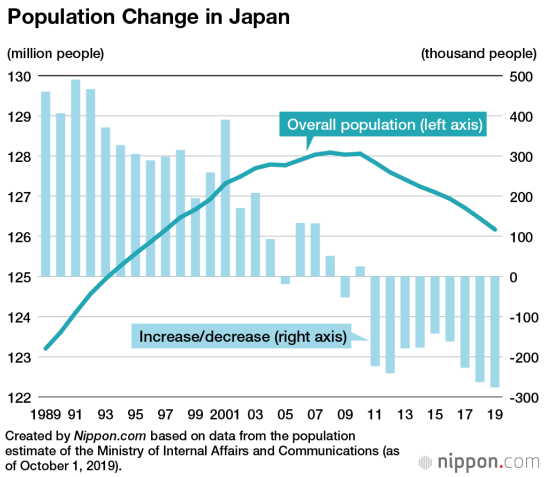
An Investigation to find the most suitable place

to open a Coffeshop in Ibaraki-Prefecture, Japan

1. Introduction

1.1. Background

- Ibaraki Prefecture (Ibaraki-ken) is a prefecture of Japan located in the Kantō region, so it locates quite near from Tokyo. There're 2 train lines running through many citites in Ibaraki directly to Tokyo, those are Janan Railway's Joban-line & Tsukuba Express line. Comparing to expensive living cost in Tokyo. Ibaraki may be a good place to for people to live.



- On the other hand, Japan's population began to decline in 2011 & the falling speed increasing year by year. This caused by the very low birth rate & aging in Japan. For specific city or area, unless many people come from other cities or foreigners come from overseas, this city may face many problems as lack of labors, decrease in taxt income causing bad public services. And also, it's risky & difficult to decide to open a new business in that kind of city or area.

1.2. Business problem

Writer of this report is thinking about opening a new Coffee shop in Ibaraki-prefecture. Considering about the location & population trending mentioned in above, he set a list of criteria to clarify which location should be the most suitable place to open a Coffeshop.

- The coffee shop should be located in city which in *top populated cities in Ibaraki*.
- The location should be *near Tokyo* so South of Ibaraki is preferrable.
- The city's *population should be stable or increasing* (many people coming to this city to cover the natural decrease cause by low birth rate & aging).
- The location should be *near some train stations*.
- There are *not too many competitors* in the area.

In this report, the writer demonstrates a data science approaching to find a place in Ibaraki-prefecture

that meets all criterias above to open a new Coffee shop.

2. Data & Methodology Explanation

2.1. Data

- Open source data about Ibaraki-prefecture's Population: https://www.pref.ibaraki.jp/kikaku/tokei/fukyu/tokei/betsu/jinko/getsu/jinko2001.html (the website is in Japanese but in next sections, datasheets will be translated to English)
- Foursquare location data: https://foursquare.com/city-guide

2.2. Methodology

- Applying "Request" & "BeautifulSoup" to retrieve, wrangling, creating pandas datasheets from Ibaraki-prefecture's website. (Translation Indexes & Columns name to English)
- Visualizing population data using Pythons Vizualization tools to clarify the best appropriate city.
- Indentifying the train stations location (longtitudes & latittudes) in this city.
- Using Foursquare API & Foursquare location data to investigate Neighboorhoods in these stations, comparing to find which place has not too many competiors (type of avenue is coffee shop).

3. Exploring the most attractive city in Ibaraki-prefecture by population data

3.1. Data retrieval, exploration and wrangling

Writer found that open-soured data provided by Ibaraki-prefecture government could be a good material for the investigation.

URL: https://www.pref.ibaraki.jp/kikaku/tokei/fukyu/tokei/betsu/jinko/getsu/jinko2001.html

There were totally 7 tables of data about population in Ibaraki which were updated in 2020, Janary.

List of the tables:

- Table 1: PUPULATION AND HOUSEHOLD TRENDS FROM 1955 to 2019
- Table 2: PREFECTURE HOUSEHOLDS, POPULATION, MIGRATION BY MONTHS IN 2019
- Table 3: IBARAKI'S AREAS HOUSEHOLDS, POPULATION, MIGRATION
- Talbe 4-1: CITIES, TOWNS & VILLAGES AREAS HOUSEHOLDS, POPULATION, MIGRATION (CITIES PART)
- Table 4-2: CITIES, TOWNS & VILLAGES AREAS HOUSEHOLDS, POPULATION, MIGRATION (TOWN PART)
- Table 5: TABLE 5: POPULATION RANKS BY CITIES, TOWNS & VILLAGES
- Table 6: CHANGES IN POPULATION OVER THE LAST YEAR BY CITIES

"Requests" & "beautifulsoup" library was used to retrieve & wrangle data in the sites. Because all the sites is in Japanese, while scraping, "r.encoding" was used to prevent Japanese characters from corruption.

```
# Step 1: Sending a HTTP request to a URL
url = "https://www.pref.ibaraki.jp/kikaku/tokei/fukyu/tokei/betsu/jinko/getsu/jinko2001.html"
# Make a GET request to fetch the raw HTML content
r = requests.get(url)
content_type_encoding = r.encoding if r.encoding != 'ISO-8859-1' else None
```

```
# Step 2: Parse the html content
soup = BeautifulSoup(r.content, 'html.parser', from_encoding=content_type_encoding)
#print(soup.prettify()) # print the parsed data of html
```

```
tables = soup.find_all('table', class_='datatable')
```

After the data is retrieved and convert into pandas dataframe, columns's name & indexes were translated to English.

```
#Columns & Index were Japanese, I changed all of them to English for readers convenience.
df0newcolumns = ['Total_Household', 'Ave_People_in_Household', 'Total_Population', 'Man', 'Women', 'Population_compared_to_previous_Investigatior
df0.columns = df0newcolumns
df0newindex = ['1955_Oct', '1960-Oct', '1965_Oct', '1970_Oct', '1975_Oct', '1980_Oct', '1985_Oct', '1990_Oct', '1995_Oct', '2000_Oct',
               '2005_Oct', '2010_Oct', '2015_Oct', '2018_Dec', '2020_Jan']
df0.index = df0newindex
```

Table 1, 2, 3, 4-1 & 6 will be used for investigating to choose the most most atrattive city.
(*please refer to the source code for more detail*)

TABLE 1: PUPULATION AND HOUSEHOLD TRENDS

	Total_Household	Ave_People_in_Household	Total_Population	Man	Women	Population_compared_to_previous_Investigation	Percentage_compared_to_previous_Investigation	Density_(People/kmsquare)
1955_Oct	382315	5.39	2064037	1006093	1057944	24619	1.2	338.9
1960-Oct	409465	5.03	2047024	1000184	1046840	-17013	-0.8	336.2
1965_Oct	447871	4.55	2056154	1007852	1048302	9130	0.4	337.7
1970_Oct	508537	4.16	2143551	1054003	1089548	87397	4.3	352.1
1975_Oct	590131	3.92	2342198	1159707	1182491	198647	9.3	384.6
1980_Oct	692855	3.66	2558007	1272533	1285474	215809	9.2	419.9
1985_Oct	758085	3.56	2725005	1357963	1367042	166998	6.5	447.1
1990_Oct	833634	3.39	2845382	1419117	1426265	120377	4.4	467.0
1995_Oct	922745	3.17	2955530	1476437	1479093	110148	3.9	485.0
2000_Oct	985829	2.99	2985676	1488340	1497336	30146	1.0	489.8
2005_Oct	1032476	2.84	2975167	1479941	1495226	-10509	-0.4	488.1
2010_Oct	1088411	2.68	2969770	1479779	1489991	-5397	-0.2	487.2
2015_Oct	1124349	2.55	2916976	1453594	1463382	-52794	-1.8	478.4
2018_Dec	1175302	2.44	2867267	1432226	1435041	-870	-0.0	470.2
2020_Jan	1175894	2.44	2866325	1431725	1434600	-942	-0.0	470.1

TABLE 6: CHANGES IN POPULATION OVER THE LAST YEAR

	2018_Dec	2019_Jan	2019_Feb	2019-Mar	2019-Apr	2019-May	2019-Jun	2019-Jul	2019-Aug	2019-Sep	2019-Oct	2019-Nov	2019-Dec	Total2019
Tsukuba-shi	213	253	102	-303	1384	350	174	215	138	713	437	174	210	3847
Moriya-shi	101	-2	86	11	137	89	6	90	72	80	129	112	54	864
Ami-cho	-19	-20	51	-26	103	-12	-12	41	37	17	44	-6	10	227
Tsukubamirai-shi	-5	27	17	-12	68	75	12	9	8	-9	0	25	-15	205
Kamisu-shi	-38	-26	-13	-133	96	103	21	50	-10	-3	-10	31	37	143
Tokai-mura	24	23	-10	-84	5	13	39	45	-27	16	13	-19	-8	6
Goka-cho	9	-1	1	-32	-14	-5	-8	12	-5	2	-16	-25	-10	-101
Sakai-cho	-38	-27	-6	-49	2	-32	-7	-17	8	-29	42	-3	-14	-132
Ushiku-shi	-25	-3	27	-93	70	-47	-12	-29	-8	-33	33	-35	-46	-176
Oarai-cho	-19	-34	-43	-6	-10	5	-24	-23	-31	-30	-10	1	4	-201
Kawachi-cho	-14	-22	-15	-24	-31	-5	-19	-21	-6	-23	-6	-21	-8	-201
Yachio-cho	-8	-17	-47	-70	-8	-17	-20	-15	14	9	3	-41	4	-205
Kashima-shi	-39	-57	12	-147	68	48	-3	-57	39	18	-31	-44	-53	-207
Miho-mura	-39	-23	-34	-84	-51	-19	1	15	-31	-17	-15	-15	11	-262
Tone-cho	18	-9	-41	-8	-23	19	-43	-16	-47	-31	21	-40	-45	-263

3.2. Ibaraki-prefecture population Data Visualization

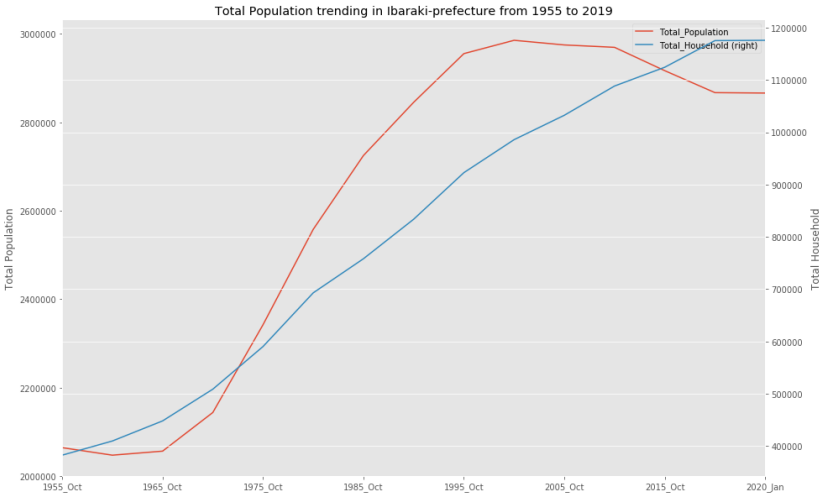
"matplotlib" was used to visualize the data in data frames.

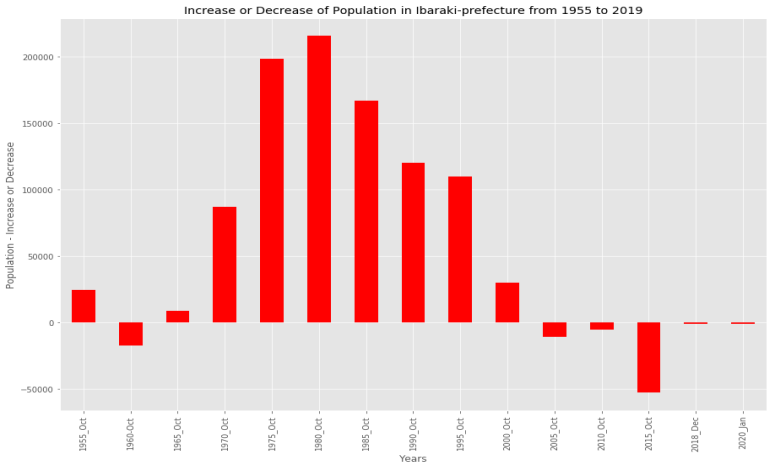
```
# we are using the inline backend
%matplotlib inline
import matplotlib as mpl
from mpl_toolkits.axes_grid1 import host_subplot
import matplotlib.pyplot as plt
print ('Matplotlib version: ', mpl.__version__) # >= 2.0.0
```

```
Matplotlib version:  3.0.2
```

```
print(plt.style.available)
mpl.style.use(['ggplot']) # optional: for ggplot-like style
```

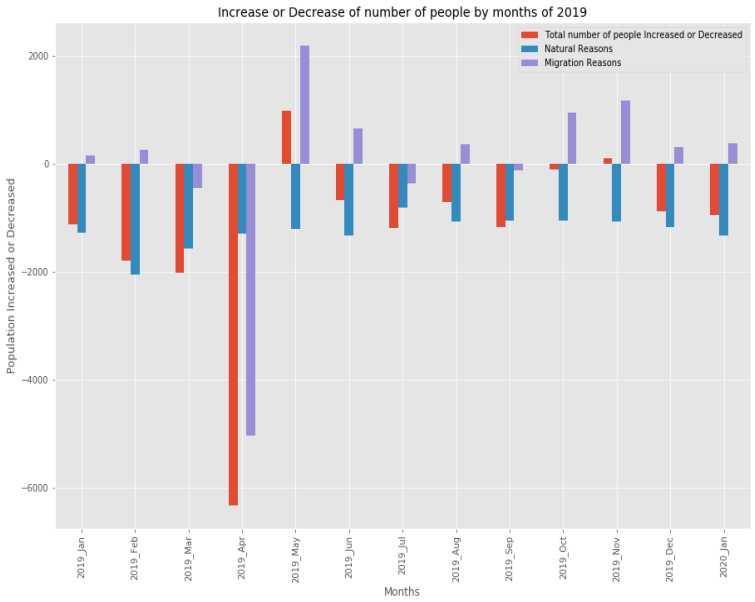
- A line plot was created to show the trending in Total Population. Looking in the trending of Total Population from 1955 to 2020, we see that the total population in Ibaraki got peak in 2000, after that the population is decreasing lightly and get stable from 2015. May come migration helped covering the decreasing trend caused by low birth rate.





- A bar graph of Increase & Decrease in population from 1955 to 2020 was created for more details.

- Looking in the data of 2019, let's see how population changing during 1 years.

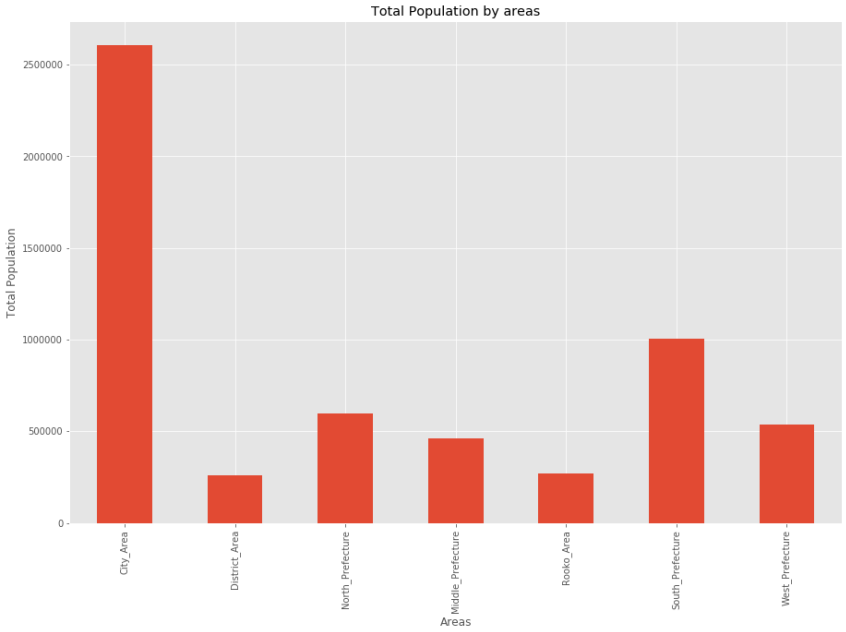


We can see that:

+ In every month of the years, population decreased by Natural Reasons. We can conclude that the birth rate is low and cannot cover the number people dead every month.

+ Almost months in the years, people coming to Ibaraki help the population increasing. Only in March & April when starting of fiscal year in Japan, a huge number of people get out may cause by the transformation order in their company.

+ It's looks like Ibaraki seem quite attractive for people to come for business.



- A bar graph was made to compare population in each area in Ibaraki.

+ Cities areas are more crowded than District areas.

+ South area is the most crowded area in the prefecture.

⇨ The coffee shop should be open in a city in South area.

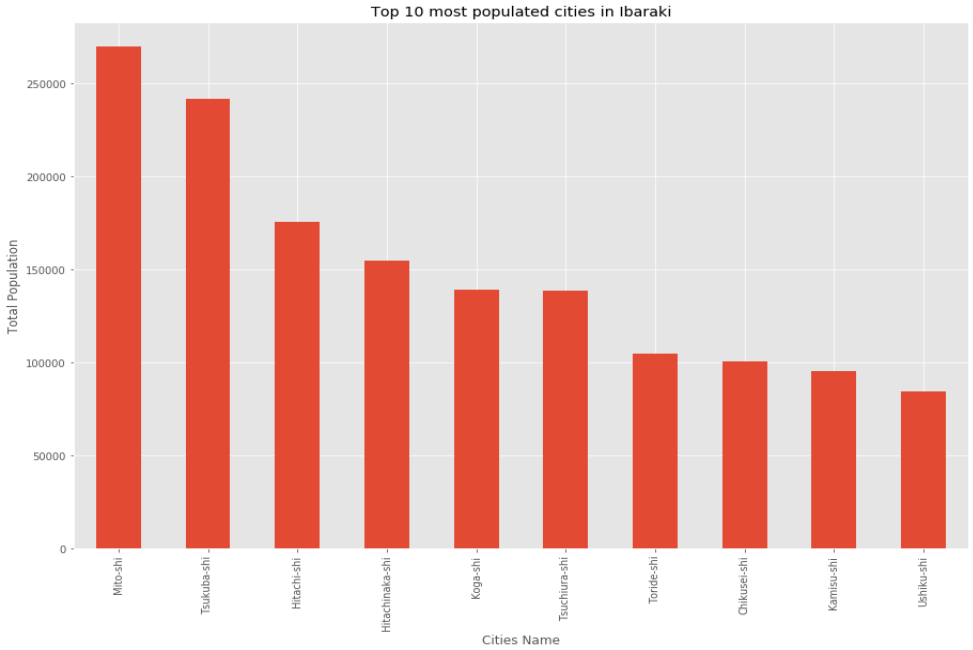
⇨ This also meet the criterias defined in business problem.

- One of the criteria is "The coffee shop should be located in city which in *top populated cities in Ibaraki*". Writer implemented a "sort" function to table 4-1 (df3) to creates a list of top 10 populated cities. And the target would be in this top 10 one.

```
In [26]: # Sort dataframe by Total_Population
         df3_sorted = df3.sort_values(['Total_Population'], ascending=False, axis=0)
         df3_sorted.head(10)
```

Out[26]:

	Total_Household	Total_Population	Man	Women	Increase_or_Decrease	Natural_Reasons	Birth	Dead	Migration_Reasons	In	Out
Mito-shi	122398	269763	132189	137574	-51	-68	155	223	17	593	576
Tsukuba-shi	107915	241808	122187	119621	210	52	214	162	158	984	826
Hitachi-shi	78163	175635	87549	88086	-158	-96	70	166	-62	284	346
Hitachinaka-shi	64351	155045	78367	76678	-50	-40	91	131	-10	348	358
Koga-shi	55670	139107	69691	69416	-46	-63	75	138	17	495	478
Tsuchiura-shi	60369	138557	69126	69431	14	-60	79	139	74	480	406
Toride-shi	45323	104611	51354	53257	6	-51	40	91	57	536	479
Chikusei-shi	37290	100658	49916	50742	-97	-46	60	106	-51	164	215
Kamisu-shi	40279	95428	49402	46026	37	-7	64	71	44	263	219
Ushiku-shi	34709	84589	41709	42880	-46	-23	33	56	-23	241	264



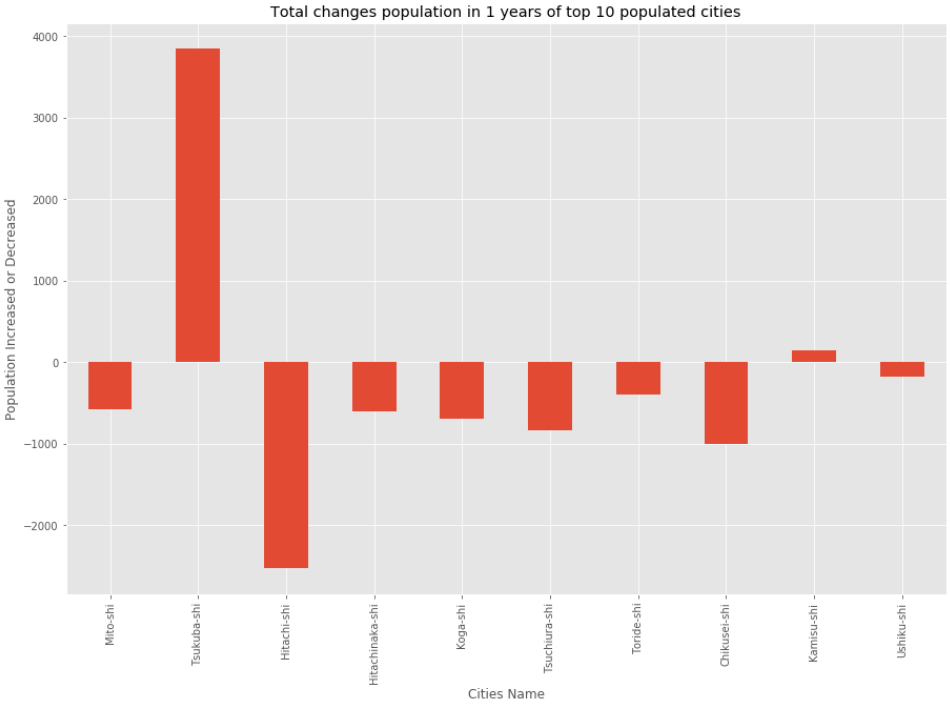
- To see how population changed in top 10 cities in 2019, writer performed a merge function between top 10 of table 4-1 (df3) & table 6 (df6) to create a new data frame.

```
In [28]: # Creat a dataframe of changes in population in 2019 of top 10 populated cities
df6_df3_top10 = pd.merge(df3_sorted_top10, df6, left_index=True, right_index=True)
df6_df3_top10.drop(['Total_Population'], axis=1, inplace=True)
df6_df3_top10
```

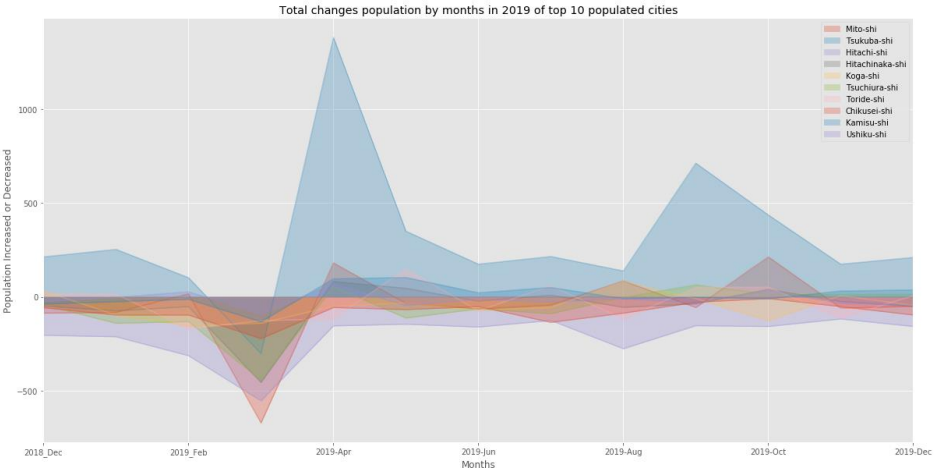
Out[28]:

	2018_Dec	2019_Jan	2019_Feb	2019-Mar	2019-Apr	2019-May	2019-Jun	2019-Jul	2019-Aug	2019-Sep	2019-Oct	2019-Nov	2019-Dec	Total2019
Mito-shi	-87	-83	16	-673	181	-35	-65	-44	85	-57	213	-60	-51	-573
Tsukuba-shi	213	253	102	-303	1384	350	174	215	138	713	437	174	210	3847
Hitachi-shi	-205	-213	-314	-555	-155	-146	-161	-126	-277	-154	-158	-118	-158	-2535
Hitachinaka-shi	-49	-75	-52	-457	82	45	-23	9	-56	-37	40	-25	-50	-599
Koga-shi	30	-97	-156	-141	-50	-33	-72	-58	83	-10	-127	6	-46	-701
Tsuchiura-shi	-43	-141	-132	-446	53	-113	-64	-90	-3	64	13	13	14	-832
Toride-shi	22	14	-171	-128	-117	149	-75	59	-118	51	52	-114	6	-392
Chikusei-shi	-58	-97	-97	-223	-57	-68	-52	-136	-87	-29	-9	-52	-97	-1004
Kamisu-shi	-38	-26	-13	-133	96	103	21	50	-10	-3	-10	31	37	143
Ushiku-shi	-25	-3	27	-93	70	-47	-12	-29	-8	-33	33	-35	-46	-176

This data frame was visualized to compared the changing in 1 years and each month of the year.



We can see that there are only 2 cities have population increased during 2019. Even in the impact of aging population, Tsukuba-shi keeps a very significant Increase during almost months of the year. From that, we can understand that many people come to this city for business last years. At this time, writer can choose Tsukuba-shi as the selection for more investigation to select the best place for open a coffee shop.

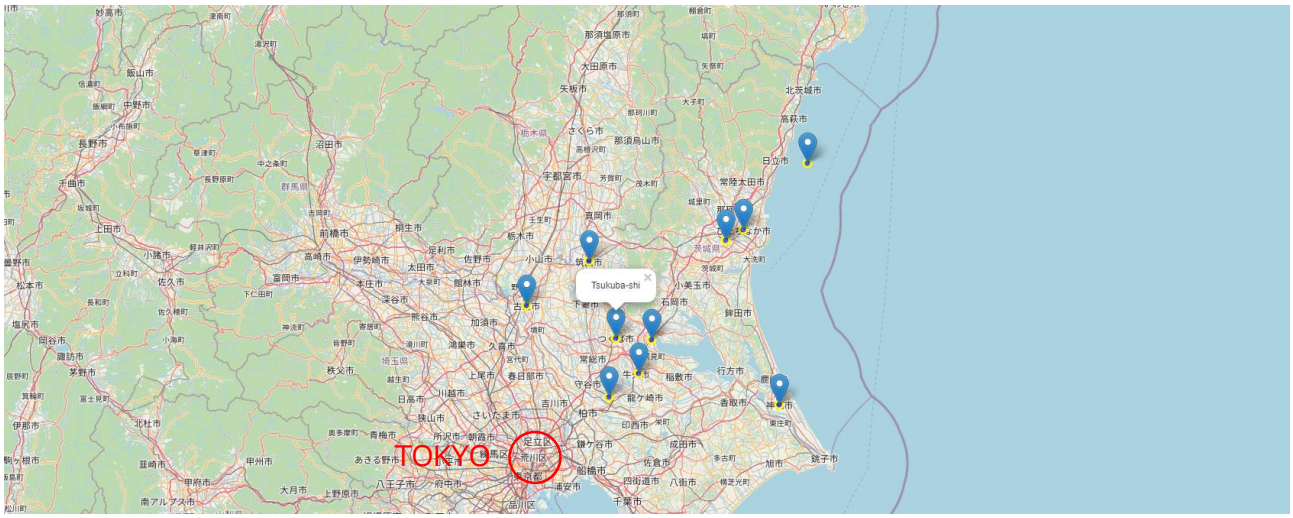


3.3. Mapping top 10 populated cities to see geographical positions

- Folium library was installed to created map of top 10 cities.
- OpenCageGeocode & API from https://opencagedata.com to get the geocode (latitude & longtitude) data of top 10 cities

	2018_Dec	2019_Jan	2019_Feb	2019-Mar	2019-Apr	2019-May	2019-Jun	2019-Jul	2019-Aug	2019-Sep	2019-Oct	2019-Nov	2019-Dec	Total2019	Latitude	Longtitude
Mito-shi	-87	-83	16	-673	181	-35	-65	-44	85	-57	213	-60	-51	-573	36.365779	140.471393
Tsukuba-shi	213	253	102	-303	1384	350	174	215	138	713	437	174	210	3847	36.083388	140.076510
Hitachi-shi	-205	-213	-314	-555	-155	-146	-161	-126	-277	-154	-158	-118	-158	-2535	36.589084	140.763958
Hitachinaka-shi	-49	-75	-52	-457	82	45	-23	9	-56	-37	40	-25	-50	-599	36.396124	140.535340
Koga-shi	30	-97	-156	-141	-50	-33	-72	-58	83	-10	-127	6	-46	-701	36.178025	139.755364
Tsuchiura-shi	-43	-141	-132	-446	53	-113	-64	-90	-3	64	13	13	14	-832	36.078630	140.204593
Toride-shi	22	14	-171	-128	-117	149	-75	59	-118	51	52	-114	6	-392	35.911532	140.050178
Chikusei-shi	-58	-97	-97	-223	-57	-68	-52	-136	-87	-29	-9	-52	-97	-1004	36.305194	139.979090
Kamisu-shi	-38	-26	-13	-133	96	103	21	50	-10	-3	-10	31	37	143	35.889900	140.664575
Ushiku-shi	-25	-3	27	-93	70	-47	-12	-29	-8	-33	33	-35	-46	-176	35.980022	140.158128

- Mapping the data:



3.4. Middle result

After data visualization & mapping, the writer clarifies that Tsukuba-shi is the most attractive city in Tsukuba. It meets all the criteria defined such as:

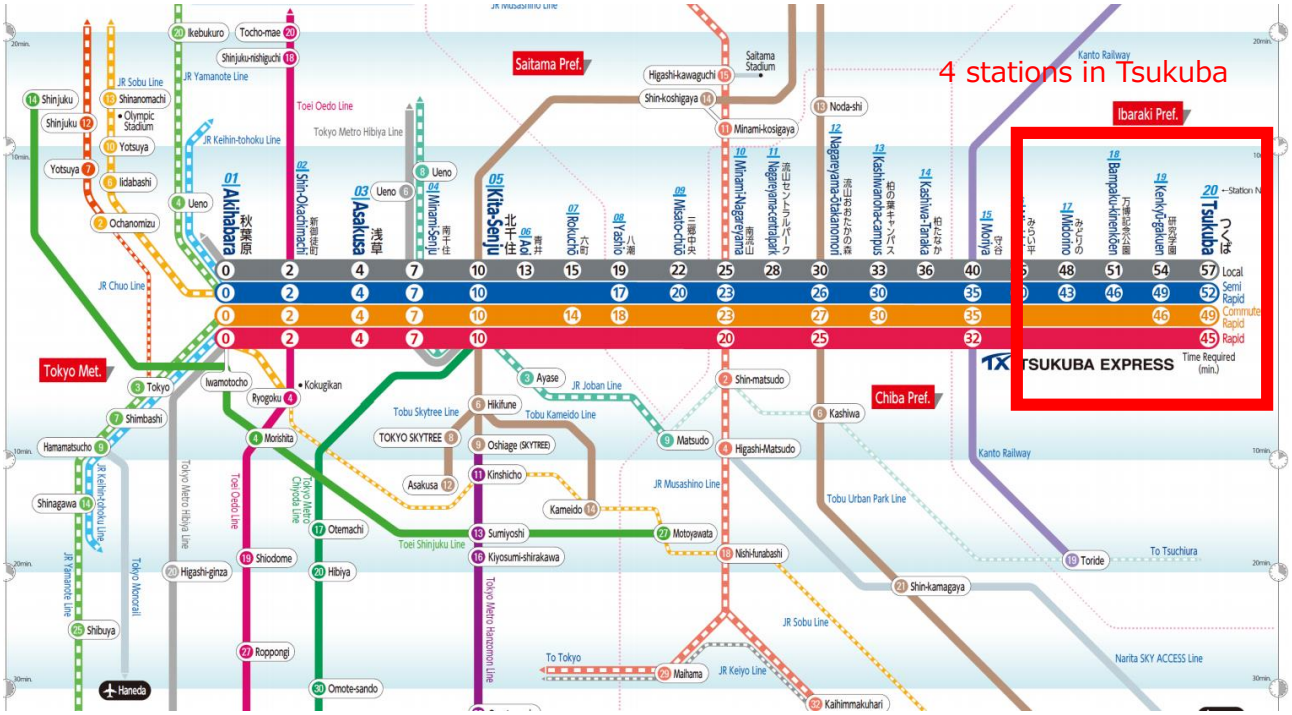
- The coffee shop should be located in city which in *top populated cities in Ibaraki*.
- The location should be *near Tokyo* so South of Ibaraki is preferable.
- The city's *population should be stable or increasing* (many people coming to this city to cover the natural decrease cause by low birth rate & aging).

Now it's time to look specific position meets the 2 last criteria.

- The location should be *near some train stations*.
- There are *not too many competitors* in the area.

4. Exploring the neighborhood in Tsukuba-shi

After a few searching, writer know that there are 4 train station located in Tsukuba: Tsukuba-station, Kenkyugakuen-station, Banpaku Kinenkoen-station, Midorino-station.



https://www.mir.co.jp/en/howto/route_map/pdf/route_map.pdf

- OpenCageGeocode & API from https://opencagedata.com to get the geocode (latitude & longtitude) data of the stations

Get the latitude & longtitude data of Tsukuba-station

```
[35]: station1 = 'Tsukuba-station, Tsukuba-shi, Ibaraki'
      results1 = geocoder.geocode(station1)
      lat1 = results1[0]['geometry']['lat']
      lng1 = results1[0]['geometry']['lng']
      print (lat1, lng1)
```

```
36.0825835 140.111247
```

Get the latitude & longtitude data of Kenkyugakuen-station

```
[36]: station2 = 'Kenkyugakuen-station, Tsukuba-shi, Ibaraki'
      results2 = geocoder.geocode(station2)
      lat2 = results2[0]['geometry']['lat']
      lng2 = results2[0]['geometry']['lng']
      print (lat2, lng2)
```

```
36.0844107 140.0863296
```

Get the latitude & longtitude data of Banpaku Kinenkoen-station

```
[37]: station3 = 'Banpaku Kinenkoen-station, Tsukuba-shi, Ibaraki'
      results3 = geocoder.geocode(station3)
      lat3 = results3[0]['geometry']['lat']
      lng3 = results3[0]['geometry']['lng']
      print (lat3, lng3)
```

```
36.10532 140.08174
```

Get the latitude & longtitude data of Midorino-station

```
[38]: station4 = 'Midorino-station, Tsukuba-shi, Ibaraki'
      results4 = geocoder.geocode(station4)
      lat4 = results4[0]['geometry']['lat']
      lng4 = results4[0]['geometry']['lng']
      print (lat4, lng4)
```

```
36.0298606 140.0563392
```

- Foursquare API was used to get the top 100 venues that are near each stations within a radius of 2km

```
[40]: # Format url
      LIMIT = 100
      radius = 2000
      url = 'https://api.foursquare.com/v2/venues/explore?client_id={}&client_secret={}&ll={},{}&v={}&radius={}&limit={}'.format(CLIENT_ID, CLIENT_SECRET, lat1, lng1, VERSION, radius, LIMIT)
      url
```

```
[40]: 'https://api.foursquare.com/v2/venues/explore?client_id=2ALIJHDF3E3IUXS2XEHT14JR5CV1J0Z214PQNAEAFX1MR5G3&client_secret=QQDZYRZRQX14JYWX2GNI5GX2XU0HT0I2SQ1HWAGXNWHVNR1T&ll=36.0825835,140.1
t=100'
```

```
[41]: results = requests.get(url).json()
```

```
# function that extracts the category of the venue
def get_category_type(row):
    try:
        categories_list = row['categories']
    except:
        categories_list = row['venue.categories']

    if len(categories_list) == 0:
        return None
    else:
        return categories_list[0]['name']
```

Get a list of venues

```
venues = results['response']['groups'][0]['items']

nearby_venues = json_normalize(venues) # flatten JSON

# filter columns
filtered_columns = ['venue.name', 'venue.categories', 'venue.location.lat', 'venue.location.lng']
nearby_venues =nearby_venues.loc[:, filtered_columns]

# filter the category for each row
nearby_venues['venue.categories'] = nearby_venues.apply(get_category_type, axis=1)

# clean columns
nearby_venues.columns = [col.split(".")[-1] for col in nearby_venues.columns]

nearby_venues
```

Result

	name	categories	lat	lng
0	SAZA COFFEE つくば駅前店	Coffee Shop	36.081613	140.112569
1	Ton Q (とんQ)	Tonkatsu Restaurant	36.077033	140.110646
2	Starbucks	Coffee Shop	36.081894	140.111416
3	Tsukuba Expo Center (つくばエキスポセンター)	Science Museum	36.086025	140.110567
4	La Cote d'Azur (コート・ダジュール本店)	Dessert Shop	36.076966	140.113734
...	...	...	...	...
95	珍来 千現店	Noodle House	36.073195	140.125970
96	Tsukuba Creo Square Q't (つくばクレオスクエア Q't)	Shopping Mall	36.081766	140.111592
97	Saizeriya (サイゼリヤ)	Italian Restaurant	36.082084	140.111194
98	Lawson (ローソン つくば駅バスターミナル店)	Convenience Store	36.081788	140.113088
99	FamilyMart (ファミリーマート)	Convenience Store	36.091717	140.106250

100 rows × 4 columns

- The result was grouped to see the number of venues by categories.

Group the venues by categories and create a dataframe

```
[44]: group = nearby_venues.groupby('categories').count()
      venues_tsukuba = group.drop(['lat', 'lng'], axis=1)
      venues_tsukuba
```

[44]:

categories	name
Arcade	2
BBQ Joint	3
Bakery	2
Bar	1
Bookstore	1
Botanical Garden	1
Burger Joint	1
Café	4
Chinese Restaurant	2
Clothing Store	2
Coffee Shop	4
Concert Hall	1
Convenience Store	10
Dessert Shop	1

- After the process was implanted for the others station, a merge function was executed to give the number of coffee shop venues near each stations.

3.1. Number of venues by categories comparing among 4 stations

```
[61]: # Merge all dataframes & Rename the columns
merge_venues = venues_tsukuba.merge(venues_kenkyugakuen, left_index=True, right_index=True).merge(venues_banpakukinenkoen, left_index=True, right_index=True).merge(venues_midorino, left_index=True, right_index=True)
merge_venues_newcolums = ['Tsukuba-station', 'Kenkyugakuen-station', 'Banpaku_Kinenkoen-station', 'Midorino-station']
merge_venues.columns = merge_venues_newcolums
```

```
[62]: # Though only "Cafe" % "Coffee Shop" seem to be competitors, remove others index & some these 2 rows.
coffe_venues = merge_venues[:2]
coffe_venues = coffe_venues.sum(axis=0)
coffe_venues
```

```
[62]: Tsukuba-station             8
Kenkyugakuen-station         9
Banpaku_Kinenkoen-station    4
Midorino-station             2
dtype: int64
```

Conclusion: Midorino-Station may the best place to open a new Coffee Shop!

5. Discussion

After the investigation from population of Ibaraki-prefecture the writer understands that among the cities in prefecture, Tsukuba-shi is currently the most attractive city because people migrate to this city can even cover the population decrease caused by low birth rate.

After city selection, using the foursquare API help writer know which station is the good place to open business in the view point of number of competitors.

In this investigation project, cluster algorithm or current trending information (use comment & accessing data) was not used. By using these, writer may get more information to help him go on with final decision.

Writer also didn't concern about the view point of budget in this investigation. So, in the future, some more research on the house price in Ibaraki may help more.

6. Conclusion

A data science approaching helps the writer to decide where is the best place to open a new Coffee Shop in Ibaraki-prefecture. It's the neighborhood of Midorino-Station in Tsukuba-shi, where meets all the criteria defined before the research:

- The coffee shop should be located in city which in *top populated cities in Ibaraki*.
- The location should be *near Tokyo* so South of Ibaraki is preferrable.
- The city's *population should be stable or increasing* (many people coming to this city to cover the natural decrease cause by low birth rate & aging).
- The location should be *near some train stations*.
- There are *not too many competitors* in the area.