\usepackage{fvextra} \DefineVerbatimEnvironment{Highlighting}{Verbatim}{breaklines,commandchars=\\\{\}}

\RecustomVerbatimEnvironment{verbatim}{Verbatim}{ showspaces = false, showtabs = false, breaksymbolleft={}, breaklines }

Problem Set 6 - Waze Shiny Dashboard

AUTHOR
Peter Ganong, Maggie Shi, and Andre Oviedo

PUBLISHED
November 24, 2024

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!

```
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.

```
import zipfile

zip_path = "/Users/tsaili-ting/Uchicago/Year2/Y2Fall/Python2/ps6/waze_data.zip"
out_path = "/Users/tsaili-ting/Uchicago/Year2/Y2Fall/Python2/ps6"

# read the file
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(out_path)

# load csv file
df_s = pd.read_csv("/Users/tsaili-ting/Uchicago/Year2/Y2Fall/Python2/ps6/waze_data_sample
df_s.head(5)
```

	Unnamed: 0	city	confidence	nThumbsUp	street	uuid	country	type	subtype
0	584358	Chicago, IL	0	NaN	NaN	c9b88a12-79e8-44cb-aadd-a75855fc4bcb	US	JAM	NaN
1	472915	Chicago, IL	0	NaN	I-90 E	7c634c0a-099c-4262-b57f-e893bdebce73	US	ROAD_CLOSED	ROAD_CLOS

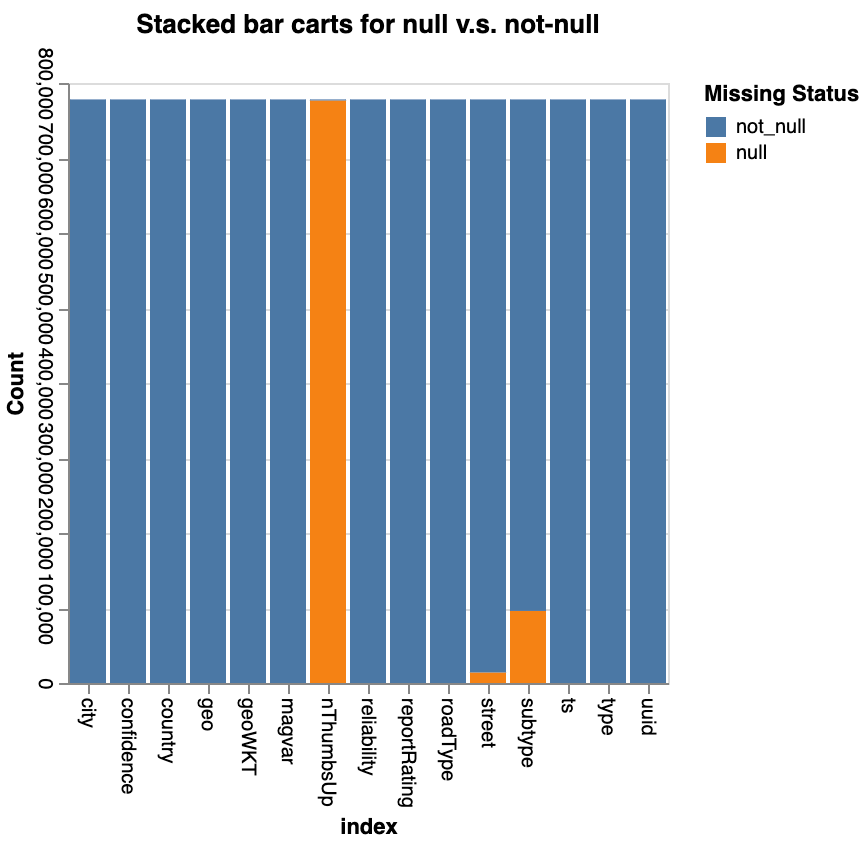
	Unnamed: 0	city	confidence	nThumbsUp	street	uuid	country	type	subtype
2	550891	Chicago, IL	0	NaN	I-90 W	7aa3c61a-f8dc-4fe8-bbb0-db6b9e0dc53b	US	HAZARD	HAZARD_ON
3	770659	Chicago, IL	0	NaN	NaN	3b95dd2f-647c-46de-b4e1-8ebc73aa9221	US	HAZARD	HAZARD_ON
4	381054	Chicago, IL	0	NaN	N Pulaski Rd	13a5e230-a28a-4bf4-b928-bc1dd38850e0	US	JAM	JAM_HEAVY

city: Nominal confidence: Ordinal nThumbsUp: Quantitative street: Nominal uuid: Nominal country: Nominal type: Nominal subtype: Nominal roadType: Nominal reliability: Quantitative magvar: Nominal reportRating: Ordinal

2.

```
df = pd.read_csv("/Users/tsaili-ting/Uchicago/Year2/Y2Fall/Python2/ps6/waze_data.csv")

null = df.isnull().sum()
not_null = df.notnull().sum()

null_df = pd.DataFrame({
    'null': null,
    'not_null': not_null
}).reset_index()
```

```
long_df = null_df.melt(id_vars='index', value_vars=['null', 'not_null'], var_name='Missin

# Create a stacked bar chart using Altair
bar = alt.Chart(long_df).mark_bar().encode(
    x='index:N',
    y='Count:Q',
    color='Missing Status:N',
    tooltip=['index:N', 'Missing Status:N', 'Count:Q']
).properties(
    title="Stacked bar carts for null v.s. not-null"
).configure_axis(
    labelAngle=90
)
bar
```



nThumbs has the most na value and higheset share of na.

3.

```
df['subtype'] = df['subtype'].fillna('NA')

grouped_df = df.groupby(["type","subtype"]).agg(count = ("subtype", "count")).reset_index
grouped_df
```

	type	subtype	count
0	ACCIDENT	ACCIDENT_MAJOR	6669
1	ACCIDENT	ACCIDENT_MINOR	2509
2	ACCIDENT	NA	24359
3	HAZARD	HAZARD_ON_ROAD	34069
4	HAZARD	HAZARD_ON_ROAD_CAR_STOPPED	5482
5	HAZARD	HAZARD_ON_ROAD_CONSTRUCTION	32094
6	HAZARD	HAZARD_ON_ROAD_EMERGENCY_VEHICLE	8360
7	HAZARD	HAZARD_ON_ROAD_ICE	234
8	HAZARD	HAZARD_ON_ROAD_LANE_CLOSED	541
9	HAZARD	HAZARD_ON_ROAD_OBJECT	16050
10	HAZARD	HAZARD_ON_ROAD_POT_HOLE	28268
11	HAZARD	HAZARD_ON_ROAD_ROAD_KILL	65

	type	subtype	count
12	HAZARD	HAZARD_ON_ROAD_TRAFFIC_LIGHT_FAULT	4874
13	HAZARD	HAZARD_ON_SHOULDER	40
14	HAZARD	HAZARD_ON_SHOULDER_ANIMALS	115
15	HAZARD	HAZARD_ON_SHOULDER_CAR_STOPPED	176751
16	HAZARD	HAZARD_ON_SHOULDER_MISSING_SIGN	76
17	HAZARD	HAZARD_WEATHER	2146
18	HAZARD	HAZARD_WEATHER_FLOOD	2844
19	HAZARD	HAZARD_WEATHER_FOG	697
20	HAZARD	HAZARD_WEATHER_HAIL	7
21	HAZARD	HAZARD_WEATHER_HEAVY_SNOW	138
22	HAZARD	NA	3212
23	JAM	JAM_HEAVY_TRAFFIC	170442
24	JAM	JAM_LIGHT_TRAFFIC	5
25	JAM	JAM_MODERATE_TRAFFIC	4617
26	JAM	JAM_STAND_STILL_TRAFFIC	142380
27	JAM	NA	55041
28	ROAD_CLOSED	NA	13474
29	ROAD_CLOSED	ROAD_CLOSED_CONSTRUCTION	129
30	ROAD_CLOSED	ROAD_CLOSED_EVENT	42393
31	ROAD_CLOSED	ROAD_CLOSED_HAZARD	13

Accident, Hazard, Jam, Road closed has NA as subtype. Accident has 24359 na in subtype, Hazard has only 3212, Jam has 55041, and Road closed has 13474. As for the number, Accident, Jame, and Road closed will have enough information to generate sub-subtypes.

List ACCIDENT - Minor - Major - Unclassified

JAM - Moderate - Heavy - Standstill - Light - Unclassified

HAZARD - Road - Hazard - Car_Stopped - Consturction - Emergency_Vihicle - Ice - Lane_Closed - Object - Pot_Hole - Road_Kill - Shoulder - Hazard - Animals - Car_Stopped - Missing_sign - Weather - Flood - Fog - Snow - Unclassified

ROAD_CLOSED - Consturction - Even - Hazard - Unclassified

Yes, I believe we should keep the NA, since they have large amount of data entries.

4.

a.

```
# create hierachy
df_new = pd.DataFrame(columns=['type', 'subtype', 'update_type', 'update_subtype', 'subsu
```

b.

```
# create hierarchy
hierarchy = {
    'ACCIDENT': {
        'Minor': ['Unclassified'],
        'Major': ['Unclassified'],
       "Unclassified" : ["Unclassfied"]
    },
    'JAM': {
        'Moderate': ['Unclassified'],
        'Heavy': ['Unclassified'],
        'Standstill': ['Unclassified'],
        'Light': ['Unclassified'],
       "Unclassified" : ["Unclassfied"]
    },
    'HAZARD': {
        'Road': ['Hazard', 'Car_Stopped', 'Construction', 'Emergency_Vehicle',
                 'Ice', 'Lane_Closed', 'Object', 'Pot_Hole', 'Road_Kill',"Traffict_light_
        'Shoulder': ['Hazard', 'Animals', 'Car_Stopped', 'Missing_sign',
                     'Weather', 'Flood','Fog', "Hail",'Snow'],
        'Unclassified': []
    },
    'ROAD_CLOSED': {
        'Construction': ['Unclassified'],
        'Event': ['Unclassified'],
        'Hazard': ['Unclassified'],
       "Unclassified" : ["Unclassfied"]
    }
}

crosswalk = []
for t, subtypes in hierarchy.items():
    for s, subsubtypes in subtypes.items():
        if subsubtypes:
            for subsub in subsubtypes:
                crosswalk.append({'update_type': t, 'update_subtype': s, 'subsubtype': su
        else:
            crosswalk.append({'update_type': t, 'update_subtype': s, 'subsubtype': 'Uncla
crosswalk = pd.DataFrame(crosswalk)

df_new["type"] = grouped_df["type"]
df_new["subtype"] = grouped_df["subtype"]
df_new["update_type"] = crosswalk["update_type"]
df_new["update_subtype"] = crosswalk["update_subtype"]
df_new["subsubtype"] = crosswalk["subsubtype"]
```

```
df_new['subtype'].fillna('Unclassified', inplace=True)
```

```
/var/folders/xp/13j7_6qs0bjcjm3r53h36v0h0000gn/T/ipykernel_4321/2967832714.py:1:
FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through
chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the
intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col:
value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the
operation inplace on the original object.
```

```
df_new['subtype'].fillna('Unclassified', inplace=True)
```

c.

```
# merge with the original dataset
df = df.merge(df_new, how='outer', on=['type', 'subtype'])
df.head(10)
```

	city	confidence	nThumbsUp	street	uuid	country	type	subtype	roadTy
0	Chicago, IL	2	NaN	DuSable Lake Shore Dr	926ee9d1-05e8-4bcf-8229-a201e474ea99	US	ACCIDENT	ACCIDENT_MAJOR	6
1	Chicago, IL	0	NaN	DuSable Lake Shore Dr	23dbf87f-2233-4f12-86a3-8b566b7051e7	US	ACCIDENT	ACCIDENT_MAJOR	6
2	Chicago, IL	1	NaN	DuSable Lake Shore Dr	84e5682d-7f94-45ba-81d6-f2d8a3a2e783	US	ACCIDENT	ACCIDENT_MAJOR	6
3	Chicago, IL	1	NaN	I-90 E	3f488e3c-72c9-4362-9813-b3aa8a8f6228	US	ACCIDENT	ACCIDENT_MAJOR	3
4	Chicago, IL	0	NaN	I-90 W	0c1f5bc2-fc54-43a8-b323-3cfadf5f7037	US	ACCIDENT	ACCIDENT_MAJOR	3
5	Chicago, IL	0	NaN	I-94 E	22fa06bf-66fb-4ccc-b29a-f2cd56a838dc	US	ACCIDENT	ACCIDENT_MAJOR	3

	city	confidence	nThumbsUp	street	uuid	country	type	subtype	roadTy
6	Chicago, IL	1	NaN	N Ashland Ave	ac0439e1-4659-42a3-8923-a1df475d5aa9	US	ACCIDENT	ACCIDENT_MAJOR	7
7	Chicago, IL	0	NaN	N California Ave	292eeae0-d1b2-4230-afbe-07f492f88896	US	ACCIDENT	ACCIDENT_MAJOR	2
8	Chicago, IL	0	NaN	W Fillmore St	bcd6f28e-1130-4715-9f16-e9cdbc2cbfdf	US	ACCIDENT	ACCIDENT_MAJOR	1
9	Chicago, IL	5	NaN	I-90 E	5bf94814-321a-420b-b63e-b9f5ec9c50f0	US	ACCIDENT	ACCIDENT_MAJOR	3

d.

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

1.

```
df["longitude"] = 1
df["latitude"] = 2

for i in range(len(df)):
  txt = df["geo"][i]
  clean_txt = re.sub(r'POINT\(|\)', "", txt)
  split_txt = clean_txt.split()
  df["longitude"][i]= split_txt[0]
  df["latitude"] [i]= split_txt[1]

df.head()
```

```
/var/folders/xp/13j7_6qs0bjcjm3r53h36v0h0000gn/T/ipykernel_4321/891565506.py:8:
FutureWarning: ChainedAssignmentError: behaviour will change in pandas 3.0!
You are setting values through chained assignment. Currently this works in certain cases,
but when using Copy-on-Write (which will become the default behaviour in pandas 3.0) this
will never work to update the original DataFrame or Series, because the intermediate
object on which we are setting values will behave as a copy.
A typical example is when you are setting values in a column of a DataFrame, like:

df["col"][row_indexer] = value

Use `df.loc[row_indexer, "col"] = values` instead, to perform the assignment in a single
step and ensure this keeps updating the original `df`.
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

  df["longitude"][i]= split_txt[0]
/var/folders/xp/13j7_6qs0bjcjm3r53h36v0h0000gn/T/ipykernel_4321/891565506.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df["longitude"][i]= split_txt[0]
/var/folders/xp/13j7_6qs0bjcjm3r53h36v0h0000gn/T/ipykernel_4321/891565506.py:8:
FutureWarning: Setting an item of incompatible dtype is deprecated and will raise an
error in a future version of pandas. Value '-87.647848' has dtype incompatible with
int64, please explicitly cast to a compatible dtype first.
  df["longitude"][i]= split_txt[0]
/var/folders/xp/13j7_6qs0bjcjm3r53h36v0h0000gn/T/ipykernel_4321/891565506.py:9:
FutureWarning: ChainedAssignmentError: behaviour will change in pandas 3.0!
You are setting values through chained assignment. Currently this works in certain cases,
but when using Copy-on-Write (which will become the default behaviour in pandas 3.0) this
will never work to update the original DataFrame or Series, because the intermediate
object on which we are setting values will behave as a copy.
A typical example is when you are setting values in a column of a DataFrame, like:

df["col"][row_indexer] = value

Use `df.loc[row_indexer, "col"] = values` instead, to perform the assignment in a single
step and ensure this keeps updating the original `df`.

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

  df["latitude"] [i]= split_txt[1]
/var/folders/xp/13j7_6qs0bjcjm3r53h36v0h0000gn/T/ipykernel_4321/891565506.py:9:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df["latitude"] [i]= split_txt[1]
/var/folders/xp/13j7_6qs0bjcjm3r53h36v0h0000gn/T/ipykernel_4321/891565506.py:9:
FutureWarning: Setting an item of incompatible dtype is deprecated and will raise an
error in a future version of pandas. Value '41.967935' has dtype incompatible with int64,
please explicitly cast to a compatible dtype first.
  df["latitude"] [i]= split_txt[1]
```

	city	confidence	nThumbsUp	street	uuid	country	type	subtype	roadTyp
0	Chicago, IL	2	NaN	DuSable Lake	926ee9d1-05e8-4bcf-	US	ACCIDENT	ACCIDENT_MAJOR	6

	city	confidence	nThumbsUp	street	uuid	country	type	subtype	roadTyp
				Shore Dr	8229-a201e474ea99				
1	Chicago, IL	0	NaN	DuSable Lake Shore Dr	23dbf87f-2233-4f12-86a3-8b566b7051e7	US	ACCIDENT	ACCIDENT_MAJOR	6
2	Chicago, IL	1	NaN	DuSable Lake Shore Dr	84e5682d-7f94-45ba-81d6-f2d8a3a2e783	US	ACCIDENT	ACCIDENT_MAJOR	6
3	Chicago, IL	1	NaN	I-90 E	3f488e3c-72c9-4362-9813-b3aa8a8f6228	US	ACCIDENT	ACCIDENT_MAJOR	3
4	Chicago, IL	0	NaN	I-90 W	0c1f5bc2-fc54-43a8-b323-3cfadf5f7037	US	ACCIDENT	ACCIDENT_MAJOR	3

b.

```
df['longitude'] = pd.to_numeric(df['longitude'], errors='coerce')
df['latitude'] = pd.to_numeric(df['latitude'], errors='coerce')

df['longitude'] = df['longitude'].round(2)
df['latitude'] = df['latitude'].round(2)
df['longitude_latitude'] = list(zip(df['longitude'], df['latitude']))
bin_count = df.groupby(["longitude_latitude"]).size().reset_index(name='count')
bin_sort = bin_count.sort_values(by = "count",ascending = False)
bin_sort
```

	longitude_latitude	count
492	(-87.65, 41.88)	21325
493	(-87.65, 41.89)	19996
175	(-87.75, 41.96)	16309
176	(-87.75, 41.97)	14570
458	(-87.66, 41.9)	14197
...	...	...
207	(-87.73, 41.69)	1
504	(-87.65, 42.0)	1
87	(-87.79, 41.76)	1
31	(-87.86, 41.95)	1

	longitude_latitude	count
658	(-87.59, 41.91)	1

700 rows × 2 columns
(-87.65 , 41.88) has the highest occurance with 21325 count

c. Collapse the data down to the level of aggregation needed to plot the top 10 latitude-longitude bins with the highest number of alerts for a chosen type and subtype (Note: no sub-subtype). Save DataFrame as top_alerts_map.csv file in the top_alerts_map folder you created. What is the level of aggregation in this case? How many rows does this DataFrame have?

```
#top 10 latitude-longitude bins
top_10_bins = bin_sort[0:10]
top_10_bins_df = df[df['longitude_latitude'].isin(top_10_bins['longitude_latitude'])]

top_alerts = top_10_bins_df.groupby(["longitude", "latitude","update_type","update_subtyp

top_alerts.to_csv("/Users/tsaili-ting/Uchicago/Year2/Y2Fall/Python2/ps6/top_alerts_map/to

top_alerts.shape
```

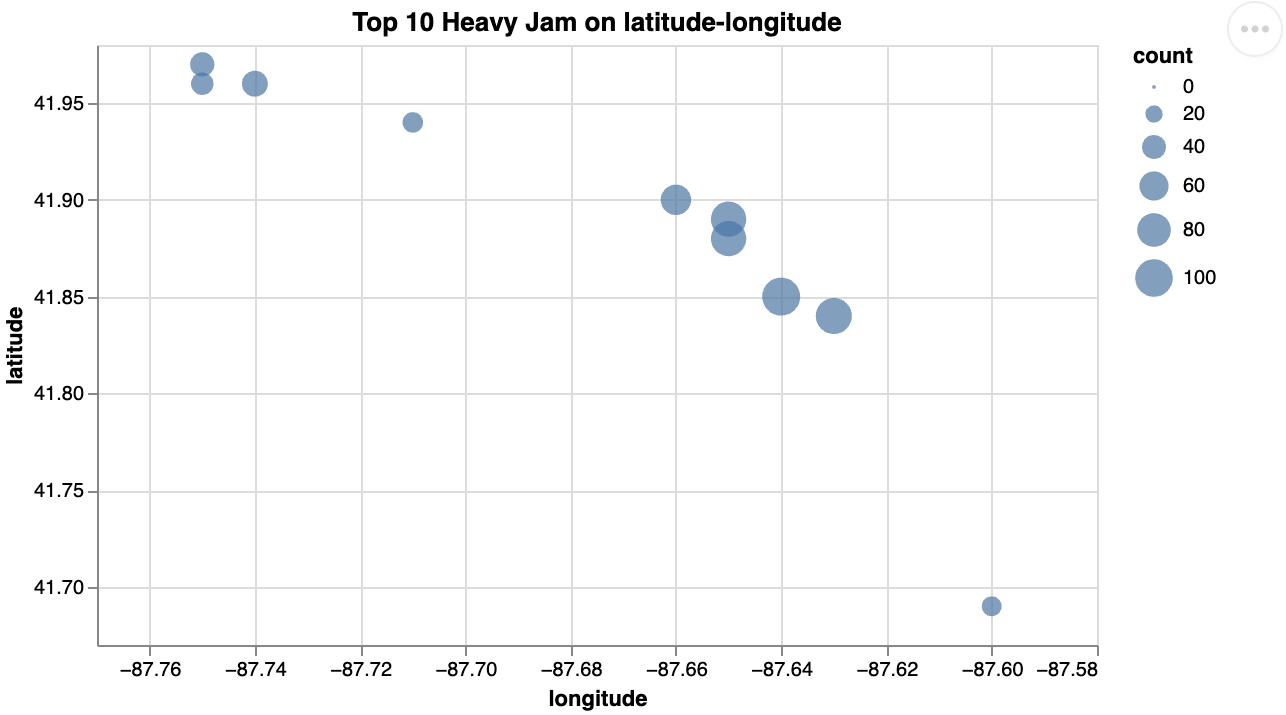
```
(136, 5)
```

```
top_10_bins_df.to_csv("/Users/tsaili-ting/Uchicago/Year2/Y2Fall/Python2/ps6/waze_top10.cs
```

The data is aggregated to the city's neighborhood level. There are 136 row.

2.

```
# look for top 10 heavy jam
top_heavy_jam = top_alerts.loc[(top_alerts["update_type"]=="JAM") & (top_alerts["update_s

# plot the

scatter = alt.Chart(top_heavy_jam).mark_circle().encode(
  x = alt.X("longitude",scale = alt.Scale(domain = [-87.77, -87.58])),
  y = alt.Y("latitude",scale = alt.Scale(domain = [41.67,41.98])),
  size = "count"
).properties(
    title='Top 10 Heavy Jam on latitude-longitude',
    width=500,
    height=300
)

scatter
```



3.

a.

```
url = "https://data.cityofchicago.org/api/geospatial/bbvz-uum9?method=export&format=GeoJS
folder_path = "/Users/tsaili-ting/Uchicago/Year2/Y2Fall/Python2/ps6/top_alerts_map"
file_path = os.path.join(folder_path, "Boundaries - Neighborhoods.geojson")
response = requests.get(url)
```

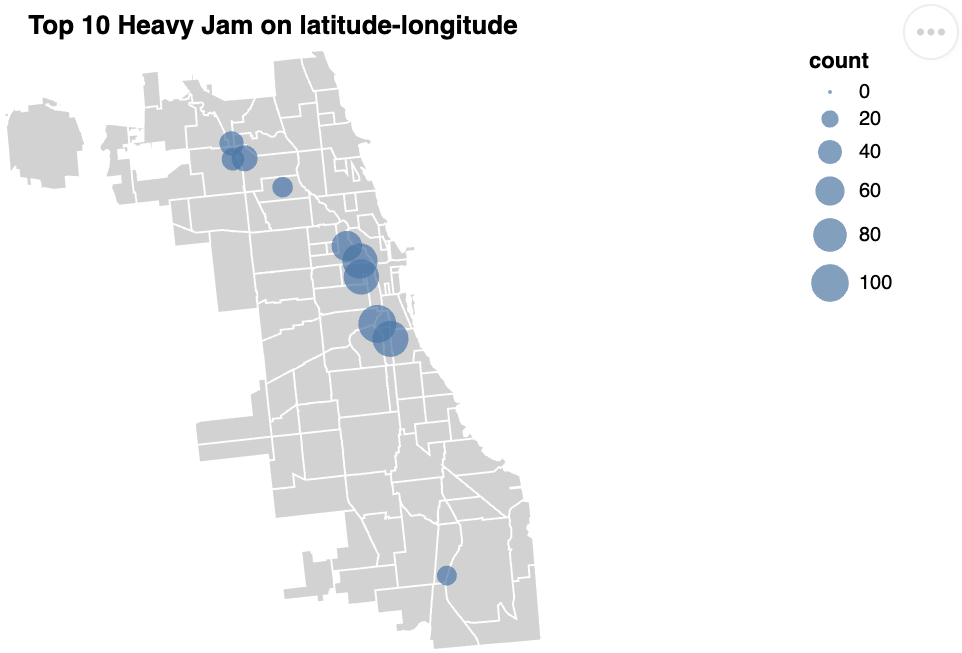
b.

```
# MODIFY ACCORDINGLY
#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.

```
background = alt.Chart(geo_data).mark_geoshape(
    fill='lightgray',
    stroke='white'
).project('albersUsa').properties(
    width=500,
    height=300
)
```

```
scatter = alt.Chart(top_heavy_jam).mark_circle().encode(
  longitude='longitude:Q',
  latitude='latitude:Q',
  size = "count"
).properties(
    title='Top 10 Heavy Jam on latitude-longitude',
    width=500,
    height=300
).project("albersUsa")

background+scatter
```



5.

a.

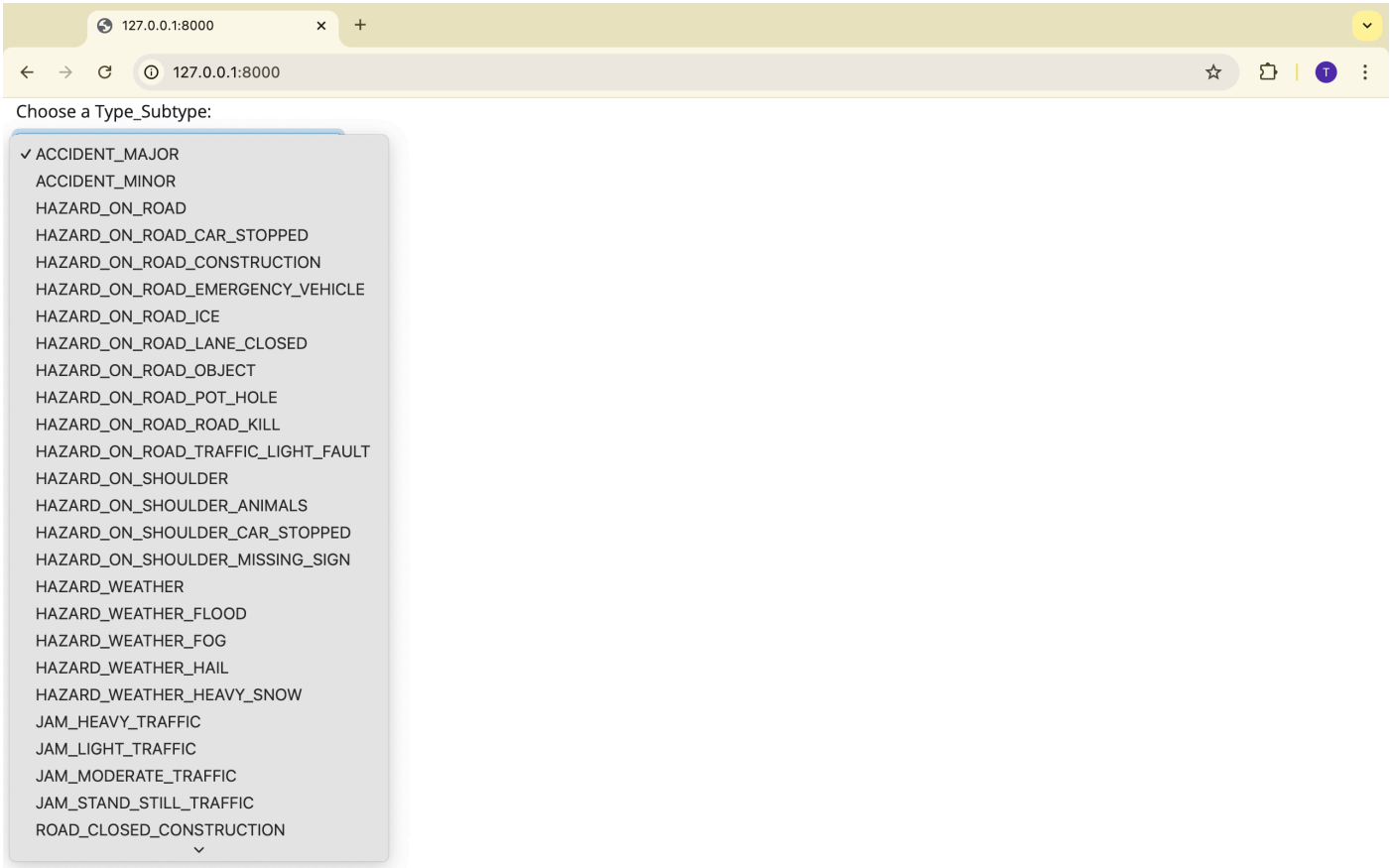
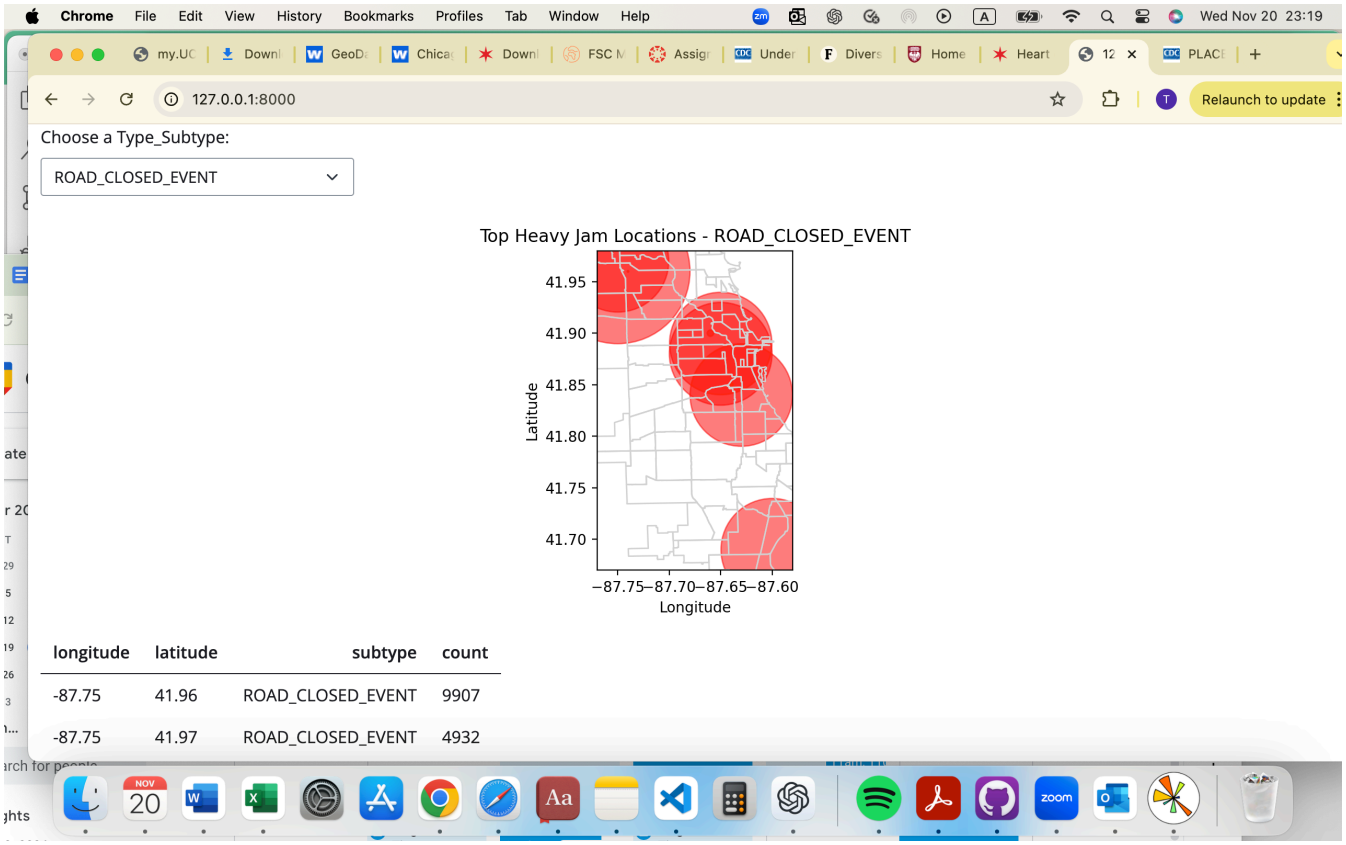


Image 1

b.

```
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("/Users/tsaili-ting/Uchicago/Year2/Y2Fall/Python2/ps6/top_alerts_map/
```

c.



The most common road closed due to event area is -87.75 41.96 ROAD_CLOSED_EVENT 9907, the area is close to downtown Chicago.

d. Question: Where is the most common major accident alert?

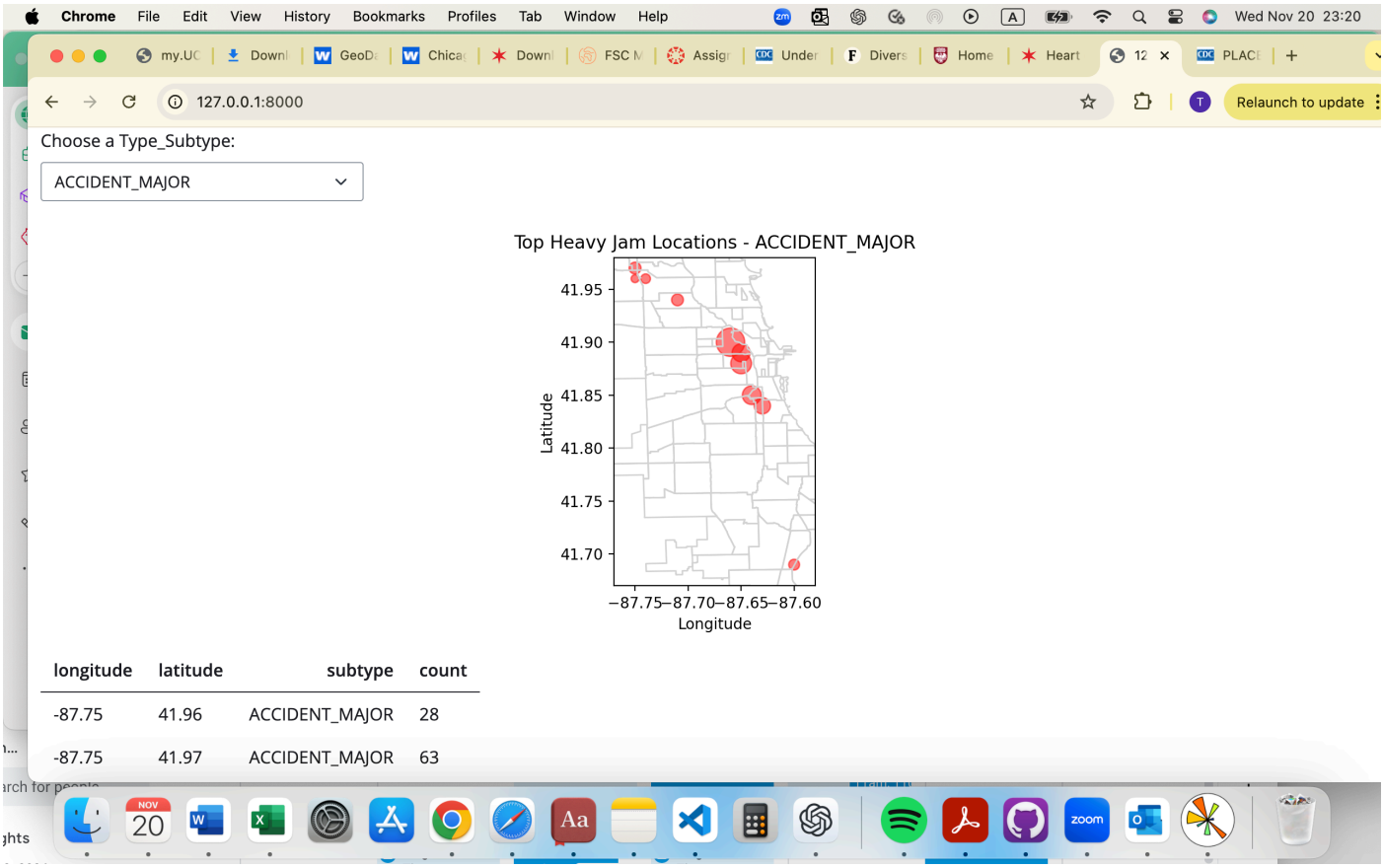


Image 1

Answer: the area (-87.66, 41.90) has the most common major accident alert. The area is more close to O'hare airport.

e.

We separate the type and subtype, and if users do not select a subtype, we will display the larger group of the alert type. This approach will make it easier to analyze the alerts in a more aggregated manner

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

1.

a.

```
df["ts"].head()
```

```
0    2024-02-04 03:02:09 UTC
1    2024-02-04 12:55:23 UTC
2    2024-02-04 01:18:47 UTC
3    2024-02-04 21:46:11 UTC
4    2024-02-04 16:46:46 UTC
Name: ts, dtype: object
```

I don't think it's a good idea. Since the time stamps are counted in seconds, and it probalbly won't be alerts happened in the same time in the seconds. Therefore, it has no meaning to collaps by this column.

b.

```
# create a column "hour"
df["ts"] = pd.to_datetime(df["ts"])
df["hour"] = df["ts"].dt.floor("h").dt.strftime("%H:00")

# compile the data
df_hour = df.groupby(["type","subtype", 'geo', 'geoWKT', 'longitude', 'latitude', 'longit
df_hour.to_csv("/Users/tsaili-ting/Uchicago/Year2/Y2Fall/Python2/ps6/top_alerts_map_byhou



len(df_hour)
```

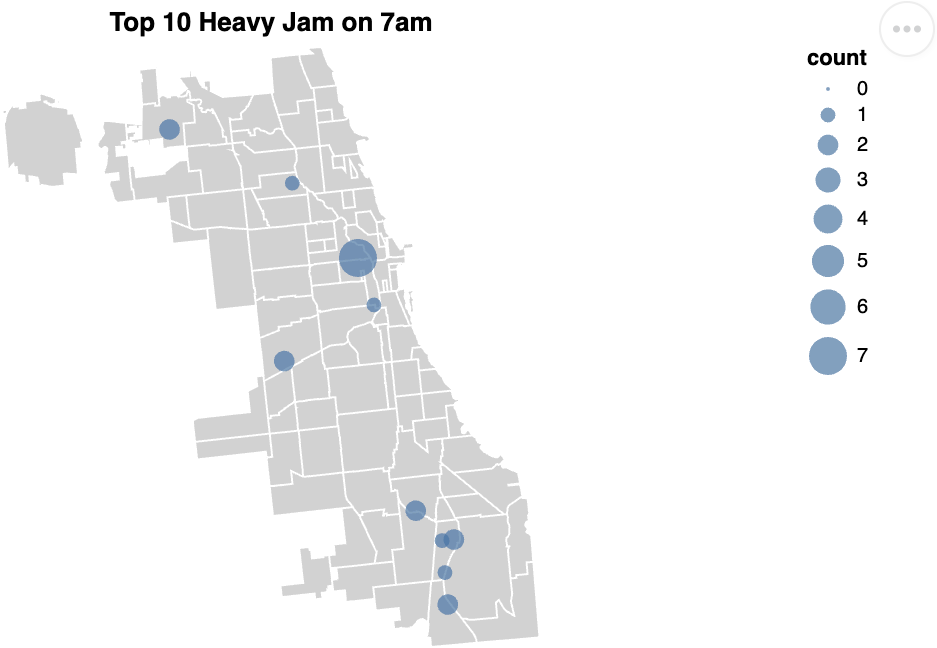
```
724709
```

There are 724709 rows

c.

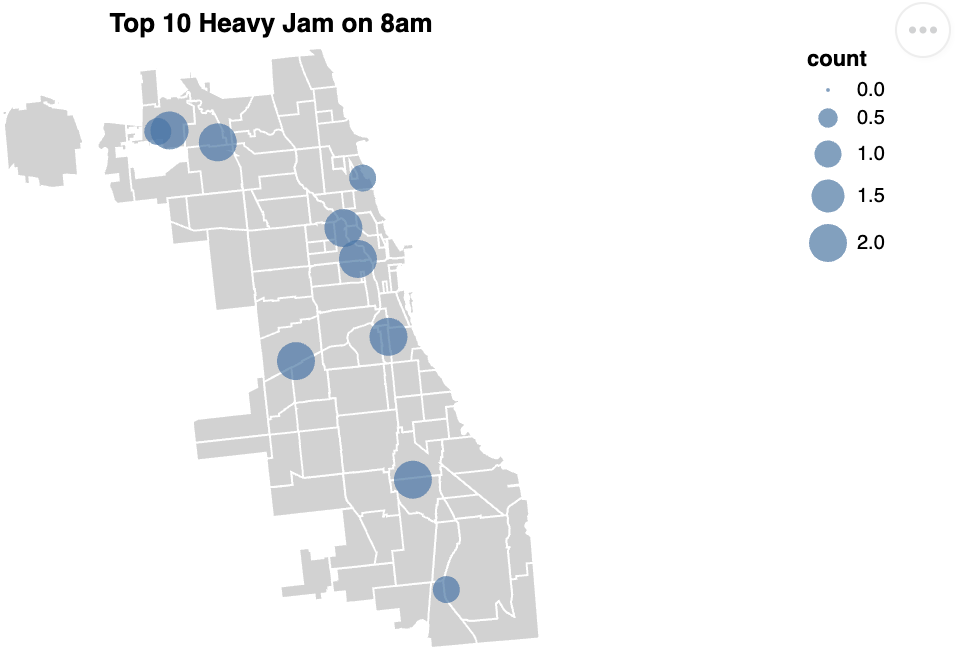
```
# prepare the filtered data
df_filtered = df_hour.loc[df_hour["subtype"]=="JAM_HEAVY_TRAFFIC"]
df_by_hour = df_filtered.groupby(["hour","longitude","latitude","longitude_latitude"]).ag
```

```
# create plot for 07:00
df_7am = df_by_hour.loc[df_by_hour["hour"]=="07:00"].sort_values("count",ascending=False)
plot_7am = alt.Chart(df_7am).mark_circle().encode(
      longitude='longitude:Q',
  latitude='latitude:Q',
  size = "count"
).properties(
    title='Top 10 Heavy Jam on 7am',
    width=500,
    height=300
).project("albersUsa")

background + plot_7am
```

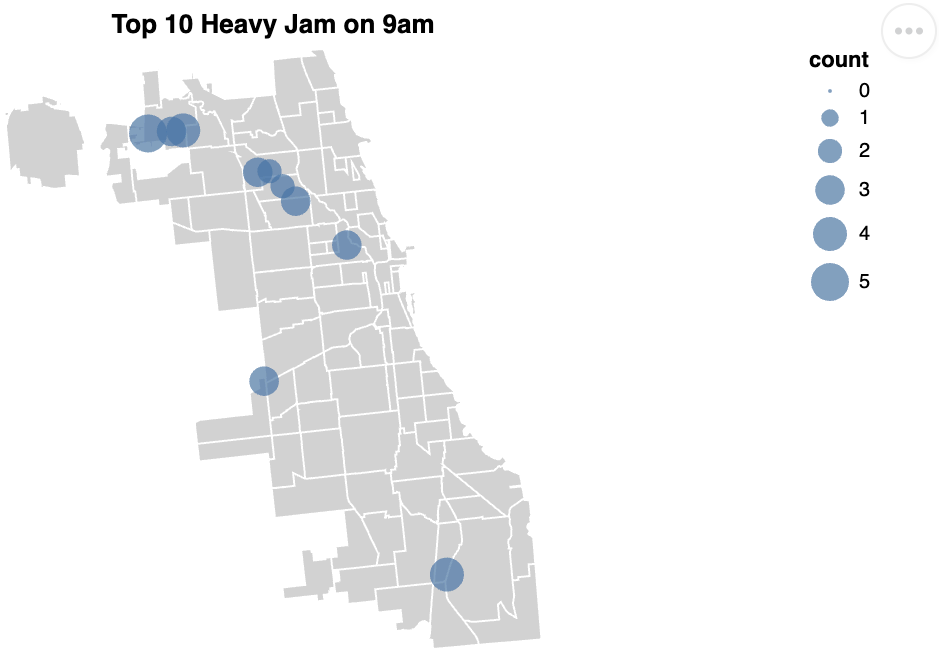



```
# create plot for 08:00
df_8am = df_by_hour.loc[df_by_hour["hour"]=="08:00"].sort_values("count",ascending=False)
plot_8am = alt.Chart(df_8am).mark_circle().encode(
      longitude='longitude:Q',
  latitude='latitude:Q',
  size = "count"
).properties(
    title='Top 10 Heavy Jam on 8am',
    width=500,
    height=300
).project("albersUsa")
```

```
background + plot_8am
```




```
# create plot for 09:00
df_9am = df_by_hour.loc[df_by_hour["hour"]=="09:00"].sort_values("count",ascending=False)
plot_9am = alt.Chart(df_9am).mark_circle().encode(
      longitude='longitude:Q',
  latitude='latitude:Q',
  size = "count"
).properties(
    title='Top 10 Heavy Jam on 9am',
    width=500,
    height=300
).project("albersUsa")

background + plot_9am
```



2.

a.

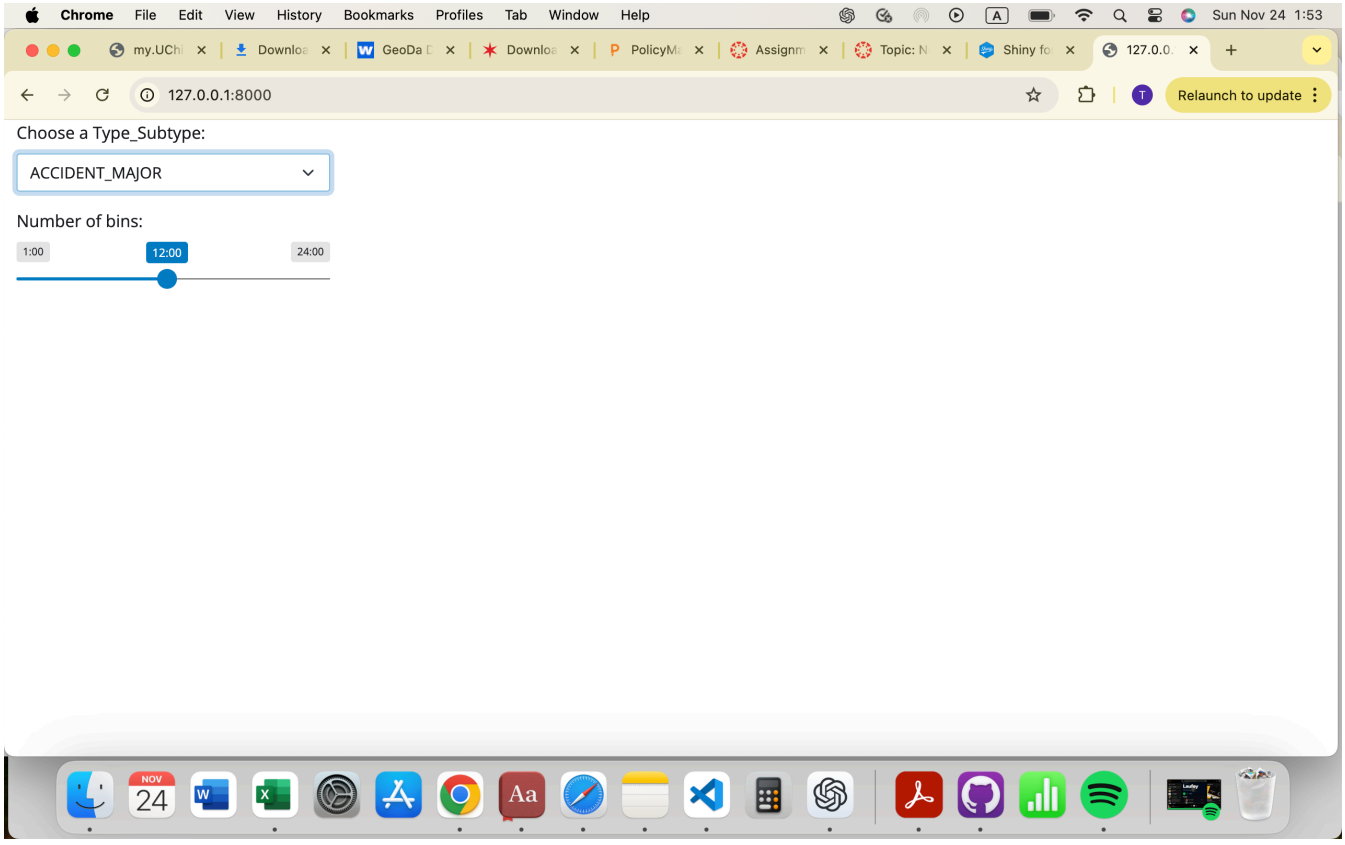


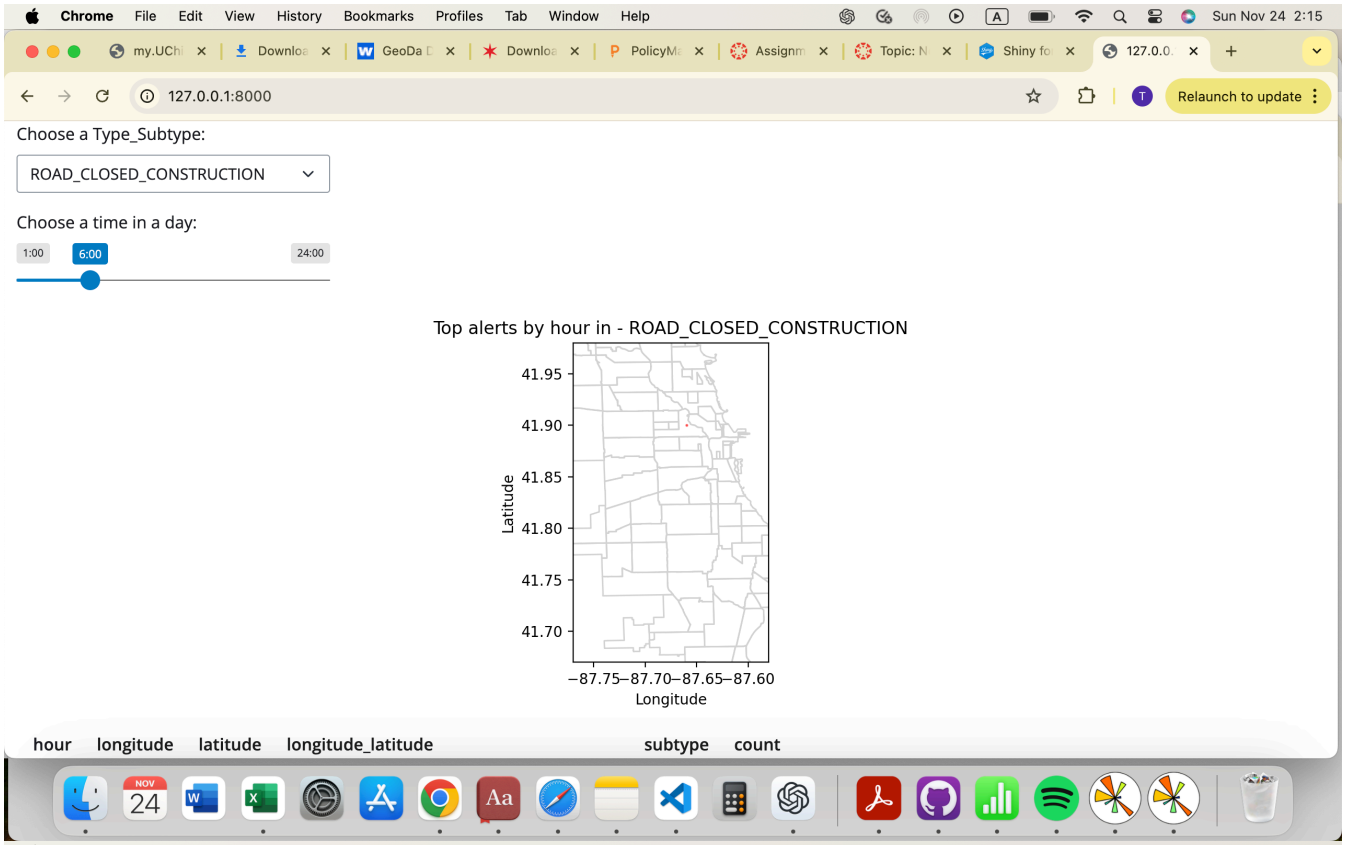
Image 1

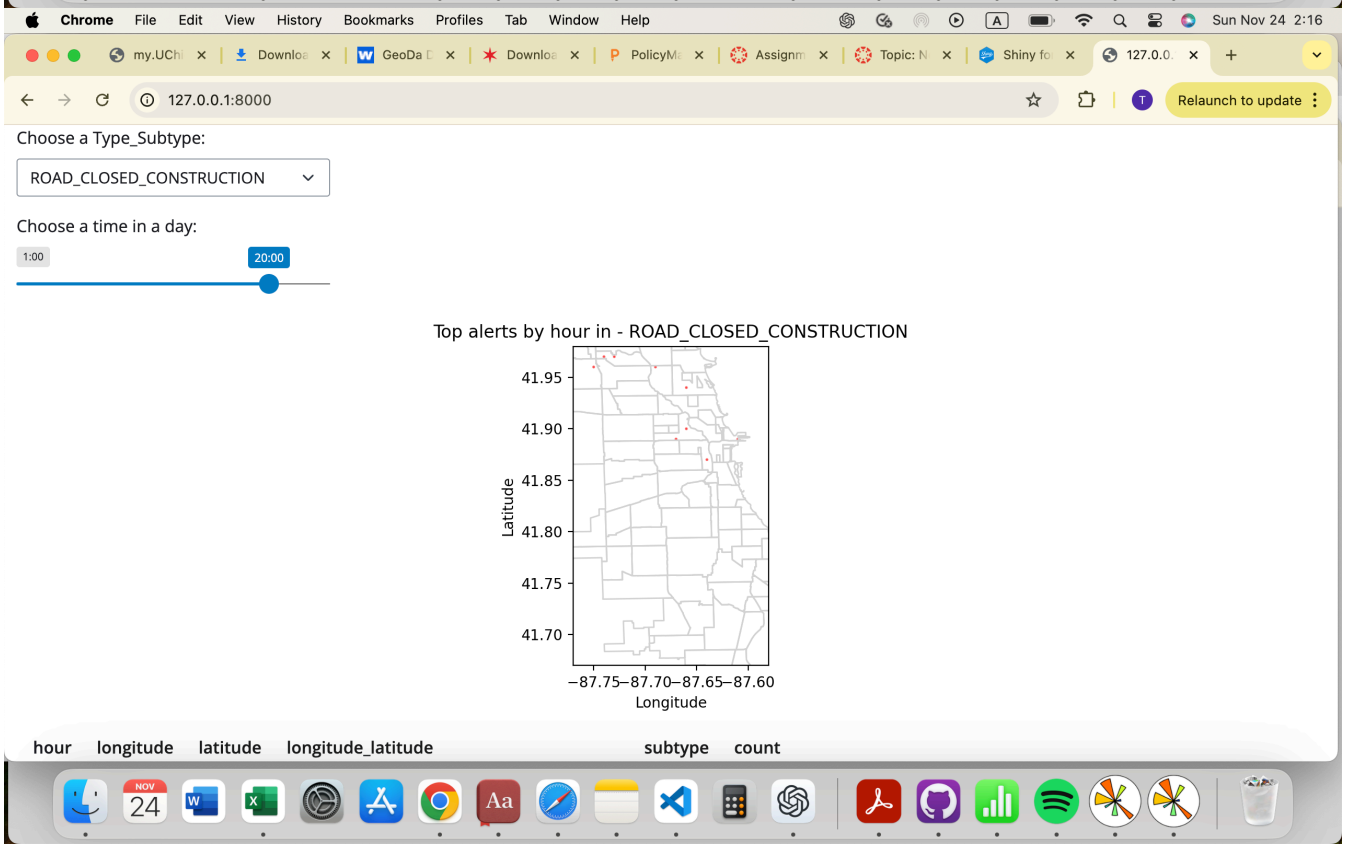
b.

```
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("/Users/tsaili-ting/Uchicago/Year2/Y2Fall/Python2/ps6/top_alerts_map_
```

C.





From above two plot we can see that in the morning(6am) there's not much construction happened. On the other hand, while it's in the night (8pm), there are more dots exist in the plot. Therefore, most of the construction happened at night time.

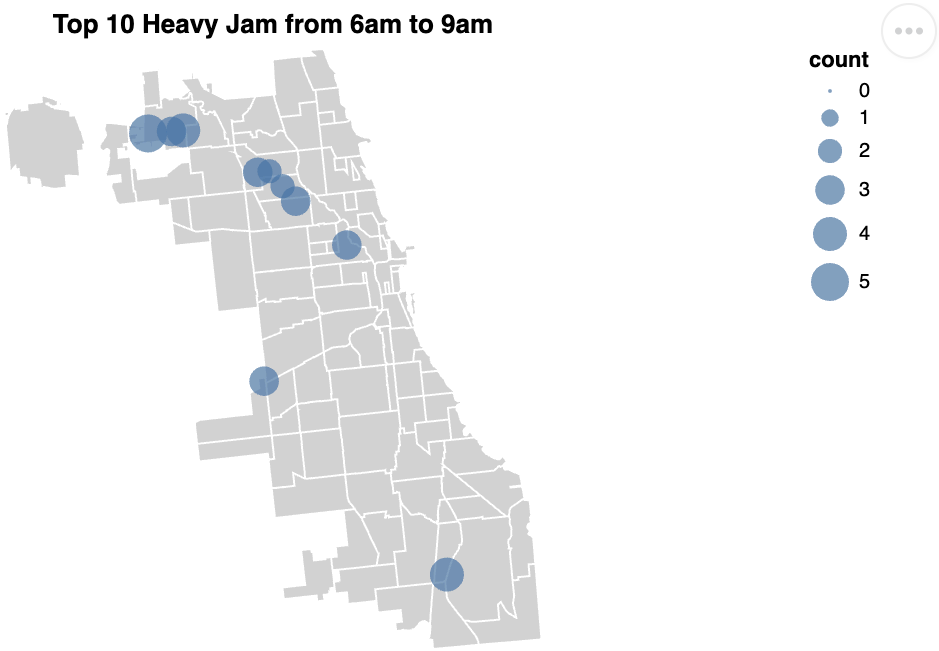
App #3: Top Location by Alert Type and Hour Dashboard (20 points)

1.

a. No, it will not be a good idea. If we collapsed data by range, then we will lose the pattern in anyother types of range and the granularity.

b.

```
df_6_9am = df_by_hour.loc[(df_by_hour["hour"] <="09:00")&(df_by_hour["hour"] >="06:00")].

plot_6_9am = alt.Chart(df_9am).mark_circle().encode(
      longitude='longitude:Q',
  latitude='latitude:Q',
  size = "count"
).properties(
    title='Top 10 Heavy Jam from 6am to 9am',
    width=500,
    height=300
).project("albersUsa")

background + plot_6_9am
```



2.

a.

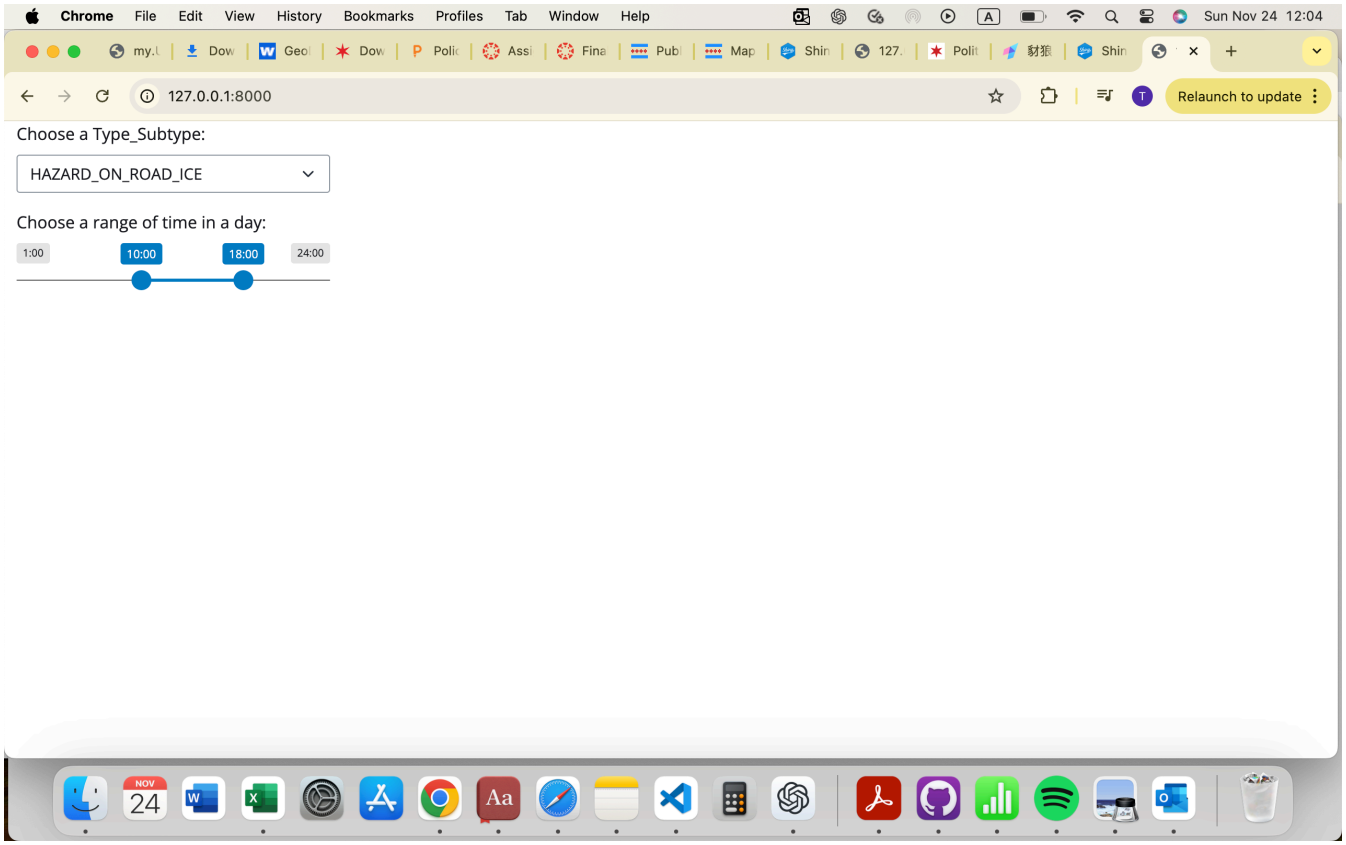


Image 1

b.

```
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("/Users/tsaili-ting/Uchicago/Year2/Y2Fall/Python2/ps6/top_alerts_map_
```

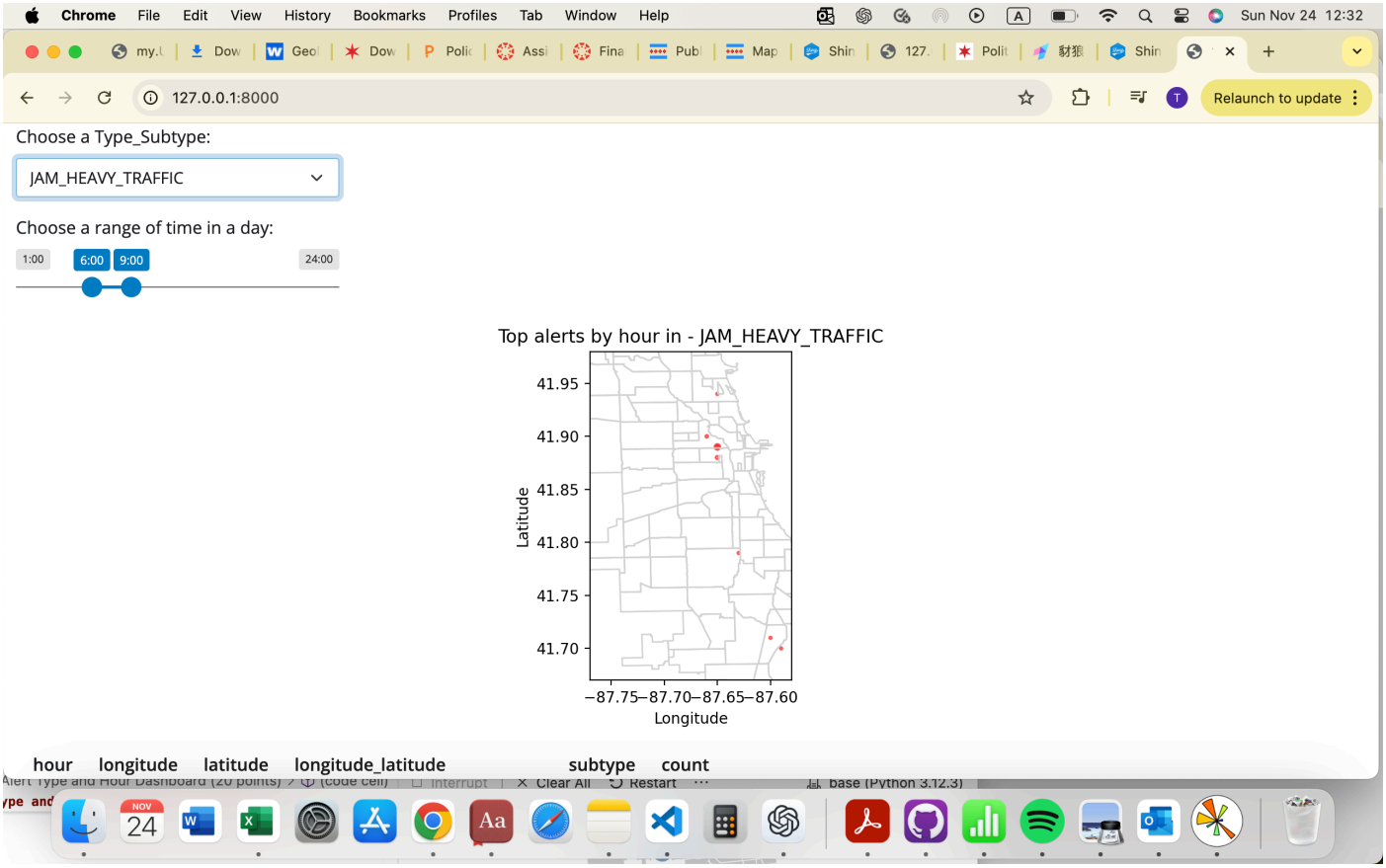


Image 1

3.

a.
b.
c.
d.