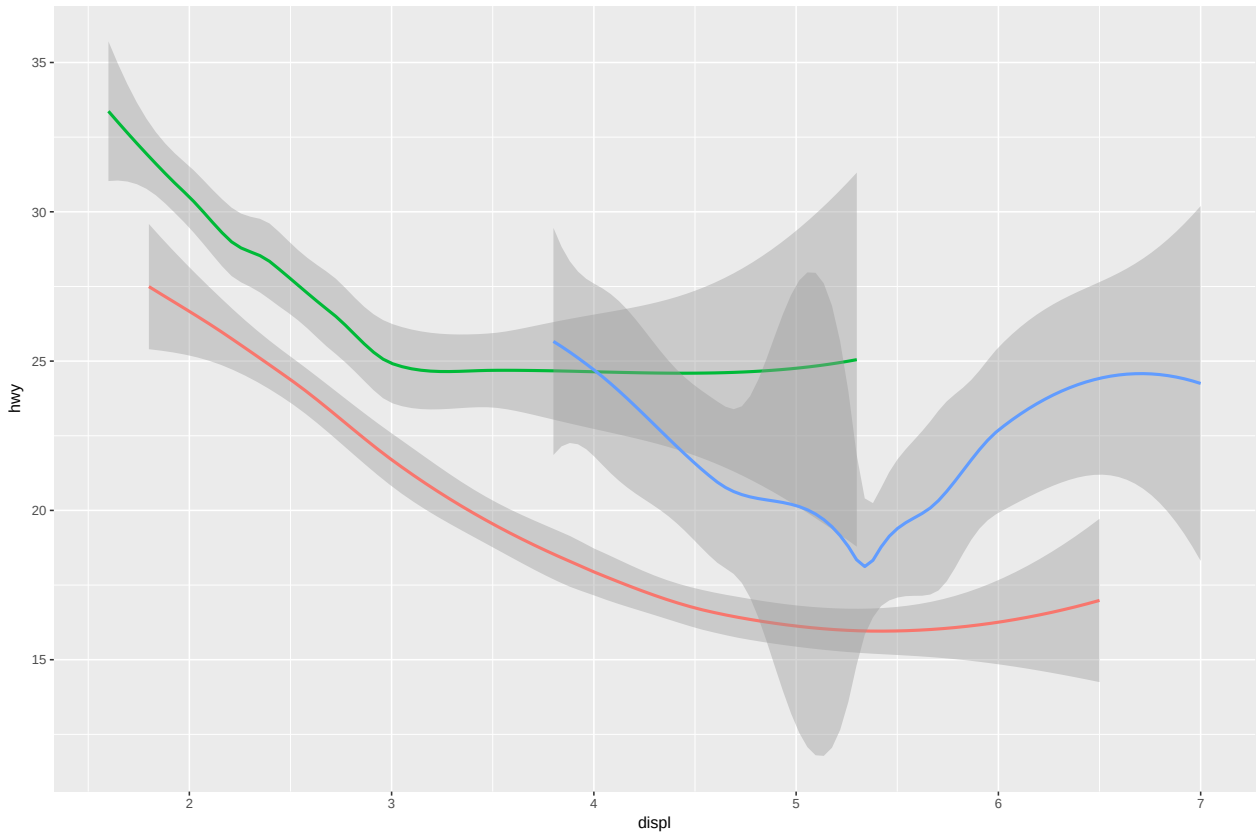

```
# Plot a single geom to display the data
ggplot(data = mpg) +
  geom_smooth(mapping = aes(x = displ, y = hwy))
```



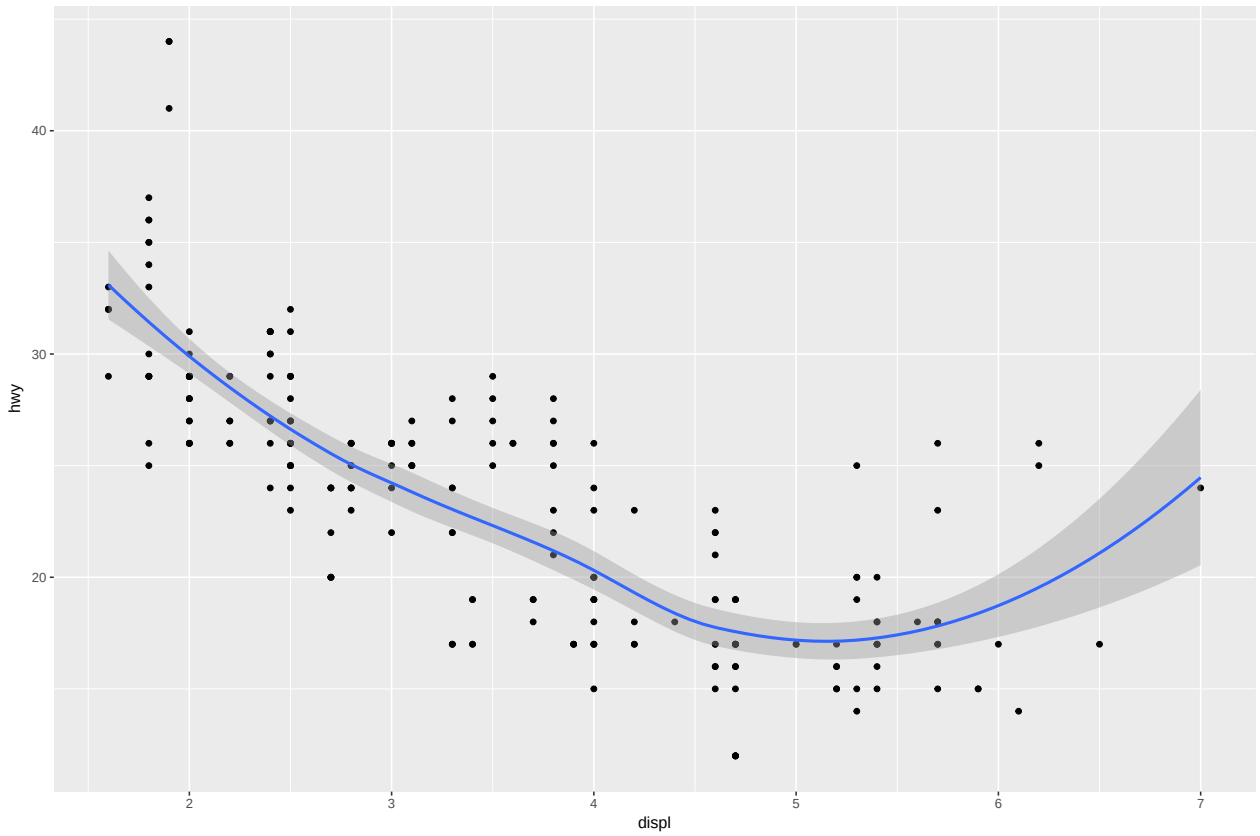

```
# Set the `group` aesthetic to `drv` to draw separate geoms for each unique value of the variable
ggplot(data = mpg) +
  geom_smooth(mapping = aes(x = displ, y = hwy, group = drv))
```



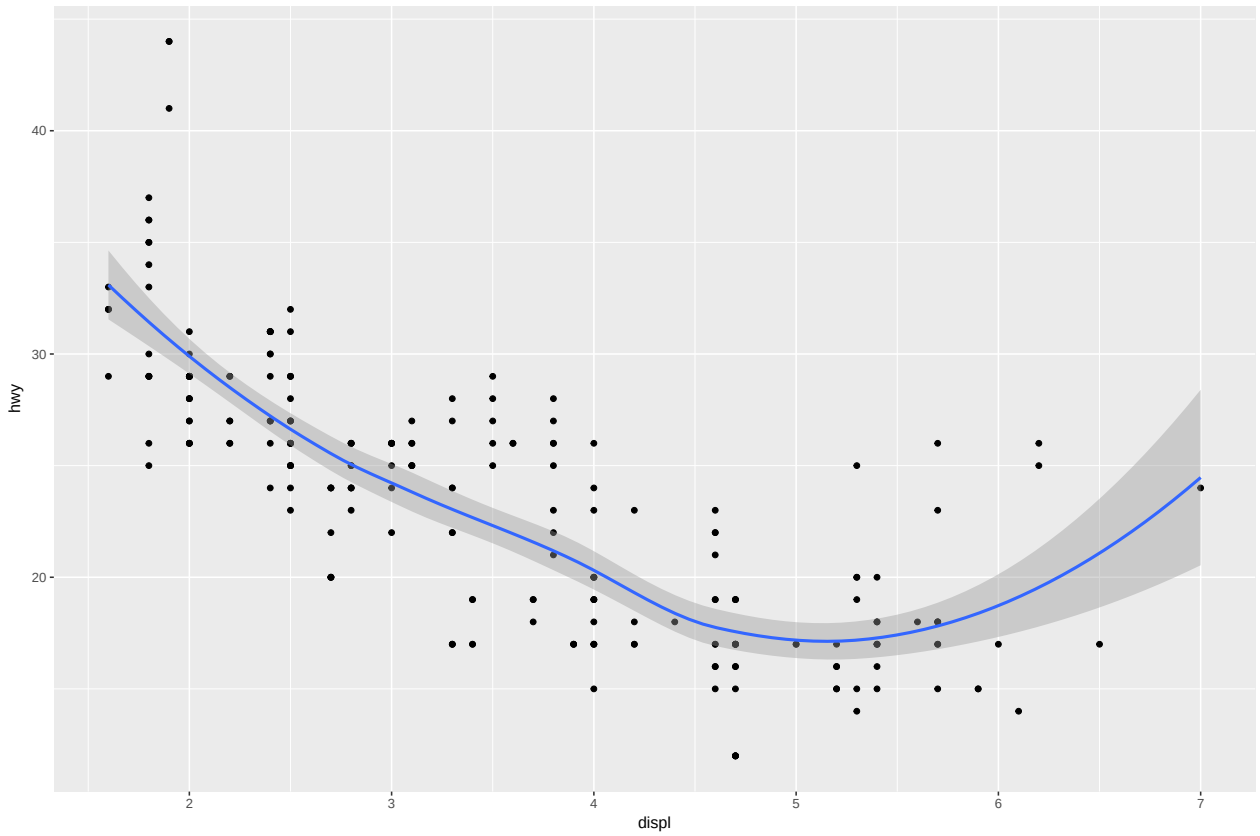

```
# Set the color aesthetic to `drv` to automatically group the data by drivetrain and distinguish them by
ggplot(data = mpg) +
  geom_smooth(
    mapping = aes(x = displ, y = hwy, color = drv),
    show.legend = FALSE
  )
```



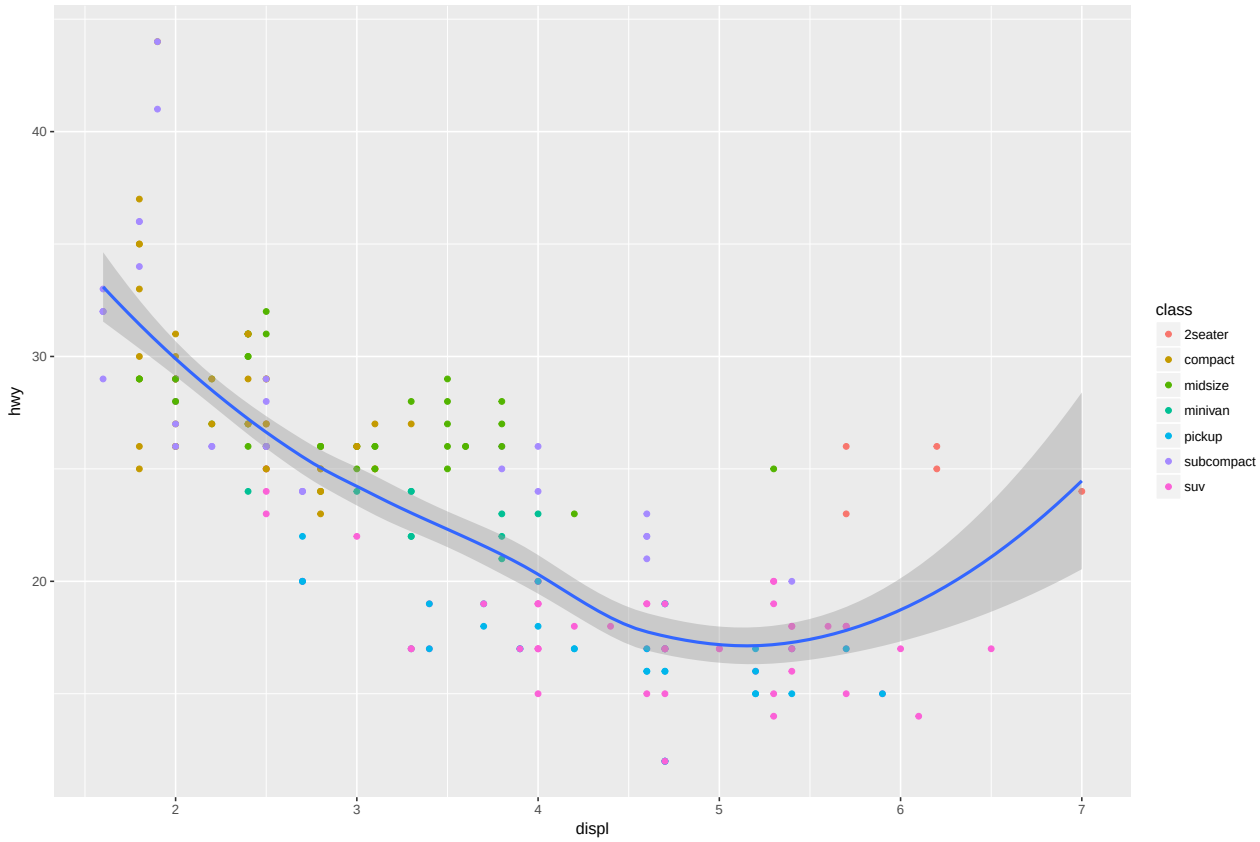

```
# Plot a smooth geom over a scatterplot of the data, the verbose way
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  geom_smooth(mapping = aes(x = displ, y = hwy))
```



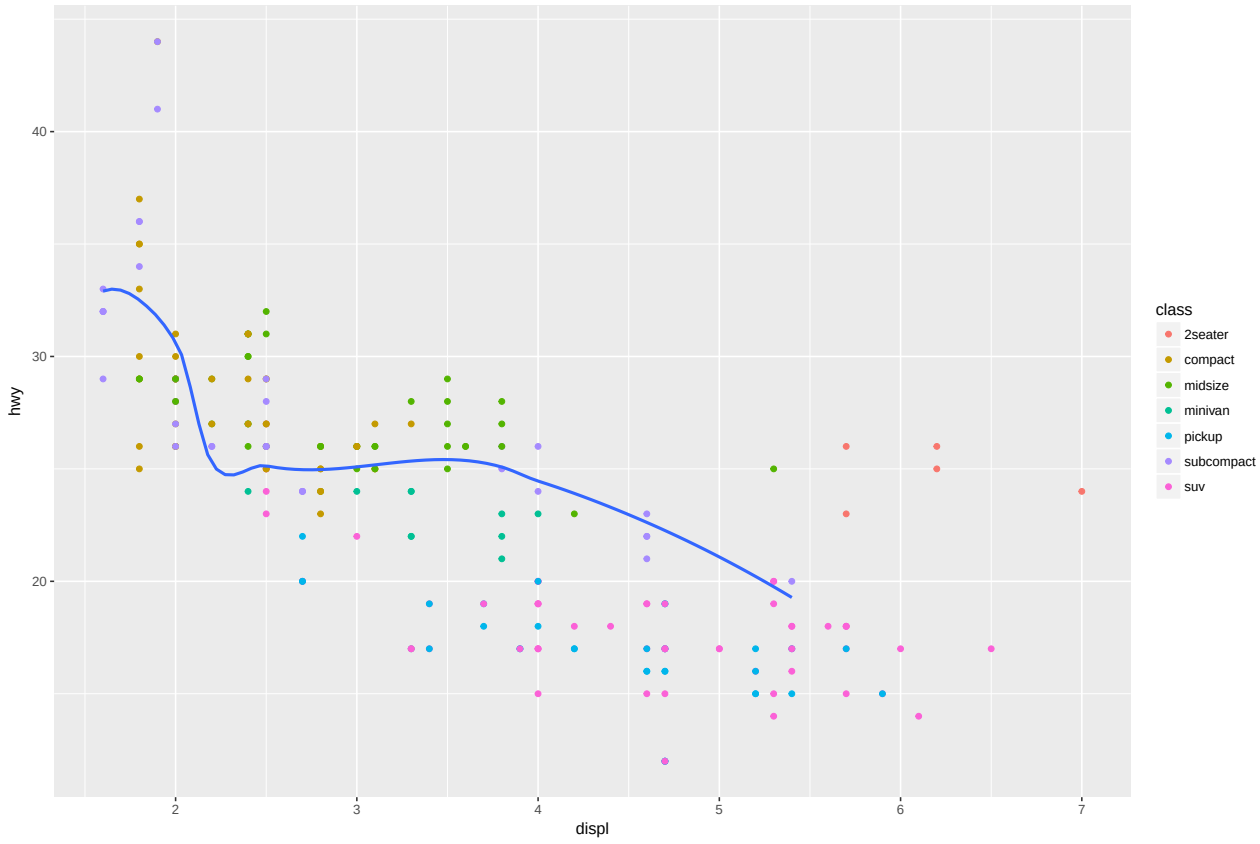

```
# The same plot with mappings passed to `ggplot()`
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth()
```




```
# Color to the points by car class
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(mapping = aes(color = class)) +
  geom_smooth()
```




```
# Plot all classes of car, but draw a smooth line geom for only cars of the subcompact class
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(mapping = aes(color = class)) +
  geom_smooth(data = filter(mpg, class == "subcompact"), se = FALSE)
```



Exercises 3.6.1

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

What geom would you use to draw a(n)
- line chart: `geom_line()` - boxplot: `geom_boxplot()` - histogram: `geom_histogram()` - area chart: `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.

The output will be a scatterplot with engine displacement mapped to the x-axis, miles per gallon highway on the y-axis, and the points colored by type of drivetrain. The scatterplot will be overlaid by one solid smooth line for each drivetrain type and will not expand to indicate the confidence interval. The color of the line for a given type of drivetrain will match the color of the points for the same.

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

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 = FALSE` indicates that a legend should not be included in the plot. If it is removed, the default is to include a legend if any aesthetics are mapped. It may have been used earlier in the chapter to introduce us to the option, to make the final graph in the set of three match the first two, which did not have legends, or for another unknown reason.

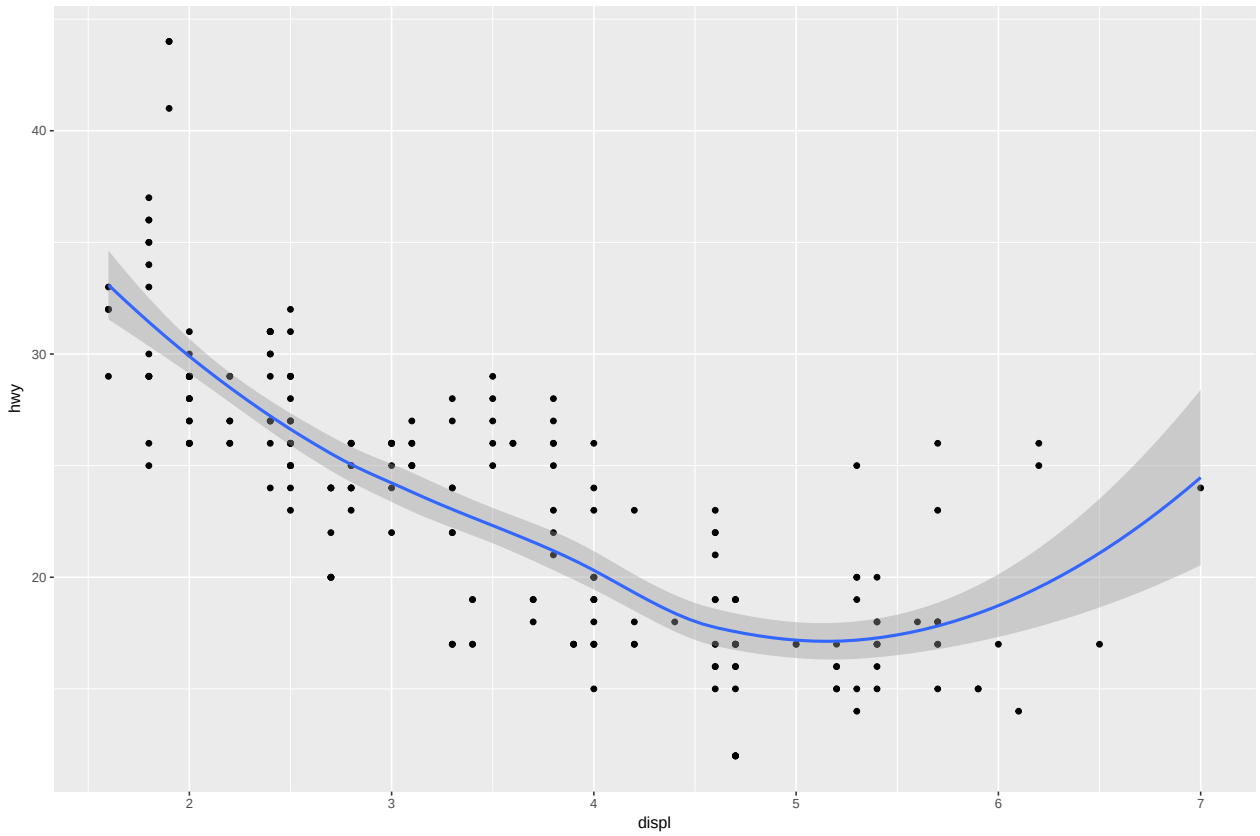
4. What does the se argument to geom_smooth() do?

The `se` argument to `geom_smoot()` indicates whether to include confidence intervals around smooth. The default is TRUE.

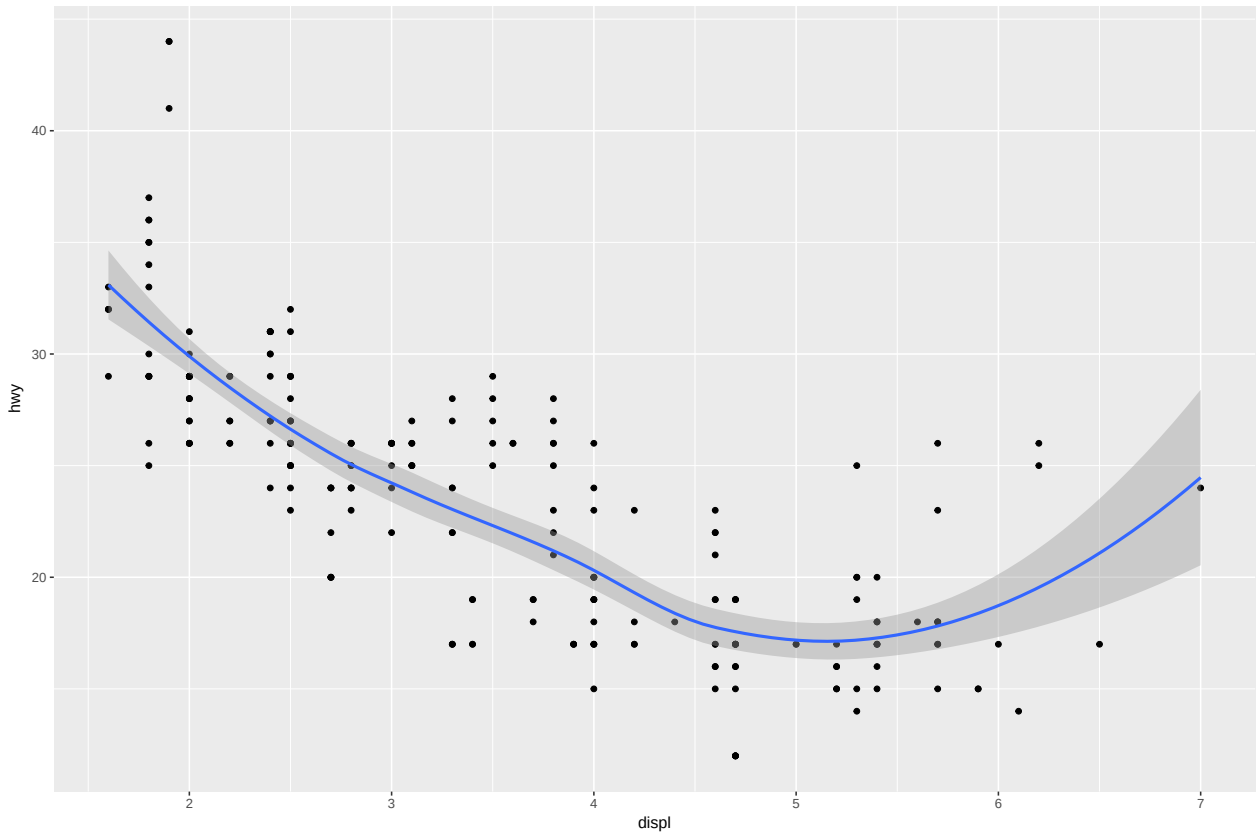
5. Will these two graphs look different? Why/why not?

The two plots will be identical. In the first polot, the selection o `mpg` as the data frame from which to draw the variables and the mapping of `disp` to the x-axis and `hwy` to the y-axis is specified in the global mappings, within `ggplot()`. These mappings extend to the following layers, in this case `geom_point()` and `geom_smooth()` unless those layers explicitly overwrite the global mappings, which they don't here. In the second plot, the same mappings are specified as in the global settings of the first plot. Therefore, in both plots, the mappings specified for `geom_point()` and `geom_smooth()` are identical, though they are explicitly laid out for each layer in plot 2, whereas they are carried over from the global mappings in plot 1.

```
# Plot using global mappings
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth()
```

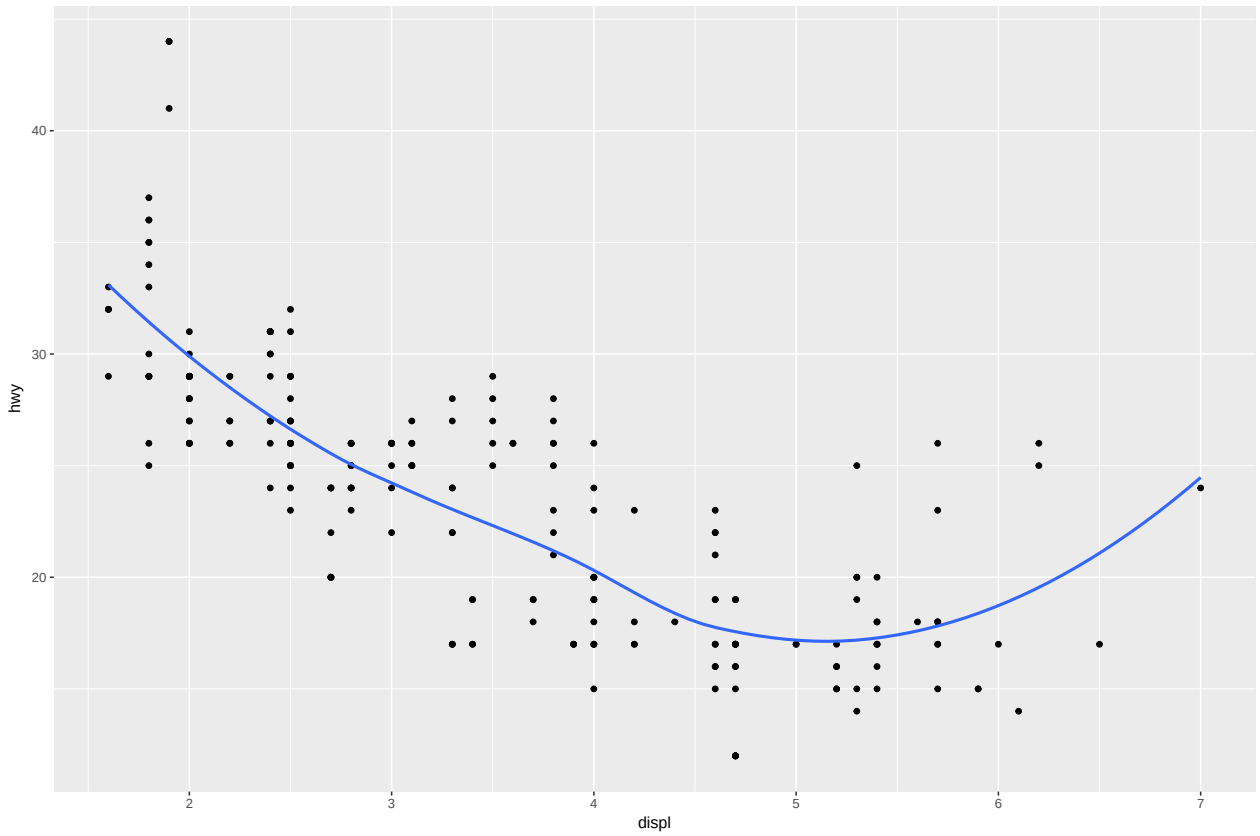



```
# Plot specifcing mappings in each geom layer
ggplot() +
  geom_point(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_smooth(data = mpg, mapping = aes(x = displ, y = hwy))
```

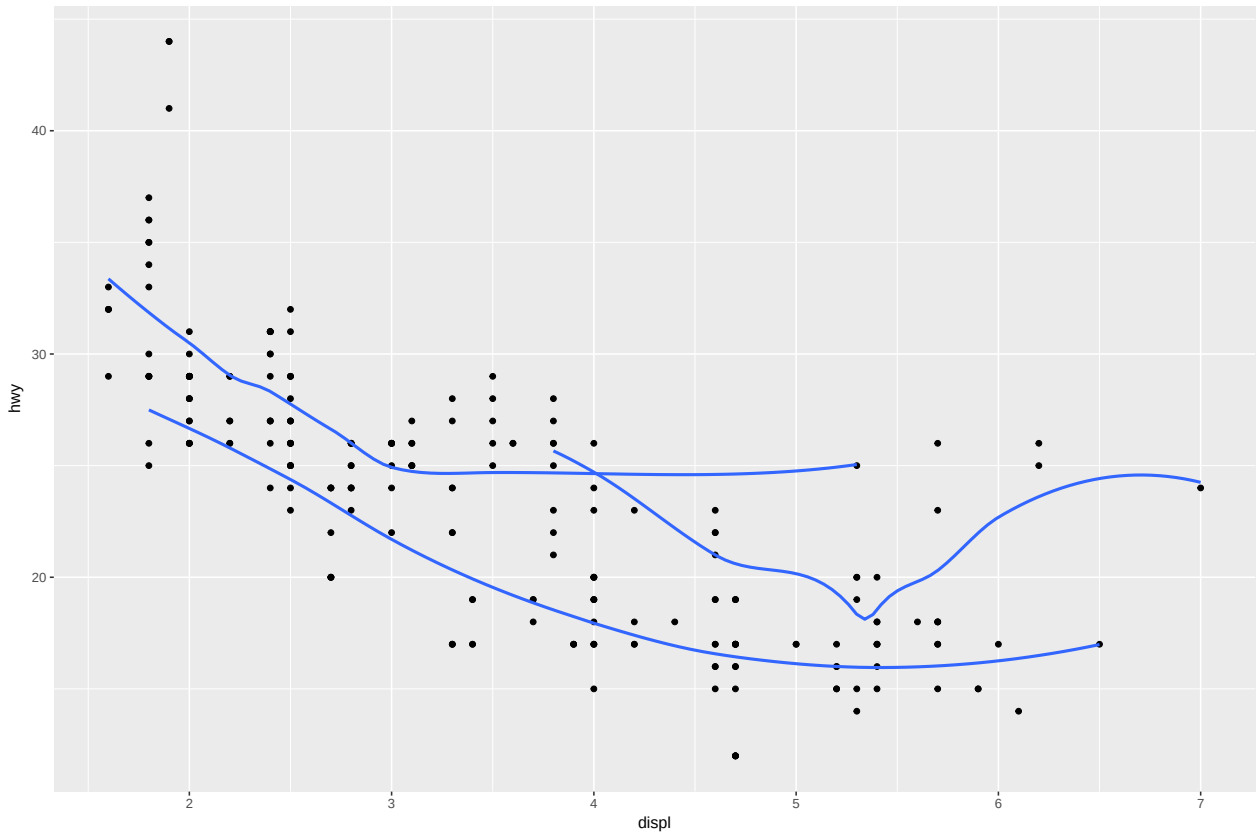


6. Recreate the R code necessary to generate the following graphs.

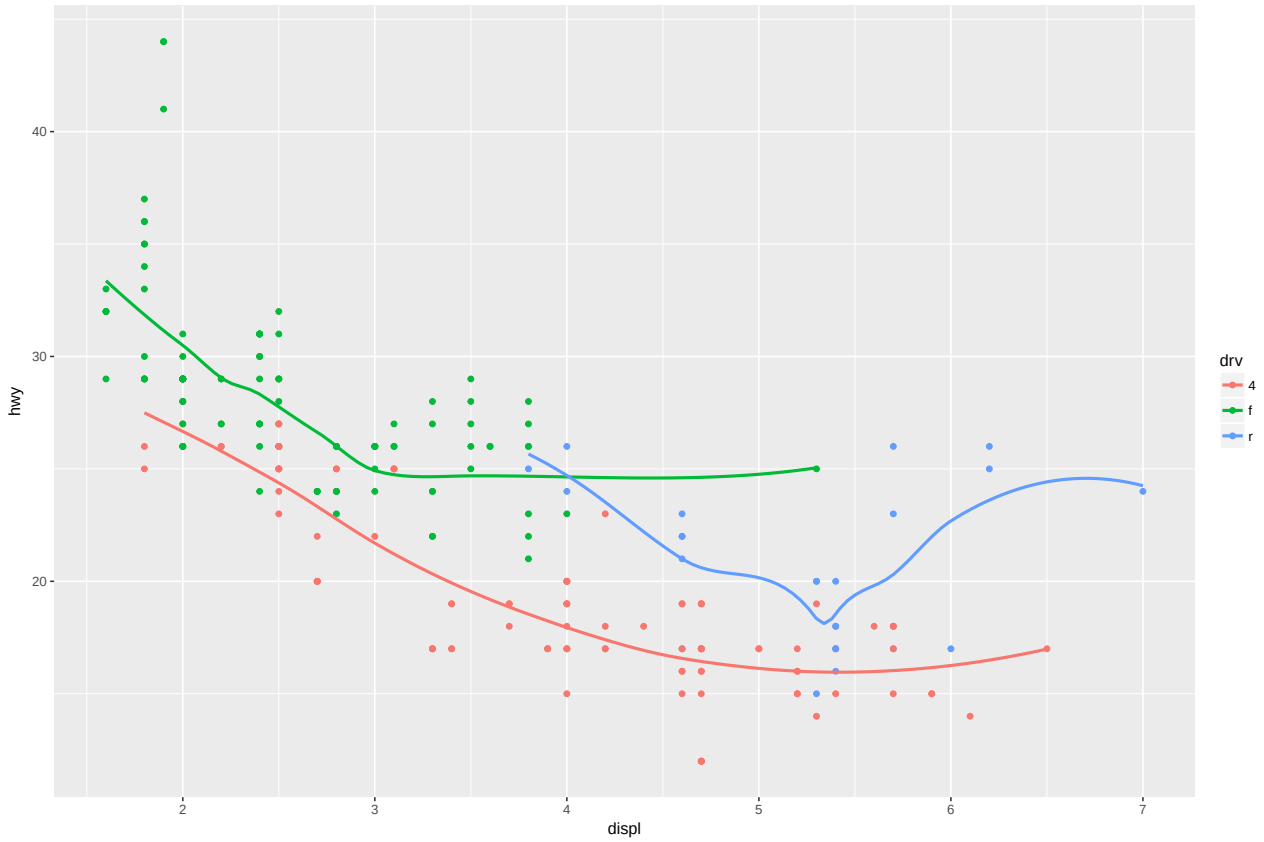
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
    geom_point() +
    geom_smooth(se=FALSE)
```




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



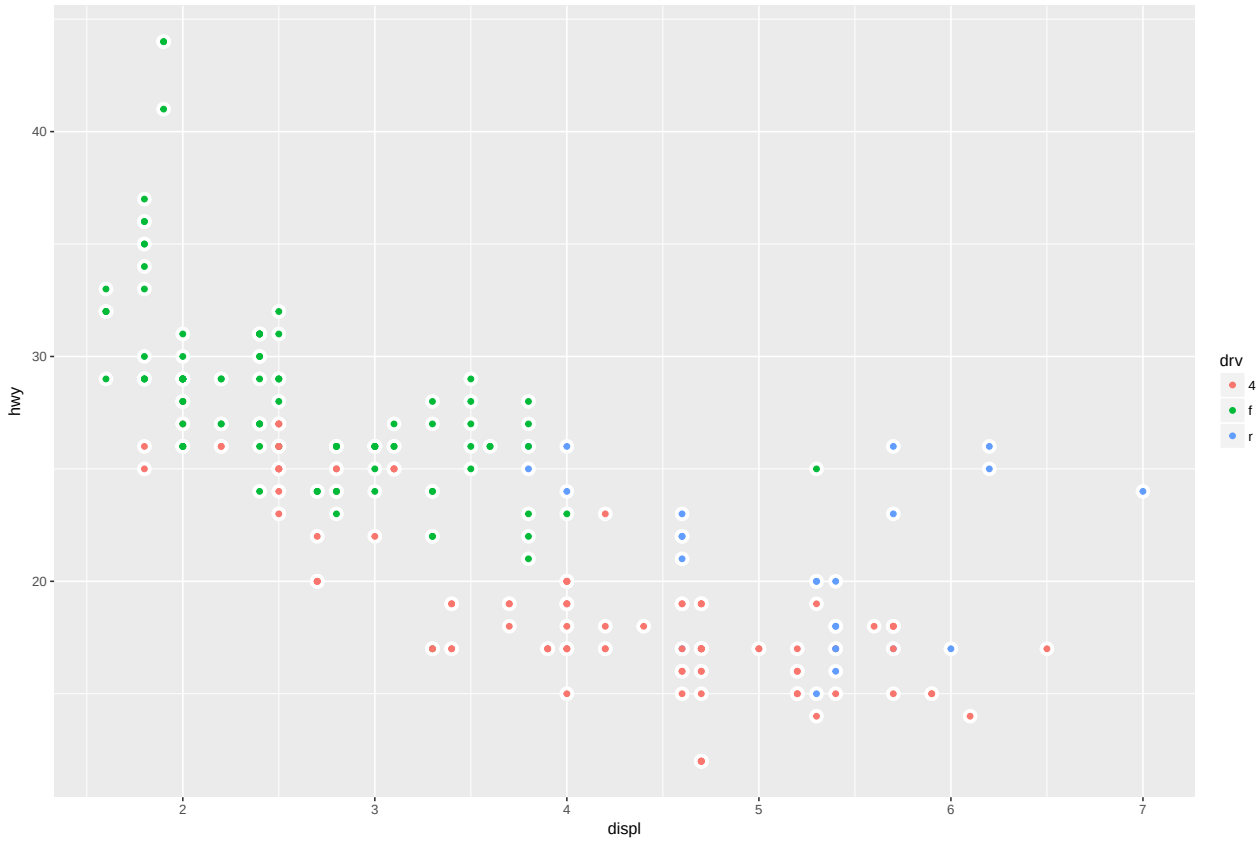

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = drv)) +
    geom_point() +
    geom_smooth(se=FALSE)
```




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

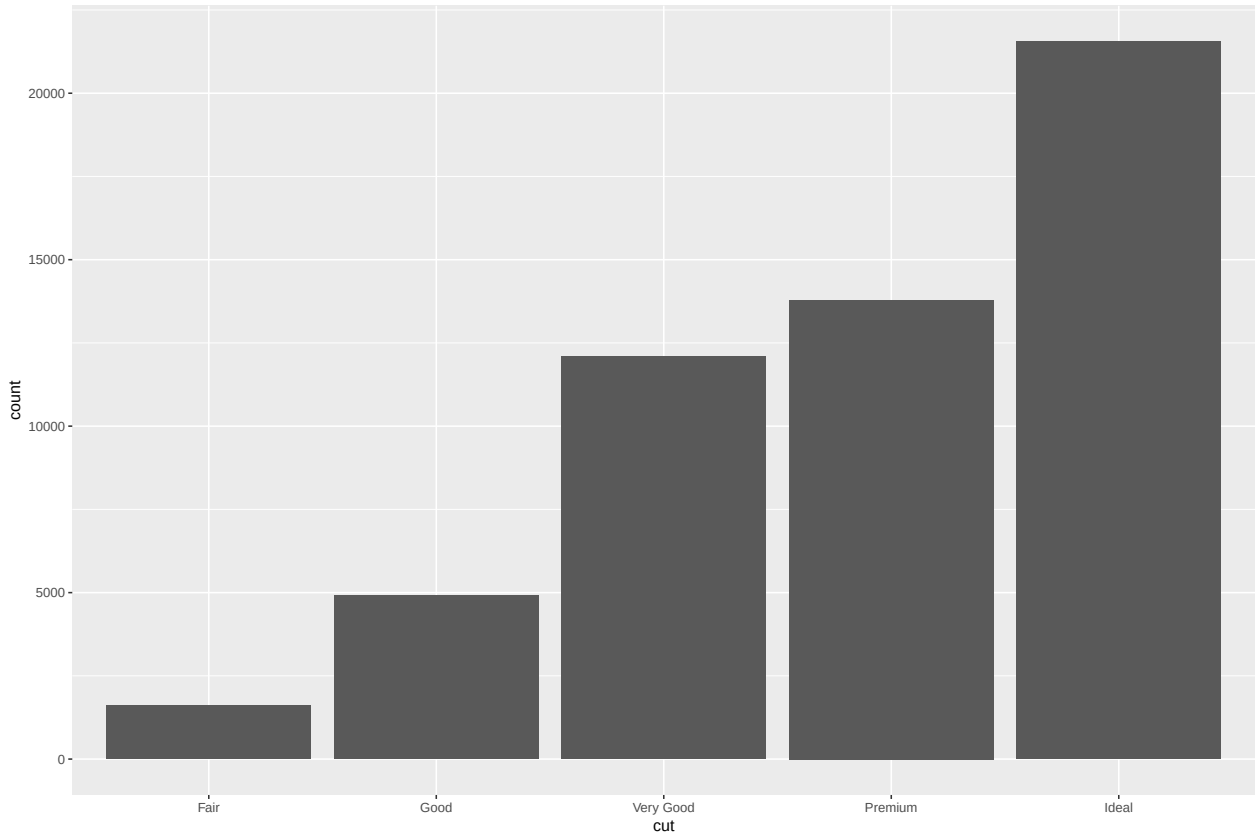

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
    geom_point(aes(color=drv)) +
    geom_smooth(aes(linetype=drv),se=FALSE)
```


```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(size = 4, colour = "white") +
  geom_point(aes(colour = drv))
```

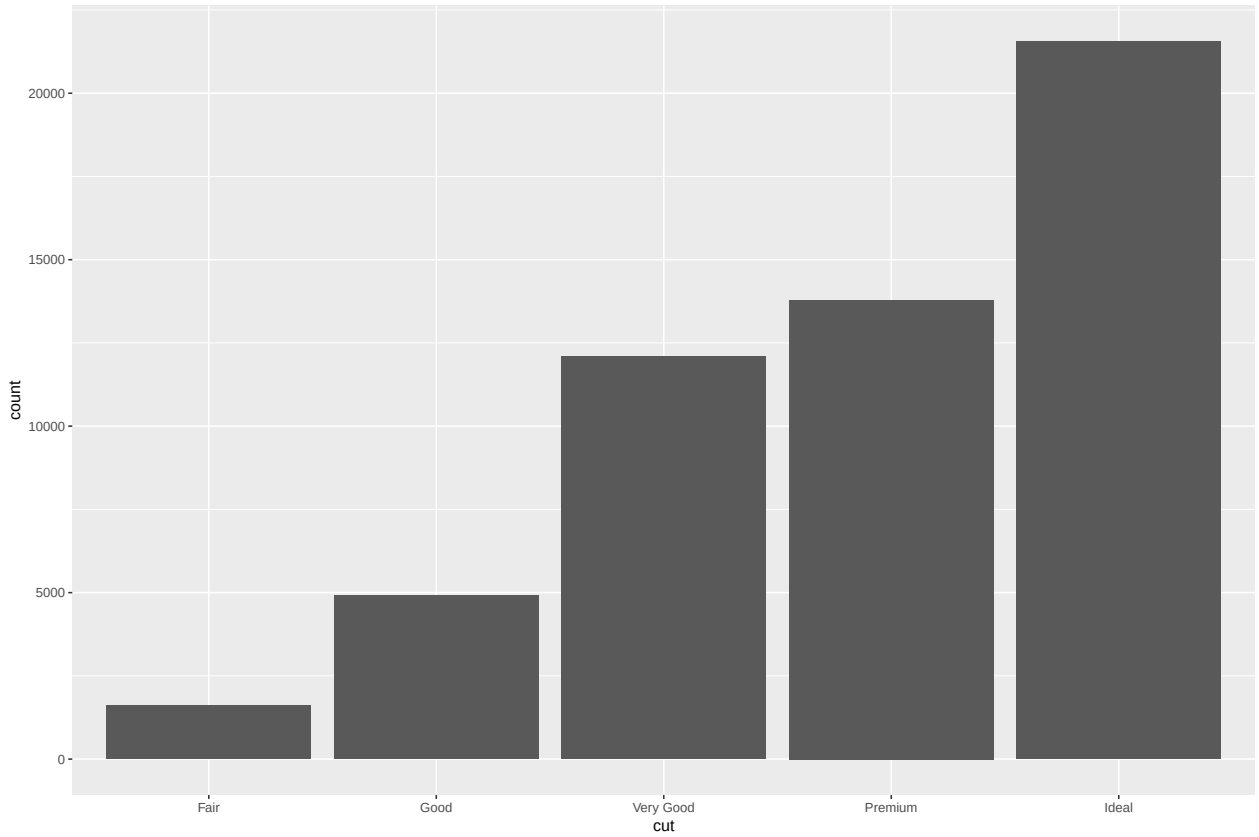


Statistical transformations

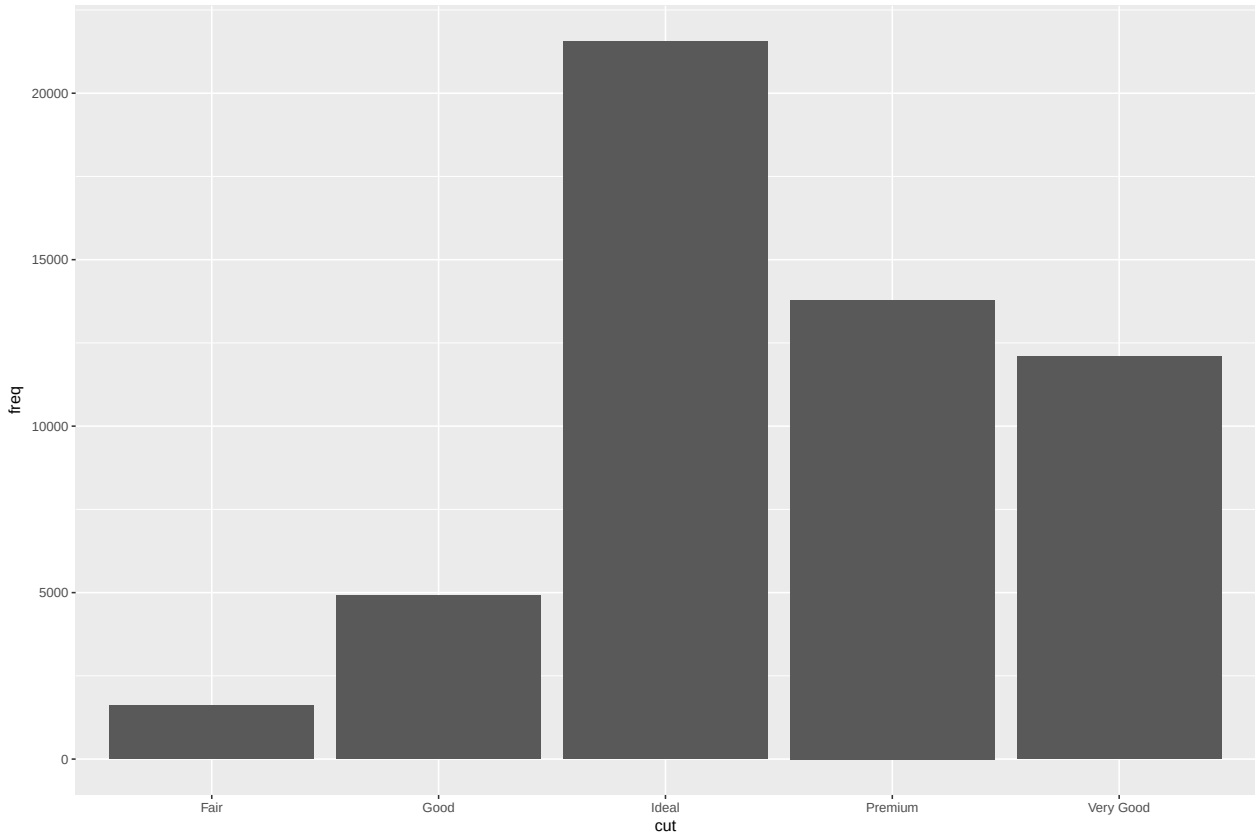
```
 # Simple bar chart using `geom_bar()`
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut))
```



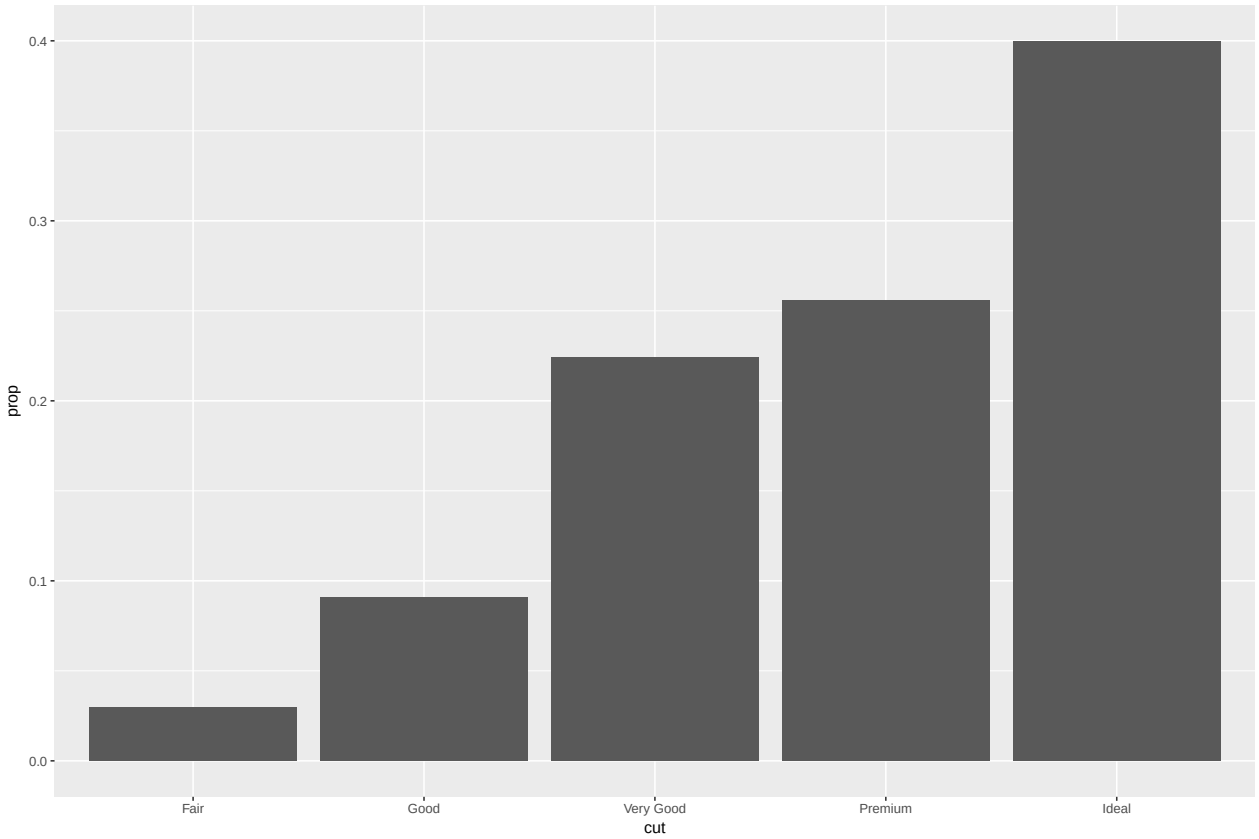

```
?geom_bar

# The same plot using `stat_count()`
ggplot(data = diamonds) +
  stat_count(mapping = aes(x = cut))
```



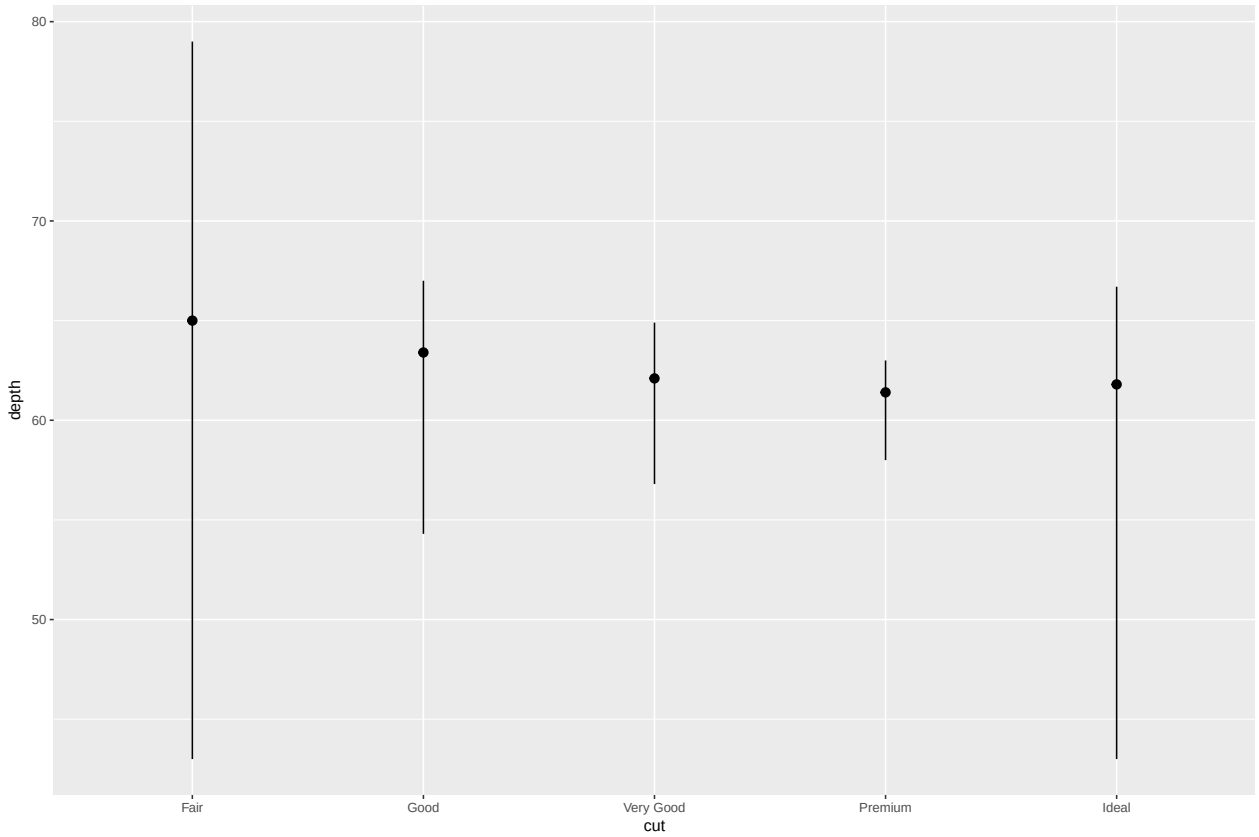

```r
# Create a data set of type tribble and call it "demo"
demo <- tribble(
  ~cut,          ~freq,
  "Fair",        1610,
  "Good",        4906,
  "Very Good",   12082,
  "Premium",     13791,
  "Ideal",       21551
)

# Create a bar chart where the height of the bars is indicated by the value in the `freq` column of `de
ggplot(data = demo) +
  geom_bar(mapping = aes(x = cut, y = freq), stat = "identity")
```




```
# Create a bar chart of proportion rather than count
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, y = ..prop.., group = 1))
```




```r
# Summarize the y values for each unique x value
ggplot(data = diamonds) +
  stat_summary(
    mapping = aes(x = cut, y = depth),
    fun.ymin = min, # get the minimum depth value for each level of cut
    fun.ymax = max, # get the maximum depth value for each level of cut
    fun.y = median # Get the median depth for each level of cur
  )
```

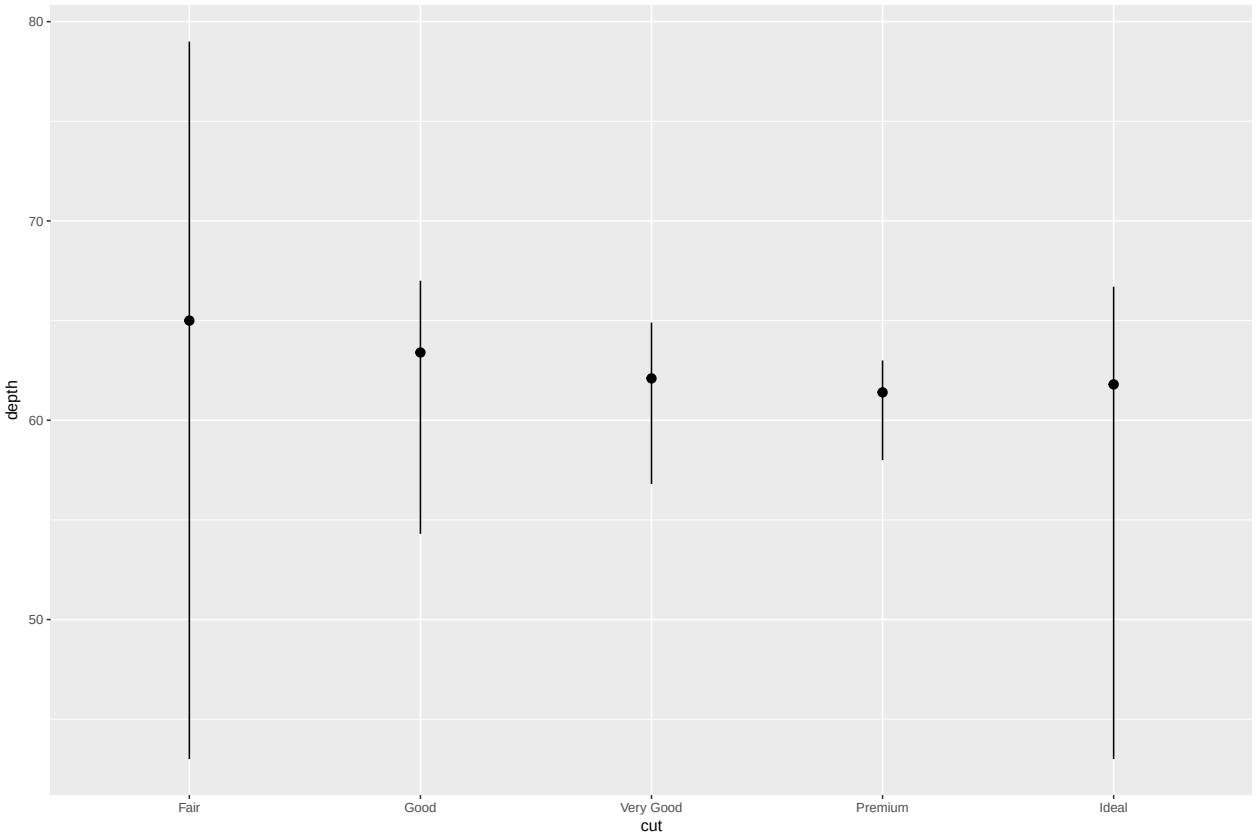


Exercises 3.7.1

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 geom associated with `stat_summary()` is `geom_pointrange()`.

```
?stat_summary

# Use the geom function "pointrange"
ggplot(data = diamonds) +
  geom_pointrange(mapping = aes(x = cut, y = depth),
                  stat = "summary",
                  fun.ymin = min,
                  fun.ymax = max,
                  fun.y = median)
```



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

`geom_col()`does not use `stat = "count"`. Identical to `geom_bar(stat="identity")`, it assumes that the values are already appropriately transformed into single values. In contrast, `geom_bar()` counts the rows using `stat_count()` to draw the bar chart.

From the documentation:

"There are two types of bar charts: `geom_bar` makes the height of the bar proportional to the number of cases in each group (or if the weight aethetic 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. `geom_col` uses `stat_identity`: it leaves the data as is."

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?

Table 1: Which stat each geom pairs with

geom	stat
ggplot2::geom_bar	count
ggplot2::geom_bin2d	bin2d
ggplot2::geom_blank	identity
ggplot2::geom_boxplot	boxplot
ggplot2::geom_contour	contour
ggplot2::geom_count	sum
ggplot2::geom_density	density
ggplot2::geom_density_2d	density2d
ggplot2::geom_errorbarh	identity

geom	stat
ggplot2::geom_hex	binhex
ggplot2::geom_freqpoly	bin
ggplot2::geom_jitter	identity
ggplot2::geom_crossbar	identity
ggplot2::geom_map	identity
ggplot2::geom_path	identity
ggplot2::geom_point	identity
ggplot2::geom_polygon	identity
ggplot2::geom_quantile	quantile
ggplot2::geom_ribbon	identity
ggplot2::geom_rug	identity
ggplot2::geom_segment	identity
ggplot2::geom_smooth	smooth
ggplot2::geom_spoke	identity
ggplot2::geom_label	identity
ggplot2::geom_raster	identity
ggplot2::geom_violin	ydensity

Table 2: Which geom each stat pairs with
Many of the geoms and stats share their default pairmate.

stat	geom
stat_count	bar
stat_bin2d	tile
stat_identity	point
stat_boxplot	boxplot
stat_contour	contour
stat_sum	point
stat_density	area
stat_density2d	density_2d
stat_bin_hex	hex
stat_bin	bar
stat_quantile	quantile
stat_smooth	smooth
stat_ydensity	violin

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

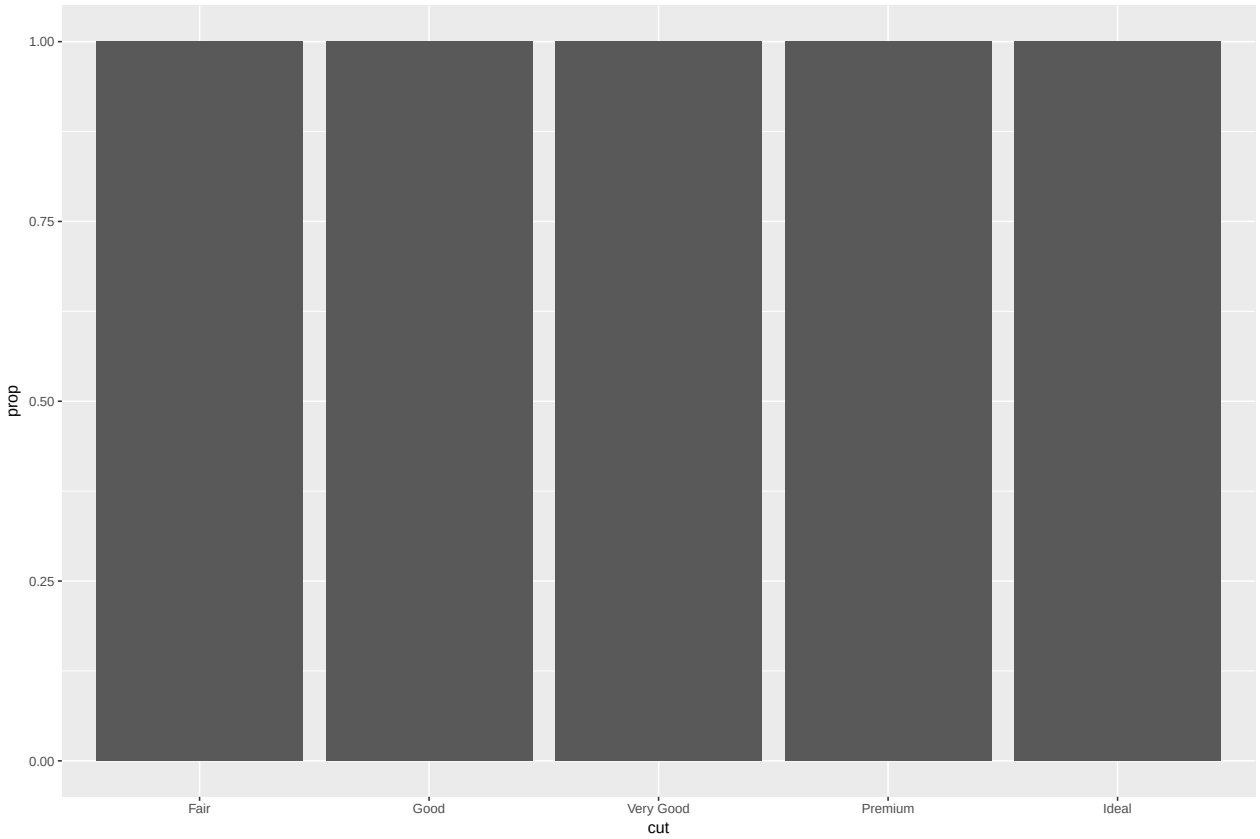
——— | — `y` | predicted value `ymin` | lower pointwise confidence interval around the mean `ymax` | upper pointwise confidence interval around the mean `se` | standard error

Table: `stat_smooth()` computes

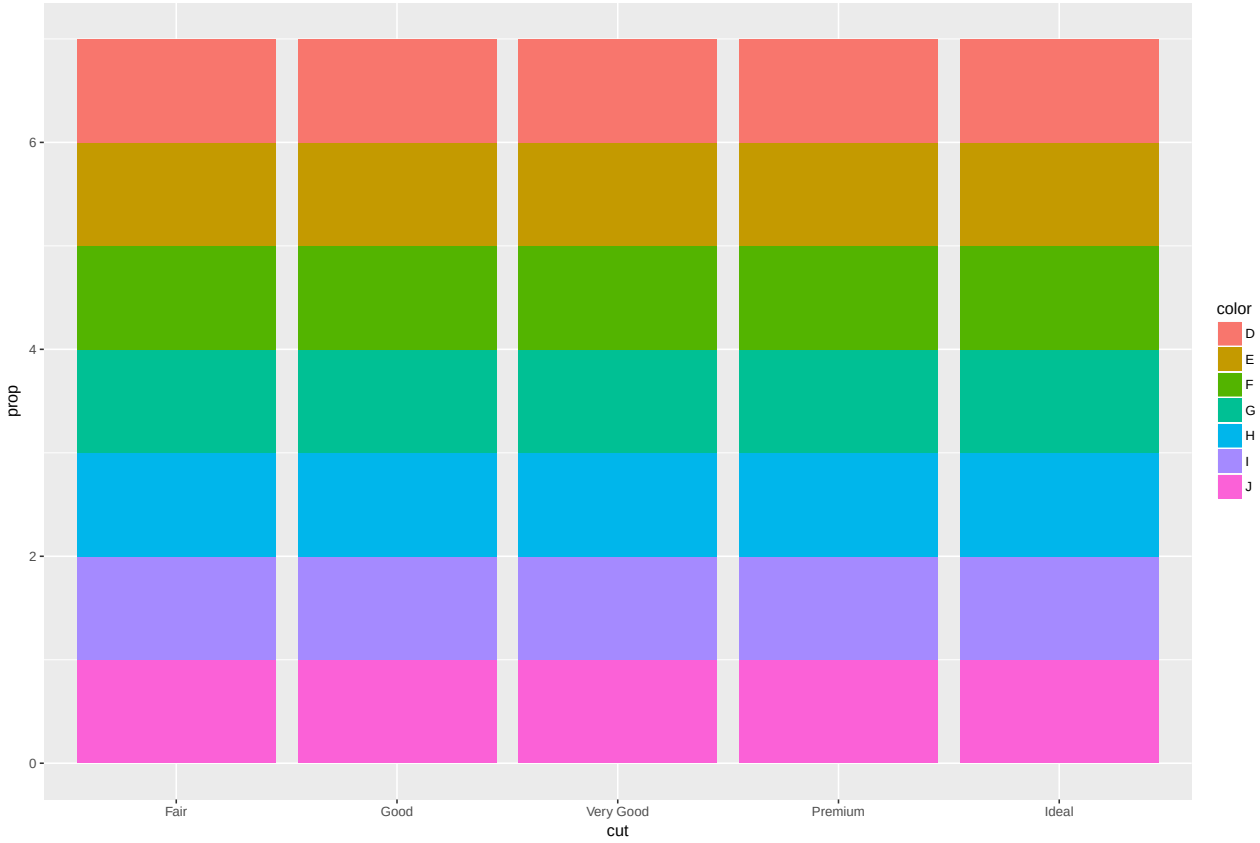
The parameters of `stat_ _smooth` are listed in its documentation. See `?stat_smooth`

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

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

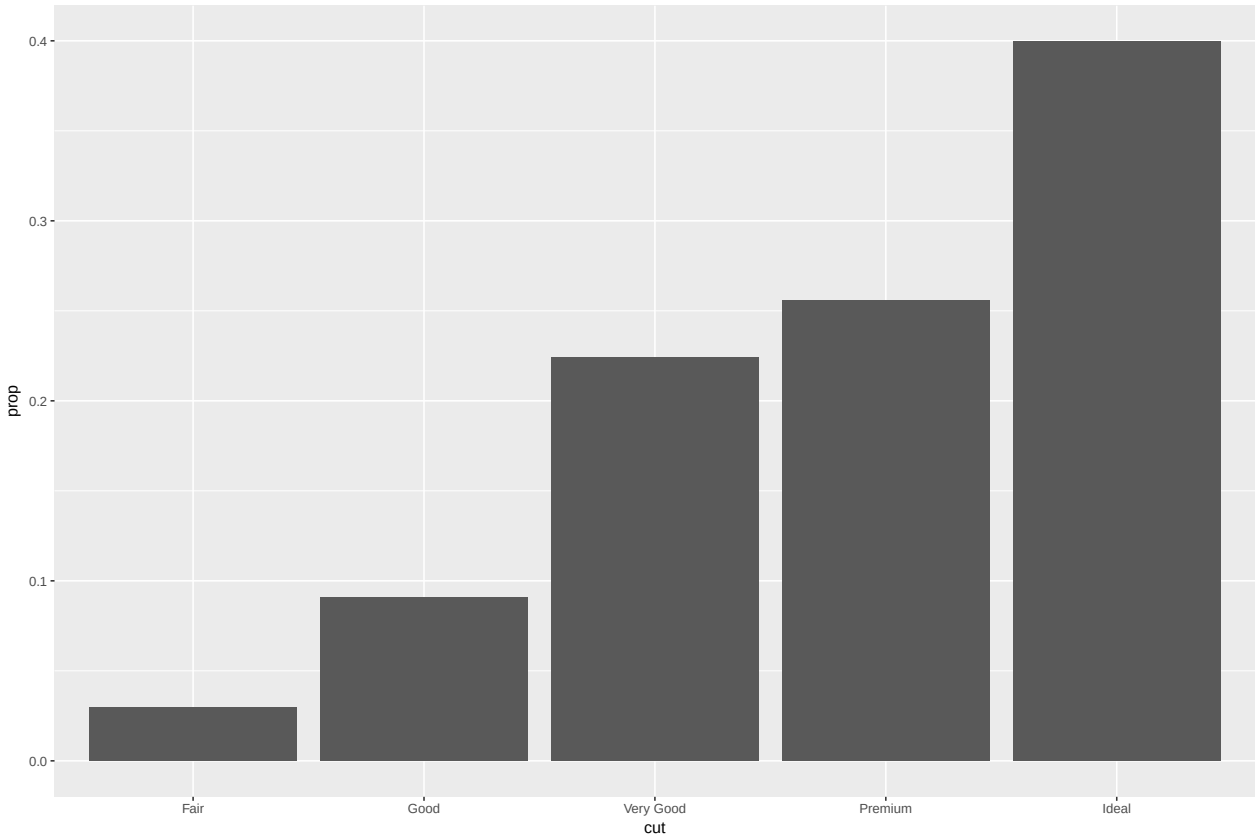



```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = color, y = ..prop..))
```

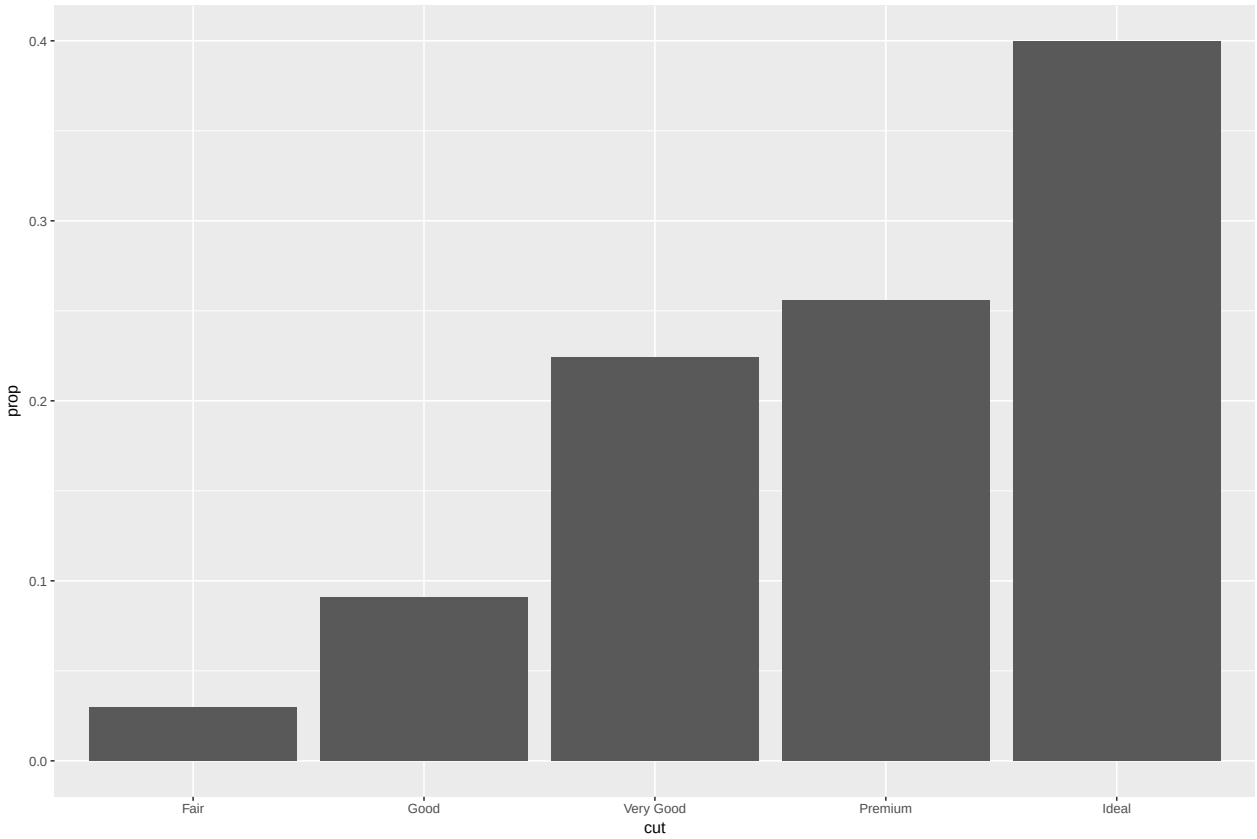


According to the documentation, we need to set `group=1` so that "the height of the bar [is] proportional to the number of cases in each group." When `group=1` the proportions are calculated for level of `cut`, rather than with the entire dataset. See the corrected plots below:

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

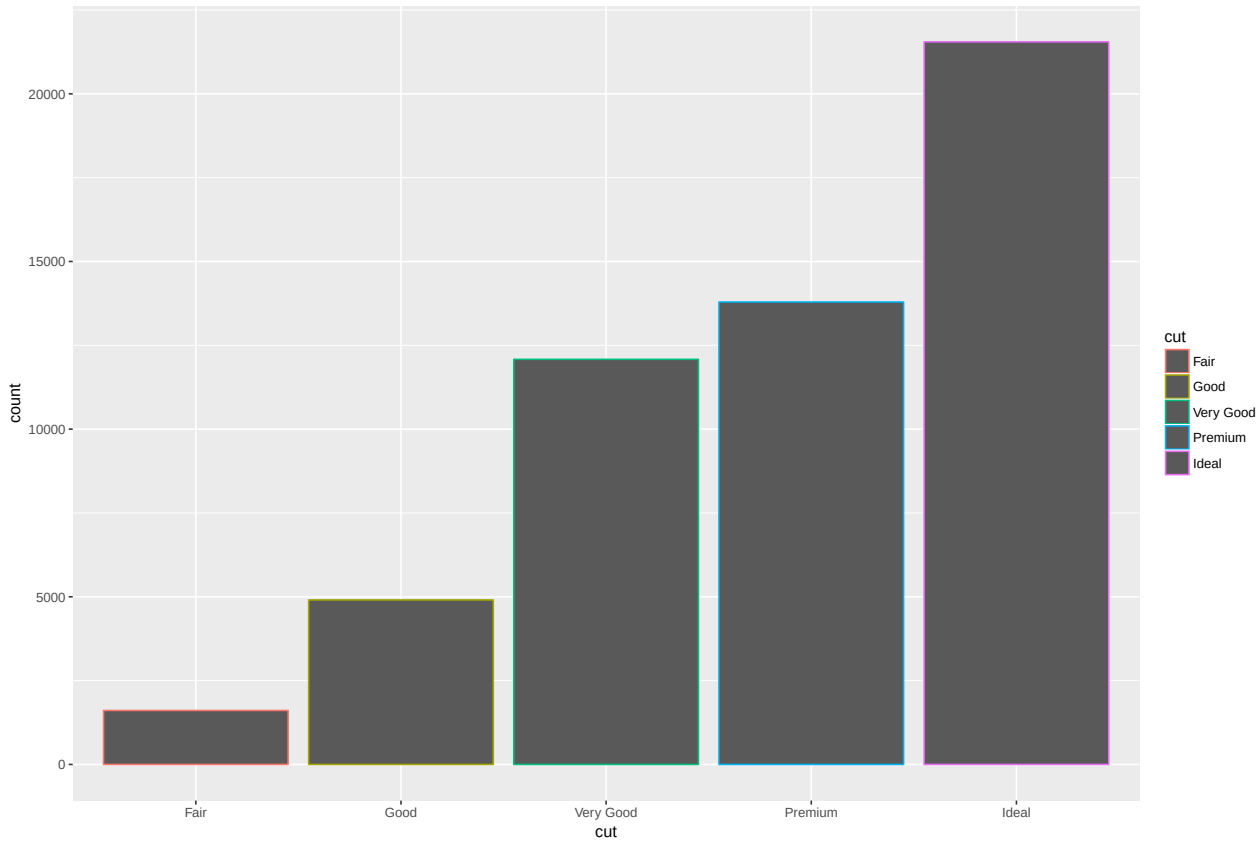



```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = color, y = ..prop.., group=1))
```

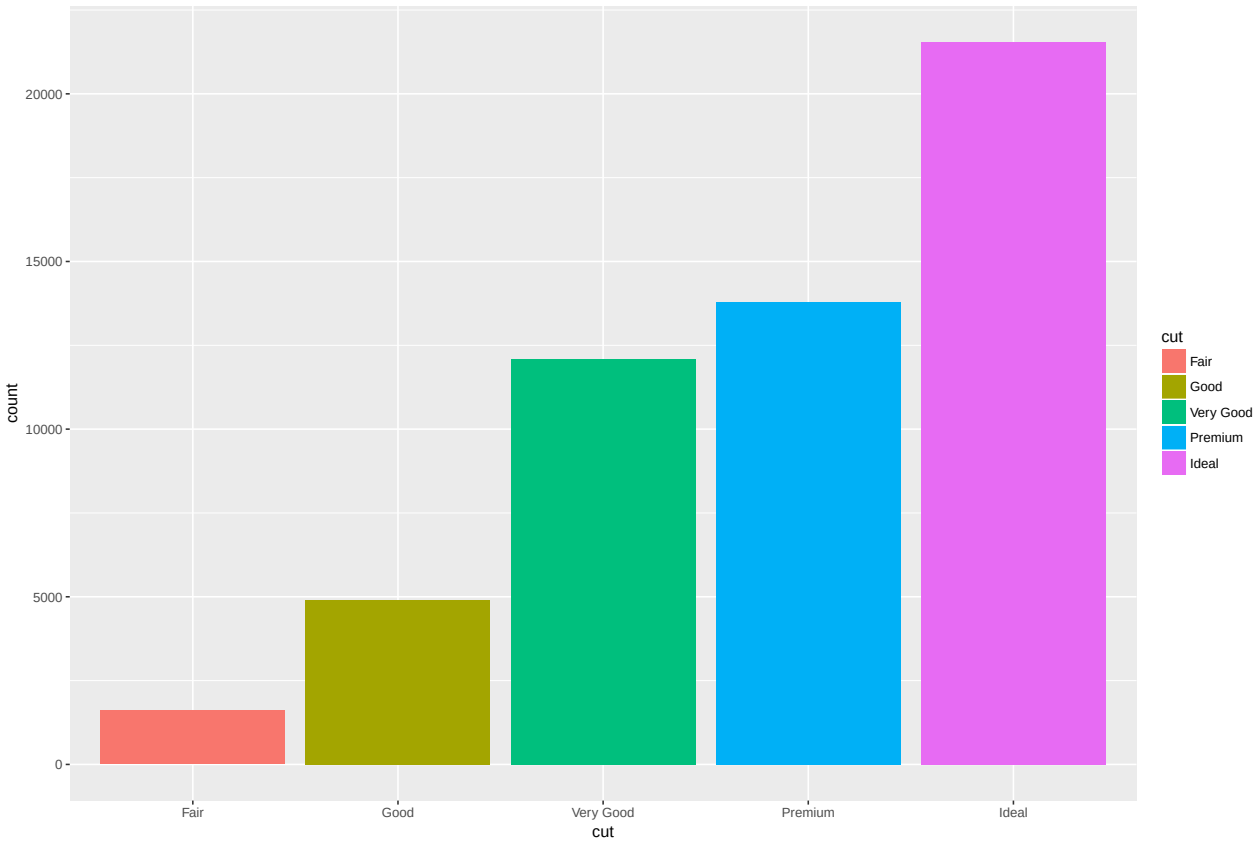


Position adjustments

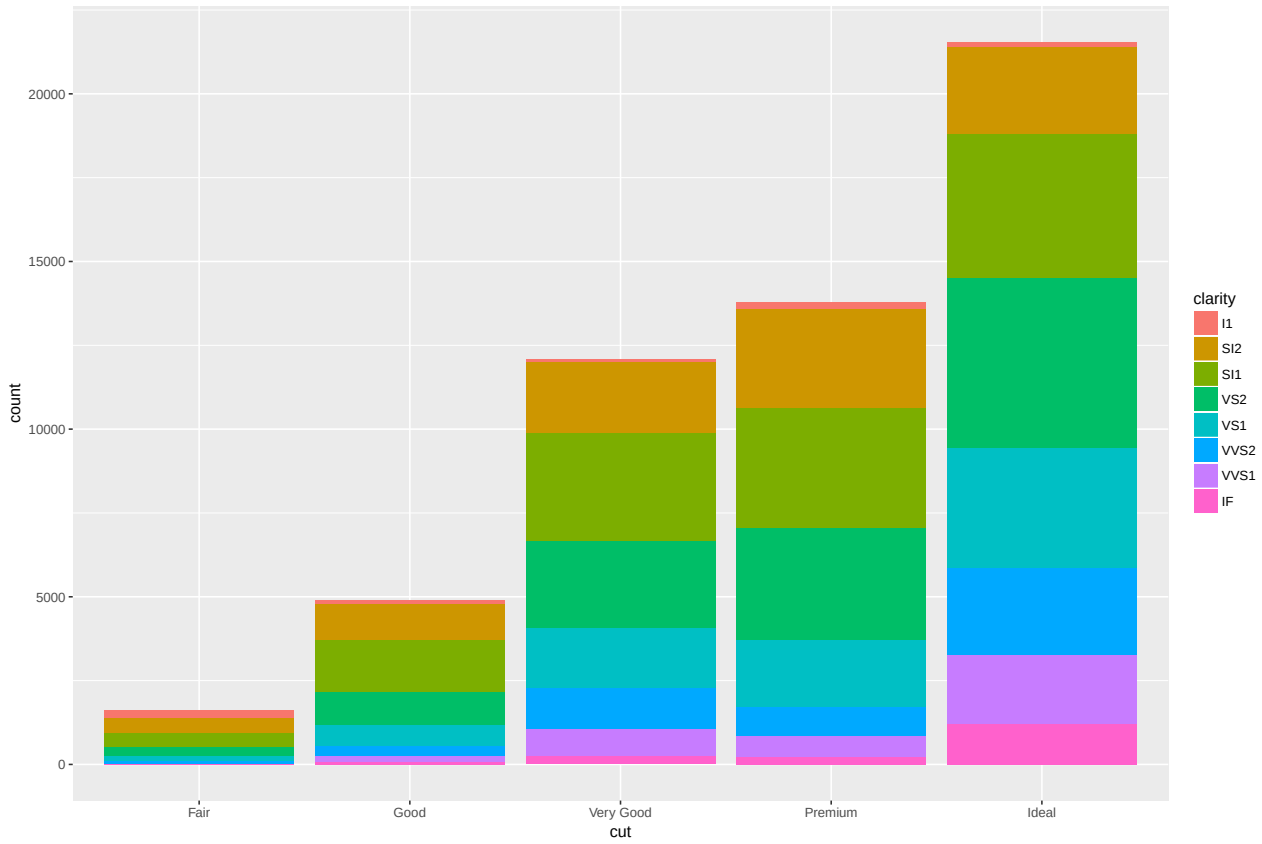
```
# Plot a histogram of the frequency of each type of cut and outline the bar for each cut in a different
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, colour = cut))
```



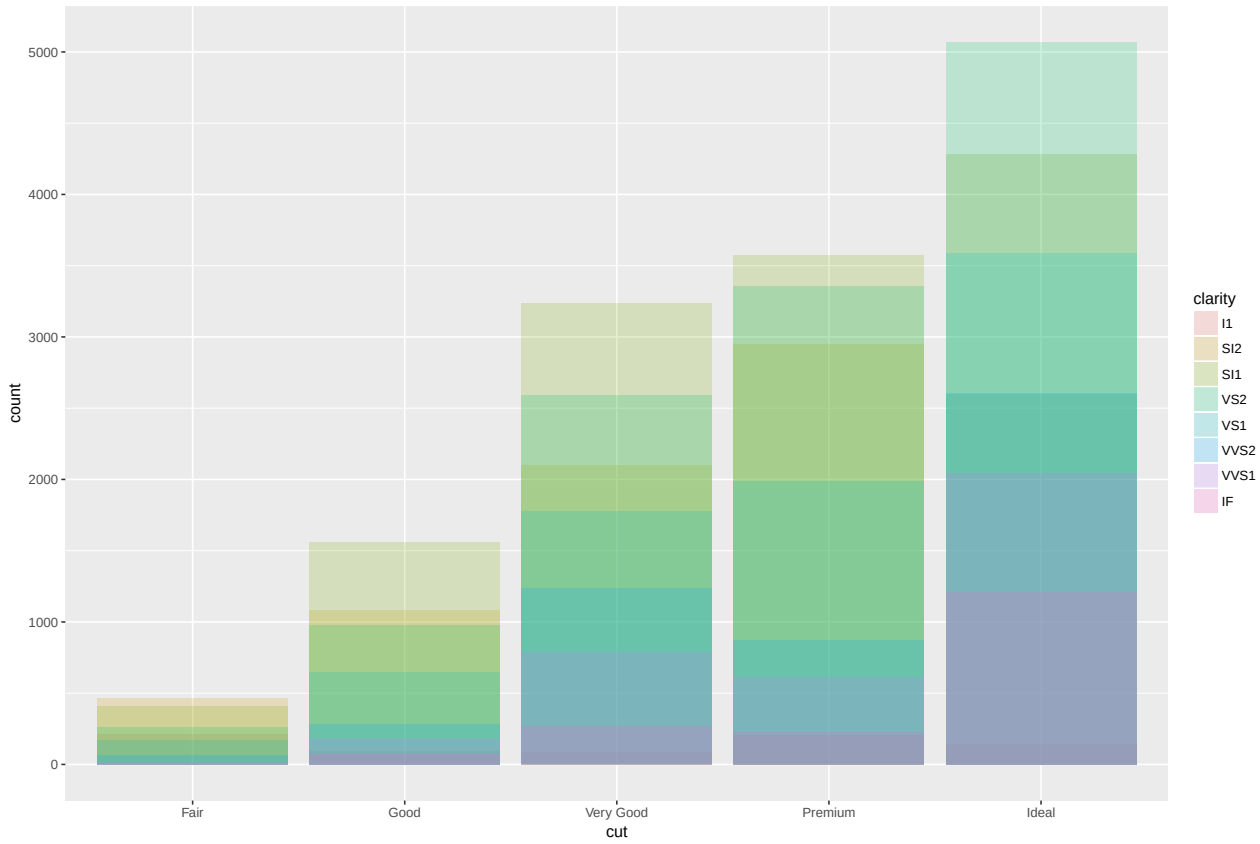

```
# Plot a histogram of the frequency of each type of cut and fill the bar for each cut with a different
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = cut))
```



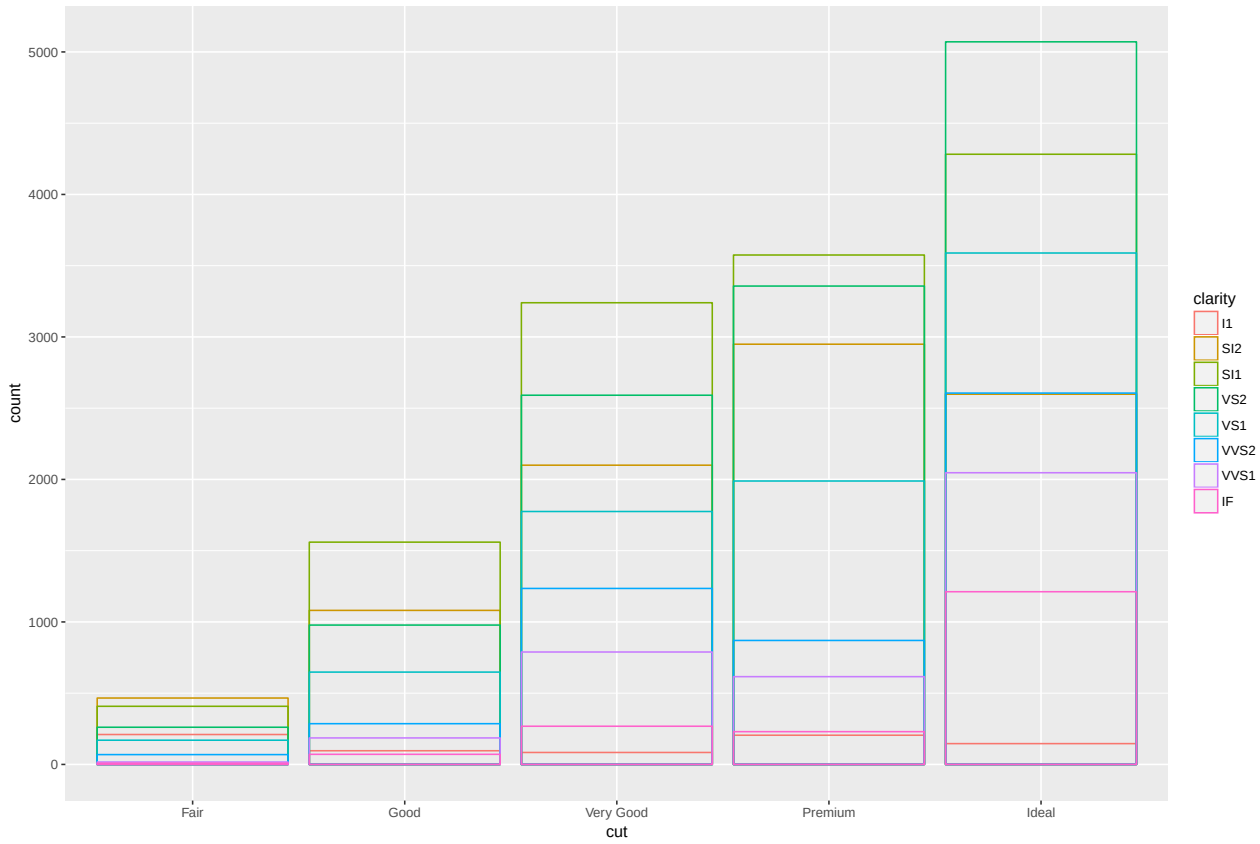

```
# Use a different color for each level of clarity and fill the `cut` bars with the appropriate proporti
ggplot(data=diamonds) +
    geom_bar(mapping = aes(x=cut, fill=clarity))
```



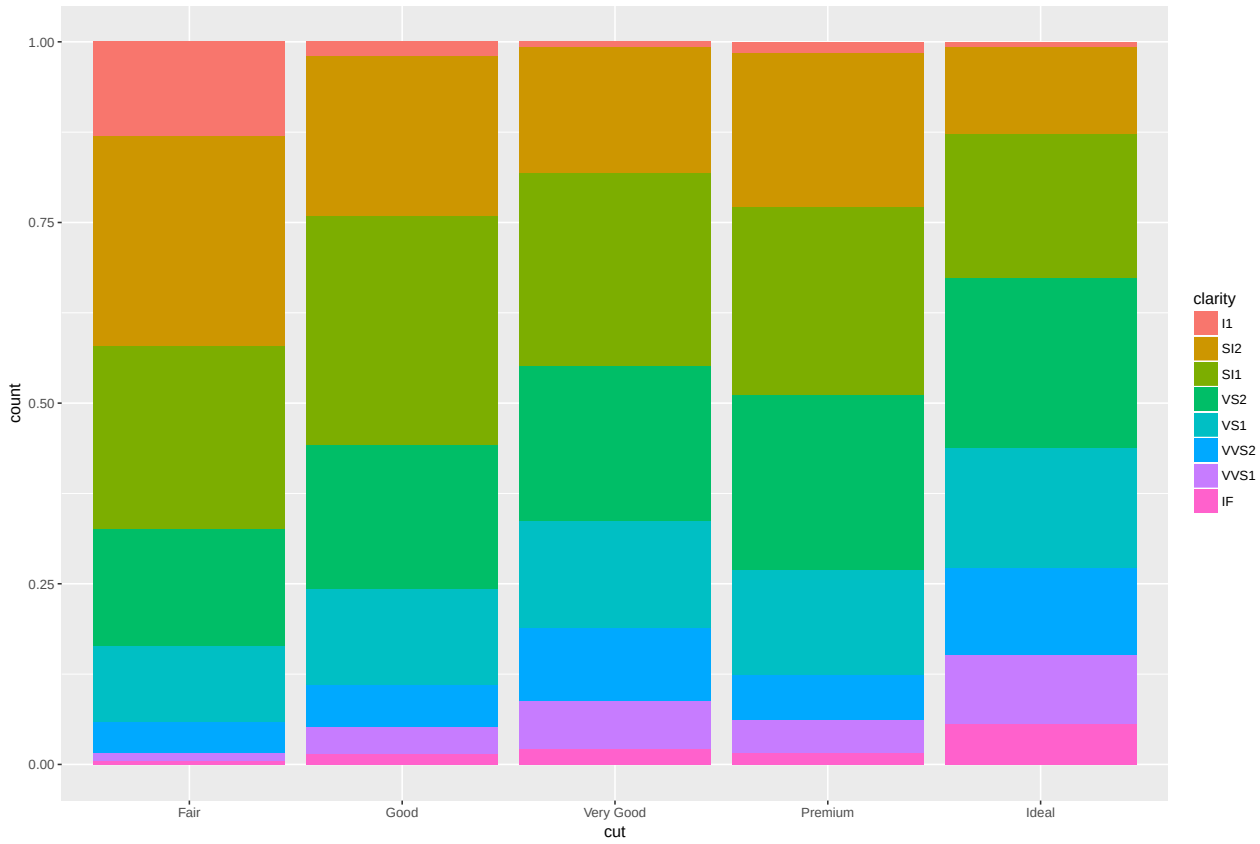

```
# Make the bars semi transparant so they can be seen despite overlapping
ggplot(data = diamonds, mapping = aes(x = cut, fill = clarity)) +
  geom_bar(alpha = 1/5, position = "identity") # Set alpha low to make bars semi-transparent
```



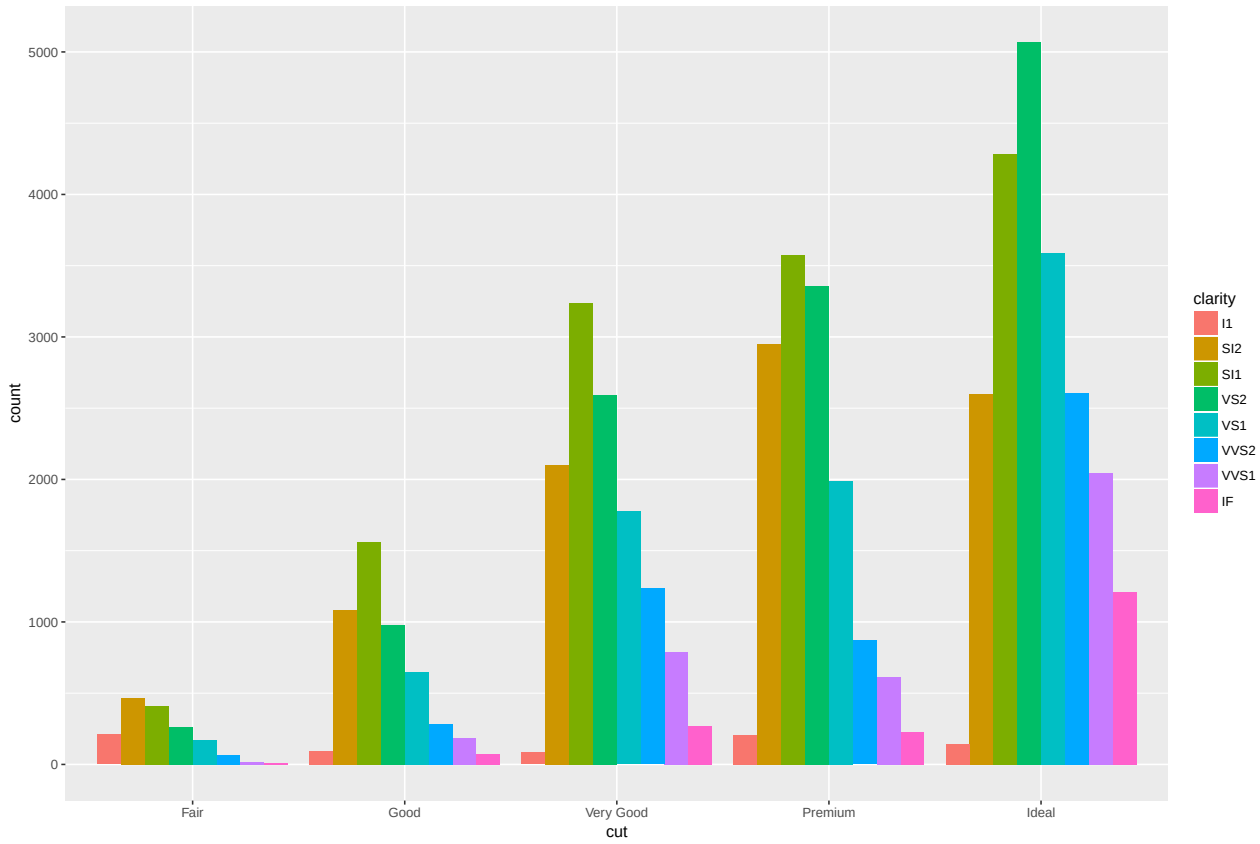

```
# Make bars entirely transparent
ggplot(data = diamonds, mapping = aes(x = cut, colour = clarity)) +
  geom_bar(fill = NA, position = "identity") # set `fill=NA` to make the bars entirely transparent
```



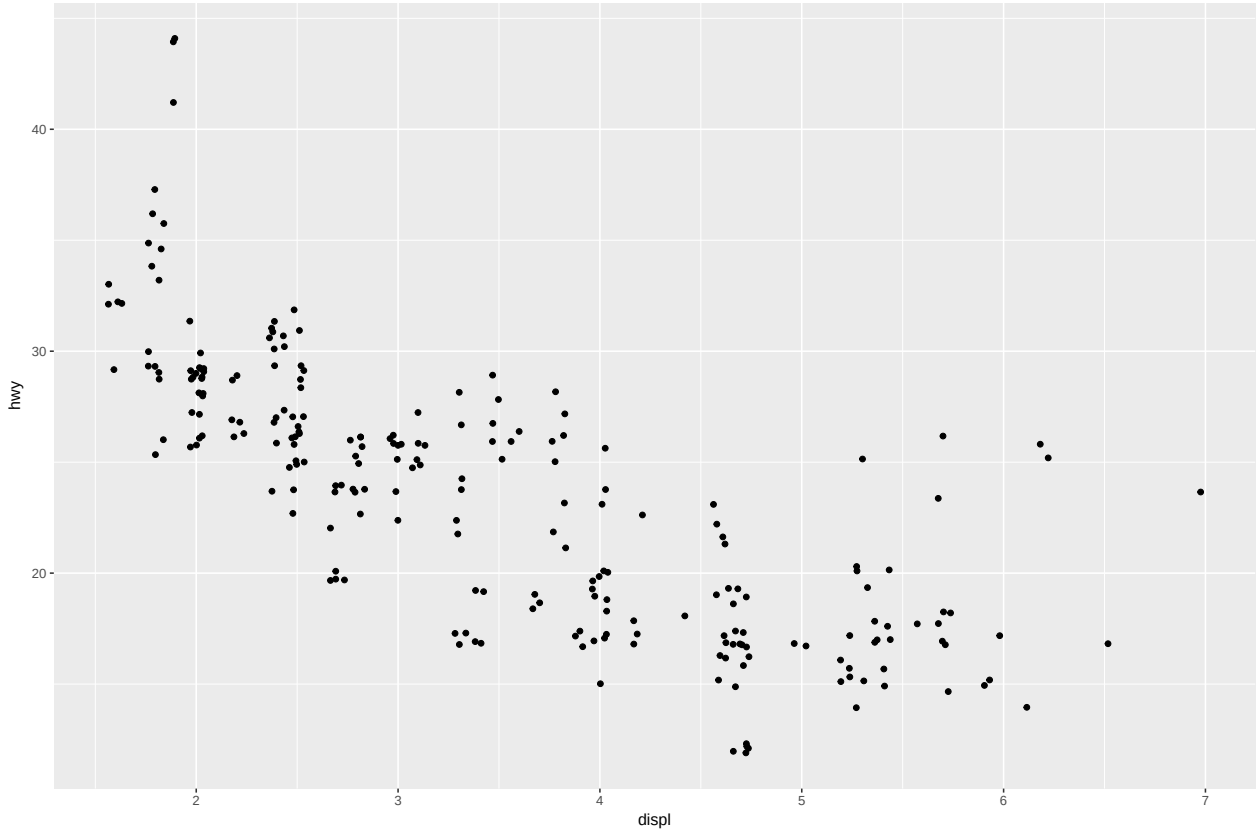

```
# Make each set of stacked bars the same height for easy comparison
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = clarity), position = "fill")
```




```
# Place overlapping bars beside one another
ggplot(data=diamonds) +
    geom_bar(mapping = aes(x=cut, fill = clarity), position = "dodge")
```




```
# Add a small amount of random noise to each point to see ovverlapping points
ggplot(data=mpg) +
    geom_point(mapping = aes(x = displ, y = hwy), position = "jitter")
```



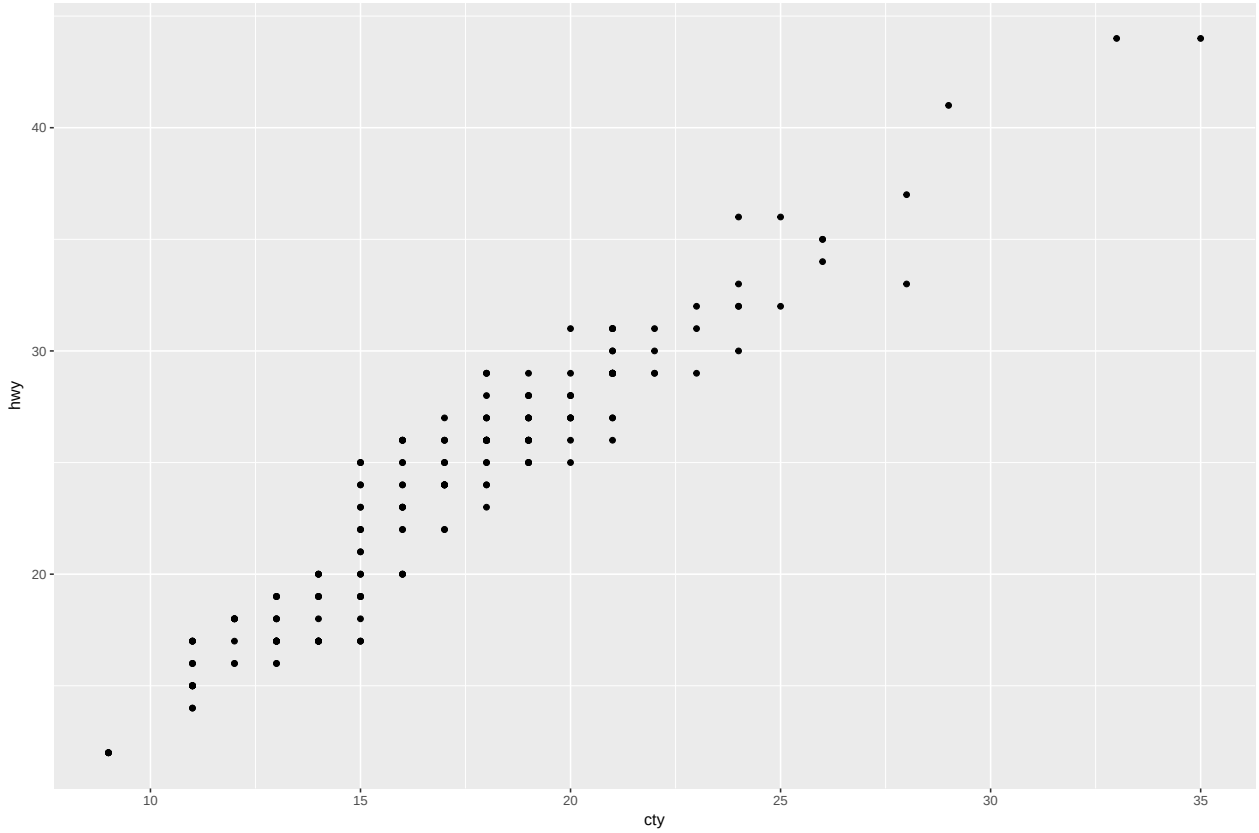
Help pages associated with adjustments

```
?position_dodge
?position_fill
?position_identity
position_jitter
?position_stack
```

Exercises 3.8.1

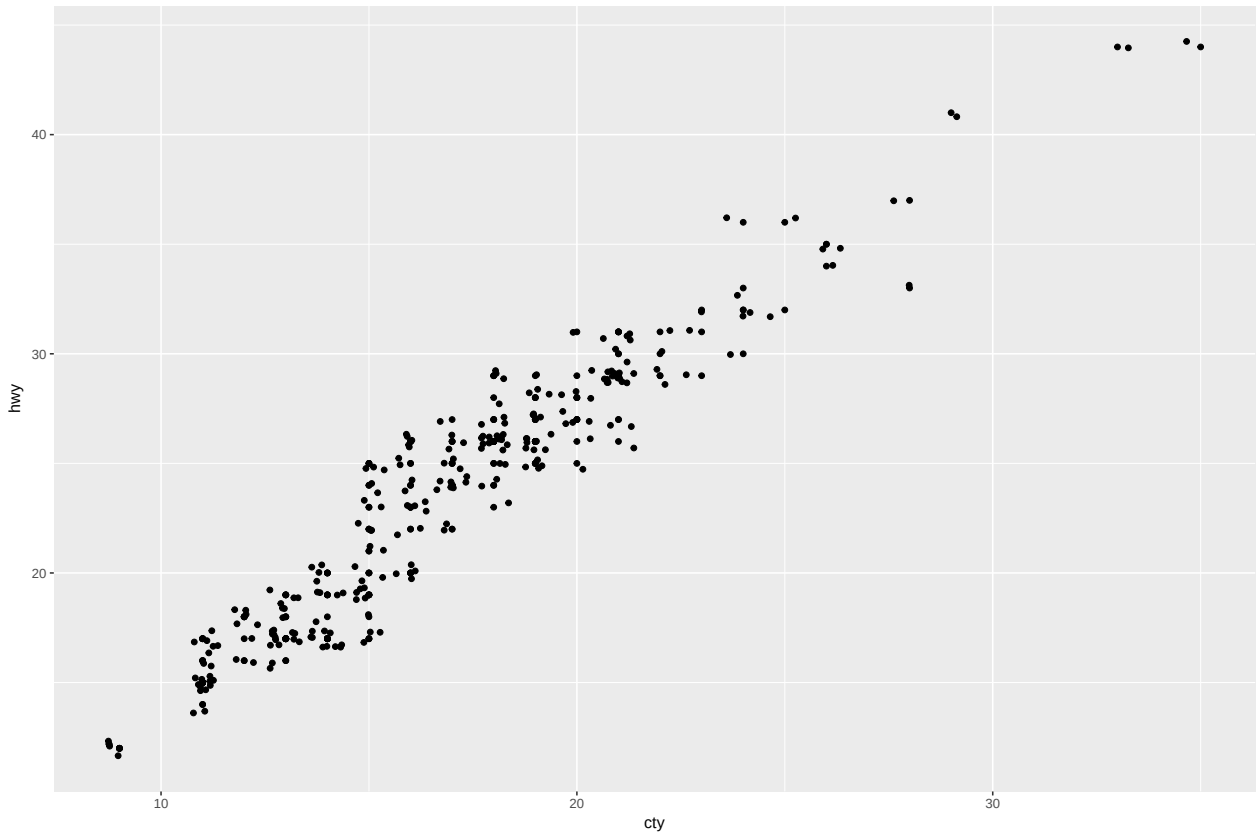
1. What is the problem with this plot? How could you improve it?

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



The above plot suffers from overplotting; it has overlapping data points. As a result, we cannot use the plot to see the true spread of the data. To improve the plot, we can add jitter. See the plot below.

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



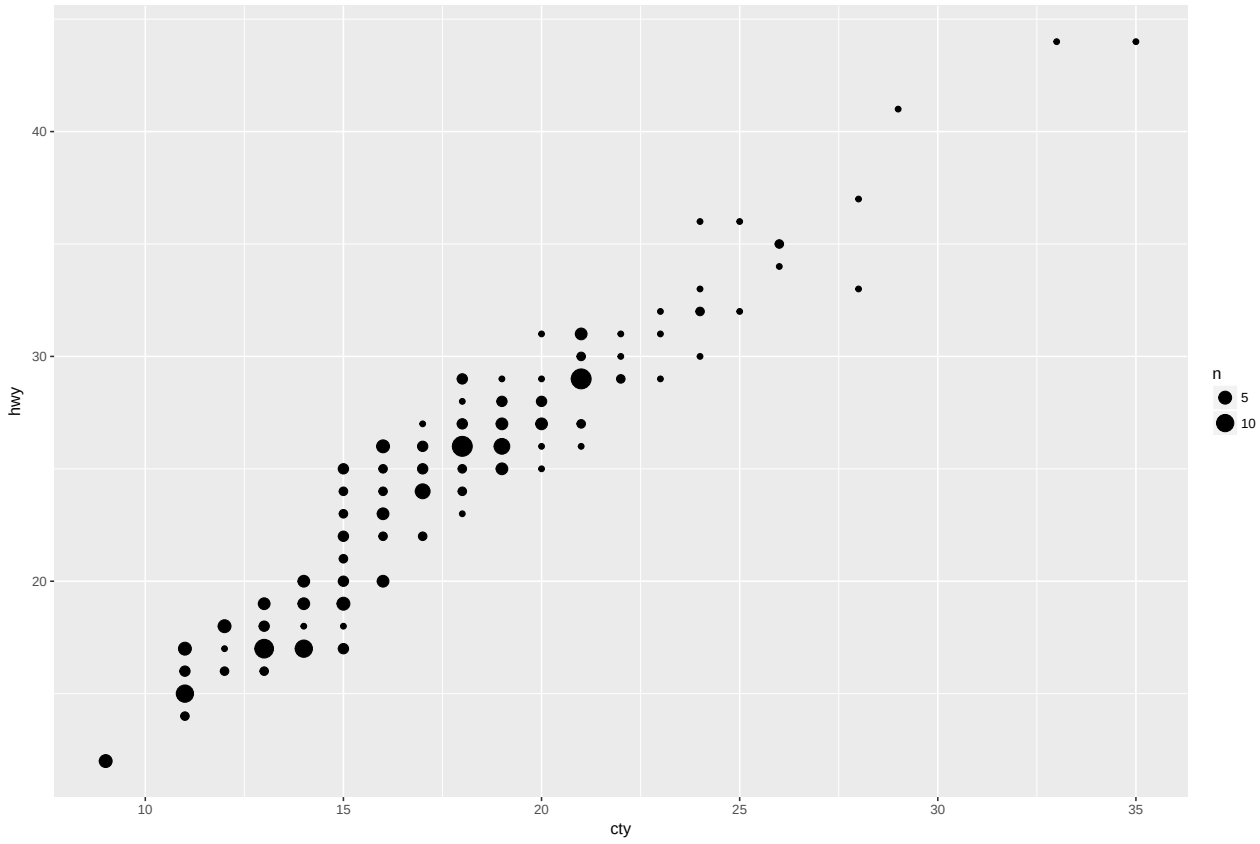
2. What parameters to geom_jitter() control the amount of jittering?

According to the documentation, `width` and `height` control the amount of jittering. See `?geom_jitter()` for more detail.

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

According to the documentation, `geom_count` "is a variant geom_point that counts the number of observations at each location, then maps the count to point area. It useful when you have discrete data and overplotting." The resultant plot looks like:

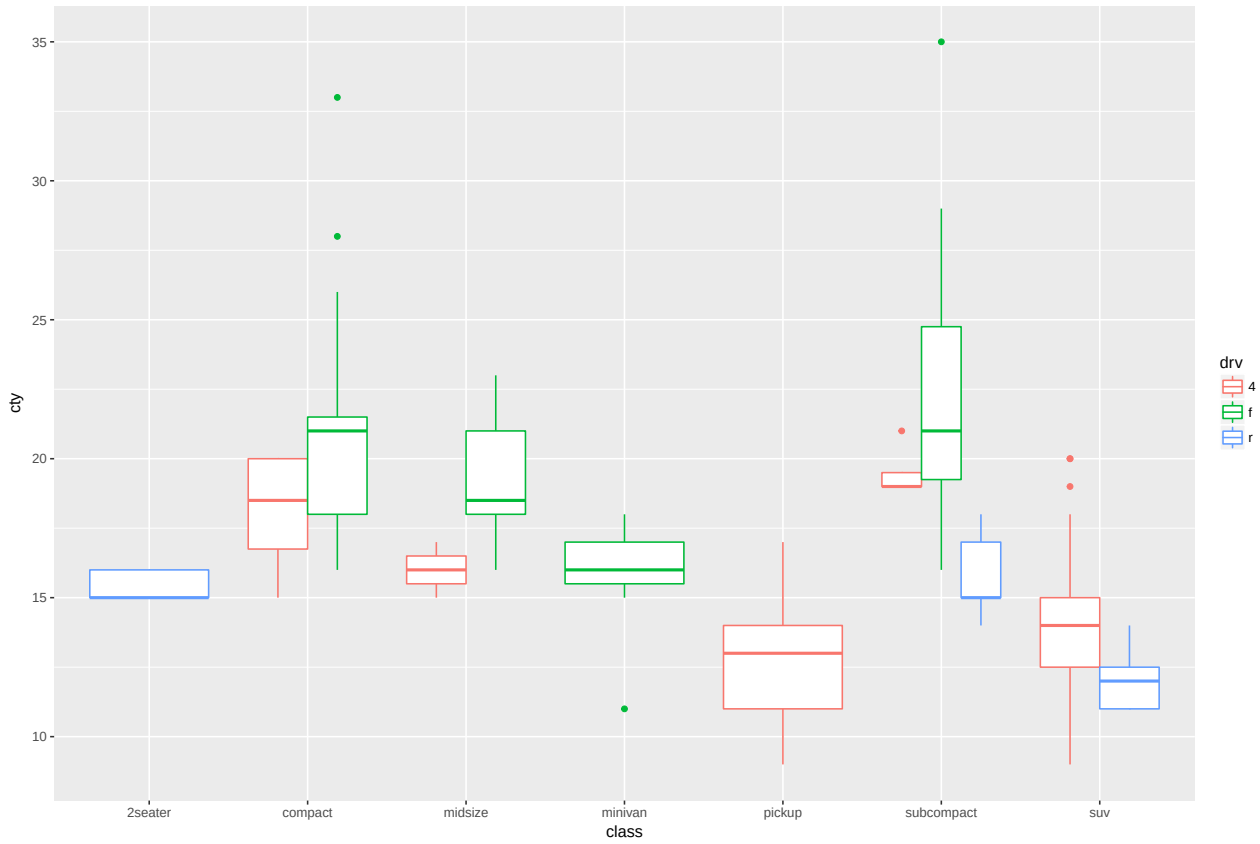
```
ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +
    geom_point() +
    geom_count()
```



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

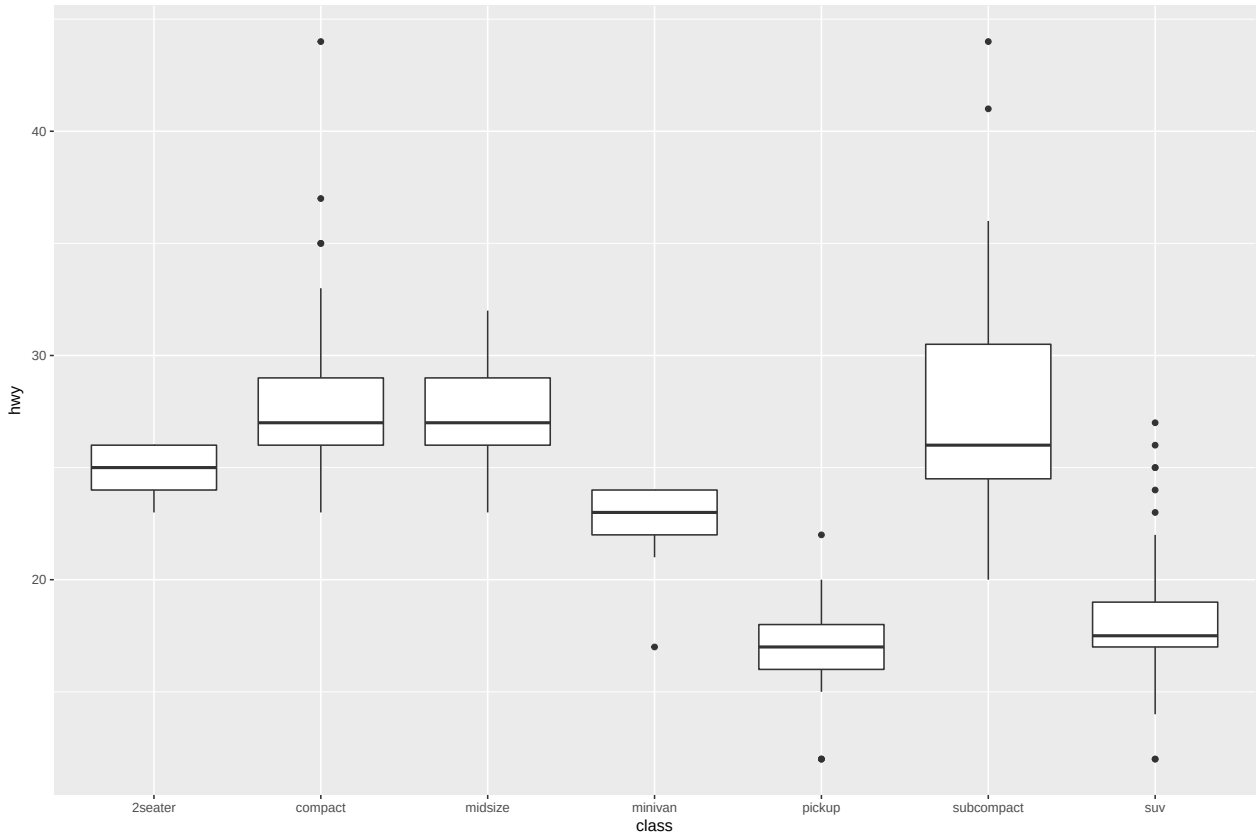
The default position adjustment for `geom_boxplot()` is dodge. Note that the geoms for the different drivetrain types are positioned next to one another in the plot, rather than overlapping.

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

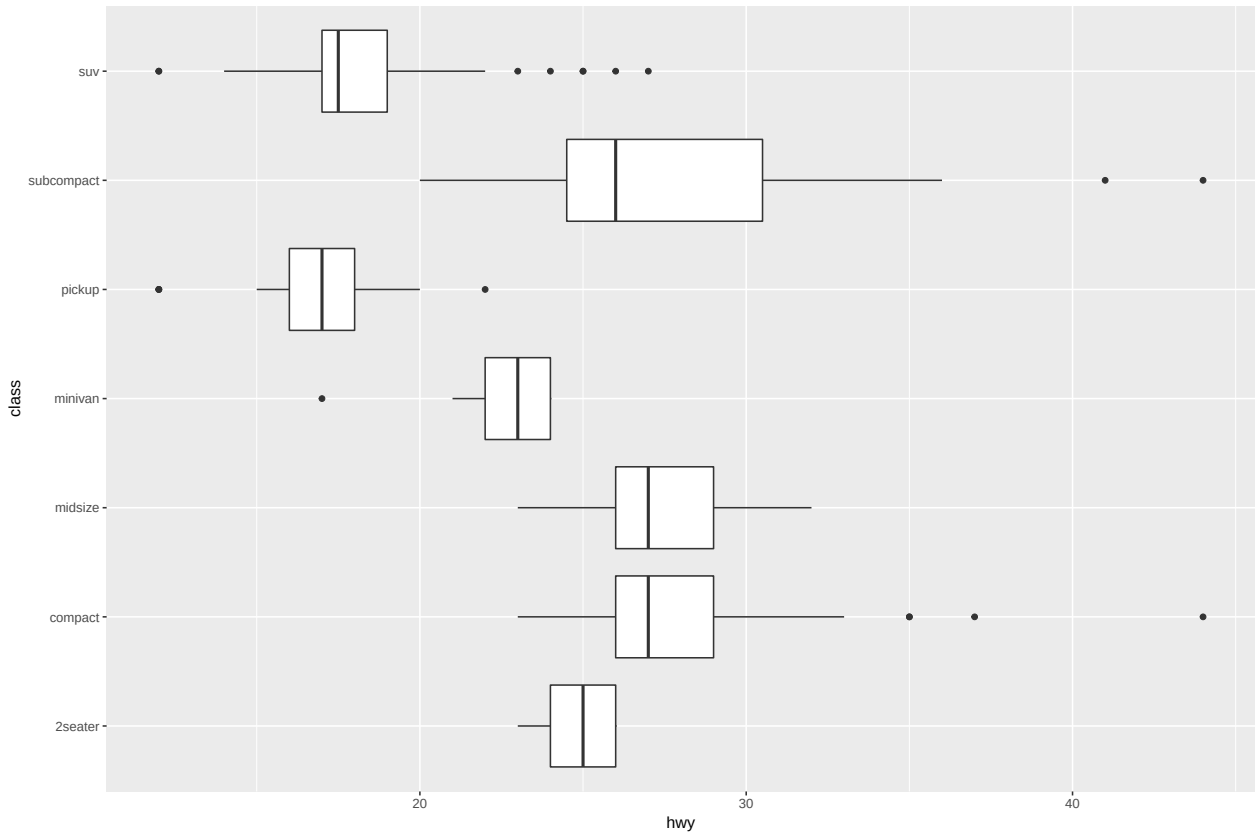


Coordinate systems

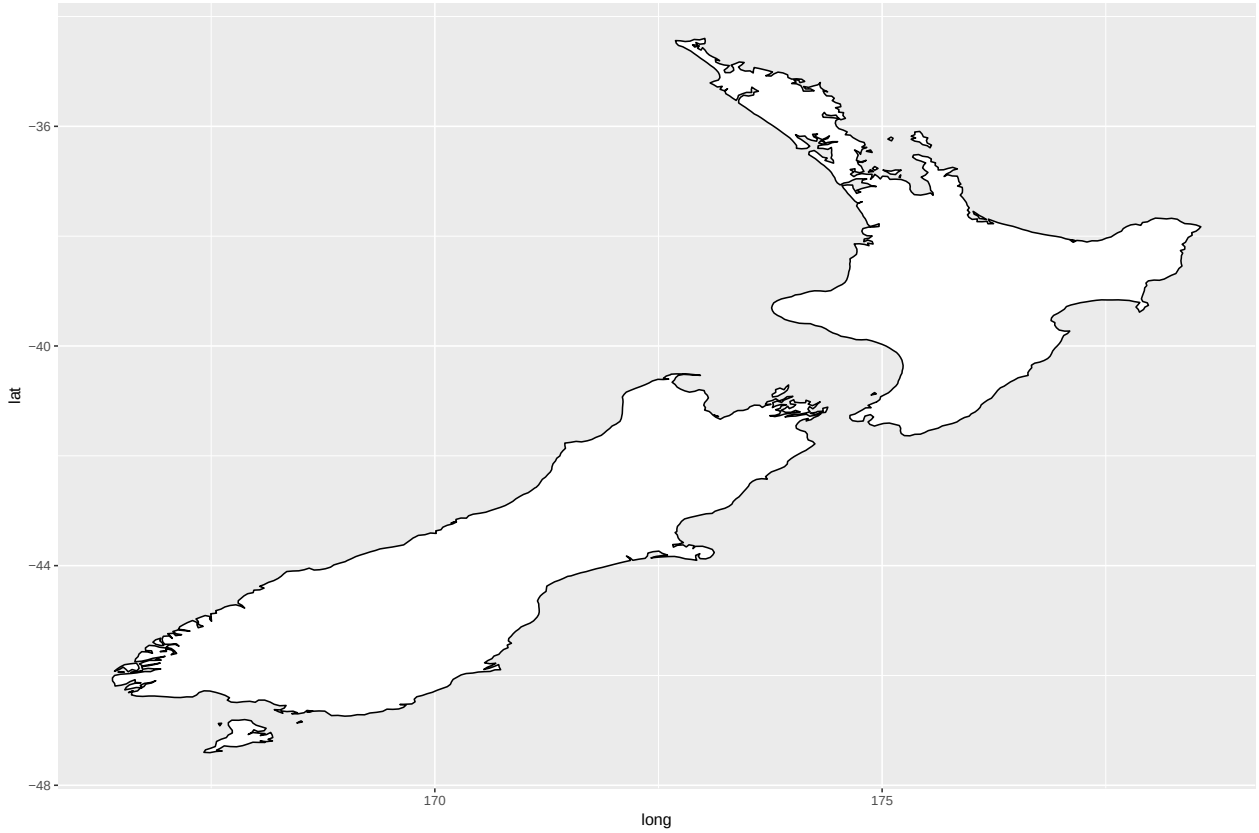
```
# Boxplot of highway miles per gallon by car class
ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +
  geom_boxplot()
```



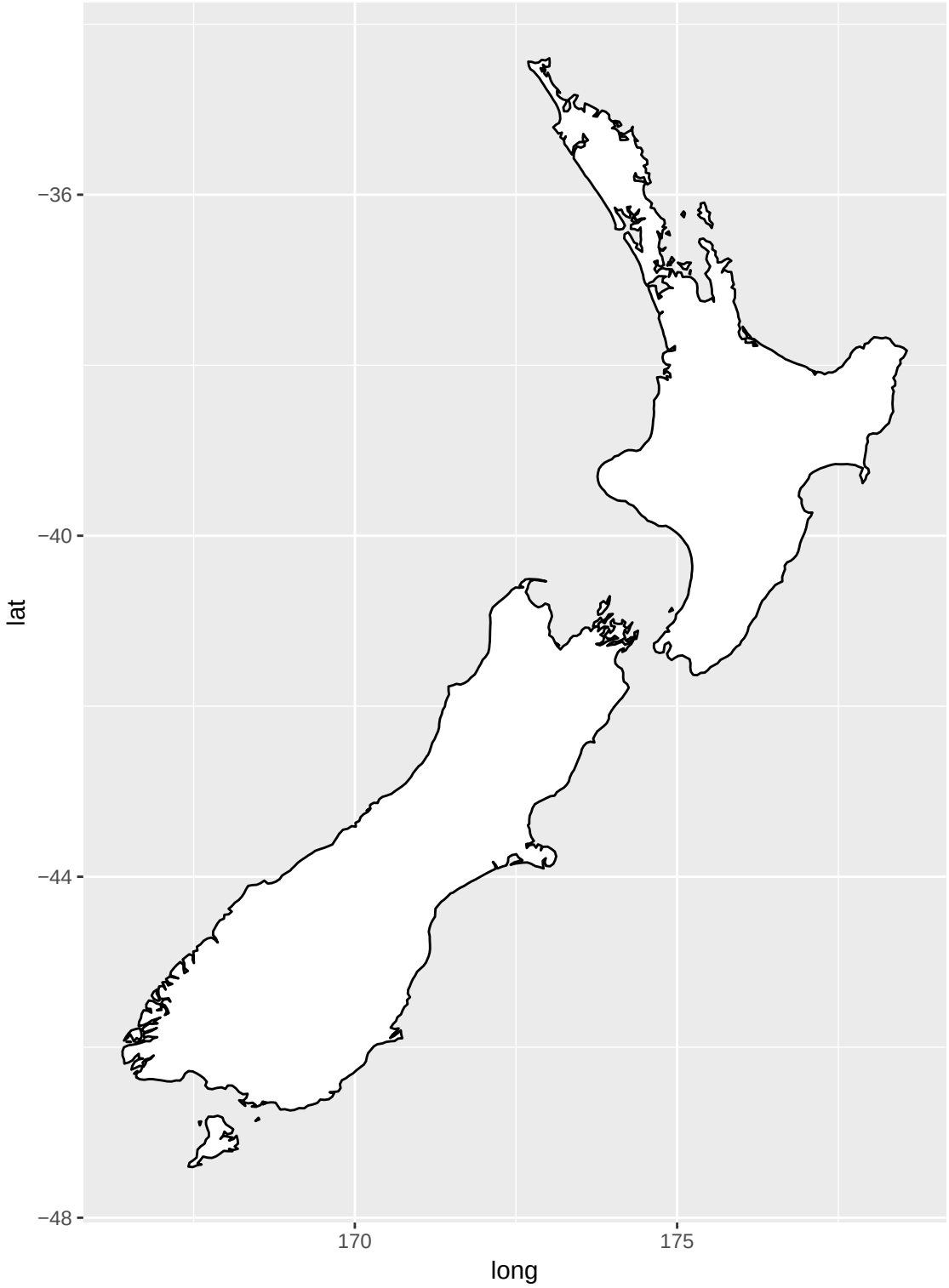

```
# Same boxplot with the x and y axes flipped
ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +
  geom_boxplot() +
  coord_flip()
```




```
# Create a data frame from a subset of the maps data for region `nz`
nz <- map_data("nz")

# Plot the map data
ggplot(nz, aes(long, lat, group = group)) +
  geom_polygon(fill = "white", colour = "black")
```

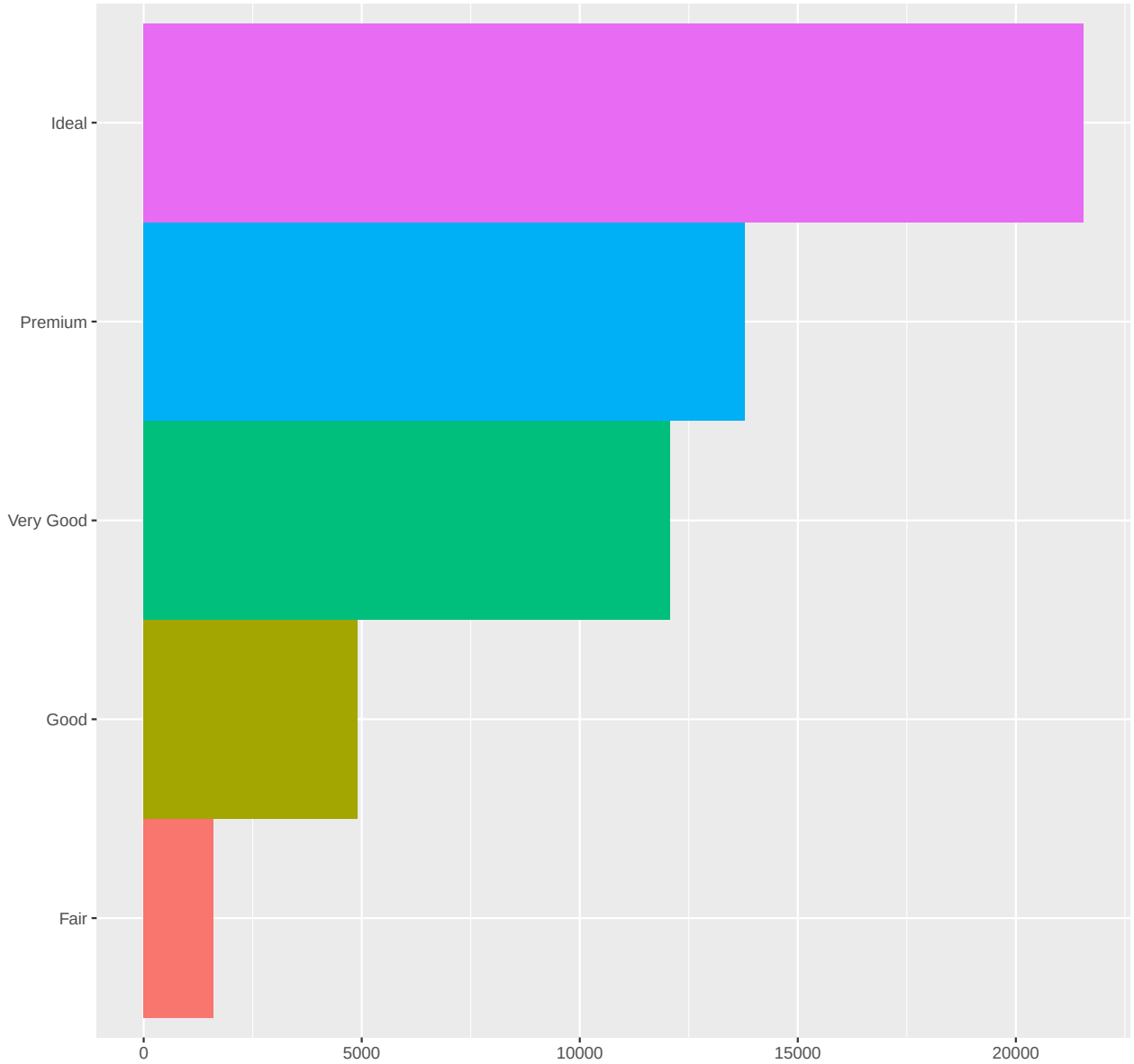



```
# Use `coord_quickmap()` to correctly set the aspect ratio for the map
ggplot(nz, aes(long, lat, group = group)) +
  geom_polygon(fill = "white", colour = "black") +
  coord_quickmap()
```

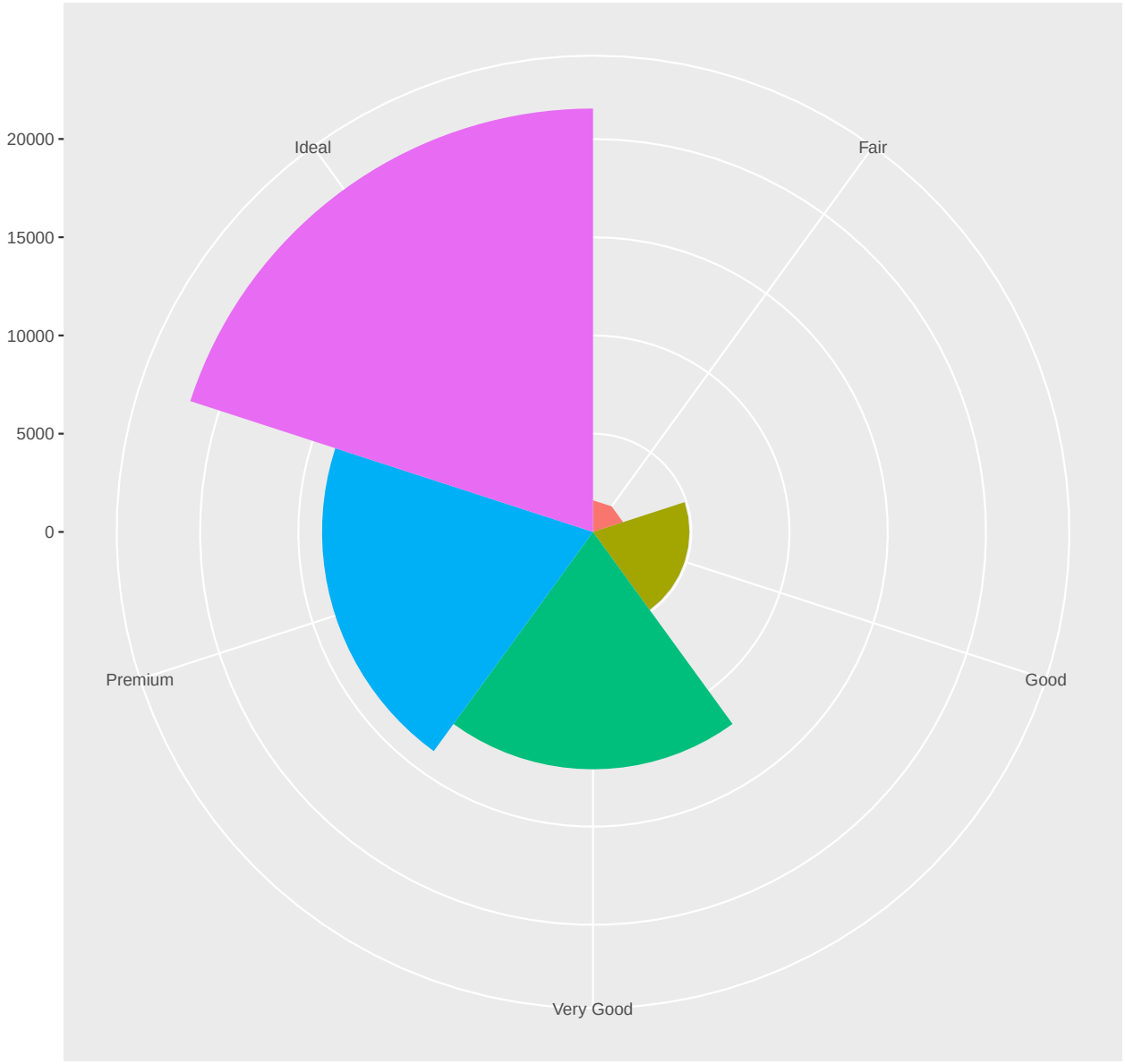



```
# Create a bar chart and save it to the variable `bar`
bar <- ggplot(data = diamonds) +
  geom_bar(
    mapping = aes(x = cut, fill = cut),
    show.legend = FALSE,
    width = 1
```

```
  ) +
  theme(aspect.ratio = 1) +
  labs(x = NULL, y = NULL)

# Flip the bar chart x and y axes and print it
bar + coord_flip()
```




```
# Create a Coxocomb chart from the bar chart by mapping the data to polar coordinates
bar + coord_polar()
```

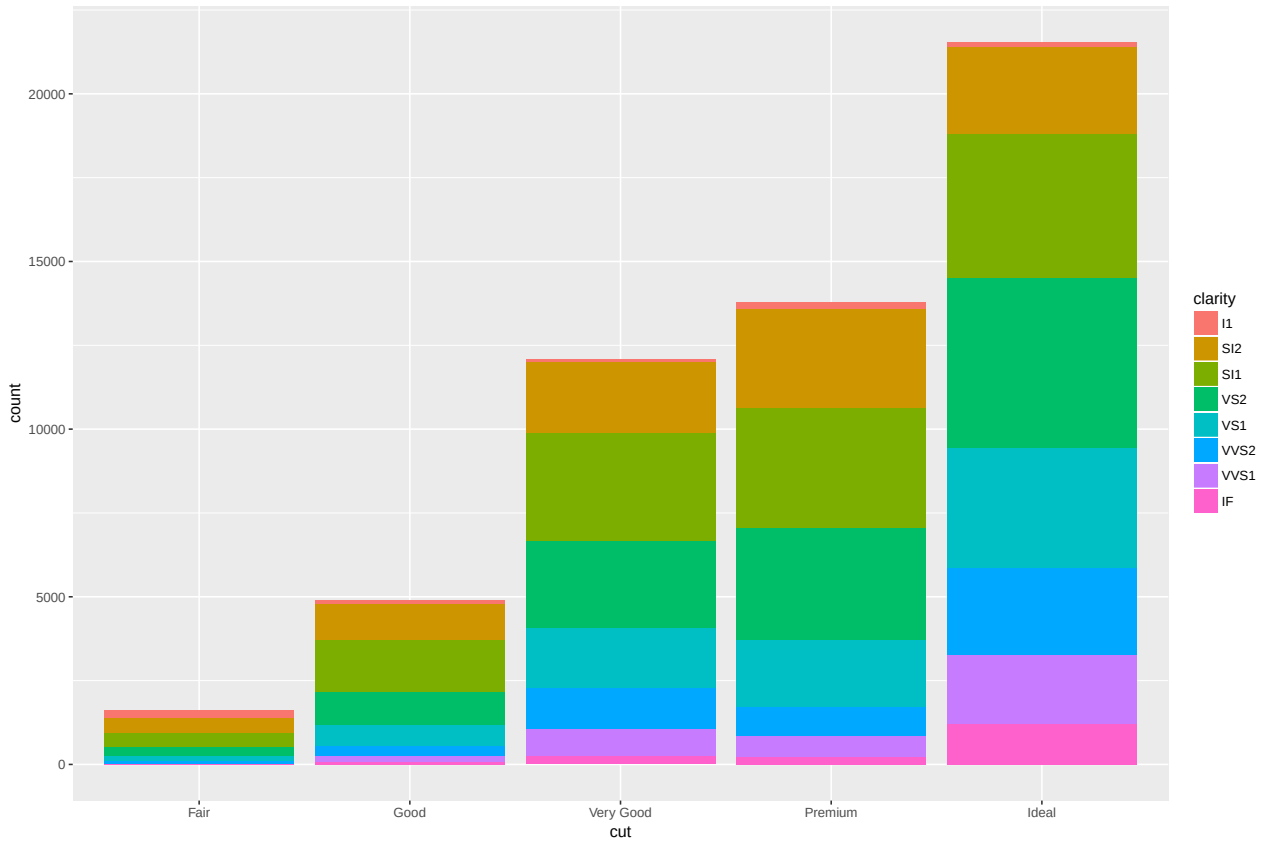


Exercises 3.9.1

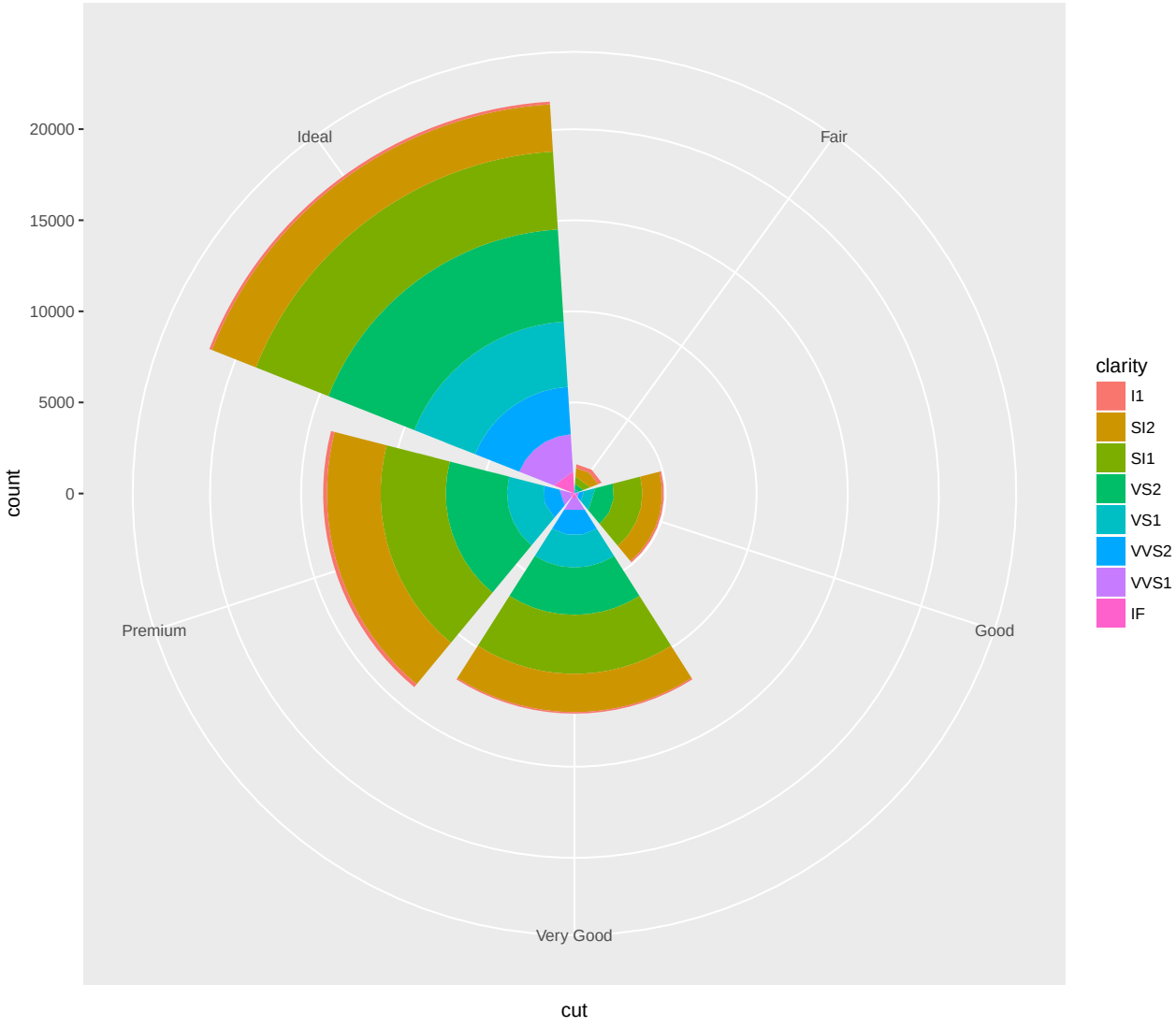
1. Turn a stacked bar chart into a pie chart using coord_polar().

Let's use our stacked bar plot from earlier and map it to polar coordinates.

```
# Stacked
ggplot(data=diamonds) +
    geom_bar(mapping = aes(x=cut, fill=clarity))
```




```
# Polar
ggplot(data=diamonds) +
    geom_bar(mapping = aes(x=cut, fill=clarity)) +
    coord_polar()
```



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

`labs()` allows the user to manually specify title, subtitle, axis, and legend labels of plots. See `?labs()`

3. What's the difference between coord_quickmap() and coord_map()?

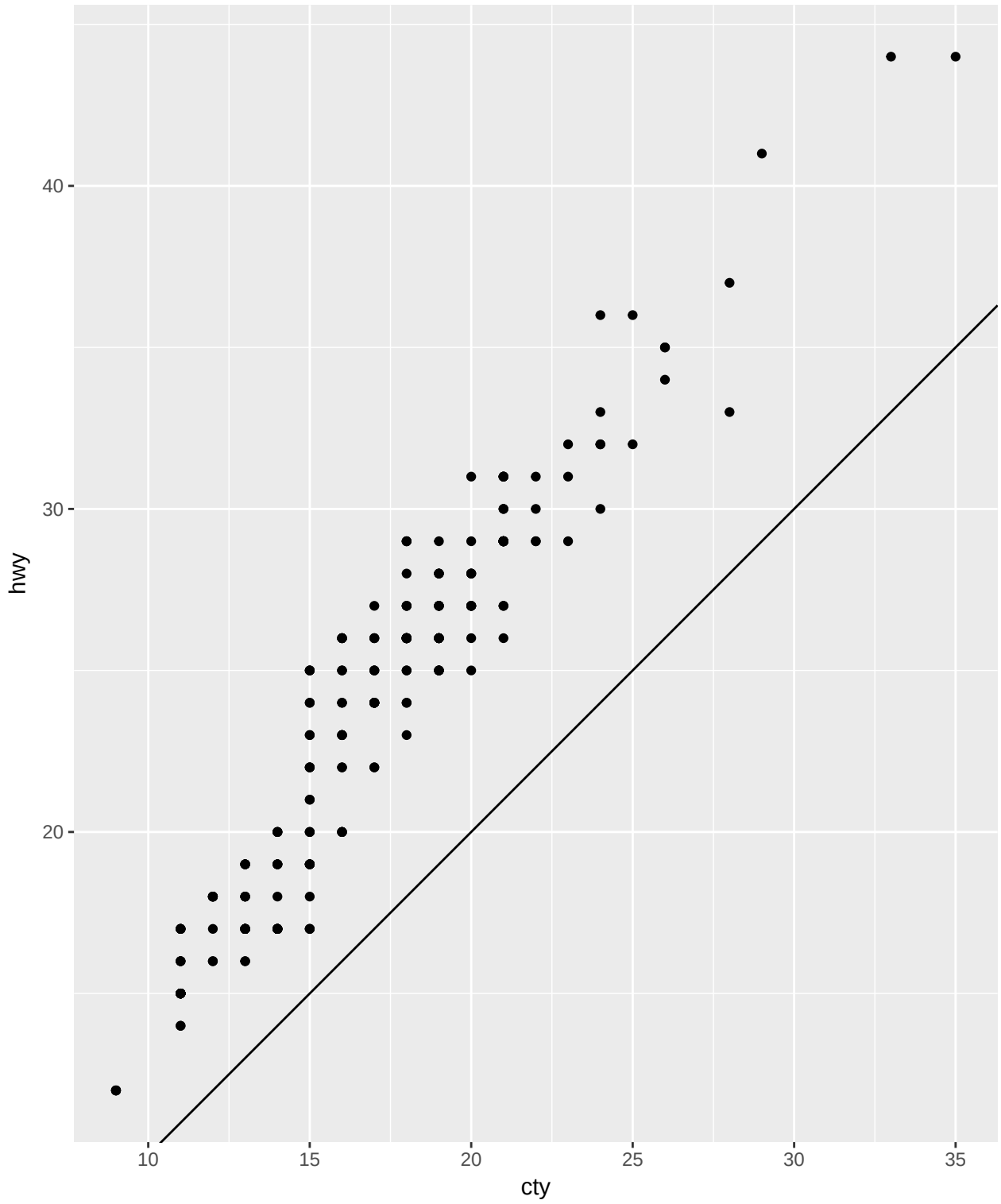
From the documentation:
"`coord_map` projects a portion of the earth, which is approximately spherical, onto a flat 2D plane using any projection defined by the mapproj package. Map projections do not, in general, preserve straight lines, so this requires considerable computation. `coord_quickmap` is a quick approximation that does preserve straight lines. It works best for smaller areas closer to the equator."

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

The relationship between city and highway mpg is positive and approximately linear such that the higher a car's city mpg is, the higher it's highway mpg tends to be. `coord_fixed()` is important because forces the units on the x and y axes to coform making them directly comparable. `geom_abline()` adds reference

lines to the plot to which we can compare the data. The default line has an intercept of 0 and slope of 1. Comparing our data to the line, we see that highway mpg is slightly higher than city mpg on average.

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


40
30
20
hwy
10
15
20
25
30
35
cty

The layered grammar of graphics

Code template with position adjustments, stats, coordinate systems, and faceting.

```
ggplot(data = <DATA>) +     <GEOM_FUNCTION>(        mapping = aes(<MAPPINGS>),        stat
= <STAT>,         position = <POSITION>     ) +     <COORDINATE_FUNCTION> +     <FACET_FUNCTION>
```

Steps for transforming data into a plot