

Plotting

Introduction to Data Science with R

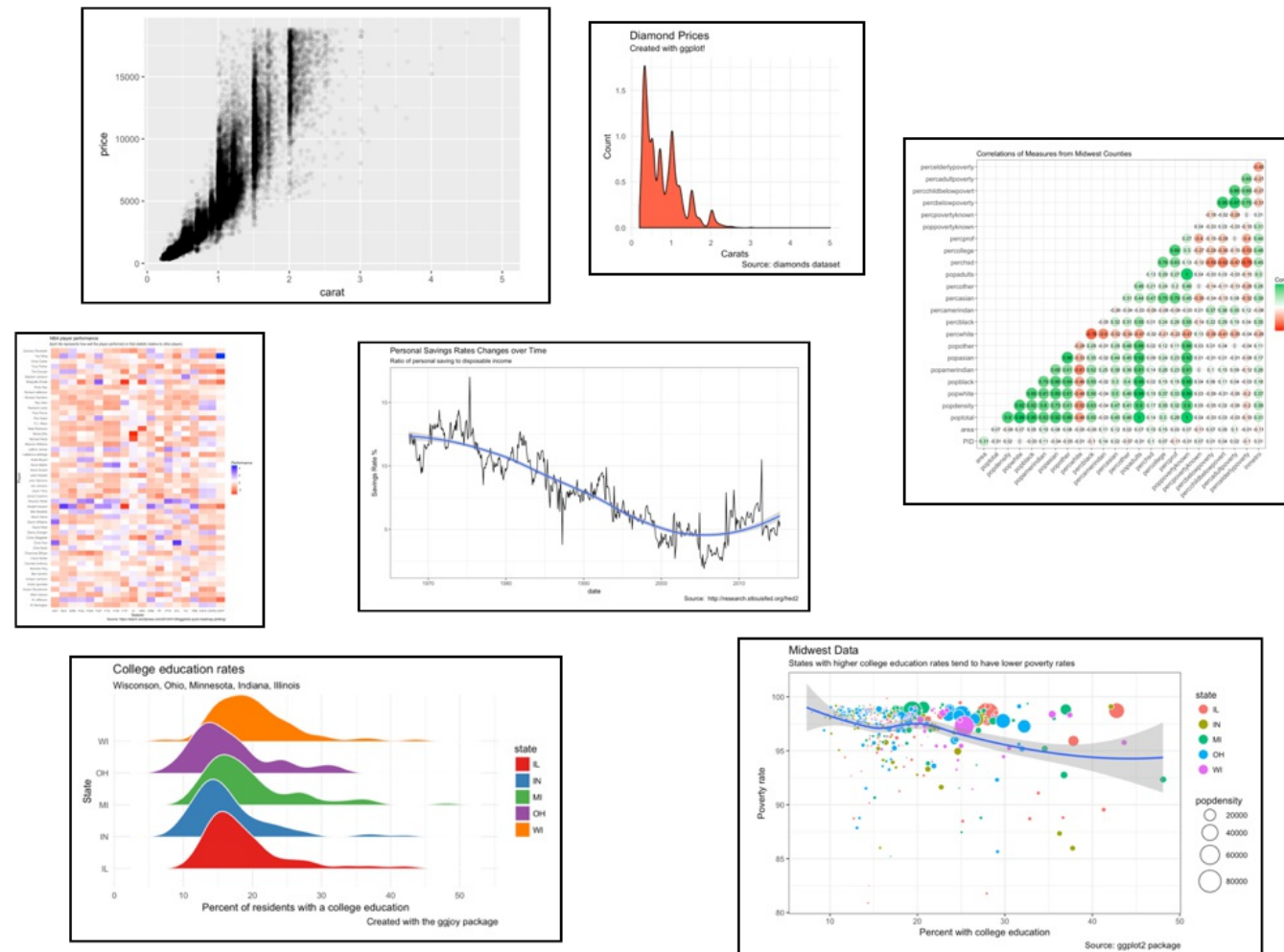
www.therbootcamp.com

@therbootcamp

October 2018

As good as R is for statistics,
it's as good if not better for
data visualisation

Nathaniel D. Phillips



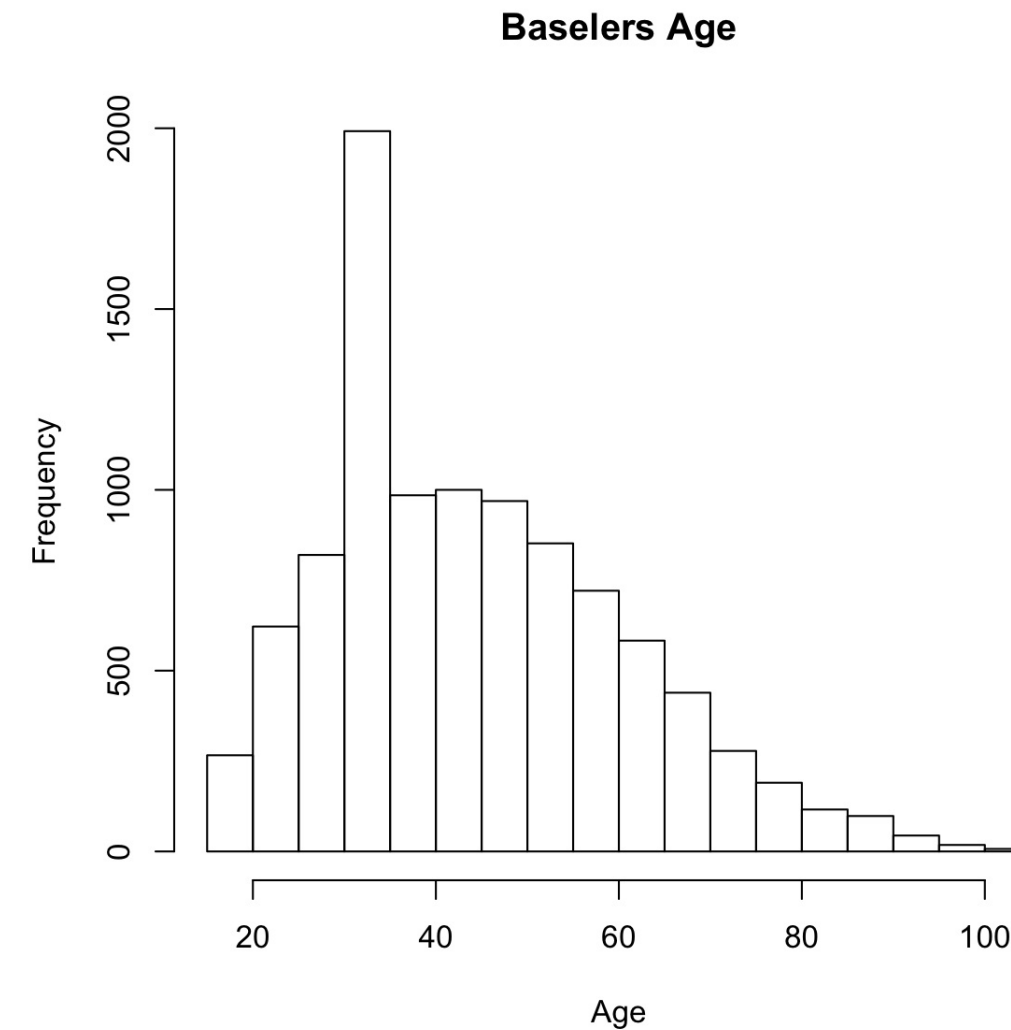
Base R Plotting

The **classic framework** of plotting.

Contains separate **function for each 'type'** of plot.

E.g. `barplot()` for a bar plot, `boxplot()` for a box plot, and `plot()` for a scatterplot.

```
# Histogram in base R  
hist(x = baselers$age,  
      xlab = "Age",  
      ylab = "Frequency",  
      main = "Baselers Age")
```



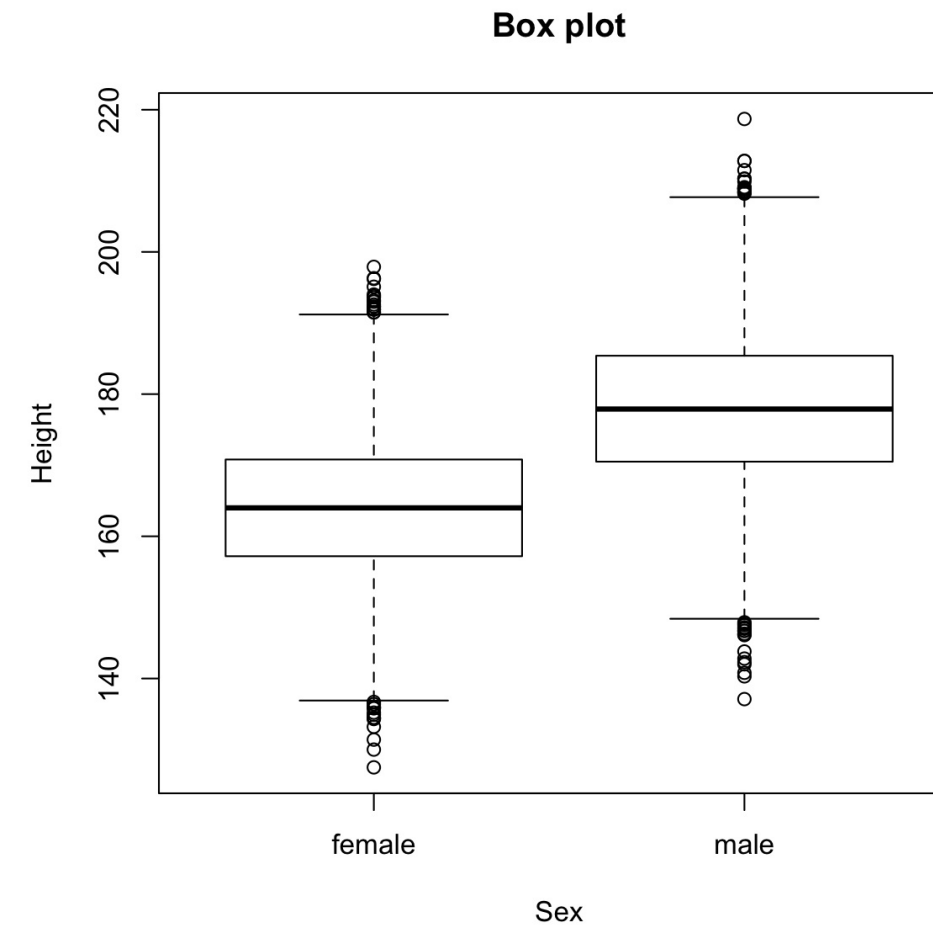
Base R Plotting

The **classic framework** of plotting.

Contains separate **function for each 'type'** of plot.

E.g. `barplot()` for a bar plot, `boxplot()` for a box plot, and `plot()` for a scatterplot.

```
# Boxplot in base R  
boxplot(formula = height ~ sex,  
        data = baselers,  
        xlab = "Sex",  
        ylab = "Height",  
        main = "Box plot")
```



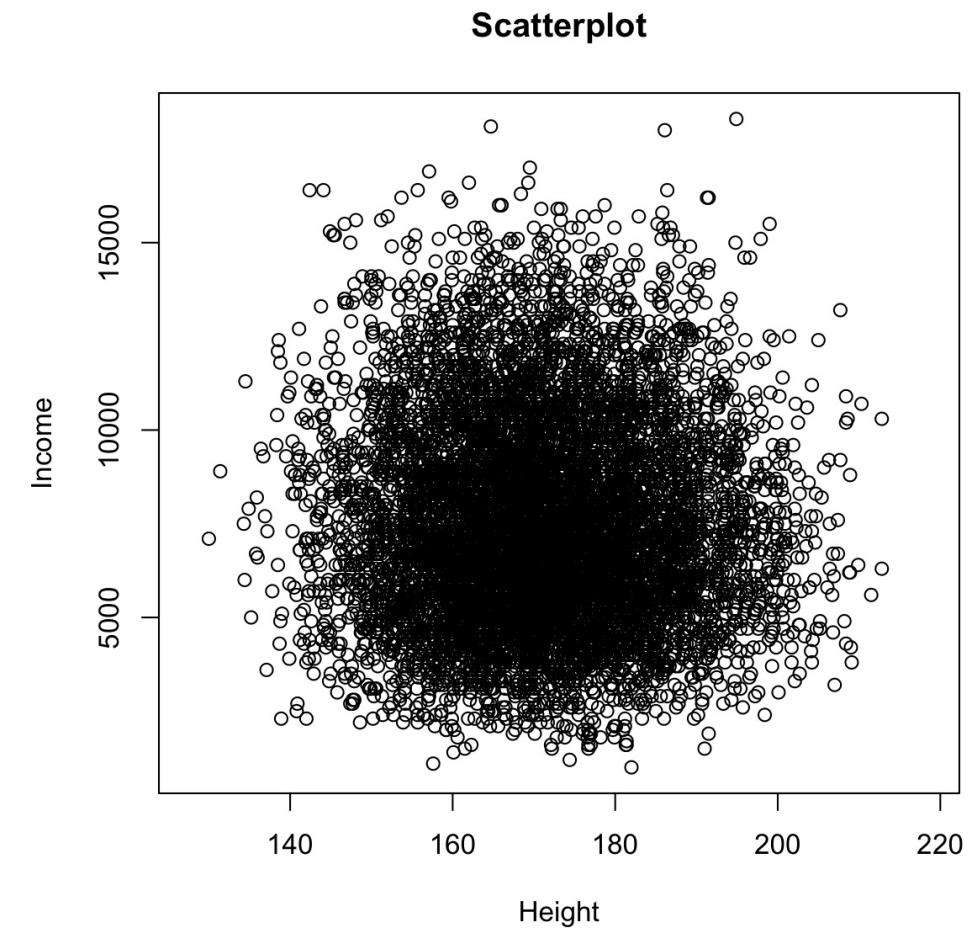
Base R Plotting

The **classic framework** of plotting.

Contains separate **function for each 'type'** of plot.

E.g. `barplot()` for a bar plot, `boxplot()` for a box plot, and `plot()` for a scatterplot.

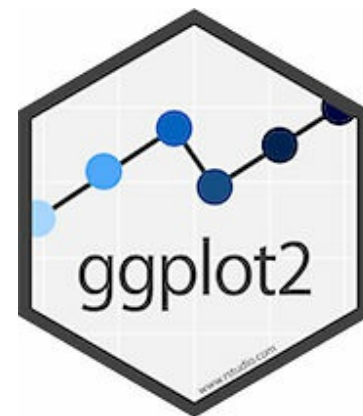
```
# Scatterplot in base R  
plot(x = baselers$height,  
      y = baselers$income,  
      xlab = "Height",  
      ylab = "Income",  
      main = "Scatterplot")
```



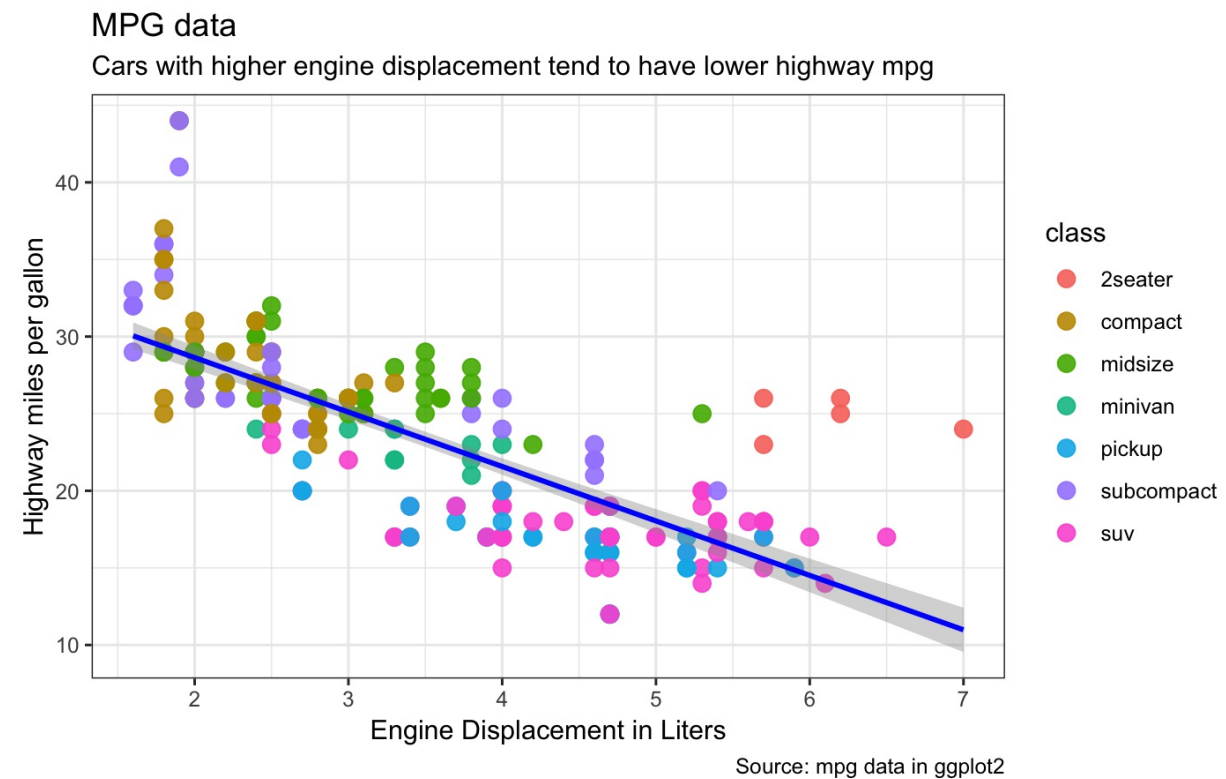
Problems with Base R plotting

- Default plots look pretty **outdated**.
- Plots can quickly require a **LOT of code**.
- Can't store plots as **objects** to reference and update later

Solution: `ggplot2`



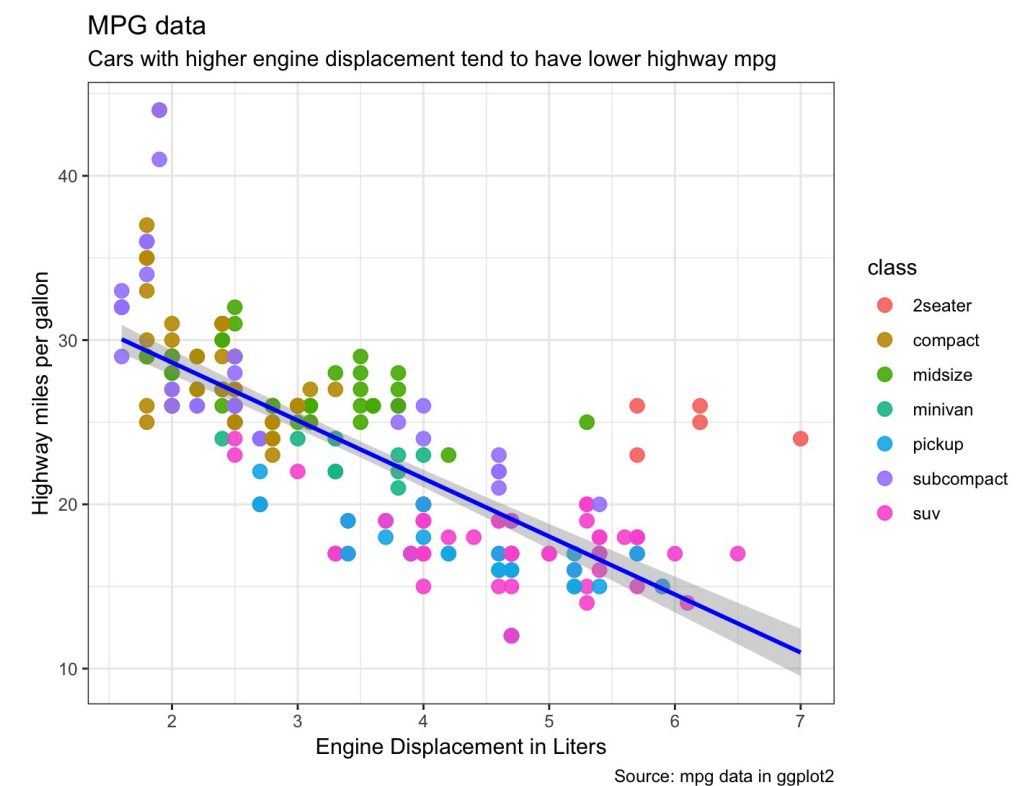
This plot would take **a lot of code in Base R** but **just 10 lines of code**, 5 of which controlling the labels, in **ggplot2**.



Grammar of Graphics in ggplot2

The **Grammar of graphics** breaks down plots into several key pieces:

Aesthetics	Description
Data	What dataframe contains the data?
axes	What does the x-axis, y-axis, color (etc) represent?
color	What does color represent?
size	What does size represent?
geometries	What kind of geometric object do you want to plot?
facets	Should there be groups of plots?



Our goal: Creating this plot

Data

- Use the mpg tibble

Aesthetics

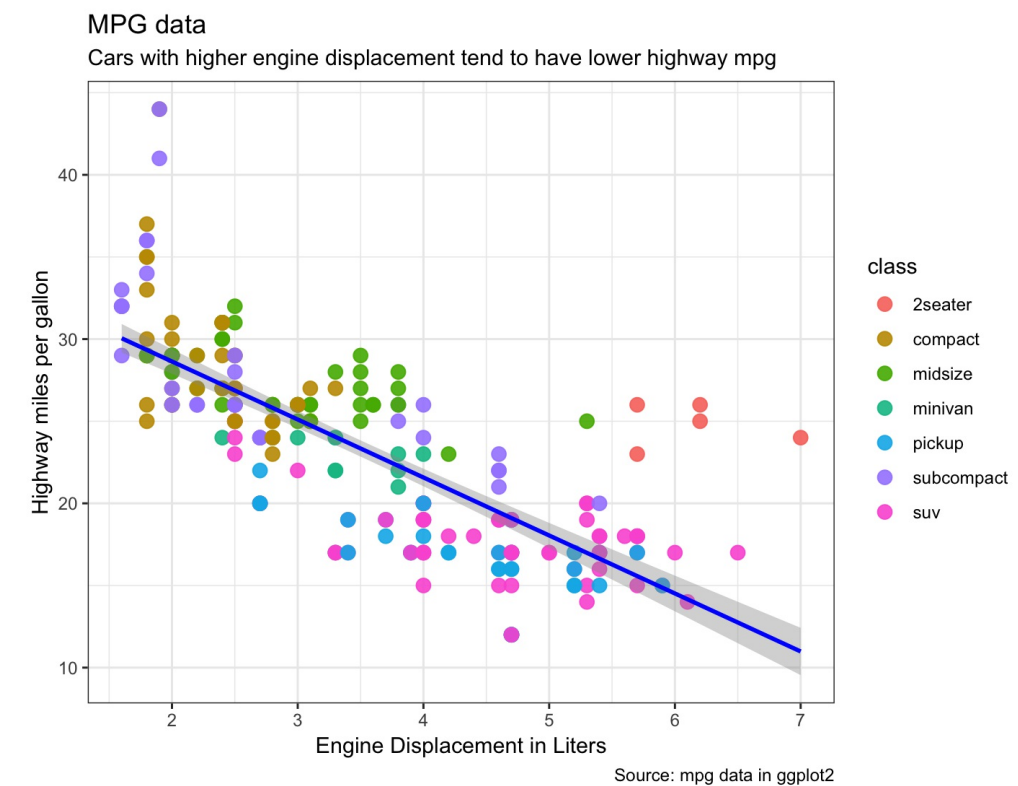
- Engine displacement (displacement) on the x axis
- Highway miles per gallon (hwy) on the y-axis
- Color plotting elements by the class of car

Geometric objects

- Show data as points
- Add a regression line

Labels and themes

- Add plotting labels
- Use a black and white plotting theme



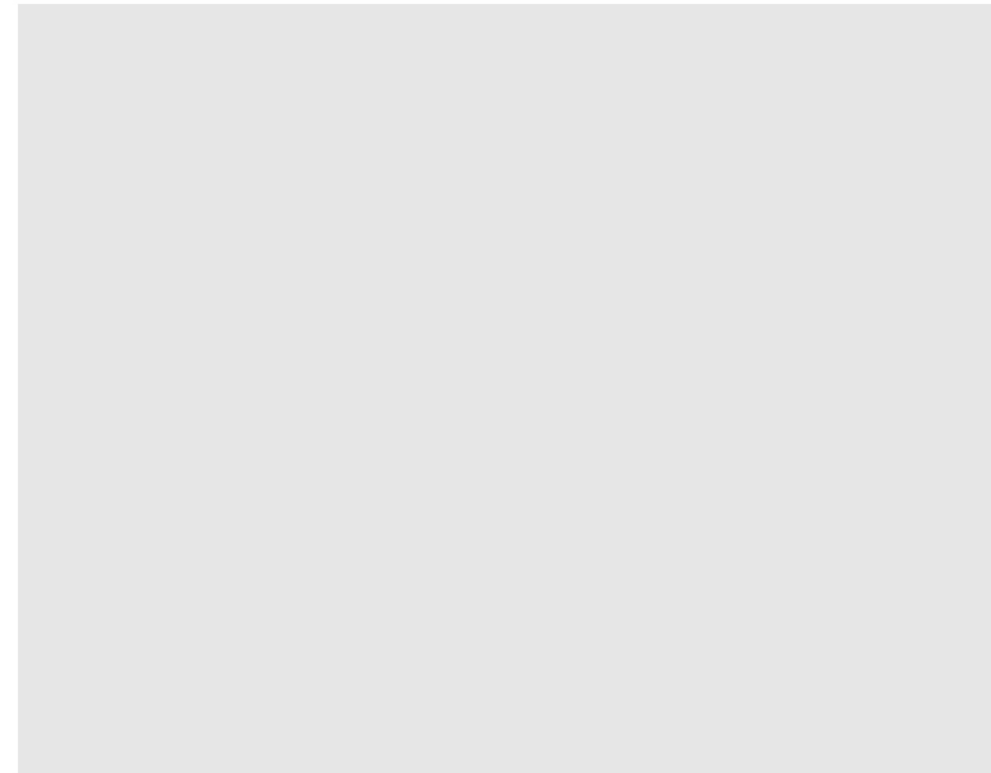
ggplot()

To **create a ggplot2 object**, use the `ggplot()` function

`ggplot()` has two main arguments:

- `data` - A data frame (aka tibble)
- `mapping` - A call to `aes()`

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



ggplot()

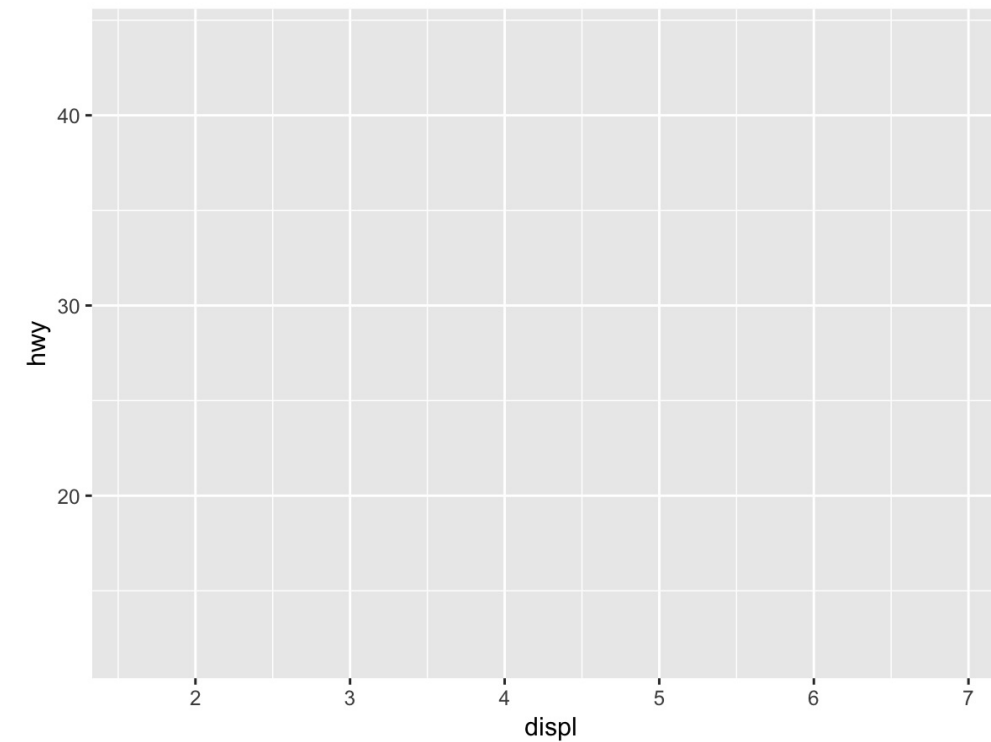
An **aesthetic mapping** is a visual property of the objects in your plot.

Use `aes()` to assign columns in your dataframe to properties in your plot.

Common aesthetics are...

aesthetics	Description
x, y	Data mapped to coordinates
color, fill	Border and fill colors
alpha	Transparency
size	Size
shape	Shape

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

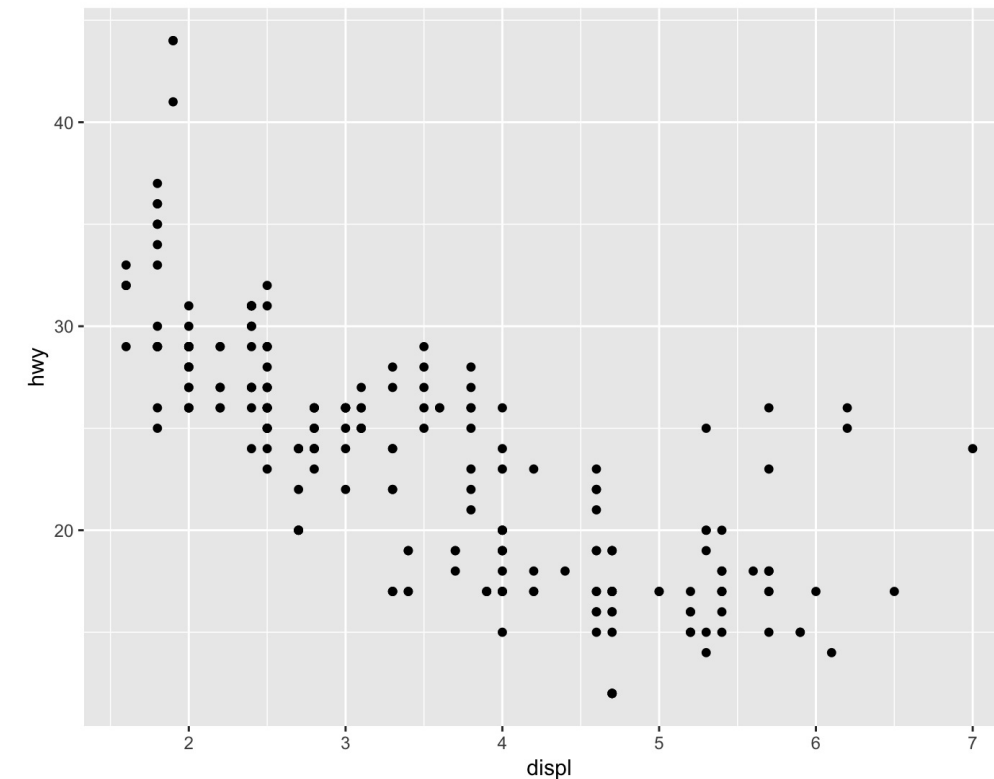


Adding elements to plots with +

Once you have specified the data argument, and global aesthetics with `mapping = aes()`, **add additional elements to the plot with +**.

The + operator works just like the pipe `%>%` in `dplyr`. **It just means "and then..."**

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



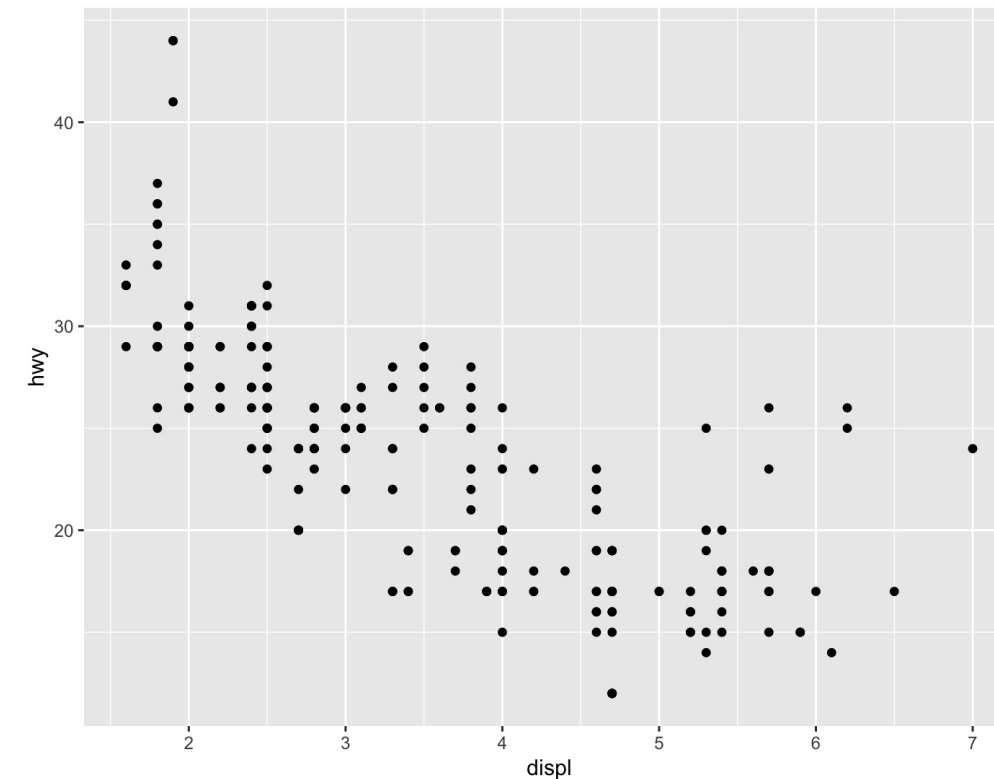
Geometric objects (geom)

A **geom** is a geometric object in a plot that represents data

To add a geom to a plot, just include `geom_X()` where X is the type of geom.

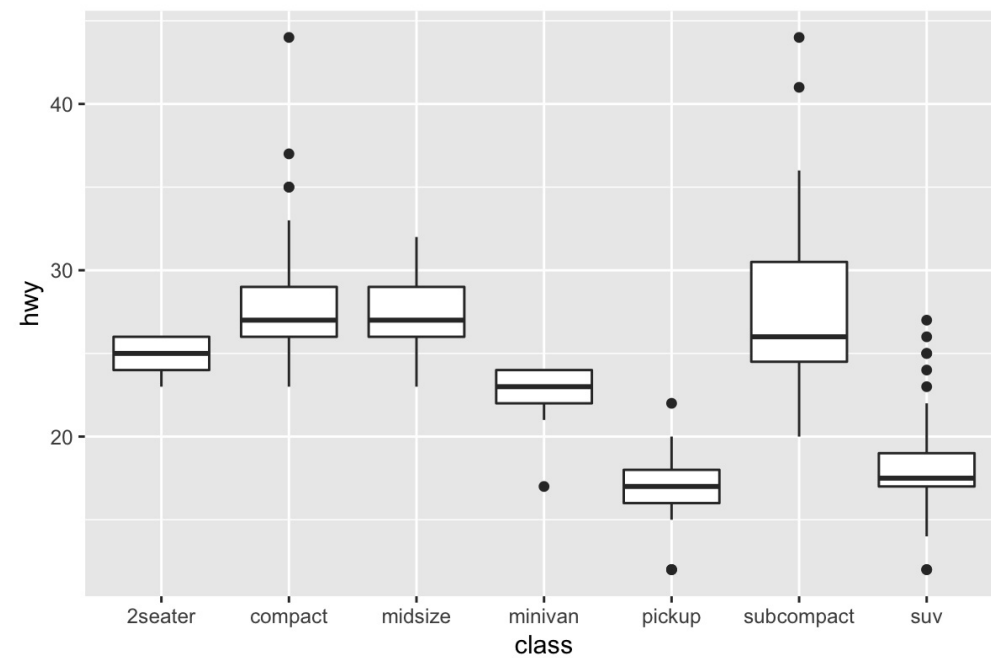
Common geoms are...

geom	output
<code>geom_point()</code>	Points
<code>geom_bar()</code>	Bar
<code>geom_boxplot()</code>	Boxplot
<code>geom_count()</code>	Points with size reflecting frequency
<code>geom_smooth()</code>	Smoothed line



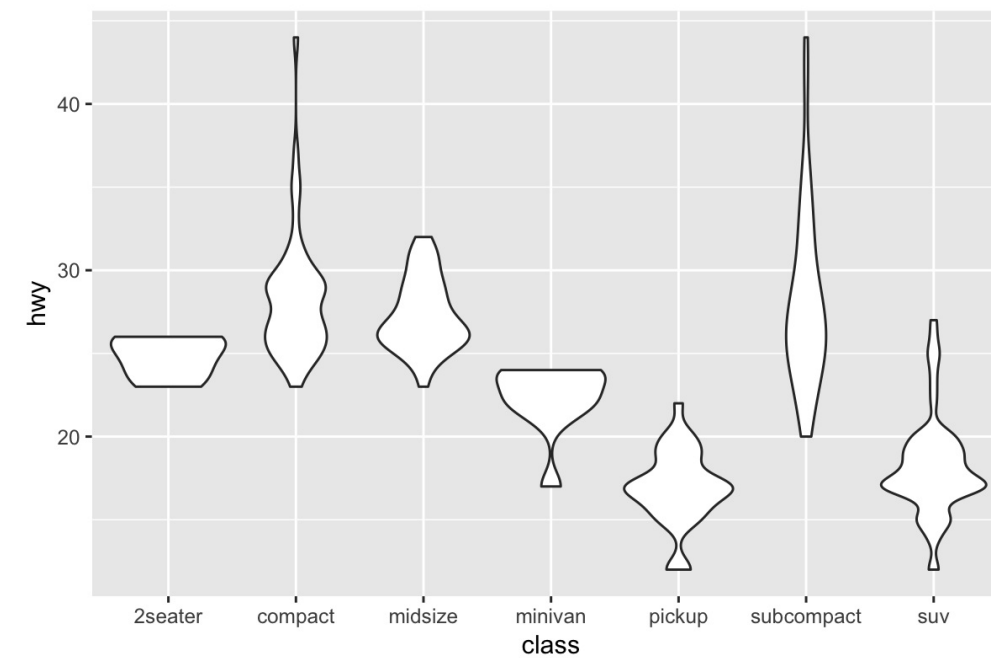
geom_boxplot()

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



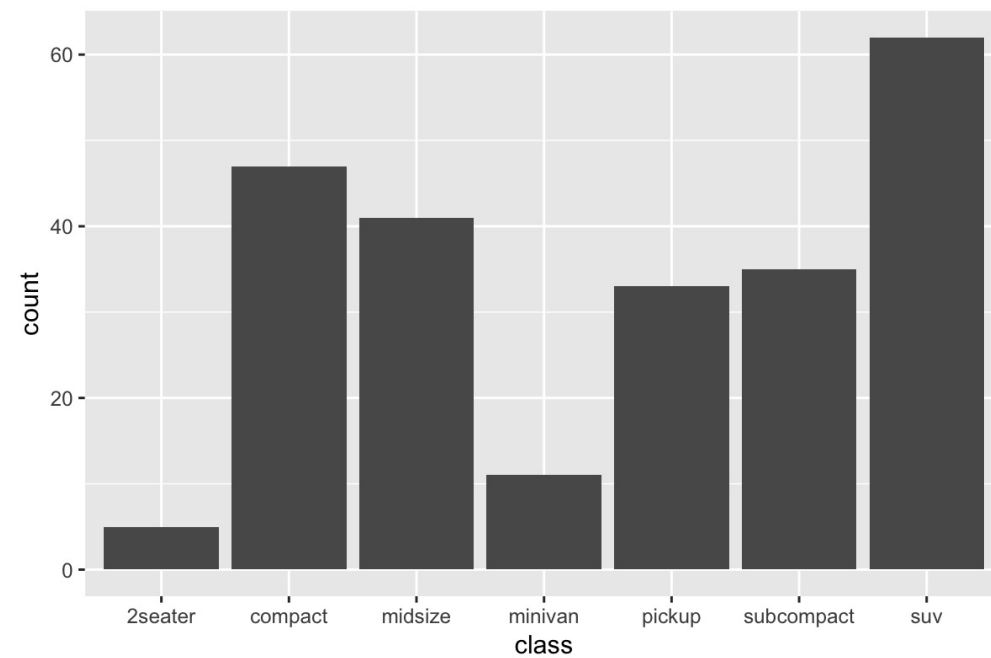
geom_violin()

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



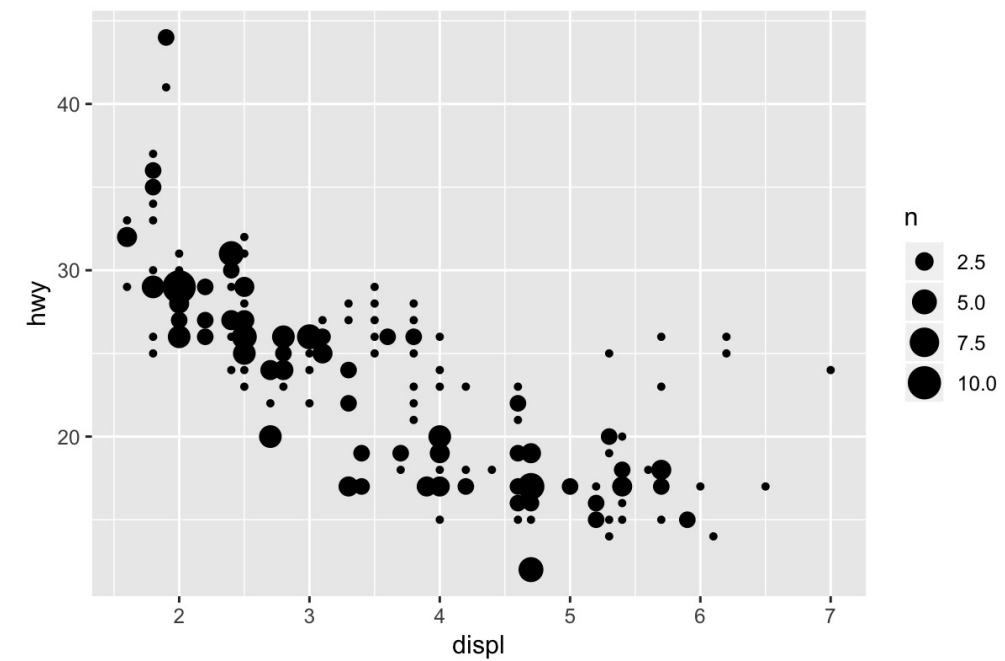
geom_bar()

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



geom_count()

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



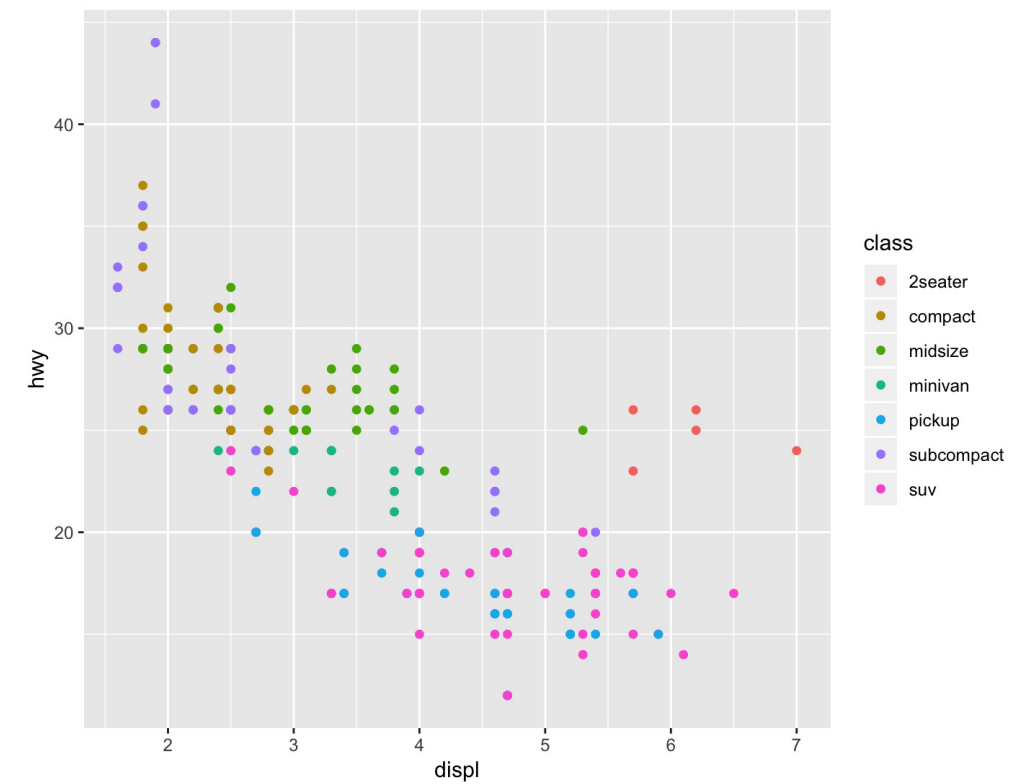
aes()

color geoms according to a variable.

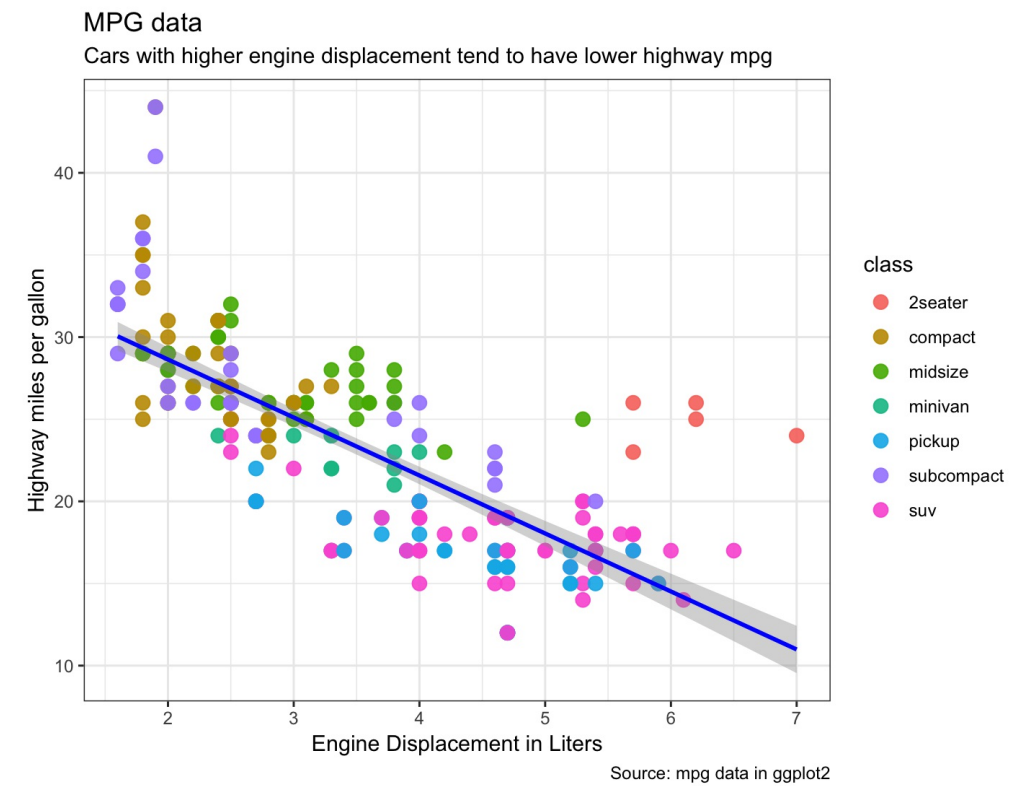
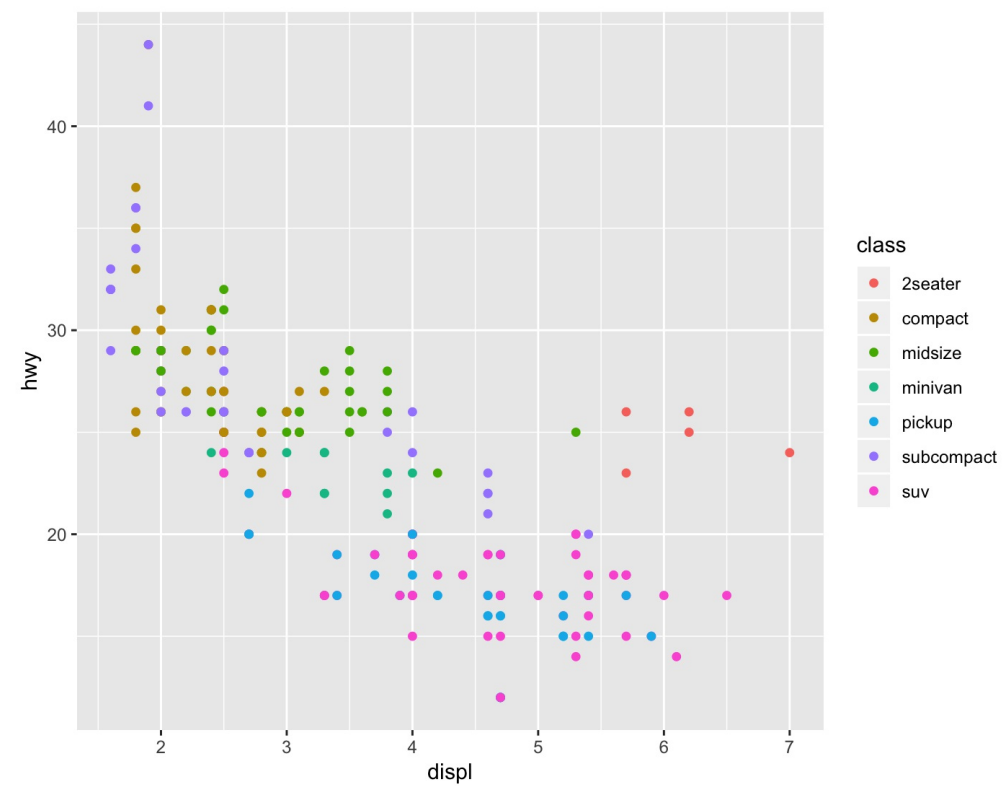
```
ggplot(data = mpg,  
       mapping = aes(x = displ,  
                     y = hwy,  
                     color = class)) +  
  geom_point()
```

`mpg`

displ	hwy	class	year
2.5	29	subcompact	2008
3.0	25	midsize	1999
3.4	19	pickup	1999
2.5	28	subcompact	2008
2.4	29	compact	1999



What's next?



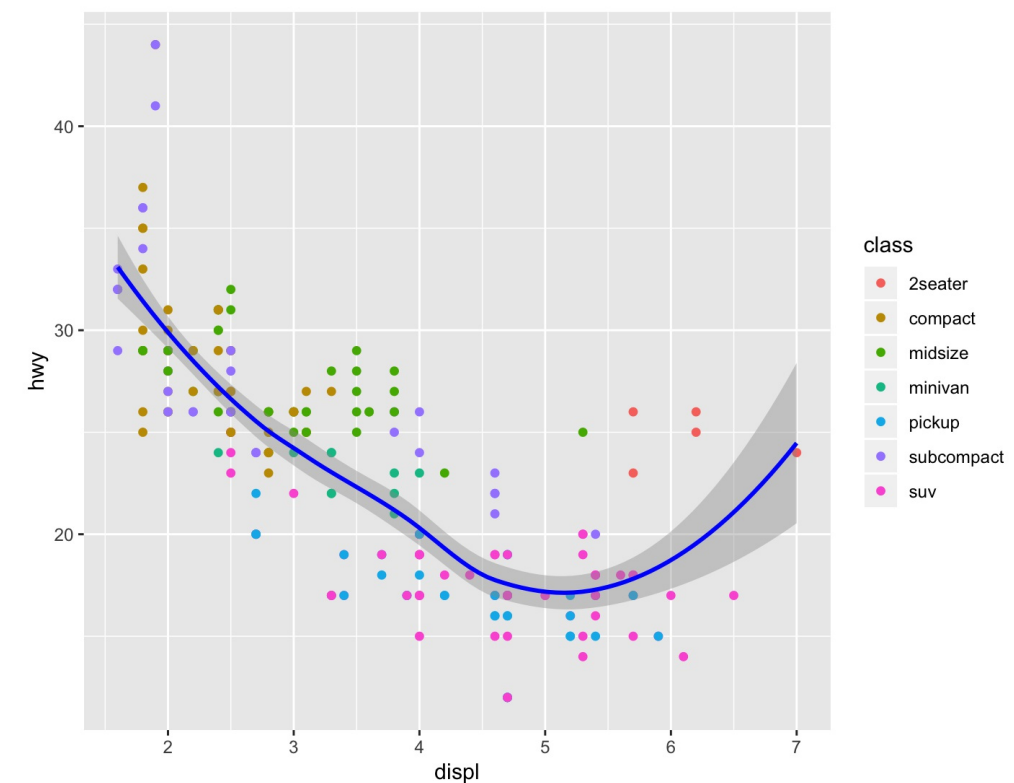
geom_smooth()

geom_smooth() adds a **smoothed line**.

Change how the line is created with method (e.g., method = lm).

Color the line with col.

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



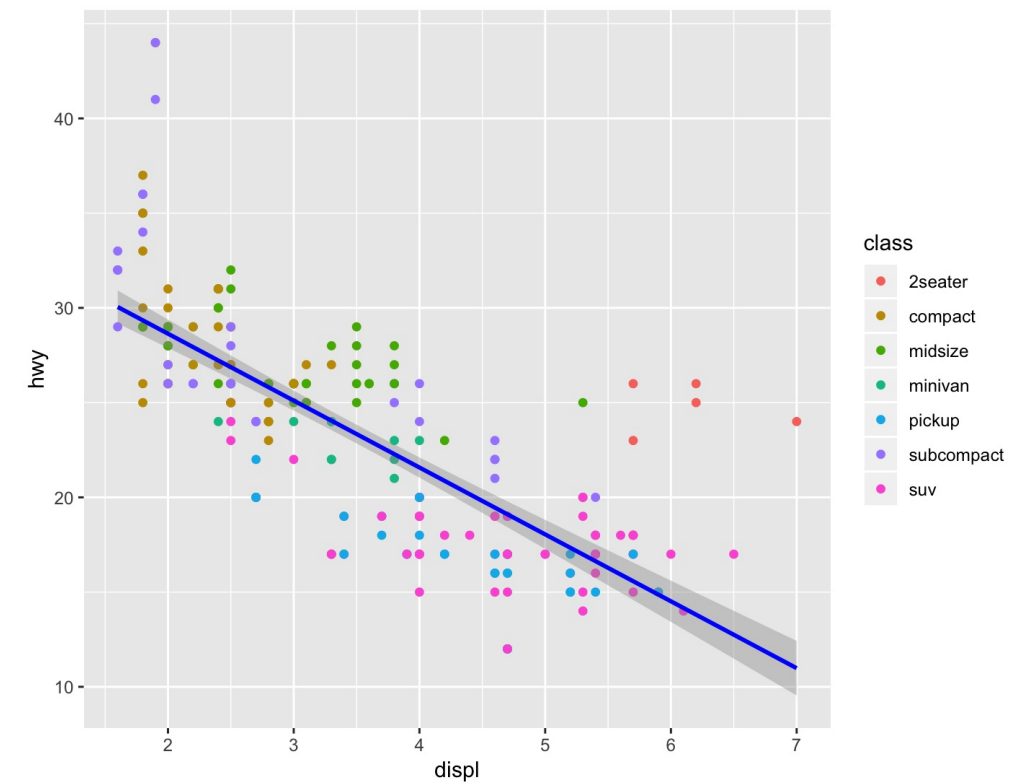
geom_smooth()

geom_smooth() adds a **smoothed line**.

Change how the line is created with method (e.g., method = lm).

Color the line with col

```
ggplot(data = mpg,
       mapping = aes(x = displ, y = hwy,
                     col = class)) +
  geom_point() +
  geom_smooth(col = "blue",
             method = "lm")
```

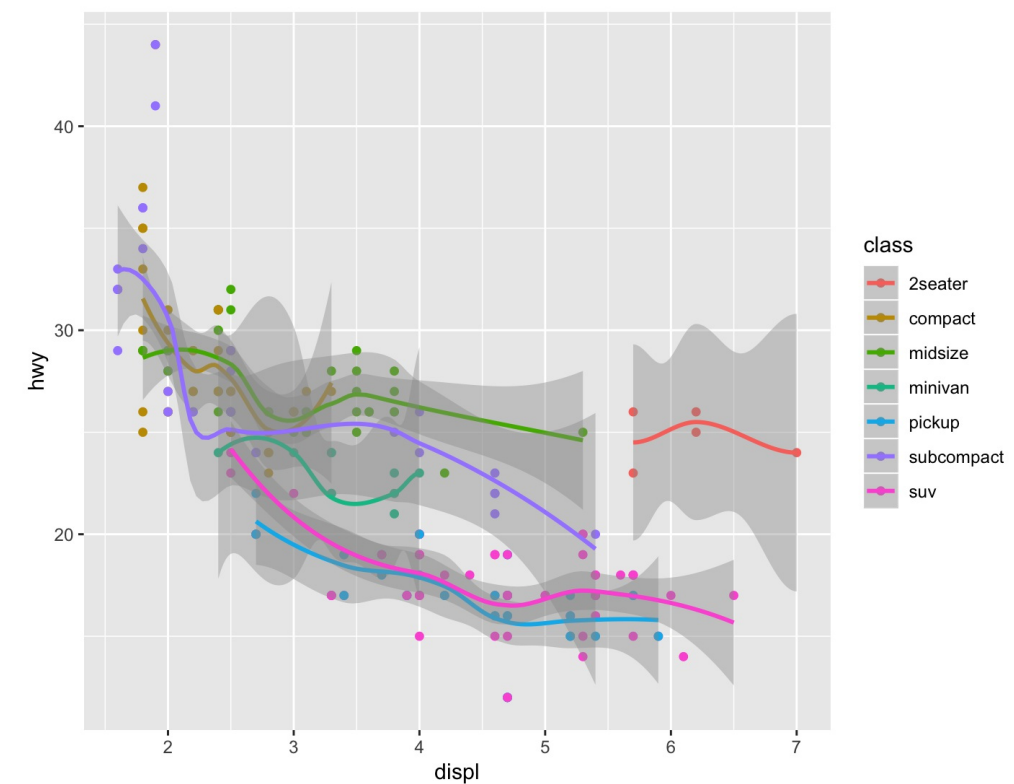


Overriding aesthetics

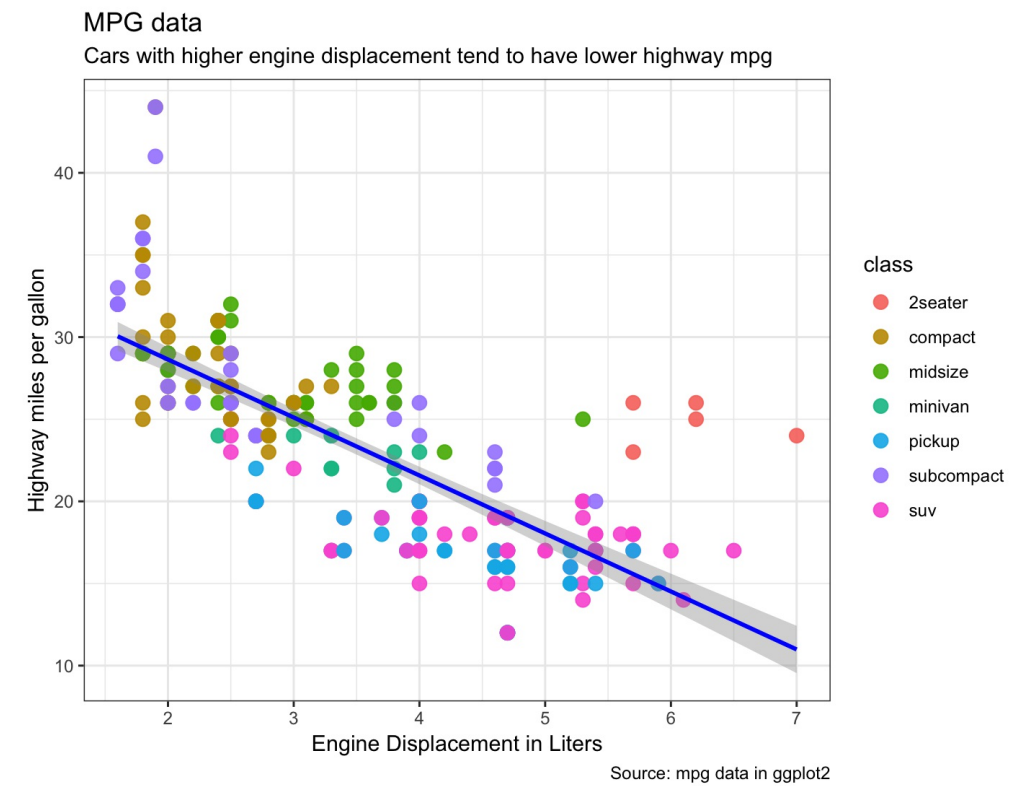
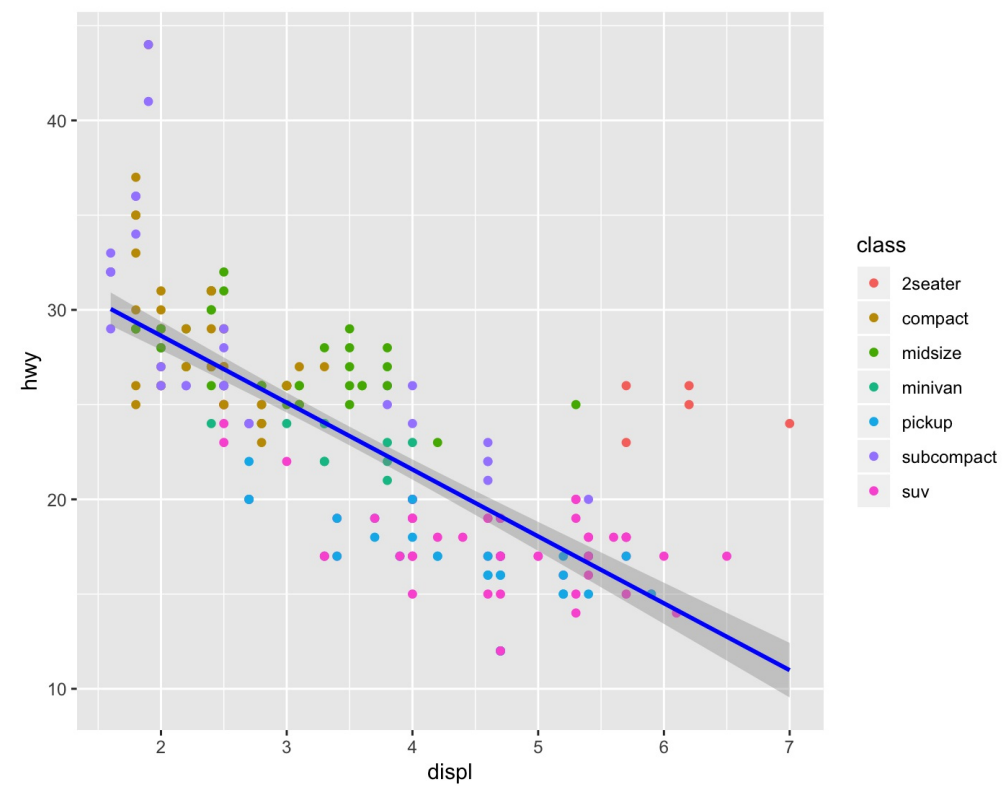
If you add additional plotting aesthetics, they will **override** the general plotting aesthetics.

This is what happens, when you don't override...

```
ggplot(data = mpg,
       mapping = aes(x = displ, y = hwy,
                     col = class)) +
  geom_point() +
  geom_smooth() # no overriding
```



What's next?



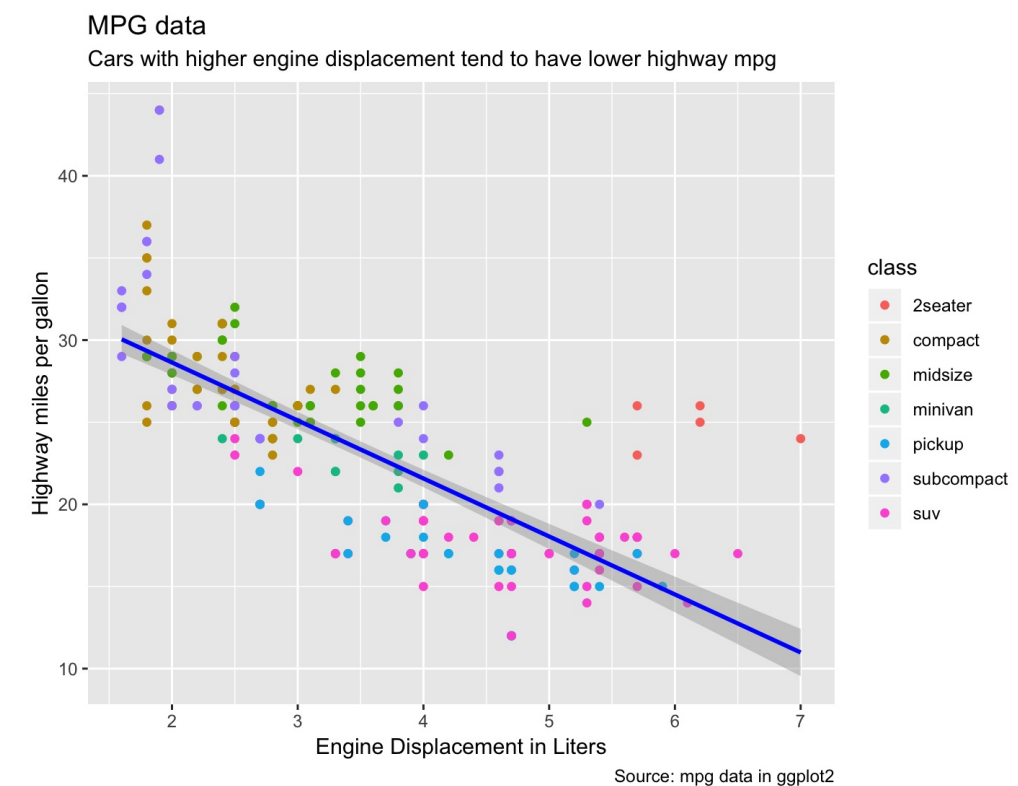
labs()

You can add **labels** to a plot with the `labs()` function

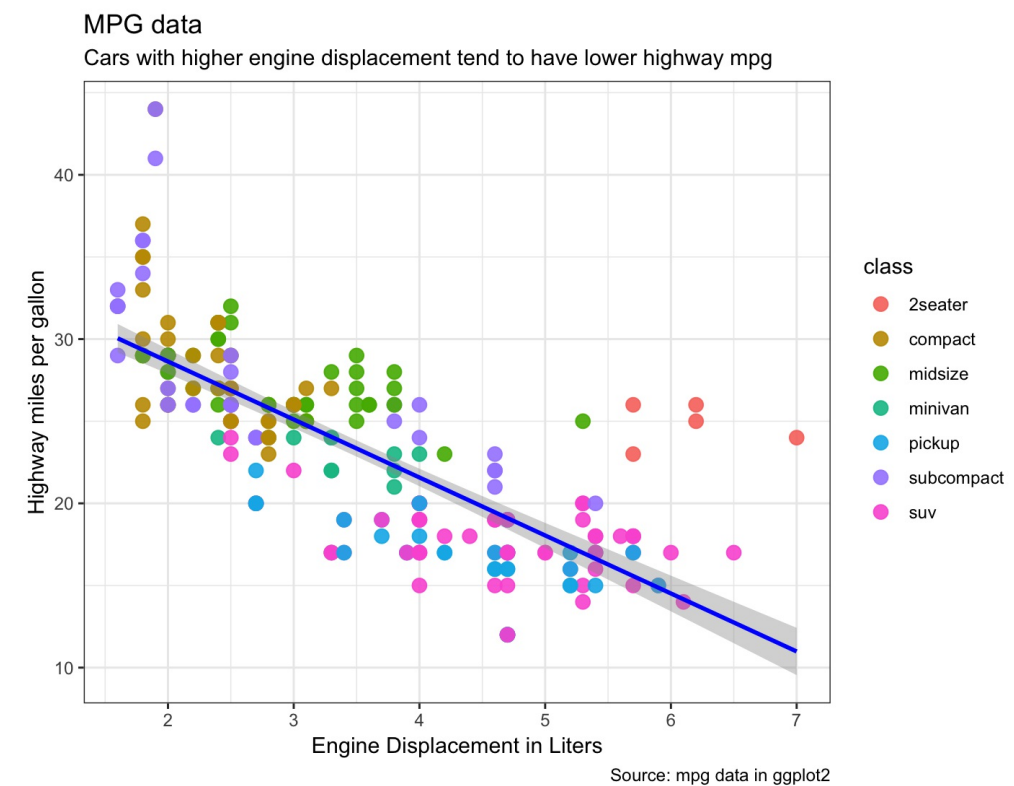
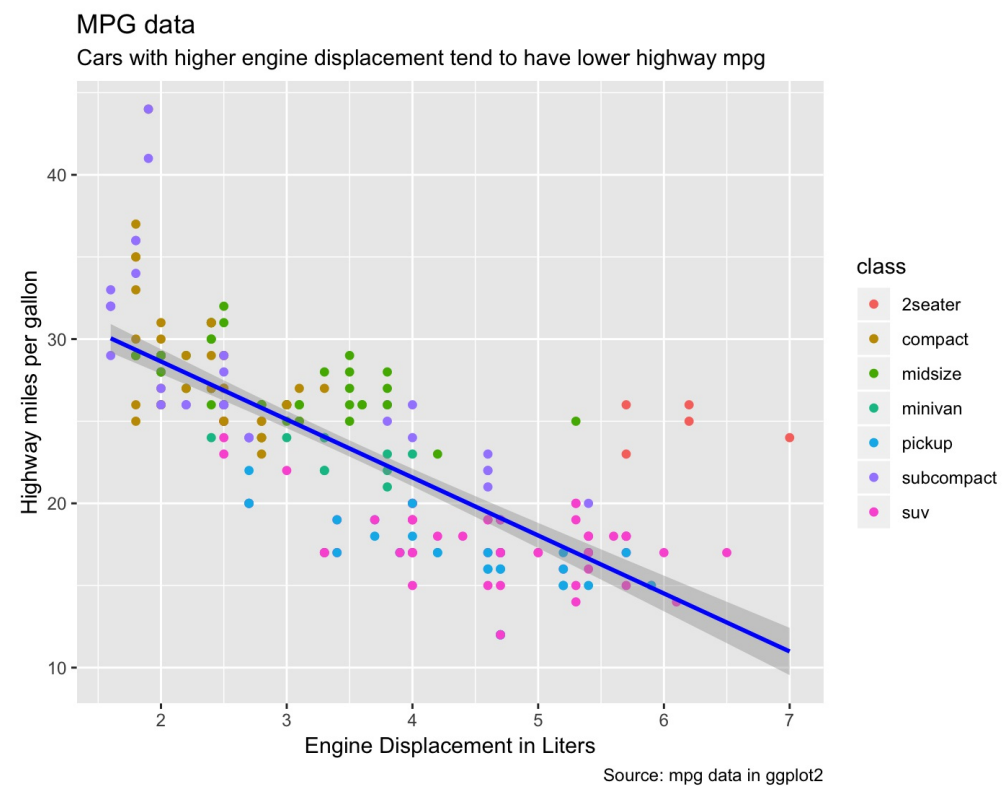
`labs()` arguments are ...

- `title` - Main title
- `subtitle` - Subtitle
- `caption` - Caption below

```
ggplot(...) +  
  labs(x = "Engine Displ...",  
       y = "Highway miles...",  
       title = "MPG data",  
       subtitle = "Cars with ...",  
       caption = "Source...")
```



What's next?



Themes with theme_XX()

A plotting **theme** controls many aspects of its **overall look**, from the background, to the grid lines, to the label font to the spacing between plot labels and the plotting space.

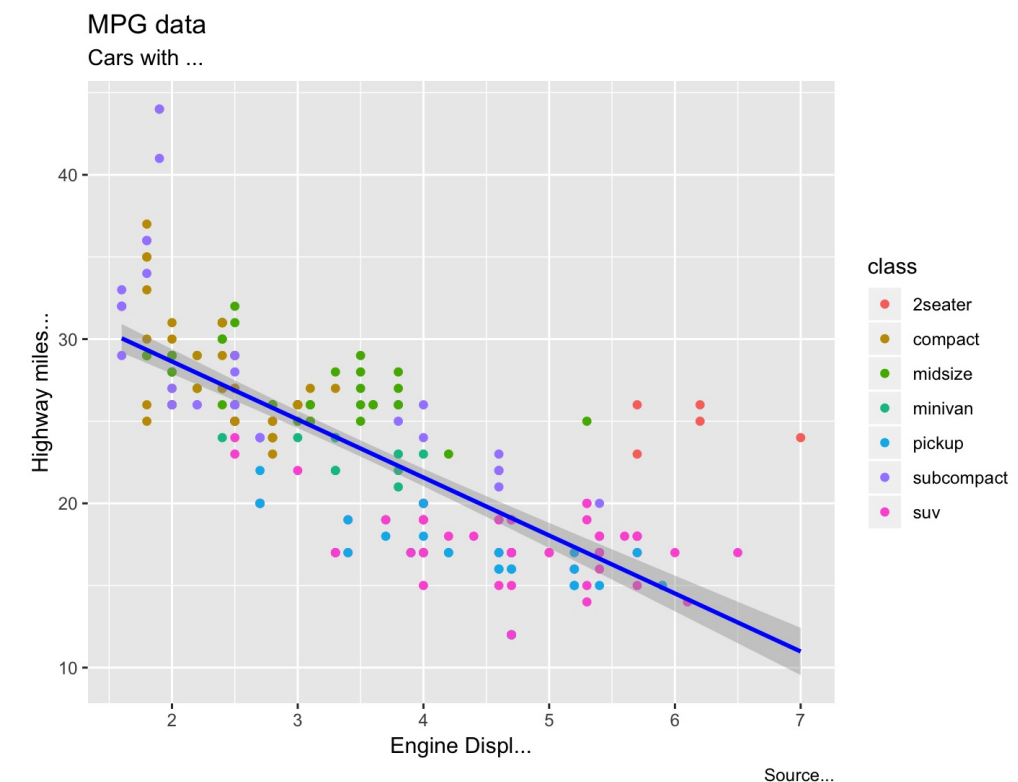
Themes built into ggplot2

```
theme_bw()  
theme_minimal()  
theme_classic()  
theme_light()  
theme_void()
```

Themes from the ggthemes package

```
theme_excel()  
theme_economist()  
etc.
```

```
ggplot(...) +  
  theme_gray() # The Default theme
```



Themes with `theme_XX()`

A plotting **theme** controls many aspects of its **overall look**, from the background, to the grid lines, to the label font to the spacing between plot labels and the plotting space.

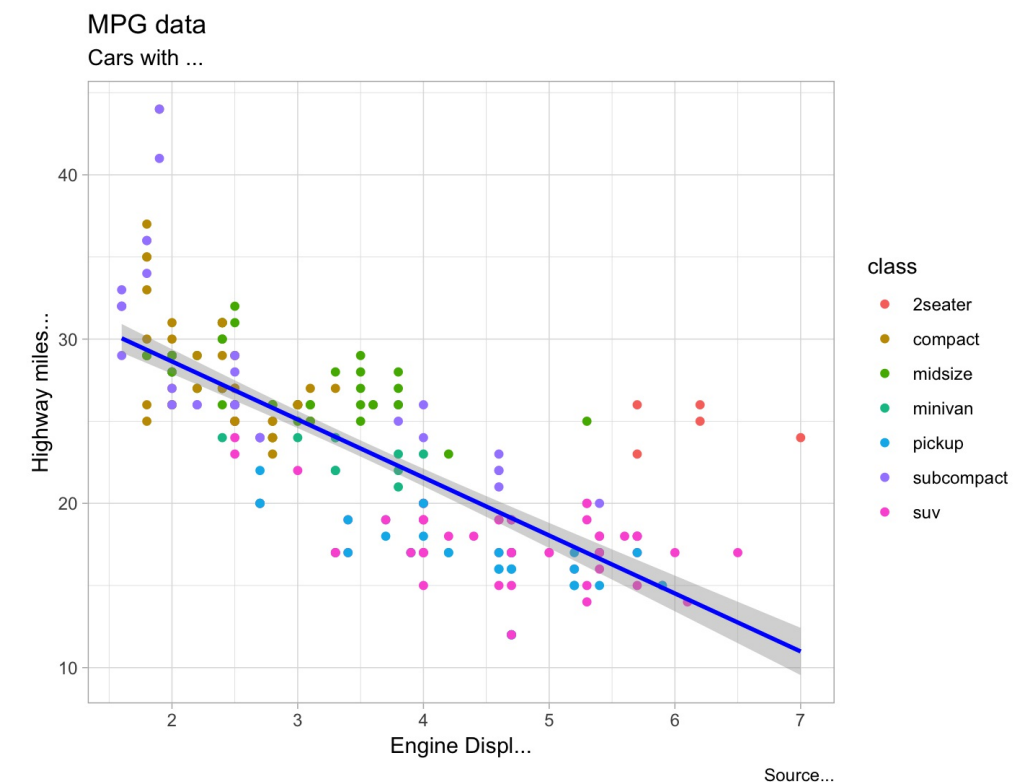
Themes built into `ggplot2`

```
theme_bw()  
theme_minimal()  
theme_classic()  
theme_light()  
theme_void()
```

Themes from the `ggthemes` package

```
theme_excel()  
theme_economist()  
etc.
```

```
ggplot(...) +  
  theme_light()
```



Themes with `theme_XX()`

A plotting **theme** controls many aspects of its **overall look**, from the background, to the grid lines, to the label font to the spacing between plot labels and the plotting space.

Themes built into `ggplot2`

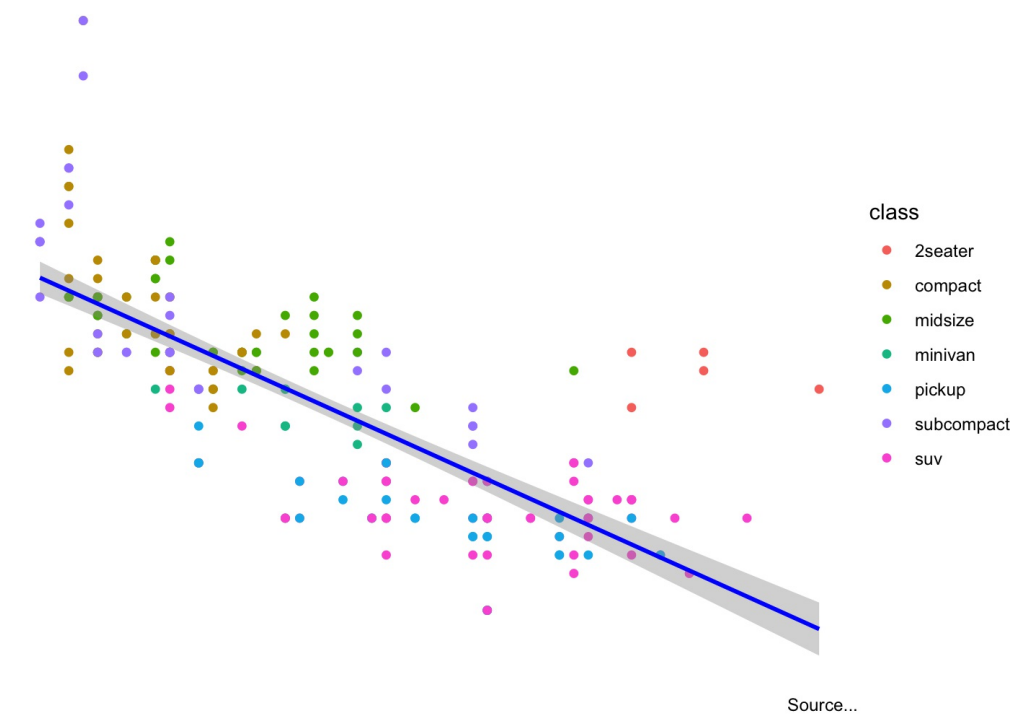
```
theme_bw()  
theme_minimal()  
theme_classic()  
theme_light()  
theme_void()
```

Themes from the `ggthemes` package

```
theme_excel()  
theme_economist()  
etc.
```

```
ggplot(...) +  
  theme_void()
```

MPG data
Cars with ...



Themes with `theme_XX()`

A plotting **theme** controls many aspects of its **overall look**, from the background, to the grid lines, to the label font to the spacing between plot labels and the plotting space.

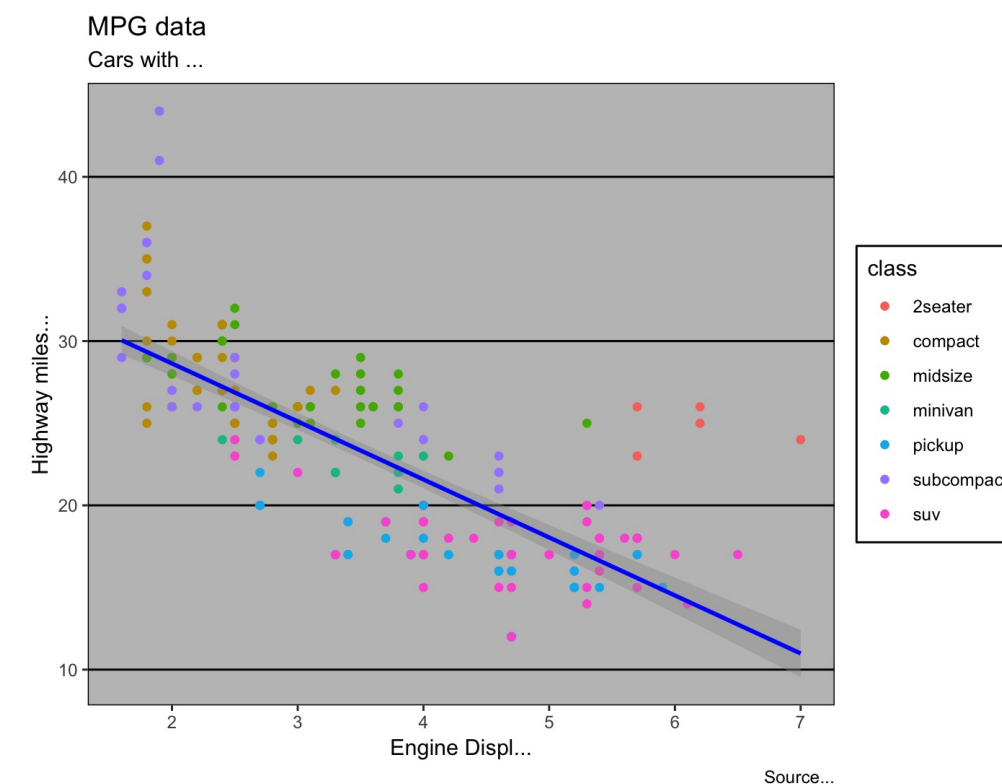
Themes built into `ggplot2`

```
theme_bw()  
theme_minimal()  
theme_classic()  
theme_light()  
theme_void()
```

Themes from the `ggthemes` package

```
theme_excel()  
theme_economist()  
etc.
```

```
ggplot(...) +  
  theme_excel()
```



Themes with `theme_XX()`

A plotting **theme** controls many aspects of its **overall look**, from the background, to the grid lines, to the label font to the spacing between plot labels and the plotting space.

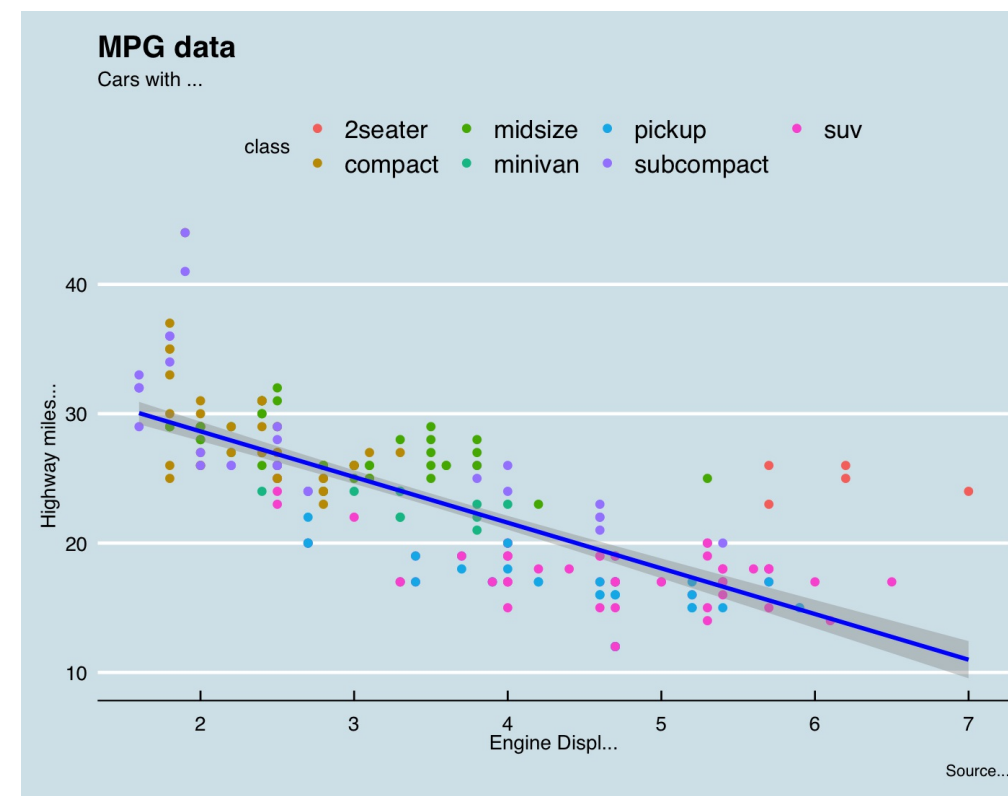
Themes built into `ggplot2`:

```
theme_bw()  
theme_minimal()  
theme_classic()  
theme_light()  
theme_void()
```

Themes from the `ggthemes` package

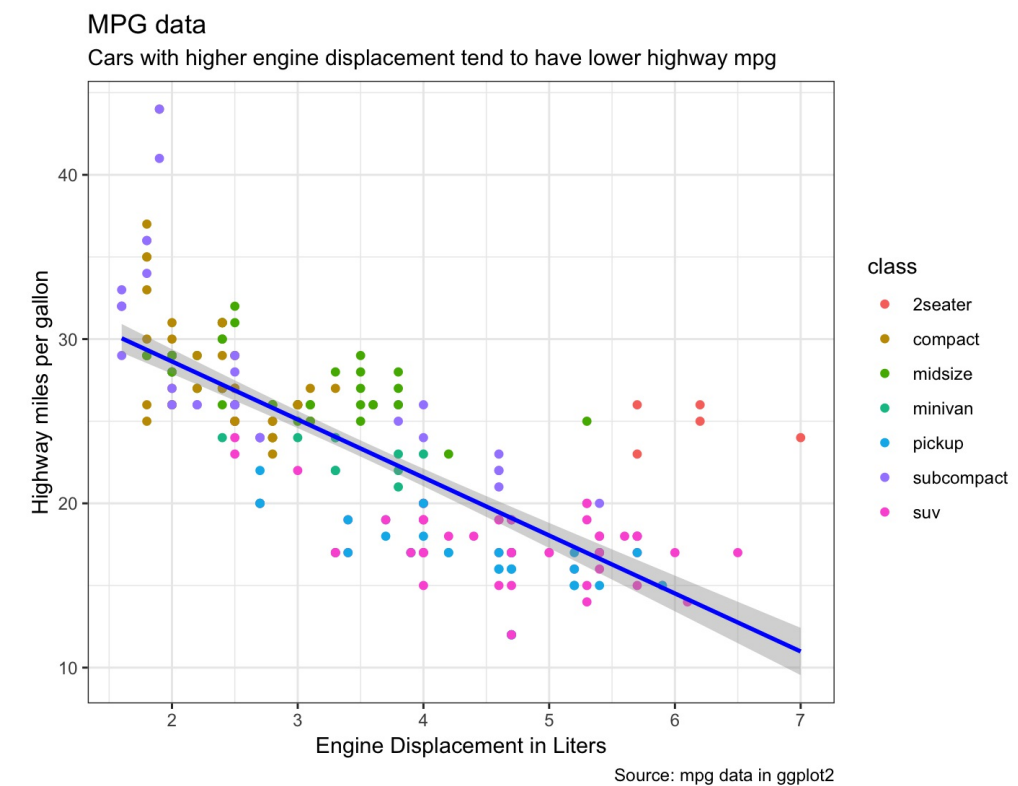
```
theme_excel()  
theme_economist()  
etc.
```

```
ggplot(...) +  
  theme_economist()
```



Final result!

```
ggplot(data = mpg,
       mapping = aes(x = displ,
                     y = hwy,
                     col = class)) +
  geom_point() +
  geom_smooth(col = "blue",
             method = "lm") +
  labs(x = "Engine Displ. in Liters",
       y = "Highway miles per gallon",
       title = "MPG data",
       subtitle = "Cars with higher...",
       caption = "Source: mpg data...") +
  theme_bw()
```

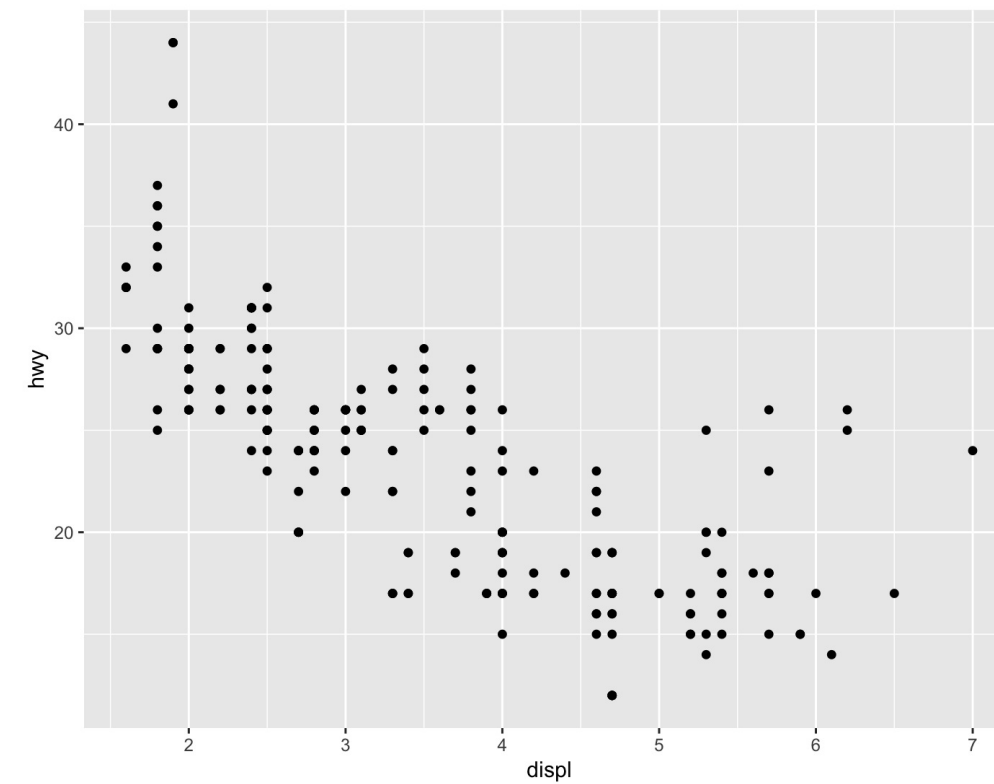


facet_wrap()

Faceting = **same plot for different groups**.

To facet plots, use facet_wrap()

```
# Without faceting  
ggplot(data = mpg,  
       mapping = aes(x = displ,  
                     y = hwy)) +  
  geom_point()
```

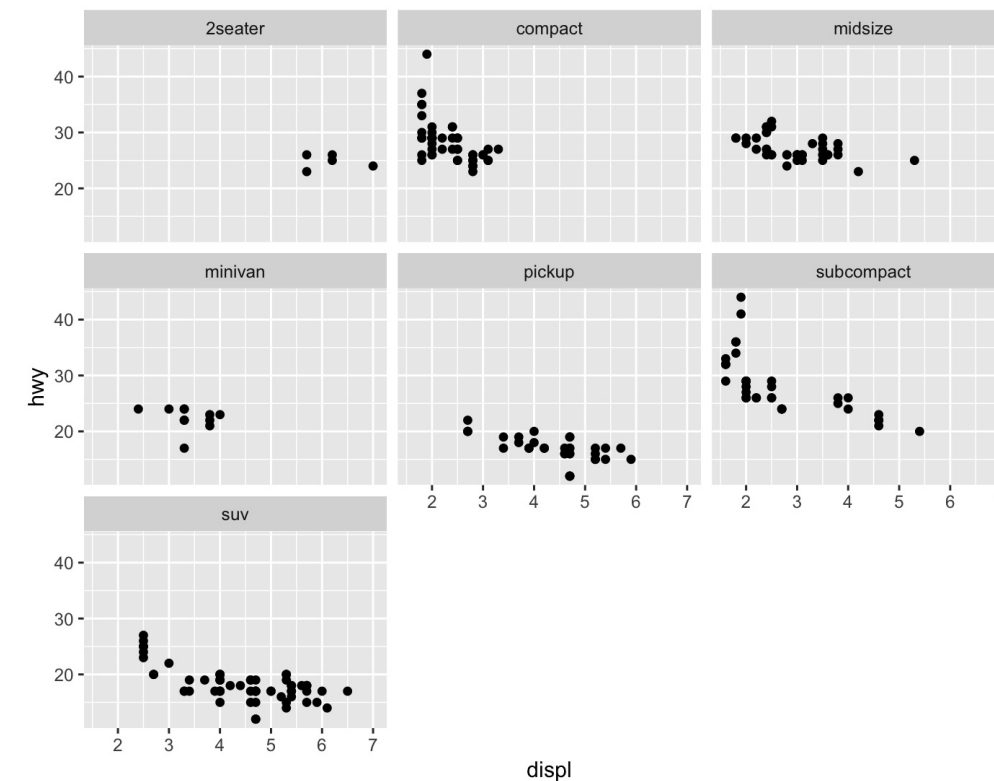


facet_wrap()

Faceting = **same plot for different groups**.

To facet plots, use `facet_wrap()`

```
# With faceting  
ggplot(data = mpg,  
       mapping = aes(x = displ,  
                     y = hwy)) +  
  geom_point() +  
  facet_wrap(~ class) # Tilde first
```



Assigning a ggplot to an object

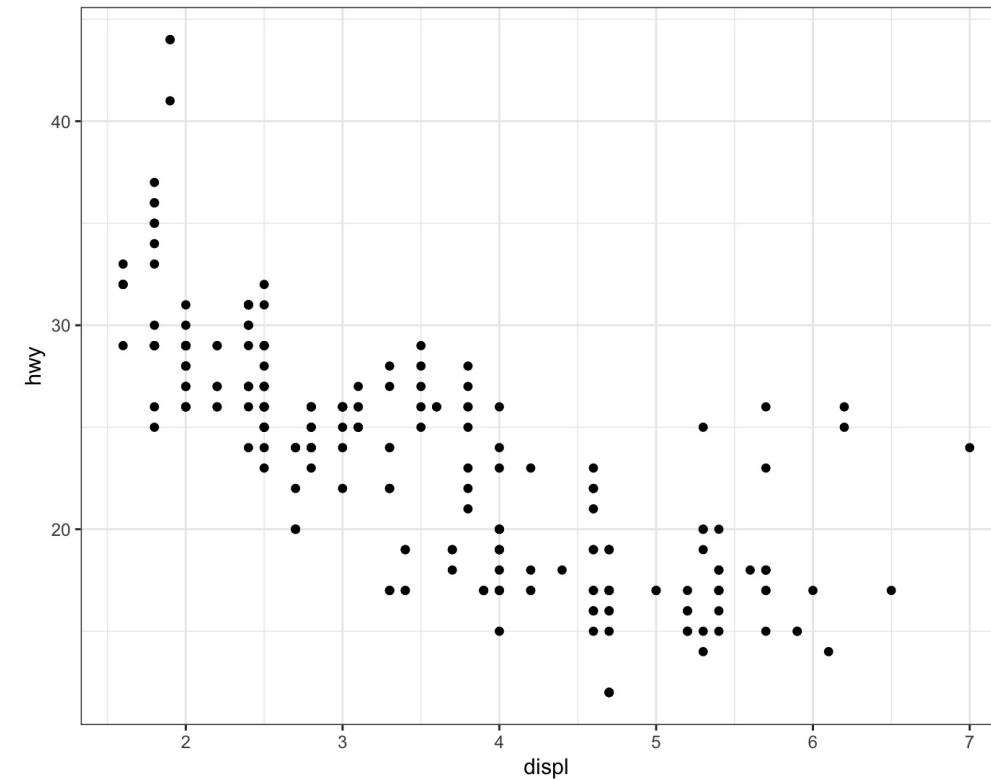
- 1) ggplot returns an object of the class gg.
- 2) You can assign the result of ggplot to an **object**.
- 3) Evaluating the object will show the plot.
- 4) You can even edit existing **ggplot objects**.

```
# Create myplot
myplot <- ggplot(data = mpg,
                 aes(x = displ,
                     y = hwy)) +
  geom_point() +
  theme_bw()

class(myplot)
```

```
[1] "gg"      "ggplot"
```

```
myplot # Evaluate myplot
```



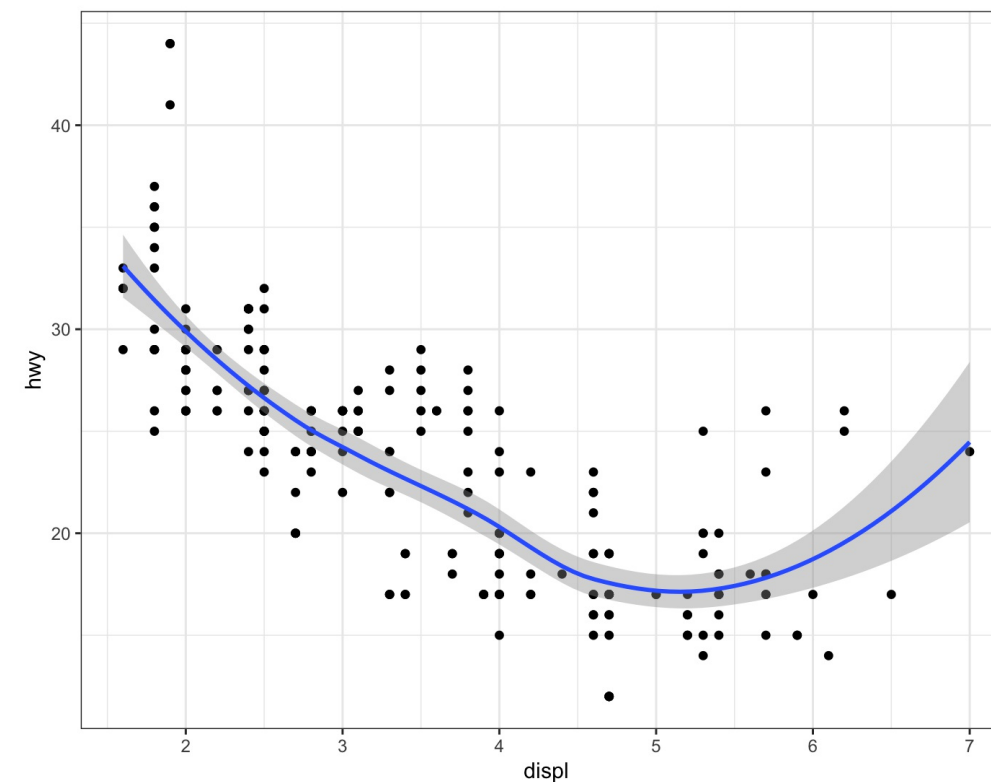
Assigning a ggplot to an object

- 1) ggplot returns an object of the class gg.
- 2) You can assign the result of ggplot to an object.
- 3) Evaluating the object will show the plot.
- 4) You can even edit existing **ggplot objects**.

```
# Create myplot  
myplot <- ggplot(data = mpg,  
                 aes(x = displ,  
                     y = hwy)) +  
  geom_point() +  
  theme_bw()  
  
class(myplot)
```

```
[1] "gg"      "ggplot"
```

```
myplot +  
  geom_smooth() # add geom
```



ggsave()

To **save plots** to a file (e.g., .jpg, .pdf, .png), use the `ggsave()` function.

`ggsave()` main arguments are...

Argument	Definition
filename	File name
plot	Plotting object
device	File type (e.g.; "pdf", "jpeg", "png")
path	File path to save plot
width	Plot width (inches)
height	Plot height (inches)

Save ggplot object called `myplot` to a pdf file...

```
# Create myplot object
myplot <- ggplot(...)

# Create "myplot.pdf", from myplot
ggsave(filename = "myplot.pdf",
        plot = myplot,
        device = "pdf",
        path = "figures",
        width = 6,
        height = 4)
```

Practical

[Link to practical](#)