# Lab Exercise #4

This week, we will use some of the point pattern analysis tools available in the R package spatstat to investigate some point patterns of crime in St. Louis, Missouri.

You will need to add the spatstat and maptools packages. You should already have added spatstat. To add maptools, use the Packages - Install package(s)... menu option as before.

Please download Lab4Data folder and unzip it, which contain StLouisCrime2014.txt (crime data), city_limits_km.shapefile (a shapefile of city limit for which these crime data were collected), and StLouisCrime2014.gdb (various data layers relating to St. Louis).

First, open Arcmap and load the above data into Arcmap. If you don't have access to ArcMap, you can install one on your personal computer, please reach out to me for the installation package. Then you can use the license file previously received from MyESRI. You can also access the software via the Lobo Virtual Desktop. Please refer to the course syllabus for details. You may also access it from the classroom in Dane Smith.  Examine the data first. You can explore the point patterns in Arcmap first. Run a "*K**ernel Density***" analysis on crime_burglary layer, crime_hitandrun, and crime_disorderly layer separately. Set the search radius to 2km.

**Q1. Take a screenshot of your map. (10 points)**

**Q2. Run an "*Average Nearest Neighbor*" function for crime_burglary layer and describe the pattern. (10 points)**

**Q3. Run a "*Multi-Distance Spatial Cluster Analysis (Ripleys K Function)*" for crime_burglary layer. Take a screenshot of the result and describe the pattern. (15 points)**

Then close Arcmap and start R, change the directory to wherever you have put the city_limits_km shapefile and crime data file.

To get started, we need to first get the city limits (i.e., the study area) into R, so that it can be associated with the point data. Here's how:

```
>library(maptools)

> library(rgdal)

> S <- readOGR(dsn = ".", layer = "city_limits_km")

> SP <- as(S, "SpatialPolygons")

> W <- as(SP, "owin")
```

In order, this: loads up the maptools and rgdal package, reads the shapefile into data object S, converts S into a collection of polygons SP, and then converts SP into an 'owin' object, which is the format that spatstat requires so that the data can be used as an analysis window. You can plot W to see what you are dealing with:

```
> plot(W)
```

Next, read in the crime data:

```
> xy <- read.table("StLouisCrime2014.txt", header=T, sep="\t")
```

You can inspect the contents of xy, by typing xy, and see the names of this dataset's attributes by typing names(xy). Then convert it to a spatstat point pattern object, with the different crime types as an identifying mark:

```
> attach(xy)
```

```
> pp <- ppp(X, Y, window=W, marks=CRIME)
```

Remember that you will have to load the spatstat library using library(spatstat) before you can use any of its functions. Note that the attach() command above makes the various attributes of the raw data xy available for direct access by name. You can now make a map:

```
> plot(pp)
```

We are going to work with each crime as a distinct dataset, so it's convenient to split them permanently:

```
> gun <- pp[CRIME=="DISORDERLY"]
```

```
> rob <- pp[CRIME=="BURGLARY"]
```

```
> hit <- pp[CRIME=="HITANDRUN"]
```

And you can make maps of each individually like this:

```
> plot(density(gun))
```

```
> contour(density(gun), add=T)
```

```
> plot(gun, add=T)
```

Kernel density visualization is performed in spatstat using the density() function which we have already seen in action. The only additional piece of information you need to know is how to vary the bandwidth:

```
> plot(density(gun, 0.25))
```

The second parameter in the density function is the bandwidth. R's definition of bandwidth requires some care in its use. Because it is the standard deviation of a Gaussian (i.e., normal) kernel function, it is actually only around 1/2 of the radius across which events will be 'spread' by the kernel function. Remember that the spatial units we are using here are kilometers. It's probably best to add contours to a plot by storing the result of the density analysis:

```
> d250 <- density(gun, 0.25)
```

```
> plot(d250)
```

```
> contour(d250, add=T)
```

and you can also add the points themselves, if you wish:

```
> plot(gun, add=T)
```

R provides a function that will suggest an optimal bandwidth to use:

```
> r <- bw.diggle(gun)
> r
```

which will tell you the value it has calculated. You can then use this with d_opt <- density(gun, r). You may not feel that the optimal value is optimal. Or you may find it useful to consider what is 'optimal' about this setting.

**Q4. Create density maps (in R) of the gun homicide data, experimenting with different kernel density bandwidths. Provide a commentary discussing the most suitable bandwidth choice for this analysis visualization method. (15 points)**

The spatstat nearest neighbor function is nndist.ppp():

```
> nnd <- nndist.ppp(gun)
```

which returns a list of all the nearest neighbor distances in the pattern. You can plot these:

```
> hist(nnd)
```

and also summarize them:

```
> summary(nnd)
```

For a quick statistical assessment, you can also compare the mean value to that expected for an IRP/CSR pattern of the same intensity:

```
> mnnd <- mean(nnd)

> exp_nnd <- 0.5 / sqrt(gun$n / area.owin(W))

> print (mnnd / exp_nnd)
```

**Q5. Is this pattern clustered? Or evenly-spaced? (10 points)**

Like nearest neighbor distance analysis, quadrat analysis is a relatively limited method for the analysis of a point pattern, as has been discussed in the lecture.

However, it is easy to perform in R, and can provide useful insight into the distribution of events in a pattern. The functions you need in spatstat are quadratcount() and quadrat.test():

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

> q <- quadratcount(hit, 4, 8)

> plot(q)

> plot(hit, add=T)

> quadrat.test(hit, 4, 8)

> quadrat.test(hit, 4, 8, alternative="clustered")

> quadrat.test(hit, 4, 8, alternative="regular")
```

The second and third parameters supplied to these functions are the number of quadrats to create across the study area in the x (east-west) and y (north-south) directions. “alternative“ is the character string (partially matched) specifying the alternative hypothesis. The null hypothesis is that the data pattern is a realization of Complete Spatial Randomness.

**Q6. Take a screenshot of the plotted figure. Is it a clustered, regular, or random pattern? (10 points)**

The real workhorses of contemporary point pattern analysis are the distance-based functions: G, F, K (and its relative L) and the more recent pair correlation function.

Once again, spatstat provides full support for all of these, using the built-in functions, Gest(), Fest(), Kest(), Lest() and pcf(). In each case, the 'est' suffix refers to the fact the function is an estimate based on the empirical data. Calculation is straightforward:

```
> g_gun <- Gest(gun)
```

```
> plot(g_gun)
```

When you plot the functions, you will see that spatstat actually provides a number of different estimates of the function. Without getting into the details, the different estimates are based on various possible corrections that can be applied for edge effects.

To make a statistical assessment of any of these functions for our patterns, we need to compare the estimated functions to those we expect to see for IRP/CSR. Given the complexity involved, the easiest way to do this is to calculate the function for a set of simulated realizations of IRP/CSR in the same study area. This is done using the envelope() function:

```
> g_gun_env <- envelope(gun, Gest, nsim=99, nrank=1)
```

```
> plot(g_gun_env)
```

**Q7. Take a screen shot of the figure. What does the plot show us? (10 points)**

Also, you can change the rank setting. This will mean that the 'hi' and 'lo' lines in the plot will be placed at the corresponding low or high values in the range produced by the simulated realizations of IRP/CSR. So, for example:

```
> G_e <- envelope(hit, Gest, nsim=99, nrank=5)
```

will run 99 simulations of and place high and low limits on the envelope at the 5th highest and 5th lowest values in the set of simulated patterns.

**Q8. Take a screenshot of the figure. What does the plot show us? (10 points)**

Now run a K function using > K_e <- envelope(rob, Kest, nsim=19, nrank=1)

**Q9. Plot it and describe the pattern. (10 points)**

Your answers should be in a Word document with each answer numbered. The format for naming your labs is:

lastname_ lab#.doc      ie   Smith_ Lab4