taipei_restaurant_distribution

October 6, 2020

1 Taipei Restaurant Distribution

We would also consider the competitor in seeing the situation. In this we would use the goverment data of travel network. What we would do in the analysis is: - Simulate restaurant distribution in Taipei from the sample data - Get simulated restaurant per area data

[1]:
```
# initial setup, import packages, path, and config
import json
import os

import geopandas as gpd
import numpy as np
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio
from shapely.geometry import MultiPoint
pd.options.mode.chained_assignment = None  # not show dataframe copy slice␣
↪warning
pio.renderers.default = 'jupyterlab'

from lib import shared_lib
from shared_lib import data_processor
from data_processor.lib.geocoding import GeoCoder
from data_processor.lib.geolib_helper import get_shp_filepath,␣
↪load_normalize_gov_shp_data

from lib.plotly_helper import add_chart_title, add_chart_annotation

# setup path
ANALYSIS_NAME = 'taipei_restaurant_distribution'

CURRENT_DIR = os.path.dirname(os.path.abspath('__file__'))
BASE_DIR = os.path.dirname(CURRENT_DIR)
ANALYSIS_DIR = os.path.join(BASE_DIR, 'analysis', ANALYSIS_NAME)

# setup plotly default config
plotly_default_config_chart = dict(
```

```
    displayModeBar=True,
    responsive=False,
    modeBarButtonsToRemove=['zoomIn2d', 'zoomOut2d', 'select2d', 'lasso2d',␣
 ↪'toggleSpikelines'],
    displaylogo=False
)

plotly_default_config_geo = dict(
    displayModeBar=True,
    responsive=False,
    scrollZoom=False,
    modeBarButtonsToRemove=['select2d', 'lasso2d'])
```

1.1 Simulate restaurant distribution in Taipei

First we would make our sample into the overall (population) data. We would do this by: - Make a sample-to-population multiplier - Randomize new data based on current sample data

[2]:
```
# setup data
# - area dimension table
area_dimension_table = pd.read_csv('../data/normalized-data_warehouse/
 ↪area_dimension_table.csv')
area_dimension_table = area_dimension_table.astype({'village_code':str})
area_dimension_table.set_index('village_code', inplace=True)

# - taipei area data, village detail
village_shp_path = get_shp_filepath(os.path.join(BASE_DIR, 'data',␣
 ↪'taiwan_twd97_map_data_village'))
village_gpd = load_normalize_gov_shp_data(village_shp_path)

taipei_village_gpd = village_gpd[village_gpd['county_chinese_name'] == '  ']
taipei_village_gpd.set_index('village_code', drop=False, inplace=True)

taipei_village_gpd = pd.merge(
    taipei_village_gpd, area_dimension_table[['township_english_name']],
    left_index=True, right_index=True
)

# - restaurant sample data
taipei_network_restaurant_data = pd.read_csv('../data/taipei_travel_network/
 ↪structured/taipei_travel_network.csv')
```

1.1.1 Setting sample-to-population multiplier

Some fact that we would use are: - World Cities Culture Forum show us in 2017 Taipei number of restaurant per 100,000 population is 307.6 - Goverment data show that Taipei population in 2016 was 2,695,704 people

Therefore we would use this assumption: - Current sample data distribution represent all of the population data (this is a bit heuristic and not really accurate, but for this case we would like to have a ballpark)

Therefore we would use this sample-to-population multiplier formula

$$sample\text{-}to\text{-}population\ multiplier = \left(\frac{Taipei\ population}{100\,000} * 307.6 \right) \Big/ Total\ sample\ data \tag{1}$$

$$\tag{2}$$

The calculation above would *26.92*. So we would make the sample data *27* times larger to simulate the population data.

1.1.2 Make the distribution formula

As our data sample is a bit specific and might not represent the real data, we try to create wider distribution. In this case, we would use radius limit of 5 km (1 point in lat/long coordinate is about 111 km), it would be rarely pass 5 km limit although it is possible. Therefore the distribution formula that we would use is exponential, which is:

$$f(x; \frac{1}{\beta}) = \left(\frac{1}{\beta} \exp(-\frac{x}{\beta}) \Big/ 5 \right) * max\ radius\ level \tag{3}$$

with: - $x > 0$ - $\beta = 1.5$ - make possibility into two identitcal side (negative / positive)

Therefore, the final formula would be

$$new\ data_{longitude} = (sample\ data_{longitude}) + \frac{f(x)}{111} \tag{4}$$

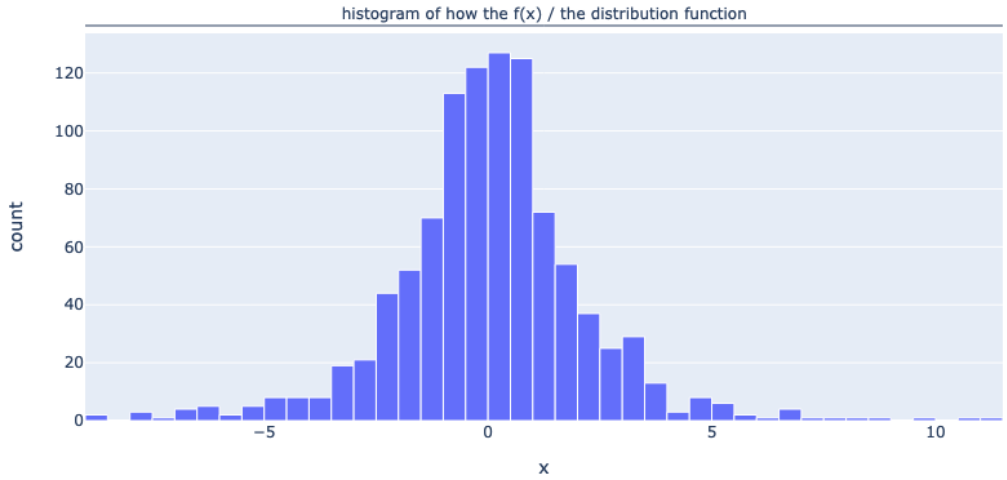
$$new\ data_{latitude} = (sample\ data_{latitude}) + \frac{f(x)}{111} \tag{5}$$

$$\tag{6}$$

And we would make duplicate the data by 27 times using the formula

[3]:
```
def dist_delta_function(max_value: float = 5):
    delta = np.random.exponential(1.5)
    if np.random.rand() >= 0.5:
        delta = delta * -1

    return (delta / 5) * max_value

hist_data = [dist_delta_function(5) for _ in range(1000)]

fig = px.histogram(x=hist_data)
fig.update_traces(
    marker=dict(
        line={'width':1, 'color':'white'}))

add_chart_title(fig, 'histogram of how the f(x) / the distribution function')
```

```
fig.show(config=plotly_default_config_chart)
fig.write_image(os.path.join(ANALYSIS_DIR,␣
↪'distribution_delta_function_probability.png'))
```



1.1.3 Compute the distribution coordinate

From the distribution function above, compute the simulated restaurant data

[4]:
```
sample_multiplier = 27

# generate the distribution sample data
def generate_new_long_lat(long_lat_dict: dict, rand_radius: float = 1) -> dict:
    new_long = long_lat_dict.get('longitude') + (dist_delta_function(5) /111)
    new_lat = long_lat_dict.get('latitude') + (dist_delta_function(5) /111)
    return {'longitude':new_long, 'latitude':new_lat}

sample_data_coordinate = []
simulated_restaurant_coordinate = []
for long, lat in zip(taipei_network_restaurant_data['longitude'],␣
↪taipei_network_restaurant_data['latitude']):
    _sample_coordinate = {'longitude': long, 'latitude': lat}
    sample_data_coordinate.append(_sample_coordinate)
    for _ in range(sample_multiplier-1):
        simulated_restaurant_coordinate.
↪append(generate_new_long_lat(_sample_coordinate))
```

1.1.4 Save and analyze data

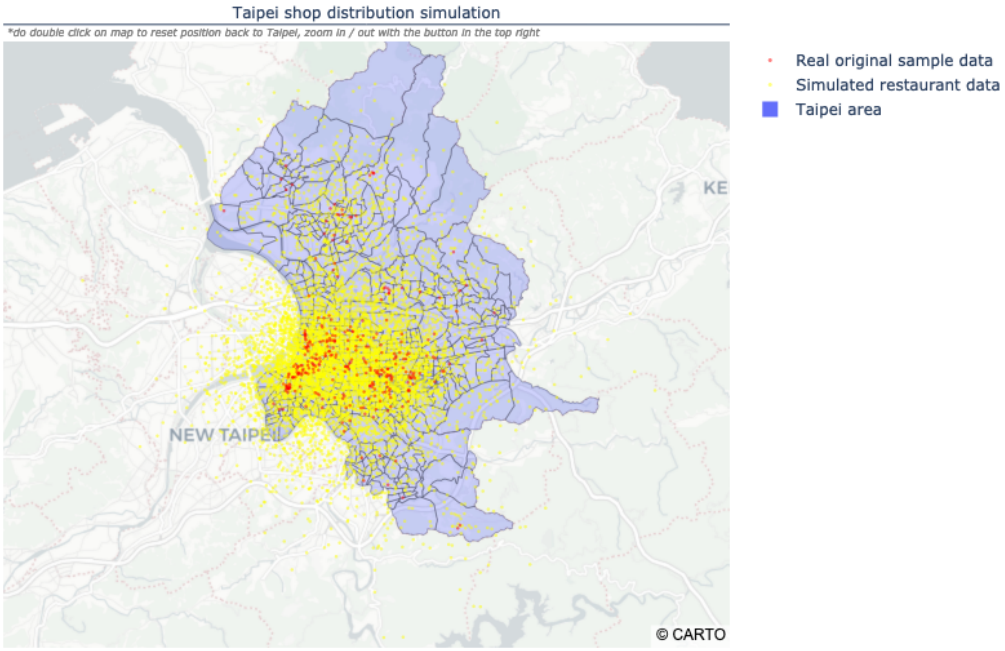
We would have the data saved and analyzed

[5]:
```
# prepare area data
taipei_village_geojson = json.loads(taipei_village_gpd.geometry.to_json())

center_point = MultiPoint(taipei_village_gpd['geometry'].apply(lambda x: x.
 ↪centroid)).centroid
```

[6]:
```
# plot the main graph
fig = px.choropleth_mapbox(taipei_village_gpd, geojson=taipei_village_geojson,
                           locations='village_code',
                           hover_name='village_english_name',
                           hover_data=['village_chinese_name',␣
 ↪'township_chinese_name'],
                           labels={'village_chinese_name': 'Village Chinese␣
 ↪Name',
                                   'township_chinese_name': 'Township Chinese␣
 ↪Name'},
                           opacity=0.3,
                           mapbox_style='carto-positron',
                           center={'lon':center_point.x, 'lat':center_point.y},
                           zoom=10)

fig.update_traces(dict(
                    name='Taipei area',
                    hovertemplate=fig['data'][-1]['hovertemplate']\
                        .replace('<br>village_code=%{location}<br>','')\
                        .replace('=', ' = ')))

fig.add_trace(go.Scattermapbox(
                name='Simulated restaurant data',
                lon=list(map(lambda x: x.get('longitude'),␣
 ↪simulated_restaurant_coordinate)),
                lat=list(map(lambda x: x.get('latitude'),␣
 ↪simulated_restaurant_coordinate)),
                marker=dict(
                  color='yellow',
                  size=3,
                  sizemode='area',
                  opacity=0.5
                ),
))

fig.add_trace(go.Scattermapbox(
                name='Real original sample data',
```

```
                lon=list(map(lambda x: x.get('longitude'),␣
 ↪sample_data_coordinate)),
                lat=list(map(lambda x: x.get('latitude'),␣
 ↪sample_data_coordinate)),
                marker=dict(
                  color='red',
                  size=3,
                  sizemode='area',
                  opacity=0.5
                ),
#                 customdata=taipei_mrt_map_df['station_name'],
#                 hovertemplate='Station Chinese Name = %{customdata}'
))

add_chart_title(fig, 'Taipei shop distribution simulation', 1.2)

add_chart_annotation(fig,
                     '<i>*do double click on map to reset position back to␣
 ↪Taipei, '
                     'zoom in / out with the button in the top right</i>')

fig.update_layout(dict(
    legend={'traceorder': 'reversed'}
))

fig.update_layout(dict(
    title=dict(
        text="Most restaurant is on mid-west side of Taipei",
        yanchor='top',
        yref='container', y=0.9,
    ),
    margin={'t':150},
    height=700
))

fig.show(config=plotly_default_config_geo)
fig.write_image(os.path.join(ANALYSIS_DIR,␣
 ↪'taipei_restaurant_distribution_simulation.png'))
```

Most restaurant is on mid-west side of Taipei



1.2 Get per area data

We would aggregate simulated restaurant per area to and then combine with other data points to make some solid recommendation.

[7]:
```
# get output filepath data
data_mart_dir = os.path.join(BASE_DIR, 'data', 'aggregated-data_mart')
save_taipei_restaurant_distribution_filepath = os.path.join(data_mart_dir,␣
 ↪ANALYSIS_NAME+'.csv')

# setup data
# - get area data, same as previous
taipei_village_gpd = taipei_village_gpd

# - get geo coder class
geo_coder = GeoCoder(village_shp_path, taipei_only=True)

simulated_restauran_coordinate_df = pd.
 ↪DataFrame(simulated_restaurant_coordinate)
```

```
sample_restaurant_coordinate_df = pd.DataFrame(sample_data_coordinate)

# - get restaurant coordinate
restaurant_coordinate_df = pd.concat(
    [simulated_restauran_coordinate_df, sample_restaurant_coordinate_df])
restaurant_coordinate_df.reset_index(drop=True, inplace=True)
```

1.2.1 Map data inside the village area

We would use the geo coder to map our restaurant data point, map it into Taipei village area

[8]:
```
# put area data into the map
# geocoding restaurant coordinate
lat_long_village_code_dict = geo_coder.long_lat_tuple_to_dict_multiprocessing(
    zip(restaurant_coordinate_df['longitude'],␣
 ↪restaurant_coordinate_df['latitude'])
)

restaurant_coordinate_df['village_code'] = restaurant_coordinate_df \
    .apply(lambda x: lat_long_village_code_dict.get((x['longitude'],␣
 ↪x['latitude'])), axis=1)

restaurant_coordinate_df[restaurant_coordinate_df['village_code'].isnull()]

restaurant_coordinate_agg_village = \
    restaurant_coordinate_df.groupby('village_code')['village_code'].count()
restaurant_coordinate_agg_village.rename('restaurant_count', inplace=True)
restaurant_coordinate_agg_village_df = pd.
 ↪DataFrame(restaurant_coordinate_agg_village)

taipei_village_gpd = pd.merge(taipei_village_gpd,␣
 ↪restaurant_coordinate_agg_village,
        how='left', left_index=True, right_index=True)

taipei_village_gpd['restaurant_count'].fillna(0, inplace=True)
taipei_village_gpd = taipei_village_gpd.astype({'restaurant_count':int})
```

1.2.2 Save and visualize data

After we compute the data, we would save and visualize the data.

[9]:
```
# prepare area data
taipei_village_geojson = json.loads(taipei_village_gpd.geometry.to_json())
center_point = MultiPoint(taipei_village_gpd['geometry'].apply(lambda x: x.
 ↪centroid)).centroid
```

```
taipei_township_restaurant_agg = taipei_village_gpd.groupby(['township_code',␣
↪'township_english_name'])['restaurant_count'].sum().reset_index()

taipei_township_restaurant_agg.sort_values('restaurant_count', ascending=False,␣
↪inplace=True)

save_df = taipei_village_gpd.loc[:, taipei_village_gpd.columns != 'geometry']
save_df.to_csv(save_taipei_restaurant_distribution_filepath, index=False)
```

[10]:
```
# draw first chart the map
fig = px.choropleth_mapbox(taipei_village_gpd, geojson=taipei_village_geojson,
                           locations='village_code',
                           color='restaurant_count',
                           hover_name='village_english_name',
                           hover_data=['township_english_name'],
                           labels={'township_english_name': 'Township English␣
↪Name',
                                   'restaurant_count': 'Restaurant Count'},
                           color_continuous_scale='OrRd',
                           range_color=(0,100),
                           opacity=0.5,
                           mapbox_style='carto-positron',
                           center={'lon':center_point.x, 'lat':center_point.y},
                           zoom=10)

fig.update_traces(hovertemplate=fig['data'][-1]['hovertemplate']\
                  .replace('village_code=%{location}<br>','')\
                  .replace('=',' = ')\
                  .replace('{z}','{z:,.2r}')
                 )

add_chart_title(fig, "Taipei color scale map based number of simulated␣
↪restaurant per village", 1.2)

add_chart_annotation(fig,
                     '<i>*do double click on map to reset position back to␣
↪Taipei, '
                     'zoom in / out with the button in the top right</i>')

fig.update_layout(
    title='There is lots of reastaurant on mid-west area of Taipei',
    margin={'t':120},
    height=700
)

fig.show(config=plotly_default_config_geo)
```

```
fig.write_image(os.path.join(ANALYSIS_DIR, 'taipei_restaurant_distribution-1.
 ↪png'))

# draw second chart, top 5 bar chart
fig = px.bar(taipei_village_gpd,
             x='township_english_name',
             y='restaurant_count',
             labels={
                 'township_english_name': 'Township English Name',
                 'restaurant_count': 'Restaurant Count',
                 'village_chinese_name': 'Village Chinese Name'
             },
             color='village_chinese_name')

fig.update_traces(hovertemplate=fig['data'][-1]['hovertemplate']\
                  .replace('=',' = ')\
                 )

fig.update_traces(marker={'color': 'blue'})

fig.update_xaxes(categoryorder='array',
                 ␣
 ↪categoryarray=taipei_township_restaurant_agg['township_english_name'])

fig.update_layout(showlegend=False)
fig.update_xaxes(fixedrange=True)
fig.update_yaxes(fixedrange=True)

add_chart_title(fig, "Simulated restaurant count, stacked per township", 2)

fig.add_shape(type='rect',
              xref='x', x0=-0.6, x1=2.5, yref='paper', y0=0, y1=1,
              line=dict(
                  color='orange',
                  width=4
              ))

fig.show(config=plotly_default_config_chart)
fig.write_image(os.path.join(ANALYSIS_DIR, 'taipei_restaurant_distribution-2.
 ↪png'))
```

There is lots of reastaurant on mid-west area of Taipei