



DATA WRANGLING AND HUSBANDRY

COURSE PROJECT

TOPIC: EDA ON NYC EMS INCIDENT DISPATCH DATA

Team Members:

Anirudh Gaur (216007356)

Prashanth Aripirala (219000924)

Rishik Salver (217001710)


Rohit Macherla (219008045)

ABOUT THE DATASET

The dataset is sourced from the NYC Open Data API, which is a repository of public data sets provided by the City of New York.



It contains information on over 2 million emergency EMS calls made to the New York City Fire Department (FDNY) between January 2019 and October 2021.



The dataset includes a variety of fields such as the type of emergency, location, time of day, and response times by FDNY.

OBTAINING THE DATASET

- The URL provided is used to fetch the data from the NYC EMS Incident Dispatch Dataset.
- The 'fromJSON()' function is used to convert the fetched data from JSON format to a usable R data frame.
- The fetched data is limited to 2,300,000 records using the \$limit parameter in the URL to prevent overload or excessive memory usage.

```
```{r}
url<- "https://data.cityofnewyork.us/resource/76xm-jjuj.json?$limit=2300000"
```
```

```
```{r}
json_results <- url %>% fromJSON()
```
```

```
```{r}
glimpse(json_results)
```
```

Rows: 2,300,000
Columns: 32

| | |
|-----------------------------------|--|
| \$ X | <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16... |
| \$ cad_incident_id | <int> 220904738, 220904735, 220904734, 220904733, 220904732... |
| \$ incident_datetime | <chr> "2022-03-31T23:59:52.000", "2022-03-31T23:59:45.000",... |
| \$ initial_call_type | <chr> "STNDBY", "DRUG", "DRUG", "TRAUMA", "SICK", "INJURY",... |
| \$ initial_severity_level_code | <int> 8, 4, 4, 2, 6, 5, 2, 3, 3, 2, 7, 5, 2, 3, 4, 5, 5, 2,... |
| \$ final_call_type | <chr> "STNDBY", "DRUG", "DRUG", "TRAUMA", "SICK", "INJURY",... |
| \$ final_severity_level_code | <int> 8, 4, 4, 2, 6, 5, 2, 3, 3, 2, 7, 5, 2, 3, 4, 5, 5, 2,... |
| \$ valid_dispatch_rspns_time_indc | <chr> "N", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y",... |
| \$ dispatch_response_seconds_qy | <int> 0, 0, 11, 33, 61, 34, 24, 10, 26, 22, 11, 18, 10, 30,... |
| \$ valid_incident_rspns_time_indc | <chr> "N", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y",... |
| \$ incident_close_datetime | <chr> "2022-03-31T23:59:52.000", "2022-04-01T00:45:52.000",... |
| \$ held_indicator | <chr> "N", "N", "N", "N", "N", "N", "N", "N", "N", "N", "N",... |
| \$ borough | <chr> "MANHATTAN", "BRONX", "MANHATTAN", "BRONX", "BROOKLYN",... |
| \$ incident_dispatch_area | <chr> "M7", "B2", "M3", "B4", "K4", "M2", "Q6", "M2", "M6",... |
| \$ zipcode | <int> 10035, 10458, 10019, 10470, 11212, 10014, 11373, 1000... |
| \$ policeprecinct | <int> 25, 48, 18, 47, 73, 6, 110, 9, 26, 112, 13, 23, 18, 1... |
| \$ citycouncildistrict | <int> 8, 15, 3, 11, 41, 3, 25, 2, 7, 29, 2, 8, 4, 32, 26, 3... |
| \$ communitydistrict | <int> 111, 206, 104, 212, 316, 102, 404, 103, 109, 406, 105... |
| \$ communityschooldistrict | <int> 4, 10, 2, 11, 23, 2, 24, 1, 3, 28, 2, 4, 2, 27, 30, 1... |
| \$ congressionaldistrict | <int> 12, 15, 10, 16, 9, 10, 6, 12, 10, 6, 12, 13, 12, 8, 1... |
| \$ reopen_indicator | <chr> "N", "N", "N", "N", "N", "N", "N", "N", "N", "N", "N",... |
| \$ special_event_indicator | <chr> "N", "N", "N", "N", "N", "N", "N", "N", "N", "N", "N",... |
| \$ standby_indicator | <chr> "Y", "N", "N", "N", "N", "N", "N", "N", "N", "N", "N",... |
| \$ transfer_indicator | <chr> "N", "N", "N", "N", "N", "N", "N", "N", "N", "N", "N",... |
| \$ first_assignment_datetime | <chr> NA, "2022-03-31T23:59:45.000", "2022-03-31T23:59:13.0... |
| \$ first_activation_datetime | <chr> NA, "2022-03-31T23:59:45.000", "2022-03-31T23:59:19.0... |
| \$ first_on_scene_datetime | <chr> NA, "2022-03-31T23:59:45.000", "2022-04-01T00:02:02.0... |
| \$ incident_response_seconds_qy | <int> NA, 0, 180, 170, 1140, 591, 409, 359, 433, 455, 203, ... |
| \$ incident_travel_tm_seconds_qy | <int> NA, 0, 169, 137, 1079, 557, 385, 349, 407, 433, 192, ... |
| \$ first_to_hosp_datetime | <chr> NA, "2022-04-01T00:01:50.000", "2022-04-01T00:15:33.0... |
| \$ first_hosp_arrival_datetime | <chr> NA, "2022-04-01T00:06:15.000", "2022-04-01T00:20:22.0... |
| \$ incident_disposition_code | <int> NA, 82, 82, 90, 82, 82, 82, 82, 82, 90, 82, 93, 82, 9... |

PRE-PROCESSING OF DATASET

- We preprocessed the data by dropping unnecessary columns, renaming some of the columns, and converting data types to improve the quality of the data.
- To ensure data quality and accuracy, we removed rows with missing or NA values from the dataset.

```
'data.frame': 2198698 obs. of 18 variables:
 $ incident_id      : int  220904735 220904734 220904733 220904732 220904731
220904730 220904729 220904728 220904727 220904726 ...
 $ incident_datetime : POSIXct, format: "2022-03-31 23:59:45" "2022-03-31 23:59:02"
...
 $ initial_call_reason : chr  "DRUG" "DRUG" "TRAUMA" "SICK" ...
 $ initial_severity_level_code: int  4 4 2 6 5 2 3 3 2 7 ...
 $ final_call_reason : chr  "DRUG" "DRUG" "TRAUMA" "SICK" ...
 $ final_severity_level_code : int  4 4 2 6 5 2 3 3 2 7 ...
 $ time_elapsed_assignment : int  0 11 33 61 34 24 10 26 22 11 ...
 $ incident_close_datetime : POSIXct, format: "2022-04-01 00:45:52" "2022-04-01 00:55:56"
...
 $ held_indicator      : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ zipcode            : int   10458 10019 10470 11212 10014 11373 10003 10027 11375
10016 ...
 $ reopen_indicator    : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ special_event_indicator : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ standby_indicator   : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ transfer_indicator  : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ first_assignment_datetime : POSIXct, format: "2022-03-31 23:59:45" "2022-03-31 23:59:13"
...
 $ time_incident_response : int   0 180 170 1140 591 409 359 433 455 203 ...
 $ time_incident_travel  : int   0 169 137 1079 557 385 349 407 433 192 ...
 $ incident_disposition_code : int   82 82 90 82 82 82 82 82 90 82 ...
```

```
```{r}
Define the regex pattern for a valid US zip code
zip_pattern <- "^[0-9]{5}(?:-[0-9]{4})?$"

Apply the regex pattern to the zipcode column in clean_data
invalid_zips <- which(!str_detect(clean_data$zipcode, zip_pattern))

Display the count of invalid zip codes
cat("Count of invalid zip codes:", length(invalid_zips), "\n")
```
```

Count of invalid zip codes: 3

```
#Dispatch Code validation, check unique values and convert to character.

```{r}
clean_data <- subset(clean_data, incident_disposition_code %in% c("82",
"83", "87", "90", "91", "92", "93", "94", "95", "96", "CANCEL", "DUP", "NOTSNT", "ZZZZZZ"))

```{r}
code_meaning_map <- c("82" = "transporting patient", "83" = "patient pronounced dead", "87" =
"cancelled", "90" = "unfounded", "91" = "condition corrected", "92" = "treated not
transported", "93" = "refused medical aid", "94" = "treated and transported", "95" = "triaged
at scene no transport", "96" = "patient gone on arrival", "CANCEL" = "cancelled", "DUP" =
"duplicate incident", "NOTSNT" = "unit not sent", "ZZZZZZ" = "no disposition")
```
```

# ADDRESSING INVALID AND OUT OF RANGE VALUES

- A regex pattern was defined for a valid US zip code. The pattern was applied to the zip code column in the data to filter out the invalid zip code.
- The data set is processed to consider values in a given subset and the rest of the rows are dropped. Another column is added to classify each value of dispatch code.

# EDGE CASE DETECTION

```
```{r}
# count number of rows where incident_close_datetime is before or equal to incident_datetime
n_before <- sum(clean_data$first_assignment_datetime < clean_data$incident_datetime, na.rm = TRUE)

# print the result
print(n_before)
```
```

[1] 0

```
```{r}
# count number of rows where incident_close_datetime is before or equal to incident_datetime
n_before <- sum(clean_data$incident_close_datetime < clean_data$first_assignment_datetime, na.rm = TRUE)

# print the result
print(n_before)
```
```

[1] 52

```
```{r}
# Check for out-of-range values in initial_severity_level_code
invalid_initial_severity <- !between(clean_data$initial_severity_level_code, 1, 8)

# Count the number of out-of-range values
cat(paste("Number of out-of-range initial severity levels:", sum(invalid_initial_severity),
"\n"))

# Check for out-of-range values in final_severity_level_code
invalid_final_severity <- !between(clean_data$final_severity_level_code, 1, 8)

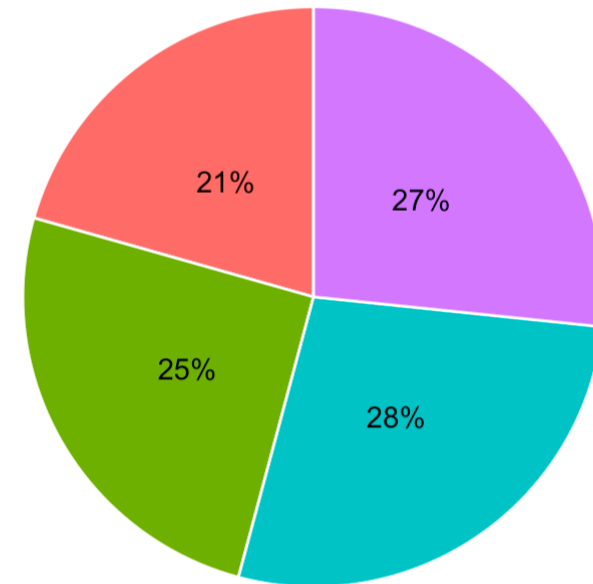
# Count the number of out-of-range values
cat(paste("Number of out-of-range final severity levels:", sum(invalid_final_severity), "\n"))
```
```

Number of out-of-range initial severity levels: 7  
Number of out-of-range final severity levels: 0

# CALL TIME CATEGORY DISTRIBUTION

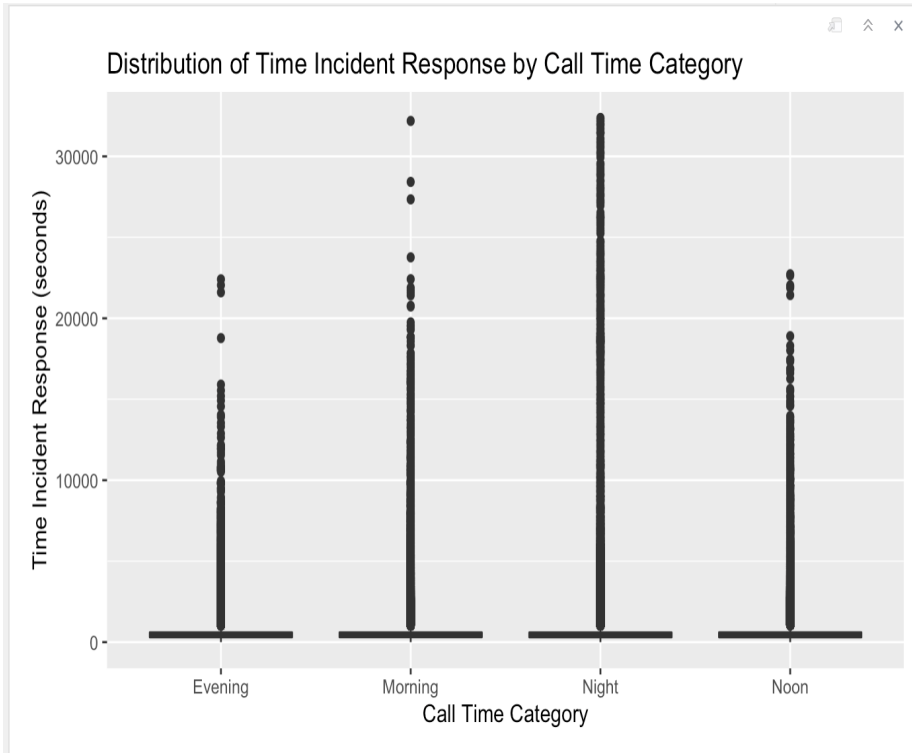
1. We created a new function `categorize_call_time` to categorize the call time into four categories: Morning, Noon, Evening, and Night based on the hour of the day.
2. The Pie chart helps us to identify the patterns at which calls are made.
3. Useful for identifying which time of the day would require more resources and work force.

Call Time Category Distribution



Call Time Category ■ Evening ■ Morning ■ Night ■ Noon

# REMOVING OUTLIERS



**A boxplot was used to visualize the distribution of Time Incident Response across different Call Time Categories.**



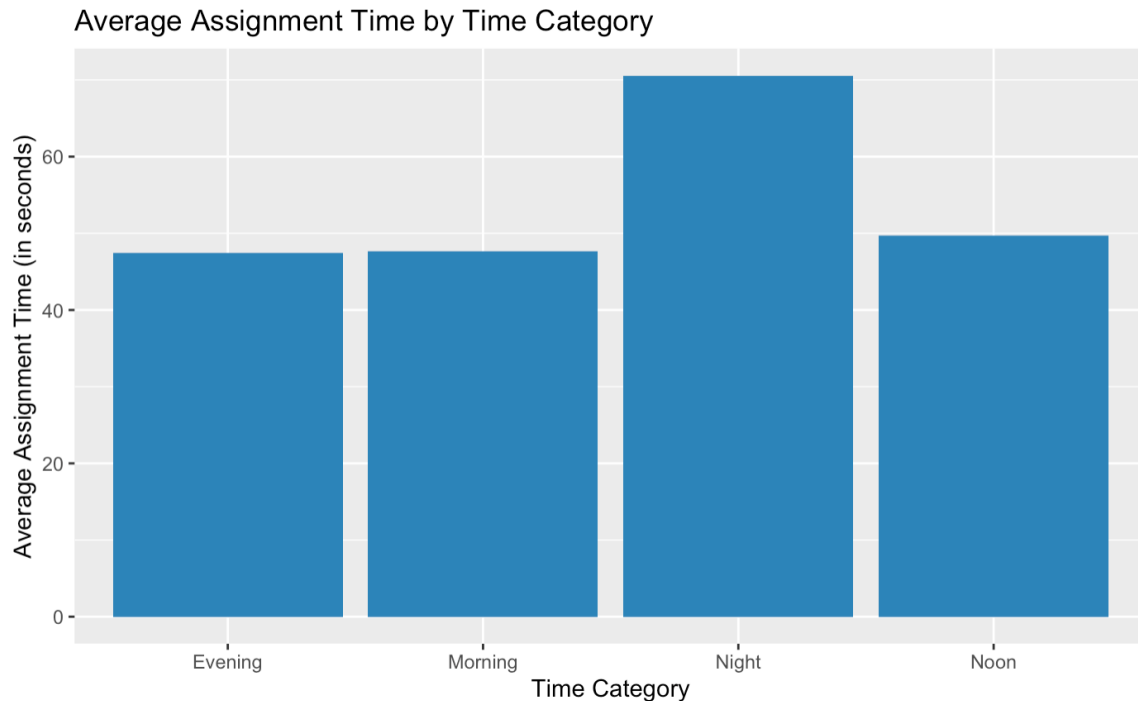
**The plot can help us understand how the Time Incident Response varies across different Call Time Categories.**



**We defined a function called `remove_outliers` which takes the data and a variable as input and removes the outliers from that variable using the interquartile range (IQR) method.**

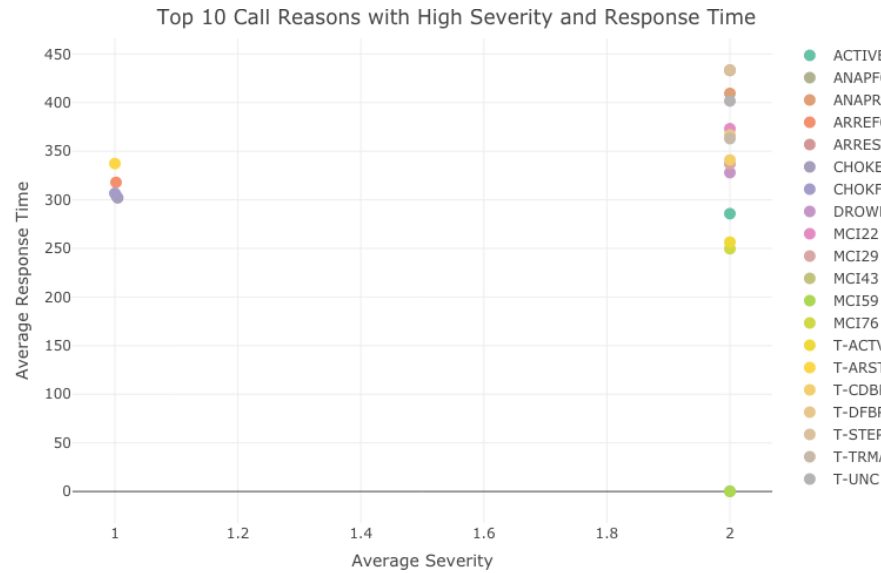
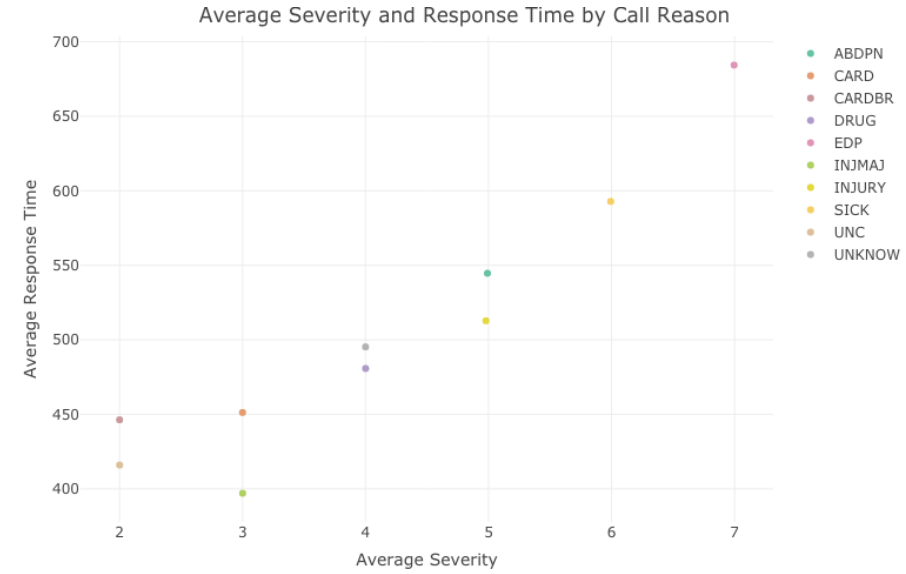
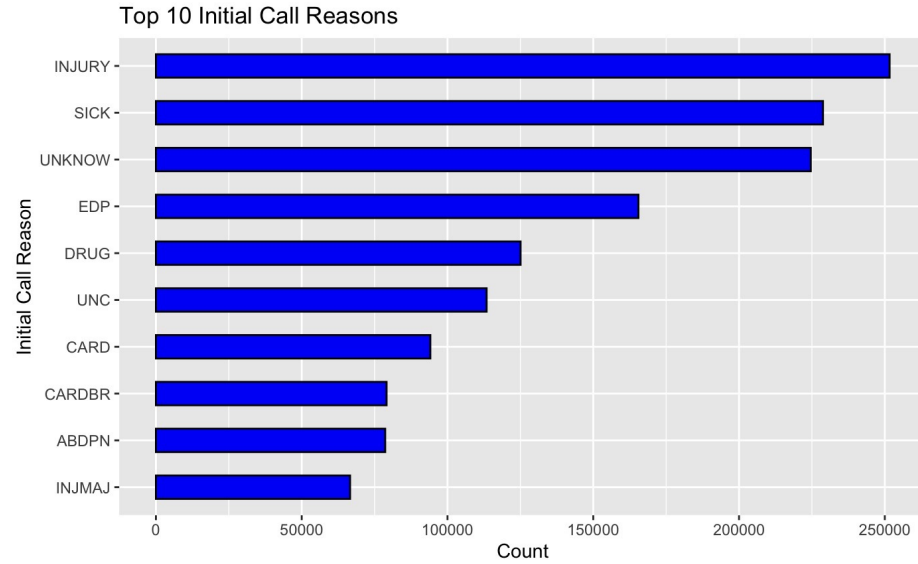


# ANALYZING RESPONSE & ASSIGNMENT TIMES



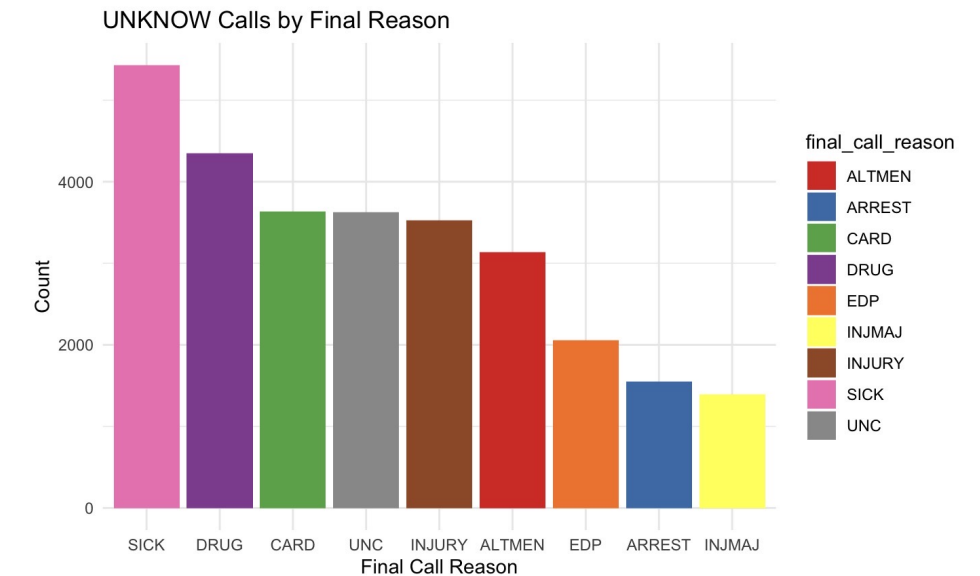
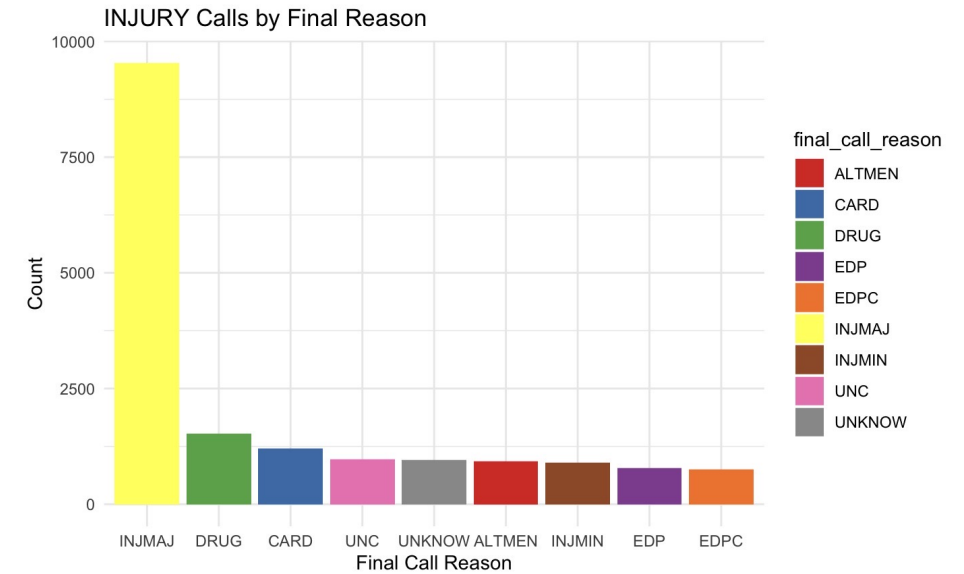
- The graph shows the average assignment time for incidents, grouped by time category.
- It can be used to analyze how well the department is equipped to handle the workload during different times of the day.
- The data can be used to identify if additional resources are required to reduce the assignment time during certain periods.

# EXPLORING THE REASONS BEHIND EMS CALLS



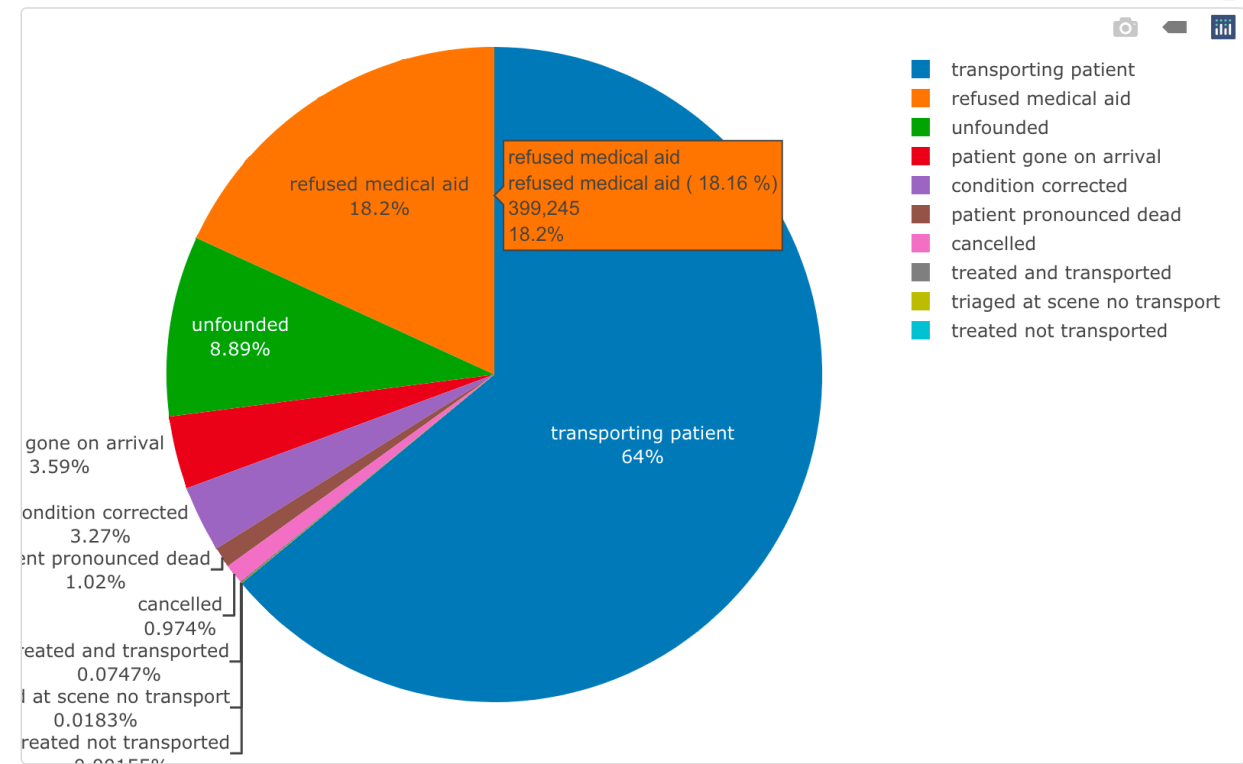
# EMS CALLS: FINAL REASONS DIFFER FROM INITIAL

- Approximately 300k incidents which have different initial and final call reasons.
- This analysis will assist in improving the accuracy of emergency response times by helping dispatchers and EMS workers anticipate the most likely final reasons for a given type of initial call.

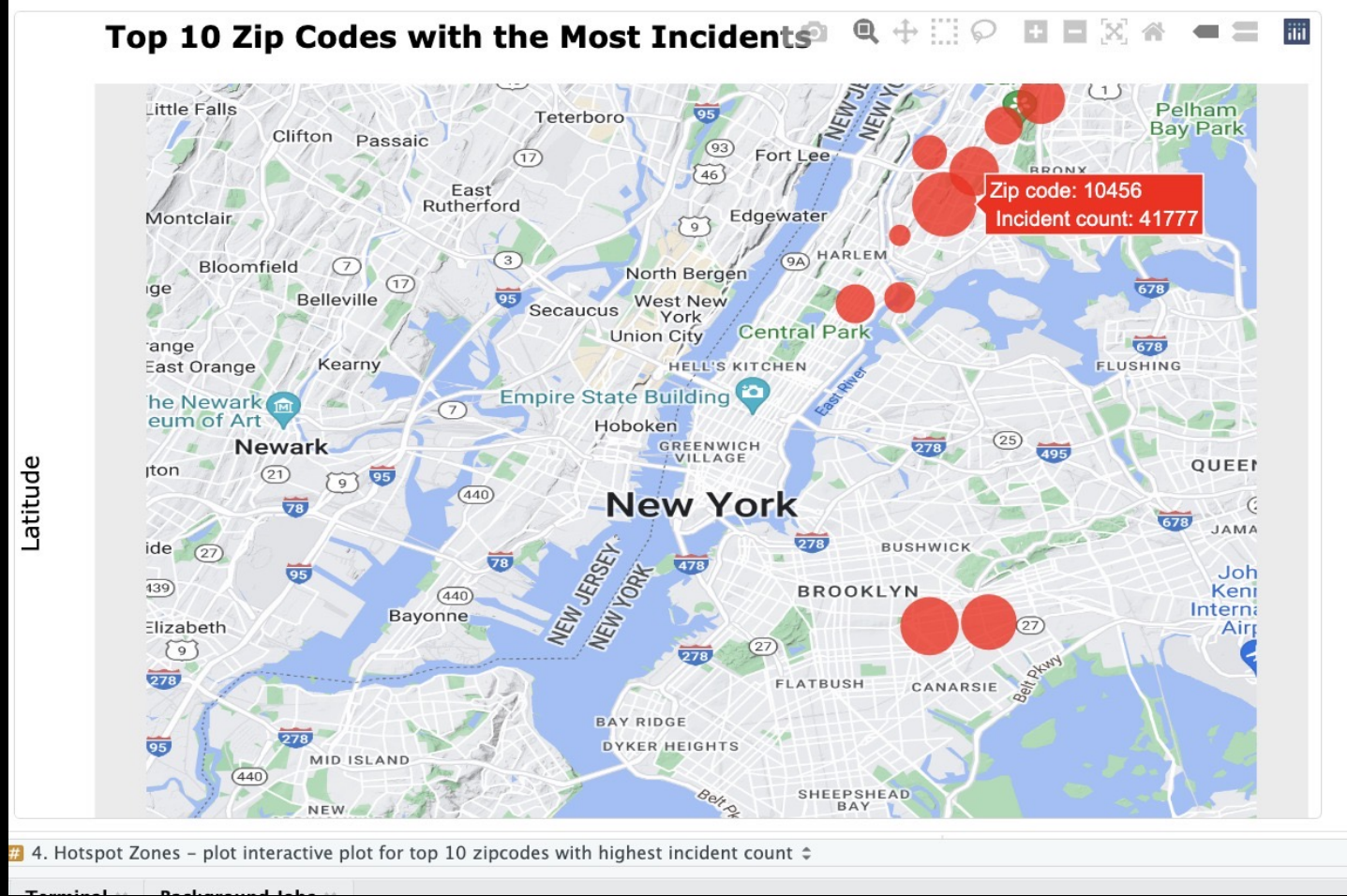


# DISTRIBUTION OF INCIDENT DISPOSITION CODE

- These are the codes assigned while closing the incident by the EMS operator, indicating the response by the emergency services.
- The pie chart shows the distribution of incident disposition codes, indicating which types of incidents were most frequent and the kind of response by emergency services.
- This information can be useful for identifying patterns and trends in emergency incidents and for informing resource allocation and training for emergency services personnel.



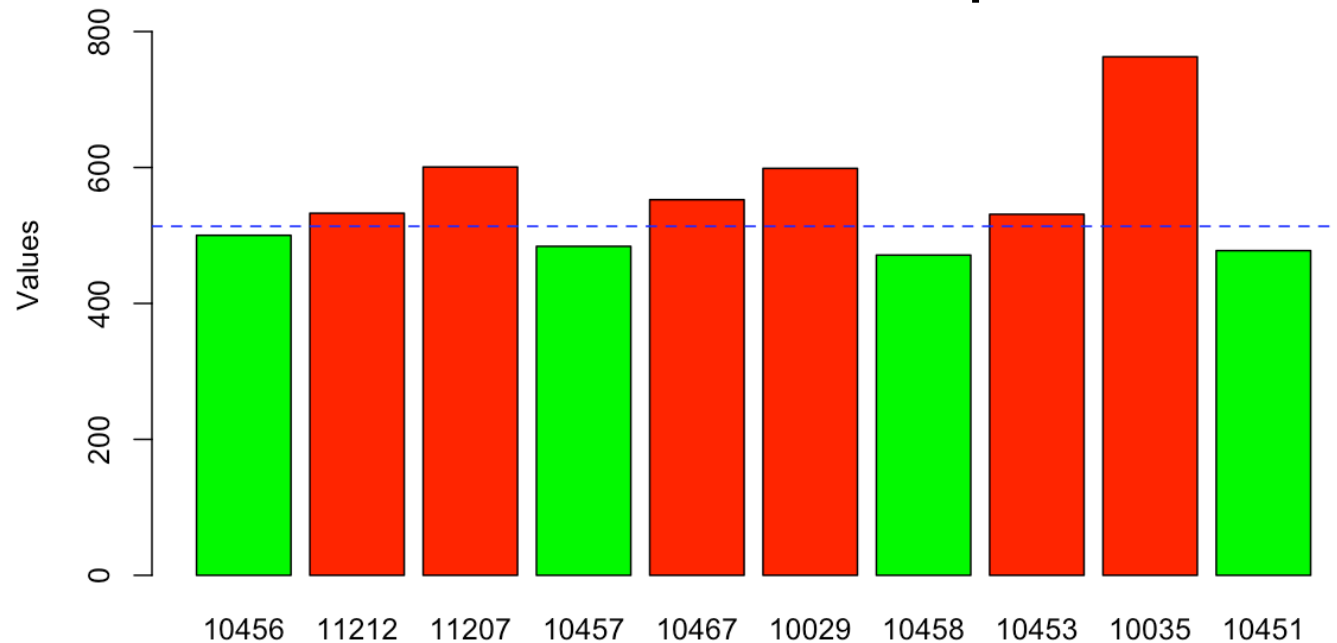
- In this, we are first calculating the number of incidents per zip code. The top 10 zip codes with the highest incident counts are considered.
- We used the maps API from Google Cloud to generate an NYC map. Zip codes have been plotted on the map as circles proportional to their incident counts.
- This plot could help the department to stock in additional resources for the areas which are more prone.



# GEOGRAPHICAL DISTRIBUTION OF AREAS WITH THE MOST INCIDENTS

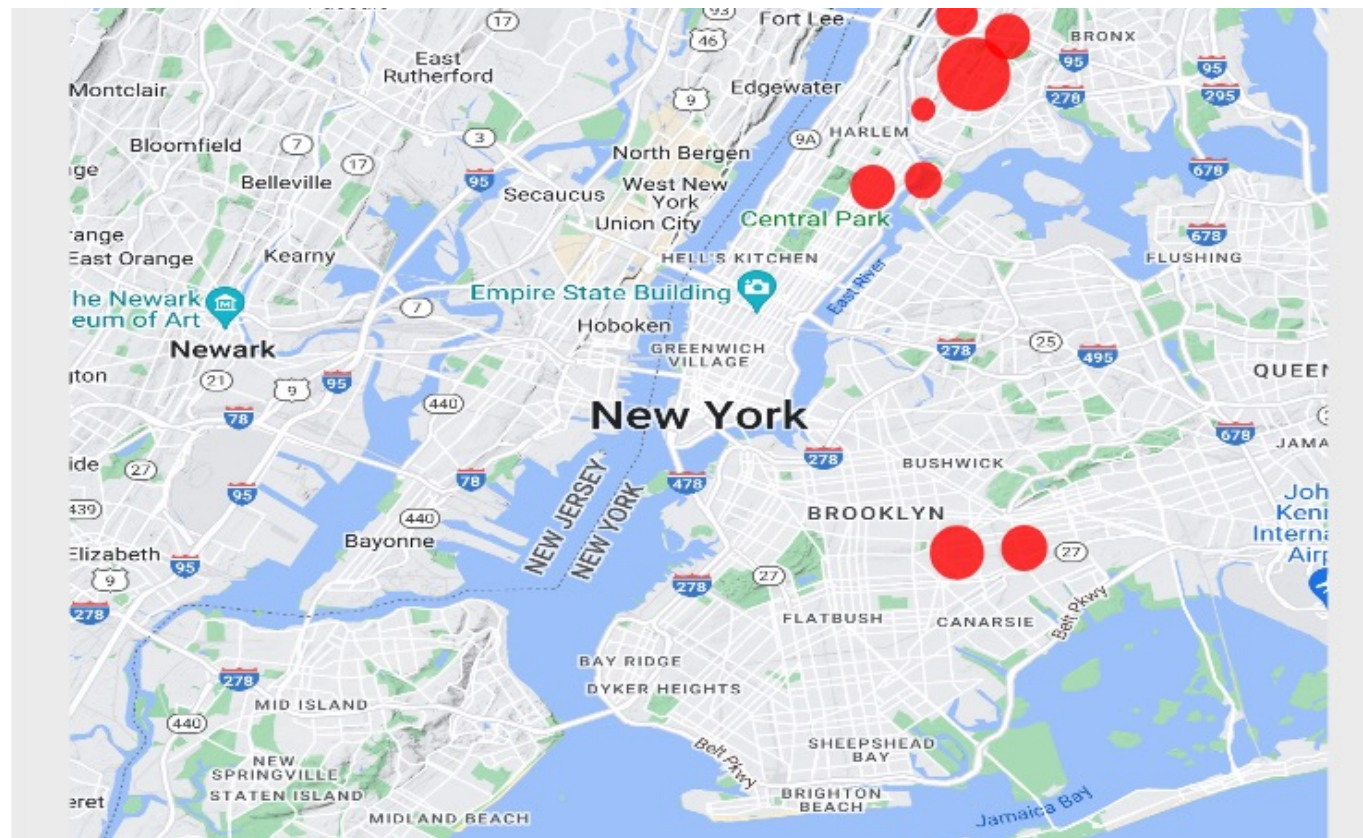
# HOTSPOT RESPONSE TIMES

Although many hotspot zones have effectively managed to cope with the higher incident count, there are still some areas where improvements could be made.





Latitude



**GEOGRAPHICAL DISTRIBUTION OF AREAS WITH THE HIGHEST RESPONSE TIME**

# CONCLUSION



- Overall, our analysis has revealed some key insights into the NYC EMS response system.
- Through our examination of call volume, hotspot zones, etc. we were able to identify key trends and patterns which can be helpful for the department.
- Overall, these findings suggest that there may be opportunities for improving the efficiency and effectiveness of the EMS response system in NYC through targeted resource allocation and strategic planning.