



BIMM 143

Hands-on Lab Session

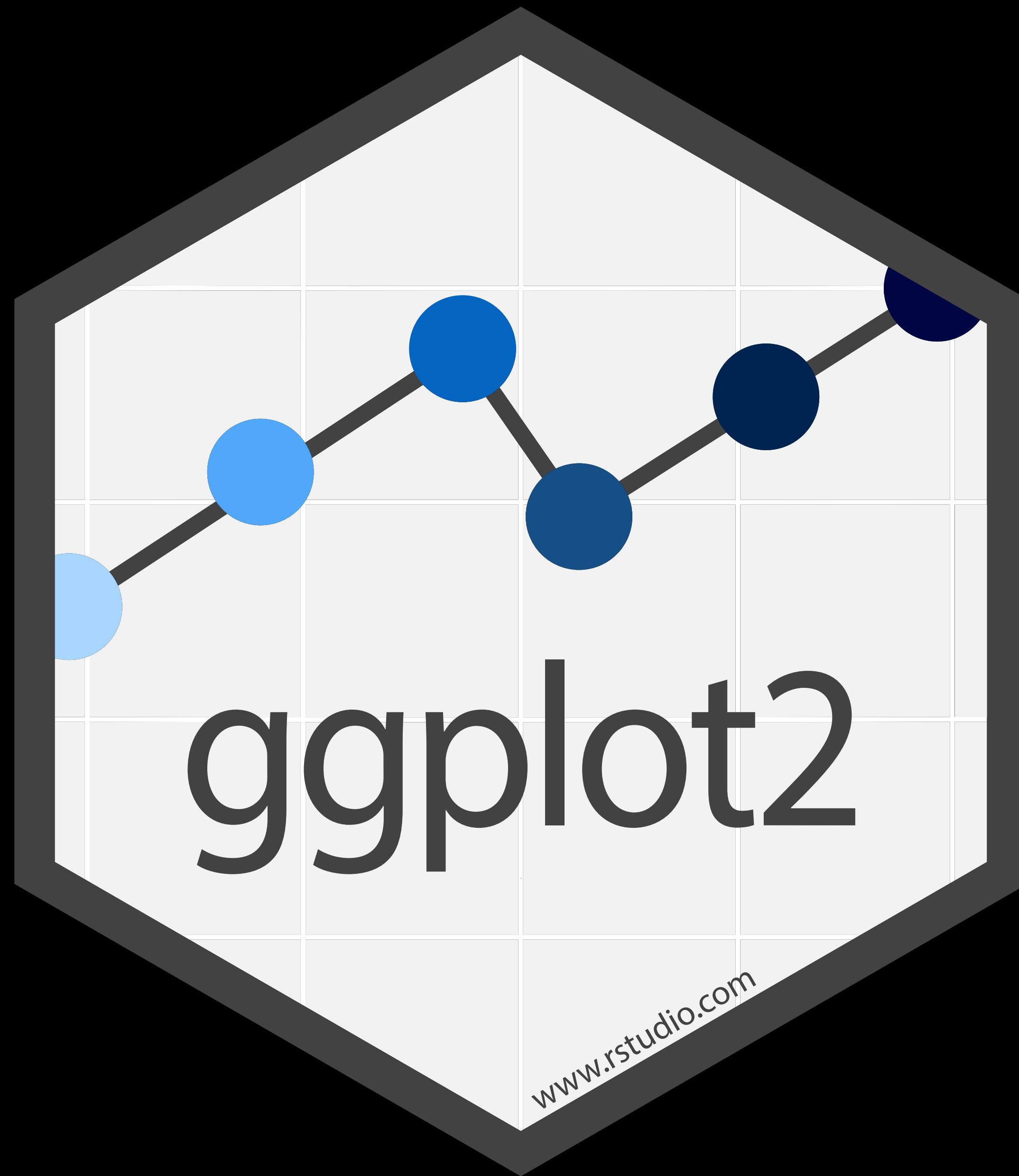
Class 05

Barry Grant

UC San Diego

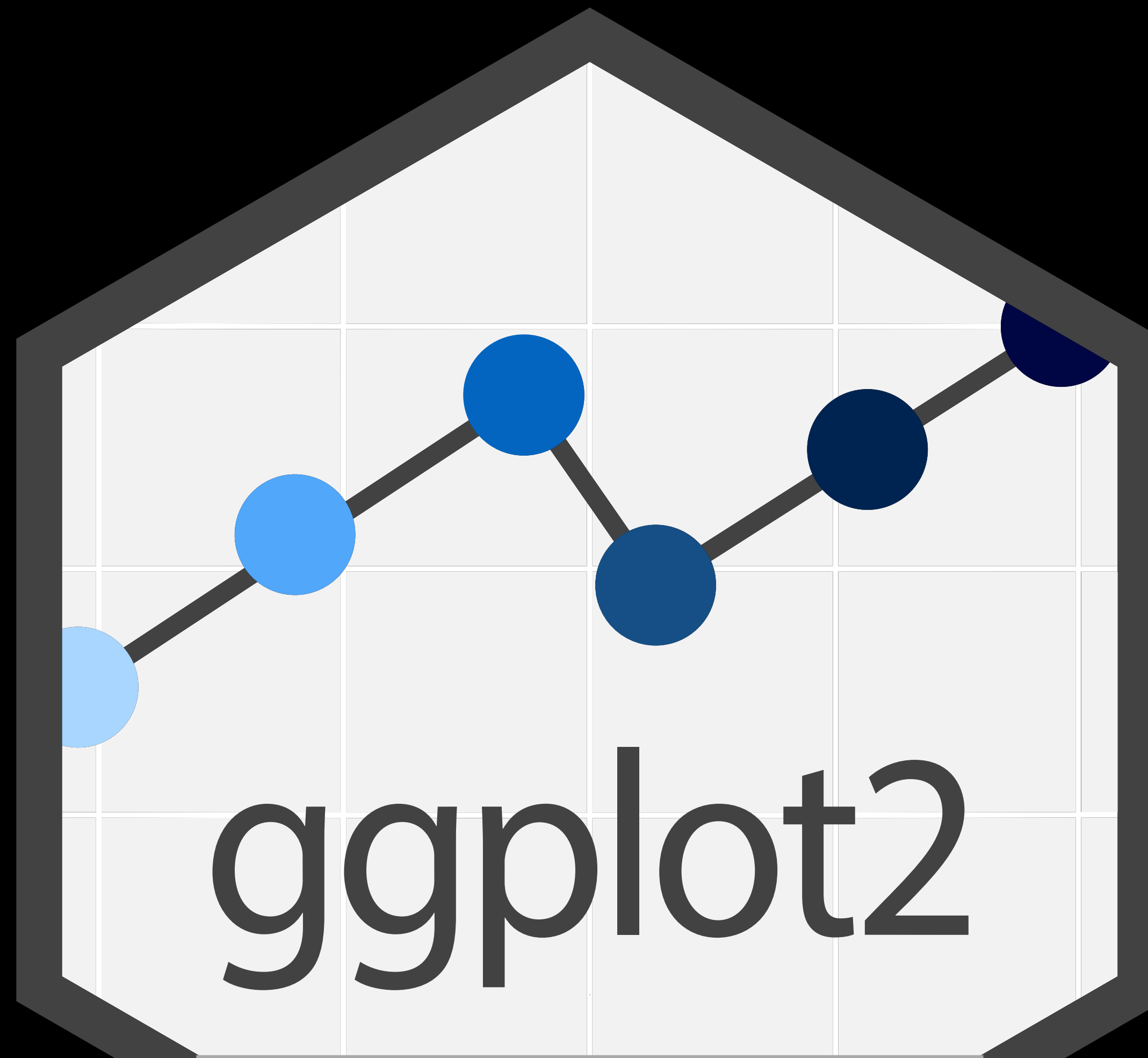
<http://thegrantlab.org/bimm143>

**How do we make informative
and compelling figures?**



ggplot2

www.rstudio.com

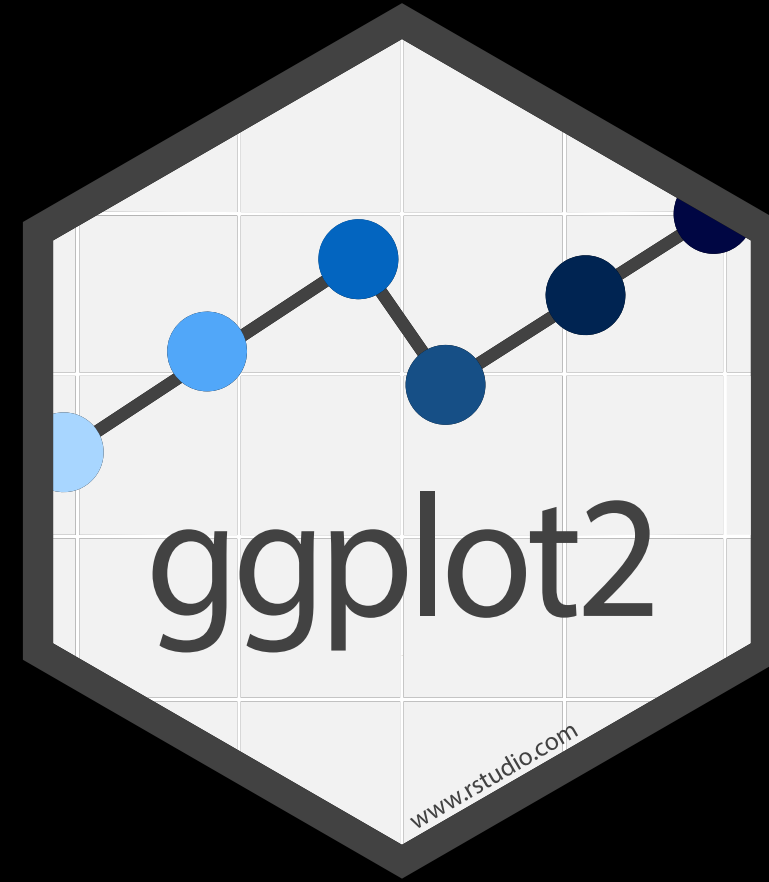


Currently the premier plotting
library on the planet!

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

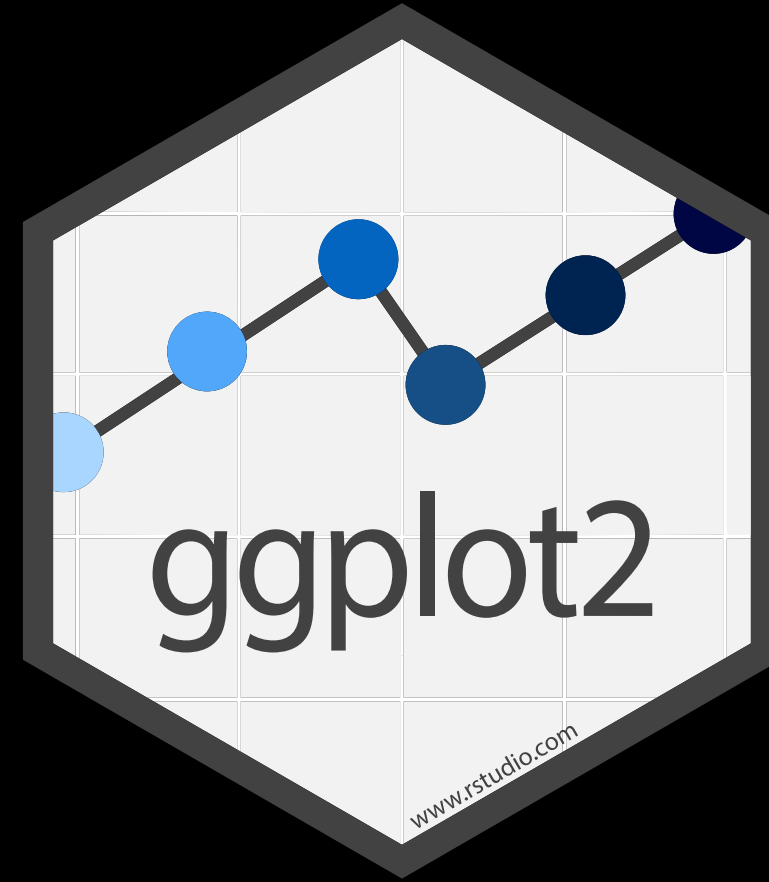
Key Insight: All visualizations
map data into quantifiable aesthetics
features of the resulting graphic

data → aesthetics



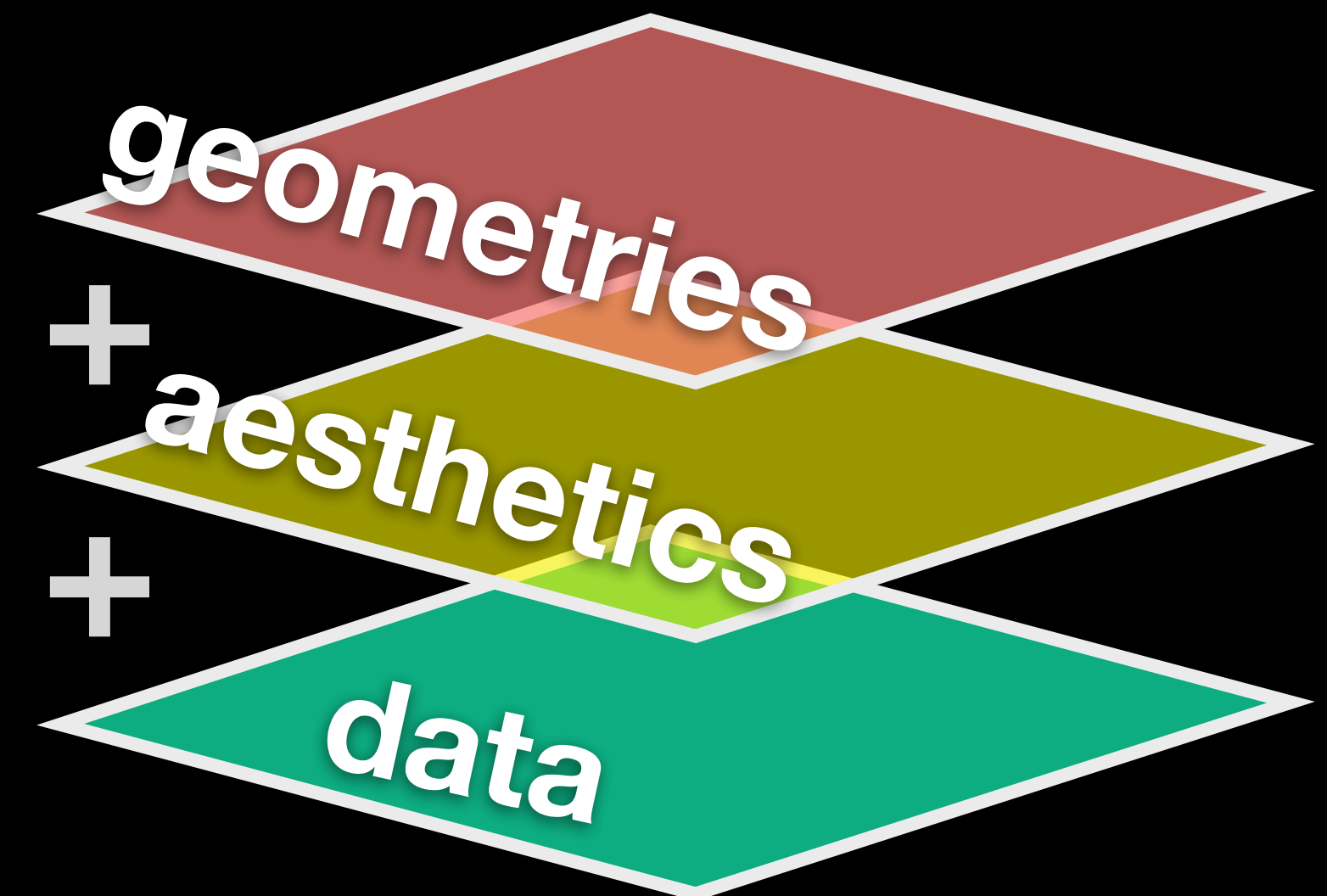
data + **aes**thetics + **geom**etrys

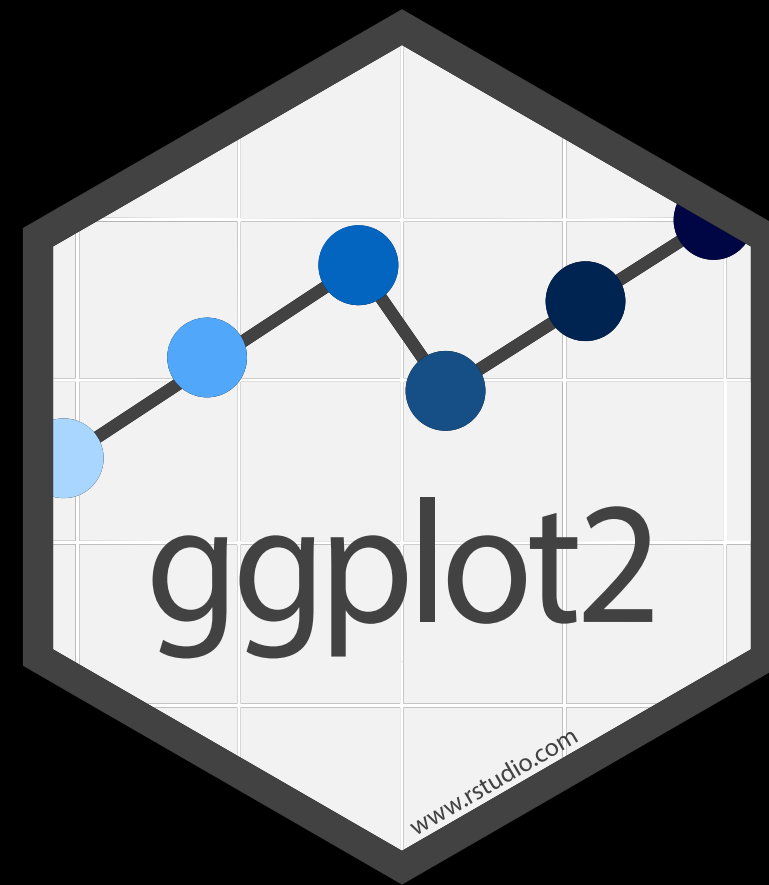
Three main "layers"
that are in every ggplot



data + **aes**thetics + **geom**etrys

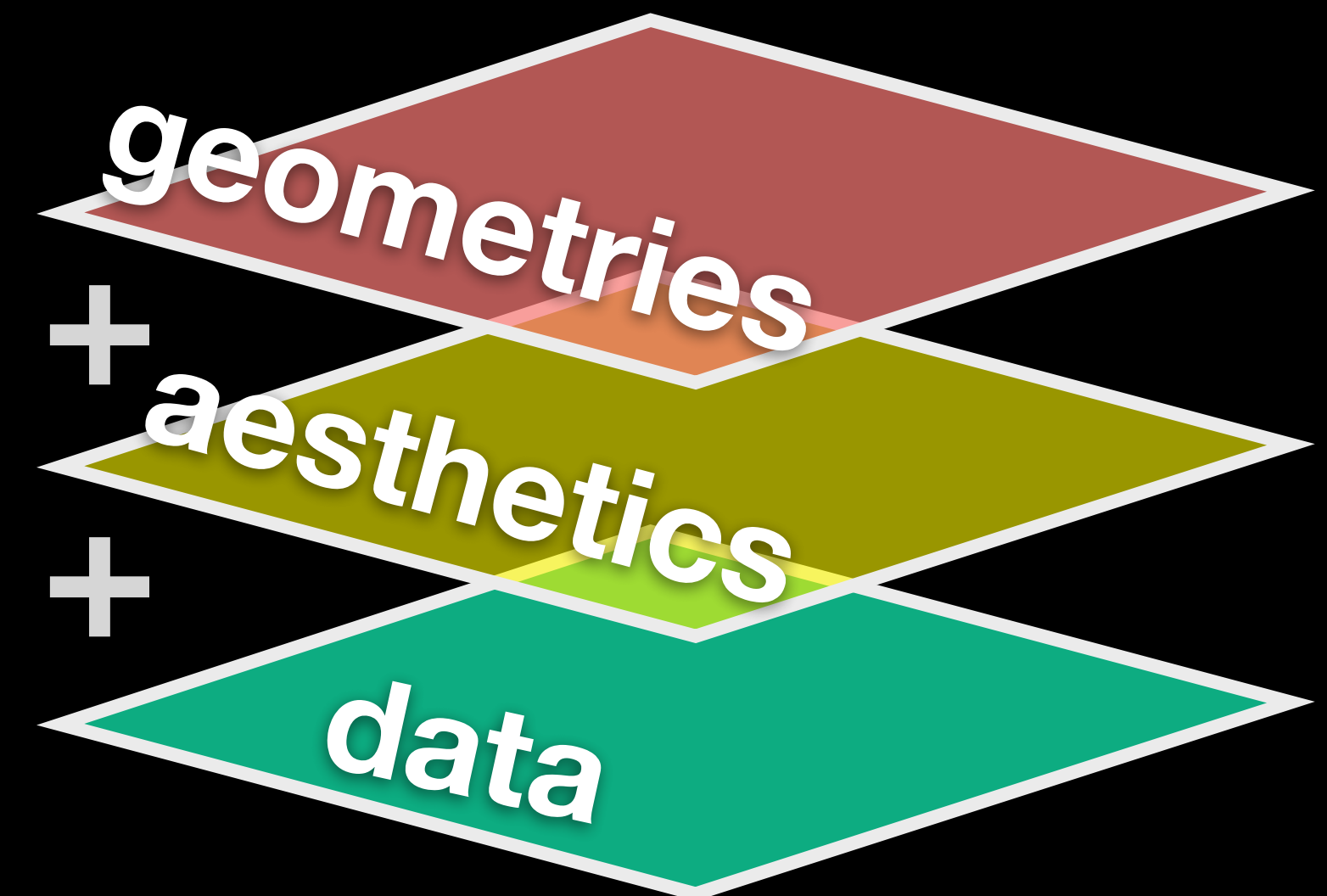
Three main "layers"
that are in every ggplot

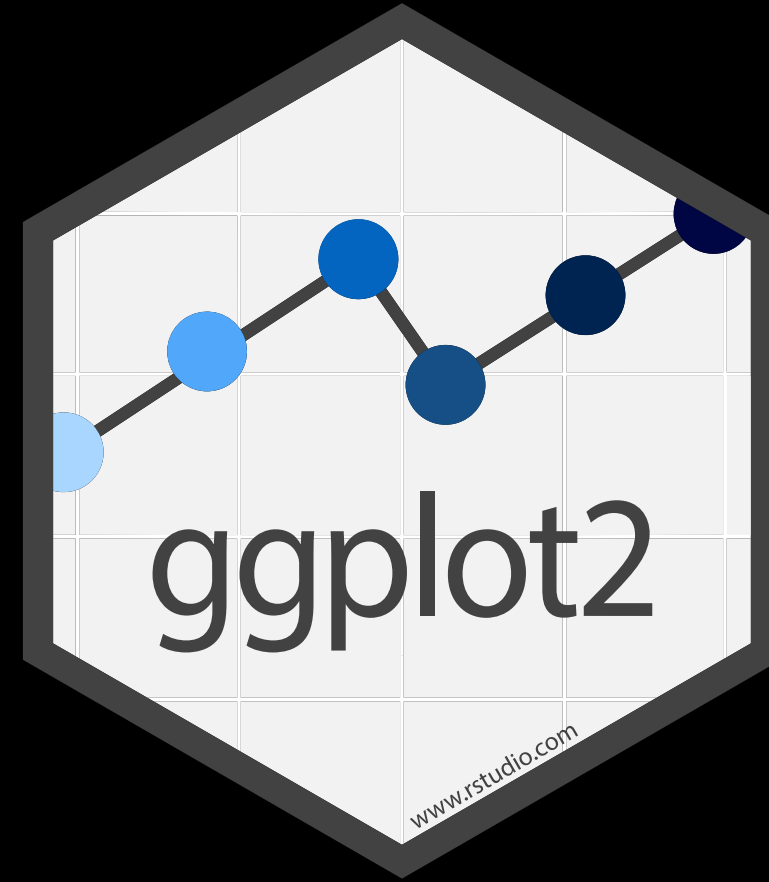




data + **aes**thetics + **geom**etrys

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





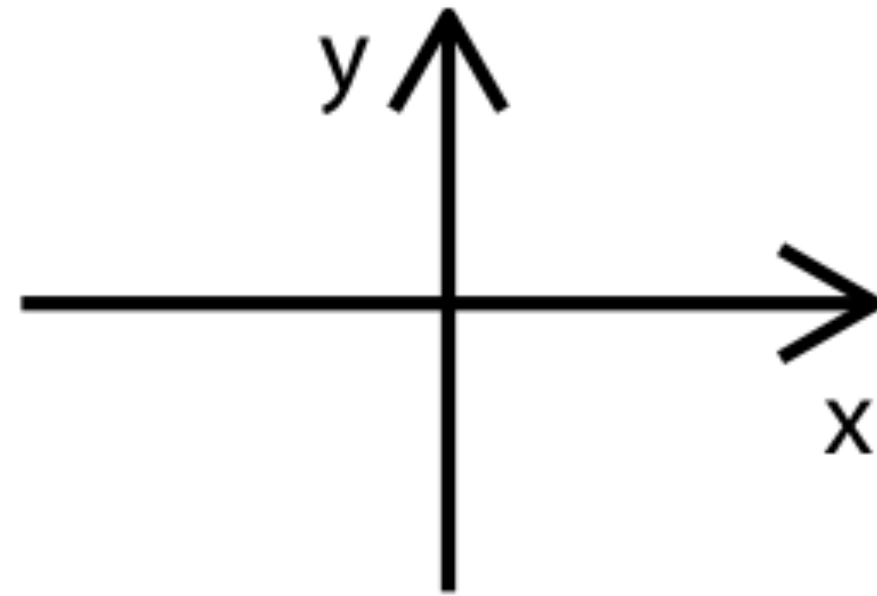
data + **aesthetics** + **geometries**

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



Common aesthetics include

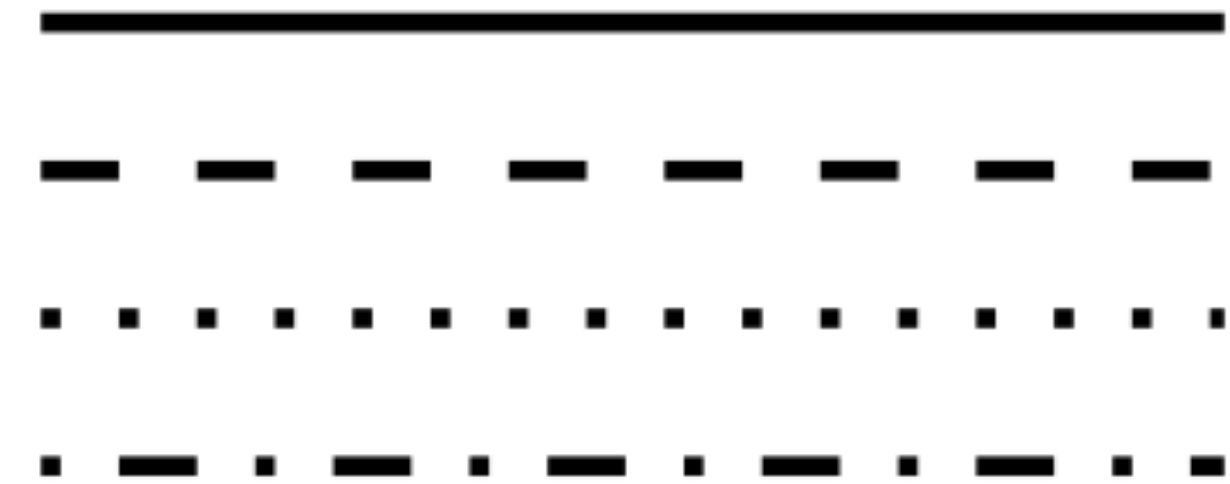
position



size



line type



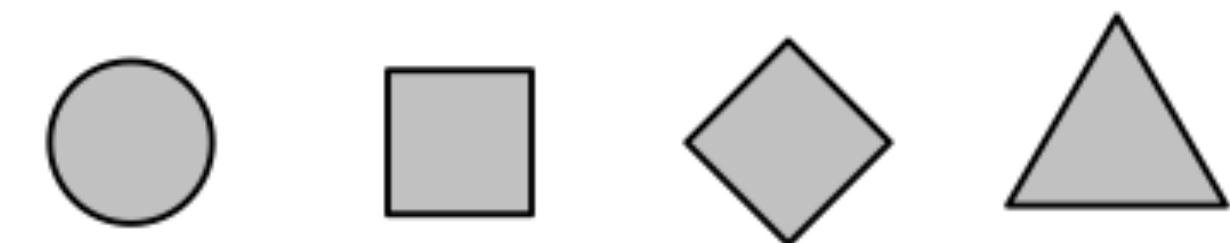
line width

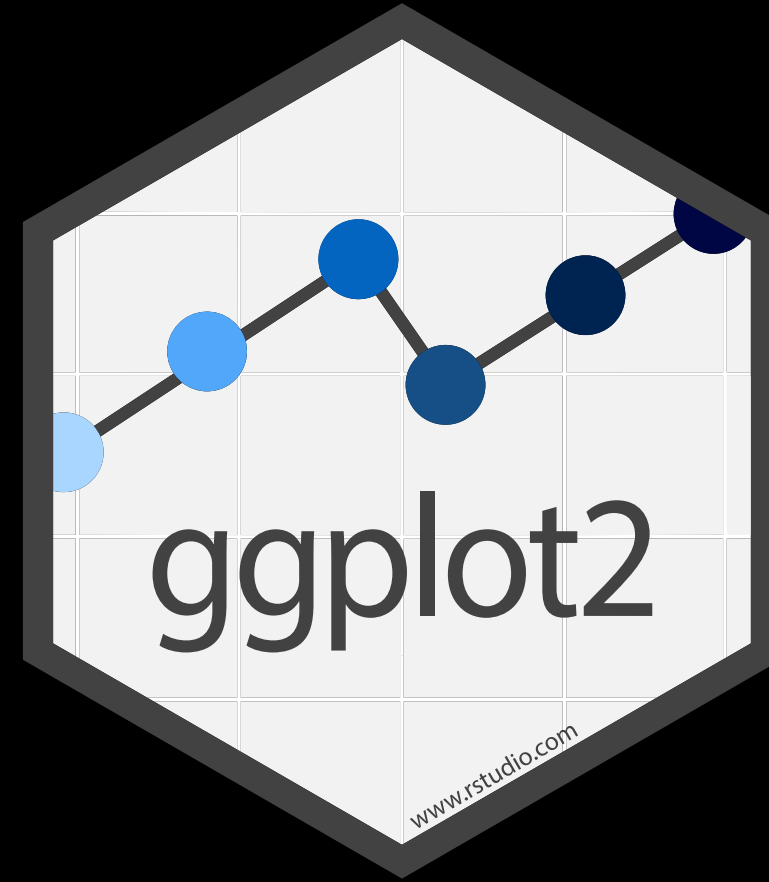


color



shape

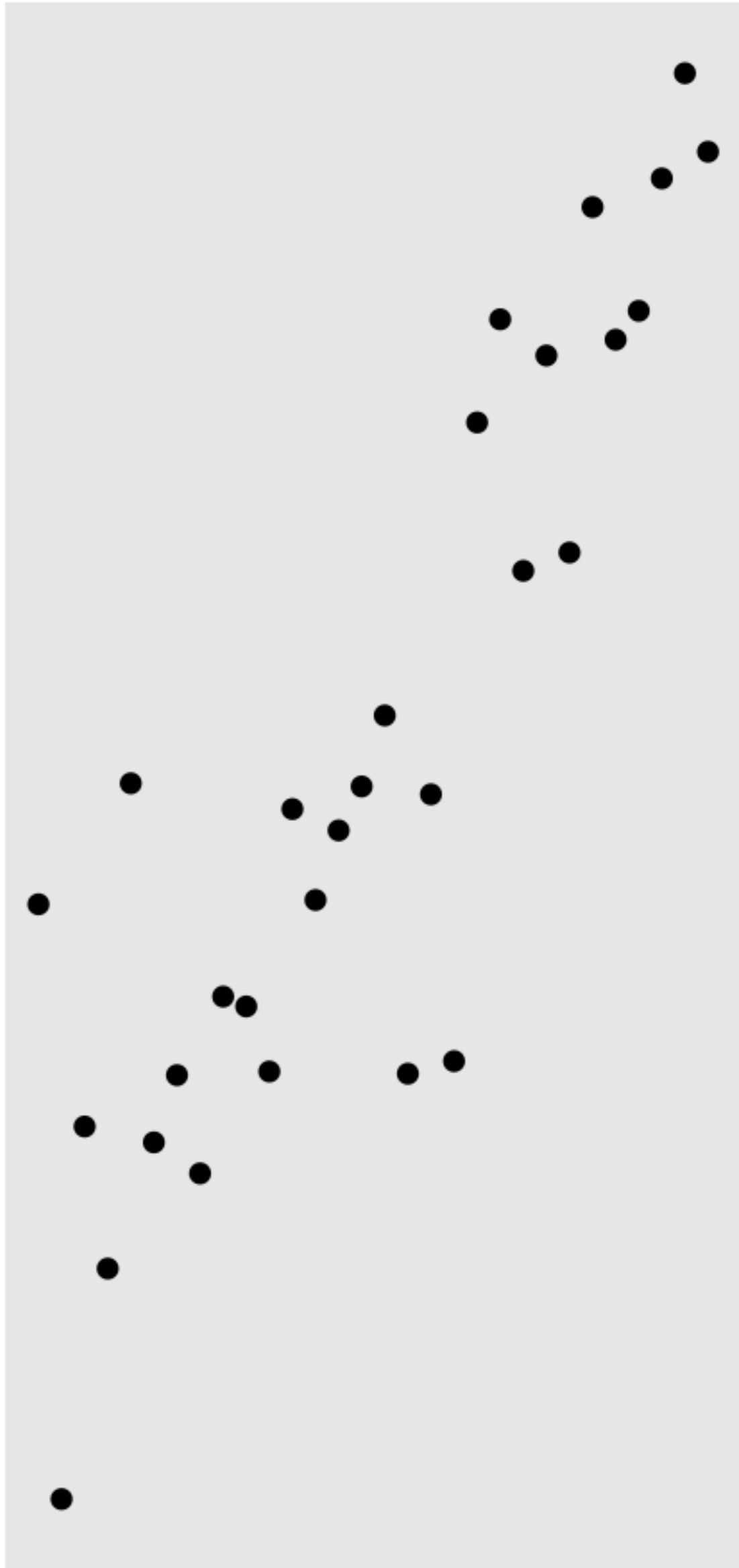




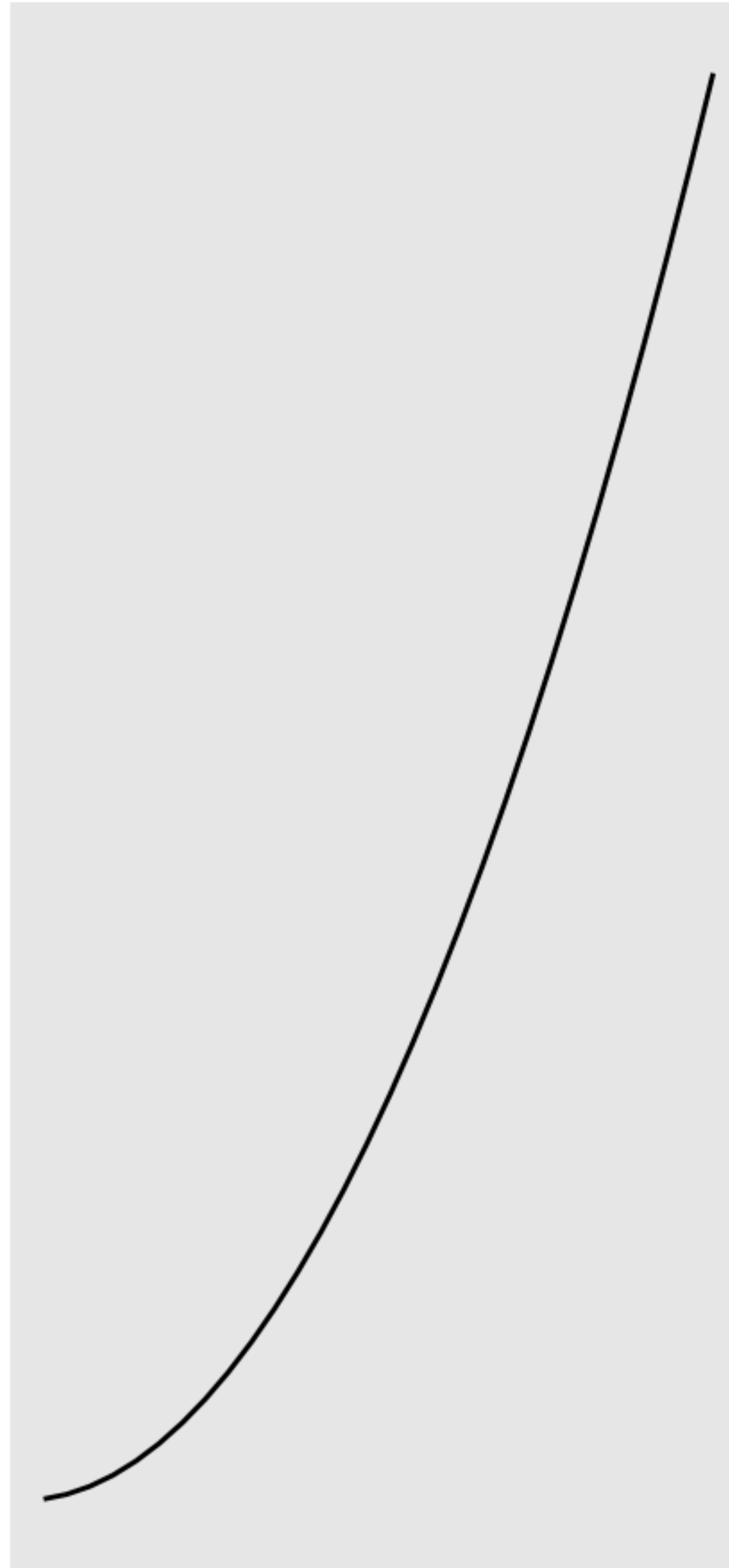
data + **aes**thetics + **geom**etrys

Three main "layers"
that are in every ggplot

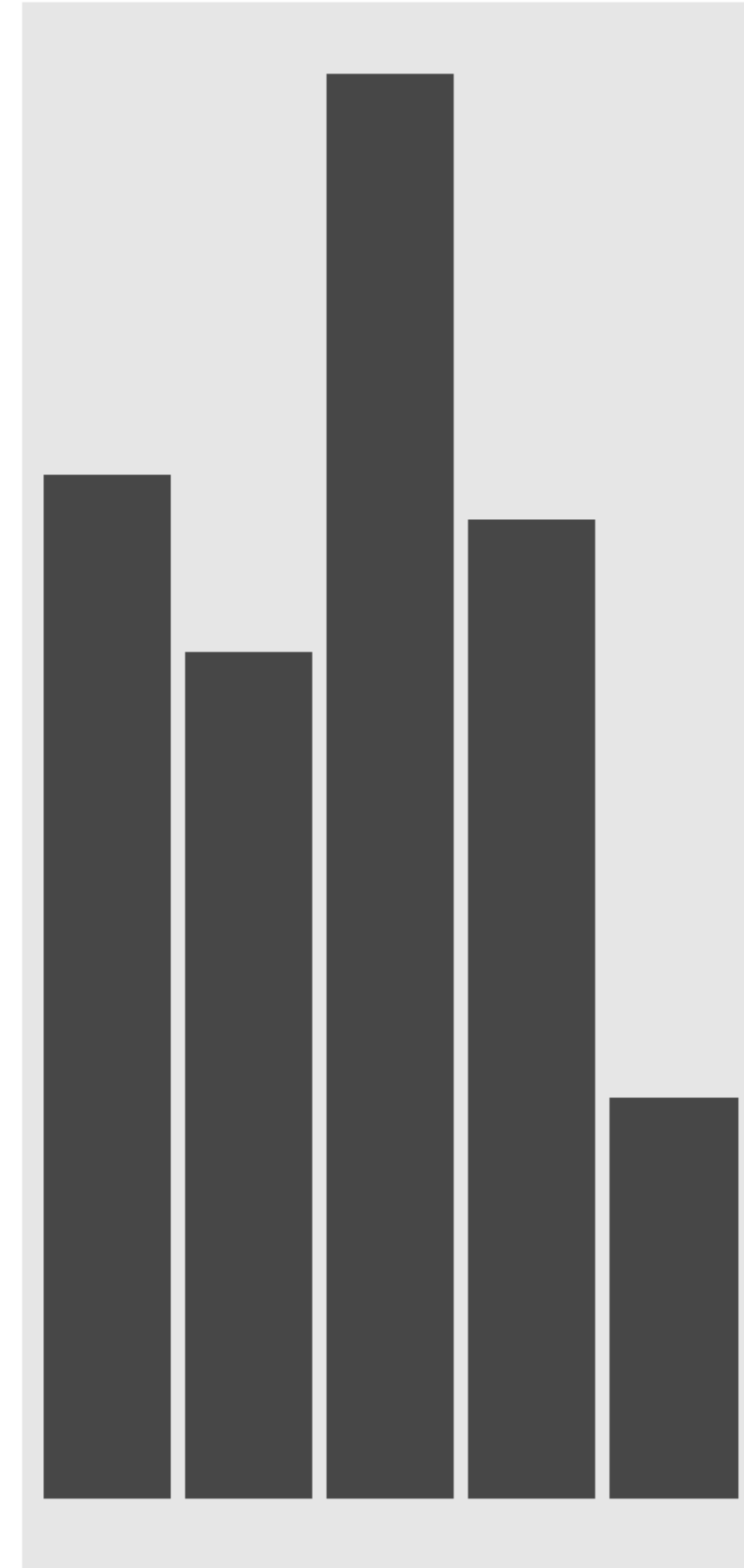
geom_point()



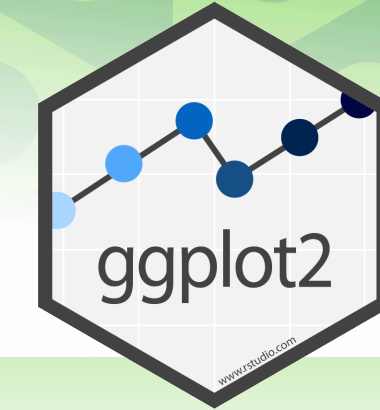
geom_line()



geom_col()



Data Visualization with ggplot2 : : CHEAT SHEET

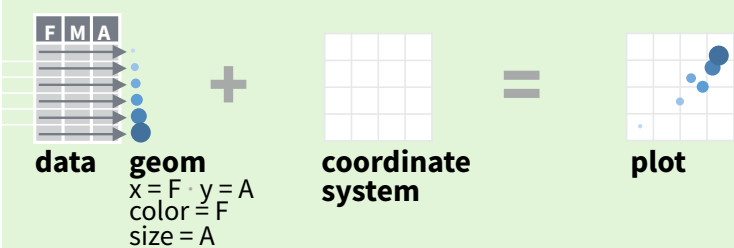


Basics

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>
```

required
Not required, sensible defaults supplied

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



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, unemployment))  
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(ymin = unemployment - 900, ymax = unemployment + 900)) - x, 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 + 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 + geom_bar**() x, alpha, color, fill, linetype, size, weight

TWO VARIABLES

continuous x, continuous y

- 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 + geom_col**() x, y, alpha, color, fill, group, linetype, size
- f + geom_boxplot**() x, y, lower, middle, upper, ymax, 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 + 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)

continuous bivariate distribution

- 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 + 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

- 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, width (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

maps

- data <- data.frame(murder = USArrests\$Murder, state = tolower(rownames(USArrests)))
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

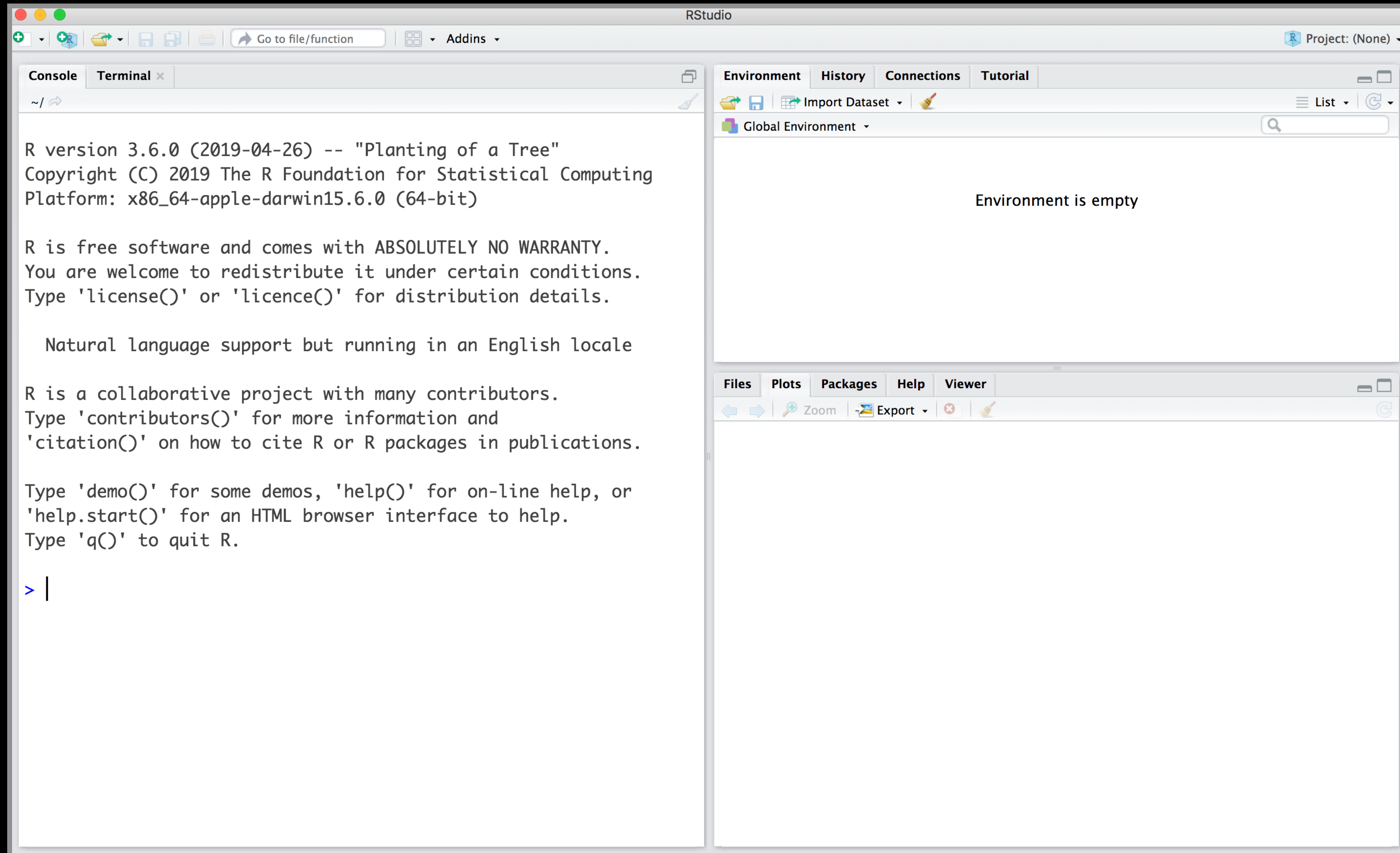
Learn more about core geom_FUNCTIONS()

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



R Studio

Follow Along!



The screenshot displays the RStudio application window. The top-left pane is the Console, showing the R startup message and a prompt. The top-right pane is the Environment pane, which is currently empty. The bottom-right pane is the Files pane, which is also empty. The interface includes a menu bar at the top with options like 'Go to file/function', 'Addins', and 'Project: (None)'. The console output includes the R version (3.6.0), copyright information, and instructions on how to use R.

```
R version 3.6.0 (2019-04-26) -- "Planting of a Tree"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```


R Studio

Follow Along!

The screenshot shows the RStudio interface. The top-left pane is the Source editor. The bottom-left pane is the Console, which contains the following text:

```
R version 3.6.0 (2019-04-26) -- "Planting of a Tree"  
Copyright (C) 2019 The R Foundation for Statistical Computing  
Platform: x86_64-apple-darwin15.6.0 (64-bit)  
  
R is free software and comes with ABSOLUTELY NO WARRANTY.  
You are welcome to redistribute it under certain conditions.  
Type 'license()' or 'licence()' for distribution details.  
  
Natural language support but running in an English locale  
  
R is a collaborative project with many contributors.  
Type 'contributors()' for more information and  
'citation()' on how to cite R or R packages in publications.  
  
Type 'demo()' for some demos, 'help()' for on-line help, or  
'help.start()' for an HTML browser interface to help.  
Type 'q()' to quit R.  
  
> library(ggplot2)  
> ggplot(mpg, aes(displ, hwy, color=class)) +  
+ geom_point()  
> |
```

The bottom-right pane is the Plots window, displaying a scatter plot of highway mileage (hwy) versus engine displacement (displ) for different car classes. The plot is colored by class, with a legend on the right:

- 2seater (red)
- compact (yellow)
- midsize (green)
- minivan (teal)
- pickup (cyan)
- subcompact (purple)
- suv (pink)

The Environment pane shows "Global Environment" and "Environment is empty".

Input

Output

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**

bioboot.github.io

Home Gmail Gcal GitHub BIMM143 BGGN213 GDrive Atmosphere CloudLaunch BIMM194 Blink News 0/9

- 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

• 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?



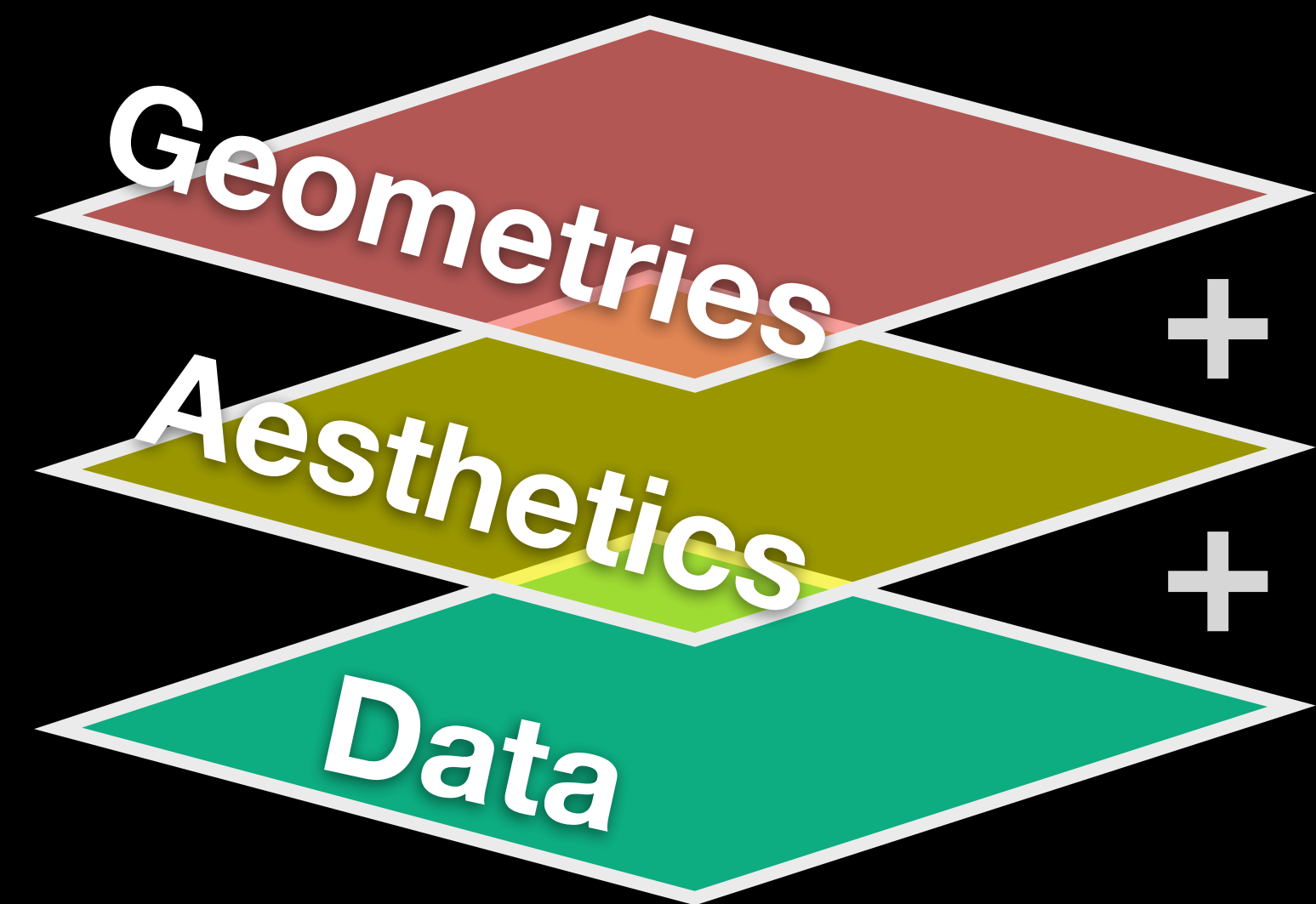
- If so you can now generate a nice **PDF report** of your work for upload to **GradeScope...**

[Optional Sections get you bonus points!]

data + aesthetics + 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()`, ...)

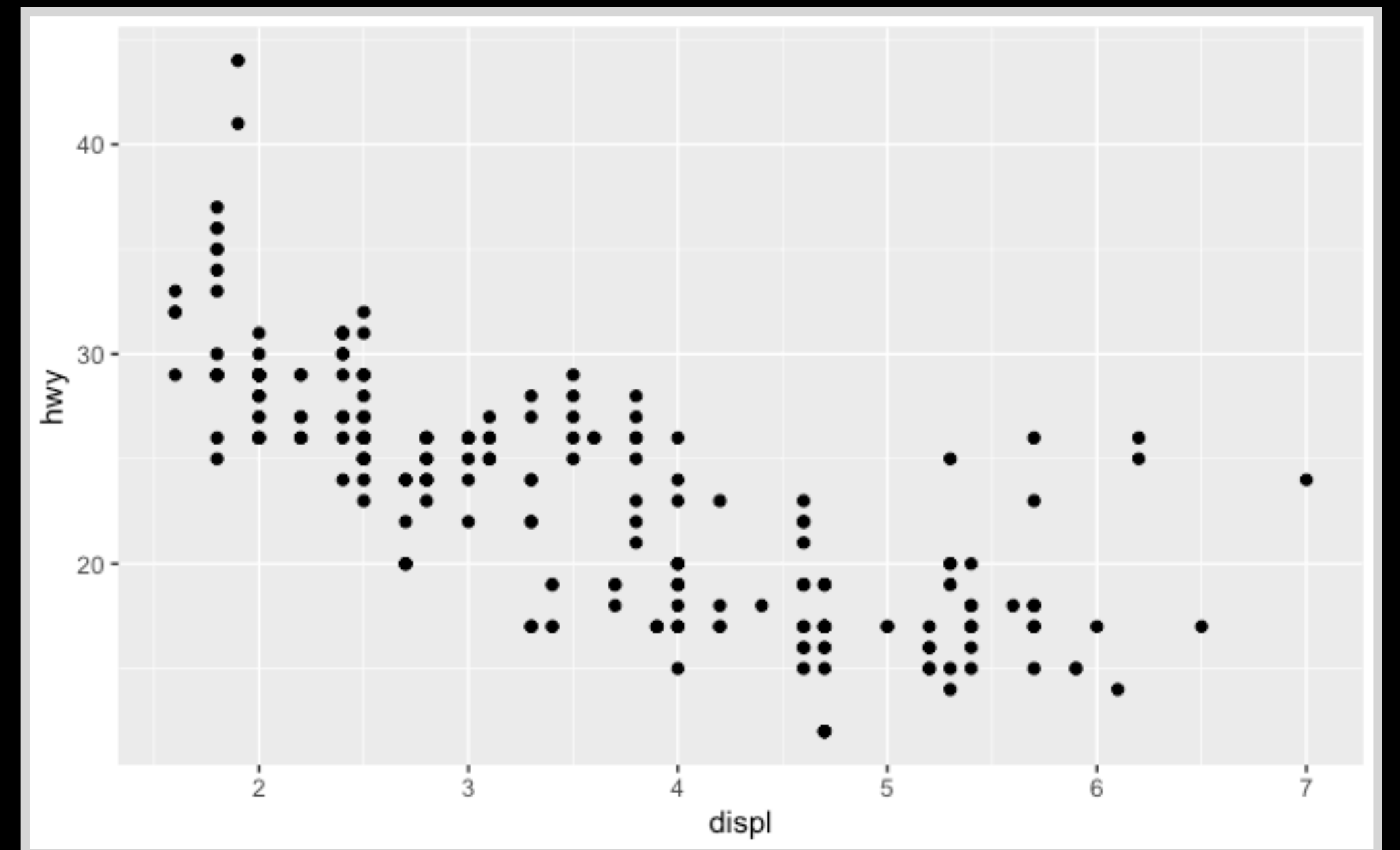
```
ggplot(data=mpg) +  
  aes(x=displ, y=hwy) +  
  geom_point()
```



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()*, ...)

```
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 )
```

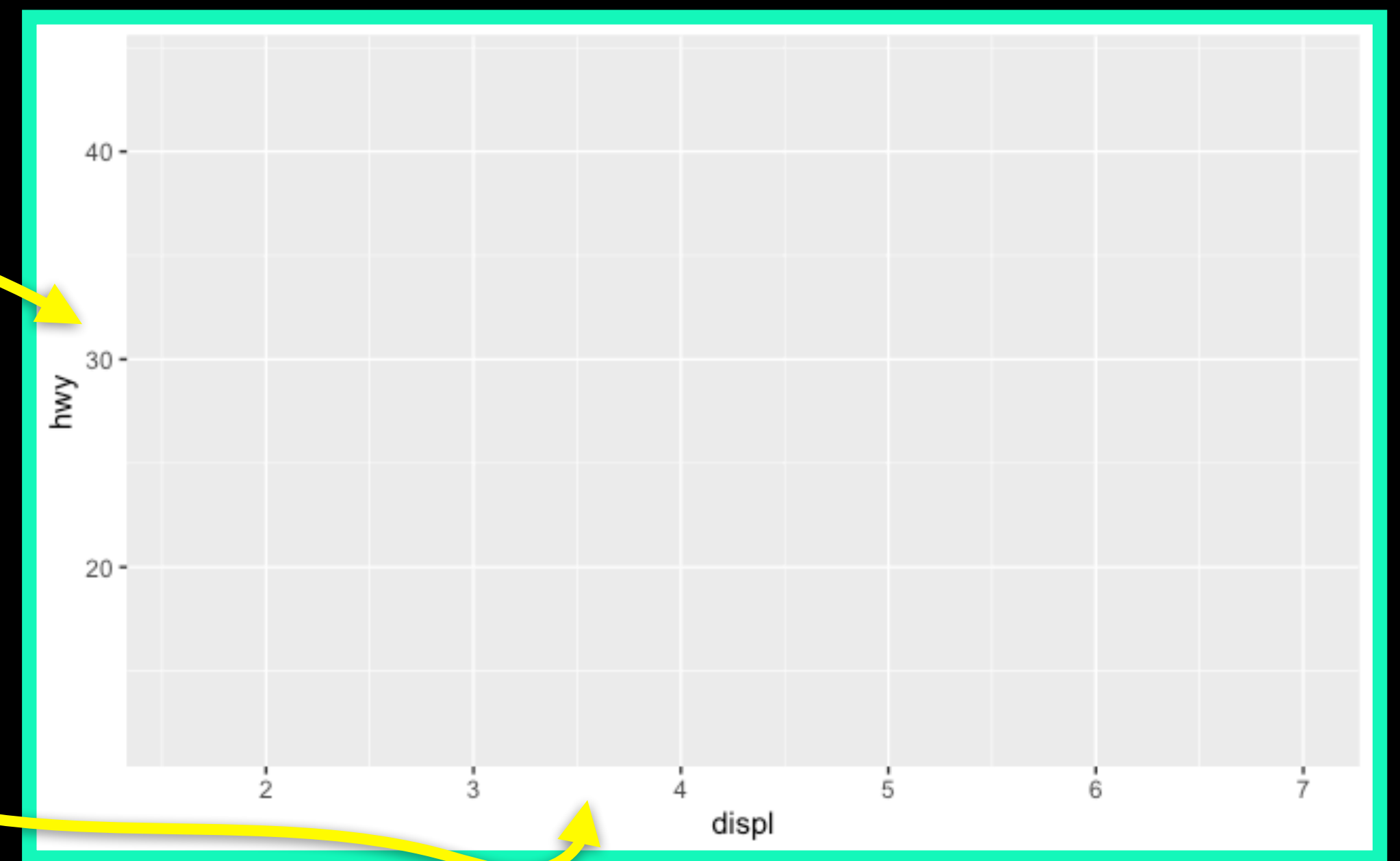


The diagram illustrates the process of creating a ggplot. On the left, a dark blue box contains the R code `ggplot (data=mpg)`. A red arrow points from the underlined `data=mpg` argument to a large, empty white box on the right, which represents the resulting plot area.

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()*, ...)

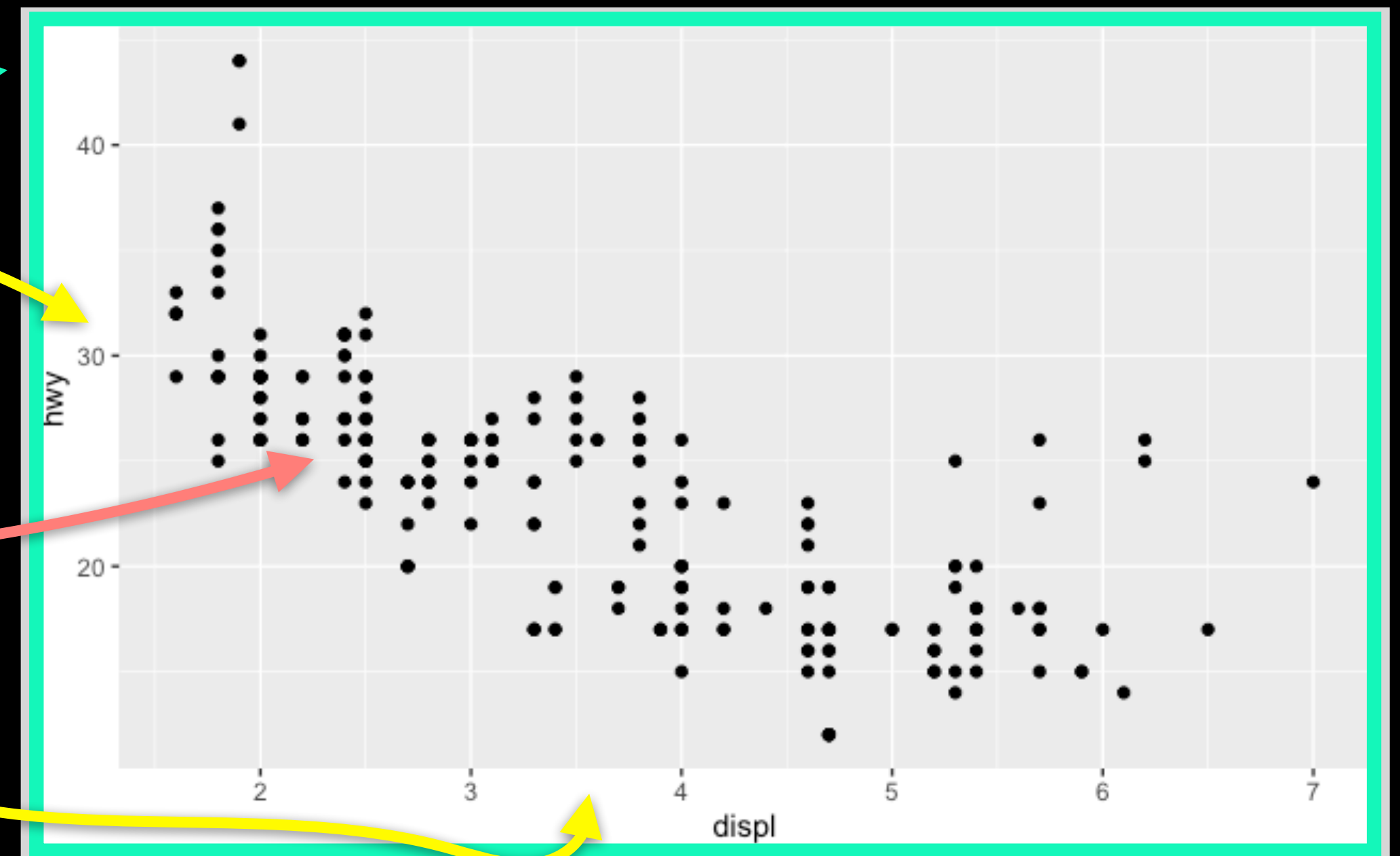
```
ggplot(data=mpg) +  
  aes(x=displ, y=hwy)
```



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()*, ...)

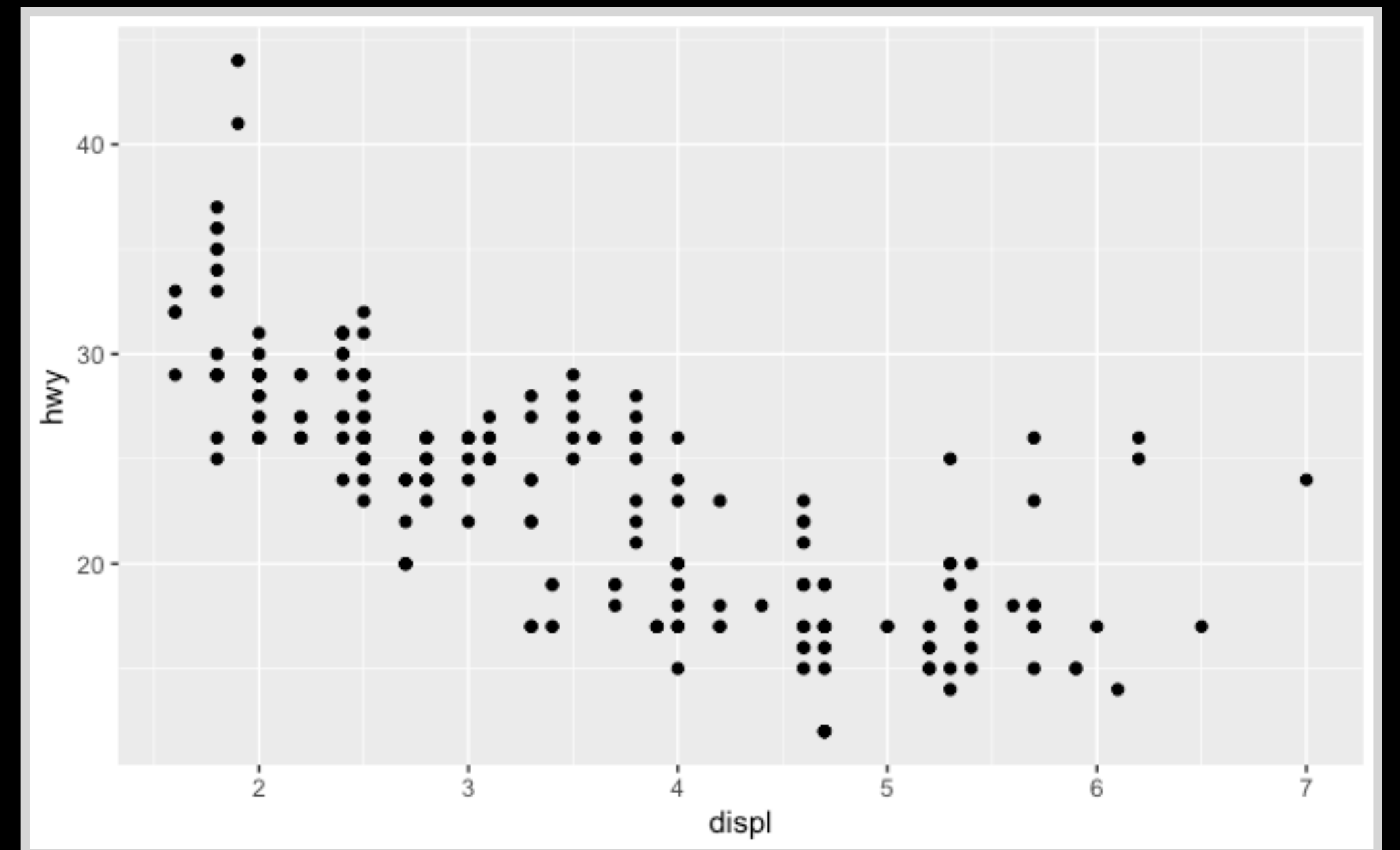
```
ggplot(data=mpg) +  
  aes(x=displ, y=hwy) +  
  geom_point()
```



data + aesthetics + geometrys

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

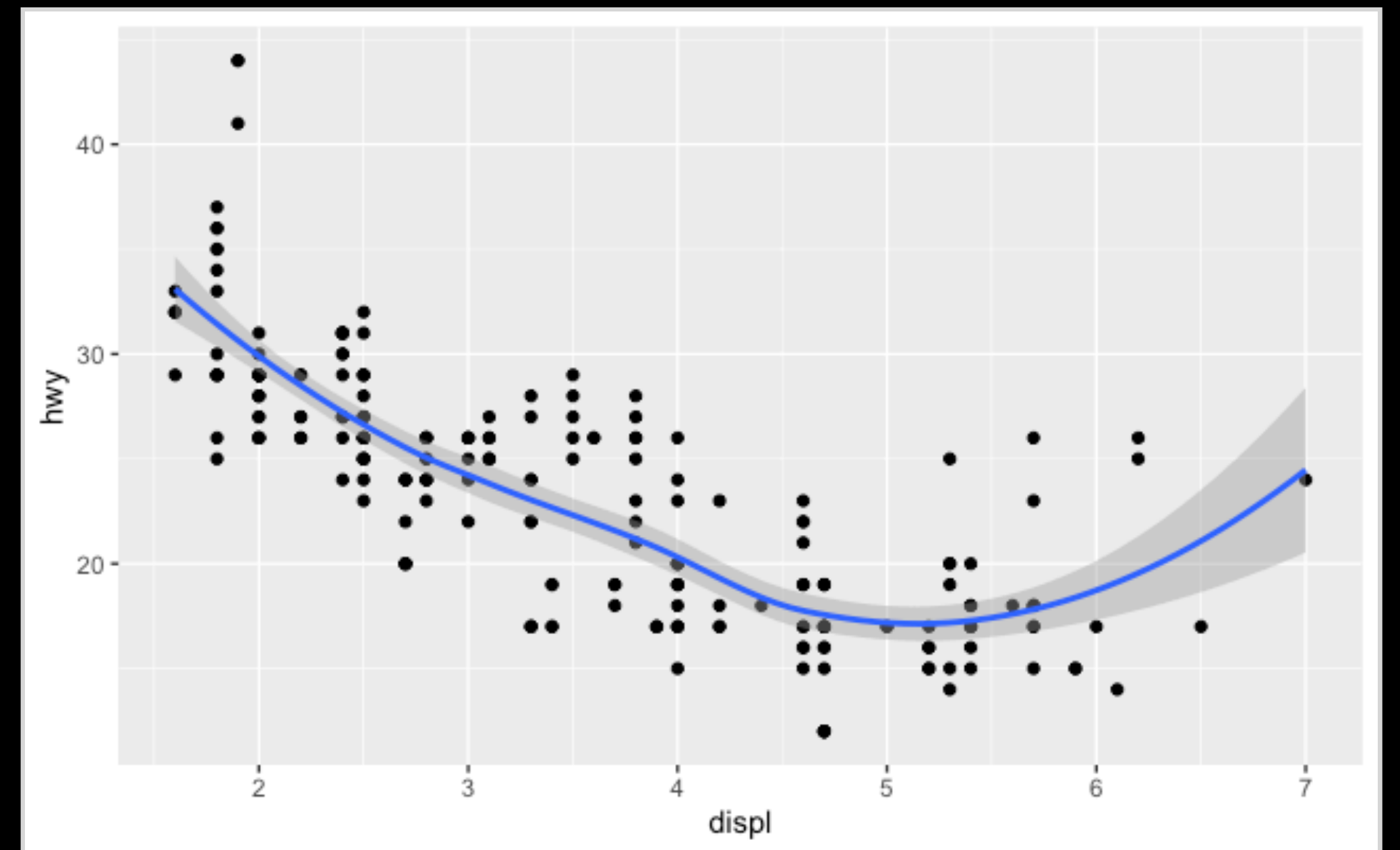
```
ggplot(data=mpg) +  
  aes(x=displ, y=hwy) +  
  geom_point()
```



data + aesthetics + geometrys

- 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...

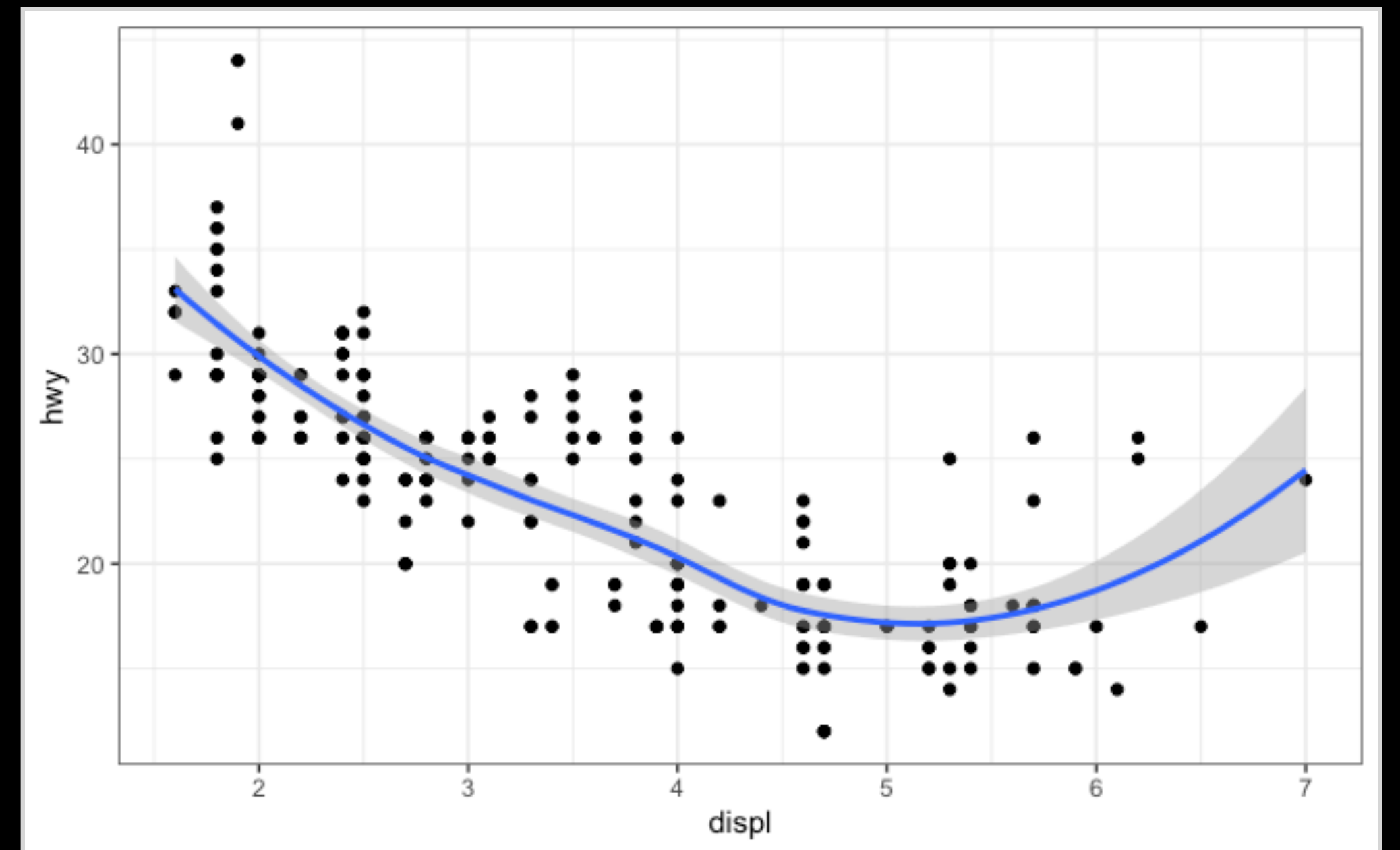
```
ggplot(data=mpg) +  
  aes(x=displ, y=hwy) +  
  geom_point() +  
  geom_smooth()
```



data + aesthetics + geometrys

- We can also add other customizations like [themes...](#)

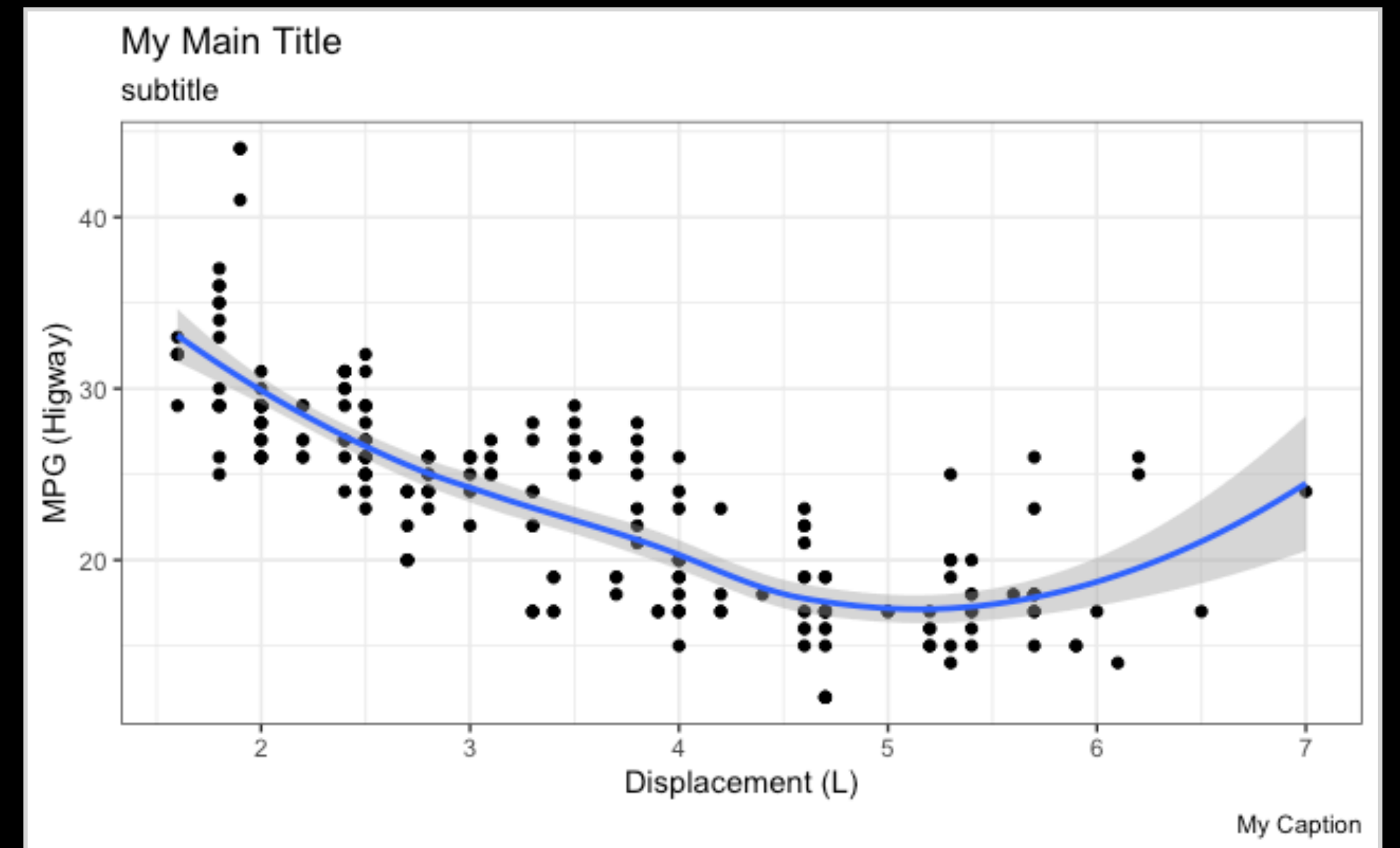
```
ggplot(data=mpg) +  
  aes(x=displ, y=hwy) +  
  geom_point() +  
  geom_smooth() +  
  theme_bw()
```



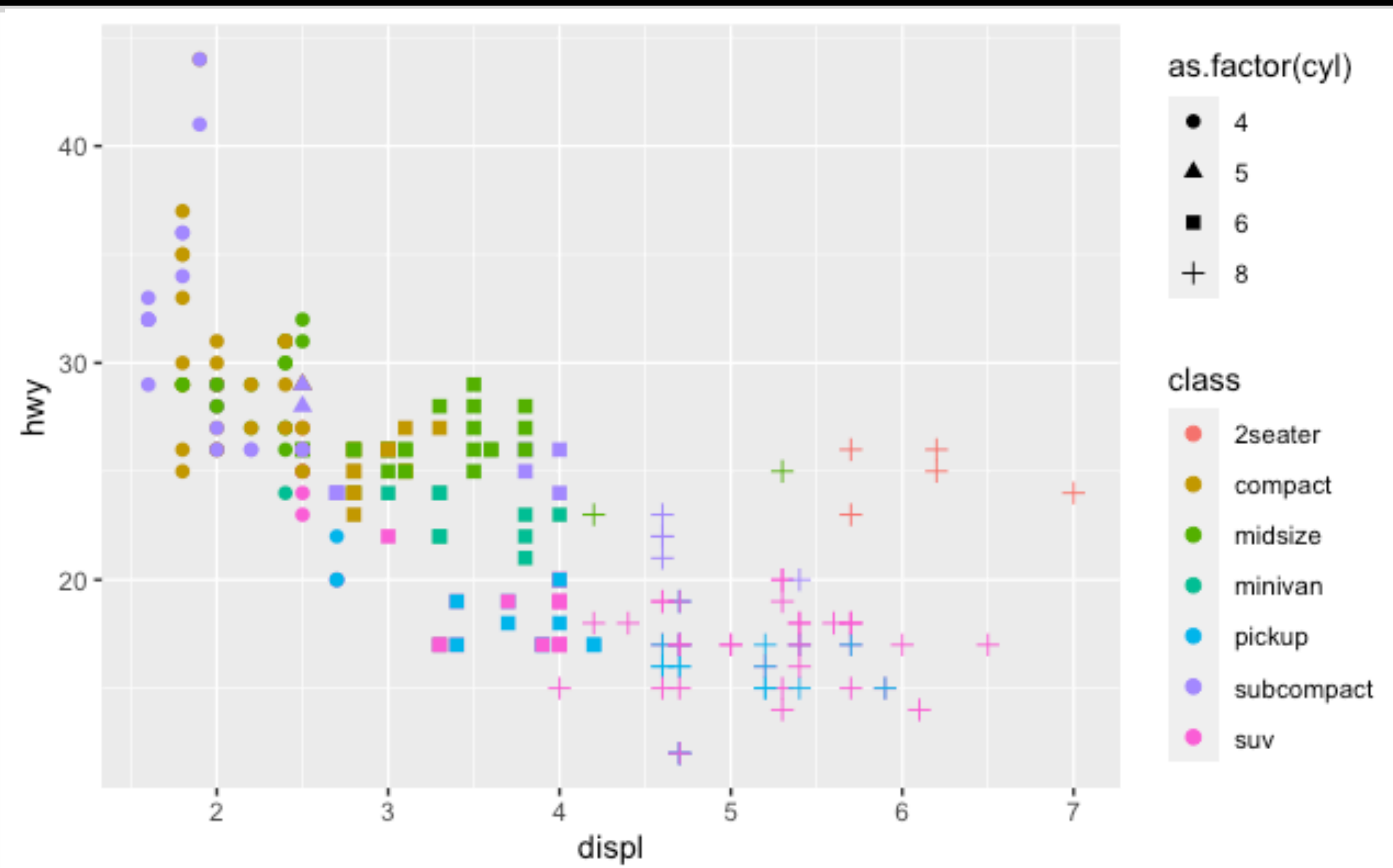
data + aesthetics + geometrys

- And various custom annotation labels...

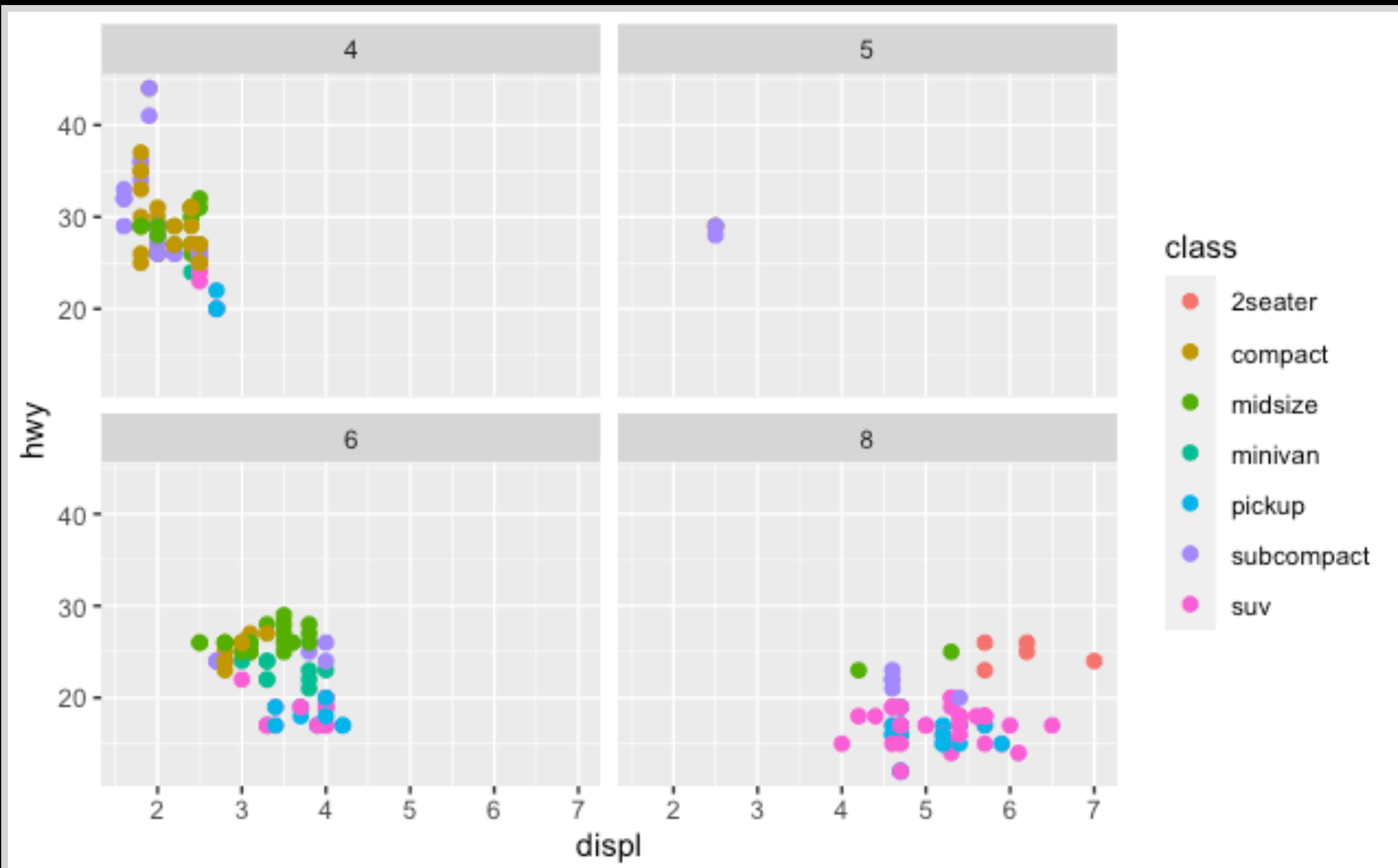
```
ggplot(data=mpg) +  
  aes(x=displ, y=hwy) +  
  geom_point() +  
  geom_smooth() +  
  theme_bw() +  
  labs(title="My Main Title",  
        subtitle = "subtitle",  
        caption = "My Caption",  
        x="Displacement (L)",  
        y="MPG (Higway)")
```



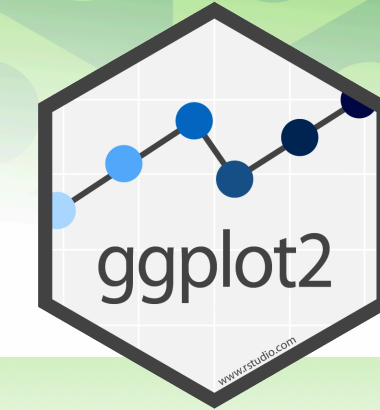
```
ggplot(data=mpg) +  
  aes(x=displ, y=hwy, color=class,  
      shape=factor(cyl)) +  
  geom_point()
```



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



Data Visualization with ggplot2 : : CHEAT SHEET

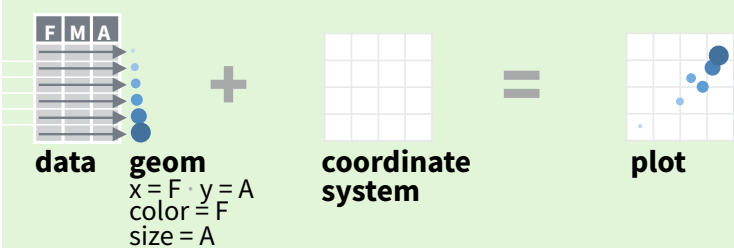


Basics

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>
```

required
Not required, sensible defaults supplied

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



`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) - xmin, xmax, ymin, ymax, alpha, color, fill, linetype, size)
- a + geom_ribbon**(aes(ymin = unemploy - 900, ymax = unemploy + 900)) - x, 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 + 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 + geom_bar**() x, alpha, color, fill, linetype, size, weight

TWO VARIABLES

continuous x, continuous y

- 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 + geom_col**() x, y, alpha, color, fill, group, linetype, size
- f + geom_boxplot**() x, y, lower, middle, upper, ymax, 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 + geom_count**() x, y, alpha, color, fill, shape, size, stroke

THREE VARIABLES

- 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

continuous bivariate distribution

- 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 + 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

- 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, width (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

maps

- 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>