Lecture 4: Data Visualization

The Grammar of Graphics: ggplot2

Yujin Jeong

STATS 195

ggplot2

Tidyverse package

First, let's install the tidyverse package

```
install.packages("tidyverse")
library(tidyverse)
```

You only need to install a package once, but you need to reload it every time you start a new session.

The "mpg" data frame

We will work with the "mpg" data frame found in ggplot2. The "mpg" contains observations collected by the US Environmental Protection Agency on 38 models of car.

```
mpg
```

```
#> # A tibble: 234 × 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(l5)   f        18    29 p     compa…
#> 2 audi         a4      1.8  1999     4 manual(m5) f        21    29 p     compa…
#> 3 audi         a4      2    2008     4 manual(m6) f        20    31 p     compa…
#> 4 audi         a4      2    2008     4 auto(av)   f        21    30 p     compa…
#> 5 audi         a4      2.8  1999     6 auto(l5)   f        16    26 p     compa…
#> 6 audi         a4      2.8  1999     6 manual(m5) f        18    26 p     compa…
#> # … with 228 more rows
```

Among the variables in mpg are:

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

The "mpg" data frame

We will work with the "mpg" data frame found in ggplot2. The "mpg" contains observations collected by the US Environmental Protection Agency on 38 models of car.

```
mpg
```

```
#> # A tibble: 234 × 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(l5)   f        18    29 p     compa…
#> 2 audi         a4      1.8  1999     4 manual(m5) f        21    29 p     compa…
#> 3 audi         a4      2    2008     4 manual(m6) f        20    31 p     compa…
#> 4 audi         a4      2    2008     4 auto(av)   f        21    30 p     compa…
#> 5 audi         a4      2.8  1999     6 auto(l5)   f        16    26 p     compa…
#> 6 audi         a4      2.8  1999     6 manual(m5) f        18    26 p     compa…
#> # … with 228 more rows
```

From the data set, we can ask questions such as:

- Do cars with big engines use more fuel than cars with small engines?
- What does the relationship between engine size and fuel efficiency look like?

ggplot2

The function `ggplot()` initializes a basic graph structure. It cannot produce a plot alone by itself. You need to add extra components to generate a graph.

Building blocks of a **ggplot2** graphical objects:

- data
- aesthetic mapping
- geometric objects
- statistical transformations
- facets
- scales
- coordinate system
- positioning adjustments

```
ggplot(data = <DATA>) +
  GEOM_FUNCTION(
    mapping = aes(<mappings>),
    stat = <statistic transformation>,
    position = <potision options>,
    color = <fixed color>,
    <other arguments>) +
  FACET_FUNCTION(<facet options>) +
  SCALE_FUNCTION(<scale options>) +
  theme(<theme elements>)
```

ggplot2: Data and Aesthetic Mapping

The function `ggplot()` initializes a graph.

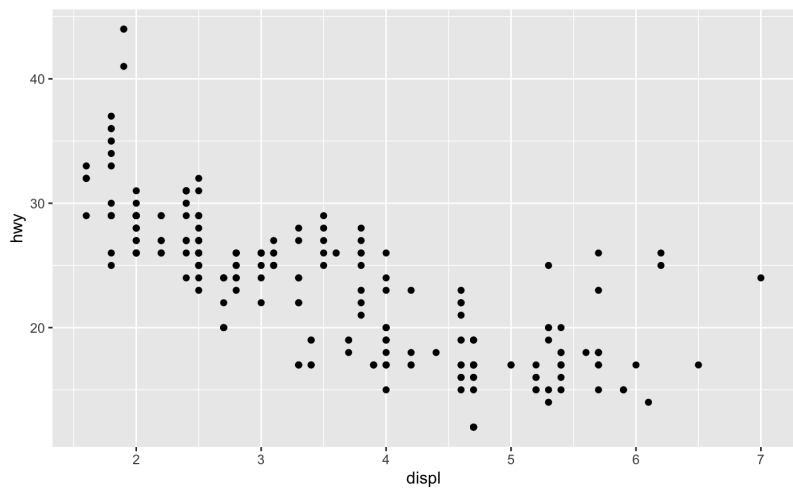
```
ggplot(data = mpg)
```

ggplot2: Data and Aesthetic Mapping

The aesthetic mapping tells ggplot which features to use for what.

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

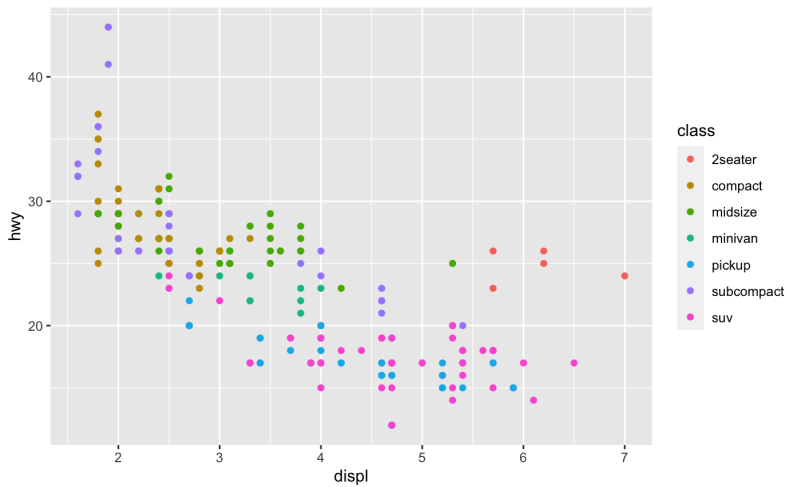
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```



ggplot2: Data and Aesthetic Mapping

You can convey information about your data by mapping the aesthetics in your plot to the variables in the dataset. For example, you can map the colors of points to the class variable to reveal the class of each car.

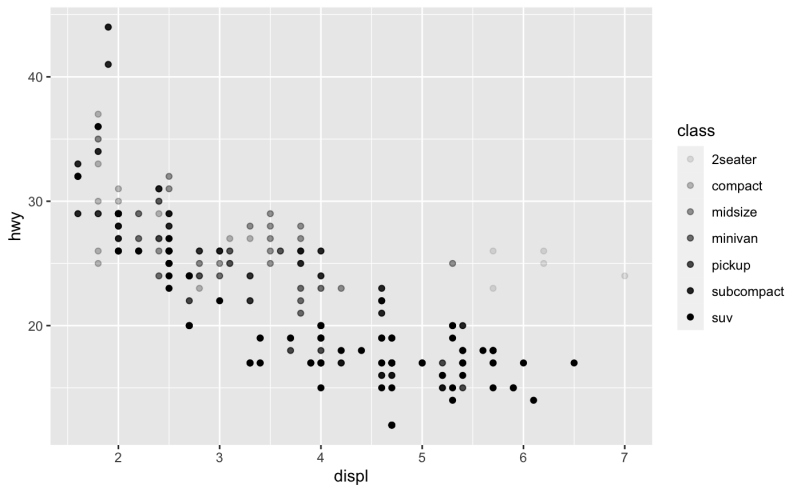
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = class))
```



ggplot2: Data and Aesthetic Mapping

We can also map class to the alpha aesthetic, which controls the transparency of the points, or to the shape aesthetic, which controls the shape of the points.

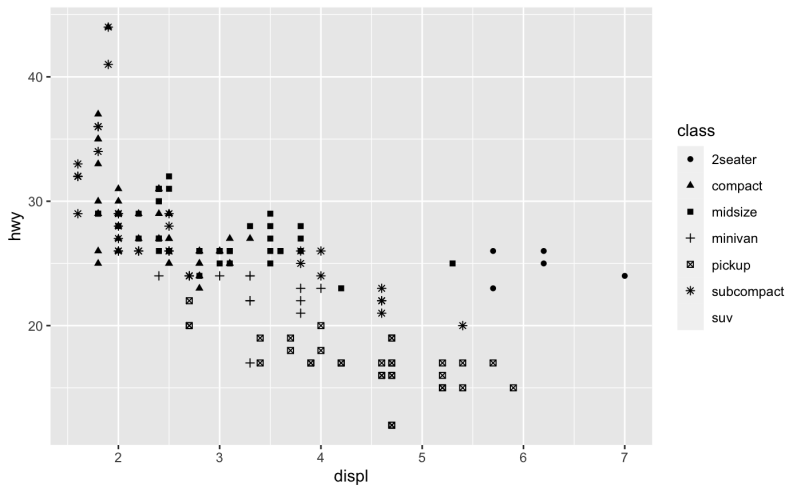
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, alpha = class))
```



ggplot2: Data and Aesthetic Mapping

We can also map class to the alpha aesthetic, which controls the transparency of the points, or to the shape aesthetic, which controls the shape of the points.

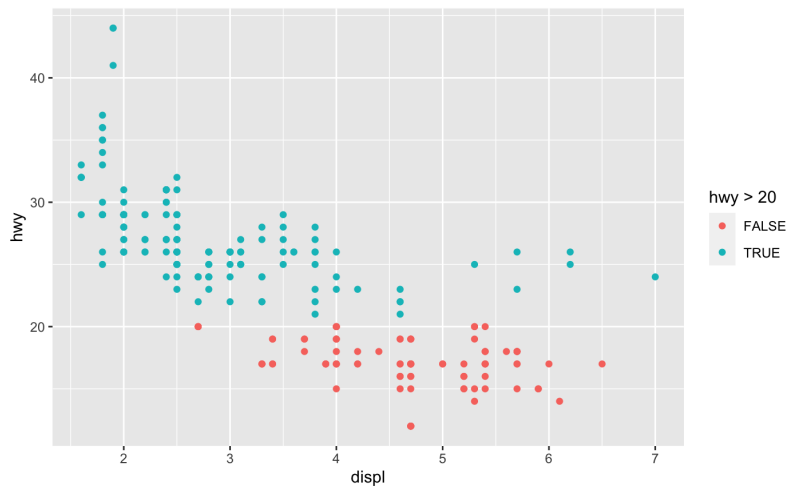
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, shape = class))
```



ggplot2: Data and Aesthetic Mapping

If an aesthetic is linked to data it is put into `aes()`.

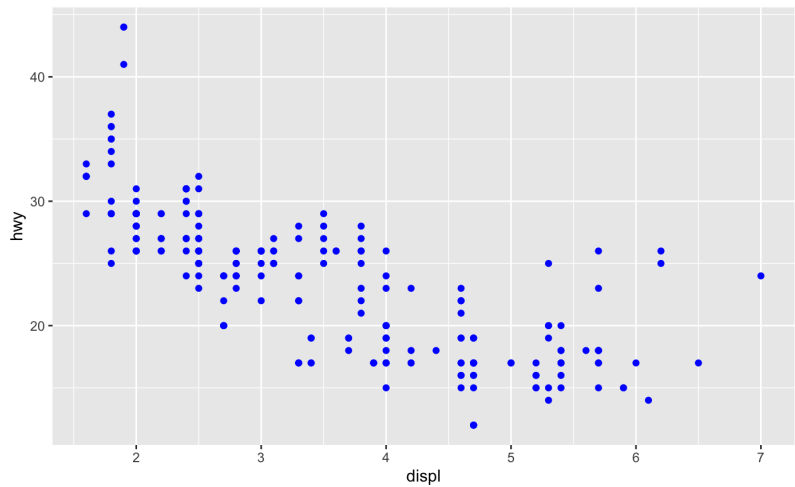
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = hwy > 20))
```



ggplot2: Data and Aesthetic Mapping

You can also set the aesthetic properties of your geom manually. For example, we can make all of the points blue. Note that as the color is simply set to a value, it is put outside of `aes()`.

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

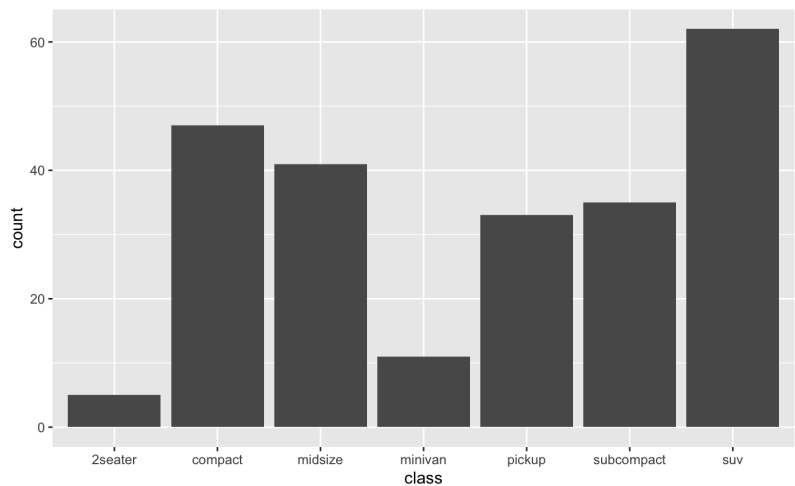


ggplot2: Geometric objects

A **geom** is the geometrical object that a plot uses to represent data. For example, bar charts use bar geoms, line charts use line geoms, boxplots use boxplot geoms, and scatterplots use point geoms.

To change the geom in your plot, change the geom function that you add to ggplot().

```
ggplot(data = mpg) +
  geom_bar(mapping = aes(x = class))
```

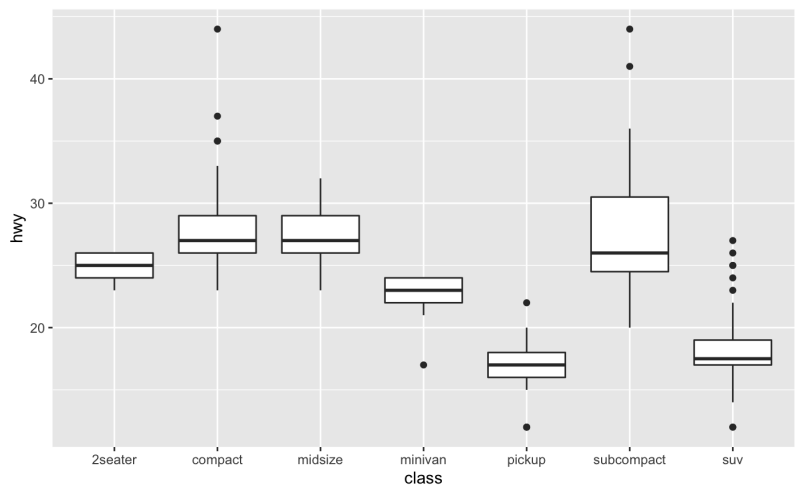


ggplot2: Geometric objects

A **geom** is the geometrical object that a plot uses to represent data. For example, bar charts use bar geoms, line charts use line geoms, boxplots use boxplot geoms, and scatterplots use point geoms.

To change the geom in your plot, change the geom function that you add to ggplot().

```
ggplot(data = mpg) +
  geom_boxplot(mapping = aes(x = class, y = hwy))
```

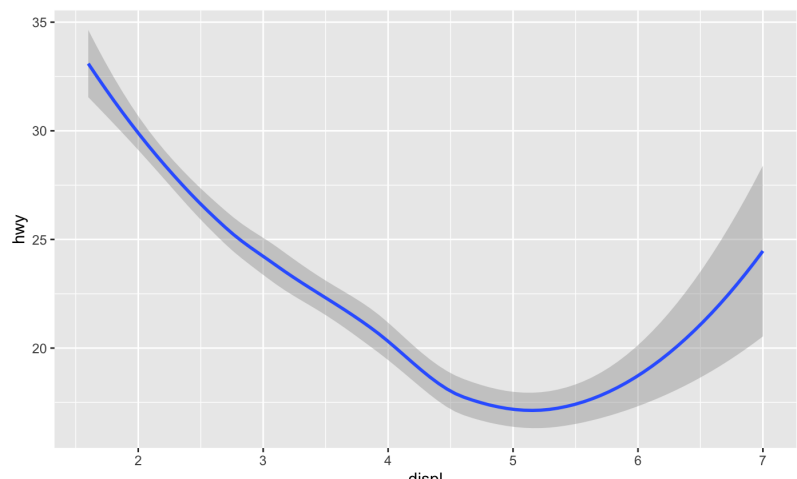


ggplot2: Geometric objects

A **geom** is the geometrical object that a plot uses to represent data. For example, bar charts use bar geoms, line charts use line geoms, boxplots use boxplot geoms, and scatterplots use point geoms.

To change the geom in your plot, change the geom function that you add to ggplot().

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

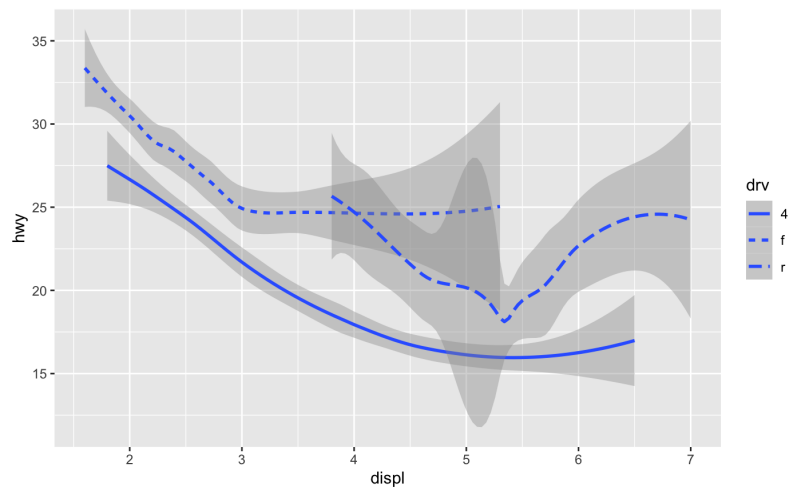


ggplot2: Geometric objects

Every geom function in ggplot2 takes a mapping argument.

The geom_smooth() function will draw a different line, with a different linetype, for each unique value of the variable that you map to linetype.

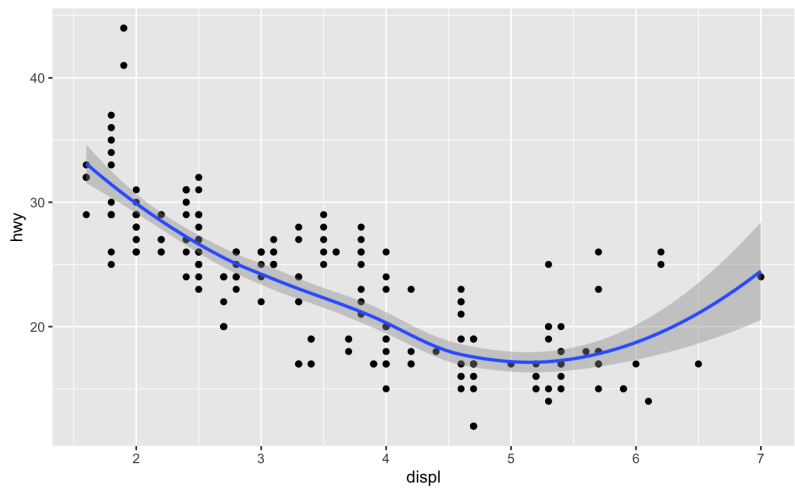
```
ggplot(data = mpg) +
  geom_smooth(mapping = aes(x = displ, y = hwy, linetype = drv))
```



ggplot2: Geometric objects

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

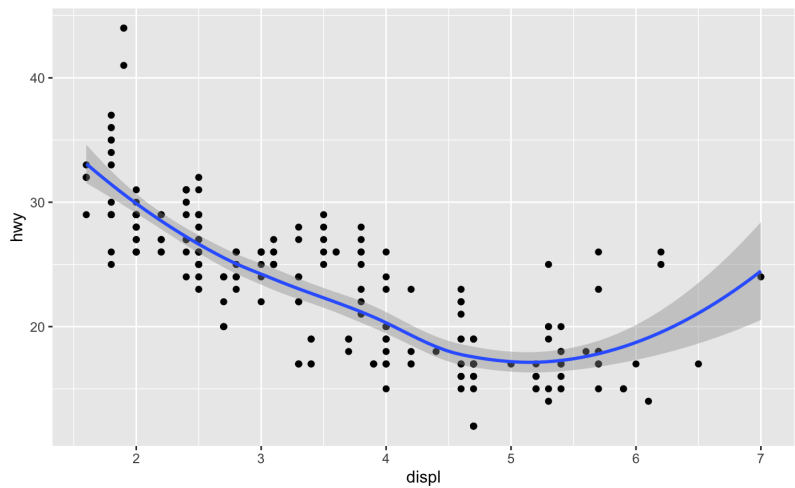
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  geom_smooth(mapping = aes(x = displ, y = hwy))
```



ggplot2: Geometric objects

Note that data and mapping can be given both as global (in ggplot()) or per layer. For example, the code below produces the same plot as the previous code.

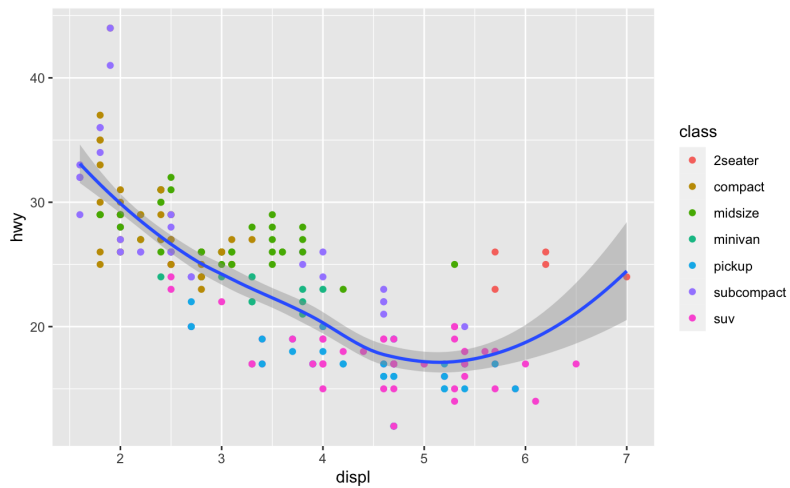
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth()
```



ggplot2: Geometric objects

We can still display different aesthetics in different layers.

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



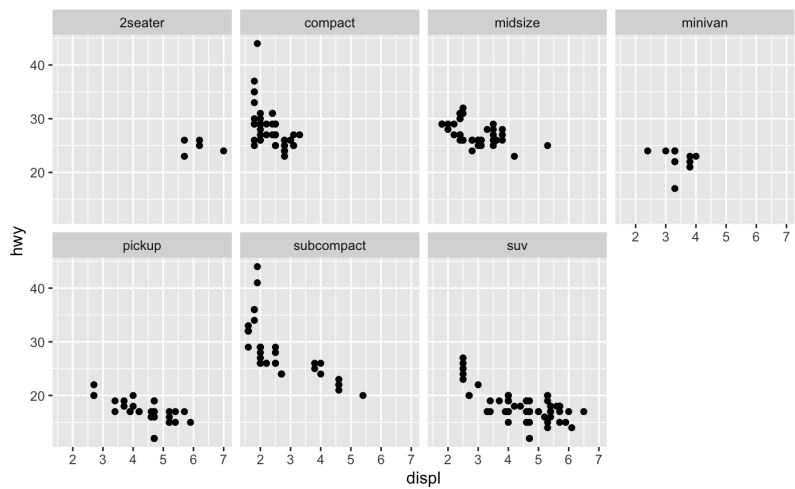
ggplot2: Facets

One way to add additional variables is with aesthetics. Another way, particularly useful for categorical variables, is to split your plot into **facets**, subplots that each display one subset of the data.

ggplot2: Facets

To facet your plot by a single variable, use facet_wrap(). The first argument should be a formula, which you create with ~ followed by a variable name. The variable that you pass to facet_wrap() should be discrete.

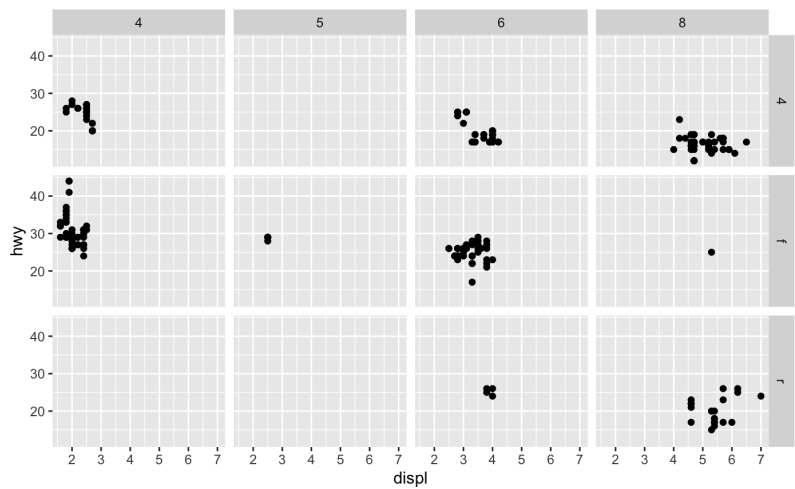
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_wrap(~ class, nrow = 2)
```



ggplot2: Facets

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

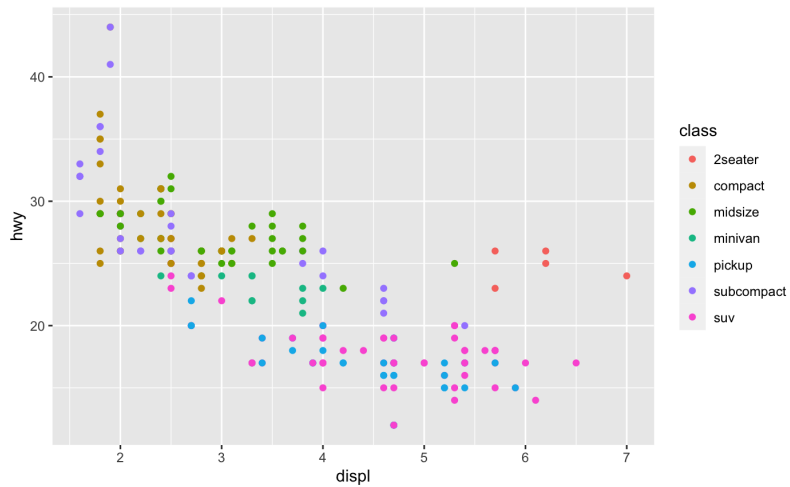
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_grid(drv ~ cyl)
```



ggplot2: Scales

Scales define how the mapping you specify inside `aes()` should happen. All mappings have an associated scale even if not specified.

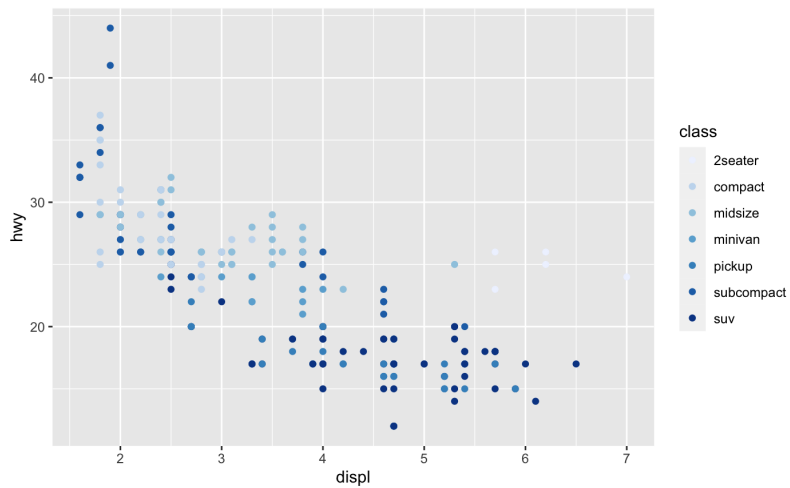
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = class))
```



ggplot2: Scales

Scales define how the mapping you specify inside `aes()` should happen.

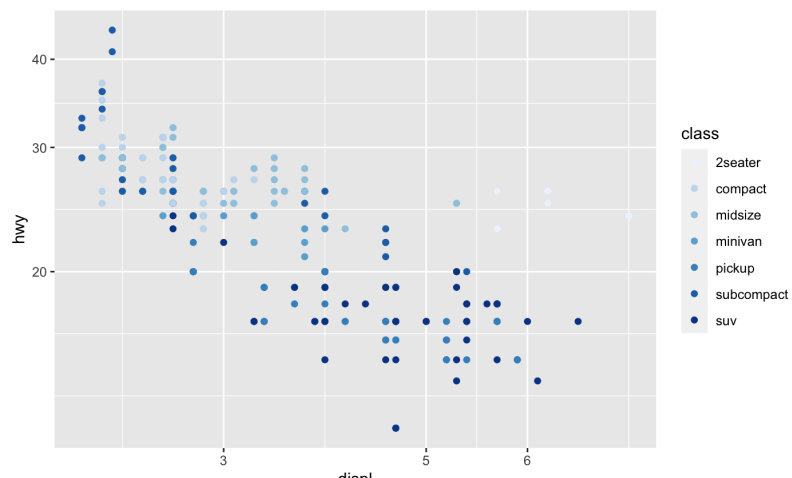
```
ggplot(mpg) +
  geom_point(aes(x = displ, y = hwy, colour = class)) +
  scale_colour_brewer(palette = 1)
```



ggplot2: Scales

Positional mappings (x and y) also have associated scales.

```
ggplot(mpg) +
  geom_point(aes(x = displ, y = hwy, colour = class)) +
  scale_colour_brewer(palette = 1) +
  scale_x_continuous(breaks = c(3, 5, 6)) +
  scale_y_continuous(trans = 'log10')
```

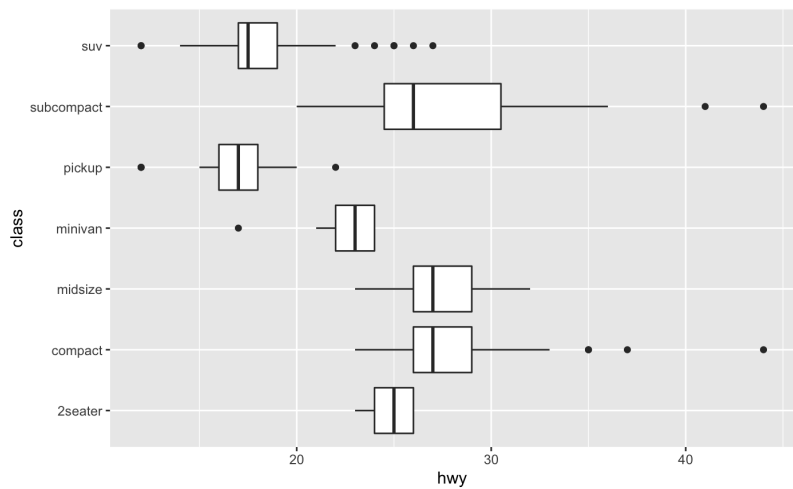


ggplot2: Coordinate System

The coordinate system is the fabric you draw your layers on in the end. The default `coord_cartesion` provides the standard rectangular x-y coordinate system.

For example, the `coord_flip()` function switches the x and y axes.

```
ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +
  geom_boxplot() +
  coord_flip()
```

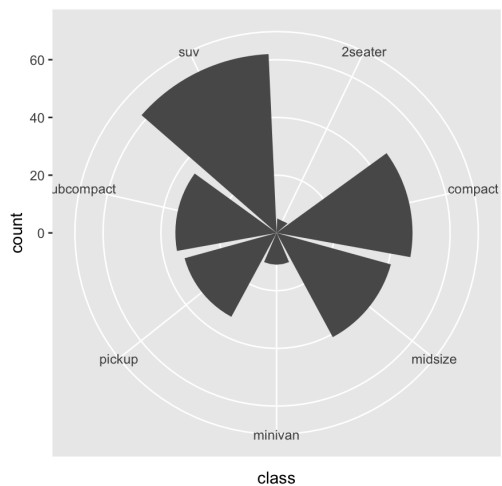


ggplot2: Coordinate System

The coordinate system is the fabric you draw your layers on in the end. The default `coord_cartesion` provides the standard rectangular x-y coordinate system.

For example, the `coord_polar()` function uses polar coordinates.

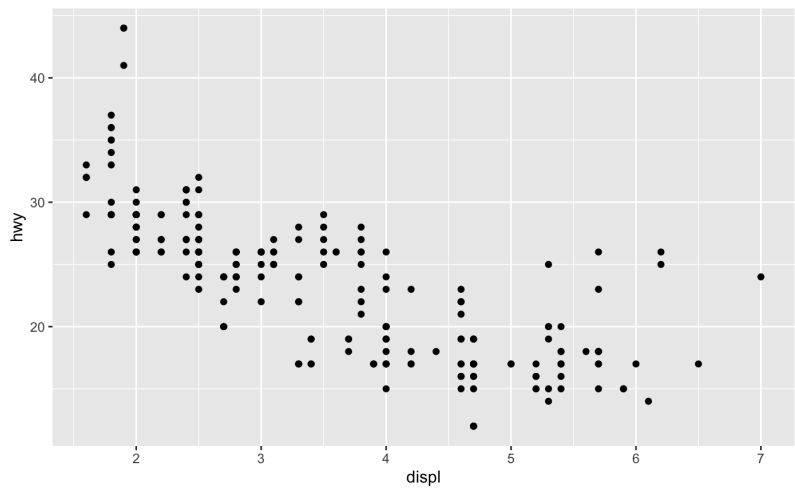
```
ggplot(mpg) +
  geom_bar(aes(x = class)) +
  coord_polar()
```



ggplot2: Positioning Adjustments

Note that the plot displays only 126 points, even though there are 234 observations in the dataset. This is because the values of hwy and displ are rounded so the points appear on a grid and many points overlap each other.

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



ggplot2: Positioning Adjustments

You can avoid this gridding by setting the position adjustment to “jitter”. position = "jitter" adds a small amount of random noise to each point.

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