Exploratory Data Analysis 1

Data Analysis and Visualization (Fall 2019)

Instructor: Debopriya Ghosh

Data Exploration is the art of looking at the data, generating hypotheses, testing them, and repeating the process.

DATA VISUALIZATION

"The simple graph has brought more information to the data analyst's mind than any other device" – John Tukey

Here we will visualize the data using **ggplot2** package

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.2.0     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.1
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

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

The dataset *mpg* contains observations collected by the US Environment Protection Agency on 38 models of cars.

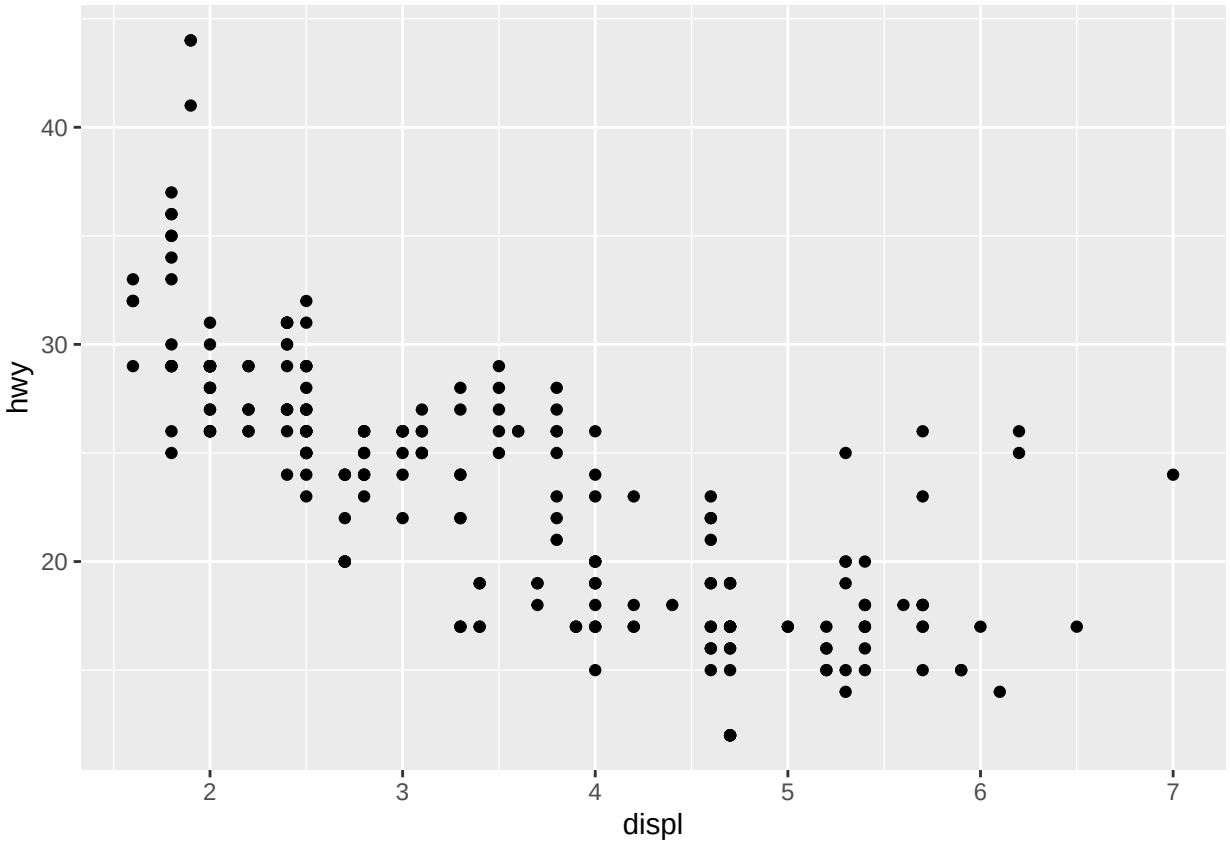
```
data(mpg)
mpg
```

```
## # A tibble: 234 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~ f        18    29 p     comp~
##  2 audi         a4      1.8  1999     4 manu~ f        21    29 p     comp~
##  3 audi         a4      2    2008     4 manu~ f        20    31 p     comp~
##  4 audi         a4      2    2008     4 auto~ f        21    30 p     comp~
##  5 audi         a4      2.8  1999     6 auto~ f        16    26 p     comp~
##  6 audi         a4      2.8  1999     6 manu~ f        18    26 p     comp~
##  7 audi         a4      3.1  2008     6 auto~ f        18    27 p     comp~
##  8 audi         a4 q~   1.8  1999     4 manu~ 4        18    26 p     comp~
##  9 audi         a4 q~   1.8  1999     4 auto~ 4        16    25 p     comp~
## 10 audi         a4 q~   2    2008     4 manu~ 4        20    28 p     comp~
## # ... with 224 more rows
```

Let us use graphs to answer the following question. (i) Do cars with big engines use more fuel than cars with small engines? (ii) What does the relationship between engine size and fuel efficiency look like? Is it positive? Negative? Linear? or Non-linear?

Lets look at the variables: displ, a car's engine size, in litres hwy, a car's fuel efficiency on highway, in miles per gallon (mpg).

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



The plot shows negative relationship between engine size (displ) and fuel efficiency (hwy). In other words, cars with big engines use more fuel. Does this confirm or refute the hypothesis about fuel efficiency and engine size?

ggplot()

This function creates a coordinate system that we can add layers to. The first argument is the dataset.**ggplot2** comes with many geom function that each add a different type of layer to a plot. Each geom function takes a mapping argument. This defines how variables in the dataset are mapped to visual properties. The mapping argument is paired with aes(), and the *x* and *y* arguments of *aes()* specify the variables to map to the x and y axes.

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

EXERCISES

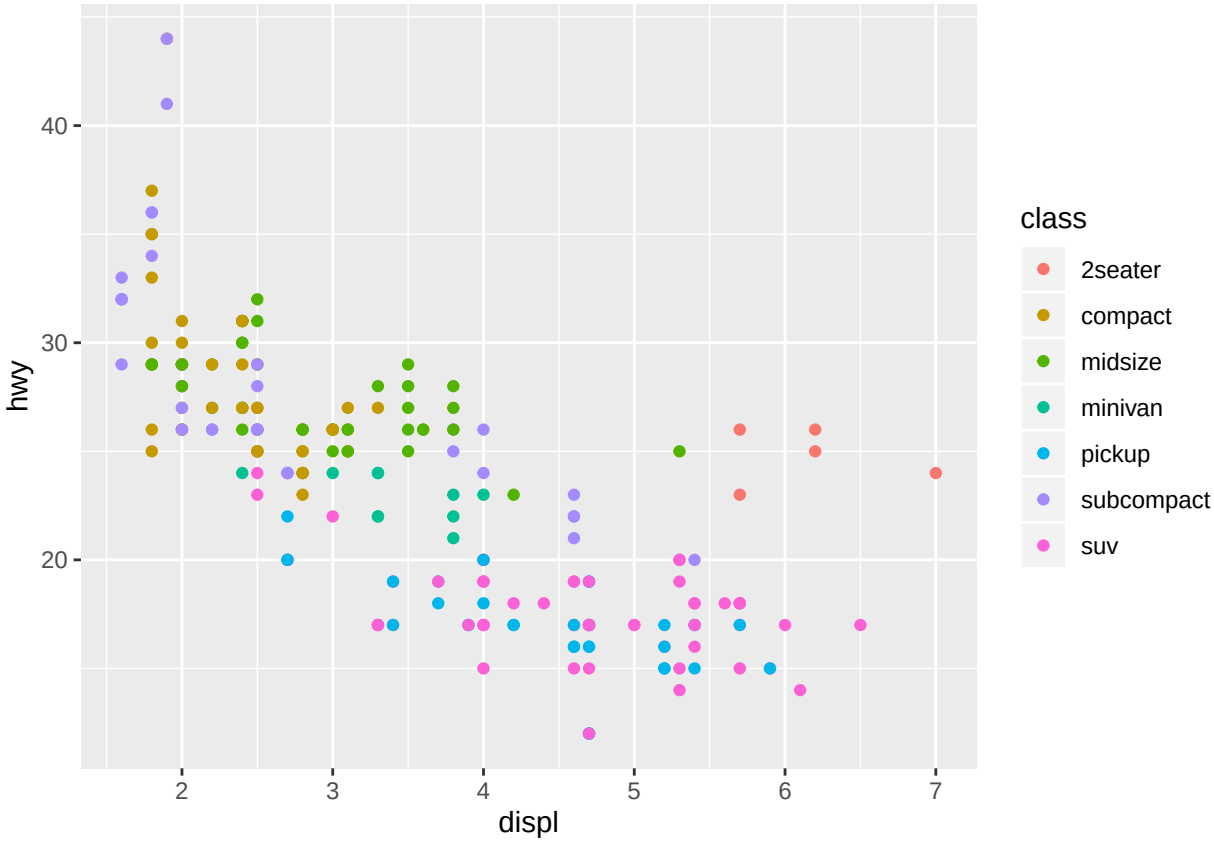
1. Run ggplot(data = mpg). what do you see?
2. What does the *drv* variable describe?
3. Make a scatterplot of hwy versus cyl.
4. What happens if you make a scatterplot of class vs. drv? Is it useful?

Lets hypothesize that the cars are hybrid.

Test the hypothesis: (1) Look at the class value for each car.The class variable of the *mpg* dataset classifies cars into groups such as compact, midsize, and SUV. If the outlying points are hybrids, they should be classified as compact cars or, subcompact cars.(Data was collected before hybrid trucks and SUVs became popular)

Lets add a third variable, class, to the two-dimensional scatterplot.

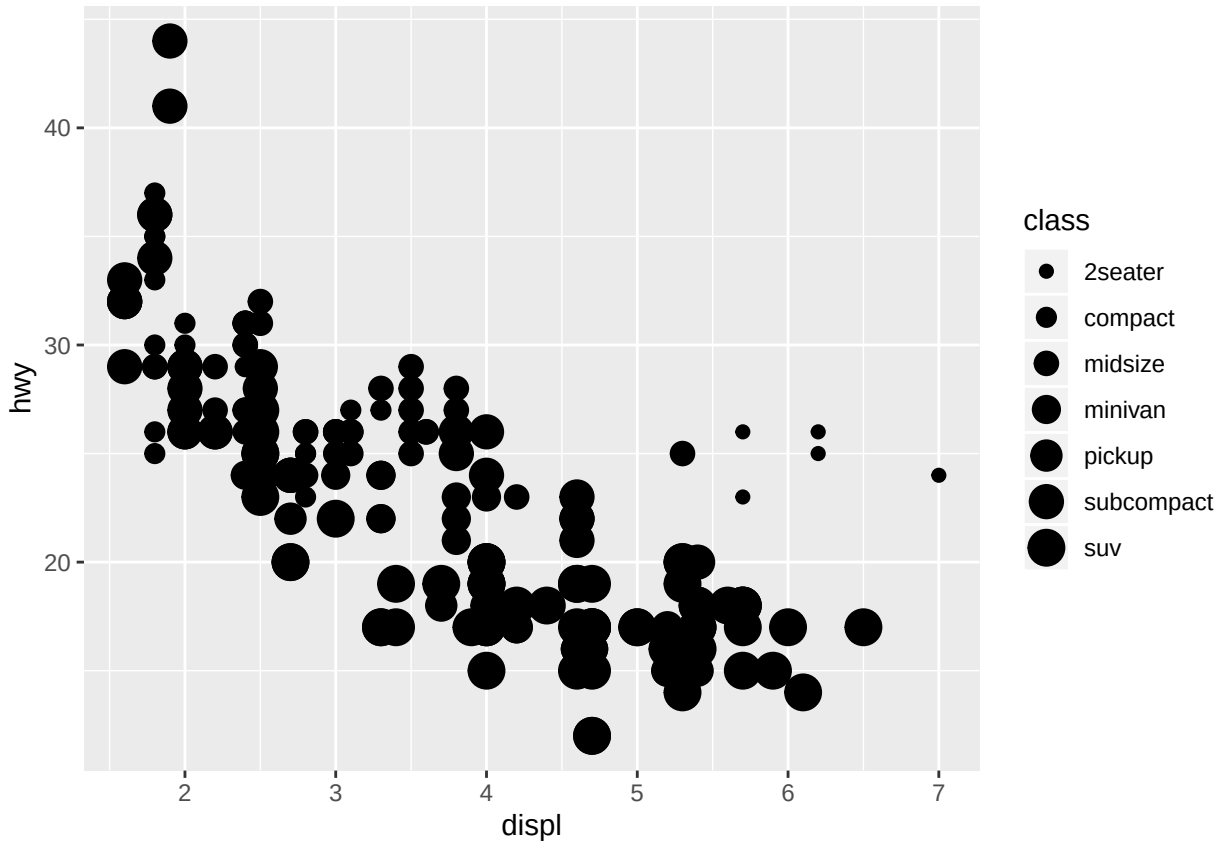
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x=displ, y = hwy, color = class))
```



The colors reveal that many of the unusual points are two-seater cars. These cars don't seem like hybrids, and are, in fact sports cars. Sports cars have large engines like SUVs and pickup trucks, but small bodies like midsize and compact cars, which improve their gas mileage.

What would happen if we used size instead of color?

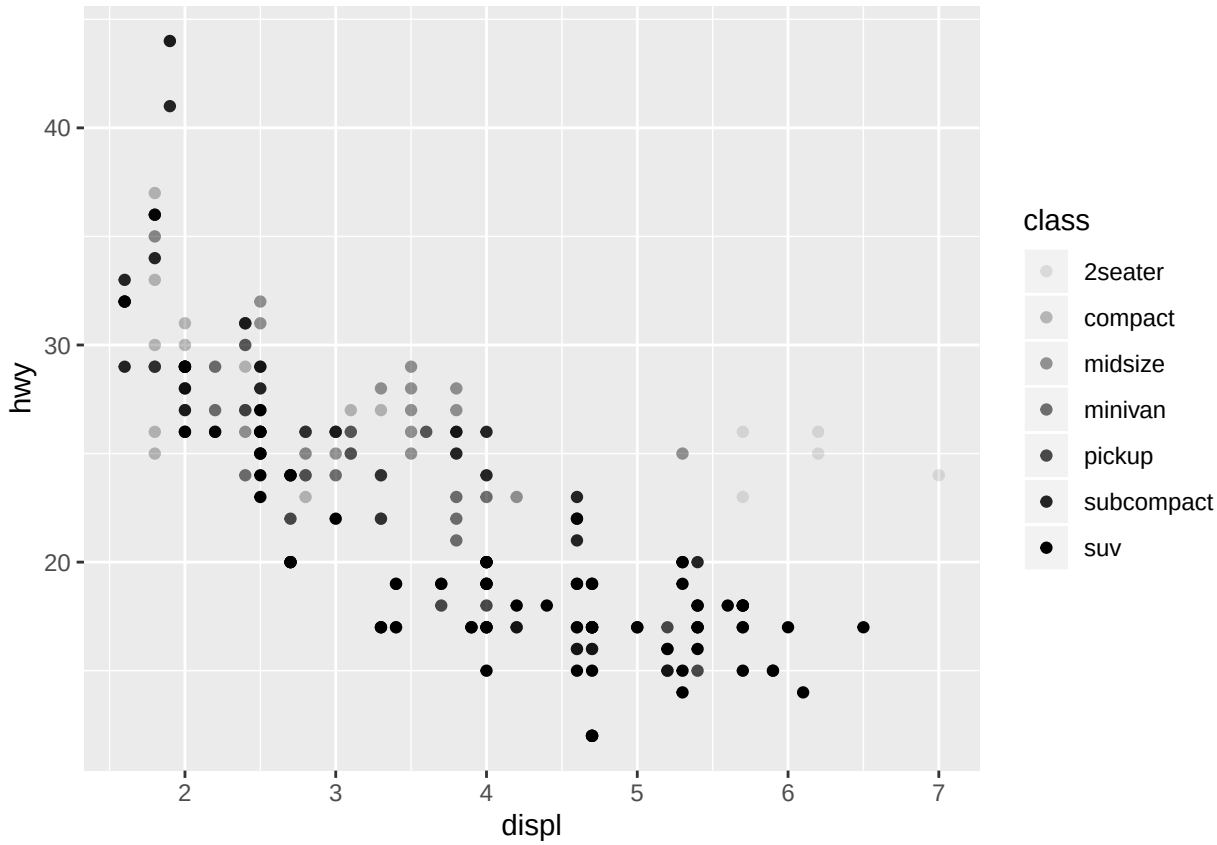
```
## Warning: Using size for a discrete variable is not advised.
```



We could also use alpha or shape instead.

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

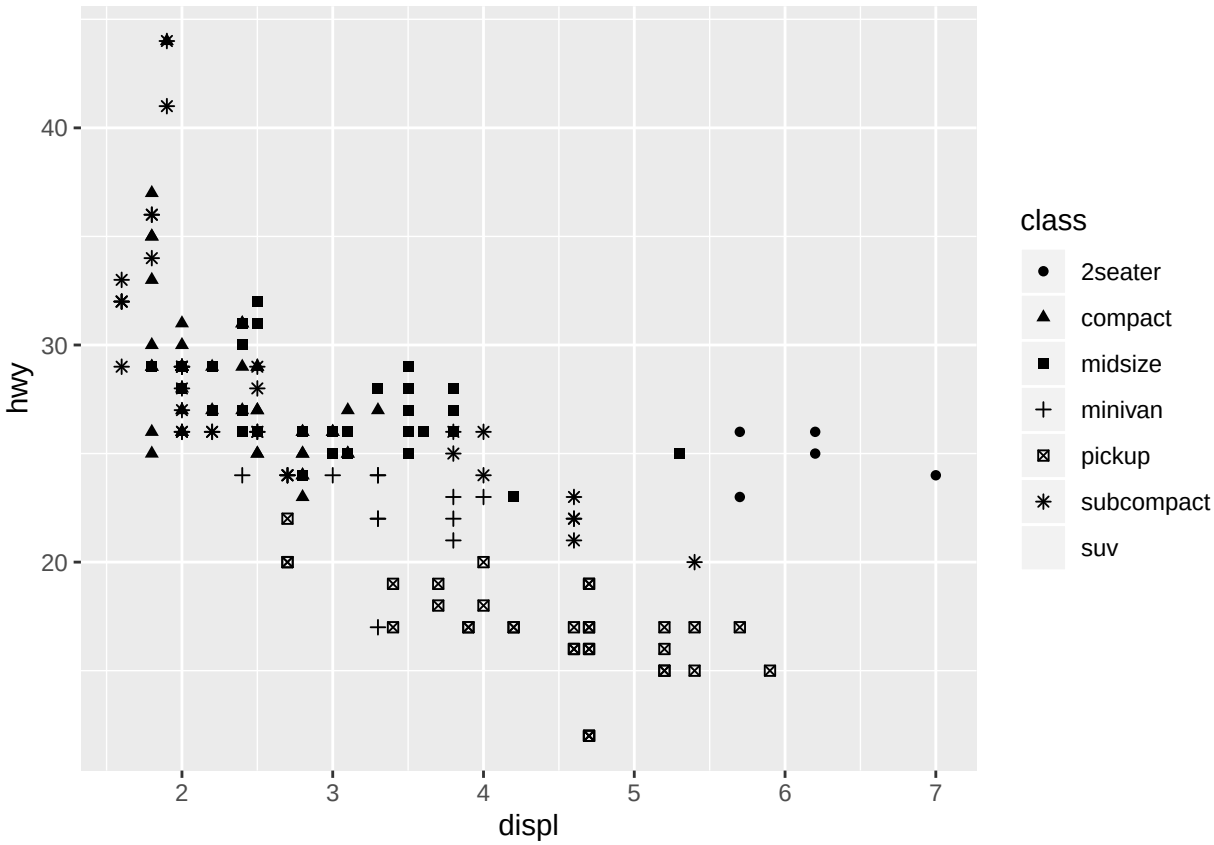
```
## Warning: Using alpha for a discrete variable is not advised.
```




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

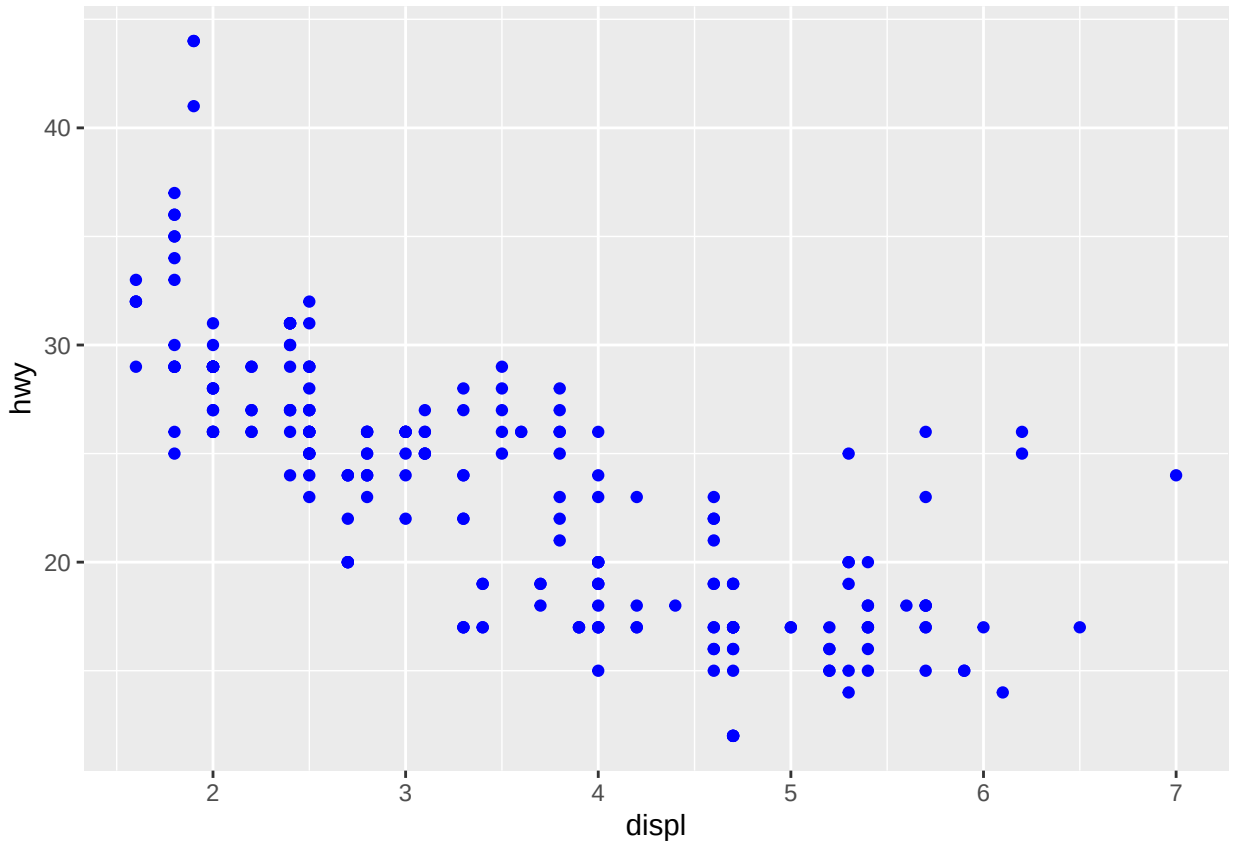
```
## Warning: The shape palette can deal with a maximum of 6 discrete values
## because more than 6 becomes difficult to discriminate; you have 7.
## Consider specifying shapes manually if you must have them.
```

```
## Warning: Removed 62 rows containing missing values (geom_point).
```



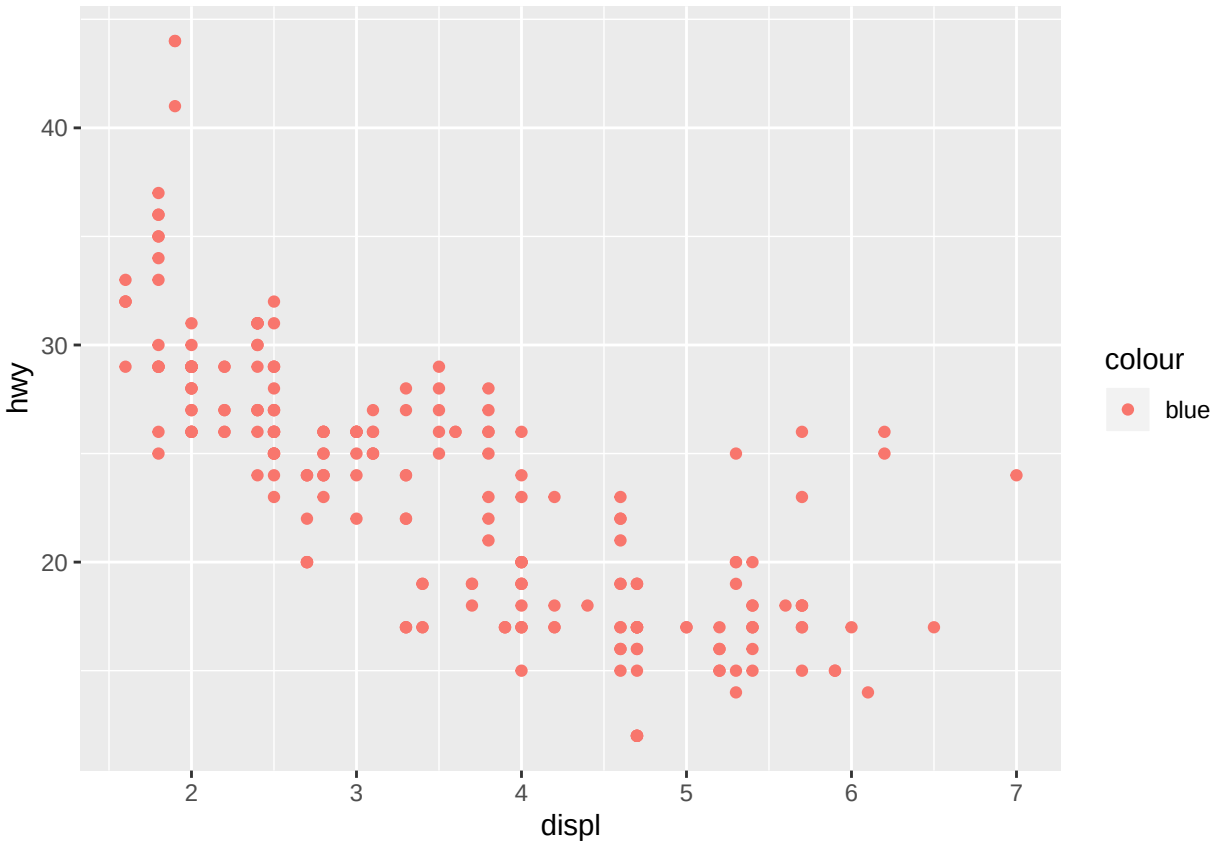
We can also set the aesthetics properties of the geom manually. For example, if we want to make all the points blue –

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

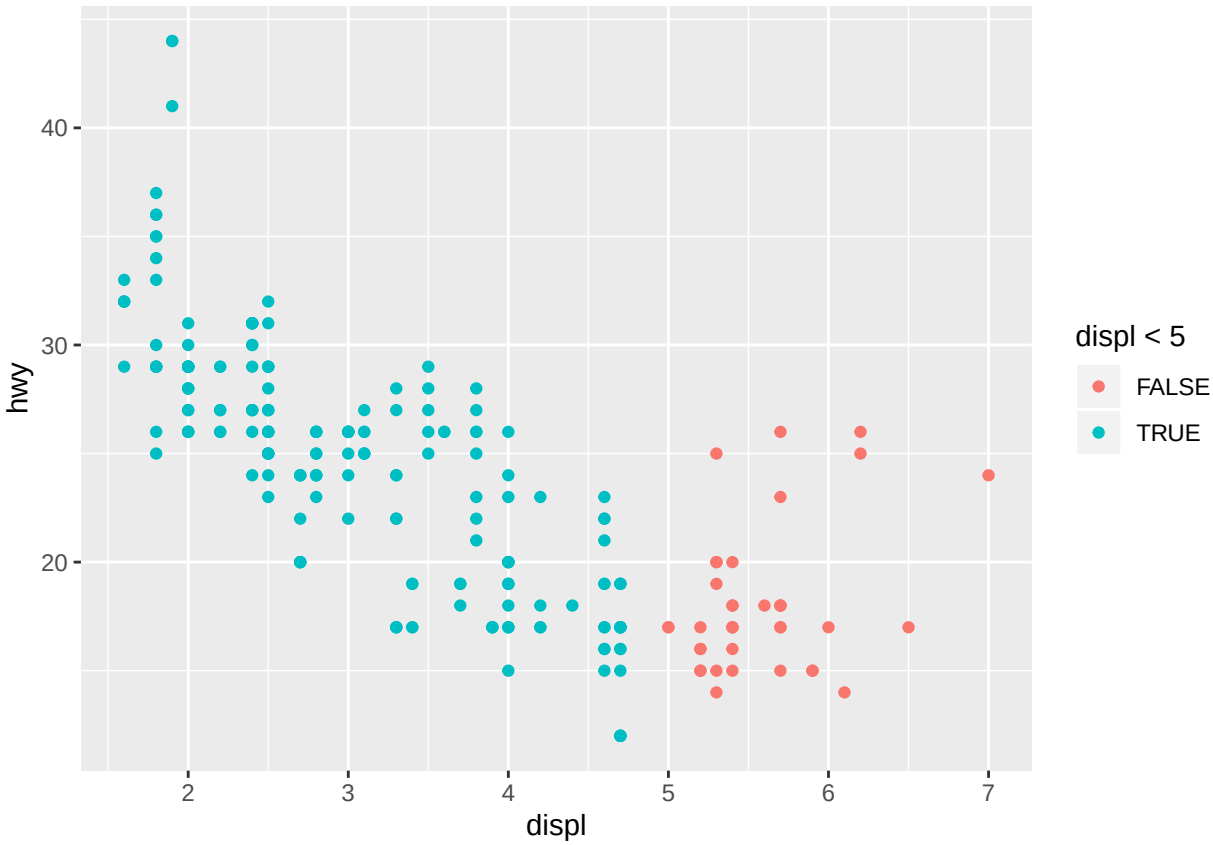


What's wrong with this code? Why are points not blue?

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



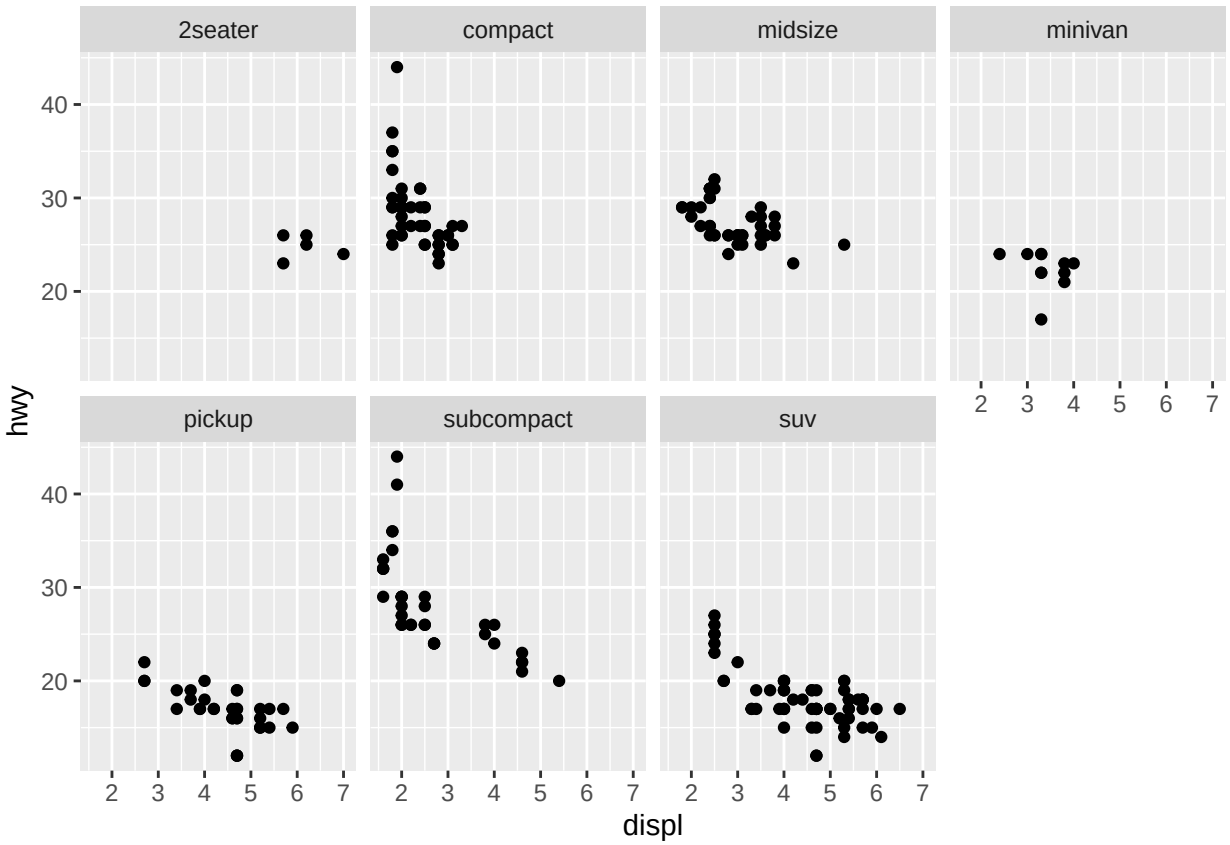
EXERCISES (1) Which variables are categorical? Which variable are continuous? (2) Map a continuous variable to color,size, and shape. How do these aesthetics behave differently for categorical versus continuous variables? (3) What happens if you map the same variable to multiple aesthetics? (4)What happens if you map an aesthetic to something other than a varable name (color = displ < 5)?



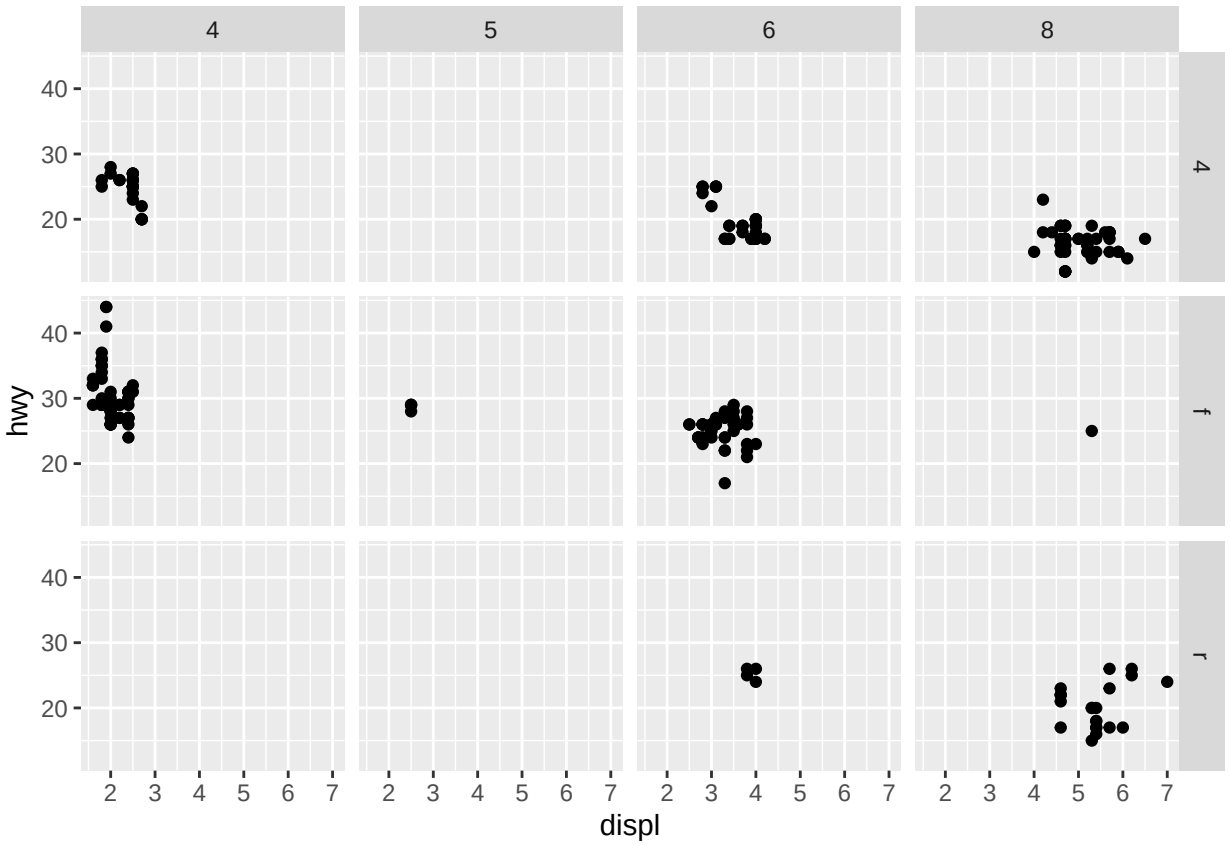
Facets

One way to add additional variables is with aesthetics. Another way, is to split the plot into facets, subplots that display one subset of the data.

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

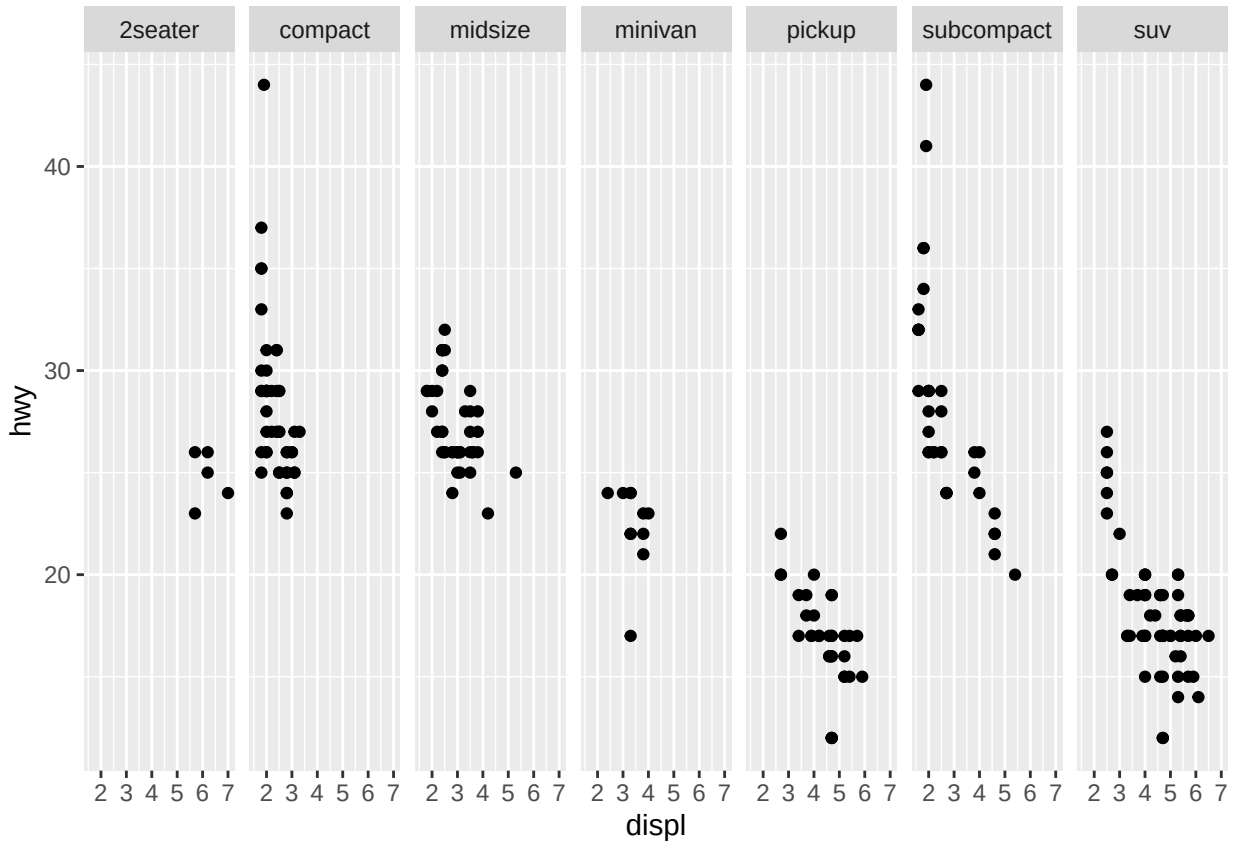



```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_grid( drv ~ cyl) # facet using two variables
```

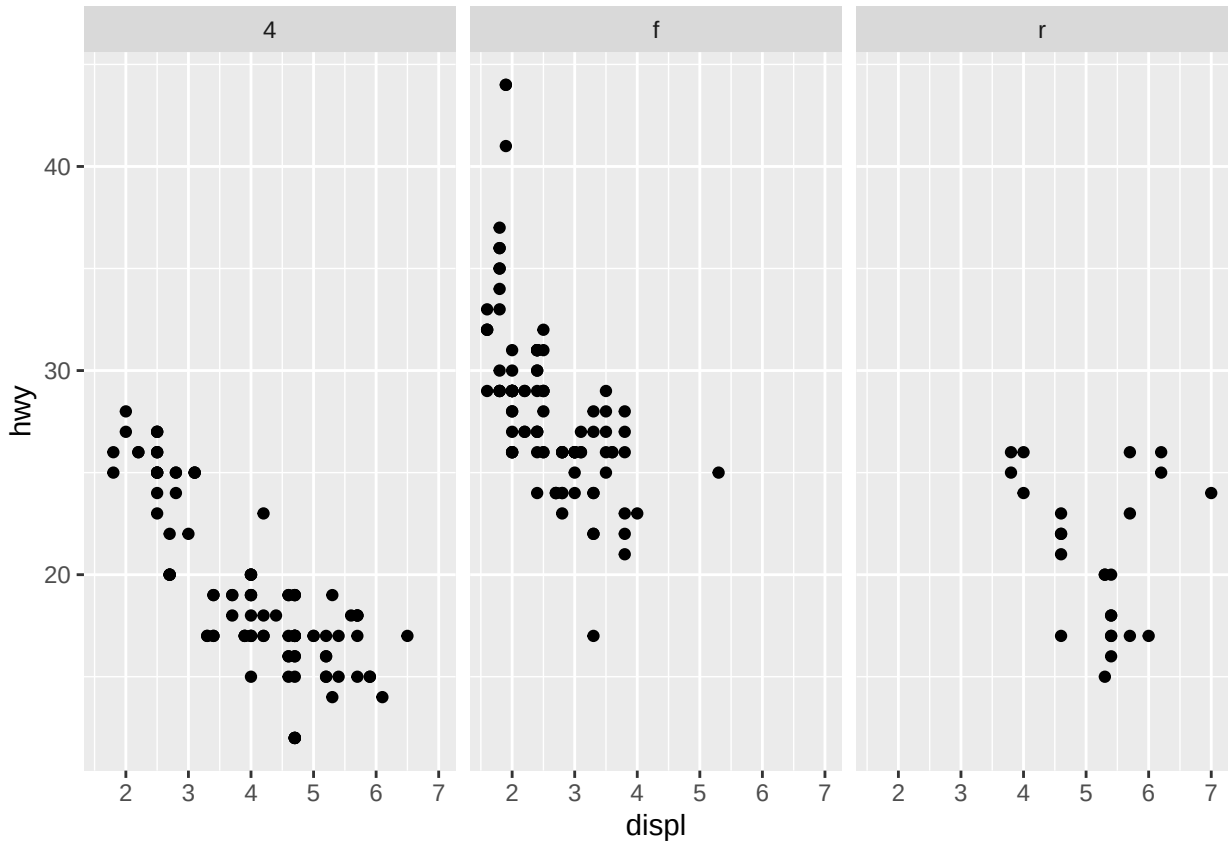


EXERCISES (1) What happens if you facet on a continuous variable? (2) What plots does the following code make?

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

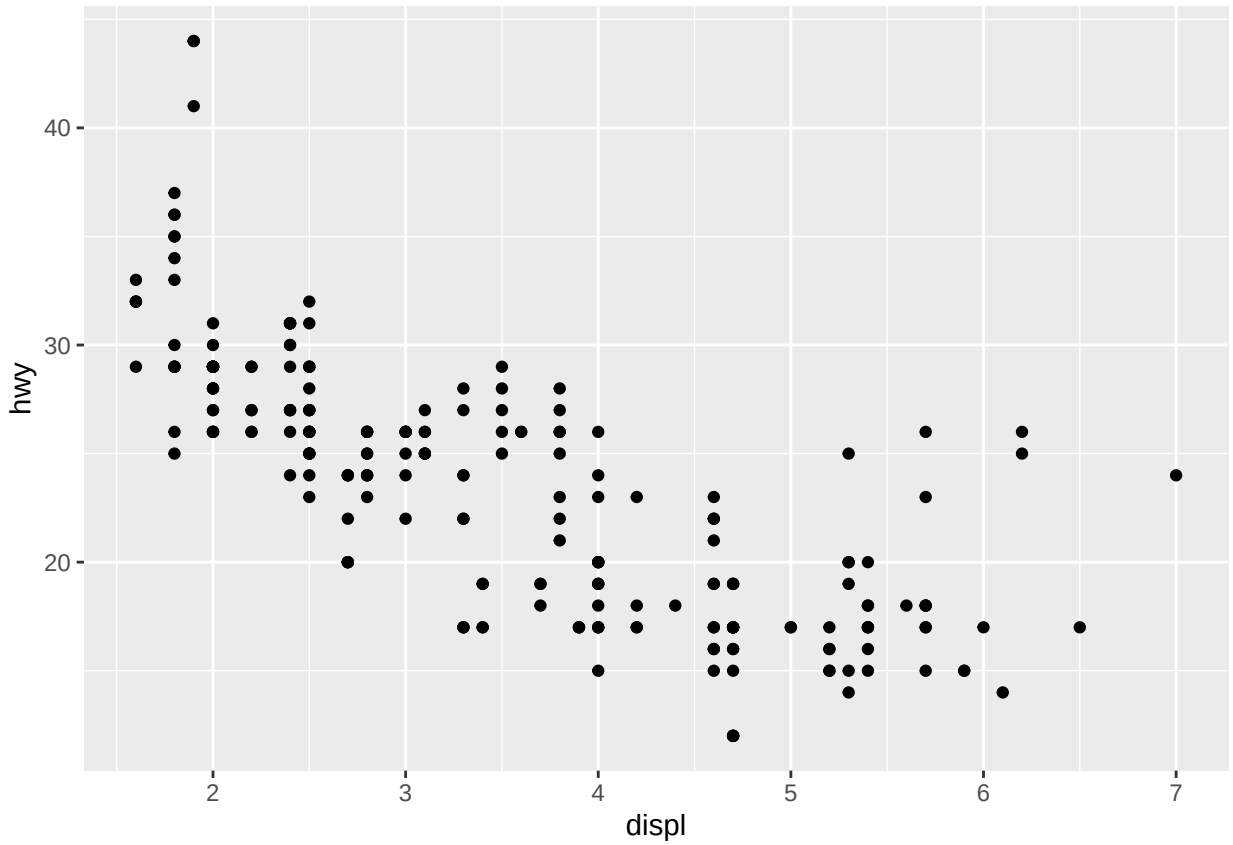



```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_wrap( drv ~ .)
```



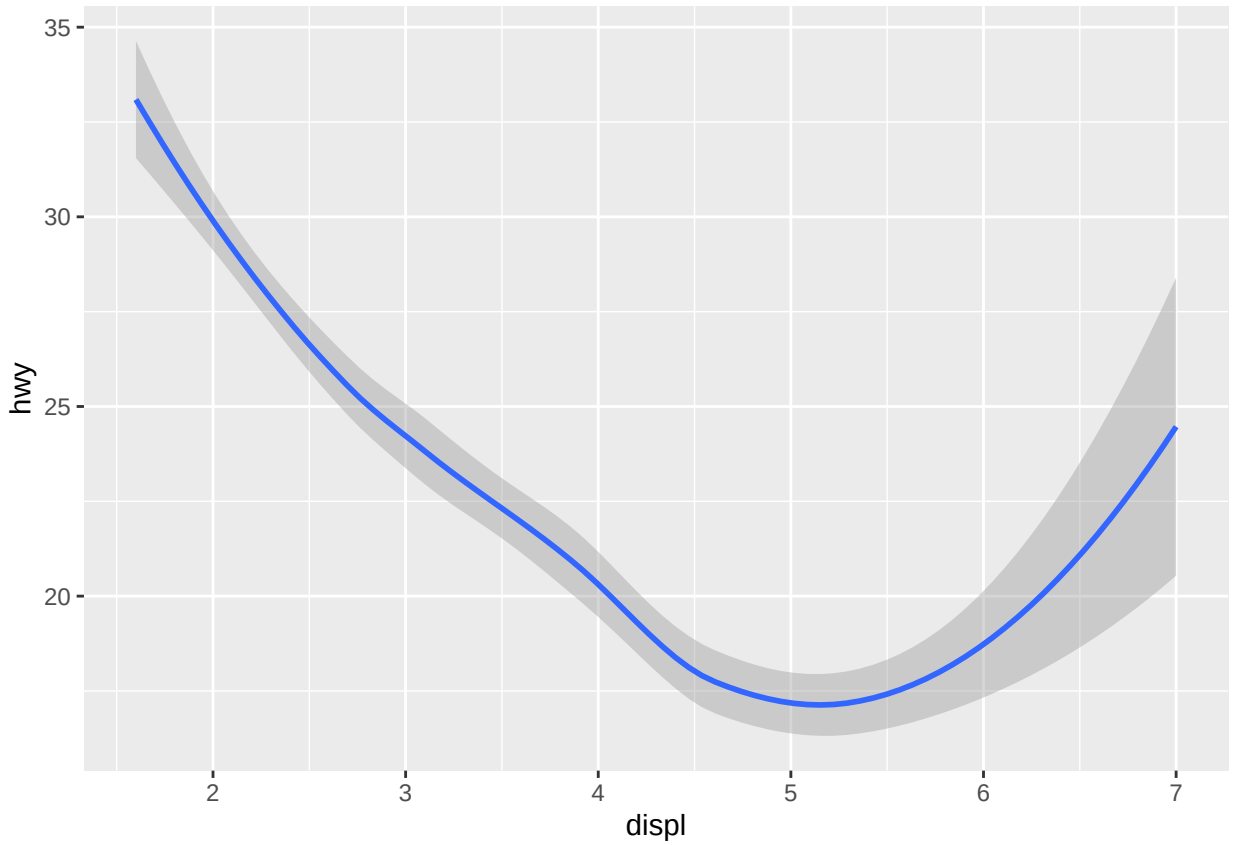
Geometric Objects

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



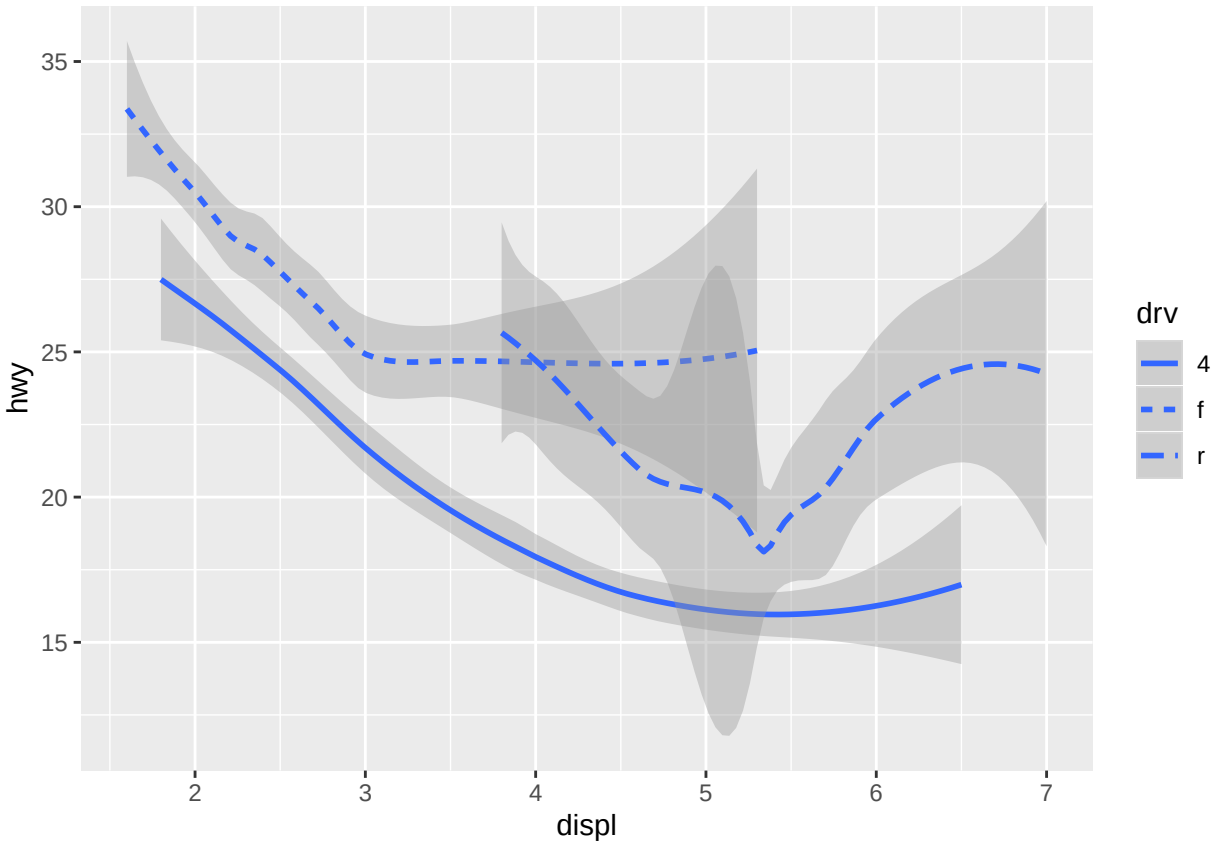

```
ggplot(data = mpg)+
  geom_smooth(mapping = aes(x = displ, y = hwy))
```

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




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

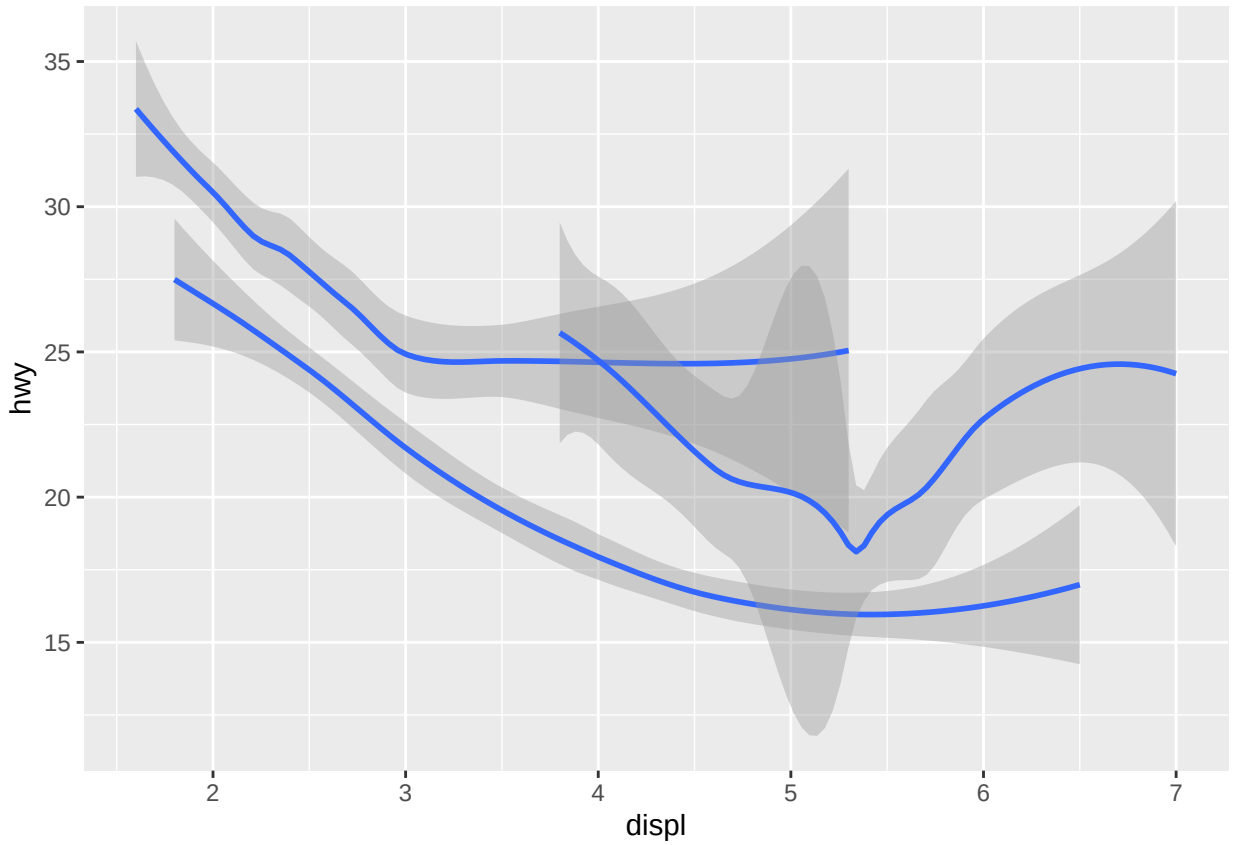
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



The above plot separates the cars into three lines based on their drv value, which describes a car's drivetrain. One line describes all of the points with a 4 value, one line describes all of the points with an f value, and one line describes all of the points with an r value. Here, 4 stands for four-wheel drive, f for front wheel drive, and r for rear wheel drive.

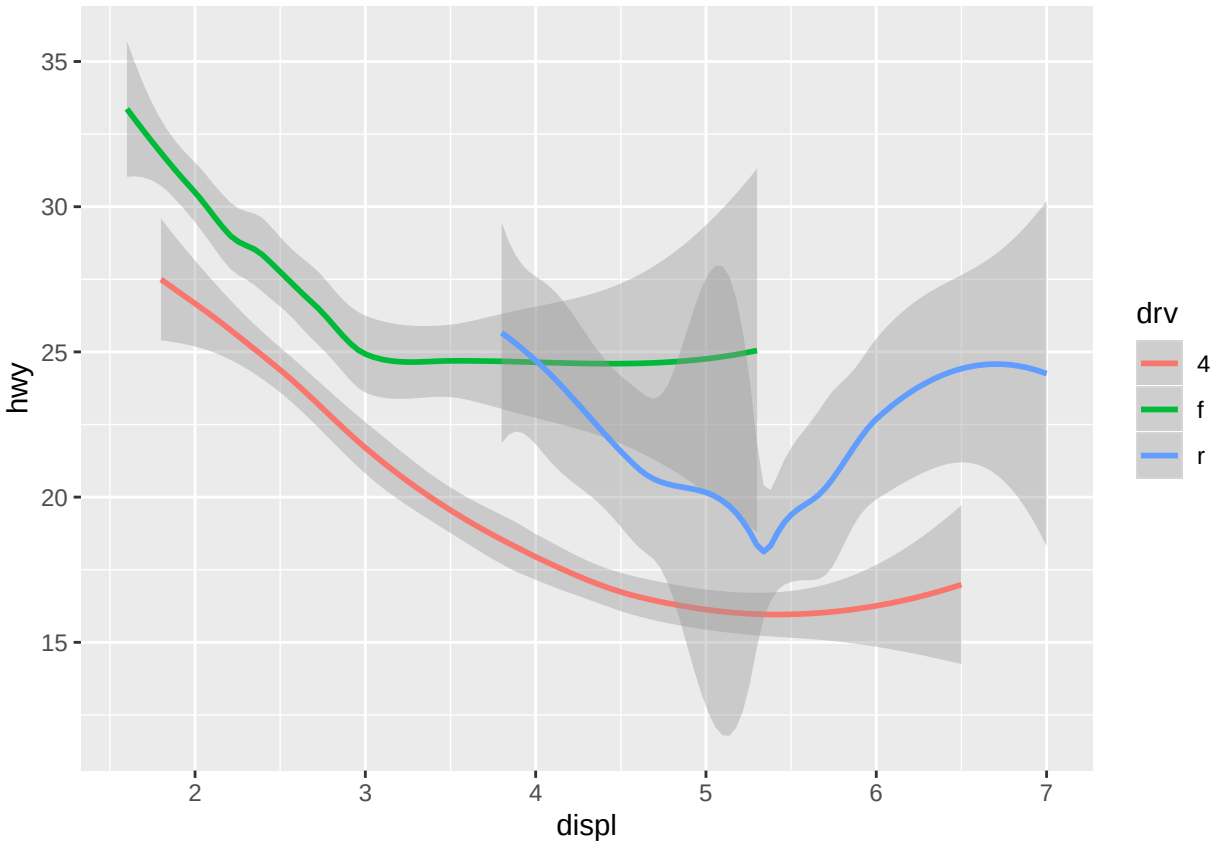
```
ggplot(data = mpg)+
  geom_smooth(mapping = aes(x = displ, y = hwy, group = drv))
```

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




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

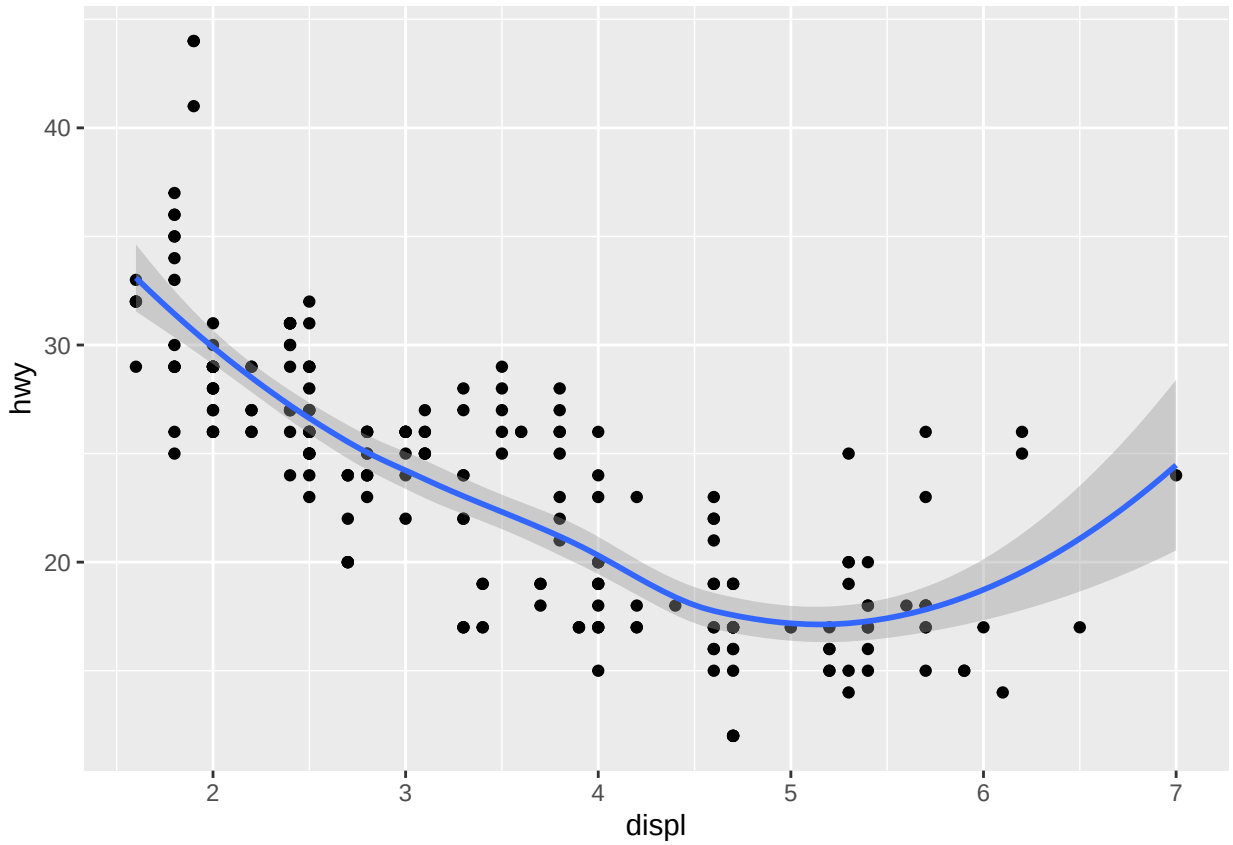
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Display multiple geoms

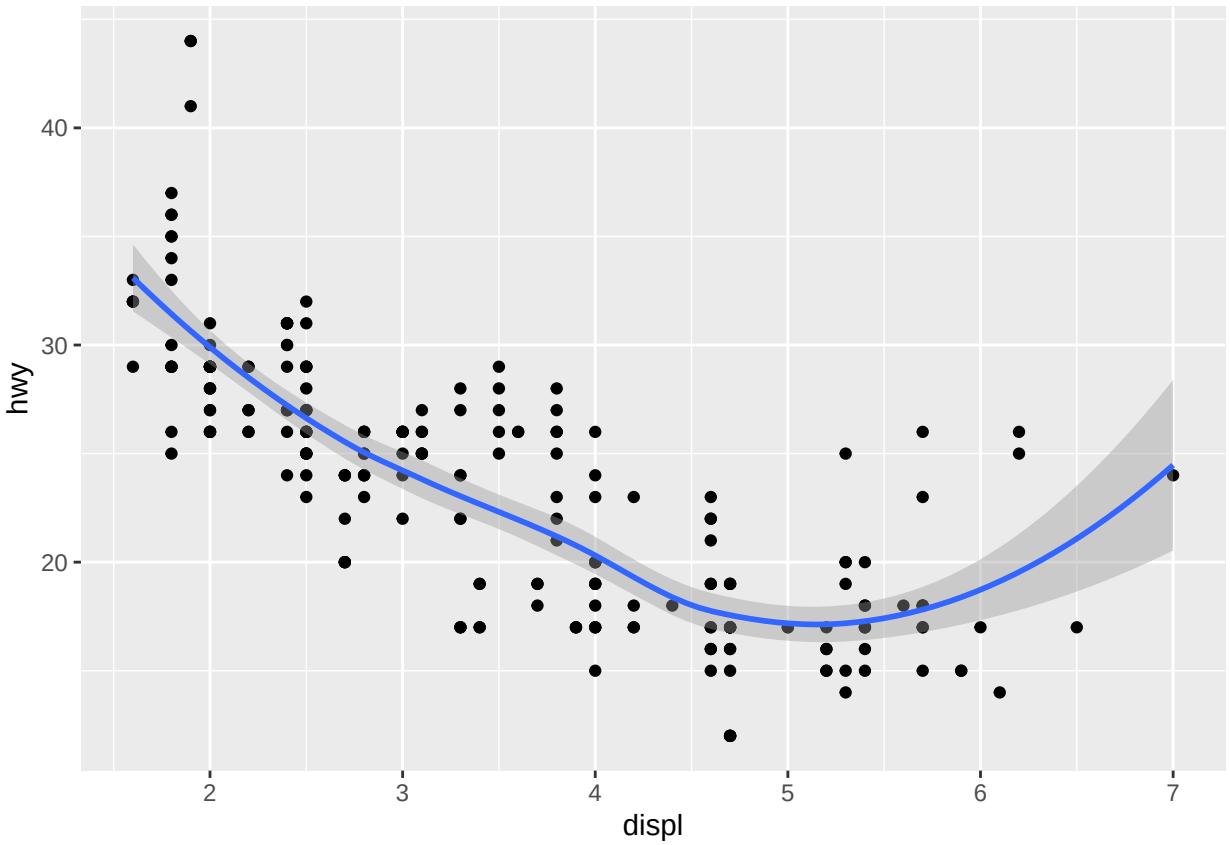
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  geom_smooth(mapping = aes(x = displ, y = hwy))
```

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




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

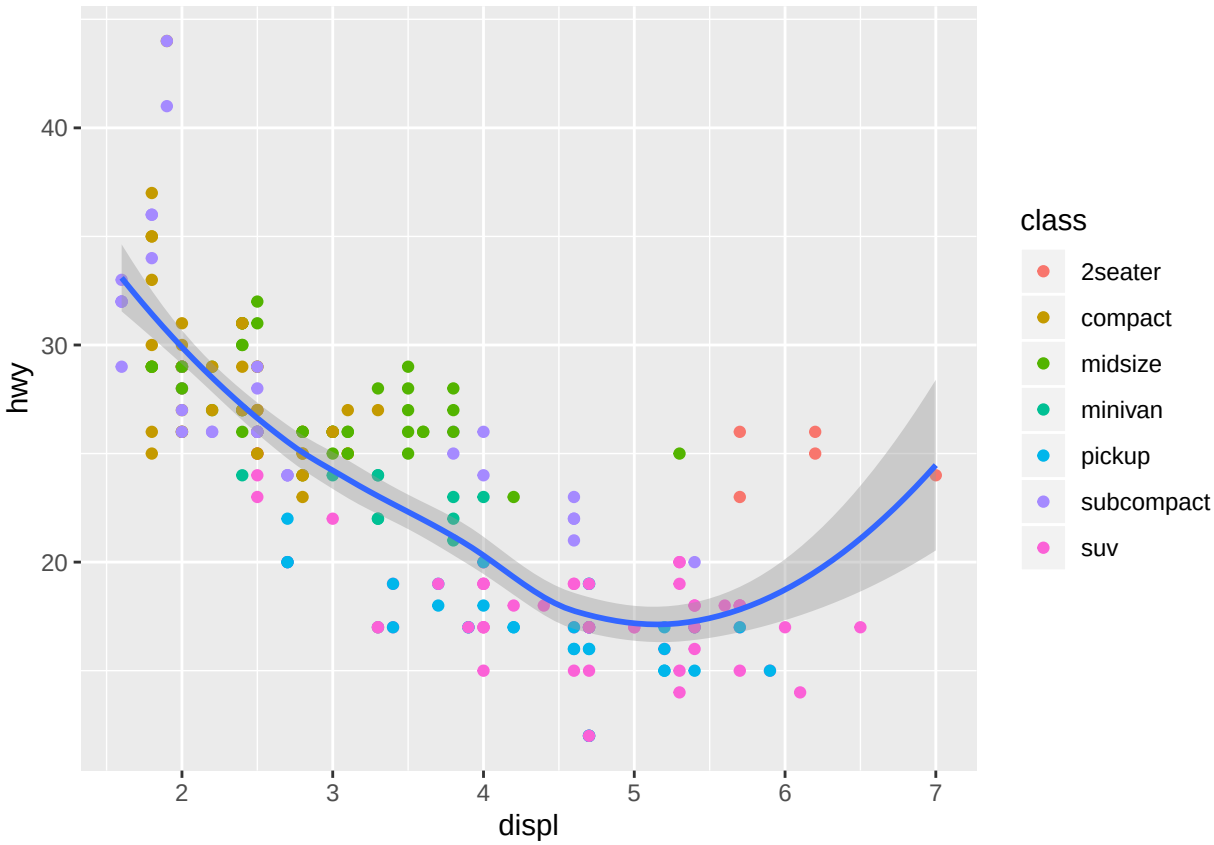
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



If mappings are placed in a geom function, they are treated as local mappings for the layer. It will use these mappings to extend or over write the global mappings for that layer only.This makes it possible to display different aesthetics in different layers.

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

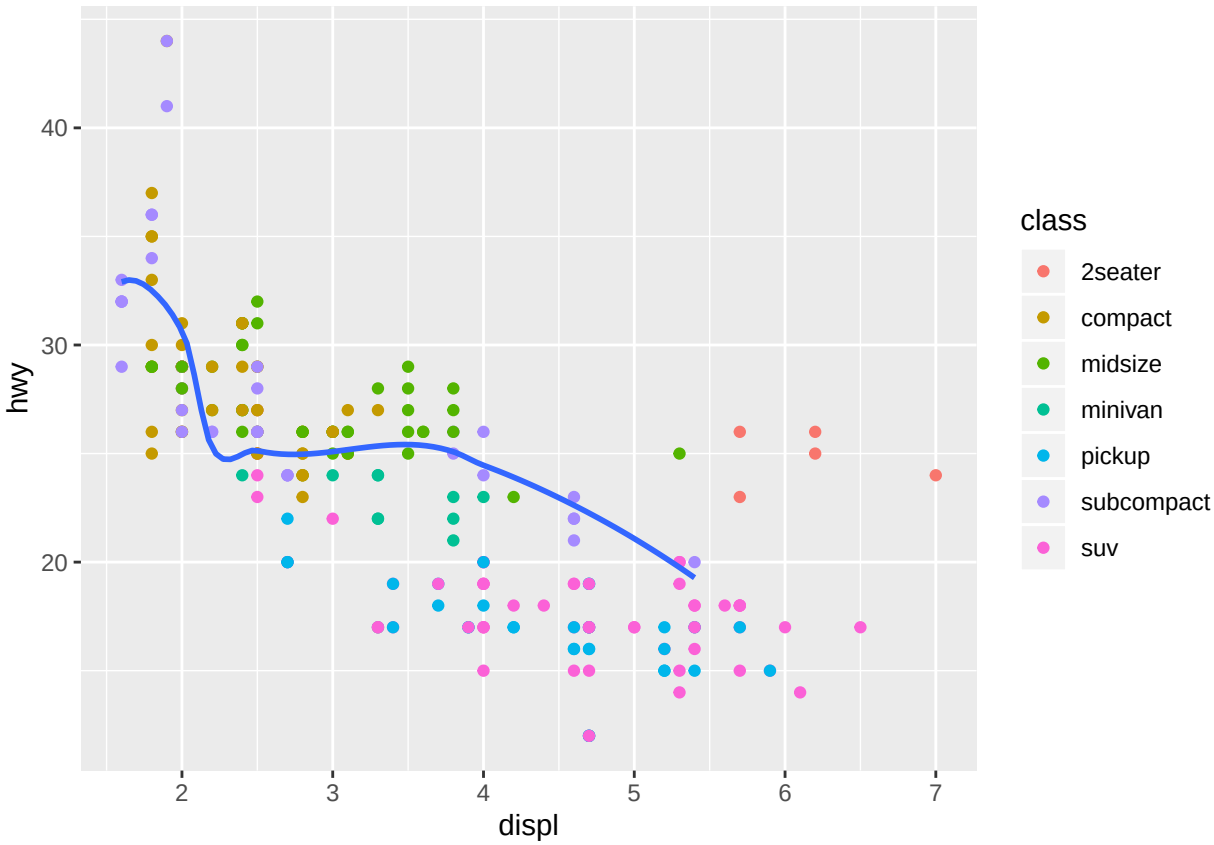
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



We can also specify different data for each layer.

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

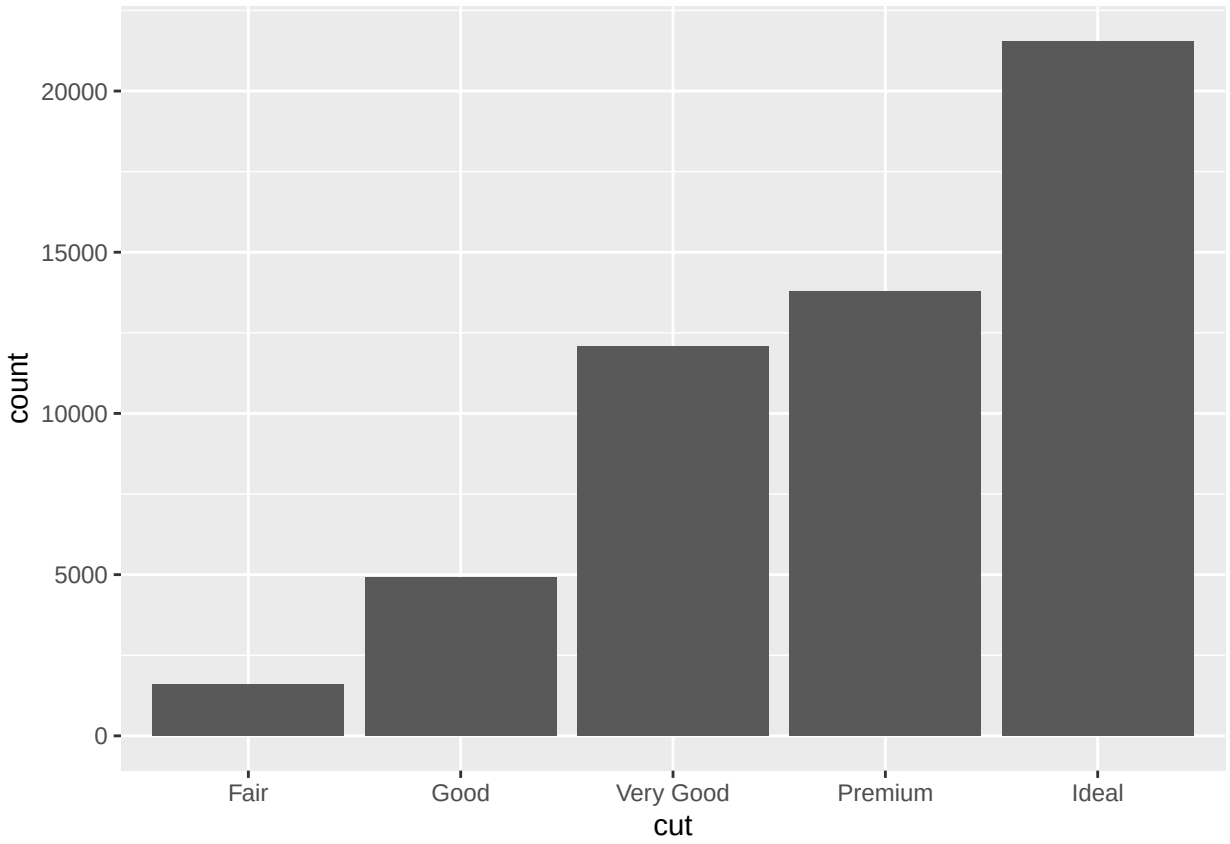
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Statistical Transformations

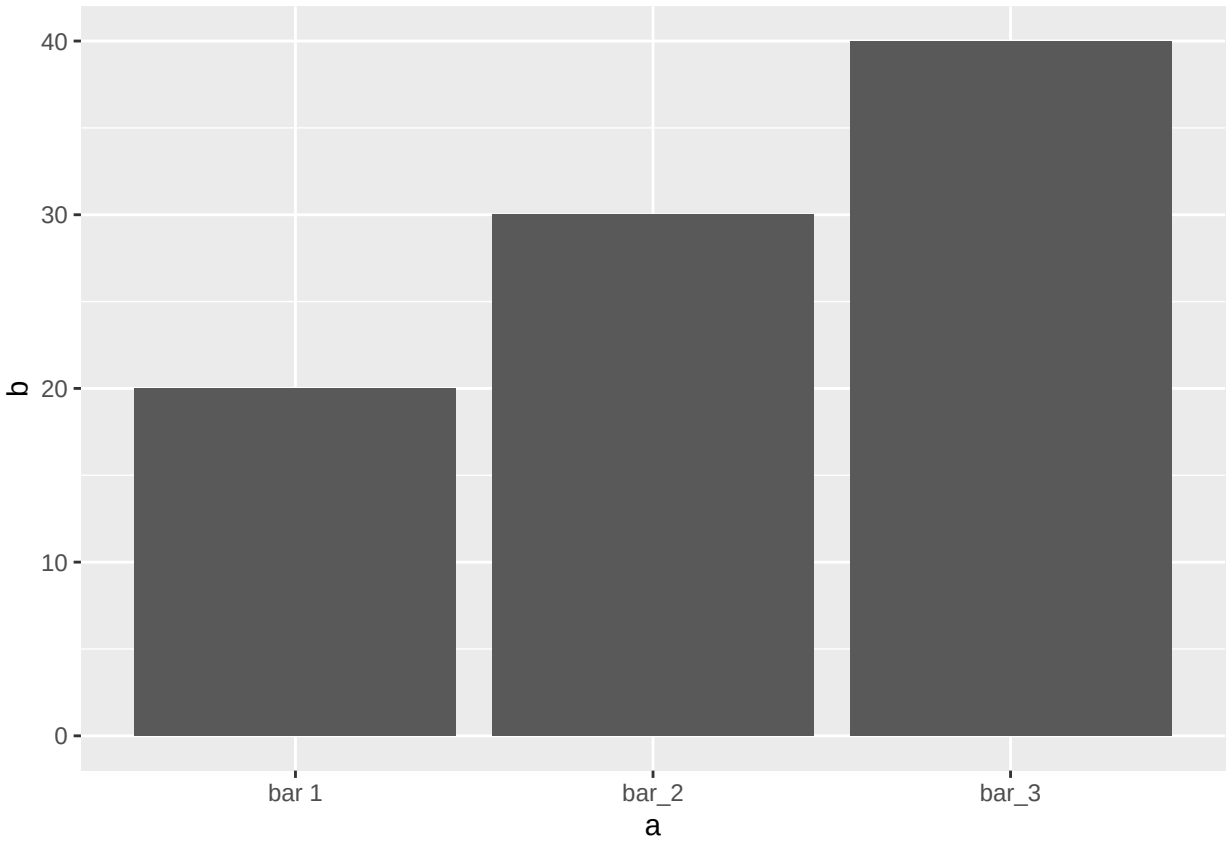
The diamonds dataset contain information about approximately 54,000 diamonds, including price, carat, color, clarity, and cut of each diamond.

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



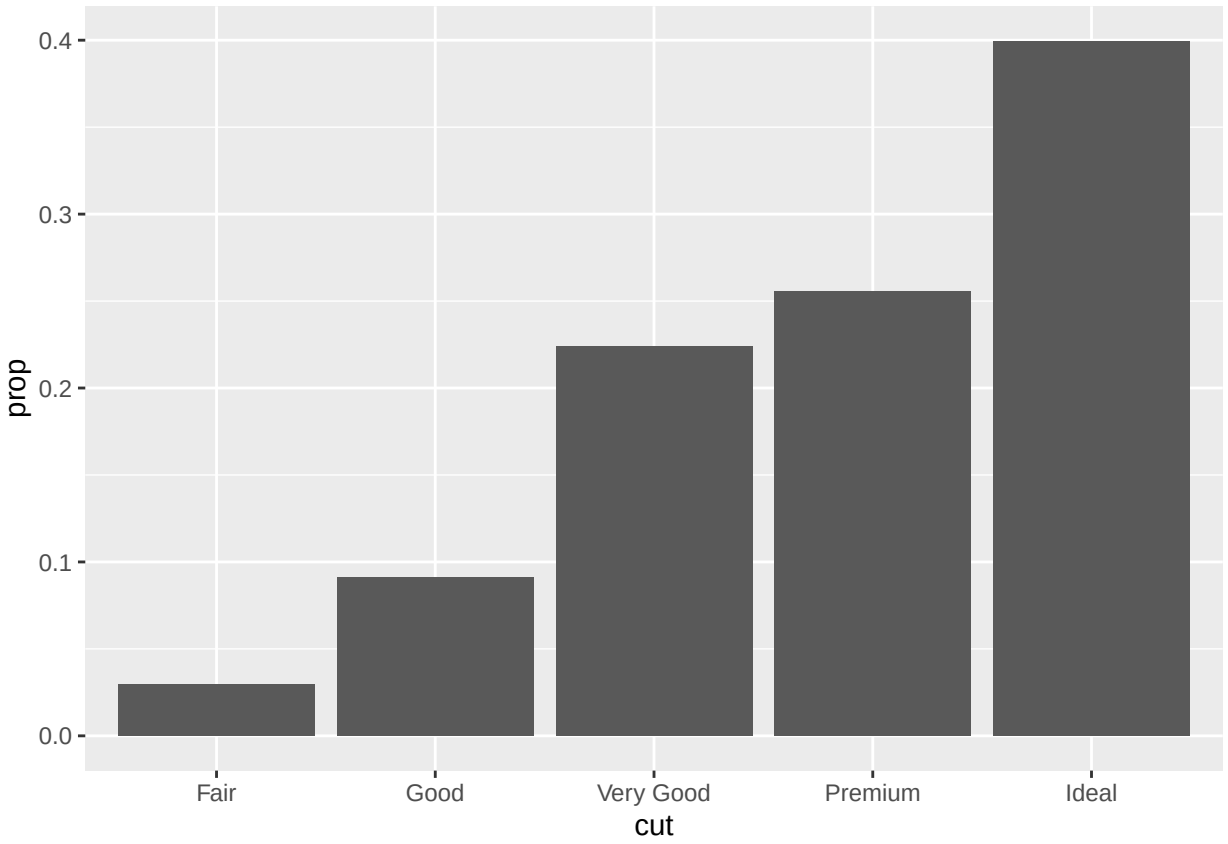
To check which stat a geom uses, see the default value for stat argument. The dfeault stat is "count" for geom_bar(). To override the default stat by specifying the stat explicitly.

```
demo = tribble(
  ~a,      ~b,
  "bar 1", 20,
  "bar_2", 30,
  "bar_3", 40
)
ggplot(data = demo)+
  geom_bar(
    mapping = aes(x = a, y = b), stat = "identity"
  )
```



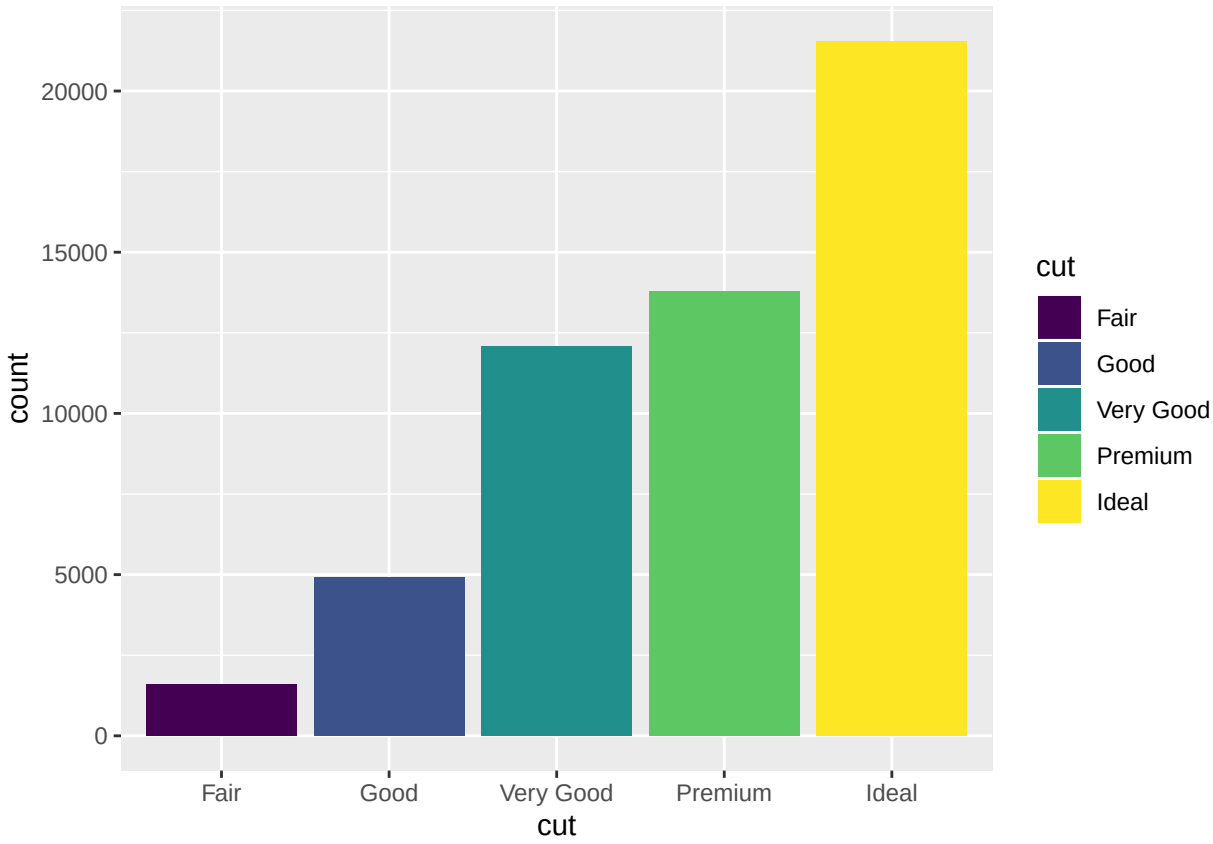
We can display a bar chart of proportion, rather than count.

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

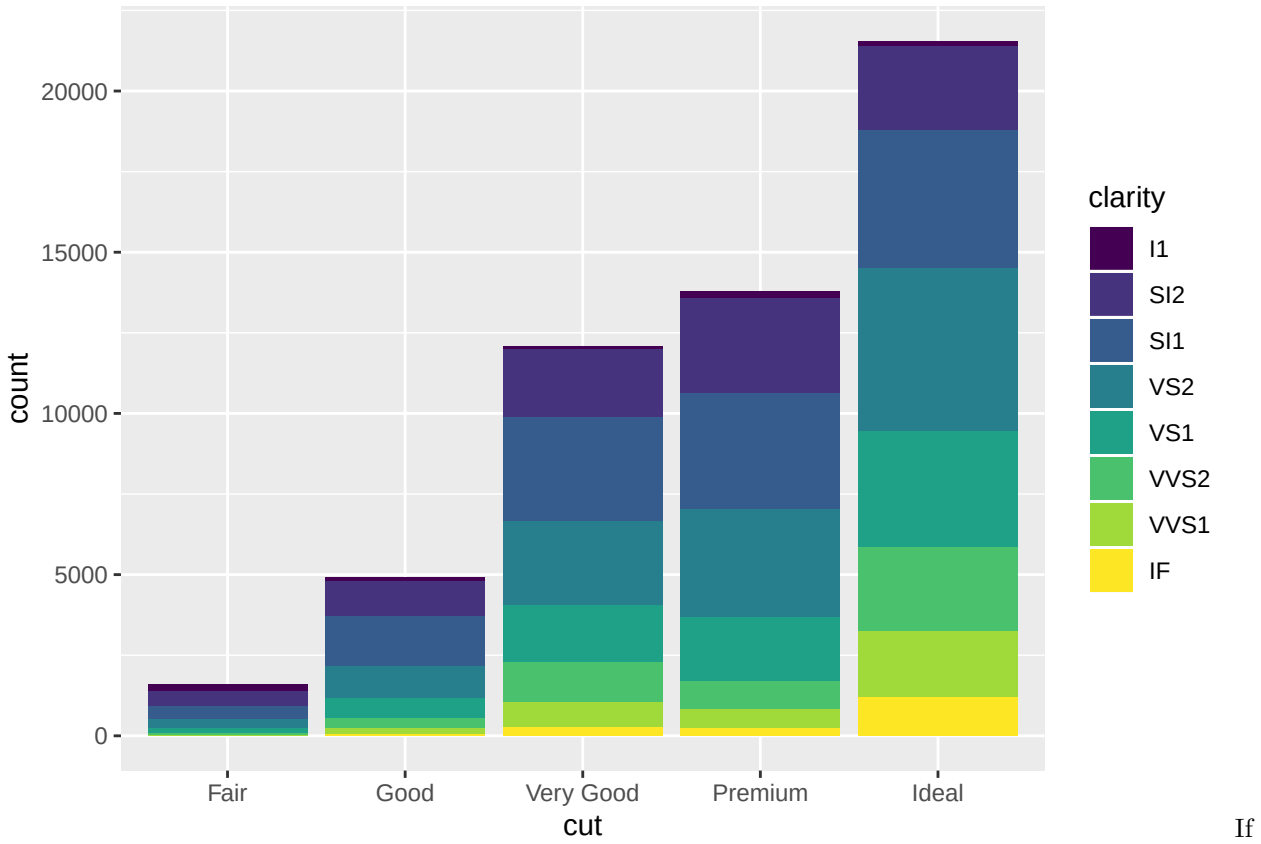


Position Adjustments

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



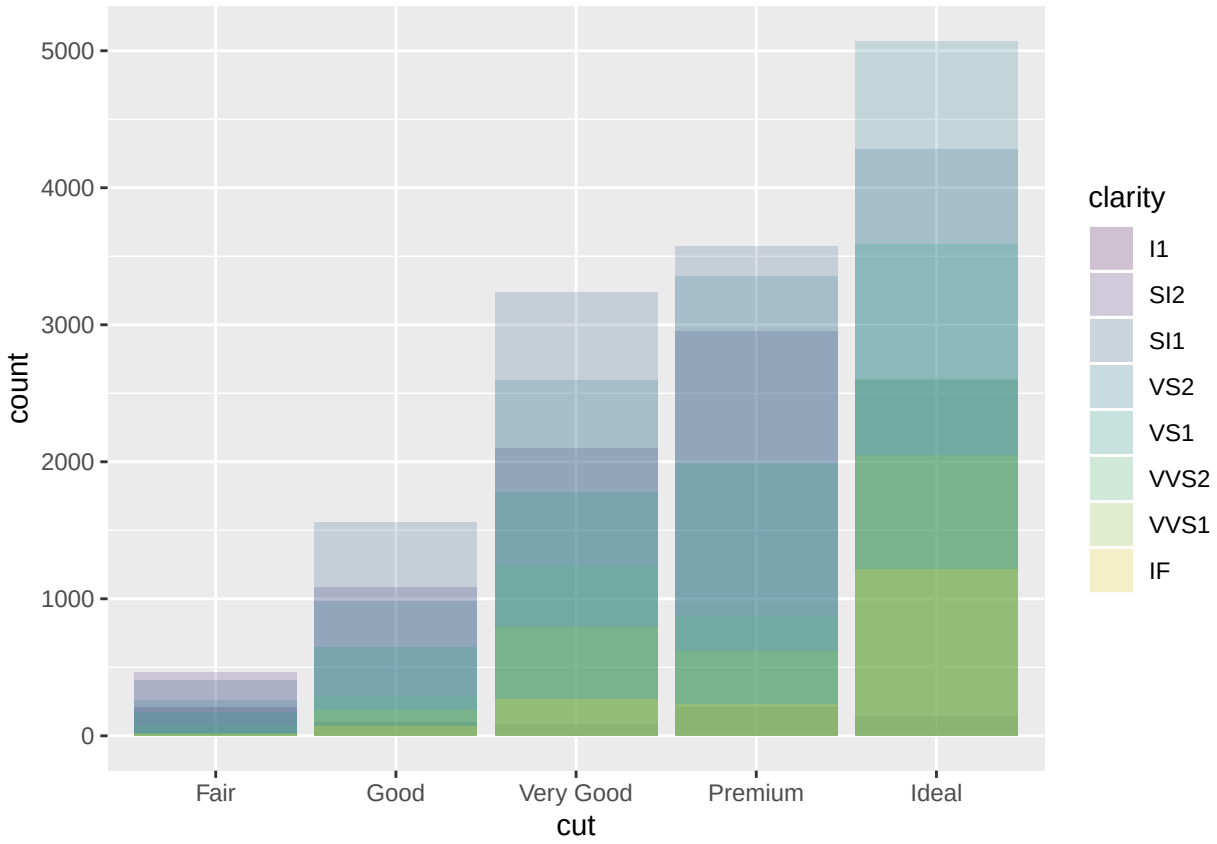

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = clarity)) # using another variable
```



If we use fill aesthetic to another variable, the bars are automatically stacked. Stacking can also be performed automatically by position adjustment specified by the position argument.

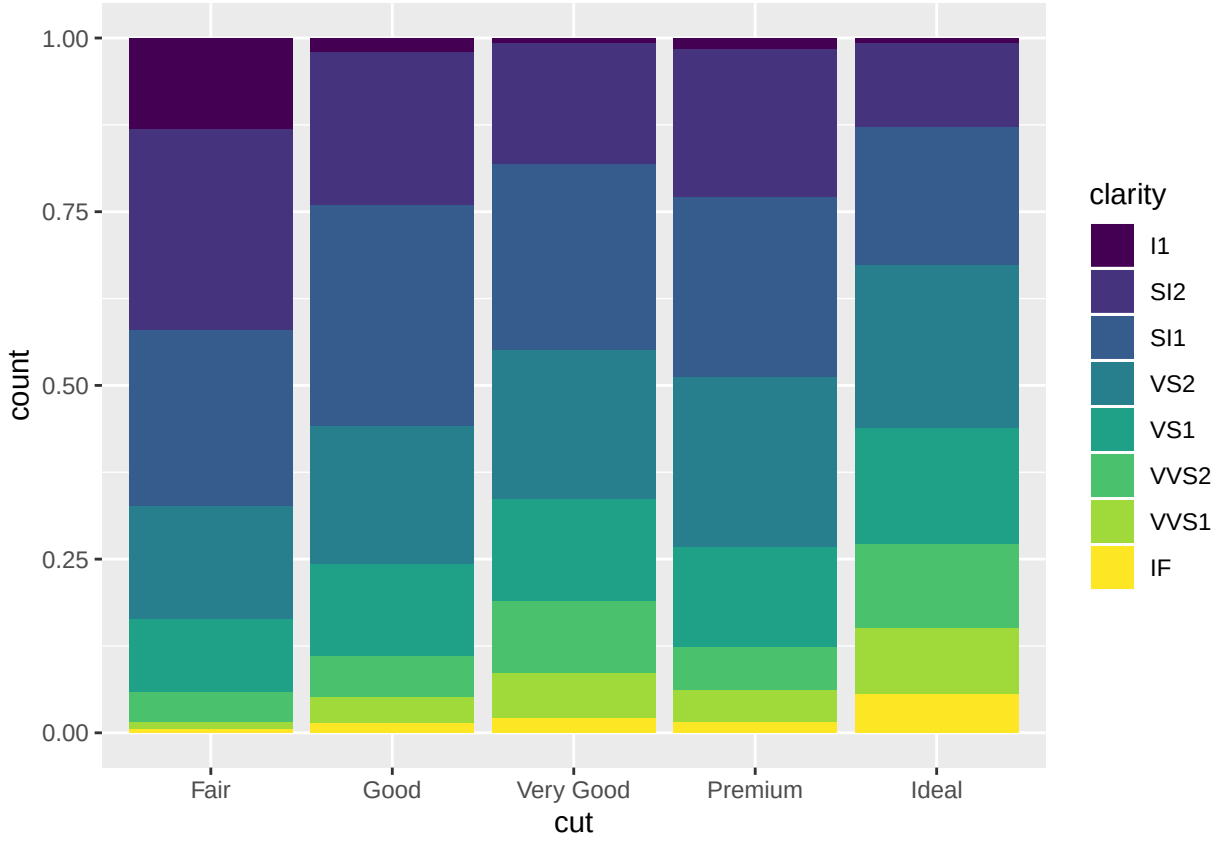
(i) position = "identity" places each object exactly where it falls within the context of the graph

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



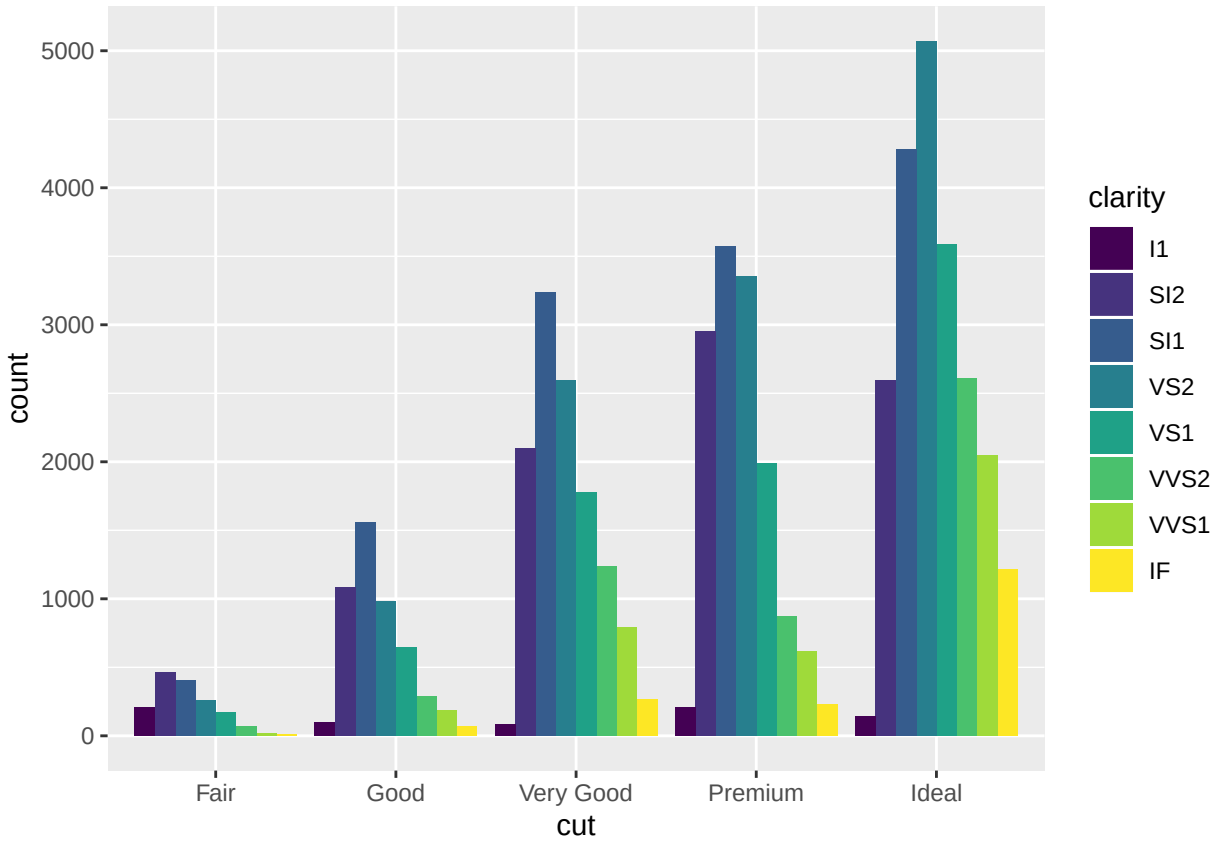
This is not very useful for bars because they overlap. (ii) position = "fill". Works like stacking, but m akes each set of stack bar the same height. Helps to compare proportions across groups.

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

  )
```



(iii) position = "dodge"

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

  )
```


5000
4000
3000
2000
1000
0
count
Fair
Good
Very Good
Premium
Ideal
cut
clarity
I1
SI2
SI1
VS2
VS1
VVS2
VVS1
IF