Plotting Part I

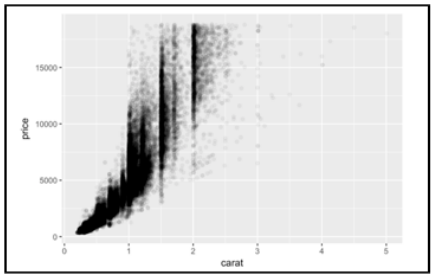
ggplot2

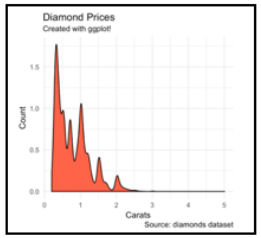
The R Bootcamp
Twitter: @therbootcamp

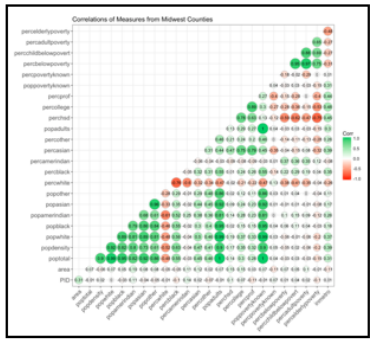
September 2017

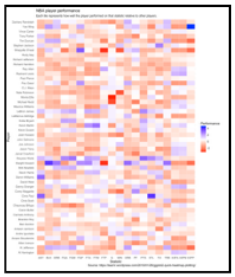
You can do amazing plots in R!

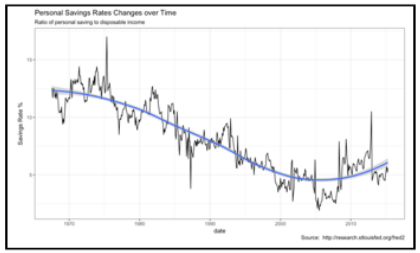
- As good as R is for statistics, it's as good if not better for plots.

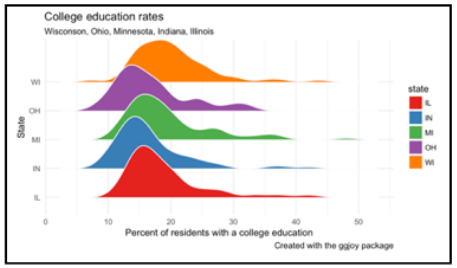












Plotting in R

There are different frameworks for doing plotting in R.

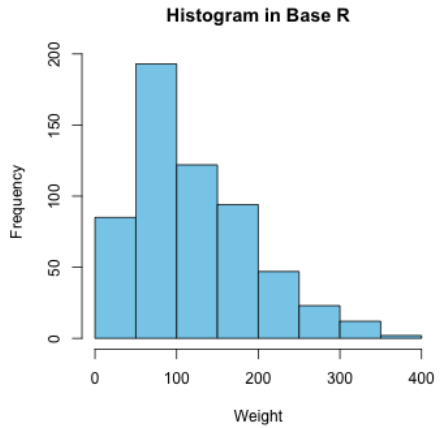
The classic framework is known as `base R` plotting.

In Base-R plotting, there is a separate function for each 'type' of plot

Plot type	Function
Bar plot	`barplot()`
Box plot	`boxplot()`
Scatterplot	`plot()`
Pirateplot	`pirateplot()`

```
# Histogram in base R

hist(x = ChickWeight$weight,
     xlab = "Weight",
     ylab = "Frequency",
     col = "skyblue",
     main = "Histogram in Base R")
```



Plotting in R

There are different frameworks for doing plotting in R.

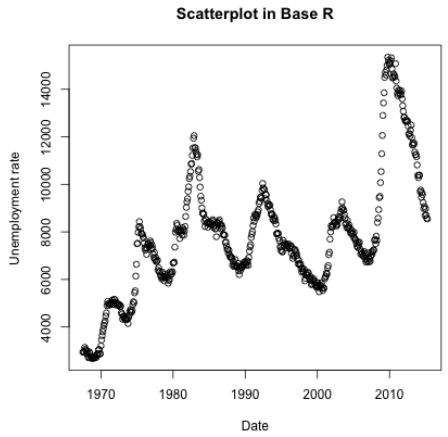
The classic framework is known as `base R` plotting.

In Base-R plotting, there is a separate function for each 'type' of plot

Plot type	Function
Bar plot	`barplot()`
Box plot	`boxplot()`
Scatterplot	`plot()`
Pirateplot	`pirateplot()`

```
# Scatterplot in base R

plot(x = economics$date,
     y = economics$unemploy,
     xlab = "Date",
     ylab = "Unemployment rate",
     main = "Scatterplot in Base R"
     )
```



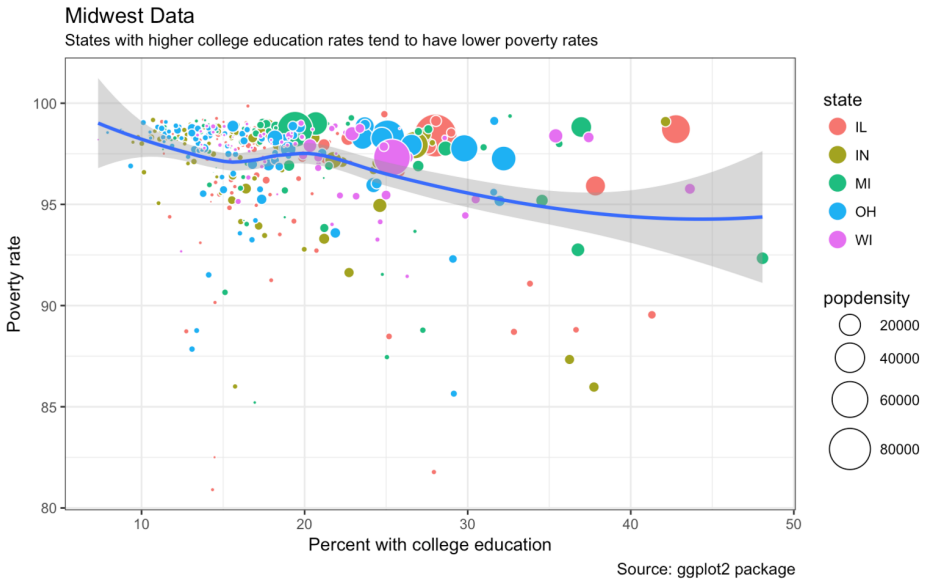
Plotting in R

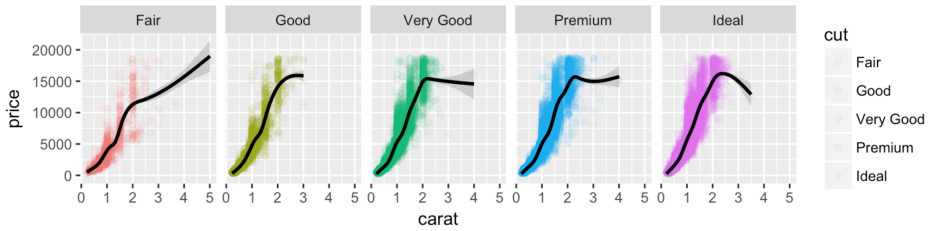
Problems with base R plotting:

- Complex plots can quickly require a lot of code.
- There is no unifying framework between plots. Every plot is on its own.
- It is difficult to parts of code in different plots.

Solution: Grammar of Graffics with `ggplot`

These plots would take a lot of code in base R





Grammar of Graphics

1. A plot is built of simple building blocks

2. By combining different building blocks, plots of any complexity can be created.

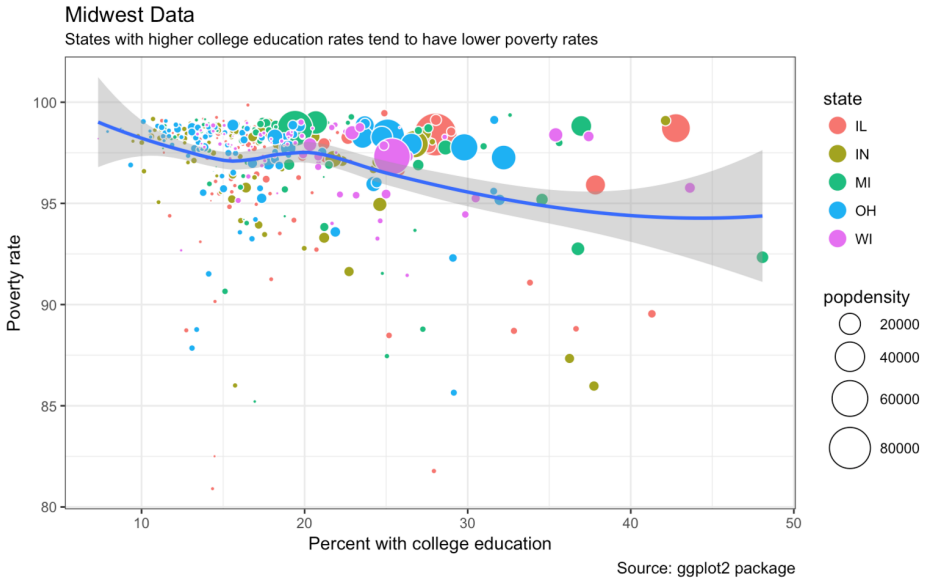
3. Plots that look superficially different, can actually be created with very similar code

Leland Wilkonson

Grammar of Graphics

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

aesthetics	Description
Data	What dataframe contains the data?
Aesthetics	What does the x-axis, y-axis, color (etc) represent?
Geometries	What kind of geometric object do you want to plot?
Facets	Should there be groups of plots?
Statistics	What statistic summaries / transformations should be done?
Coordinates	What is the scale of the axes?
Theme	What should the overall plot look like?



ggplot2

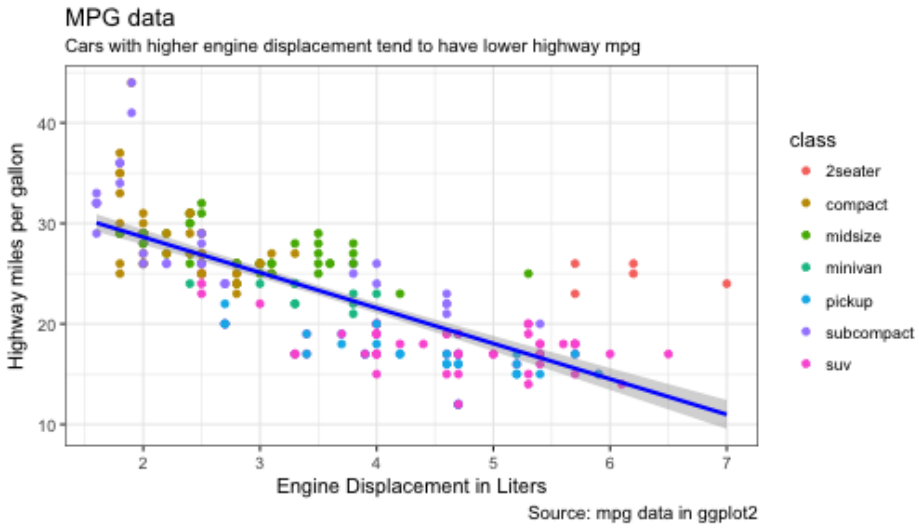
How do we make elegant, easy to program plots according to the grammar of graphics in R?

Answer: ggplot2

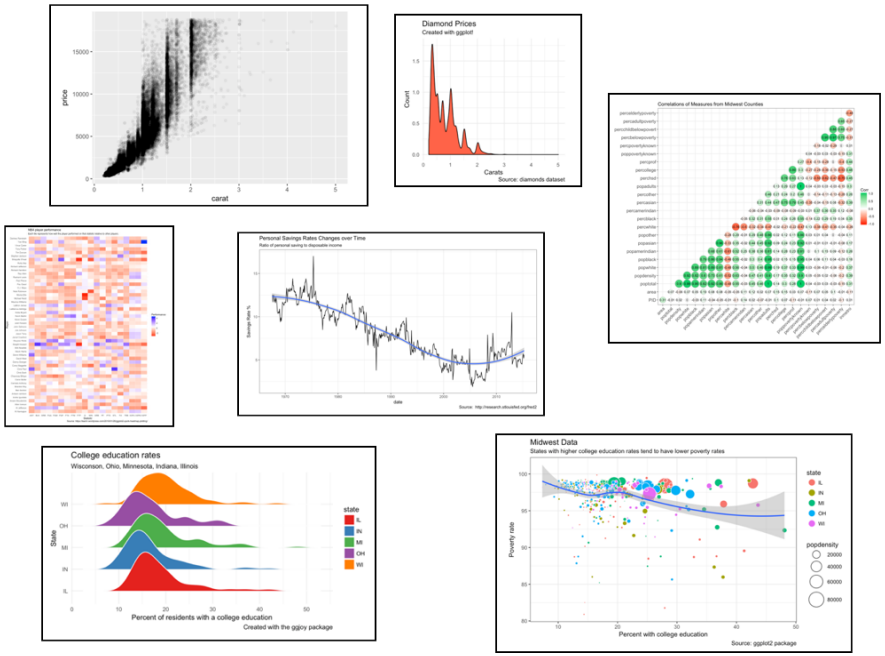
By far one of the most popular R packages, used to generate the vast majority of plots from R.

Our goal

- In this introduction, we'll introduce the basic building blocks of creating plots with ggplot2 by creating the following plot from the ground-up:



In the practical, you will create all of these!!



ggplot2

Load the ggplot2 package

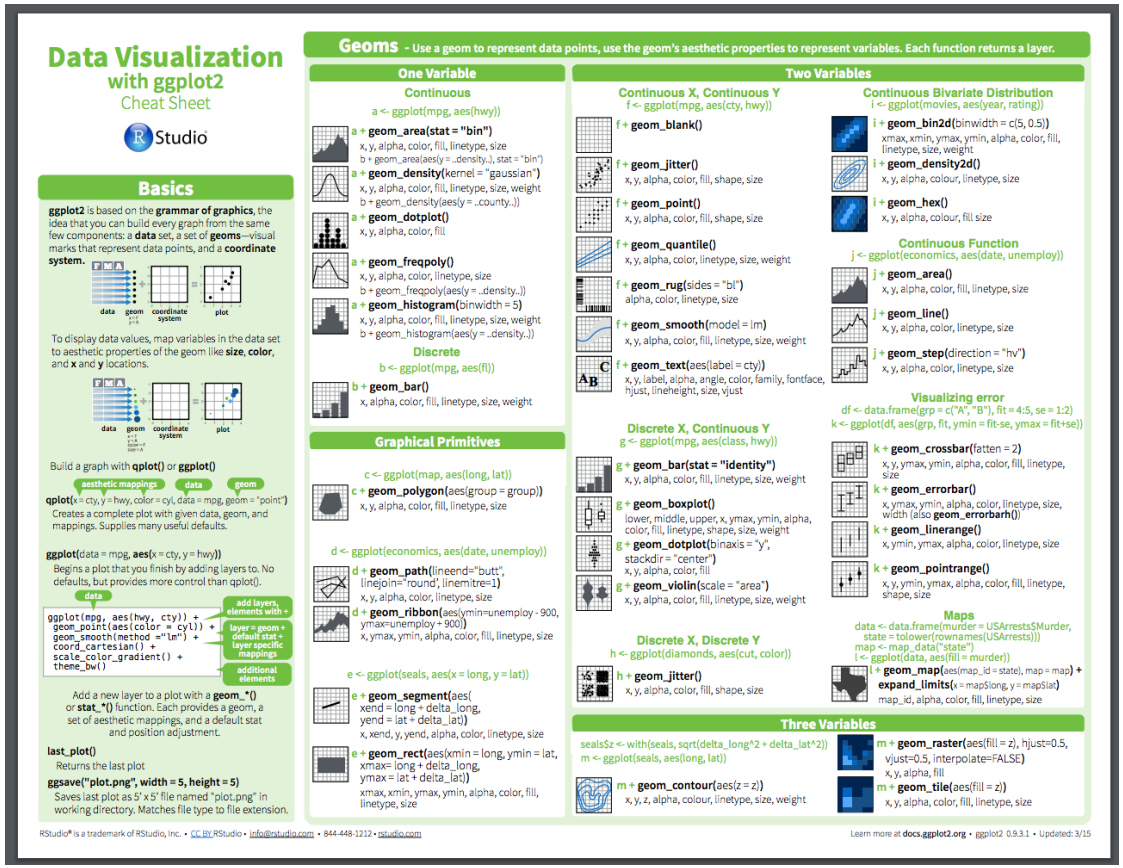
```
# Load the tidyverse (includes ggplot2
library(tidyverse)
```

Or

```
# Load ggplot2 directly
library(ggplot2)
```

Cheat Sheet!

https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf



Data Visualization with ggplot2 Cheat Sheet

RStudio

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same few components: a **data** set, a set of **geoms**—visual marks that represent data points, and a **coordinate system**.

To display data values, map variables in the data set to aesthetic properties of the geom like **size**, **color**, and **x** and **y** locations.

Build a graph with **qplot()** or **ggplot()**

qplot(x = cty, y = hwy, color = cyl, data = mpg, geom = "point")
Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

ggplot(data = mpg, **aes**(x = cty, y = hwy))
Begins a plot that you finish by adding layers to. No defaults, but provides more control than qplot().

```
ggplot(mpg, aes(hwy, cty)) +
geom_point(aes(color = cyl)) +
geom_smooth(method ="lm") +
coord_cartesian() +
scale_color_gradient() +
theme_bw()
```

add layers, elements with +
layer = geom + default stat + layer specific mappings
additional elements

Add a new layer to a plot with a **geom_*()** or **stat_*()** function. Each provides a geom, a set of aesthetic mappings, and a default stat and position adjustment.

last_plot()
Returns the last plot

ggsave("plot.png", width = 5, height = 5)
Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.

Geoms - Use a geom to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

One Variable

Continuous

a <- ggplot(mpg, aes(hwy))

- a + **geom_area(stat = "bin")** — x, y, alpha, color, fill, linetype, size; b + geom_area(aes(y = ..density..), stat = "bin")
- a + **geom_density(kernel = "gaussian")** — x, y, alpha, color, fill, linetype, size, weight; b + geom_density(aes(y = ..county..))
- a + **geom_dotplot()** — x, y, alpha, color, fill
- a + **geom_freqpoly()** — x, y, alpha, color, linetype, size; b + geom_freqpoly(aes(y = ..density..))
- a + **geom_histogram(binwidth = 5)** — x, y, alpha, color, fill, linetype, size, weight; b + geom_histogram(aes(y = ..density..))

Discrete

b <- ggplot(mpg, aes(fl))

- b + **geom_bar()** — x, alpha, color, fill, linetype, size, weight

Graphical Primitives

c <- ggplot(map, aes(long, lat))

- c + **geom_polygon**(aes(group = group)) — x, y, alpha, color, fill, linetype, size

d <- ggplot(economics, aes(date, unemploy))

- d + **geom_path**(lineend="butt", linejoin="round", linemitre=1) — x, y, alpha, color, linetype, size
- d + **geom_ribbon**(aes(ymin=unemploy - 900, ymax=unemploy + 900)) — x, ymax, ymin, alpha, color, fill, linetype, size

e <- ggplot(seals, aes(x = long, y = lat))

- e + **geom_segment**(aes(xend = long + delta_long, yend = lat + delta_lat)) — x, xend, y, yend, alpha, color, linetype, size
- e + **geom_rect**(aes(xmin = long, ymin = lat, xmax= long + delta_long, ymax = lat + delta_lat)) — xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

Two Variables

Continuous X, Continuous Y

f <- ggplot(mpg, aes(cty, hwy))

- f + **geom_blank()**
- f + **geom_jitter()** — x, y, alpha, color, fill, shape, size
- f + **geom_point()** — x, y, alpha, color, fill, shape, size
- f + **geom_quantile()** — x, y, alpha, color, linetype, size, weight
- f + **geom_rug(sides = "bl")** — alpha, color, linetype, size
- f + **geom_smooth(model = lm)** — x, y, alpha, color, fill, linetype, size, weight
- f + **geom_text(aes(label = cty))** — x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

Discrete X, Continuous Y

g <- ggplot(mpg, aes(class, hwy))

- g + **geom_bar(stat = "identity")** — x, y, alpha, color, fill, linetype, size, weight
- g + **geom_boxplot()** — lower, middle, upper, x, ymax, ymin, alpha, color, fill, linetype, shape, size, weight
- g + **geom_dotplot(binaxis = "y", stackdir = "center")** — x, y, alpha, color, fill
- g + **geom_violin(scale = "area")** — x, y, alpha, color, fill, linetype, size, weight

Discrete X, Discrete Y

h <- ggplot(diamonds, aes(cut, color))

- h + **geom_jitter()** — x, y, alpha, color, fill, shape, size

Continuous Bivariate Distribution

i <- ggplot(movies, aes(year, rating))

- i + **geom_bin2d(binwidth = c(5, 0.5))** — xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size, weight
- i + **geom_density2d()** — x, y, alpha, colour, linetype, size
- i + **geom_hex()** — x, y, alpha, colour, fill size

Continuous Function

j <- ggplot(economics, aes(date, unemploy))

- j + **geom_area()** — x, y, alpha, color, fill, linetype, size
- j + **geom_line()** — x, y, alpha, color, linetype, size
- j + **geom_step(direction = "hv")** — x, y, alpha, color, linetype, size

Visualizing error

df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
k <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))

- k + **geom_crossbar(fatten = 2)** — x, y, ymax, ymin, alpha, color, fill, linetype, size
- k + **geom_errorbar()** — x, ymax, ymin, alpha, color, linetype, size, width (also **geom_errorbarh()**)
- k + **geom_linerange()** — x, ymin, ymax, alpha, color, linetype, size
- k + **geom_pointrange()** — x, y, ymin, ymax, alpha, color, fill, linetype, shape, size

Maps

data <- data.frame(murder = USArrests$Murder, state = tolower(rownames(USArrests)))
map <- map_data("state")
l <- ggplot(data, aes(fill = murder))

- l + **geom_map**(aes(map_id = state), map = map) + **expand_limits**(x = mapSlong, y = mapSlat) — map_id, alpha, color, fill, linetype, size

Three Variables

seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2))
m <- ggplot(seals, aes(long, lat))

- m + **geom_contour**(aes(z = z)) — x, y, z, alpha, colour, linetype, size, weight
- m + **geom_raster**(aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE) — x, y, alpha, fill
- m + **geom_tile**(aes(fill = z)) — x, y, alpha, color, fill, linetype, size

mpg data

The `mpg` data is a tibble of car data contained in the `ggplot2` package

manufacturer	model	cty	hwy	class
land rover	range rover	11	15	suv
hyundai	sonata	19	28	midsize
toyota	camry	21	31	midsize
jeep	grand cherokee 4wd	15	20	suv
chevrolet	c1500 suburban 2wd	14	20	suv
ford	mustang	15	23	subcompact
volkswagen	new beetle	29	41	subcompact
ford	expedition 2wd	11	17	suv

Creating this plot

Data

- Use the `mpg` tibble

Aesthetics

- Show engine displacement (`disp`) on the x axis
- Show highway miles per gallon (`hwy`) on the y-axis
- Color plotting elements by the class of car (`class`)

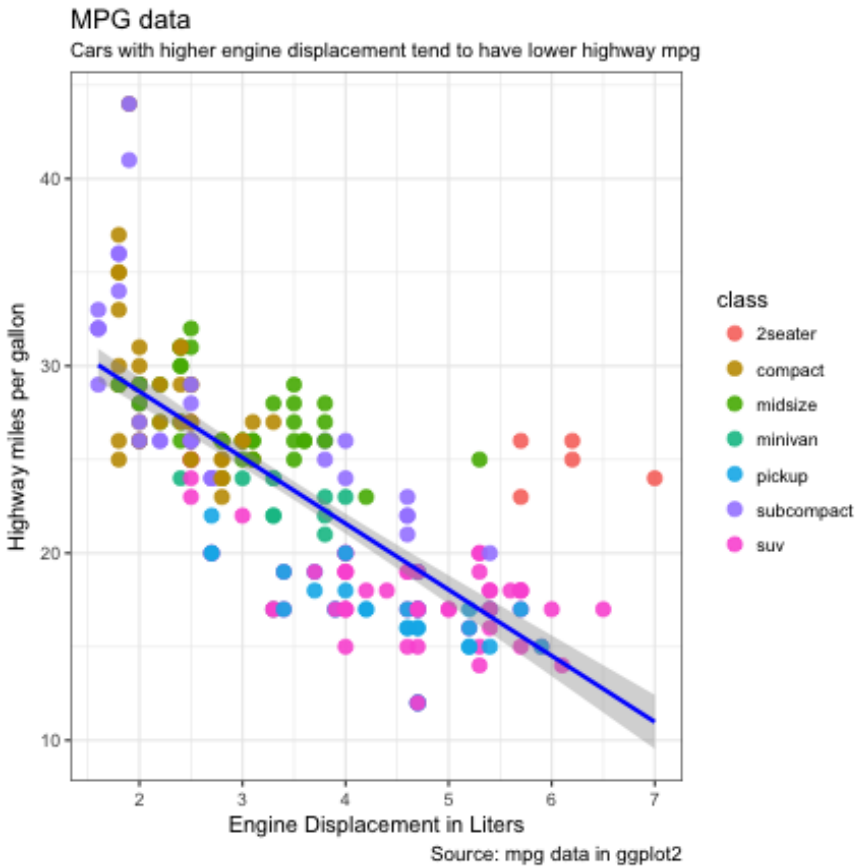
Geometric objects

- Show data as points.
- Add a regression line

Labels and themes

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

Our goal is to build the following plot step by step:



`data`

- To create a ggplot2 object, use the `ggplot()` function.
- Start with the `data` argument.
    - `data` should be a dataframe (or Tibble)
- Including only a `data` argument returns a blank plotting space, because we haven't specified any plotting *aesthetics* or *geometric objects*

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

Aesthetics

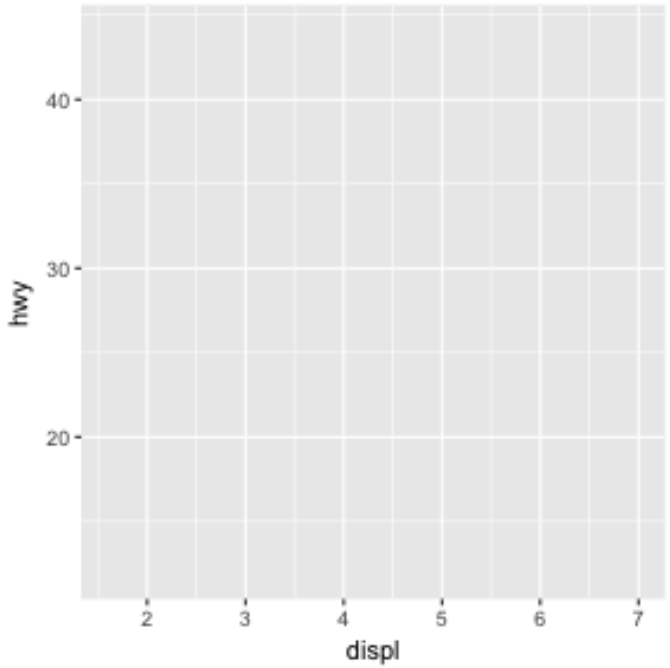
- An **aesthetic** is a visual property of the objects in your plot.

Common aesthetics

aesthetics	Description
`x`	Data mapped to x-coordinate
`y`	Data mapped to y-coordinate
`color`, `fill`	Color and filling
`alpha`	Transparency
`shape`	Overall shape
`size`	Size

- Add plotting aesthetics with the `aes()` function

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

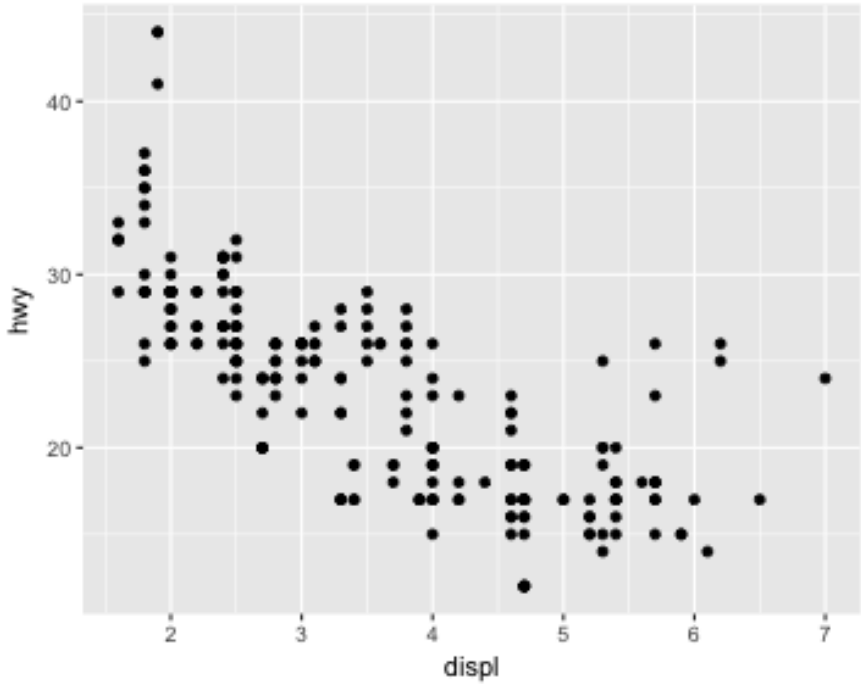


Adding elements to plots with '+'

- Once you have specified data with `data` argument, and global aesthetics with `mapping = aes()`, you can add additional elements to the plot with `+`
- The `+` sign works just like the pipe `%>%` in dplyr. It just means "and then..."

```
ggplot(data = mpg,
       mapping = aes(x = displ, y = hw
       # MORE    + (and then...)
       # MORE    + (and then...)
```

ggplot uses + to add additional elements to a plot



Geometric objects (geoms)

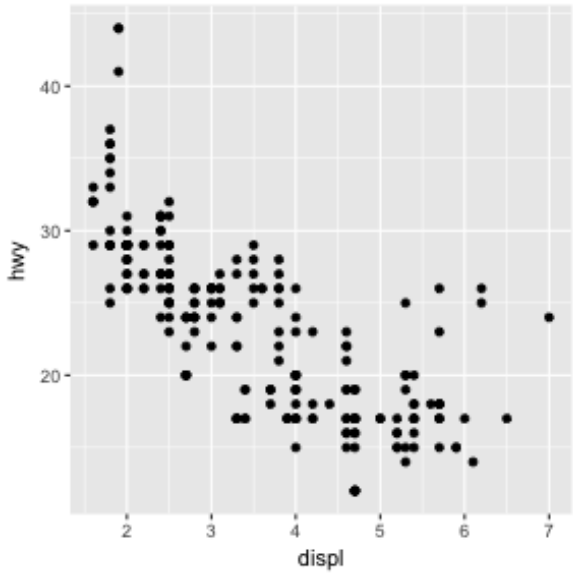
- A **geom** is a geometric object in a plot that represents data
- To add a geom to a plot, just include + `geom_X()` where X is the type of geom.

Common geoms

geom	plot type
`geom_point()`	Scatterplot
`geom_bar()`	barplot
`geom_boxplot()`	A boxplot
`geom_count()`	Points representing counts
`geom_smooth()`	Regression line

Add a point geom with `geom_point()`

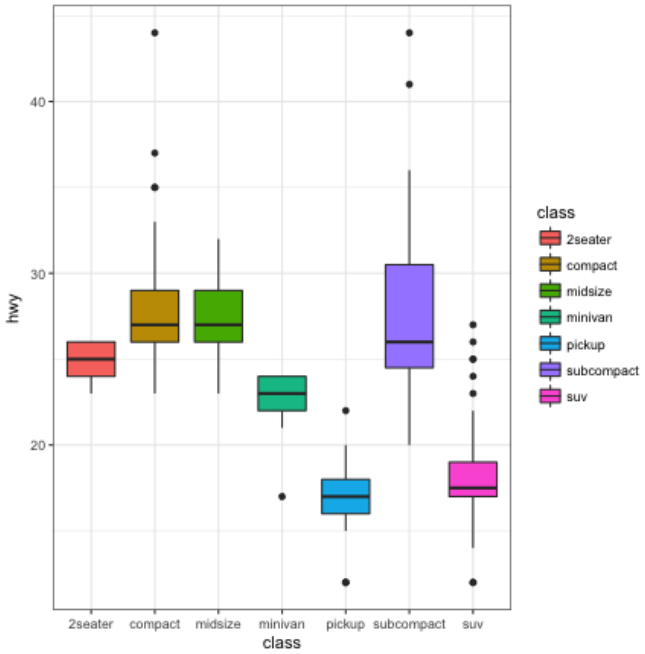
```
ggplot(data = mpg,
       mapping = aes(x = displ, y = hwy)) +
  geom_point()
```



Additional geoms

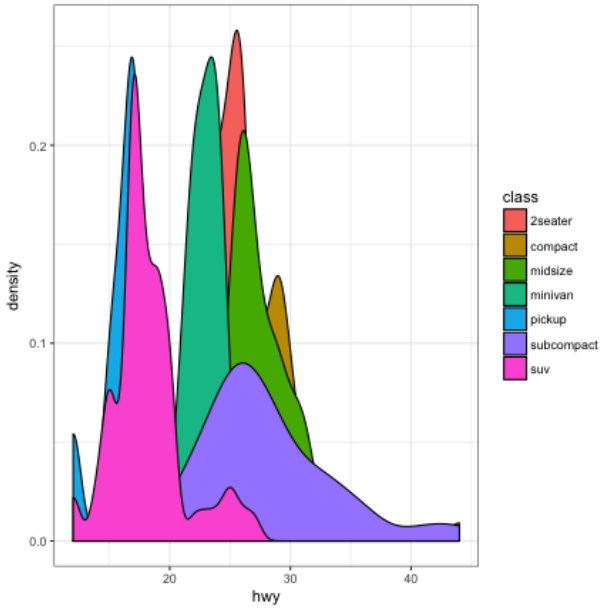
geom_boxplot()

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



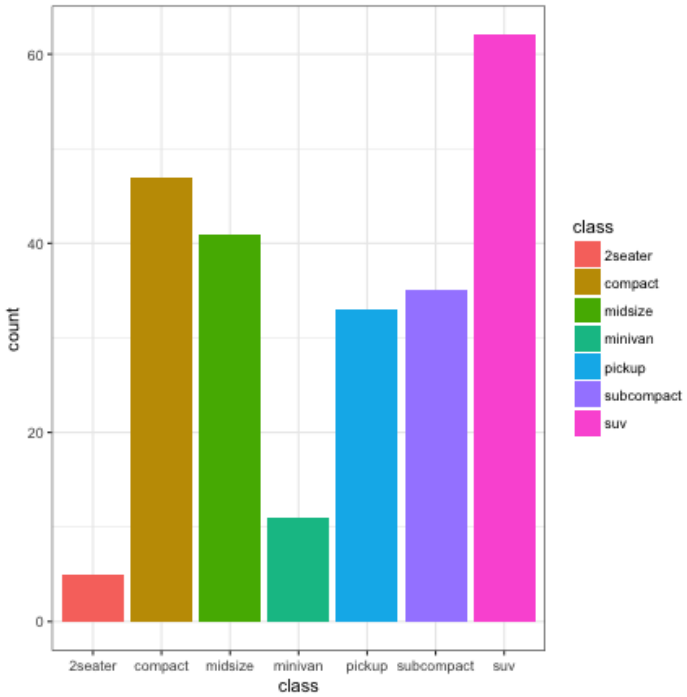
geom_density()

```
ggplot(data = mpg,
  mapping = aes(x = hwy, fill = class)) +
  geom_density()  +
  theme_bw()
```



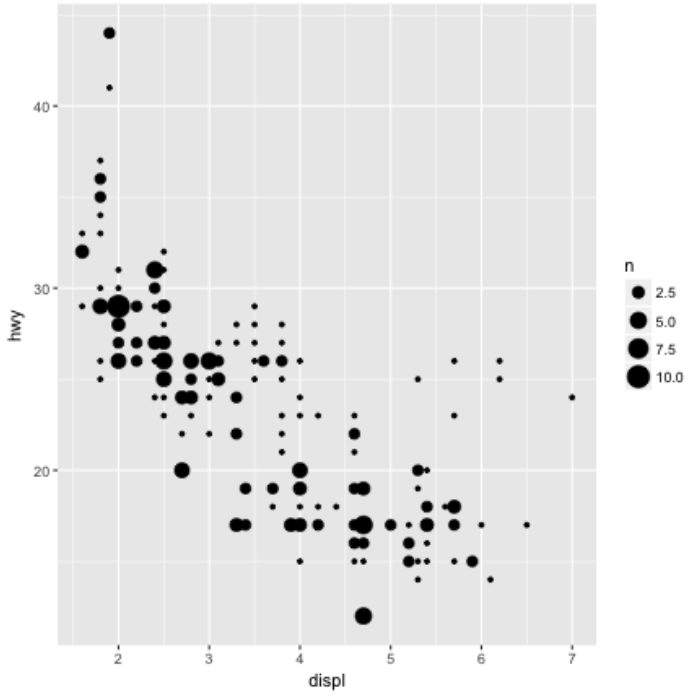
geom_bar()

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



geom_count()

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

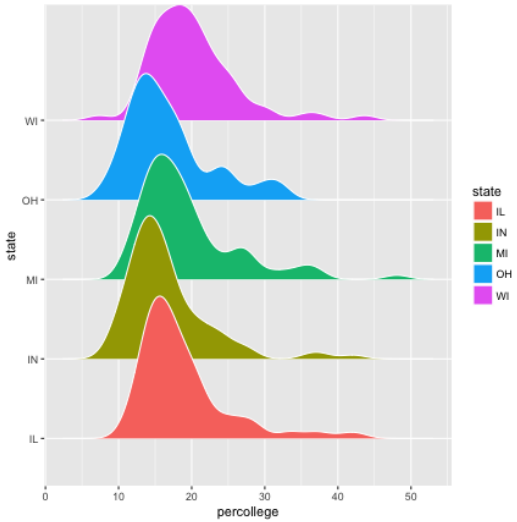


geom_joy()

> From the ggjoy package

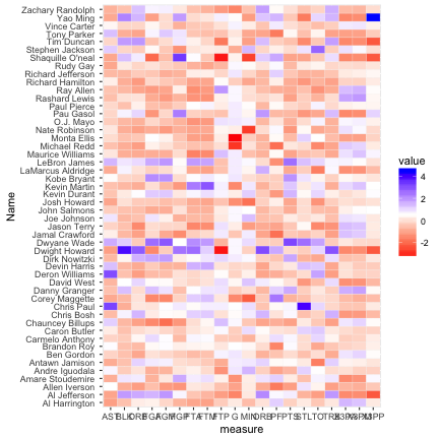
```
library(ggjoy) # Load the ggjoy package

ggplot(data = midwest,
       mapping = aes(percollege,
                     y = state,
                     fill = state)) +
  geom_joy(col = "white")
```



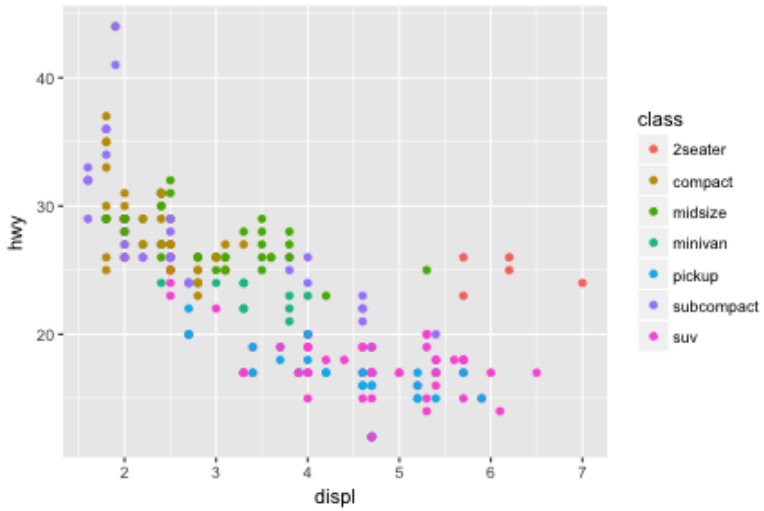
geom_tile()

```
ggplot(nba_long,
       mapping = aes(x = Name,
                     y = measure,
                     fill = value)) +
  geom_tile(col = "white") +
  scale_fill_gradientn(colors = c("red",
                                  "white",
                                  "blue"))
  coord_flip()
```



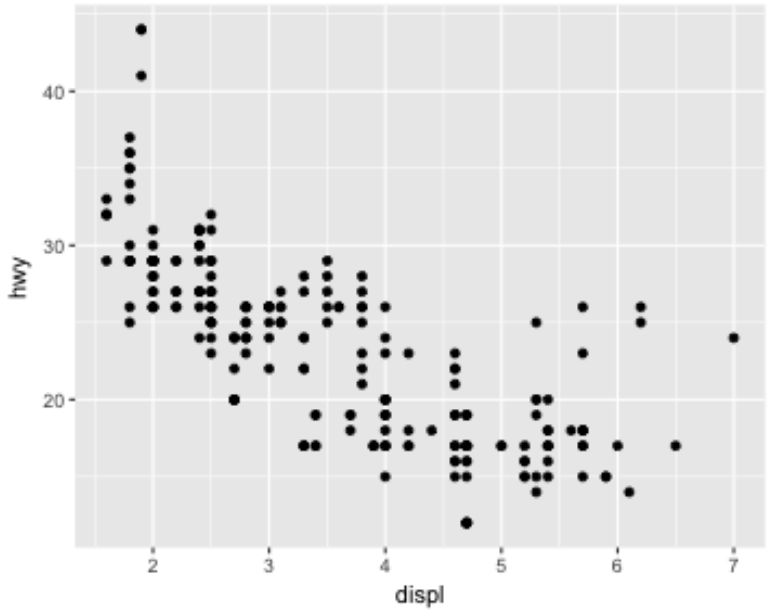
Geometric objects (geoms)

How do we get points to be shown in different colors like the plot below?



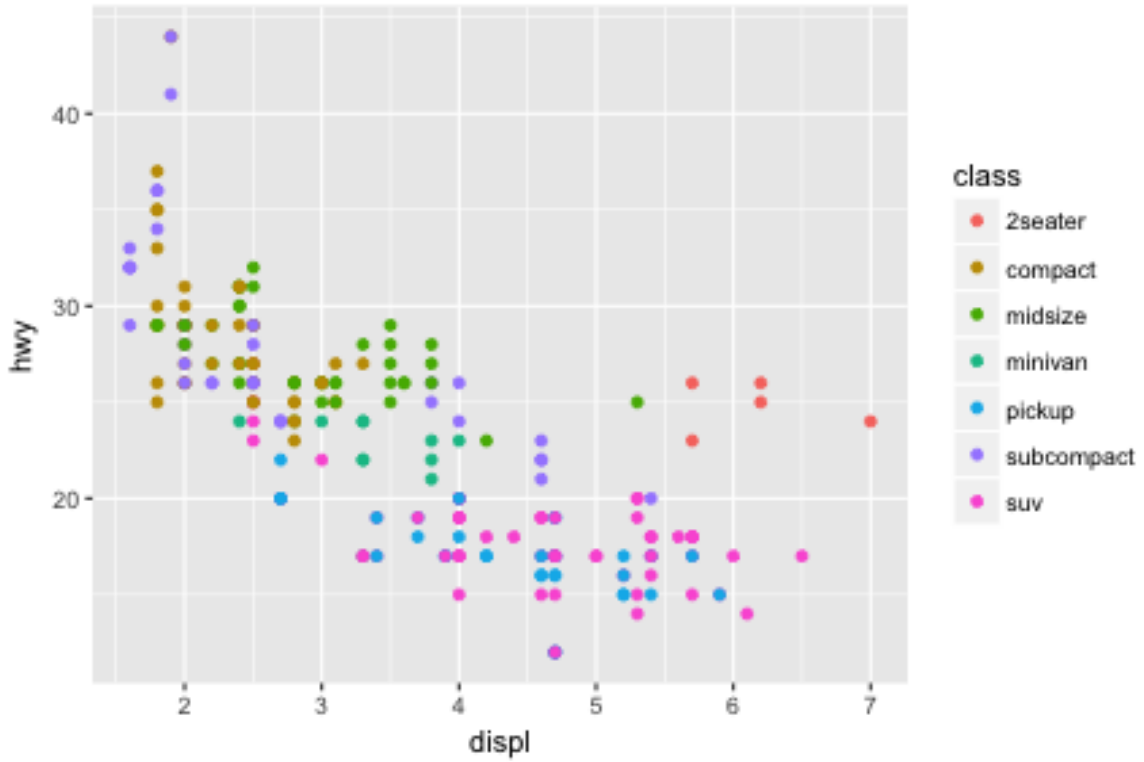
> What plotting aesthetic is missing from ths code below to make our plot on the left?

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



Color aesthetic

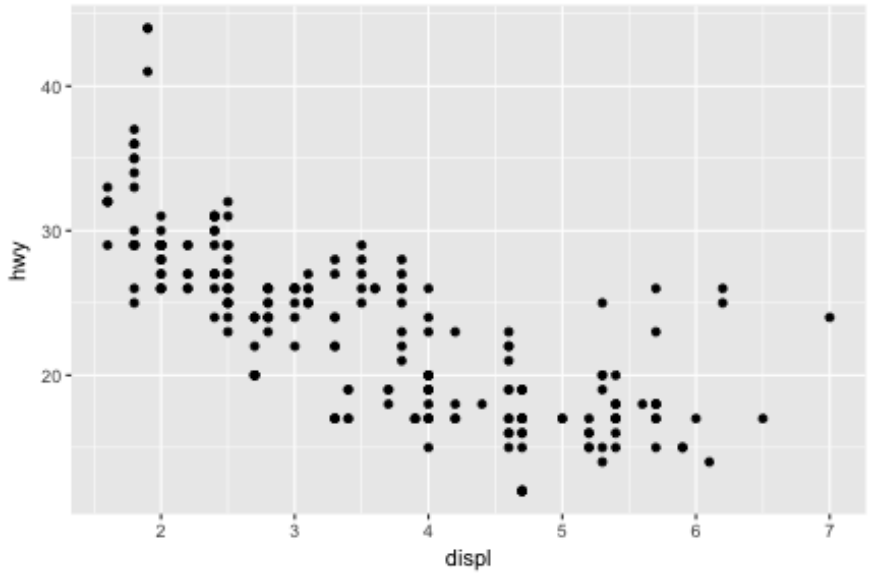
```
ggplot(data = mpg,
       mapping = aes(x = displ, y = hwy, col = class)) +  # Map mpg$class to color aes
  geom_point()   # Add points that respect the aesthetics
```



Geometric objects (geoms)

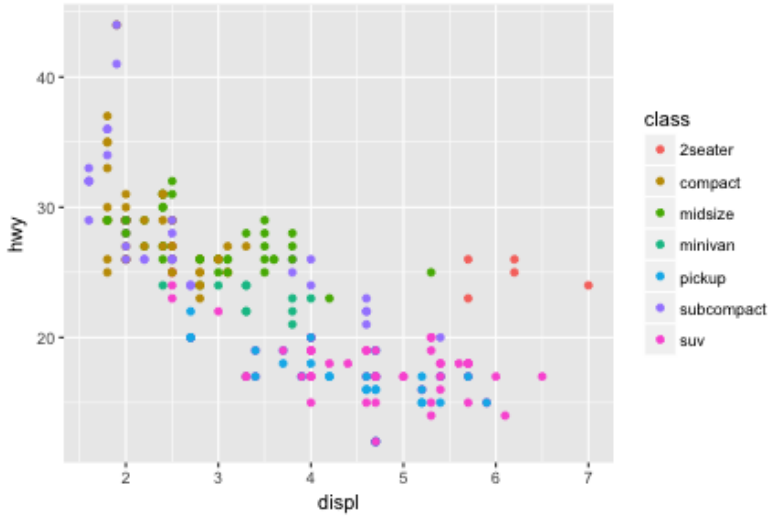
Code WITHOUT color aesthetic

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



Code WITH color aesthetic

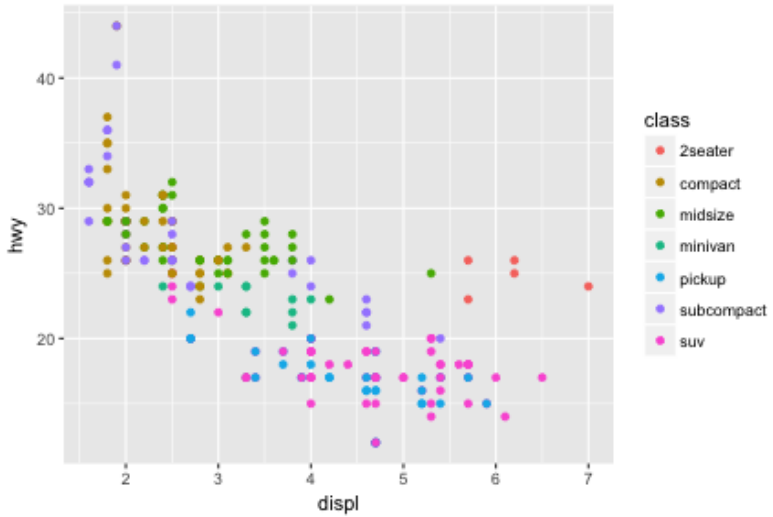
```
ggplot(data = mpg,
       mapping = aes(x = displ,
                     y = hwy,
                     col = class)) +
  geom_point()
```



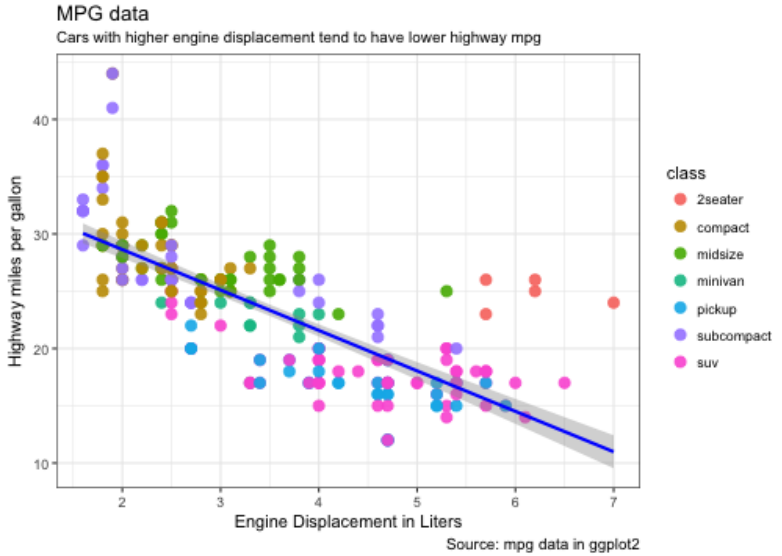
What's next?

> Where are at

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



Our goal



Smoothed lines with `geom_smooth()`

To add a smoothed line to a plot, use `geom_smooth()`

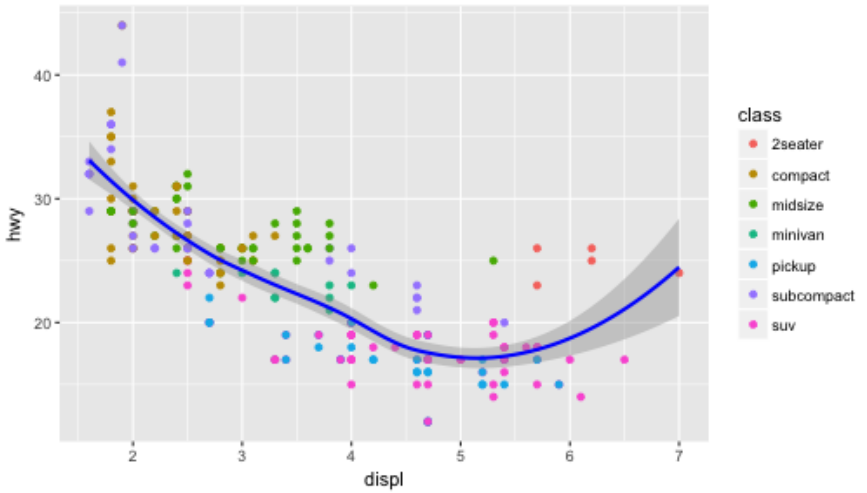
`geom_smooth()` arguments

Arguments	Description
`method`	How should the line be generated?
`level`	Confidence
`col, size, ...`	Other plotting aesthetics

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

Add a blue smoothed line

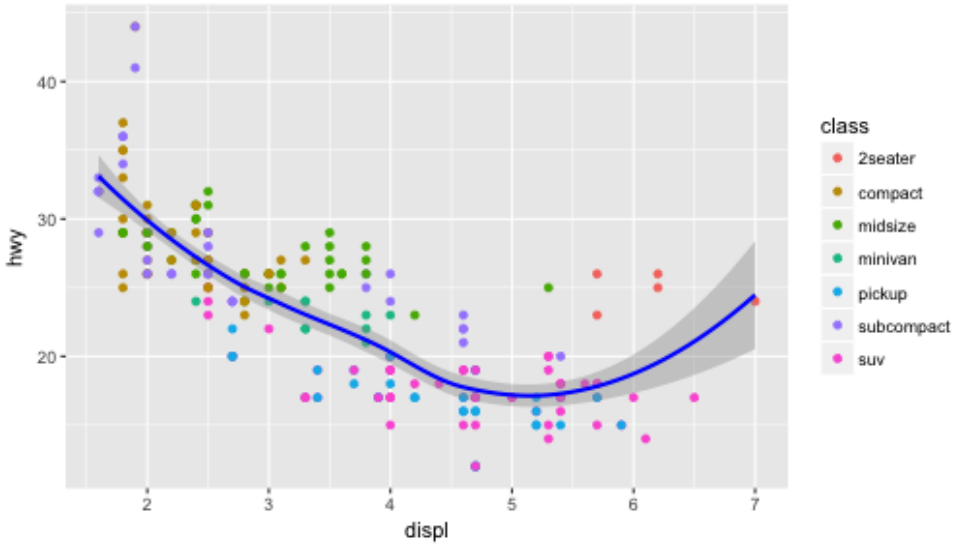
```
ggplot(data = mpg,
       mapping = aes(x = displ, y = hwy, co
  geom_point() +
  geom_smooth(col = "blue")
```



Overriding aesthetics

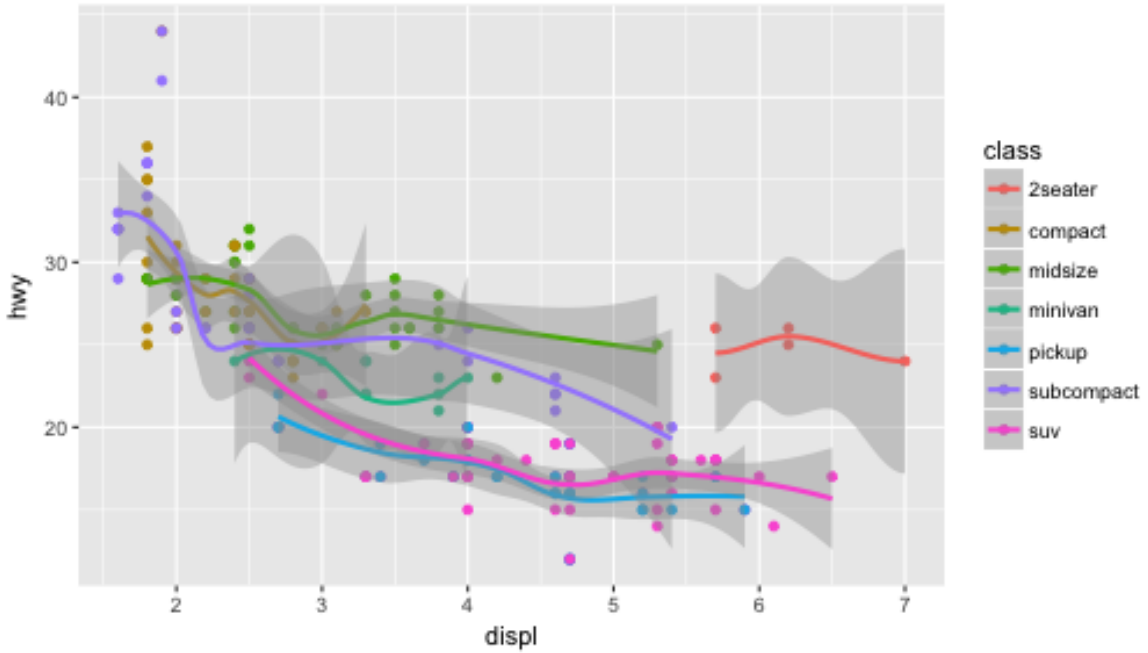
- You can include additional aesthetics, like color, shape, and size, in *any* geom.
- This will override the *global* aesthetics

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



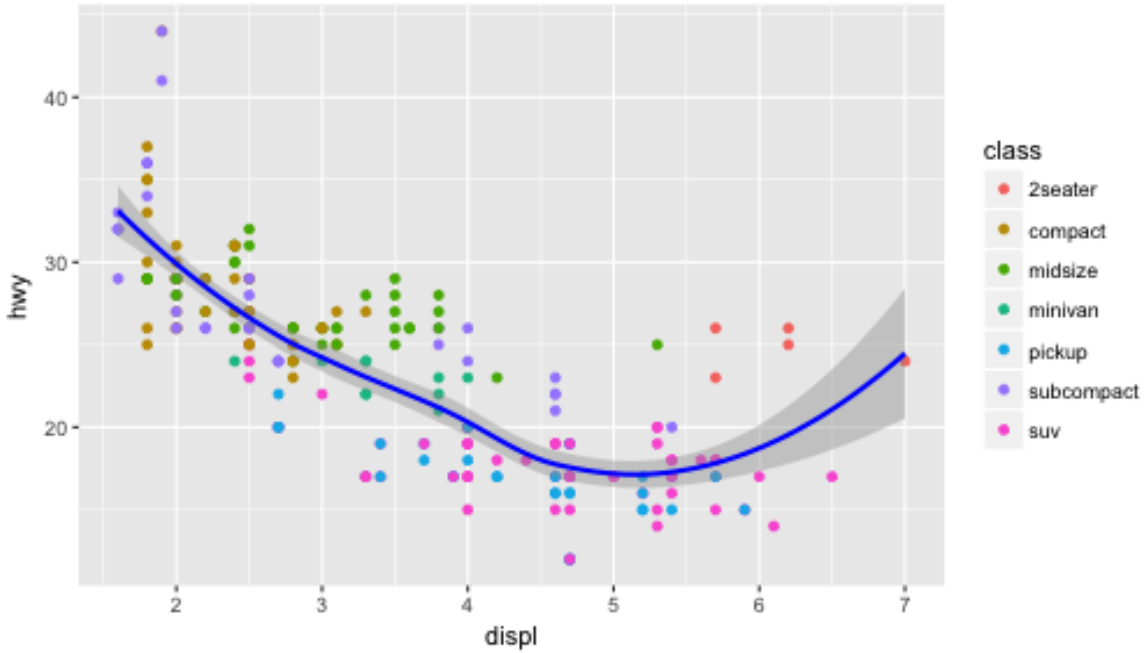
Example A: No overriding

```
ggplot(data = mpg,
       mapping = aes(x = displ, y = hwy, col = class)) +
  geom_point() + # geom_point RESPECTS global col aesthetic (class)
  geom_smooth()  # geom_smooth RESPECTS global col aesthetic (class)
```



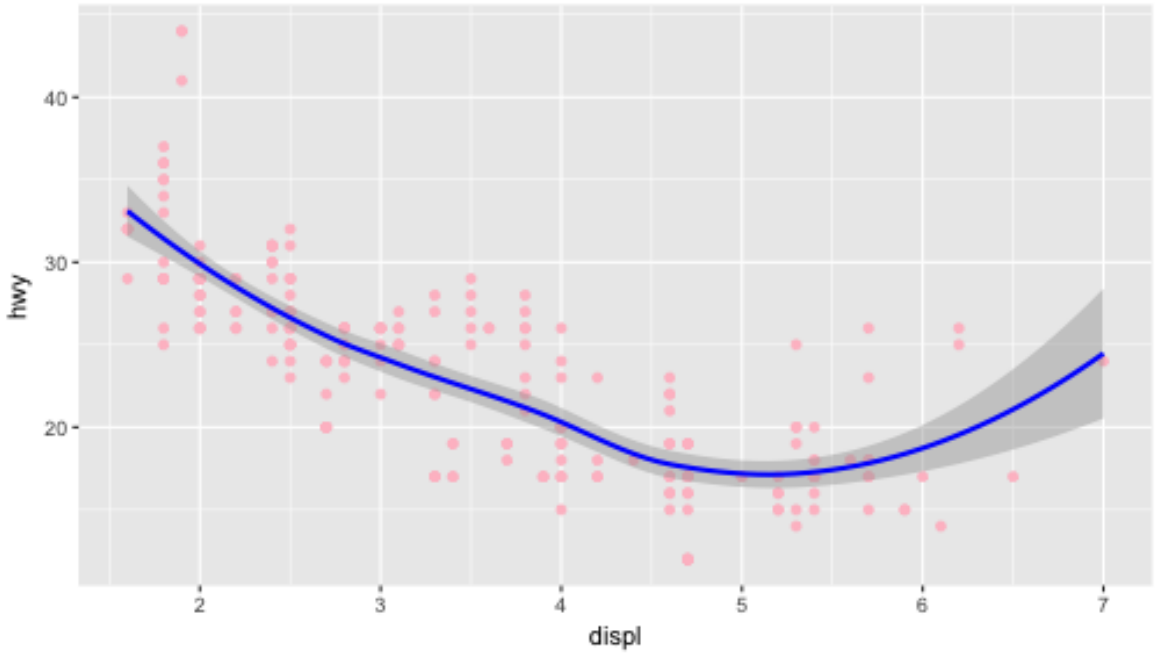
Example B: Override col aesthetic in geom_smooth()

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



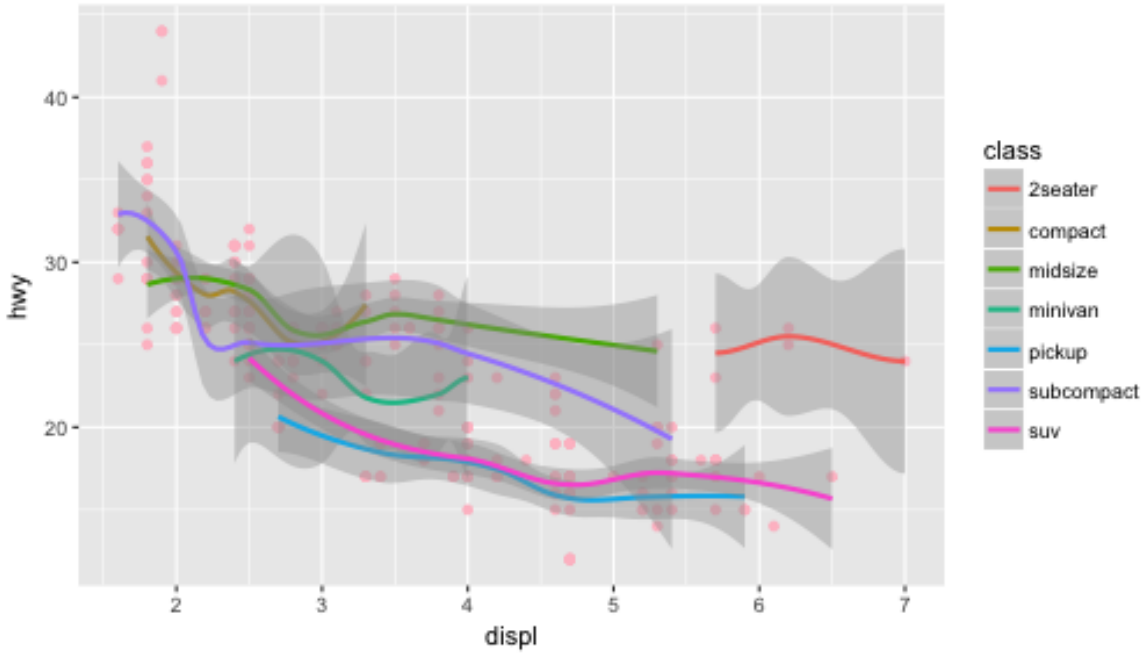
Example C: Override col aesthetic in geom_smooth() & geom_point()

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



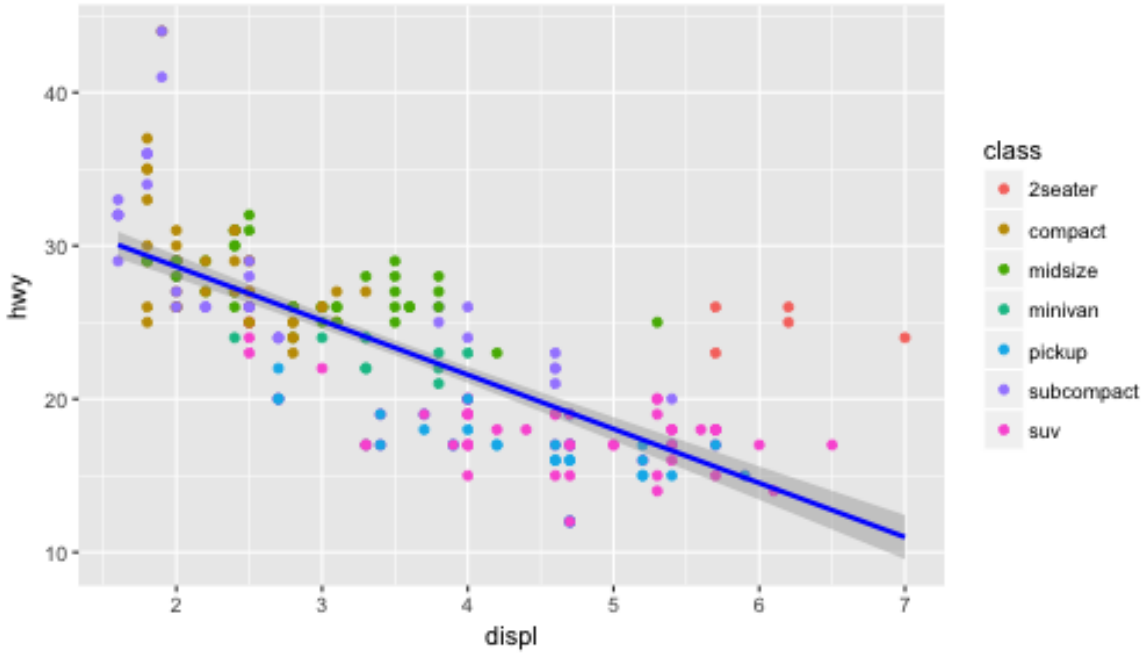
Example D: Override col aesthetic in geom_point()

```
ggplot(data = mpg,
       mapping = aes(x = displ, y = hwy, col = class)) +
  geom_point(col = "pink") + # Overrides global col aesthetic (class)
  geom_smooth()              #
```



What we want

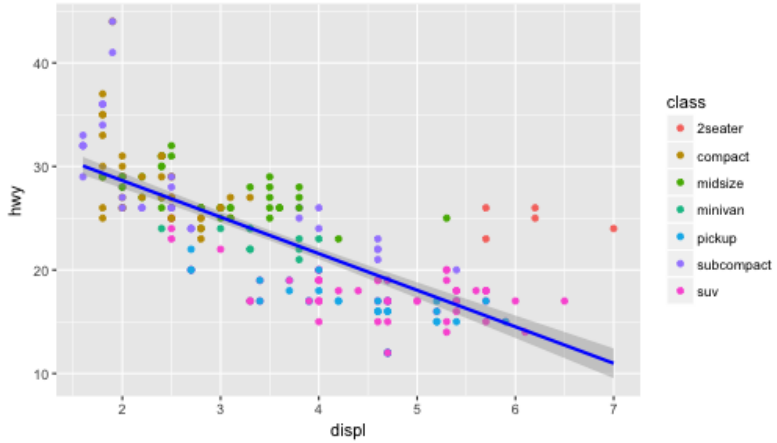
```
ggplot(data = mpg,
       mapping = aes(x = displ, y = hwy, col = class)) +
  geom_point() +
  geom_smooth(col = "blue",  # Overrides global col aesthetic (class)
              method = "lm") # Use lm (linear model) smoothing line
```



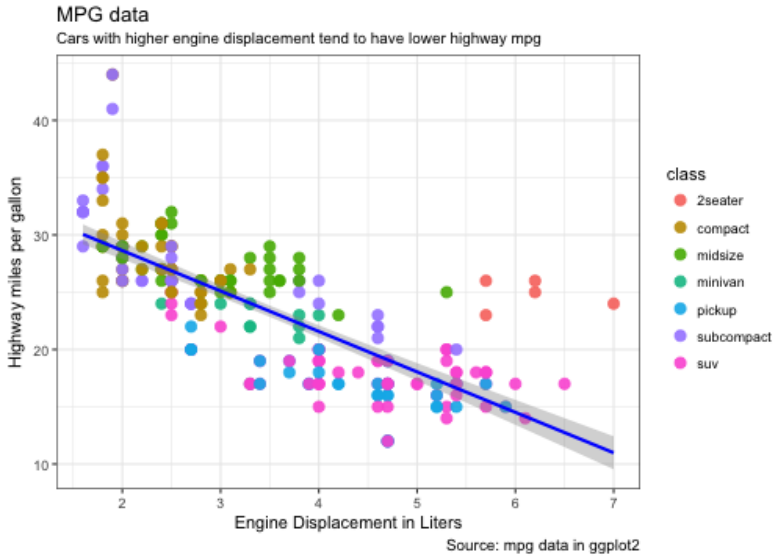
What's next?

Where we are at.

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



Our goal



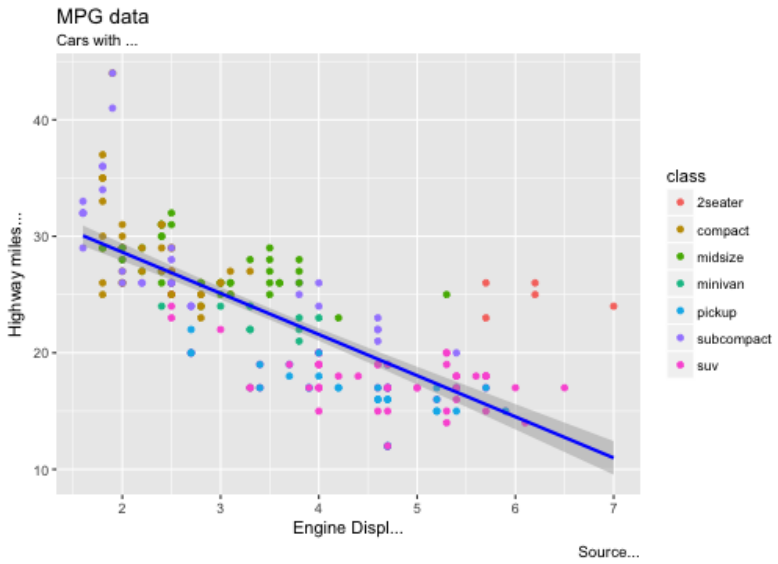
Add labels with `labs()`

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

`labs()` arguments

Arguments	Description
`title`	How should the line be generated?
`subtitle`	Confidence
`caption`	Caption
`col, size, ...`	Other plotting aesthetics

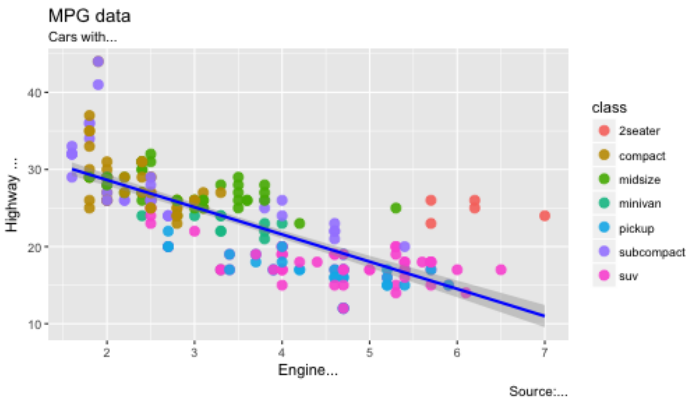
```
ggplot(data = mpg +
       # Past code +
  labs(x = "Engine Displ...",
       y = "Highway miles...",
       title = "MPG data",
       subtitle = "Cars with ...",
       caption = "Source...")
```



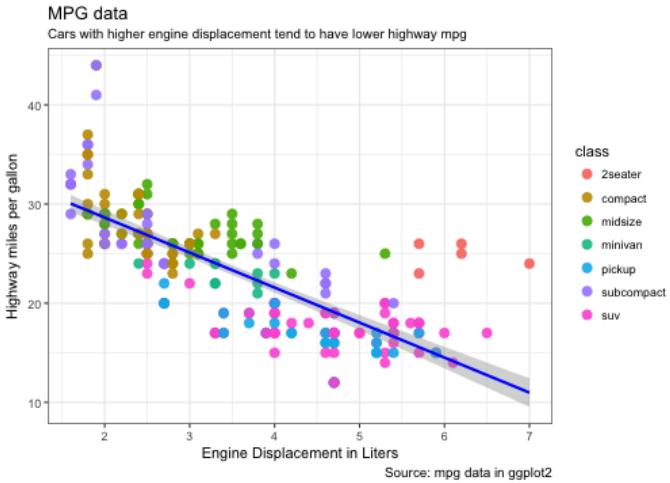
What's next?

Where are are at

```
ggplot(data = mpg,
       mapping = aes(x = displ, y = hwy, col = class)) +
  geom_point(size = 3, alpha = .9) +
  geom_smooth(col = "blue", method = "lm")+
  labs(x = "Engine...",
       y = "Highway ...",
       title = "MPG data",
       subtitle = "Cars with...",
       caption = "Source:...")
```



Our goal



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.

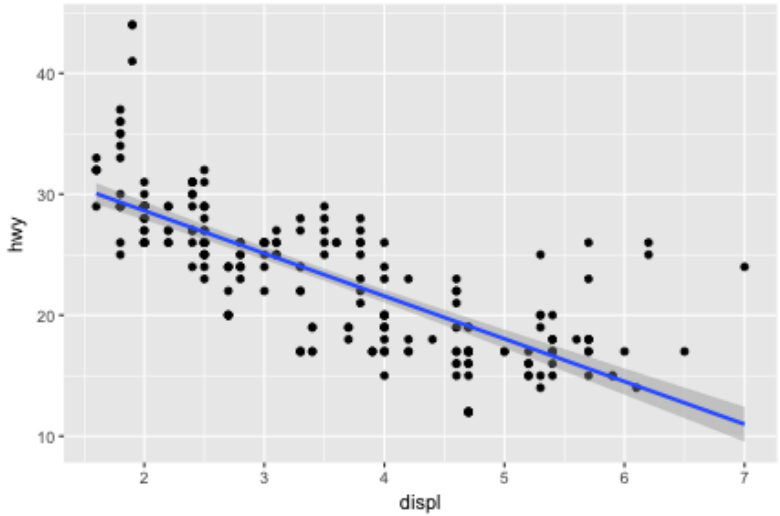
Common themes

Themes
`theme_bw()`
`theme_minimal()`
`theme_classic()`
`theme_light()`
`theme_dark()`

- You can easily add a theme to a plot by including `+ theme_XX()`

No theme specified

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



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.

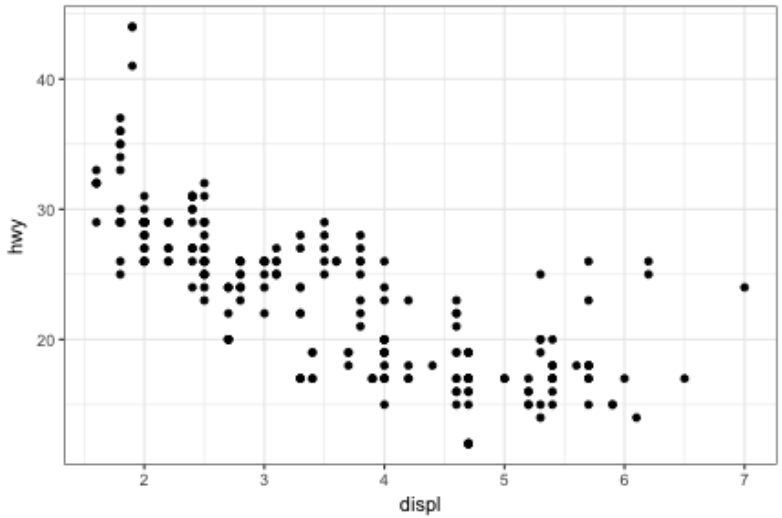
Common themes

Themes
`theme_bw()`
`theme_minimal()`
`theme_classic()`
`theme_light()`
`theme_dark()`

- You can easily add a theme to a plot by including `+ theme_XX()`

+ theme_bw()

```
ggplot(data = mpg,
       mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  theme_bw()   # Use the black and white th
```



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.

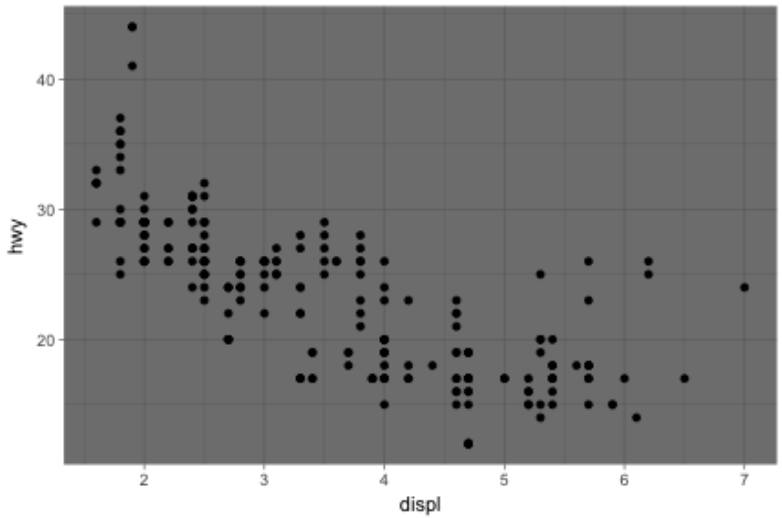
Common themes

Themes
`theme_bw()`
`theme_minimal()`
`theme_classic()`
`theme_light()`
`theme_dark()`

- You can easily add a theme to a plot by including `+ theme_XX()`

+ theme_dark()

```
ggplot(data = mpg,
       mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  theme_dark()   # Use the dark 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.

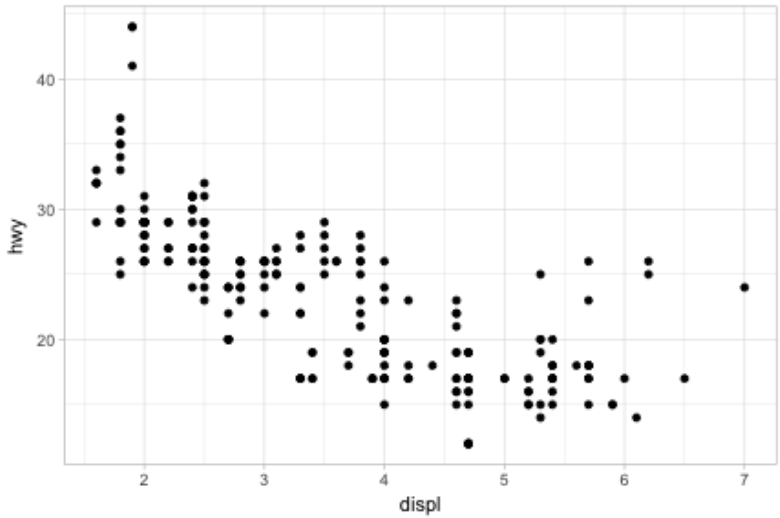
Common themes

Themes
`theme_bw()`
`theme_minimal()`
`theme_classic()`
`theme_light()`
`theme_dark()`

- You can easily add a theme to a plot by including `+ theme_XX()`

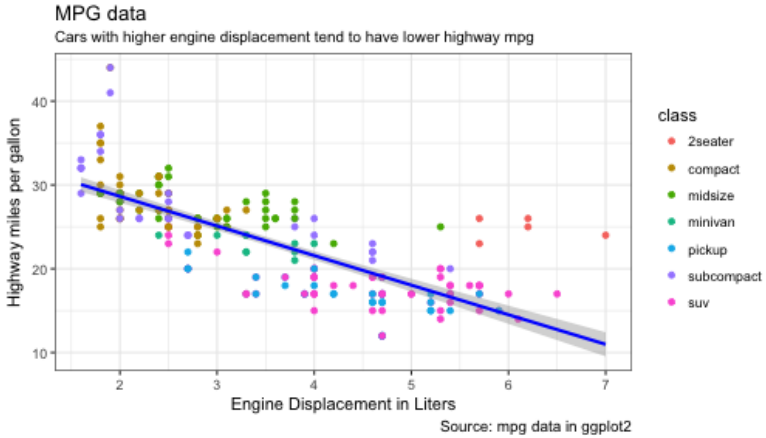
+ theme_light()

```
ggplot(data = mpg,
       mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  theme_light()   # Use the light theme
```



Final result!

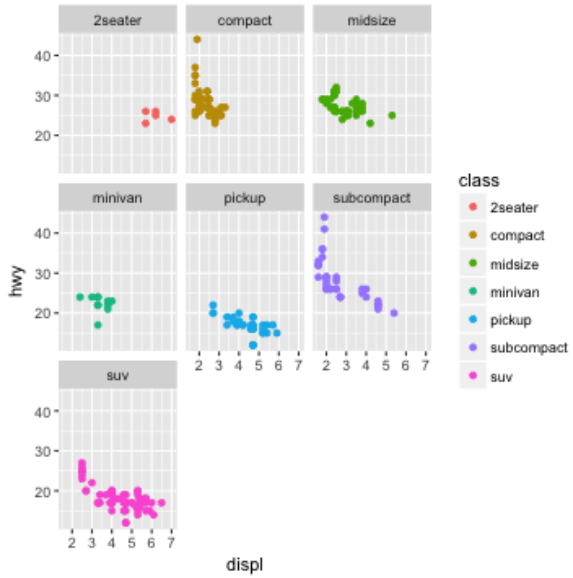
```
ggplot(data = mpg,
       mapping = aes(x = displ, y = hwy, col = class)) +
  geom_point() +
  geom_smooth(col = "blue", method = "lm")+
  labs(x = "Engine Displacement in Liters",
       y = "Highway miles per gallon",
       title = "MPG data",
       subtitle = "Cars with higher engine displacement tend to have lower highway mpg",
       caption = "Source: mpg data in ggplot2") +
  theme_bw()
```



Facetting with `facet_wrap()`

- *Facetting* = Create different plots for different groups
- To facet plots, use `facet_wrap()`

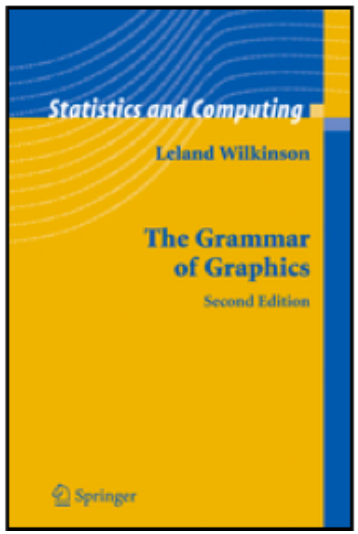
```
ggplot(data = mpg,
       mapping = aes(x = displ, y = hwy, color = class)) +
  geom_point()  +
  facet_wrap(~ class)
```



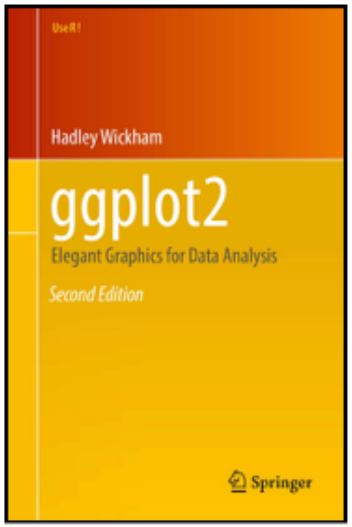
So much more

- We have only touched the surface of what you can do with `ggplot`
- http://ggplot2.tidyverse.org/index.html
- http://r4ds.had.co.nz/data-visualization.html

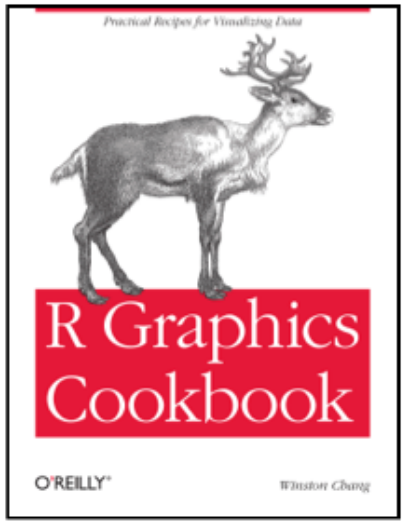
The Grammar of Graphics
Wilkinson



ggplot2
Wickham



R Graphics Cookbook
Cheng



Questions?

Plotting Pratical

Link to Plotting practical

<!--

-->

<!--

-->