MB5370 Module 04. Workshop 1 - Introduction

Shreyash G Bhandary

2024-05-15

```
#tinytex::install_tinytex()
```

Install and load tidyverse packages

```
## install.packages("tidyverse") # Delete this line once installed
library("tidyverse")
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4          v readr     2.1.5
## v forcats   1.0.0          v stringr   1.5.1
## v ggplot2   3.5.1.9000     v tibble    3.2.1
## v lubridate 1.9.3          v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

Load the data (mpg is built into ggplot2)

```
data(mpg)
```

Quick data checks

```
head(mpg)
```

```
## # A tibble: 6 x 11
##   manufacturer model displ  year   cyl trans      drv     cty   hwy fl    class
##   <chr>        <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr> <chr>
## 1 audi         a4      1.8  1999     4 auto(15)   f        18    29 p     compa~
## 2 audi         a4      1.8  1999     4 manual(m5) f        21    29 p     compa~
## 3 audi         a4      2    2008     4 manual(m6) f        20    31 p     compa~
```

```
## 4 audi         a4      2    2008     4 auto(av)   f        21    30 p     compa~
## 5 audi         a4      2.8  1999     6 auto(l5)   f        16    26 p     compa~
## 6 audi         a4      2.8  1999     6 manual(m5) f        18    26 p     compa~
```

```
glimpse(mpg)
```

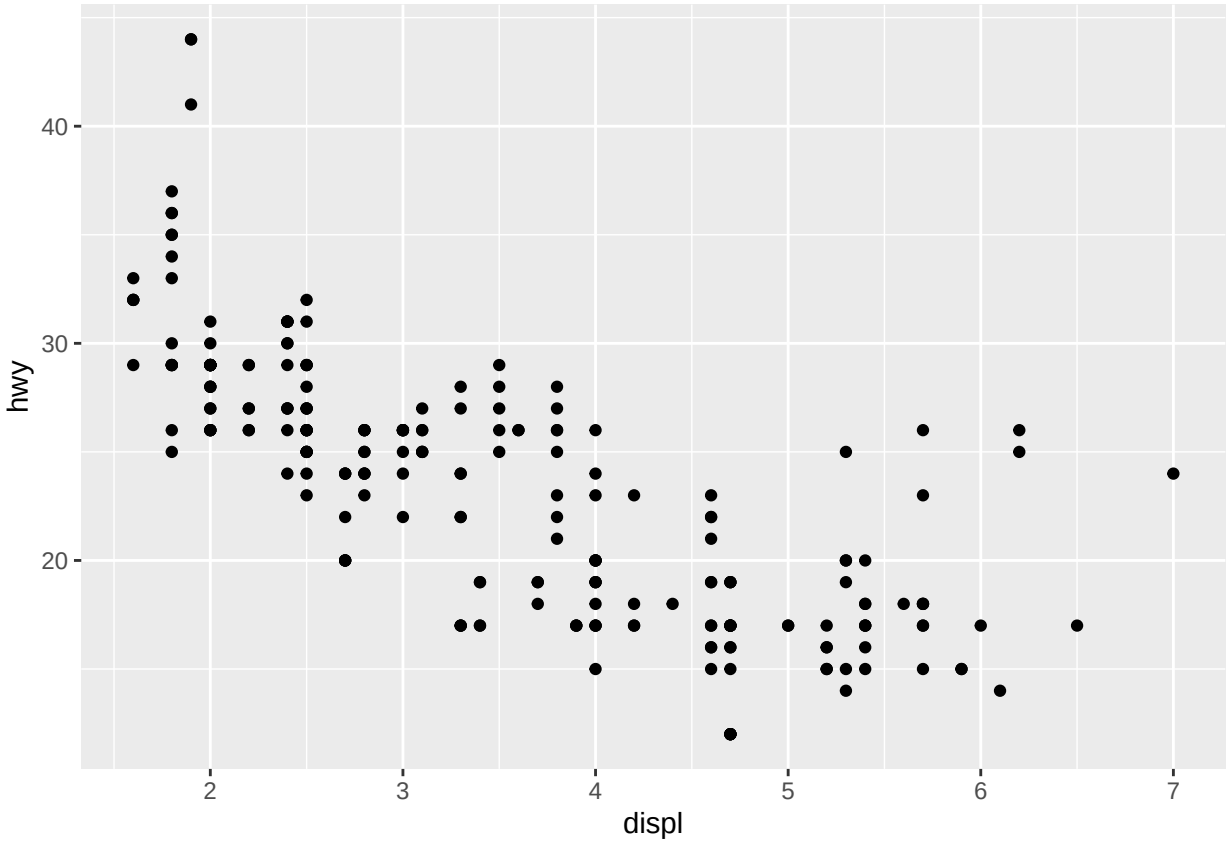
```
## Rows: 234
## Columns: 11
## $ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi", "audi", "~
## $ model        <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro", "~
## $ displ        <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, 2.0, 2.~
## $ year         <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, 200~
## $ cyl          <int> 4, 4, 4, 4, 6, 6, 6, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 8, 8, ~
## $ trans        <chr> "auto(l5)", "manual(m5)", "manual(m6)", "auto(av)", "auto~
## $ drv          <chr> "f", "f", "f", "f", "f", "f", "f", "4", "4", "4", "4", "4~
## $ cty          <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17, 1~
## $ hwy          <int> 29, 29, 31, 30, 26, 26, 27, 26, 25, 28, 27, 25, 25, 25, 2~
## $ fl           <chr> "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p~
## $ class        <chr> "compact", "compact", "compact", "compact", "compact", "c~
```

```
summary(mpg)
```

```
##  manufacturer          model               displ            year
##  Length:234         Length:234         Min.   :1.600   Min.   :1999
##  Class :character   Class :character   1st Qu.:2.400   1st Qu.:1999
##  Mode  :character   Mode  :character   Median :3.300   Median :2004
##                                        Mean   :3.472   Mean   :2004
##                                        3rd Qu.:4.600   3rd Qu.:2008
##                                        Max.   :7.000   Max.   :2008
##       cyl           trans               drv                 cty
##  Min.   :4.000   Length:234         Length:234         Min.   : 9.00
##  1st Qu.:4.000   Class :character   Class :character   1st Qu.:14.00
##  Median :6.000   Mode  :character   Mode  :character   Median :17.00
##  Mean   :5.889                                         Mean   :16.86
##  3rd Qu.:8.000                                         3rd Qu.:19.00
##  Max.   :8.000                                         Max.   :35.00
##       hwy             fl                class
##  Min.   :12.00   Length:234         Length:234
##  1st Qu.:18.00   Class :character   Class :character
##  Median :24.00   Mode  :character   Mode  :character
##  Mean   :23.44
##  3rd Qu.:27.00
##  Max.   :44.00
```

Create first plot

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



The plot shows a negative relationship between engine size (displ) and fuel efficiency (hwy). In other words, cars with big engines use more fuel. What does this say about fuel efficiency and engine size?

The fuel efficiency and engine size are inversely proportional to each other. Cars with larger engine size will have lower engine efficiency.

Understanding grammar of graphics

What happens if we just try to run ggplot on its own?

```
ggplot()
```

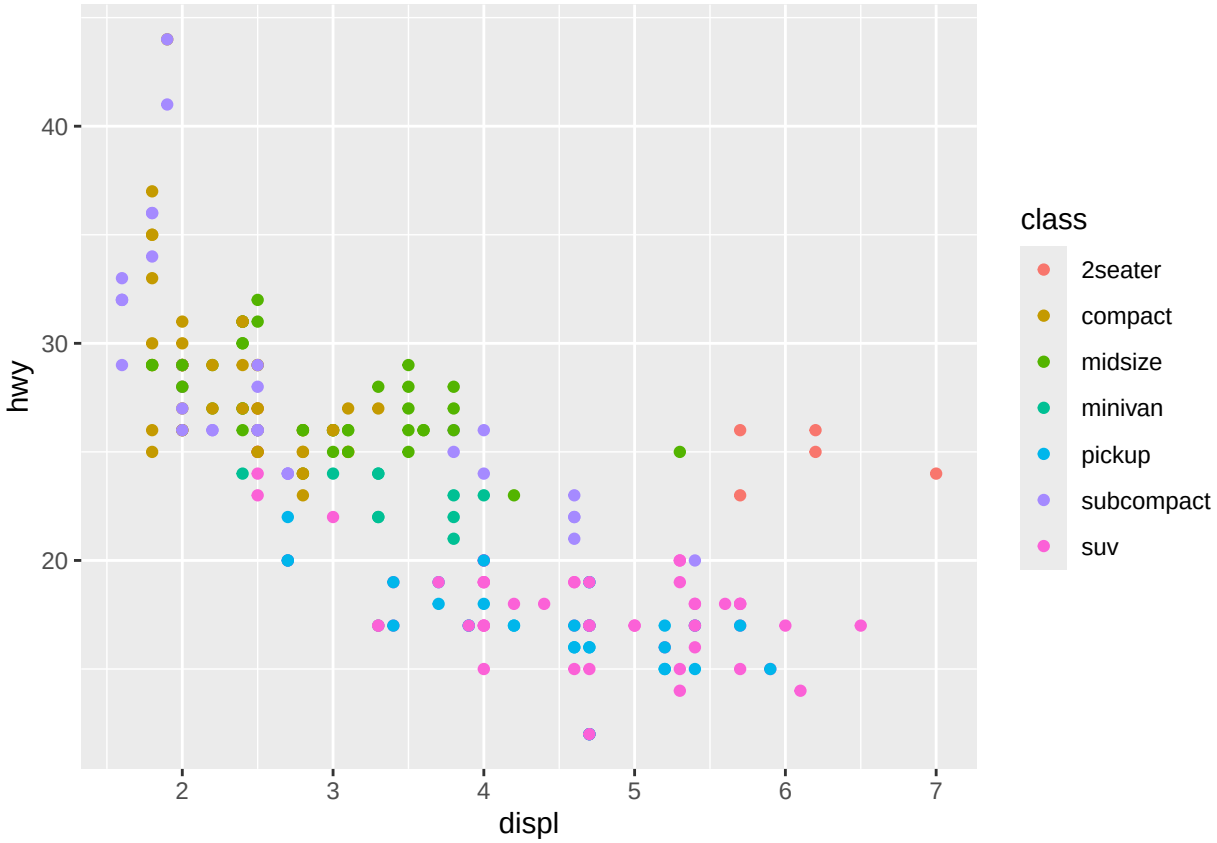
```
# Or with just the data specified?
ggplot(data = mpg)
```

We get a blank graph with no defined X or Y axis without any data.

We need to map some aesthetics!!

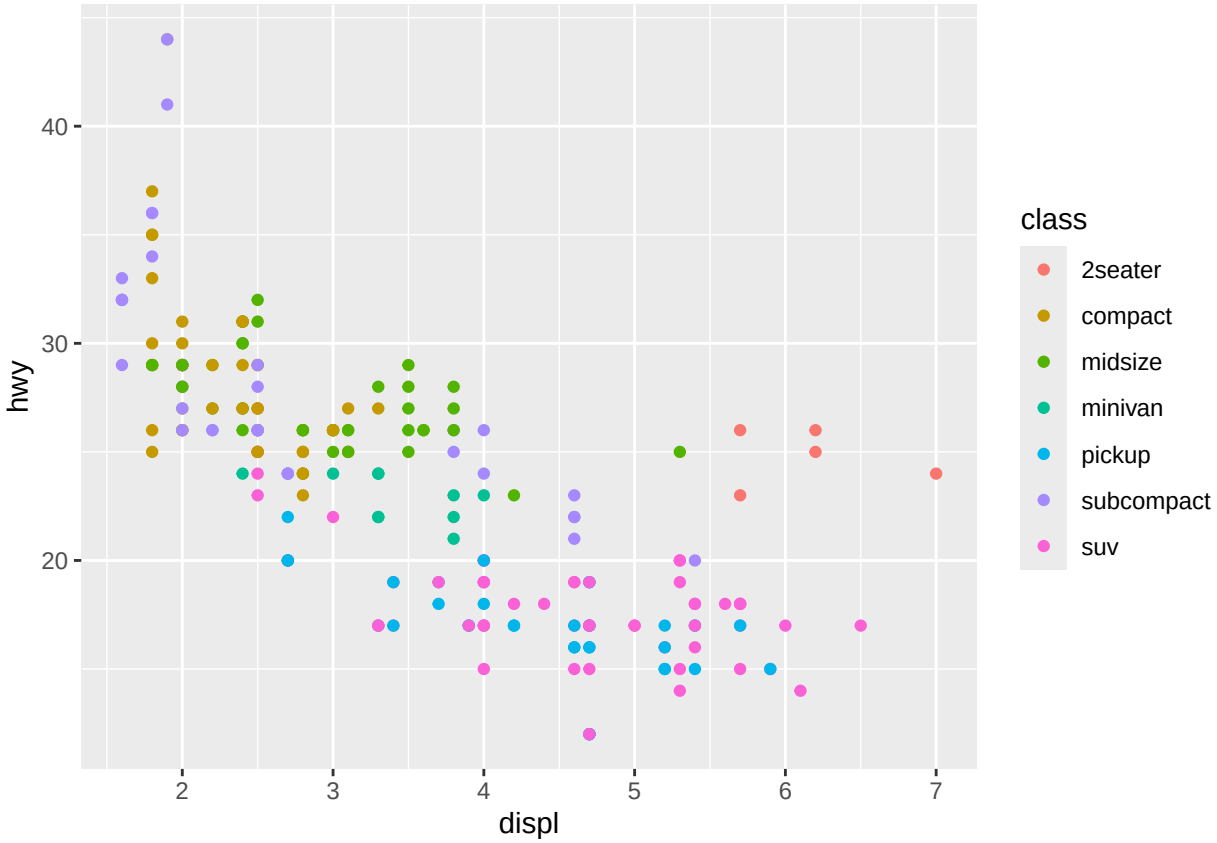
When you're creating a plot, you essentially need two attributes of a plot: a geom and aesthetics.

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



Change point colour by class:

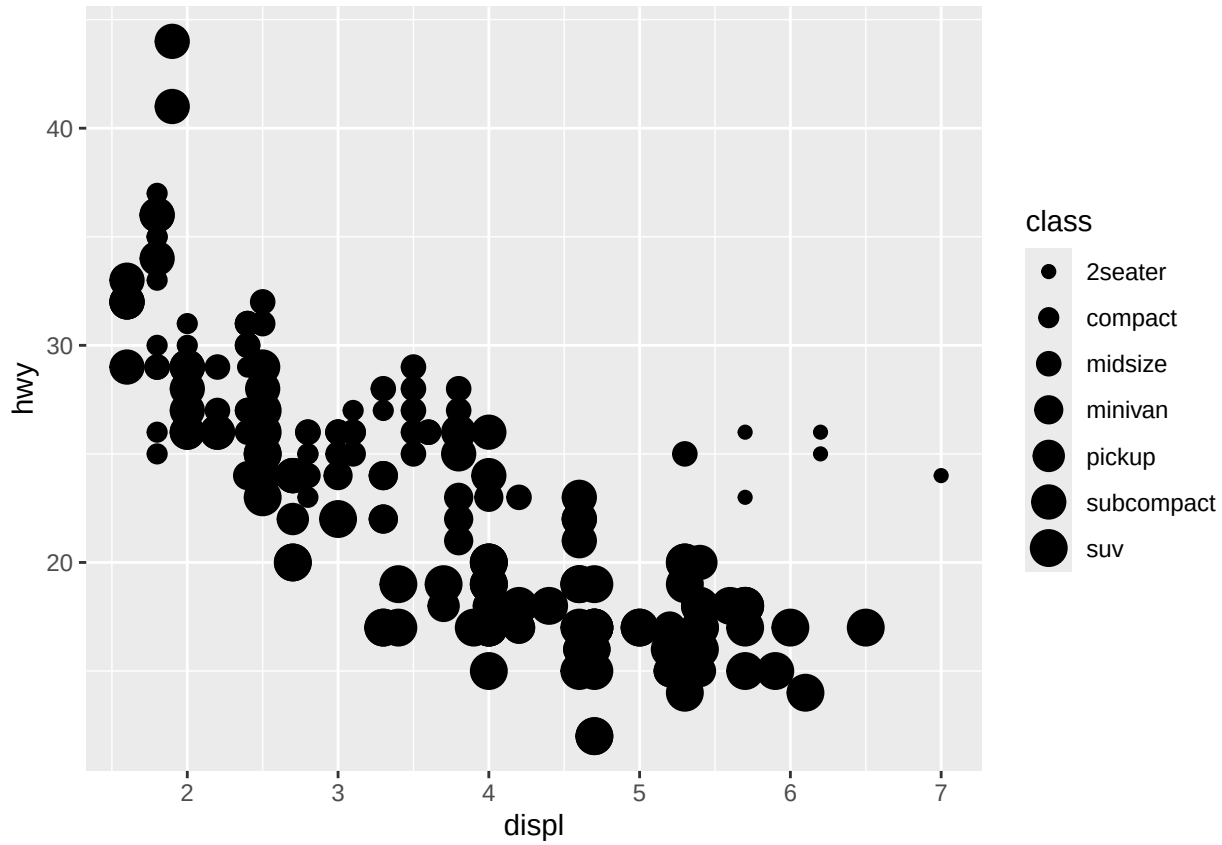
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, colour = class))
```



Change point size by class:

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

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

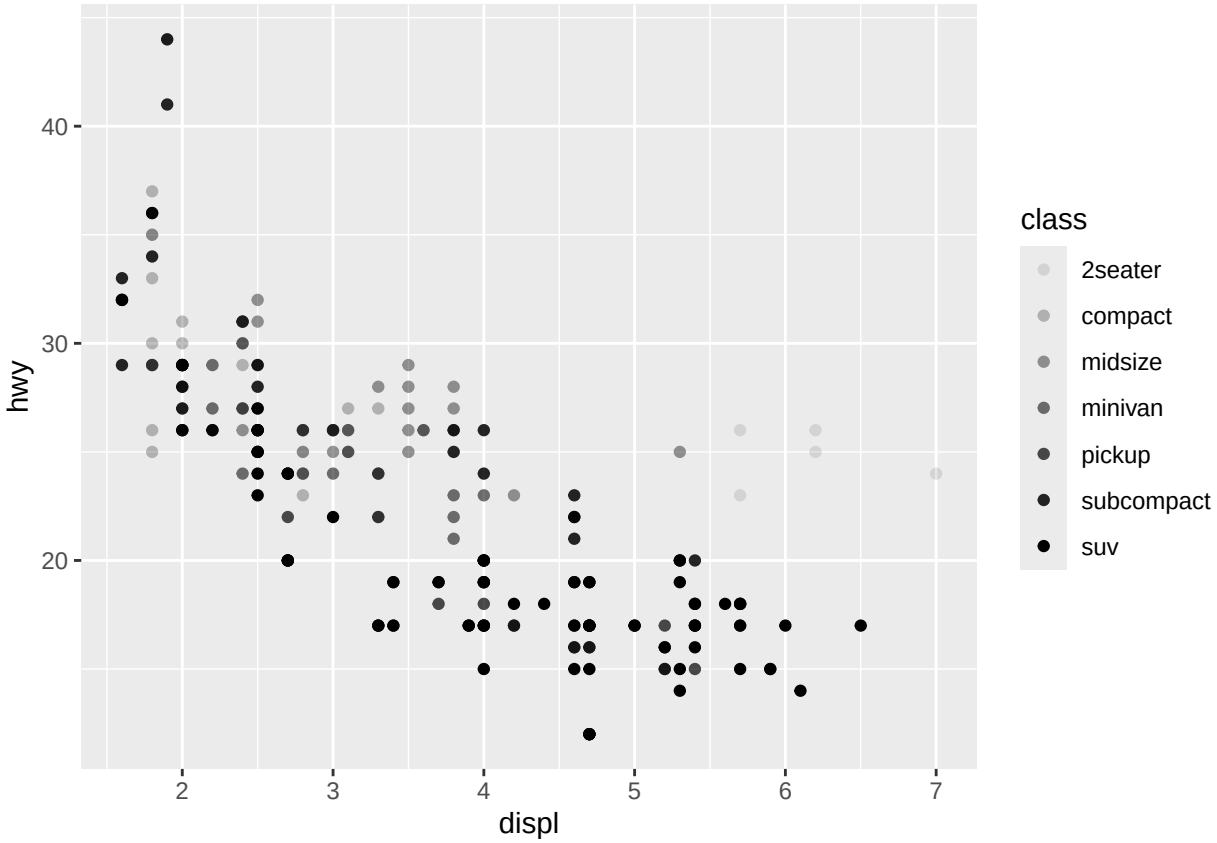


Note the warning!!!

Change transparency (alpha) by class

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

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



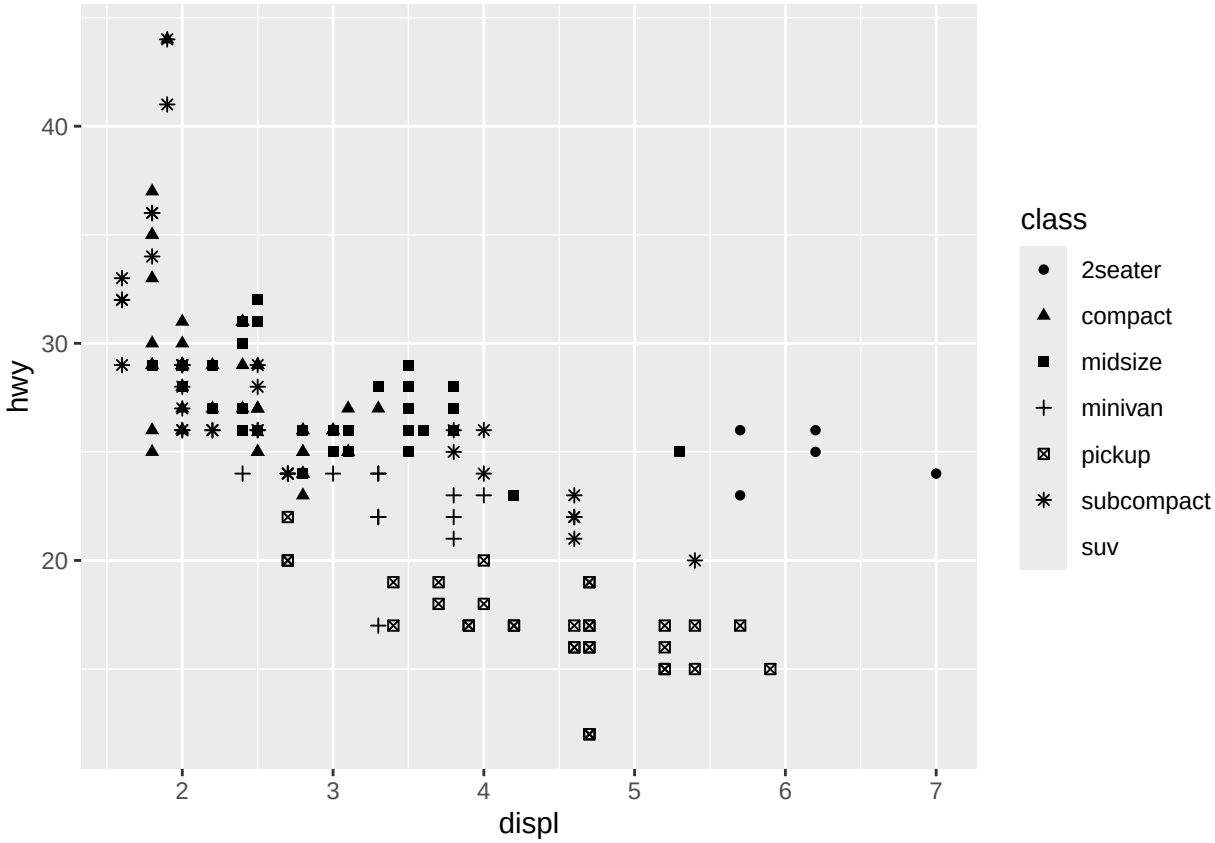
Another warning!! Question: When would using alpha (or size be appropriate??)

Change point shape by class:

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

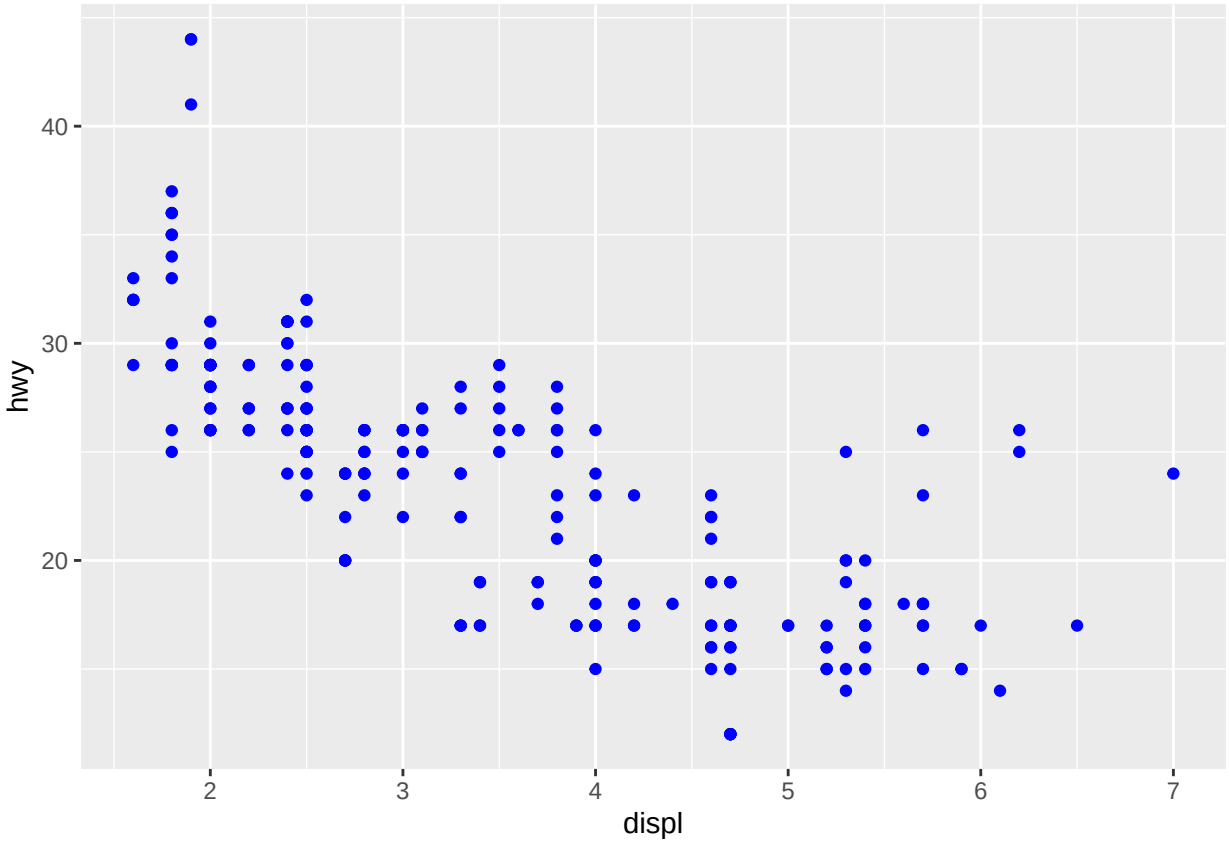
```
## Warning: The shape palette can deal with a maximum of 6 discrete values because more
## than 6 becomes difficult to discriminate
## i you have requested 7 values. Consider specifying shapes manually if you need
##   that many have them.
```

```
## Warning: Removed 62 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



Make all points blue

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



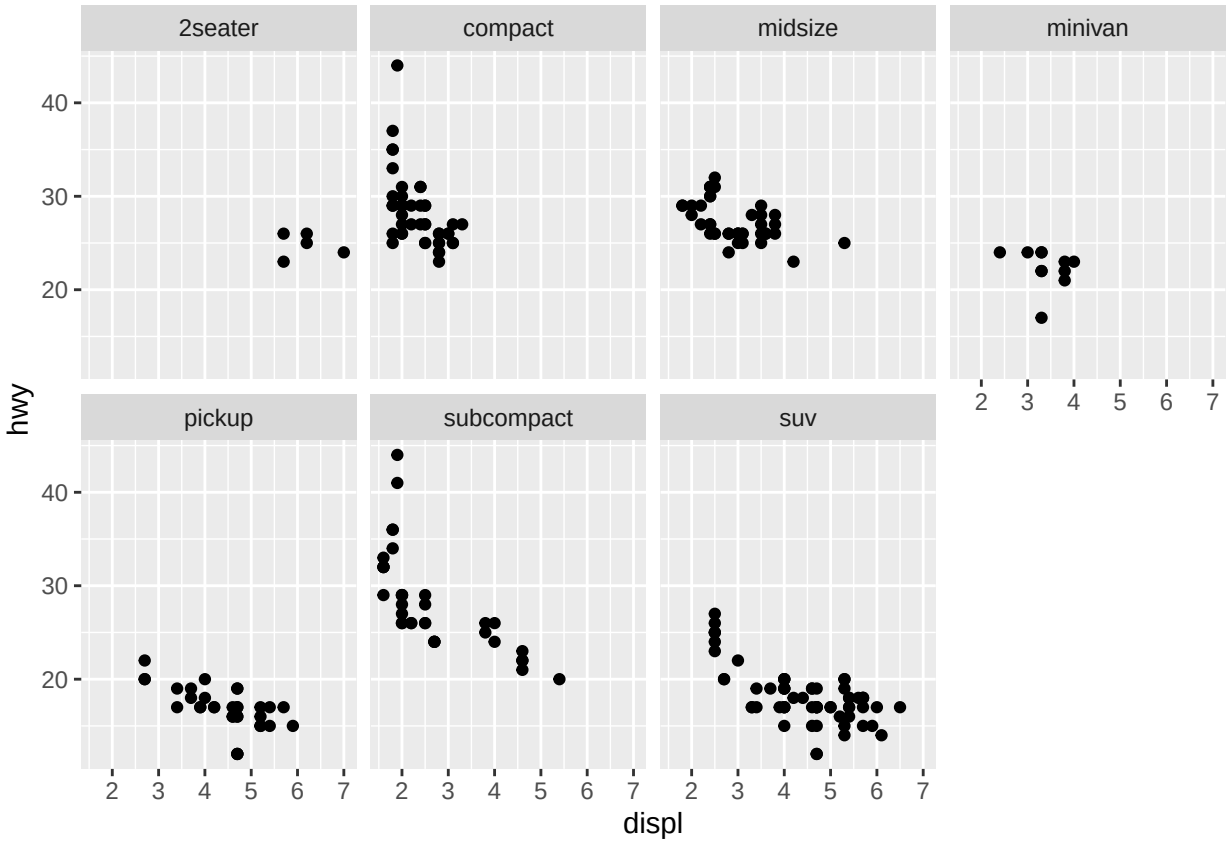
Troubleshooting

Faceting

facet_wrap

Split the plot out by car type (class)

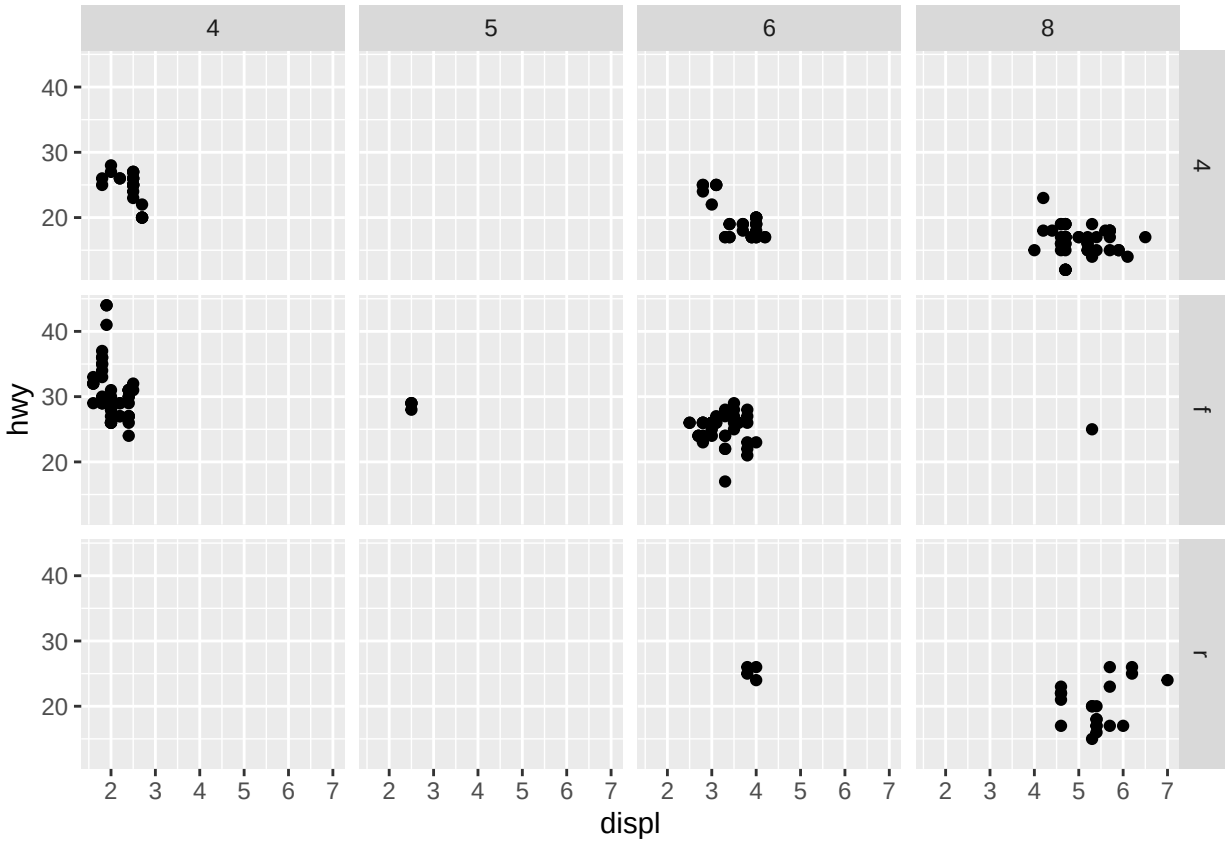
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_wrap(~ class, nrow = 2)
```



facet_grid

A separate facet for each combination of drive-type (e.g. 4WD) * number of cylinders

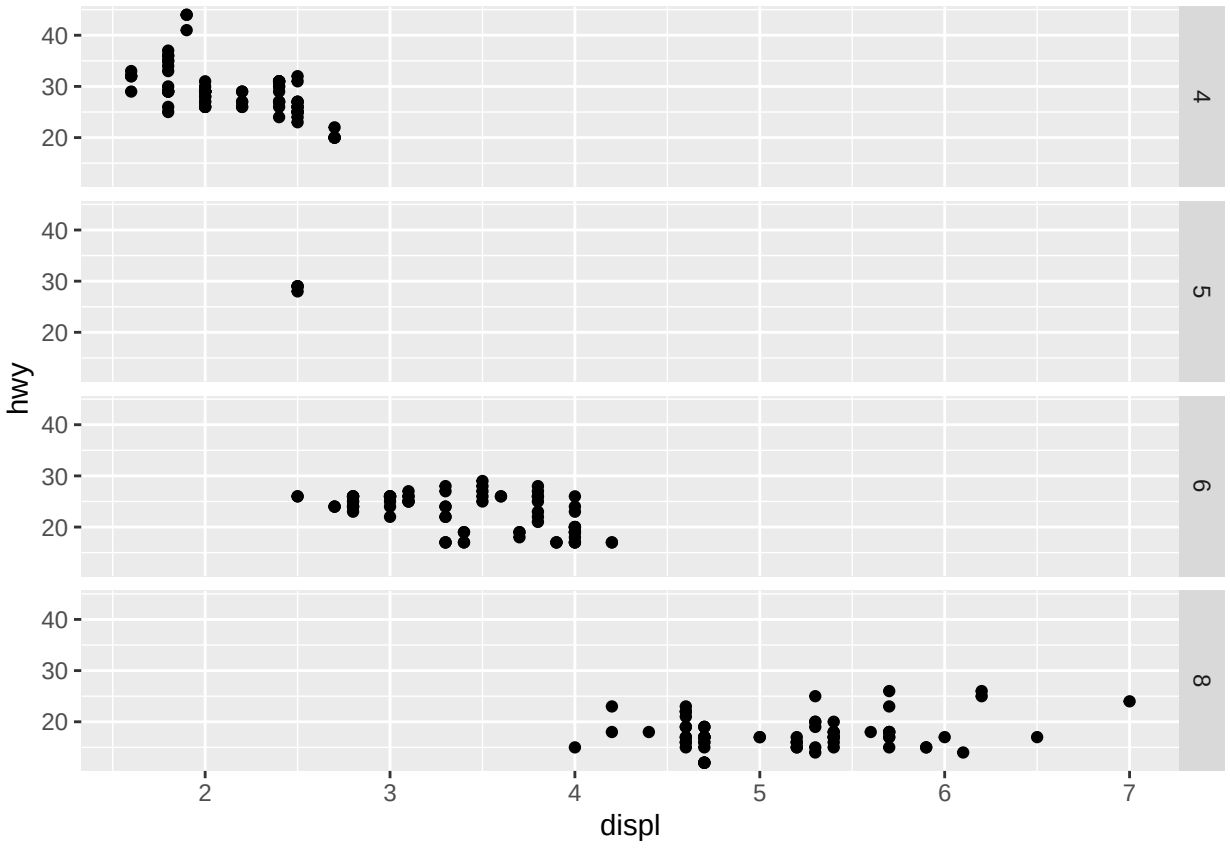
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_grid(drv ~ cyl)
```



Note that there are no occurrences of 5 cylinder 4WDs OR RWD vehicles!

facet_grid by just row (or column)

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

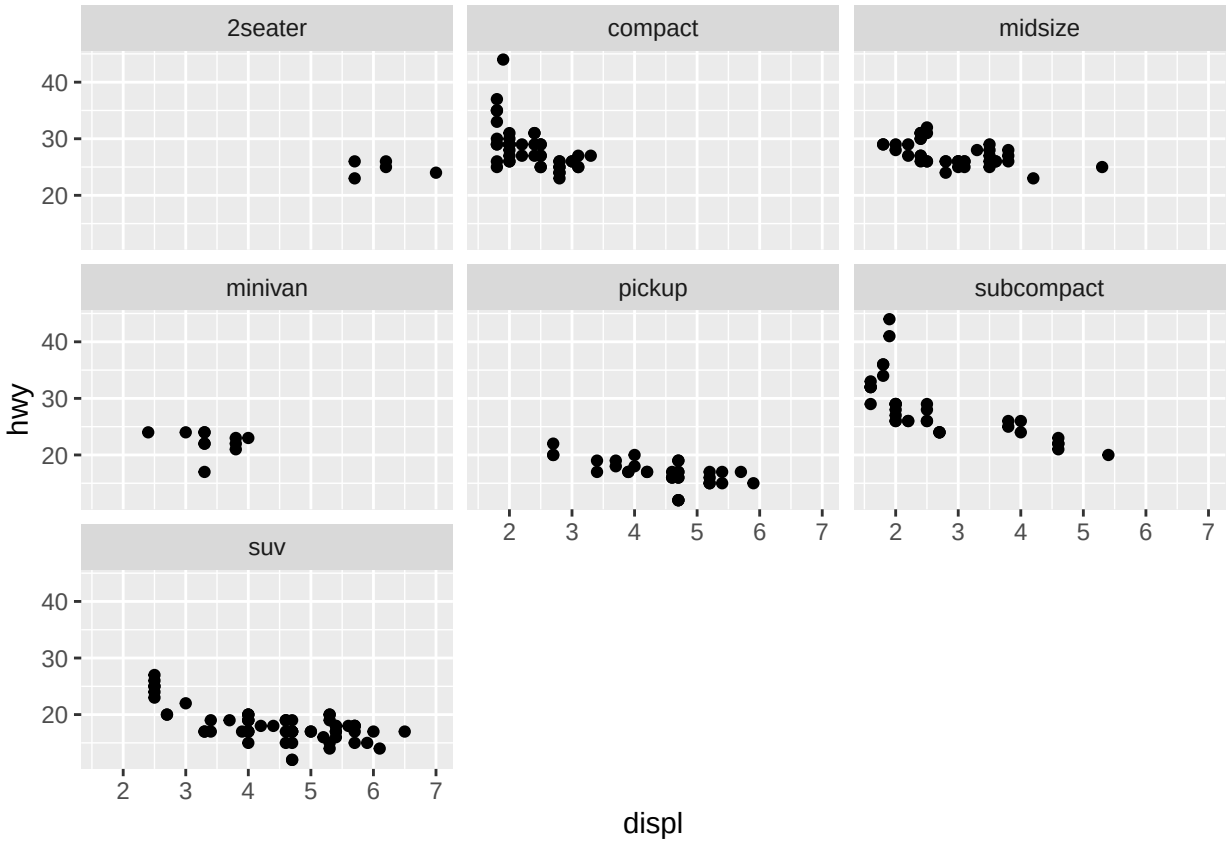



```
  #facet_grid(. ~ cyl) # Alternatively
```

Exercise:

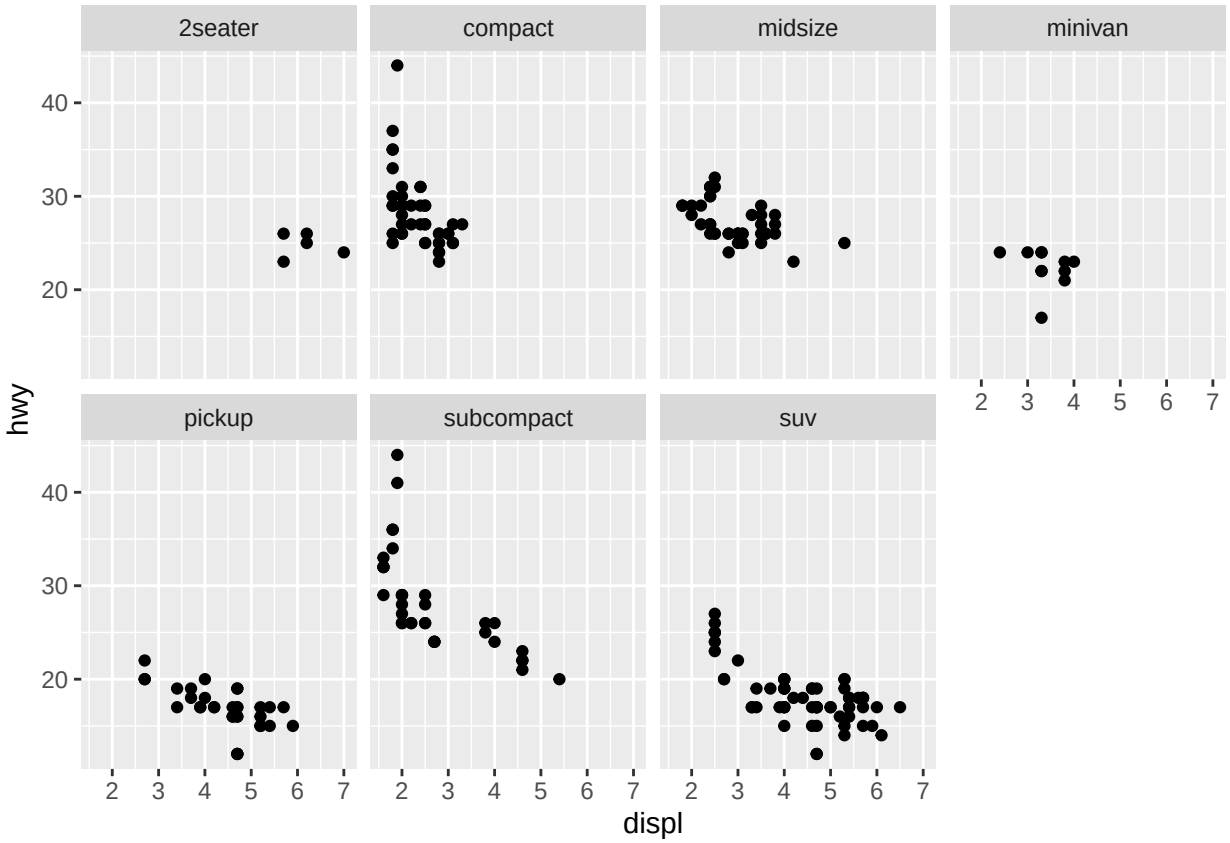
Read ?facet_wrap. What does nrow do? What does ncol do? What other options control the layout of the individual panels?

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




```
#nrow is used to define the no. of rows that you want in the facet wrap output.
```

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_wrap(~ class, ncol = 4)
```

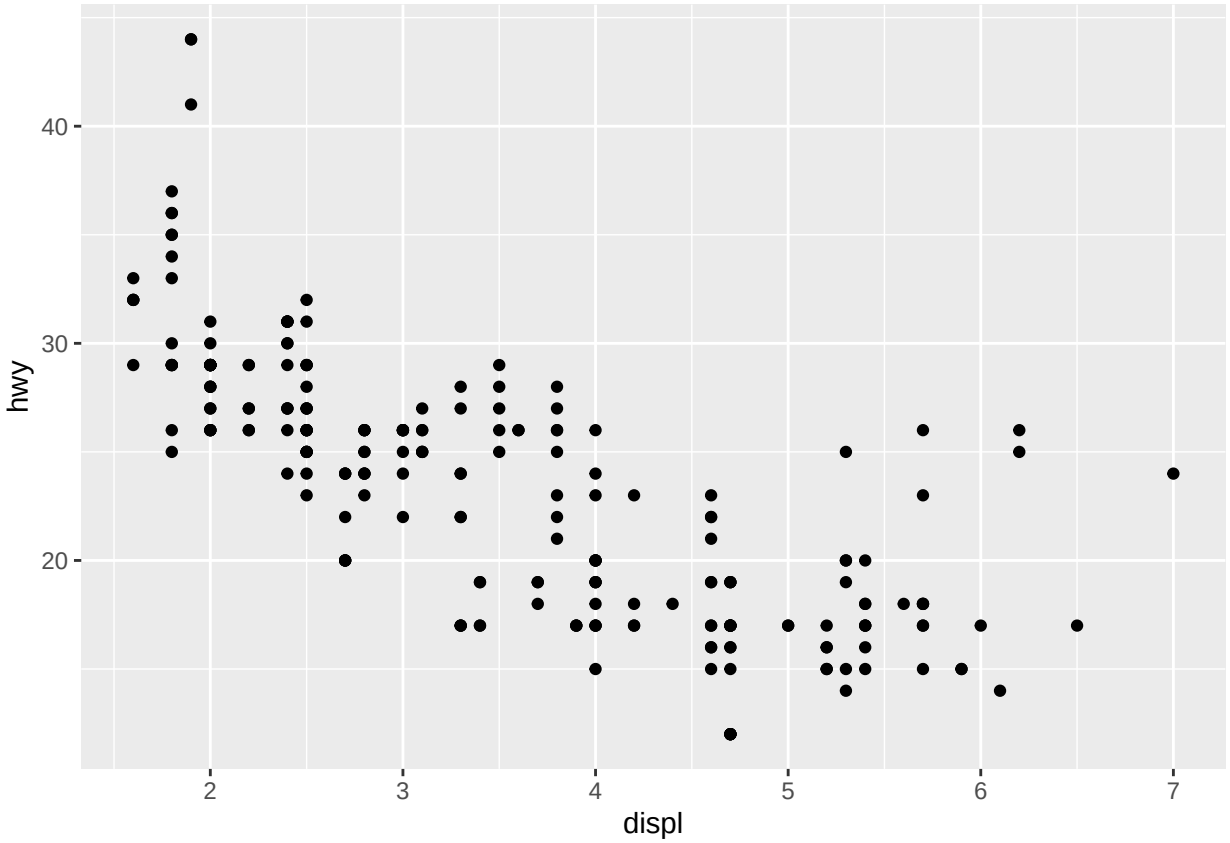



```
#ncol is used to define the no. of columns that you want in the facet wrap output.
```

Lines

We already used points:

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



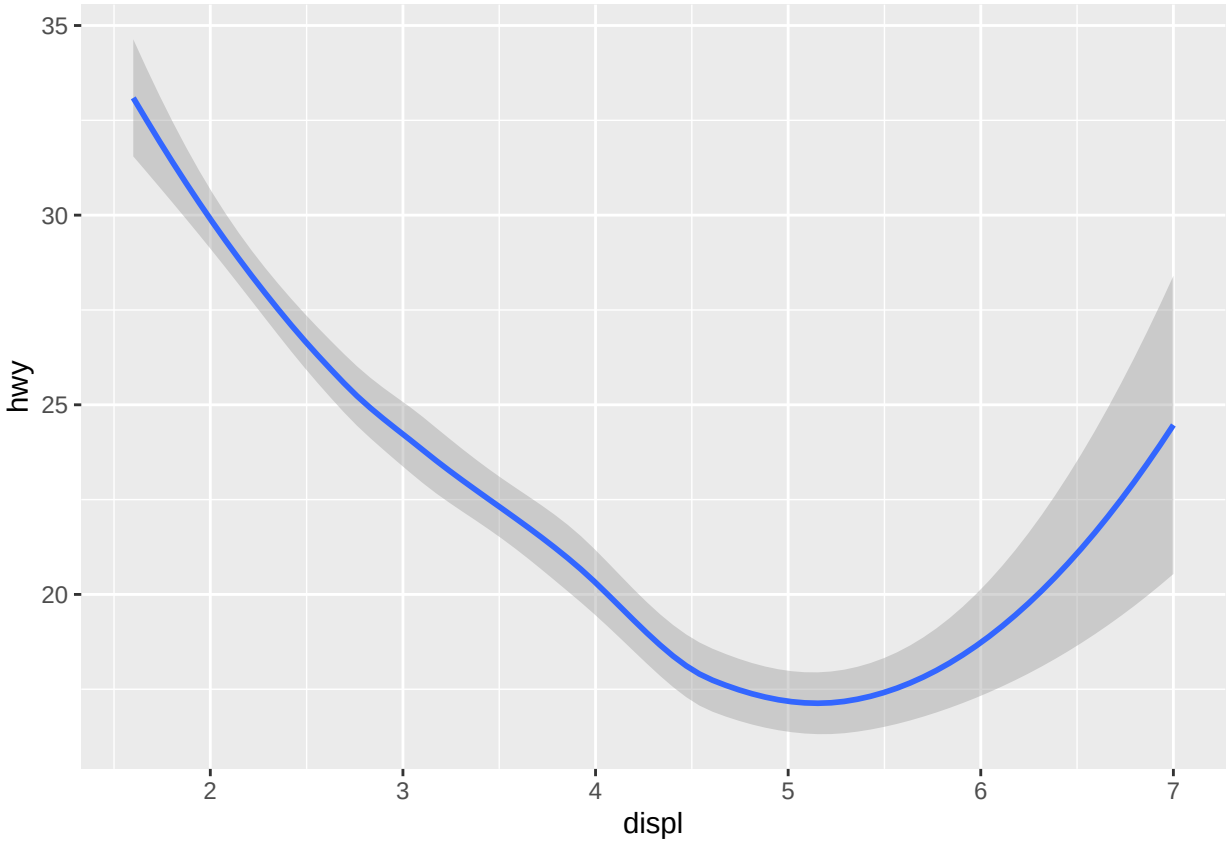
There was an error in the geom_point line with the erorr being x = displx. Answer = x = displ

However, ggplot2 can use a variety of geom objects to represent the data. Here, we might want to use bar plots, line charts, boxplots and so on. Well we can handle this issue in ggplot directly using a different geom to plot the same data. Here, instead of plotting points, we will use a smooth line.

To display the same data as a smooth line fit through the points use geom_smooth().

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

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



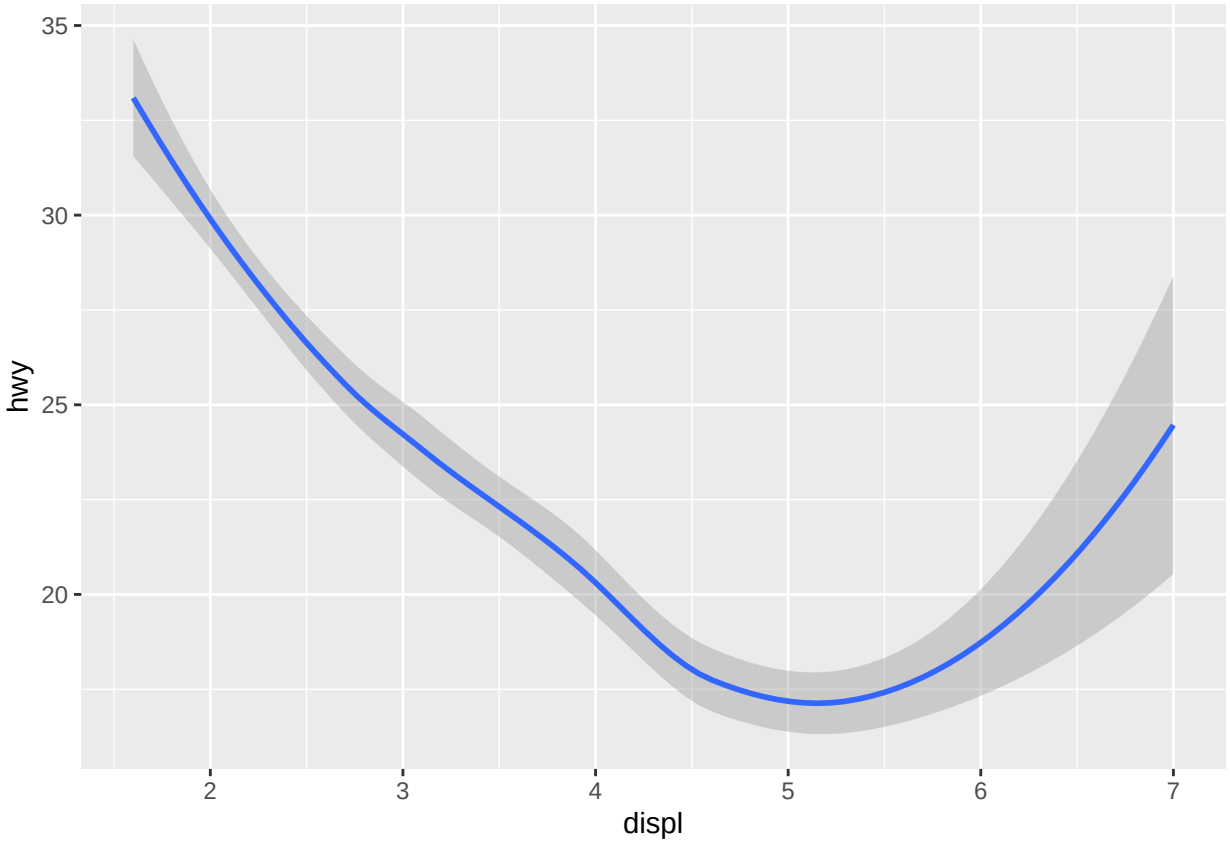
Question: whey don't we use geom_line() here? What would that look like? ### # goem_line would make our data look clustered and may not give a proper visual re-presentation of the data that we are looking for.

So let's recap. A geom is an object that your plot uses to represent the data. To change the geom type in your plot, simply change the geom function that you add to your plot template. Sometimes you may want to try a few things out, in which case you could use comments to help you remember what worked and what didn't.

Using comments (#)

```
ggplot(data = mpg) +
  #geom_point(mapping = aes(x = displ, y = hwy)) + # points horrible
  geom_smooth(mapping = aes(x = displ, y = hwy)) # try smooth line
```

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



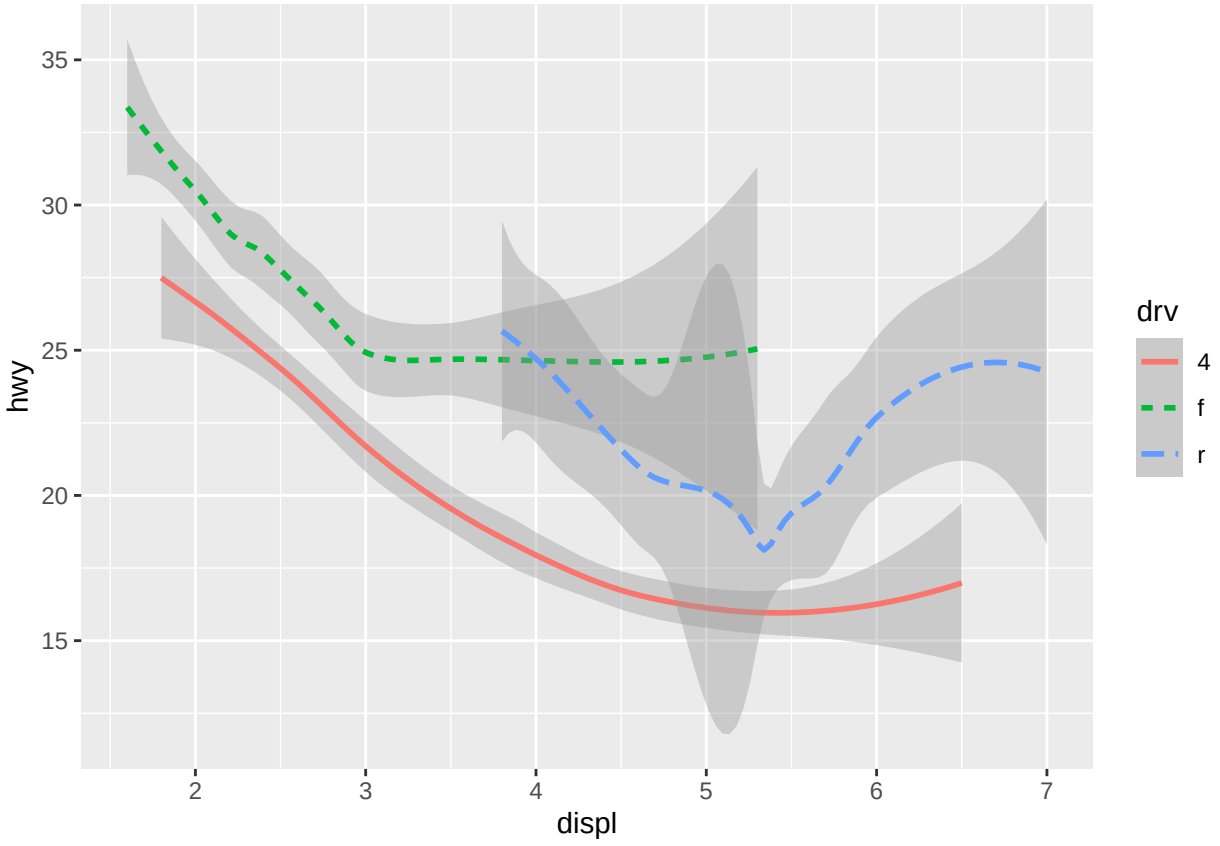

```
#Error in the geom_smooth line with maping as "mappings"
```

Question: how does R work out where to plot the line??? Can use the chunk output to inform us. Can also use the help menu.

Changing linetype

```
ggplot(data = mpg) +
  geom_smooth(mapping = aes(x = displ, y = hwy, linetype = drv, colour = drv)) # Can also use "lty = " ,
```

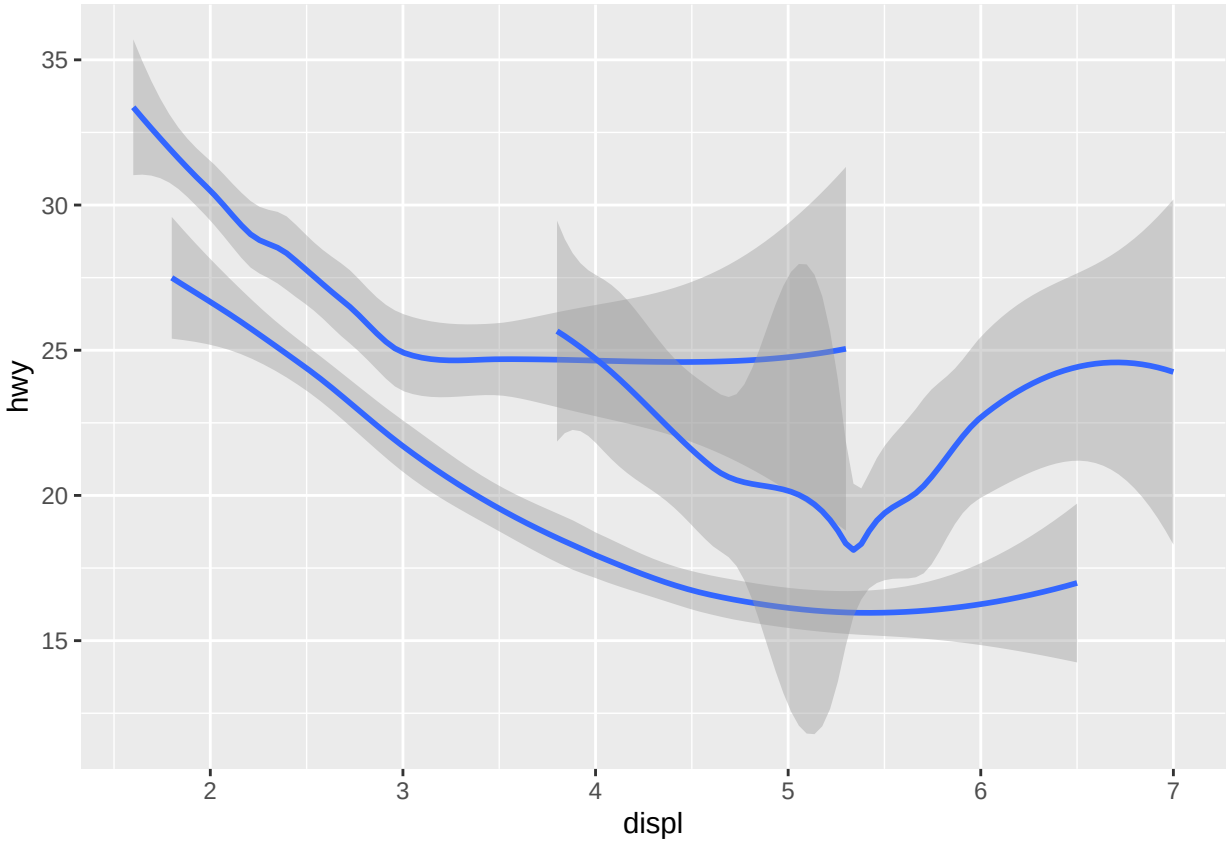
```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



Grouping

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

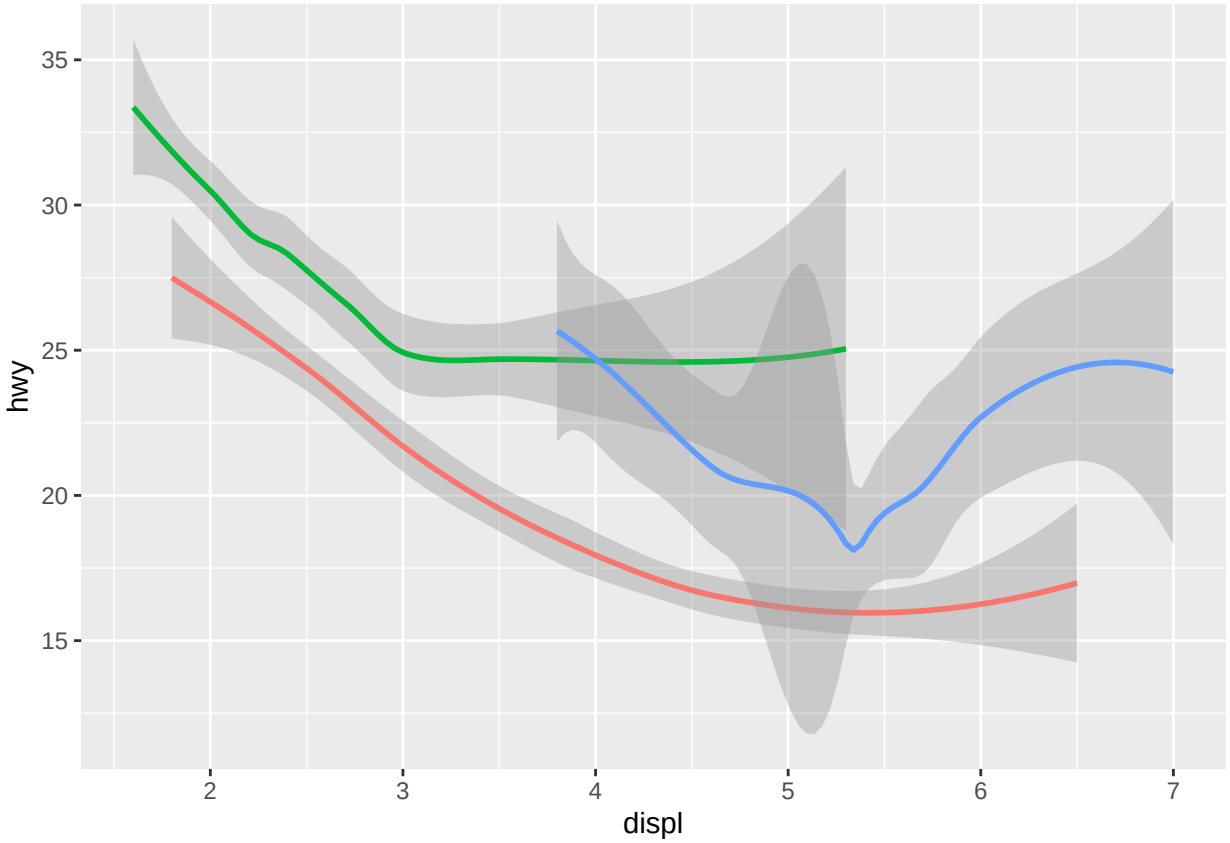
```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



Change line colour based on drv value

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

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

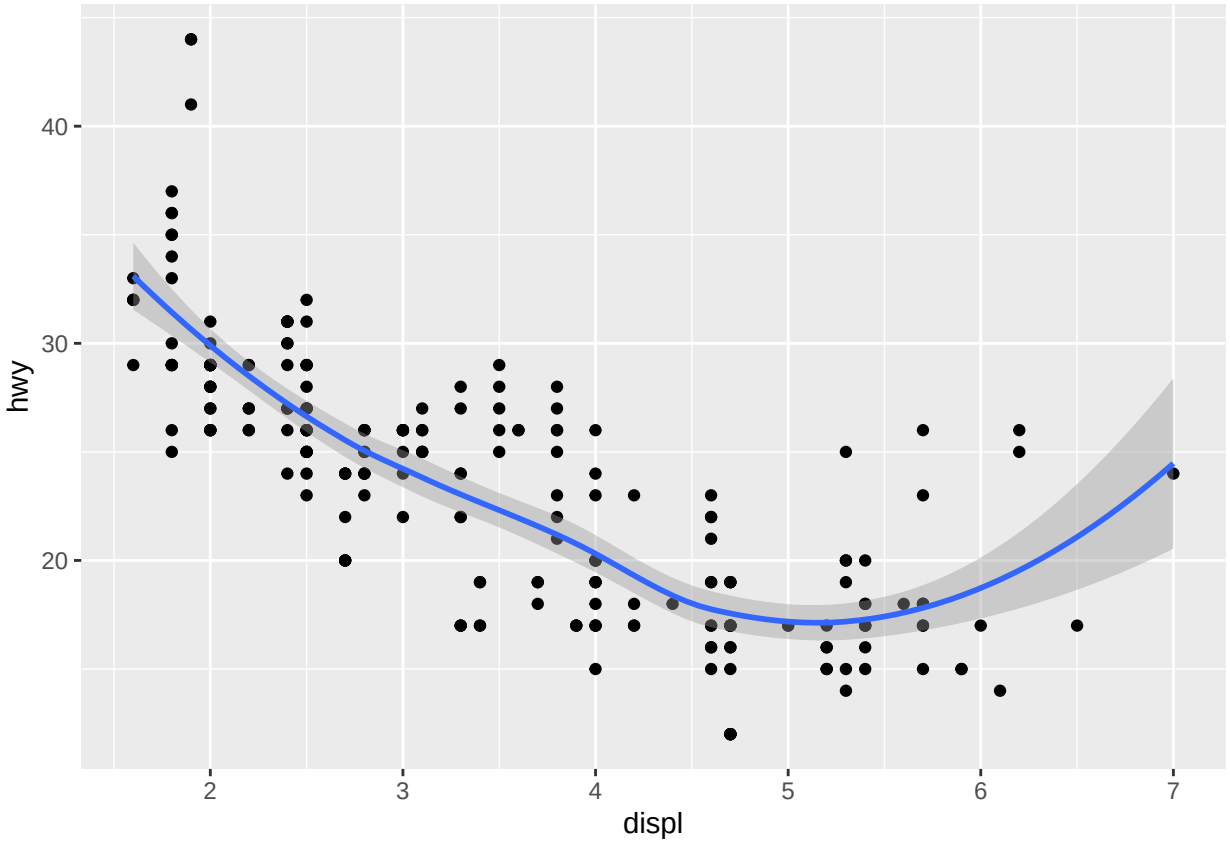


Multiple geoms

We already did this one way

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

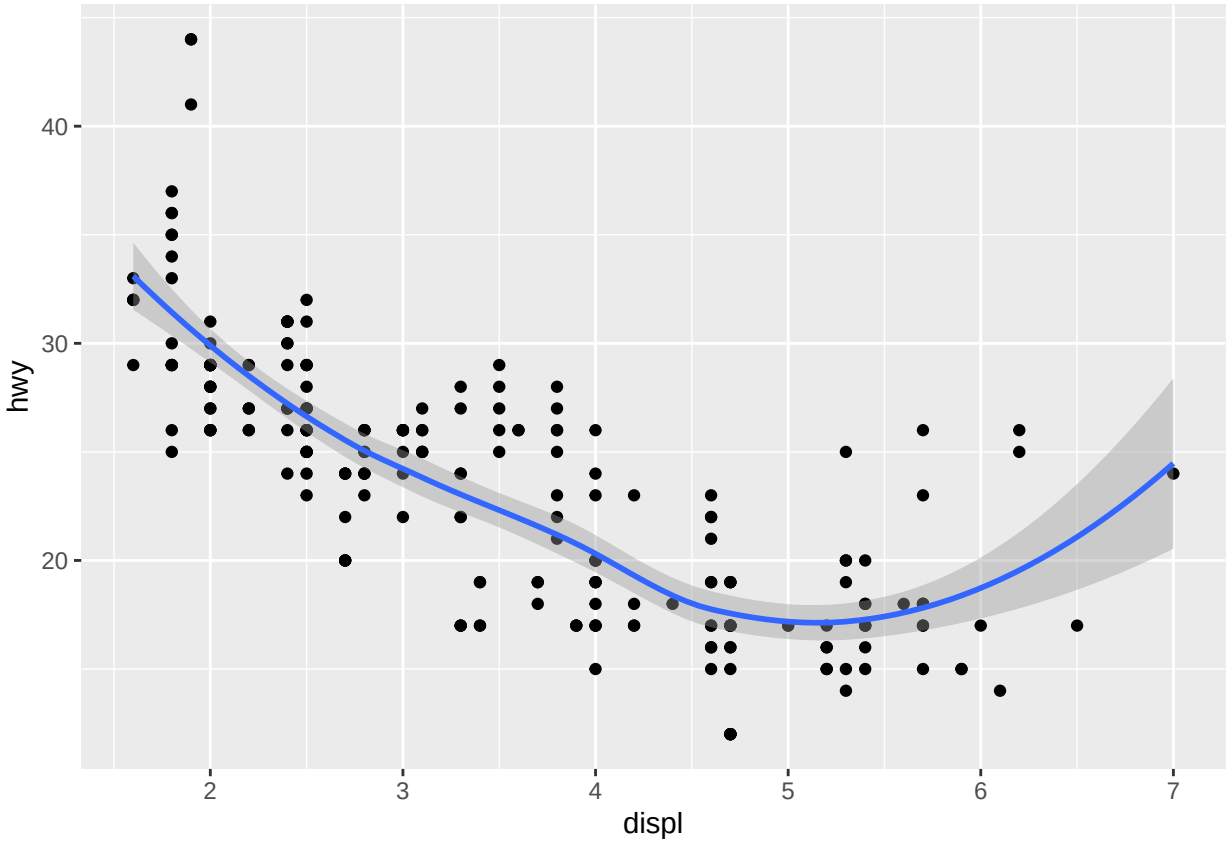
```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



A better way...

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

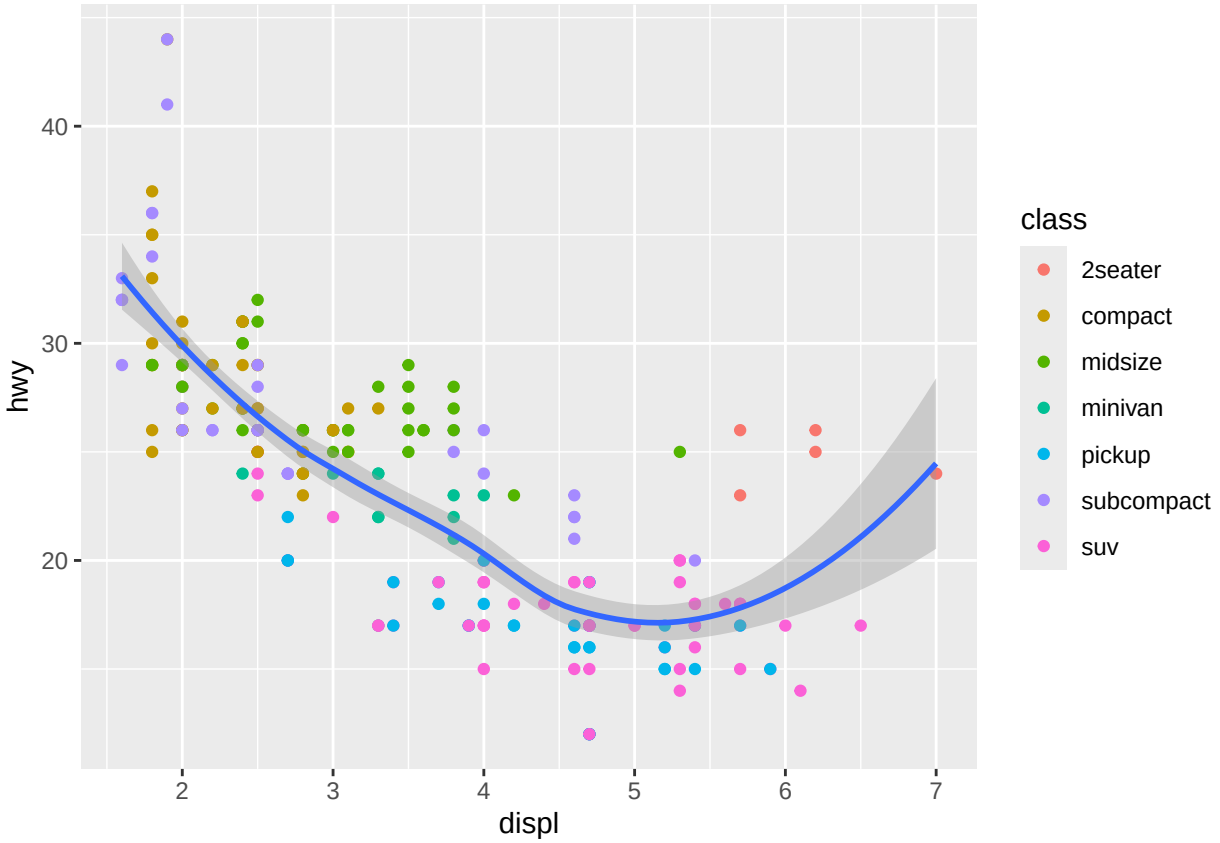
```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



Can still manipulate each geom/layer separately:

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

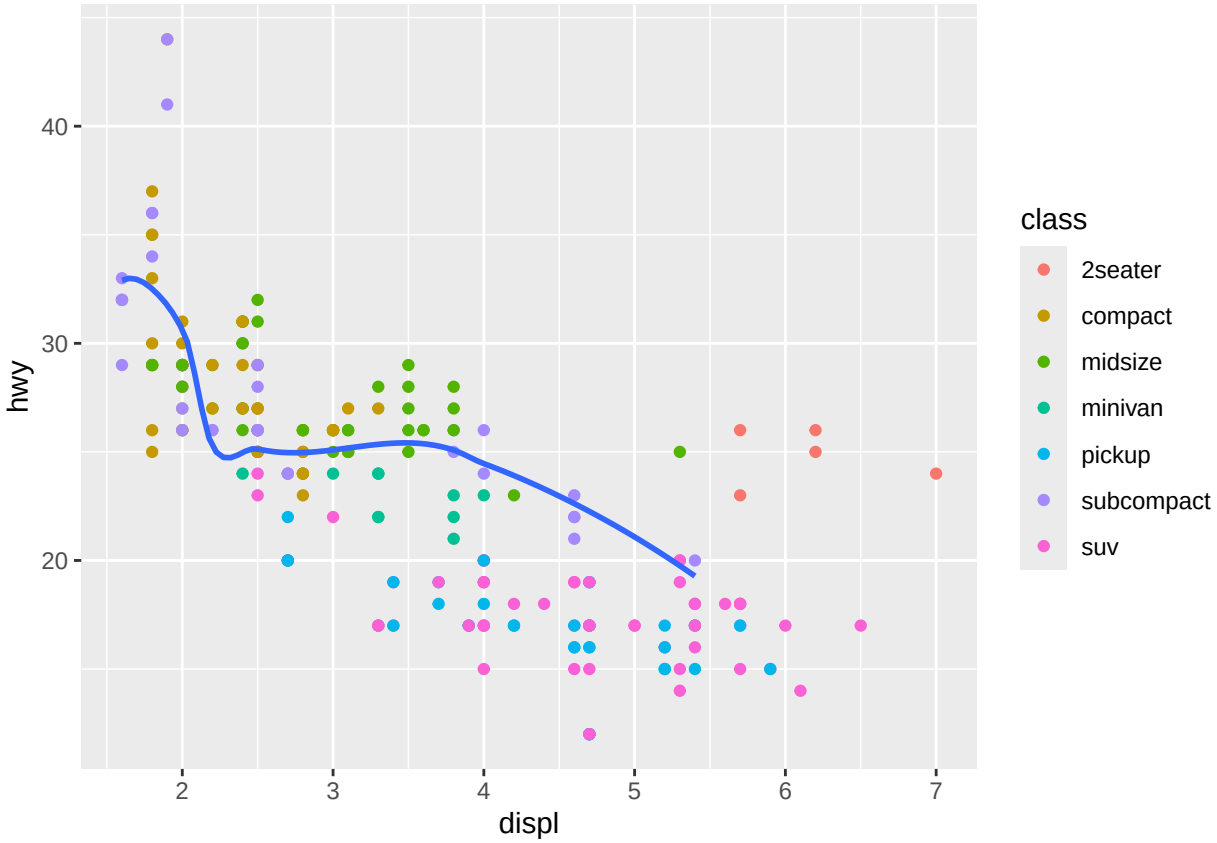
```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



Now we have different colours to indicate the raw data (geom_point) and one colour for the overall relationship (geom_smooth). Neat!

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

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

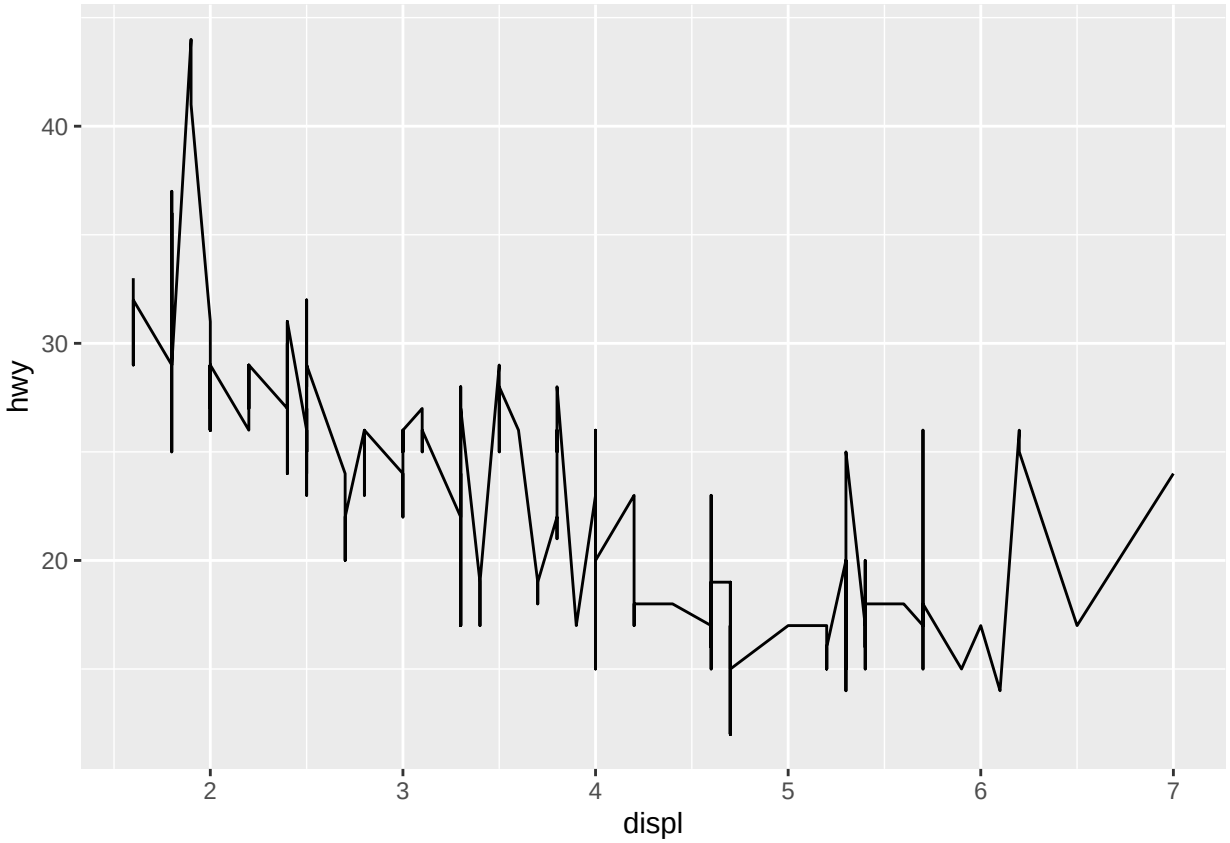


Question: what does the "se = FALSE" do? This function will turn off the standard error intervals for the line.

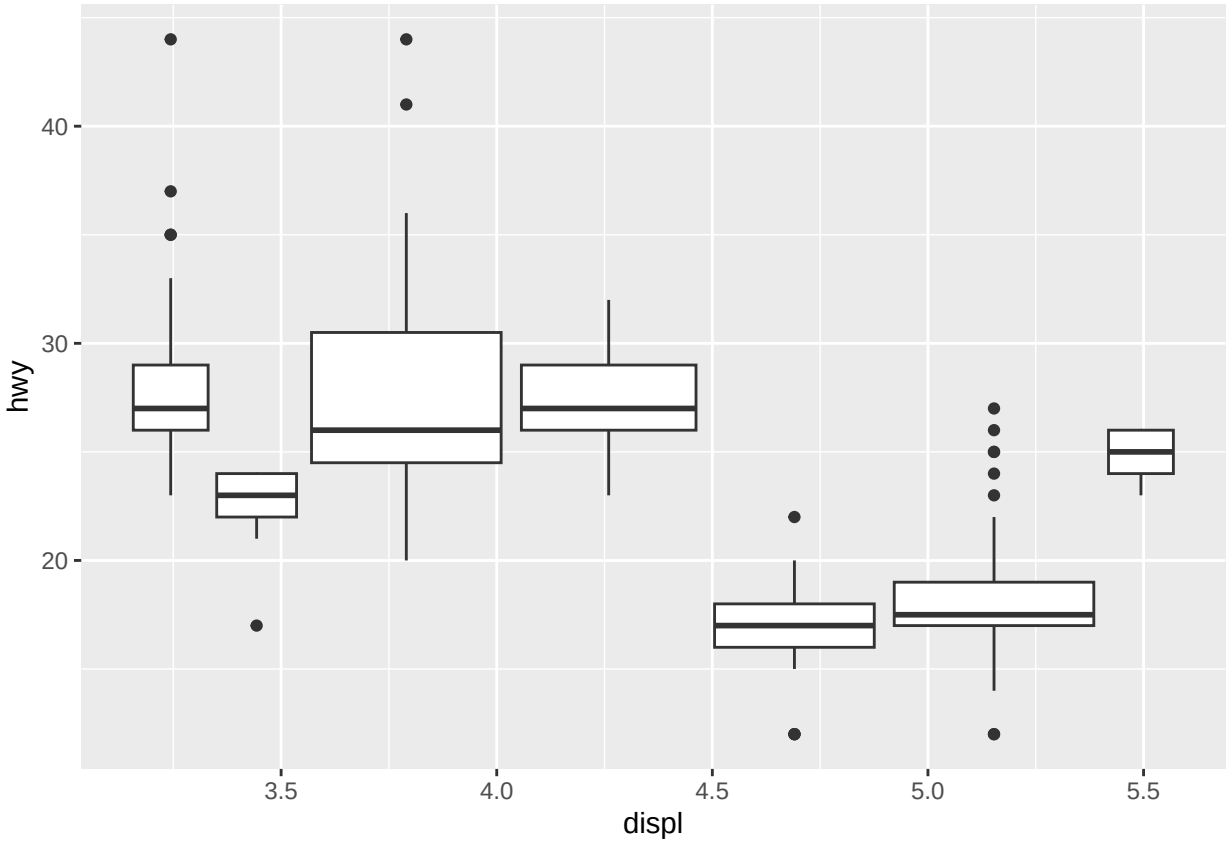
Exercise:

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

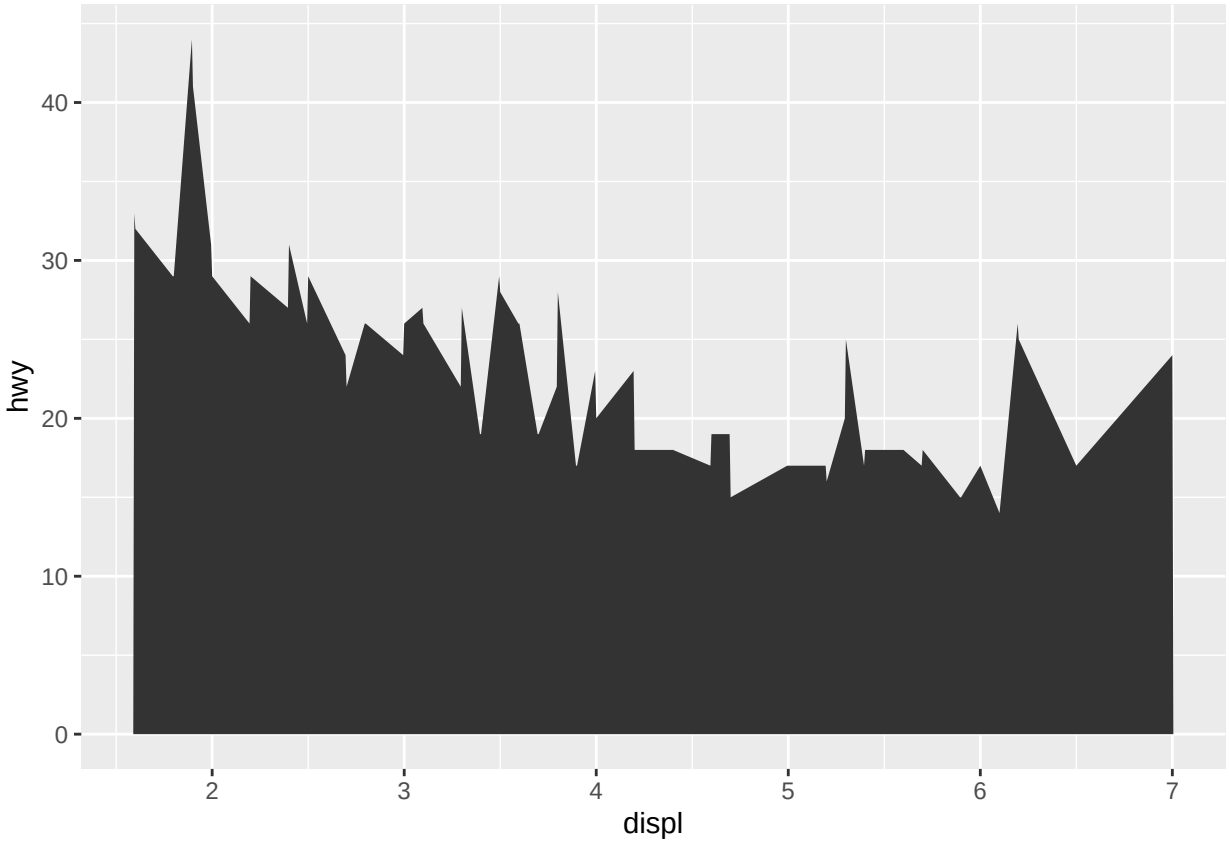
```
ggplot(data = mpg) +
  geom_line(mapping = aes(x = displ, y = hwy))
```




```
#ggplot for a line graph
  ggplot(data = mpg) +
    geom_boxplot(mapping = aes(x = displ, y = hwy, group = class))
```




```
#ggplot for a boxplot
  # Added grouping cause we are dealing with multi-class data.
  ggplot(data = mpg) +
    geom_area(mapping = aes(x = displ, y = hwy))
```

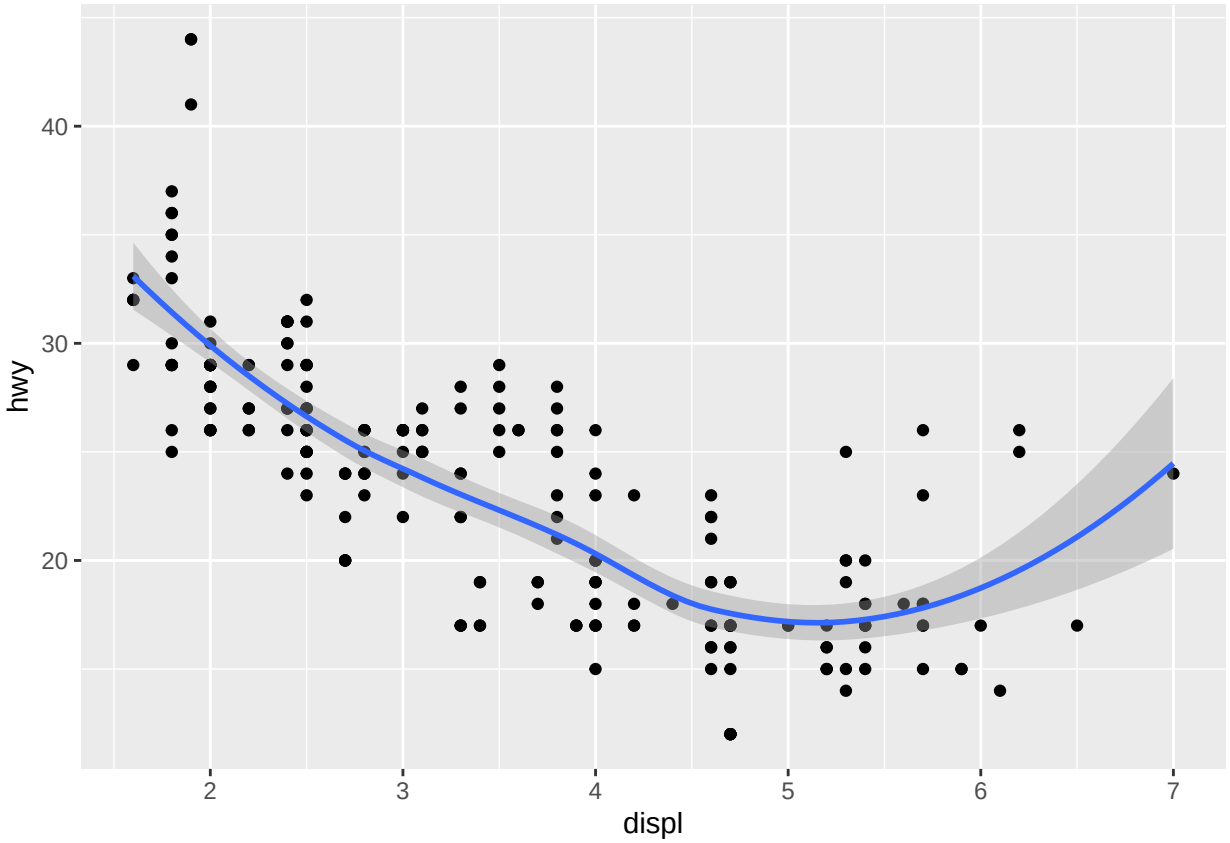



```
#ggplot for a area-map
```

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

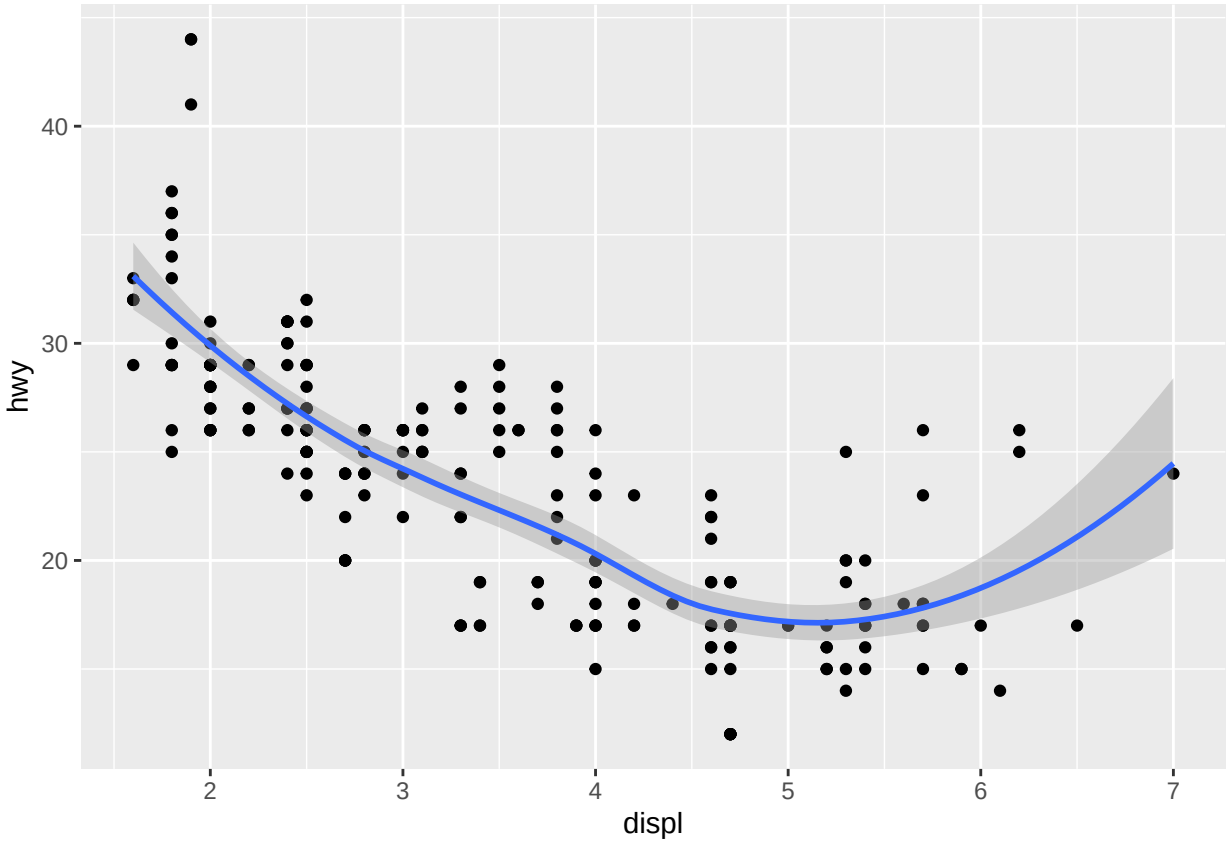
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth()
```

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




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

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



The graphs are the exact same. They look same irrespective of the code as the first code is a simplified fitted version of a regular code (i.e. the 2nd one) which includes all the defined paramters for the geom right after the function.

Transformations and stats

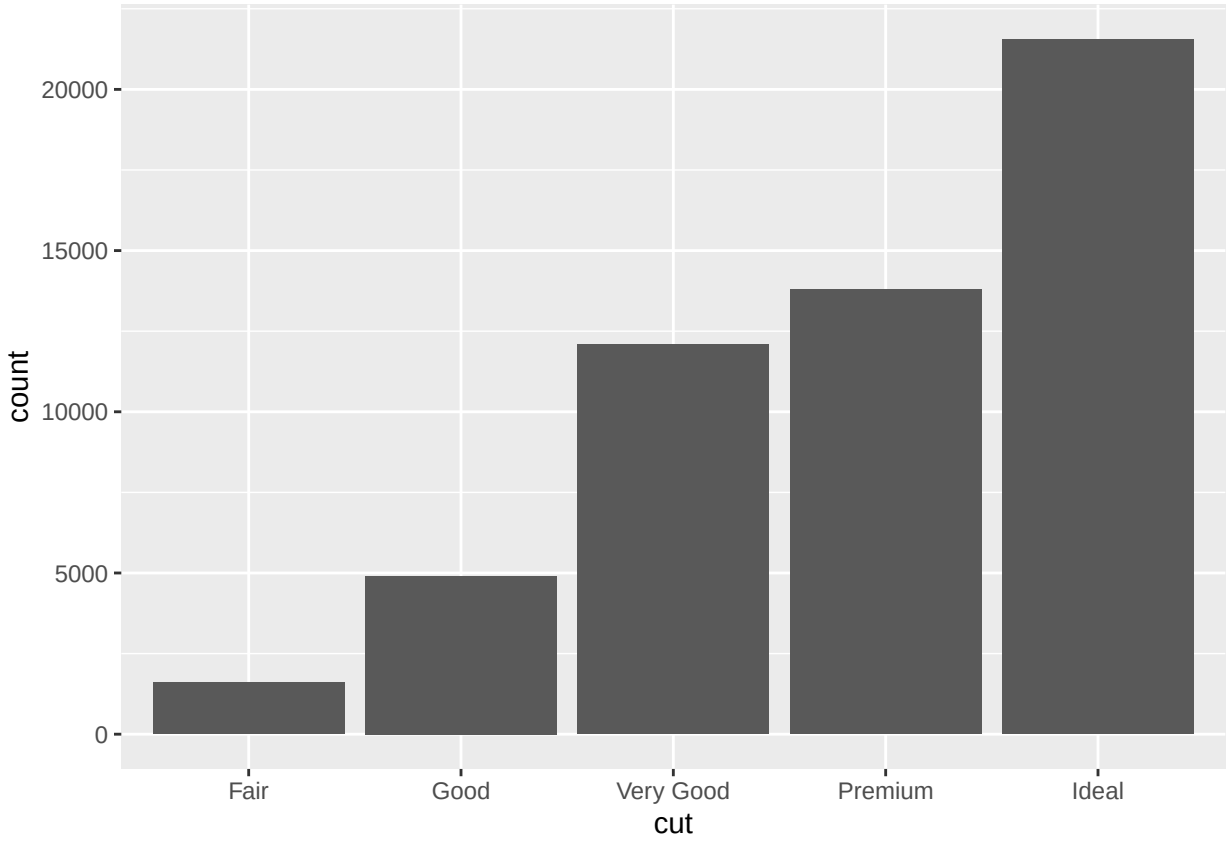
Load data

```
data("diamonds")
glimpse(diamonds)
```

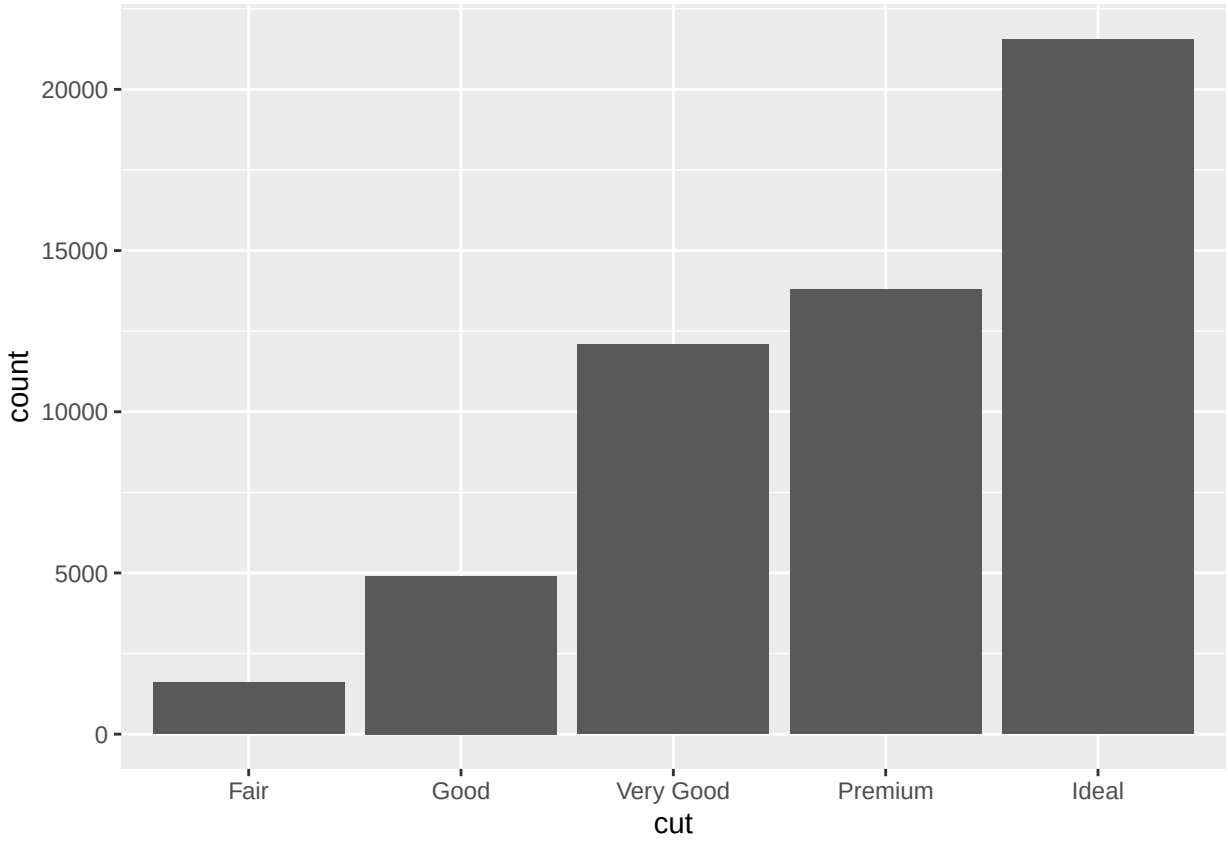
```
## Rows: 53,940
## Columns: 10
## $ carat   <dbl> 0.23, 0.21, 0.23, 0.29, 0.31, 0.24, 0.24, 0.26, 0.22, 0.23, 0.~
## $ cut     <ord> Ideal, Premium, Good, Premium, Good, Very Good, Very Good, Ver~
## $ color   <ord> E, E, E, I, J, J, I, H, E, H, J, J, F, J, E, E, I, J, J, J, I,~
## $ clarity <ord> SI2, SI1, VS1, VS2, SI2, VVS2, VVS1, SI1, VS2, VS1, SI1, VS1, ~
## $ depth   <dbl> 61.5, 59.8, 56.9, 62.4, 63.3, 62.8, 62.3, 61.9, 65.1, 59.4, 64~
## $ table   <dbl> 55, 61, 65, 58, 58, 57, 57, 55, 61, 61, 55, 56, 61, 54, 62, 58~
## $ price   <int> 326, 326, 327, 334, 335, 336, 336, 337, 337, 338, 339, 340, 34~
## $ x       <dbl> 3.95, 3.89, 4.05, 4.20, 4.34, 3.94, 3.95, 4.07, 3.87, 4.00, 4.~
## $ y       <dbl> 3.98, 3.84, 4.07, 4.23, 4.35, 3.96, 3.98, 4.11, 3.78, 4.05, 4.~
## $ z       <dbl> 2.43, 2.31, 2.31, 2.63, 2.75, 2.48, 2.47, 2.53, 2.49, 2.39, 2.~
```

Bar plot

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




```
ggplot(data = diamonds) +
  stat_count(mapping = aes(x = cut))
```




```
#using stat count
```

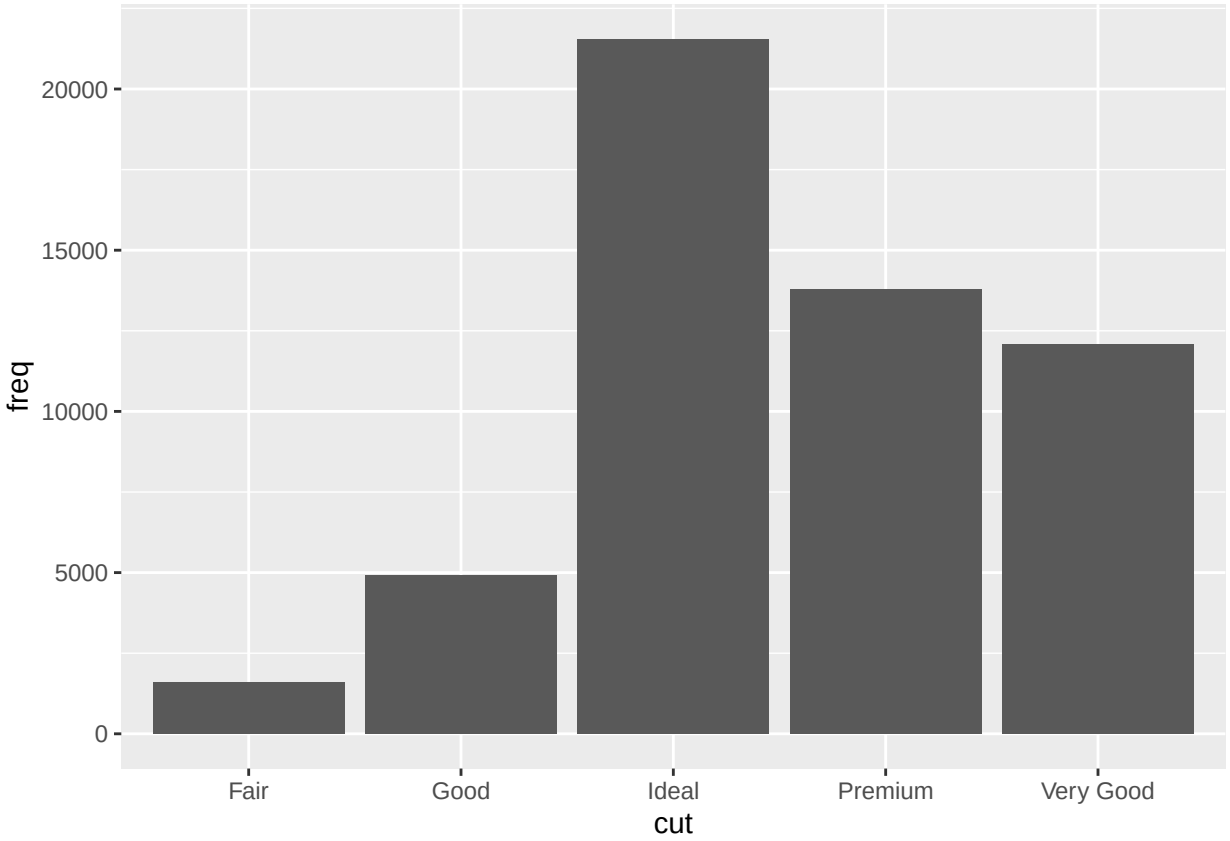
Overriding defaults

Make some new data

```
demo <- tribble(
  ~cut,          ~freq,
  "Fair",        1610,
  "Good",        4906,
  "Very Good",   12082,
  "Premium",     13791,
  "Ideal",       21551
)
demo
```

```
## # A tibble: 5 x 2
##   cut        freq
##   <chr>     <dbl>
## 1 Fair       1610
## 2 Good       4906
## 3 Very Good 12082
## 4 Premium   13791
## 5 Ideal     21551
```

```
ggplot(data = demo) +
  geom_bar(mapping = aes(x = cut, y = freq), stat = "identity")
```

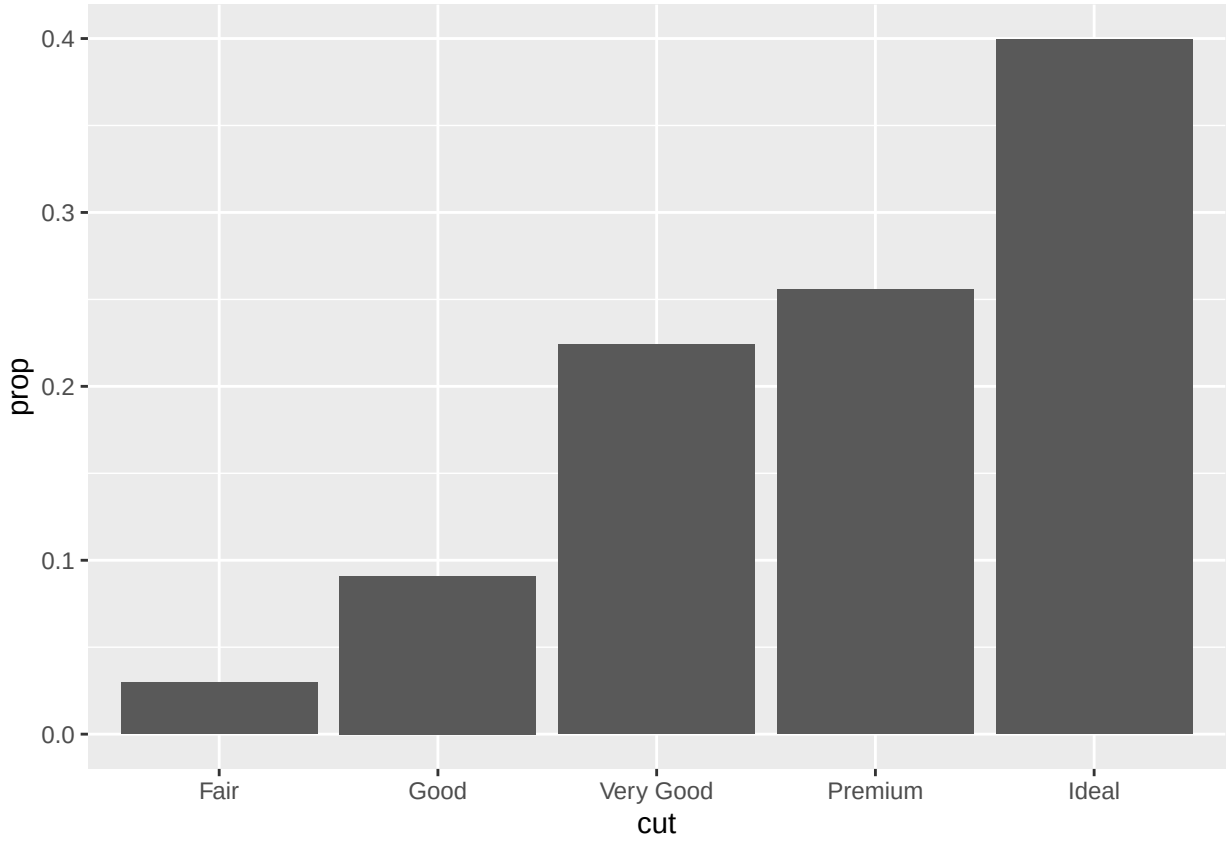


Question: Why did we specify now specify a y axis? #The x-axis by default will only give the occurances of each unique value. In this case, the types of cuts and use bars to display them. #The usage of Y-axis will specify the y variable for the ggplot function to plot against x.

What if we want to know the proportion rather than the absolute frequency?

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

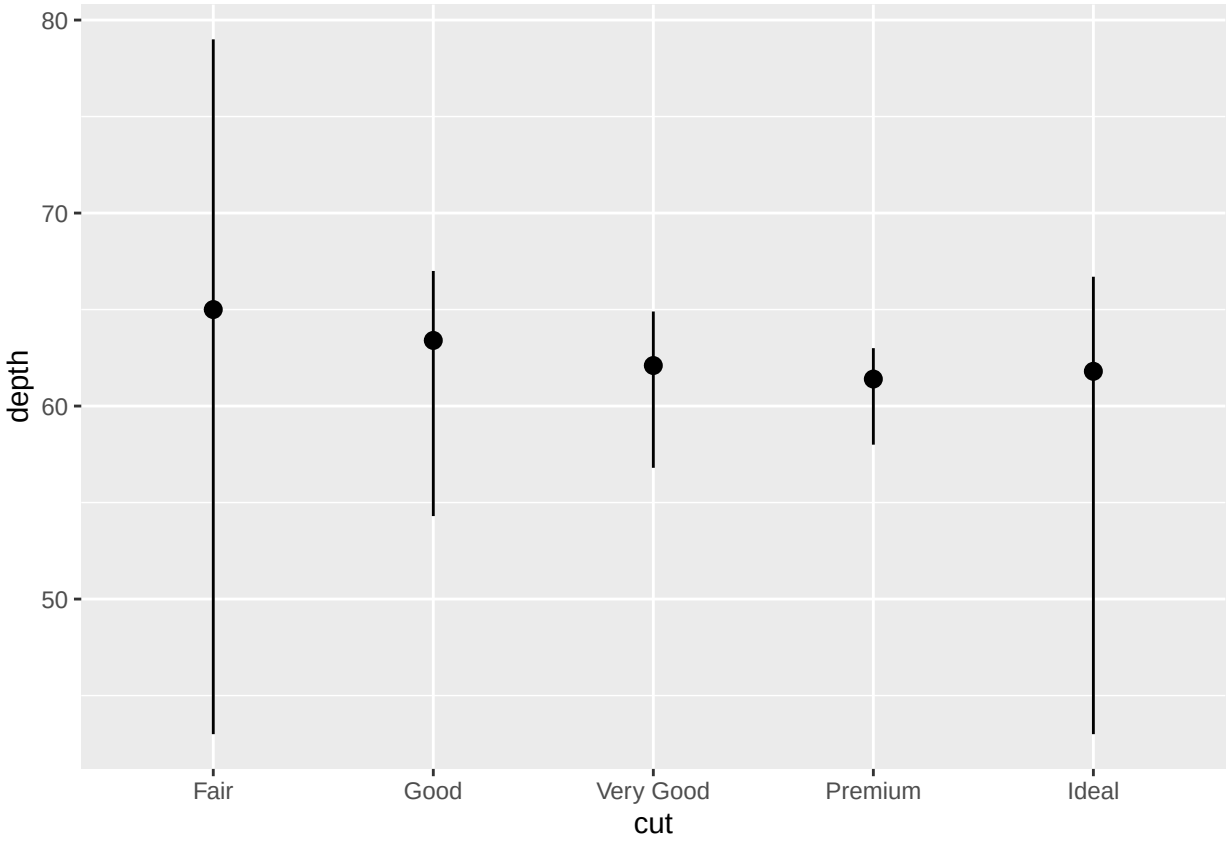
```
## Warning: `stat(prop)` was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(prop)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



Question: does anyone get the warning that's in the workbook? Warning: `stat(prop)` was deprecated in ggplot2 3.4.0. If so, can use # geom_bar(mapping = aes(x = cut, y = stage(after_stat = prop), group = 1))

Plotting statistical details

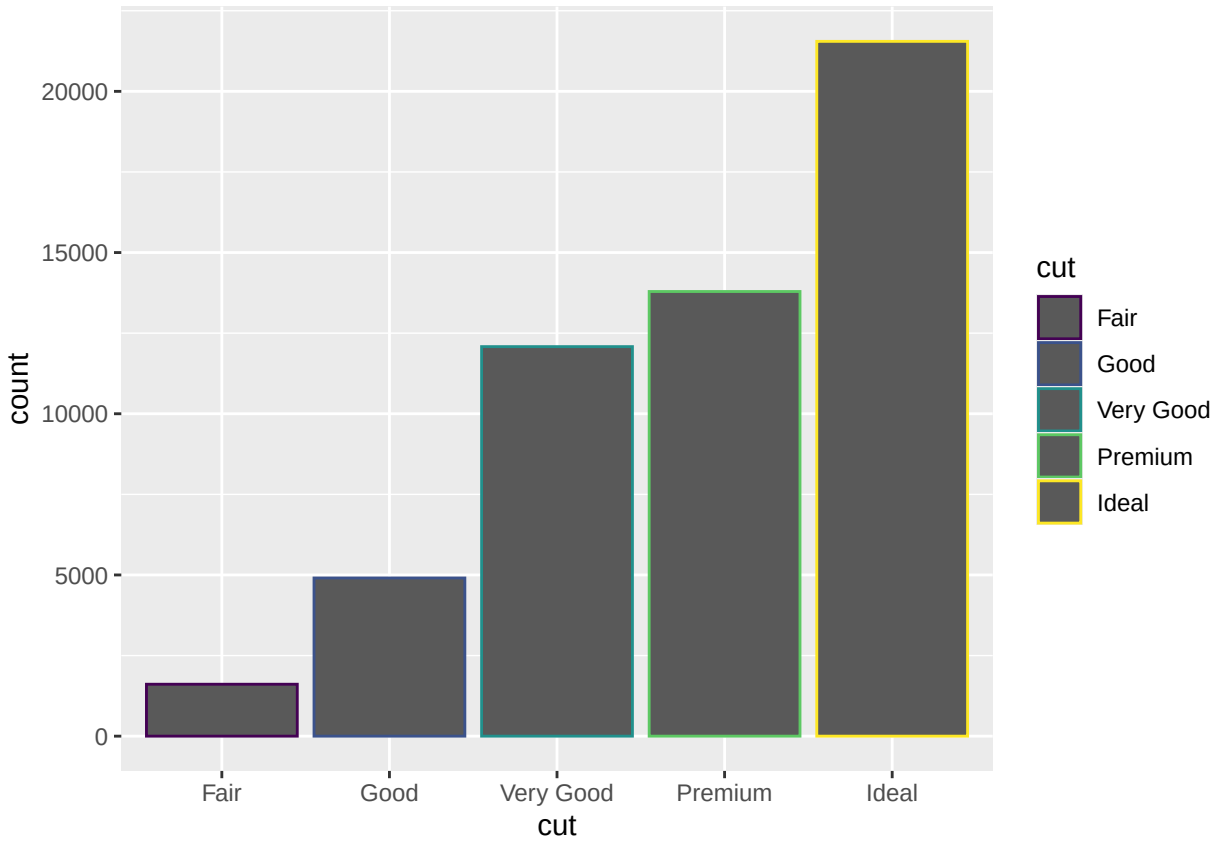
```
ggplot(data = diamonds) +
  stat_summary(
    mapping = aes(x = cut, y = depth),
    fun.min = min,
    fun.max = max,
    fun = median
  )
```



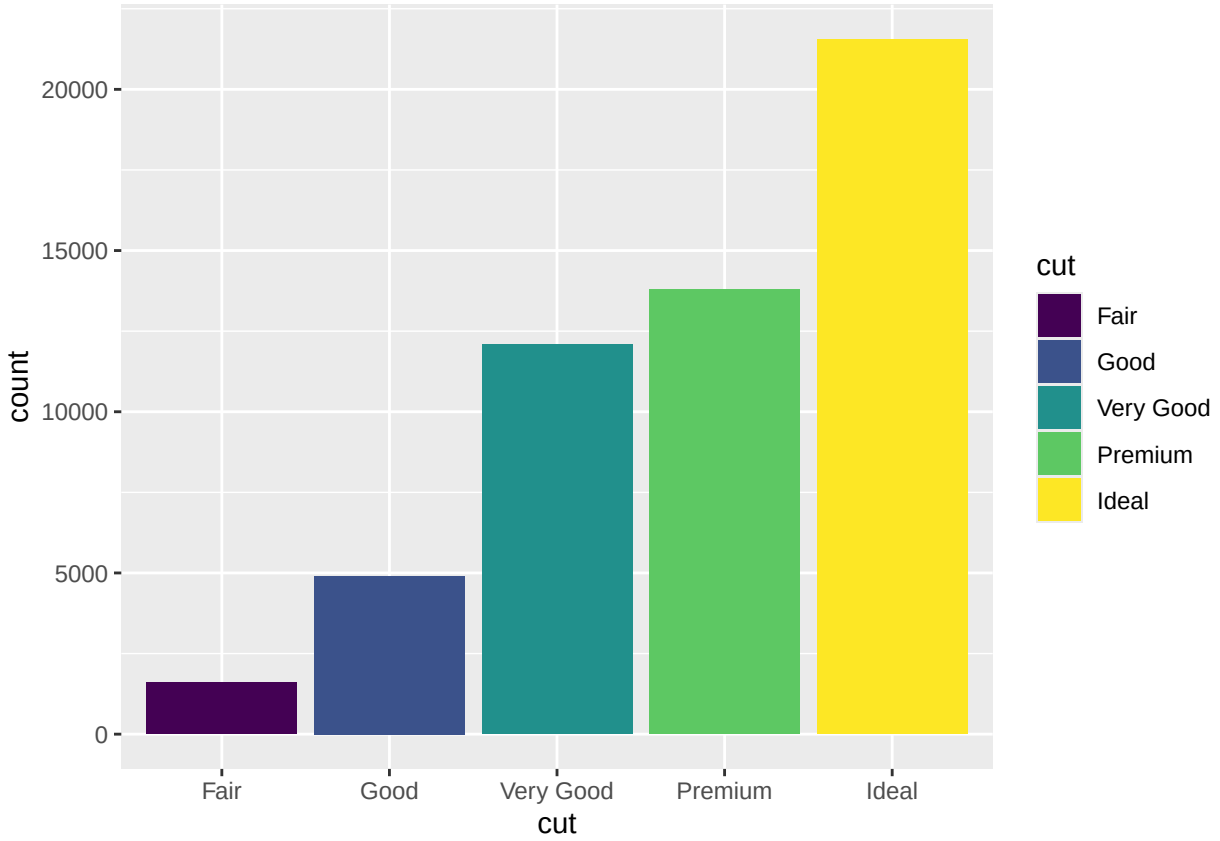
Aesthetic adjustments adjustments

Another way to boost the way you can convey information with plots using ggplot2 is to use aesthetics like colour or fill to change aspects of bar colours. We already did this once, but there are multiple options available to you, including changing the fill our outline colours.

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, colour = cut))
```

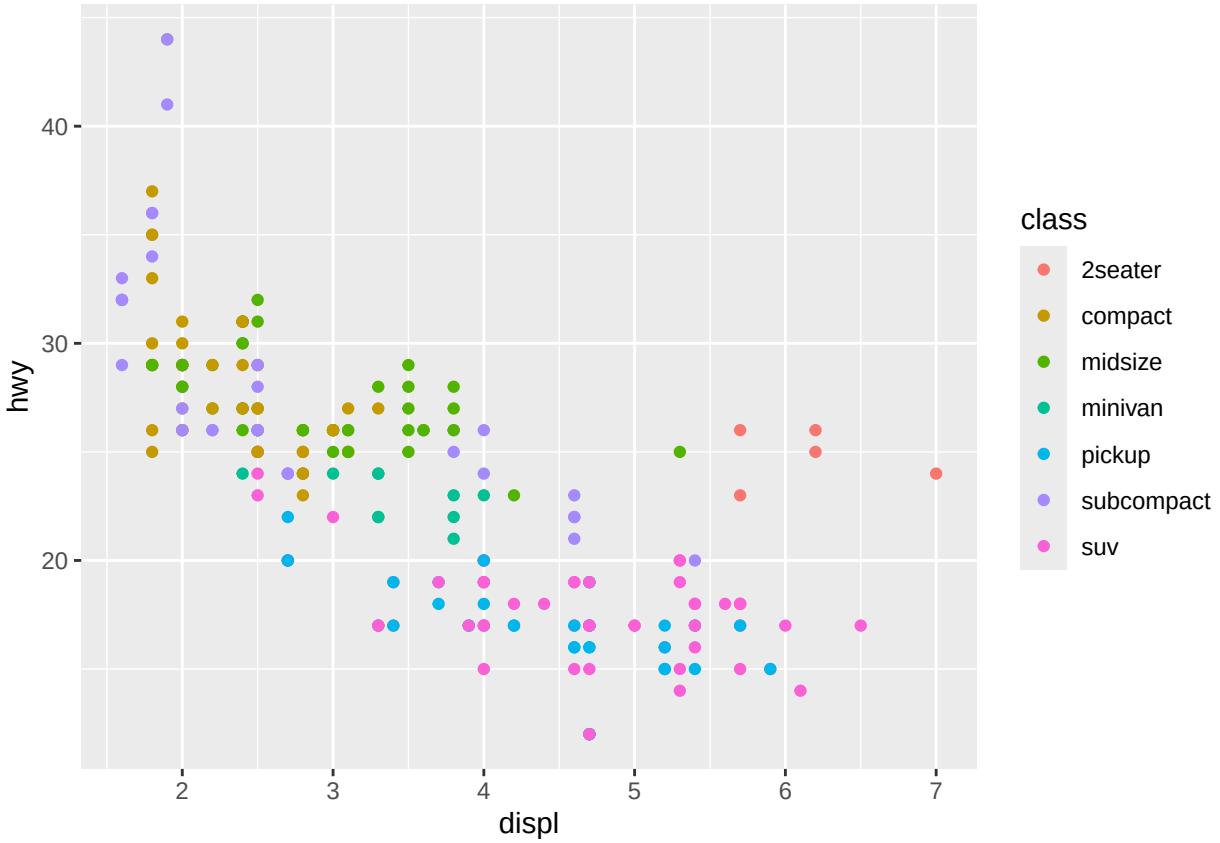



```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = cut))
```

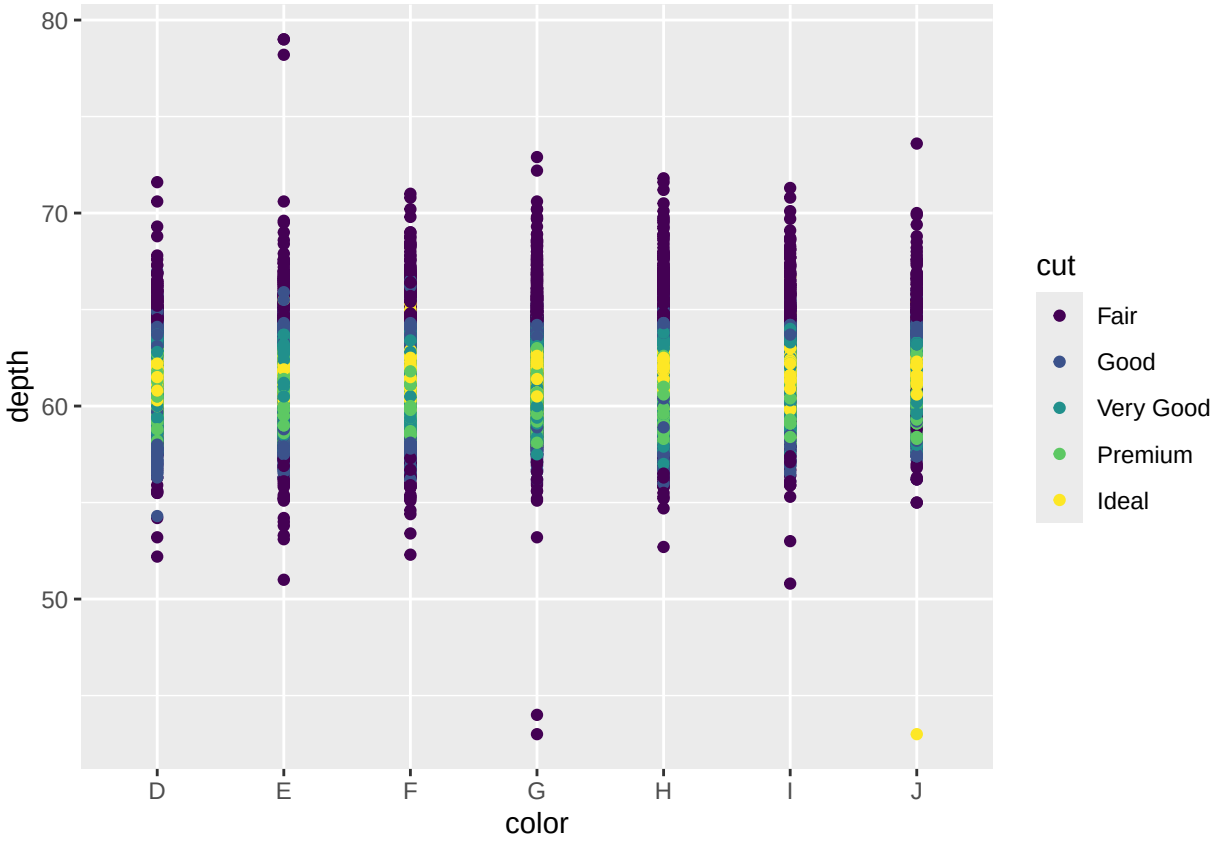


Question: Does anyone notice anything different in the colour scheme of this plot? (Hint: It's in the viridis colour palette (colour blind friendly), but why is it different from the colour palette we used earlier?) Check out the difference:

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



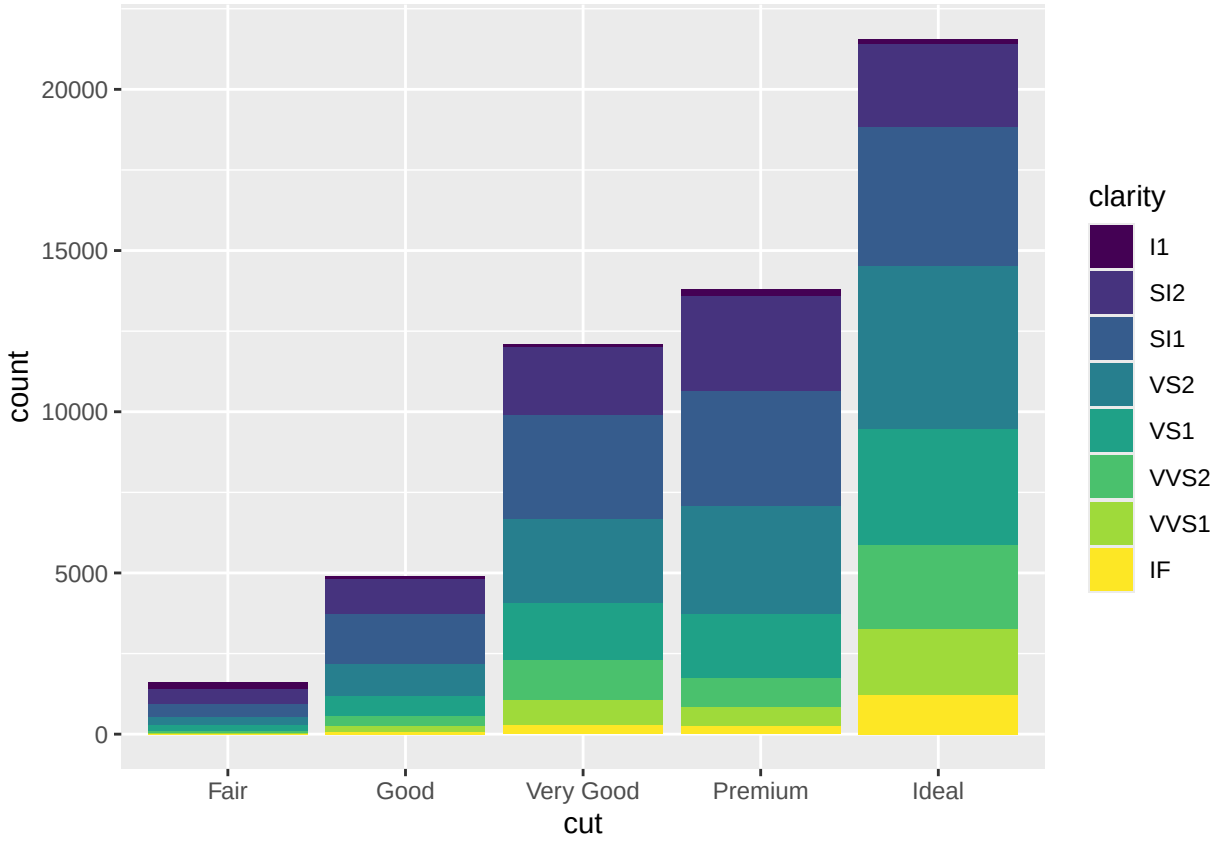

```
ggplot(data = diamonds, mapping = aes(x = color, y = depth)) +
  geom_point(mapping = aes(color = cut))
```



Filling by a variable

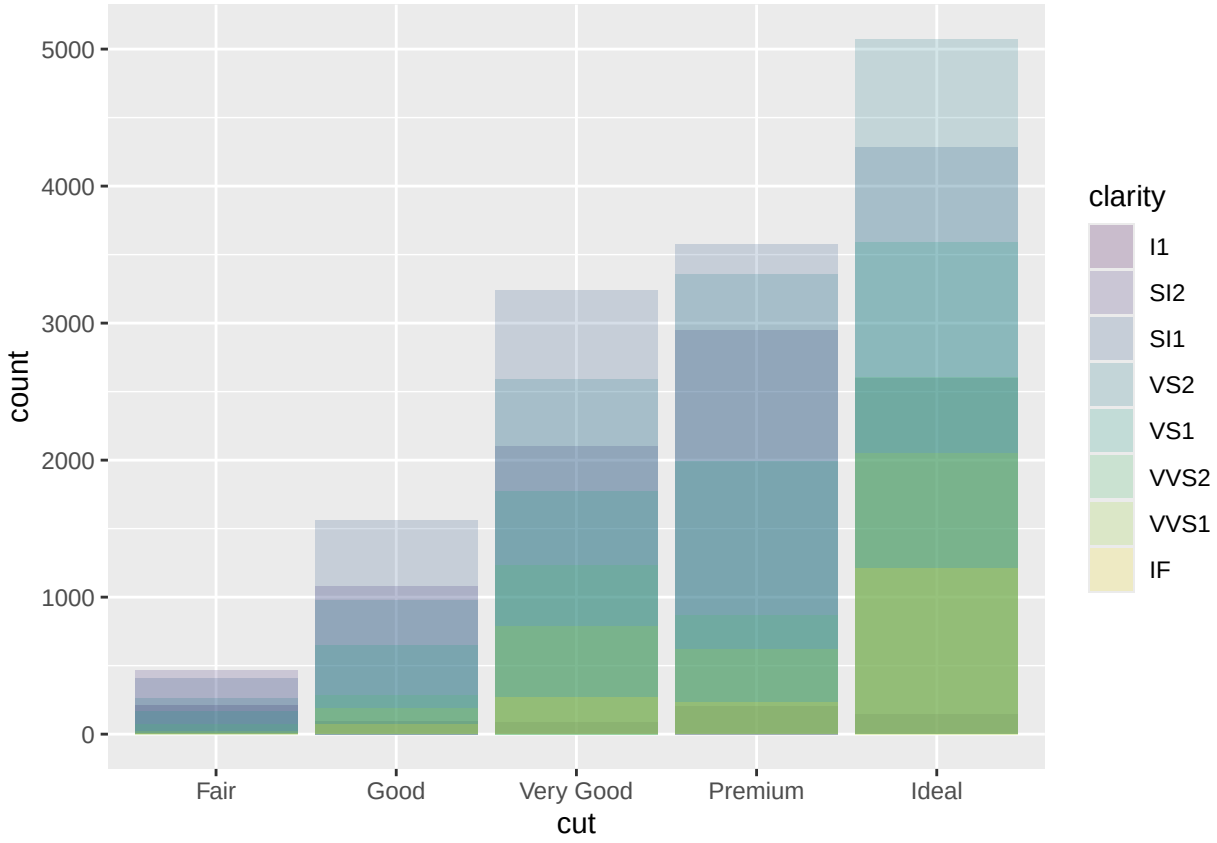
Now try using these aesthetics to colour by another variable like clarity. Notice how the stacking is done automatically. This is done behind the scenes with a position argument.

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



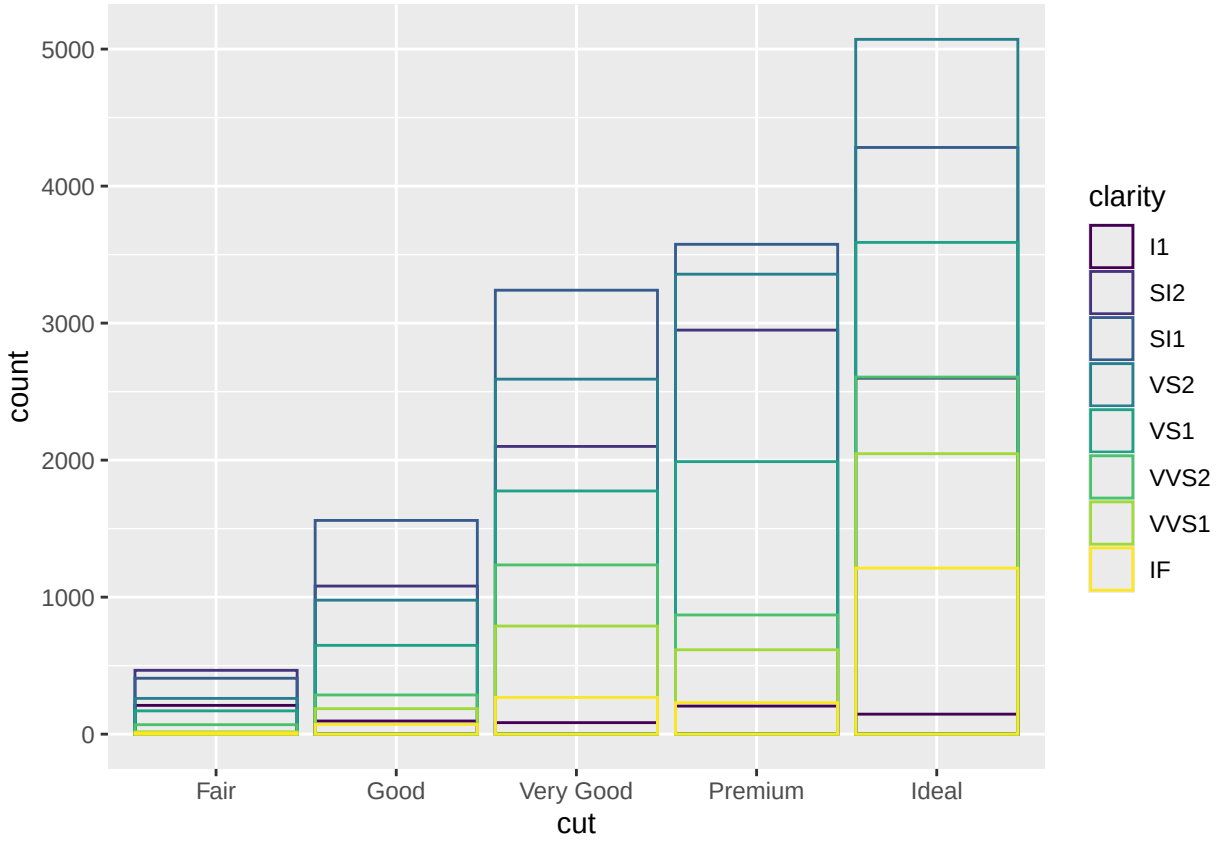
To alter transparency (alpha)

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



To color the bar outlines with no fill color

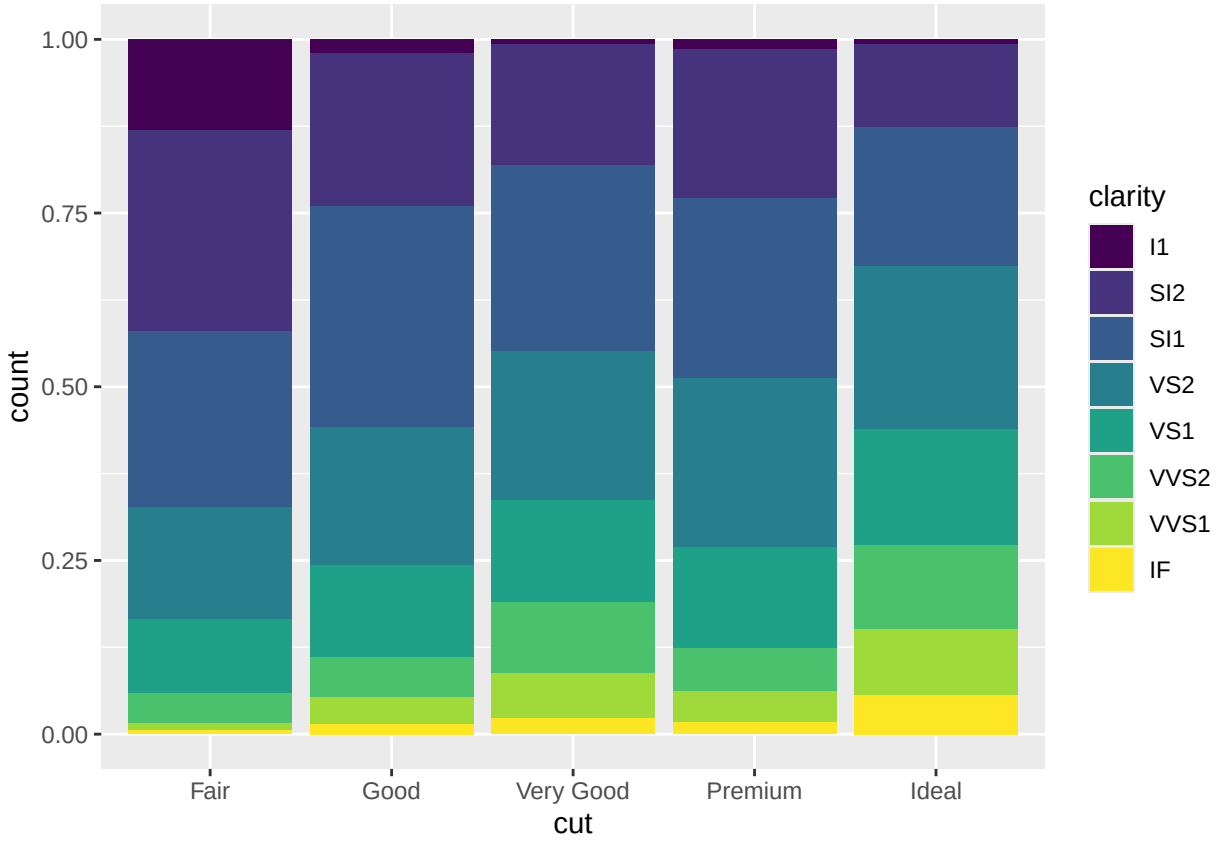
```
ggplot(data = diamonds, mapping = aes(x = cut, colour = clarity)) +
  geom_bar(fill = NA, position = "identity")
```



Position adjustments

position = "fill" works like stacking, but makes each set of stacked bars the same height.

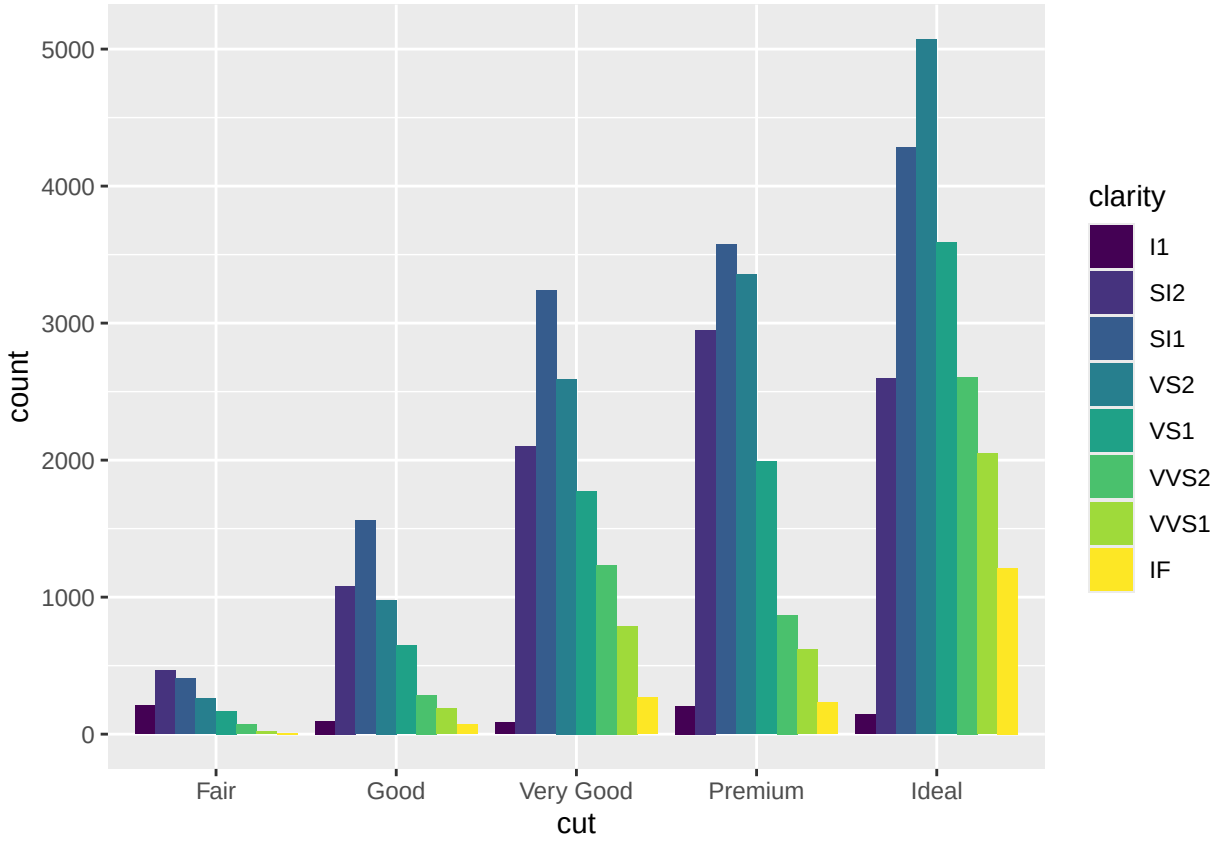
```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = clarity), position = "fill")
```



position = "dodge"

Places overlapping objects directly beside one another.

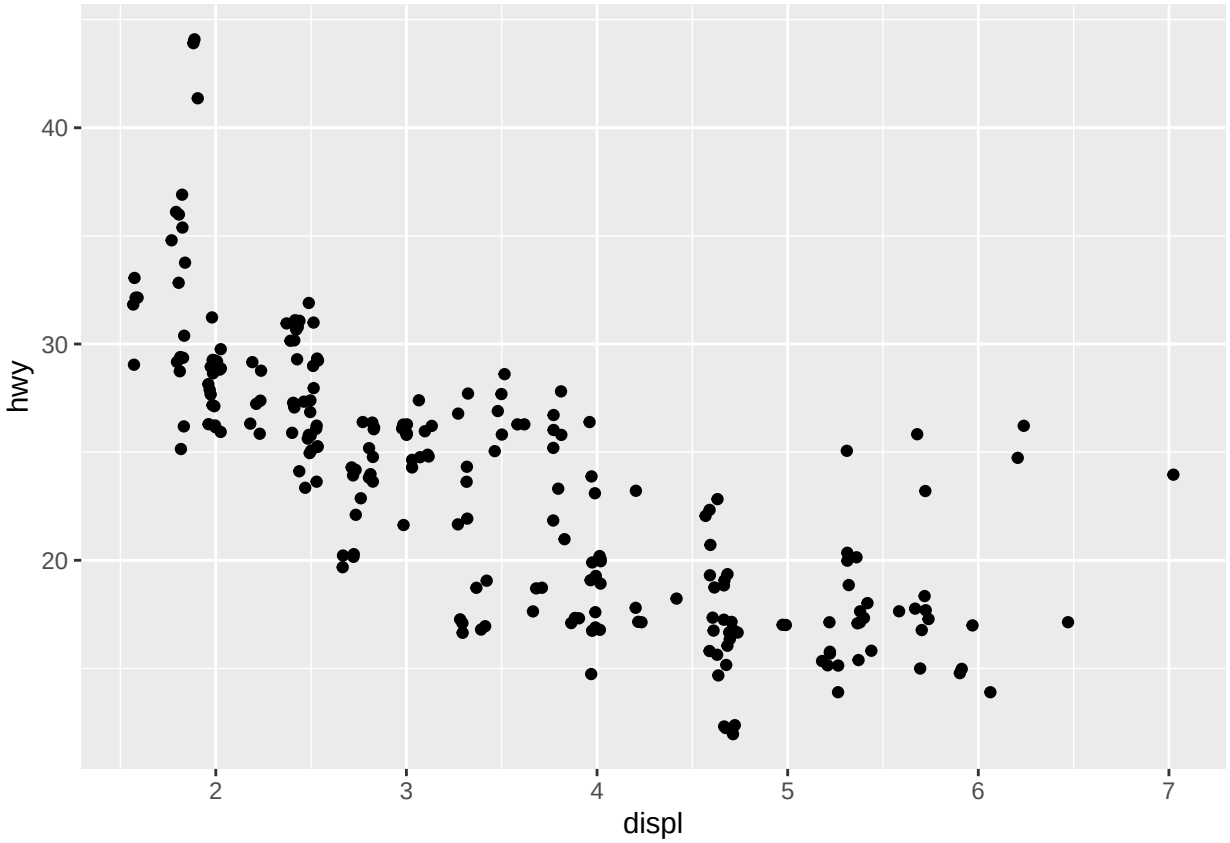
```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = clarity), position = "dodge")
```



Jittering

position = "jitter" adds a small amount of random noise to each point to avoid overplotting when points overlap. This is useful for scatterplots but not barplots.

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


40
30
20
hwy
2
3
4
5
6
7
displ