nypd

1/28/2022

PROJECT 2: NYPD DATASET

In this write-up I will be going over the NYPD csv file as directed in the class from the data.gov website. Specifically I will be using the 'NYPD Shooting Incident Data(Historic)' file which can be found at this link.

Data Importing

To begin I will import some libraries that I will plan on using, mainly 'tidyr', 'tidyverse', 'lubridate', and 'ggplot2'. Next I will import the csv file that I will be working on into a local variable named 'nypd_csv'.

```
library(tidyr)
library(tidyverse)
library(lubridate)
library(ggplot2)

url <- 'https://data.cityofnewyork.us/api/views/833y-fsy8/rows.csv?accessType=DOWNLOAD'
nypd_csv <- read_csv(url)
```

Data Wrangling

As there are a good number of columns in the data, due to the fact that I do not require the large majority of them I created a new dataframe based on the original csv but with columns that I determine as being usable. This will cut down greatly the number of data to work with and ease the burden of wading through many un-necessary rows. The new dataframe will simply be called 'df' for ease of typing throughout the document.

```
keep <- c("OCCUR_DATE", "OCCUR_TIME", "BORO", "PRECINCT", "LOCATION_DESC", "PERP_AGE_GROUP", "PERP_SEX"
df <- nypd_csv[keep]

summary(df)
```

```
##   OCCUR_DATE         OCCUR_TIME           BORO              PRECINCT
##  Length:23585       Length:23585      Length:23585       Min.   :  1.00
##  Class :character   Class1:hms        Class :character   1st Qu.: 44.00
##  Mode  :character   Class2:difftime   Mode  :character   Median : 69.00
##                     Mode  :numeric                        Mean   : 66.21
##                                                          3rd Qu.: 81.00
##                                                          Max.   :123.00
##  LOCATION_DESC      PERP_AGE_GROUP       PERP_SEX         VIC_AGE_GROUP
```

```
##  Length:23585       Length:23585       Length:23585       Length:23585
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##    VIC_SEX             Latitude       Longitude
##  Length:23585       Min.   :40.51   Min.   :-74.25
##  Class :character   1st Qu.:40.67   1st Qu.:-73.94
##  Mode  :character   Median :40.70   Median :-73.92
##                     Mean   :40.74   Mean   :-73.91
##                     3rd Qu.:40.82   3rd Qu.:-73.88
##                     Max.   :40.91   Max.   :-73.70
```

The next step will be to check my new dataframe for any NaN or NA values.

```
colSums(is.na(df))/nrow(df)
```

```
##     OCCUR_DATE     OCCUR_TIME           BORO       PRECINCT  LOCATION_DESC
##      0.0000000      0.0000000      0.0000000      0.0000000      0.5758321
## PERP_AGE_GROUP       PERP_SEX  VIC_AGE_GROUP        VIC_SEX       Latitude
##      0.3517066      0.3502650      0.0000000      0.0000000      0.0000000
##      Longitude
##      0.0000000
```

As we can see, both the location description columns as well as information on the perpetrator both have significant amounts of invalid values, with location description being more than half filled with NAs. Due to the large percentage, I determined that simply dropping the rows would be too harmful to the amount of data so I decided to fill in all the NA values with the phrase 'UNKNOWN'. Luckily, both the perpetrator age group and sex columns are set to be 'char' values as well so this will be an easy process of replacing one value with another. If we run another 'colSums()' we can see that there are no more NA values to be found.

```
df <- df %>% replace_na(list(LOCATION_DESC = 'UNKNOWN', PERP_AGE_GROUP = 'UKNOWN', PERP_SEX = 'UNKNOWN'))
colSums(is.na(df))/nrow(df)
```

```
##     OCCUR_DATE     OCCUR_TIME           BORO       PRECINCT  LOCATION_DESC
##              0              0              0              0              0
## PERP_AGE_GROUP       PERP_SEX  VIC_AGE_GROUP        VIC_SEX       Latitude
##              0              0              0              0              0
##      Longitude
##              0
```

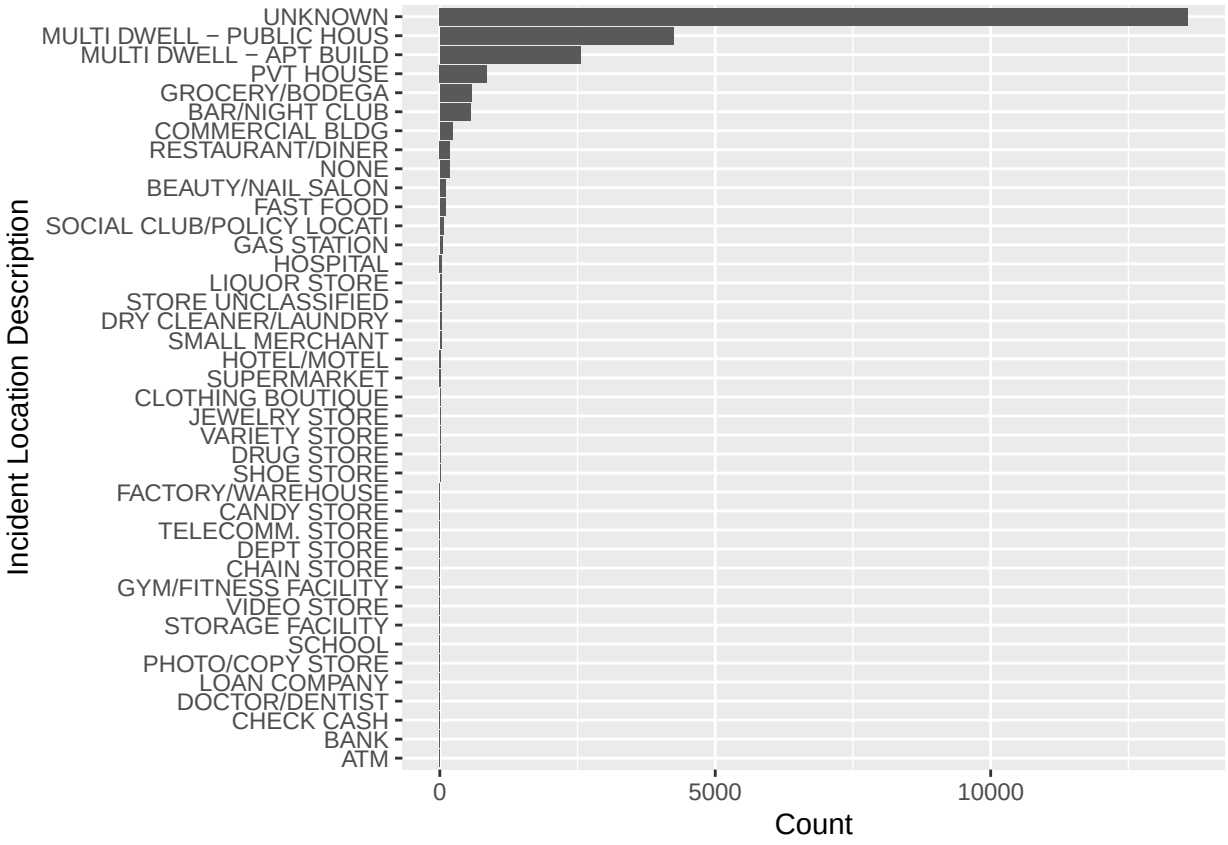
Another bit of cleaning that I have done includes combining the perpetrator and victim columns into one column. By combining the two, it will give a better sense of the demographic involved in the incident which I will use for visualizations later in the document.

```
df <- df %>% unite('PERP', PERP_AGE_GROUP:PERP_SEX, remove = TRUE)
df <- df %>% unite('VIC', VIC_AGE_GROUP:VIC_SEX, remove = TRUE)
```

Plots

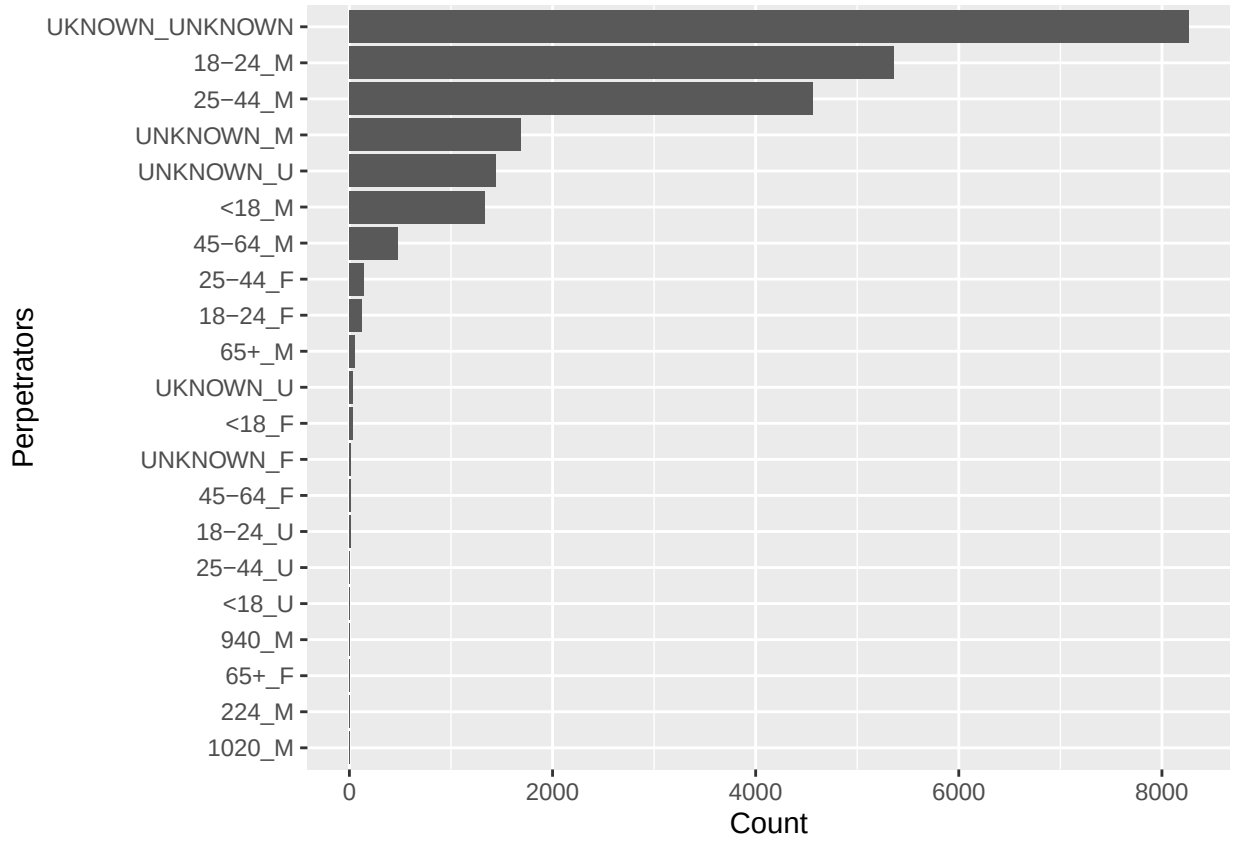
The first plot that I have done involves looking at the type of locations the incidents are more likely to happen in. Ignoring the fact that most incidents do not have a stated location (unknown), we can see that of the incidents that do have their location stated, the large majority are located in residential buildings.

```
g <- ggplot(df, aes(y=reorder(LOCATION_DESC,LOCATION_DESC, function(y)+length(y)))) + geom_bar()
g + labs(x = "Count", y = "Incident Location Description")
```



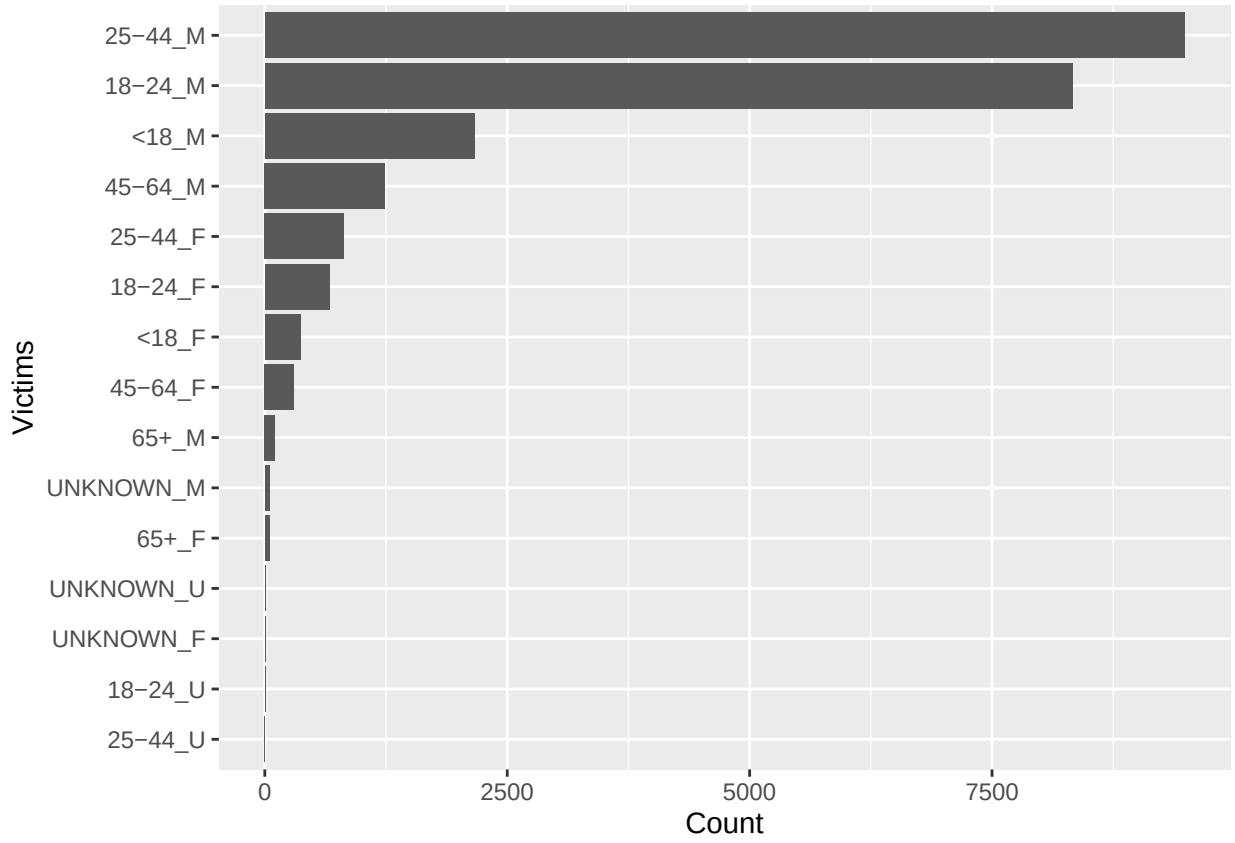
The second plot is determining the top demographic that the perpetrator is more likely to come from. While unknown values are still rampant in many incidents, we can see that males vastly take up the rankings when it comes to being the perpetrator, with 18-24 age range at the top followed by 25-44, then less than 18.

```
g <- ggplot(df, aes(y=reorder(PERP,PERP, function(y)+length(y)))) + geom_bar()
g + labs(x = "Count", y = "Perpetrators")
```



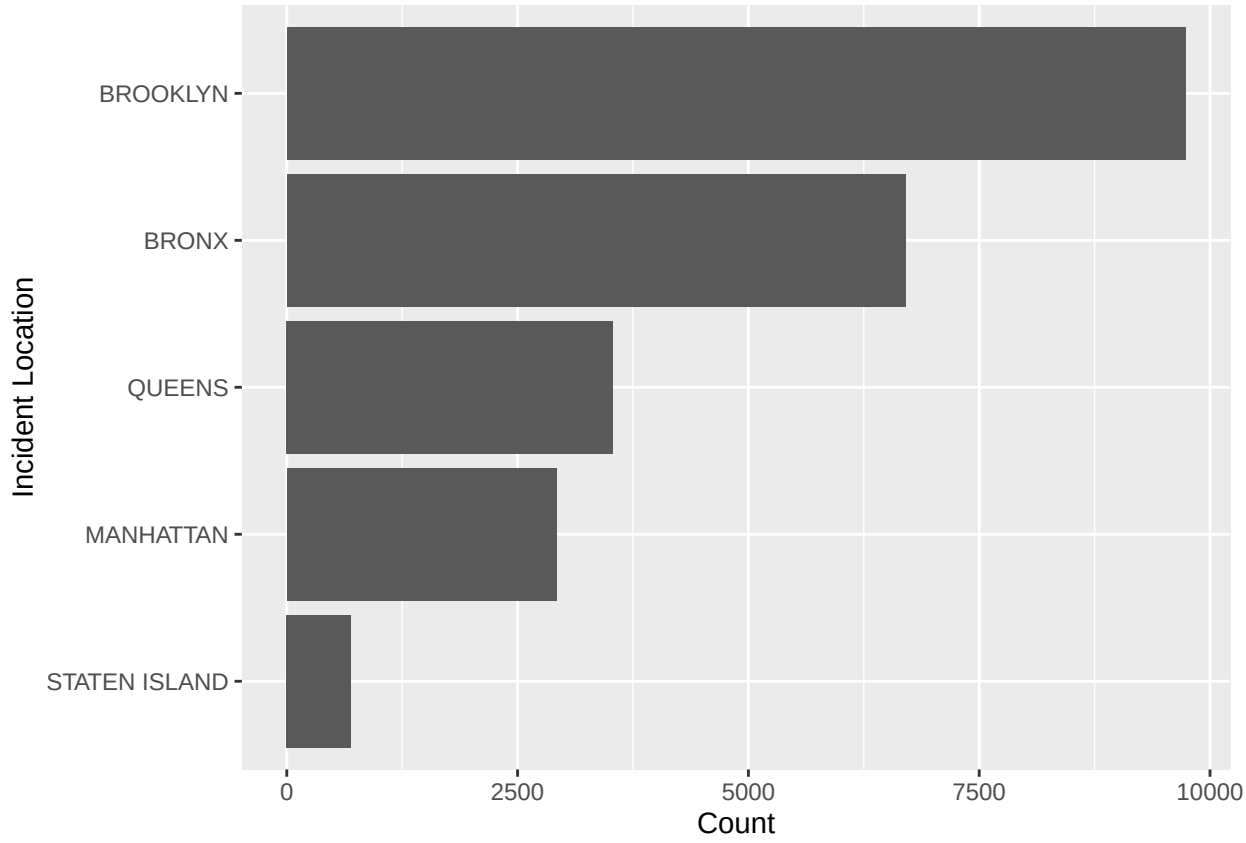
On the other hand, the third plot looks at the victims for each incident; unlike the perpetrators, these are much more well documented so the number of unknowns is significantly lower. From the rankings we can see that similarly males from ages 18-44 are the most common victims.

```
g <- ggplot(df, aes(y=reorder(VIC,VIC, function(y)+length(y)))) + geom_bar()
g + labs(x = "Count", y = "Victims")
```



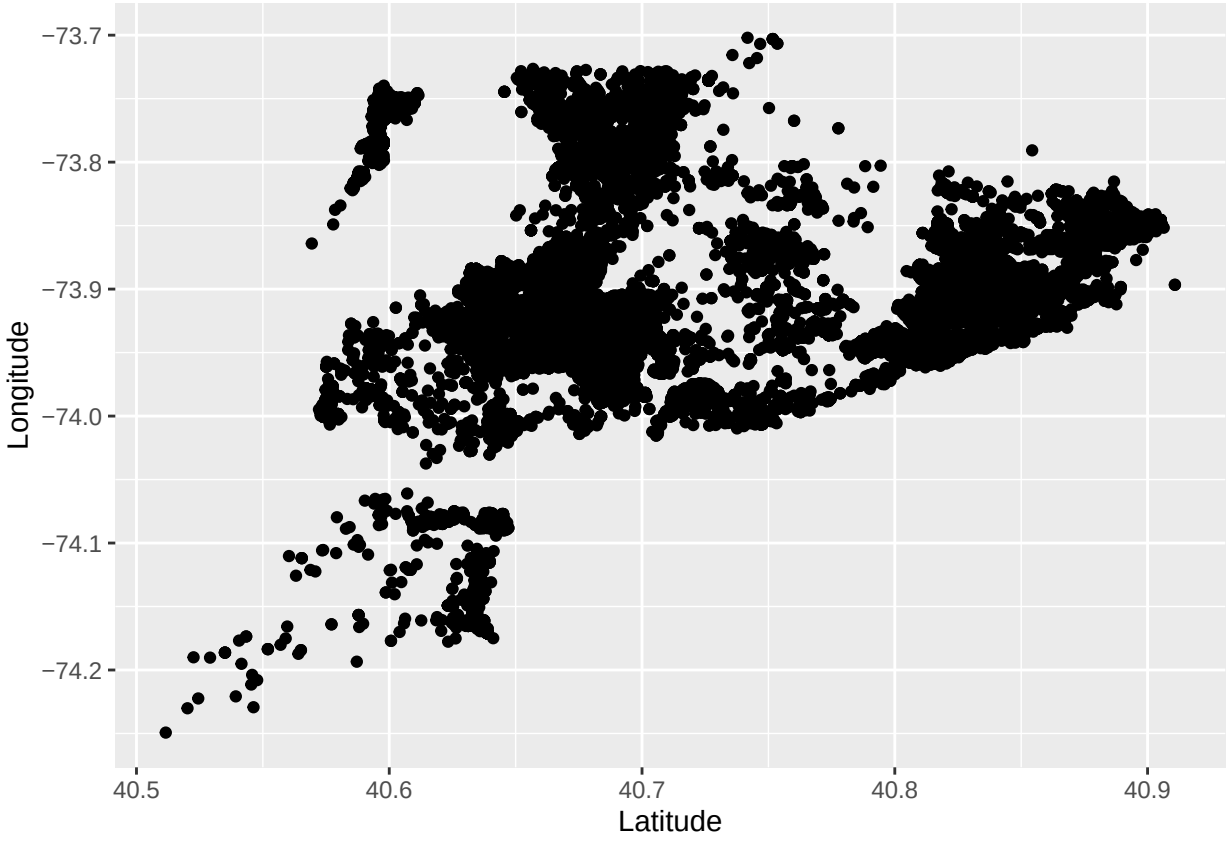
The fourth plot looks at the location where most of the incidents take place when it comes to official boroughs in New York. The Brooklyn borough takes the significant lead over the other locations, with almost more than the 2nd and 3rd place combined.

```
g <- ggplot(df, aes(y=reorder(BORO,BORO, function(y)+length(y)))) + geom_bar()
g + labs(x = "Count", y = "Incident Location")
```



Lastly in the fifth plot we have the longitude and latitude for each incident plotted. While I do not have the technical expertise yet to overlap a map of New York onto this plot, judging from my 4th plot I can connect the two and assume that the highest ranking boros on the countplot are also located where the dots in this plot are clumped together.

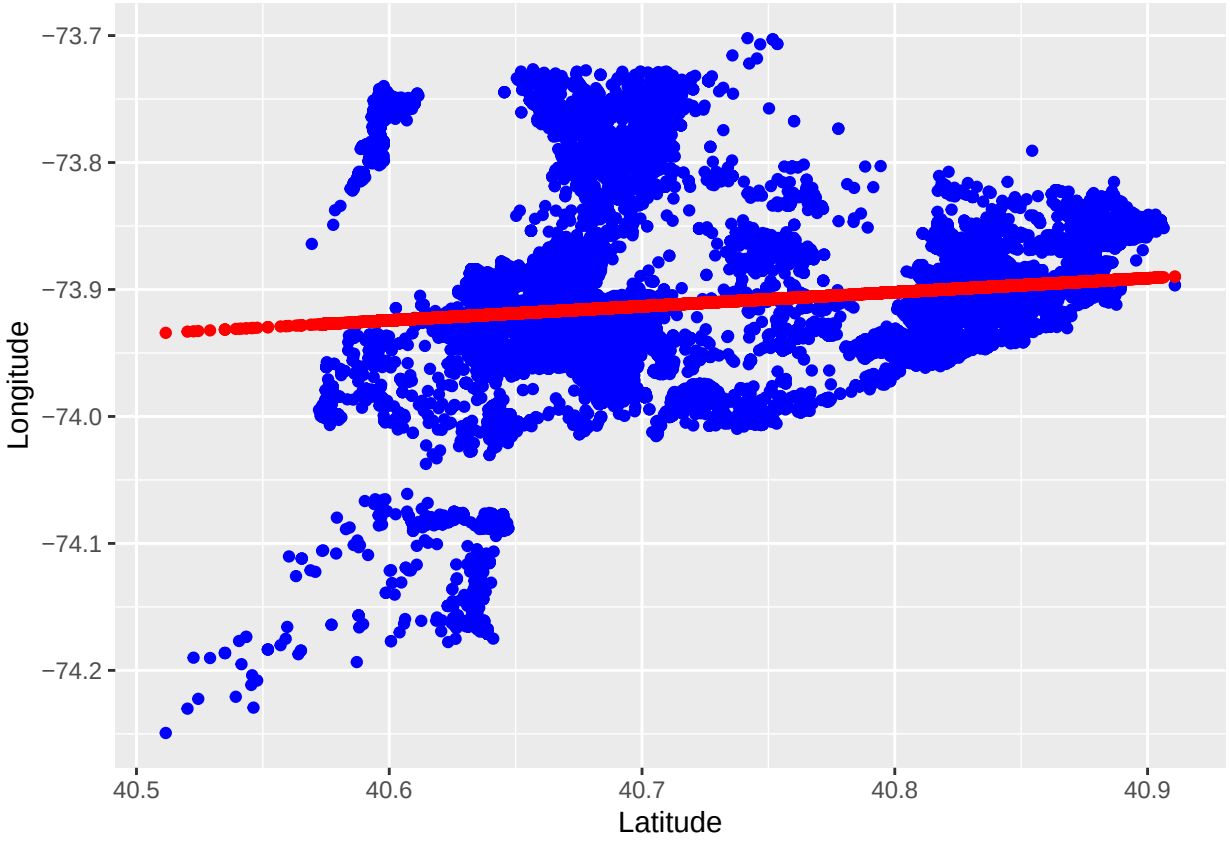
```
g <- ggplot(df, aes(x=Latitude, y=Longitude)) + geom_point()
g + labs(x = "Latitude", y = "Longitude")
```



Modeling

For my model, I decided to use my previous plot since it plotted two convenient numerous variables together and utilize them for a linear model. I wanted to predict an incident's assumed Longitude value given a Latitude value so I created a grid of sequential values from 40.5 to 40.9 (roughly the minimum and maximum latitude values of New York in our given dataset) with an increment of 0.1 between each value as inputs for the potential model. After creating a model built on the original dataset, I then created a new one where I mutated the old dataframe and added a new 'pred' column which applied my model coefficients to my artificially created inputs, then plotted it along with the original values. While the accuracy of this model is undoubtedly questionable due to the way longitude and latitude values work, at the very least from this plot we can infer that the large majority of the crimes are centered around the red line.

```
mod <- lm(Longitude ~ Latitude, data = df)
x_grid <- seq(40.5, 40.9, 0.1)
df_w_pred <- df %>% mutate(pred = predict(mod))
df_w_pred %>% ggplot() + geom_point(aes(x = Latitude, y = Longitude), color = "blue") + geom_point(aes(
```



Conclusion and Bias

In conclusion, when it comes to bias, I believe that the dataset itself isn't very biased due to being official documents from a government agency. I trust (or hope) that police data will be unbiased and attempt to accurately document the incident as best as they can due to the necessity for important investigations. However I will admit that I do have my own personal biases, as whenever I think of the phrase 'criminal' I picture a young adult male. To combat this I tried making the section on the perpetrators' demographic

just a single part of the document, and instead focus on other pieces of information that can be inferred from the data such as location and location description. By looking at a map of boroughs in New York with longitudes and latitudes, it is indeed true that the borough with the highest amount of incidents, Brooklyn, also contains the majority of the dots in our plot of incidents.

Session Info

```
## R version 4.1.2 (2021-11-01)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19043)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
## system code page: 932
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] lubridate_1.8.0 forcats_0.5.1   stringr_1.4.0   dplyr_1.0.7
##  [5] purrr_0.3.4     readr_2.0.2     tibble_3.1.5    ggplot2_3.3.5
##  [9] tidyverse_1.3.1 tidyr_1.1.4
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.7       assertthat_0.2.1 digest_0.6.28    utf8_1.2.2
##  [5] R6_2.5.1         cellranger_1.1.0 backports_1.3.0  reprex_2.0.1
##  [9] evaluate_0.14    highr_0.9        httr_1.4.2       pillar_1.6.4
## [13] rlang_0.4.12     curl_4.3.2       readxl_1.3.1     rstudioapi_0.13
## [17] rmarkdown_2.11   labeling_0.4.2   bit_4.0.4        munsell_0.5.0
## [21] broom_0.7.10     compiler_4.1.2   modelr_0.1.8     xfun_0.29
## [25] pkgconfig_2.0.3  htmltools_0.5.2  tidyselect_1.1.1 fansi_0.5.0
## [29] crayon_1.4.2     tzdb_0.2.0       dbplyr_2.1.1     withr_2.4.2
## [33] grid_4.1.2       jsonlite_1.7.2   gtable_0.3.0     lifecycle_1.0.1
## [37] DBI_1.1.1        magrittr_2.0.1   scales_1.1.1     cli_3.1.0
## [41] stringi_1.7.5    vroom_1.5.5      farver_2.1.0     fs_1.5.0
## [45] xml2_1.3.2       ellipsis_0.3.2   generics_0.1.1   vctrs_0.3.8
## [49] tools_4.1.2      bit64_4.0.5      glue_1.4.2       hms_1.1.1
## [53] parallel_4.1.2   fastmap_1.1.0    yaml_2.2.1       colorspace_2.0-2
## [57] rvest_1.0.2      knitr_1.36       haven_2.4.3
```