Week 3 – Loading Data, Tidyverse, HateCrimes Tutorial and the Elevator Pitch
Saidi - DATA 110

- Tidyverse using categorical and continuous data
- Load data from 3 different sources
- 1. Load Data from a URL
    - Making side-by-side boxplots
- 2. Load from Built-in Dataset
    - Make Scatterplot with Alt tags for accessibility and screen readers
- 3. Load from CSV file saved on your computer (working directory)
    - Cleaning the data
    - Mutate
    - Pivot from Wide to Long
    - Facet-wrap
- Communication techniques and the **elevator pitch**
- Hate Crimes Tutorial for Homework
- Homework Week 3

Tidyverse

When you run `install.packages("tidyverse")`, R installs the following packages (and more) for you in one simple step:

- `ggplot2`
- `dplyr`
- `tidyr`
- `readr`
- `purrr`
- `tibble`
- `hms`
- `stringr`
- `lubridate`
- `forcats`
- `DBI`
- `haven`
- `jsonlite`
- `readxl`
- `rvest`
- `xml2`
- `modelr`
- `broom`

Tidyverse Using Categorical and Continuous Data

Set your working directory

Now we can set the working directory to this folder by selecting from the top menu `Session>Set Working Directory>To Source File Location.` (Doing so means we can load the files in this directory without having to refer to the full path for their location, and anything we save will be written to this folder.)

Notice how this code appears in the console (or something like this):

```
setwd("~/Desktop/week3notes")
```

Save your data

The panel at top right has two tabs, the first showing the `Environment`, or all of the "objects" loaded into memory for this R session. We can save this as well, so we don't have to load and process data again if we return to return to a project later.

(The second tab shows the `History` of the operations you have performed in RStudio.)

Comment your code

Anything that appears on a line after # will be treated as a comment, and will be ignored when the code is run. Get into the habit of commenting your code: Don't trust yourself to remember what it does!

Load Data from Three Different Sources

In the following notes, you will load data from three different sources:

1. Directly from a URL
2. Directly from pre-built datasets in R
3. From a file you save in your own folder

Reading Data in 3 Ways

In the following notes, you will load data directly from a URL, directly from pre-build datasets in R, and finally from a file you save in your own folder.

Load Data Method 1: Load Data from a URL

You can load data from a folder or you can load data directly from a URL. The next example loads the dataset, "Test Scores", directly from the URL where it resides.

```
library(tidyverse)  # you will use the readr package in tidyverse to read in this data
allscores <- read_csv("https://goo.gl/MJyzNs")
head(allscores)
```

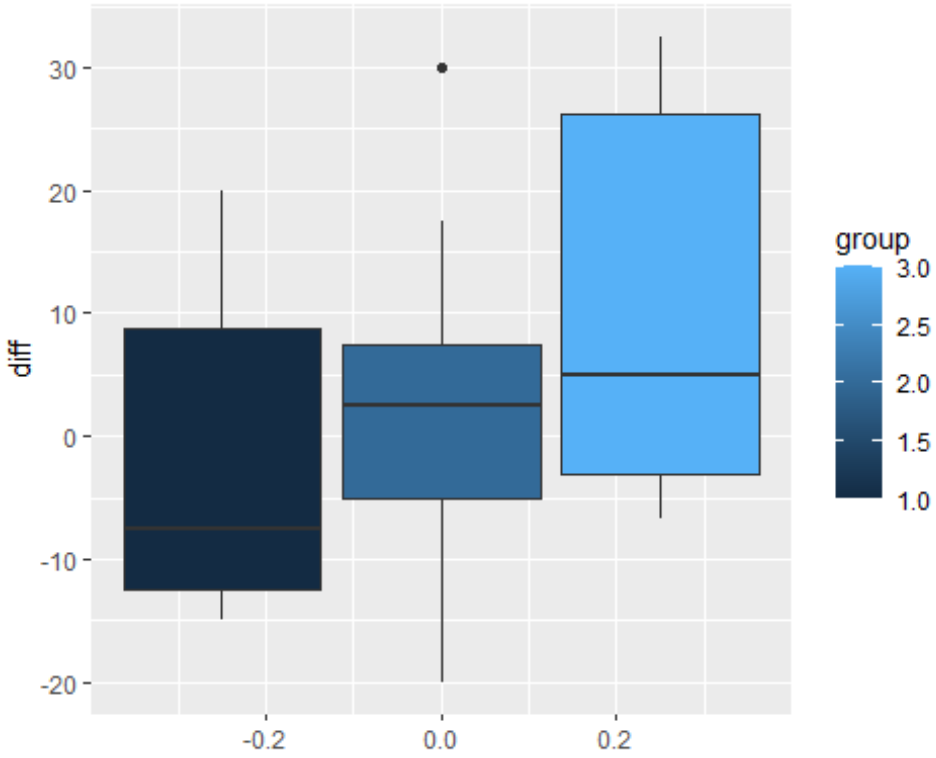
```
# A tibble: 6 × 4
  group   pre  post  diff
  <dbl> <dbl> <dbl> <dbl>
1     3  45    47.5   2.5
2     1  65    65     0
3     1  52.5  40   -12.5
4     1  65    52.5 -12.5
5     1  60    52.5  -7.5
6     2  55    35   -20
```

Notice R interprets the variable “group” as continuous values (col_double). We will fix this later. The command “dim” provides the dimensions of the data, which are 22 observations (rows) by 4 variables (columns).

Use Side-by-Side Boxplots

Here is some easy code to create 3 groups of boxplots with some easy-to-access data, filled by group. Since the groups are discrete, you can get rid of the shading.

```
boxpl <- allscores |>
  ggplot(aes(y=diff, group = group, fill = group)) +
  geom_boxplot()
boxpl
```

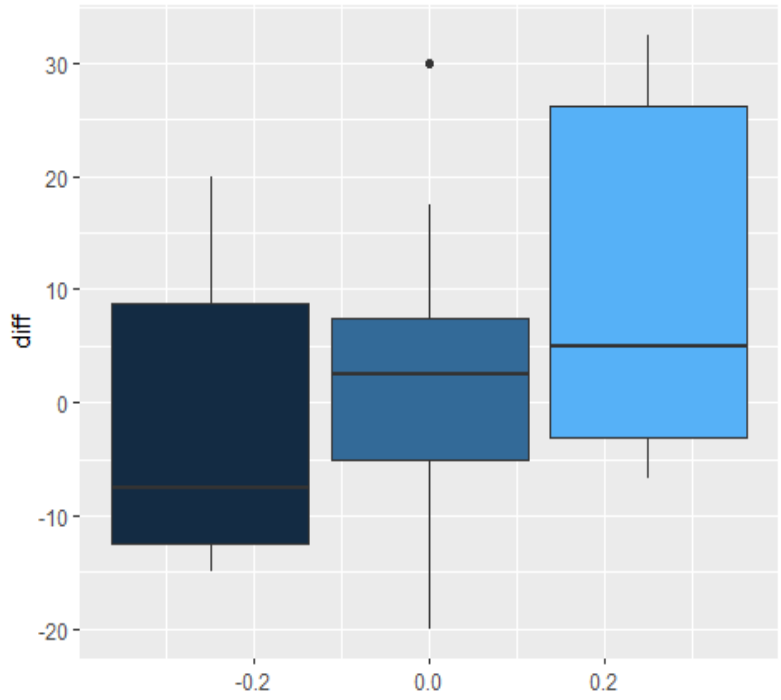


Side-by-Side boxplots of pre- and post-test scores

Notice that the legend give a continuous range of values for the scores, even though the scores are only 1, 2, or 3. The code guides(fill = FALSE) will get rid of the legend. Also, the x-axis labels make no sense. We will deal with that later.

Try to correct for the misrepresented legend

```
boxpl2 <- boxpl + guides(fill = FALSE)
boxpl2
```



Add your own color choices for the 3 different boxes

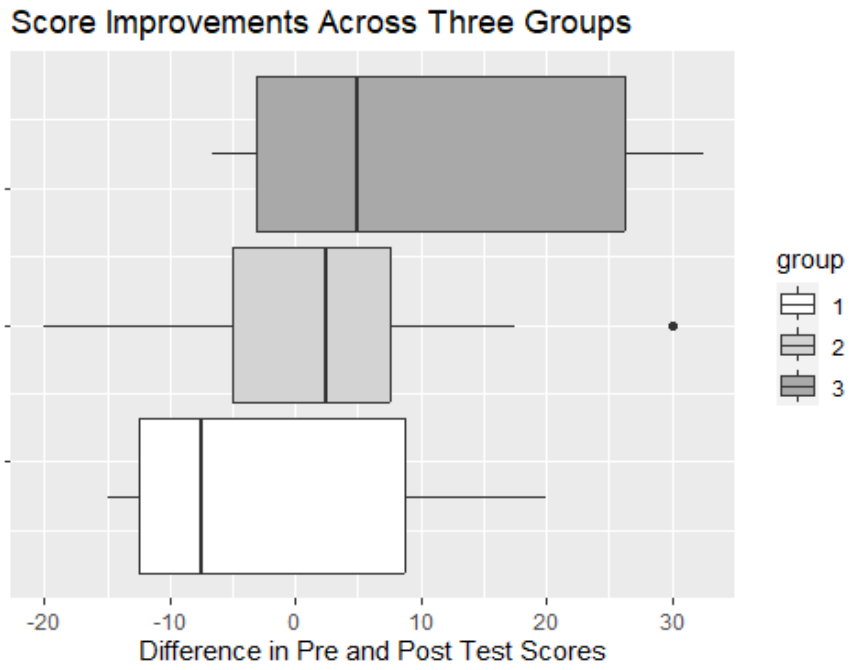
Ensure that the groups are considered as factors, rather than numbers.

Use as.factor as another way to ensure numerical values are read as categorical

```
allscores$group <- as.factor(allscores$group)
head(allscores)
```

Then manually fill with the 3 colors: white, light gray, and dark gray. Make the boxplots orient horizontally.

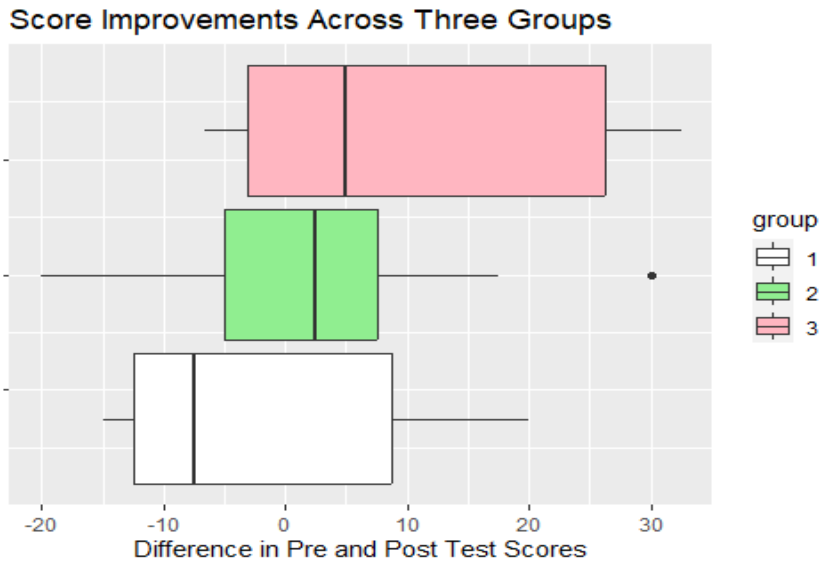
```
boxpl3 <- allscores |>
  ggplot() + geom_boxplot(aes(y=diff, group=group, fill=group)) +
  scale_fill_manual(values=c("white","lightgray","darkgray")) +
  theme(axis.text.y=element_blank()) + # Remove the useless y-axis tick values.
  labs(title = "Score Improvements Across Three Groups",
          y = "Difference in Pre and Post Test Scores") +
  coord_flip()
boxpl3
```



Another way to convert to factor levels

Use the factor function:

```
boxpl3 <- allscores |>
  mutate(group=factor(group, levels=c("1","2","3"), ordered=TRUE)) |>
  ggplot() + geom_boxplot(aes(y=diff, group=group, fill=group)) +
  scale_fill_manual(values=c("white","lightgreen","lightpink")) +
  theme(axis.text.y=element_blank()) +
  labs(title = "Score Improvements Across Three Groups",
          y = "Difference in Pre and Post Test Scores") +
  coord_flip()
boxpl3
```



Load Data Method 2: Use prebuilt dataset

We will use the penguins dataset that is pre-build in the "palmerpenguins" package to create scatterplots.

Load the package and feed data into global environment

```
library(palmerpenguins)
data("penguins")  # loads the penguins dataset into your global environment
```

It is essential to recognize that variables may be: int (integer), num (numeric), or double vs char (character) and factor (for categories)

Typically, chr or factor are used for discrete variables and int, dbl, or num for continuous variables.

Use head() function to view the tibble and variable types

```
head(penguins)
```

```
# A tibble: 6 × 8
  species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
  <fct>   <fct>              <dbl>         <dbl>             <int>       <int>
1 Adelie  Torgersen           39.1          18.7               181        3750
2 Adelie  Torgersen           39.5          17.4               186        3800
3 Adelie  Torgersen           40.3          18                 195        3250
4 Adelie  Torgersen           NA            NA                  NA          NA
5 Adelie  Torgersen           36.7          19.3               193        3450
6 Adelie  Torgersen           39.3          20.6               190        3650
# ℹ 2 more variables: sex <fct>, year <int>
```

Combine fig.cap for the Figure label and fig.alt for the alt text

fig.cap and fig.alt are YAML code embedded in chunks - these are tags for screen readers to improve accessibility in your document. The colors darkorange, purple, and cyan4 improve visibility of colors for colorblind access.

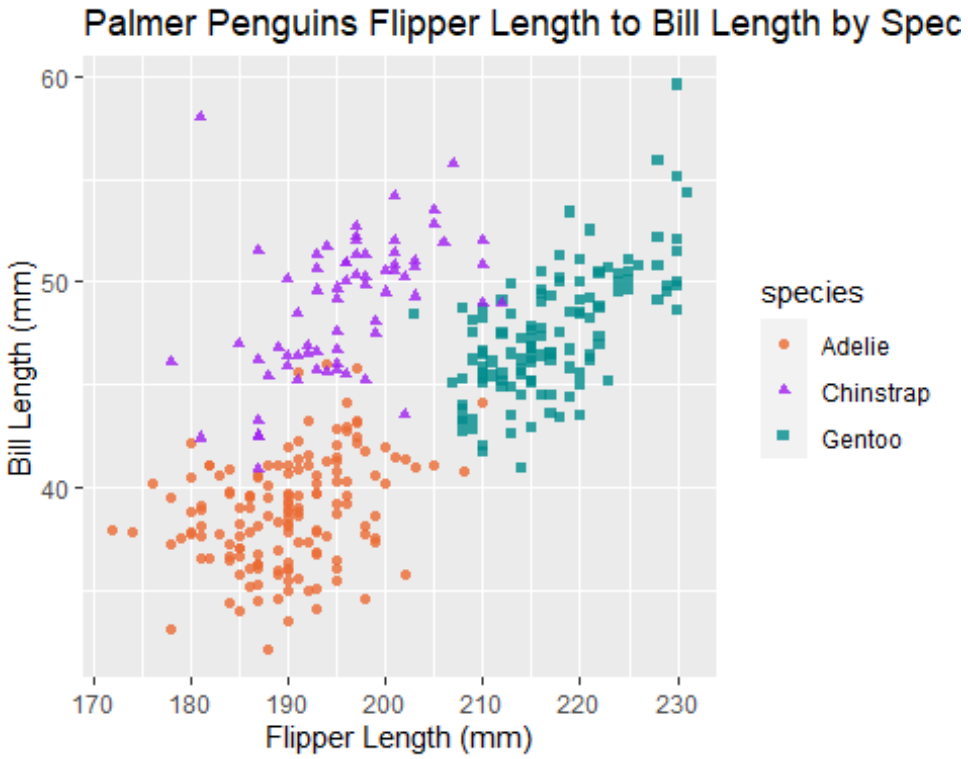
Example of using fig.cap and fig.alt in the next chunk:

{r fig.cap="Bigger flippers, bigger bills", fig.alt = "Scatterplot of flipper length by bill length of 3 penguin species, where we show penguins with bigger flippers have bigger bills."}

```
scatter1 <- ggplot(data = penguins,
                   aes(x = flipper_length_mm,
                       y = bill_length_mm,
                       color = species)) +
  geom_point(aes(shape = species), alpha = 0.8) +
  scale_color_manual(values = c("#eb6b34","purple","cyan4"))  +
  labs(title = "Palmer Penguins Flipper Length to Bill Length by Species",
```

```
       x = "Flipper Length (mm)",
       y = "Bill Length (mm)")
scatter1
```

```
Warning: Removed 2 rows containing missing values (`geom_point()`).
```



Bigger flippers, bigger bills

Notice both SHAPE and COLOR are used to help the viewer see the differences in the 3 penguin species.

Load Data Method 3: read in a saved csv from your computer

This is the most important skill, to be able to read in a csv file that is saved on your computer in order to load it into R Studio to work with.

Find a dataset, save it in a folder on your computer. This folder is called your *working directory*. You will need to **set your working directory**.

Clean data headings and variable names

Data often is messy and not ready to use right away!

The data might require some cleaning. Here are some important points to check:

1. Be sure the format is .csv
2. Be sure there are no spaces between variable names (headers).
3. Set all variable names to lowercase so you do not have to keep track of capitalizing.

Load the libraries

```
# install.packages("zoo")
library(zoo)  # this package will help us re-format the period to be a useable date.


Attaching package: 'zoo'

The following objects are masked from 'package:base':

    as.Date, as.Date.numeric
```

Loading Data from a Working Directory

The easiest way to find out what your current working directory is, use the command getwd().

```
getwd()

[1] "C:/Users/rsaidi/Dropbox/Rachel/MontColl/DATA110/Notes"
```

This command shows you (in your console below) the path to your directory. My current path is: [1] "C:/Users/rsaidi/Dropbox/Rachel/MontColl/DATA110/Notes"

If you want to change the path, there are several ways to do so. I find the easiest way to change it is to click the "Session" tab at the top of R Studio. Select "Set Working Directory", and then arrow over to "Choose Directory". At this point, it will take you to your computer folders, and you need to select where your data is held. I suggest you create a folder called "Datasets" and keep all the data you load for this class in that folder.

Notice that down in the console below, it will show the new path you have chosen: setwd("C:/Users/rsaidi/Dropbox/Rachel/MontColl/Datasets/Datasets"). At this point, I copy that command and put it directly into a new chunk.

Load the data

The following data comes from New York Fed (https://www.newyorkfed.org/microeconomics/hhdc.html) regarding household debt for housing and non-housing expenses. Debt amounts are in $ trillions for all US households.

Download this dataset, Household_debt, from http://bit.ly/2P3084E and save it in your dataset folder. Change your working directory to load the dataset from YOUR folder. Then run this code.

```
# be sure to change this to your own directory
setwd("C:/Users/rsaidi/Dropbox/Rachel/MontColl/Datasets/Datasets")
household <- read_csv("household_debt.csv")

Rows: 64 Columns: 8
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (1): Period
dbl (7): Mortgage, HE Revolving, Auto Loan, Credit Card, Student Loan, Other...
```

```
head(household)
```

```
# A tibble: 6 × 8
  Period Mortgage `HE Revolving` `Auto Loan` `Credit Card` `Student Loan` Other
  <chr>     <dbl>          <dbl>       <dbl>         <dbl>          <dbl> <dbl>
1 03:Q1      4.94           0.24        0.64          0.69           0.24  0.48
2 03:Q2      5.08           0.26        0.62          0.69           0.24  0.49
3 03:Q3      5.18           0.27        0.68          0.69           0.25  0.48
4 03:Q4      5.66           0.3         0.7           0.7            0.25  0.45
5 04:Q1      5.84           0.33        0.72          0.7            0.26  0.45
6 04:Q2      5.97           0.37        0.74          0.7            0.26  0.42
# ⓘ 1 more variable: Total <dbl>
```

Clean data headings and variable names

Very soon, you will find data from other sources. The data will require some cleaning. Here are some important points to check: 1. Be sure the format is .csv 2. Be sure there are no spaces between variable names (headers). 3. Set all variable names to lowercase so you do not have to keep track of capitalizing.

Here are some useful cleaning commands:

Make all headings (column names) lowercase. Remove all spaces between words in headings and replace them with underscores with the gsub command. Then look at it with “head”.

```
names(household) <- tolower(names(household))
names(household) <- gsub(" ","_",names(household))
# gsub will remove spaces in between words in the headers and replace them with underscor
e
head(household)
```

```
# A tibble: 6 × 8
  period mortgage he_revolving auto_loan credit_card student_loan other total
  <chr>     <dbl>        <dbl>     <dbl>       <dbl>        <dbl> <dbl> <dbl>
1 03:Q1      4.94         0.24      0.64        0.69         0.24  0.48  7.23
2 03:Q2      5.08         0.26      0.62        0.69         0.24  0.49  7.38
3 03:Q3      5.18         0.27      0.68        0.69         0.25  0.48  7.56
4 03:Q4      5.66         0.3       0.7         0.7          0.25  0.45  8.07
5 04:Q1      5.84         0.33      0.72        0.7          0.26  0.45  8.29
6 04:Q2      5.97         0.37      0.74        0.7          0.26  0.42  8.46
```

Look at the dimensions and the structure of the data. Note that it will be listed as a tibble.

```
dim(household)
```

```
[1] 64  8
```

Mutate

Mutate is a powerful command in tidyverse. It creates a new variable (column) in your dataset. In our dataset, “period” is not anything useful if we want to plot chronological data. So we will use mutate from “tidyverse” with the package “zoo” to create a useable date format.

Create a new DATE variable from “period”

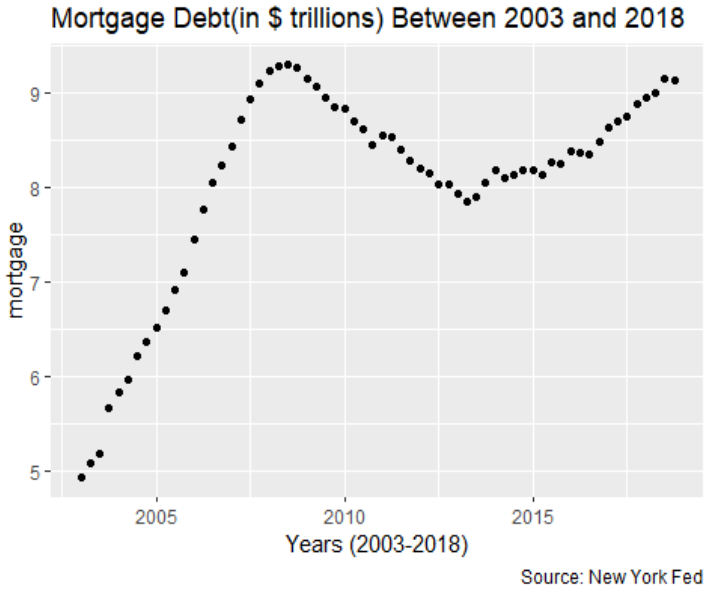
You should see that there are 64 observations and 8 variables. All variables are “col_double” (continuous values) except “period”, which is interpreted as characters.We need to use the library “zoo” package to fix the unusual format of the “period”. We will mutate it to create a new variable, date.

```
household_debt <- household |>
   mutate(date = as.Date(as.yearqtr(period, format = "%y:Q%q")))
head(household_debt)

# A tibble: 6 × 9
  period mortgage he_revolving auto_loan credit_card student_loan other total
  <chr>     <dbl>        <dbl>     <dbl>       <dbl>        <dbl> <dbl> <dbl>
1 03:Q1      4.94         0.24      0.64        0.69         0.24  0.48  7.23
2 03:Q2      5.08         0.26      0.62        0.69         0.24  0.49  7.38
3 03:Q3      5.18         0.27      0.68        0.69         0.25  0.48  7.56
4 03:Q4      5.66         0.3       0.7         0.7          0.25  0.45  8.07
5 04:Q1      5.84         0.33      0.72        0.7          0.26  0.45  8.29
6 04:Q2      5.97         0.37      0.74        0.7          0.26  0.42  8.46
# ⓘ 1 more variable: date <date>
```

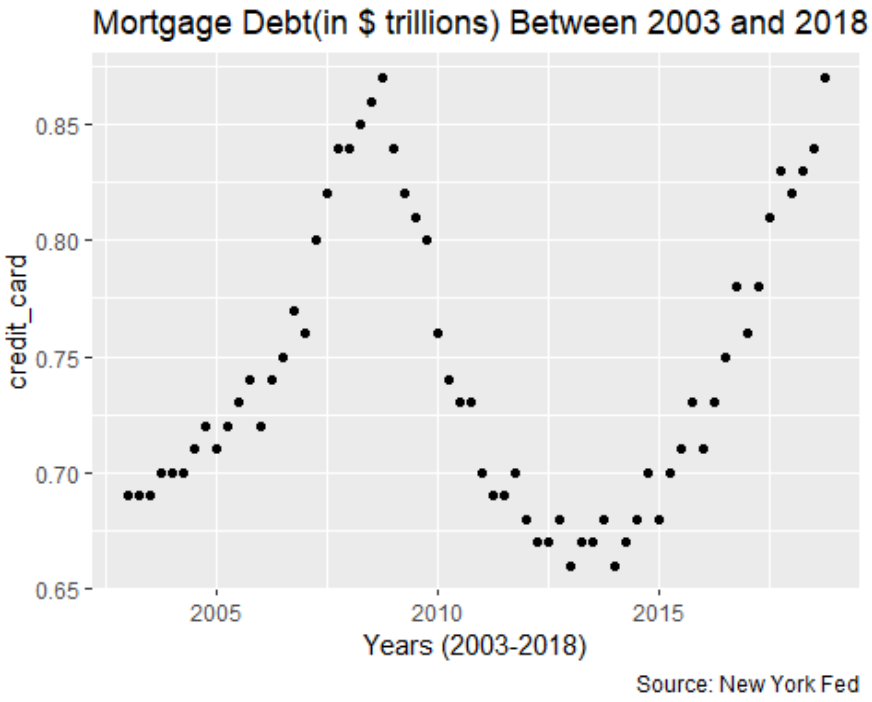
Plot various loan types

```
plot1 <- household_debt |>
  ggplot(aes(date, mortgage)) +
  geom_point() +
  labs(title = "Mortgage Debt(in $ trillions) Between 2003 and 2018",
       x = "Years (2003-2018)",
       caption = "Source: New York Fed")
plot1
```



Mortgage Debt Between 2003 and 2018

```
plot2 <- household_debt |>
  ggplot(aes(date, credit_card)) +
  geom_point() +
  labs(title = "Mortgage Debt(in $ trillions) Between 2003 and 2018",
       x = "Years (2003-2018)",
       caption = "Source: New York Fed")
plot2
```



Credit Card Debt Between 2003 and 2018

Pivot the table from “wide” to “long” format

This will enable use to use the “facet” function in the next step.

```
house_long <- household_debt |>
  pivot_longer(
        cols = 2:7,
    names_to = "debt_type",
    values_to = "debt_amnt")
head(house_long)
```

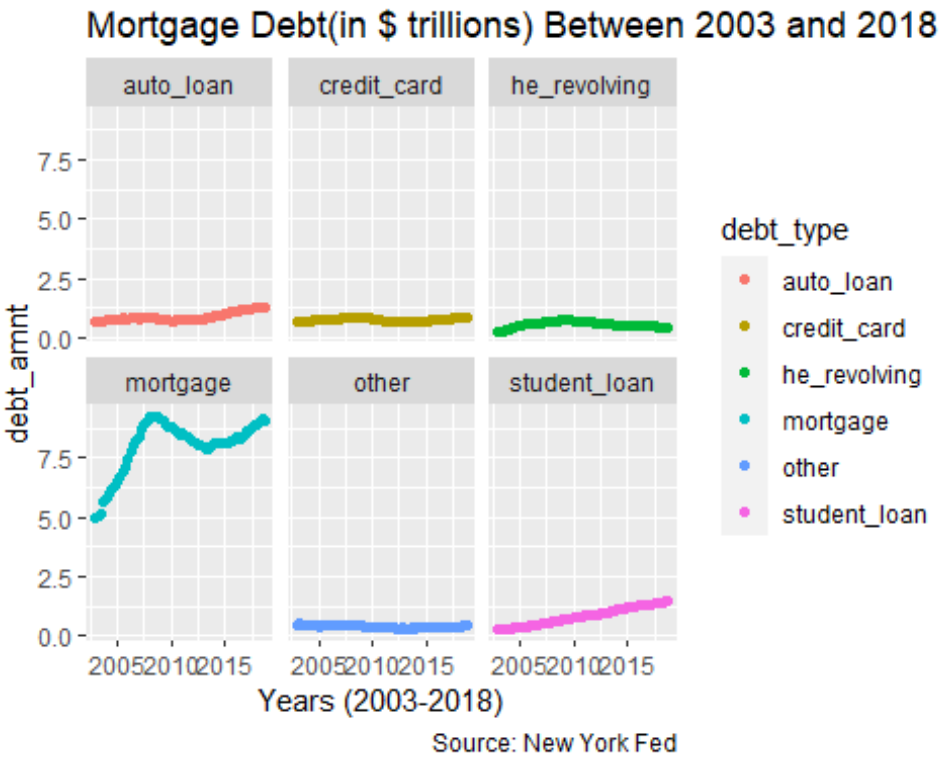
```
# A tibble: 6 × 5
  period total date       debt_type    debt_amnt
  <chr>  <dbl> <date>     <chr>            <dbl>
1 03:Q1   7.23 2003-01-01 mortgage          4.94
2 03:Q1   7.23 2003-01-01 he_revolving      0.24
3 03:Q1   7.23 2003-01-01 auto_loan         0.64
4 03:Q1   7.23 2003-01-01 credit_card       0.69
5 03:Q1   7.23 2003-01-01 student_loan      0.24
6 03:Q1   7.23 2003-01-01 other             0.48
```

Use “facet_wrap” to show all types of debt together

Facet_wrap allows you to plot all variables together for comparison.

In order to do this, you have to “reshape the”data from a wide format to a long format. Use **gather** from tidyr package to do this.

```
plot3 <- house_long %>%
  ggplot(aes(x=date, y= debt_amnt))+
  geom_point(aes(color = debt_type))+
  facet_wrap(~debt_type) +
  labs(title = "Mortgage Debt(in $ trillions) Between 2003 and 2018",
       x = "Years (2003-2018)",
       caption = "Source: New York Fed")
plot3
```



Facet Wrap of All Types of Household Debt 2003-2018

Communication Techniques and the Elevator Pitch

Consider various forms of communication:

- Impression management strategies
- Interpersonal communication
- Situational communication

- Bias
- The elevator pitch
- Data for Good

Impression management - also known as self-presentation or projecting a particular image for a particular audience.

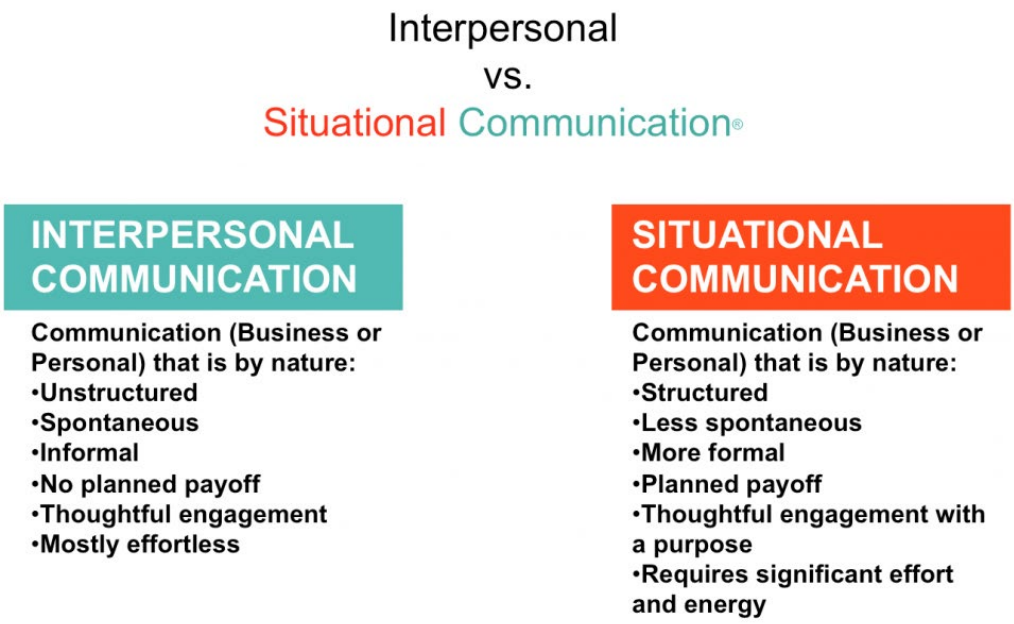
It is a conscious or subconscious process to influence others' perceptions by controlling information in social interaction. In 1959, it was first conceptualized by Erving Goffman, a Canadian-American sociologist, social psychologist, and writer. Impression management could be another's perception of a certain person (including you), a material possession, or an event. The theory goes on to explain that we try to make the perception consistent with our goals.

➔ **Example:**
At an important soccer game, a young player wants to showcase herself in the best light possible, because there are college recruiters watching. She will try and perform her best to show off her skills. Her main goal may be to impress the college recruiters in a way that maximizes her chances of being chosen for a college team rather than winning the game.

How often have you wondered what someone will think of you if you do this or that, or if you don't do it? We strive to have others view us positively, because we tend to put emphasis on other views in ways that impact our self-esteem. As far as marketing goes, businessmen are going to present a product in the best light possible. Their job relies on managing the impressions of the audience in specific ways that boosts revenues. Also, in their understanding of human behavior, they might even imply that if you own this product you may be more liked by others.

Interpersonal Communication is the process by which people exchange ideas, feelings and thoughts using verbal and non-verbal massages. People usually engage in interpersonal communication to generate shared meanings and accomplish social goals. Typically, this is a face-to-face interaction because both spoken and body language are used to communicate. Interpersonal communication can take place in a business or personal setting and is generally thought to be more social in nature. In other words, it tends to be more spontaneous and informal – without a planned payoff in mind. Think of it as a thoughtful engagement that's most often effortless.

Situational Communication - a clear, concise and focused communication and relating strategy that maximizes a minimum amount of time to achieve successful results and effective relationships. Although it is often used in personal settings, situational communication is mostly reserved for business and professional interactions (especially those that are challenging, where the points of view, interests or preferred solutions of the communicators are different or in conflict). It's is the kind of communication leaders must be capable of planning for and executing in order to be both successful and effective. It represents thoughtful engagement – *with a purpose*. These types of interactions are more formal, structured and well planned, requiring significant energy and focus (like your data presentations).



Bias in communication - the disproportionate weight in favor of or against one thing, person, or group compared with another, usually in a way considered to be unfair. Biases can be learned implicitly within cultural contexts. Bias is a personal and sometimes unreasoned judgment or prejudice.

As we saw at the beginning of these notes, **bias** may occur due to an unintended consequence of who is creating the technology and how they are designing the technology. As data scientists, we must be very careful about the biases we bring into our data analyses and data visualizations.

The Elevator Pitch

An elevator pitch or elevator speech is a short overview of your business, products or services, and is typically used in business settings such as face-to-face networking. An elevator pitch can be one of the simplest yet most powerful tools for a small business owner.

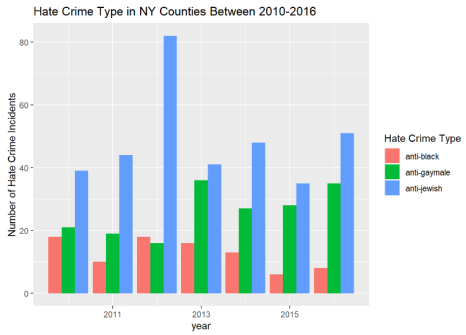
An elevator pitch is meant to be short, and as the name implies, delivered in the time it takes to complete your average elevator ride. The length can vary, but you typically want to be able to present your elevator pitch comfortably without rushing in under two minutes, ideally in under one minute. Your goal length should be 150-250 words.

Here is my elevator pitch, so you can see an example of what I expect from you:

http://youtu.be/xFRX0e6nXVM?hd=1

Material for Your Homework:
Hate Crimes Tutorial (More exploration with categorical variables)

In class, we will discuss the dataset, the limitations of the dataset, and the code used to create the visualizations and calculations.



Flawed hate crime data collection - we should know how the data was collected

(Nathan Yau of Flowing Data, Dec 5, 2017)

Data can provide you with important information, but when the collection process is flawed, there's not much you can do. Ken Schwencke, reporting for ProPublica, researched the tiered system that the FBI relies on to gather hate crime data for the United States:

"Under a federal law passed in 1990, the FBI is required to track and tabulate crimes in which there was 'manifest evidence of prejudice' against a host of protected groups, regardless of differences in how state laws define who's protected. The FBI, in turn, relies on local law enforcement agencies to collect and submit this data, but can't compel them to do so."

This is a link to the ProPublica Article: https://www.propublica.org/article/why-america-fails-at-gathering-hate-crime-statistics

Here is a data visualization of where hate crimes do NOT get reported around the country (Ken Schwencke, 2017): https://projects.propublica.org/graphics/hatecrime-map

Hate Crimes Tutorial https://rpubs.com/rsaidi/1000514

For your homework, you will follow the tutorial, copying and pasting text and code. You will explore what the hatecrimes dataset can and cannot explain. You will also learn to access census population data and join it with a subset of the hatecrimes dataset to calculate proportions of subgroups.

Week 3 Homework Assignment

1. **(Ungraded)** Reread these notes and try copying, pasting, and running the code provided. Remember – you are responsible for all code presented in these notes.

2. **(Worth up to 10 points) Elevator Pitch Discussion Post.**

   Choose one of the two articles below on how to create an Elevator Pitch. Decide whether you want to pitch an idea or yourself to be hired for a position at a company. Create that elevator pitch**. It should take no more than one minute to speak, and it should be persuasive and compelling.** Your goal length should be about 100-200 words. You will record your elevator pitch in a video and post it on Blackboard Discussions.

   The video of your pitch will be due by 11:59 pm on Friday, Sept 20th. Then you will view your classmates' pitches and **respond to at least 3 classmates' pitches by 11:59 pm on Sunday, Sept 22nd.**

   You will find the two possible articles in this google drive: http://bit.ly/Data110Articles

   a. *Seven Steps for Writing a Powerful Elevator Pitch*
   b. *How to Write an Elevator Pitch that Sounds like You AND Gets You the Job*

3. **(Worth up to 10 points) Recreate the Hate Crimes Tutorial** https://rpubs.com/rsaidi/1000514

   - Copy, paste and run all code for the HateCrimes Tutorial in your own markdown or quarto file. Pay attention to where you merge two datasets.

   - **At the end of your markdown/quarto file** with the hatecrimes code, include an essay of approximately 200-400 words which that answers the following questions:

     1. Write about the positive and negative aspects of this hatecrimes dataset.
     2. List 2 different paths you would like to (***hypothetically***) study about this dataset.
     3. Describe 2 things you would do to follow up after seeing the output from the hatecrimes tutorial.

   Publish your markdown file and submit the link in this Assignment Dropbox **by 11:59 pm on _____.** We will present your submissions in class next week.