Packages

```
import streamlit as st
import pandas as pd
import plotly.graph_objects as go
import pydeck as pdk
import numpy as np
import os
```

Basic Streamlit

Load data with cache

```
@st.cache_data
def load_data(url):
    df = pd.read_csv(url)
    df['date'] = pd.to_datetime(df['date'])
    df['date'] = df['date'].apply(lambda x: x.strftime('%d %b %y'))  # Change date
format
    return df

df = load_data('./RainDaily_Tabular.csv')
```

Text

```
st.title('Streamlit Rain Daily Tabular')
st.header('ปริมาณน้ำฝนรายวันของประเทศไทย')
st.subheader('ข้อมูลของจังหวัด', province)
st.write('ปริมาณน้ำฝนของพื้นที่', area, 'ในจังหวัด', province)
```

Sidebar

```
st.sidebar.header('Sidebar')
province = st.sidebar.selectbox(
    'จังหวัด',
    df['province'].unique()
)
date = st.sidebar.date_input(
    'วันที่',
    min_value=pd.to_datetime(df['date']).dt.date.min(),
    max_value=pd.to_datetime(df['date']).dt.date.max(),
    value=pd.to_datetime(df['date']).dt.date.min()
)
```

```
area = st.sidebar.selectbox(
    'พื้นที่',
    df[df['province'] == province]['name'].unique()
)

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",
    ]
)
```

Column

```
col1, col2 = st.columns(2)

province_data = df[df['province'] == province]
with col1:
    st.write('ปริมาณน้ำฝนในวันที่', date.strftime('%d/%m/%Y'))
    st.bar_chart(province_data[province_data['date'] ==
str(date)].set_index('name')['rain'])


with col2:
    st.write('ปริมาณน้ำฝนเฉลี่ย')
    st.bar_chart(province_data.groupby('name')['rain'].mean())
```

Checkbox to show/hide

```
with st.container():
    if st.checkbox('Show raw data', key='raw_data'):
        st.dataframe(df.reset_index(drop=True))
```

Expander

```
st.header("Code")
with st.expander("See code"):
    # Get the path of the current script
    file_path = os.path.realpath(__file__)
    st.code(open(file_path, 'r', encoding='utf-8').read())
```

Graph (Streamlit or Ploty)

Bar Chart

Streamlit

```
st.bar_chart(data=None, *, x=None, y=None, color=None, width=0, height=0,
use_container_width=True)
```

If x = None, uses the data index for the x-axis.

```
st.bar_chart(data=df, x='date', y='rain')
st.bar_chart(date_data_df.groupby('province')['rain'].mean())
```

Plotly

```
fig = go.Figure()

fig.add_traces( go.Bar(name='Rain', y=avg_rain["amphoe"], x=avg_rain["rain"],
                orientation='h'))

st.plotly_chart(fig, config={'staticPlot': True})
```

Line Chart

Streamlit

```
st.line_chart(data=None, *, x=None, y=None, color=None, width=0, height=0,
use_container_width=True)
```

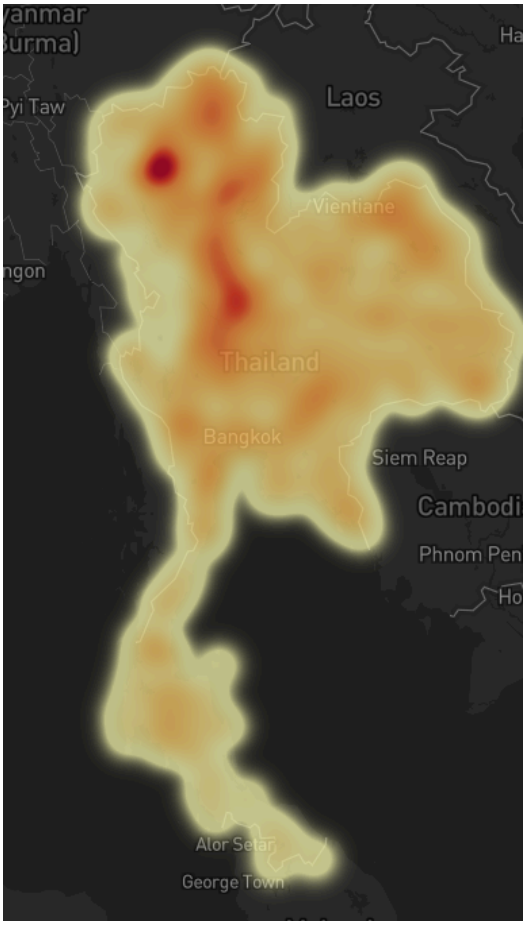
```
st.line_chart(province_data.groupby('date')['rain'].mean())
```

Plotly

```
fig = go.Figure()
avg_rain_all = df.groupby('date')['rain'].mean().reset_index()
fig.add_trace(go.Scatter(x=avg_rain_all['date'],
           y=avg_rain_all['rain'], mode='lines', name='All Provinces'))
st.plotly_chart(fig, config={'staticPlot': True})
```

Pydeck

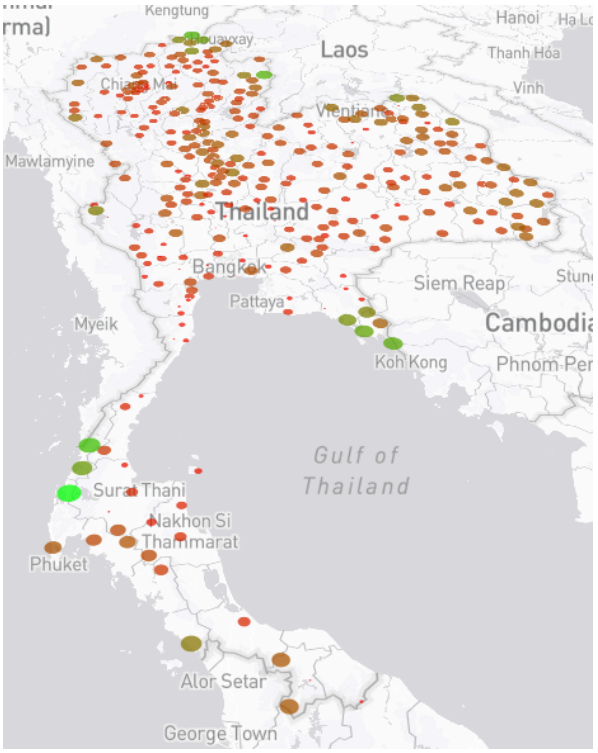
Heatmap (Nont's)




```
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",
    ]
)

def create_map(dataframe):

    layer = pdk.Layer(
        "HeatmapLayer",
        dataframe,
        get_position=["longitude", "latitude"],
        #get_weight="exits",
        opacity=0.5,
        pickable=True
    )
```

```
    view_state = pdk.ViewState(
        longitude=dataframe['longitude'].mean(),
        latitude=dataframe['latitude'].mean(),
        zoom=1
    )

    return pdk.Deck(layers=[layer], initial_view_state=view_state,
map_style=map_style)
```

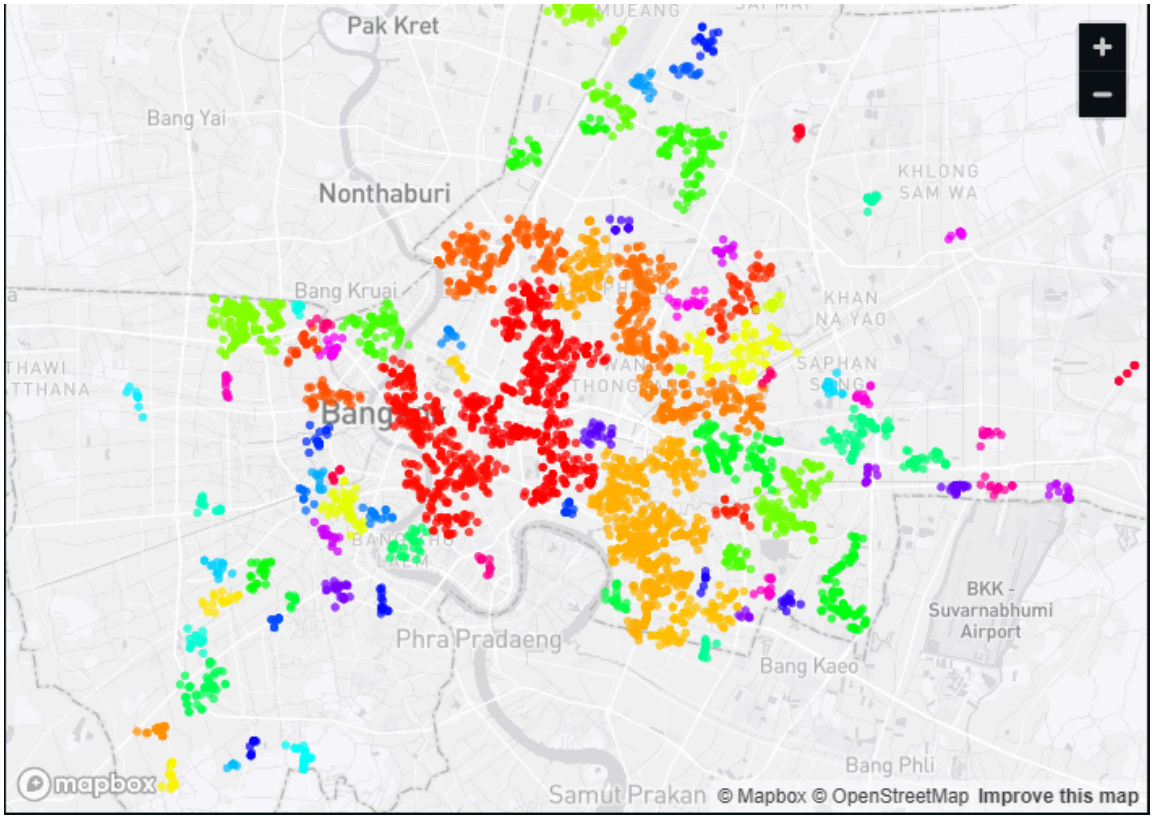
Circle (Tee's)




```
avg_rain = df.groupby(['province', 'latitude',
                       'longitude'])['rain'].mean().reset_index()

view_state = pdk.ViewState(
        latitude=avg_rain['latitude'].mean(),
        longitude=avg_rain['longitude'].mean(),
        zoom=5,
        pitch=50,
    )

    # Define a color for provinces
    def color_gradient(val, min_val, max_val):
        # Normalize val
        normalized_val = (val - min_val) / (max_val - min_val)
        return [int(255 * (1 - normalized_val)), int(255 * normalized_val), 0]  #
RGB color
```

```
    min_rain = avg_rain['rain'].min()
    max_rain = avg_rain['rain'].max()

    avg_rain['color'] = avg_rain['rain'].apply(lambda x: color_gradient(x,
min_rain, max_rain))

    avg_rain['radius'] = np.log(avg_rain['rain'] + 1) * 5e3

    layer = pdk.Layer(
        'ScatterplotLayer',
        data=avg_rain,
        get_position='[longitude, latitude]',
        get_radius='radius',
        get_fill_color='color',
        opacity=0.5,
        pickable=True
    )

    r = pdk.Deck(
        map_style='mapbox://styles/mapbox/light-v11',
        initial_view_state=view_state,
        layers=[layer],
    )

    st.pydeck_chart(r)
```

DBSCAN




```python
import streamlit as st
import pandas as pd
import pydeck as pdk
import matplotlib.pyplot as plt

# DBSCAN clustering
from sklearn.cluster import DBSCAN

@st.cache_data
def load_data(url):
    df = pd.read_csv(url)
    return df

df = load_data('flood.csv')

# split columns "coords" into two columns "latitude" and "longitude" in float
df[['longitude', 'latitude']]=  df['coords'].str.split(',', expand=True)
df['longitude'] = df['longitude'].astype(float)
df['latitude'] = df['latitude'].astype(float)

# DBSCAN clustering
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",
    data=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
)

r = pdk.Deck(
    map_style='mapbox://styles/mapbox/light-v11',
    initial_view_state=view_state,
    layers=[scatter_layer],
    # tooltip={"text": "{cluster}\n{subdistrict} {district}\n{timestamp}"}
)

st.pydeck_chart(r)
```