Analyzing Geospatial Data with Python

A practical data analysis post with Python code.

Gustavo Santos

.

Published in

Towards Data Science

.

8 min read

.

Aug 19

Introduction

is one of my areas of interest. I find it fascinating how we can visualize data on a map and how — many times — the relationships between the data points present great insights real quickly.

I believe the applicability of this sub area of data science is pretty useful for any business, namely grocery stores, car rentals, logistics, real estate *etc*. In this post, we will go over a dataset from *AirBnb* for the city of Asheville, NC, in USA.

Side note: In that city lies one of the most amazing real estates in America, — *and I would dare to say in the world*. The property pertains to the Vanderbilt family and, during a long time, it was the largest private property in the country. Well, it is so worth a visit, but that's not the core subject here.

Stephanie Klepacki on Unsplash.

The datasets to be used in this exercise are the AirBnb rentals for the city of Asheville. They can be downloaded directly from this web site in http://insideairbnb.com/get-the-data, under the Creative Commons Attribution 4.0 International License.

Let's get to work.

Geospatial Data Science

The knowledge from this post is mostly from the book referred below (Applied Geospatial Data Science with Python, by David S. JORDAN). So let's begin importing some modules to our session.

```
import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt
import pysal
import splot
import re
import seaborn as sns
import folium
# For points map
import geoplot.crs as gcrs
import geoplot as gplt
```

Now notice that some of them might be new for you, as they are for me as well. If needed, use `pip install module_name` to install any package needed. In my case, `pysal` and `geoplot` are new to me, so they had to be installed.

Next, we will read the data from AirBnb.

```
# Open listings file
listings = pd.read_csv('/content/listings.csv',
                       usecols=['id', 'property_type', 'neighbourhood_cleansed',
                                'bedrooms', 'beds', 'bathrooms_text', 'price',
```

```
                                    'latitude','longitude'])
#listings.columns
listings.sample(4)
```

	id	neighbourhood_cleansed	latitude	longitude	property_type	bathrooms_text	bedrooms	beds	price
2176	54326499	28806	35.61798	-82.65003	Entire home	2 baths	3.0	3.0	$142.00
2945	853531231228833355	28806	35.57821	-82.62767	Entire rental unit	1 bath	1.0	1.0	$130.00
516	21682891	28803	35.54933	-82.50863	Entire home	2 baths	3.0	3.0	$254.00
2798	795911646107857978	28803	35.50759	-82.52592	Entire townhouse	1.5 baths	2.0	2.0	$86.00

if we code `listings.info()` , we will see that the `price` variable is as object. That is because of the dollar sign `$` mark before the numbers. Something easily corrected in Python.

```
# Correct Price to Float (Replace $ and , with nothing)
listings['price'] = (listings['price']
                     .replace("[$,]", "", regex=True)
                     .astype(float)
                     )
```

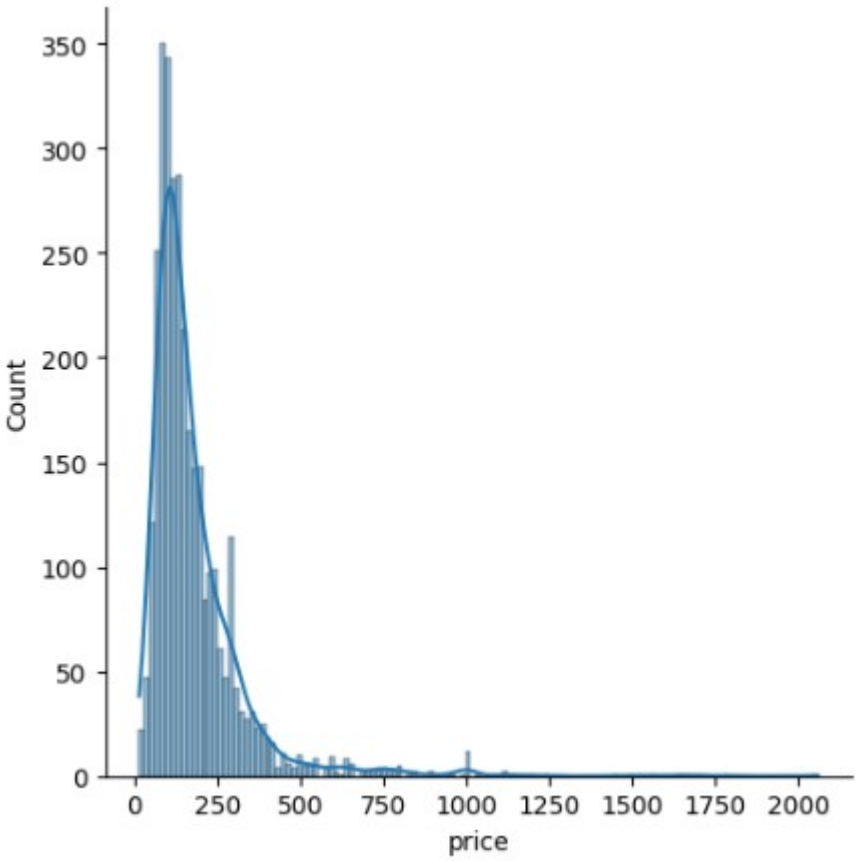
Exploratory Analysis

Now we can check the stats for that variable.

```
#price stats
listings.price.describe()
count    3239.000000
mean      179.771843
std       156.068212
min        14.000000
25%        95.000000
50%       135.000000
75%       212.500000
max      2059.000000
Name: price, dtype: float64
```

Interesting. Average of almost $180 dollars, but median on $135. It indicates that we might have some high values distorting the distribution, skewing it to the right. To make sure, we can check the price range.

```
# Check price range
sns.displot(listings['price'], kde=True);
```



As expected, we have the bulk of our data up until $500 or so. The rest is mostly outliers, with little counts. We can confirm that by running the code `np.quantile(listings['price'], 0.97)` and checking that the 97th percentile is $538 dollars.

Very well.

Now the next step is to start putting the data on maps. Let's visualize the data points and start drawing some insights. For that, first we need to convert the Pandas data frame to Geopandas.

```
# Convert Pandas df to Geopandas df
listings_gpd = gpd.GeoDataFrame(listings,

geometry=gpd.points_from_xy(listings.longitude, listings.latitude, crs=4326))
# Look at the geometry variable created
listings_gpd.head(2)
```

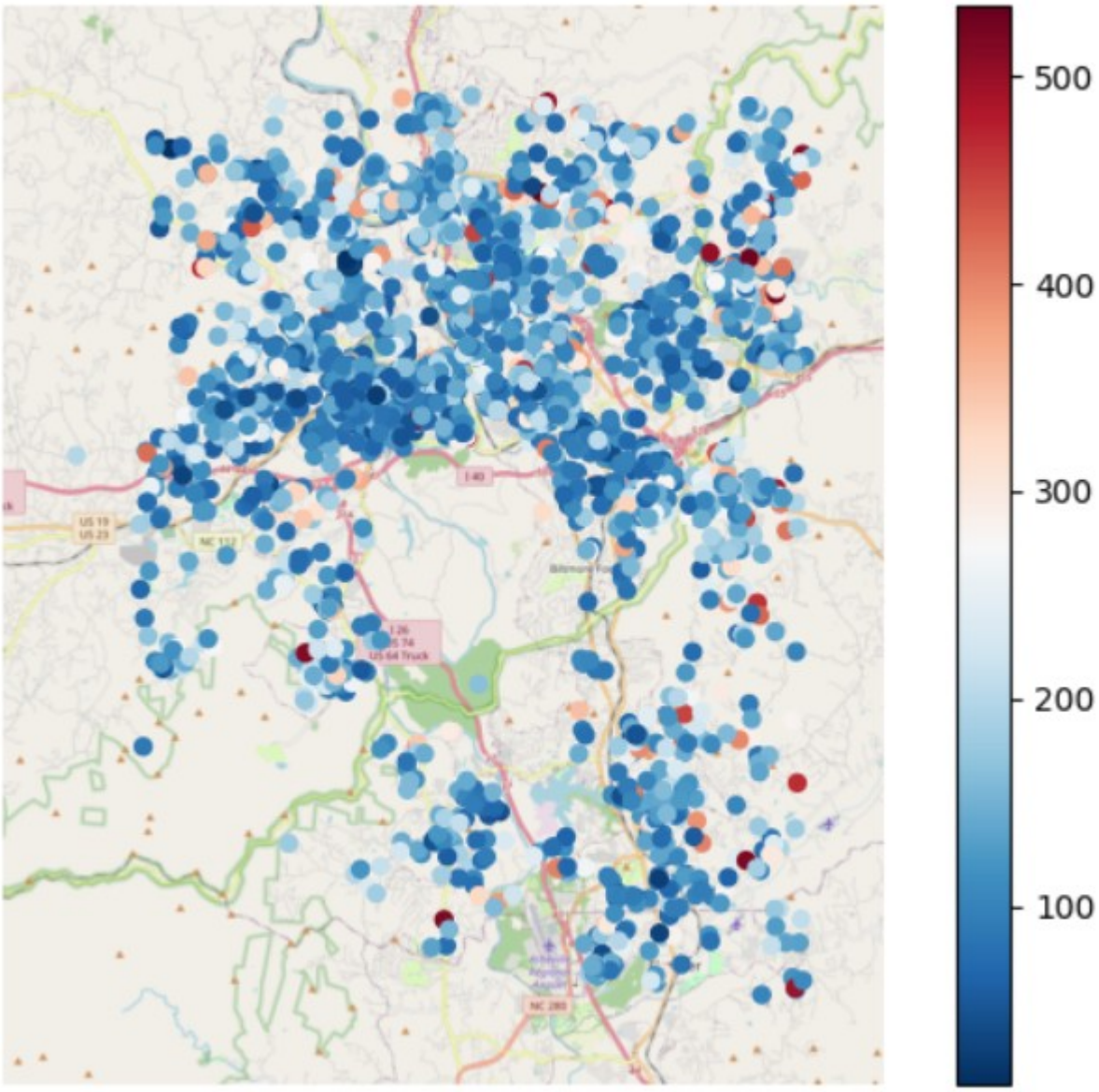
Looking at the dataset created, observe that it now brings a `geometry` variable.

	id	neighbourhood_cleansed	property_type	bedrooms	beds	bathrooms	latitude	longitude	price	geometry
0	108061	28801	Entire rental unit	1.0	1.0	1	35.60670	-82.55563	100.0	POINT (-82.55563 35.60670)
1	4394761	28801	Entire rental unit	2.0	1.0	1	35.61244	-82.55724	114.0	POINT (-82.55724 35.61244)

With that done, plotting the points is easy enough with geoplot. We are using only values under $538, so our colors look better distributed, otherwise the skew of the data would make a big purple blend of points.

```
# Points map using geoplot
ax = gplt.webmap(listings_gpd.query('price < 538'),
projection=gcrs.WebMercator())
```

```
gplt.pointplot(listings_gpd.query('price < 538'), ax=ax, hue= 'price',
legend=True);
```



Ok. From here we can start to see some nice things already:

- Most of the listings are (as expected) floating around the average
- There is a concentration of rental properties over the I-40 hwy.
- There is a good mixture of dark blue and light blue dots, so it shows that the prices are around 100 to 200 dollars.
- The appearances of red dots is very sparse, so it should no be very common to find too expensive rentals in Asheville.

Heatmap

Next, I believe we can go for creating heatmaps.

One option, as shown in JORDAN's book is using the geoplot. We can download the geojson file from the *insideairbnb.com* and use it to create a heatmap.

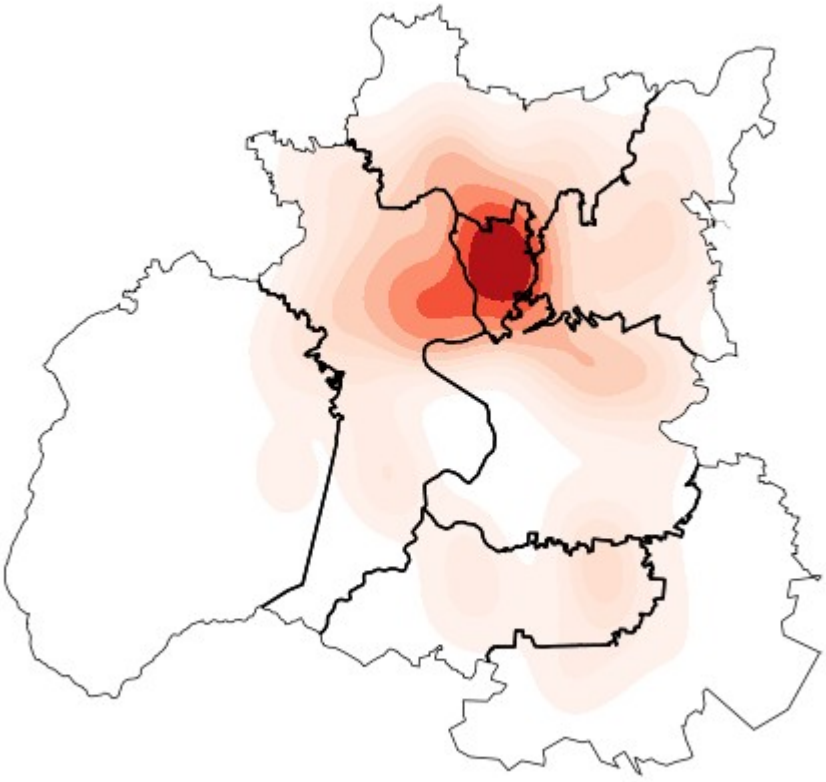
Here, we will read the file with geopandas and transform it into the coordinate system 4326, which is the standard for GPS.

```
# Reading the Asheville polygon shapefile (geojson)
geofile = '/content/neighbourhoods.geojson'
asheville = gpd.read_file(geofile)
asheville = asheville.to_crs(4326)
```

Then, creating a heatmap is easy with these few lines of code. First we create a density plot (kde) to be projected over the map and we use polyplot to display both layers.

```
# Heatmap
ax = gplt.kdeplot(listings_gpd,
          fill=True, cmap='Reds',
          clip=asheville.geometry,
          projection=gcrs.WebMercator())
# Plotting the heatmap on top of the geometry
gplt.polyplot(asheville, ax=ax, zorder=1);
```

Here is the beautiful result. The listings are very concentrated in the central zone of the city.



Moving on, we will now use `Folium` module to create a heatmap of the prices per region and a choropleth map.

First, the heatmap. Considering that we want to see the data points in different colors, separated by price range bins, here is a code to create those groups. First we use `pandas.cut` to create bins for each 100 dollars and then we use `map()` to transform the labels into colors. This will be used in our next step.

```
from numpy import Inf
# Create clip levels for prices
listings['price_bins']= pd.cut(listings.price,
       bins= [-Inf, 100, 200, 300, 400, 500, Inf],
       labels= ['0-100', '100-200', '200-300', '300-400', '400-500', '500+'])
# Create bin colors
listings['colors'] = listings.price_bins.map({'0-100': 'lightblue', '100-
200':'blue', '200-300':'gold', '300-400':'orange', '400-500':'red',
'500+':'black'})
```

Let's create a base map, with starting point at downtonw Asheville, NC.

```
# Creating a base map
m = folium.Map(location= [35.5951, -82.5515], zoom_start=10)
```

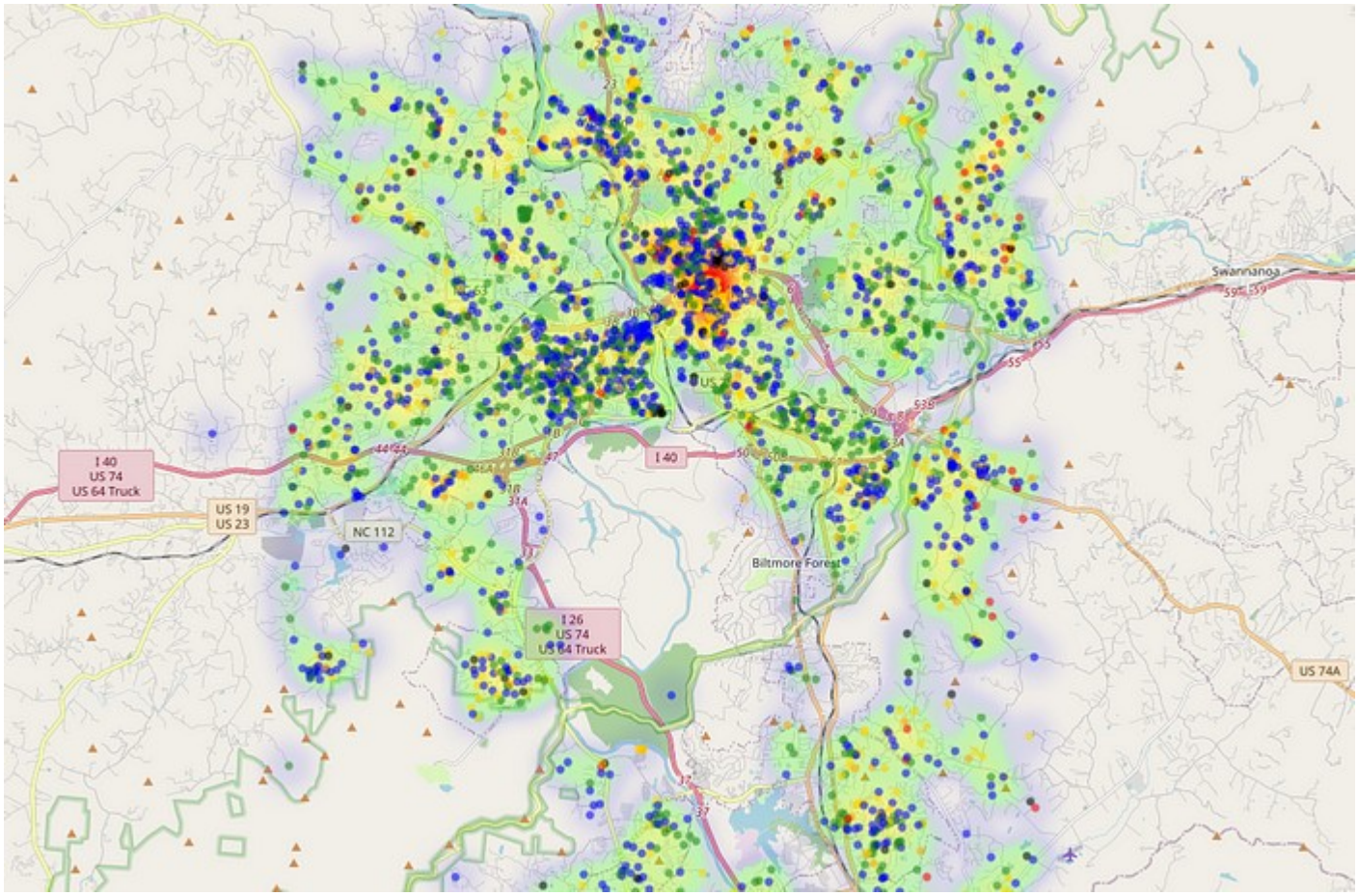
Then we can add the points.

```
# Adding the points
for lat, lon, ptcolor in zip(listings.latitude, listings.longitude,
listings.colors):
  folium.CircleMarker(
     location=[lat, lon],
     radius=2,
     opacity=0.5,
     color=ptcolor,
     fill=True,
     fill_color=ptcolor,
  ).add_to(m)
```

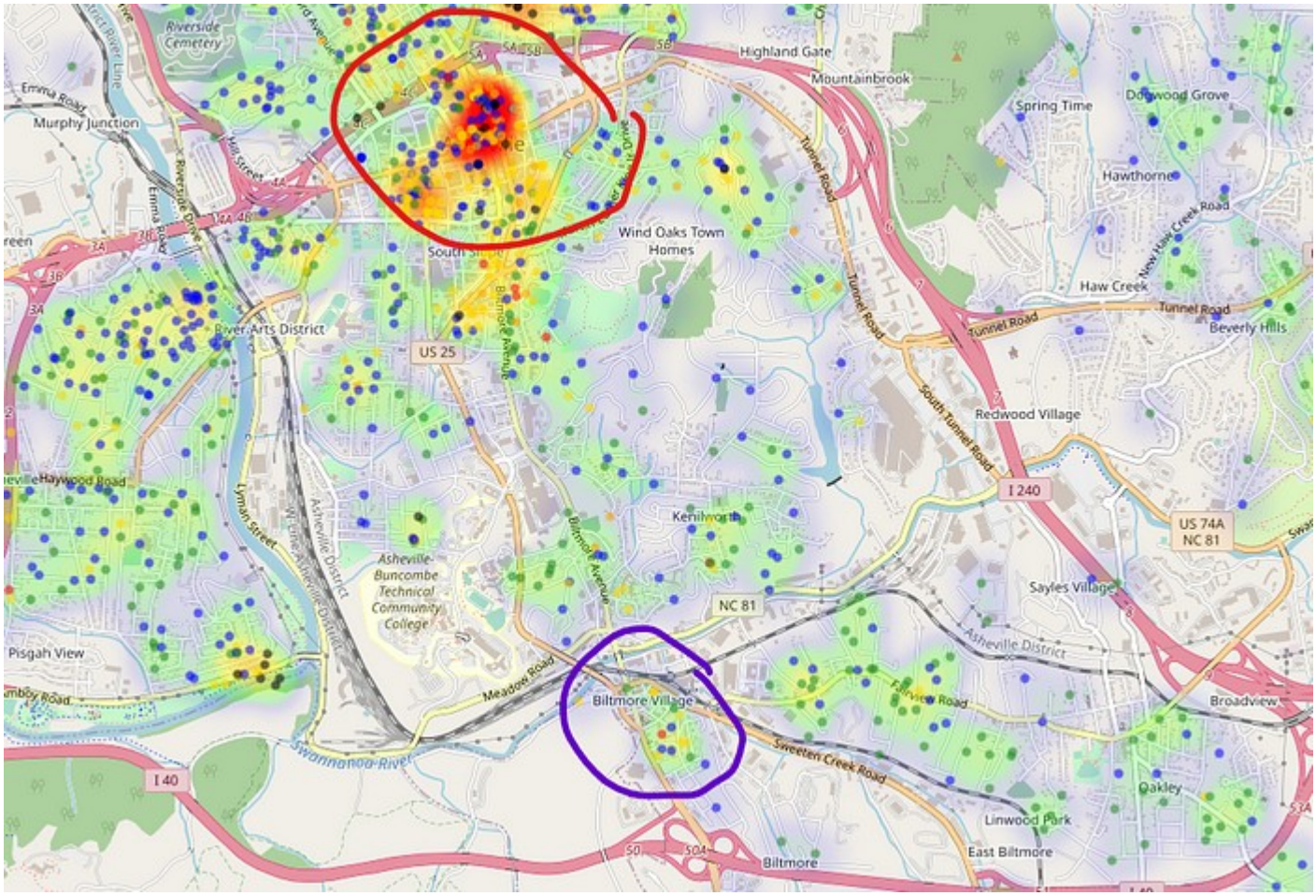
Now we will create the heatmap. We must transform the data to a list of values. Only lat, long and price. The heatmap will be by price. Then we import `HeatMap from folium.plugins` and add it to the base map.

```
from folium import plugins
# Preparing data for plot
data = listings[['latitude','longitude', 'price']].values
data =data.tolist()
# Create Heat Map with Folium
hm = plugins.HeatMap(data,gradient={0.1: 'blue', 0.2: 'lime', 0.4: 'yellow',
0.6: 'orange', 0.9: 'red'},
                min_opacity=0.1,
                max_opacity=0.9,
                radius=20,
                use_local_extrema=False)
# Add to base map
hm.add_to(m);
# Display
m
```

Here is the result. A beautiful heat map that shows good insights.



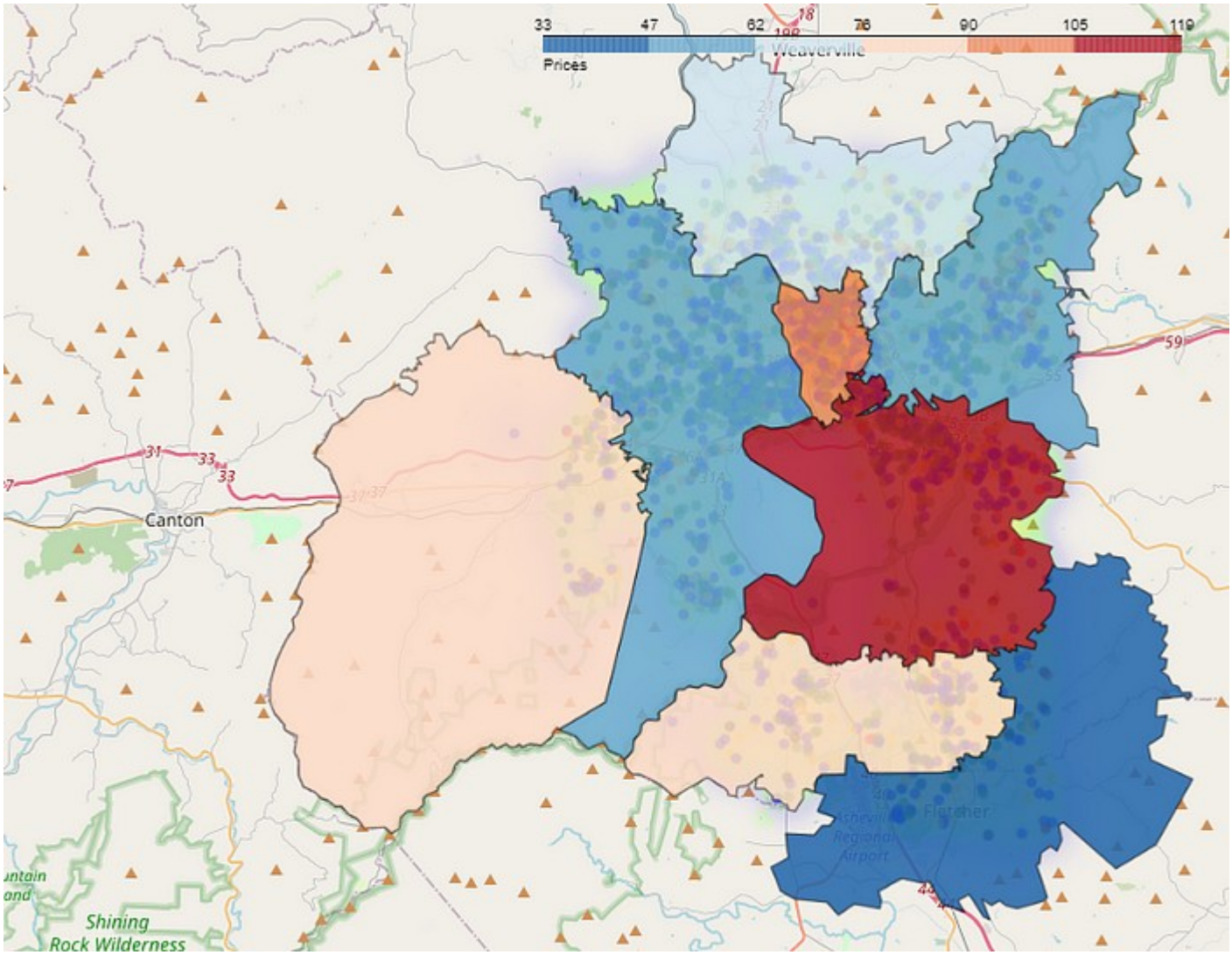
Look how the prices are higher next to the city downtown. And closer to the Biltmore attraction, it’s not that crowded with listings. There are a few of them, some with lower range prices, probably due to the distance to downtonwn.



Choropleth

Finally, if we want to create a quick Choropleth from this data, here's the code snippet. We can use the `asheville` file created previously as our geo data, the `listings` file is where the prices come from and the `neighbourhood` variable is the link between the geojson and the data frame.

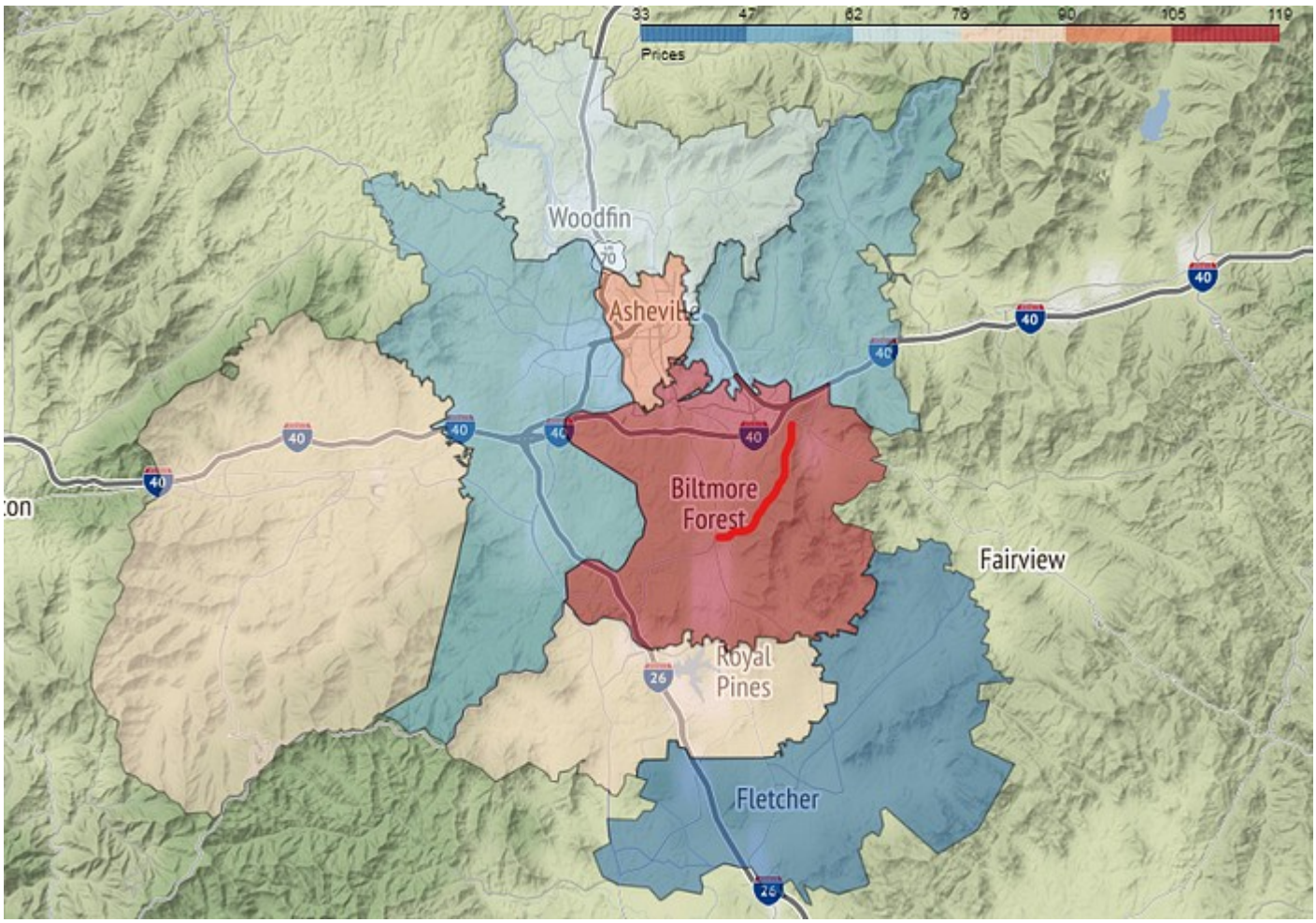
```
# Add a choropleth layer
folium.Choropleth(
    geo_data=asheville,
    name="choropleth",
    data=listings,
    columns=["neighbourhood", "price"],
    key_on="feature.properties.neighbourhood",
    fill_color="RdBu_r",
    fill_opacity=0.5,
    line_opacity=0.5,
    legend_name="Prices",
).add_to(m)
m
```



Looking at the map, we see that the highest prices are on the right side. If you don't know the area, Asheville is a city by the Blue Ridge Mountains, a famous and beautiful place in North Carolina, especially during the fall season. The *Blue Ridge Parkway* is one of the most famous driveways in the country, being constantly mentioned as one of the most beautiful drives in USA. So let's plot another choropleth, but now with the terrain mode on, then we can see where the mountains are.

```
# Base map with Terrain mode
m = folium.Map(location= [35.5951, -82.5515], zoom_start=10, tiles="Stamen
Terrain")
# Add a choropleth layer to a terrain map
folium.Choropleth(
    geo_data=asheville,
    name="choropleth",
    data=listings,
    columns=["neighbourhood", "price"],
    key_on="feature.properties.neighbourhood",
    fill_color="RdBu_r",
    fill_opacity=0.5,
    line_opacity=0.5,
    legend_name="Prices",
).add_to(m)
m
```

The red line is where the Blue Ridge Parkway is located.



This is one of the views of the Blue Ridge Parkway. I think you can have an idea why those rental properties are more expensive, right? So beautiful view!

Before You Go

I how you had fun with this little project. Geospatial data is super powerful and can bring a lot of insights. To work with it, packages like Geopandas, Geoplot and Folium are a must.

Here is the full code for this exercise:

Studying/Python/Geospatial at master · gurezende/Studying

This is a repository with my tests and studies of new packages - Studying/Python/Geospatial at master ·…

github.com

If you liked this content, follow my blog for more. Also, find me on LikedIn too.

Gustavo Santos - Medium

Read writing from Gustavo Santos on Medium. Data Scientist. I extract insights from data to help people and companies…

gustavorsantos.medium.com

Reference

JORDAN, David S. 2023.*Applied Geospatial Data Science with Python*. 1 ed. Pactk Publishing.