taipei_passerby_buying_power

October 6, 2020

1 Taipei Passerby Buying Power

In order to know the demand / the market, we would try to predict passerby buying power (for dining out) as one of the factor (later combined with number of buyer / passerby). In determining the buying power, we would do: - Get average income from each people, based on their living area - Translate it into area of their activities, based on the source of passerby

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

import pandas as pd
import geopandas as gpd
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'

import dask
import dask.dataframe as dd
from dask.distributed import Client, LocalCluster

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

# dask config
cluster = LocalCluster(
#     n_workers=os.cpu_count()  #  if want to setup number of worker
)
client = Client(cluster)
```

```
# setup path
ANALYSIS_NAME = 'taipei_passerby_buying_power'

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 Get average monthly income for each passenger

We would use the tax data to geth monthly income for each passenger (based on living area). From the income, we would make dining out budget based on income.

[2]:
```
# setup data source:
data_dir = os.path.join(BASE_DIR, 'data')
data_warehouse_dir = os.path.join(data_dir, 'normalized-data_warehouse')

# - 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
)

# - taipei income data
taipei_income_by_village_filepath = os.path.join(data_warehouse_dir,␣
↪'taipei_income_by_village.csv')
taipei_income_by_village_df = pd.read_csv(taipei_income_by_village_filepath)
taipei_income_by_village_df =\
    taipei_income_by_village_df[taipei_income_by_village_df['county_id']=='A']
taipei_income_by_village_df =\
    taipei_income_by_village_df[~taipei_income_by_village_df['village_code'].
↪isnull()]
taipei_income_by_village_df = taipei_income_by_village_df.
↪astype({'village_code':str})
taipei_income_by_village_df['village_code'] =\
    taipei_income_by_village_df['village_code'].apply(lambda x: x.split('.')[0])
taipei_income_by_village_df.set_index('village_code', inplace=True)
```

1.1.1 Compute dining out budget

This is based on: - Taipei news that said in 2018, on average Taiwanese income breakdown are: 15.6% for dining out.

Therefore, to compute the passerby we use:

$$\begin{aligned} \textit{possible food budget percentage} &= \textit{dining out budget percentage} && (1)\\ &= 15.6\% && (2) \end{aligned}$$

[3]:
```
taipei_village_gpd = pd.merge(taipei_village_gpd,␣
↪taipei_income_by_village_df[['average']],
        how='left', left_index=True, right_index=True)

taipei_village_gpd['average'] = (taipei_village_gpd['average'] * 1000) / 12
taipei_village_gpd.loc[taipei_village_gpd['average'].isnull(), 'average'] = 0
taipei_village_gpd = taipei_village_gpd.rename(columns={'average':
↪'monthly_average_income'})

taipei_village_gpd['dining_out_average_monthly_budget'] =\
    taipei_village_gpd['monthly_average_income'] * 0.156

taipei_village_gpd['dining_out_average_weekly_budget'] =\
    taipei_village_gpd['dining_out_average_monthly_budget'] / (30 / 7)

taipei_village_gpd['dining_out_average_daily_budget'] =\
    taipei_village_gpd['dining_out_average_monthly_budget'] / 30
```

1.2 Calculate passerby buying power on thier activities region

We want the buying power not based on living area, but based on their activites. Therefore we would combine the transportation data and income data to make passerby buying power data

[4]:
```
# setup the data source
data_dir = os.path.join(BASE_DIR, 'data')
data_warehouse_dir = os.path.join(data_dir, 'normalized-data_warehouse')

# - get 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 and taipei income data, from previous calculation
taipei_village_gpd = taipei_village_gpd

# - taipei_mrt_info
taipei_mrt_info_dirpath = os.path.join(data_warehouse_dir, 'taipei_mrt_info')
taipei_mrt_info_urlpath = os.path.join(taipei_mrt_info_dirpath,␣
 ↪'taipei_mrt_passenger_data_*.csv')

source_df = dd.read_csv(
    taipei_mrt_info_urlpath,
    dtype={
        'station_in_village_code':str,
        'station_out_village_code':str
    }
)

# - taipei village distance matrix
taipei_village_distance_matrix_filepath =\
    os.path.join(data_warehouse_dir,␣
 ↪'taipei_village_centroid_distance_km_matrix.csv')
taipei_village_distance_matrix_df = pd.
 ↪read_csv(taipei_village_distance_matrix_filepath)
taipei_village_distance_matrix_df.set_index('village_code', inplace=True)
```

1.2.1 Demonstrate method

Firstly we would demonstrate how is the method is looks like. We would use simplification (aggregated) data of township level for this purpose.

[5]:
```
# get aggregate data
df = source_df
df = df[(df['time_period'] >= 6) & (df['time_period'] <= 11)]
```

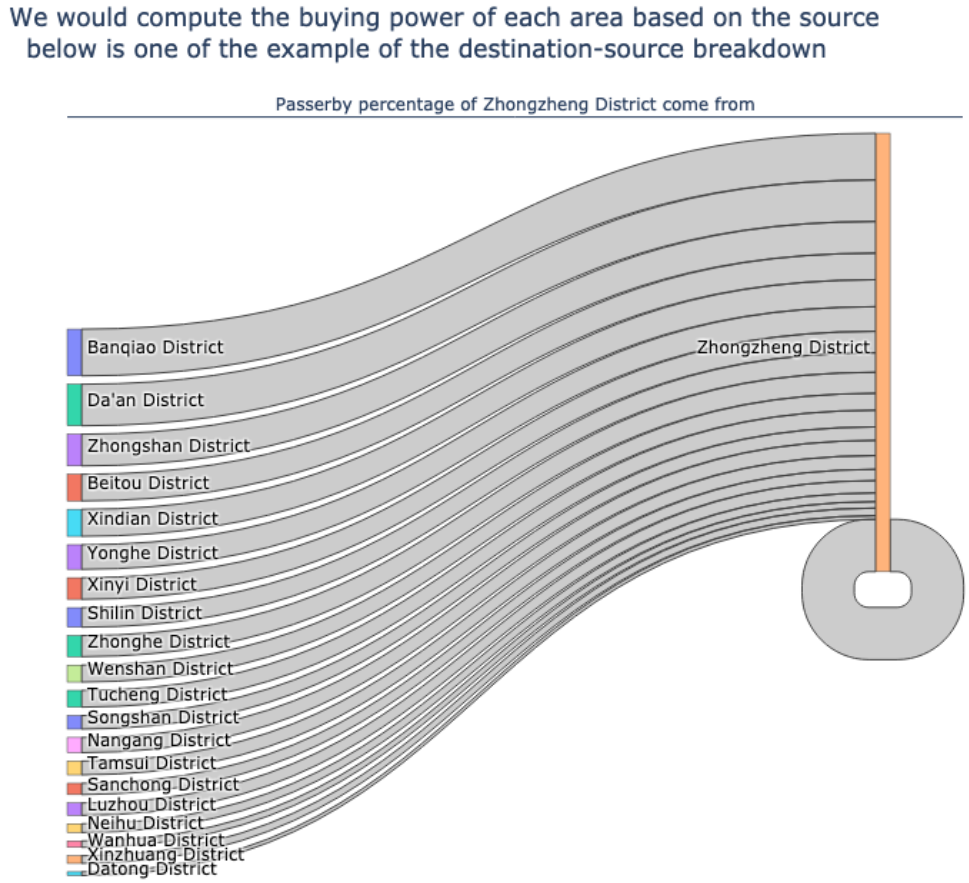
```
df = dd.merge(df, area_dimension_table['township_code'],
              how='left', left_on='station_in_village_code', right_index=True)
df = df.rename(columns={'township_code': 'station_in_township_code'})

df = dd.merge(df, area_dimension_table['township_code'],
              how='left', left_on='station_out_village_code', right_index=True)
df = df.rename(columns={'township_code': 'station_out_township_code'})

agg_df = df\
    .groupby(['station_in_township_code', 'station_out_township_code', 'date'])\
        ['person_times'].sum()\
    .reset_index()

agg_df = agg_df\
    .groupby(['station_in_township_code', 'station_out_township_code'])\
        ['person_times'].mean()\
    .reset_index()

_agg_df_right = agg_df\
    .groupby(['station_out_township_code'])['person_times']\
        .apply(lambda x: x / float(x.sum()), meta='float')\
    .reset_index(drop=True)
_agg_df_right = _agg_df_right.rename('person_times')

agg_df = dd.merge(agg_df[['station_in_township_code',␣
↪'station_out_township_code']], _agg_df_right,
        left_index=True, right_index=True)

taipei_mrt_passenger_come_from_agg = agg_df.compute()
```

[6]:
```
# prepare visualization data
mrt_dimension_table_agg = area_dimension_table.groupby(
    ['township_code', 'township_english_name', 'township_chinese_name']).
↪count().index

mrt_dimension_table_agg = pd.DataFrame(
    list(mrt_dimension_table_agg), columns=mrt_dimension_table_agg.names)

mrt_dimension_table_agg['sankey_index'] = range(len(mrt_dimension_table_agg))

mrt_dimension_table_agg = mrt_dimension_table_agg.astype({'township_code':str})
mrt_dimension_table_agg.set_index('township_code', inplace=True)

# mapping the data
taipei_mrt_passenger_come_from_agg = taipei_mrt_passenger_come_from_agg.astype({
    'station_in_township_code':str,
    'station_out_township_code':str
```

```
})

taipei_mrt_passenger_come_from_agg['station_in_sankey_index'] =\
    taipei_mrt_passenger_come_from_agg['station_in_township_code']\
        .apply(lambda x: mrt_dimension_table_agg.loc[str(x), 'sankey_index'])

taipei_mrt_passenger_come_from_agg['station_out_sankey_index'] =\
    taipei_mrt_passenger_come_from_agg['station_out_township_code']\
        .apply(lambda x: mrt_dimension_table_agg.loc[str(x), 'sankey_index'])

# use Zhongzheng ( ), Taipei, as example
sample_df = \
    taipei_mrt_passenger_come_from_agg\
        ␣
 ↪[taipei_mrt_passenger_come_from_agg['station_out_township_code']=='63000050']
```

[7]:
```
# draw sankey diagram
fig = go.Figure()

fig.add_trace(go.Sankey(
    valueformat='.2f',
    valuesuffix=' %',
    node=dict(
        label = mrt_dimension_table_agg.
 ↪sort_values('sankey_index')['township_english_name'],
        pad = 3,
        thickness = 10,
    ),
    link=dict(
        source = sample_df['station_in_sankey_index'],
        target = sample_df['station_out_sankey_index'],
        value = sample_df['person_times'].apply(lambda x: x * 100),
        line = {'width':1}
    )
))

fig.update_layout(
    height=800,
)

add_chart_title(fig, 'Passerby percentage of Zhongzheng District come from')

fig.update_layout(
    title='We would compute the buying power of each area based on the␣
 ↪source<br>'
        '  below is one of the example of the destination-source breakdown',
    margin={'t':150}
```

```
)

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



1.2.2 Compute buying power on village level

We would use village level detail to compute the buying power.

[8]:
```
# save path
data_mart_dir = os.path.join(BASE_DIR, 'data', 'aggregated-data_mart')
```

```
save_taipei_passeryby_buying_power_filepath = os.path.join(data_mart_dir,␣
↪ANALYSIS_NAME + '.csv')

# data used for computation
df = source_df
df = df[(df['time_period'] >= 6) & (df['time_period'] <= 11)]

agg_df = df\
    .groupby(['station_in_village_code', 'station_out_village_code', 'date'])\
        ['person_times'].sum()\
    .reset_index()

agg_df = agg_df\
    .groupby(['station_in_village_code', 'station_out_village_code'])\
        ['person_times'].mean()\
    .reset_index()

_agg_df_right = agg_df\
    .groupby(['station_out_village_code'])['person_times']\
        .apply(lambda x: x / float(x.sum()), meta='float')\
    .reset_index(drop=True)
_agg_df_right = _agg_df_right.rename('person_times')

agg_df = dd.merge(agg_df[['station_in_village_code',␣
↪'station_out_village_code']], _agg_df_right,
        left_index=True, right_index=True)

taipei_mrt_passenger_come_from = agg_df.compute()
```

[9]:
```
# make dictionary per village out
village_average_dining_out_weekly_budget_dict =\
    dict(taipei_village_gpd['dining_out_average_weekly_budget'])

def dict_helper(lookup_dict: dict, key: any) -> any:
    if key in lookup_dict:
        return lookup_dict.get(key)
    else:
        return 0

taipei_mrt_passenger_come_from['no_income_data'] = \
    taipei_mrt_passenger_come_from['station_in_village_code']\
        .apply(lambda x: 1 if x in␣
↪village_average_dining_out_weekly_budget_dict else 0)

# getting normalized proportion as there is no income data case
taipei_mrt_passenger_come_from.groupby(['station_in_village_code',␣
↪'no_income_data'])
```

```
_normalized_proportion = \
    taipei_mrt_passenger_come_from.groupby(['station_out_village_code',␣
 ↪'no_income_data'])['person_times'].sum().reset_index()
_normalized_proportion =␣
 ↪_normalized_proportion[_normalized_proportion['no_income_data']==1]
normalized_proportion_dict = dict(zip(
    _normalized_proportion['station_out_village_code'],
    _normalized_proportion['person_times']
))

taipei_mrt_passenger_come_from['normalized_proportion'] = \
    taipei_mrt_passenger_come_from['station_out_village_code'].apply(lambda x:␣
 ↪normalized_proportion_dict.get(x))

taipei_mrt_passenger_come_from['dining_out_average_weekly_budget_part'] =\
    taipei_mrt_passenger_come_from\
        .apply(
            lambda x: dict_helper(
                village_average_dining_out_weekly_budget_dict,
                str(x['station_in_village_code'])
            ) * (x['person_times'] / x['normalized_proportion']),
            axis=1
        )

per_passerby_dining_out_weekly_budget_dict = taipei_mrt_passenger_come_from\
    .
 ↪groupby('station_out_village_code')['dining_out_average_weekly_budget_part'].
 ↪sum().to_dict()
```

1.2.3 simulate walking passerby

Noted on distribution formula, based on passerby formula: The distribution formula is based on: - previous maximum determined radius, 2 km distribution function:

$$f(x) = \frac{3x^2}{10} - \frac{17x}{6} + 1 \tag{3}$$

with result no more than 1 or less than 0 - from that, compute the passerby average weekly dining out budget proportionate to the final distributed portion of the value

$$passerby\ weekly\ dining\ out\ budget_{final} = \sum_{n=1}^{\infty} f_2(f(x)) * passerby\ weekly\ dining\ out\ budget_{village\ in} \tag{4}$$

with:

$f_2(f(x)) = \frac{f(x)}{\sum_{n=1}^{\infty} f(x)_n}$ if $passerby_{village\ out}$ exist

$f_2(f(x)) = \frac{f(x)}{\sum_{n=1}^{\infty} f(x)_n - 1}$ if $passerby_{village\ out}$ not exist

[10]:
```
# calculate simulation using the formula
def distribution_function(x):
    if x > 2:
        return 0
    elif x <= 0:
        return 1
    else:
        fx = (3*(pow(x,2))/10) - ((7*x)/6) + 1.14
        if fx > 1:
            return 1
        elif fx < 0:
            return 0
        else:
            return fx

taipei_village_distance_matrix_df = taipei_village_distance_matrix_df\
    .applymap(lambda x: distribution_function(x))

# normalize the for not exist situation
taipei_village_distance_matrix_sum = taipei_village_distance_matrix_df.sum()
available_village_out_set =␣
↪set(taipei_mrt_passenger_come_from['station_out_village_code'])

for index, value in taipei_village_distance_matrix_sum.iteritems():
    if index not in available_village_out_set:
        taipei_village_distance_matrix_sum.loc[index] = value - 1

distance_matrix_distribution_total_dict = taipei_village_distance_matrix_sum.
↪to_dict()

final_passerby_weekly_dining_out_budget_dict = {}
for index, row in taipei_village_distance_matrix_df.iterrows():
    _weekly_budget = dict_helper(per_passerby_dining_out_weekly_budget_dict,␣
↪str(index))
    _calculated_row = {}
    for col_name, row_value in row.iteritems():
        _total_dist_matrix = distance_matrix_distribution_total_dict.
↪get(str(col_name))
        if _total_dist_matrix and row_value and _weekly_budget:
            _calculated_row_value = (row_value / _total_dist_matrix) *␣
↪_weekly_budget
            _calculated_row = {**_calculated_row, **{col_name:␣
↪_calculated_row_value}}
        else:
            _calculated_row = {**_calculated_row, **{col_name: 0}}
```

```
    final_passerby_weekly_dining_out_budget_dict = {
        **final_passerby_weekly_dining_out_budget_dict,
        **{index: _calculated_row}
    }

final_passerby_weekly_dining_out_budget_df = \
    pd.DataFrame.from_dict(final_passerby_weekly_dining_out_budget_dict,␣
 ↪orient='index')

final_passerby_weekly_dining_out_budget_dict =␣
 ↪final_passerby_weekly_dining_out_budget_df.sum().to_dict()
```

1.2.4 Save and visualize the data

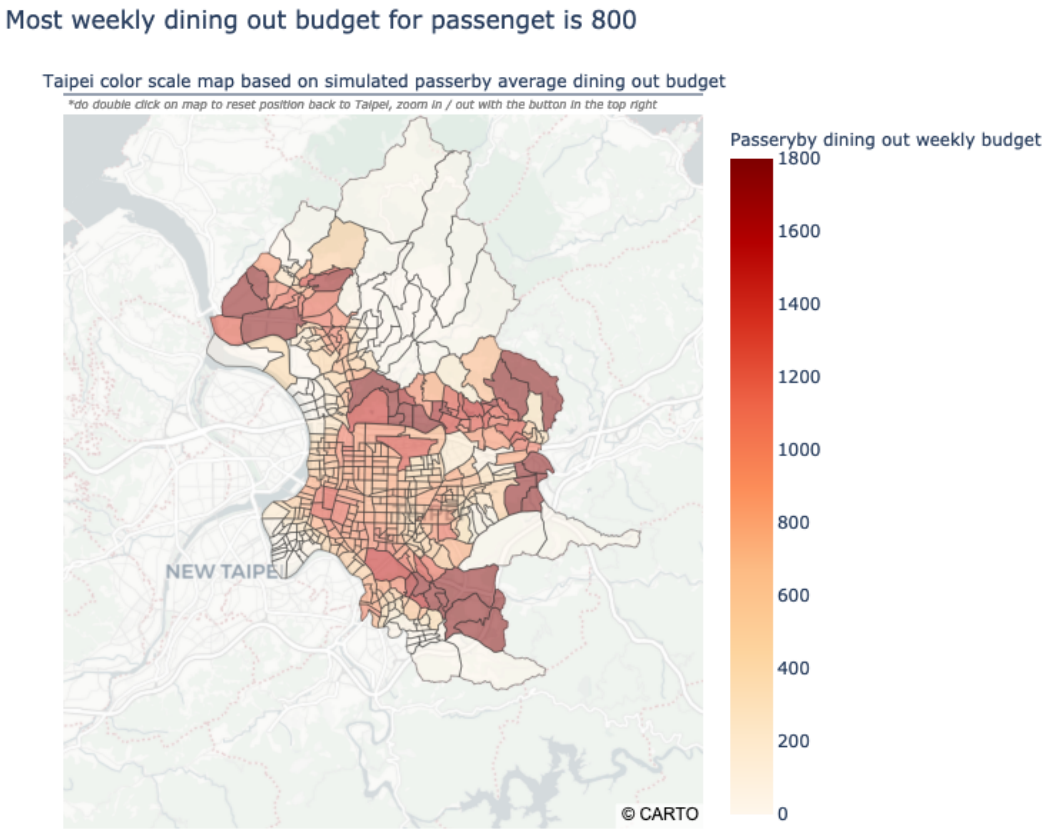
From all we make the final data, save, and visualize it.

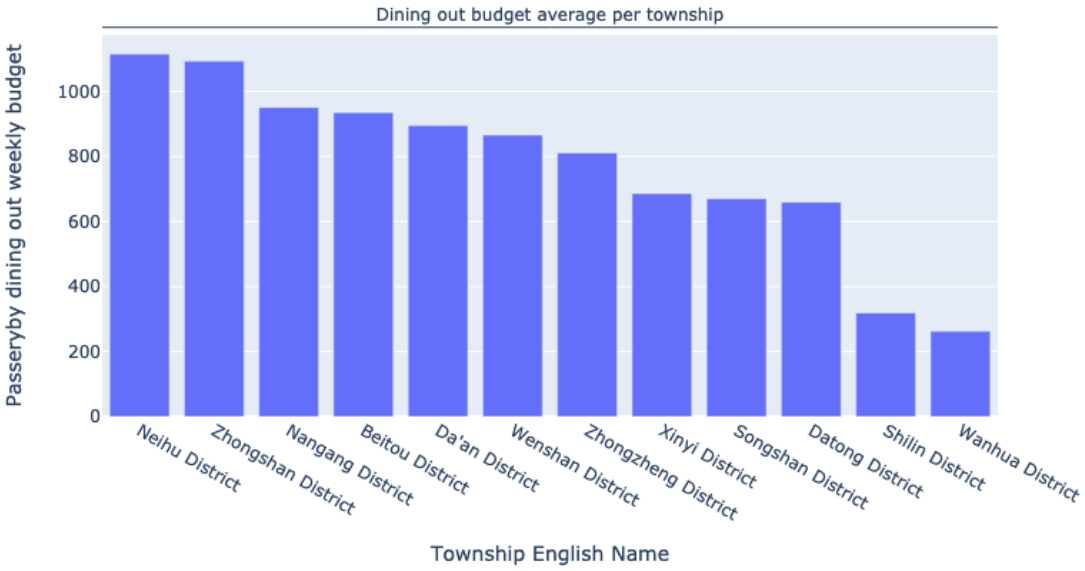
[11]:
```
# prepare map data
taipei_village_geojson = json.loads(taipei_village_gpd.geometry.to_json())

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

taipei_village_gpd['passerby_dining_out_weekly_budget'] =\
    taipei_village_gpd['village_code']\
        .apply(lambda x:␣
 ↪dict_helper(final_passerby_weekly_dining_out_budget_dict, str(x)))

taipei_township_dining_out_budget_average = \
    taipei_village_gpd.groupby(['township_code',␣
 ↪'township_english_name'])['passerby_dining_out_weekly_budget'].mean().
 ↪reset_index()

taipei_township_dining_out_budget_average = \
    taipei_township_dining_out_budget_average.
 ↪sort_values('passerby_dining_out_weekly_budget', ascending=False)

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

[15]:
```
# draw first chart, map
fig = px.choropleth_mapbox(taipei_village_gpd, geojson=taipei_village_geojson,
                           locations='village_code',
                           color='passerby_dining_out_weekly_budget',
                           hover_name='village_english_name',
                           hover_data=['township_english_name'],
```

```
                            labels={'township_english_name': 'Township English␣
 ↪Name',
                                    'passerby_dining_out_weekly_budget':␣
 ↪"Passeryby dining out weekly budget"},
                            color_continuous_scale='OrRd',
                            range_color=(0,1800),
                            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 on simulated passerby␣
 ↪average dining out budget", 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='Most weekly dining out budget for passenget is 800',
    margin={'t':120},
    height=700
)

fig.show(config=plotly_default_config_geo)
fig.write_image(os.path.join(ANALYSIS_DIR, 'dining_out_budget_distribution-1.
 ↪png'))

# draw second chart, bar chart of average
fig = px.bar(taipei_township_dining_out_budget_average,
             x='township_english_name',
             y='passerby_dining_out_weekly_budget',
             labels={'township_english_name': 'Township English Name',
                     'passerby_dining_out_weekly_budget': "Passeryby dining out␣
 ↪weekly budget"})

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

```
fig.update_yaxes(fixedrange=True)

add_chart_title(fig, 'Dining out budget average per township')

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




Dining out budget average per township
Passeryby dining out weekly budget
0
200
400
600
800
1000
Neihu District
Zhongshan District
Nangang District
Beitou District
Da'an District
Wenshan District
Zhongzheng District
Xinyi District
Songshan District
Datong District
Shilin District
Wanhua District
Township English Name