STAT 1261/2260: Principles of Data Science

Lecture 5 - Data Visualization (2): ggplot2 (1/3)

Where are we?

- *What is Data Science?*
- *How do we learn Data Science?*
- *Data visualization: What is a good graphic?*
- *Composing/Dissecting Data Graphics*
- Implementing the grammar of graphics using ggplot2

Prerequisites: Loading packages

The textbook offers the mdsr package for R, which contains all of the data sets referenced in this book.

- In particular, loading `mdsr` also loads the `mosaic` package, which in turn loads `dplyr` and `ggplot2`.
- The `mosaic` package includes data sets and utilities from Project MOSAIC (http://mosaic-web.org) that are used to teach mathematics, statistics, computation and modeling.
- Packages `dplyr` and `ggplot2` are also part of `tidyverse`.
- The `tidyverse` (https://www.tidyverse.org/) is a collection of R packages designed for data science, managed by a group of people including Hadley Wickham, statistician and chief scientist at RStudio, Inc.

```
library(mdsr)
library(tidyverse)
```

ggplot2

`ggplot2` is an `R` package which implements the grammar of graphics, a coherent system for describing and building graphs.

- The `ggplot2` package is the primary tool of data visualization in the book "The Grammar of Graphics" by Leland Wilkinson, now chief scientist at h2o, Inc.
- The four elements of graphics identified by Yau (Visual Cues, Coordinate System, Scale and Context) are also found in the grammar of graphics, albeit by different terms. Thus, it is essential to understand the taxonomy of graphics in order to use `ggplot2`.

ggplot2::mpg data

We will follow the examples in *R for data science*. Let's first look at the data set.

- `mpg` contains observations collected by the US Environment Protection Agency on 38 models of car.
- `mpg` is a `tibble`, which is a simplified `data.frame`, modified for better handling large data. For now, it is okay to think a `tibble` as a `data.frame`.

```
class(mpg)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

ggplot2::mpg data

```
head(mpg[,1:6],3)
```

```
## # A tibble: 3 x 6
##   manufacturer model displ  year   cyl trans
##   <chr>        <chr> <dbl> <int> <int> <chr>
## 1 audi         a4      1.8  1999     4 auto(l5)
## 2 audi         a4      1.8  1999     4 manual(m5)
## 3 audi         a4      2    2008     4 manual(m6)
```

```
head(mpg[,7:11],3)
```

```
## # A tibble: 3 x 5
##   drv     cty   hwy fl    class
##   <chr> <int> <int> <chr> <chr>
## 1 f        18    29 p     compact
## 2 f        21    29 p     compact
## 3 f        20    31 p     compact
```

`ggplot2::mpg` data

A data frame with 234 rows and 11 variables

1. manufacturer
2. model – model name
3. displ – engine displacement, in litres
4. year – year of manufacture
5. cyl – number of cylinders
6. trans – type of transmission
7. drv: f = front-wheel drive, r = rear wheel drive, 4 = 4wd
8. cty – city miles per gallon
9. hwy – highway miles per gallon
10. fl – fuel type
11. class – "type" of car

ggplot2::mpg data example (cont.)

Question: do cars with big engines use more fuel than cars with small engines?

Let's use graphs to answer this question.

Among the variables in `mpg` are:

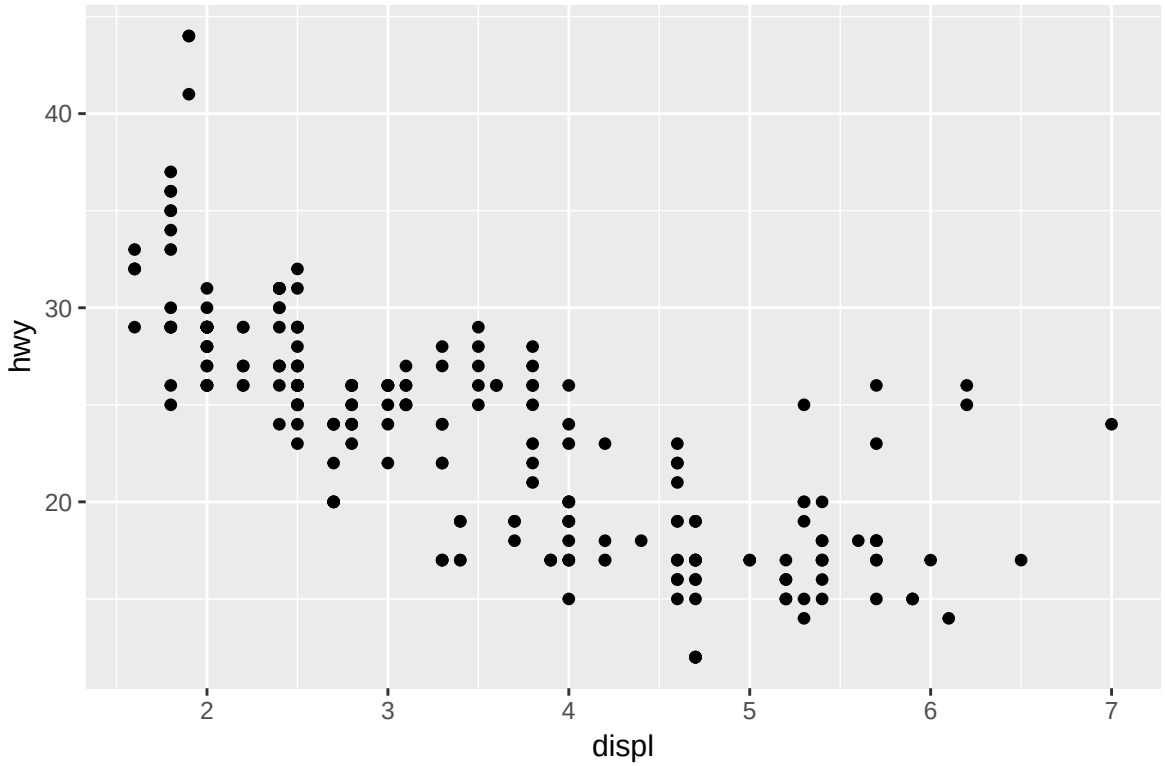
1. `displ`: measures a car's engine size, in litres.
2. `hwy`: measures a car's fuel efficiency on the highway, in miles per gallon (mpg).

   A car with a low fuel efficiency consumes more fuel than a car with a high fuel efficiency when they travel the same distance.

To learn more about `mpg`, open its help page by running `?mpg`.

Creating a scatterplot using ggplot

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



The plot shows a negative relationship between engine size (`displ`) and fuel efficiency (`hwy`).

Creating a scatterplot using ggplot (cont.)

Let's analyze the code.

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

With ggplot2, you begin a plot with the function `ggplot()`.

- `ggplot()` creates a coordinate system that you can add layers to.
- The first argument of `ggplot()` is the dataset to use in the graph.
- You complete your graph by adding one or more layers to `ggplot()`.
  - The function `geom_point()` adds a layer of points to your plot, which creates a scatterplot.

Creating a scatterplot using ggplot (cont.)

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

- **geom** stands for "geometric object."
- `ggplot2` comes with many geom functions that each add a different type of layer to a plot. For example, `geom_smooth`, `geom_bar`, `geom_polygon`, etc.
- Each geom function in ggplot2 takes a `mapping` argument. This defines how variables in your dataset are mapped to visual properties.
  - The `mapping` argument is always paired with `aes()`, and the `x` and `y` arguments of `aes()` specify which variables to map to the x and y axes.
  - `ggplot2` looks for the mapped variable in the data argument, in this case, `mpg`.

A graphing template

The code in the last slide can be turned into a reusable template for making graphs with `ggplot2`. To make a graph, replace the bracketed sections in the code below with a dataset, a geom function, or a collection of mappings.

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

In connection to the four elements of data graphics,

1. `ggplot()` (by default) sets the coordinate system as the *Cartesian coordinate system*;
2. Visual cue used is the *position*, set by `mapping = aes(x = ..., y = ...)`, paired with the use of `geom_point()`;
3. *scale* is automatically chosen as appropriate as possible;
4. *context* is (minimally) given by the axis labels.

Adding more variables via visual cues

In the plot we just created, one group of points seems fall outside of the linear trend.

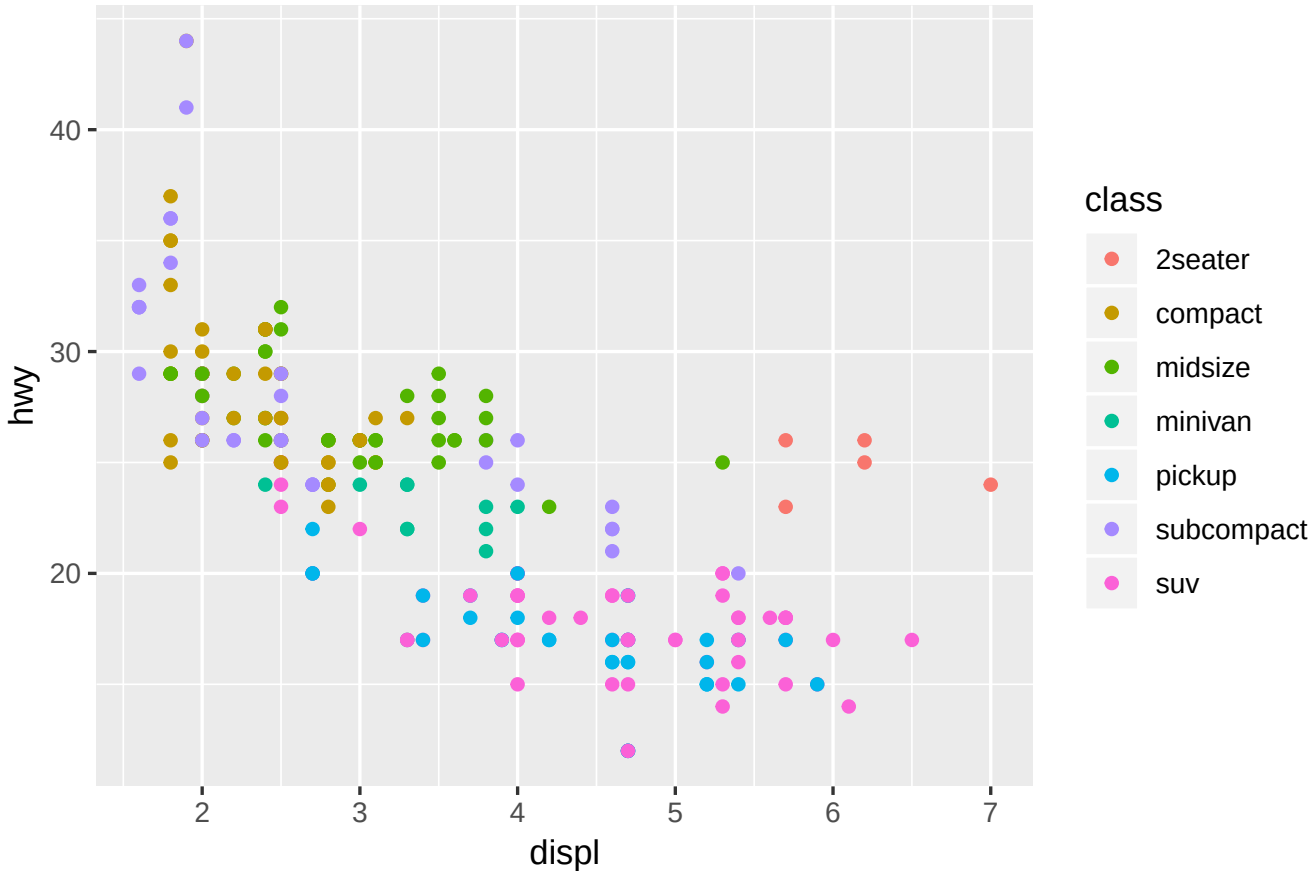
How can you explain these cars? Maybe some other variable(s) also affects fuel efficiency.

You can add a third variable, like `class`, to a two dimensional scatterplot by mapping an aesthetic (eg. color) to it.

- An aesthetic is a visual property of the objects in your plot.
- Aesthetics include things like the size, the shape, or the color of your points.

Adding more variables via visual cues (cont.)

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



Adding more variables via visual cues (cont.)

Try mapping the `class` variable using the visual cues `size`, `shape`, or `alpha` (transparency), `fill` (with set `shape = 22`) .

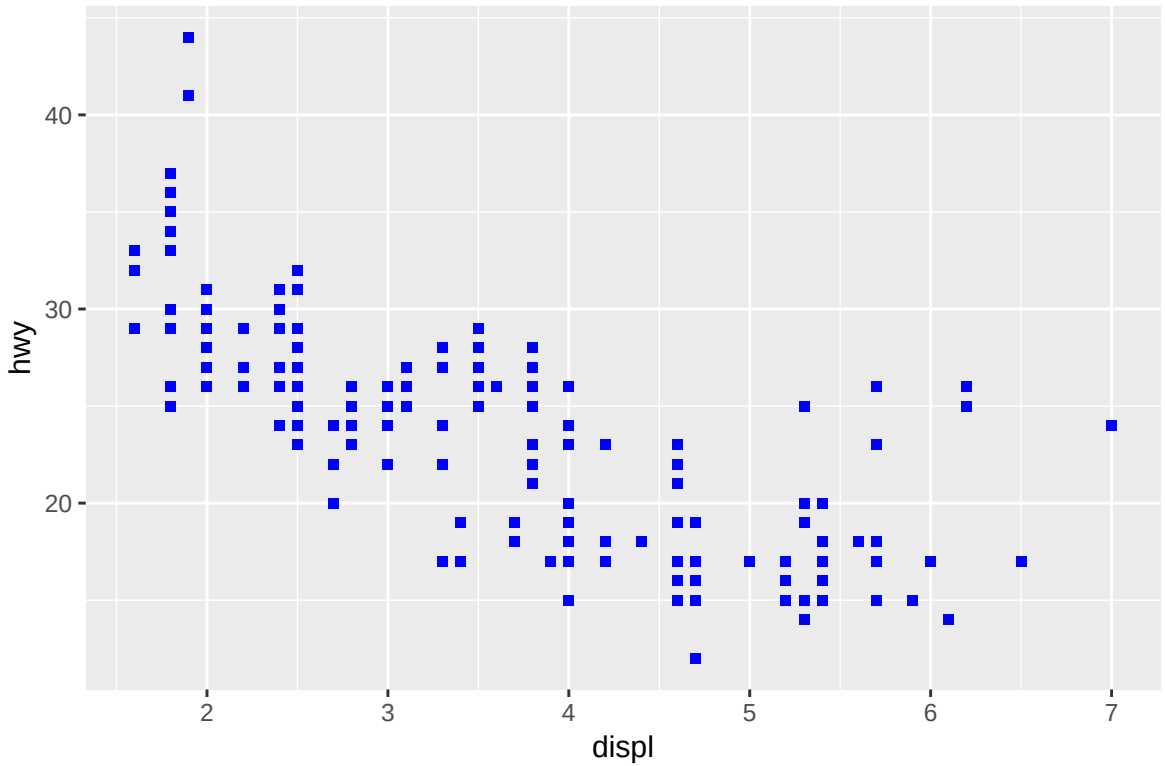
```
g <- ggplot(data = mpg, mapping = aes(x = displ, y = hwy))
g + geom_point(mapping = aes(size = class))
g + geom_point(mapping = aes(shape = class))
g + geom_point(mapping = aes(alpha = class))
g + geom_point(mapping = aes(fill = class), shape = 22)
```

Caution: `ggplot2` will only use six shapes at a time. By default, additional groups will go unplotted when you use this aesthetic.

Adding more variables via visual cues (cont.)

You can also *set* the aesthetic properties of your geom manually. For example, we can make all of the points in our plot blue with square shape:

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy),
             color = "blue", shape = 15)
```



Adding more variables via Facets

Another way to add additional variables to your plots is to split your plot into facets, subplots that each displays one subset of the data.

This is particularly useful for categorical variables,

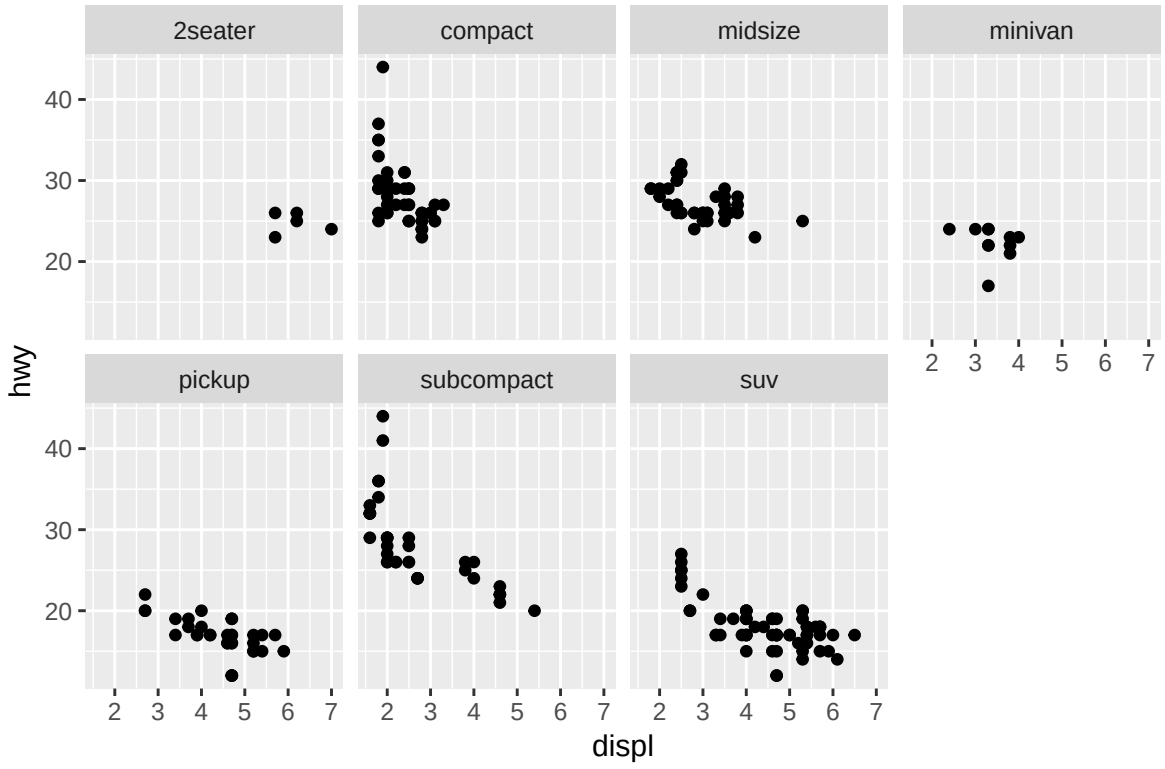
To facet your plot by a single variable, use `facet_wrap()`.

- The first argument of `facet_wrap()` should be a formula, which you create with ~ followed by a variable name.
- The variable that you pass to `facet_wrap()` should be discrete.

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

Facets by one variable

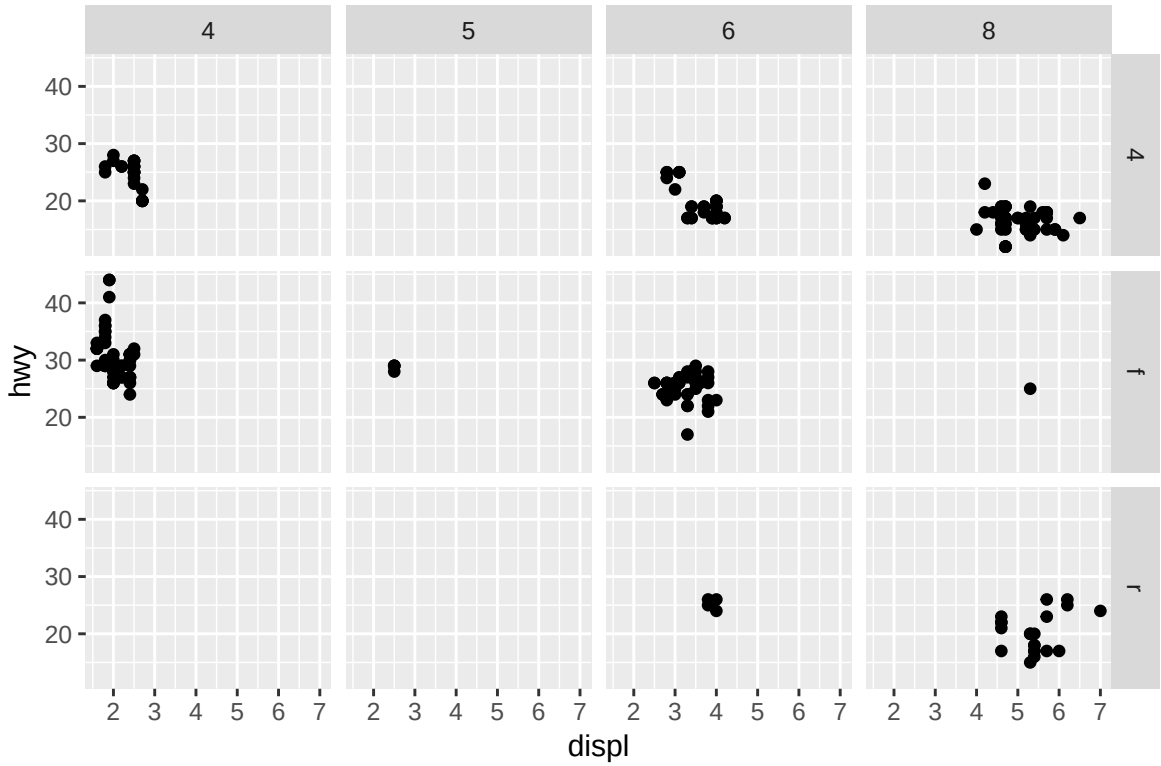
```
g <- ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
g + facet_wrap(~ class, nrow = 2)
```



Facets by two variables

To facet your plot on the combination of two variables, add `facet_grid()` to your plot call. The first argument of `facet_grid()` is also a formula. This time the formula should contain two variable names separated by `~`.

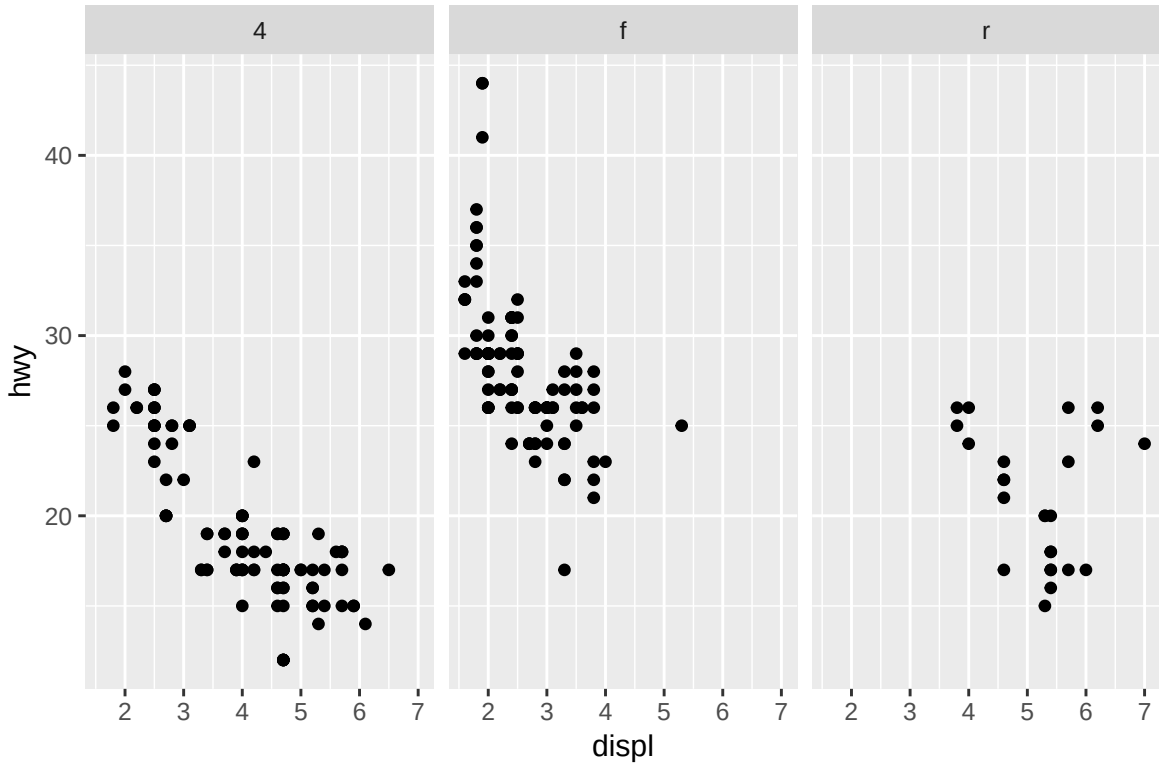
```
g + facet_grid(drv ~ cyl)
```



Facet into columns

We can also use facet_grid() to facet into columns based on `drv`

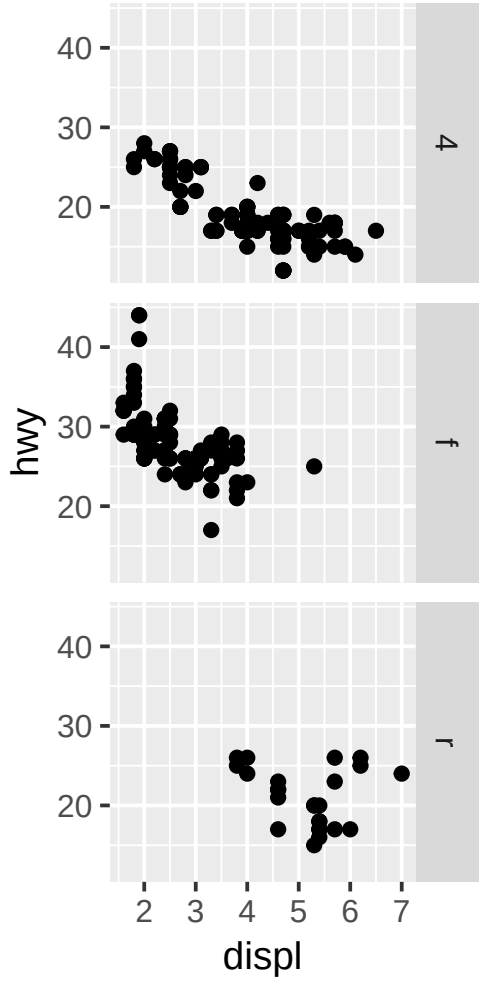
```
g + facet_grid(. ~ drv)
```



Facet into rows

Similarly, we can use facet_grid() to facet into rows based on `drv`

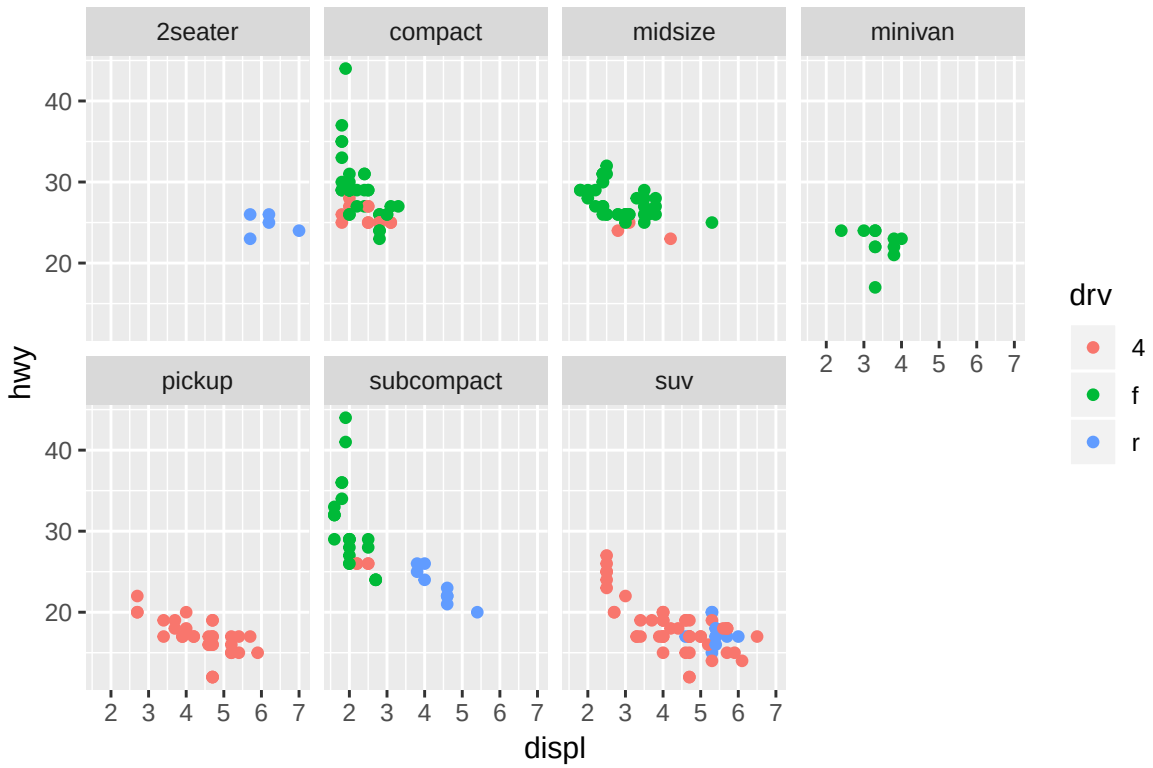
```
g + facet_grid(drv ~ . )
```



ggplot2::mpg data example continued

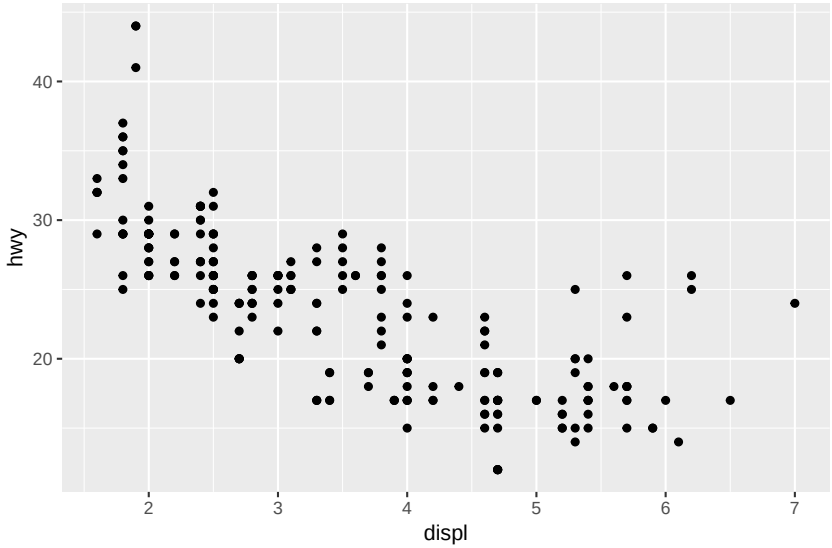
Add one more variable `drv` and map color to it.

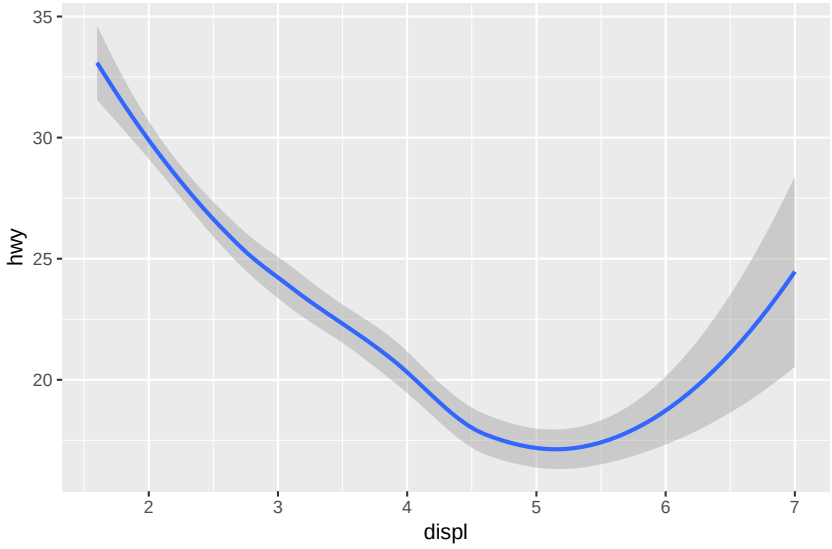
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy,
                           color = drv)) +
  facet_wrap(~ class, nrow = 2)
```



Geometric objects (1)

What is the difference between the following two plots?





They use different geoms!

Geometric objects (1)(cont.)

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

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

Geometric objects (2)

People often describe plots by the type of geom that the plot uses. For example:

- bar charts use bar geoms
- line charts use line geoms
- boxplots use boxplot geoms
- scatterplots use the point geom

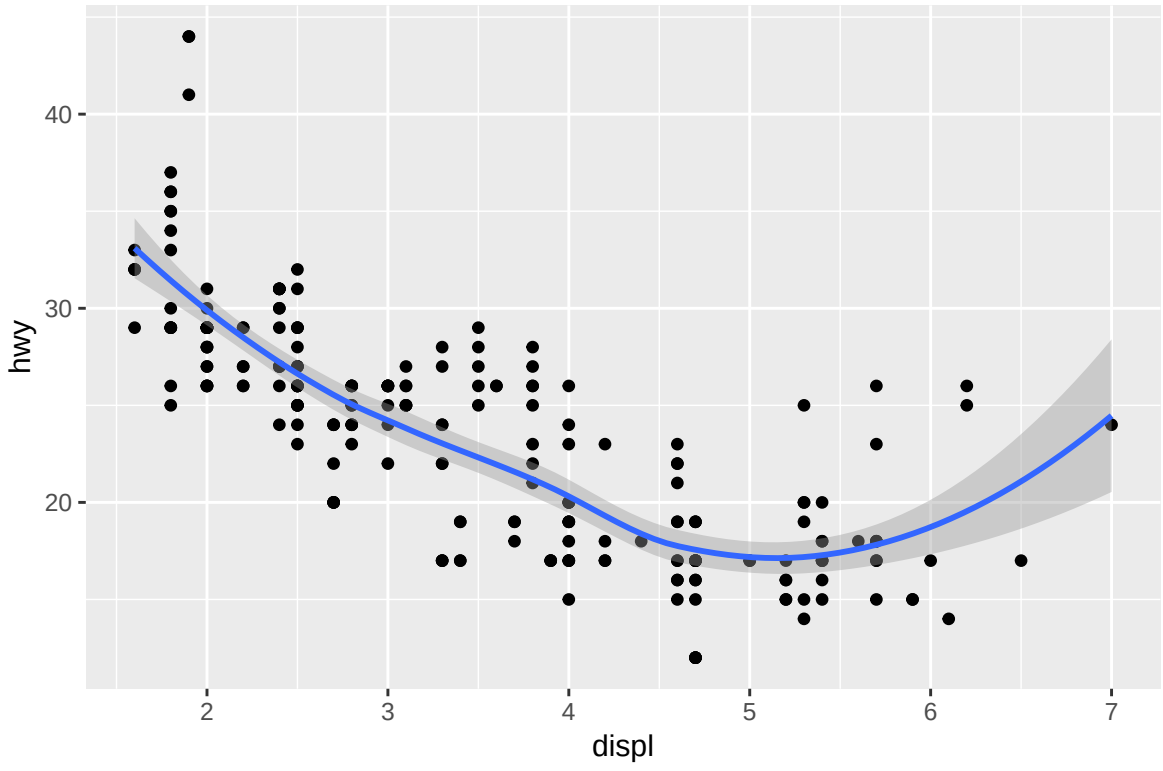
`ggplot2` provides over 30 geoms, and extension packages provides even more.

The best way to get a comprehensive overview is the ggplot2 cheatsheet.

Geometric objects (3)

To display multiple geoms in the same plot, add multiple geom functions to `ggplot()`:

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



Geometric objects (4)

If you place mappings in a geom function, ggplot2 will treat them as local mappings for the layer.

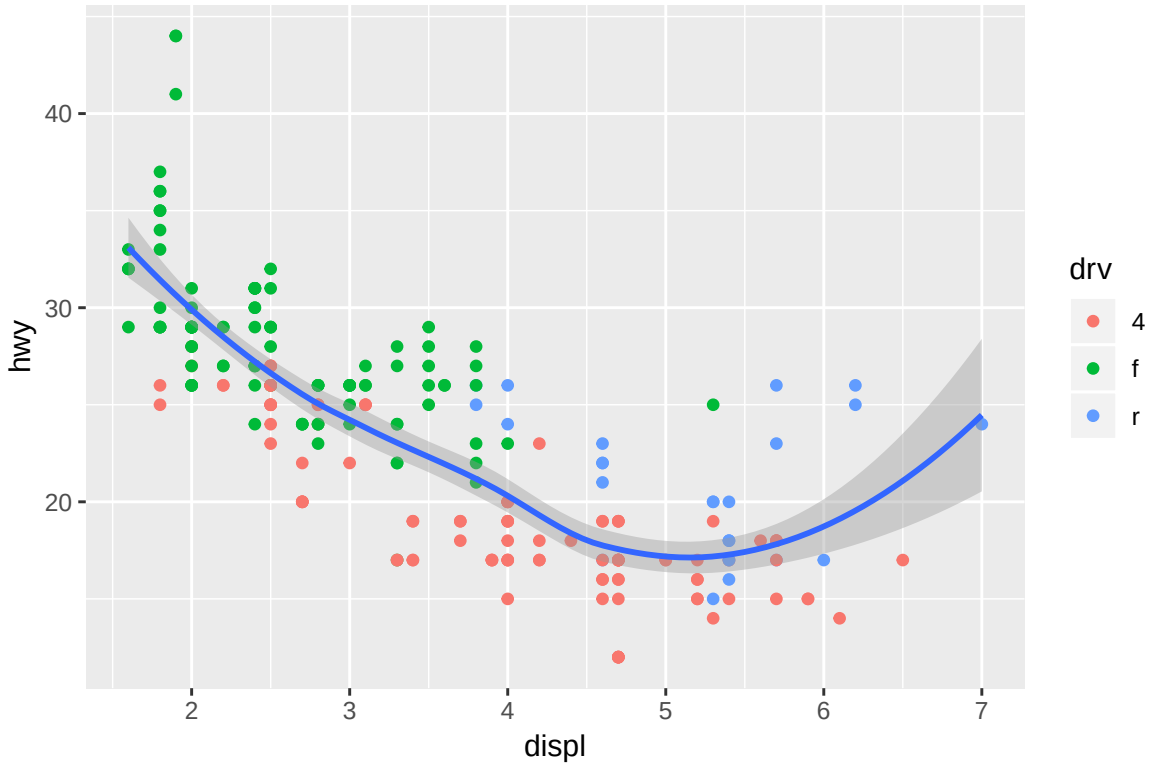
It will use these mappings to extend or overwrite the global mappings for that layer only.

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 = drv)) +
  geom_smooth()
```

Geometric objects (4)

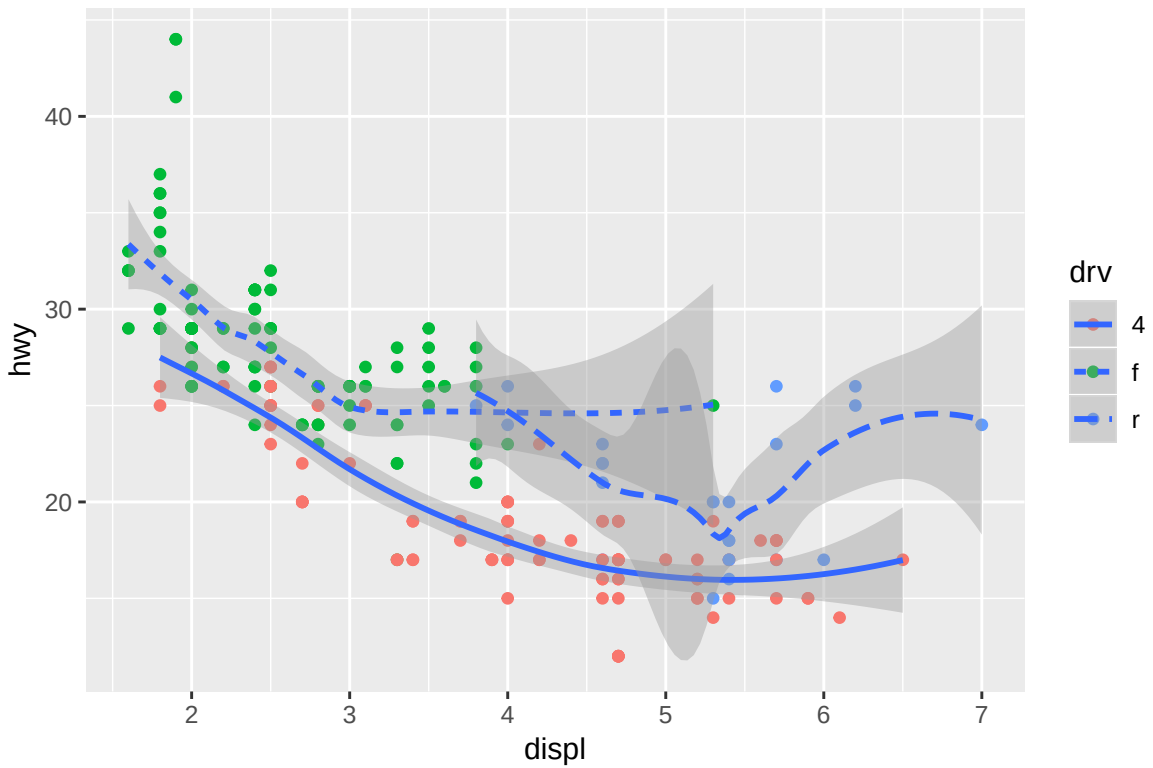
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(mapping = aes(color = drv)) +
  geom_smooth()
```



Geometric objects (5)

Every geom function in ggplot2 takes a `mapping` argument. In the following code, `geom_smooth()` will draw a different line for each unique value of the variable that you map to linetype.

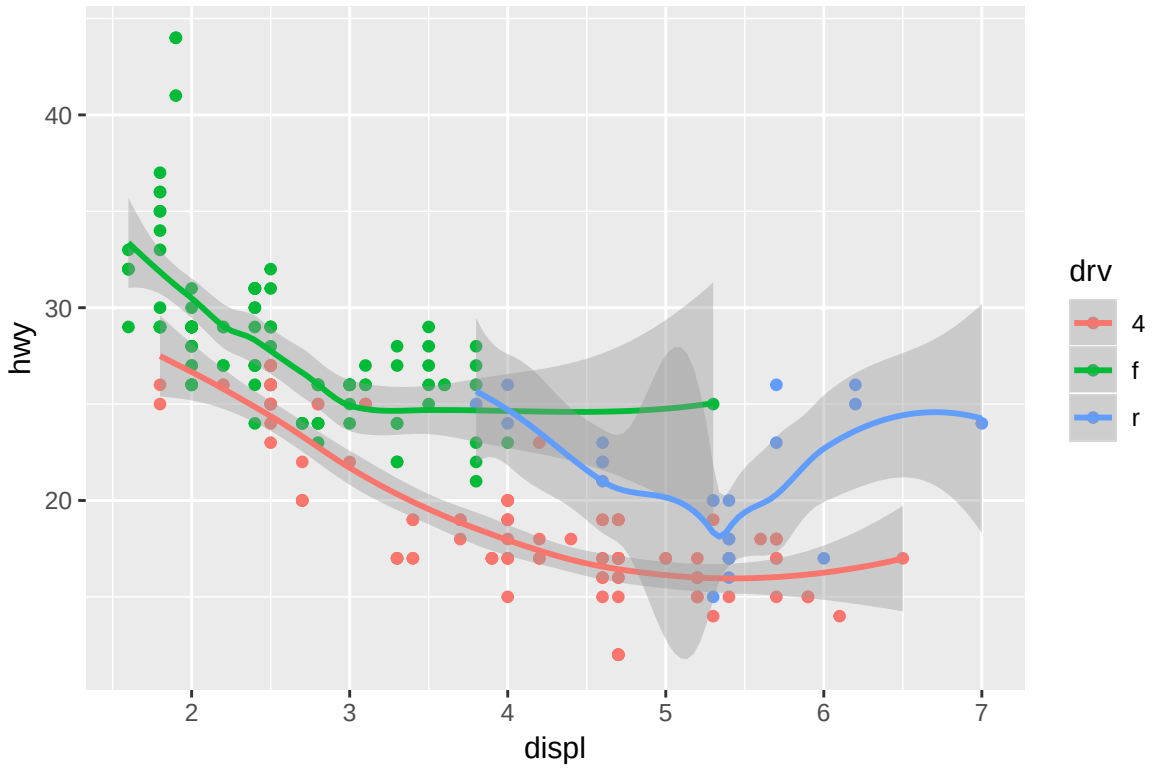
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(mapping = aes(color = drv)) +
  geom_smooth(mapping = aes(linetype = drv))
```



Geometric objects (6)

You may also map `color` to `drv` globally in `ggplot()`.

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



Geometric objects (geom)

Think about this:

1. What geom would you use to draw a line chart? A boxplot? A histogram? An area plot?
2. What aesthetics can you use to each geom?

To get answers, `Help > Cheatsheets > Data Visualization with ggplot2`

ggplot2 continues in next lecture.