ggplot

Byteflow Dynamics

10/15/2017

ggplot review

Explore the data set *mpg*. This data set contains observations collected by the US Environment Protection Agency on 38 models of car.

```r
head(mpg)
```

```
## # A tibble: 6 x 11
##   manufacturer model displ  year   cyl      trans   drv   cty   hwy    fl
##          <chr> <chr> <dbl> <int> <int>      <chr> <chr> <int> <int> <chr>
## 1         audi    a4   1.8  1999     4   auto(l5)     f    18    29     p
## 2         audi    a4   1.8  1999     4 manual(m5)     f    21    29     p
## 3         audi    a4   2.0  2008     4 manual(m6)     f    20    31     p
## 4         audi    a4   2.0  2008     4   auto(av)     f    21    30     p
## 5         audi    a4   2.8  1999     6   auto(l5)     f    16    26     p
## 6         audi    a4   2.8  1999     6 manual(m5)     f    18    26     p
## # ... with 1 more variables: class <chr>
```

Scatter plots: geom_point( )

Use a scatter plot to answer the following question: Do cars with big engines use more fuel than cars with small engines?

What's your prediction?

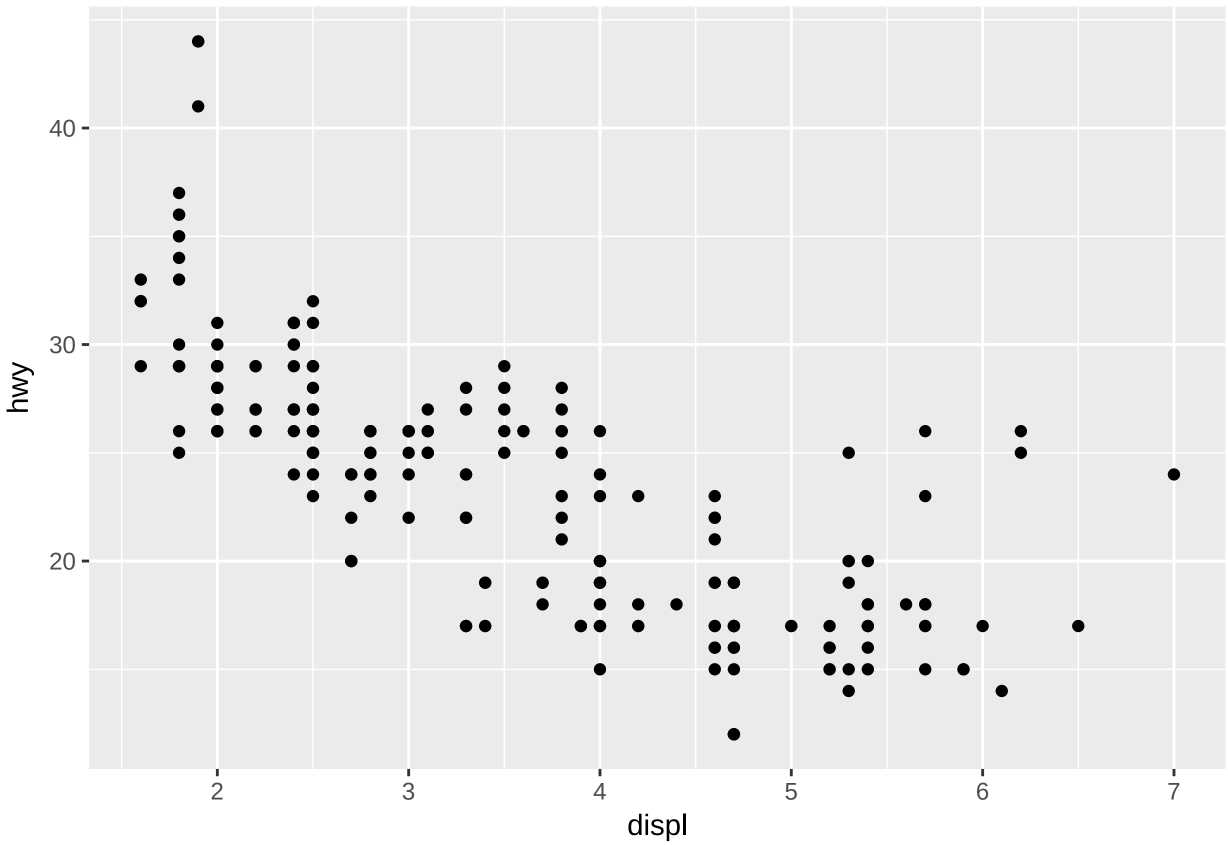
In the mpg data set, what are the relevant variables?

We can predict that cars with big engines are less fuel efficient than cars with small engines. The corresponding variables are:

- engine size: displ
- fuel efficiency: hwy

Let's plot the relationship between the two variables.

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

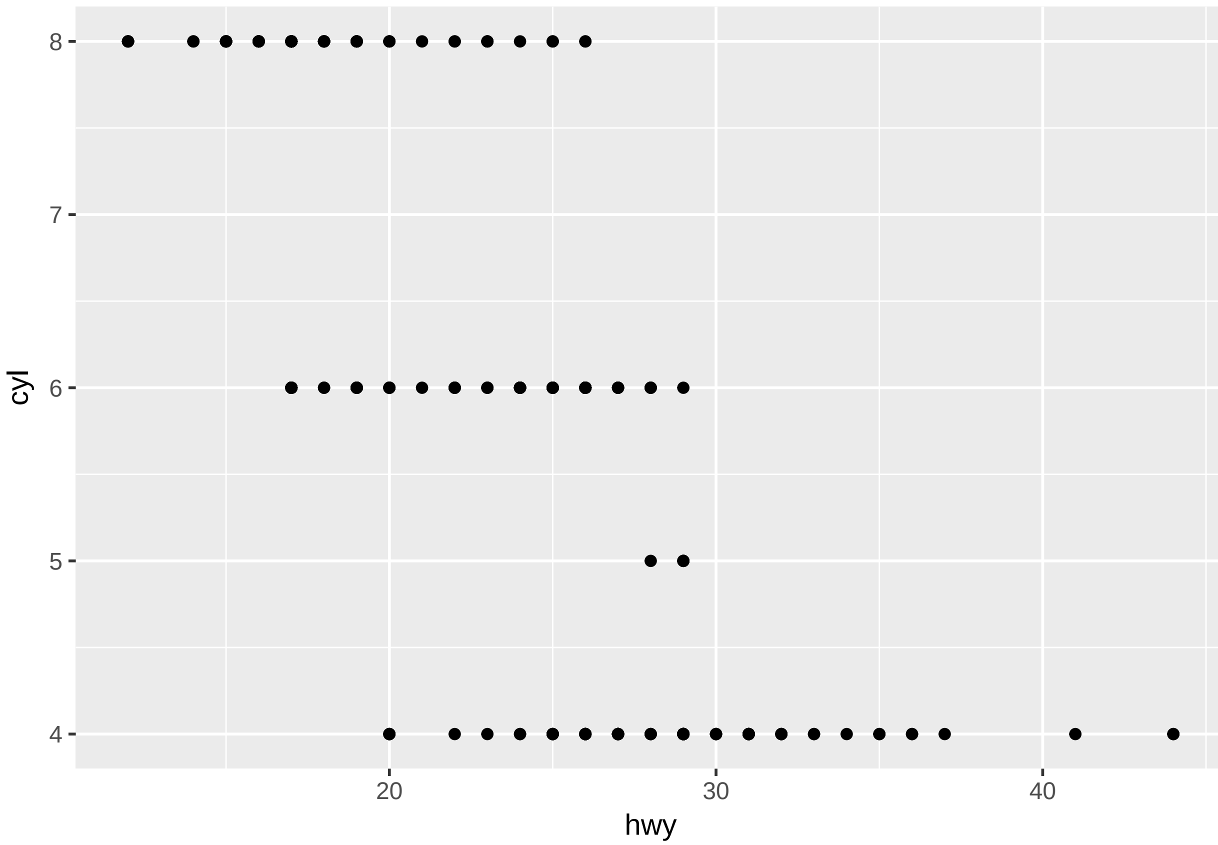


This plot shows a negtive relationship between displ and hwy, which means that cars with bigger engines use more fuel. Did you predict right?

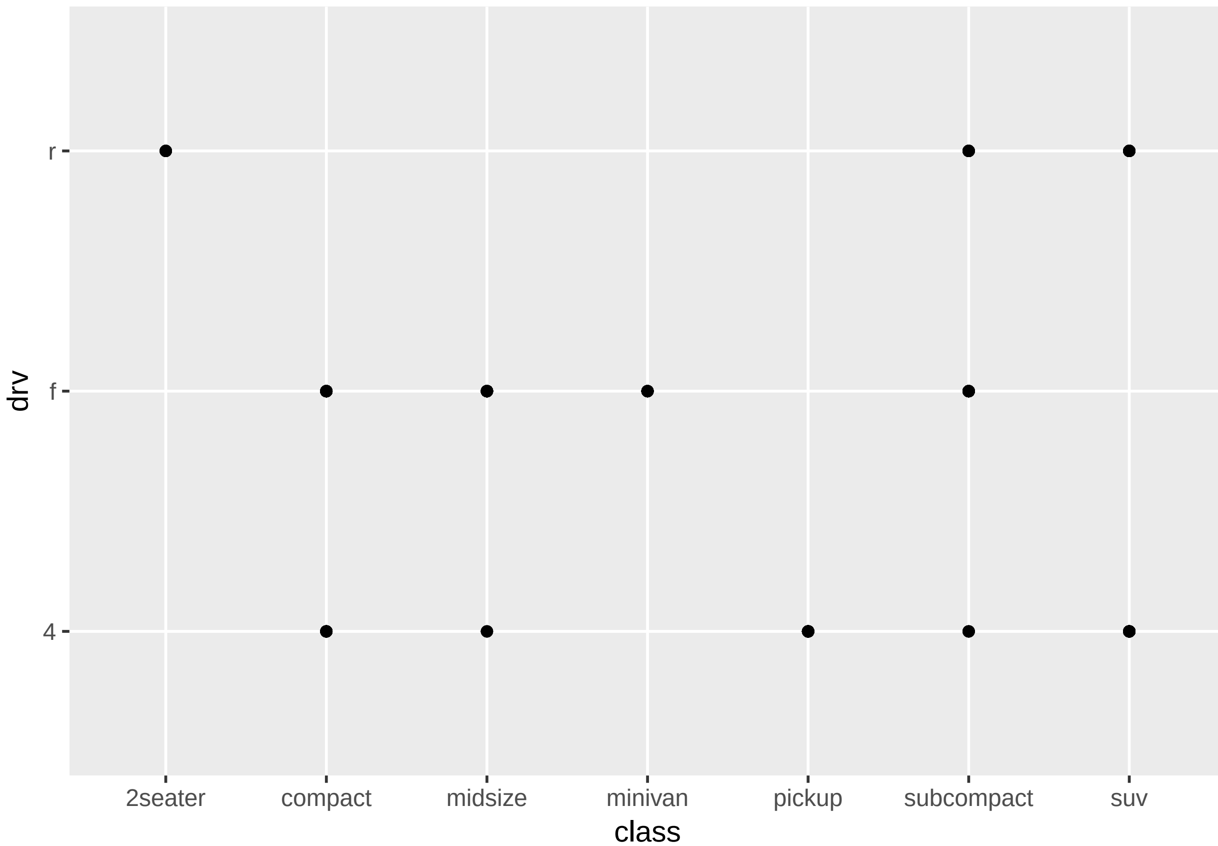
Exercises

1. Make a scatterplot of hwy vs cyl. Explain the result.
2. What happens if you make a scatterplot of class vs drv? Is the plot useful or not?

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

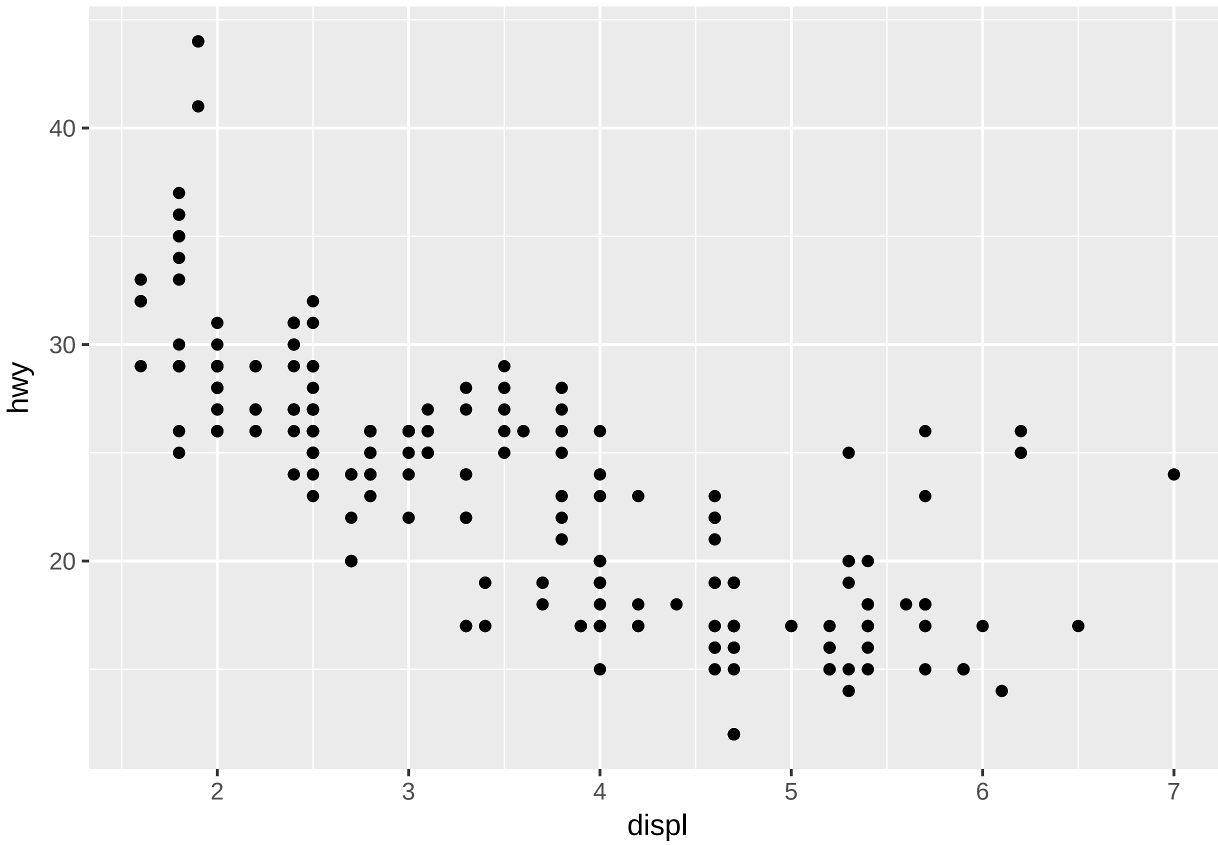



```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = class, y = drv))
```



Add a third variable (aesthetic mappings)

Let's look at the first scatter plot again.



There are a group of cars on the right side above the main trend that have higher fuel efficiency than other cars with similar engine sizes. Can you explain these outliers?

To find out what makes the outliers more fuel efficient, we can add another variable by mapping it to an aesthetic. An aesthetic is a visual property of the objects in your plot including the size, the shape, or the color of your points.

Additional variables should be added after the x and y aesthetics. For each aesthetic, ggplot automatically creates a legend.

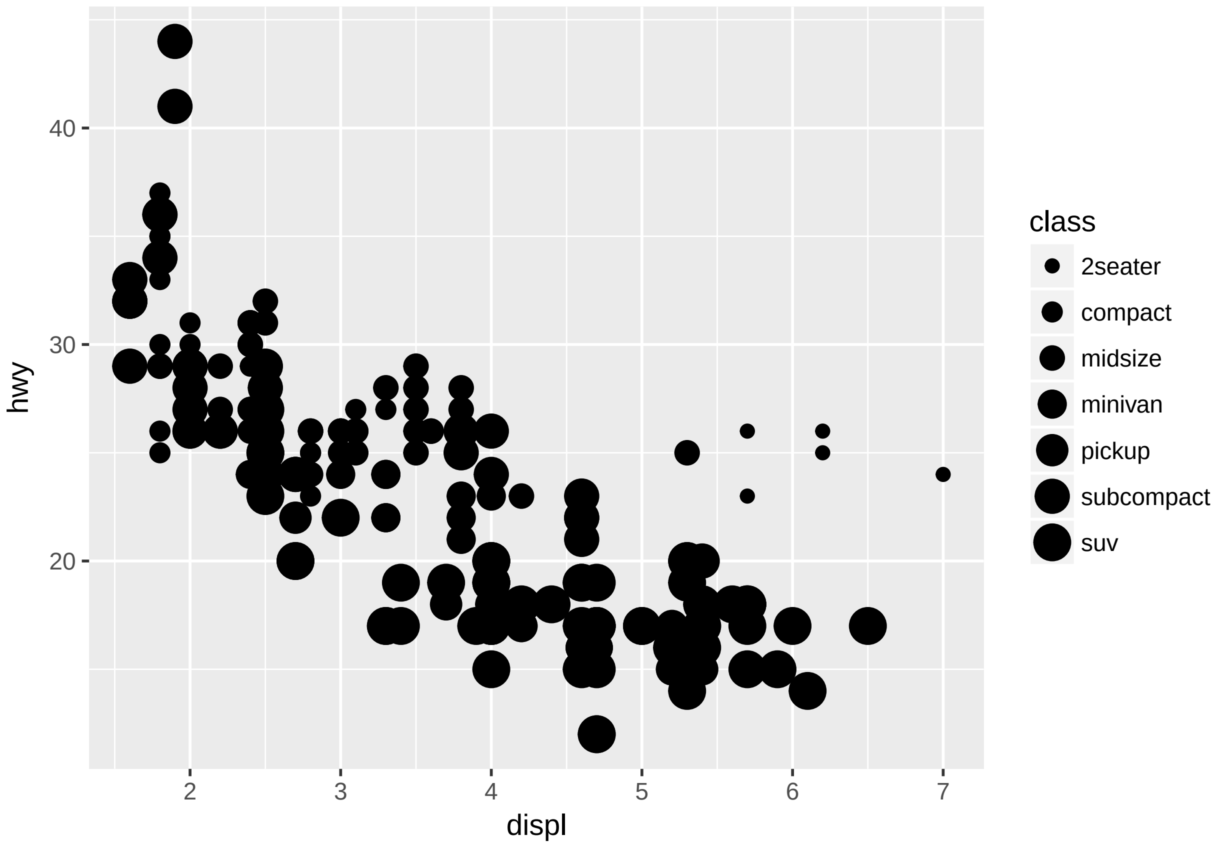
Let's add *class* as the third variable.

- Size

Vary the size with *class* so the exact size of each point would reveal its class affiliation

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

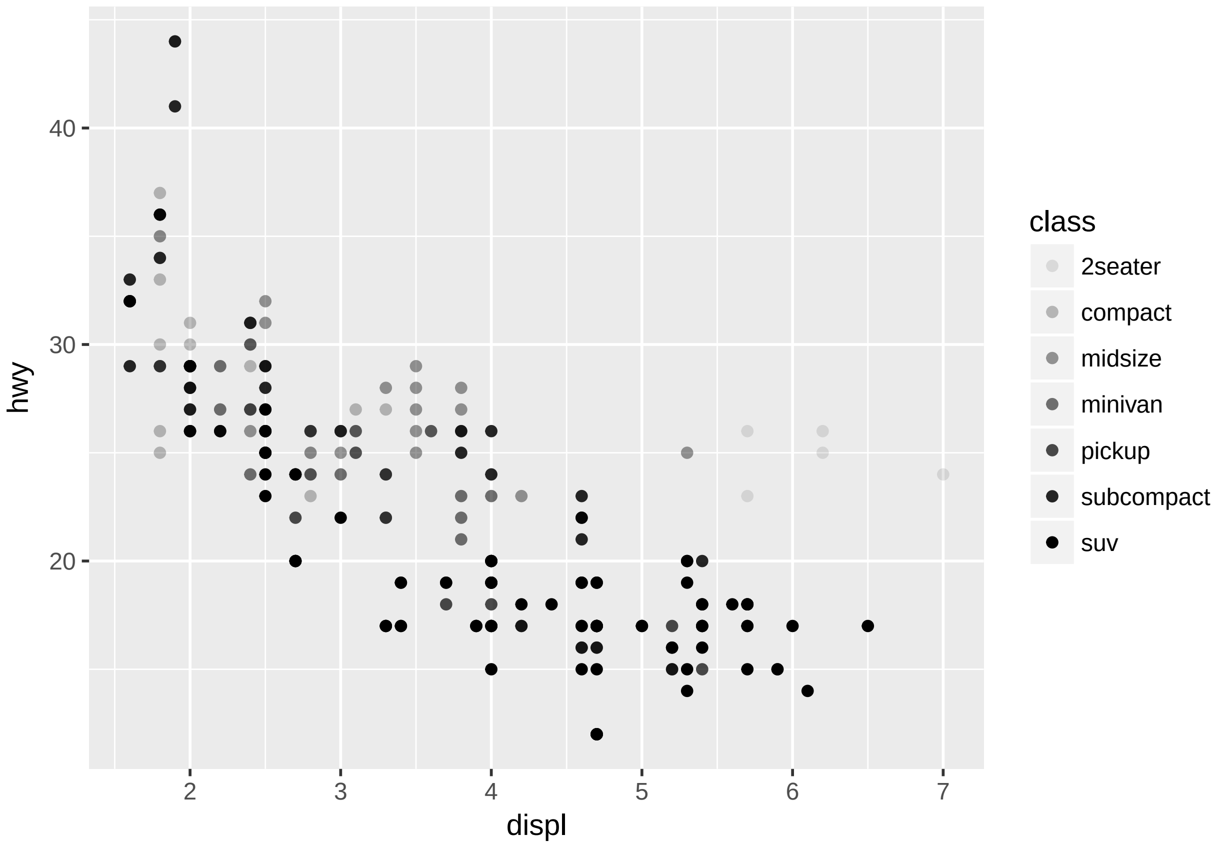
```
## Warning: Using size for a discrete variable is not advised.
```



- Alpha and Shape

We can also map *class* to the alpha aesthetic (opacity parameter) or the shape of the points.

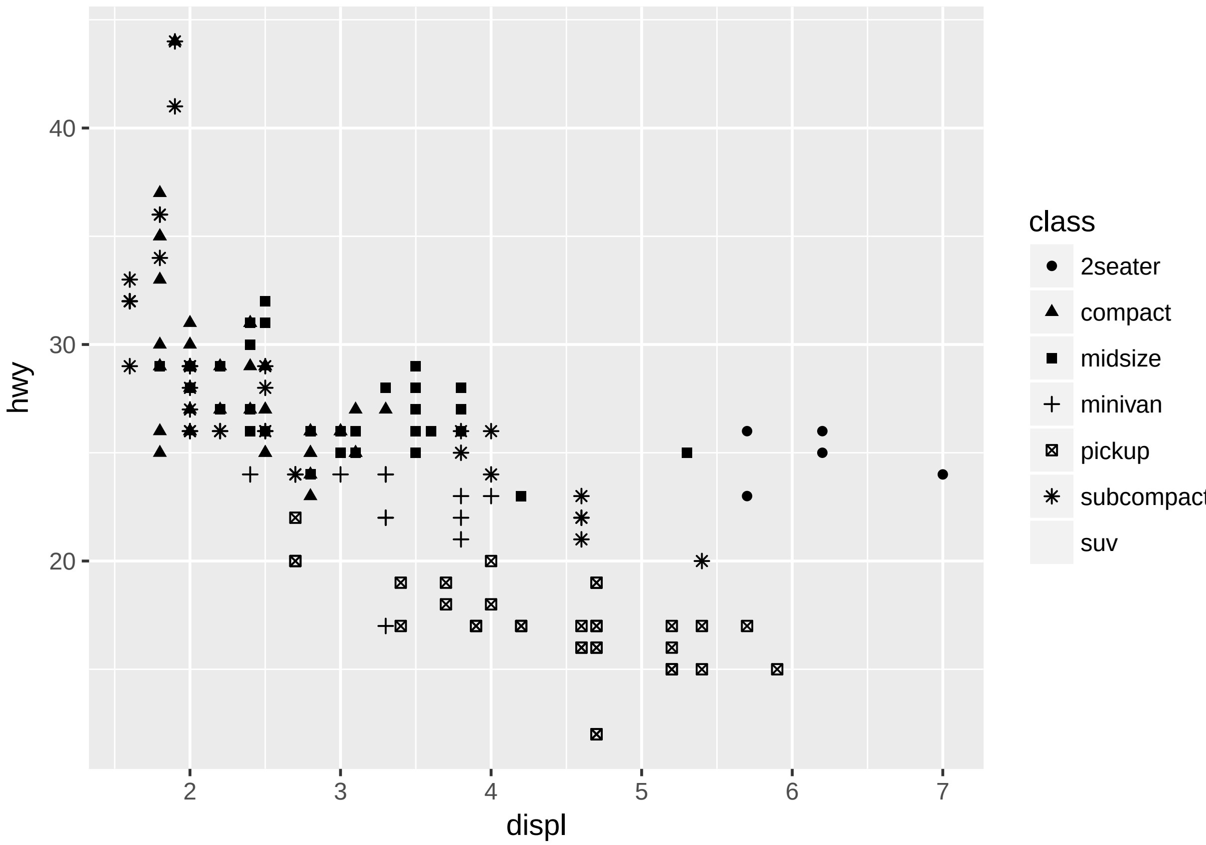
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, alpha = class))
```




```
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; you have 7.
## Consider specifying shapes manually if you must have them.
```

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

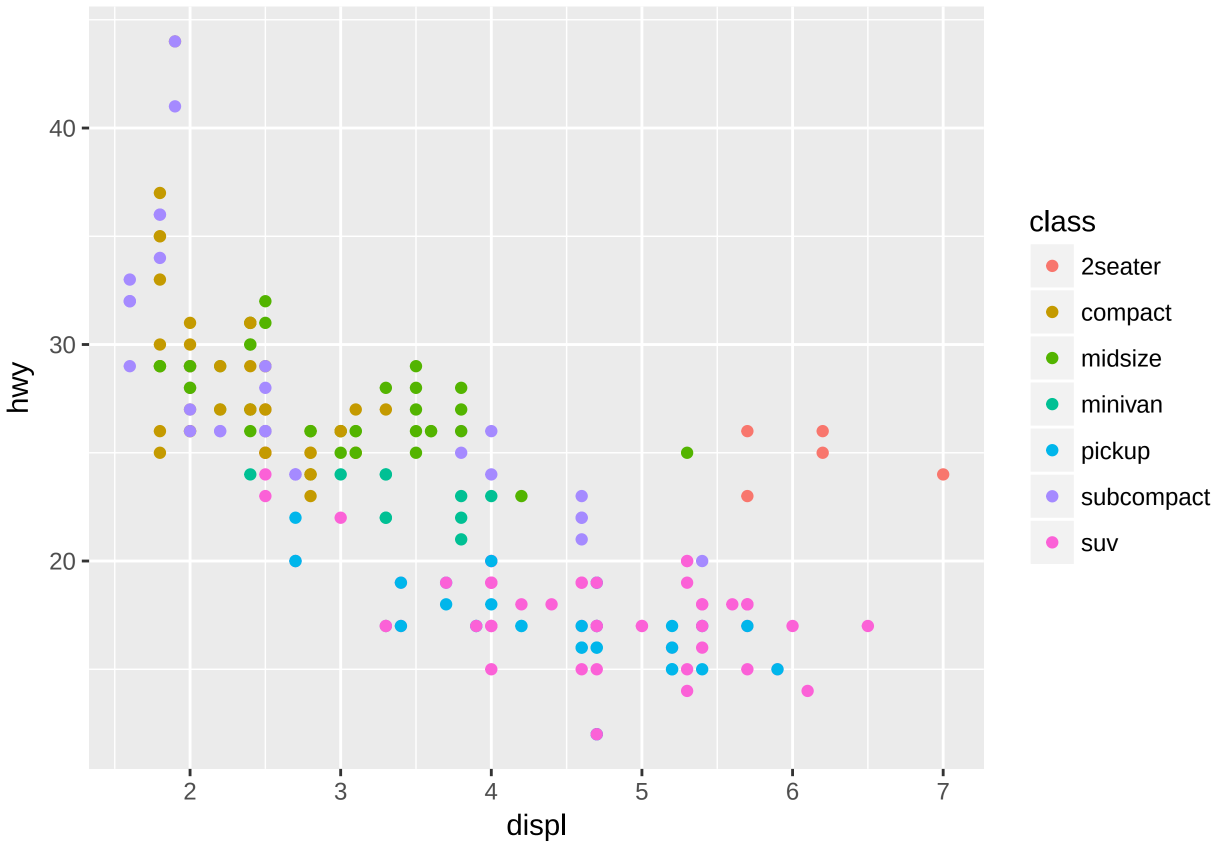


Notice the SUVs are not plotted here, because ggplot uses up to 6 shapes by default.

- Color

Last aesthetic is the color of each point.

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



Discrete and Continuous Variables

- Discrete variables have finite values (buckets). They are **counted** or **categorical**.
  - Example: number of students in a class, a person's age, phone model,...
- Continuous variables can have an infinite number of values. They are **measured**.
  - height of a child, speed of a car,...

Question: Which variables in mpg are discrete? Which are continuous? (3 min)

- Discrete: manufacturer, model, year, cyl, trans, drv, fl, class
- Continuous: displ, cty, hwy

Some of the aesthetic properties we've looked at are more suitable for discrete variables, and some are more suitable for continuous.

Exercise: Can you tell if aesthetics color, size, and shape are more suitable for continuous or discrete variables? Think about the aesthetics: Which ones are continuous and which are discrete?

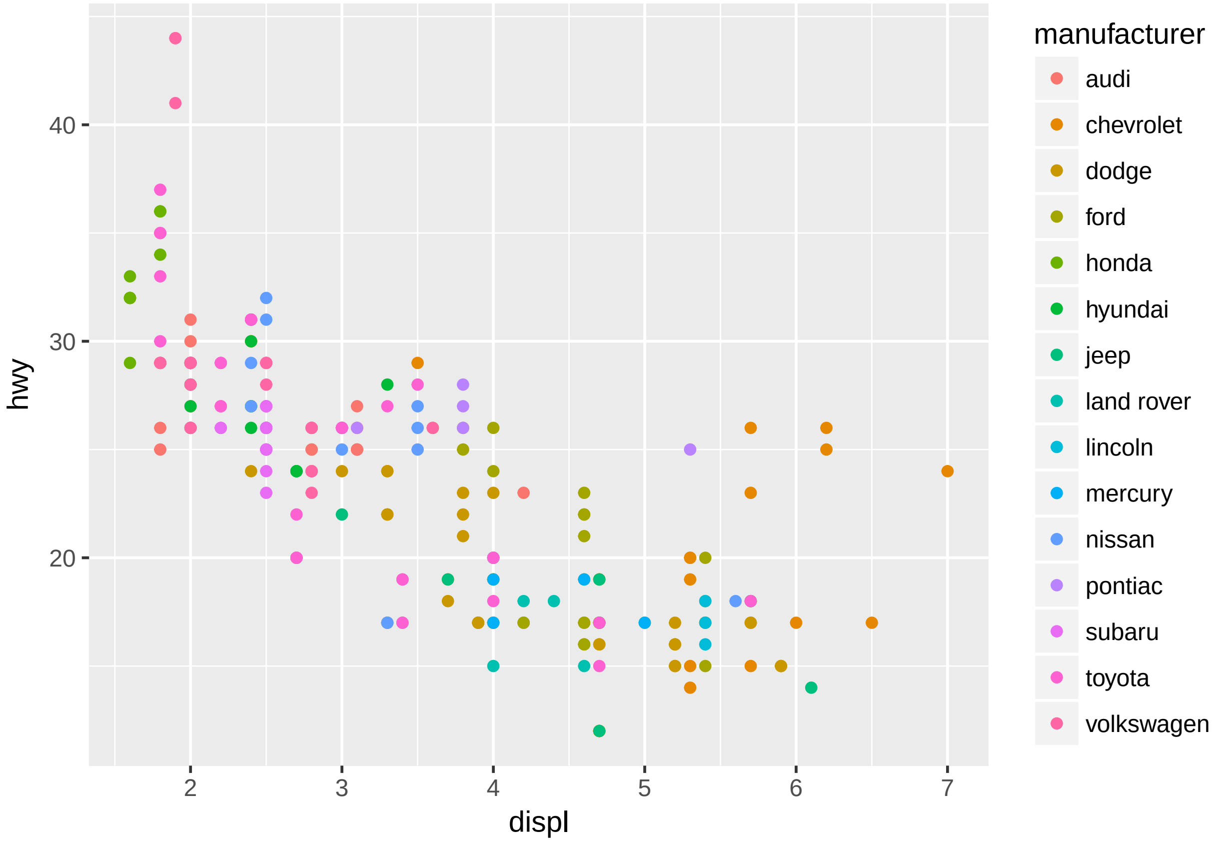
Check by mapping a discrete and a continuous variables to each aesthetic.

---

color

- discrete

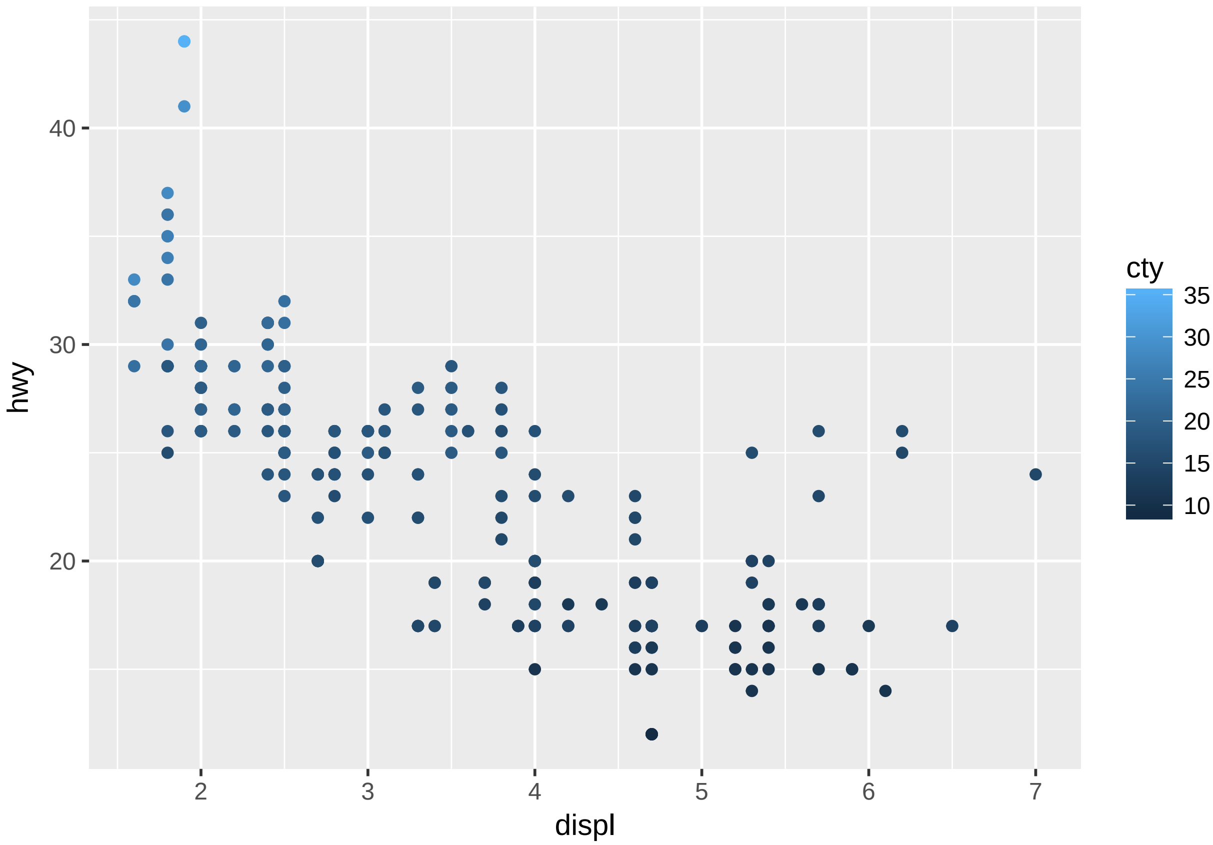
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = manufacturer))
```



Works fine

- continuous

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



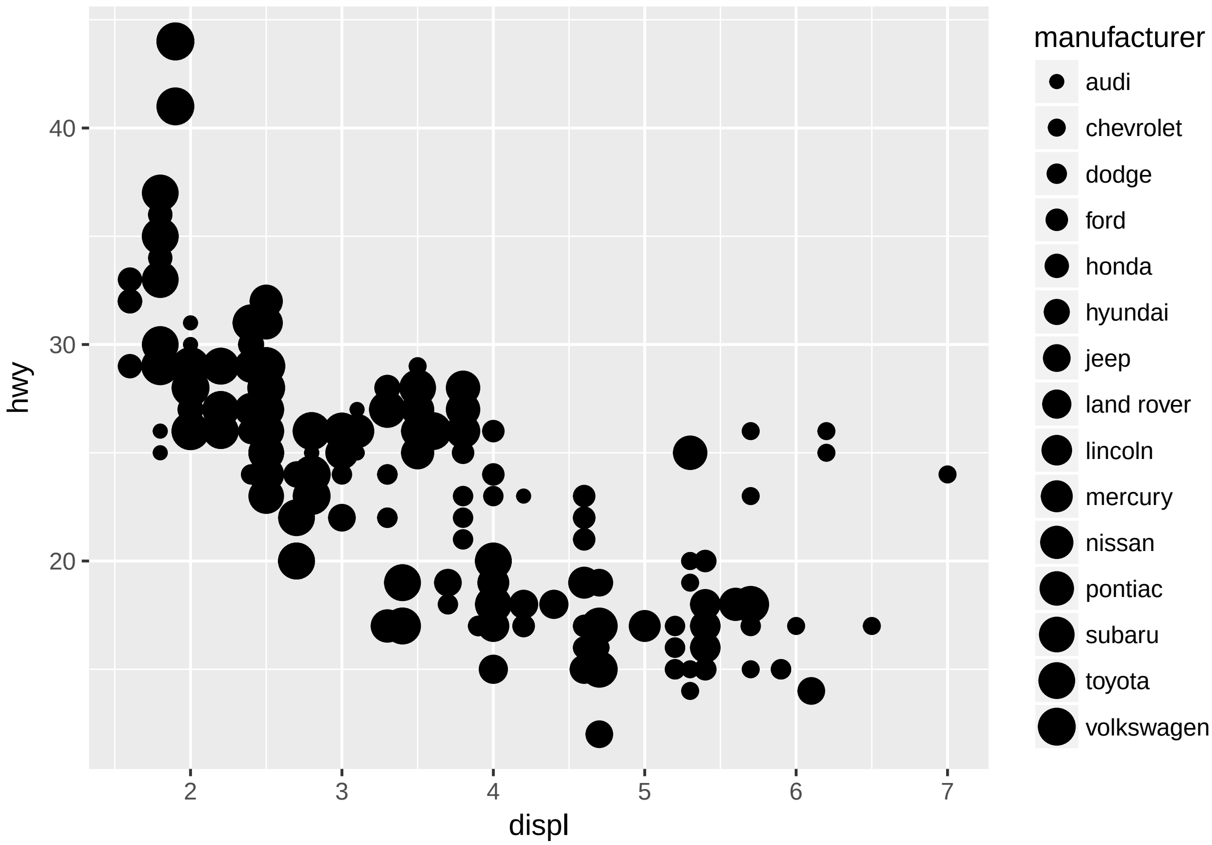
Works fine too. Notice the color legend is continuous now.

size

- discrete

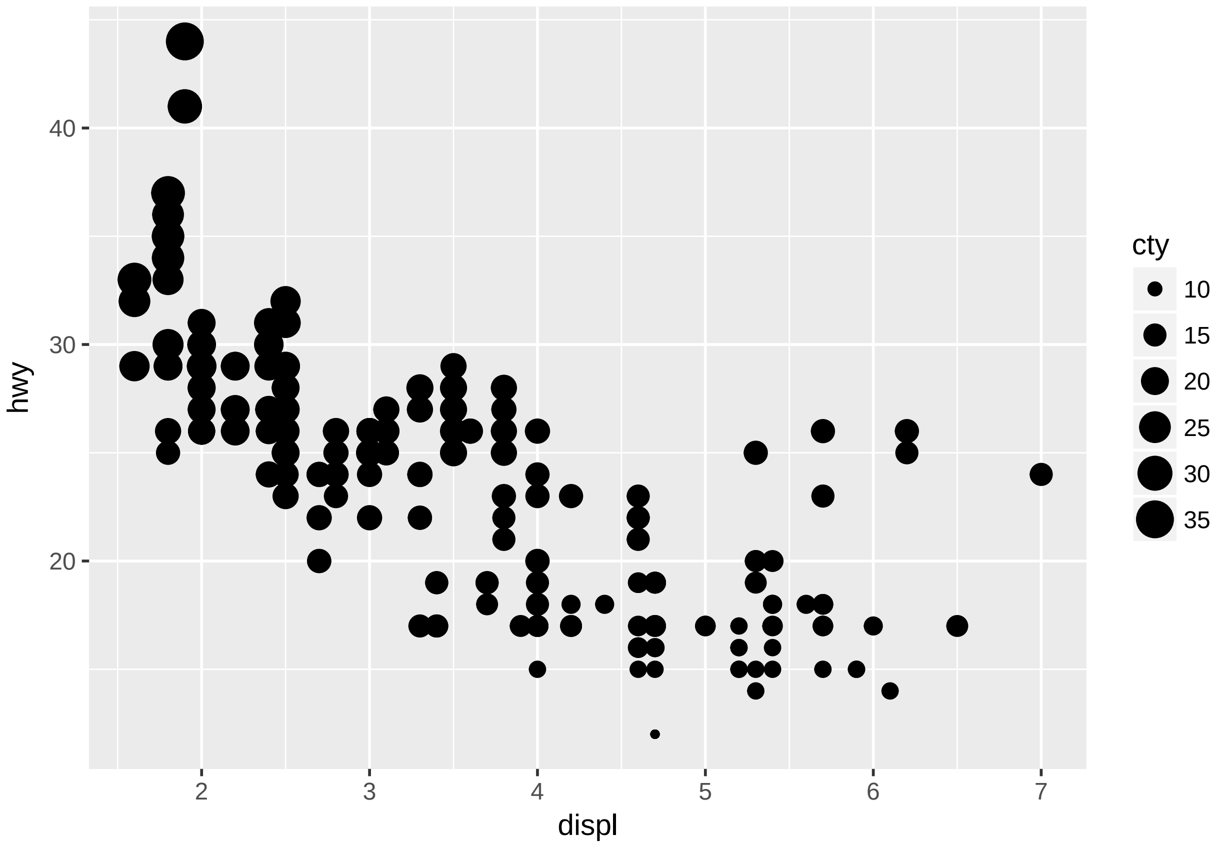
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, size = manufacturer))
```

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



Works ok, but mapping a discrete variable to a continuous aesthetic *size* is not advised.

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

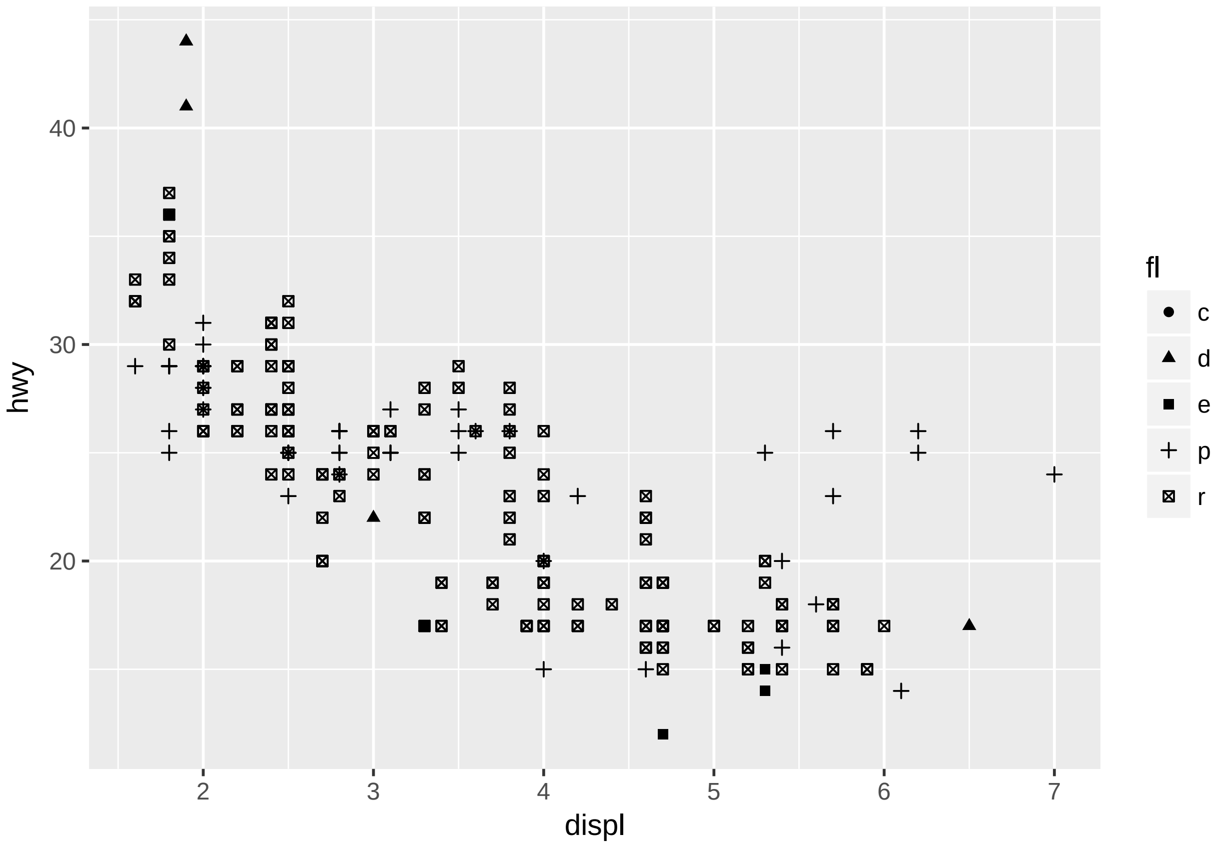


Works well.

shape

- discrete

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



Works best for a variable with a few discrete values.

- continuous

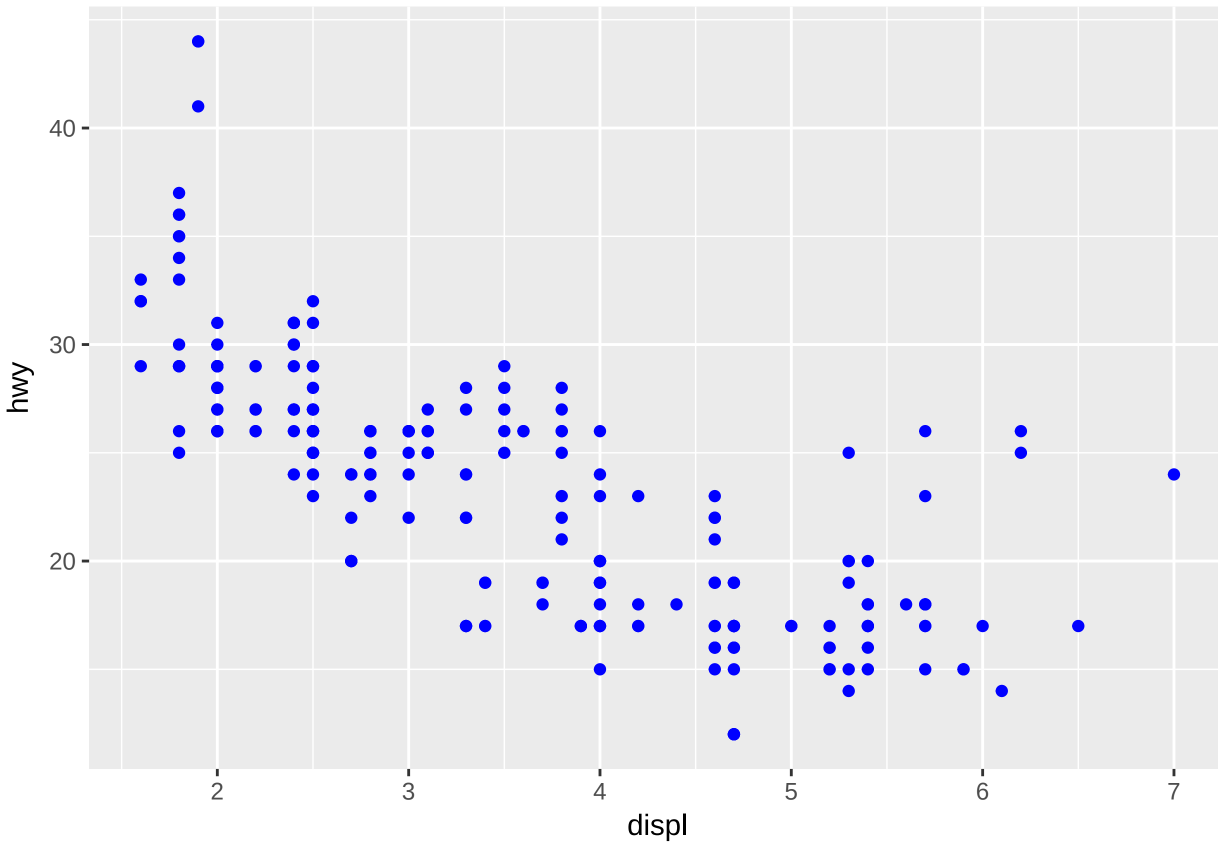
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, shape = cty))
```

Doesn't work

Set aesthetics for entire plot

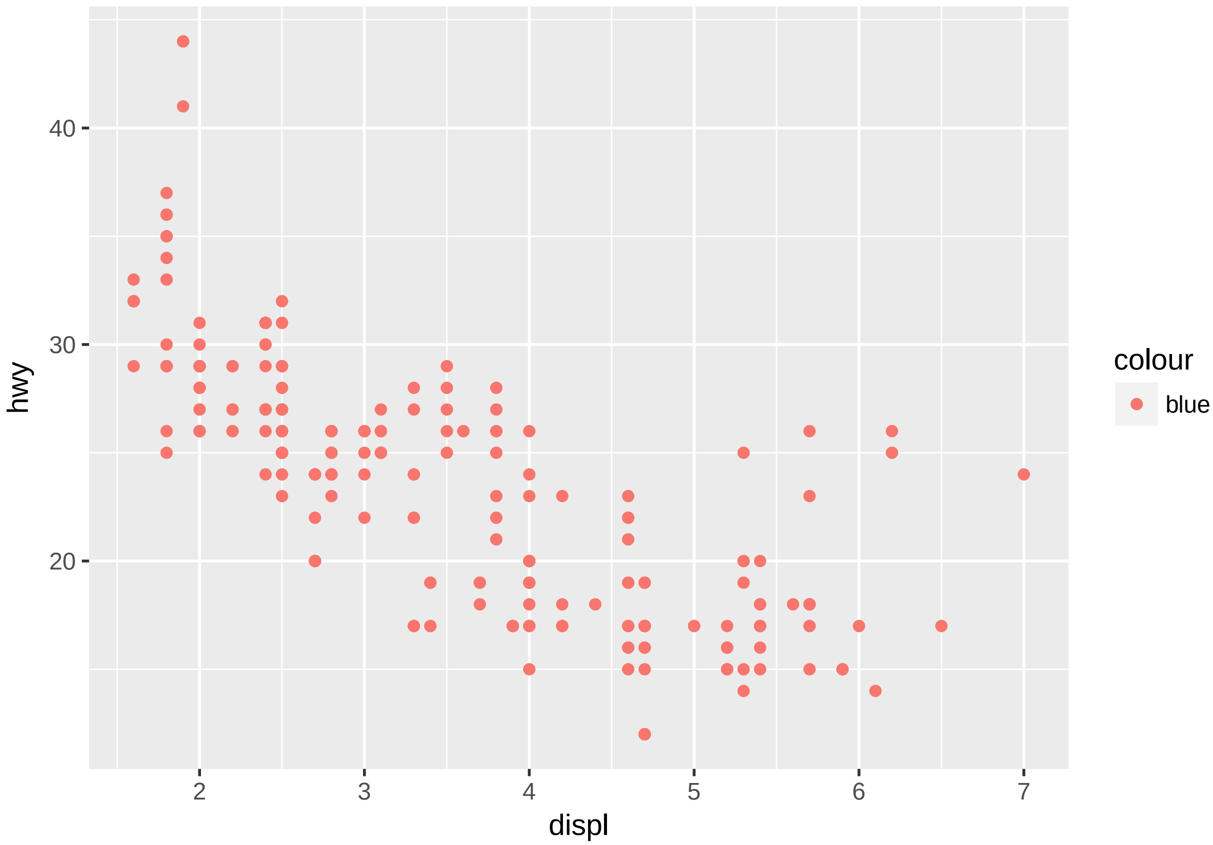
You can set the aesthetic properties manually. The code below makes all the points blue. How is this code different from the previous codes?

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



Compare with

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



In the first code, *color* is not a variable, and it is outside the *aes( )*. You can change the appearance of the entire plot similarly by setting the size, alpha, shape, etc.

- color: Specify the name of a color as a character string (inside " “)
- size: Specify the size of a point in mm
- shape: Specify the shape of a point as a number, as shown below

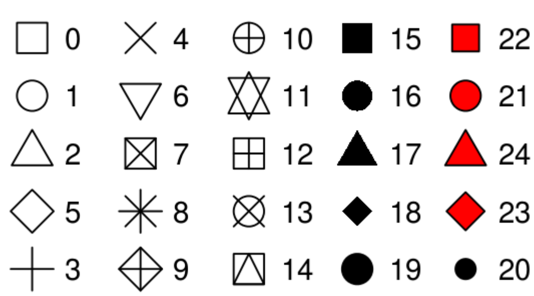


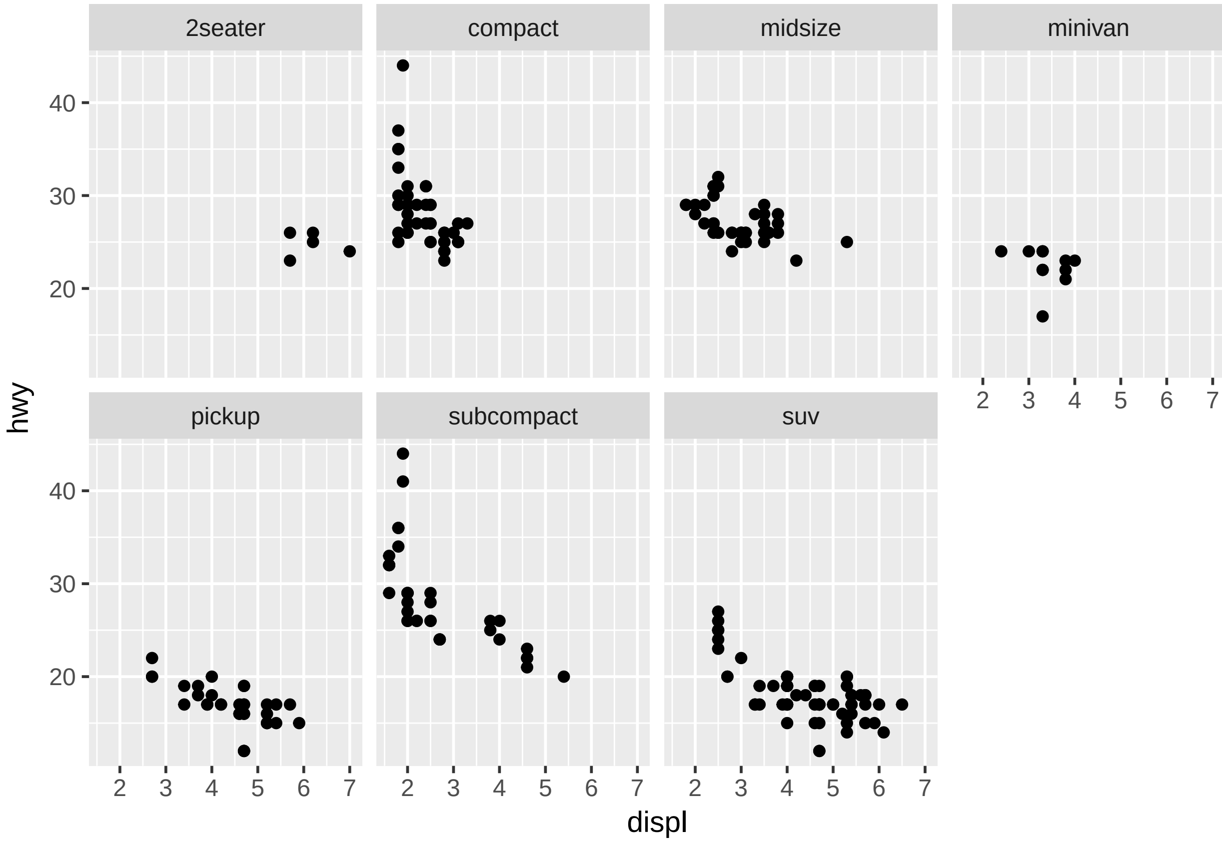
Figure 1:

Facets

Another way to add additional variables (other than aesthetics) is to split your plot into **facets**. Facets are subplots each of which displays a subset of data. Facets are useful for discrete variables.

To use a single variable to facet your plot, use **facet_wrap( )** after geom_point( ). The argument of facet_wrap( ) should start with ~ followed by a variable name. Let's facet on **class**.

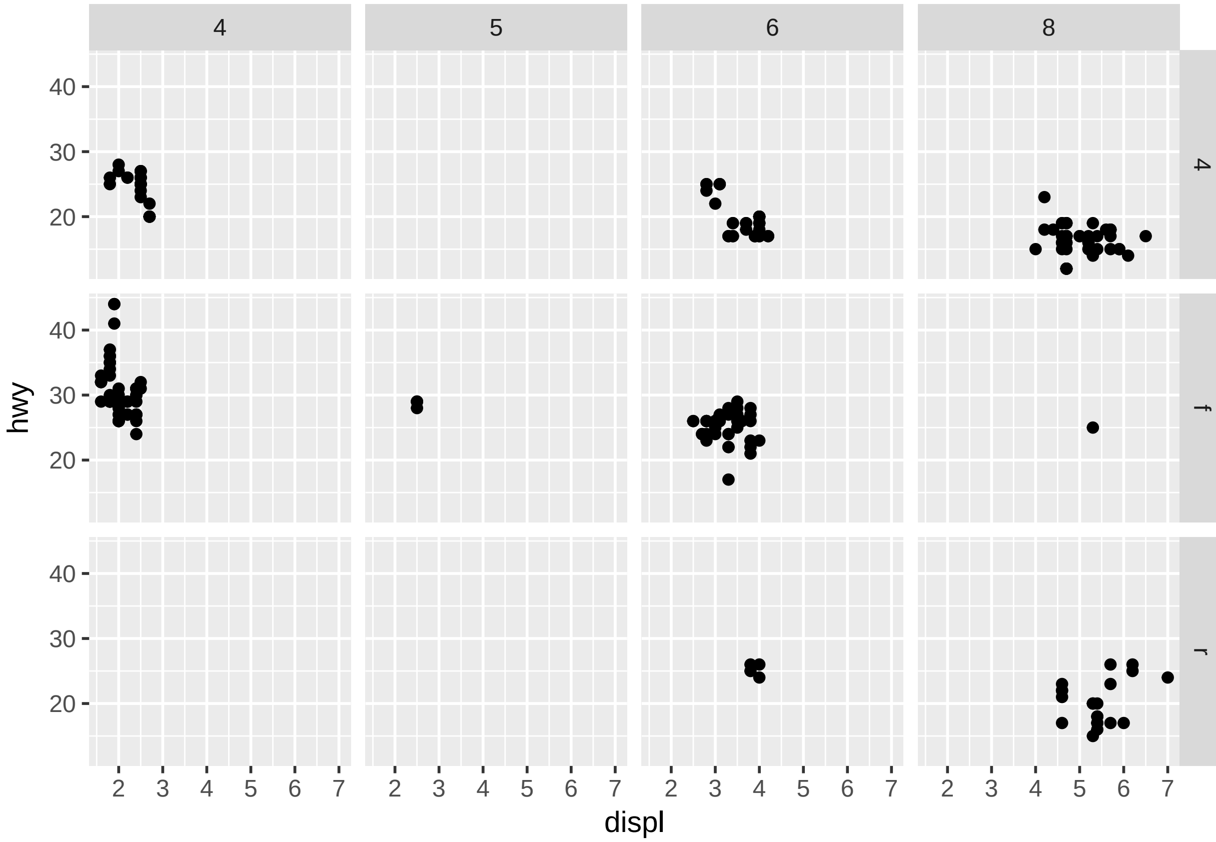
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_wrap(~ class, nrow = 2)
```



To facet on a combination of variables, use **facet_grid( )**. This time the argument is two variable names separated by ~.

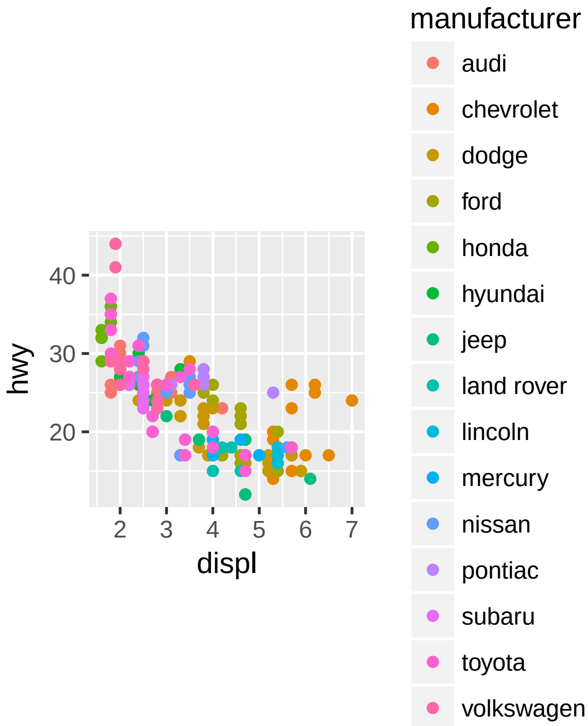
The code below facets on drv and cyl.

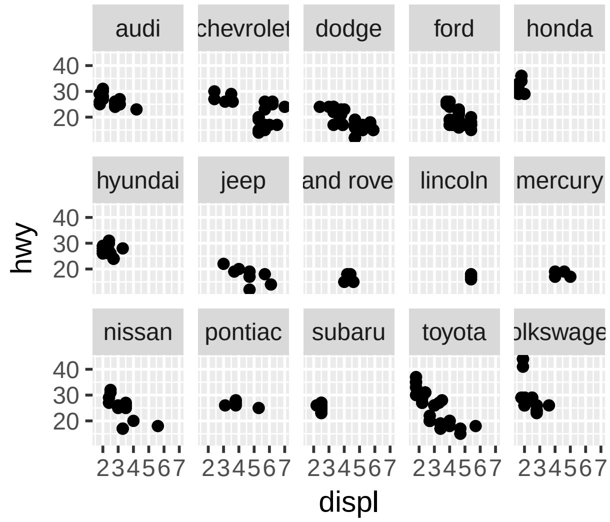
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_grid(drv ~ cyl)
```



Exercises

1. Recreate following two plots.

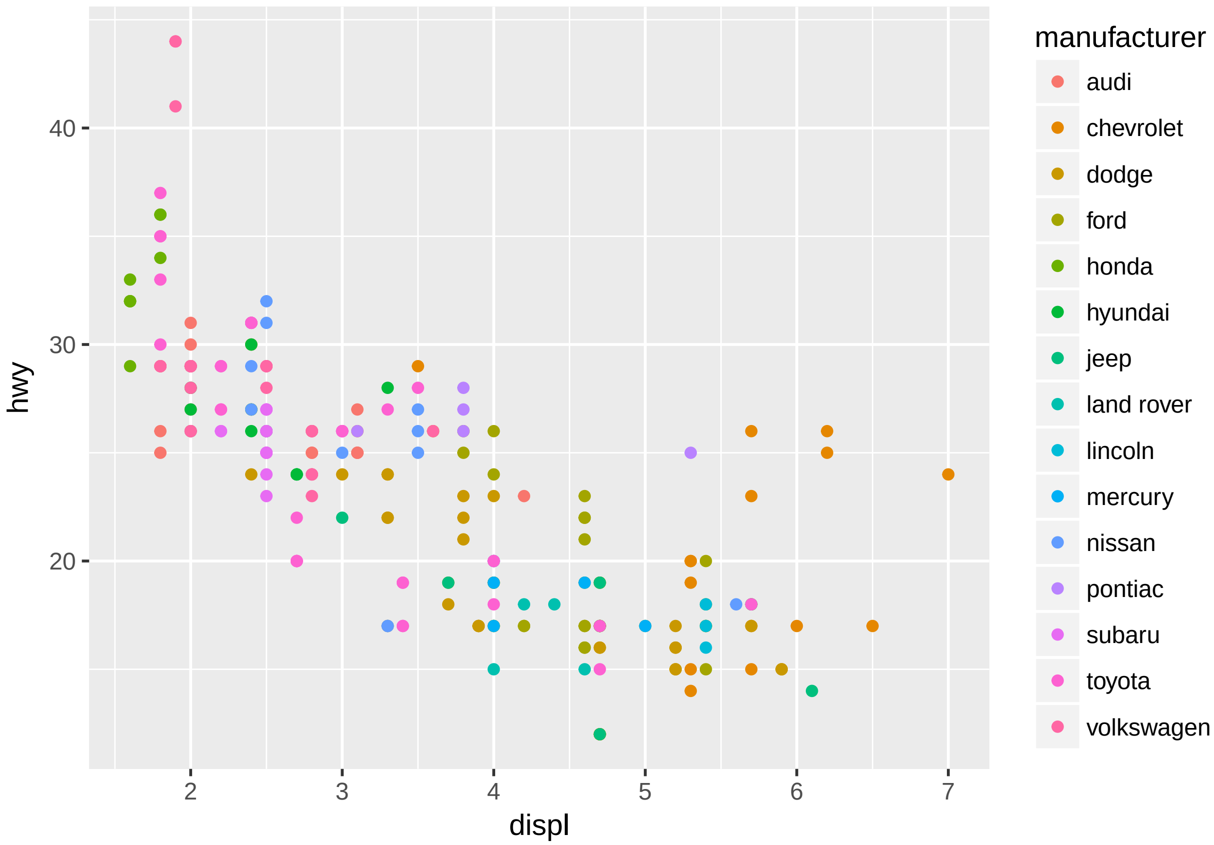




2. What are the advantages of using facets instead of the color aesthetic? Consider the above two plots. What if we had more data points?

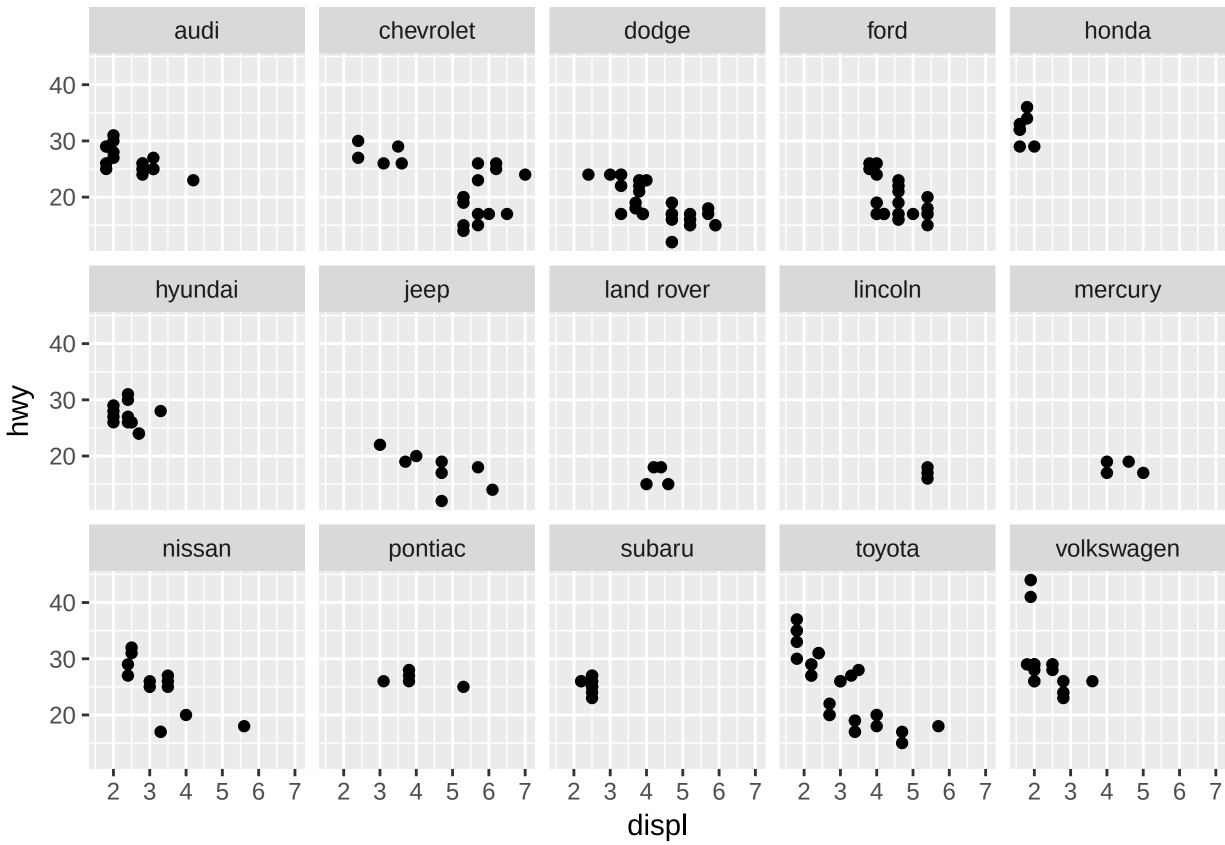
Plot 1

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



Plot 2

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_wrap(~ manufacturer, ncol = 5)
```



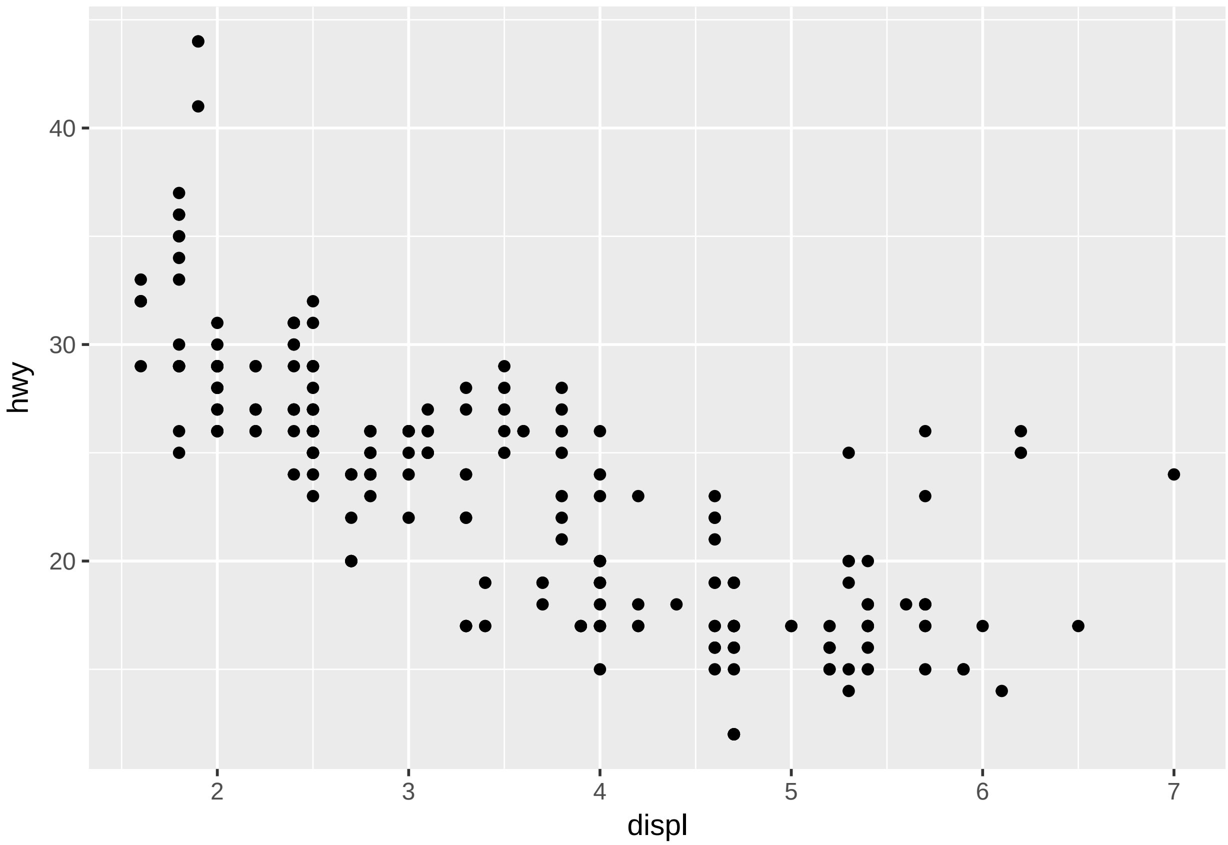
Facets are useful when the data is large (many data points), or when a variable has a lot of values.

Fitting a smooth line: geom_smooth( )

To plot a smooth line fit to the data points, use a geom function **geom_smooth( )**.

For example, let's fit a line to this plot

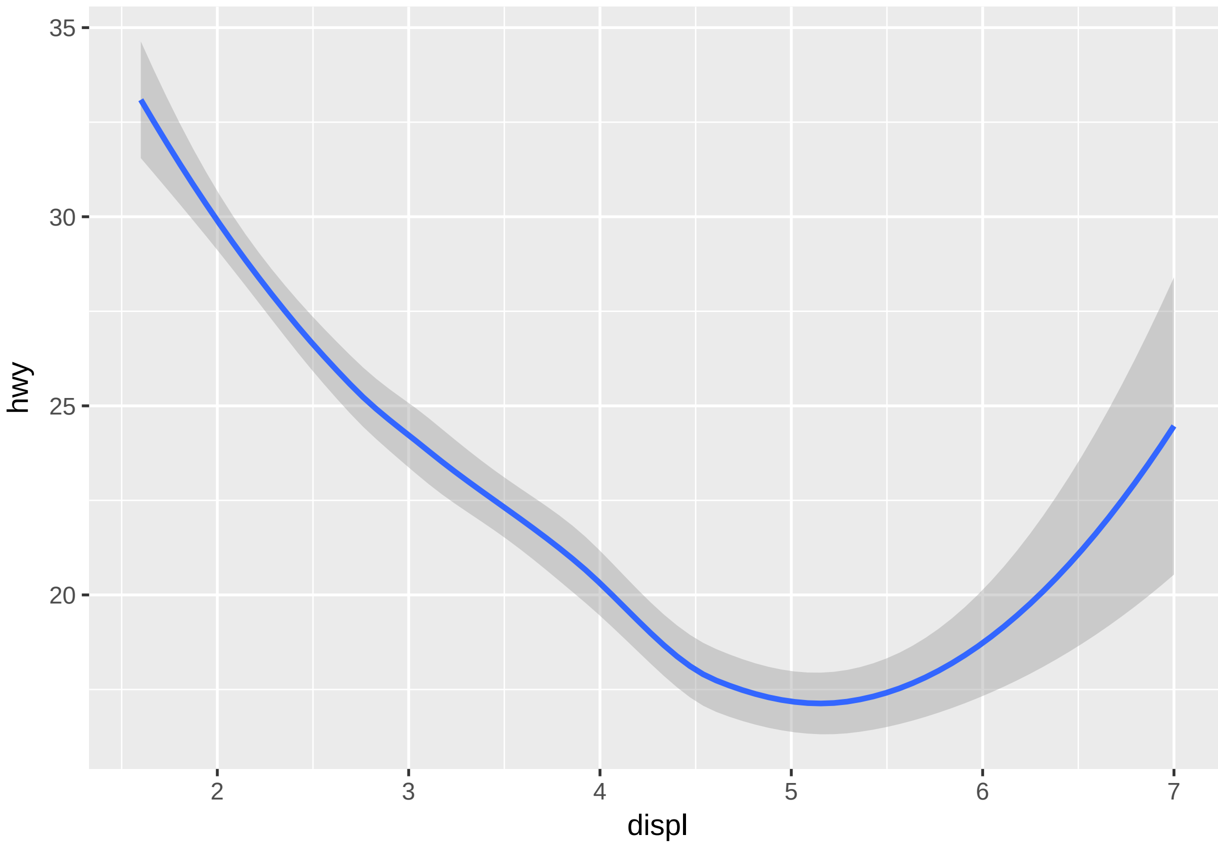
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```



We can just replace the geom function

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

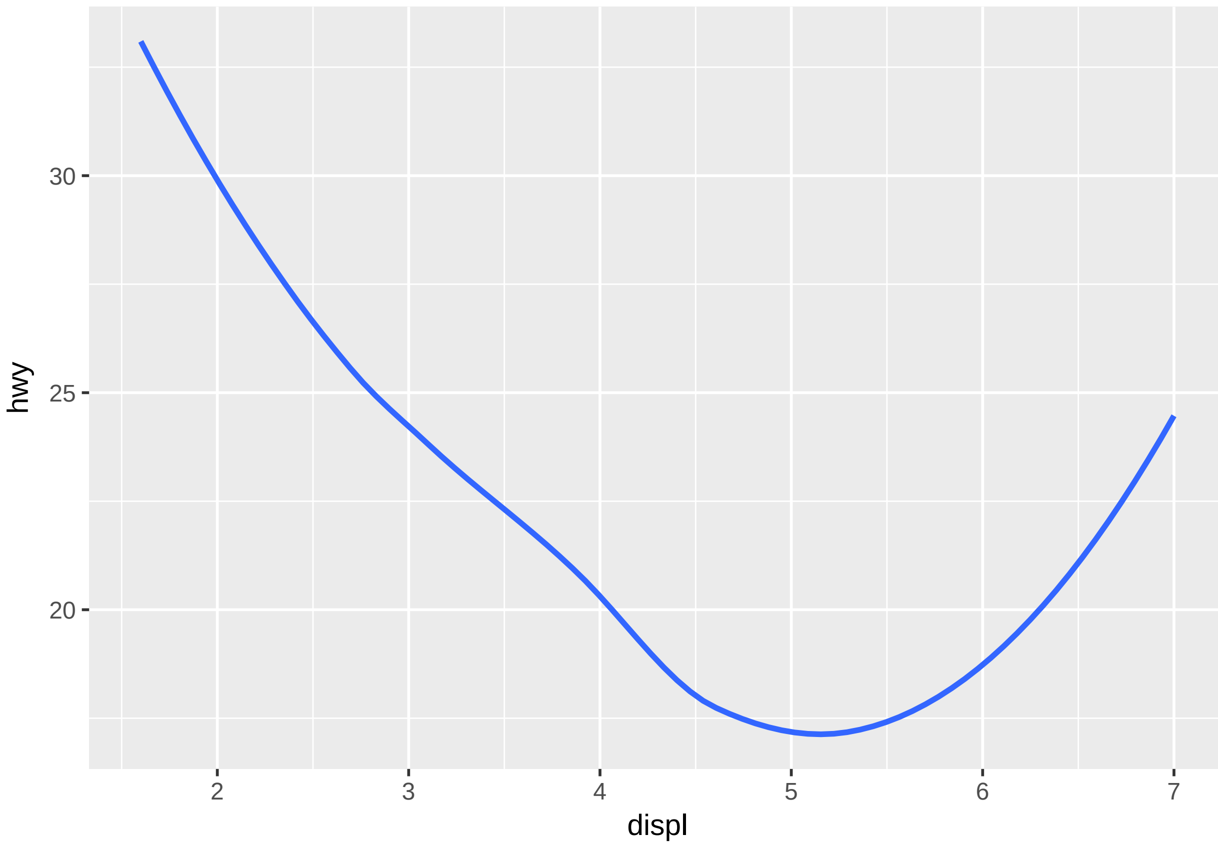
```
## `geom_smooth()` using method = 'loess'
```



Blue solid line is the fit, gray shade is the confidence interval. By default, confidence level is set to 95%, which means there is a 95% probability that true best fit line lies within gray shade. To remove confidence interval, set se = FALSE

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

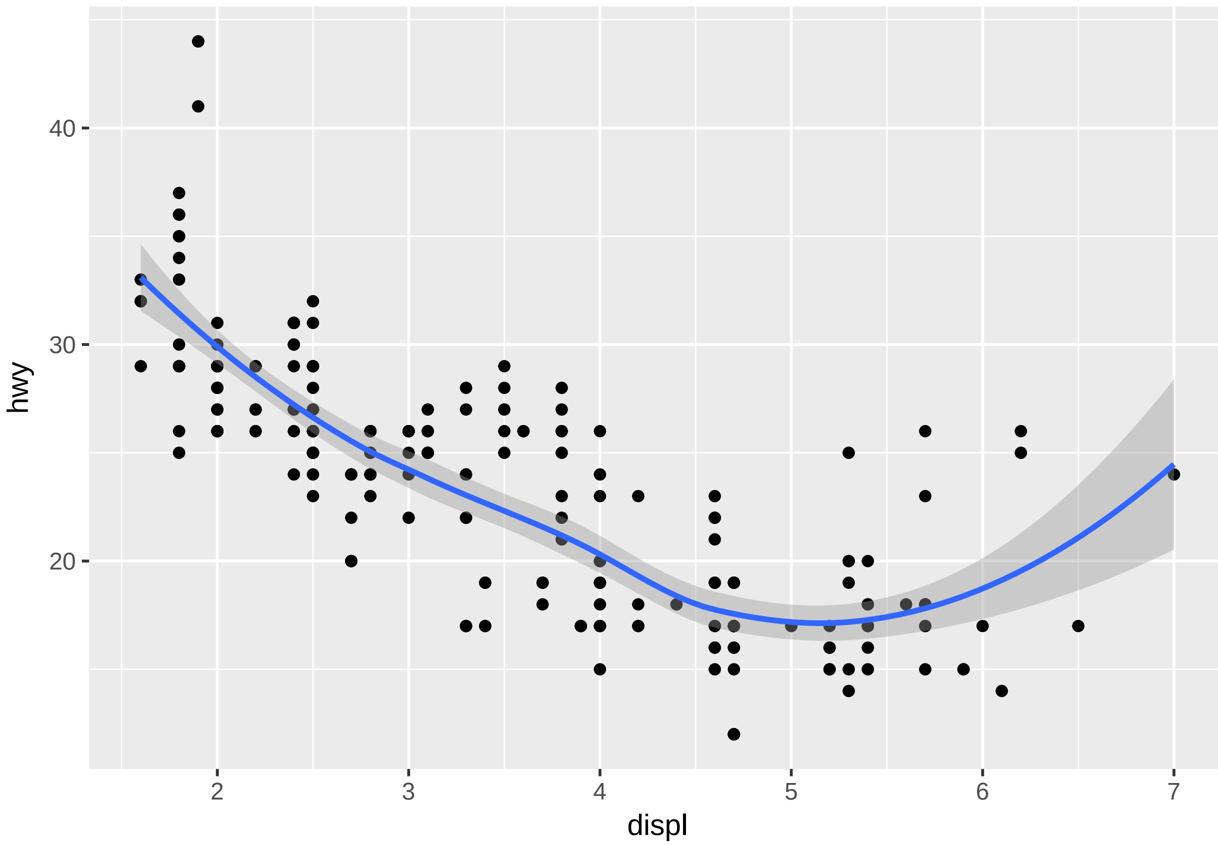
```
## `geom_smooth()` using method = 'loess'
```



To check the line fits the data, plot both points and line

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

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

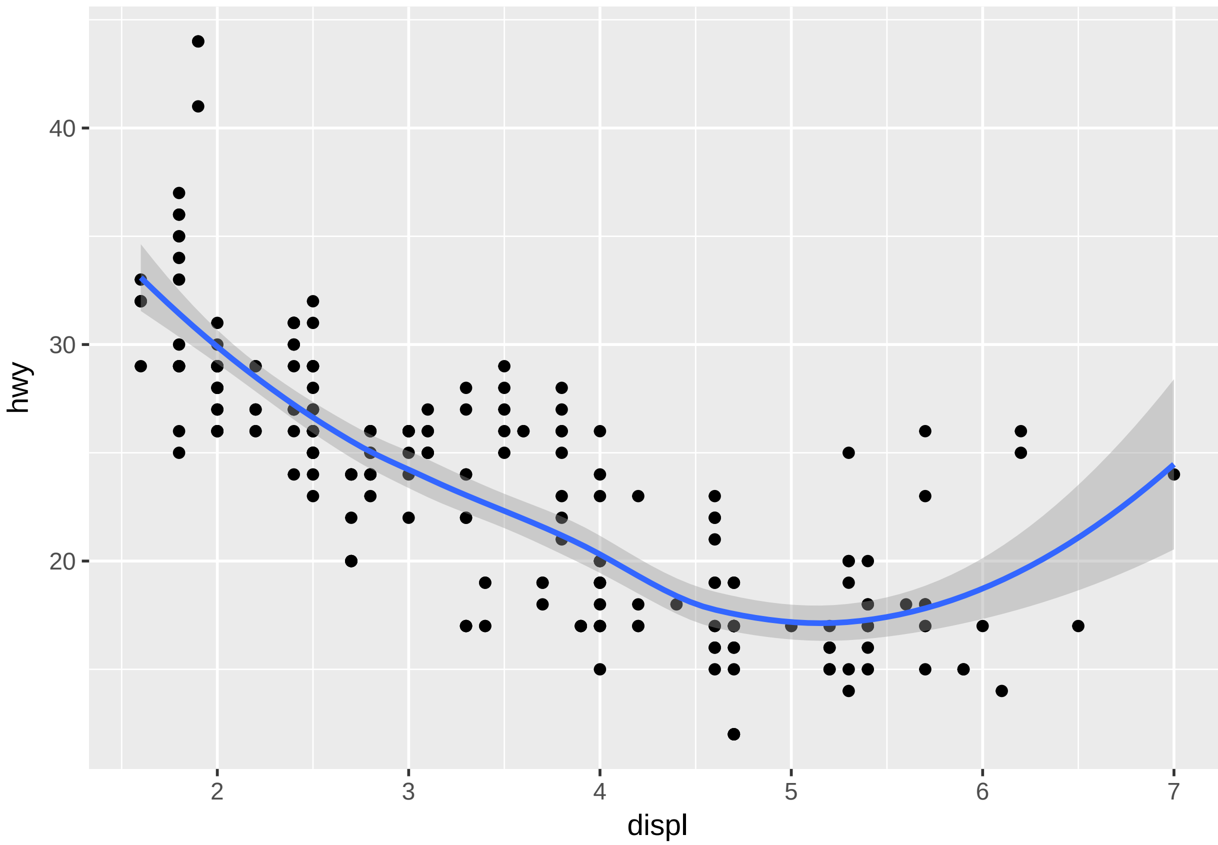


Does the line fit the data points? Why does the gray shade wider on the right side?

The code chunk used to produce above plot looks redundant. There is a way to avoid repeating the same mapping argument. You can pass the mapping argument to ggplot. Then the argument will be applied to each geom function.

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

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

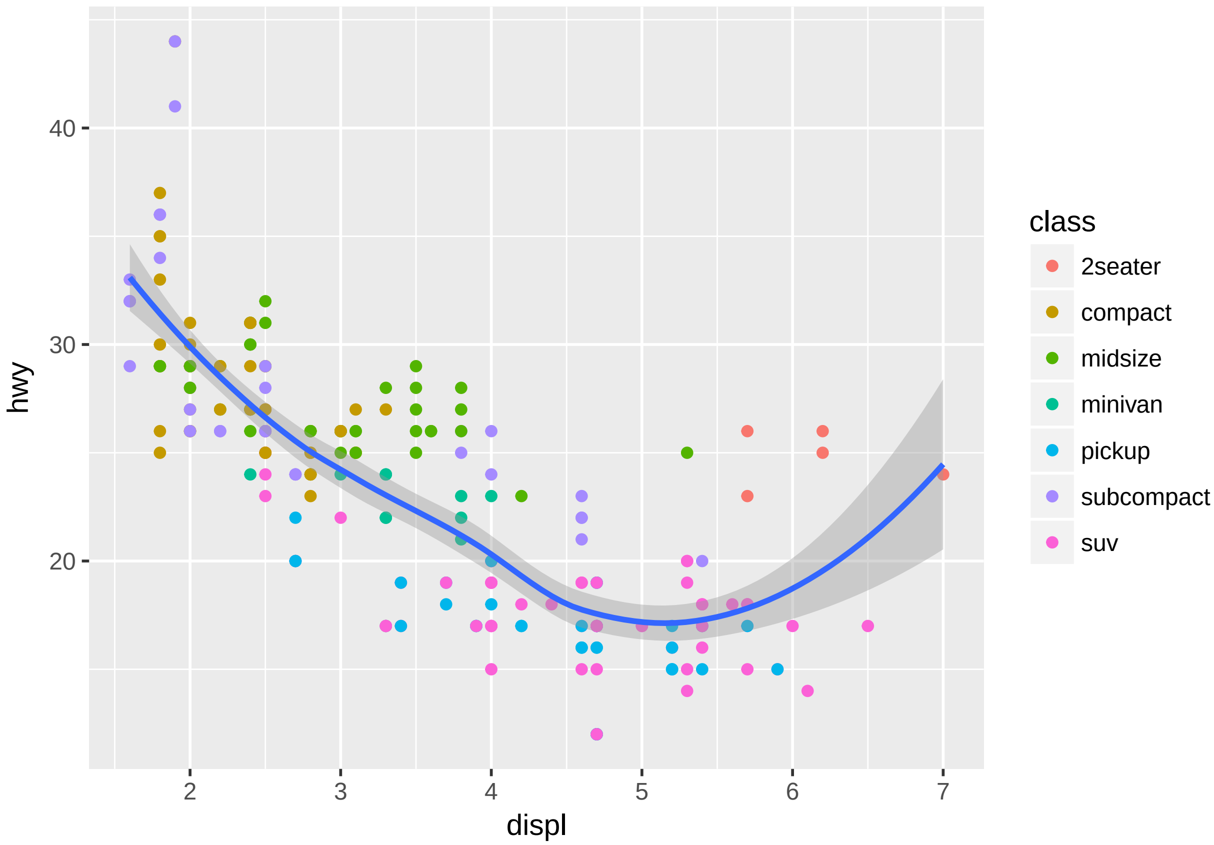


The result is identical, but this code looks simpler.

Mappings added to each geom function will be applied only to the geom. For example, add a color variable to geom_point.

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

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



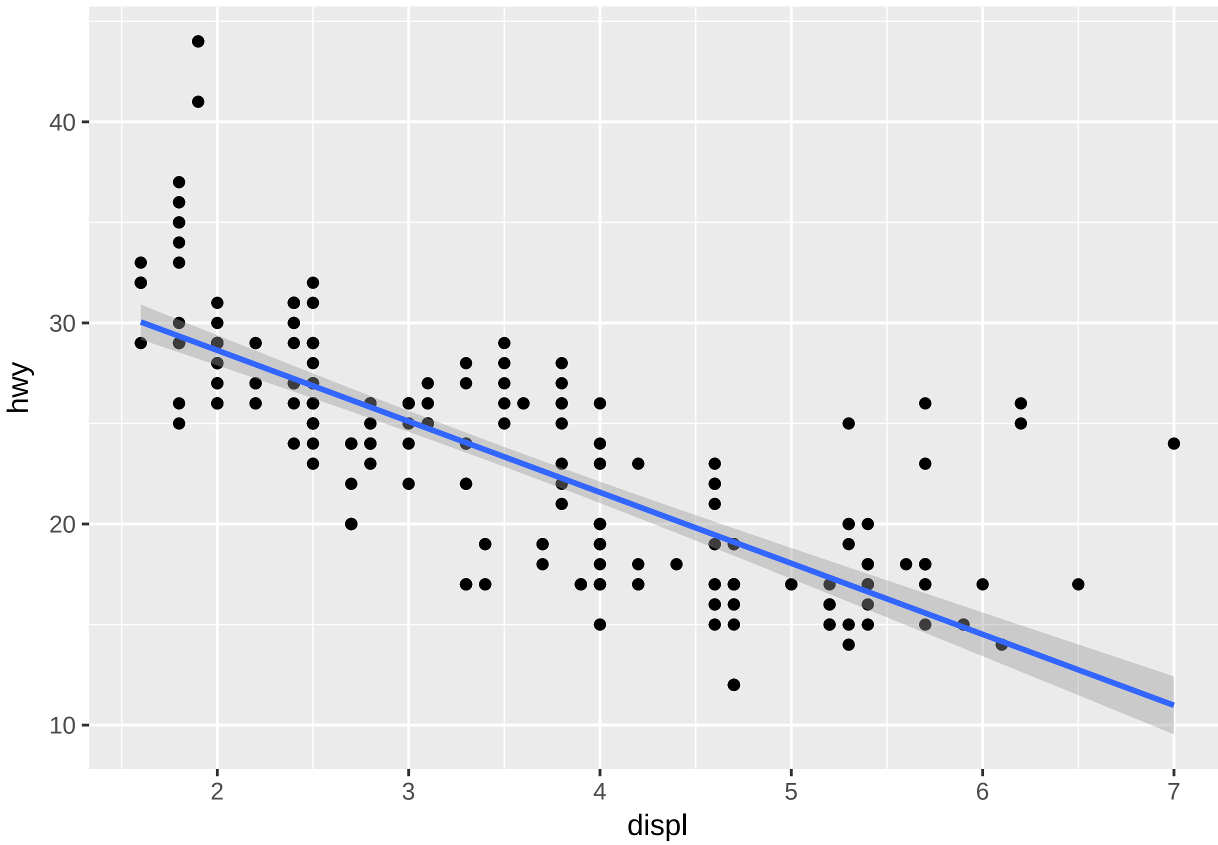
The color argument is only applied to points, not line.

Different fitting methods

Default method is set to loess (Locally Weighted Scatter-plot Smoother), which is a non-parametric regression method.

To use a different fitting method, you can specify it. If you want to fit a linear model, add method = “lm”

```
ggplot(mpg, aes(displ, hwy)) +
  geom_point() +
  geom_smooth(method = "lm")
```



Add a mapping to geom_smooth( )

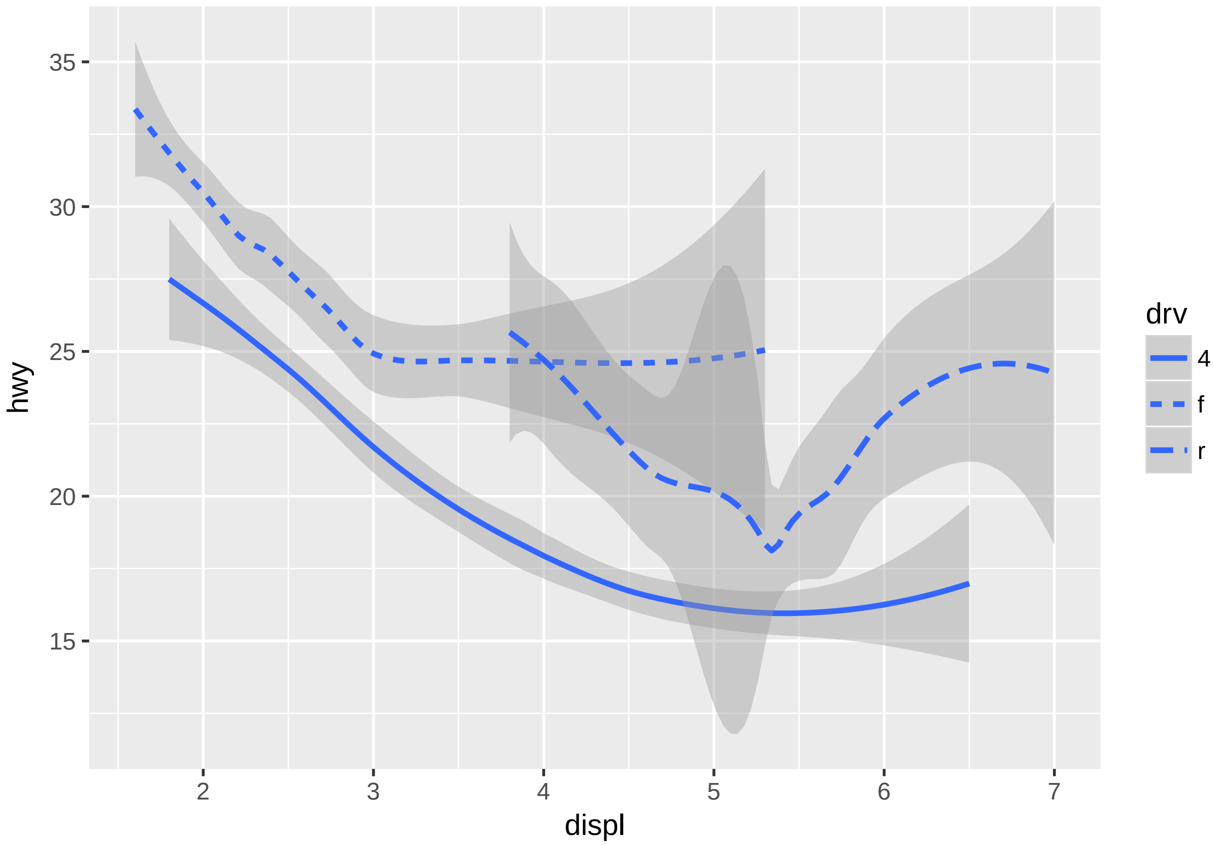
Like geom_points( ), you can add a mapping to geom_smooth( ) in addition to x and y. If you set an additional mapping, ggplot will draw multiple lines. Mapping aesthetics you can add include:

- linetype: each line will have different linetype such as solid, dashed, dotted,...
- color: each line will be in different color
- group: draw a line per unique value

First, let's assign variable *drv* to *linetype*.

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

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

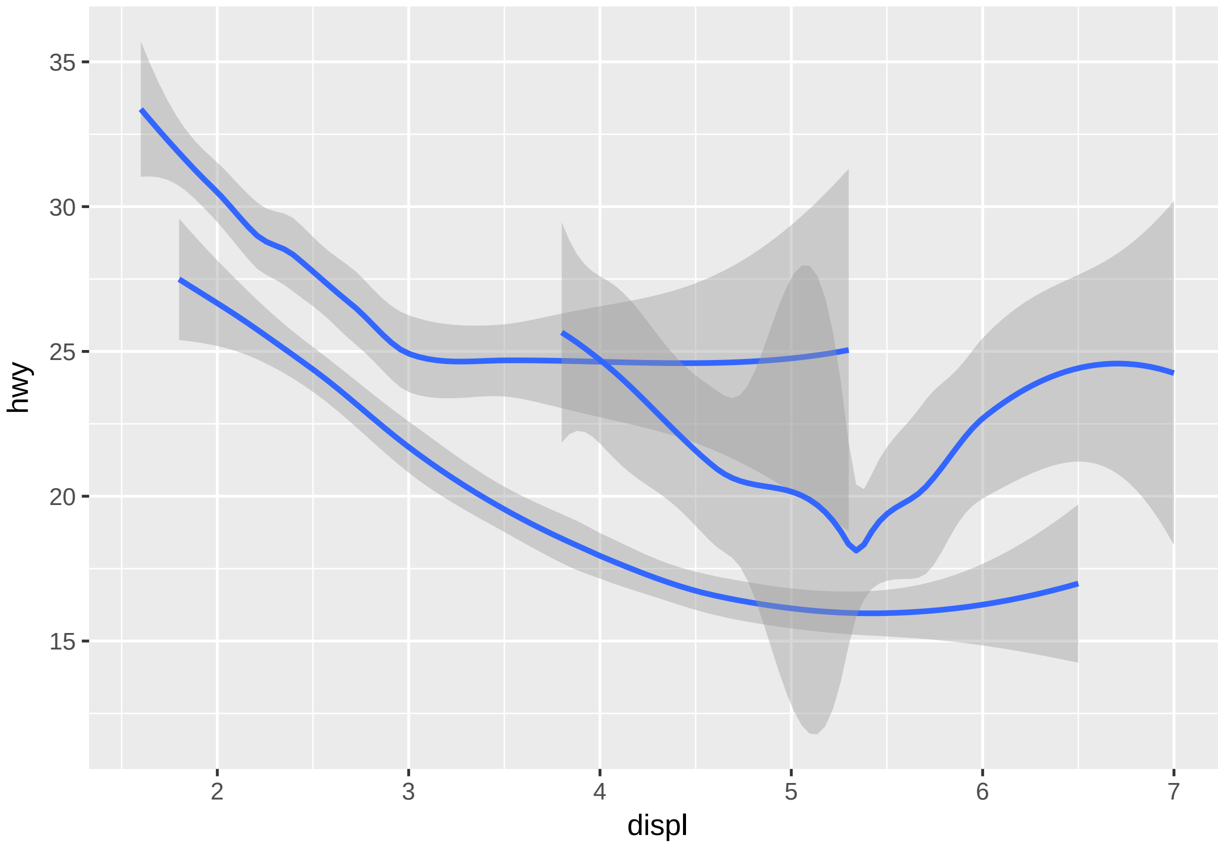


Here the data is split into 3 based on drv values. Each drv value has its own line with different linetype.

Next, change *linetype* to *group*

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

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

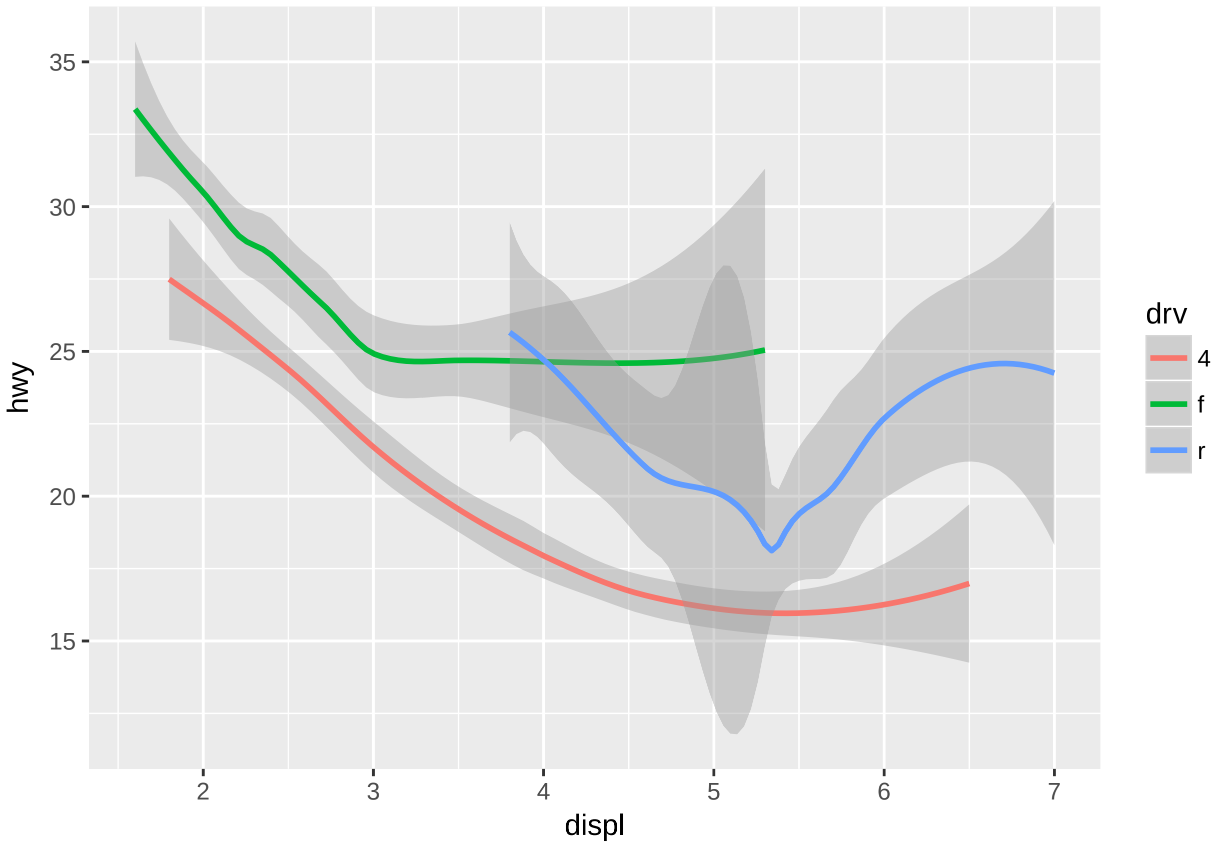


Notice the difference. Here we have the same 3 groups but the linetypes are the same.

We can also use *color*.

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

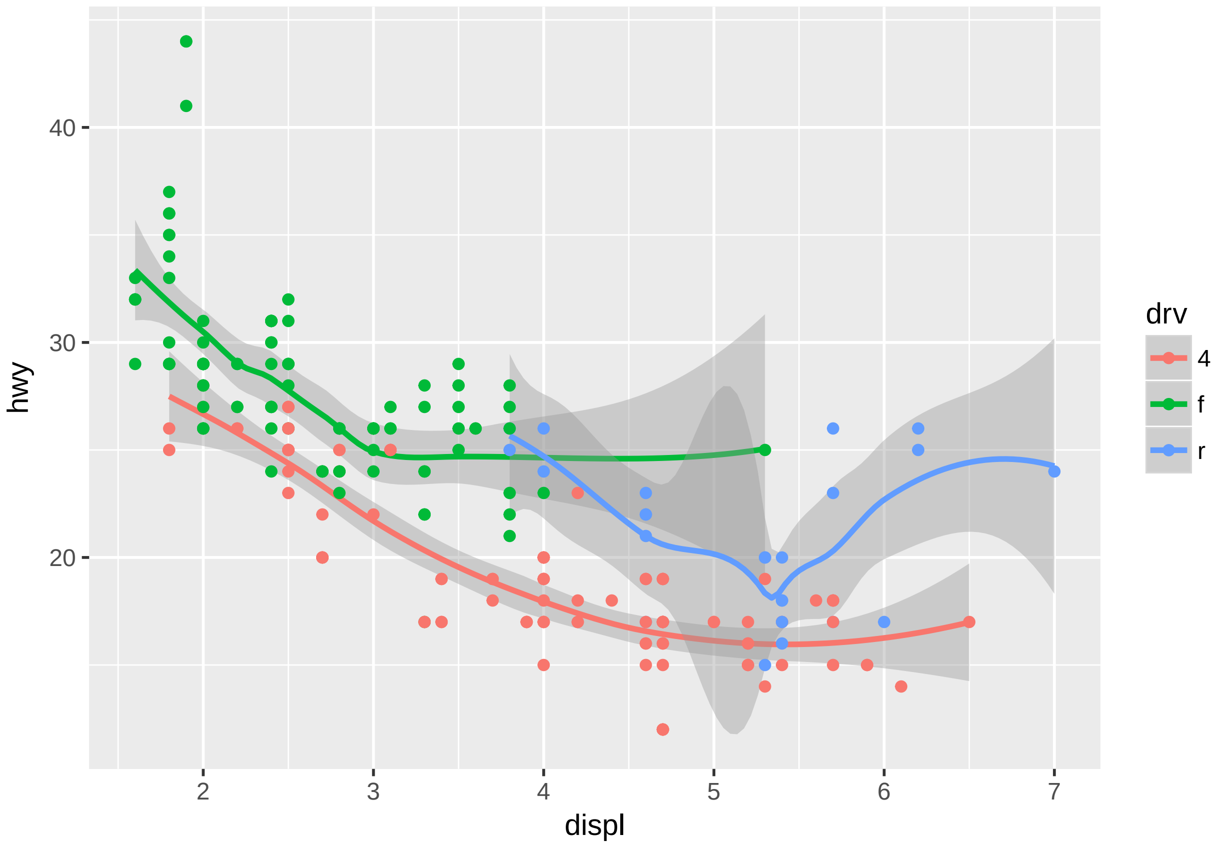
```
## `geom_smooth()` using method = 'loess'
```



You will see what the lines are doing more clearly if we plot both lines and data points, and set *color* based on drv.

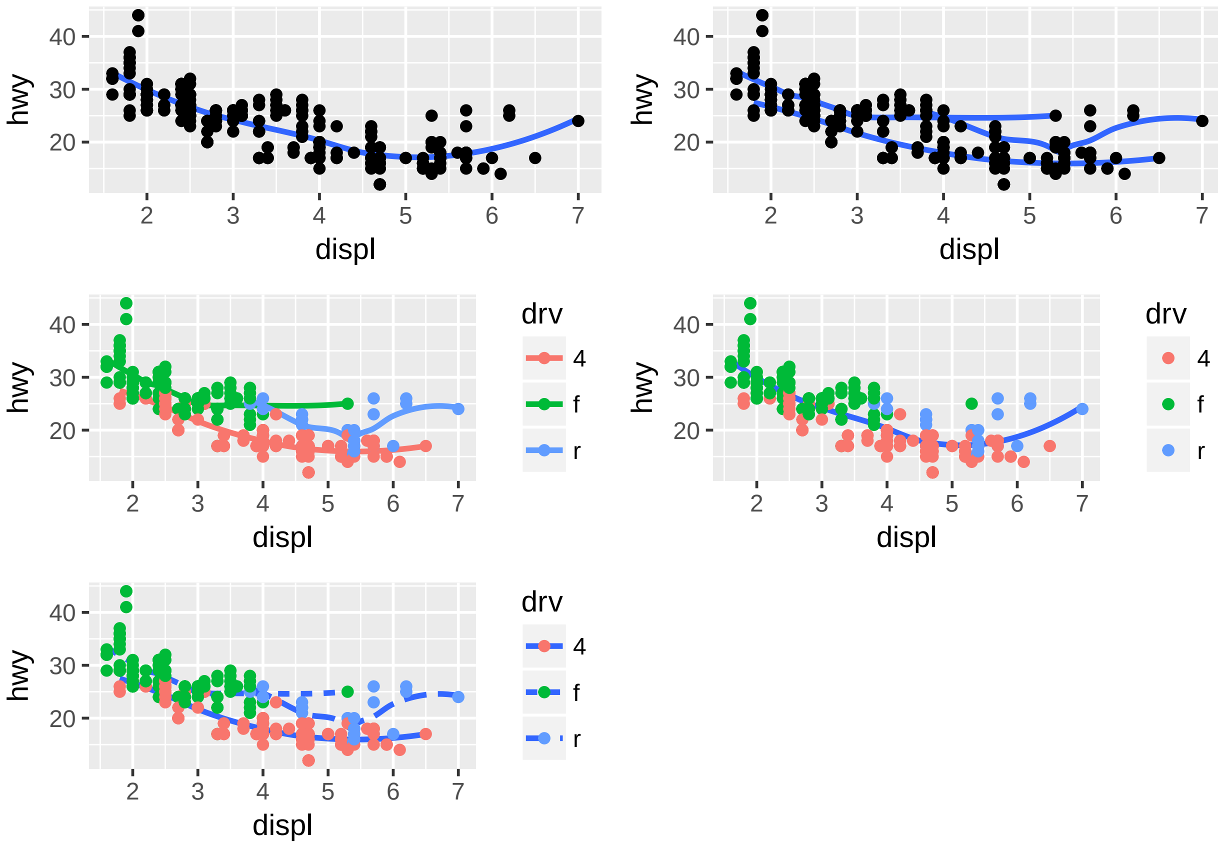
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = drv)) +
  geom_smooth() +
  geom_point()
```

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



Exercises:

Recreate following plots.




```r
p1 <- ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_smooth(se = FALSE) +
  geom_point()

p2 <- ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_smooth(mapping = aes(group = drv), se = FALSE) +
  geom_point()

p3 <- ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = drv)) +
  geom_smooth(se = FALSE) +
  geom_point()

p4 <- ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_smooth(se = FALSE) +
  geom_point(mapping = aes(color = drv))

p5 <- ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_smooth(mapping = aes(linetype = drv), se = FALSE) +
  geom_point(mapping = aes(color = drv))
```

Maps

With ggplot, you can plot pretty nice maps. For that, we will use these packages:

- maps: contains outlines of continents

Install and load it.

```
library(maps)
```

```
##
## Attaching package: 'maps'
## The following object is masked from 'package:purrr':
##
##     map
```

maps package provides map outlines and points for cities, counties, states, etc. We're going to use *usa* map data.

To create a data frame of map data, we use a ggplot function **map_data( )**.

```
usa <- map_data("usa")
```

Explore the usa data frame for a minute.

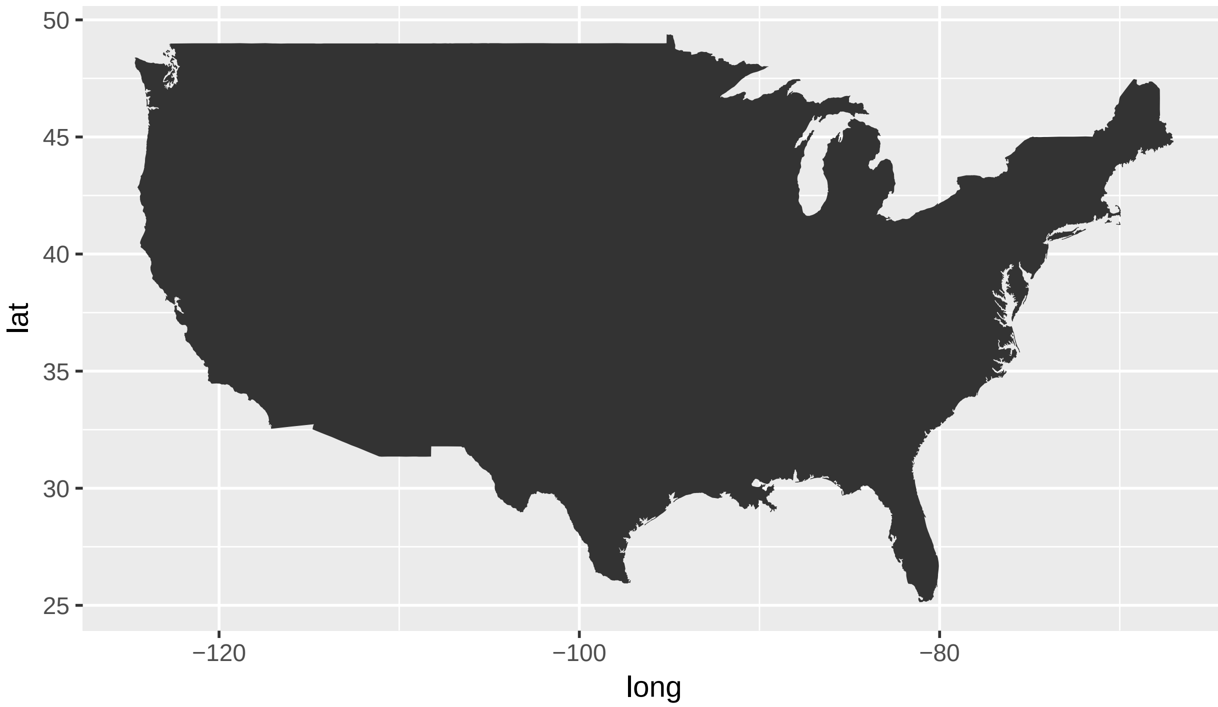
Map data frame structures

- long: longitude. If it's west of Greenwich, it's a negative value.
- lat: latitude
- order: This shows in which order ggplot should "connect the dots"
- region / subregion: what region or subregion a set of points surrounds
- group: controls whether adjacent points should be connected by lines (This is very important!)

Plot USA map

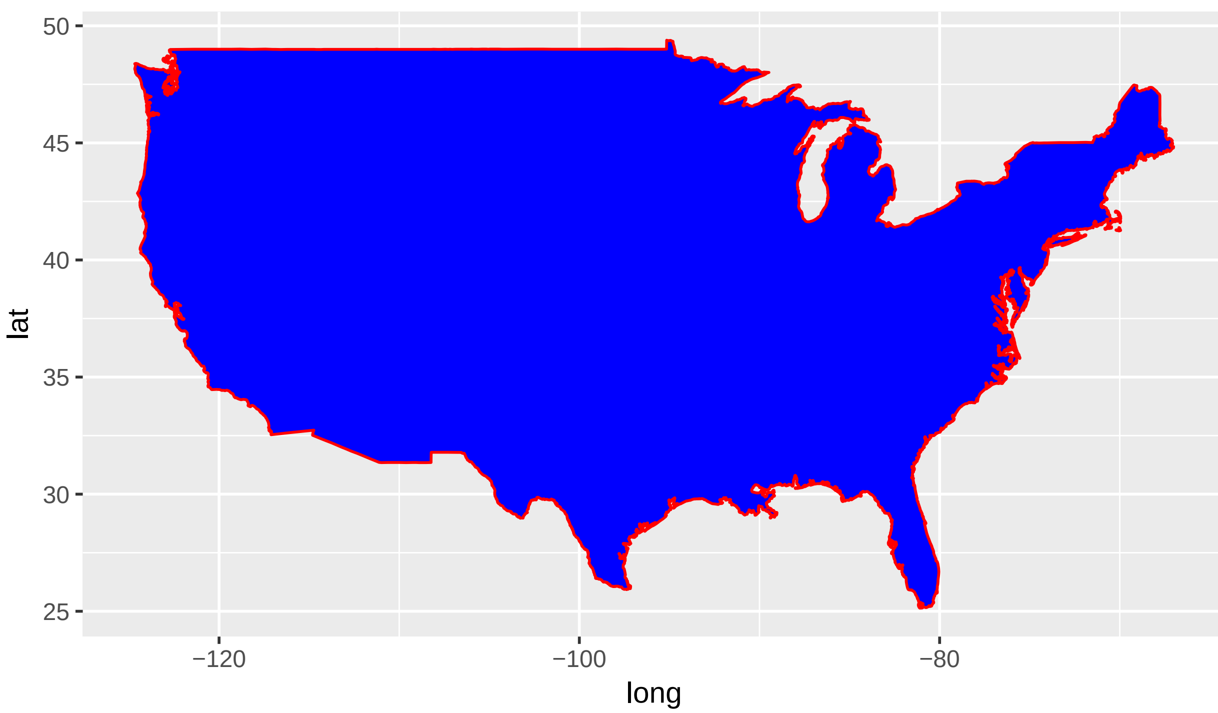
To plot a map, we use **geom_polygon( )**, which connects points and closes them up. Set x = long, y = lat, and group = group. Use coord_fixed(1.3) to fix the relationship between x and y units. In this case, we are setting one y unit is 1.3 times one x unit.

```
ggplot(data = usa) +
  geom_polygon(mapping = aes(x=long, y = lat, group = group)) +
  coord_fixed(1.3)
```



You can change the fill color and line color with fill and color arguments.

```
ggplot(data = usa) +
  geom_polygon(mapping = aes(x=long, y = lat, group = group), fill = "blue", color = "red") +
  coord_fixed(1.3)
```



Exercise: remove group = group to check why it's important

State maps

Now let's plot state boundaries

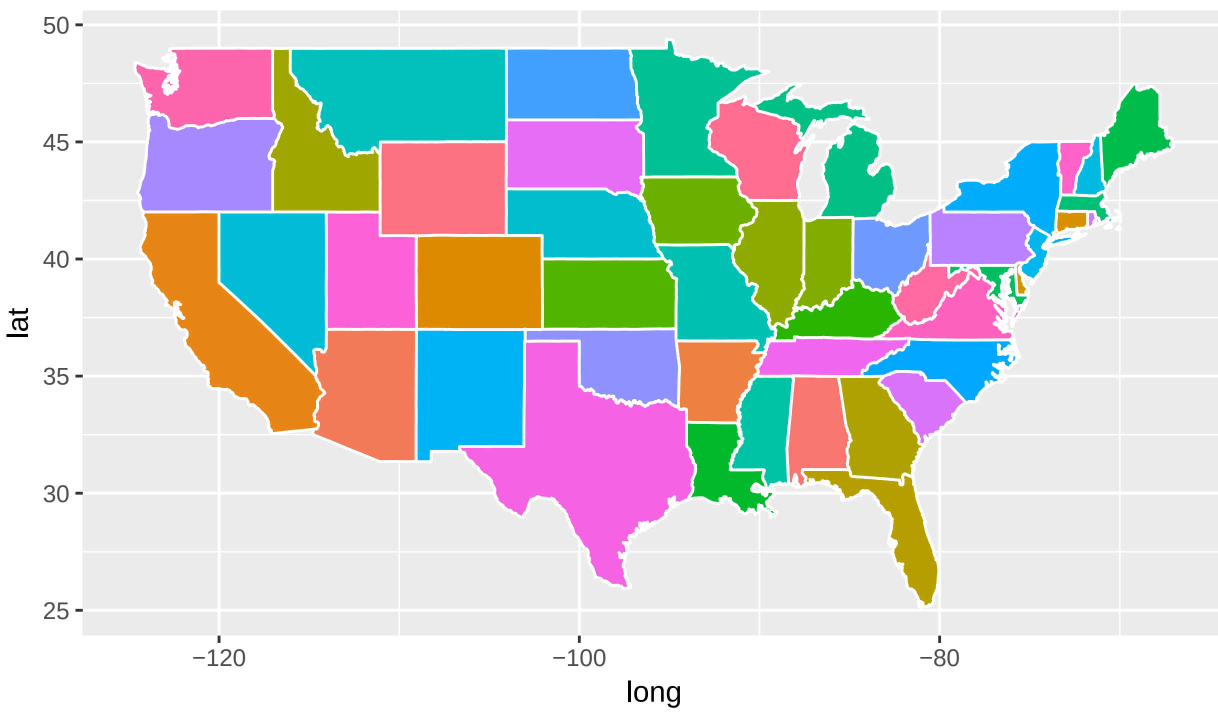
```
states <- map_data("state")

head(states)
```

```
##        long      lat group order  region subregion
## 1 -87.46201 30.38968     1     1 alabama      <NA>
## 2 -87.48493 30.37249     1     2 alabama      <NA>
## 3 -87.52503 30.37249     1     3 alabama      <NA>
## 4 -87.53076 30.33239     1     4 alabama      <NA>
## 5 -87.57087 30.32665     1     5 alabama      <NA>
## 6 -87.58806 30.32665     1     6 alabama      <NA>
```

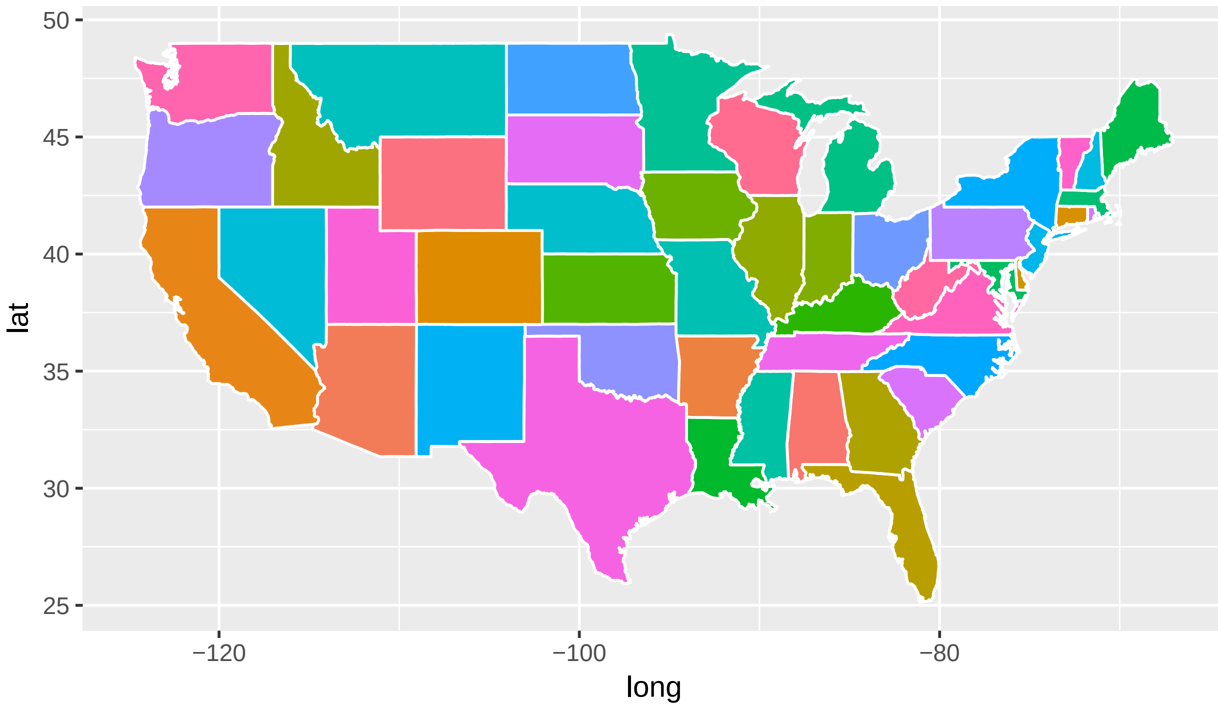
Do a quick data exploration.

Exercise: Can you recreate this plot? (3 min)

To remove the legend, use **guides(fill=FALSE)**



```
ggplot(data = states) +
  geom_polygon(aes(x = long, y = lat, fill = region, group = group), color = "white")  +
  guides(fill=FALSE) +
  coord_fixed(1.3)
```



Plot just NY state

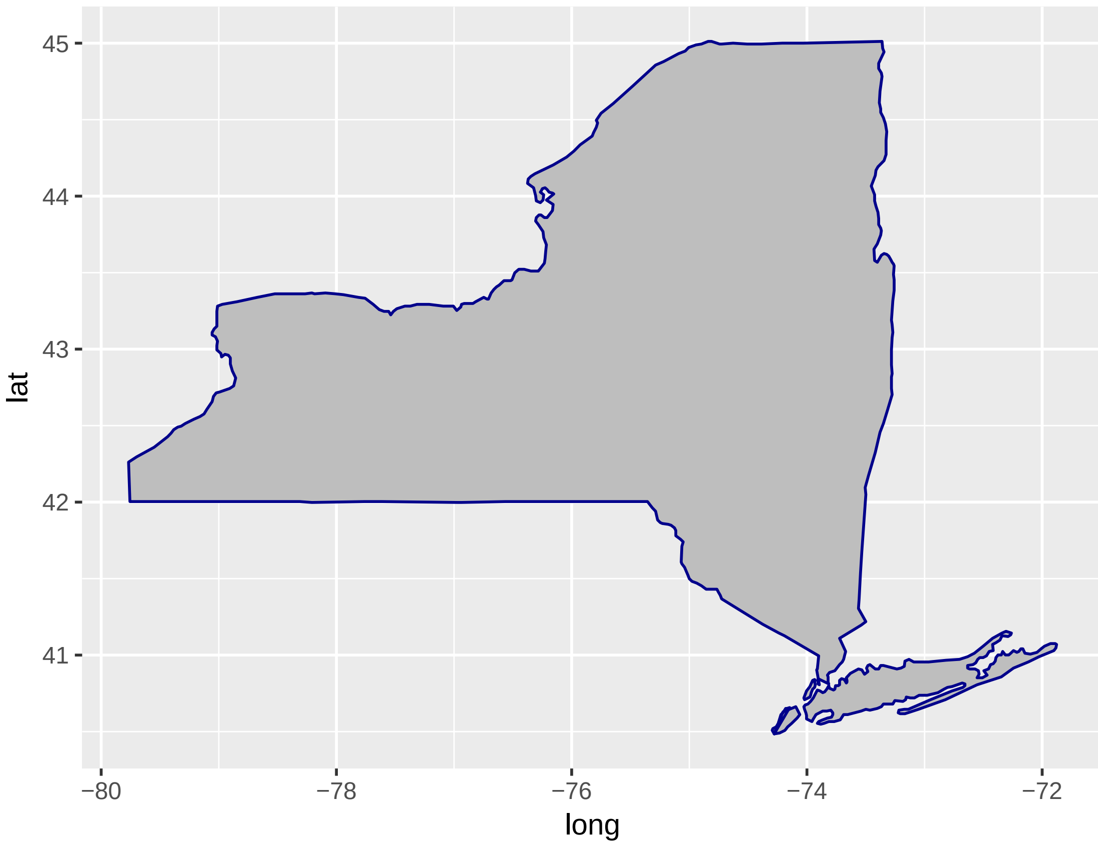
To subset a data frame, use a function **subset( )**

```
ny <- subset(states, region == "new york")

head(ny)
```

```
##           long      lat group order   region subregion
## 9050 -73.92874 40.80605    34  9050 new york manhattan
## 9051 -73.93448 40.78886    34  9051 new york manhattan
## 9052 -73.95166 40.77741    34  9052 new york manhattan
## 9053 -73.96312 40.75449    34  9053 new york manhattan
## 9054 -73.96885 40.73730    34  9054 new york manhattan
## 9055 -73.97458 40.72584    34  9055 new york manhattan
```

Let's plot the NY state map.

```
nystate <- ggplot(data = ny) +
  geom_polygon(mapping = aes(x=long, y = lat, group = group), color = "dark blue", fill = "gray") +
  coord_fixed(1.3)

nystate
```



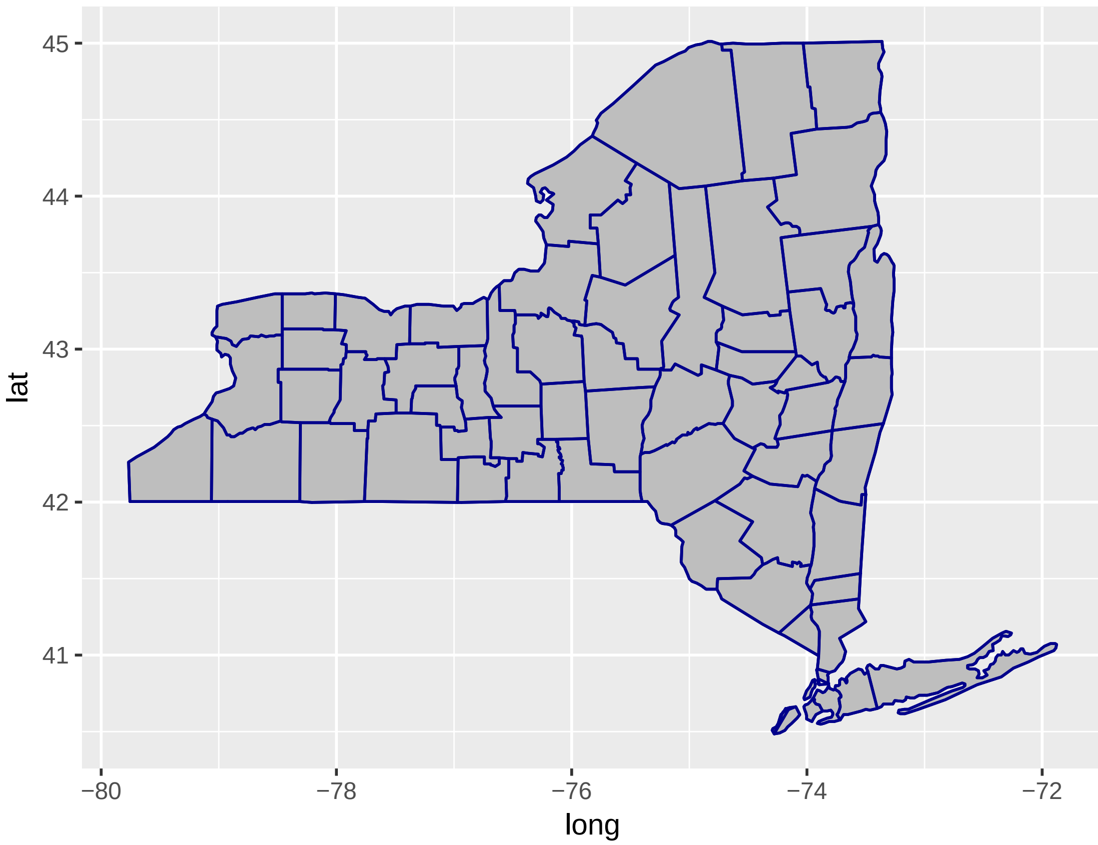
Now let's get county lines in NY

```
counties <- map_data("county")
ny_county <- subset(counties, region == "new york")
```

Check all the NY state counties are there.

Let's plot the county boundaries in dark blue on top of the previous map.

```
nycounties <- nystate +
  geom_polygon(data = ny_county, mapping = aes(x = long, y = lat, group = group), fill = NA, color = "da

nycounties
```



Now let's plot cities on the map. US city data is in us.cities.

```r
data(us.cities)

head(us.cities)
```

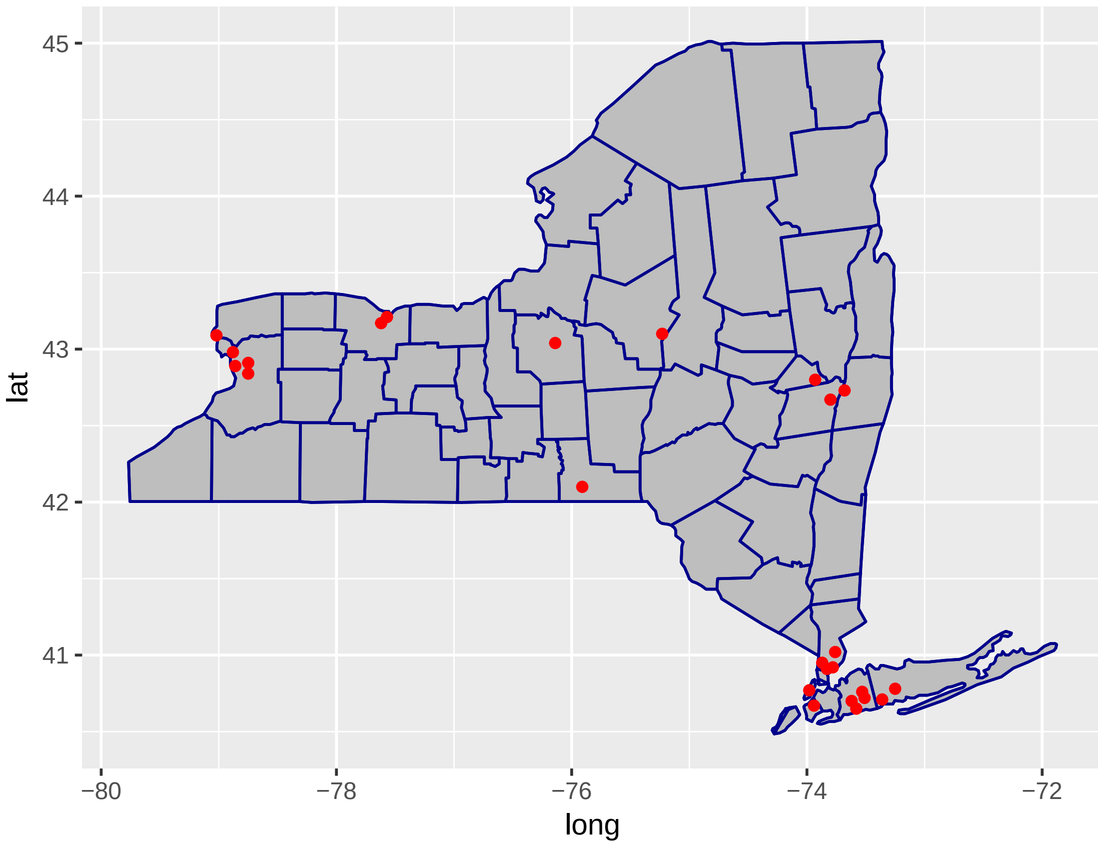
```
##          name country.etc    pop   lat    long capital
## 1 Abilene TX          TX 113888 32.45  -99.74       0
## 2   Akron OH          OH 206634 41.08  -81.52       0
## 3 Alameda CA          CA  70069 37.77 -122.26       0
## 4  Albany GA          GA  75510 31.58  -84.18       0
## 5  Albany NY          NY  93576 42.67  -73.80       2
## 6  Albany OR          OR  45535 44.62 -123.09       0
```

We just want NY state data.

```r
nycities <- subset(us.cities, country.etc == "NY")
```

Now plot this data on the NY county map.

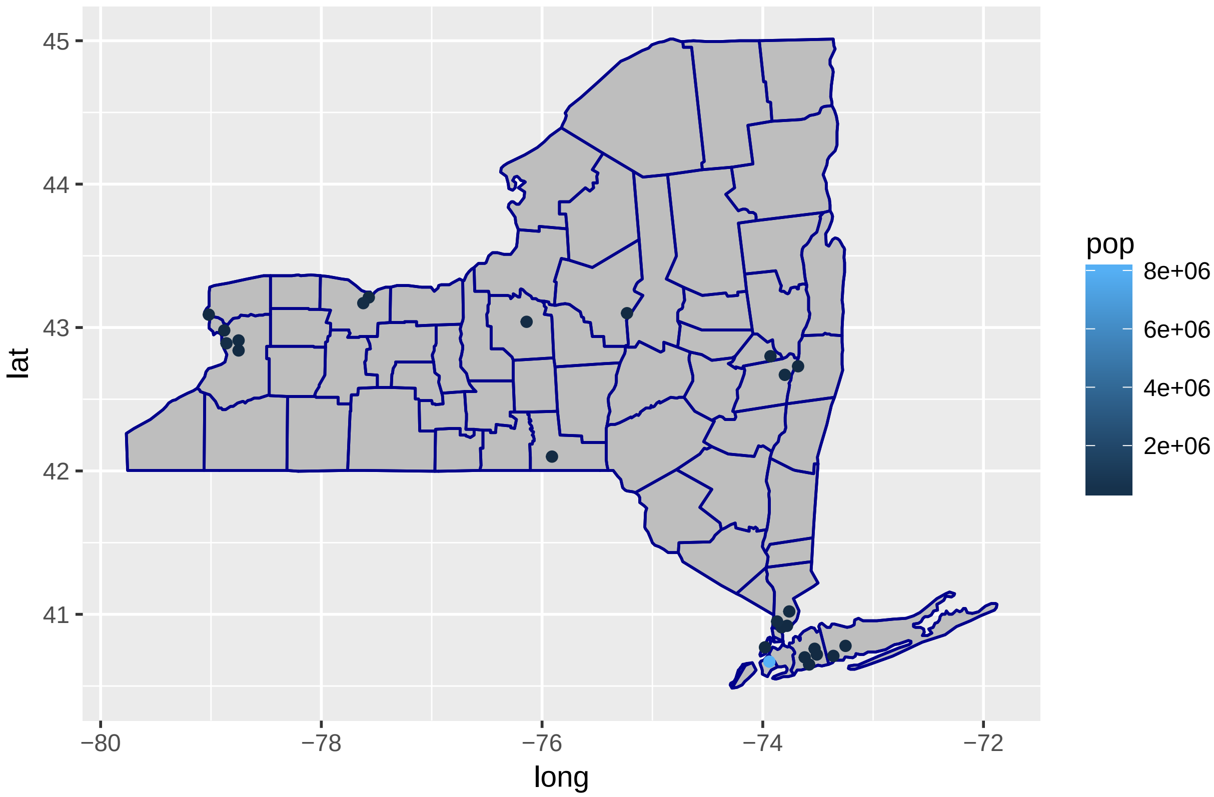
```r
nycounties +
  geom_point(data = nycities, mapping = aes(x = long, y = lat), color = "red")
```



This map has information about where cities are in NY state. Can we make it more informative?

How about changing colors with population?

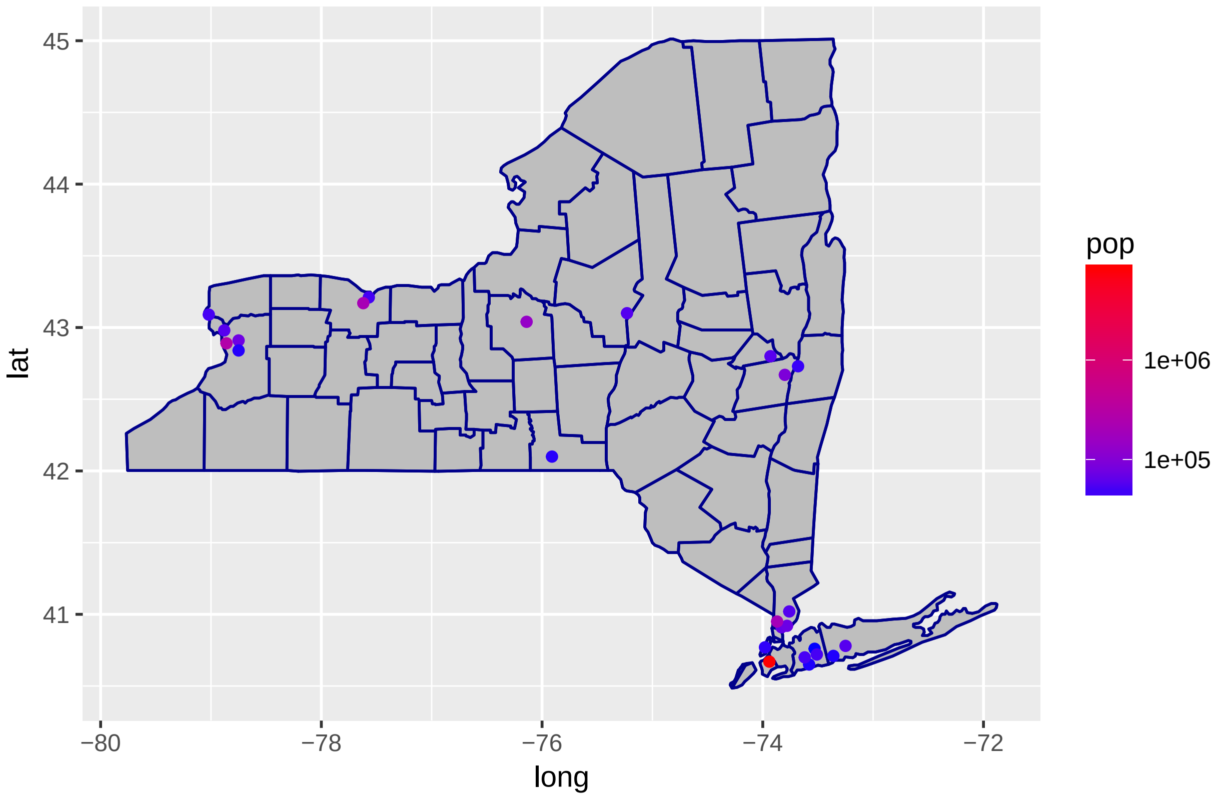
```
nycounties +
  geom_point(data = nycities, mapping = aes(x = long, y = lat, color = pop))
```



This one is not very informative because NYC is so much more populated than other cities and you can't see differences between them.

We can fix it by plotting log-base-10 of population. Also let's change color scheme.

```
nycityplot <- nycounties +
  geom_point(data = nycities, mapping = aes(x = long, y = lat, color = pop)) +
  scale_colour_gradient(high = "red", low = "blue", trans = "log10")

nycityplot
```



Finally, if you just want the map without axes and grids, you can add **theme_void**( )

```
nycityplot +
  theme_void()
```