Final Project

Elizabeth Hora

12/6/2020

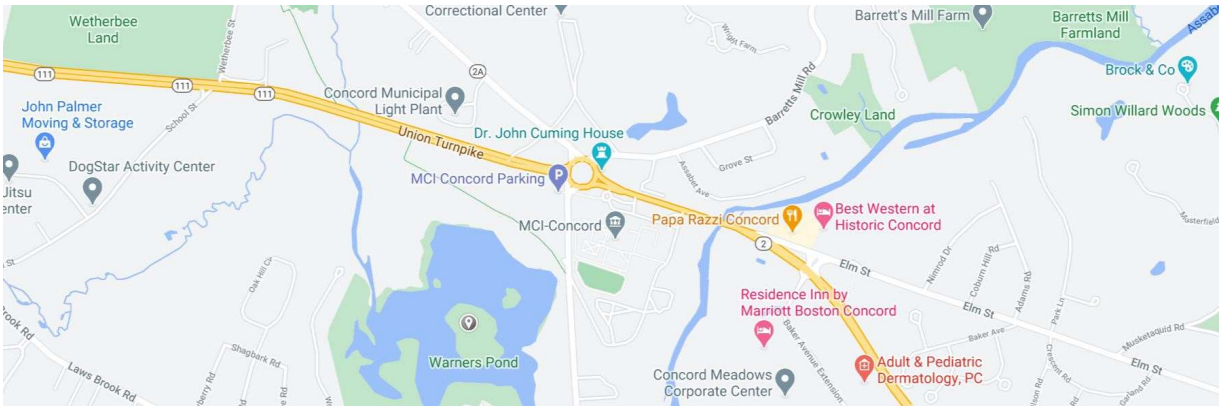
```
# Loading Packages
library(tidyverse)
library(ggplot2)
library(readxl)
library(janitor)
library(naniar)
library(visdat)
library(lubridate)
library(stringr)
library(gganimate)
library(lvplot)
```

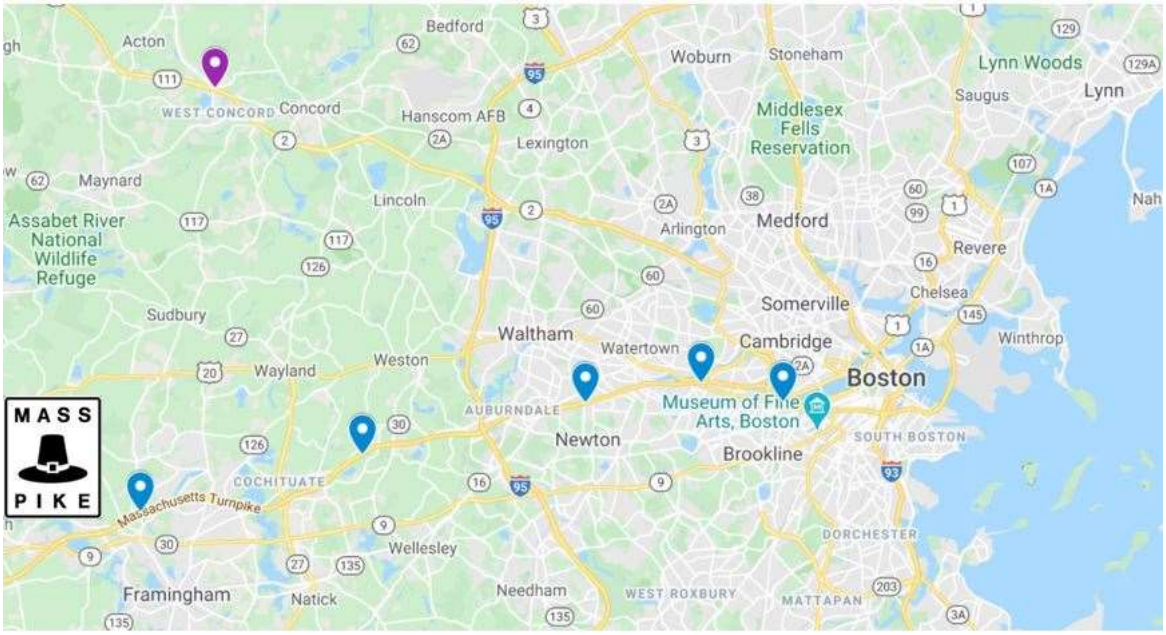
Data Source

The information that I will use for my final project is provided by the Massachusetts Department of Transportation, or MassDOT (https://mhd.ms2soft.com/tcds/tsearch.asp?loc=Mhd&mod=) for short. MassDOT provides public access to a Transportation Data Management System, which has multiple types of data: counts of vehicles per hour at a given location per day per month, which way cars drive through an intersection, and how fast cars traveled on average at given times. The data that I will focus on is the number of cars that pass through a given intersection per hour. Depending on how the system collects the traffic data, the total number of cars can be divided into Eastbound & Westbound or Northbound & Southbound. I am able to obtain these data by clicking through the calendars and downloading a monthly report in an Excel spreadsheet. Using the `readxl` package, I was able to load the Excel files into tibbles, which I worked with using techniques learned in class so far. I downloaded several years' worth of data for particular intersections by downloading the monthly reports available for a given year. Each monthly report has a similar name format: `MonthlyVolumeReport_m_Y.xlsx` . I made subfolders within my data folder to keep the locations separated from each other to collect and save data to prevent any confusion for each of the locations.

Geography

One intersection I am particularly interested in analyzing is a rotary (also called a traffic circle or roundabout. Massachusetts is the only state to use the word 'rotary') that is on a highway called Route 2. Although I do not live in Concord, Route 2 runs through my town and when traveling East Bound, leads to Cambridge and Boston. Route 2 in Concord is also called `elm_street` , which is what the variables for this location are called in my coding (this is all shown in the first image below). Although there is more than one busy rotary in Concord, I will refer to the Concord Rotary on Elm Street as simply the Concord Rotary. However, in my code, I refer to it as `elm_street` , as was in the heading of each Excel file. The second image shows this rotary by a purple pin to show where it is in relation to Boston- Massachusetts's sole reference mark for people who are not from Massachusetts. I'm sure Bostonians wouldn't mind being dragged into this. The other blue pins are locations along the Massachusetts Turnpike (no one calls it that, with Massachusetts residents choosing to say the abridged version, Mass Pike, instead. Sometimes even that name is too long and people within Boston will refer to it as The Pike). The Mass Pike goes from downtown Boston out to Western Massachusetts, connecting to a highway leading to Albany, New York. The locations I have selected, starting from the most Westward to the closest to downtown Boston are: Framingham (`fr`), Cochituate (`co`), Newton (`ne`), pre-Allston Interchange (`pr`), and post-Allston Interchange (`po`). In the last few years, a company called Raytheon (a defense contractor (https://www.rtx.com/) based out of Waltham, MA) was paid 130 million dollars to convert the outdated toll booths into electronic tolls that are able to process cars with and without an EZ pass transponder. The hope was to reduce the risk of accidents from people slowing down and merging at toll booths and to the workers, as talked about in The Boston Globe (https://www.bostonglobe.com/business/2014/08/13/raytheon-team-will-install-all-electronic-tolling-system-massachusetts-turpike/8JG6ROlJ9ekk77aaFACHuL/story.html#:~:text=The%20Massachusetts%20Department%20of%20Transportation,Williams%20tunnels%2C%20Raytheon%2
The plan to make toll booths electronic was unveiled in 2014, but not materialized until Friday October 28th, 2018 according to wbur (https://www.wbur.org/radioboston/2016/10/26/electronic-tolling). The result is shown in the third image from CBS Boston (https://boston.cbslocal.com/2016/10/27/mass-pike-electronic-tolling-what-you-should-know/).





Why these Data

As someone who lives in a somewhat rural suburb of Boston, I was always interested in going into Cambridge and Boston whenever I had the chance. After my senior year of high school, I was fortunate to have an internship where my Dad works at the Center for Astrophysics | Harvard & Smithsonian, located in Cambridge. For about two months, I rode along with my Dad to the office five days a week. Although the office was only 21.1 miles away, it took around 55 minutes each way to get there because of how bad morning and afternoon/evening traffic is. Before the coronavirus pandemic, millions of people commuted into Boston or Cambridge every work day. I'm interested to see if traffic has changed over time with the increasing registration of vehicles each year. I would also like to see if traffic has significantly improved with less cars on the road from more people working from home because of COVID. One of the main challenges for me was not to be overwhelmed by the amount of data that I'm downloading from MassDOT, and to make the right comparisons to properly analyze the data. This particular intersection has data extending back into the 1980s, so it would be really interesting to see how the volume of cars passing through has changed over the course of decades. The other challenge was learning how to conclude writing the report, because with each new plot, I had more questions I wanted to answer.

Initial Data Issues

```
elm_1_1997_raw <- read_excel("../data/elm_street/MonthlyVolumeReport_403_WB_1_1997.xlsx")
elm_1_1997_raw
```

```
## # A tibble: 39 x 27
##    `Massachusetts ~  ...2     ...3     ...4 ...5    ...6    ...7   ...8     ...9
##    <chr>            <dbl>    <dbl>    <dbl> <chr>  <dbl>   <dbl>  <dbl>    <dbl>
##  1 403_WB: Monthly~    NA  NA       NA      <NA>  NA      NA      NA      NA
##  2 <NA>                NA  NA       NA      <NA>  NA      NA      NA      NA
##  3 Location ID:        NA  NA       NA      403_~ NA      NA      NA      NA
##  4 County:             NA  NA       NA      Midd~ NA      NA      NA      NA
##  5 Funcationl Class    NA  NA       NA      3     NA      NA      NA      NA
##  6 Location:           NA  NA       NA      ELM ~ NA      NA      NA      NA
##  7 <NA>                NA  NA       NA      <NA>  NA      NA      NA      NA
##  8 <NA>                 0   0.0417   0.0833 0.125  0.167   0.208   0.25    0.292
##  9 1                  150 257      228      133   52      49      86     238
## 10 2                  189  72       42      46    45     150     624    2057
## # ... with 29 more rows, and 18 more variables: ...10 <dbl>, ...11 <dbl>,
## #   ...12 <dbl>, ...13 <chr>, ...14 <dbl>, ...15 <dbl>, ...16 <dbl>,
## #   ...17 <chr>, ...18 <dbl>, ...19 <dbl>, ...20 <dbl>, ...21 <dbl>,
## #   ...22 <dbl>, ...23 <dbl>, ...24 <dbl>, ...25 <dbl>, ...26 <chr>,
## #   ...27 <chr>
```

When I initially load the Excel file, I can immediately see that there is some sort of header at the top of the document. These data are not necessarily useful for me because I already know what month, year as well as location of the data points. Therefore, I will need to skip that every time I load a file.

```
times <- as.character(0:23)
elm_8_2018_skipped <- read_excel("../data/elm_street/MonthlyVolumeReport_403_WB_8_2018.xlsx", skip = 10, col_names = c("da
y", times, "Total", "QC Status"))
elm_8_2018_skipped
```

```
## # A tibble: 31 x 27
##      day   `0`   `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`  `10`  `11`
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1     1   353   103    84    56    92   289   675  1187  1438  1336  1322  1390
##  2     2   222   131    68    63    82   290   709  1144  1390  1308  1278  1401
##  3     3   312   140    99    79    98   258   691  1111  1296   741   535   670
##  4     4   336   193   119    86    74   179   390   673   812   808   609   596
##  5     5    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
##  6     6   278   192    81    41    87   281   656  1157  1085   752   864   504
##  7     7    49    49    19    20    44   242   607  1105   990   902   517   935
##  8     8   109    73    30    22    56   200   530   997   905   523  1107  1551
##  9     9   251   122    72    49    84   249   666  1224  1308  1352  1321  1523
## 10    10   357   131    91    72    96   271   671  1117  1300  1312  1420  1475
## # ... with 21 more rows, and 14 more variables: `12` <dbl>, `13` <dbl>,
## #   `14` <dbl>, `15` <dbl>, `16` <dbl>, `17` <dbl>, `18` <dbl>, `19` <dbl>,
## #   `20` <dbl>, `21` <dbl>, `22` <dbl>, `23` <dbl>, Total <dbl>, `QC
## #   Status` <chr>
```

When I skip the header, I see that I have tidy data because every column has a variable and every row has its own observation. However, a lot of those observations are NA values because data was not collected on that day for some reason. Therefore, I will need to find a way to remove all of those rows with NA observations in order to calculate monthly averages or plot all of the points in a month on a graph. I tried formatting it already, but I will need to replace the original column names with either numbers or numbers in the form of letters to prevent r from becoming confused. Another challenge will be choosing which types of graphs to plot my data depending on what question I'm trying to answer. For example, if I want to compare seasonal traffic to figure out which season has the highest traffic volume or if there is any major distinction between seasons, I would need a way to combine all of the summer months, winter months, etc.

Solving the Data Issues for the Concord Rotary

Since I have hundreds of Excel spreadsheets, reading them in one by one is not an option. Fortunately, for loops will be able to do all of the specified tasks for all of the Excel spreadsheets.

Step 1

```
# Part 1
files <- dir("../data/elm_street/", pattern = "\\.xlsx$", full.names = TRUE)

# Part 2
file_list <- list()
```

Part 1: The first step is to make a vector with a length equaling the number of downloaded Excel files. I also specify which type of files that should be read in by using a regular expression. The `\\` tells the string argument in pattern that the period should be treated as a period and not a special character. The `$` indicates that each file will have the `.xlsx` ending. There is also only one directory, allowing me to hard code in which subfolder of my final project data folder. **Part 2:** The second part of the first step involves setting up a blank list to hold all of the tibbles made by the for loop.

Step 2

```
# Part 1
times <- as.character(0:23)

# Part 2
for (i in files){
  file_list[[i]] <- read_excel(i, skip = 10, col_names = c("day", times, "Total", "QC Status")) %>%
    pivot_longer(c(`0`,`1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`, `10`, `11`, `12`, `13`, `14`, `15`, `16`, `17`, `18`, `1
9`, `20`, `21`, `22`, `23`),
                 names_to = "hour", values_to = "cars_per_hour") %>%
    select("day", "hour", "cars_per_hour")
}
```

Part 1: If I were to load in the times as is from the original Excel Spreadsheet, the program runs into problems reading the columns names in and performing math calculations. I sidestep this issue by defining a times vector that defines each number as a character to prevent there from being any funky calculations that mess up my column names. **Part 2:** Starting with file list 1, the loop reads the Excel file path vector (starting with January 1997) then pipes it into the pivot command. Each of what is inside of years matches the information in the file path because it was extracted from the path. This ultimately makes a list full of tibbles.

Step 3

```
# Part 1
months <- str_extract(names(file_list), "_403_[A-Z]B_\\d+") %>%
  str_sub(9) %>%
  as.double()

# Part 2
years <- str_extract(names(file_list), "\\d{4}") %>%
  as.double()

# Part 3
travel_direction <- str_extract(names(file_list), "_403_[A-Z]B") %>%
  str_sub(6)
```

For **Part 1**, I use this regular expression to get to the month. I am not sure what the `403` stands for, but it is a part of every name, and I treat it is a placeholder. There are two directions: EB (East Bound) or WB (West Bound). Both start with either E or W (basically any character) and end with a B. The next number of digits is what I'm interested in: 1 digit for months before October, 2 digits for the rest of the months. For **Part 2**, `years` is the only part that contains 4 numbers each time. For **Part 3**, I want to be able to distinguish between the direction of travel when making my master tibble.

Step 4

```
for (i in 1:length(files)){
  file_list[[i]] <- file_list[[i]] %>%
    mutate(hour = as.numeric(hour)) %>%
    mutate(full_date = make_datetime(year = years[i], month = months[i], day = day, hour = hour)) %>%
    mutate(day_of_week = wday(full_date, label = TRUE, abbr = FALSE)) %>%
    mutate(direction = travel_direction[i])
}
```

For each of the file paths stored in the files vector. Starting with file list 1- reading the Excel file path vector (starting with January 1997) then pipes it into a series of `mutate()` commands. Each of what is inside of years matches the information in the file path because it was extracted from the path. This ultimately makes a list full of tibbles

Step 5

```
elm_master <- bind_rows(file_list)
elm_master
```

```
## # A tibble: 101,496 x 6
##      day  hour cars_per_hour full_date           day_of_week direction
##    <dbl> <dbl>         <dbl> <dttm>              <ord>       <chr>
##  1     1     0           217 1997-01-01 00:00:00 Wednesday   EB
##  2     1     1           331 1997-01-01 01:00:00 Wednesday   EB
##  3     1     2           535 1997-01-01 02:00:00 Wednesday   EB
##  4     1     3           380 1997-01-01 03:00:00 Wednesday   EB
##  5     1     4           191 1997-01-01 04:00:00 Wednesday   EB
##  6     1     5            69 1997-01-01 05:00:00 Wednesday   EB
##  7     1     6            40 1997-01-01 06:00:00 Wednesday   EB
##  8     1     7            87 1997-01-01 07:00:00 Wednesday   EB
##  9     1     8           219 1997-01-01 08:00:00 Wednesday   EB
## 10     1     9           247 1997-01-01 09:00:00 Wednesday   EB
## # ... with 101,486 more rows
```

I finally have a tibble with all of the necessary information needed to visualize it.

Solving the Data Issues for the Mass Pike Locations

Step 1

```
# Part 1
paths_to_open <- c("../data/mass_pike/framingham",
                   "../data/mass_pike/cochituate",
                   "../data/mass_pike/newton",
                   "../data/mass_pike/pre_allston",
                   "../data/mass_pike/post_allston"
                   )

# Part 2
files <- dir(paths_to_open, pattern = "\\.xlsx$", full.names = TRUE)

# Part 3
pike_file_list <- list()
```

Part 1: The first part of the first step is documenting the different paths for each of the subfolders that contain all of the same Excel spreadsheets. **Part 2:** Making a vector with length = number of downloaded Excel files. **Part 3:** Creating an empty list to store the outputs from the next two for loops.

Step 2

```
for (i in files){
  pike_file_list[[i]] <- read_excel(i, skip = 10, col_names = c("day", times, "Total", "QC Status")) %>%
    pivot_longer(c(`0`,`1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`, `10`, `11`, `12`, `13`, `14`, `15`, `16`, `17`, `18`, `1
9`, `20`, `21`, `22`, `23`),
                 names_to = "hour", values_to = "cars_per_hour") %>%
    select("day", "hour", "cars_per_hour", "Total")
}
```

This part is set up exactly the same as `Step 2` is in the Solving the Data Issues for the `elm_street` location. The same `times` character vector is used.

Step 3

```
# Part 1
months <- str_extract(names(pike_file_list), "_[A-Z]B_\\d+") %>%
  str_sub(start = 5) %>%
  as.double()

# Part 2
years <- str_extract(names(pike_file_list), "\\d{4}") %>%
  as.double()

# Part 3
travel_direction <- str_extract(names(pike_file_list), "_[A-Z]B_") %>%
  str_sub(start = 2L, end = -2L)

# Part 4
location <- str_extract(names(pike_file_list), "/mass_pike/..") %>%
  str_sub(start = 12L) %>%
  as.character()
```

Part 1: The files have more or less the same format:
`MonthlyVolumeReport_AET[unique identifier]_[direction of travel]_[month]_[year].xlsx` . For the months, I use this regular expression to get to the month. I am not sure what the `AET` stands for, but it is a part of every name, and I treat it is a placeholder. There are two directions: EB (East Bound) or WB (West Bound). Both start with either E or W (basically any character) and end with a B. The next number of digits is what I'm interested in: 1 digit for months before October, 2 digits for the rest of the months. **Part 2:** `Years` is the only part that contains 4 numbers each time. **Part 3:** I want to be able to distinguish between the direction of travel when making my master tibble. **Part 4:** This is the extra step that is not needed for the `elm_street` location. For the `location` on the Mass Pike, I selected several spots along the Mass Pike, so I needed to distinguish that (corresponding to the pins in the image of Eastern Massachusetts).

Step 4

```
for (i in 1:length(files)){
  pike_file_list[[i]] <- pike_file_list[[i]] %>%
    mutate(hour = as.numeric(hour)) %>%
    mutate(full_date = make_datetime(year = years[i], month = months[i], day = day, hour = hour)) %>%
    mutate(day_of_week = wday(full_date, label = TRUE, abbr = FALSE)) %>%
    mutate(direction = travel_direction[i]) %>%
    mutate(location = location[i])
}
```

This part is set up exactly the same as `Step 4` is in the Solving the Data Issues for the `elm_street` location. However, the main difference this for loop is that multiple file paths are opened and the addition of a location column to identify each observation.

Step 5

```
# Part 1
pike_master <- bind_rows(pike_file_list)

# Part 2
pike_master <- pike_master %>%
  mutate(daily_percent = cars_per_hour / Total * 100)
```

Part 1: All of the steps up until `Step 4` make a tibble out of each Excel file. `file_list` is now a list of tibbles with the same number of columns, but not necessarily the same number of observations because not all months had the same amount of data (Some months have 30 while others have 31 days, or the sensors could have been turned off that day). The `bind_rows()` function stacks all of the observations with the same number of columns on top of each other to make a `master tibble`, or a tibble with all of the information from all of the read in Excel spreadsheets. **Part 2:** The `daily_percent` column needs to be defined here. It calculates the percentage of cars traveling through that intersection in a particular hour of the total cars that pass through on that day.

Missing Data

Concord Rotarty

Looking at the observations as a whole, only `7.9%` of the data are missing. When I look at each location in particular and facet the missing data plots by year, there does not seem to be any pattern or large gaps in missing data. This implies that the data that I have selected are reliable and I can feel more comfortable in interpreting future aspects of the project for the Concord Rotary location. The work behind this can be found in the Missing Data section of the Supplemental Materials section.

Mass Pike

Looking at the observations as a whole, only `3.8%` of the data are missing. When I look at each location in particular and facet the missing data plots by year, there does not seem to be any pattern or large gaps in missing data. This implies that the data that I have selected are reliable and I can feel more comfortable in interpreting future aspects of the project for the Mass Pike location. The work behind this can be found in the Missing Data section of the Supplemental Materials section.

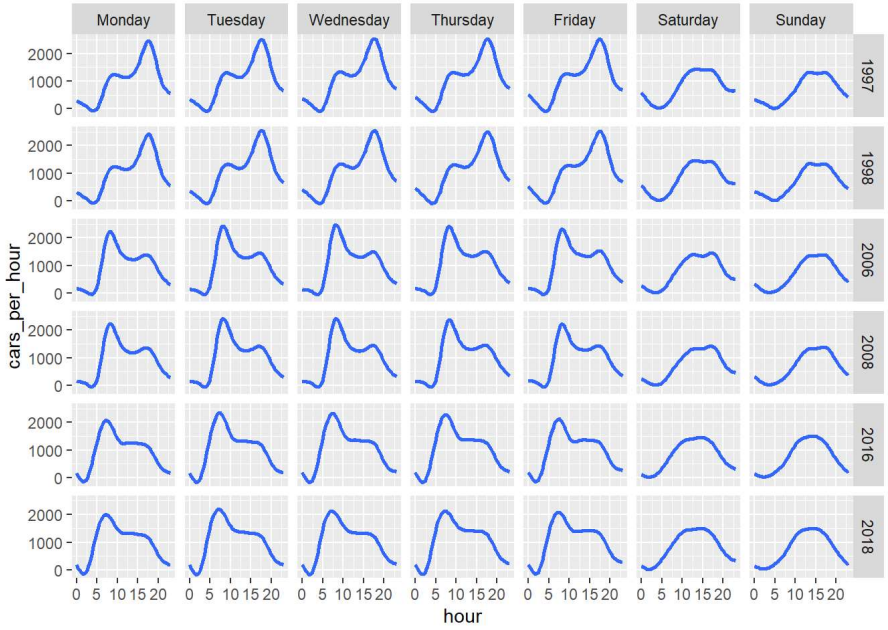
Weekends vs Weekdays

The purpose of my project was to understand commuting patterns over time. When I was in high school, I used to drive through the Concord Rotary for soccer practice after school while traveling East Bound, and could not help but notice how empty the East Bound lanes were in comparison to the congested the West Bound traffic during rush hour. On the weekends, when my family would drive into Newton, the Concord Rotary seemed almost bare. I wanted to see if traffic on the weekends would negatively affect `geom_smooth()` and its ability to characterize the typical commuting traffic patterns. In terms of the code used, I took advantage of the `filter()` function to select the two directions one at a time. Since there are so many points in this data set, it will help to divide the data set into half and look at each half separately. `fct_relevel()` to group Saturday and Sunday next to each other to see them better with respect to weekdays when making the comparison.

Concord Rotary

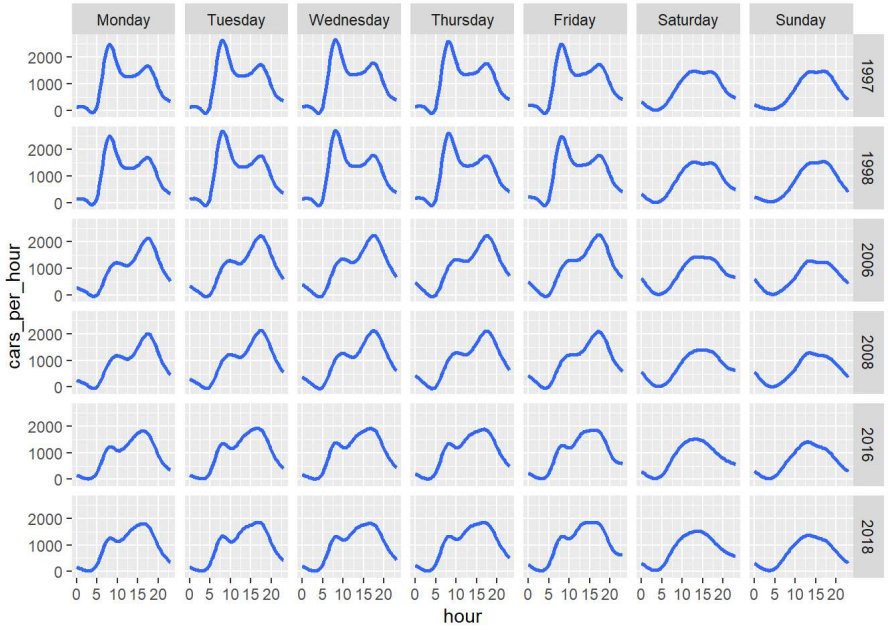
East Bound

```
elm_master %>%
  filter(direction == "EB") %>%
  mutate(day_of_week = fct_relevel(day_of_week, c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunda
y"))) %>%
  mutate(year = as.factor(year(full_date))) %>%
  ggplot(aes(x = hour, y = cars_per_hour)) +
  geom_smooth(se = FALSE) +
  facet_grid(year~day_of_week)
```



West Bound

```
elm_master %>%
  filter(direction == "WB") %>%
  mutate(day_of_week = fct_relevel(day_of_week, c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunda
y"))) %>%
  mutate(year = as.factor(year(full_date))) %>%
  ggplot(aes(x = hour, y = cars_per_hour)) +
  geom_smooth(se = FALSE) +
  facet_grid(year~day_of_week)
```



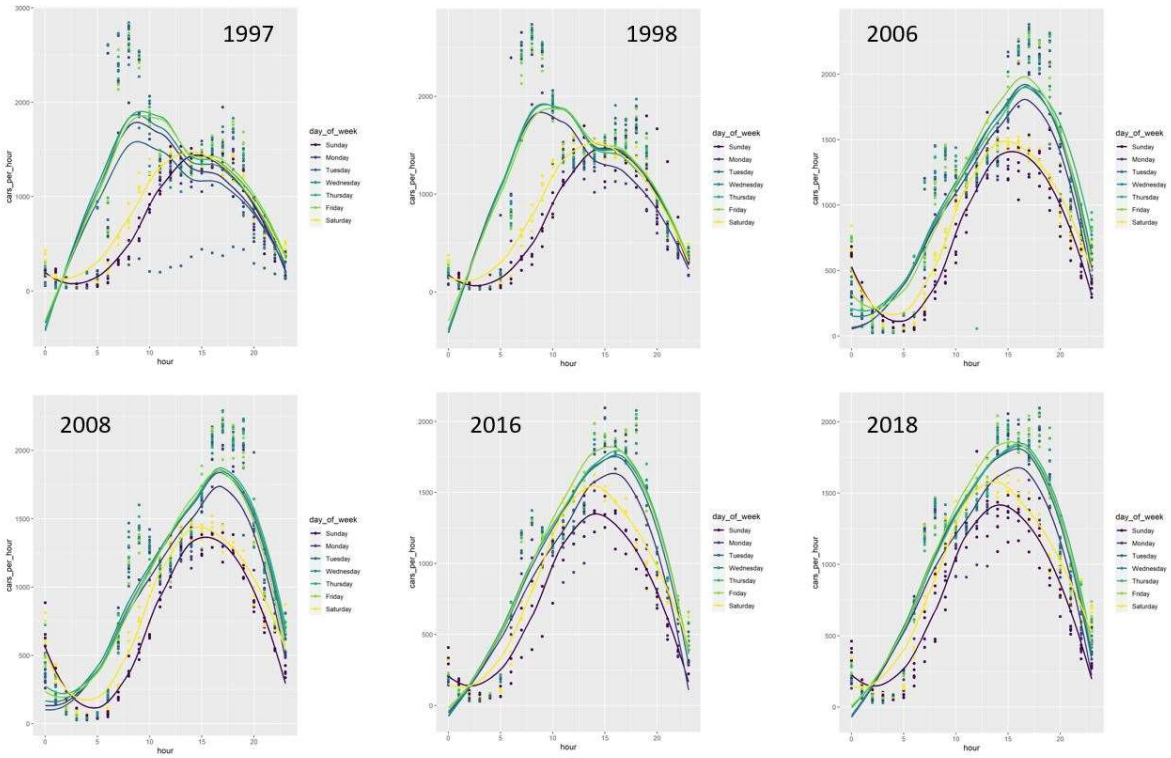
Looking at the plots, it highlights how the data from `1997` and `1998` differ from data after `2006`. As the dates approach the present day, there is a broadening of the main peaks, extending to earlier times in the morning (meaning an earlier and longer rush "hour") and later times in the evening (meaning a longer rush "hour").

Panel View

This is an alternative way of looking at weekdays versus weekends (taking one month and looking at it across several years). I made a function that made saved `ggplot` images for each month in each year of data.

```
for (i in 1:length(files)){
  file_list[[i]] %>%
    ggplot() +
    aes(x = hour, y = cars_per_hour, color = day_of_week) +
    geom_point() +
    geom_smooth(se = FALSE)
    ggsave(filename = paste(years[i], months[i], ".png", sep = " "))
```

The function works by taking the contents of each read in Excel file using `file_list[[i]]`, and then plots accordingly, coloring each line by day of the week. the `ggsave` uses information from the file in the `file_list` and uses the `year` and the `month` information to name it while saving it as a `png` file. The `paste()` function combines all of the information to tell `ggsave` what the `filename` is. I made a panel of the month of April varying over time with the years labeled accordingly. The `geom_smooth()` function using the `y~x` argument has trouble representing the two peaks for the two rush hours during the week while it seems to represent the one peak of traffic on the weekends.

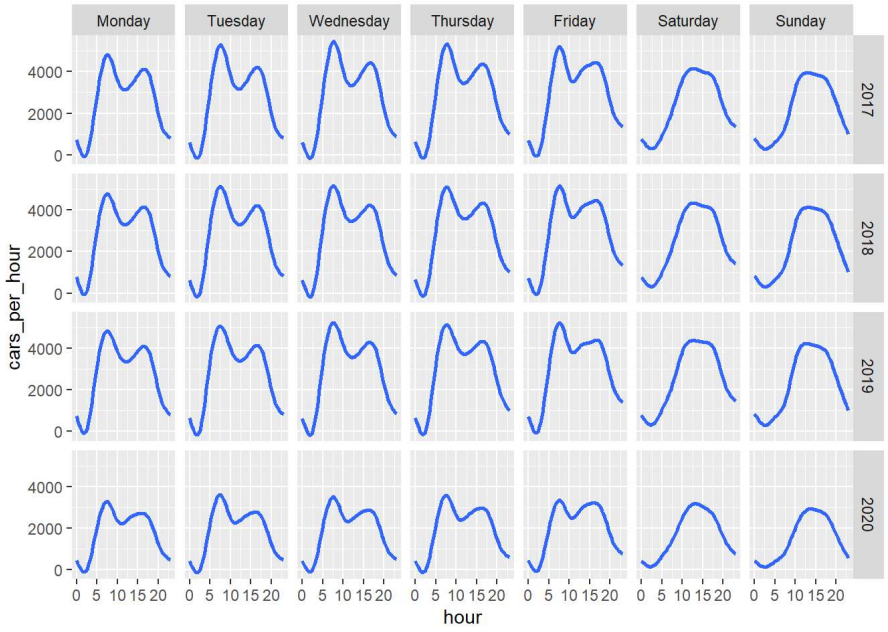


Regardless of year, Saturdays and Sundays have a lower average than any of the commuting days.

Mass Pike

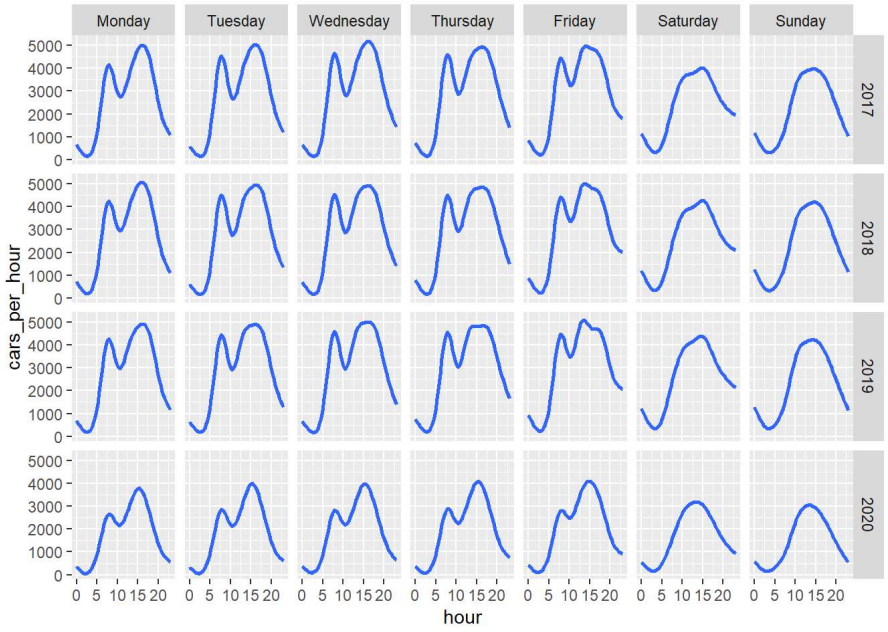
East Bound

```
# East Bound
pike_master %>%
  filter(direction == "EB") %>%
  mutate(day_of_week = fct_relevel(day_of_week, c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunda
y"))) %>%
  mutate(year = as.factor(year(full_date))) %>%
  ggplot(aes(x = hour, y = cars_per_hour)) +
  geom_smooth(se = FALSE) +
  facet_grid(year~day_of_week)
```



West Bound

```
pike_master %>%
  filter(direction == "WB") %>%
  mutate(day_of_week = fct_relevel(day_of_week, c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunda
y"))) %>%
  mutate(year = as.factor(year(full_date))) %>%
  ggplot(aes(x = hour, y = cars_per_hour)) +
  geom_smooth(se = FALSE) +
  facet_grid(year~day_of_week)
```



At an overall level, there are similar patterns in these plots as in the plots from the Concord Rotary section. The Concord Rotary location does not have available data for `2020`, which is rather unfortunate (maybe I need to call someone at MassDOT and plead with them to turn the sensors back on). Fortunately, the Mass Pike data has `2020` and shows the impact the coronavirus pandemic has had with the overall amount of traffic dropping on average. Looking at the previous years, it is not as clear if the peaks have shifted earlier for East Bound and later for West Bound, but the valley in between the two peaks becomes shallower over time. This suggests that there is a fair amount of travel around lunch time, perhaps to avoid traffic peak hours.

Filtering out the Weekends for Both Sets

```
# Concord Rotary Without Weekends
elm_master_EB <- elm_master %>%
  filter(direction == "EB") %>%
  filter(day_of_week != "Saturday" & day_of_week != "Sunday")

elm_master_WB <- elm_master %>%
  filter(direction == "WB") %>%
  filter(day_of_week != "Saturday" & day_of_week != "Sunday")


# Mass Pike Without Weekends
pike_master_EB <- pike_master %>%
  drop_na() %>%
  filter(day_of_week != "Saturday" & day_of_week != "Sunday") %>%
  filter(direction == "EB") %>%
  mutate(location = fct_relevel(location, c("fr", "co", "ne", "pr", "po")))

pike_master_WB <- pike_master %>%
  drop_na() %>%
  filter(day_of_week != "Saturday" & day_of_week != "Sunday") %>%
  filter(direction == "WB") %>%
  mutate(location = fct_relevel(location, c("fr", "co", "ne", "pr", "po")))
```

Flux

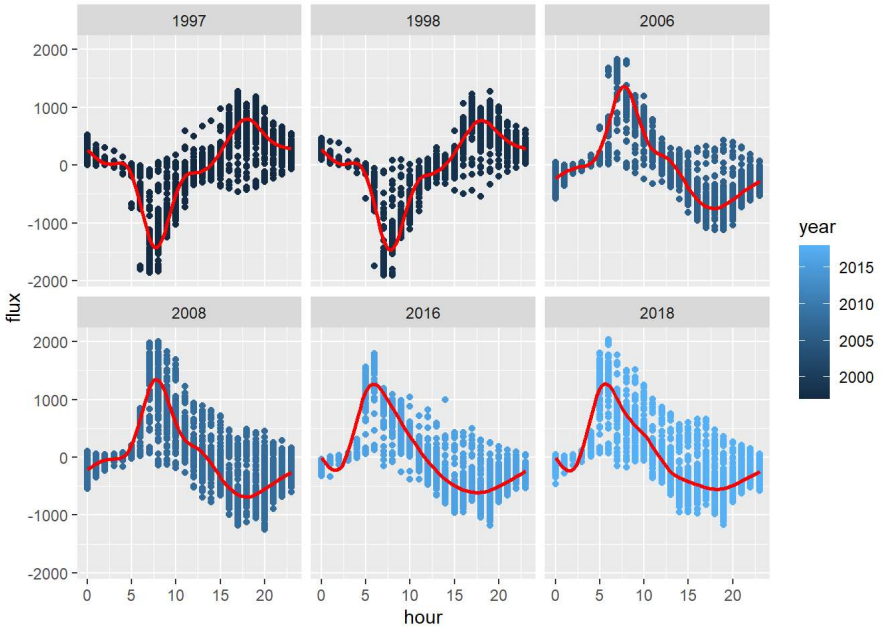
The next metric that I looked at is a term that is traditionally used in subjects like Astronomy or my Physics E&M and calculus classes, but the concept can be applied here. Since not every person who drives in takes the same route home (with reasons ranging from stopping by at the grocery store, picking kids up from school, or taking the back roads to avoid highway congestion). In this context, I define the Flux as the number of cars heading in the East Bound direction per hour minus the number of cars heading in the West Bound direction per hour. A positive Flux corresponds to more cars heading in the East Bound direction while a negative Flux means more cars are heading in the West Bound direction at that time. The sign of the Flux does not mean that less people overall are traveling in a certain direction, just that less people are choosing this specific location.

Concord Rotary

```
# Part 1
elm_master_EB_flux <- elm_master_EB %>%
  select(day, hour, cars_per_hour, day_of_week, direction, full_date)
elm_master_WB_flux <- elm_master_WB %>%
  select(full_date, day, hour, cars_per_hour, day_of_week, direction)

# Part 2
elm_master_flux_per_hour <-
  left_join(elm_master_EB_flux, elm_master_WB_flux, by = c("full_date", "hour")) %>%
  mutate(day = day.x) %>%
  mutate(month = month(full_date)) %>%
  mutate(year = year(full_date)) %>%
  mutate(cars_per_hour_EB = cars_per_hour.x) %>%
  mutate(cars_per_hour_WB = cars_per_hour.y) %>%
  mutate(day_of_week = day_of_week.x) %>%
  mutate(flux = (cars_per_hour_EB - cars_per_hour_WB)) %>%
  select(full_date, day, hour, month, year, cars_per_hour_EB, cars_per_hour_WB, day_of_week, flux)
```

Part 1: I wanted to select only the columns that were necessary to identify each observation and to calculate the Flux. **Part 2:** I used `left_join()` with `full_date` and `hour` as the keys used to join the `elm_master_EB_flux` and the `elm_master_WB_flux` tibbles. I chose to make a `left join` because I wanted to be able to easily calculate the Flux: `cars_per_hour_EB - cars_per_hour_WB` . Since the tibbles had the same column names, in order to avoid adding confusion by having names like `day.x` / `day.y` and `day_of_week.x` / `day_of_week.y` , I changed those to `day` and `day_of_week` respectively using the `mutate()` function.

```
ggplot(data = elm_master_flux_per_hour, mapping = aes(x = hour, y = flux, color = year)) +
  geom_point() +
  geom_smooth(color = "RED") +
  facet_wrap(~year, nrow = 2)
```



One of the first observations I make is that `1997` and `1998` are different from the rest of the years onward. Looking at the *Weekends vs Weekdays* section earlier, the data from `1997` and `1998` do not follow the trend. One possible explanation is that the sensor was in a different location from where it is today. Another possibility is that the data were entered incorrectly. The third explanation I can think of is that there used to be a large amount of people who commuted out West or at least avoided taking the main roads at peak times to avoid traffic (I think that this is the least likely of the three scenarios). Looking at the plots from `2006`, `2008`, `2016`, and `2018`, one thing is clear: rush hour is starting earlier in the morning and rush hour in the evening has less cars passing through per hour, but for a longer period of time. More and more people are avoiding the rotary in the evening if they can. Although the peak is declining, from personal experience, this does not mean that rush "hour" has gotten better. Most people take the exit directly across from where they enter (staying on Route 2), but perhaps traffic from the three side streets that enter the rotary at other points cause the rotary to slow down, decreasing the capacity of cars that can pass through per hour.

Mass Pike

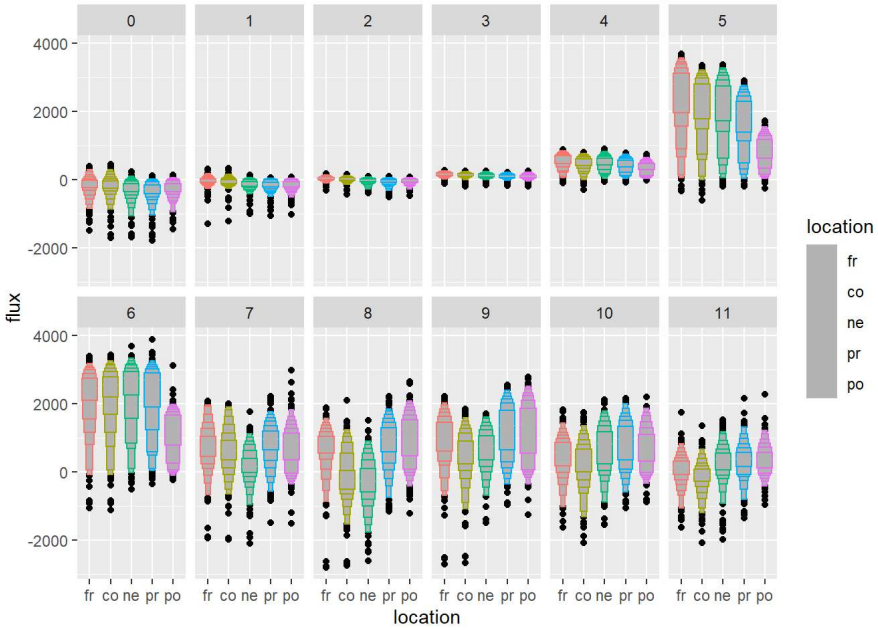
Flux Per Hour

```
# Part 1
pike_master_EB_flux <- pike_master_EB %>%
  select(day, hour, cars_per_hour, day_of_week, direction, location, full_date)
pike_master_WB_flux <- pike_master_WB %>%
  select(full_date, day, hour, cars_per_hour, day_of_week, direction, location)

# Part 2
pike_master_flux_per_hour <-
  left_join(pike_master_EB_flux, pike_master_WB_flux, by = c("full_date", "location", "hour")) %>%
  mutate(day = day.x) %>%
  mutate(month = month(full_date)) %>%
  mutate(year = year(full_date)) %>%
  mutate(cars_per_hour_EB = cars_per_hour.x) %>%
  mutate(cars_per_hour_WB = cars_per_hour.y) %>%
  mutate(day_of_week = day_of_week.x) %>%
  mutate(flux = (cars_per_hour_EB - cars_per_hour_WB)) %>%
  select(full_date, day, hour, month, year, cars_per_hour_EB, cars_per_hour_WB, day_of_week, location, flux)
```

Similar to the previous section, for **Part 1,** I wanted to select only the columns that were necessary to identify each observation and to calculate the Flux. **Part 2:** I used again `left_join()`, but this time using `full_date`, `hour`, and `location` as the keys used to join the `pike_master_EB_flux` and the `pike_master_WB_flux` tibbles. When I did not include `location` as a key, I ran into a problem where my joined tibble was about 5 times the length as either of my `pike_master_EB` or `pike_master_WB` tibbles. It took some time to figure out, but I also realized that I had 5 different locations. It turns out that my `left_join()` was joining the same data point in my `pike_master_EB` data set with each of the 5 locations in the `pike_master_WB` data set. After adding the `location` as a key, this problem went away, and I was able to plot the results accurately. Plotting all of the hours on one plot was too small to make sense of, so I separated `AM` hours (before 12) and `PM` hours (including 12 and onward) and plotted them as two separate graphs.

AM Hours

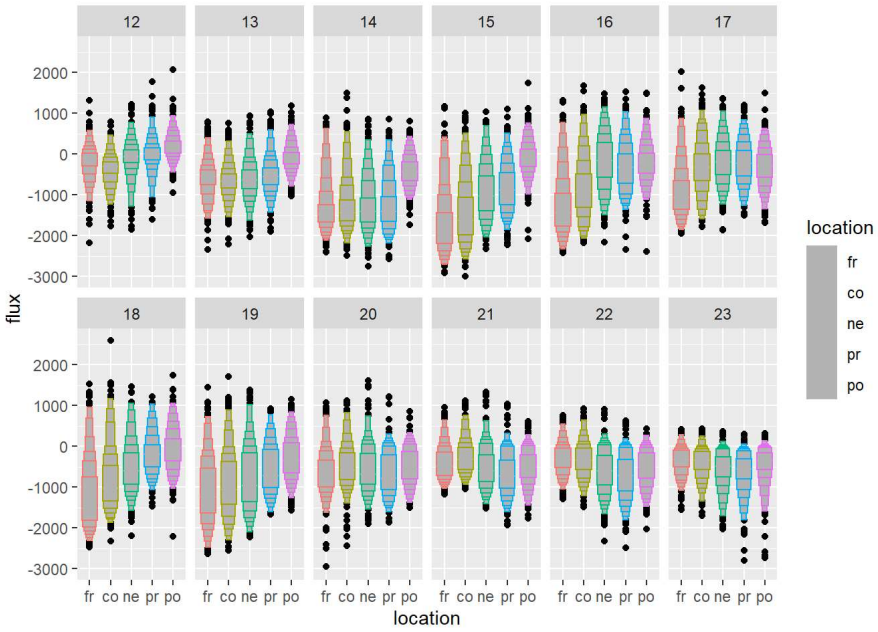
```
pike_master_flux_per_hour %>%
  filter(hour < 12) %>%
  ggplot(aes(x = location, y = flux, color = location)) +
  geom_lv() +
  facet_wrap(~hour, nrow = 2)
```



It makes sense that Flux is positive in the morning (`cars_per_hour_EB - cars_per_hour_WB` ; where `cars_per_hour_EB > cars_per_hour_WB`). The main jump in `Flux` occurs around anywhere from `5AM` to `6AM` .

PM Hours

```
# Looking at PM hours
pike_master_flux_per_hour %>%
  filter(hour >= 12) %>%
  ggplot(aes(x = location, y = flux, color = location)) +
  geom_lv() +
  facet_wrap(~hour, nrow = 2)
```



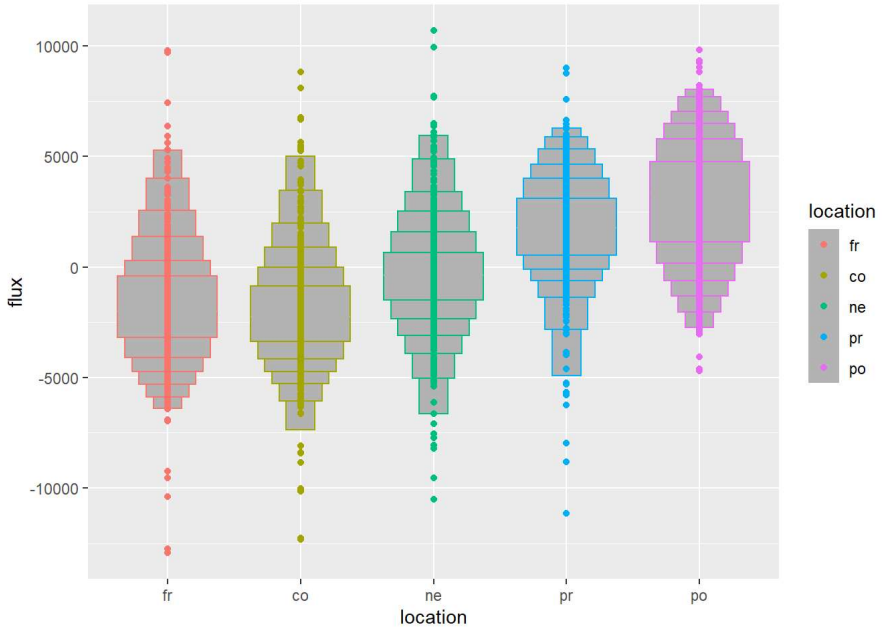
It makes sense that Flux is negative in the evening (`cars_per_hour_EB - cars_per_hour_WB` ; where `cars_per_hour_EB < cars_per_hour_WB`). The main decrease in `Flux` occurs around anywhere from `7PM` to `8PM` .

Flux Overall

```
# Part 1
pike_master_EB_11_hour <- pike_master_EB %>%
  filter(hour == 11) %>%
  select(day, hour, Total, day_of_week, direction, location, full_date)
pike_master_WB_11_hour <- pike_master_WB %>%
  filter(hour == 11) %>%
  select(full_date, day, hour, Total, day_of_week, direction, location)

# Part 2
pike_master_EB_11_hour_location <- pike_master_EB_11_hour %>%
  group_by(location)
pike_master_WB_11_hour_location <- pike_master_WB_11_hour %>%
  group_by(location)

# Part 3
pike_master_flux_overall <-
  left_join(pike_master_EB_11_hour_location, pike_master_WB_11_hour_location, by = c("full_date", "location")) %>%
  mutate(day = day.x) %>%
  mutate(month = month(full_date)) %>%
  mutate(hour = hour.x) %>%
  mutate(year = year(full_date)) %>%
  mutate(Total_EB = Total.x) %>%
  mutate(Total_WB = Total.y) %>%
  mutate(day_of_week = day_of_week.x) %>%
  mutate(flux = (Total_EB - Total_WB)) %>%
  mutate(full_date = as.character(full_date)) %>%
  select(full_date, day, hour, month, year, Total_EB, Total_WB, day_of_week, location, flux)
```

Part 1: Since the `Total` column is the same for all hours of the day (all 24 hours in a day have the same number of total cars that traveled that day), I will filter out an hour that meets the pun-making requirement. **Part 2:** I want to `group_by()` the `location` because I want to calculate the `Flux` per location. **Part 3:** The `pike_master_EB_11_hour_location` and `pike_master_WB_11_hour_location` data sets need to be joined by 2 keys, otherwise the `left_join()` function will recycle values from the `pike_master_WB_11_hour_location` data set and make a joined table that is 5 times longer than the desired length (recycling through the 5 different locations). Since `day.x = day.y`, `hour.x = hour.y`, `Total_EB = Total.x`, `Total_WB = Total.y`, and `day_of_week = day_of_week.x`, I used multiple `mutate()` commands to simplify the names.

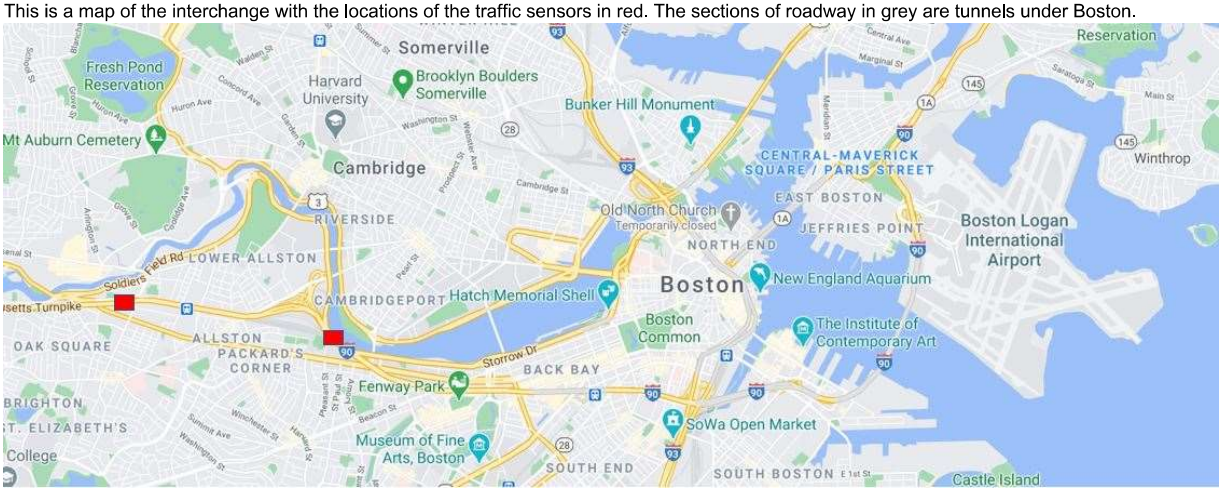
```
ggplot(pike_master_flux_overall, aes(x = location, y = flux, color = location)) +
  geom_lv() +
  geom_point()
```



One way to think of this is the net hourly flux. This shows how all of the hours fit. In the Framingham location, it seems that most of the time, people tend to travel West Bound because the bulk of the `Flux` is negative. However, near the Post Allston Interchange location, `Flux` is positive, indicating that most of the people travel East Bound. Considering that this is a major route to Logan International Airport, perhaps airport traffic contributes to this.

Map of the Allston Interchange

This is a map of the interchange with the locations of the traffic sensors in red. The sections of roadway in grey are tunnels under Boston.



gganimate

Necessary Code

From previous parts such as *Weekends vs Weekdays* and *Flux at the Concord Rotary*, I noticed that data prior to the year `2000` tended to run contrary to traffic trends seen today. The same trend is observed here when animated.

```
# Part 1
elm_master_EB_wkd_with_year <- elm_master_EB_wkd %>%
  mutate(year = year(full_date))

# Part 2
elm_master_EB_wkd_with_year_anim <- ggplot(elm_master_EB_wkd_with_year) +
  geom_point(aes(x = hour, y = cars_per_hour, color = year)) +
  transition_states(year,
                    transition_length = 2,
                    state_length = 1)
# Part 3
animate(elm_master_EB_wkd_with_year_anim,
        renderer = file_renderer(dir = "../data/elm_street/gganimate_EB_wkd", prefix = "gganim_frame", overwrite = FALSE)
        )
```

Part: 1 For some reason, it does not like it when I do `year(full_date)`. Therefore, I chose add an extra column to deal with this. This process is about saving my 100 animation pictures neatly into their own little folders. **Part 2:** This part specifies how long in between years I want to show (`transition_length`) and how long I want the animation to show each year (`state_length`). **Part 3:** The `file_renderer()` function takes into account which directory, or subfolder, I want to save the images to and what the file name will be called. From there, I used Photoshop to stitch the 100 image output into videos. I did this process for `EB` and `WB`, splitting up all of the data, and data after the year `2000`. I included other videos in the Supplemental Materials section.

Concord Rotary, East Bound, weekdays:

0:00 / 0:20

Concord Rotary, West Bound, weekdays:

0:00 / 0:20

Mass Pike Congestion Visualization

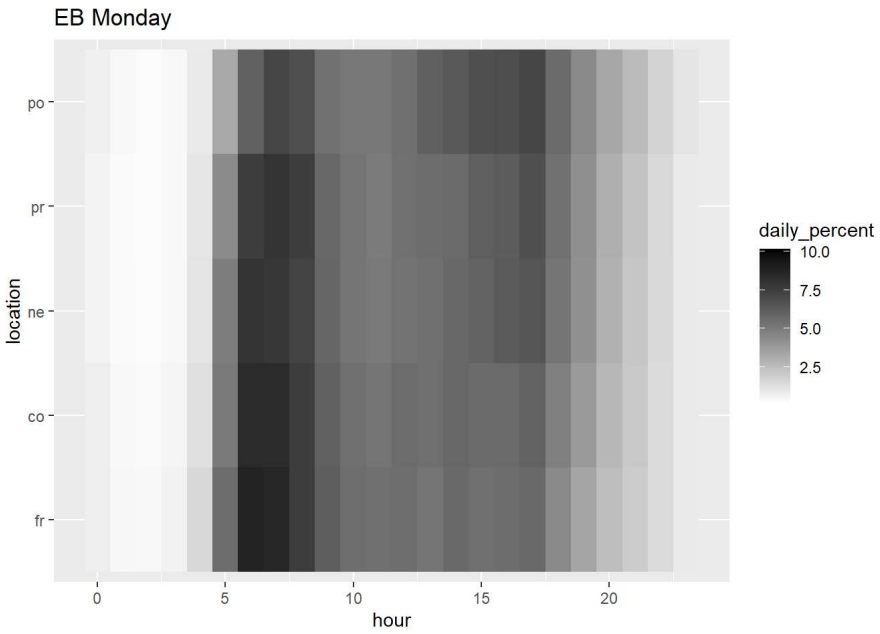
Weekdays

The following tile plots show data for Mondays (as an example) in both the East Bound and West Bound directions. The tile plots for the rest of the days are found in the Supplemental Materials section. These plots show the percentage of the total traffic per a given hour at each location on the Mass Pike. Darker tiles indicate higher traffic levels. The locations starting from the bottom (Framingham) are the most distant from Boston while the location at the top is closest to the center of Boston.

East Bound

```
pike_master_EB %>%
  filter(day_of_week == "Monday") %>%
  group_by(day_of_week) %>%
  group_map(~ggplot(., aes(x = hour, y = location)) +
              geom_tile(aes(fill = daily_percent)) +
              scale_fill_gradient(low = "white", high = "black") +
              labs(title = "EB Monday"))
```

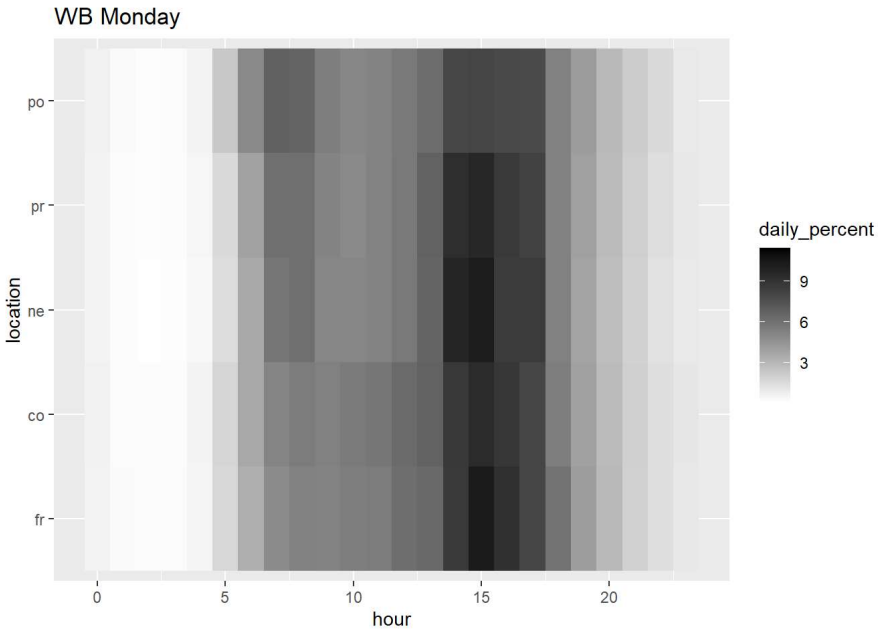
```
## [[1]]
```



West Bound

```
pike_master_WB %>%
  filter(day_of_week == "Monday") %>%
  group_by(day_of_week) %>%
  group_map(~ggplot(., aes(x = hour, y = location)) +
              geom_tile(aes(fill = daily_percent)) +
              scale_fill_gradient(low = "white", high = "black") +
              labs(title = "WB Monday"))
```

```
## [[1]]
```



Theoretically, if there was no rush hour, then cars would randomly travel at any hour of the day. If people leaving for work was random, then I would expect to see 100%/24 hours, or about 4.2% of traffic traveling at any given hour. I define this as `daily_percent`. When 10% of the total cars that pass through an intersection (`daily_percent` ~ 10)do so at one hour in particular, that intersection can be considered congested. Looking in the Supplemental Materials section, there does not appear to be some clear pattern of traffic on weekdays. Traffic on Friday perhaps seems to not be as bad, but I would need to do a more in depth analysis to make any determination. Congestion seems to be worst in the more distant from Boston locations.

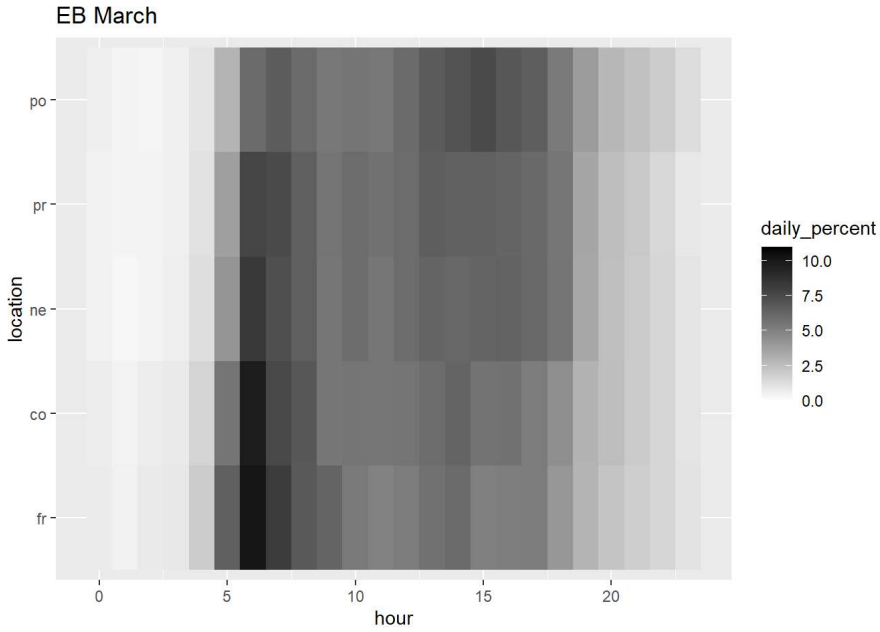
Months

The following tile plots show data for March (as an example) in both the East Bound and West Bound directions. The tile plots for the rest of the months are found in the Supplemental Materials section. Similar to the previous section, these plots show the percentage of the total traffic per a given hour at each location on the Mass Pike. Darker tiles indicate higher traffic levels.

East Bound

```
pike_master_EB %>%
  mutate(month = month(full_date)) %>%
  filter(month == 3) %>%
  group_by(month) %>%
  group_map(~ggplot(., aes(x = hour, y = location)) +
              geom_tile(aes(fill = daily_percent)) +
              scale_fill_gradient(low = "white", high = "black", limits = c(0, 11)) +
              labs(title = "EB March"))
```

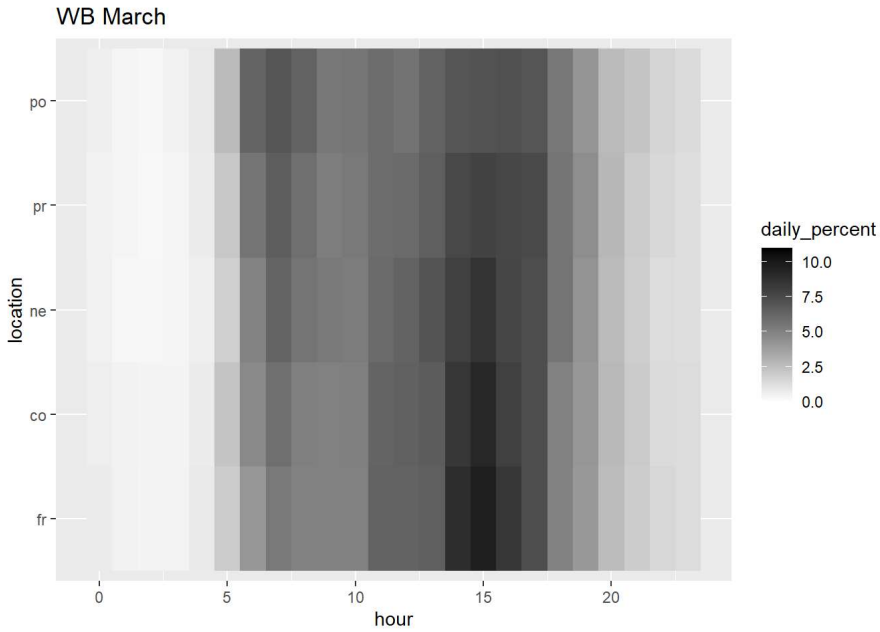
```
## [[1]]
```



West Bound

```
pike_master_WB %>%
  mutate(month = month(full_date)) %>%
  filter(month == 3) %>%
  group_by(month) %>%
  group_map(~ggplot(., aes(x = hour, y = location)) +
              geom_tile(aes(fill = daily_percent)) +
              scale_fill_gradient(low = "white", high = "black", limits = c(0, 11)) +
              labs(title = "WB March"))
```

```
## [[1]]
```



Looking at information in the Supplemental Materials section, November looks like it is the worst month of congestion, with Framingham, Cochituate, and Newton locations experiencing heavily congested hours both in the morning and in the evening. November tends to have about four hours in the morning for East Bound and in the evening for West Bound where `daily_percent` is nearly 10%. However, in months like June and July, the rush hour in the morning and the evening does not look as intense because the tiles are not as dark.

Before and After Electronic Tolls on the Mass Pike

The way I approached this problem is kind of like realizing I have been trying to stick a square peg in a round hole. I was unable to locate the square peg, but I got out my hammer and used **Brute Force**™ to make it fit.

```
# Setting up the Next Code Chunk
pike_master_flux_day <-
  left_join(pike_master_EB_11_hour_location, pike_master_WB_11_hour_location, by = c("full_date", "location")) %>%
  mutate(day = day.x) %>%
  mutate(month = month(full_date)) %>%
  mutate(hour = hour.x) %>%
  mutate(year = year(full_date)) %>%
  mutate(Total_EB = Total.x) %>%
  mutate(Total_WB = Total.y) %>%
  mutate(day_of_week = day_of_week.x) %>%
  mutate(flux = (Total_EB - Total_WB)) %>%
  mutate(full_date = as.character(full_date)) %>%
  select(full_date, day, hour, month, year, Total_EB, Total_WB, day_of_week, location, flux)
```

Same reasoning as the section creating `pike_master_flux_hour` .

Building Datasets for `2017` and `2019`

```
# 2017
# Part 1
pike_master_flux_day_2017 <- pike_master_flux_day %>%
  filter((full_date >= ymd(20170101)) & (full_date <= ymd(20171231))) %>%
  mutate(pre_year_total = sum(Total_EB + Total_WB)) %>%
  count(pre_year_total) %>%
  mutate(location = fct_relevel(location, c("fr", "co", "ne", "pr", "po")))

# Part 2
tibble_2017 <- tibble(
  location = c("fr", "co", "ne", "pr", "po"),
  time = "2017"
) %>%
  mutate(location = fct_relevel(location, c("fr", "co", "ne", "pr", "po")))

# Part 3
pike_master_flux_day_2017 <-
  left_join(pike_master_flux_day_2017, tibble_2017, by = "location")
```

Part 1: This part filters for all dates including January 1st to December 31st of 2017. **Part 2:** The goal is to get the number of observations and record whether those observations were made before (pre) or after (post) the implementation of electronic tolls on the Mass Pike. The Supplemental Materials sections details each step and explains why I selected `2017` and `2019` instead of all of the days before and all of the days after electronic toll implementation. **Part 3:** This is the part where I use **Brute Force**™ to make this work. Certainly not the most pretty code, but it gets the job done. Essentially this and the previous part add a column to the first tibble called `time` where I specify the year, `2017`.

```
# 2019
# Part 1
pike_master_flux_day_2019 <- pike_master_flux_day %>%
  filter((full_date >= ymd(20190101)) & (full_date <= ymd(20191231))) %>%
  mutate(post_year_total = sum(Total_EB + Total_WB)) %>%
  count(post_year_total) %>%
  mutate(location = fct_relevel(location, c("fr", "co", "ne", "pr", "po")))

# Part 2
tibble_2019 <- tibble(
  location = c("fr", "co", "ne", "pr", "po"),
  time = "2019"
) %>%
  mutate(location = fct_relevel(location, c("fr", "co", "ne", "pr", "po")))

# Part 3
pike_master_flux_day_2019 <-
  left_join(pike_master_flux_day_2019, tibble_2019, by = "location")
```

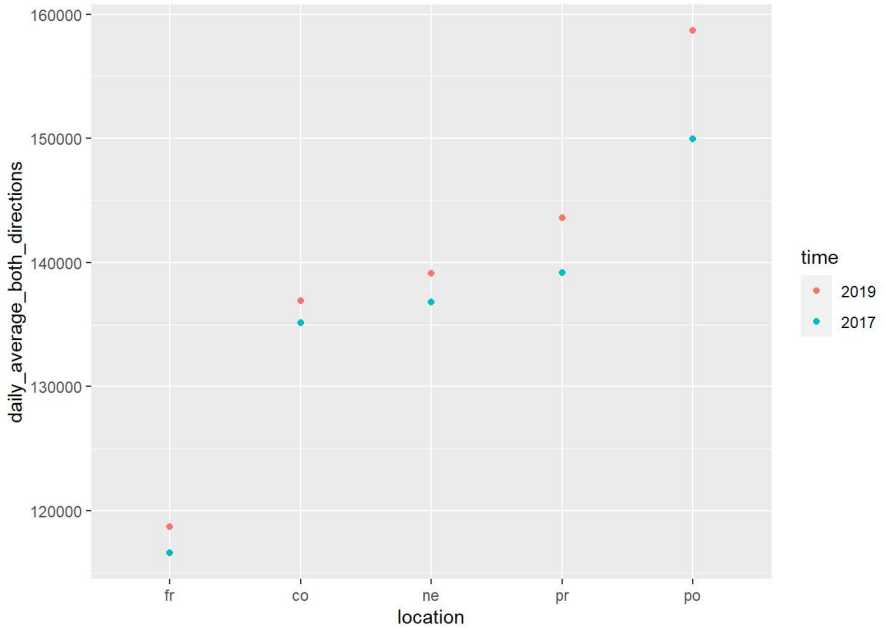
This next part used the same steps and reasoning as in the previous code chunk, replacing `2017` with `2019`.

```
# Part 1
pike_master_flux_2017_2019_toll <- bind_rows(pike_master_flux_day_2017, pike_master_flux_day_2019)

# Part 2
pike_master_flux_2017_2019_toll[is.na(pike_master_flux_2017_2019_toll)] <- 0

# Part 3
pike_master_flux_2017_2019_toll <- pike_master_flux_2017_2019_toll %>%
  mutate(year_total = pre_year_total + post_year_total) %>%
  mutate(daily_average_both_directions = year_total / n) %>%
  mutate(time = as.factor(time)) %>%
  select(location, n, year_total, time, daily_average_both_directions)
```

This explanation is based on what I discussed in the Supplemental Materials section. **Part 1:** Since both `pike_master_flux_day_2017` and `pike_master_flux_day_2019` both have the same column names, this `bind_rows()` will essentially stack the two tibbles on top of each other, resulting in an easy to use, tidy data set. **Part 2:** From the previous part, observations made before tolls do not have observations made afterwards. This leaves a bunch of `NA` values. Converting the `NA` values to `0` and then adding that to the total, while keeping `time` as `2017` or `2018` will allow me to plot things nice and neat. **Part 3:** Adding `0` (used to be `NA`) to a value will not change the value (It turns out learning the `Identity Property of Addition` in elementary school was super helpful). This is an attempt to normalize things because although I included all available data for `2017` and `2019`, that does not mean that there are the same number of observations per year.

Plotting Results

```
pike_master_flux_2017_2019_toll %>%
  mutate(time = fct_relevel(time, c("2019", "2017"))) %>%
  ggplot(aes(x = location, y = daily_average_both_directions, color = time)) +
  geom_point()
```



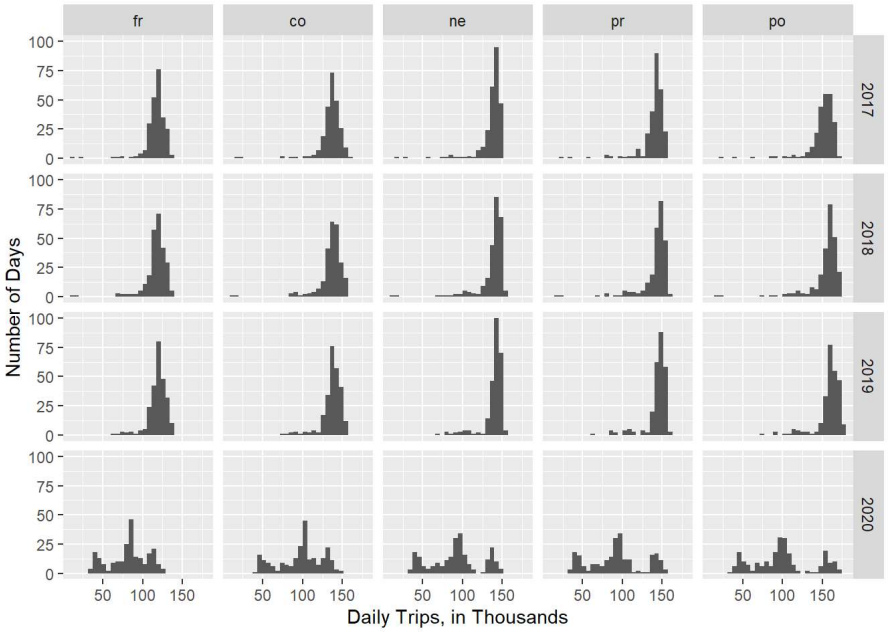
Although there was an increase in the `daily_average_both_directions` across all locations, I cannot say for sure it was because of the switch to electronic tolls, or if there are more drivers on the roads. Perhaps this would be a question for the modeling part of the course where I can try to see how much of the increase is accounted for by typical annual increase, and what portion of the increase is not accounted for, which might be electronic tolls. One thing that is screaming at me is the amount of daily trips made at the `po` location, or Post Allston Interchange. This portion of Mass Pike is used commonly to get to Logan International Airport. (An aside: you can only get to the airport by tunnel under the Boston Harbor when coming from and returning to Boston. You can reach it by land from the North Shore area, but that is not a practical option for a majority of the state. These tunnels were part of the Big Dig (https://www.mass.gov/info-details/the-big-dig-project-background), intending to connect parts of Boston via stretches of tunnels over several miles, also trying to relieve traffic.) The average number of cars per day traveling through the intersection is nearly `160000` cars per day! My simpleton mind cannot comprehend that, so I investigated that further in the Supplemental Materials section. The conclusion is that my eyes were not deceiving me, and there were 105 days with cars traveling West Bound and 185 days with cars traveling East Bound that exceeded 75000 in `2017`. In `2019`, there were 219 with cars traveling West Bound and 237 days with cars traveling East Bound exceeding 75000 cars per day (The reason: EB + WB or 75000 + 75000 = 150000, which is what I had a hard time understanding at first). I guess that could contribute to the number of potholes on the roads. It is hard to imagine over 150000 of anything!

Distribution of Total Cars per Day for Locations on the Mass Pike

```
pike_master_flux_day_distribution <- pike_master_flux_day %>%
  mutate(total_trips = Total_EB + Total_WB) %>%
  mutate(total_trips_in_1000s = total_trips / 1000) %>%
  mutate(year = (year(full_date)))
```

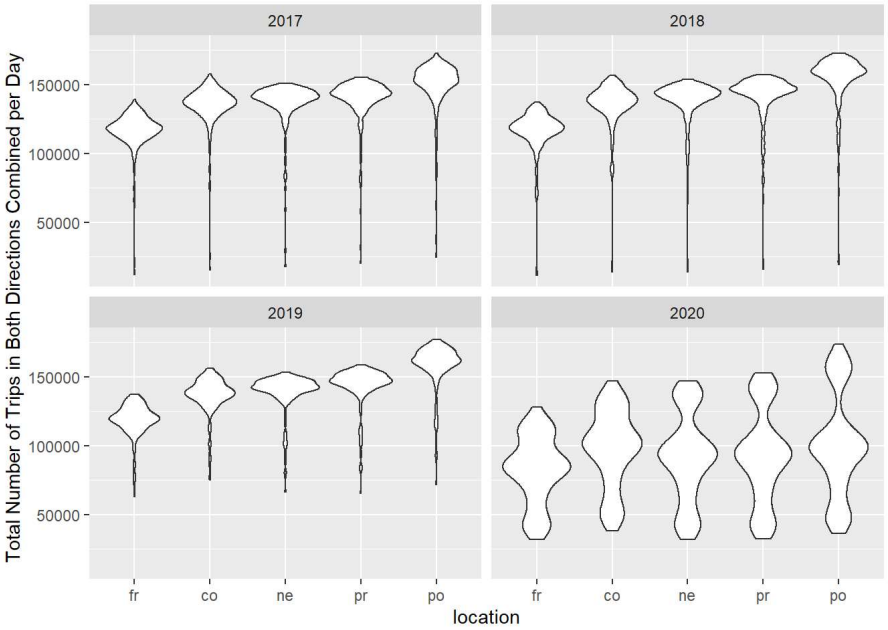
This part sets up the plots by redefining `total_trips` and `year` column because working with lubridate within `ggplot` gets tricky.

```
ggplot(pike_master_flux_day_distribution, aes(x = total_trips_in_1000s)) +
  geom_histogram() +
  facet_grid(year~location) +
  labs(x = "Daily Trips, in Thousands", y = "Number of Days")
```



Looking at this, I can see the bins shifting to the right (more trips on average per day). As mentioned earlier, `2020` is off doing its own thing, not following any sort of pattern.

```
ggplot(pike_master_flux_day_distribution, aes(x = location, y = total_trips)) +
  geom_violin() +
  facet_wrap(~year, ncol = 2) +
  labs(y = "Total Number of Trips in Both Directions Combined per Day")
```



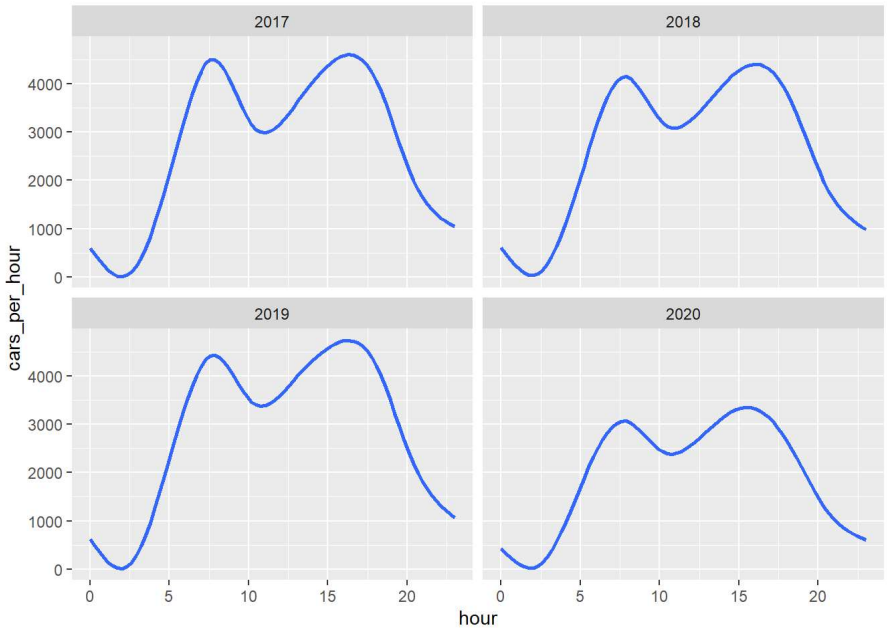
For the heck of it, I made violin plots. `2020` is quite different because of the coronavirus pandemic due to traffic decreasing substantially after the State of Emergency, which is brought up in the next section, *Coronavirus Pandemic*. The tops of the violin plots for `2020` are at similar levels as the previous years.

Coronavirus Pandemic

2020 has been a strange year. I decided to make a few panels showing the amount of traffic in the month of March, April, and May when a majority of the state shut down or reduced activity. The rest of the locations of the Mass Pike are located in the Supplemental Materials section.

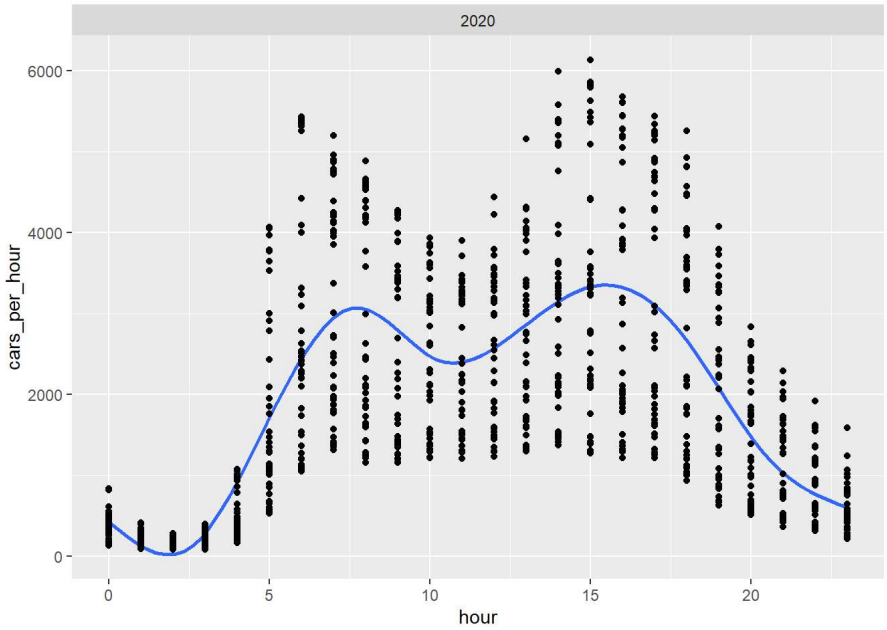
Cochituate in March

```
pike_master %>%
  filter(day_of_week != "Saturday" & day_of_week != "Sunday",
         location == "co",
         month(full_date) == 3
         ) %>%
  ggplot(aes(x = hour, y = cars_per_hour)) +
  geom_smooth(se = FALSE) +
  facet_wrap(~year(full_date), ncol = 2)
```



I was a little curious about March in 2020 because there was a COVID-19 State of Emergency (https://www.mass.gov/info-details/covid-19-state-of-emergency) issued in the middle of the month on March 18th, encouraging people to work from home.

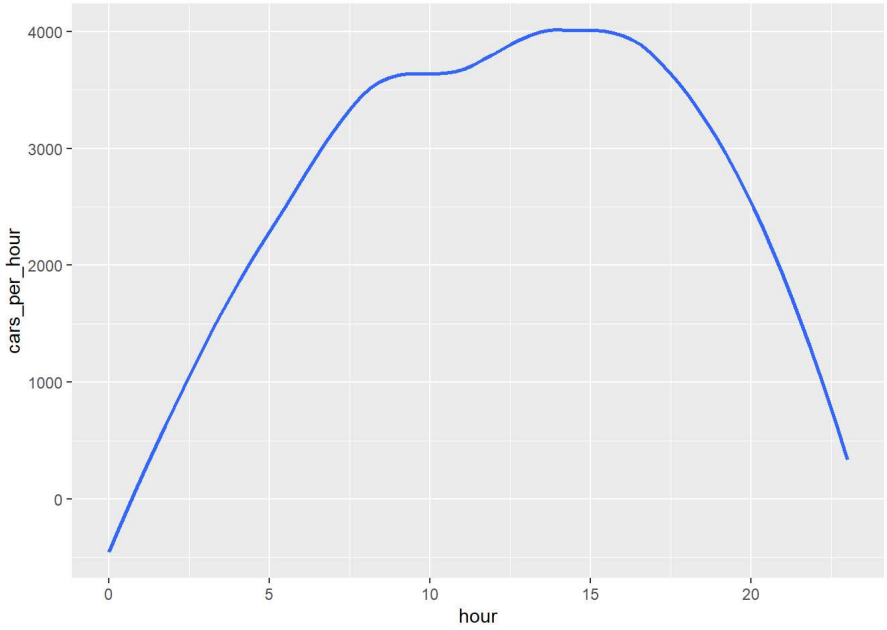
```
pike_master %>%
  filter(day_of_week != "Saturday" & day_of_week != "Sunday",
         location == "co",
         month(full_date) == 3,
         year(full_date) == 2020
         ) %>%
  ggplot(aes(x = hour, y = cars_per_hour)) +
  geom_smooth(se = FALSE) +
  geom_point() +
  facet_wrap(~year(full_date), ncol = 2)
```



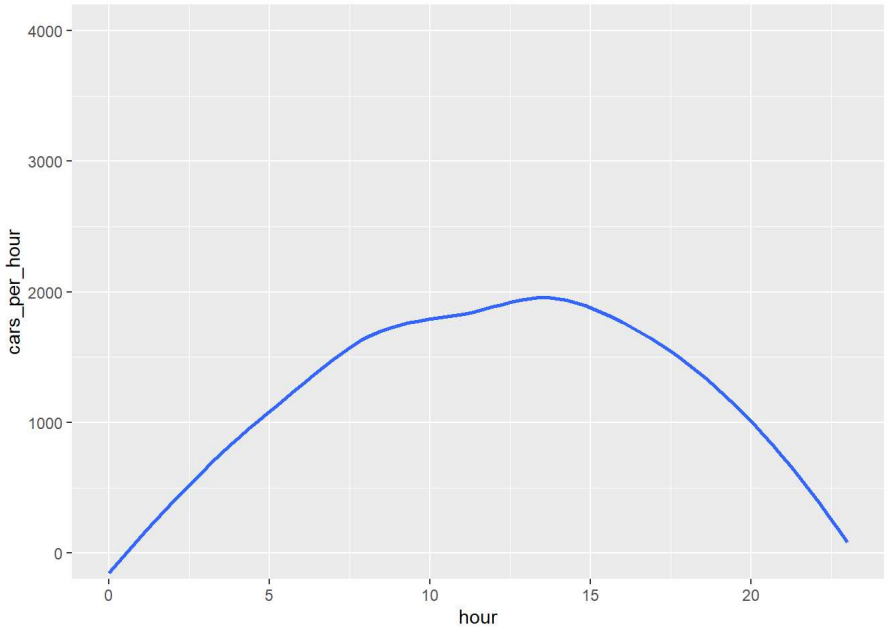
There is a fairly wide spread in the month of March. When I split it up based on before and after the State of Emergency was issued, there looks like there is a difference.

Before and After State of Emergency Issued

```
# Before
pike_master %>%
  filter(day_of_week != "Saturday" & day_of_week != "Sunday",
         location == "co",
         full_date > ymd(20200301) & full_date < ymd(20200318)
  ) %>%
  ggplot(aes(x = hour, y = cars_per_hour)) +
  geom_smooth(se = FALSE)
```

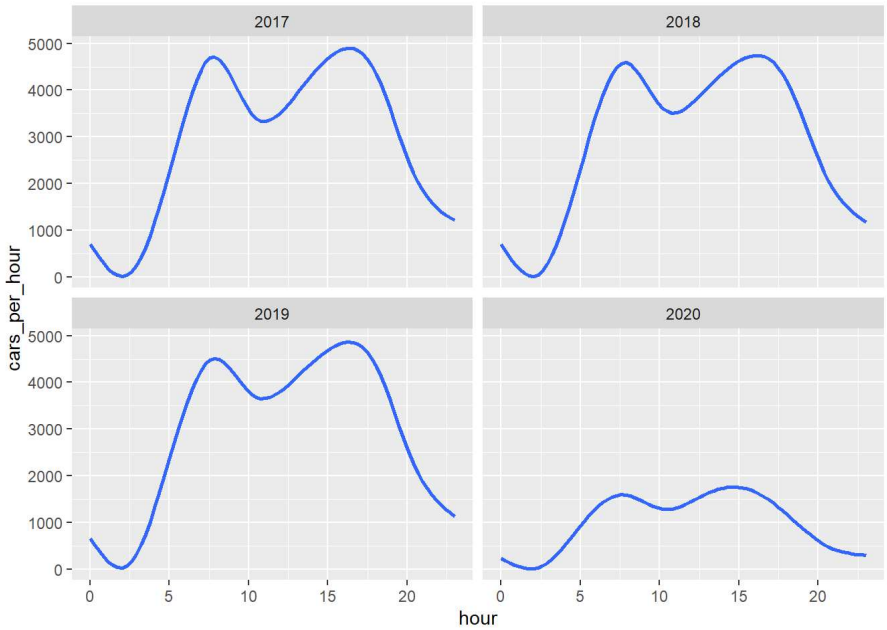



```
# After
pike_master %>%
  filter(day_of_week != "Saturday" & day_of_week != "Sunday",
         location == "co",
         full_date >= ymd(20200318) & full_date <= ymd(20200331)
  ) %>%
  ggplot(aes(x = hour, y = cars_per_hour)) +
  geom_smooth(se = FALSE) +
  coord_cartesian(ylim = c(0, 4000))
```



Cochituate in April

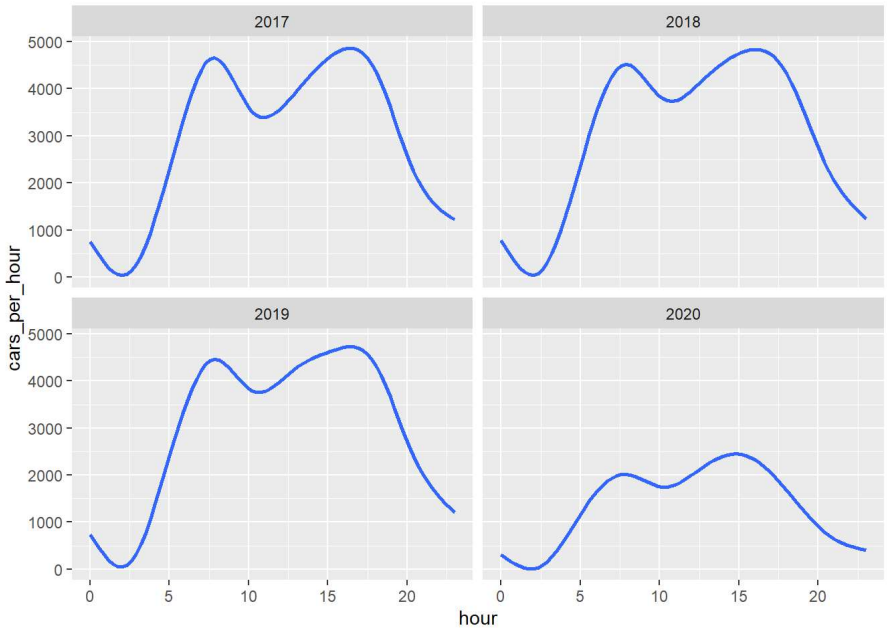
```
pike_master %>%
  filter(day_of_week != "Saturday" & day_of_week != "Sunday",
         location == "co",
         month(full_date) == 4
  ) %>%
  ggplot(aes(x = hour, y = cars_per_hour)) +
  geom_smooth(se = FALSE) +
  facet_wrap(~year(full_date), ncol = 2)
```



Interesting that even though April has much lower traffic, the two humps from rush hour are still there. Perhaps this is due to the amount of people who are essential workers or have jobs that needed to be in person.

Cochituate in May

```
pike_master %>%
  filter(day_of_week != "Saturday" & day_of_week != "Sunday",
       location == "co",
       month(full_date) == 5
  ) %>%
  ggplot(aes(x = hour, y = cars_per_hour)) +
  geom_smooth(se = FALSE) +
  facet_wrap(~year(full_date), ncol = 2)
```



Traffic starts to rebound a little bit, but still about half of the previous years.

Final Notes

The first thing that comes to mind is that I really learned a lot! I learned more about coding while learning more about traffic in Massachusetts. I had a fun time playing around with different types of graphs and faceting, and I am sure I will continue to get better at selecting graphs and learning how to display information in the clearest way possible as I progress through the classes for the Data Science minor. The dream would be to extend my analysis to as many other intersections in Massachusetts as possible. I would need to learn how to datascrape to download all of the Excel files for that and likely would need to invest in more memory. MassDOT has highlighted portions of Route 2 closer to Boston as being severely congested in the Report to the Governor 2019 (https://www.mass.gov/doc/congestion-in-the-commonwealth/download). I think it would be worthwhile for the state to purchase and install other traffic sensors near the intersection highlighted. That way, the state could prioritize resources to improve intersections and hopefully would eventually benefit commuters. According to INRIX (https://inrix.com/scorecard/), an analytics company that focuses on traffic, Boston is the worst city in the United States for traffic. They estimate that commuters in Boston lose on average 149 hours per year due to congestion. Obviously the state is failing to understand traffic and needs to prioritize public transportation to help commuters both inside and outside of Boston get to work.