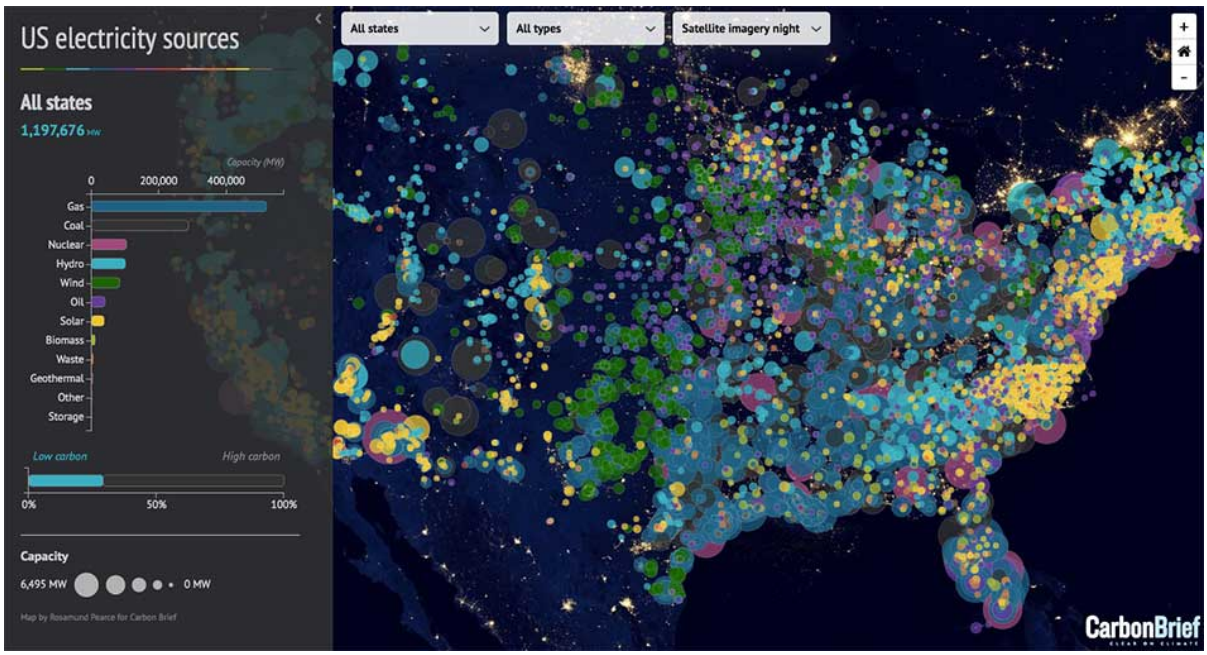
Geospatial Visualization Technique Folium Map with GeoJSON

Geospatial Visualization Technique



A narrative description of the visualization

Geospatial visualization is the use of geospatial visualization tools and technologies to analyze geospatial data. Geospatial data visualization allows adding interactive visualization to traditional maps; This helps to analyze the different factors of geographical area and location simultaneously, Explore the different layers of the map, and experiment with the map's appearance.

The statement by Alberto Cairo in his book The Functional Art: An Introduction to Information Graphics and Visualization eloquently expresses the points above: "Graphics should not simplify messages. They should clarify them, highlight trends, uncover patterns, and reveal realities not visible before." Using maps instead of other charting forms allows us to highlight trends, uncover patterns, and reveal realities not visible before when it comes to spatial data. It also helps us in gaining clarity about the data, more than just simplifying the data itself.

In summary, there are five key benefits of Geospatial visualization (IOT Agenda,2020)

1. Greater ability to build geospatial applications and dashboards
2. Enables better decision making
3. Improves understanding and relevance
4. Increases operational efficiency
5. Provides visible indicators of abnormalities

Which circumstances this visualization should and should not be used

When to consider Geospatial visualization: -When the data contains location information, we want the user to understand the data from a glance.

Limitation of Geospatial visualization(Map) technique: -.Of course, you must-have geographical data (Latitude/ Longitude) ,or JSON file that contains the geographical data. You can't use map visualization if you don't halve location information. -. It may also not be appropriate when there are many dimensions of data. For example, if we want to plot 2 dimensions into the map, the first is the province temperature and the second is the province humidity, both in one province boundary. This is somewhat difficult or not appropriate to visualize because you'll need some adjustment or a color combination in your map boundary -.Some libraries can provide an advanced feature or add more layers, but sometimes, putting too much data on the map can create graphic junks. or look messy

It would be best to avoid the Geospatial visualization: Avoid using a map is not easy to tell the trend or the portion. Or process - Dealing with detailed statistical data, we may think about using other topics.

The substitute form of Geospatial visualization can consider a table form with the location names, but it will be hard to understand vs. the map.

Visualization Library :Folium

about Folium

Folium is the library that I am going to use primarily to plot and draw a map. The library is also sound at in visualizing data on interactive maps. Folium is an open-source python project built along with Leaflet.js Maps.

folium builds on the data wrangling strengths of the Python ecosystem and the mapping strengths of the leaflet.js library. Manipulate your data in Python, then visualize it in on a Leaflet map via folium.

Concepts folium makes it easy to visualize data that’s been manipulated in Python on an interactive leaflet map. It enables both the binding of data to a map for choropleth visualizations as well as passing rich vector/raster/HTML visualizations as markers on the map.

The library has a number of built-in tilesets from OpenStreetMap, Mapbox, and Stamen, and supports custom tilesets with Mapbox or Cloudmade API keys. folium supports both Image, Video, GeoJSON and TopoJSON overlays.

The full detail of Folium documentation can see at .https://python-visualization.github.io/folium/

Folium can be integrate with Jupyter notebook , just installed using pip or anaconda.

```
pip install folium
```

```
conda install -c conda-forge folium
```

let's see how it works ,thank you a article of Thank to the article of "Creating a Simple Map with Folium and Python", towarddatascince . you can promptly understand..

Folium General approach and limitations of this library

It is declarative tool , hand-on , it's easy to understand , less time for interpret the result , creating right away the output . not come with complicating coding .

It also can integrate well with Jupyter

There are severval Geospatial or mapping libraries for you to choose, you can see a brief comparison form https://gisgeography.com/python-libraries-gis-mapping. I choose Folium for this documentation because it's very hand-on , Super easy to use, and it comewith Choropleth method that can easily create a map boudary. with color encoding .and It also support functin of building interactive web maps

However, this library has some limitations. It has some problems dealing with a large number of data points, the library sometimes freeze or hang. Some of its functions require GeoJSON or Shapre file to work. I also find that it's not easy to encode multidimensional data or use another column to encode, for example, if you want to encode the "shape" or size of an object overlay to the map, it's quite complicated to do so. the printing of the Folium may requires adding extention

Let's see how easy it's

In [1]:
```
import folium
import pandas as pd
```

In [2]:
```
location = "https://data.smartdublin.ie/dataset/33ec9fe2-4957-4e9a-ab55-c5e91
bike_station_locations = pd.read_csv(location)
bike_station_locations.head(5)
```

Out[2]:

	Number	Name	Address	Latitude	Longitude
0	42	SMITHFIELD NORTH	Smithfield North	53.349562	-6.278198
1	30	PARNELL SQUARE NORTH	Parnell Square North	53.353462	-6.265305
2	54	CLONMEL STREET	Clonmel Street	53.336021	-6.262980
3	108	AVONDALE ROAD	Avondale Road	53.359405	-6.276142
4	56	MOUNT STREET LOWER	Mount Street Lower	53.337960	-6.241530

keep the Latitude, Longitude and the Name of the location. The former two columns will allow me to map the locations and the latter I will use to give each location pin a name:

In [3]:
```
map = folium.Map(location=[bike_station_locations.Latitude.mean(), bike_stati
```

For the map, the first step is to create a map of the location I want. Using the location parameter, I pass in the mean of the latitude and longitude coordinates I have to centre the map there.

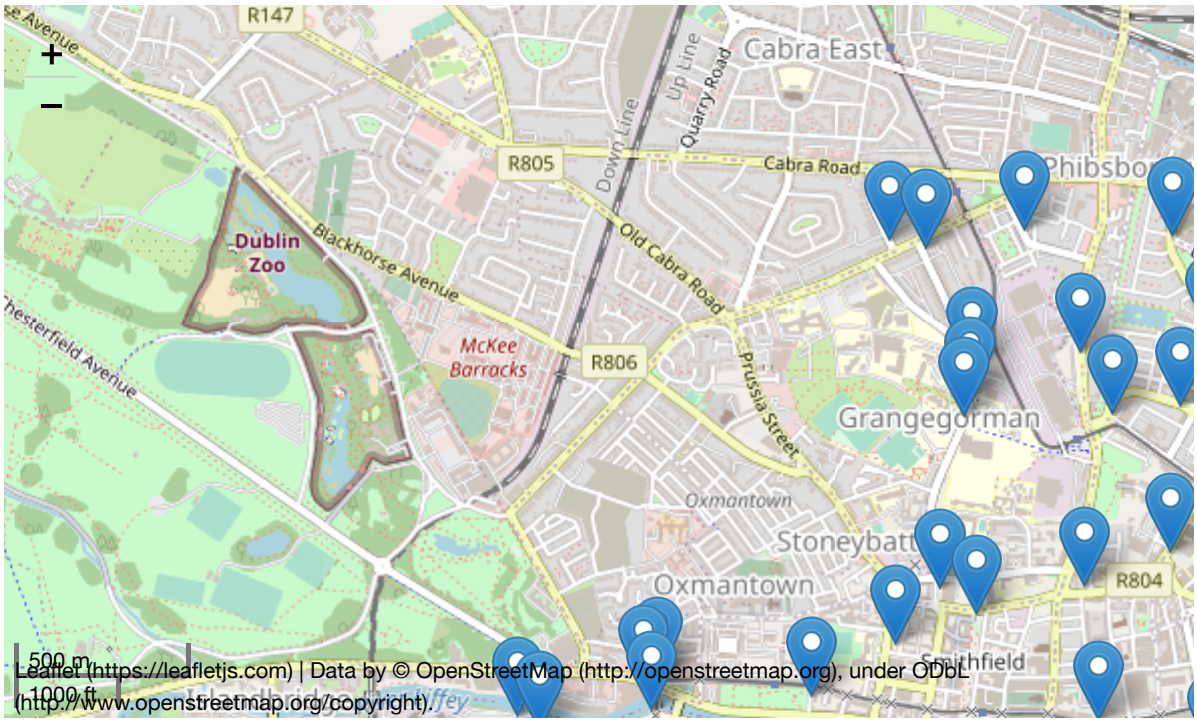
In [4]:
```
map
```

Out[4]:

Now adding the points for each bike station location to the map. Iterating through each row of the dataframe, I pass the location latitude and longitudes to folium.Marker as a list and pass the name to the popup parameter. And for each location I add to map:

In [5]:
```
for index, location_info in bike_station_locations.iterrows():
    folium.Marker([location_info["Latitude"], location_info["Longitude"]], po
```

In [6]:
```
map
```

Out[6]:


Demonstration : Folium with boundaries color coding

Now is about an advance feature of Folium , which is Map Boundary

To create the map boundary ,it require the upload of a GeoJSON file which is a file that encoding a variety of geographic data structures in JSON format. It also support varieties of geometry object types i.e. Point, LineString, Polygon,

To understand more about GeoJson , you can check the detail at
https://en.wikipedia.org/wiki/GeoJSON

What I will do with Folium

This document will demonstrate a usage of this library by plotting Thailand's map with

province map boundaty, and encode color code of each province match to thier income data of each province ,

A variety of tools can be used for geospatial Visualization , The tools that we are using here is "Folium" Library .

Folium library is a hand-on and easy to use ,the user can just create a data frame which contain Latitude", "Longitude" column and the Folium will plot them to a map automatically.

For the dataset it used Thailand province monthly income from Thailand's National statistical office (http://statbbi.nso.go.th/staticreport/page/sector/th/08.aspx). The data contain Thailand's provinces name and with monthly income. I choose the year 2019 for the income information. However, the provinces name were in Thai so I have to manually add the english name for each provinces.

In [7]:
```
import folium
from folium import plugins
from folium.plugins import BeautifyIcon

import json
import pandas as pd
```

In [8]:
```
thailand_income_df = pd.read_csv('./thailand_province_income.csv', encoding =
```

In [9]:
```
thailand_income_df.head()
```

Out[9]:

	Province_TH	Province_EN	Earning_2019
0	เชียงใหม่	Chiang Mai	20443.21
1	เชียงราย	Chiang Rai	15056.38
2	เพชรบุรี	Phetchaburi	26814.37
3	เพชรบูรณ์	Phetchabun	24623.56
4	เลย	Loei	25422.29

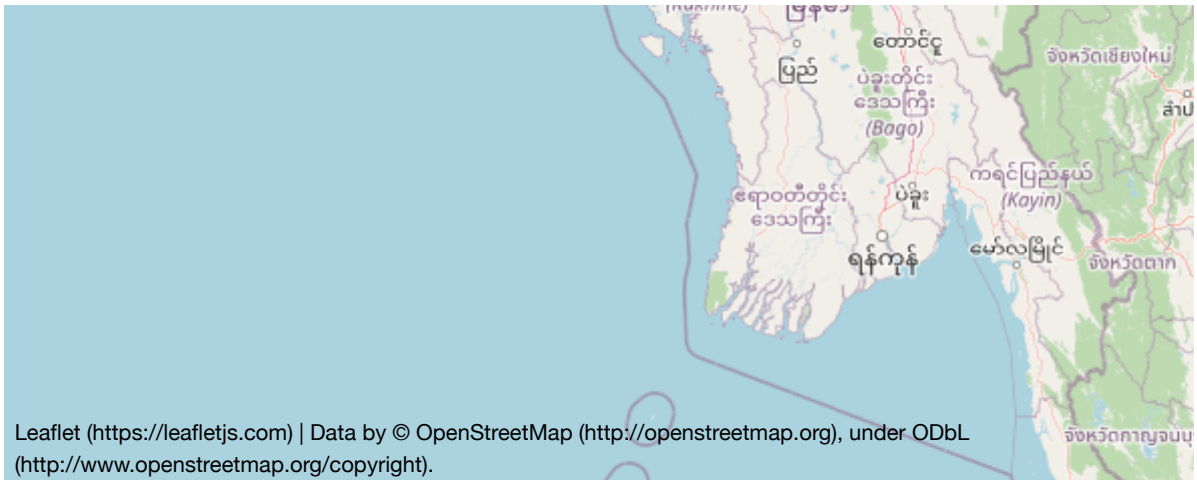
In [ ]:
```

```

The folium map can be easily created by following generic code. folium_map = folium.Map(location=start_coords, zoom_start, tiles)

In [10]:
```
thailand_map = folium.Map(location = [13.5954, 100.6072], zoom_start = 6 ,Tit
```

In [11]:
```
thailand_map
```

Out[11]:



Next we're going to draw the boundary of each Thailand's province into the folium map. then , we have to load Json file of the map to the base map. in this case, we are interested to work on Thailand GeoJSON, and the data can be found at : (https://github.com/apisit/thailand.json)

In [12]:
```
import requests
from os import getcwd
import urllib.request
import json

url='https://raw.githubusercontent.com/apisit/thailand.json/master/thailand.j
filename, headers = urllib.request.urlretrieve(url, filename="thailand1.json"


with open(filename, 'r') as j:
     thailand_geo_json = json.loads(j.read())
```

Now , let's check the name of the province

In [13]:
```
provinces= [province['properties']['name'] for province in thailand_geo_json[
print(provinces[0:5])
```

```
['Mae Hong Son', 'Chumphon', 'Nakhon Si Thammarat', 'Phuket', 'Phangnga']
```

let's check number of provinces.

In [14]:
```
num_province=len(provinces)
print('there are :',num_province,' provinces in this Json file')
```

```
there are : 77  provinces in this Json file
```

So, Now, we can see that the number of provinces and name are correct, you can re-check again with the provinces name from thailand_income_df , to ensure that name are exactly match ,(some manual data manupulation may requires to change the name)

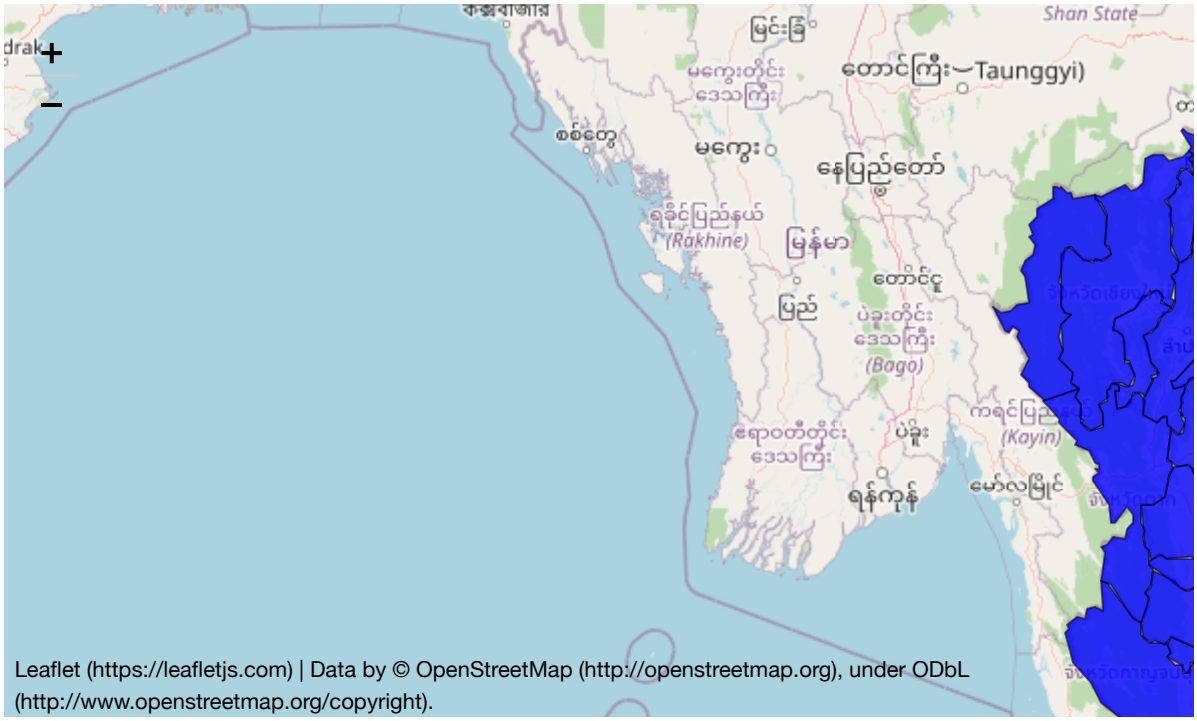
then, we can load this Json to Folium ,by using. the below code:

In [15]:
```
choropleth = folium.Choropleth(
    geo_data = thailand_geo_json,
    fill_opacity = 0.8,
    line_opacity = 0.5,
).add_to(thailand_map)
```

Now , we will able to see the boudary of each city in Thailand . it look like this>

In [16]:
```
thailand_map
```

Out[16]:



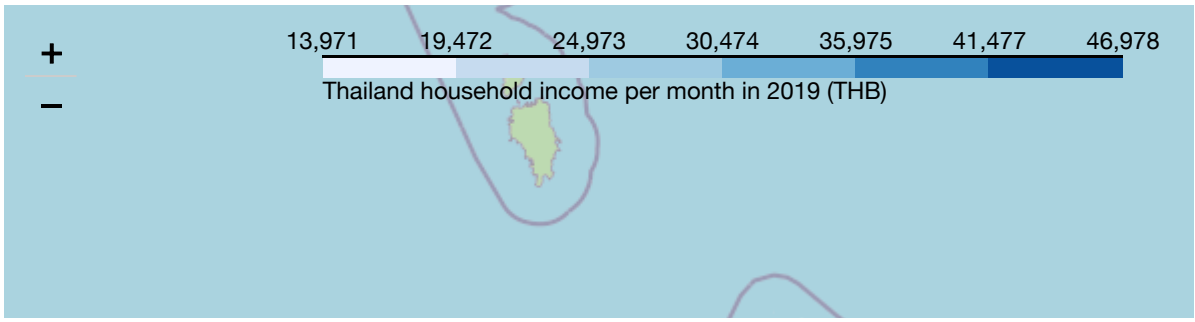
Next step we are goint to color each province base on the income of thier population. I choose the "choropleth" method which is is a thematic map in which areas are shaded according to the statistical variable being displayed on the map. This visualization is really useful to represent the variability of a measurement across a region using a sequential color scheme , the higher the measurement the stronger the color. In this case ,we just chose the varible to map color = Privince_EN , and Earning_2019

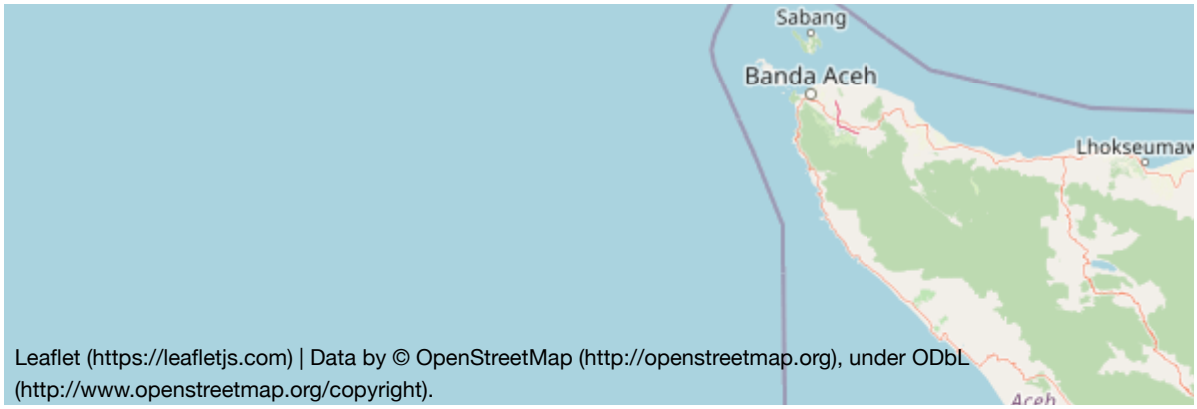
If you are interested to use anoter color coding , you can check the detail in Folium documents

In [17]:
```
choropleth = folium.Choropleth(
    geo_data = thailand_geo_json,
    name = 'Thailand household income',
    data = thailand_income_df,
    columns=['Province_EN', 'Earning_2019'],
    key_on = 'properties.name',
    fill_color = 'Blues',
    fill_opacity = 0.8,
    line_opacity = 0.5,
    legend_name = 'Thailand household income per month in 2019 (THB)',
).add_to(thailand_map)
```

In [18]:
```
thailand_map
```

Out[18]:





Here we can visualise how each provinces monthly income compare to other provinces. We can see that most high income provinces were around the center of Thailand.

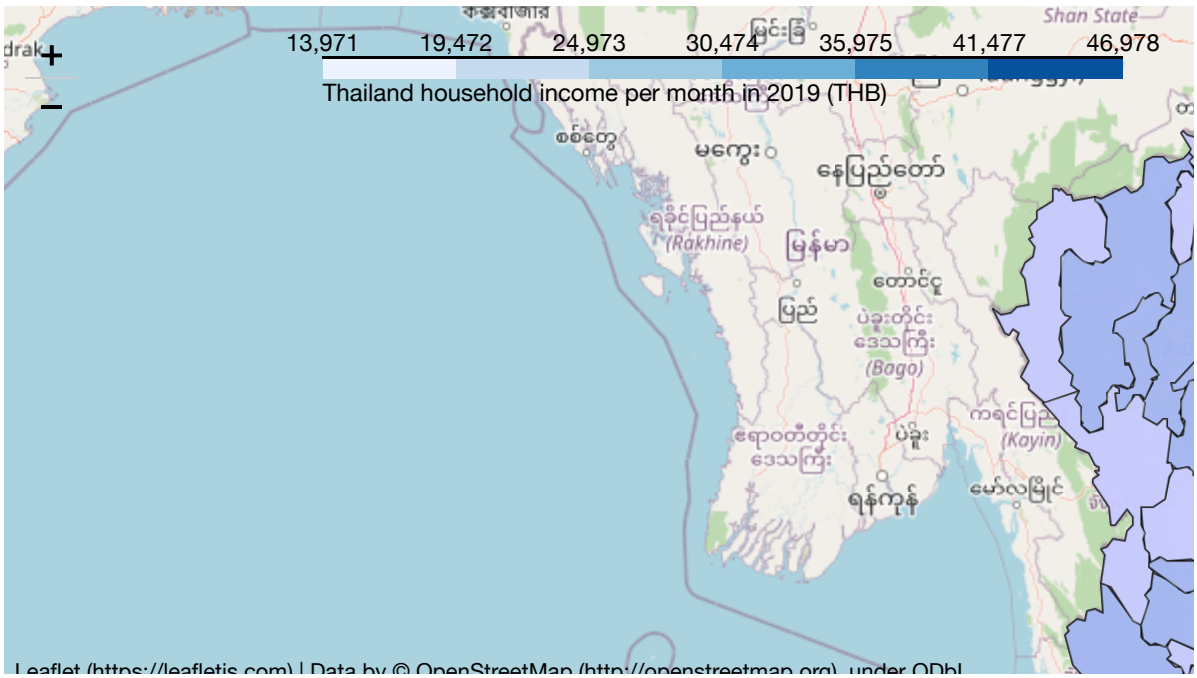
But for anyone who's not familar with thai geography how will they what the provinces name is shown in this map? The next step will add the tooltip pop up when we hover the mouse over each provinces.

In [19]:
```
for index, row in thailand_income_df.iterrows():

    for index, key in enumerate(thailand_geo_json['features']):

        province_name = key['properties']['name']

        if province_name == row['Province_EN']:

            tooltip_text = row['Province_EN'] + ' Monthly Income : ' + str(ro
            thailand_geo_json['features'][index]['properties']['tooltip'] = t
```

In [20]:
```
choropleth.geojson.add_child(
    folium.features.GeoJsonTooltip(['tooltip'], labels=False)
)
```

Out[20]:
```
<folium.features.GeoJson at 0x7f88427c6850>
```

In [21]:
```
thailand_map
```

Out[21]:


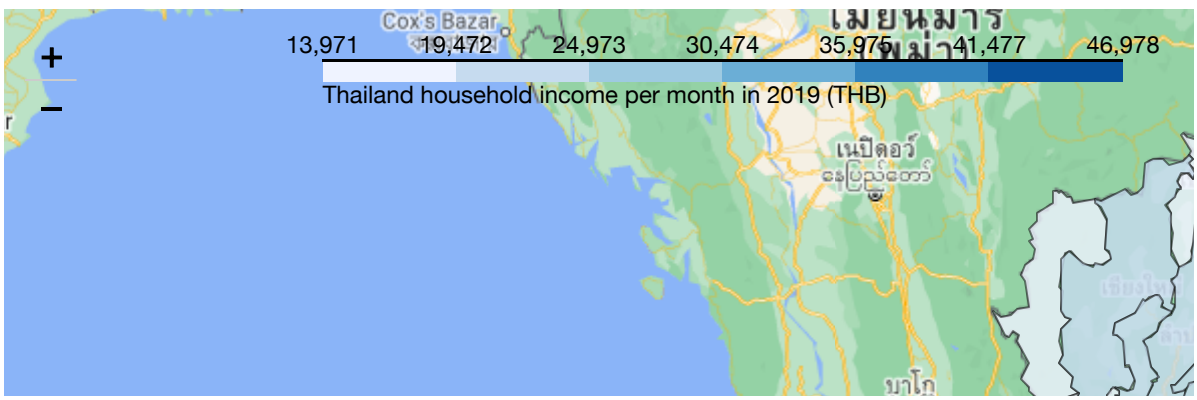
Lastly, we can add another map layer into folium map i.e. Google Satellite.

A multiple map layer can be draw into one folium map using layer control which going to appear in map and can be switch interactily.

In [22]:
```
basemaps = {
    'Google Maps': folium.TileLayer(
        tiles = 'https://mt1.google.com/vt/lyrs=m&x={x}&y={y}&z={z}',
        attr = 'Google',
        name = 'Google Maps',
        overlay = True,
        control = True
    ),
    'Google Satellite': folium.TileLayer(
        tiles = 'https://mt1.google.com/vt/lyrs=s&x={x}&y={y}&z={z}',
        attr = 'Google',
        name = 'Google Satellite',
        overlay = True,
        control = True
    )
}
```

In [23]:
```
thailand_map = folium.Map(location = [13.5954, 100.6072], zoom_start = 6)

folium.TileLayer('cartodbpositron').add_to(thailand_map)
basemaps['Google Satellite'].add_to(thailand_map)
basemaps['Google Maps'].add_to(thailand_map)

choropleth = folium.Choropleth(
    geo_data = thailand_geo_json,
    name = 'Thailand household income',
    data = thailand_income_df,
    columns=['Province_EN', 'Earning_2019'],
    key_on = 'properties.name',
    fill_color = 'Blues',
    fill_opacity = 0.8,
    line_opacity = 0.5,
    legend_name = 'Thailand household income per month in 2019 (THB)',
).add_to(thailand_map)

folium.LayerControl().add_to(thailand_map)

choropleth.geojson.add_child(
    folium.features.GeoJsonTooltip(['tooltip'], labels=False)
)

thailand_map
```

Out[23]:



Rule et al's rules for computational analyses.

Rule 1: Tell a story for an audience:

One key benefit of using Jupyter Notebooks is being able to interleave explanatory text with code and results to create a computational narrative . :, I start the story from what /how /benefit /disadvantage and demo step by step:

Rule 2: Document the process, not just the results

Avoid long cells (we suggest that anything over 100 lines or one page is too long). Put low-level documentation in code comments. Use descriptive markdown headers to organize your notebook into sections that can be used to easily navigate the notebook and add a table of contents. I would ensure my cell split and keep not too long , I will break them with a header(I chose level #6)

Rule 3: Use cell divisions to make steps clear

Computational notebooks' interactivity makes it quick and easy to try out and compare different approaches or parameters—so quick and easy that we often fail to document those interactive investigations at the time we perform them. : The users can inter-act witht the above maps to make them see data once the mouse click

Rule 4: Share and explain your data

Having access to a clearly annotated notebook is of little use to those wanting to reproduce or extend your results if the underlying data are locked away. Strive to make your data or a sample of your data publicly available along with the notebook. While sharing your data takes careful planning, notebooks make it easy to provide a description of your input data and upstream processing steps, which are essential for interpreting results.

Reference https://www.researchgate.net/publication/221025187_Geographic_Visualization
https://www.analyticsvidhya.com/blog/2020/06/guide-geospatial-analysis-folium-python/
https://github.com/apisit/thailand.json https://python-visualization.github.io/folium/
https://en.wikipedia.org/wiki/GeoJSON
https://www.pluralsight.com/guides/map-visualizations-in-python-using-folium