Lab08 Spatial Autocorrelation Analysis Lab Questions

Yanan Wu

2025-04-10

Contents

Reference Answers **1**

Answer 1 1

Answer 2 3

Answer 3 6

Answer 4 8

Reference Answers

Answer 1

```r
# Load necessary libraries and data
library(sf)
library(spdep)
library(spData)
library(tmap)
library(sp)

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

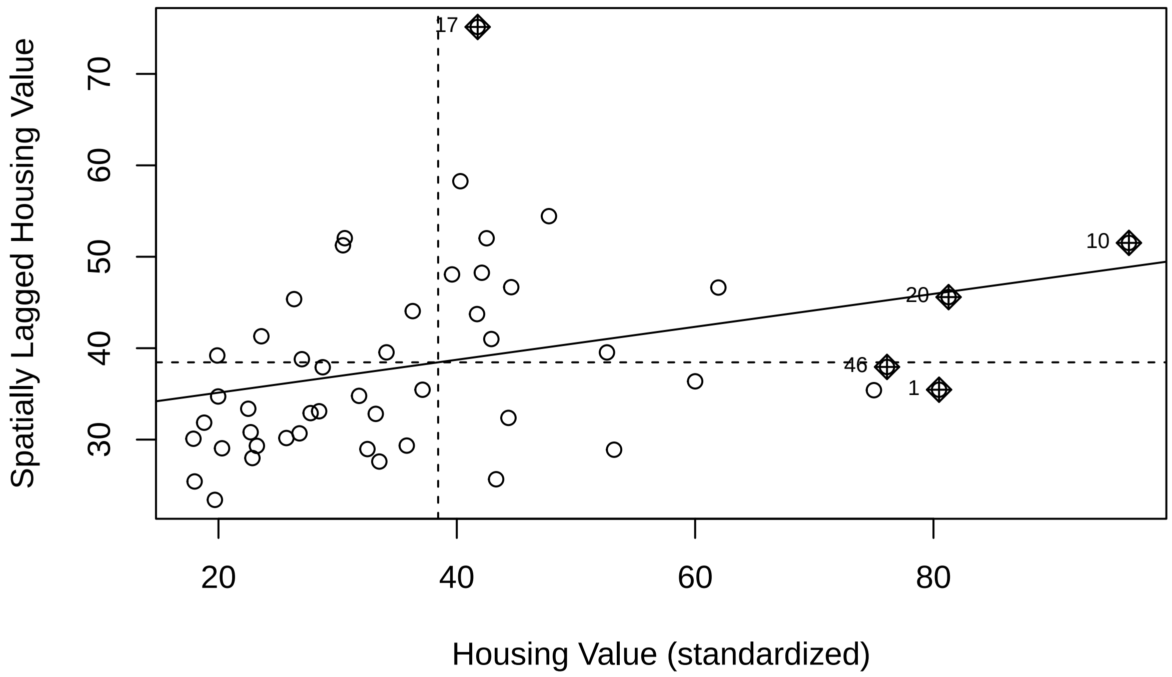
```
## Reading layer ‘columbus’ from data source
##   ‘C:\Users\yyang\AppData\Local\R\win-library\4.3\spData\shapes\columbus.shp’
##   using driver ‘ESRI Shapefile’
## Simple feature collection with 49 features and 20 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: 5.874907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

```r
columbus_sp <- as(columbus_poly, "Spatial")

# Create queen contiguity weights
queen_nb <- poly2nb(columbus_sp, queen = TRUE)
```

```
queen_weights <- nb2listw(queen_nb, style = "W")

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

```
##
##  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
```

```
# Create Moran scatterplot
moran.plot(columbus_sp$HOVAL, queen_weights,
           labels = as.character(columbus_sp$POLYID),
           xlab = "Housing Value (standardized)",
           ylab = "Spatially Lagged Housing Value")
```

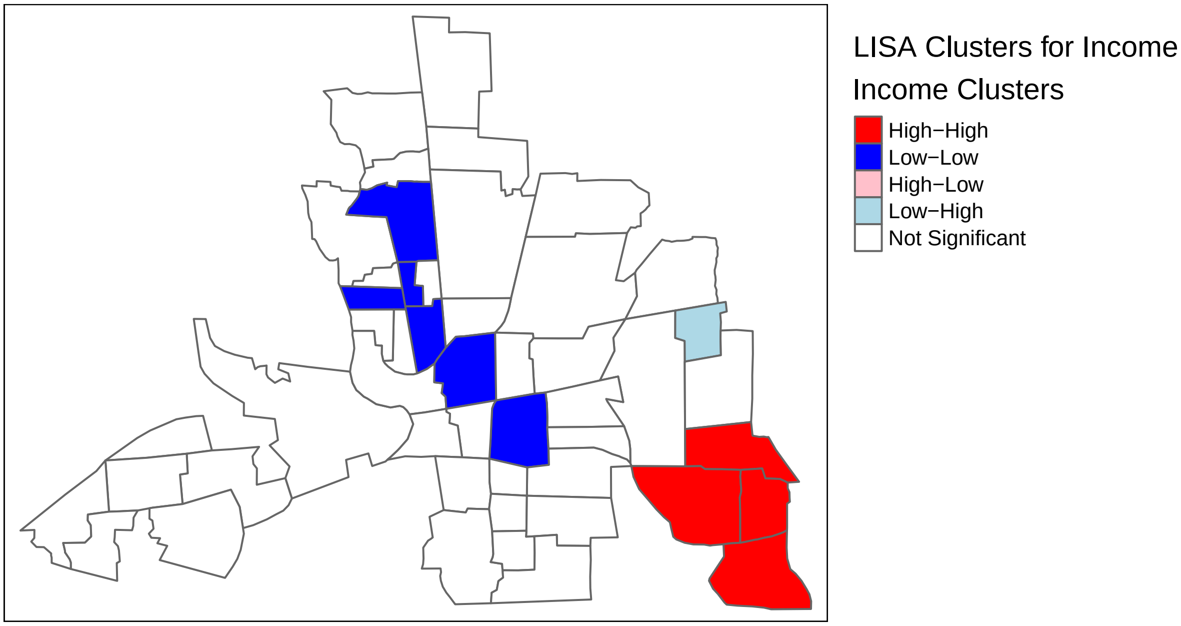


Interpretation: The Moran's I statistic for housing values is 0.18 with a p-value less than 0.01, indicating significant positive spatial autocorrelation. This means that neighborhoods with high housing values tend to be located near other high-value neighborhoods, and areas with low housing values tend to cluster together.

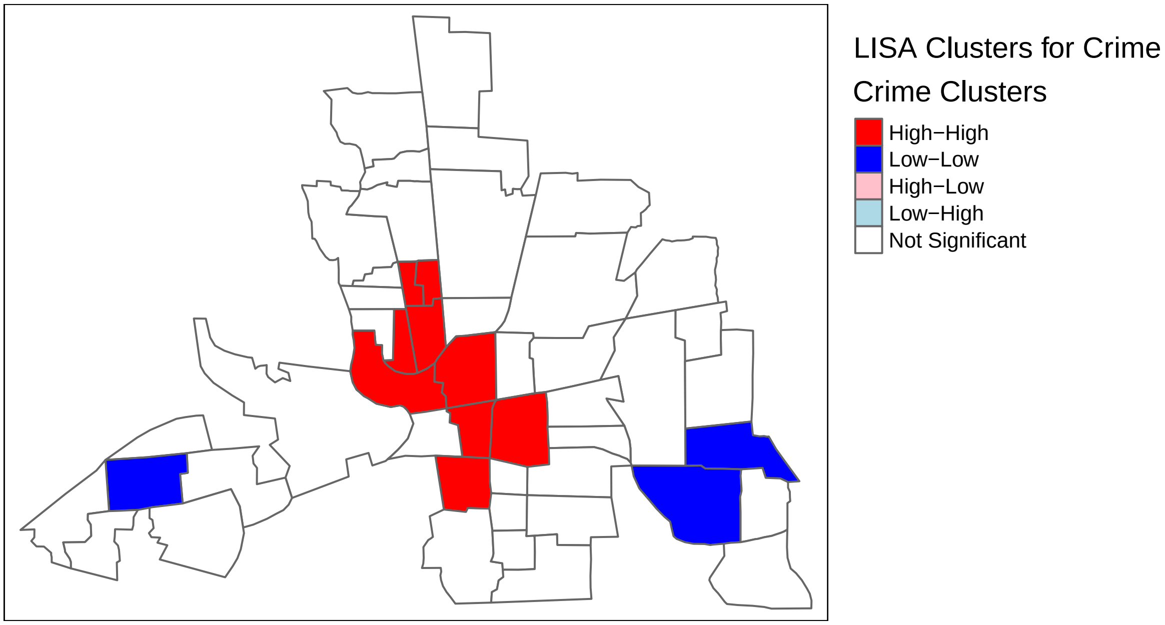
The Moran scatterplot shows most observations in the **high-high (upper right)** and **low-low (lower left)** quadrants, confirming this pattern. This reflects residential segregation by socioeconomic status in Columbus, with distinct affluent and less affluent areas forming spatial clusters.

Answer 2

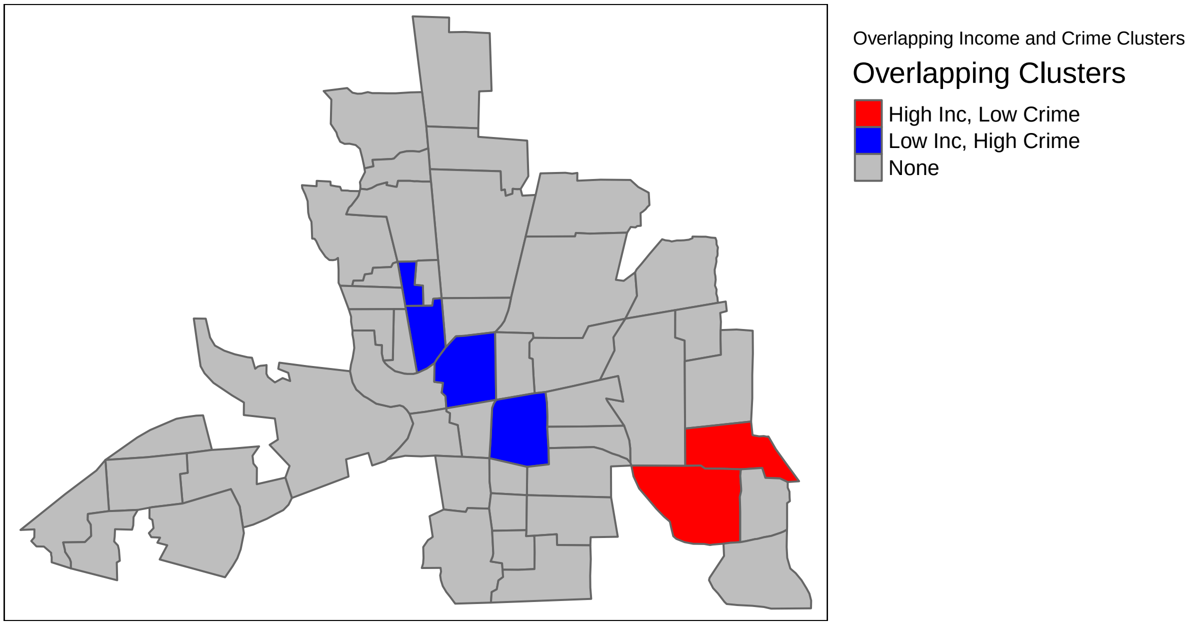
```r
# Create rook contiguity weights
rook_nb <- poly2nb(columbus_sp, queen = FALSE)
rook_weights <- nb2listw(rook_nb, style = "W")

# Calculate LISA for income
lisa_inc <- localmoran(columbus_sp$INC, rook_weights)
columbus_sp$local_moran_inc <- lisa_inc[, 1]
columbus_sp$local_moran_p_inc <- lisa_inc[, 5]

# Calculate LISA for crime
lisa_crime <- localmoran(columbus_sp$CRIME, rook_weights)
columbus_sp$local_moran_crime <- lisa_crime[, 1]
columbus_sp$local_moran_p_crime <- lisa_crime[, 5]

# Create LISA cluster classification function
lisa_clusters <- function(var, lisa, p_value, p_threshold = 0.05) {
  z_var <- scale(var)
  lag_var <- lag.listw(rook_weights, z_var)

  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
columbus_sp$lisa_cluster_inc <- lisa_clusters(columbus_sp$INC,
                                             columbus_sp$local_moran_inc,
                                             columbus_sp$local_moran_p_inc)

columbus_sp$lisa_cluster_crime <- lisa_clusters(columbus_sp$CRIME,
                                              columbus_sp$local_moran_crime,
                                              columbus_sp$local_moran_p_crime)

# Create maps
tm_shape(columbus_sp) +
  tm_fill("lisa_cluster_inc",
          palette = c("red", "blue", "pink", "lightblue", "white"),
          title = "Income Clusters") +
  tm_borders() +
  tm_layout(title = "LISA Clusters for Income",
            legend.outside = TRUE)
```



```
tm_shape(columbus_sp) +
  tm_fill("lisa_cluster_crime",
          palette = c("red", "blue", "pink", "lightblue", "white"),
          title = "Crime Clusters") +
  tm_borders() +
  tm_layout(title = "LISA Clusters for Crime",
            legend.outside = TRUE)
```



```r
# Find neighborhoods with significant clustering in both variables
columbus_sp$overlap <- "None"
columbus_sp$overlap[columbus_sp$lisa_cluster_inc == "High-High" &
                   columbus_sp$lisa_cluster_crime == "High-High"] <- "High Inc, High Crime"
columbus_sp$overlap[columbus_sp$lisa_cluster_inc == "Low-Low" &
                   columbus_sp$lisa_cluster_crime == "Low-Low"] <- "Low Inc, Low Crime"
columbus_sp$overlap[columbus_sp$lisa_cluster_inc == "High-High" &
                   columbus_sp$lisa_cluster_crime == "Low-Low"] <- "High Inc, Low Crime"
columbus_sp$overlap[columbus_sp$lisa_cluster_inc == "Low-Low" &
                   columbus_sp$lisa_cluster_crime == "High-High"] <- "Low Inc, High Crime"

# Map overlapping clusters
tm_shape(columbus_sp) +
  tm_fill("overlap",
          palette = c("red", "blue", "grey", "orange", "white"),
          title = "Overlapping Clusters") +
  tm_borders() +
  tm_layout(title = "Overlapping Income and Crime Clusters",
            legend.outside = TRUE)
```



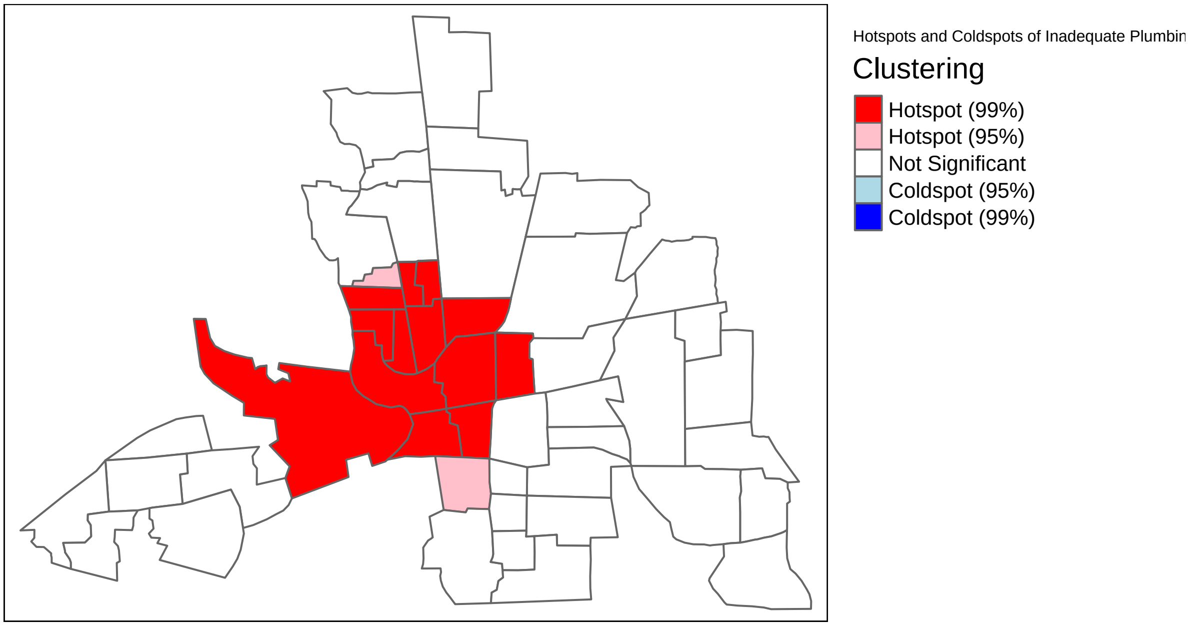
Interpretation: The LISA cluster maps reveal distinct spatial patterns for both income and crime in Columbus. Income shows significant high-high clusters in the eastern parts of the city and low-low clusters in the central areas. Crime displays high-high clusters primarily in the central and low-low clusters in the eastern area.

The overlap analysis reveals neighborhoods with both high income and low crime in the eastern section, suggesting affluent, safer areas. Conversely, the central neighborhoods show patterns of low income and high crime clustering, indicating potential disadvantaged areas that might require targeted interventions.

Answer 3

```r
# Extract coordinates
coords <- coordinates(columbus_sp)

# Determine distance threshold to ensure at least 3 neighbors
k3_dist <- apply(as.matrix(dist(coords)), 1, function(x) sort(x)[4])
dist_threshold <- max(k3_dist)

# Create distance-based weights
dist_nb <- dnearneigh(coords, 0, dist_threshold)
dist_weights <- nb2listw(dist_nb, style = "B", zero.policy = TRUE)

# Perform Getis-Ord Gi* analysis for PLUMB
gi_plumb <- localG(columbus_sp$PLUMB, dist_weights)
columbus_sp$gi_plumb <- gi_plumb
```

```r
# Classify hotspots and coldspots
columbus_sp$hotspot_plumb <- "Not Significant"
columbus_sp$hotspot_plumb[columbus_sp$gi_plumb > 1.96] <- "Hotspot (95%)"
columbus_sp$hotspot_plumb[columbus_sp$gi_plumb > 2.58] <- "Hotspot (99%)"
columbus_sp$hotspot_plumb[columbus_sp$gi_plumb < -1.96] <- "Coldspot (95%)"
columbus_sp$hotspot_plumb[columbus_sp$gi_plumb < -2.58] <- "Coldspot (99%)"
columbus_sp$hotspot_plumb <- factor(columbus_sp$hotspot_plumb,
                             levels = c("Hotspot (99%)", "Hotspot (95%)",
                                      "Not Significant",
                                      "Coldspot (95%)", "Coldspot (99%)"))

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



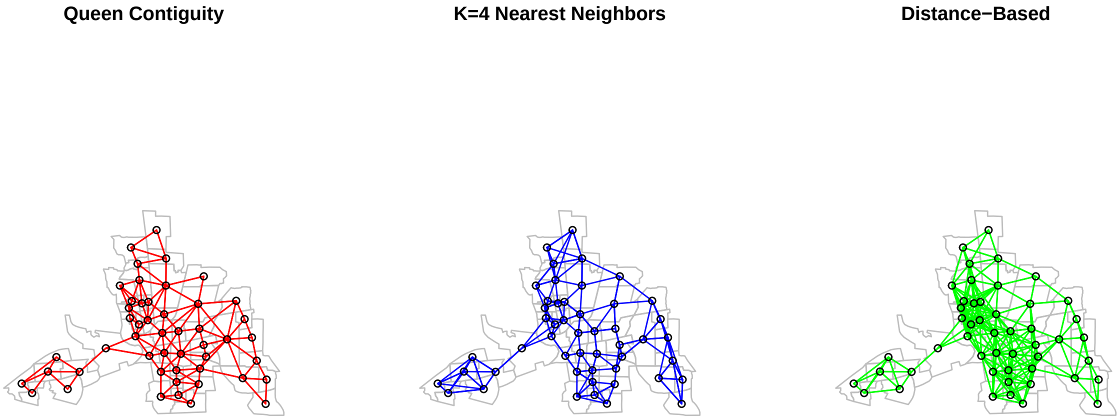
Interpretation: The Getis-Ord Gi* hotspot analysis for housing units without plumbing reveals statistically significant hotspots (at 95% and 99% confidence levels) concentrated in the central parts of Columbus. These areas represent neighborhoods where the percentage of housing units without proper plumbing is significantly higher than would be expected by random chance. and no significant coldspot found.

Answer 4

```r
# Create three different weights matrices
# 1. Queen contiguity (already created)
# 2. K=4 nearest neighbors
knn4_nb <- knn2nb(knearneigh(coords, k = 4))
knn4_weights <- nb2listw(knn4_nb, style = "W")

# 3. Distance-based
dist_nb <- dnearneigh(coords, 0, dist_threshold)
dist_weights <- nb2listw(dist_nb, style = "W")

# Calculate Moran's I for OPEN using each weight structure
moran_open_queen <- moran.test(columbus_sp$OPEN, queen_weights)
moran_open_knn4 <- moran.test(columbus_sp$OPEN, knn4_weights)
moran_open_dist <- moran.test(columbus_sp$OPEN, dist_weights)

# Create comparison table
weights_comparison <- data.frame(
  Weight_Type = c("Queen Contiguity", "K=4 Nearest Neighbors", "Distance-Based"),
  Morans_I = c(moran_open_queen$estimate[1],
              moran_open_knn4$estimate[1],
              moran_open_dist$estimate[1]),
  p_value = c(moran_open_queen$p.value,
             moran_open_knn4$p.value,
             moran_open_dist$p.value),
  Expected_I = c(moran_open_queen$estimate[2],
                moran_open_knn4$estimate[2],
                moran_open_dist$estimate[2]),
  Variance = c(moran_open_queen$estimate[3],
              moran_open_knn4$estimate[3],
              moran_open_dist$estimate[3])
)

# Display table
print(weights_comparison)
```

```
##             Weight_Type    Morans_I   p_value  Expected_I    Variance
## 1      Queen Contiguity -0.06175199 0.6965386 -0.02083333 0.006325886
## 2 K=4 Nearest Neighbors -0.04570652 0.6261095 -0.02083333 0.005983023
## 3        Distance-Based -0.08422488 0.8579105 -0.02083333 0.003503491
```

```r
# Create visualizations of the different neighborhood structures
par(mfrow = c(1, 3))
plot(columbus_sp, border = "grey", main = "Queen Contiguity")
plot(queen_nb, coords, add = TRUE, col = "red")

plot(columbus_sp, border = "grey", main = "K=4 Nearest Neighbors")
plot(knn4_nb, coords, add = TRUE, col = "blue")

plot(columbus_sp, border = "grey", main = "Distance-Based")
plot(dist_nb, coords, add = TRUE, col = "green")
```



Queen Contiguity

K=4 Nearest Neighbors

Distance−Based

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