Moran_point_pattern

December 16, 2024

[1]:
```
#| echo: false
#| output: false
import warnings
warnings.filterwarnings('ignore')

import pandas as pd
import numpy as np
from scipy.stats import pearsonr, pointbiserialr, chi2_contingency
import matplotlib.pyplot as plt
from matplotlib.font_manager import FontProperties

# Read the data
file_path = 'data/listings.csv'     #which is the path in this repository
airbnb_data = pd.read_csv(file_path)
```

[2]:
```
#| echo: false
#| output: false
# Calculate the estimation of nights booked for each listing
airbnb_data = airbnb_data[airbnb_data['availability_365'] > 0]
airbnb_data['estimated_nights_booked'] = airbnb_data['reviews_per_month'] * 12␣
↪* airbnb_data['minimum_nights'] * 2
```

[3]:
```
#| echo: false
#| output: false
#Data cleaning: assign the estimated nights booked to each borough
# Replace NaN with 0
airbnb_data['estimated_nights_booked'] = airbnb_data['estimated_nights_booked'].
↪fillna(0)

# Convert the column to integers
airbnb_data['estimated_nights_booked'] = airbnb_data['estimated_nights_booked'].
↪astype(int)

#Count the number of listings in each borough using 'neighbourhood' column
borough_counts = airbnb_data['neighbourhood'].value_counts()
```

```
# Filter the DataFrame to include only rows where estimated_nights_booked is␣
↪greater than 90
filtered_data = airbnb_data[airbnb_data['estimated_nights_booked'] > 90]

#Count the number of listings with estimation of nights booked larger than 90␣
↪days in each borough
borough_counts_90 = filtered_data['neighbourhood'].value_counts()
```

[4]:
```
#| echo: false
#| output: false
# Merge the two series into a DataFrame
combined_data = pd.concat([borough_counts, borough_counts_90], axis=1,␣
↪keys=['Total_listings', 'More_than_90'])

# Calculate the ratio of listings with more than 90 booked nights per total␣
↪listings
combined_data['Ratio_of_more_than_90'] = combined_data['More_than_90'] /␣
↪combined_data['Total_listings']

# Fill any NaN values that might occur if there are boroughs with no listings >␣
↪90 nights
combined_data['Ratio_of_more_than_90'] = combined_data['Ratio_of_more_than_90'].
↪fillna(0)

# Data formatting and round to four decimal places
combined_data['Ratio_of_more_than_90'] = combined_data['Ratio_of_more_than_90'].
↪apply(lambda x: round(x, 4))

# Rename the index label to 'Borough_name'
combined_data.index.rename('Borough_name', inplace=True)
```

[5]:
```
#| echo: false
#| output: false
# Load the borough codes
borough_code_file_path = 'data/borough_name_code.csv'
borough_codes = pd.read_csv(borough_code_file_path)

# Reset index in combined_data to turn the index into a regular column
combined_data.reset_index(inplace=True)
borough_codes.reset_index(inplace=True)

#Combine the ratio data and borough name with borough code by borough name
combined_data = pd.merge(combined_data, borough_codes[['Borough_name',␣
↪'Borough_code']], on='Borough_name', how='left')

# Set 'Borough_name' back as the index
```

```python
combined_data.set_index('Borough_name', inplace=True)

# Save the updated DataFrame
combined_data.to_csv('data/borough_listings_ratio.csv', index=True)
```

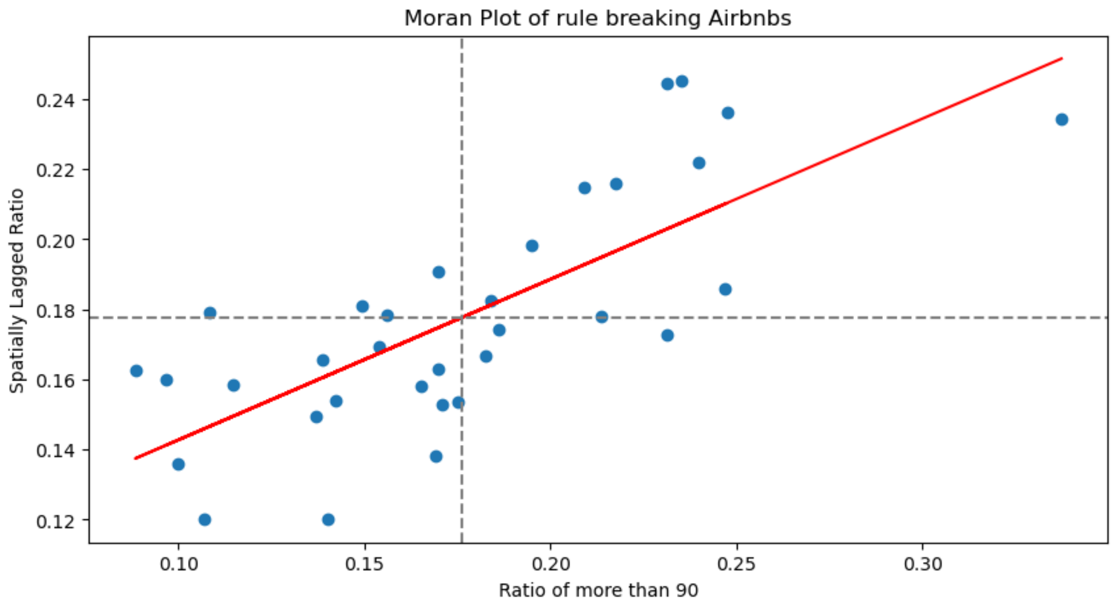
[6]:
```python
#| echo: false
#| output: false
import geopandas as gpd
import libpysal
from esda.moran import Moran, Moran_Local
import matplotlib.pyplot as plt
from libpysal.weights import Queen, KNN
import seaborn as sns
import os

# Load data
ratio = pd.read_csv("data/borough_listings_ratio.csv")
borough = gpd.read_file("data/statistical-gis-boundaries-london/ESRI/
 ↪London_Borough_Excluding_MHW.shp")

# merge
borough_ratio = borough.merge(ratio, left_on="GSS_CODE",␣
 ↪right_on="Borough_code")
```

[7]:
```python
#| echo: false
#| output: false
# Calculate neighbors using Queen contiguity
weights = Queen.from_dataframe(borough_ratio)
weights.transform = 'r'  # Row-standardize the weights
```

[8]:
```python
#| echo: false
#| output: false
os.makedirs('plots/raw', exist_ok=True)
```

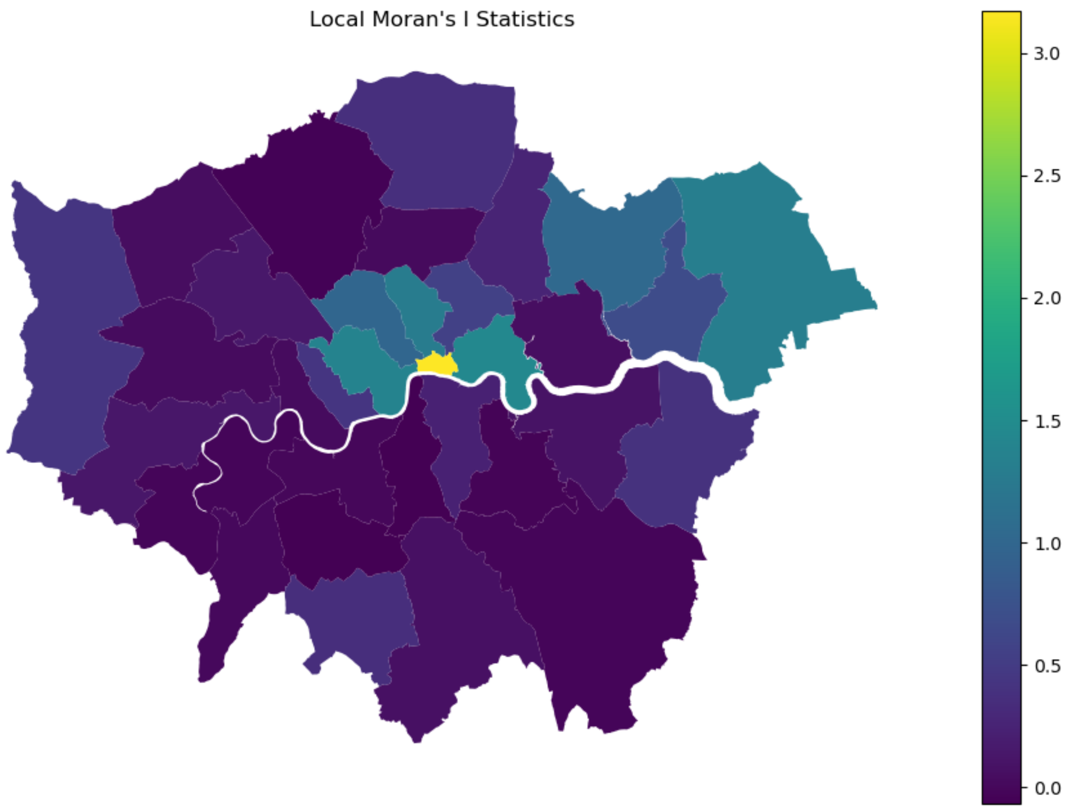
[9]:
```python
#| echo: false
#| output: false
# Global Moran's I
y = borough_ratio['Ratio_of_more_than_90']
moran = Moran(y, weights)
print(f"Global Moran's I: {moran.I:.3f}")
print(f"P-value: {moran.p_sim:.3f}")

# Moran Plot
def moran_plot(y, weights):
    lag = weights.sparse.dot(y)
    slope, intercept = np.polyfit(y, lag, 1)
    plt.figure(figsize=(10, 5))
    plt.scatter(y, lag)
    plt.plot(y, slope * y + intercept, 'r')
    plt.xlabel('Ratio of more than 90')
    plt.ylabel('Spatially Lagged Ratio')
    plt.title("Moran Plot of rule breaking Airbnbs")
    plt.axvline(y.mean(), color='gray', linestyle='--')
    plt.axhline(lag.mean(), color='gray', linestyle='--')
    plt.savefig('plots/raw/Moran_rule_breaking.png')
    plt.show()

moran_plot(y, weights)
```

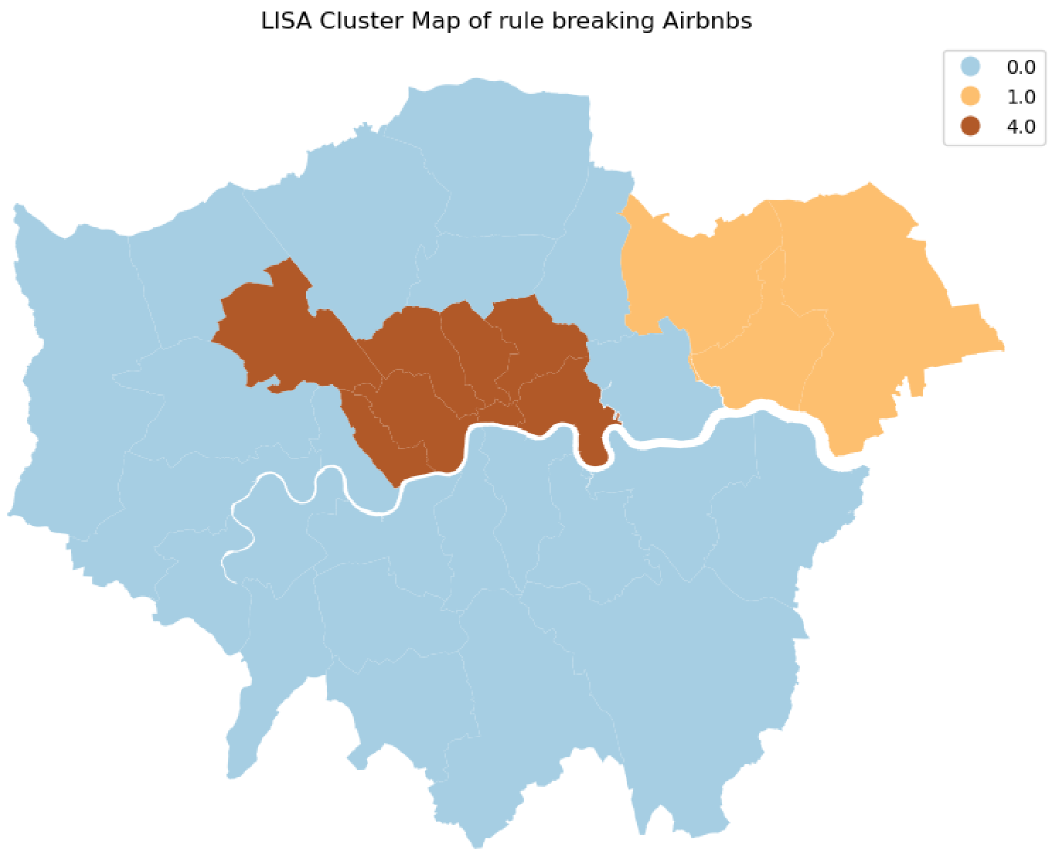
```
Global Moran's I: 0.458
P-value: 0.001
```

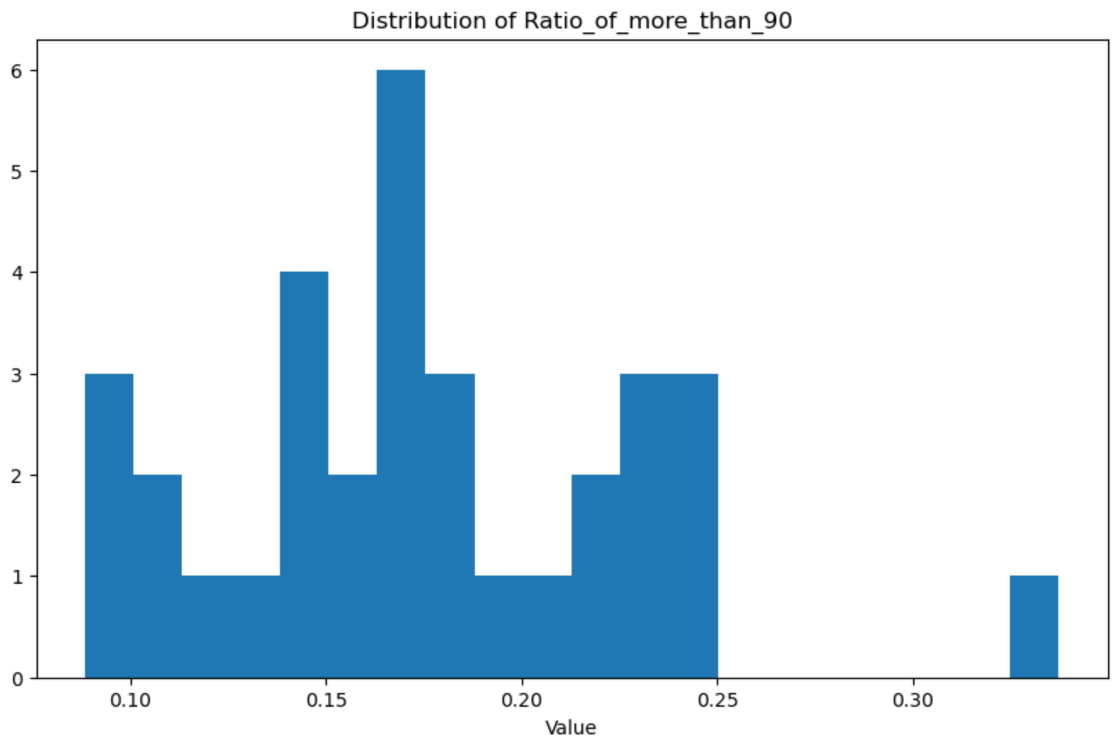


[10]:
```
#| echo: false
#| output: false
# Local Moran's I
local_moran = Moran_Local(y, weights)
borough_ratio['Ii'] = local_moran.Is
borough_ratio['p_value'] = local_moran.p_sim

# Plot Local Moran's I
fig, ax = plt.subplots(figsize=(12, 8))
borough_ratio.plot(column='Ii', legend=True, ax=ax)
plt.title("Local Moran's I Statistics")
```

```python
plt.axis('off')
plt.show()

# LISA Cluster Map
sig = 0.1
labels = ['Not Significant', 'Low-Low', 'Low-High', 'High-Low', 'High-High']
colors = ['white', 'blue', 'lightblue', 'pink', 'red']

# Standardize the variable of interest
y_std = (y - y.mean()) / y.std()
lag_std = weights.sparse.dot(y_std)

# Create significance masks
sig_mask = local_moran.p_sim < sig

# Create cluster categories
borough_ratio['quadrant'] = np.zeros(len(y))
borough_ratio.loc[sig_mask, 'quadrant'] = np.where(y_std < 0,
    np.where(lag_std < 0, 1, 2),
    np.where(lag_std < 0, 3, 4))[sig_mask]

# Plot LISA clusters
fig, ax = plt.subplots(figsize=(10, 10))
borough_ratio.plot(column='quadrant', categorical=True, k=5, cmap='Paired',
                  legend=True, ax=ax)
plt.title('LISA Cluster Map of rule breaking Airbnbs')
plt.axis('off')
plt.savefig('plots/raw/LISA_rule_breaking.png')
plt.show()

# Additional analysis plots
plt.figure(figsize=(10, 6))
plt.hist(y, bins=20)
plt.title('Distribution of Ratio_of_more_than_90')
plt.xlabel('Value')
plt.show()

print(y.describe())
# print(local_moran.Is.describe())
print(pd.Series(local_moran.Is).describe())

print(f"Number of significant clusters: {(local_moran.p_sim < 0.1).sum()}")
```


Local Moran's I Statistics
3.0
2.5
2.0
1.5
1.0
0.5
0.0

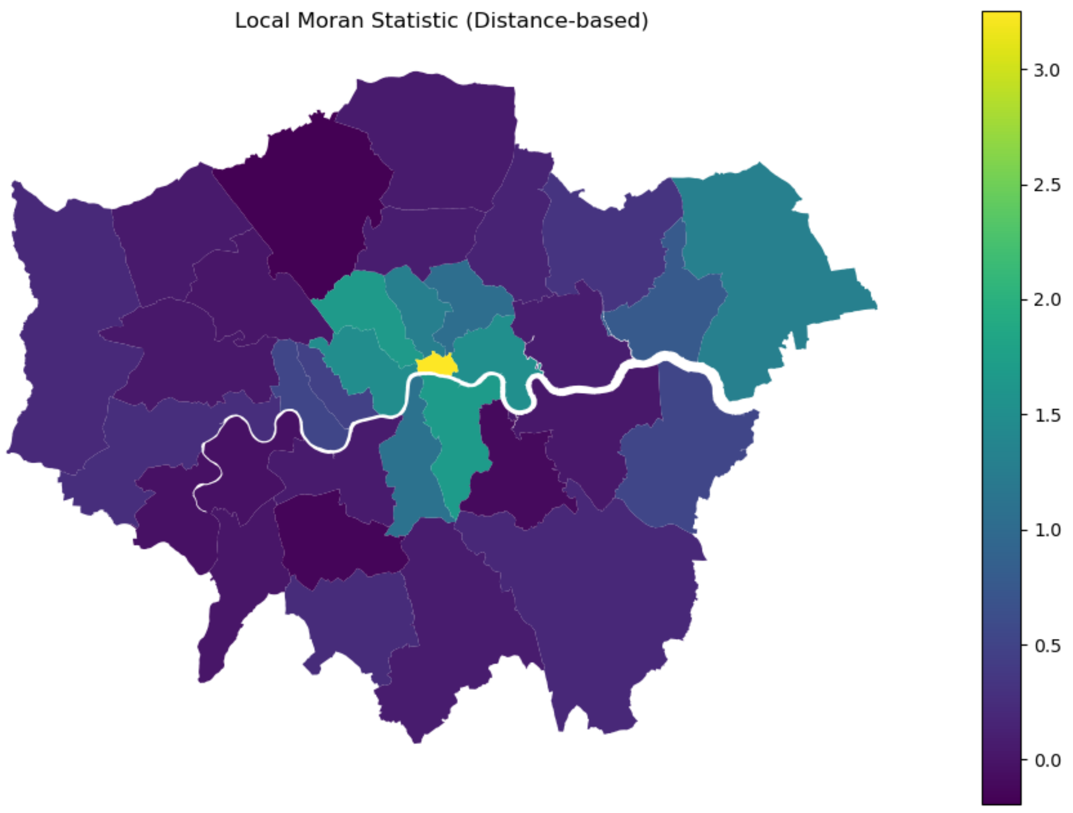

LISA Cluster Map of rule breaking Airbnbs
0.0
1.0
4.0



```
count    33.000000
mean      0.176067
std       0.054405
min       0.088400
25%       0.140200
50%       0.169800
75%       0.213800
max       0.337300
Name: Ratio_of_more_than_90, dtype: float64
count    33.000000
mean      0.444497
std       0.677907
min      -0.066242
25%       0.017070
50%       0.142216
75%       0.559219
max       3.172065
dtype: float64
Number of significant clusters: 11
```

[11]:
```
#| echo: false
#| output: false
# Distance-based weights (20km)
```

```
centroids = borough_ratio.geometry.centroid
coords = np.column_stack((centroids.x, centroids.y))
knn = KNN.from_dataframe(borough_ratio, k=4)  # Approximate 20km neighbors
knn.transform = 'r'

# Calculate Local Moran's I with distance weights
local_moran_dist = Moran_Local(y, knn)

# Add results to GeoDataFrame
borough_ratio['Ii_dist'] = local_moran_dist.Is

# Plot results with distance-based weights
fig, ax = plt.subplots(figsize=(12, 8))
borough_ratio.plot(column='Ii_dist', legend=True, ax=ax)
plt.title("Local Moran Statistic (Distance-based)")
plt.axis('off')
plt.show()
```



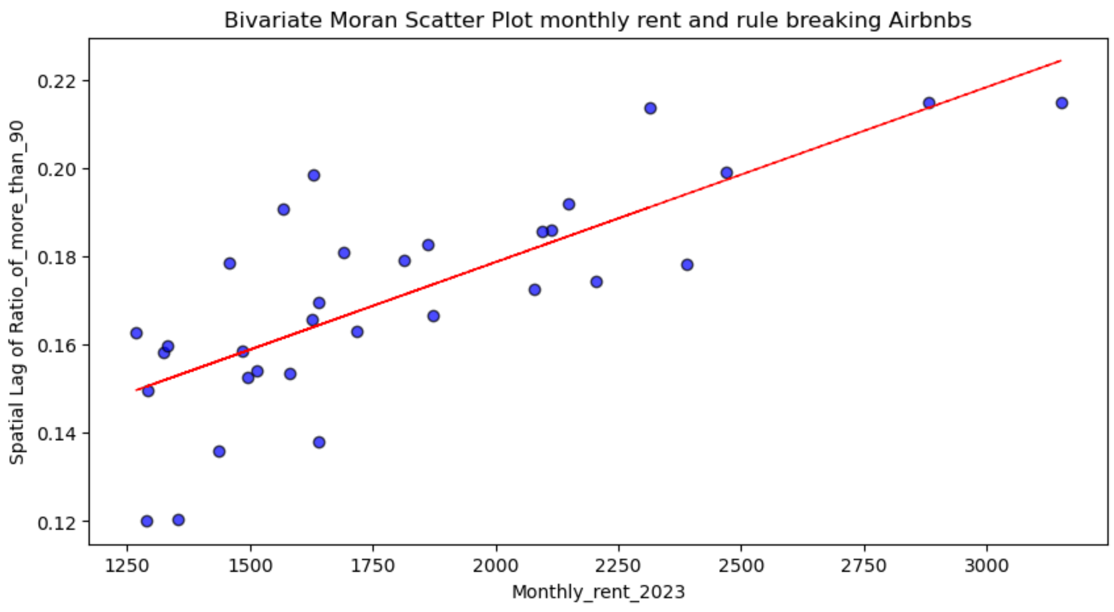
[12]:
```
#| echo: false
#| output: false
from libpysal.weights import Queen, lag_spatial
from esda.moran import Moran_BV, Moran_Local_BV

# load data
connect = pd.read_csv("data/connect.csv")
borough = gpd.read_file("data/statistical-gis-boundaries-london/ESRI/
↪London_Borough_Excluding_MHW.shp")

# merge the data
borough_connect = borough.merge(connect, left_on="GSS_CODE",␣
↪right_on="Borough_code")
```

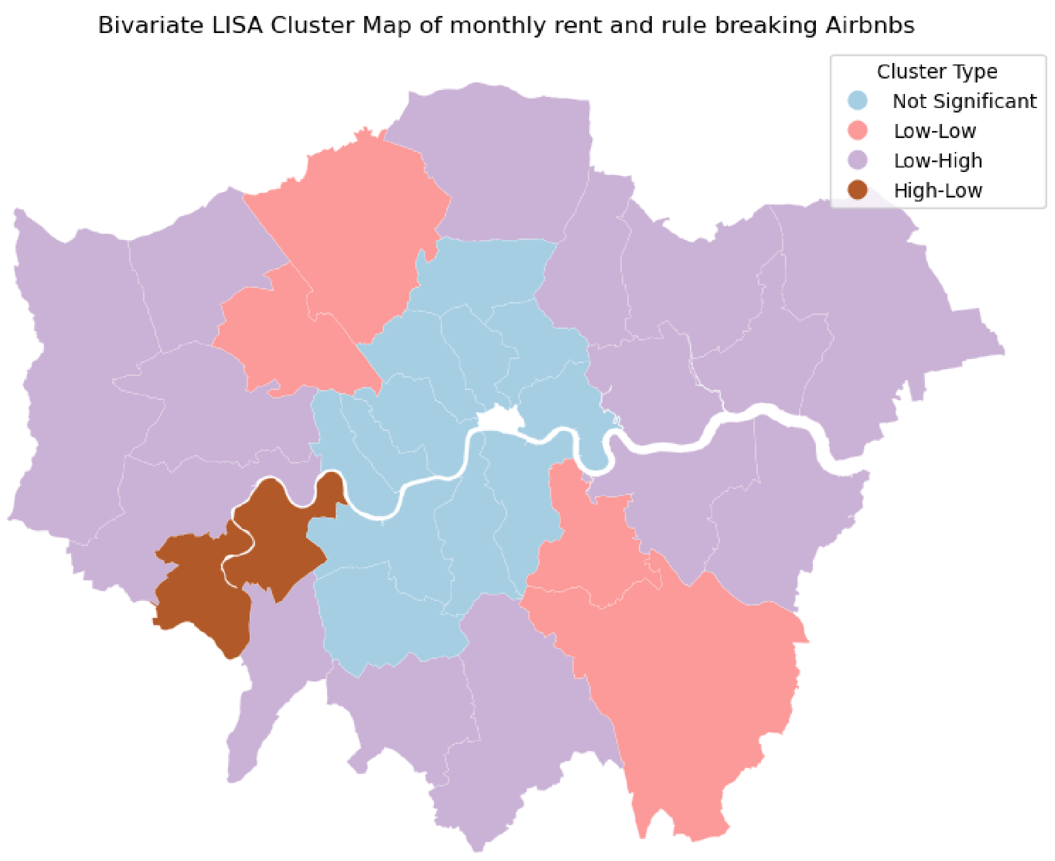
[13]:

```
#| echo: false
#| output: false
# analyse the spatial autocorrelation of monthly rent and airbnbs breaking the␣
↪rule
# Variables
var1 = 'Monthly_rent_2023'
var2 = 'Ratio_of_more_than_90'

# Check for and handle missing data
borough_connect.dropna(subset=[var1, var2], inplace=True)

# Create weights and row-standardize them
weights = Queen.from_dataframe(borough_connect, use_index=True)
weights.transform = 'r'

# Bivariate Moran's I
moran_bv = Moran_BV(borough_connect[var1], borough_connect[var2], weights)
print(f"Bivariate Moran's I between {var1} and {var2}: {moran_bv.I:.3f}")
print(f"p-value: {moran_bv.p_sim:.3f}")

# Bivariate Moran Plot
fig, ax = plt.subplots(figsize=(10, 5))
spatial_lag_var2 = lag_spatial(weights, borough_connect[var2])  # Calculate the␣
↪spatial lag of var2
scatter = ax.scatter(borough_connect[var1], spatial_lag_var2, color='blue',␣
↪edgecolor='k', alpha=0.7)
fit = np.polyfit(borough_connect[var1], spatial_lag_var2, 1)
ax.plot(borough_connect[var1], np.polyval(fit, borough_connect[var1]),␣
↪color='red', linestyle='--', linewidth=1)
ax.set_title('Bivariate Moran Scatter Plot monthly rent and rule breaking␣
↪Airbnbs')
ax.set_xlabel(var1)
ax.set_ylabel(f"Spatial Lag of {var2}")
plt.savefig('plots/raw/Moran_monthly_rent.png')
plt.show()
```

```
# Bivariate Local Moran's I
local_moran_bv = Moran_Local_BV(borough_connect[var1], borough_connect[var2],␣
 ↪weights)

# LISA Plot (Bivariate)
fig, ax = plt.subplots(figsize=(10, 10))
borough_connect.assign(cl=local_moran_bv.q).plot(column='cl', categorical=True,
                                                  cmap='Paired', linewidth=0.1,␣
 ↪ax=ax,
                                                  edgecolor='white', legend=True)
labels = ['Not Significant', 'Low-Low', 'Low-High', 'High-Low', 'High-High']
legend = ax.get_legend()
if legend:
    legend.set_bbox_to_anchor((1, 1))
    legend.set_title('Cluster Type')
    for text, label in zip(legend.get_texts(), labels):
        text.set_text(label)

ax.set_title('Bivariate LISA Cluster Map of monthly rent and rule breaking␣
 ↪Airbnbs')
ax.set_axis_off()
plt.savefig('plots/raw/LISA_monthly_rent.png')
plt.show()
```

```
Bivariate Moran's I between Monthly_rent_2023 and Ratio_of_more_than_90: 0.397
p-value: 0.001
```

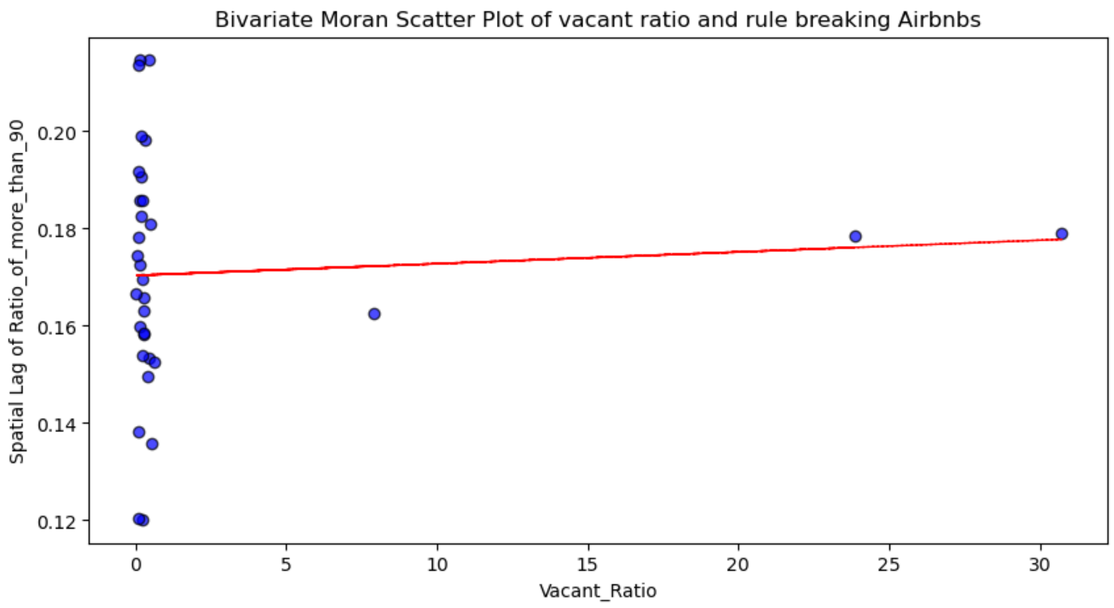


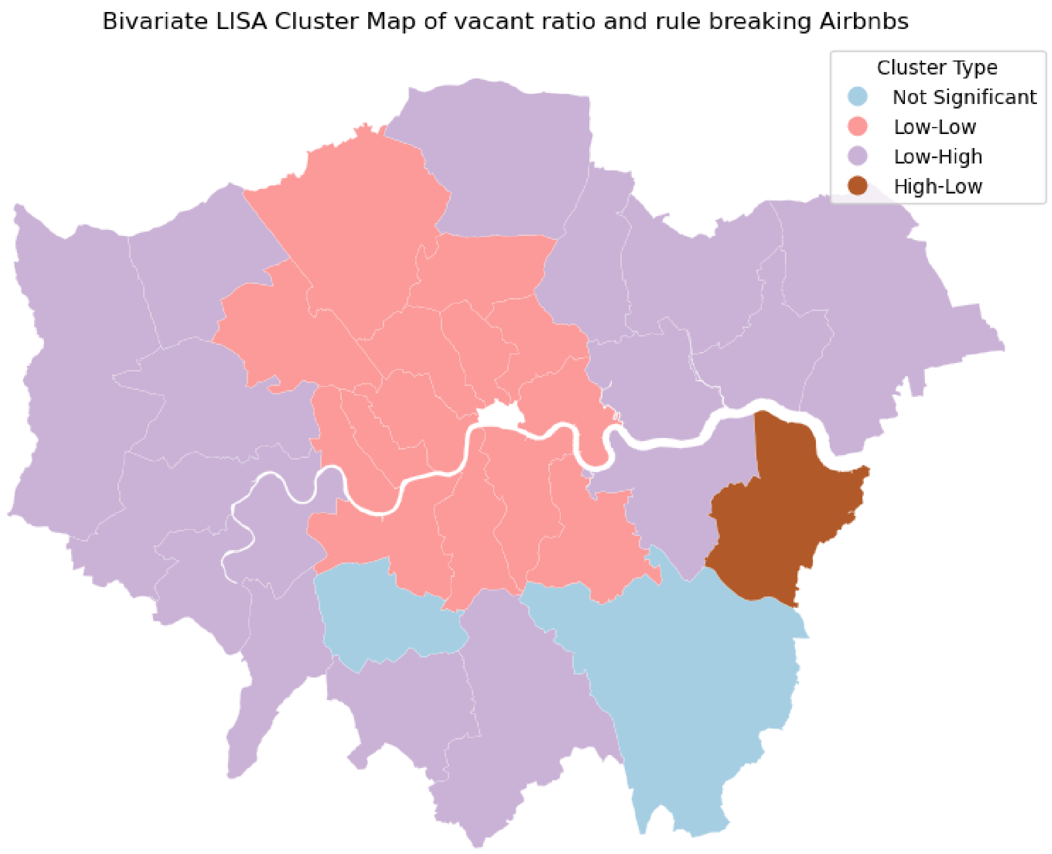

Bivariate LISA Cluster Map of monthly rent and rule breaking Airbnbs
Cluster Type
Not Significant
Low-Low
Low-High
High-Low

```
[14]: #| echo: false
      #| output: false
      # analyse the spatial autocorrelation of vacant ratio and airbnbs breaking the␣
       ↪rule
      # Variables
      var1 = 'Vacant_Ratio'
      var2 = 'Ratio_of_more_than_90'

      # Check for and handle missing data
      borough_connect.dropna(subset=[var1, var2], inplace=True)

      # Create weights and row-standardize them
      weights = Queen.from_dataframe(borough_connect, use_index=True)
      weights.transform = 'r'

      # Bivariate Moran's I
      moran_bv = Moran_BV(borough_connect[var1], borough_connect[var2], weights)
      print(f"Bivariate Moran's I between {var1} and {var2}: {moran_bv.I:.3f}")
      print(f"p-value: {moran_bv.p_sim:.3f}")
```

```
# Bivariate Moran Plot
fig, ax = plt.subplots(figsize=(10, 5))
spatial_lag_var2 = lag_spatial(weights, borough_connect[var2])  # Calculate the␣
↪spatial lag of var2
scatter = ax.scatter(borough_connect[var1], spatial_lag_var2, color='blue',␣
↪edgecolor='k', alpha=0.7)
fit = np.polyfit(borough_connect[var1], spatial_lag_var2, 1)
ax.plot(borough_connect[var1], np.polyval(fit, borough_connect[var1]),␣
↪color='red', linestyle='--', linewidth=1)
ax.set_title('Bivariate Moran Scatter Plot of vacant ratio and rule breaking␣
↪Airbnbs')
ax.set_xlabel(var1)
ax.set_ylabel(f"Spatial Lag of {var2}")
plt.savefig('plots/raw/Moran_vacant_ratio.png')
plt.show()

# Bivariate Local Moran's I
local_moran_bv = Moran_Local_BV(borough_connect[var1], borough_connect[var2],␣
↪weights)

# LISA Plot (Bivariate)
fig, ax = plt.subplots(figsize=(10, 10))
borough_connect.assign(cl=local_moran_bv.q).plot(column='cl', categorical=True,
                                                 cmap='Paired', linewidth=0.1,␣
↪ax=ax,
                                                 edgecolor='white', legend=True)
labels = ['Not Significant', 'Low-Low', 'Low-High', 'High-Low', 'High-High']
legend = ax.get_legend()
if legend:
    legend.set_bbox_to_anchor((1, 1))
    legend.set_title('Cluster Type')
    for text, label in zip(legend.get_texts(), labels):
        text.set_text(label)

ax.set_title('Bivariate LISA Cluster Map of vacant ratio and rule breaking␣
↪Airbnbs')
ax.set_axis_off()
plt.savefig('plots/raw/LISA_vacant_ratio.png')
plt.show()
```

```
Bivariate Moran's I between Vacant_Ratio and Ratio_of_more_than_90: 0.035
p-value: 0.329
```


Bivariate Moran Scatter Plot of vacant ratio and rule breaking Airbnbs
Spatial Lag of Ratio_of_more_than_90
Vacant_Ratio
0.12
0.14
0.16
0.18
0.20
0
5
10
15
20
25
30

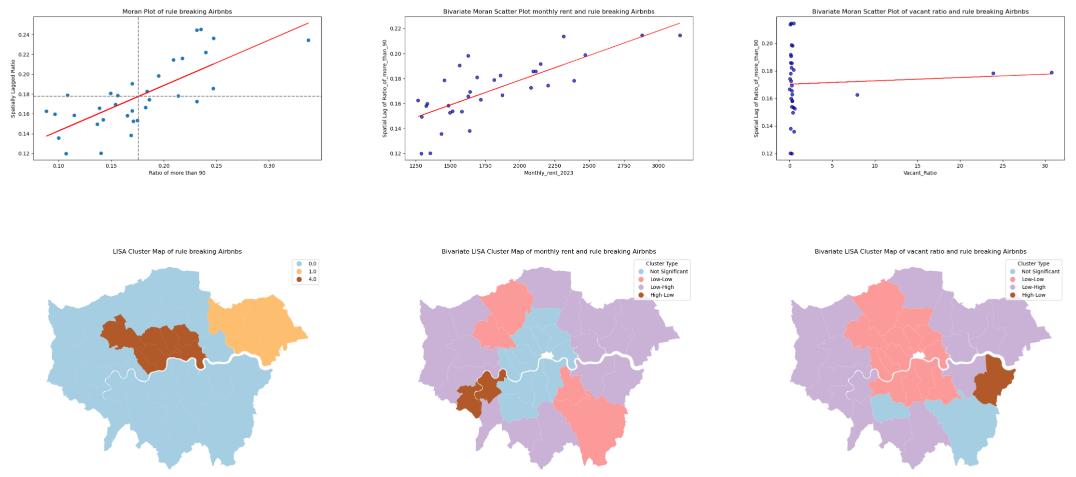

Bivariate LISA Cluster Map of vacant ratio and rule breaking Airbnbs
Cluster Type
Not Significant
Low-Low
Low-High
High-Low

[19]:
```python
#| echo: false
#| output: false
# Plotting the combined figure showing the rusults of Moran scatter plot and␣
 ↪LISA cluster map
from PIL import Image, ImageDraw, ImageFont

# Paths to the images
morans = ['plots/raw/Moran_rule_breaking.png', 'plots/raw/Moran_monthly_rent.
 ↪png', 'plots/raw/Moran_vacant_ratio.png']
lisas = ['plots/raw/LISA_rule_breaking.png', 'plots/raw/LISA_monthly_rent.png',␣
 ↪'plots/raw/LISA_vacant_ratio.png']

# Load all images
images = [Image.open(img) for img in morans + lisas]

# Calculate total width and height for the new image
total_width = images[0].width * 3
max_height = images[0].height + images[3].height

# Create a new image with the appropriate size
new_im = Image.new('RGB', (total_width, max_height))

# Paste each Moran plot into the new image
for i, img in enumerate(images[:3]):  # First three are Moran plots
    new_im.paste(img, (img.width * i, 0))

# Paste each LISA plot into the new image
for i, img in enumerate(images[3:]):  # Last three are LISA plots
    new_im.paste(img, (img.width * i, images[0].height))  # Paste below the␣
 ↪Moran plots

new_im.save('plots/combined_of_Moran_and_LISA.png')
new_im.show()
```

```
/usr/bin/xdg-open: 882: www-browser: not found
/usr/bin/xdg-open: 882: links2: not found
/usr/bin/xdg-open: 882: elinks: not found
/usr/bin/xdg-open: 882: links: not found
/usr/bin/xdg-open: 882: lynx: not found
/usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening '/tmp/tmpqnldmm0j.PNG'
```

[22]:
```python
#| echo: false
import matplotlib.image as mpimg

# Open the image
image = mpimg.imread('plots/combined_of_Moran_and_LISA.png')
```

```
# Display the image
plt.figure(figsize=(30, 15))
plt.imshow(image)
plt.axis('off')  # Hide axes
plt.show()
```



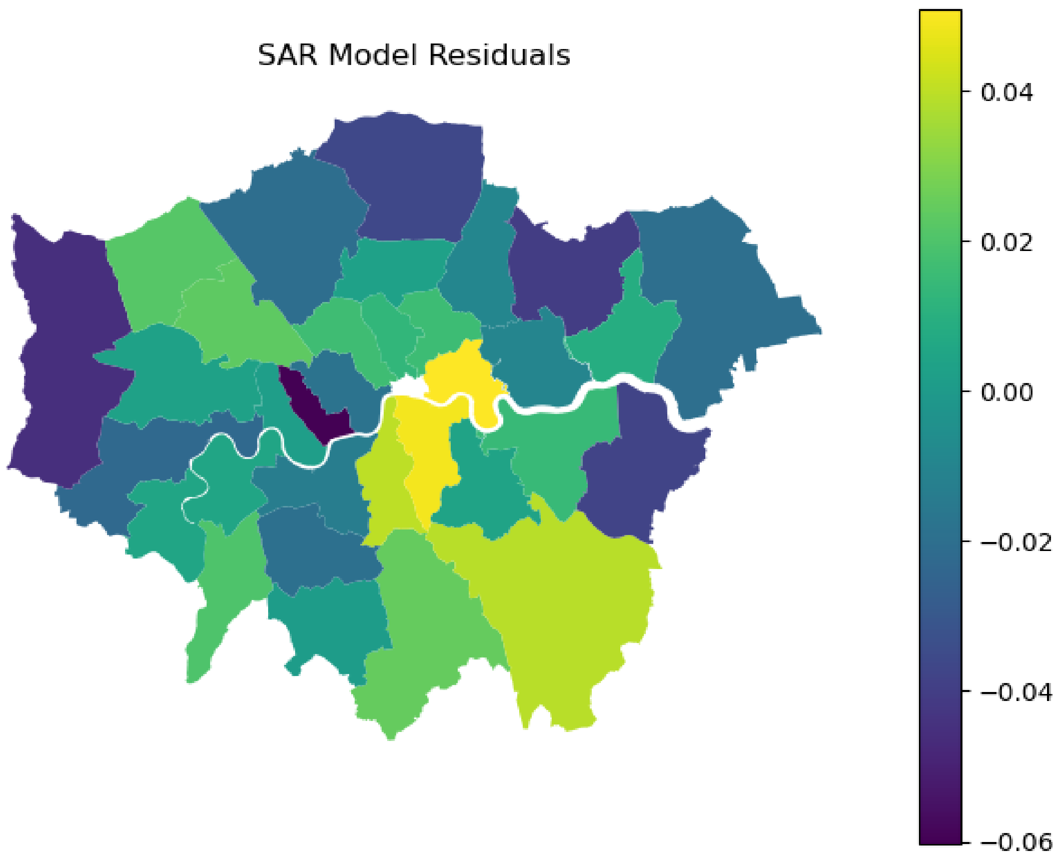
#| echo: false #| output: false # SAR model from spreg import ML_Lag

[43]:
```
#| echo: false
#| output: false
# Import data
data = pd.read_csv("data/connect.csv")
shp = gpd.read_file("data/statistical-gis-boundaries-london/ESRI/
 ↪London_Borough_Excluding_MHW.shp")

# Merge data and transform coordinate system
zone = shp.merge(data, left_on="GSS_CODE", right_on="Borough_code")
zone = zone.to_crs("EPSG:27700")

# Check and remove missing values
columns = ['Monthly_rent_2023', 'Vacant_Ratio', 'Ratio_of_more_than_90']
print("Missing values:\n", zone[columns].isna().sum())
zone = zone.dropna(subset=columns)

# Construct spatial weights matrix
w = Queen.from_dataframe(zone)
w.transform = 'r'
```

```python
# Prepare variables
y = zone['Ratio_of_more_than_90'].values.reshape(-1, 1)
X = zone[['Monthly_rent_2023', 'Vacant_Ratio']].values

# Fit Spatial Lag Model
sar_model = ML_Lag(y, X, w=w,
                   name_y='Ratio_of_more_than_90',
                   name_x=['Monthly_rent_2023', 'Vacant_Ratio'],
                   name_w='w')

# Output model results
print("=== SAR Model Results ===")
print(sar_model.summary)

# Visualize residuals
zone['residuals'] = sar_model.u
fig, ax = plt.subplots(figsize=(8, 6))
zone.plot(column='residuals', cmap='viridis', legend=True, ax=ax)
plt.title("SAR Model Residuals")
plt.axis('off')
plt.show()
```

```
Missing values:
 Monthly_rent_2023        1
Vacant_Ratio            0
Ratio_of_more_than_90    0
dtype: int64
=== SAR Model Results ===
REGRESSION RESULTS
------------------

SUMMARY OF OUTPUT: MAXIMUM LIKELIHOOD SPATIAL LAG (METHOD = FULL)
-----------------------------------------------------------------
Data set            :     unknown
Weights matrix      :           w
Dependent Variable  :Ratio_of_more_than_90                Number of
Observations:          32
Mean dependent var  :      0.1710                Number of Variables   :
4
S.D. dependent var  :      0.0468                Degrees of Freedom    :
28
Pseudo R-squared    :      0.6509
Spatial Pseudo R-squared:  0.6014
Log likelihood      :     69.5087
Sigma-square ML     :      0.0007                Akaike info criterion :
-131.017
S.E of regression   :      0.0272                Schwarz criterion     :
```

```
-125.155

--------------------------------------------------------------------------------
----
            Variable     Coefficient       Std.Error     z-Statistic
Probability
--------------------------------------------------------------------------------
----
            CONSTANT         0.01272         0.02601         0.48893
0.62489
   Monthly_rent_2023         0.00006         0.00001         4.93130
0.00000
        Vacant_Ratio        -0.00161         0.00073        -2.21386
0.02684
W_Ratio_of_more_than_90         0.30154         0.17053         1.76822
0.07702
--------------------------------------------------------------------------------
----

SPATIAL LAG MODEL IMPACTS
Impacts computed using the 'simple' method.
            Variable         Direct        Indirect           Total
   Monthly_rent_2023         0.0001          0.0000          0.0001
        Vacant_Ratio        -0.0016         -0.0007         -0.0023
================================ END OF REPORT
=====================================
```



```
[59]: #| echo: false
      #| output: false
      # GWR model
      from mgwr.gwr import GWR
      from mgwr.sel_bw import Sel_BW
```

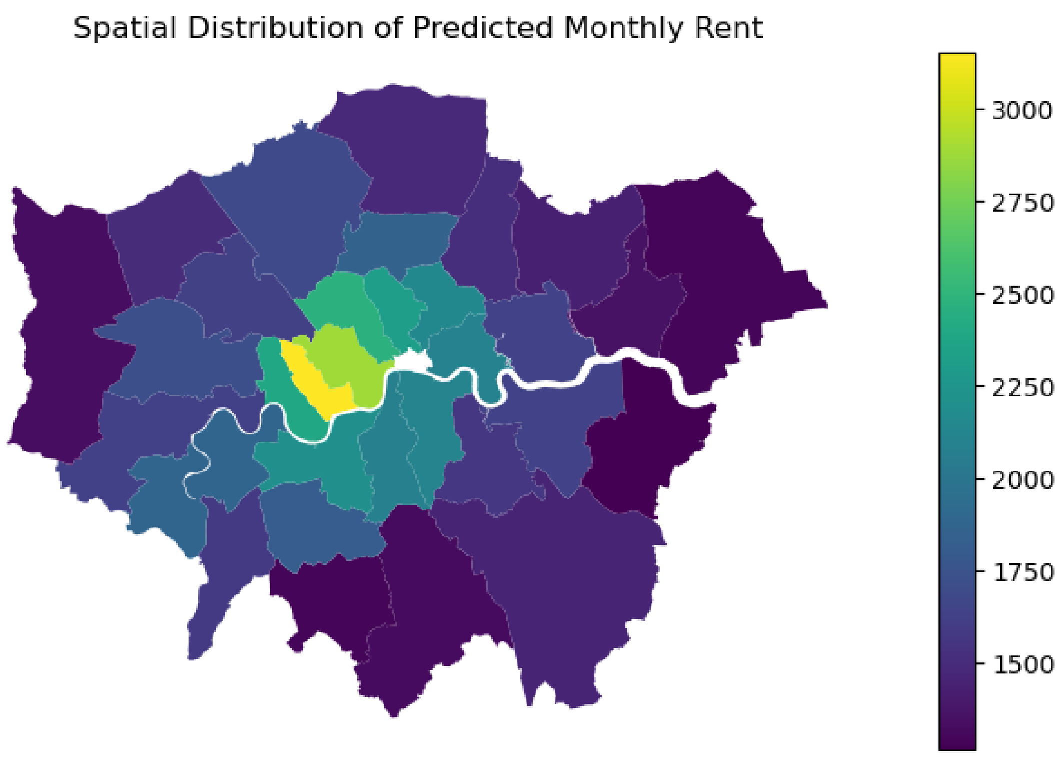
```
[60]: #| echo: false
      #| output: false
      zone = zone.to_crs("EPSG:27700")
      zone['centro'] = zone.geometry.centroid
      zone['X'] = zone['centro'].x
      zone['Y'] = zone['centro'].y
      g_y_rent = zone['Monthly_rent_2023'].values.reshape((-1, 1))
      g_X_rent = zone[['Ratio_of_more_than_90']].values
      g_coords = list(zip(zone['X'], zone['Y']))

      # Automatically set bw_min and bw_max based on the number of observations
      n_obs = len(g_coords)  # Number of observations
      bw_min = 2  # Minimum bandwidth, should be a positive integer
```

```
bw_max = max(bw_min, n_obs - 1)  # Ensures bw_max does not exceed n_obs - 1

# Initialize bandwidth selector with dynamic bandwidth settings
gwr_selector_rent = Sel_BW(g_coords, g_y_rent, g_X_rent, fixed=False)

# Search for optimal bandwidth using the golden section search method
gwr_bw_rent = gwr_selector_rent.search(search_method='golden_section',␣
 ↪criterion='AICc', bw_min=bw_min, bw_max=bw_max)
print('Optimal Bandwidth Size for Rent:', gwr_bw_rent)

# Fit GWR model with the determined optimal bandwidth
gwr_results_rent = GWR(g_coords, g_y_rent, g_X_rent, gwr_bw_rent, fixed=False,␣
 ↪kernel='bisquare').fit()
print(gwr_results_rent.summary())
```

```
Optimal Bandwidth Size for Rent: 27.0
===========================================================================
Model type                                                         Gaussian
Number of observations:                                                  32
Number of covariates:                                                     2

Global Regression Results
---------------------------------------------------------------------------
Residual sum of squares:                                        2861190.859
Log-likelihood:                                                    -227.822
AIC:                                                                459.644
AICc:                                                               462.502
BIC:                                                            2861086.887
R2:                                                                   0.580
Adj. R2:                                                              0.566

Variable                              Est.         SE  t(Est/SE)    p-value
------------------------------- ---------- ---------- ---------- ----------
X0                                 498.858    209.905      2.377      0.017
X1                                7632.613   1185.077      6.441      0.000

Geographically Weighted Regression (GWR) Results
---------------------------------------------------------------------------
Spatial kernel:                                           Adaptive bisquare
Bandwidth used:                                                      27.000

Diagnostic information
---------------------------------------------------------------------------
Residual sum of squares:                                        2397300.358
Effective number of parameters (trace(S)):                            4.965
Degree of freedom (n - trace(S)):                                    27.035
Sigma estimate:                                                     297.780
```

```
Log-likelihood:                                                    -224.992
AIC:                                                                461.913
AICc:                                                               465.232
BIC:                                                                470.656
R2:                                                                   0.648
Adjusted R2:                                                          0.581
Adj. alpha (95%):                                                     0.020
Adj. critical t value (95%):                                          2.450

Summary Statistics For GWR Parameter Estimates
---------------------------------------------------------------------------
Variable                   Mean        STD        Min     Median        Max
-------------------- ---------- ---------- ---------- ---------- ----------
X0                      523.732     79.898    379.795    523.955    719.631
X1                     7712.747    485.361   6923.059   7671.554   8700.834
===========================================================================

None
```

[61]:
```
#| echo: false
#| output: false
fig, ax = plt.subplots(figsize=(10, 5))
zone.plot(column='Monthly_rent_2023', cmap='viridis', legend=True, ax=ax)
ax.set_title("Spatial Distribution of Predicted Monthly Rent")
ax.set_axis_off()
plt.show()
```



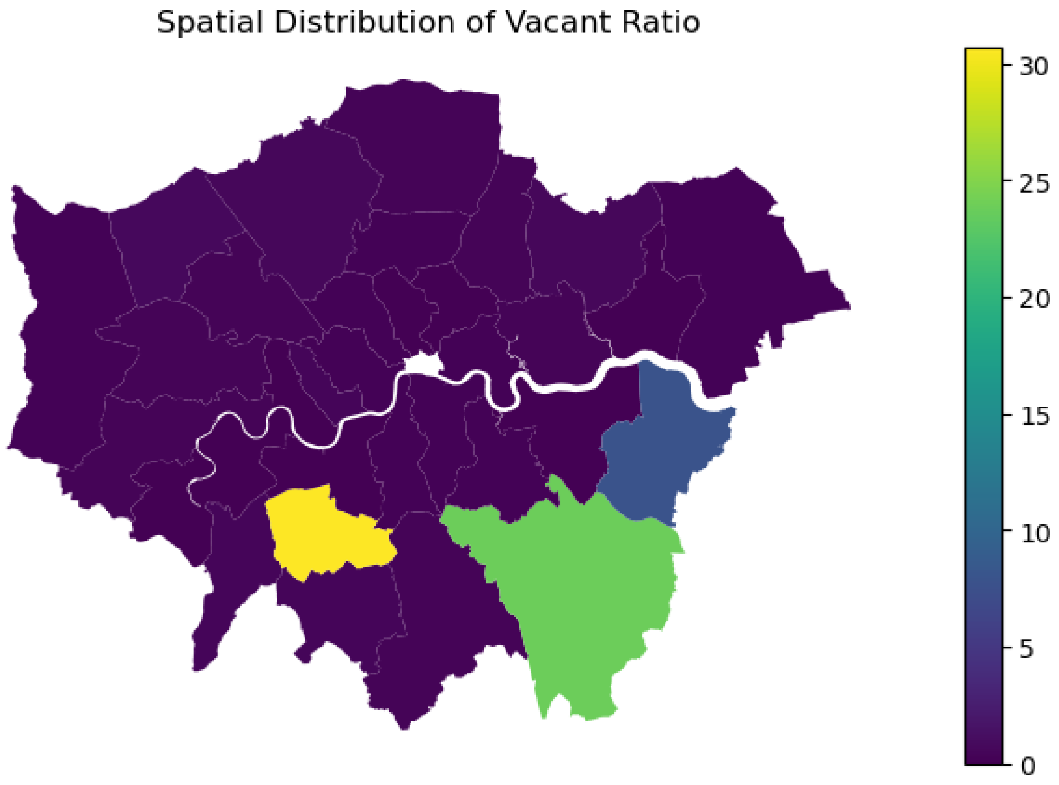
[62]:
```
#| echo: false
#| output: false
g_coords = list(zip(zone['X'], zone['Y']))

# Define independent and dependent variables for the Vacant_Ratio model
g_y_vacant = zone['Vacant_Ratio'].values.reshape((-1, 1))
g_X_vacant = zone[['Ratio_of_more_than_90']].values

# Automatically set bw_min and bw_max based on the number of observations
n_obs = len(g_coords)  # Number of observations
bw_min = 2  # Minimum bandwidth, should be a positive integer
bw_max = max(bw_min, n_obs - 1)  # Ensures bw_max does not exceed n_obs - 1

# Initialize bandwidth selector with dynamic bandwidth settings for Vacant_Ratio
gwr_selector_vacant = Sel_BW(g_coords, g_y_vacant, g_X_vacant, fixed=False)

# Search for optimal bandwidth using the golden section search method for␣
 ↪Vacant_Ratio
gwr_bw_vacant = gwr_selector_vacant.search(search_method='golden_section',␣
 ↪criterion='AICc', bw_min=bw_min, bw_max=bw_max)
print('Optimal Bandwidth Size for Vacant Ratio:', gwr_bw_vacant)
```

```python
# Fit GWR model with the determined optimal bandwidth for Vacant_Ratio
gwr_results_vacant = GWR(g_coords, g_y_vacant, g_X_vacant, gwr_bw_vacant,␣
 ↪fixed=False, kernel='bisquare').fit()
print(gwr_results_vacant.summary())
```

```
Optimal Bandwidth Size for Vacant Ratio: 28.0
===========================================================================
Model type                                                         Gaussian
Number of observations:                                                  32
Number of covariates:                                                     2

Global Regression Results
---------------------------------------------------------------------------
Residual sum of squares:                                           1298.658
Log-likelihood:                                                    -104.660
AIC:                                                                213.319
AICc:                                                               216.176
BIC:                                                               1194.686
R2:                                                                   0.090
Adj. R2:                                                              0.060

Variable                              Est.         SE  t(Est/SE)    p-value
------------------------------- ---------- ---------- ---------- ----------
X0                                   9.627      4.472      2.153      0.031
X1                                 -43.587     25.248     -1.726      0.084

Geographically Weighted Regression (GWR) Results
---------------------------------------------------------------------------
Spatial kernel:                                           Adaptive bisquare
Bandwidth used:                                                      28.000

Diagnostic information
---------------------------------------------------------------------------
Residual sum of squares:                                            868.773
Effective number of parameters (trace(S)):                            4.765
Degree of freedom (n - trace(S)):                                    27.235
Sigma estimate:                                                       5.648
Log-likelihood:                                                     -98.228
AIC:                                                                207.985
AICc:                                                               211.076
BIC:                                                                216.435
R2:                                                                   0.391
Adjusted R2:                                                          0.281
Adj. alpha (95%):                                                     0.021
Adj. critical t value (95%):                                          2.432

Summary Statistics For GWR Parameter Estimates
---------------------------------------------------------------------------
```

```
Variable                   Mean        STD        Min     Median        Max
-------------------- ---------- ---------- ---------- ---------- ----------
X0                       12.145      8.322      0.633      9.808     27.976
X1                      -55.655     39.325   -132.867    -41.327     -2.029
===========================================================================

None
```

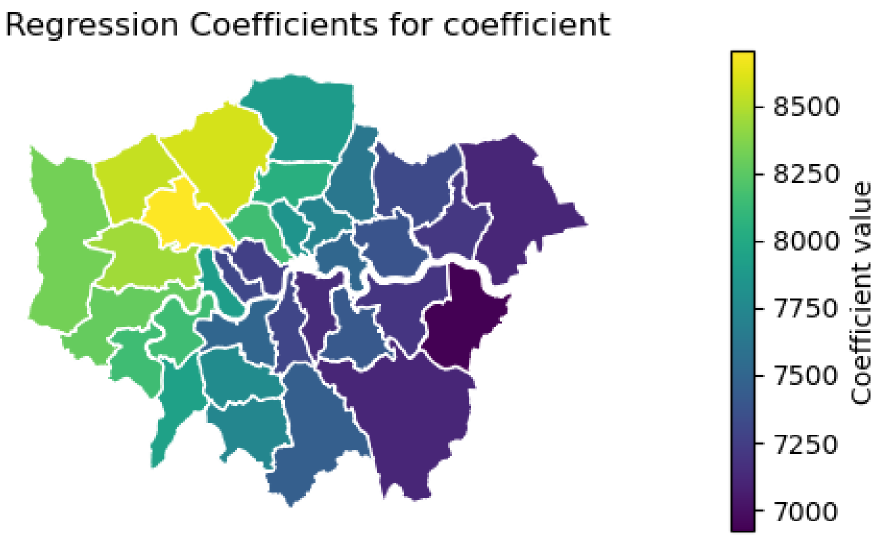
[63]:
```
#| echo: false
#| output: false
fig, ax = plt.subplots(figsize=(10, 5))
zone.plot(column='Vacant_Ratio', cmap='viridis', legend=True, ax=ax)
ax.set_title("Spatial Distribution of Vacant Ratio")
ax.set_axis_off()
plt.show()
```



[64]:
```
#| echo: false
#| output: false
zone['coefficient'] = gwr_results_rent.params[:, 1]  # Add coefficients
zone['t_values'] = gwr_results_rent.tvalues[:, 1]  # Add t-values
```

[65]:

```
#| echo: false
#| output: false
# Define the variable names to be visualized, corresponding to the regression␣
↪results added
var_names = ['coefficient']  # Adjust this if more variables from the model␣
↪should be visualized

fig, axes = plt.subplots(1, len(var_names), figsize=(12, 3))

# Ensure `axes` is iterable
if len(var_names) == 1:
    axes = [axes]

for i, var in enumerate(var_names):
    ax = axes[i]  # Access each subplot axis
    zone.plot(column=var, cmap='viridis', legend=True, ax=ax,␣
↪edgecolor='white', legend_kwds={'label': "Coefficient value"})
    ax.set_title(f'Regression Coefficients for {var}')
    ax.set_axis_off()

    # Highlight non-significant areas based on a significance threshold
    threshold = 1.96
    non_significant = zone['t_values'].abs() < threshold  # Ensuring the use of␣
↪absolute value for significance checking
    zone.loc[non_significant].plot(ax=ax, color='lightgrey', edgecolor='white')

plt.tight_layout()
plt.show()
```

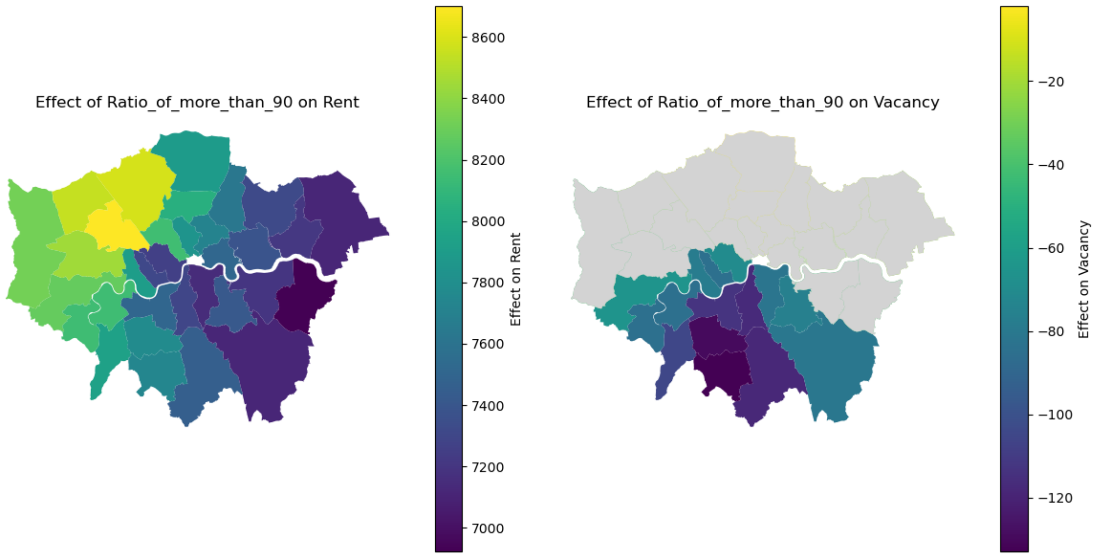


[66]:
```
#| echo: false
#| output: false
# Fit GWR for Monthly_rent_2023
gwr_model_rent = GWR(g_coords, zone['Monthly_rent_2023'].values.reshape((-1,␣
 ↪1)),
                     zone[['Ratio_of_more_than_90']].values.reshape((-1, 1)),␣
 ↪gwr_bw_rent).fit()

# Fit GWR for Vacant_Ratio
gwr_model_vacant = GWR(g_coords, zone['Vacant_Ratio'].values.reshape((-1, 1)),
                       zone[['Ratio_of_more_than_90']].values.reshape((-1, 1)),␣
 ↪gwr_bw_vacant).fit()

# Extract coefficients and t-values for each model
rent_coefs = pd.DataFrame(gwr_model_rent.params, columns=['Intercept',␣
 ↪'Effect_of_Ratio_of_more_than_90_on_Rent'])
rent_tvals = pd.DataFrame(gwr_model_rent.tvalues, columns=['t_Intercept',␣
 ↪'t_Effect_on_Rent'])

vacant_coefs = pd.DataFrame(gwr_model_vacant.params, columns=['Intercept',␣
 ↪'Effect_of_Ratio_of_more_than_90_on_Vacancy'])
vacant_tvals = pd.DataFrame(gwr_model_vacant.tvalues, columns=['t_Intercept',␣
 ↪'t_Effect_on_Vacancy'])
```

[67]:
```
#| echo: false
#| output: false
# Add results directly to zone GeoDataFrame
zone['Rent_Effect'] = rent_coefs['Effect_of_Ratio_of_more_than_90_on_Rent']
zone['Vacancy_Effect'] =␣
 ↪vacant_coefs['Effect_of_Ratio_of_more_than_90_on_Vacancy']

# Check significance and add to zone
zone['Significant_Rent'] = rent_tvals['t_Effect_on_Rent'].abs() > 1.96
zone['Significant_Vacancy'] = vacant_tvals['t_Effect_on_Vacancy'].abs() > 1.96
```

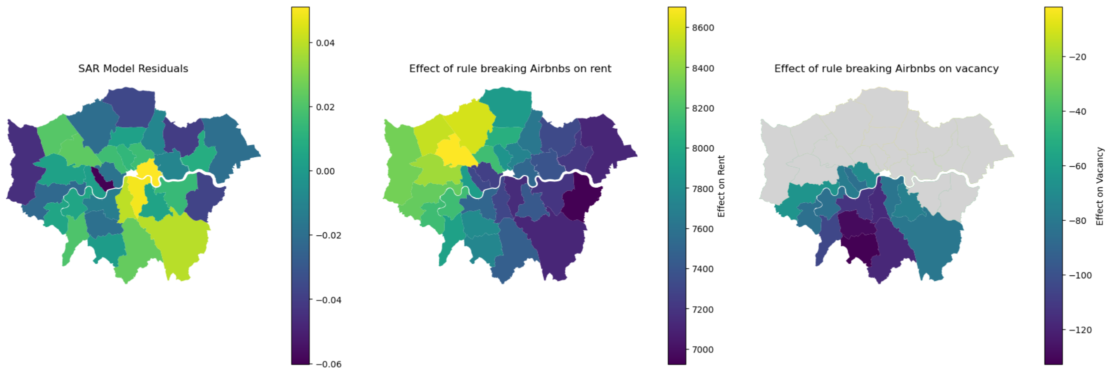
[68]:
```
#| echo: false
#| output: false
fig, ax = plt.subplots(1, 2, figsize=(12, 6))

# Plot for Rent
zone.plot(column='Rent_Effect', cmap='viridis', ax=ax[0], legend=True,
          legend_kwds={'label': "Effect on Rent"})
zone[~zone['Significant_Rent']].plot(color='lightgrey', ax=ax[0])
ax[0].set_title('Effect of Ratio_of_more_than_90 on Rent')
ax[0].set_axis_off()

# Plot for Vacancy
```

```
zone.plot(column='Vacancy_Effect', cmap='viridis', ax=ax[1], legend=True,
          legend_kwds={'label': "Effect on Vacancy"})
zone[~zone['Significant_Vacancy']].plot(color='lightgrey', ax=ax[1])
ax[1].set_title('Effect of Ratio_of_more_than_90 on Vacancy')
ax[1].set_axis_off()

plt.tight_layout()
plt.show()
```



[70]:
```
#| echo: false
#| output: false
# combing the plots to a new plot
zone['residuals'] = sar_model.u

# Create a figure with three subplots (one row, three columns)
fig, ax = plt.subplots(1, 3, figsize=(18, 6))  # Adjust the figure size as␣
 ↪needed

# Plot for Residuals
zone.plot(column='residuals', cmap='viridis', ax=ax[0], legend=True)
ax[0].set_title('SAR Model Residuals')
ax[0].set_axis_off()

# Plot for Rent Effect
zone.plot(column='Rent_Effect', cmap='viridis', ax=ax[1], legend=True,␣
 ↪legend_kwds={'label': "Effect on Rent"})
zone[~zone['Significant_Rent']].plot(color='lightgrey', ax=ax[1])
ax[1].set_title('Effect of rule breaking Airbnbs on rent')
```

```
ax[1].set_axis_off()

# Plot for Vacancy Effect
zone.plot(column='Vacancy_Effect', cmap='viridis', ax=ax[2], legend=True,␣
 ↪legend_kwds={'label': "Effect on Vacancy"})
zone[~zone['Significant_Vacancy']].plot(color='lightgrey', ax=ax[2])
ax[2].set_title('Effect of rule breaking Airbnbs on vacancy')
ax[2].set_axis_off()

#output
plt.savefig('plots/Results_of_SAR_and_GWR_model.png', dpi=600,␣
 ↪bbox_inches='tight')

# Adjust layout
plt.tight_layout()
plt.show()
```



```
[ ]:
```