Spatial Analysis and Data Exploration in History and Archaeology, Spring 2021
LDA-H313

WEEK 3 EXERCISE: Points Pattern Analysis & Linear Regression

This week we will continue examining points patterns, and look at linear regression analysis. First, please place the **week3** folder, which contains the data we will examine, in the **sade2021** folder and set your workspace. Make sure that the path is exactly as indicated, for example without a space in the folder name:

```
Windows:
      setwd("C:\\sade2021\\week3")
Mac:
      setwd("~/Documents/sade2021/week3")
```

Then load the following libraries. You may need to use **`install.packages()`** to download some of them first (we have not used GISTools before).

```
library(rgdal)
library(raster)
library(spatstat)
library(maptools)
library(GISTools)
```

1. Poisson Distribution & Complete Spatial Randomness

1.1 We will first look at Complete Spatial Randomness (CRS), a key concept in spatial statistics. First, lets create some example data. We'll set a study window with the first command, and fill it with 1000 randomly distributed points with the second.

```
window <- owin(c(0,10000), c(0,10000))

plot(runifpoint(n=1000, win=window))
```

Run the second command (**`plot`**) a couple of times. You'll see that everyone time you do, a new random distribution has been created.

1.2 You can also set the point of departure for programme's random number generator with the **`set.seed()`** command. If the seed is set at, say, 1, everyone who does this exercise should see the same pattern (unless there are differences in different versions of R). Try it:

```
set.seed(1)
plot(runifpoint(n=1000, win=window))
```

As you look at the various plots, it may appear that there is structure to them (such as clusters), but in reality the overall pattern is statistically random. Our brain's inherent tendency to look for patterns even when there is none - our pattern recognition skill - is something to bear in mind when looking at real-world examples of data. This is one important use of spatial statistics: examining, confirming or disproving the existence of patterns we suspect may exist.

1.3 Interesting and perhaps counterintuitively, spatial randomness has an expected statistical structure to it. This is a called the Poisson distribution and it describes Complete Spatial Randomness or CSR. We will now look at what this means with a further example. First, load and plot the study area, a spatial grid made up of 10 x 10 polygon cells.

```
grid <- readOGR(dsn="grid", layer="grid")
plot(grid)
```

Now lets recreate that random distribution from above, and plot it in the grid. You'll notice that naturally enough varying amounts of points fall into different cells.

```
set.seed(1)
randomp <- as.SpatialPoints.ppp(runifpoint(n=1000, win=grid))
crs(randomp) <- crs(proj4string(grid))
points(randomp, cex=0.5)
```

We can count the number of points that fall into each cell with the **`poly.count()`** command, and plot a histogram that shows how many cells have a certain number of points falling into them. We'll also calculate what the mean distribution is.

```
hist(poly.counts(randomp,grid), xlim=c(0, 25))
mean(poly.counts(randomp,grid))
```

As expected, the mean distribution is 10 (= 1000 points / 100 cells), but some cells contain more and some contain fewer points. This is also expected, since the pattern is **random** as opposed to, for example, **regular**!

While each individual point pattern will be different, the overall shape of the distribution is no surprise. It tends towards what is called a Poisson distribution, or the expected distribution given the parameters of the pattern, such as the expected

average of instances. We can create a curve describing an idealised distribution given the mean of 10 points per cell:

```
plot(dpois(x=0:20, lambda=10))
lines(dpois(x=0:20, lambda=10), col="blue")
```

Now lets do all the three plots side by side for comparison. The fit of the Poisson curve (at the right) is not necessarily perfect, but you can see how the histogram summarising our random pattern (middle) does resemble its shape.

```
dev.new(device=pdf, width=9, height=3)
par(mfrow=c(1,3))
plot(grid, main="Random point pattern")
points(randomp, cex=0.5)
hist(poly.counts(randomp,grid), xlim=c(0, 20), main="Histogram
of point distribution")
plot(dpois(x=0:20, lambda=10), main="Expected Poisson")
lines(dpois(x=0:20, lambda=10), col="blue")
```

Keep the blue line in mind. We will see idealised Poisson distributions in the K-Function, L-function and PCF plots further down. In short, because we can work out what the mathematical shape of an **idealised random distribution** is, we can compare **actual spatial data** (such as archaeological sites in a landscape) to it, and consequently work out whether patterns in the data are random or something else (clustered or regular).

2 Nearest Neighbour Revisited

2.1 Looking at the data

Next we will see how these lessons can be applied to historical settlement data from England. The locations of these sites were originally identified from maps created by the Ordnance Survey in Britain in the nineteenth century (*Ordnance Survey First Series*[1]) by B.K. Roberts and S. Wrathmell (*An Atlas of Rural Settlement in England*: 2000) and digitised by A. Lowerre (*Rural Settlements in England*: 2014). These locations represent *nucleated* settlements, which is to say communities living together in villages or towns rather than dispersed across the landscape in individual farms and houses.

[1] https://www.visionofbritain.org.uk/maps

To start, clear your desktop by closing all of your currently open plot windows, then load a polygon describing the historical borders of England, a Digital Elevation model and the settlement data, and plot them. The DEM may take a few moments to draw.

```
polyg <- readOGR(dsn="england", layer="england_historic")
dem <- raster("dem_england/dem_england_historic.tif")
settl <- readOGR(dsn="nucleations", layer="Nucleations")

plot(polyg)
plot(dem, add=T, col=terrain.colors(10))
points(settl, pch=19, cex=0.1)
```

2.2 Nearest Neighbour

Just like the markets data from last week, already at this large scale you can probably see how the topography (a first order effect) influences the point distribution. As a warm-up we'll run some of the same exercises we did last week to examine the data. First, convert the settlement data into a spastat object, setting our analytical windows to the England polygon. We will also need to remove a handful of points that fall outside this polygon window (using the first command).

```
settl <- settl[polyg, ]
sp_settl <- as.ppp(coordinates(settl),as.owin(polyg))
```

Let create a histogram (in a new window) showing the distribution of distances to the nearest neighbouring settlement (opening it in new window), and take the mean value.

```
dev.new(device=pdf, height=6, width=6)
hist(nndist(sp_settl), xlim=c(0,6000), breaks=100)
mean(nndist(sp_settl))
```

The mean is just over 2000 m, and you can see that a large proportion of English settlements are between 1 and 2 km of their nearest neighbouring settlement - a pretty dense distribution! Now do Clark and Evans test.

```
clarkevans.test(sp_settl, corrections="all")
```

R = 1.1709, and the p-value is well below 0.05. Does this mean the distribution of settlements tends, in average, towards being clustered or regular?

There are two problems with the Clark and Evans test. The first is that the **edge correction problem**: the edges of the study region may introduce a bias into the analysis; we'll return to this later below. The second is that it gives a single value for the whole point pattern. In reality, point patterns may have different properties at different scales. Take burial sites: at small scales individual graves may be regularly paced with reference to each other. But at a larger scale we may, however, see that graves belonging to the same kinship groups form clusters distinct from each other.

3 K-function, L-function and Pair Correlation Function (PCF)

We'll now look at a more sophisticated set of analysis that takes this potential variation into account. This is computationally more intensive, so we will restrict our analysis to two sub-regions within England.

3.1 Setting up

Load two polygons describing the new study areas.

```
south <- readOGR(dsn="southernengland", layer="eng_south")
mid <- readOGR(dsn="midlands", layer="midlands")
```

Then close the histogram window, so that the window with the map of England becomes the active one. (If you had closed that map window previously, just replot it now.) Now plot the study areas. We will be looking at region surrounding the river Avon catchment area in the south red), and the central Midlands (light blue).

```
plot(south, border="red", add=T)
plot(mid, border="cyan", add=T)
```

3.2 Study settlements

Now create two objects that contain the settlements falling into theses regions, and create spastat objects out of them. You can check that you have successfully done so by plotting the points in colour on the map.

```
south_settl <- settl[south, ]
sp_south_settl <-
as.ppp(coordinates(south_settl),as.owin(south))
points(sp_south_settl, pch=19, cex=0.1, col="red")

mid_settl <- settl[mid, ]
sp_mid_settl <- as.ppp(coordinates(mid_settl),as.owin(mid))
```

```
points(sp_mid_settl, pch=19, cex=0.1, col="cyan")
```

3.3 Now lets point the distributions side by side for a closer look. Give a moment for the DEM to draw.

```
dev.new(device=pdf, height=5, width=10)
par(mfrow=c(1,2), mai=c(1, 1, 1, 1))

plot(south, main="Avon region")
plot(dem, add=T, col=terrain.colors(20))
points(sp_south_settl, pch=19, cex=0.5)

plot(mid, main="Midlands")
plot(dem, add=T, col=terrain.colors(20))
points(sp_mid_settl, pch=19, cex=0.5)
```

3.4 You can see that there is a much clearer apparent pattern in the distribution of settlement in the River Avon region (or "south"), with many settlements organising as strings by the waterways. Lets run Nearest Neighbour tests and compare results

```
clarkevans.test(sp_south_settl, correction="none")
clarkevans.test(sp_mid_settl, correction="none")
```

Both gives results indicating a regular dispersion, with the Midlands producing a higher R-value than the southern region (1.328 vs 1.1343).

From the map we can see that the southern pattern indicates significant clustering (not regular dispersion!) at river valleys, so this result alone seems unsatisfactory. We must look at how interactions behave at different spatial scales.

3.5 K function

From your reading in week 2 (Chapter 4: Spatial Point Patterns and Processes) you will recall that the K-function uses a method similar to the Clark and Evans test, but is capable of expressing the result in a range values at different spatial scales. In other words, we can for example work out that a given pattern is clustered at some interval of distances between points, but spatially random at another. Lets see how this work in practice, taking the southern study area as our ccase study.

Run the K-function and plot it in a new window. We'll set the x-axis to show the first 5000 m in order or to investigate the start of the graph, but you can also set the limit higher to see how it develops at longer distances.

```
k_func_south <- Kest(sp_south_settl)
dev.new(device=pdf)
plot(k_func_south, xlim=c(0,5000), main="K-Function South
close-up")
```

You can see there are a number of different coloured lines drawn on the map, most of them overlapping. The “iso”, “trans” and “bord” lines (which in this graph are pretty much the same), are different mathematical solutions to the **edge correction problem**. There may be points (i.e. settlements) just outside the edge of the study a area, as we indeed know there are in the real world. The these are various mathematic corrections intended to reduce the bias. Type **?Kest** for more. Here we do not really need to worry about it, as all the offered solutions are identical, but you should bear this in mind if you apply this method to other data.

The last line “pois” is the expected **Pois**son distribution, if our point pattern conformed to Complete Spatial Randomness. This of it as the blue line from the start of this exercise. If the other lines overlay the Poisson line, then the pattern would be random. But when the other lines are below the Poisson line the pattern is regular, and when they are above the Poisson line the pattern is clustered. In other words, the settlement pattern in regular at distances of up to 2 km between sites, and clustered above this distance (or at least up to 3 km, when our chart ends).

This does make sense, since (if you remember from the histogram) the majority pf settlements are within 1-2 km of their most immediate neighbour. Larger settlement clusters would also start to emerge only as we zoom out beyond this range.

You can see how this gives more useful information than the basic Nearest Neighbour / Clark and Evans test, which gives only a single value to represent an entire spatial scale.

3.6 L-Function

The L-Function is the same as the K-Function, but the line for expected Poisson is straightened, and the other lines mathematically transformed to conform to this new shape. It can reveal more detail, and I find it often easier to read.

```
l_func_south <- Lest(sp_south_settl)
plot(l_func_south, xlim=c(0,5000))
```

3.7 PCF

The pair correlation function (PCF) uses the same principle as the K-function, but a different process. Whereas the K-function is cumulative, in that the mean nearest neighbour buffers continually increases including more points within it, the PCF uses annular or donut-shaped regions to calculate the points; each further region excluded the previous ranges.

It gives different, results and I find that for many different archaeological or historical data it can provide a better representation. Try it:

```
pc_func_south <- pcf(sp_south_settl)
plot(pc_func_south, xlim=c(0,5000))
```

The expected Poisson is now a horizontal line. The PCF function in fact indicates that while clustering peaks around 1-2 km, the pattern subsequently tends towards spatial randomness (the horizontal line). Perhaps this is a feature of the large of settlement being located in a line in river valleys. In fact, if you set the **xlim** value from 5000 to 20000 and replot, you can see there is a pretty choppy progression.

Lets do all the plots side by side for a comparison. What do you find easiest to read visually? If you apply this approach to different datasets, it might be a good idea to use both the cumulative (K & L-function) and annular (PCF) methods, and study the differences in the results.

```
dev.new(device=pdf, width=9, height=3)
par(mfrow=c(1,3))
plot(k_func_south, xlim=c(0,5000), main="K-Function")
plot(l_func_south, xlim=c(0,5000), main="L-Function")
plot(pc_func_south, xlim=c(0,5000), main="PCF")
```

4 Monte Carlo Simulation

A further and powerful refinement to these methods allows us to better distinguish between random and non-random patterns. Using principles of Monte Carlo simulation, and technique that has wide applicability in statistics (you will encounter it again in aoristic analysis), we can create a number of random point distributions in the study area. Think of it as running the exercise in secrtion **1.1** over and over again. We can then map the results of these distributions in a K, L, or pair correlation function graph to construct a probability envelope around the line marking the idealised Poisson distribution. When the line of the actual observed points (“obs”)

we are studying dips within this probability envelope, we can say that the point pattern is random at that spatial range.

4.1 To see what this looks like in practice, let us it using the PCF. The below command will calculate the observed PCF, and in addition will do that for 99 simulated CSR patterns. We can add them to the graph.

```
dev.new(device=pdf)
pc_func_100_south <- envelope(sp_south_settl, pcf, nsim=99)
plot(pc_func_100_south, xlim=c(0,20000), main="South: PCF with
99 MC Simulations")
```

You should now see a grey envelope surrounding the theoretical expected Poisson ("theo"). We can see that the pattern is still regular at low (<1000 m) ranges, and clustered at about 1000-3000 m, but after 3000 m for the observed points it mostly falls within the envelope. While within the envelope, the patterns can, in fact, be considered random even if it is a bit above or below the horizontal line.

4.2 A further refinement of the technique removes the uppermost and lowermost 2.5% of the simulations (they might include statistical outliers, for instance), giving us 95% of the random point patterns. This is quite similar to a standard 5% significance test. We'll need to create a larger batch of simulations for, which will take some time to compute, but this is required in actual scientific research (in fact, you could use much larger simulated datasets!).

```
pc_func_1000_south <- envelope(sp_south_settl, pcf, nsim=999,
nrank=25)
plot(pc_func_1000_south, xlim=c(0,20000), main="South: PCF
with 999 MC Simulations")
```

If you want, you could run the above for the K and L-functions too. Just replace the argument **`pcf`** in the **`envelope`** function with **`Kest`** or **`Lest`**. You may wish to restrict the graph to the first few kilometres.

4.3 As a comparison, run this exercise with the settlement data from the Midlands and plot it side-by-side with the southern data. Again, this will take a while to compute.

```
pc_func_1000_mid <- envelope(sp_mid_settl, pcf, nsim=999,
nrank=25)

dev.new(device=pdf, width=8, height=4)
par(mfrow=c(1,2))
```

```
plot(pc_func_1000_south, xlim=c(0,5000), main="South: PCF with
999 MC Simulations")
plot(pc_func_1000_mid, xlim=c(0,5000), main="Midlands: PCF
with 999 MC Simulations")
```

The Midlands data (right) the line of observed points exhibits weaker clustering, and a large scale. We can see it barely manages to get out of the probability envelope denoting a random pattern. It seems that the historical settlements in the Midlands did not strongly cohere into spatial clusters, which says something about the historical process that led to the formation of demographic landscape. Of course, we have only looked at the settlement data in aggregate. Perhaps we will start getting different results if we look at different settlement types (large and small towns, villages, hamlets) separately or in relation to each other.

You can close all active plot windows now.

5 Linear Regression

For this exercise, we will be looking at data explored by Fulford and Hodder in their 1974 article, which you can find in the *Further readings* folder. The script and data is adopted from examples written by Andy Bevan at the UCL. It uses regression analysis to study the distribution around southern England of Roman pottery manufactured at the site of modern Oxford.

5.1 Load and examine data

First, load the study data: two spatial points data frames: one that describe the location of Oxford, and another that describes sites where Oxford pottery has been found. Plot the sites against the England polygon.

```
sites <- readOGR(dsn="pottery_sites", layer="pottery_sites")
oxford <- readOGR(dsn="oxford/oxford.shp", layer="oxford")

plot(polyg)
points(oxford, pch=15, cex=2)
points(sites)
```

The Oxford object contains no interesting information (other than the location of Oxford), but inspect the sites data frame with:

```
head(sites, n=30)
```

You can see that alongside an ID column, there is the name of the site and the percentage of Oxford-manufactured pottery out of all recovered pottery (so 5.5% at Wroxester, for instance).

5.2 Calculating distances

We have consequently a **dependant variable**, the proportion of Oxford pottery at each sites. For a regression analysis we will need also an **independent variable**, which here will be distance of the site from Oxford. Since we have coordinates for both Oxford and the sites encoded within the spatial points, we can calculate the distances using the following commands and add them to the sites data frame as a new column.

```
coords <- rbind(coordinates(sites), coordinates(oxford))
```

The **rbind** function combines two data frames together by rows, so this creates a single list combining the coordinates of the sites with that of Oxford (as the last coordinate pair). (The **cbind** does the same, but by columns. Experiment with these later, as they are highly useful when you need to combine tables.)

```
d_matrix <- as.matrix(dist(coords))
```

This calculates a matrix that gives the distance of each point to each other point, rather like in a Nearest Neighbour test. Type the object name to see what this means.

```
sites$distox <- d_matrix[31,1:30]
```

And here we take the penultimate row of the matrix, which gives distance to Oxford, and add it as a new column to the original sites data frame. This is a handy workflow when you need to calculate distances to a place. Consider what applications in your other studies could you have for this.

5.3 Scatter plot

Now we create a simple scatter plot comparing distance to the proportion of Oxford pottery. We do here the same thing that we did in week 1, when comparing axe head size with weight.

```
plot(sites$distox,sites$oxpots)
```

5.4 Look at the X and Y axis, and the overall shape of the distribution? Do you think there is some structure to it? It might be hard to see, so lets carry out a linear regression analysis on it. Then we can plot the regression line on top of the scatterplot.

```
res <- lm(sites$oxpots ~ sites$distox)
abline(res, col="red")
```

This reveals that there is an overall correlation between nearness to Oxford, and the proportional amount of Oxford pottery found at a site (higher proportions nearer to Oxford, clearly).

We can study the data from the analysis:

```
summary(res)
```

There is a lot of information here, but first of all lets look at the "Multiple R-squared" values. Basically, this gives you the Pearson correlation coefficient squared. This will be number between 0 and 1. The closer to 1 this value is, the stronger the relationship between the two variables (i.e. here the distance and the proportion of pottery). 1 would indicate a perfect 100% correlation. We have only 35.59% correlation, not that high but not insignificant either. The second value ("Adjusted R-squared") adjust this value by the degree of freedom in the analysis, which can be significant in **multivariate regression** (more than one independent variable) analysis. This is not the case here, as we have just one independent variable (distance).

5.5 Taking the residuals into account

We have now carried out a basic linear regression analysis, and obtained $\mathbf{r^2}$, or the Pearson correlation coefficient squared, as a result. The low value and the poor fit of the line may mean we want to interrogate the date further. We can do this, for instance, by looking at the residual values. These are values designating the vertical distance between each point on the scatter plot and the regression line. You can see their summary statistics under residuals in the data you printed out using the **`summary()`** function above.

In order to work with them further, we will standardise the residual values by dividing the with the residual standard error, and adding the results again as a new column to **sites**.

```
sites$sr <- residuals(res) / summary(res)$sigma
```

The residual standard error is the sum of the residuals squared, and divided by the degree of freedom. The degree of freedom is (sample size) - (number of independent variables) -1, so here is it 28. You don't need to have a deep understanding of all the mathematics behind regression analysis in order to carry it out, so don't worry about it while you completing the exercise. But these basics are good to be aware of for future reference if you apply this in your further work.

Now lets take a look at the results as a boxplot and a scatter plot.

```
dev.new(device=pdf, width=8, height=4)
par(mfrow=c(1,2))
boxplot(sites$sr)
plot(sites$distox,sites$sr)
```

5.6 Mapping the residual values

The standardised residual values range between approximately -2 and 2, depending on whether the sites are above or below the regression line in the original scatterplot. This tells us **how much each site deviates from the expected result**; it has either less or more Oxford pottery than what might be expected in average given the overall regression analysis.

This is very useful information, since we can now how these distributions work out across England. Copy the below lines. Blue circles indicate sites there is less than the expected proportion of Oxford pottery, and red circles indicate those where there is more. Furthermore, the size of the circle indicates the size of the deviation. What kind of a pattern do you think they describe?

```
dev.new(device=pdf)
plot(polyg)
points(oxford, pch=15, cex=2)
points(sites[sites$sr >=-3 & sites$sr <=-2,], pch=1,
cex=3, col="blue")
points(sites[sites$sr >=-2 & sites$sr <=-1,], pch=1,
cex=2, col="blue")
points(sites[sites$sr >=-1 & sites$sr <=0,], pch=1,
cex=1, col="blue")
points(sites[sites$sr >=0 & sites$sr <=1,], pch=1,
cex=1, col="red")
points(sites[sites$sr >=1 & sites$sr <=2,], pch=1,
cex=2, col="red")
```

It seems that there is more Oxford pottery found at sites located at an east-west centred at Oxford than at a north-south one, and also near coasts. Fulford and Hodder argued that this showed the importance of waterways - especially the River Thames - in transporting bulk goods such as ceramics. Can you think of any other historical processes that could be studied using this approach?

Now save your workspace and data. You have finished with this week's exercise.