1.3 — Data Visualization with ggplot2

ECON 480 • Econometrics • Fall 2021

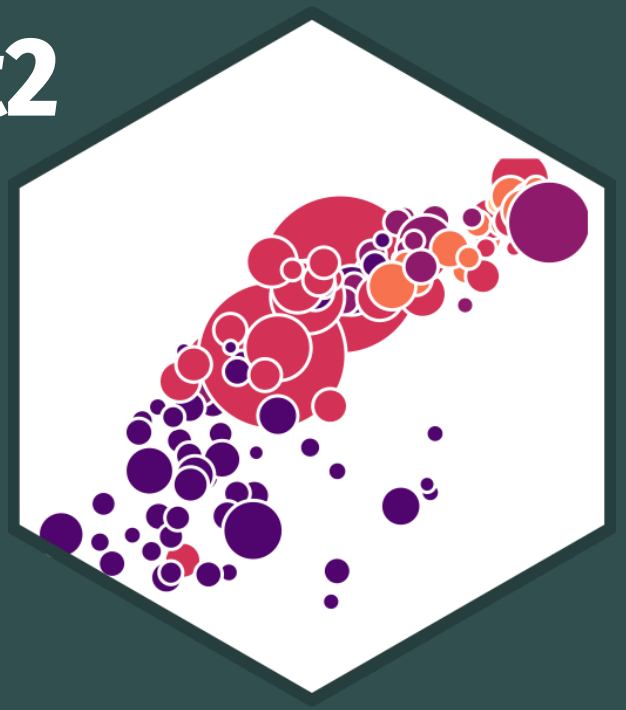
Ryan Safner

Assistant Professor of Economics

safner@hood.edu

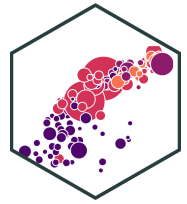
ryansafner/metricsF21

metricsF21.classes.ryansafner.com



Outline

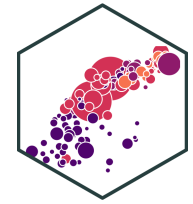
Graphics and Statistics



- Admittedly, we still need to cover basic descriptive statistics and data fundamentals
    - continuous, discrete, cross-sectional, time series, panel data
    - mean, median, variance, standard deviation
    - random variables, distributions, PDFs, Z-scores
    - bargraphs, boxplots, histograms, scatterplots
- All of this is coming in 2 weeks as we return to statistics and econometric theory
- But let's start with the fun stuff right away, even if you don't fully know the *reasons*: **data visualiation**

Plotting in Base R

Our Data Source

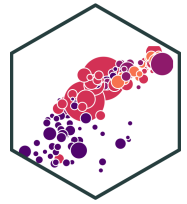


- For our examples, we'll use a dataset `mpg` from the `ggplot2` library

```
library(ggplot2)

head(mpg)
```

```
## # A tibble: 6 × 11
##   manufacturer model displ  year   cyl trans      drv     cty   hwy fl    class
##   <chr>        <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr> <chr>
## 1 audi         a4      1.8  1999     4 auto(l5)   f        18    29 p     compa…
## 2 audi         a4      1.8  1999     4 manual(m5) f        21    29 p     compa…
## 3 audi         a4      2    2008     4 manual(m6) f        20    31 p     compa…
## 4 audi         a4      2    2008     4 auto(av)   f        21    30 p     compa…
## 5 audi         a4      2.8  1999     6 auto(l5)   f        16    26 p     compa…
## 6 audi         a4      2.8  1999     6 manual(m5) f        18    26 p     compa…
```

Plotting in Base R



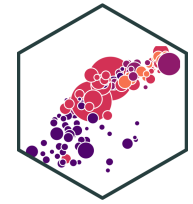
- Base `R` is very powerful and intuitive to plot, but not very sexy
- Basic syntax for most types of plots:

```
plot_type(my_df$variable)
```

- If using multiple variables, you can avoid typing `$` by just typing the variable names and then in another argument to the plotting function, specify `data = my_df`

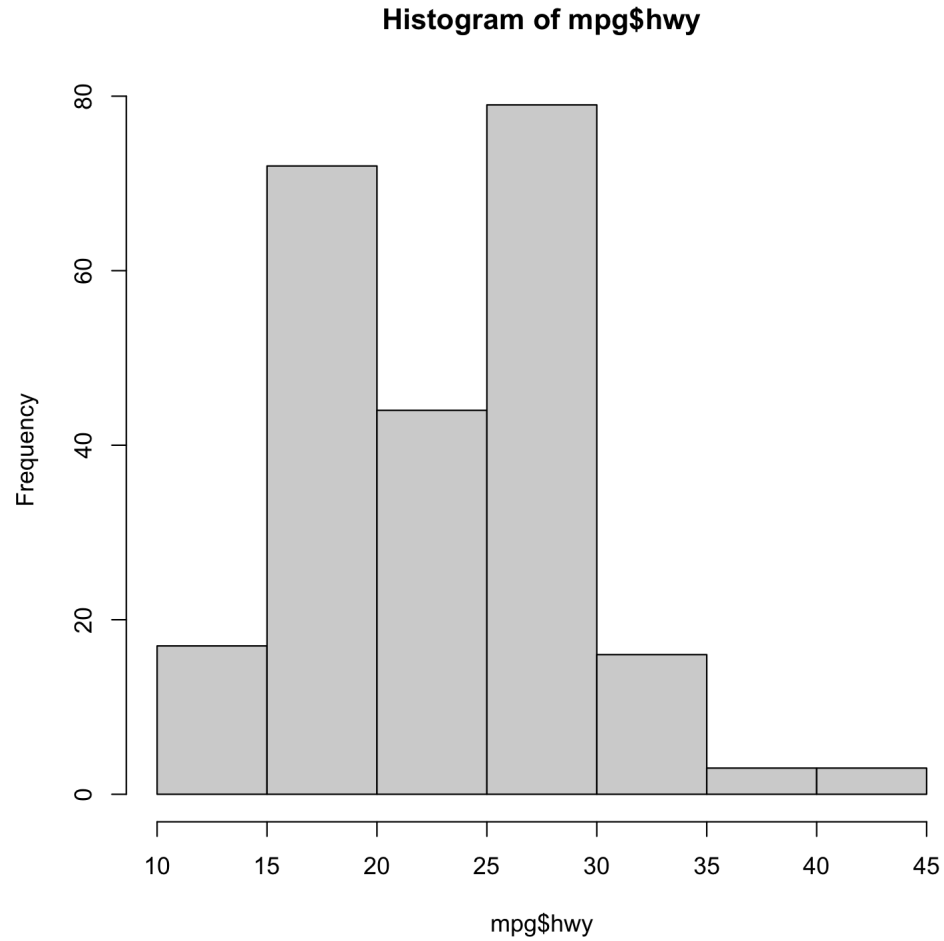
```
plot_type(my_df$variable1, my_df$variable2, data = my_df)
```

Plotting in Base R: Histogram

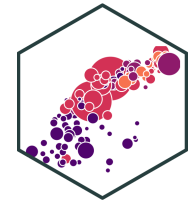


- Using the `mpg` data, plotting a **histogram** of `hwy`

```
hist(mpg$hwy)
```

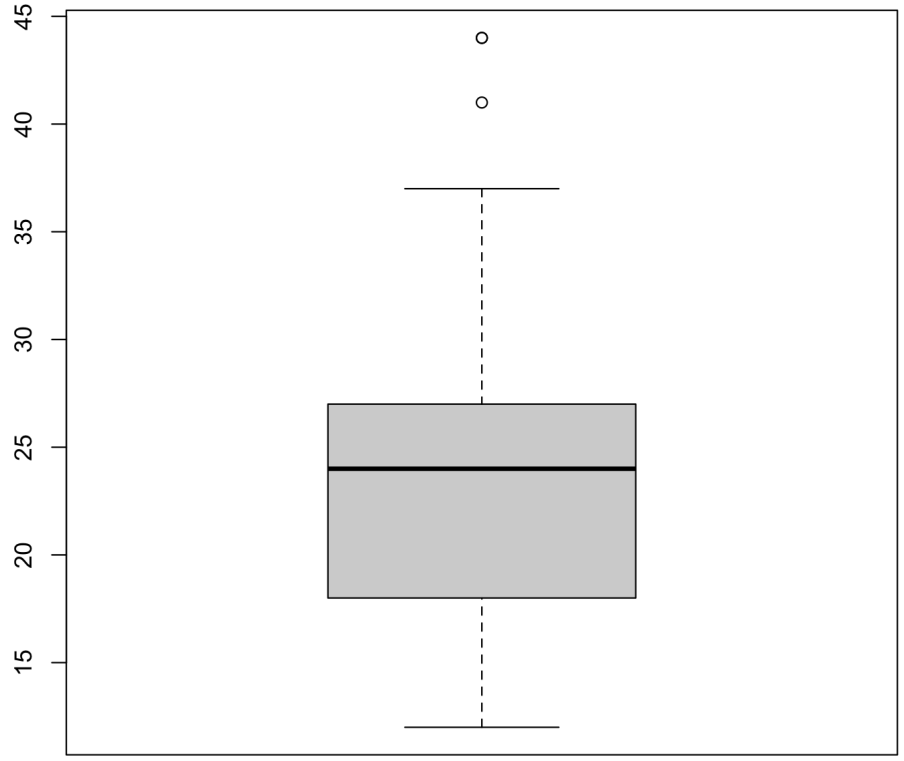


Plotting in Base R: Boxplot

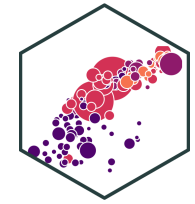


- Using the `mpg` data, plotting a **boxplot** of `hwy`

```
boxplot(mpg$hwy)
```



Plotting in Base R: Boxplot by Category

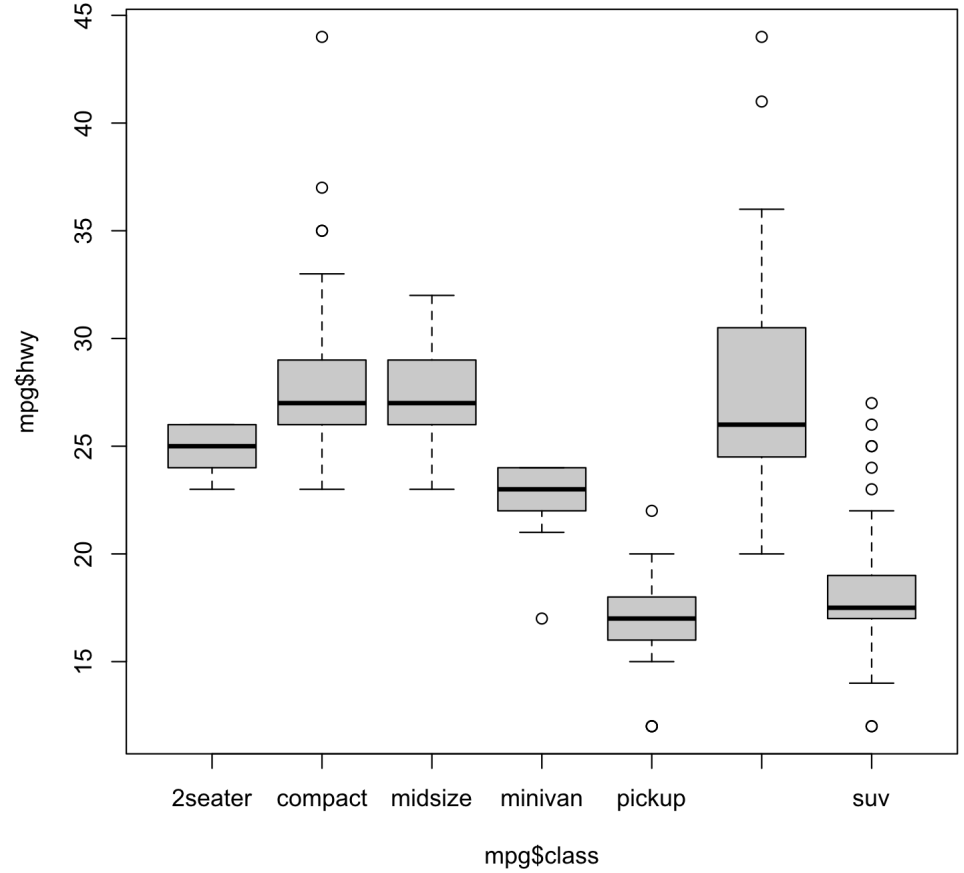


- Using the `mpg` data, plotting a **boxplot** of `hwy` **by** `class`

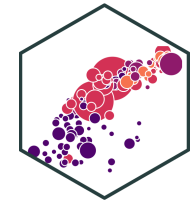
```
boxplot(mpg$hwy ~ mpg$class)
```

```
# second method
boxplot(mpg ~ class, data = mtcars)
```

- The `~` is part of `R`'s **"formula notation"**:
  - Dependent variable goes to left
  - Independent variable(s) to right, separated with `+`'s
  - Think `y~x+z` means "`y` is explained by `x` and `z`"



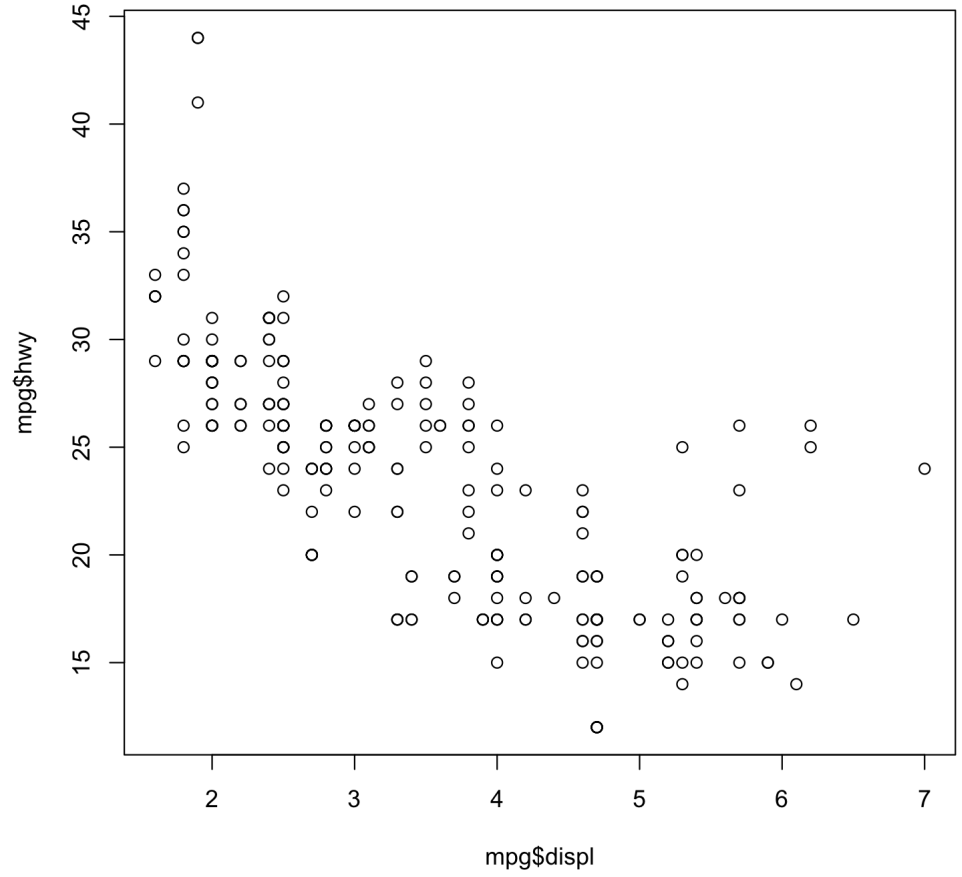
Plotting in Base R: Scatterplot



- Using the `mpg` data, plotting a **scatterplot** of `hwy` against `displ`

```
plot(mpg$hwy ~ mpg$displ)
```

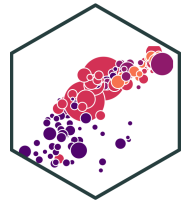
```
# second method
plot(hwy ~ displ, data = mpg)
```



ggplot2 and the tidyverse


tidyverse

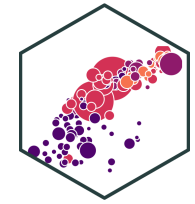
The tidyverse



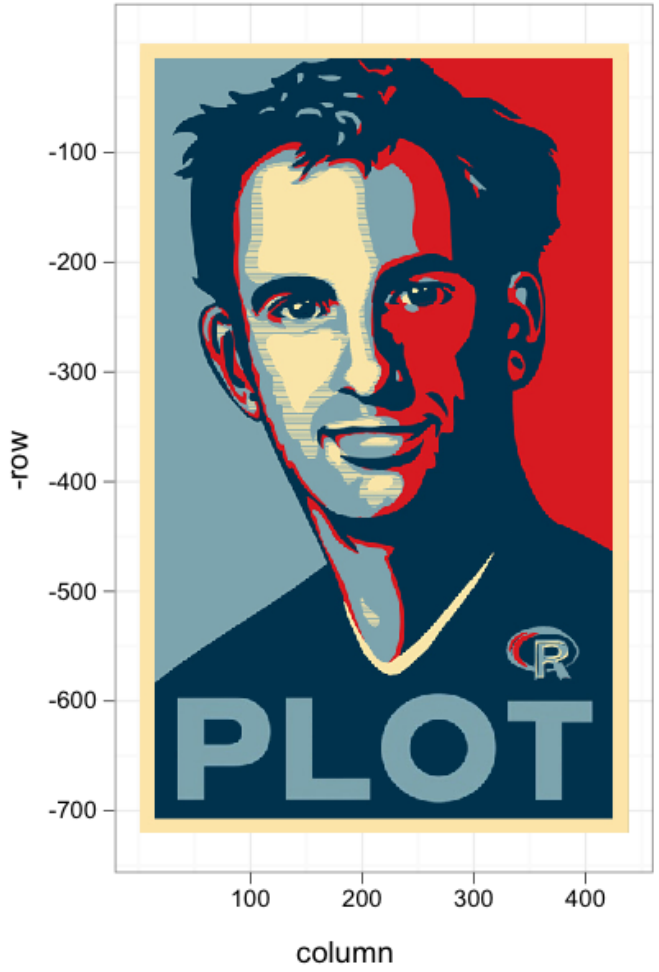
> "The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures.

- Largely (but not only) created by Hadley Wickham
- We will look at this much more extensively next week!
- This "flavor" of `R` will make your coding life *so much easier!*

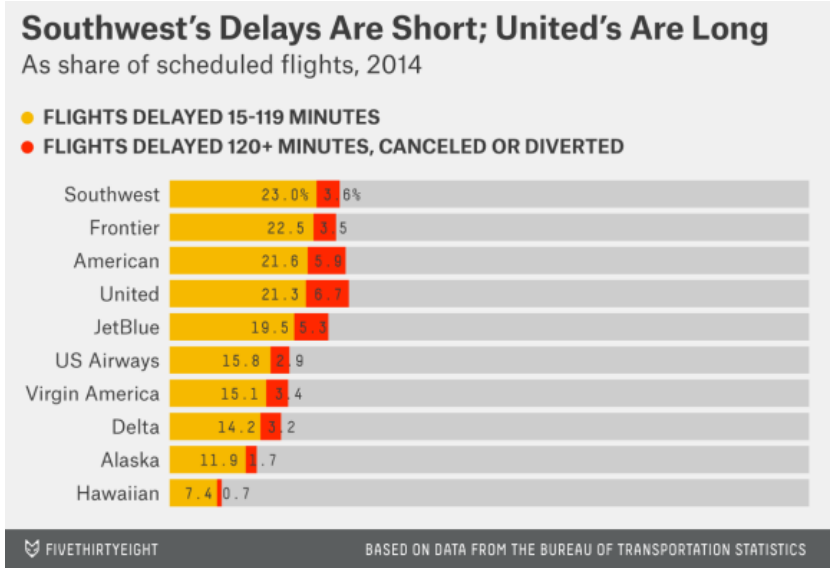
ggplot



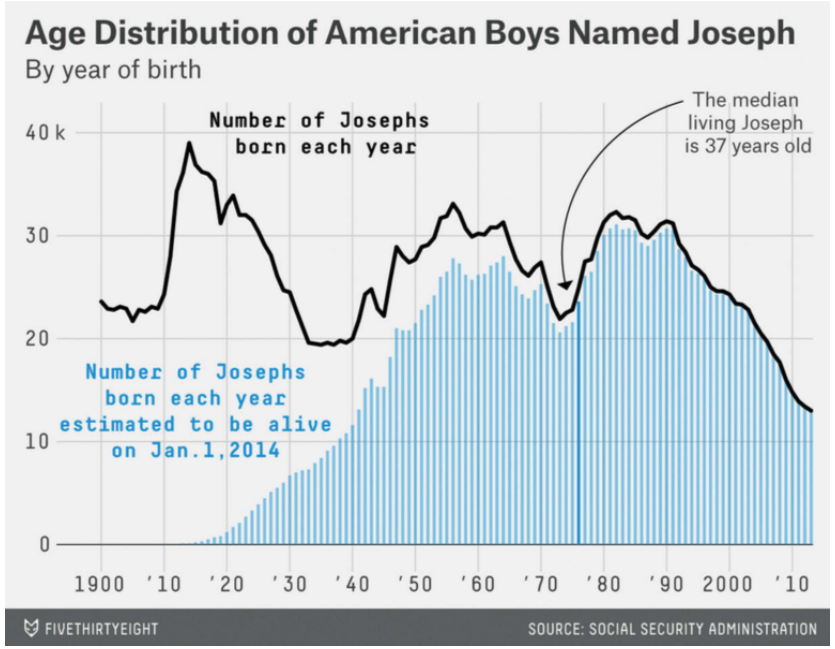
- `ggplot2` is perhaps the most popular package in `R` and a core element of the `tidyverse`
- `gg` stands for a **grammar of graphics**
- Very powerful and beautiful graphics, very customizable and reproducible, but requires a bit of a learning curve
- All those "cool graphics" you've seen in the New York Times, fivethirtyeight, the Economist, Vox, etc use the grammar of graphics



ggplot: All Your Figure are Belong to Us

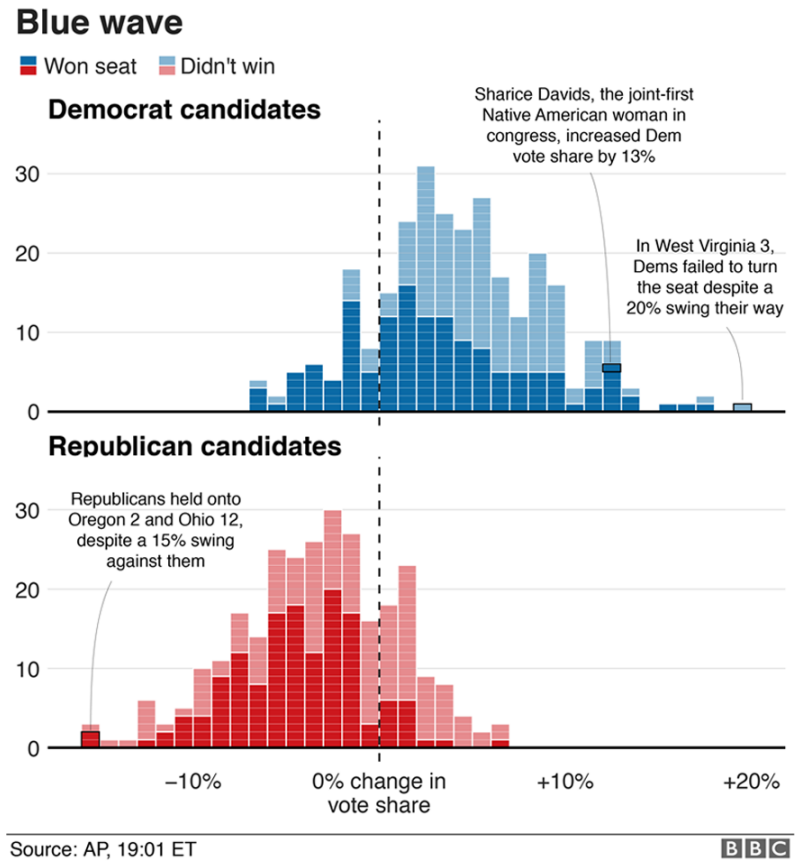


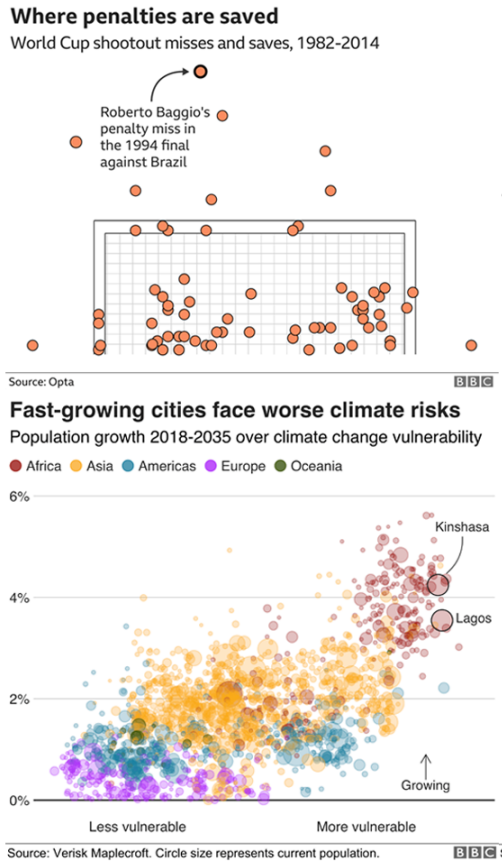
Source: fivethirtyeight



Source: fivethirtyeight

ggplot: All Your Figure are Belong to Us





Source: [BBC's bbplot]

Why Go gg?

Hadley Wickham

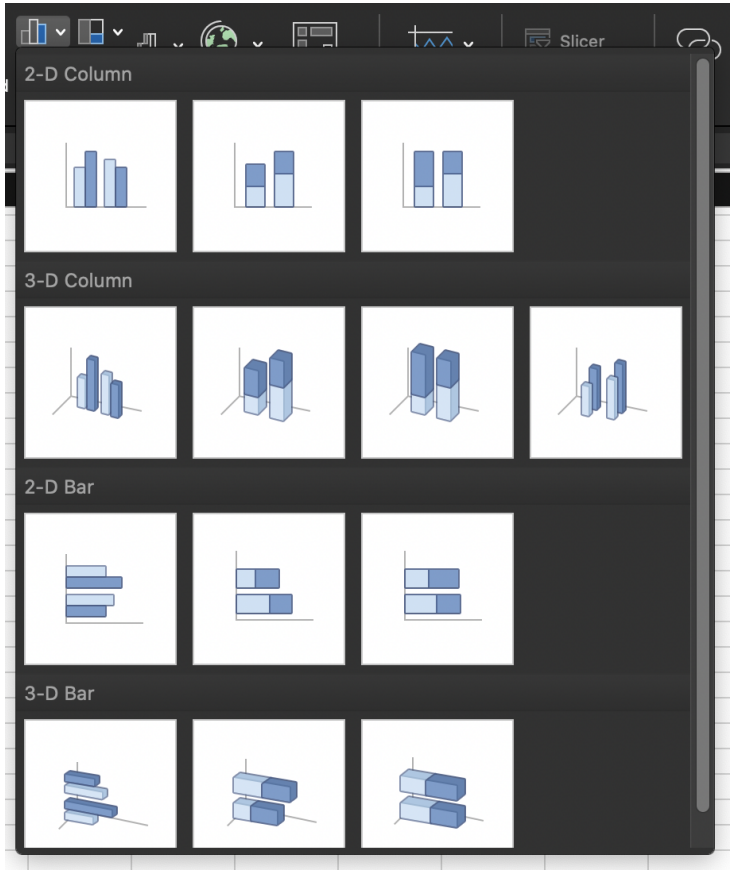
Chief Scientist, R Studio

> "The transferrable skills from ggplot2 are not the idiosyncracies of plotting syntax, but a powerful way of thinking about visualisation, as a way of **mapping between variables and the visual properties of geometric objects** that you can perceive."

http://disq.us/p/sv640d

The Grammar of Graphics (gg)

- This is a true *grammar*
- We *don't* talk about specific chart **types**
    - That you have to hunt through in Excel and reshape your data to fit it
- Instead we talk about specific chart **components**

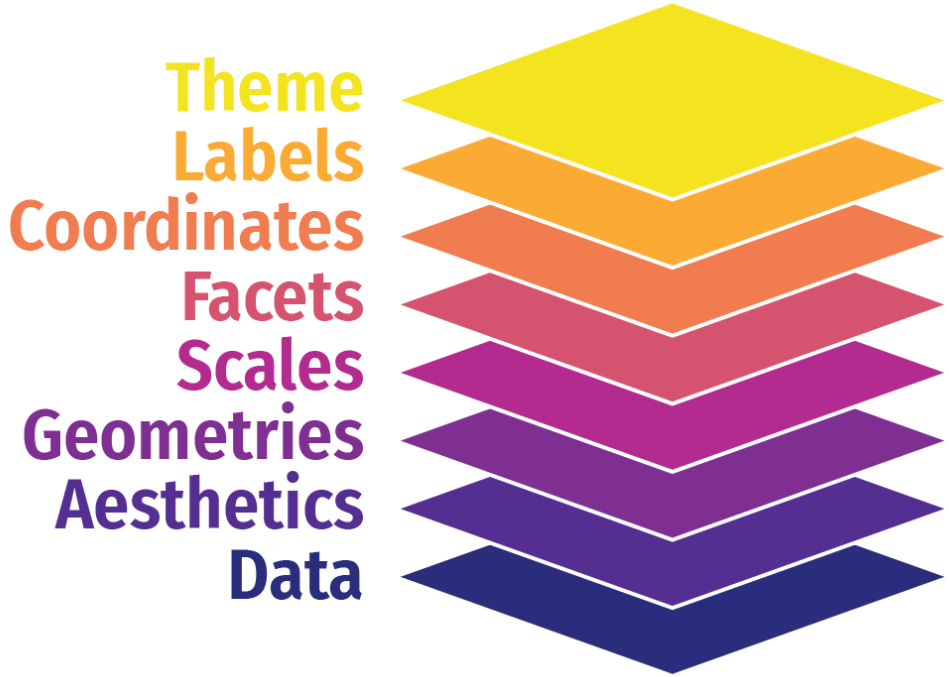


The Grammar of Graphics (gg) I

- Any graphic can be built from the same components:

    1. **Data** to be drawn from
    2. **Aesthetic mappings** from data to some visual marking
    3. **Geometric objects** on the plot
    4. **Scales** define the range of values
    5. **Coordinates** to organize location
    6. **Labels** describe the scale and markings
    7. **Facets** group into subplots
    8. **Themes** style the plot elements

- Not every plot needs *every* component, but all plots *must* have the first 3!

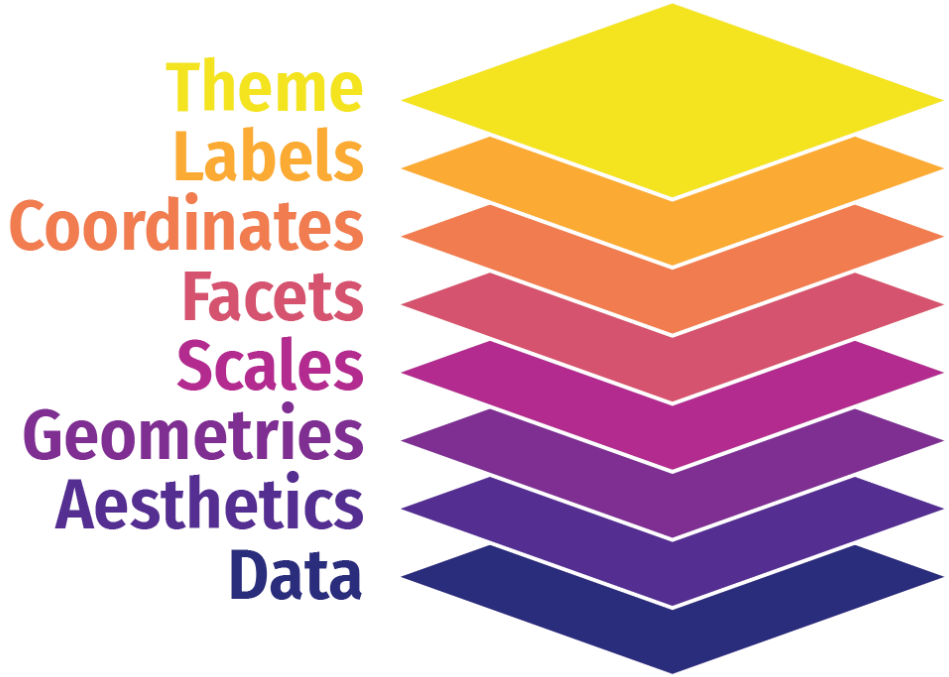


The Grammar of Graphics (gg) II

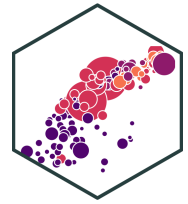
- Any graphic can be built from the same components:

    1. `data` to be drawn from
    2. `aes`**thetic mappings** from data to some visual marking
    3. `geom`**metric objects** on the plot
    4. `scale` define the range of values
    5. `coord`**inates** to organize location
    6. `labels` describe the scale and markings
    7. `facet` group into subplots
    8. `theme` style the plot elements

- Not every plot needs *every* component, but all plots *must* have the first 3!



The Grammar of Graphics (gg): All at Once

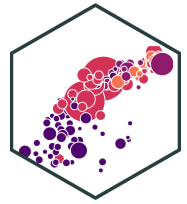


All in one command

- Produces plot output in viewer
- Does not save plot
    - Save with `Export` menu in viewer
- Adding layers requires whole code for new plot

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

The Grammar of Graphics (gg): As R Objects



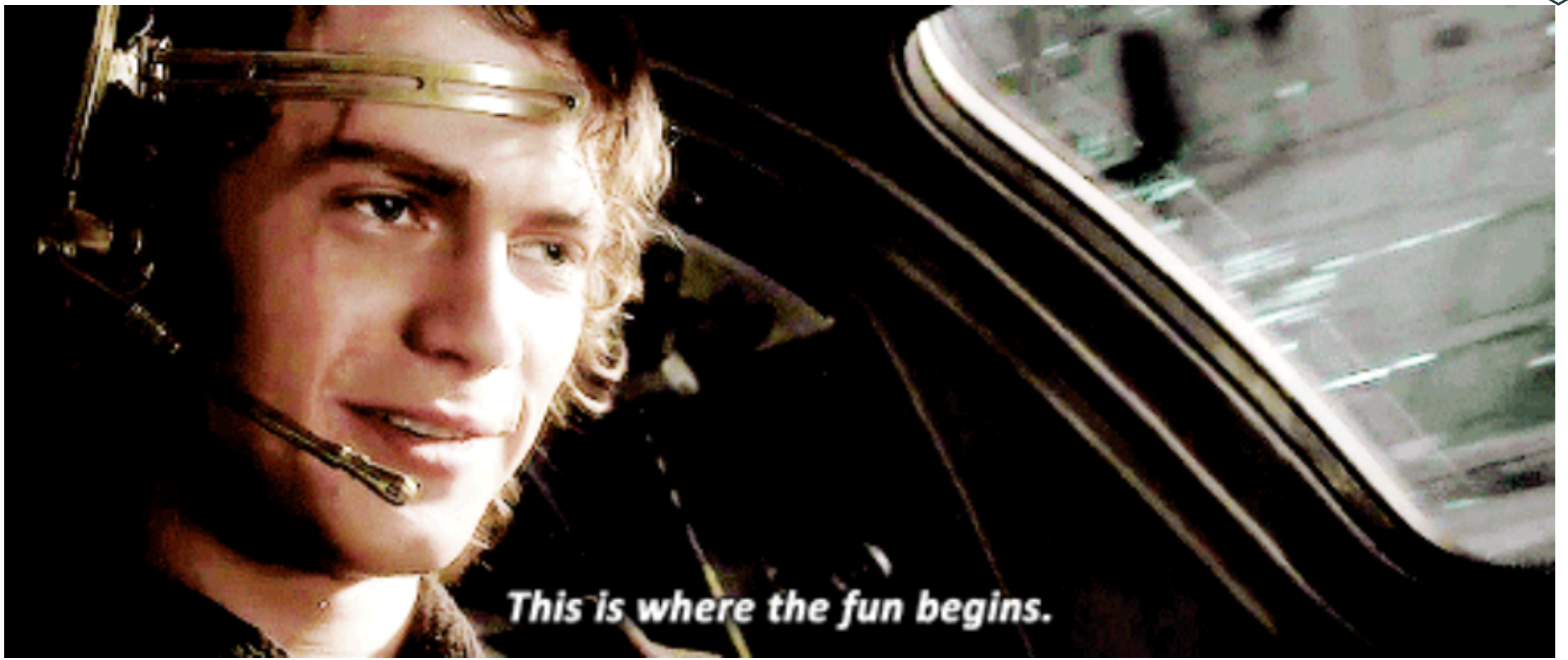
Saving as an object

- Saves your plot as an `R` object
- Does *not* show in viewer
    - Execute the name of your object to see it
- Can add layers by calling the original plot name

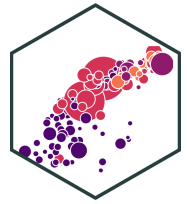
```
# make and save plot
p <- ggplot(data = mpg)+
  aes(x = displ,
      y = hwy)+
  geom_point()

p # view plot

# add a layer

p + geom_smooth() # shows new plot

p <- p + geom_smooth() # saves and overwrites p
p2 <- p + geom_smooth() # saves as different ob
```

Plot Layers

The Grammar of Graphics



The Grammar of Graphics (gg): Data



Data

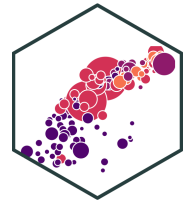
```
ggplot(data = mpg)
```

Data is the source of our data. As part of the `tidyverse`, `ggplot2` requires data to be **"tidy"**[1]:

1. Each variable forms a column
2. Each observation forms a row
3. Each observational unit forms a table

[1] Data "tidyness" is the core element of all `tidyverse` packages. Much more on all of this next class.

The Grammar of Graphics (gg): Adding Layers



Data

- Add a layer with `+` at the end of a line (never at the beginning!)
- Style recommendation: start a new line after each `+` to improve legibility!
- We will build a plot layer-by-layer

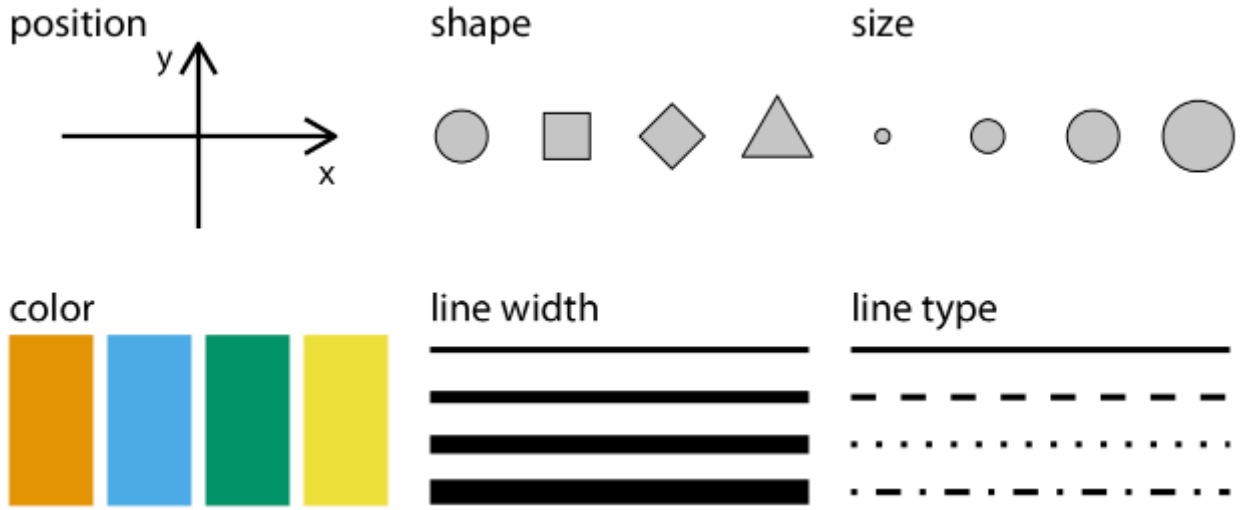
The Grammar of Graphics (gg): Aesthetics I

Data

Aesthetics

```
+ aes()
```

Aesthetics map data to visual elements or parameters



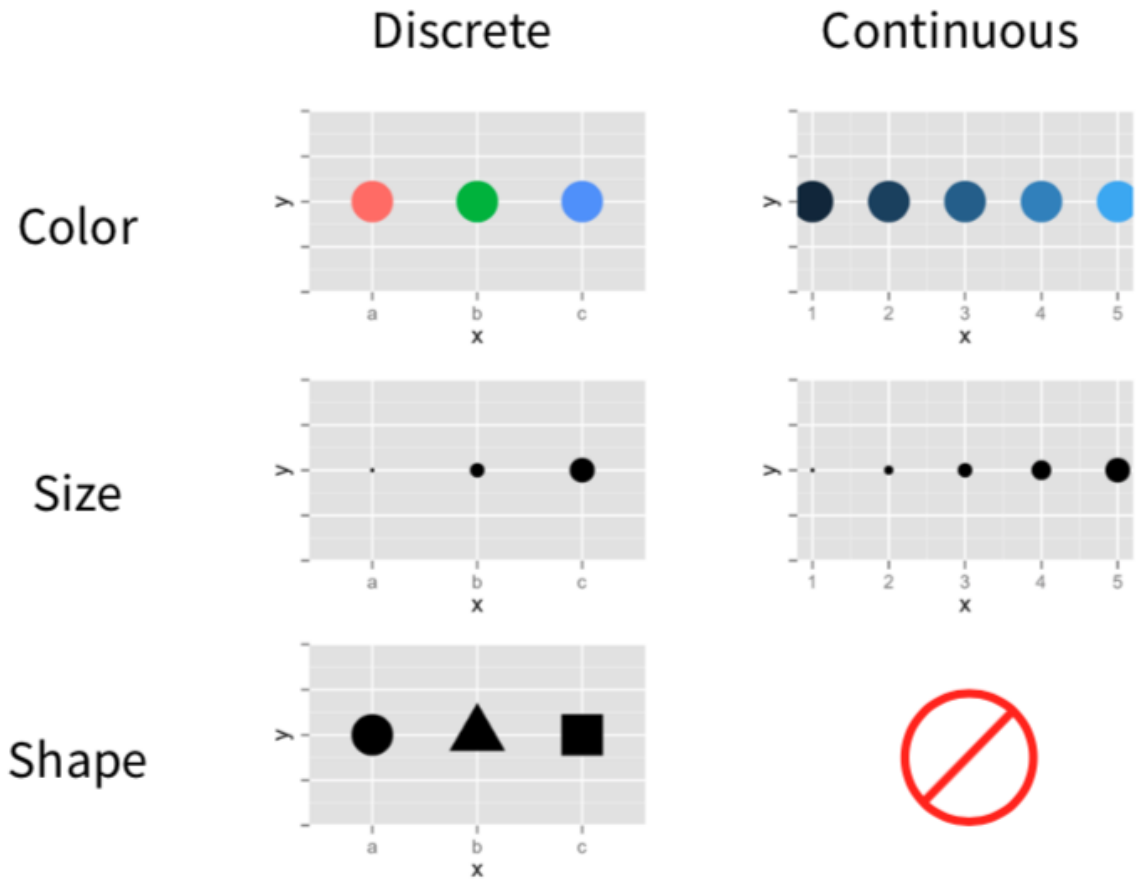
The Grammar of Graphics (gg): Aesthetics II

Data

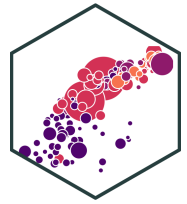
Aesthetics

```
+ aes()
```

Aesthetics map data to visual elements or parameters



The Grammar of Graphics (gg): Aesthetics III



Data

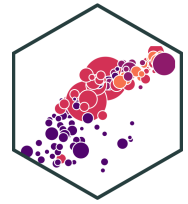
Aesthetics

`+ aes()`

Aesthetics map data to visual elements or parameters

- `displ`
- `hwy`
- `class`

The Grammar of Graphics (gg): Aesthetics III



Data

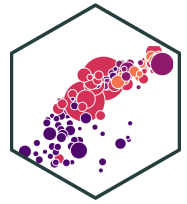
Aesthetics

`+ aes()`

Aesthetics map data to visual elements or parameters

- `displ` → **x**
- `hwy` → **y**
- `class` → *shape*, *size*, **color**, etc.

The Grammar of Graphics (gg): Aesthetics IV

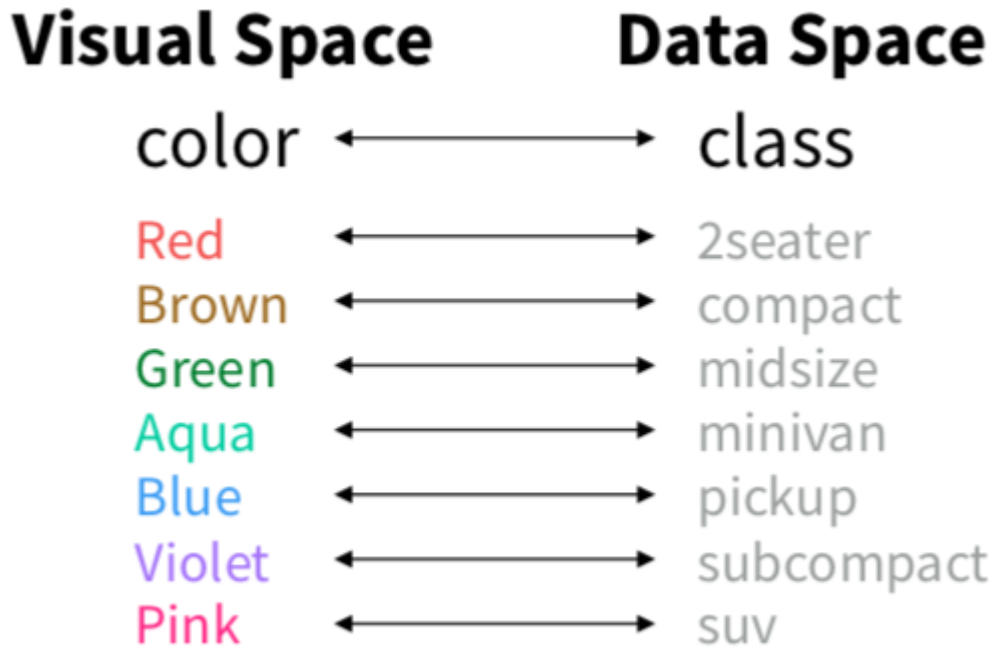


Data

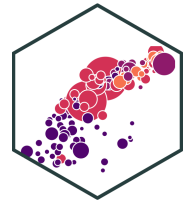
Aesthetics

```
+ aes()
```

Aesthetics map data to visual elements or parameters



The Grammar of Graphics (gg): Aesthetics IV



Data

Aesthetics

```
+ aes()
```

Aesthetics map data to visual elements or parameters

```
aes(x = displ,
    y = hwy,
    color = class)
```

The Grammar of Graphics (gg): Geoms I

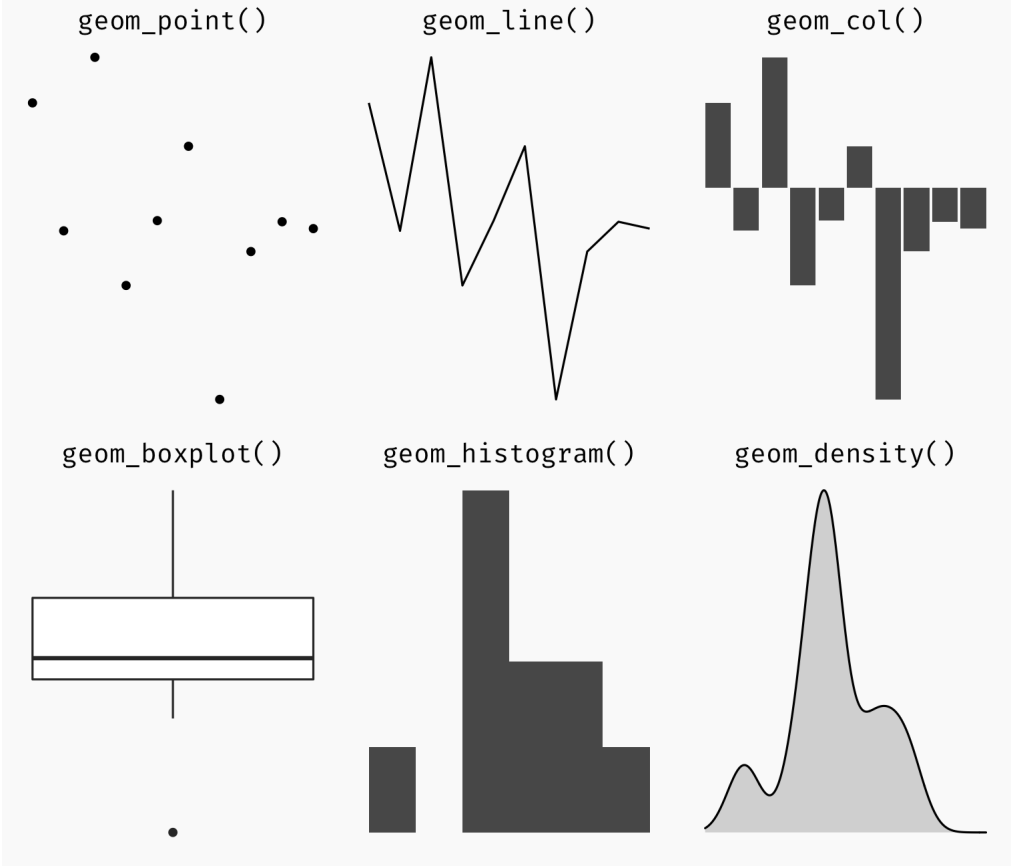
Data

Aesthetics

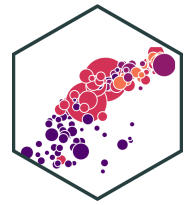
Geoms

```
+ geom_*()
```

Geometric objects displayed on the plot



The Grammar of Graphics (gg): Geoms II



Data

Aesthetics

Geoms

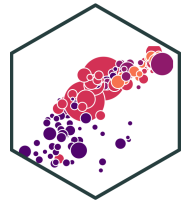
`+ geom_*()`

Geometric objects displayed on the plot

- What `geom`s you should use depends on what you want to show:

Type	`geom`
Point	`geom_point()`
Line	`geom_line()`, `geom_path()`
Bar	`geom_bar()`, `geom_col()`
Histogram	`geom_histogram()`
Regression	`geom_smooth()`

The Grammar of Graphics (gg): Geoms III



Data

Aesthetics

Geoms

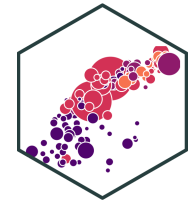
```
+ geom_*()
```

Geometric objects displayed on the plot

```
##  [1] "geom_abline"     "geom_area"       "geom_bar"
##  [5] "geom_blank"      "geom_boxplot"    "geom_col"
##  [9] "geom_count"      "geom_crossbar"   "geom_curve"
## [13] "geom_density_2d" "geom_density2d"  "geom_dotplot"
## [17] "geom_errorbarh"  "geom_freqpoly"   "geom_hex"
## [21] "geom_hline"      "geom_jitter"     "geom_label"
## [25] "geom_linerange"  "geom_map"        "geom_path"
## [29] "geom_pointrange" "geom_polygon"    "geom_qq"
## [33] "geom_quantile"   "geom_raster"     "geom_rect"
## [37] "geom_rug"        "geom_segment"    "geom_sf"
## [41] "geom_sf_text"    "geom_smooth"     "geom_spoke"
## [45] "geom_text"       "geom_tile"       "geom_violin"
```

See http://ggplot2.tidyverse.org/reference for many more options

The Grammar of Graphics (gg): Geoms IV



Data

Aesthetics

Geoms

`+ geom_*()`

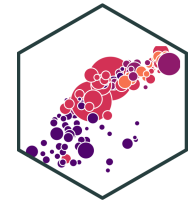
Geometric objects displayed on the plot

Or just start typing `geom_` in R Studio!

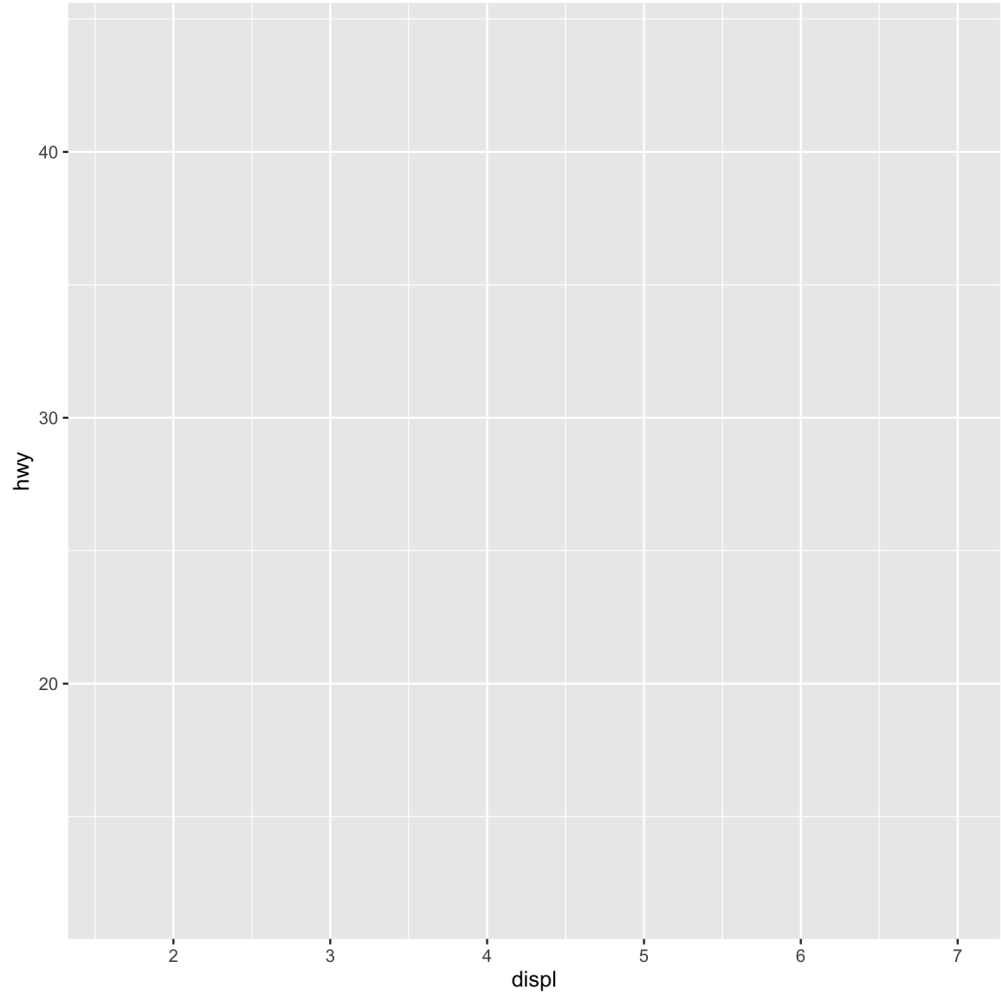
Let's Make a Plot!

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

Let's Make a Plot!

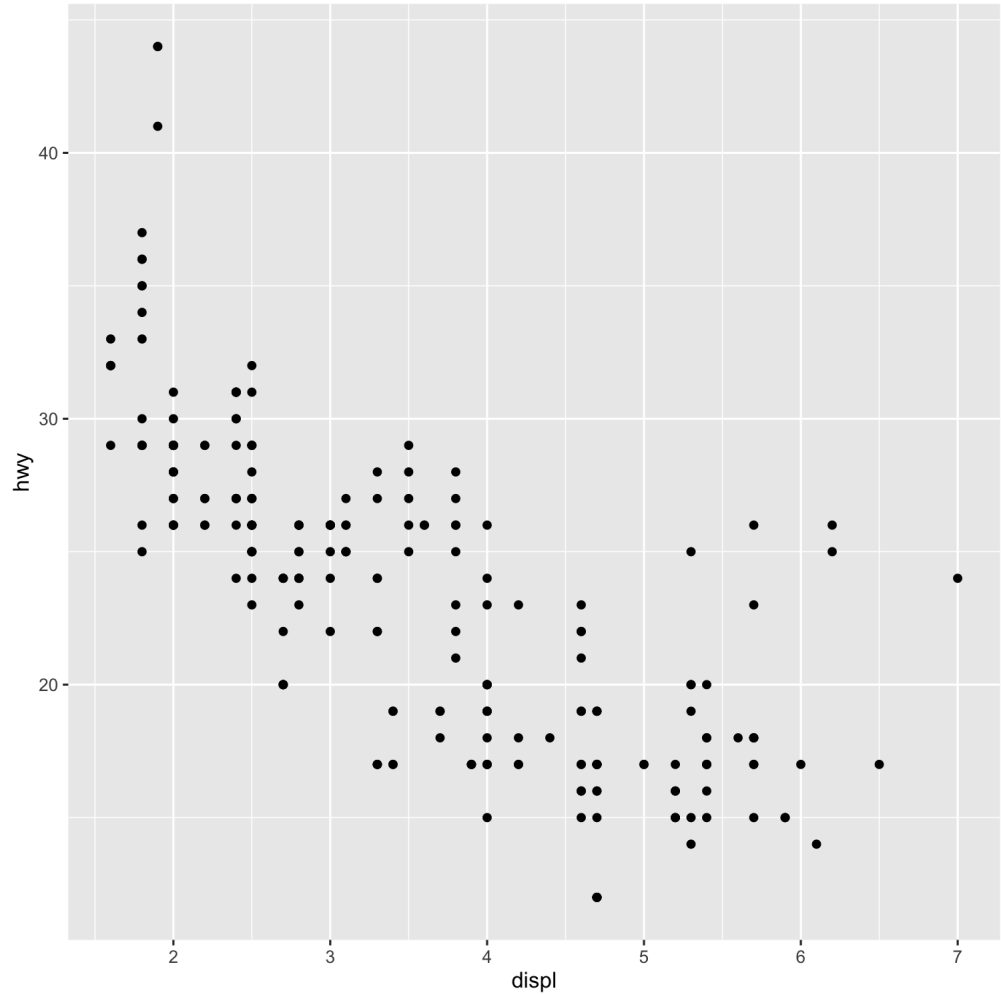


```
ggplot(data = mpg)+
  aes(x = displ,
      y = hwy)
```

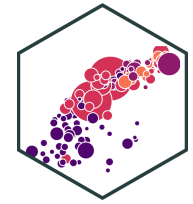


Let's Make a Plot!

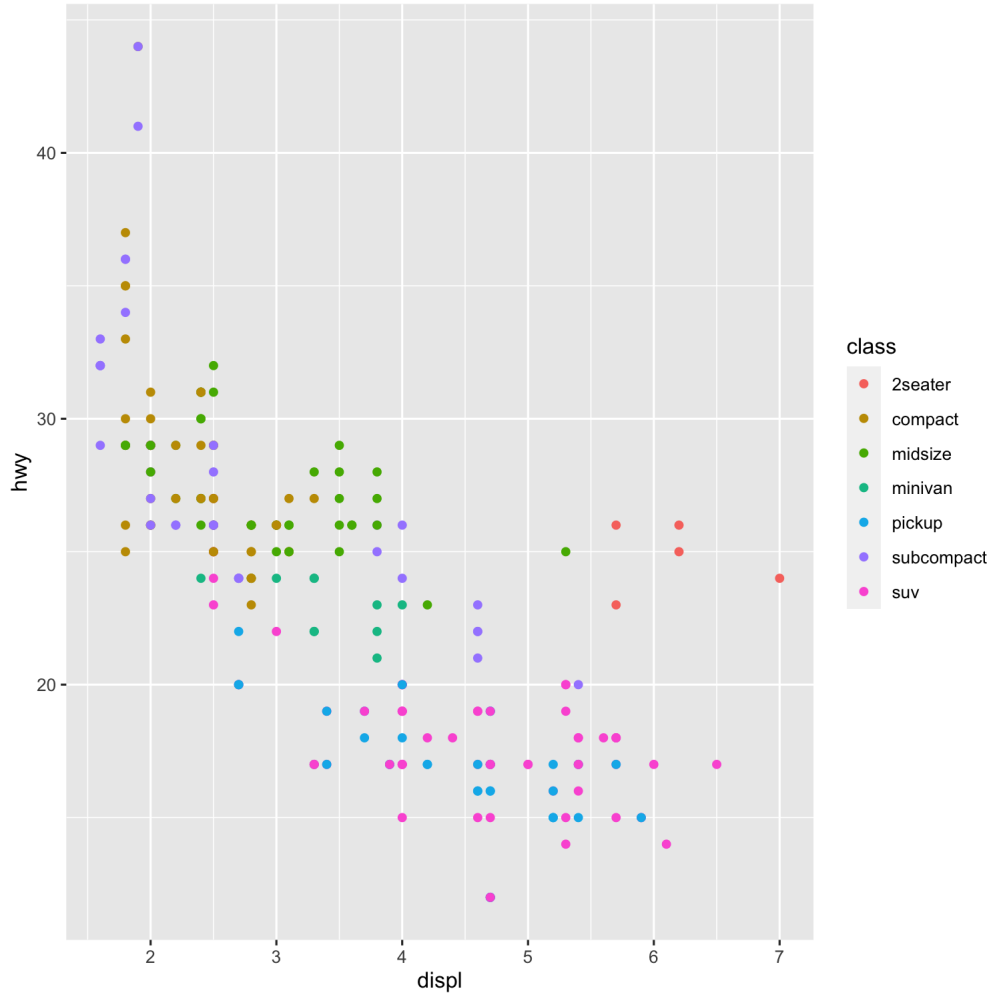
```
ggplot(data = mpg)+
  aes(x = displ,
      y = hwy)+
  geom_point()
```



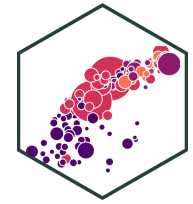
Let's Make a Plot!



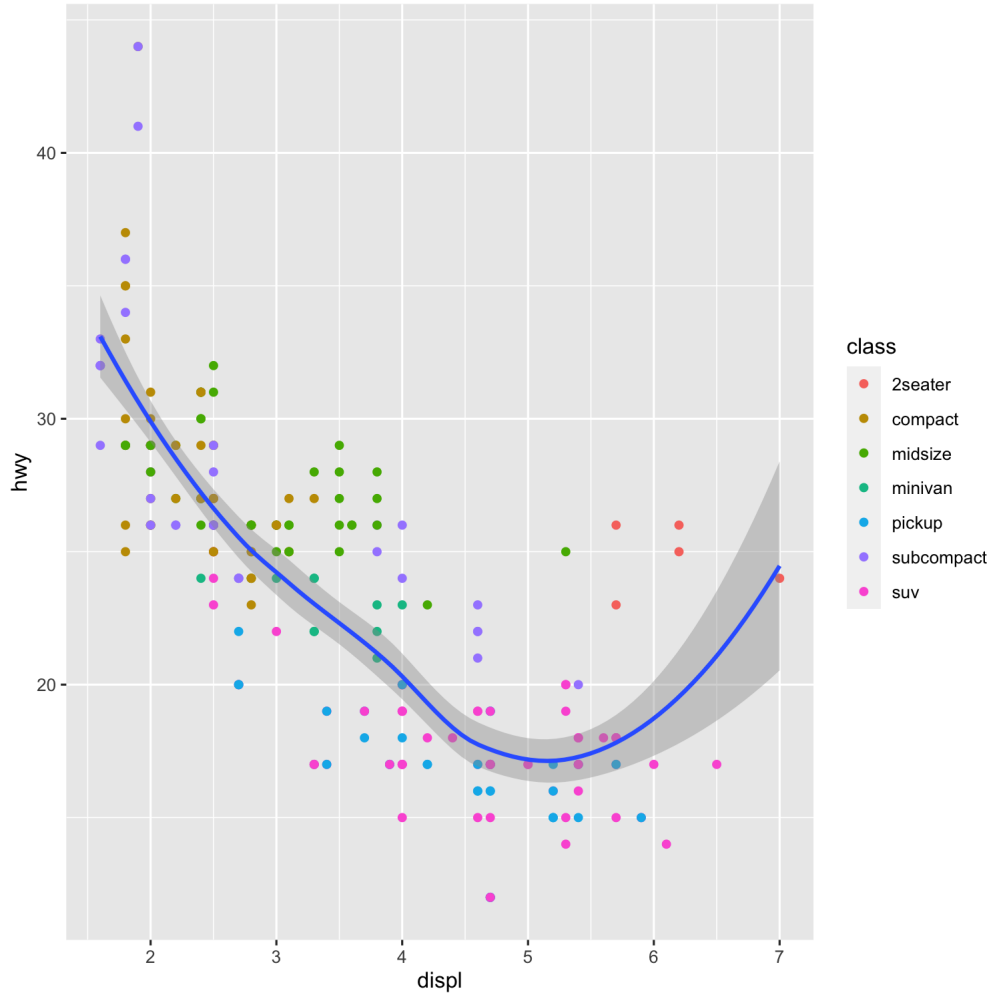
```
ggplot(data = mpg)+
  aes(x = displ,
      y = hwy)+
  geom_point(aes(color = class))
```



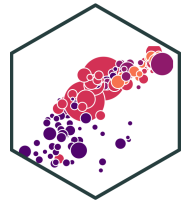
Let's Make a Plot!



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



More Geoms



Data

Aesthetics

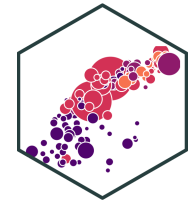
Geoms

`+ geom_*()`

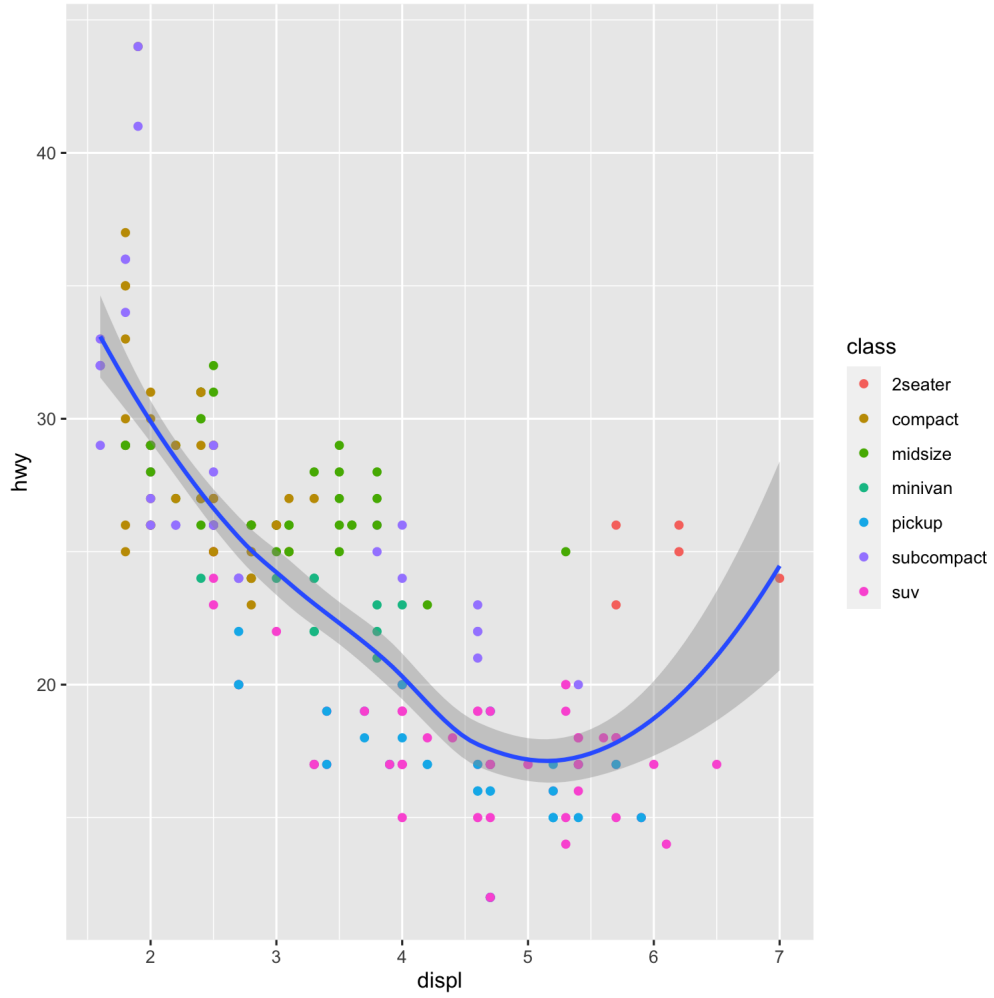
`geom_*(aes, data, stat, position)`

- `data`: geoms can have their own data
  - has to map onto global coordinates
- `aes`: geoms can have their own aesthetics
  - inherits global aesthetics by default
  - different geoms have different available aesthetics

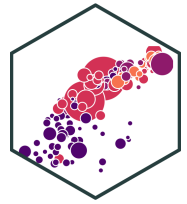
Change Our Plot



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



More Geoms II



Data

Aesthetics

Geoms

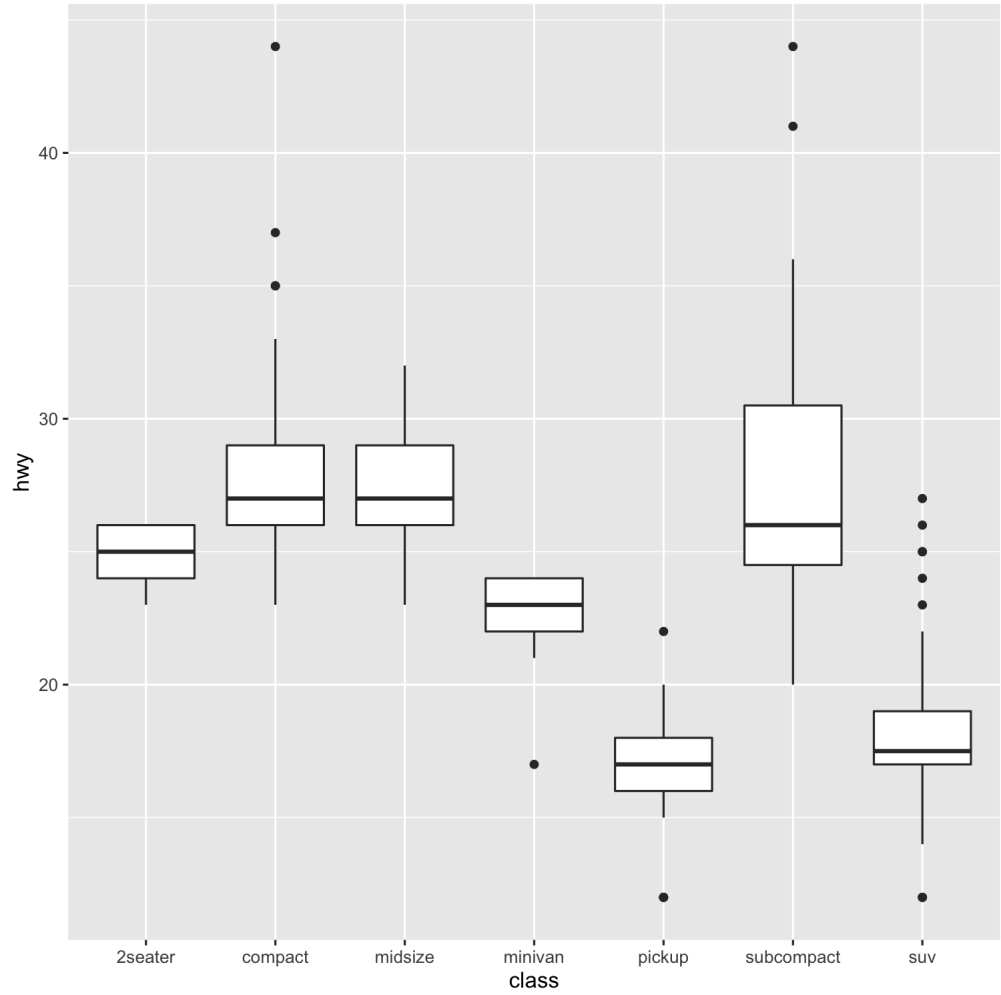
`+ geom_*()`

`geom_*(aes, data, stat, position)`

- `stat`: some geoms statistically transform data
  - `geom_histogram()` uses `stat_bin()` to group observations into bins
- `position`: some adjust location of objects
  - `dodge`, `stack`, `jitter`

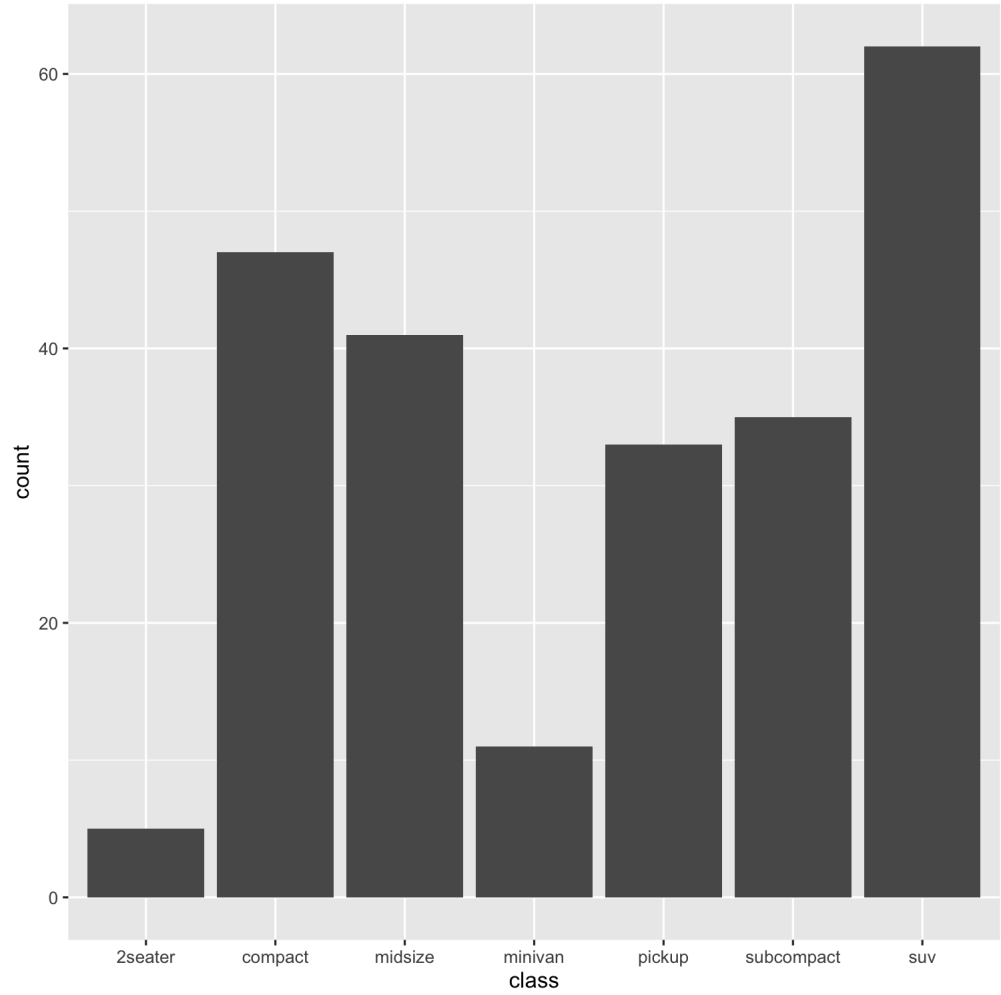
Let's Change Our Plot

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

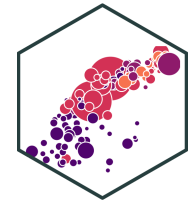


Let's Change Our Plot

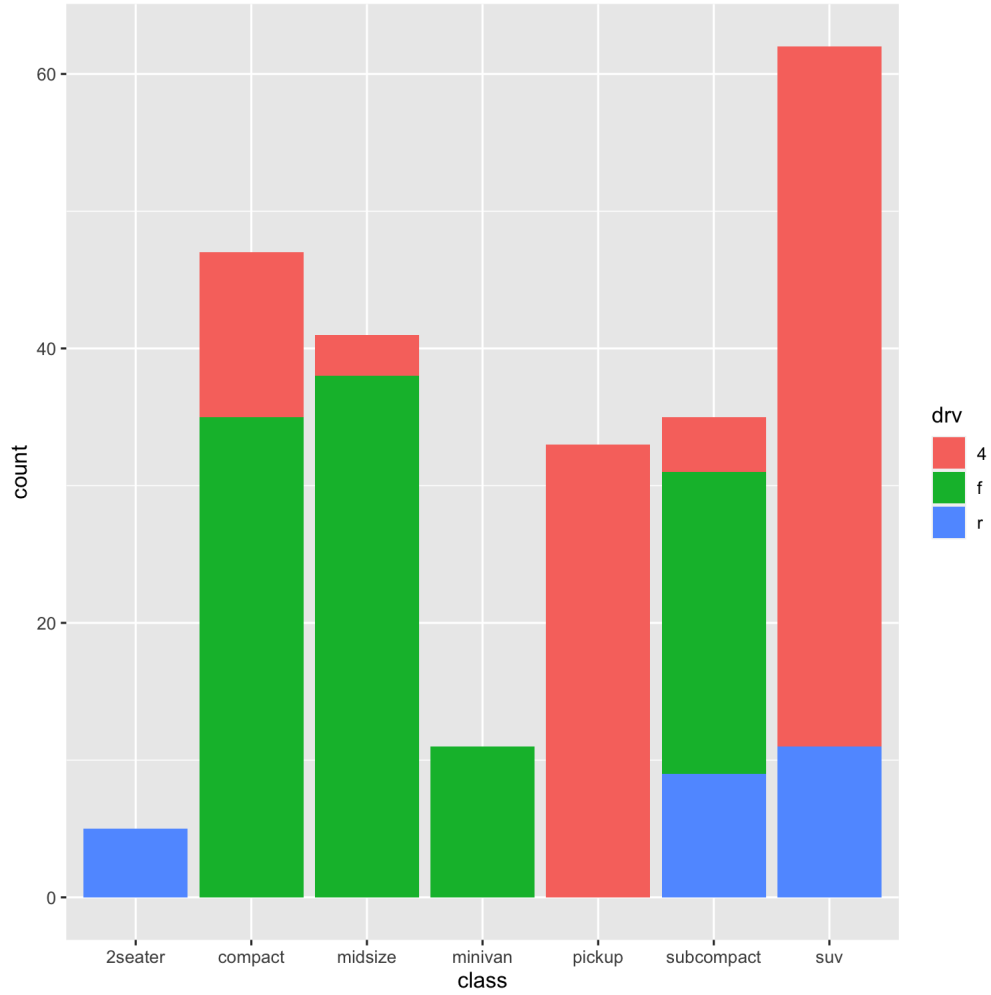
```
ggplot(data = mpg)+
  aes(x = class)+
  geom_bar()
```



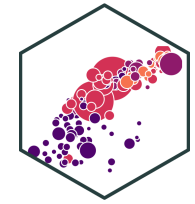
Let's Change Our Plot



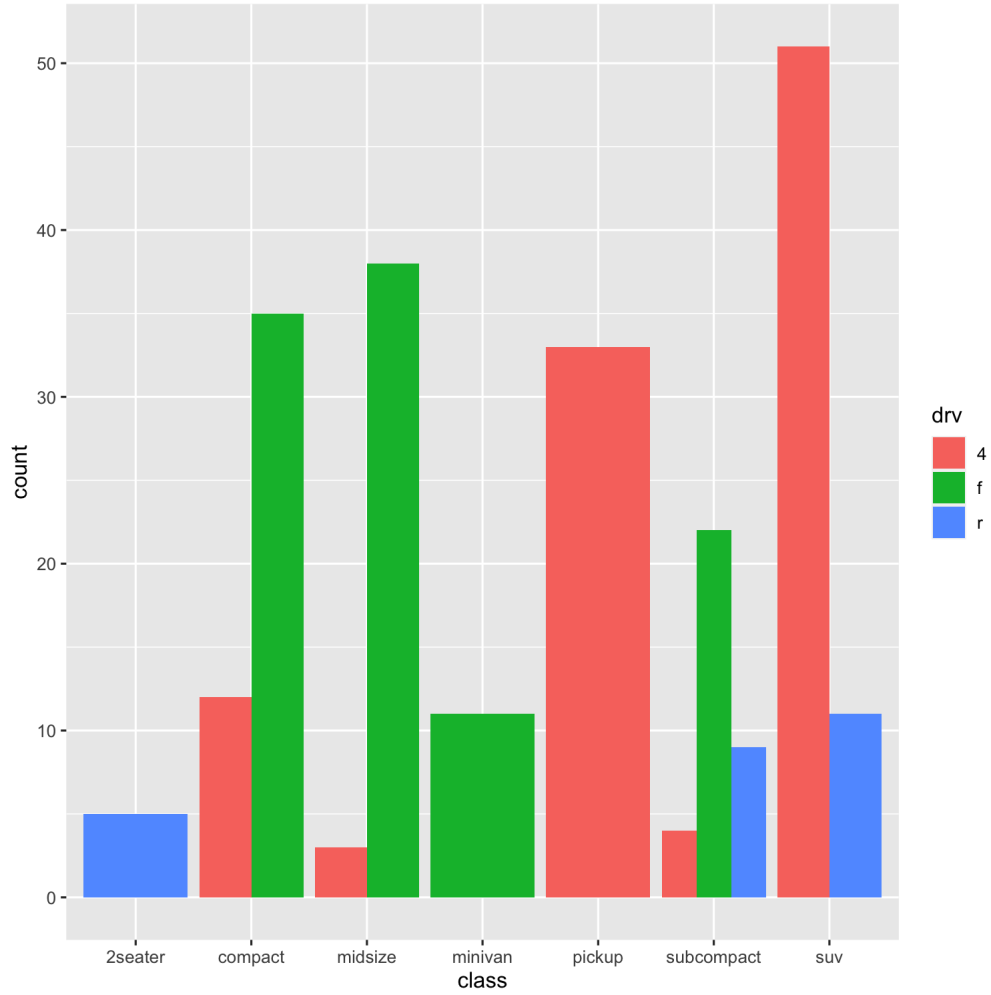
```
ggplot(data = mpg)+
  aes(x = class,
      fill = drv)+
  geom_bar()
```



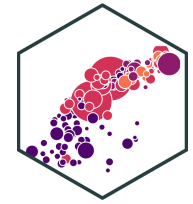
Let's Change Our Plot



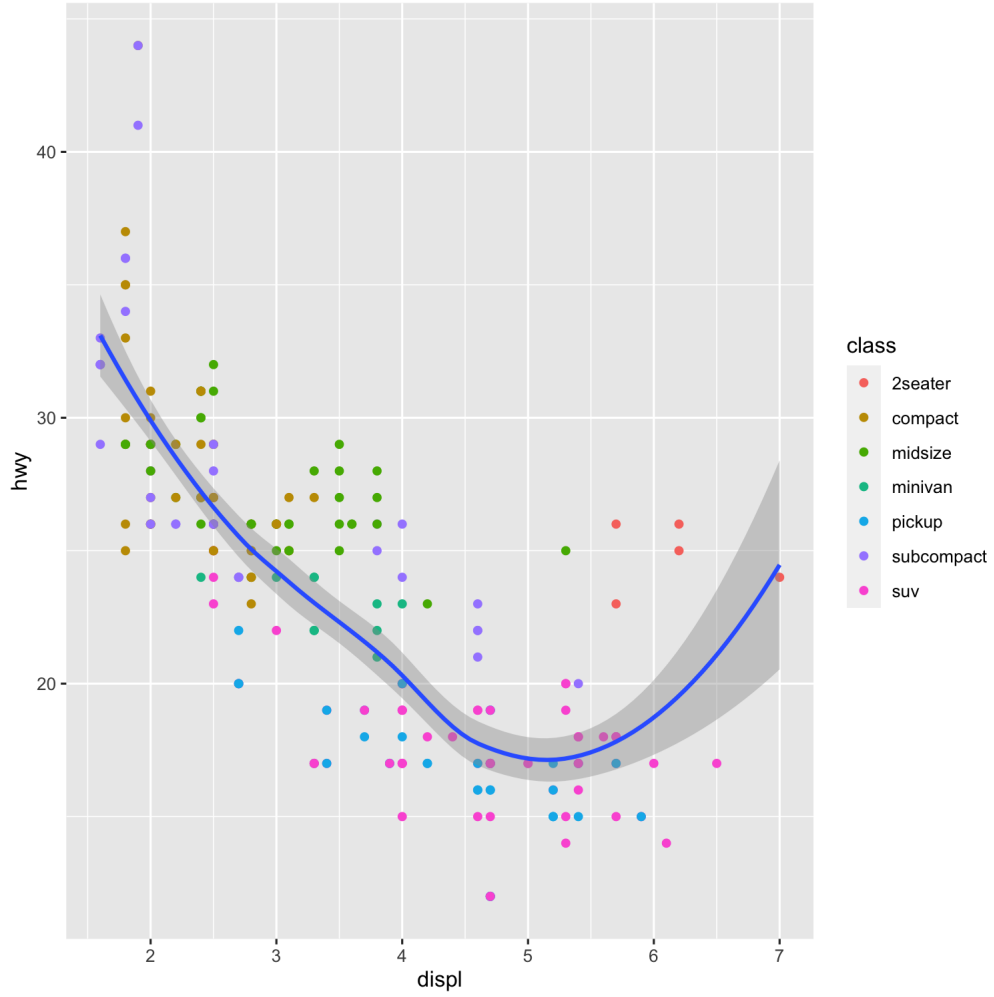
```
ggplot(data = mpg)+
  aes(x = class,
      fill = drv)+
  geom_bar(position = "dodge")
```



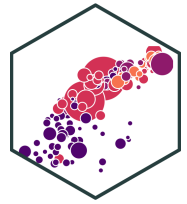
Back to the Original (and saving it)



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

p # show plot
```



The Grammar of Graphics (gg): Facets I



Data

Aesthetics

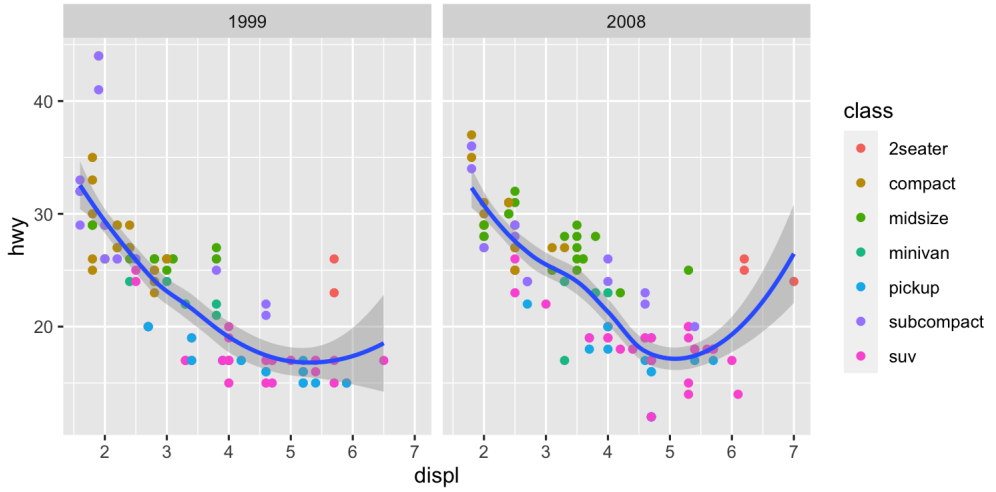
Geoms

Facets

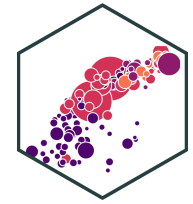
```
+ facet_wrap()
```

```
+ facet_grid()
```

```
p + facet_wrap(~year)
```



The Grammar of Graphics (gg): Facets II



Data

Aesthetics

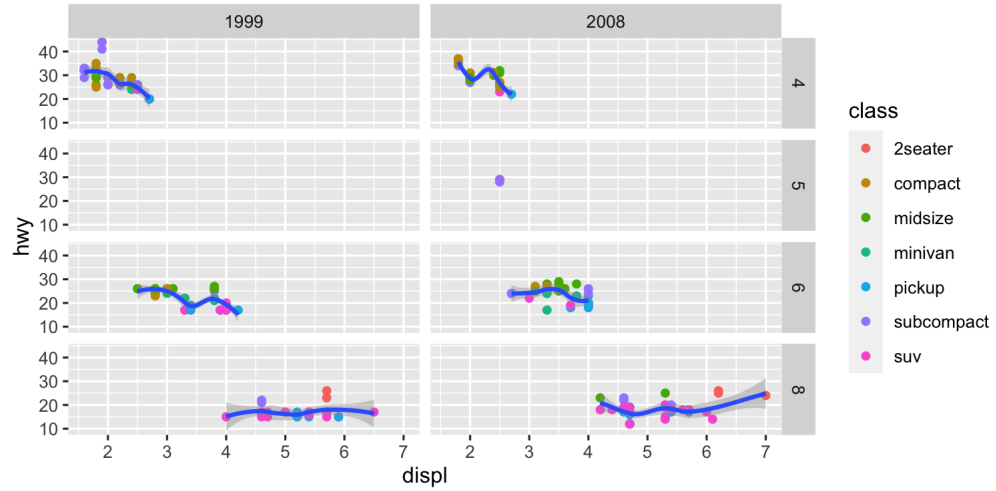
Geoms

Facets

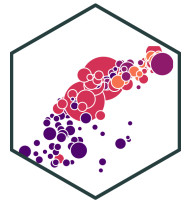
```
+ facet_wrap()
```

```
+ facet_grid()
```

```
p + facet_grid(cyl~year)
```



The Grammar of Graphics (gg): Labels



Data

Aesthetics

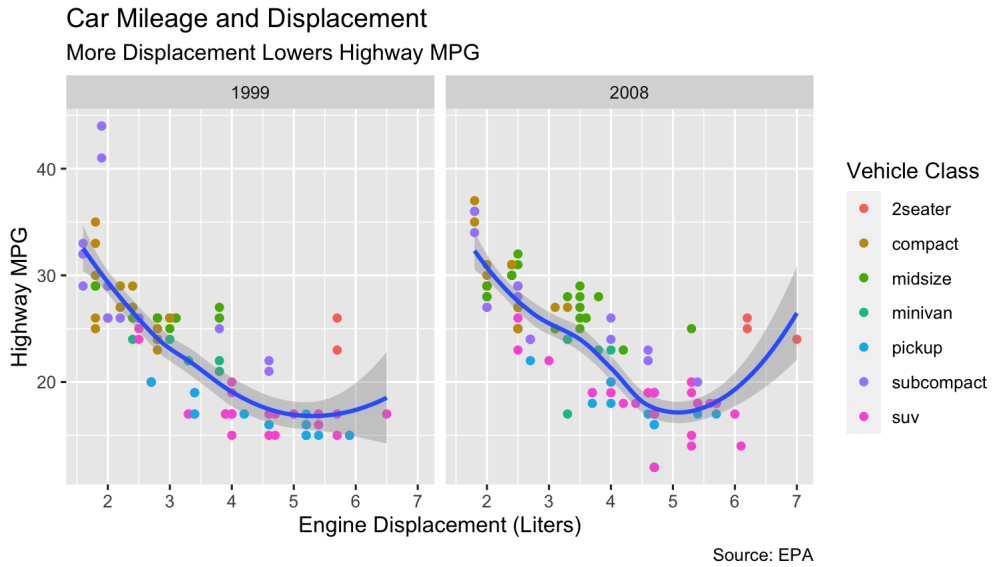
Geoms

Facets

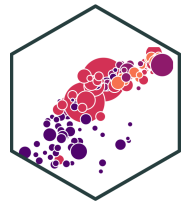
Labels

```
+ labs()
```

```
p + facet_wrap(~year)+
  labs(x = "Engine Displacement (Liters)",
       y = "Highway MPG",
       title = "Car Mileage and Displacement",
       subtitle = "More Displacement Lowers Highway MPG",
       caption = "Source: EPA",
       color = "Vehicle Class")
```



The Grammar of Graphics (gg): Scales



Data

Aesthetics

Geoms

Facets

Labels

Scales

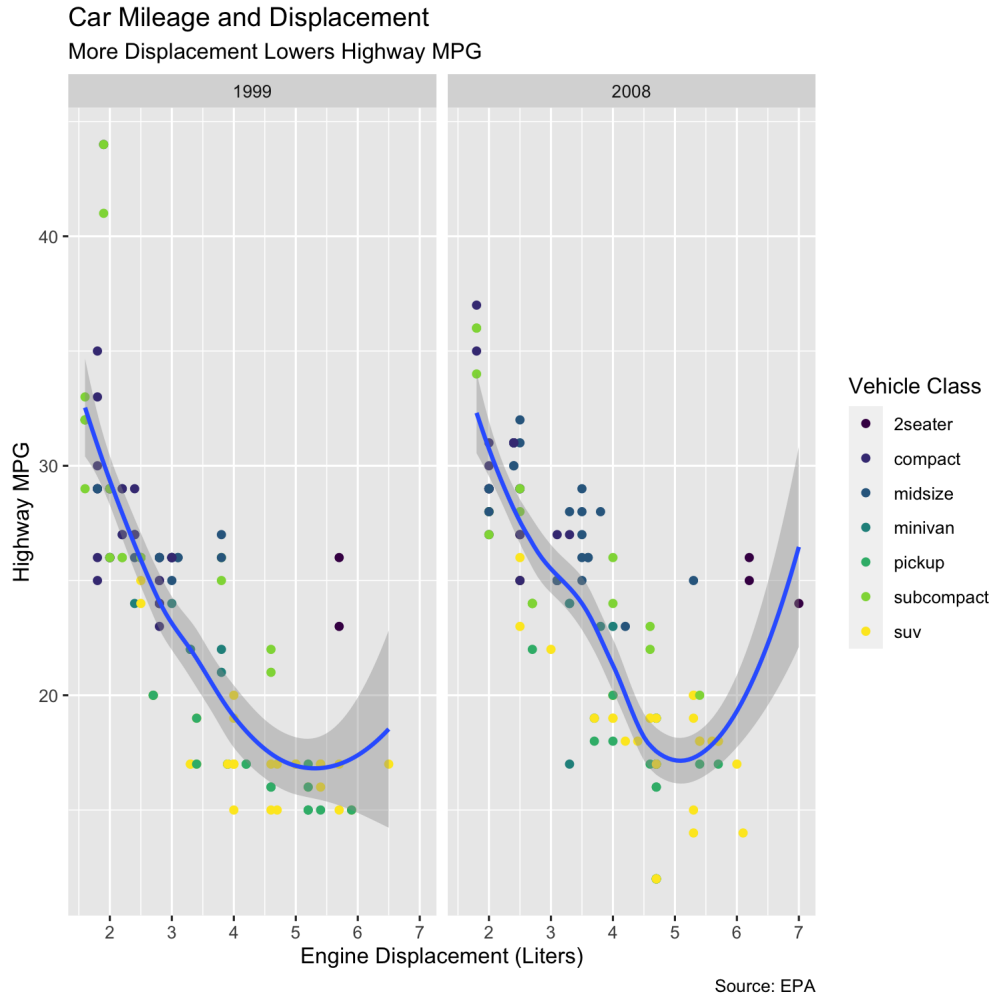
`+ scale_*_*()`

`scale` + `_` + `<aes>` + `_` + `<type>` + `()`

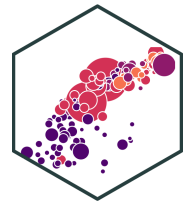
- `<aes>`: parameter you want to adjust
- `<type`: type of parameter
- I want to change my discrete x-axis: `scale_x_discrete()`
- I want to change my continuous y-axis: `scale_y_continuous()`
- I want to rescale x-axis to log: `scale_x_log10()`
- I want to use a different color palette: `scale_fill_discrete()`,

The Grammar of Graphics (gg): Scales

```
ggplot(data = mpg)+
  aes(x = displ,
      y = hwy)+
  geom_point(aes(color = class))+
  geom_smooth()+
  facet_wrap(~year)+
  labs(x = "Engine Displacement (Liter
       y = "Highway MPG",
       title = "Car Mileage and Displa
       subtitle = "More Displacement L
       caption = "Source: EPA",
       color = "Vehicle Class")+
  scale_color_viridis_d()
```



The Grammar of Graphics (gg): Themes



Data

Aesthetics

Geoms

Facets

Labels

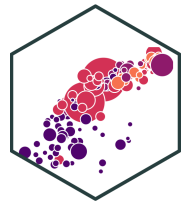
Scales

Theme

Theme changes appearance of plot decorations (things not mapped to data)

- Some themes that come with `ggplot2`:
- `+ theme_bw()`
- `+ theme_dark()`
- `+ theme_gray()`
- `+ theme_minimal()`
- `+ theme_light()`
- `+ theme_classic()`

The Grammar of Graphics (gg): Themes



Data

Aesthetics

Geoms

Facets

Labels

Scales

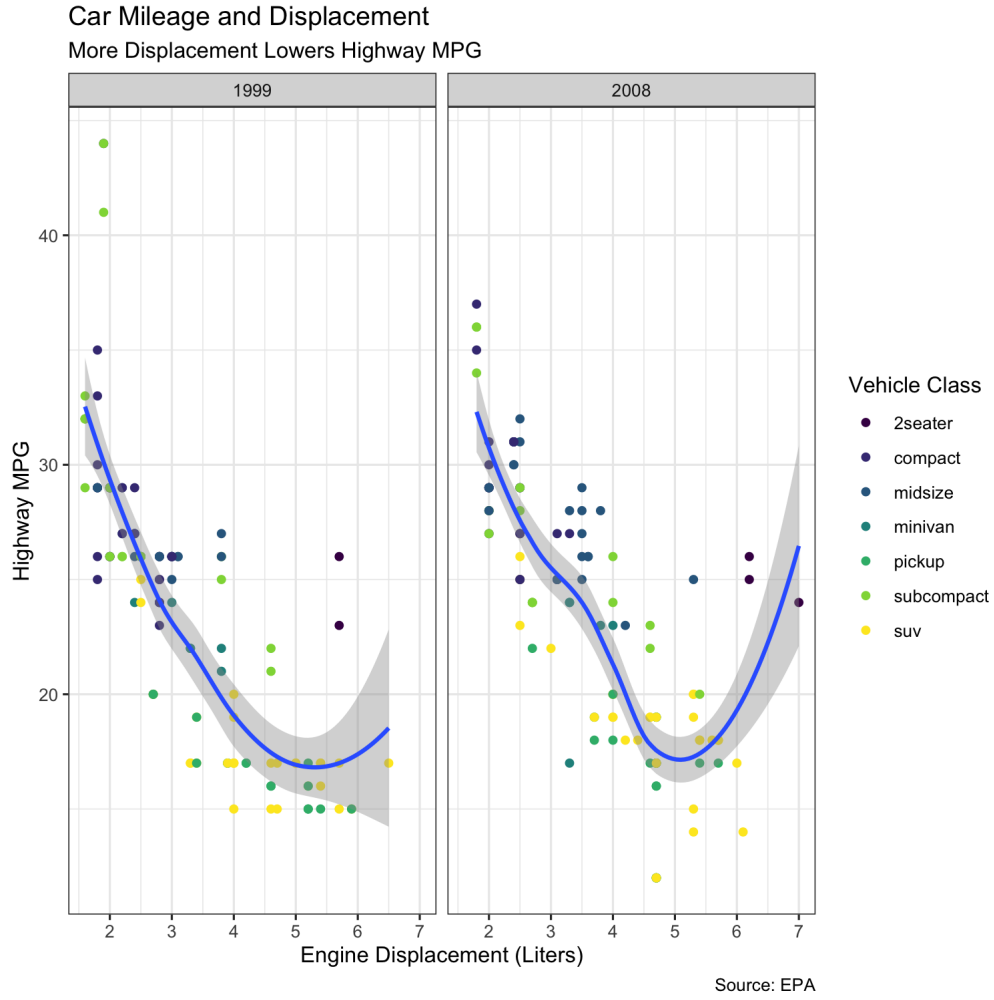
Theme

Theme changes appearance of plot decorations (things not mapped to data)

- Many parameters we could change
- Global options: `line`, `rect`, `text`, `title`
- `axis`: x-, y-, or other axis title, ticks, lines
- `legend`: plot legends for fill or color
- `panel`: actual plot area
- `plot`: whole image
- `strip`: facet labels

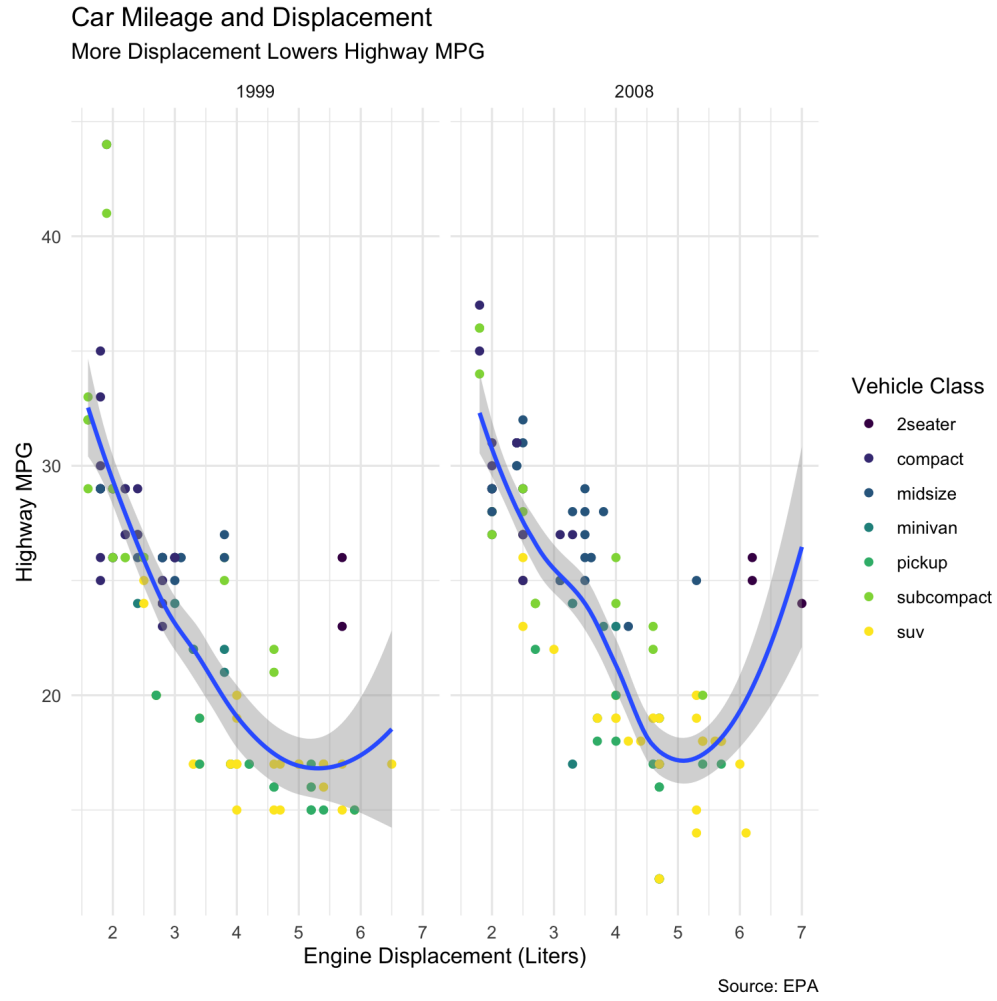
The Grammar of Graphics (gg): Themes

```
ggplot(data = mpg)+
  aes(x = displ,
      y = hwy)+
  geom_point(aes(color = class))+
  geom_smooth()+
  facet_wrap(~year)+
  labs(x = "Engine Displacement (Liter
       y = "Highway MPG",
       title = "Car Mileage and Displa
       subtitle = "More Displacement L
       caption = "Source: EPA",
       color = "Vehicle Class")+
  scale_color_viridis_d()+
  theme_bw()
```



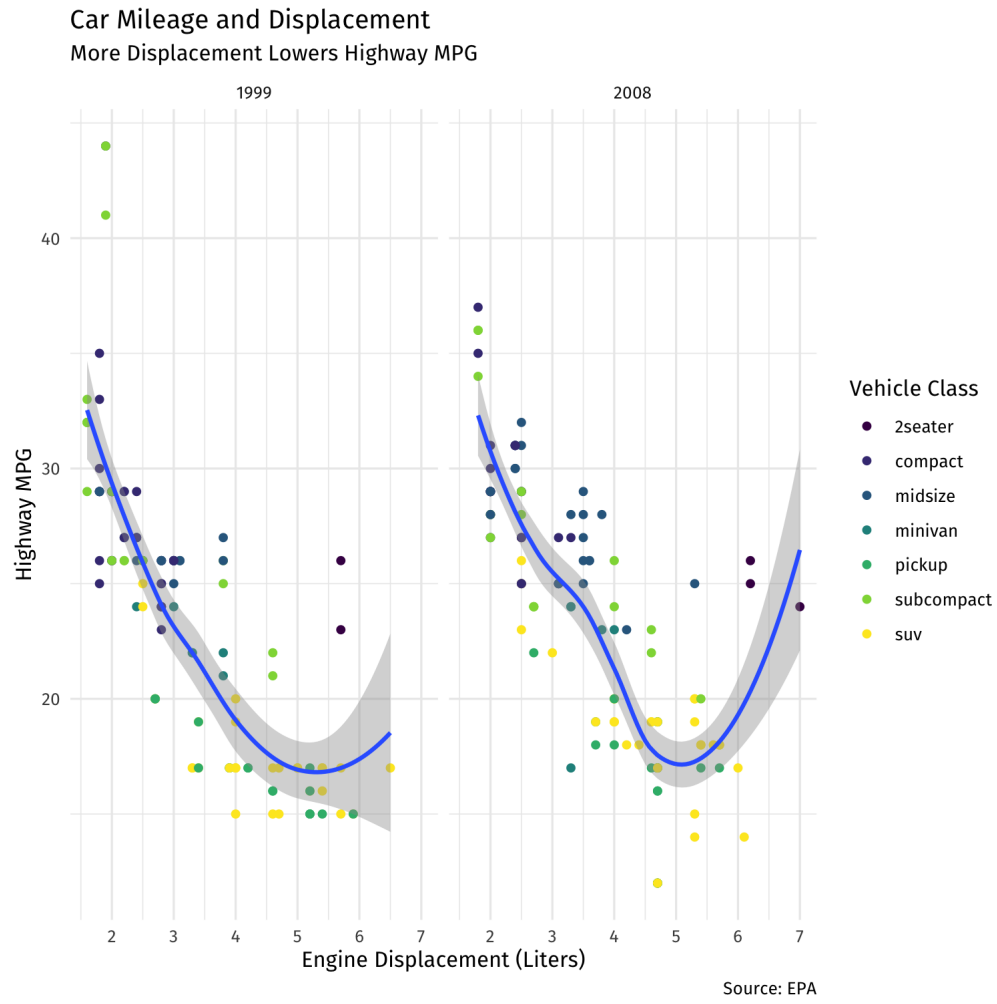
The Grammar of Graphics (gg): Themes II

```
ggplot(data = mpg)+
  aes(x = displ,
      y = hwy)+
  geom_point(aes(color = class))+
  geom_smooth()+
  facet_wrap(~year)+
  labs(x = "Engine Displacement (Liter
       y = "Highway MPG",
       title = "Car Mileage and Displa
       subtitle = "More Displacement L
       caption = "Source: EPA",
       color = "Vehicle Class")+
  scale_color_viridis_d()+
  theme_minimal()
```

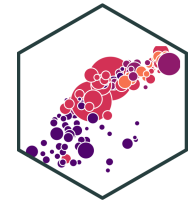


The Grammar of Graphics (gg): Themes III

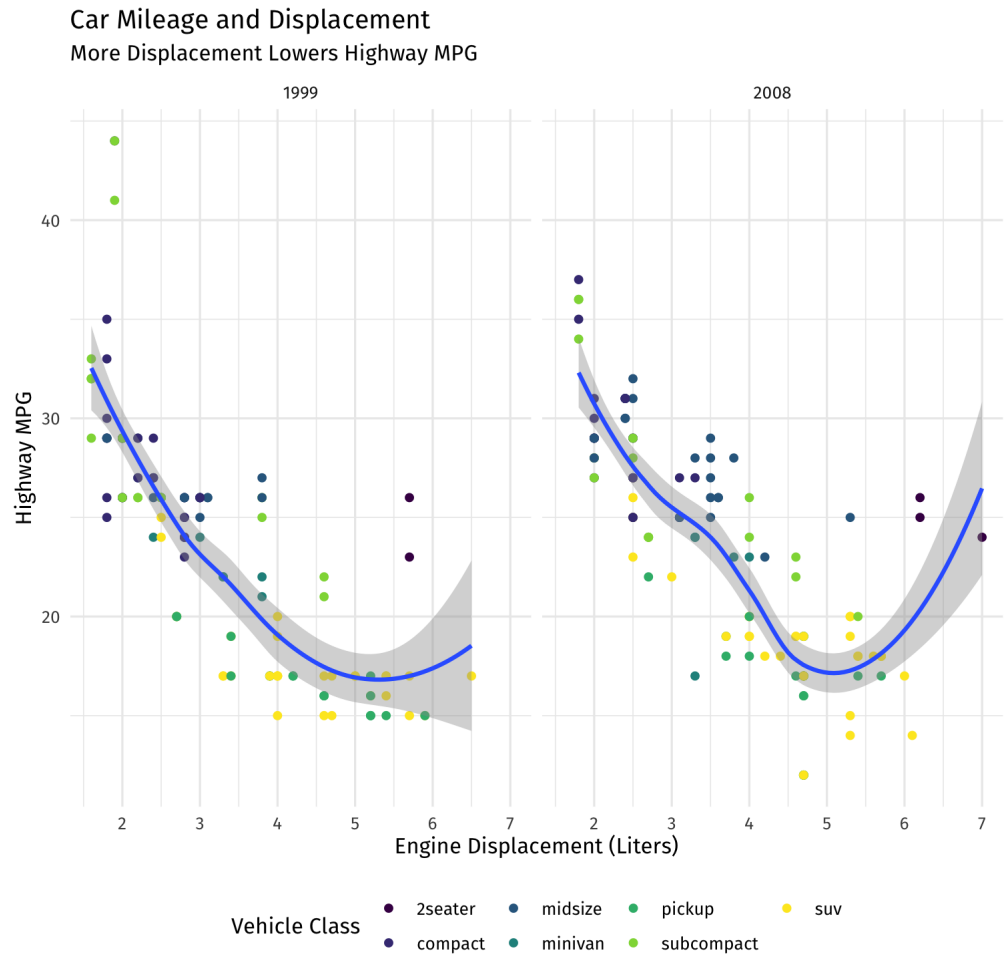
```
ggplot(data = mpg)+
  aes(x = displ,
      y = hwy)+
  geom_point(aes(color = class))+
  geom_smooth()+
  facet_wrap(~year)+
  labs(x = "Engine Displacement (Liter
       y = "Highway MPG",
       title = "Car Mileage and Displa
       subtitle = "More Displacement L
       caption = "Source: EPA",
       color = "Vehicle Class")+
  scale_color_viridis_d()+
  theme_minimal()+
  theme(text = element_text(family = "
```



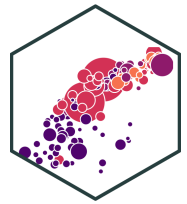
The Grammar of Graphics (gg): Themes III



```
ggplot(data = mpg)+
  aes(x = displ,
      y = hwy)+
  geom_point(aes(color = class))+
  geom_smooth()+
  facet_wrap(~year)+
  labs(x = "Engine Displacement (Liter
       y = "Highway MPG",
       title = "Car Mileage and Displa
       subtitle = "More Displacement L
       caption = "Source: EPA",
       color = "Vehicle Class")+
  scale_color_viridis_d()+
  theme_minimal()+
  theme(text = element_text(family = "
        legend.position="bottom")
```



The Grammar of Graphics (gg): Themes (ggthemes)



Data

Aesthetics

Geoms

Facets

Labels

Scales

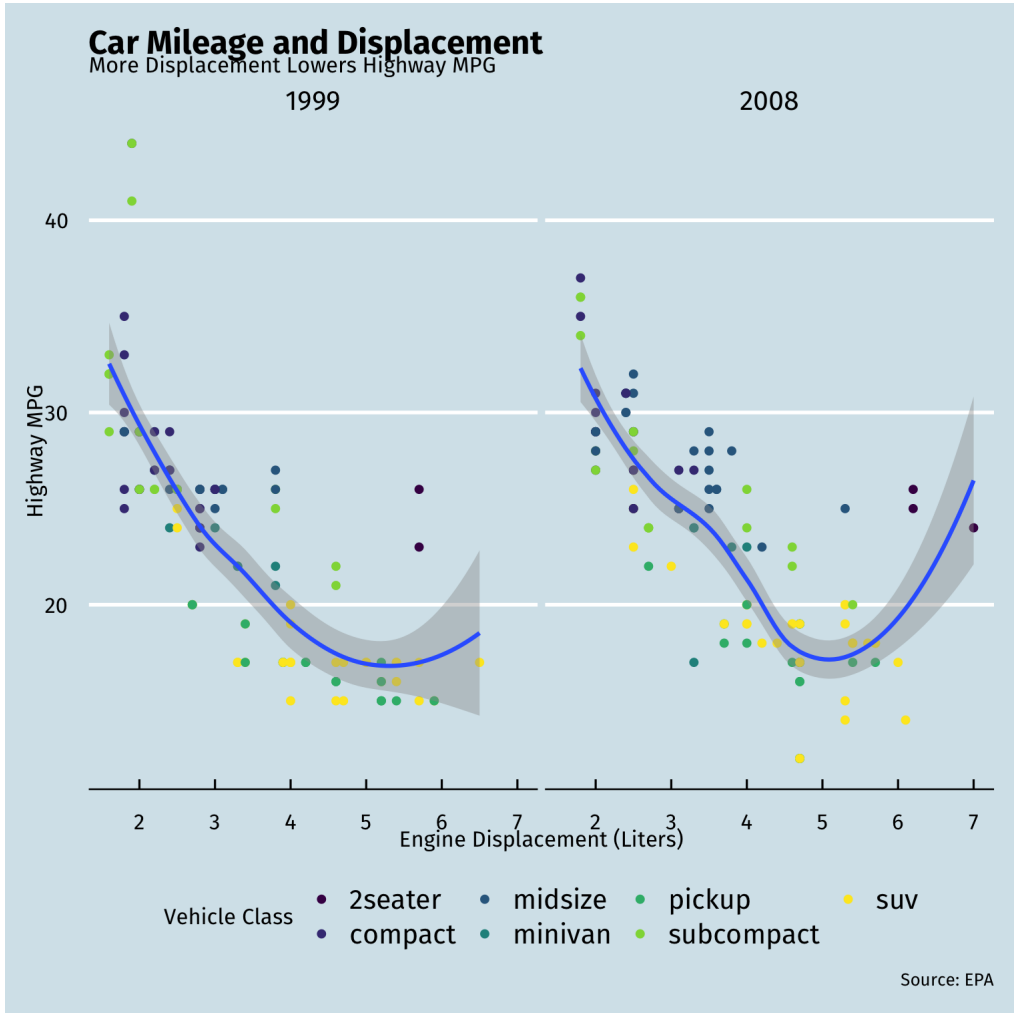
Theme

- `ggthemes` package adds some other nice themes

```
# install if you don't have it
# install.packages("ggthemes")
library("ggthemes") # load package
```

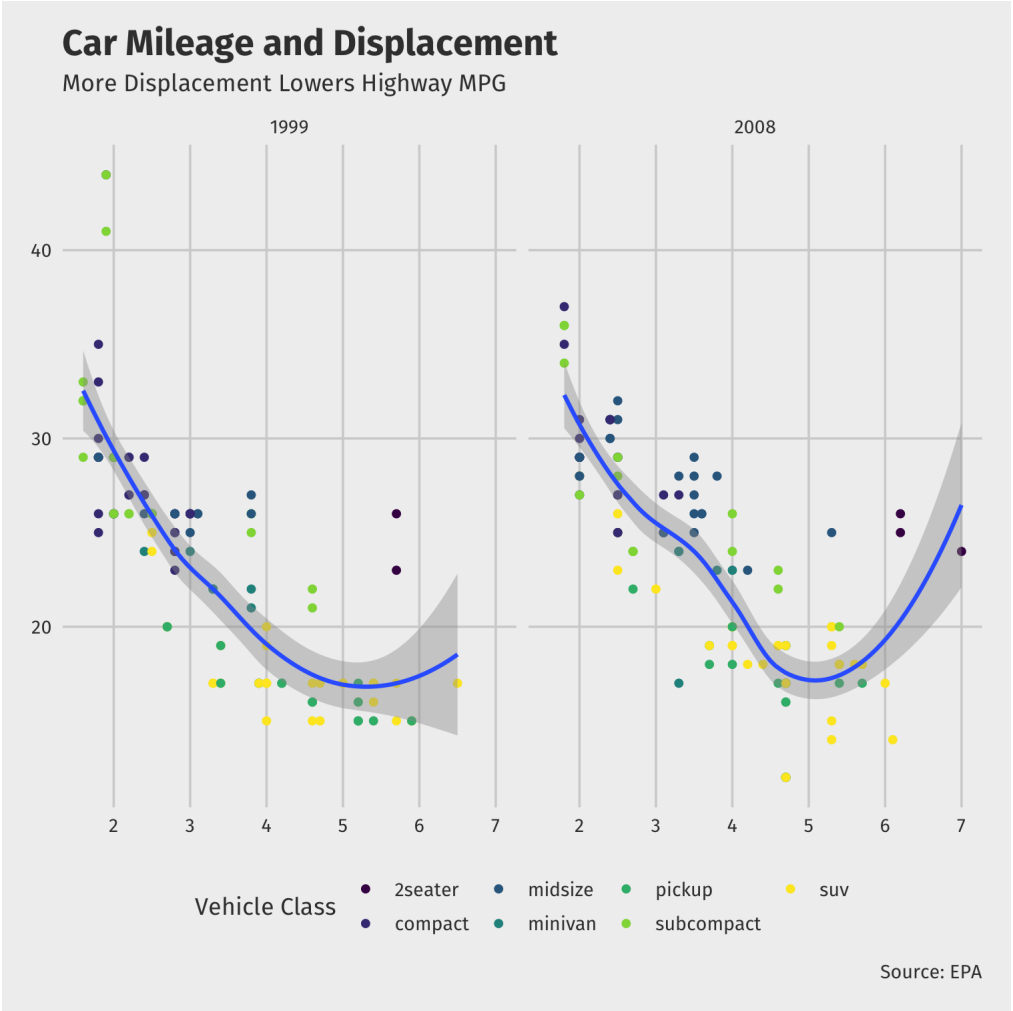
The Grammar of Graphics (gg): Themes IV

```
library("ggthemes")
ggplot(data = mpg)+
  aes(x = displ,
      y = hwy)+
  geom_point(aes(color = class))+
  geom_smooth()+
  facet_wrap(~year)+
  labs(x = "Engine Displacement (Liter
       y = "Highway MPG",
       title = "Car Mileage and Displa
       subtitle = "More Displacement L
       caption = "Source: EPA",
       color = "Vehicle Class")+
  scale_color_viridis_d()+
  theme_economist()+
  theme(text = element_text(family = "
        legend.position="bottom")
```



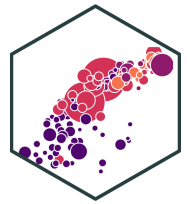
The Grammar of Graphics (gg): Themes V

```
library("ggthemes")
ggplot(data = mpg)+
  aes(x = displ,
      y = hwy)+
  geom_point(aes(color = class))+
  geom_smooth()+
  facet_wrap(~year)+
  labs(x = "Engine Displacement (Liter
       y = "Highway MPG",
       title = "Car Mileage and Displa
       subtitle = "More Displacement L
       caption = "Source: EPA",
       color = "Vehicle Class")+
  scale_color_viridis_d()+
  theme_fivethirtyeight()+
  theme(text = element_text(family = "
        legend.position="bottom")
```



Some Troubleshooting

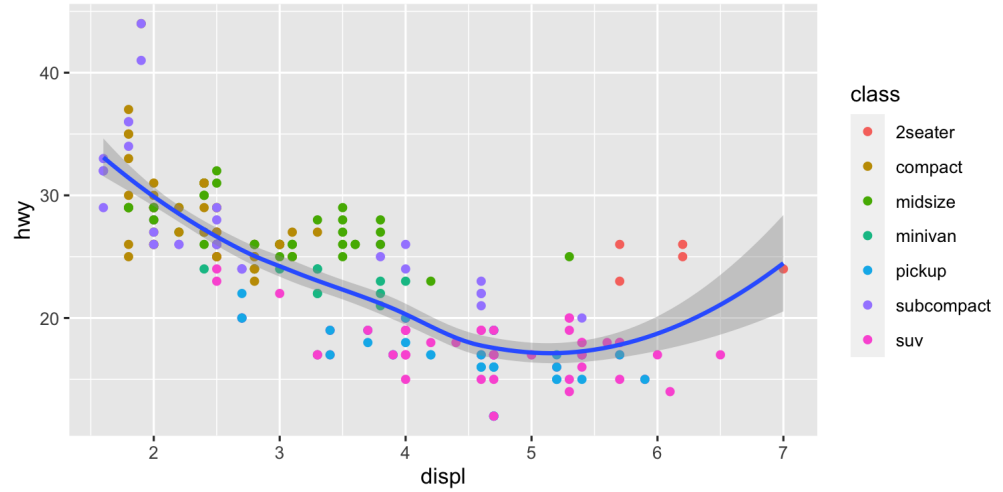
Global vs. Local Aesthetics



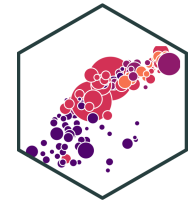
- `aes()` can go in base (`data`) layer and/or in individual `geom()` layers
- All `geoms` will inherit global `aes` from `data` layer unless overridden

```
# ALL GEOMS will map data to colors
ggplot(data = mpg, aes(x = displ,
                       y = hwy,
                       color = class))+
  geom_point()+
  geom_smooth()
```

```
# ONLY points will map data to colors
ggplot(data = mpg, aes(x = displ,
                           y = hwy))+
  geom_point(aes(color = class))+
  geom_smooth()
```

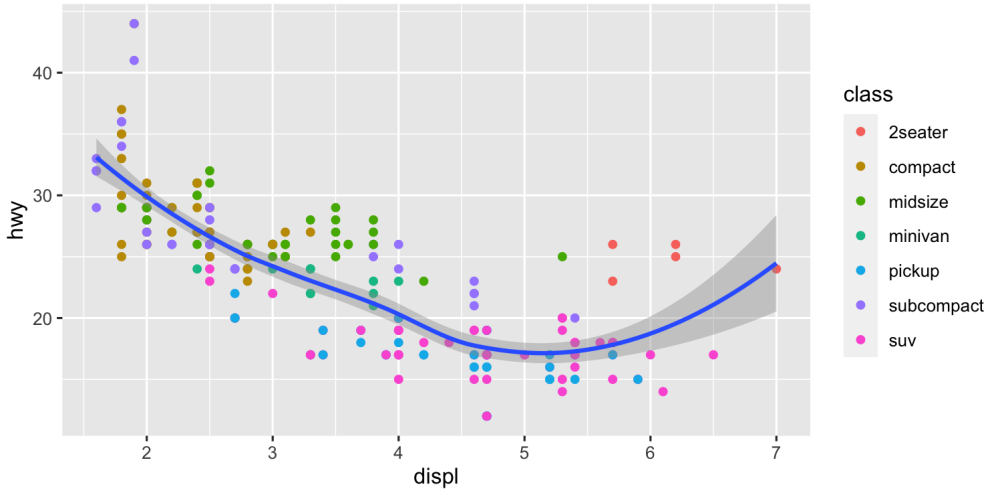


Mapped vs. Set Aesthetics

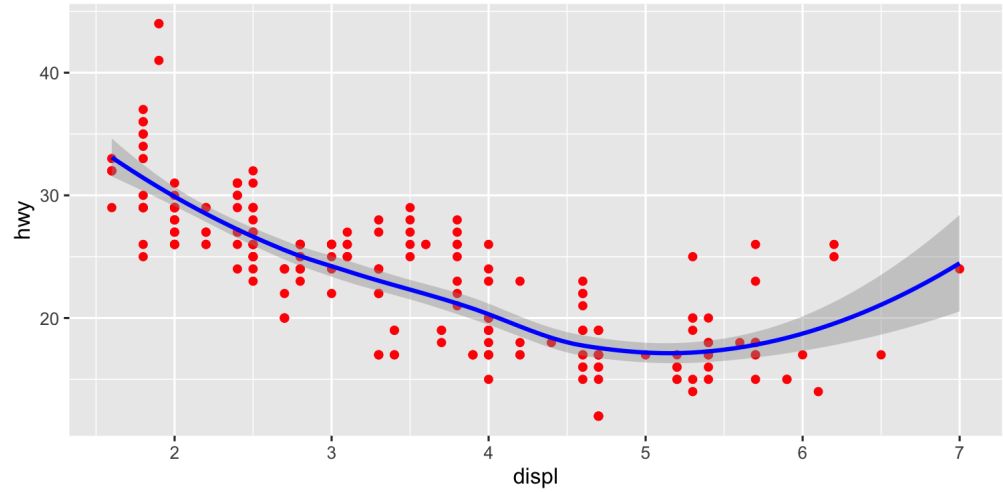


- `aes`thetics such as `size` and `color` can be mapped from data or set to a single value
- Map *inside* of `aes()`, set *outside* of `aes()`

```
# Point colors are mapped from class data
ggplot(data = mpg, aes(x = displ,
                       y = hwy))+
  geom_point(aes(color = class))+
  geom_smooth()
```

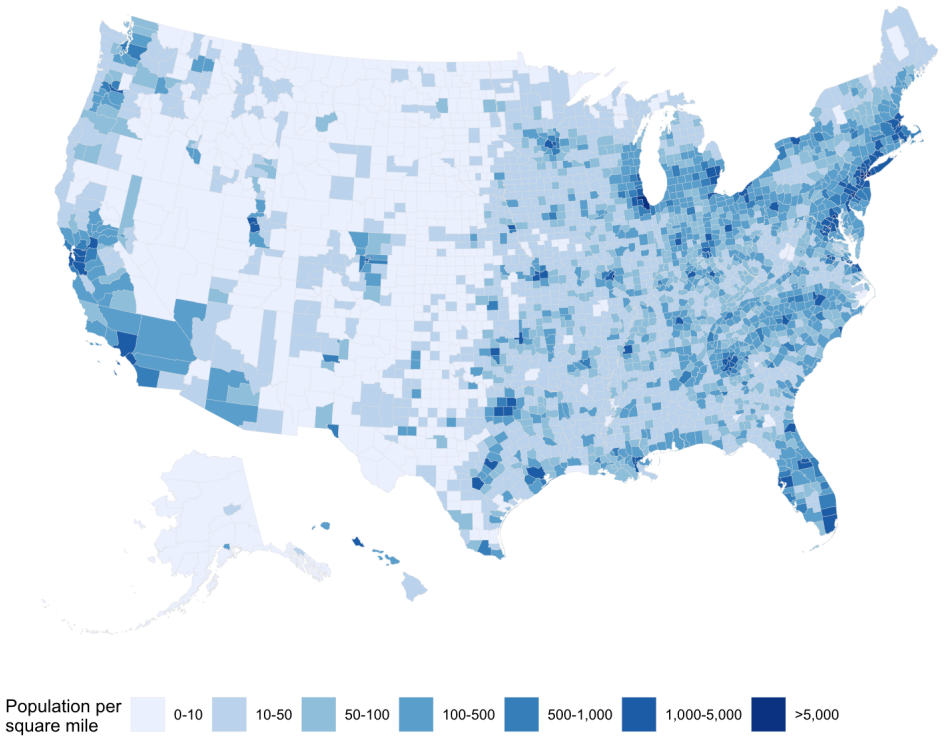



```
# Point colors are all set to blue
ggplot(data = mpg, aes(x = displ,
                       y = hwy))+
  geom_point(aes(), color = "red")+
  geom_smooth(aes(), color = "blue")
```

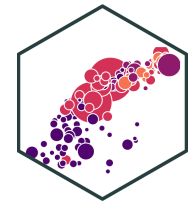


Go Crazy I

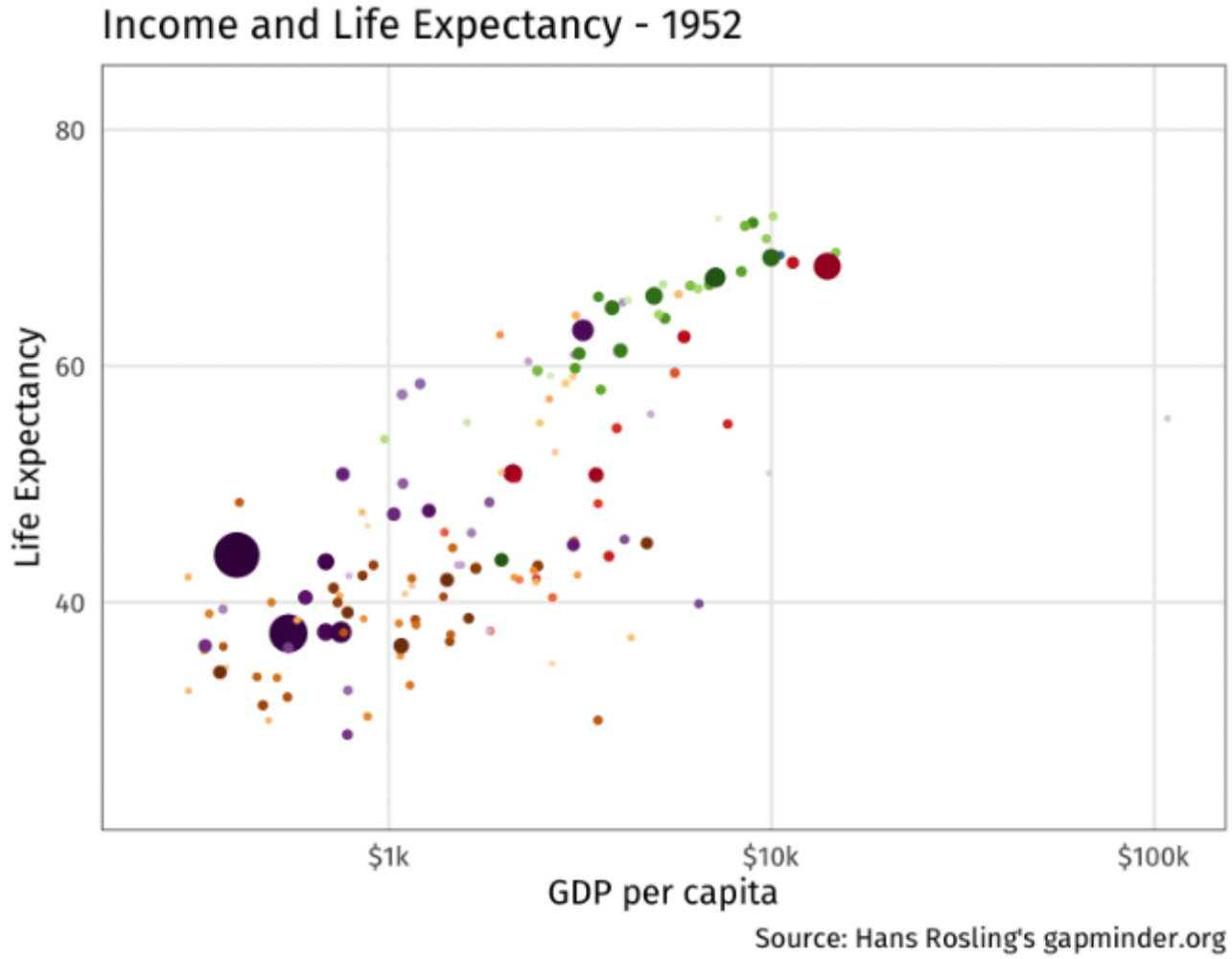
```
# I did some (hidden) data work before
ggplot(data = county_full,
            mapping = aes(x = long, y
                          fill = pop_d
                          group = grou
  geom_polygon(color = "gray90", size
  coord_equal()+
  scale_fill_brewer(palette="Blues",
                            labels =

  labs(fill = "Population per\nsquare
    theme_map() +
    guides(fill = guide_legend(nrow =
    theme(legend.position = "bottom")
```



Go Crazy II



```
library("gapminder")
library("gganimate")
ggplot(gapminder) +
  aes(x = gdpPercap, y = lifeExp, size
  geom_point() +
  guides(color = FALSE, size = FALSE)
  scale_x_log10(
    breaks = c(10^3, 10^4, 10^5),
    labels = c("$1k", "$10k", "$100k")
  scale_color_manual(values = gapminde
  scale_size(range = c(0.5, 12)) +
  labs(
    x = "GDP per capita",
    y = "Life Expectancy",
    caption = "Source: Hans Rosling's
  theme_minimal(14, base_family = "Fir
  theme(
    strip.text = element_text(size = 1
    panel.border = element_rect(fill =
```



Data Visualization and Graphic Design Principles

- We will return to various graphics as we cover descriptive statistics and regression
- I hope to cover some basic principles of good graphic design for figures and plots
    - If not in class, I will make a page on the website, and/or a video

Remember:

Less is More

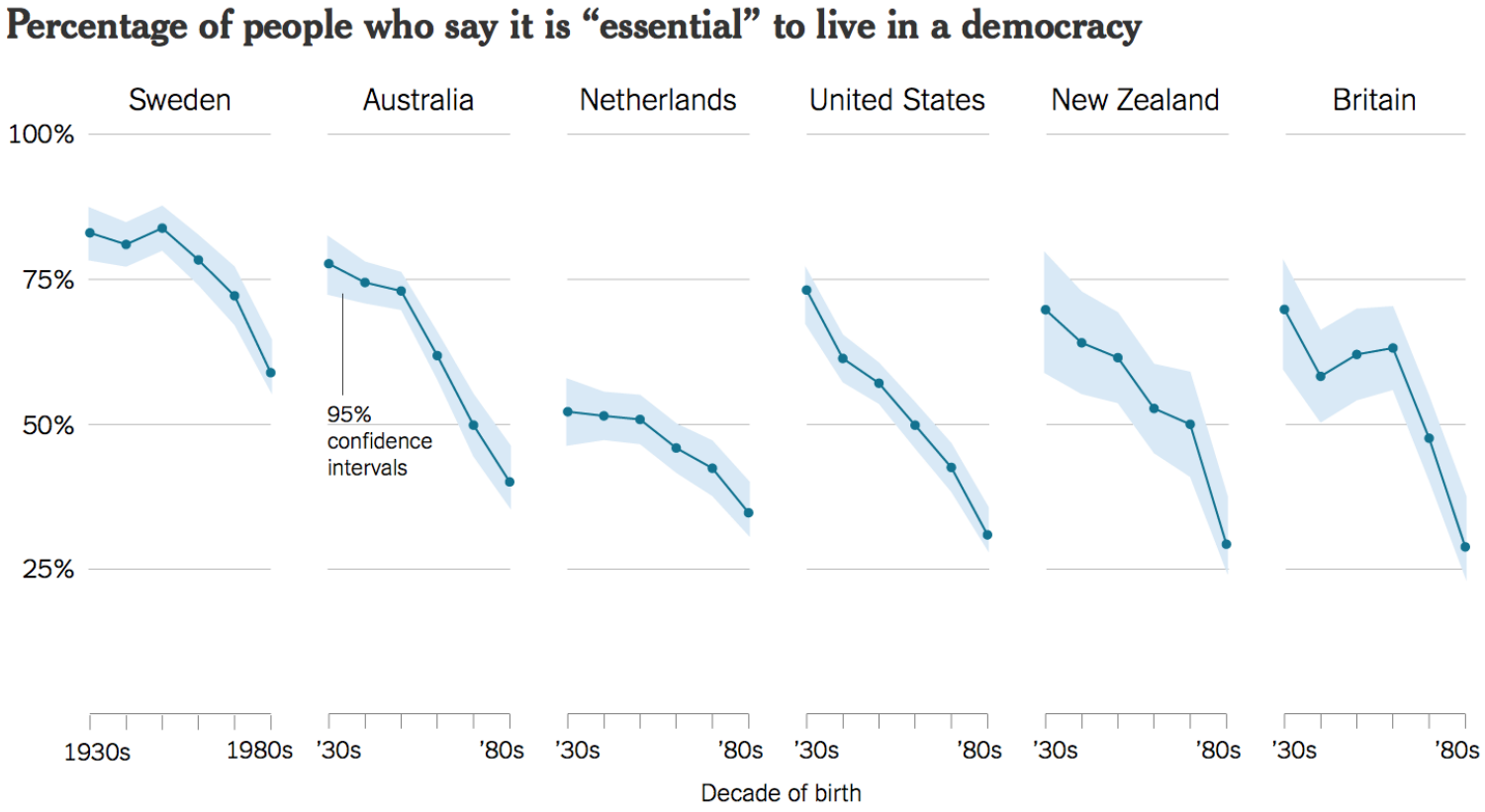
"Shoot me"

Less is More:

Remove
to improve
(the **data-ink** ratio)

Created by Darkhorse Analytics www.darkhorseanalytics.com

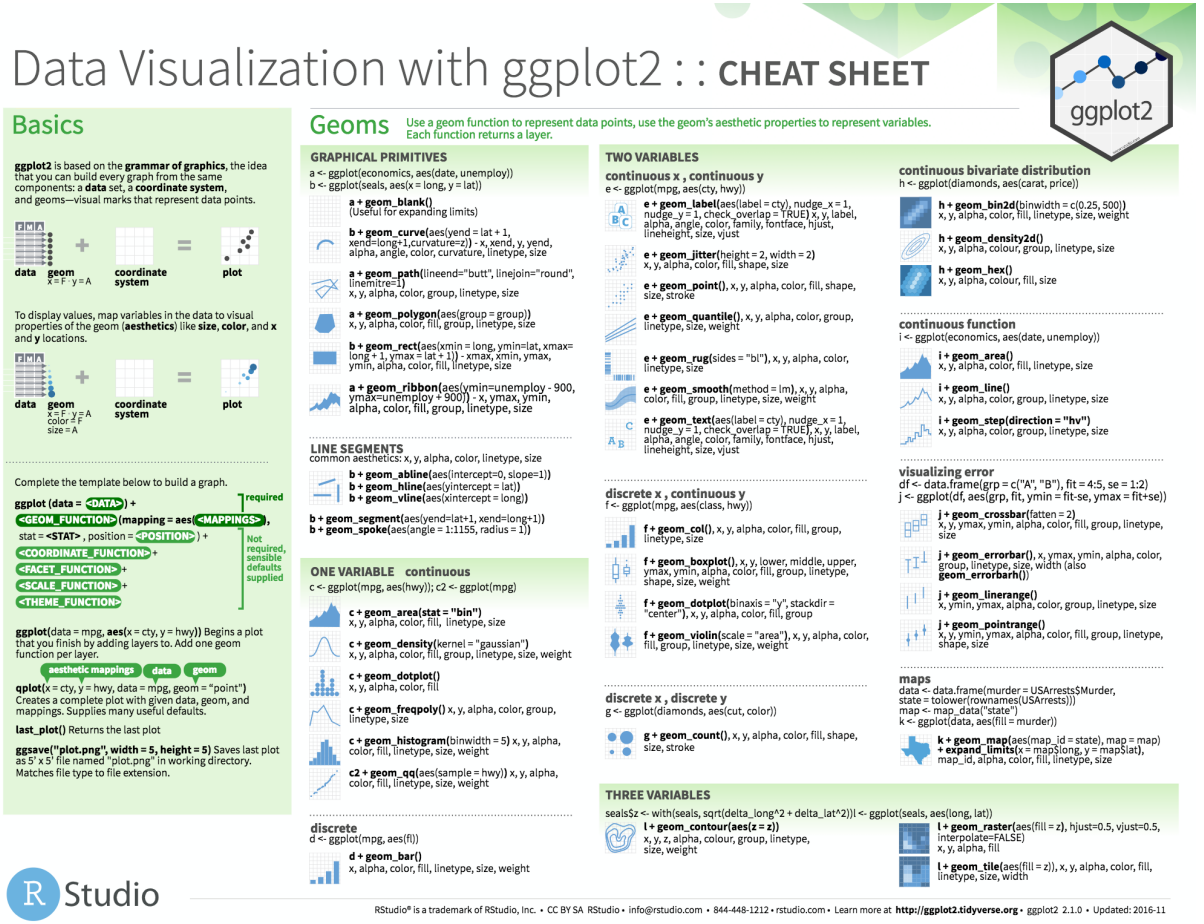
Try to Show One Trend Really Clearly



Source: Yascha Mounk and Roberto Stefan Foa, “The Signs of Democratic Deconsolidation,” Journal of Democracy | By The New York Times

New York Times: "How Stable Are Democracies? ‘Warning Signs Are Flashing Red’", Nov 29, 2016

Reference: R Studio Makes Great "Cheat Sheet"s!



Data Visualization with ggplot2 : : **CHEAT SHEET**

ggplot2

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.

data **geom** x = F, y = A | coordinate system | plot

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

data **geom** x = F, y = A, color = F, size = A | coordinate system | plot

Complete the template below to build a graph.

```
ggplot (data = <DATA>) +                                   required
<GEOM_FUNCTION> (mapping = aes( <MAPPINGS> ),
 stat = <STAT> , position = <POSITION> ) +                 Not required, sensible defaults supplied
<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=z)) - 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=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 <- 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, 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 <- ggplot(diamonds, aes(cut, color))

g + geom_count(), x, y, alpha, color, fill, shape, size, stroke

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, 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

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

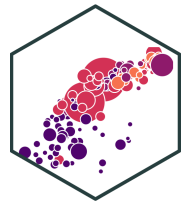
l + geom_raster(aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE) x, y, alpha, fill

l + geom_tile(aes(fill = z)), x, y, alpha, color, fill, linetype, size, width

RStudio

RStudio: ggplot2 Cheat Sheet

Reference



On `ggplot2`

- **R Studio's ggplot2 Cheat Sheet**
- **`ggplot2`'s website reference section**
- Hadley Wickham's R for Data Science book chapter on ggplot2
- STHDA's be awesome in ggplot2
- r-statistic's top 50 ggplot2 visualizations

On data visualization

- **Kieran Healy's Data Visualization: A Practical Guide**
- **Claus Wilke's Fundamentals of Data Visualization**
- PolicyViz Better Presentations
- Karl Broman's How to Display Data Badly