

# **Geopandas2**

**Data analysis with GeoPandas**

Kunal Khurana

2023-01-07

# Table of contents

Geocoding . . . . .	2
GeoDataFrame Table joins . . . . .	4
Spatial join . . . . .	6
Potentially suitable areas for starbucks . . . . .	8

```
import geopandas as gpd
import pandas as pd
import numpy as np
from shapely.geometry import LineString

# for interactive maps
import folium
from folium import Choropleth, Circle, Marker
from folium.plugins import HeatMap, MarkerCluster
import math

# for Geocoding
from geopy.geocoders import Nominatim
```

## Geocoding

process of converting name of a place or an address to a location on a Map

example location=[54, 15];

Latitude coordinate, 54 (ranges from -90 to 90 degrees); positive value - northern hemisphere

Longitude coordinate, 15 (ranges from -180 to 180 degrees; positive value- eastern hemisphere)

```
geolocator = Nominatim (user_agent = 'Kunal Khurana')
location = geolocator.geocode('Taj Mahal')
```

```
print(location.point)
print(location.address)
```

27 10m 30.027s N, 78 2m 31.5645s E  
Taj Mahal, Taj East Gate Road, Taj Ganj, Agra, Agra District, Uttar Pradesh, 282004, India

```
point = location.point
print("Latitude:", point.latitude)
print("Longitude:", point.longitude)
```

Latitude: 27.1750075  
Longitude: 78.04210126365584

```
# top universities
universities = pd.read_csv('data_for_all_courses\\top_universities.csv')
universities.head()
```

	Name
0	University of Oxford
1	University of Cambridge
2	Imperial College London
3	ETH Zurich
4	UCL

```
# using lambda function to apply geocoder to every row in DataFrame
def my_geocoder(row):
    #use try/except where geocoding is unsuccessful
    try:
        point = geolocator.geocode(row).point
        return pd.Series({'Latitude': point.latitude, 'Longitude': point.longitude})
    except:
        return None

universities[['Latitude', 'Longitude']] = universities.apply(lambda x: my_geocoder(x['Name']))

print(f"{{(1-sum(np.isnan(universities['Latitude'])) / len(universities)) * 100}}% of addresses have valid coordinates")
```

95.0% of addresses were geocoded!

```
# drop universities that were not successfully geocoded
universities = universities.dropna(subset= ['Latitude']).copy()
universities = gpd.GeoDataFrame(universities,
                                geometry = gpd.points_from_xy (universities.Longitude, universities.Latitude),
                                crs = universities.crs)
universities.head()
```

	Name	Latitude	Longitude	geometry
0	University of Oxford	51.758708	-1.255668	POINT (-1.25567 51.75871)
1	University of Cambridge	52.210946	0.092005	POINT (0.09200 52.21095)
2	Imperial College London	51.498959	-0.175641	POINT (-0.17564 51.49896)
3	ETH Zurich	47.413218	8.537491	POINT (8.53749 47.41322)
4	UCL	51.521785	-0.135151	POINT (-0.13515 51.52179)

```
# create a map
m = folium.Map(location = [20, 79], titles = 'openstreetmap', zoom_start=2)

# add points
for idx, row in universities.iterrows():
    Marker([row['Latitude'], row['Longitude']], popup= row['Name']).add_to(m)

# display
m
```

<folium.folium.Map at 0x1d8c284c110>

## GeoDataFrame Table joins

```
world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
europe = world.loc[world.continent == "Europe"].reset_index(drop = True)
```

C:\Users\Khurana\_Kunal\AppData\Local\Temp\ipykernel\_13672\2106073632.py:1: FutureWarning: The `loc` accessor is deprecated. Use `loc` instead.  
world = gpd.read\_file(gpd.datasets.get\_path('naturalearth\_lowres'))

```
europa.head()
```

	pop_est	continent	name	iso_a3	gdp_md_est	geometry
0	144373535.0	Europe	Russia	RUS	1699876	MULTIPOLYGON (((180.00000 71.51571, 180.
1	5347896.0	Europe	Norway	NOR	403336	MULTIPOLYGON (((15.14282 79.67431, 15.5
2	67059887.0	Europe	France	FRA	2715518	MULTIPOLYGON (((-51.65780 4.15623, -52.2
3	10285453.0	Europe	Sweden	SWE	530883	POLYGON ((11.02737 58.85615, 11.46827 59.
4	9466856.0	Europe	Belarus	BLR	63080	POLYGON ((28.17671 56.16913, 29.22951 55.

```
europe_stats = europe[['name', 'pop_est', 'gdp_md_est']]
europe_boundaries = europe[['name', 'geometry']]
```

```
europa_stats.head()
```

	name	pop_est	gdp_md_est
0	Russia	144373535.0	1699876
1	Norway	5347896.0	403336
2	France	67059887.0	2715518
3	Sweden	10285453.0	530883
4	Belarus	9466856.0	63080

```
europa_borders.head()
```

	name	geometry
0	Russia	MULTIPOLYGON (((180.00000 71.51571, 180.00000 ...
1	Norway	MULTIPOLYGON (((15.14282 79.67431, 15.52255 80...
2	France	MULTIPOLYGON (((-51.65780 4.15623, -52.24934 3...
3	Sweden	POLYGON ((11.02737 58.85615, 11.46827 59.43239...
4	Belarus	POLYGON ((28.17671 56.16913, 29.22951 55.91834...

```
# use 'name' to merge europe_boundaries and europe_stats
```

```
europe2 = europe_boundaries.merge(europe_stats, on= 'name')
europe2.head()
```

	name	geometry	pop_est	gdp_md_est
0	Russia	MULTIPOLYGON (((180.00000 71.51571, 180.00000 ...	144373535.0	1699876
1	Norway	MULTIPOLYGON (((15.14282 79.67431, 15.52255 80...	5347896.0	403336
2	France	MULTIPOLYGON (((-51.65780 4.15623, -52.24934 3...	67059887.0	2715518
3	Sweden	POLYGON ((11.02737 58.85615, 11.46827 59.43239...	10285453.0	530883
4	Belarus	POLYGON ((28.17671 56.16913, 29.22951 55.91834...	9466856.0	63080

## Spatial join

joining data based on geometry

```
# match universities to countries

european_universities = gpd.sjoin(universities, europe2)
```

C:\Users\Khurana\_Kunal\AppData\Local\Temp\ipykernel\_13672\546710102.py:3: UserWarning: CRS mismatch. Use `to\_crs()` to reproject one of the input geometries to match the CRS of the other.

Left CRS: None

Right CRS: EPSG:4326

```
european_universities = gpd.sjoin(universities, europe2)
```

```
# check the crs for europe2
europe2.crs
```

```
<Geographic 2D CRS: EPSG:4326>
Name: WGS 84
Axis Info [ellipsoidal]:
- Lat[north]: Geodetic latitude (degree)
- Lon[east]: Geodetic longitude (degree)
Area of Use:
- name: World.
- bounds: (-180.0, -90.0, 180.0, 90.0)
Datum: World Geodetic System 1984 ensemble
- Ellipsoid: WGS 84
- Prime Meridian: Greenwich
```

```
# lets provide same crs value to universities
universities = universities.set_crs('epsg: 4326')

european_unviersities = gpd.sjoin(universities, europe2)
european_universities.head()
```

	Name	Latitude	Longitude	geometry
0	University of Oxford	51.758708	-1.255668	POINT (-1.25567 51.75871)
1	University of Cambridge	52.210946	0.092005	POINT (0.09200 52.21095)
2	Imperial College London	51.498959	-0.175641	POINT (-0.17564 51.49896)
4	UCL	51.521785	-0.135151	POINT (-0.13515 51.52179)
5	London School of Economics and Political Science	51.514261	-0.116734	POINT (-0.11673 51.51426)

```
# investigate the result
print (f"We located {len(universities)} universities.")

print (f"Out of 95, {len(european_universities)} "
      f"of the universities were located in Europe in "
      f"{len(european_universities.name.unique())}"
      f" different countries." )
```

We located 95 universities.

Out of 95, 89 of the universities were located in Europe in 15 different countries.

```
european_universities.head()
```

	Name	Latitude	Longitude	geometry
0	University of Oxford	51.758708	-1.255668	POINT (-1.25567 51.75871)
1	University of Cambridge	52.210946	0.092005	POINT (0.09200 52.21095)
2	Imperial College London	51.498959	-0.175641	POINT (-0.17564 51.49896)
4	UCL	51.521785	-0.135151	POINT (-0.13515 51.52179)
5	London School of Economics and Political Science	51.514261	-0.116734	POINT (-0.11673 51.51426)

## Potentially suitable areas for starbucks

```
# for maps visualization
def embed_map(m, file_name):
    from IPython.display import IFrame
    m.save(file_name)
    return IFrame(file_name, width='100%', height='500px')

starbucks = pd.read_csv('data_for_all_courses\starbucks_locations.csv')
starbucks.head()
```

	Store Number	Store Name	Address	City
0	10429-100710	Palmdale & Hwy 395	14136 US Hwy 395 Adelanto CA	Adelanto
1	635-352	Kanan & Thousand Oaks	5827 Kanan Road Agoura CA	Agoura Hills
2	74510-27669	Vons-Agoura Hills #2001	5671 Kanan Rd. Agoura Hills CA	Agoura Hills
3	29839-255026	Target Anaheim T-0677	8148 E SANTA ANA CANYON ROAD ANAHEIM CA	ANAHEIM
4	23463-230284	Safeway - Alameda 3281	2600 5th Street Alameda CA	Alameda

```
# print individual missing values
print(f"The DataFrame starbucks has {starbucks.isnull().sum()} missing values")
```

```
The DataFrame starbucks has Store Number    0
Store Name    0
Address    0
City    0
Longitude    5
Latitude    5
dtype: int64 missing values
```

```
# printing missing rows
missing_value_rows = starbucks[starbucks.isnull().any(axis=1)]
print(missing_value_rows)
```

```
Store Number    Store Name \
153    5406-945    2224 Shattuck - Berkeley
154    570-512    Solano Ave
155    17877-164526    Safeway - Berkeley #691
```



```

156 19864-202264      Telegraph & Ashby
157   9217-9253      2128 Oxford St.

```

	Address	City	Longitude	Latitude
153	2224 Shattuck Avenue Berkeley CA	Berkeley	NaN	NaN
154	1799 Solano Avenue Berkeley CA	Berkeley	NaN	NaN
155	1444 Shattuck Place Berkeley CA	Berkeley	NaN	NaN
156	3001 Telegraph Avenue Berkeley CA	Berkeley	NaN	NaN
157	2128 Oxford Street Berkeley CA	Berkeley	NaN	NaN

```

# print total missing values
print(f"The DataFrame starbucks has {starbucks.isnull().sum().sum()} missing values")

```

The DataFrame starbucks has 10 missing values

```

# removing missing values
starbucks_clean = print (f"We are removing missing values from the starbucks "
                        f"{starbucks.dropna(inplace = True)} dataframe.")

```

We are removing missing values from the starbucks None dataframe.

```

# checking
print(starbucks.isnull().sum())

```

```

Store Number    0
Store Name      0
Address         0
City            0
Longitude       0
Latitude        0
dtype: int64

```

## inference

All the missing values are in Berkley city.

```
# create geocoder that adds latitude and longitude values

geolocator = Nominatim (user_agent = "Kunal Khurana ")

def my_geocoder(row):
    point = geolocator.geocode(row).point
    return pd.Series({'Latitude': point.latitude, 'Longitude': point.longitude})

berkley_locations = missing_value_rows.apply(lambda x: my_geocoder(x['Address']), axis = 1)
starbucks.update(berkley_locations)

starbucks.describe
```

```
<bound method NDFrame.describe of          Store Number          Store Name \
0      10429-100710      Palmdale & Hwy 395
1          635-352      Kanan & Thousand Oaks
2      74510-27669      Vons-Agoura Hills #2001
3      29839-255026      Target Anaheim T-0677
4      23463-230284      Safeway - Alameda 3281
...          ...          ...
2816  14071-108147  Hwy 20 & Tharp - Yuba City
2817   9974-98559  Yucaipa & Hampton, Yucaipa
2818  79654-108478      Vons - Yucaipa #1796
2819   6438-245084      Yucaipa & 6th
2820   6829-82142  Highway 62 & Warren Vista

          Address          City  Longitude \
0      14136 US Hwy 395 Adelanto CA      Adelanto      -117.40
1          5827 Kanan Road Agoura CA      Agoura      -118.76
2      5671 Kanan Rd. Agoura Hills CA  Agoura Hills      -118.76
3      8148 E SANTA ANA CANYON ROAD AHAHEIM CA      AHAHEIM      -117.75
4          2600 5th Street Alameda CA      Alameda      -122.28
...          ...          ...          ...
2816   1615 Colusa Hwy, Ste 100 Yuba City CA      Yuba City      -121.64
2817   31364 Yucaipa Blvd., A Yucaipa CA      Yucaipa      -117.12
2818   33644 YUCAIPA BLVD YUCAIPA CA      YUCAIPA      -117.07
2819   34050 Yucaipa Blvd., 200 Yucaipa CA      Yucaipa      -117.06
2820  57744 29 Palms Highway Yucca Valley CA  Yucca Valley      -116.40

Latitude
0      34.51
```

```

1      34.16
2      34.15
3      33.87
4      37.79
...      ...
2816    39.14
2817    34.03
2818    34.04
2819    34.03
2820    34.13

```

```
[2816 rows x 6 columns]>
```

```
starbucks.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
Index: 2816 entries, 0 to 2820
```

```
Data columns (total 6 columns):
```

#	Column	Non-Null Count	Dtype
0	Store Number	2816 non-null	object
1	Store Name	2816 non-null	object
2	Address	2816 non-null	object
3	City	2816 non-null	object
4	Longitude	2816 non-null	float64
5	Latitude	2816 non-null	float64

```
dtypes: float64(2), object(4)
```

```
memory usage: 154.0+ KB
```

```
# base map
```

```
m_base = folium.Map(location= [37.88, -122.26], tiles='openstreetmap', zoom_start = 12)
```

```
# add markers
```

```
for idx, row in starbucks[starbucks['City']=='Berkeley'].iterrows():
```

```
    Marker([row['Latitude'], row['Longitude']]).add_to(m_base)
```

```
# display
```

```
m_base
```

```
<folium.folium.Map at 0x1d8c3d7f410>
```

```
starbucks.head()
```

	Store Number	Store Name	Address	City
0	10429-100710	Palmdale & Hwy 395	14136 US Hwy 395 Adelanto CA	Adelanto
1	635-352	Kanan & Thousand Oaks	5827 Kanan Road Agoura CA	Agoura Hills
2	74510-27669	Vons-Agoura Hills #2001	5671 Kanan Rd. Agoura Hills CA	Agoura Hills
3	29839-255026	Target Anaheim T-0677	8148 E SANTA ANA CANYON ROAD ANAHEIM CA	Anaheim
4	23463-230284	Safeway - Alameda 3281	2600 5th Street Alameda CA	Alameda