

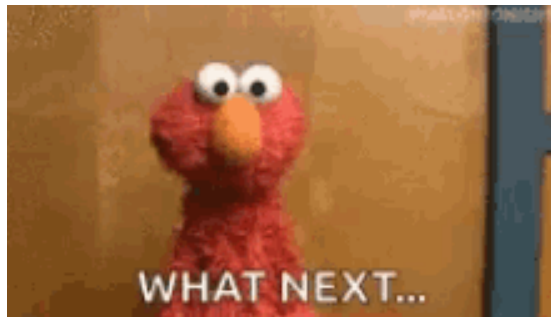
# Data Visualization

## Visualization with Purpose: Refine Your Plots

Ciara Zogheib

Data Sciences Institute, University of Toronto

# A Question...



- Visualizing qualitative data?
- Tableau Public intro?
- Data viz with Python intro?
- Other design and visualization stuff? (Testing with users? Features surrounding your data viz?)
- Return to previous topics? Which ones?

## In today's class, we will...

- Work through practice code from Chapter 8 (*Refine Your Plots*) of Healy, K. (2018). *Data Visualization: A Practical Introduction*. Princeton University Press. This code will let us
  - Purposefully choose and use colour in our ggplots
  - Layer colour and text
  - Explore ggplot themes and use them substantively
- Use case studies to learn how to fix common mistakes and improve 'bad' data visualizations

- By now, we have learned how to make a variety of different data visualizations, and to add different elements to those data visualizations with ggplot
- If we are exploring data for own purposes, the process can end there, but what if the data visualization is being designed to communicate with an audience?
- Once we have created our plot, we often need to make aesthetic changes, either according to our own tastes or to specific [format requirements](#)

# **Review: Building our base ggplot**

# Loading our sample dataset

- We will use a new sample dataset (**asasec**, from the **socviz** library) containing membership data from special interest sections of the American Sociological Association

```
library(socviz)
head(asasec)
```

- To start, we will explore the relationship between the variables `Beginning` (each section's reserves) and `Revenues` (each section's income) in a single year

## **Activity: Explore our data**

Use what we have learned about ggplot to create a scatterplot and smoothed graph comparing membership and revenues for the year 2014.

(Hint: subset!)

## Activity: Explore our data

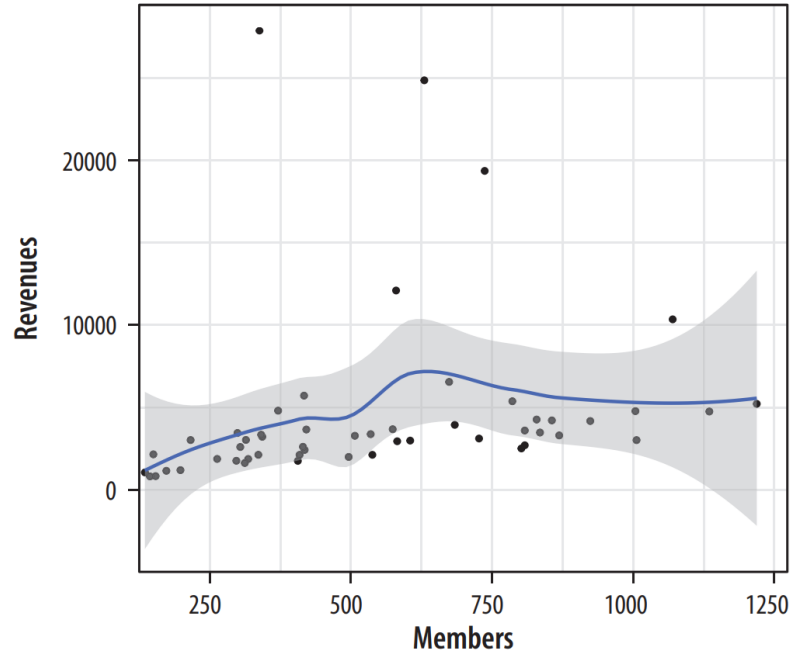
Use what we have learned about ggplot to create a scatterplot and smoother graph **comparing membership and revenues** for the **year 2014**.

(Hint: subset!)

```
p <- ggplot(data = subset(asasec, Year == 2014), mapping = aes(x =  
  Members, y = Revenues, label = Sname))  
  
p + geom_point() + geom_smooth()
```



# Activity: Explore our data



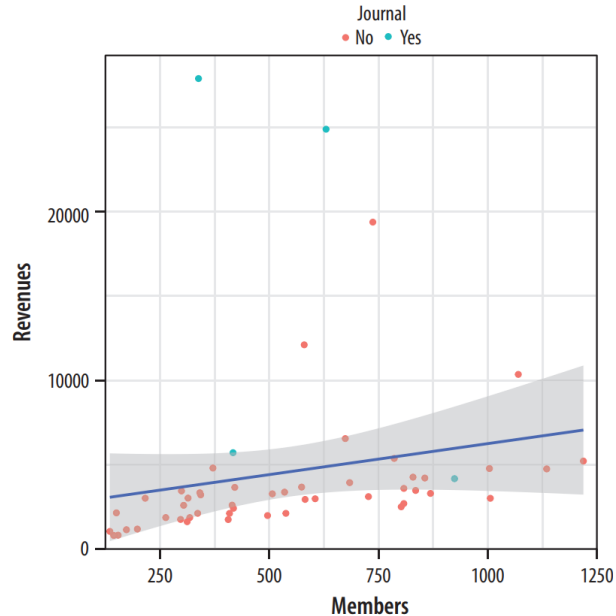
```
p = ggplot(data = subset(asasec, Year == 2014), mapping = aes(x =  
Members, y = Revenues))  
  
p + geom_point() + geom_smooth()
```

# Modifying our base plot

- We have already learned some basic modifications that we can make to our initial plot
- First, we will colour our points based on the variable `Journal`, and switch our `geom_smooth()` from the default `stat_` of loess to a linear model

```
p <- ggplot(data = subset(asasec, Year == 2014), mapping =  
aes(x = Members, y = Revenues))  
  
p + geom_point(mapping = aes(color = Journal)) +  
geom_smooth(method = "lm")
```

# Modifying our base plot



```
p = ggplot(data = subset(asasec, Year == 2014), mapping = aes(x =  
Members, y = Revenues, label = Sname))  
  
p + geom_point(mapping = aes(color = Journal)) + geom_smooth(method =  
"lm")
```

# Using intermediate ggplot objects

- As we start to add more and more layers to our ggplot object (p), it can be helpful to use intermediate objects (p1, p2, etc)
- For example, if we want to add text labels to points where Revenues is > 7000 to our current plot:

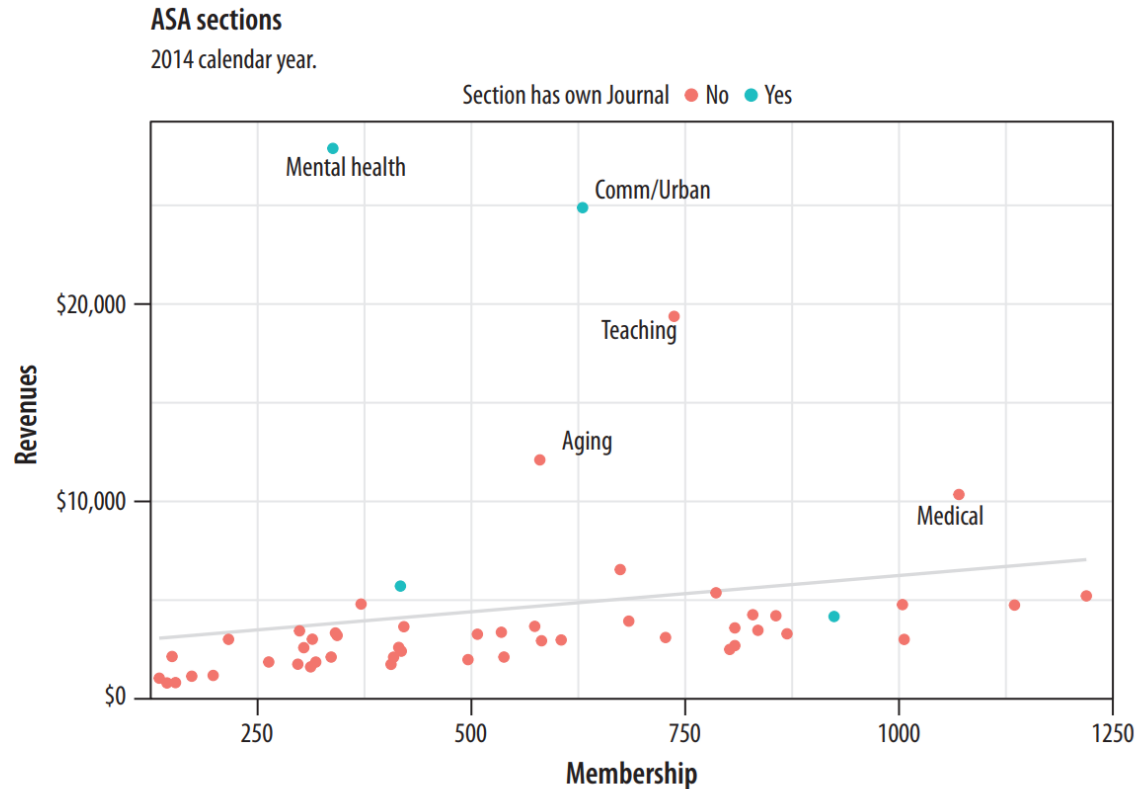
```
p0 <- ggplot(data = subset(asasec, Year == 2014), mapping =  
aes(x = Members, y = Revenues, label=Sname))  
  
p1 <- p0 + geom_smooth(method = "lm", se = FALSE, color =  
"gray80") + geom_point(mapping = aes(color = Journal))  
  
p2 <- p1 + geom_text_repel(data = subset(asasec, Year == 2014 &  
Revenues > 7000), size = 2)
```

# Using intermediate ggplot objects

- We can also add axis labels, create a plot title, and move our legend

```
p3 <- p2 + labs(x="Membership",  
               y="Revenues",  
               color = "Section has own Journal",  
               title = "ASA Sections",  
               subtitle = "2014 Calendar year.",  
               caption = "Source: ASA annual report.")  
p4 <- p3 + scale_y_continuous(labels = scales::dollar) +  
  theme(legend.position = "top")
```

# Using intermediate ggplot objects - result



Source: ASA annual report.

# Using Colour

# Colour palettes

- The `RColorBrewer` package provides several pre-made colour palettes to use with our data visualizations
- We should choose a colour palette based on its ability to express the data we are trying to communicate. For example:
  - **Sequential palettes** should be used for ordered data (eg. less to more, earlier to later)
  - **Qualitative palettes** should be used for unordered, categorical data (eg. countries, genders)
  - **Diverging palettes** should be used for ordered data centred on mid-range values with extremes at either end of the range of values



# Colour palettes

- The R Graph Gallery includes [a guide](#) to the appearances and names of premade palettes in RColorBrewer
- We can also view them in R using:

```
library(RColorBrewer)
par(mar=c(3,4,2,2))
display.brewer.all()
```

# Colour palettes



```
library(RColorBrewer)

par(mar=c(3,4,2,2))

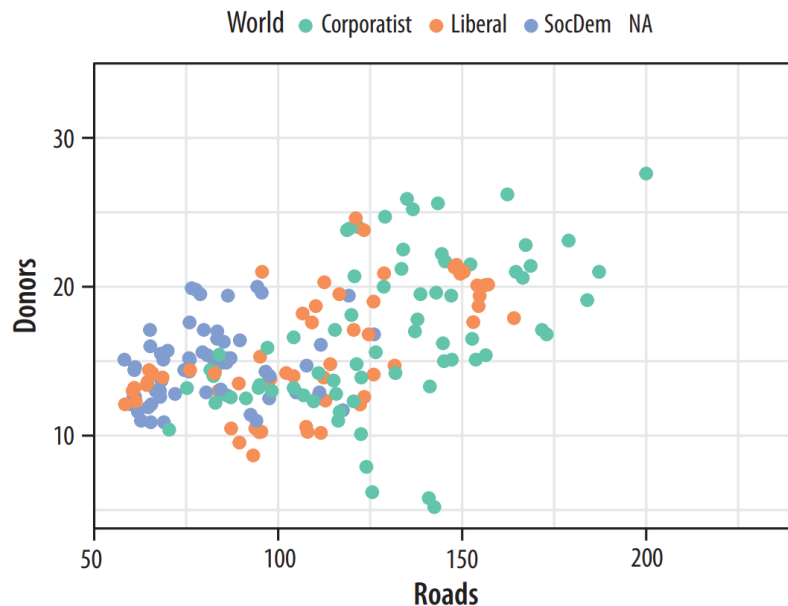
display.brewer.all()
```

## Colour palettes - Example

- We use the RColorBrewer palettes with ggplot using the `scale_color_brewer()` or `scale_fill_brewer()` functions
- For example, using the `organdata` dataset from previous classes:

```
p <- ggplot(data = organdata, mapping = aes(x = roads, y =  
donors, color = world))  
  
p + geom_point(size = 2) +  
  scale_color_brewer(palette = "Set2") +  
  theme(legend.position = "top")
```

# Colour palettes - Example



```
p <- ggplot(data = organdata, mapping = aes(x = roads, y = donors, color = world))  
p + geom_point(size = 2) + scale_color_brewer(palette = "Set2") + theme(legend.position =  
"top")
```

# Manually choosing colours

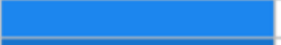






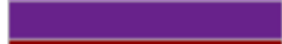


- We can also manually create a vector of colours, then apply it to our ggplots using `scale_color_manual()` or `scale_fill_manual()` functions
- We can create our colour vector either by using hexadecimal RGB values or by using colour names known by R

# Hexadecimal values

- Hexadecimal values are a way of encoding colours in the format of #rrggbb, or
  - A hash or pound character (#) followed by
  - Three pairs of “hex” numbers, each of which is a two digit code for a colour in the red, green, and blue channels
- [Google](#) offers a hexadecimal colour picker, while [palleton.com](#) lets you pick a colour and generate hexadecimal codes for your chosen colour and a range of automatically generated palettes

# Colour names in R

- Certain colour names are already defined in R
- [A helpful document](#) by Dr. Ying Wei contains the names of the colours predefined in R (only a small sample shown here)

	darkolivegreen3		dodgerblue
	darkolivegreen4		dodgerblue1
	darkorange		dodgerblue2
	darkorange1		dodgerblue3
	darkorange2		dodgerblue4
	darkorange3		firebrick
	darkorange4		firebrick1
	darkorchid		firebrick2
	darkorchid1		firebrick3
	darkorchid2		firebrick4
	darkorchid3		floralwhite
	darkorchid4		forestgreen
	darkred		gainsboro
	darksalmon		ghostwhite

## Manually choosing colours

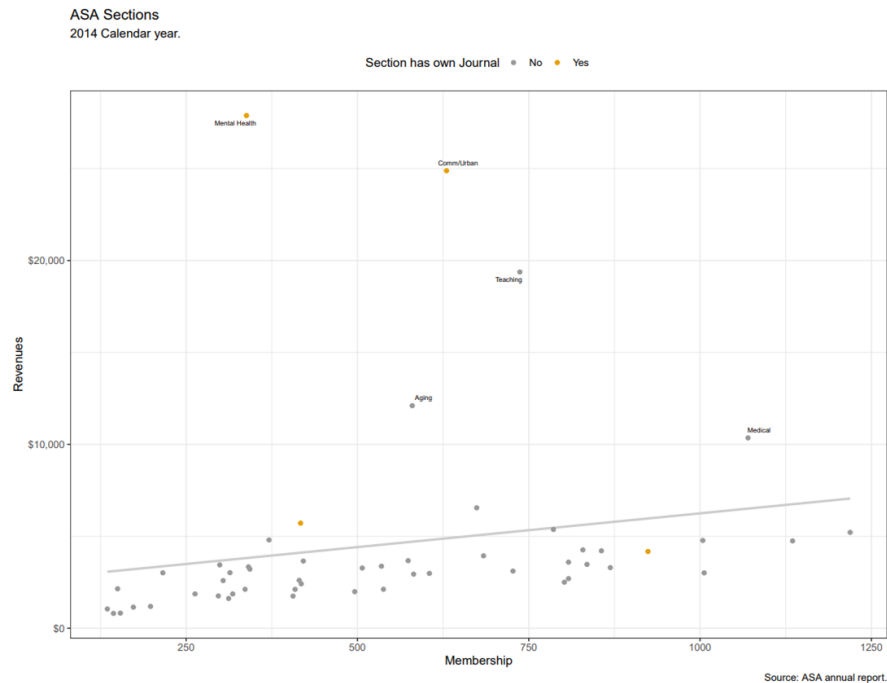
- So, using our ASA membership plot (p4), we can manually create a palette and use it to colour our visualization

```
cb_palette <- c("#999999", "#E69F00", "#56B4E9", "#009E73",  
"#F0E442", "#0072B2", "#D55E00", "#CC79A7")  
  
p4 + scale_color_manual(values = cb_palette)
```

- If we preferred, we could have also defined `cv_palette` using colour names



# Manually choosing colours



```
cb_palette <- c("#999999", "#E69F00", "#56B4E9", "#009E73", "#F0E442",  
               "#0072B2", "#D55E00", "#CC79A7")  
  
p4 + scale_color_manual(values = cb_palette)
```

**Layer colour and text**

## Highlighting data with colour

- We will use the `county_data` dataset from the `socviz` library to create a data visualization about the U.S. general election in 2016
- First, we define a blue and red colour for the Democratic and Republican parties, respectively

```
party_colors <- c("#2E74C0", "#CB454A")
```

## Highlighting data with colour

- Next, we create the first layer of our visual: a scatterplot of counties that did **not** flip parties in the election, with the x-axis showing population on a logarithmic scale and the y-axis showing the percentage of the population that is Black

```
p0 <- ggplot(data = subset(county_data, flipped == "No"),  
             mapping = aes(x = pop, y = black/100))  
p1 <- p0 + geom_point(alpha = 0.15, color = "gray50") +  
      scale_x_log10(labels=scales::comma)  
p1
```

## Highlighting data with colour

- The second layer of our visual will show counties that **did** flip party affiliation in the election
- This time, we specify a manual colour scale using the `party_colors` variable we defined previously

```
p2 <- p1 + geom_point(data = subset(county_data, flipped ==  
"Yes"), mapping = aes(x = pop, y = black/100, color =  
partywinner16)) +  
  scale_color_manual(values = party_colors)  
p2
```

# Highlighting data with colour

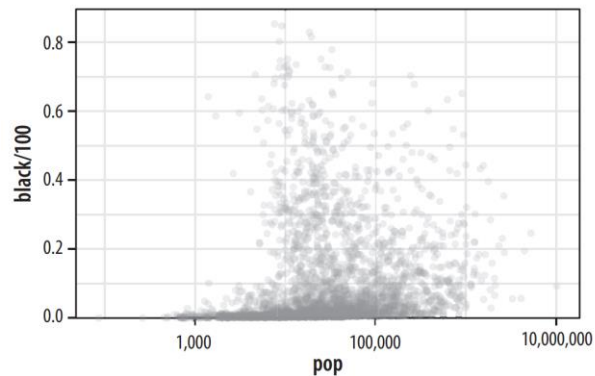
- Our next layer will label our axes, graph, and legend

```
p3 <- p2 + scale_y_continuous(labels=scales::percent) +  
  labs(color = "County flipped to ... ",  
        x = "County Population (log scale)",  
        y = "Percent Black Population",  
        title = "Flipped counties, 2016",  
        caption = "Counties in gray did not flip.")
```

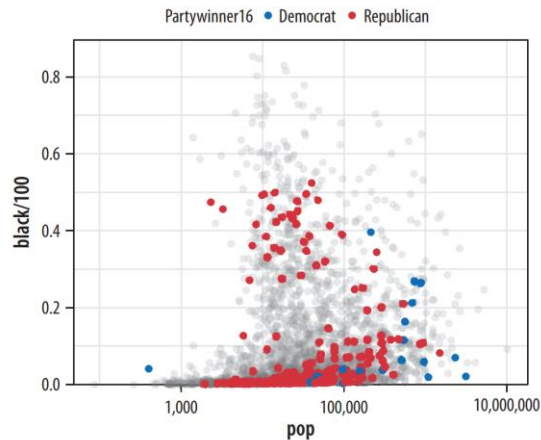
p3

# Highlighting data with colour

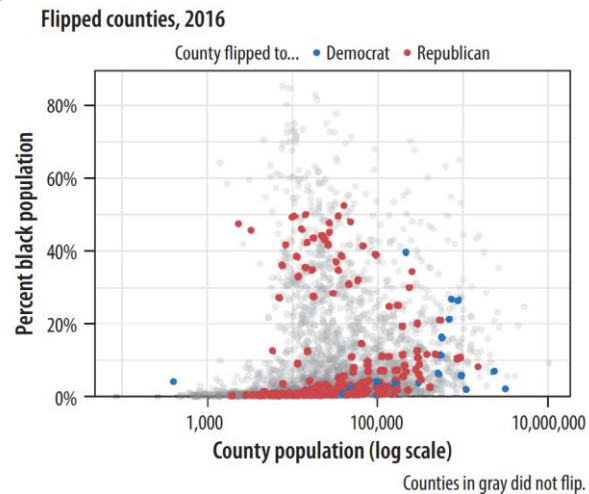
p1



p2



p3



From left to right, our p1, p2, and p3 plots

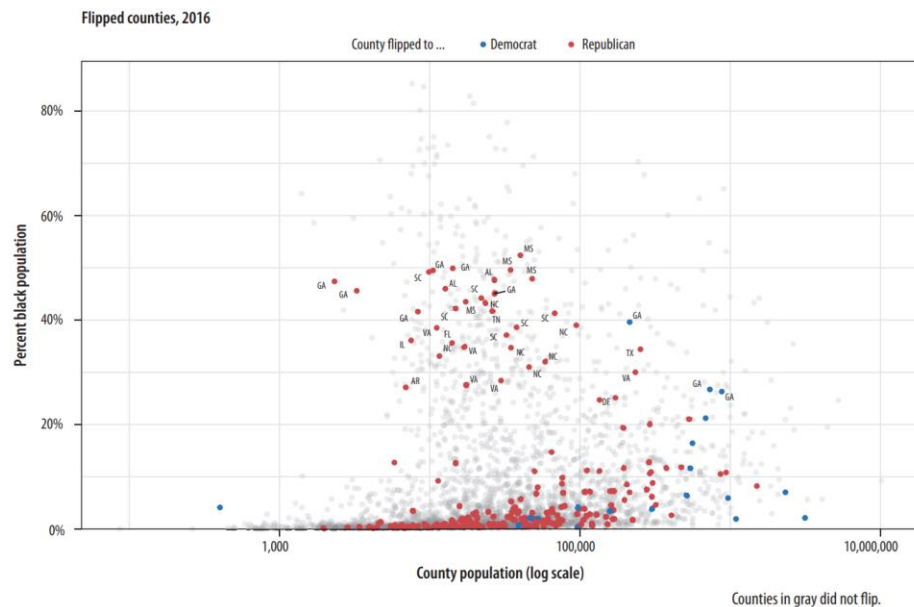
## Highlighting data with colour

- Finally, we add data labels to highlight the flipped counties that have a relatively high percentage of Black residents

```
p4 <- p3 + geom_text_repel(data = subset(county_data, flipped ==  
"Yes" & black > 25), mapping = aes(x = pop, y = black/100, label  
= state), size = 2)  
  
p4 + theme_minimal() + theme(legend.position="top")
```



# Highlighting data with colour - Result



```
p4 <- p3 + geom_text_repel(data = subset(county_data, flipped ==  
"Yes" & black > 25), mapping = aes(x = pop, y = black/100, label =  
state), size = 2)  
  
p4 + theme_minimal() + theme(legend.position="top")
```

**Change plot appearance with themes**

# ggplot themes

- Most of the ggplot data visualizations that we have seen and made so far have a similar **theme** - a white background with light grey gridlines
- We can alter the themes of our ggplots by using the `theme_set()` function, with a theme name as the argument. For example, using our election data plot:

```
theme_set(theme_dark())  
p4
```

# ggplot themes

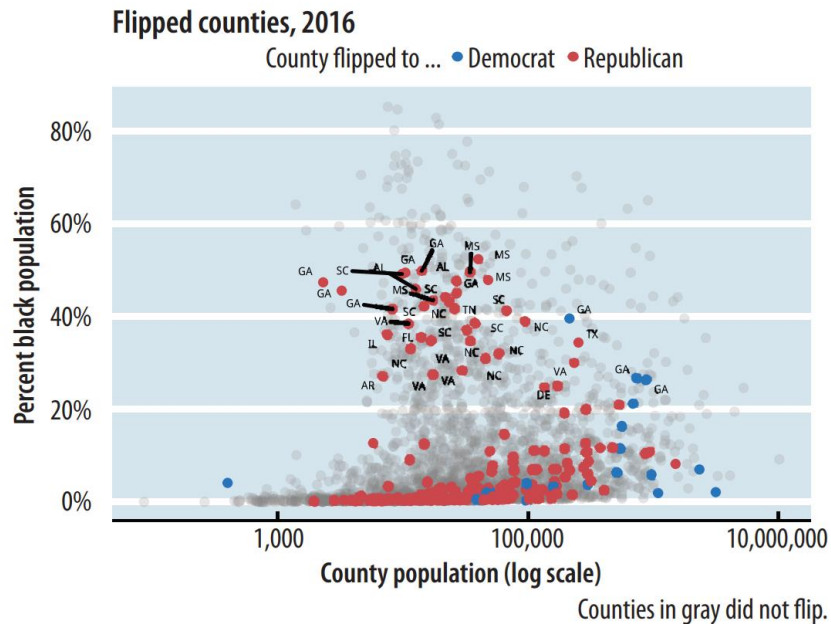
- Tidyverse documentation contains [a list of included ggplot themes](#)
- We can also install the `ggthemes` package for [additional options](#), including *Economist* and *Wall Street Journal* formats, applied in the same way

```
install.packages("ggthemes")  
  
library(ggthemes)  
  
theme_set(theme_economist())  
  
p4
```

# ggthemes - Example

We can make our elections plot look as though it was published in the *Economist*.

```
install.packages("ggthemes")  
library(ggthemes)  
theme_set(theme_economist())  
p4
```



# ggplot themes

- We can also customize themes by altering individual elements; for example, by making axis titles disappear or by removing ticks on either axis
- A complete list of the graphical elements that can be modified via `theme()` is included with the [ggplot documentation](#)

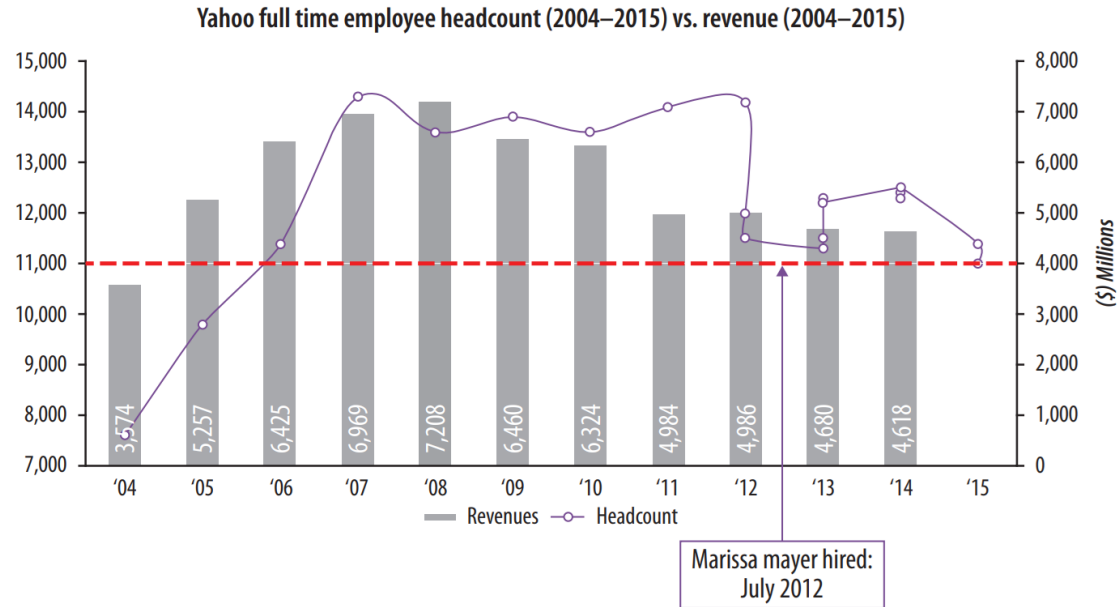
# **Activity: Fixing bad data visualizations**

# Activity

- By this point, our understanding of ggplot and of data visualization best practices in general should be sufficient such that we can recognize a ‘bad’ data visualization and create a better one
- This activity will put that to the test!
- **For each of the examples that follow:**
  - Why do you think Healy (2018) identifies this as a bad data visualization?
  - Apply the changes scripted in chapter 8.5 of Healy (2018)
  - What additional/different changes do you think should be made?

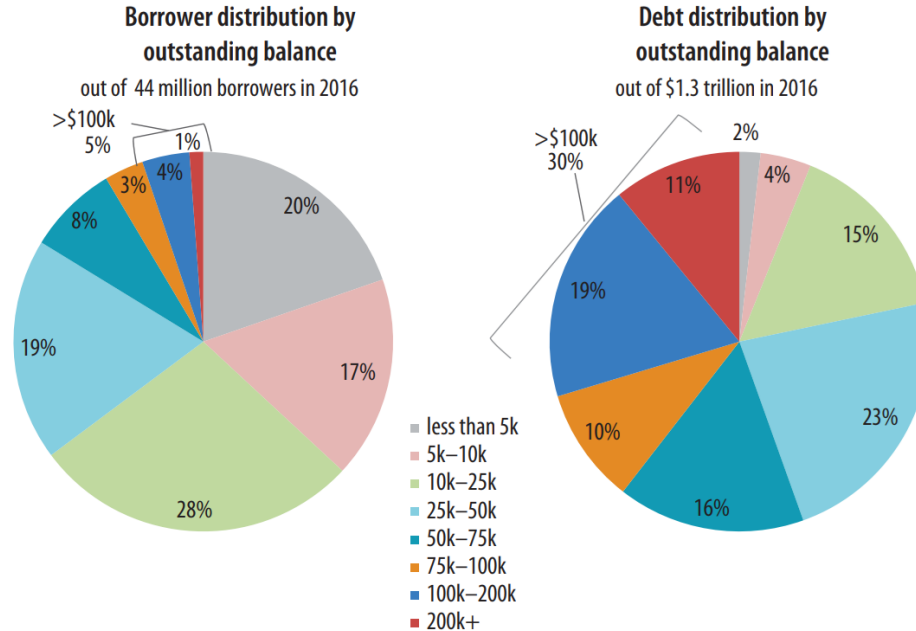


# Activity - Case study #1



Source: Company filings (10K), analyst calls

# Activity - Case study #2



[Page 224 of Healy \(2018\)](#)

## Next...

- Accessible Data Visualization
- How can we modify colour, text, and image descriptions to make our data visualizations accessible to as many people as possible?