Data Analysis for Economics

Topic 2: Prices and Taxation IV
(Spatial Competition with Gas Stations)

Marco Nieddu

Fall 2025

Outline

- We want to study the (possible) heterogeneity in the pass-through, especially with respect to **local competition**.
- We thus need to:
  - Provide a visualization of the gas station geographical distribution
  - Compute the number of stations per municipality
  - Compute the number of competitors of each gas station (within a given radius)

1. Import and Clean Data

Goal: Load the `pooled_2019_2024.csv` (built in previous classes) and perform basic cleaning.

Steps:

- Inspect the data: if any column (e.g. `date`) is in the wrong format, convert to the proper dtype.
- The data is a panel at gas station $\times$ day. Since we focus on time-invariant features, keep a single day (e.g. `'2022-03-14'`).
- Keep only rows with valid (non-missing) latitude/longitude and within Sardinia's bounding box.

AI prompt: *"How do I remove rows with missing values on a certain variable in Python/pandas?"*

2. Mapping

Goal: Show the geographical distribution of gas stations in Sardinia.
Steps:

- Create a `GeoDataFrame` from the `DF` using latitude and longitude (**HOW?**)
- Plot the points to visualize spatial coverage across Sardinia.
- Load Italy's municipality shapefile (`Com01012022_g`) and keep only Sardinia (`COD_REG = 20`).
- Check and convert both layers to the same CRS (**HOW?**)
- Overlay gas stations on Sardinia's municipality map.

AI prompt: *"How do I define a GeoDataFrame from the latitude and longitude in a DataFrame?"*

2. Mapping: Help 1

Hints: The original DataFrame (`df2022`) includes latitude and longitude, but it is not a GeoDataFrame. Define it directly with CRS WGS84 (geographic coordinates):

```
gdf2022 = gpd.GeoDataFrame(
    df2022,
    geometry=gpd.points_from_xy(
        df2022['Longitudine'],
        df2022['Latitudine']
    ),
    crs="EPSG:4326"
)
```

This step both creates the geometry and assigns the correct CRS (latitude/longitude degrees).

2. Mapping: Help 2

Hints: Each GeoDataFrame must have a defined coordinate system (CRS) to align layers correctly.

- **Convert (reproject)** both layers to a common projected CRS (in meters, suitable for distance calculations and maps):

```
gdf_1 = gdf_1.to_crs(epsg=32632)
gdf_2 = gdf_2.to_crs(epsg=32632)
```

This transformation ensures both datasets use the same coordinate system: `EPSG:32632` (UTM Zone 32N).

3. Define the Gas Station Density

Goal: Compute the number of stations by municipality.

Steps:

- Define a new object containing the number of stations in each municipality - e.g. counting the number of rows (gas stations) per municipality.

```
nstations=groupby('Comune').size()
```

- Standardize the municipality names in both datasets by converting them to uppercase and stripping leading and trailing whitespace.
- Merge back to the Sardinia shapefile on municipality name.
- Replace missing values (municipalities with zero stations) with 0.

AI prompt: *"How do I make a string variable uppercase and remove blank spaces?"* / *"How do I replace missing (NaN) values with zeros in Pandas?"*

4. Build a Distance Matrix

Goal: Compute pairwise distances (km) between stations to define competition.
Steps:

- Use `geopy.distance.geodesic()` to compute distances (**HOW?**).
- Create an empty square matrix (`DataFrame`) indexed by station IDs.
- Loop through stations, compute distances, and fill the matrix (**HOW?**).
- (OPTIONAL) Optimize by computing only the upper triangle and mirroring results.

AI prompt: *"How can I compute distances between latitude/longitude points using geopy in Python?"*

4. Build a Distance Matrix — Help 1

Goal: Calculate the distance (in km) between two coordinates.
Example 1:

```
from geopy.distance import geodesic
coord1 = (40.8, 9.1)
coord2 = (39.9, 8.6)
dist = geodesic(coord1, coord2).kilometers
```

Example 2 (from the DataFrame):

```
coord1 = (df2022.loc[52136, 'Latitudine'], df2022.loc[52136, 'Longitudine']
coord2 = (df2022.loc[9565,  'Latitudine'], df2022.loc[9565,  'Longitudine']
dist = geodesic(coord1, coord2).kilometers
```

Hint: `geodesic()` returns great-circle (real-world) distances on Earth's surface.

4. Build a Distance Matrix — Help 2

Goal: Create a matrix that stores the distance (in km) between every pair of gas stations. Each row and column represent a specific station. **Step 1: Create an empty matrix where rows and columns are gas station IDs (we make IDs the indexes.**

```
df2022 = df2022.set_index('idImpianto')
df_dist2022 = pd.DataFrame(index=df2022.index, columns=df2022.index)
```

Step 2: Loop logic

- Outer loop: select station *i*.
- Inner loop: compute the distance between *i* and every other station *j*.
- Store the result in the matrix cell `[i, j]`.

Hint: The resulting matrix is symmetric, and diagonal values equal 0.

5. Competition Measures

Goal: Compute competition measures and assemble the final dataset for export.

Steps:

- For each radius $r = \{1, 5, 10, 20\}$, create a boolean matrix (True=closer than r; False=further than r).
- Sum each row to count competitors within r km, subtracting 1 (the station itself).
- Merge back station-level data with municipality identifiers and station counts.
- Combine all variables into a single dataset and export
- Which competition measure makes more sense? How much variability we have when we increase the radius?

AI prompt: *"How can I count how many points fall within a certain distance using a distance matrix and then merge the results into one CSV file?"*

Possible extensions

What you have is a dataset at the gas-station level that includes two measures of competition:

- The number of gas stations falling in the same municipality
- The number of gas stations falling in a radius r.

Possible extensions:

- Compare competition over time (2019–2024).
- Exclude same-brand stations to measure effective competition.
- Is there a more efficient way to compute the number of gas station per municipalities? Look at the spatial join method.