Spatial Autocorrelation Analysis in R

Yanan Wu

2025-04-09

Contents

1. Introduction to Spatial Autocorrelation

Spatial autocorrelation measures the degree to which a variable is correlated with itself across space. In other words, it quantifies whether observations close to each other are more similar (positive autocorrelation) or more different (negative autocorrelation) than would be expected by random chance.

As Waldo Tobler's First Law of Geography states: > "Everything is related to everything else, but near things are more related than distant things."

This fundamental concept drives spatial autocorrelation analysis - the idea that geographic proximity often correlates with similarity in characteristics. For example, in population data, we often see spatial clusters of similar values because people tend to reside in similar neighborhoods due to factors like housing prices, proximity to workplaces, and cultural factors.

Why is Spatial Autocorrelation Important?

Understanding spatial autocorrelation is crucial because:

1. It indicates the presence of spatial dependence in your data
2. It can reveal underlying spatial processes
3. It may violate the independence assumption of standard statistical models
4. It can help identify spatial clusters or hotspots

Types of Spatial Autocorrelation

- **Positive spatial autocorrelation**: Similar values cluster together (high near high, low near low)
- **Negative spatial autocorrelation**: Dissimilar values cluster together (high near low)
- **No spatial autocorrelation**: Values are randomly distributed in space

2. Setting Up the Environment

Let's start by installing and loading the necessary packages:

```r
# Install necessary packages (if not already installed)
packages <- c("sf", "spdep", "tmap", "tmaptools", "spData", "dplyr",
              "ggplot2", "classInt", "RColorBrewer", "viridis", "knitr")
install.packages(packages[!packages %in% installed.packages()[,"Package"]],
                 repos = "https://cran.r-project.org/")

# Load the packages
library(tmap)        # For thematic maps
library(tmaptools)   # Additional mapping tools
library(dplyr)       # For data manipulation
library(ggplot2)     # For plotting
library(classInt)    # For classification
library(RColorBrewer) # For color palettes
library(viridis)     # For color palettes
library(knitr)       # For nice table output
library(sp)

# Set tmap to plot mode
# tmap_mode("plot")
```

3. Loading and Exploring Spatial Data

For this tutorial, we'll use the `columbus` dataset from the `spData` package. This dataset contains 49 neighborhoods in Columbus, Ohio with various socioeconomic variables, including crime rates, housing values, and income.

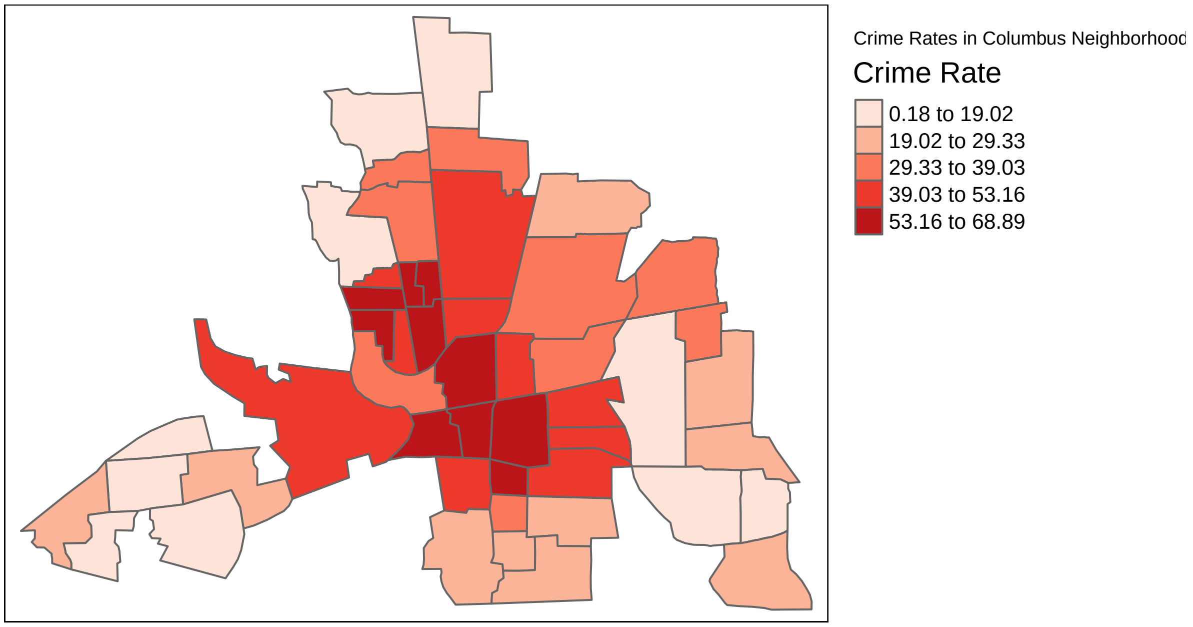
```r
# Load required packages
library(sf)
library(spdep)
library(spData)

# Load the Columbus polygon shapefile correctly
columbus_poly <- st_read(system.file("shapes/columbus.shp", package="spData"))
```

```
## Reading layer ‘columbus’ from data source
##   ‘C:\Users\yyang\AppData\Local\R\win-library\4.3\spData\shapes\columbus.shp’
##   using driver ‘ESRI Shapefile’
## Simple feature collection with 49 features and 20 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: 5.874907 ymin: 10.78863 xmax: 11.28742 ymax: 14.74245
## CRS:           NA
```

```r
# View the first few rows
head(columbus_poly)
```

```
## Simple feature collection with 6 features and 20 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: 7.950089 ymin: 12.86109 xmax: 10.1806 ymax: 14.74245
## CRS:           NA
##       AREA PERIMETER COLUMBUS_ COLUMBUS_I POLYID NEIG  HOVAL    INC    CRIME
## 1 0.309441  2.440629         2          5      1    5 80.467 19.531 15.72598
## 2 0.259329  2.236939         3          1      2    1 44.567 21.232 18.80175
## 3 0.192468  2.187547         4          6      3    6 26.350 15.956 30.62678
## 4 0.083841  1.427635         5          2      4    2 33.200  4.477 32.38776
## 5 0.488888  2.997133         6          7      5    7 23.225 11.252 50.73151
## 6 0.283079  2.335634         7          8      6    8 28.750 16.029 26.06666
##       OPEN    PLUMB DISCBD     X     Y NSA  NSB  EW  CP THOUS NEIGNO
## 1 2.850747 0.217155   5.03 38.80 44.07   1    1   1   0  1000   1005
## 2 5.296720 0.320581   4.27 35.62 42.38   1    1   0   0  1000   1001
## 3 4.534649 0.374404   3.89 39.82 41.18   1    1   1   0  1000   1006
## 4 0.394427 1.186944   3.70 36.50 40.52   1    1   0   0  1000   1002
## 5 0.405664 0.624596   2.83 40.01 38.00   1    1   1   0  1000   1007
## 6 0.563075 0.254130   3.78 43.75 39.28   1    1   1   0  1000   1008
##                         geometry
## 1 POLYGON ((8.624129 14.23698...
## 2 POLYGON ((8.25279 14.23694,...
## 3 POLYGON ((8.653305 14.00809...
## 4 POLYGON ((8.459499 13.82035...
## 5 POLYGON ((8.685274 13.63952...
## 6 POLYGON ((9.401384 13.5504,...
```

```r
# Create a simple map to visualize the areas
tm_shape(columbus_poly) +
  tm_fill("CRIME", style = "quantile", palette = "Reds", title = "Crime Rate") +
  tm_borders() +
  tm_layout(title = "Crime Rates in Columbus Neighborhoods",
            legend.outside = TRUE)
```



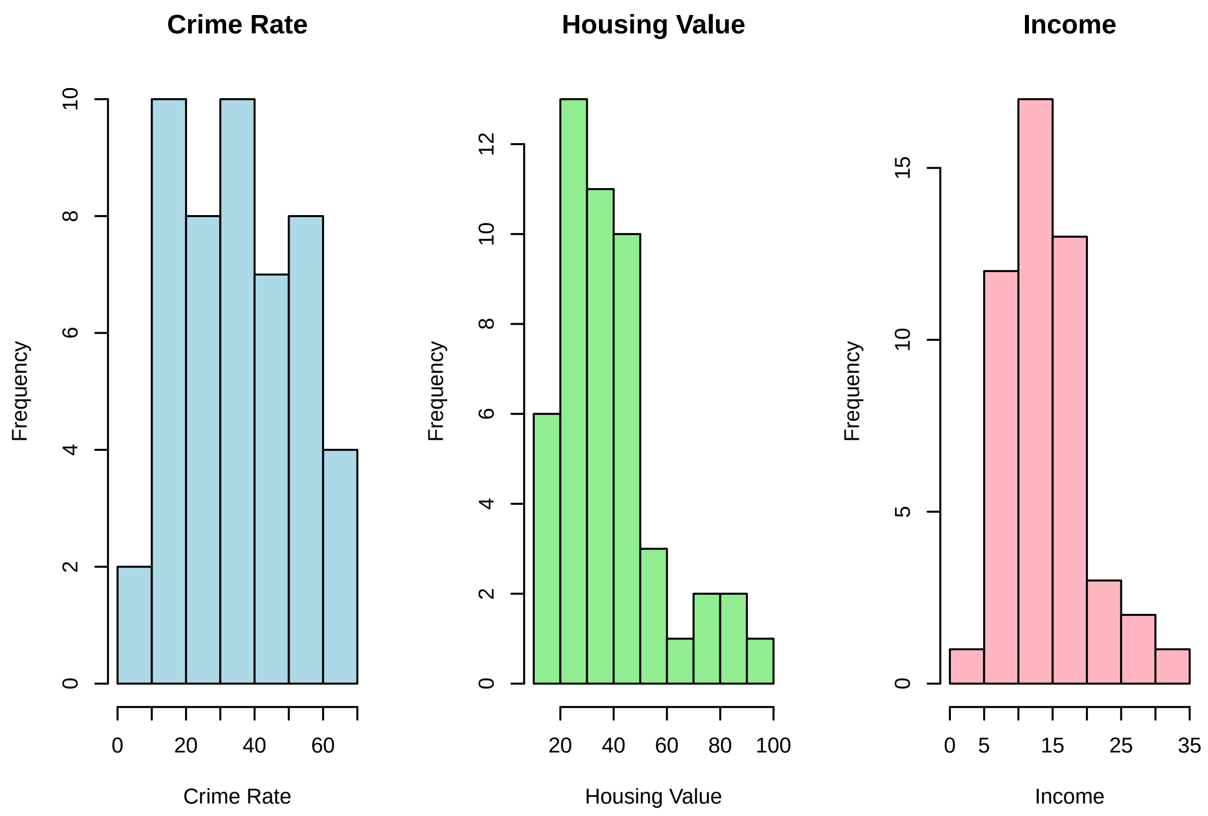
The Columbus dataset includes the following key variables: - `CRIME`: Crime rates - `HOVAL`: Housing values - `INC`: Income

Let's explore the distribution of these variables:

```r
# Summary statistics
summary(columbus_poly[, c("CRIME", "HOVAL", "INC")])
```

```
##      CRIME             HOVAL            INC            geometry
##  Min.   : 0.1783   Min.   :17.90   Min.   : 4.477   POLYGON:49
##  1st Qu.:20.0485   1st Qu.:25.70   1st Qu.: 9.963   epsg:NA: 0
##  Median :34.0008   Median :33.50   Median :13.380
##  Mean   :35.1288   Mean   :38.44   Mean   :14.375
##  3rd Qu.:48.5855   3rd Qu.:43.30   3rd Qu.:18.324
##  Max.   :68.8920   Max.   :96.40   Max.   :31.070
```

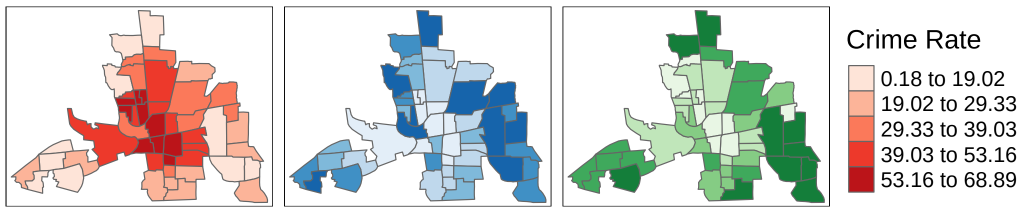
```r
# Histograms of key variables
par(mfrow = c(1, 3))
hist(columbus_poly$CRIME, main = "Crime Rate", xlab = "Crime Rate", col = "lightblue")
hist(columbus_poly$HOVAL, main = "Housing Value", xlab = "Housing Value", col = "lightgreen")
hist(columbus_poly$INC, main = "Income", xlab = "Income", col = "lightpink")
```



```r
par(mfrow = c(1, 1))

# Create a series of choropleth maps for key variables
```

```r
tm_shape(columbus_poly) +
  tm_fill(c("CRIME", "HOVAL", "INC"),
          style = "quantile",
          palette = list("Reds", "Blues", "Greens"),
          title = c("Crime Rate", "Housing Value", "Income")) +
  tm_borders() +
  tm_facets(ncol = 3) +
  tm_layout(legend.outside = TRUE)
```

```
## Warning: Currect projection of shape columbus_poly unknown. Long-lat (WGS84) is
## assumed.
```



Looking at the maps, we can visually assess potential spatial patterns. For instance, we might observe that high crime areas tend to cluster together, or that neighborhoods with higher housing values are spatially proximate. However, to quantify these patterns, we need formal spatial autocorrelation analysis.

4. Creating Spatial Weights

Before we can analyze spatial autocorrelation, we need to define neighborhood relationships between areas. This is done using a spatial weights matrix.

Types of Spatial Weights

There are several ways to define neighborhood relationships:

1. **Contiguity-based**: Areas that share boundaries
    - Queen: Areas that share any point (borders or corners)
    - Rook: Areas that share borders only
2. **Distance-based**: Areas within a certain distance of each other
    - Fixed distance band
    - K-nearest neighbors
3. **General weights**: Custom weights based on other criteria

Queen Contiguity Weights

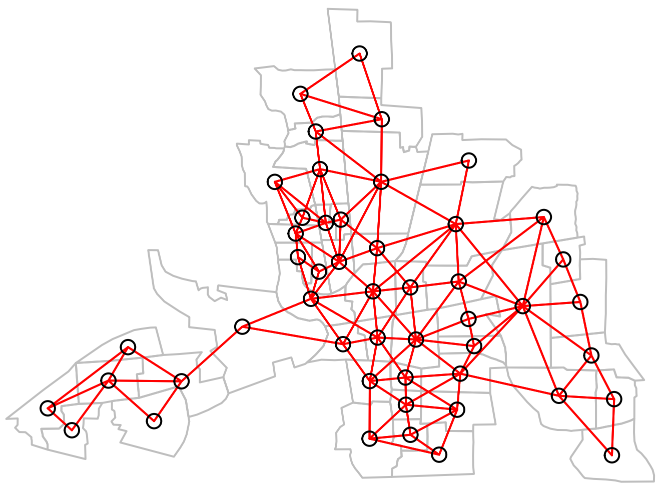
Let's create queen contiguity weights, where regions are neighbors if they share even a single boundary point:

```r
# Create queen contiguity weights
queen_nb <- poly2nb(columbus_poly, queen = TRUE)
summary(queen_nb)
```

```
## Neighbour list object:
## Number of regions: 49
## Number of nonzero links: 236
## Percentage nonzero weights: 9.829238
## Average number of links: 4.816327
## Link number distribution:
##
##  2  3  4  5  6  7  8  9 10
##  5  9 12  5  9  3  4  1  1
## 5 least connected regions:
## 1 6 42 46 47 with 2 links
## 1 most connected region:
## 20 with 10 links
```

```r
# Extract coordinates
columbus_sp <- as(columbus_poly, "Spatial")
coords <- coordinates(columbus_sp)

# Plot the queen contiguity network

plot(columbus_sp, border = "grey")
plot(queen_nb, coords, add = TRUE, col = "red", lwd = 1)
title("Queen Contiguity Neighbors")
```

Queen Contiguity Neighbors



```r
# Convert to weights list object
queen_weights <- nb2listw(queen_nb, style = "W")
```

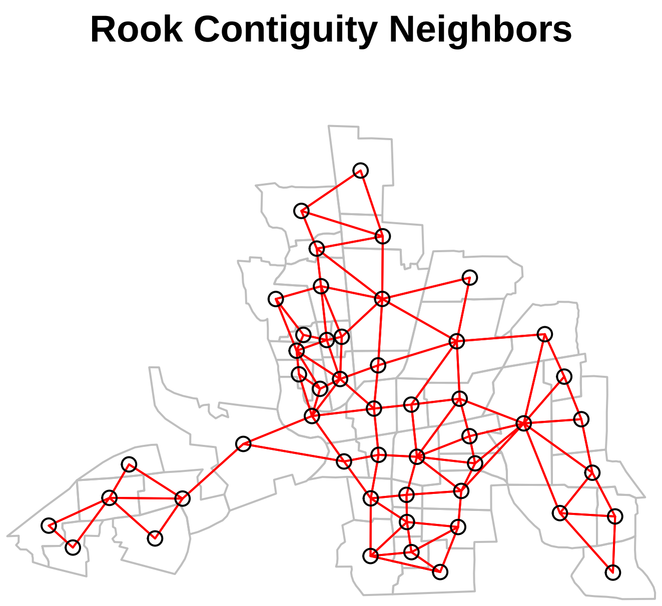
Rook Contiguity Weights

Alternatively, we can create rook contiguity weights, where regions are neighbors only if they share a border segment:

```r
# Create rook contiguity weights
rook_nb <- poly2nb(columbus_poly, queen = FALSE)
summary(rook_nb)
```

```
## Neighbour list object:
## Number of regions: 49
## Number of nonzero links: 200
## Percentage nonzero weights: 8.329863
## Average number of links: 4.081633
## Link number distribution:
##
##  2  3  4  5  6  7  9
##  7 10 17  8  3  3  1
## 7 least connected regions:
## 1 6 31 39 42 46 47 with 2 links
## 1 most connected region:
## 20 with 9 links
```

```
# Plot the rook contiguity network
plot(columbus_sp, border = "grey")
plot(rook_nb, coords, add = TRUE, col = "red", lwd = 1)
title("Rook Contiguity Neighbors")
```



```
# Convert to weights list object
rook_weights <- nb2listw(rook_nb, style = "W")
```

Distance-Based Weights

We can also create distance-based weights, where neighborhoods within a certain distance are considered neighbors:

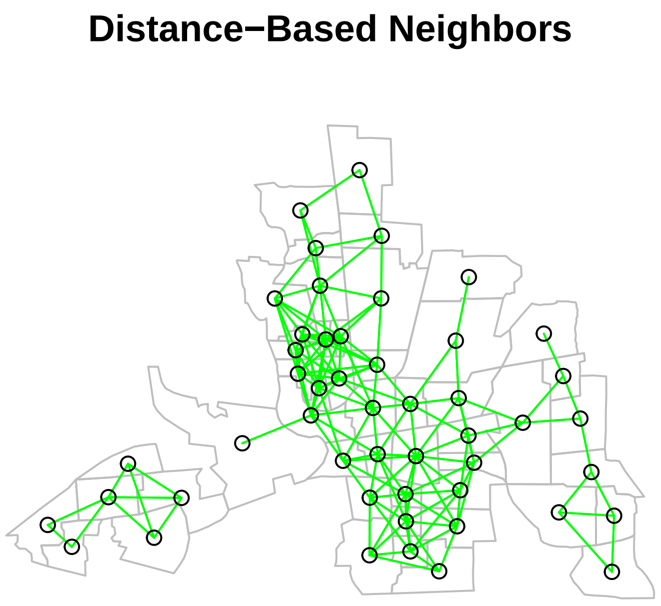
```
# Create distance-based neighbors
# First, determine an appropriate distance threshold
coords_dist <- as.matrix(dist(coords))
knn1_dist <- apply(coords_dist, 1, function(x) sort(x)[2])
max_knn1 <- max(knn1_dist)
max_knn1
```

```
## [1] 0.6188642
```

```r
# Create distance-based neighbors list with a threshold that ensures all areas have at least one neighb
dist_nb <- dnearneigh(coords, 0, max_knn1 * 1.1)
summary(dist_nb)
```

```
## Neighbour list object:
## Number of regions: 49
## Number of nonzero links: 286
## Percentage nonzero weights: 11.9117
## Average number of links: 5.836735
## 2 disjoint connected subgraphs
## Link number distribution:
##
##  1  2  3  4  5  6  7  8  9 10 11 12
##  3  4 10  3  4  2  7  2  8  3  2  1
## 3 least connected regions:
## 6 10 21 with 1 link
## 1 most connected region:
## 28 with 12 links
```

```r
# Plot the distance-based neighbor network
plot(columbus_sp, border = "grey")
plot(dist_nb, coords, add = TRUE, col = "green", lwd = 1)
title("Distance-Based Neighbors")
```



```
# Convert to weights list object
dist_weights <- nb2listw(dist_nb, style = "W")
```

K-Nearest Neighbors Weights

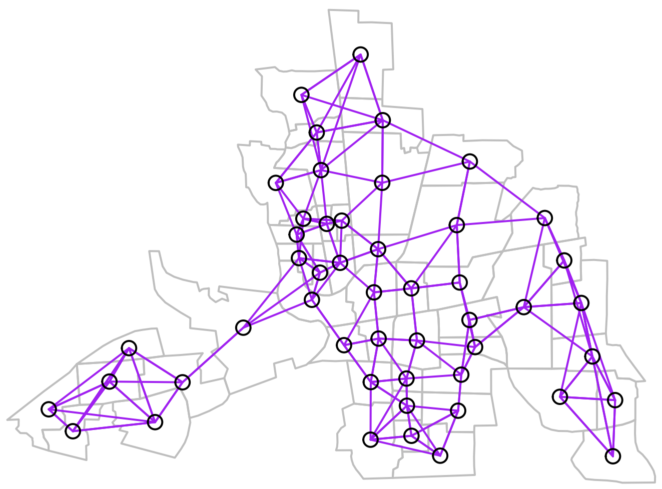
Another approach is to use k-nearest neighbors, where each region is connected to its k closest neighbors:

```
# Create k-nearest neighbors (k=4)
knn_nb <- knn2nb(knearneigh(coords, k = 4))
summary(knn_nb)
```

```
## Neighbour list object:
## Number of regions: 49
## Number of nonzero links: 196
## Percentage nonzero weights: 8.163265
## Average number of links: 4
## Non-symmetric neighbours list
## Link number distribution:
##
##  4
## 49
## 49 least connected regions:
## 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 3
## 49 most connected regions:
## 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 3
```

```
# Plot the k-nearest neighbors network
plot(columbus_sp, border = "grey")
plot(knn_nb, coords, add = TRUE, col = "purple", lwd = 1)
title("K-Nearest Neighbors (k=4)")
```

K−Nearest Neighbors (k=4)



```r
# Convert to weights list object
knn_weights <- nb2listw(knn_nb, style = "W")
```

5. Global Spatial Autocorrelation

Global spatial autocorrelation measures the overall degree of similarity among spatially close observations. Moran's I is the most common statistic for this purpose.

Moran's I

Moran's I ranges from -1 to 1: - Values close to 1 indicate positive spatial autocorrelation (clustering) - Values close to -1 indicate negative spatial autocorrelation (dispersion) - Values close to 0 indicate random spatial pattern

Let's calculate Moran's I for crime rates:

```r
# Calculate Moran's I for crime rates using queen weights
moran_crime_queen <- moran.test(columbus_sp$CRIME, queen_weights)
print(moran_crime_queen)
```

```
##
##  Moran I test under randomisation
##
```

```
## data:  columbus_sp$CRIME
## weights: queen_weights
##
## Moran I statistic standard deviate = 5.5894, p-value = 1.139e-08
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##       0.500188557      -0.020833333       0.008689289
```

```
# Calculate Moran's I for housing values
moran_hoval_queen <- moran.test(columbus_sp$HOVAL, queen_weights)
print(moran_hoval_queen)
```

```
##
##  Moran I test under randomisation
##
## data:  columbus_sp$HOVAL
## weights: queen_weights
##
## Moran I statistic standard deviate = 2.2071, p-value = 0.01365
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##       0.180093114      -0.020833333       0.008287783
```

```
# Calculate Moran's I for income
moran_inc_queen <- moran.test(columbus_sp$INC, queen_weights)
print(moran_inc_queen)
```

```
##
##  Moran I test under randomisation
##
## data:  columbus_sp$INC
## weights: queen_weights
##
## Moran I statistic standard deviate = 4.7645, p-value = 9.467e-07
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##       0.415628778      -0.020833333       0.008391926
```

Interpreting Moran's I

The output of the Moran's I test includes:

1. **Moran I statistic**: The actual I value, ranging from -1 to 1
2. **Expected value**: The expected value of I under the null hypothesis of no spatial autocorrelation (usually close to 0)
3. **Variance**: The variance of I
4. **Z-score**: The standard score, indicating how many standard deviations the I statistic is from the expected value

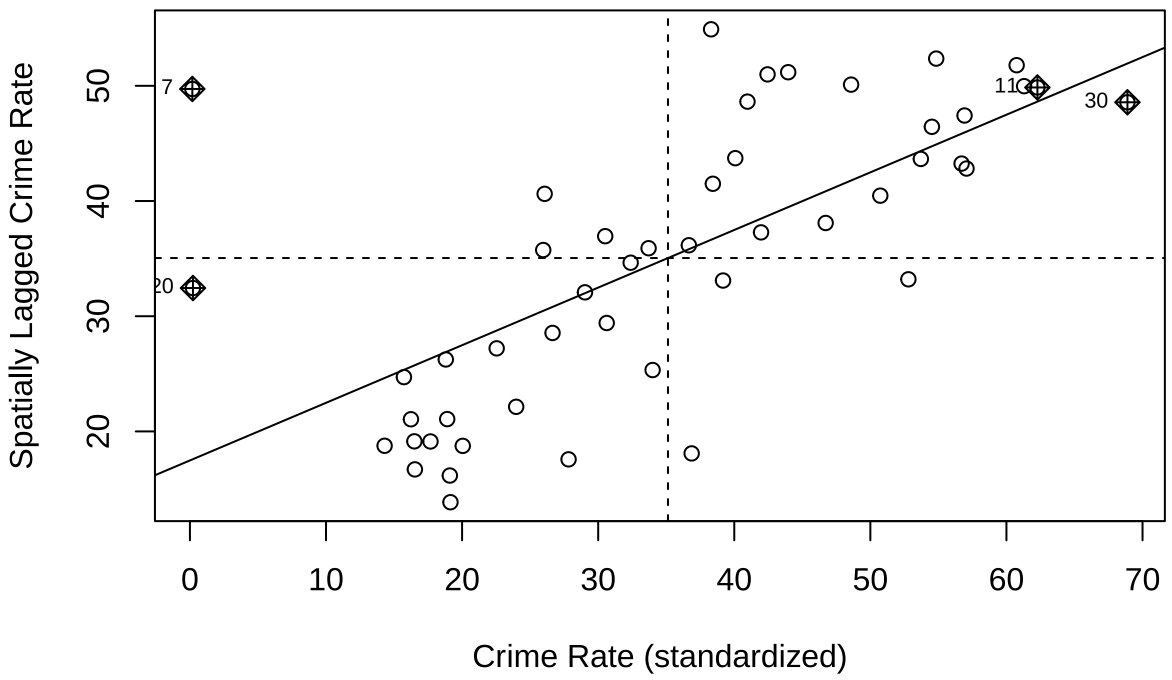
5. **p-value**: The probability of observing the calculated I value (or a more extreme one) under the null hypothesis

A significant positive I value (with a low p-value) indicates positive spatial autocorrelation - similar values tend to cluster together geographically. A significant negative I value indicates negative spatial autocorrelation - dissimilar values tend to cluster together.

Moran Scatterplot

A Moran scatterplot visualizes the relationship between a variable and its spatial lag (the weighted average of neighboring values). The x-axis represents the standardized variable values, and the y-axis represents the standardized spatial lag values.

```r
# Create Moran scatterplot for crime rates
moran.plot(columbus_sp$CRIME, queen_weights,
           labels = as.character(columbus_sp$POLYID),
           xlab = "Crime Rate (standardized)",
           ylab = "Spatially Lagged Crime Rate")
```



The Moran scatterplot has four quadrants: - **High-High (upper right)**: High values surrounded by high values - **Low-Low (lower left)**: Low values surrounded by low values - **High-Low (lower right)**: High values surrounded by low values - **Low-High (upper left)**: Low values surrounded by high values

The slope of the regression line through these points is equivalent to Moran's I.

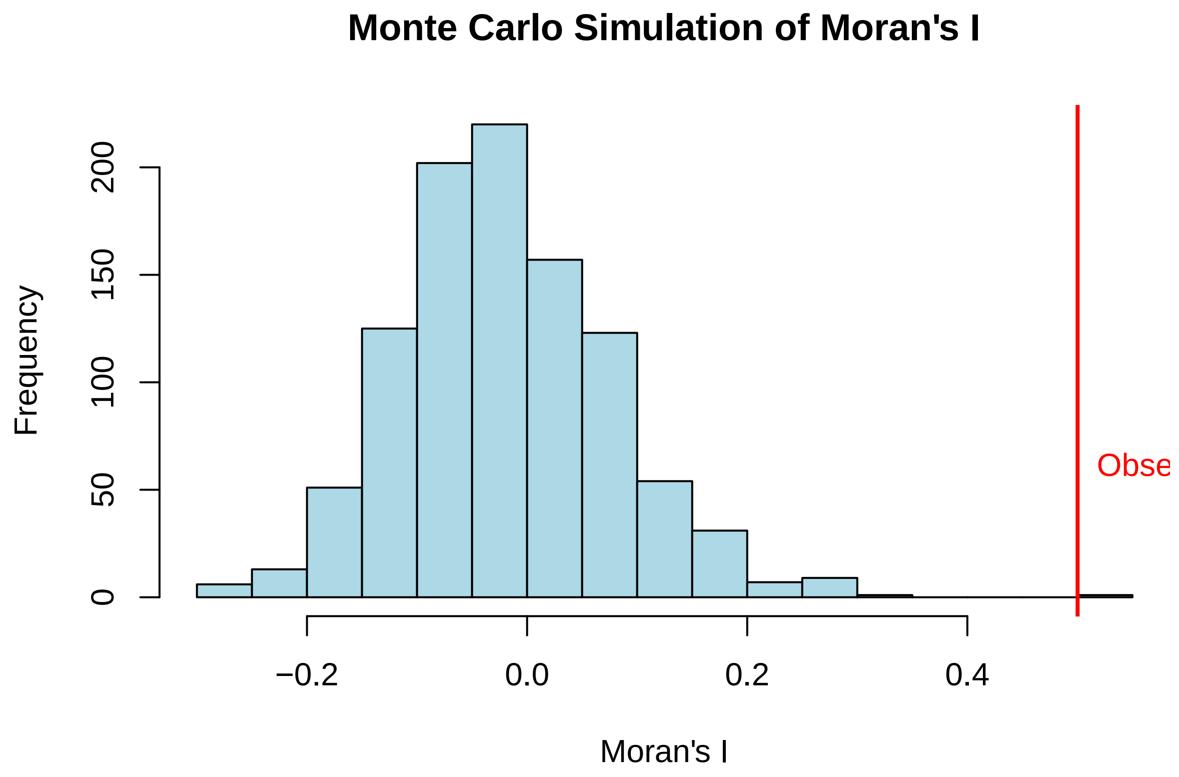
Monte Carlo Simulation for Moran's I

To assess the significance of Moran's I more robustly, we can use a Monte Carlo simulation approach:

```r
# Perform Monte Carlo simulation for Moran's I of crime rates
moran_mc_crime <- moran.mc(columbus_sp$CRIME, queen_weights, nsim = 999)
print(moran_mc_crime)
```

```
##
##  Monte-Carlo simulation of Moran I
##
## data:  columbus_sp$CRIME
## weights: queen_weights
## number of simulations + 1: 1000
##
## statistic = 0.50019, observed rank = 1000, p-value = 0.001
## alternative hypothesis: greater
```

```r
# Plot the simulation results
hist(moran_mc_crime$res, main = "Monte Carlo Simulation of Moran's I",
     xlab = "Moran's I", col = "lightblue")
abline(v = moran_mc_crime$statistic, col = "red", lwd = 2)
text(moran_mc_crime$statistic, 60,
     paste("Observed I =", round(moran_mc_crime$statistic, 3)),
     pos = 4, col = "red")
```



6. Local Spatial Autocorrelation

While global measures tell us about the overall pattern, local measures identify specific clusters or outliers at the local level.

Local Indicators of Spatial Association (LISA)

LISA statistics, introduced by Luc Anselin, decompose global spatial autocorrelation into contributions from individual locations. They help identify: - Local clusters (hotspots and coldspots) - Spatial outliers

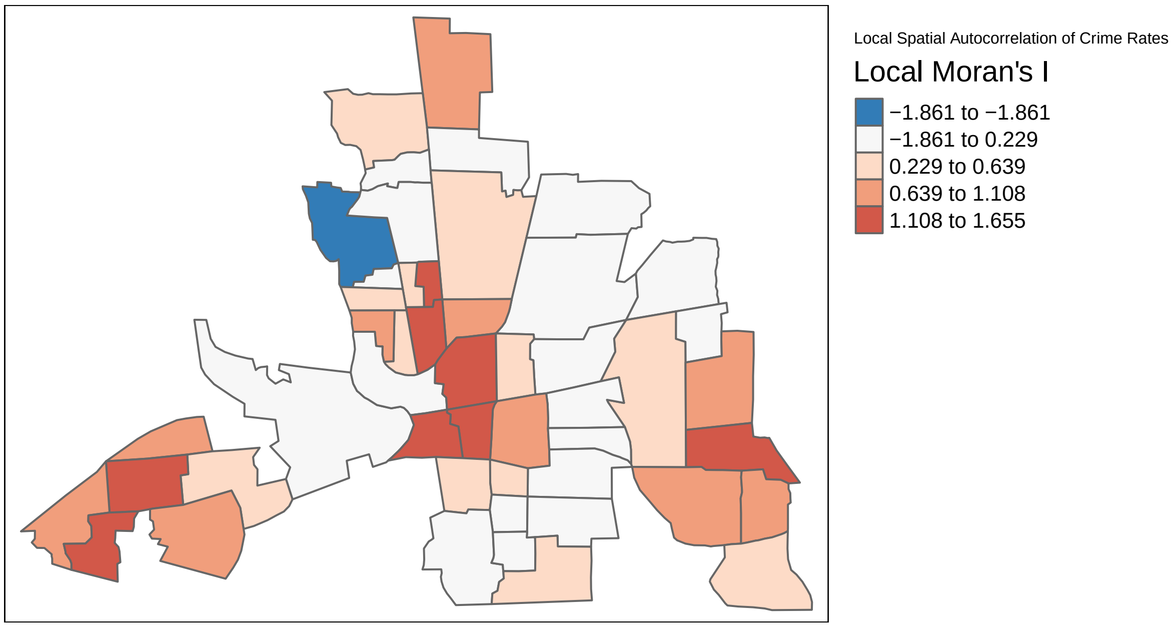
Let's calculate local Moran's I:

```r
# Calculate local Moran's I for crime rates
lisa_crime <- localmoran(columbus_sp$CRIME, queen_weights)
head(lisa_crime)
```

```
##             Ii          E.Ii      Var.Ii        Z.Ii Pr(z != E(Ii))
## 1  0.736818491 -0.0285985420 0.666144891  0.93780765      0.3483433
## 2  0.528777013 -0.0202502140 0.310266063  0.98565912      0.3243004
## 3  0.093850742 -0.0015396867 0.017630072  0.71841891      0.4724990
## 4  0.004820967 -0.0005707572 0.006541757  0.06666232      0.9468505
## 5  0.303606125 -0.0184931910 0.094617920  1.04713603      0.2950368
## 6 -0.181570517 -0.0062384562 0.148657003 -0.45474575      0.6492922
```

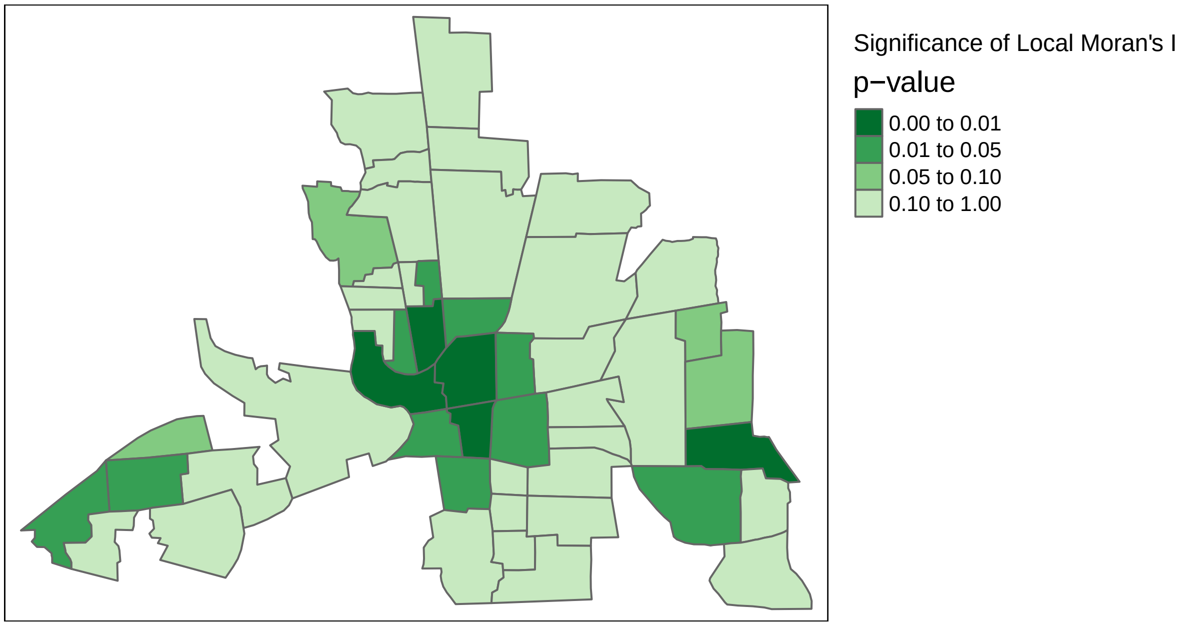
```r
# Add local Moran statistics to our spatial data
columbus_sp$local_moran <- lisa_crime[, 1]  # Local Moran's I statistic
columbus_sp$local_moran_p <- lisa_crime[, 5]  # P-value

# Map the local Moran's I values
tm_shape(columbus_sp) +
  tm_fill("local_moran", style = "jenks", palette = "-RdBu",
          midpoint = 0, title = "Local Moran's I") +
  tm_borders() +
  tm_layout(title = "Local Spatial Autocorrelation of Crime Rates",
            legend.outside = TRUE)
```

```
## Warning: Currect projection of shape columbus_sp unknown. Long-lat (WGS84) is
## assumed.
```



```r
# Map the significance of local Moran's I
tm_shape(columbus_sp) +
  tm_fill("local_moran_p", style = "fixed",
          breaks = c(0, 0.01, 0.05, 0.1, 1),
          palette = "-Greens", title = "p-value") +
  tm_borders() +
  tm_layout(title = "Significance of Local Moran's I",
            legend.outside = TRUE)
```

```
## Warning: Currect projection of shape columbus_sp unknown. Long-lat (WGS84) is
## assumed.
```

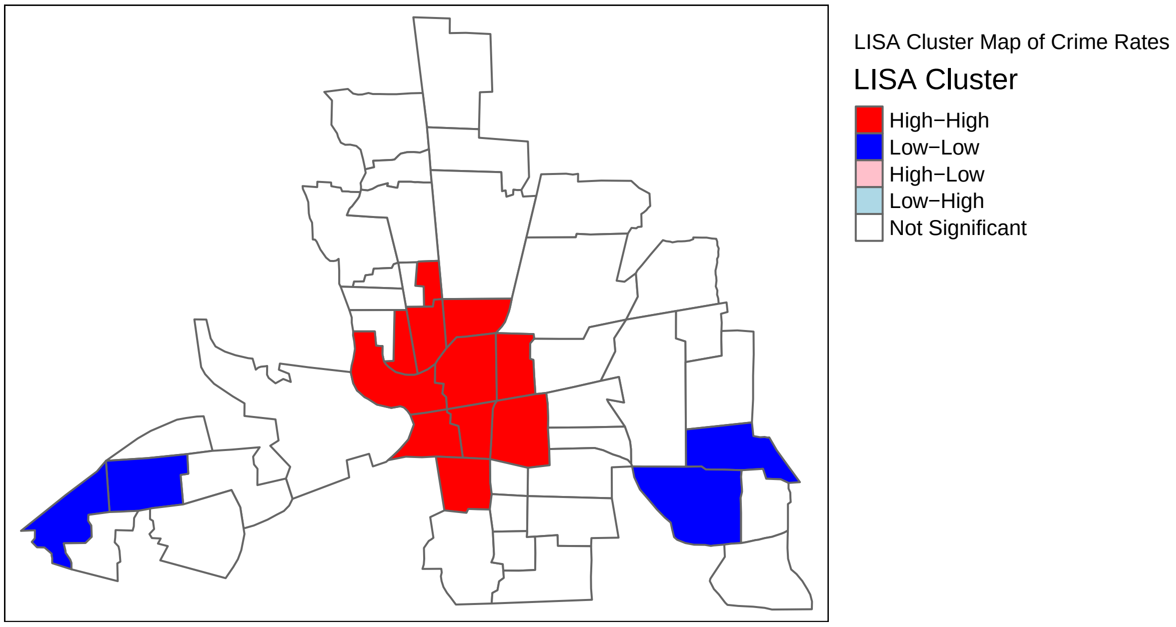


Creating a LISA Cluster Map

LISA cluster maps classify locations into significant clusters and outliers:

```r
# Create a function to classify LISA clusters
lisa_clusters <- function(var, lisa, p_value, p_threshold = 0.05) {
  # Standardize the variable
  z_var <- scale(var)

  # Calculate spatial lag of standardized variable
  lag_var <- lag.listw(queen_weights, z_var)

  # Create classifications
  clusters <- rep("Not Significant", length(var))
  clusters[lisa > 0 & p_value <= p_threshold & z_var > 0 & lag_var > 0] <- "High-High"
  clusters[lisa > 0 & p_value <= p_threshold & z_var < 0 & lag_var < 0] <- "Low-Low"
  clusters[lisa < 0 & p_value <= p_threshold & z_var > 0 & lag_var < 0] <- "High-Low"
  clusters[lisa < 0 & p_value <= p_threshold & z_var < 0 & lag_var > 0] <- "Low-High"

  return(factor(clusters, levels = c("High-High", "Low-Low", "High-Low", "Low-High", "Not Significant")
}

# Create LISA clusters for crime rates
columbus_sp$lisa_cluster <- lisa_clusters(columbus_sp$CRIME,
                                          columbus_sp$local_moran,
                                               columbus_sp$local_moran_p)

# Map the LISA clusters
tm_shape(columbus_sp) +
  tm_fill("lisa_cluster",
          palette = c("red", "blue", "pink", "lightblue", "white"),
          title = "LISA Cluster") +
  tm_borders() +
  tm_layout(title = "LISA Cluster Map of Crime Rates",
            legend.outside = TRUE)
```

```
## Warning: Currect projection of shape columbus_sp unknown. Long-lat (WGS84) is
## assumed.
```



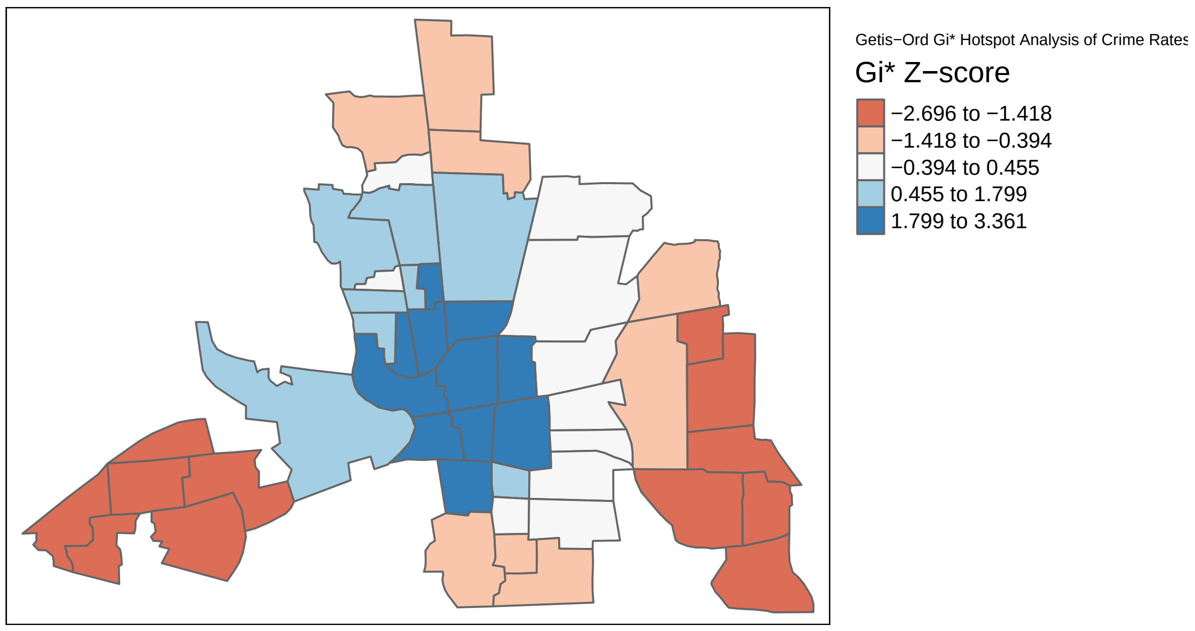
The LISA cluster map identifies: - **High-High clusters (red)**: High-crime areas surrounded by high-crime areas (hotspots) - **Low-Low clusters (blue)**: Low-crime areas surrounded by low-crime areas (coldspots) - **High-Low outliers (pink)**: High-crime areas surrounded by low-crime areas - **Low-High outliers (light blue)**: Low-crime areas surrounded by high-crime areas - **Not significant (white)**: Areas where the pattern is not statistically significant

Getis-Ord Gi* Hotspot Analysis

Another approach to local spatial autocorrelation is the Getis-Ord Gi* statistic (pronounced "G-i-star"), which identifies statistically significant hot spots (clusters of high values) and cold spots (clusters of low values).

```r
# Calculate Getis-Ord Gi* statistic for crime rates
# First, we need a binary spatial weights matrix
binary_weights <- nb2listw(queen_nb, style = "B", zero.policy = TRUE)

# Calculate the Gi* statistic
gi_crime <- localG(columbus_sp$CRIME, binary_weights)
columbus_sp$gi_star <- gi_crime

# Map the Gi* statistic
tm_shape(columbus_sp) +
  tm_fill("gi_star", style = "jenks", palette = "RdBu",
          midpoint = 0, title = "Gi* Z-score") +
  tm_borders() +
  tm_layout(title = "Getis-Ord Gi* Hotspot Analysis of Crime Rates",
            legend.outside = TRUE)
```

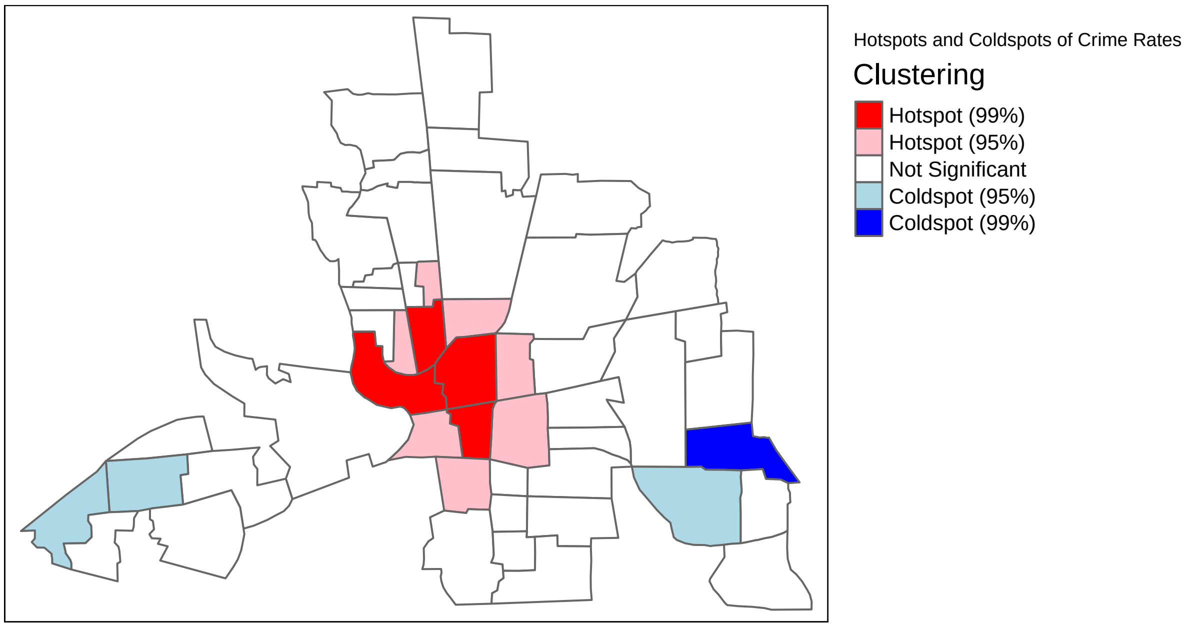
```
## Warning: Currect projection of shape columbus_sp unknown. Long-lat (WGS84) is
## assumed.
```



```r
# Classify into hotspots and coldspots based on significance
columbus_sp$hotspot <- "Not Significant"
columbus_sp$hotspot[columbus_sp$gi_star > 1.96] <- "Hotspot (95%)"
columbus_sp$hotspot[columbus_sp$gi_star > 2.58] <- "Hotspot (99%)"
columbus_sp$hotspot[columbus_sp$gi_star < -1.96] <- "Coldspot (95%)"
columbus_sp$hotspot[columbus_sp$gi_star < -2.58] <- "Coldspot (99%)"
```

```r
columbus_sp$hotspot <- factor(columbus_sp$hotspot,
                         levels = c("Hotspot (99%)", "Hotspot (95%)",
                                    "Not Significant",
                                    "Coldspot (95%)", "Coldspot (99%)"))

# Map the hotspots and coldspots
tm_shape(columbus_sp) +
  tm_fill("hotspot", palette = c("red", "pink", "white", "lightblue", "blue"),
          title = "Clustering") +
  tm_borders() +
  tm_layout(title = "Hotspots and Coldspots of Crime Rates",
            legend.outside = TRUE)
```

```
## Warning: Currect projection of shape columbus_sp unknown. Long-lat (WGS84) is
## assumed.
```



7. Comparing Different Spatial Weights

The choice of spatial weights can affect the results of spatial autocorrelation analysis. Let's compare the Moran's I values obtained using different weight structures:

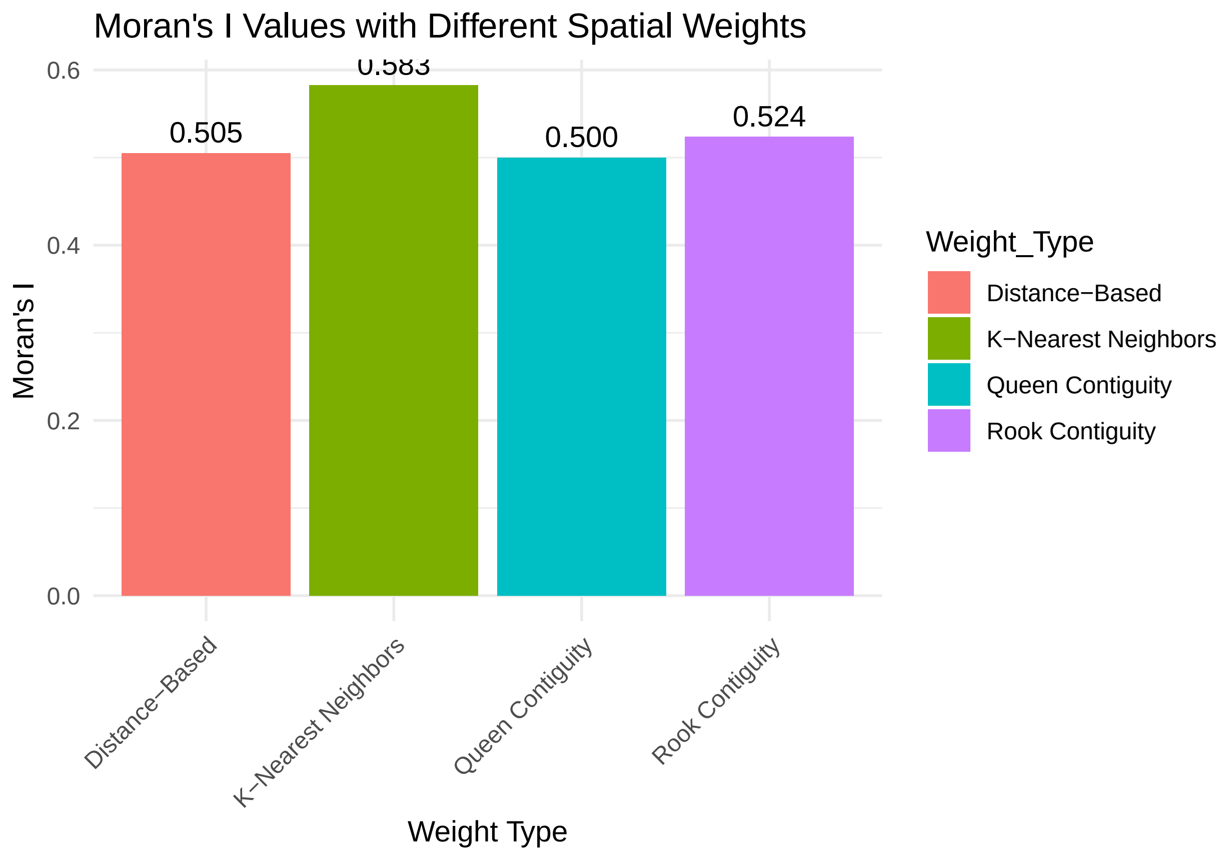
```r
# Calculate Moran's I for crime rates using different weights
moran_queen <- moran.test(columbus_sp$CRIME, queen_weights)
```

```r
moran_rook <- moran.test(columbus_sp$CRIME, rook_weights)
moran_dist <- moran.test(columbus_sp$CRIME, dist_weights)
moran_knn <- moran.test(columbus_sp$CRIME, knn_weights)

# Create a comparison table
weights_comparison <- data.frame(
  Weight_Type = c("Queen Contiguity", "Rook Contiguity", "Distance-Based", "K-Nearest Neighbors"),
  Morans_I = c(moran_queen$estimate[1], moran_rook$estimate[1],
              moran_dist$estimate[1], moran_knn$estimate[1]),
  p_value = c(moran_queen$p.value, moran_rook$p.value,
             moran_dist$p.value, moran_knn$p.value)
)

# Display the comparison table
kable(weights_comparison, digits = 5,
      caption = "Comparison of Moran's I with Different Spatial Weights")
```

Table 1: Comparison of Moran's I with Different Spatial Weights

Weight_Type	Morans_I	p_value
Queen Contiguity	0.50019	0
Rook Contiguity	0.52367	0
Distance-Based	0.50531	0
K-Nearest Neighbors	0.58255	0

```r
# Create a bar plot of Moran's I values
ggplot(weights_comparison, aes(x = Weight_Type, y = Morans_I, fill = Weight_Type)) +
  geom_bar(stat = "identity") +
  theme_minimal() +
  labs(title = "Moran's I Values with Different Spatial Weights",
       x = "Weight Type",
       y = "Moran's I") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  geom_text(aes(label = sprintf("%.3f", Morans_I)), vjust = -0.5)
```



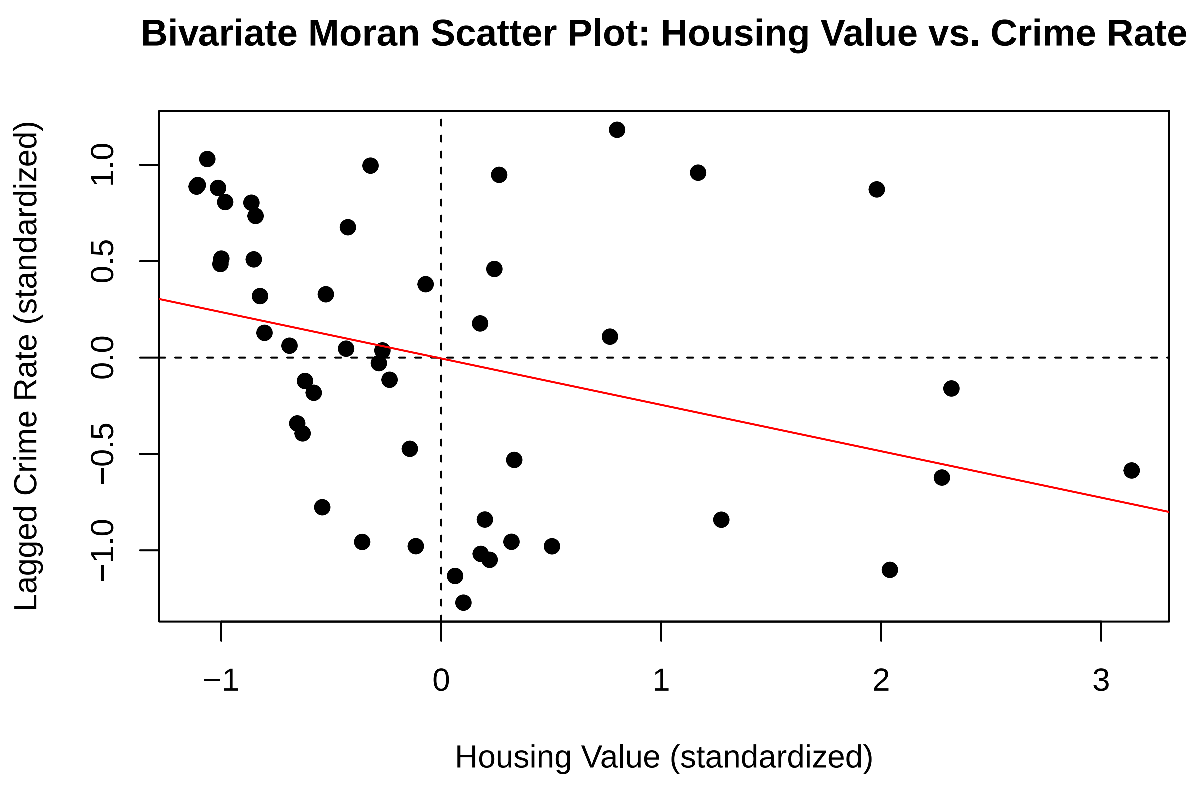
8. Multivariate Spatial Autocorrelation

So far, we've focused on univariate spatial autocorrelation - the spatial pattern of a single variable. We can also examine the spatial relationship between different variables using bivariate Moran's I.

```r
# Calculate bivariate Moran's I between crime rate and housing value
# Standardize both variables
crime_z <- scale(columbus_sp$CRIME)
hoval_z <- scale(columbus_sp$HOVAL)

# Calculate spatial lag of housing value
lag_hoval_z <- lag.listw(queen_weights, hoval_z)

# Calculate bivariate Moran's I
biv_moran_I <- sum(crime_z * lag_hoval_z) / length(crime_z)
print(paste("Bivariate Moran's I between Crime and Housing Value:", round(biv_moran_I, 4)))
```

```
## [1] "Bivariate Moran's I between Crime and Housing Value: -0.1681"
```

```r
# Create a bivariate Moran scatter plot
plot(hoval_z, lag.listw(queen_weights, crime_z),
     xlab = "Housing Value (standardized)",
     ylab = "Lagged Crime Rate (standardized)",
     main = "Bivariate Moran Scatter Plot: Housing Value vs. Crime Rate",
     pch = 19)
abline(h = 0, v = 0, lty = 2)
abline(lm(lag.listw(queen_weights, crime_z) ~ hoval_z), col = "red")
```



A negative bivariate Moran's I indicates an inverse spatial relationship: areas with high housing values tend to be surrounded by areas with low crime rates, and vice versa.

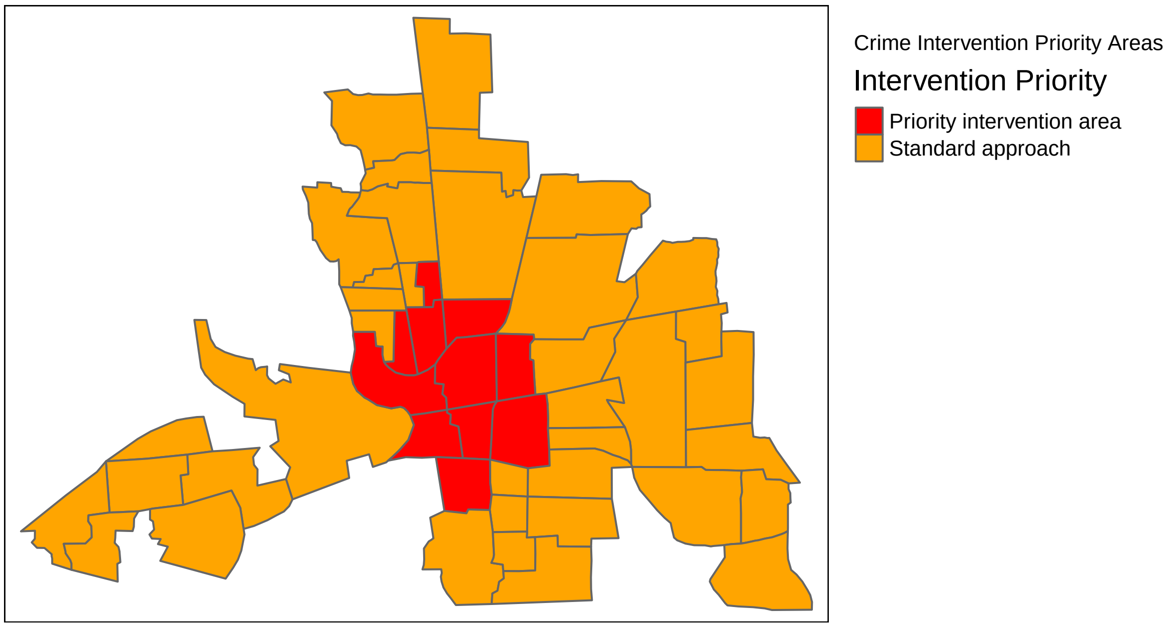
9. Practical Interpretation and Application

Let's interpret these results in a practical context:

```
# Combine results from different methods for a comprehensive view
columbus_sp$insight <- "Standard approach"

# Areas with significant crime clustering (from LISA)
high_crime_clusters <- (columbus_sp$lisa_cluster == "High-High")

# Areas that are hotspots from Getis-Ord Gi*
hotspots <- (columbus_sp$hotspot == "Hotspot (95%)" | columbus_sp$hotspot == "Hotspot (99%)")

# Combine insights
columbus_sp$insight[high_crime_clusters & hotspots] <- "Priority intervention area"
columbus_sp$insight[high_crime_clusters & !hotspots] <- "Emerging cluster"
```

```r
columbus_sp$insight[!high_crime_clusters & hotspots] <- "Potential concern"

# Map the combined insights
tm_shape(columbus_sp) +
  tm_fill("insight", palette = c("red", "orange", "yellow", "grey"),
          title = "Intervention Priority") +
  tm_borders() +
  tm_layout(title = "Crime Intervention Priority Areas",
            legend.outside = TRUE)
```

```
## Warning: Currect projection of shape columbus_sp unknown. Long-lat (WGS84) is
## assumed.
```


Crime Intervention Priority Areas
Intervention Priority
🟥 Priority intervention area
🟧 Standard approach

This integrated approach can help policymakers:

1. Identify areas with significant clustering of high crime rates
2. Prioritize interventions in areas that show consistent patterns across different methods
3. Target resources more effectively based on spatial patterns
4. Design context-specific interventions based on local conditions

10. Advanced Topics and Considerations

Modifiable Areal Unit Problem (MAUP)

The MAUP refers to how results of spatial analyses can be affected by the specific boundaries used to aggregate data. Different boundary definitions can lead to different patterns and conclusions.

Edge Effects

Edge effects occur because areas at the boundary of the study region have fewer neighbors, which can affect the calculation of spatial statistics.

Statistical Significance vs. Practical Significance

Statistical significance doesn't always translate to practical importance. Consider the magnitude of the effects alongside p-values.

Spatio-temporal Autocorrelation

Real-world processes often have both spatial and temporal dimensions. Methods exist to analyze spatio-temporal autocorrelation.

11. Conclusion

In this tutorial, we've explored:

1. **Global spatial autocorrelation** using Moran's I to detect overall clustering patterns
2. **Local spatial autocorrelation** using LISA and Getis-Ord Gi* to identify specific clusters and hotspots
3. **Different spatial weight structures** and their effects on the analysis
4. **Bivariate spatial relationships** between different variables
5. **Practical application** of these methods for policy interventions

Spatial autocorrelation analysis provides a powerful framework for understanding spatial patterns, identifying clusters, and informing targeted interventions. By applying these techniques, researchers and policymakers can gain deeper insights into spatial processes and make more informed decisions.