Coding Lab: Visualizing data with `ggplot2`

Ari Anisfeld

Summer 2020

How to use `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

There are whole books on how to use `ggplot`. This is a quick introduction!

Understanding `ggplot()`

By itself, `ggplot()` tells R to prepare to make a plot.

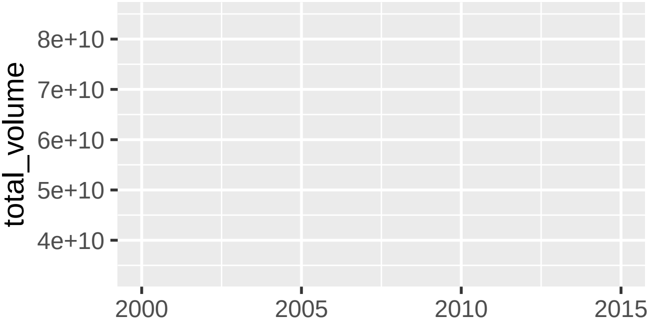
```r
texas_annual_sales <-
  texas_housing_data %>%
  group_by(year) %>%
  summarize(total_volume = sum(volume, na.rm = TRUE))

ggplot(data = texas_annual_sales)
```

Adding a `mapping`

Adding `mapping = aes()` says how the data will map to "aesthetics".

- e.g. tell R to make x-axis `year` and y-axis `total_volume`.
- Each row of the data has (`year`, `total_volume`).
  - R will map that to the coordinate pair (`x`,`y`) .
  - Look at the data before moving on!

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



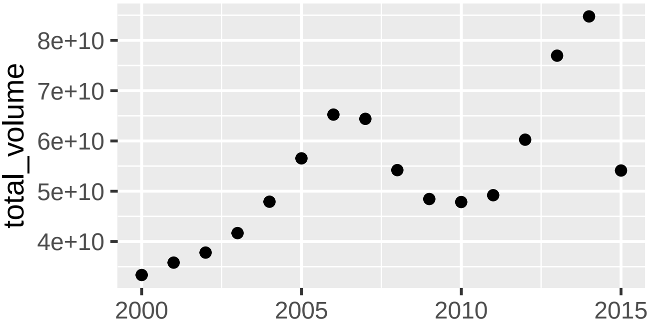
Visualizing the mapping with a geom

`geom_<name>` tells R what type of visualization to produce.

Here we see points.

- Each row of the data has (`year`, `total_volume`).
- R will map that to the coordinate pair (`x`,`y`).

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

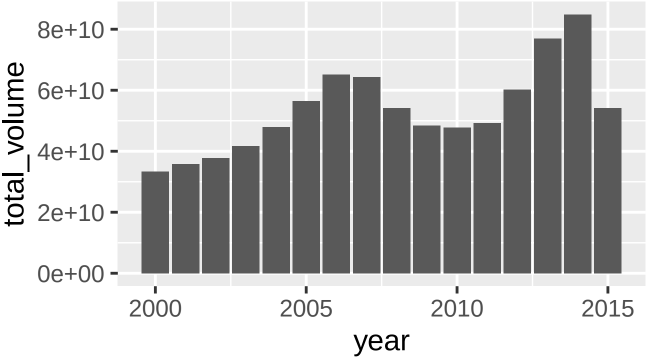


Visualizing the mapping with a geom

Here we see bars.

- Each row of the data has (year, total_volume).
- R will map that to the coordinate pair (x,y)

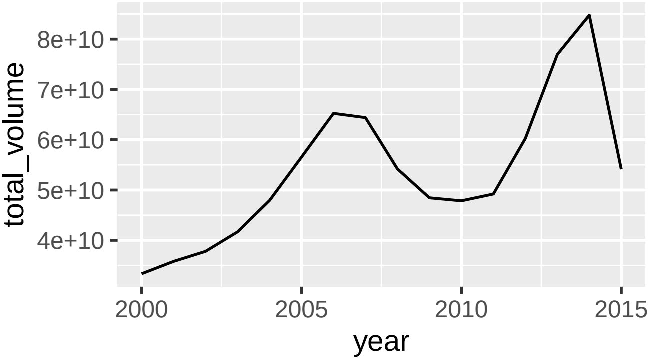
```
ggplot(data = texas_annual_sales,
       mapping = aes(x = year, y = total_volume)) +
  geom_col()
```



Visualizing the mapping with a geom

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

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



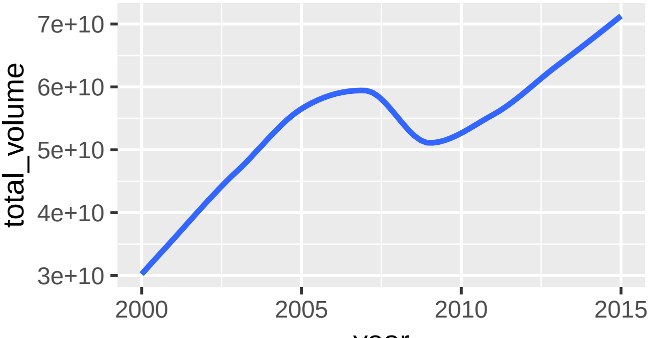
Visualizing the mapping with a geom

Here we see a smooth line. R does a statistical transformation!

- Now R doesn't visualize the mapping (`year`, `total_volume`) to each (x,y) pair
- Instead it fits a model to the (x,y) and then plots the "smooth" line

```
ggplot(data = texas_annual_sales,
       mapping = aes(x = year, y = total_volume)) +
  geom_smooth(se = FALSE)
```

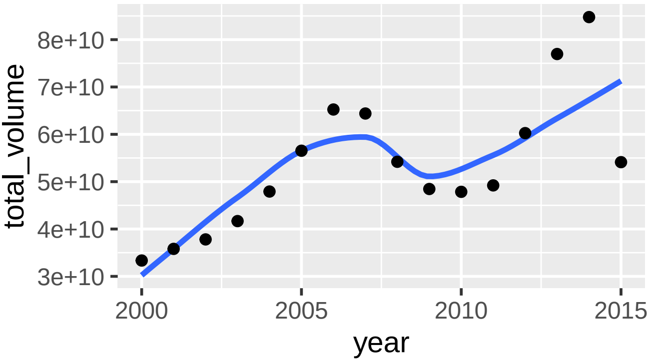
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Visualizing the mapping with a geom

We can overlay several geom.

```
ggplot(data = texas_annual_sales,
       mapping = aes(x = year, y = total_volume)) +
  geom_smooth(se = FALSE) +
  geom_point()
```



Visualizing the mapping with a `geom`

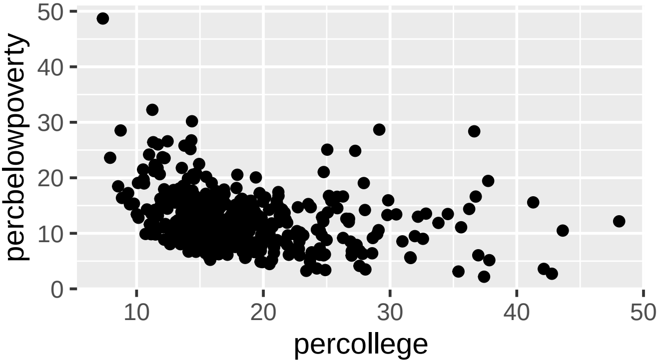
- We saw that we can visualize a relationship between two variables mapping data to `x` and `y`
- The data can be visualized with different geoms that can be composed (+) together.
- We can even calculate new variables with statistics and plot those on the fly.

Next: Now we'll look at aesthetics that go beyond `x` and `y` axes.

Using aesthetics to explore data.

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

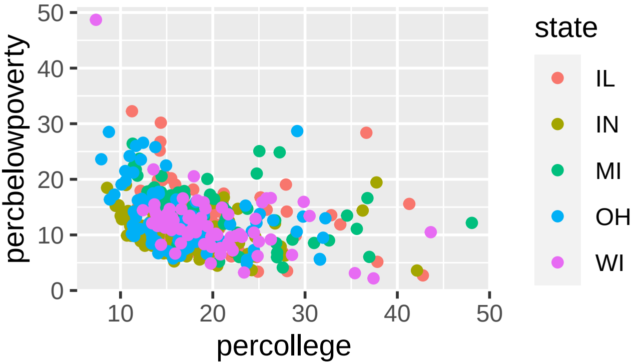
```
midwest %>%
  ggplot(aes(x = percollege,
             y = percbelowpoverty)) +
    geom_point()
```



Using aesthetics to explore data.

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

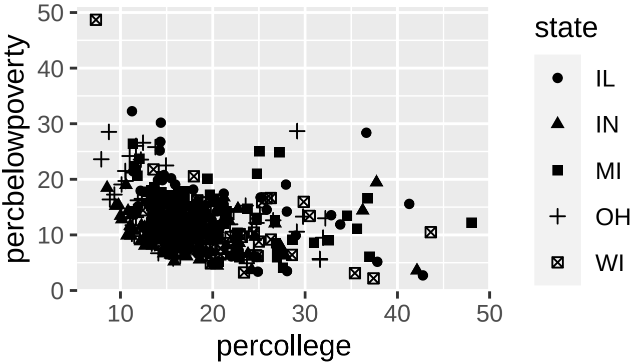
```
midwest %>%
  ggplot(aes(x = percollege,
             y = percbelowpoverty,
             color = state)) +
    geom_point()
```



Using aesthetics to explore data.

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

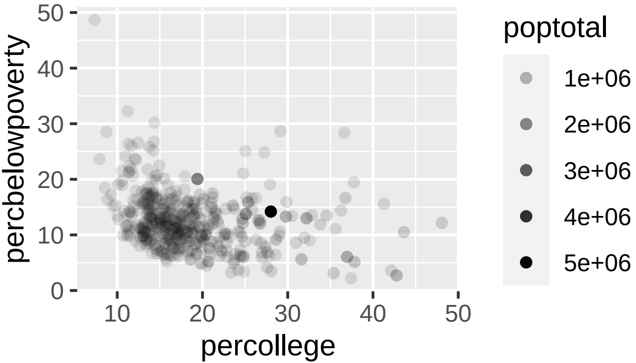
```
midwest %>%
   ggplot(aes(x = percollege,
              y = percbelowpoverty,
              shape = state)) +
     geom_point()
```



Using aesthetics to explore data.

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

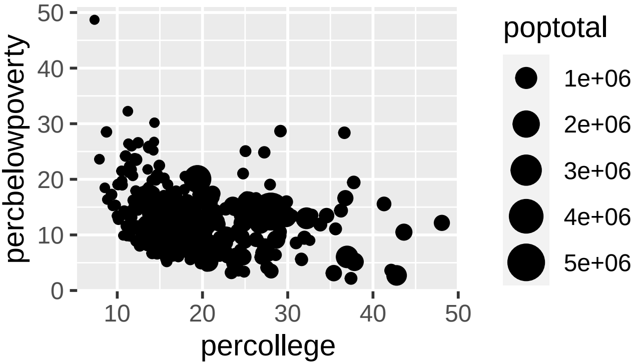
```
midwest %>%
   ggplot(aes(x = percollege,
              y = percbelowpoverty,
              alpha = poptotal)) +
     geom_point()
```



Using aesthetics to explore data.

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

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

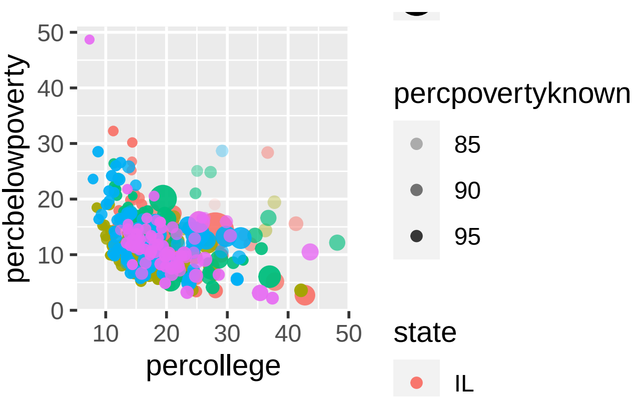


Using aesthetics to explore data.

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()
```

Using aesthetics to explore data.



Using aesthetics to explore data

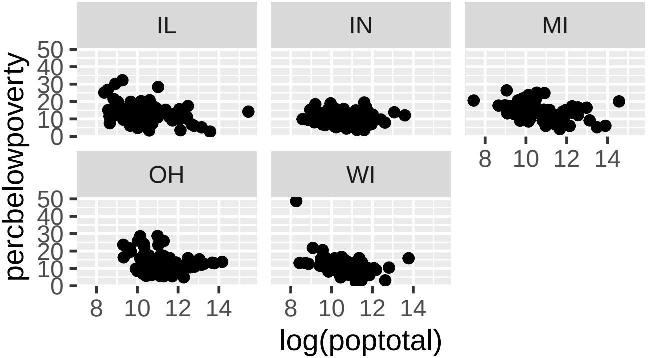
Different geoms have specific aesthetics that go with them.

- use `?` to see which aesthetics a geom accepts (e.g `?geom_point`)
    - the bold aesthetics are required.
- the ggplot cheatsheet shows all the `geoms` with their associated aesthetics

Facets

Facets provide an additional tool to explore multidimenional data

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

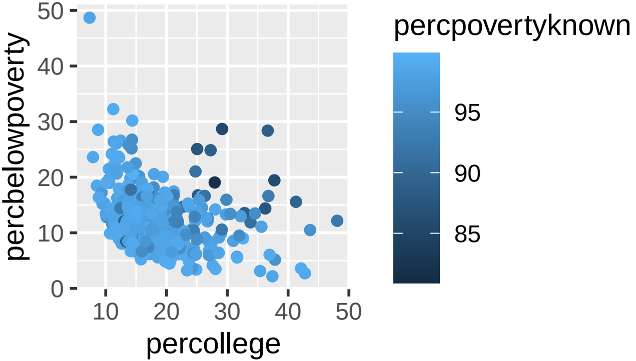


discrete

vs continuous data

color can be continuous

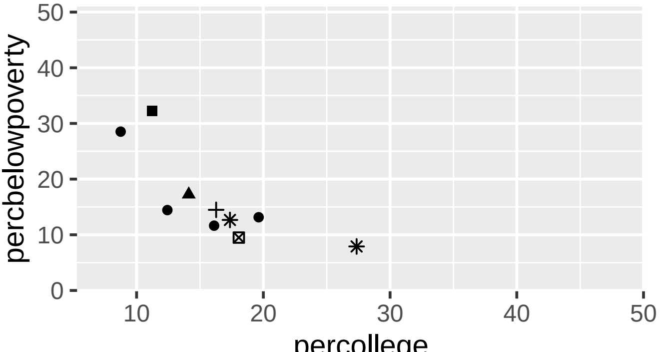
```
midwest %>%
    ggplot(aes(x = percollege,
               y = percbelowpoverty,
               color = percpovertyknown)) +
  geom_point()
```



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

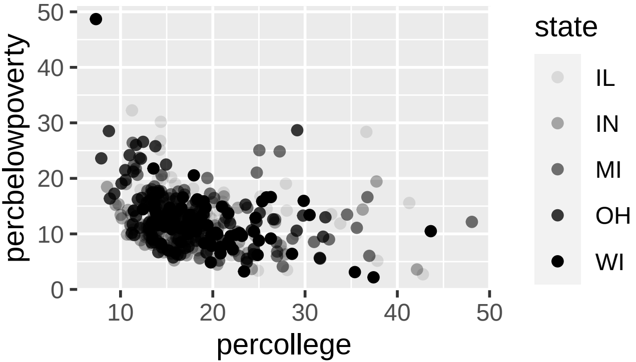
```
midwest %>%
    ggplot(aes(x = percollege,
               y = percbelowpoverty,
               shape = county)) +
  geom_point() +
  # legend off, otherwise it overwhelms
  theme(legend.position = "none")
```



alpha and size can be misleading with discrete data

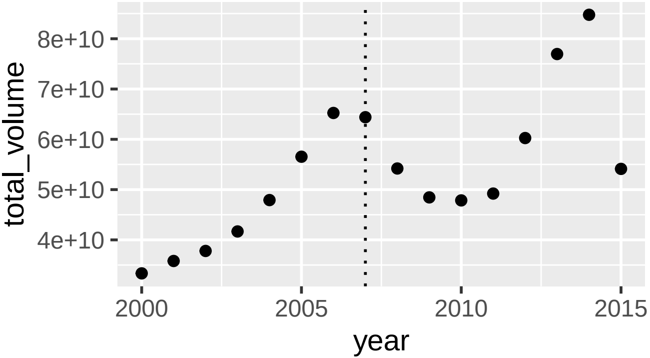
```
midwest %>%
    ggplot(aes(x = percollege,
               y = percbelowpoverty,
               alpha = state)) +
  geom_point()
```

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



Adding vertical lines

```
texas_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 using `geom_abline()` by providing a slope

Key take aways

- `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.
  - We need to be aware of 'continuous' vs. 'discrete' variables.
- Then, we use `geoms` to create a visualization based on the mapping.
  - Again we need to be aware of 'continuous' vs. 'discrete' variables.
- Making quick plots helps us understand data and makes us aware of data issues

Resources: R for Data Science chap. 3 (r4ds.had.co.nz); RStudio's ggplot cheatsheet.

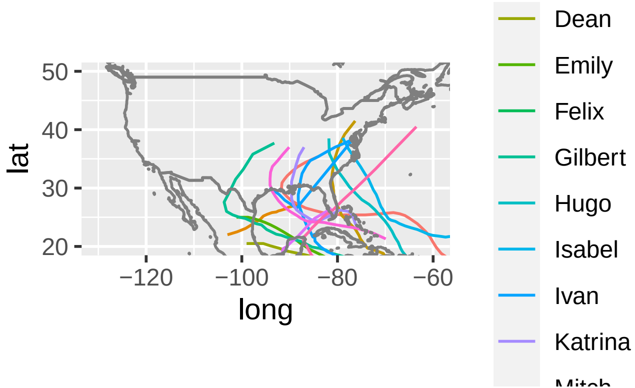
Appendix: Some graphs you made along the way

lab 0: a map

`geom_path` is like `geom_line`, but connects (x, y) pairs in the order they appear in the data set.

```
storms %>%
  group_by(name, year) %>%
  filter(max(category) == 5) %>%
ggplot(aes(x = long, y = lat, color = name)) +
  geom_path() +
  borders("world") +
  coord_quickmap(xlim = c(-130, -60), ylim = c(20, 50))
```

lab 0: a map

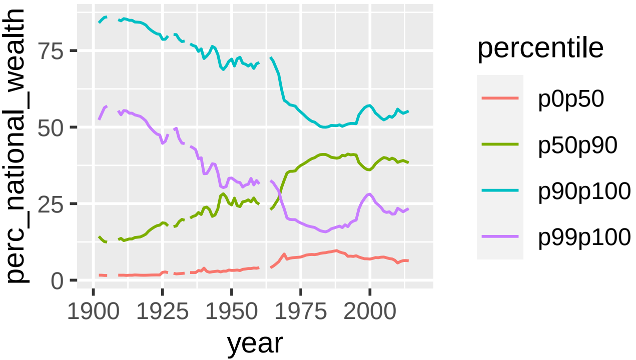


50
40
30
20
lat
−120
−100
−80
−60
long
Dean
Emily
Felix
Gilbert
Hugo
Isabel
Ivan
Katrina

lab 1: a line plot

```
french_data <-
  wid_data %>%
  filter(type == "Net personal wealth",
         country == "France") %>%
  mutate(perc_national_wealth = value * 100)

french_data %>%
  ggplot(aes(y = perc_national_wealth,
             x = year,
             color = percentile)) +
  geom_line()
```

lab 1: a line plot

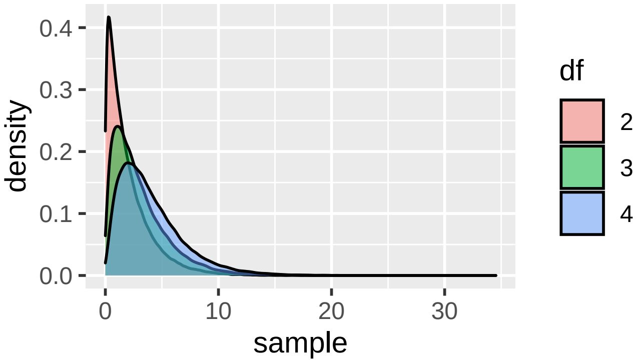


lab 2: distributions

- `geom_density()` only requires an `x` asthetic 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")
```

lab 2: distributions

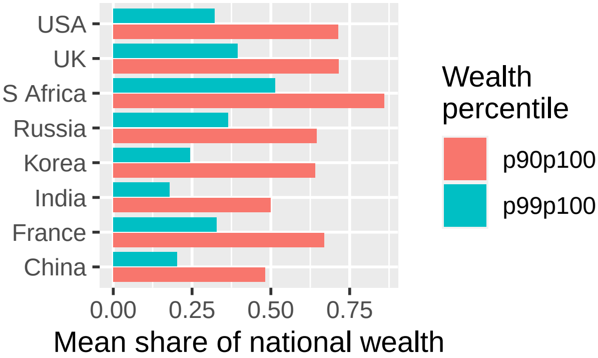


lab 4: grouped bar graphs

- `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")
```

lab 4: grouped bar graphs



lab 4: 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))
```

lab 4: faceted bar graph