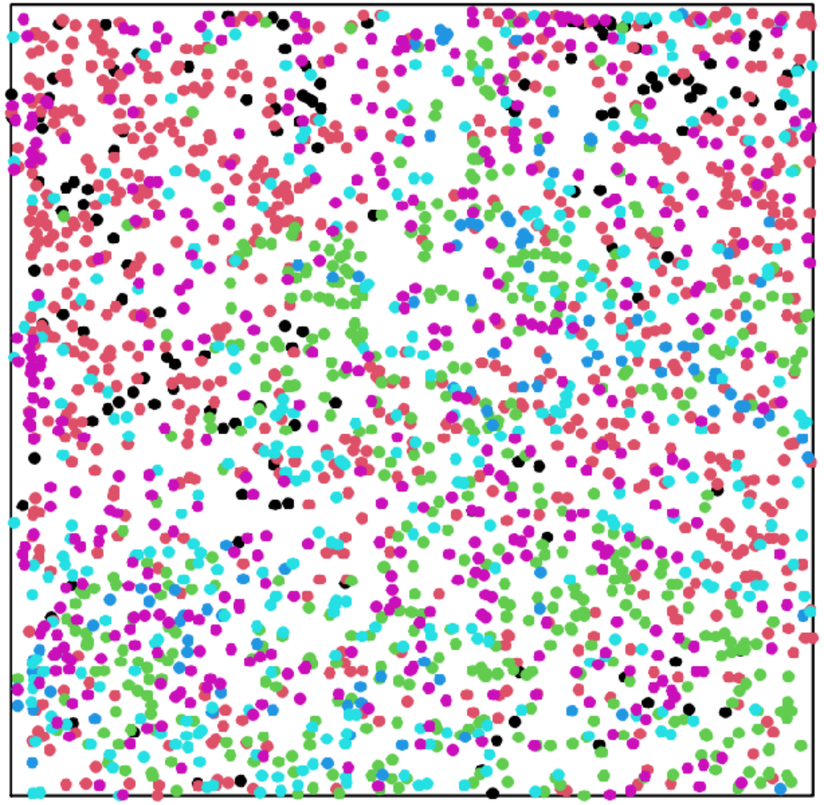
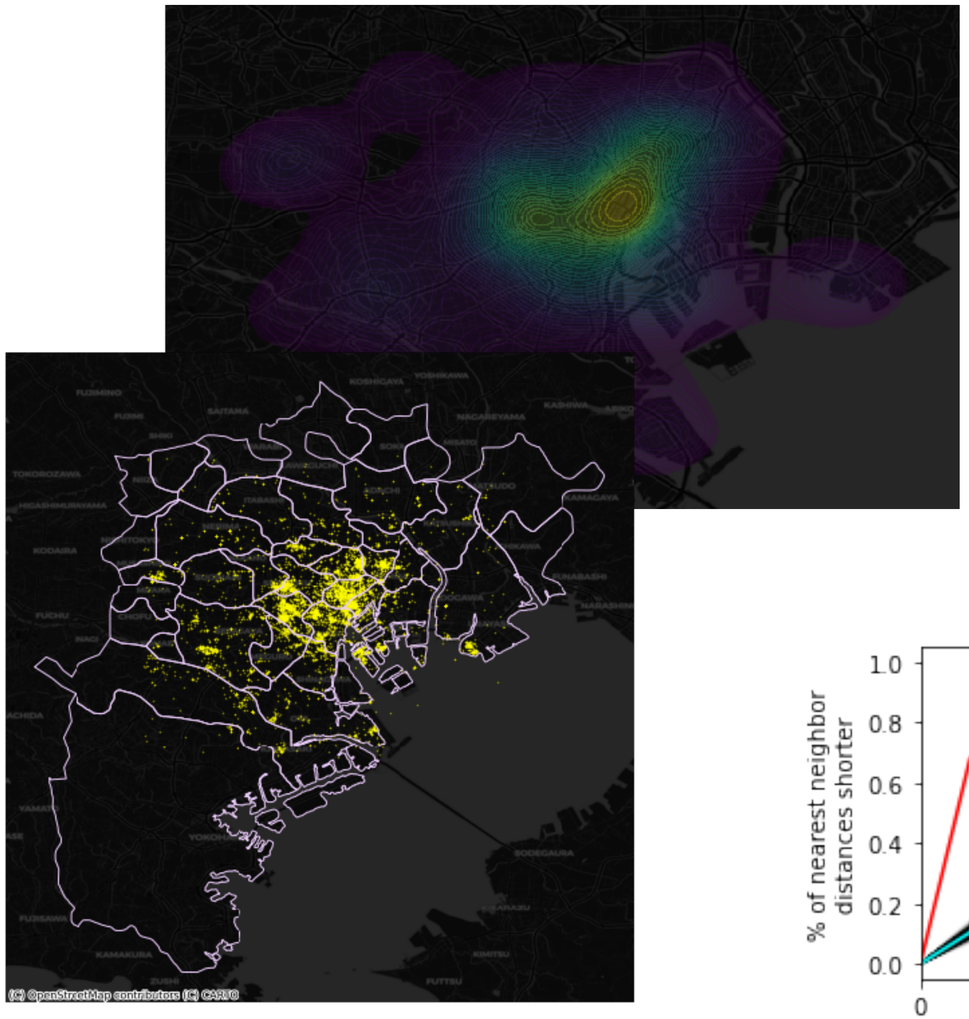
Lecture 7: Point pattern analysis

Instructor: Michael Szell
Mar 17, 2022

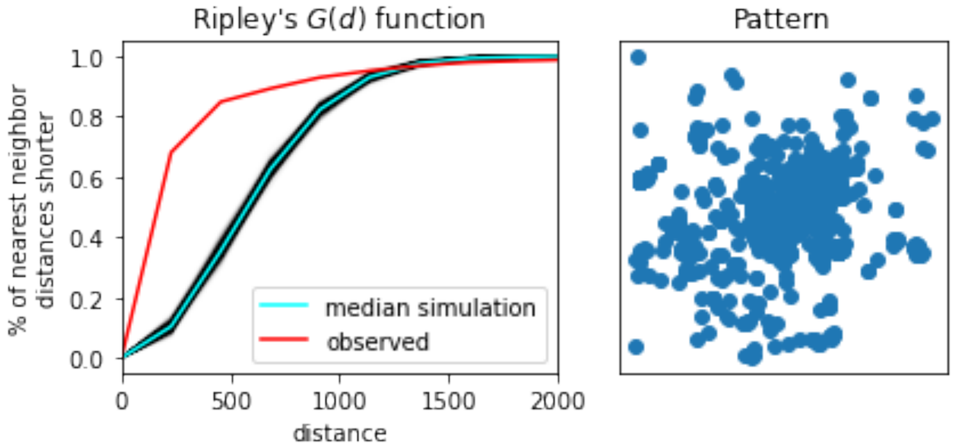


Today you will learn about point pattern analysis

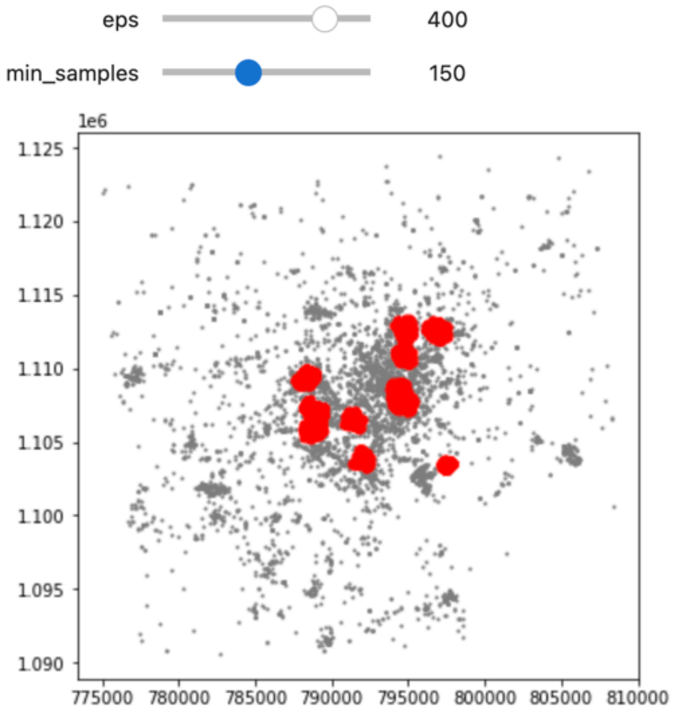
Visual analysis and aggregation



Ripley's functions



Clustering

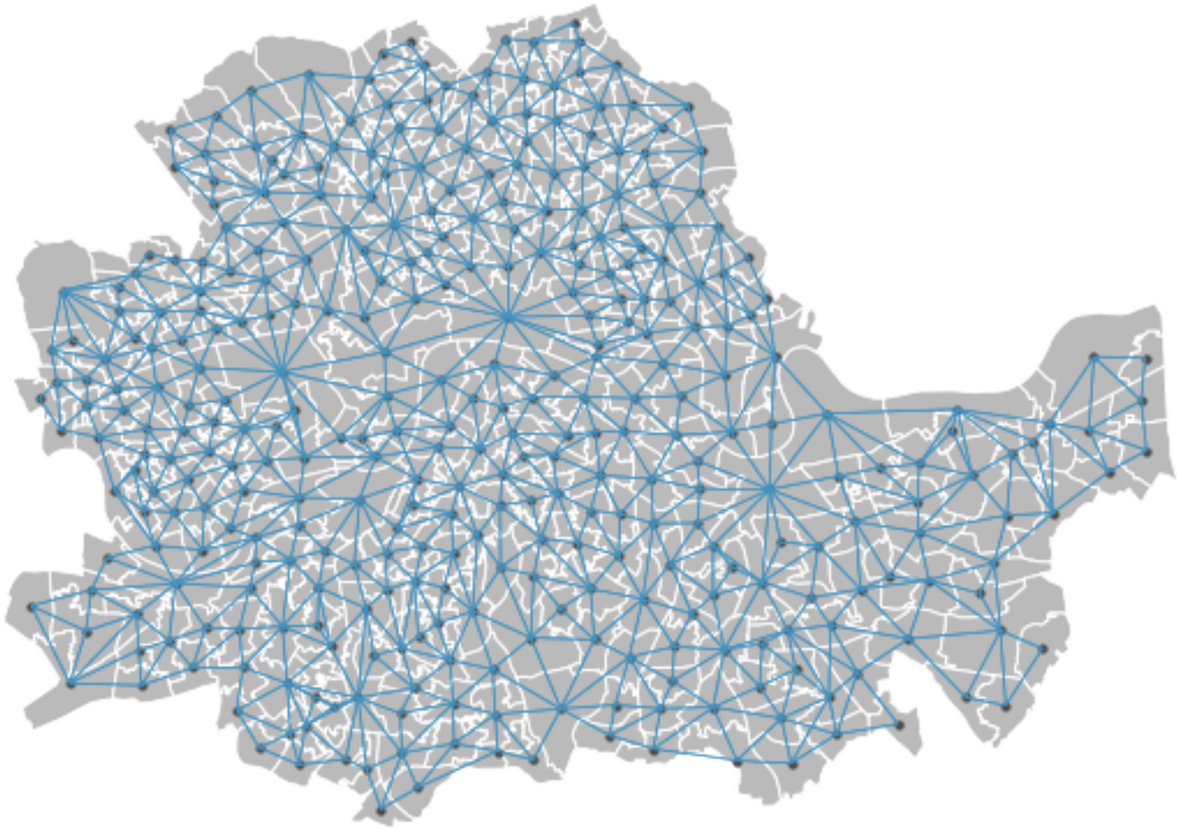


Points as static objects

When location of points is fixed:

Do same analysis as with polygons

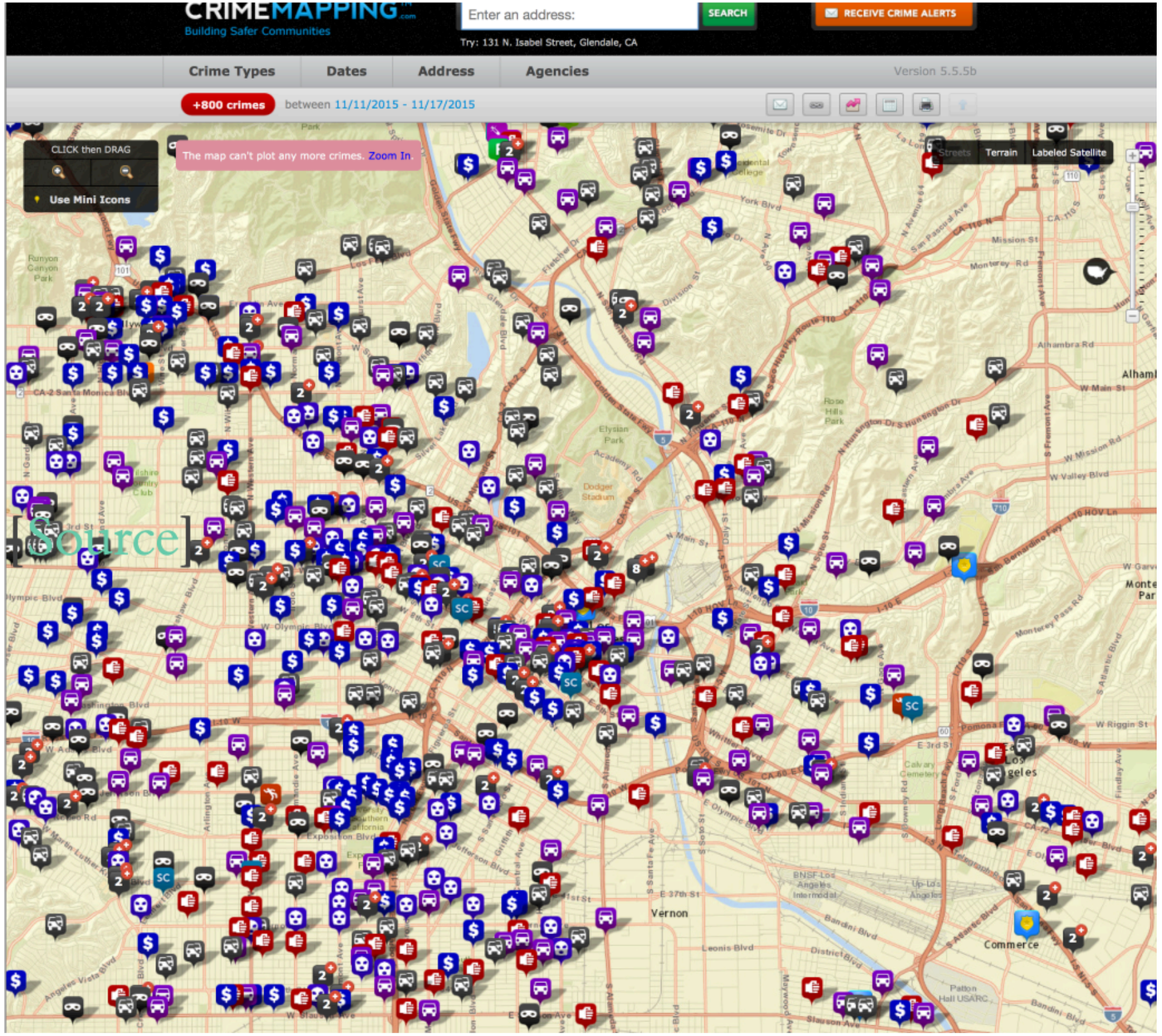
Static objects with fixed location:
cities, regions, buildings



Points as events

When points stand for events that could happen anywhere:

We try to understand the locations



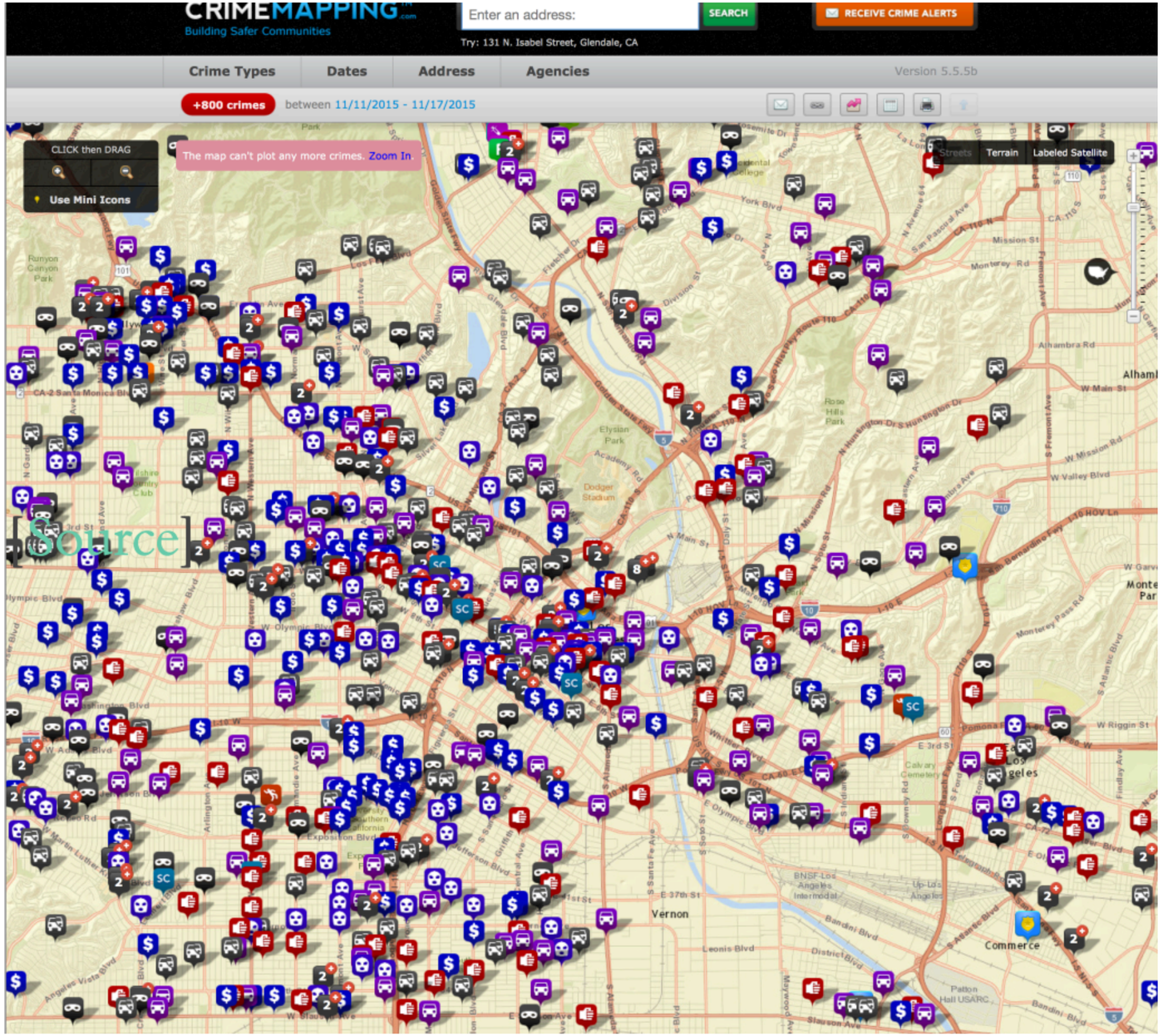
Points as events

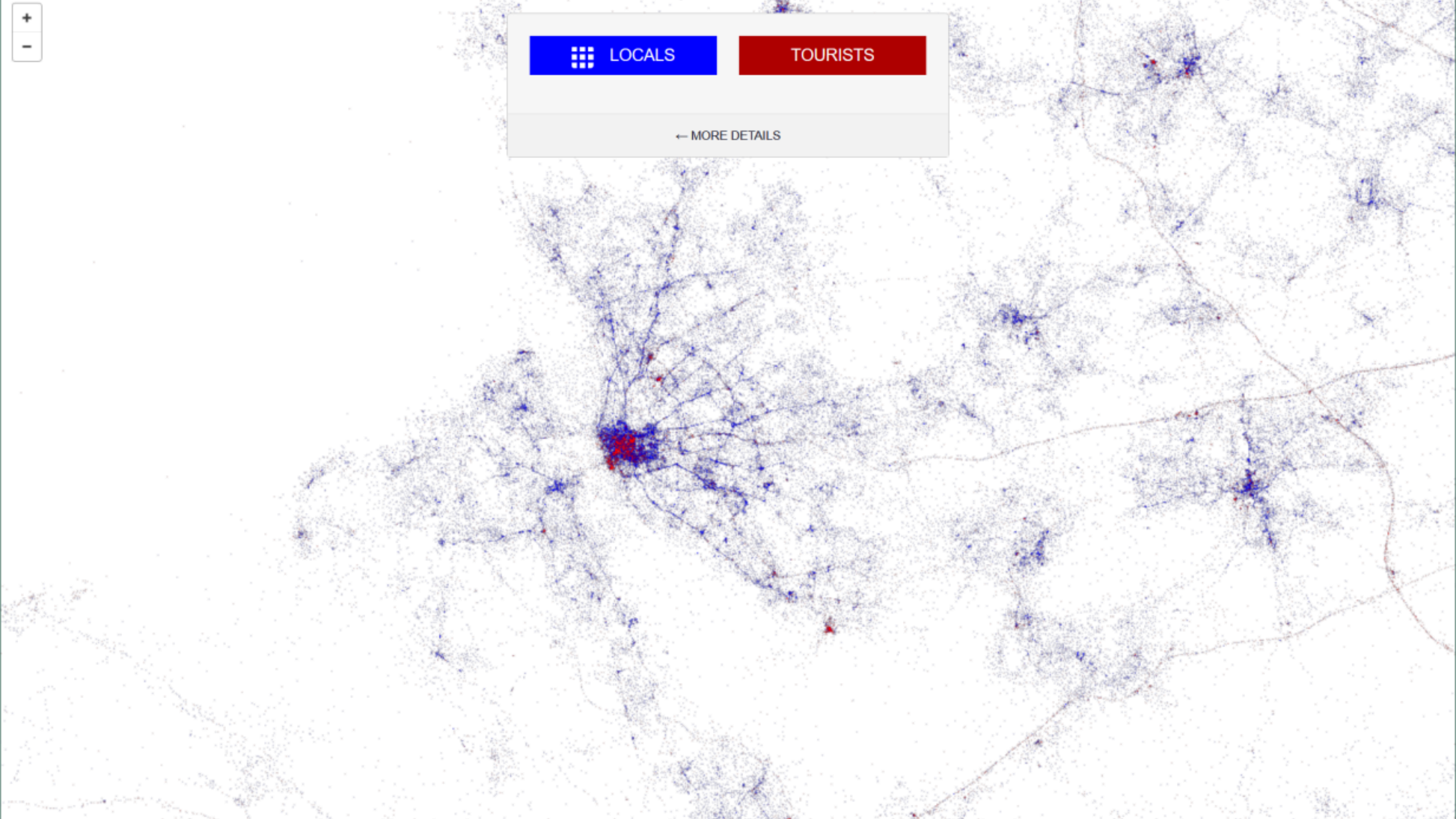
When points stand for events that could happen anywhere:

We try to understand the locations

We want to characterize the spatial pattern of the points

crime, trees, taxi pickups, tweets



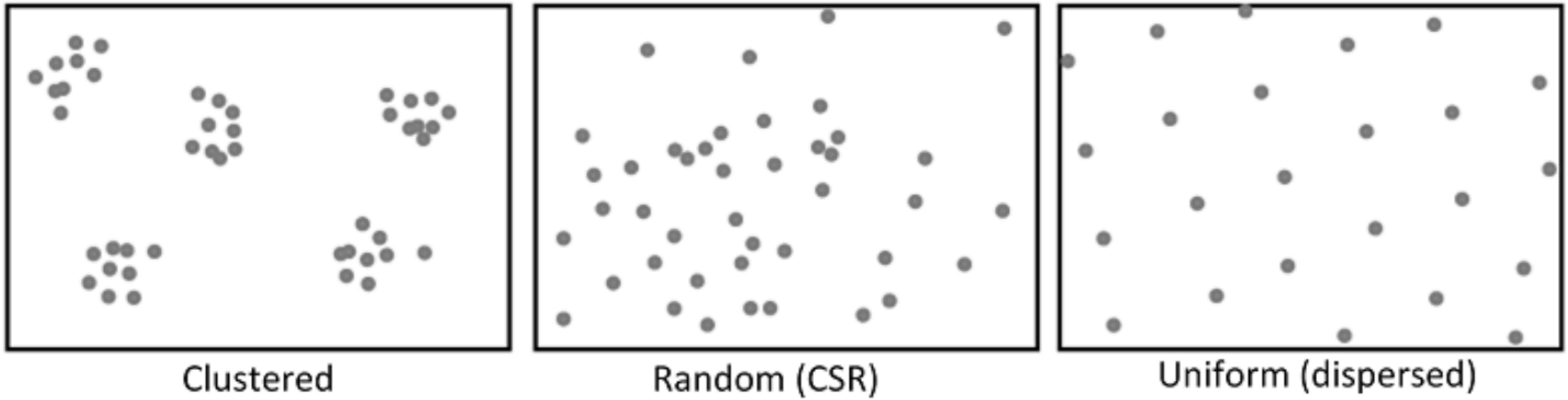

+
−
LOCALS
TOURISTS
← MORE DETAILS


150 Mio taxi trips in NYC in 2011 (HubCab)

Point patterns

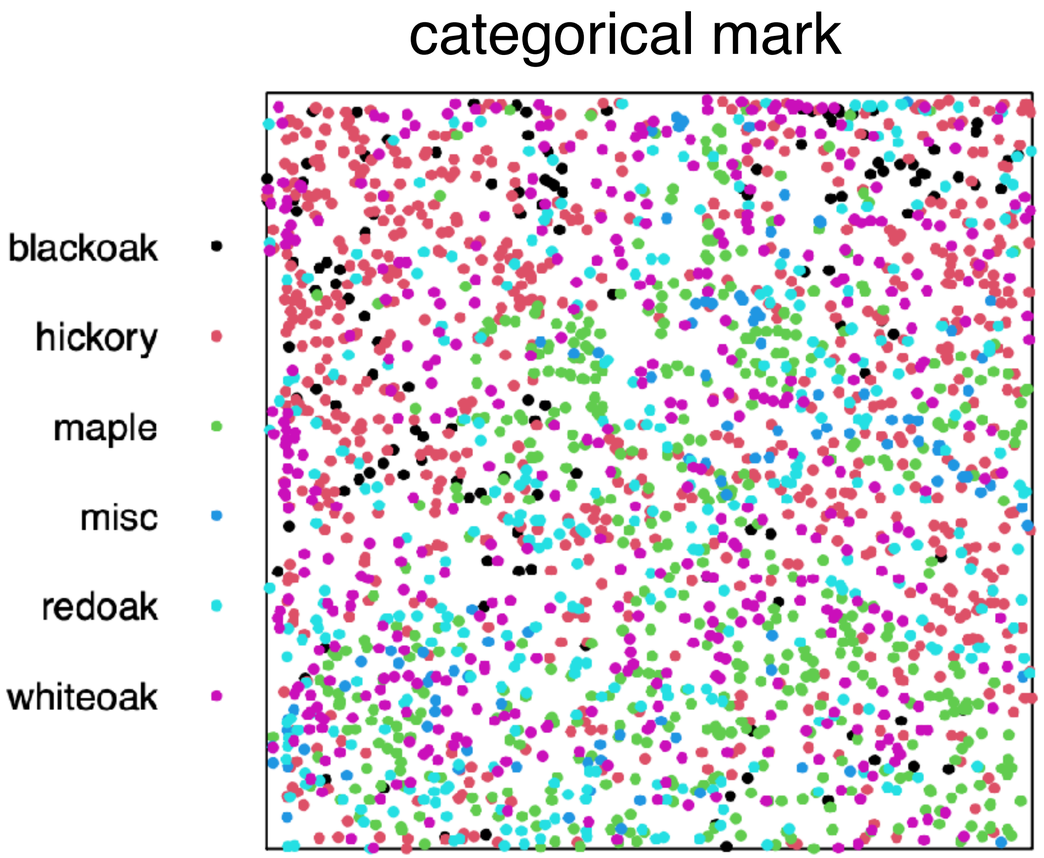
A point pattern is a distribution of points over space

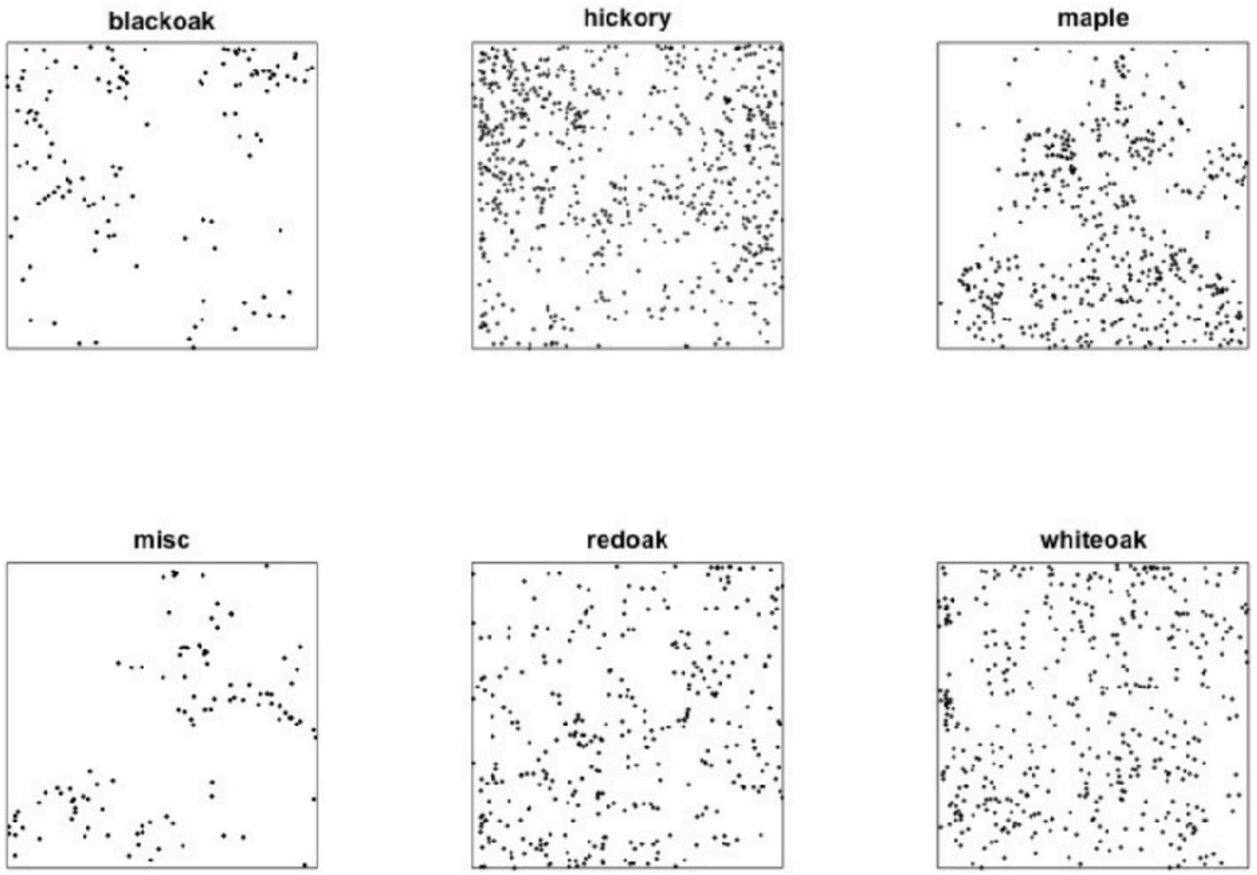
These points could happen anywhere, but are only observed in certain positions



https://gistbok.ucgis.org/bok-topics/point-pattern-analysis

Point patterns: marked vs unmarked





Point pattern analysis

aims to describe and explain the data generating process through:

1) visualization
2) modeling the underlying process
3) clustering algorithms

Visualization

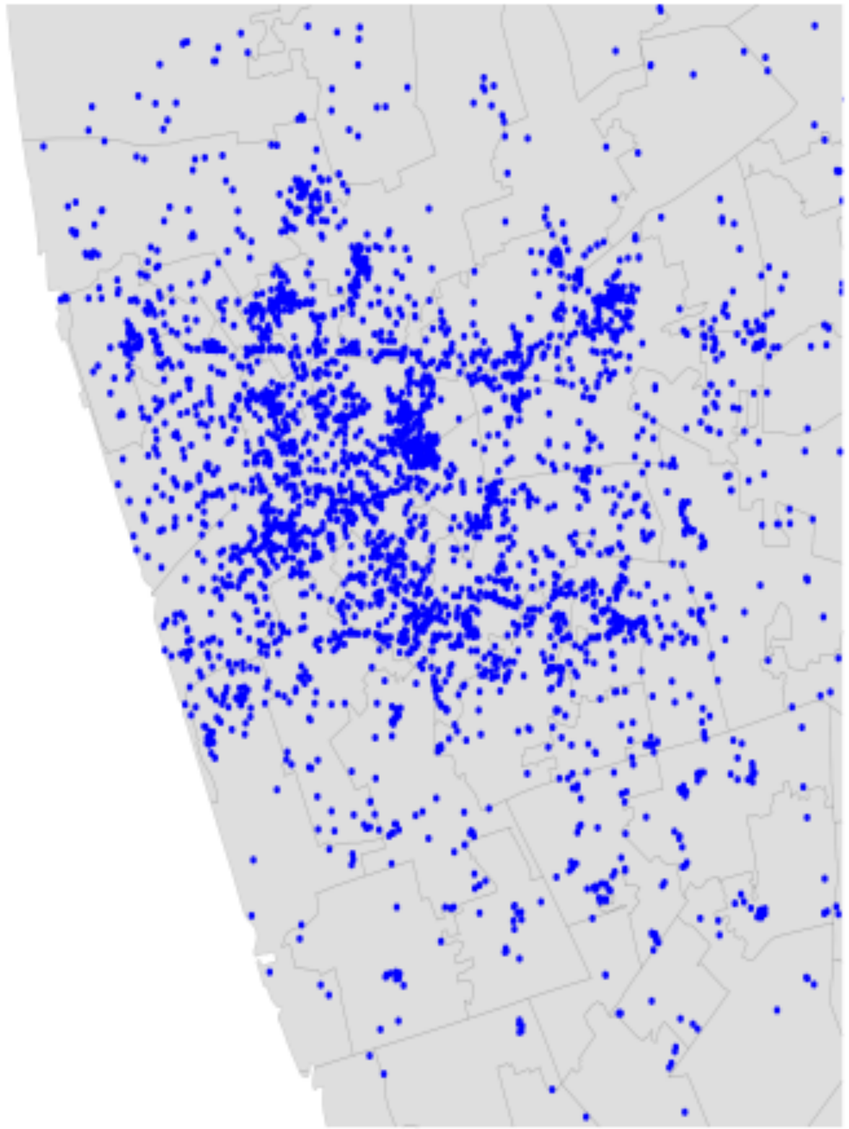
Visualization: One-to-one (scatter)

Intuitive

Effective for small data

No aggregation = no MAUP

Useless for large or very clustered data

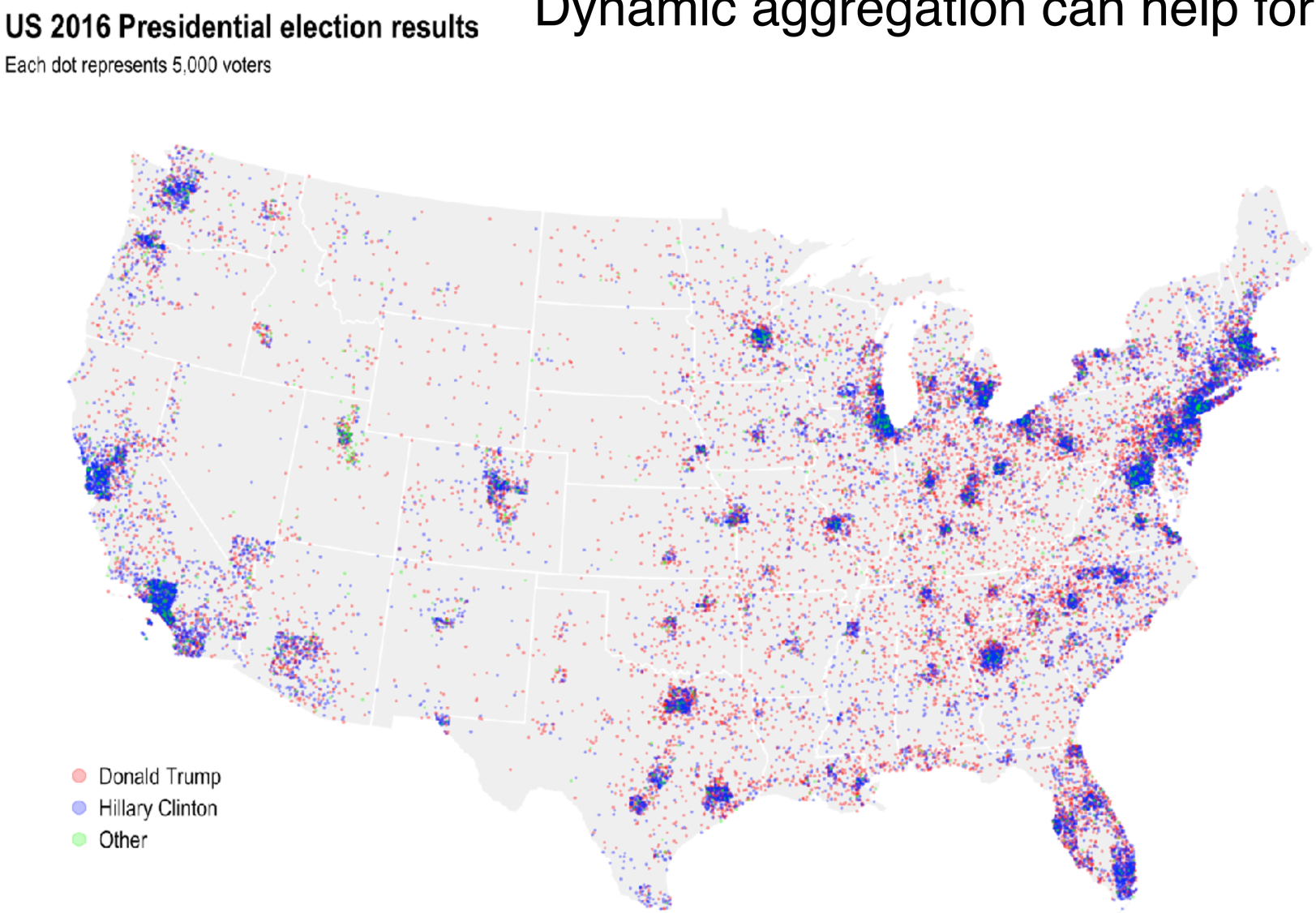


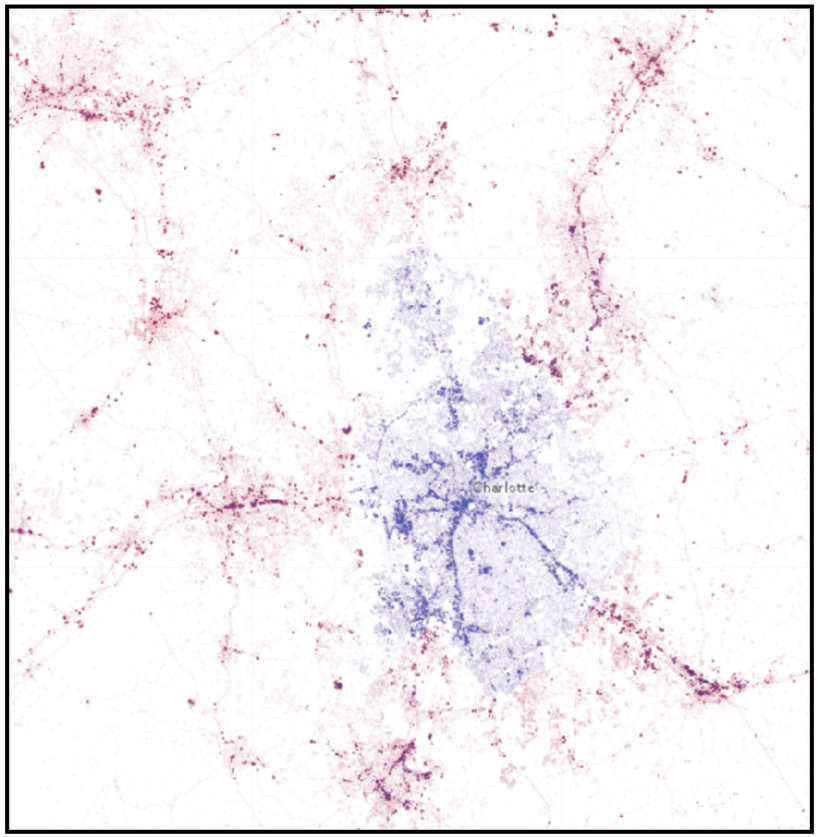
Visualization: One-to-one (scatter)

Dynamic aggregation can help for large data sets

Visualization: One-to-one (scatter)

Dynamic aggregation can help for large data sets





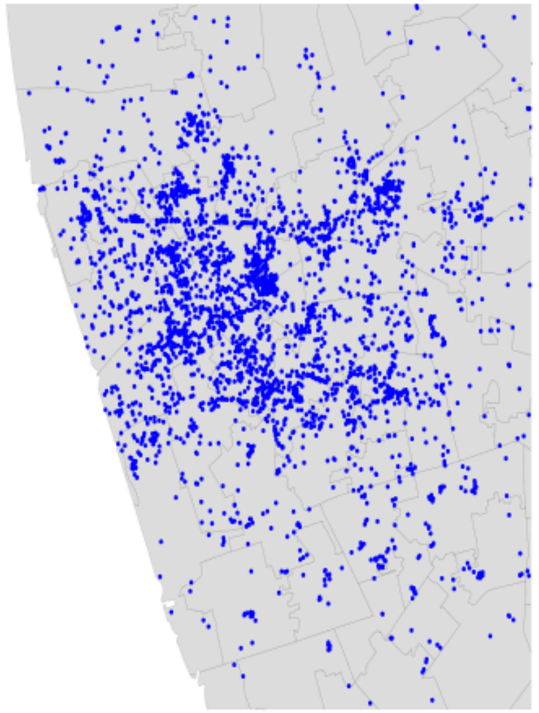
https://www.andybeger.com/2018/05/11/dot-density-map-of-the-2016-election/
https://www.maproomblog.com/2018/04/kenneth-fields-dot-density-election-map-redux/

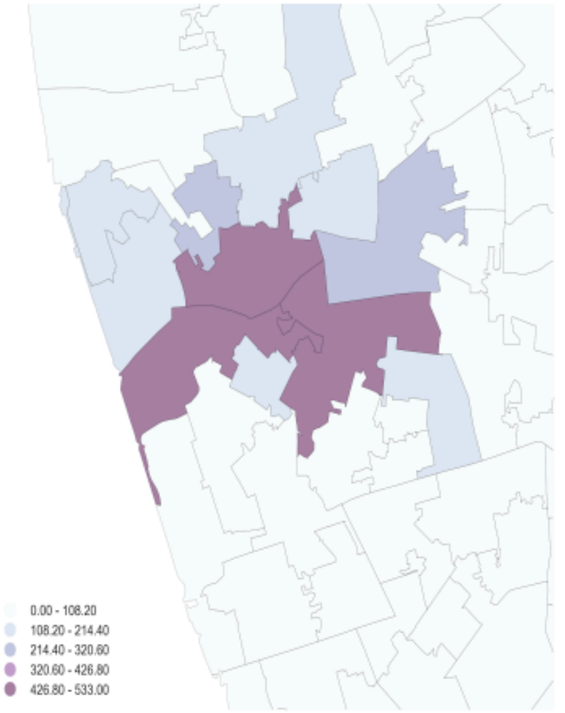
Visualization: Aggregate (choropleth)

Aggregation into polygons

reduces complexity

MAUP





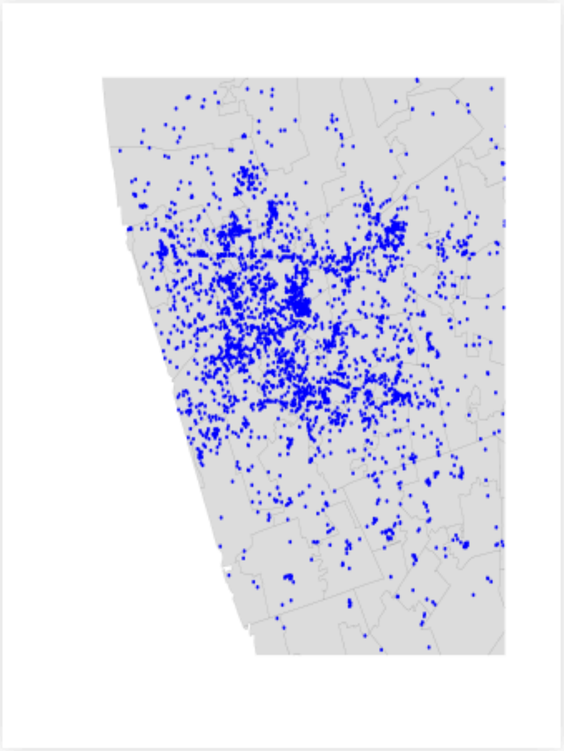
Visualization: Aggregate (tesselation)

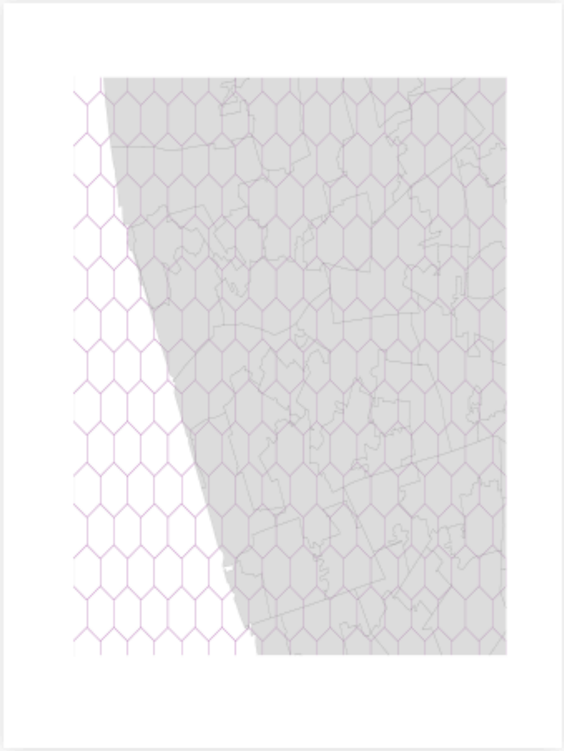
Hexagonal tesselation:

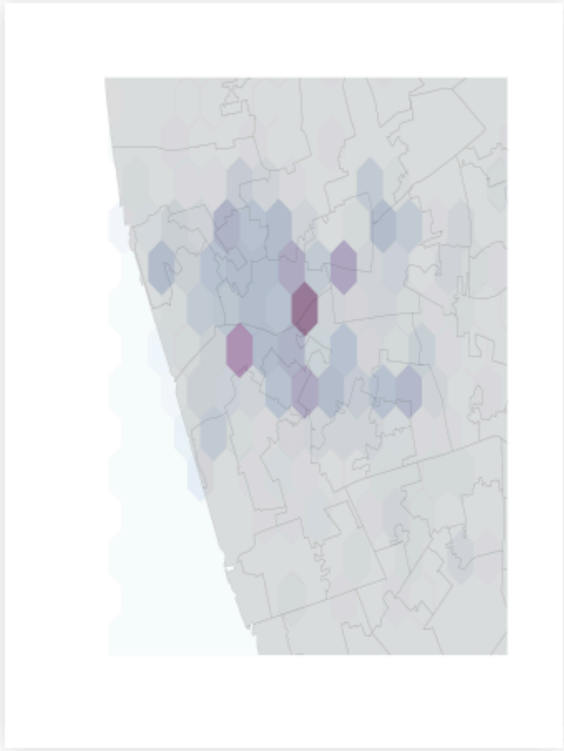
- Regular
- Exhaust space
- Many sides

  (no rook/queen issues)

MAUP





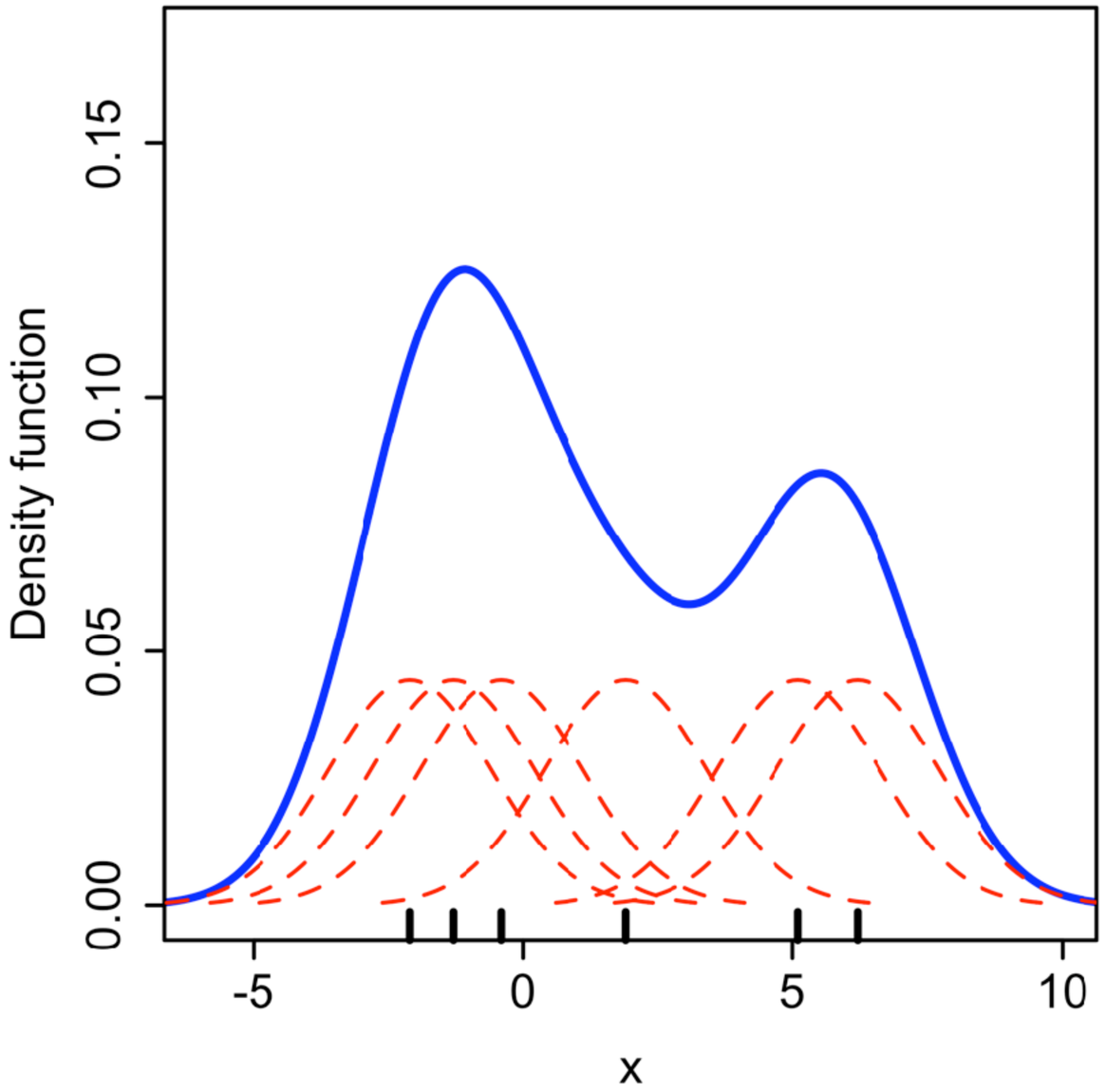


Visualization: Smooth (KDE)

Kernel Density Estimation

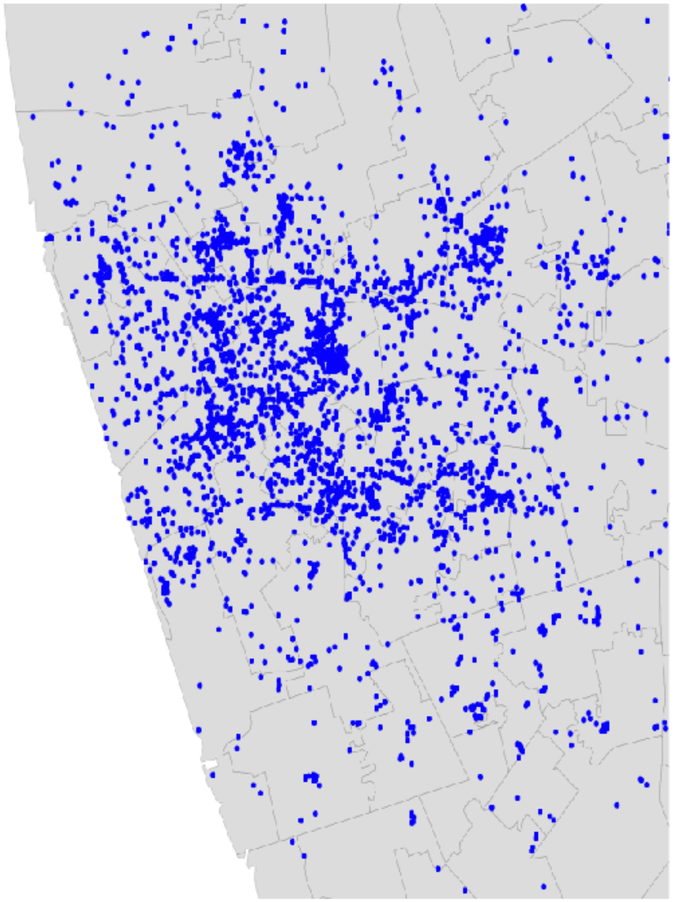
"Continuous histogram"

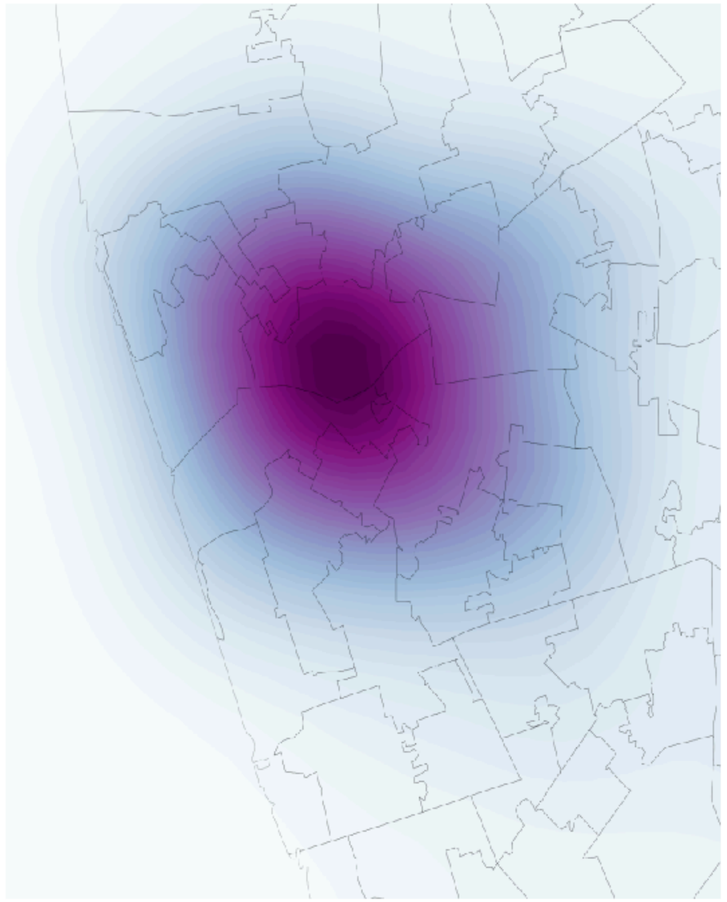
Solves MAUP



https://en.wikipedia.org/wiki/Kernel_density_estimation

Visualization: Smooth (KDE)





Modeling the underlying process

Point patterns come from a data-generating process

We can only observe the pattern.

To infer the process we compare synthetic patterns from a null model with observed patterns. If different, we can reject the null model.

Point patterns come from a data-generating process

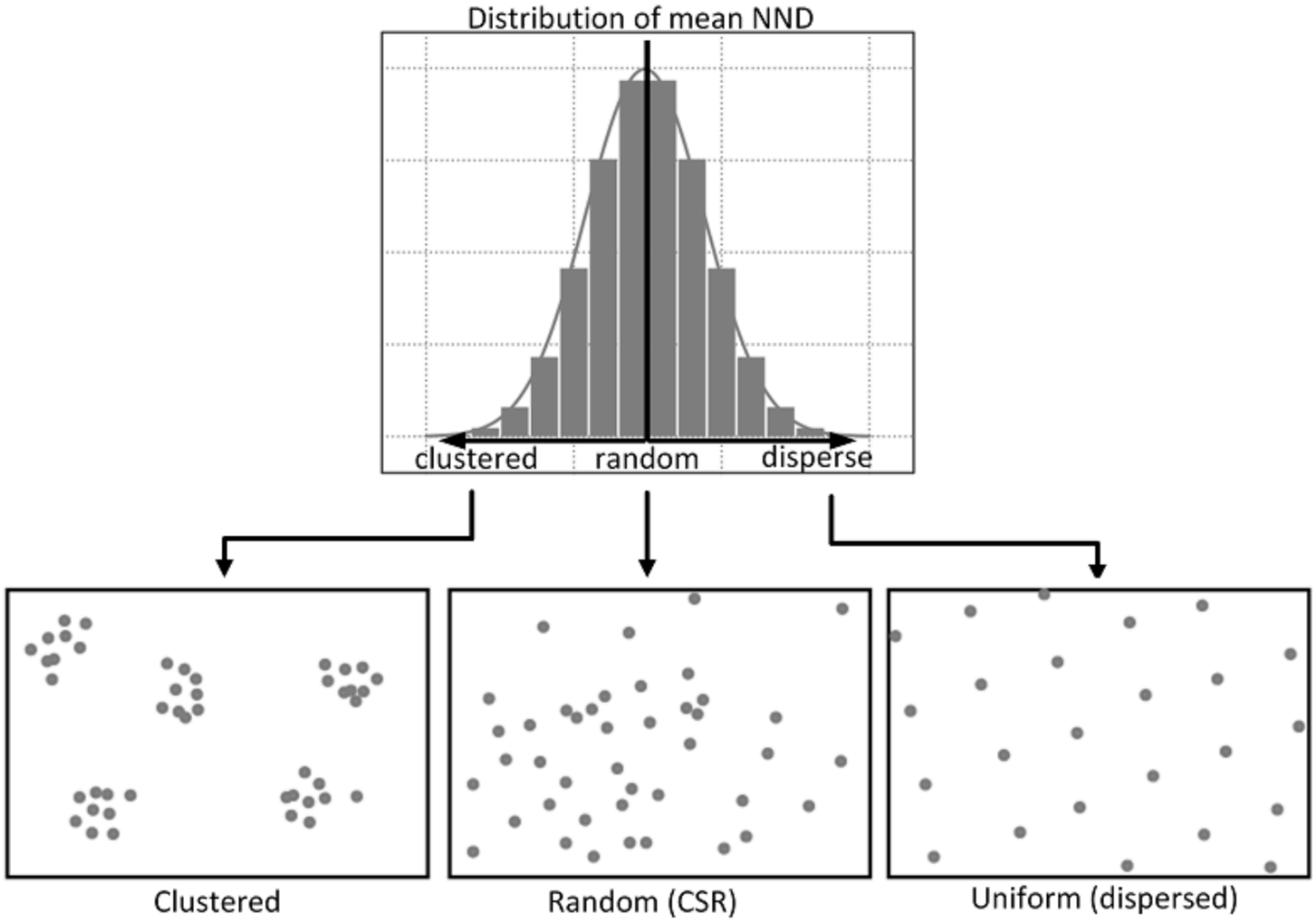
We can only observe the pattern.

To infer the process we compare synthetic patterns from a null model with observed patterns. If different, we can reject the null model.

Here we just consider as null model complete spatial randomness (CSR), generated by a homogeneous spatial Poisson process

The nearest neighbor distance is a useful characteristic

$$\bar{D} = \frac{1}{n}\sum_{i=1}^{n} d_i$$



https://gistbok.ucgis.org/bok-topics/point-pattern-analysis

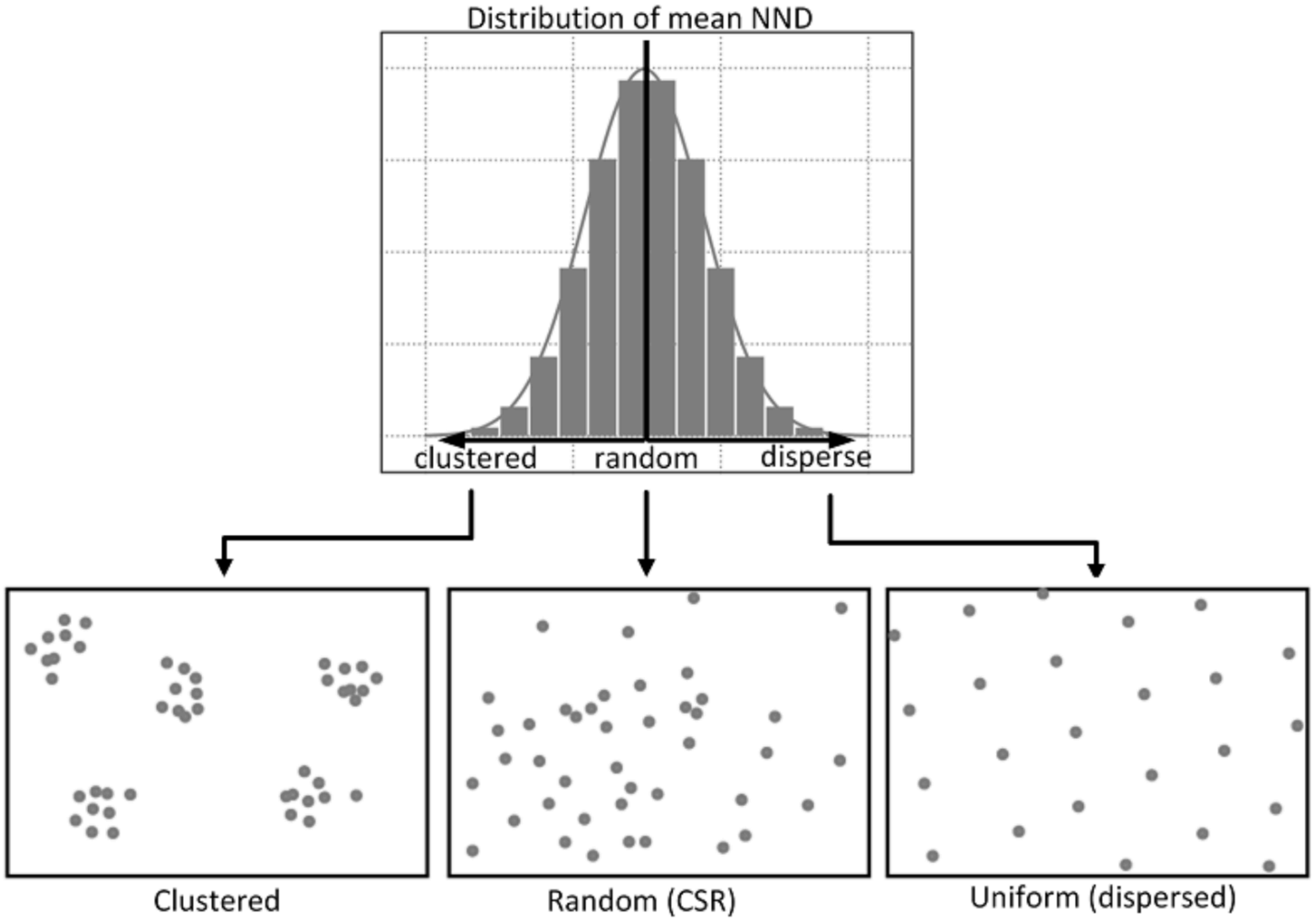
The nearest neighbor distance is a useful characteristic

$$\bar{D} = \frac{1}{n}\sum_{i=1}^{n} d_i$$

$$\bar{D}_{CSR} = \frac{0.5}{\sqrt{\frac{n}{A}}}$$

$$z = \frac{\bar{D} - \bar{D}_{CSR}}{SE}$$

$$SE = \frac{0.261}{\sqrt{n^2/A}}$$



https://gistbok.ucgis.org/bok-topics/point-pattern-analysis

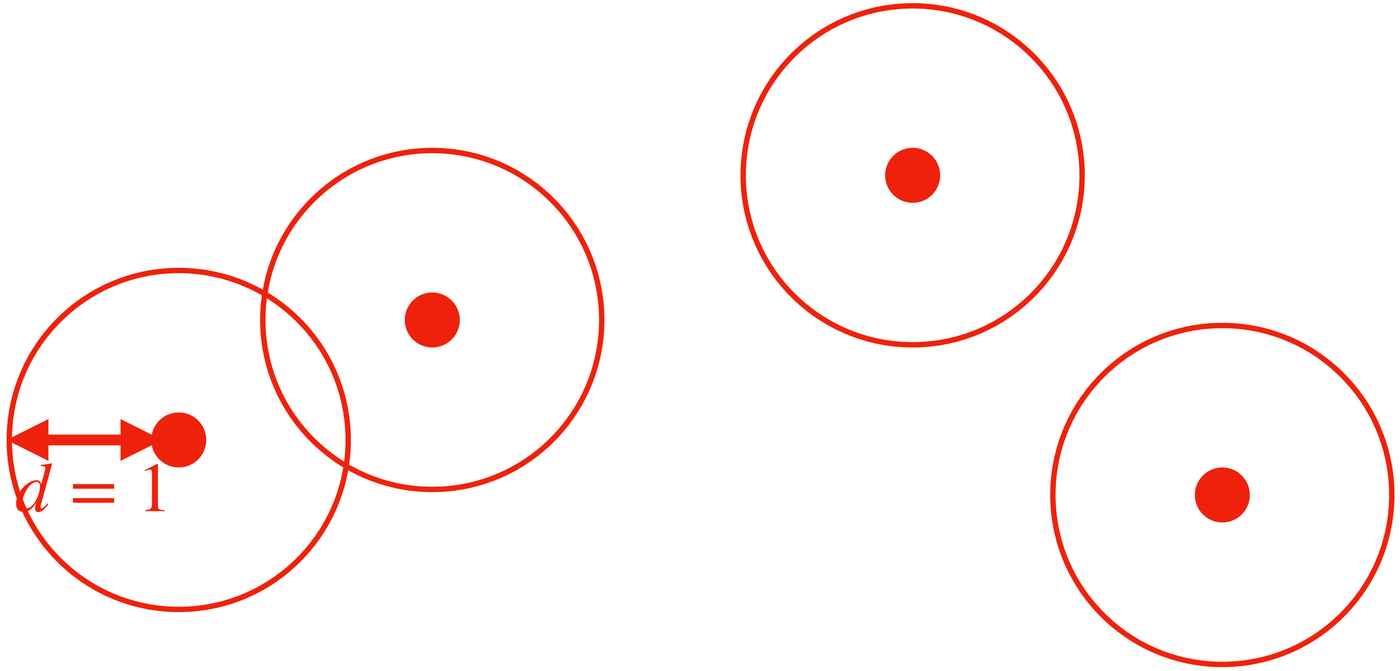
Ripley's $G(d)$ function measures clustering for a distance d

Ripley's $G(d)$ function is the fraction of points which have a neighbor within distance d

Ripley's $G(d)$ function measures clustering for a distance d

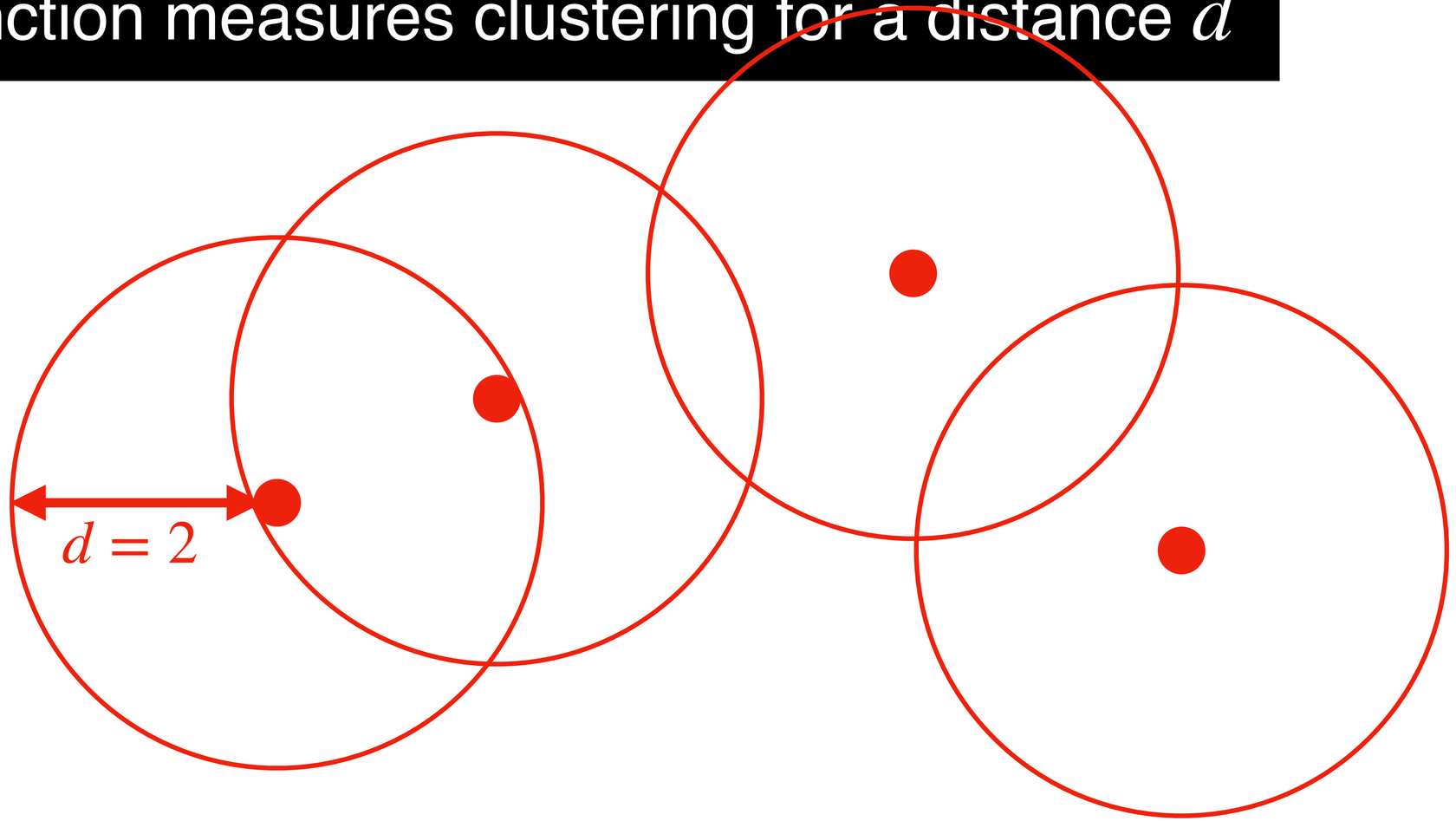
Data points

Ripley's $G(d)$ function measures clustering for a distance d



$$G(1) = 0/4 = 0$$

Ripley's $G(d)$ function measures clustering for a distance d



$G(1) = 0/4 = 0$

$G(2) = 2/4 = 0.5$

Ripley's $G(d)$ function measures clustering for a distance d

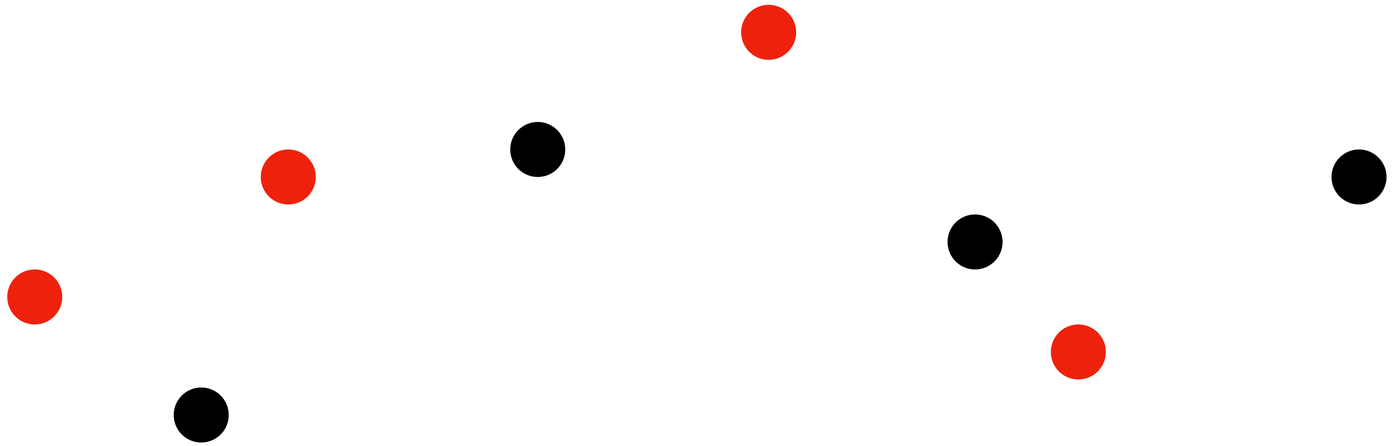
$G(1) = 0/4 = 0$

$G(2) = 2/4 = 0.5$

$G(3) = 4/4 = 1$

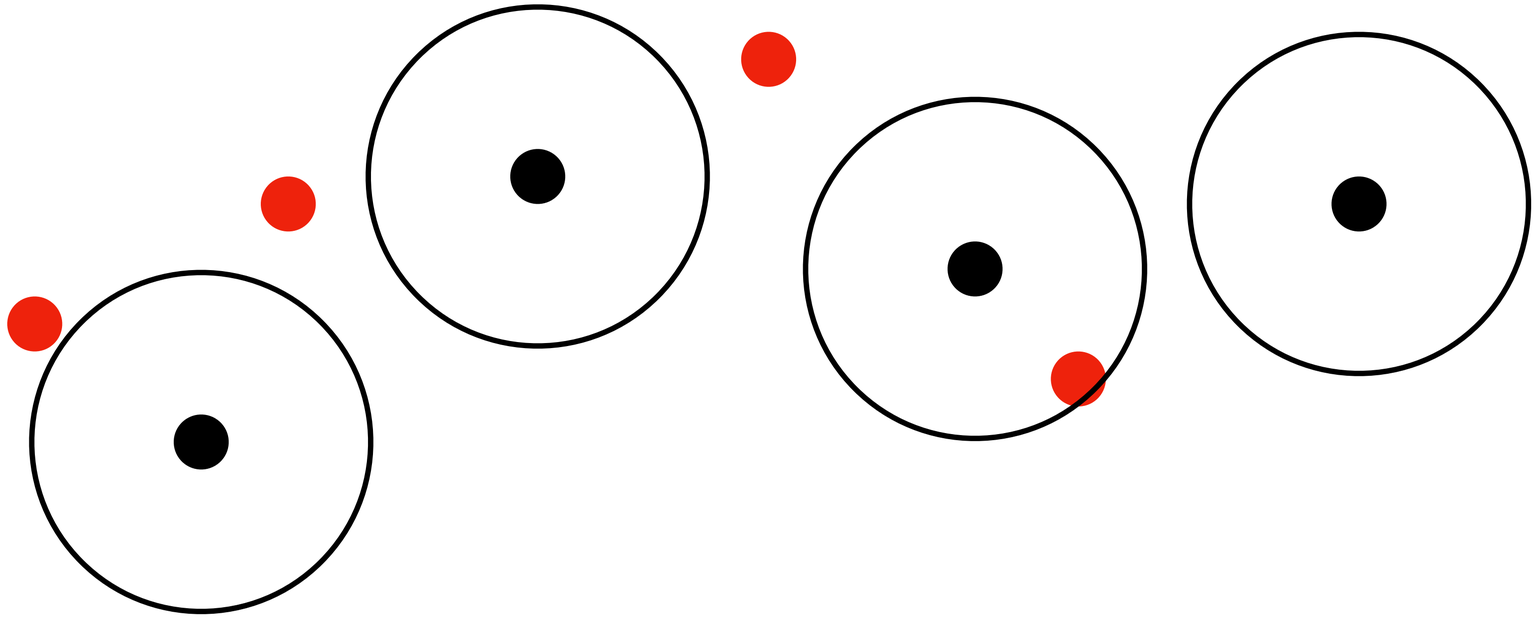
Ripley's $G(d)$ function measures clustering for a distance d

Now compare it to random points



$$G(1) = 0/4 = 0$$
$$G(2) = 2/4 = 0.5$$
$$G(3) = 4/4 = 1$$

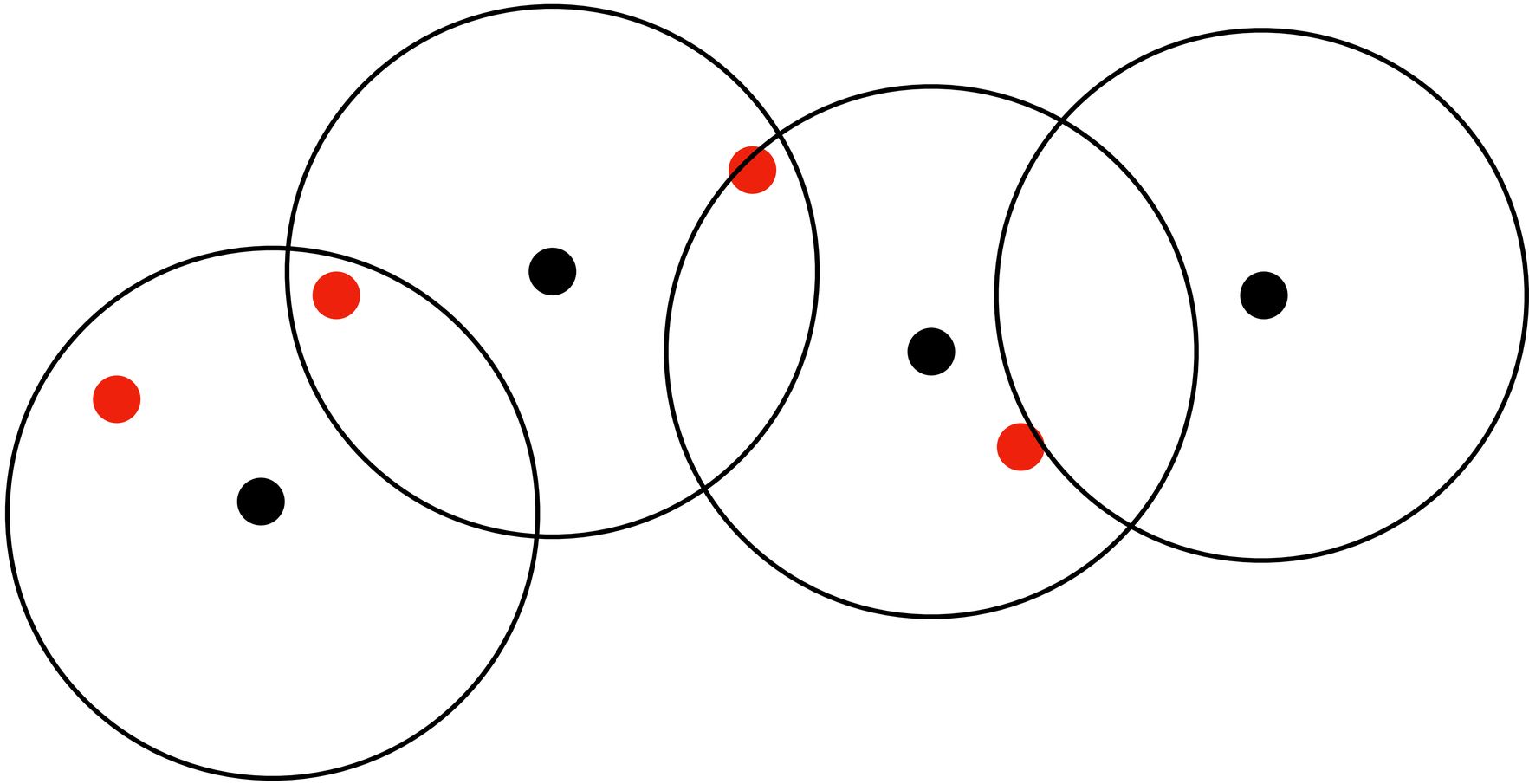
Ripley's $G(d)$ function measures clustering for a distance d



$G(1) = 0/4 = 0$
$G(2) = 2/4 = 0.5$
$G(3) = 4/4 = 1$

$G_{\text{rand}}(1) = 0/4 = 0$

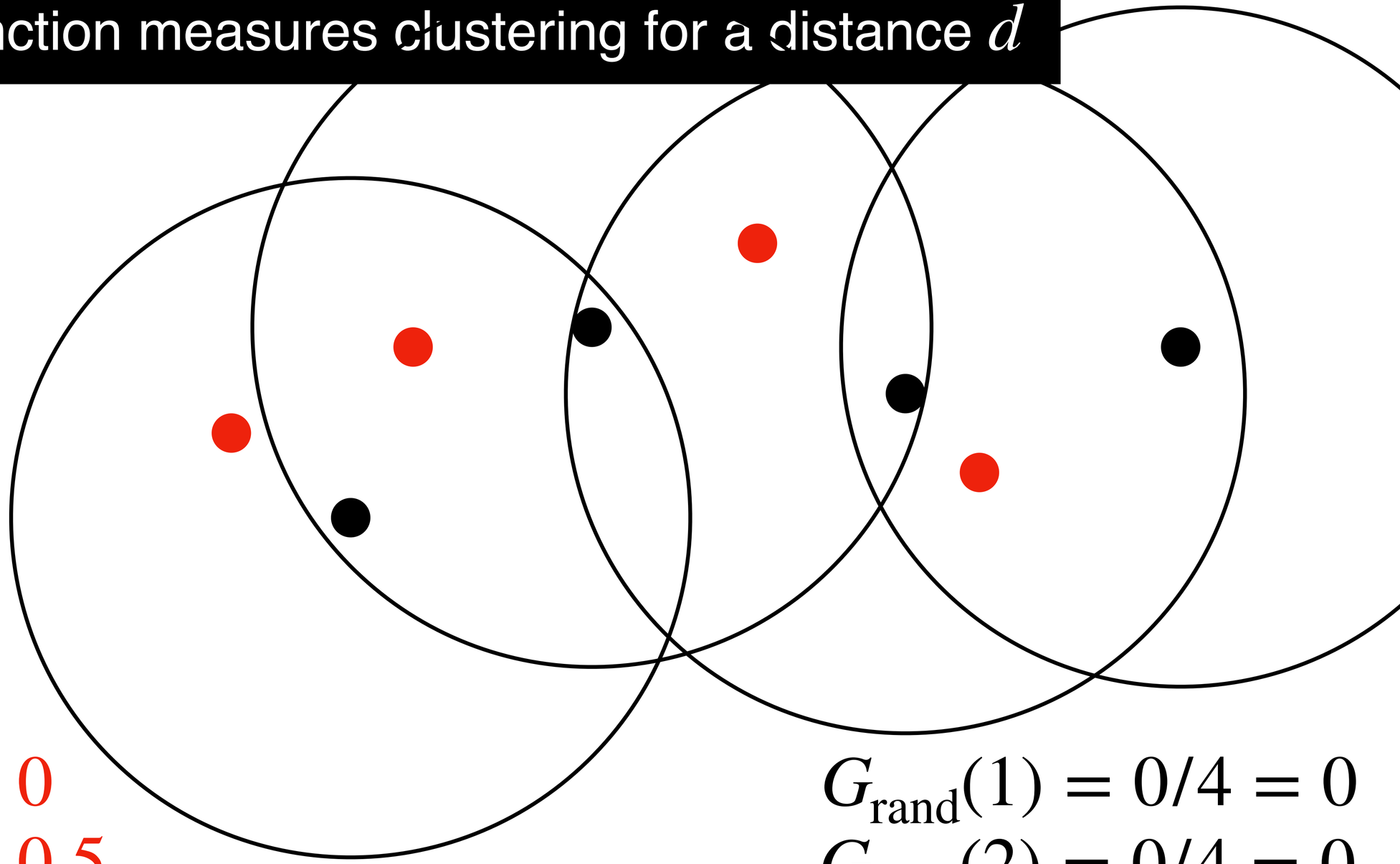
Ripley's $G(d)$ function measures clustering for a distance d



$G(1) = 0/4 = 0$
$G(2) = 2/4 = 0.5$
$G(3) = 4/4 = 1$

$G_{\text{rand}}(1) = 0/4 = 0$
$G_{\text{rand}}(2) = 0/4 = 0$

Ripley's $G(d)$ function measures clustering for a distance d

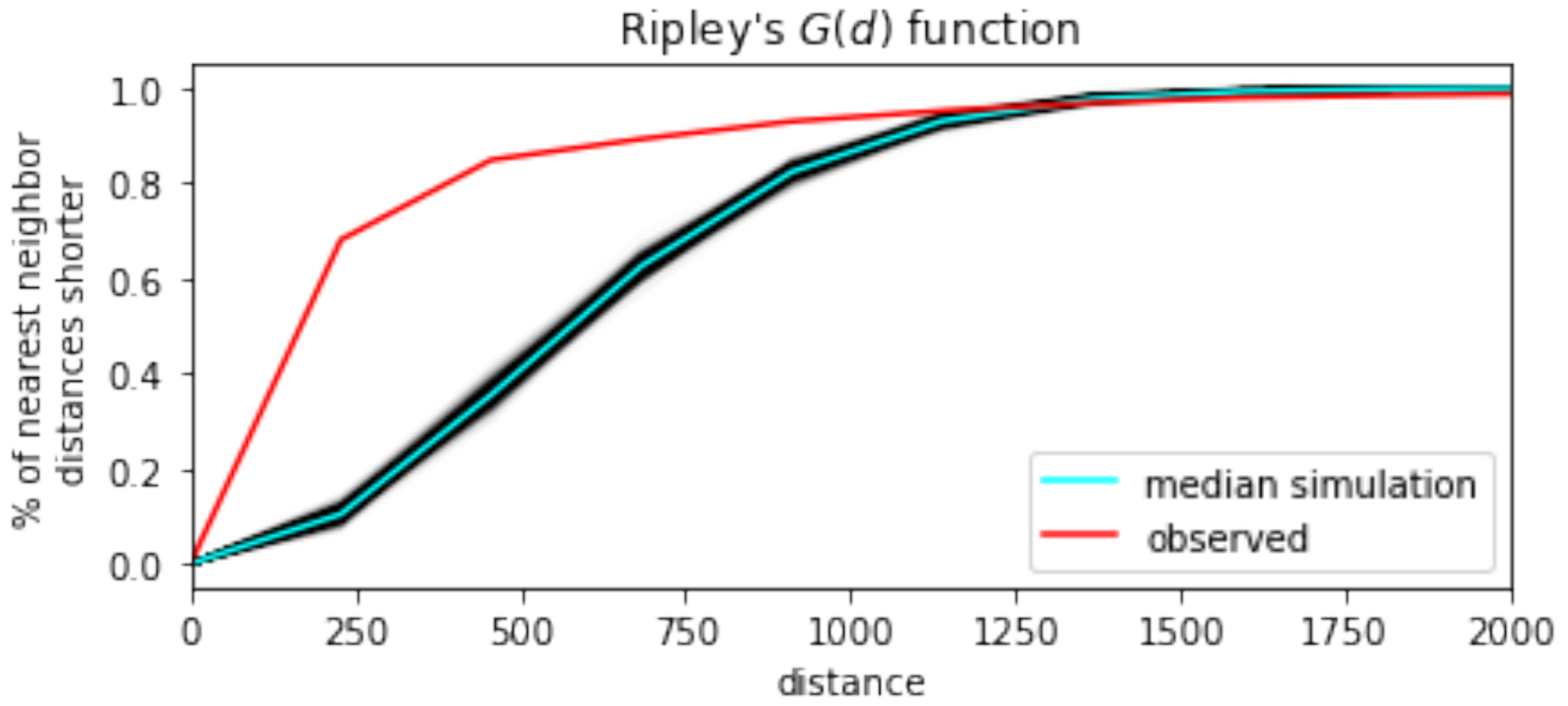


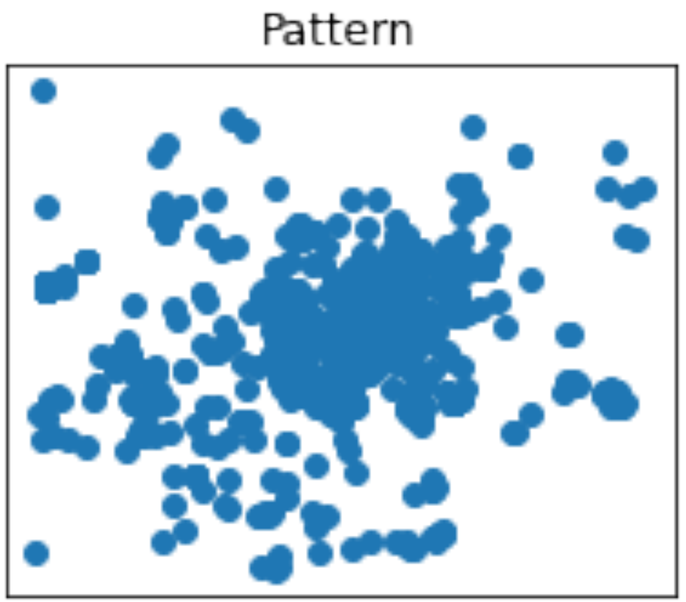
$G(1) = 0/4 = 0$
$G(2) = 2/4 = 0.5$
$G(3) = 4/4 = 1$

$G_{\text{rand}}(1) = 0/4 = 0$
$G_{\text{rand}}(2) = 0/4 = 0$
$G_{\text{rand}}(3) = 4/4 = 1$

Ripley's $G(d)$ function measures clustering for a distance d

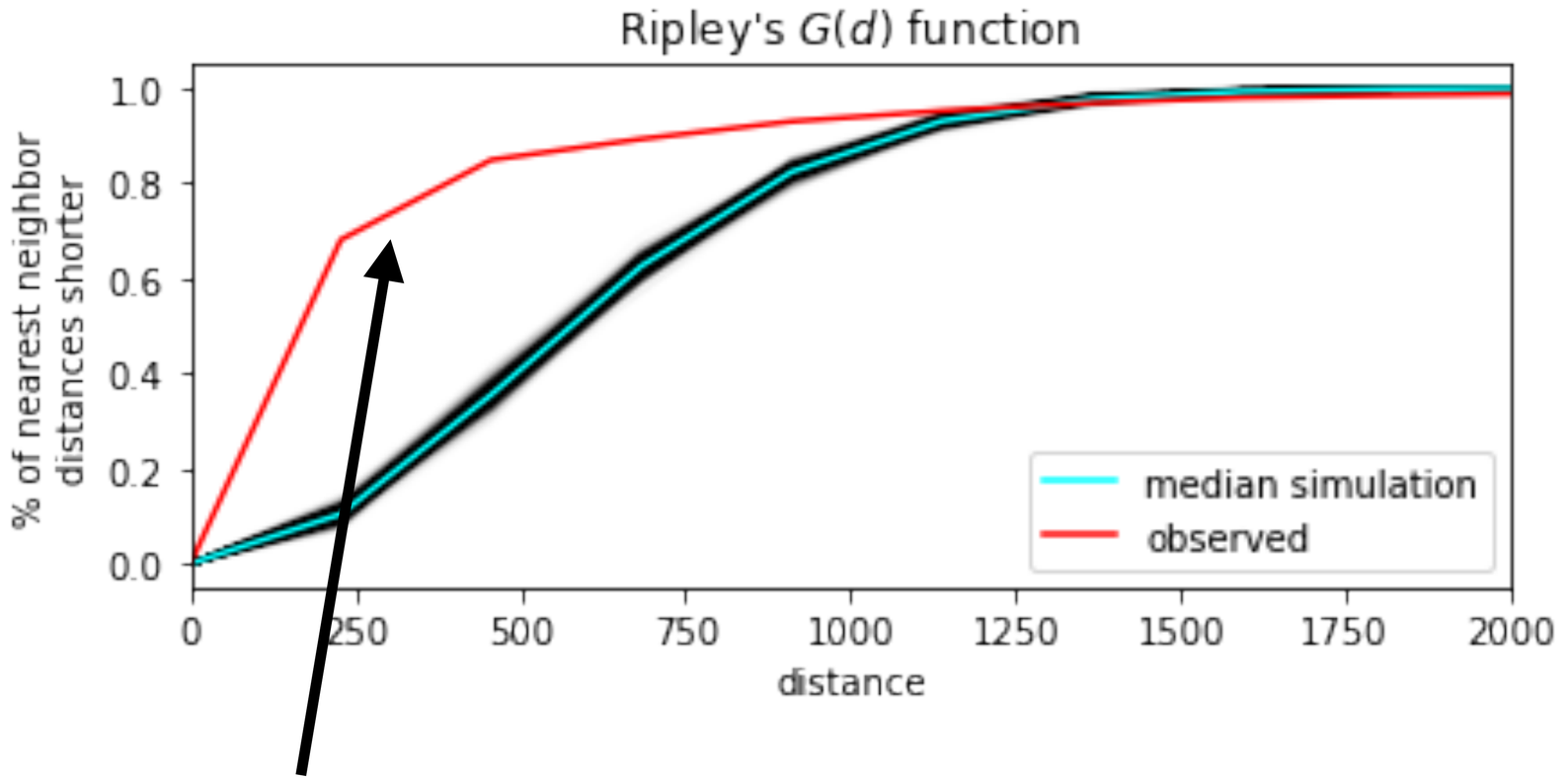
Sample many times to get a simulation envelope with a confidence level

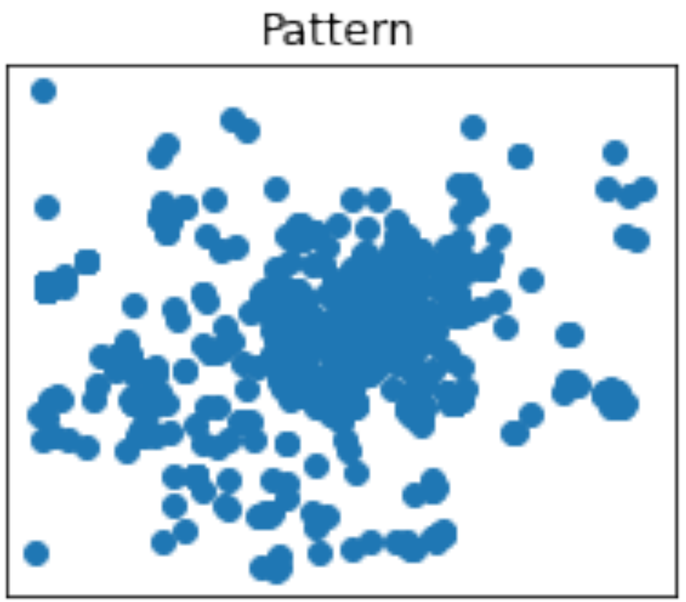




Ripley's $G(d)$ function measures clustering for a distance d

Sample many times to get a simulation envelope with a confidence level





For those distances where the observed curve is above the envelope, we have significant clusters

Ripley's $F(d)$ function measures empty space for a distance d

Ripley's $F(d)$ function is the fraction of random points which have a neighbor from the data within distance d

Ripley's $F(d)$ function measures empty space for a distance d

Ripley's $F(d)$ function is the fraction of random points which have a neighbor from the data within distance d

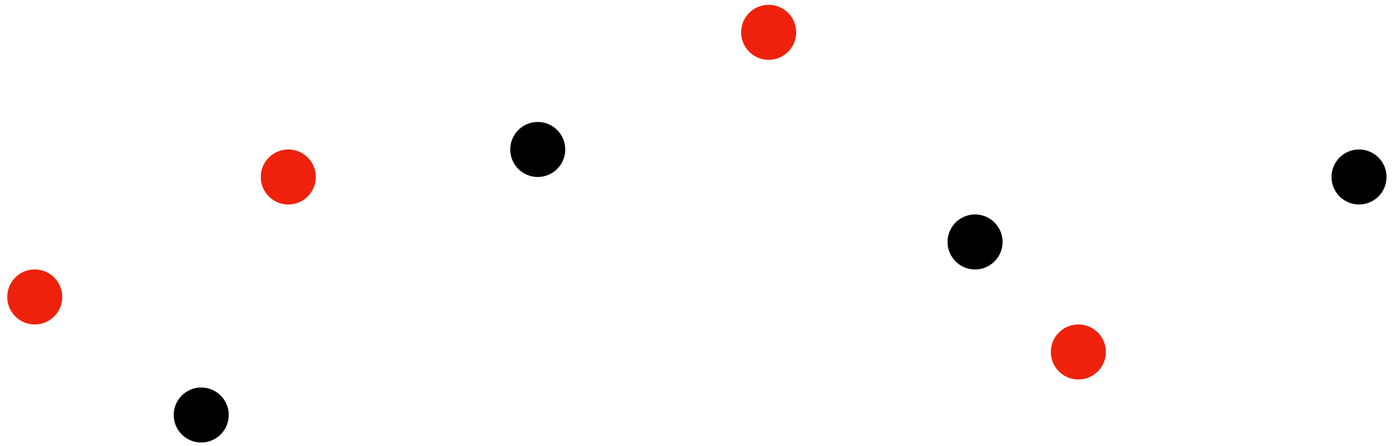
It is also called the "empty space function" because it measures the typical distance from arbitrary points in empty space.

It can be better to use $F(d)$ than $G(d)$ when there are few points.

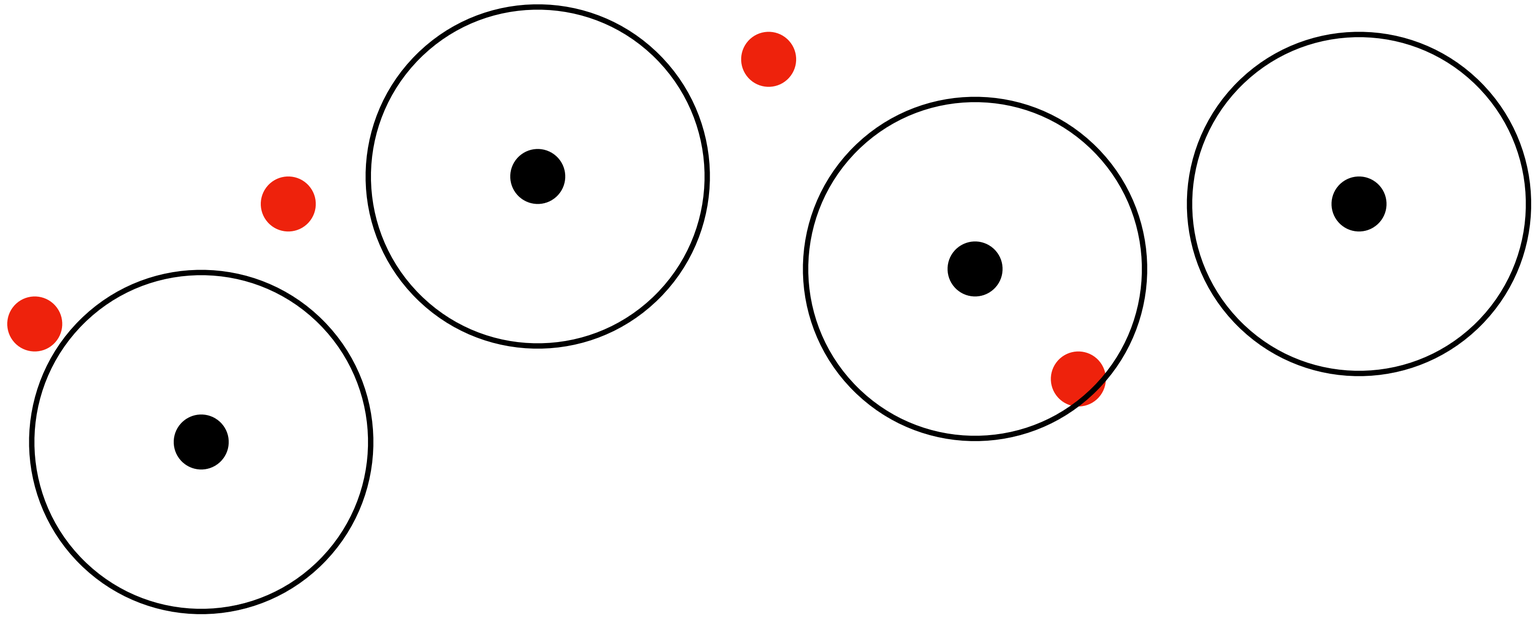
Ripley's $F(d)$ function measures empty space for a distance d

Data points

Random points

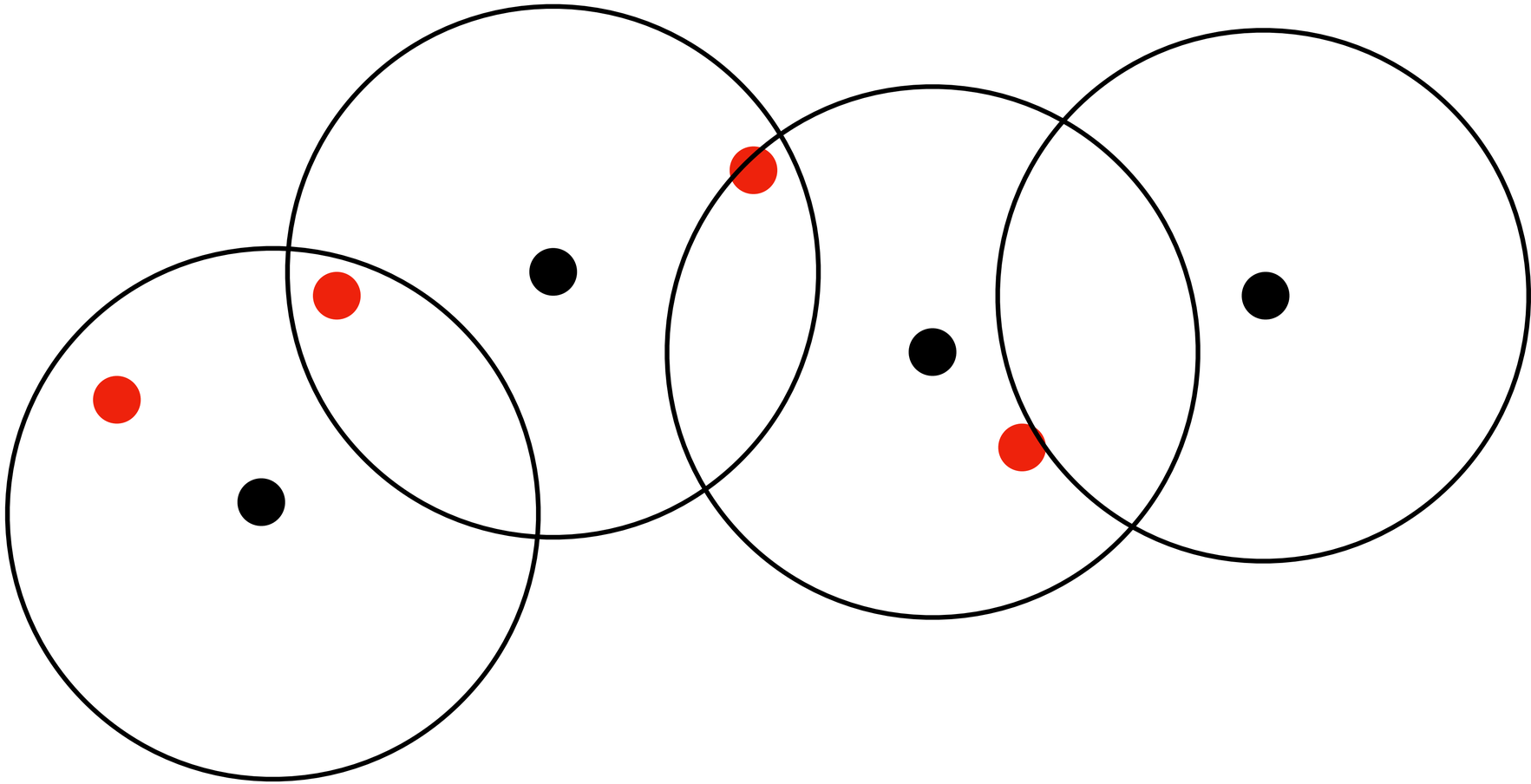


Ripley's $F(d)$ function measures empty space for a distance d



$F(1) = 1/4 = 0.25$

Ripley's $F(d)$ function measures empty space for a distance d

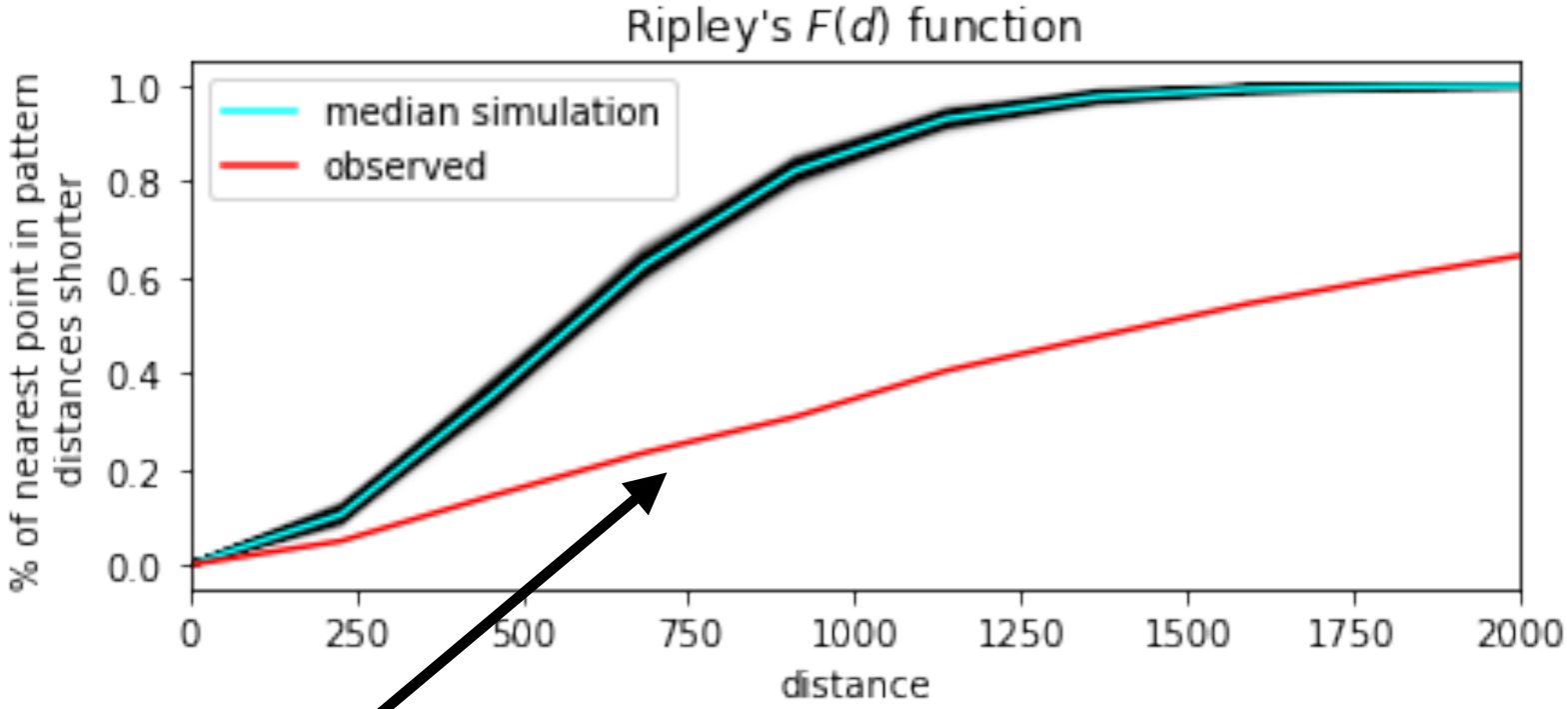


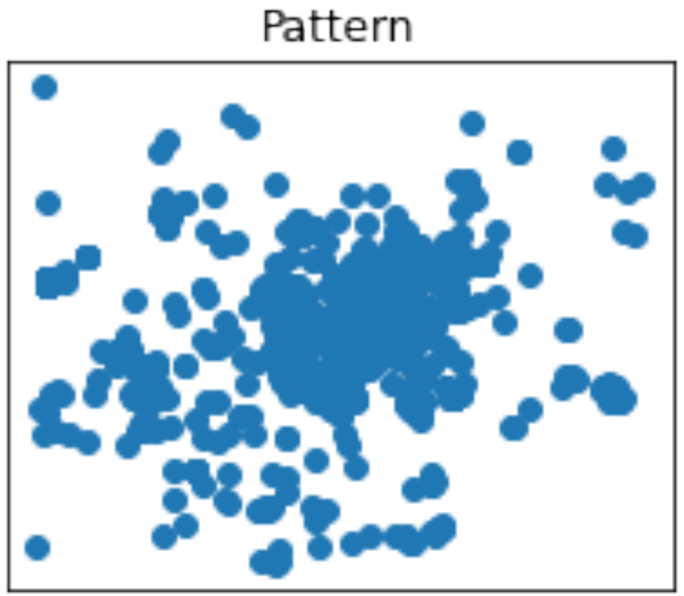
$F(1) = 1/4 = 0.25$

$F(2) = 4/4 = 1$

Ripley's $F(d)$ function measures empty space for a distance d

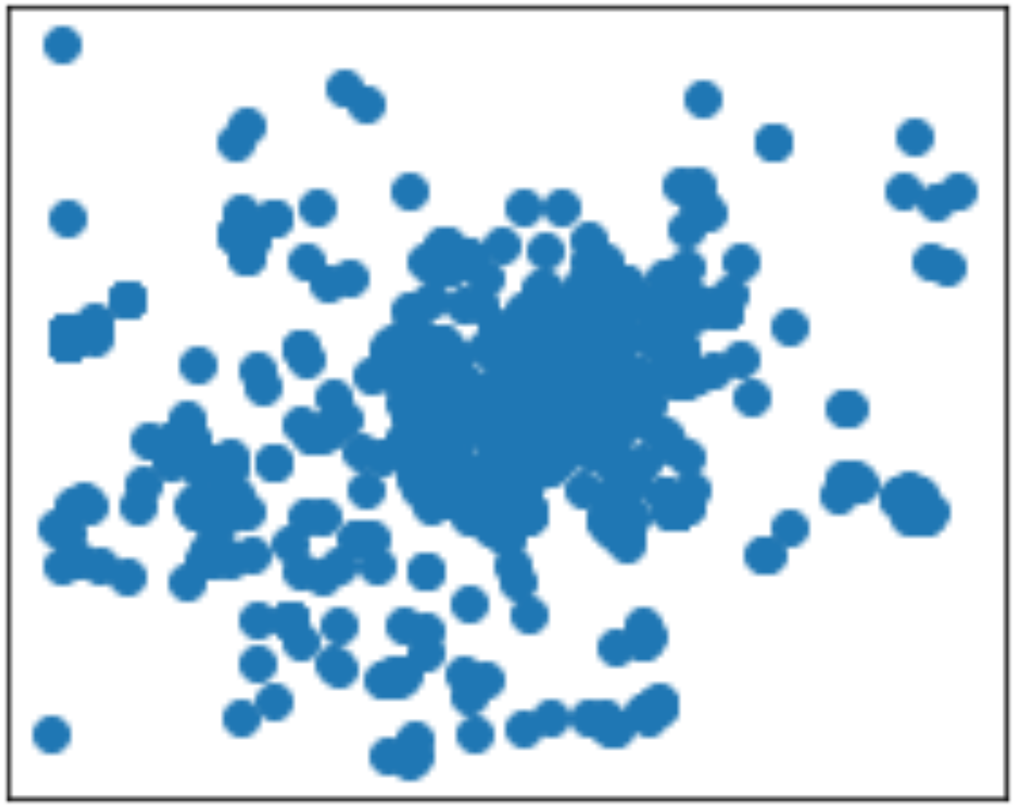
Sample many times to get a simulation envelope with a confidence level



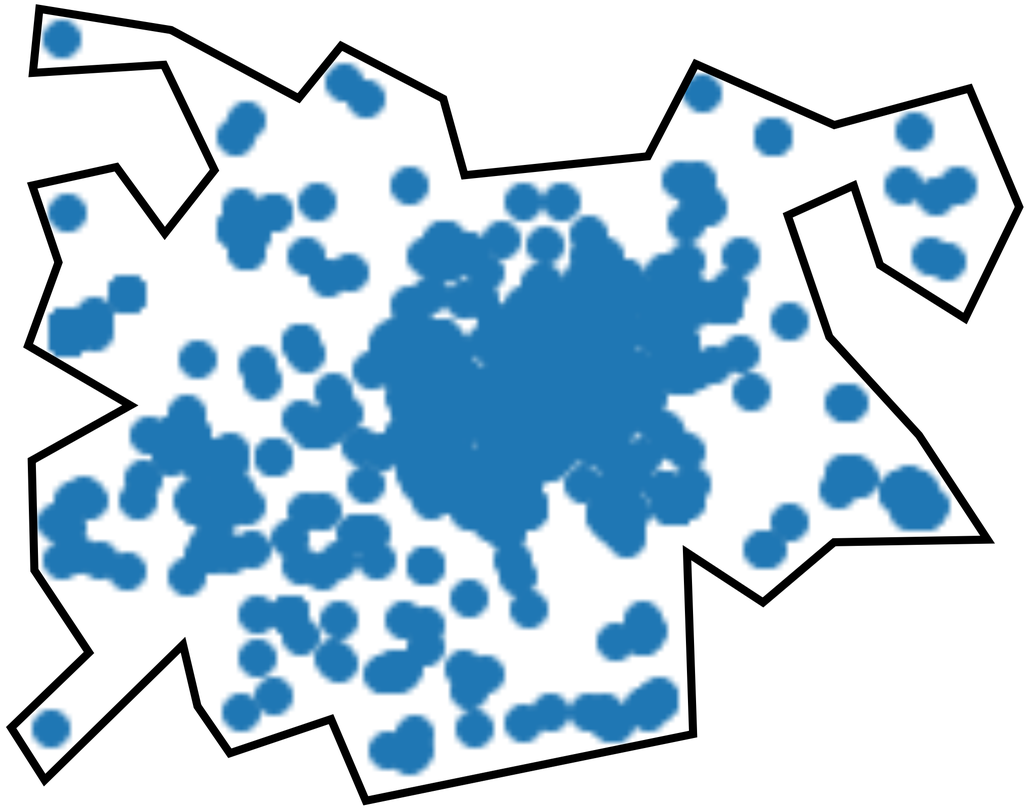


For those distances where the observed curve is below the envelope, we have significant gaps

Ripley's $F(d)$ is sensitive to the window!



$\neq$



Ripley's $F(d)$ and $G(d)$ are first-order

First-order processes consider that all points happen as separate occurences without interactions between positions

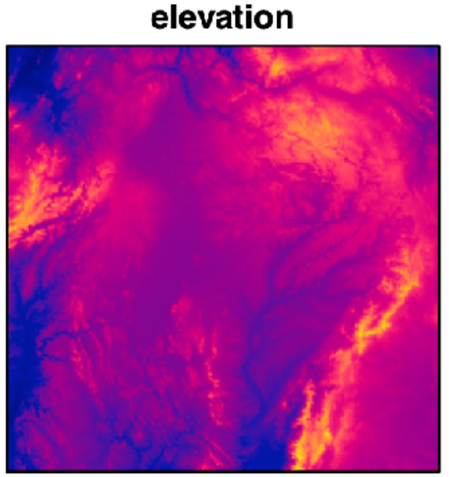
To model second-order processes with interactions like competition (e.g. territory) or synergy (e.g. seed dispersal) use other measures like Ripley's $K(d)$

There are much more complex spatial models

Null: complete spatial randomness

Trend: inhomogeneous processes

Covariates: dependence on covariates

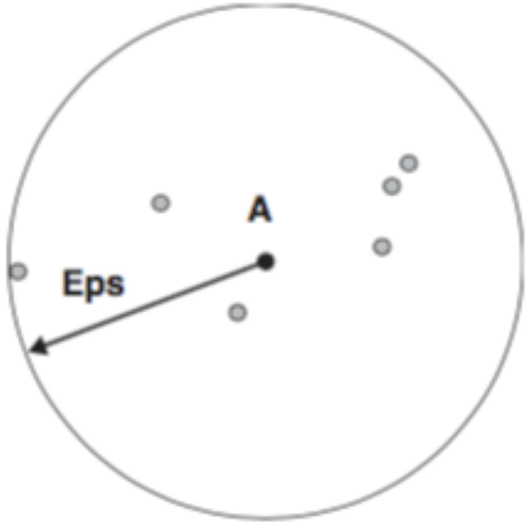


Interaction: clustering/repulsion

Finding clusters with DBSCAN

Density-based spatial clustering of applications with noise

DBSCAN: Density-based clustering



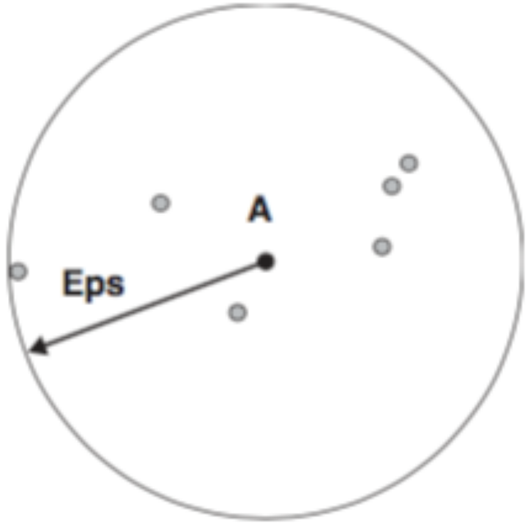
density of A = 7

1) Select a radius Eps

2) Count all points within Eps (including the central point itself). This is the density of the point.

What are the extreme cases for the radius?

DBSCAN: Density-based clustering



density of A = 7

1) Select a radius Eps

2) Count all points within Eps (including the central point itself). This is the density of the point.

What are the extreme cases for the radius?

Too small: Every point has density 1

Too big: Every point has density n

DBSCAN: Density-based clustering

1) Select a radius Eps

2) Count all points within Eps
(including the central point itself).
This is the density of the point.

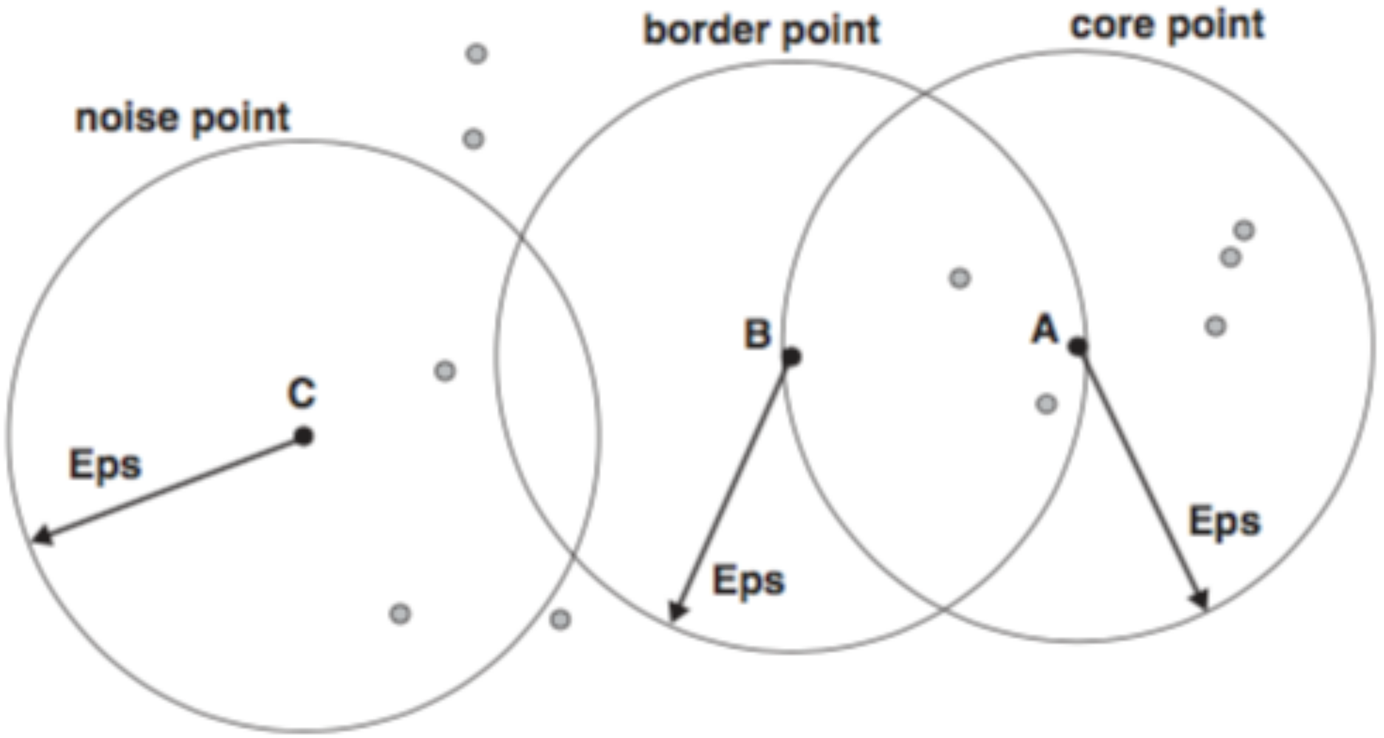
3) Select a parameter MinPts

DBSCAN: Density-based clustering

1) Select a radius Eps

2) Count all points within Eps (including the central point itself). This is the density of the point.

3) Select a parameter MinPts



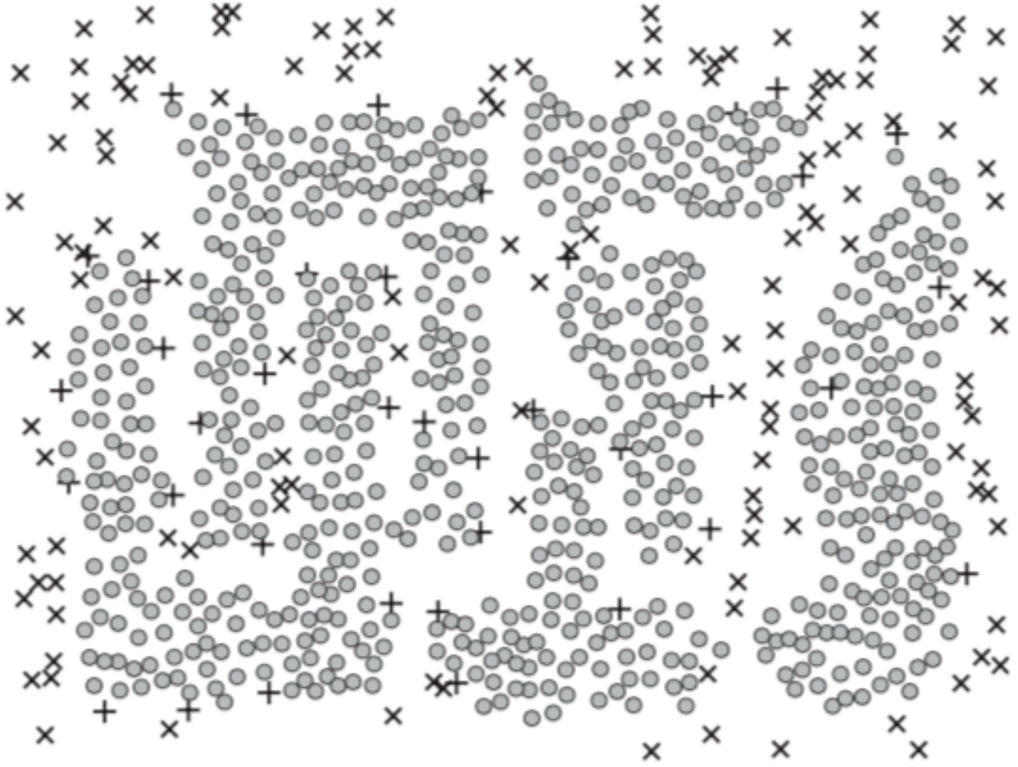
Core point: Has density ≥ MinPts

Border point: No core point but is in the neighborhood of a core point

Noise point: Neither core nor border point

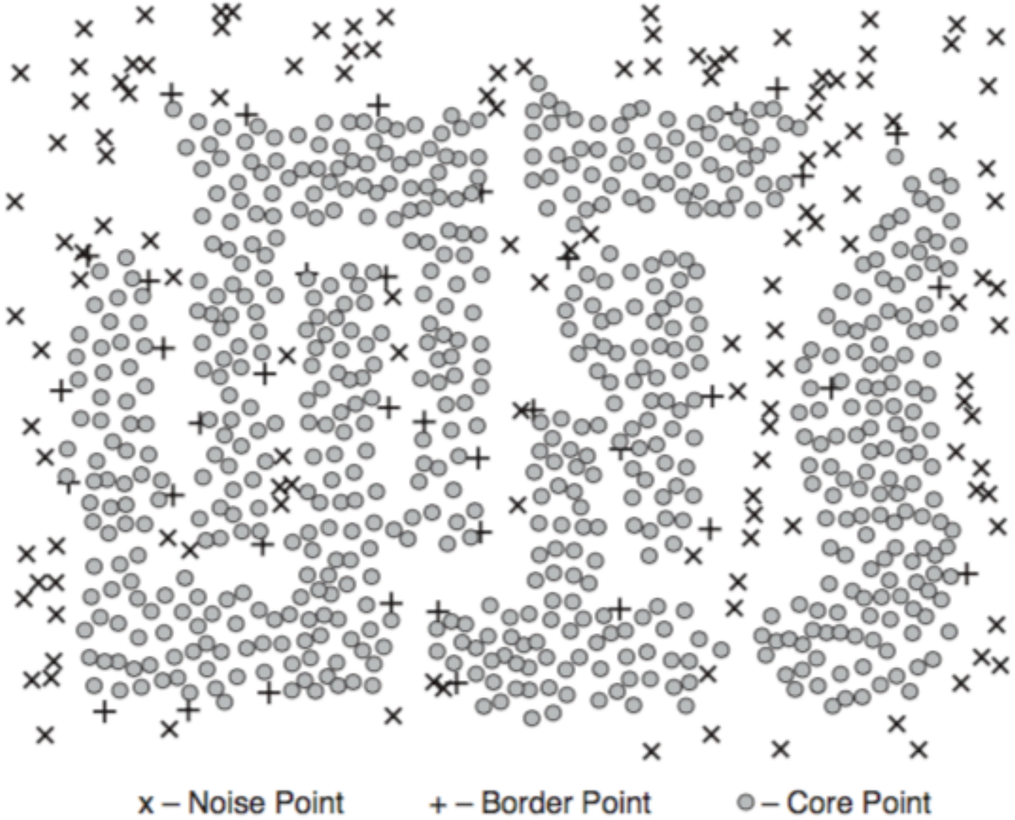
Algorithm 8.4 DBSCAN algorithm.

1: Label all points as core, border, or noise points.
2: Eliminate noise points.
3: Put an edge between all core points that are within *Eps* of each other.
4: Make each group of connected core points into a separate cluster.
5: Assign each border point to one of the clusters of its associated core points.



Algorithm 8.4 DBSCAN algorithm.

1: Label all points as core, border, or noise points.
2: Eliminate noise points.
3: Put an edge between all core points that are within *Eps* of each other.
4: Make each group of connected core points into a separate cluster.
5: Assign each border point to one of the clusters of its associated core points.

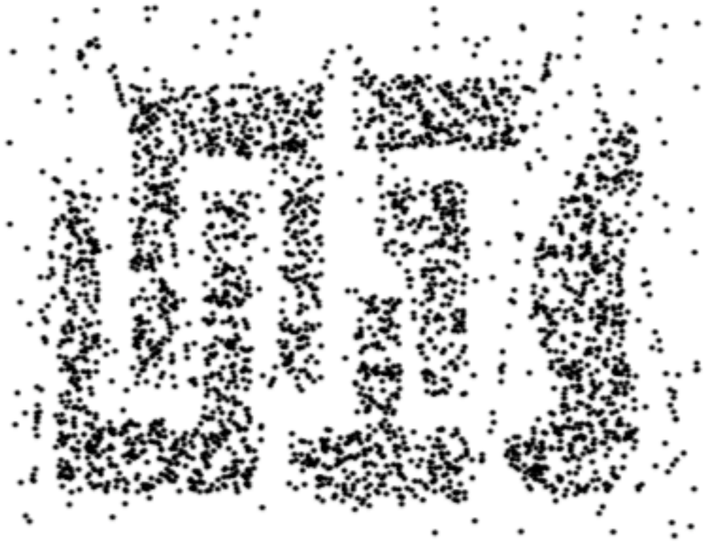


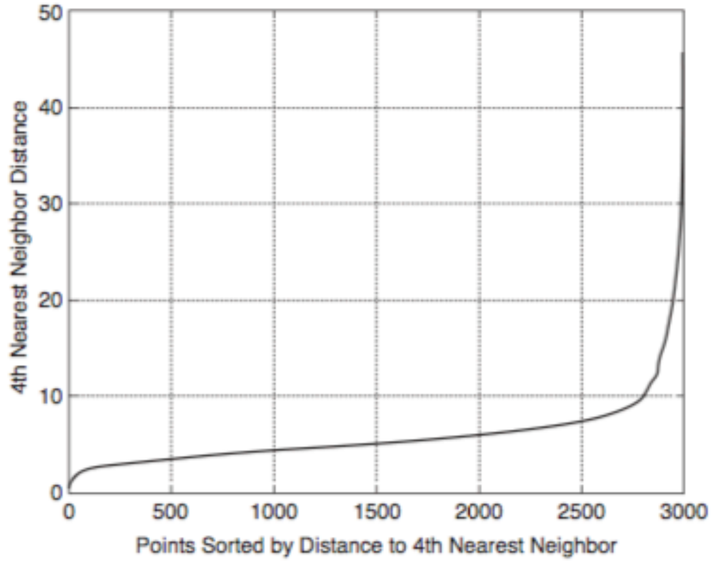
This is a partial clustering: Not all data points are assigned a cluster!

DBSCAN: How to choose parameters?

How to choose Eps and MinPts?

k-dist plot k=4



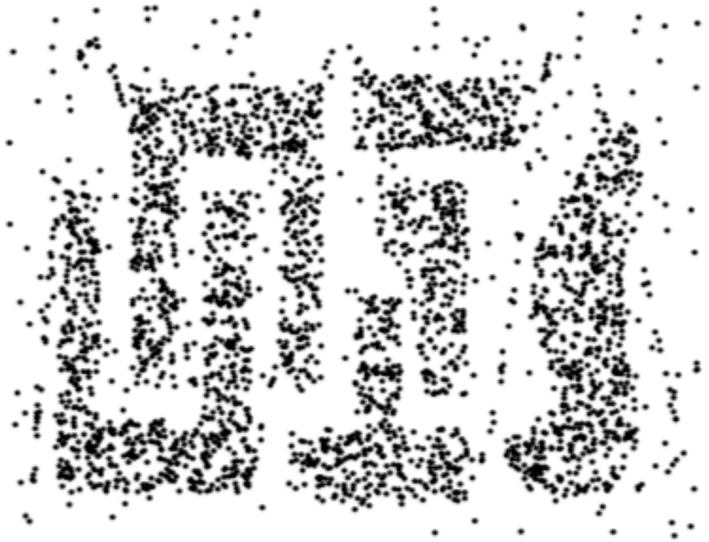


Look at the behavior of the distance from a point to it's kth nearest neighbor

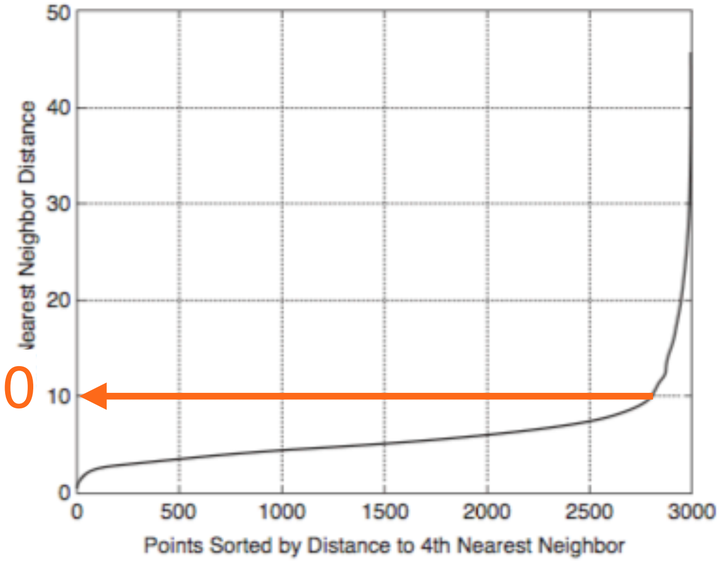
DBSCAN: How to choose parameters?

How to choose Eps and MinPts?

k-dist plot k=4



Eps=10



Look at the behavior of the distance from a point to it's kth nearest neighbor

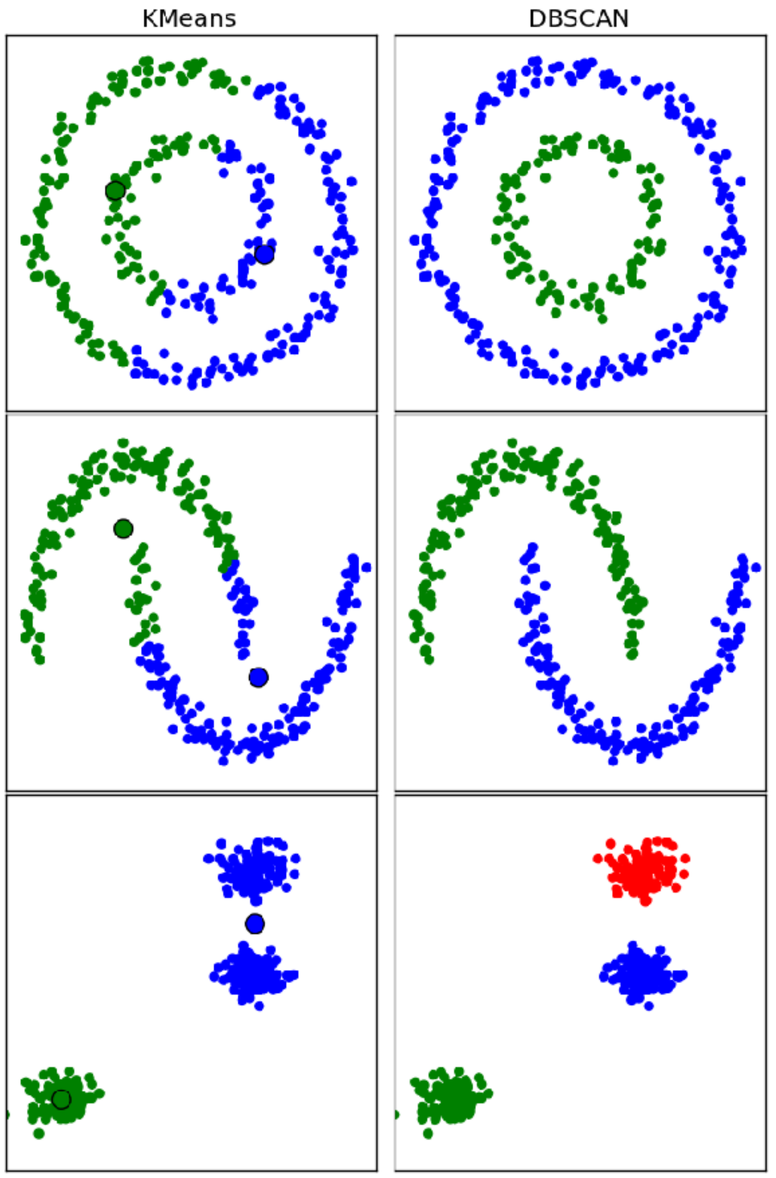
Noise points will have large distance. If they are systematic, there is a bend. This gives us a good Eps for the given k.

DBSCAN: Advantages

- Resistant to noise

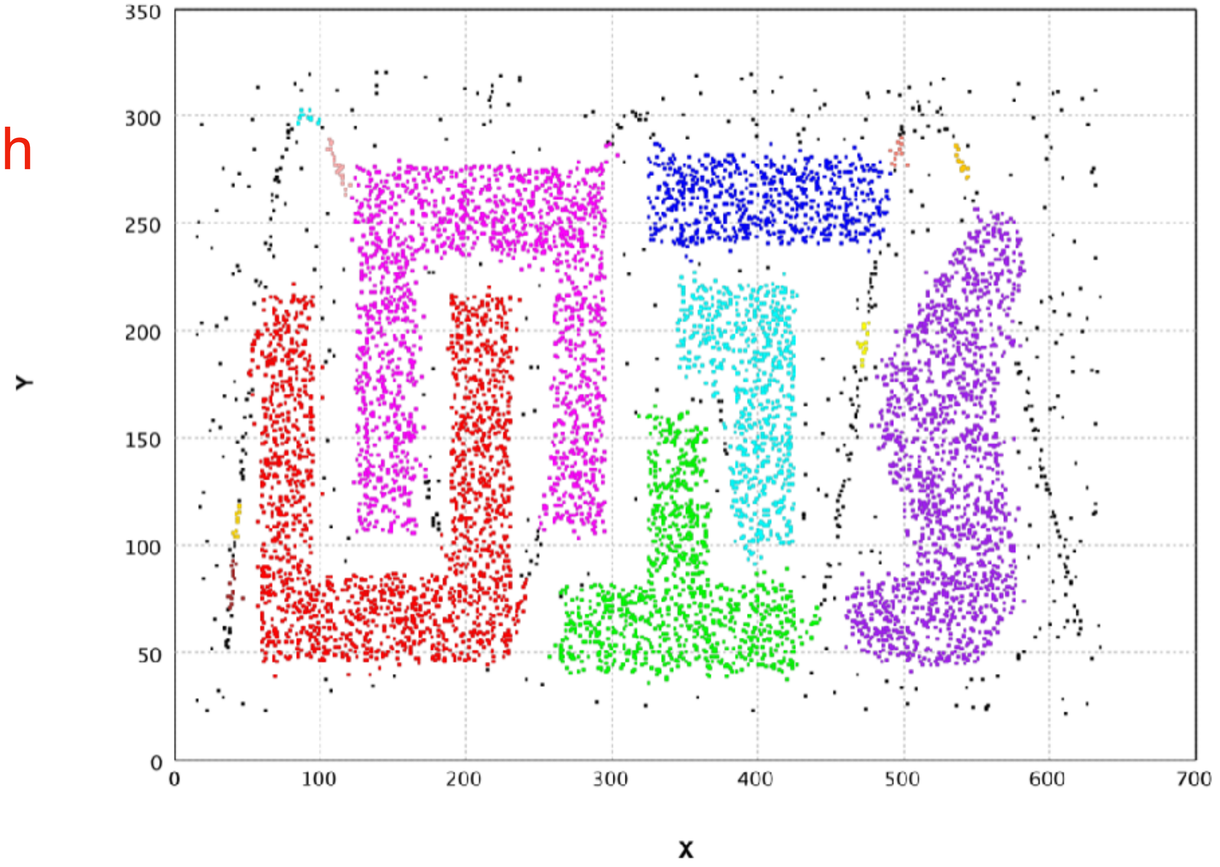
- Can handle clusters of arbitrary shape and size

- Not necessarily spatial



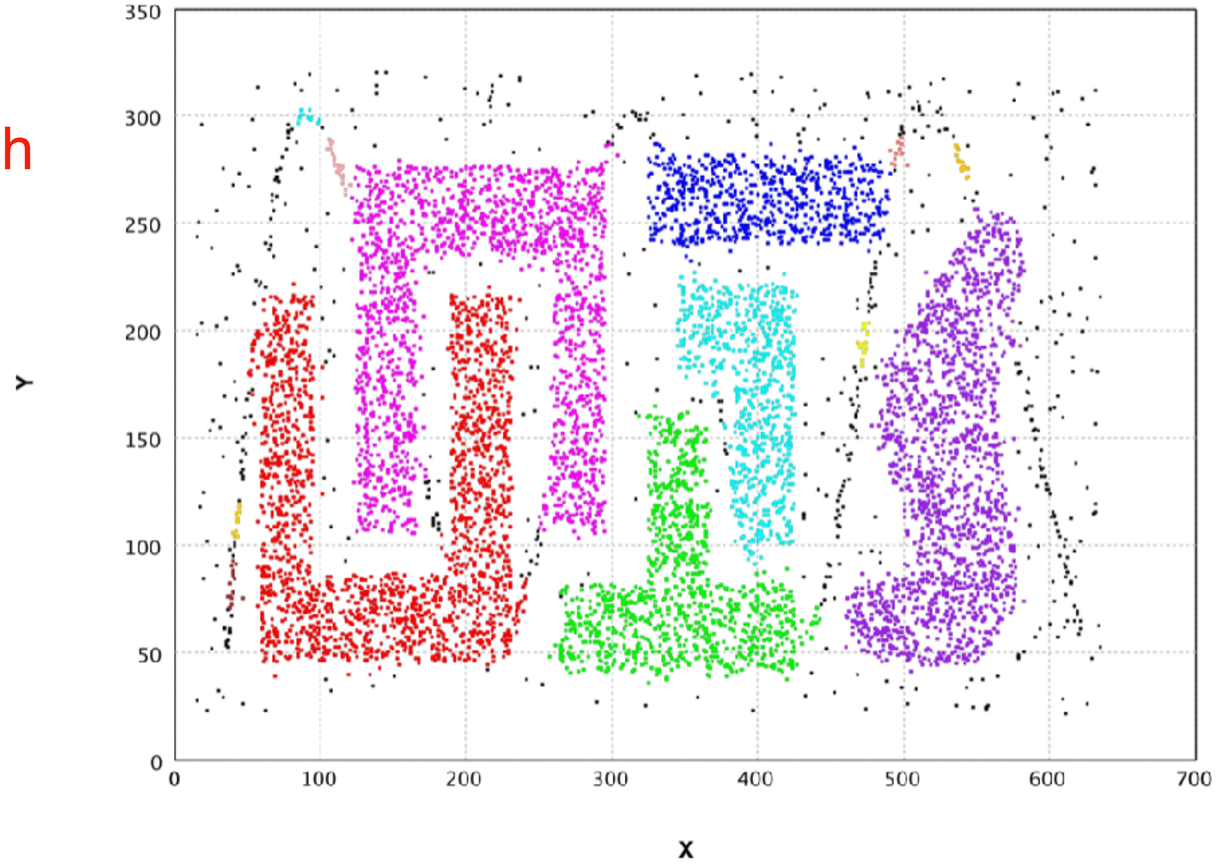
DBSCAN: Disadvantages

- Cannot handle clusters with different densities



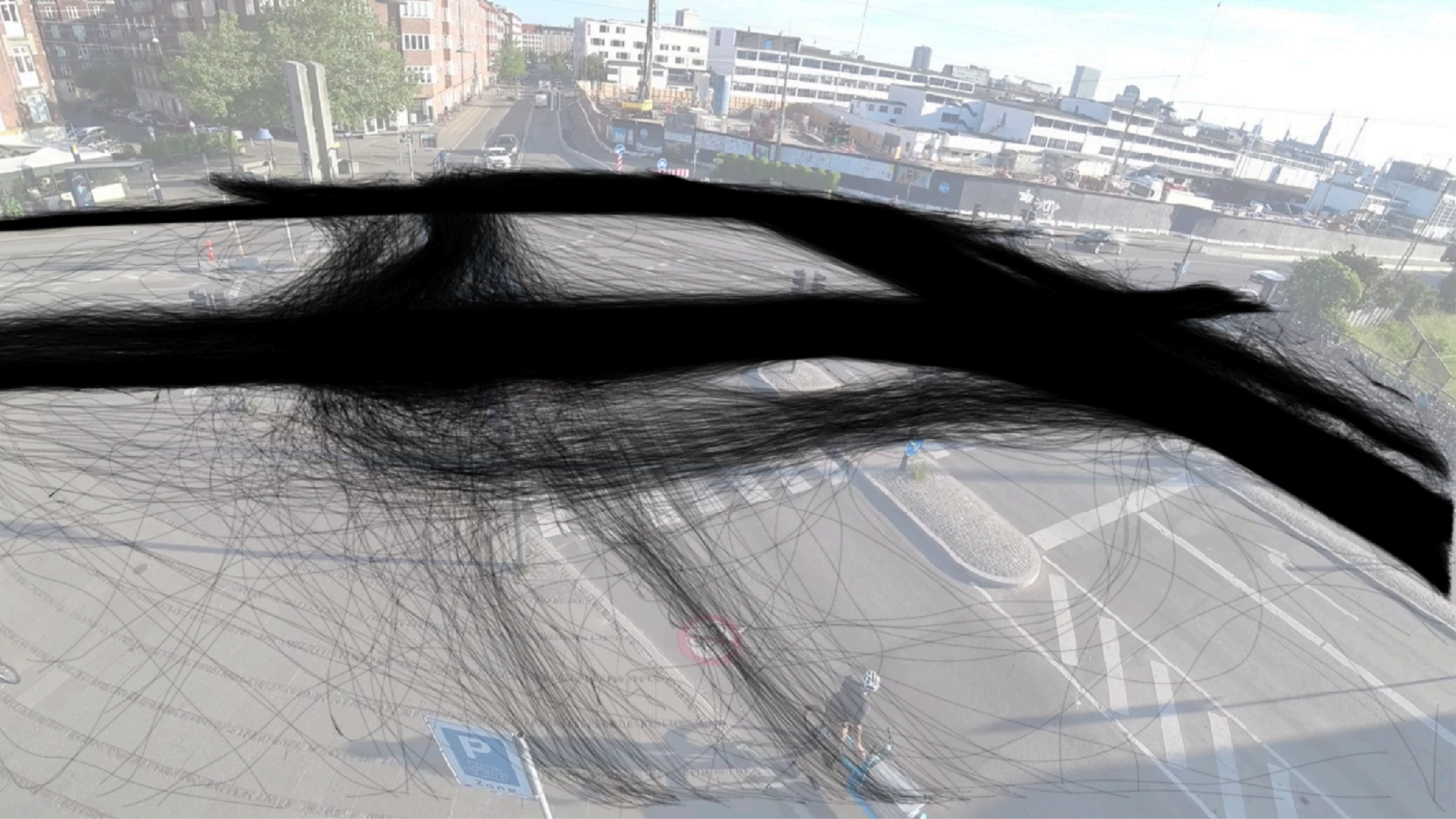
DBSCAN: Disadvantages

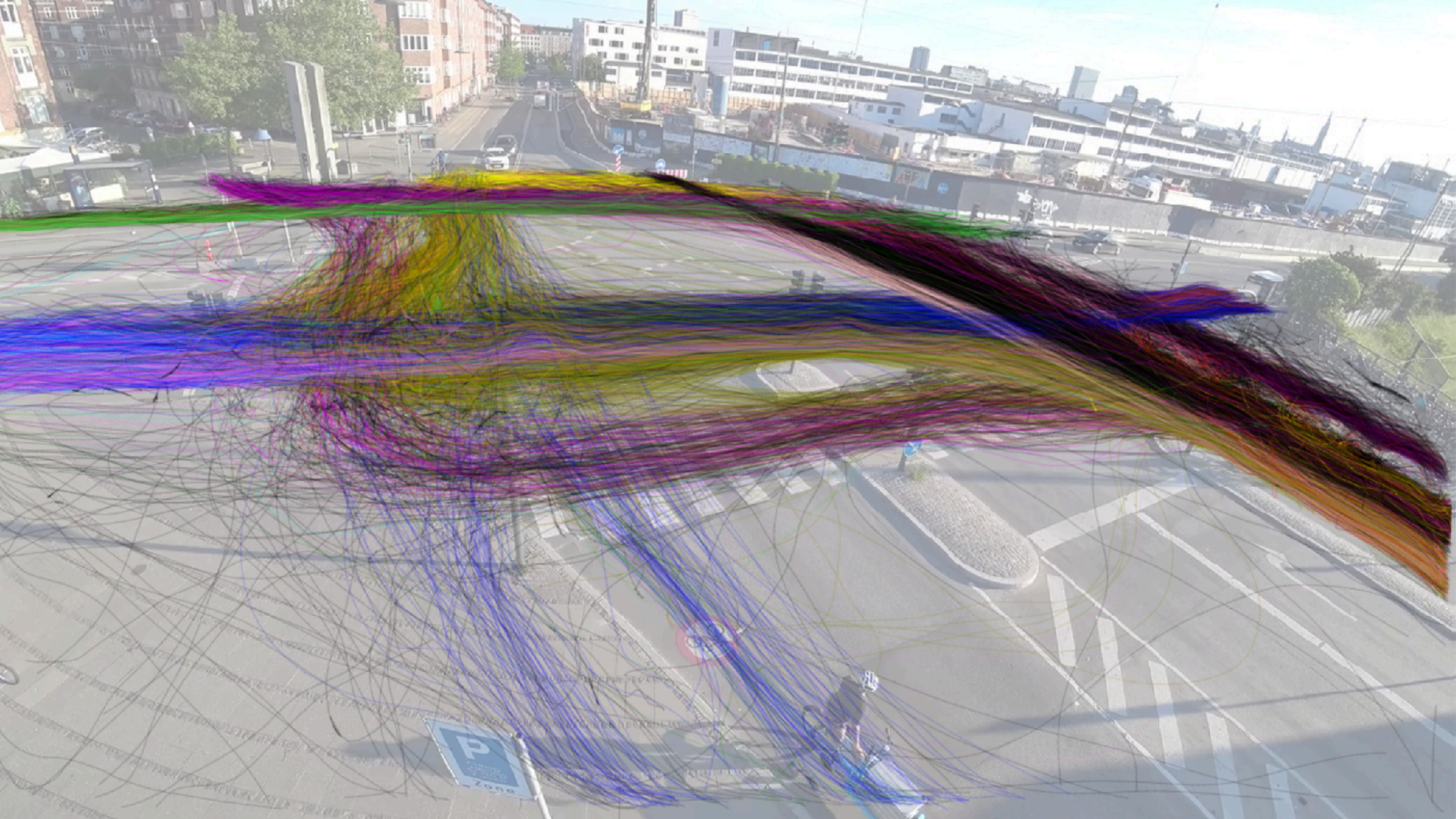
- Cannot handle clusters with different densities



- Not based on a probabilistic model like LISA: no inference, no significance

- DBSCAN can be computationally expensive $O(n^2)$, but faster than most point pattern methods





Jupyter

Sources and further materials for today's class

Geographic Data Science with Python

https://geographicdata.science/book/notebooks/08_point_pattern_analysis.html

https://darribas.org/gds_course/content/bH/concepts_H.html