Problem Set 6 - Waze Shiny Dashboard

Zidan Kong

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: **ZK**
2. "I have uploaded the names of anyone I worked with on the problem set **here**" **ZK** (2 point)
3. Late coins used this pset: **ZK** 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. Submit 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 the gradescope repo assignment (5 points).
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!

```
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.

```
# get sample data
import os
base_path = r'/Users/kongzidan/Documents/GitHub/DAP-PS6'
path = os.path.join(base_path, 'waze_data_sample.csv')
df_sample = pd.read_csv(path)
```

```
# drop 'ts', 'geo', 'geoWKT' columns
sample_2 = df_sample.drop(columns=['ts', 'geo', 'geoWKT'])

# define the function to get alt data type from dtype
def altair_types(dataframe):
    altair_type = {}
```

```
    for column in dataframe.columns:
        dtype = dataframe[column].dtype
        if pd.api.types.is_numeric_dtype(dtype):
            altair_type[column] = 'Quantitative'
        elif pd.api.types.is_datetime64_any_dtype(dtype):
            altair_type[column] = 'Temporal'
        elif pd.api.types.is_categorical_dtype(dtype):
            altair_type[column] = 'Ordinal' if dataframe[column].cat.ordered
 ↪  else 'Nominal'
        else:
            altair_type[column] = 'Nominal'

    return pd.DataFrame(list(altair_type.items()), columns=['Variable',
     ↪  'Altair_Type'])

# call function
result_df = altair_types(sample_2)
print(result_df)
```

```
        Variable   Altair_Type
0     Unnamed: 0  Quantitative
1           city       Nominal
2     confidence  Quantitative
3      nThumbsUp  Quantitative
4         street       Nominal
5           uuid       Nominal
6        country       Nominal
7           type       Nominal
8        subtype       Nominal
9       roadType  Quantitative
10   reliability  Quantitative
11        magvar  Quantitative
12  reportRating  Quantitative
```

```
/var/folders/6q/zpwycs2x7_j_fsrh_zrqn4k80000gn/T/ipykernel_89616/2941963284.py:13:
DeprecationWarning: is_categorical_dtype is deprecated and will be removed in
a future version. Use isinstance(dtype, pd.CategoricalDtype) instead
  elif pd.api.types.is_categorical_dtype(dtype):
```

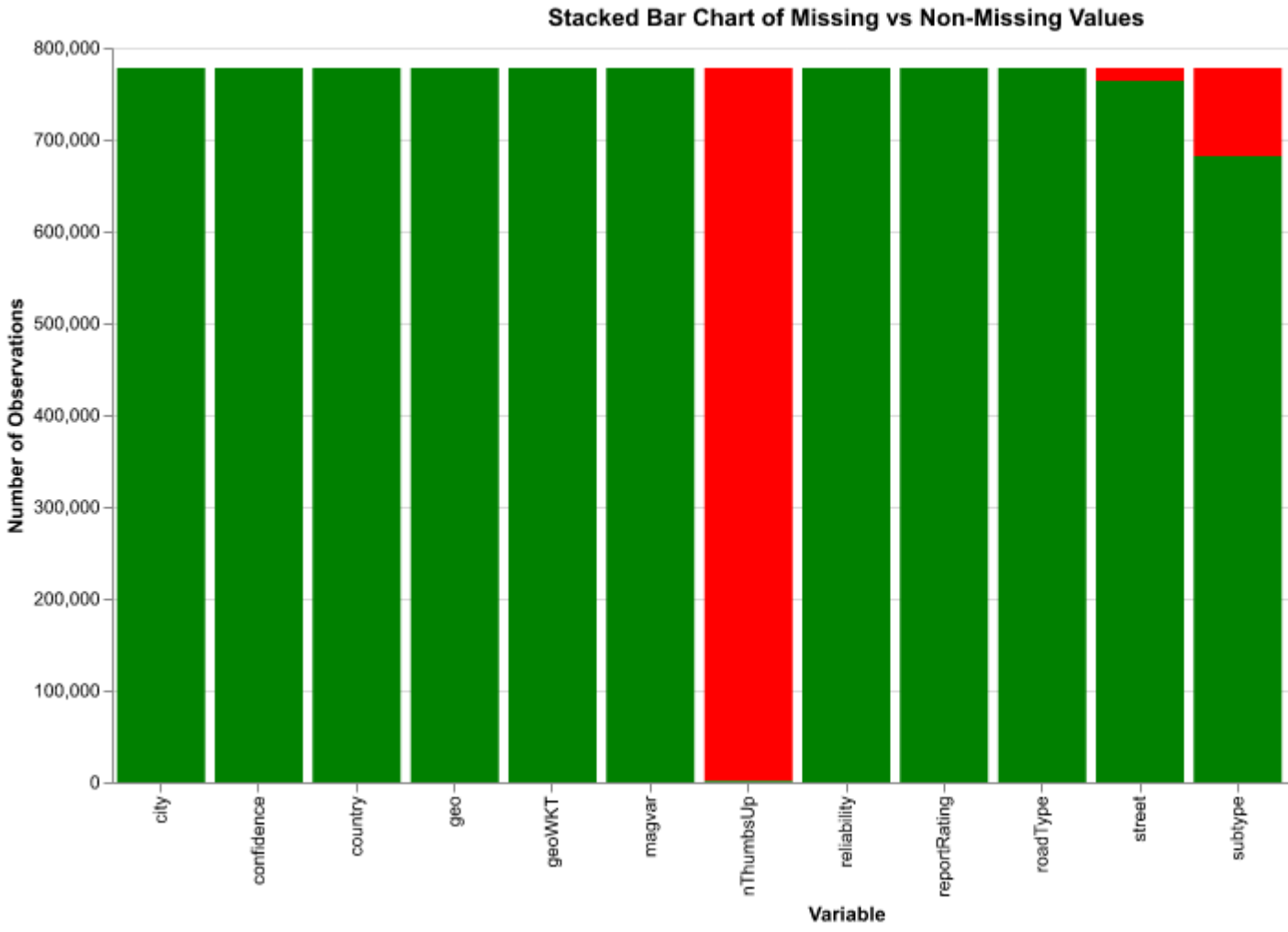
referencing stackoverflow for cheching dtype:https://stackoverflow.com/questions/40353079/pandas-how-to-check-dtype-for-all-columns-in-a-dataframe, https://stackoverflow.com/questions/54426845/,how-to-check-if-a-pandas-dataframe-contains-only-numeric-values-column-wise/67950383#67950383,

https://stackoverflow.com/questions/21030174/how-do-i-test-if-an-object-is-a-pandas-datetime-index/78482582#78482582, https://stackoverflow.com/questions/26924904/check-if-dataframe-column-is-categorical/26925340#26925340 Referencing Chatgpt for mapping dtype to altair datatype, referencing chatgot for orgainzing the for loop dictionary to a dataframe.

2.

```
# get full data
path2 = os.path.join(base_path, 'waze_data.csv')
df = pd.read_csv(path2)

# define function to calculate null and notnull


def count_nulls(dataframe):
    missing_counts = dataframe.isnull().sum()
    non_missing_counts = dataframe.notnull().sum()

    summary_df = pd.DataFrame({
        'Variable': dataframe.columns,
        'Missing': missing_counts.values,
        'Not Missing': non_missing_counts.values
    })
    return summary_df

# call function
count_null = count_nulls(df)

# plot bar chart
chart = alt.Chart(count_null).transform_fold(
    ['Missing', 'Not Missing'],
    as_=['Status', 'Count']
).mark_bar().encode(
    alt.X('Variable:N', title='Variable'),
    alt.Y('Count:Q', title='Number of Observations'),
    alt.Color('Status:N',
              scale=alt.Scale(domain=['Missing', 'Not Missing'],
                              range=['red', 'green'])),).properties(
    title='Stacked Bar Chart of Missing vs Non-Missing Values',
    width=800,
    height=400
)
```

```
chart.show()

# calculating null shares
count_null['null_share'] = count_null['Missing'] / \
    (count_null['Missing']+count_null['Not Missing'])

# sorting to get the highest null share variable
highest_share = count_null.sort_values(by='null_share', ascending=False)
highest_share.head(1)
```



	Variable	Missing	Not Missing	null_share
2	nThumbsUp	776723	1371	0.998238

'nThumbsUp', 'street' and 'subtype' has the null values. 'nThumbsUp' variable has the highest share of missing value.

refencing stackoveflow for drawing stacked bar chart: https://stackoverflow.com/questions/64418356/plot-a-trellis-stacked-bar-chart-in-altair-by-combining-column-values https://stackoverflow.com/questions/74283103/ a-stacked-bar-chart-with-transform-fold-altair referencing chatgpt for error, should not specify column when using .sum() default with axis=0,

3.

```
# find unique type and subtype
unique_types = df['type'].dropna().unique()
unique_subtypes = df['subtype'].dropna().unique()
print(unique_types)
print(unique_subtypes)
# get subtype null df
sub_na = df[df['subtype'].isna()]
# find the type unique in subtype null df
types_na_sub = sub_na['type'].unique()
print(len(types_na_sub))
# create a df that has no null value in subtype
nosubna_df = df.dropna(subset=['subtype'])
# find unique type-subtype pair
no_na_type_groups = nosubna_df.groupby('type')['subtype'].unique()
print(no_na_type_groups)
# clean type and subtype expression
types_subtypes = {}
for type_val, subtypes in no_na_type_groups.items():
    if len(subtypes) >= 2:
        cleaned_subtypes = [subtype
                            .replace("ACCIDENT_", "")
                            .replace("HAZARD_", "")
                            .replace("ROAD_CLOSED_", "")
                            .replace("JAM_", "")
                            .replace("_", " ")
                            .capitalize()
                            for subtype in subtypes
                            ]
        types_subtypes[type_val.capitalize()] = cleaned_subtypes
```

```python
# writing bullet list
for type_val, subtypes in types_subtypes.items():
    print(f"- {type_val}:")
    for subtype in subtypes:
        print(f"  - {subtype}")
```

```
['JAM' 'ACCIDENT' 'ROAD_CLOSED' 'HAZARD']
['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']
4
type
ACCIDENT                          [ACCIDENT_MAJOR, ACCIDENT_MINOR]
HAZARD         [HAZARD_ON_ROAD, HAZARD_ON_ROAD_CAR_STOPPED, H...
JAM            [JAM_HEAVY_TRAFFIC, JAM_MODERATE_TRAFFIC, JAM_...
ROAD_CLOSED    [ROAD_CLOSED_EVENT, ROAD_CLOSED_CONSTRUCTION, ...
Name: subtype, dtype: object
- Accident:
  - Major
  - Minor
- Hazard:
  - On road
  - On road car stopped
  - On road construction
  - On road emergency vehicle
  - On road ice
  - On road object
  - On road pot hole
  - On road traffic light fault
  - On shoulder
  - On shoulder car stopped
  - Weather
  - Weather flood
```

```
  - On road lane closed
  - Weather fog
  - On road road kill
  - On shoulder animals
  - On shoulder missing sign
  - Weather heavy snow
  - Weather hail
- Jam:
  - Heavy traffic
  - Moderate traffic
  - Stand still traffic
  - Light traffic
- Road_closed:
  - Event
  - Construction
  - Hazard
```

There are 4 types of variables that have a subtype taht is NA. The type that have enough information to be considered can have a sub-subtype is 'Hazard'. For example, it can create a sub-subtype under subtype of 'Hazard on road'.

i condsider to keep the NA subtypes. Even though the obeservation has no information on subtypes, but it can still be verified under types. Therefore, the obeservations with NA in subtypes also mean something. i will substitute them to 'Umnclassified'.

```
#fill nul with 'Unclassified'
df['subtype'] = df['subtype'].fillna('Unclassified')
```

referencing chatgpt to understanding writing a bullet list.

4.
5.

```
#drop duplicates for type-subtype,get the type and subtype column
crosswalk = df[['type', 'subtype']].drop_duplicates().reset_index(drop=True)
#create three columns
crosswalk['updated_type'] = 'TBD'
crosswalk['updated_subtype'] = 'TBD'
crosswalk['updated_subsubtype'] = 'TBD'
```

2.

```
# based on type and subtype, fillings the remaing three columns
for index, row in crosswalk.iterrows():
    type_value = row['type']
    subtype_value = row['subtype']
    # format expression
    crosswalk.at[index, 'updated_type'] = type_value.capitalize()

    # filling null with 'Unclassified'
    if pd.isna(subtype_value):
        crosswalk.at[index, 'updated_subtype'] = 'Unclassified'
        crosswalk.at[index, 'updated_subsubtype'] = 'Unclassified'
    else:
        # if not null, find what subtype it belongs then format expression
        cleaned_subtype = subtype_value.replace("ACCIDENT_",
↪   "").replace("HAZARD_", "").replace(
            "ROAD_CLOSED_", "").replace("JAM_", "").replace("_", "
            ↪  ").capitalize()
        crosswalk.at[index, 'updated_subtype'] = cleaned_subtype
        # creating subsubtype
        if "HAZARD_ON_ROAD_" in subtype_value:
            subsubtype = subtype_value.replace(
                "HAZARD_ON_ROAD_", "").replace("_", " ").capitalize()
            crosswalk.at[index, 'updated_subsubtype'] = subsubtype
        elif "HAZARD_WEATHER_" in subtype_value:
            subsubtype = subtype_value.replace(
                "HAZARD_WEATHER_", "").replace("_", " ").capitalize()
            crosswalk.at[index, 'updated_subsubtype'] = subsubtype
        else:
            crosswalk.at[index, 'updated_subsubtype'] = 'Unclassified'
```

referencing chatgpt to modify my for loop iterating over rows,e.g at[index,]. 3.

```
# merge data
merged_data = df.merge(crosswalk, on=['type', 'subtype'], how='left')
# get data of Accident-Unclassified
accident_unclassified = merged_data[
    (merged_data['updated_type'] == 'Accident') &
    (merged_data['updated_subtype'] == 'Unclassified')]
print(len(accident_unclassified))
```

24359

There are 24359 rows are there for Accident - Unclassified. 4.

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

1.

a.

```
# extract longitutde and latitude, creating columns
merged_data['longitude'] = merged_data['geo'].str.extract(
    r'POINT\((-?\d+\.\d+)')
merged_data['latitude'] = merged_data['geo'].str.extract(
    r'POINT\(-?\d+\.\d+\s(-?\d+\.\d+)\)')
merged_data['longitude'] = merged_data['longitude'].astype(float)
merged_data['latitude'] = merged_data['latitude'].astype(float)
```

referencing chatgpt, asking how to get latitude and longitute from the geo coordinates.

b.

```
# identify steo
step = 0.01

# create bins
merged_data['latBin'] = (merged_data['latitude'] // step) * step
merged_data['lonBin'] = (merged_data['longitude'] // step) * step
# group by bins
bin_groups = merged_data.groupby(
    ['latBin', 'lonBin']).size().reset_index(name='count')
# find the longitude-latitude bin that have greatest number of obeservations
highest_combine = bin_groups.sort_values(by='count', ascending=False)
highest_combine.head(1)
```

	latBin	lonBin	count
589	41.96	-87.75	26537

The binned combination of (41.96,-87.75) has the greatest number of obeservations.

referencing stackoverflow for binning,https://stackoverflow.com/questions/39254704/pandas-group-bins-of-data-per-longitude-latitude, searching bin latitude c.

```
# groupby based on 'latBin', 'lonBin', 'updated_type', 'updated_subtype'
grouped_type_bin = merged_data.groupby(
    ['latBin', 'lonBin', 'updated_type',
 ↪  'updated_subtype']).size().reset_index(name='counts')

# sort by data
sorted_type_bin = grouped_type_bin.sort_values(by='counts', ascending=False)
# get data for chosen Accdient-Major
acc_major_bin = sorted_type_bin[(sorted_type_bin['updated_type'] ==
 ↪  'Accident') & (
    sorted_type_bin['updated_subtype'] == 'Major')]

# get top 10 count bins for chosen Accdient-Major
top_alerts_map = acc_major_bin.head(10)
# save top 10 of Accident-Major to folder
top_alerts_map.to_csv(

     ↪  '/Users/kongzidan/Documents/GitHub/DAP-PS6/top_alerts_map/top_alerts_map.csv')
```

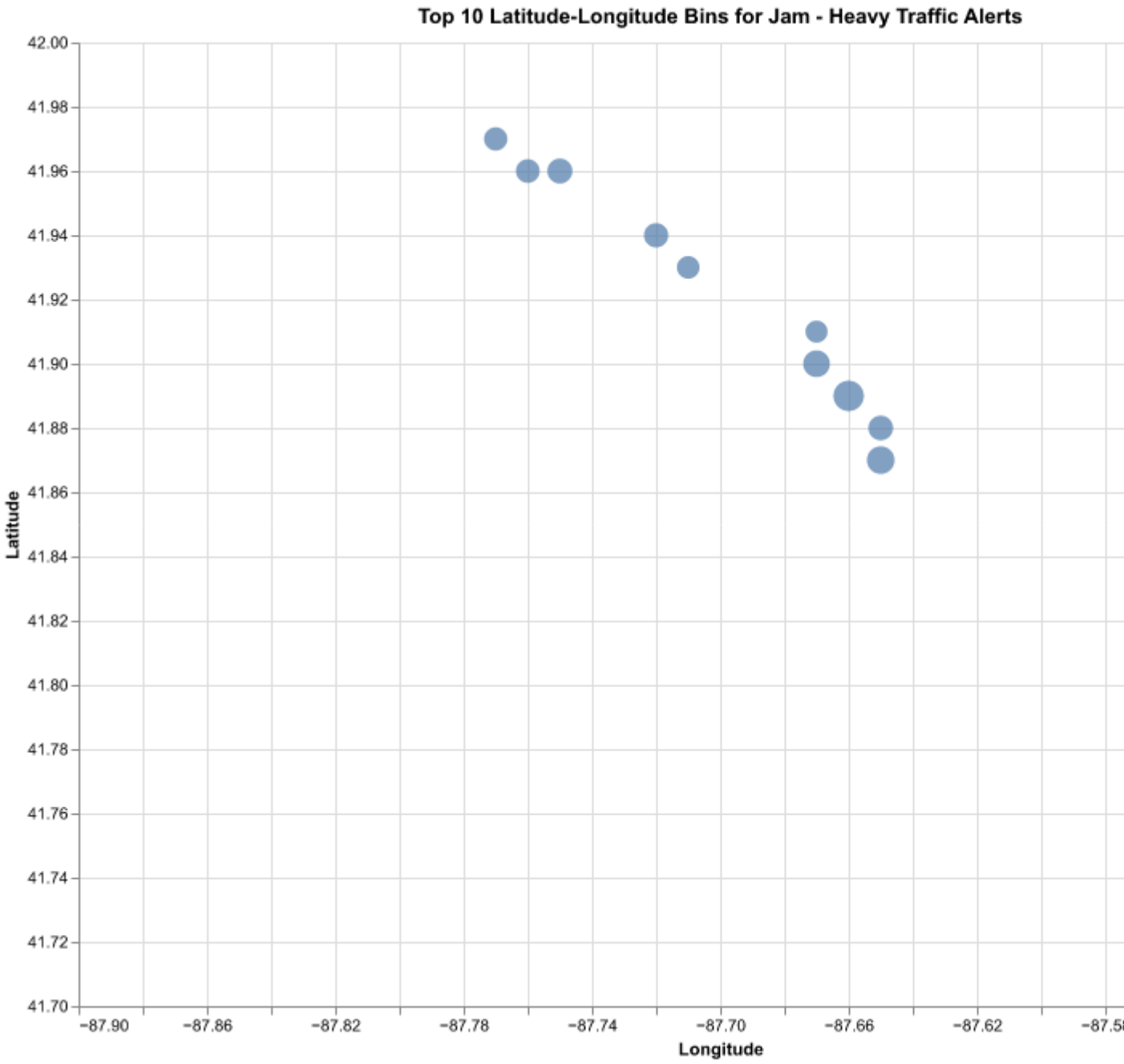
For the level of aggregation, this question is group by 4 variables of 'latBin', 'lonBin', 'updated_type' and 'updated_subtype'. The dataframe has 10 rows since i choose the top 10 latitude-longitude bins for Accident-Major.

referencing stackoverflow for saving csv to file,https://stackoverflow.com/questions/67196334/save-the-file-in-a-different-folder-using-python, search save df to folder

2.

```
# get jam_heavy counts by bins
jam_heavy_bin = sorted_type_bin[(sorted_type_bin['updated_type'] == 'Jam') &
 ↪  (
    sorted_type_bin['updated_subtype'] == 'Heavy traffic')]
# get the top 10
jam_heavy_10 = jam_heavy_bin.head(10)
# plot scatter plot
chart_jam = alt.Chart(jam_heavy_10).mark_circle().encode(
    alt.X('lonBin:Q', title='Longitude',
          scale=alt.Scale(domain=[-87.9, -87.5])),
    alt.Y('latBin:Q', title='Latitude', scale=alt.Scale(domain=[41.7,
 ↪  42.0])),
    size=alt.Size('counts:Q', title='Number of Alerts')
).properties(
```

```
    title='Top 10 Latitude-Longitude Bins for Jam - Heavy Traffic Alerts',
    width=800,
    height=600
)
#show plot
chart_jam.show()
```



3.

a.

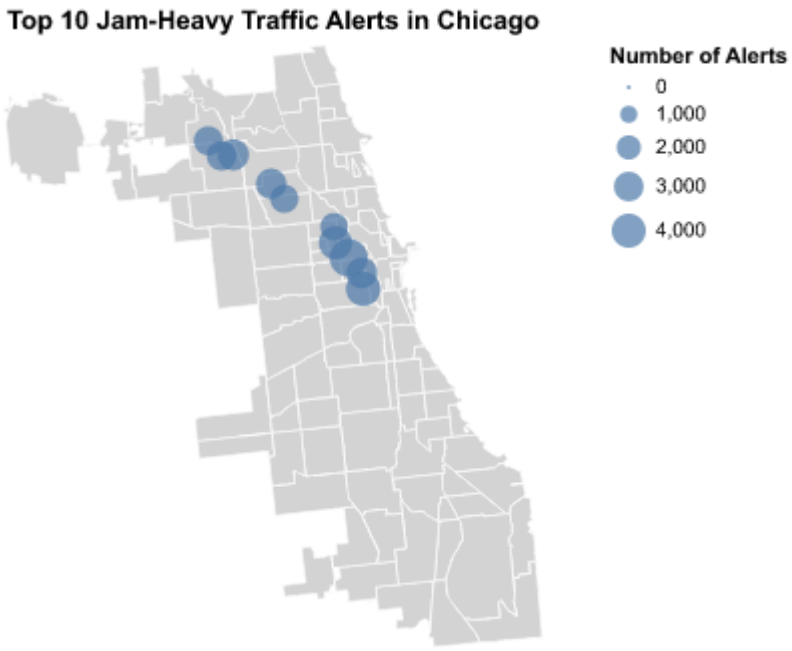
download directly from the link

b.

```
file_path =
 ↪  "/Users/kongzidan/Documents/GitHub/DAP-PS6/top_alerts_map/Boundaries -
 ↪  Neighborhoods.geojson"
#----

with open(file_path) as f:
    chicago_geojson = json.load(f)
#get geo_data
geo_data = alt.Data(values=chicago_geojson["features"])
```

4.

```
# draw background
background = alt.Chart(geo_data).mark_geoshape(
    fill='lightgray',
    stroke='white'
).project('albersUsa')
# draw points layer
points = alt.Chart(jam_heavy_10).mark_circle().encode(
    longitude='lonBin:Q',
    latitude='latBin:Q',
    size=alt.Size('counts:Q', title='Number of Alerts'),
).properties(
    title='Top 10 Jam-Heavy Traffic Alerts in Chicago'
)
# show plot
background+points
```



referencing documentation, https://altair-viz.github.io/altair-tutorial/notebooks/09-Geographic-plots.html.

5.

a.

```
# get the selected list
dropdown_menu = merged_data[['updated_type', 'updated_subtype']].copy()
dropdown_menu["type_subtype"] = dropdown_menu["updated_type"] + \
    " - " + dropdown_menu["updated_subtype"]
dropdown_list = dropdown_menu["type_subtype"].unique()
# don't need unclassified subtype
dropdown_list = [item for item in dropdown_list if "Unclassified" not in
 ↪  item]
print(dropdown_list)
len(dropdown_list)
```

```
['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']
```

```
28
```

There are 28 type x subtype combinations in my dropdown menu.

referencing chatgpt for adding "-" between type and subtype.

```
#save dataset
sorted_type_bin.to_csv('/Users/kongzidan/Documents/GitHub/DAP-PS6/sorted_type_bin.csv')
```

```
![Dropdown
 ↪  Screenshot](/Users/kongzidan/Documents/GitHub/DAP-PS6/dropdown-screeshot.png)
```

```
/bin/bash: -c: line 0: syntax error near unexpected token `('
/bin/bash: -c: line 0: `[Dropdown
Screenshot](/Users/kongzidan/Documents/GitHub/DAP-PS6/dropdown-screeshot.png)'
```

b.

```
![Jam-Heavy traffic
 ↪  Screenshot](/Users/kongzidan/Documents/GitHub/DAP-PS6/Jam-Heavy-traffic-dashboard.png)
```

```
/bin/bash: -c: line 0: syntax error near unexpected token `('
/bin/bash: -c: line 0: `[Jam-Heavy traffic
Screenshot](/Users/kongzidan/Documents/GitHub/DAP-PS6/Jam-Heavy-traffic-dashboard.png)'
```

referencing chatgpt for understanding shiny does not support alt, try to make alt plot saved to image then reload image, not working. Eventually used plt, referencing chatgpt for 'Data' object has no attribute 'plot'. Read geo_data with gpd. referencing chatgpt for scaling, i found out the matplotlib cannot adjust circle size based on counts level, circles are super big for some type-subtypes some are very small, adjusting by scaled trasformation.

c.

```
![Road Closure - Event
↪  Screenshot](/Users/kongzidan/Documents/GitHub/DAP-PS6/Road-Closure-Event.png)
```

```
/bin/bash: -c: line 0: syntax error near unexpected token `('
/bin/bash: -c: line 0: `[Road Closure - Event
Screenshot](/Users/kongzidan/Documents/GitHub/DAP-PS6/Road-Closure-Event.png)'
```

The area with latitude between 41.95~42 and longitude between -87.8~-87.7 has the alerts for road closures due to events most common.

d. question: Comapre the types of Accident-Major and Accident-Minor, do they have similar patterns?

```
![Accident-Major
↪  Screenshot](/Users/kongzidan/Documents/GitHub/DAP-PS6/Accident-Major.png)

![Accident-Minor
↪  Screenshot](/Users/kongzidan/Documents/GitHub/DAP-PS6/Accident-Minor.png)
```

```
/bin/bash: -c: line 0: syntax error near unexpected token `('
/bin/bash: -c: line 0: `[Accident-Major
Screenshot](/Users/kongzidan/Documents/GitHub/DAP-PS6/Accident-Major.png)'
/bin/bash: -c: line 0: syntax error near unexpected token `('
/bin/bash: -c: line 0: `[Accident-Minor
Screenshot](/Users/kongzidan/Documents/GitHub/DAP-PS6/Accident-Minor.png)'
```

Yes, majority of the top 10 counts areas of both Accident-Major and Accident-Minor types are gathered around the area with latitude between 41.95~41.80 and longitude between -87.7~-87.6.

e.

I would suggest to add a time column, which can help review teh information that in what period of time do the alerts for certain type-subtype happen more frequently.

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

1.

a.
b.
c.

2.

a.
b.
c.

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

1.

a.
b.

2.

a.
b.

3.

a.
b.
c.
d.