# Faceting with ggplot2

For best results, open this file in RStudio and click the button labelled "Visual" at the top left of the screen. It's even better if you keep it in the same folder as your other notes for this class!

### Recap

We need to run `install.packages("tidyverse")` **once**, then we load the package every time we start R:

```
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     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 be
```

We worked with the `mpg` data, which is built into `ggplot2`, which is a package that you get when you install `tidyverse`:

```
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~
```

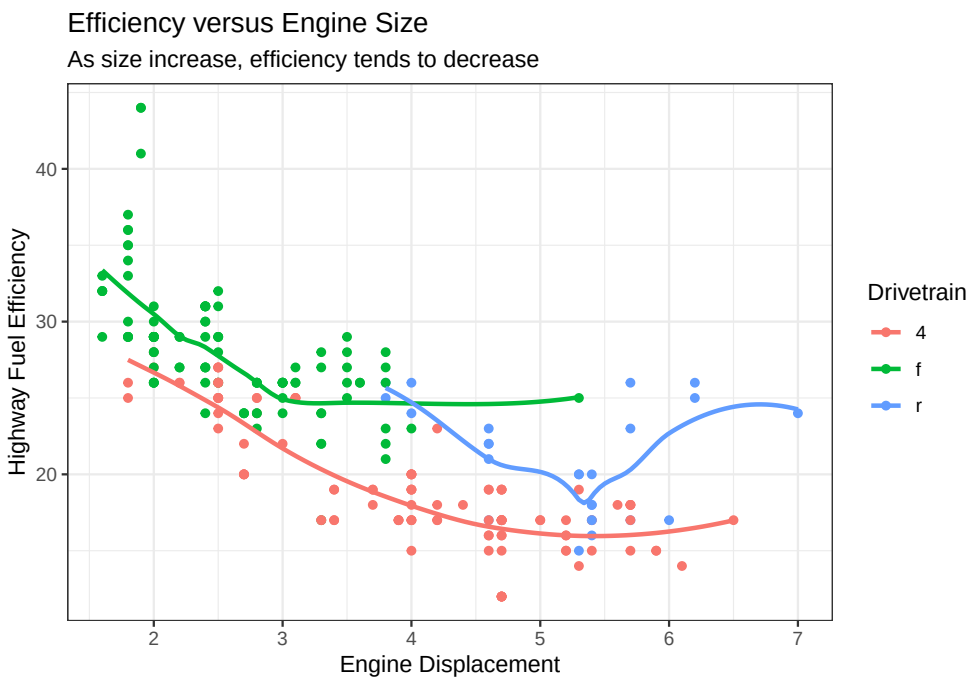
- `mpg` is the box that holds the data.
- `displ`, `hwy`, and `drv` are examples of columns within the data. R can't find these unless you tell it to look inside `mpg`.

We also worked with the `palmerpenguins` data, but we're switching to `mpg` for this lesson.

We slowly built up something like the following ggplot (note that I've made it a little bit better):

```
ggplot(data = mpg) +
    aes(x = displ, y = hwy, colour = factor(drv)) +
    geom_point() +
    geom_smooth(se = FALSE) +
    labs(
        x = "Engine Displacement",
        y = "Highway Fuel Efficiency",
        colour = "Drivetrain",
        title = "Efficiency versus Engine Size",
        subtitle = "As size increase, efficiency tends to decrease"
    ) +
    theme_bw()
```

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

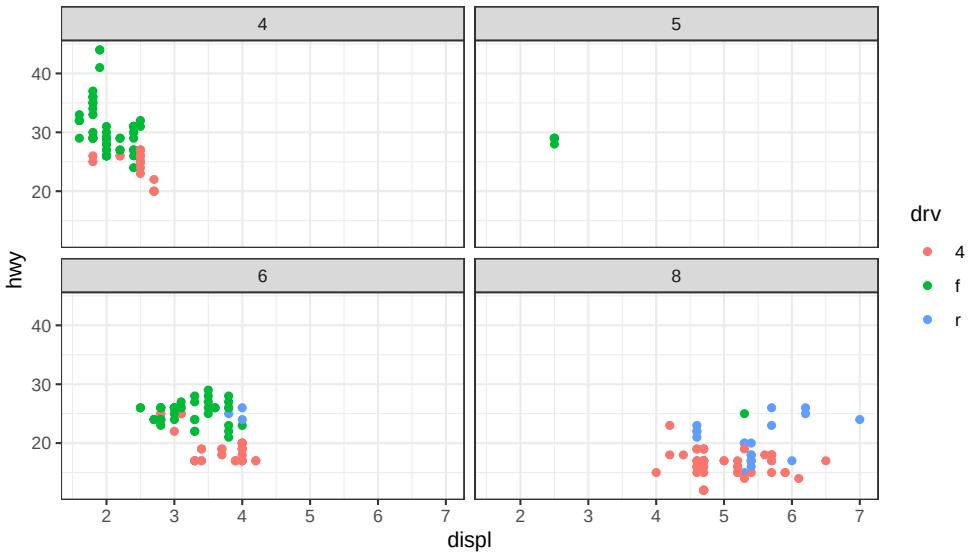


In class I'll talk for a while at this point. Take some notes here:

- 
- 
- 
- 
- 
- 
- 

### `facet_wrap()` with respect to a variable

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



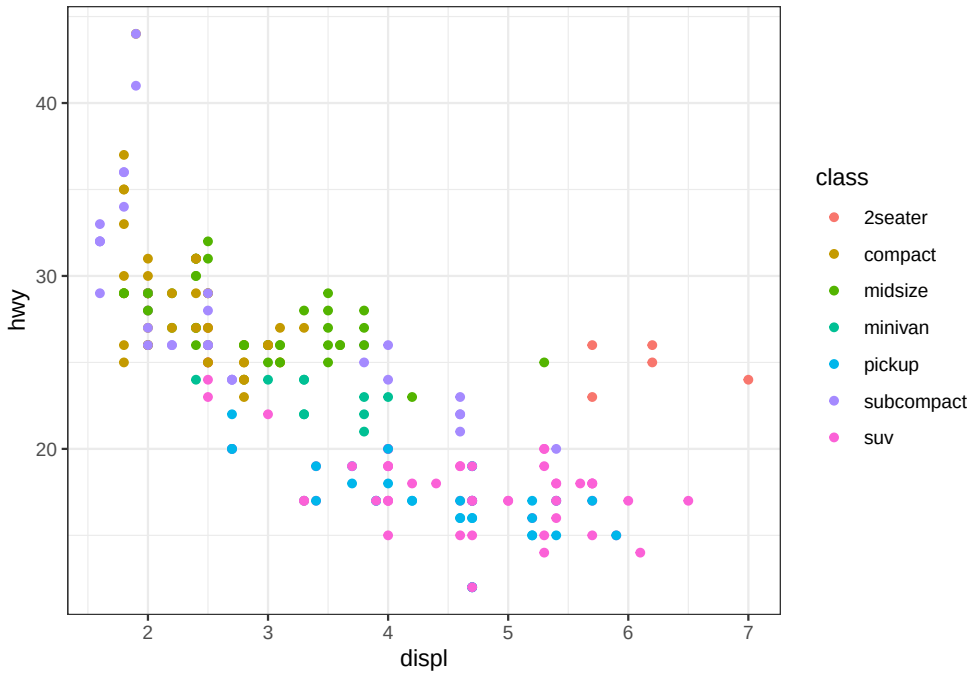

Why is there a `~` before the column name? It's complicated, but here's a simplified version:

- The name `cyl` is part of the `mpg` dataframe. It doesn't exist outside `mpg`.
- The `~` makes it a special kind of object called a **formula**. R won't search for things in the formula until later.
- `facet_wrap()` will tell R to look for `cyl` only when it needs to, and will direct it to the `mpg` dataframe.

### `facet_grid()` with respect to two variables

This generates a grid of plots, labelled across columns and rows by the values in the respective variable

```
ggplot(data = mpg) +
    geom_point(mapping = aes(x = displ, y = hwy, colour = class))+
    #facet_grid(rows = vars(drv), cols = vars(cyl))
    facet_grid() +
    theme_bw()
```




### Try out on your own

- Use `facet_grid(. ~ cyl)` or `facet_grid(drv ~ .)` in the last code snippet to see what happens.
- Compare `facet_grid(. ~ cyl)` and `facet_wrap(~ cyl)`.
- With the various choices of `facet_grid()` in the last code snippet, try add `colour` representing `class` to the plot, and see if more information can be extracted just by looking.

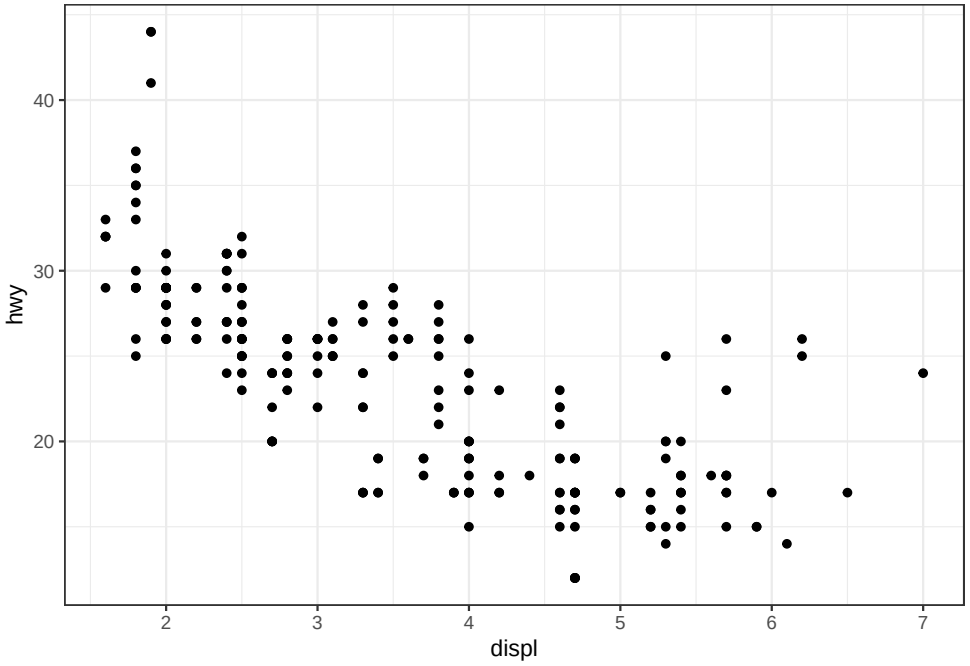
### More scatter plots

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)

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

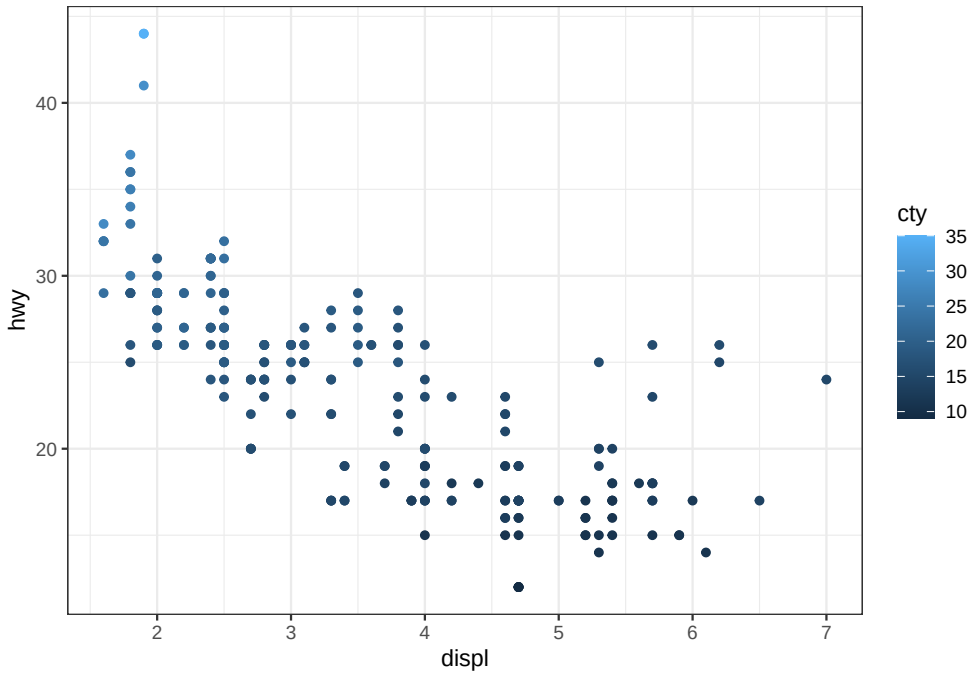



- We'll use this as a basis.
  - Follow along in RStudio!

### Including the `cty` (in-town mileage) variable

- Decorate the points differently according to its `cty` value, using one of the `aesthetics`.
  - Try `colour`, `shape`, `alpha` (shading), `size`

```
# comment / uncomment the appropriate lines to test the corresponding aesthetics
ggplot(data = mpg) +
    theme_bw() +
    # what's the difference with `colour = class`
    geom_point(mapping = aes(x=displ, y = hwy, colour = cty))
```



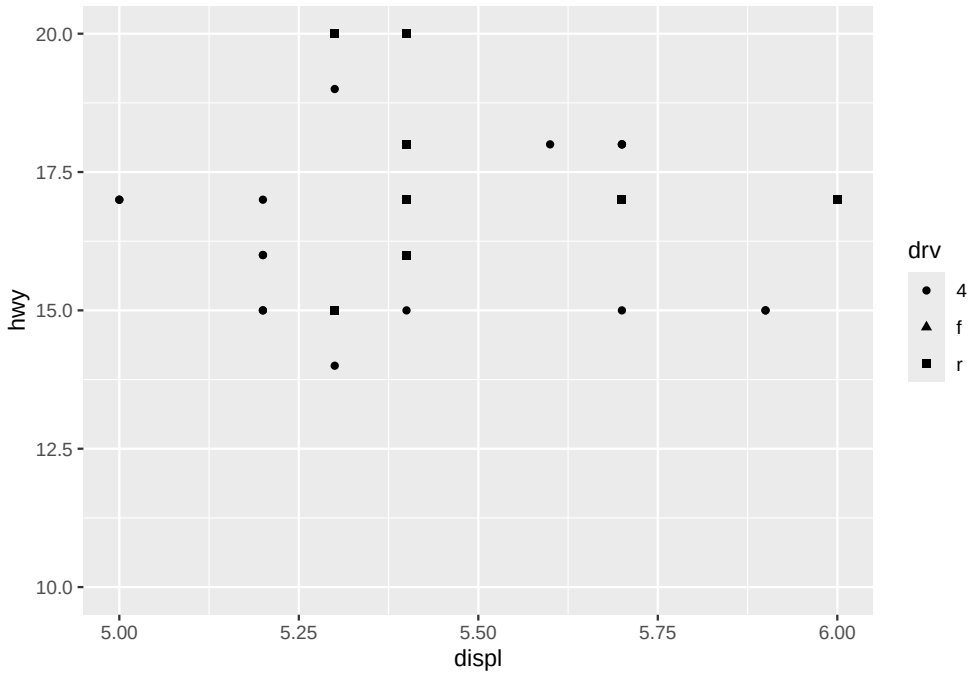

```
    # would it work?
    #geom_point(mapping = aes(x=displ, y = hwy, shape = cty))
    # how does this compare with `colour = cty`?
    #geom_point(mapping = aes(x=displ, y = hwy, alpha = cty))
    # talking about intuitive
    #geom_jitter(mapping = aes(x=displ, y = hwy, size = cty), alpha=0.3)
```

Try out `shape` for the `drv` (drive train)

- overlapping points become evident
- use `xlim()` and `ylim()` to scale the part of plots for more details

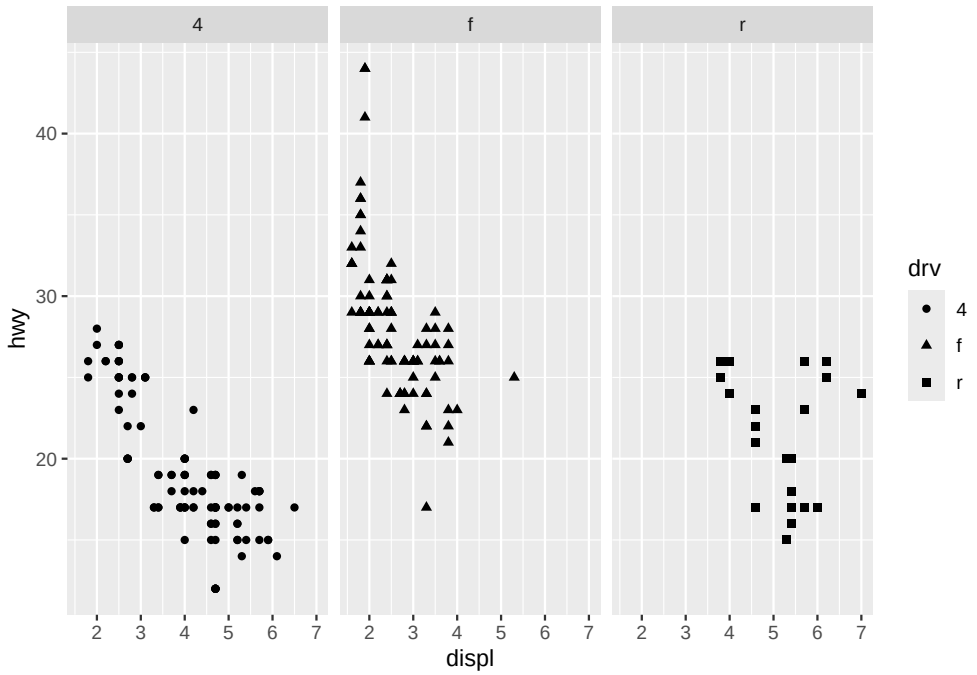
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x=displ, y = hwy, shape = drv)) +
  ylim(10,20) +
  xlim(5,6)
```

```
## Warning: Removed 204 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




We can also try to `facet_` it

```
# comment / uncomment the appropriate lines to test the corresponding aesthetics
ggplot(data = mpg) +
  geom_point(mapping = aes(x=displ, y = hwy, shape = drv)) +
  facet_wrap(~ drv)       # it works
```

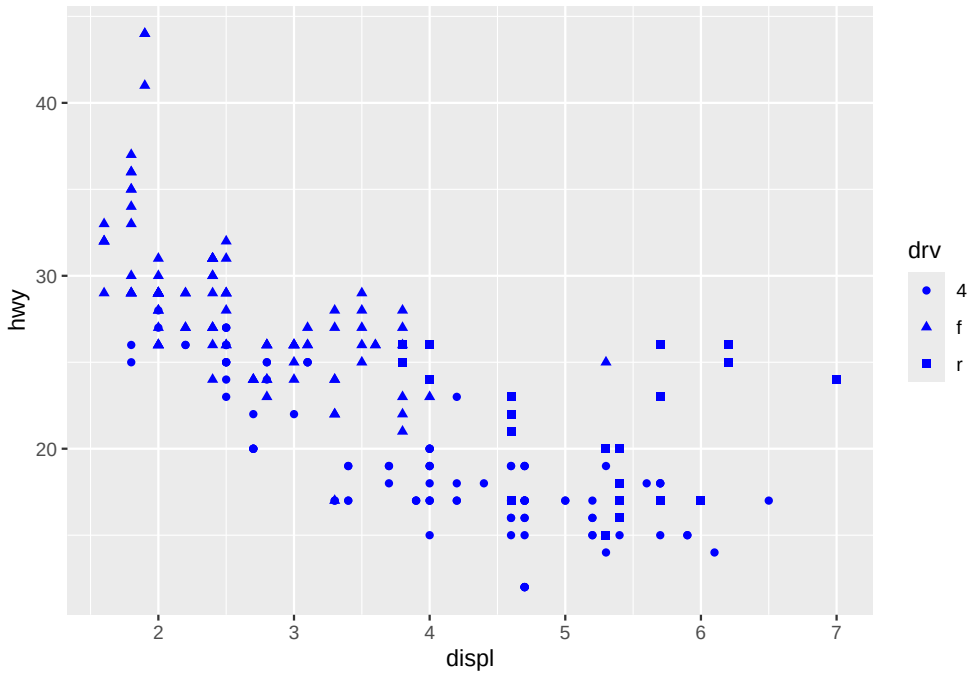



```
  #facet_grid(drv ~ .)      # this way seems to be better
  #facet_wrap(drv ~ .)        # could this work as well?
```

Can change overall aesthetics as well. Combination of effects can be very fancy

**Caution**: could go too far and end up with too much to handle

```
# comment / uncomment the appropriate lines to test the corresponding aesthetics
ggplot(data = mpg) +
  geom_point(mapping = aes(x=displ, y = hwy, shape = drv), colour='blue')
```

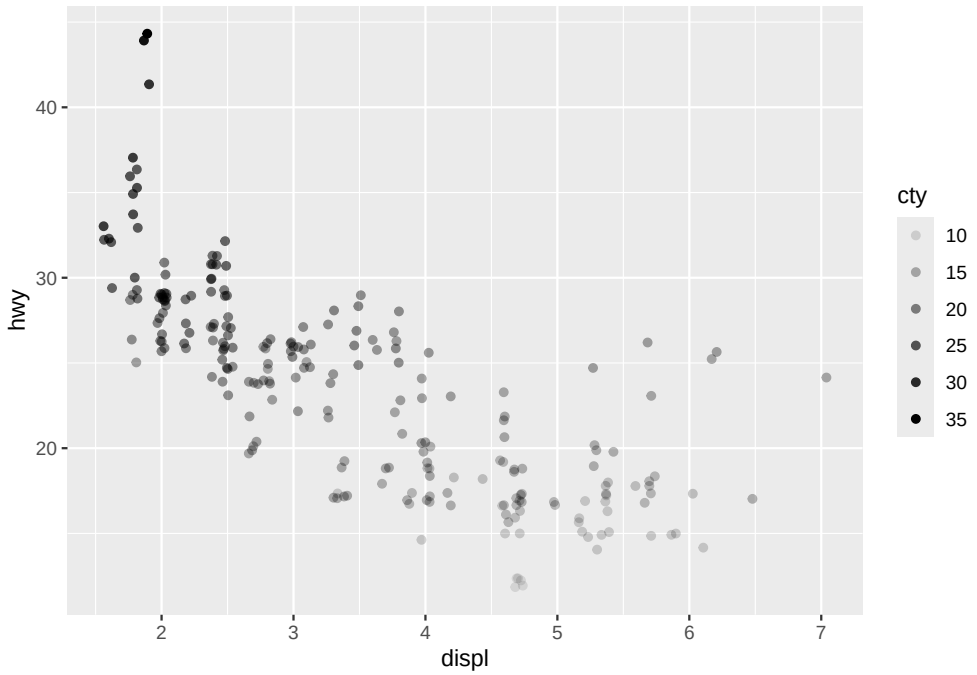



```
  #geom_point(mapping = aes(x=displ, y = hwy, shape = drv), size=3, alpha=0.3)
  #geom_point(mapping = aes(x=displ, y = hwy, colour = drv, fill=class), size=3, alpha=0.5, s
```

**Question**: What about overlapping points (*overplot*)?

- `jitter` them, with the `position` parameter
- or use `geom_jitter` directly
  - Rerun the code block to see that `jittering` really is *random*
  - Do not over-use `geom_jitter` unless it is necessary

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




**Use `geom_smooth`**

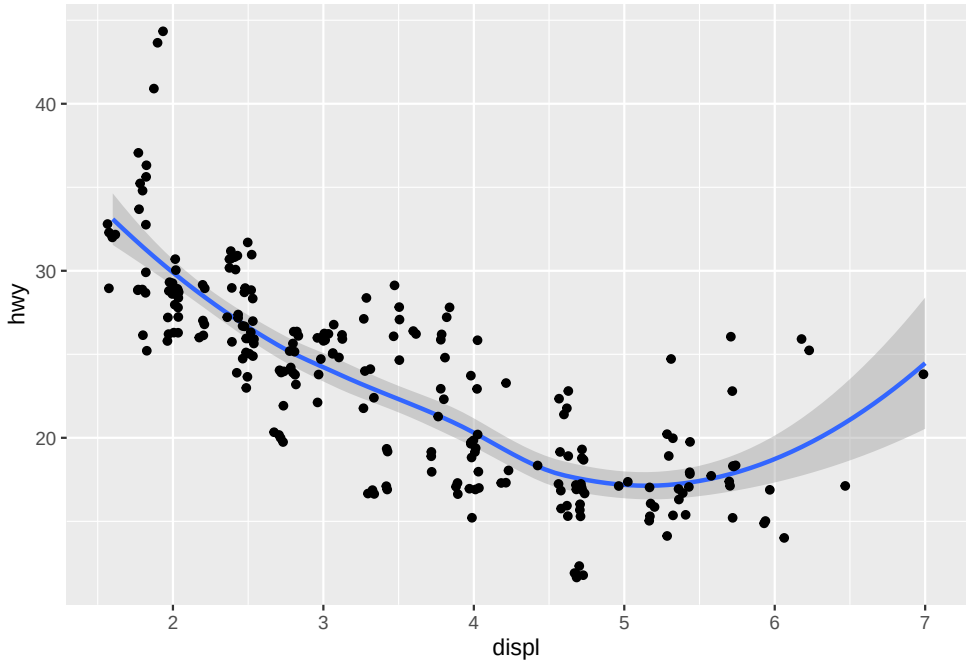
There are different ways to present data graphically.

Each function in the `geom_` family of functions in `ggplot2` is suitable in displaying some statistical property or properties. For instance

- `geom_point` gives raw concept of how data look like
- `geom_smooth` does statistics in the backstage, and outputs a *smooth* curve
  - illustrating trends, but can also be misleading taken alone

```
# Did this before, here with jittering
ggplot(data = mpg) +
  geom_smooth(mapping = aes(x = displ, y = hwy)) +
  geom_jitter(mapping = aes(x = displ, y = hwy))
```

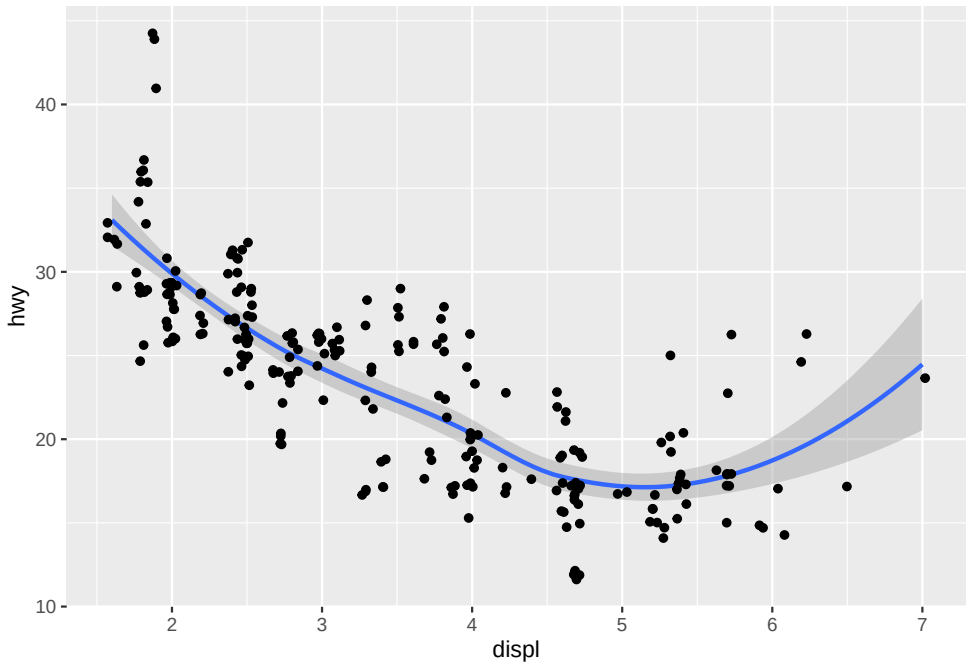
```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```




**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()
```

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



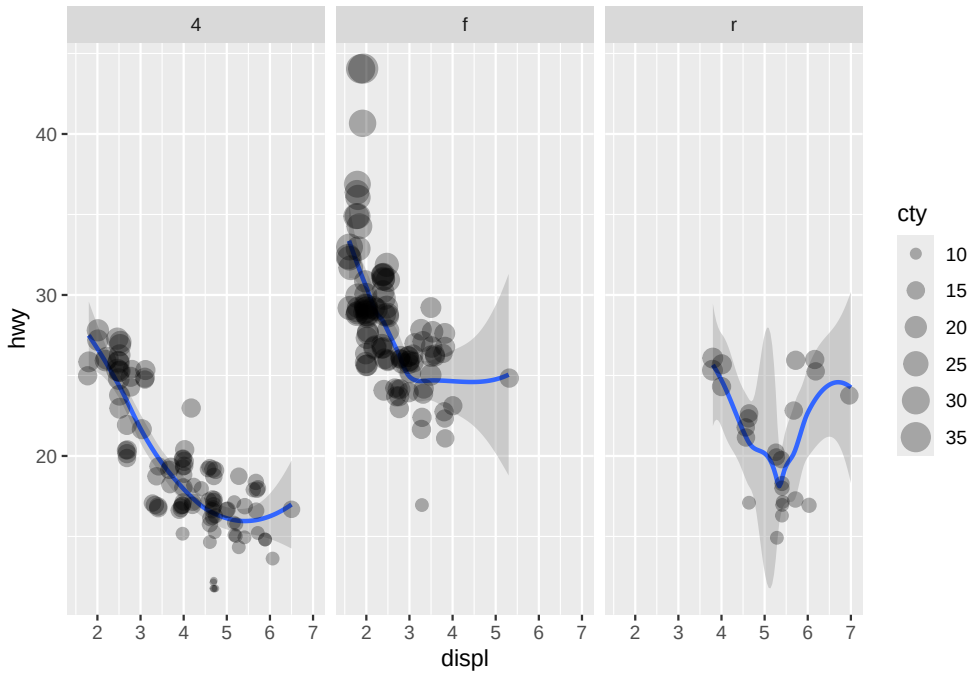

Aesthetics for `geom_smooth`

- `group` data points by values in a `categorical` variable, and do stats separately to generate separate curves
- `linetype`, similar to `group` and changes the type of line used
- `colour`, similar to `group` and changes the colour of line used

**Caution**: Could end up with too much to handle

```
# comment / uncomment the appropriate lines to test the corresponding aesthetics
ggplot(data=mpg, mapping = aes(x=displ, y=hwy)) +
  #geom_smooth(mapping = aes(group = drv)) #+
  geom_smooth() +
  geom_jitter(aes(size = cty), alpha = 0.3) +
  facet_wrap(~drv)
```

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




```
  #facet_grid(drv~.)
```

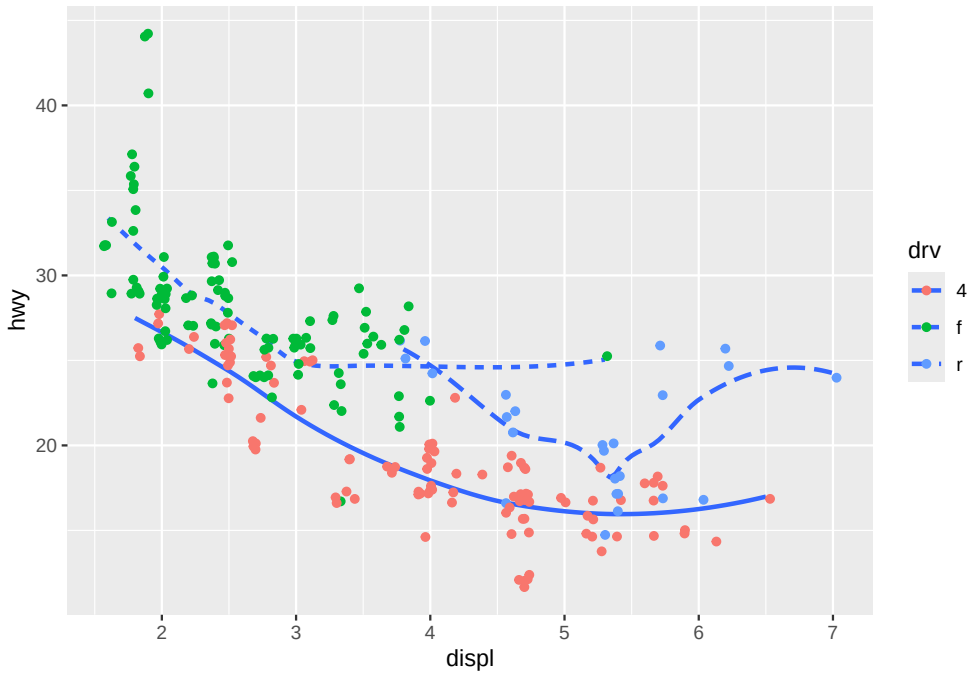
If only want the smooth line (without the shades) can use `se=FALSE`

- this gets rid of the shades around the curve, which represents the uncertainty of the estimates
- more precisely, it relates to the notion of `confidence interval` – which will be in your stats course

Use `?geom_smooth` in `Console` to see more details.

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

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


hwy
40
30
20
2
3
4
5
6
7
displ
drv
4
f
r