Problem Set 6 - Waze Shiny Dashboard

Peter Ganong, Maggie Shi, and Andre Oviedo

2024-11-24

1. **ps6:** Due Sat 23rd at 5:00PM Central. Worth 100 points (80 points from questions, 10 points for correct submission and 10 points for code style) + 10 extra credit.

We use (`*`) to indicate a problem that we think might be time consuming.

Steps to submit (10 points on PS6)

1. "This submission is my work alone and complies with the 30538 integrity policy." Add your initials to indicate your agreement: **___**
2. "I have uploaded the names of anyone I worked with on the problem set **here**" **___** (2 point)
3. Late coins used this pset: **___** Late coins left after submission: **___**
4. Before starting the problem set, make sure to read and agree to the terms of data usage for the Waze data here.
5. Knit your `ps6.qmd` as a pdf document and name it `ps6.pdf`.
6. Push your `ps6.qmd`, `ps6.pdf`, `requirements.txt`, and all created folders (we will create three Shiny apps so you will have at least three additional folders) to your Github repo (5 points). It is fine to use Github Desktop.
7. Submit `ps6.pdf` and also link your Github repo via Gradescope (5 points)
8. Tag your submission in Gradescope. For the Code Style part (10 points) please tag the whole correspondingsection for the code style rubric.

Notes: see the Quarto documentation (link) for directions on inserting images into your knitted document.

IMPORTANT: For the App portion of the PS, in case you can not arrive to the expected functional dashboard we will need to take a look at your `app.py` file. You can use the following *code chunk template to "import" and print the content of that file. Please, don't forget to also tag the corresponding code chunk as part of your submission!*

````python
def print_file_contents(file_path):
    """Print contents of a file."""
    try:
        with open(file_path, 'r') as f:
            content = f.read()
            print("```python")
            print(content)
            print("```")
    except FileNotFoundError:
        print("```python")
        print(f"Error: File '{file_path}' not found")
        print("```")
    except Exception as e:
        print("```python")
        print(f"Error reading file: {e}")
        print("```")

print_file_contents("./top_alerts_map_byhour/app.py") # Change accordingly
````

Background

Data Download and Exploration (20 points)

1.

````python
# Load the waze_data_sample.csv file
waze_sample_df = pd.read_csv('./waze_data/waze_data_sample.csv')

# Exclude 'ts', 'geo', 'geoWKT' and determine Altair-style data types
altair_data_types = []
for column in waze_sample_df.columns:
    if column not in ['ts', 'geo', 'geoWKT']:
        dtype = waze_sample_df[column].dtype
        if pd.api.types.is_numeric_dtype(dtype):
            altair_data_types.append((column, 'Quantitative'))
        elif pd.api.types.is_datetime64_any_dtype(dtype):
            altair_data_types.append((column, 'Temporal'))
````

```
        else:
            altair_data_types.append((column, 'Nominal'))

# Print variable names and data types
print(altair_data_types)
```

```
[('Unnamed: 0', 'Quantitative'), ('city', 'Nominal'), ('confidence',
'Quantitative'), ('nThumbsUp', 'Quantitative'), ('street', 'Nominal'),
('uuid', 'Nominal'), ('country', 'Nominal'), ('type', 'Nominal'), ('subtype',
'Nominal'), ('roadType', 'Quantitative'), ('reliability', 'Quantitative'),
('magvar', 'Quantitative'), ('reportRating', 'Quantitative')]
```

2.

```
waze_df = pd.read_csv('./waze_data/waze_data.csv')

# Calculate NULL and non-NULL counts for each column
null_counts = waze_df.isnull().sum().reset_index()
not_null_counts = waze_df.notnull().sum().reset_index()

# Rename columns for Altair compatibility
null_counts.columns = ['variable', 'null_count']
not_null_counts.columns = ['variable', 'not_null_count']

# Merge into a single DataFrame
null_data = pd.merge(null_counts, not_null_counts, on='variable')
null_data = null_data.melt(id_vars='variable',
                           value_vars=['null_count', 'not_null_count'],
                           var_name='status',
                           value_name='count')

# Replace status labels for clarity
null_data['status'] = null_data['status'].replace({
    'null_count': 'NULL',
    'not_null_count': 'Not NULL'
})

# Create a stacked bar chart
chart = alt.Chart(null_data).mark_bar().encode(
    x=alt.X('variable:N', title='Variables',
 ↪  sort=null_data['variable'].unique()),
    y=alt.Y('count:Q', title='Count'),
    color=alt.Color('status:N', title='Status', scale=alt.Scale(scheme='tableau20'))
).properties(
    title='Stacked Bar Chart of NULL vs Not NULL Observations',
    width=800,
    height=400
).configure_axis(
    labelAngle=45
)

chart.show()
chart.save('./plots/plot1.png')
```

```
alt.Chart(...)
```

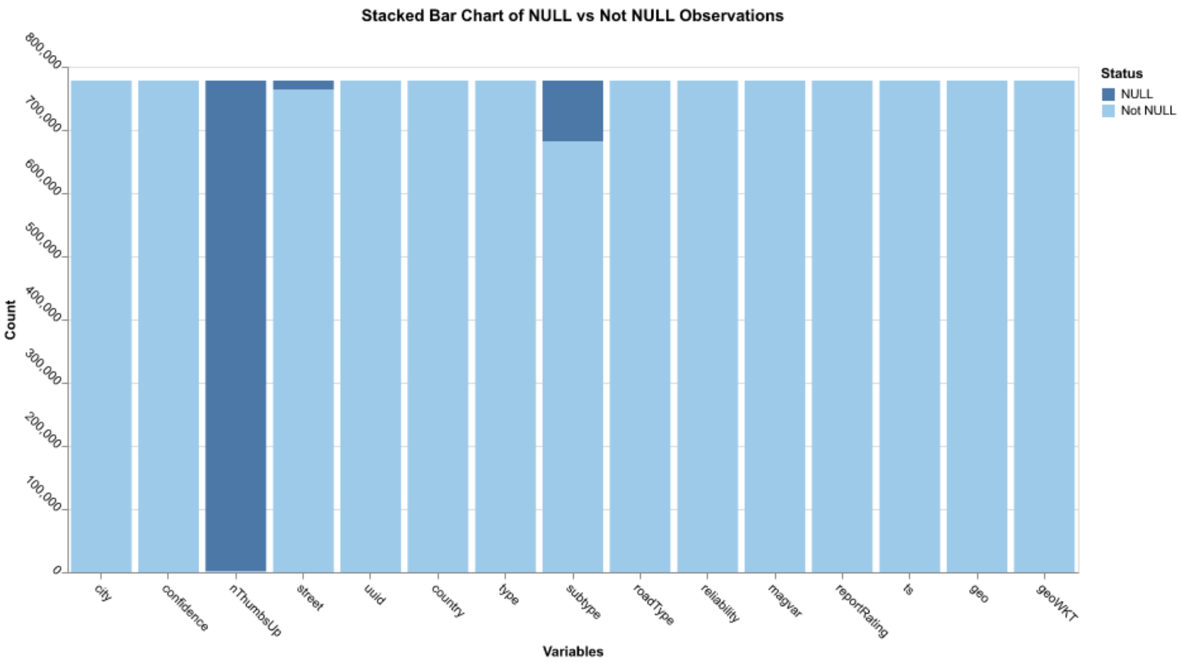


Figure 1: Image1

3.

```
# Extract unique values for 'type' and 'subtype'
unique_types = waze_df['type'].fillna('Unclassified').unique()
unique_subtypes = waze_df['subtype'].fillna('Unclassified').unique()
```

```
#sort_subtypes = waze_sample_df.apply(
#    lambda row: f"{row['type']} Unclassified" if pd.isna(row['subtype'])
 ↪  else row['subtype'],
#    axis=1
#)
# Count how many types have a 'subtype' that is NA
types_with_na_subtype = waze_df[waze_df['subtype'].isna()]['type'].unique()
num_types_with_na_subtype = len(types_with_na_subtype)

# Group types and subtypes to determine hierarchy
type_subtype_counts = waze_df.groupby(['type',
 ↪  'subtype']).size().reset_index(name='count')
types_with_informative_subtypes = type_subtype_counts['type'].value_counts()


unique_types, unique_subtypes, types_with_na_subtype,
 ↪  num_types_with_na_subtype
```

```
(array(['JAM', 'ACCIDENT', 'ROAD_CLOSED', 'HAZARD'], dtype=object),
 array(['Unclassified', 'ACCIDENT_MAJOR', 'ACCIDENT_MINOR',
        'HAZARD_ON_ROAD', 'HAZARD_ON_ROAD_CAR_STOPPED',
        'HAZARD_ON_ROAD_CONSTRUCTION', 'HAZARD_ON_ROAD_EMERGENCY_VEHICLE',
        'HAZARD_ON_ROAD_ICE', 'HAZARD_ON_ROAD_OBJECT',
        'HAZARD_ON_ROAD_POT_HOLE', 'HAZARD_ON_ROAD_TRAFFIC_LIGHT_FAULT',
        'HAZARD_ON_SHOULDER', 'HAZARD_ON_SHOULDER_CAR_STOPPED',
        'HAZARD_WEATHER', 'HAZARD_WEATHER_FLOOD', 'JAM_HEAVY_TRAFFIC',
        'JAM_MODERATE_TRAFFIC', 'JAM_STAND_STILL_TRAFFIC',
        'ROAD_CLOSED_EVENT', 'HAZARD_ON_ROAD_LANE_CLOSED',
        'HAZARD_WEATHER_FOG', 'ROAD_CLOSED_CONSTRUCTION',
        'HAZARD_ON_ROAD_ROAD_KILL', 'HAZARD_ON_SHOULDER_ANIMALS',
        'HAZARD_ON_SHOULDER_MISSING_SIGN', 'JAM_LIGHT_TRAFFIC',
        'HAZARD_WEATHER_HEAVY_SNOW', 'ROAD_CLOSED_HAZARD',
        'HAZARD_WEATHER_HAIL'], dtype=object),
 array(['JAM', 'ACCIDENT', 'ROAD_CLOSED', 'HAZARD'], dtype=object),
 4)
```

```
print(f"There are {num_types_with_na_subtype} types that have subtypes in
 ↪  NA")
```

```
There are 4 types that have subtypes in NA
```

There might be types like HAZARD_ON_ROAD_ROAD_KILL, means haraed, on road, road kill, road kill would be subsubtype.

```python
subtype_total = len(waze_sample_df[waze_sample_df['subtype'].isna()])
subtype_total
```

```
1004
```

- Accident
  - Major
  - Minor
  - Unclassified
- Hazard
  - On Road
    - Car Stopped
    - Construction
    - Emergency Vehicle
    - Ice
    - Lane Closed
    - Object
    - Pot Hole
    - Road Kill
    - Traffic Light Fault
  - On Shoulder
    - Animals
    - Car Stopped
    - Missing Sign
  - Weather
    - Flood
    - Fog
    - Hail
    - Heavy Snow
  - Unclassified
- Jam
  - Heavy Traffic
  - Light Traffic
  - Moderate Traffic
  - Stand Still Traffic
  - Unclassified

- Road Closed
    - Construction
    - Event
    - Hazard
    - Unclassified

We should keep the NA as Unclassified since there are too many columns that are NA, but have corresponding types. Simply dropping them will lose too much information. 4.

```
# Manually define the crosswalk for all levels of the hierarchy
crosswalk_data = [
    # ACCIDENT
    {"type": "ACCIDENT", "subtype": "ACCIDENT_MINOR", "updated_type":
↪   "Accident", "updated_subtype": "Minor", "updated_subsubtype": "Minor"},
    {"type": "ACCIDENT", "subtype": "ACCIDENT_MAJOR", "updated_type":
↪   "Accident", "updated_subtype": "Major", "updated_subsubtype": "Major"},
    {"type": "ACCIDENT", "subtype": "Unclassified", "updated_type":
↪   "Accident", "updated_subtype": "Unclassified", "updated_subsubtype":
↪   "Unclassified"},

    # JAM
    {"type": "JAM", "subtype": "JAM_MODERATE_TRAFFIC", "updated_type": "Jam",
↪   "updated_subtype": "Traffic", "updated_subsubtype": "Moderate"},
    {"type": "JAM", "subtype": "JAM_HEAVY_TRAFFIC", "updated_type": "Jam",
↪   "updated_subtype": "Traffic", "updated_subsubtype": "Heavy"},
    {"type": "JAM", "subtype": "JAM_STAND_STILL_TRAFFIC", "updated_type":
↪   "Jam", "updated_subtype": "Traffic", "updated_subsubtype": "Stand
↪   Still"},
    {"type": "JAM", "subtype": "JAM_LIGHT_TRAFFIC", "updated_type": "Jam",
↪   "updated_subtype": "Traffic", "updated_subsubtype": "Light"},
    {"type": "JAM", "subtype": "Unclassified", "updated_type": "Jam",
↪   "updated_subtype": "Unclassified", "updated_subsubtype": "Unclassified"},

    # WEATHERHAZARD / HAZARD
    {"type": "HAZARD", "subtype": "HAZARD_ON_ROAD", "updated_type": "Hazard",
↪   "updated_subtype": "On Road", "updated_subsubtype": "Unclassified"},
    {"type": "HAZARD", "subtype": "HAZARD_ON_SHOULDER", "updated_type":
↪   "Hazard", "updated_subtype": "On Shoulder", "updated_subsubtype":
↪   "Unclassified"},
    {"type": "HAZARD", "subtype": "HAZARD_WEATHER", "updated_type": "Hazard",
↪   "updated_subtype": "Weather", "updated_subsubtype": "Unclassified"},
    {"type": "HAZARD", "subtype": "HAZARD_ON_ROAD_OBJECT", "updated_type":
↪   "Hazard", "updated_subtype": "On Road", "updated_subsubtype": "Object"},
    {"type": "HAZARD", "subtype": "HAZARD_ON_ROAD_POT_HOLE", "updated_type":
↪   "Hazard", "updated_subtype": "On Road", "updated_subsubtype": "Pot
↪   Hole"},
    {"type": "HAZARD", "subtype": "HAZARD_ON_ROAD_ROAD_KILL", "updated_type":
↪   "Hazard", "updated_subtype": "On Road", "updated_subsubtype": "Road
↪   Kill"},
    {"type": "HAZARD", "subtype": "HAZARD_ON_SHOULDER_CAR_STOPPED",
↪   "updated_type": "Hazard", "updated_subtype": "On Shoulder",
↪   "updated_subsubtype": "Car Stopped"},
    {"type": "HAZARD", "subtype": "HAZARD_ON_SHOULDER_ANIMALS",
↪   "updated_type": "Hazard", "updated_subtype": "On Shoulder",
↪   "updated_subsubtype": "Animals"},
    {"type": "HAZARD", "subtype": "HAZARD_ON_SHOULDER_MISSING_SIGN",
↪   "updated_type": "Hazard", "updated_subtype": "On Shoulder",
↪   "updated_subsubtype": "Missing Sign"},
    {"type": "HAZARD", "subtype": "HAZARD_WEATHER_FOG", "updated_type":
↪   "Hazard", "updated_subtype": "Weather", "updated_subsubtype": "Fog"},
    {"type": "HAZARD", "subtype": "HAZARD_WEATHER_HAIL", "updated_type":
↪   "Hazard", "updated_subtype": "Weather", "updated_subsubtype": "Hail"},
    {"type": "HAZARD", "subtype": "HAZARD_WEATHER_HEAVY_SNOW",
↪   "updated_type": "Hazard", "updated_subtype": "Weather",
↪   "updated_subsubtype": "Heavy Snow"},
    {"type": "HAZARD", "subtype": "HAZARD_WEATHER_FLOOD", "updated_type":
↪   "Hazard", "updated_subtype": "Weather", "updated_subsubtype": "Flood"},
    {"type": "HAZARD", "subtype": "HAZARD_ON_ROAD_LANE_CLOSED",
↪   "updated_type": "Hazard", "updated_subtype": "On Road",
↪   "updated_subsubtype": "Lane Closed"},
    {"type": "HAZARD", "subtype": "HAZARD_ON_ROAD_ICE", "updated_type":
↪   "Hazard", "updated_subtype": "On Road", "updated_subsubtype": "Ice"},
    {"type": "HAZARD", "subtype": "HAZARD_ON_ROAD_CONSTRUCTION",
↪   "updated_type": "Hazard", "updated_subtype": "On Road",
↪   "updated_subsubtype": "Construction"},
    {"type": "HAZARD", "subtype": "HAZARD_ON_ROAD_CAR_STOPPED",
↪   "updated_type": "Hazard", "updated_subtype": "On Road",
↪   "updated_subsubtype": "Car Stopped"},
    {"type": "HAZARD", "subtype": "HAZARD_ON_ROAD_TRAFFIC_LIGHT_FAULT",
↪   "updated_type": "Hazard", "updated_subtype": "On Road",
↪   "updated_subsubtype": "Traffic Light Fault"},
    {"type": "HAZARD", "subtype": "HAZARD_ON_ROAD_EMERGENCY_VEHICLE",
↪   "updated_type": "Hazard", "updated_subtype": "On Road",
↪   "updated_subsubtype": "Emergency Vehicle"},
    {"type": "HAZARD", "subtype": "Unclassified", "updated_type": "Hazard",
↪   "updated_subtype": "Unclassified", "updated_subsubtype": "Unclassified"},
    # ROAD_CLOSED
    {"type": "ROAD_CLOSED", "subtype": "ROAD_CLOSED_HAZARD", "updated_type":
 ↪  "Road Closed", "updated_subtype": "Hazard", "updated_subsubtype":
 ↪  "Unclassified"},
    {"type": "ROAD_CLOSED", "subtype": "ROAD_CLOSED_CONSTRUCTION",
 ↪  "updated_type": "Road Closed", "updated_subtype": "Construction",
 ↪  "updated_subsubtype": "Unclassified"},
    {"type": "ROAD_CLOSED", "subtype": "ROAD_CLOSED_EVENT", "updated_type":
 ↪  "Road Closed", "updated_subtype": "Event", "updated_subsubtype":
 ↪  "Unclassified"},
    {"type": "ROAD_CLOSED", "subtype": "Unclassified", "updated_type": "Road
 ↪  Closed", "updated_subtype": "Unclassified", "updated_subsubtype":
 ↪  "Unclassified"},
]

# Convert the crosswalk into a DataFrame
crosswalk_df = pd.DataFrame(crosswalk_data)

# Replace NA subtypes in the original dataset with None
waze_df['subtype'] = waze_df['subtype'].replace({pd.NA: "Unclassified"})

# Merge the crosswalk with the original dataset
merged_data = pd.merge(
    waze_df,
    crosswalk_df,
    how='left',
    on=['type', 'subtype']
)

# Verify the results
print("Crosswalk DataFrame:")
print(crosswalk_df.head())

print("\nMerged Dataset:")
print(merged_data.head())
```

```
Crosswalk DataFrame:
       type               subtype updated_type updated_subtype  \
0  ACCIDENT        ACCIDENT_MINOR     Accident           Minor
1  ACCIDENT        ACCIDENT_MAJOR     Accident           Major
```

```
2  ACCIDENT          Unclassified     Accident    Unclassified
3       JAM  JAM_MODERATE_TRAFFIC          Jam         Traffic
4       JAM     JAM_HEAVY_TRAFFIC          Jam         Traffic

  updated_subsubtype
0              Minor
1              Major
2       Unclassified
3           Moderate
4              Heavy

Merged Dataset:
          city  confidence  nThumbsUp street  \
0  Chicago, IL           0        NaN    NaN
1  Chicago, IL           1        NaN    NaN
2  Chicago, IL           0        NaN    NaN
3  Chicago, IL           0        NaN  Alley
4  Chicago, IL           0        NaN  Alley

                                   uuid country         type       subtype  \
0  004025a4-5f14-4cb7-9da6-2615daafbf37      US          JAM  Unclassified
1  ad7761f8-d3cb-4623-951d-dafb419a3ec3      US     ACCIDENT  Unclassified
2  0e5f14ae-7251-46af-a7f1-53a5272cd37d      US  ROAD_CLOSED  Unclassified
3  654870a4-a71a-450b-9f22-bc52ae4f69a5      US          JAM  Unclassified
4  926ff228-7db9-4e0d-b6cf-6739211ffc8b      US          JAM  Unclassified

   roadType  reliability  magvar  reportRating                       ts  \
0        20            5     139             3  2024-02-04 16:40:41 UTC
1         4            8       2             2  2024-02-04 20:01:27 UTC
2         1            5     344             2  2024-02-04 02:15:54 UTC
3        20            5     264             2  2024-02-04 00:30:54 UTC
4        20            5     359             0  2024-02-04 03:27:35 UTC

                           geo                      geoWKT updated_type  \
0  POINT(-87.676685 41.929692)  Point(-87.676685 41.929692)          Jam
1  POINT(-87.624816 41.753358)  Point(-87.624816 41.753358)     Accident
2  POINT(-87.614122 41.889821)  Point(-87.614122 41.889821)  Road Closed
3  POINT(-87.680139 41.939093)  Point(-87.680139 41.939093)          Jam
4   POINT(-87.735235 41.91658)   Point(-87.735235 41.91658)          Jam

  updated_subtype updated_subsubtype
0    Unclassified       Unclassified
1    Unclassified       Unclassified
```

```
2    Unclassified       Unclassified
3    Unclassified       Unclassified
4    Unclassified       Unclassified
```

Below is the way of checking if they have the same type and subtype.

```
unique_types_merged = merged_data['type'].unique()
unique_subtypes_merged = merged_data['subtype'].unique()
crosswalk_df = pd.DataFrame(crosswalk_data)
unique_types_crosswalk = crosswalk_df['type'].unique()
unique_subtypes_crosswalk = crosswalk_df['subtype'].unique()
print(np.array_equal(np.sort(unique_types_crosswalk),
 ↪  np.sort(unique_types_merged)))
print(np.array_equal(np.sort(unique_subtypes_crosswalk),
 ↪  np.sort(unique_subtypes_merged)))
```

```
True
True
```

App #1: Top Location by Alert Type Dashboard (30 points)

1.

```
import re

def extract_coordinates(geo_string):
    match = re.match(r"POINT\((-?\d+\.\d+)\s(-?\d+\.\d+)\)", geo_string)
    if match:
        longitude, latitude = match.groups()
        return float(latitude), float(longitude)
    return None, None

# Apply the function to extract latitude and longitude
merged_data['latitude'], merged_data['longitude'] =
 ↪  zip(*merged_data['geo'].apply(extract_coordinates))
```

```
# Bin latitude and longitude
merged_data["longitude_bin"] = (np.floor(merged_data["longitude"] * 100) /
 ↪  100).apply(lambda x: f"{x:.6f}")
merged_data["latitude_bin"] = (np.floor(merged_data["latitude"] * 100) /
 ↪  100).apply(lambda x: f"{x:.6f}")
```

```
# Count the occurrences of binned combinations
binned_counts = merged_data.groupby(["latitude_bin",
 ↪  "longitude_bin"]).size().reset_index(name="count")

# Find the binned latitude-longitude combination with the greatest count
max_binned_combination = binned_counts[binned_counts["count"] ==
 ↪  binned_counts["count"].max()]

binned_counts, max_binned_combination
```

```
(    latitude_bin longitude_bin  count
 0       41.640000    -87.560000     21
 1       41.640000    -87.580000    290
 2       41.640000    -87.590000    140
 3       41.640000    -87.620000     14
 4       41.650000    -87.560000     67
 ..            ...           ...    ...
 707     42.010000    -87.830000     25
 708     42.010000    -87.840000      1
 709     42.010000    -87.870000      9
 710     42.020000    -87.670000    123
 711     42.020000    -87.680000     19

 [712 rows x 3 columns],
     latitude_bin longitude_bin  count
 586     41.960000    -87.750000  26540)
```

```
chosen_type = "JAM"
chosen_subtype = "JAM_STAND_STILL_TRAFFIC"
filtered_df = merged_data[(merged_data["type"] == chosen_type) &
 ↪  (merged_data["subtype"] == chosen_subtype)]

# Aggregate data to find the top 10 latitude-longitude bins with the most
 ↪  alerts
alert_counts = (
    filtered_df.groupby(["latitude_bin", "longitude_bin"])
    .size()
    .reset_index(name="alert_count")
    .head(10)
)
```

```
alert_counts
```

	latitude_bin	longitude_bin	alert_count
0	41.640000	-87.560000	1
1	41.640000	-87.580000	8
2	41.640000	-87.590000	5
3	41.650000	-87.560000	2
4	41.650000	-87.570000	23
5	41.650000	-87.580000	21
6	41.650000	-87.590000	132
7	41.650000	-87.610000	1
8	41.650000	-87.620000	17
9	41.660000	-87.560000	10

```
df_alert_counts = (
    merged_data.groupby(["latitude_bin", "longitude_bin", "type", "subtype",
 ↪  "updated_type", "updated_subtype", "updated_subsubtype"])
    .size()
    .reset_index(name="alert_count")
    .sort_values(by="alert_count", ascending=False)
)
```

```
#df_alert_counts_path = './top_alerts_map/df_alert.csv'
#df_alert_counts.to_csv(df_alert_counts_path, index=False)
```

The level of aggraration is [“latitude_bin”, “longitude_bin”, “type”, “subtype”, “updated_type”, “updated_subtype”, “updated_subsubtype”], the rows are 11231.

```
#merged_data.to_csv('./df_merged_data.csv', index=False)
```

2.

```
chosen_type = "ROAD_CLOSED"
chosen_subtype_heavy = "ROAD_CLOSED_EVENT"
filtered_heavy_df = merged_data[(merged_data["type"] == chosen_type) &
 ↪  (merged_data["subtype"] == chosen_subtype_heavy)]
```

```
# Aggregate data to find the top 10 latitude-longitude bins with the most
 ↪  alerts
alert_counts_heavy = (
    filtered_heavy_df.groupby(["latitude_bin", "longitude_bin"])
    .size()
    .reset_index(name="alert_count")
    .sort_values(by="alert_count", ascending=False)
    .head(10)
)

# Create scatter plot
scatter_plot = (
    alt.Chart(alert_counts_heavy)
    .mark_circle()
    .encode(
        x=alt.X("longitude_bin:Q", title="Longitude",
 ↪  scale=alt.Scale(domain=[(merged_data['longitude_bin'].max()),
 ↪  (merged_data['longitude_bin'].min())])),
        y=alt.Y("latitude_bin:Q", title="Latitude",
 ↪  scale=alt.Scale(domain=[(merged_data['latitude_bin'].min()),
 ↪  (merged_data['latitude_bin'].max())])),
        color=alt.Color("alert_count:Q", title="Number of Alerts"),
        tooltip=["latitude_bin", "longitude_bin", "alert_count"],
    )
    .properties(
        title="Top 10 Latitude-Longitude Bins with Highest 'Jam - Heavy
 ↪  Traffic' Alerts",
        width=400,
        height=400,
    )
)

scatter_plot.save('./plots/plot2.png')
scatter_plot
```

```
alt.Chart(...)
```

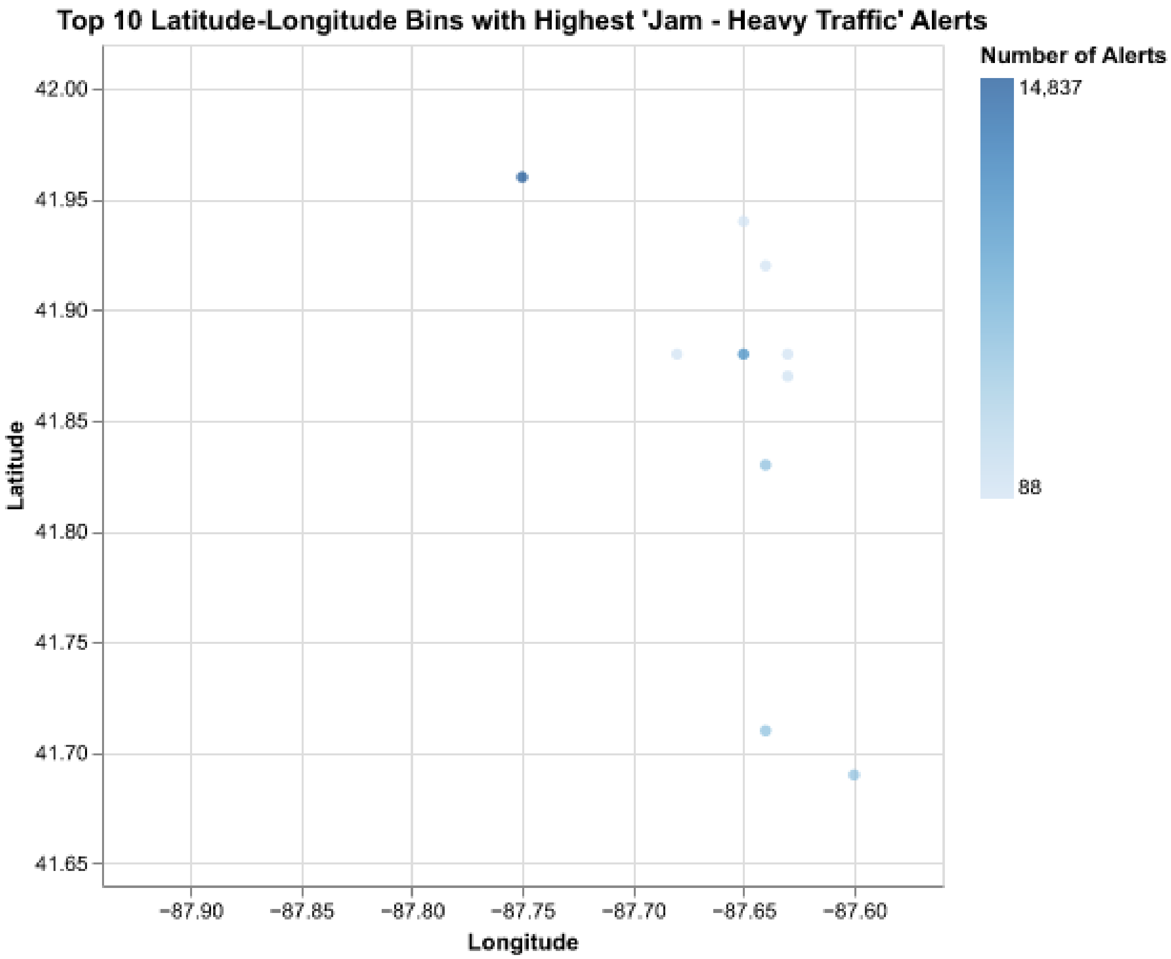


Figure 2: Image1

3.

```
file_path = "./top_alerts_map/chicago-boundaries.geojson"
#----
with open(file_path) as f:
    chicago_geojson = json.load(f)
geo_data = alt.Data(values=chicago_geojson["features"])
```

4.

```
points = alt.Chart(alert_counts_heavy).mark_circle().encode(
    longitude=alt.X("longitude_bin:Q"),
    latitude=alt.Y("latitude_bin:Q"),
    tooltip=["latitude_bin", "longitude_bin", "alert_count"],

)
map_layer = (
    alt.Chart(geo_data).mark_geoshape(fill="lightgray", stroke="black")
    .properties(
        width=400,
        height=400
    )
    .project("identity", reflectY=True)  # Ensure correct alignment with
 ↪  coordinates
)

combined_plot = (
    map_layer + scatter_plot
).properties(title="Top 10")

combined_plot.save('./plots/plot3.png')
combined_plot
```

```
alt.LayerChart(...)
```

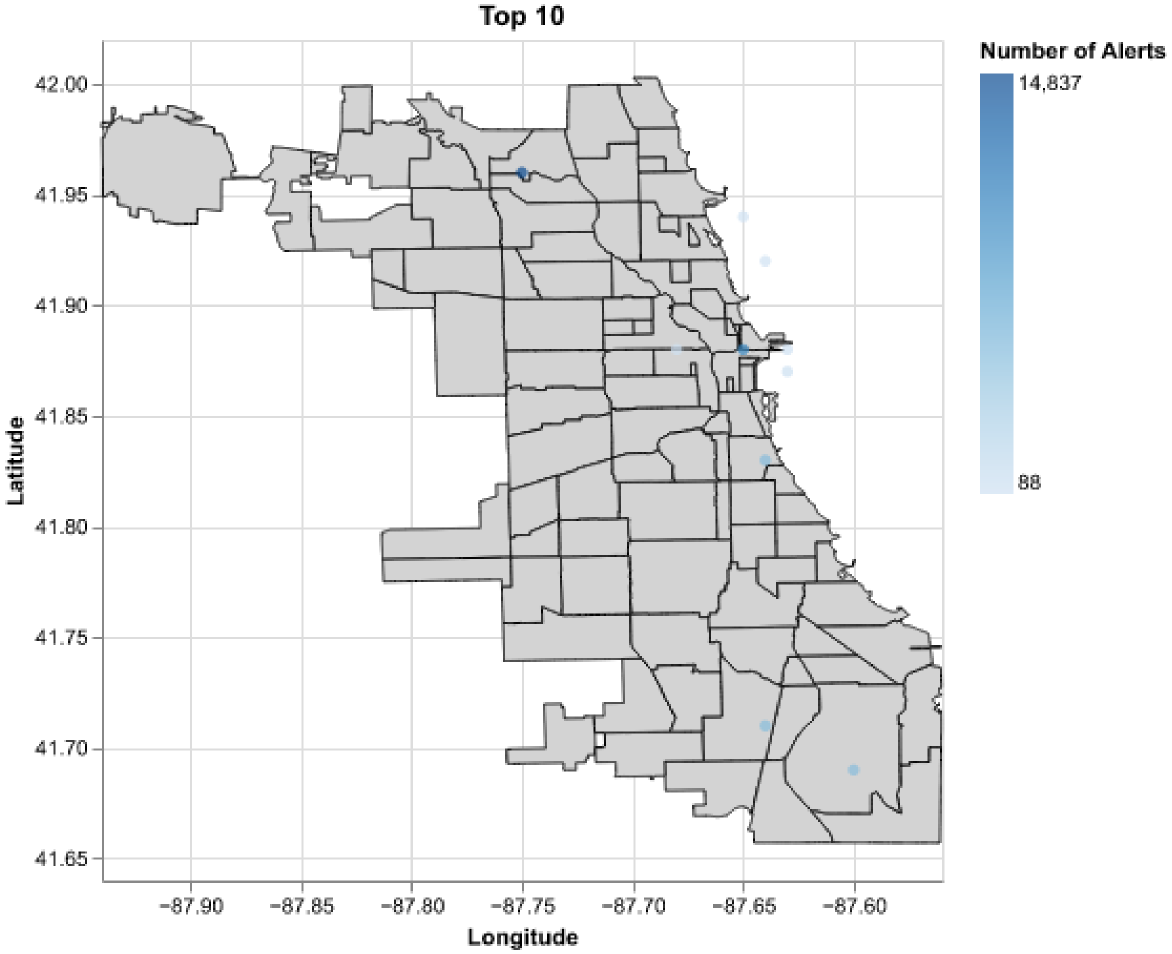
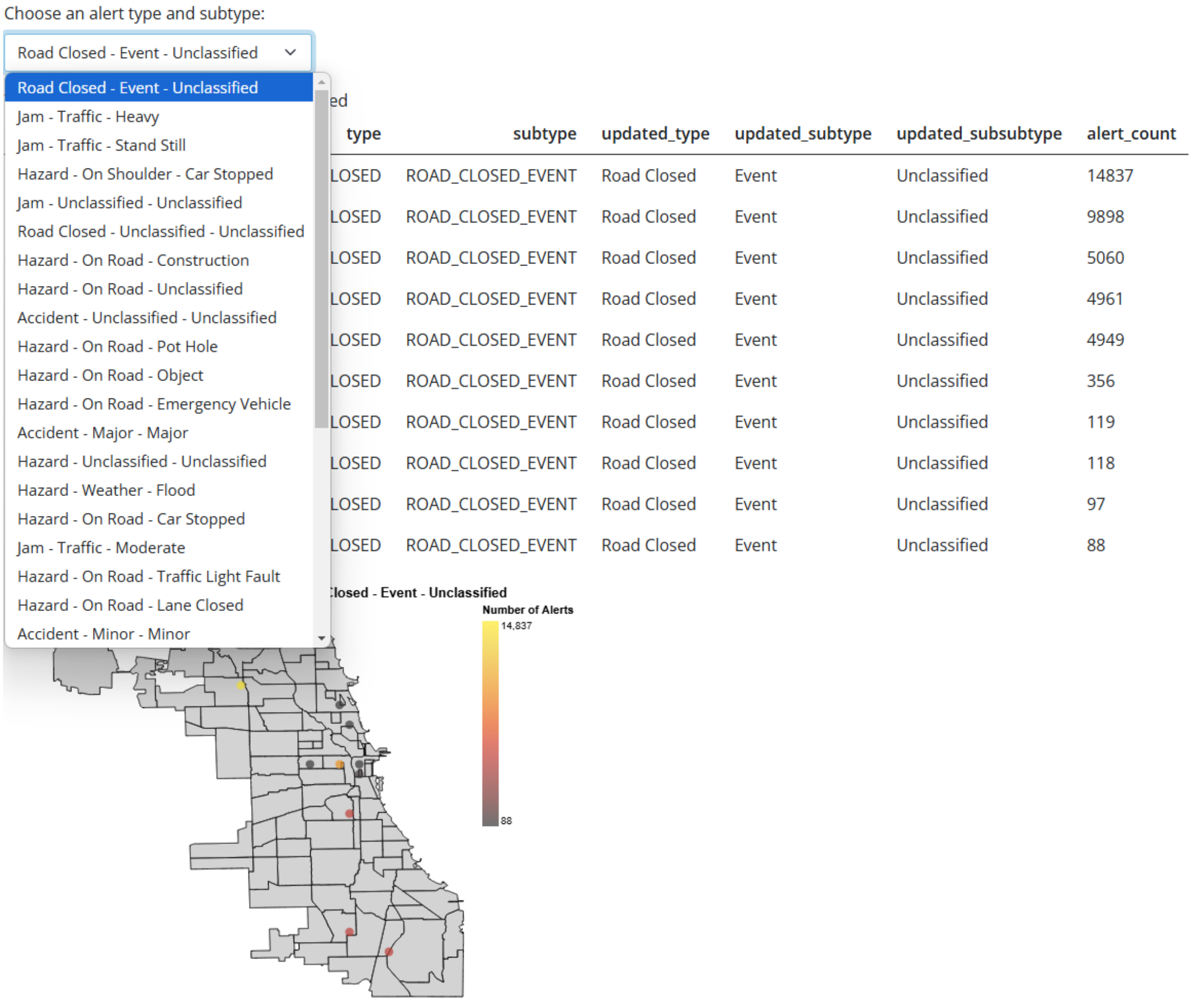


Figure 3: Image1

The plot have points that diverges from the original location, on the lake. I removed the coordinates the next graph, used the latitude coordinates, and the points are back to normal.

5.



Choose an alert type and subtype:

Road Closed - Event - Unclassified

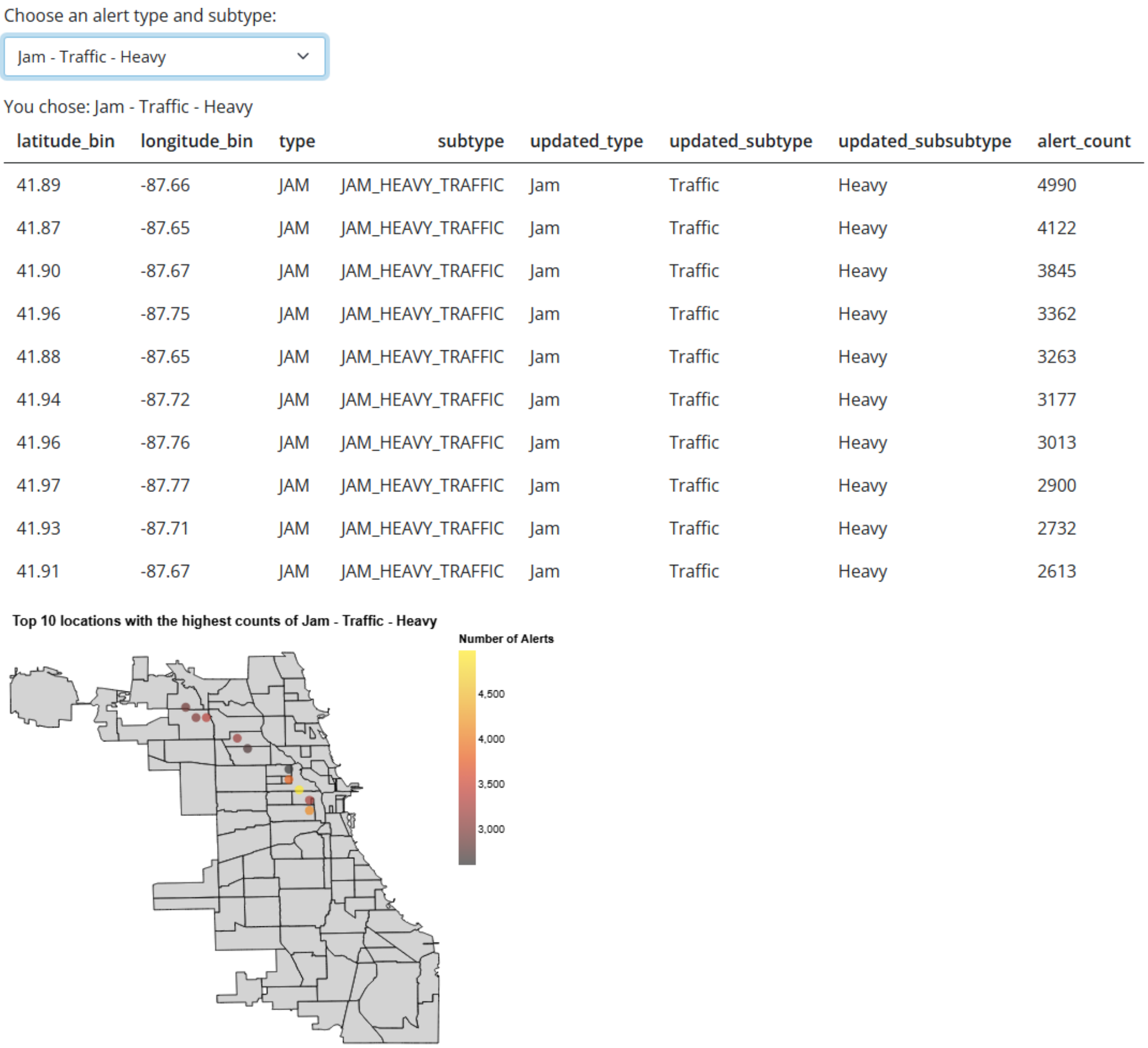
Road Closed - Event - Unclassified
Jam - Traffic - Heavy
Jam - Traffic - Stand Still
Hazard - On Shoulder - Car Stopped
Jam - Unclassified - Unclassified
Road Closed - Unclassified - Unclassified
Hazard - On Road - Construction
Hazard - On Road - Unclassified
Accident - Unclassified - Unclassified
Hazard - On Road - Pot Hole
Hazard - On Road - Object
Hazard - On Road - Emergency Vehicle
Accident - Major - Major
Hazard - Unclassified - Unclassified
Hazard - Weather - Flood
Hazard - On Road - Car Stopped
Jam - Traffic - Moderate
Hazard - On Road - Traffic Light Fault
Hazard - On Road - Lane Closed
Accident - Minor - Minor

ed

type	subtype	updated_type	updated_subtype	updated_subsubtype	alert_count
LOSED	ROAD_CLOSED_EVENT	Road Closed	Event	Unclassified	14837
LOSED	ROAD_CLOSED_EVENT	Road Closed	Event	Unclassified	9898
LOSED	ROAD_CLOSED_EVENT	Road Closed	Event	Unclassified	5060
LOSED	ROAD_CLOSED_EVENT	Road Closed	Event	Unclassified	4961
LOSED	ROAD_CLOSED_EVENT	Road Closed	Event	Unclassified	4949
LOSED	ROAD_CLOSED_EVENT	Road Closed	Event	Unclassified	356
LOSED	ROAD_CLOSED_EVENT	Road Closed	Event	Unclassified	119
LOSED	ROAD_CLOSED_EVENT	Road Closed	Event	Unclassified	118
LOSED	ROAD_CLOSED_EVENT	Road Closed	Event	Unclassified	97
LOSED	ROAD_CLOSED_EVENT	Road Closed	Event	Unclassified	88

Figure 4: Image1

32 types



Choose an alert type and subtype:

Jam - Traffic - Heavy

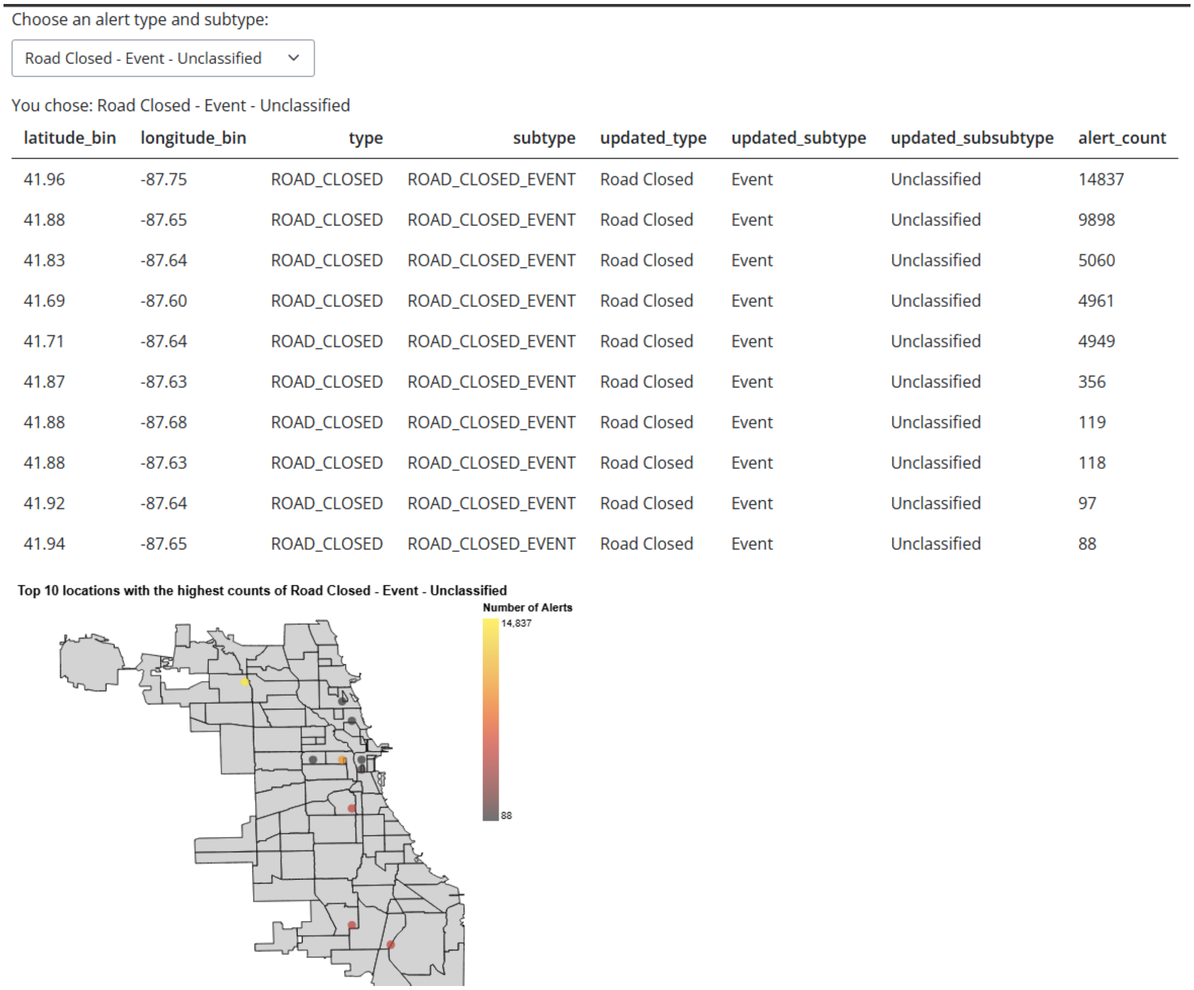
You chose: Jam - Traffic - Heavy

latitude_bin	longitude_bin	type	subtype	updated_type	updated_subtype	updated_subsubtype	alert_count
41.89	-87.66	JAM	JAM_HEAVY_TRAFFIC	Jam	Traffic	Heavy	4990
41.87	-87.65	JAM	JAM_HEAVY_TRAFFIC	Jam	Traffic	Heavy	4122
41.90	-87.67	JAM	JAM_HEAVY_TRAFFIC	Jam	Traffic	Heavy	3845
41.96	-87.75	JAM	JAM_HEAVY_TRAFFIC	Jam	Traffic	Heavy	3362
41.88	-87.65	JAM	JAM_HEAVY_TRAFFIC	Jam	Traffic	Heavy	3263
41.94	-87.72	JAM	JAM_HEAVY_TRAFFIC	Jam	Traffic	Heavy	3177
41.96	-87.76	JAM	JAM_HEAVY_TRAFFIC	Jam	Traffic	Heavy	3013
41.97	-87.77	JAM	JAM_HEAVY_TRAFFIC	Jam	Traffic	Heavy	2900
41.93	-87.71	JAM	JAM_HEAVY_TRAFFIC	Jam	Traffic	Heavy	2732
41.91	-87.67	JAM	JAM_HEAVY_TRAFFIC	Jam	Traffic	Heavy	2613

Top 10 locations with the highest counts of Jam - Traffic - Heavy
Number of Alerts
4,500
4,000
3,500
3,000

Figure 5: Image1

Downtown and Lincoln Park



Choose an alert type and subtype:

Road Closed - Event - Unclassified

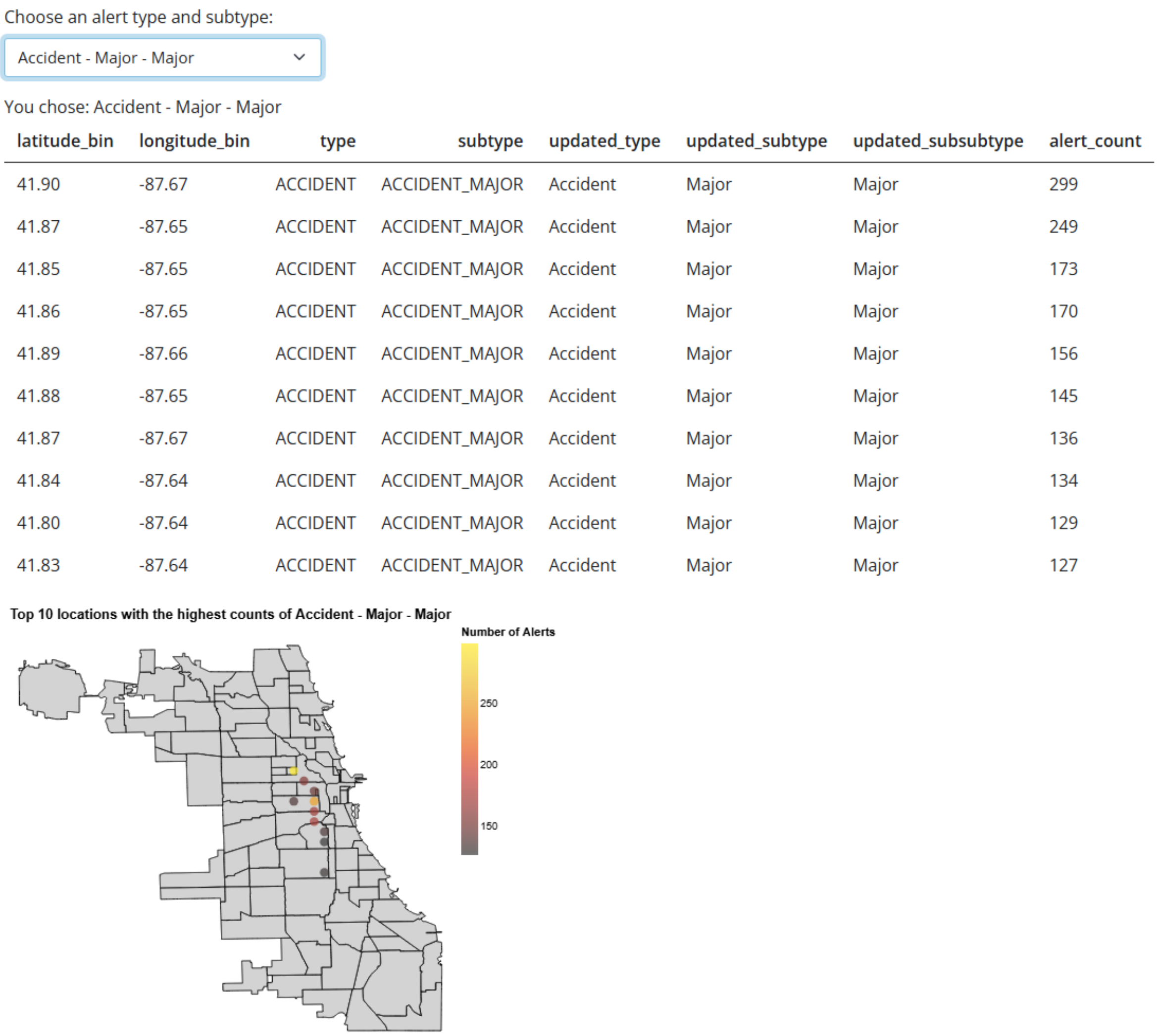
You chose: Road Closed - Event - Unclassified

latitude_bin	longitude_bin	type	subtype	updated_type	updated_subtype	updated_subsubtype	alert_count
41.96	-87.75	ROAD_CLOSED	ROAD_CLOSED_EVENT	Road Closed	Event	Unclassified	14837
41.88	-87.65	ROAD_CLOSED	ROAD_CLOSED_EVENT	Road Closed	Event	Unclassified	9898
41.83	-87.64	ROAD_CLOSED	ROAD_CLOSED_EVENT	Road Closed	Event	Unclassified	5060
41.69	-87.60	ROAD_CLOSED	ROAD_CLOSED_EVENT	Road Closed	Event	Unclassified	4961
41.71	-87.64	ROAD_CLOSED	ROAD_CLOSED_EVENT	Road Closed	Event	Unclassified	4949
41.87	-87.63	ROAD_CLOSED	ROAD_CLOSED_EVENT	Road Closed	Event	Unclassified	356
41.88	-87.68	ROAD_CLOSED	ROAD_CLOSED_EVENT	Road Closed	Event	Unclassified	119
41.88	-87.63	ROAD_CLOSED	ROAD_CLOSED_EVENT	Road Closed	Event	Unclassified	118
41.92	-87.64	ROAD_CLOSED	ROAD_CLOSED_EVENT	Road Closed	Event	Unclassified	97
41.94	-87.65	ROAD_CLOSED	ROAD_CLOSED_EVENT	Road Closed	Event	Unclassified	88

Top 10 locations with the highest counts of Road Closed - Event - Unclassified
Number of Alerts
14,837
88

Figure 6: Image1

Where are Major accidents most common? Highways, mostly I-90



Choose an alert type and subtype:

Accident - Major - Major

You chose: Accident - Major - Major

latitude_bin	longitude_bin	type	subtype	updated_type	updated_subtype	updated_subsubtype	alert_count
41.90	-87.67	ACCIDENT	ACCIDENT_MAJOR	Accident	Major	Major	299
41.87	-87.65	ACCIDENT	ACCIDENT_MAJOR	Accident	Major	Major	249
41.85	-87.65	ACCIDENT	ACCIDENT_MAJOR	Accident	Major	Major	173
41.86	-87.65	ACCIDENT	ACCIDENT_MAJOR	Accident	Major	Major	170
41.89	-87.66	ACCIDENT	ACCIDENT_MAJOR	Accident	Major	Major	156
41.88	-87.65	ACCIDENT	ACCIDENT_MAJOR	Accident	Major	Major	145
41.87	-87.67	ACCIDENT	ACCIDENT_MAJOR	Accident	Major	Major	136
41.84	-87.64	ACCIDENT	ACCIDENT_MAJOR	Accident	Major	Major	134
41.80	-87.64	ACCIDENT	ACCIDENT_MAJOR	Accident	Major	Major	129
41.83	-87.64	ACCIDENT	ACCIDENT_MAJOR	Accident	Major	Major	127

Top 10 locations with the highest counts of Accident - Major - Major

Figure 7: Image1

what are the road types most common for major accidents? Add Road Type column.

App #2: Top Location by Alert Type and Hour Dashboard (20 points)

1.

a. it would not be a good idea to collapse by ts since it is down to seconds, not meaningful to group by ts with this much of precision.

b.

```
merged_data['ts'] = pd.to_datetime(merged_data['ts'])
```

```
merged_data['hour'] = merged_data['ts'].dt.strftime('%H:00')
```

```
chosen_type = "JAM"
chosen_subtype = "JAM_HEAVY_TRAFFIC"
hr_alert_counts = (
    merged_data.groupby(["latitude_bin", "longitude_bin", "type", "subtype",
 ↪  "updated_type", "updated_subtype", "updated_subsubtype", "hour"]).size()
    .reset_index(name="alert_count")
    .sort_values(by="alert_count", ascending=False)
)
```

There are 13669 columns

```
hr_alert_counts_path = './top_alerts_map_byhour/top_alerts_map_byhour.csv'
hr_alert_counts.to_csv(hr_alert_counts_path, index=False)
```

c.

```
hr_alert_counts_by_010203 = hr_alert_counts[(hr_alert_counts['hour'] ==
 ↪  '01:00') | (hr_alert_counts['hour'] == '02:00') |
 ↪  (hr_alert_counts['hour'] == '03:00') & (hr_alert_counts["type"] ==
 ↪  chosen_type) & (hr_alert_counts["subtype"] == chosen_subtype)]
```

```
hr_alert_counts_by_02 = hr_alert_counts[hr_alert_counts['hour'] == '02:00']
```

```
hr_alert_counts_by_03 = hr_alert_counts[hr_alert_counts['hour'] == '03:00']
```

```
points = alt.Chart(hr_alert_counts_by_010203.head(10)).mark_circle().encode(
    longitude=alt.X("longitude_bin:Q"),
    latitude=alt.Y("latitude_bin:Q"),
    tooltip=["latitude_bin", "longitude_bin", "alert_count"],

)
map_layer = (
    alt.Chart(geo_data).mark_geoshape(fill="lightgray", stroke="black")
    .properties(
        width=400,
        height=400
    )
    .project("identity", reflectY=True)  # Ensure correct alignment with coordinates
)

combined_plot = (
    map_layer + points
).properties(title="Top 10")

combined_plot.save('./plots/plot4.png')
combined_plot
```

```
alt.LayerChart(...)
```

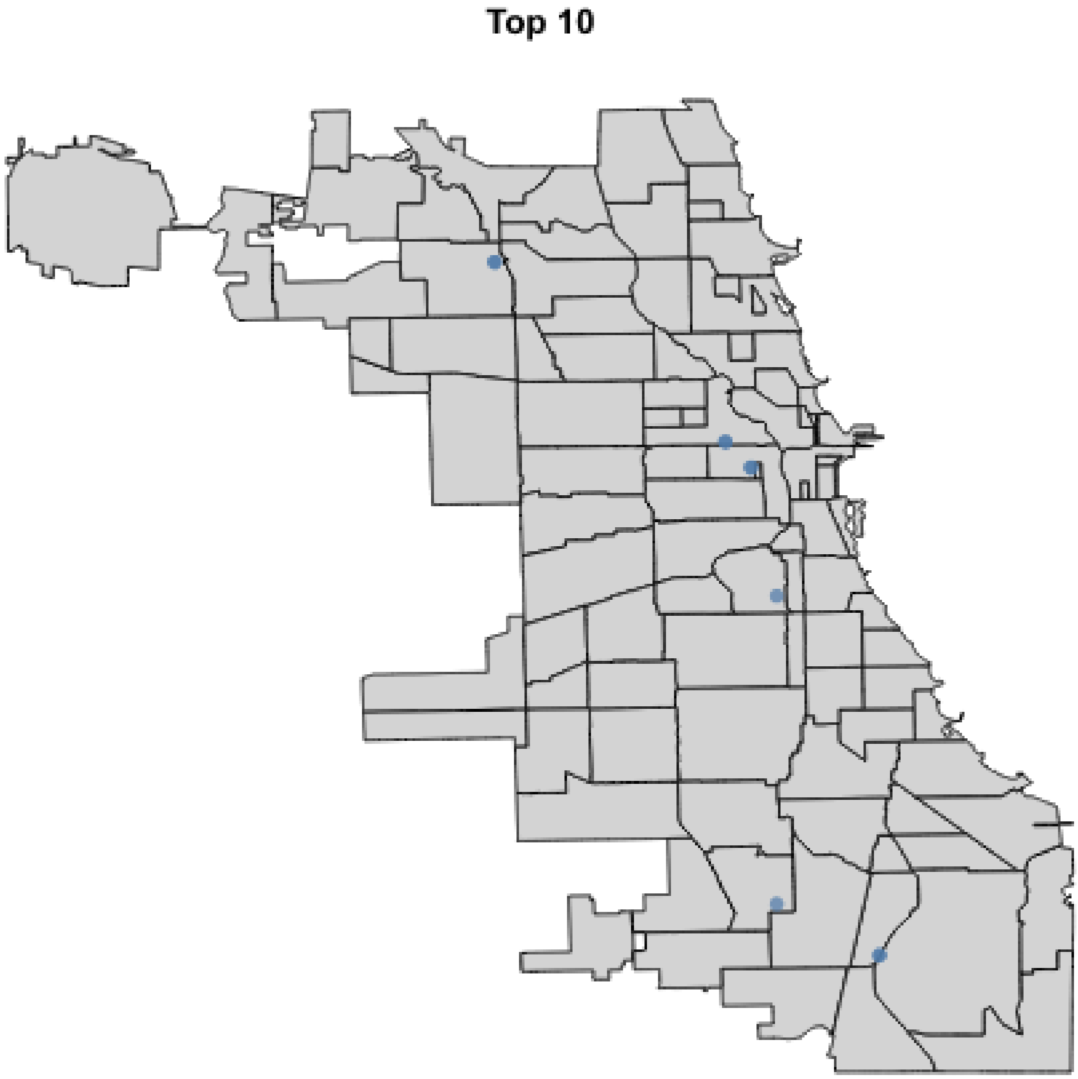
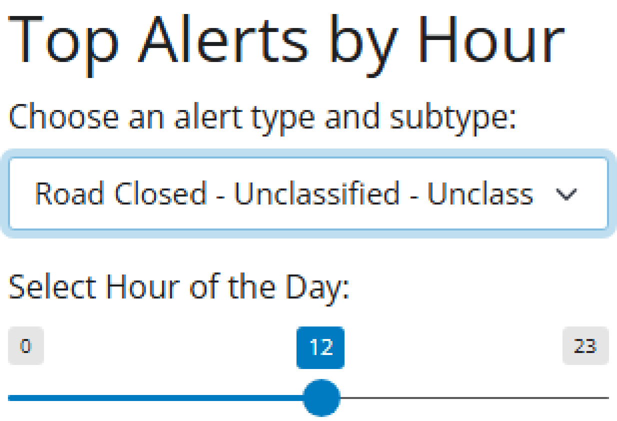


Figure 8: Image1

This time I used the latitide longitude without the xy axis. the points are normal 2. a.



Top Alerts by Hour

Choose an alert type and subtype:

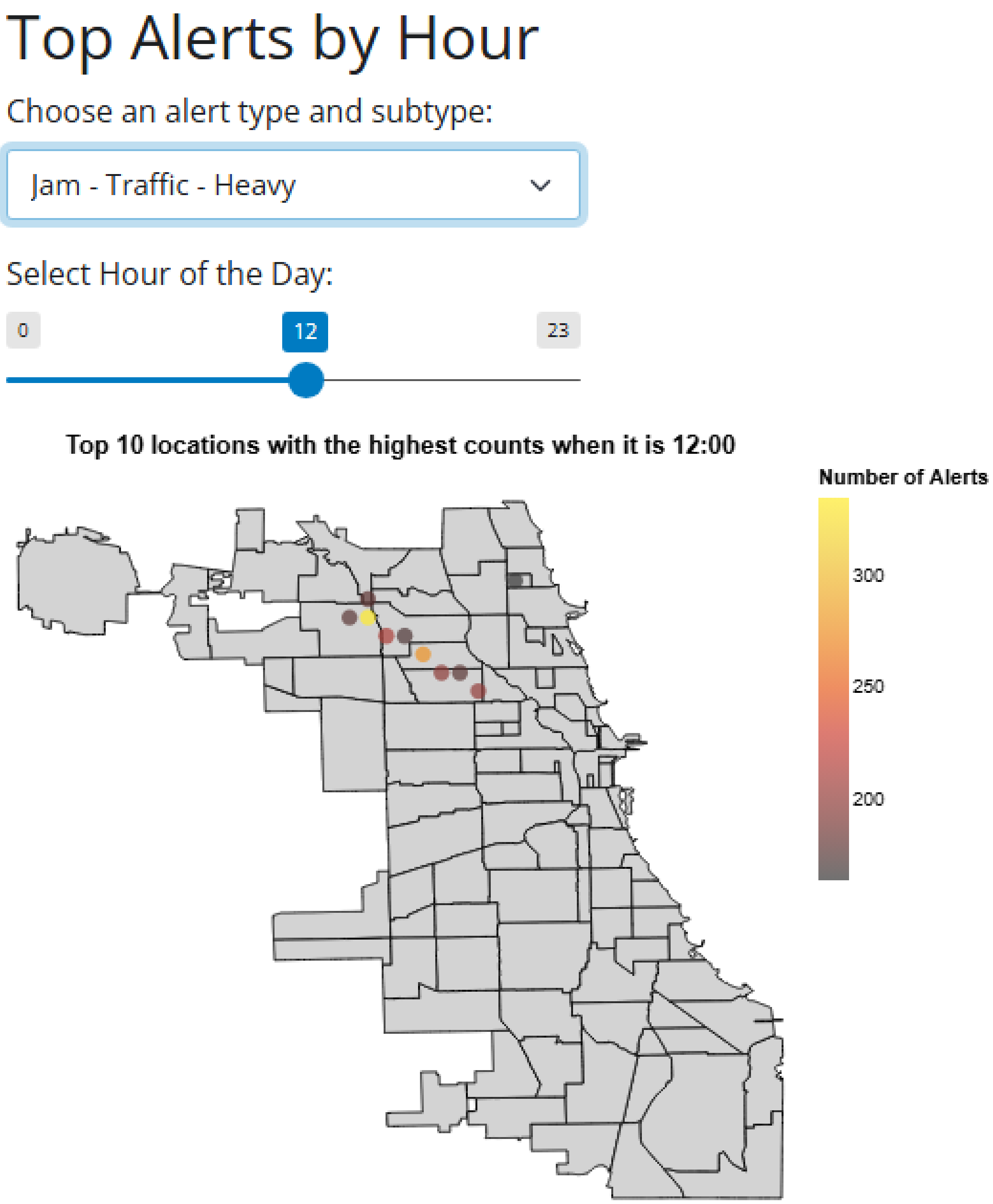
Road Closed - Unclassified - Unclass

Select Hour of the Day:

12

Figure 9: Image1

b.



Top Alerts by Hour

Choose an alert type and subtype:

Jam - Traffic - Heavy

Select Hour of the Day:

12

Figure 10: Image1

c.

Looks like it is done more in the night hours; To better answer this question we need to use

App3.

App #3: Top Location by Alert Type and Hour Dashboard (20 points)

1.

a.

No. There are too many ranges.

b.

```
df_alert_hr_counts = (
    merged_data.groupby(["latitude_bin", "longitude_bin", "type", "subtype",
 ↪  "updated_type", "updated_subtype", "updated_subsubtype", "hour"])
    .size()
    .reset_index(name="alert_count")
    .sort_values(by="alert_count", ascending=False)
)
```

```
filtered_heavy_df = df_alert_hr_counts[(df_alert_hr_counts["type"] ==
 ↪  chosen_type) & (df_alert_hr_counts["subtype"] == chosen_subtype)]
```

```
filtered_heavy_df['hour_numeric'] =
 ↪  filtered_heavy_df['hour'].str.split(":").str[0].astype(int)
```

```
C:\Users\15535\AppData\Local\Temp\ipykernel_34272\988405720.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versu
  filtered_heavy_df['hour_numeric'] =
  filtered_heavy_df['hour'].str.split(":").str[0].astype(int)
```

```
hr_alert_counts_b_69 = filtered_heavy_df[(filtered_heavy_df['hour_numeric']
 ↪  >= 6) & (filtered_heavy_df['hour_numeric'] <= 9)]
hr_alert_counts_b_69.head(10)
```

	latitude_bin	longitude_bin	type	subtype	updated_type	updated_subtyp
50139	41.890000	-87.660000	JAM	JAM_HEAVY_TRAFFIC	Jam	Traffic
46625	41.880000	-87.650000	JAM	JAM_HEAVY_TRAFFIC	Jam	Traffic
81186	41.980000	-87.800000	JAM	JAM_HEAVY_TRAFFIC	Jam	Traffic
46626	41.880000	-87.650000	JAM	JAM_HEAVY_TRAFFIC	Jam	Traffic
4832	41.700000	-87.600000	JAM	JAM_HEAVY_TRAFFIC	Jam	Traffic
65020	41.940000	-87.650000	JAM	JAM_HEAVY_TRAFFIC	Jam	Traffic
752	41.650000	-87.590000	JAM	JAM_HEAVY_TRAFFIC	Jam	Traffic
41433	41.870000	-87.650000	JAM	JAM_HEAVY_TRAFFIC	Jam	Traffic
26864	41.810000	-87.750000	JAM	JAM_HEAVY_TRAFFIC	Jam	Traffic
43887	41.870000	-87.730000	JAM	JAM_HEAVY_TRAFFIC	Jam	Traffic

```python
points = alt.Chart(hr_alert_counts_b_69.head(10)).mark_circle().encode(
    longitude=alt.X("longitude_bin:Q"),
    latitude=alt.Y("latitude_bin:Q"),
    tooltip=["latitude_bin", "longitude_bin", "alert_count"],

)
map_layer = (
    alt.Chart(geo_data).mark_geoshape(fill="lightgray", stroke="black")
    .properties(
        width=400,
        height=400
    )
    .project("identity", reflectY=True)  # Ensure correct alignment with coordinates
)

combined_plot = (
    map_layer + points
).properties(title="Top 10")

combined_plot.save('./plots/plot5.png')
combined_plot
```

```
alt.LayerChart(...)
```

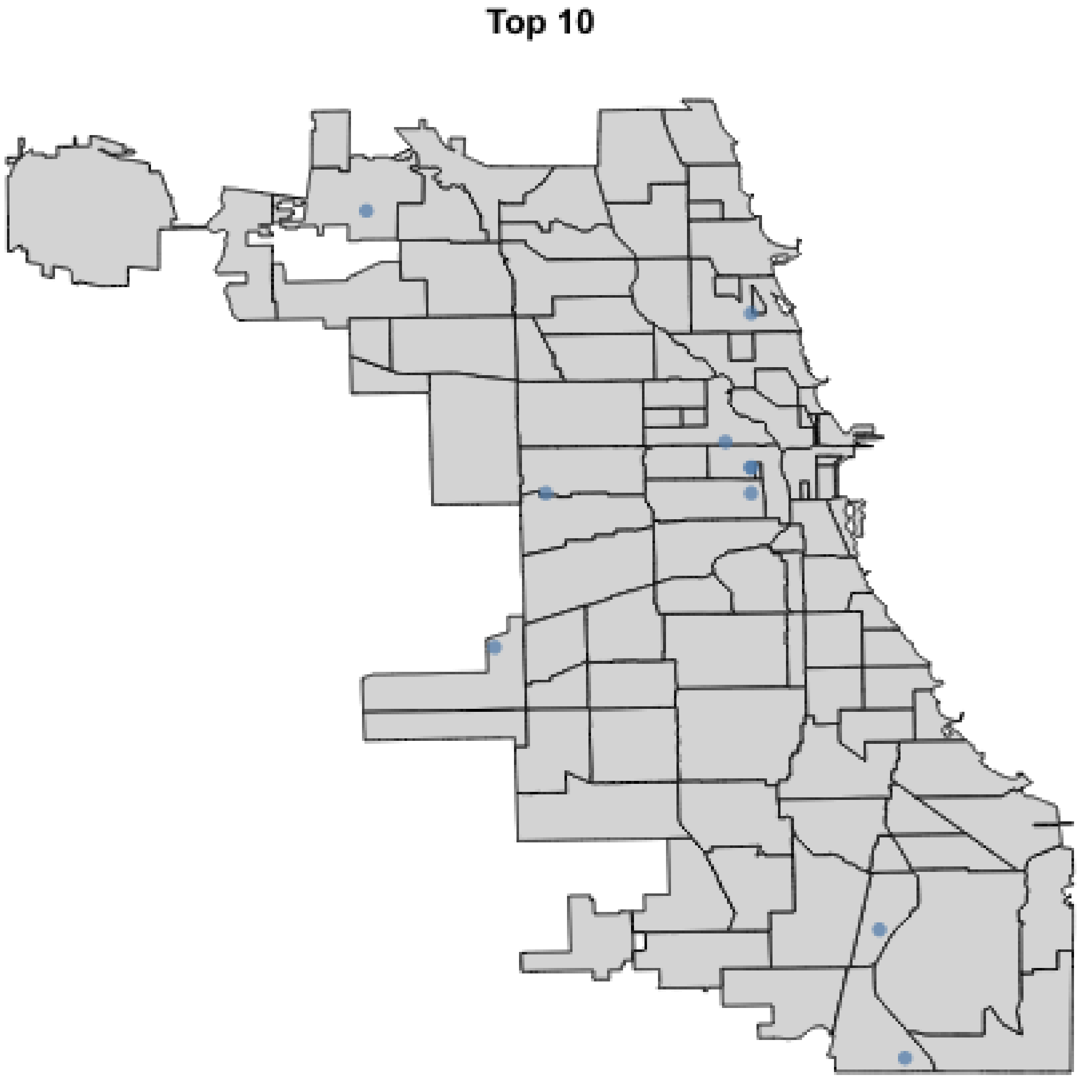
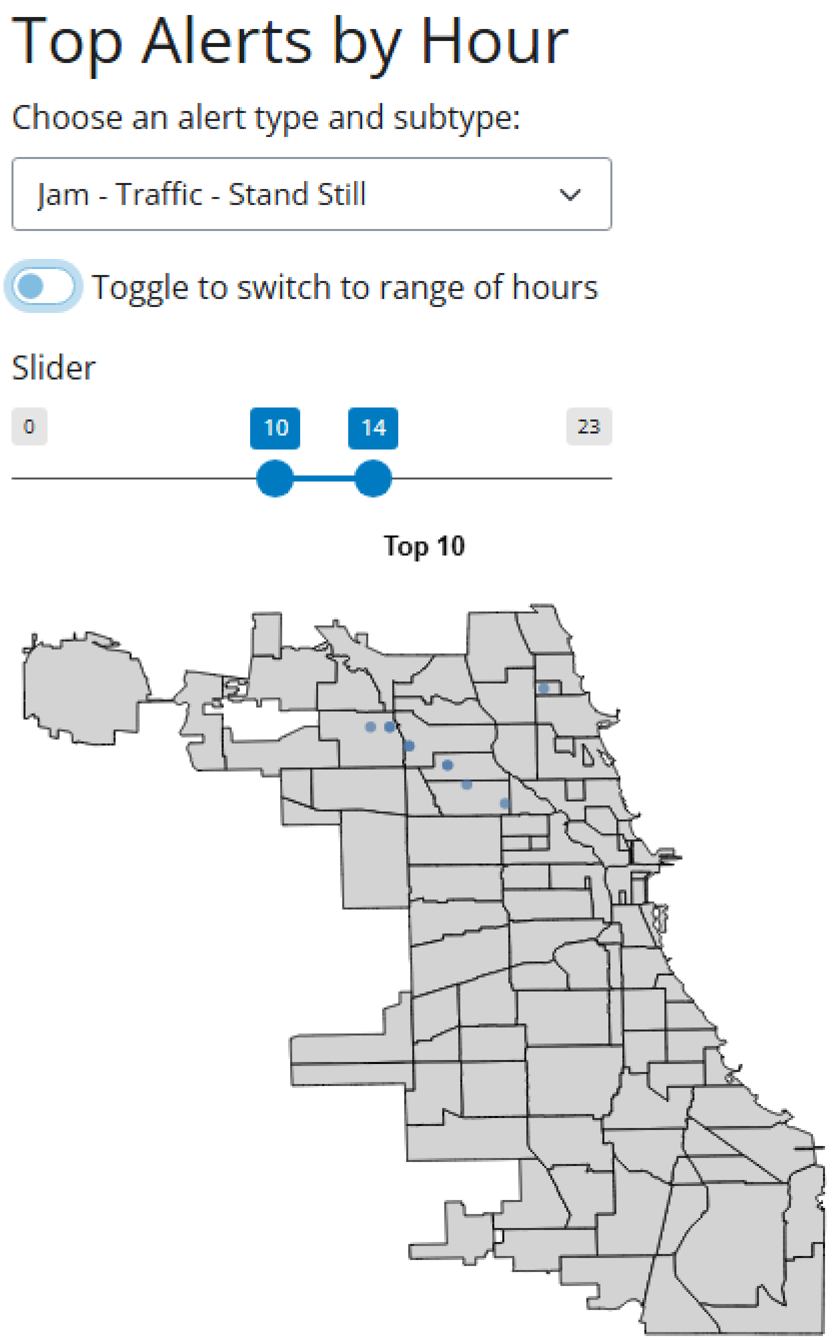


Figure 11: Image1

2.



Top Alerts by Hour

Choose an alert type and subtype:

Jam - Traffic - Stand Still

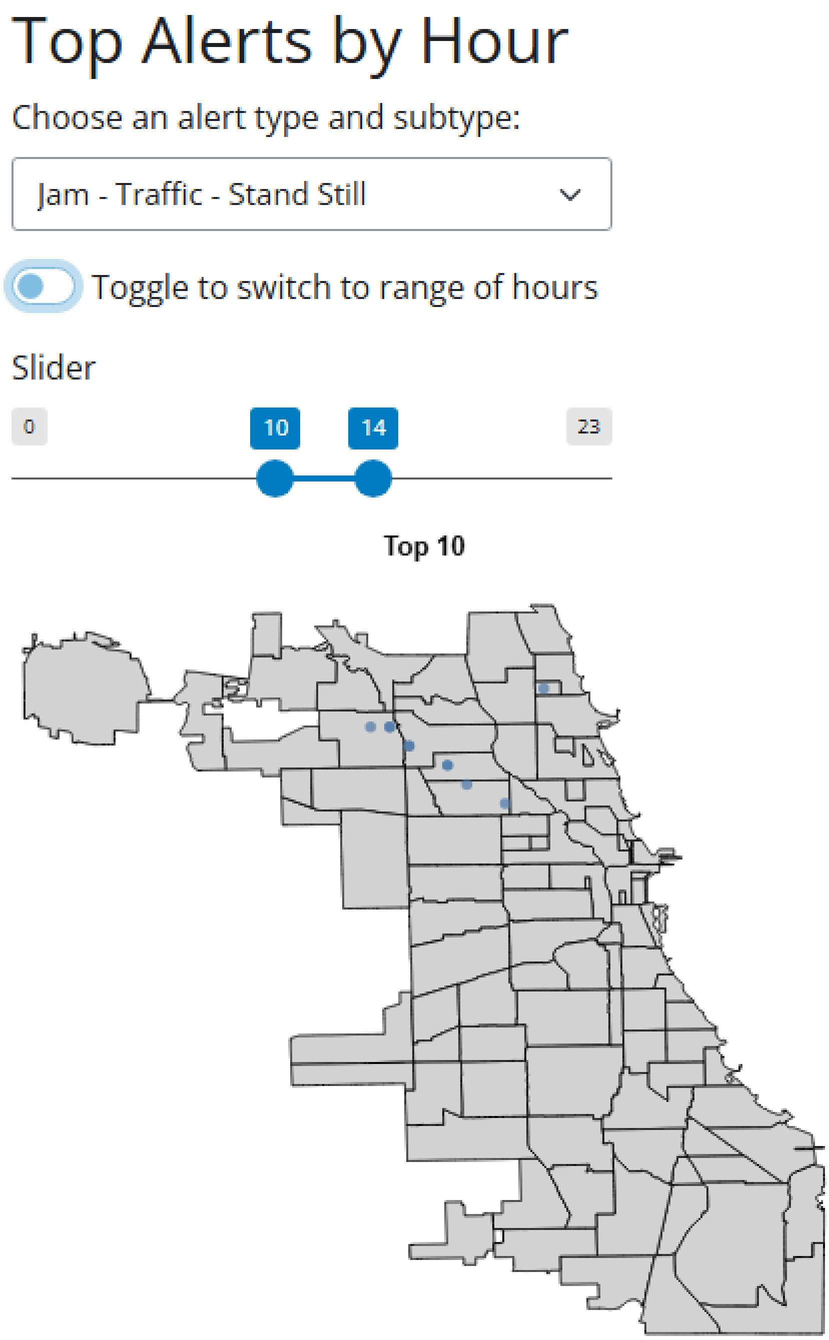
○ Toggle to switch to range of hours

Slider

0 10 14 23

Figure 12: Image1

a.



Top Alerts by Hour

Choose an alert type and subtype:

Jam - Traffic - Stand Still

○ Toggle to switch to range of hours

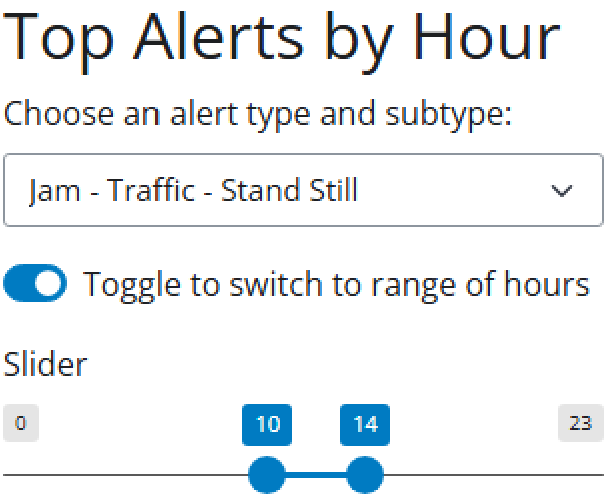
Slider

Top 10

Figure 13: Image1

b.

3.



Top Alerts by Hour

Choose an alert type and subtype:

Jam - Traffic - Stand Still

Toggle to switch to range of hours

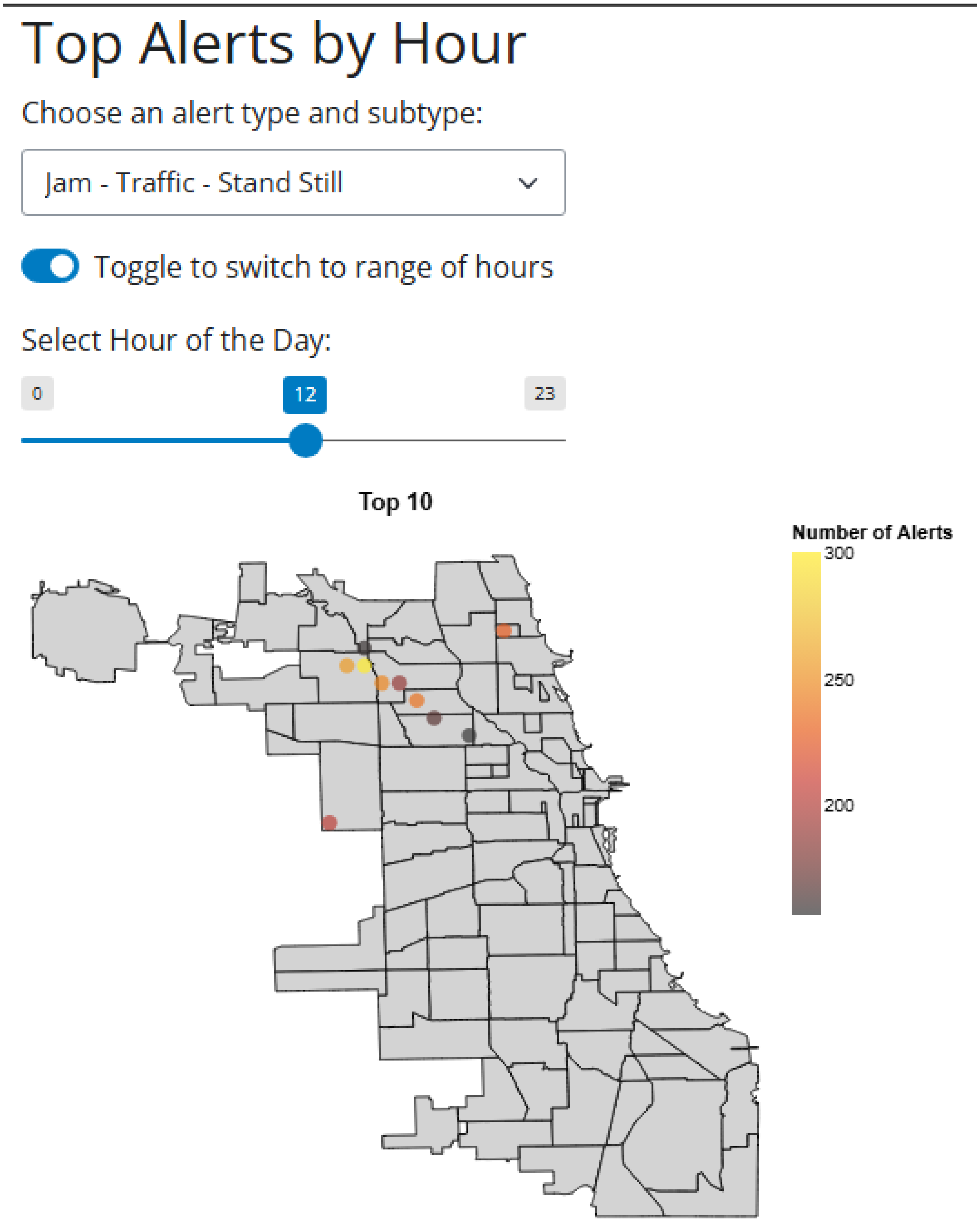
Slider

Figure 14: Image1

a.

True and Flase

b.



Top Alerts by Hour

Choose an alert type and subtype:

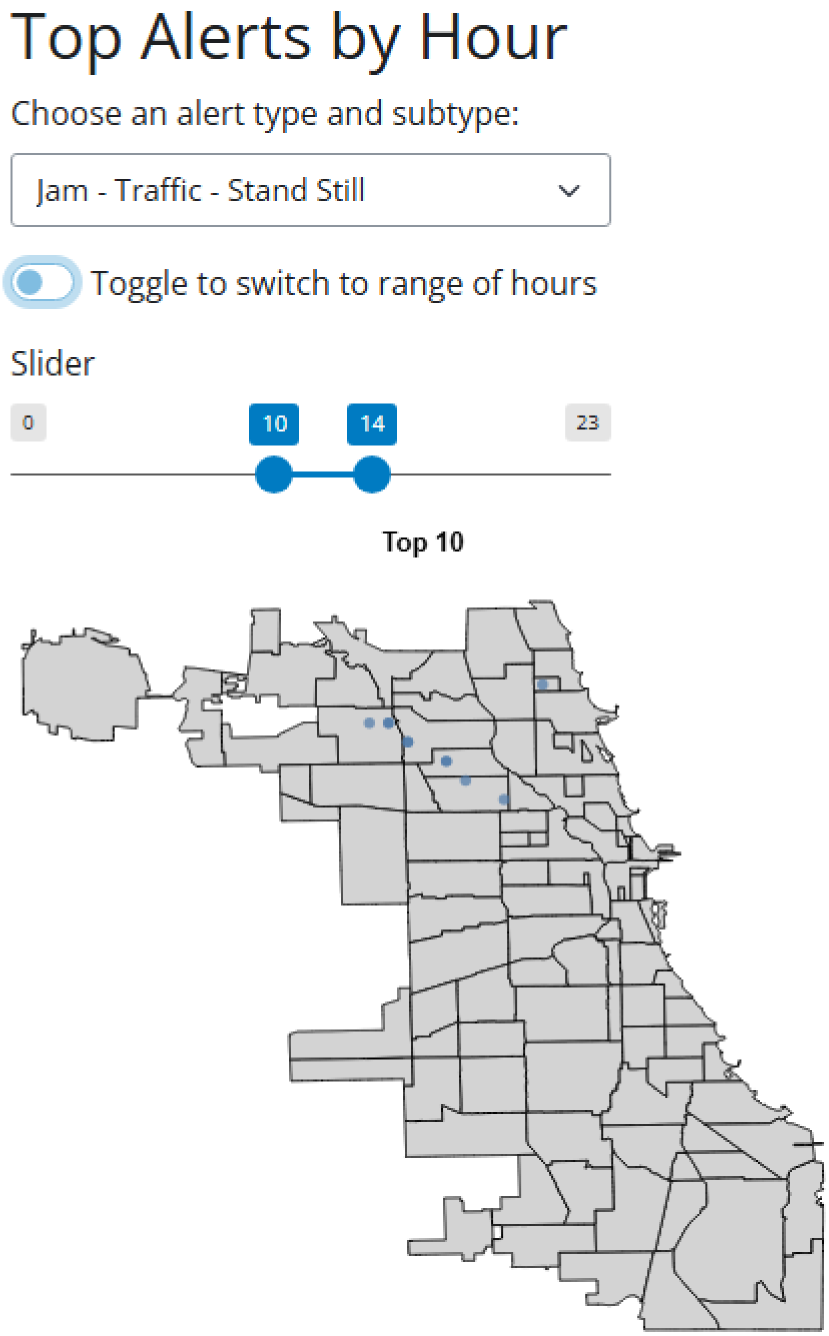
Jam - Traffic - Stand Still

Toggle to switch to range of hours

Select Hour of the Day:

0 12 23

Figure 15: Image1



Top Alerts by Hour

Choose an alert type and subtype:

Jam - Traffic - Stand Still

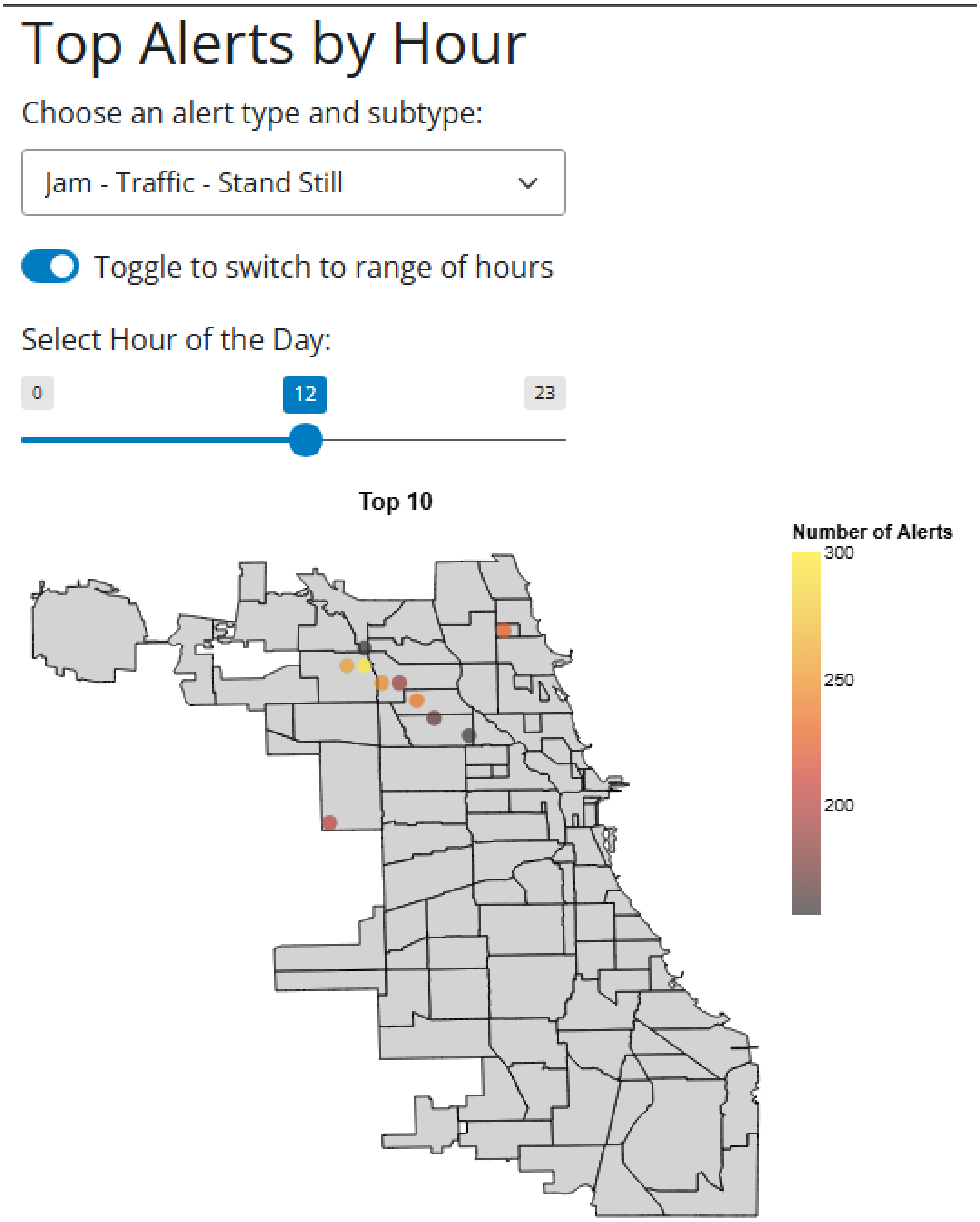
Toggle to switch to range of hours

Slider

Top 10

Figure 16: Image1

c.



Top Alerts by Hour

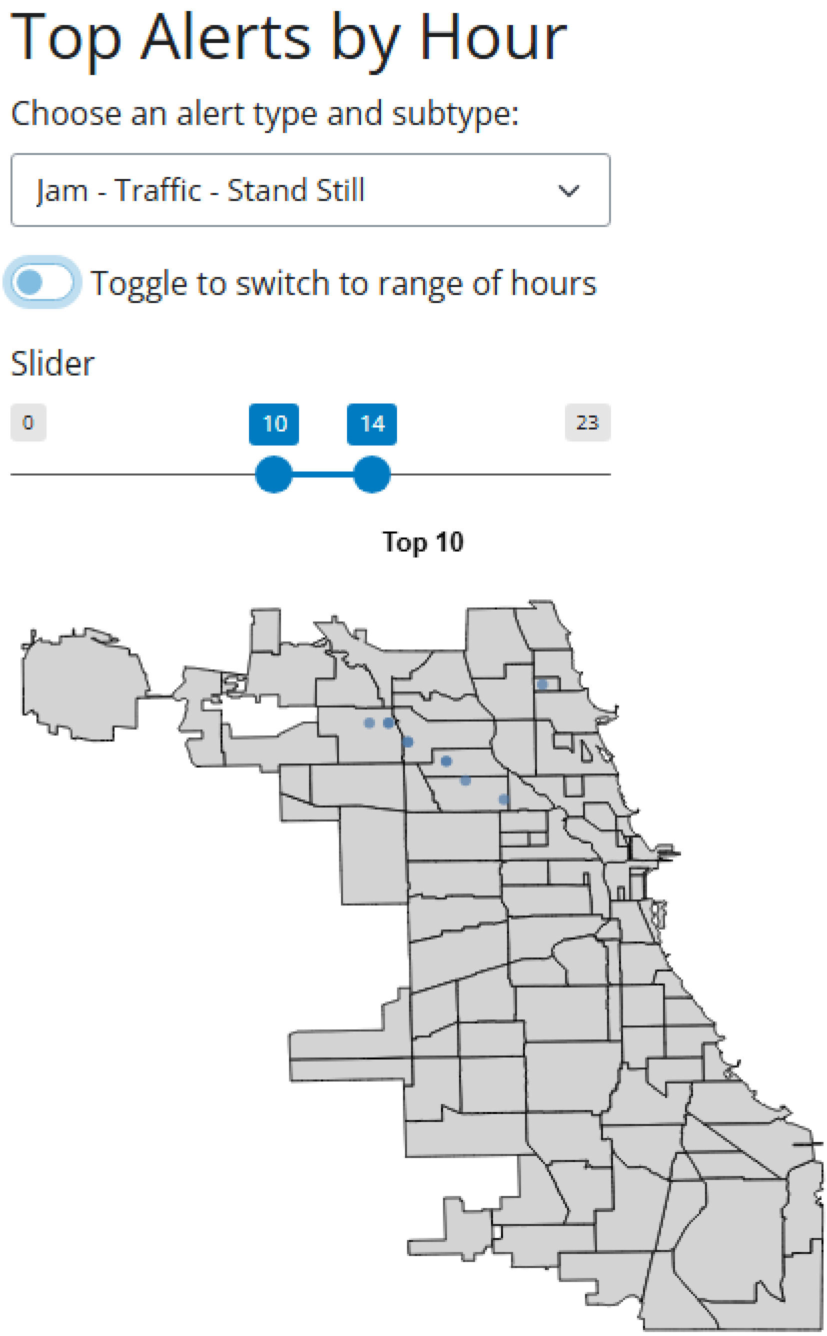
Choose an alert type and subtype:

Jam - Traffic - Stand Still

Toggle to switch to range of hours

Select Hour of the Day: 12

Figure 17: Image1



Top Alerts by Hour

Choose an alert type and subtype:

Jam - Traffic - Stand Still

Toggle to switch to range of hours

Slider

Figure 18: Image1

d. delete the slider, add code to judge if the time of the column is morning and afternoon then display accordingly.