Lab 2

Math 241, Week 2

```
# Put all necessary libraries here
# I got you started!
# The first time you want to install the dsbox package; then you can comment it out.
# If you have not installed the devtools package, you will need to do so first
#install.packages("devtools")
#library(devtools)
#install.packages("tidyverse")
#install.packages("viridis")
#install.packages("lintr")
#install.packages("datapasta")
#install.packages("tidytuesday"
elements_by_episode <- read.csv("C:/Users/agjjo/Downloads/Documents/GitHub/math241/data/elements-by-epis



#devtools::install_github("tidyverse/dsbox")
library(dsbox)
#install.packages("styler")
library(styler)
library(tidyverse)
library(viridis)
#data(accidents)
library(magrittr)
#data(mpg)
library(lintr)
data("elements-by-episode.df")
library(tidytuesdayR)
data("rent")
#view(elements_by_episode.df)
#view(mpg)
#?mpg
#?accidents
#view(accidents)
```

Due: Thursday, February 8th at 8:30am

Goals of this lab

1. Practice coding to adhere to the Tidyverse Style Guide.
2. Practice creating and refining graphs with `ggplot2`.
3. Consider the strengths and weaknesses of various `geom`s and `aes`thetics for telling a data story.

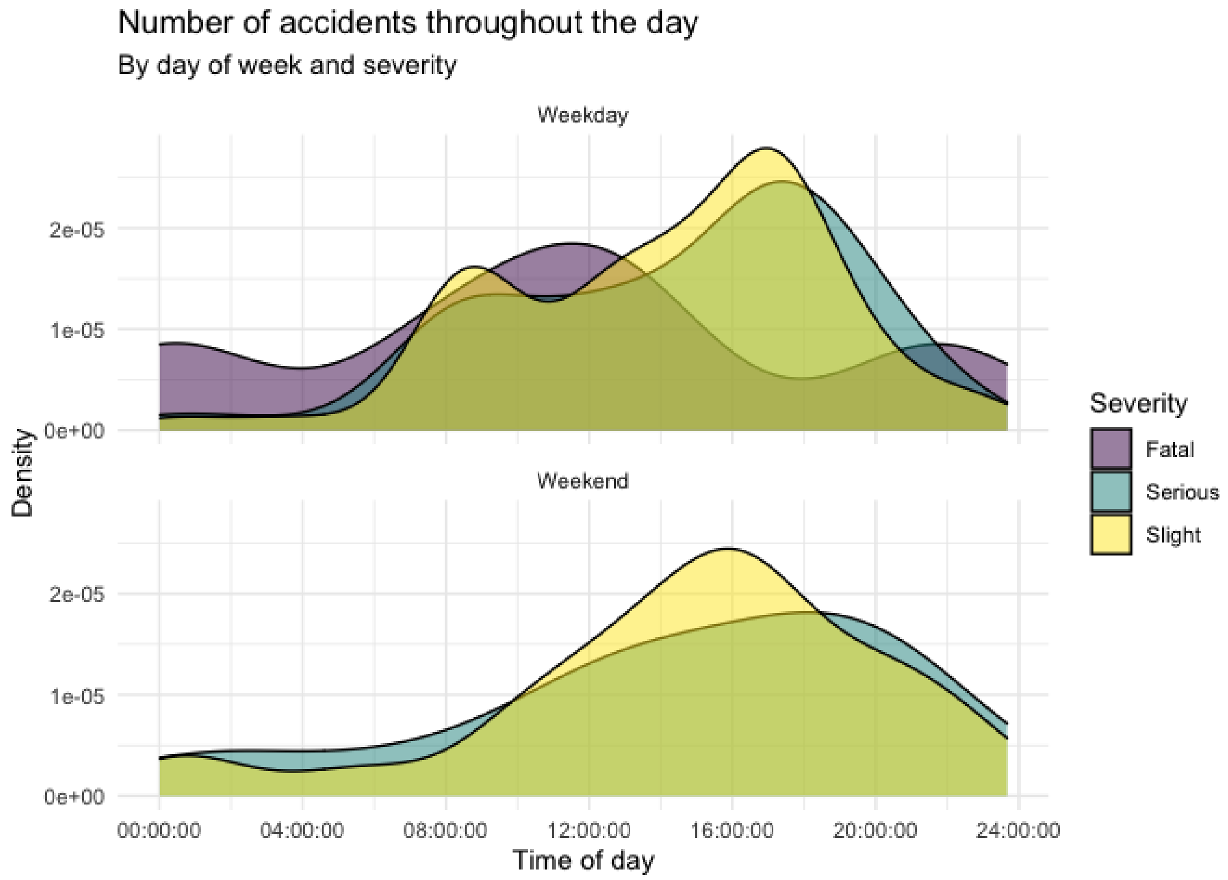
Notes:

- When creating your graphs, consider context (i.e. axis labels, title, . . .)!
- If I provide partially completed code, I will put `eval = FALSE` in the chunk. Make sure to change that to `eval = TRUE` once you have completed the code in the chunk.
- Be prepared to ask for help from me, Simon, and your classmates! We scratched the surface of `ggplot2` in class. But I encourage you to really dig in and make your graphs your own (i.e. don't rely on defaults).

Problems

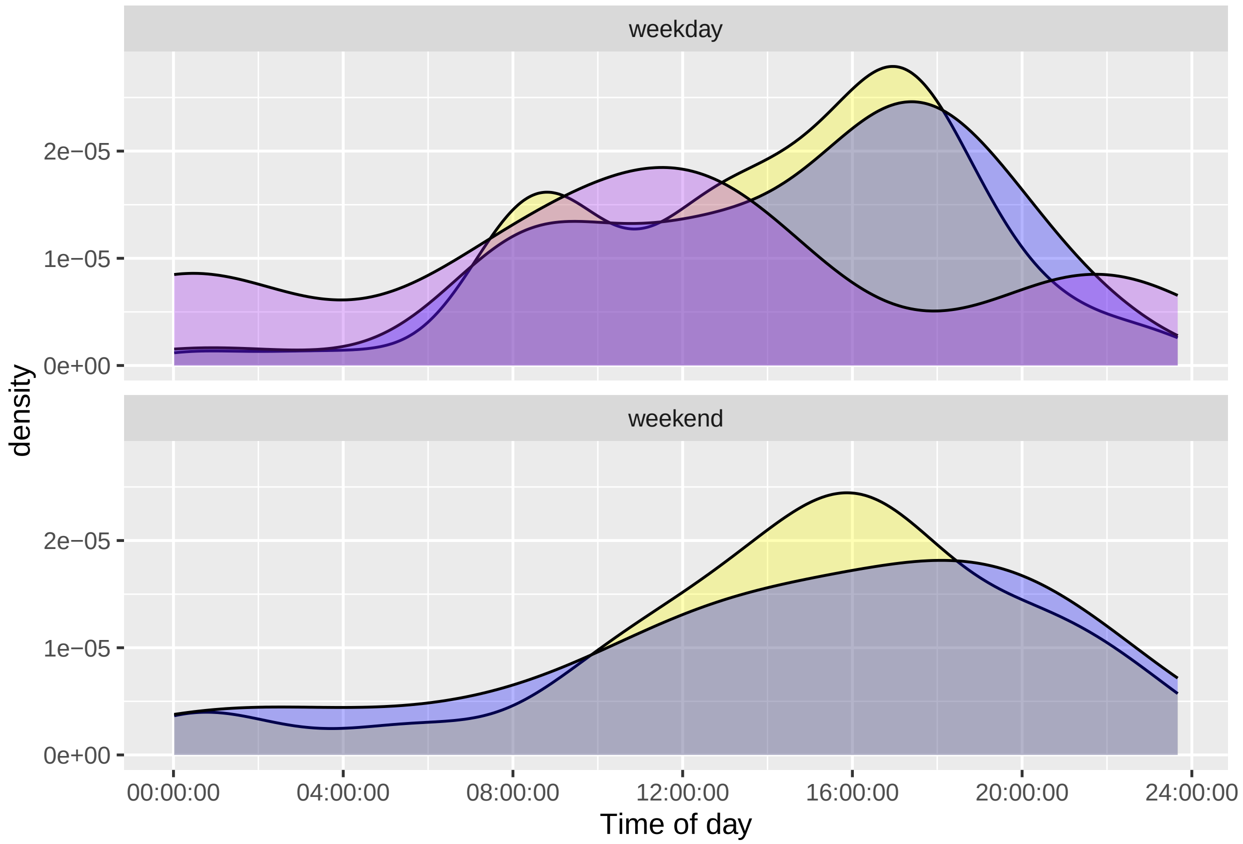
Probem 1: Road traffic injuries in Edinburgh, Scotland

The dataset can be found in the `dsbox` package, and is called `accidents`. It covers all recorded accidents in Edinburgh in 2018; compared to the dataset made available by the UK government, some of the variables were modified for the purposes of the package. You can find out more about the dataset by inspecting its documentation with `?accidents`. Recreate the following plot, and interpret the results.



```
#mutating weekend variable
accidents <- mutate(accidents, weekend = ifelse(day_of_week %in% c("Monday", "Tuesday", "Wednesday", "Th

#wrangling
slight_accidents <- accidents %>%
  filter(severity == "Slight")
```

```r
serious_accidents <- accidents %>%
  filter(severity == "Serious")

fatal_accidents <- accidents %>%
  filter(severity == "Fatal")



#plotting

accident_plot <- ggplot(data = slight_accidents, aes(x = time)
                       ) +
  geom_density(fill = "yellow", alpha = 0.3
               ) +
  geom_density(data = serious_accidents, fill = "blue", alpha = 0.3
               ) +
  geom_density(data = fatal_accidents, fill = "purple", alpha = 0.3
               ) +
  xlab("Time of day"
       ) +
  facet_wrap(~weekend, nrow = 2
             )



accident_plot
```



```
#could not figure out how to add a legend
```

##Interpretation

The first thing that strikes me is that there are no fatal accidents observed on weekends, this could be due to the fact that someone is less likely to be in a rush while driving or to be distracted because many people do not work on the weekend (by the same toke n there are likely to be fewer cars on the road). The second thing I notice is that accidents seem to peak around 4pm, this makes sense as that is traditionally rush hours where the most people in the biggest rush are on the road. This trend also holds true on the weekend which makes sense because many people still do work weekends.

##my first approach

#weekday wrangling accidents_weekday <- accidents %>% filter(day_of_week %in% c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday"))

slight_accidents_weekday <- accidents_weekday %>% filter(severity == "Slight")

serious_accidents_weekday <- accidents_weekday %>% filter(severity == "Serious")

fatal_accidents_weekday <- accidents_weekday %>% filter(severity == "Fatal")

#weekend wrangling accidents_weekend <- accidents %>% filter(day_of_week %in% c("Saturday", "Sunday"))

slight_accidents_weekend <- accidents_weekend %>% filter(severity == "Slight")

serious_accidents_weekend <- accidents_weekend %>% filter(severity == "Serious")

fatal_accidents_weekend <- accidents_weekend %>% filter(severity == "Fatal")

#weekday plots weekday_plot <- ggplot(data = slight_accidents_weekday, aes(x = time)) + geom_density(fill = “yellow”, alpha = 0.3) + geom_density(data = serious_accidents_weekday, fill = “blue”, alpha = 0.3)+ geom_density(data = fatal_accidents_weekday, fill = “purple”, alpha = 0.3)

weekday_plot

#weekend plots weekday_plot <- ggplot(data = slight_accidents_weekend, aes(x = time)) + geom_density(fill = “yellow”, alpha = 0.3) + geom_density(data = serious_accidents_weekend, fill = “blue”, alpha = 0.3)+ geom_density(data = fatal_accidents_weekend, fill = “purple”, alpha = 0.3)

weekday_plot

Problem 2: One Dataset, Visualized ~~25~~ 5 Ways

##using this: https://abcnews.go.com/Health/1500-americans-dying-covid-week/story?id=106237143

Inspired by Nathan Yau’s One Dataset, Visualized 25 Ways, I want you to create 5 visualizations of the same data. You can use the `mpg` dataset or another dataset of your choosing, including the `accidents` dataset above. Make sure you have the data manual open for this problem!

a. Pick 3 - 4 variables you want to explore. Provide their code names here. police urban_rural speed_limit vehicle

b. Create 5 graphs. A few things to consider:

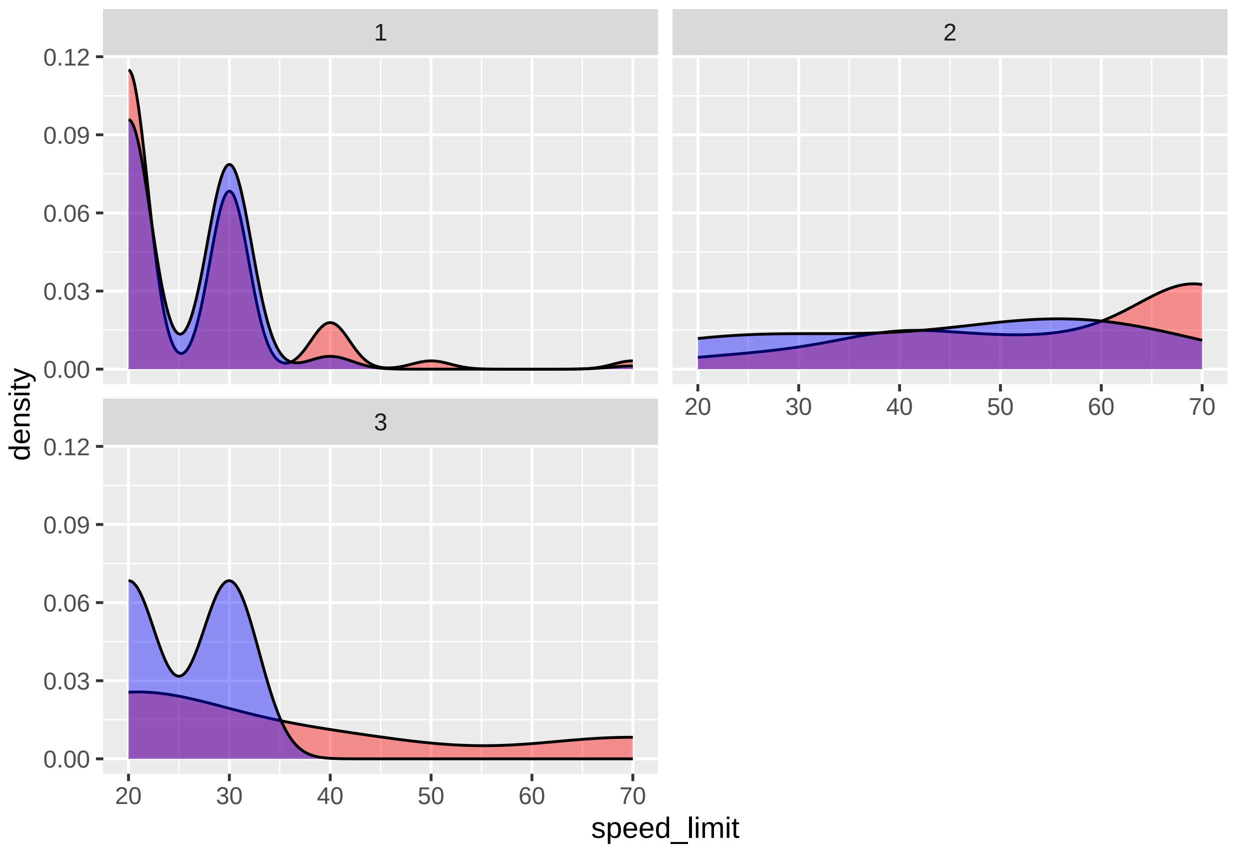
    - Like Nathan’s graphs, they don’t all have to contain every one of your selected variables.
    - You can’t use the same `geom` for all four graphs but you can use the same `geom` more than once.
    - Think carefully about color, the coordinate system, and scales.
    - Feel free to subset or wrangling the dataset if you want to but it isn’t required.

##graphs

graph a

```
#wrangling

  #figure out how to recode urban_rural to list "urban" or "rural" rather than 1 2 and apparently 3???
#accidents <- accidents %>%
  #mutate(accidents, urban_rural = ifelse(urban_rural = ifelse(urban_rural = 1) 'urban', 'rural'))

accidents_police <- accidents %>%
  filter(police == "Yes")

accidents_no_police <- accidents %>%
  filter(police == "No")


#plotting

a <- ggplot(data = accidents_police, aes(x = speed_limit)) +
      geom_density(fill = "red", alpha = 0.4
                   ) +
      geom_density(data = accidents_no_police, fill = "blue", alpha = 0.4
                   ) +
      facet_wrap(~urban_rural, nrow = 2)
```

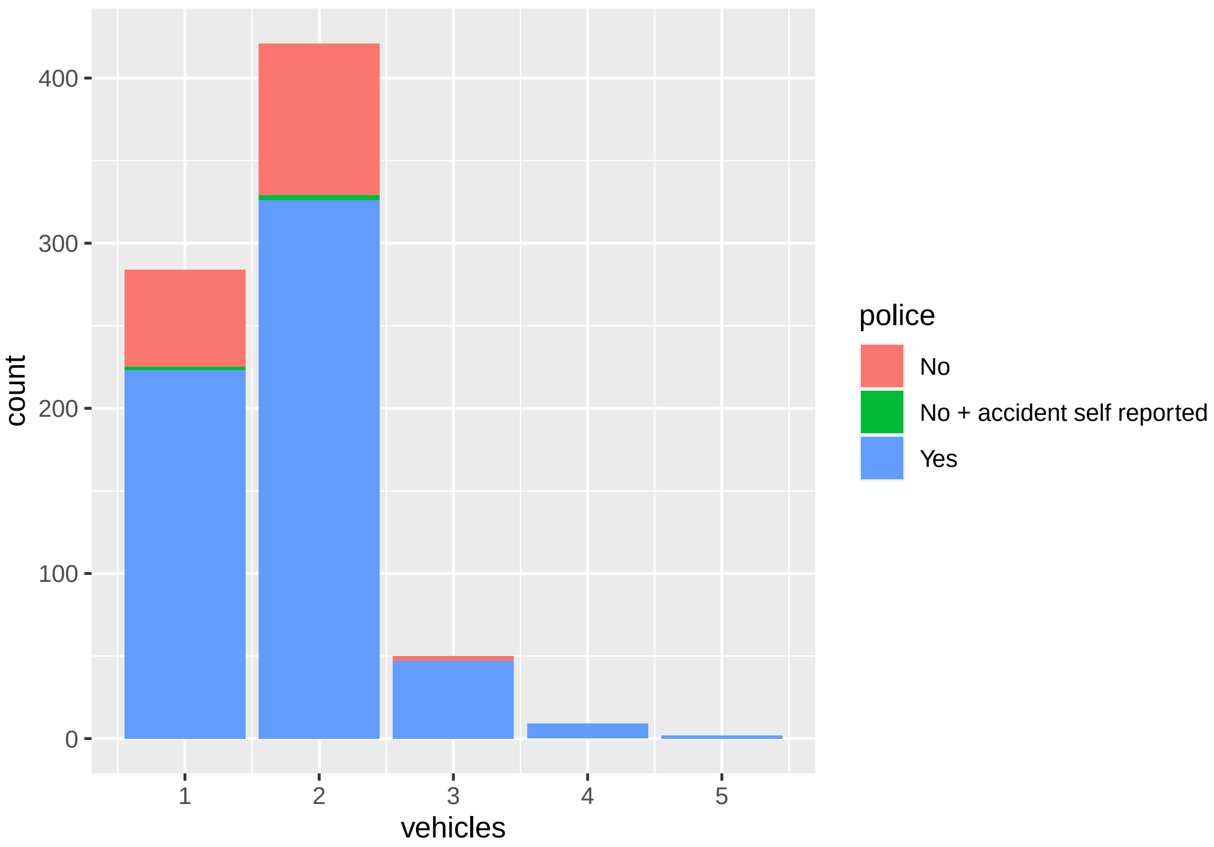
```
a
```



```
#Note: the density curve for accidents across speed limits is real interesting/funk and bears some inve


#Pros:
    #allows us to see if there is a trend relating speed limit to accident density (there is, a weird o
    #allows us to see if  there is a difference in police responsiveness relating to speed limit
    #compares these trends across urban and rural environments

#cons
    #hard to glean precise understanding of trends, you i=only really see the shape
    #doesn't control for conditions or vehicle count, which feel relavent
```

graph b

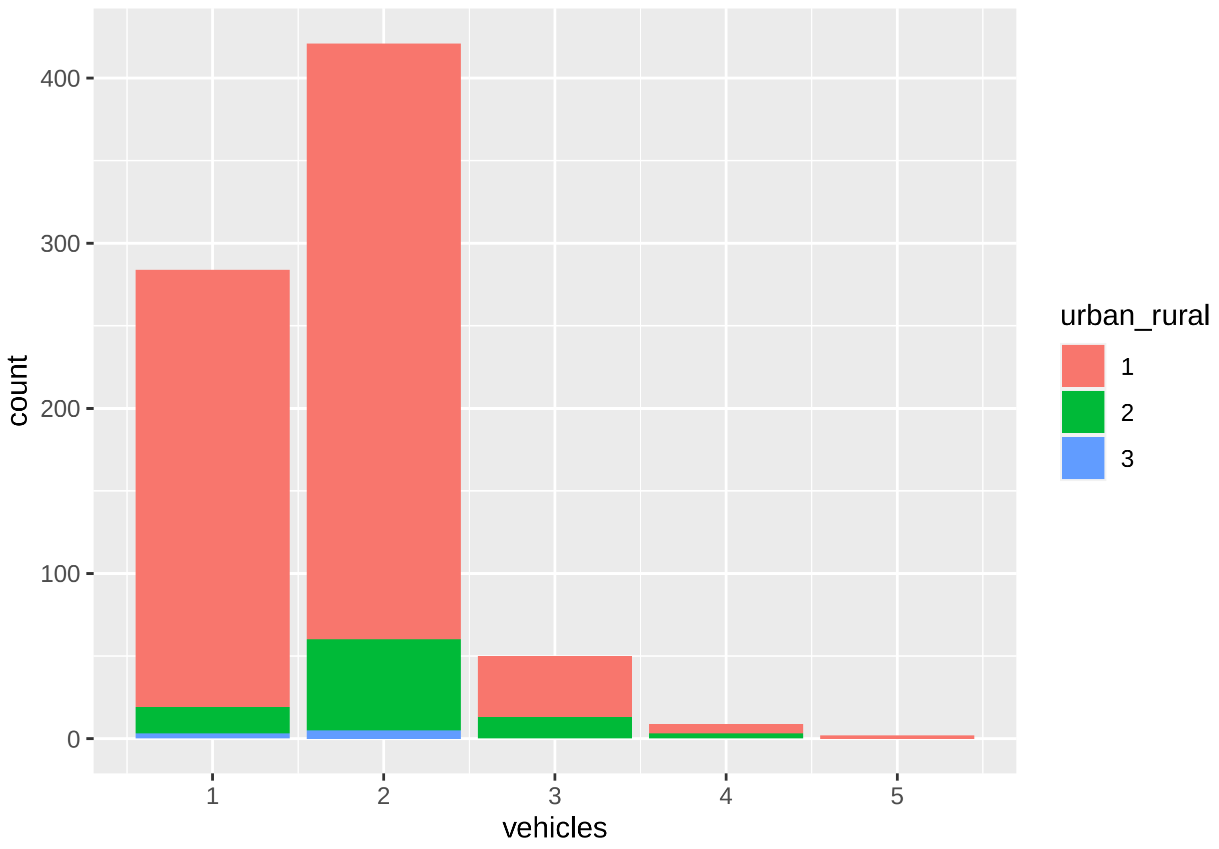
```
accidents <- accidents %>%
  filter(vehicles < 6)

ggplot(data = accidents, aes(x = vehicles, fill = police)) +
  geom_bar()
```



```
#how do I choose what color is associated with what??

#pros
    #both b & c give a sense of the relative number of crashes across vehicle count
    #shows trend of police response relative to vehicles involved

#cons
    #would need to make the bars equally sized to understand the true relative rates
```

graph c

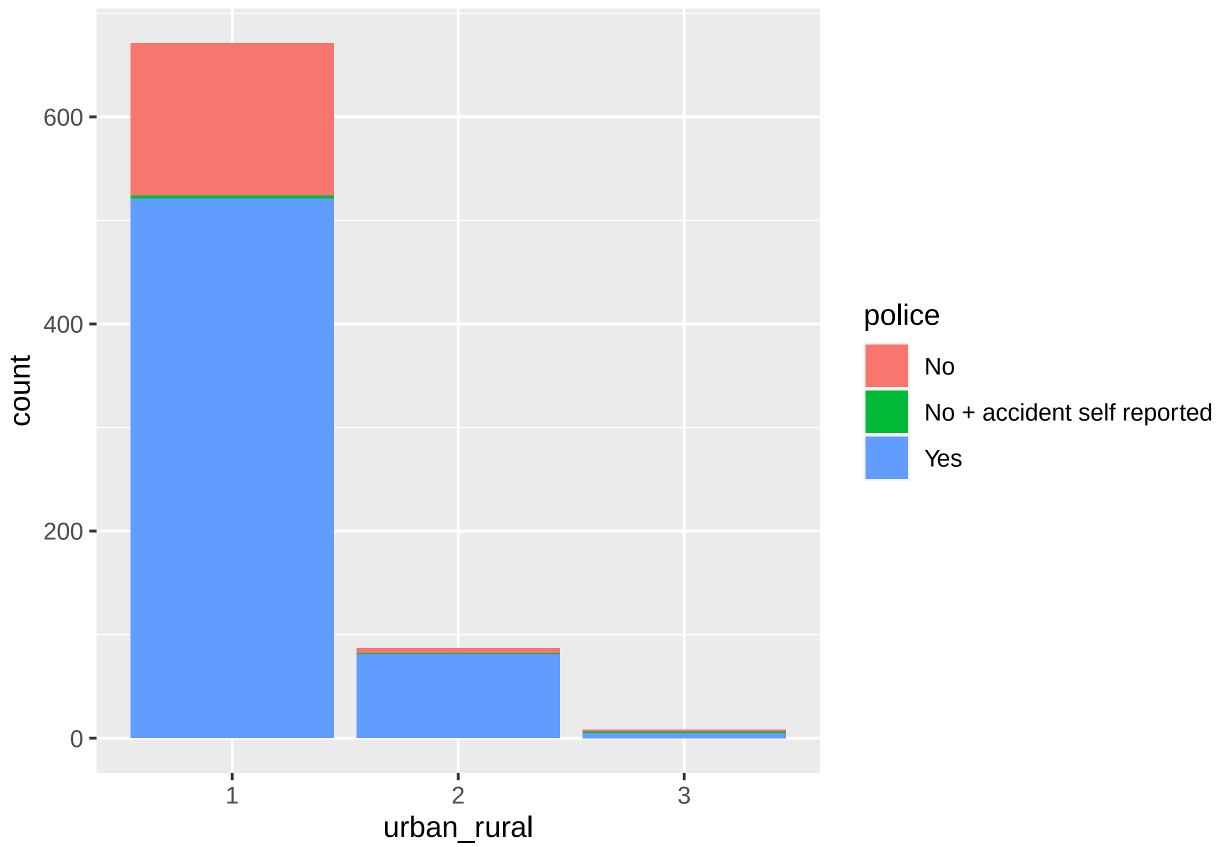
```
accidents <- accidents %>%
  filter(vehicles < 6)

# Convert urban_rural to factor
#ChatGPT explained this, I was already trying to find a way using mutate() to reclass urban_rural as ca
accidents$urban_rural <- factor(accidents$urban_rural)

ggplot(data = accidents, aes(x = vehicles, fill = urban_rural)) +
  geom_bar()
```



```
#still don't know what the third category in urban_rural is


#pros
    #both b & c give a sense of the relative number of crashes across vehicle count
    #shows trend of urban/rural environment relative to vehicles involved

#cons
    #would need to make the bars equally sized to understand the true relative rates
```

graph d

```
accidents$urban_rural <- factor(accidents$urban_rural)

ggplot(data = accidents, aes(x = urban_rural, fill = police)) +
  geom_bar()
```



```
#pros
    #both b & c give a sense of the relative number of crashes across vehicle count
    #shows trend of police response relative to urban/rural environment

#cons
    #would need to make the bars equally sized to understand the true relative rates
    #I'd like to do this with one of these but I'm not sure which
```

c. Discuss the pros/cons of your graphs. What useful information can be gleaned? How do the different geoms and aesthetics impact the story?

There seems to be a sin-wave form with reducing local maximums describing the relationship between accident density and speed limits, I don't know what that is called or what it means. Police seem to respond most often at 20, 40, & 70 mph speed limits. Police seem to respond almost always to accidents involving 3+ cars. Accidents involving more than one vehicle make up a larger proportion of accidents in rural settings than in urban ones. Police appear to respond more consistently to crashes in rural environments.

Problem 3: Style This Code!

Take the following code and don't change its functionality but DO change its style. Use the Tidyverse Style Guide!

```
# Creating the dataframe with the correct dimensions
animal_data <- data.frame(
  theanimalsweightisthisnumber = c(runif(3), NA),
  y = c("cat", "mouse", "dog", "rat")
)

# Mutating the dataframe to add 'animal_weight' column
animal_data <- animal_data %>%
  mutate(animal_weight = theanimalsweightisthisnumber) %>%
  select(-theanimalsweightisthisnumber)  # Remove the column
  view(animal_data)

# Calculating median, mean, and variance
median_val <- median(animal_data$animal_weight, na.rm = TRUE)
mean_val <- mean(animal_data$animal_weight, na.rm = TRUE)
variance_val <- var(animal_data$animal_weight, na.rm = TRUE)

# Printing the results
cat("Median:", median_val, "\n")
```

```
## Median: 0.1959391
```

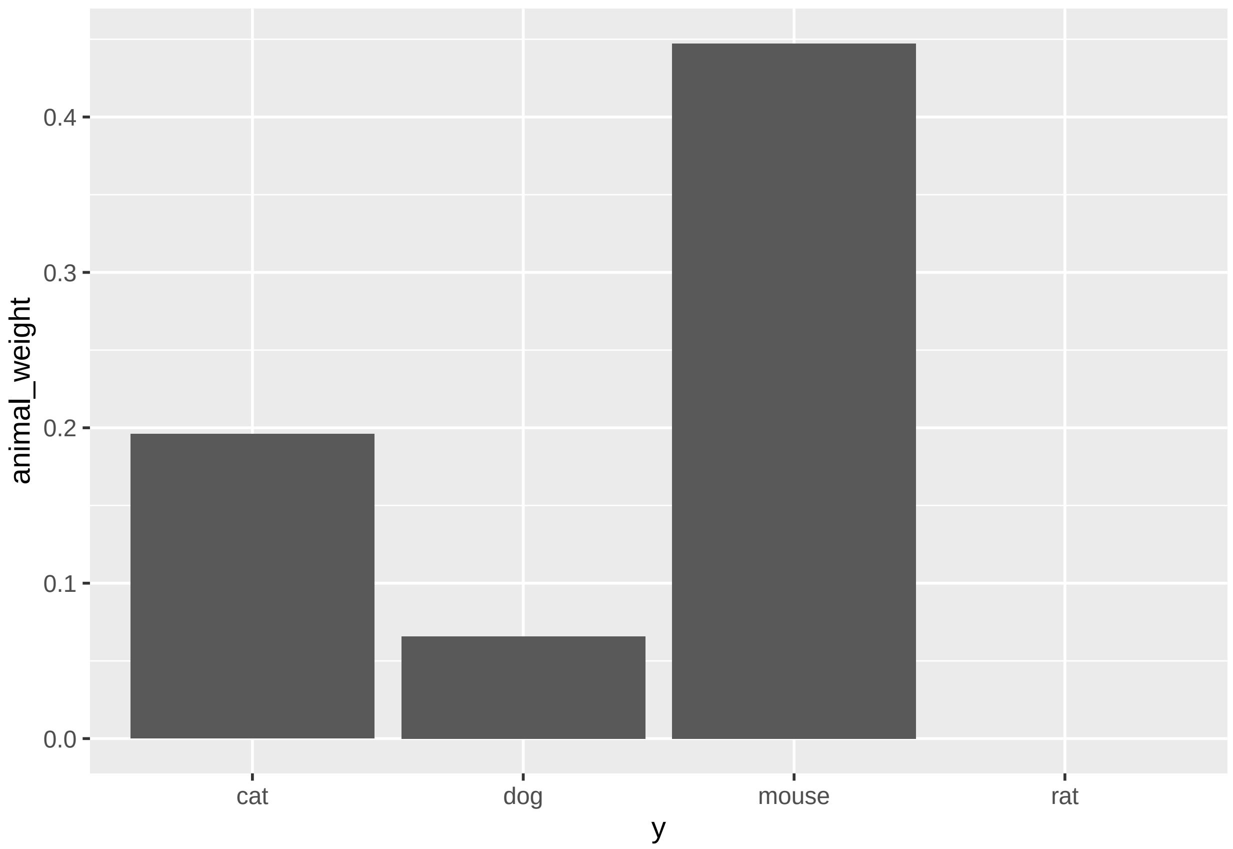
```
cat("Mean:", mean_val, "\n")
```

```
## Mean: 0.2363512
```

```
cat("Variance:", variance_val, "\n")
```

```
## Variance: 0.03763382
```

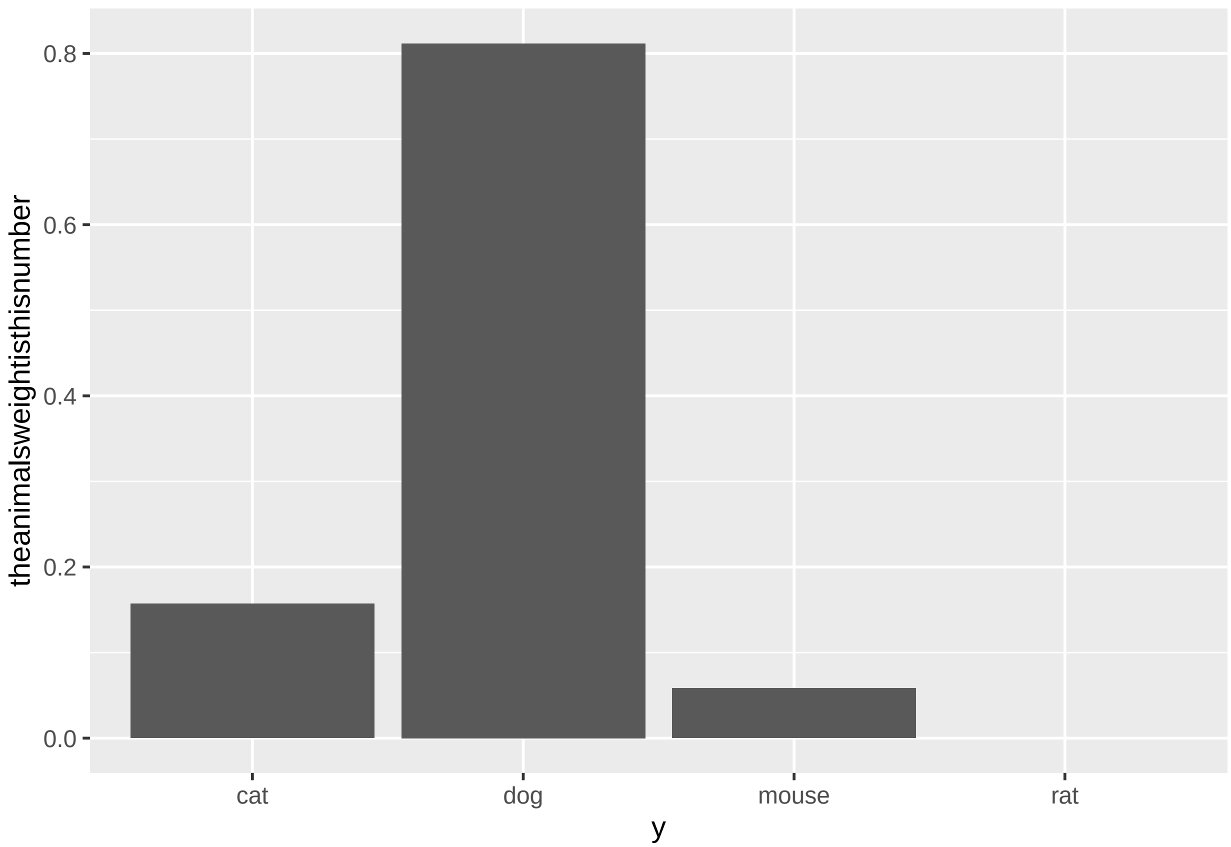
```
# Plotting the data
ggplot(animal_data, aes(y = animal_weight, x = y)) +
  geom_col() +
  scale_y_continuous()
```



```
thing.132232=data.frame(theanimalsweightisthisnumber=c(runif(3),NA),y=c("cat","mouse","dog","rat"))
median(thing.132232$theanimalsweightisthisnumber, TRUE);mean(thing.132232$theanimalsweightisthisnumber,
```

```
## [1] 0.1571413

## [1] 0.3424606

## [1] 0.1676859
```

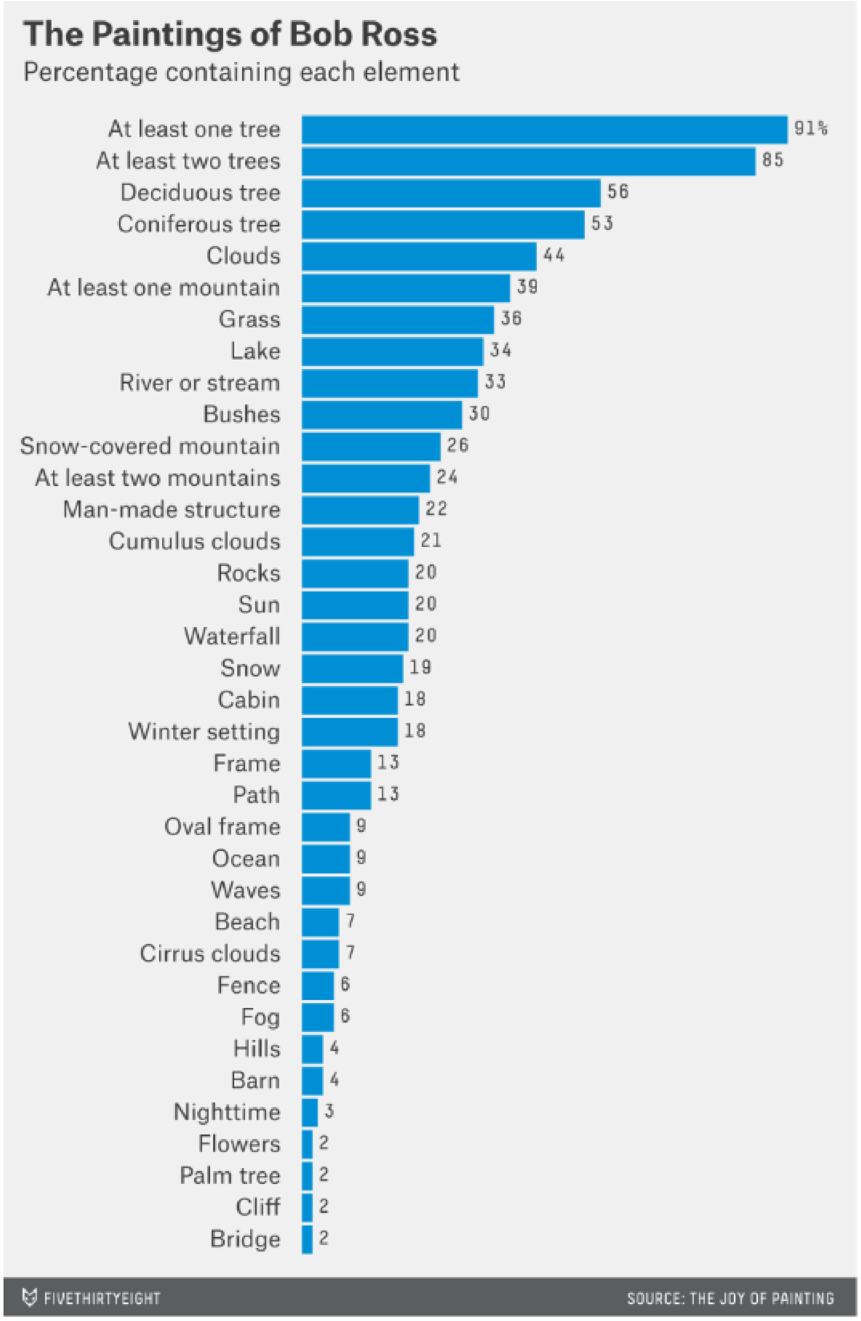
```
ggplot(thing.132232, aes(y=theanimalsweightisthisnumber,x=y))+geom_col()+scale_y_continuous()
```



Problem 4: Imitation is the Sincerest Form of Flattery

For this problem, I want you to try to recreate a FiveThirtyEight.com graphic. Awesomely, they share their data with the world here. (Note: You don't need to recreate all their branding/background color scheme.)

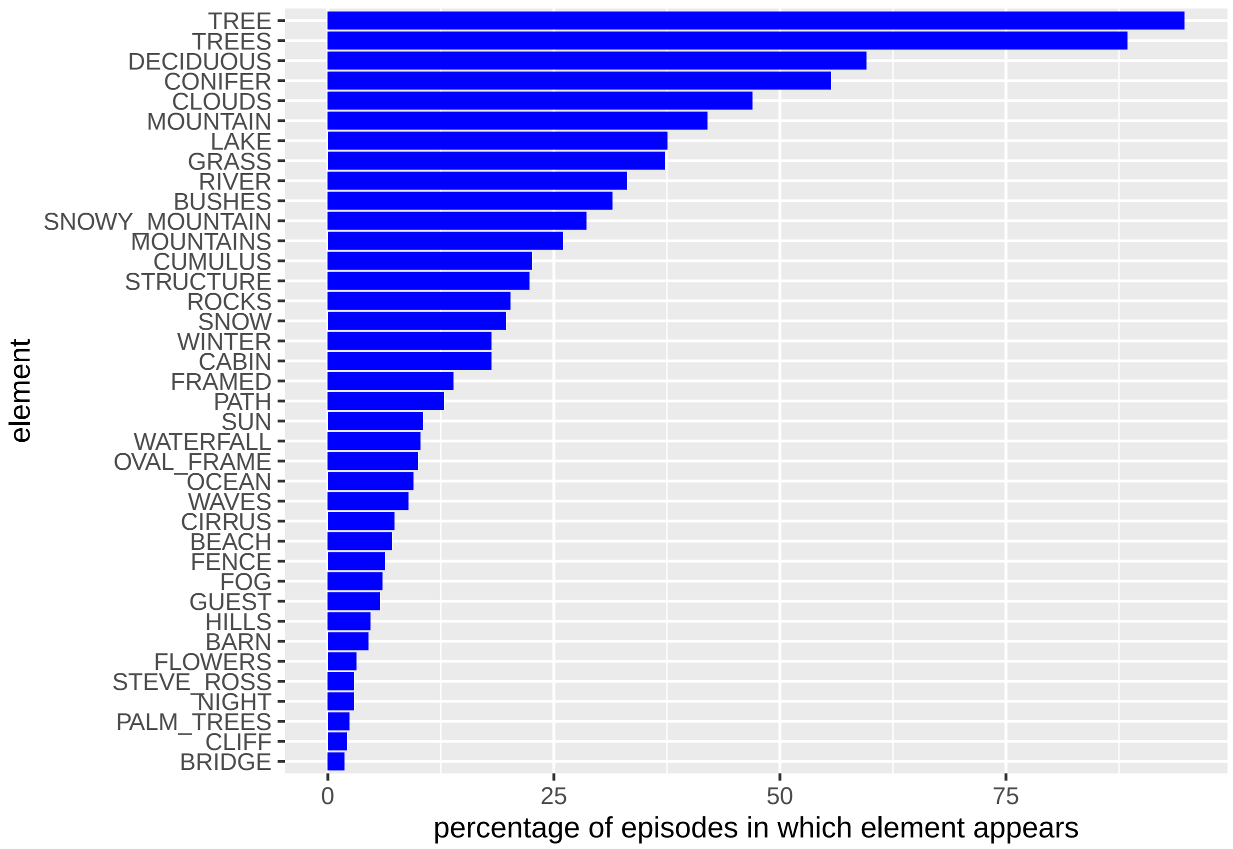
a. Take a screenshot of the graph, upload it to the same folder on the server where you have saved your lab, and insert the file name below. Then change the `eval = FALSE` to `eval = TRUE`.



b. Load the data and recreate the graph as best as you can.

```
# Turning the elements_by_episode df into a tibble
elem <- tibble::as_tibble(elements_by_episode)[,-2][,-1]

# Taking the sums of all columns in the tibble
sums <- colSums(elem)

# Creating a data frame from the sums
sums_bob_ross <- as.data.frame(sums, row.names = NULL, optional = FALSE)
names(sums_bob_ross) <- "sums"  # Naming the column
```

```r
#write.csv(sums_bob_ross, "bob.csv",row.names = TRUE)
props <- read.csv("bob.csv")

filtered_props <- props %>%
  filter(sums > 5)

filtered_props <- filtered_props %>%
  arrange(sums)

# Filtering out rows with a 'sums' value less than 5 (this was done by the author)
#props <- filter(sums > 5) %>%
#  mutate(prop = (sums / 381)*100)
#df_br <- as.data.frame(t(as.matrix(props)))

#plotting

#ggplot(data = filtered_props, aes(y = X, x= prop)) + geom_point()

#ggplot(data = sums_bob_ross, aes(y = factor(1), x = prop)) + geom_bar(stat = "identity")


ggplot(filtered_props, aes(x = factor(X, levels = X[order(prop)]), y = prop)) + #dont know what "factor
  geom_bar(stat = "identity", fill = "blue") +
  coord_flip() +
  xlab("element") +
  ylab("percentage of episodes in which element appears")
```



c. Now make the graph better somehow. i think I would change the title to better reflect that each bar represents the pecentage of episodes in the show that each painted element appears inr

d. Justify why your rendition of this `FiveThirtyEight.com` graph is more effective at telling the data story than the original.

Problem 5: Rental apartments in SF

The data for this exercise comes from `TidyTuesday`, and is on rental prices in San Francisco. You can find out more about the dataset by inspecting its documentation here. The dataset you'll be using is called `rent`. Create a visualization that will help you compare the distribution of rental prices (`price`) per bedroom (`beds`) across neighborhoods (`nhood`) in the city of San Francisco `(city == "san francisco")`, over time.

Limit your analysis to rentals where the full unit is available, i.e. (`room_in_apt == 0`). You have the flexibility to choose which years and which neighborhoods. Note that you should have a maximum of 8 neighborhoods on your visualization, but one or more of them can be a combination of many (e.g., an "other" category). Your visualization should also display some measure of the variability in your data. You get to decide what type of visualization to create and there is more than one correct answer! In your answer, include a brief description of why you made the choices you made as well as an interpretation of the findings of how rental prices vary over time and neighborhoods in San Francisco.

```
# Get the Data

# Read in with tidytuesdayR package
# Install from CRAN via: install.packages("tidytuesdayR")
# This loads the readme and all the datasets for the week of interest
```

```r
library(tidytuesdayR)
tuesdata <- tidytuesdayR::tt_load('2022-07-05') # this could take a minute

rent <- tuesdata$rent

data(tuesdata)
```