Geospatial Information Systems

Dr. Claire Dooley

(Slides adapted from Adam Dennett & Andy MacLachlan's previous years' lectures)

Outline

- The importance of patterns
- Patterns of categorical point data – Point Pattern Analysis
    - Quadrat Analysis
    - Ripley's K
    - DBSCAN
    - HDBSCAN
- Patterns of spatially referenced continuous observations
    - Spatial autocorrelation
    - Defining near and distant things
    - Measuring spatial autocorrelation
    - Moran's I
    - LISA (Local indicators of spatial association)

Part 1: Point Pattern Analysis

Questions we can ask / set

Points

Are these points distributed in a random way or is there some sort of pattern (uniform or clustered)?

Spatially continuous observations (e.g. values of polygons)

How (dis)similar are our values assigned to geographic units across geographic space

Quantifying Spatial Patterns

What is fixed?

Point Pattern Analysis

- Properties are fixed (e.g. binary - present or not)
- Discrete objects - present or not, binary, yes or no.
- Examples: fly tipping, stop and search, blue plaques, pharmacies

Properties fixed, but **space (location - x,y)** can vary

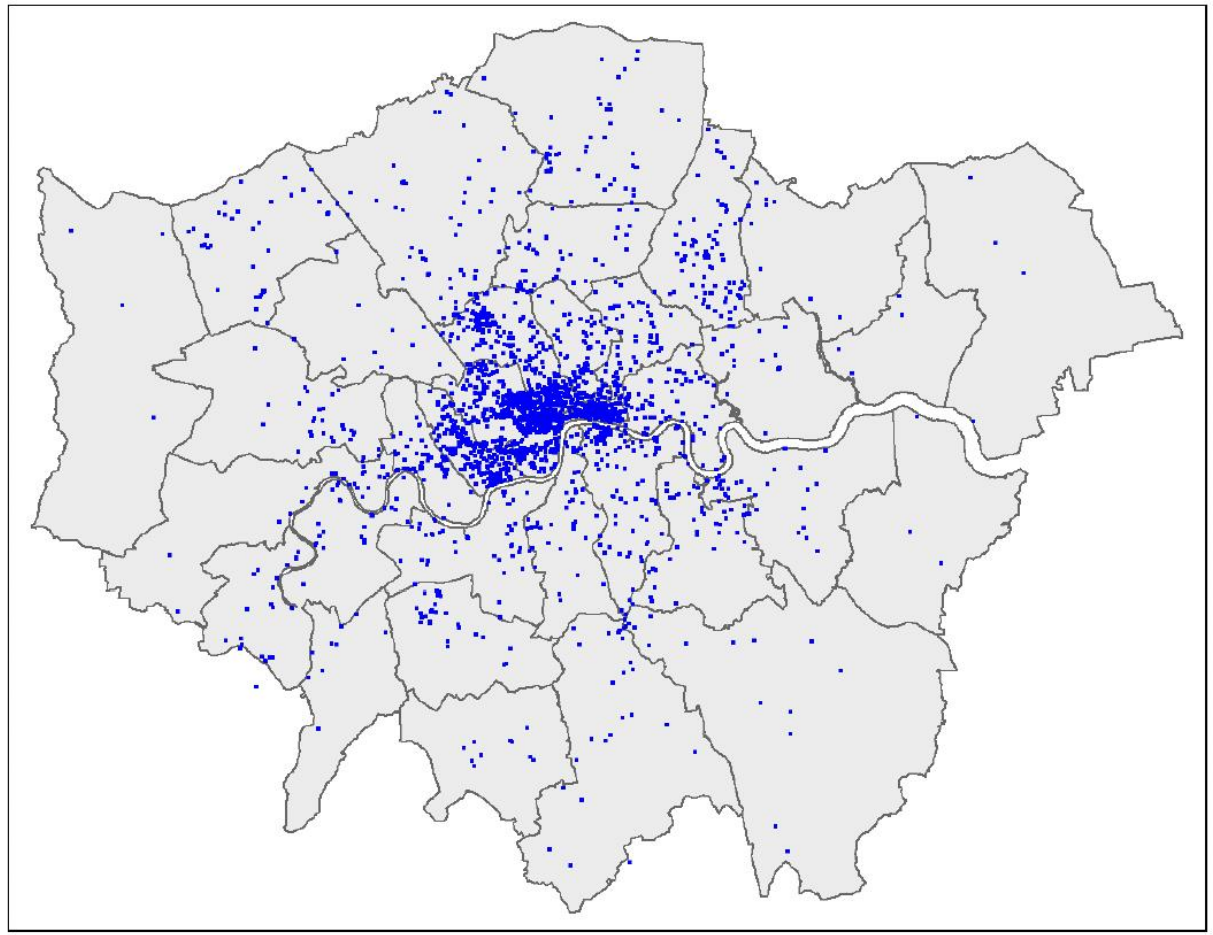
Spatial Autocorrelation

- Space (e.g. the location of the spatial units - wards, boroughs etc) is fixed
- The values of the spatial units vary
- Where the values are similar we say they exhibit Spatial Autocorrelation

Space is fixed, but **properties (values)** can vary

Examples

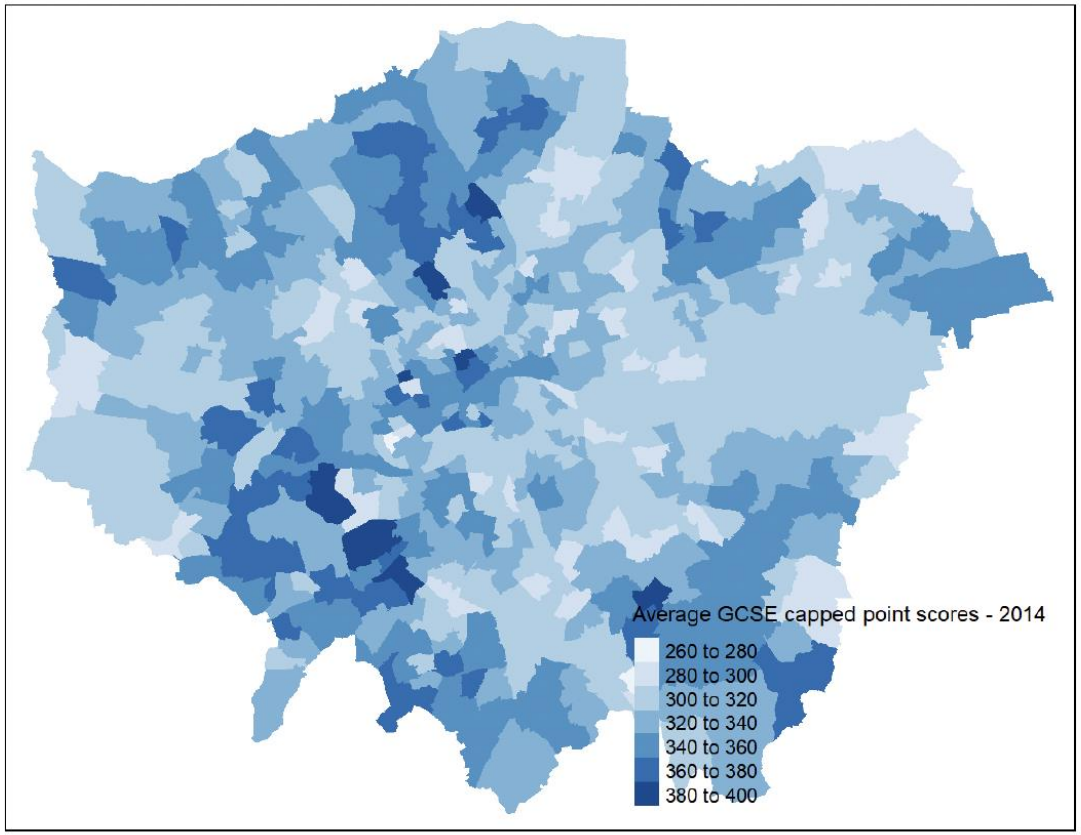
- Location of blue plaques in London
- Question: Are the points clustered or are they random?



Source:CASA0005

Examples

- Average GCSE point score 2014 per London ward
- Question: Are the values similar between certain wards?



Source:CASA0005

Now how do we calculate clustering and spatial autocorrelation...

Observed vs Expected

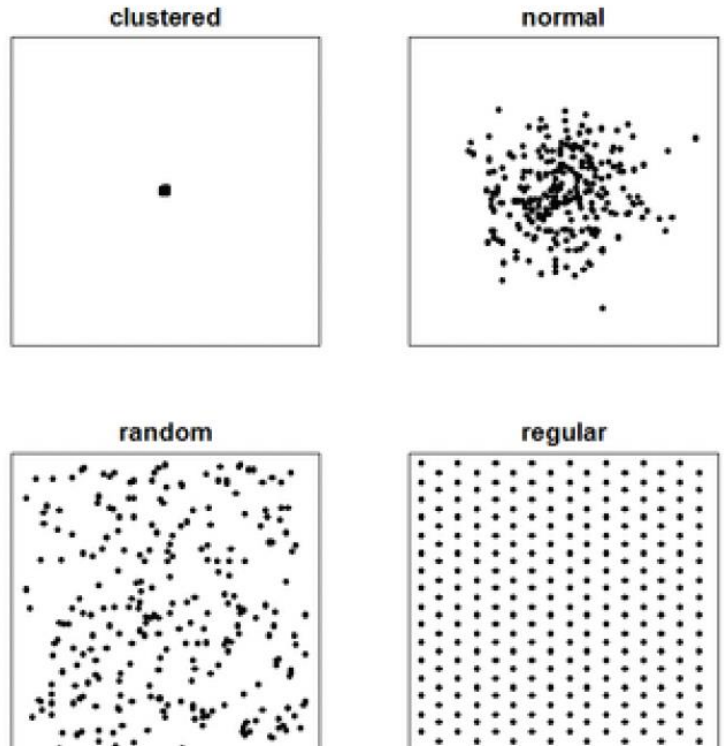
- Comparing what we observe in the real world against what we might expect is fundamental to most spatial (and other sorts of) analyses.
- If what we observe differs in some significant way from what we might expect, then there might be something interesting going on
- But how do we know what is expected?
    - We should expect randomness
    - Randomness conforms to known probability distributions

Point Pattern Anaysis

The core question..

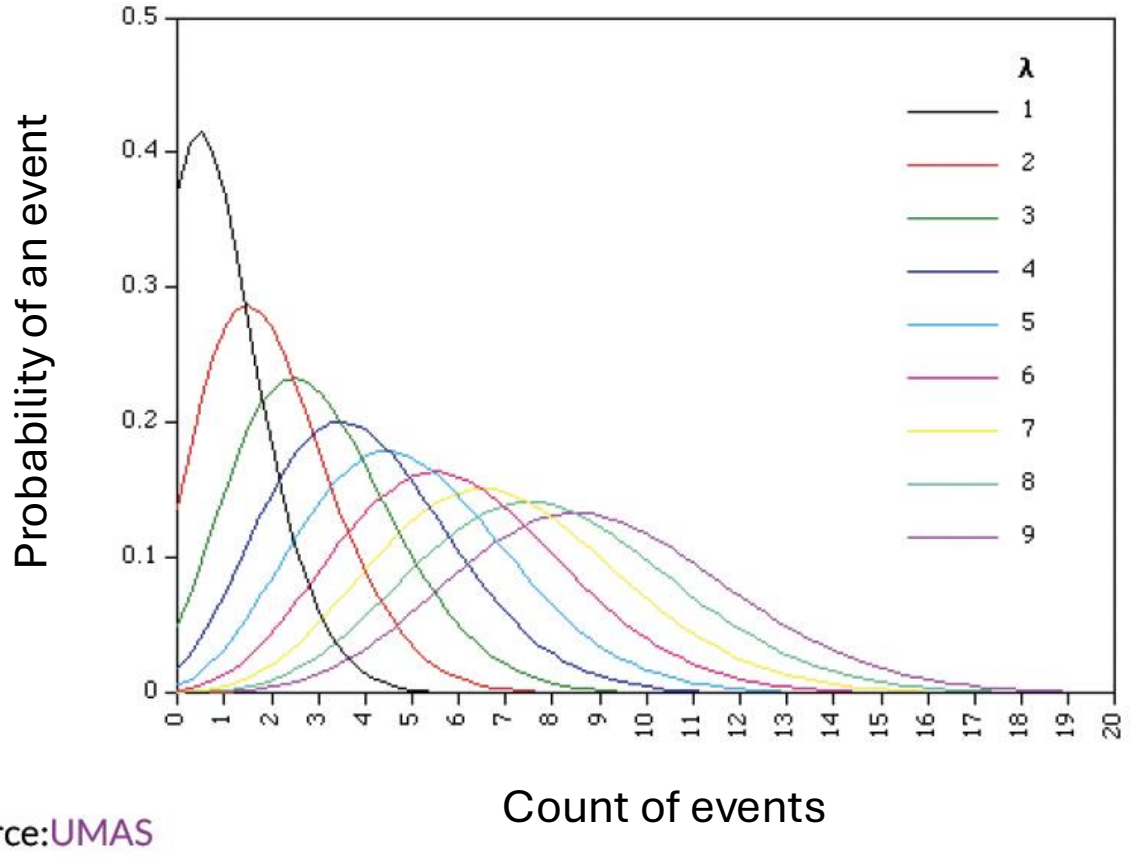
> Are these points distributed in a random way or is there some sort of pattern (uniform or clustered)?

- The **expected random model** is known as Complete Spatial Randomness (CSR)
- A random distribution of points is said to have a Poisson distribution
- By comparing the distribution of observed points with a CSR Poisson model, we can tell if we have an interesting point distribution....



The Poisson Distribution

- Describes the probability or rate of an event happening over a fixed interval of time or space
- Where the total number of events in a fixed unit is small (e.g. Breweries in a London Borough), then the probability of getting a low rate is higher
- As number of events increases, the mean (λ – lambda) increases and the probability distribution changes



Source:UMAS

The Poisson Distribution 2

https://andrewmaclachlan.github.io/CASA0005-lecture-6/#42

The Poisson Distribution 3

Rules / applies when

- The events are discrete and can be counted in integers
- Events are independent of each other
- The average number of events over space (or time) is known

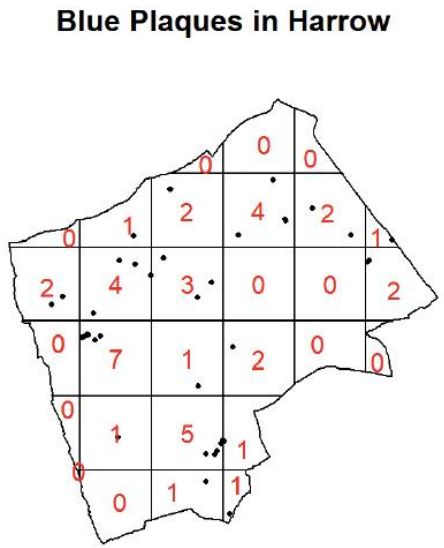
Use

- It's very useful in Point Pattern Analysis as it allows us to compare a random expected model to our observations
- Where our data **do not fit the Poisson model, then something interesting might be going on!**
- Our events might not be independent of each other – they might be clustered or dispersed and something might be causing this...

Testing for CSR - Point Pattern Analysis

Quadrat Analysis

- Developed and used frequently by ecologists
- Grid of squares
- Count number of incidents (burglaries, cholera deaths, hippos etc.) in each cell – store results in a table



Source:CASA0005

Compare the observed occurrences with a CSR Poisson model...

How do we apply this to spatial data ?

In code...

For point pattern analysis we need a point pattern (ppp) object...and an observation window...

```
window <- as.owin(Harrow)

BluePlaquesSub.ppp <- ppp(x=BluePlaquesSub@coords[,1],
                          y=BluePlaquesSub@coords[,2],
                          window=window)
```

The we can create a grid...

```
BluePlaquesSub.ppp %>%
  quadratcount(.,nx = 6, ny = 6)
```

Then pull out the results...

```
Qcount <- BluePlaquesSub.ppp %>%
  quadratcount(.,nx = 6, ny = 6) %>%
  as.data.frame() %>%
  dplyr::count(Var1=Freq)%>%
  dplyr::rename(Freqquadratcount=n)
```

Quadrat Analysis

- The (Poisson) probability (Pr) of an event (brewery in a quadrat square) is calculated

$$Pr = (X = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

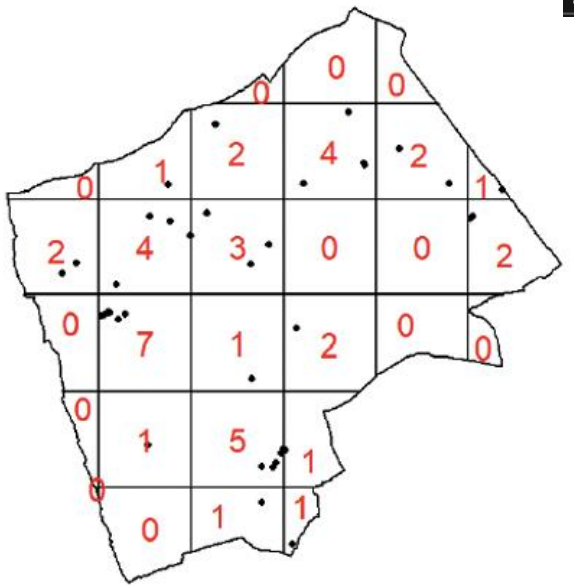
where:

- x is the number of occurrences
- λ is the mean number of occurrences
- e is a constant- 2.718
- $k!$ is the factorial of the number of occurences $((e.g. 4! = 1 * 2 * 3 * 4))$
- Here, remember $X = k$ - they are the same here.

Plugging in the numbers

	A	B	C	D	E	F	G
1	Plaques_per_cell	Freq_cells	Total_plaques_across_cells	Obs_probability	Lambda	Exp_probability	Exp_freq_cells
2	0	12	0	0.414	1.379	0.252	7
3	1	7	7	0.241		0.347	10
4	2	5	10	0.172		0.239	7
5	3	1	3	0.034		0.110	3
6	4	2	8	0.069		0.038	1
7	5	1	5	0.034		0.010	0
8	7	1	7	0.034		0.000	0
9		Total_cells	Total_plaques				
10		29	40				

Blue Plaques in Harrow



Plaques_per_cell = plaque count within a grid cell

Observed: orange columns

Freq_cells = freq of grid cells with each plaque count

Total_plaques_across_cells = Plaques_per_cell * Freq_cells

Obs_probability = Freq_cells / Total_cells

Lambda = mean

Expected: blue columns

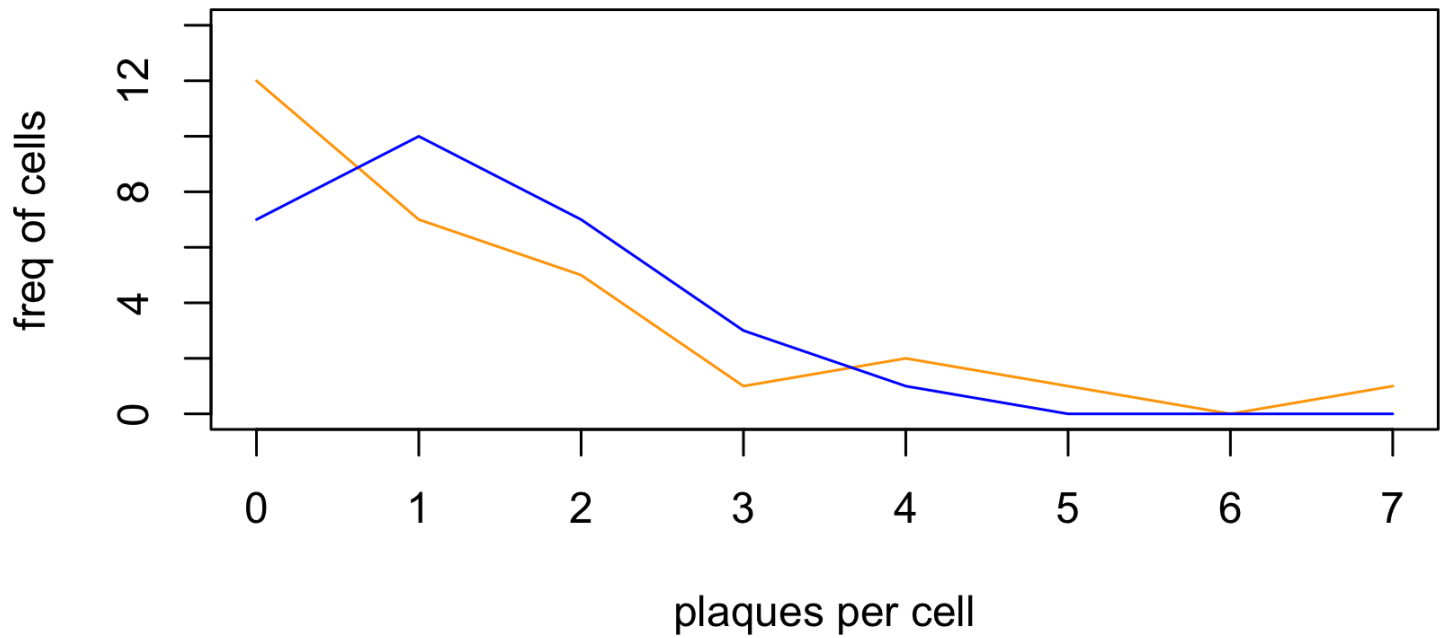
Exp_probability = plugging lambda & k into the Poisson equation

Exp_freq_cells = Exp_probability * Total_cells

$$\frac{\lambda^k e^{-\lambda}}{k!}$$

Confirming spatial randomness

- We would expect the probability distribution of X (blue plaques in London) to have a Poisson distribution if they exhibit Complete Spatial Randomness...we can look at our plot...



- We can test for CSR by comparing the observed and expected counts and using a test such as the chi-squared
- If our p-value is > 0.05 then this indicates that we have CSR and there is no pattern in our points.
- If it is < 0.05, this indicates that we do have clustering in our points.
- Here our p-value = 0.2594, implying complete spatial randomness

What are the issues with quadrat analysis ?

(small pause)

Quadrat size

Shape

Both MAUP

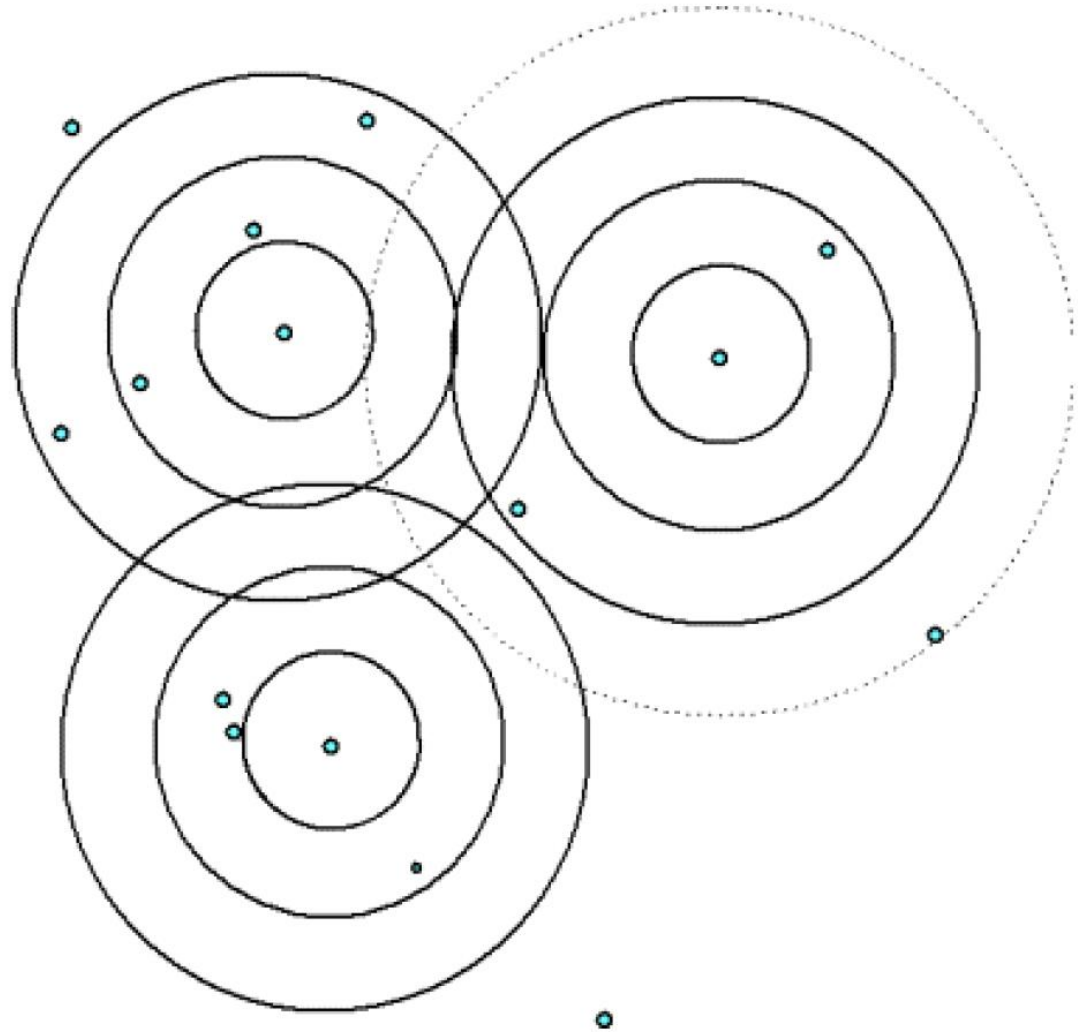
the question isn't that helpful...

Ripley's K

- To avoid scale and zoning problems associated with quadrat analysis, Ripley's K tests for CSR for circles of varying radii around each point

$$K(r) = \lambda^{-1} \sum i \sum j \frac{I(d_ij < r)}{n}$$

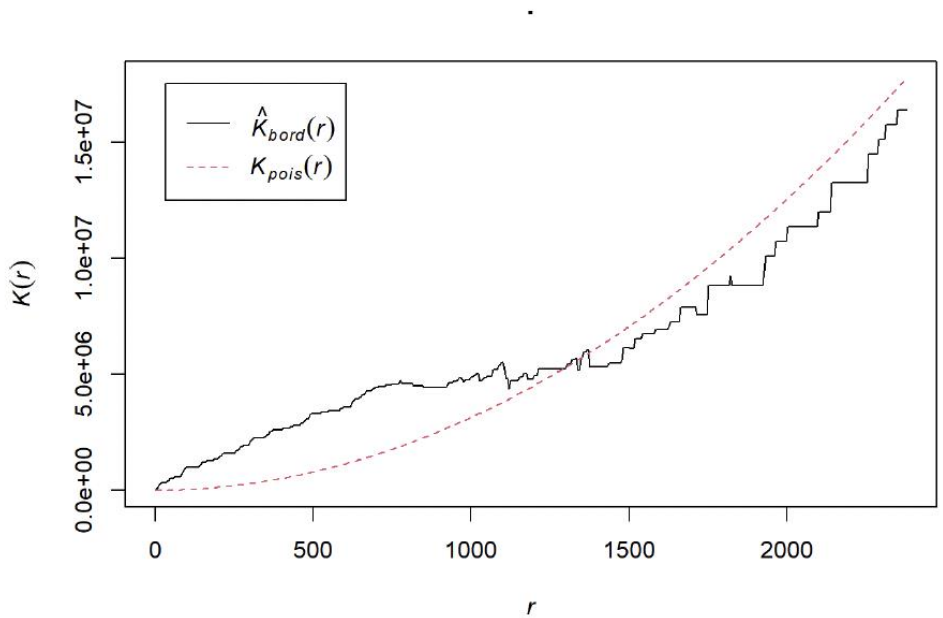
- In English: The average density of points for the entire study region (of all locations) $\lambda = (n / \Pi r^2))$
- Multiplied by the sum of the distances d_ij between all points within that search radius
- Divided by the total number of points, n
- I = 1 or 0 depending if $d_ij < r$



Ripley's K is simply...

```
K <- BluePlaquesSub.ppp %>%
  Kest(., correction="border") %>%
  plot()
```

- The correction specifies how points towards the edge are dealt with
- Border means that points towards the edge are ignored for the calculation but are included for the central points
- In R type `?kest` into the console for more info
- Red line is Poisson distribution (complete spatial randomness)



- Where the value of K falls above the line, the data appear to be clustered at that distance (up to 1300 metres).
- Where the value of K is below the line, the data are dispersed

- But what about planning constraints? Parks? Streets? Relative or absolute clustering?

?Kest

Kest {spatstat.explore} R Documentation

K-function

Description

Estimates Ripley's reduced second moment function $K(r)$ from a point pattern in a window of arbitrary shape.

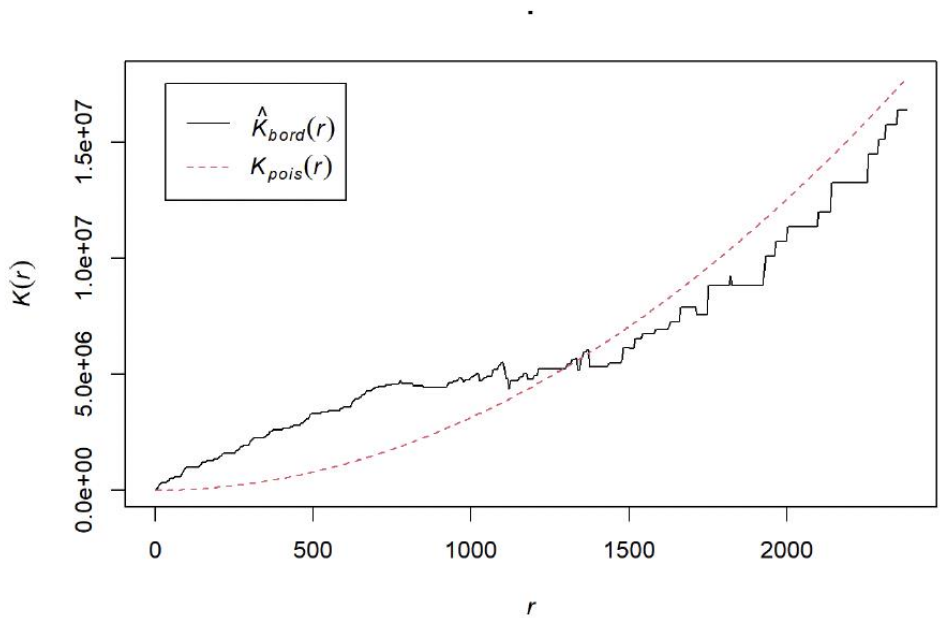
Usage

```
Kest(X, ..., r=NULL, rmax=NULL, breaks=NULL,
   correction=c("border", "isotropic", "Ripley", "translate"),
```

Ripley's K is simply...

```
K <- BluePlaquesSub.ppp %>%
  Kest(., correction="border") %>%
  plot()
```

- The correction specifies how points towards the edge are dealt with
- Border means that points towards the edge are ignored for the calculation but are included for the central points
- In R type `?kest` into the console for more info
- Red line is Poisson distribution (complete spatial randomness)



- Where the value of K falls above the line, the data appear to be clustered at that distance (up to 1300 metres).
- Where the value of K is below the line, the data are dispersed

- But what about planning constraints? Parks? Streets? Relative or absolute clustering?

What are the issues with Ripley's K ?

(small pause)

Computationally intensive when there are lots of points to consider

Extent of the study area can affect the calculation... eventually the radius around any point will include all other points in the study area

Phenomenon being studied cannot just occur anywhere (e.g. a river) - it's not network distance but Euclidean

We need clustering relative to other points (e.g. hotels, schools, dentists) in our study area?

We still don't know where the clusters are ?

Density-based spatial clustering of applications with noise (DBSCAN)

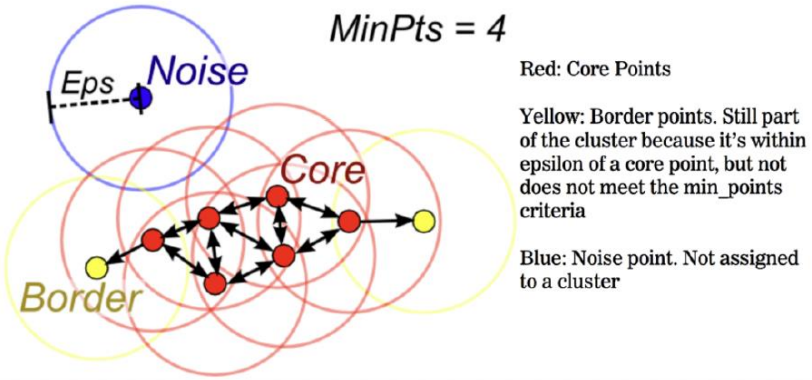
Popular because can detect non-linear clusters

2 parameters:

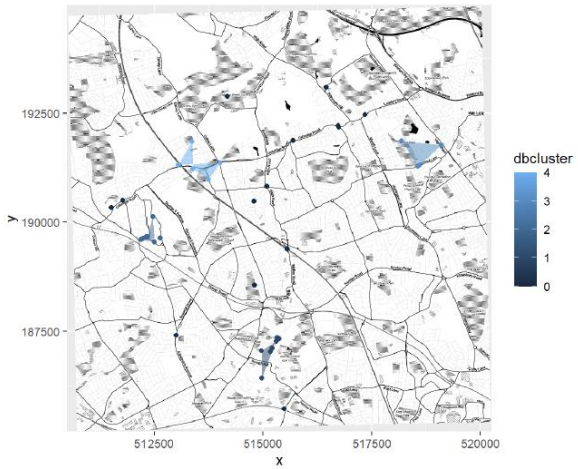
- Epsilon (eps) = size of neighbourhood within which to search for other points
- Minimum number of points to search for (MinPts)

If a point has >= MinPts in neighbourhood then defined as 'Core'

If point in neighbourhood of a core point but has <MinPts in its own neighbourhood, then defined as 'Border'



DBSCAN. Source:Medium



Source:CASA0005

DBSCAN

In R DBCAN is simply...

```
db <- BluePlaquesSubPoints %>%
  fpc::dbscan(.,eps = 700, MinPts = 4)
```

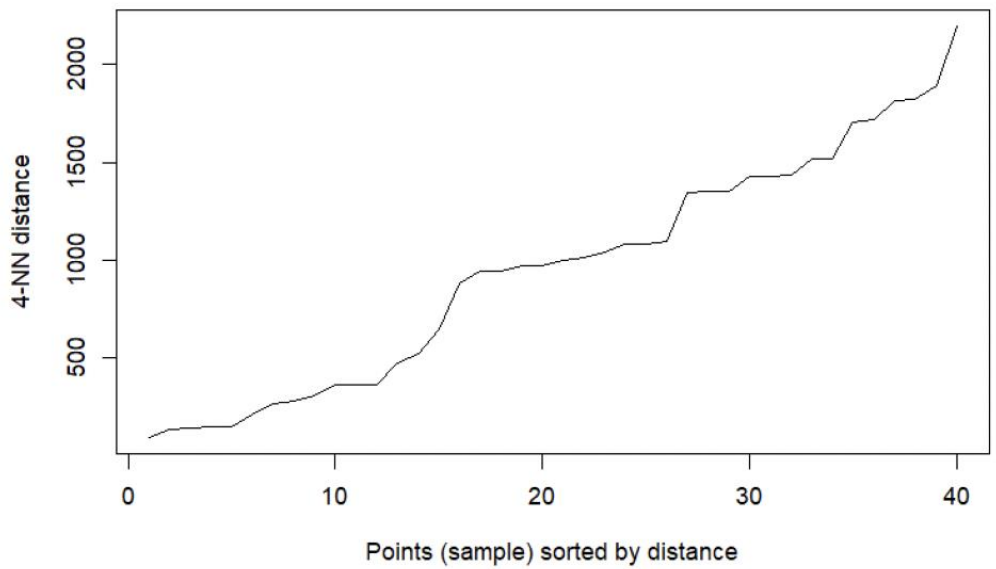
How to select values

- Eps: use the Ripley's K buldge - where the line shows the greatest difference to the theoretical value from a few slides ago...around 700 meters
- Or plot the **average distance of each point to k neighbours**, which are then plotted in ascending (low to high) order.
    - The knee is where this value (of distance to neighbours) increases and the value we use.

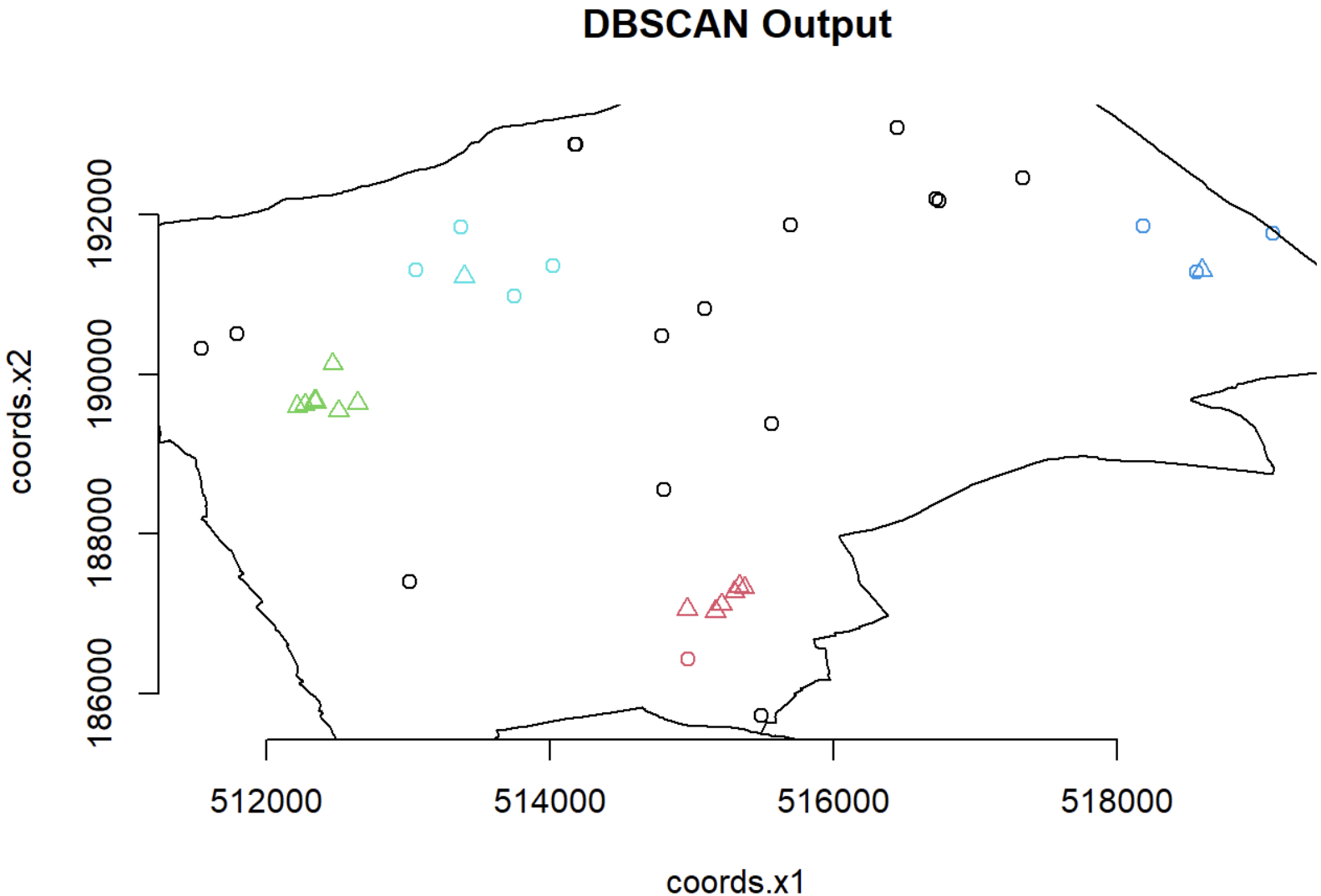
```
BluePlaquesSubPoints%>%
  dbscan::kNNdistplot(.,k=4)
```

DBSCAN

...this gives...



- plotted: average distance to the k neighbours, which are then plotted in ascending order
    - The knee is where this value (of distance to neighbours) increases and the value we use
- Min points: start with what you think is appropriate
- OPTICS - extends DBSCAN and uses an algorithm to find optimum distance thresholds for clustering.


DBSCAN Output
coords.x2
192000
190000
188000
186000
512000
514000
516000
518000
coords.x1

HDBSCAN - not in practical

- Hierarchical Density-based Spatial Clustering of Applications with Noise

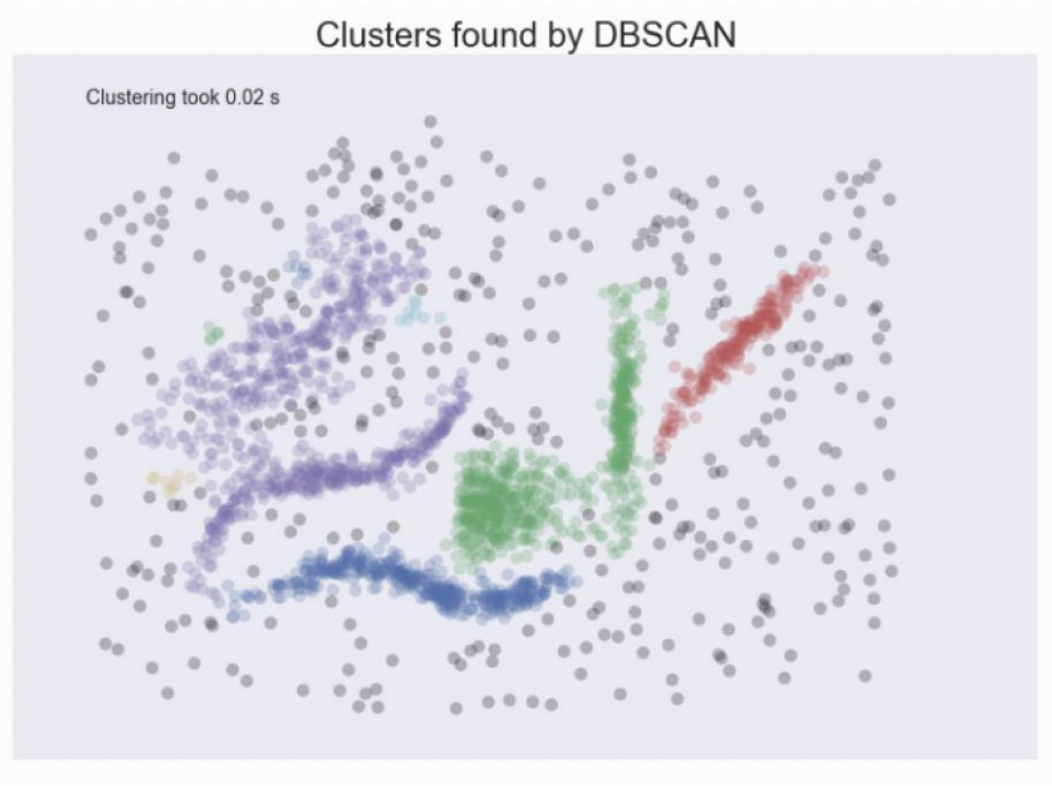
Best when:

- Arbitrary shapes - not all circular
- Clusters of different sizes - not constrained by 1 argument (e.g. distance or K neighbours)
- Clusters with different densities
- Noise / outliers

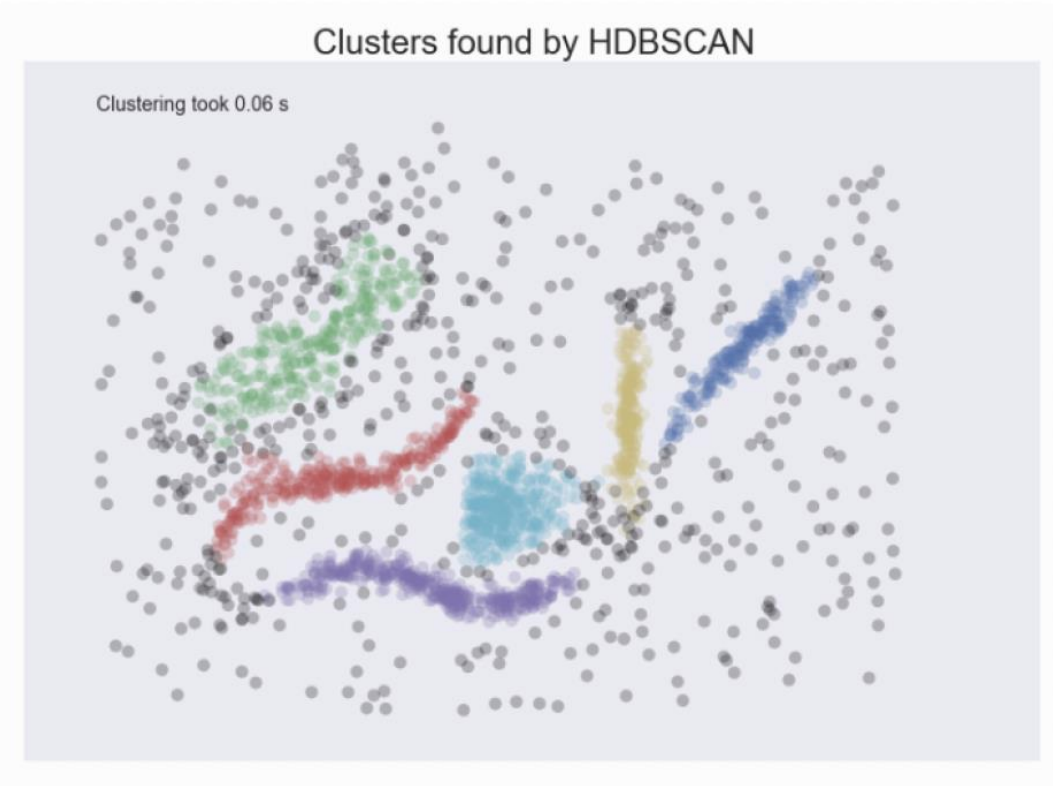
HDBSCAN. Source:Pepe Berba

HDBSCAN...differences...

- From the same authors that created DBSCAN
- it is the same but...**there is no epsilon (distance requirement)** and it can **VARY**



Source:Comparing Python Clustering Algorithms



Source:Comparing Python Clustering Algorithms

The core question..

> Are these points distributed in a random way or is there some sort of pattern (uniform or clustered)?