



# BIMM 143

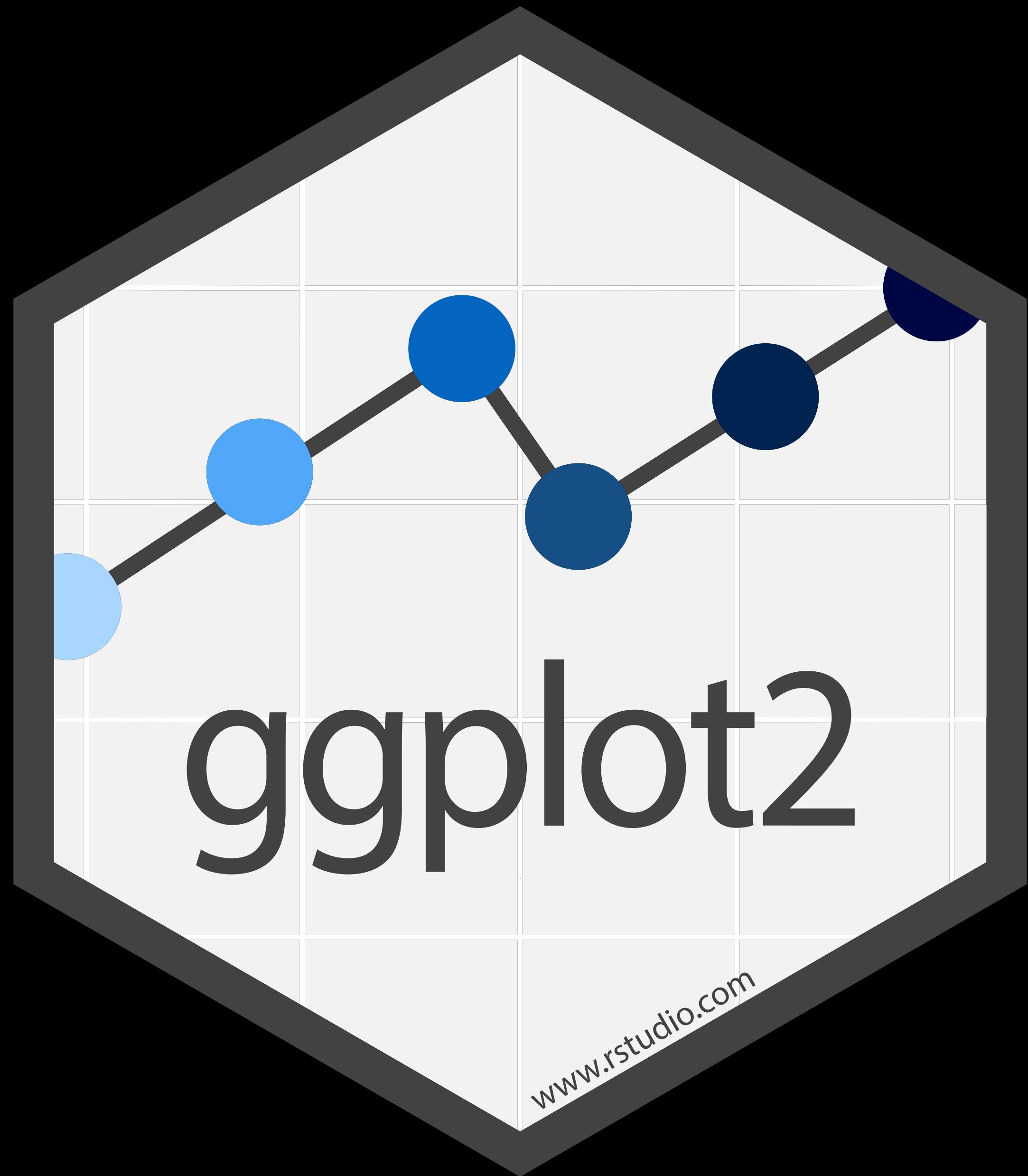
## Hands-on Lab Session

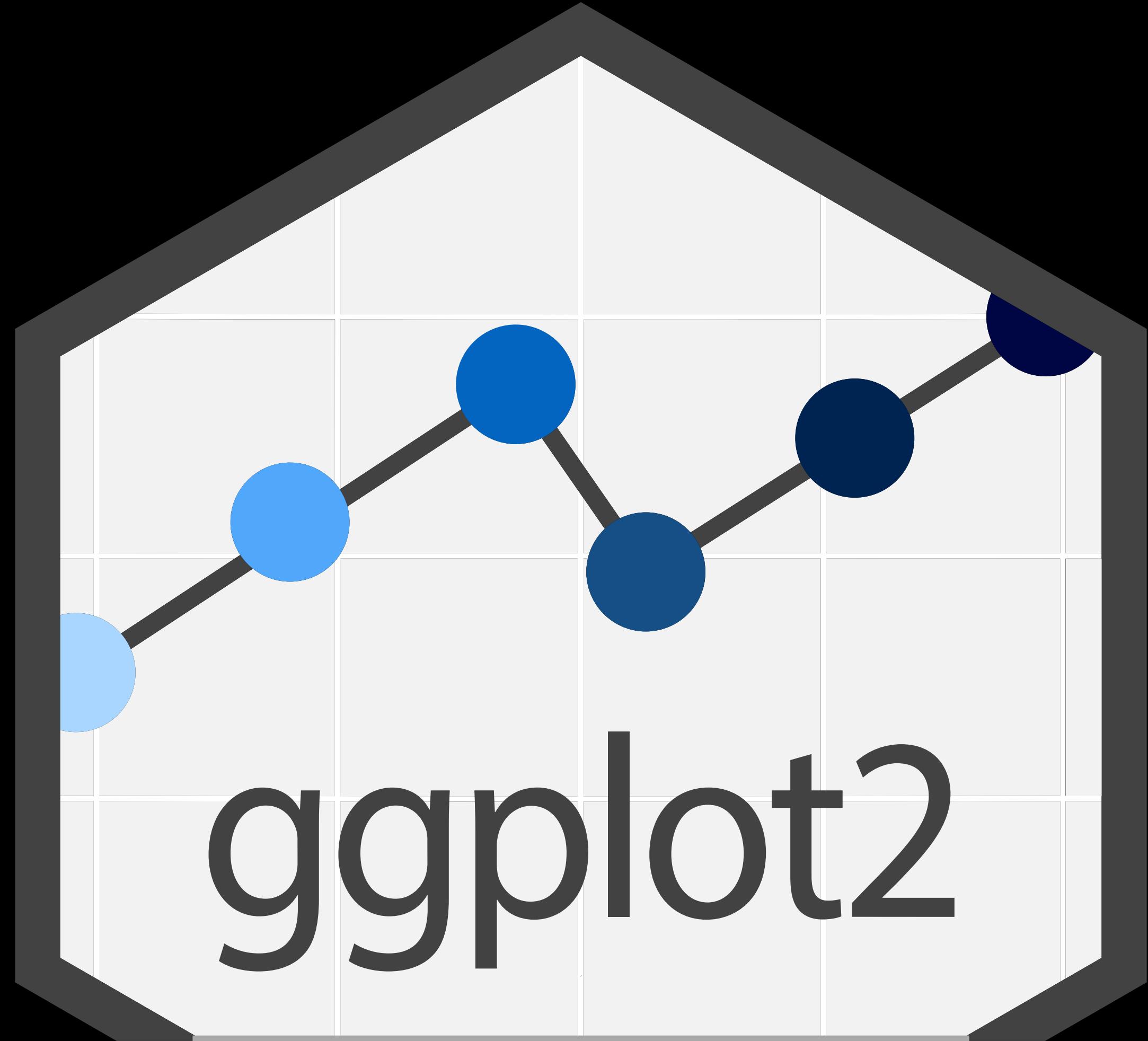
### Class 05

Barry Grant  
UC San Diego

<http://thegrantlab.org/bimm143>

How do we make informative  
and compelling figures?



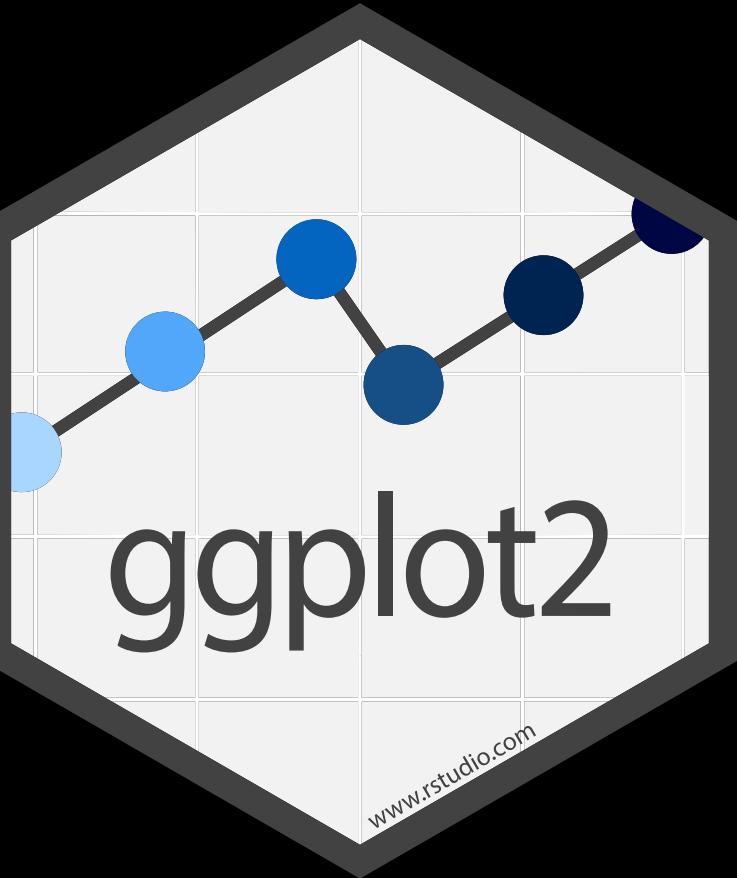


Currently the premier plotting  
library on the planet!

**Key Insight:** All visualizations  
map data into quantifiable aesthetic  
features of the resulting graphic

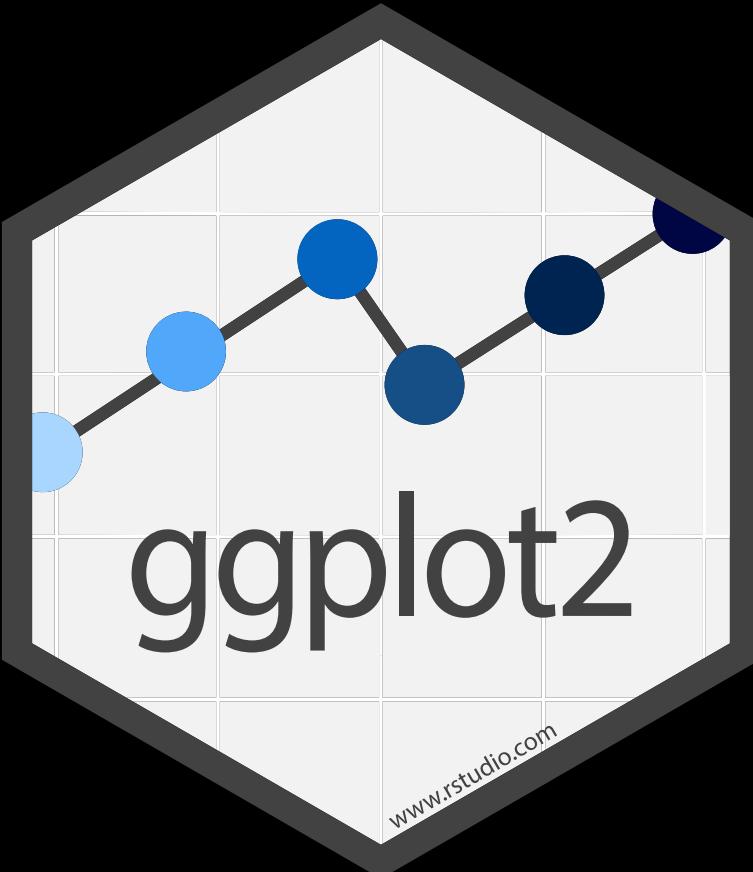
**Key Insight:** All visualizations  
map data into quantifiable aesthetics  
features of the resulting graphic

data → aesthetics



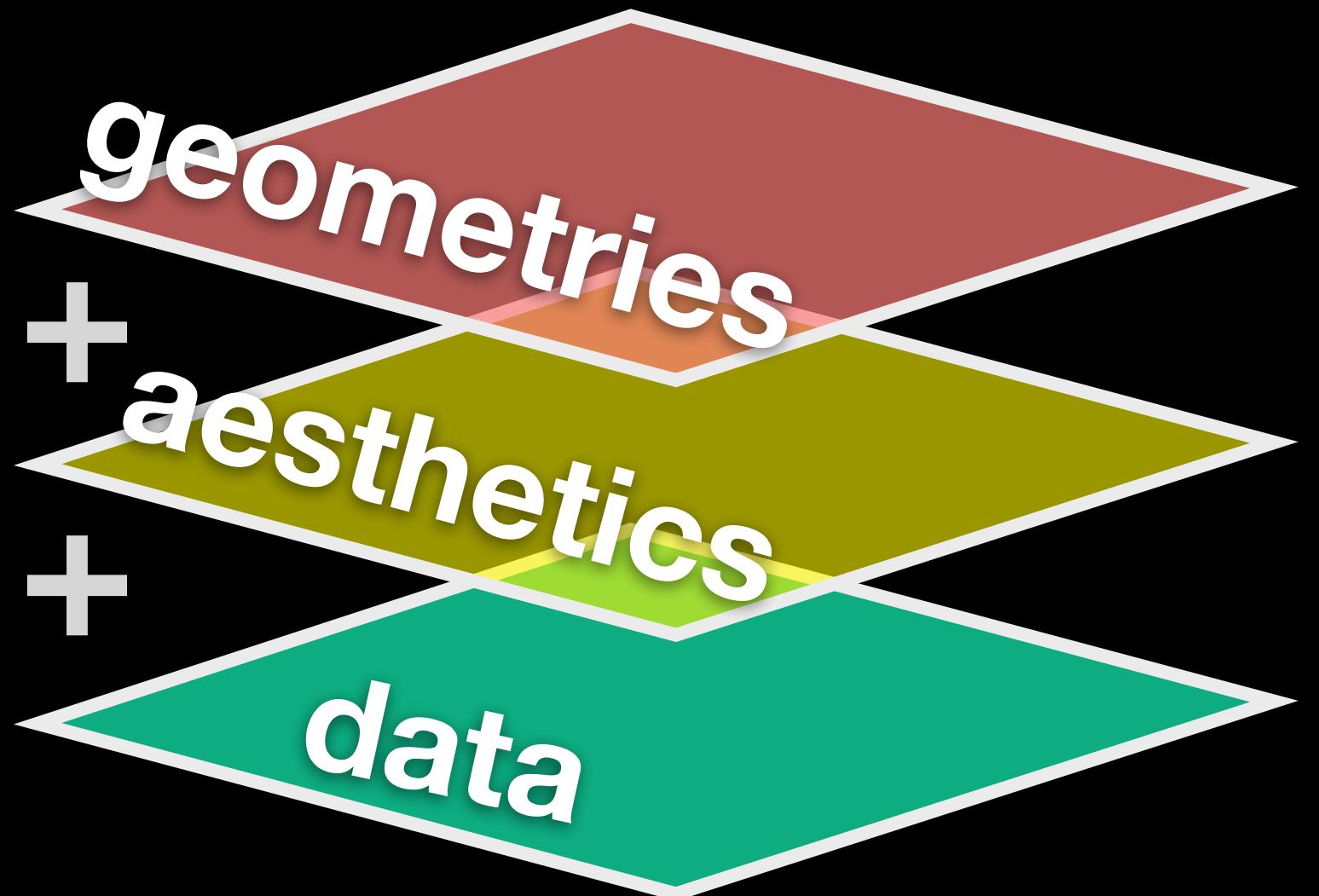
# data + aesthetics + geometrys

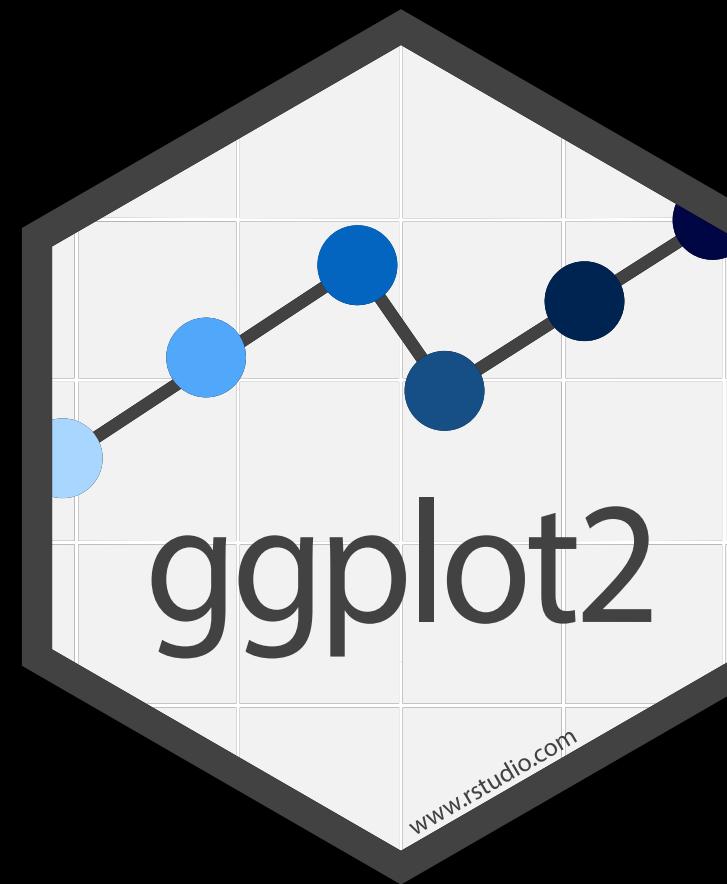
Three main "layers"  
that are in every ggplot



# data + aesthetics + geometrys

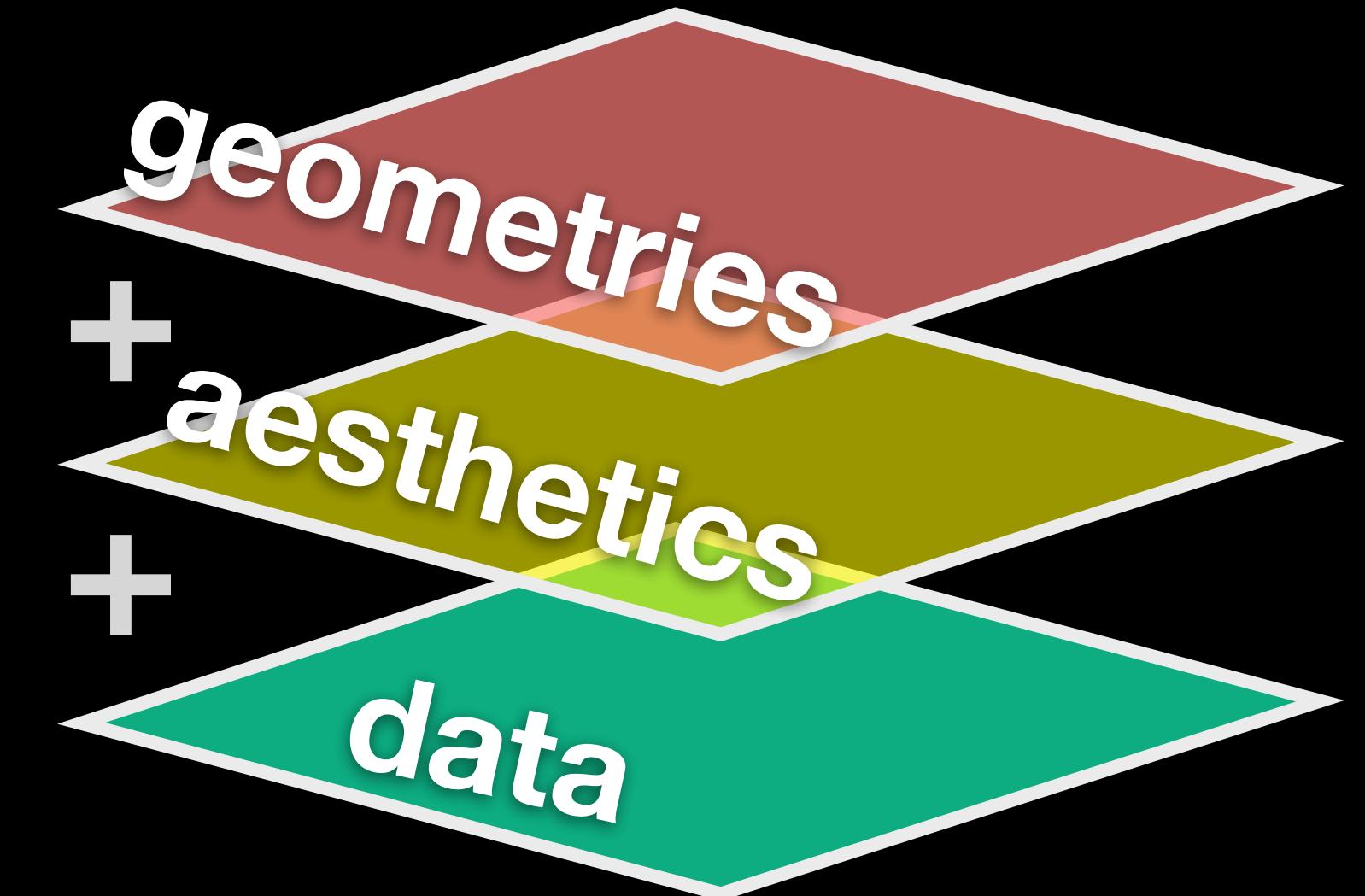
Three main "layers"  
that are in every ggplot

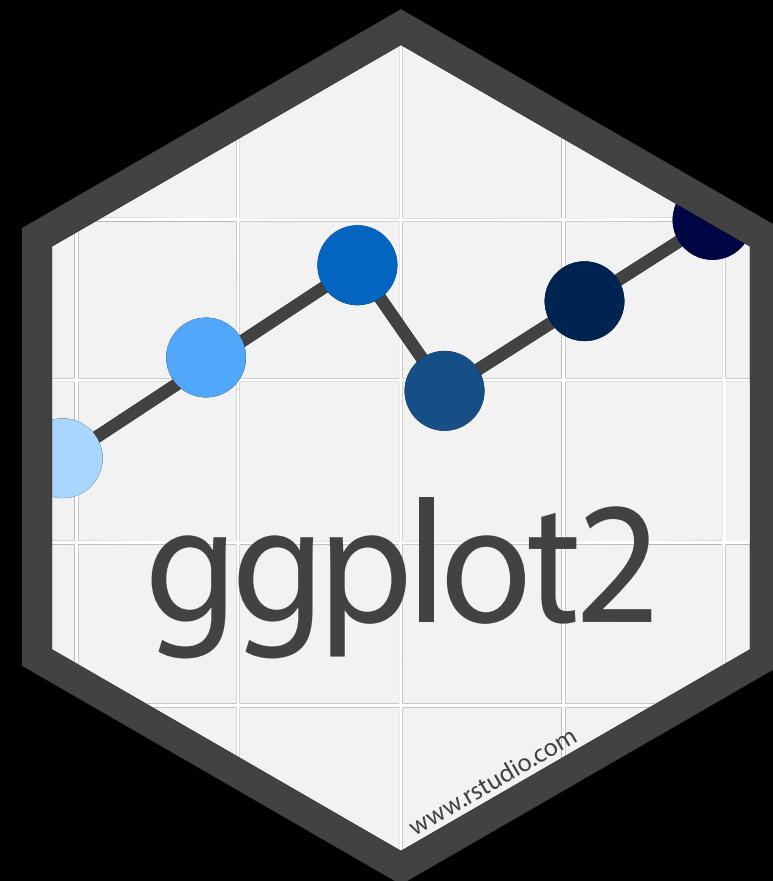




# data + aesthetics + geometrys

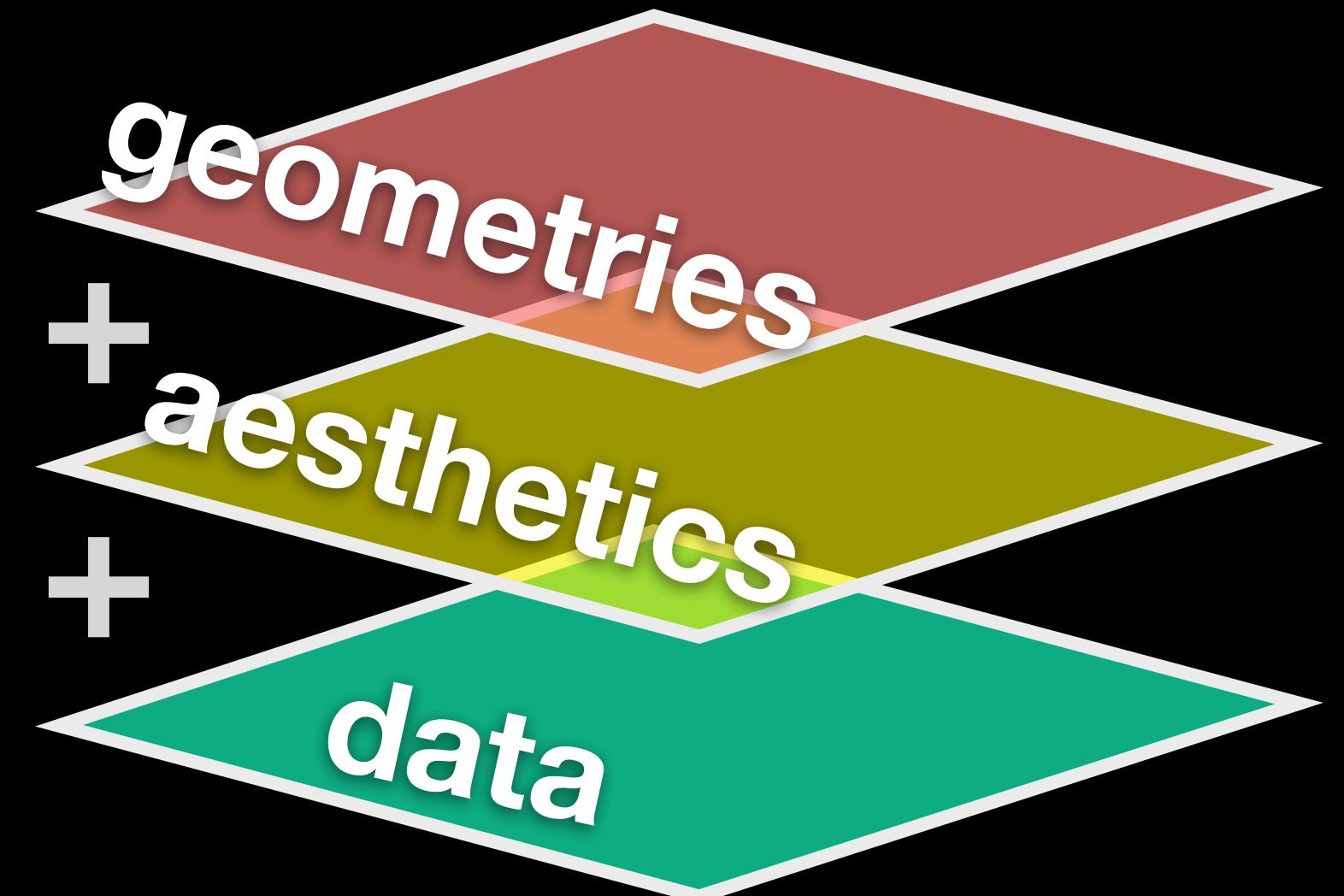
```
ggplot(data=mpg) +  
  aes(x=displ, y=hwy, color=class) +  
  geom_point()
```





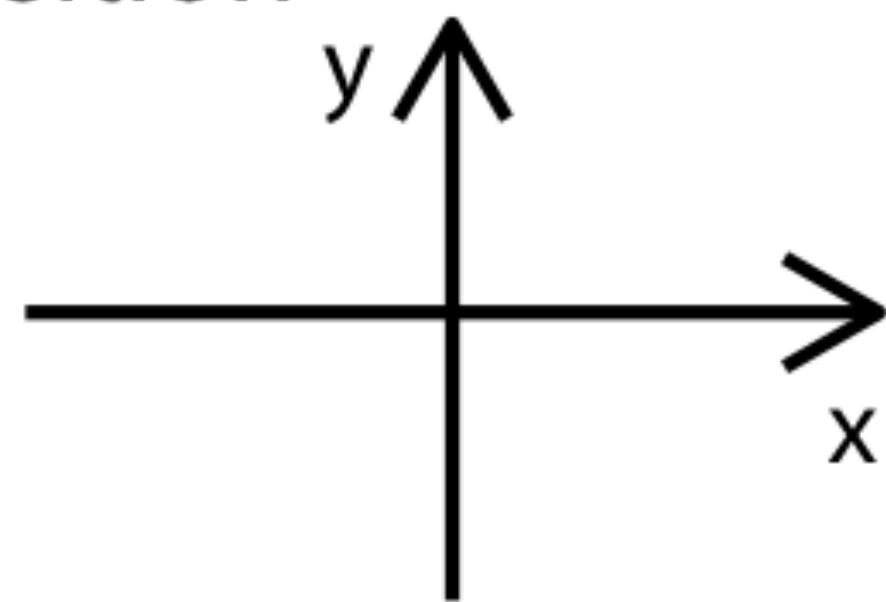
**data** + **aesthetics** + **geometries**

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



# Common aesthetics include

position



size



line type



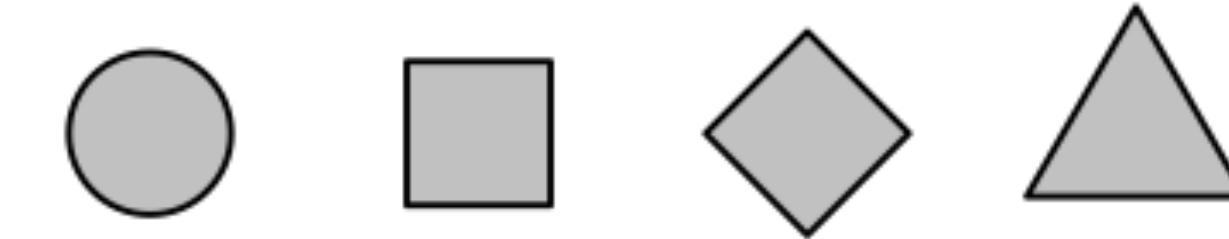
line width



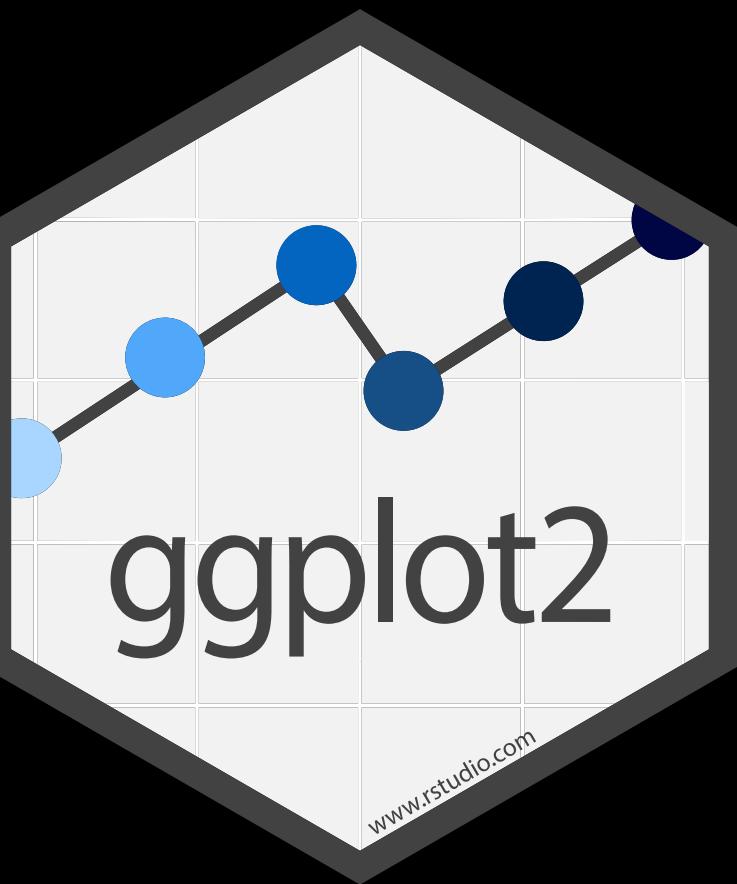
color



shape



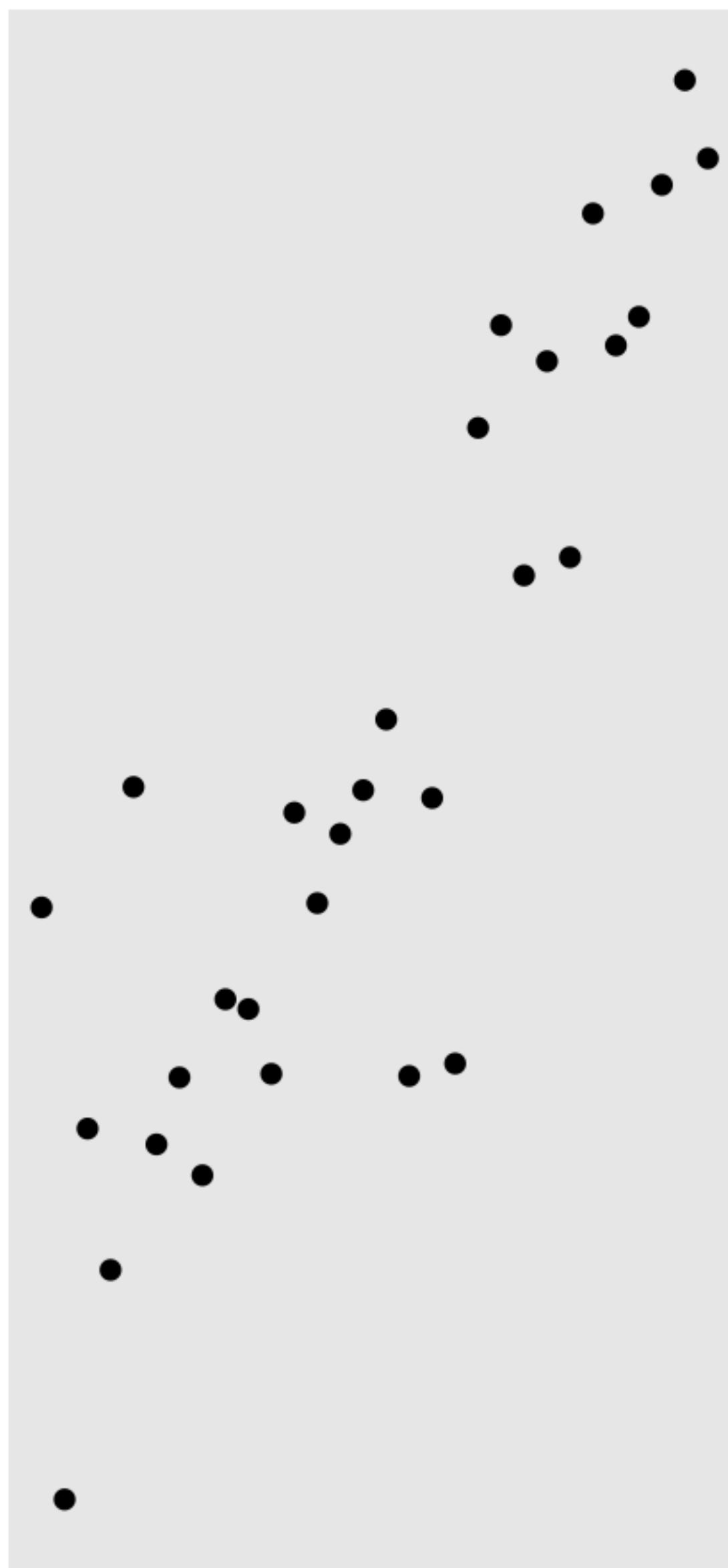
Modified from: Wilke (2019)



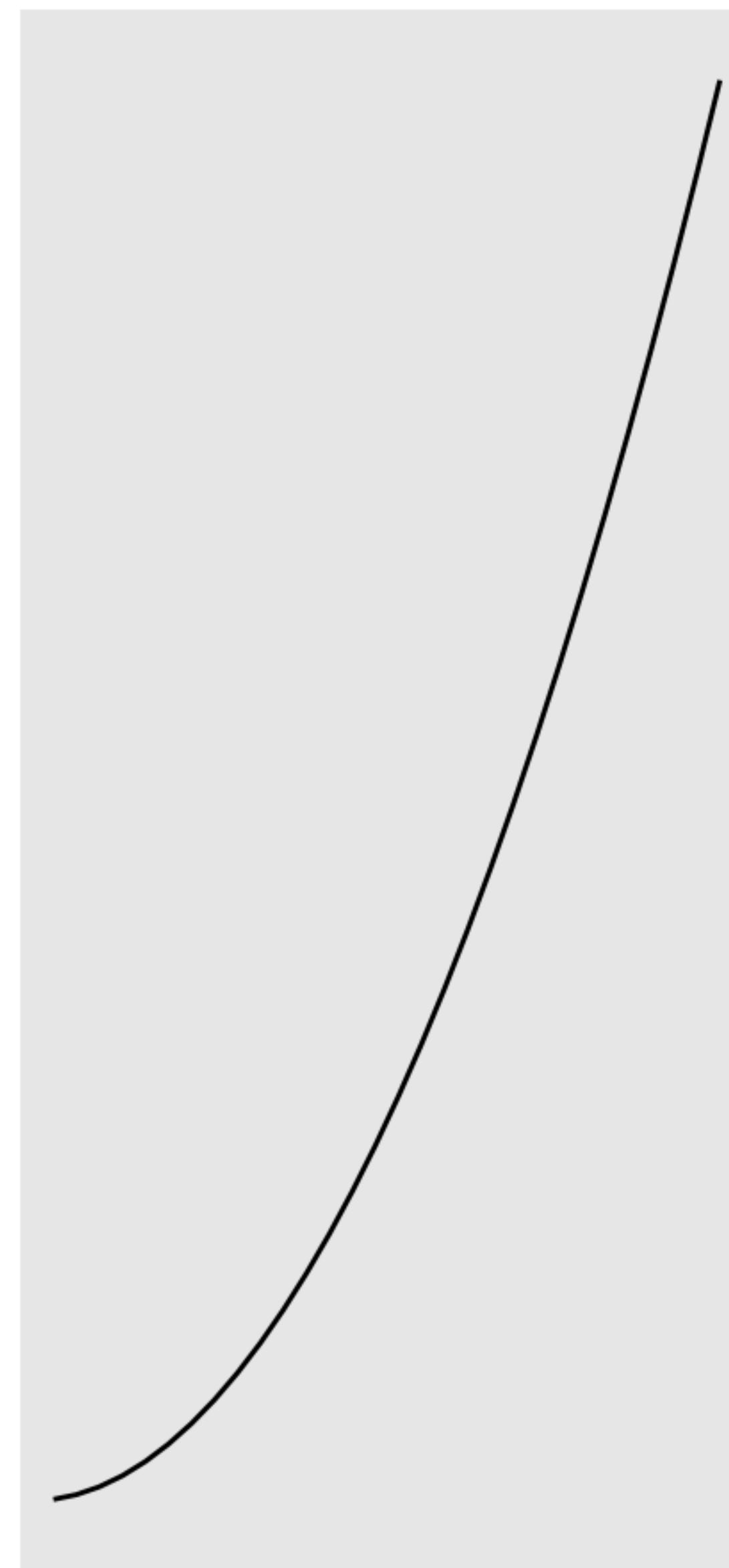
**data + aesthetics + geometrys**

Three main "layers"  
that are in every ggplot

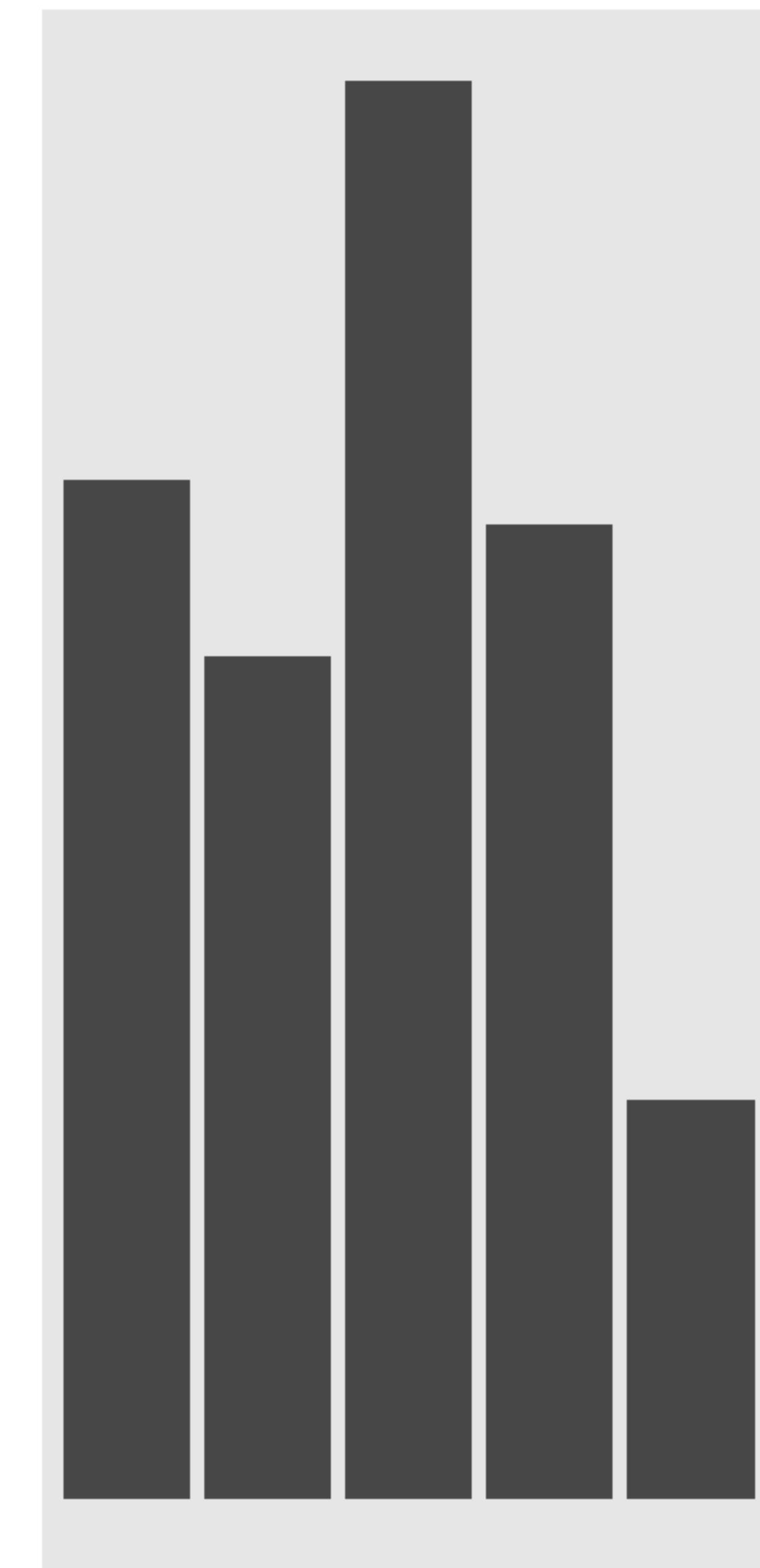
`geom_point()`



`geom_line()`



`geom_col()`



# Data Visualization with ggplot2 :: CHEAT SHEET

## Basics

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



To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.



Complete the template below to build a graph.

```
ggplot(data = <DATA>) +  
  <GEOM_FUNCTION>(mapping = aes(< MAPPINGS >),  
  stat = <STAT>, position = <POSITION>) +  
  <COORDINATE_FUNCTION> +  
  <FACET_FUNCTION> +  
  <SCALE_FUNCTION> +  
  <THEME_FUNCTION>
```

`ggplot(data = mpg, aes(x = cty, y = hwy))` Begins a plot that you finish by adding layers to. Add one geom function per layer.

**aesthetic mappings**   **data**   **geom**

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

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

### GRAPHICAL PRIMITIVES

```
a <- ggplot(economics, aes(date, unemploy))  
b <- ggplot(seals, aes(x = long, y = lat))
```

**a + geom\_blank()**  
(Useful for expanding limits)

**b + geom\_curve(aes(yend = lat + 1,  
xend=long+1),curvature=1) - x, xend, y, yend,  
alpha, angle, color, curvature, linetype, size**

**a + geom\_path(lineend="butt", linejoin="round",  
linemetre=1)  
x, y, alpha, color, group, linetype, size**

**a + geom\_polygon(aes(group = group))  
x, y, alpha, color, fill, group, linetype, size**

**b + geom\_rect(aes(xmin = long, ymin=lat, xmax=  
long + 1, ymax = lat + 1)) - xmax, xmin, ymax,  
ymin, alpha, color, fill, linetype, size**

**a + geom\_ribbon(aes(ymax=unemploy - 900,  
ymax=unemploy + 900)) - y, ymax, ymin,  
alpha, color, fill, group, linetype, size**

### LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

**b + geom\_abline(aes(intercept=0, slope=1))  
b + geom\_hline(aes(yintercept = lat))  
b + geom\_vline(aes(xintercept = long))**

**b + geom\_segment(aes(yend=lat+1, xend=long+1))  
b + geom\_spoke(aes(angle = 1:1155, radius = 1))**

### ONE VARIABLE continuous

```
c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)
```

**c + geom\_area(stat = "bin")  
x, y, alpha, color, fill, linetype, size**

**c + geom\_density(kernel = "gaussian")  
x, y, alpha, color, fill, group, linetype, size, weight**

**c + geom\_dotplot()  
x, y, alpha, color, fill**

**c + geom\_freqpoly()  
x, y, alpha, color, group, linetype, size**

**c + geom\_histogram(binwidth = 5)  
x, y, alpha, color, fill, linetype, size, weight**

**c2 + geom\_qq(aes(sample = hwy))  
x, y, alpha, color, fill, linetype, size, weight**

### discrete

```
d <- ggplot(mpg, aes(fl))
```

**d + geom\_bar()  
x, alpha, color, fill, linetype, size, weight**

### TWO VARIABLES

#### continuous x , continuous y

**e <- ggplot(mpg, aes(cty, hwy))**

**e + geom\_label(aes(label = cty), nudge\_x = 1,  
nudge\_y = 1, check\_overlap = TRUE) x, y, label,  
alpha, angle, color, family, fontface, hjust,  
lineheight, size, vjust**

**e + geom\_jitter(height = 2, width = 2)  
x, y, alpha, color, fill, shape, size**

**e + geom\_point(), x, y, alpha, color, fill, shape,  
size, stroke**

**e + geom\_quantile(), x, y, alpha, color, group,  
linetype, size, weight**

**e + geom\_rug(sides = "bl") x, y, alpha, color,  
linetype, size**

**e + geom\_smooth(method = lm) x, y, alpha,  
color, fill, group, linetype, size, weight**

**e + geom\_text(aes(label = cty), nudge\_x = 1,  
nudge\_y = 1, check\_overlap = TRUE) x, y, label,  
alpha, angle, color, family, fontface, hjust,  
lineheight, size, vjust**

#### discrete x , continuous y

**f <- ggplot(mpg, aes(class, hwy))**

**f + geom\_col(), x, y, alpha, color, fill, group,  
linetype, size**

**f + geom\_boxplot(), x, y, lower, middle, upper,  
ymin, alpha, color, fill, group, linetype,  
shape, size, weight**

**f + geom\_dotplot(binaxis = "y", stackdir =  
"center") x, y, alpha, color, fill, group**

**f + geom\_violin(scale = "area") x, y, alpha, color,  
fill, group, linetype, size, weight**

#### discrete x , discrete y

**g <- ggplot(diamonds, aes(cut, color))**

**g + geom\_count(), x, y, alpha, color, fill, shape,  
size, stroke**

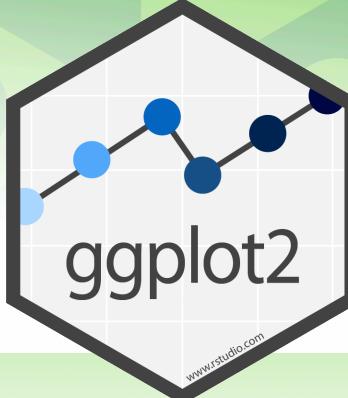
### THREE VARIABLES

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

**l + geom\_contour(aes(z = z))  
x, y, z, alpha, colour, group, linetype,  
size, weight**

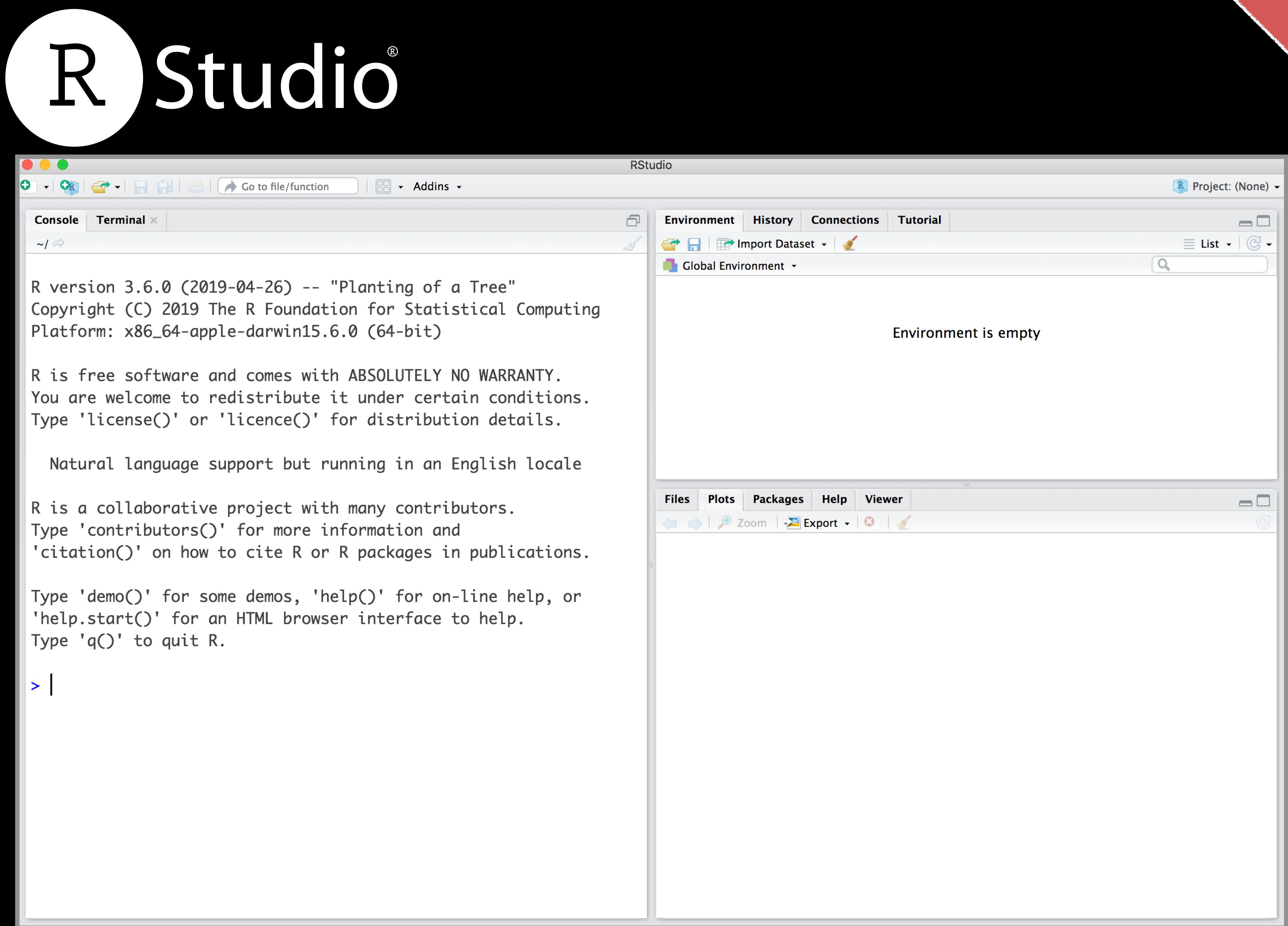
**l + geom\_raster(aes(fill = z), hjust = 0.5, vjust = 0.5,  
interpolate = FALSE) x, y, alpha, fill**

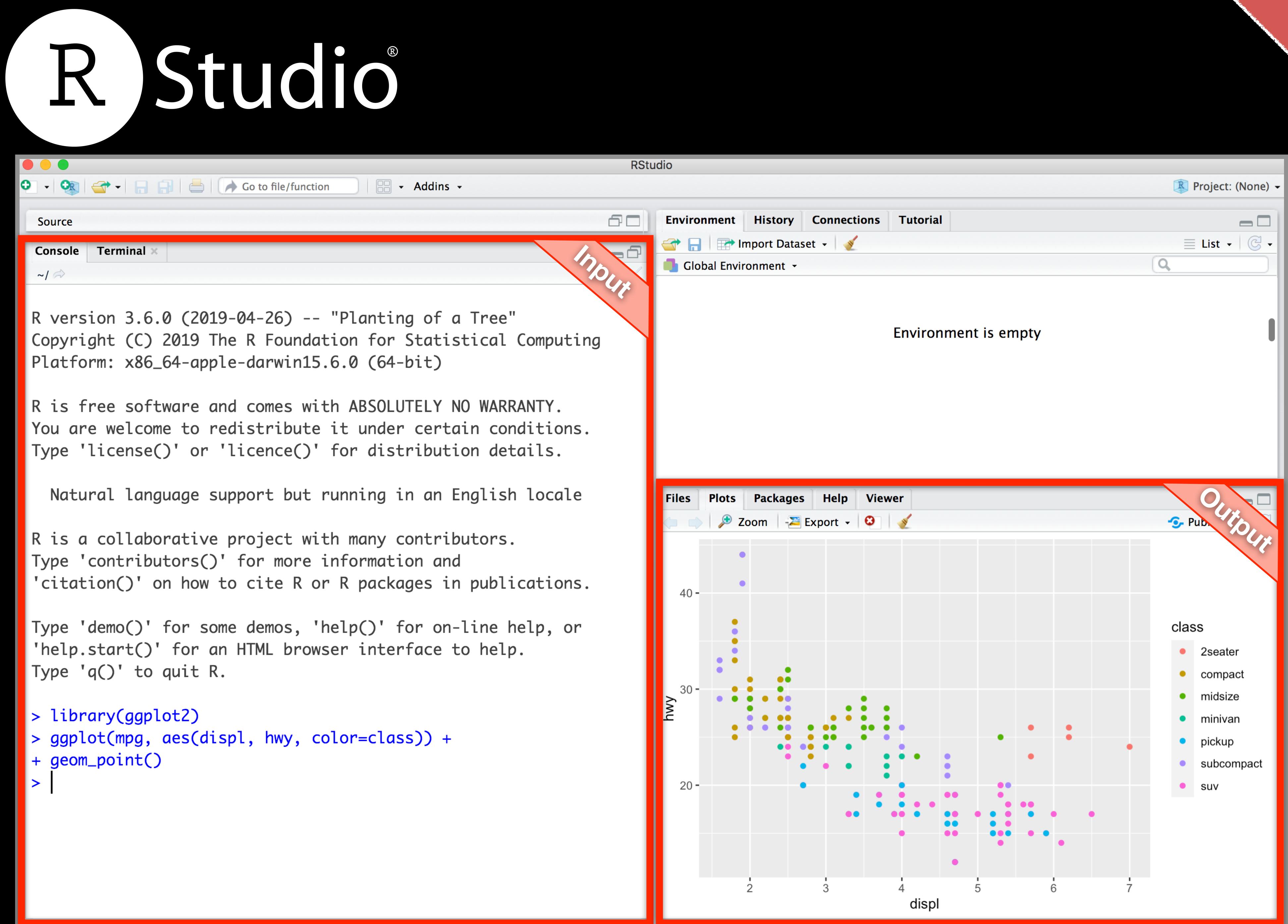
**l + geom\_tile(aes(fill = z)) x, y, alpha, color, fill,  
linetype, size, width**



Learn more about core  
geom\_FUNCTIONS()

There are > 40 core "geom" functions.  
See cheat-sheet link on class website!





Follow Along!

# RStudio >

Create a new **Project** and open a new **R Script**  
(N.B. make a **report** with notes and plots)

# In addition to your **PDF lab report** answer the **inbuilt questions**

1. Overview  
2. Background  
3. Getting Organized  
4. Common Plot Types  
**5. Creating Scatter Plots**  
Introduction to scatter plots  
Specifying a dataset with `ggplot()`  
Specifying aesthetic mappings with `aes()`  
Specifying a geom layer with `geom_point()`  
Adding more plot aesthetics through `aes()`  
6. OPTIONAL: Going Further  
7. OPTIONAL: Bar Charts  
About this document

Home Gmail Gcal GitHub BIMM143 BGNN213 GDrive Atmosphere CloudLaunch BIMM194 Blink News +

0/9

• Q. Which plot types are typically NOT used to compare distributions of numeric variables?

✓ Density plots  
**Network graphs**  
Histograms  
Violin plots  
Box plots

## 5. Creating Scatter Plots

In this section we will focus on:

- Defining a dataset for your plot using the main `ggplot()` function.
- Specifying how your data maps to plot aesthetics with the `aes()` function.
- Adding geometric layers using the `geom_point()` function.
- Combining the above function calls with `+` operator to make your plot

**Question Counter**

**Questions**

# Making a HTML Lab Report

- Save your **R script** (make sure it has some plots and comments)
- Can you **source** this **R script** file to re-generate all your plots without error?



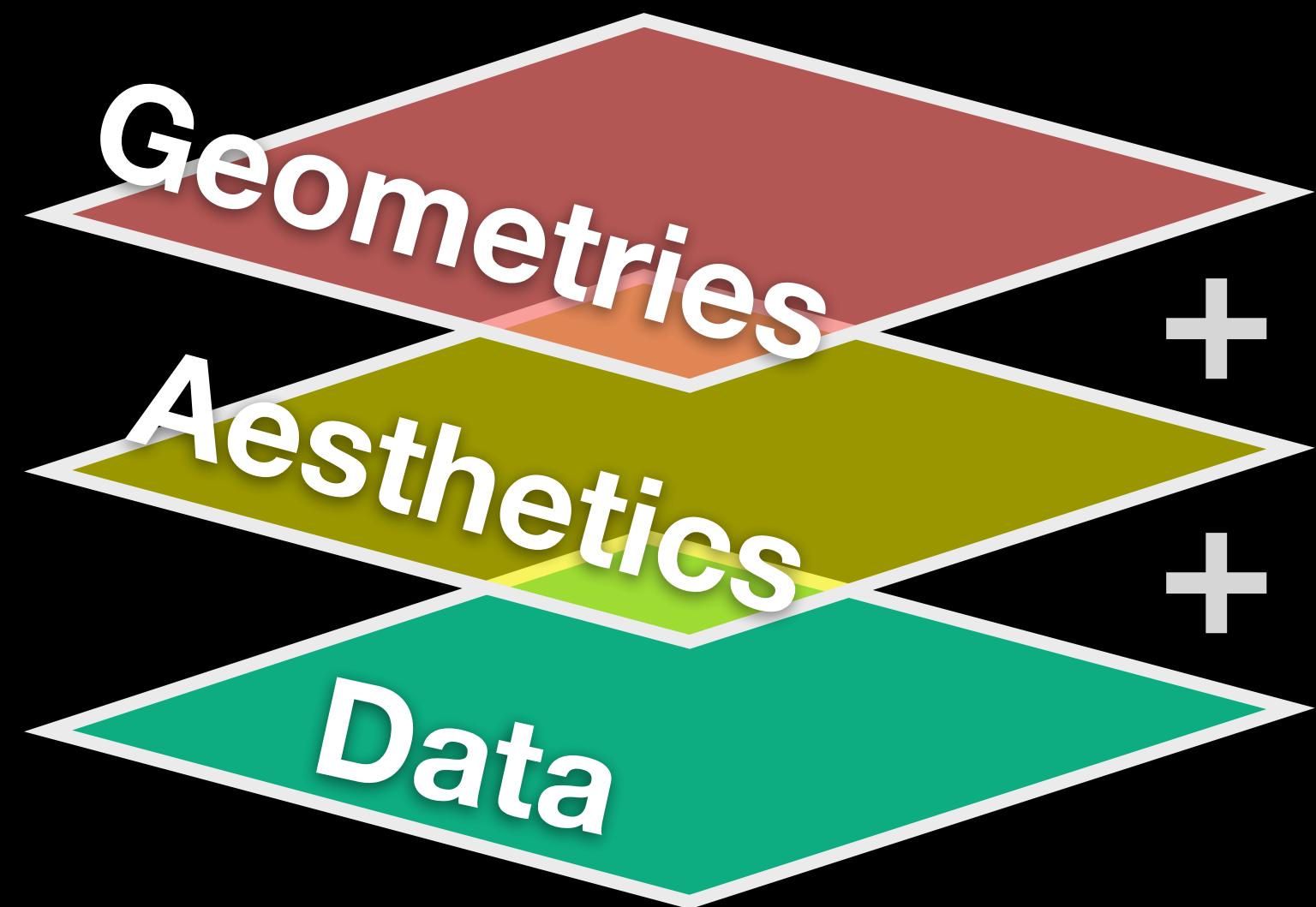
- If so you can now generate a nice **PDF report** of your work for upload to **GradeScope**...

[Optional Sections get you bonus points!]

# data + aesthetics + geometries

- **Summary:** ggplot takes an input *data.frame*, a mapping of columns to aesthetics and one or more geom *layers* (e.g. `geom_point()`, `geom_line()`, ...)

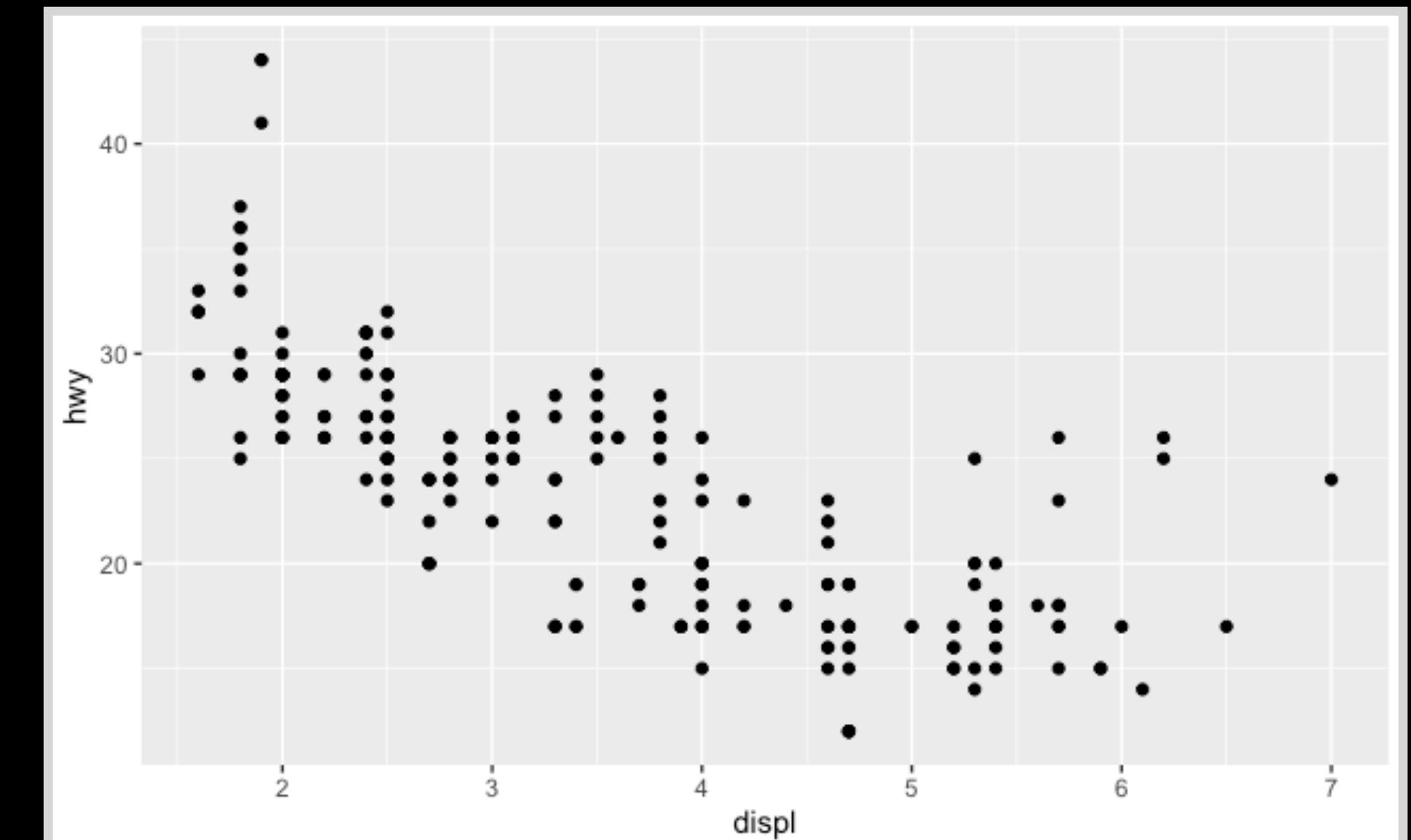
```
ggplot(data=mpg) +  
  aes(x=displ, y=hwy) +  
  geom_point()
```



# data + aesthetics + geometrys

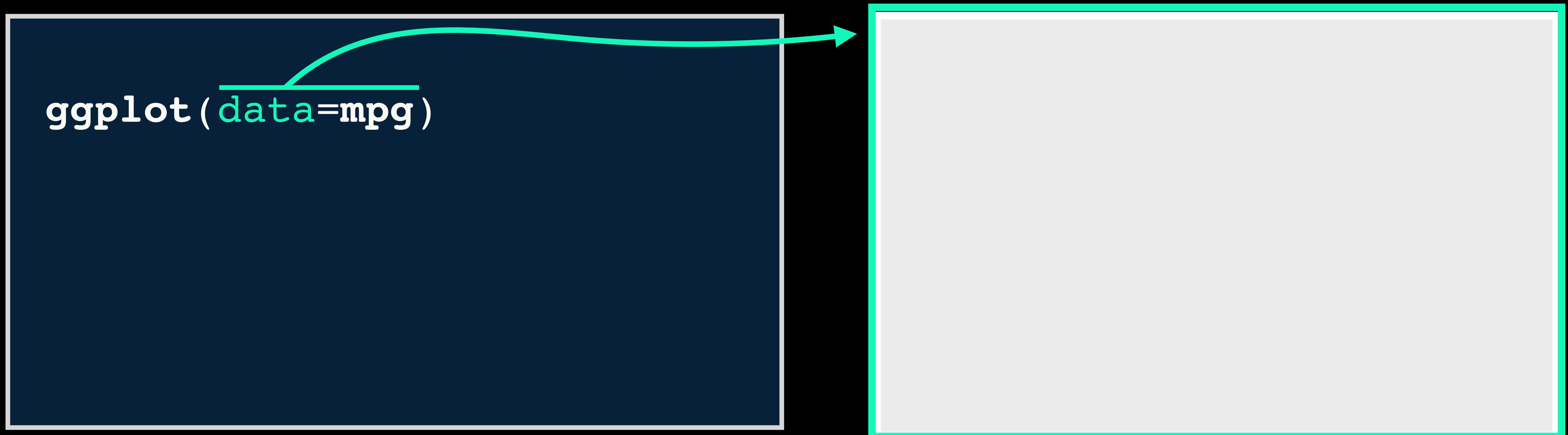
- **Summary:** ggplot takes an input *data.frame*, a mapping of columns to aesthetics and one or more geom *layers* (e.g. `geom_point()`, `geom_line()`, ...)

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



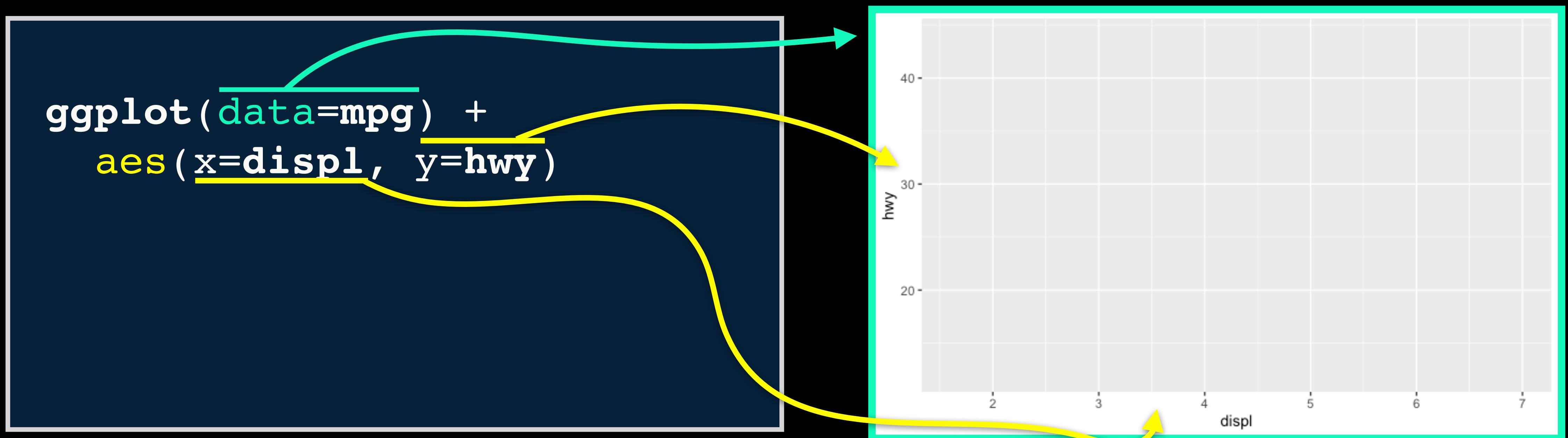
# data + aesthetics + geometrys

- **Summary:** ggplot takes an input *data.frame*, a mapping of columns to aesthetics and one or more geom *layers* (e.g. `geom_point()`, `geom_line()`, ...)



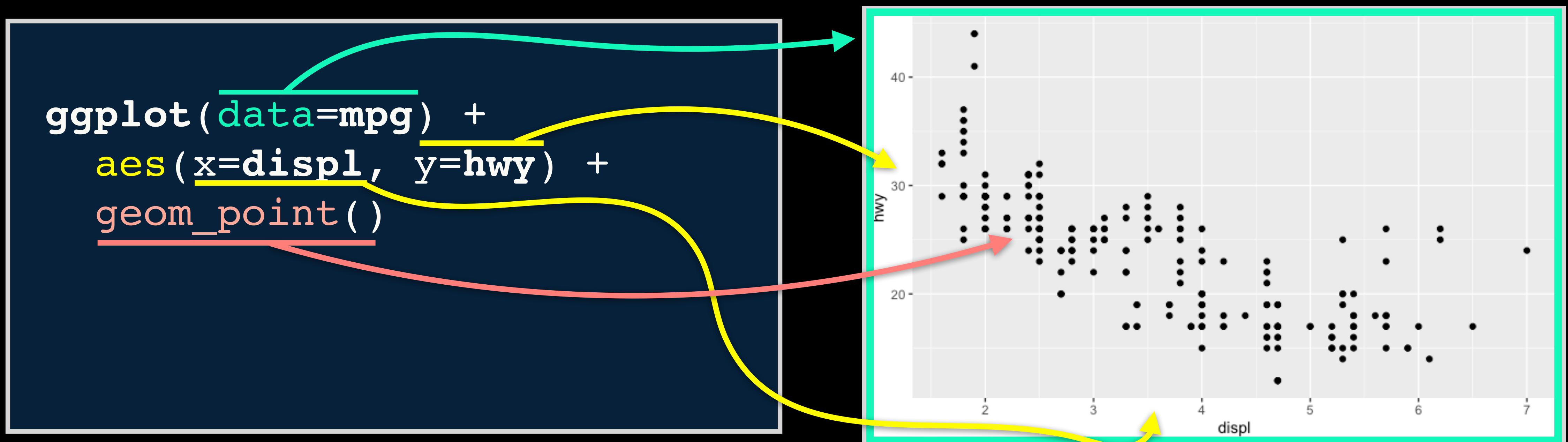
# data + aesthetics + geometrys

- **Summary:** ggplot takes an input *data.frame*, a mapping of columns to aesthetics and one or more geom *layers* (e.g. `geom_point()`, `geom_line()`, ...)



# data + aesthetics + geometrys

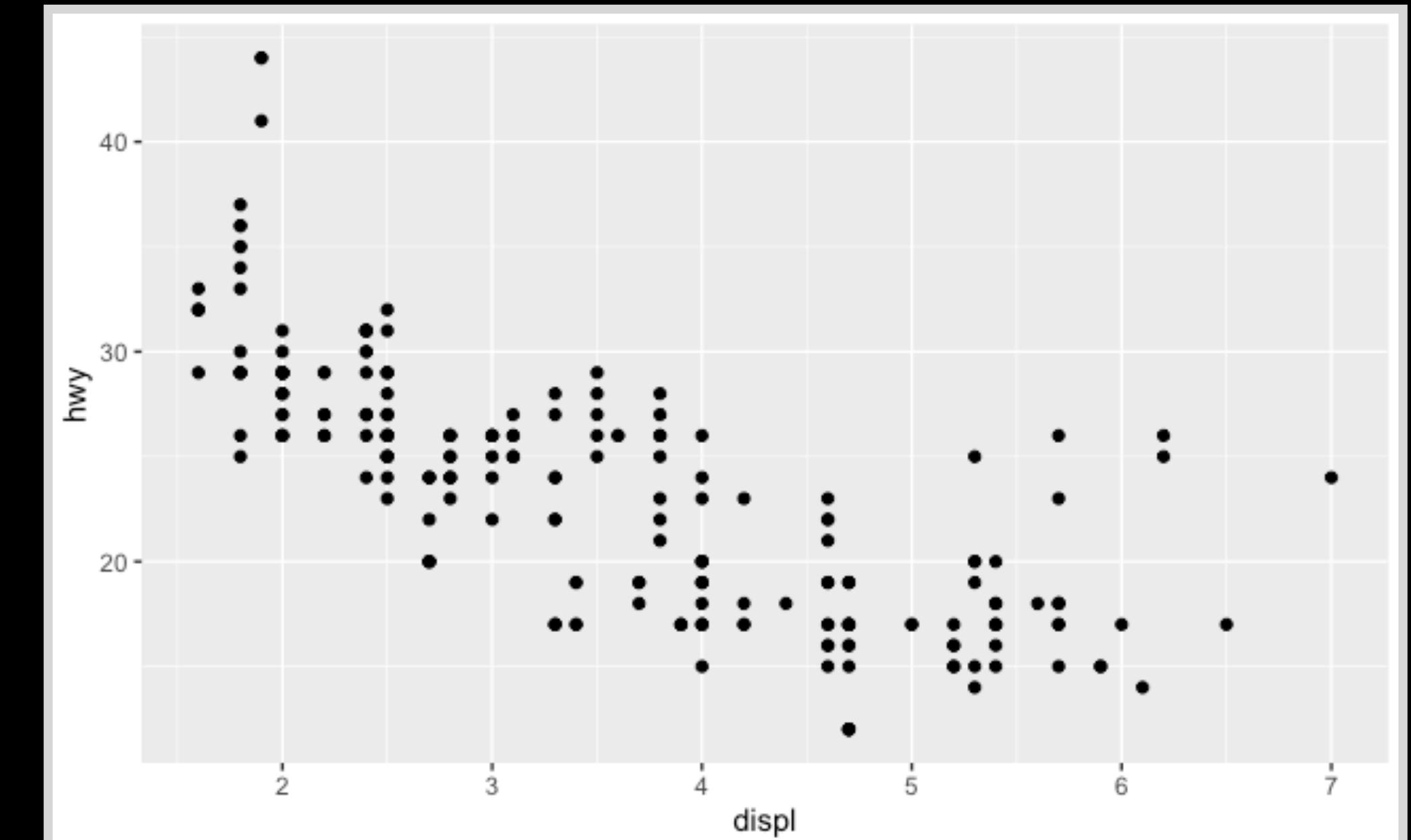
- **Summary:** ggplot takes an input *data.frame*, a mapping of columns to aesthetics and one or more geom *layers* (e.g. `geom_point()`, `geom_line()`, ...)



# data + aesthetics + geometrys

- We can keep building more complicated plots by adding more **layers**

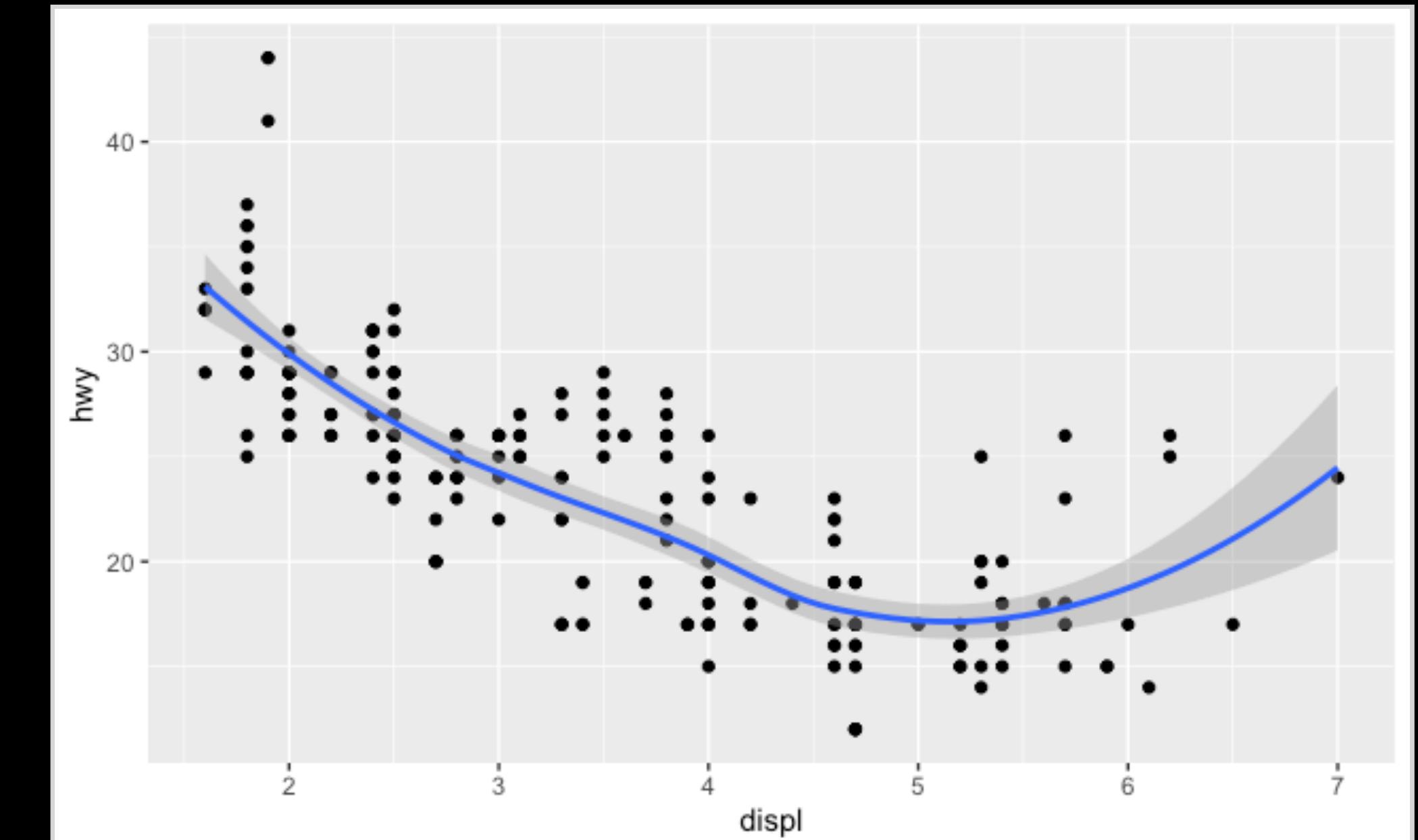
```
ggplot(data=mpg) +  
  aes(x=displ, y=hwy) +  
  geom_point()
```



# data + aesthetics + geometries

- We can keep building more complicated plots by adding more **layers**
- For example lets add another **geom**, in this case a smooth line fitted to the data...

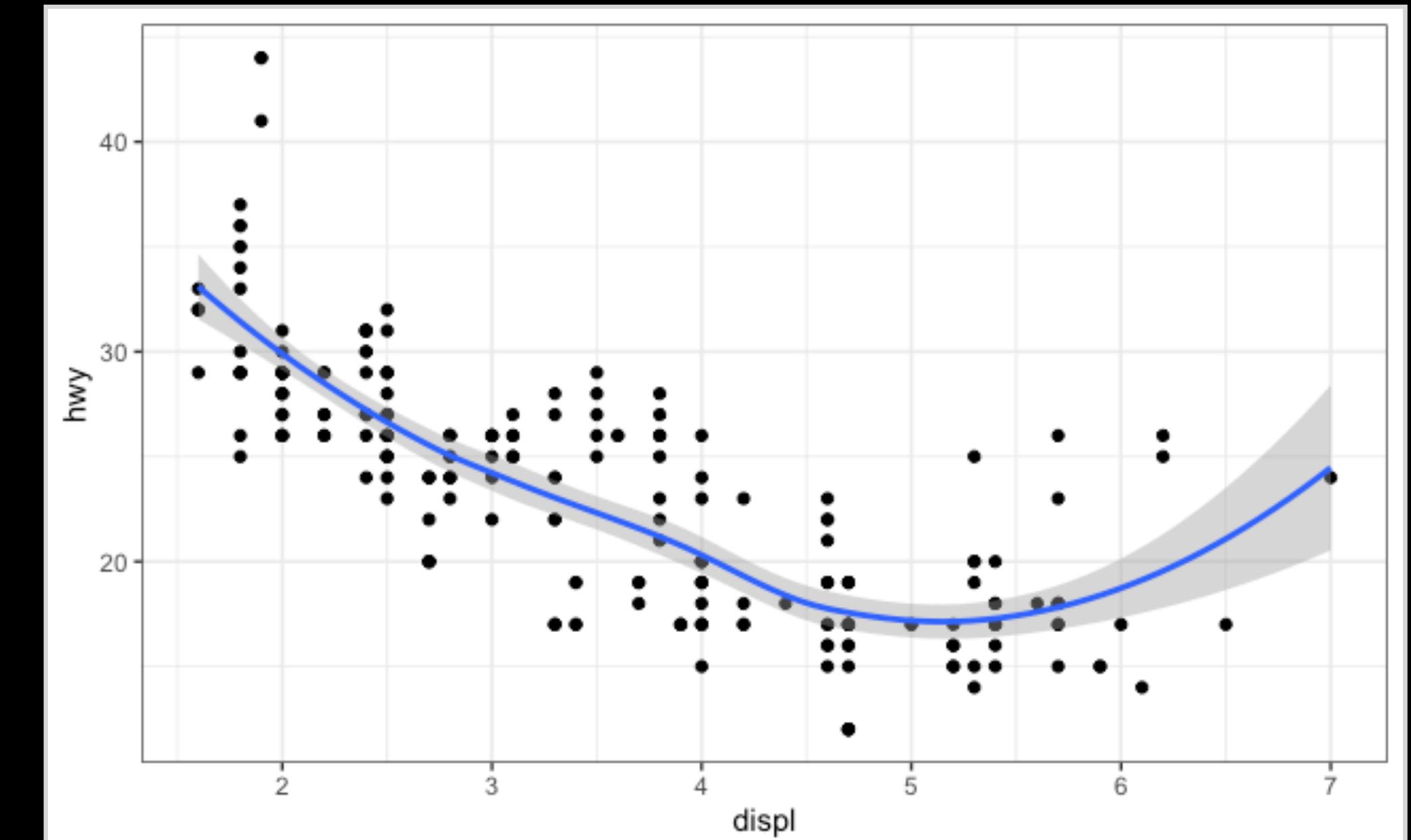
```
ggplot(data=mpg) +  
  aes(x=displ, y=hwy) +  
  geom_point() +  
  geom_smooth()
```



# data + aesthetics + geometries

- We can also add other customizations like themes...

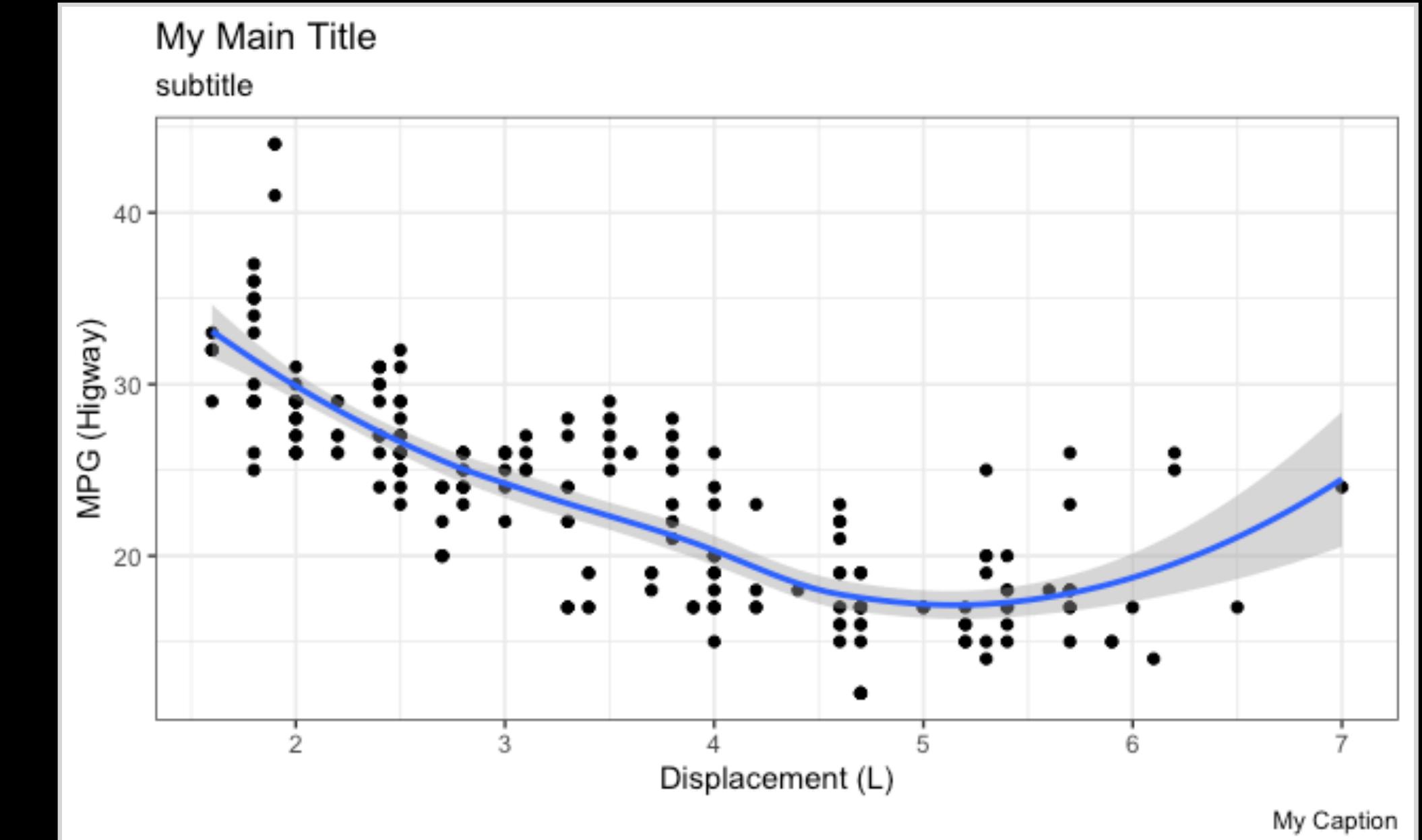
```
ggplot(data=mpg) +  
  aes(x=displ, y=hwy) +  
  geom_point() +  
  geom_smooth() +  
  theme_bw()
```



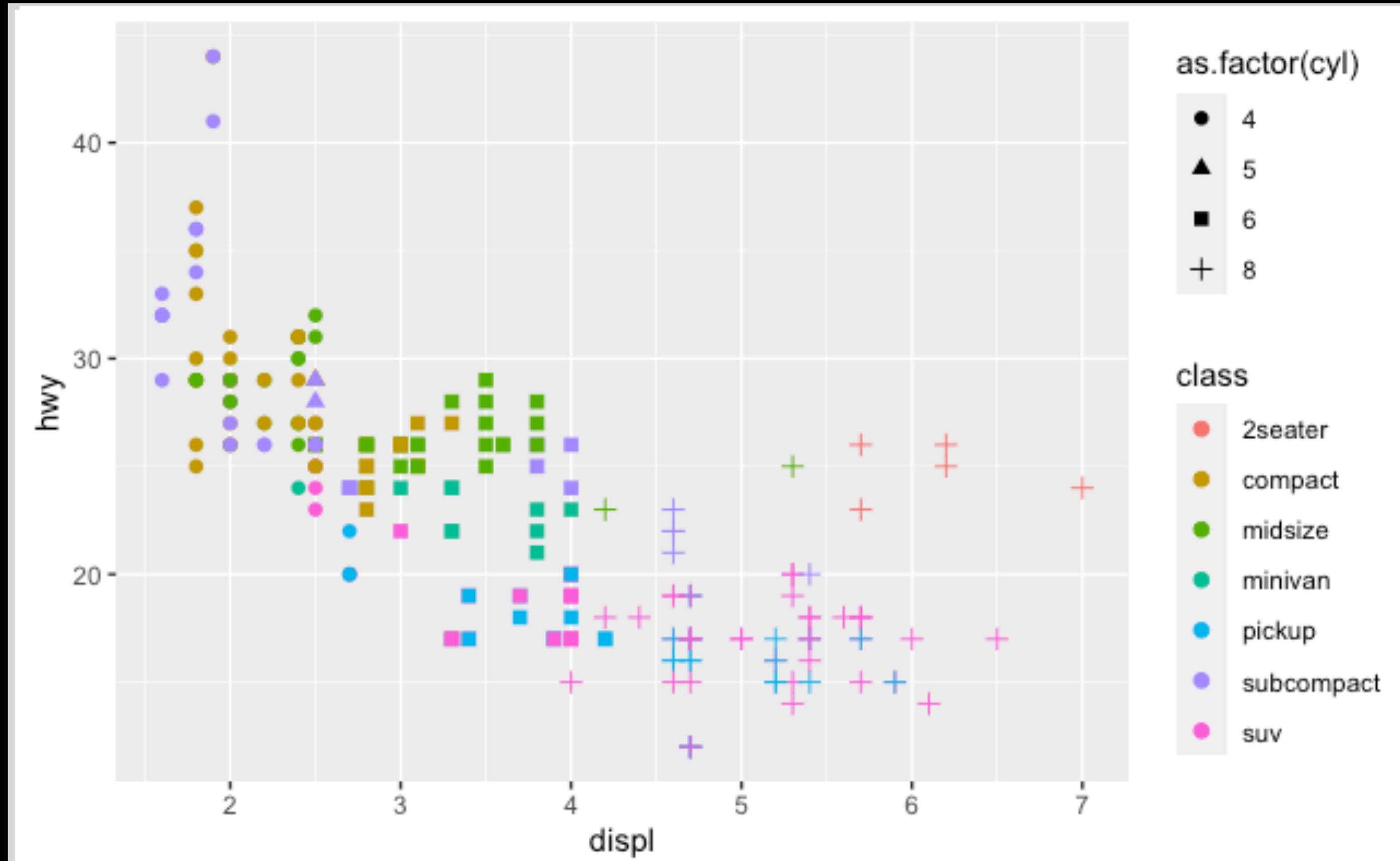
# data + aesthetics + geometrys

- And various custom annotation labels...

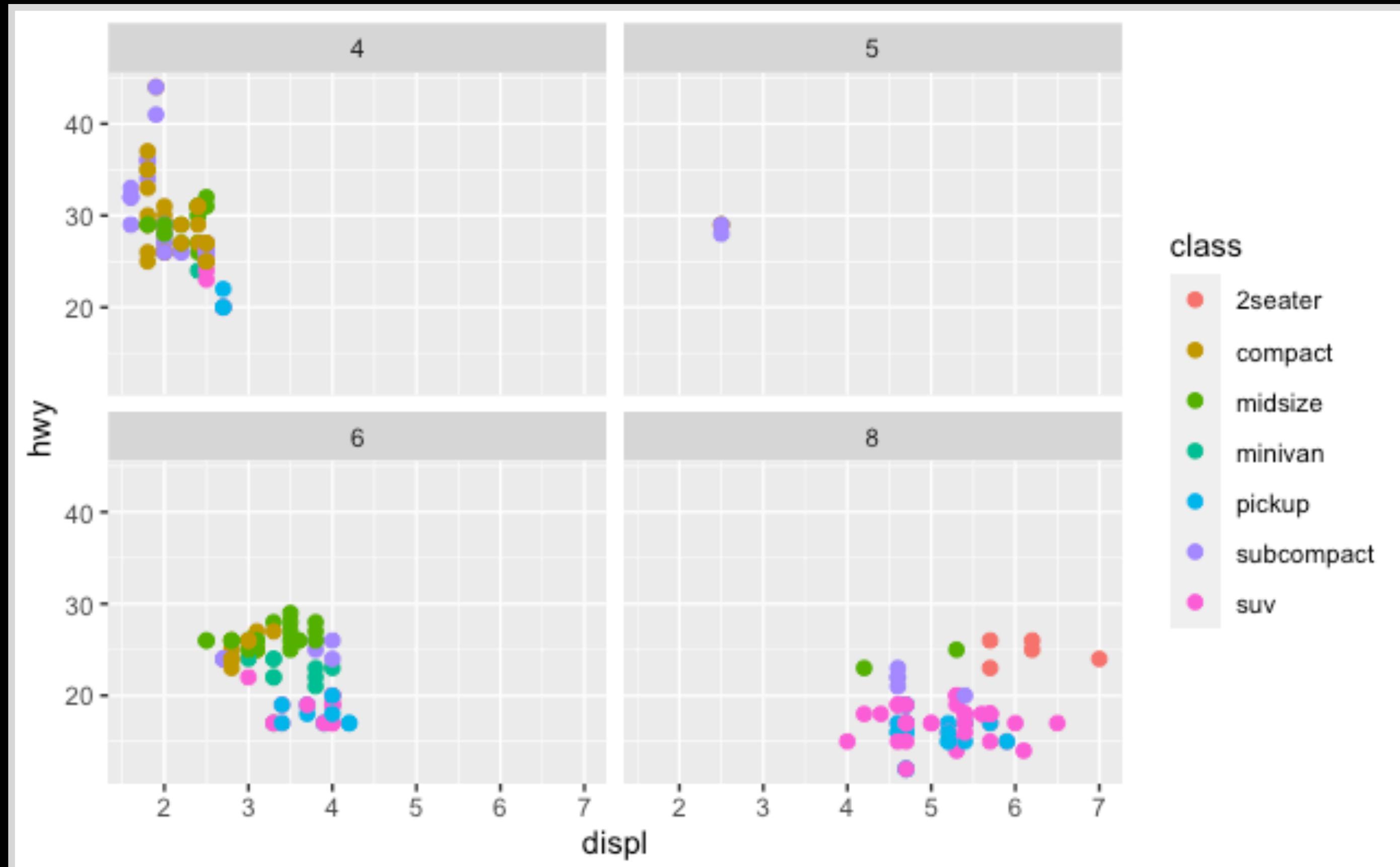
```
ggplot(data=mpg) +  
  aes(x=displ, y=hwy) +  
  geom_point() +  
  geom_smooth() +  
  theme_bw() +  
  labs(title="My Main Title",  
       subtitle = "subtitle",  
       caption = "My Caption",  
       x="Displacement (L)",  
       y= "MPG (Higway)")
```



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



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



# Data Visualization with ggplot2 :: CHEAT SHEET

## Basics

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



To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.



Complete the template below to build a graph.

```
ggplot(data = <DATA>) +  
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),  
    stat = <STAT>, position = <POSITION>) +  
  <COORDINATE_FUNCTION> +  
  <FACET_FUNCTION> +  
  <SCALE_FUNCTION> +  
  <THEME_FUNCTION>
```

ggplot(data = mpg, aes(x = cty, y = hwy)) Begins a plot that you finish by adding layers to. Add one geom function per layer.

aesthetic mappings    data    geom

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

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

### GRAPHICAL PRIMITIVES

```
a <- ggplot(economics, aes(date, unemploy))  
b <- ggplot(seals, aes(x = long, y = lat))
```

a + geom\_blank()  
(Useful for expanding limits)

b + geom\_curve(aes(yend = lat + 1,  
xend = long + 1), curvature = 1) - x, xend, y, yend,  
alpha, angle, color, curvature, linetype, size

a + geom\_path(lineend = "butt", linejoin = "round",  
linemitre = 1)  
x, y, alpha, color, group, linetype, size

a + geom\_polygon(aes(group = group))  
x, y, alpha, color, fill, group, linetype, size

b + geom\_rect(aes(xmin = long, ymin = lat, xmax =  
long + 1, ymax = lat + 1)) - xmax, xmin, ymax,  
ymin, alpha, color, fill, linetype, size

a + geom\_ribbon(aes(ymax = unemploy - 900,  
ymin = unemploy + 900)) - y, ymax, ymin,  
alpha, color, fill, group, linetype, size

### LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

b + geom\_abline(aes(intercept = 0, slope = 1))  
b + geom\_hline(aes(yintercept = lat))  
b + geom\_vline(aes(xintercept = long))

b + geom\_segment(aes(yend = lat + 1, xend = long + 1))  
b + geom\_spoke(aes(angle = 1:1155, radius = 1))

### ONE VARIABLE continuous

```
c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)
```

c + geom\_area(stat = "bin")  
x, y, alpha, color, fill, linetype, size

c + geom\_density(kernel = "gaussian")  
x, y, alpha, color, fill, group, linetype, size, weight

c + geom\_dotplot()  
x, y, alpha, color, fill

c + geom\_freqpoly()  
x, y, alpha, color, group, linetype, size

c + geom\_histogram(binwidth = 5)  
x, y, alpha, color, fill, linetype, size, weight

c2 + geom\_qq(aes(sample = hwy))  
x, y, alpha, color, fill, linetype, size, weight

### discrete

```
d <- ggplot(mpg, aes(fl))
```

d + geom\_bar()  
x, alpha, color, fill, linetype, size, weight

### TWO VARIABLES

continuous x , continuous y

```
e <- ggplot(mpg, aes(cty, hwy))
```

e + geom\_label(aes(label = cty), nudge\_x = 1,  
nudge\_y = 1, check\_overlap = TRUE) x, y, label,  
alpha, angle, color, family, fontface, hjust,  
lineheight, size, vjust

e + geom\_jitter(height = 2, width = 2)  
x, y, alpha, color, fill, shape, size

e + geom\_point(), x, y, alpha, color, fill, shape,  
size, stroke

e + geom\_quantile(), x, y, alpha, color, group,  
linetype, size, weight

e + geom\_rug(sides = "bl") x, y, alpha, color,  
linetype, size

e + geom\_smooth(method = lm) x, y, alpha,  
color, fill, group, linetype, size, weight

e + geom\_text(aes(label = cty), nudge\_x = 1,  
nudge\_y = 1, check\_overlap = TRUE) x, y, label,  
alpha, angle, color, family, fontface, hjust,  
lineheight, size, vjust

### discrete x , continuous y

```
f <- ggplot(mpg, aes(class, hwy))
```

f + geom\_col()  
x, y, alpha, color, fill, group, linetype, size

f + geom\_boxplot()  
x, y, lower, middle, upper,  
ymin, alpha, color, fill, group, linetype,  
shape, size, weight

f + geom\_dotplot(binaxis = "y", stackdir =  
"center") x, y, alpha, color, fill, group

f + geom\_violin(scale = "area") x, y, alpha, color,  
fill, group, linetype, size, weight

### discrete x , discrete y

```
g <- ggplot(diamonds, aes(cut, color))
```

g + geom\_count()  
x, y, alpha, color, fill, shape,  
size, stroke

### THREE VARIABLES

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

l + geom\_contour(aes(z = z))  
x, y, z, alpha, colour, group, linetype,  
size, weight

### continuous bivariate distribution

```
h <- ggplot(diamonds, aes(carat, price))
```

h + geom\_bin2d(binwidth = c(0.25, 500))  
x, y, alpha, color, fill, linetype, size, weight

h + geom\_density2d()  
x, y, alpha, colour, group, linetype, size

h + geom\_hex()  
x, y, alpha, colour, fill, size

### continuous function

```
i <- ggplot(economics, aes(date, unemploy))
```

i + geom\_area()  
x, y, alpha, color, fill, linetype, size

i + geom\_line()  
x, y, alpha, color, group, linetype, size

i + geom\_step(direction = "hv")  
x, y, alpha, color, group, linetype, size

### visualizing error

```
df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
```

```
j <- ggplot(df, aes(grp, fit, ymin = fit - se, ymax = fit + se))
```

j + geom\_crossbar(fatten = 2)  
x, y, ymax, ymin, alpha, color, fill, group, linetype,  
size

j + geom\_errorbar()  
x, ymax, ymin, alpha, color, group, linetype, size  
(also geom\_errorbarh())

j + geom\_linerange()  
x, ymin, ymax, alpha, color, group, linetype, size

j + geom\_pointrange()  
x, y, ymin, ymax, alpha, color, fill, group, linetype,  
shape, size, weight

### maps

```
data <- data.frame(murder = USAreests$Murder,
```

```
state = tolower(rownames(USAreests)))
```

```
map <- map_data("state")
```

```
k <- ggplot(data, aes(fill = murder))
```

k + geom\_map(aes(map\_id = state), map = map)  
+ expand\_limits(x = map\$long, y = map\$lat),  
map\_id, alpha, color, fill, linetype, size

Learn more about core  
geom\_FUNCTIONS()

DataCamp course!



# BIMM 143

## Hands-on Lab Session

### Class 05

Barry Grant  
UC San Diego

<http://thegrantlab.org/bimm143>