Data Science as a Field - R Markdown Assignment

Chandan Saha

2/26/2022

NYPD Shooting Data 2006-2020

Packages used for this Project include:

```
tidyverse
lubridate
```

Purpose

The purpose of this study is to take a deeper dive into the shooting data of the NYPD from 2006 to 2020. In New York City, there are two schools of thought regarding the NYPD. One side wants to de-fund the police and redistribute those funds to other city agencies. The other side states that there is a rise in crime in NYC and the NYPD is needed now more than ever.
We will take a look at the shooting data to see if there is a rise in crime and if so, how and when it has happened.

Import

We will be importing public data involving NYPD shooting incident rates (Historic).

To import the data we will use the public data located at

https://data.cityofnewyork.us/api/views/833y-fsy8/rows.csv?accessType=DOWNLOAD

Using the read_csv command, we can import the data into R.

```
shooting_data = read_csv("https://data.cityofnewyork.us/api/views/833y-fsy8/rows.csv?accessType=DOWNLOA
```

Tidy and Clean

Now that the data in imported will first take a look at the data, then we will proceed to clean it.

To take a quick look at the data we can use commands such as summary.

```
summary(shooting_data)
```

```
##   INCIDENT_KEY        OCCUR_DATE         OCCUR_TIME           BORO
##  Min.   :  9953245   Length:23585       Length:23585       Length:23585
##  1st Qu.: 55322804   Class :character   Class1:hms         Class :character
##  Median : 83435362   Mode  :character   Class2:difftime    Mode  :character
##  Mean   :102280741                      Mode  :numeric
##  3rd Qu.:150911774
##  Max.   :230611229
##
##     PRECINCT      JURISDICTION_CODE LOCATION_DESC      STATISTICAL_MURDER_FLAG
##  Min.   :  1.00   Min.   :0.000     Length:23585       Mode :logical
##  1st Qu.: 44.00   1st Qu.:0.000     Class :character   FALSE:19085
##  Median : 69.00   Median :0.000     Mode  :character   TRUE :4500
##  Mean   : 66.21   Mean   :0.333
##  3rd Qu.: 81.00   3rd Qu.:0.000
##  Max.   :123.00   Max.   :2.000
##                   NA's   :2
##  PERP_AGE_GROUP       PERP_SEX           PERP_RACE          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             VIC_RACE           X_COORD_CD        Y_COORD_CD
##  Length:23585       Length:23585       Min.   : 914928   Min.   :125757
##  Class :character   Class :character   1st Qu.: 999925   1st Qu.:182539
##  Mode  :character   Mode  :character   Median :1007654   Median :193470
##                                        Mean   :1009379   Mean   :207300
##                                        3rd Qu.:1016782   3rd Qu.:239163
##                                        Max.   :1066815   Max.   :271128
##
##     Latitude       Longitude        Lon_Lat
##  Min.   :40.51   Min.   :-74.25   Length:23585
##  1st Qu.:40.67   1st Qu.:-73.94   Class :character
##  Median :40.70   Median :-73.92   Mode  :character
##  Mean   :40.74   Mean   :-73.91
##  3rd Qu.:40.82   3rd Qu.:-73.88
##  Max.   :40.91   Max.   :-73.70
##
```

Using this summary method, we are able to see all the columns and even some more information about the columns themselves, such as the data types that are in those columns. In this case, using the summary method, we were able to see that the **OCCUR_DATE** column is a character class, when it should really be a date time class. To fix this, we can use the *lubridate* library to turn the date into a date-time format. In this case we would use the following code:

```
library(lubridate)
shooting_data$OCCUR_DATE <- mdy(shooting_data$OCCUR_DATE, tz=Sys.timezone())
shooting_data
```

```
## # A tibble: 23,585 x 19
##    INCIDENT_KEY OCCUR_DATE          OCCUR_TIME BORO    PRECINCT JURISDICTION_CO~
```

```
##           <dbl> <dttm>              <time>     <chr>      <dbl>            <dbl>
##  1     24050482 2006-08-27 00:00:00 05:35      BRONX         52                0
##  2     77673979 2011-03-11 00:00:00 12:03      QUEENS       106                0
##  3    203350417 2019-10-06 00:00:00 01:09      BROOKL~       77                0
##  4     80584527 2011-09-04 00:00:00 03:35      BRONX         40                0
##  5     90843766 2013-05-27 00:00:00 21:16      QUEENS       100                0
##  6     92393427 2013-09-01 00:00:00 04:17      BROOKL~       67                0
##  7     73057167 2010-06-05 00:00:00 21:16      BROOKL~       77                0
##  8    211362213 2020-03-20 00:00:00 21:27      BROOKL~       81                0
##  9    137564752 2014-07-04 00:00:00 00:25      QUEENS       101                0
## 10    147024011 2015-10-18 00:00:00 01:33      QUEENS       106                0
## # ... with 23,575 more rows, and 13 more variables: LOCATION_DESC <chr>,
## #   STATISTICAL_MURDER_FLAG <lgl>, PERP_AGE_GROUP <chr>, PERP_SEX <chr>,
## #   PERP_RACE <chr>, VIC_AGE_GROUP <chr>, VIC_SEX <chr>, VIC_RACE <chr>,
## #   X_COORD_CD <dbl>, Y_COORD_CD <dbl>, Latitude <dbl>, Longitude <dbl>,
## #   Lon_Lat <chr>
```

You can see that the **OCCUR_DATE** column now is represented with *dttm* instead of *chr*.

Next, looking at all the different columns, we see that there are many that have NA's and some columns we would not be interested in for this project.

- Columns we aren't interested in:
  - latitude
  - longitude
  - x_coord
  - y_coord
  - location_desc
  - perp_age_group
  - perp_sex,
  - perp_race

To remove these unwanted columns we can simply make shooting_data a subset of the original dataset.

```
shooting_data <- select(shooting_data, -Latitude, - Longitude, -Lon_Lat, -X_COORD_CD, -Y_COORD_CD, -PERP
```

Visualization and Analysis

Now our data looks much more usable and from this data, I see that there are 5 boroughs in New York City, and that makes me curious as to the relative rates of shootings in these different Boroughs. Looking at the total number of shootings in each Borough can give us a deeper insight into how these Boroughs compare when it comes to the number of shootings from 2006-2020.

To do this we need to make a smaller table adding up the instances of shootings in each Borough.

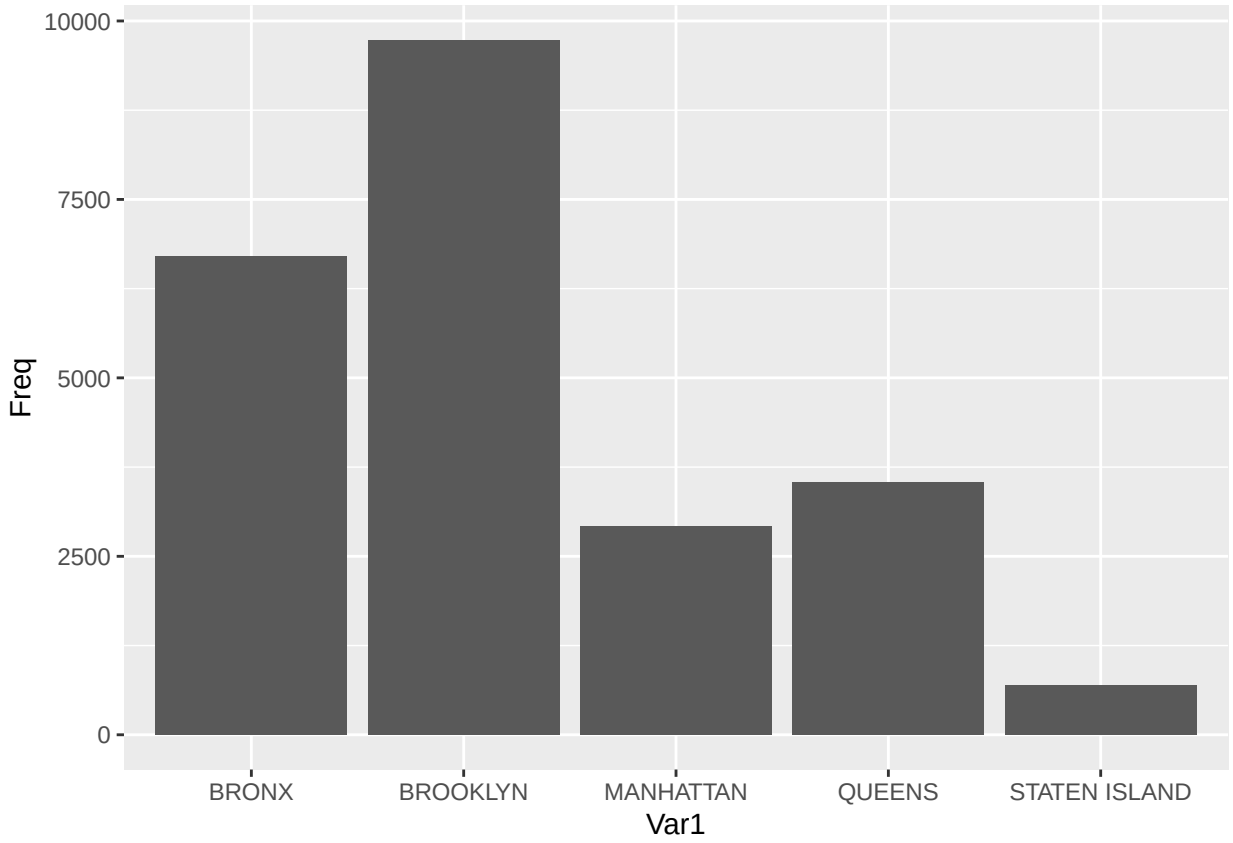
```
boro_data <- table(shooting_data$BORO)
boro_data <- as.data.frame(boro_data)
boro_data
```

```
##            Var1 Freq
## 1         BRONX 6701
```

```
## 2      BROOKLYN 9734
## 3     MANHATTAN 2922
## 4        QUEENS 3532
## 5 STATEN ISLAND  696
```

Graphing this data will make the relationship much more clear. To do this we use the **ggplot** command.

```
ggplot(data = boro_data, mapping = aes( x = Var1, y = Freq)) + geom_col()
```



Now that we have the data comparing the Boroughs to each other, we also should look at the Boroughs individually to see if they follow this pattern every year or if there have been changes over the course of the time period of this data.

Further Analysis

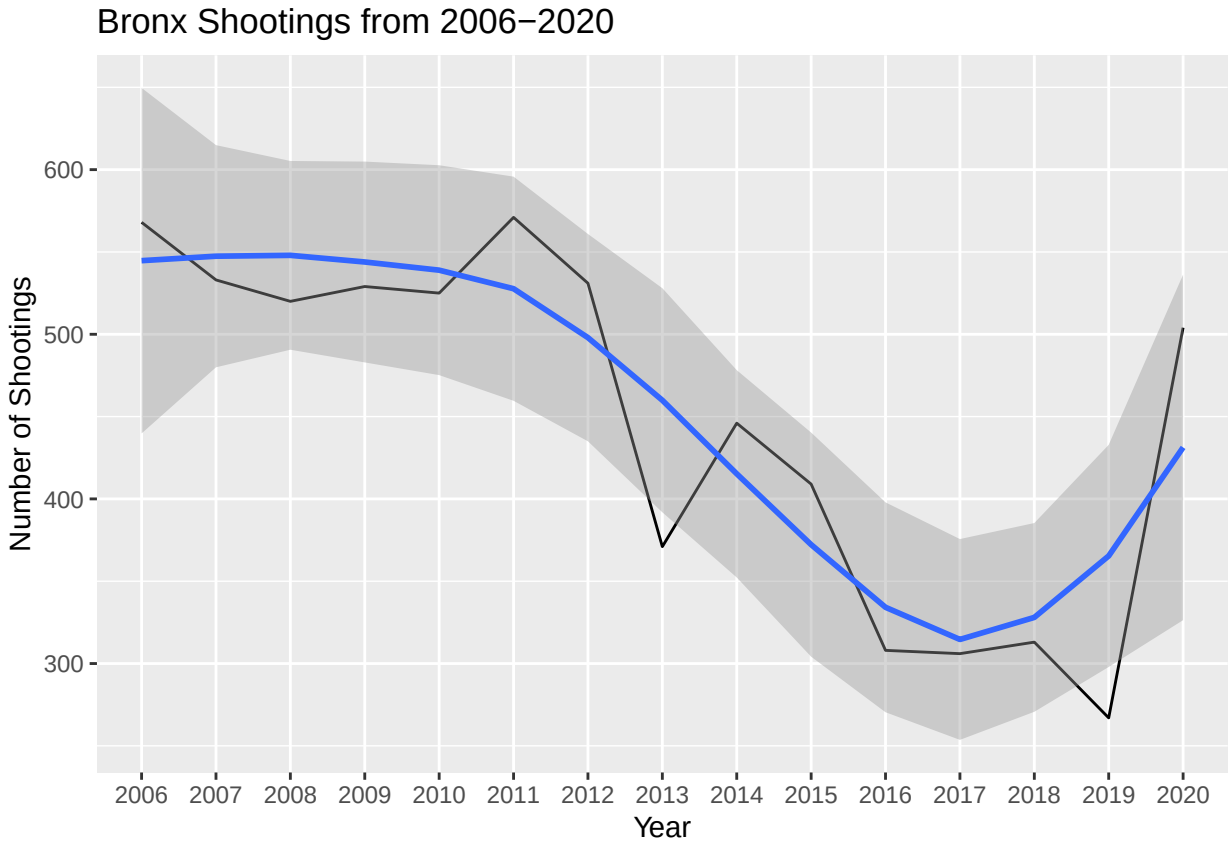
In order to look at the Boroughs individually, we first take a subset of the **shooting_data** representing one of the Boroughs and its **OCCUR_DATE**.

```
Bronx <- subset(shooting_data, BORO=='BRONX', select = c(BORO, OCCUR_DATE))
```

Since our data for **OCCUR_DATE** is in year/month/day format, and we want only the year, we use the *lubridate* library to extract just the year. Afterwards we are able to graph the data. We add a trendline to model our data using **geom_smooth**

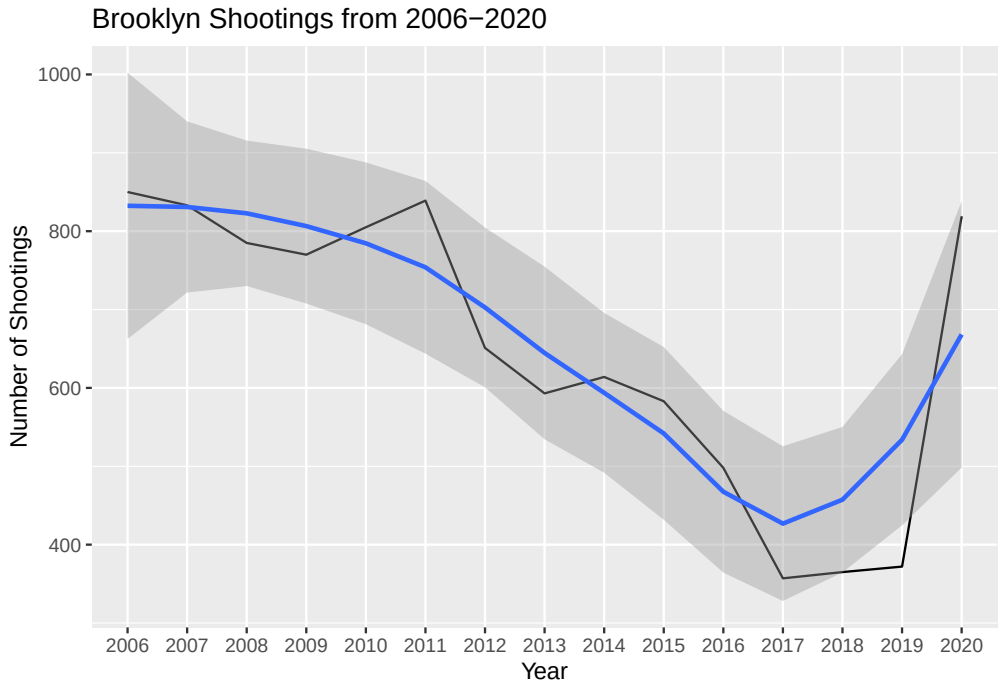
```
Bronx$YEAR <- year(Bronx$OCCUR_DATE)
Bronx <- subset(Bronx, select = -c(OCCUR_DATE))
Bronx <- table(Bronx$YEAR)
Bronx <- as.data.frame(Bronx)
ggplot(data = Bronx, mapping = aes( x = Var1, y = Freq, group = 1)) + geom_line() + geom_smooth() + labs
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

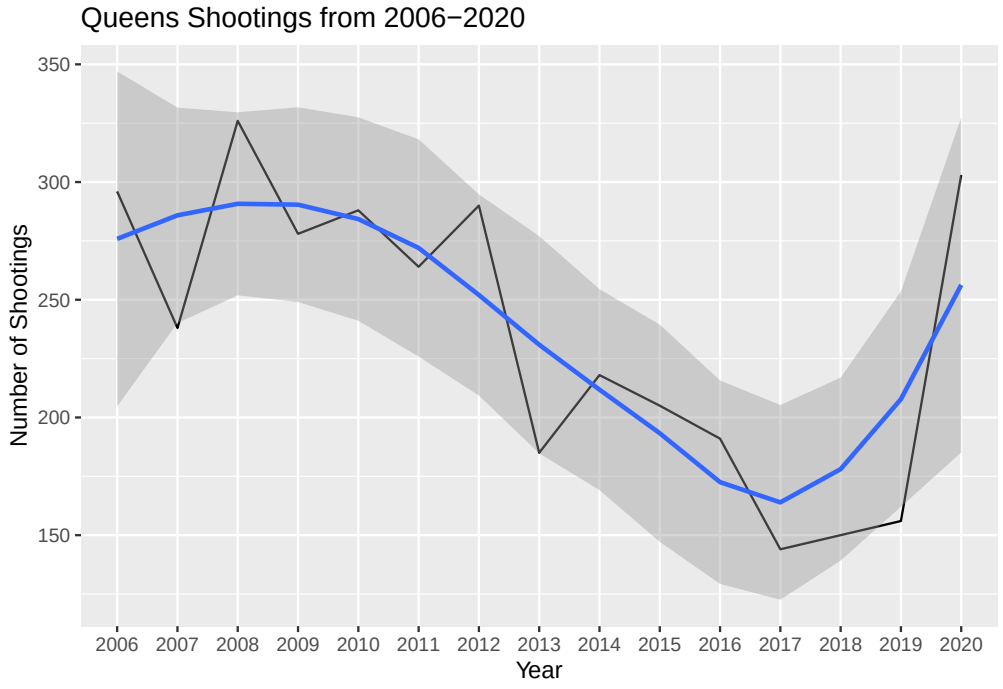


Repeating this for the other 4 Boroughs we obtain:

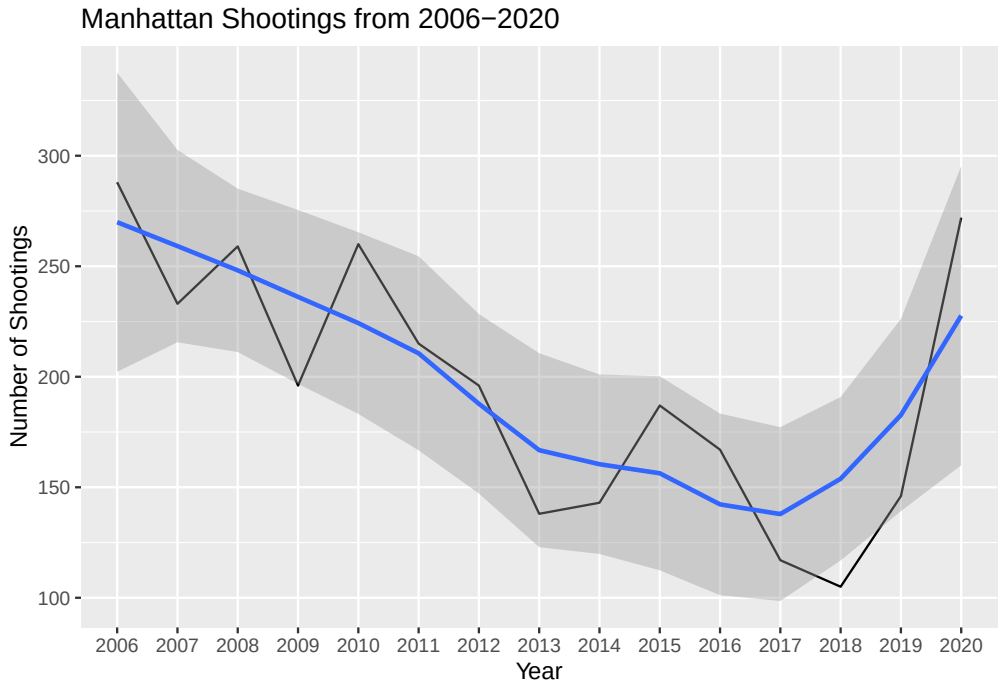
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



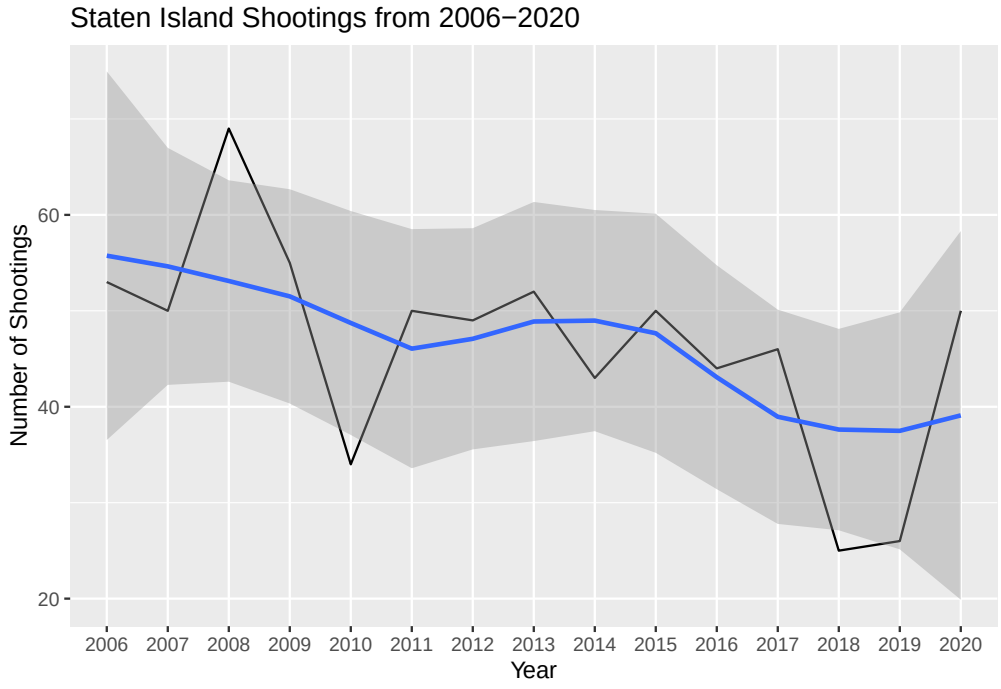

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Bias

Sources of bias include levels of personal bias and possible bias in the data collected. Living in NYC for the last 27 years inherently makes me biased for this data set since I live in the area this data was collected from. This bias is impossible to eliminate without working with a different data set.

Some other sources of bias in this data set could just be the way the Police System is set up in New York City. Boroughs with more police stations or areas with more density could have more shooting reports. Cases where these incidents are not recorded could be more prevalent in Boroughs which are less dense. This bias was slightly mitigated by looking at the overall Borough populations. Going further you can calculate population density and police station density and use these numbers to better frame the results.

Conclusion

After looking at the data and the graphs, some additional information was also researched. In order to get a better idea of the relationship of the populations of each Borough, the Census data was looked at to find the reported populations of each NYC Borough. The data can be found at https://www.citypopulation.de/en/usa/newyorkcity/

- Population Census 2020-04-01
    1. Brooklyn - 2,736,074
    2. Queens - 2,405,464
    3. Manhattan - 1,694,251
    4. Bronx - 1,472,654
    5. Staten Island - 495,747

This data shows us that while Brooklyn has the highest population and also the highest incidents of shooting, it still has almost 3 times more incidents than Queens, which has about 87% the population of Brooklyn. The rate of incidents is also high for the Bronx when compared to the other Boroughs.

When looking at the Borough data individually, there are some interesting patterns that emerge. Every Borough seems to follow the same trend, which is that the rate of shooting incidents, from 2006-2020 follows a downward trend until 2020. For the year 2020, every Borough had a major uptick in incidents. A likely reason for this would be due to the COVID-19 pandemic.

Thank you for reading!

```
sessionInfo()
```

```
## R version 4.1.2 (2021-11-01)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.3 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/atlas/libblas.so.3.10.3
## LAPACK: /usr/lib/x86_64-linux-gnu/atlas/liblapack.so.3.10.3
##
## locale:
##  [1] LC_CTYPE=C.UTF-8       LC_NUMERIC=C           LC_TIME=C.UTF-8
##  [4] LC_COLLATE=C.UTF-8     LC_MONETARY=C.UTF-8    LC_MESSAGES=C.UTF-8
##  [7] LC_PAPER=C.UTF-8       LC_NAME=C              LC_ADDRESS=C
## [10] LC_TELEPHONE=C         LC_MEASUREMENT=C.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] reshape2_1.4.4  lubridate_1.8.0 forcats_0.5.1   stringr_1.4.0
```

```
##  [5] dplyr_1.0.7     purrr_0.3.4     readr_2.1.2     tidyr_1.2.0
##  [9] tibble_3.1.6    ggplot2_3.3.5   tidyverse_1.3.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.8       lattice_0.20-45  assertthat_0.2.1 digest_0.6.29
##  [5] utf8_1.2.2       R6_2.5.1         cellranger_1.1.0 plyr_1.8.6
##  [9] backports_1.4.1  reprex_2.0.1     evaluate_0.14    highr_0.9
## [13] httr_1.4.2       pillar_1.7.0     rlang_1.0.1      curl_4.3.2
## [17] readxl_1.3.1     rstudioapi_0.13  Matrix_1.3-4     rmarkdown_2.11
## [21] splines_4.1.2    labeling_0.4.2   bit_4.0.4        munsell_0.5.0
## [25] broom_0.7.12     compiler_4.1.2   modelr_0.1.8     xfun_0.29
## [29] pkgconfig_2.0.3  mgcv_1.8-38      htmltools_0.5.2  tidyselect_1.1.1
## [33] fansi_1.0.2      crayon_1.4.2     tzdb_0.2.0       dbplyr_2.1.1
## [37] withr_2.4.3      grid_4.1.2       nlme_3.1-153     jsonlite_1.7.3
## [41] gtable_0.3.0     lifecycle_1.0.1  DBI_1.1.2        magrittr_2.0.2
## [45] scales_1.1.1     cli_3.1.1        stringi_1.7.6    vroom_1.5.7
## [49] farver_2.1.0     fs_1.5.2         xml2_1.3.3       ellipsis_0.3.2
## [53] generics_0.1.2   vctrs_0.3.8      tools_4.1.2      bit64_4.0.5
## [57] glue_1.6.1       hms_1.1.1        parallel_4.1.2   fastmap_1.1.0
## [61] yaml_2.2.2       colorspace_2.0-2 rvest_1.0.2      knitr_1.37
## [65] haven_2.4.3
```