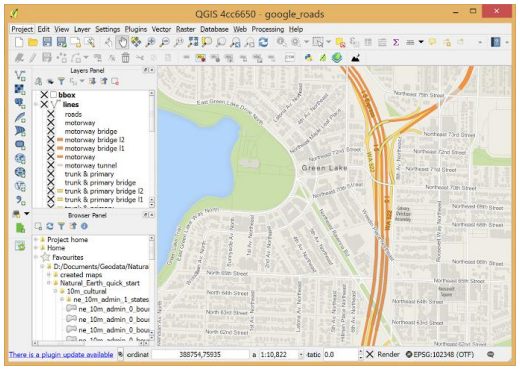
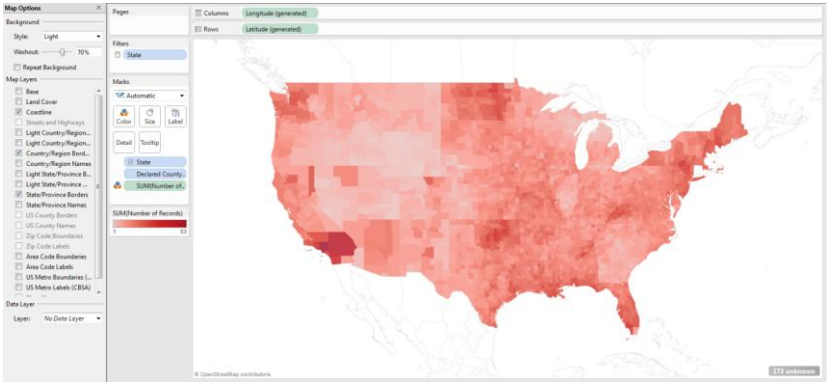
Spatial Analysis and Visualization with Python

Veera Muangsin

Spatial Analysis & Viz Tools

- GIS software
  - QGIS https://www.qgis.org/
  - ESRI ArcGIS
- Business Intelligence
  - PowerBI
  - Tableau
- Python Libraries
  - Geopandas
  - Folium
  - PyDeck



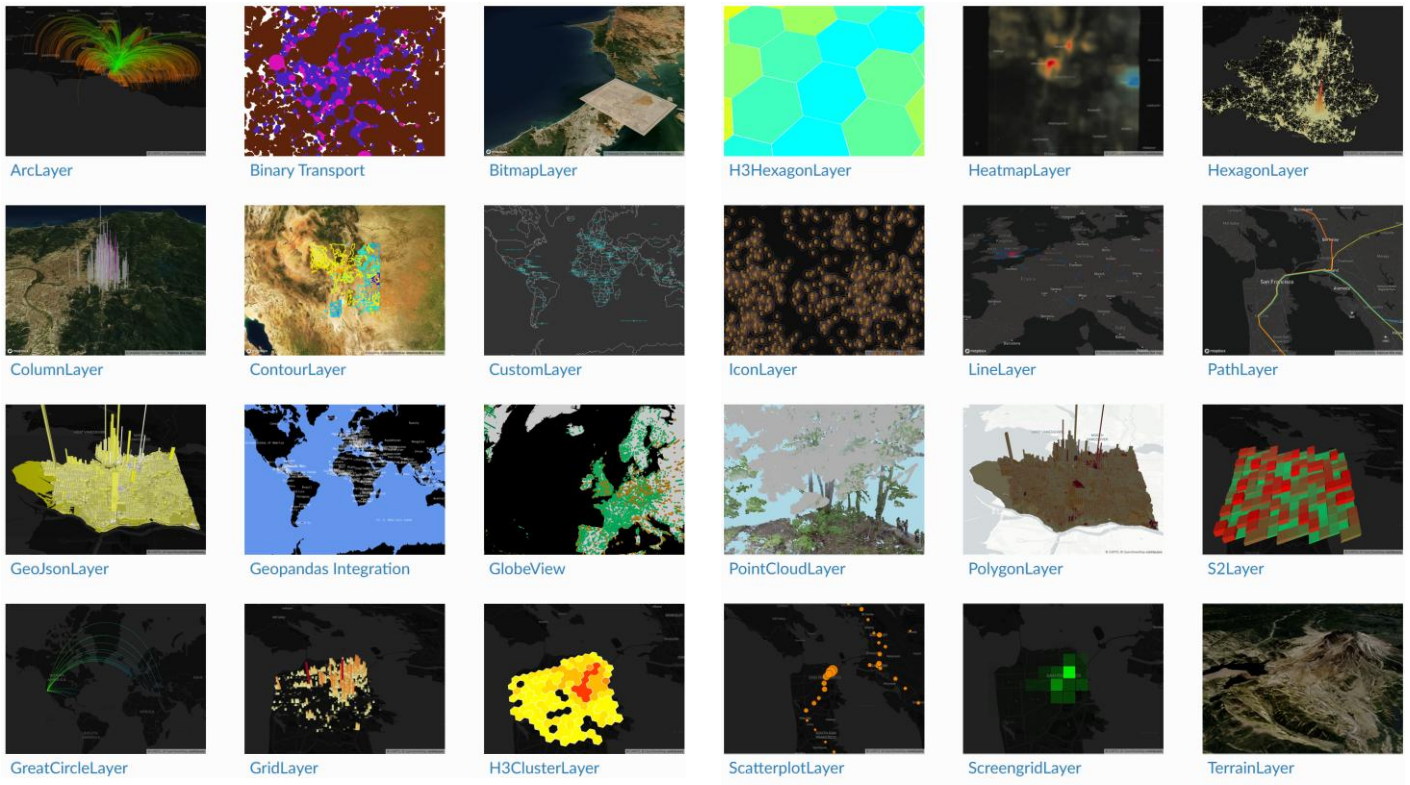


Python Geospatial Libraries

Layer	Package	Description
Spatial Data I/O	**fiona**	Reading, writing, and processing geospatial vector data (Shapefile, GeoJSON)
	rasterio	Reading, writing, and processing geospatial raster data (GeoTIFF)
Geoprocessing	**geopandas**	Extend Pandas to handle geospatial data
	shapely	Provide tools for geometric operations and spatial analysis
	pyproj	Handle coordinate transformations and projections
Geovisualization	**folium**	Interactive maps based on leaflet.js
	pydeck	Visualizing large-scale datasets on interactive and intuitive maps powered by deck.gl, a WebGL-powered framework.
	ipyleaflet	Interactive maps directly within Jupyter notebooks
	kepler	Browser-based interactive maps for large number of data points
	geoplot	Static high-quality maps
Spatial Analysis	**PySAL**	Spatial econometrics, exploratory spatial data analysis, and geographic data visualization.
	Scikit-learn **Scikit-gstat** **Scikit-mobility**	Machine Learning Libraries: statistical modeling, geostatistical analysis, and mobility pattern mining, suitable for a range of tasks from predictive analytics to spatial data interrogation and movement prediction.

PyDeck

- PyDeck is a Python wrapper for the JavaScript library Deck.gl, which is a powerful, web-based data visualization framework created by Uber.
  - https://deck.gl/
  - https://deckgl.readthedocs.io/en/latest/
- `pip install pydeck`



Pydeck Components

- Layer
  - Define a layer to display on the map
  - `pdk.Layer()`
  - https://deckgl.readthedocs.io/en/latest/layer.html

- ViewState
  - In PyDeck, a viewport represents the area through which users view the map. The viewport is
  - configured via a ViewState object.
  - `pdk.ViewState()`
  - `pdk.data_utils.compute_view()`
  - https://deckgl.readthedocs.io/en/latest/view_state.html
- Draw
  - Render the map based on the layer(s), initial view state, and other configurations.
  - `pdk.Deck()`
  - https://deckgl.readthedocs.io/en/latest/deck.html

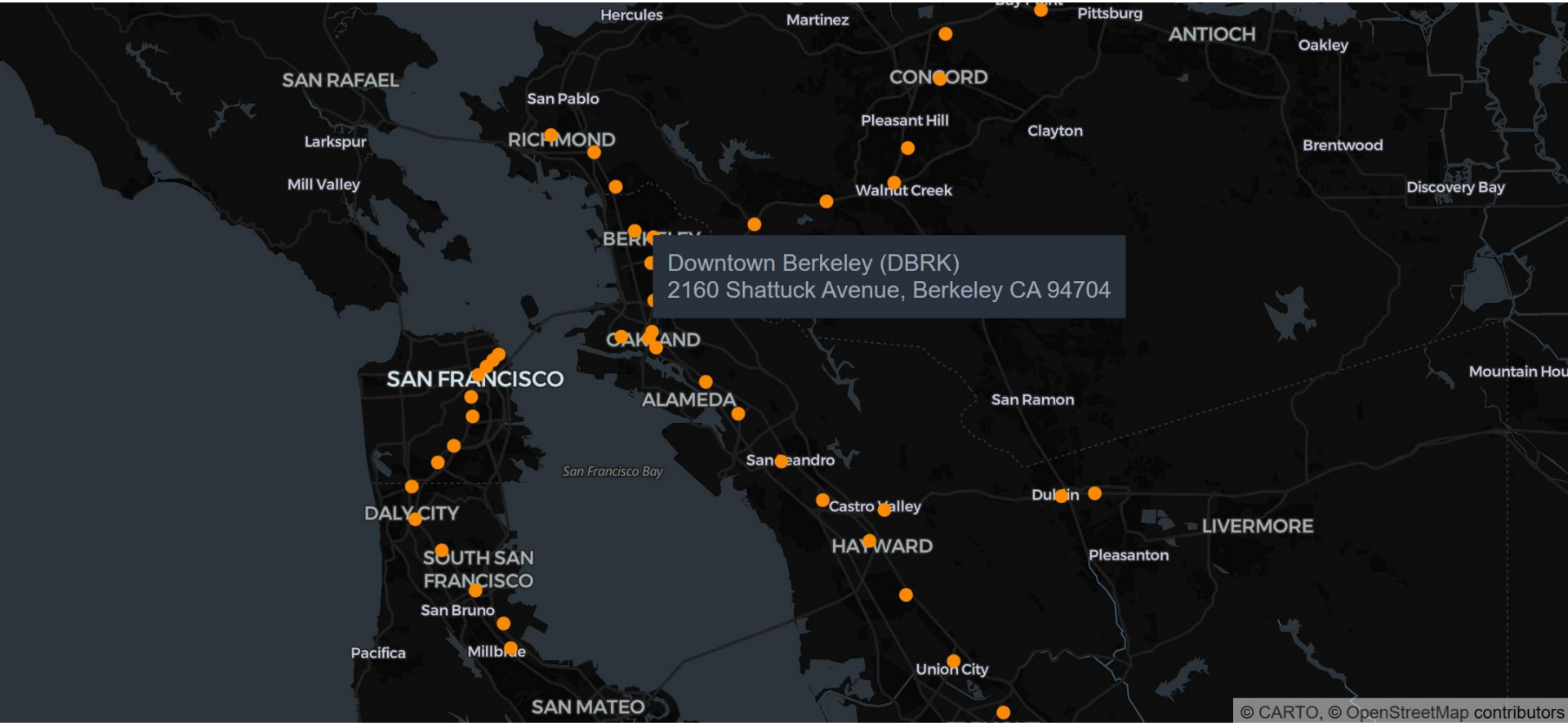
Pydeck: Scatter Plot

Simplify version of the example in https://deckgl.readthedocs.io/en/latest/gallery/scatterplot_layer.html

```
import pydeck as pdk
import pandas as pd

df = pd.read_json("https://raw.githubusercontent.com/visgl/deck.gl-data/master/website/bart-stations.json")

# Define a layer to display on a map
layer = pdk.Layer(
    "ScatterplotLayer",
    df,
    get_position="coordinates",
    get_radius=500,
    get_fill_color=[255, 140, 0],
    pickable=True
)

# Set the viewport location
view_state = pdk.ViewState(latitude=37.7749295, longitude=-122.4194155, zoom=10)

# Render
r = pdk.Deck(layers=[layer], initial_view_state=view_state, tooltip={"text": "{name}\n{address}"})

# Save as HTML
r.to_html("pydeck_scatterplot.html")
```

	name	code	address	entries	exits	coordinates
0	Lafayette (LAFY)	LF	3601 Deer Hill Road, Lafayette CA 94549	3481	3616	[-122.123801, 37.893394]
1	12th St. Oakland City Center (12TH)	12	1245 Broadway, Oakland CA 94612	13418	13547	[-122.271604, 37.803664]
2	16th St. Mission (16TH)	16	2000 Mission Street, San Francisco CA 94110	12409	12351	[-122.419694, 37.765062]

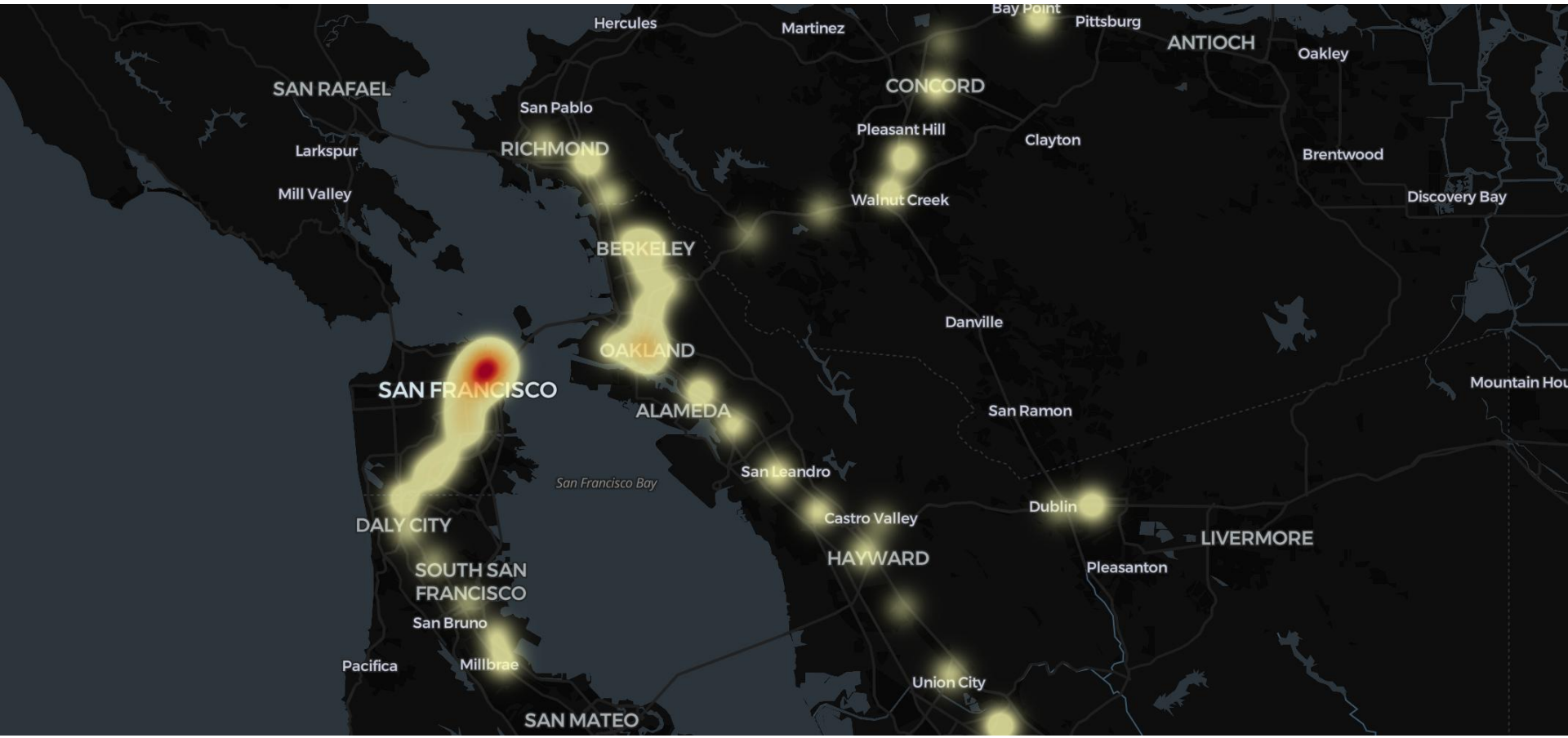
Pydeck: Scatter Plot



Pydeck: Heatmap

```
import pydeck as pdk
import pandas as pd

df = pd.read_json("https://raw.githubusercontent.com/visgl/deck.gl-data/master/website/bart-stations.json")

# Define a layer to display on a map
layer = pdk.Layer(
    "HeatmapLayer",
    df,
    get_position="coordinates",
    get_weight="exits",
    opacity=0.6,
)

# Set the viewport location
view_state = pdk.ViewState(latitude=37.7749295, longitude=-122.4194155, zoom=10)

# Render
pdk.Deck(layers=[layer], initial_view_state=view_state)
```

Pydeck: Heatmap



Streamlit

https://docs.streamlit.io/get-started/tutorials/create-an-app

streamlit_uber.py

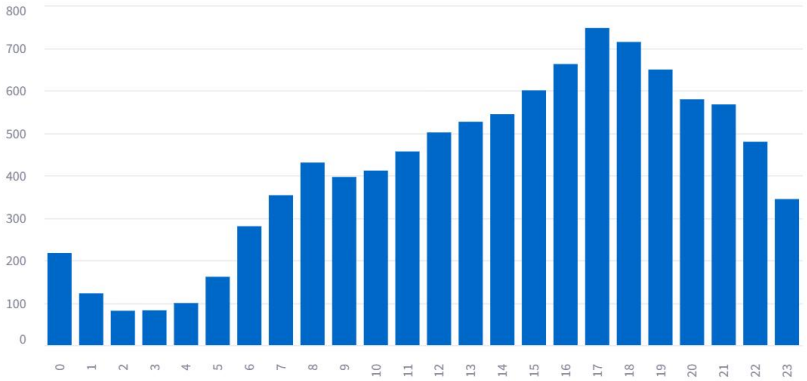
```
import streamlit as st
import pandas as pd
import numpy as np

st.title('Uber pickups in NYC')

DATE_COLUMN = 'date/time'
DATA_URL = ('https://s3-us-west-2.amazonaws.com/streamlit-demo-data/uber-raw-data-sep14.csv.gz')

@st.cache_data
def load_data(nrows):
    data = pd.read_csv(DATA_URL, nrows=nrows)
    lowercase = lambda x: str(x).lower()
    data.rename(lowercase, axis='columns', inplace=True)
    data[DATE_COLUMN] = pd.to_datetime(data[DATE_COLUMN])
    return data

data_load_state = st.text('Loading data...')
data = load_data(10000)
data_load_state.text("Done! (using st.cache_data)")

if st.checkbox('Show raw data'):
    st.subheader('Raw data')
    st.write(data)

st.subheader('Number of pickups by hour')
hist_values = np.histogram(data[DATE_COLUMN].dt.hour, bins=24, range=(0,24))[0]
st.bar_chart(hist_values)

# Some number in the range 0-23
hour_to_filter = st.slider('hour', 0, 23, 17)
filtered_data = data[data[DATE_COLUMN].dt.hour == hour_to_filter]

st.subheader('Map of all pickups at %s:00' % hour_to_filter)
st.map(filtered_data)
```

Uber pickups in NYC

Done! (using st.cache_data)

Show raw data

Number of pickups by hour

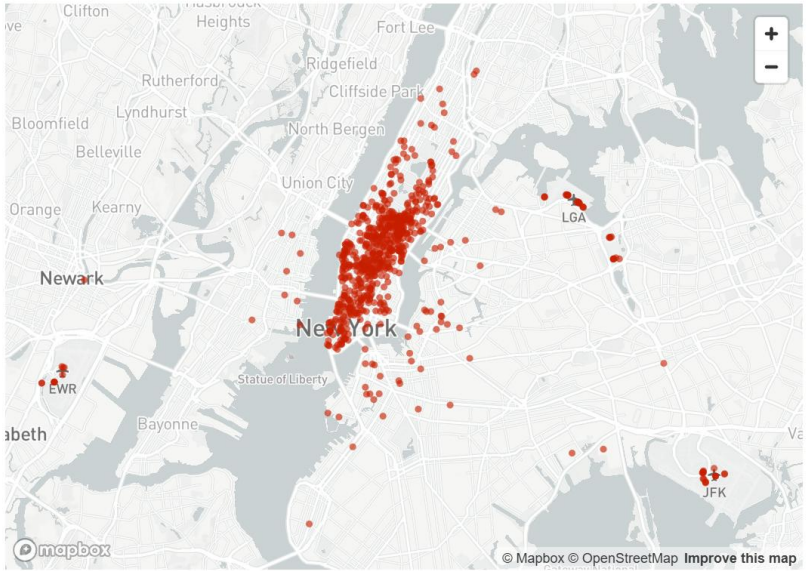


hour

17

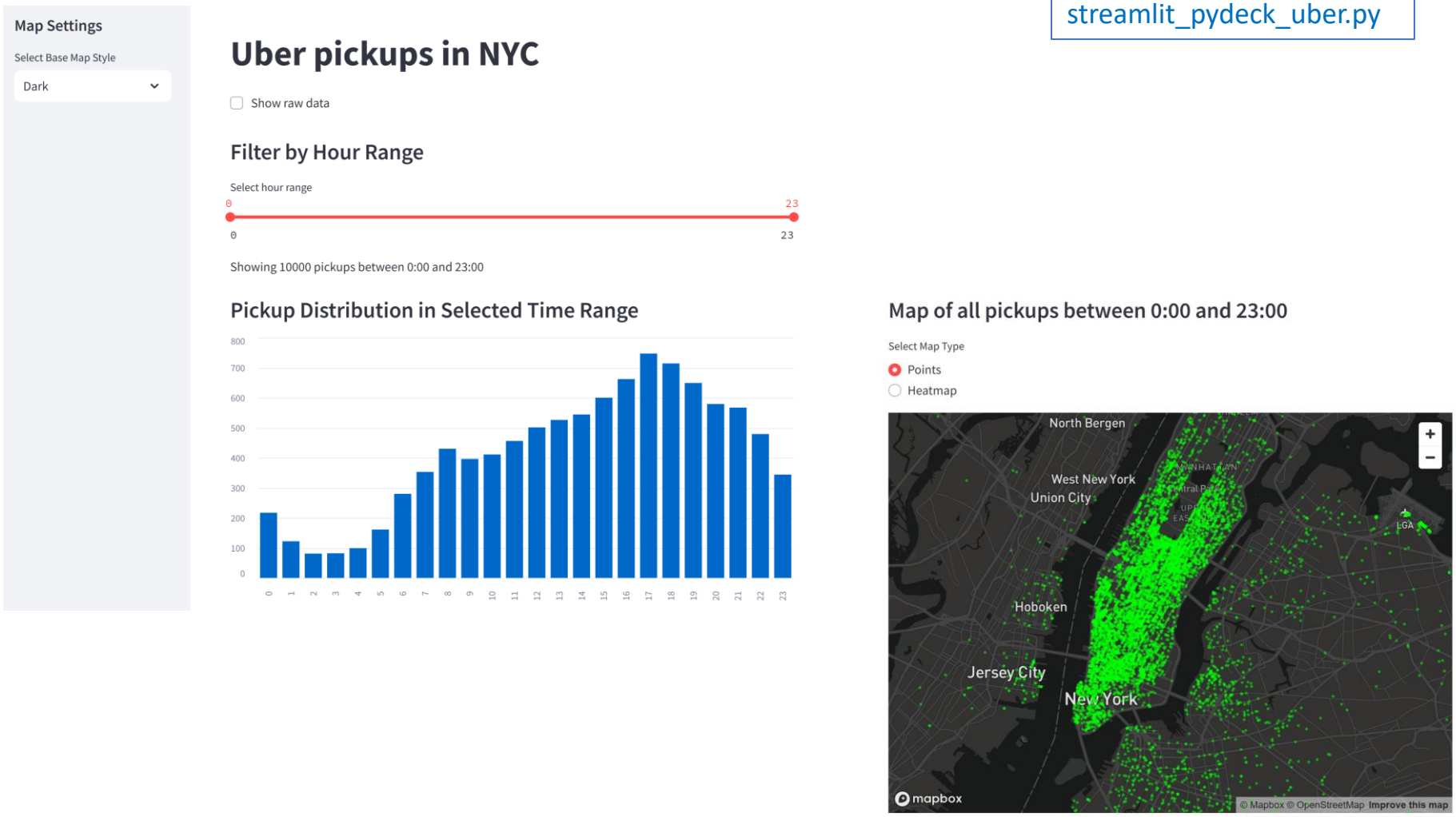
0 23

Map of all pickups at 17:00



Streamlit + PyDeck: Uber

streamlit_pydeck_uber.py



Streamlit + PyDeck

Map Type

- (•) ScatterplotLayer
- ( ) HeatmapLayer

Map Style

- ( ) mapbox://styles/mapbox/light-v11
- (•) mapbox://styles/mapbox/dark-v11
- ( ) mapbox://styles/mapbox/streets-v11
- ( ) mapbox://styles/mapbox/satellite-v9

Opacity

0.50

0.00 — 1.00

Radius Scale

5.00

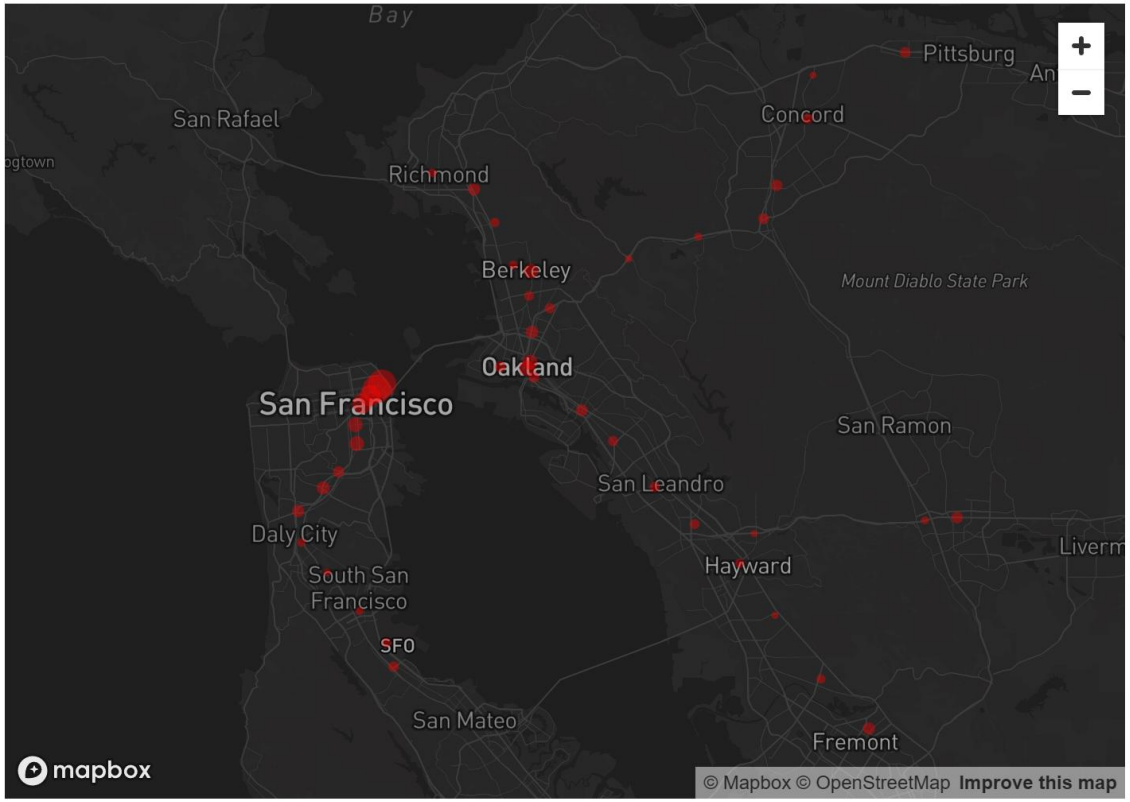
1.00 — 10.00

Color Palette

- (•) Reds
- ( ) Blues
- ( ) Greens
- ( ) Purples
- ( ) Oranges

PyDeck Demo

Map



	name	code	address	entries	exits
0	Lafayette (LAFY)	LF	3601 Deer Hill Road, Lafayette CA 94549	3,481	3,61
1	12th St. Oakland City Center (12TH)	12	1245 Broadway, Oakland CA 94612	13,418	13,54

Streamlit + PyDeck (1 of 2)

streamlit_pydeck_demo.py

```
import streamlit as st
import pandas as pd
import pydeck as pdk
import math

# Function to load data
@st.cache_data
def load_data():
    df = pd.read_json("https://raw.githubusercontent.com/visgl/deck.gl-data/master/website/bart-stations.json")
    df[['longitude', 'latitude']] = pd.DataFrame(df['coordinates'].tolist(), index=df.index)
    df["exits_radius"] = df["exits"].apply(lambda exits_count: math.sqrt(exits_count))
    return df

df = load_data()

# Sidebar controls
map_layer_type = st.sidebar.radio('Map Type', ["ScatterplotLayer", "HeatmapLayer"])
map_style = st.sidebar.radio(
    "Map Style",
    [
        "mapbox://styles/mapbox/dark-v11",
        "mapbox://styles/mapbox/light-v11",
        "mapbox://styles/mapbox/streets-v11",
        "mapbox://styles/mapbox/satellite-v9",
    ]
)
opacity = st.sidebar.slider('Opacity', min_value=0.0, max_value=1.0, value=0.5)
radius_scale = st.sidebar.slider('Radius Scale', min_value=1.0, max_value=10.0, value=5.0)
color_choices = st.sidebar.radio('Color Palette', ['Reds', 'Blues', 'Greens', 'Purples', 'Oranges'])

# Color mapping based on choice
color_mapping = {
    "Reds": [255, 0, 0, 140],
    "Blues": [0, 0, 255, 140],
    "Greens": [0, 255, 0, 140],
    "Purples": [128, 0, 128, 140],
    "Oranges": [255, 165, 0, 140]
}
color = color_mapping[color_choices]
```

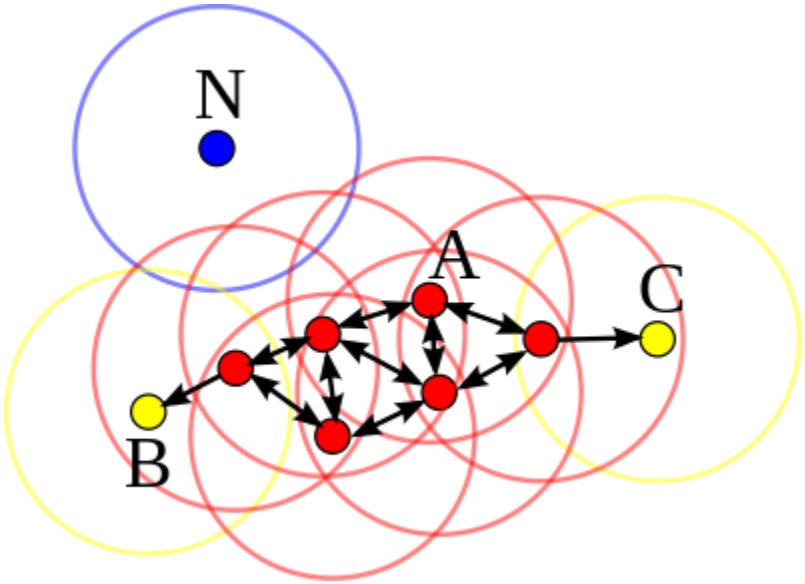
Streamlit + PyDeck (2 of 2)

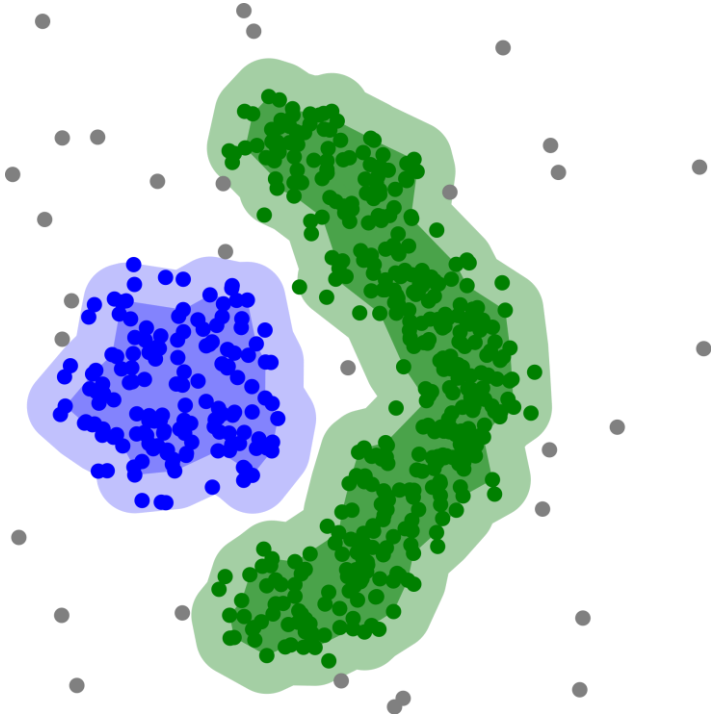
streamlit_pydeck_demo.py

```
# Main app
st.title('PyDeck Demo')

# Function to create map
def create_map(dataframe):
    if map_layer_type == "ScatterplotLayer":
        layer = pdk.Layer(
            "ScatterplotLayer",
            dataframe,
            get_position=["longitude", "latitude"],
            get_color=color,
            get_radius="exits_radius",
            radius_scale=radius_scale,
            opacity=opacity,
            pickable=True
        )
    elif map_layer_type == "HeatmapLayer":
        layer = pdk.Layer(
            "HeatmapLayer",
            dataframe,
            get_position=["longitude", "latitude"],
            get_weight="exits",
            opacity=opacity,
            pickable=True
        )

    view_state = pdk.ViewState(
        longitude=dataframe['longitude'].mean(),
        latitude=dataframe['latitude'].mean(),
        zoom=9
    )

    return pdk.Deck(layers=[layer], initial_view_state=view_state, map_style=map_style, tooltip={"text": "{name}\n{address}"})

# Display Map
st.write('### Map')
map = create_map(df)
st.pydeck_chart(map)

# Display data
st.dataframe(df)
```

Spatial Analysis Example: DBSCAN

- DBSCAN (Density-Based Spatial Clustering of Applications with Noise)
  - A popular density-based clustering algorithm
  - Given a set of points in some space, it groups together points that are closely packed together (points with many nearby neighbors) and marks them as clusters.
  - Points in low-density regions (whose nearest neighbors are too far away) are marked as noise.





https://en.wikipedia.org/wiki/DBSCAN

DBSCAN Algorithm

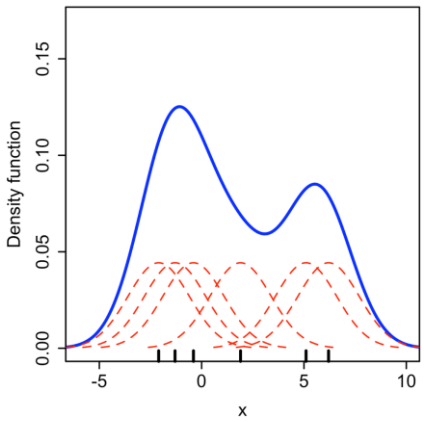
1. **Finding the ε (eps) Neighborhood:**
   - For each point in the dataset, DBSCAN starts by finding all the points within an ε (epsilon) distance from it. This ε neighborhood includes the point itself and any point within ε distance.
   - A core point is defined as a point that has at least minPts (minimum points) within its ε neighborhood. This minimum number of points includes the point itself. The idea here is that core points are those that are in dense regions of the data space. The values of ε and minPts are parameters that need to be specified before running the algorithm and are crucial for the resulting cluster structure.

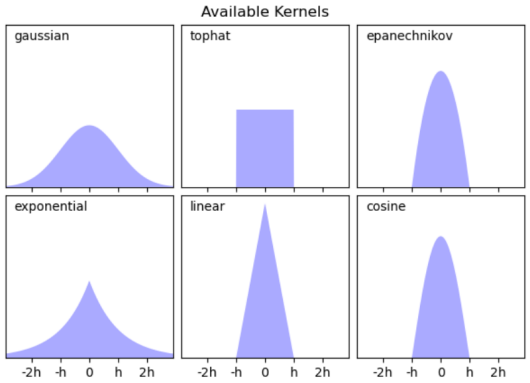
2. **Finding Connected Components:**
   - Once the core points have been identified, DBSCAN then proceeds to form clusters by connecting core points that are neighbors to each other (i.e., within the ε distance of each other).
   - This forms a graph where the core points are vertices, and an edge exists between two core points if they are within ε distance of each other. Connected components of this graph correspond to individual clusters.
   - Non-core points are not yet assigned to any clusters. They are either border points (close to a core point but not enough neighbors to be core themselves) or noise points.

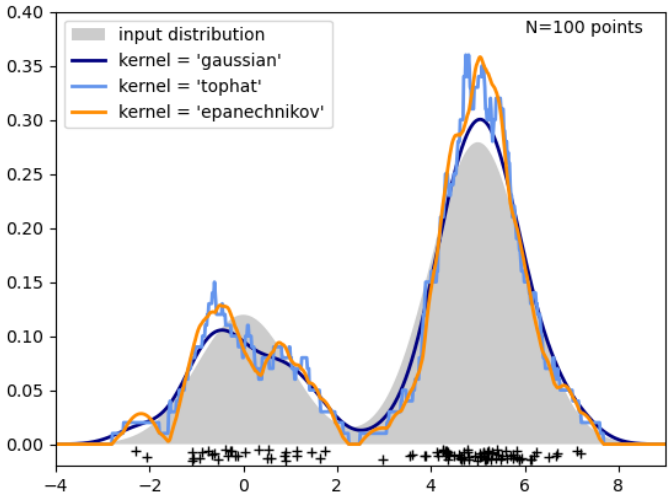
3. **Assigning Non-Core Points:**
   - After identifying the clusters based on connected core points, each non-core point is examined. If a non-core point is within ε distance of any core point of a cluster, it is assigned to that cluster as a border point.
   - If a non-core point is not within ε distance of any core point, it remains as noise, which means it does not belong to any cluster according to the density criteria set by ε and minPts.

Spatial Analysis Example: KDE

- KDE (Kernel Density Estimation)
  - A popular density estimation method
  - Given a set of points in some space, it estimates the probability density function by placing a smooth kernel (bump) at each data point and summing them together.
  - KDE creates a continuous density surface where higher values indicate regions with more concentrated data







https://en.wikipedia.org/wiki/Kernel_density_estimation

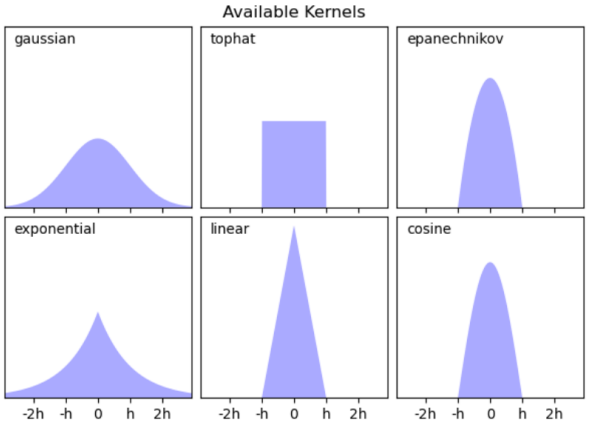
https://scikit-learn.org/stable/modules/density.html#kernel-density-estimation

KDE Algorithm

The Algorithm

1. **Place kernels** - Put a smooth, symmetric bump function (e.g., Gaussian) at each data point
2. **Sum kernels** - Add up all kernels to create a smooth density surface (higher = more concentrated)
3. **Normalize** - Ensure total area = 1 (makes it a proper probability density)

Kernel Functions



- **Gaussian** - smooth bell curve (most popular)
- **Epanechnikov** - parabolic, computationally efficient
- **Tophat** - uniform within bandwidth, zero outside
- **Exponential/Linear/Cosine** - other smooth options
- *Choice depends on:* desired smoothness, computational efficiency

Bandwidth Parameter

- **Small bandwidth:** Very detailed, may show noise, looks spiky
- **Large bandwidth:** Very smooth, may hide patterns, looks flat
- Practical use: Start with automatic selection, then adjust visually

Traffy Fondue Data

สำหรับผู้นำข้อมูลไปพัฒนาต่อยอด

สามารถขอข้อมูลในระบบ เพื่อนำไปจัดทำสถิติหรือวิเคราะห์ข้อมูล ผ่าน API (ข้อมูลจะอัปเดตทุกๆ 10 นาที)

หมายเหตุ : หากท่านได้มีการนำข้อมูลหรือผลการวิเคราะห์ไปเผยแพร่ต่อ ขอความกรุณา
– แจ้งรายละเอียดของท่าน จุดประสงค์การนำไปใช้งาน ลิงค์เผยแพร่ผลงาน เมื่อดาวน์โหลดข้อมูล csv
– ในส่วนผลงานของท่าน กรุณาใส่ลิงค์ traffy.in.th หรือหากเป็นรูปภาพ กรุณาใส่รูปโลโก้ของ Traffy Fondue ด้วย เพื่อเป็นการระบุแหล่งข้อมูลต้นทาง และเป็นการให้เครดิตทีมพัฒนา Traffy Fondue /ขอบคุณครับ

การจำกัดปริมาณข้อมูล
เนื่องจากข้อมูลมีปริมาณมาก จึงจำกัดจำนวนผลลัพธ์ไว้ ดังนี้
– หากไม่ระบุตัวตน (เช่น กดดาวน์โหลดโดยตรงจากเว็บ bangkok.traffy.in.th หรือไม่ได้ส่งค่าตัวแปร name, org, ฯลฯ มา) จะถูกจำกัด limit ไว้ไม่เกิน 1,000 รายการ
– หากระบุตัวตน (ส่งค่าตัวแปร name, org, ฯลฯ มา) จะถูกจำกัด limit ไว้ไม่เกิน 25,000 รายการ
– หากต้องการข้อมูลทั้งหมด สามารถดาวน์โหลดได้ที่ลิงค์ด้านล่างนี้ (ประมาณ 300MB) โดยข้อมูลจะอัปเดตทุกๆ 3:00 น. ของทุกวัน
https://publicapi.traffy.in.th/dump-csv-chadchart/bangkok_traffy.csv

Resource URL

JSON format *(อัปเดตทุกๆ 10 นาที)*
https://publicapi.traffy.in.th/teamchadchart-stat-api/geojson/v1?output_format=json

CSV format *(อัปเดตทุกๆ 10 นาที)*
https://publicapi.traffy.in.th/teamchadchart-stat-api/geojson/v1?output_format=csv

https://www.traffy.in.th/?page_id=27351

ข้อมูลล่าสุด (อัปเดตทุกๆ 10 นาที)

- หากไม่ระบุตัวตน limit ไม่เกิน 1000 รายการ
https://publicapi.traffy.in.th/teamchadchart-stat-api/geojson/v1?output_format=json

- หากระบุตัวตน limit ไม่เกิน 25,000 รายการ
https://publicapi.traffy.in.th/teamchadchart-stat-api/geojson/v1?output_format=csv&limit=2000&name=test&org=test&purpose=test&email=test@test.org

https://publicapi.traffy.in.th/teamchadchart-stat-api/geojson/v1?output_format=csv&name=a&org=a&purpose=a&email=a@a.com&limit=10000&text=น้ำท่วม&start=2023-05-01&end=2023-10-31

ข้อมูลทั้งหมด (อัปเดตทุกๆ 3:00 น. ของทุกวัน)
(ขนาดประมาณ 500MB เมื่อเดือน พ.ย. 2566)

https://publicapi.traffy.in.th/dump-csv-chadchart/bangkok_traffy.csv

Traffy Fondue: Flood

https://publicapi.traffy.in.th/teamchadchart-stat-api/geojson/v1?output_format=csv&name=a&org=a&purpose=a&email=a@a.com&limit=10000&text=**น้ำท่วม**&start=2023-05-01&end=2023-10-31

traffy_flood.csv

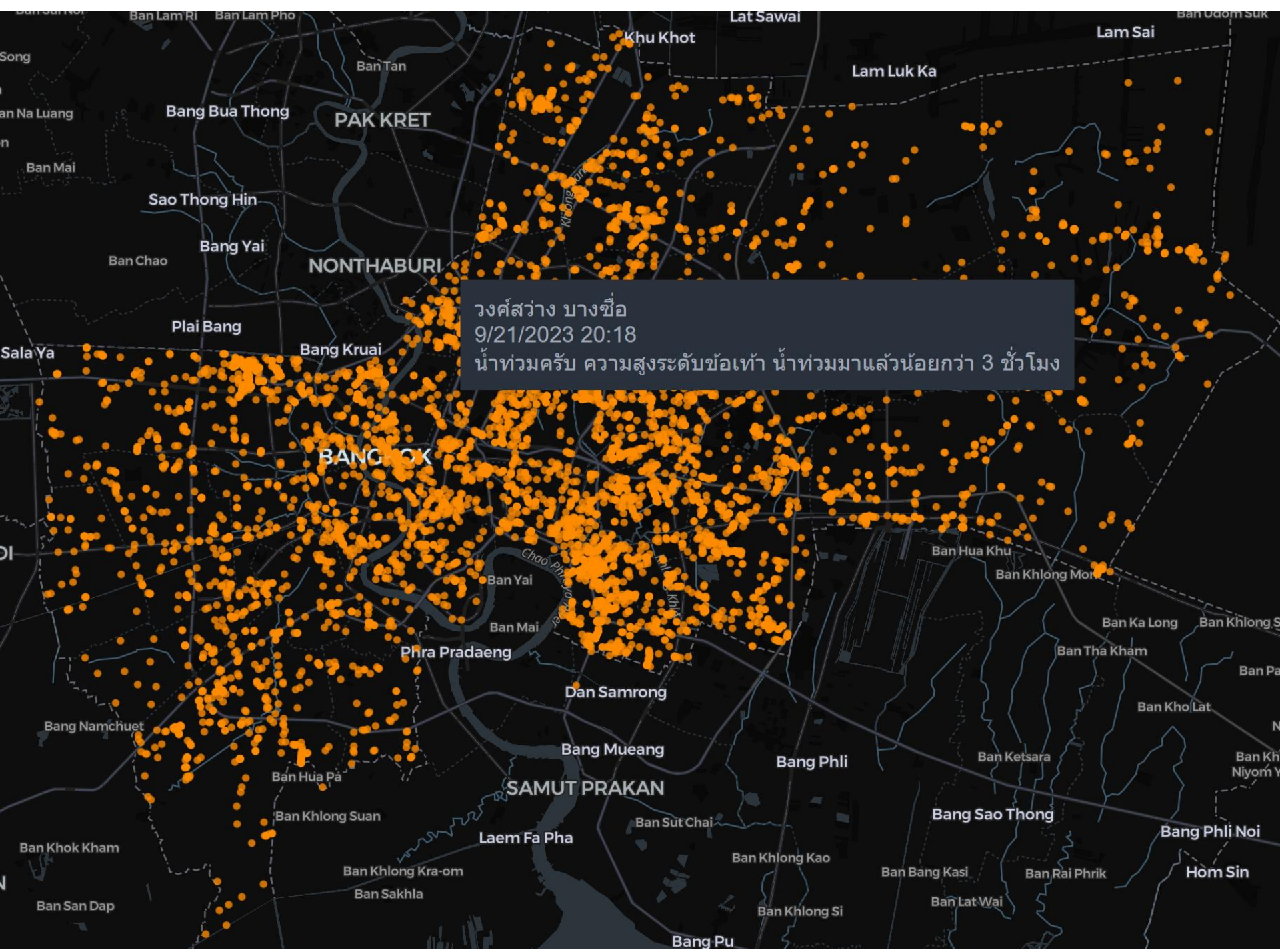
ticket_id	type	organization	organization_ac	comment	coords	photo	photo_after	address	subdistrict	district	province	timestamp
2024-B2ZLKK	น้ำท่วม	กรุงเทพมหานคร,	เขตจอมทอง, ฝ่า	สูบน้ำคลองมาใช้เล่นสาดน้ำ	100.46373,13.69901	https://storage.google		89/3 ถ. เอก	บางขุนเทีย	จอมทอง	กรุงเทพมห	4/14/2024 21:12
2024-FTC3WN	น้ำท่วม	กรุงเทพมหานคร,	เขตคลองสามวา	น้ำขังจากข้างบ้าน ความสูงร	100.73738,13.90355	https://storage.google		98 ซอย นิมิ	สามวาตะวัน	คลองสามว	กรุงเทพมห	4/14/2024 9:22
7C7B97	น้ำท่วม	ร้องทุกข์ กทม. 15	ฝ่ายโยธา เขตธ	พบน้ำท่วมขังเนื่องจากน้ำ	100.4877,13.72751	https://stor	https://stor	PFGQ+X35	หิรัญรูจี	ธนบุรี	กรุงเทพมห	4/14/2024 8:57
2024-AYETBW	ถนน	พรรคก้าวไกล ดอ	ฝ่ายโยธา เขตด	ซ.สรณคมน์ 25 แยก2-5 ไม่มี	100.59554,13.93467	https://storage.google		1606 ซอย	ลีกัน	ดอนเมือง	กรุงเทพมห	4/13/2024 19:27
2024-AWTL8U	น้ำท่วม	กรุงเทพมหานคร,	กองระบบคลอง	จัดคนมาขุดลอกคลองเลียบ	100.59076,13.93169	https://storage.google		310/1333 ซ	ลีกัน	ดอนเมือง	กรุงเทพมห	4/13/2024 12:26
2024-4HQ6W9	น้ำท่วม	กรุงเทพมหานคร,	เขตบางซื่อ, ฝ่าย	เพื่อนบ้านติดรางน้ำ ตรงรอย	100.51005,13.82232	https://storage.google		1564/4 ถ. ท่	วงศ์สว่าง	บางซื่อ	กรุงเทพมห	4/13/2024 7:17
UH2W2K	น้ำท่วม	ร้องทุกข์ กทม. 15	ฝ่ายโยธา เขตย	ปัญหา: ภายในซอยแยก	100.5323,13.70121	https://storage.google		215/29 ถน	ช่องนนทรี	ยานนาวา	กรุงเทพมห	4/12/2024 13:37
2024-C6D7LN	ทางเท้า	กรุงเทพมหานคร,	ฝ่ายโยธา เขตบ	เส้นทางเดินเท้า หลังจากมีก	100.36757,13.73007	https://storage.google		86 ซ. อัสสั	บางไผ่	บางแค	กรุงเทพมห	4/12/2024 7:55
2024-6X94YU	น้ำท่วม	กรุงเทพมหานคร,	เขตบางซื่อ, ฝ่าย	ขอทราบผลการดำเนินการเรื่	100.53244,13.82019	https://storage.google		229/5 ซอย	บางซื่อ	บางซื่อ	กรุงเทพมห	4/11/2024 22:49
H3N3T2	น้ำท่วม	ร้องทุกข์ กทม. 15	เขตบางพลัด, กร	มีน้ำท่วมสูงประมาณ 25	100.49159,13.76554	https://stor	https://stor	264 ถ. อรุณ	บางยี่ขัน	บางพลัด	กรุงเทพมห	4/11/2024 22:25
2024-33F7AK	น้ำท่วม	กรุงเทพมหานคร,	ฝ่ายโยธา เขตวั	แจ้งร้องเรียนครับ อยากให้ท	100.6143,13.77133	https://storage.google		94 ซอยลา	พลับพลา	วังทองหลา	กรุงเทพมห	4/11/2024 18:47
2024-A29D8H	น้ำท่วม	ก้าวไกลคลองสาม	ฝ่ายโยธา เขตค	น้ำท่วมขังมาเป็นระยะเวลาน	100.69867,13.84137	https://storage.google		RMRX+HFM	บางชัน	คลองสามว	กรุงเทพมห	4/11/2024 16:49
2024-479QTP	น้ำท่วม	กรุงเทพมหานคร,	เขตจตุจักร, ฝ่าย	ในซอยวิภาวดีรังสิต 20 เข้าซ	100.56203,13.80325	https://storage.google		RH36+8V4	จอมพล	จตุจักร	กรุงเทพมห	4/11/2024 13:31
2024-7DBX9L	น้ำท่วม	กรุงเทพมหานคร,	เขตบางกอกน้อย	ขอความอนุเคราะห์ เขตบาง	100.47663,13.75166	https://stor	https://stor	15/7 ซ. อิส	บ้านช่างหล่	บางกอกน้อ	กรุงเทพมห	4/11/2024 8:13
YTN9VM	น้ำท่วม	ร้องทุกข์ กทม. 15	เขตบางกอกน้อย	เครื่องสูบน้ำแต่ไม่มีการ	100.47712,13.77371	https://stor	https://stor	เลขที่ 197	อรุณอมรินท	บางกอกน้อ	กรุงเทพมห	4/11/2024 6:41
2024-CBQHYP	จุดเสี่ยง,คว	เพื่อนชัชชาติ, เข	เขตสวนหลวง, ส	บริเวณนี้ทุกเช้า จะมีน้ำเน่าท	100.63015,13.73615	https://storage.google		PJPH+CW6	สวนหลวง	สวนหลวง	กรุงเทพมห	4/10/2024 22:49
2024-D3MGDG	ความสะอา	กรุงเทพมหานคร,	ฝ่ายสิ่งแวดล้อม	ห้องน้ำห้องส้วมแคมป์คนงาน	100.3361,13.785	https://stor	https://stor	18/19 ซอย	ศาลาธรรมส	ทวีวัฒนา	กรุงเทพมห	4/10/2024 18:07
2024-DU23ML	น้ำท่วม	กรุงเทพมหานคร,	ฝ่ายโยธา เขตดุ	อยากให้เจ้าหน้าที่ช่วยเข้ามา	100.52588,13.78975	https://stor	https://stor	158/2 ซอย	ถนนนครไช	ดุสิต	กรุงเทพมห	4/10/2024 17:14
7HTCKN	เสนอแนะ,บ	ร้องทุกข์ กทม. 15	เขตพระนคร, กร	พื้นผิวถนนทั้ง 50 เขต ของ	100.50182,13.75391	https://storage.google		173 ถนน ดิ	เสาชิงช้า	พระนคร	กรุงเทพมห	4/10/2024 15:20

Split this column into longitude and latitude

Traffy Fondue: Flood: Scatter Plot

traffy_flood.ipynb

```
# Define a layer to display on a map
layer = pdk.Layer(
    "ScatterplotLayer",
    df,
    get_position=["longitude", "latitude"],
    get_radius=200,
    get_fill_color=[255, 140, 0],
    opacity=0.6,
    pickable=True
)

# Set the viewport location
view_state = pdk.data_utils.compute_view(df[["longitude", "latitude"]])
view_state.zoom = 10

# Render
deck = pdk.Deck(layers=[layer], initial_view_state=view_state,
                tooltip={"text": "{subdistrict} {district}\n{timestamp}\n{comment}"})
deck.to_html("pydeck_traffy.html")
```

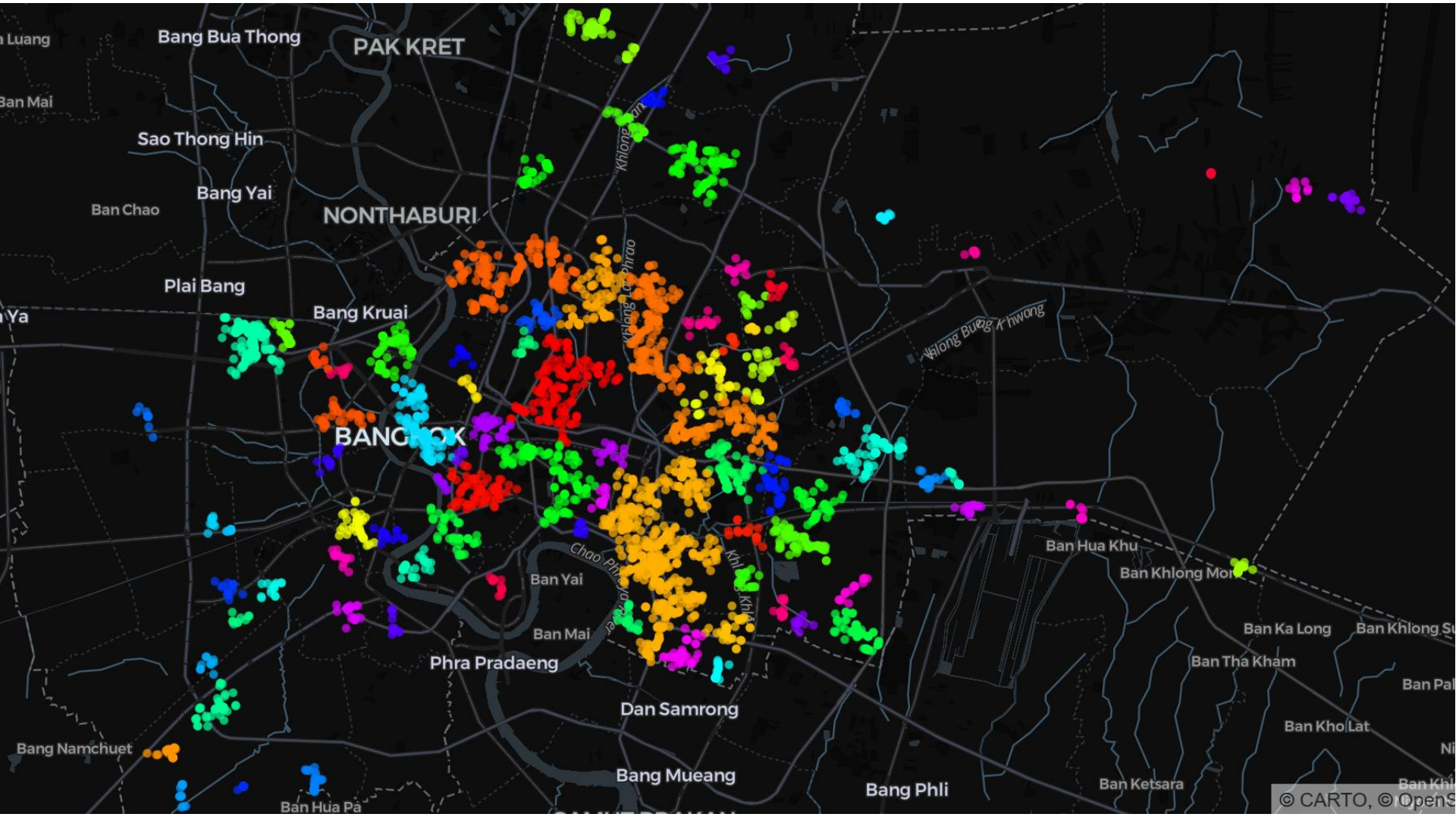

วงศ์สว่าง บางซื่อ
9/21/2023 20:18
น้ำท่วมครับ ความสูงระดับข้อเท้า น้ำท่วมมาแล้วน้อยกว่า 3 ชั่วโมง
Lat Sawai
Khu Khot
Lam Sai
Lam Luk Ka
Ban Tan
Bang Bua Thong
PAK KRET
Ban Mai
Sao Thong Hin
Bang Yai
Ban Chao
NONTHABURI
Plai Bang
Sala Ya
Bang Kruai
BANGKOK
Ban Yai
Ban Mai
Phra Pradaeng
Dan Samrong
Ban Hua Khu
Ban Khlong Mon
Ban Ka Long
Ban Tha Kham
Ban Kho Lat
Bang Namchuet
Bang Mueang
Bang Phli
Ban Ketsara
Ban Hua Pa
SAMUT PRAKAN
Ban Khlong Suan
Ban Sut Chai
Bang Sao Thong
Bang Phli Noi
Laem Fa Pha
Ban Khok Kham
Ban Khlong Kao
Ban Bang Kasi
Ban Rai Phrik
Hom Sin
Ban Khlong Kra-om
Ban Sakhla
Ban San Dap
Ban Lat Wai
Ban Khlong Si
Bang Pu

Traffy Fondue: Flood: DBSCAN

traffy_flood.ipynb

```
# DBSCAN clustering
from sklearn.cluster import DBSCAN

coords = df[['latitude', 'longitude']]
db = DBSCAN(eps=0.005, min_samples=10).fit(coords)
df['cluster'] = db.labels_

# Filter out noise points
df = df[df['cluster'] != -1]

# Count the number of points in each cluster
clusters_count = df['cluster'].value_counts()

# Exclude the '-1' cluster, which represents noise
clusters_count = clusters_count[clusters_count.index != -1]

unique_clusters = df['cluster'].unique()
num_clusters = len(unique_clusters)

# Use a continuous colormap to generate colors, ensure we have enough colors for all clusters.
colormap = plt.get_cmap('hsv')
cluster_colors = {cluster: [int(x*255) for x in colormap(i/num_clusters)[:3]]
                      for i, cluster in enumerate(unique_clusters)}

# Map cluster ID to color for each row in the dataframe
df['color'] = df['cluster'].map(cluster_colors)

# Define the scatter plot layer
scatter_layer = pdk.Layer(
    "ScatterplotLayer",
    df,
    get_position="[longitude, latitude]",
    get_color='color',
    get_radius=200,
    opacity=0.5,
    pickable=True
)

view_state = pdk.ViewState(
    latitude=df['latitude'].mean(),
    longitude=df['longitude'].mean(),
    zoom=10
)
pdk.Deck(layers=[scatter_layer], initial_view_state=view_state, tooltip={"text": "{cluster}\n{subdistrict}
{district}\n{timestamp}"})
```

Traffy Fondue: Flood: DBSCAN

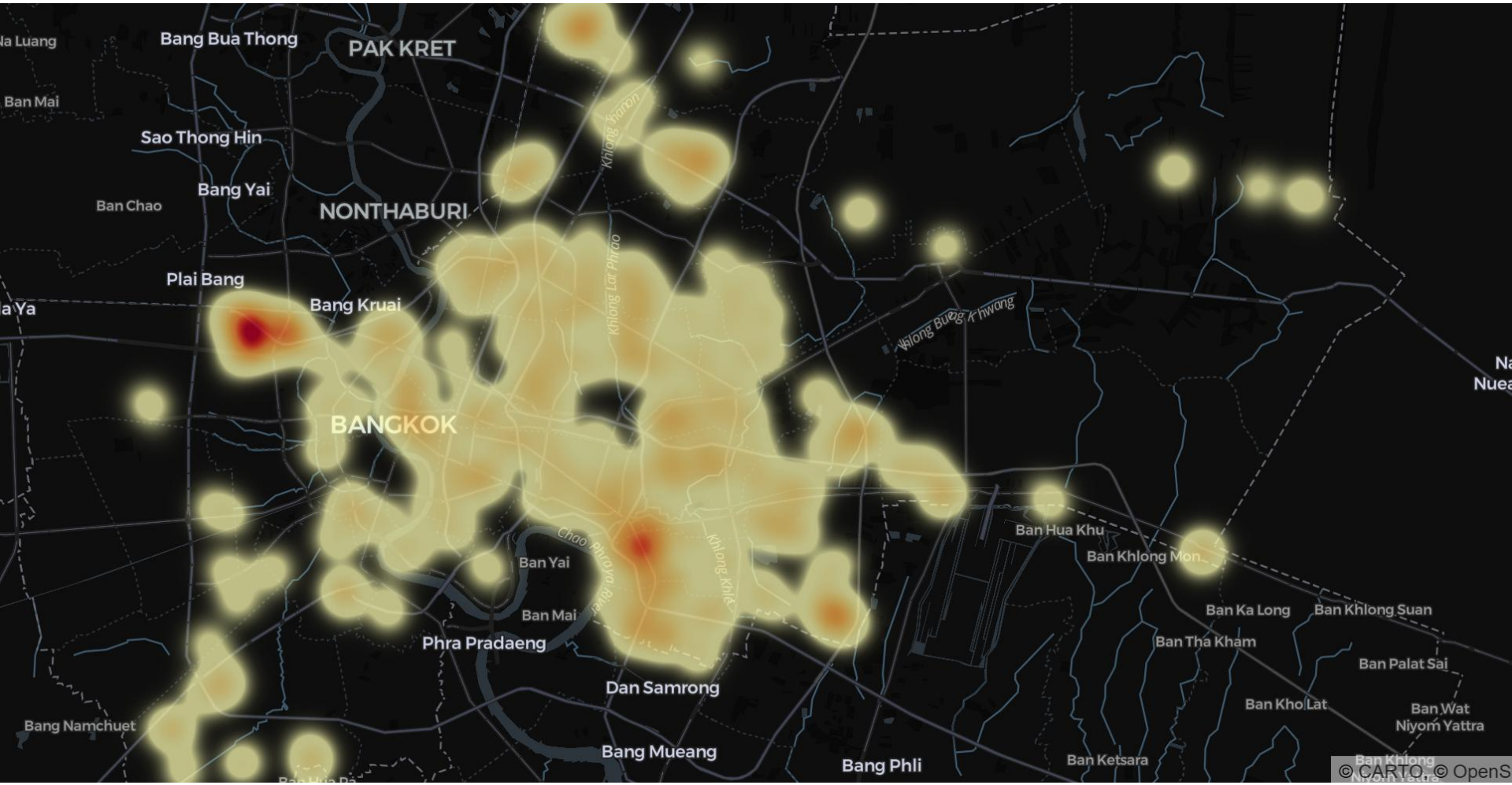


Traffy Fondue: Flood: DBSCAN

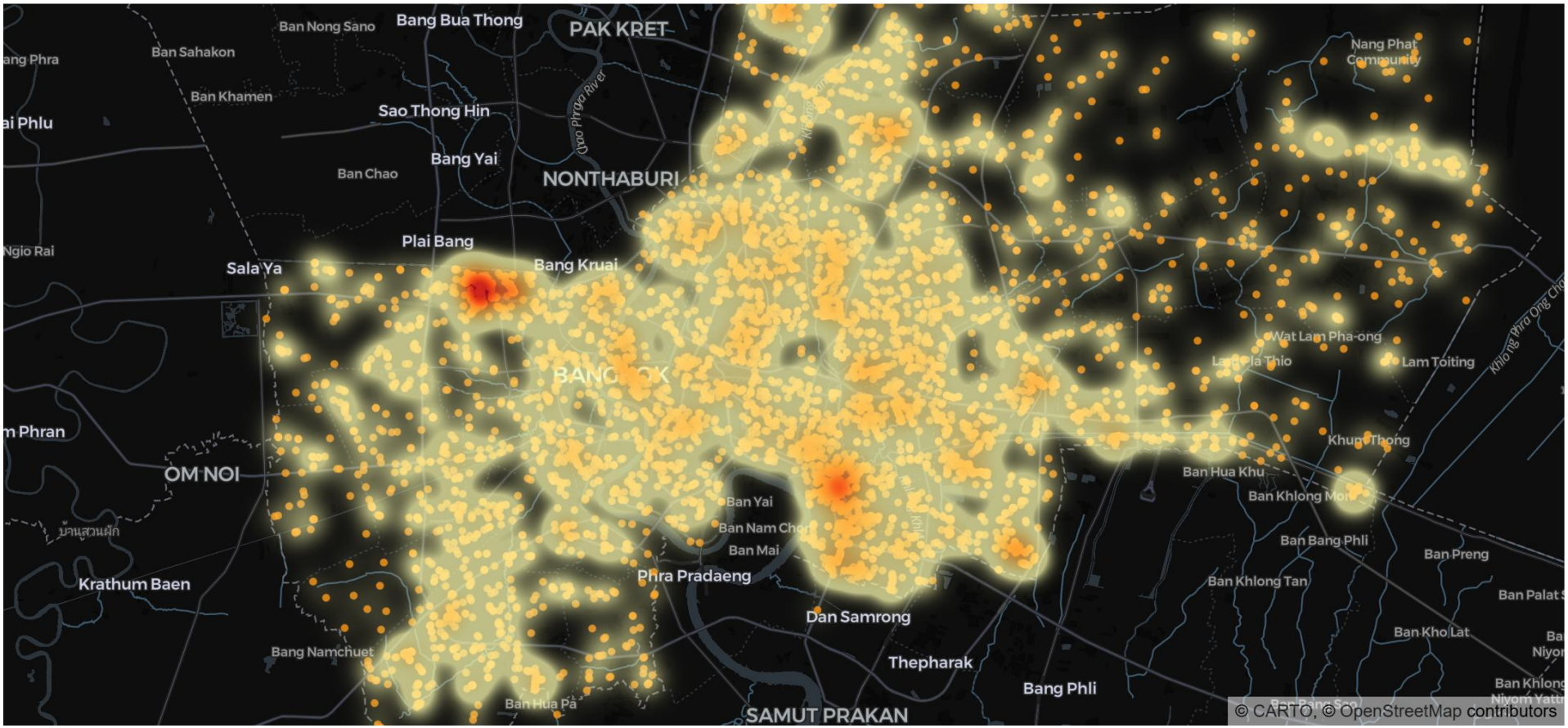
```
# Define the heatmap layer
heatmap_layer = pdk.Layer(
    "HeatmapLayer",
    df,
    get_position="[longitude, latitude]",
    opacity=0.5,
    pickable=True
)

view_state = pdk.ViewState(
    latitude=df['latitude'].mean(),
    longitude=df['longitude'].mean(),
    zoom=10
)
pdk.Deck(layers=[heatmap_layer], initial_view_state=view_state)
```

traffy_flood.ipynb

Traffy Fondue: Flood: DBSCAN



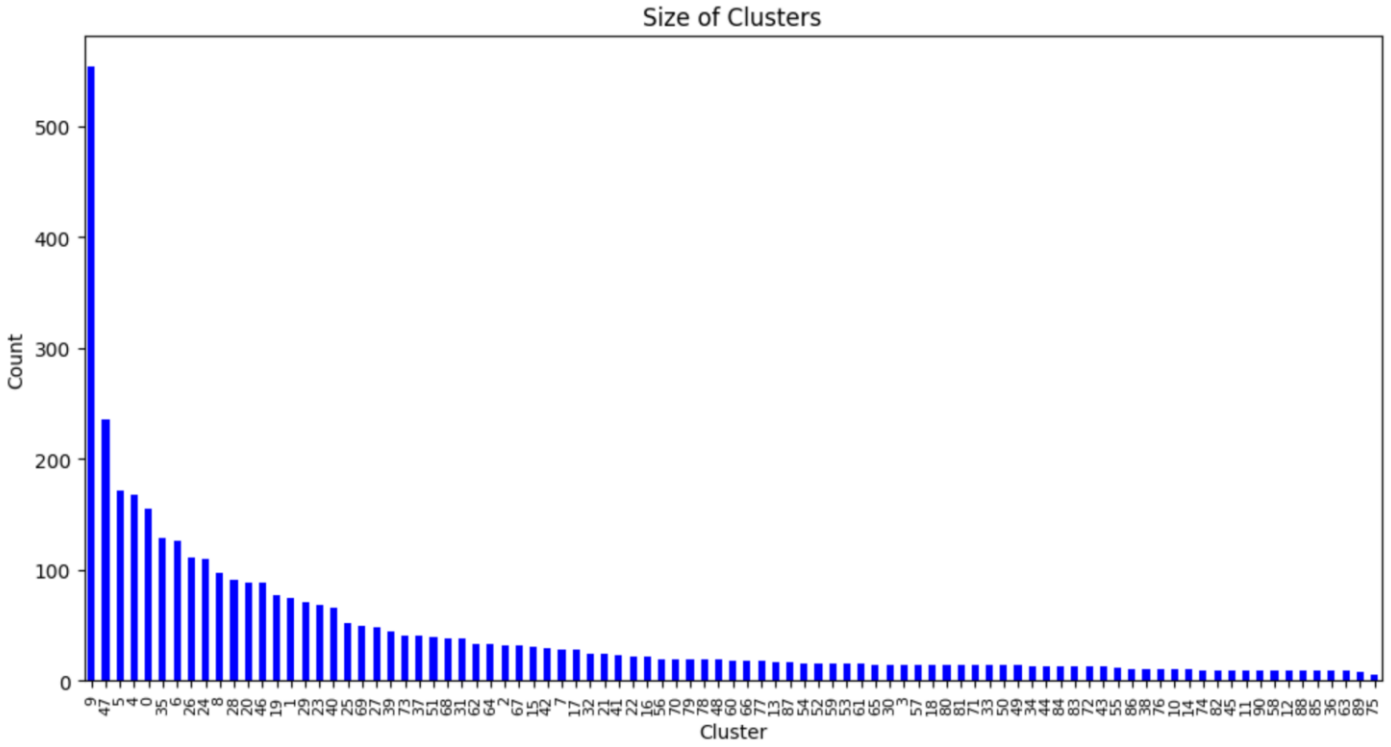
Traffy Fondue: Flood: DBSCAN



Traffy Fondue: Flood: DBSCAN

```
import matplotlib.pyplot as plt

# Plotting the data
clusters_count.plot(kind='bar', color='blue')  # You can customize the color

plt.xticks(fontsize=8)

# Optional: adjust figure size if labels still overlap
plt.gcf().set_size_inches(12, 6)  # Adjust the size as needed

plt.xlabel('Cluster')  # Set x-axis label, if needed
plt.ylabel('Count')    # Set y-axis label
plt.title('Size of Clusters')  # Set title
plt.show()
```

traffy_flood.ipynb



Traffy Fondue: Flood: KDE

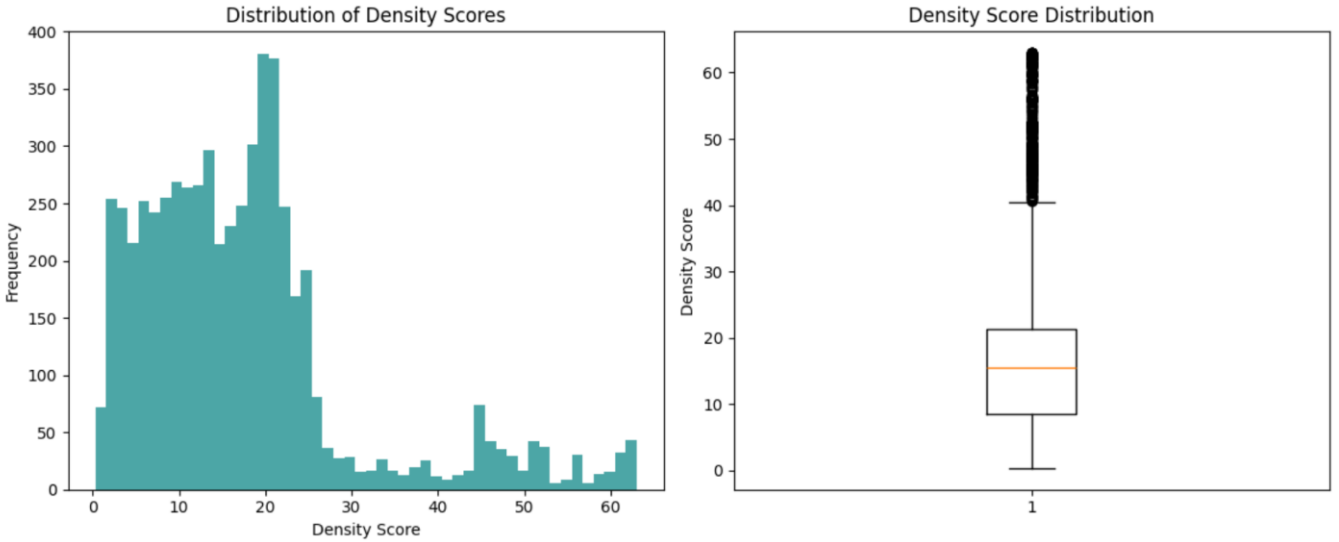
traffy_flood.ipynb

```
# Kernel Density Estimation Analysis
from sklearn.neighbors import KernelDensity

df = pd.read_csv("traffy_flood.csv")
coords = df[['latitude', 'longitude']]

# Fit KDE model
# Bandwidth controls smoothing - smaller = more detailed, larger = smoother
kde = KernelDensity(bandwidth=0.01, kernel='gaussian')
kde.fit(coords)

# Calculate density score for each point
log_density = kde.score_samples(coords)
density = np.exp(log_density)

# Add density scores to dataframe
df['density'] = density

# Normalize density for better visualization (0-1 range)
df['density_normalized'] = (df['density'] - df['density'].min()) / (df['density'].max() - df['density'].min())
```

Traffy Fondue: Flood: KDE

```
# Visualize density distribution
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 5))

# Histogram of density values
plt.subplot(1, 2, 1)
plt.hist(df['density'], bins=50, color='teal', alpha=0.7)
plt.xlabel('Density Score')
plt.ylabel('Frequency')
plt.title('Distribution of Density Scores')

# Box plot
plt.subplot(1, 2, 2)
plt.boxplot(df['density'], vert=True)
plt.ylabel('Density Score')
plt.title('Density Score Distribution')

plt.tight_layout()
plt.show()
```

traffy_flood.ipynb



Traffy Fondue: Flood: KDE

traffy_flood.ipynb

```
# Create color mapping based on density (red = high density, blue = low density)
def density_to_color(density_normalized):
    """Convert normalized density to RGB color (blue to red gradient)"""
    # Blue (low) to Red (high) through green
    r = int(density_normalized * 255)
    g = int((1 - abs(2 * density_normalized - 1)) * 255)
    b = int((1 - density_normalized) * 255)
    return [r, g, b]

# Apply color mapping
df['density_color'] = df['density_normalized'].apply(density_to_color)

# Define the scatter plot layer with density-based colors
kde_scatter_layer = pdk.Layer(
    "ScatterplotLayer",
    df,
    get_position="[longitude, latitude]",
    get_color='density_color',
    get_radius=200,
    opacity=0.6,
    pickable=True
)

view_state = pdk.ViewState(
    latitude=df['latitude'].mean(),
    longitude=df['longitude'].mean(),
    zoom=10
)

pdk.Deck(
    layers=[kde_scatter_layer],
    initial_view_state=view_state,
    tooltip={
        "text": "Density: {density}\n{subdistrict} {district}\n{timestamp}"
    }
)
```

Traffy Fondue: Flood: KDE