# Spatial Autocorrelation

Wei Kang
University of California Riverside
wei.kang@ucr.edu

**Abstract**

Spatial autocorrelation is a key concept in quantitative spatial analysis that measures the degree of similarity or dissimilarity between spatially distributed observations. This paper introduces the concept and measurement of global and local spatial autocorrelation. An open-source Python workflow is provided to demonstrate how to measure, visualize, and interpret global and local Moran's I statistics—the most widely used method for spatial autocorrelation—by analyzing the spatial patterns of neighborhood housing affordability in the City of Riverside, California. The workflow includes guidance and code to address variance instability in proportion variables and the multiple testing issue in local statistics. Additionally, extensions and innovations to univariate spatial autocorrelation, such as multivariate and uncertainty-aware frameworks, are discussed.

## Introduction

Spatial autocorrelation is a fundamental concept in quantitative spatial analysis, providing a framework to measure the degree of similarity or dissimilarity between spatially distributed observations. This concept is rooted in Tobler's First Law of Geography, which states: "Everything is related to everything else, but near things are more related than distant things" (Tobler, 1970). Understanding spatial autocorrelation is critical for examining spatial processes, detecting spatial patterns, and identifying clusters or outliers within geographic data (Anselin and Williams, 2016).

Spatial autocorrelation can arise for a variety of reasons, including underlying spatial processes such as diffusion, neighborhood effects, or spatial spillovers. For example, in housing studies, affordability patterns may cluster due to shared socioeconomic conditions, policy interventions, or spatially dependent housing markets. Additionally, spatial autocorrelation may be induced by data collection or processing. Specifically, when the unit of observation does not match the unit of analysis (i.e., mismatched boundaries or the more general more general modifiable areal unit problem (MANP) (Manley, 2021)), spatial dependencies can arise. This issue may stem from different definitions of spatial boundaries

(e.g., census geographies not aligning with functional regions) or from measurement errors (e.g., inaccuracies during map digitization in GIS). In such cases, the spatial structure of the data may unintentionally introduce spatial autocorrelation.

The presence of spatial autocorrelation has important implications, as it violates the assumption of independence commonly used in classical statistical models. Ignoring spatial dependence may lead to biased estimates, inefficient models, or incorrect statistical inferences. Recognizing and quantifying spatial autocorrelation, therefore, is essential for accurate analysis, policy design, and decision-making.

While the concept of spatial autocorrelation and its measurements has been extensively discussed in many textbooks and journal articles (O'Sullivan and Unwin, 2010; De Smith et al., 2007; Bivand et al., 2013; Rey et al., 2023), this paper adds to the field by providing a practical and interactive example of its implementation. Using open-source Python packages, this notebook explores the spatial distribution of neighborhood housing affordability in the City of Riverside of Inlanda Southern California. The analysis applies both global Moran's I to measure overall spatial autocorrelation and local Moran's I to identify spatial clusters and outliers, offering a reproducible workflow for spatial autocorrelation analysis.

This notebook is organized into several sections. It begins with an introduction to the computational environment and data, outlining the tools and datasets used for analysis. The data section details the sources and preprocessing steps necessary for the analysis. The next section introduces the concept of spatial autocorrelation and provides a basic conceptual understanding. This is followed by a demonstration of spatial autocorrelation analysis applied to the housing affordability dataset, featuring visualizations and statistical insights. The notebook concludes with a summary of recent developments in spatial autocorrelation analysis.

## Computational environment

A wide array of specialized sopen source oftware are available for carrying out spatial autocorrelation analysis, including `spdep` (R) (Pebesma and Bivand, 2023), `esda`/`pysal` (Python) (Rey and Anselin, 2007), and `rgeoda` (R) and `pygeoda` (python) which are based on `libgeoda` (C++), the core of the widely-used Graphic User Interface (GUI)-based software `GeoDa` (Anselin et al., 2022). The choice would be much of a personal preference as they are all open source, easy to be integrated into a computational workflow, and producing similar output with only minor differences (Bivand, 2022).

In this computational notebook, I will work in a Python 3 environment and rely on Python Spatial Analysis Library (PySAL), a mainstay open-source Python ecosystem for spatial data science (Kang, 2020), to demonstrate spatial auto-

correlation analysis. Within the ecosystem, `pysal` is a meta package pulling together about 20 subpackages in the ecosystem capable of conducting a comprehensive variety of spatial analyses, including exploratory spatial data analysis (ESDA), spatial regressions, and spatial optimization (Rey et al., 2022). The subpackages are more lightweighted and can be installed and used independently to accomplish specific spatial analysis tasks. In this notebook, I will mainly work with `esda` and `libpysal`, two of the subpackages in the PySAL ecosystem, which are sufficient for the demonstration needs: `esda` includes a variety of classic and state-of-the-art global and local spatial autocorrelation statistics; `libpysal` is the foundation of all the other subpackages in the PySAL ecosystem, which inclueds foundational algorithms and data structures needed for spatial analysis, such as the construction of various spatial weights. I will also draw on functionalities of `splot`, a light-weight visualization package in the PySAL ecosystem for generating statistical plots assisting the undertanding of the results of the spatial autocorrelation analysis (Lumnitz et al., 2020). For the complete workflow, I will use `geopandas` for spatial data reading and wrangling (den Bossche et al., 2024), `libpysal` for constructing spatial weights, `esda` for the estimation and inference of global and local spatial autocorrelation, `splot`, `matplotlib`, and `contextily` for statistical and geospatial visualization, and the built-in python module `random` to ensure replication and reproducibility of the inference results.

In the code cell below, all the required python packages, classes, and functions are imported:

```
# loading packages for spatial data manipulation and analysis
import geopandas as gpd
import libpysal
import esda

# loading visualization packages
from splot.esda import lisa_cluster
import matplotlib.pyplot as plt
import contextily

# python built-in module for generating pseudo-random numbers
# of permutation based inference
import random

# python package for scientific computing
import numpy as np

# printing the name/version of all imported modules
#%load_ext watermark
#%watermark --iversions
```

# Data

Both the geographies and attributes used in this analysis are sourced from the U.S. Census Bureau for the investigation of the spatial patterns of neighborhood housing affordability in the City of Riverside, CA. onsistent with the literature, neighborhoods are represented by census tracts, and tract polygons are sourced from the 2020 Census Bureau's TIGER/Line Shapefiles[1]. A total of 93 tract polygons are included as they either fall within or intersect the city boundary.

The primary attribute of interest is the proportion of housholds that are burdened by housing costs. Specifically, households spending 30% or more of their gross incomes on housing costs, which could include rent or mortgage payments, utilities, and related expenses, are considered housing cost-burdened. This definition of housing affordability focuses on the ability of households to afford housing without experiencing financial strain and has been widely adopted by government agencies (e.g., the U.S. Department of Housing and Urban Development (HUD)) and affordable housing programs (e.g., HUD's Housing Choice Voucher Program).

The dataset includes three continuous attributes related to housing cost burden, derived from the 2018–2022 5-year American Community Survey (ACS) estimates: (1) `All30C`: the number of housing cost-burdened households. (2) `AllC`: the total number of households in the tract. (3) `All30P`: the proportion of housing cost-burdened households, calculated as All30C divided by AllC.

These variables reflect all households, including renters and homeowners, with and without a mortgage.

A Jupyter Notebook (Python) named "0_DataCollection" is included in the repository. It demonstrates how to use the Census API to query the source data (i.e., ACS and TIGER/Line Shapefiles) and perform the necessary transformations, calculations, and merges.

The analysis begins by reading the spatial data file and projecting the geographic coordinate system to a planar coordinate system that uses linear measurements for the coordinates.

```
# read the spatial data set (shapefile) as a GeoDataframe
gdf = gpd.read_file("data/CostBurden_Riverside.shp")

# set the projected coordinate system
gdf = gdf.to_crs("epsg:2770")

gdf.info()
```

```
<class 'geopandas.geodataframe.GeoDataFrame'>
RangeIndex: 93 entries, 0 to 92
```

[1] Census 2020 TIGER/Line Shapefiles: https://www.census.gov/geographies/mapping-files/time-series/geo/tiger-line-file.2020.html

```
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   GEOID     93 non-null     object
 1   AllC      93 non-null     int64
 2   All30C    93 non-null     int64
 3   All30P    92 non-null     float64
 4   geometry  93 non-null     geometry
dtypes: float64(1), geometry(1), int64(2), object(1)
memory usage: 3.8+ KB
```

## Basic conceptual intuition

When our research question involves spatial entities, one of the first steps is often to visualize the attribute of interest on a map. From the map, we could visually identify where the high and low values occur. Most often than not, these patterns are not spatially random. In other words, areas with low values are more likely to be surrounded by other low values, and similarly for high values. This type of spatial association is referred to as spatial dependence, which invalidates the assumption of independence commonly used in classical statistical models. Consequently, it is important to determine whether the observed spatial pattern is significant—that is, whether it deviates from what would be expected under spatial randomness.

Spatial autocorrelation statistics were developed for this purpose. These statistics measure the correlation of an attribute across space by simultaneously evaluating locational and attribute similarity. Observations that are similar in attribute values and geographically close contribute positively to spatial autocorrelation statistics. Conversely, observations that are dissimilar in attribute values but geographically close contribute negatively.

Global spatial autocorrelation (GSA) statistics are summary measures produced for the entire map, indicating the extent to which higher (or lower) values are overall geographically clustered. They work by assessing locational and attribute similarities across all pairs of observations. A generic form of a GSA statistic is given by Equation (1):

$$\Gamma = \sum_i \sum_j w_{ij} y_{ij} \tag{1}$$

where $\Gamma$ is a GSA statistic, $Y$ is a matrix of attribute similarity, and $W$ is a matrix representing locational similarity. Different functions have been proposed to measure attribute similarity for continuous values, leading to various GSA statistics, such as Moran's I and Geary's C. The matrix $W$, known as the spatial weights matrix, encodes our perception about spatial relationships between observations. The weights in $W$ can be based on contiguity, distance,

or accessibility.
Only the elements of $W$ that correspond to pairs of neighboring observations take non-zero values, making it a sparse matrix even for moderately large datasets. For instance, with a queen-based contiguity weight for polygon geometries, two observations ($i \neq j$) are considered neighbors if they share a vertex or an edge and the $(i, j)$ entry in $W$ is nonzero.

In addition to summarizing spatial autocorrelation across the entire map using a single statistic, it is often important to identify spatial clusters of high or low values, commonly referred to as spatial hotpots and coldspots. Additionally, spatial outliers, defined defined as low (high) values surrounded by high (low) values, may also be of interest. Local spatial autocorrelation (LSA) statistics, which are local decompositions of GSA statistics, are designed for such purposes. LSA statistics provide a measure of spatial autocorrelation for each observation by using information from neighboring observations This approach reveals the spatial nonstationarity of spatial autocorrelation, meaning that even in the absence of significant global spatial autocorrelation, local pockets of strong spatial autocorrelation may still exist. A generic form of an LSA statistic is given by Equation (2):

$$\Gamma_i = \sum_j w_{ij} y_{ij} \tag{2}$$

where $\Gamma_i$ is the LSA statistic for observation $i$. Similar to the GSA statistics, different LSA statistics exist based on various definitions of attribute similarity. For instance, local Moran's I and local Geary's C are local decompositions of Moran's I and Geary's C, respectively(Anselin, 1995).

# Application

In this section, I will demonstrate how to analyze the spatial distribution of neighborhood housing affordability in the City of Riveride using gloabl and local Moran's Is, the most widely used methods for assessing global and local spatial autocorrelation. These statistics measures spatial autocorrelation by utilizing deviations from the mean to define attribute similarity. Local Moran's I evaluates the coexistence of attribute and locational similarity by multipling the focal unit's deviation and by spatial lag, which is the average of the deviations of its neighbors as defined in the spatial weights matrix (Anselin, 1995). Global Moran's I is proportional to the summation of Local Moran's Is. A positive estimate for global (local) Moran's I suggest positive gloabl (local) spatial autocorrelation, meaning that similar values (high or low) are geographically clustered. The larger the estimate, the stronger the spatial autocorrelation. Conversely, a large negative estimate indicates negative spatial autocorrelation, where dissimilar values (e.g., high values surrounded by low values) are more likely to be near each other.

Since we are interested in the spatial patterns of a proportion variable, proper adjustment are necessary to address its intrinsic variance instability. Specifically, census tracts with a smaller number of households tend to have lower precision in their rate estimates, which could lead to the incorrect identification of spurious outliers.

## Choropleth mapping of the spatial pattern

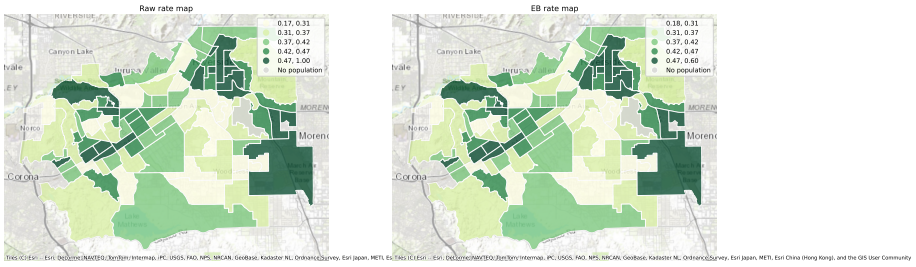
The first step of the ESDA is often to visually inspect the spatial distribution. As mentioned earlier, the raw rate, calculated by dividing the number of cost-burdened households by the total number of households, suffer from variance instability. One way to address this issue is by applying Empirical Bayes (EB) smoothing, where the EB rate is the weighted average between the raw rate and the citywide average, with larger weights given to the city average for tracts with a smaller number of households. The function `Empirical_Bayes` from `esda`'s `smoothing` can be used to easily estimate the EB rates. The estimated EB rate will be added to the dataset as a new column named `All30EBP`.

```
gdf_temp = gdf[gdf.AllC>0]
gdf_temp= gdf_temp.assign(
  All30EBP=esda.smoothing.Empirical_Bayes(
    gdf_temp.All30C.values,
    gdf_temp.AllC.values).r)

gdf = gdf.set_index("GEOID").merge(
  gdf_temp[["All30EBP","GEOID"]].set_index("GEOID"),
  how="outer",
  right_index=True,
  left_index=True)
```

Next, we visualize the raw rate map and the EB rate map side-by-side for comparison. We use call the `plot()` method from `GeoDataFrame` to produce a choropleth map with quintile classification, providing a quick preview of the spatial distribution of the housing cost-burden rate across neighborhoods in the City of Riverside. Additionally, a basemap could be overlaid on the choropleth map to give geographic context to the study area using the `contextily` package.

```
f, axes = plt.subplots(1,2, figsize=(20, 10))
gdf.plot(
  "All30P",
  cmap="YlGn",
  k=5,
  scheme='quantiles',
  edgecolor="white",
  linewidth=1,
  alpha=0.75,
  legend=True,
```

```
  ax=axes[0],
  missing_kwds={
    "color": "lightgrey",
    "label": "No population"
    }
  )
contextily.add_basemap(
  axes[0],
  crs=gdf.crs,
  source=contextily.providers.Esri.WorldTopoMap
  )
axes[0].set_title("Raw rate map")
axes[0].set_axis_off();

gdf.plot(
  "All30EBP",
  cmap="YlGn",
  k=5,
  scheme='quantiles',
  edgecolor="white",
  linewidth=1,
  alpha=0.75,
  legend=True,
  ax=axes[1],
  missing_kwds={
    "color": "lightgrey",
    "label": "No population"
    }
  )
contextily.add_basemap(
  axes[1],
  crs=gdf.crs,
  source=contextily.providers.Esri.WorldTopoMap
  )
axes[1].set_title("EB rate map")
axes[1].set_axis_off();
```



The general spatial patterns are similar across the raw and EB rate maps despite small differences in the range of rates. Visually, neighborhoods with similar values seem to be geographically close. For instance, neighborhoods that were more prevalent with housing cost-burdened households are concentrated in the northeastern and western parts of the city, as well as the southeastern neighborhoods that overlap significantly with the City of Moreno Valley. In contrast, lower rates concentrated in southern Riverside, which are known for newer developments and suburban-style housing. However, our visual preception could be deceiving, which is particularly relevant as the polygons vary in size. We therefore adopt spatial autocorrelation methods to formally test aganist spatial randomness and identify local hot/cold spots and outliers.

## Spatial autocorrelation analysis

In `esda`, the estimation and inference of the spatial autocorrelation statistics, global and local Moran's Is, are implemented as the respective Python `classes`: `Moran` and `Moran_Local`. For each, locational similarity (i.e., spatial weights) must be defined and calculated prior to instantiation of these classes whereas attribute similarity is assessed internally, making attribute data a required parameter. As we are analyzing proportion variable which could suffer from variance instability, the Empirical Bayes (EB) standardization has been proposed to correct Moran's I statistics as a solution (Assunção and Reis, 1999). The `esda` classes `Moran_Rate` and `Moran_Rate_Local` implemented this adjustment. In contrast to `Moran` and `Moran_Local` where the attribute of interest should be provided, these two classes require values for the number of the events (i.e., housing cost-burdened households) and the population at risk (i.e., total households).

### Locational similarity and spatial weights

The first step of spatial autocorrelation analysis is to construct spatial weights for representing locational similarity. In this application, a queen contiguity spatial weight matrix will be constructed using `esda`'s newest module, `graph`, a modern implementation of spatial weights (Rey, 2024).

```
#tracts without any population are removed before the analysis
gdf_r = gdf[gdf.AllC>0]

# construct queen-based spatial weights
g_queen = libpysal.graph.Graph.build_contiguity(gdf_r, rook=False)
```

We can call its adjacency attribute to inspect the neighbor information for each tract:

```
g_queen.adjacency
```

```
focal        neighbor
06065030101  06065030103    1
```

```
             06065030104    1
             06065030502    1
             06065042209    1
             06065042300    1
                           ..
06065050901  06065046700    1
             06065050902    1
06065050902  06065042206    1
             06065042212    1
             06065050901    1
Name: weight, Length: 528, dtype: int64
```

The returned value is a `pandas` multi-index Series. For each focal tract, the second index lists tracts that share a vertex or edge, which are its queen-contiguity neighbors, and the last column (value of the Series) is the corresponding weight for each neighbor. In the queen-contiguity spatial weight, every weight has the value of 1.

Futher, we need to row-standardize the spatial weight matrix for the estimation of the global and local Moran's I statistics. This can be accomplished by calling the `transform` method and passing a `string` value of `"r"` on the `graph` object. After the transformation, for each focal tract, the weights sum up to 1.

```
gr = g_queen.transform("r")
gr.adjacency
```

```
focal        neighbor
06065030101  06065030103    0.200000
             06065030104    0.200000
             06065030502    0.200000
             06065042209    0.200000
             06065042300    0.200000
                               ...
06065050901  06065046700    0.200000
             06065050902    0.200000
06065050902  06065042206    0.333333
             06065042212    0.333333
             06065050901    0.333333
Name: weight, Length: 528, dtype: float64
```

### Estimation and inference

The discussion so far has been focused on the estimation of spatial autocorrelation statistics. While a large positive Moran's I statistic suggests a tendency for high (low) values to cluster near other high (low) values, formal inference is required to reject spatial randomness—the null hypothesis of GSAs.

Both analytical and permutation-based approaches have been proposed for in-

ference regarding Moran's I. he analytical approach, which assumes independent and normally distributed random variates under the null hypothesis, is straightforward to implement. However, it relies on assumptions of normality, spatial stationarity, and a sufficiently large sample size, all of which are easily violated in practice.

In contrast, the permutation-based approach is more robust, as it does not require these assumptions. Instead, it constructs a reference distribution for Moran's I by repeatedly permuting the observed values across locations. From this reference distribution, a pseudo p-value is calculated to assess the significance of the observed Moran's I.

For local Moran's I, conditional random permutation is performed, where the value of the focal tract remains fixed while the values of other observations are spatially permuted. This process generates a unique reference distribution for each tract, which is then used to calculate its corresponding pseudo p-value. The inference of local Moran's Is would potentially suffer from multiple testing issues as we are conducting a lot of tests (i.e., number of observations) simultaneously. By random chance, 5% of tests would be rejected at the 5% significance level. While several methods for addressing multiple testing have been proposed, the False Discovery Rate (FDR) method is employed in this demonstration (Benjamini and Yekutieli, 2001; de Castro and Singer, 2006).

### Global Spatial autocorrelation

The `esda` classes are designed in such manner that estimation and inference are simutaneously conducted. For the permutation-based inference, we need to set the desired number of (conditional) random spatial permutations. The number of permutations to be 99999 to be consistent across global and local inference as the local inference requires the number of permutations to be sufficiently large to yield a pseudo p-value that is small enough once the multiple testing is adjusted for with the FDR method.

```
# initialize the random number generator with a given seed to make sure
# the permutation based inference can be replicated on different computers
# and at different times
random.seed(5)

# create an instance of the Moran class from esda
moran = esda.Moran_Rate(
  gdf_r.All30C.values,
  gdf_r.AllC.values,
  gr,
  permutations=99999)

print("Moran's I estimate:", moran.I)
print("Peudo p-value:", moran.p_sim)
```

```
Moran's I estimate: 0.34004457985355707
Peudo p-value: 1e-05
```

The estimate of Moran's I statitic is a positive 0.34. The peudo p-value under 99,999 permutations is 1e-05, meaning that none of the permuted spatial pattern yielded a statistic larger than the actual observed data. These results indicate a strong deviation from spatial randomness in neighborhood housing affordability. In other words, neighborhoods with higher (or lower) concentrations of cost-burdened households were more likely to be adjacent to similar neighborhoods.

### Local spatial autocorrelation

Next, we use local Moran's I to decompose the global statistic and to detect potential hot/cold spots or spatial outliers. The instantiation of the local Moran class is similar to that for the global class:

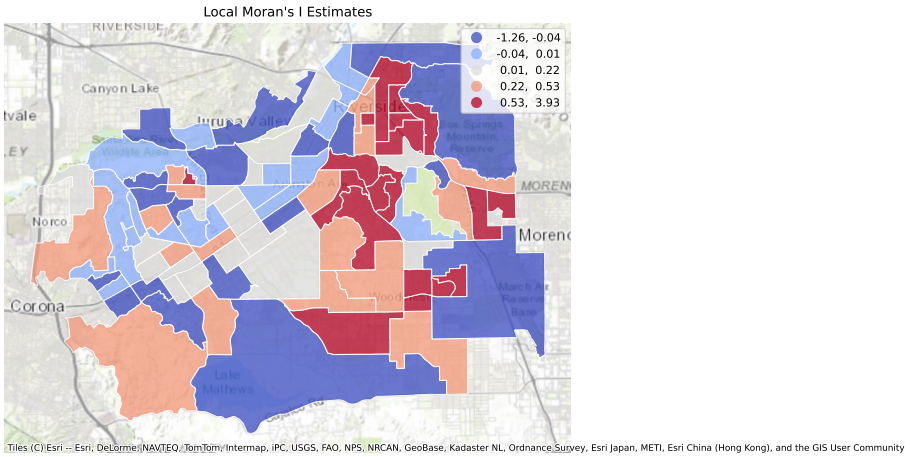
```
random.seed(5)

# create an instance of the Moran_Local class from esda
moran_loc = esda.moran.Moran_Local_Rate(
  gdf_r.All30C.values,
  gdf_r.AllC.values,
  g_queen,
  transformation = "r",
  permutations=99999
  )
moran_loc.Is.round(2)
```

```
array([ 0.31, -0.02, -0.03,  0.14, -0.02, -0.26,  0.26,  0.52,  1.36,
        0.67,  0.53,  0.27,  0.88, -0.05,  0.15, -0.26, -0.02,  0.02,
        0.01,  0.4 , -0.05, -0.  , -0.  ,  0.07,  0.1 ,  0.22,  0.27,
       -0.01,  0.05, -0.14,  0.12,  0.01, -0.05, -0.14, -0.03,  0.02,
        0.39, -0.1 , -0.02, -0.  ,  0.11, -0.03,  0.  ,  0.  ,  1.45,
        0.25, -0.43,  0.01,  0.31, -0.1 , -0.04, -0.26,  0.07,  0.26,
       -0.03,  0.02,  0.09, -0.16,  0.15,  0.49,  0.16, -0.13,  0.24,
       -0.26,  0.26,  0.83,  0.45,  0.5 , -0.08,  0.26,  0.02,  1.28,
        1.97,  1.38,  1.22,  0.08,  1.14,  2.02,  1.69,  1.8 ,  0.25,
        0.85, -0.76, -0.04, -0.06,  0.98,  0.74,  3.69,  3.93, -1.26,
        0.22, -0.13])
```

As shown above, for each neighborhood, a local Moran's I statistic was estimated and these values vary - some are larger than the global Moran's I estimate of 0.34 and others are smaller. A choropleth map of the local statistics would be useful for understanding which neighborhoods have larger and smaller local spatial autocorrelation:

```
# add local Moran's I estimates as a new column
# named "lisa" to the GeoDataFrame
```

```
gdf_r = gdf_r.assign(lisa=moran_loc.Is)

f, ax = plt.subplots(1,1, figsize=(10, 7))
gdf_r.plot(
  "lisa",
  cmap="coolwarm",
  k=5,
  scheme='quantiles',
  edgecolor="white",
  linewidth=1,
  alpha=0.75,
  legend=True,
  ax=ax
  )

contextily.add_basemap(
  ax,
  crs=gdf_r.crs,
  source=contextily.providers.Esri.WorldTopoMap
  );
ax.set_title("Local Moran's I Estimates");
ax.set_axis_off();
```




As shown in the map, large positive values clustered in the northeastern and central city - an indication of potential hot or cold spots depending on the housing affordability levels of the focal tract. Additionally, despite the positive estimate obtained for the global measure, some neighborhoods exhibited negative local spatial autocorrelation, an indication of low (high) levels of housing affordability being surrounded by high (low) levels of housing affordability, or spatial outliers.

In order to determine whether our visual perception is statistically sound, we resort to the peudo p-values obtained for each tract with conditional spatial permutation. As described before, we adjust for multiple local testing with FDR. The adjusted threshold for the overall 5% significance level could be calculated using `esda`'s function `fdr`:

```
p_adj = esda.fdr(moran_loc.p_sim, 0.05)
print(p_adj)
```

```
0.004891304347826087
```

The adjusted threshold of 0.005 is much smaller than 0.05. Next, we used this threshold to conduct inference about the local Moran's I statistics, which could be accomplished by calling the method `get_cluster_labels()` from the `Moran_Local_Rate` instance and passing the adjusted threshold. This method could generate one of five unique values for each neighborhoods: Insignificant, High-High, Low-Low, Low-High, and High-Low. Apart from "Insignificant", the last four values indicate statistically significant local statistics with the family significance level of 5%. Additionally, for these four values, the value on the left of - refers to the attribute level of the focal unit, whether it is larger (i.e., High) or smaller (i.e., Low) than the global average; similarly, the value on the right of - refers to the attribute level of the spatial lag of the focal unit - in other words, whether the average of the neighboring units is larger (i.e., High) or smaller (i.e., Low) than the global average of all spatial lags. Following these definitions, High-High and Low-Low are indicators of hot and cold spots where focal units are very similar to their neighbors. and Low-High, and High-Low are indicators of spatial outliers where focal units are very different from their neighbors.

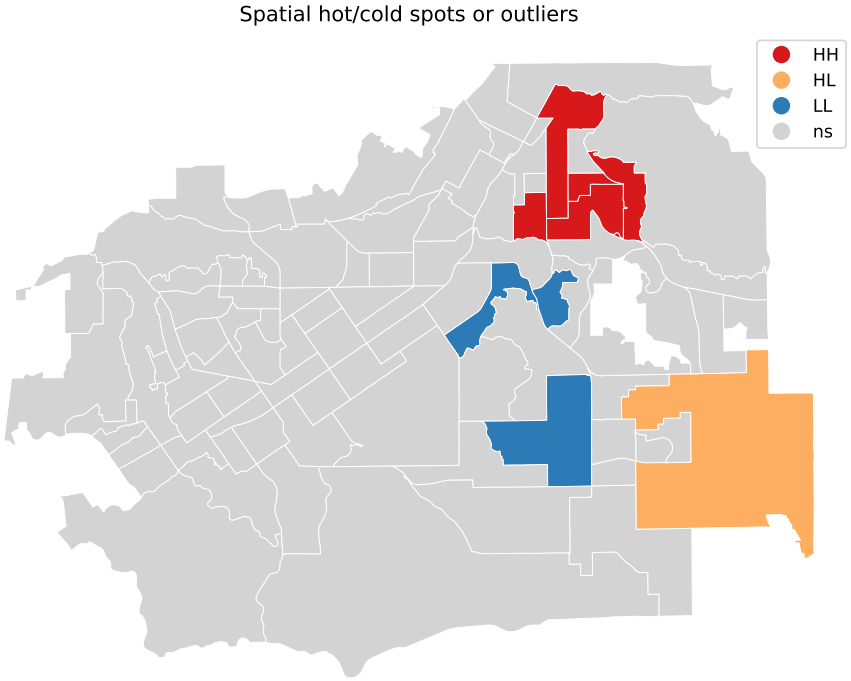
```
gdf_r = gdf_r.assign(
  p_sim=moran_loc.p_sim,
  cluster=moran_loc.get_cluster_labels(crit_value=p_adj))

# query statistically significant local statistics
gdf_r[gdf_r.cluster!="Insignificant"].drop(columns=["geometry"])
```

| GEOID | AllC | All30C | All30P | All30EBP | lisa | p_sim | cluster |
|---|---|---|---|---|---|---|---|
| 06065030501 | 1207 | 506 | 0.419 | 0.418234 | 0.255135 | 0.00304 | High-High |
| 06065030602 | 1145 | 370 | 0.323 | 0.325583 | 0.667500 | 0.00327 | Low-Low |
| 06065042005 | 1386 | 471 | 0.340 | 0.341352 | 0.496329 | 0.00118 | Low-Low |
| 06065042208 | 885 | 197 | 0.223 | 0.230379 | 2.022947 | 0.00271 | Low-Low |
| 06065042209 | 1219 | 711 | 0.583 | 0.576698 | 1.685610 | 0.00066 | High-High |
| 06065042213 | 1173 | 541 | 0.461 | 0.458711 | 0.847304 | 0.00058 | High-High |
| 06065046501 | 21 | 21 | 1.000 | 0.590517 | 3.686008 | 0.00205 | High-High |
| 06065046502 | 940 | 576 | 0.613 | 0.603045 | 3.934338 | 0.00005 | High-High |
| 06065046700 | 1262 | 704 | 0.558 | 0.552332 | -1.255149 | 0.00431 | High-Low |

| GEOID | AllC | All30C | All30P | All30EBP | lisa | p_sim | cluster |
|---|---|---|---|---|---|---|---|

We obtained 9 tracts that are identified to be either spatial hot/cold spots or outliers. Only 1 out of these 9 tracts is identified to be the center of a "High-Low" cluster with a negative local statistic. 5 tracts are centers of "High-High" clusters and 3 are centers of "Low-Low" clusters. We can further visualize them on a map:

```
f, ax = plt.subplots(1,1, figsize=(10, 7))
lisa_cluster(moran_loc, gdf_r, p=p_adj, ax=ax)
ax.set_title("Spatial hot/cold spots or outliers");
```




On the map, tracts that are identified to be "High-High" (HH) are highlighted in red and they form one cluster in the northestern part of the city. On the other hand, two "Low-Low" (LL) clusters are formed as highlighted in blue and they are located pretty closely. The sole tract that is the center of a "High-Low" (HL) cluster is located on the intersection of two cities, Riverside and Moreno Valley. On the other hand, none of the spatial "anomalies" are located in the west of the city which was considered to host neighborhoods of high levels of housing-cost burdend households upon visual inspection of the housing affordability map earlier.

## Conclusion

Spatial autocorrelation is a fundamental concept in quantitative human geography. This notebook provides a comprehensive, open-source Python workflow using the `esda` package to measure, infer, and visualize global and local Moran's I statistics for a proportion variable. These tools are essential for understanding spatial patterns and relationships in geographic data.

While Moran's I is one of the most widely used statistics for spatial autocorrelation, alternative measures like Geary's C and Getis-Ord G are also commonly applied for continuous variables, each with a unique definition of attribute similarity. For categorical data, join count statistics can be used. Readers interested in exploring these measures further are encouraged to refer to foundational texts such as (O'Sullivan and Unwin, 2010) and recent developments by (Rey et al., 2023). These additional methods are also available in the `esda` package (PySAL-team, 2025).

Substantial advancements in spatial autocorrelation analysis have been made in recent decades. For example, the concept has been extended to multivariate contexts, allowing the analysis of spatial relationships across two or more variables. (Anselin and Li, 2020) introduced multivariate Geary's C statistics, which assess the extent to which neighbors in a multi-attribute space are also neighbors in geographic space. Additionally, (Wolf, 2024) proposed conditional local Moran statistics, which account for confounding factors to measure the local spatial structure of an outcome variable. Beyond single-variable and multivariate analyses, spatial autocorrelation has been extended to address new spatial entities, such as human mobility vectors (Liu et al., 2015). Researchers have also begun incorporating attribute and spatial data uncertainties into spatial autocorrelation analysis (Jung et al., 2019; Arbia et al., 2016), an area that remains ripe for further exploration.(Jung et al., 2019; Arbia et al., 2016), an area that remains ripe for further exploration.

This notebook serves as a starting point for applying spatial autocorrelation techniques, with potential applications in fields such as urban planning, public health, and environmental studies. Future developments in the field are likely to further refine these methods, enhance computational efficiency, and broaden their applicability to complex spatial phenomena.

## References

Anselin, Luc (1995), "Local indicators of spatial association-LISA." *Geographical Analysis*, 27, 93–115.

Anselin, Luc and Xun Li (2020), "Tobler's law in a multivariate world." *Geographical Analysis*, 52, 494–510, URL https://onlinelibrary.wiley.com/doi/abs/10.1111/gean.12237.

Anselin, Luc, Xun Li, and Julia Koschinsky (2022), "Geoda, from the desktop to

an ecosystem for exploring spatial data." *Geographical Analysis*, 54, 439–466, URL https://onlinelibrary.wiley.com/doi/abs/10.1111/gean.12311.

Anselin, Luc and Sarah Williams (2016), "Digital neighborhoods." *Journal of Urbanism: International Research on Placemaking and Urban Sustainability*, 9, 305–328, URL https://doi.org/10.1080/17549175.2015.1080752.

Arbia, Giuseppe, Giuseppe Espa, and Diego Giuliani (2016), "Dirty spatial econometrics." *The Annals of Regional Science*, 56, 177–189, URL http://dx.doi.org/10.1007/s00168-015-0726-5.

Assunção, Renato M. and Edna A. Reis (1999), "A new proposal to adjust moran's i for population density." *Statistics in Medicine*, 18, 2147–2162, URL http://dx.doi.org/10.1002/(SICI)1097-0258(19990830)18:16<2147::AID-SIM179>3.0.CO;2-I.

Benjamini, Yoav and Daniel Yekutieli (2001), "The control of the false discovery rate in multiple testing under dependency." *The Annals of Statistics*, 29, 1165–1188, URL http://www.jstor.org/stable/2674075.

Bivand, Roger (2022), "R packages for analyzing spatial data: A comparative case study with areal data." *Geographical Analysis*, 54, 488–518, URL https://onlinelibrary.wiley.com/doi/abs/10.1111/gean.12319.

Bivand, Roger S, Edzer J Pebesma, Virgilio Gómez-Rubio, and Edzer Jan Pebesma (2013), *Applied spatial data analysis with R*. Springer.

de Castro, Marcia Caldas and Burton H. Singer (2006), "Controlling the false discovery rate: A new application to account for multiple and dependent tests in local statistics of spatial association." *Geographical Analysis*, 38, 180–208, URL http://dx.doi.org/10.1111/j.0016-7363.2006.00682.x.

De Smith, Michael John, Michael F Goodchild, and Paul Longley (2007), *Geospatial analysis: a comprehensive guide to principles, techniques and software tools*. Troubador publishing ltd.

den Bossche, Joris Van, Kelsey Jordahl, Martin Fleischmann, Matt Richards, James McBride, Jacob Wasserman, Adrian Garcia Badaracco, Alan D. Snow, Brendan Ward, Jeff Tratner, Jeffrey Gerard, Matthew Perry, cjqf, Geir Arne Hjelle, Mike Taves, Ewout ter Hoeven, Micah Cochran, Ray Bell, rraymondgh, Matt Bartos, Pieter Roggemans, Lucas Culbertson, Giacomo Caria, Nicholas YS Tan, Nick Eubank, sangarshanan, John Flavin, Sergio Rey, and James Gardiner (2024), "geopandas/geopandas: v1.0.1." URL https://doi.org/10.5281/zenodo.12625316.

Jung, Paul H, Jean-Claude Thill, and Michele Issel (2019), "Spatial autocorrelation and data uncertainty in the american community survey: a critique." *International Journal of Geographical Information Science*, 33, 1155–1175.

Kang, Wei (2020), "Pysal and spatial statistics libraries." In *The Geographic Information Science & Technology Body of Knowledge* (John P. Wilson, ed.),

3rd quarter 2020 edition, UCGIS, URL https://doi.org/10.22224/gistbok/2020.3.1.

Liu, Yu, Daoqin Tong, and Xi Liu (2015), "Measuring spatial autocorrelation of vectors." *Geographical Analysis*, 47, 300–319.

Lumnitz, Stefanie, Dani Arribas-Bell, Renan Cortes, James Gaboardi, Verena Griess, Wei Kang, Taylor Oshan, Levi Wolf, and Sergio Rey (2020), "'splot' - visual analytics for spatial statistics." *Journal of Open Source Software*, 5, 1882, URL https://doi.org/10.21105/joss.01882.

Manley, David (2021), "Scale, aggregation, and the modifiable areal unit problem." In *Handbook of regional science*, 1711–1725, Springer.

O'Sullivan, D. and D.J. Unwin (2010), *Geographic information analysis*. John Wiley & Sons Inc.

Pebesma, Edzer and Roger S. Bivand (2023), *Spatial Data Science With Applications in R*. Chapman & Hall, URL https://r-spatial.org/book/.

PySAL-team (2025), "GitHub - pysal/esda: statistics and classes for exploratory spatial data analysis — github.com." https://github.com/pysal/esda. [Accessed 04-01-2025].

Rey, Serge (2024), "W to Graph Migration Guide; libpysal v4.12.1 Manual — pysal.org." https://pysal.org/libpysal/user-guide/graph/w_g_migration.html. [Accessed 16-12-2024].

Rey, Sergio, Dani Arribas-Bel, and Levi John Wolf (2023), *Geographic data science with python*. Chapman and Hall/CRC.

Rey, Sergio J. and Luc Anselin (2007), "PySAL: A Python Library of Spatial Analytical Methods." *The Review of Regional Studies*, 37, 5–27.

Rey, Sergio J., Luc Anselin, Pedro Amaral, Dani Arribas-Bel, Renan Xavier Cortes, James David Gaboardi, Wei Kang, Elijah Knaap, Ziqi Li, Stefanie Lumnitz, Taylor M. Oshan, Hu Shao, and Levi John Wolf (2022), "The pysal ecosystem: Philosophy and implementation." *Geographical Analysis*, 54, 467–487, URL https://onlinelibrary.wiley.com/doi/abs/10.1111/gean.12276.

Tobler, W. R. (1970), "A computer movie simulating urban growth in the detroit region." *Economic Geography*, 46, 234–240.

Wolf, Levi John (2024), "Confounded local inference: Extending local moran statistics to handle confounding." *Annals of the American Association of Geographers*, 0, 1–16, URL https://doi.org/10.1080/24694452.2024.2326541.