

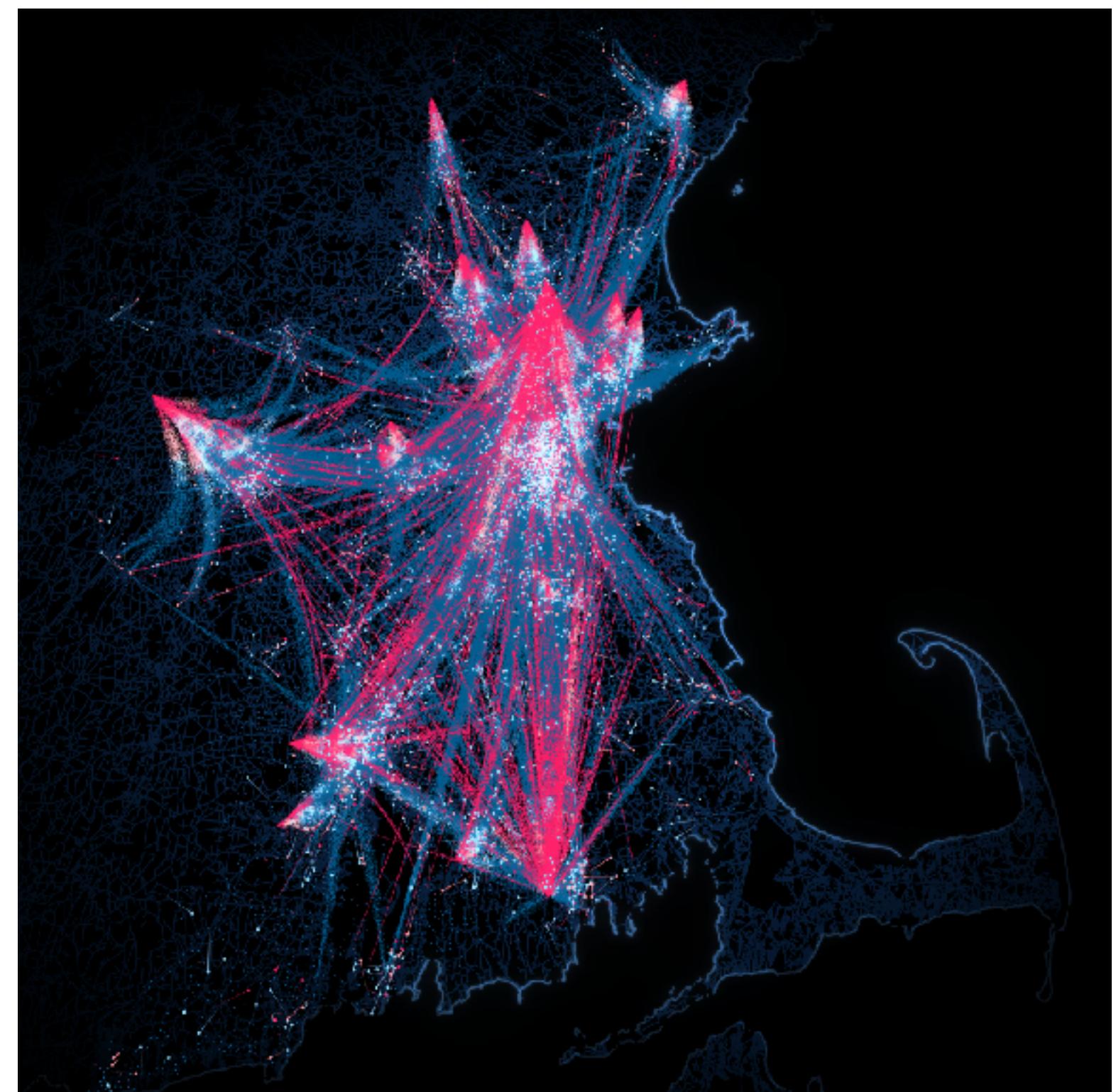
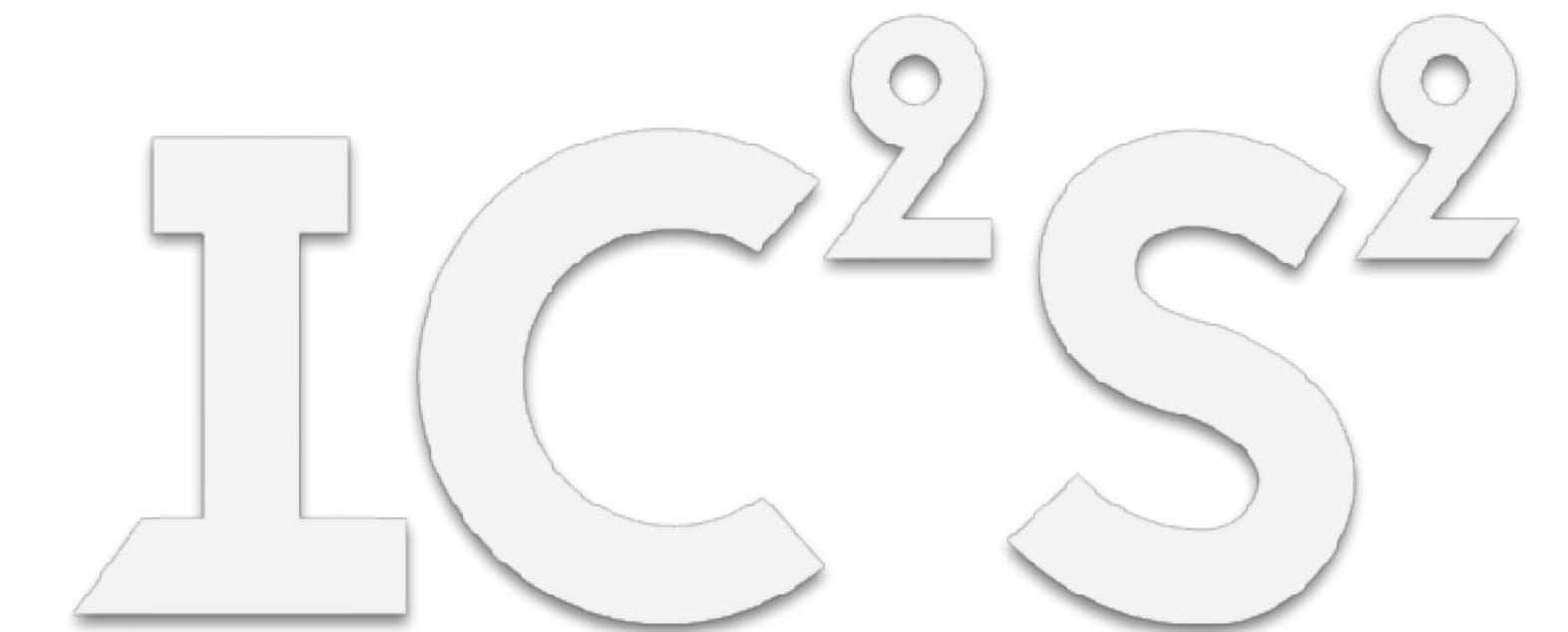
## Part 0: Introduction

Michael Szell

Ane Rahbek Vierø

Anastassia Vybornova

Jul 17, 2023



# We are NERDS, ITU ([nerds.itu.dk](http://nerds.itu.dk))



**Michael Szell**  
Assoc Prof  
Network & Data Science

[misz@itu.dk](mailto:misz@itu.dk)

[michael.szell.net](http://michael.szell.net)



**Ane Rahbek Vierø**  
PhD Student  
GIS, Urban Planning

[anev@itu.dk](mailto:anev@itu.dk)

[anerv.github.io](http://anerv.github.io)

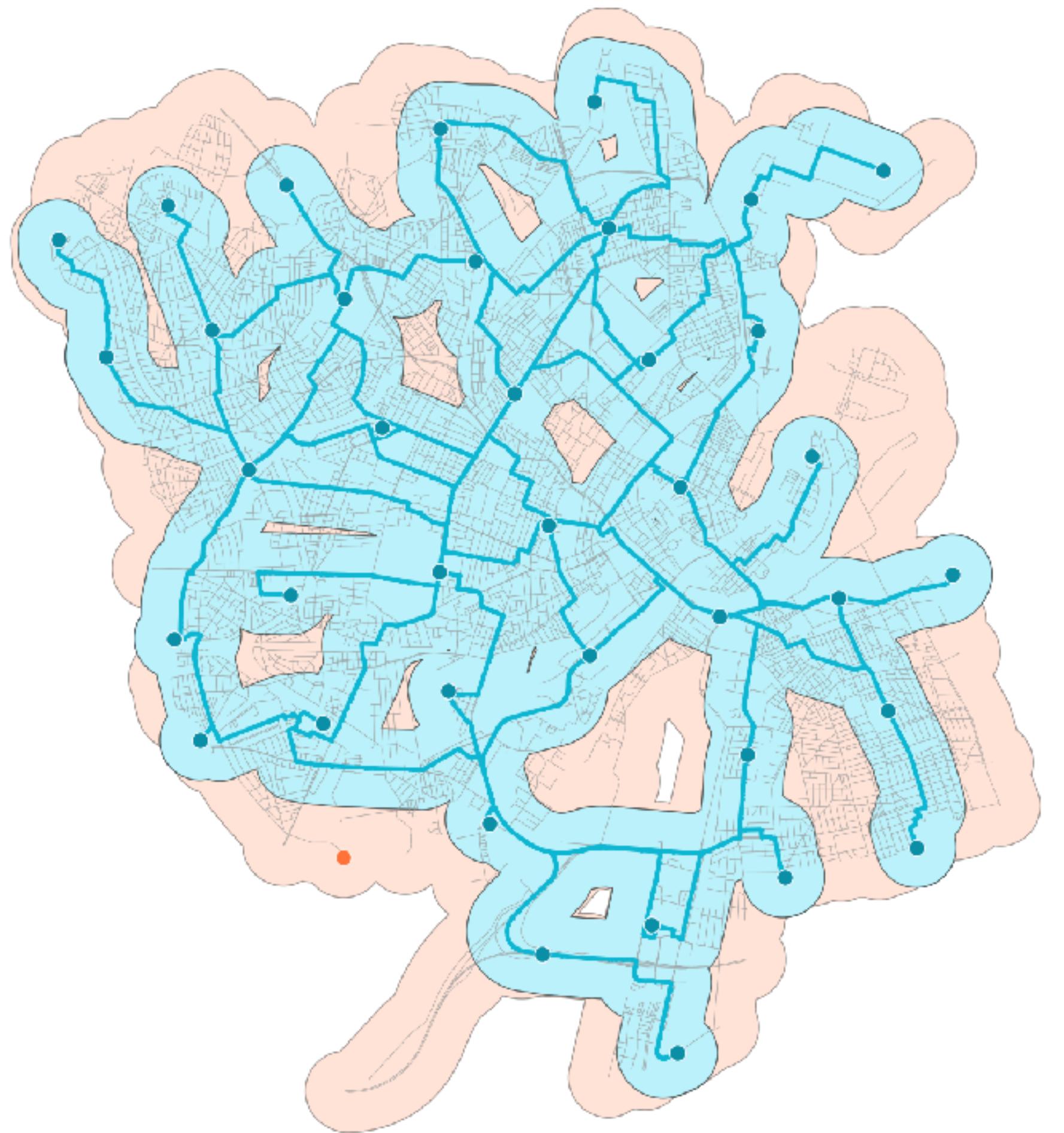


**Anastassia Vybornova**  
PhD Student  
Physics, Env Sci, Comm

[anvy@itu.dk](mailto:anvy@itu.dk)

[github.com/anastassiavybornova](http://github.com/anastassiavybornova)

# Our current research:



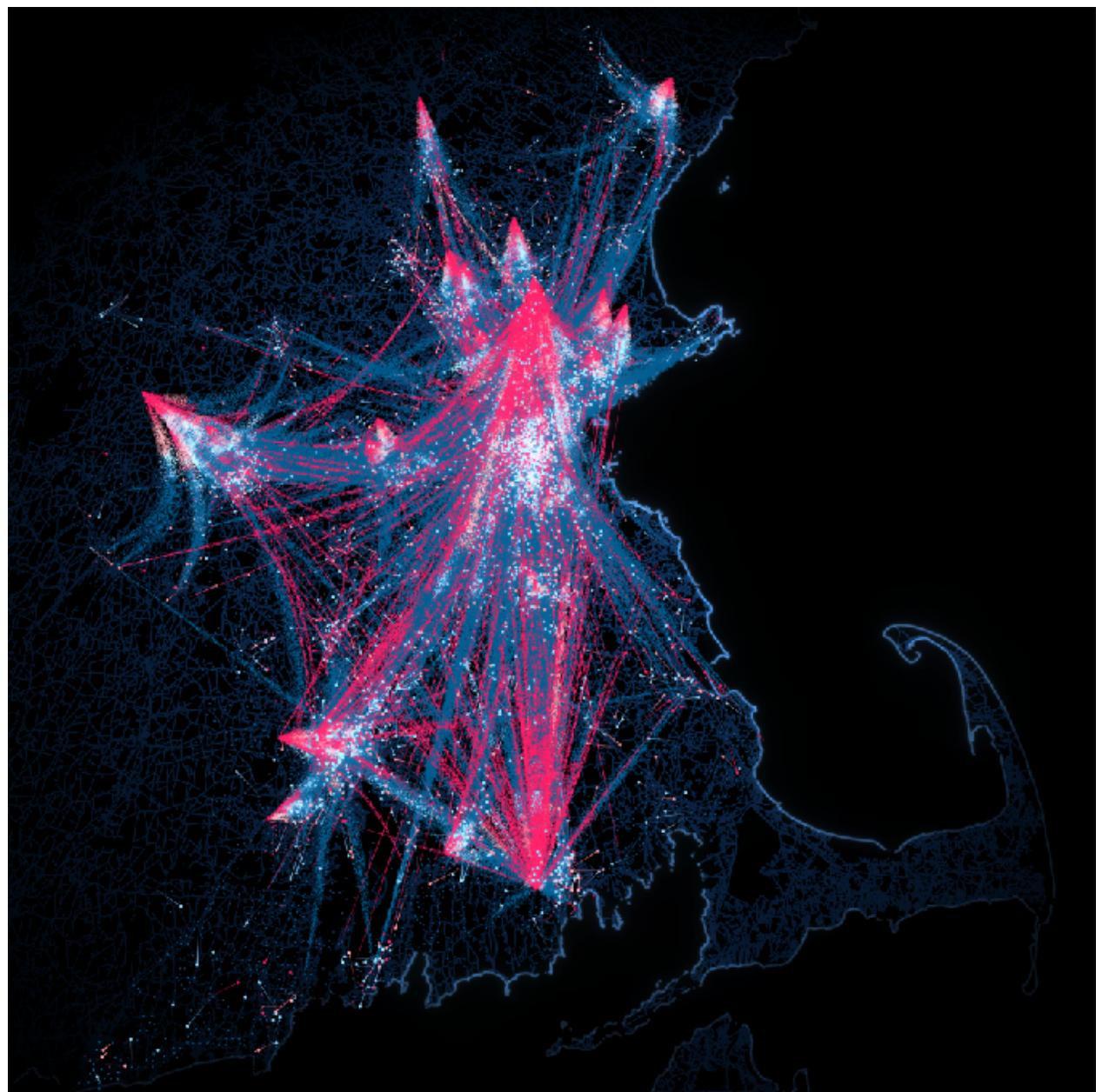
Bicycle networks  
Python



Human mobility

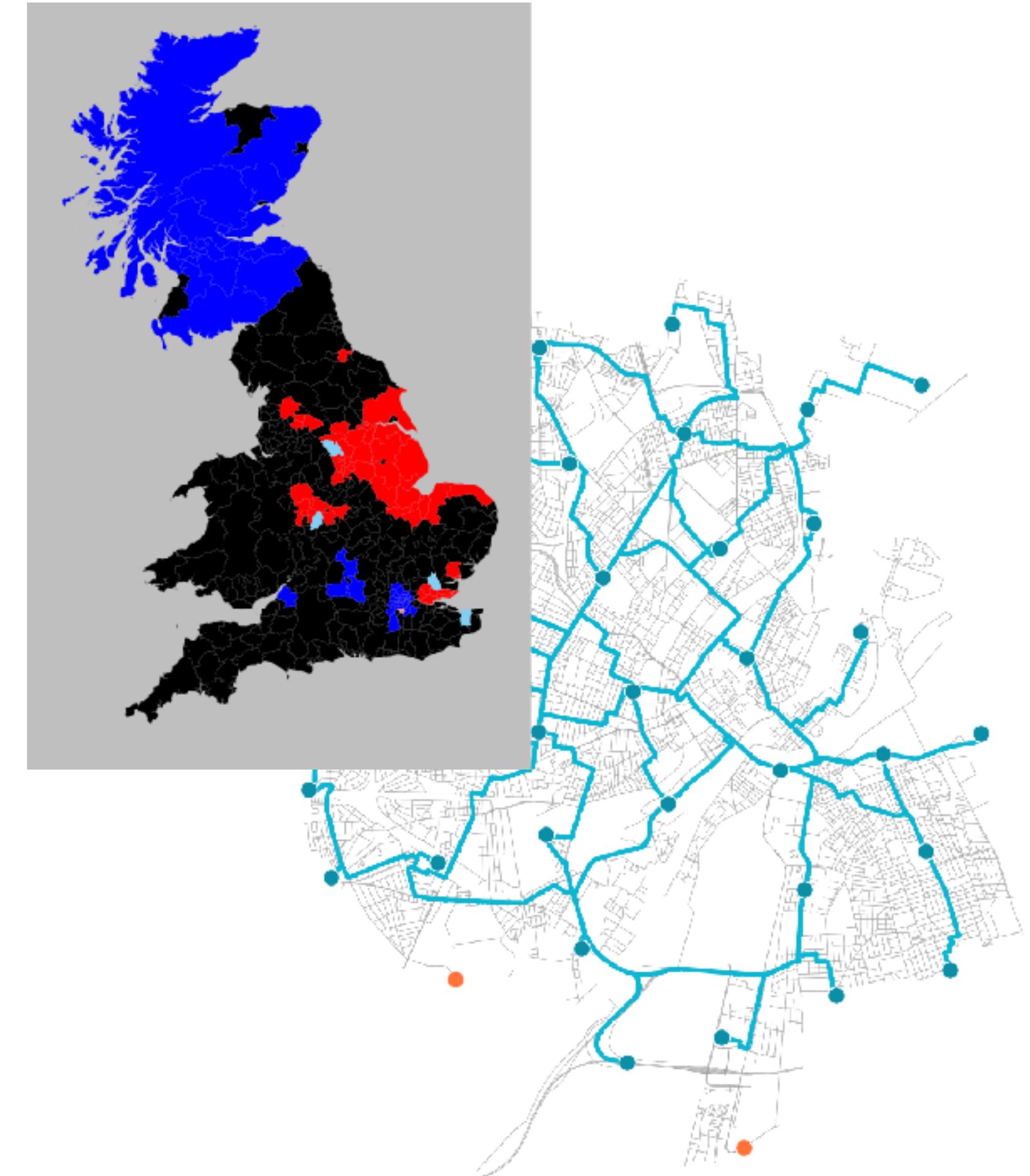
# Today you will learn about Geospatial Data Science (GDS)

What is GDS?  
Why learn about it?



Statistical tools & applications

Data & Python libraries



**Geospatial Data Science** is all the things that exist in ‘regular’ data science - but with a focus on **space** and **location**

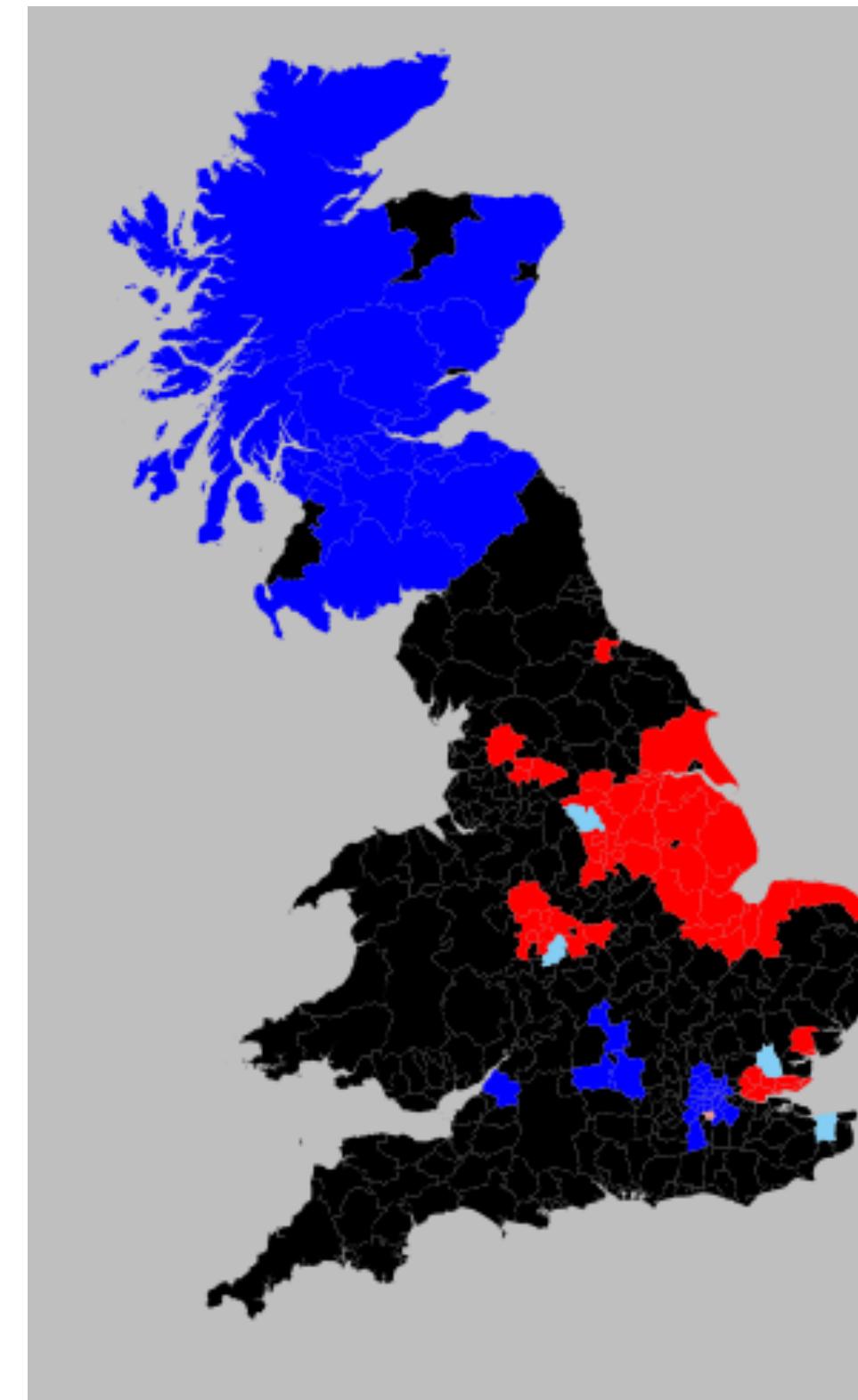
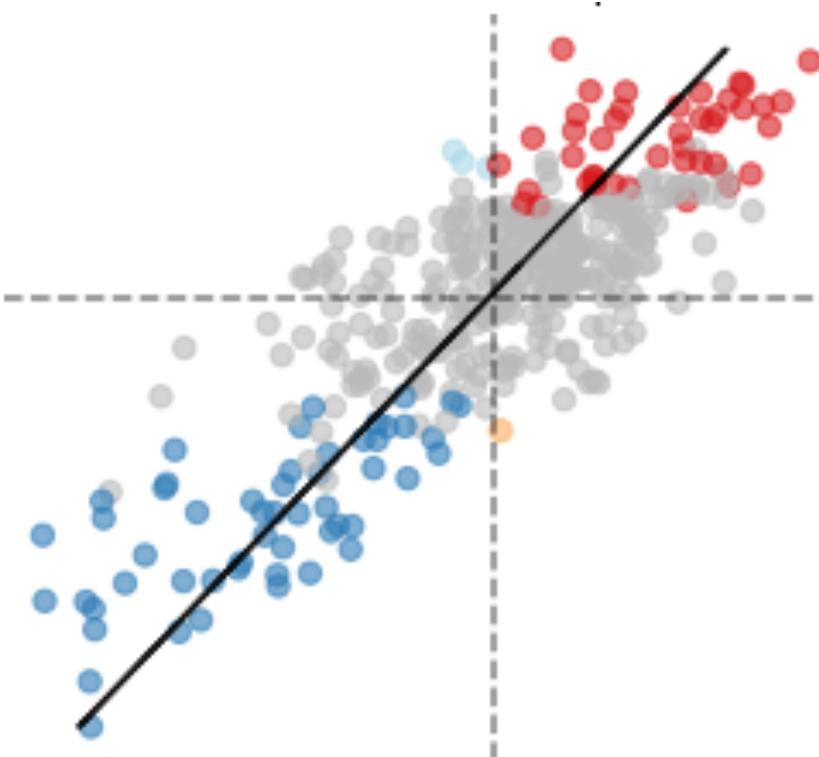
# Why is this relevant to computational social scientists?

More and more social data are spatial

We need the statistical/computational tools to correctly handle space

$$\mathbf{y}_{\text{lag}} = \left( \sum_{j=1}^n w_{ij} y_j \right)_i = W\mathbf{y}$$

$$I = \frac{n}{\sum_i \sum_j w_{ij}} \frac{\sum_i \sum_j w_{ij} z_i z_j}{\sum_i z_i^2}$$



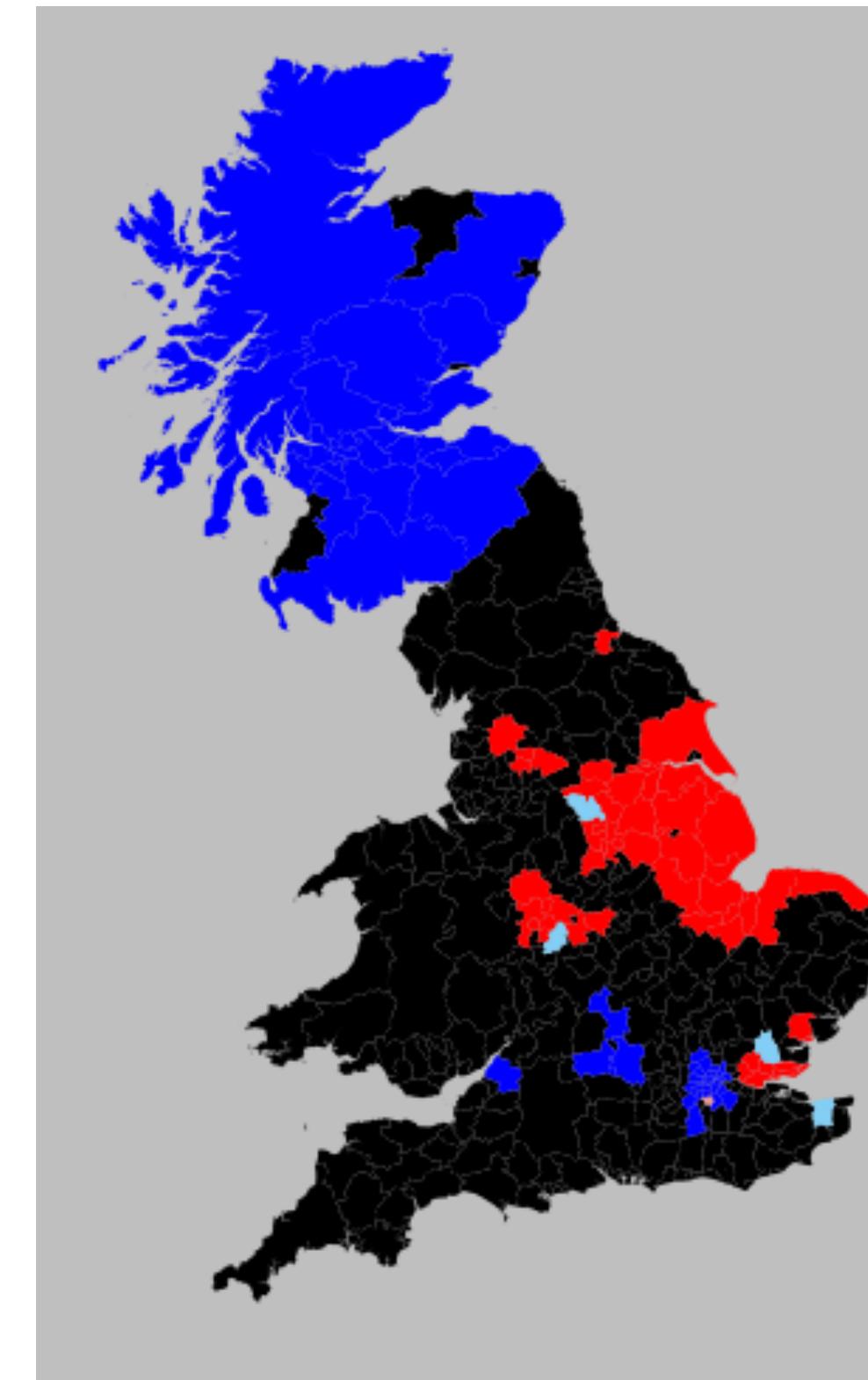
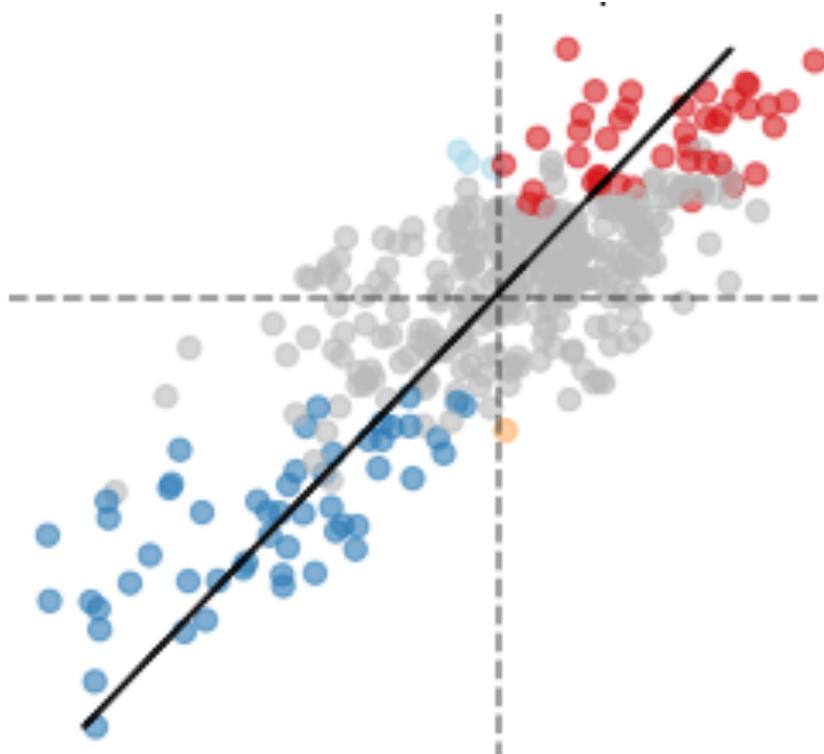
# Why is this relevant to computational social scientists?

More and more social data are spatial

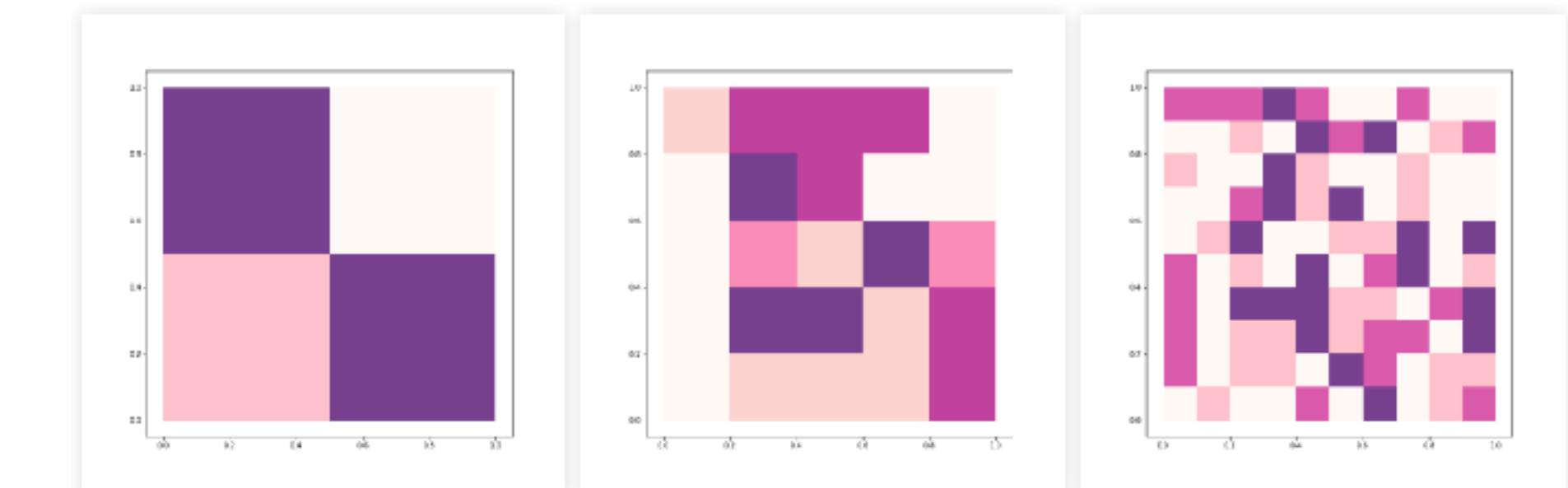
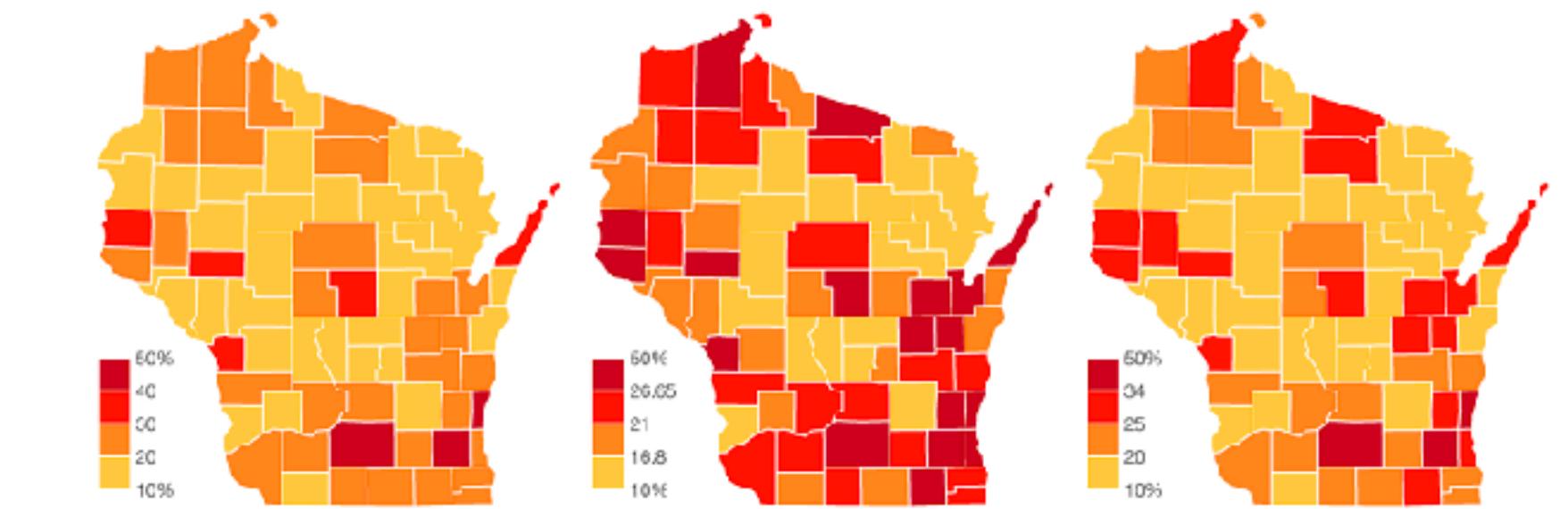
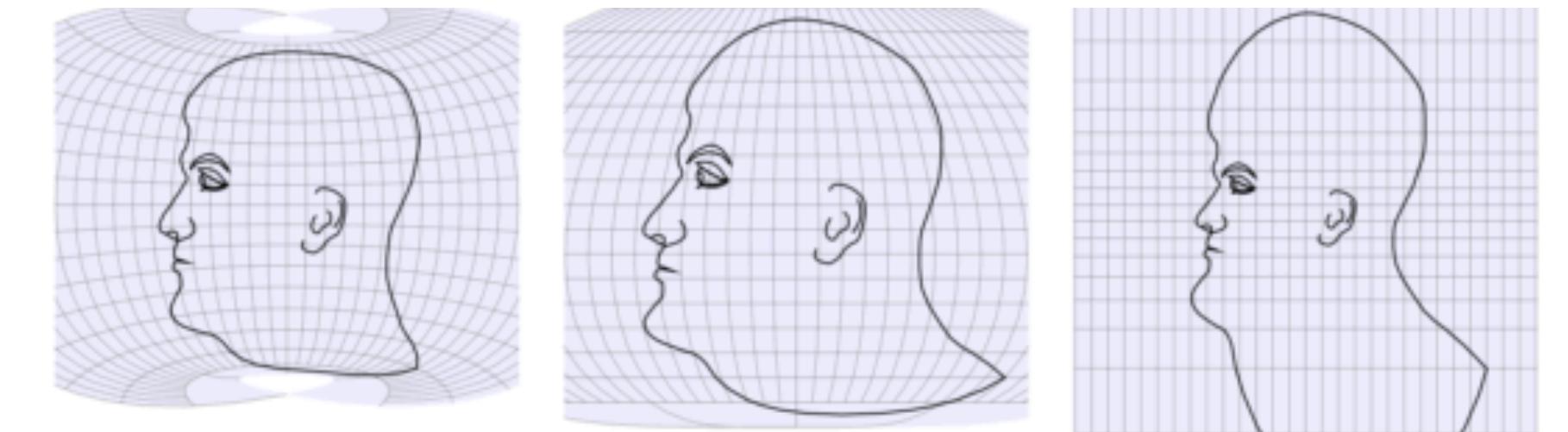
We need the statistical/computational tools to correctly handle space

$$\mathbf{y}_{\text{lag}} = \left( \sum_{j=1}^n w_{ij} y_j \right)_i = W\mathbf{y}$$

$$I = \frac{n}{\sum_i \sum_j w_{ij}} \frac{\sum_i \sum_j w_{ij} z_i z_j}{\sum_i z_i^2}$$

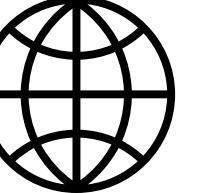
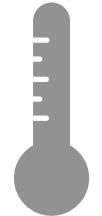


We need to be aware of pitfalls



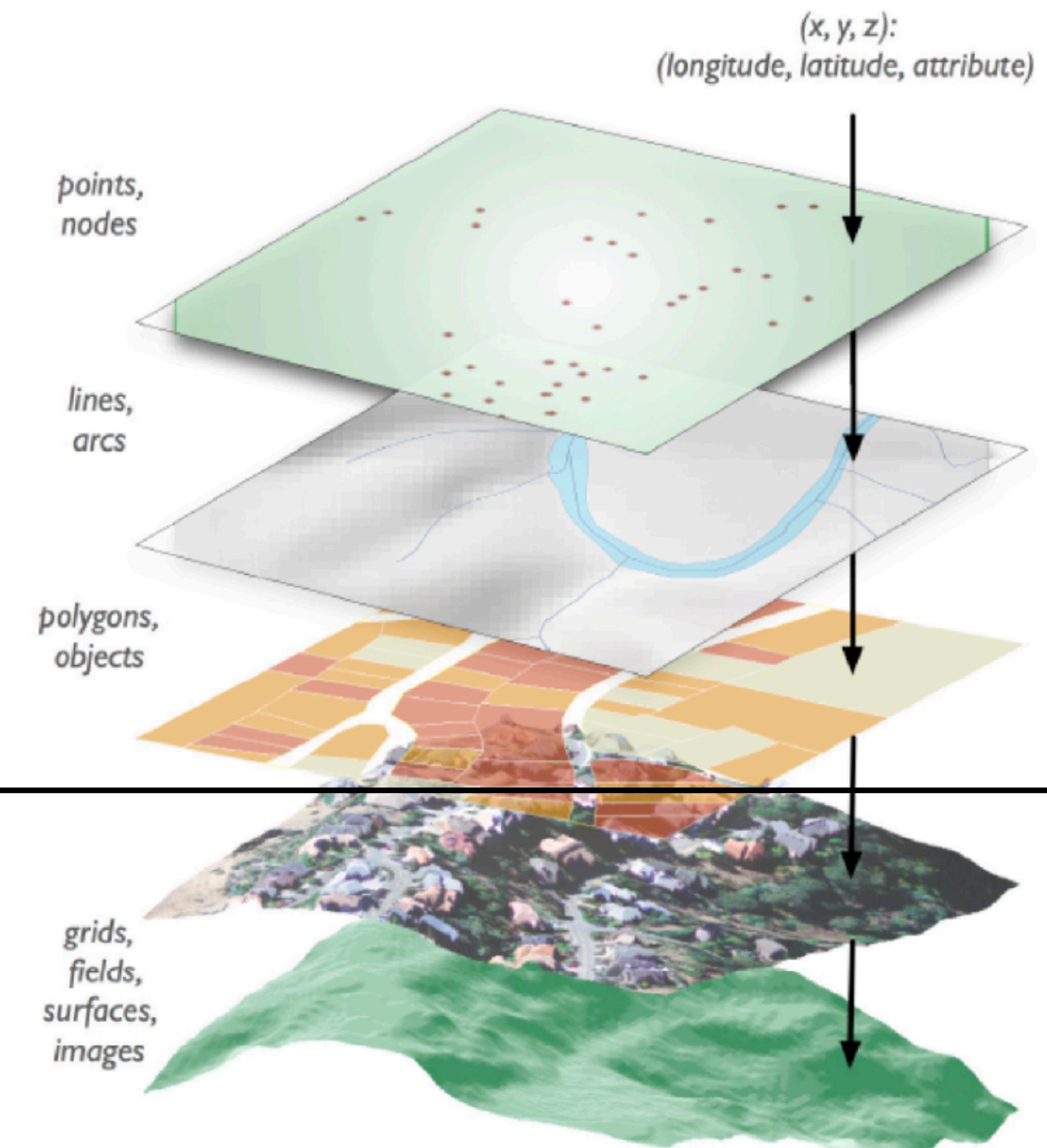
# What is Geospatial Data?

**Geospatial data** is information that describes objects, events or other features with a location on or near the surface of the earth.

- Coordinates 
- Attributes 
- Temporal information 

# There is **vector** and **raster** data

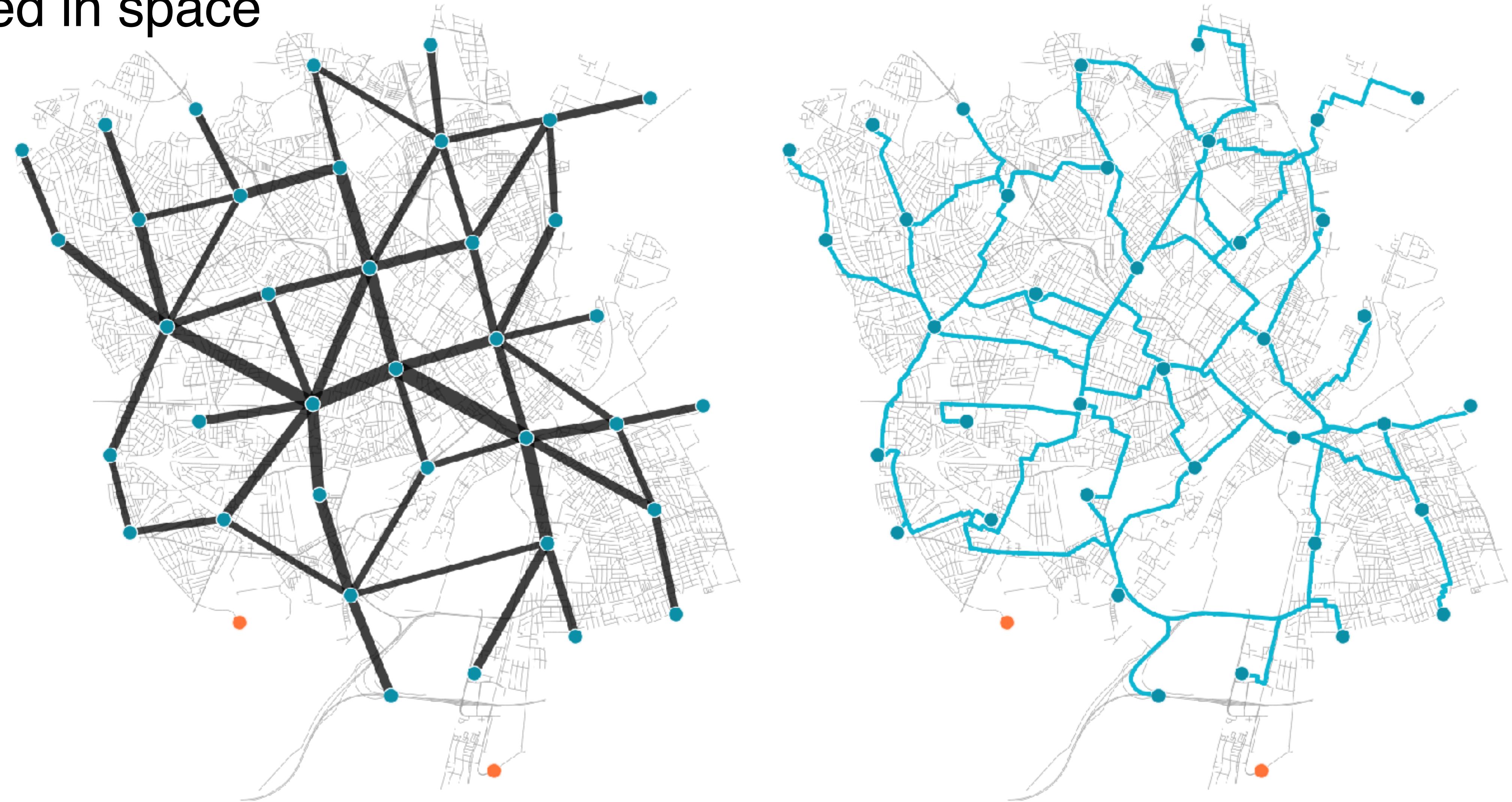
Vector: Geometric objects  
.gpkg, .shp, .svg, geojson



Raster: Grid of pixels  
.tif, .jpg, .png, .bmp

# There is **network** data

Structure (topology),  
embedded in space



GDS asks: How do things relate in space?

Everything is related to everything else,  
but near things are more related than  
distant things.

Tobler's 1st law of geography

GDS asks: How do things relate in space?

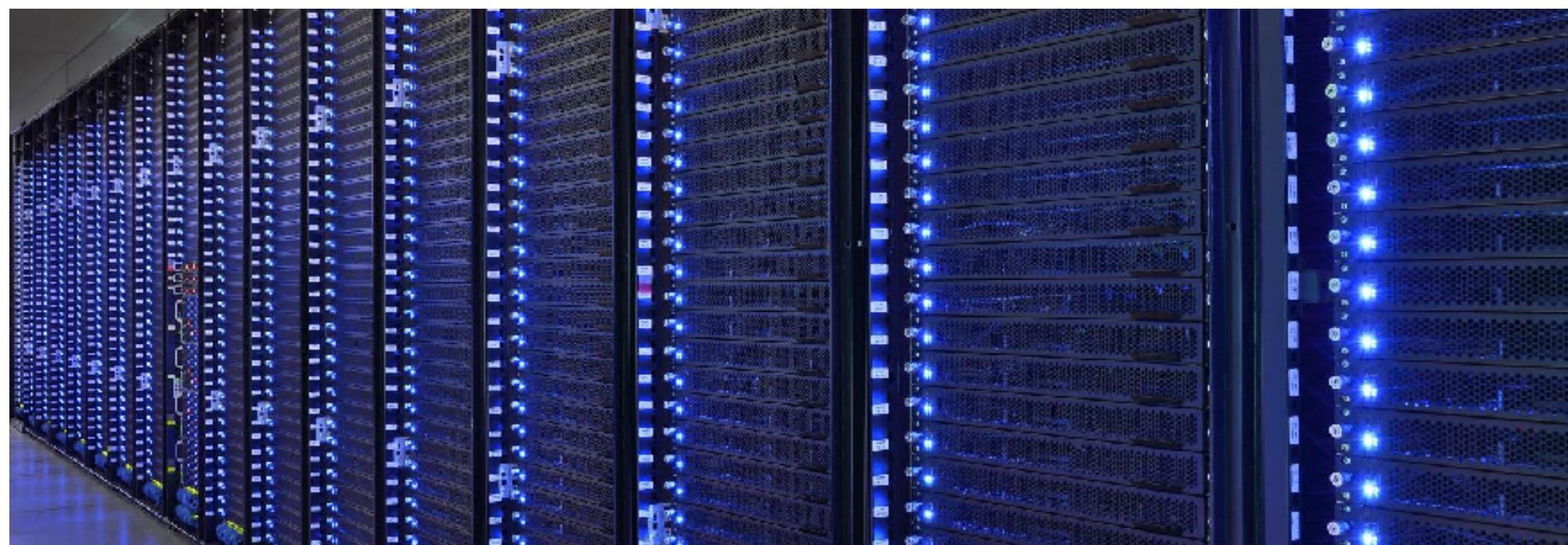
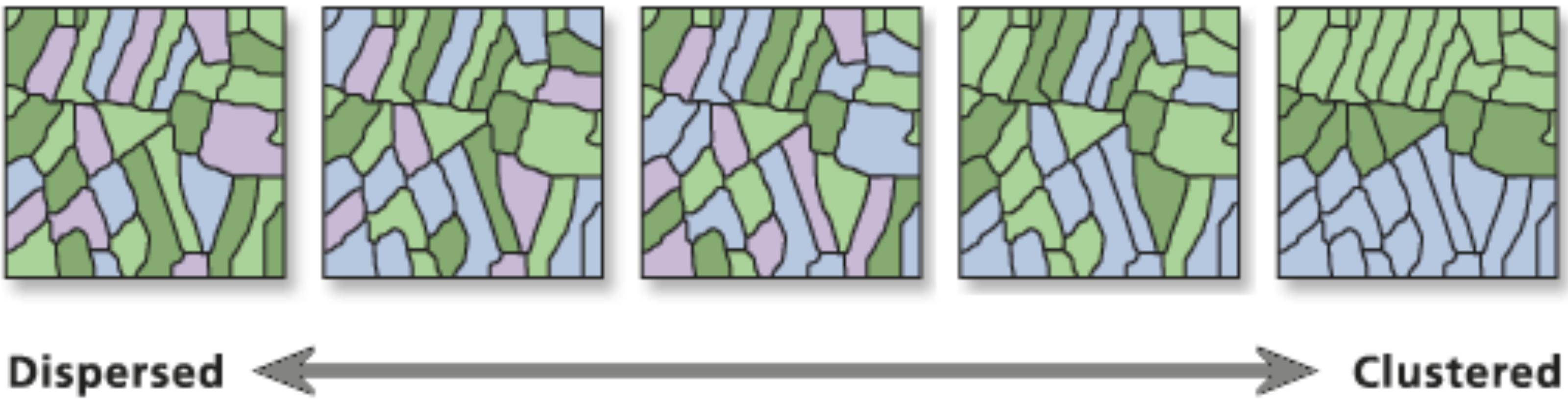
How can we formalize this question?

How can we operationalize this question?

How can we visualize this question?

# GDS is mathematical, visual, and computational

$$I = \frac{N}{W} \frac{\sum_i \sum_j w_{ij}(x_i - \bar{x})(x_j - \bar{x})}{\sum_i (x_i - \bar{x})^2}$$



# We use the Geographic Data Science platform gds\_env



## Preparations

*Before the tutorial,* install `gds_py` (version 9.0) from `gds_env` :

[https://darribas.org/gds\\_env/stacks/gds\\_py/#install](https://darribas.org/gds_env/stacks/gds_py/#install)



<https://github.com/NERDSITU/gdstutorial>

[https://darribas.org/gds\\_env/](https://darribas.org/gds_env/)

# Tutorial schedule

July 17, 2023	Part	Topic
09:00 - 09:05	0. Introduction	Introduction
09:05 - 09:40	1. Data Handling	Data & Geometry, CRS, Libraries
09:40 - 09:50		<i>10 min break</i>
09:50 - 10:20	2. Spatial Statistics	Choropleth Maps, Spatial Autocorrelation
10:30 - 11:00		<i>30 min break</i>
11:00 - 11:35	3. OpenStreetMap	Introduction to OSM & OSMnx
11:35 - 11:45		<i>10 min break</i>
11:45 - 12:30	4. Spatial Networks	Spatial Networks with Geopandas

<https://github.com/NERDSITU/gdstutorial>

# We will only cover the procedural part of GDS (in Python)

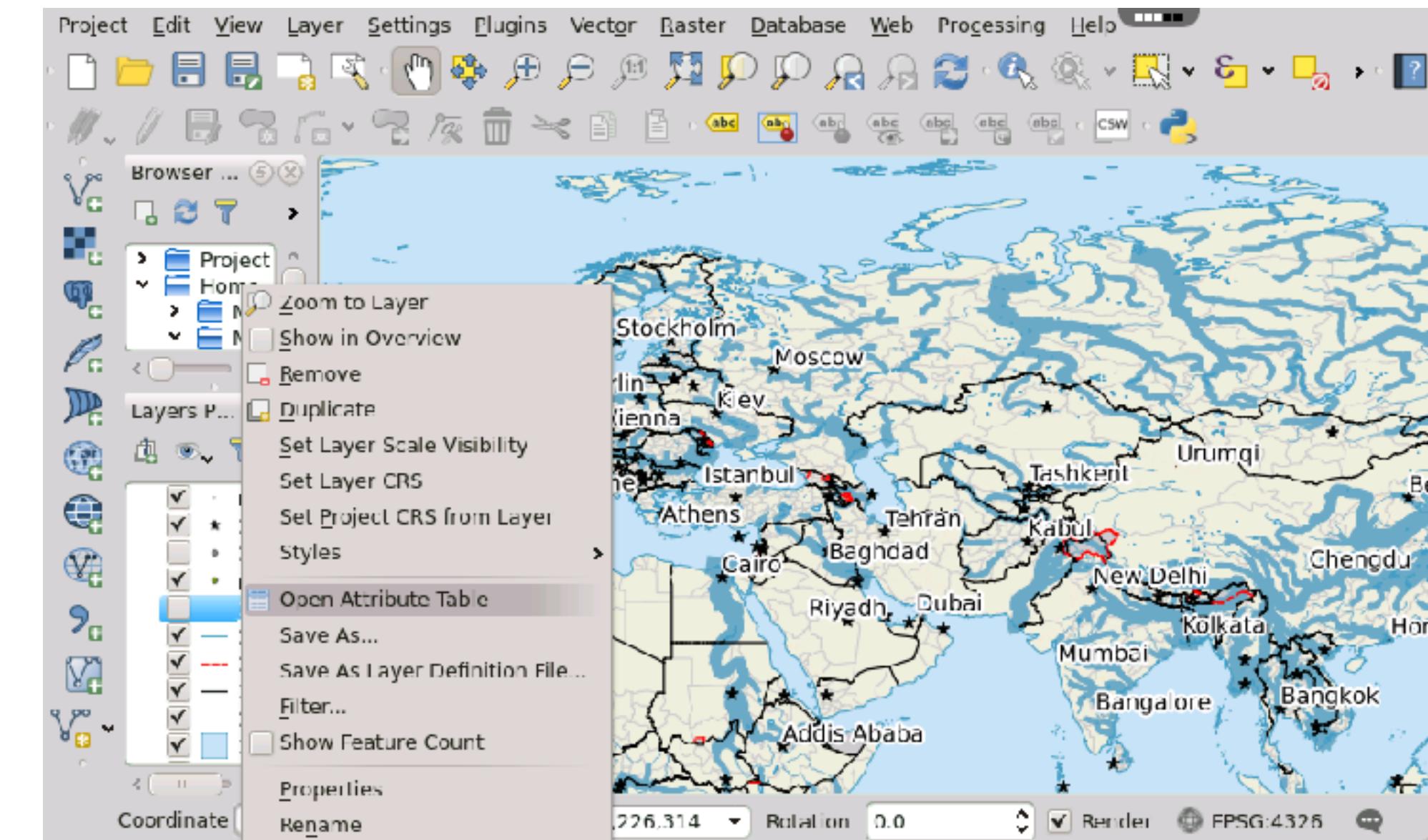
## DB handling



## Procedural

```
31     def __init__(self, file=None, fingerprints=None, logduplicates=False, debug=False, logger=logging.getLogger(__name__)):
32         self.file = file
33         self.fingerprints = set()
34         self.logduplicates = logduplicates
35         self.debug = debug
36         self.logger = logging.getLogger(__name__)
37         if path:
38             self.file = open(path, 'ab')
39             self.file.seek(0)
40             self.fingerprints.update(fingerprint for fingerprint in self.file)
41
42     @classmethod
43     def from_settings(cls, settings):
44         debug = settings.getbool('supersecret.debug')
45         return cls(job_dir(settings), debug)
46
47     def request_seen(self, request):
48         fp = self.request_fingerprint(request)
49         if fp in self.fingerprints:
50             return True
51         self.fingerprints.add(fp)
52         if self.file:
53             self.file.write(fp + os.linesep)
54
55     def request_fingerprint(self, request):
56         return request_fingerprint(request)
```

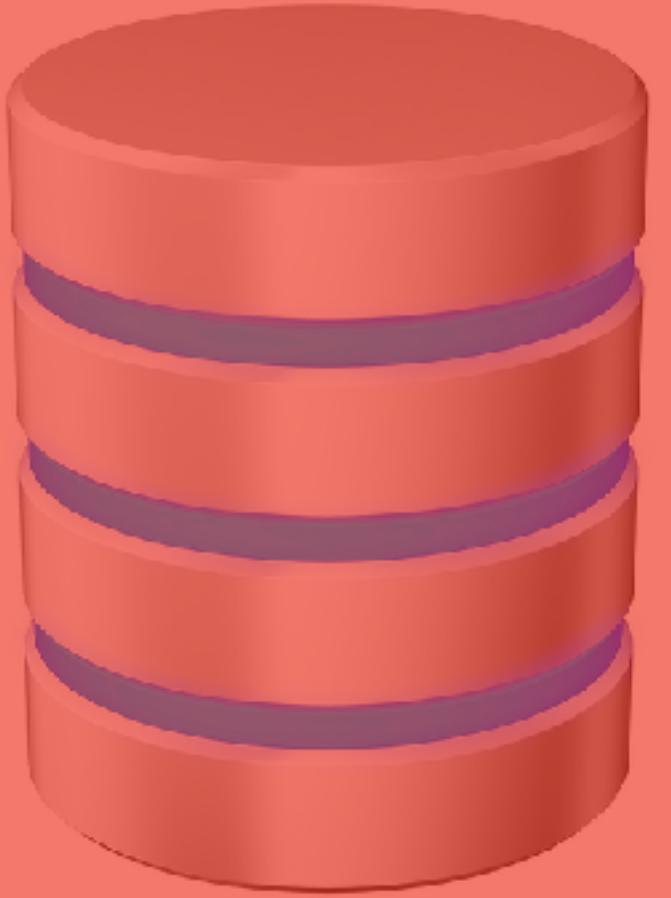
## Point & Click GIS



ArcGIS

# We will only cover the procedural part of GDS (in Python)

## DB handling

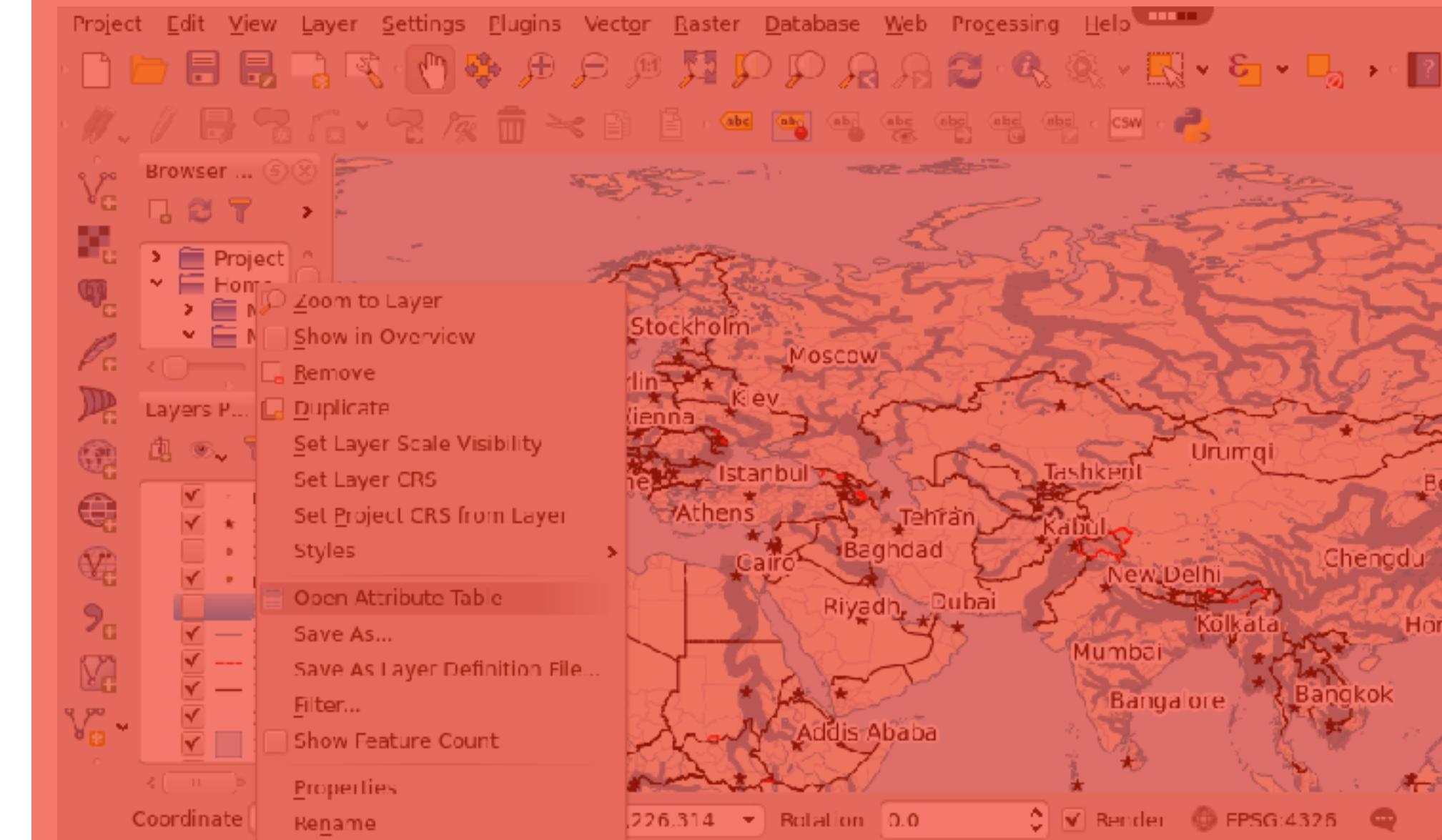


## Procedural

```
31     def __init__(self, settings):
32         self.file = None
33         self.fingerprints = set()
34         self.logduplicates = True
35         self.debug = debug
36         self.logger = logging.getLogger(__name__)
37         if path:
38             self.file = open(path, 'ab')
39             self.file.seek(0)
40             self.fingerprints.update(fp for fp in self.file)
41
42     @classmethod
43     def from_settings(cls, settings):
44         debug = settings.getbool('superuser.debug')
45         return cls(job_dir(settings), debug)
46
47     def request_seen(self, request):
48         fp = self.request_fingerprint(request)
49         if fp in self.fingerprints:
50             return True
51         self.fingerprints.add(fp)
52         if self.file:
53             self.file.write(fp + os.linesep)
54
55     def request_fingerprint(self, request):
56         return request_fingerprint(request)
```



## Point & Click GIS



ArcGIS