Accelerated Lecture 5: Data Visualization

Harris Coding Camp – Accelerate Track

Summer 2022

Today's lesson

- Why visualize data?
- How to do so with ggplot
  - How to map data to aesthetics with `aes()` (and what that means)
  - How to visualize the mappings with `geoms`
  - How to get more out of your data by using multiple aesthetics
  - How to use facets to add dimensionality
- Some base R tips

We have entire courses on data visualization. This is just a sample.

Data Visualization: Motivation

Suppose we want to know the following info:

- How have annual housing sales in Texas changed over time?
- How do these trends compare between cities?

Data Visualization: Motivation

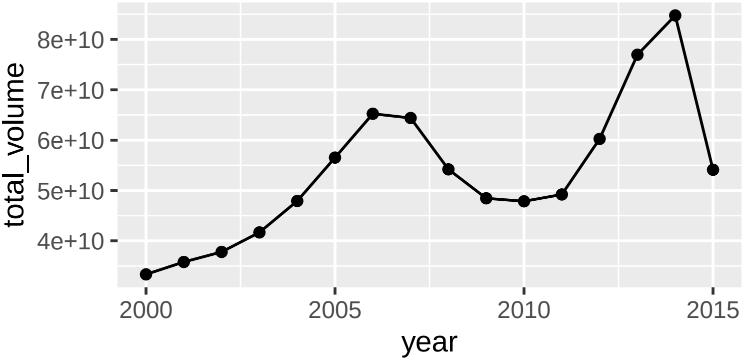
Is simply looking at all these data helpful?

```
## # A tibble: 10 x 2
##     year total_volume
##    <int>        <dbl>
##  1  2000  33342410971
##  2  2001  35804815138
##  3  2002  37798888462
##  4  2003  41674204834
##  5  2004  47913188880
##  6  2005  56534755111
##  7  2006  65237510783
##  8  2007  64393979596
##  9  2008  54198855809
## 10  2009  48450447327
```

Data Visualization: Motivation

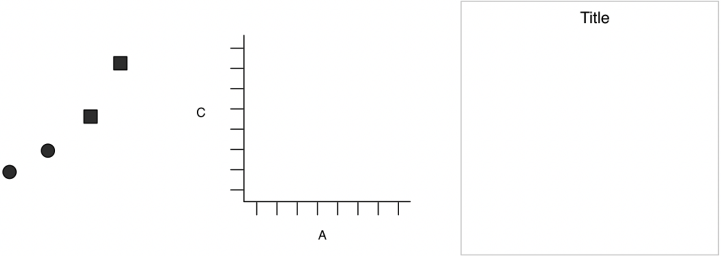
What if we make a plot of annual housing sales over time. . .

- Now we can quickly understand and communicate about our data



Challenge: How do we efficiently communicate how to visualize data to the computer?

Introducing the "grammar of graphics" and `ggplot`



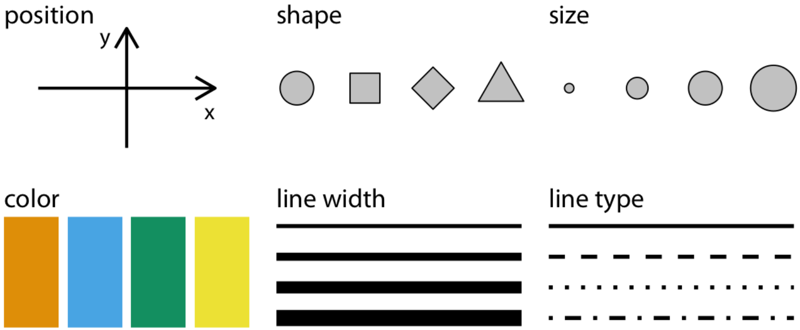
- gg = Grammar of graphics (Wickham 2010, Wilkinson et. al 2005, Bertin 1983)
  - Rules for how to put graph-parts together to make a logical expression
  - Implemented in R with `ggplot2`, a data visualization package in the tidyverse

Basic Components of `ggplot` (Layers)

- Layer 1: Scales and coordinates (`ggplot()`)
    - A data frame
    - Aesthetic mapping: how data are mapped to x-axis, y-axis, color, size, etc
- Layer 2: Geometry (`geom_xxx()`)
    - geometric objects like points, lines, shapes
- Layer 3: Labels (`labs()`)
    - title, legend, etc

What is Aesthetic?

- An aesthetic is a visual property of the objects in your plot
  - Including things like size, shape, color or x and y locations
- To display values, map variables in the data to visual properties of the `geom` (aesthetics)



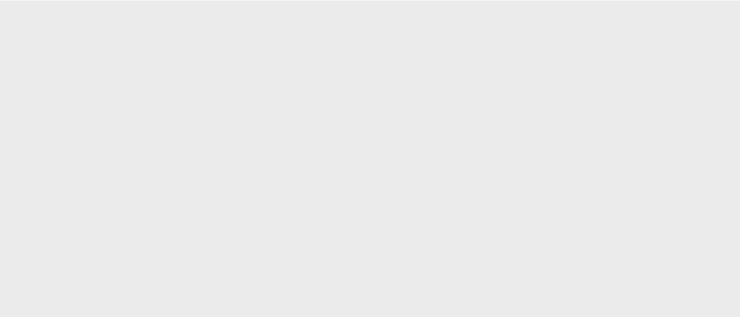
Simplest ggplot code structure

```
ggplot(data = dataset,
       mapping = aes(x = x_variable,
                     y = y_variable))
  geom_<name>()
```

ggplot() tells R to prepare to make a plot.

```r
# Let's prepare new data frame 'annual_sales'
annual_sales <-
  txhousing %>%
  group_by(year) %>%
  summarize(total_volume = sum(volume, na.rm = TRUE))
```

```r
# Layer 1, data frame
ggplot(data = annual_sales)
```

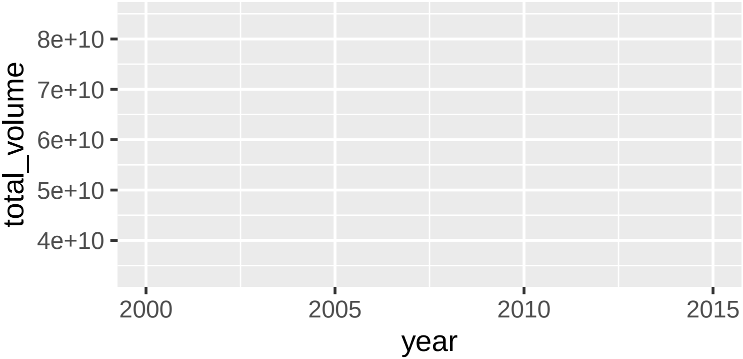


Layer 1: adding an aesthetic mapping

`mapping = aes()` declares how to map the data to "aesthetics":

- R will map each row of the data (`year`, `total_volume`) to the (`x`,`y`)
- i.e. tell R to make x-axis `year` and y-axis `total_volume`

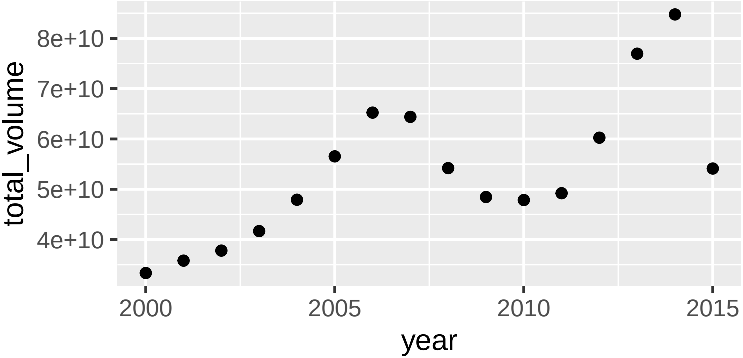
```
ggplot(data = annual_sales,
       mapping = aes(x = year, y = total_volume))
```



Layer 2: visualizing the mapping with geom

Here we see points by using geom_point():

```
ggplot(data = annual_sales,
       mapping = aes(x = year, y = total_volume)) +
  geom_point()
```

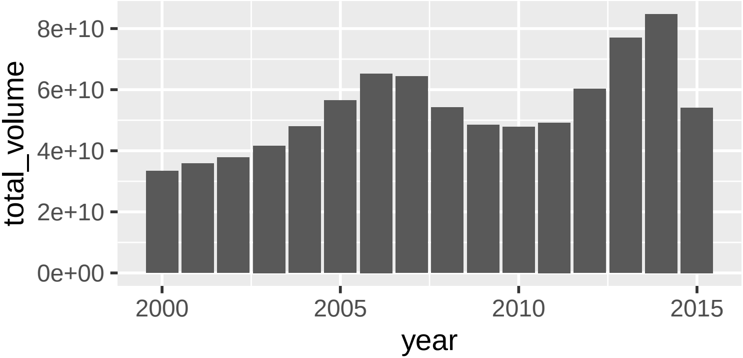


Layer 2: visualizing the mapping with geom

Here we see bars by using `geom_col()`.

- Each *observation* or row has a (`year`, `total_volume`) mapped to the coordinate pair (`x`,`y`)

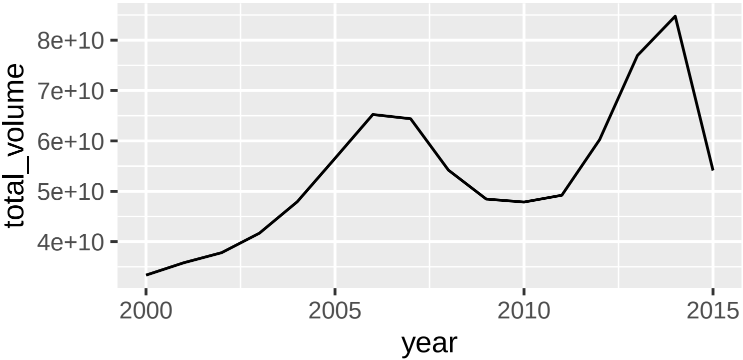
```
ggplot(data = annual_sales,
       mapping = aes(x = year, y = total_volume)) +
  geom_col()
```



Layer 2: visualizing the mapping with geom

Here we see a line connecting each (x,y) pair using geom_line().

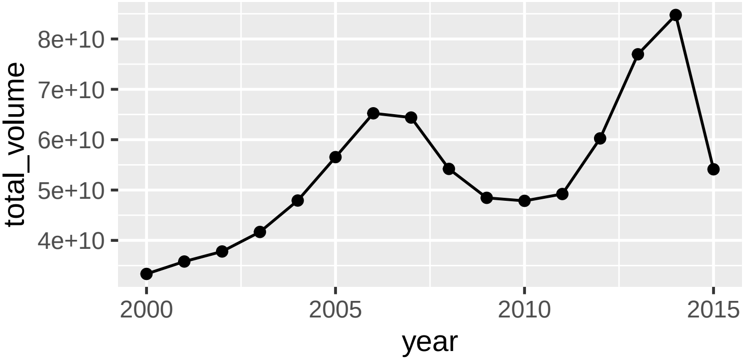
```
ggplot(data = annual_sales,
       mapping = aes(x = year, y = total_volume)) +
  geom_line()
```



Layer 2: visualizing the mapping with geom

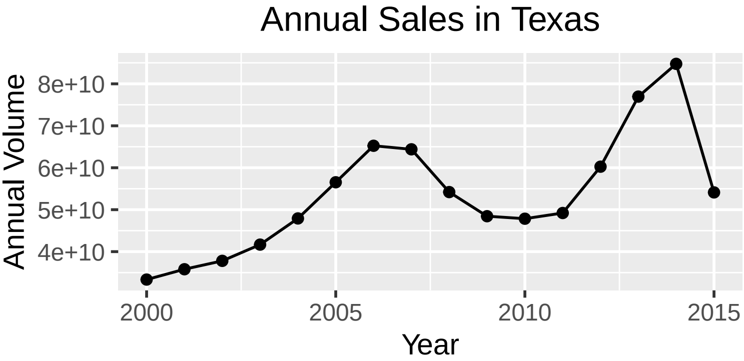
The data can be visualized with different geoms that can be composed (+) together:

```
ggplot(data = annual_sales,
       mapping = aes(x = year, y = total_volume)) +
  geom_line() +
  geom_point()
```



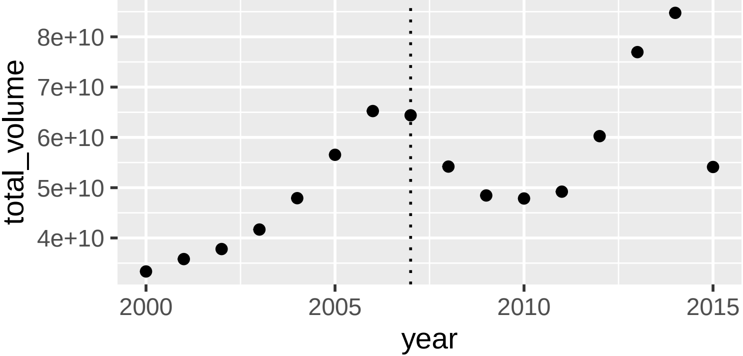
Layer 3: Adding labels makes the plot more readable:

```
ggplot(data = annual_sales,
       mapping = aes(x = year, y = total_volume)) +
  geom_line() +
  geom_point() +
  labs(x = "Year", y = "Annual Volume",
       title = "Annual Sales in Texas") +
  theme(plot.title = element_text(hjust = 0.5)) #center the title
```



Over laying multiple geoms: adding vertical lines

```
annual_sales %>%
  ggplot(aes(x = year, y = total_volume)) +
    geom_point() +
    geom_vline(aes(xintercept = 2007),
                   linetype = "dotted")
```

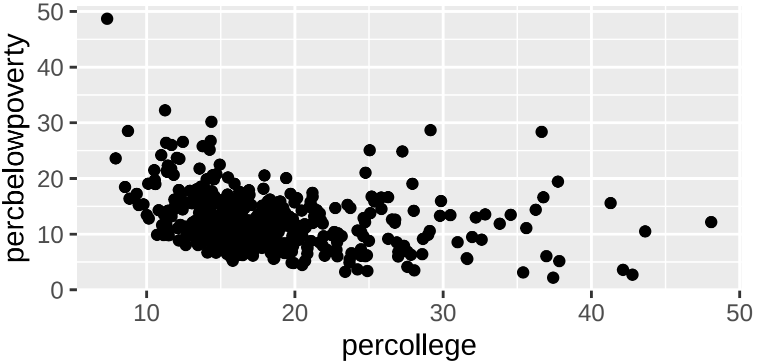


- add horizontal lines with `geom_hline()`
- add any linear fit with `geom_abline()` by providing a slope and intercept

aesthetics beyond the x and y position

We'll use midwest data and start with only mapping to x and y

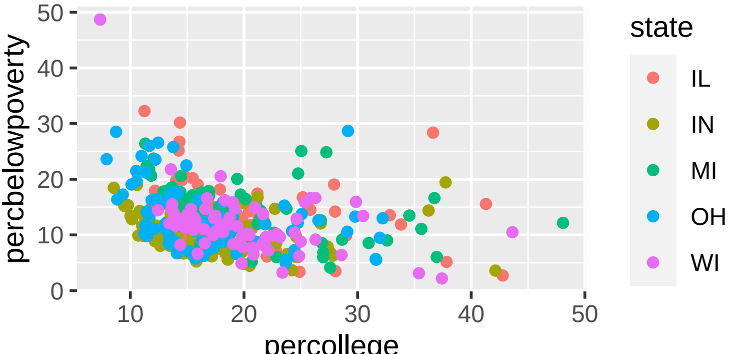
```
midwest %>%
   ggplot(aes(x = percollege,
              y = percbelowpoverty)) +
      geom_point()
```



ggplot(): Using color

- color maps data to the color of points or lines
  - Each state is assigned a color
  - This works with discrete data and continuous data

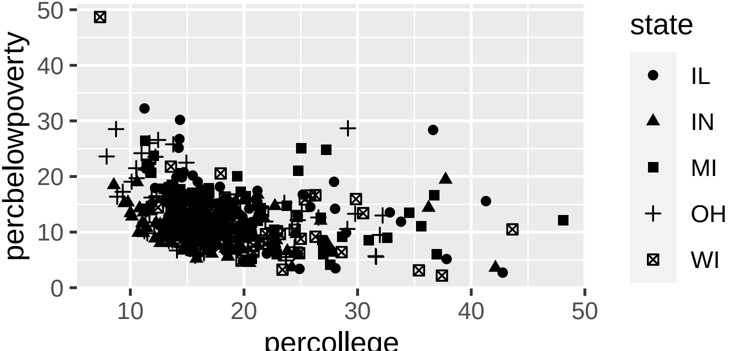
```
midwest %>%
  ggplot(aes(x = percollege,
             y = percbelowpoverty,
             color = state)) +
    geom_point()
```



ggplot(): Using shape

- shape maps data to the shape of points
  - Each state is assigned a shape
  - This works with discrete data only

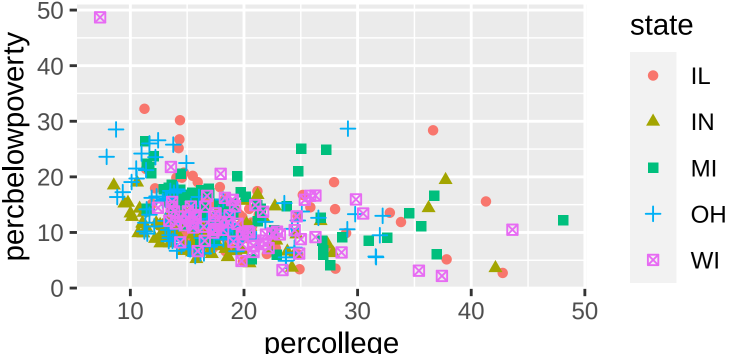
```
midwest %>%
  ggplot(aes(x = percollege,
             y = percbelowpoverty,
             shape = state)) +
    geom_point()
```



ggplot(): Using color + shape

- Combining color and shape:
  - Each state is assigned a shape and color

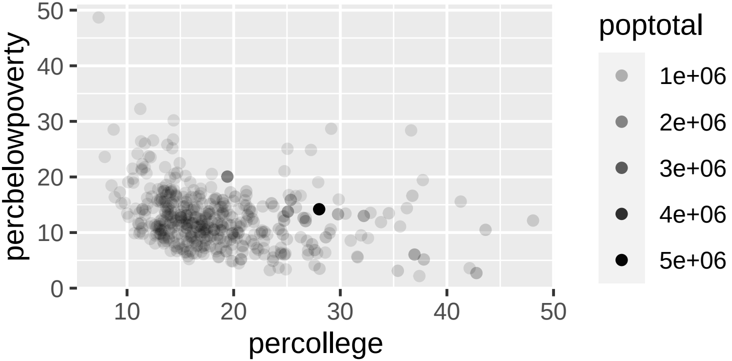
```
midwest %>%
  ggplot(aes(x = percollege,
             y = percbelowpoverty,
             color = state,
             shape = state)) +
    geom_point()
```



ggplot(): Using alpha

- alpha maps data to the transparency of points
- we map the percentage of people within a known poverty status to alpha

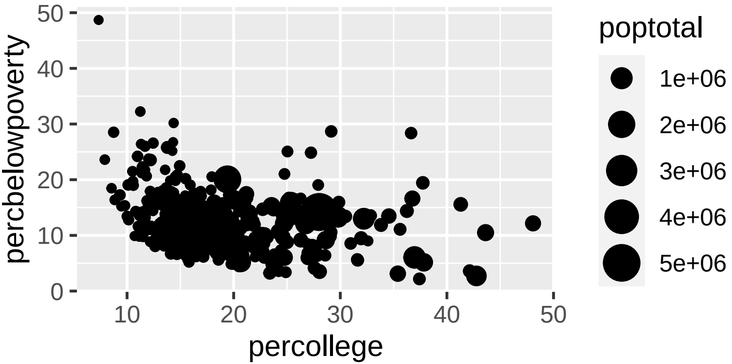
```
midwest %>%
  ggplot(aes(x = percollege,
             y = percbelowpoverty,
             alpha = poptotal)) +
    geom_point()
```



ggplot(): Using size

- size maps data to the size of points and width of lines.
- we map the percentage of people within a known poverty status to size

```
midwest %>%
  ggplot(aes(x = percollege,
             y = percbelowpoverty,
             size = poptotal)) +
    geom_point()
```

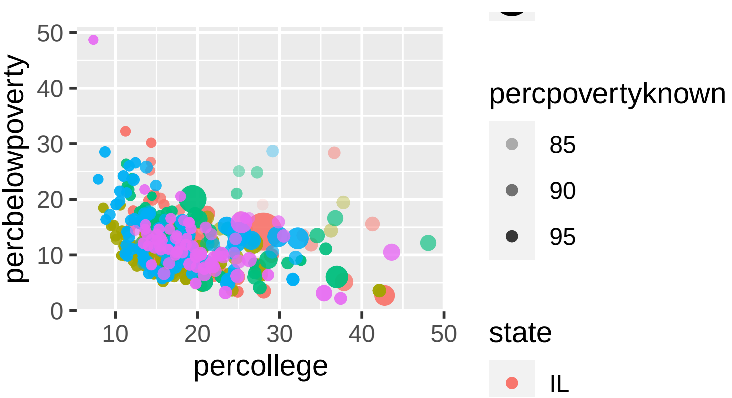


ggplot(): Using multiple aesthetics together

We can combine any and all aesthetics, and even map the same variable to multiple aesthetics

```
midwest %>%
   ggplot(aes(x = percollege,
              y = percbelowpoverty,
              alpha = percpovertyknown,
              size = poptotal,
              color = state)) +
      geom_point()
```

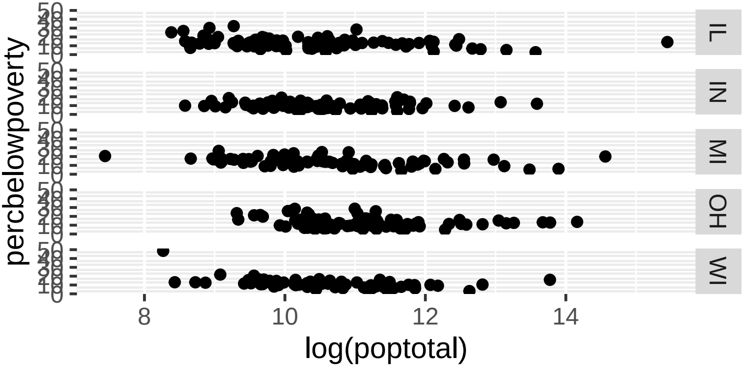
ggplot(): Using multiple aesthetics together



ggplot(): Facets (facet_grid)

Facets provide an additional tool to explore multidimensional data:

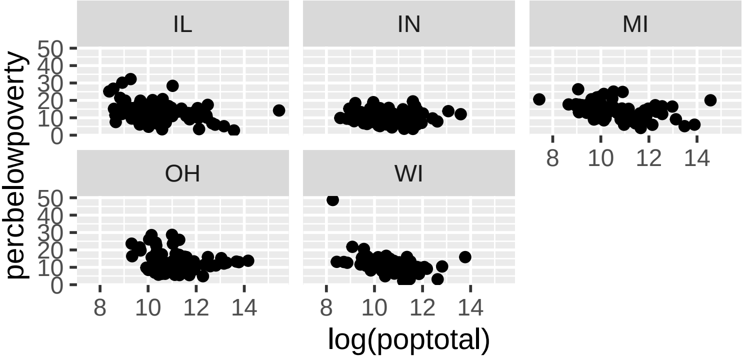
```
midwest %>%
   ggplot(aes(x = log(poptotal),
              y = percbelowpoverty)) +
      geom_point() +
      facet_grid(vars(state))
```



ggplot(): Facets (facet_wrap)

Facets provide an additional tool to explore multidimensional data:

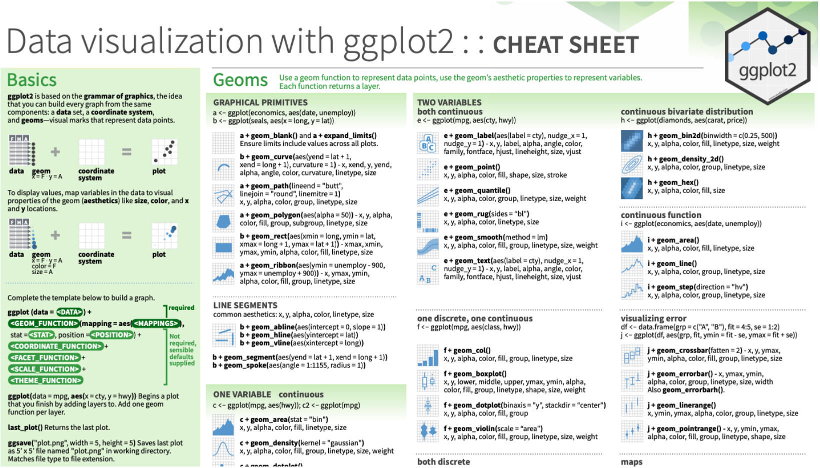
```
midwest %>%
  ggplot(aes(x = log(poptotal),
             y = percbelowpoverty)) +
    geom_point() +
    facet_wrap(vars(state))
```



ggplot(): Using aesthetics to explore data

Different geoms have specific aesthetics that go with them.

- the ggplot cheatsheet shows all the geoms with their associated aesthetics



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.

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() and **a + expand_limits()**
Ensure limits include values across all plots.

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(alpha = 50)**)** - x, y, alpha, color, fill, group, subgroup, 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()

TWO VARIABLES

both continuous

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

e + geom_label(aes(label = cty), nudge_x = 1, nudge_y = 1**)** - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

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**)** - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

one discrete, one continuous

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

both discrete

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_density_2d()
x, y, alpha, color, group, linetype, size

h + geom_hex()
x, y, alpha, color, 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()**.

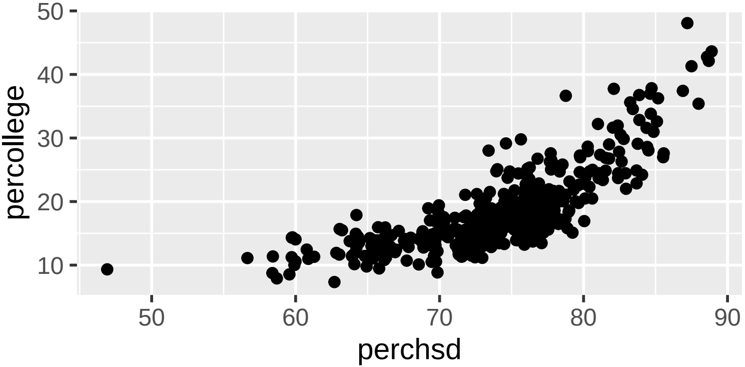
j + geom_linerange()
x, ymin, ymax, alpha, color, group, linetype, size

j + geom_pointrange() - x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

maps

Try it yourself: Plot 1

1. Adjust code to reproduce the following plot (sample codes provided in the next slide):

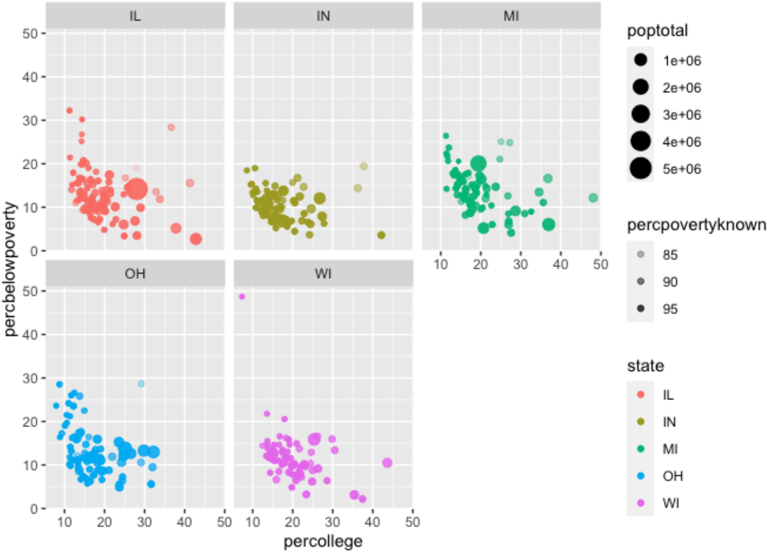


Try it yourself: Plot 1

```r
midwest %>%
  ggplot(aes(x = ?,
             y = ?,
            )) +
  geom_?()
```

Try it yourself: Plot 2

2. Adjust code to reproduce the following plot (sample codes provided in the next slide):



Try it yourself: Plot 2

```
midwest %>%
  ggplot(aes(x = ?,
             y = ?,
             color = state,
             size = ?,
             alpha = percpovertyknown)) +
  geom_?() +
  facet_wrap(vars(?))
```

discrete vs continuous data

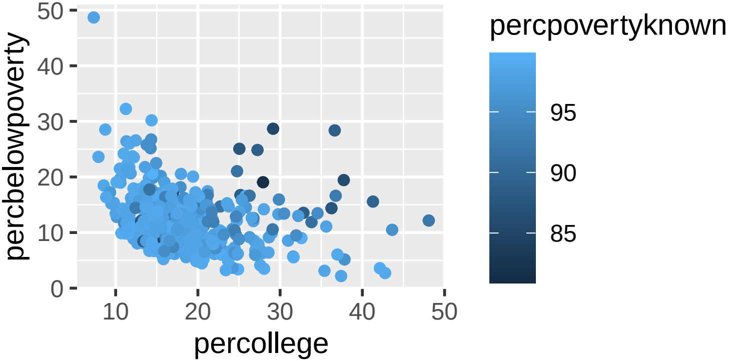
aes	discrete	continuous
	limited number of classes	unlimited number of classes
	usually `chr` or `lgl`	numeric
`x`, `y`	yes	yes
`color`, `fill`	yes	yes
`shape`	yes (6 or fewer categories)	no
`size`, `alpha`	not advised	yes
`facet`	yes	not advised

Here, discrete and continuous have different meaning than in math

- For `ggplot` meaning is more fluid.
- If there are fewer than 6 to 10 groups, discrete visualizations can work
- If your "discrete" data is numeric, use `as.character()` or `as_factor()` to enforce the decision.

color can be continuous

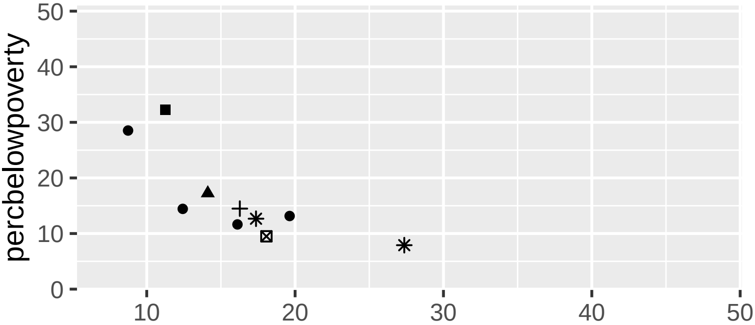
```
midwest %>%
    ggplot(aes(x = percollege,
               y = percbelowpoverty,
               color = percpovertyknown)) +
  geom_point()
```



shape does not play well with many categories

- Will only map to 6 categories, the rest become `NA`.
- We can override this behavior and get up to 25 distinct shapes

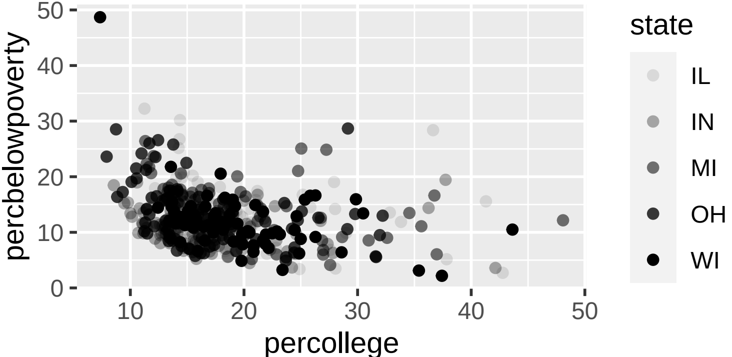
```
midwest %>%
    ggplot(aes(x = percollege,
               y = percbelowpoverty,
               shape = county)) +
  geom_point() +
  # legend off, otherwise it overwhelms
  theme(legend.position = "none")
```



alpha and size can be misleading with discrete data

```
midwest %>%
    ggplot(aes(x = percollege,
               y = percbelowpoverty,
               alpha = state)) +
  geom_point()
```

```
## Warning: Using alpha for a discrete variable is not adv
```



Type of figures

1. Distribution of **univariate (single variable)**

- bar plot, histogram, density plot, etc

2. Relationship between **bivariate (two variables)**

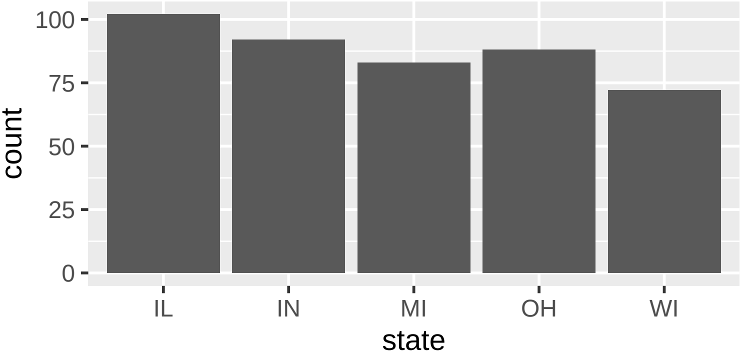
- scatter plot, line plot, boxplot, (segmented) bar plot, etc

3. Relationship between **many variables** at once

- usually focusing on the relationship between two while conditioning for others

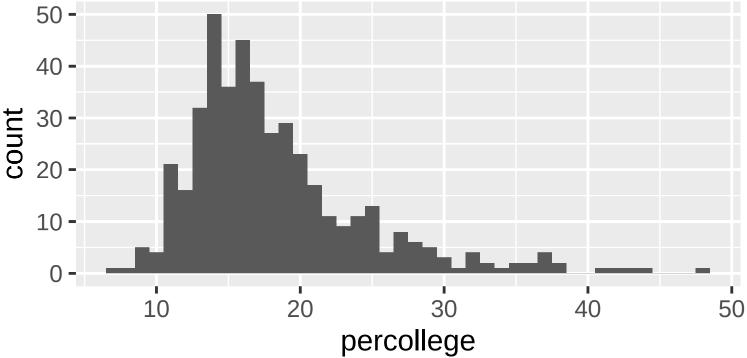
Univariate: bar plot

```
midwest %>%
   ggplot(aes(x = state)) +
      geom_bar()
```



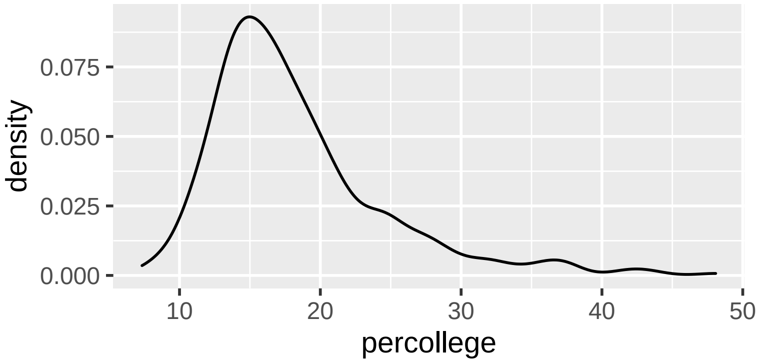
Univariate: histogram

```r
midwest %>%
  ggplot(aes(x = percollege)) +
    geom_histogram(binwidth = 1)
```



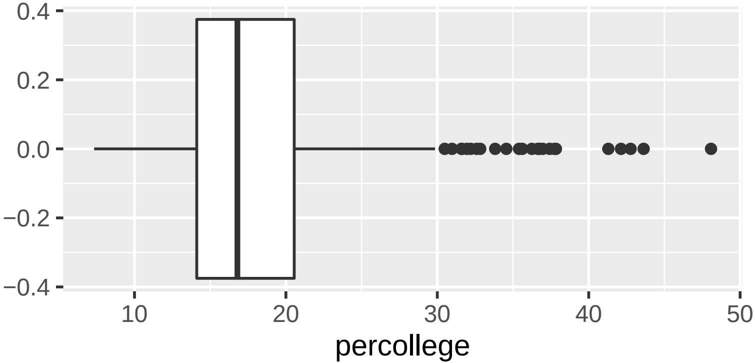
Univariate: density

```
midwest %>%
   ggplot(aes(x = percollege)) +
      geom_density()
```



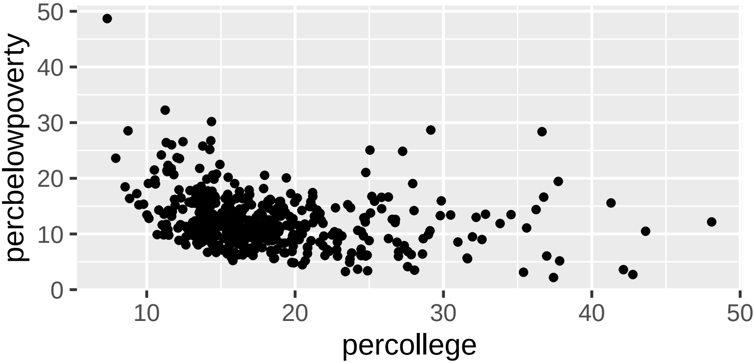
Univariate: box plots

```
midwest %>%
   ggplot(aes(x = percollege)) +
      geom_boxplot()
```



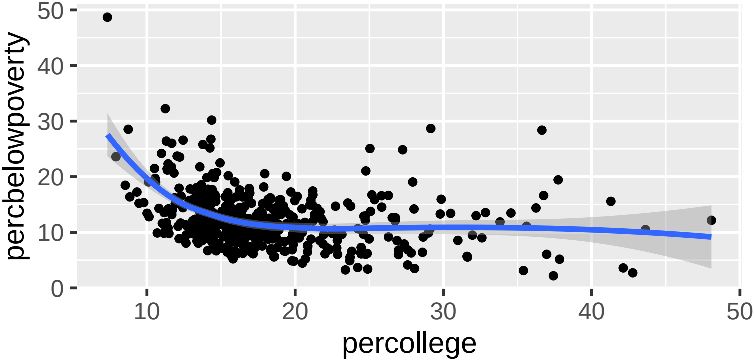
Bivariate: scatter plot

```
midwest %>%
    ggplot(aes(x = percollege,
               y = percbelowpoverty)) +
      geom_point(size=1)
```



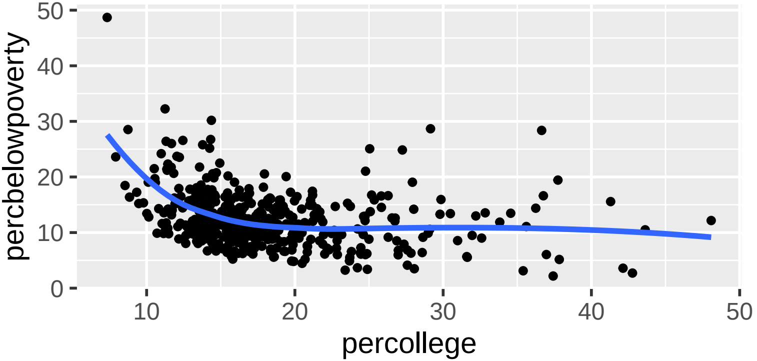
Bivariate: scatter + smooth Line plot

```
midwest %>%
    ggplot(aes(x = percollege,
               y = percbelowpoverty)) +
        geom_point(size = 1) +
        geom_smooth()
```



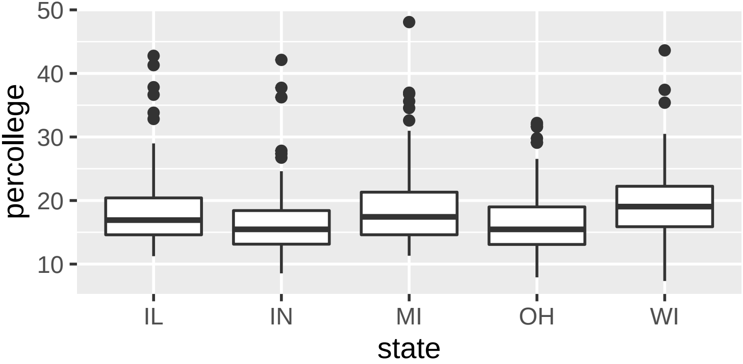
Bivariate: scatter + smooth Line plot

```
midwest %>%
   ggplot(aes(x = percollege,
              y = percbelowpoverty)) +
      geom_point(size = 1) +
      geom_smooth(se = FALSE)  # turn off std errors
```



Bivariate: box plots

```
midwest %>%
   ggplot(aes(x = state,
              y = percollege)) +
      geom_boxplot()
```



Recap

- Visualizing our data can help lead to powerful insights between variable relationships
  - Making quick plots helps us understand data and makes us aware of data issues
- `ggplot` starts by mapping data to "aesthetics"
  - e.g. What data shows up on `x` and `y` axes and how `color`, `size` and `shape` appear on the plot
- Then, we use `geoms` to create a visualization based on the mapping
- We many consider adding labels to make plots more readable
- There are many ways you can visualize your data!

Next steps

Labs

- Today: Data visualization with `ggplot` (may run into tomorrow)
- Tomorrow: Introducing plotting in base R

I can produce basic plots to explore and communicate about data

Lecture

- Data manipulation and analysis with groups

Appendix: Some graphs you made along the way

- Distributions
- Grouped bar graph
- Faceted bar graph

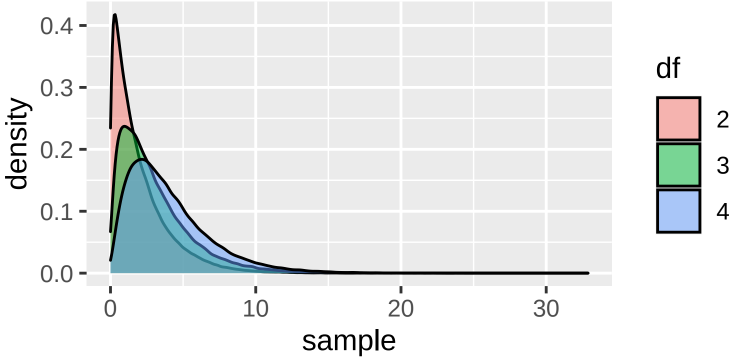
Appendix: distributions

- geom_density() only requires an x aesthetic and it calculates the distribution to plot.
- We can set the aesthetics manually, independent of data for nicer graphs.

```r
chi_sq_samples <-
 tibble(x = c(rchisq(100000, 2),
              rchisq(100000, 3),
              rchisq(100000, 4)),
        df = rep(c("2", "3", "4"), each = 1e5))

chi_sq_samples %>%
  ggplot(aes(x = x, fill = df)) +
  geom_density( alpha = .5) +
  labs(fill = "df", x = "sample")
```

Appendix: distributions

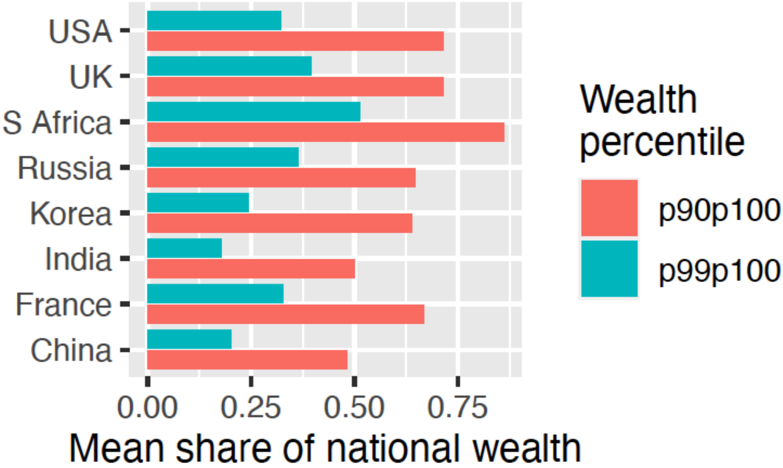


Appendix: grouped bar graph

- `position = "dodge2"` tells R to put bars next to each other, rather than stacked on top of each other.
- Notice we use `fill` and not `color` because we're “filling” an area.

```
mean_share_per_country %>%
  ggplot(aes(y = country,
             x = mean_share,
             fill = percentile)) +
  geom_col(position = "dodge2") +
  labs(x = "Mean share of national wealth",
       y = "",
       fill = "Wealth\npercentile")
```

Appendix: grouped bar graph

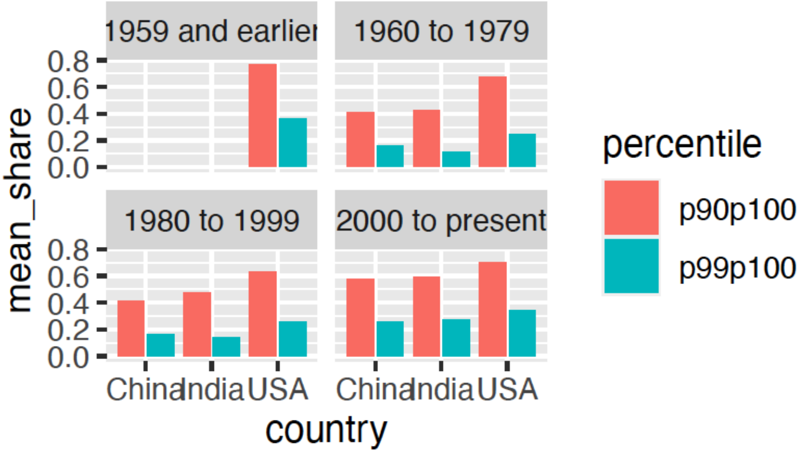


Appendix: faceted bar graph

- Notice that we manipulate our data to the right specification before making this graph
- Using `facet_wrap` we get a distinct graph for each time period.

```r
mean_share_per_country_with_time %>%
 ggplot(aes(x = country,
            y = mean_share,
            fill = percentile)) +
   geom_col(position = "dodge2") +
   facet_wrap(vars(time_period))
```

Appendix: faceted bar graph


1959 and earlier
1960 to 1979
1980 to 1999
2000 to present
mean_share
China
India
USA
country
percentile
p90p100
p99p100