Homework Assignment 2: Spatial Point Pattern Analysis and Spatial Autocorrelation Using R

SSCI 683 Spring 2022
March 18, 2022

Andy Kampfschulte

Contents

Learning objectives:
The objectives of this homework assignment is for you to:

- Explore spatial autocorrelation using global Moran's I and Moran Scatterplot
- Use kernel density estimation (KDE) and demonstrate the KDE map results
- Use Ripley's K-function to evaluate the point pattern of your spatial data
- Interpret the output results for the dataset's spatial dependency and spatial patterns

Before getting into your own work in this assignment, you may want to follow the R scripts and practice the example datasets in Handout 3 for spatial autocorrelation and Handout 4 for KDE and Ripley's K.

1. Import a spatial dataset of your interest. If your data is a polygon data (e.g. a polygon shapefile), you can build the spatial weights matrix using contiguity-based spatial relationship as Handout 3. This in turns means you'd need to find another point dataset or, better, take centroids of your polygon data for KDE and Ripley's K in the next steps. If your data is a point data, your approach to build a spatial weights matrix would need to be distance-based (e.g. k nearest neighbors or fixed distance).

2. Conduct a series of spatial analysis in R, including:

   - Creating a spatial weights matrix (SWM) appropriate for your choice of variable
   - Applying global Moran's I and Moran scatterplot for the variable of interest using the

Data Description

For this assignment, I will be working with Wildfire data within the state of California, in particular the frequency of wildfires larger than 1,000 acres within each census tract across the state. The wildfire data cover 30 years of fires, ranging from 1990 - 2020.

California state and census-tract data were obtained using the `tigris` `r` package. This package operates on an API from the U.S. Census Bureau's TIGER database. Wildfire data were obtained from several different extracts from CAL-FIRE. The data include a shapefile of the area for each wildfire, along with dates, total acreage, overseeing response agencies, and a slew of additional information. To obtain the count of fires within each tract, an intersection was run between census tracts and fire shapefiles, and any part of the fire perimeter contained within a tract was counted. For brevity, I did not include the entirity of the data management scripts in this assignment, but can these be found in the attached .RMD file.

```
Widfire_Frequency = lengths(st_intersects(tract_geometry, fire_data))
```

Given the nature of this data, and that fires often span across multiple tracts, it is likely that this will contribute to spatial autocorrelation.

Some brief summary statistics are provided in Table 1. In short, there are 8,057 census tracts in California, and the number of Wildfires in these tracts from 1990-2020 ranged from 0 to 50.

Table 1: Summary Statistics of Wildfire Data intersected with Census Tracts

	Value
Numberof Tracts	8057.000
Mean Wildfire Frequency	0.371
Median Wildfire Frequency	0.000
Minimum Wilfire Frequency	0.000
Maximum Wilfire Frequency	50.000
Mean Tract Area (km^2)	52.621
Median Tract Area (km^2)	1.939

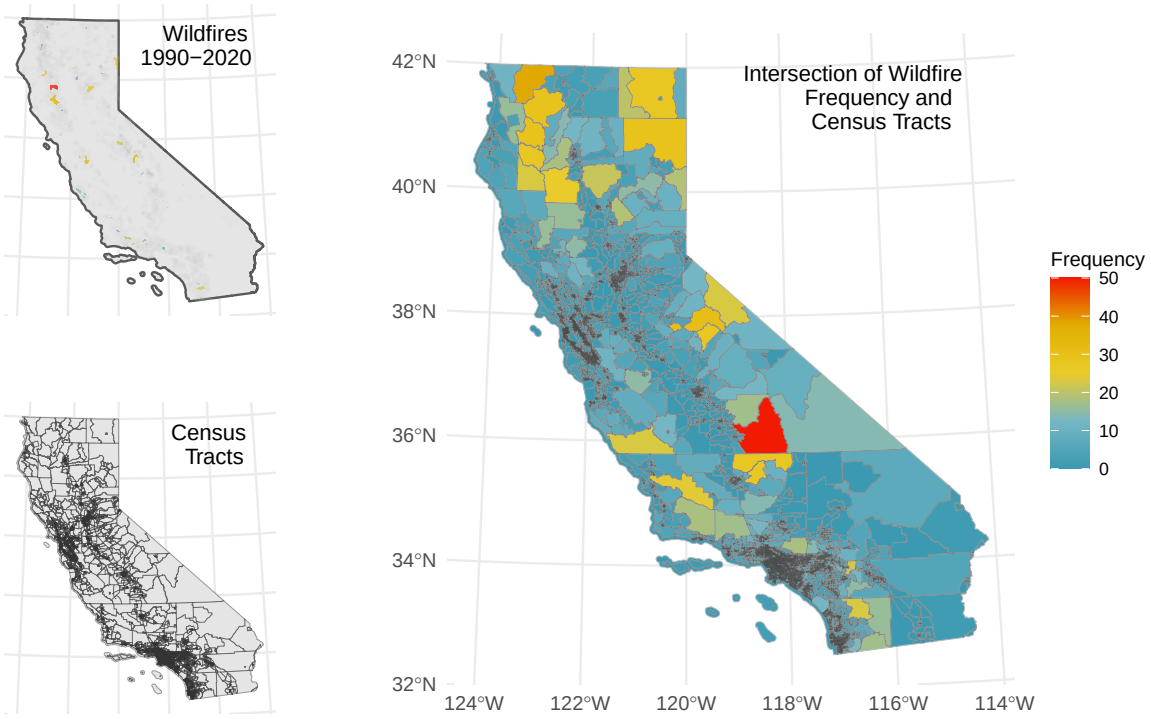

Figure 1: **Top Left:** Distribution of Wildfires (1990-2020), shaded by total area of individual fire. **Bottom Left:** Census Tracts in California. **Right:** Results of the Intersection of Wildfire data and Census Tracts yeidling the total frequency of wildfires within each tract.

Spatial Analyses

Global Moran's I

Starting with creating a neighborhood. The below code, using the `spdep` package, generates a neighborhood of California Census Tracts using the Rook's case.

```
neighbors <- poly2nb(eh, queen = FALSE)
```

```
list <- nb2listw(neighbors, zero.policy = TRUE,
                 style = "minmax")

GM <- moran.test(as(eh,"Spatial")$count, list,
                 zero.policy = TRUE)
```

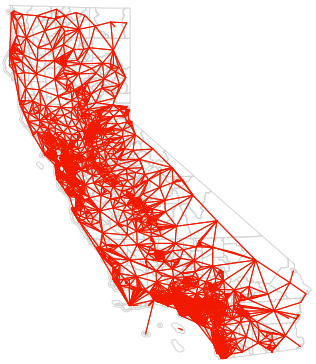

Figure 2: Plot of neighborhood of census tracts in California.

That neighborhood list is then out through the `nb2listw` function to create weights of the neighborhood. I'm using the `style = "minmax"` for the coding scheme of the weights. The default coding scheme is `style = "W"`, which is row-standardized, summing over all links to a given polygon. `minmax`, on the other hand, *"divides the weights by the minimum of the maximum row sums and maximum column sums of the input weights"*. I chose this for no particular reason, other than to explore the effects of different weighting schemes on the Global Moran's I. Running everything through the `moran.test` function (and cleaned up using my own functions). We can see that the Global Moran's I is 0.656, and the p-value is 0. Meaning that there is positive spatial autocorrelation in Wildfire frequency between census tracts in California. This makes total sense, the large swaths of forest in northern California are going to be more susceptible to fires than the metropolitan areas in Southern California, and census tracts near one another are gong to have similar levels of wildfire risks.

Table 2: Global Moran's I Estimate

Moran's I	Expectation	Variance	p-value
0.6560921	-0.0001241	4.44e-05	<0.001

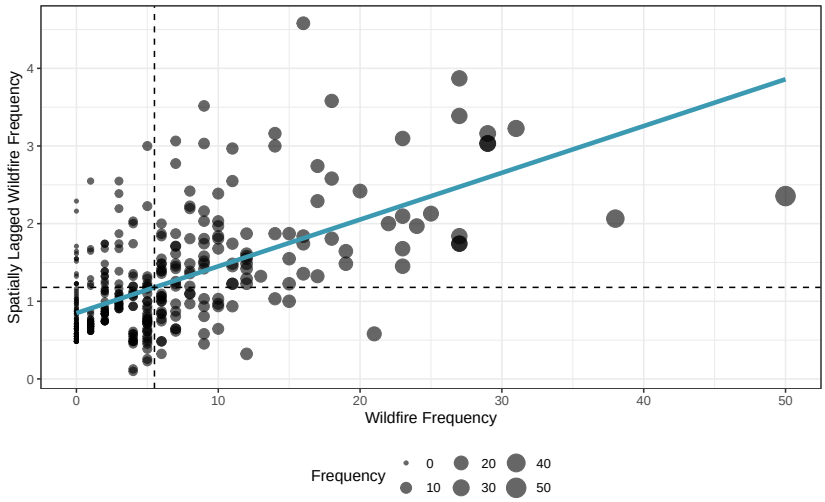

Figure 3: Moran's I scatterplot. Points are sized based on Frequency of Wildfires.

Local Moran's I

Local Moran's I was generated using the `localmoran` function. These Local *I* values were then added to the data set and plotted (Figure 4). Fisher-style cuts in the distribution of I-values were made to create breaks and make the map easier to interpret.

As you can see, there's very strong spatial autocorrelation in Northern & Eastern California. Again, this makes total sense, as these are the areas of the state with the highest potential to have a wildfire. Conversely, the San Fernando Valley, the high-population density areas (the bay area, and LA-San Diego), and the desert all have near zero - or at least relatively low - spatial autocorrelation. When p-values > 0.05 are eliminated, and tracts are cluster based on their quadrant position in the Moran plot (Low-Low, Low-High, High-Low, High-High)m we can see that only High-High and Low-High autocorrelated tracts are significant. Plotted onto a map, we see the High-High areas being in highly forested areas, and the Low-High tracts being directly adjacent to High-High tracts. WIthin the context of the data, this seems to make sense, as the statistically insignificant area really have no fire history, and therefore not much correlation between adjacent tracts.

```
LM <- localmoran(as(eh,"Spatial")$count, list,
                 zero.policy = TRUE)

LM.map <- cbind(eh, LM)
```

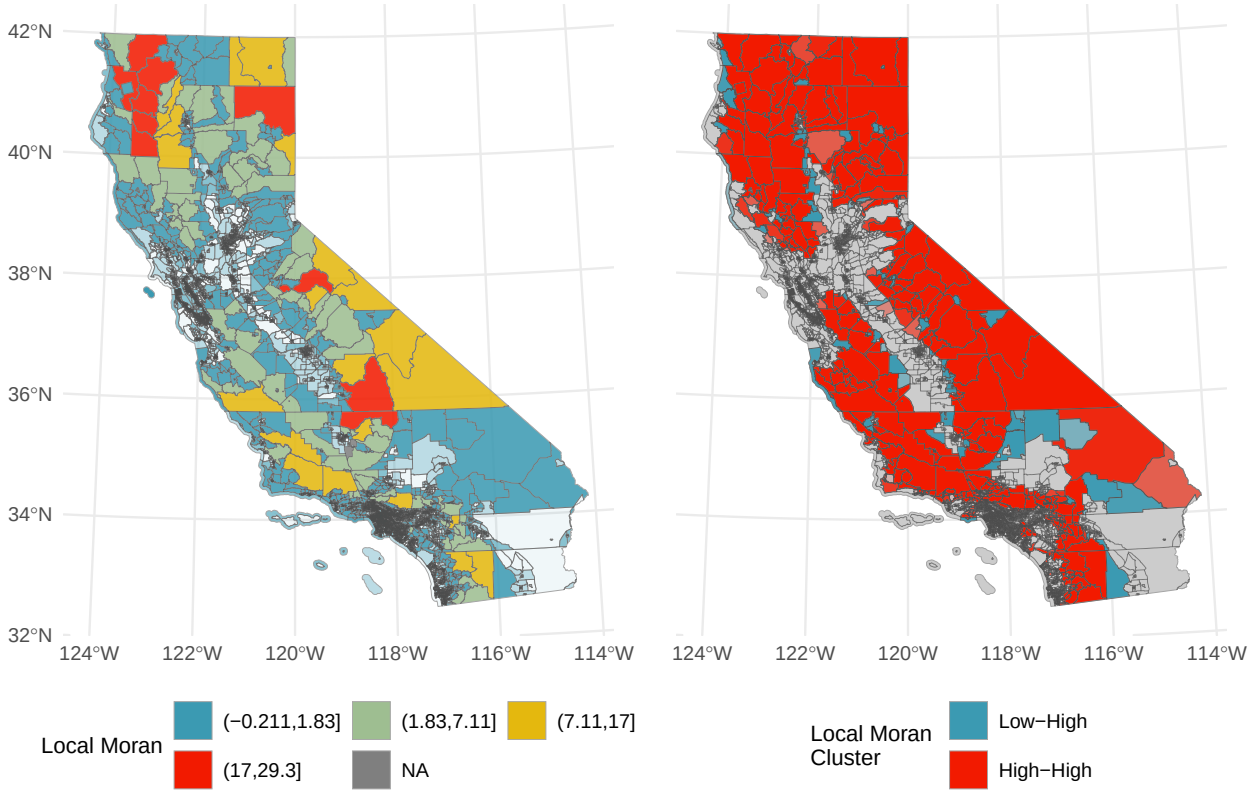

Figure 4: Results of local Moran's I statistic. The figure on the left displays the Local Moran's I statistic for each census tract in california, binned into 4 categories using the Fisher's Jenks algorithm. Alpha shaded was added to fade out tracts with a p-value > 0.05. The right-hang figure shows the clustering of tracts based on Moran's I.

Kernel Density Estimation

The `SpatialKDE` package was used to calculate kernel density of census tracts. I wanted to explore KDE estimate functions that could accept `sf` objects. There is a seperate function that manually does this in the appendix. Cell size was set 5Km and the bandwidth was set to 20,000. It could have been set to a higher value to get more peaks in additional metropolitan areas, but at the set bandwidth we can see clear densities in the LA and San Francisco Bay Area (Figure 5). When compared to fire frequency, the KDE hot spots correlate with low-fire areas with little spatial autocorrelation.

```
grid <- eh %>%
  create_grid_rectangular(cell_size = 5000,
                          side_offset = 20000)

KDE <- st_centroid(eh) %>%
  kde(band_width = 20000, kernel = "quartic", grid = grid)
```

```
## Warning in st_centroid.sf(eh): st_centroid assumes attributes are constant over
## geometries of x
```

```
KDE2 <- st_intersection(KDE, cali)
```

```
## Warning: attribute variables are assumed to be spatially constant throughout all
## geometries
```

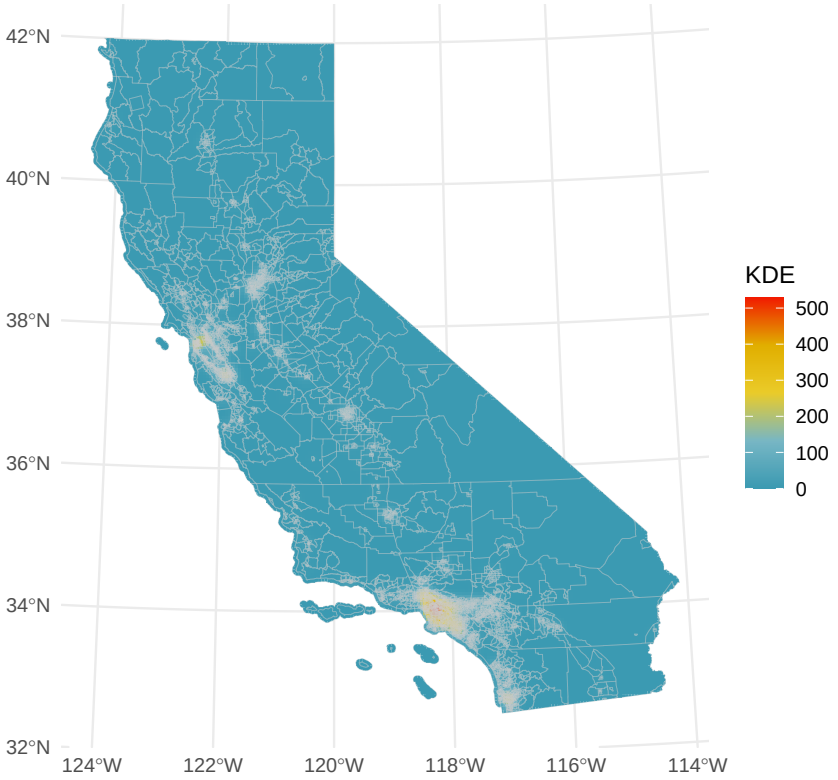

Figure 5: KDE Clustering of Census Tracts in California

Ripley's K

Ripley's K was performed using the `Kest` function in the `spatstat.core` package. This is really straightforward, with the most difficult aspect was having to convert an `sf-polygon`'s centroids into an `sp-SpatialPoints` object, and then to a `ppp-object`. While there are a few different functions to accomplish converting into a `ppp-object`, I found the function in the `maptools` package to be the most consistent and agreeable with `Rmarkdown` rendering.

In addition to Ripley's K, which tests clustering under a Complete Spatial Randomness (CSR) assumption, I thought it was appropriate to also run the Ripley's K derived for inhomogeneous point-patterns, as census tracts are obviously more densly clustered in population centers. An envelope was created for each Ripley's K based on a lengthy 500 simulations (thank goodness for `Rmarkdown`'s caching). As you can see in figure 5, the standard observed Ripley's K is obscenely higher than the theoretical K under the CSR assumption, indicating High amounts of clustering. The Inhomogeneous Ripley's K is much more revealing. Given that the California-Albers 3310 projection uses a scale in meters, I divided the x-axis to convert to Km. We can see that from 0 to about 125 Km, there is higher than expected clustering of census tracts. However, beyond 150Km, there is lower than expected clustering. This is understandable, as the size of census tracts tends to increase considerably in low-population areas, dispersing the point pattern.

```
pp <- as(st_centroid(eh), "Spatial")

ppp <- maptools::as.ppp.SpatialPoints(pp)

## Normal Ripley's K
K <- Kest(ppp, correction = "best", var.approx = TRUE)
K.e <- envelope(ppp, Kest, correction = "best",
                verbose = FALSE, nsim = 500)

## Ripley's K for Inhomogenous point patterns

Kinh <- Kinhom(ppp)
Kinh.e <- envelope(ppp, Kinhom, correction = "best",
                   verbose = FALSE, nsim = 500)
```

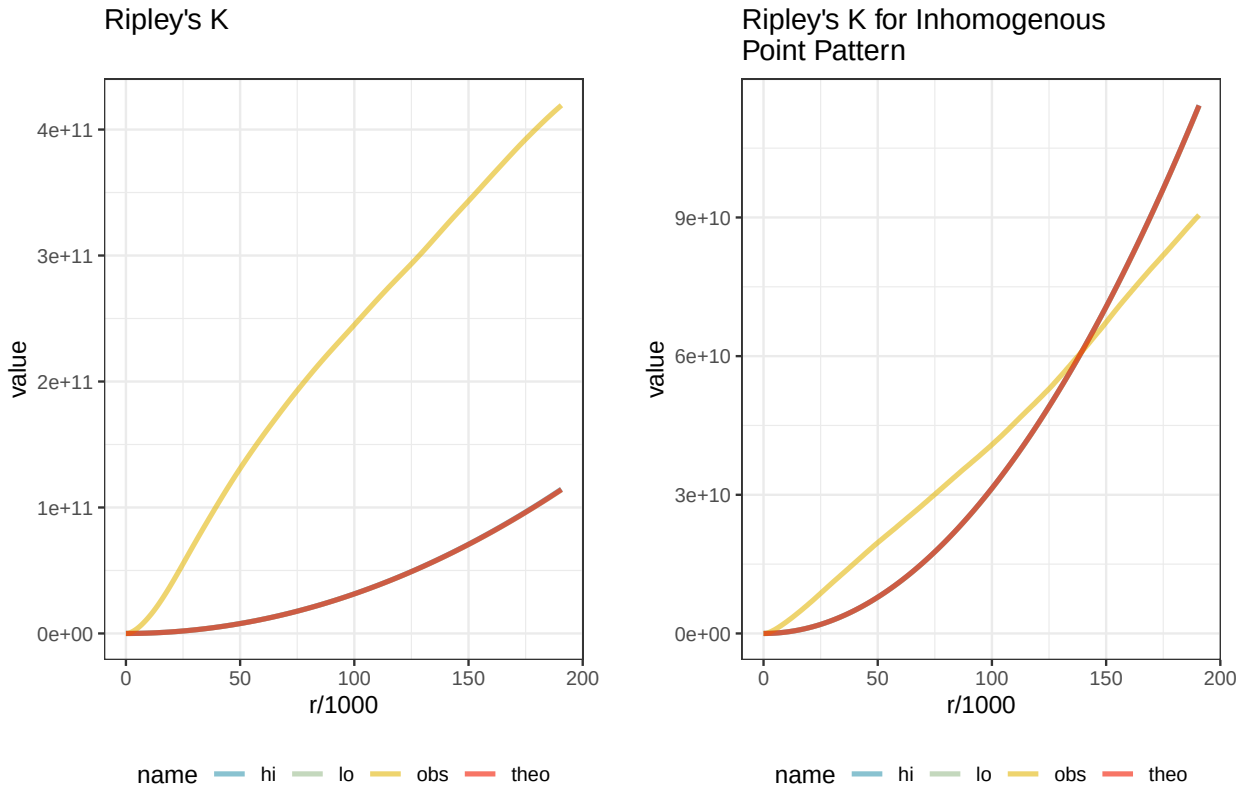

Figure 6: Plots of Ripley's K Functions

Appendix

Below are the pre-defined functions used throughout the assignment

```
# Colour Palette Function

pal <- function(x){
  p <- wesanderson::wes_palette("Zissou1", x, "continuous")
}


# A function to take an sf object and create a KDE raster
# (Not Used in this Assignment)

st_kde <- function(points,cellsize, bandwith, extent = NULL){
  require(MASS)
  require(raster)
  require(sf)
  if(is.null(extent)){
    extent_vec <- st_bbox(points)[c(1,3,2,4)]
  } else{
    extent_vec <- st_bbox(extent)[c(1,3,2,4)]
  }

  n_y <- ceiling((extent_vec[4]-extent_vec[3])/cellsize)
  n_x <- ceiling((extent_vec[2]-extent_vec[1])/cellsize)

  extent_vec[2] <- extent_vec[1]+(n_x*cellsize)-cellsize
  extent_vec[4] <- extent_vec[3]+(n_y*cellsize)-cellsize

  coords <- st_coordinates(points)
  matrix <- kde2d(coords[,1],coords[,2],
                  h = bandwith,n = c(n_x,n_y),
                  lims = extent_vec)
  raster(matrix)
}

## Function to create a data frame to manually Plot Moran's I

prepare_data <- function(data, x, listw){
  # prepare a dataframe with variables x and wx,
  # from the x and listw arguments
  # this dataframe will be the base data for the ggplot() call
  plot_data <- data %>%
```

```
    mutate(
      x = !!enquo(x),
      wx = lag.listw(listw, x, zero.policy = TRUE),
      label = as.character(attr(listw, "region.id"))
    )

  # Prepare other needed objects that don't fit into dataframe
  xwx.lm <- lm(plot_data$wx ~ plot_data$x)
  infl.xwx <- influence.measures(xwx.lm)

  # add non variables objects as attributes
  attr(plot_data,
       which = "is.inf") <- which(apply(infl.xwx$is.inf,
                                        1, any))
  attr(plot_data, which = 'xwx.lm') <- xwx.lm

  return(plot_data)
}
```