Problem Set 2

Mariya Meleganich

Fall 2024

Collaborators

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

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

Due: September 29, 2024 at 11:59pm

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.

New Edits commented in Code

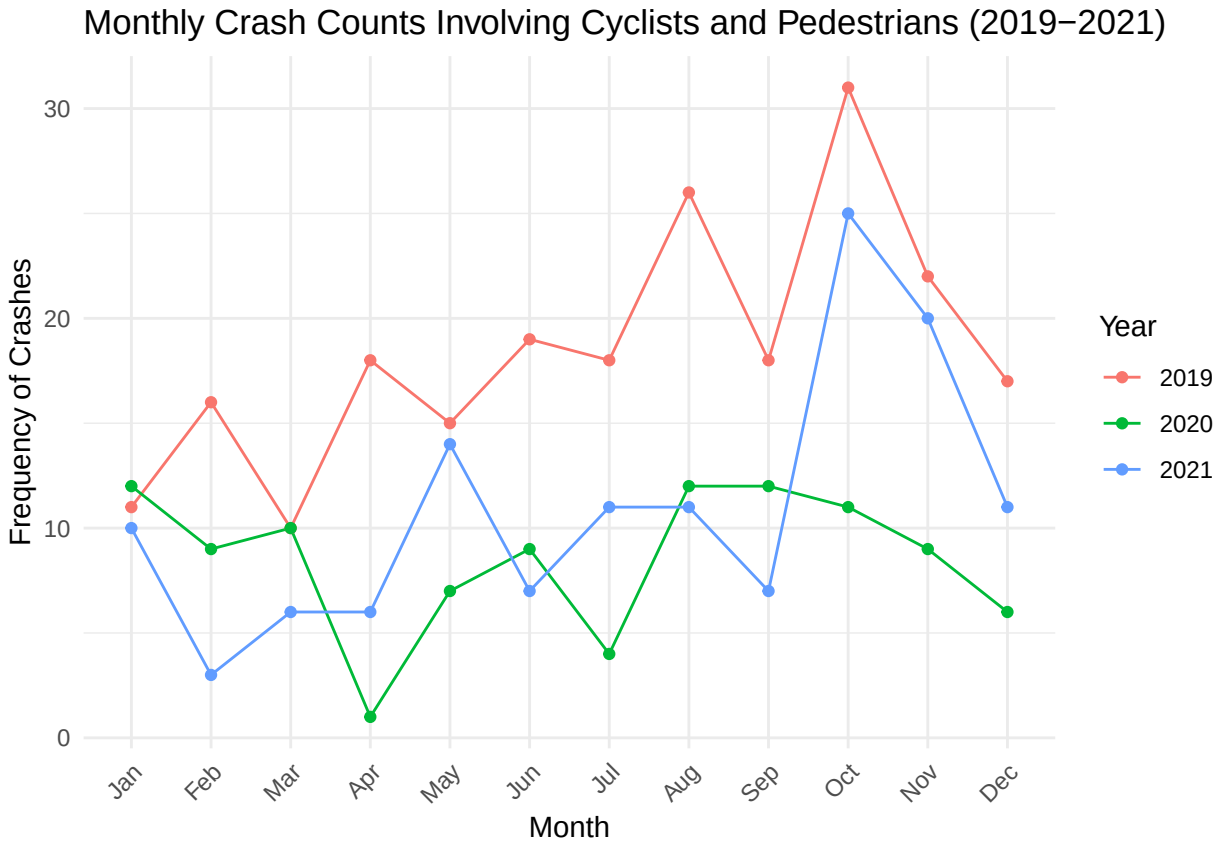
```r
#ORIGINAL GRAPH FROM PSET 1

library(ggplot2)
library(dplyr)

crash_data <- mutate(crash_data,
month = month(mdy(crash_date), label = TRUE))

crash_data <- crash_data[crash_data$year %in% 2019:2021,]
monthly_counts <- count(crash_data, month, year)

ggplot(monthly_counts, aes(x = month, y = n, color = factor(year), group = year)) +
  geom_line() +
  geom_point() +
  labs(title = "Monthly Crash Counts Involving Cyclists and Pedestrians (2019-2021)",
       x = "Month",
       y = "Frequency of Crashes", color = "Year") +
theme_minimal() +
scale_x_discrete(limits = month.abb) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



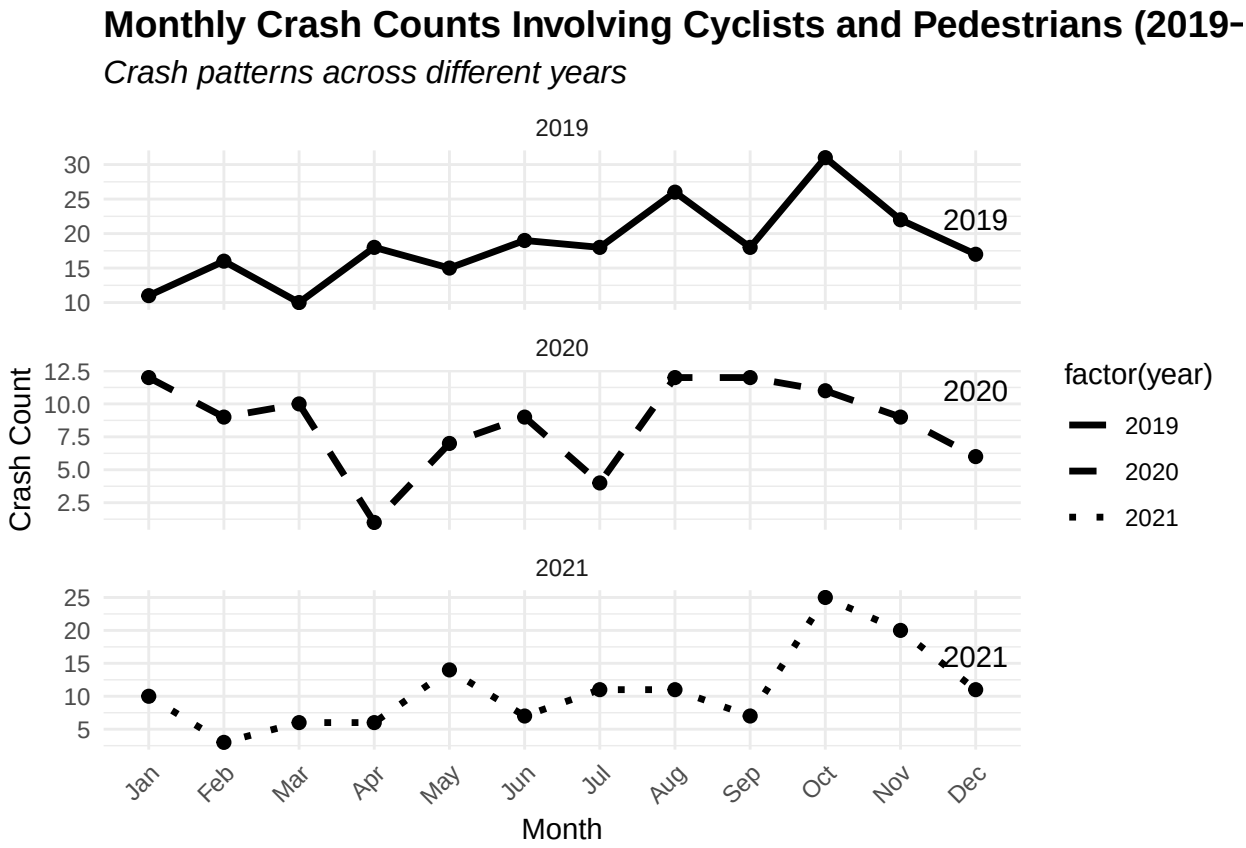

```r
#NEW GRAPH WITH EDITS

library(ggplot2)
library(dplyr)
library(lubridate)
```

```r
crash_data <- mutate(crash_data,
                     month = month(mdy(crash_date), label = TRUE))

crash_data <- crash_data[crash_data$year %in% 2019:2021,]
monthly_counts <- count(crash_data, month, year)

#plot with direct labels and faceting
ggplot(monthly_counts, aes(x = month, y = n, group = year)) +
  geom_line(aes(linetype = factor(year)), size = 1.2) + # Different line types
  geom_point(size = 2) +
  labs(
    title = "Monthly Crash Counts Involving Cyclists and Pedestrians (2019-2021)",
    subtitle = "Crash patterns across different years",
    x = "Month",
    y = "Crash Count"
  ) +
  scale_linetype_manual(values = c("solid", "dashed", "dotted")) + # Custom line types
  scale_x_discrete(limits = month.abb) +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),
    plot.title = element_text(face = "bold", size = 14),
    plot.subtitle = element_text(face = "italic", size = 12)
  ) +
  # Directly label the lines for clarity
  geom_text(data = monthly_counts %>% filter(month == "Dec"),
            aes(label = year, y = n + 5),
            size = 4, hjust = 0.5) +
  facet_wrap(~ year, scales = "free_y", ncol = 1) # Facet by year
```



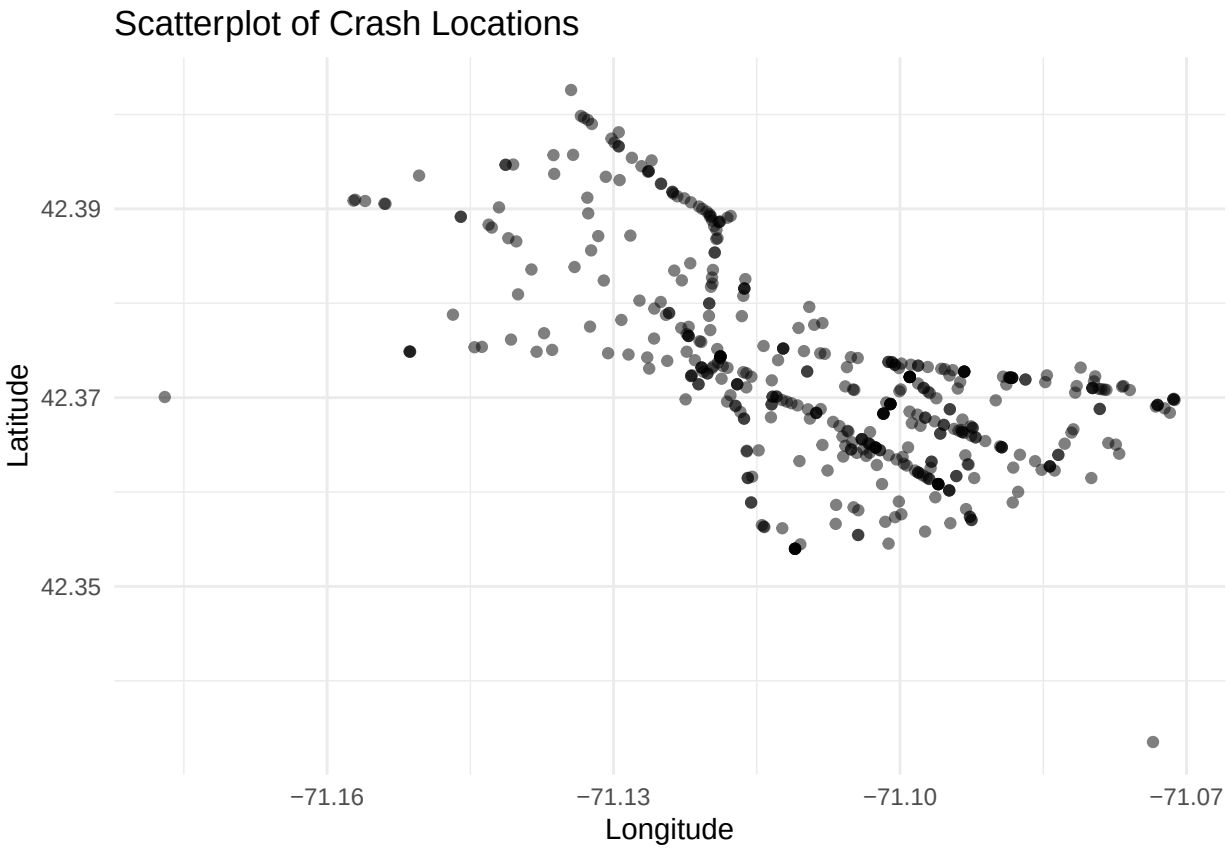
Problem 2

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).)

```
library(ggplot2)

ggplot(crash_data, aes(x = lon, y = lat)) +
  geom_point(alpha = 0.5, size = 1.5) +
  labs(title = "Scatterplot of Crash Locations",
       x = "Longitude",
       y = "Latitude") +
  theme_minimal()
```



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

Suggestions:

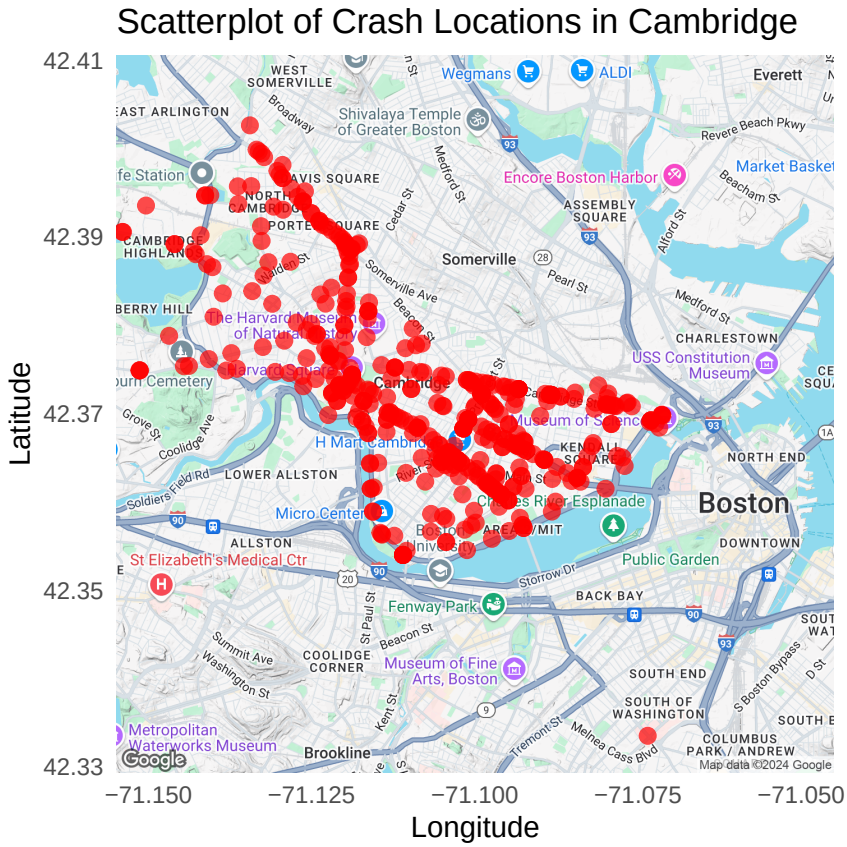
1. You might need to play around with the zoom argument a bit when pulling the map.
2. Don't use the watercolor map type.
3. Play around with the color and size of your points so that they are visible on the map.

```
library(ggmap)
library(ggplot2)

register_google(key = "AIzaSyDpEMlwKKTTUF3rxw33H59VYXJJ9vaH0oM")

cambridge_map <- get_map(location = c(lon = -71.1, lat = 42.37), zoom = 13, maptype = "terrain")

ggmap(cambridge_map) +
  geom_point(data = crash_data, aes(x = lon, y = lat),
             alpha = 0.7, color = "red", size = 2.5) +
  labs(title = "Scatterplot of Crash Locations in Cambridge",
       x = "Longitude",
       y = "Latitude") +
  theme_minimal()
```



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

```
library(ggplot2)
library(ggmap)
library(ggimage)

register_google(key = "AIzaSyDpEM1wKKTTUF3rxw33H59VYXJJ9vaH0oM")

cambridge_map <- get_map(location = "Cambridge, MA", zoom = 14, maptype = "roadmap")

car_image_url <- "https://media.istockphoto.com/id/828695144/vector/red-car.jpg?s=612x612&w=0&k=20&c=W7

ggmap(cambridge_map) +
  geom_image(data = crash_data, aes(x = lon, y = lat, image = car_image_url),
             size = 0.05) +
  labs(title = "Crash Locations on Cambridge Map (with Car Icons)",
       x = "Longitude",
       y = "Latitude") +
  theme_minimal()
```

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.

```
# Load necessary libraries
library(ggplot2)
library(ggmap)
library(ggimage)
library(forcats)
```

```
library(dplyr)

register_google(key = "AIzaSyDpEMlwKKTTUF3rxw33H59VYXJJ9vaH0oM")

cambridge_map <- get_map(location = "Cambridge, MA", zoom = 14, maptype = "roadmap")

car_image_url <- "https://media.istockphoto.com/id/828695144/vector/red-car.jpg?s=612x612&w=0&k=20&c=W7
```

```
crash_data <- crash_data %>%
  mutate(hour = format(strptime(crash_time_2, format="%H:%M:%S"), "%H"))

crash_data$hour <- fct_relevel(crash_data$hour, as.character(0:23))

ggmap(cambridge_map) +
  geom_image(data = crash_data, aes(x = lon, y = lat, image = car_image_url),
             size = 0.05) +
  labs(title = "Crash Locations on Cambridge Map (with Car Icons)",
       x = "Longitude",
       y = "Latitude") +
  theme_minimal() +
  facet_wrap(~ hour, ncol = 4)
```

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.

```
library(gganimate)
library(ggmap)
library(ggimage)
library(ggplot2)
library(forcats)
library(dplyr)

register_google(key = "AIzaSyDpEMlwKKTTUF3rxw33H59VYXJJ9vaH0oM")

cambridge_map <- get_map(location = "Cambridge, MA", zoom = 14, maptype = "roadmap")

car_image_url <- "https://cdn.pixabay.com/photo/2015/10/01/17/17/car-967387_1280.png"

crash_data <- crash_data %>%
  mutate(hour = format(strptime(crash_time_2, format="%H:%M:%S"), "%H"))

crash_data$hour <- fct_relevel(crash_data$hour, as.character(0:23))

animated_plot <- ggmap(cambridge_map) +
  geom_image(data = crash_data, aes(x = lon, y = lat, image = car_image_url),
             size = 0.05) +
  labs(title = "Crash Locations on Cambridge Map (with Car Icons)",
       x = "Longitude",
       y = "Latitude") +
  theme_minimal() +
```

```
  transition_states(hour, transition_length = 2, state_length = 1) +
  labs(subtitle = "Hour of the day: {closest_state}") +
  enter_fade() +
  exit_fade()

animate(animated_plot, nframes = 100, fps = 10)
```

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

Graph (d): Faceted by Hour of the Day

Effectiveness Static comparison: Faceting allows the viewer to see all hours side by side, making it easier to compare patterns.

Clarity: Each hour is clearly separated into its own facet, so there's no confusion or overlap.

Efficiency: This static layout offers a comprehensive view without requiring interaction.

Limitations: Visual Overload: If there are too many facets and the graph might feel cluttered, requiring more effort to scan.

Storytelling: The static nature limits the ability to highlight how crash patterns evolve over time.

Graph (e): Animated Transition Over Hours

Effectiveness: Dynamic Storytelling: The animation clearly shows how crash data evolves over time, making it ideal for capturing trends/changes throughout the day.

Engagement: Animated graphs are often more engaging, keeping the viewer's attention longer.

Highlighting Trends: Since the animation can show the evolution of crash locations dynamically, it can help reveal certain time-based patterns.

Limitations: Comparison Difficulty: Although animation is great for showing trends over time, it's harder for the viewer to make side-by-side comparisons of different hours.

Time Investment: The viewer has to wait for the animation to run, which can be less efficient.

Which Is Better at Telling an Interesting Story? In terms of storytelling, graph (e) (the animated one) is more effective.

Focus: The animation brings attention to the time-based nature of the data, which adds to the narrative aspect.

Engagement: The animation is more likely to hold a viewer's attention, making the data "come to life."

Clarity of Change: By showing one hour at a time, it becomes easier to follow changes in crash locations and frequency.

However, for tasks that need us to look at different hours, graph (d) would be more practical for side-by-side views.

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.)

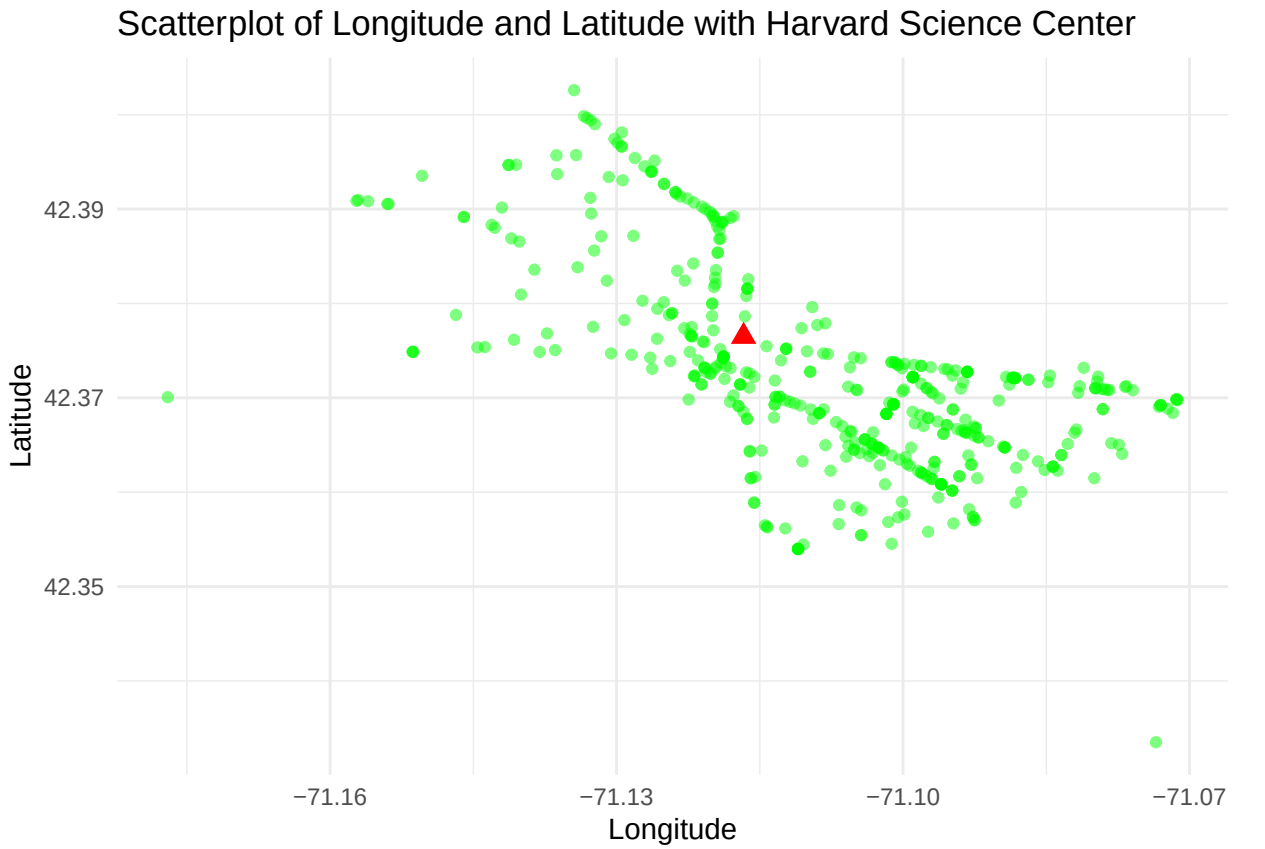
a. Grab the code and recreate the static graph here.

```
#graph from pset 1

library(ggplot2)
```

```r
harvard_science_center <- data.frame(lat = 42.3765, lon = -71.1167)

ggplot(crash_data, aes(x = lon, y = lat)) +
  geom_point(alpha = 0.5, size = 1.5, color = "green") +
  geom_point(data = harvard_science_center, aes(x = lon, y = lat),
             color = "red", size = 3, shape = 17) +
  labs(title = "Scatterplot of Longitude and Latitude with Harvard Science Center",
       x = "Longitude",
       y = "Latitude") +
  theme_minimal()
```



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!

```r
library(gganimate)
library(ggplot2)

# Define Harvard Science Center coordinates
harvard_science_center <- data.frame(lat = 42.3765, lon = -71.1167)

# Assuming crash_data has a 'time' variable, or we will simulate a time variable
```

```r
crash_data$time <- seq(1, nrow(crash_data))  # Simulating time sequence for illustration

# Create animated scatterplot
animated_plot <- ggplot(crash_data, aes(x = lon, y = lat)) +
  geom_point(alpha = 0.5, size = 1.5, color = "green") +
  geom_point(data = harvard_science_center, aes(x = lon, y = lat),
             color = "red", size = 3, shape = 17) +
  labs(title = "Scatterplot of Longitude and Latitude with Harvard Science Center",
       subtitle = "Frame: {frame_time}",  # Show time in subtitle
       x = "Longitude",
       y = "Latitude") +
  theme_minimal() +
  transition_time(time) +  # Animate over 'time' variable
  ease_aes('linear') +  # Smooth animation
  enter_fade() +  # Fade-in effect
  exit_fade()     # Fade-out effect

# Animate the plot with controls over speed
animate(animated_plot, nframes = 100, fps = 10)
```

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

How the Animation Improved the Plot: Time-Based Patterns: The animation reveals how crash locations evolve over time, making trends easier to spot. Engagement: Movement captures attention and adds narrative. Reduces Clutter: By spacing out data points, the animation prevents overcrowding. Consistent Emphasis: The Harvard Science Center remains highlighted, helping the viewer maintain focus on this key location.

How the Animation Worsened the Plot: Loss of Overview: Unlike the static plot, the animation prevents seeing all data at once, making it harder to compare time periods. Time-Consuming: It takes longer to interpret since the viewer must watch the entire sequence. Missing Details: Fast transitions may overlook important data. Focus: The emphasis on time can detract from spatial insights if location is more important.

Conclusion: The animation enhances storytelling and engagement but sacrifices overview and direct comparisons, making it more suitable for exploring time-based trends.

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

```r
library(ggplot2)
library(plotly)

# Define Harvard Science Center coordinates
harvard_science_center <- data.frame(lat = 42.3765, lon = -71.1167)

# Create the static scatterplot using ggplot2
static_plot <- ggplot(crash_data, aes(x = lon, y = lat)) +
  geom_point(alpha = 0.5, size = 1.5, color = "green") +
  geom_point(data = harvard_science_center, aes(x = lon, y = lat),
             color = "red", size = 3, shape = 17) +
  labs(title = "Scatterplot of Longitude and Latitude with Harvard Science Center",
       x = "Longitude",
       y = "Latitude") +
  theme_minimal()

# Convert the ggplot2 plot to a plotly interactive plot
interactive_plot <- ggplotly(static_plot)
```

```
# Display the interactive plot
interactive_plot
```

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

How Interactivity Improved the Plot: Exploration: The ability to zoom, pan, and hover over points allows for exploration of specific areas. Immediate Feedback: Hover tooltips provide instant access to data details without cluttering. Engagement: Interactive elements make the plot more engaging.

How Interactivity Worsened the Plot: Complexity: The added interactivity can overwhelm users who just want a simple overview. Immediate Insight: Key insights might require more manual exploration, but a static plot presents everything at once. Performance: Interactive plots can be slower to load or navigate, especially with large datasets.

Conclusion: Interactivity enhances user engagement and detailed exploration but can sacrifice simplicity.

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"`.

```
library(dplyr)
library(forcats)

crash_data <- read_csv("https://raw.githubusercontent.com/harvard-stat108s23/materials/main/psets/data/

crash_data <- crash_data %>%
  mutate(ambnt_light_descr = fct_collapse(ambnt_light_descr,
                                          Dark = c("Dark - Lighted", "Dark - Not Lighted", "Dark - Unkn
                                          Dawn = "Dawn",
                                          Daylight = "Daylight",
                                          Dusk = "Dusk",
                                          other_level = "NULL"))

head(crash_data)
```

```
## # A tibble: 6 x 24
##   crash_numb city_town_name crash_date crash_severity_descr crash_status
##        <dbl> <chr>          <chr>      <chr>                <chr>
## 1    4175194 CAMBRIDGE      01/11/2016 Not Reported         Closed
## 2    4181538 CAMBRIDGE      02/17/2016 Non-fatal injury     Closed
## 3    4181543 CAMBRIDGE      02/17/2016 Non-fatal injury     Closed
## 4    4181575 CAMBRIDGE      02/18/2016 Non-fatal injury     Closed
## 5    4181601 CAMBRIDGE      02/18/2016 Fatal injury         Closed
## 6    4181834 CAMBRIDGE      02/19/2016 Non-fatal injury     Closed
## # i 19 more variables: crash_time_2 <time>, year <dbl>,
## #   max_injr_svrty_cl <chr>, numb_vehc <dbl>, polc_agncy_type_descr <chr>,
## #   sptroop <chr>, age_drvr_oldest <chr>, crash_hour <chr>,
## #   first_hrmf_event_descr <chr>, ambnt_light_descr <fct>,
## #   manr_coll_descr <chr>, road_surf_cond_descr <chr>, numb_fatal_injr <dbl>,
## #   weath_cond_descr <chr>, hit_run_descr <chr>, most_hrmfl_evt_cl <chr>,
```

```
## #   speed_limit <dbl>, lat <dbl>, lon <dbl>
```

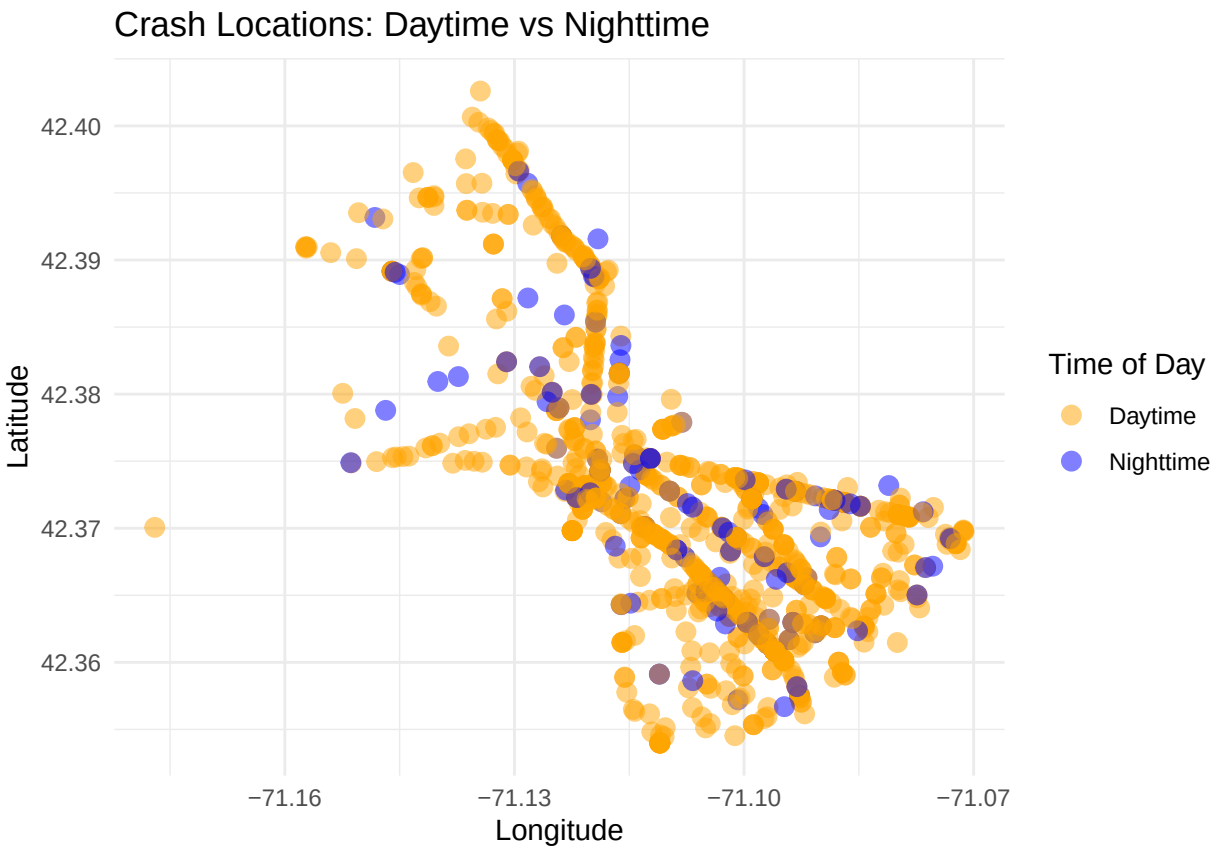
b. Do crashes tend to happen in different places during the day as opposed to at night? Using only data from crashes that occurred either with dark or daylight levels of ambient light, recreate your scatterplot from Problem 2(a) from P-Set 1. Be sure that the distinction between daytime versus nighttime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

```r
# Load necessary libraries
library(ggplot2)
library(dplyr)

df <- read.csv('https://raw.githubusercontent.com/harvard-stat108s23/materials/main/psets/data/cambridg

crash_data_filtered <- df %>%
  filter(ambnt_light_descr %in% c("Daylight", "Dark - lighted", "Dark - unlighted", "Dusk", "Dawn")) %>%
  mutate(day_night = ifelse(ambnt_light_descr == "Daylight", "Daytime", "Nighttime"))

# Create the scatterplot
ggplot(crash_data_filtered, aes(x = lon, y = lat, color = day_night)) +
  geom_point(size = 3, alpha = 0.5) +
  scale_color_manual(values = c("Daytime" = "orange", "Nighttime" = "blue")) +
  labs(title = "Crash Locations: Daytime vs Nighttime",
       x = "Longitude",
       y = "Latitude",
       color = "Time of Day") +
  theme_minimal() +
  theme(legend.position = "right")
```



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.

It appears that the distribution of crashes during the day and night does not show a strong geographical separation. Both daytime (orange) and nighttime (blue) crashes are concentrated in similar areas, especially around central locations. Nevertheless, there are a few areas with variation:

Overlap in dense areas: There are certain areas experiencing crashes both during the day and at night, highlighting potentially hazardous locations.

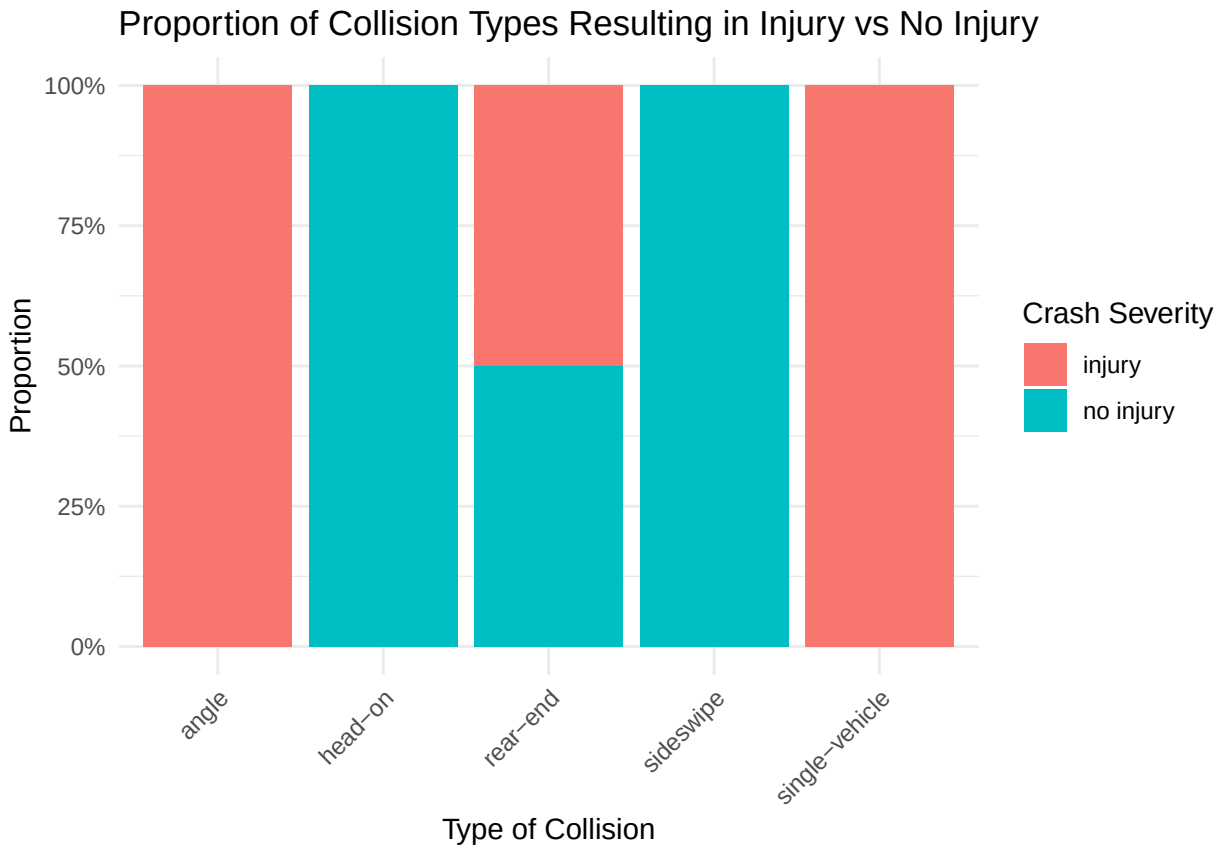
Slight differences: While the overall pattern shows crashes in roughly the same locations, there are some spots with a higher density of daytime crashes compared to nighttime crashes, and vice versa based on the streaks of crashes and intensity of crashes seen above.

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.

```r
# Load necessary libraries
library(ggplot2)
library(dplyr)

# Example dataset for the manner of collision and crash severity
collision_data <- data.frame(
  manr_coll_descr = c('angle', 'head-on', 'rear-end', 'sideswipe', 'single-vehicle',
                      'rear-end', 'angle', 'sideswipe'),
  crash_severity_descr = c('injury', 'no injury', 'injury', 'no injury', 'injury',
                           'no injury', 'injury', 'no injury')
)

# Create a bar plot to show proportion of injury vs no injury by collision type
ggplot(collision_data, aes(x = manr_coll_descr, fill = crash_severity_descr)) +
  geom_bar(position = 'fill') +  # Create proportional bar plot
  labs(title = 'Proportion of Collision Types Resulting in Injury vs No Injury',
       x = 'Type of Collision',
       y = 'Proportion',
       fill = 'Crash Severity') +
  scale_y_continuous(labels = scales::percent) +  # Show y-axis as percentages
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



The bar chart compares the proportions of collision types that result in injury versus no injury for five different categories: angle, head-on, rear-end, sideswipe, and single-vehicle collisions.

Key Insights from the Visualization: Angle and Single-Vehicle Collisions: Both angle and single-vehicle collisions result in injuries 100% of the time based on this dataset. This indicates that these types of collisions are highly dangerous.

Head-On Collisions: All head-on collisions resulted in no injuries in this dataset. This might be surprising, but in this case, the dataset indicates no injury outcomes.

Rear-End Collisions: Rear-end collisions show a mix, with approximately 50% resulting in injury and the other 50% not leading to any injuries. This suggests that rear-end collisions can vary in severity.

Sideswipe Collisions: No sideswipe collisions in this dataset resulted in injuries, meaning all of these types of collisions only caused property damage.

Story the Visualization Communicates: This visualization highlights that collision types greatly influence injury outcomes. Overall, it suggests that angle and single-vehicle crashes might need special attention in terms of safety measures as they are most likely to cause harm.

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?

```
library(dplyr)

df <- read.csv('https://raw.githubusercontent.com/harvard-stat108s23/materials/main/psets/data/cambridg
```

```r
dark_conditions <- c("Dark - lighted", "Dark - unlighted", "Dusk", "Dawn")

df_dark_2022 <- df %>%
  filter(ambnt_light_descr %in% dark_conditions, year == 2022) %>%
  select(crash_time_2, manr_coll_descr, max_injr_svrty_cl, speed_limit) %>%
  filter(!is.na(speed_limit)) %>%
  arrange(desc(speed_limit))

print(df_dark_2022)
```

```
##    crash_time_2           manr_coll_descr            max_injr_svrty_cl
## 1      00:03:00      Single vehicle crash Suspected Serious Injury (A)
## 2      18:00:00                   Head-on   Suspected Minor Injury (B)
## 3      22:18:00                     Angle   Suspected Minor Injury (B)
## 4      06:15:00                   Unknown Suspected Serious Injury (A)
## 5      20:52:00                     Angle          Possible Injury (C)
## 6      22:13:00                     Angle Suspected Serious Injury (A)
## 7      20:41:00                     Angle       No Apparent Injury (O)
## 8      19:10:00                     Angle Suspected Serious Injury (A)
## 9      14:46:00                     Angle   Suspected Minor Injury (B)
## 10     17:17:00      Single vehicle crash   Suspected Minor Injury (B)
## 11     18:03:00      Single vehicle crash   Suspected Minor Injury (B)
## 12     18:31:00                 Rear-end       No Apparent Injury (O)
## 13     22:04:00                   Head-on   Suspected Minor Injury (B)
## 14     18:29:00      Single vehicle crash   Suspected Minor Injury (B)
## 15     19:53:00 Sideswipe, same direction   Suspected Minor Injury (B)
## 16     18:36:00      Single vehicle crash       No Apparent Injury (O)
## 17     20:44:00                   Unknown   Suspected Minor Injury (B)
##    speed_limit
## 1           35
## 2           25
## 3           25
## 4           25
## 5           25
## 6           25
## 7           25
## 8           25
## 9           25
## 10          25
## 11          25
## 12          20
## 13          20
## 14          20
## 15          20
## 16          20
## 17          20
```

```r
head(df_dark_2022)
```

```
##   crash_time_2      manr_coll_descr            max_injr_svrty_cl speed_limit
## 1     00:03:00 Single vehicle crash Suspected Serious Injury (A)          35
## 2     18:00:00              Head-on   Suspected Minor Injury (B)          25
## 3     22:18:00                Angle   Suspected Minor Injury (B)          25
## 4     06:15:00              Unknown Suspected Serious Injury (A)          25
```

```
## 5     20:52:00                Angle          Possible Injury (C)          25
## 6     22:13:00                Angle Suspected Serious Injury (A)          25
```

The crash at the highest speed limit occurred at 00:03:00 (12:03 AM).

Yes, there were injuries. The injury status for this crash is Suspected Serious Injury (A).

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?

```
library(dplyr)

df <- read.csv('https://raw.githubusercontent.com/harvard-stat108s23/materials/main/psets/data/cambridg

dark_conditions <- c("Dark - lighted", "Dark - unlighted", "Dusk", "Dawn")

df_dark_2022 <- df %>%
  filter(ambnt_light_descr %in% dark_conditions, year == 2022) %>%
  select(max_injr_svrty_cl) %>%
  group_by(max_injr_svrty_cl) %>%
  summarise(num_injuries = n()) %>%
  arrange(desc(num_injuries))

print(df_dark_2022)
```

```
## # A tibble: 4 x 2
##   max_injr_svrty_cl            num_injuries
##   <chr>                               <int>
## 1 Suspected Minor Injury (B)              9
## 2 Suspected Serious Injury (A)            4
## 3 No Apparent Injury (O)                  3
## 4 Possible Injury (C)                     1
```

```
most_common_injury <- df_dark_2022 %>%
  slice(1) %>%
  pull(max_injr_svrty_cl)

cat("The most common type of injury reported was:", most_common_injury)
```

```
## The most common type of injury reported was: Suspected Minor Injury (B)
```

Problem 5

We will cover git/GitHub later this week, so you should wait until then to do this problem. If needed, we can extend the pset date accordingly.

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.

COMPLETED