Capstone - Final

Katie Wilson

June 4, 2018

INTRODUCTION:

New York City – the largest city in the United States. It has over 27,000 people per square mile which is the largest population density in the country. Brooklyn, on its own, would be listed as the fourth largest city in the US. With the constantly increasing population in the city that never sleeps, car accidents happen all over the city at all times of the day and night. Any sort of data analysis on the types of accidents, location, and time would make prevention for the police much more manageable. Certain areas of the city are clearly more populous and, therefore, harder for the police to narrow down where accidents are stemming from and how to address the cause. In this project, I will attempt to identify areas, days, and times throughout New York where most car accidents happen. The end goal of this research would be to provide the police, city planners, and eventually the public with up-to-date information for better decision making. These few pages of data manipulation, analysis, and results is meant to be a foundation for further research and development of the concept.

```
library(tidyr)
library(ggplot2)
library(ggthemes)
library(dplyr)
library(readr)
library(maptools)
library(lubridate)
library(RSocrata)
library(scales)
library(stats)
library(viridis)
library(forcats)
```

DATA:

This report is mainly based on one dataset from the NYPD. However, the process to choose this specific information was not as simple as that. There is quite a lot of data out there that addresses this topic. Originally, I had looked at a dataset on kaggle (https://www.kaggle.com/nypd/vehicle-collisions) but decided against it due to the lack of location data listed. The city of New York has a similar dataset (https://nycopendata.socrata.com/browse?q=traffic) as well as the Department of Transportation (see Appendix 1). There is information regarding speed limits (see Appendix 2), precipitation, and population density as well. One dataset on the nyc.gov website I looked at is no longer available (http://www.nyc.gov/html/nypd/html/traffic_reports/traffic_report_archive_2013.shtml). All these datasets could be used to a different degree to accomplish a similar task, but the NYPD dataset was the most complete and consistent information I found. Some of the other compilations would have required quite a lot more manipulation or joining with other data to give as much complete information as the NYPD set has. I had originally started working on this project using the DOT dataset and did quite a bit of manipulation and wrangling before deciding that the information it provided wasn't consistent, wasn't formatted well, and only had a few thousand rows of data (again, see Appendix 1). One dataset I found was in an excel format but had been formatted in a way that was nice to look at, but not nice to work with. There were multiple data points in a cell and would have taken more wrangling and correcting than was ultimately worth the time. There was also information on traffic volume counts (https://data.cityofnewyork.us/NYC-BigApps/Traffic-Volume-Counts-2012-2013-/p424-amsu)

which I wanted to try to use to normalize the data I found, but it turned out to be a bigger task than what was worth it at the time. There was another website that included information on the Uber, Lyft, and taxi data in the city (http://toddwschneider.com/posts/taxi-uber-lyft-usage-new-york-city/). Lastly, there was also a similar project I found that was not incorporated into this report but is interesting to note (https://ny.curbed.com/2017/11/7/16618956/most-dangerous-pedestrian-intersection-nyc).

After wading through a lot of these websites and datasets, I settled on the one from the NYPD for many reasons. The first advantage was the sheer volume of records that are included. The information goes back to 2012 and is concisely organized. There was limited wrangling that was necessary because it already included one observation per line. Another advantage of using the NYPD data is that it is a constantly updating set of information. As you'll see, I used a specific subset of this data in this project to keep the results consistent. Even with that subset, the data contained over a million lines of observations. Although this paper is only including that portion of the data, the goal in further development would be to include everything and to have the most up to date information available. Through my work with the DOT dataset, there were a few things I had wanted to work with that were not included in that dataset. These were such things as contributing factors, vehicle type, a specific time and date for the accidents, and the ability to sort by borough. All these things were included in the NYPD data.

The NYPD data is online on the City of New York open data website: https://data.cityofnewyork.us/Public-Safety/NYPD-Motor-Vehicle-Collisions/h9gi-nx95 To input this into R, I needed to utilize an API key, save it locally as an RDS file, and then read it into R. Using the if statement below allows the program to search for the file first and if it does not exist, pulls the dataset from the website. However, if it does exist, it loads the current file. This is done to save time and memory.

```
if(file.exists("data-raw/nypd_motor_vehicle_collisions.rds")){
  ny_df <- readRDS("data-raw/nypd_motor_vehicle_collisions.rds")
} else{
  ny_uri <- "https://data.cityofnewyork.us/resource/qiz3-axqb.csv"
  ny_api <- validateUrl(ny_uri, Sys.getenv('API_NYPD_KEY'))
  ny_df <- read.socrata(ny_api) %>%
    as_data_frame()
  saveRDS(object = ny_df, file = "data-raw/nypd_motor_vehicle_collisions.rds")
  rm(ny_api, ny_uri)
}
```

For this dataset, a couple small corrections were needed for the type of data that was listed in the original file. The date field wasn't being read by R as a date which is fixed using the lubridate package. There were also zeros listed in the latitude and longitude columns that need to be changed to NAs. Lastly, there were blanks in the boroughs column that also need to be change to NAs. I also included a subset to encompass January of 2013 through December of 2017.

```
#Date edit
ny_df$date_time <- as.POSIXct(paste(ny_df$date, ny_df$time), format = "%Y-%m-%d %H:%M")
ny_df$date <- as.Date(ny_df$date, format = "%Y-%m-%d")
#Latitude and Longitude
ny_df$latitude[ny_df$latitude == 0] <- NA
ny_df$longitude[ny_df$longitude == 0] <- NA
#Boroughs
ny_df$borough[ny_df$borough == ""] <- NA
#Subset the data
ny_df <- subset(ny_df, date > "2012-12-31" & date < "2018-01-01")
```

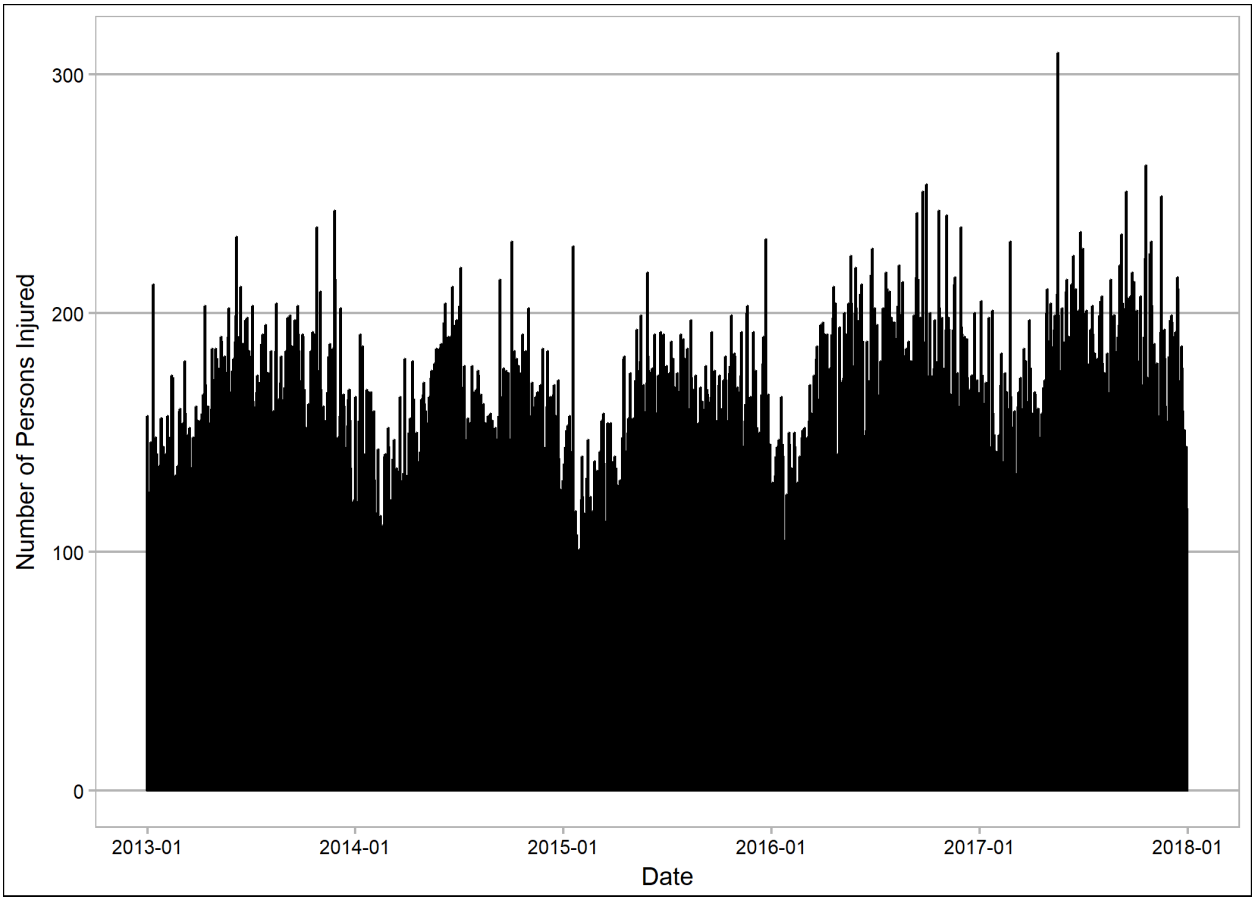
EXPLORATORY DATA ANALYSIS:

Now that the data has been cleaned and is in a consistent format, I wanted to see a few summaries of the file including preliminary graphs to identify any trends or potential outliers. These graphs are rather rough and are specifically for the purpose of identifying any further updates that may need to happen or any general trends that can be deduced from the current state of the data. This will be done, first, by assigning a column consisting only of the month and summarizing the data based on the month (to see the minimum, maximum, and the quartiles of the data). Second, two graphs are created, one based on injuries by month, and the other on fatalities by month.

```
#Date Summary by Month
ny_df$month <- month(ny_df$date_time)
summary(month(ny_df$date_time))
```

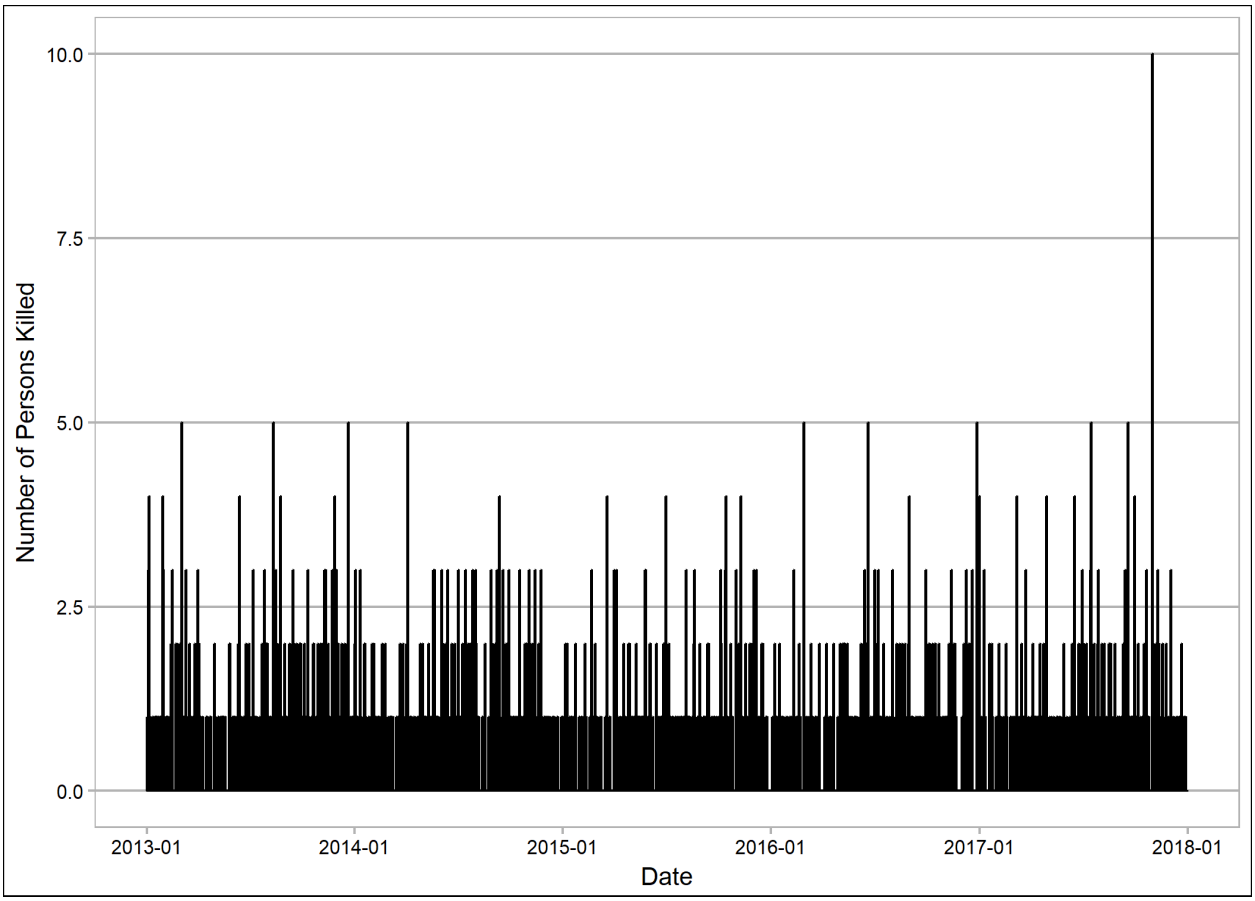
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    1.00    4.00    7.00    6.63   10.00   12.00       9
```

```
#Persons Injured by Year
ggplot(data = ny_df, aes(x = date, y = number_of_persons_injured)) +
  stat_summary(fun.y = sum, geom = "bar", colour = "black") +
  scale_x_date(labels = date_format("%Y-%m"),
    date_breaks = "1 year") +
  theme_calc() +
  labs(x = "Date", y = "Number of Persons Injured")
```




```
#Persons Killed by Year
ggplot(data = ny_df, aes(x = date, y = number_of_persons_killed)) +
```

```
  stat_summary(fun.y = sum, geom = "bar", colour = "black") +
  scale_x_date(labels = date_format("%Y-%m"),
    date_breaks = "1 year") +
  theme_calc() +
  labs(x = "Date", y = "Number of Persons Killed")
```

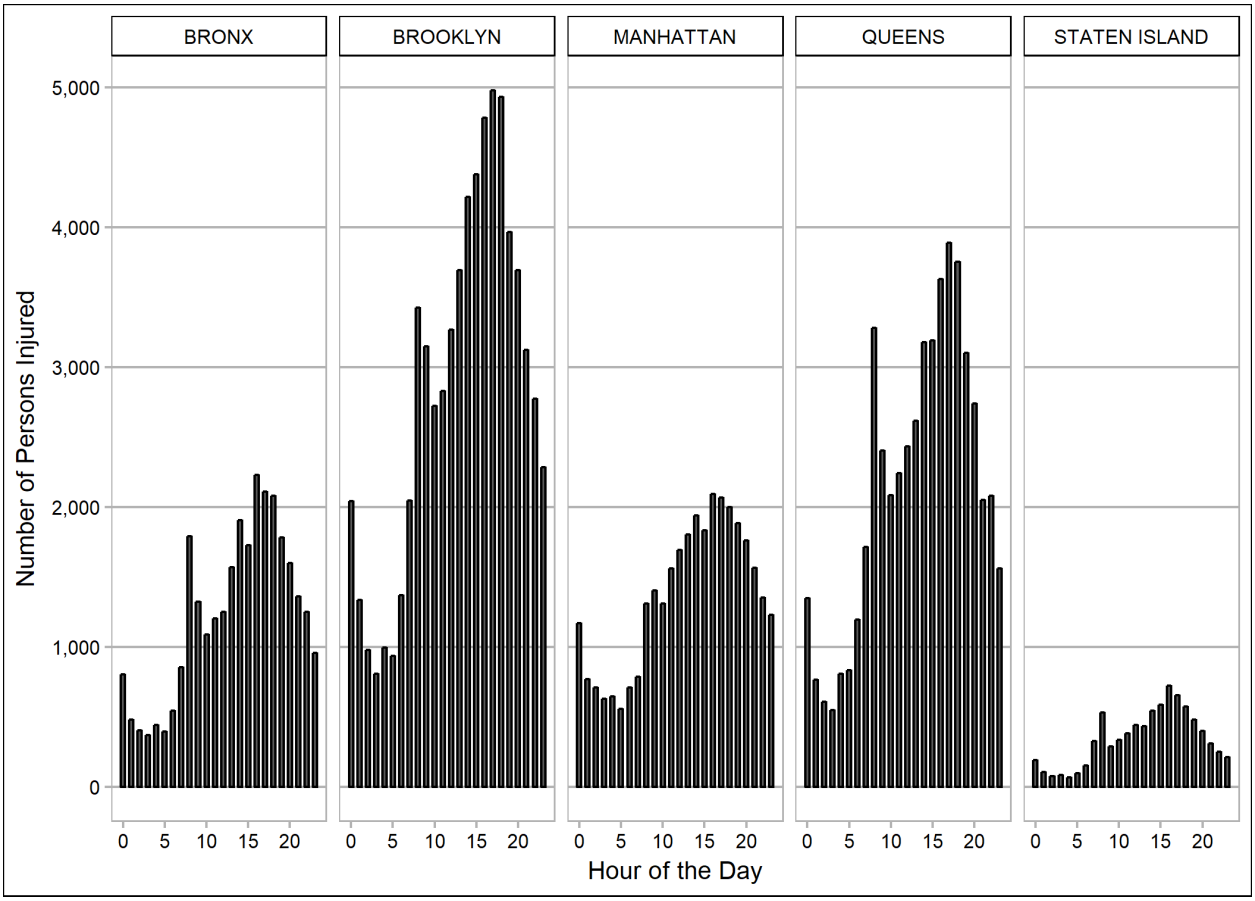


The data overall seems to be distributed across the year rather evenly. However, the mean is slightly more towards June suggesting there are more accidents in the second half of the year than the first. There also seems to be regular dips and rises throughout each year, which will be explored more in the coming sections.

The dataset also includes the time of the day the accidents occurred. Breaking out the time and hour from the date column allows us to view the data by the time of the day. The following breakdowns show the sum of people injured and the sum of fatalities by each hour of the day faceted by borough and not including any data that has an NA as the borough.

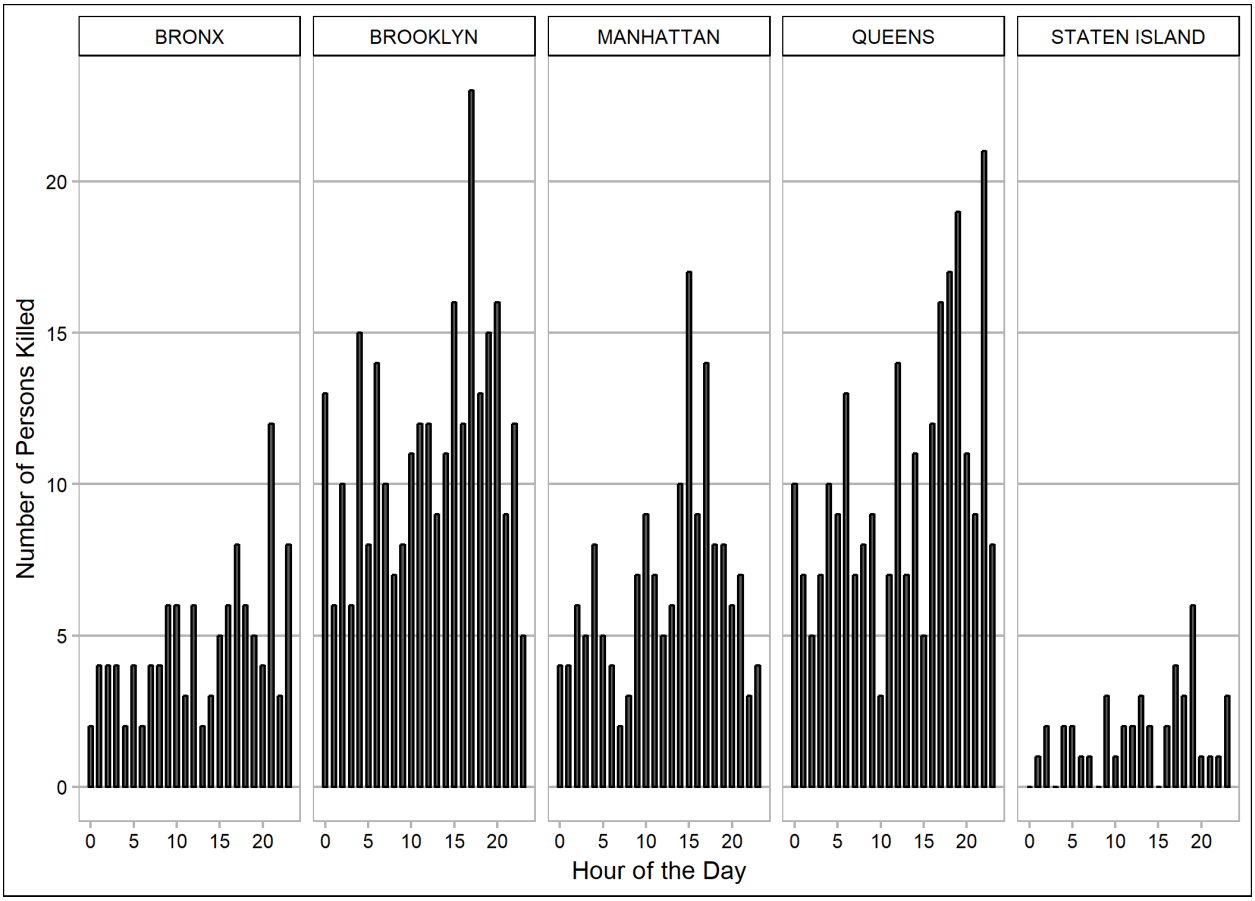
```
ny_df$hour <- hour(ny_df$date_time)
#Based on injuries
ny_df %>% filter(!is.na(borough)) %>%
  ggplot(aes(hour, number_of_persons_injured)) +
  stat_summary(fun.y = sum, geom = "bar", width = 0.5, colour = "black") +
  scale_y_continuous(labels = comma) +
  facet_grid(. ~ borough) +
  theme_calc() +
  labs(x = "Hour of the Day", y = "Number of Persons Injured")
```

```
## Warning: Removed 5 rows containing non-finite values (stat_summary).
```




```
#Based on fatalities
ny_df %>% filter(!is.na(borough)) %>%
  ggplot(aes(hour, number_of_persons_killed)) +
  stat_summary(fun.y = sum, geom = "bar", width = 0.5, colour = "black") +
  scale_y_continuous(labels = comma) +
  facet_grid(. ~ borough) +
  theme_calc() +
  labs(x = "Hour of the Day", y = "Number of Persons Killed")
```

```
## Warning: Removed 5 rows containing non-finite values (stat_summary).
```



Clearly, from these graphs, Brooklyn has the most accidents of the five boroughs which may be due to the size of that borough, but will be explored in more detail, nonetheless. As expected, Staten Island has the least amount of injuries since it is the smallest of the five. It can also be seen that the number of injuries occurring throughout the day seems to peak during after work rush hour and dip during the early hours of the morning in each of the boroughs.

There are a few other defining characteristics in this dataset including the zip code/borough, the type of vehicle, and the contributing factor. A good preliminary way to summarize these other factors is through tables. These three dimensions are put into tables below, ordered lowest to highest. Only the top ten are included for the vehicle and factor tables since, usually, the most important contributors will be the top 5-10 categories depending on the distribution. There are a few descriptors which are generally unhelpful for any sort of analysis such as the "Unknown", "Other", and blank categories within the vehicle type category and the "Unspecified" and "Other Vehicular" groups in the contributing factor category. As these do not give us any helpful trending information, they are filtered out and not included in any further work. Once this is accomplished, the tables are converted to dataframes so they can be graphed in ggplot.

```
#By borough
boroughTable <- table(ny_df$borough)
boroughTable <- boroughTable[order(boroughTable)]
boroughDF <- data_frame(values = as.numeric(boroughTable),names = names(boroughTable)) %>%
  arrange(values) %>%
  mutate(names = factor(names, levels = unique(names)))
colnames(boroughDF) <- c("ValueCount", "Borough")
boroughDF
```

```
## # A tibble: 5 x 2
##   ValueCount Borough
```

```
##        <dbl> <fct>
## 1      34187 STATEN ISLAND
## 2     103602 BRONX
## 3     193625 MANHATTAN
## 4     202755 QUEENS
## 5     237965 BROOKLYN
```

```
#By primary vehicle type
#Convert all the factors to lower case
ny_df$vehicle_type_code1 <- tolower(ny_df$vehicle_type_code1)
#Filter out the non-descriptive titles and assign to a variable
vehicleTable <- table(ny_df$vehicle_type_code1[which(!(
  ny_df$vehicle_type_code1 == "other" | ny_df$vehicle_type_code1 == "unknown" |
    ny_df$vehicle_type_code1 == ""))])
#Order lowest to highest
vehicleTable <- vehicleTable[order(vehicleTable)]
#Include the largest 10 factors
vehicleTable <- tail(vehicleTable, 10)
#Convert to a dataframe and tell R how to read in the variables
vehicleDF <- data_frame(values = as.numeric(vehicleTable),
                        names = names(vehicleTable)) %>%
  arrange(values) %>%
  mutate(names = factor(names, levels = unique(names)))
colnames(vehicleDF) <- c("ValueCount", "VehicleType")
vehicleDF
```

```
## # A tibble: 10 x 2
##    ValueCount VehicleType
##         <dbl> <fct>
##  1       6029 motorcycle
##  2       8419 livery vehicle
##  3      12369 bus
##  4      12464 large com veh(6 or more tires)
##  5      13222 "small com veh(4 tires) "
##  6      19963 pick-up truck
##  7      23638 van
##  8      42085 taxi
##  9     265158 sport utility / station wagon
## 10     620004 passenger vehicle
```

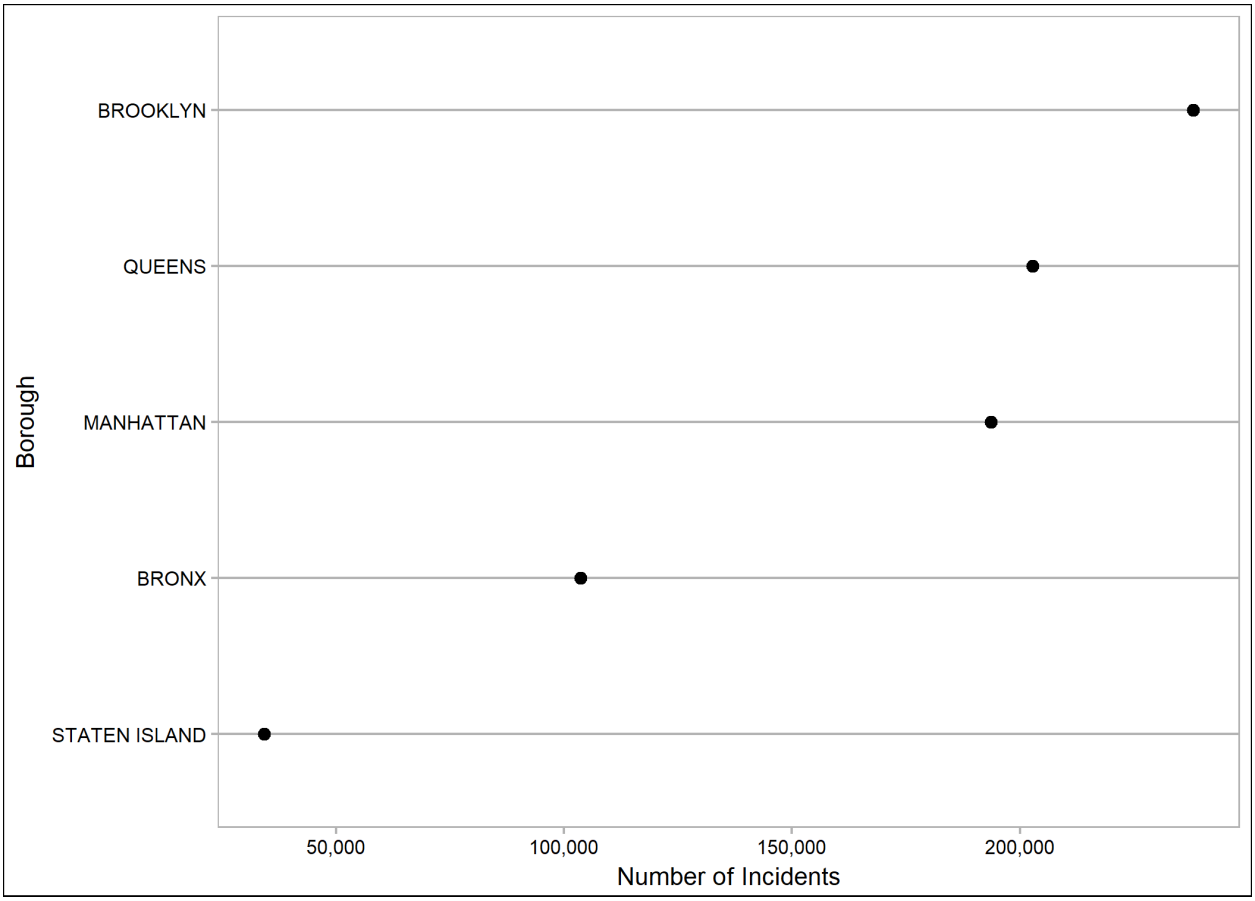
```
#By primary contributing factor
#Same process as above
ny_df$contributing_factor_vehicle_1 <- tolower(ny_df$contributing_factor_vehicle_1)
factorTable <- table(ny_df$contributing_factor_vehicle_1[which(!(
  ny_df$contributing_factor_vehicle_1 == "unspecified" |
    ny_df$contributing_factor_vehicle_1 == "other vehicular"))])
factorTable <- factorTable[order(factorTable)]
factorTable <- tail(factorTable, 10)
factorDF <- data_frame(values = as.numeric(factorTable), names = names(factorTable)) %>%
  arrange(values) %>%
  mutate(names = factor(names, levels = unique(names)))
colnames(factorDF) <- c("ValueCount", "ContributingFactor")
factorDF
```

```
## # A tibble: 10 x 2
##    ValueCount ContributingFactor
```

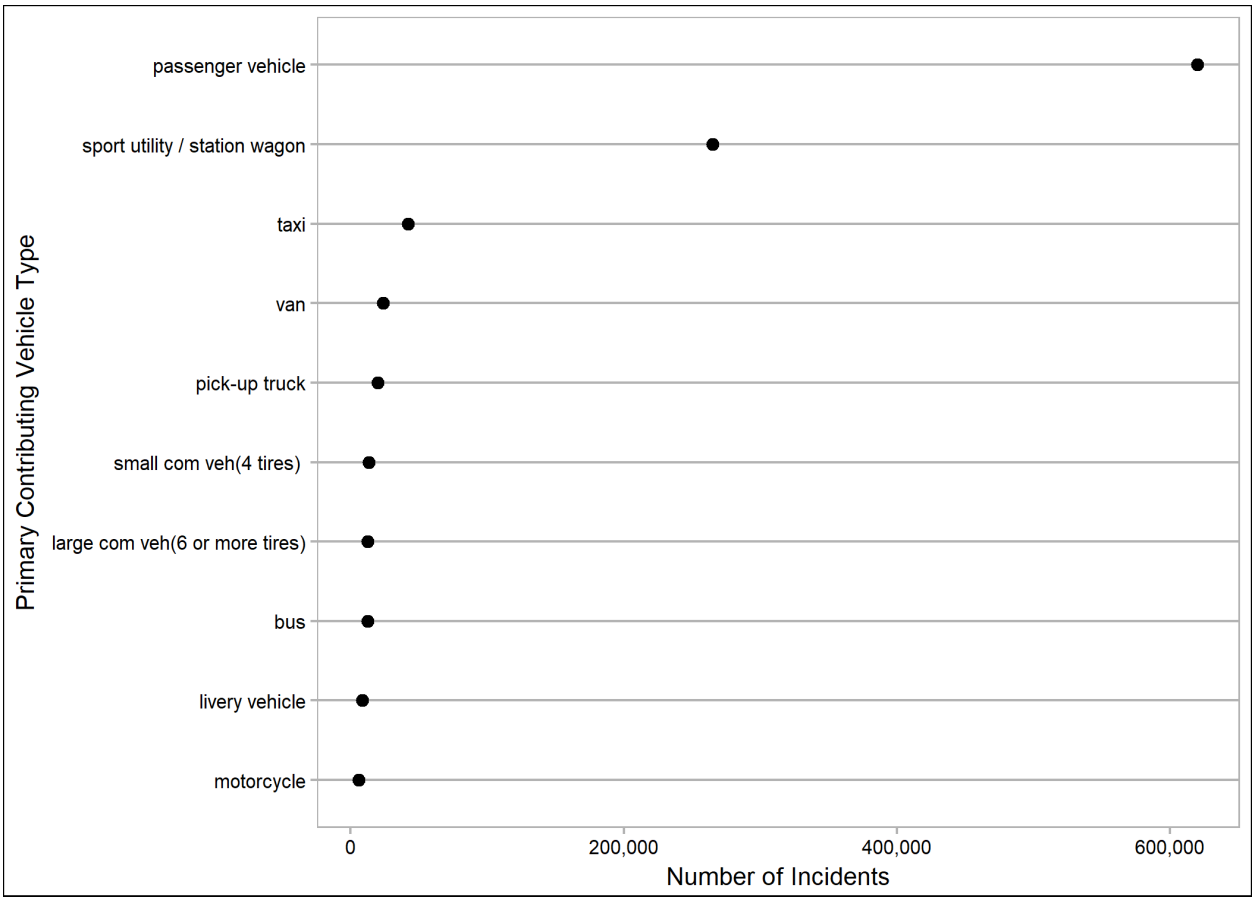
```
##          <dbl> <fct>
##  1      14245 driver inexperience
##  2      14321 passing or lane usage improper
##  3      14927 traffic control disregarded
##  4      18086 lost consciousness
##  5      26866 turning improperly
##  6      30450 following too closely
##  7      35047 backing unsafely
##  8      43472 fatigued/drowsy
##  9      52520 failure to yield right-of-way
## 10     163381 driver inattention/distraction
```

To see these tables in a simple graphical representation, see below.

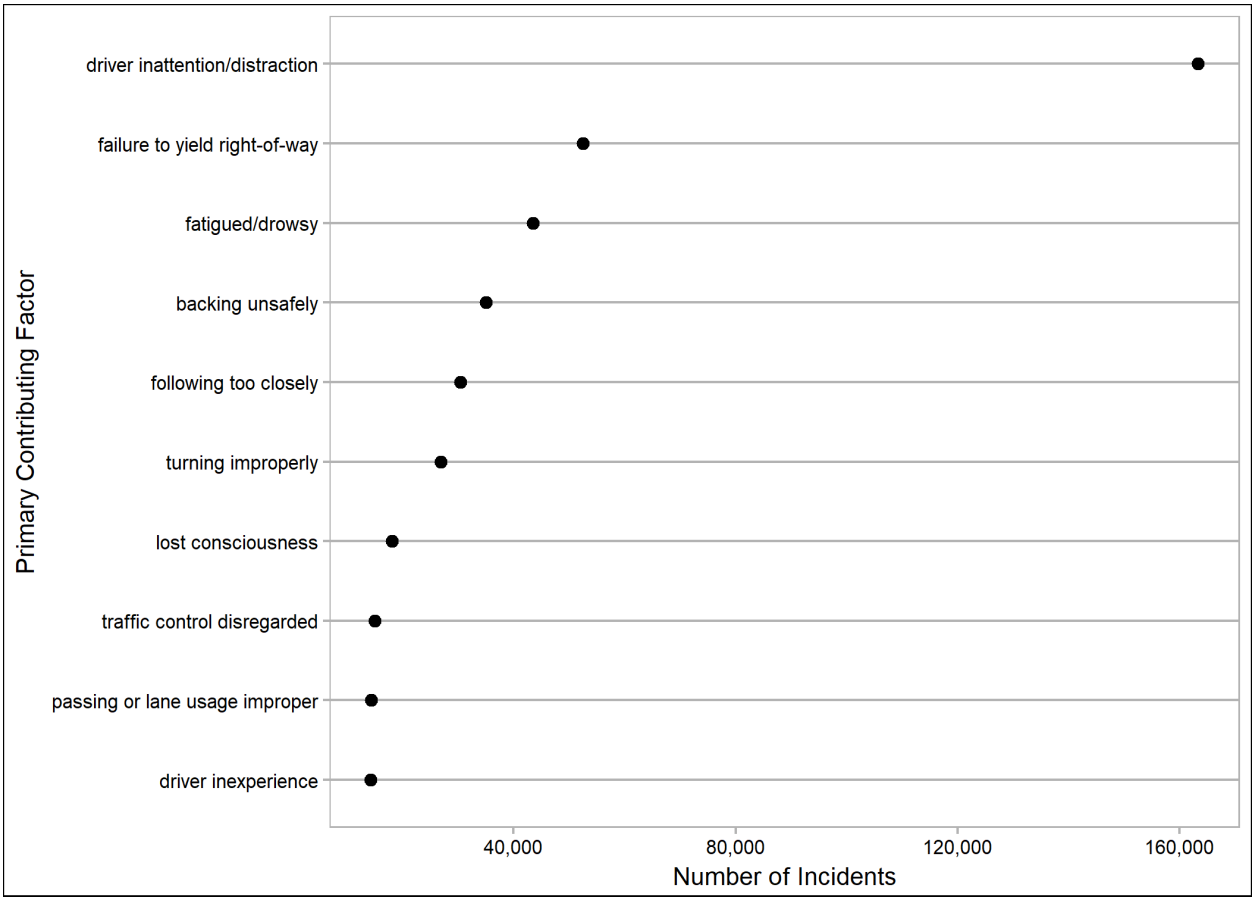
```
boroughDF %>%
  ggplot(aes(x = ValueCount, y = Borough)) +
  geom_point(size = 2) +
  scale_x_continuous(labels = comma) +
  theme_calc() +
  labs(x = "Number of Incidents")
```




```
vehicleDF %>%
  ggplot(aes(x = ValueCount, y = VehicleType)) +
  geom_point(size = 2) +
  scale_x_continuous(labels = comma) +
  theme_calc() +
  labs(x = "Number of Incidents", y = "Primary Contributing Vehicle Type")
```




```
factorDF %>%
  ggplot(aes(x = ValueCount, y = ContributingFactor)) +
  geom_point(size = 2) +
  scale_x_continuous(labels = comma) +
  theme_calc() +
  labs(x = "Number of Incidents", y = "Primary Contributing Factor")
```



Again, we can see that Brooklyn has the largest number of injuries of the boroughs. Also, it seems most accidents happen within passenger vehicles and are caused by driver inattention/distraction. The contributing factor data is particularly interesting as the top category has more than three times the number of injuries attributed to it as the second category of failing to yield to a right-of-way. A question that arises from these tables is, does Brooklyn on its own follow the same trend as the city as a whole? That is to say, are the top contributing factors and vehicles the same in Brooklyn as the city as a whole?

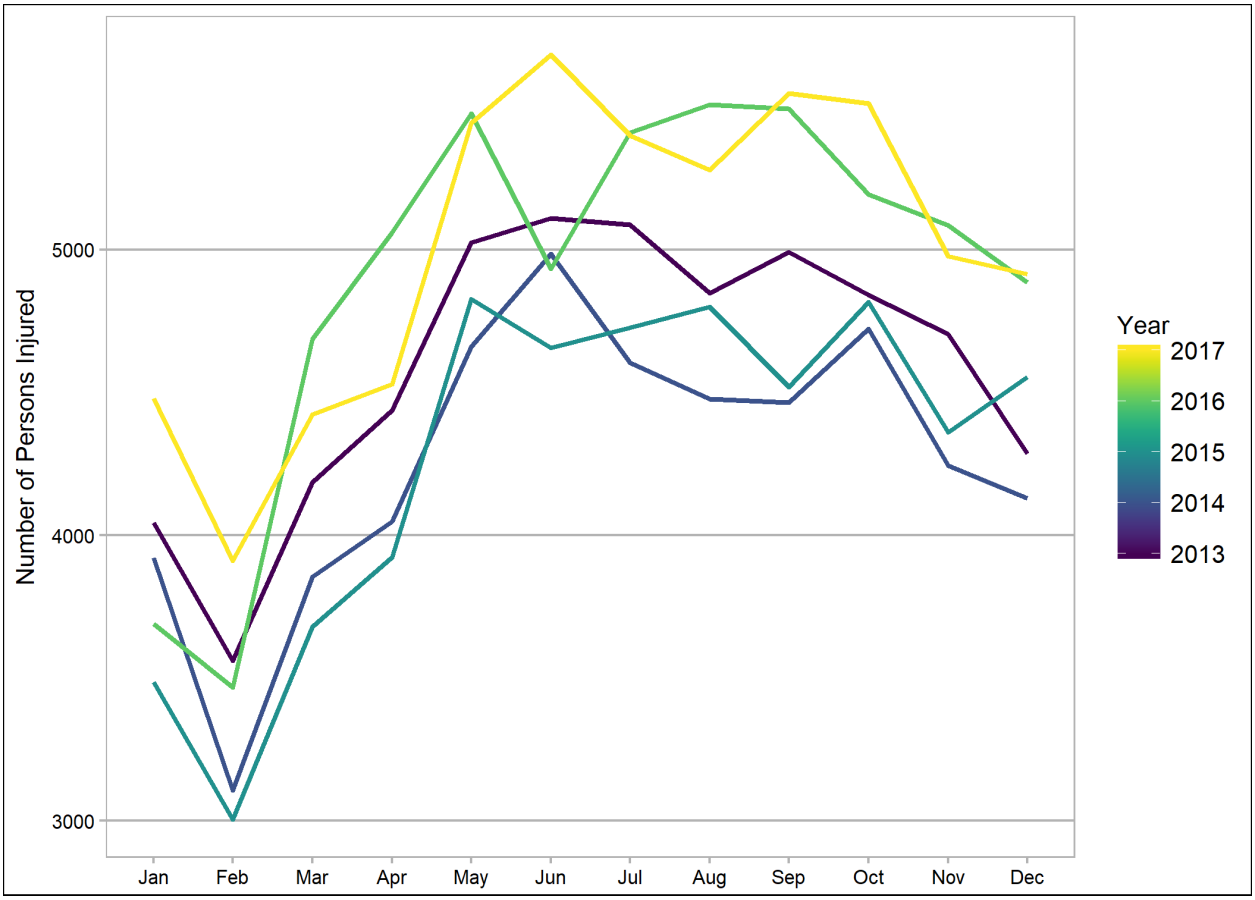
ANALYSIS:

Now that some preliminary trends have been identified and new questions are arising, digging further into the data visually, will help identify what trends can be actionable and where more research needs to be conducted. One of the most helpful indicators is related to time both throughout the year and across a day. For example, what hour of the day has the highest injury rate? The next few graphs will explore that question along with a few others.

First, the following graph shows the number of people injured each month across the five years included in this subset, with each of the years colored. Ideally, this will show if there are any months in the year more susceptible to traffic accidents.

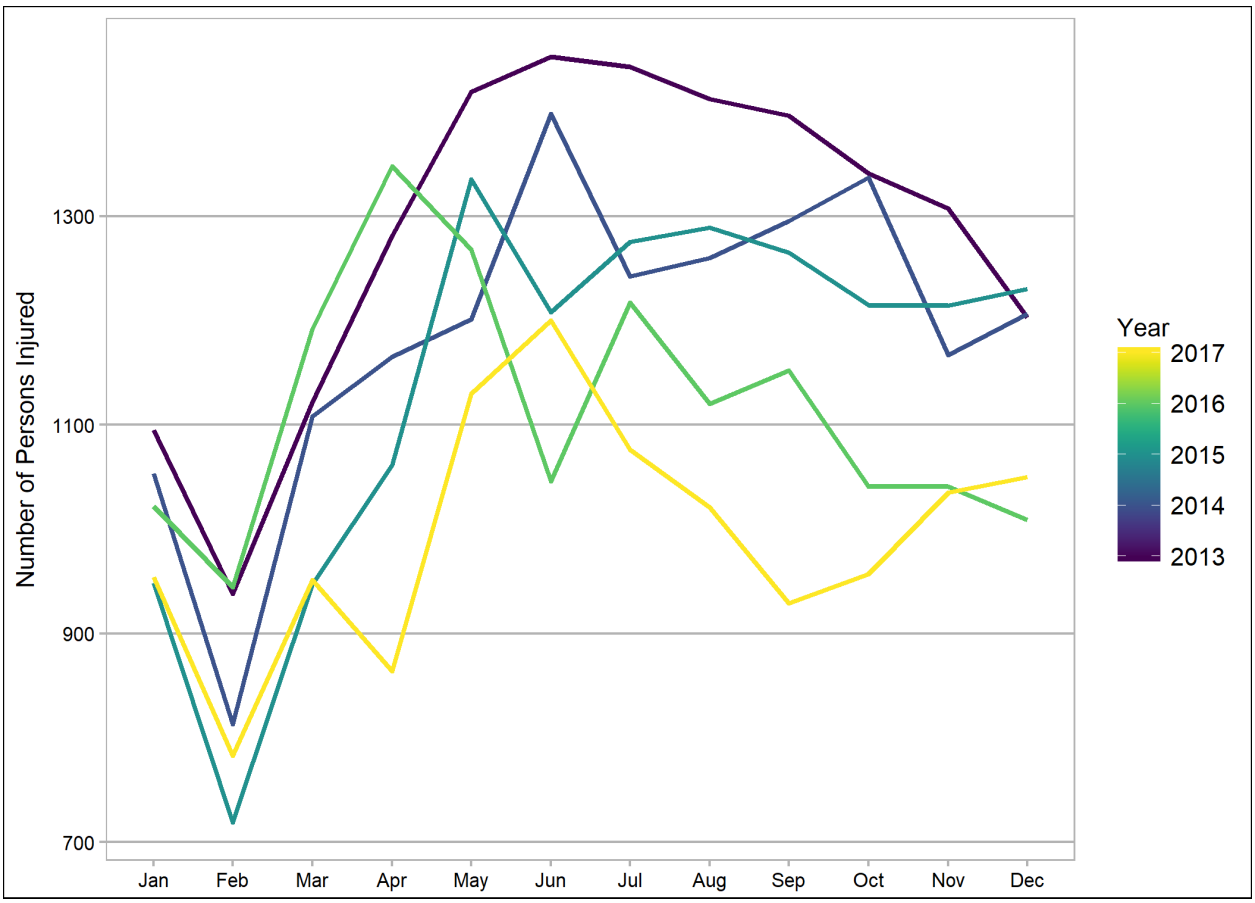
```
ny_df %>%
  ggplot(aes(month(date, label = TRUE, abbr = TRUE), number_of_persons_injured,
             group = factor(year(date)), colour = year(date))) +
  stat_summary(fun.y = sum, geom = "line", size = 1) +
  scale_color_viridis() +
  theme_calc() +
```

```
  theme(axis.title.x = element_blank()) +
  labs(y = "Number of Persons Injured", colour = "Year")
```



From here, we see the lowest month in each year is clearly February. This could be due to a number of different factors including the lack of tourists during that season. The highest month seems to change each year to anywhere between June in 2017 and 2014 and September in 2016. Again, the question arises, is this trend the same in each of the boroughs? Taking Brooklyn, specifically, this next graph shows the same layout but filtered to only include Brooklyn.

```
ny_df %>% filter(borough == "BROOKLYN") %>%
  ggplot(aes(month(date, label = TRUE, abbr = TRUE), number_of_persons_injured,
             group = factor(year(date)), colour = year(date))) +
  stat_summary(fun.y = sum, geom = "line", size = 1) +
  scale_color_viridis() +
  theme_calc() +
  theme(axis.title.x = element_blank()) +
  labs(y = "Number of Persons Injured", colour = "Year")
```

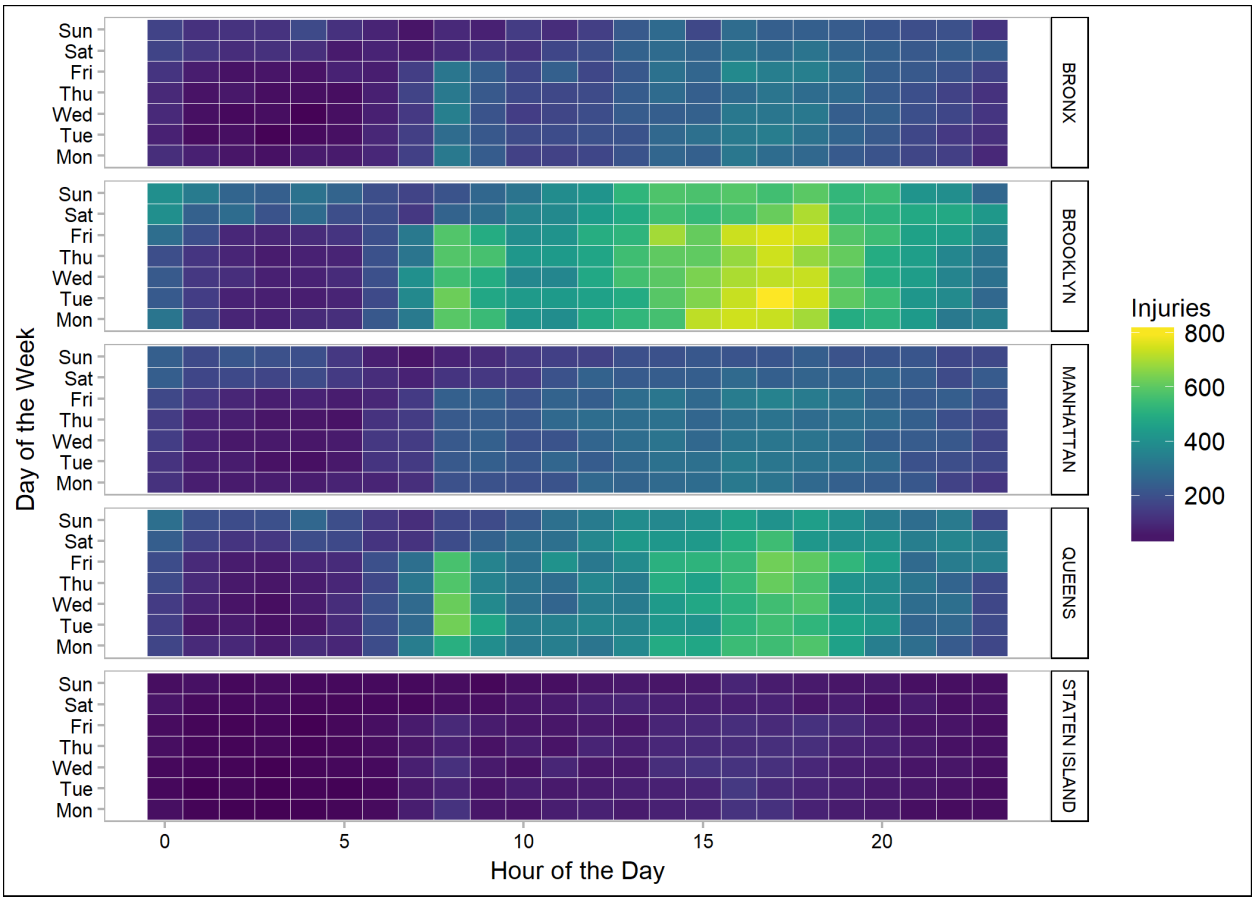


While the trend in Brooklyn seems to be the same with February as the lowest month, most of the other trends differ in this borough. While 2017 is the highest year of accidents in New York City as a whole, it is clearly the lowest in Brooklyn. The highest month in 2013, 2014, and 2017 is June while April is the highest of 2016 and May in 2015. As mentioned previously, the data offers many other aspects of each accident that could contribute to these trends. What is the largest contributing factor to these accidents and does that data have the same trend as the global dataset?

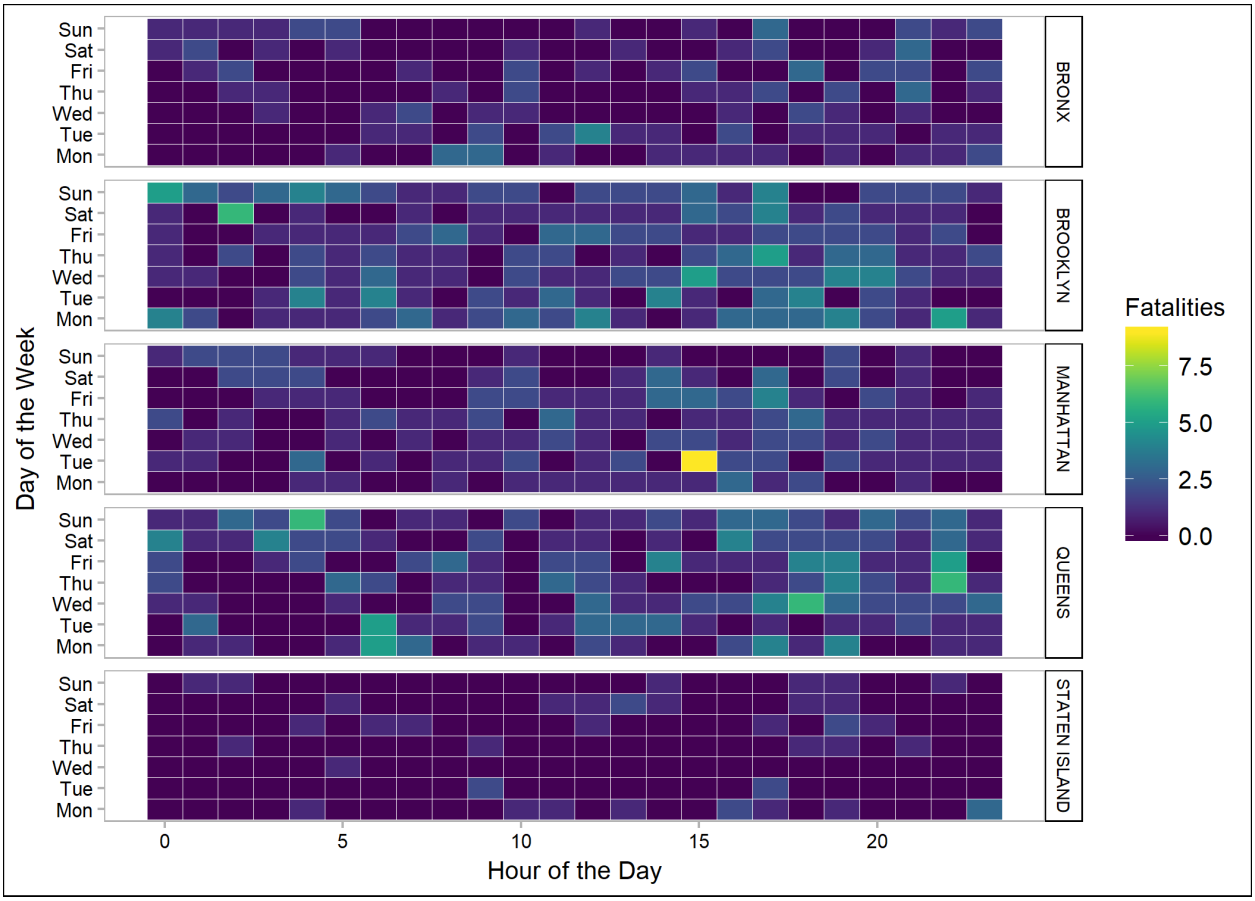
Keeping with the time analysis, another interesting aspect of this data is that it includes the time of day of each accident. A good way to represent this for the full data would be through a heat map. This will give insight into when the most common times are for accidents to occur and on what day of the week. To do this, the day of the week needs to be pulled out of the date using the lubridate package. That is done below in the first part of the code before a heat map is plotted for both the injury data as well as the fatalities. Again, these first heat maps are including the entire NYC dataset.

```
#Day of week interpreted from the date
ny_df$DOW = as.factor(wday(ny_df$date_time, label = TRUE, week_start = 1))
#Injuries heat map created
injuriesHeatMap <- ny_df %>%
  group_by(DOW, hour, borough) %>%
  summarize(Injuries = sum(number_of_persons_injured)) %>%
  na.omit() %>%
  ggplot(aes(x = hour, y = DOW, fill = Injuries)) +
  geom_tile(color = "white") +
  scale_fill_gradientn(colours = viridis(100))
#Filtered by borough
injuriesHeatMap +
```

```
  facet_grid(borough~.) +
  theme_calc() +
  theme(panel.grid.major = element_blank()) +
  labs(x = "Hour of the Day", y = "Day of the Week") +
  theme(strip.text.y = element_text(size = 7))
```

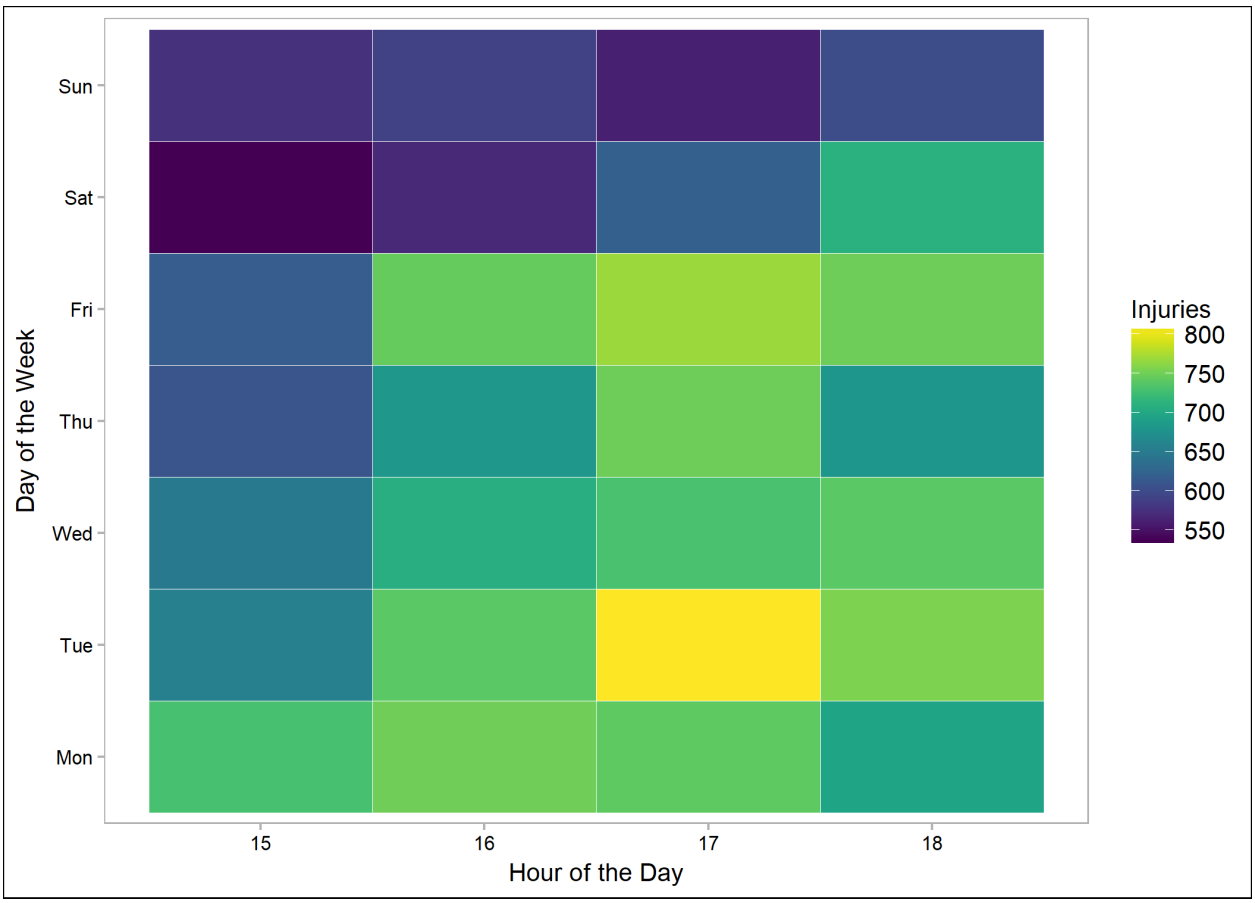



```
#Same process for the fatalities data
fatalitiesHeatMap <- ny_df %>%
  group_by(DOW, hour, borough) %>%
  summarize(Fatalities = sum(number_of_persons_killed)) %>%
  na.omit() %>%
  ggplot(aes(x = hour, y = DOW, fill = Fatalities)) +
  geom_tile(color = "white") +
  scale_fill_gradientn(colours = viridis(100))
fatalitiesHeatMap +
  facet_grid(borough~.) +
  theme_calc() +
  theme(panel.grid.major = element_blank()) +
  labs(x = "Hour of the Day", y = "Day of the Week") +
  theme(strip.text.y = element_text(size = 7))
```



With this information, it is clear that the highest volume of accidents is at 8:00 am as well as between 3:00 and 6:00 pm. In order to get a better picture of what might be happening here, let's take a subset of the data that includes those times in the evening. Also, let's look at just Brooklyn again, since that borough clearly has the highest number of accidents. Here's a heat map of just that time.

```
BrooklynEvening <- ny_df %>%
  subset(borough == "BROOKLYN" & hour <= 18 & hour >= 15)
BrooklynEvening %>% group_by(DOW, hour, borough) %>%
  summarize(Injuries = sum(number_of_persons_injured)) %>%
  na.omit() %>%
  ggplot(aes(x = hour, y = DOW, fill = Injuries)) +
  geom_tile(color = "white") +
  scale_fill_gradientn(colours = viridis(100)) +
  theme_calc() +
  theme(panel.grid.major = element_blank()) +
  labs(x = "Hour of the Day", y = "Day of the Week")
```



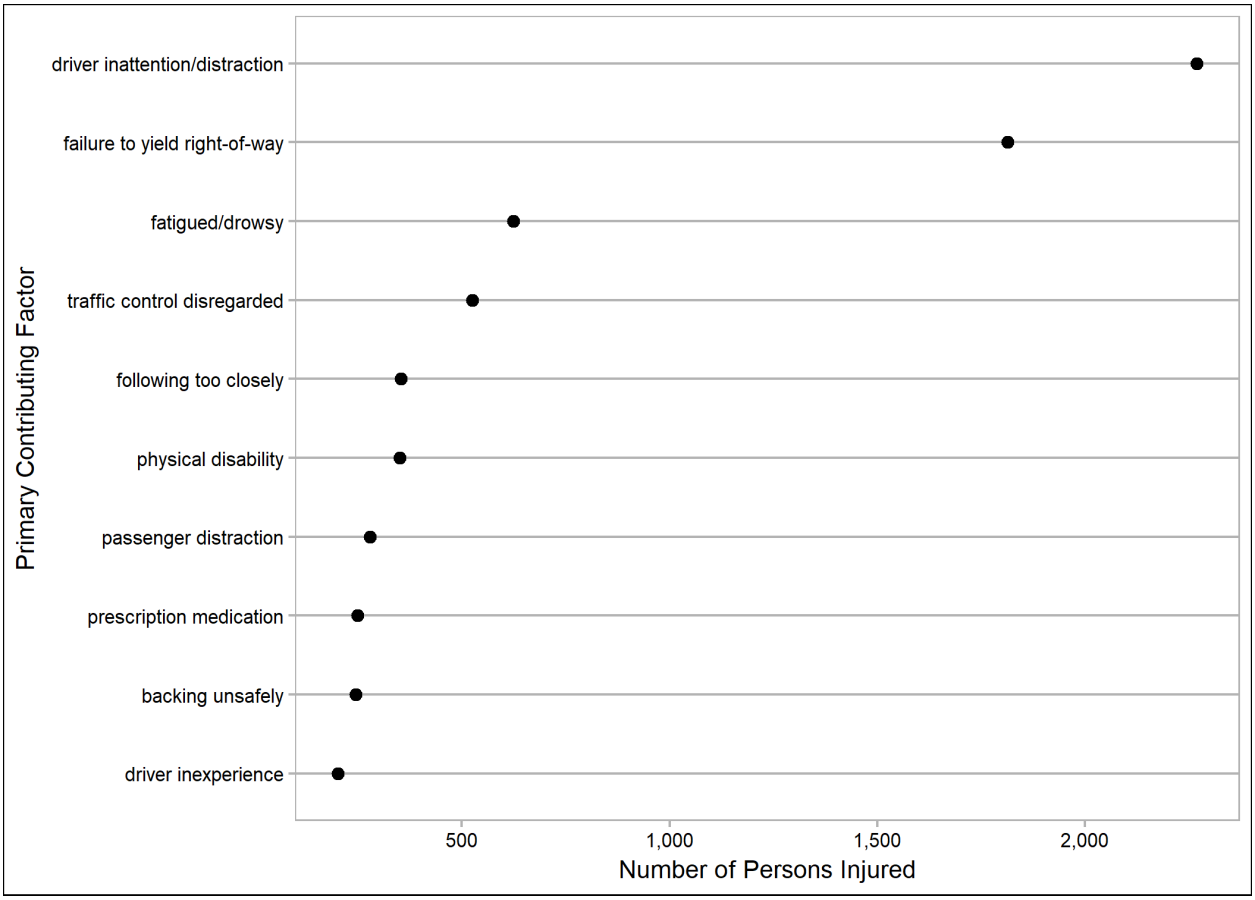
This time is clearly the highest volume of accidents for all of New York. Let's use some of the other variables included in this dataset to get a better picture of what could be causing this. Similar to above, I've included code to cut down on the top 10 contributing factors and vehicle types for this specific subset of the data (3-6 pm in Brooklyn).

```
#Create a top 10 of contributing factors by:
#excluding the non-descriptive titles, grouping by vehicle, summarizing, and arranging
top10 <- BrooklynEvening %>%
  filter(contributing_factor_vehicle_1 != "unspecified",
         contributing_factor_vehicle_1 != "other vehicular") %>%
  group_by(contributing_factor_vehicle_1) %>%
  summarise(injuries = sum(number_of_persons_injured)) %>%
  arrange(injuries) %>%
  top_n(10) %>%
  magrittr::extract2('contributing_factor_vehicle_1')
```

```
## Selecting by injuries
```

```
#Use this top 10 variable to filter the full dataset and exclude the null borough
topFactors <- BrooklynEvening %>%
  filter(contributing_factor_vehicle_1 %in%
           (top10))
#Since TopFactors is now a dataset only including those 10 factors,
#filter it down to include the evening hours and only Brooklyn;
#then plot it to show how many injuries actually occur for each of those vehicle types
topFactors <- topFactors %>%
  group_by(borough, contributing_factor_vehicle_1) %>%
```

```
  summarize(Injuries = sum(number_of_persons_injured)) %>%
  na.omit() %>%
  arrange(Injuries) %>%
  mutate(contributing_factor_vehicle_1 = factor(
    contributing_factor_vehicle_1, levels = contributing_factor_vehicle_1))
topFactors %>%
  ggplot(aes(x = Injuries, y = contributing_factor_vehicle_1)) +
  geom_point(size = 2) +
  theme_calc() +
  scale_x_continuous(labels = comma) +
  labs(x = "Number of Persons Injured", y = "Primary Contributing Factor")
```



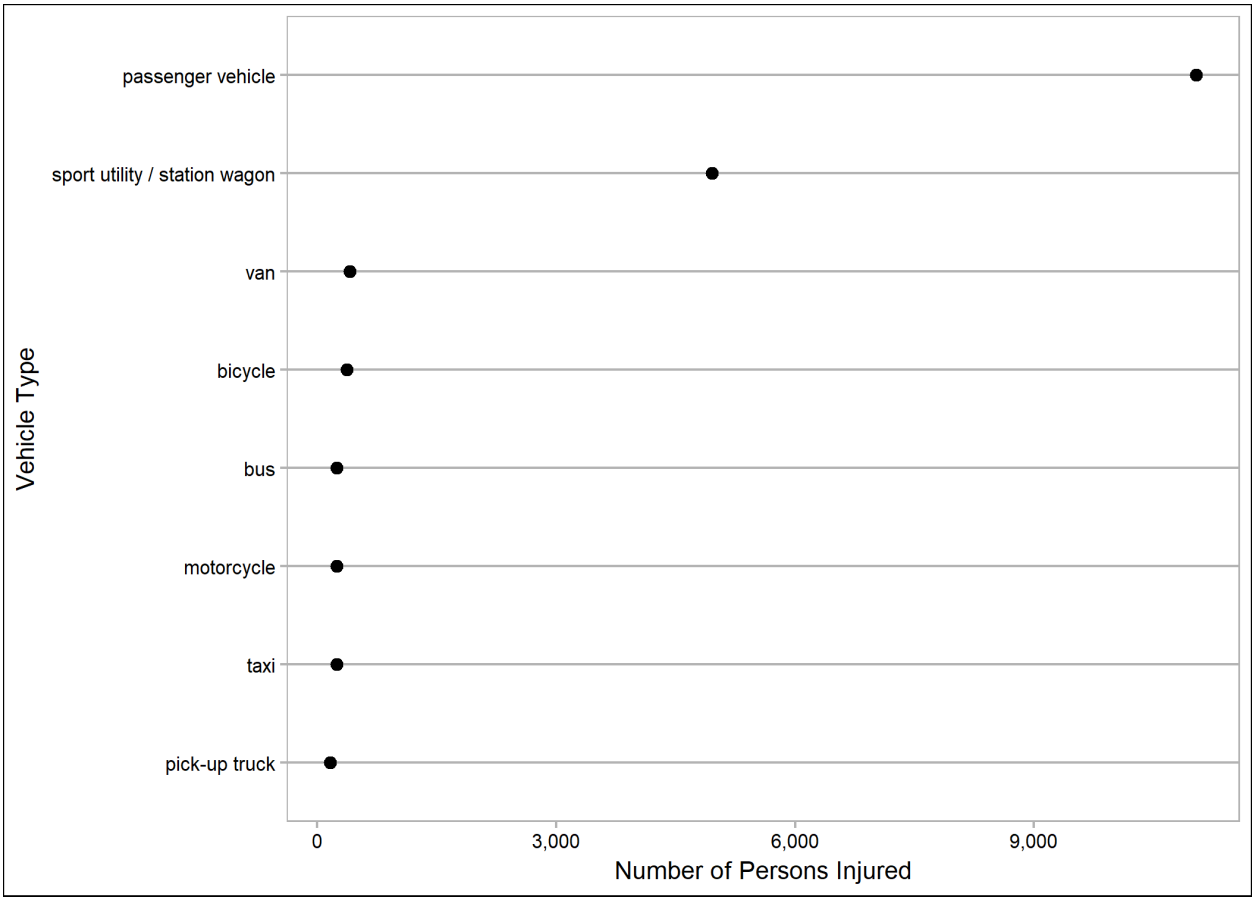
The code below does the same process with the vehicle type.

```
top10vehicles <- BrooklynEvening %>%
  group_by(vehicle_type_code1) %>%
  summarise(injuries = sum(number_of_persons_injured)) %>%
  filter(injuries > 100, !(vehicle_type_code1 %in% c("", "unknown", "other"))) %>%
  arrange(injuries) %>%
  top_n(10) %>%
  magrittr::extract2('vehicle_type_code1')
```

```
## Selecting by injuries
```

```
topVehicles <- BrooklynEvening %>%
  filter(vehicle_type_code1 %in%
```

```
          (top10vehicles)) %>%
  filter(borough != "")
topVehicles <- topVehicles %>%
  #filter(hour < 19 & hour > 14, borough == "BROOKLYN") %>%
  group_by(borough, vehicle_type_code1) %>%
  summarize(Injuries = sum(number_of_persons_injured)) %>%
  na.omit() %>%
  arrange(Injuries) %>%
  mutate(vehicle_type_code1 = factor(vehicle_type_code1, levels = vehicle_type_code1))
topVehicles %>%
  ggplot(aes(x = Injuries, y = vehicle_type_code1)) +
  geom_point(size = 2) +
  theme_calc() +
  scale_x_continuous(labels = comma) +
  labs(x = "Number of Persons Injured", y = "Vehicle Type")
```



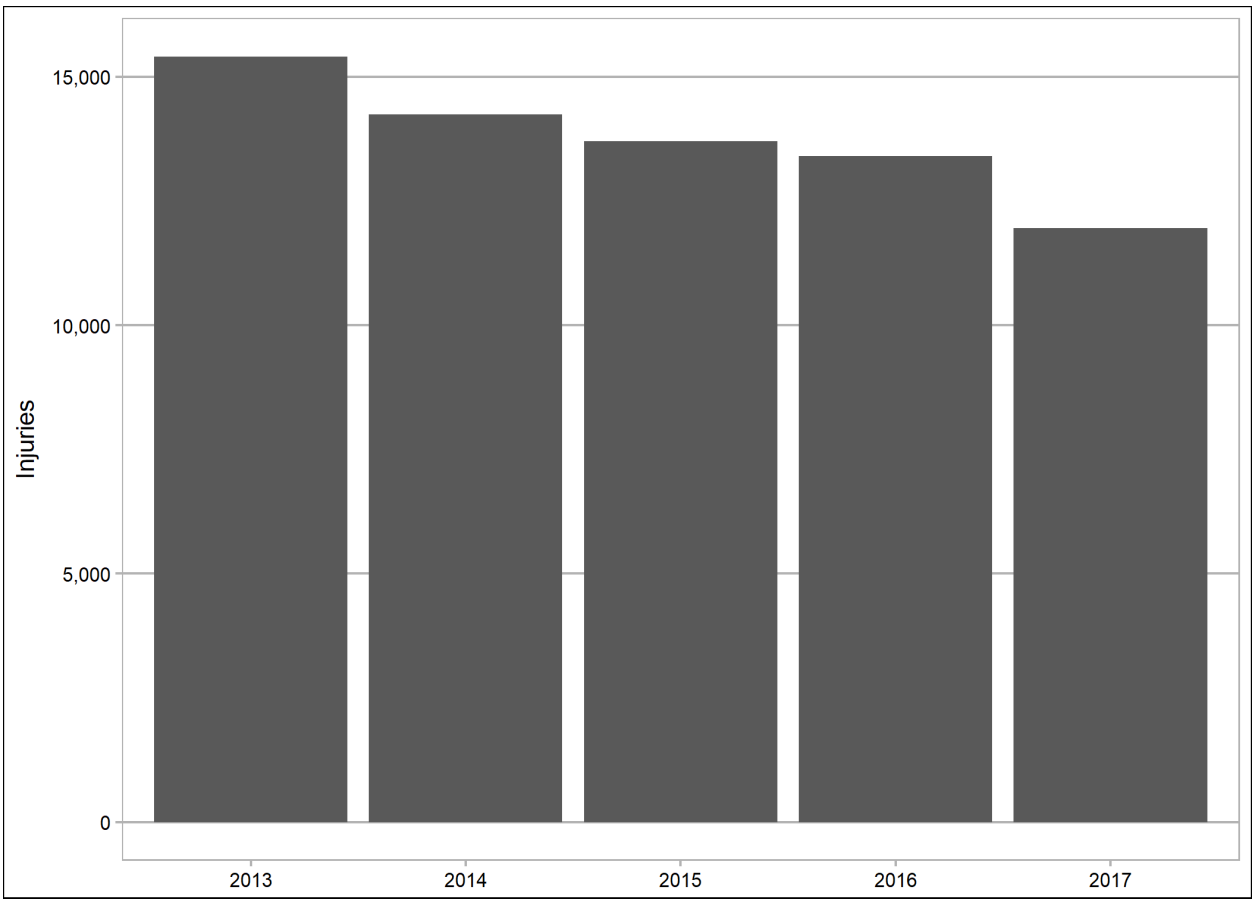
Looking at these graphs in conjunction with the same breakdown shown earlier in the report, we can see that although the highest contributing factor remains the same as the full dataset, the space between that and the second rank is much smaller than with the rest of the dataset. If we pull just that contributing factor out of the data, how has that trended over the years?

CONCLUSION:

The following graph shows the trend between the total number of injuries over the five years included in this subset followed by the number of injuries caused by the "driver inattention/distraction" contributing factor.

```
BrooklynAccidents <- ny_df %>%
  filter(borough == "BROOKLYN") %>%
  group_by(factor(year(date))) %>%
  summarise(Injuries = sum(number_of_persons_injured))
colnames(BrooklynAccidents)[colnames(BrooklynAccidents) == "factor(year(date))"] <- "Year"

BrooklynAccidents %>%
  ggplot(aes(x = Year, y = Injuries)) +
  geom_bar(stat = "identity") +
  theme_calc() +
  theme(axis.title.x = element_blank()) +
  scale_y_continuous(labels = comma)
```

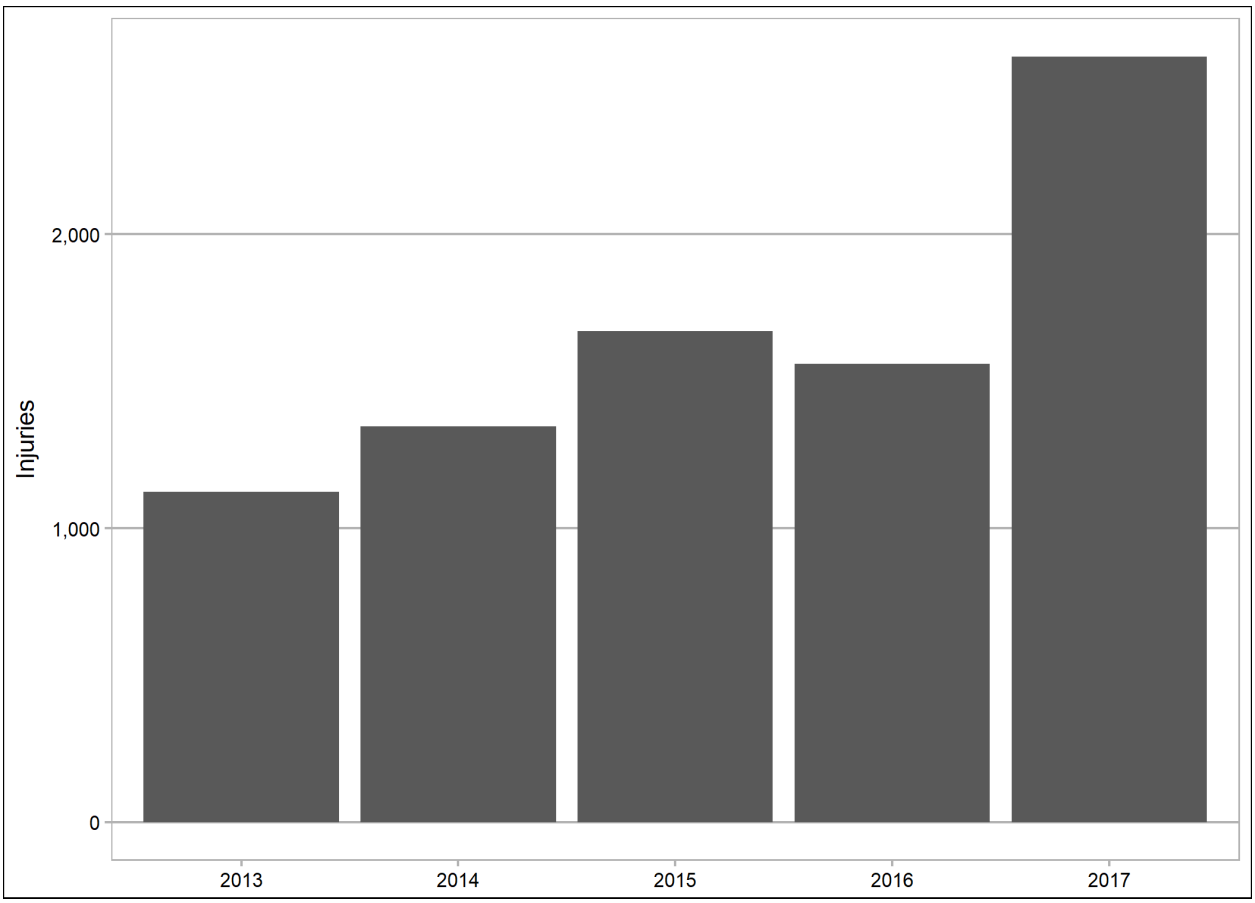


Here we can see the general trend of injuries caused by accidents in Brooklyn has decreased over these five years. Now, let's take a look at the same visualization of the top contributing factor.

```
BrooklynDistraction <- ny_df %>% filter(borough == "BROOKLYN",
                                        contributing_factor_vehicle_1 ==
                                          "driver inattention/distraction")%>%
  group_by(factor(year(date))) %>%
  summarise(Injuries = sum(number_of_persons_injured))
```

```
colnames(BrooklynDistraction)[colnames(BrooklynDistraction)==
                                  "factor(year(date))"] <- "Year"

BrooklynDistraction %>%
  ggplot(aes(x = Year, y = Injuries)) +
  geom_bar(stat = "identity") +
  theme_calc() +
  theme(axis.title.x = element_blank()) +
  scale_y_continuous(labels = comma)
```



This trend is clearly not the same. 2017 seems to have drastically increased in this category. Further research revealed that according to the Governor's Traffic Safety Committee, a law was implemented in the state of New York to prohibit distracted driving involving cell phones by fining the offender and resulting in a mandatory 120-day driver license or permit suspension (http://www.safeny.ny.gov/phon-ndx.htm). This brings us to the first recommendation coming out of this dataset. The police should focus more efforts on enforcing the distracted driving laws specifically in Brooklyn during rush hour. According to the above work, this intervention would significantly cut down on the number of injuries occurring because of car accidents in the city of New York.

There are clearly many other conclusions available which is the remarkable thing about constantly updating datasets that are as inclusive as this one. More police departments across the country are employing data analysts for jobs such as this. For example, Milwaukee was able to identify a specific block where the majority of crimes were happening in the city through data analytics. The city invested in new housing for this community and saw a drastic drop in the number of crimes committed in that neighborhood as a whole. While data analytics is the wave of the future in businesses, other places such as police departments are able

to benefit from the fruits of this crusade.

POTENTIAL FUTURE WORK:

Within this project, there are many other factors that impact these results. Just a few of them were included in the research here. Others were considered for this project but, due to time constraints, were not able to be included. Some of those are listed below.

Many tourists come into the city at all times of the year, but specifically in the summer. Are accident rates higher in the summer in areas of tourist attractions?

Unlike Los Angeles, New York experiences inclement weather during the winter. Precipitation rates in general can also be a factor here. Are certain factors, such as losing control of the vehicle, higher during the winter months?

The city has also seen significant growth over the last few years. Not only would this affect the number of people in the city, but also the number of cars on the road. How has the population density affected the accident rate?

With the implementation of Uber and Lyft over the last few years, how has that affected the number of taxis on the road and consequently the accidents caused by taxis?

Lastly, a potential hypothesis is that accidents would be more likely in an area with higher speed limits. I was able to explore some data revolving around the speed limits in New York (see Appendix 2), but was unable to incorporate it into this project.

All of these factors would warrent further exploration if one was to take this research further and produce any sort of general information available to the public.

APPENDIX 1: DOT Dataset

As mentioned above in this paper, the following is the work I had originally done on the DOT dataset before finding the NYPD data.

The New York City Department of Transportation website has open data available to the public with historical records of motor vehicle accidents. Two of these files have recorded all fatalities and injuries from 2011 to 2016. These files are in json format which can be directly read into R from the website.

```
library(ggmap)
library(maps)
library(jsonlite)
library(rgdal)
library(qcc)
```

```
json_uri <- "http://www.nyc.gov/html/dot/downloads/misc/fatality_all_monthly.json"
ny_fatalities_js <- fromJSON(txt = json_uri, flatten = TRUE)
ny_fatalities_df <- ny_fatalities_js$features

json_uri2 <- "http://www.nyc.gov/html/dot/downloads/misc/injury_all_monthly.json"
ny_injuries_js <- fromJSON(txt = json_uri2, flatten = TRUE)
ny_injuries_df <- ny_injuries_js$features
rm(json_uri, json_uri2)
```

With the format of these files, there needs to be some tidying done in order for R to read it as a dataframe and recognize the coordinates as geometric properites. They are currently being read in as a list within a column which does not convert to a dataframe. In order to correct this, each part of the list needs to be read into its own column. This saves the coordinates into a longitude and latitude column. After that, they need

to be written back into the original dataframe and the column containing the list can be deleted to minimize the size. In the following section of code, I also change the properties of a few other columns in order to more easily edit the data.

```
#Read the two items in the lists into separate variables for the fatality and injury data
x <- rep(NA, times = length(ny_fatalities_df$geometry.coordinates))
y <- rep(NA, times = length(ny_fatalities_df$geometry.coordinates))
for(i in 1:length(ny_fatalities_df$geometry.coordinates)) {
  x[i] <- ny_fatalities_df$geometry.coordinates[[i]][1]
  y[i] <- ny_fatalities_df$geometry.coordinates[[i]][2]
}

z <- rep(NA, times = length(ny_injuries_df$geometry.coordinates))
m <- rep(NA, times = length(ny_injuries_df$geometry.coordinates))
for(i in 1:length(ny_injuries_df$geometry.coordinates)) {
  z[i] <- ny_injuries_df$geometry.coordinates[[i]][1]
  m[i] <- ny_injuries_df$geometry.coordinates[[i]][2]
}

#Once they are split, assign them back into the original dataframe as integers
ny_fatalities_df <- ny_fatalities_df %>%
  mutate(latitude = x,
         longitude = y,
         geometry.coordinates = NULL,
         properties.YR = as.integer(properties.YR),
         properties.MN = as.integer(properties.MN))

ny_injuries_df <- ny_injuries_df %>%
  mutate(latitude = z,
         longitude = m,
         geometry.coordinates = NULL,
         properties.YR = as.integer(properties.YR),
         properties.MN = as.integer(properties.MN))

#Remove the variables that are no longer needed
rm(i, m, x, y, z)
```

Another section of this dataframe that needs to be converted into a more usable format is the date. There are a few things that need to be done to get it into a state that can be used more easily: adding in a column with the day, uniting the columns with the month, day, and year, reading that column in as a date, and removing any rows that do not have a date associated with them.

```
#Add in the day column
ny_fatalities_df$properties.Day <- 1L
ny_injuries_df$properties.Day <- 1L
#Unite the month, day, and year colums
ny_fatalities_df <- unite(ny_fatalities_df, "FullDate",
                                c("properties.YR", "properties.MN", "properties.Day")
                          , sep = "-")
ny_injuries_df <- unite(ny_injuries_df, "FullDate",
                              c("properties.YR", "properties.MN", "properties.Day")
                        , sep = "-")
#Read that column in as a date
ny_fatalities_df$FullDate <- as.Date(ny_fatalities_df$FullDate, format = "%Y-%m-%d")
ny_injuries_df$FullDate <- as.Date(ny_injuries_df$FullDate, format = "%Y-%m-%d")
```

```
#Check for any lines that do not have a date associated with them
#Also remove any lines that are NA
sum(is.na(ny_fatalities_df))
```

```
## [1] 14
```

```
ny_fatalities_df <- ny_fatalities_df[complete.cases(ny_fatalities_df$FullDate), ]
sum(is.na(ny_fatalities_df))
```

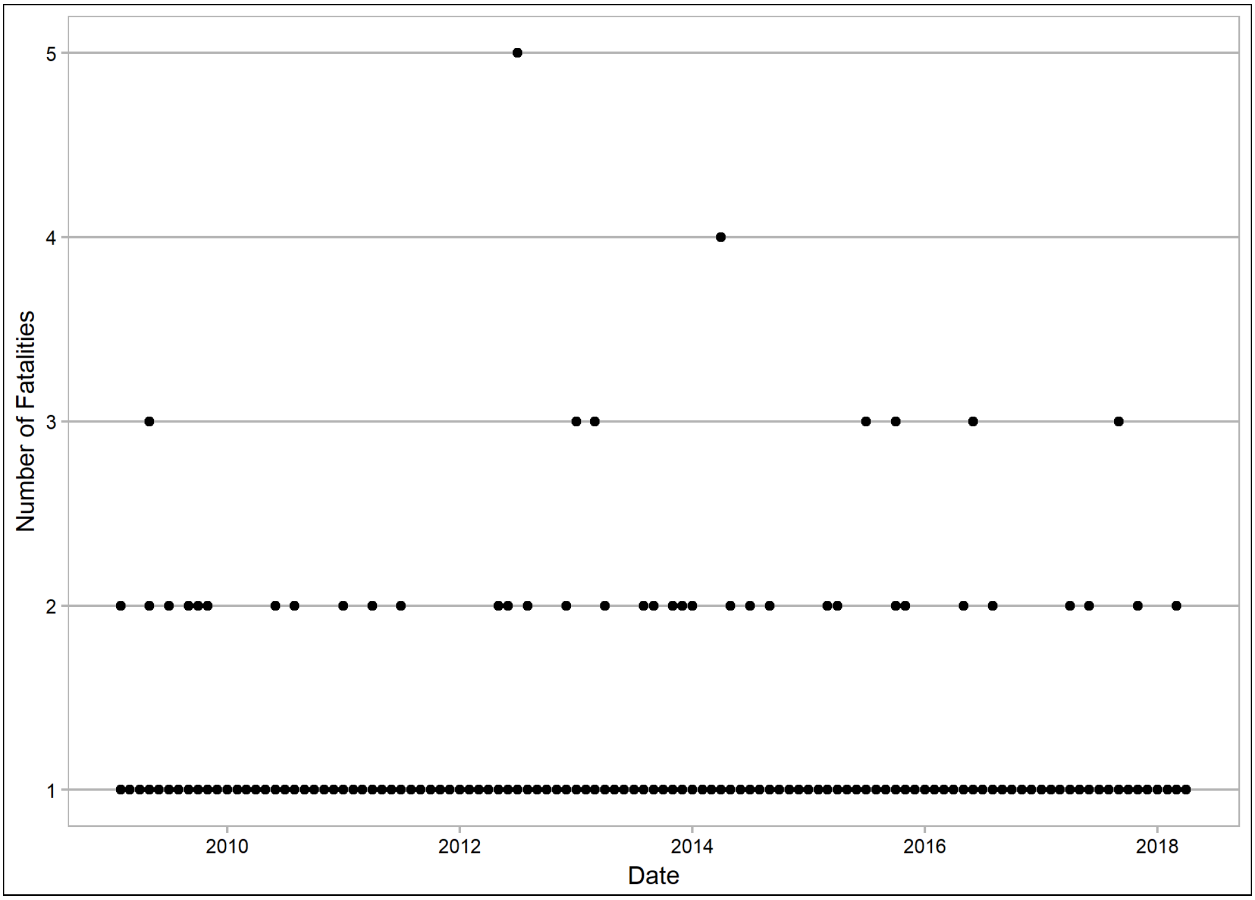
```
## [1] 0
```

```
sum(is.na(ny_injuries_df))
```

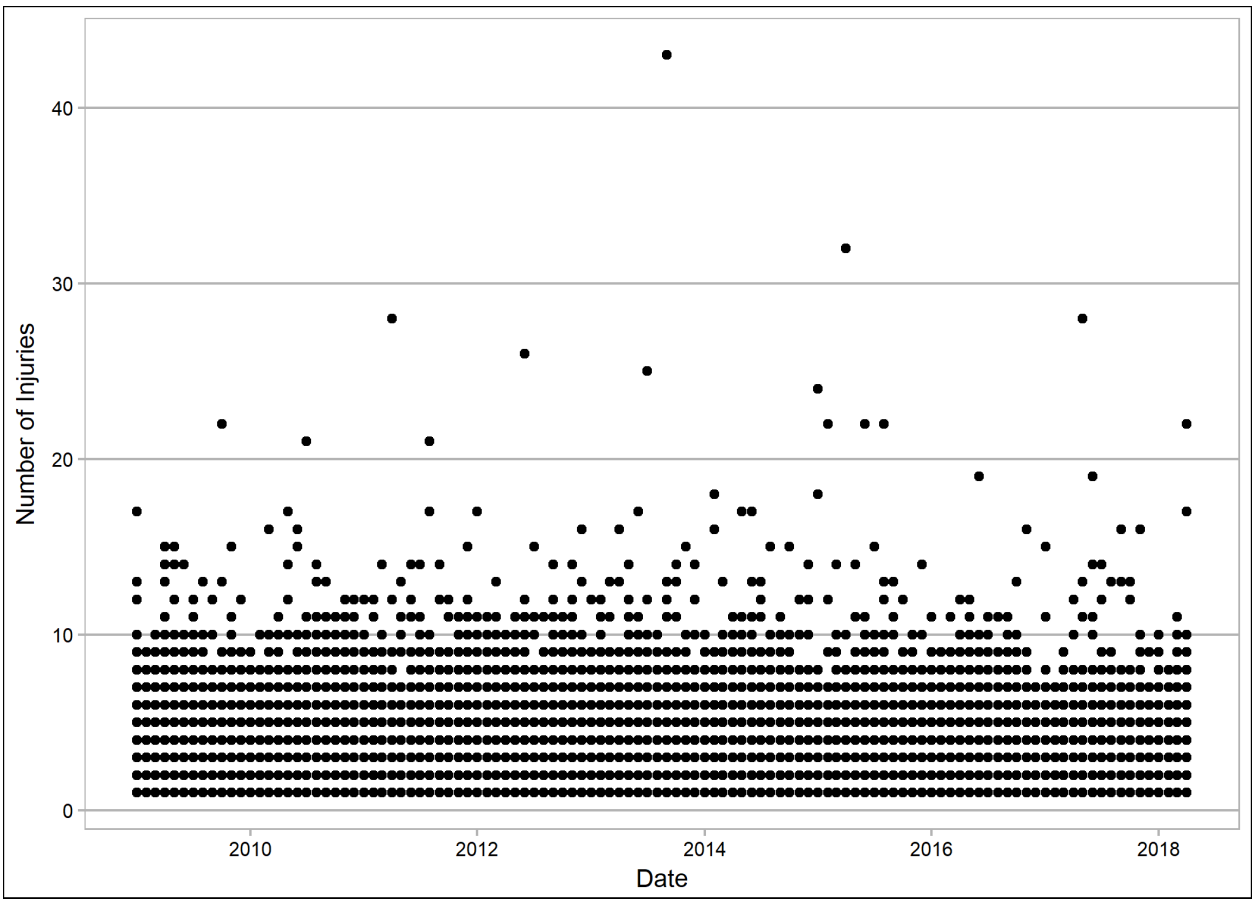
```
## [1] 0
```

Since the sum of the is.na(ny_injuries_df) = 0, there are no rows to be removed as there were in the fatalities dataframe. From here, we are able to do a general, basic plot of each dataframe by the date.

```
ggplot(ny_fatalities_df, aes(x = FullDate, y = properties.Fatalities)) +
  geom_point() +
  theme_calc() +
  labs(x = "Date", y = "Number of Fatalities")
```




```
ggplot(ny_injuries_df, aes(x = FullDate, y = properties.Injuries)) +
  geom_point() +
  theme_calc() +
  labs(x = "Date", y = "Number of Injuries")
```



Although these plots are helpful and generally informative, I would like to be able to look at the injury and fatality data on the same plot. Before combining the datasets, it is necessary to determine if there are any duplicates between the two files. To do this, I need to find a way to identify each accident with its own identifier. There are already latitude and longitude indicators on each accident along with the month in which it occurred.

A problem arises if there is more than one accident at the same location within the same month. To account for this, first, I use paste() to combine the date, latitude, and longitude to create a unique identifier for each accident. Second, I checked the unique count of this identifier against the length of the entire dataset. If those numbers are equal, there are no duplicates between the month and the location.

```
#Combine date, latitude, and longitude
ny_fatalities_df$Identifier <-
  paste(ny_fatalities_df$FullDate,
        ny_fatalities_df$latitude + ny_fatalities_df$longitude,
        sep = "")
#Check that the unique number of this identifier matches the number of rows in the whole dataframe
length(unique(ny_fatalities_df$Identifier)) == nrow(ny_fatalities_df)
```

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

```
#Same as above for the inuries data
ny_injuries_df$Identifier <-
  paste(ny_injuries_df$FullDate,
        ny_injuries_df$latitude + ny_injuries_df$longitude,
        sep = "")
length(unique(ny_injuries_df$Identifier)) == nrow(ny_injuries_df)
```

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

Because both statements for the injuries and the fatalities returned true, this identifier is in fact unique for each accident record. In order to join these two datasets, full_join() is used to include all instances on both datasets for any of these identifiers. This new dataset is called "accident_monthly". The two totals printed from this section will show the number of lines result from the combined dataset as well as how many lines between the two datasets referred to the same accident.

```
nrow(ny_fatalities_df) + nrow(ny_injuries_df)
```

```
## [1] 269286
```

```
accident_monthly <- full_join(ny_fatalities_df, ny_injuries_df,
                              by = c("Identifier" = "Identifier"))
(nrow(ny_fatalities_df) + nrow(ny_injuries_df)) - nrow(accident_monthly)
```

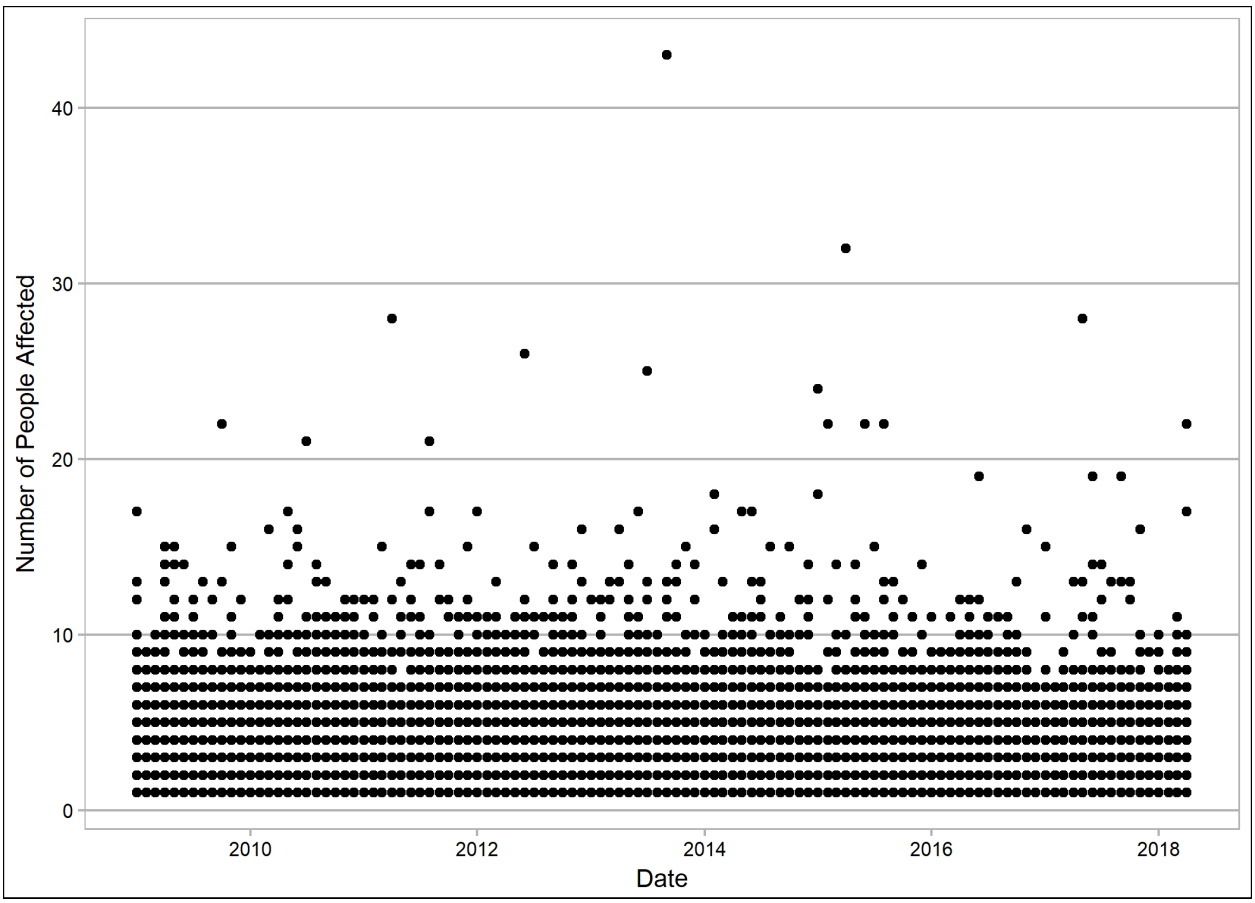
```
## [1] 726
```

A few more summary columns are necessary now that the data has been combined. The following code creates four summaries around the number of people affected in each of the categories - pedestrians, cyclists, drivers/passengers, and people total.

```
accident_monthly$TotalPedestrian <- rowSums(accident_monthly[,c("properties.PedFatalit",
                                            "properties.PedInjurie")], na.rm = TRUE)
accident_monthly$TotalBike <- rowSums(accident_monthly[,c("properties.BikeFatali",
                                            "properties.BikeInjuri")], na.rm = TRUE)
accident_monthly$TotalMVO <- rowSums(accident_monthly[,c("properties.MVOFatalit",
                                            "properties.MVOInjurie")], na.rm = TRUE)
accident_monthly$PeopleAffected <- rowSums(accident_monthly[,c("properties.Fatalities",
                                            "properties.Injuries")], na.rm = TRUE)
```

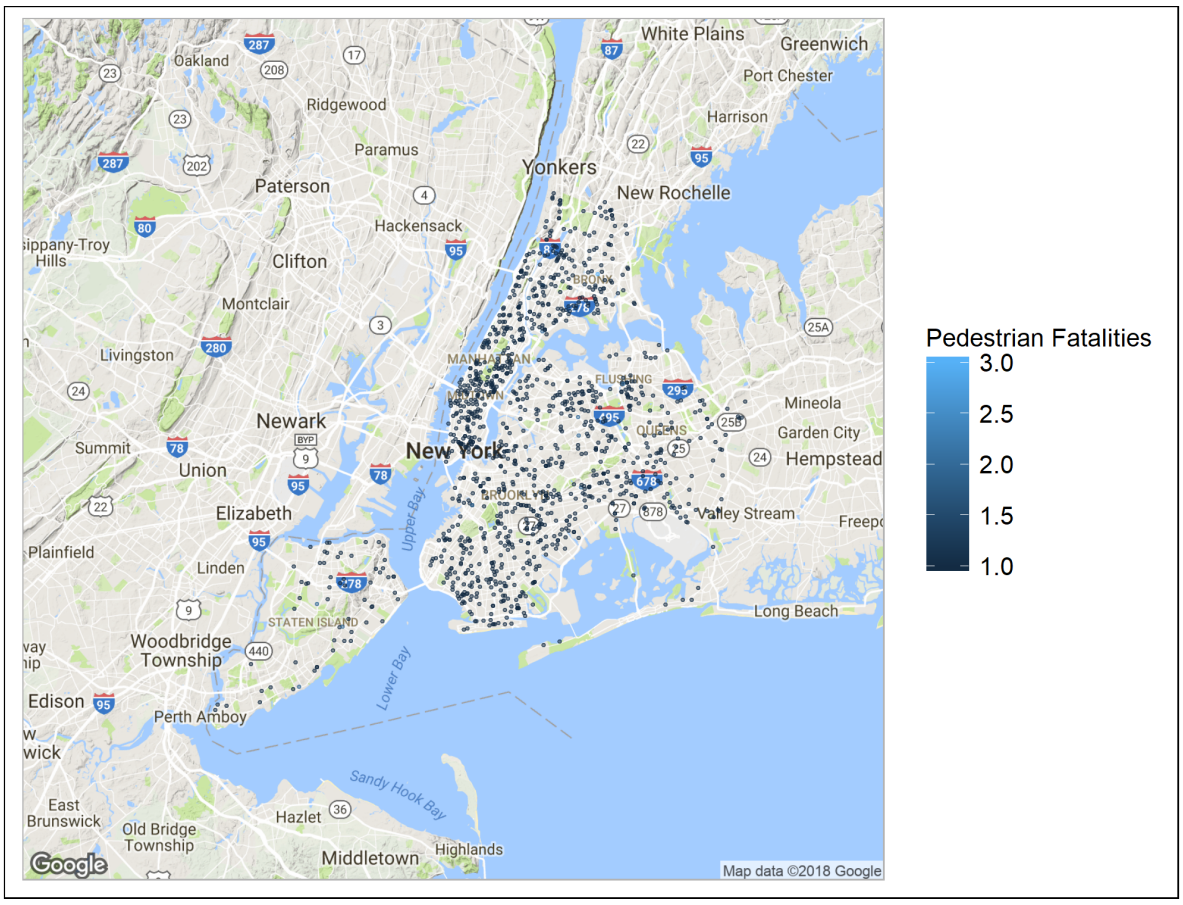
Now that the data has been combined and summarized, we can recreate the basic plots from earlier using the full dataset. The only step that needs to still be completed is to combine all the dates from the separate datasets into one column to use that as the x-axis.

```
#Create one full date to apply to the entire dataset
accident_monthly$FullDate.x[is.na(accident_monthly$FullDate.x)] <-
  (accident_monthly$FullDate.y[is.na(accident_monthly$FullDate.x)])
#Plot this using the new full date as the x-axis and total people affected as the y-axis
ggplot(accident_monthly, aes(x = FullDate.x, y = PeopleAffected)) +
  geom_point() +
  theme_calc() +
  labs(x = "Date", y = "Number of People Affected")
```



The following map gives a general outline for where the accidents take place. This gives a good picture of the city with the fatality causing accidents. Due to the volume of data, this is rather overwhelming and not entirely helpful to use with the full dataset at this point.

```
#Download the map to use as the background
nyc_map <- get_map("new york", zoom = 10)
#Filter out any accidents with no fatalities
fatalitiesPed_df <- ny_fatalities_df %>%
  filter(properties.PedFatalit > 0)
#Plot the data over the map
ggmap(nyc_map) +
  geom_point(data = fatalitiesPed_df,
                      aes(x = latitude, y = longitude,
                          colour = properties.PedFatalit),
             alpha = 0.5, size = 0.5) +
  theme_calc() +
  theme(axis.title.x = element_blank(), axis.title.y = element_blank(),
        axis.text = element_blank(), axis.ticks = element_blank()) +
  labs(colour = "Pedestrian Fatalities")
```




```
#Remove the filter created above to keep the environment as clean as possible
rm(fatalitiesPed_df)
```

APPENDIX 2: Speed Limit Data

The DOT also has a great deal of information on speed limits within the city of New York. As this would help to understand whether accidents tend to happen in areas with higher or lower speed limits, I jumped at the chance to be able to input and analyze this data. It also comes in a json format which is read into R below; however, the website has one file per borough so all of the data wrangling must be done five times. In order to keep the environment clear and as much memory free as possible, all the temporary variables will be removed after their usefulness expires as can be seen in the last line of code below.

```
json_uri3 <- "http://www.nyc.gov/html/dot/downloads/misc/speed_limit_bronx.json"
speedlimits_Bronx_js <- fromJSON(txt = json_uri3, flatten = TRUE)
speedlimits_Bronx_df <- speedlimits_Bronx_js$features

json_uri4 <- "http://www.nyc.gov/html/dot/downloads/misc/speed_limit_brooklyn.json"
speedlimits_Brooklyn_js <- fromJSON(txt = json_uri4, flatten = TRUE)
speedlimits_Brooklyn_df <- speedlimits_Brooklyn_js$features

json_uri5 <- "http://www.nyc.gov/html/dot/downloads/misc/speed_limit_manhattan.json"
speedlimits_Manhattan_js <- fromJSON(txt = json_uri5, flatten = TRUE)
speedlimits_Manhattan_df <- speedlimits_Manhattan_js$features

json_uri6 <- "http://www.nyc.gov/html/dot/downloads/misc/speed_limit_queens.json"
```

```
speedlimits_Queens_js <- fromJSON(txt = json_uri6, flatten = TRUE)
speedlimits_Queens_df <- speedlimits_Queens_js$features

json_uri7 <- "http://www.nyc.gov/html/dot/downloads/misc/speed_limit_statenisland.json"
speedlimits_StatenIsland_js <- fromJSON(txt = json_uri7, flatten = TRUE)
speedlimits_StatenIsland_df <- speedlimits_StatenIsland_js$features

rm(json_uri3, json_uri4, json_uri5, json_uri6, json_uri7)
```

In looking into the coordinate columns in this dataset, you can see that they read in as a list inside the dataframe, similar to how the accident data was in the first section. There isn't a lot of detail about this dataset on the website, but from a cursory look at the coordinate sections, these are most likely areas of a road that have a certain speed limit. These have been put in the dataset in list form. In order to properly view this as a dataframe, the list will need to be read into four separate columns - start latitude, start longitude, end latitude, and end longitude. The following code is an attempt to turn this list into those four separate columns using a for loop; however, as mentioned previously, there are five sets of data - one from each borough. This loop needs to be run on each of those datasets.

```
#Create variables for each latitude and longitude
a <- rep(NA, times = length(speedlimits_Bronx_df$geometry.coordinates))
b <- rep(NA, times = length(speedlimits_Bronx_df$geometry.coordinates))
c <- rep(NA, times = length(speedlimits_Bronx_df$geometry.coordinates))
d <- rep(NA, times = length(speedlimits_Bronx_df$geometry.coordinates))
#Read each of these columns back into the original dataframe
for(i in 1:length(speedlimits_Bronx_df$geometry.coordinates)) {
  a[i] <- speedlimits_Bronx_df$geometry.coordinates[[i]][1]
  b[i] <- speedlimits_Bronx_df$geometry.coordinates[[i]][2]
  c[i] <- speedlimits_Bronx_df$geometry.coordinates[[i]][3]
  d[i] <- speedlimits_Bronx_df$geometry.coordinates[[i]][4]
}

speedlimits_Bronx_df <- speedlimits_Bronx_df %>%
  mutate(start_latitude = a,
         start_longitude = c,
         end_latitude = b,
         end_longitude = d,
         geometry.coordinates = NULL)

#Same process for Brooklyn
a <- rep(NA, times = length(speedlimits_Brooklyn_df$geometry.coordinates))
b <- rep(NA, times = length(speedlimits_Brooklyn_df$geometry.coordinates))
c <- rep(NA, times = length(speedlimits_Brooklyn_df$geometry.coordinates))
d <- rep(NA, times = length(speedlimits_Brooklyn_df$geometry.coordinates))
for(i in 1:length(speedlimits_Brooklyn_df$geometry.coordinates)) {
  a[i] <- speedlimits_Brooklyn_df$geometry.coordinates[[i]][1]
  b[i] <- speedlimits_Brooklyn_df$geometry.coordinates[[i]][2]
  c[i] <- speedlimits_Brooklyn_df$geometry.coordinates[[i]][3]
  d[i] <- speedlimits_Brooklyn_df$geometry.coordinates[[i]][4]
}

speedlimits_Brooklyn_df <- speedlimits_Brooklyn_df %>%
  mutate(start_latitude = a,
         start_longitude = c,
         end_latitude = b,
```

```
         end_longitude = d,
         geometry.coordinates = NULL)

#Same process for Manhattan
a <- rep(NA, times = length(speedlimits_Manhattan_df$geometry.coordinates))
b <- rep(NA, times = length(speedlimits_Manhattan_df$geometry.coordinates))
c <- rep(NA, times = length(speedlimits_Manhattan_df$geometry.coordinates))
d <- rep(NA, times = length(speedlimits_Manhattan_df$geometry.coordinates))
for(i in 1:length(speedlimits_Manhattan_df$geometry.coordinates)) {
  a[i] <- speedlimits_Manhattan_df$geometry.coordinates[[i]][1]
  b[i] <- speedlimits_Manhattan_df$geometry.coordinates[[i]][2]
  c[i] <- speedlimits_Manhattan_df$geometry.coordinates[[i]][3]
  d[i] <- speedlimits_Manhattan_df$geometry.coordinates[[i]][4]
}

speedlimits_Manhattan_df <- speedlimits_Manhattan_df %>%
  mutate(start_latitude = a,
         start_longitude = c,
         end_latitude = b,
         end_longitude = d,
         geometry.coordinates = NULL)

#Same process for Queens
a <- rep(NA, times = length(speedlimits_Queens_df$geometry.coordinates))
b <- rep(NA, times = length(speedlimits_Queens_df$geometry.coordinates))
c <- rep(NA, times = length(speedlimits_Queens_df$geometry.coordinates))
d <- rep(NA, times = length(speedlimits_Queens_df$geometry.coordinates))
for(i in 1:length(speedlimits_Queens_df$geometry.coordinates)) {
  a[i] <- speedlimits_Queens_df$geometry.coordinates[[i]][1]
  b[i] <- speedlimits_Queens_df$geometry.coordinates[[i]][2]
  c[i] <- speedlimits_Queens_df$geometry.coordinates[[i]][3]
  d[i] <- speedlimits_Queens_df$geometry.coordinates[[i]][4]
}

speedlimits_Queens_df <- speedlimits_Queens_df %>%
  mutate(start_latitude = a,
         start_longitude = c,
         end_latitude = b,
         end_longitude = d,
         geometry.coordinates = NULL)

#Same process for Staten Island
a <- rep(NA, times = length(speedlimits_StatenIsland_df$geometry.coordinates))
b <- rep(NA, times = length(speedlimits_StatenIsland_df$geometry.coordinates))
c <- rep(NA, times = length(speedlimits_StatenIsland_df$geometry.coordinates))
d <- rep(NA, times = length(speedlimits_StatenIsland_df$geometry.coordinates))
for(i in 1:length(speedlimits_StatenIsland_df$geometry.coordinates)) {
  a[i] <- speedlimits_StatenIsland_df$geometry.coordinates[[i]][1]
  b[i] <- speedlimits_StatenIsland_df$geometry.coordinates[[i]][2]
  c[i] <- speedlimits_StatenIsland_df$geometry.coordinates[[i]][3]
  d[i] <- speedlimits_StatenIsland_df$geometry.coordinates[[i]][4]
}
```

```
speedlimits_StatenIsland_df <- speedlimits_StatenIsland_df %>%
  mutate(start_latitude = a,
         start_longitude = c,
         end_latitude = b,
         end_longitude = d,
         geometry.coordinates = NULL)

rm(a, b, c, d, i)
```

Now that all of these are converted to separate columns and the temporary variables have been removed from the environment, they can all be combined into one dataframe.

```
speedlimits_NY <- bind_rows(speedlimits_Bronx_df, speedlimits_Brooklyn_df,
                            speedlimits_Manhattan_df, speedlimits_Queens_df,
                            speedlimits_StatenIsland_df)
```

In reviewing this dataset, it appears some of the start combinations and end combinations of coordinates are not pairing up correctly between latitude and longitude. I went back to the original datasets pulled from the DOT website and it appears that these lists are not always of length 4. Some of them go up to 14. With a lack of explanation on the website as to what these actually refer to as well as the inconsistency in lengths of this list, I'm forced to move on without this dataset. Any attempts to clean this up and make it usable extend quite far beyond the point of this project and course. Perhaps at a later time, I'll be able to come back and look into this further.