Data Visualization using ggplot2

J. Wall

January, 2019

Abstract

This is adapted from content in the book *R for Data Science*.
https://r4ds.had.co.nz/data-visualisation.html

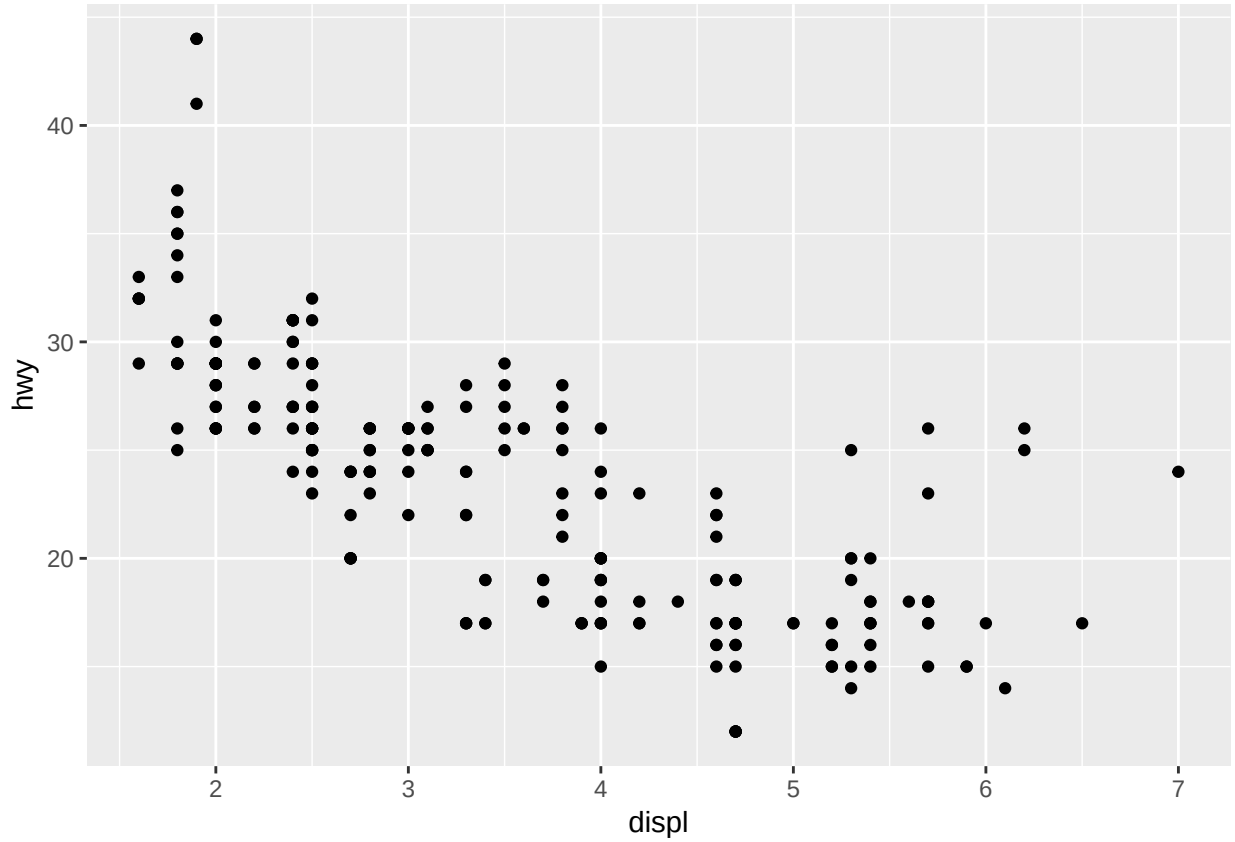
Getting started with ggplot

The "gg" in ggplot2 stands for "grammar of graphics", a coherent system for describing and building graphs. We will use the mpg dataframe to explore ggplot capabilities. Type ?mpg or help(mpg) to get info on the dataset.

```
library(ggplot2)
library(dplyr)
```

Note that displ is the car's engine size and hwy is the mpg for the car on the highway. ggplot call creates axes on which to plot the selected data. we then add a layer of geometric points that represent the x and y coordinates that we want to show.

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



.

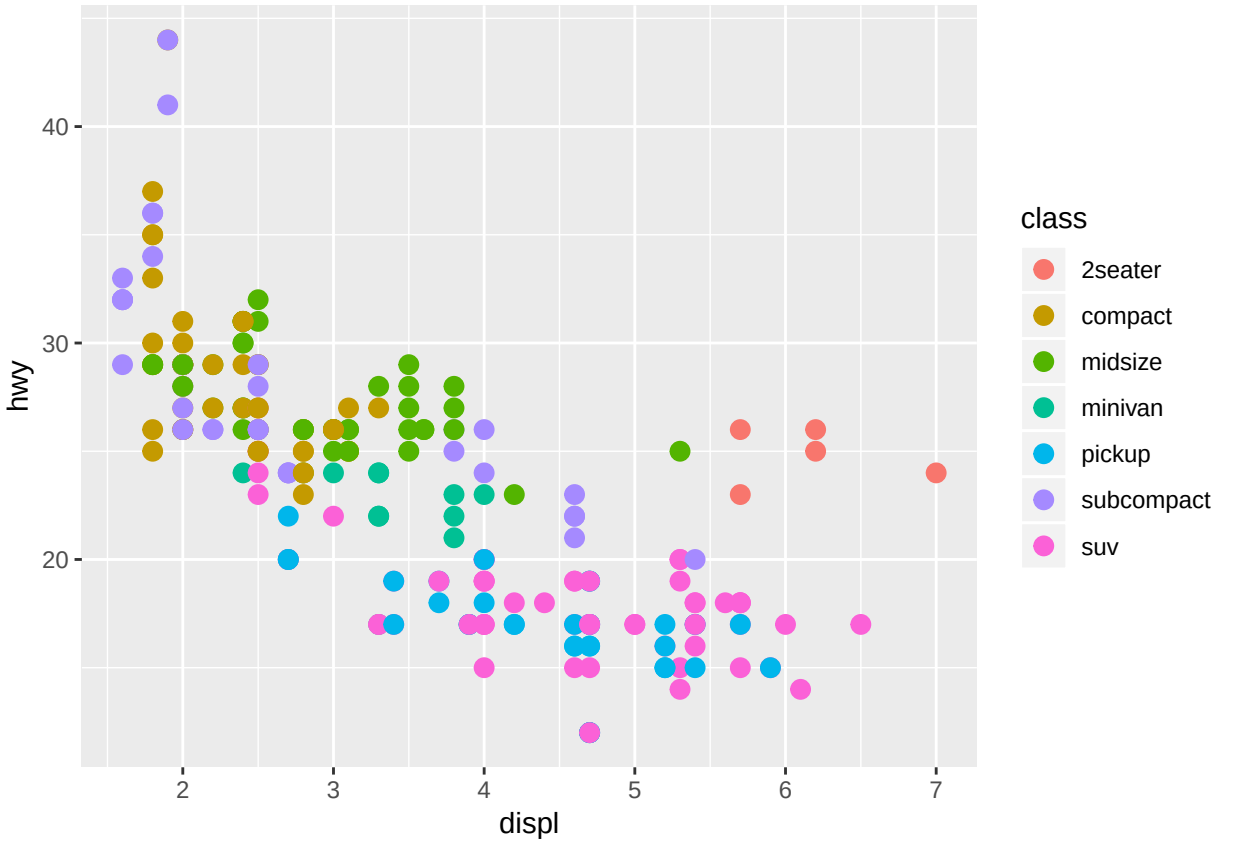
Exercises:

1. Run ggplot(data = mpg) What do you see? Why?
2. What does the drv variable describe? Read the help for ?mpg to find out.
3. How many rows are in mtcars? How many columns?
4. Make a scatterplot of hwy versus cyl in the mpg data set.

Aesthetic mappings

An aesthetic is a visual property of the objects in your plot. Includes shape, size, color of the points. We can set the aesthetic (aes) values to depend on a variable value. If a property does not convey information about a variable, but just changes its appearance, it is set outside aes() but inside the mapping.

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy,  color = class),size = 3)
```



Experiment with setting different shapes (choose a number from 0-20). Size is given in mm. Color is a character string of a color.

Exercises:

1) Which variables in mpg are categorical? Which are continuous? You can type ?mpg or View(mpg) for help.
2) Map a continuous variable to color, size, and shape. How do these aesthetics behave differently for categorical versus continuous variables?
3) Can you map the same variables to multiple aesthetics?

http://r4ds.had.co.nz/data-visualisation.html

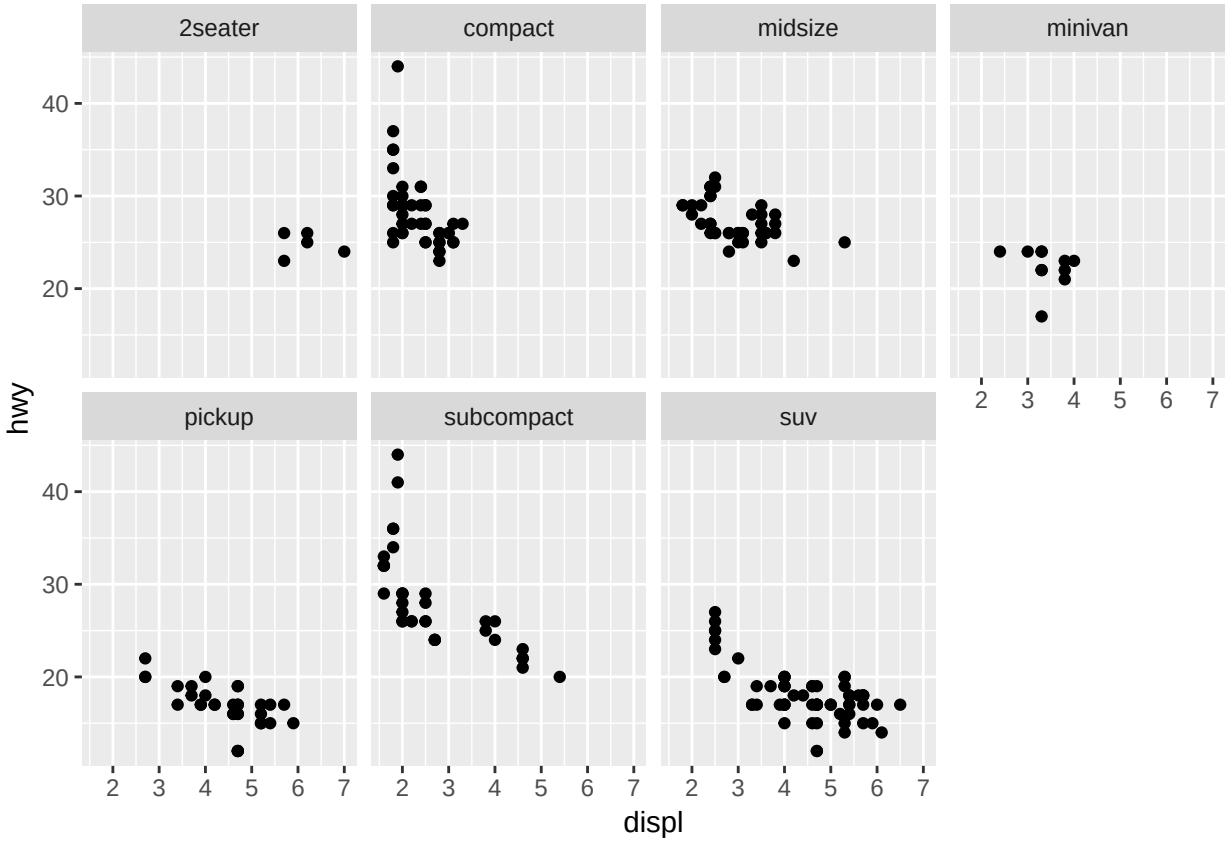
Common Problem

Make sure the + is at the end of the line.

Facets

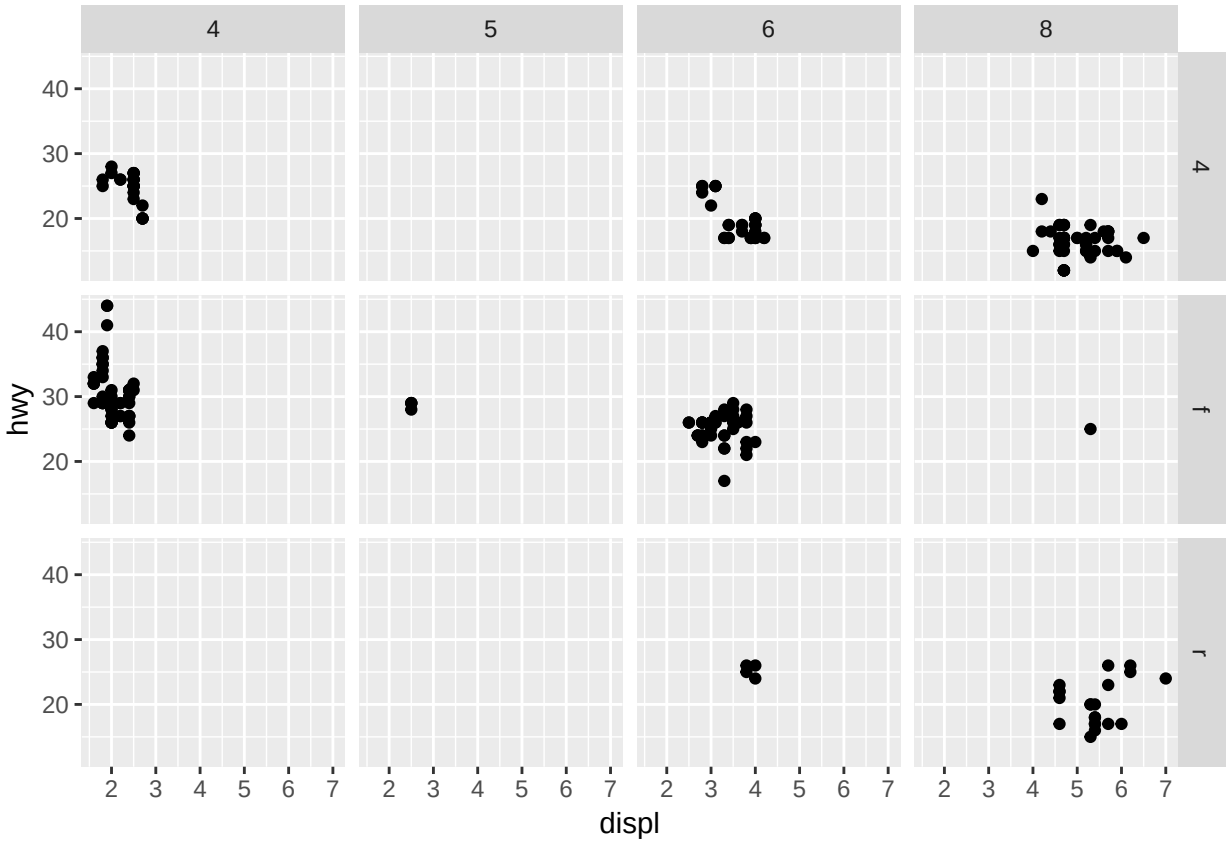
Split data into separate plots - one for each category of a categorical variable. Use facet_wrap() which needs a formula for an argument. Formulas (equations) are given using ~ ; variable passed to facet_wrap should be discrete

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



Use facet_grid to plot on a comination of two variales

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



Exercises:

1) What happens if you facet on a continuous variable?
2) What do empty cells in the previous plot signal?
3) What does . do? For example, try these:

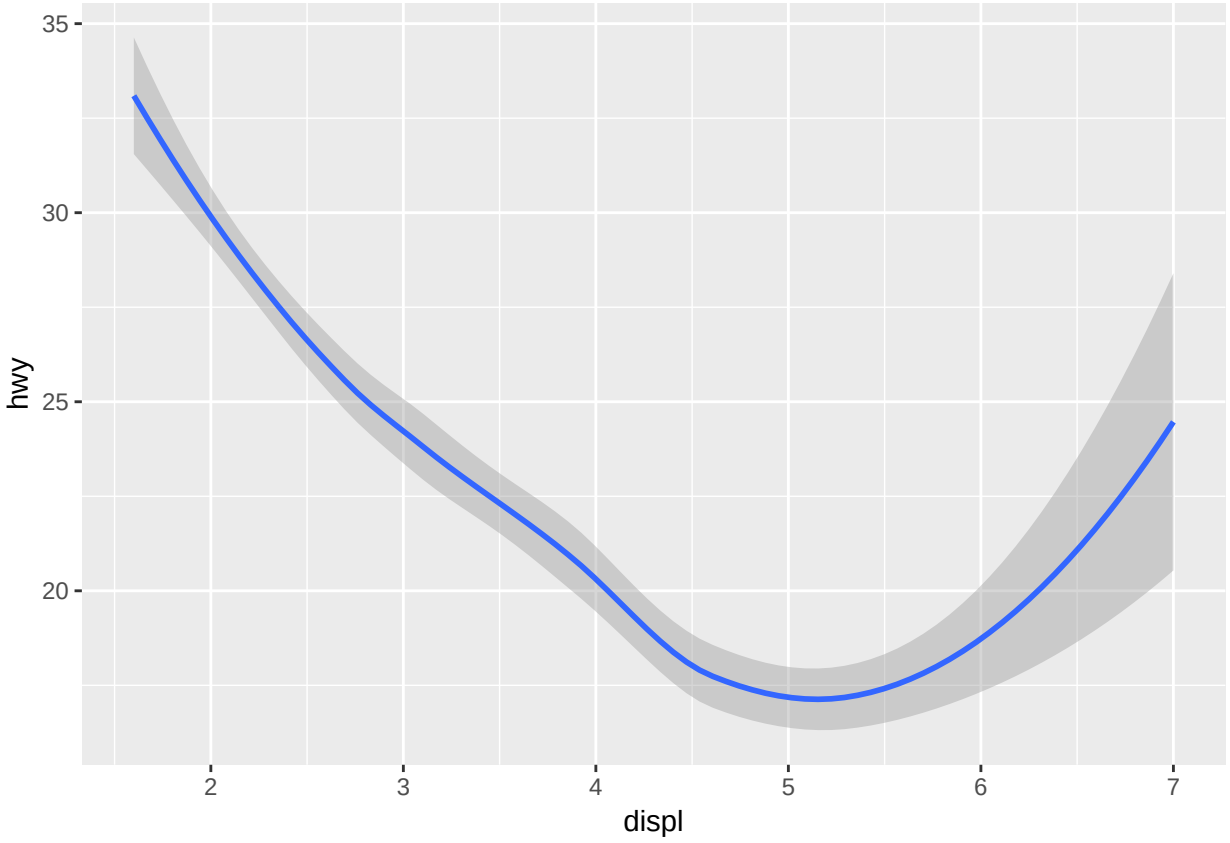
   ggplot(data=mpg) + geom_point(mapping=aes(displ,hwy)) + facet_grid(drv ~ .)

   ggplot(data=mpg) + geom_point(mapping=aes(displ,hwy)) + facet_grid(. ~ cyl)
4) Read ?facet_wrap. What does nrow do? What does ncol do? Why doesn't facet_grid have these options?

Geometric Objects

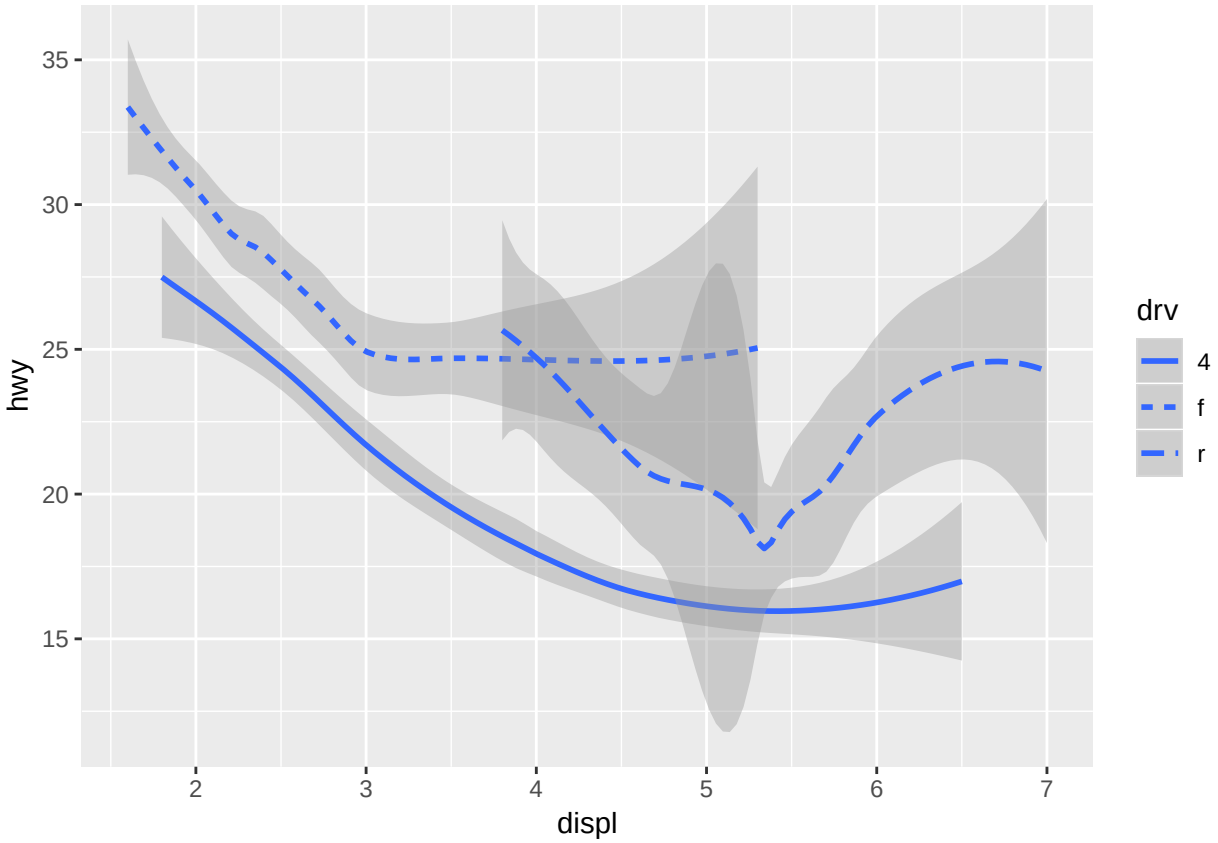
A geom is a geometrical object that a plot uses to represent the data. Examples include bars, lines, points, boxplots, etc. Let's try changing our geom in the above examples from point to smooth.

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



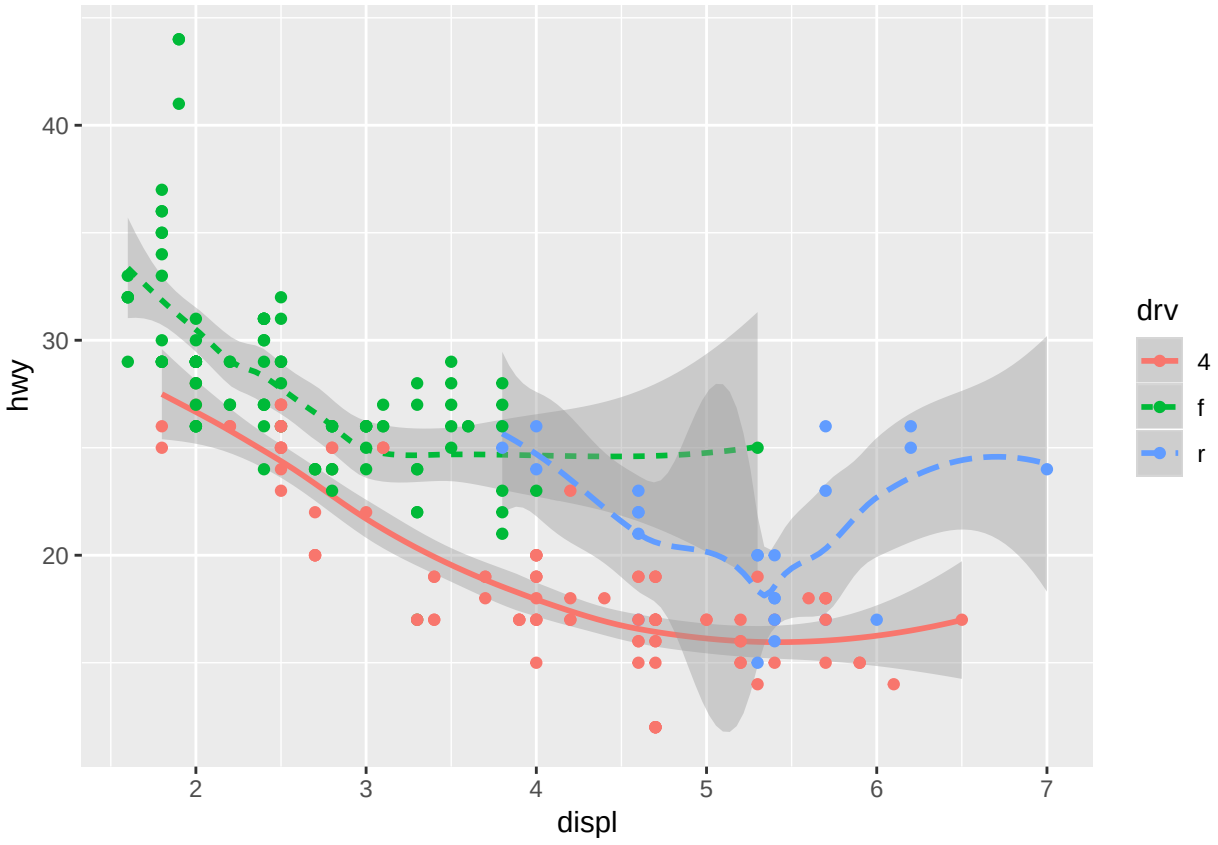
Note that not every aesthetic works with every geom. For instance, shape works for points, but not lines and linetype works for lines and not points.

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



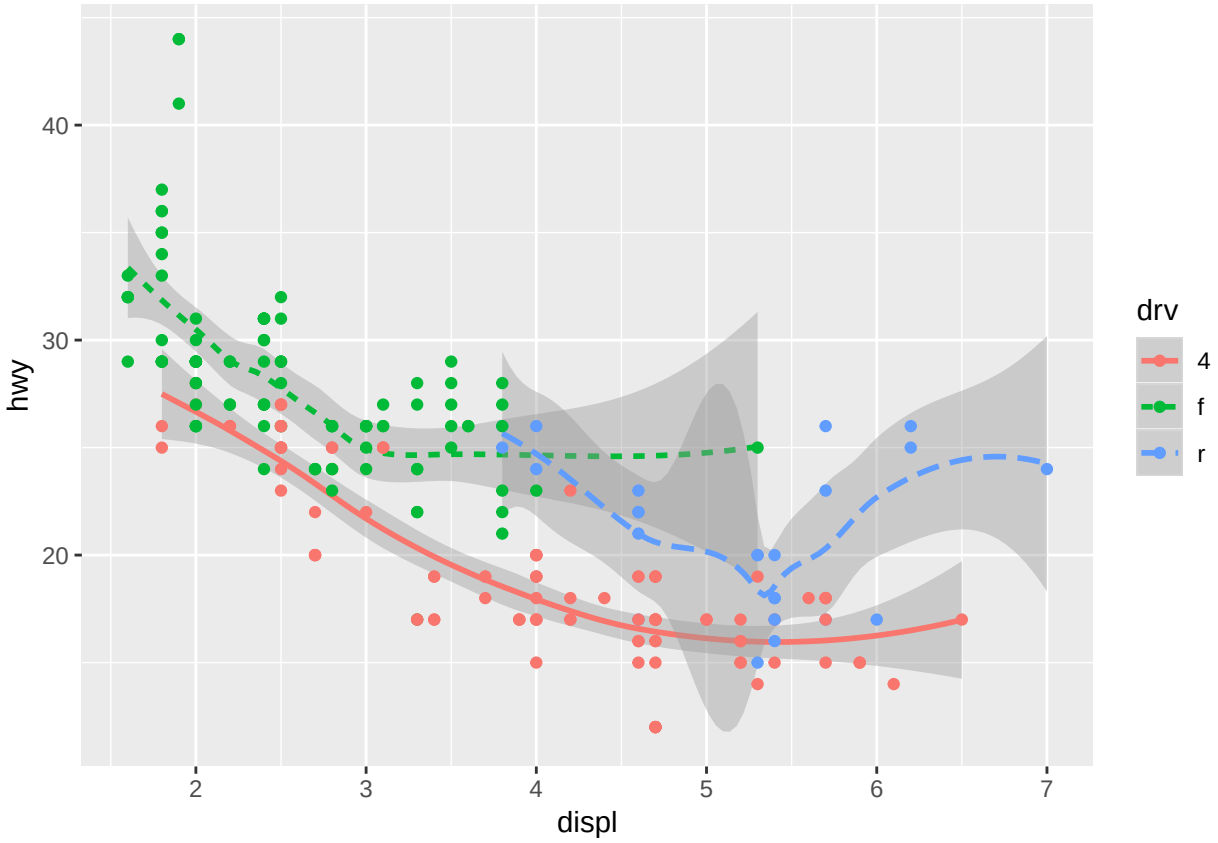
Maybe this looks strange without showing the points. See if you can add the points into the graph and color code both the points and the lines according to drv.

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



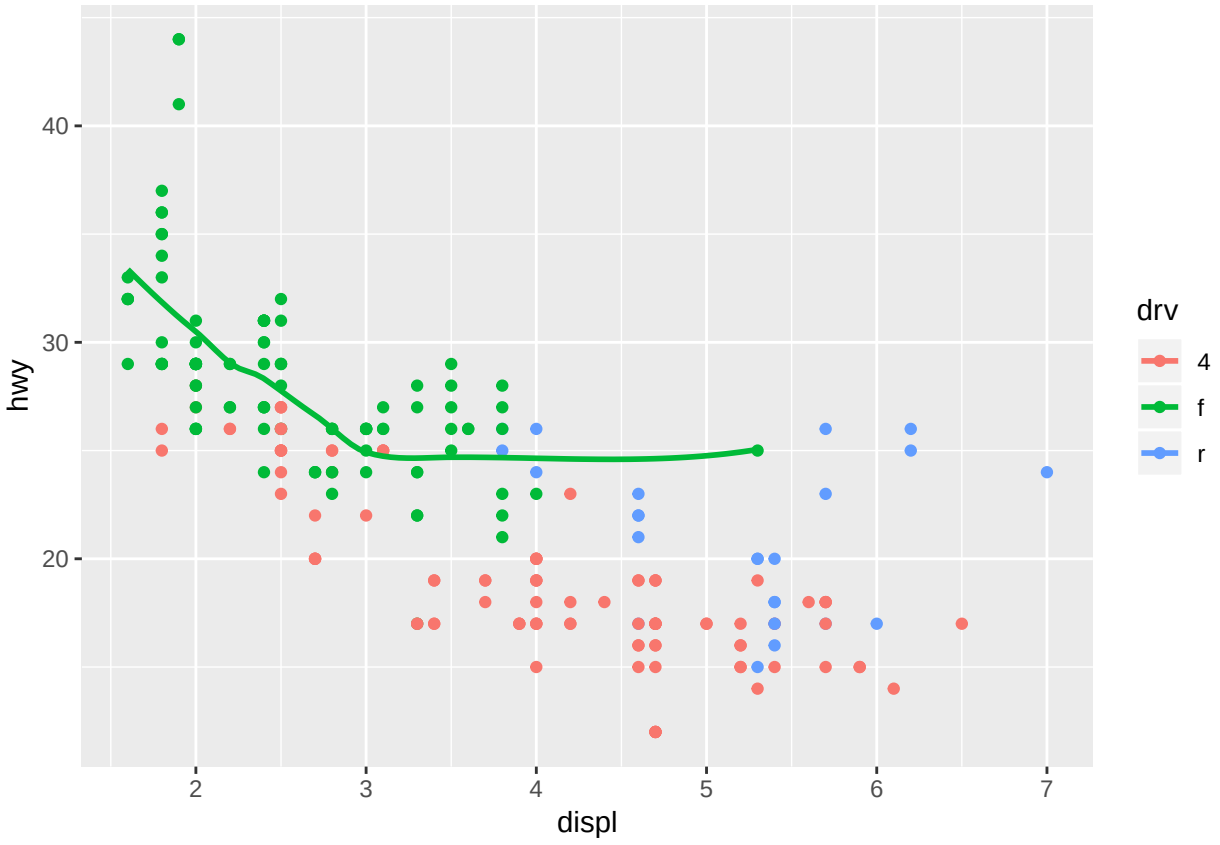
However, this was repetitious. We could instead set aesthetics for all the geoms globally in the ggplot function and then use any aesthetics in specific geoms to overwrite or add to what we specified globally. Try it.

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



We could add just one trend line to the data if we want. Here is how to add just the trend line for front wheel drive cars

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

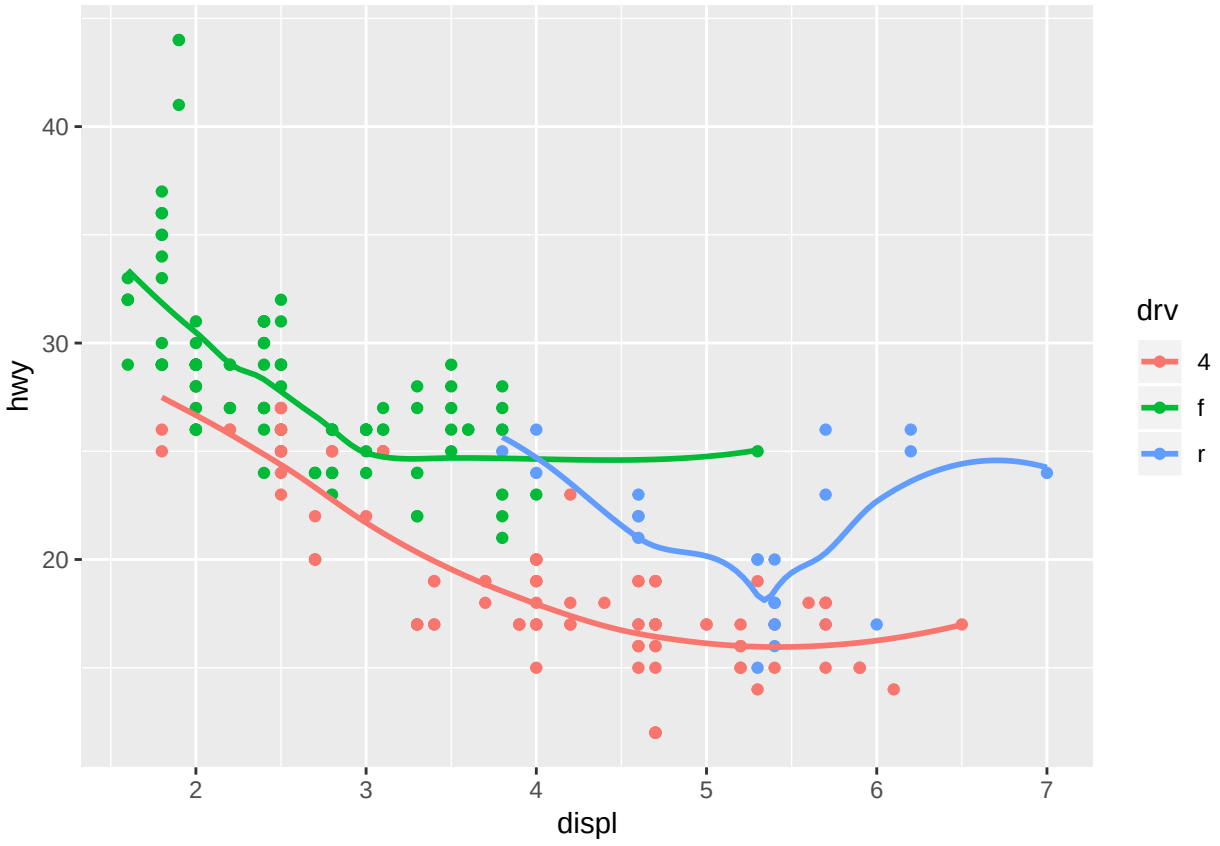


Exercises:

1) What geom would you use for a line chart? A boxplot? A histogram?

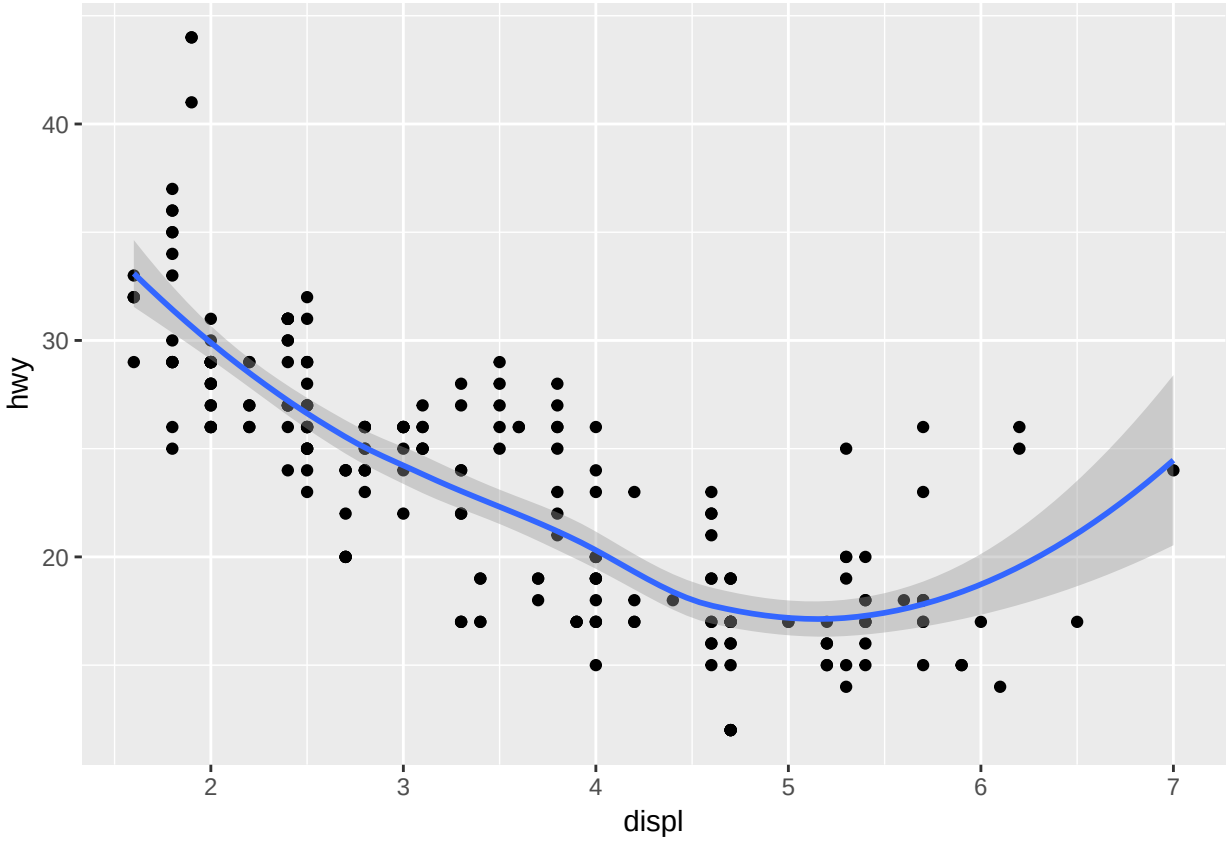
2) Run this code and discuss:

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

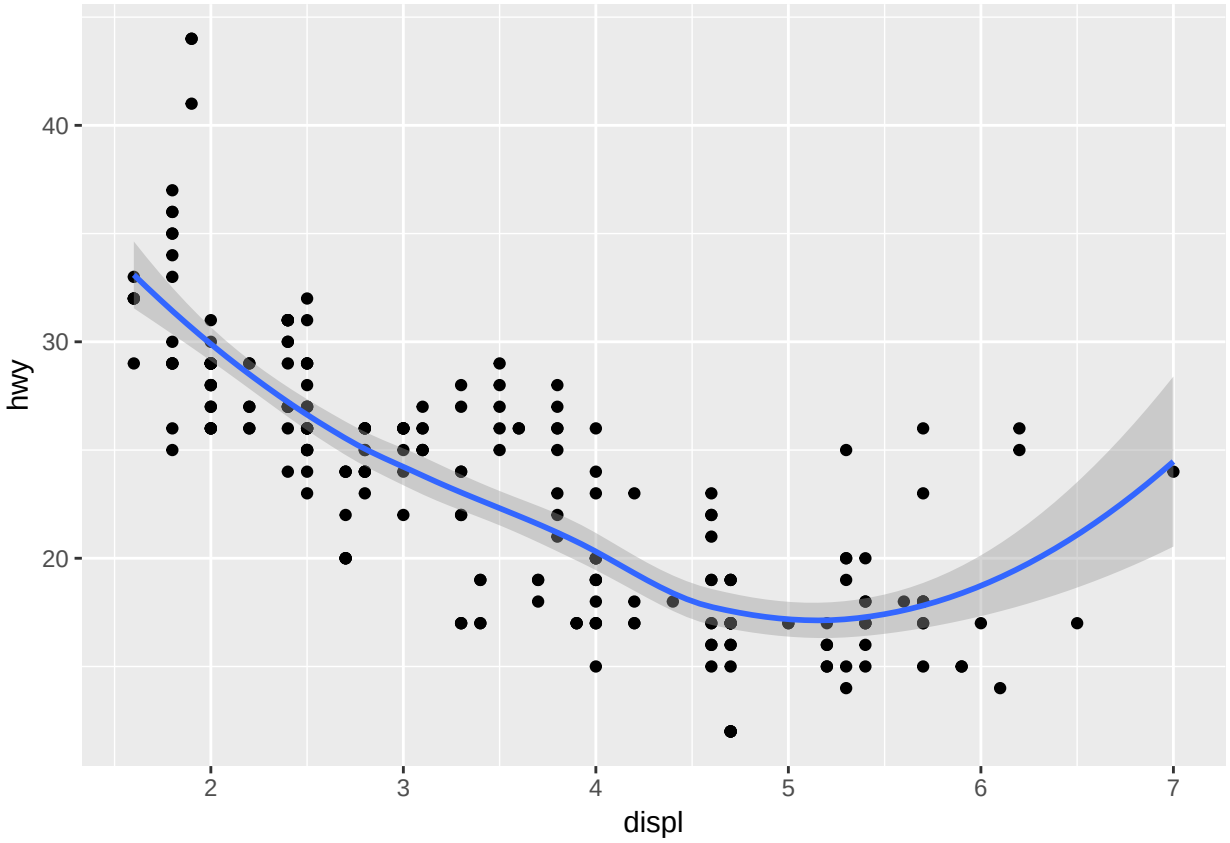


3) Are the following two bits of code equivalent?

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



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

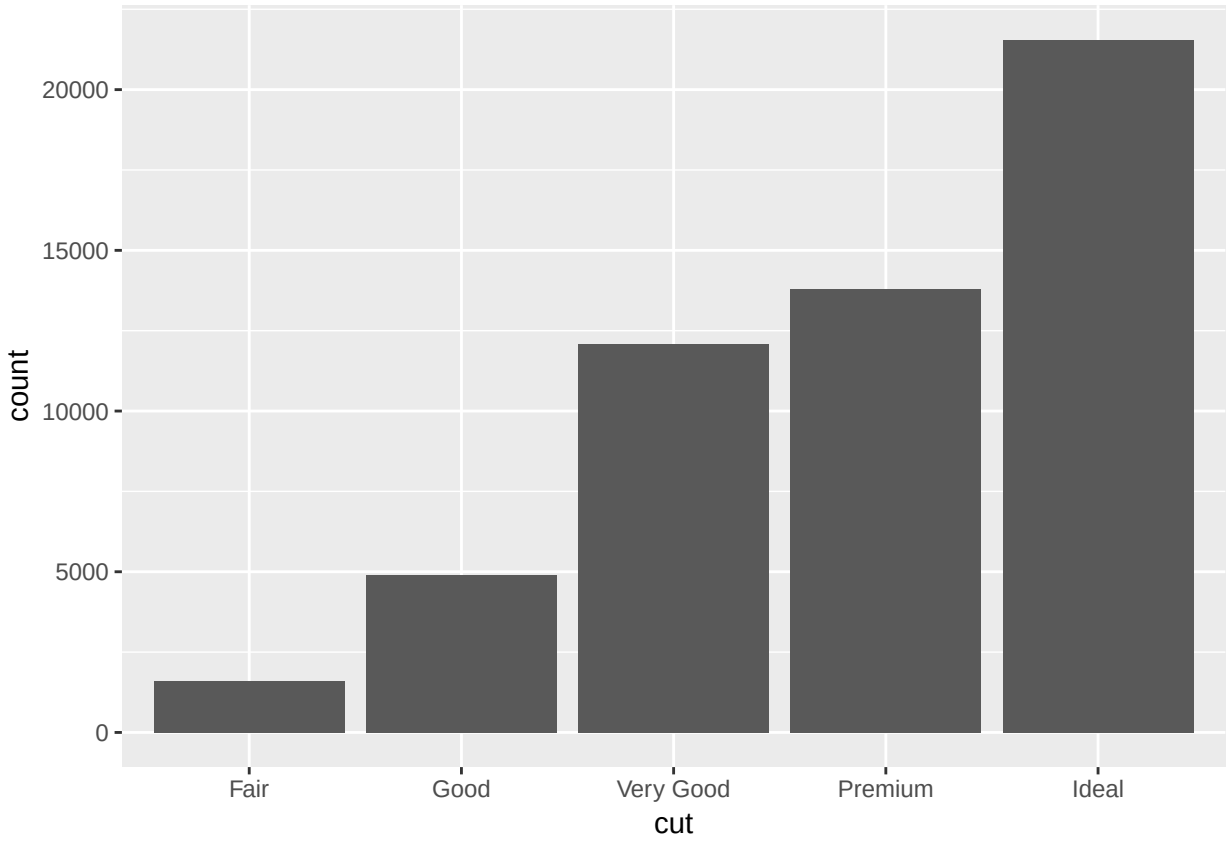


Statistical Transformations

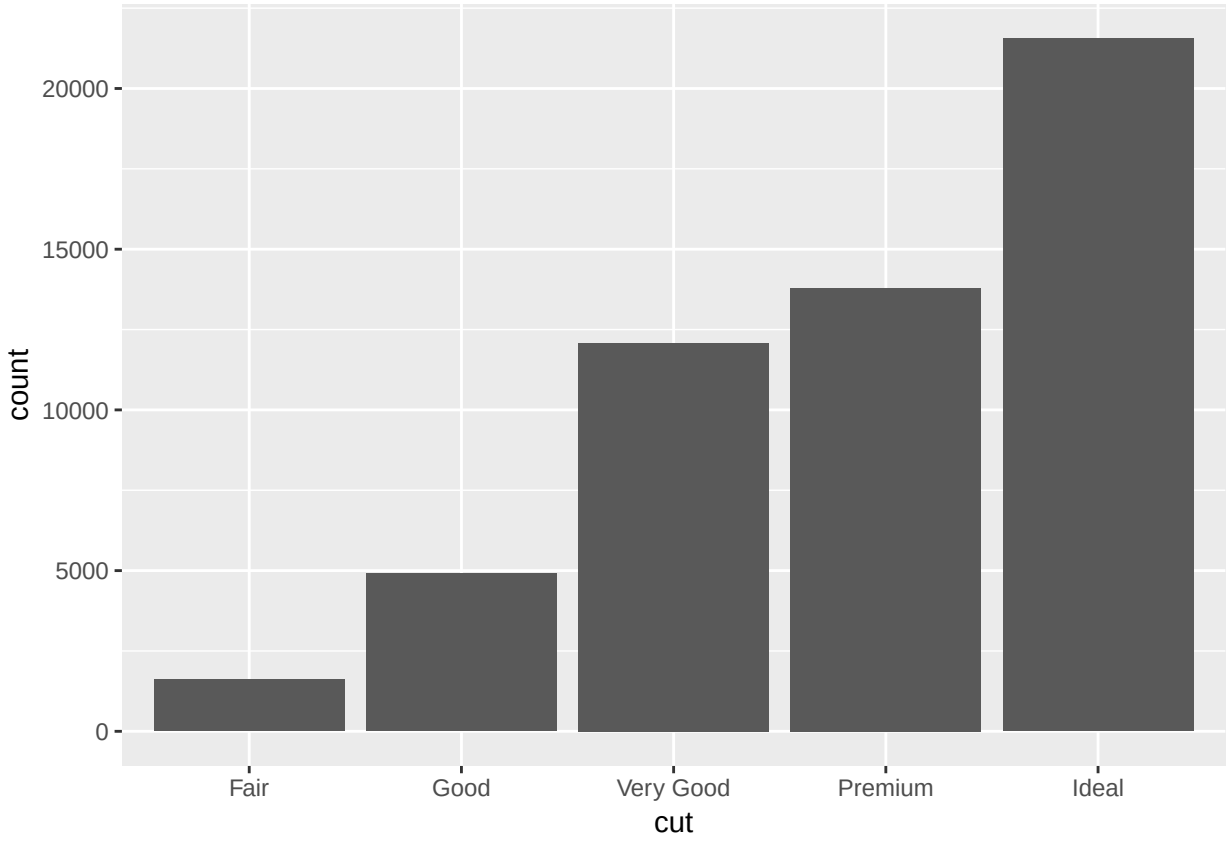
Some geoms use multiple rows to create an intermediate result which is then plotted. For instance, bar chart uses that statistical function count and then plots the results. Similarly, histograms and frequency polygons also bin your data and then count the number of rows that fall into each bin. Smoothers fit a model to the data and then plot predictions. Boxplots use a statistical summary of the data and then plot the results.

Stat (statistical transformation) is the way the new (intermediate) values are calculated. Look at ?geom_bar and look for the section on stat. Each stat has a default geom and each geom has a default stat, so they can generally be interchanged if desired.

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



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

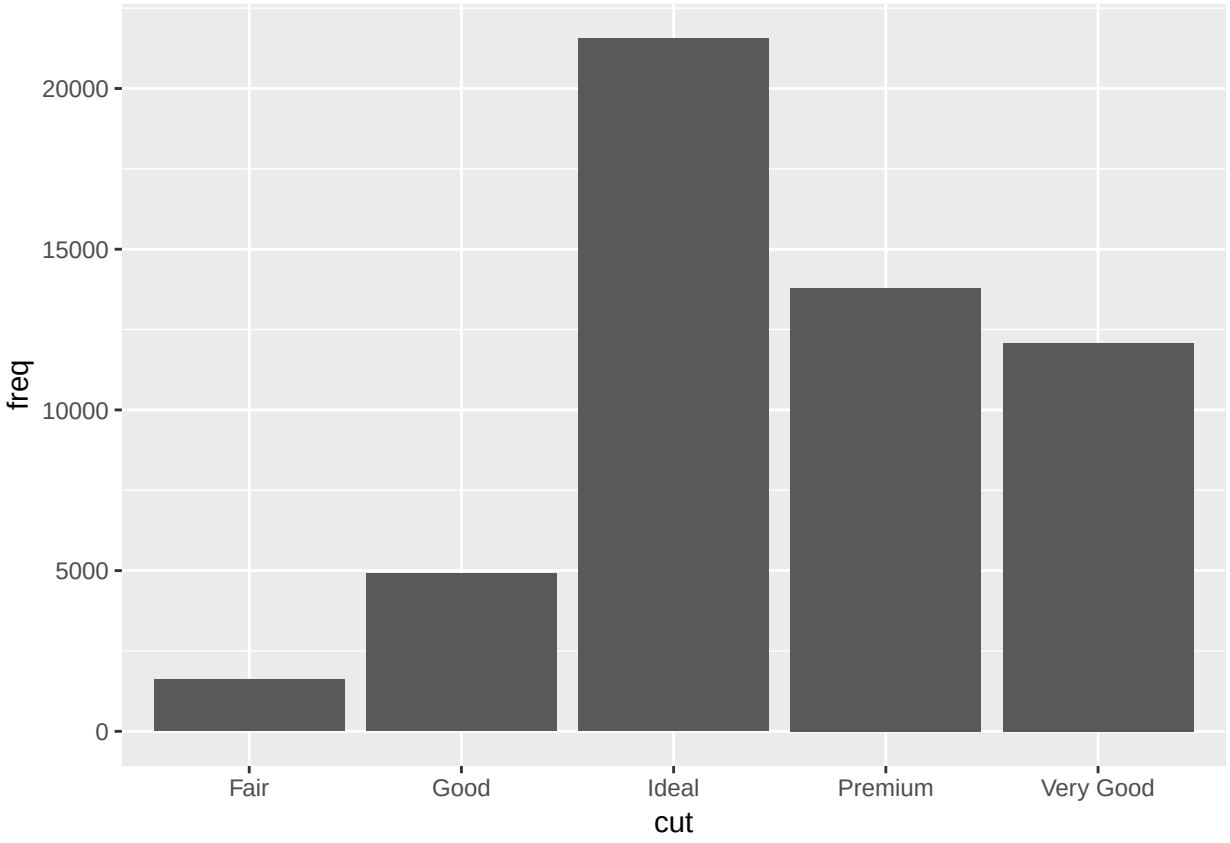


If you have the actual values that you want to plot already in a table, you can use the identity stat to pick up those values.

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

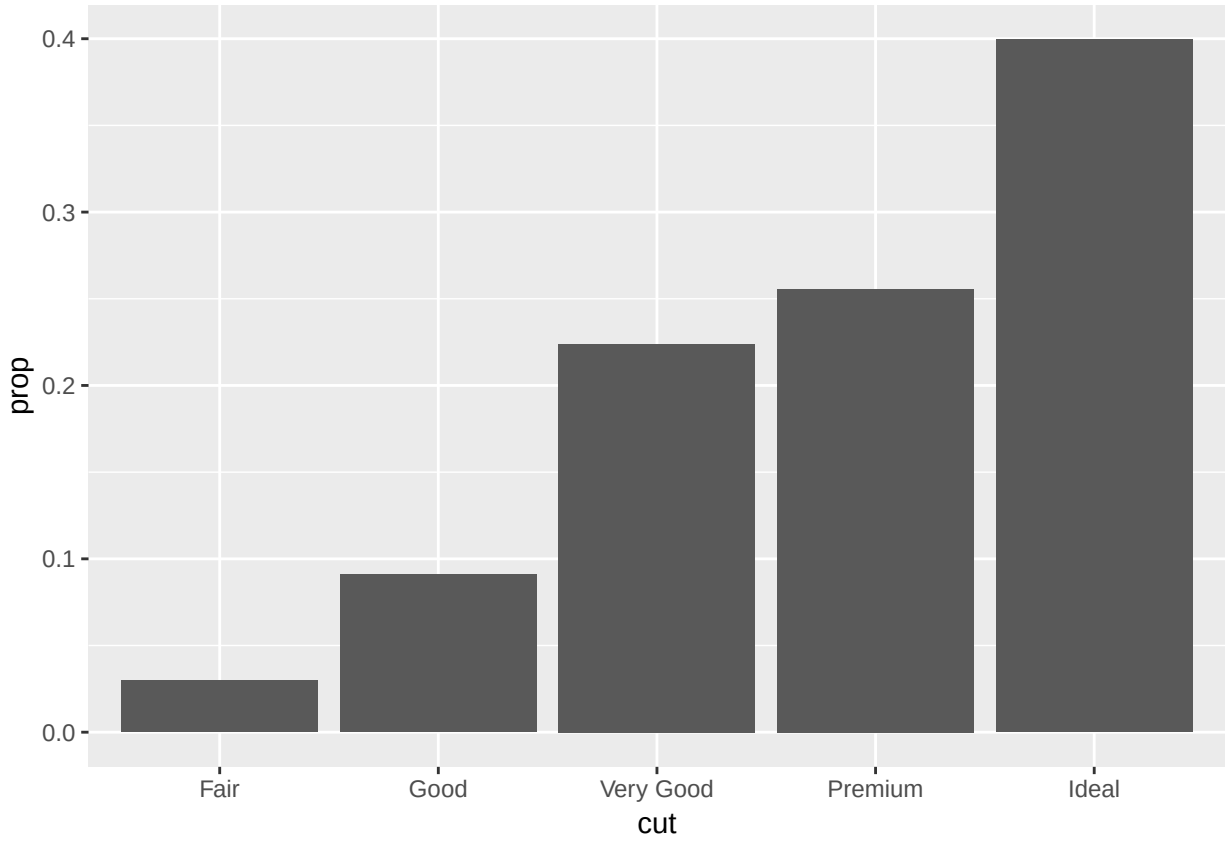
```
## # A tibble: 5 x 2
##   cut        freq
##   <chr>     <dbl>
## 1 Fair       1610
## 2 Good       4906
## 3 Very Good 12082
## 4 Premium   13791
## 5 Ideal     21551
```

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



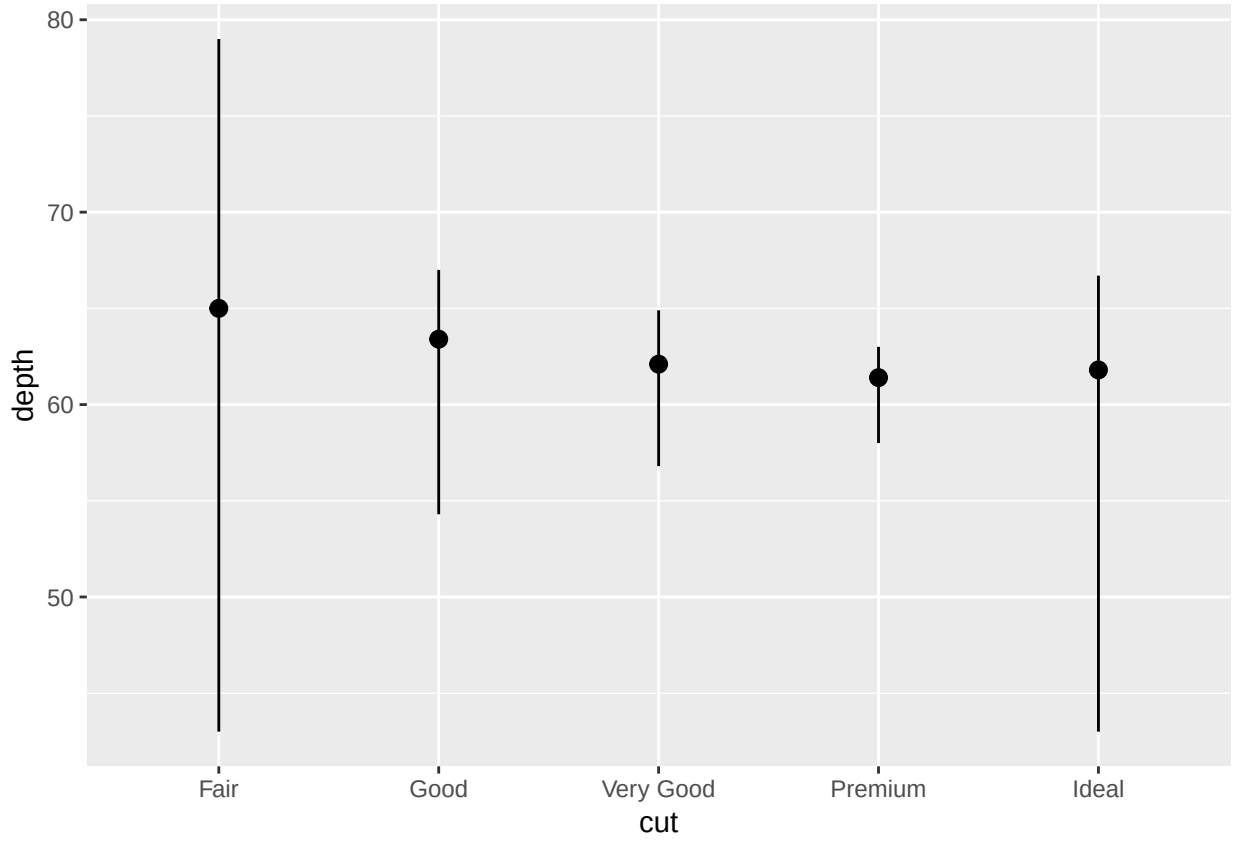
If instead you want to see the proportion, try overriding the stat value.

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



Check out the ggplot2 cheatsheet for a list of stats that you can use. To see what a particular one does (e.g. stat_bin), use ? (like ?stat_bin) Look up stat_summary and see what it does. How does that map to the following code?

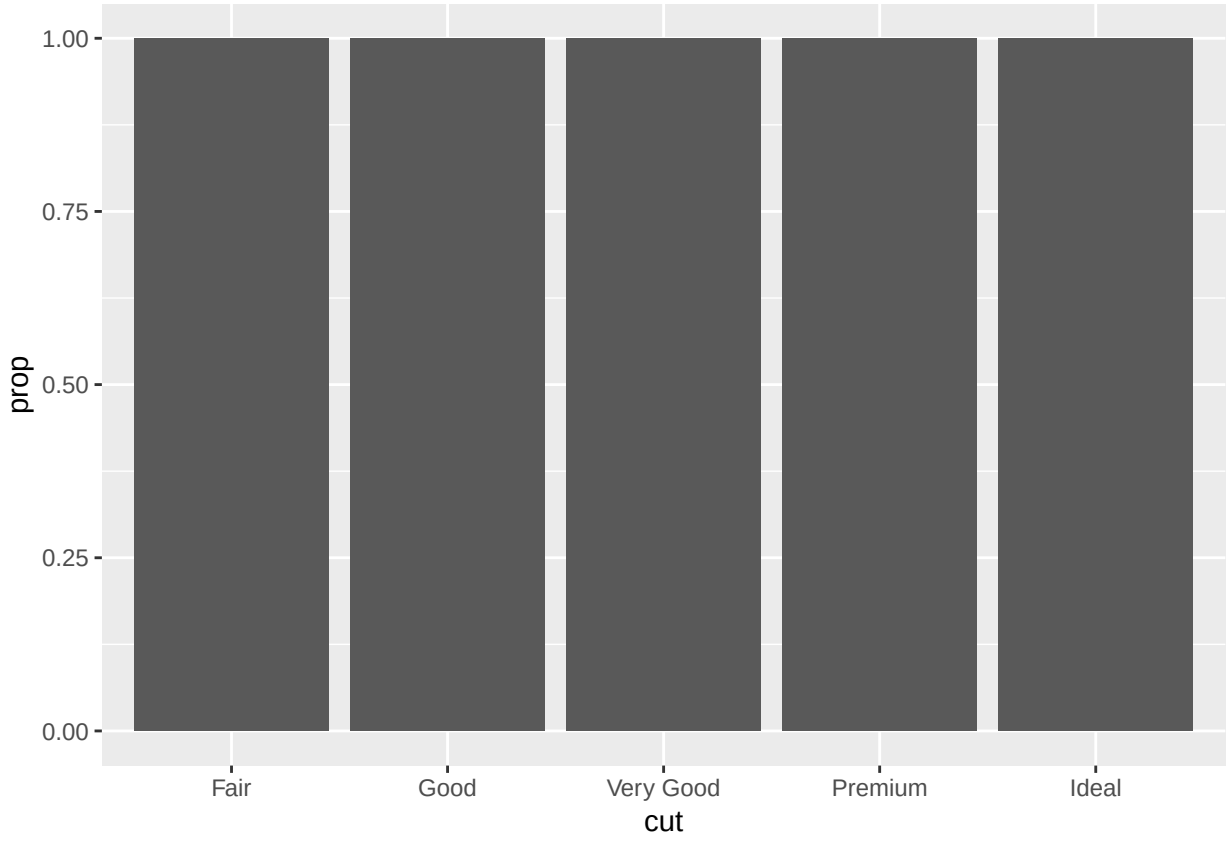
```
ggplot(data = diamonds) +
  stat_summary(
    mapping = aes(x = cut, y = depth),
    fun.ymin = min,
    fun.ymax = max,
    fun.y = median
  )
```



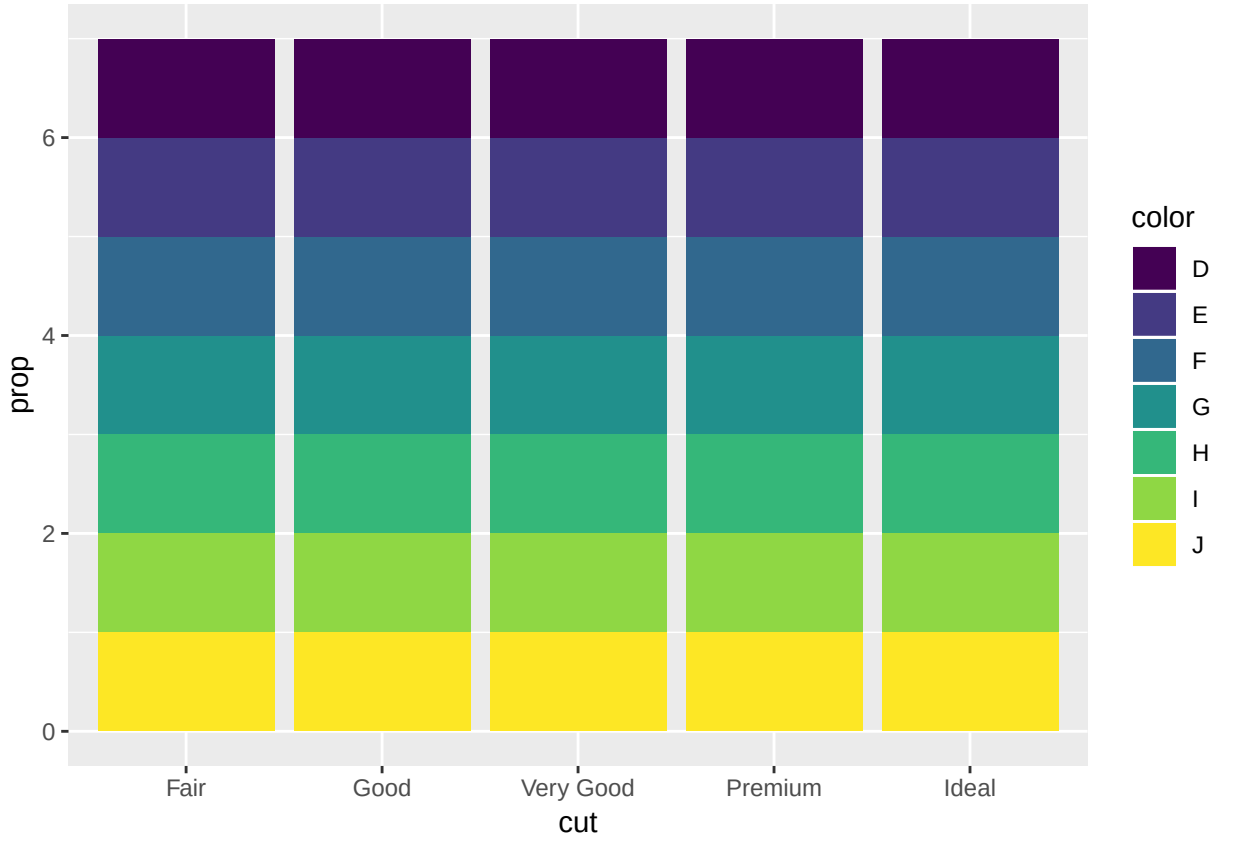
Exercizes

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?
2. What does geom_col() do? How is it different to geom_bar()?
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?
4. What variables does stat_smooth() compute? What parameters control its behaviour?
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..))
```



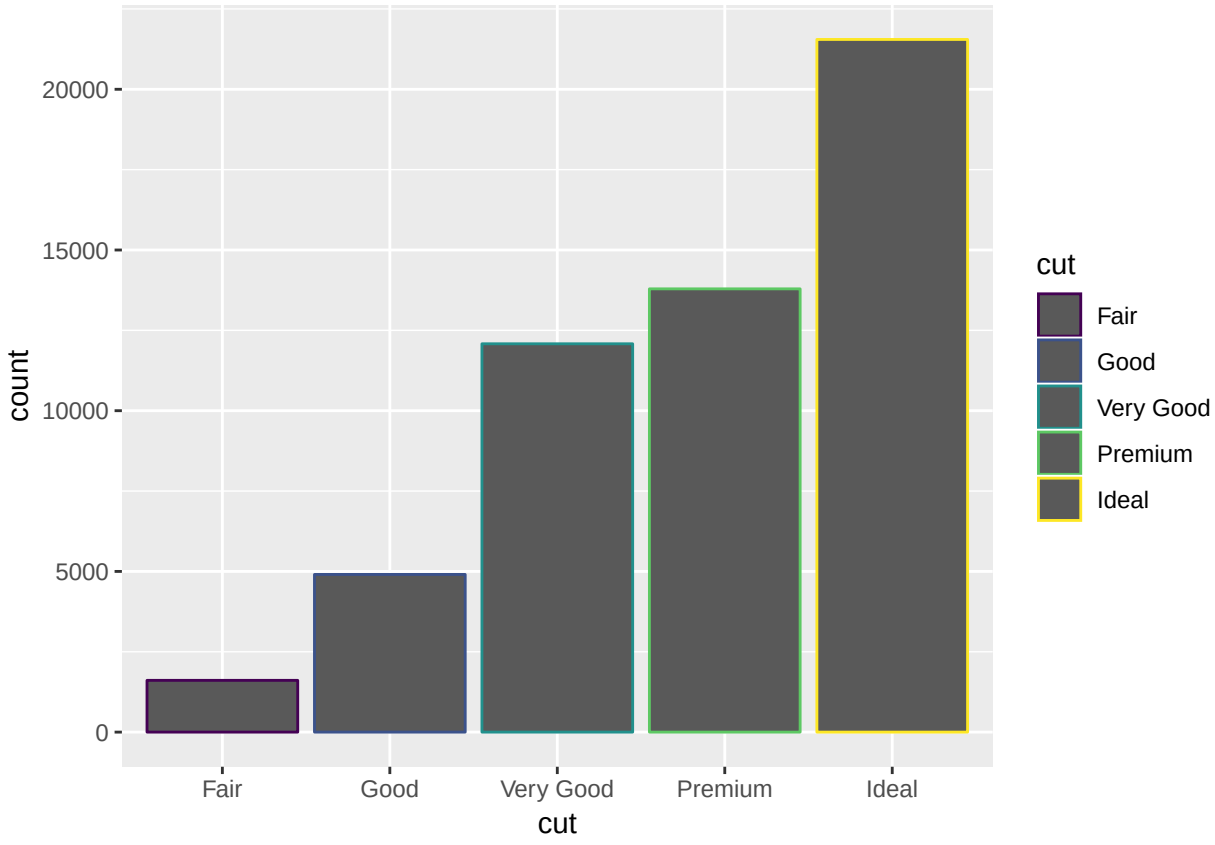
```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = color, y = ..prop..))
```



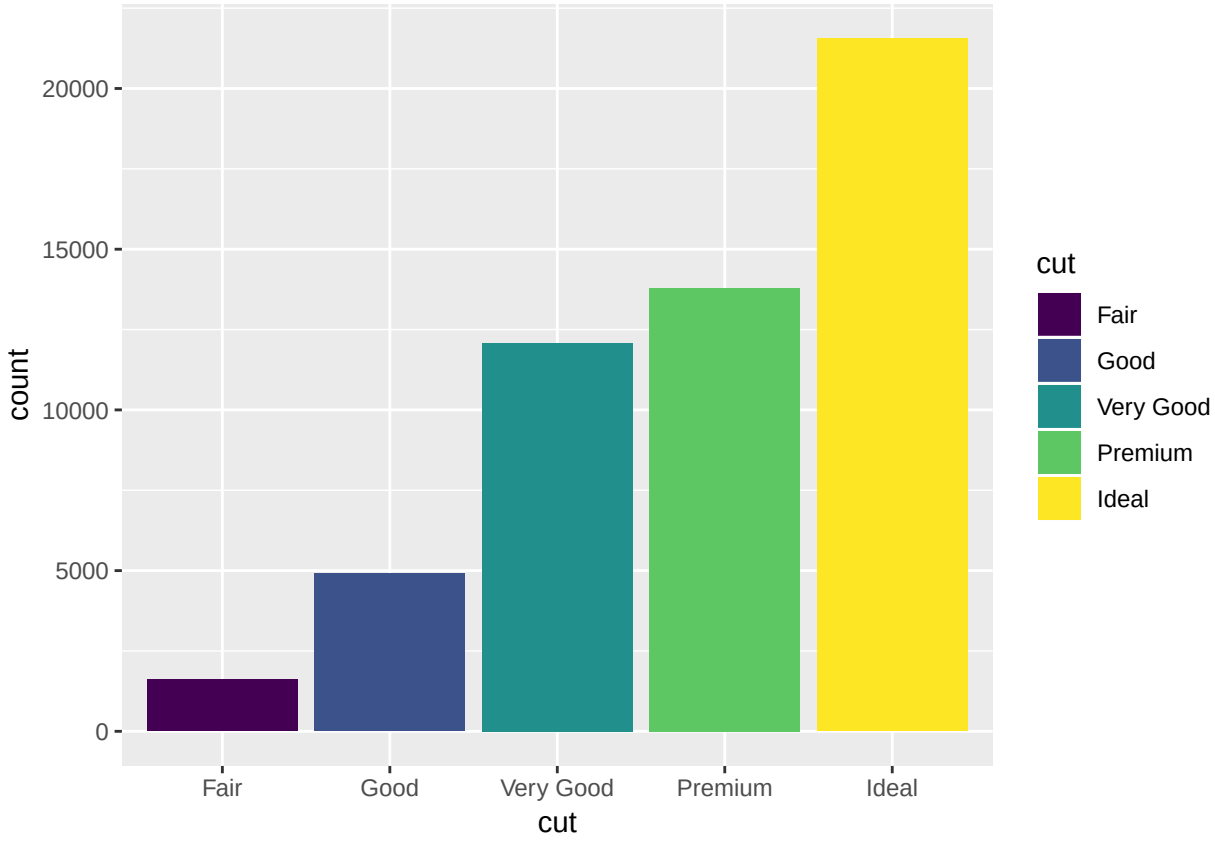
Position Adjustments

We can change the colors of our barplots. If you use the color in the mapping, it will just change the line color. To change the color of the whole bar, use fill instead.

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

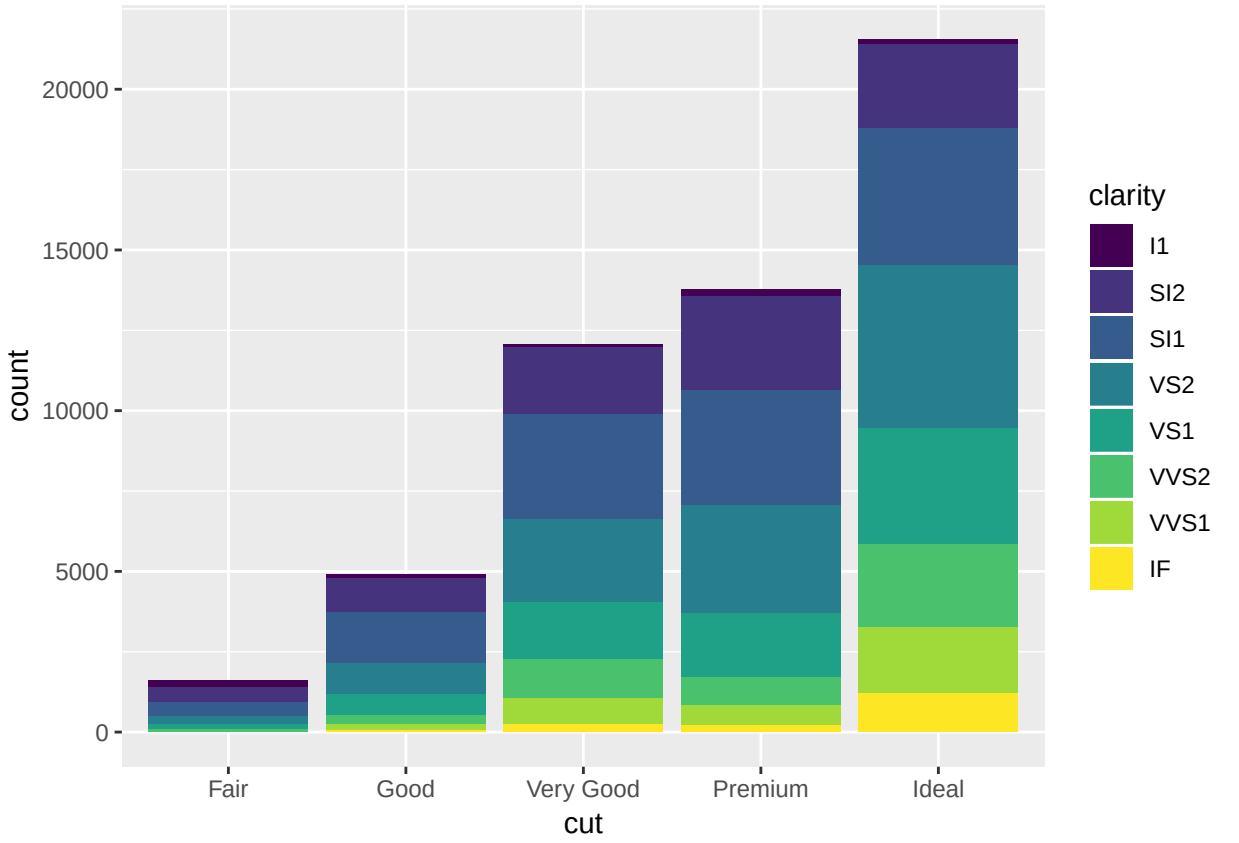


```
ggplot(data=diamonds)+
  geom_bar(mapping = aes(x = cut, fill = cut))
```



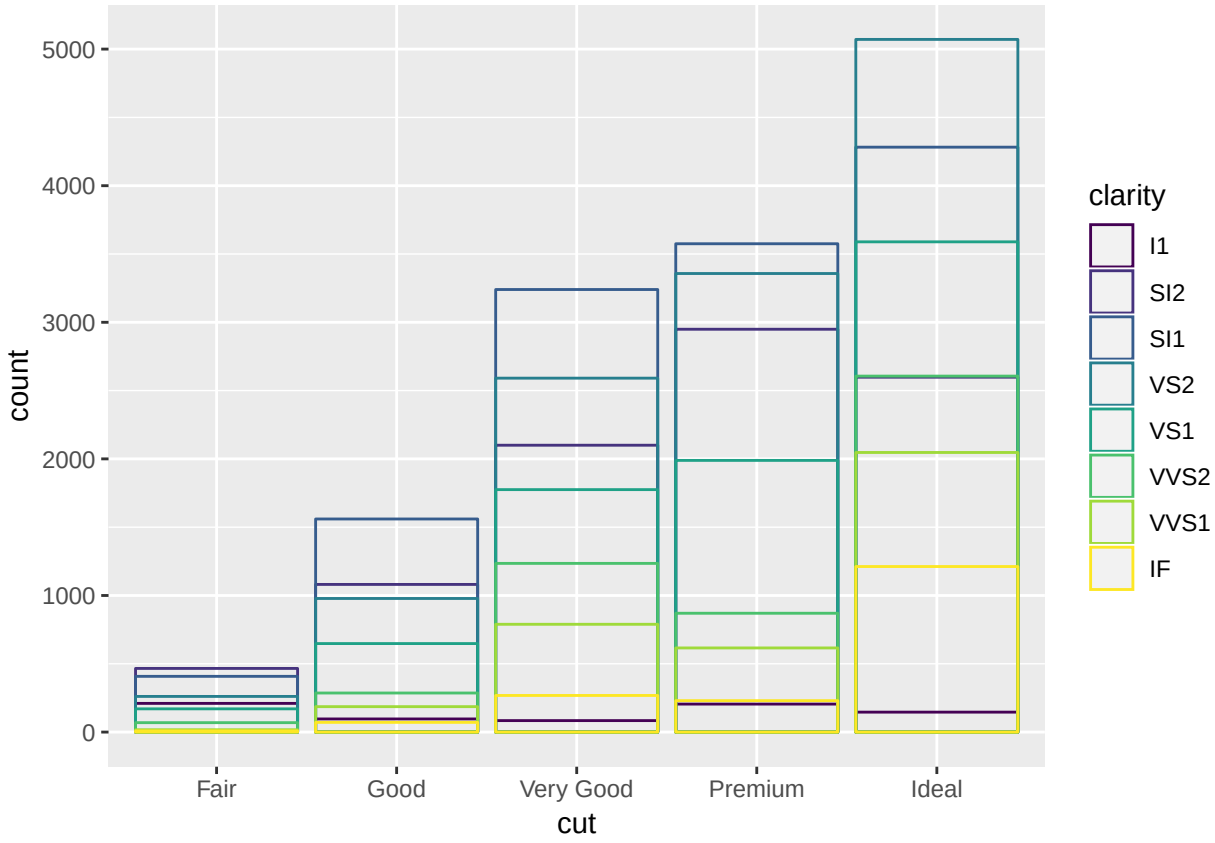
If you use a different variable for the fill aesthetic, you get stacked bars. This is because the default for position adjustment is stacking.

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

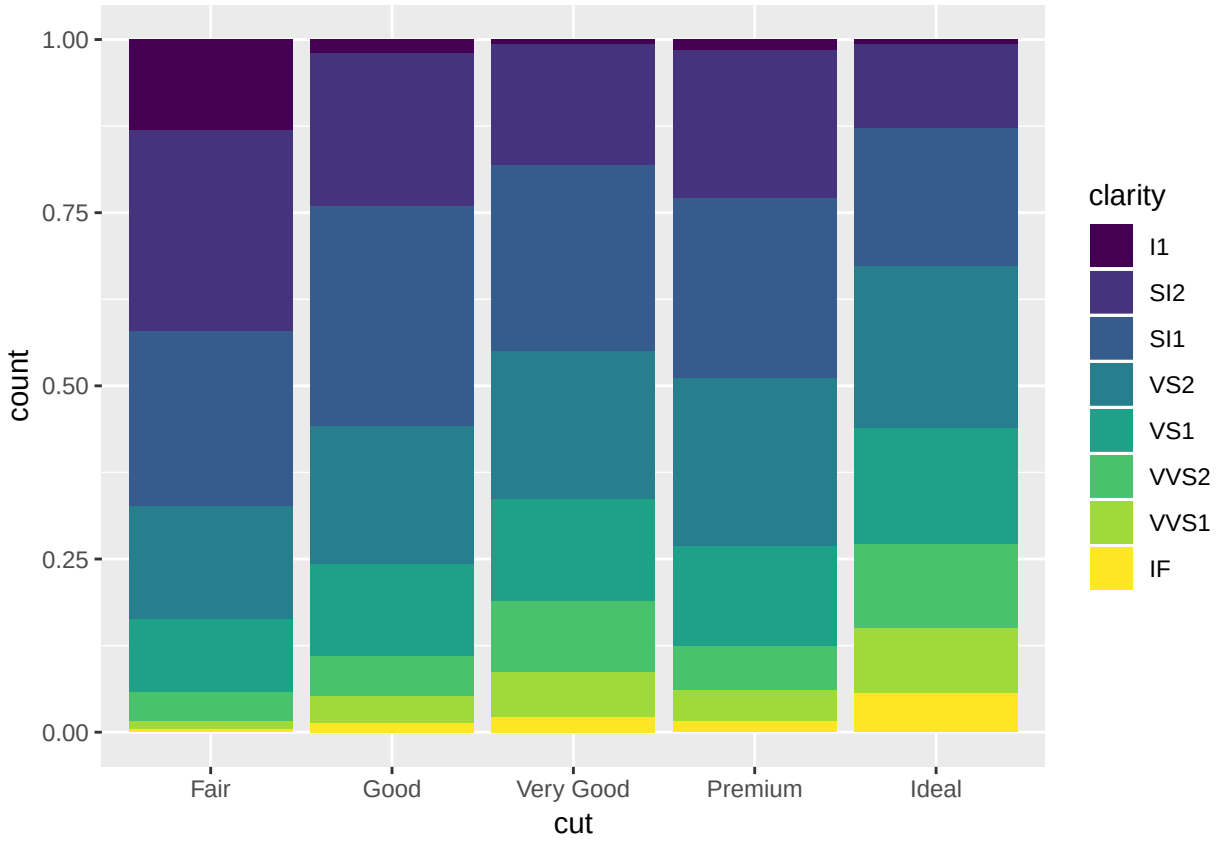


The other 3 options for position are Identity, fill and dodge. Let's see what each one does.

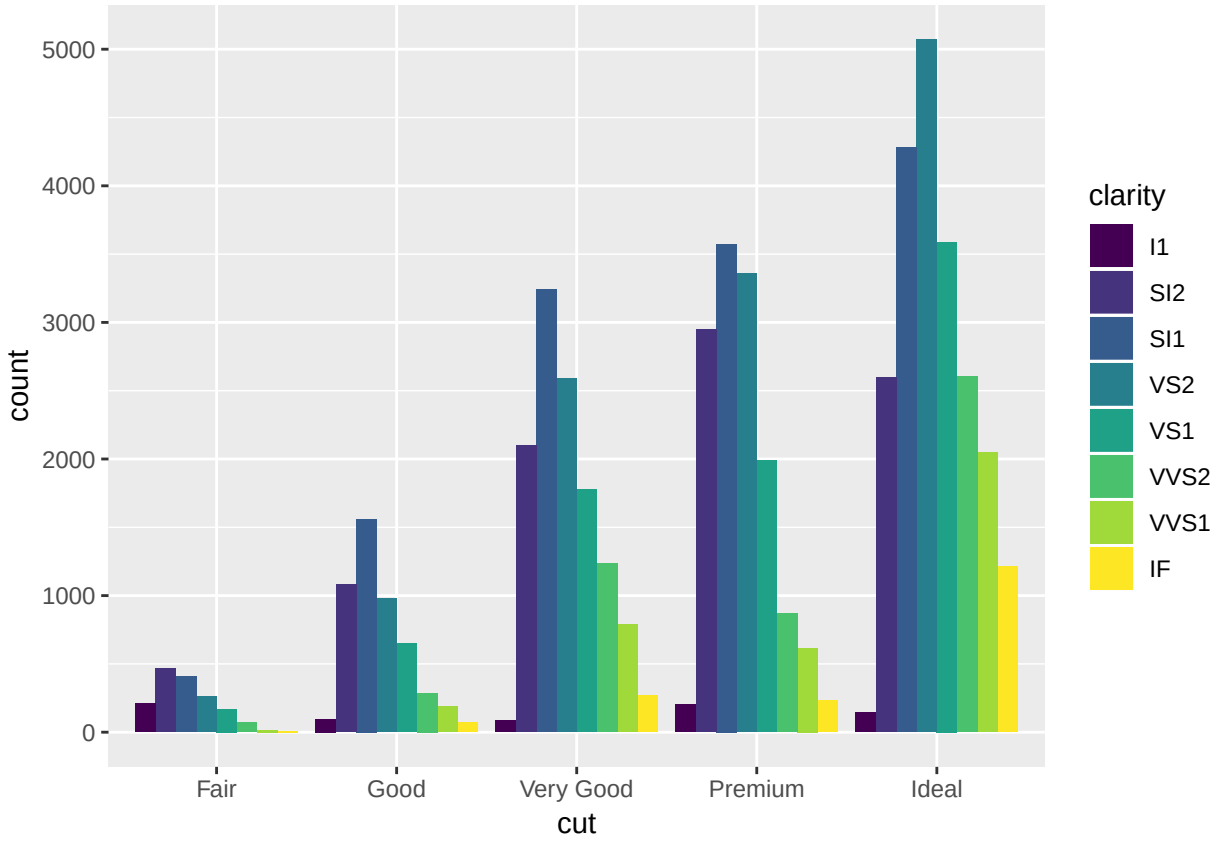
```
ggplot(data = diamonds,
       mapping = aes(x=cut, color = clarity)) +
  geom_bar(fill=NA, position = "identity")
```



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

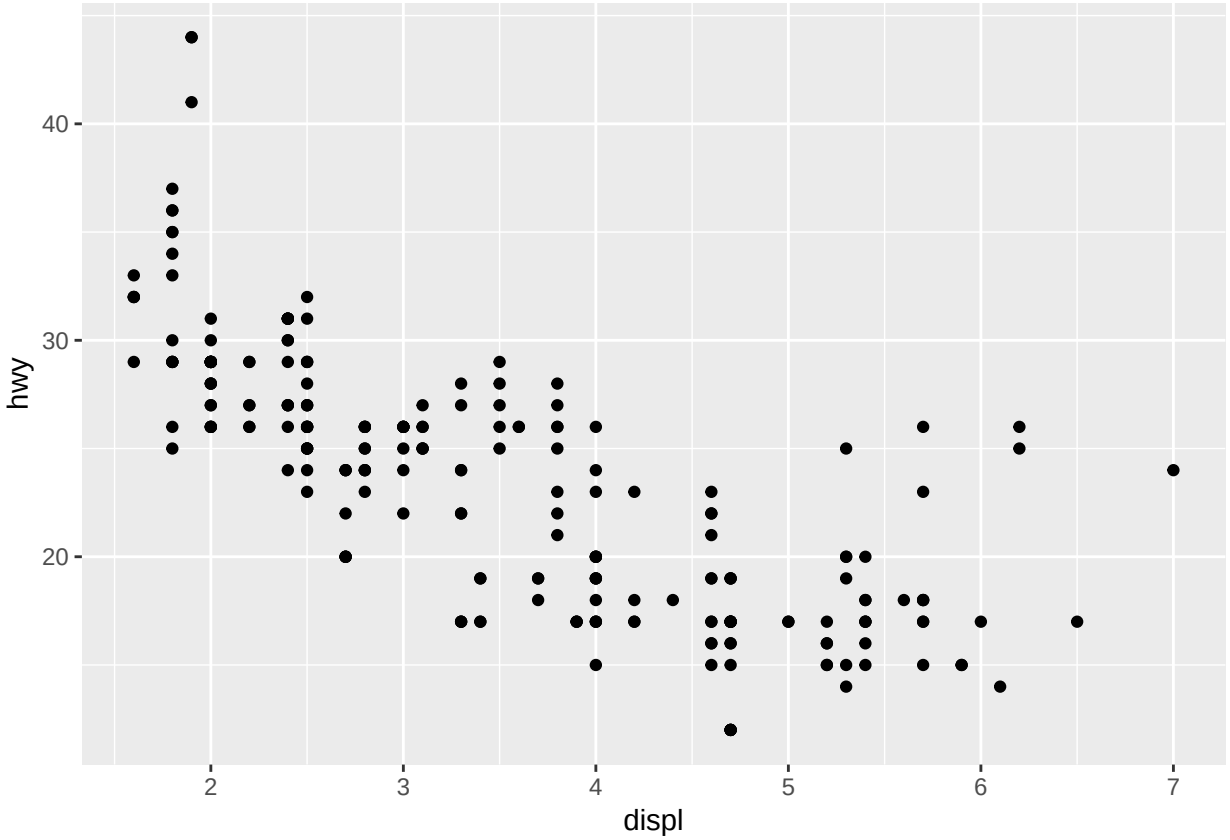


```
ggplot(data = diamonds,
       mapping = aes(x=cut, fill = clarity)) +
  geom_bar(position = "dodge")
```

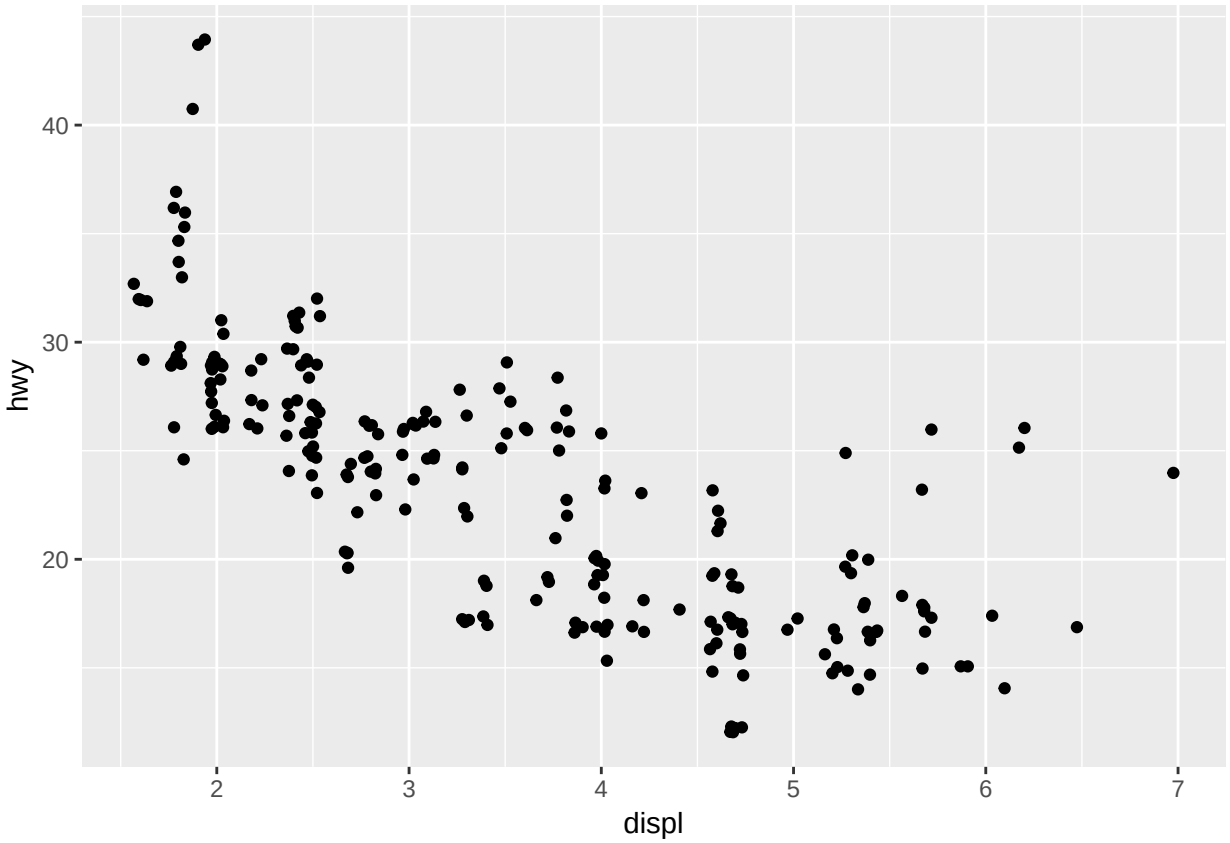


One final position adjustment is sometimes useful for points, and that is jitter.

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



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

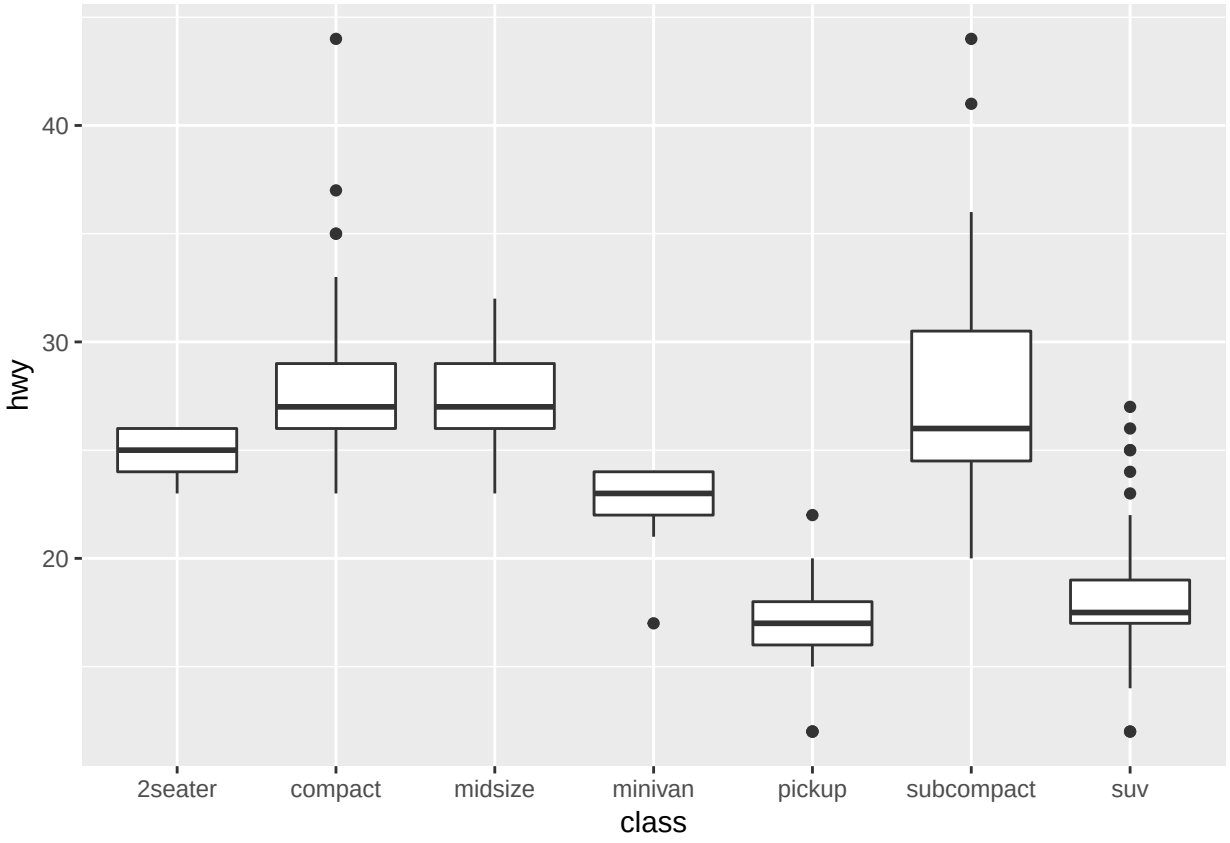


Coordinate Systems

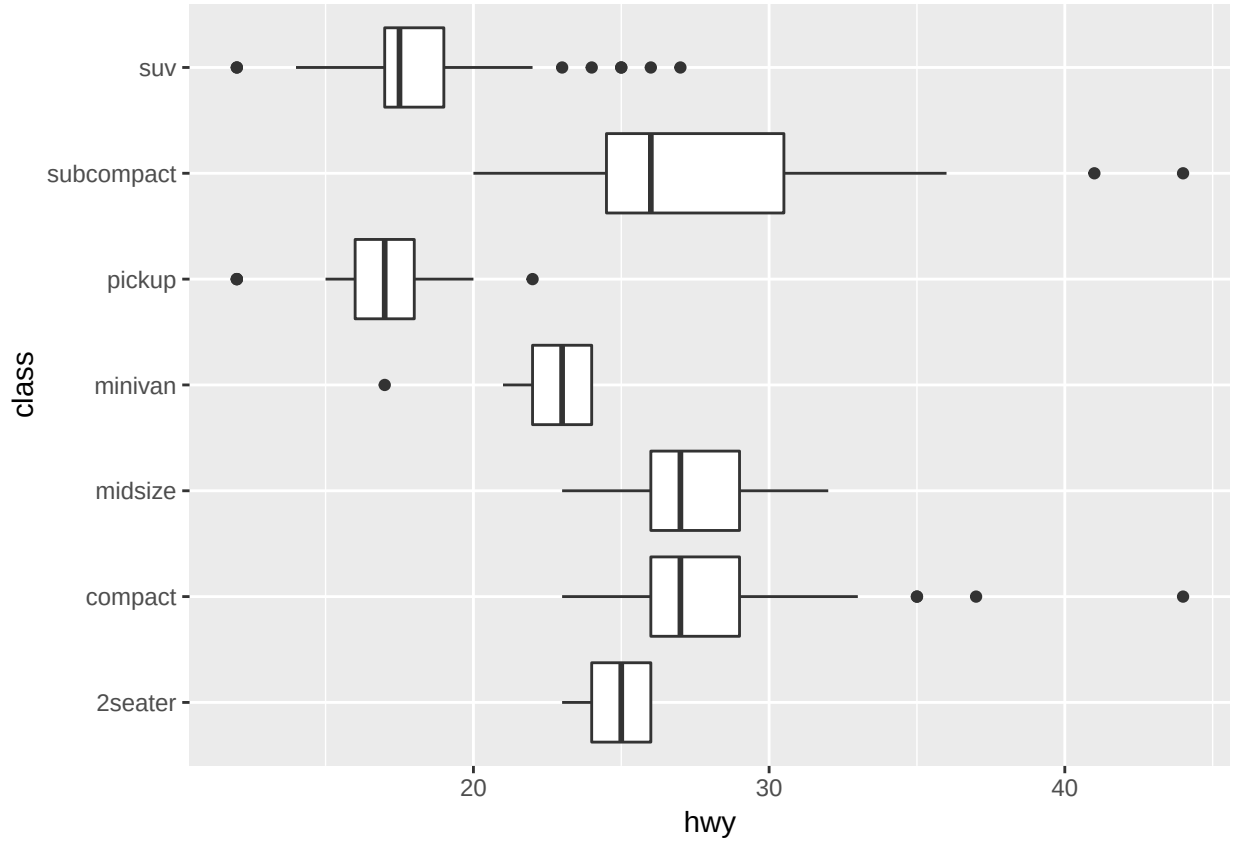
Default is the Cartesian coordinate system with x and y positions acting independently to determine the location of each point.

A few other coordinate systems include * coord_flip (flips x and y axes) * coord_quickmap (sets aspect ratio for maps) * coord_polar (uses polar coordinates)

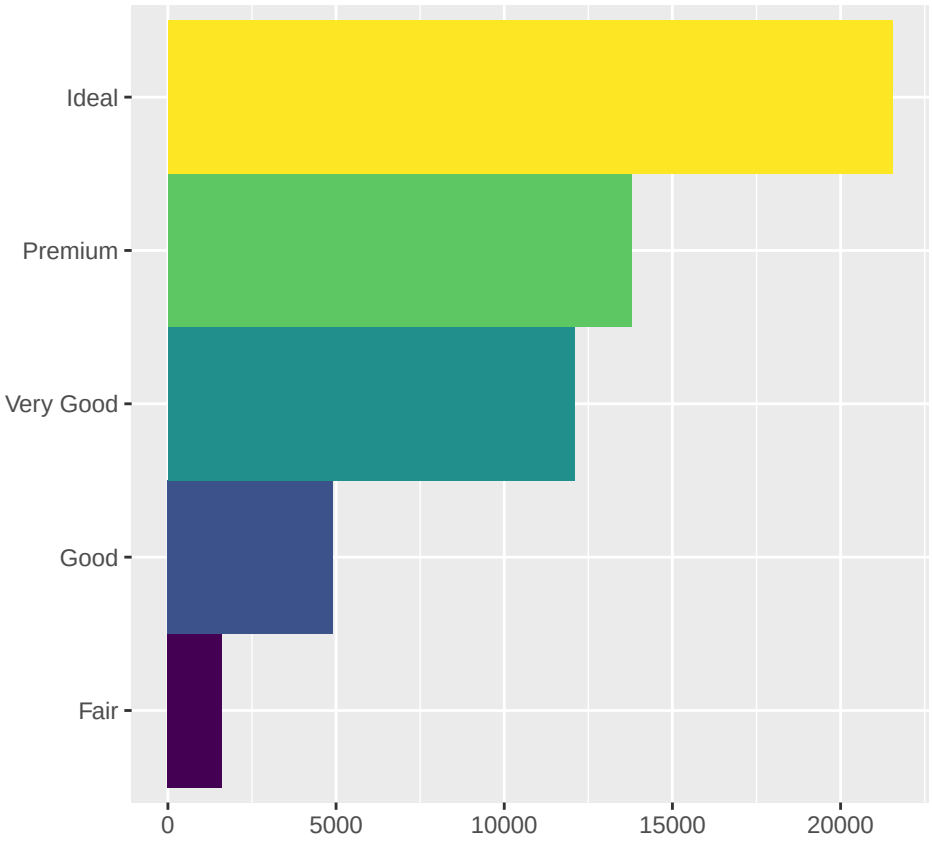
```
boxes <- ggplot(data = mpg,
                mapping=aes(x=class, y =hwy))
boxes + geom_boxplot()
```



```
boxes + geom_boxplot() + coord_flip()
```



```
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)
bar + coord_flip()
```



```
bar + coord_polar()
```