

Week 5 - Introduction to Data Viz

Alex Lishinski
September 14, 2021

Welcome!

Welcome to *week 5*!

Record the meeting

Review of last week's class

Last week we discussed wrangling and tidying data:

1. Reshaping data
 - `pivot_wide()` and `pivot_long()`
2. Joining data
 - `left_join()`, `inner_join()`, and others
3. Grouped data operations with dplyr
 - `group_by()` and `summarize()`

Review of last week's class

Reading

- From R for Data Science: <https://r4ds.had.co.nz/tidy-data.html>
- tidy data:
 - every variable has its own column
 - every observation has its own row
 - every value has its own cell
- tidy data makes it easier to use similar tools (even with very different datasets and types of data)
- tidy data works well with R

Review of last week's class

TB cases

- Where is the year variable represented?
- Where is the cases variable represented?
- How many observations does each row represent?

```
library(tidyverse)
```

```
## — Attaching packages ————— tidyverse 1.3.1 —
```

```
## ✓ tibble 3.1.3      ✓ forcats 0.5.1  
## ✓ purrr 0.3.4
```

```
## — Conflicts ————— tidyverse_conflicts() —
```

```
## x tidyr::extract() masks magrittr::extract()  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()  
## x purrr::set_names() masks magrittr::set_names()
```

```
table4a
```

```
## # A tibble: 3 × 3  
##   country    `1999` `2000`  
## * <chr>    <int> <int>  
## 1 Afghanistan    745   2666  
## 2 Brazil      37737  80488  
## 3 China      212258 213766
```

This week's topics

Overview

1. Introduction to data viz
2. A bit more tidying data

We are by no means done with the data tidying functions we discussed last week!

1. Intro to Data Viz

Outline

- A. Why visualize data?
- B. How can we visualize data in R?
- C. And, how can we make our visualizations aesthetically pleasing?

1A: Why visualize data?

One answer:

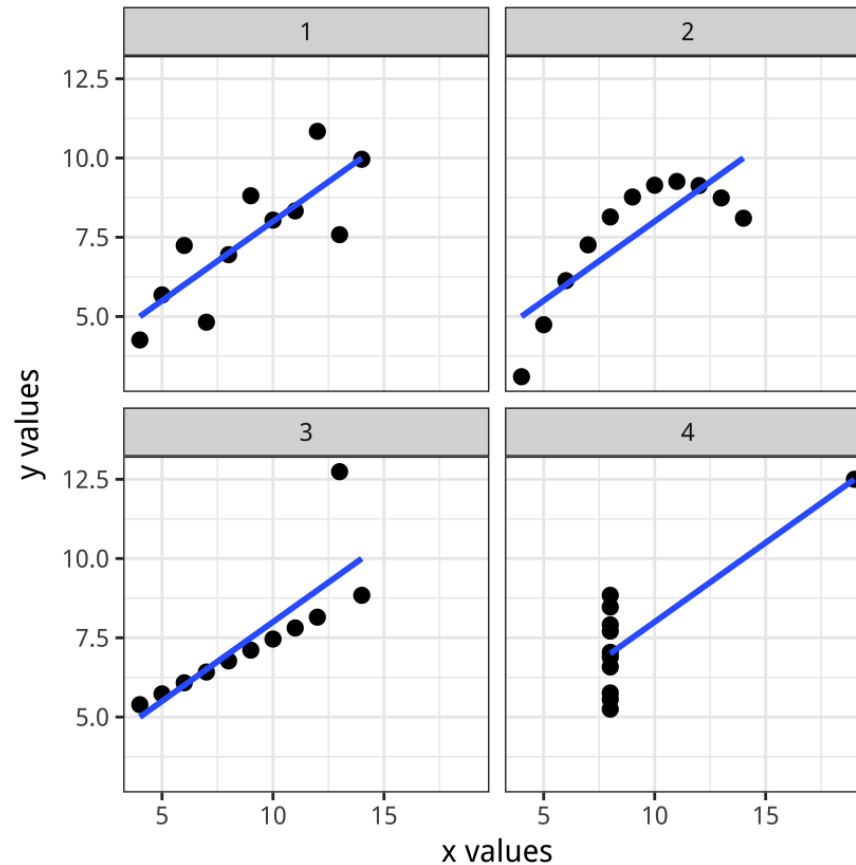
"You should look at your data." ([Healy, 2018](#))

To elaborate on this:

- Visualizations allow to *understand the structure and nature of your data*, and to begin to understand what might relate to what else
- Just like we want to be constantly looking at our data in its spreadsheet/table/data frame format (e.g., `str()`, `glimpse()`, and `View()`), visualizing our data can help us to make sure our data contains what we think it does-and it can alert us to when it does not

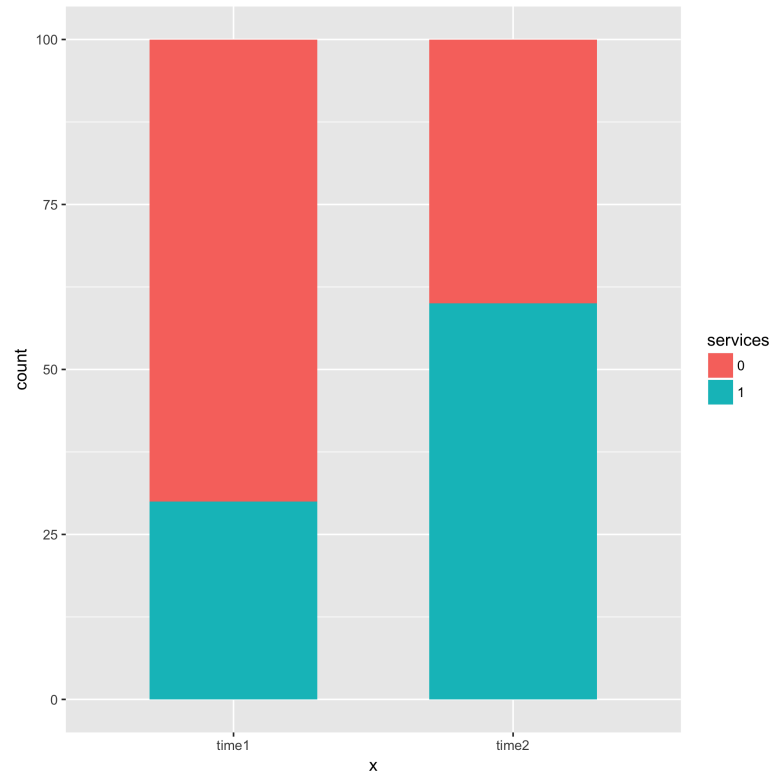
1A: Why visualize data?

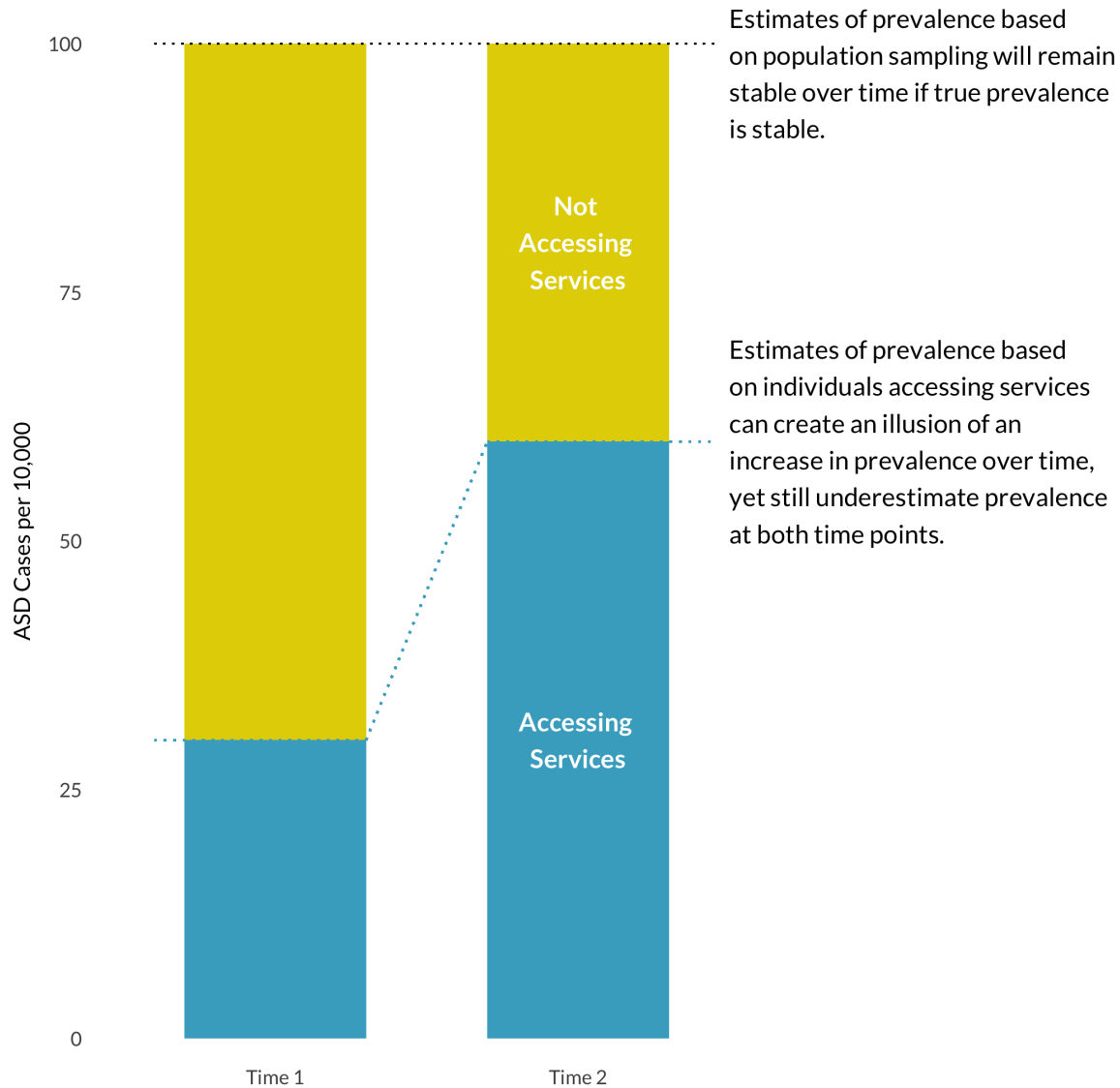
These four different data sets have the same correlation (type `anscombe` in R to view the data), but are very different



1A: Why visualize data

Another reason to visualize data is to *communicate with others*; you can use visualizations to communicate your findings or results. In example:





1B: How to visualize data

One way to think about visualizing data is in terms of the *type* of visualization you create:

- Histogram
- Density plot
- Scatter plot
- Bar chart
- Pie chart (gasp!)
- Time series plot/line chart

1B: How to visualize data

Another way to think about visualizing data is in terms of the elements that make up a plot.

The ***grammar of graphics*** ([Wickham, 2010](#), [Wilkinson, 2012](#)) has a particular answer to the question of what a plot includes:

Why a grammar of graphics?

- gain insight into complex figures
- reveal deeper relationships between what may appear to be unrelated visualizations
- more flexibly and creatively visualize data--including in ways that do not fit well into one type of plot
- suggest what makes a good figure

1B: How to visualize data?

One view of visualizations is that they consist of four components:

1. Data
2. One or more geometric objects (shape, point, line, etc.)
3. A mapping between variables in the data and the geometric objects and their characteristics (including their size and color)
4. A theme

1B: How to visualize data?

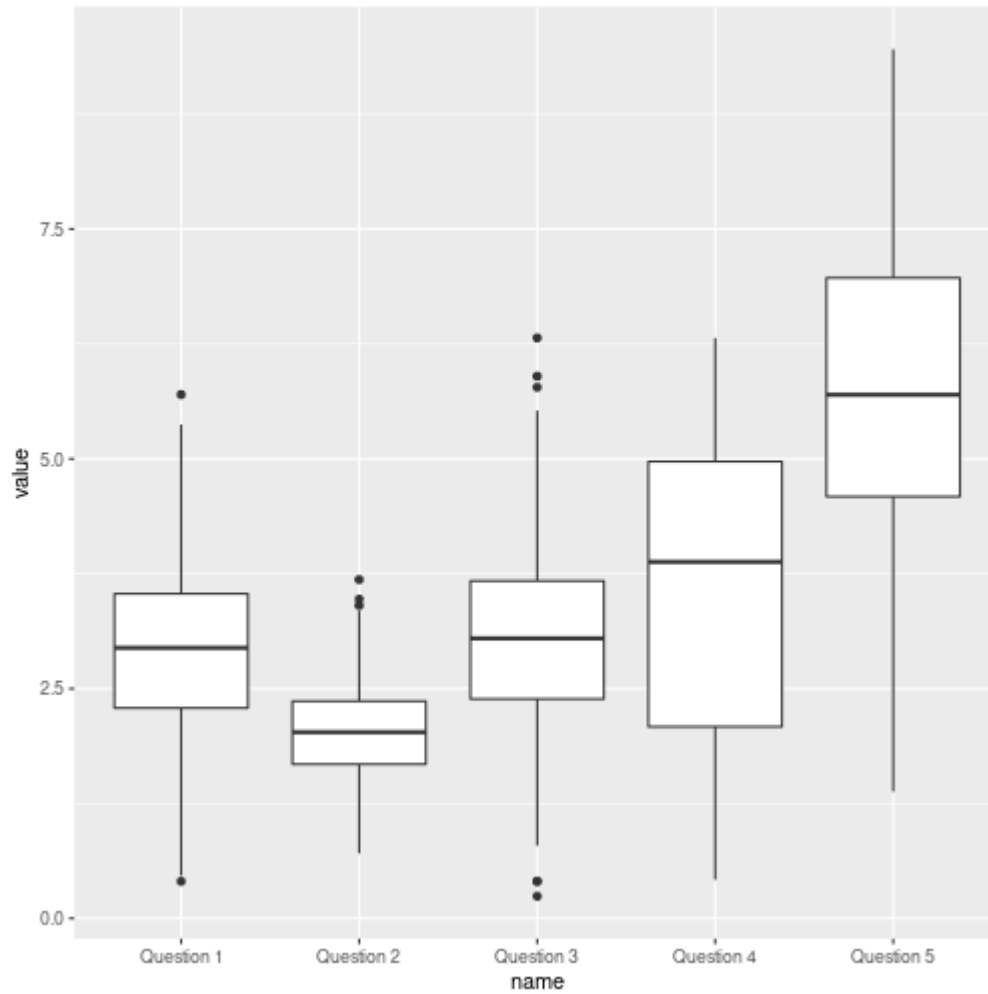
Let's see how this might appear:

```
data
```

```
## # A tibble: 1,618 × 2
##   name      value
##   <chr>    <dbl>
## 1 Question 1  2.90
## 2 Question 1  3.46
## 3 Question 1  2.80
## 4 Question 1  2.64
## 5 Question 1  3.06
## 6 Question 1  3.34
## 7 Question 1  4.36
## 8 Question 1  2.43
## 9 Question 1  2.58
## 10 Question 1  3.32
## # ... with 1,608 more rows
```

```
data %>%
  ggplot(aes(x = name, y = value)) +
  geom_boxplot()
```

1B: How to visualize data?



1B: How to visualize data

- The previous slide contained a potentially *useful* plot
- However, we might be able to improve both its interpretability and its aesthetic

Data Visualization with ggplot2 : : CHEAT SHEET



Basics

Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables.

ggplot2

ggplot is based on the grammar of **graphics**, the idea that you could build every graph from the same components: **a data set**, a **coordinate system**, and **geoms**—visual marks that represent data points.

data + geom = coord

coord + plot = 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 = coord

coord + plot = plot

Complete the table below to build a graph.

| | |
|---|---|
| ggplot(data = DATA) + | required |
| GEOM FUNCTION(mapping = aes(MAPPINGS)) | |
| coord + "STAT", position = POSITION() | |
| COORDINATE FUNCTION() | |
| SCALE FUNCTION() | not required, aesthetic defaults supplied |
| 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.

@geom_name(mapping = data, geom)

geom_c = cty, y = hwy, data = mpg, geom = "point" Creates a complete plot with given data, geom, and mapping. Supplies many useful defaults.

last_plot() Returns the last plot

ggsave("plot.png", width = 5, height = 5) Saves last plot as x × 5 file named "plot.png" in working directory. Matches file type for extension.

GRAPHICAL PRIMITIVES

a = **ggplot**(economics, aes(date, unemployment))
b = **ggplot**(diamonds, aes(x = long, y = lat))

(useful for blank limits)

b = **geom_curve**(aes(long = lat + 1, size = long * curvature * 2)) ~ x, yend, ymid, alpha, angle, color, curvature, linetype, size

a = **geom_path**(linetype = "butt", linetype = "round", linewidth = 1)
x, y, alpha, color, group, linetype, size

a = **geom_polygon**(aes(x = group))
x, y, alpha, color, fill, group, linetype, size

b = **geom_rect**(aes(xmin = long - ymid * sin(x), xmax = long + ymid * sin(x), ymin = x, ymax = x + alpha, fill = alpha))
x, y, alpha, color, fill, group, linetype, size

a = **geom_ribbon**(aes(min = ymid * sin(x), max = ymid * sin(x) + 900)) ~ x, ymid, ymin, alpha, color, fill, group, linetype, size

LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

b = **geom_abline**(aes(intercept = slope - 1))

b = **geom_hline**(aes(yintercept = lat))

b = **geom_vline**(aes(xintercept = long))

b = **geom_segment**(aes(yend = lat1, xend = long1))

b = **geom_spath**(aes(x = 1:155, yend = lat))

ONE VARIABLE CONTINUOUS

c = **ggplot**(diamonds, aes(x = carat, y = price))

c = **geom_area**(aes(x = "bin", y = price))

c = **geom_bar**(aes(x = "bin", y = price))

c = **geom_density**(kernel = "gaussian", x = "bin", y = price)

c = **geom_dotplot**(aes(x = "bin", y = price))

c = **geom_freqpoly**(aes(x = "bin", y = price))

c = **geom_histogram**(aes(x = "bin", y = price))

c = **geom_jitter**(aes(x = "bin", y = price))

c = **geom_qq**(aes(sample = y, weight = 1))

c = **geom_ridge**(aes(x = "bin", y = price))

discrete

d = **ggplot**(mpg, aes(x))

d = **geom_bar**(aes(x = "bin", y = price))

d = **geom_jitter**(aes(x = "bin", y = price))

TWO VARIABLES

continuous x, continuous y

e = **geom_label**(aes(x = cty, y = hwy, size = 1, nudge_x = 1, check_overlap = TRUE)) ~ x, label, alpha, angle, color, family, fontface, hjust, linetype, size, vjust

e = **geom_liner**(aes(x = cty, y = hwy, size = 1, nudge_x = 1, check_overlap = TRUE)) ~ x, label, alpha, angle, color, family, fontface, hjust, linetype, size, vjust

e = **geom_point**(aes(x = cty, y = hwy, size = 1, nudge_x = 1, check_overlap = TRUE)) ~ x, label, alpha, angle, color, family, fontface, hjust, linetype, size, vjust

e = **geom_quantile**(aes(x = cty, y = hwy, size = 1, nudge_x = 1, check_overlap = TRUE)) ~ x, label, alpha, angle, color, family, fontface, hjust, linetype, size, vjust

e = **geom_rug**(aes(x = cty, y = hwy, size = 1, nudge_x = 1, check_overlap = TRUE)) ~ x, label, alpha, angle, color, family, fontface, hjust, linetype, size, vjust

e = **geom_smooth**(aes(x = cty, y = hwy, size = 1, nudge_x = 1, check_overlap = TRUE)) ~ x, label, alpha, angle, color, family, fontface, hjust, linetype, size, vjust

e = **geom_text**(aes(x = cty, y = hwy, size = 1, nudge_x = 1, check_overlap = TRUE)) ~ x, label, alpha, angle, color, family, fontface, hjust, linetype, size, vjust

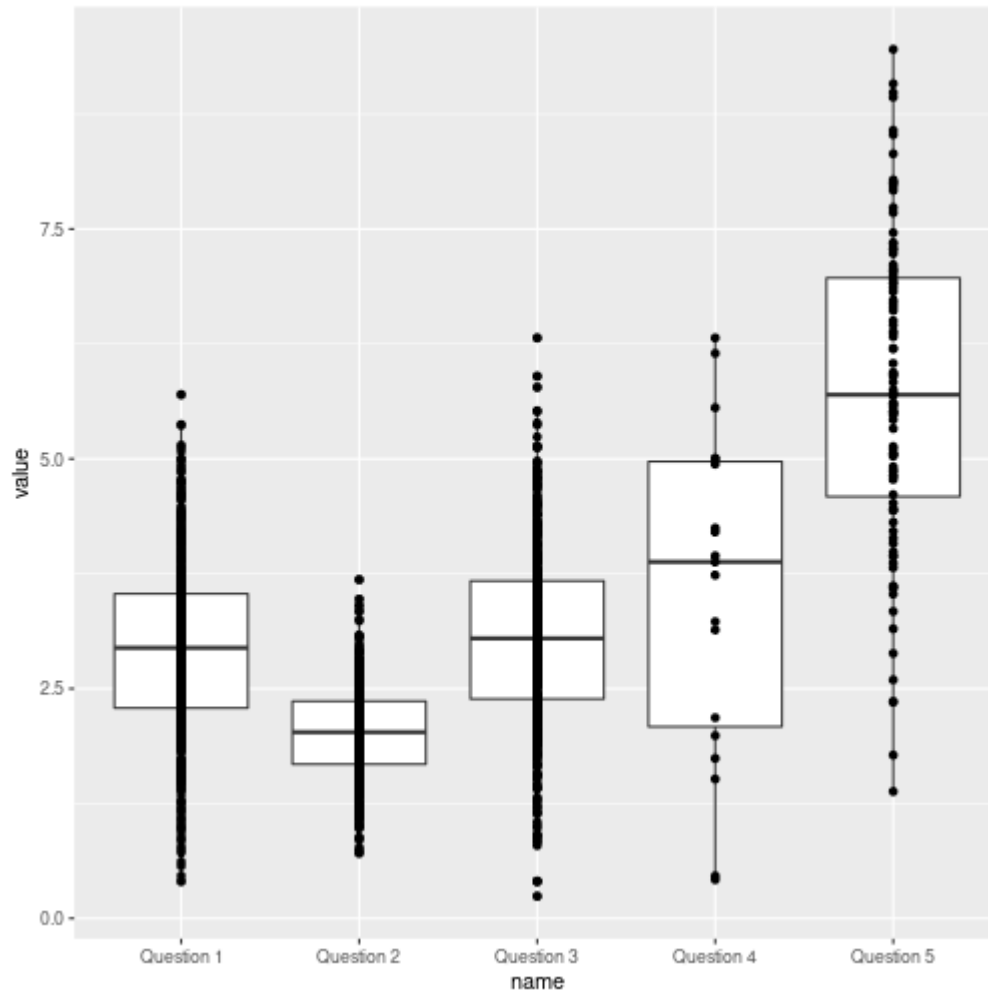
e = **geom_violin**(aes(x = cty, y = hwy, size = 1, nudge_x = 1, check_overlap = TRUE)) ~ x, label, alpha, angle, color, family, fontface, hjust, linetype, size, vjust

e = **geom_xerrorbar**(aes(x = cty, y = hwy, size = 1, nudge_x = 1, check_overlap = TRUE)) ~ x, label, alpha, angle, color, family, fontface, hjust, linetype, size, vjust

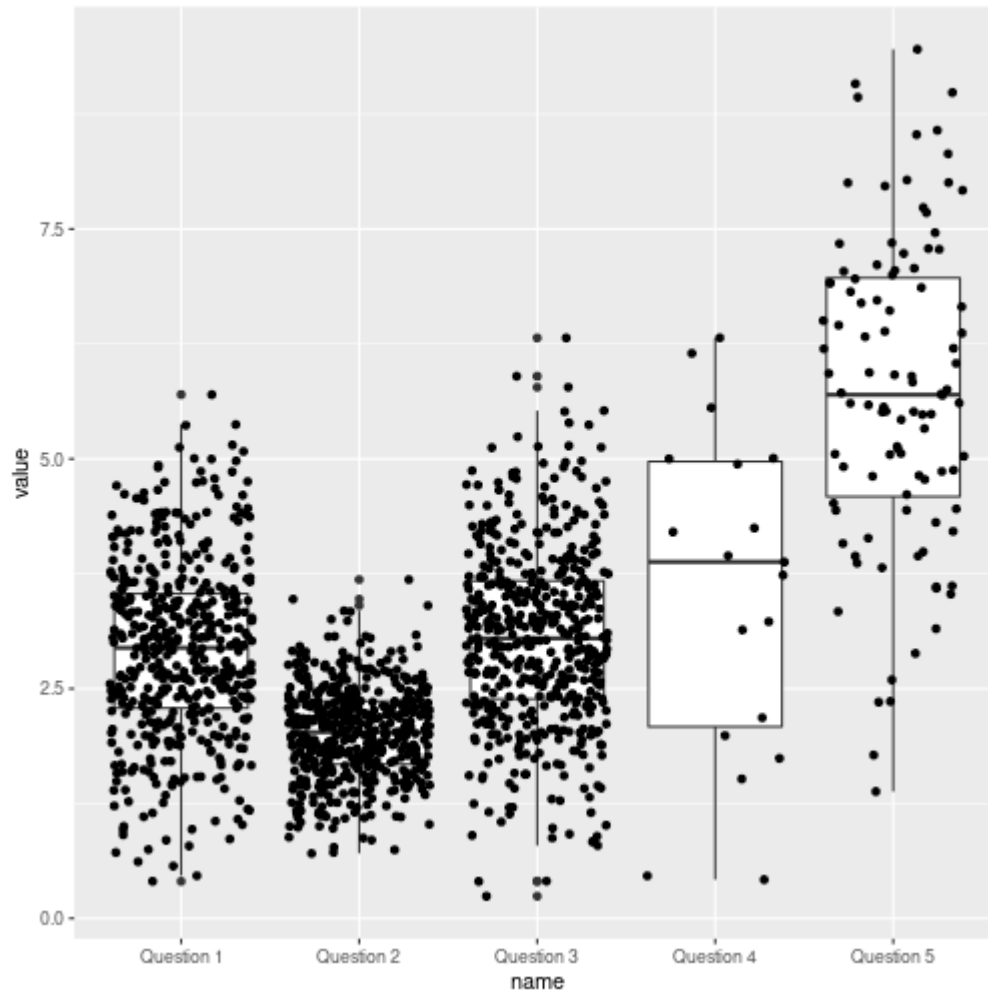
e = **geom_yerrorbar**(aes(x = cty, y = hwy, size = 1, nudge_x = 1, check_overlap = TRUE)) ~ x, label, alpha, angle, color, family, fontface, hjust, linetype, size, vjust

e = **geom_zscore**(aes(x = cty, y = hwy, size = 1, nudge_x = 1, check_overlap = TRUE)) ~ x, label, alpha, angle, color, family, fontface, hjust, linetype, size, vjust

1B: How to visualize data?



1B: How to visualize data?

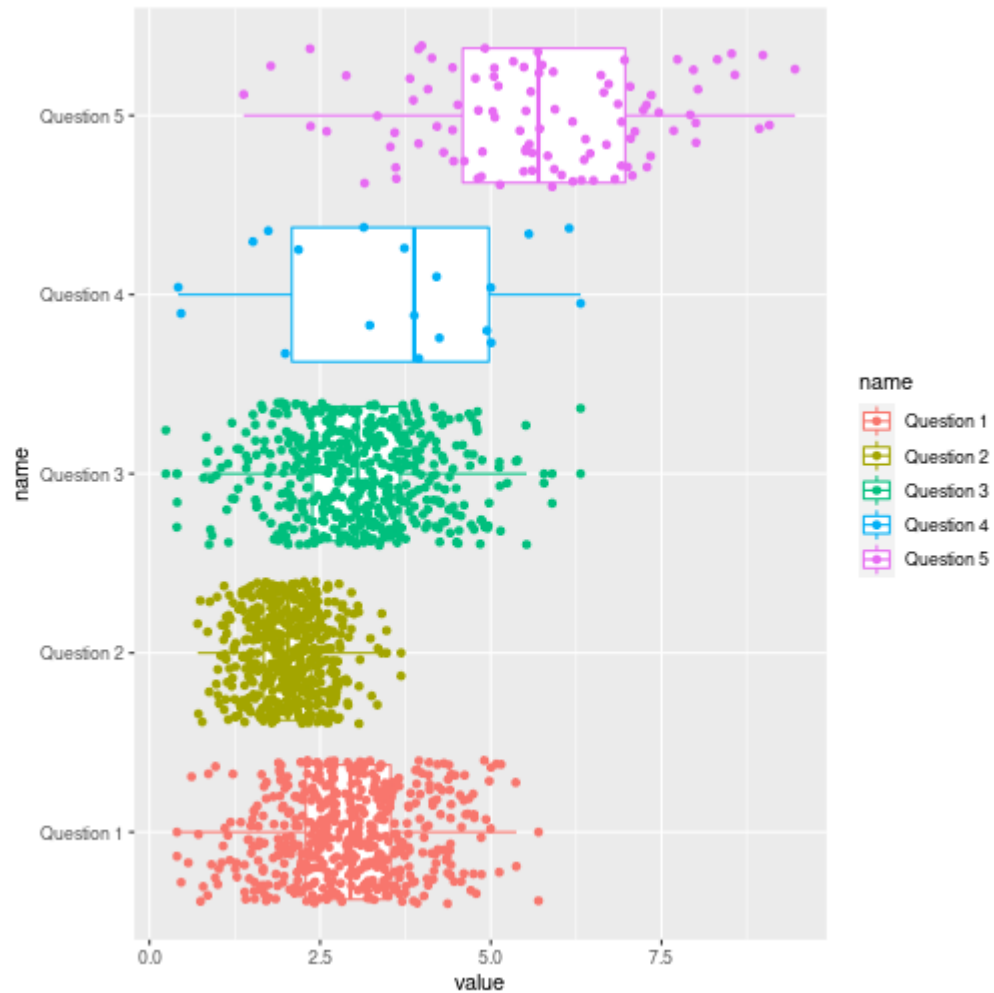


1B: How to visualize data

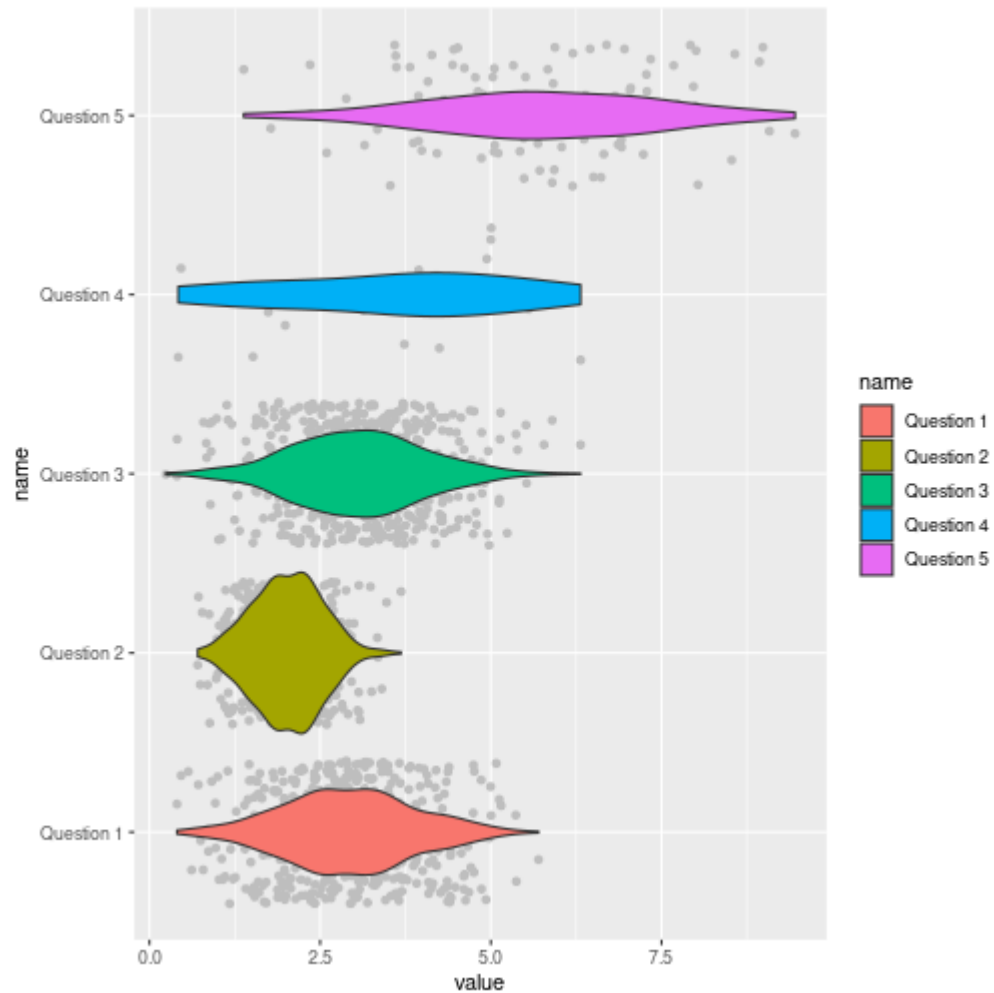
You can create different plots by:

- Changing the aesthetic *mapping* between variables in the data and geometric objects
- Changing the geometric objects

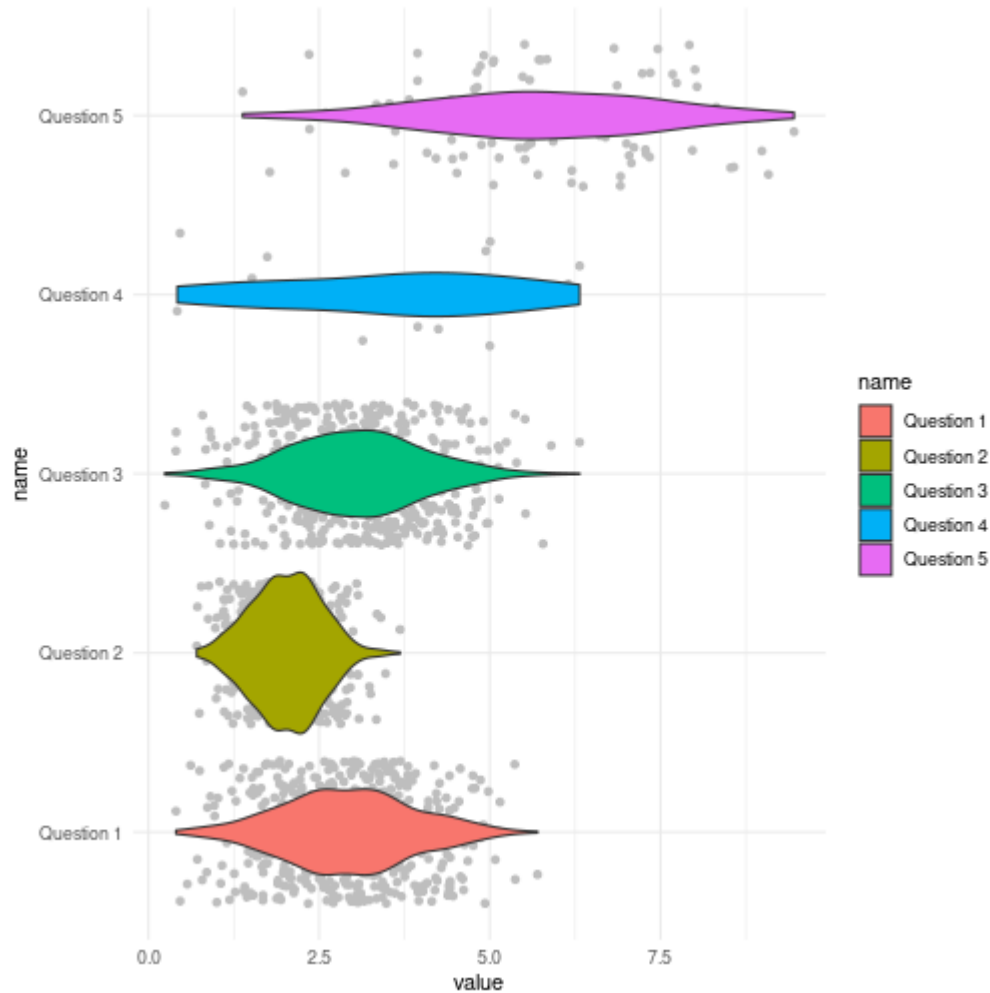
1B: How to visualize data?



1B: How to visualize data?



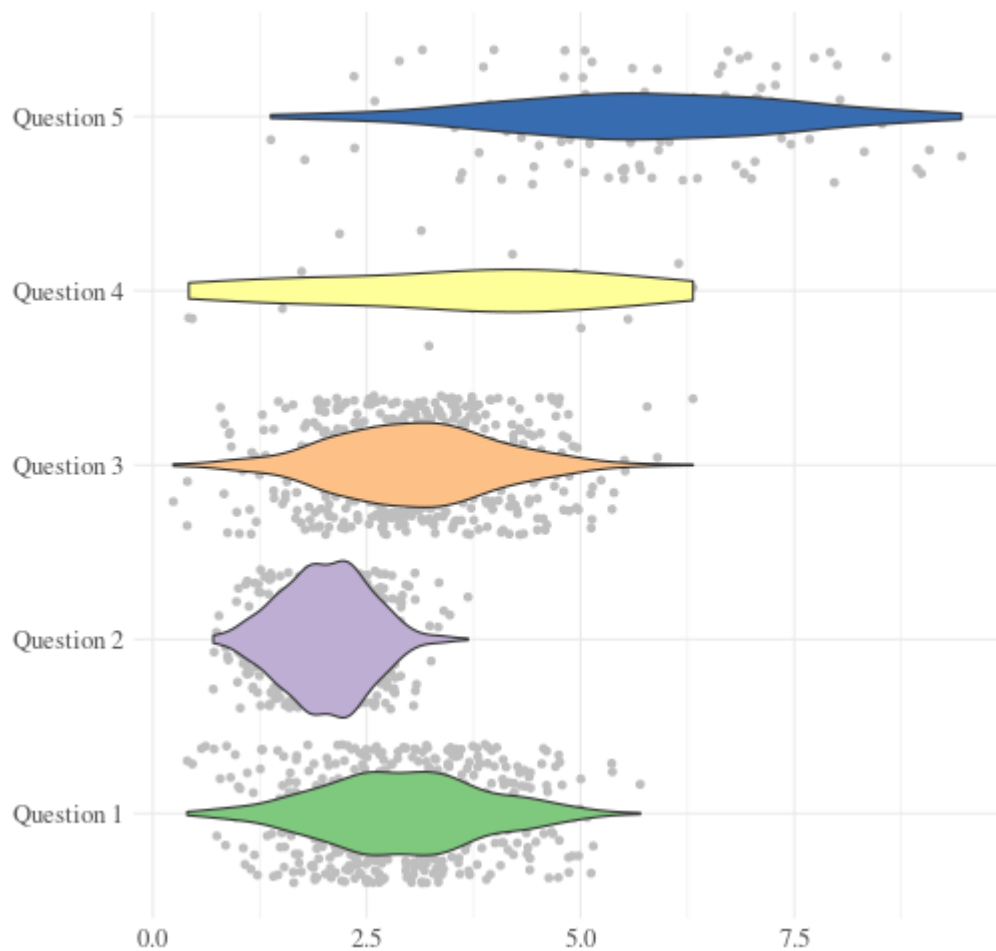
1C: How to make visualizations aesthetically pleasing



Theming and fine-tuning

```
data %>%  
  ggplot(aes(x = value, y = name, fill = name)) +  
  geom_jitter(color = "gray") +  
  geom_violin() +  
  theme_minimal() +  
  scale_fill_brewer("", type = "qual") +  
  ylab(NULL) +  
  xlab(NULL) +  
  theme(text = element_text(size = 16, family = "Times"),  
        legend.position = "none") +  
  ggtitle("Distributions for the Five Questions")
```


Distributions for the Five Questions



2: How does tidying data relate to data viz?

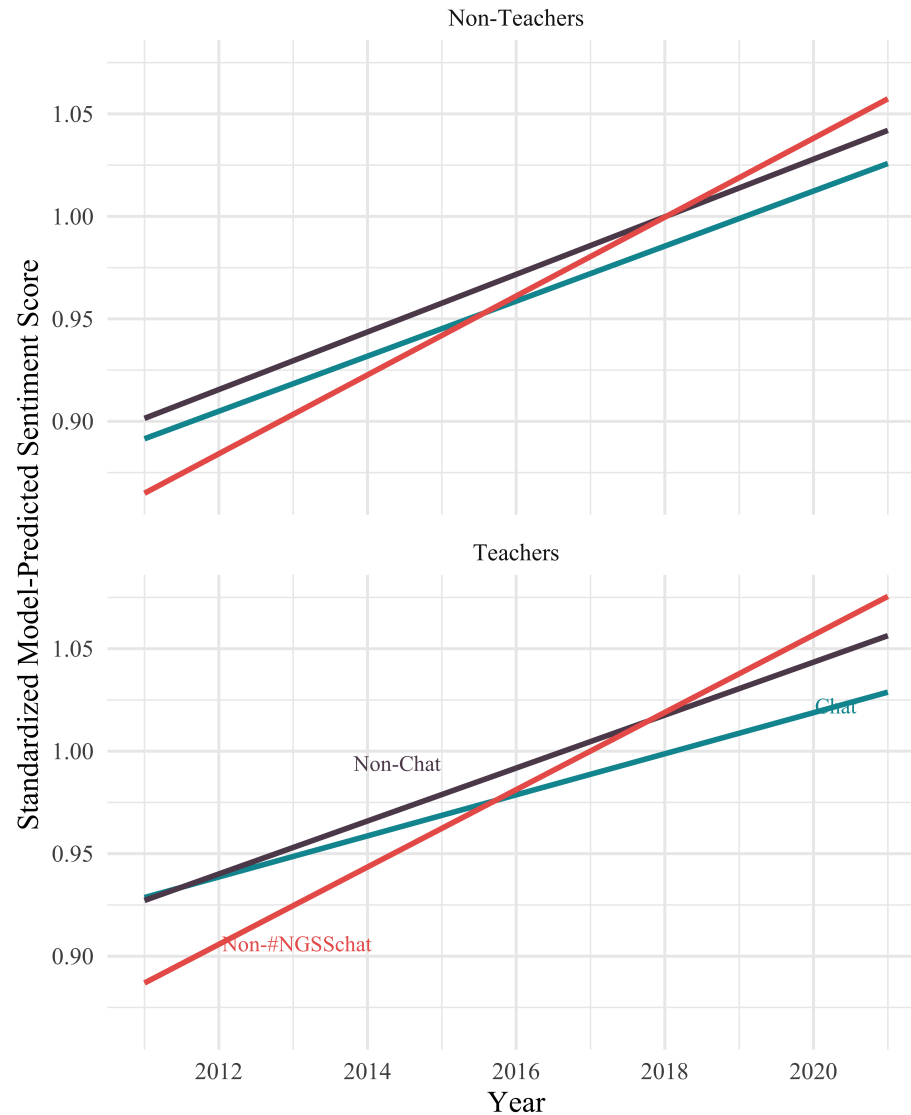
Often, we have to make changes to our data frame in order to create the visualization we would like to create.

2: How does tidying data relate to data viz?

Making a new variable prior to plotting the data

```
pred_frame %>%  
  mutate(isTeacher = ifelse(isTeacher == 0, "Non-Teachers", "Teachers")) %>%  
  ggplot(aes(year_of_post_centered + 2016, prediction)) +  
  geom_line(aes(color = type_of_tweet),  
            size = 1.3) +  
  geom_text(aes(label = label, color = type_of_tweet),  
            family = "Times New Roman",  
            data = label_frame) +  
  facet_wrap(~isTeacher, ncol = 1) +  
  scale_x_continuous("Year", breaks = seq(2010, 2020, 2)) +  
  labs(x = "Year",  
       y = "Standardized Model-Predicted Sentiment Score")
```

2: How does tidying data relate to data viz?



2: How does tidying data relate to data viz?

Other data tidying steps we might take prior to visualizing data:

- **recoding** variables
- **creating a factor** (so that we can order elements of a plot as we wish for them to be ordered)
- **grouping** and **summarizing** to plot a summary statistic
- realizing that your data processing and tidying was not quite sufficient, so **returning to those stages** before finalizing your visualization
- **re-running our analysis** ([.Rmd](#) file) because we discovered an issue with our data

Course Logistics

This week

- Homework 3: Due by Thursday, 9/16
- Homework 4: Due by Tuesday, 9/21
- Readings
 - 1: A Layered Grammar of Graphics ([Wickham, 2010](#))
 - 2: Data visualization ([Wickham & Grolemund, 2018](#))

Coming up

- *Just begin* to think and to ask questions about what you may want to do for [a final project](#): something that will advance your research and allow you to exhibit and extend what you do in class

Wrapping up

In the class-checkout Slack channel:

- What is one thing you learned today?
- What is something you want to learn more about?
- **Also**, in GIF form (type `/giphy` in Slack, and then a random term), summarize how you are feeling about R