# Analyzing download speeds in Kentucky counties using Python

In this tutorial I will talk about how to:

```
* Download the Ookla open dataset
* Geocode the tiles to Kentucky counties
* Make a table of the top and bottom 20 counties by download speed
* Map the counties
```

There are two main ways to join these tiles to another geographic dataset: quadkeys and spatial joins. This tutorial will use the spatial join approach.

```
# %matplotlib inline

from datetime import datetime

import geopandas as gp
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

from shapely.geometry import Point
from adjustText import adjust_text
```

---

## Download data

First, download the data using the link below

```
import rasterio
import geopandas as gpd
from rasterio.mask import mask
from shapely.geometry import box
import numpy as np
import pandas as pd

# Filepaths
raster_file = r"C:\Users\shaho\Desktop\
VNL_v2_npp_2021_global_vcmslcfg_c202203152300.cf_cvg.tif"
boundary_file = r"C:\Users\shaho\Downloads\geoBoundaries-ARE-ADM1-all\
geoBoundaries-ARE-ADM1_simplified.geojson"
output_usa_csv = r"C:\Users\shaho\Documents\output.csv"
```

```
# Load united emerites's Boundary
united emerites = gpd.read_file(boundary_file)
united emerites = united emerites.to_crs(epsg=4326)

# Open the Raster and Check Overlap
with rasterio.open(raster_file) as src:
    raster_bounds = box(*src.bounds)
    print("Raster Bounds:", src.bounds)
    print("Indonesia Bounds:", united emerites.total_bounds)

    if not raster_bounds.intersects(united emerites.union_all()):
        raise ValueError("united emerites's boundary does not overlap
with the raster extent.")

    # Clip the raster
    Indonesia_geom_list = [feature["geometry"] for feature in
Indonesia.__geo_interface__["features"]]
    clipped_raster, clipped_transform = mask(src, Indonesia_geom_list,
crop=True)

# Extract Raster Values
light_intensity = clipped_raster[0]
rows, cols = np.where(~np.isnan(light_intensity))
values = light_intensity[rows, cols]
x_coords, y_coords = rasterio.transform.xy(clipped_transform, rows,
cols)

data = pd.DataFrame({
    'longitude': x_coords,
    'latitude': y_coords,
    'light_intensity': values
})
data.to_csv(output_csv, index=False, mode='w')

print(f"Extracted data saved to {output_csv}")
```

```
Raster Bounds: BoundingBox(left=-180.00208333335, bottom=-
65.00208445335001, right=180.00208621335, top=75.00208333335)
Indonesia Bounds: [-179.14735527  -14.55254202  179.77845474
71.3525607 ]
```

```
Raster Metadata: {'driver': 'GTiff', 'dtype': 'uint16', 'nodata':
None, 'width': 86401, 'height': 33601, 'count': 1, 'crs':
CRS.from_wkt('GEOGCS["WGS 84",DATUM["WGS_1984",SPHEROID["WGS
84",6378137,298.257223563,AUTHORITY["EPSG","7030"]],AUTHORITY["EPSG","
6326"]],PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],UNIT["degree",0
.0174532925199433,AUTHORITY["EPSG","9122"]],AXIS["Latitude",NORTH],AXI
S["Longitude",EAST],AUTHORITY["EPSG","4326"]]'), 'transform':
```

```
Affine(0.0041666667, 0.0, -180.00208333335,
       0.0, -0.0041666667, 75.00208333335)}
Raster Bounds: (-180.00208333335, -65.00208445335001, 180.00208621335,
75.00208333335)
CSV file saved: C:\Users\pepob\Desktop\Data Analysis\
USA_2021_detailed.csv
```

```
def quarter_start(year: int, q: int) -> datetime:
    if not 1 <= q <= 4:
        raise ValueError("Quarter must be within [1, 2, 3, 4]")

    month = [1, 4, 7, 10]
    return datetime(year, month[q - 1], 1)


def get_tile_url(service_type: str, year: int, q: int) -> str:
    dt = quarter_start(year, q)

    base_url =
"https://ookla-open-data.s3-us-west-2.amazonaws.com/shapefiles/perform
ance"
    url = f"{base_url}/type%3D{service_type}/year%3D{dt:%Y}/quarter
%3D{q}/{dt:%Y-%m-%d}_performance_{service_type}_tiles.zip"
    return url

tile_url = get_tile_url("fixed", 2021, 4)
# 4 --> mn 9:12
tile_url
```

```
'https://ookla-open-data.s3-us-west-2.amazonaws.com/shapefiles/
performance/type%3Dfixed/year%3D2021/quarter%3D4/2021-10-
01_performance_fixed_tiles.zip'
```

```
tiles = gp.read_file(tile_url)

tiles
```

```
                  quadkey  avg_d_kbps  avg_u_kbps  avg_lat_ms  tests
devices  \
0        0022133222330032       26210       32253          28      1
1
1        0022332203013331        8077        2766          27     13
3
2        0022332203013333      547932       60364          28      4
2
3        0022332203031111      236319       39674          30     13
5
4        0022332203031112      268726       47344          44     13
2
...                   ...         ...         ...         ...    ...
...
```

```
4354753  3131120230000011       21687       15309           20       4
1
4354754  3131120300000300       49173       23851           13       3
1
4354755  3211031203221110        1047        1136          609       2
1
4354756  3313010232110233         269         974         1206       1
1
4354757  3313010232110322        1512        1510         1201       2
1

                                                  geometry
0        POLYGON ((-160.03784 70.63631, -160.03235 70.6...
1        POLYGON ((-162.60315 66.89991, -162.59766 66.8...
2        POLYGON ((-162.60315 66.89775, -162.59766 66.8...
3        POLYGON ((-162.60315 66.8956, -162.59766 66.89...
4        POLYGON ((-162.60864 66.89344, -162.60315 66.8...
...                                                    ...
4354753  POLYGON ((169.4696 -46.55886, 169.4751 -46.558...
4354754  POLYGON ((170.17822 -46.08847, 170.18372 -46.0...
4354755  POLYGON ((76.36597 -69.38031, 76.37146 -69.380...
4354756  POLYGON ((164.10828 -74.6934, 164.11377 -74.69...
4354757  POLYGON ((164.11377 -74.6934, 164.11926 -74.69...

[4354758 rows x 7 columns]
```

```
df = pd.DataFrame(tiles)
df.to_csv('Ookla.csv', index=False)

print("CSV file created successfully!")
```

```
CSV file created successfully!
```

```
import geopandas as gpd
import pandas as pd
from shapely.geometry import Point
from shapely.wkt import loads

# File paths
polygons_csv_path = r"C:\Users\shaho\Ookla.csv"
points_csv_path = r"C:\Users\shaho\Documents\output.csv"
output_csv_path = "joined_data.csv"

# Step 1: Load Polygons CSV
print("Loading polygons...")
polygons = pd.read_csv(polygons_csv_path)
polygons['geometry'] = polygons['geometry'].apply(loads)
polygons_gdf = gpd.GeoDataFrame(polygons,
geometry='geometry').dropna(subset=['geometry'])

# Step 2: Load Points CSV
```

```
print("Loading points...")
points = pd.read_csv(points_csv_path)
geometry = [Point(xy) for xy in zip(points['longitude'],
points['latitude'])]
points_gdf = gpd.GeoDataFrame(points,
geometry=geometry).dropna(subset=['geometry'])

# Step 3: Ensure CRS is Consistent
target_crs = "EPSG:4326"
print("Ensuring CRS is consistent...")
if polygons_gdf.crs != target_crs:
    polygons_gdf = polygons_gdf.set_crs(target_crs,
allow_override=True)
if points_gdf.crs != target_crs:
    points_gdf = points_gdf.set_crs(target_crs, allow_override=True)

# Step 4: Perform Spatial Join
print("Performing spatial join in chunks...")
chunk_size = 100000
results = []

for i in range(0, len(points_gdf), chunk_size):
    chunk = points_gdf.iloc[i:i + chunk_size]
    try:
        result = gpd.sjoin(chunk, polygons_gdf, how="left",
predicate="within")
        results.append(result)
    except Exception as e:
        print(f"Error processing chunk {i}-{i+chunk_size}: {e}")

# Step 5: Save the Result
joined_gdf = pd.concat(results, ignore_index=True)
print(f"Saving results to '{output_csv_path}'...")
joined_gdf.to_csv(output_csv_path, index=False)
print(f"Spatial join completed successfully. Output file:
'{output_csv_path}'")
```

```
Loading polygons...
Loading points...
Ensuring CRS is consistent...
Performing spatial join in chunks...
Saving results to 'joined_data.csv'...
Spatial join completed successfully. Output file: 'joined_data.csv'
```

```
import pandas as pd

# File paths
input_file = r"C:\Users\shaho\joined_data.csv" # Replace with your
input file path
output_file = 'cleaned_data.csv'  # Output file for cleaned data
```

```python
# List of columns to check for null values
columns_to_check = ['index_right', 'quadkey', 'avg_d_kbps',
'avg_u_kbps', 'avg_lat_ms', 'tests', 'devices']

# Define chunk size for processing
chunk_size = 100000  # Adjust based on your system's memory

# Initialize an empty list to hold chunks of cleaned data
cleaned_chunks = []

# Read the file in chunks
print("Processing data in chunks...")
for chunk in pd.read_csv(input_file, chunksize=chunk_size):
    # Remove rows with null values in the specified columns
    cleaned_chunk = chunk.dropna(subset=columns_to_check)
    # Append the cleaned chunk to the list
    cleaned_chunks.append(cleaned_chunk)

# Concatenate all cleaned chunks into a single DataFrame
data_cleaned = pd.concat(cleaned_chunks, ignore_index=True)

# Save the cleaned DataFrame to a new file
data_cleaned.to_csv(output_file, index=False)

print("Rows with null values in the specified columns have been
removed.")
print(f"Cleaned data saved to '{output_file}'")
```

```
Processing data in chunks...
Rows with null values in the specified columns have been removed.
Cleaned data saved to 'cleaned_data.csv'
```