Peer-graded Assignment: NYPD Shooting Incident Data Report

Mel Delgado

2024-03-01

Peer-graded Assignment: NYPD Shooting Incident Data Report

Course: DTSA-5301, Data Science as a Field

Author: Mel Delgado

Setup chunk - Add tidyverse and other packages

Before we get started, we add the necessary tools for our analysis.

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(lubridate)
library(ggplot2)
```

Read the data

The source of dataset for our analysis is available on data.gov.

This project calls for searching for the term *NYPD Shooting Incident Data (Historic)* and selecting the `.csv` file.

Store the base URL as a variable named `base_url` and the file named `rows.csv`

```
base_url <- "https://data.cityofnewyork.us/api/views/833y-fsy8/"

file_name <- c("rows.csv")

url <- str_c(base_url, file_name)
```

Store the comma separated values in a variable named `incidents` using `read_csv()`

```
incidents <- read_csv(url)
```

```
## Rows: 27312 Columns: 21
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (12): OCCUR_DATE, BORO, LOC_OF_OCCUR_DESC, LOC_CLASSFCTN_DESC, LOCATION...
## dbl   (7): INCIDENT_KEY, PRECINCT, JURISDICTION_CODE, X_COORD_CD, Y_COORD_CD...
## lgl   (1): STATISTICAL_MURDER_FLAG
## time  (1): OCCUR_TIME
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Tidy and transform data

To Tidy and transform the data, I only pull in the columns needed for the analysis I have in mind and store it in a variable named `shooting_incidents`. It stores the columns named `OCCUR_DATE`, `OCCUR_TIME`, `BORO`, and `STATISTICAL_MURDER_FLAG`. I would like the column name of `BORO` to change to `Borough`. To make the replacement, called `mutate()` and `select ()` to create a new column named `Borough` remove the column named `BORO`.

I noticed that `OCCUR_DATE` is of type `chr` which will make is not the correct type so I'll change it to type `mdy` by calling `mutate`.

```
shooting_incidents <- incidents %>%
    select(c(OCCUR_DATE, OCCUR_TIME, BORO, STATISTICAL_MURDER_FLAG)) %>%
    mutate(OCCUR_DATE = mdy(OCCUR_DATE),
       OCCUR_TIME = hms(OCCUR_TIME),
       STATISTICAL_MURDER_FLAG = as.logical(STATISTICAL_MURDER_FLAG),
       Borough = BORO,
       Year = year(OCCUR_DATE)) %>%
    select(-c(BORO))
```

Next, I would like to know the number of deaths that occurred as a result of a shooting and store the information in a variable named `shooting_deaths`. To accomplish this, I used `filter` to only select values of `STATISTICAL_MURDER_FLAG` that are equal to `TRUE`.

```
shooting_deaths <- shooting_incidents %>%
        select(c(OCCUR_DATE, Borough, STATISTICAL_MURDER_FLAG)) %>%
        filter(STATISTICAL_MURDER_FLAG == TRUE)

non_fatal_shootings <- shooting_incidents %>%
        select(c(OCCUR_DATE, Borough, STATISTICAL_MURDER_FLAG)) %>%
        filter(STATISTICAL_MURDER_FLAG == FALSE)

head(shooting_deaths)
```

```
## # A tibble: 6 x 3
##   OCCUR_DATE Borough   STATISTICAL_MURDER_FLAG
##   <date>     <chr>     <lgl>
## 1 2015-11-21 QUEENS    TRUE
## 2 2009-02-19 BRONX     TRUE
## 3 2020-10-21 BROOKLYN  TRUE
## 4 2010-03-08 BROOKLYN  TRUE
```

```
## 5 2010-07-27 MANHATTAN TRUE
## 6 2015-02-01 MANHATTAN TRUE
```

```
head(non_fatal_shootings)
```

```
## # A tibble: 6 x 3
##   OCCUR_DATE Borough STATISTICAL_MURDER_FLAG
##   <date>     <chr>   <lgl>
## 1 2021-05-27 QUEENS  FALSE
## 2 2014-06-27 BRONX   FALSE
## 3 2015-10-09 BRONX   FALSE
## 4 2012-06-17 QUEENS  FALSE
## 5 2012-02-05 QUEENS  FALSE
## 6 2012-08-26 QUEENS  FALSE
```

To be sure the data I am not missing data needed for my analysis, I call `View()` to see the tables of data as I would expect as well as `head()` to only see the first 6 lines of the data stored in a variable. In my case, I have the data needed for the analysis I have in mind.

If it were missing, I would return to the step where I read data in (perhaps from an additional source) and decide if I would need to join it to my existing set of data.

Modeling the Data

To model the data, I would like to understand the murder rate by New York City borough. In other words, I would like to know what boro has the highest number of deaths occurring from shootings.

My approach is to group the data by borough and the summation of `STATISTICAL_MURDER_FLAG` equals `TRUE` and store the grouping in a variable named `shooting_deaths_by_boro`.

```
shooting_deaths_by_boro <- shooting_deaths %>%
        group_by(Borough) %>%
        summarize(number_of_shooting_deaths = sum(STATISTICAL_MURDER_FLAG == TRUE))

non_fatal_shootings_by_boro <- non_fatal_shootings %>%
        group_by(Borough) %>%
        summarize(non_fatal_shootings = sum(STATISTICAL_MURDER_FLAG == FALSE))
```

Visualizing the Data

To visualize this data and analysis to tell a story, I used a bar chart with the number of shootings resulting in death in the y-axis and the borough on the x-axis. To create the bar charts I had in mind, I called `ggplot()` and `geom_bar()` to create a bar chart. I added descriptive text using `labs()` which is short for *labels* to include better descriptions to the chart.

```
shooting_incidents %>%
    ggplot(aes(x = Borough, fill = Borough)) +
    geom_bar() +
    labs(title = "Number of Shootings Per New York City Borough",
        subtitle = "2006 - 2022", x = "NYC Borough",
        y = "Number of Shootings",
        caption = "Illustration 1")
```

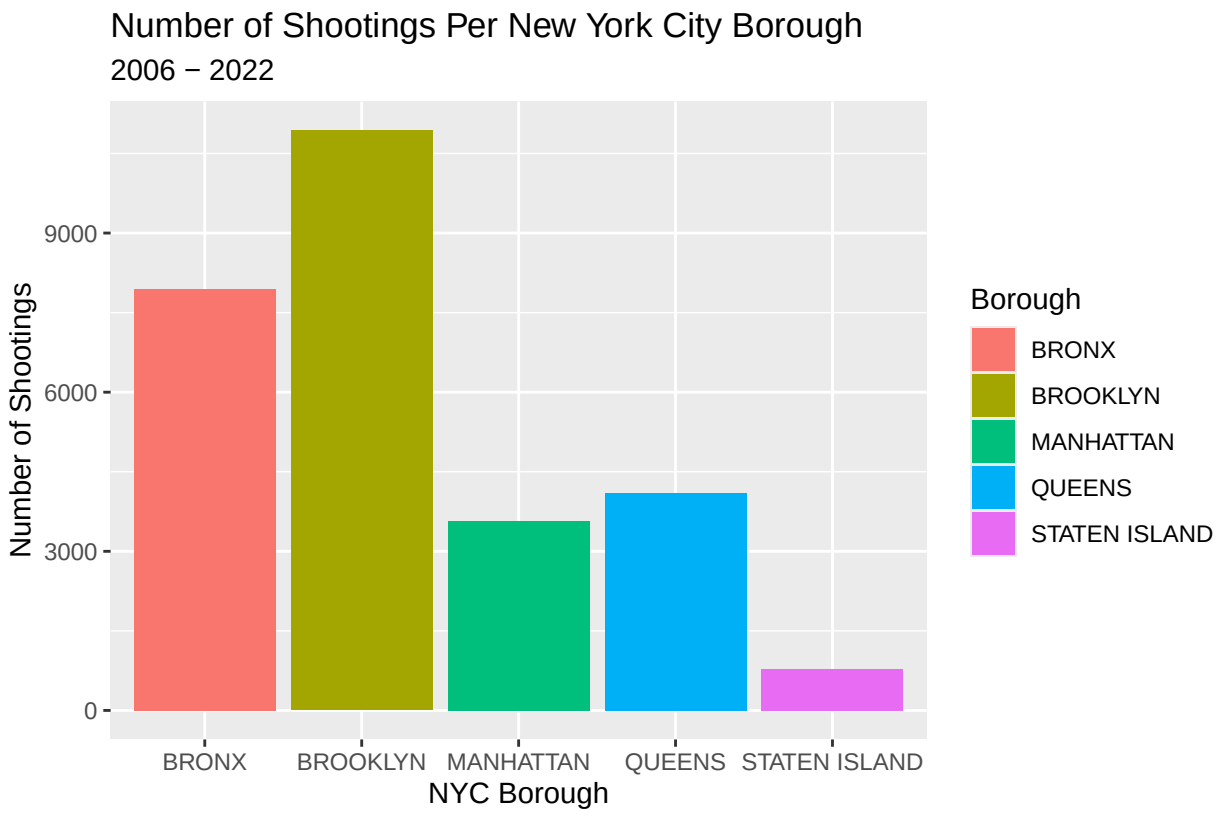

Illustration 1

I called `ggplot()` and setup the aesthetic with `aes()` and the parameters to identify the data for the x and y axis. Then, I called `geom_bar()` as a means of creating a bar chart to visualize the number of deadly shootings per New York City borough.

```
shooting_deaths_by_boro %>%
    ggplot(aes(x = Borough, y = number_of_shooting_deaths)) +
    geom_bar(stat = "identity", fill = "RED") +
    labs(title = "Number of Deaths Per New York City Borough",
        subtitle = "2006 - 2022", x = "New York City Borough",
        y = "Number of Deaths",
        caption = "Illustration 2")
```

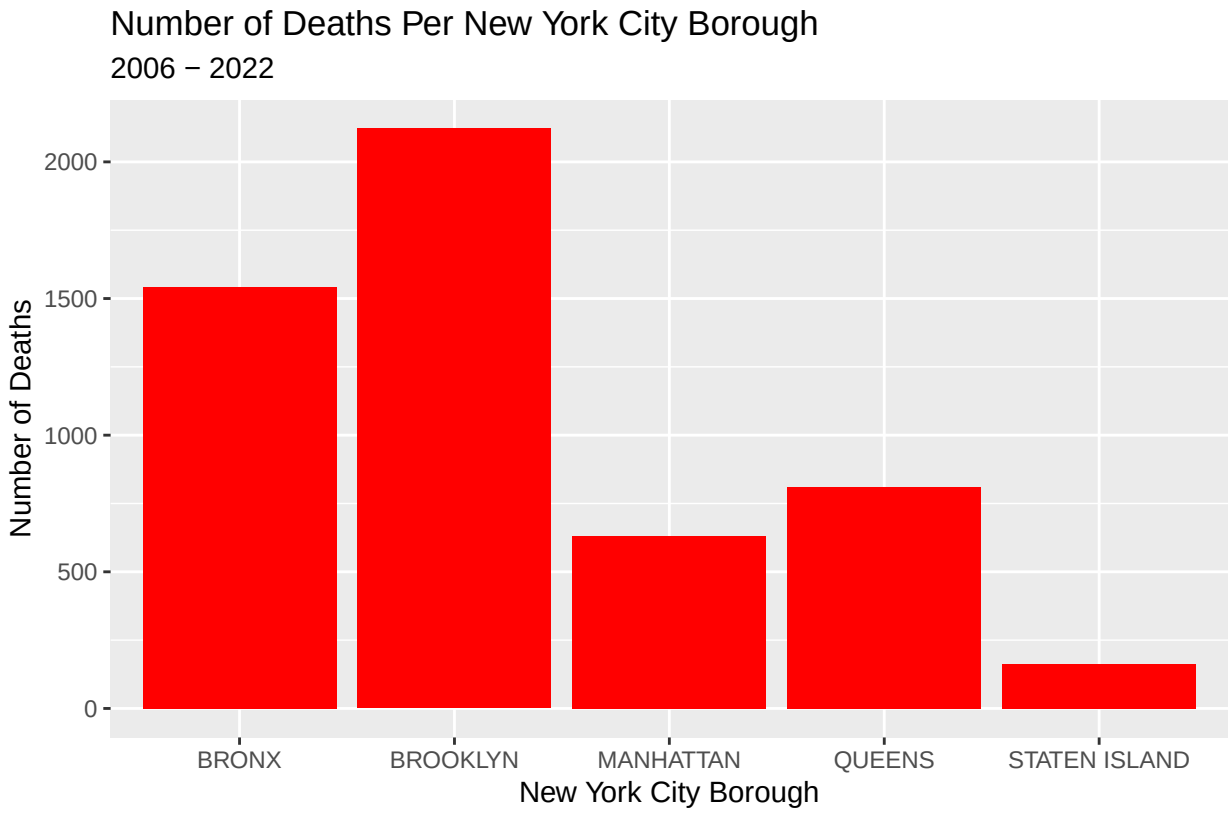

Illustration 2

After visualizing the data using two different bar charts, it raises additional questions to investigate. For example, I could continue the cycle of analyzing and modeling many more times to establish a likely percentage of deaths occurring as the result of shootings, the average age difference between the perpetrator and victim, the hour of the day in which fatal shootings are most likely to occur, and many more.

Bias Identification

My personal bias was reflected in my choice of analyzing the occurrence of shootings by borough. In doing so, it reflects how my personal bias is to associate crime with the neighborhoods where they occur. My thought was to identify the most dangerous borough in the data.

To mitigate the original bias, I looked to the data to show if shootings were fatal because I wanted to understand if shootings were a result of a perpetrator trying to harm a victim or if the shooter intended to cause death. The data suggests the highest death rate for shootings were Brooklyn and the Bronx and the same was true for the charts showing the number of fatalities that occurred. As such, I could only draw the conclusion that the number of fatalities were consistent with the number of shootings. In other words, shootings are likely to occur in death no matter where the shooting occurs.