Geospatial Analysis in R

Kim Kreiss

Introduction

Brief re-intro to spatial data and mapping

Statistical Techniques for Point Pattern Analysis

Wrap-up

Introduction

Today

- We will analyze crime data in Chicago together:
  - Combine point data and boundary file
  - Generate informative visualizations
  - Apply statistical techniques to understand the spatial distribution of homicides in Chicago
- This is an interactive workshop, where you are provided with code and data to run locally:
  - `01-geospatial-workshop.Rmd`
  - `2016-2020-chicago-crime.csv`
  - `Boundaries - Neighborhoods.geojson`
  - `Boundaries - Census Tracts - 2010.geojson`
- Rely heavily on two great resources:
  - Spatial Statistics for Data Science: Theory and Practice with R https://www.paulamoraga.com/book-spatial/index.html
  - https://michaelminn.net/tutorials/r-crime/

Brief re-intro to spatial data and mapping

Why R?

- Reproducibility and Transparency
- Flexibility and Customization
- Integration with Statistical Analysis
- Geospatial Visualization

Spatial Data

- Most commonly point data, boundary data, grid or raster data
  - point data: longitude and latitude of a unit of interest
  - boundary data: polygon/shape data delineating a geographical area
  - grid or raster data: usually squares/rectangles over the Earth's surface with specific attributes
- Uses Coordinate Reference System (CRS)

Mapping

- Common application is to show point data within a region or geographic boundaries of interest, ie, are homicides in Chicago concentrated in certain neighborhoods?
- Let's map crime in Chicago neighborhoods and census tracts

Required packages

- `sf` for handling spatial data and mapping
- `spdep` and `spatstat` for spatial statistical functions
- `tidyverse` for data manipulation, exploration and visualization
- `colorspace` for more custom color options for mapping

```
# Load required libraries
library(sf)
library(spdep)
library(spatstat)
library(tidyverse)
library(colorspace)
```

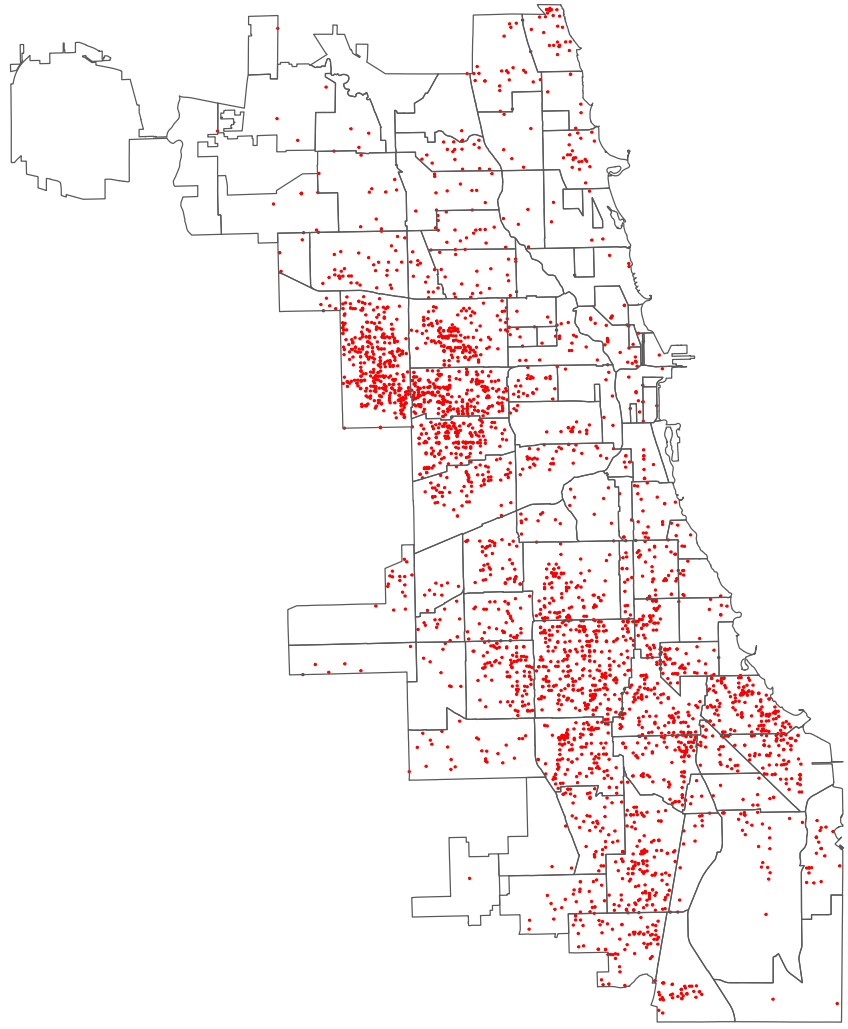
Read in the data

```
crime_data <- read.csv("../data/2016-2020-chicago-crime.csv")

neighborhoods <- st_read("../data/Boundaries - Neighborhoods.geojson",
                         quiet=T)

census_tracts <- st_read("../data/Boundaries - Census Tracts - 2010.geojson",
                         quiet=T)

homicide <- crime_data %>%
  filter(Primary.Type=="HOMICIDE")
```

Code: Homicides in Chicago Neighborhoods

```
neighborhoods_map <- neighborhoods %>%
  ggplot() +
  geom_point(data = homicide, aes(x=Longitude, y=Latitude),
             shape=1, color="red",size=.005) +
  geom_sf(aes(geometry=geometry), fill="transparent", linewidth=.25) +
  theme(axis.text.x = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks = element_blank(),
        rect = element_blank(),
        legend.position = "bottom") +
  labs(x="", y="")
```

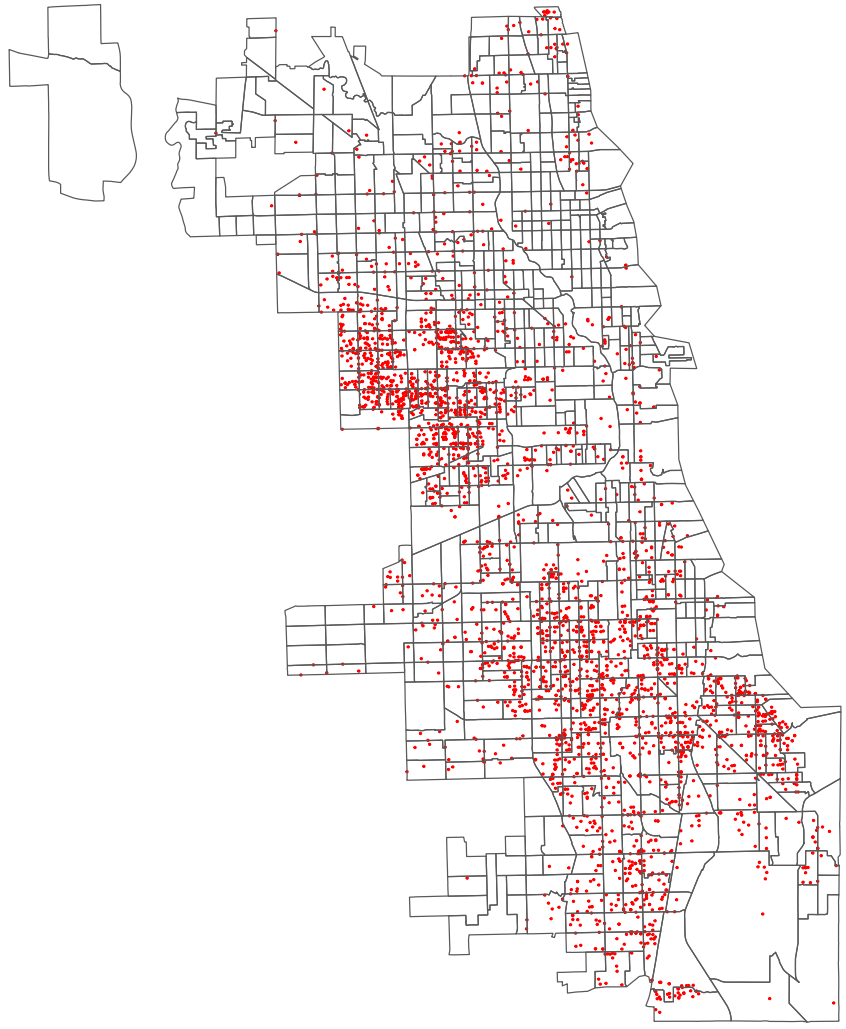
Output: Homicides in Chicago (2016-2019) by Neighborhood



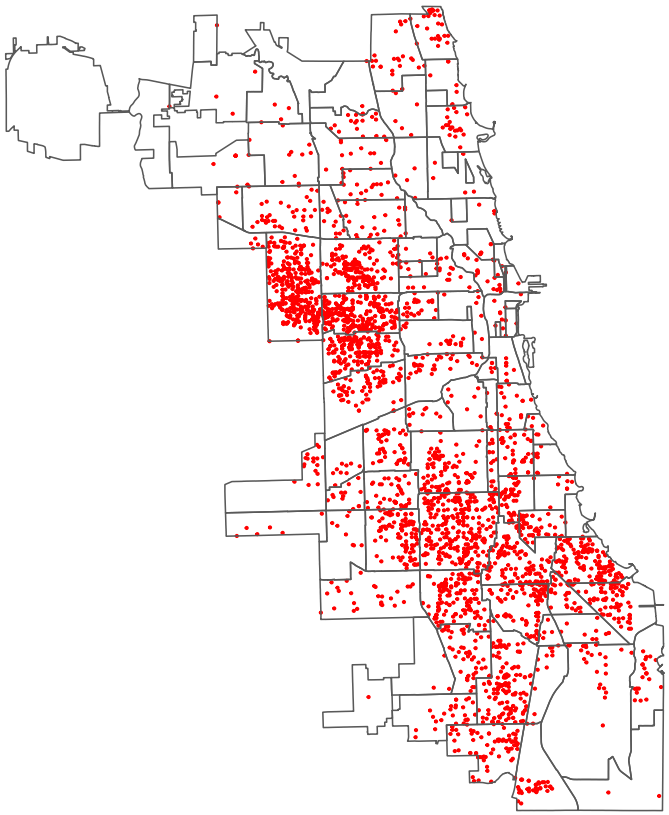
Code: Homicides in Chicago Census Tracts

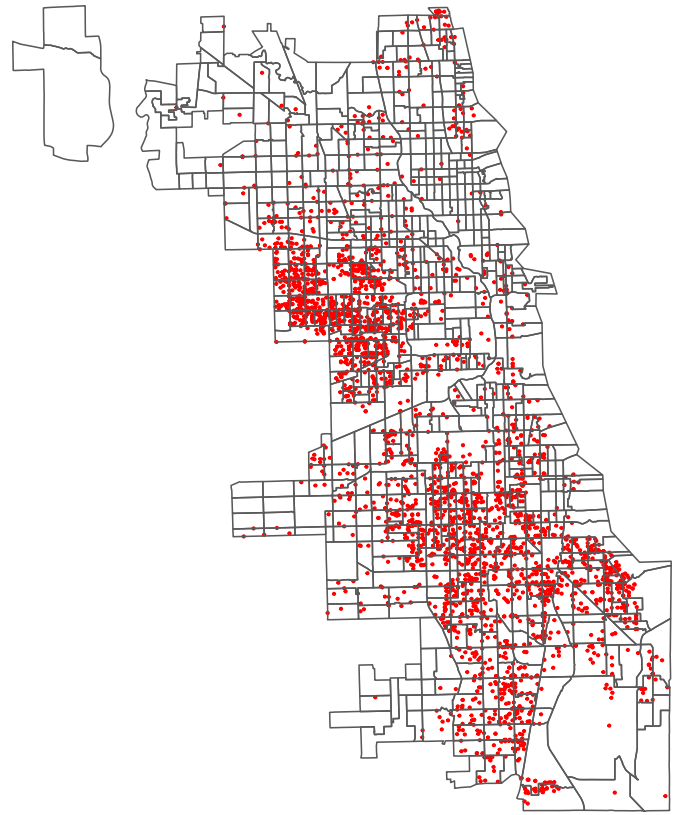
```
tracts_map <- census_tracts %>%
  ggplot() +
  geom_point(data = homicide, aes(x=Longitude, y=Latitude), shape=1, color="red
  geom_sf(aes(geometry=geometry), fill="transparent", linewidth=.25) +
  theme(axis.text.x = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks = element_blank(),
        rect = element_blank(),
        legend.position = "bottom") +
  labs(x="", y="") #, title = "Homicides in Chicago (2016-2019)",
      # subtitle = "By Neighborhood")
```

Output: Homicides in Chicago (2016-2019) by Census Tract



Comparison





Let's make neighborhood and census tract heat maps now

```
point_sf <- st_as_sf(homicide, coords = c("Longitude", "Latitude"), crs = 4326)

#crs=4326 WGS 1984 latitudes and longitudes (EPSG 4326),
#this is the most popular projection

neighborhoods$homicides <- lengths(st_intersects(neighborhoods, point_sf))
census_tracts$homicides <- lengths(st_intersects(census_tracts, point_sf))
```

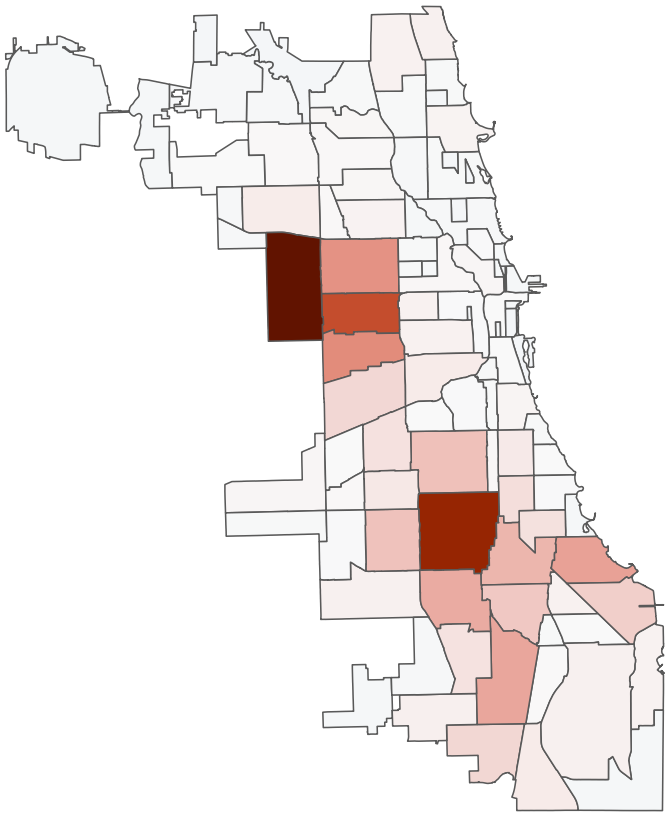
Neighborhood Heat Map

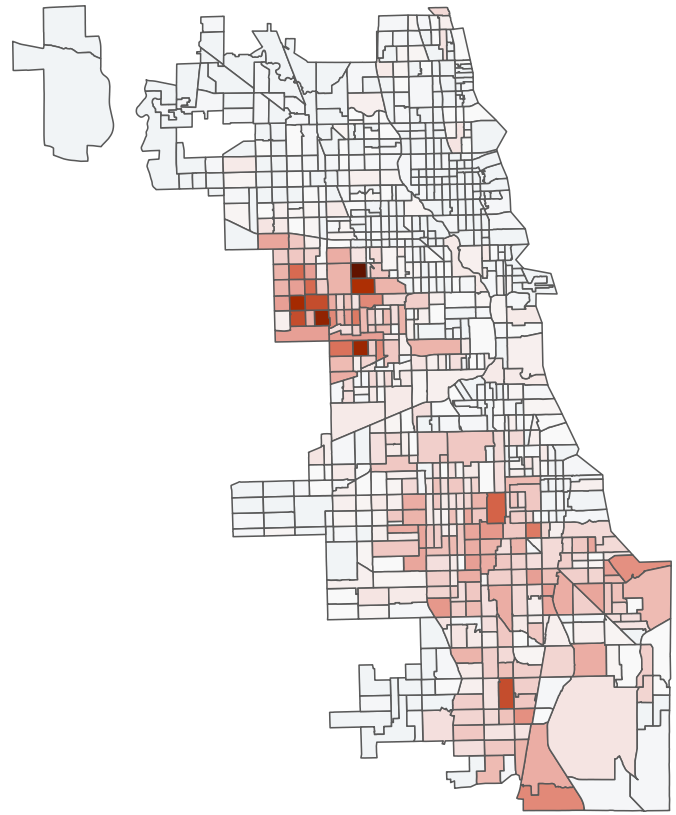
```
neighborhood_heat_map <- neighborhoods %>%
  ggplot() +
  geom_sf(aes(geometry=geometry, fill=homicides), linewidth=.25) +
  theme(axis.text.x = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks = element_blank(),
        rect = element_blank(),
        legend.position = "none") +
  labs(x="", y="") +
  scale_fill_continuous_divergingx(palette = 'RdBu',
                                   mid = median(neighborhoods$homicides),
                                   rev=T,name = "")
```

Census Tract Heat Map

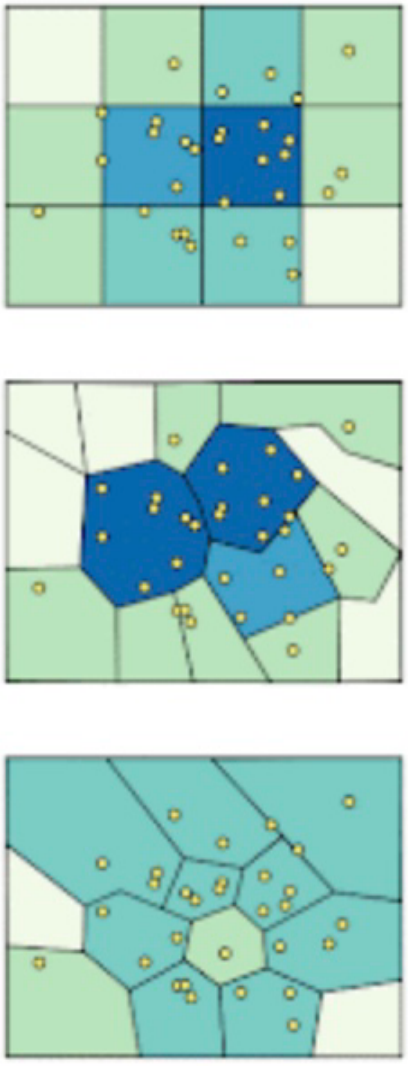
```
tract_heat_map <- census_tracts %>%
  ggplot() +
  geom_sf(aes(geometry=geometry, fill=homicides), linewidth=.25) +
  theme(axis.text.x = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks = element_blank(),
        rect = element_blank(),
        legend.position = "none") +
  labs(x="", y="") +
  scale_fill_continuous_divergingx(palette = 'RdBu',
                                   mid = median(census_tracts$homicides),
                                   rev=T,name = "")
```

Heat Map Comparison





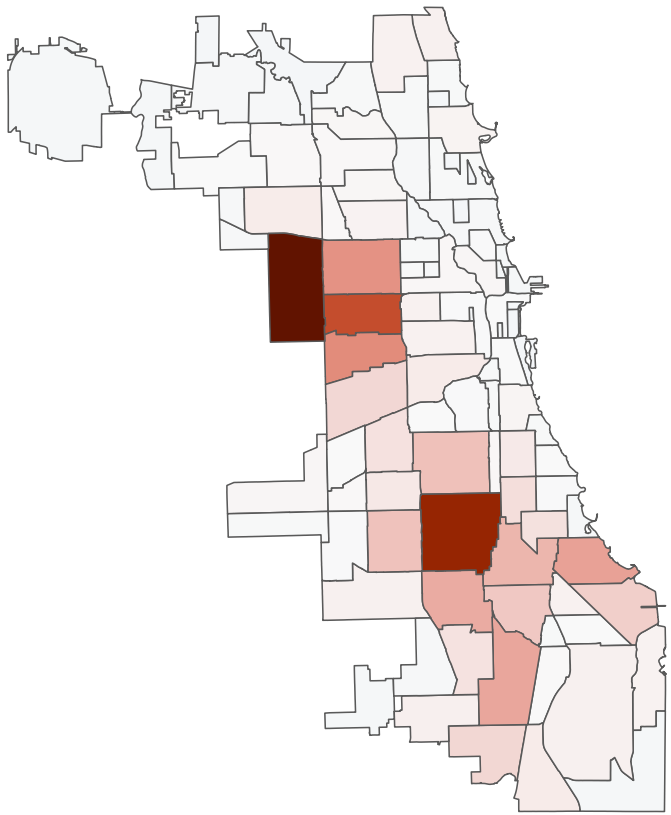
Modifiable Areal Unit Problem (MAUP)

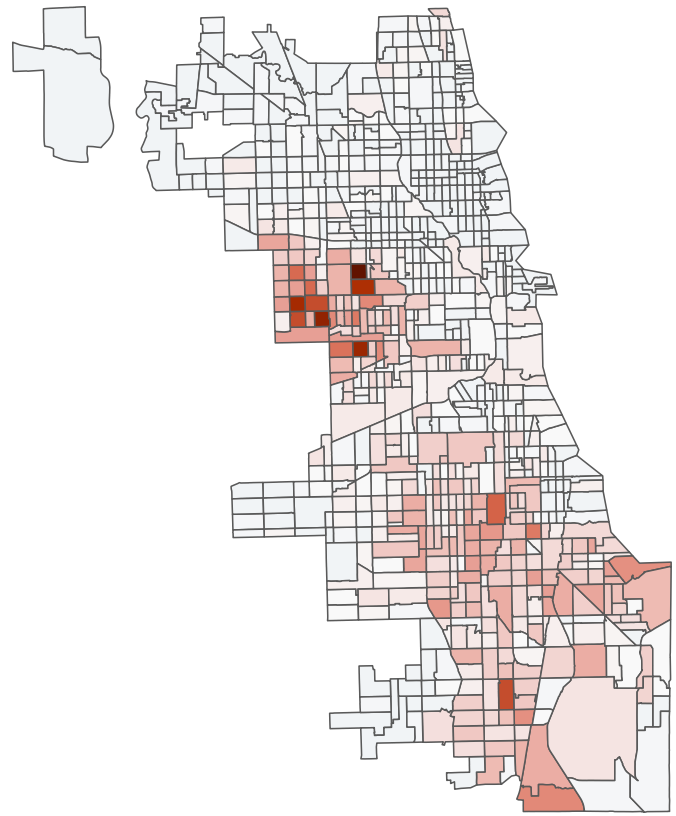


- Any spatial analysis will be sensitive to the **size** and **shape** of the geographies you are analyzing.
- A **scale effect** refers to how the results of your analysis can change when the size or scale of the geographic boundary is altered (i.e., census tract $\rightarrow$ neighborhood).
- A **zone effect** refers to how the results of your analysis can change when the boundaries of the geographic unit are altered (why are neighborhoods delineated one way and census tracts another?).

Figure 1:
Blogspot Post

Now compare with MAUP in mind. What jumps out? How to choose a geography?





Statistical Techniques for Point Pattern Analysis

Point Pattern Analysis

- Heat maps and other visuals can be informative but often want to do more rigorous analysis of spatial point patterns
- We are interested in understanding the spatial distribution of homicides in Chicago:
  - Are homicide events distributed randomly, clustered, or dispersed?
  - What is the overall density or intensity of events across the study area? What are the underlying processes driving the spatial distribution of points?
  - Are there statistically significant clusters of events in the study area?
  - Where are the areas with significantly higher or lower homicide event densities compared to the surrounding areas?
- We can apply these questions and framework to analyzing any point pattern data within a region

Three techniques

- Kernel Density Estimation
- Global and Local Moran's I
- Getis-Ord/G*

Three techniques: applications

- **Kernel Density Estimation**: particularly useful for exploratory data analysis for identifying high-density areas (clusters) and low-density areas (gaps) in your data
- **Global and Local Moran's I**: assess clustering, dispersion, or randomness in the spatial patterns of your data
- **Getis-Ord/G**: identify statistically significant hotspots or coldspots in spatial data

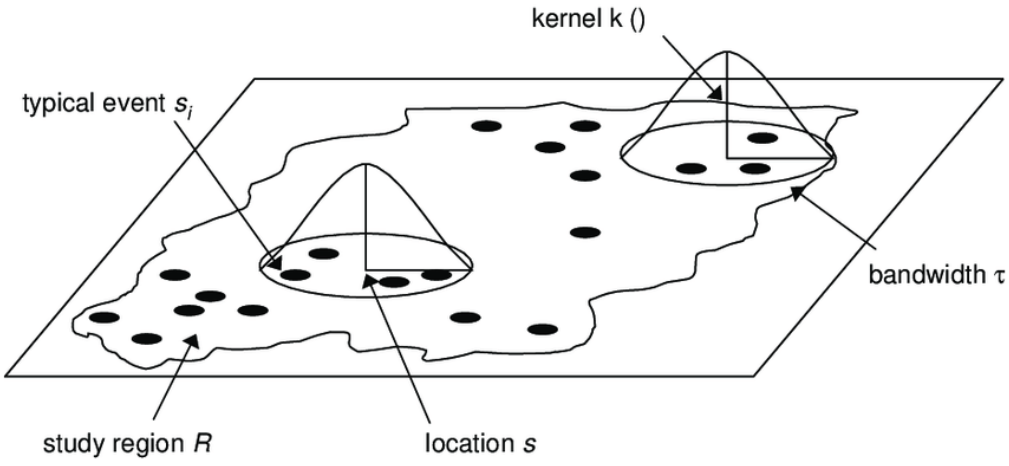
Spatial Point Patterns

- Countable sets of points that arise as realizations of stochastic point processes taking values in $A \subset R^2$
- A spatial point pattern can be denoted as $\{x_1, x_2, ..., x_{N(A)}\}$, where $N(A)$ is the number of points in A where $N(A)$ is a random variable.
- Thus, different realizations of the spatial point process may result in both different numbers and locations of points (refer to the points as events)
- See Chapter 17 of Spatial Statistics for more

Kernel Density Estimation

- Imagine you have a map with a bunch of dots representing events, like the locations where people spotted rare birds in a park.
- Kernel density estimation is like spreading out a smooth layer of paint over each dot on the map. The more dots there are in an area, the thicker the paint layer becomes at that spot.
- Now, if you look at the whole map after spreading this paint, you'll notice some areas have thicker paint layers, showing where the events are more concentrated.
- KDE helps us see where things happen more often and where they're rarer, helping us understand the overall pattern of events across a region

Kernel Density Estimation



Source: Bailey and Gatrell, 1995, p. 86.

- A kernel of a given radius is used to scan an area and the density of any particular area is the number of points within the kernel surrounding that point
- Locations surrounded by clusters of points will have higher values and ones with few or no points will have lower values (Minn)

Kernel Density Estimation

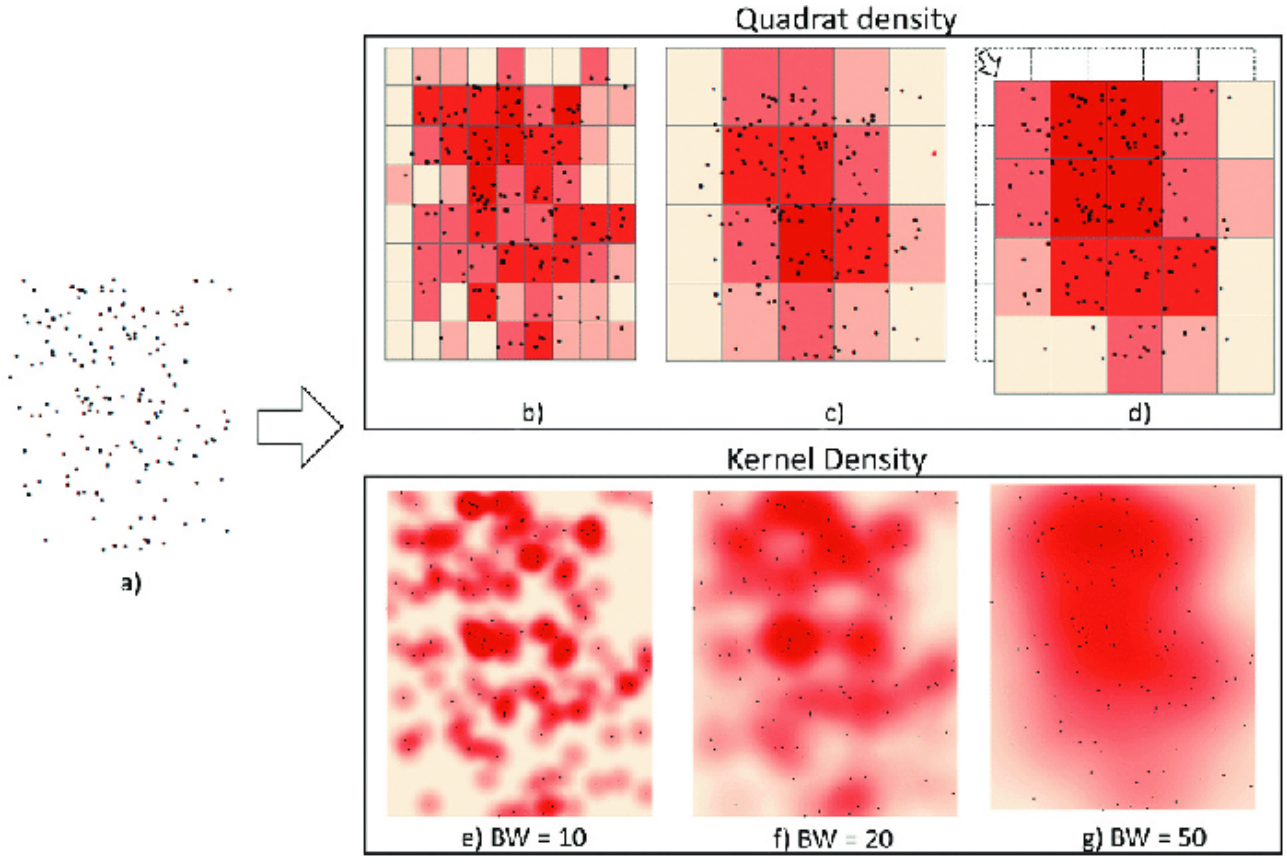


Figure 2: Source: ResearchGate

Kernel Density Estimation: Airport Density in Alaska

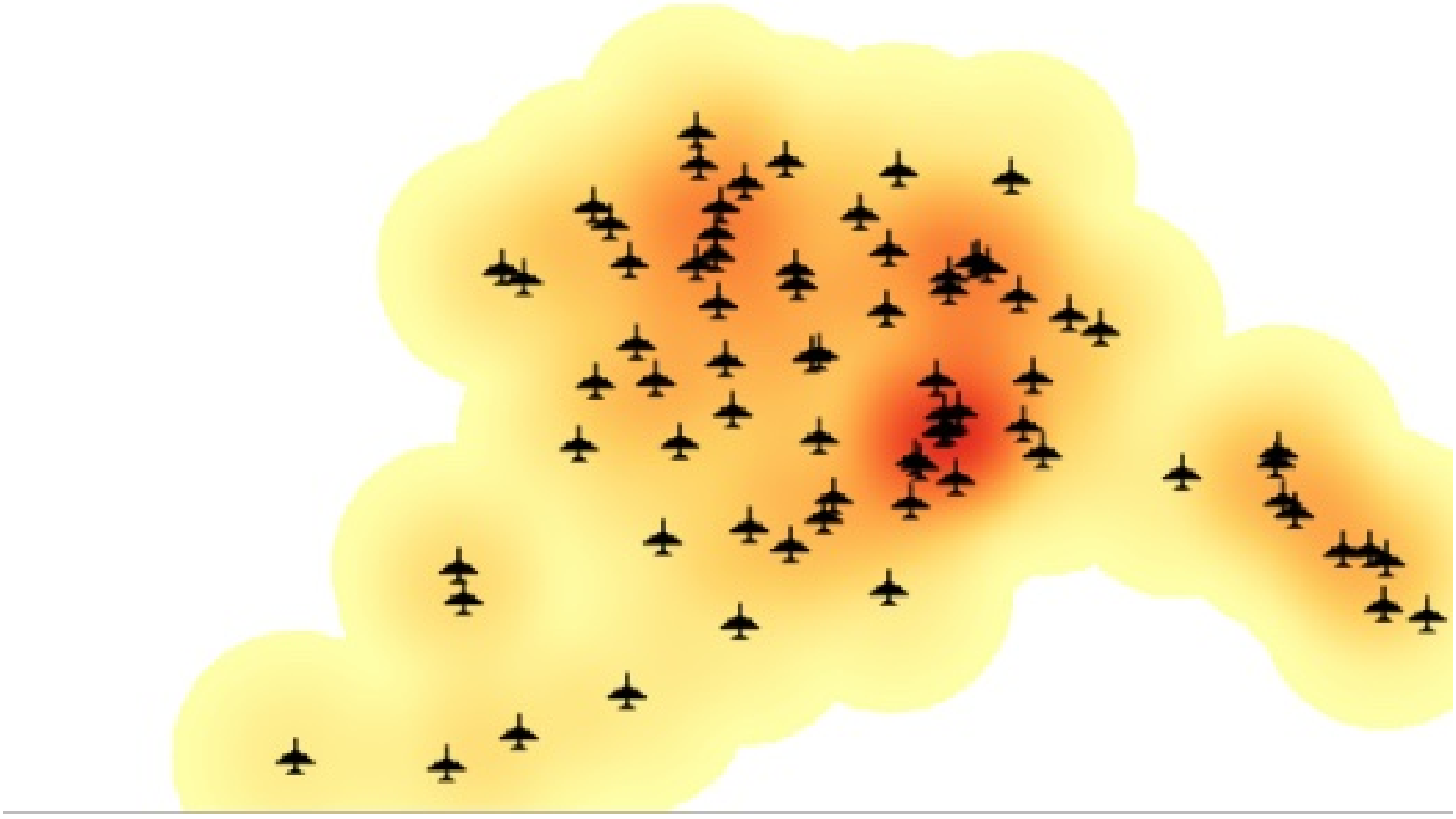


Figure 3: Source: QGIS

spatstat

- We will use `spatstat` package's `density()` function to obtain a kernel estimate of the intensity of a point pattern
- Technical notes on `spatstat`:
  - spatial point patterns are represented as objects of class `ppp` (planar point pattern)
  - the `as.ppp()` function converts a `sf` object to a `ppp` object
  - computes a kernel smoothed estimate of the intensity of a point pattern, where the intensity of a point pattern is simply the expected number of events per unit area–estimate this by using the observed number of events per unit area

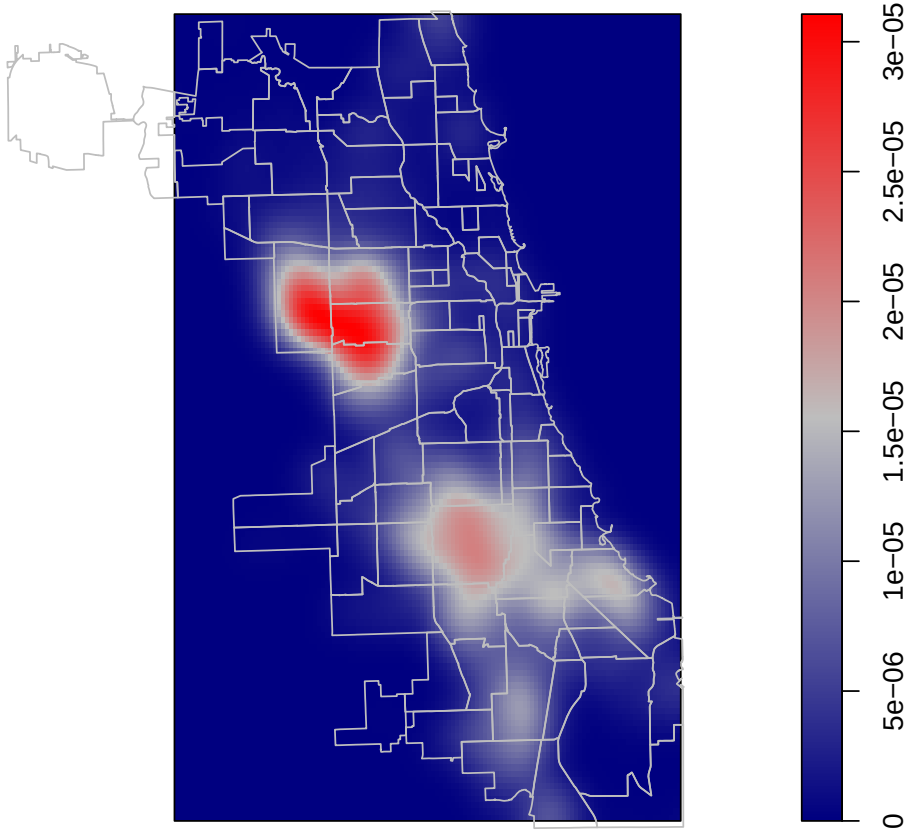
Using `density()` on Chicago Homicide Events

- **sigma**: smoothing bandwidth, smaller value is more granular clusters
- **edge**: default=T, logical value to apply edge correction
- **at**: default="pixels", defaults to computing intensity values at a grid of pixel locations

```
# project the points to Illinois East State Plane Coordinate System (crs=6454)
# then convert to ppp object
ppp.homicide <- st_transform(point_sf$geometry, crs=6454) %>%
  as.ppp()

neighborhood_borders = st_transform(neighborhoods$geometry, 6454)
tract_borders = st_transform(census_tracts$geometry, 6454)

density.homicide <- density(ppp.homicide, sigma=1000)
```

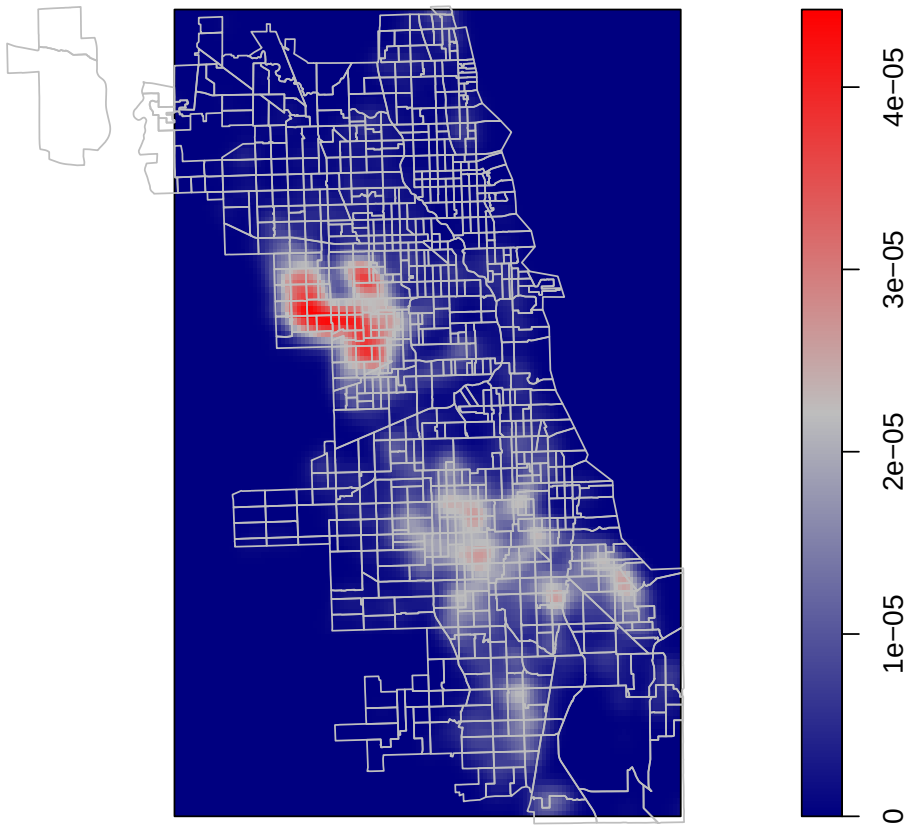
Output

```
plot(density.homicide, col=colorRampPalette(c("navy", "gray", "red")), main="")
plot(neighborhood_borders, col=NA, border="gray", add=T)
```



Let's try a smaller sigma and overlay with tracts

```
density.homicide <- density(ppp.homicide, sigma=500)

plot(density.homicide, col=colorRampPalette(c("navy", "gray", "red")), main="")
plot(tract_borders, col=NA, border="gray", add=T)
```



Moran's I: Understanding Spatial Autocorrelation

- **What is spatial autocorrelation?**
  - the degree of similarity between values of a variable at different locations in space
  - First Law of Geography: "Everything is related to everything else, but near things are more related than distant things" (Tobler 1970)
  - It assesses whether similar values are clustered together (positive autocorrelation), dispersed apart (negative autocorrelation), or randomly distributed (no autocorrelation).
- **Why Use Moran's I?**
  - Moran's I is an index that summarizes the degree to which similar observations tend to occur near each other
  - Moran's I helps us identify spatial patterns by allowing us to statistically test for the presence of spatial autocorrelation

Spatial Autocorrelation

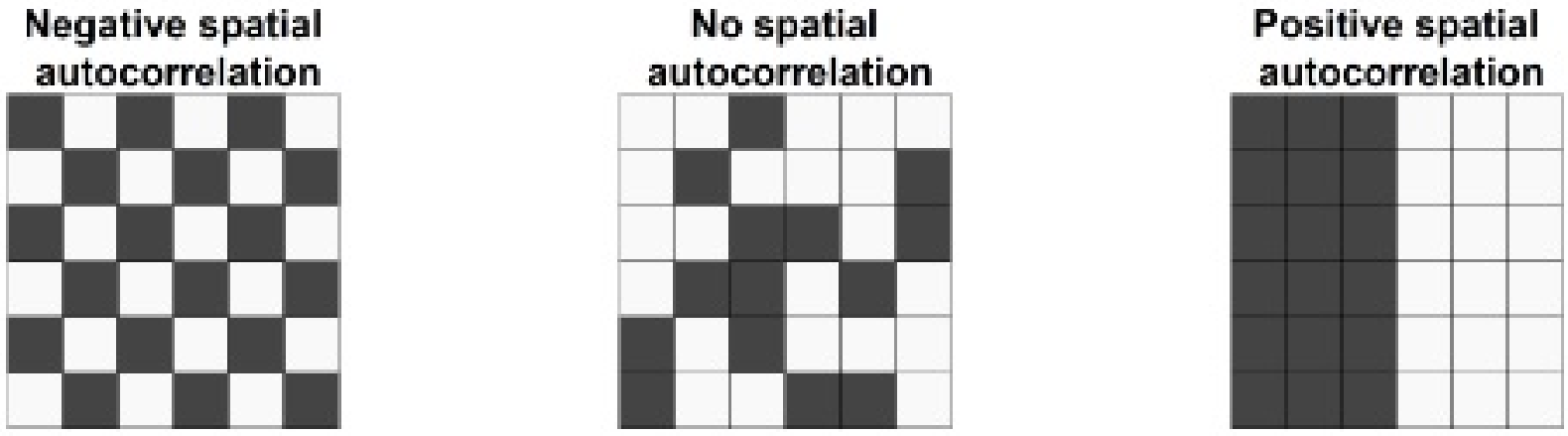


Figure 4: Source: Spatial Statistics, Ch. 8

- Positive spatial autocorrelation: observations with similar values are close together (clustered)
- Negative spatial autocorrelation: observations with dissimilar values are close together (dispersed)
- No spatial autocorrelation

Moran's I

The Global Moran's I (Moran 1950) quantifies how similar each region is with its nieghbors and averages all these assessments. It takes the form

$$I = \frac{n \sum_i \sum_j w_{ij}(Y_i - \bar{Y})(Y_j - \bar{Y})}{(\sum_{i \neq j} w_{ij}) \sum_i (Y_i - \bar{Y})^2},$$

- where n is the number of regions
- Y_i is the observed value of the variable of interest in region i
- $\bar{Y}$ is the mean of all values
- w_{ij} are spatial weights that denote the spatial proximity between regions
- i and j, with $w_{ii} = 0$ and $i, j = 1, \ldots, n$.
- The definition of the spatial weights depends on the variable of study and the specific setting. (Spatial Statistics, Ch. 8)

Global Moran's I

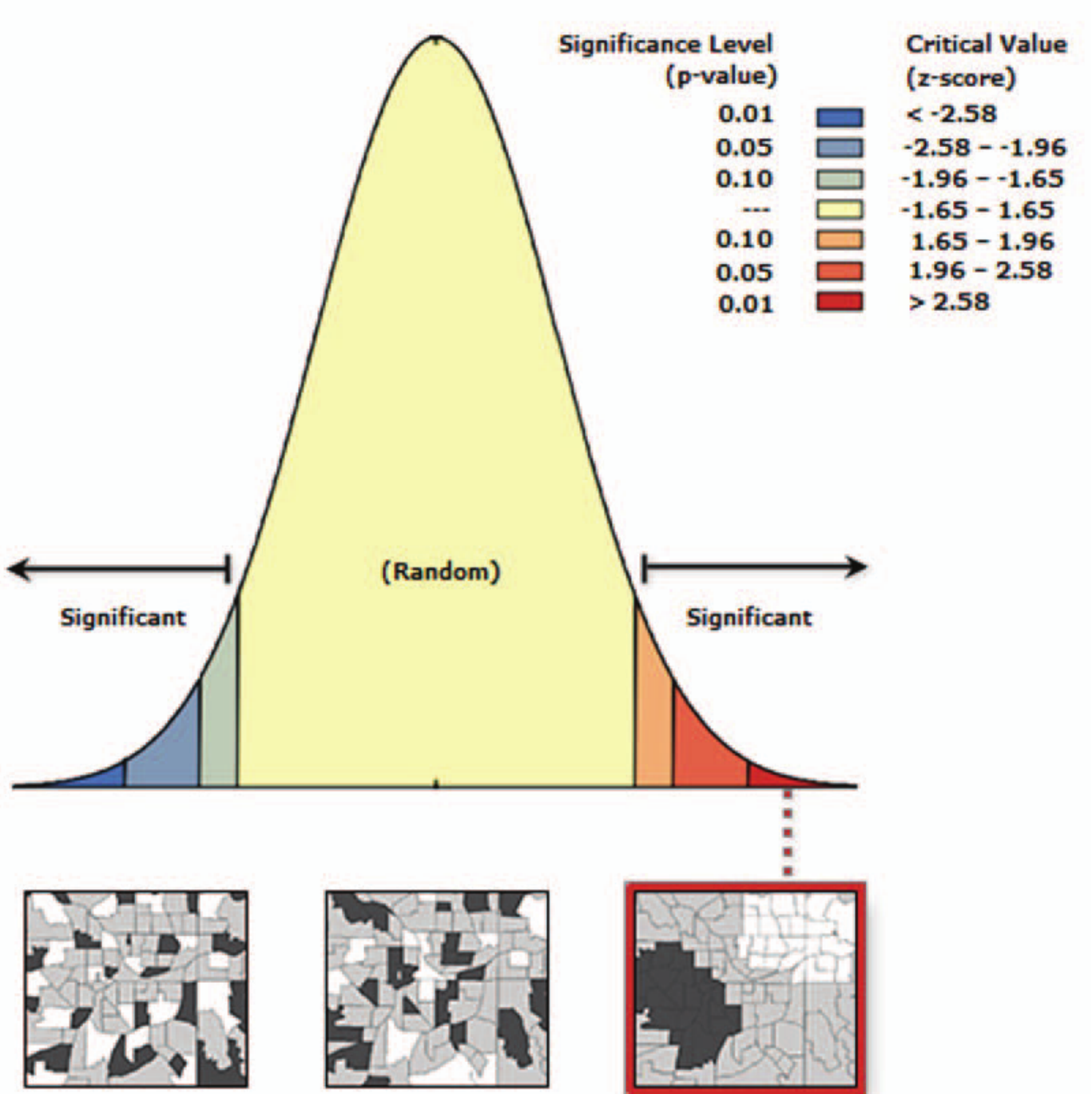
- Test the presence of spatial autocorrelation using I, which quantifies how similar each region is with its neighbors and averages all of the assessments
- Our null hypothesis is no spatial autocorrelation
- Under the null, observations Y_i are i.i.d. and I is asymptotically normally distributed with mean and variance equal to:

$$E[I] = -\frac{1}{n-1}, Var[I] = E[I^2] - E[I]^2$$

- Then, we can calculate a z-score for I:

$$z_i = \frac{I - E[I]}{\sqrt{V[I]}}$$

Global Moran's I



Choosing Neighbors

- What does it mean to be a neighbor?
  - Adjacency? ie, all the areas next to the area of interest
  - Proximity? ie, all the areas within a specific distance to the area of interest
- Common choice is queen's contiguity:

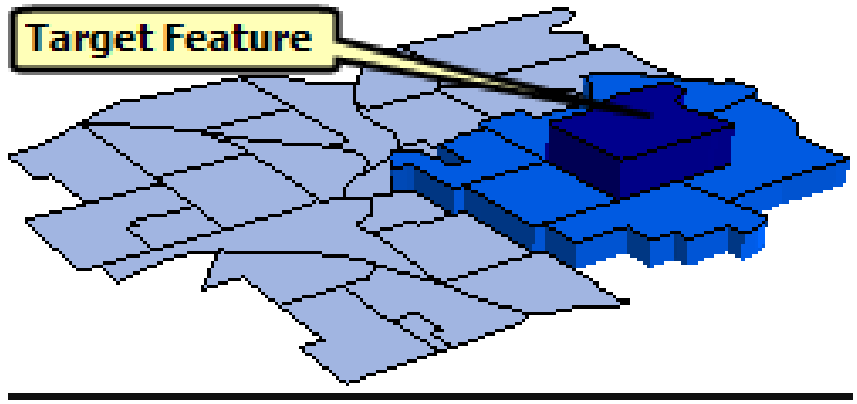


Figure 6: Source: ArcGIS

- Other contiguity, ie, rook or bishop
- Inverse distance (1/d): near things have a larger weight than things far away

Steps to testing for spatial autocorrelation using Moran's I:

1. State the null and alternative hypotheses:

$H_0 : I = E[I]$ (no spatial autocorrelation)
$H_0 : I \neq E[I]$ (no spatial autocorrelation)

2. Choose significance level

3. Calculate the test statistic:

$$z = \frac{I - E[I]}{\sqrt{V[I]}}$$

4. Find the p-value for the observed data by comparing the z-score to the standard normal distribution.

- Recall, the p-value is the probability of obtaining a test statistic as extreme as or more extreme than the one observed test statistic.

5. Reject if p-value $< \alpha$, fail to reject if p-value $\geq \alpha$

Let's see what Moran's I says about homicide in Chicago

```
# generate weights matrix using queens contiguity
neighbors = poly2nb(as_Spatial(neighborhoods), queen=T)
weights = nb2listw(neighbors, style="W", zero.policy=T)

#testing the null of negative or no spatial autocorrelation
moran.2020 = moran.test(neighborhoods$homicides,
                        weights, alternative="greater")

print(moran.2020)
```

```
##
##  Moran I test under randomisation
##
## data:  neighborhoods$homicides
## weights: weights
##
## Moran I statistic standard deviate = 6.029, p-value = 8.248e-10
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##       0.357300614      -0.010309278       0.003717762
```

```
print(moran.2020[["p.value"]] < .05)
```

```
## [1] TRUE
```

Local Moran's I

- Global Moran's I only gives you one statistic for a whole region
- Cannot tell you anything about heterogeneity within your region
- Can use Local Moran's I to get a local measure of similarity between each area's value and those of nearby areas

Local Moran's I

- **Local Moran's I Formula**
  - For the ith region, the local Moran's I is defined as:

$$I_i = \frac{n(Y_i - \bar{Y})}{\sum_j (Y_j - \bar{Y})^2} \sum_j w_{ij}(Y_j - \bar{Y})$$

  - The sum of all local Moran's I values equals the global Moran's I:

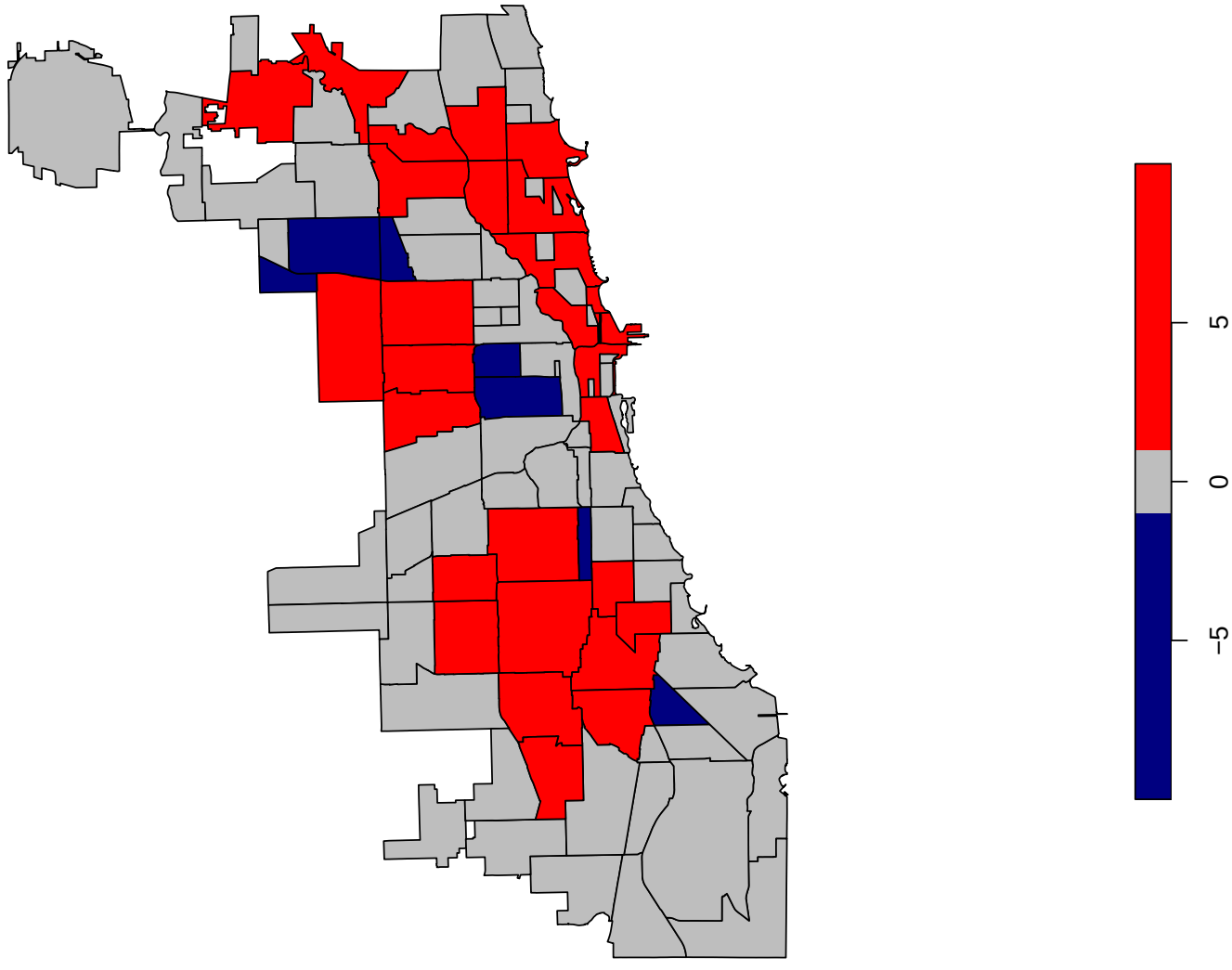
$$I = \frac{1}{\sum_{i \neq j} w_{ij}} \sum_i I_i$$

- **Interpretation of Local Moran's I**
  - High I_i values indicate areas surrounded by similar values (clustering)
  - Low I_i values indicate areas surrounded by dissimilar values (outliers)

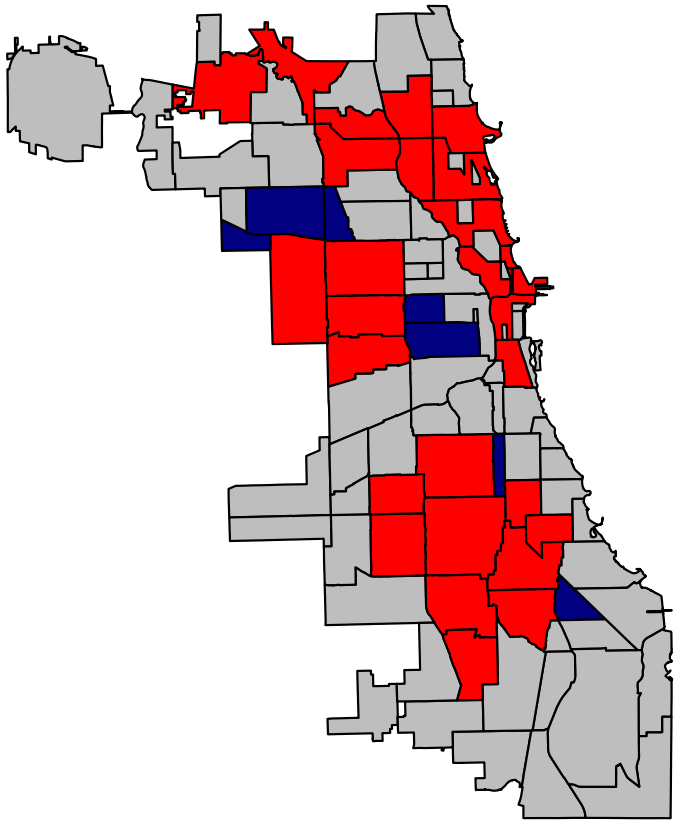
Local Moran's I

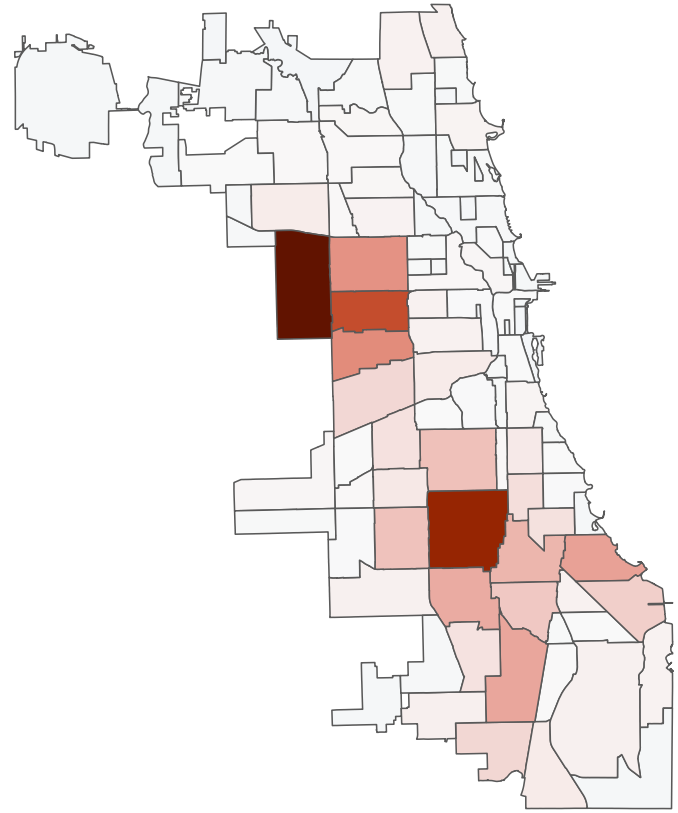
- **Interpreting Results**
  - Local Moran's I is commonly displayed as a z-score or p-values
  - A positive z-score indicates an area that is surrounded by areas with similar values
  - A negative z-score indicates an **outlier** area that is unusually low or high compared to its neighbors
- **Application Example**
  - We will use the `localmoran()` function to compute Local Moran's I for homicides in Chicago
  - set the alternative hypothesis to "greater" to test for positive spatial autocorrelation.

```
local = localmoran(neighborhoods$homicides, weights)
neighborhoods$Morans.2020 = local[,"Z.Ii"]
breaks=c(-10, -1, 1, 10)
plot(neighborhoods["Morans.2020"], breaks=breaks, main="",
         pal=colorRampPalette(c("navy", "gray", "red")))
```



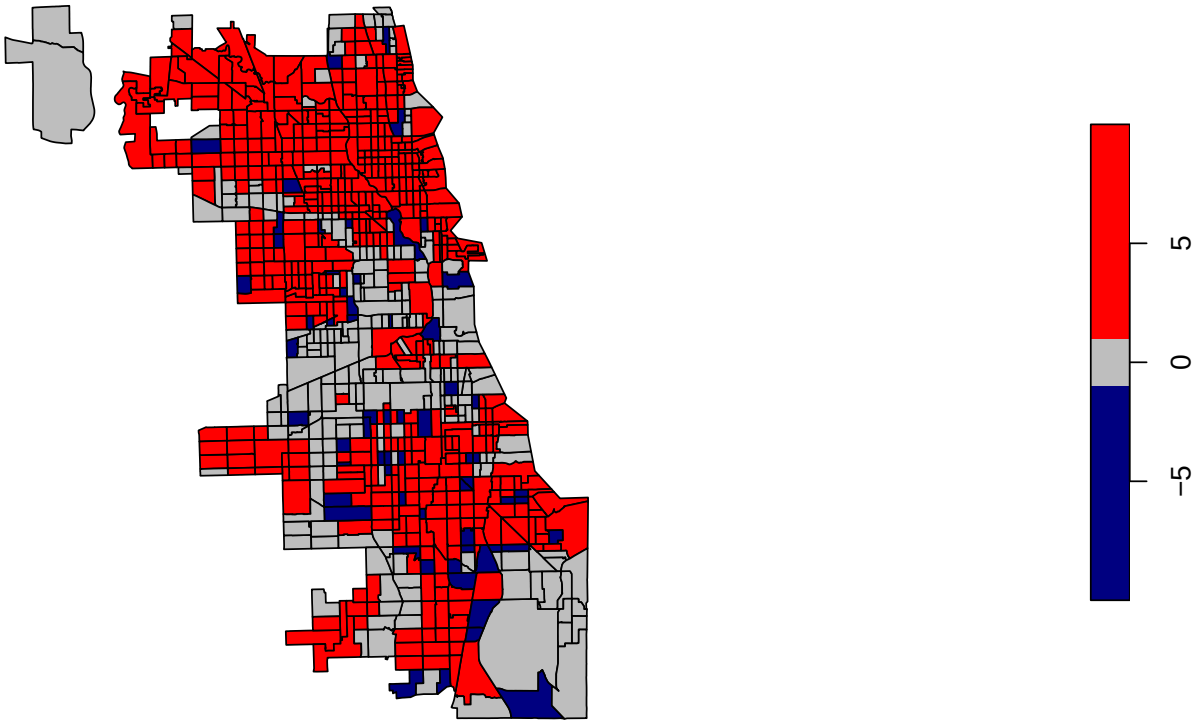
How does this compare with our heat map?



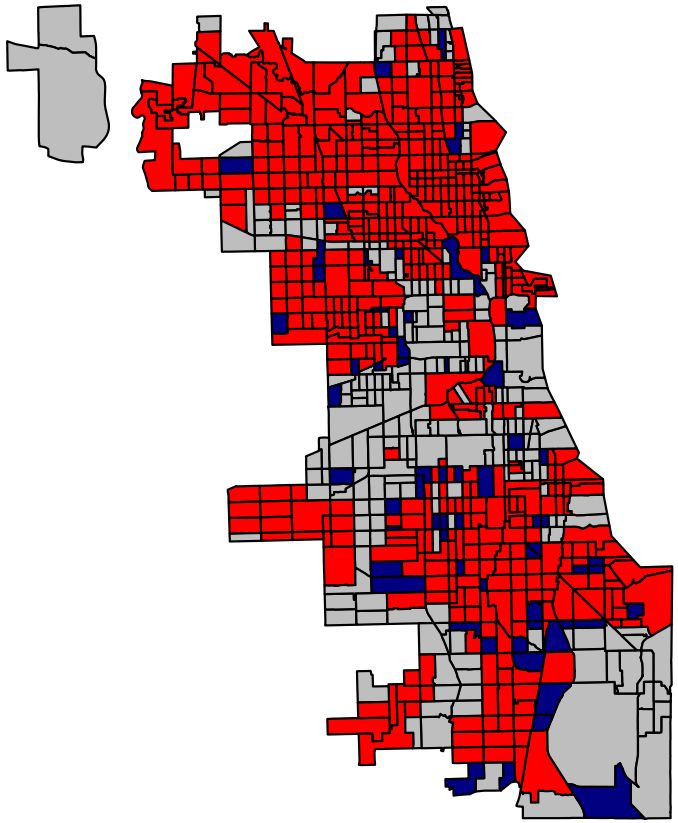


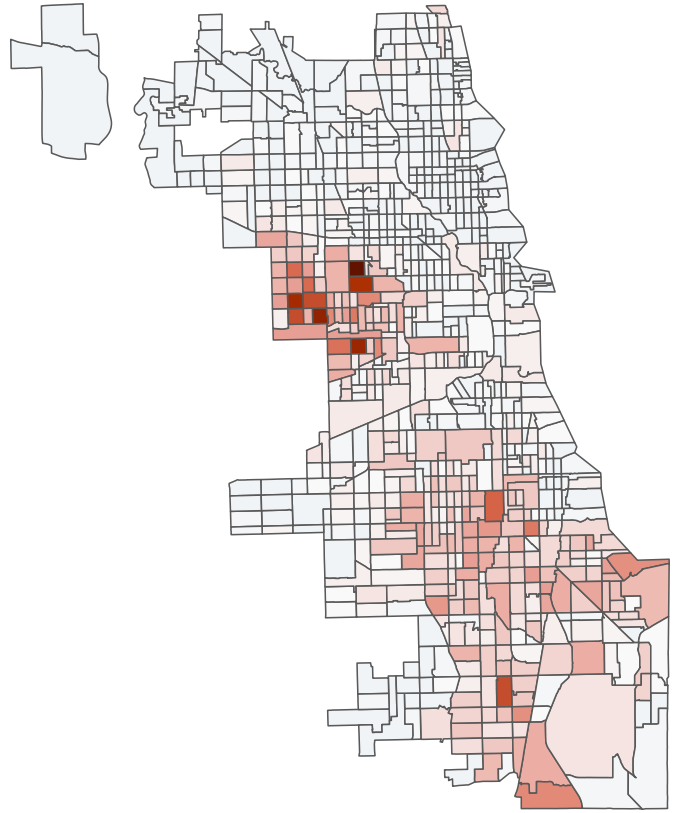
Tracts

```
neighbors_ct = poly2nb(as_Spatial(census_tracts), queen=T)
weights_ct = nb2listw(neighbors_ct, style="W", zero.policy=T)

local = localmoran(census_tracts$homicides, weights_ct)
census_tracts$Morans.2020 = local[,"Z.Ii"]
plot(census_tracts["Morans.2020"], breaks=breaks, main="",
         pal=colorRampPalette(c("navy", "gray", "red")))
```



How does this compare with our heat maps?





Getis-Ord/Gi*: Identifying Hotspots and Coldspots

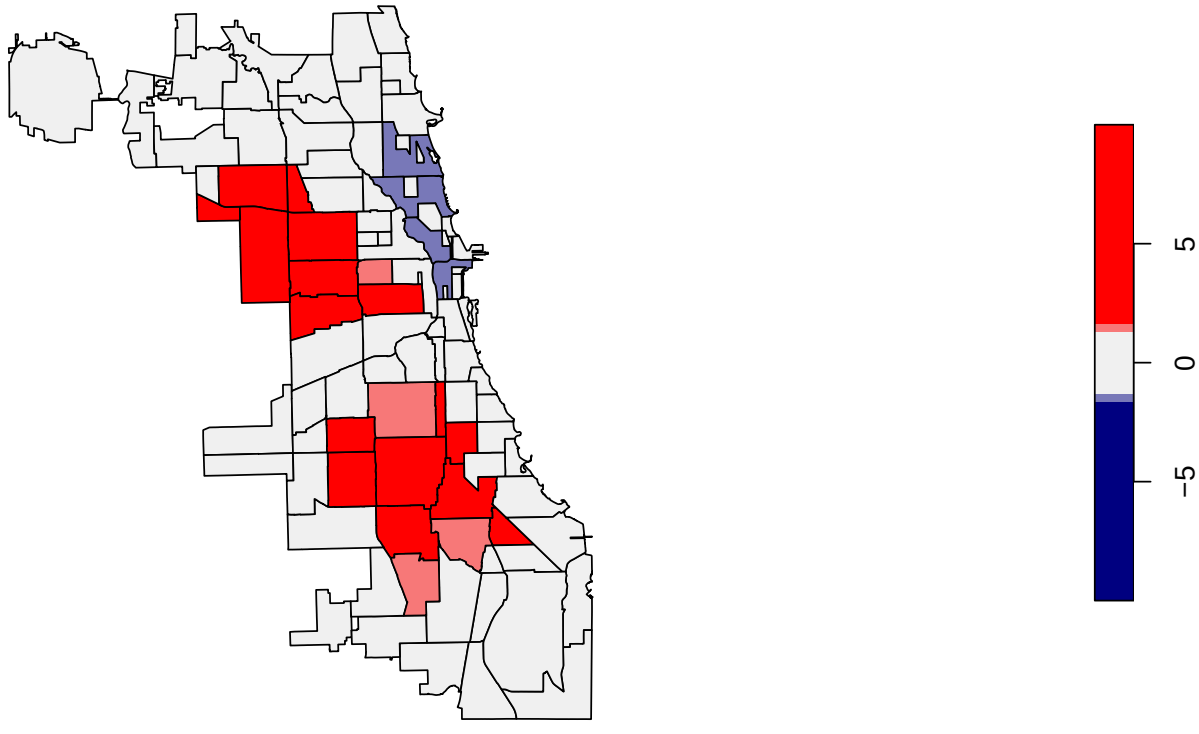
- **What is Getis-Ord?**
  - Getis-Ord (G*) is a spatial statistical technique used to identify hotspots (areas with high values) and coldspots (areas with low values) in spatial data.
  - It measures the degree of spatial clustering of values by comparing the observed value at each location with the values of its neighboring locations.
- **How Does Getis-Ord Work?**
  - Getis-Ord calculates a z-score for each location based on the values of the variable and the spatial distribution of those values.
  - Positive z-scores indicate hotspots, where high values cluster together, while negative z-scores indicate coldspots, where low values cluster together.
  - The magnitude of the z-score indicates the strength of the clustering, with larger z-scores indicating stronger clustering.

Getis Ord

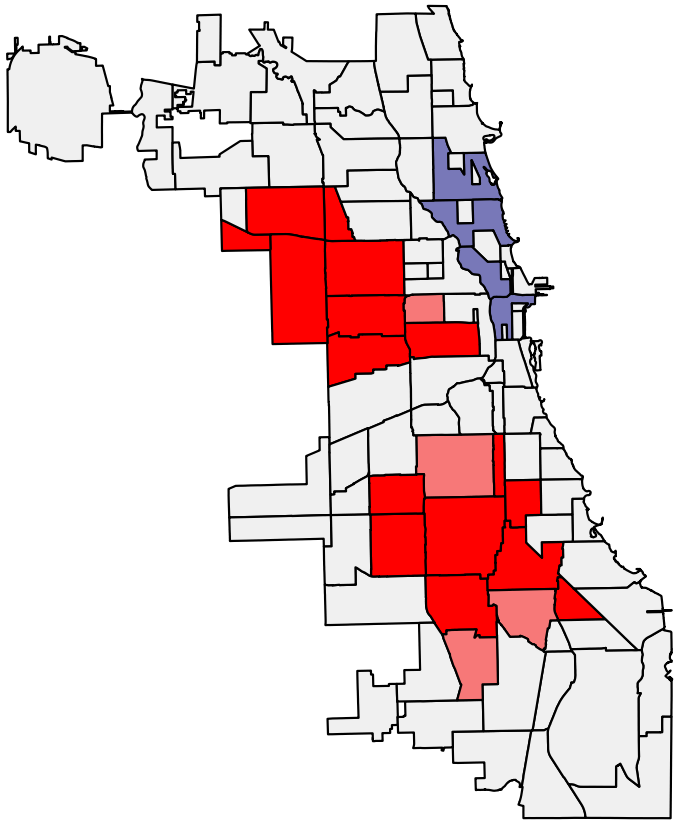
- To be a hotspot, a feature will have a high value and be surrounded by other features with high values
- The local sum for a feature and its neighbors is compared proportionally to the sum of all features
- when the local sum is very different from the expected local sum, and when that difference is too large to be the result of random chance, a statistically significant z-score results (ArcGIS)

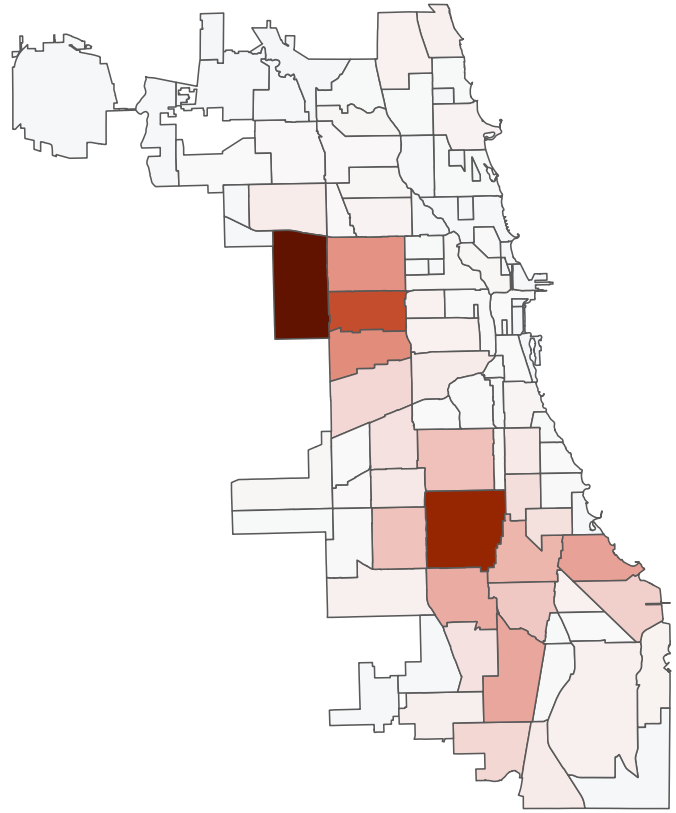
Getis Ord

```
neighborhoods$Getis.Ord.2020 = localG(neighborhoods$homicides, weights)

plot(neighborhoods["Getis.Ord.2020"], main="",
    breaks=c(-10, -1.645, -1.282, 1.282, 1.645, 10),
    pal=colorRampPalette(c("navy", "#f0f0f0", "red")))
```



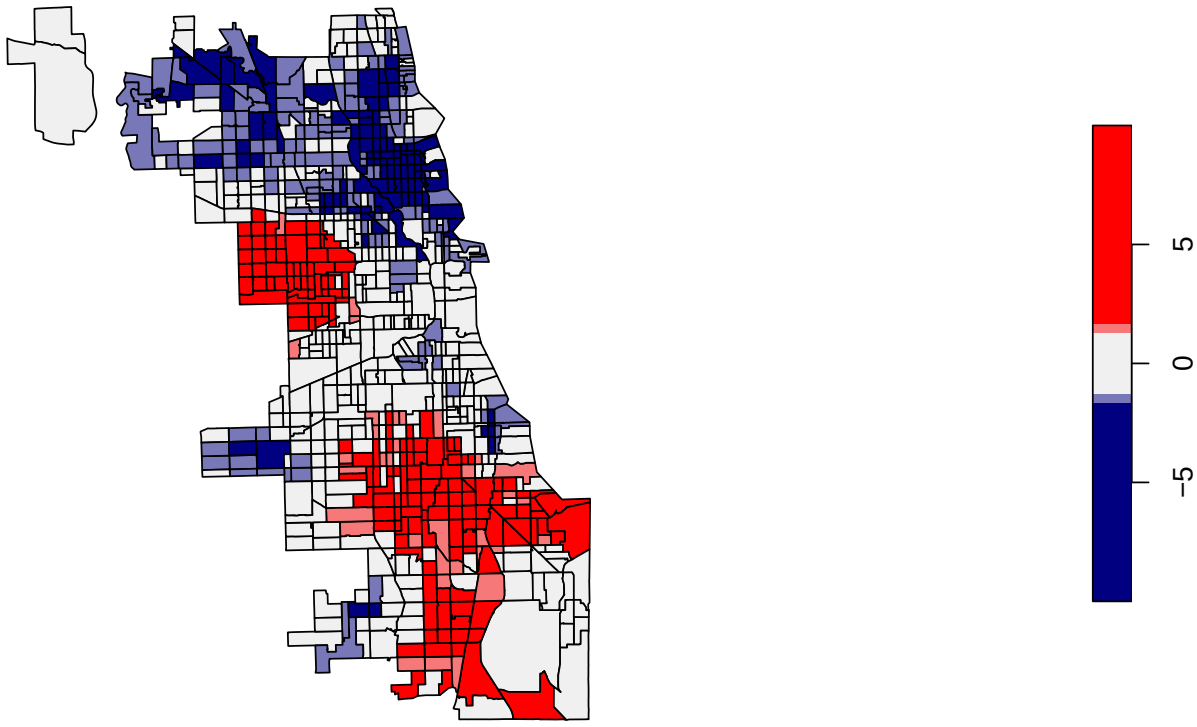
How does this compare with our heat map?



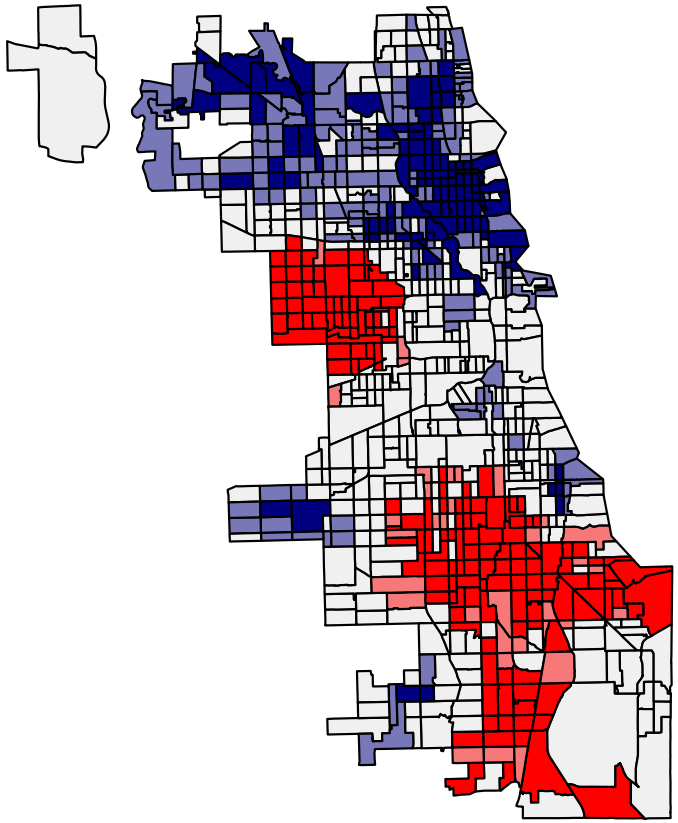


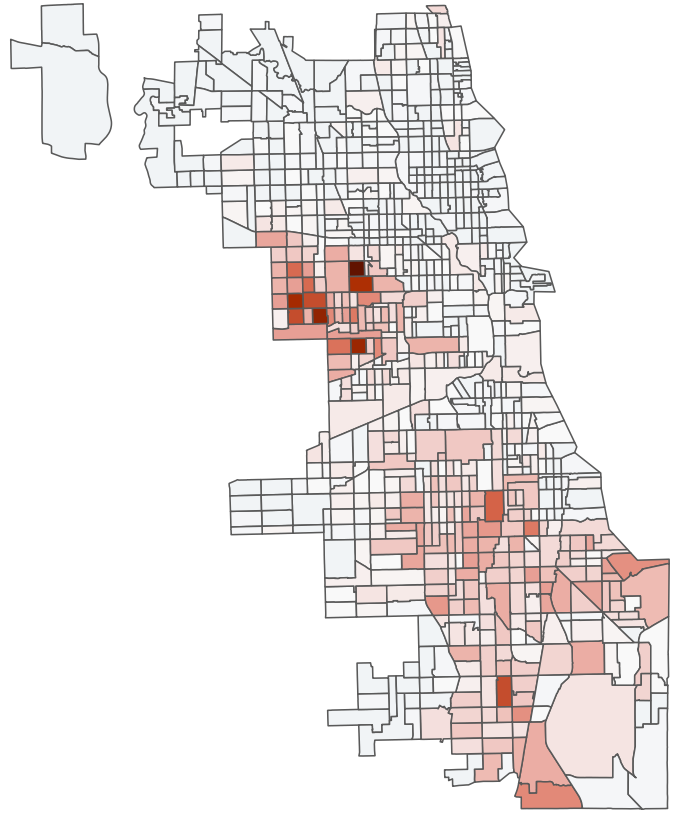
Tracts

```
census_tracts$Getis.Ord.2020 = localG(census_tracts$homicid

plot(census_tracts["Getis.Ord.2020"], main="",
    breaks=c(-10, -1.645, -1.282, 1.282, 1.645, 10),
    pal=colorRampPalette(c("navy", "#f0f0f0", "red")))
```



How does this compare with our heat map?





Wrap-up

Wrapping up

- We covered some statistical techniques to help you analyze point pattern data
- If you have point pattern data and boudary files, you can use this code to conduct similar type of analysis
- Can additionally add in more information about demographics and consider other techniques like spatial regression
- Great resources:
  - https://michaelminn.net/tutorials/r-crime/
  - Spatial Statistics for Data Science: Theory and Practice with R: https://www.paulamoraga.com/book-spatial/index.html
- Thanks!

References:

https://keith-travelsinindonesia.blogspot.com/2011/11/modifiable-areal-unit-problem-and.html

source: https://michaelminn.net/tutorials/r-crime/