DATA 100,Week 2 (A)

In particular, library(tidyverse) should appear in the beginning of every .qmd (or .rmd)file we have from now on

```
library(tidyverse)
library(ggthemes)
```

DATA 100,Week 2 (A)

More ways to read the data directly

Calling a dataframe directly by its name normally shows only a portion of it

- Both columns or rows may be truncated in the display

mpg

```
#> # A tibble: 234 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    2008     4 manual(m6) f        20    31 p
#> 4 audi         a4      2    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
#> # i 228 more rows
#> # i 1 more variable: class <chr>
```

DATA 100,Week 2 (A)

Use glimpse or view to see more of it

• glimpse turns the table around, making sure that all columns show up

• view generates a separate output tab, showing the **full dataframe** in a large table

DATA 100,Week 2 (A)

glimpse(mpg)

```
Rows: 234
Columns: 11
$ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi", "audi", "audi", "audi", "audi", "…
$ model        <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro", "a4 quattro", "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, 2.8, 2.8, 3.1, 3.1, 2.8, 3…
$ year         <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, 2008, 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, 8, 8, 8, 8, 8, 8, 8, 8, …
$ trans        <chr> "auto(l5)", "manual(m5)", "manual(m6)", "auto(av)", "auto(l5)", "manual(m5)", "au…
$ drv          <chr> "f", "f", "f", "f", "f", "f", "f", "4", "4", "4", "4", "4", "4", "4", "4", "4", "…
$ cty          <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17, 15, 15, 17, 16, 14, 11, 1…
$ hwy          <int> 29, 29, 31, 30, 26, 26, 27, 26, 25, 28, 27, 25, 25, 25, 25, 24, 25, 23, 20, 15, 2…
$ fl           <chr> "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "…
$ class        <chr> "compact", "compact", "compact", "compact", "compact", "compact", "compact", "com…
```

view(mpg)

DATA 100,Week 2 (A)

More scatter plots

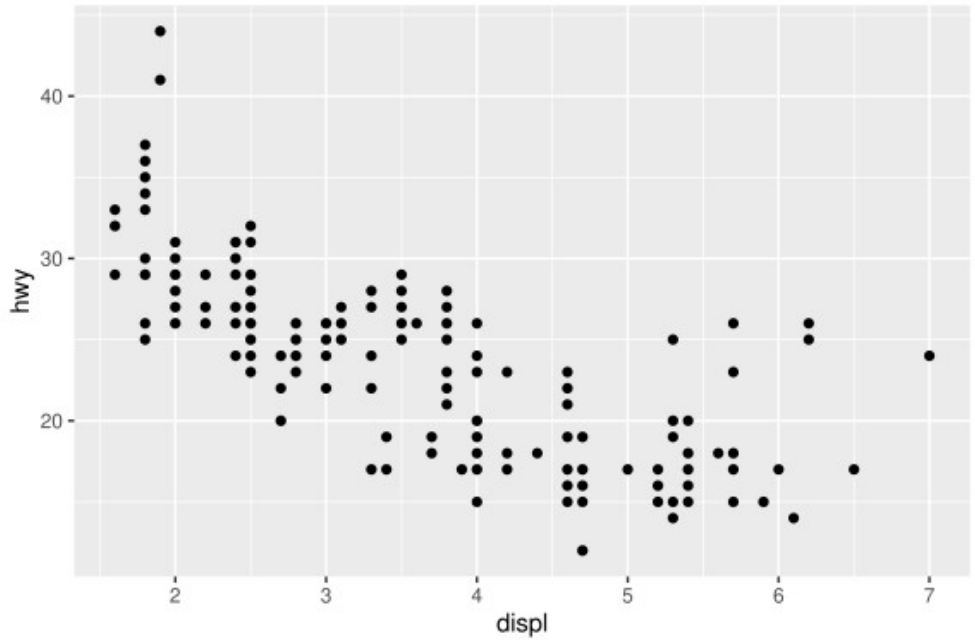
General structure of ggplot calls:

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

The same scatter plot of hwy (highway mileage) v.s. displ (engine size)

DATA 100,Week 2 (A)

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



DATA 100,Week 2 (A)

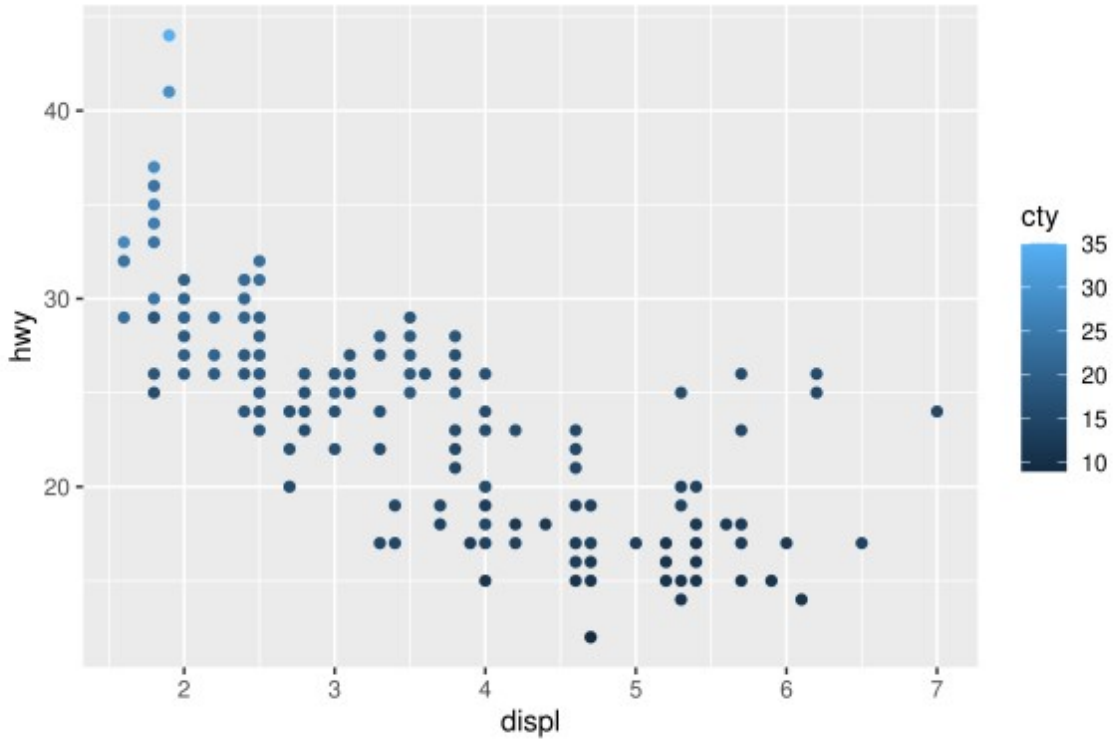
Question: How to see the cty (in-town mileage)?

• Decorate the points differently according to its cty value, using one of the aesthetics.

– Try color, shape, alpha (shading), size

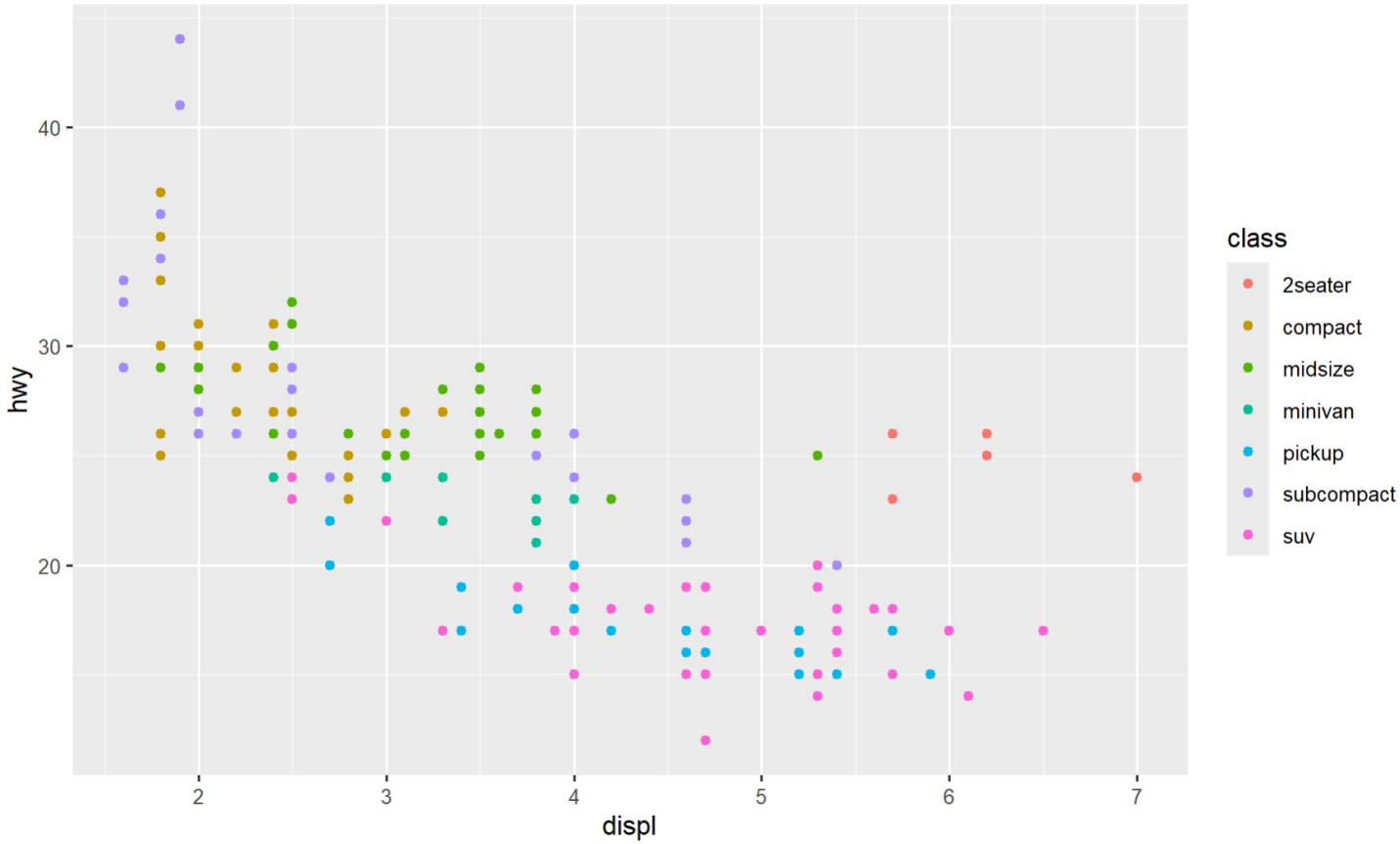
DATA 100,Week 2 (A)

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



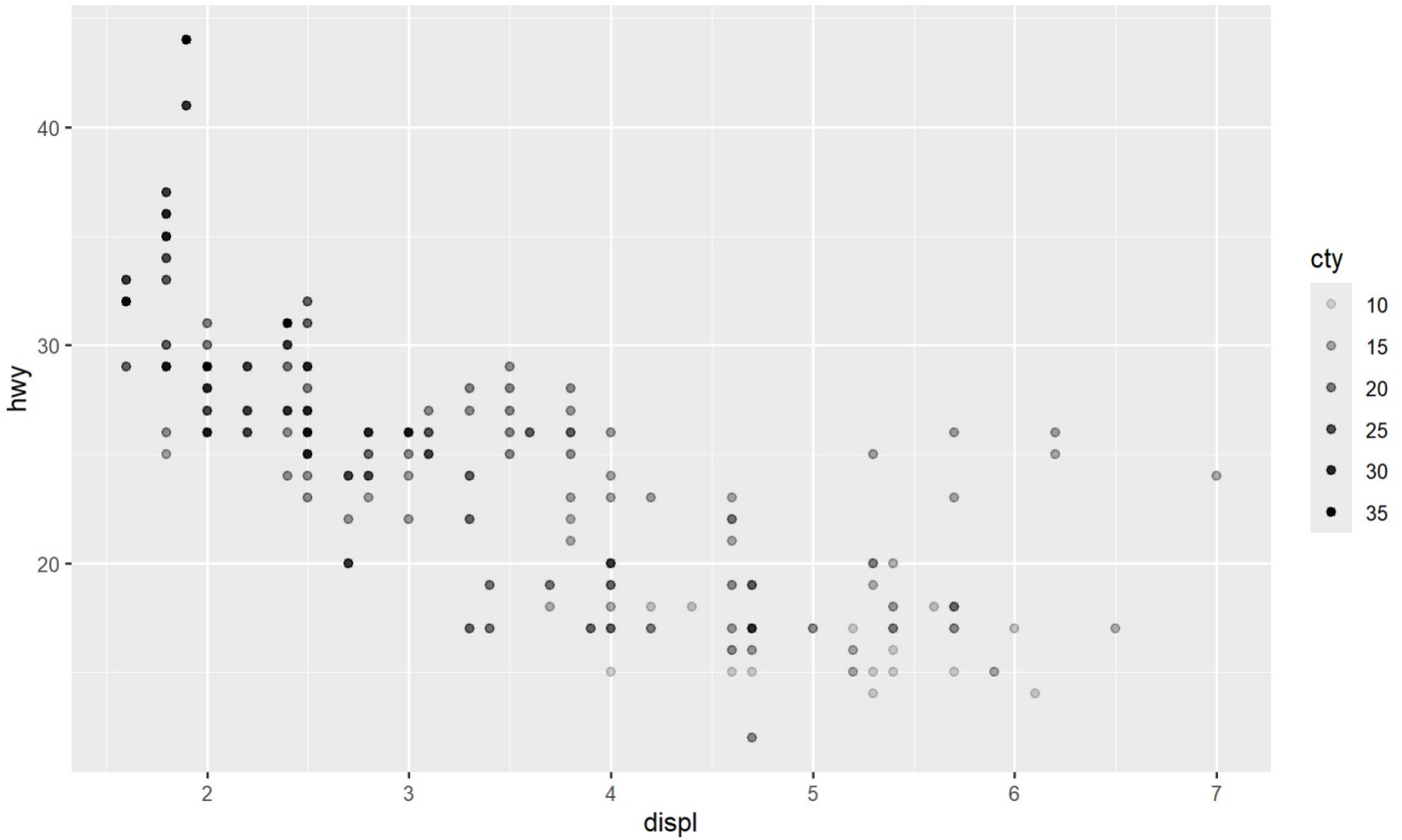
DATA 100,Week 2 (A)

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



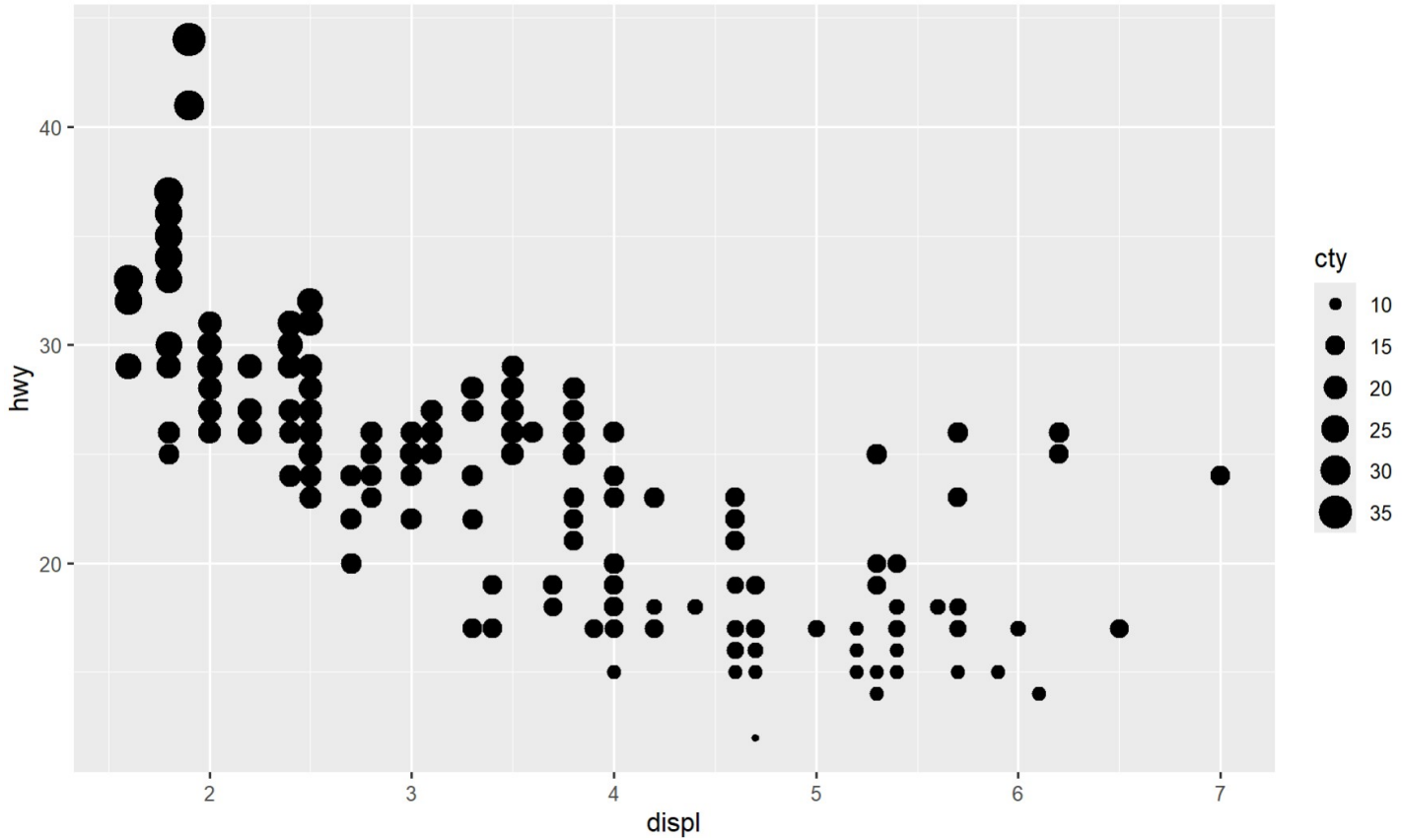
DATA 100,Week 2 (A)

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



DATA 100,Week 2 (A)

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

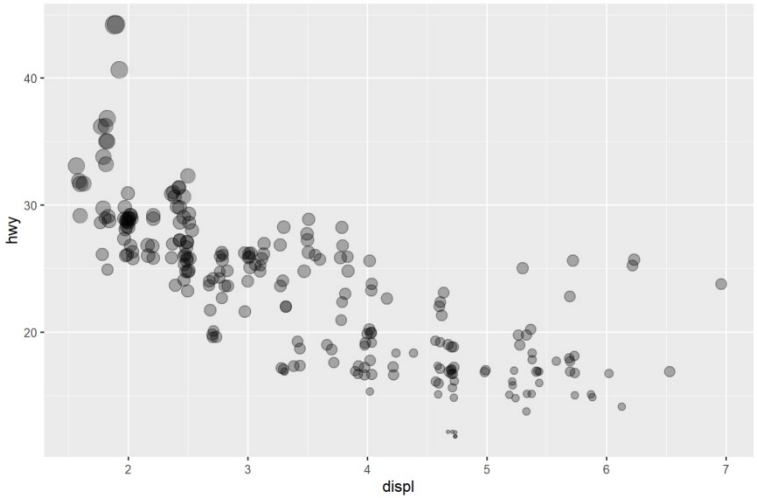


DATA 100,Week 2 (A)

The geom_jitter is a convenient shortcut for geom_point(position = "jitter").

It adds a small amount of random variation to the location of each point, and is a useful way of handling overplotting caused by discreteness in smaller datasets.

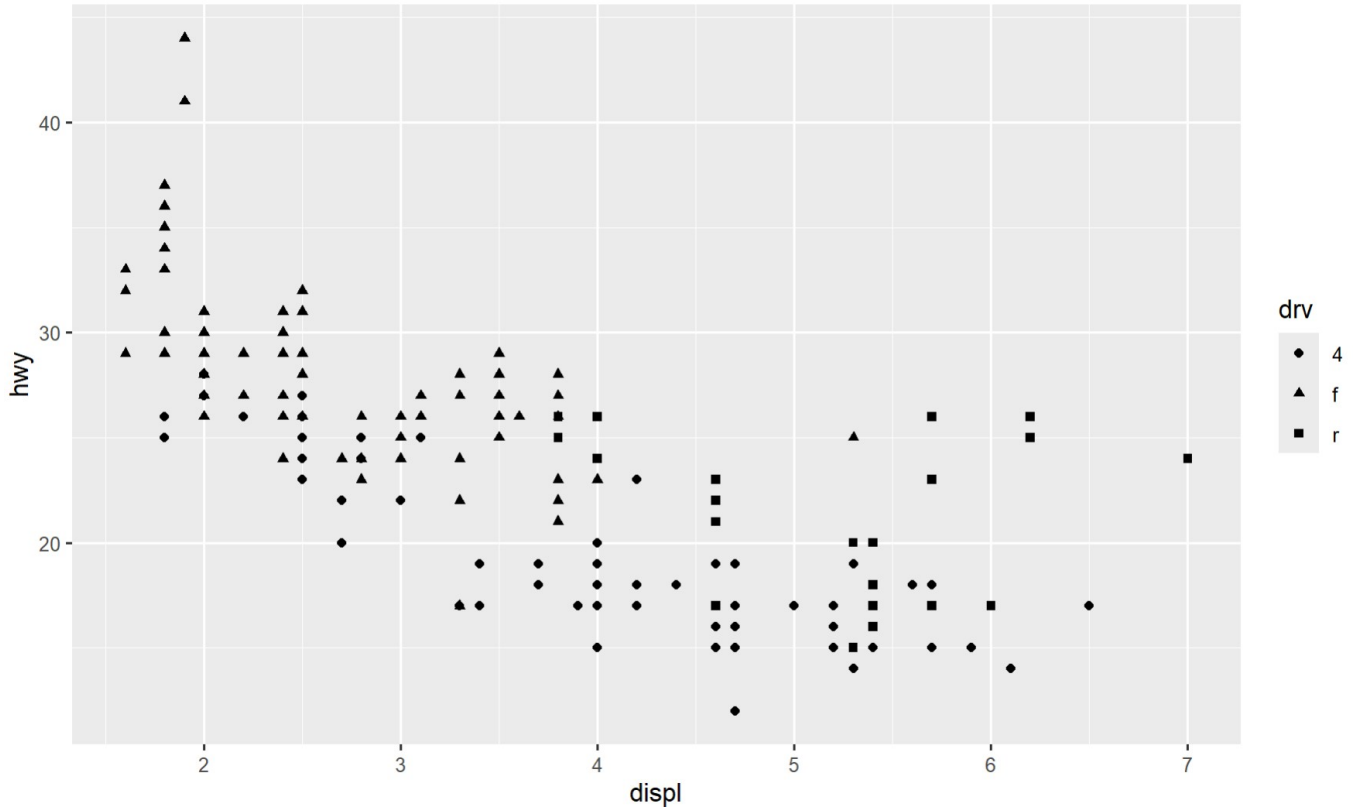
ggplot(data = mpg) +geom_jitter(mapping = aes(x=displ, y = hwy, size = cty), alpha=0.3)



DATA 100,Week 2 (A)

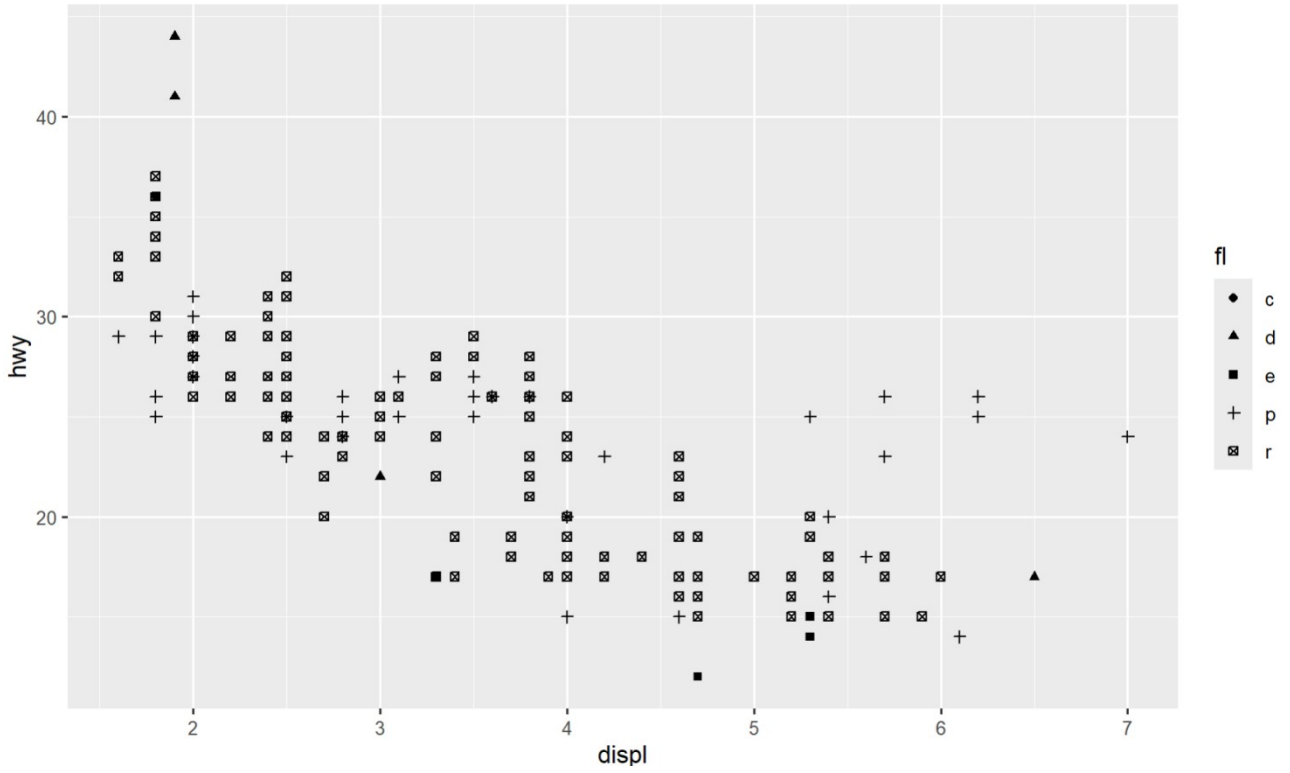
Try out shape for the drv (drive train)

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



DATA 100,Week 2 (A)

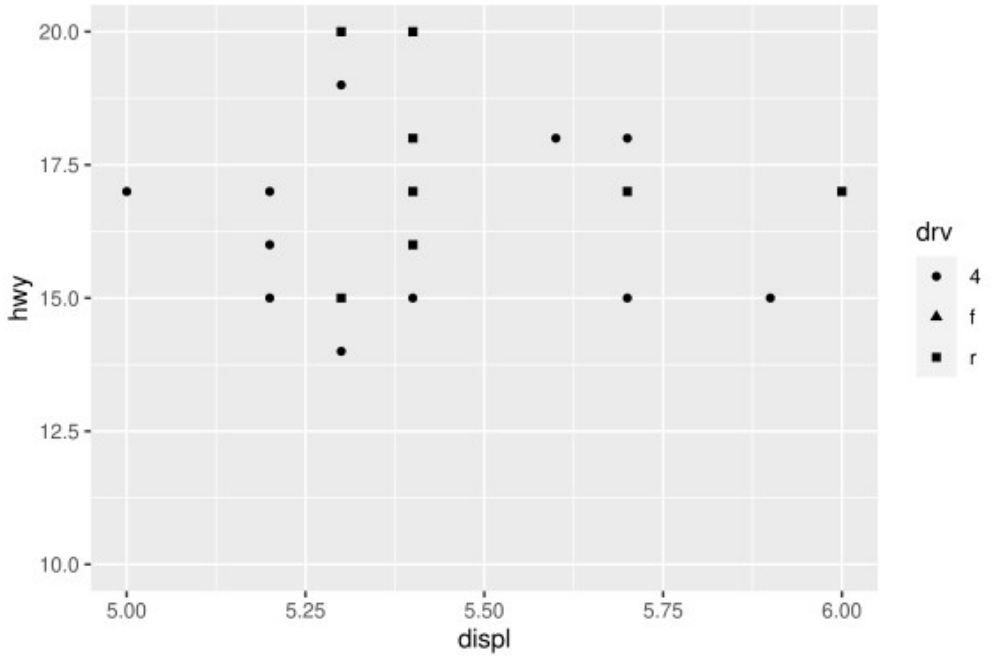
ggplot(data = mpg) +geom_point(mapping = aes(x=displ, y = hwy, shape = fl))



DATA 100,Week 2 (A)

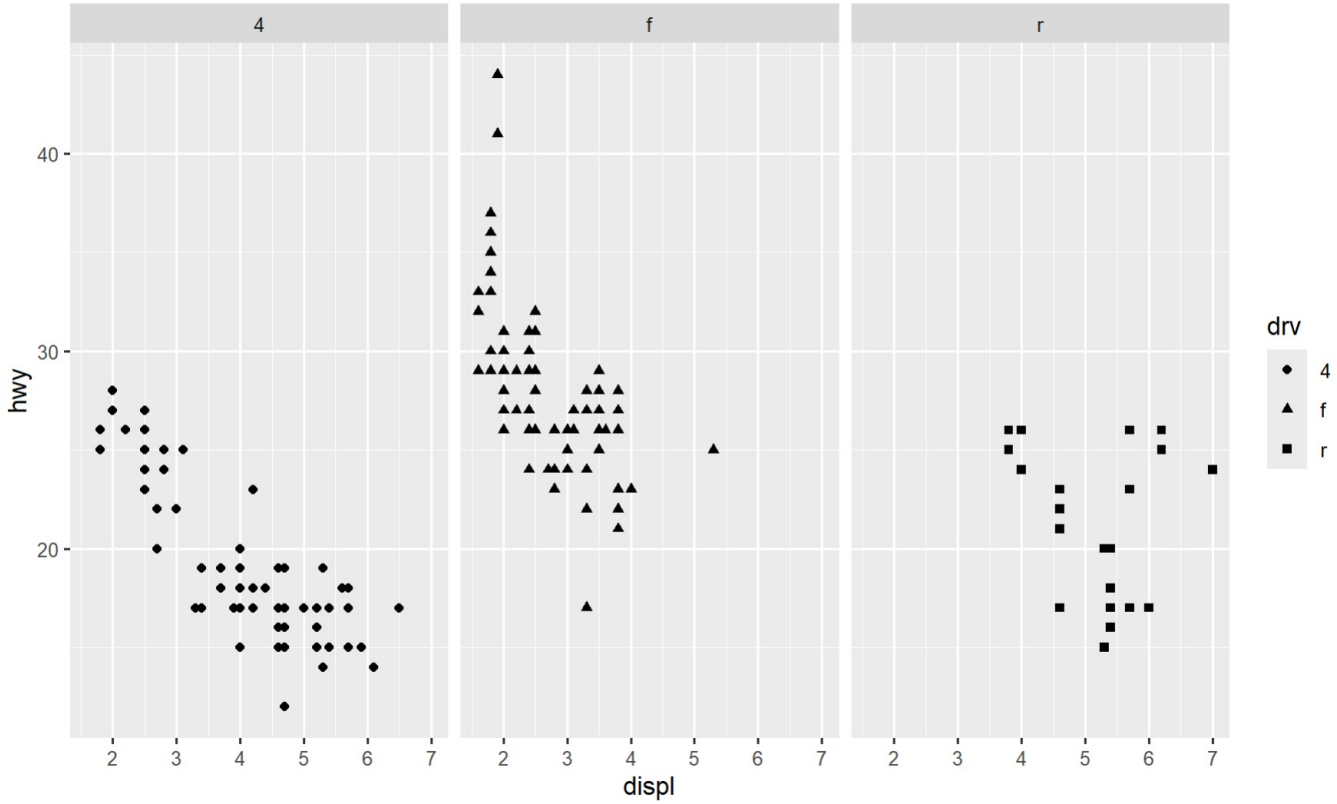
use xlim() and ylim() to scale the part of plots for more details

```
ggplot(data = mpg) +
geom_point(mapping = aes(x=displ, y = hwy, shape = drv)) + ylim(10,20) +
xlim(5,6)
```



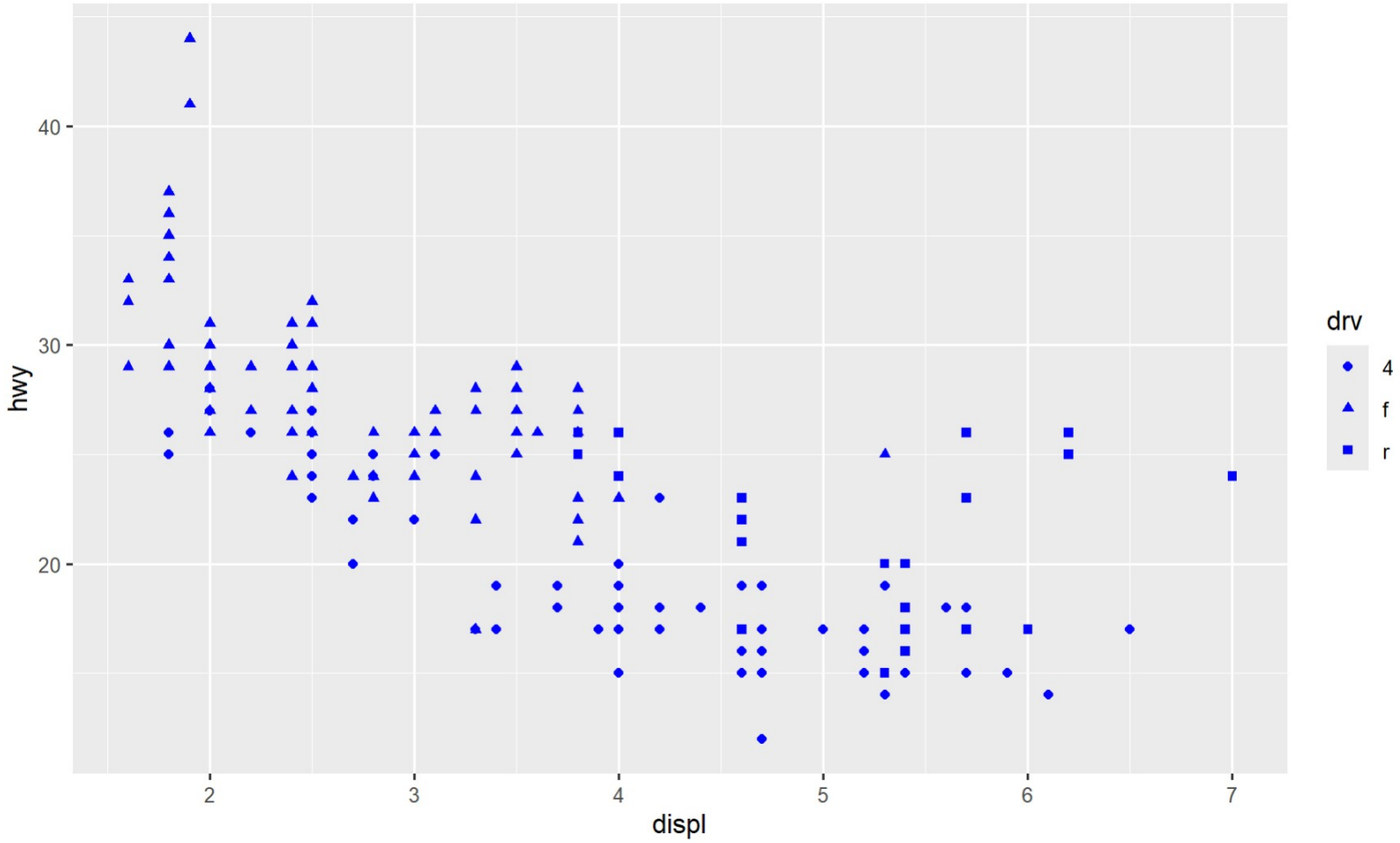
DATA 100,Week 2 (A)

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



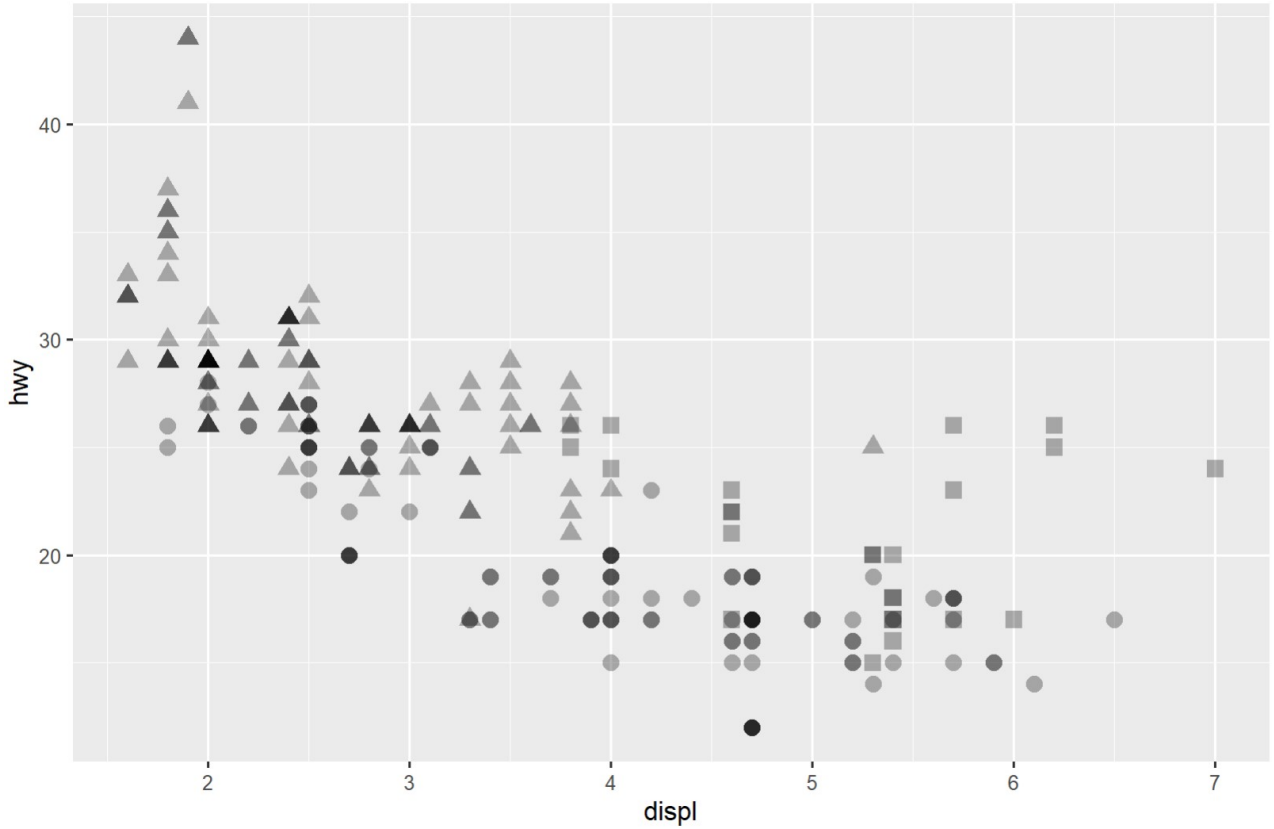
DATA 100,Week 2 (A)

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



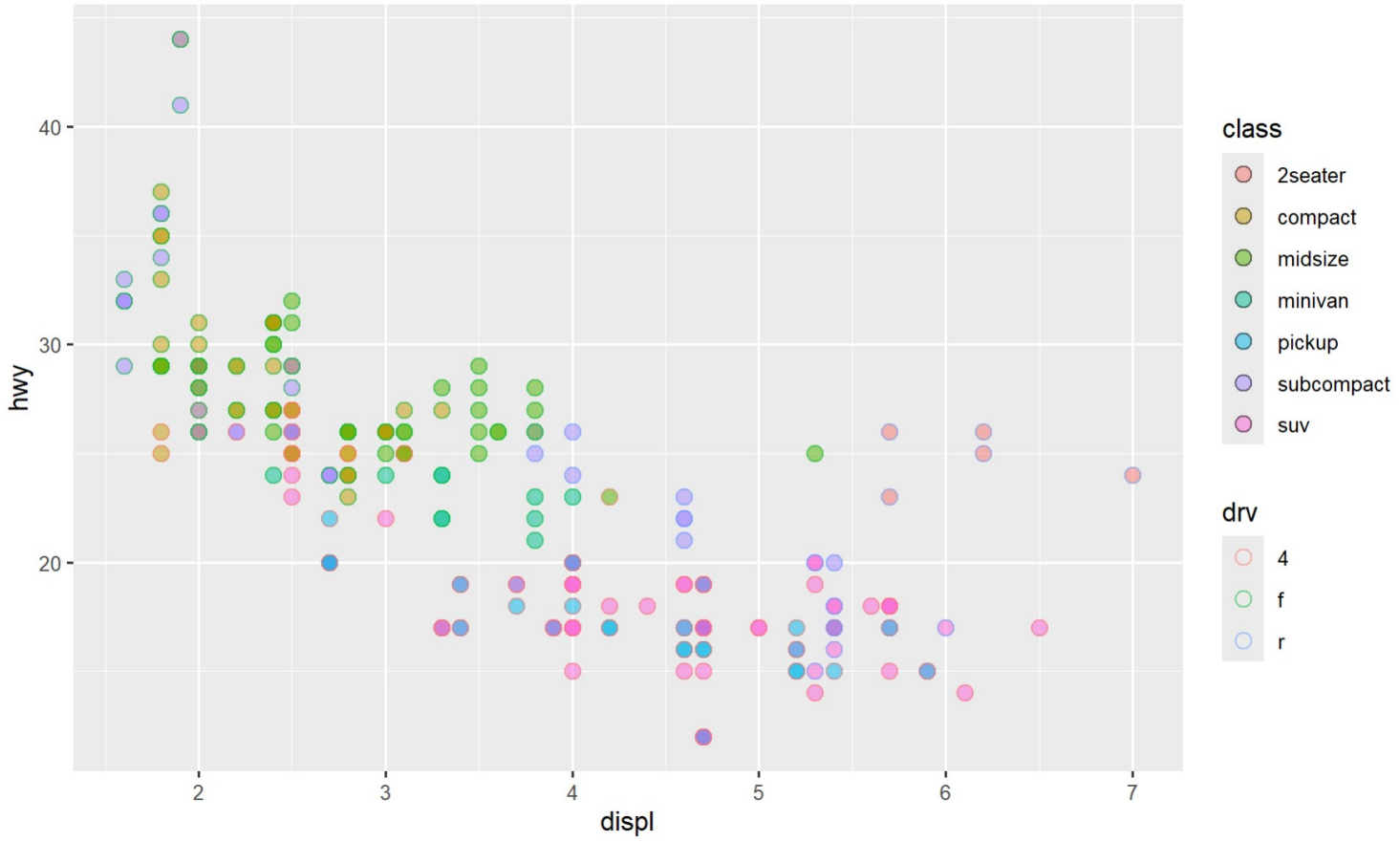
DATA 100,Week 2 (A)

ggplot(data = mpg) +geom_point(mapping = aes(x=displ, y = hwy, shape = drv), size=3, alpha=0.3)



DATA 100,Week 2 (A)

ggplot(data = mpg) +geom_point(mapping = aes(x=displ, y = hwy, color = drv, fill=class), size=3, alpha=0.5, shape=21)



DATA 100,Week 2 (A)

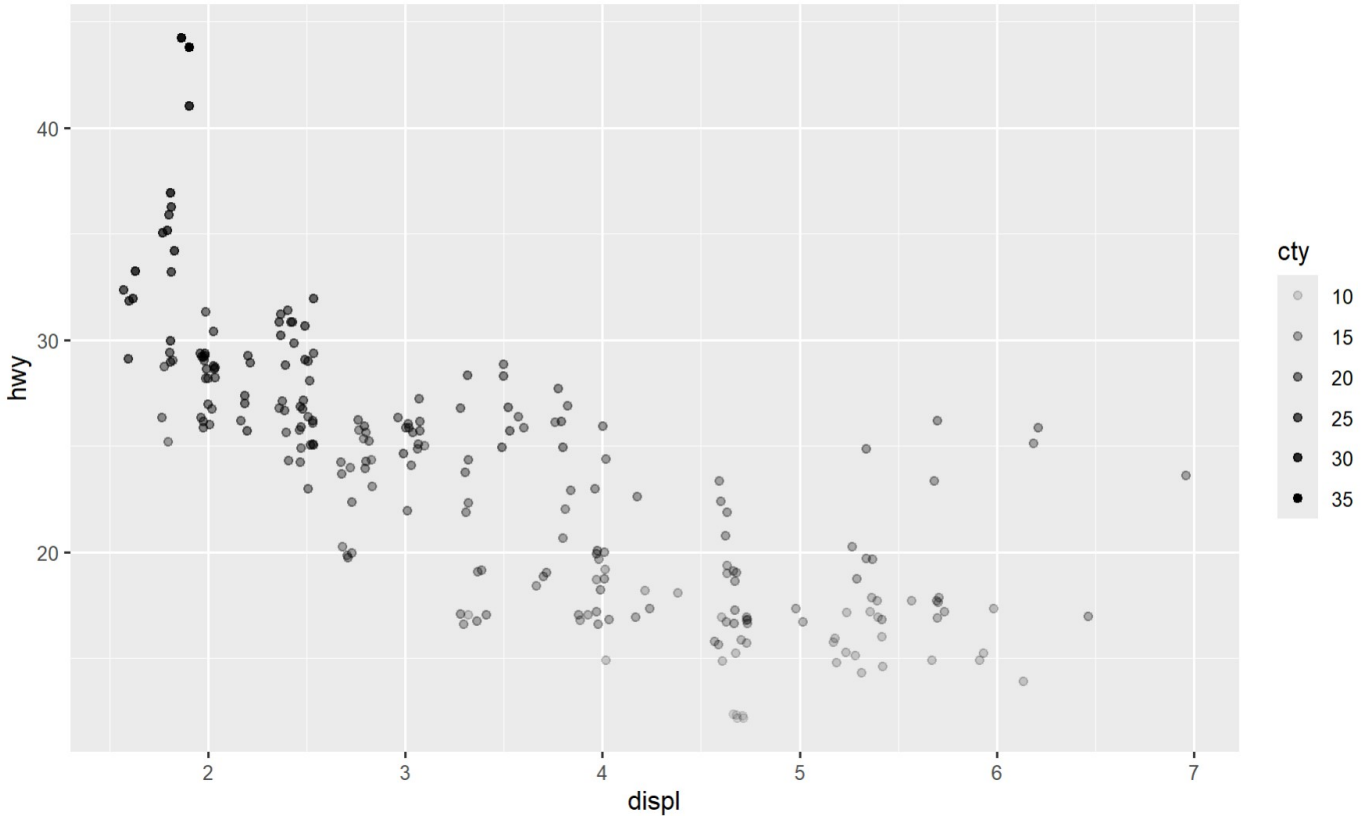
Question: What about overlapping points (overplot)?

- jitter them, with the position parameter
- or use geom_jitter directly

Note: Do not over-use geom_jitter unless it is necessary

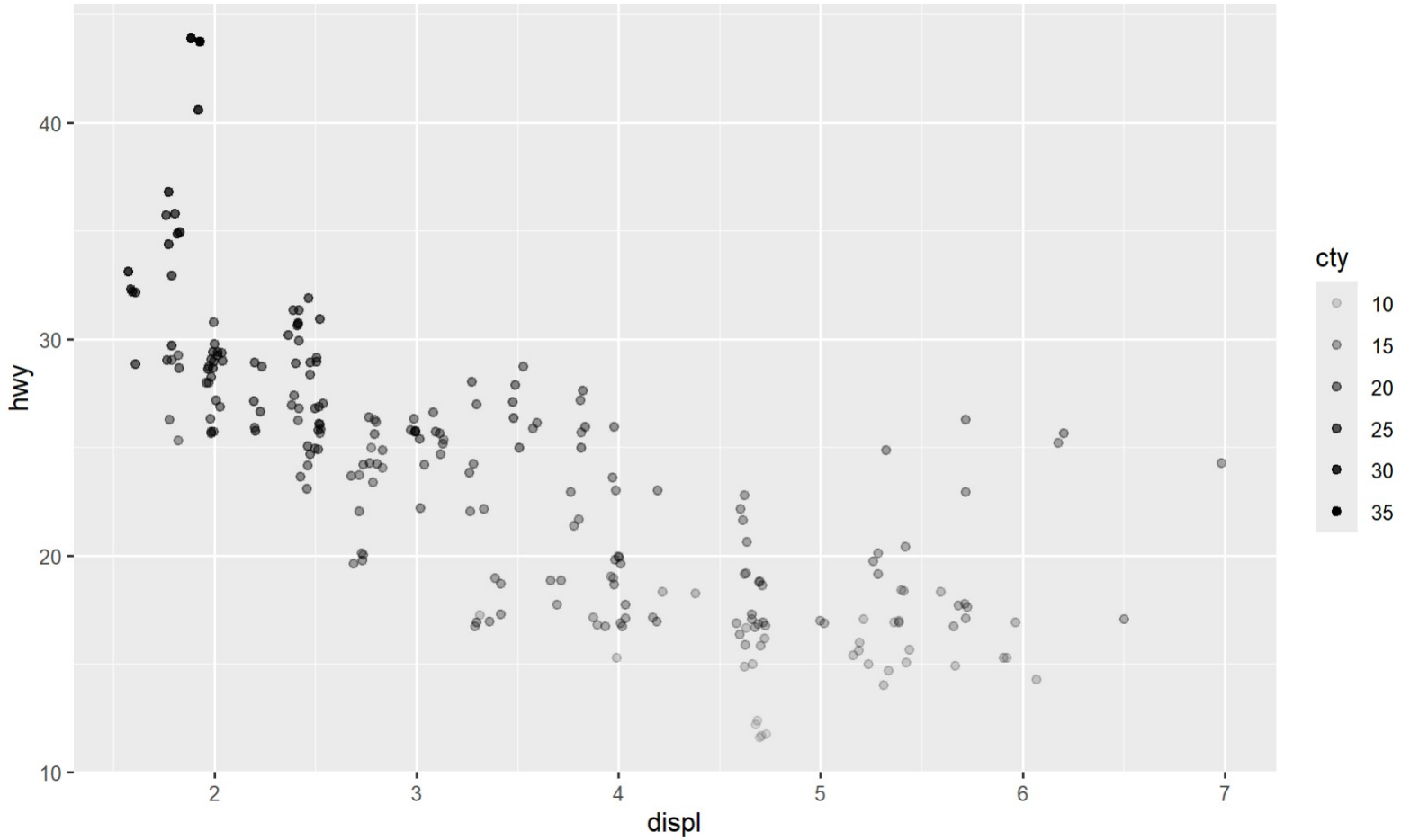
DATA 100,Week 2 (A)

```
ggplot(data = mpg) + geom_point(mapping = aes(x=displ, y = hwy, alpha = cty), position='jitter')
```



DATA 100,Week 2 (A)

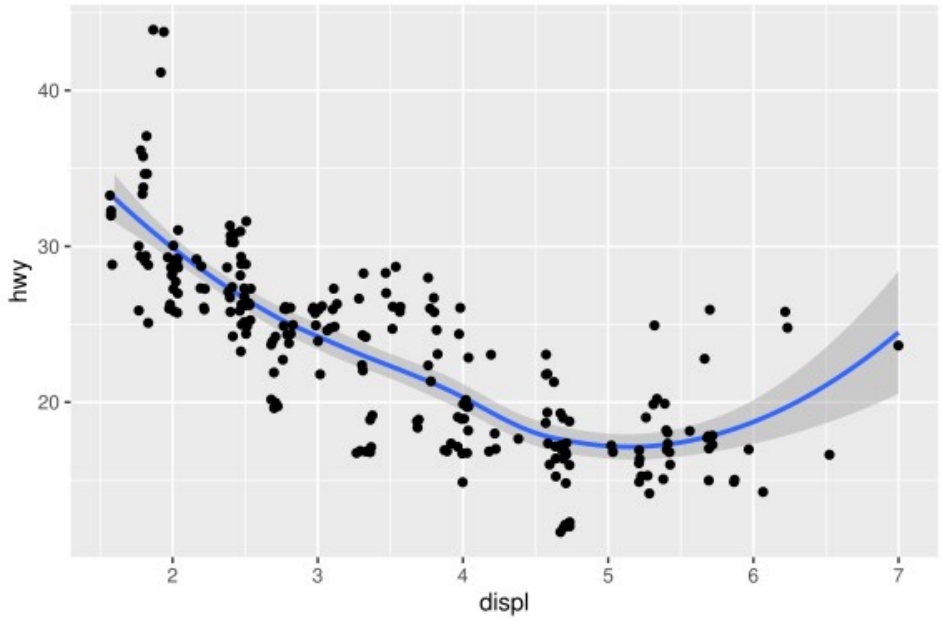
ggplot(data = mpg) +geom_jitter(mapping = aes(x=displ, y = hwy, alpha = cty))



DATA 100,Week 2 (A)

Use geom_smooth

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



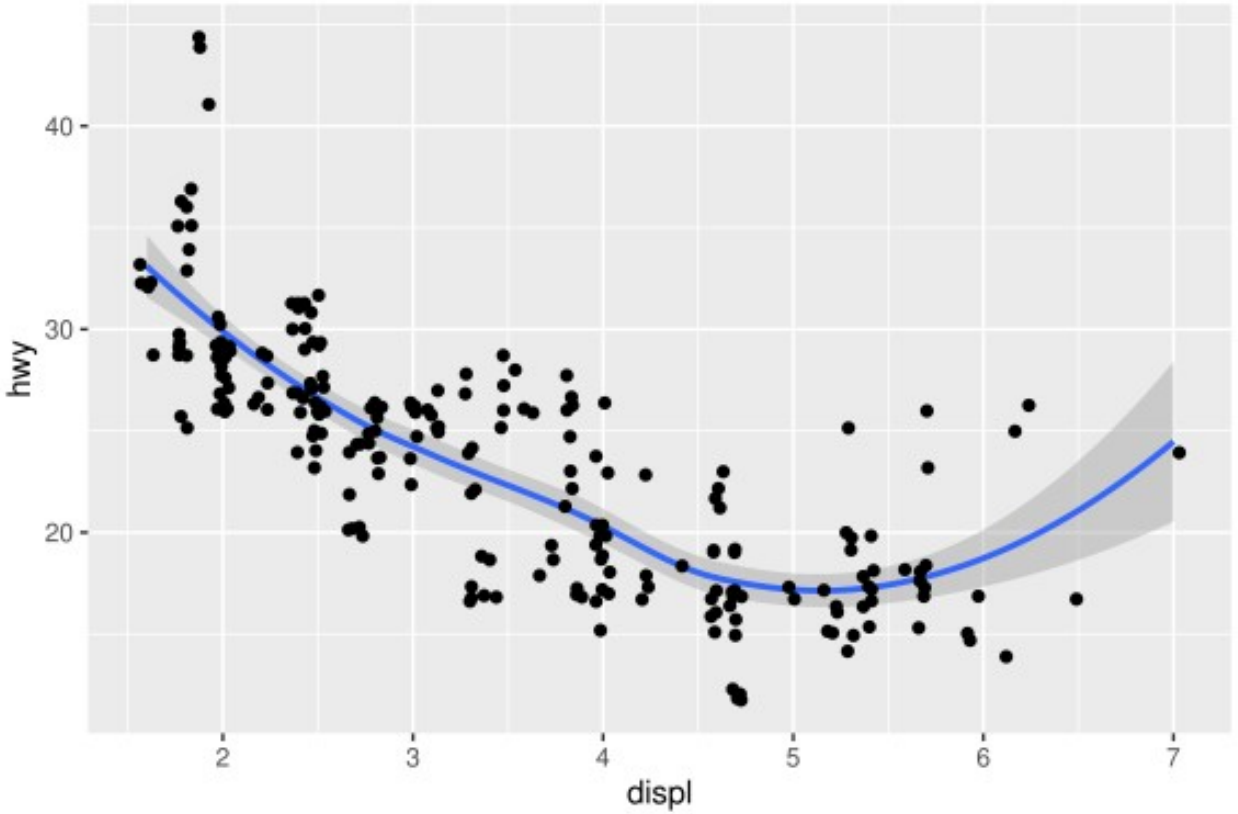
DATA 100,Week 2 (A)

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

Comment: Good coding style: *try not repeat*. The following is good, since it reduces repetition while achieving the same effects as above. Can do this because ggplot and geom_ functions are implemented this way.

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

DATA 100,Week 2 (A)

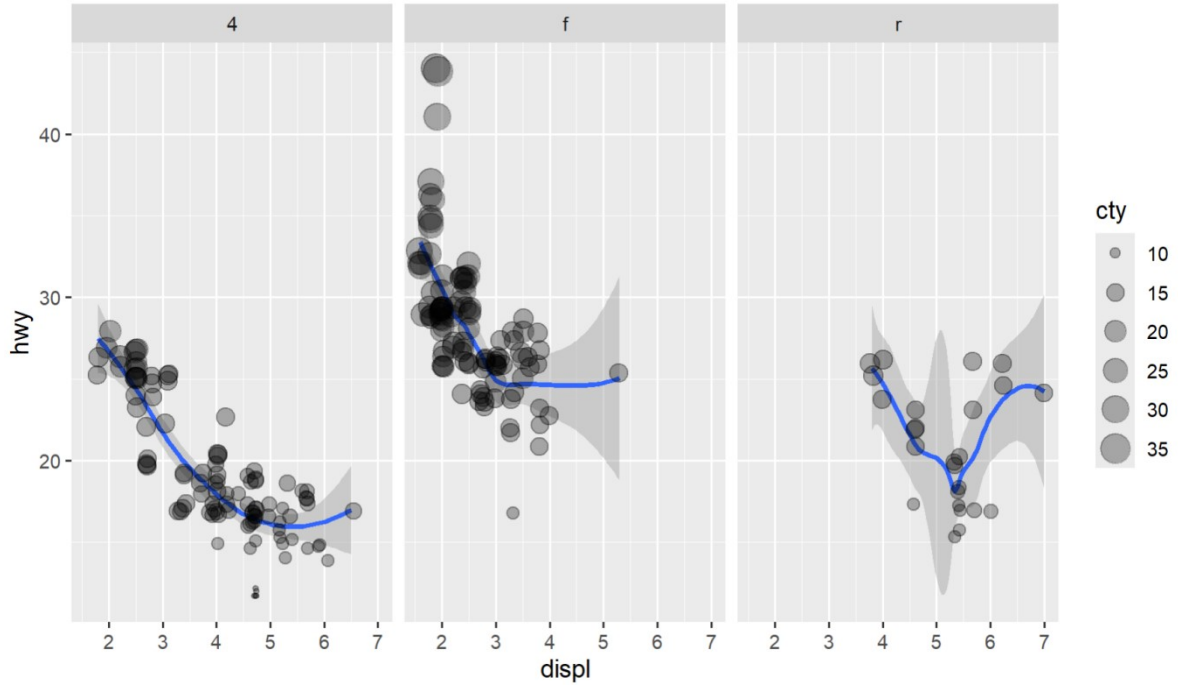


DATA 100,Week 2 (A)

- linetype

- color

```
ggplot(data=mpg, mapping = aes(x=displ, y=hwy)) + geom_smooth() +
geom_jitter(aes(size = cty), alpha = 0.3) + facet_wrap(~drv)
```



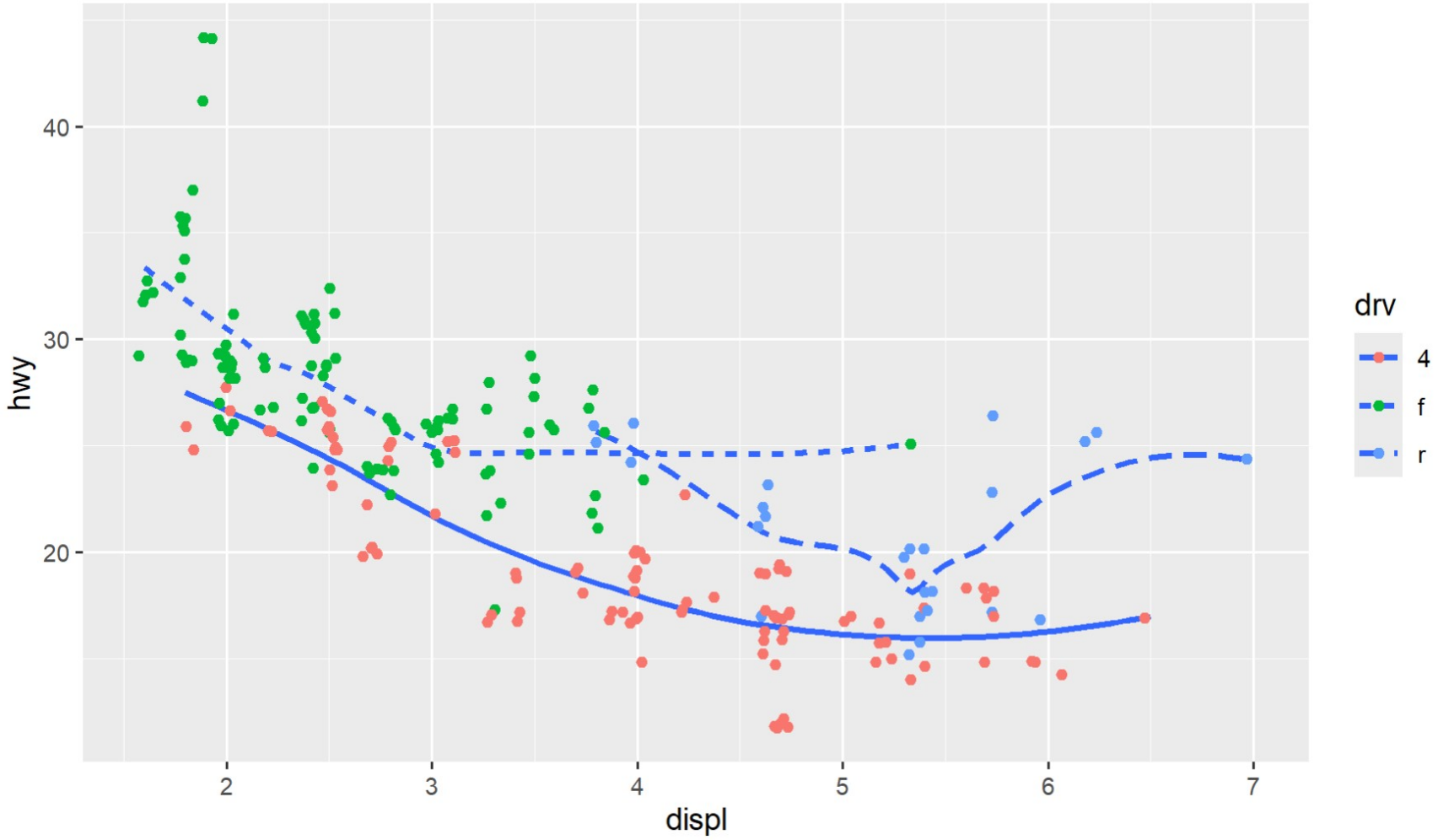
DATA 100,Week 2 (A)

If only want the smooth line (without the shades) can use se=FALSE

• this gets rid of the shades around the curve, which represents the uncertainty of the estimates

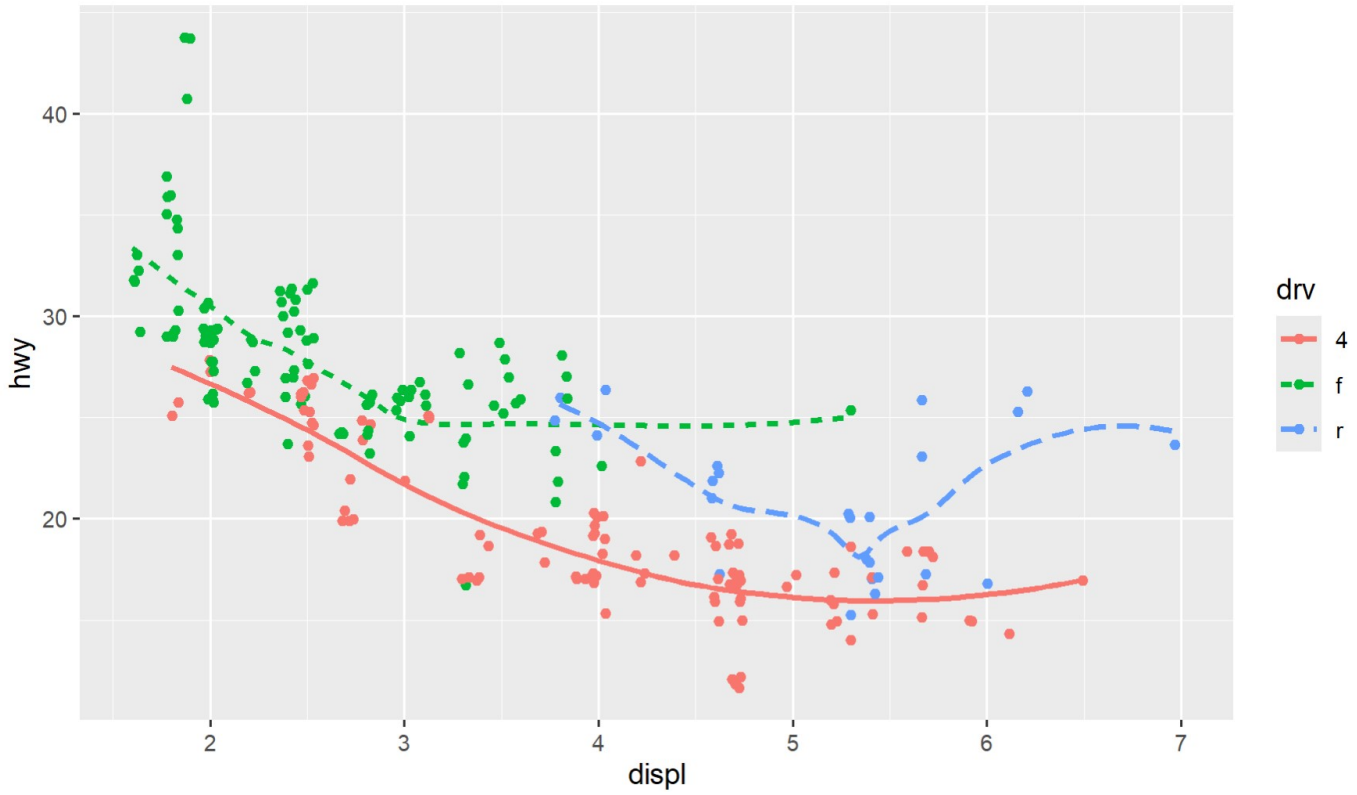
```
ggplot(data = mpg, mapping = aes(x=displ, y=hwy)) + geom_smooth(mapping = aes(linetype = drv), se=FALSE) + geom_jitter(mapping=aes(color=drv))
```

DATA 100,Week 2 (A)



DATA 100,Week 2 (A)

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



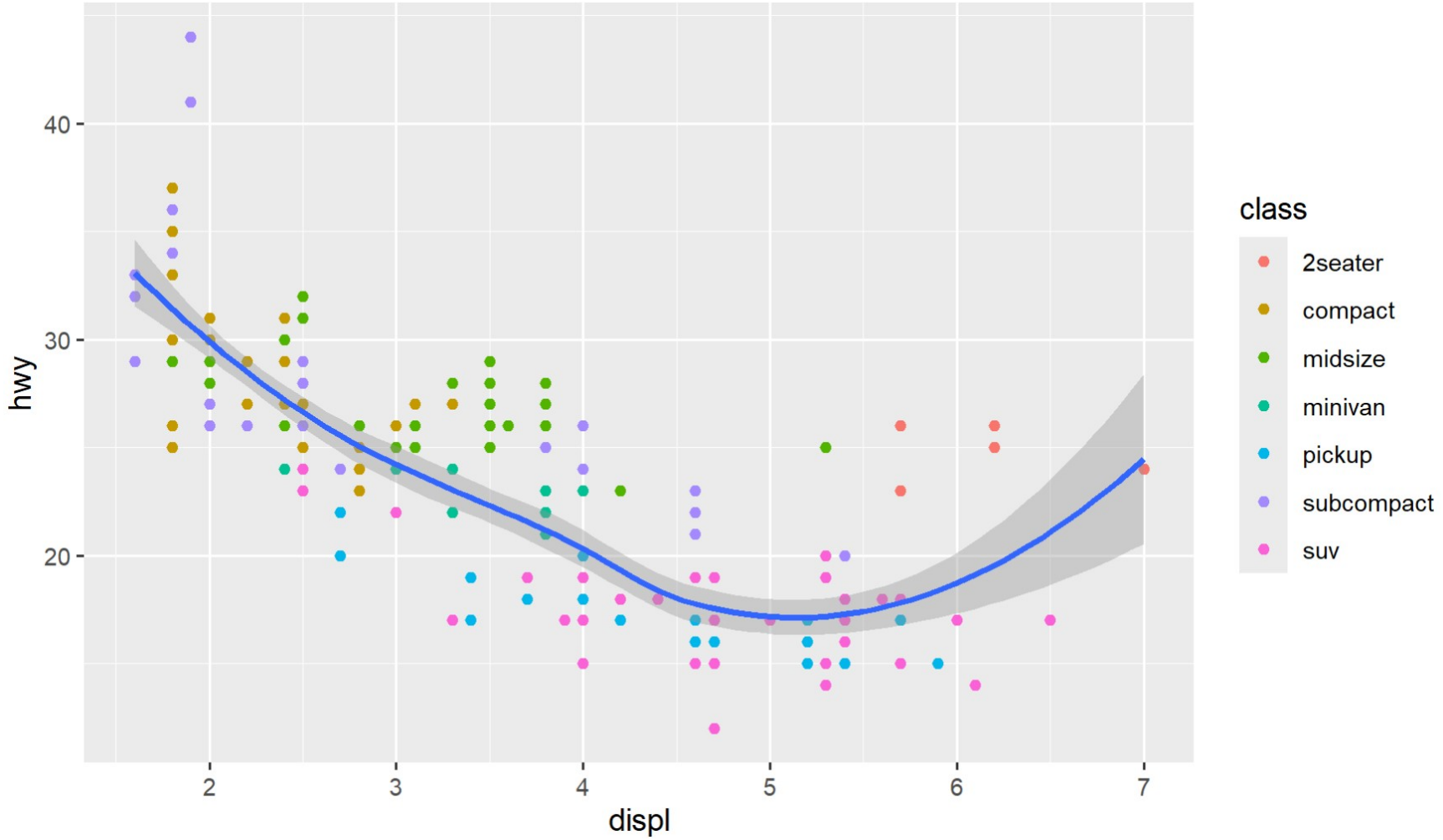
DATA 100,Week 2 (A)

Use filter

So far, either all data is used in plotting or each group is plotted separately at the same time.

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

DATA 100,Week 2 (A)



DATA 100,Week 2 (A)

filter function in dplyr package allows plotting several groups of data together, while exclude other groups.

• Criteria are set using *logic operations*, and only include the data that satisfy the criteria

Logical operations

• **!=**: ***NOT EQUALS TO***, outputs TRUE or FALSE
• **==**: ***EQUALS TO***, outputs TRUE or FALSE
• **%in%**: ***BELONGS TO***, outputs TRUE or FALSE
• **!**: ***NOT***, turns a TRUE into FALSE, and vice versa

DATA 100,Week 2 (A)

pipe

A pipe is represented by |>, which makes code easier to read

• in brief, the pipe **takes the thing on its left and passes it along to the function on its right**

• so that x |> f(y) is equivalent to f(x, y)

• and x |> f(y) |> g(z) is equivalent to g(f(x, y), z)

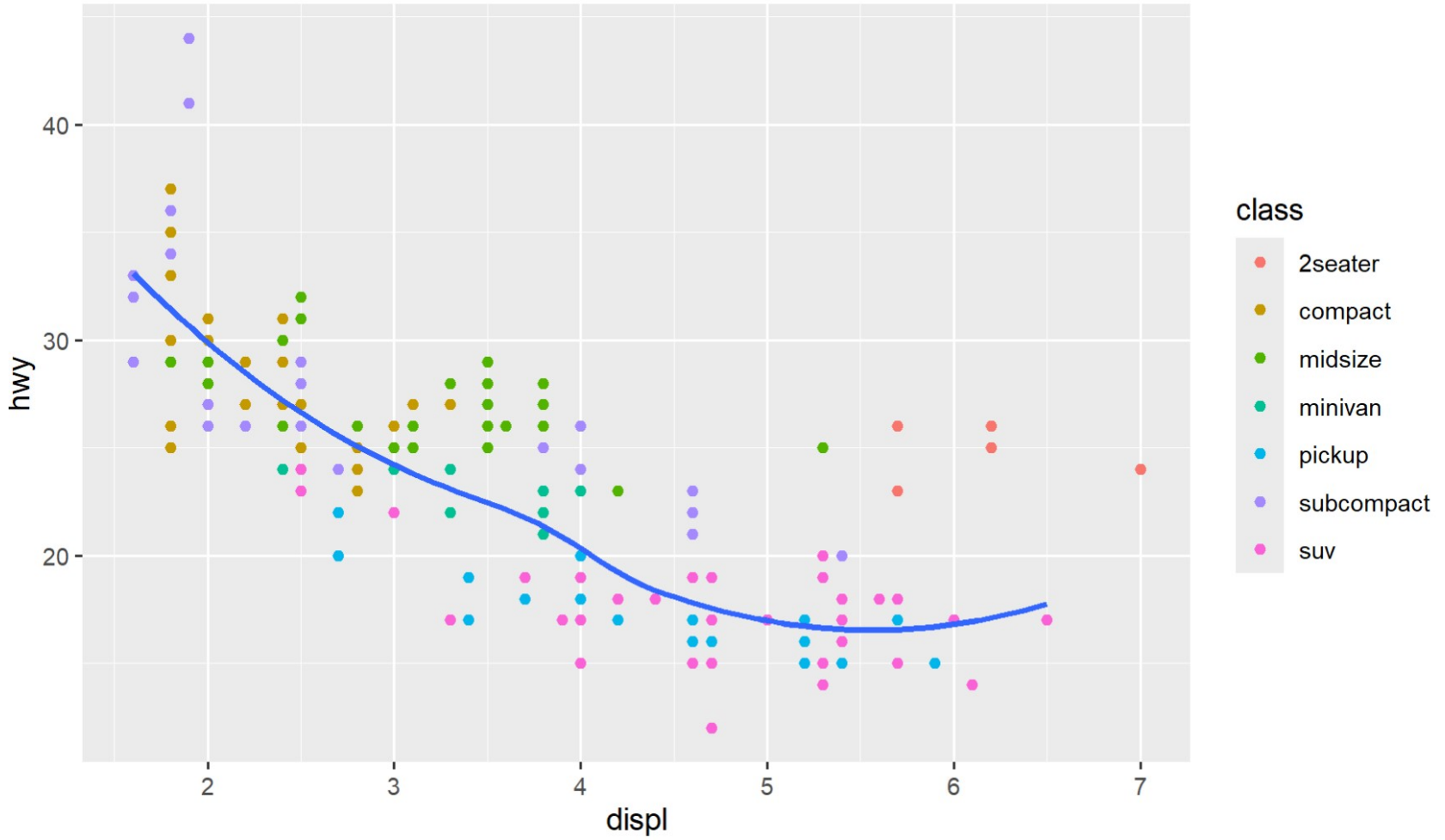
```
sum((1:5)^2, 9)
#> [1] 64
(1:5)^2 |> sum(9)
#> [1] 64
```

DATA 100,Week 2 (A)

Now get rid of the 2seaters that bent the rightmost tail up in the figure above, **only** for the geom_smooth plot

```
ggplot(
data = mpg,
mapping = aes(x = displ, y = hwy) ) +
geom_point(mapping = aes(color = class)) + geom_smooth(
data = mpg |> filter(class != "2seater"),
#data = filter(mpg, class != "2seater"),
se = FALSE
)
```

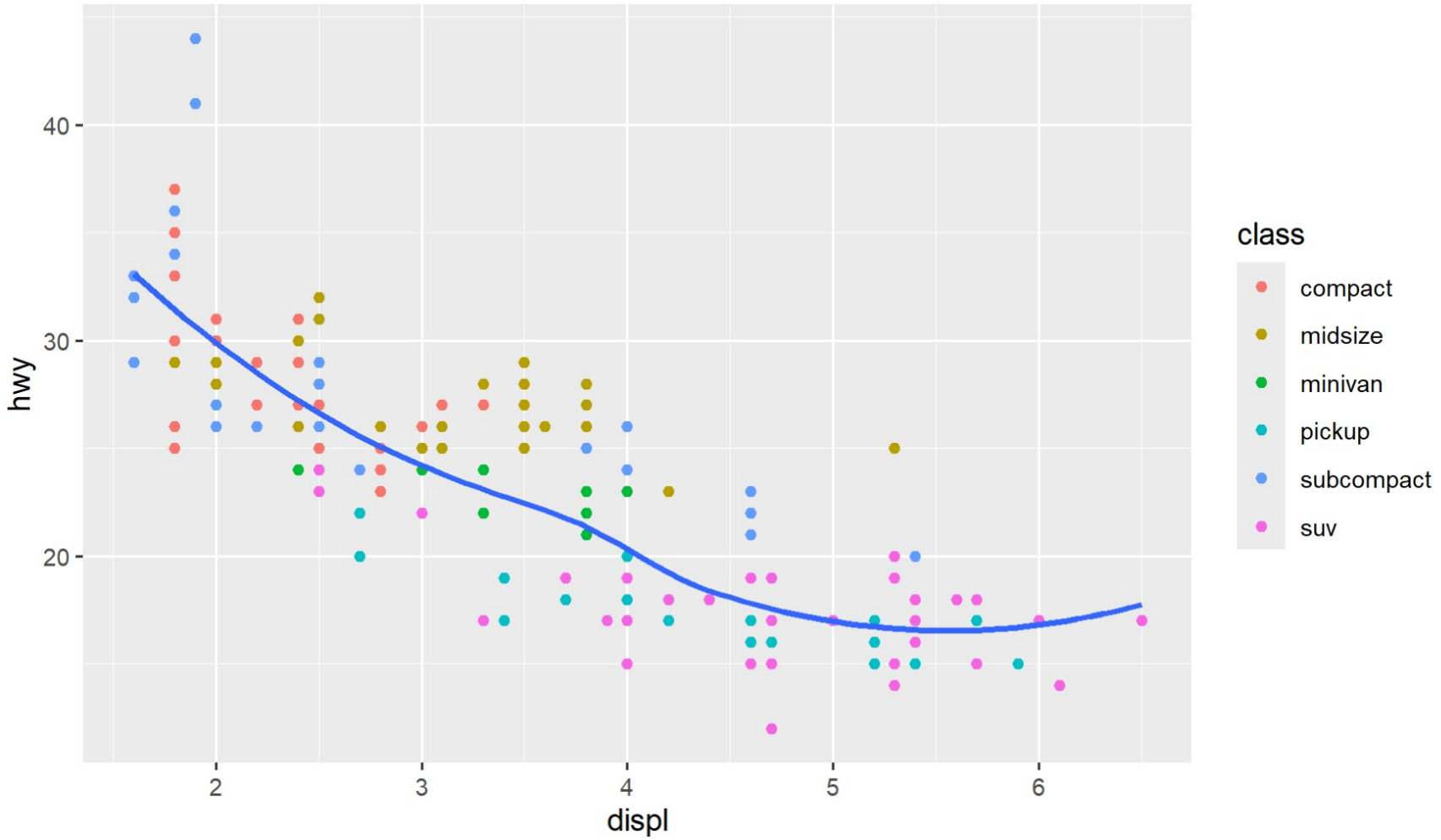
DATA 100,Week 2 (A)



DATA 100,Week 2 (A)

```
ggplot(
data = mpg,
mapping = aes(x = displ, y = hwy) ) +
geom_point(data = mpg |> filter(class != "2seater"), mapping = aes(color =
class)) + geom_smooth(
data = mpg |> filter(class != "2seater"),
#data = filter(mpg, class != "2seater"),
se = FALSE
)
```

DATA 100,Week 2 (A)



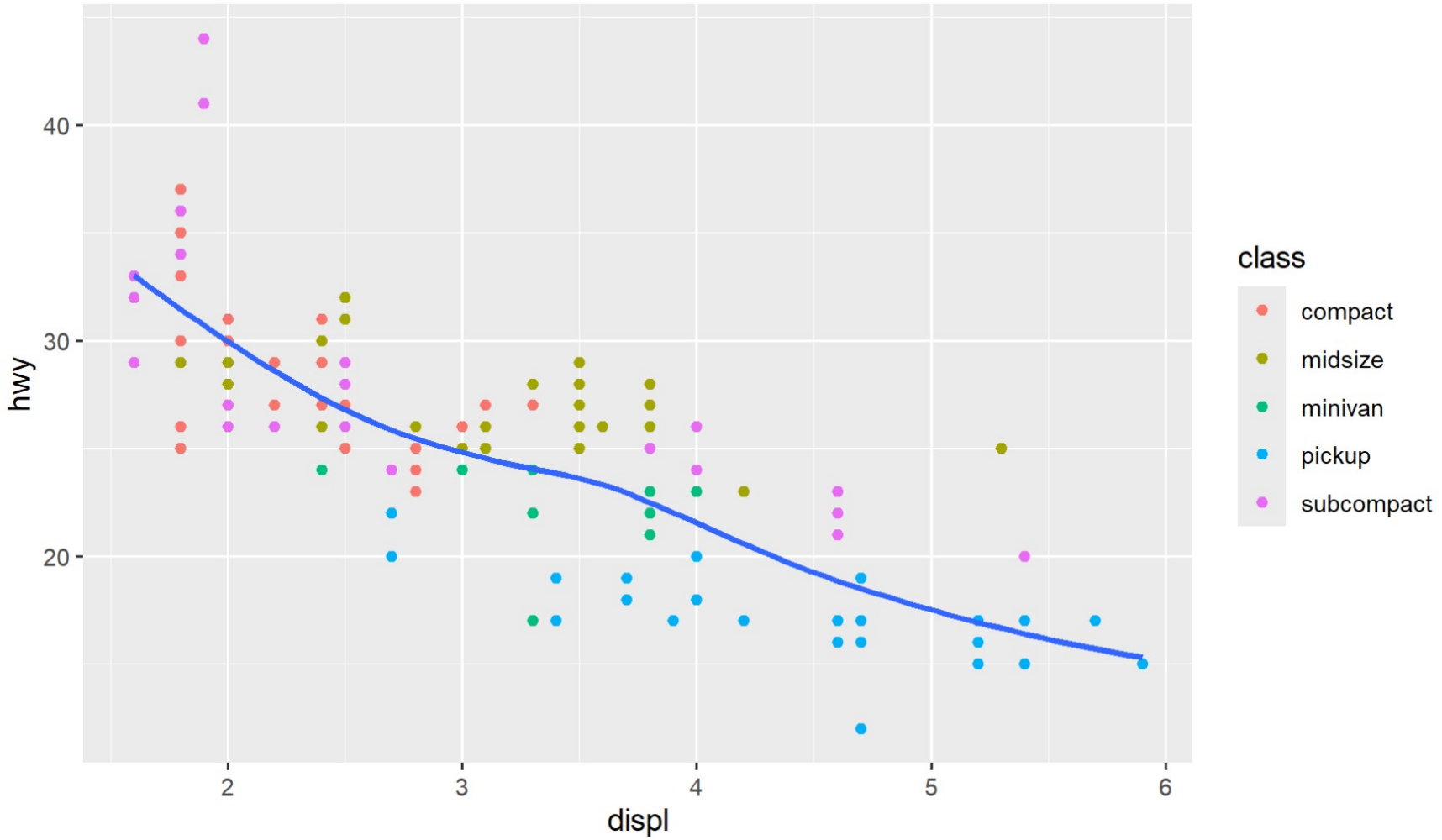
DATA 100,Week 2 (A)

Notice: The suvs really just get into the picture halfway through, and it is flattening things out there. Can we get rid of both?

• Use the c() function to make a list of things to filter out.

```
# how to also get rid of the "suv" from the scatter plot?
# mpg |>
# filter(!(class %in% c("2seater", "suv"))) |>
ggplot(
data = filter(mpg, !(class %in% c("2seater", "suv"))),
mapping = aes(x = displ, y = hwy)
) +
geom_point(mapping = aes(color = class)) + geom_smooth(
se = FALSE
)
```

DATA 100,Week 2 (A)



DATA 100,Week 2 (A)

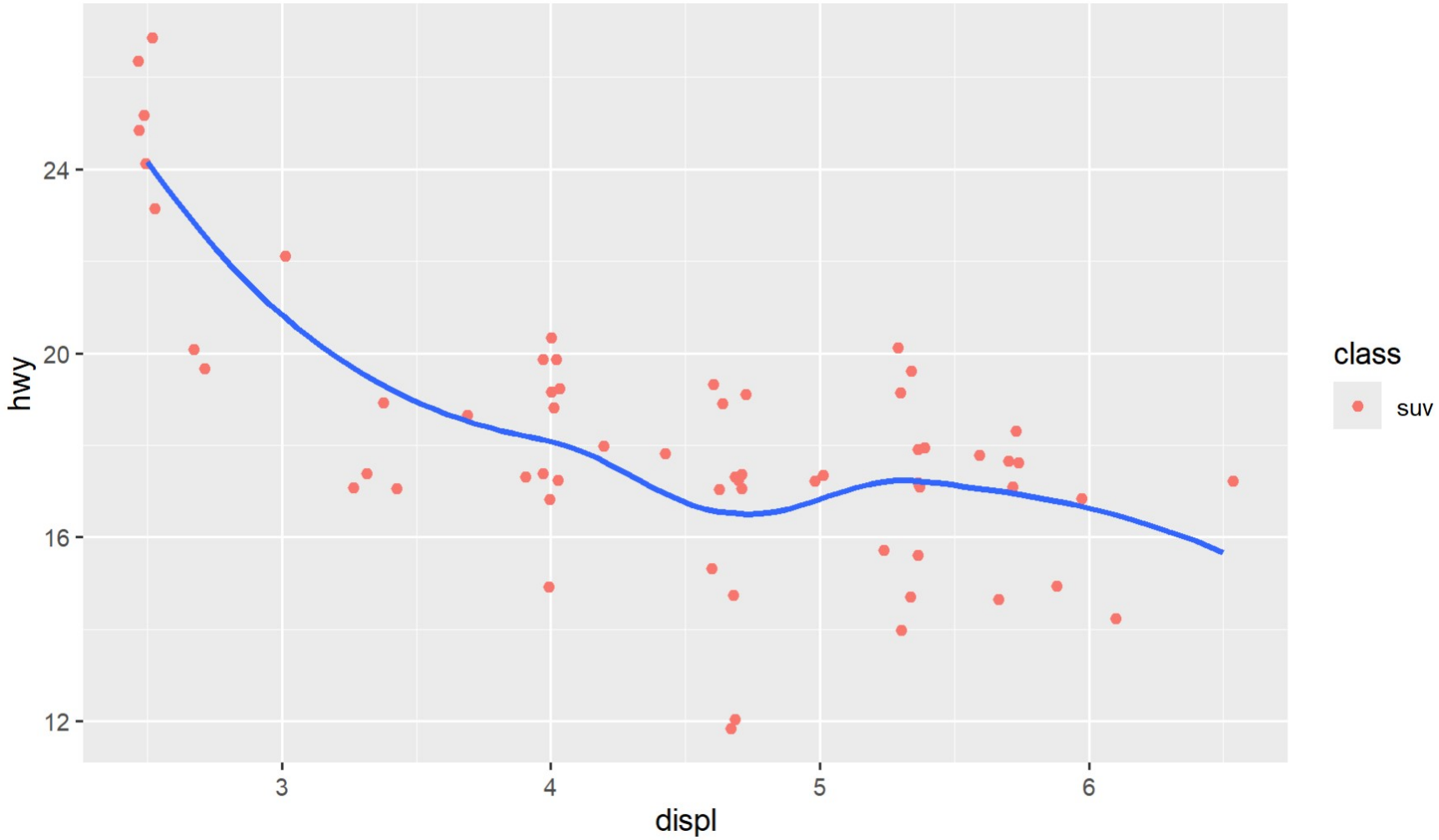
There are many suvs, and simply throwing them away does not make sense.

We use filter again to plot them separately

• We see here the flexibility in extracting information using a combination of methods / functions.

```
ggplot(data = mpg |> filter(class == "suv"),
mapping = aes(x = displ, y = hwy)) + geom_jitter(mapping = aes(color = class)) +
geom_smooth(se = FALSE)
```

DATA 100,Week 2 (A)



DATA 100,Week 2 (A)

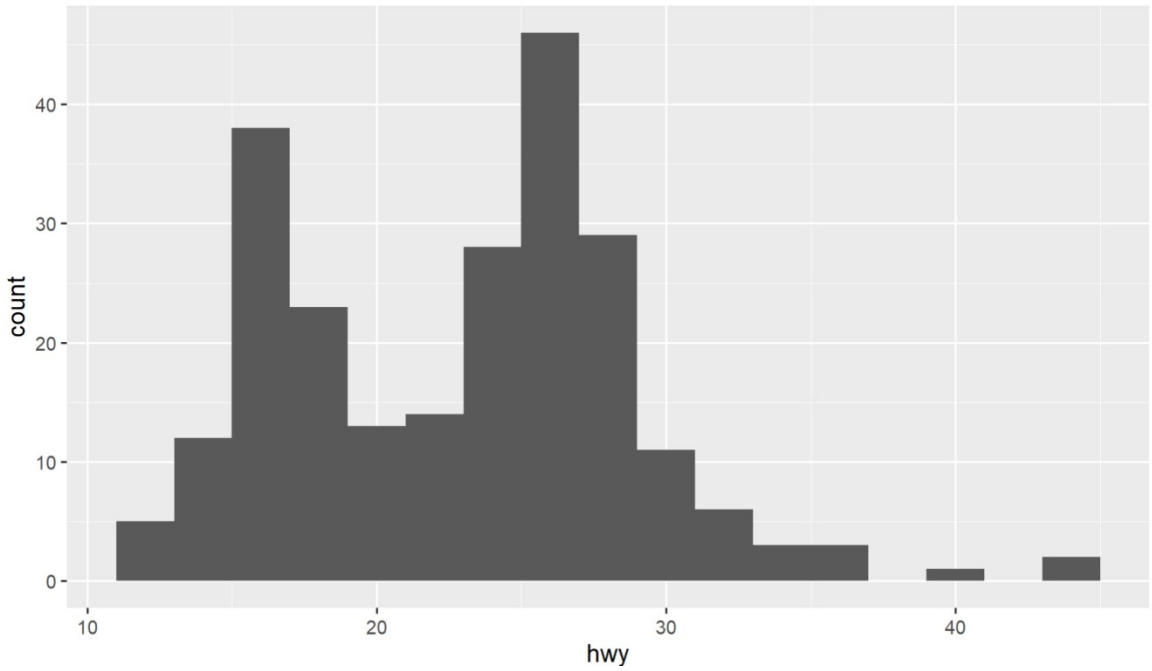
More geometries

Can completely transform the look of your plot by using a different geom_ function, which can reveal different features of your data.

• Notice that we dropped data = from the ggplot function call below. Instead, simply put the dataframe mpg as the first
parameter.

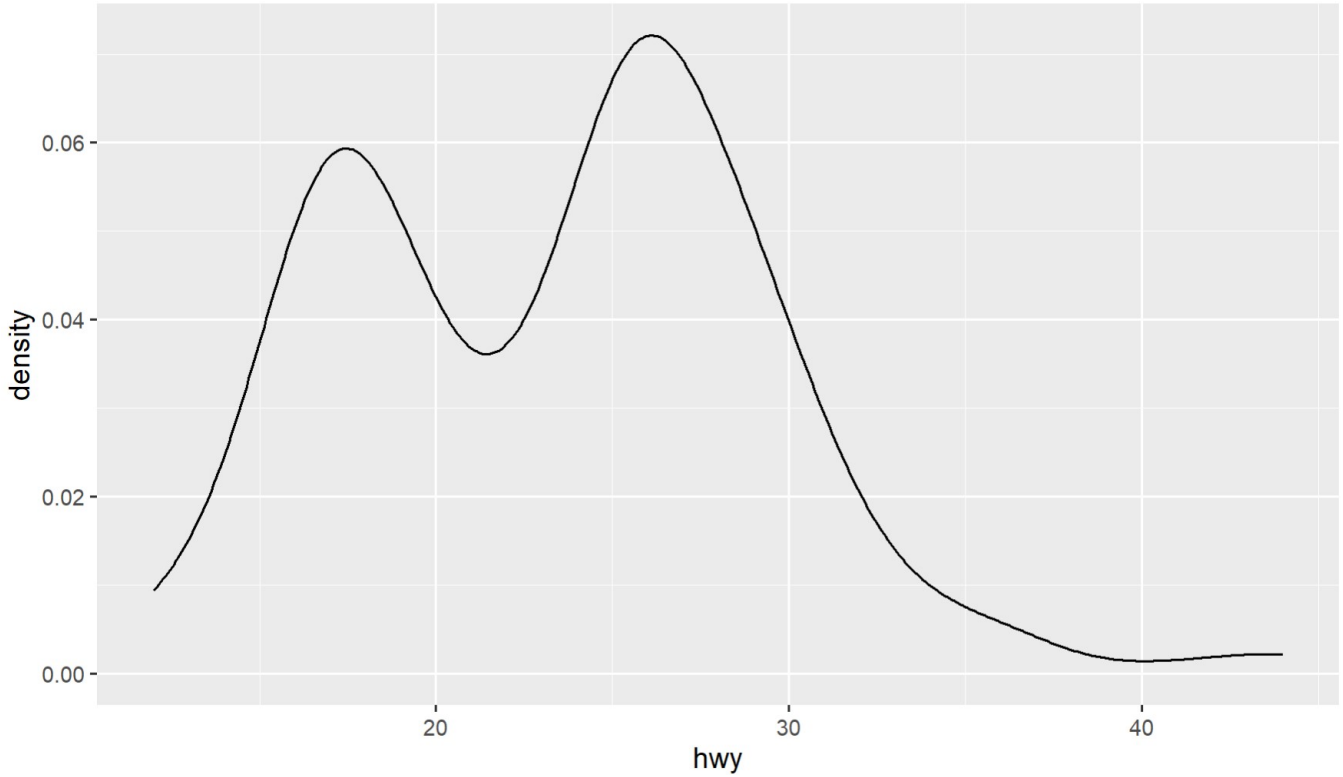
DATA 100,Week 2 (A)

```
# histogram
ggplot(mpg, aes(x = hwy)) +
geom_histogram(binwidth = 2)
```



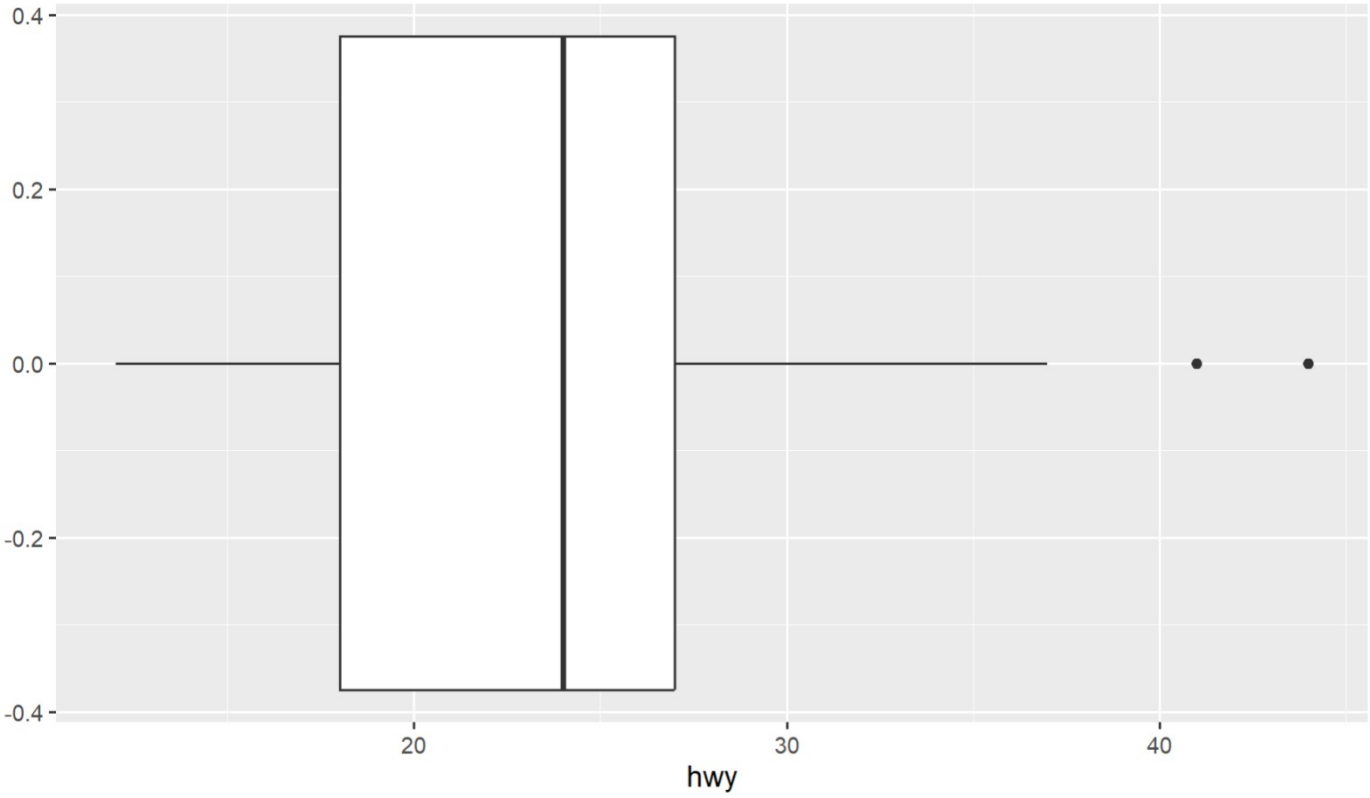
DATA 100,Week 2 (A)

```
# density
ggplot(mpg, aes(x = hwy)) + geom_density()
```



DATA 100,Week 2 (A)

```
# boxplot
ggplot(mpg, aes(x = hwy)) + geom_boxplot()
```

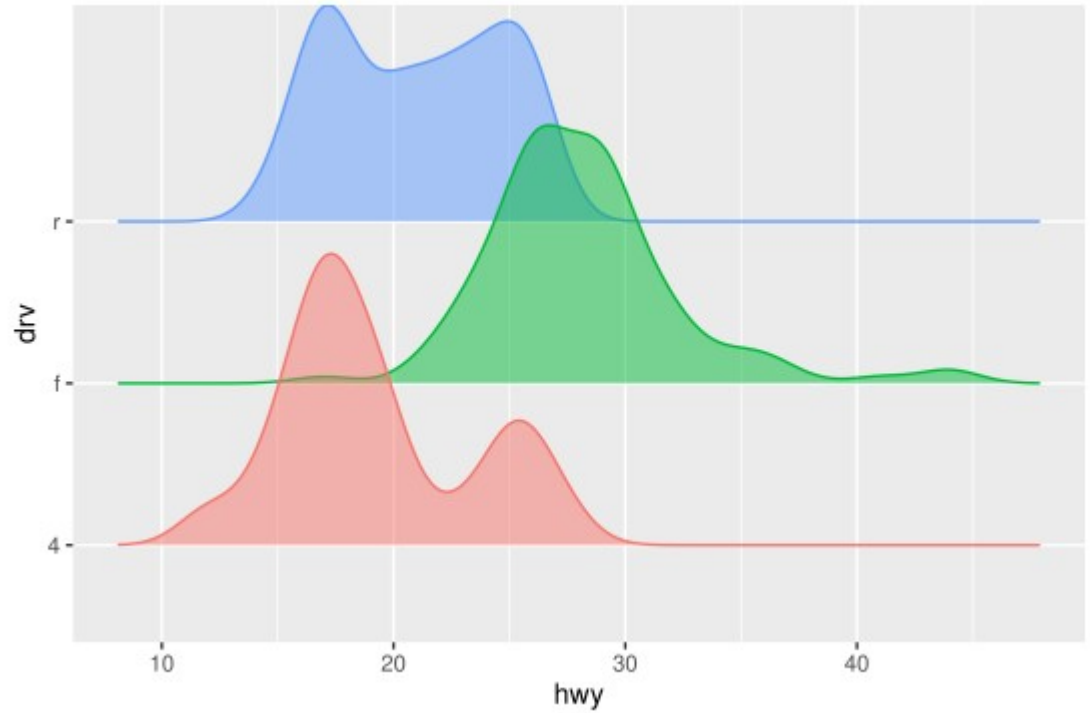


DATA 100,Week 2 (A)

```
library(ggridges)

ggplot(mpg, aes(x = hwy, y = drv, fill = drv, color = drv)) +
geom_density_ridges(alpha = 0.5, show.legend = FALSE)
#> Picking joint bandwidth of 1.28
```

DATA 100,Week 2 (A)



DATA 100,Week 2 (A)

Descriptive statistics, II

Reviewed / discussed here are range, interquartile range (IQR), outliers, and boxplot. These are related to the previously reviewed max, min, and quartile
Set up the same 20 integers

```
# The following code puts the 20 integers into the variable `sample`
## use "?c" in console to read the help messages
# The name `sample` now represents the 20 integers
sample <- c(29, 35, 33, 32, 34, 30, 28, 33, 34, 35, 10, 14, 14, 12, 12, 12, 10, 14,
16, 10)
```

DATA 100,Week 2 (A)

Range = max **-** min**: the overall spread of the data**

For sample, since max is 35, and min is 10, the range is 35 - 10 = 25

```
max(sample) - min(sample)
#> [1] 25
```

DATA 100,Week 2 (A)

The range() function outputs the max and min of the data as a pair

```
ran <- range(sample) ran
#> [1] 10 35
ran[1]
#> [1] 10
ran[2]
```

DATA 100,Week 2 (A)

Interquartile range (IQR) = Q3 - Q1

• Q3: the 75th percentile, namely, about 75% of data are below it
• Q1: the 25th percentile, namely, about 25% of data are below it
• IQR thus measures how much the middle half of the data spread
For sample, we have (from previous time) Q1 is 12, and Q2 is 33, thus IQR = 33 - 12 = 21
• the IQR() function computes the interquartile range

```
IQR(sample)
#> [1] 21
```

DATA 100,Week 2 (A)

Outliers

Basically, these are data points that are either *too large* or *too small*, out of ordinary range

• It can indicate error in data collection (human or machine), one sided distribution, etc

Outlier: data points below Q1 - 1.5 * IQR or above Q3 + 1.5 * IQR

Namely, using IQR to measure that they are *far* from the middle half of the collection. For sample, we see that Q1 is 12, IQR is 21,
Q3 is 33
• According to the criterion we use, an outlier should be either less than 12-1.5×21 = -19.5, or larger than 33+1.5×21 = 64.5

DATA 100,Week 2 (A)

• Since min of sample is 10, and max of sample is 35, there is no outlier in sample

```
summary(sample)
#> Min. 1st Qu. Median Mean 3rd Qu. Max.
#> 10.00 12.00 22.00 22.35 33.00 35.00
```

DATA 100,Week 2 (A)

Now consider a sample that has outliers:

```
sample2 <- c(43, -4, 20, 42, 24, 21, 0, 25, 21, 24, 42, 23, 2, 4, 23, 24, 5, 42, 40, 23)
summary(sample2)
#> Min. 1st Qu. Median Mean 3rd Qu. Max.
#> -4.00 16.25 23.00 22.20 28.75 43.00
IQR(sample2)
#> [1] 12.5
```

The data sample2 consists of another list of 20 integers

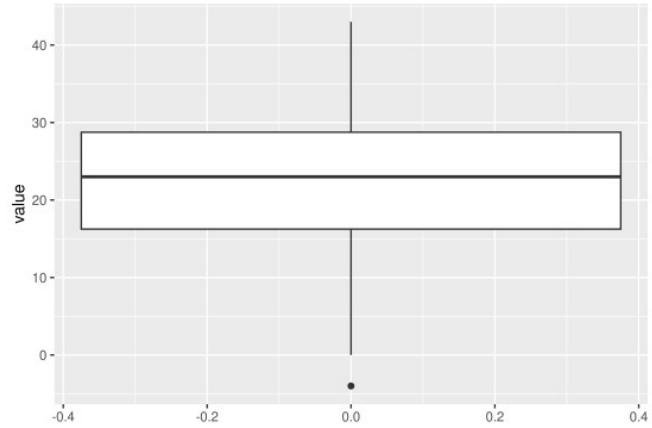
- min is -4, max is 43
- Q1 is 16.25, Q3 is 28.75, and thus IQR is 12.5
- Any potential outlier in sample2 should be either less than $16.25 - 1.5 \times 12.5 = -2.5$, or larger than $28.75 + 1.5 \times 12.5 = 47.5$
- Thus there is one outlier, that is -4

DATA 100,Week 2 (A)

Box plot captures all these information

Elements of a boxplot
- Middle bar: median
- Top edge of the box: Q3
- Lower edge of the box: Q1
- Whiskers extends to the **farthest non-outlier entry**

- Outliers: plotted separately as dots above / below the whiskers



DATA 100,Week 2 (A)

Comment: enframe wraps the *vector* sample2 into a *dataframe*, so that ggplot can be used

```
#> # A tibble: 20 x 2
#> name value
#> <int> <dbl>
#> 1 1 43
#> 2 2 -4
#> 3 3 20
#> 4 4 42
#> 5 5 24
#> 6 6 21
#> # i 14 more rows
```