311 NYPD Data: Understanding Complaint Patterns and NYPD Operations

Authors:

- Tim Kartawijaya (tak2151)
- Nico Winata (nw2408)
- Charlene Luo (cl3788)
- Donghan He (dh2925)

Link to Github Repo.

1. Introduction

1a. Motivation

NYC Open Data is a tool provided by the New York City government as part of an initiative for the city to be more transparent in their operations. Through the website, various agencies have made their data public, from the NYPD to the Fire Department, in order to "improve the accessibility, transparency, and accountability of City government" (NYC Open Data Website). From 311 calls to housing data, the NYC Open Data initiative has made it possible for data scientists to create analyses and tools that allow the public to understand how NYC is run day to day.

For this project, we decided to take on 311 data provided by the aforementioned tool, since as NYC residents we are curious how the city functions. Furthermore, we have an interest to give back to the city by developing insights and tools that agencies can use to better operations within the city. Our focus in on 311 NYPD data, since we concluded that 311 calls to the police are the best indicator of the pain points of NYC life. We approached our analysis with use cases in mind since we want to create analyses that are actionable and not simply for the aesthetic. We organize our use cases according to the kind of insights we can derive from our data:

1. **Operations**: How well the NYPD responds to 311 complaints

Use cases:

- For NYPD, understanding where and when to allocate resources to better service 311 complaints.
- For the public, provide accountability, monitoring whether NYPD is responding 311 data fairly or not.

2. **Incident Patterns**: General patterns of the different kind of complaints in regards to time and space.

Use cases:

- For NYPD, identifying complaint patterns to reveal underlying causes i.e. complaint hotspots.
- For policy-makers, identify complaint patterns to equip them in decision making (e.g. many complaints on traffic in one area, fix roads/traffic lights there).

1b. Main Question:

Using the above use cases as our guide, we can then construct the main questions for our analyses:

1. **Operations:**

- What areas, time of day, incident types are handled most/least efficiently? What are the probable causes of this level of efficiency? (e.g. the distance of police station affects processing time?)

- Sometimes some calls turn out to be not police related or false alarms. At what neighborhoods/times do these false alarms occur? How can we prevent them?

2. **Incident Patterns:**

- At which locations/time of day do specific incident types occur the most? Most notable patterns? What are the probable causes of these patterns?
- How do incidents change throughout the day and year? Are there are problems that are consistent year-to-year that should be addressed?

1c. Team Contribution

1. Donghan He: Data preprocessing, enrichment, and engineering, Description of Data and Data Quality.
2. Nico Winata: Temporal analysis, extreme event analysis, false alarm analysis, Executive summary.
3. Charlene Luo: Geographical analysis, processing time vs. distance from station, Executive summary.
4. Tim Kartawijaya: Motivation, Main Questions, Year-to-Year Analysis, Interactive Component, Team Coordinator, Git.

2. Description of Data

Source

We downloaded the 311 data from the NYC open data website [https://data.cityofnewyork.us/Social-Services/311-Service-Requests-from-2010-to-Present/erm2-nwe9], and the police precinct data from the same website [https://data.cityofnewyork.us/Public-Safety/Police-Precincts/78dh-3ptz].

The original 311 data is collected automatically through the system of 311 request processing. The data starts from 2010 to present. A total of 41 columns and 19.3 millions of rows are present in the original data set, including mostly text/string data (33 out of 41 columns), some date time data (4 out of 41 columns) and some numerical data (4 out of 41 columns).

To enrich the data set, we also acquired the police precinct data from NYC open data. This dataset contains the boundaries of all 77 police precincts in New York City. The boundaries are encoded as the multi-polygon data type. Specifically, it includes vertices of the polygon that is a given police precinct, and each vertex is described using longitude and latitude.

Noteworthy Features

The features that we are interested in are: Created Date, Closed Date, Complaint Type, Descriptor, Location Type, Incident Zip, Due Date, Resolution Description, Resolution Action Updated Date, Latitude, Longitude, Borough, and City. The created date and closed date are used to engineer the process time feature (detail given below). The latitude/longitude data is used later to build a measure of how close the caller is to the police department. Moreover, the created date and the incident zip are the most important features.

3. Analysis of Data Quality

First, let's load the tidyverse library to perform analyses.

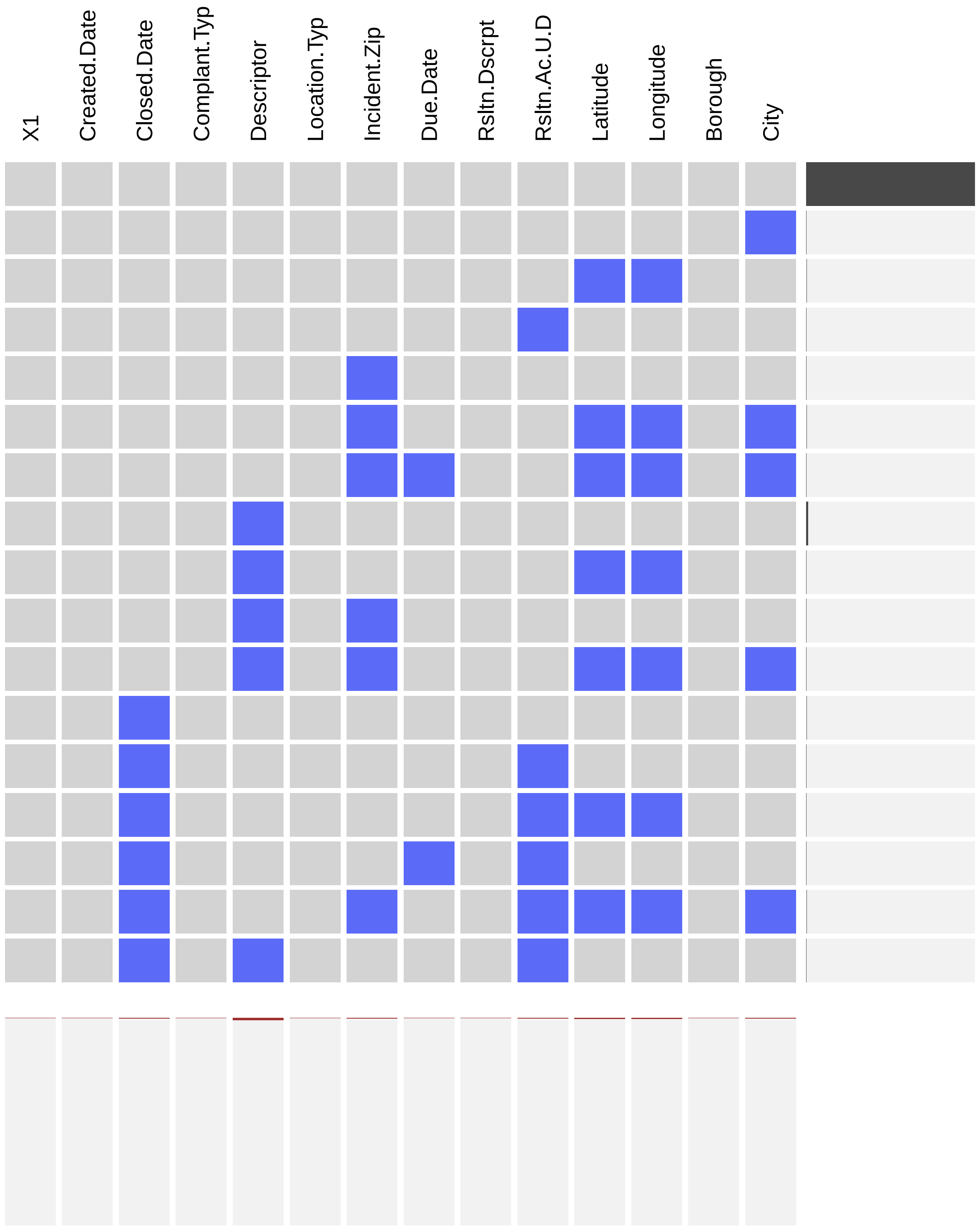
```r
library(tidyverse)
library(extracat)
library(lubridate)
library(choroplethrZip)
```

The data is of high quality before even cleaning. Only 1.7125% of data is missing, thus we can simply drop those data points with missing values. Below is a visualization of missing value patterns in the dataset to be more detailed.

```r
df_raw = read_csv("relevant_unprocessed_311_data.csv")
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```r
df_raw[df_raw=='n/a'] <-NA
visna(df_raw)
```



4. Main Analysis

Next, let's load the preprocessed data and take a peek at the data.

```
df = read.csv("cleaned_311_data.csv")
```

4a. Data Preprocessing

Note that we implemented the below steps in a separate IPython notebook. You can find the mentioned notebook here. The results of the preprocessing is a CSV file called **cleaned_311_data.csv** which you can download here here.

We keep the following columns: Created Date, Closed Date, Complaint Type, Descriptor, Location Type, Incident Zip, Due Date, Resolution Description, Resolution Action Updated Date, Latitude, and Longitude.

First and foremost, we would like to see if we can impute any data. Unfortunately, as most of the data are text/string based, we can't impute the data without complex natural language modeling. Even for the date time missing values, we don't have straightforward ways of imputing them. For most of the incident zip/longitude-latitude data, ideally we can use one part to get a good approximation of the other (for example use the longitude-latitude pair to predict the zip); however, for most of the cases, both zip and longitude-latitude are missing concurrently. For the small-percentage of cases that only the incidents zips are missing, we use the 1-nearest-neighbor method to impute the zips using longitude-latitude data. Namely, we find the closest data (in terms of longitude/latitude) that has an observed zip code.

Secondly, we engineer two features. The first feature is police efficiency. This is done by calculating how long it took for the police department to process a request. Namely, we let (closed date - created date). Moreover, we remove extreme values (requests took longer than 3.5 days to complete), since they might be simply negligence to report a finished request.

The second feature describes police proximity. This feature is a bit harder to calculate. Firstly we get the police precinct data from NYC open data as described before. Since there are no complete data on the internet that supplies all 77 precincts' longitude latitude data, we approximate them using the multi-polygon's centroids. We calculate the centroid data the following way, which is simply the arithmetic mean of the vertices.

$$police\hat{\_}loc = \frac{1}{\#vertices}\sum_i vertex\_loc_i$$

And then, we are going to calculate the geodesic distance between each incident location and each centroid's location. Note that although we can naively use the Euclidean distance by calculating $(|long_{police} - long_{incident}|^2 + |lat_{police} - lat_{incident}|^2)^{0.5}$, this approach generate huge errors for areas that are more scattered (staten island for example). Hence, we introduce the following procedure to calculate the true geodesic distance:

$$R = 6371e3(m)$$

$$inc\_loc = (rad(long_{inc}), rad(lat_{inc}))$$

$$pol\_loc = (rad(long_{pol}), rad(lat_{pol}))$$

$$\delta = pol\_loc - inc\_loc$$

$$a = \sin(\delta[0]/2)^2 + \cos(inc\_loc[0]) * \cos(pol\_loc[0]) * \sin(\delta[1]/2)^2$$

$$geodesic(police, incident) = R * 2 * \arctan(\sqrt{a}/\sqrt{1-a})$$

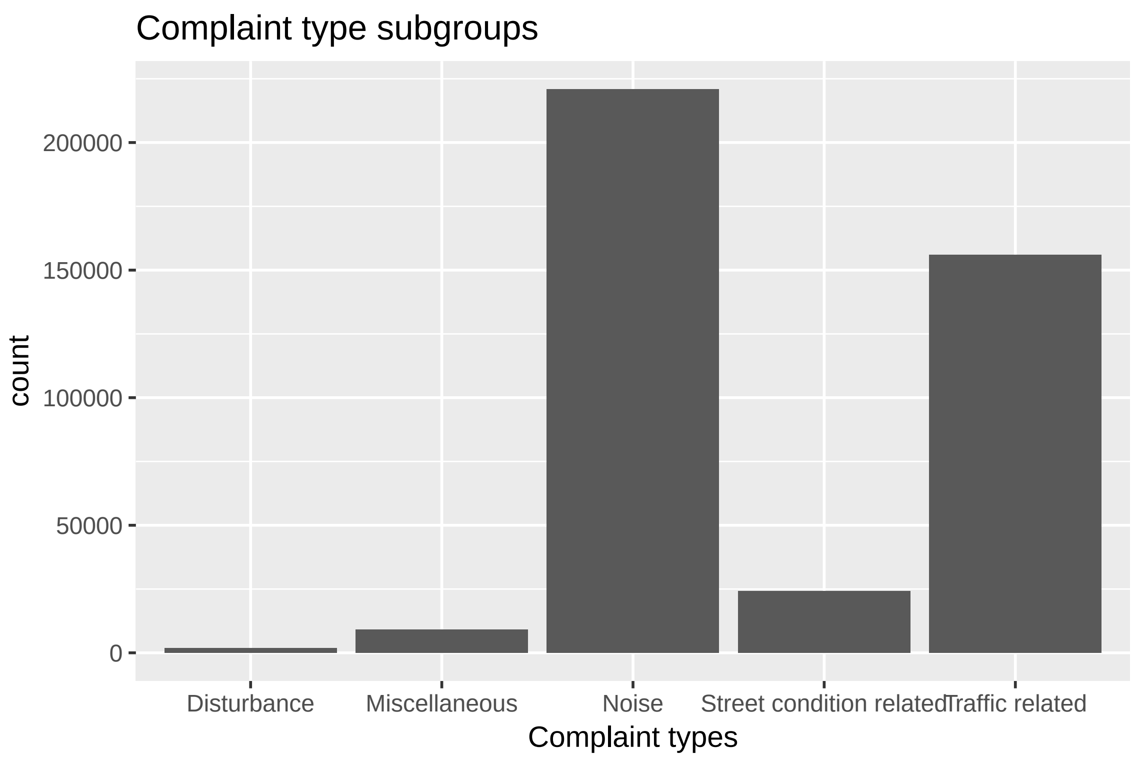
To contrast, we can compare the naive Euclidean distance to the accurate geodesic distance using the functions implemented accordingly in the python script. For example, plugging in (44,-77) and (45,-78), we get a 15 percent error ((true value-naive estimate)/true value), which is huge. The precise analytical error bound can be obtained using Taylor expansion on the geodesic distance to the first order. The functions are implemented in python and can be tested out using the ipython notebook.

After the distances to the police precinct centroids are computed, we use the 1-nearest-neighbor method to calculate the distance to police aid. The reasoning behind is that the police department will usually dispatch officers closest to the incident location. Instead of 1-nearest-neighbor, averaged k-nearest-neighbor distance can also be used (average of the top k closest police precinct centroids), and the justification could be that the naive geodesic measure is not the best way to measure closeness. Manhattan's traffic is notoriously crowded and this might confound how fast the officers get to the location, hence k nearest police stations are almost equally likely to be dispatched. Such speculation is valid, but without data to support such speculation, the 1-nearest-neighbor method should be sufficient.

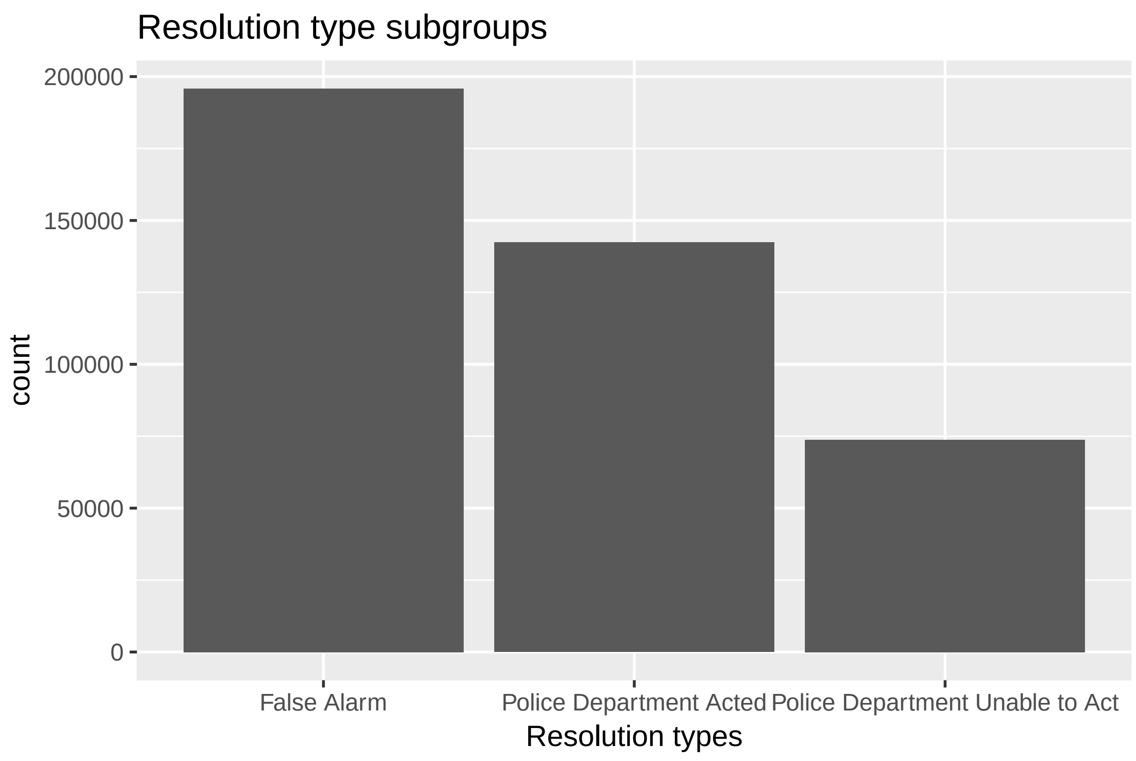
Furthermore, the complaint types and the resolution types column can be way too low-leveled. Human vision needs some sort of dimension reduction first to process messy data. There are many ways to do dimension reduction, one of the more popular ways is clustering. Since we are not allowed to do complicated natural language processing for this project, we do manual clustering. We split the complaint types and the resolution types in the following way. Complaint types are split into five groups: noise, street condition related, disturbance, traffic-related and miscellaneous. This subgrouping is based on the reasoning that viewers of this report are renters or buyers, and these categories are the most relevant to them. On the other side, we try to cluster how the police department resolves the issue, so we can see the area heterogeneity in police efficiency which would be crucial to renters and buyers. Therefore the resolution types are split into the following: police department acted, police department unable to act and false alarm.

```r
df_cl = read_csv("cleaned_311_data.csv") #read in tidyverse version
```

```r
g1 = ggplot(df_cl, aes(x = factor(`Complaint Subgroups`))) +
  geom_bar() +
  xlab('Complaint types') +
  ggtitle("Complaint type subgroups")
g2 = ggplot(df_cl, aes(x = factor(`Resolution Subgroups`))) +
  geom_bar() +
  xlab('Resolution types') +
  ggtitle("Resolution type subgroups")
g1
```

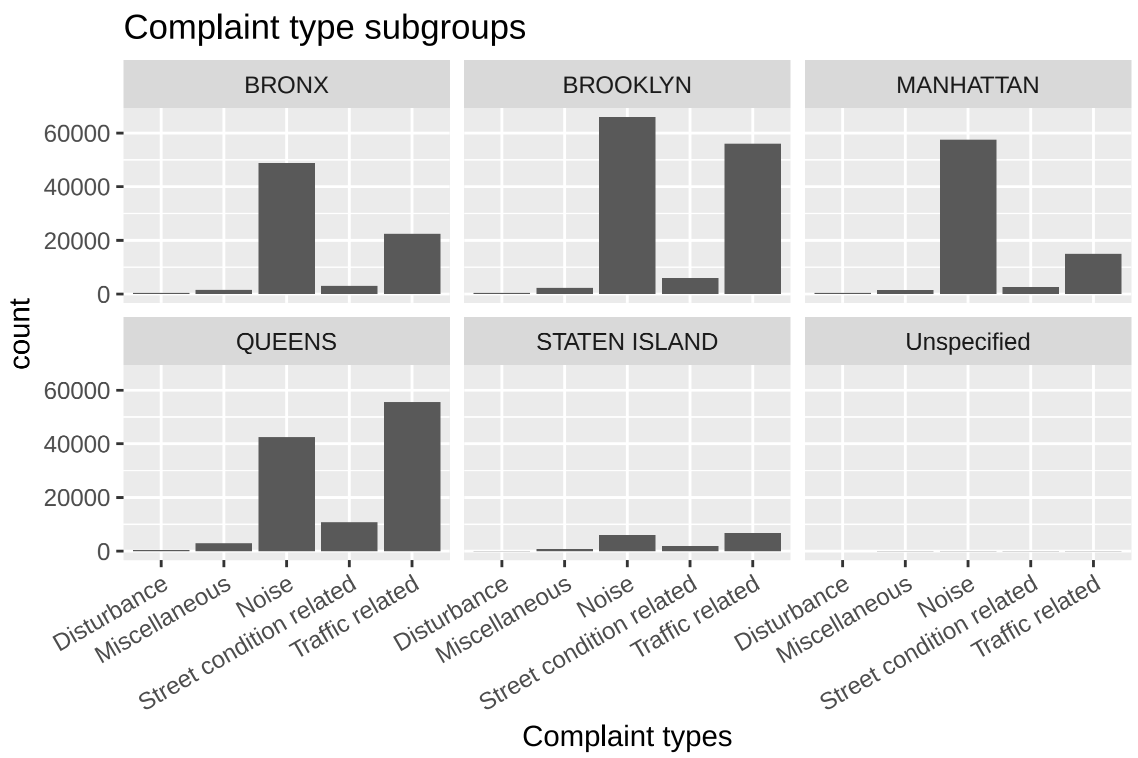


```r
g2
```



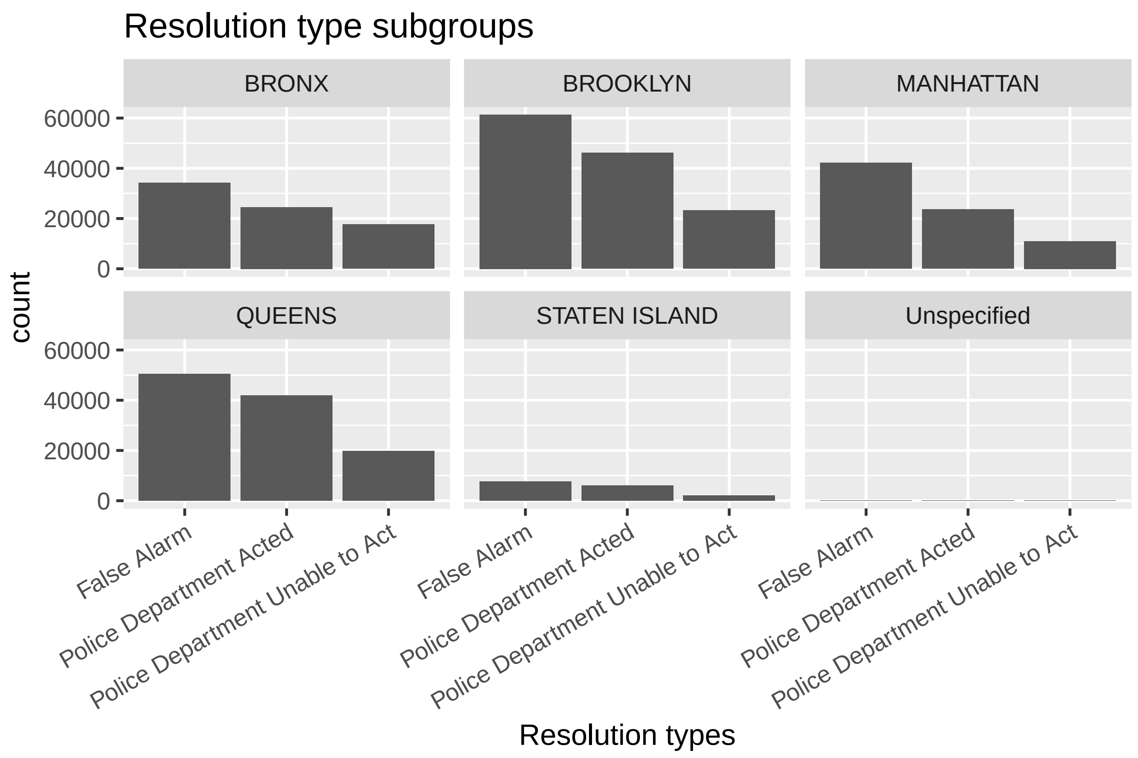
As we can see from above, noise and traffic are the top two concerns for the New Yorkers. Further we facet this plot by borough. As suspected, this phenomenon is quite robust across different regions, thus it is worth further invetigation. We also plot the faceted bar graph (faceting by Borough) for the subgrouped resolution type column. As we can see, most of them are false alarms.

```r
g3 = ggplot(df_cl, aes(x = factor(`Complaint Subgroups`))) +
  facet_wrap(~Borough) +
  geom_bar() +
  xlab('Complaint types') +
  ggtitle("Complaint type subgroups") +
  theme(axis.text.x = element_text(angle = 30, hjust = 1))
g4 = ggplot(df_cl, aes(x = factor(`Resolution Subgroups`))) +
  facet_wrap(~Borough) +
  geom_bar() +
  xlab('Resolution types') +
  ggtitle("Resolution type subgroups") +
  theme(axis.text.x = element_text(angle = 30, hjust = 1))
g3
```



Complaint type subgroups
BRONX
BROOKLYN
MANHATTAN
QUEENS
STATEN ISLAND
Unspecified
count
60000
40000
20000
0
Disturbance
Miscellaneous
Noise
Street condition related
Traffic related
Complaint types

```
g4
```



Resolution type subgroups
BRONX
BROOKLYN
MANHATTAN
QUEENS
STATEN ISLAND
Unspecified
count
60000
40000
20000
0
False Alarm
Police Department Acted
Police Department Unable to Act
Resolution types

4b. NYPD Processing Time / Efficiency

```r
df$Closed.Date = strptime(df$Closed.Date,format = "%m/%d/%Y %I:%M:%S %p")
df$Created.Date = strptime(df$Created.Date,format = "%m/%d/%Y %I:%M:%S %p")
df$Process.Time = df$Closed.Date - df$Created.Date
df$hour = hour(df$Created.Date)
```

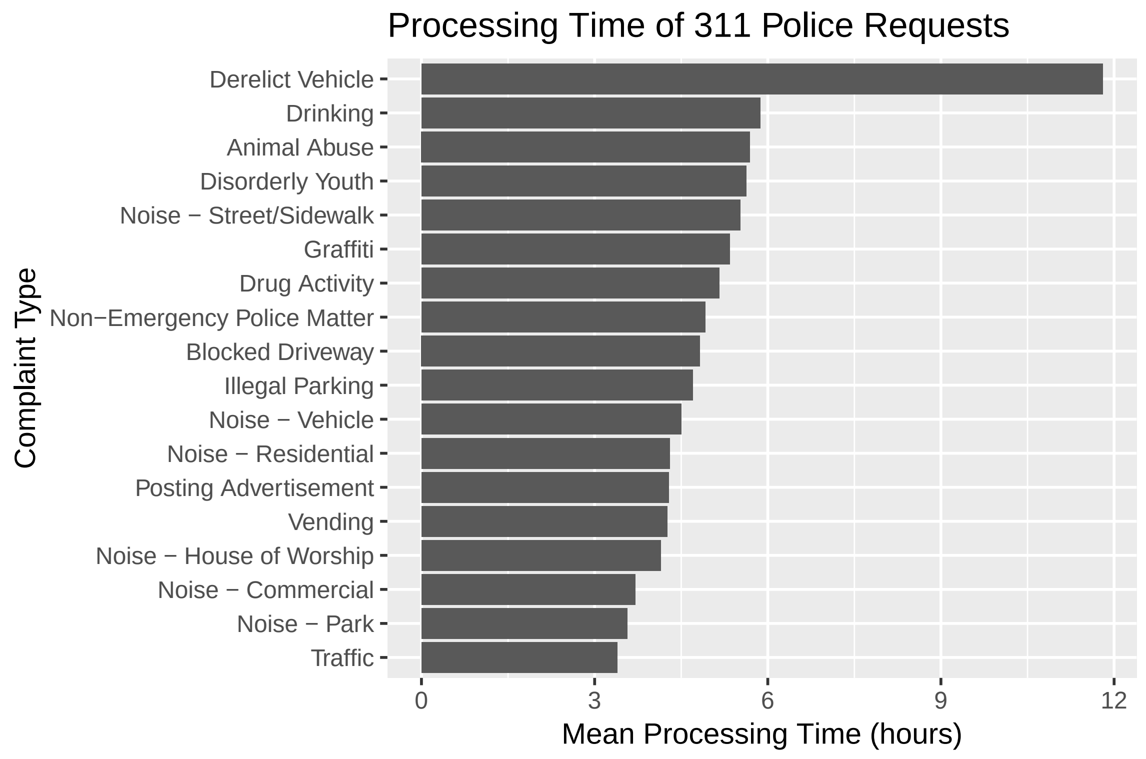
```r
df1 = df[df$Process.Time>=5000,]
```

One of the areas we want to investigate is efficiency of 311 request processing. It is of great interest to us whether we can find any evidence of structural ineffiencies in how 311 requests are handled from visualizing the dataset we have.

Patterns by Complaint Type

We first look at processing time by complaint type.

```r
mean_time <- df[is.na(df1$Complaint.Type)==FALSE,c(4,18)] %>%
  group_by(Complaint.Type) %>%
  summarize(mean.process.time = mean(Process.Time)) %>%
  arrange(mean.process.time)

mean_time$mean.process.time = mean_time$mean.process.time/60
mean_time = mean_time[is.na(mean_time$Complaint.Type) ==FALSE,]
ggplot(mean_time,aes(x=reorder(Complaint.Type,mean.process.time), weight=mean.process.time)) +
  geom_bar() +
  scale_x_discrete(name = "Complaint Type") +
  scale_y_continuous(name = "Mean Processing Time (hours)") +
  coord_flip() +
  ggtitle("Processing Time of 311 Police Requests")
```



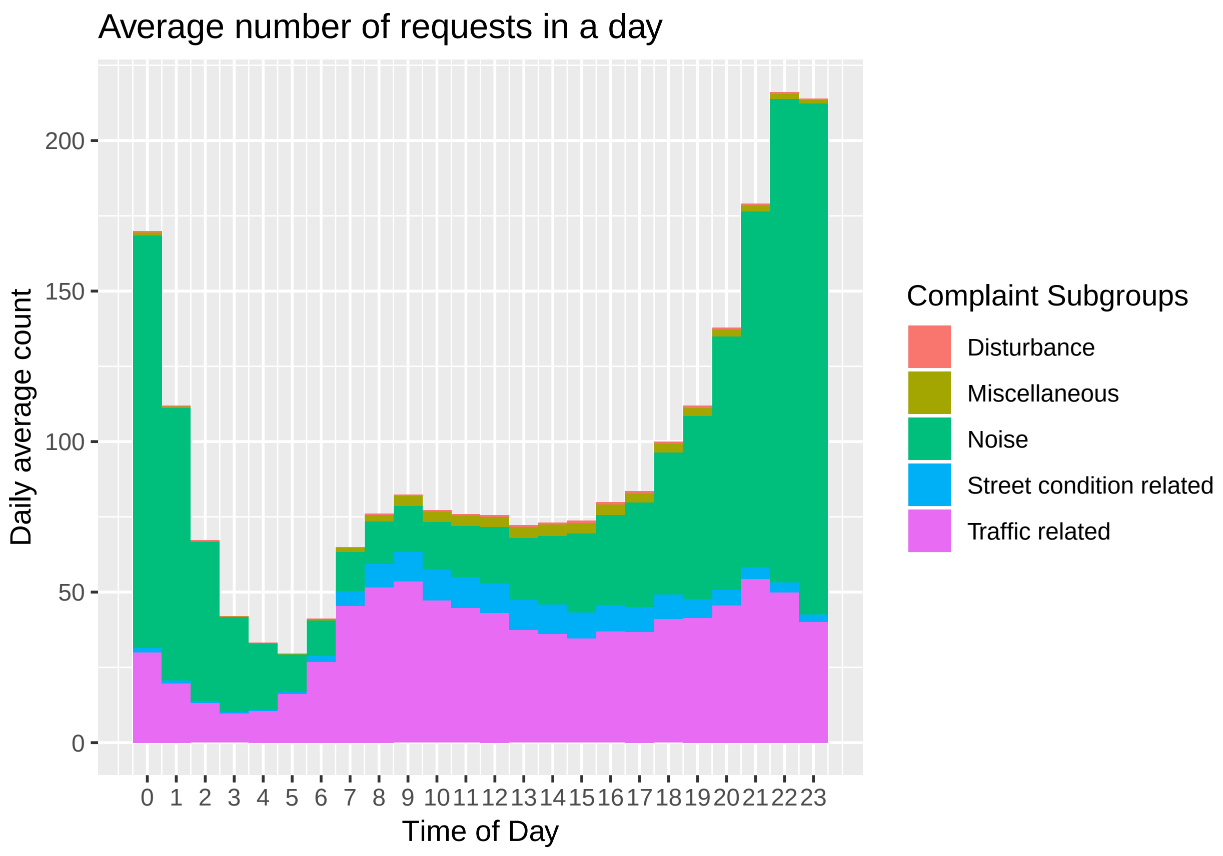
In general the mean processing times of different complaint types are sufficiently evenly spread, apart from the few types on the extreme top. Derelict vehicles requests take the longest, while traffic requests take the shortest, which agrees with one's common sense. Most requests take an average of three to six hours to complete. We will keep this in mind when assessing inefficiencies later on.

Patterns by Time

Time of Day

The distribution of requests and their complaint type is shown below. The x-axis shows the time of day. What we can see from the graph below is that the biggest complaint group is noise-related. These type of requests peak at around midnight, causing a spike the average count of requests around midnight. We also see that generally there is an inverse relationship between noise complaints and street condition complaints (and to a lesser extent, traffic-related complaints). This agrees with common sense - noise complaints are generally residential and their occurrence happens at times when people are at home. Street condition requests contain mostly requests about the removal of derelict vehicles - most of the request times occur during the day presumably because it is a non-urgent request that takes time to finish.

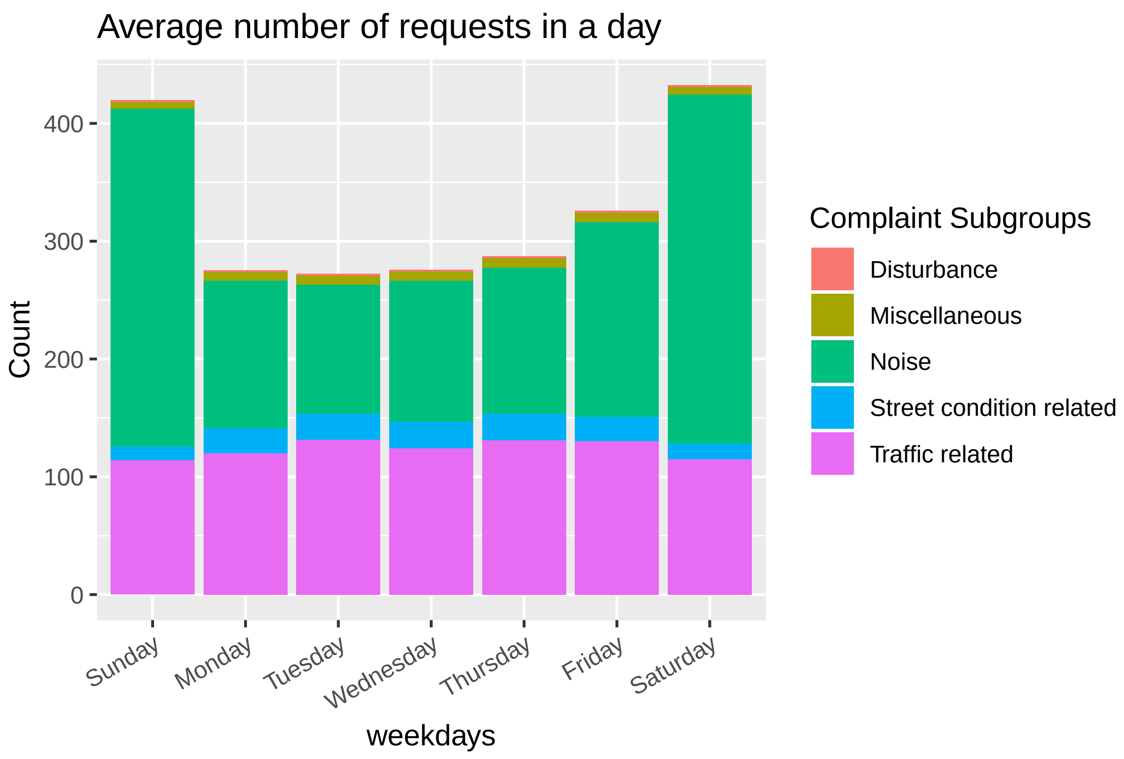
```r
ggplot(df,aes(x=hour,y=..count../180,fill=Complaint.Subgroups)) +
  geom_histogram(binwidth=1) +
  scale_x_continuous ("Time of Day", breaks= 0:23, label = 0:23) +
  scale_y_continuous("Daily average count") +
  labs(fill = "Complaint Subgroups") +
  ggtitle("Average number of requests in a day")
```



Day of Week

Below is the distribution of requests by day of the week. We see a similar pattern - traffic and street condition requests are inversely correlated with noise. Noise requests normally occur at times when people are at home - Saturday and Sunday, and street condition requests occur during working days.

```
df$weekdays = strftime(df$Created.Date,'%A')
df$weekdays = factor(df$weekdays,levels=c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday","Friday
ggplot(df,aes(x=weekdays,y=..count../180,fill=Complaint.Subgroups)) +
  geom_bar()+
  scale_y_continuous("Count") +
  ggtitle("Average number of requests in a day") +
  labs(fill = "Complaint Subgroups") +
  theme(axis.text.x = element_text(angle = 30, hjust = 1))
```

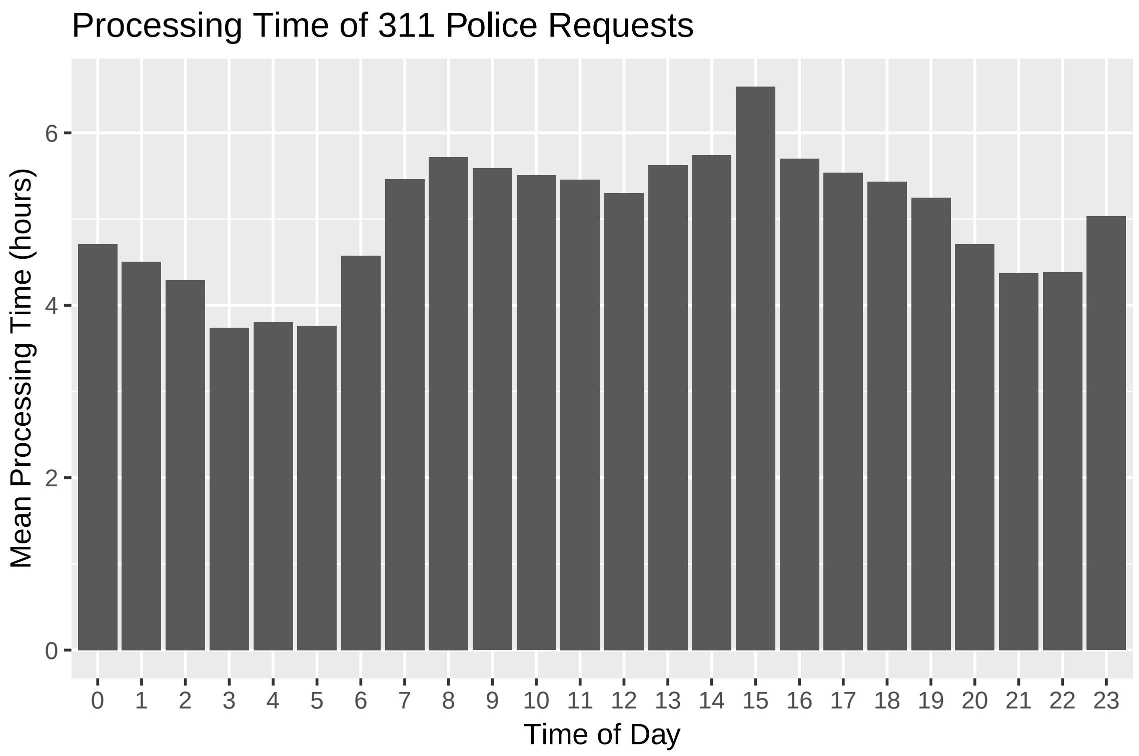


We plot the processing times by hour of the day. We see that the general shape of the distribution is that the processing times are higher during the day than in the evening. Processing times are lowest in the early hours of the morning.

Can we attribute this to an inefficiency in processing requests during the day? Are police stations understaffed during the day than in the evening? Keeping in mind that derelict vehicle requests are received mostly during the day, we think the answer is no. The increase in processing times during the day is due to the type of complaints received - noise complaints received at night are faster to resolve than other types of request, as we have shown earlier. If anything, the only inefficiency we can find is the small peak in processing time at 11 pm and 12 am compared to its surroundings. Given the similar type of request composition, we think that the peak is caused by police stations experiencing high request volumes at these times. However, the processing time difference is small and we do not think this is a significant operational issue.

```
mean_time <- df[is.na(df1$Complaint.Type)==FALSE,c(18,19)] %>% group_by(hour) %>% summarize(mean.proces
mean_time$mean.process.time = mean_time$mean.process.time/60 #to hour
mean_time = mean_time[is.na(mean_time$hour) ==FALSE,]
ggplot(mean_time,aes(x=reorder(hour,hour), weight=mean.process.time)) +
  geom_bar() +
  scale_x_discrete(name = "Time of Day") +
  scale_y_continuous(name = "Mean Processing Time (hours)") +
  ggtitle("Processing Time of 311 Police Requests")
```

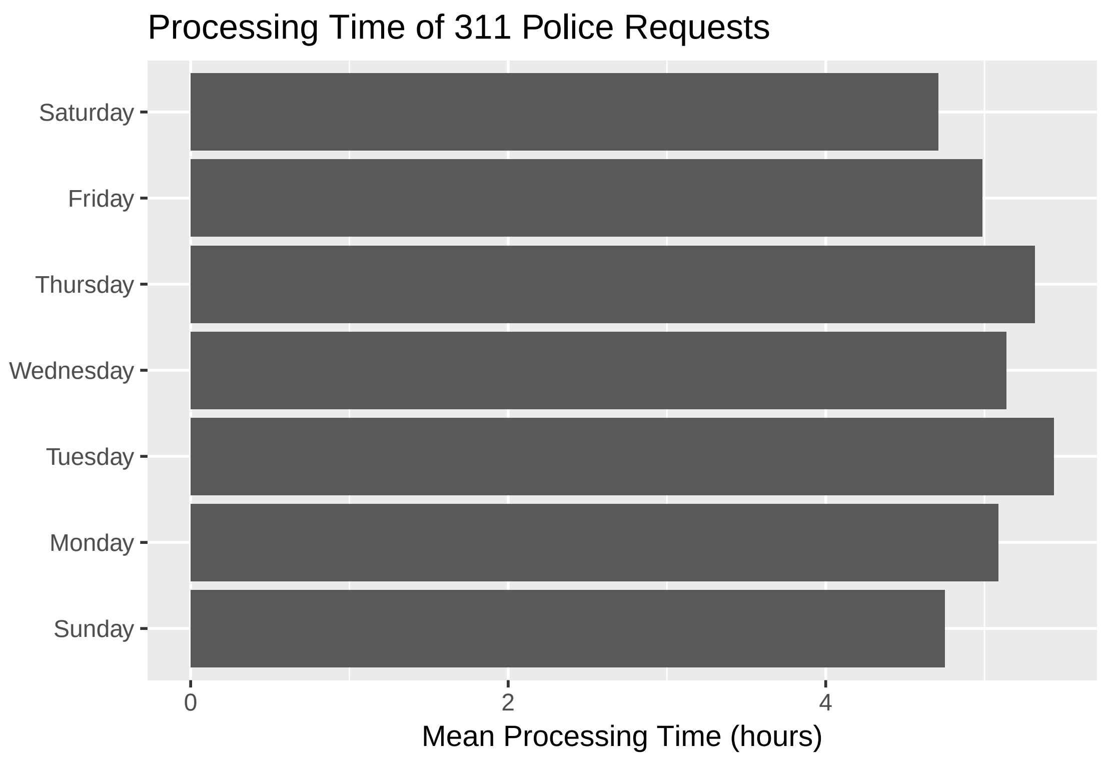
```
## Don't know how to automatically pick scale for object of type difftime. Defaulting to continuous.
```



The weekly data do not show anything we don't already know - days with more street condition requests have higher average request times and the weekends, where most of the complaints are noise-related, have lower processing times.

```
mean_time <- df[is.na(df$weekdays)==FALSE,c(18,20)] %>% group_by(weekdays) %>% summarize(mean.process.t
mean_time = mean_time[is.na(mean_time$weekdays) ==FALSE,]
mean_time$mean.process.time = mean_time$mean.process.time/60
ggplot(mean_time,aes(x=weekdays, weight=mean.process.time)) +
  geom_bar() +
  scale_x_discrete(name = "") +
  scale_y_continuous(name = "Mean Processing Time (hours)") +
  coord_flip() +
  ggtitle("Processing Time of 311 Police Requests")
```

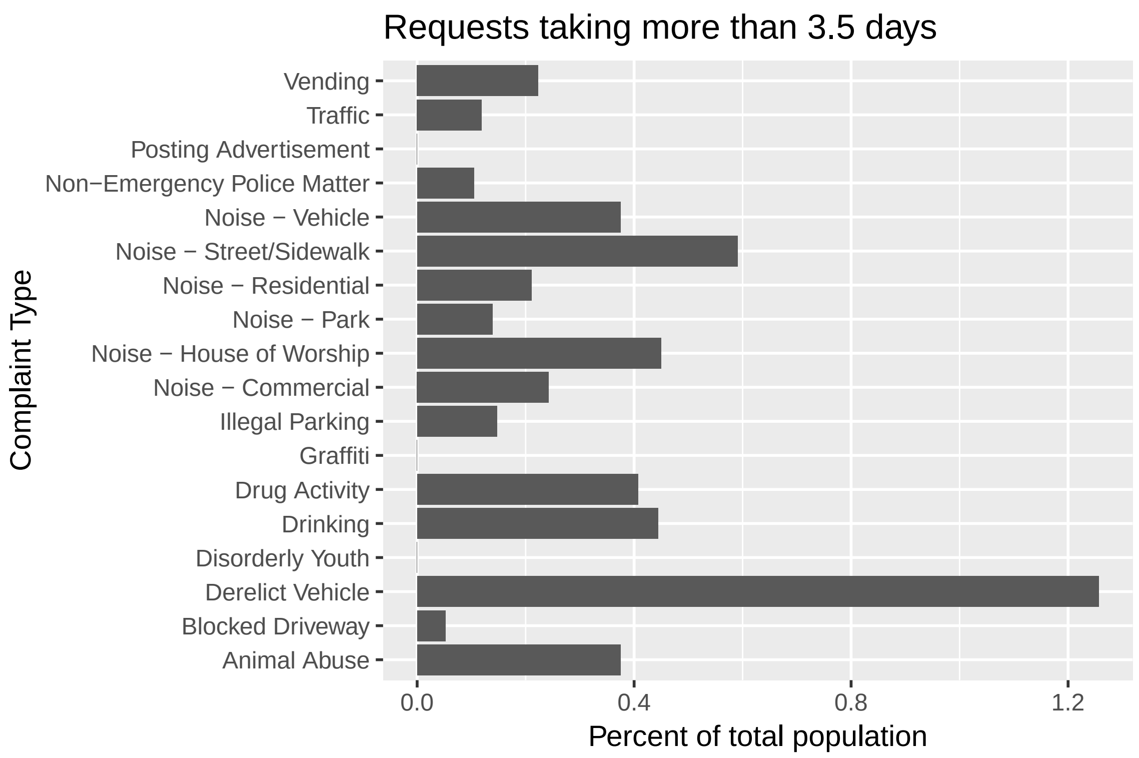
```
## Don't know how to automatically pick scale for object of type difftime. Defaulting to continuous.
```



Extreme value analysis

We want to further investigate requests that take very long to process. The percentage of complaints exceeding 3.5 days, grouped by Complaint Type, are shown below. The raw numbers look healthy - very few requests exceed this threshold. The complaint type with the highest number of long process times is a derelict vehicle, but as discussed earlier, this is to be expected. There is little evidence of structural inefficiency here due to the low numbers. If anything, noise complaints seem to be consistently higher in processing time than the others.

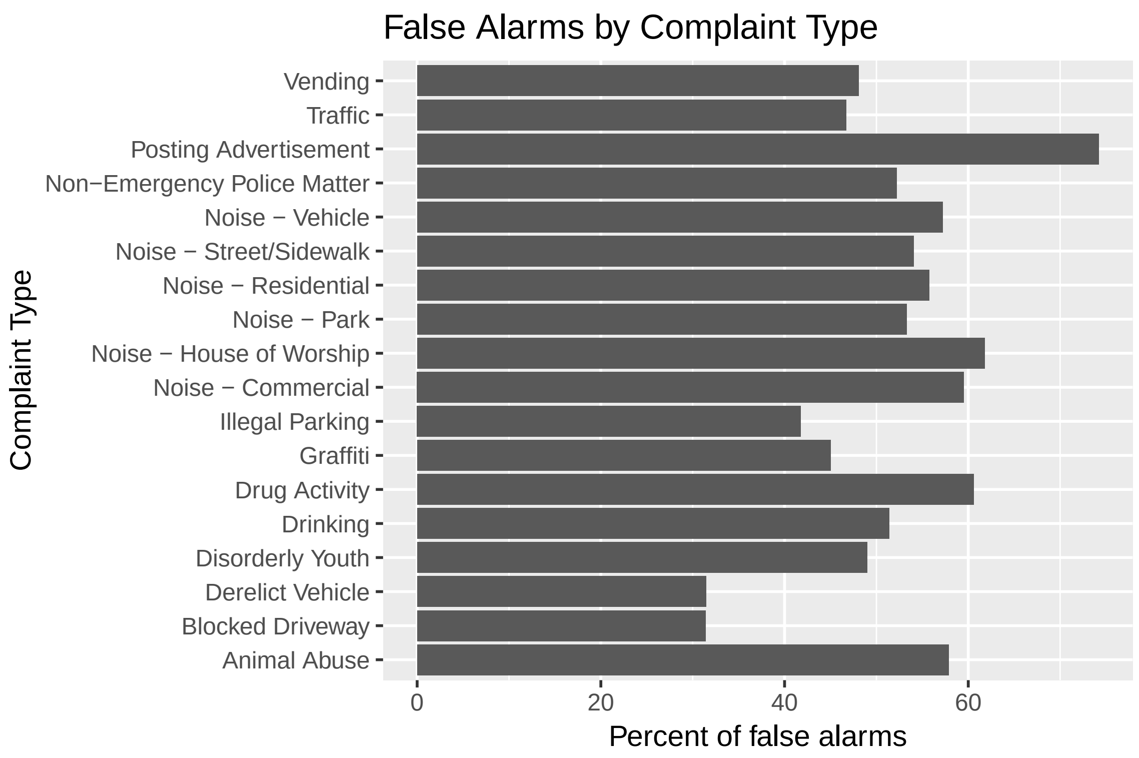
```
df$outlier = df$Process.Time > 5000
mean_time_zip <- df[,c(4,21)] %>% group_by(Complaint.Type) %>% summarize(num.outliers = sum(outlier), nu
mean_time_zip$percent.outliers = mean_time_zip$num.outliers/mean_time_zip$num * 100
ggplot(mean_time_zip,aes(x=Complaint.Type,y=percent.outliers))+geom_col() + ggtitle("Requests taking mor
```



False alarm analysis

Lastly, we find it interesting to analyze requests received by 311 that are actually false alarms. We define false alarms according to the resolution description column - these are the incidents where no action was required from the NYPD. As we can see, false alarms comprise nearly 40% of all requests. Some complaint types are more prone to false alarms than others. In particular, transient incidences such as noise, illegal advertising, animal abuse, and drug activity is prone to false alarms. It is very likely that the incident is over before the policeman arrives. This could be significant because that this suggests these incidents more time-sensitive and a delay in processing time can cause the incident to be left unaddressed.

```
df$falsealarm = (df$Resolution.Description == "The Police Department responded to the complaint and dete

mean_time_zip <- df[,c(4,22)] %>% group_by(Complaint.Type) %>% summarize(num.outliers = sum(falsealarm)
mean_time_zip$percent.false.alarm = mean_time_zip$num.outliers/mean_time_zip$num * 100
ggplot(mean_time_zip,aes(x=Complaint.Type,y=percent.false.alarm)) +
  geom_col() +
  ggtitle("False Alarms by Complaint Type") +
  coord_flip() +
  scale_y_continuous("Percent of false alarms") +
  scale_x_discrete("Complaint Type")
```

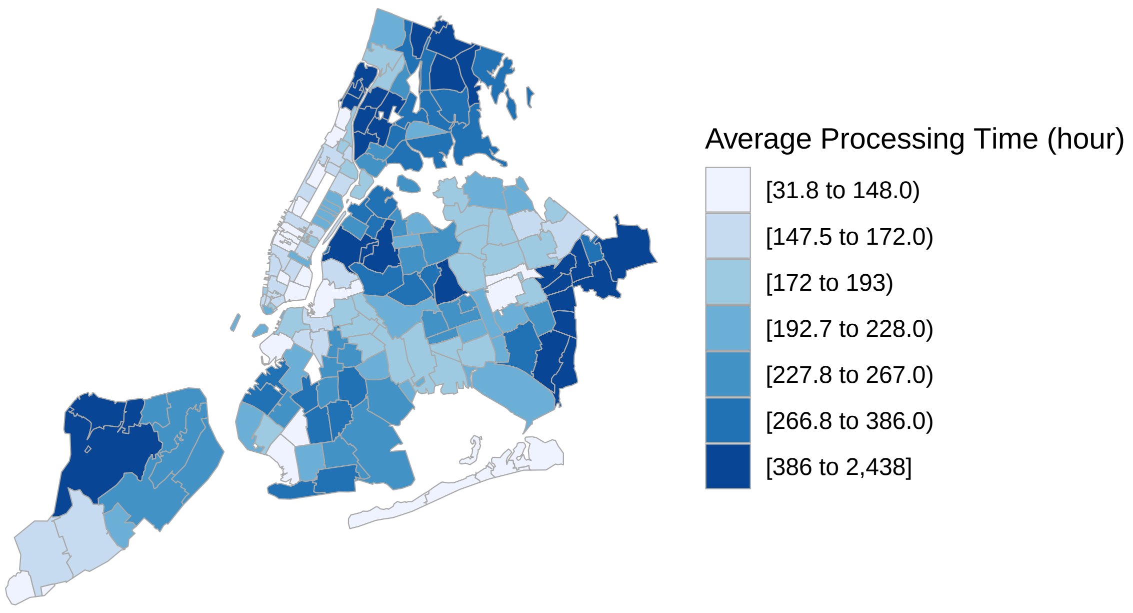


Geographical patterns

Police efficiency also has a visible geographical pattern across New York City. As seen in the following two visualizations, note that the distance to closest station is at a minimum in the requests from Manhattan while the request from parts of Staten Island and Brooklyn and Queens have much higher distances. A similar pattern appears in the processing time of the requests from these areas. Processing times in Manhattan tend to be lower than those from the Bronx, Staten Island, and parts of Brooklyn and Queens. This leads us to conclude that the distance to the closest station has a large effect on the processing time of police requests. This information could lead to increased city efforts to allocate resources to Staten Island and parts of Queens to help with the processing time in those areas.

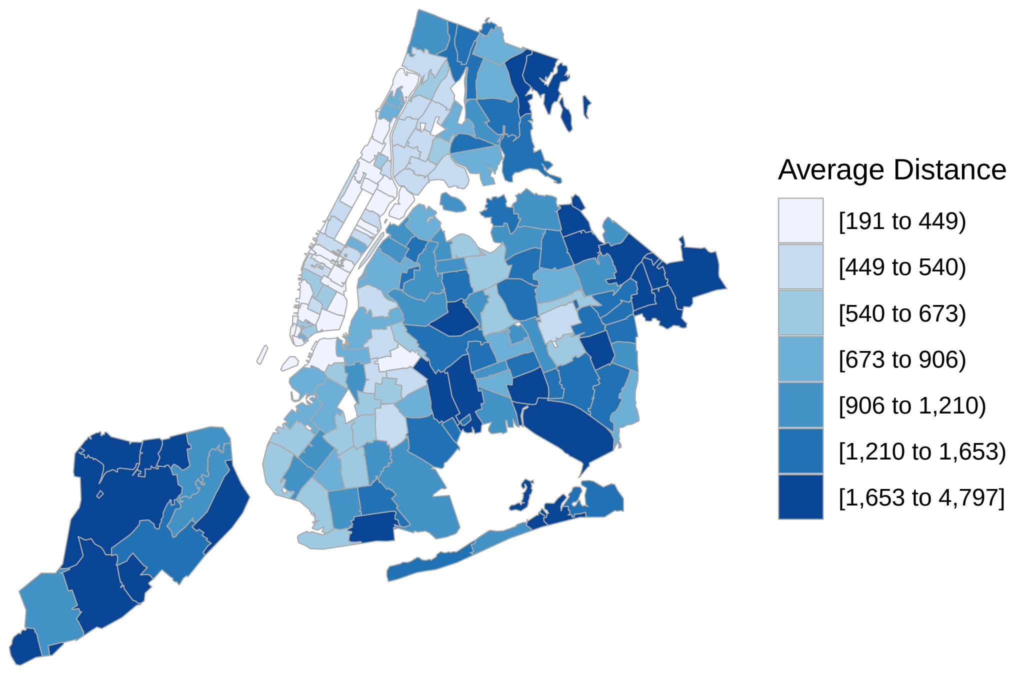
```r
library(choroplethrZip)
mean_time_zip <- df[is.na(df1$Incident.Zip)==FALSE,c(7,18)] %>% group_by(Incident.Zip) %>% summarize(mea

data(zip.regions)
names(mean_time_zip) <- c("region","value")
mean_time_zip$region = as.character(mean_time_zip$region)
mean_time_zip = mean_time_zip[mean_time_zip$region %in% zip.regions$region,]
zip_choropleth(mean_time_zip,zip_zoom = mean_time_zip$region, title="Average Processing Time of 311 Pol
```

Average Processing Time of 311 Police Requests



```
mean_time_zip <- df[is.na(df1$Incident.Zip)==FALSE,c(7,17)] %>% group_by(Incident.Zip) %>% summarize(mea

data(zip.regions)
names(mean_time_zip) <- c("region","value")
mean_time_zip$region = as.character(mean_time_zip$region)
mean_time_zip = mean_time_zip[mean_time_zip$region %in% zip.regions$region,]
zip_choropleth(mean_time_zip,zip_zoom = mean_time_zip$region, title="Average Distance from 311 Call to C
```

Average Distance from 311 Call to Closest Police Station

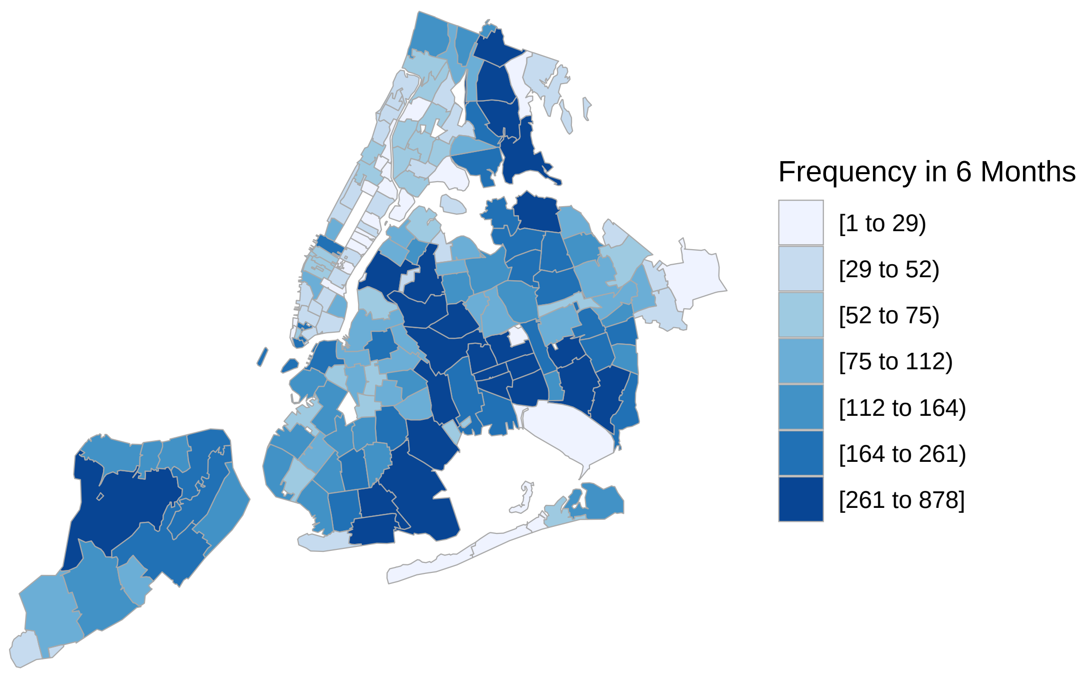


4c. Incident Patterns

Next, we will analyze incident patterns throughout the city. By looking at the zip codes of various types of incident types, we were able to visualize patterns of various incident frequencies. Below, we observed that street condition complaints were least common in Manhattan and occurred most often in parts of southern Queens and Brooklyn as well as eastern Bronx. This could be due to better road conditions in Manhattan.

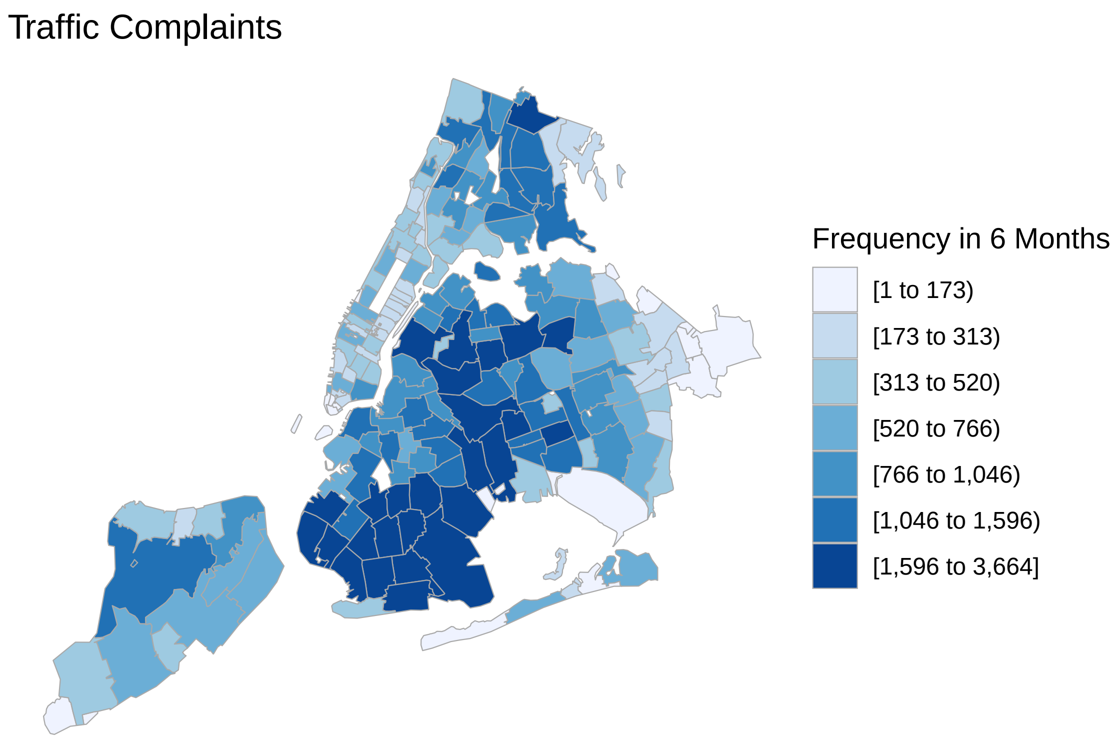
```r
mean_time_zip <- df[is.na(df1$Incident.Zip)==FALSE,c(7,15)] %>% group_by(Incident.Zip, Complaint.Subgrou

Noise = mean_time_zip[mean_time_zip$Complaint.Subgroups %in% "Street condition related",c(1,3)]
names(Noise) <- c("region","value")
Noise$region = as.character(Noise$region)
Noise = Noise[Noise$region %in% zip.regions$region,]
zip_choropleth(Noise,zip_zoom = Noise$region, title="Street Condition Complaints",legend="Frequency in 6
```

Street Condition Complaints



Next, from the map below, there is a clear region in Brooklyn and Queens that has a much higher frequency of traffic complaints than anywhere else in New York City. This information suggests that Brooklyn and Queens are areas that need the most work in terms of traffic regulation. The higher ratio of residential areas in these regions may be a factor in this.

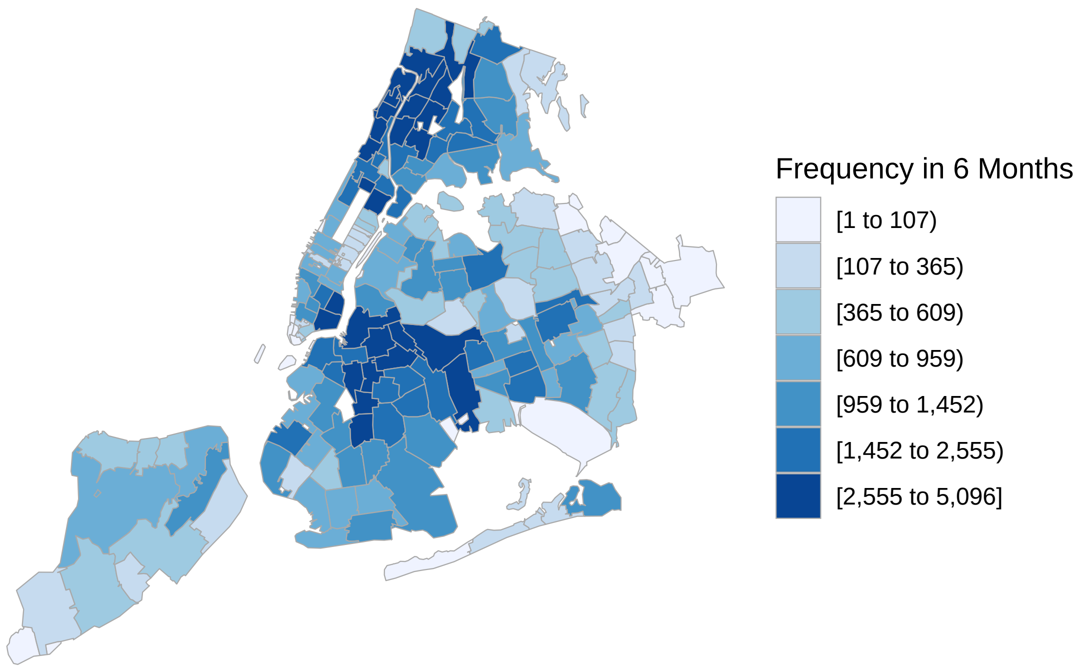
```r
mean_time_zip <- df[is.na(df1$Incident.Zip)==FALSE,c(7,15)] %>% group_by(Incident.Zip, Complaint.Subgrou

Noise = mean_time_zip[mean_time_zip$Complaint.Subgroups %in% "Traffic related",c(1,3)]
names(Noise) <- c("region","value")
Noise$region = as.character(Noise$region)
Noise = Noise[Noise$region %in% zip.regions$region,]
zip_choropleth(Noise,zip_zoom = Noise$region, title="Traffic Complaints",legend="Frequency in 6 Months")
```



From the map below, we observed that noise complaints occur most often in the Bronx, Lower East Side, and Brooklyn. This would be interesting to overlay with demographic information about resident age and occupation. These regions may have the highest noise complaints because there are more young people making noise at night. This information would also be useful for people interested in moving into New York City. Knowing which regions receive the most noise complaints could determine the best quiet (or lively) place to live that suits them.

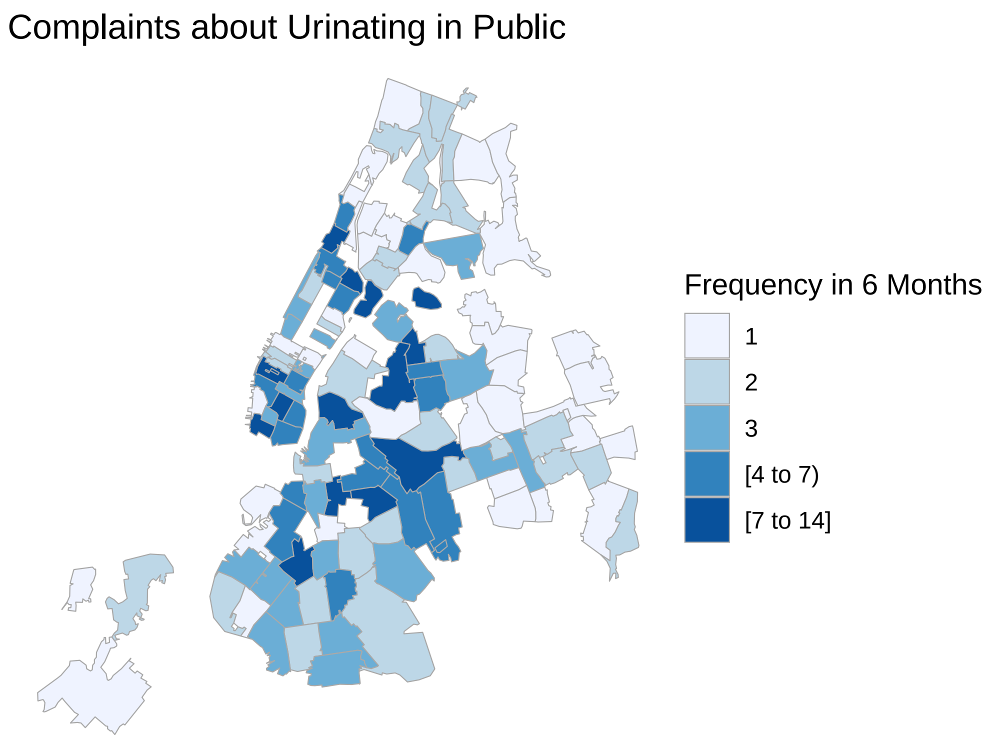
```
mean_time_zip <- df[is.na(df1$Incident.Zip)==FALSE,c(7,15)] %>% group_by(Incident.Zip, Complaint.Subgrou

Noise = mean_time_zip[mean_time_zip$Complaint.Subgroups %in% "Noise",c(1,3)]
names(Noise) <- c("region","value")
Noise$region = as.character(Noise$region)
Noise = Noise[Noise$region %in% zip.regions$region,]
zip_choropleth(Noise,zip_zoom = Noise$region, title="Noise Complaints",legend="Frequency in 6 Months")
```

Noise Complaints

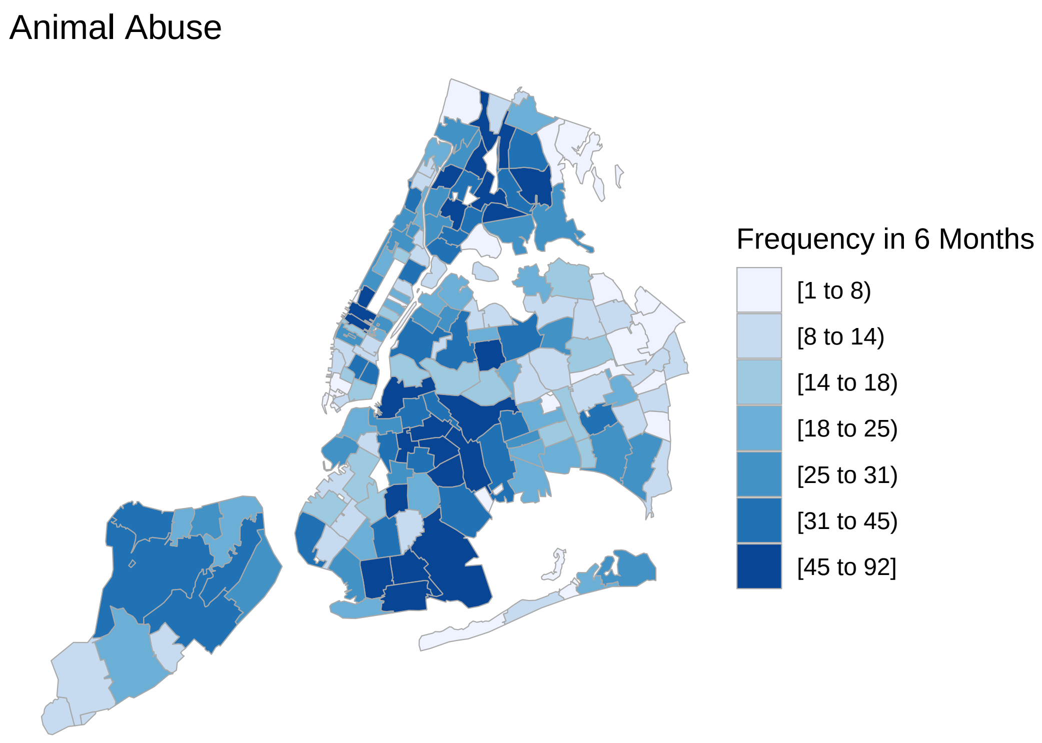


However, not all types of complaints show a clear geographical pattern for frequencies. Below, we have a map of complaints regarding urinating in public. As shown, there is no clear borough that has the most urination incidents. From this map, we can conclude that public urination does not change predictably along geographic regions. More information is needed in order to analyze the underlying causes of higher public urination frequencies. The same analysis applies to animal abuse frequencies, shown further below.

```r
dfold = read.csv("311_NYPD_6month.csv") #read old files to get additional data

mean_time_zip <- dfold[is.na(dfold$Incident.Zip)==FALSE,c(9,6)] %>% group_by(Incident.Zip, Complaint.Ty

Noise = mean_time_zip[mean_time_zip$Complaint.Type %in% "Urinating in Public",c(1,3)]
names(Noise) <- c("region","value")
Noise$region = as.character(Noise$region)
Noise = Noise[Noise$region %in% zip.regions$region,]
zip_choropleth(Noise,zip_zoom = Noise$region, title="Complaints about Urinating in Public",legend="Freq
```



```
mean_time_zip <- df[is.na(df1$Incident.Zip)==FALSE,c(7,4)] %>% group_by(Incident.Zip, Complaint.Type) %

Noise = mean_time_zip[mean_time_zip$Complaint.Type %in% "Animal Abuse",c(1,3)]
names(Noise) <- c("region","value")
Noise$region = as.character(Noise$region)
Noise = Noise[Noise$region %in% zip.regions$region,]
zip_choropleth(Noise,zip_zoom = Noise$region, title="Animal Abuse",legend="Frequency in 6 Months")
```



4d. Year-to-Year Patterns

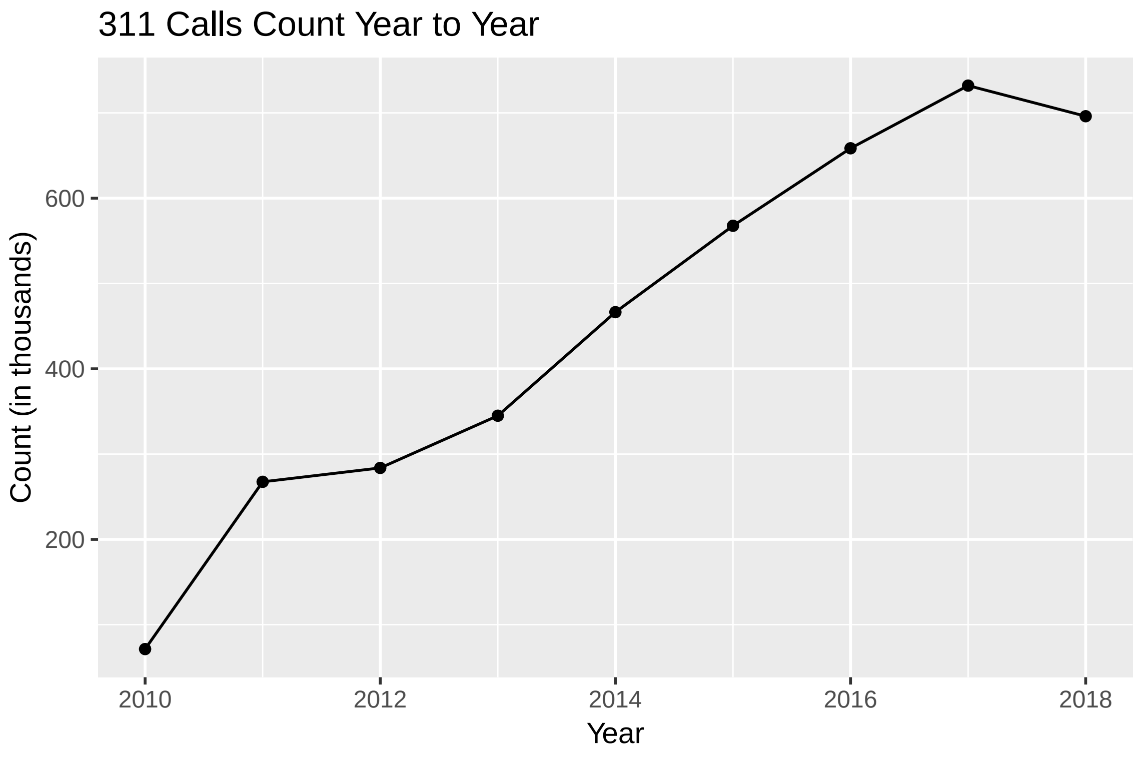
Next, let's look at how 311 processing time and incident count changes year to year. We first load the data which has already been preprocessed by an external IPython notebook, which you can find here.

```
df_allyears = read_csv("nyc_groupedby_process_count.csv")
```

```
## Parsed with column specification:
## cols(
##   zip = col_integer(),
##   year = col_integer(),
##   count = col_integer(),
##   process_time = col_double()
## )
```

First, we'd like to see how the count 311 calls change year to year.

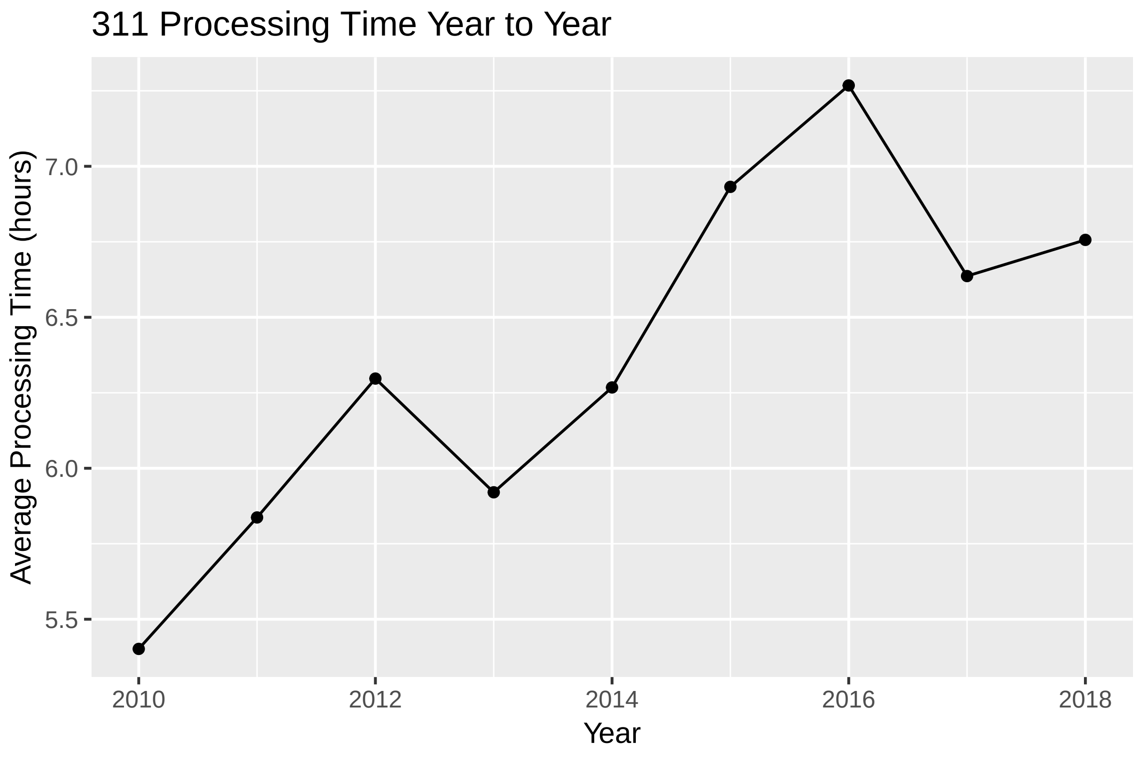
```
df_all_agg = df_allyears %>%
  group_by(year) %>%
  summarize(
    sum_count = sum(count),
    mean_process = mean(process_time)
  )

ggplot(data = df_all_agg, mapping = aes(x = year, y = sum_count)) +
  geom_line() +
  geom_point() +
  ggtitle("311 Calls Count Year to Year") +
  labs(x = "Year", y = "Count (in thousands)") +
  scale_y_continuous(labels = c(200,400,600), breaks = 10^5*c(2,4,6))
```



We can see that the number of calls year to year increase linearly. This is probably due to the increase in population as well as an awareness and increased effort of using 311 calls.

Then, we'd like to see how average processing time is affected year to year.

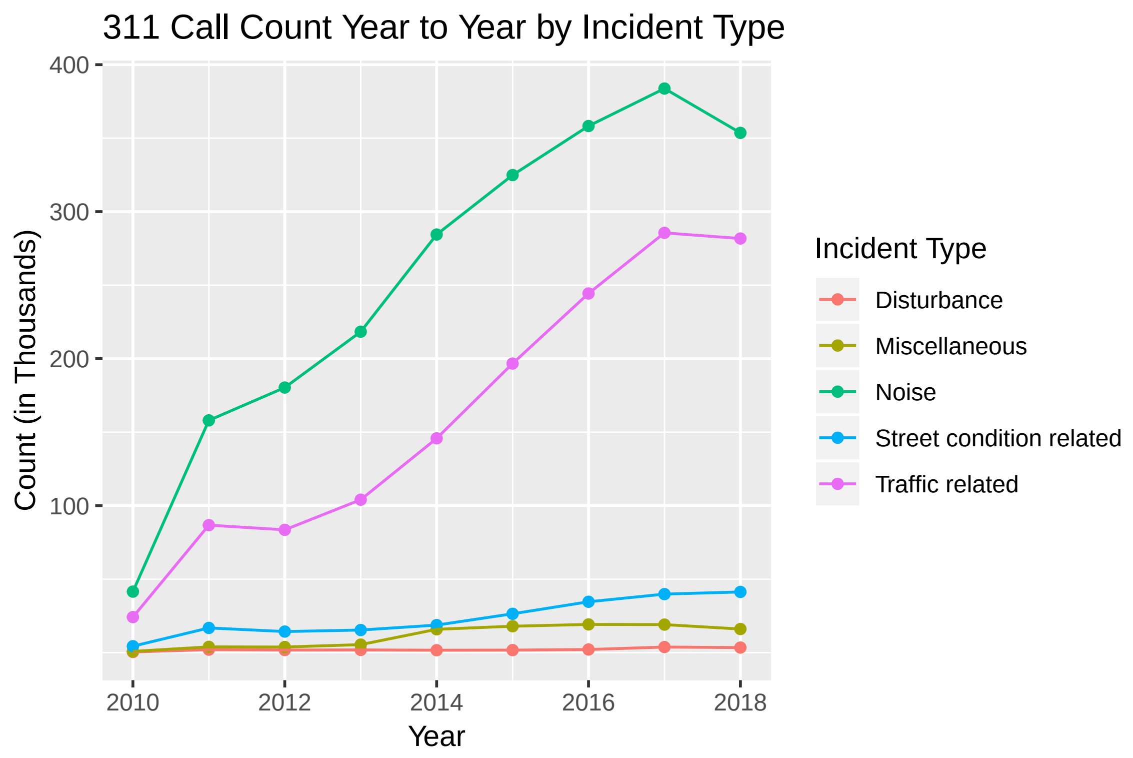
```r
ggplot(data = df_all_agg, mapping = aes(x = year, y = mean_process)) + geom_line() + geom_point() + ggt
```



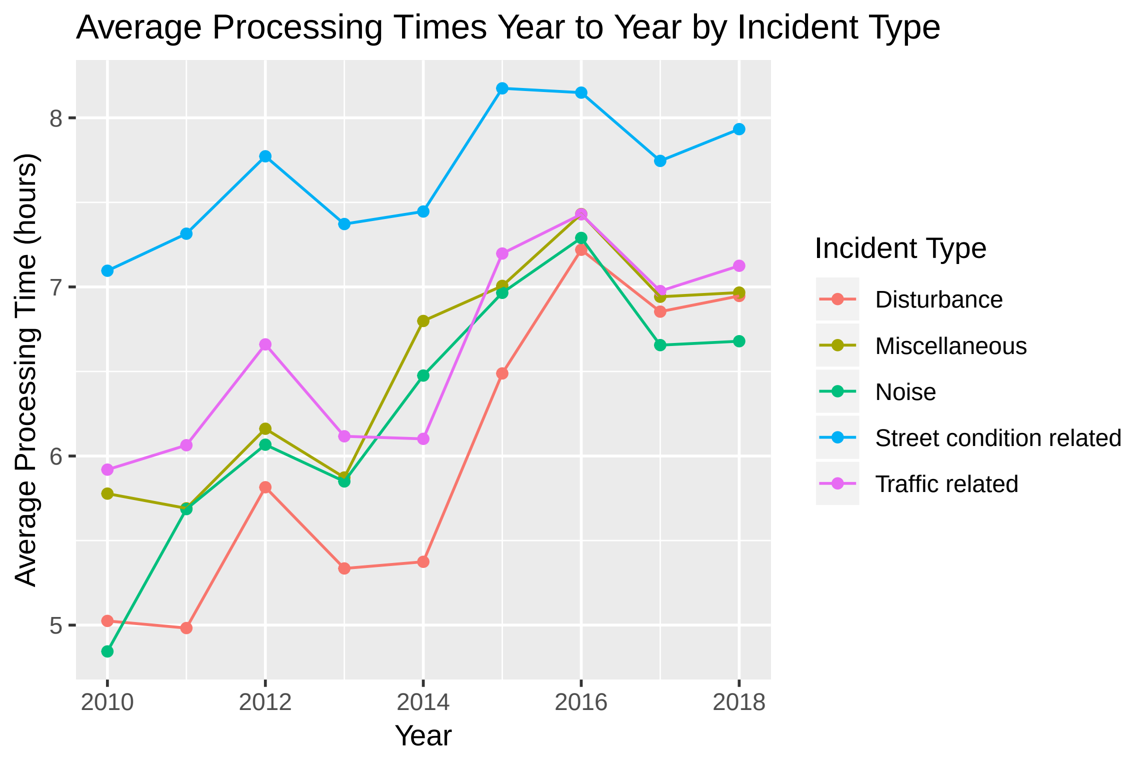
The beginning years (2010-2014) probably have a much lower processing time due to the lack of data points. However, in the later years we can see that the NYPD has improved their efficiency the past two years.

Finally, let's evaluate how the NYPD is performing in processing different kinds of incident types.

```r
df_allyears_group = read_csv("nyc_groupedby_subgroup.csv")
df_agg_group = df_allyears_group %>%
  group_by_(.dots = c("year","type")) %>%
  summarize(
    sum_count = sum(count),
    mean_process = mean(efficiency)
  ) %>%
  filter(type != 0)

ggplot(data = df_agg_group, mapping = aes(x = year, y = sum_count, color = type)) +
  geom_point() +
  geom_line() +
  ggtitle("311 Call Count Year to Year by Incident Type") +
  labs(x = "Year", y = "Count (in Thousands)", color = "Incident Type") +
  scale_y_continuous(labels = c(100,200,300,400), breaks = 10^5*c(1,2,3,4))
```



```r
ggplot(data = df_agg_group, mapping = aes(x = year, y = mean_process, color = type)) +
  geom_point() +
  geom_line() +
  ggtitle("Average Processing Times Year to Year by Incident Type") +
  labs(x = "Year", y = "Average Processing Time (hours)", color = "Incident Type")
```



From the results above, we can see that Noise and Traffic-related complaints grow much more rapidly year to year than other types of complaints, probably due to the increase in population. The interesting fact is that the increase in Noise/Traffic complaints causes other incident types to have a higher processing time, e.g. Disturbance, even though the other incidents don't grow as fast. From the results above we can reasonably conclude that a rapid increase in one type of incident causes higher processing time not only for that incident but others as well.

5. Interactive Component

In the analyses above, we were able to see patterns in regards to time and space, but not both at the same time. To address this problem, we created an interactive tool in D3 to help users visualize both spatial and chronological patterns on how 311 calls is processed year to year and how 311 complaints are submitted hour by hour.

You can find the year to year D3 tool here and the hour by hour tool here.

By using the tool above, we were able to come up with a few insights: - In the beginning years (2010-2014), it looks as if police are much more efficient, but this is due to the lack of data as shown in the line graph above. - From 2014-2018, on average NYPD are much less efficient in the Bronx and Queens. As stated before, in Queens this is probably due to the lack of police stations. However, for the Bronx, it is interesting to note that the police station average distance is not too different from Manhattan. The Bronx might display higher processing time due to a larger number of calls, but further research can be done on this fact. - As stated above, most complaints are reported near midnight. We can see that most calls originate from Upper West and on the area between Brooklyn and Queens.

Next steps for D3 Tool:

- Combine the two into one page, creating radio buttons to allow the visualization to dynamically change between one dataset to another.
- Create radio buttons to allow us to visualize the data on different time partitions (by year, month, week, etc.)
- Use tool to look at more data patterns e.g. incident count, specific incident types, by day of the week.
- Refactor code, since the code is a bit messy due to the time constraint.

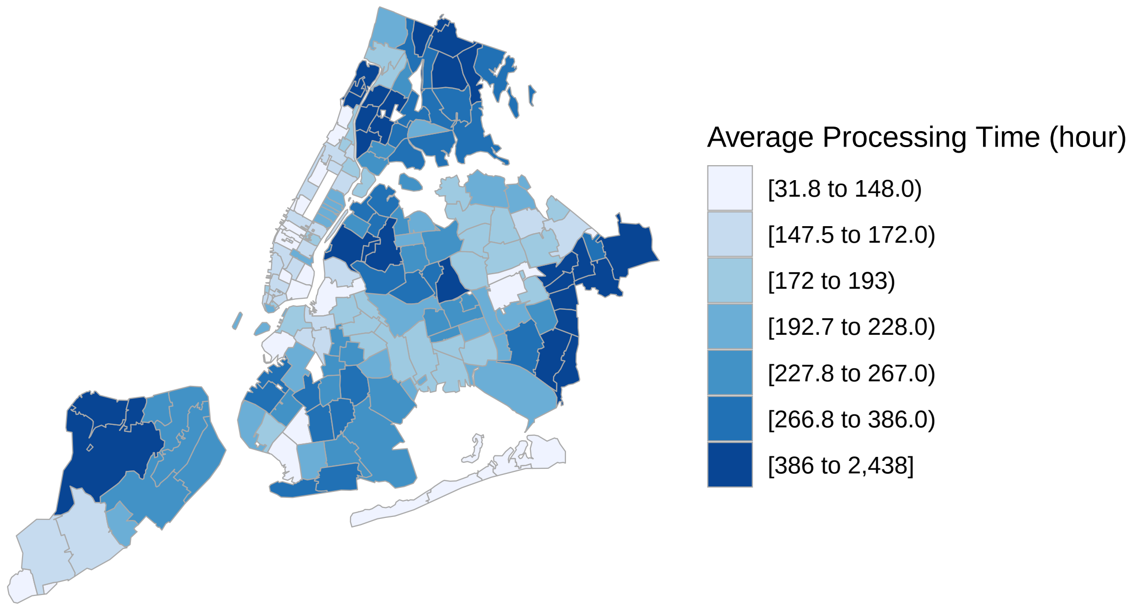
6. Executive Summary

We analyzed the patterns of 311 service requests, focusing on time-related patterns and space-related patterns. We discovered that noise complaint is the largest complaint type, comprising over 50% of all complaints received by NYPD. They are mostly received after 10 pm and before 3 am, and more frequently on weekends than weekdays. The second most common complaint type is traffic and street condition related complaint, comprising complaints regarding poor traffic, blocked driveways, illegal parking, and derelict vehicles. These complaints are mostly received during the day, making them inversely related to noise complaints. Geographically, there were many significant patterns that appeared for each incident type. For example, Brooklyn and Queens clearly have the highest frequency of traffic complaints while noise complaints are mostly concentrated in Brooklyn, Lower East Side, and the Bronx. However, not all incidents had such clear patterns. Public urination, for example, seemed to be scattered unpredictably across the map. Other complaints, encompassing things such as illegal advertisement posting, animal abuse, graffiti, and so on are received occasionally but we do not have enough data and did not show enough of a geographical pattern in order to draw any conclusions.

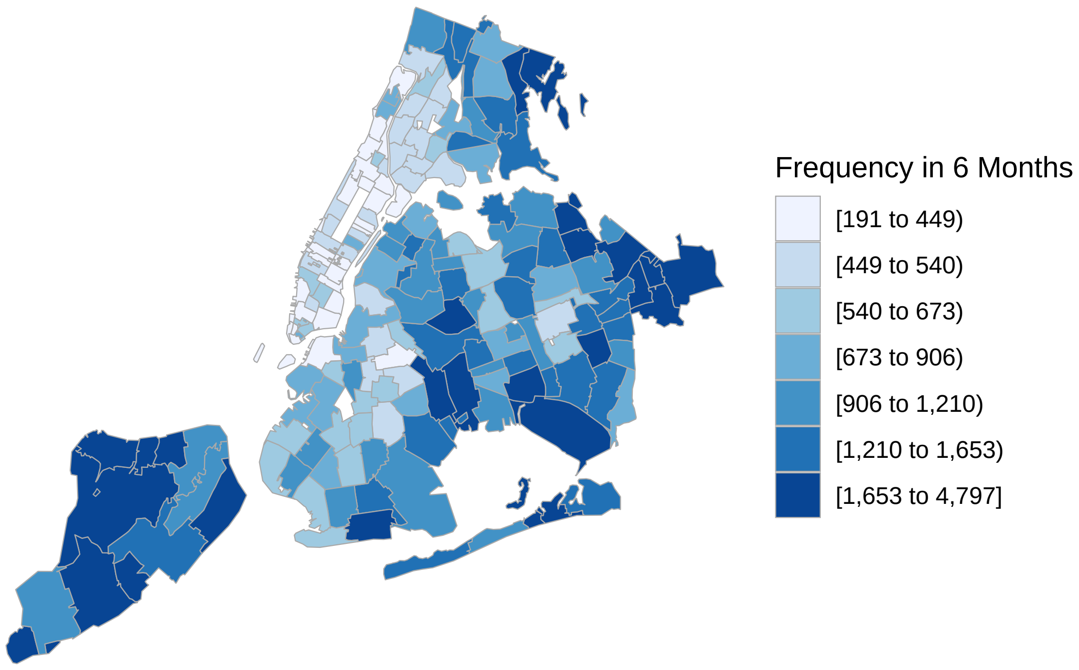
In addition to discovering patterns in complaints received, our other focus is discovering whether there are any structural inefficiencies in how the complaints are handled. Namely, we want to find out whether there are any times of day when complaints take longer to process, or whether there are any regions in New York where the police department is particularly inefficient. In regards to time, we found that most calls are generally received near midnight, especially regarding noise complaints (see D3 tool above). Even so, we found that the police processing time stays relatively the same with other time of days, thus no operational actions should be taken on this insight.

Spatially, as seen below, we found that there are similar patterns across boroughs for processing times and distance to stations. Manhattan has both the lowest processing times and the lowest distance to stations for requests. This led us to conclude that **one's distance to the nearest police station** is one of the biggest factors in determining processing time and service efficiency. We have seen that the processing time in Queens and Bronx have been **historically low** with a large number of requests, thus we recommend that more police stations are established in these high-request, low-efficiency areas. In a lower priority, it would also be beneficial to establish police stations in Staten Island and Brooklyn to improve processing time.

Average Processing Time of 311 Police Requests (last 6 months)



Average Distance from 311 Call to Closest Police Station



Furthermore, we analyzed the extreme cases – cases where the processing time is above 3 days – to see whether there are any complaint types that are more poorly handled than the others. We found that derelict vehicle, which is also the complaint type with highest mean processing time, has the highest proportion of extremal cases. Noise complaints also tend to have longer processing times than other complaint types, ostensibly due to the volume of such complaints received.

Lastly, we analyzed false alarms in the dataset, which we define as incidents for which no police action was required. Noise complaints, traffic complaints, and complaints such as illegal advertising are particularly prone to false alarms, suggesting that these incidents should be of lower priority, as they are often false alarms.

7. Conclusion

We found that the strongest predictor of high processing time was distance to a police station. Manhattan has a high density of police stations, and we found that this coincided with the low processing time for Manhattan incidents. The converse is true outside of Manhattan. Traffic and noise complaints are among the most common complaint types and they are most concentrated in places outside of Manhattan. The following are thus our recommendations to improve processing time: 1. Building more police stations in high-frequency and low-efficiency regions, such as Brooklyn and Queens. 2. Have false-alarm prone incidents to a lower priority, such illegal advertising.

8. Next Steps

In the future, there are some areas of which we can perform further analyses: 1. Deeper dive on false alarms. Understand which neighborhoods are prone to false alarms and understand why. Understand the predictors of a false alarm: what are the indicators of a call being a false alarm? Location? Time of day? Would probably require predictive modeling to answer this question. 2. Improve D3 tool to be more generalizable to more datases. 3. Find more data to pinpoint the cause of low processing time for NYPD.