Accelerated Lecture 5: Data Visualization

Harris Coding Camp – Accelerate Track

Summer 2022

Today's lesson

- Conceptual goal: How does a data visualization communicate about the underlying data?
- Coding goal: How do we tell the computer how to make the plot
  - How to map data to aesthetics with `aes()`
  - 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

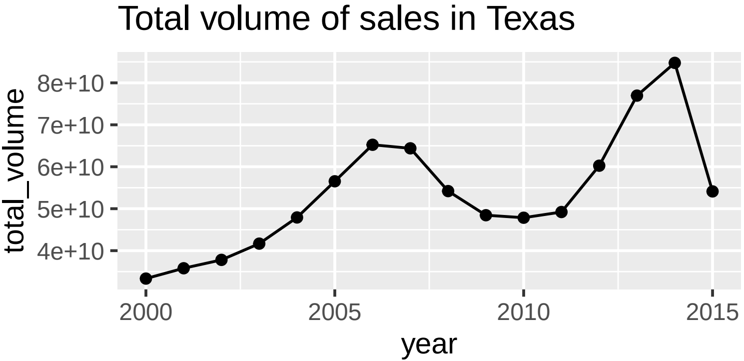
We have entire courses on data visualization. This is just a sample.

How have annual housing sales in Texas changed over time?

"Look at the data"

```
## # 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
```

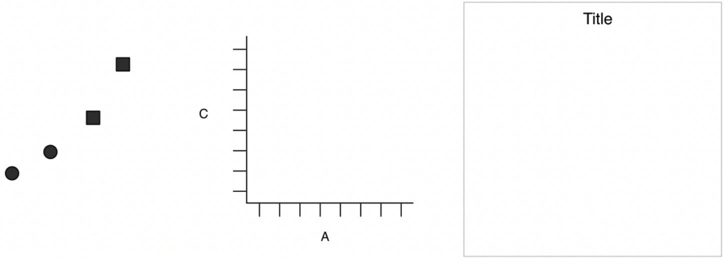
What if we make a plot of annual housing sales over time. . .



How do we tell the computer how to plot from the underlying data?

A “grammar of graphics” (Wickham 2010, Wilkinson et. al 2005, Bertin 1983)

- Grammar = “the whole system and structure of a language”
- i.e a set of rules for how to combine information in order to communicate



ggplot code structure

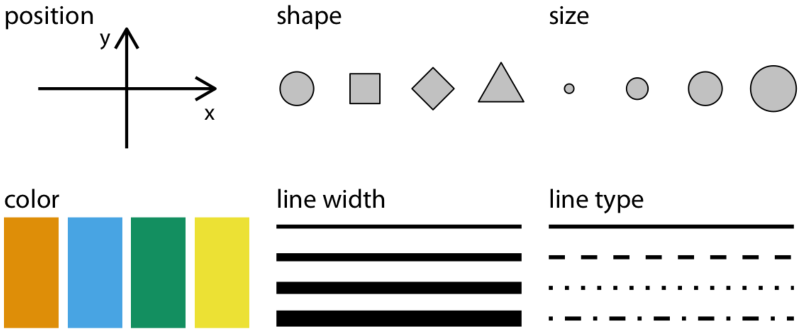
```
# layer 1
ggplot(data = dataset,
       mapping = aes(x = x_variable,
                     y = y_variable)) +
# layer 2
  geom_<type of plot>() +
# layer 3
  labs(x = "x name", y = "y name")
```

Basic Components of `ggplot` (Layers)

- Layer 1: `ggplot()`
  - bring in data
  - how data are mapped to visual information, the "aesthetic mapping"
- Layer 2: `geom_xxx()`
  - say how to visualize the aesthetic mapping – bars, dots, etc.
- Layer 3: `labs()`
  - communicate main point + what the aesthetics represent
  - title, legend, axes-labels, etc

An "aesthetic" is a visual property of the objects in your plot

- We *map* data to aesthetics
  - `col1` will be represented by color
  - `col2` will be represented by the x-position



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

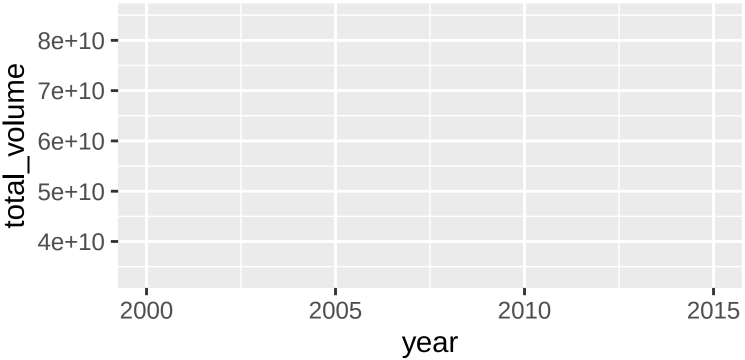
```r
# Layer 1, data frame
ggplot(data = annual_sales)
```

Layer 1: adding an aesthetic mapping

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

- map each row of the data (`year`, `total_volume`) to the (`x`,`y`)
- automatically picks scale for axes

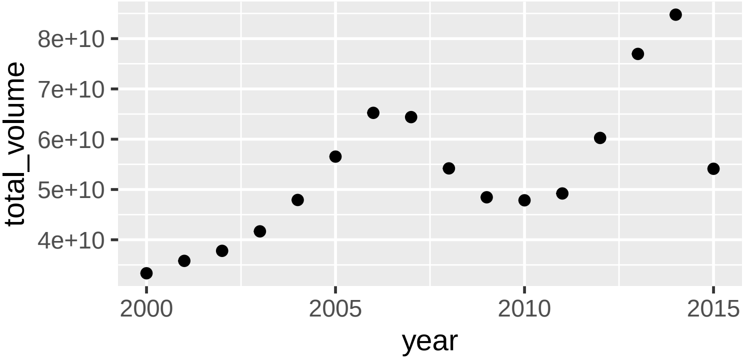
```
ggplot(data = annual_sales,
       mapping = aes(x = year, y = total_volume))
```



Layer 2: visualizing the mapping with `geom_point()`

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

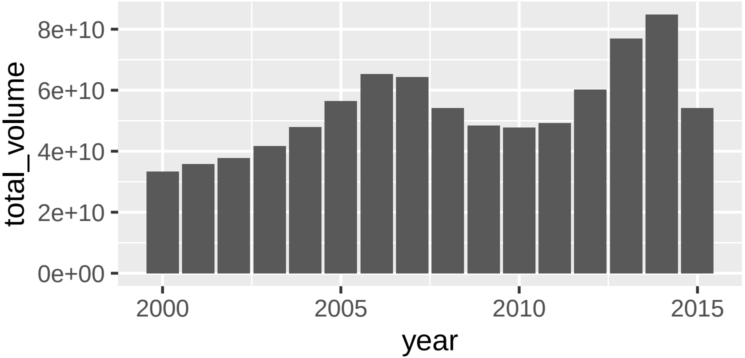
```
ggplot(data = annual_sales,
       mapping = aes(x = year, y = total_volume)) +
  geom_point()
```



Layer 2: visualizing the mapping with geom_col()

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

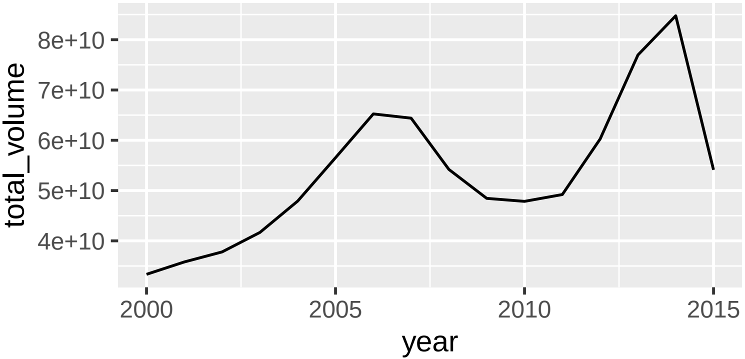
```
ggplot(data = annual_sales,
       mapping = aes(x = year, y = total_volume)) +
  geom_col()
```



Layer 2: visualizing the mapping with geom_line

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

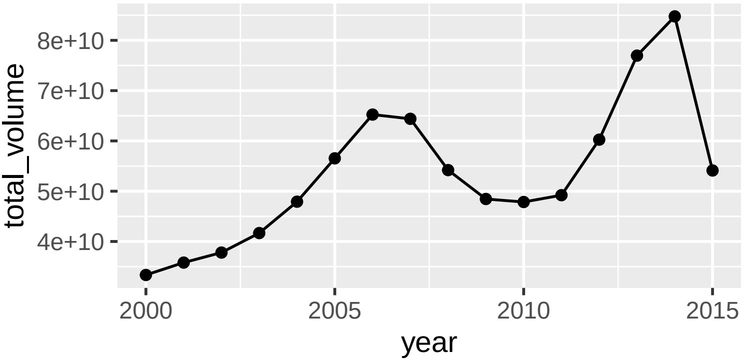
```
ggplot(data = annual_sales,
       mapping = aes(x = year, y = total_volume)) +
  geom_line()
```



Layer 2: visualizing the mapping with geom

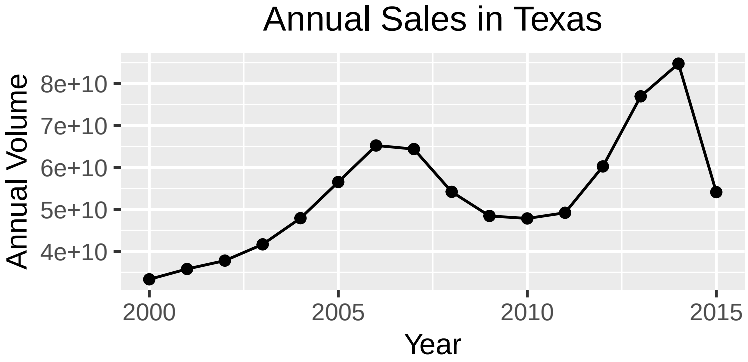
The data can be visualized with different geoms that can be composed (+) together:

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



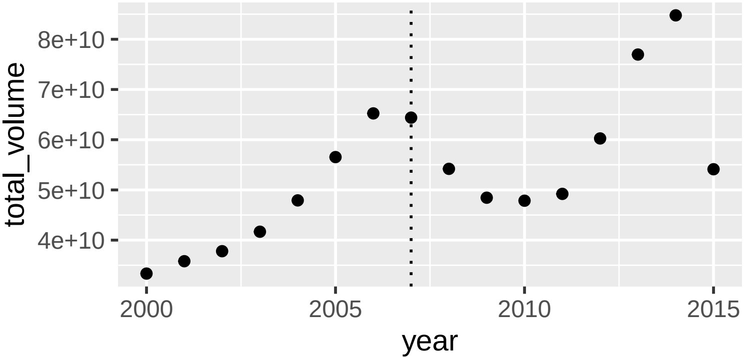
Layer 3: Adding labels makes the plot more readable:

```
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")
```

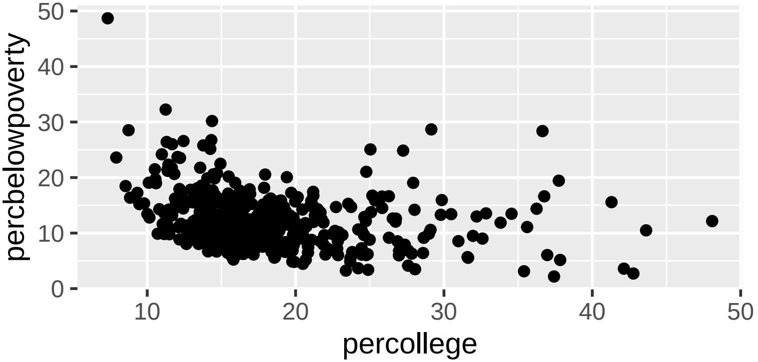


- add horizontal lines with `geom_hline()`
- add any linear fit with `geom_abline()` by providing a slope and intercept

aesthetics beyond the x and y position

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

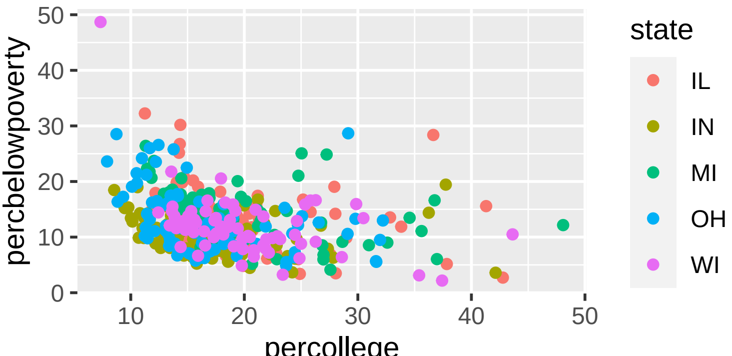
```
midwest |>
   ggplot(aes(x = percollege,
              y = percbelowpoverty)) +
      geom_point()
```



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

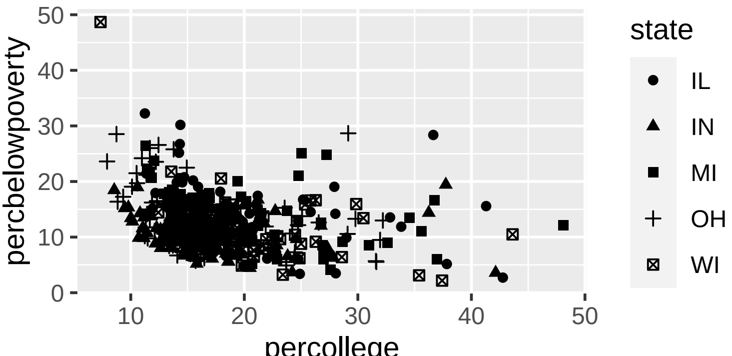
```
midwest |>
  ggplot(aes(x = percollege,
             y = percbelowpoverty,
             color = state)) +
    geom_point()
```



ggplot(): Using shape

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

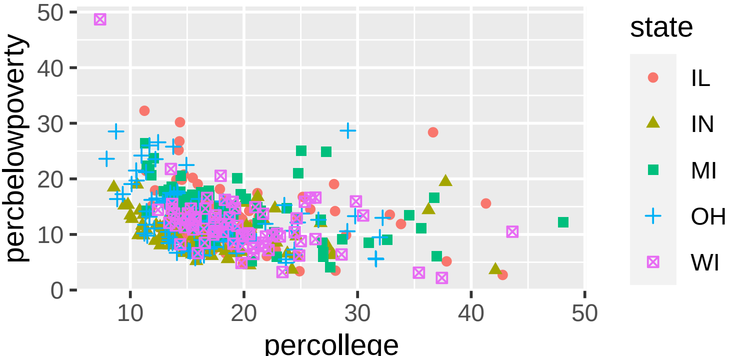
```
midwest |>
  ggplot(aes(x = percollege,
             y = percbelowpoverty,
             shape = state)) +
    geom_point()
```



ggplot(): Using color + shape

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

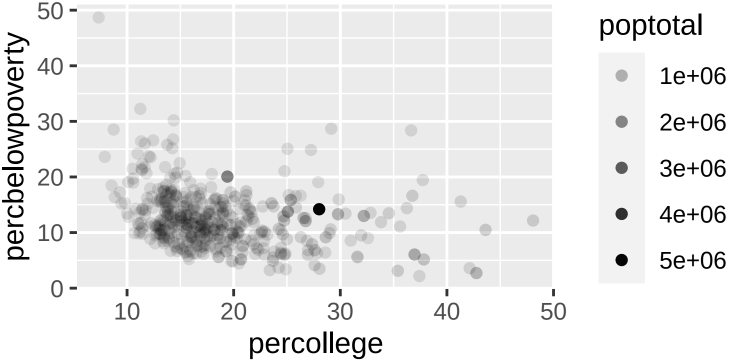
```
midwest |>
  ggplot(aes(x = percollege,
             y = percbelowpoverty,
             color = state,
             shape = state)) +
    geom_point()
```



ggplot(): Using alpha

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

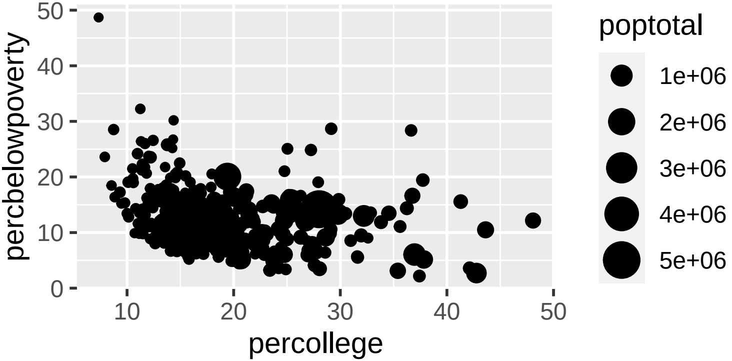
```
midwest |>
  ggplot(aes(x = percollege,
             y = percbelowpoverty,
             alpha = poptotal)) +
    geom_point()
```



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()
```

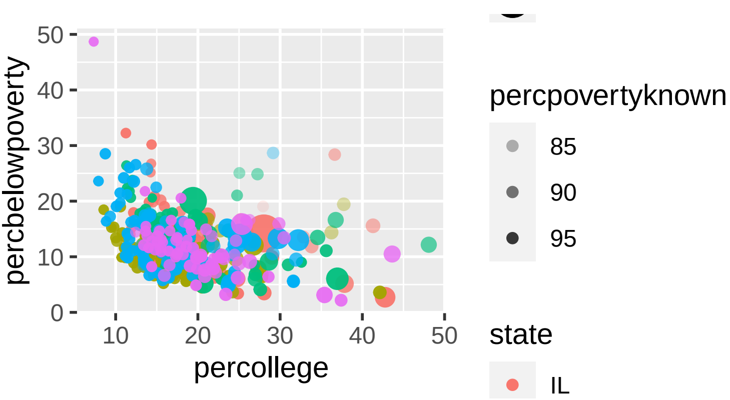


ggplot(): Using multiple aesthetics together

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()
```

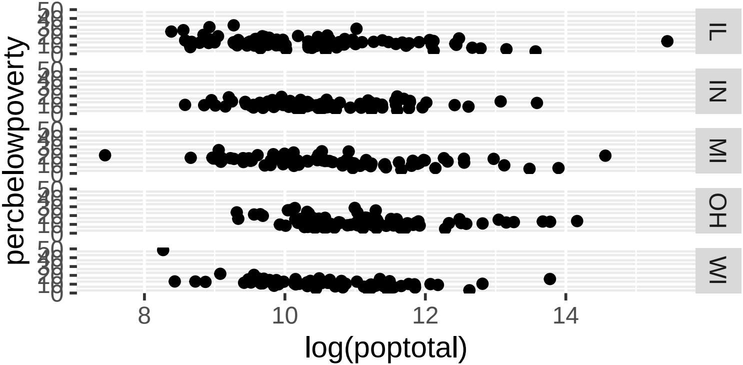
ggplot(): Using multiple aesthetics together



Facets: a tool to explore multidimensional data

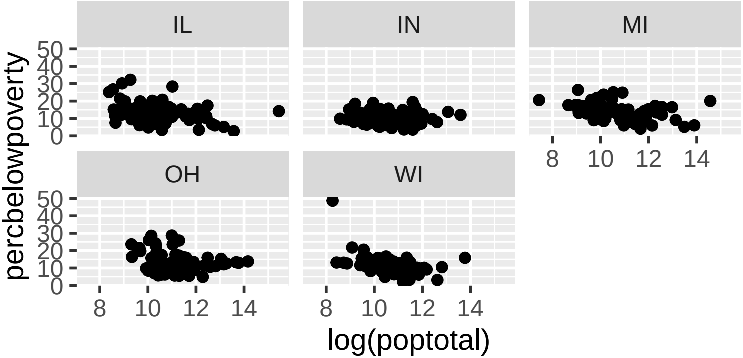
```
midwest |>
   ggplot(aes(x = log(poptotal),
              y = percbelowpoverty)) +
      geom_point() +
      facet_grid(vars(state))
```

Facets: a tool to explore multidimensional data



Facets: a tool to explore multidimensional data

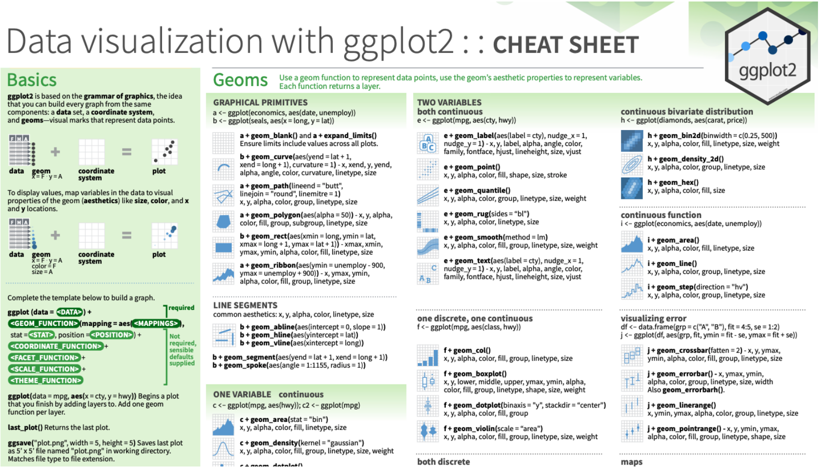
```
midwest |>
  ggplot(aes(x = log(poptotal),
             y = percbelowpoverty)) +
    geom_point() +
    facet_wrap(vars(state))
```



ggplot(): Using aesthetics to explore data

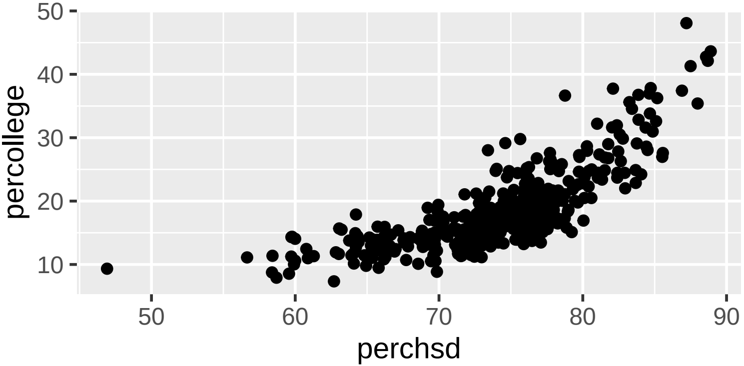
Different geoms have specific aesthetics that go with them.

- the ggplot cheat sheet shows all the geoms with their associated aesthetics



Try it yourself: Plot 1

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

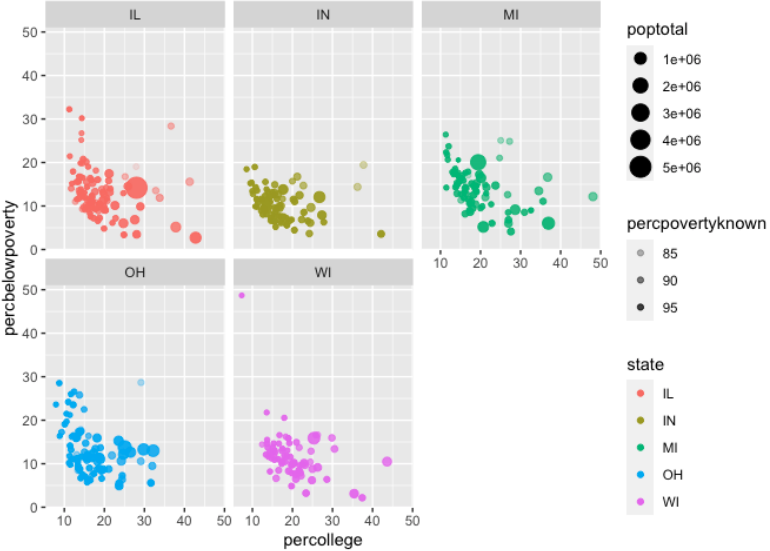


Try it yourself: Plot 1

```
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):



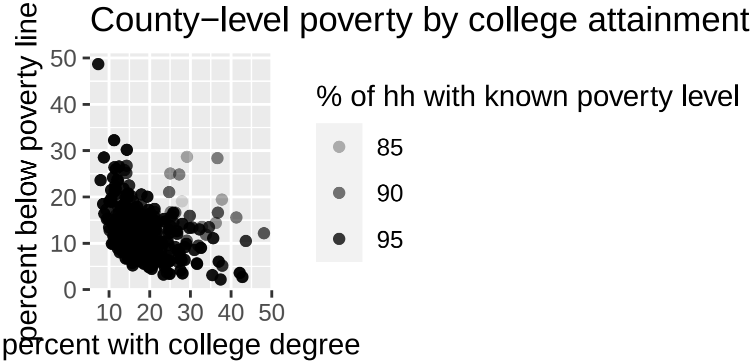
Try it yourself: Plot 2

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

Write your own labels with labs()

```
midwest |>
  ggplot(aes(x = percollege,
             y = percbelowpoverty,
             alpha = percpovertyknown)) +
  geom_point() +
  labs(title = "County-level poverty by college attainment'
       alpha = "% of hh with known poverty level",
       y = "percent below poverty line",
       x = "percent with college degree")
```

Write your own labels with `labs()`



Thinking about how underlying data maps to aesthetics

discrete vs continuous data

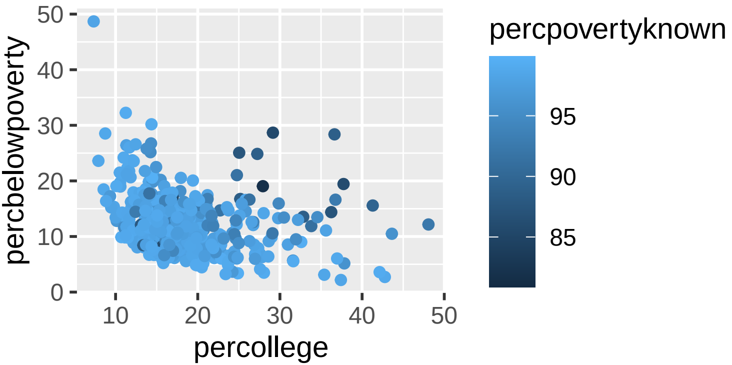
Here discrete and continuous have different meaning than in math

aes	discrete	continuous
	limited number of classes	unlimited number of values
	i.e. < 10 values	any number of values
	usually `chr` or `lgl`	numeric
`x`, `y`	yes	yes
`color`, `fill`	yes	yes
`shape`	yes (6 or fewer)	no
`size`, `alpha`	not advised	yes
`facet`	yes	not advised

- If your "discrete" data is numeric, use `as.character()` or `as_factor()` to enforce the decision.

color can be continuous (but default not great)

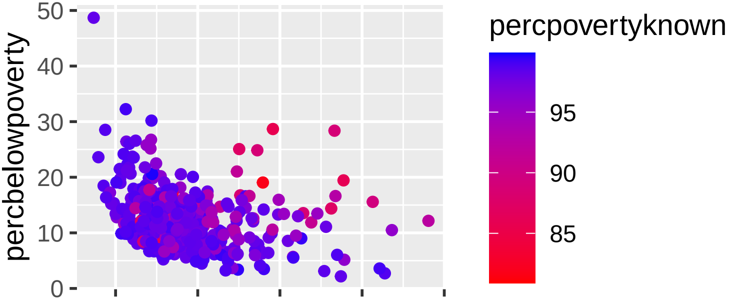
```
midwest |>
    ggplot(aes(x = percollege,
               y = percbelowpoverty,
               color = percpovertyknown)) +
  geom_point()
```



color can be continuous

- lots of tools to explore, need to think about what is being communicated

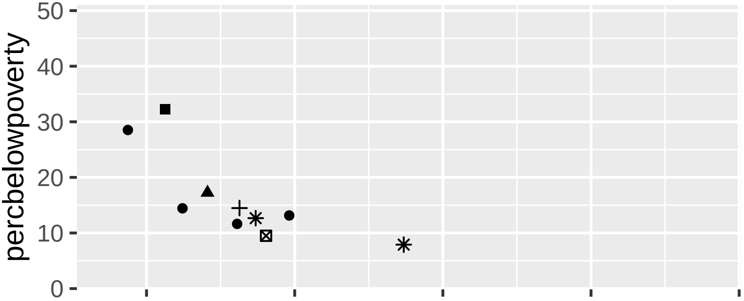
```
midwest |>
    ggplot(aes(x = percollege,
               y = percbelowpoverty,
               color = percpovertyknown)) +
  geom_point() +
  scale_colour_gradientn(colors=c("red", "blue"))
```



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

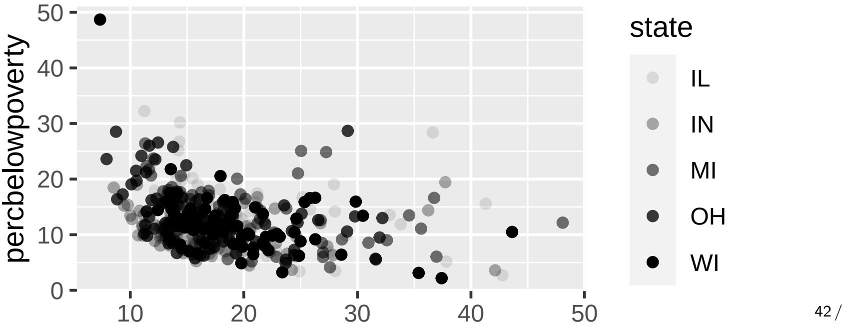
```
# Bad example
midwest |>
    ggplot(aes(x = percollege,
               y = percbelowpoverty,
               shape = county)) +
  geom_point() +
  # legend off, otherwise it overwhelms
  theme(legend.position = "none")
```



alpha and size can be misleading with discrete data

```
# Bad example
midwest |>
    ggplot(aes(x = percollege,
               y = percbelowpoverty,
               alpha = state)) +
  geom_point()
```

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



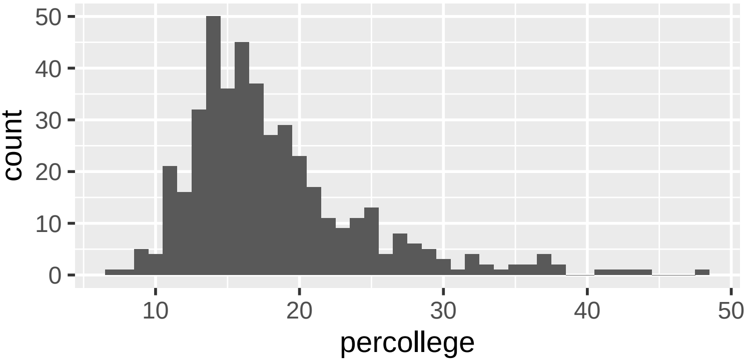
Thinking about how underlying data maps to type of figure (`geom`)

Type of figures

1. Distribution of **univariate (single variable)**
   - boxplot, 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

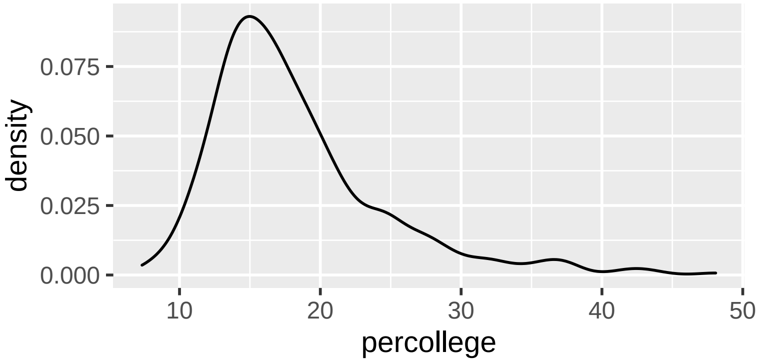
Univariate: histogram

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



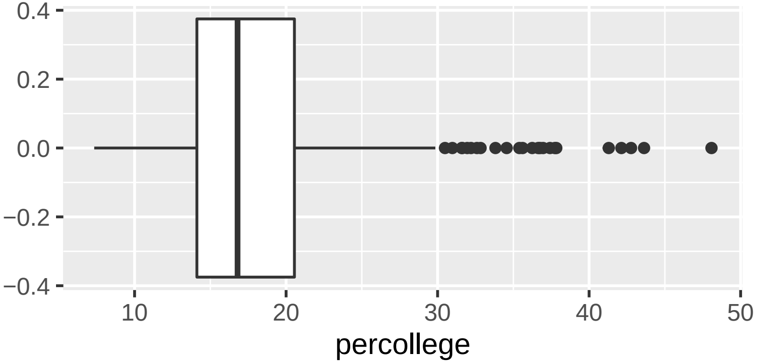
Univariate: density

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



Univariate: box plots

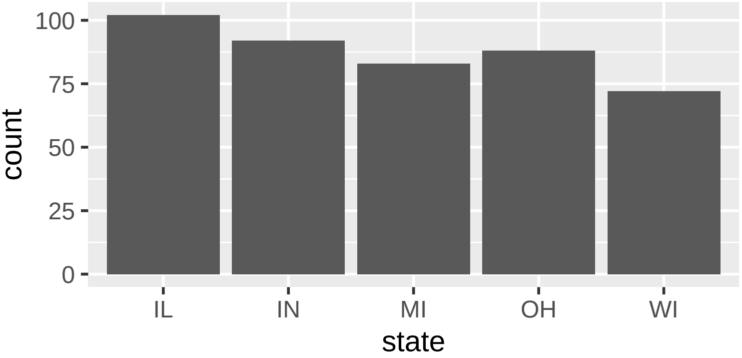
```
midwest |>
  ggplot(aes(x = percollege)) +
    geom_boxplot()
```



Univariate: bar plot

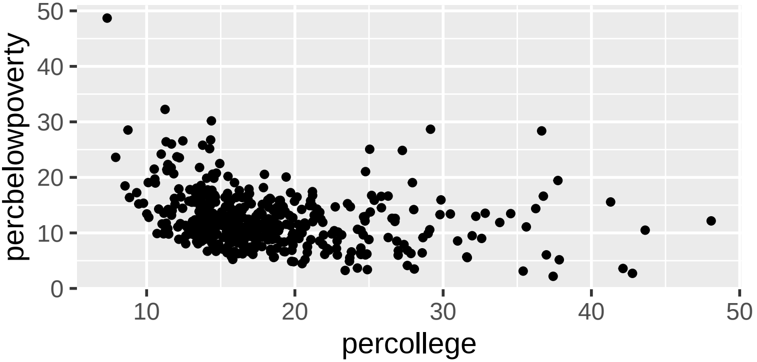
Like `geom_histogram()`, `geom_bar()` counts your data.

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



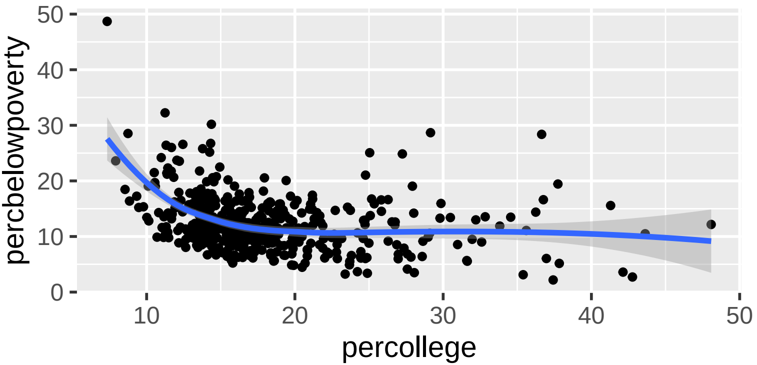
Bivariate: scatter plot

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



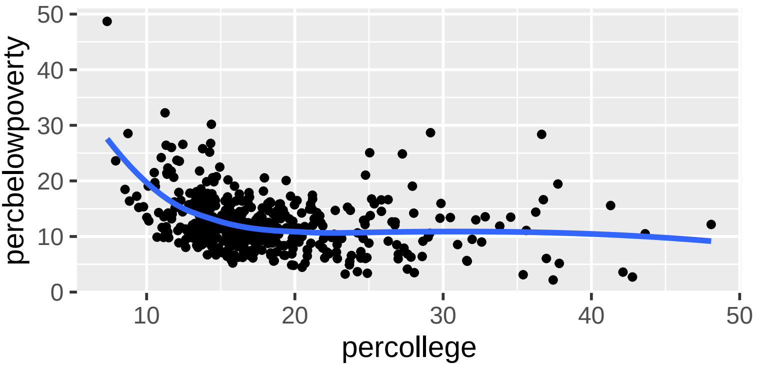
Bivariate: scatter + smooth Line plot

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



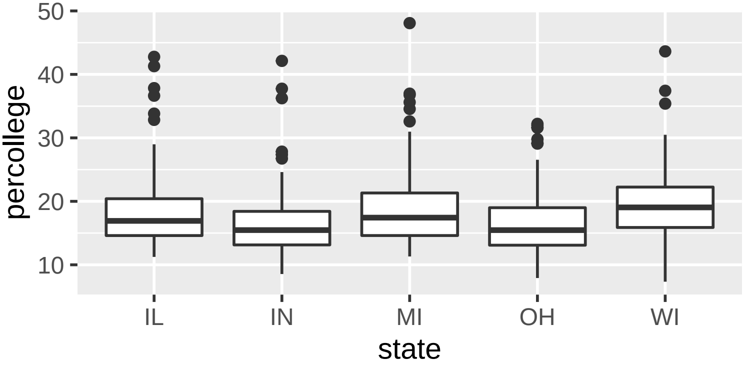
Bivariate: scatter + smooth Line plot

```
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: There are many ways you can visualize your data!

- Visualizations provide insights into 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

Next steps

Labs

- Today: Data visualization with `ggplot` (may run into tomorrow)

I can produce basic plots to explore and communicate about data

Lecture

- Data analysis with grouped data

Appendix: Some graphs you made along the way

- `annual_sales`
- Distributions
- Grouped bar graph
- Faceted bar graph

Today's data: annual_sales

```
annual_sales <-
  txhousing |>
  group_by(year) |>
  summarize(total_volume = sum(volume, na.rm = TRUE))
```

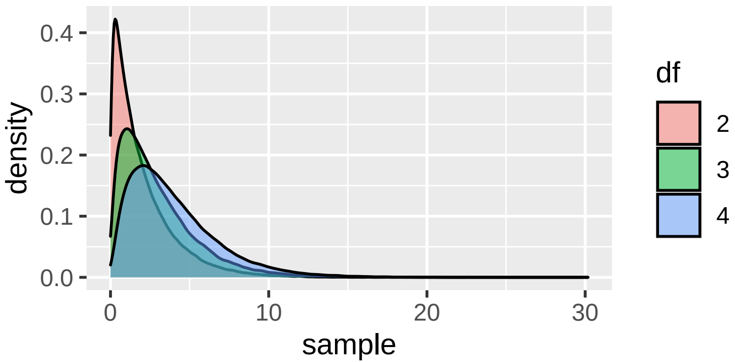
Appendix: distributions

- `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.

```
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

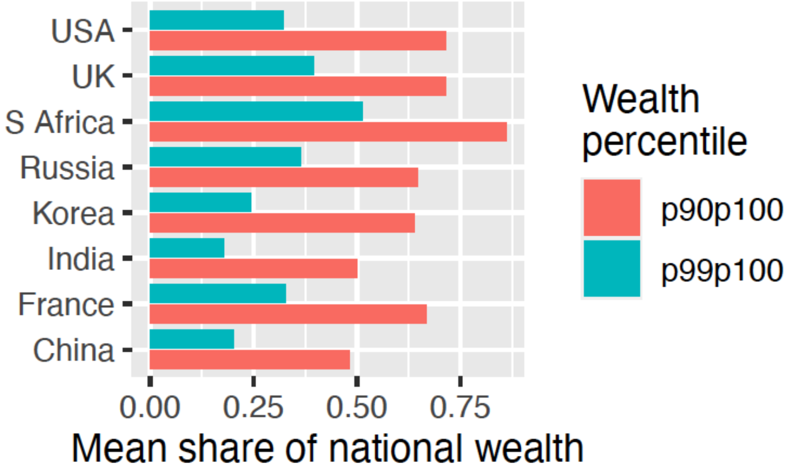


Appendix: grouped bar graph

- `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.

```
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