Problem Set 2

Insert Name

Stat 108, Week 3

Collaborators

I collaborated with... (list names of collaborators here).

```
# Put all necessary libraries here
# We got you started!
library(tidyverse)
```

Due: Wednesday, February 15th at 10:00pm

Notes on Submitting

1. Please knit to pdf and submit that on Gradescope. When knitting to pdf, include `eval = FALSE` in any code chunks that contain an animated or interactive graph.
2. Please also knit to html and push that (along with the Rmd and any other relevant files) to your `work-username` GitHub repo so that the Graders can access the html document with your animated and interactive graphs. (So include `eval = TRUE` in any code chunks that contain an animated or interactive graph when knitting to html).

Goals of this lab

1. Further explore `ggplot2`.
2. Practice some data wrangling with `dplyr` and `forcats`.
3. Practice creating static maps with `ggmap`.
4. Practice incorporating animation into a graph with `gganimate`.
5. Practice incorporating interactivity into a graph with `plotly`.

Problems

We will continue to use the `crash_data` from P-Set 1 in this p-set.

```
# Read in the data
crash_data <- read_csv("https://raw.githubusercontent.com/harvard-stat108s23/materials/main/psets/data/
```

Problem 1

Watch the Glamour of Graphics video, a talk given by William Chase at RStudio::Conf 2020. Take one of your graphs from P-Set 1 and recreate the graph while incorporating at least three of William's suggestions (that aren't already implemented by default). Note: William doesn't provide `R` code so you will need to do some sleuthing to add these features.

State the suggestions that you incorporated and show us the original as well as the new version.

Problem 2

In this problem, you are going to animate one of your graphs from P-Set 1.

a. Recreate your scatterplot from Problem 2(a) from P-Set 1 here. (Just copy and paste the same code from 2(b).)

b. Use the `ggmap` package to grab a map of Cambridge and layer your scatterplot on top of the Cambridge map tile.

Suggestions:

1. You might need to play around with the zoom argument a bit when pulling the map.
2. Don't use the watercolor map type.
3. Play around with the color and size of your points so that they are visible on the map.

c. Swap out the points in your plot from (b) for images of a car by using the `ggimage` package!

d. Now facet your graph by hour of the day. Notice that the order of the hours is probably not what we want. Use `fct_relevel()` to reorder the categories before creating the graph.

```
# Toy example of fct_relevel
toy_data <- data.frame(animal = factor(c("cat", "mouse", "dog")),
                       cuteness = c(100, -2, 76))
levels(toy_data$animal)
```

```
## [1] "cat"   "dog"   "mouse"
```

```
toy_data <- mutate(toy_data,
                   animal = fct_relevel(animal, "dog", "mouse", "cat"))
levels(toy_data$animal)
```

```
## [1] "dog"   "mouse" "cat"
```

e. Return to your static graph from (c) but this time instead of faceting by the hour of the day, we want you to animate the graph where you transition over the hours of the day using the `transition_states()` function in `gganimate`. Pick reasonable arguments for `transition_states()` to control the speed of the animation. Additionally, we want you to include an animated label that provides the hour of the day. Hint: Use the `gganimate` cheat sheet to determine the correct label variable for the `transition_states()` function.

```
# Leave in the eval = FALSE when you knit to pdf
# Put in the eval = TRUE when you knit to html
```

f. Compare and contrast the effectiveness of the graphs in (d) and (e). Which is better at telling an interesting story? Justify your answer.

Problem 3

Let's take a static plot you made in P-Set 1 and add some animation and interactivity. (It needs to be a different graph than the one in Problem 2.)

a. Grab the code and recreate the static graph here.

b. Now add animation. In particular, consider

- How you want to transition from frame to frame.
- How the data should enter and exit the plot.
- The speeds of various aspects of the animation.
- Adding frame information to the title and/or subtitle.
- Whether or not the view should change as the animation progresses.

The gganimate cheatsheet will likely be helpful here!

```
# Leave in the eval = FALSE when you knit to pdf
# Put in the eval = TRUE when you knit to html
```

c. In what ways did the animation improve the plot? In what ways did the animation worsen the plot? Explain your reasoning.

d. Now take the static graph from (a) and make it interactive.

```
# Leave in the eval = FALSE when you knit to pdf
# Put in the eval = TRUE when you knit to html
```

e. In what ways did the interactivity improve the plot? In what ways did the interactivity worsen the plot? Explain your reasoning.

Problem 4

Let's use the crash data to practice some data wrangling with `dplyr` and `forcats`.

a. The `ambnt_light_descr` variable contains information on the ambient light at time of crash. Note how the variable contains multiple categories for "dark". Use `fct_collapse()` to collapse the three categories for dark while keeping the categories for dawn, daylight, and dusk; set the other categories to `NA` by using `other_level = "NULL"`.

```
#toy example of fct_collapse
soft_serve <- data.frame(flavor = factor(c("vanilla", "chocolate",
                                           "strawberry", "peach",
                                           "earl grey", "matcha")))

levels(soft_serve$flavor)
```

```
## [1] "chocolate"  "earl grey"  "matcha"     "peach"      "strawberry"
## [6] "vanilla"
```

```
soft_serve <- mutate(soft_serve,
                     flavor = fct_collapse(flavor,
                                           "fruit" = c("strawberry", "peach"),
                                           "tea" = c("earl grey", "matcha"),
                                           other_level = "other"))

levels(soft_serve$flavor)
```

```
## [1] "tea"   "fruit" "other"
```

b. Do crashes tend to happen in different places during the day as opposed to at night? Using only data from crashes that occurred either with dark or daylight levels of ambient light, recreate your scatterplot from Problem 2(a) from P-Set 1. Be sure that the distinction between daytime versus nighttime crashes is visible on your plot.

c. Based on your graph in part (b), does it seem like crashes tend to happen in different places during the day as opposed at night? Explain your reasoning.

d. From these data, can we conclude that crashes are more likely to happen during the day than at night? Explain your reasoning.

e. Let's examine which types of collisions are more likely to result in injury, as opposed to those which result in only property damage and no injury. Create a data visualization in which it is possible to compare the proportion of collisions resulting in injury versus not between angle, head-on, rear-end, sideswipe, and single-vehicle collisions (this is stored in the manner of collision variable, `manr_coll_descr`). The

variable `crash_severity_descr` contains information on whether there was an injury or not. Describe the story that the visualization communicates.

f. The variable `max_injr_svrty_cl` provides information on injury status descriptions and `speed_limit` indicates the speed limit at the location the crash took place. Create a wrangled dataframe that displays information on the hour the crash took place, the manner of collision, injury status (`max_injr_svrty_cl`), and speed limit for the crashes that happened in the dark in 2022. Arrange the observations in order from highest to lowest speed limit using another `dplyr` function: `arrange()`. When did the crashes going at the highest speed limit happen and were there injuries?

g. Create a wrangled dataframe using data on crashes that happened in the dark in 2022 which shows the number of injuries for each possible injury status. What was the most common type of injury reported at these crashes?

Problem 5

If you are new to git/GitHub, we recommend waiting until after Monday's lecture to do this problem.

In this problem, we will practice interacting with GitHub on the site directly and from RStudio. Do this practice on **your work_username repo** so that the graders can check your progress with Git and can access the other components of P-Set 2.

a. Let's practice creating and closing **Issues**. In a nutshell, **Issues** let us keep track of our work. Within your repo on GitHub.com, create an Issue entitled "Complete P-Set 2". Once P-Set 2 is done, close the **Issue**. (If you want to learn more about the functionalities of Issues, check out this page.)

b. The landing page of your repo is a ReadMe.md file. The "md" stands for markdown and is a common format for text files. The ReadMe file is meant to provide a quick introduction to the purpose and contents of a repo.

Edit the ReadMe of your repo to include your name and a quick summary of the purpose of the repo. You can edit from within GitHub directly or within RStudio. If you edit in RStudio, make sure to push your changes to GitHub.

c. Upload your P-Set 2 .Rmd, .pdf, .html and other relevant components to your repo on GitHub. You should still submit the .pdf on Gradescope but some of the other components the graders will grade directly in your GitHub repo.