Week 6 - Further into Data Viz

Alex Lishinski

September 21, 2021

Welcome!

Welcome to *week 6*!

Record the meeting

Discussion!

Starting with whomever most wants to go first:

One question:

- What is a weird, unusual, or surprising situation that you encountered when using R in the last week?

Review of last week's class

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

Review of last week's class

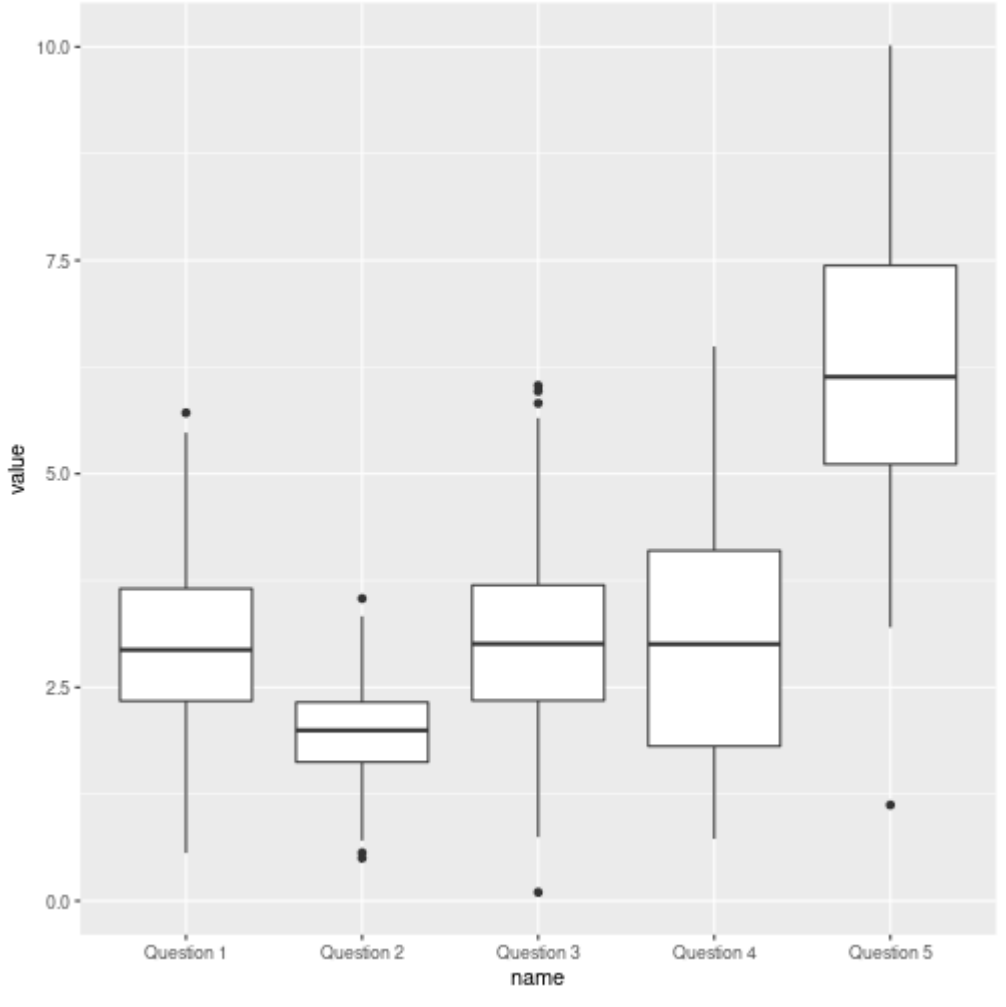
- Exploratory visualization and presentation visualization
- Basics of using base R plotting functions as well as ggplot

Review of last week's class

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

```
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")
```

Review of last week's class



Review of last week's class

This week's topics

Overview

1. Data viz ideas and details
2. Data viz and tidying operations

2 overarching goals of learning data viz in R

- Conceptual framework of visualization
    - Grammar of graphics and different mappings of data onto visual elements
- Details of implementation
    - How to build and refine plots layer by layer
    - Eventually: Interactive data viz with ggviz and shiny

Part 1/2: Data Viz Ideas

1. Data Viz Ideas

Outline

A. Review of the grammar of graphics

B. Understanding visualizations by layers

C. Understanding mapping of data to geoms

1A: Grammar of Graphics

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:

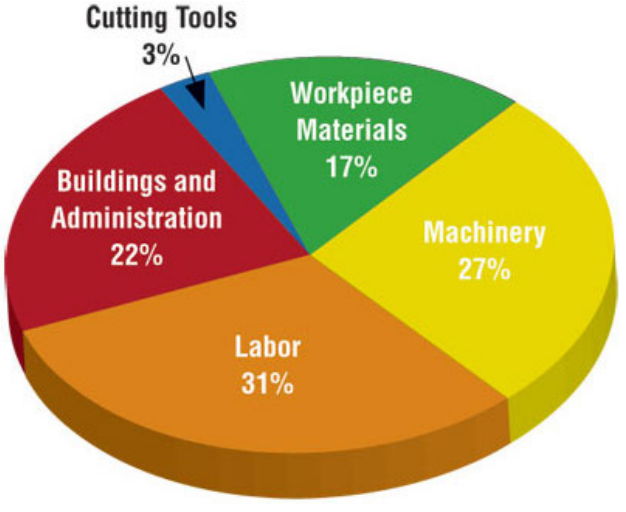
Why a grammar of graphics?

- 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

1A: More Data Viz Ideas

Some general principles for effective data viz

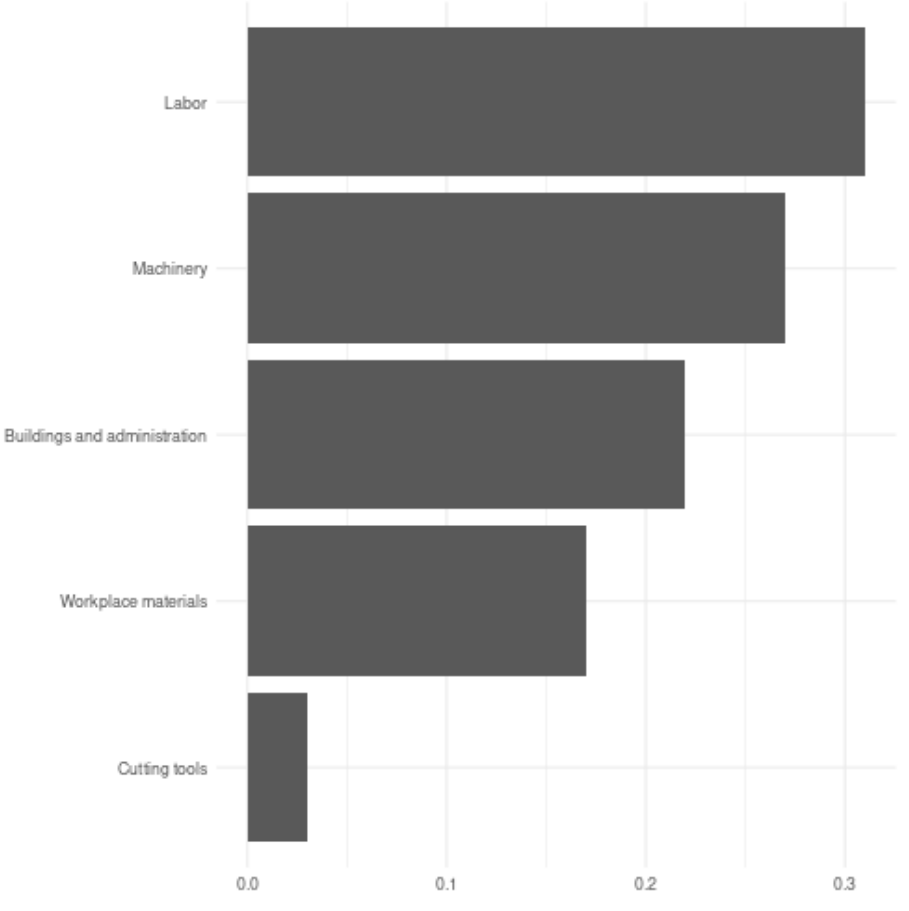
Keep it simple



1A: More Data Viz Ideas

Some general principles for effective data viz

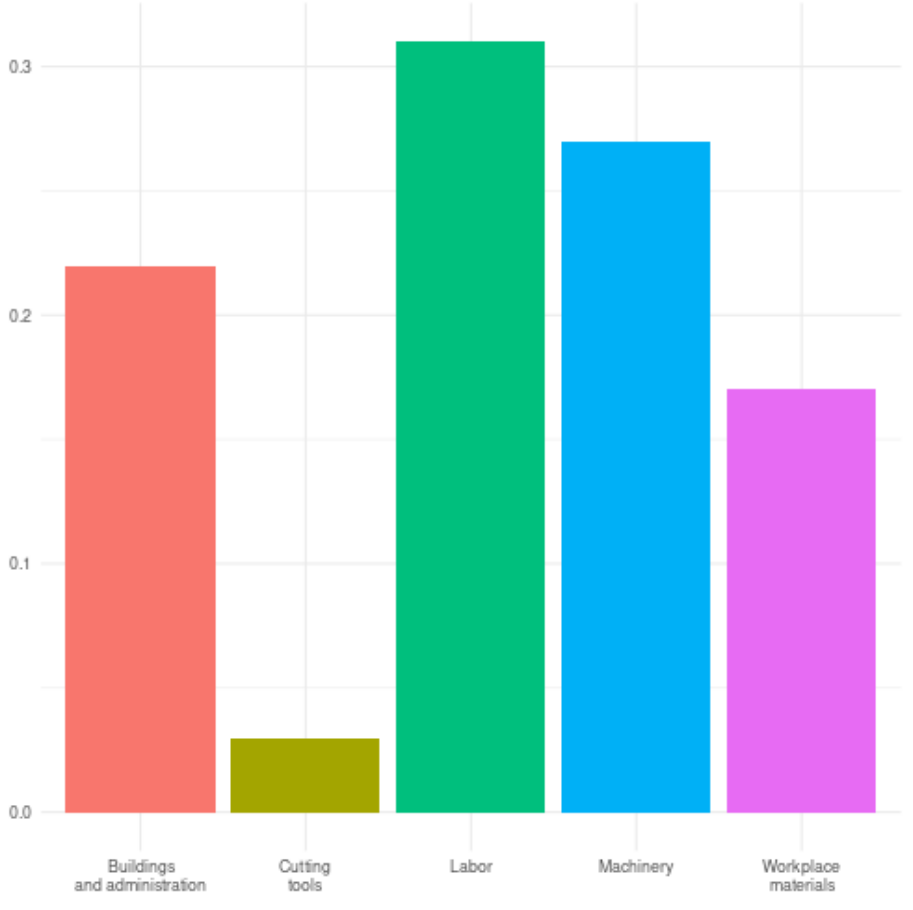
Keep it simple



1A: More Data Viz Ideas

Some general principles for effective data viz

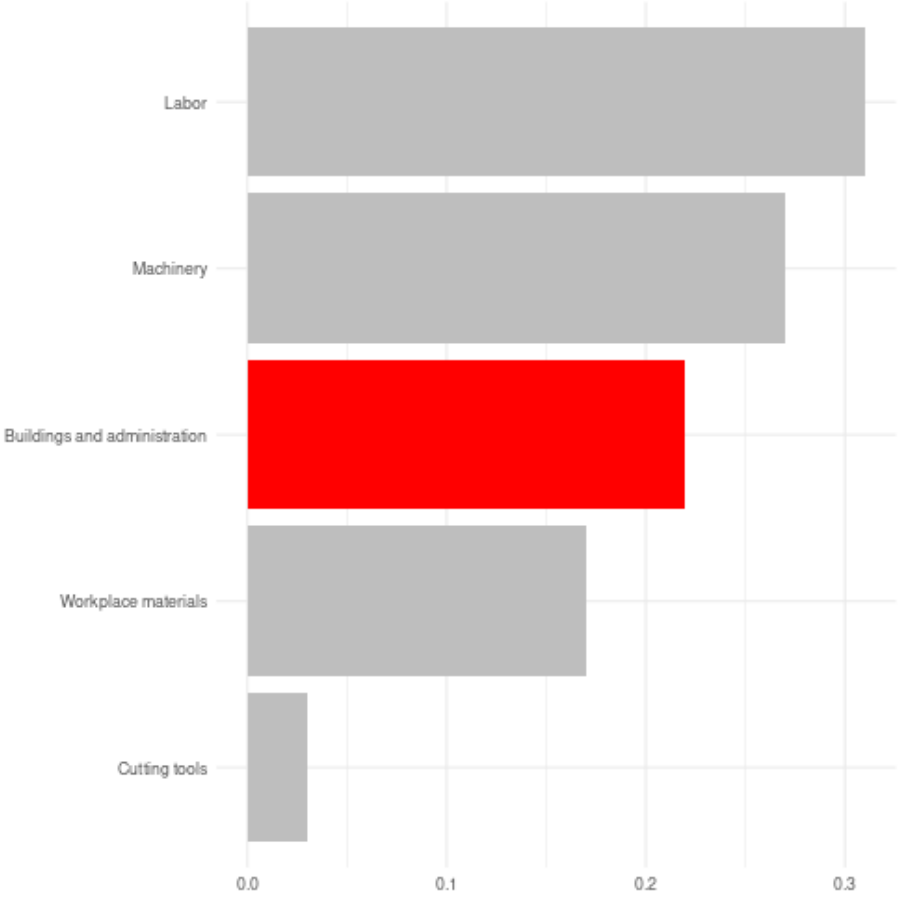
Use color to draw attention



1A: More Data Viz Ideas

Some general principles for effective data viz

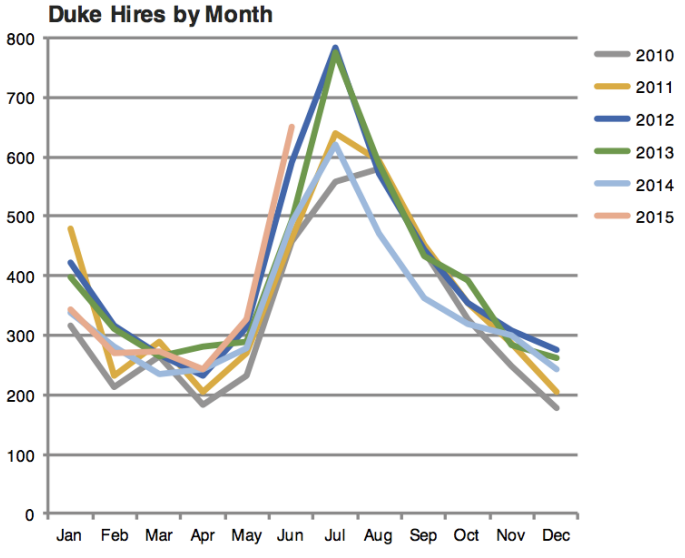
Use color to draw attention

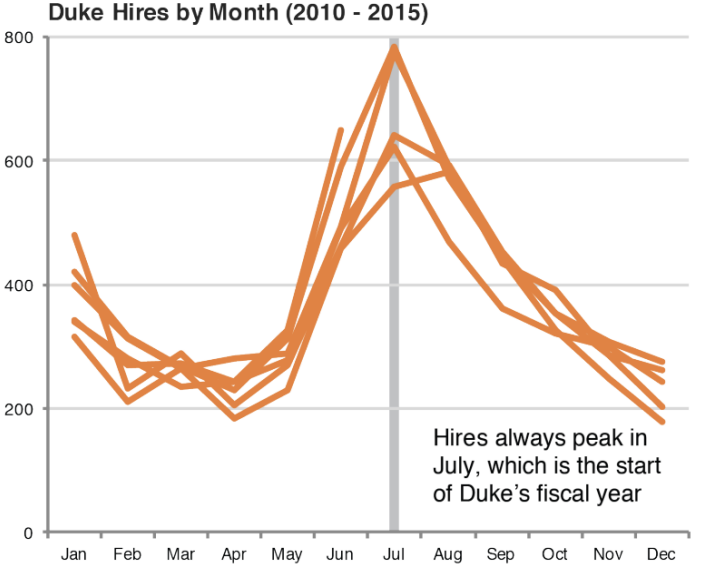


1A: More Data Viz Ideas

Some general principles for effective data viz

Tell a story





1B: Understanding visualizations by layer

Layers:

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: Understanding visualizations by layer

Data

```
mtcars
```

```
##                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4          21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag      21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710         22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive     21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout  18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant            18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360         14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D          24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230           22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280           19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C          17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Merc 450SE         16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL         17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC        15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## Cadillac Fleetwood 10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4  8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial  14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Fiat 128           32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic        30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla     33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona      21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Dodge Challenger   15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## AMC Javelin        15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Camaro Z28         13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Pontiac Firebird   19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Fiat X1-9          27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2      26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa       30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Ford Pantera L     15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Ferrari Dino       19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Maserati Bora      15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
## Volvo 142E         21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

1B: Understanding visualizations by layer

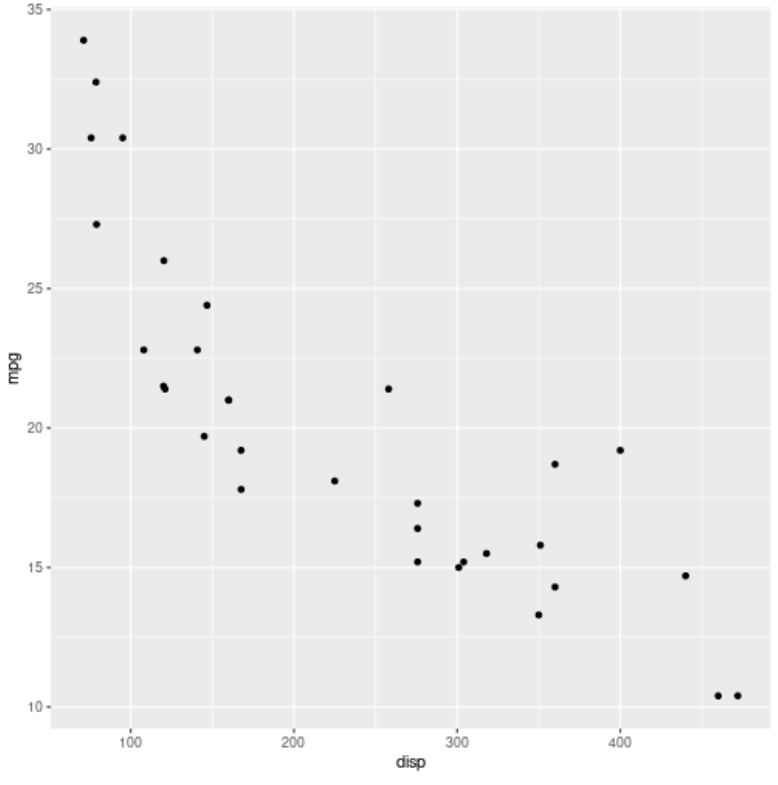
Data

```
ggplot(mtcars)
```

1B: Understanding visualizations by layer

One geom

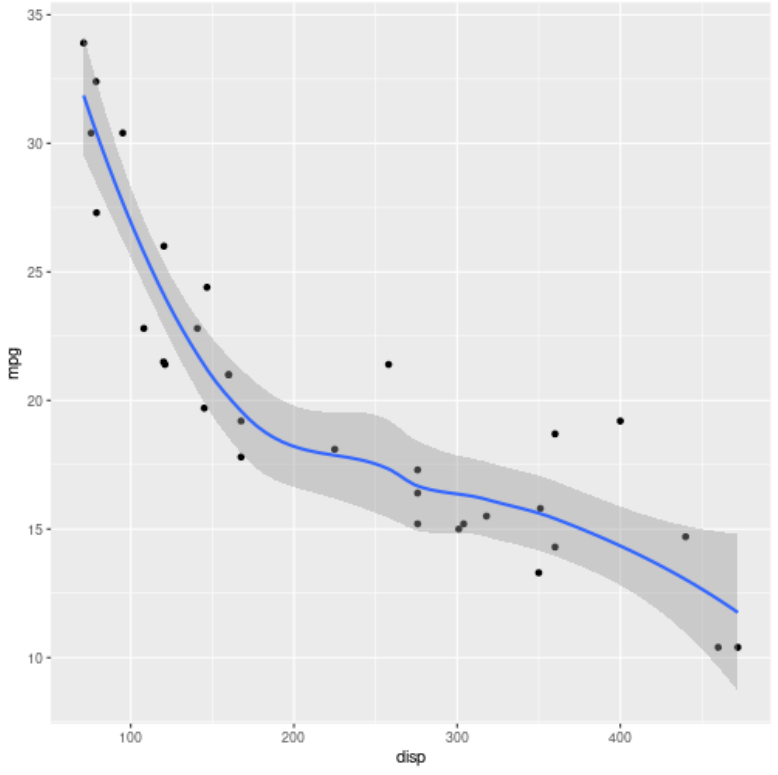
```
ggplot(mtcars) +
  geom_point(aes(x = disp, y = mpg))
```



1B: Understanding visualizations by layer

Additional Geoms

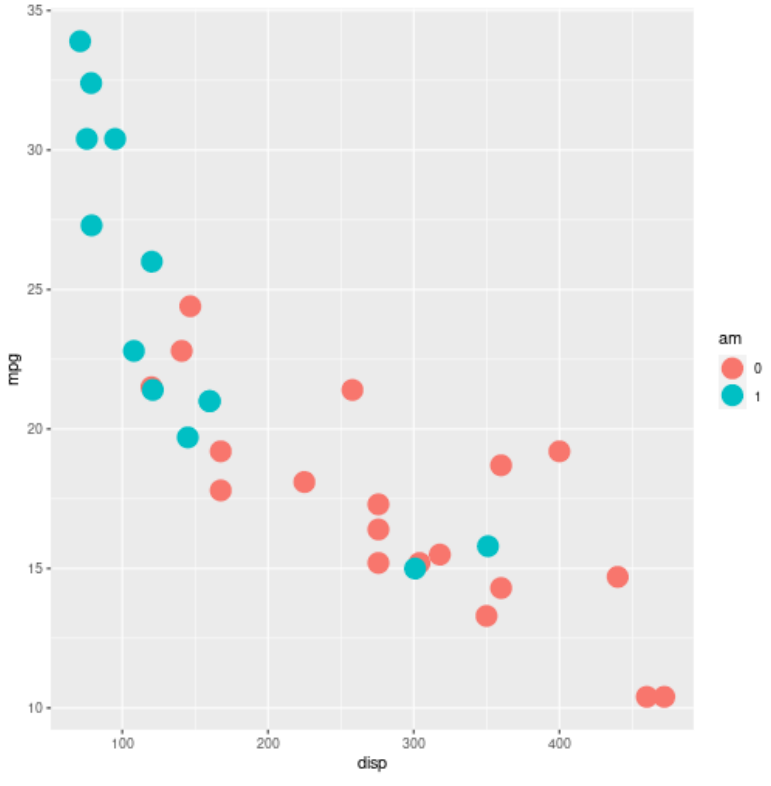
```
ggplot(mtcars) +
  geom_point(aes(x = disp, y = mpg)) +
  geom_smooth(aes(x = disp, y = mpg), method = "loess")
```



1B: Understanding visualizations by layer

Additional Aesthetic Parameters: Color

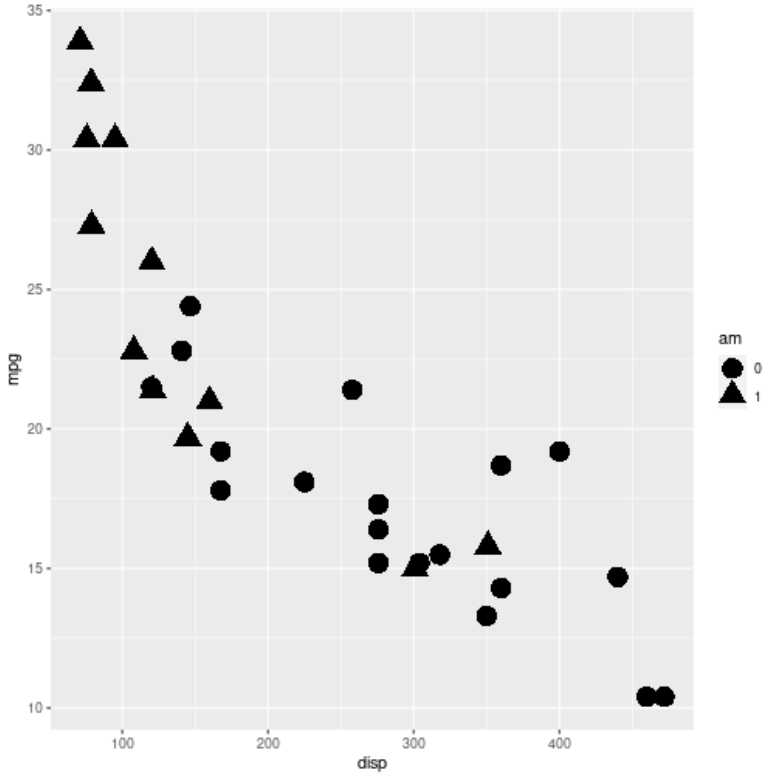
```
ggplot(mtcars) +
  geom_point(aes(x = disp, y = mpg, color = am), size = 6)
```



1B: Understanding visualizations by layer

Additional Aesthetic Parameters: Shape

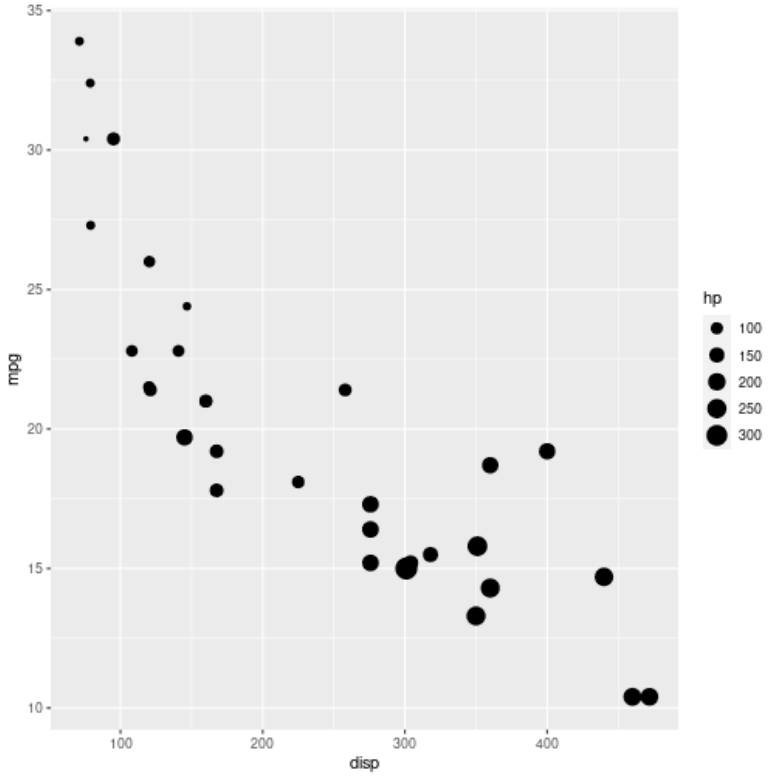
```
ggplot(mtcars) +
  geom_point(aes(x = disp, y = mpg, shape = am), size = 6)
```



1B: Understanding visualizations by layer

Additional Aesthetic Parameters: Size

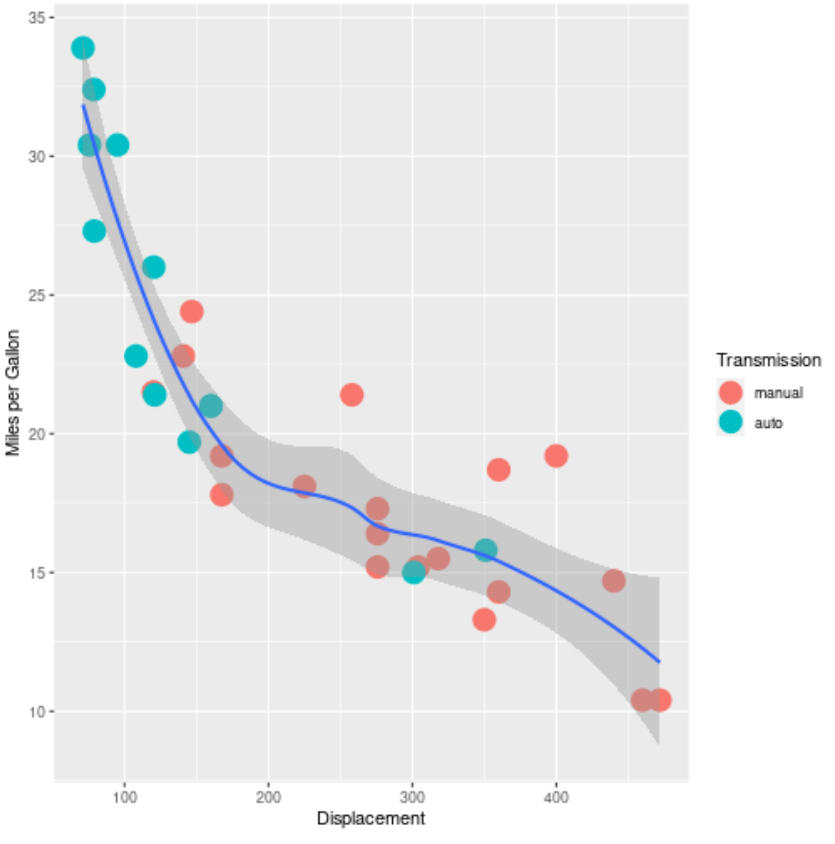
```
ggplot(mtcars) +
  geom_point(aes(x = disp, y = mpg, size = hp))
```



1B: Understanding visualizations by layer

Theme: labels

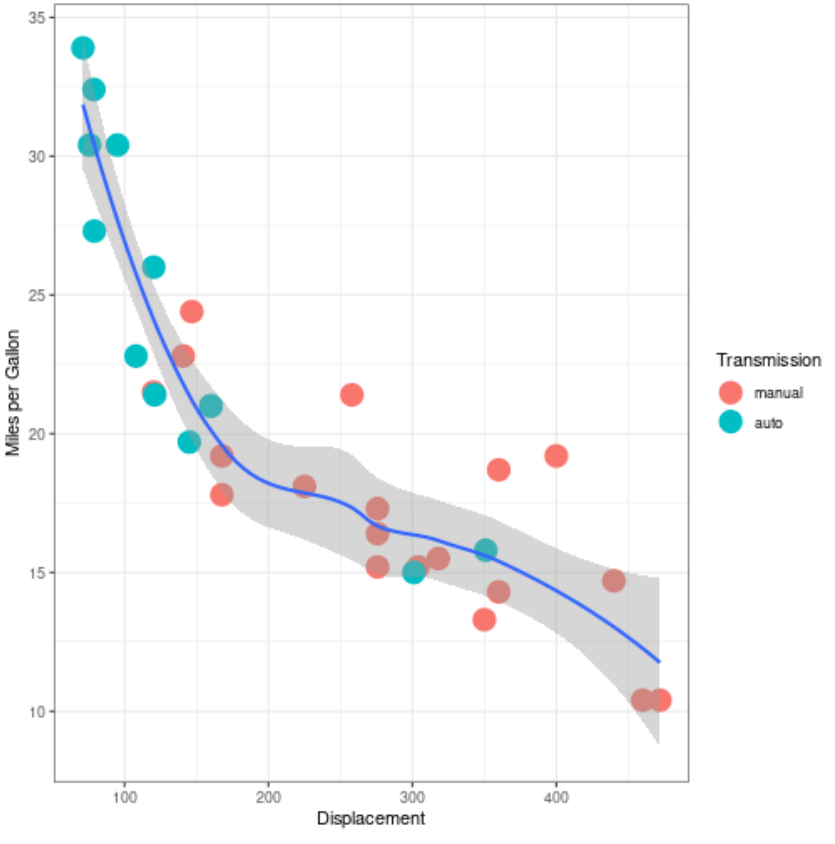
```
# code chunk here
ggplot(mtcars) +
  geom_point(aes(x = disp, y = mpg, colo
  geom_smooth(aes(x = disp, y = mpg), me
  xlab("Displacement") +
  ylab("Miles per Gallon") +
  scale_color_discrete(name = "Transmiss
```



1B: Understanding visualizations by layer

Theme: overall

```
# code chunk here
ggplot(mtcars) +
  geom_point(aes(x = disp, y = mpg, colo
  geom_smooth(aes(x = disp, y = mpg), me
  xlab("Displacement") +
  ylab("Miles per Gallon") +
  scale_color_discrete(name = "Transmiss
  theme_bw()
```



1B: Understanding visualizations by layer

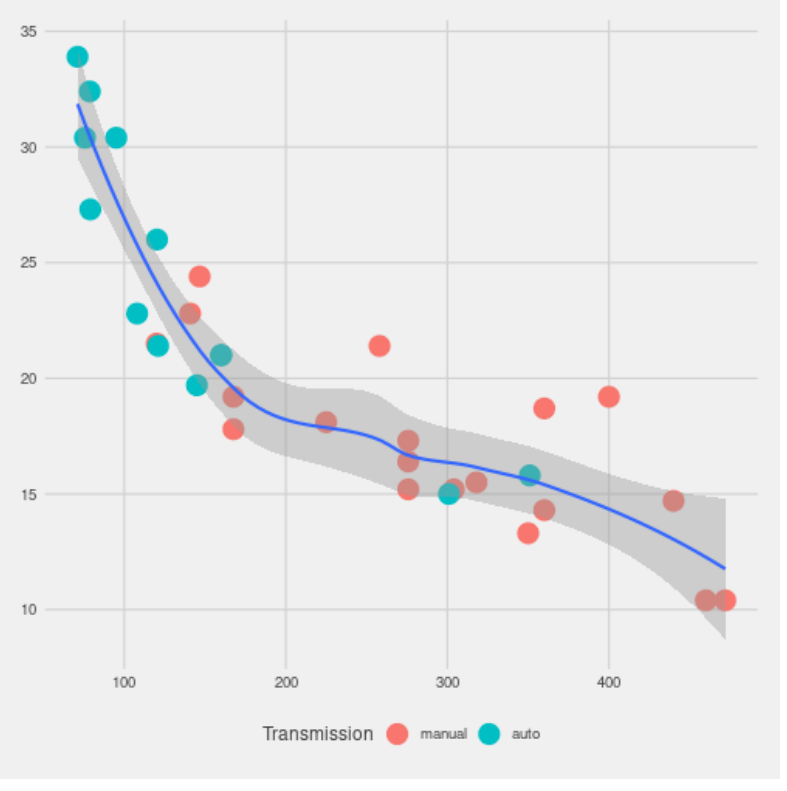
ggthemes package

```
library(ggthemes)

base_plot <- ggplot(mtcars) +
  geom_point(aes(x = disp, y = mpg, color = am), size = 6) +
  geom_smooth(aes(x = disp, y = mpg), method = "loess") +
  xlab("Displacement") +
  ylab("Miles per Gallon") +
  scale_color_discrete(name = "Transmission", labels = c("manual", "auto"))
```

1B: Understanding visualizations by layer

Fivethirtyeight style

```
base_plot + theme_fivethirtyeight()
```



1C: Understanding mapping data to geoms

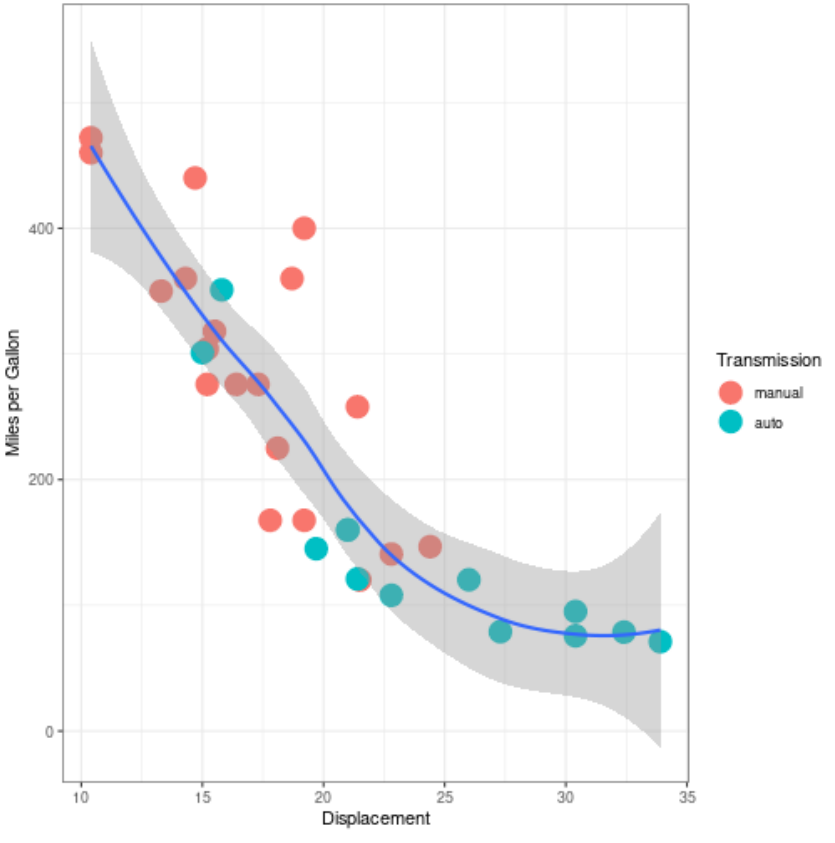
You can create different plots by:

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

1C: Understanding mapping data to geoms

Changing the mapping

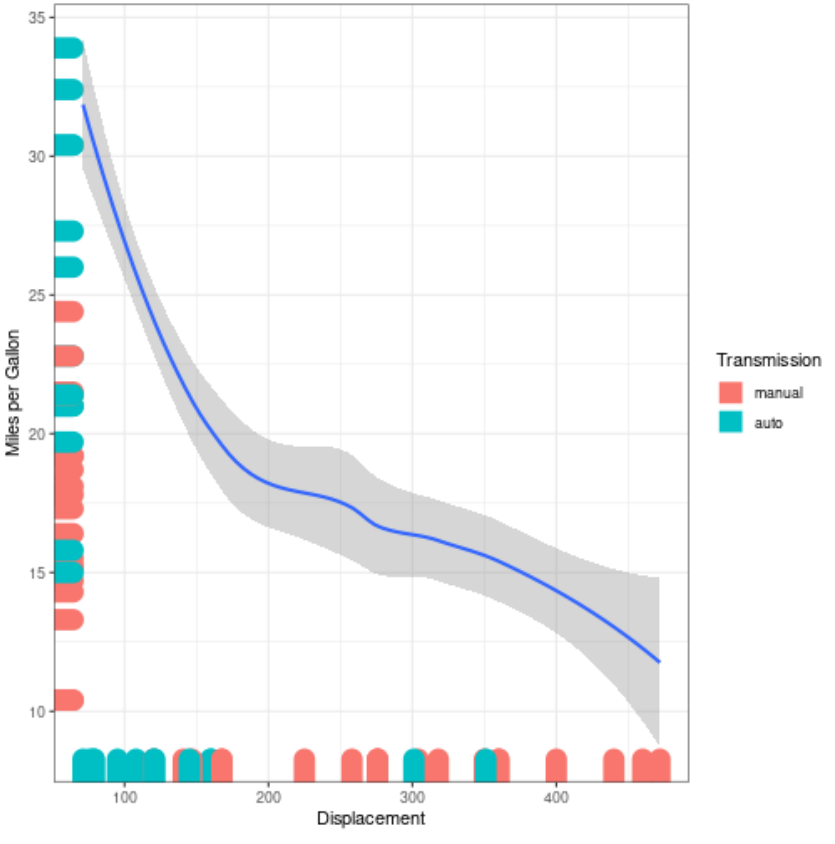
```
# code chunk here
ggplot(mtcars) +
  geom_point(aes(y = disp, x = mpg, colo
  geom_smooth(aes(y = disp, x = mpg), me
  xlab("Displacement") +
  ylab("Miles per Gallon") +
  scale_color_discrete(name = "Transmiss
  theme_bw()
```



1C: Understanding mapping data to geoms

Changing geoms

```
# code chunk here
ggplot(mtcars) +
  geom_rug(aes(x = disp, y = mpg, color
  geom_smooth(aes(x = disp, y = mpg), me
  xlab("Displacement") +
  ylab("Miles per Gallon") +
  scale_color_discrete(name = "Transmiss
  theme_bw()
```



Part 2/2: Data Viz and Tidying

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.

Making a new variable prior to plotting the data

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

2: How does tidying data relate to data viz?

Sometimes we need to recode a variable or add a new one

```
tidykids <- read_csv(here("content", "data", "tidykids.csv"))
```

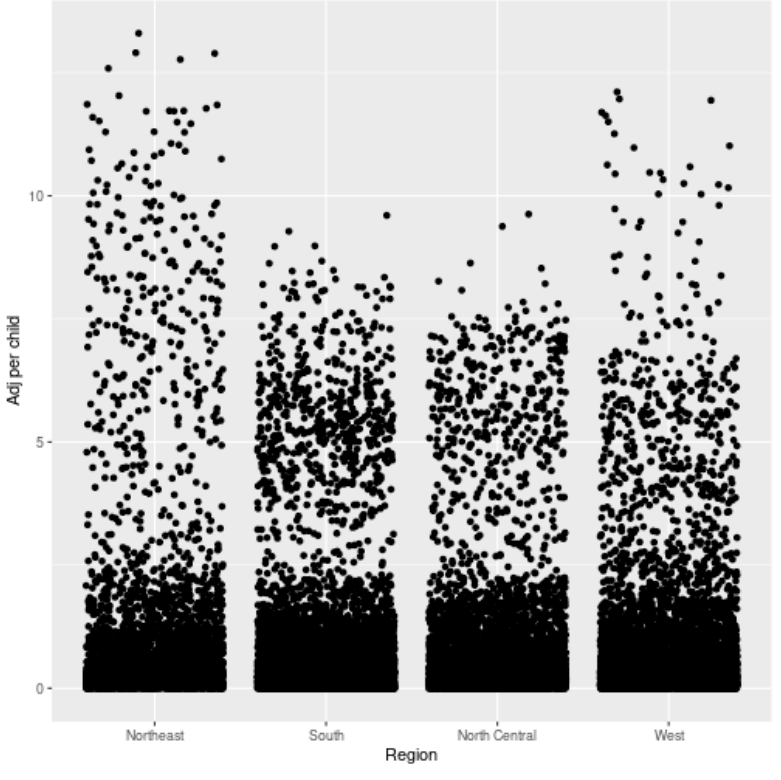
```
##
## ── Column specification ────────────────────────────────────────────────────────
## cols(
##   state = col_character(),
##   variable = col_character(),
##   year = col_double(),
##   raw = col_double(),
##   inf_adj = col_double(),
##   inf_adj_perchild = col_double()
## )
```

```
state_region <- data.frame(state.name, state.region)

tidykids_reg <- left_join(tidykids, state_region, by = c("state" = "state.name"))

tidykids_reg$timeblock <- recode(tidykids_reg$year,
       `1997` = "1997-2001", `1998` = "1997-2001", `1999` = "1997-2001", `2000` = "1997-2001", `2001`
       `2002` = "2002-2006", `2003` = "2002-2006", `2004` = "2002-2006", `2005` = "2002-2006", `2006`
       `2007` = "2007-2011", `2008` = "2007-2011", `2009` = "2007-2011", `2010` = "2007-2011", `2011`
       `2012` = "2012-2016", `2013` = "2012-2016", `2014` = "2012-2016", `2015` = "2012-2016", `2016`
```

2: How does tidying data relate to data viz?

Sometimes we need to recode a variable for plotting

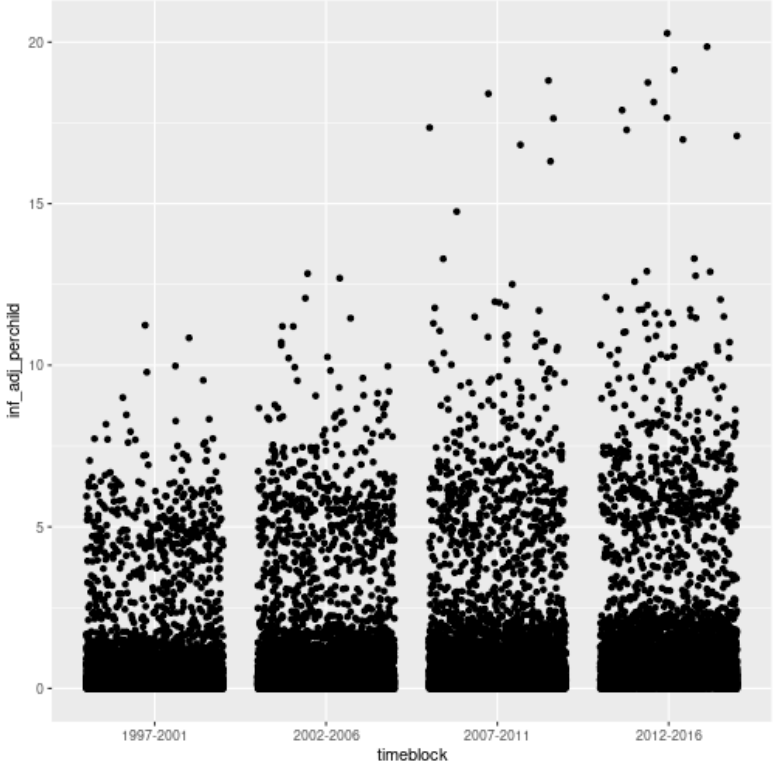
```
ggplot(na.omit(tidykids_reg)) +
  geom_jitter(aes(x = state.region, y = inf_adj_perchild)) +
  xlab("Region") +
  ylab("Adj per child")
```



2: How does tidying data relate to data viz?

Creating and reordering factors is often useful

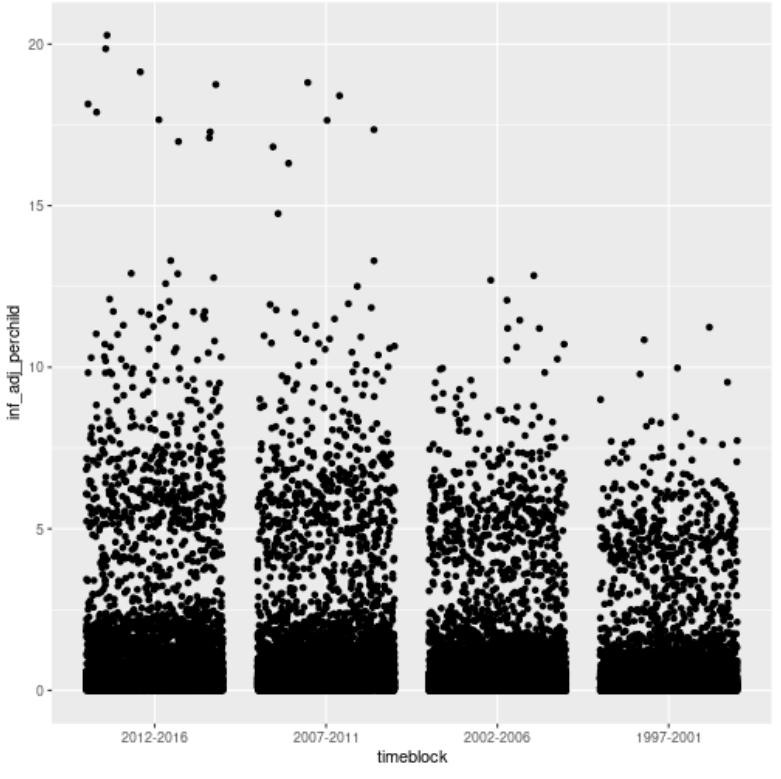
```
tidykids_reg <- tidykids_reg %>%
  mutate(timeblock = factor(timeblock))
ggplot(tidykids_reg) +
  geom_jitter(aes(timeblock, inf_adj_perchild))
```



2: How does tidying data relate to data viz?

Creating and reordering factors is often useful

```
tidykids_reg$timeblock <- fct_relevel(tidykids_reg$timeblock, c("2012-2016", "2007-2011", "2002-2006"

ggplot(tidykids_reg) +
  geom_jitter(aes(timeblock, inf_adj_perchild))
```



2: How does tidying data relate to data viz?

When we do `group_by()` and `summarize()` we can plot summary statistics

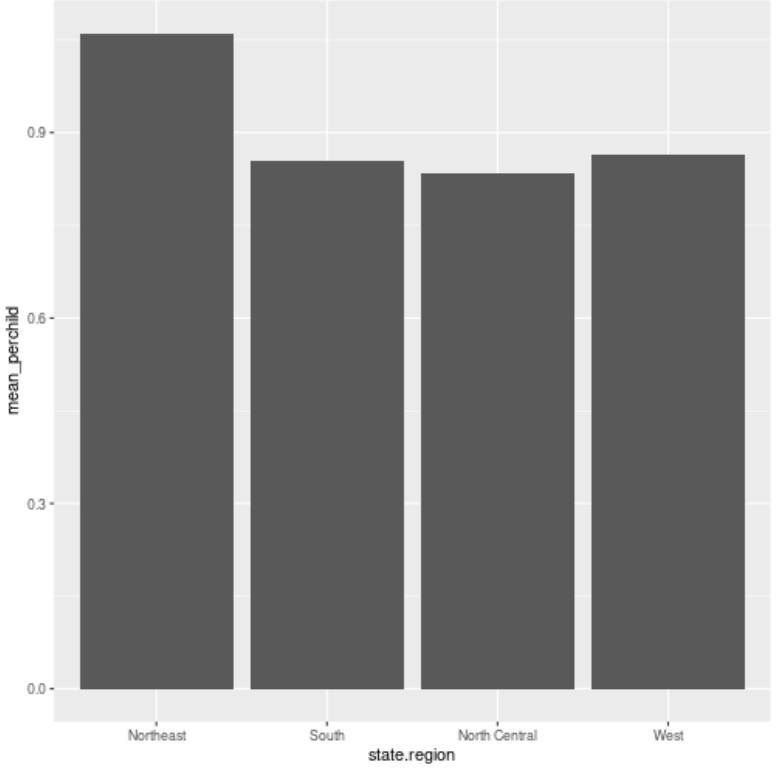
```
summ_df <- na.omit(tidykids_reg) %>%
  group_by(state.region) %>%
  summarize(mean_perchild = mean(inf_adj_perchild, na.rm = T))

summ_df
```

```
## # A tibble: 4 × 2
##   state.region  mean_perchild
##   <fct>                 <dbl>
## 1 Northeast             1.06
## 2 South                 0.855
## 3 North Central         0.834
## 4 West                  0.865
```

2: How does tidying data relate to data viz?

When we do `group_by()` and `summarize()` we can plot summary statistics

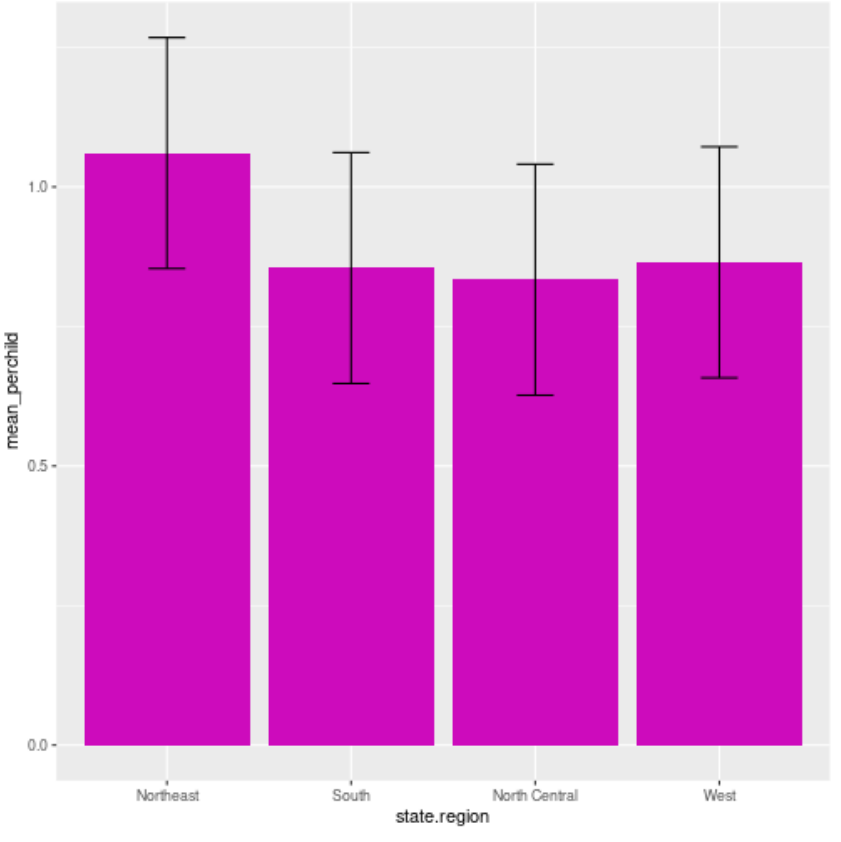
```
summ_df %>%
  ggplot() +
    geom_col(aes(state.region, mean_perchild))
```



2: How does tidying data relate to data viz?

When we do `group_by()` and `summarize()` we can plot summary statistics

```
# code chunk here
  summ_df %>%
    ggplot() +
      geom_bar(aes(state.region, mean_pe
      geom_errorbar(aes(x = state.region
                position=position_dodge(
```



Course Logistics

This week

- NO CLASS THURSDAY
- Homework 5: Available Thursday
- Video: Friday
- Readings
  - 1: https://clauswilke.com/dataviz/histograms-density-plots.html
  - 2: https://clauswilke.com/dataviz/visualizing-proportions.html

Wrapping up

On Slack channel:

- What is one thing you learned today?
- What is something you want to learn more about?