# Designing Distributed Geospatial Data-Intensive Applications

Ph.D. Course, 2022

## Instructors:

Prof. Luca Foschini, Associate Professor &

Dr. Isam Mashhour Al Jawarneh, Postdoctoral Research Fellow

{isam.aljawarneh3, Luca.foschini}@unibo.it

Department of Computer Science and Engineering (DISI), Università di Bologna

# Part 2

Designing highly efficient geospatial data-intensive solutions

22nd July 2022

# Introduction to spatial data

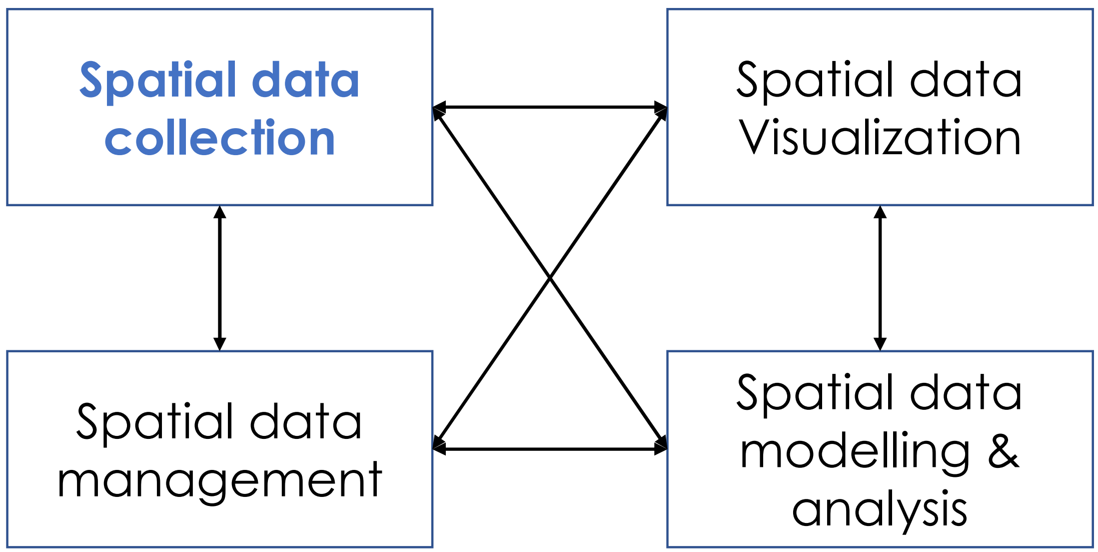
# What is spatial data

- A **spatial object** is an element for **modelling** world data into information systems (specifically **GIS**)
  - A digital **representation** of **geographical entity** or **phenomenon**
  - Defined by **spatial data**
  - **points**, **lines** and **areas**
  - Points are the primary element in GIS
    - All other objects are represented by **series of points**

# Tasks in Geographic Information Systems (GIS)

- A geographic information system (**GIS**) is a fusion of computer hardware and **software** for
  - Collecting
  - Managing
  - Analyzing
  - Displaying

  **geo-referenced** data (**geospatial**, **spatial**)



# Distributed Spatial data management

- We need extensions to the existing parallel DBMSs (**data models** & **query languages**) to be able to manage **geometrical** objects:
  - Specialized **data structures** & **indexing** methods
  - Geometrical **computation** algorithms & query **optimizers**
- A parallel spatial DBMS provides additional functionalities for dealing with spatial data (**geodata**), supporting **spatial data types** in its model & language
  - **Point**, **polygon**, **line**, etc.,
- Efficient spatial **indexing** & **join** are key elements

# Spatial data collection

## 1) Ground surveying

- Land **surveying**
  - Surveyors determine the positions of locations by triangulating from the position of known locations
- **GPS**: vehicle, phone, etc.,
- **Geocoding**
  - Attaching a **geographic** location to some sort of address information, such as a house address or zip code
  - Some form of database of addresses whose locations are precisely known
  - Unlocated addresses are matched to these known addresses in the database
- **Surveys**
  - Attributed information, and determining the **location** requires **geocoding** (e.g., **surveyor's GPS**)
  - Equip cars with **GPS receivers**, drive around recording pictures of their surroundings
- **Sensors**
  - Climate stations
    - Measuring temperature, air pressure, and precipitation
    - **GPS** specifies the **locations** of these **sensors**, or through **geocoding**

## Spatial data collection

### 2) Remote sensing

- Collecting data at a distance (far from the objects),
- e.g., plane, satellite or drone pictures

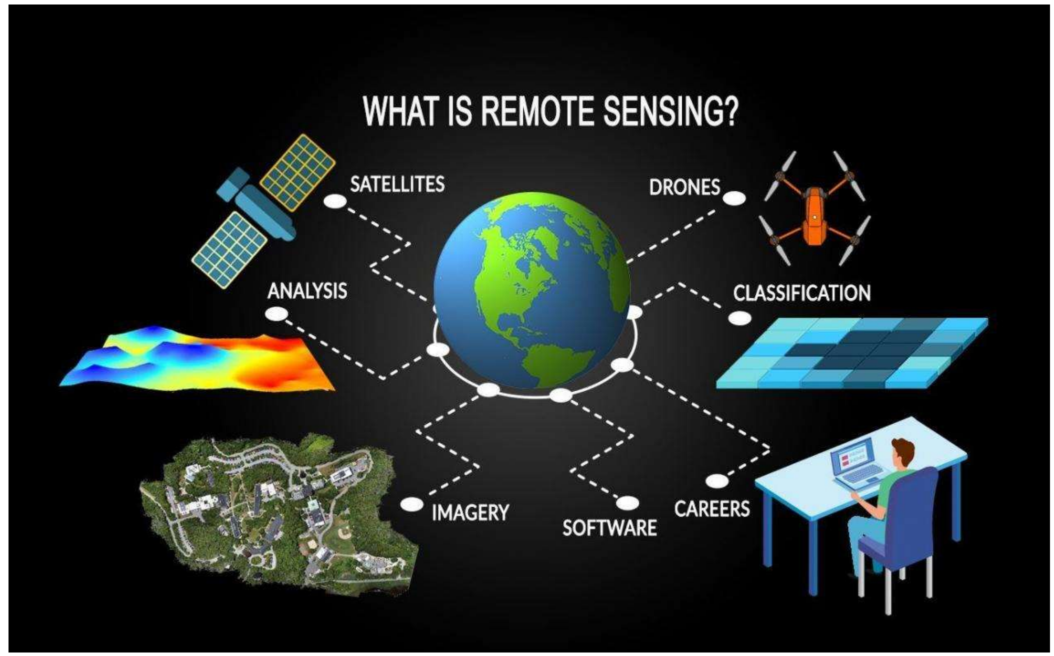



Image source

# Global positioning systems (GPS)

- Constellations of satellites that **orbit** the Earth
  - Satellites transmit signals to the earth's surface that indicate their **position** in space
  - Device equipped with an appropriate **GPS receiver** can interpret these signals and determine the device's location on the earth
  - Every mobile has a **GPS receivers**
    - Easy to record, or tag, the **location** where a picture was taken or **track** daily movements without special expertise.
    - Voluminous **spatial data** is collected daily

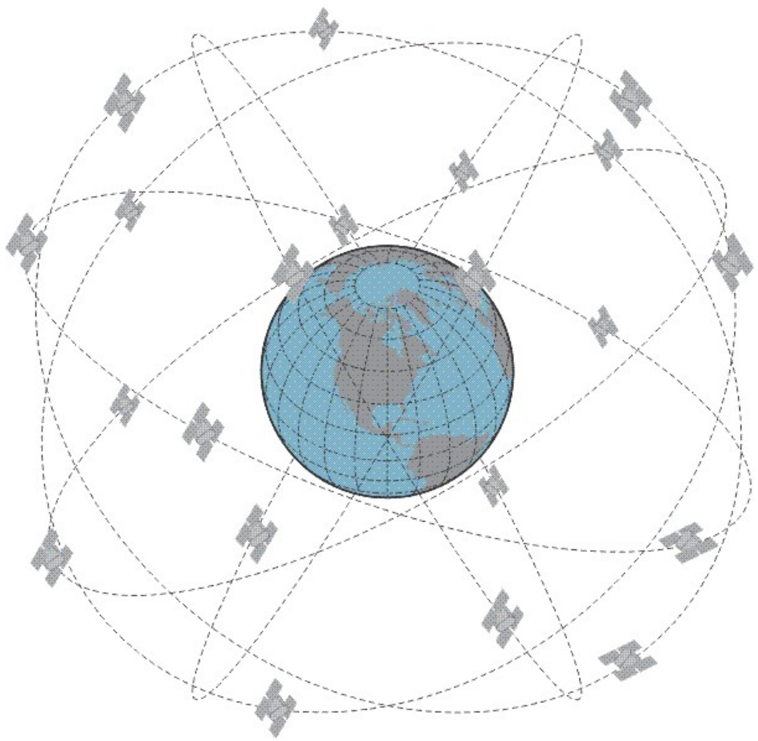
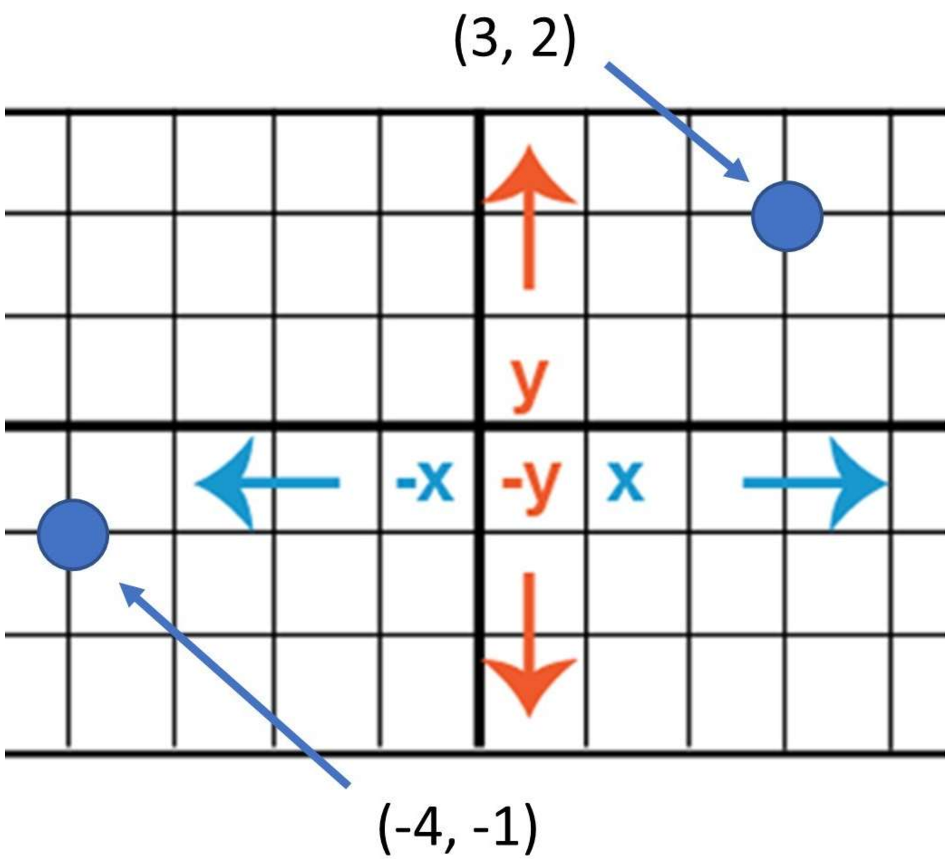


Image source

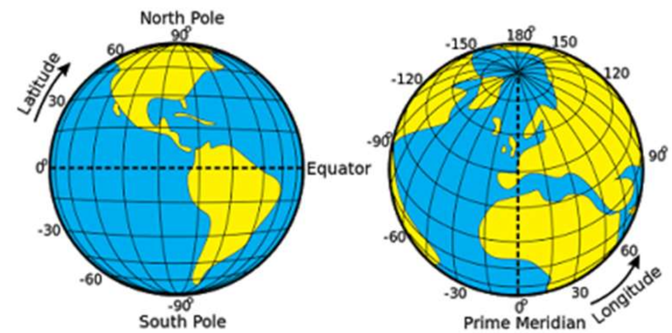
# Coordinates & Projection

- **Locations** on the earth's surface are measured in terms of ***coordinates*** (Cartesian coordinate system),
  - A set of two or more numbers that specifies a **location** in relation to some **reference system**
  - Grid formed by putting together two measurement scales, one **horizontal** (x) and one **vertical** (y)
  - The point at which both x and y equal zero is called the ***origin*** of the **coordinate system**



# Geographic coordinate system

- Define **positions** on the Earth's roughly-spherical surface
  - Uses an east-west scale, called **longitude** that ranges from +180° to -180°.
  - The north-south scale, called **latitude**, ranges from +90° (or 90° N) at the North pole to -90° (or 90° S) at the South pole
  - In simple terms, **longitude** specifies positions **east** and **west** and latitude specifies positions **north** and **south**



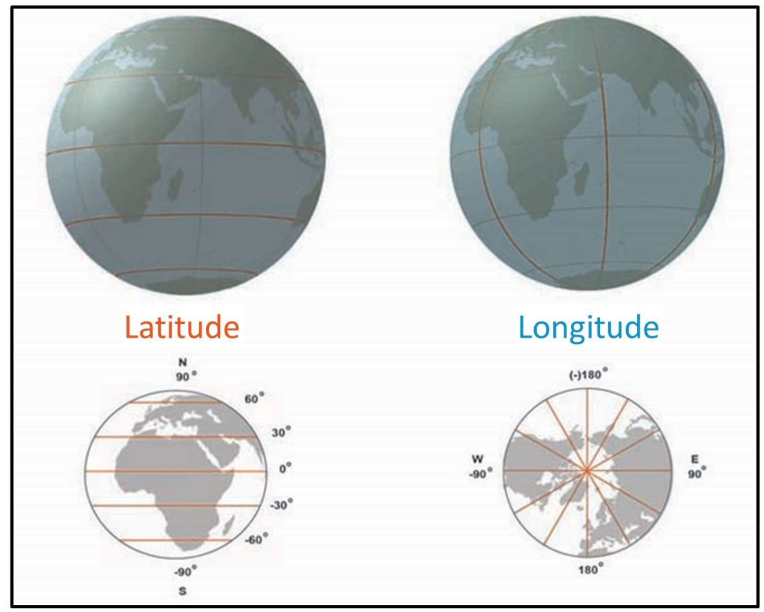
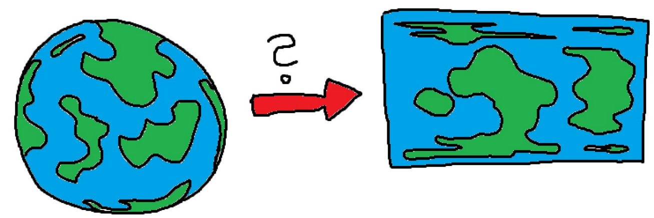




Image source

# Map Projections

- **Representation** of spatial objects need to be obtained
  - **Transforming** objects from **real geometries** into map objects (**representative** objects)
    - Normally a reduced-scale generalized model
- **Projection**: turning a **three-dimensional** globe into a **two-dimensional** map.
- How do we go from three-dimensional **graticule** to two-dimensional **geographic coordinates**
  - the process of how objects on a 3-dimensional surface (the earth) come to be represented on a flat piece of paper or computer screen
  - Our emphasis will be on the properties that different **projections** distort or maintain – area, shape, and distance



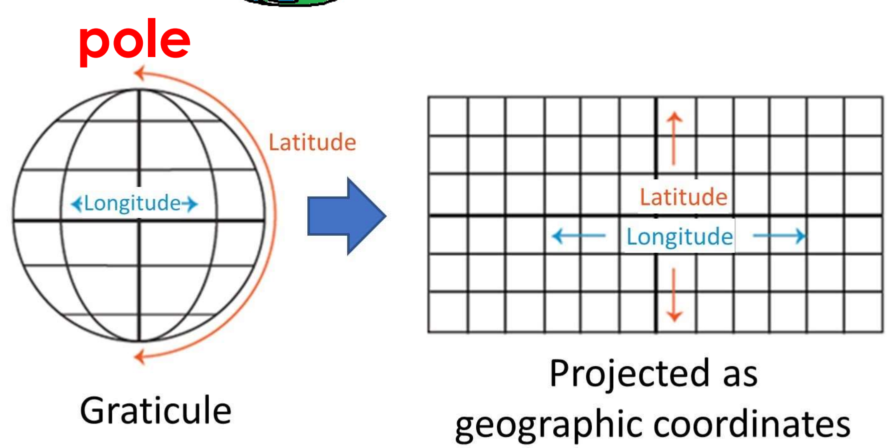




Image source

# Map Projections

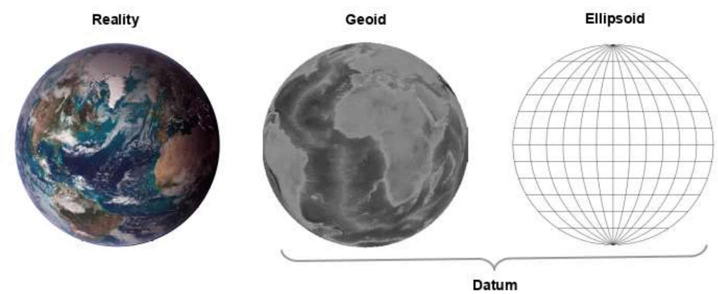
- Steps for representing the three-dimensional world as two-dimensional visualizations

(1) Lumpy surface of Earth is represented with an approximate & simplified representation called **geoid**

(2) The result is a reference system that is known as a geodetic **datum**, used as a reference for longitude and latitude degrees

(3) Datum defines **geographic coordinate system** of latitudes and longitudes that indicates where locations are on the surface of the planet

(4) GCS is **transformed** from 3-D latitude/longitude coordinates to a projected 2-D coordinate system (PCS) composed of X and Y locations corresponding to those of the GCS counterparts



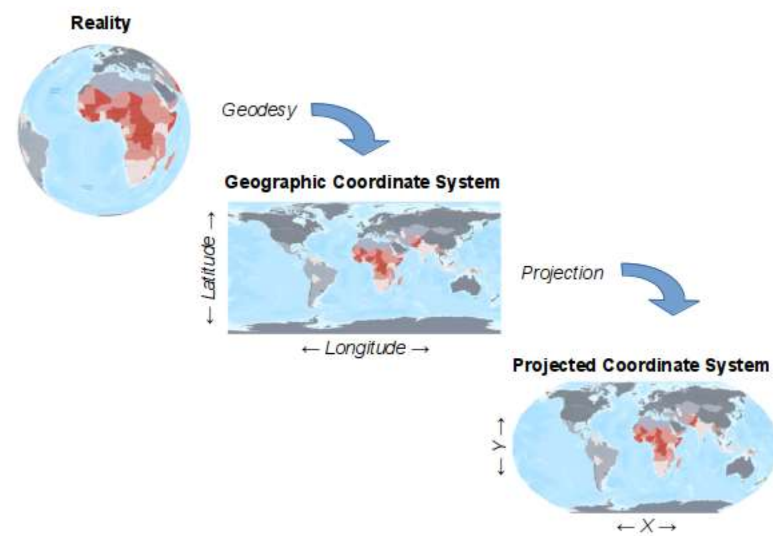
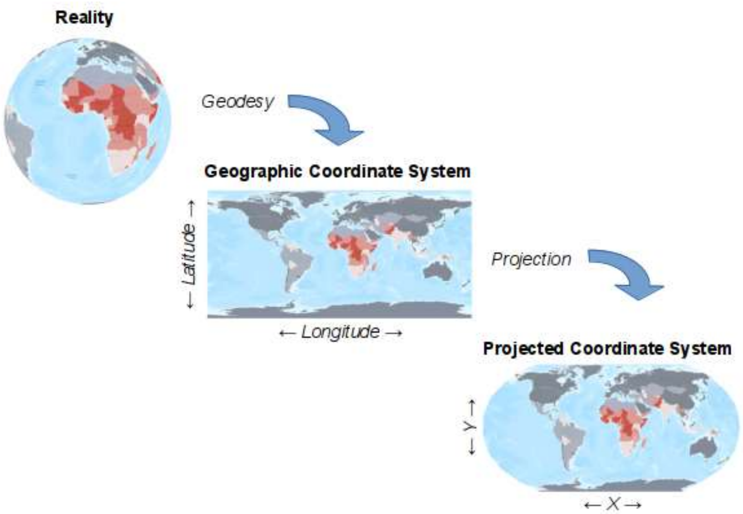




Image source

## GCS & PCS

- A **GCS** defines *where* the data is located on the earth's surface.
  - Define locations on a model of the surface of the earth. The **GCS** uses a network of imaginary lines (**longitude** and **latitude**) to define locations. This network is called a graticule




- A projected coordinate (PCS) tells the data *how* to draw on a **flat surface**, like on a **paper map** or a **computer screen**
  - *A* **projected coordinate system (PCS)** *is a GCS that has been flattened using a map projection.*
  - Maps are **flat**, so your map *must* have a PCS in order to know how to draw

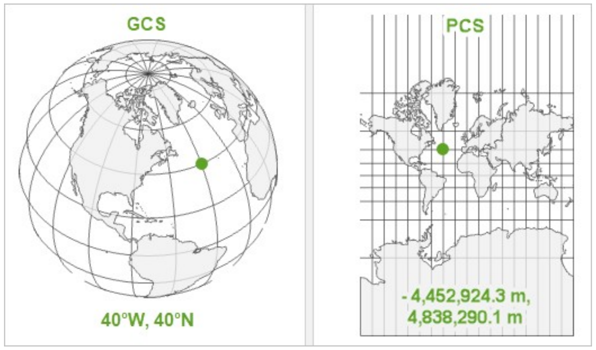
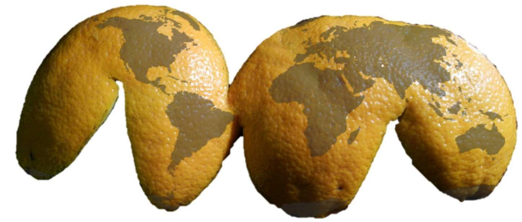
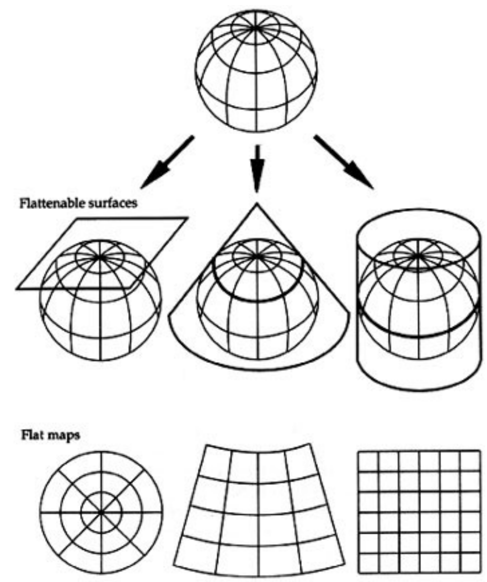



Image source

## Commonly Used Map Projections

- **Projections** deal with the methods and challenges around turning a **three-dimensional** (and sort of lumpy) earth into a **two-dimensional** map
    - The process is accomplished by a direct geometric **projection** or by a **mathematically** derived **transformation**
- Transformation from **3-D** to **2-D**
    - Earth flattening
        - Cone
        - Plane (azimuthal projection)
        - Cylinder

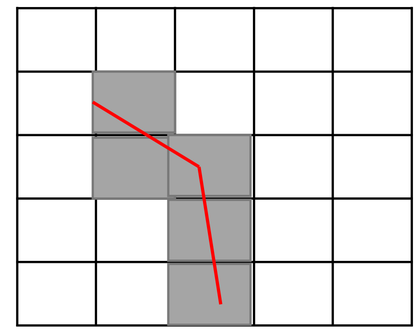


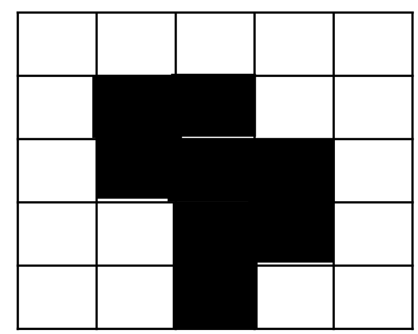


# Spatial data models

- Raster data model
  - Non-overlapping polygons (pixels) to represent spatial objects
- **Points**
- **Lines** & **areas**
  - A sequence of adjacent connected pixels
  - Line: all pixels where part of the line passes

Line

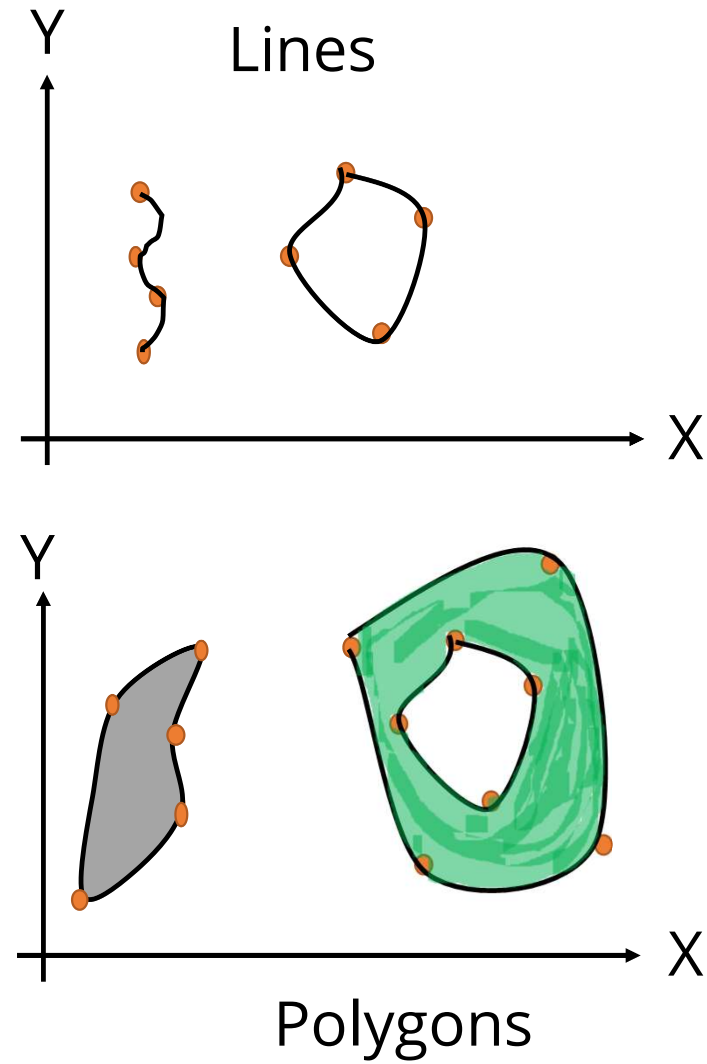




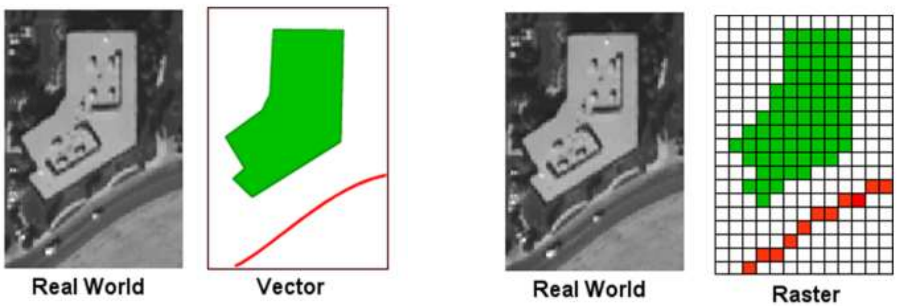
Area

# Spatial data models

- Vector data model
  - Needs a **Cartesian coordinate system** (e.g., perpendicular x, y) with **Euclidean** metrics
- Point is the core element
- **Lines** & **areas**
  - Sequence of points
    - Non-closed OR closed with no inner boundaries → line
    - Closed & boundaries → polygon
- **Loss** of **accuracy**, but lower **memory consumption** & computation **time**



# Vector & raster



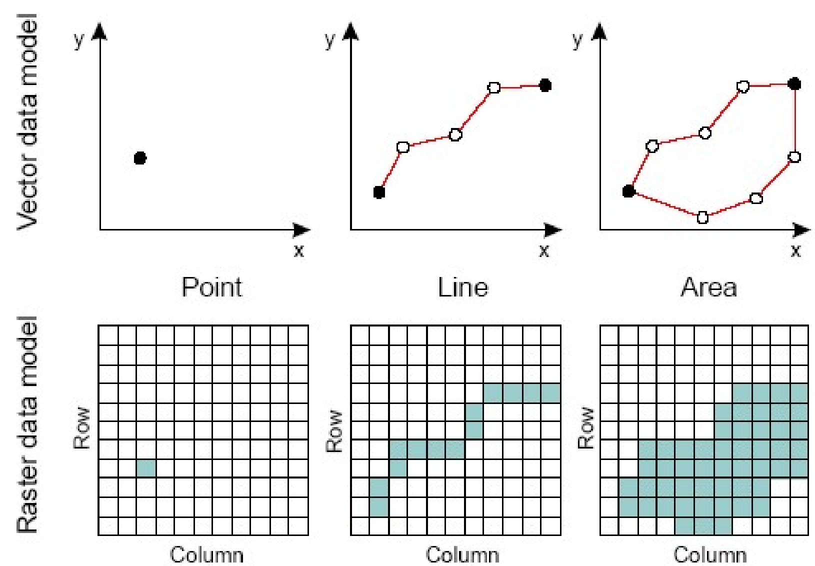
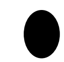
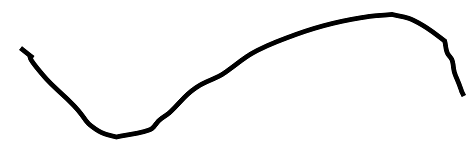




Image source

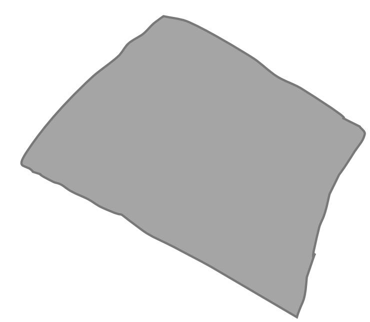
# Vector spatial data types



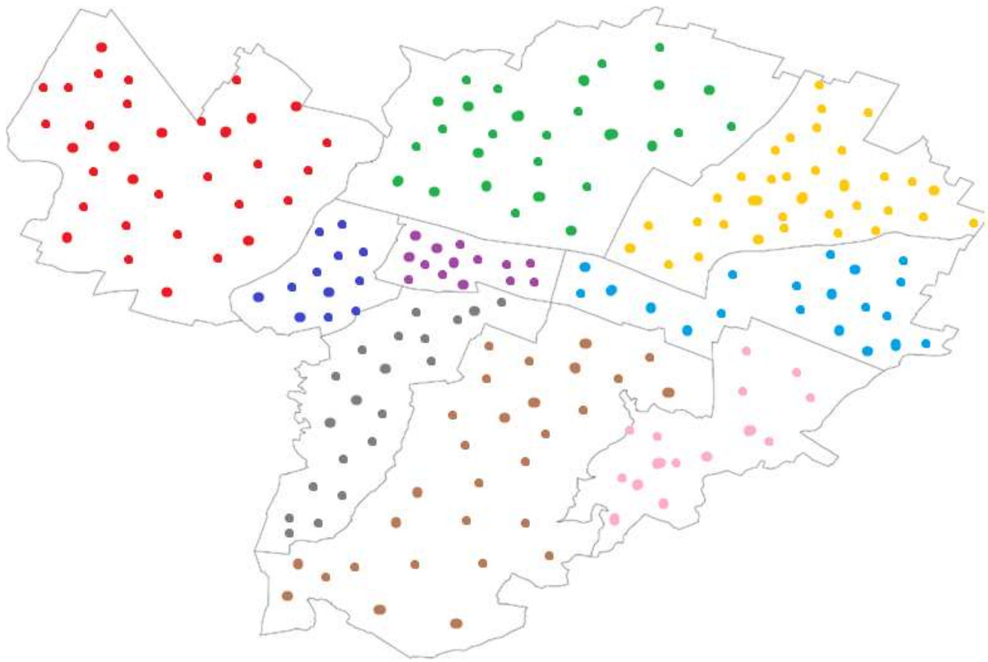
- What need to be modeled:
  - **Spatial objects**: streets, people, vehicles, cities, etc.,
  - **Embedding space**: the space from where spatial objects reside
    - Administrative **divisions** of a city (**Neighborhoods**, **districts**, **boroughs**, etc.,)



- Objects include:
  - **Points**: object location without its **extent**
    - schools, restaurants
  - **Lines**: a **trajectory** of moving spatial object or a line connecting **multiple points**
    - Streets, moving vehicle trajectory
  - **Polygons**( i.e., regions, areas): spatial objects with extents
    - Cities, countries



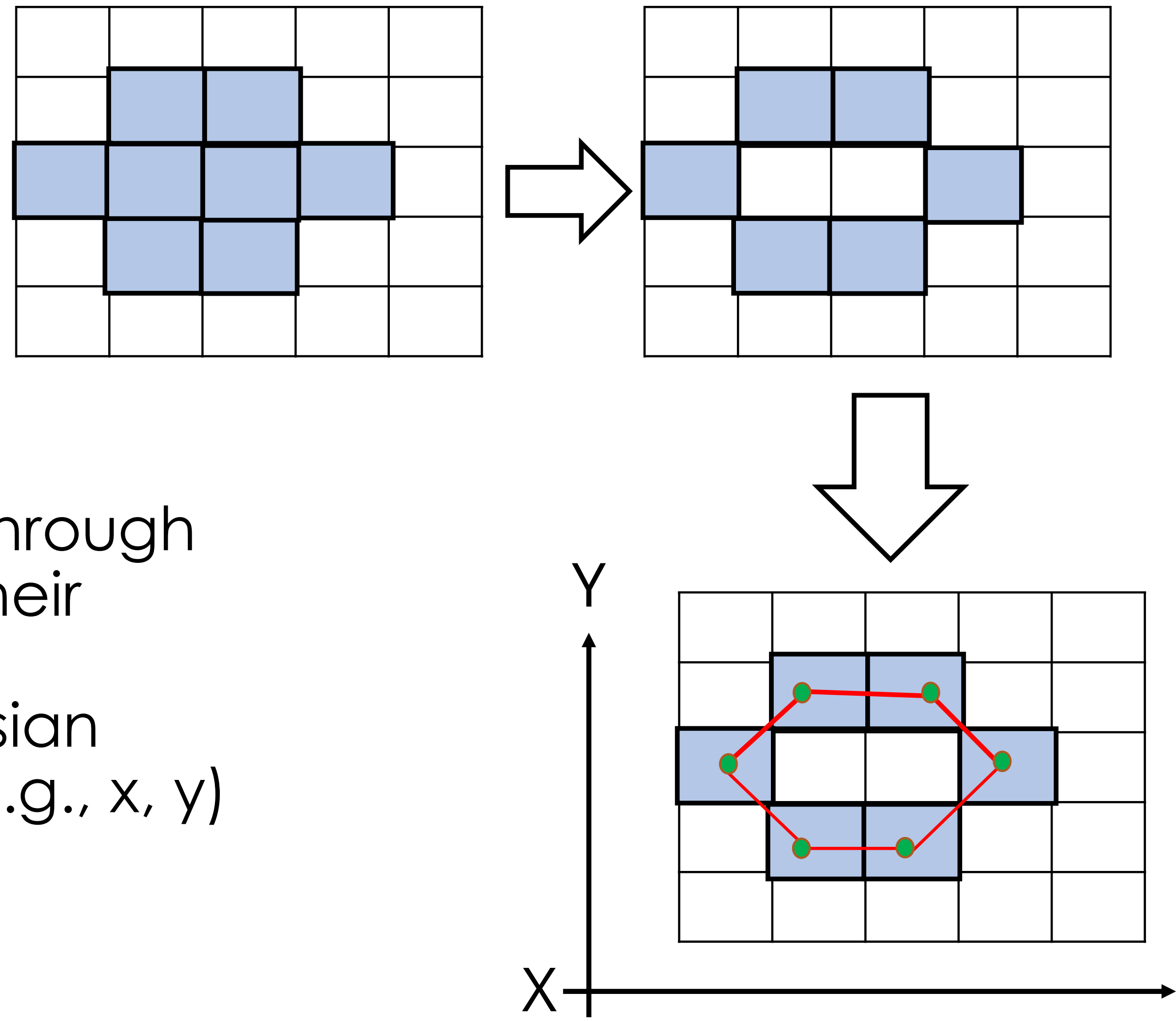
# Spatial objects



- **Embedding space** represented by **divisions** separating **administrative** areas in Bologna, Italy
- Each division contains spatial **objects** (represented as **points**): vehicles, resturants, etc.,

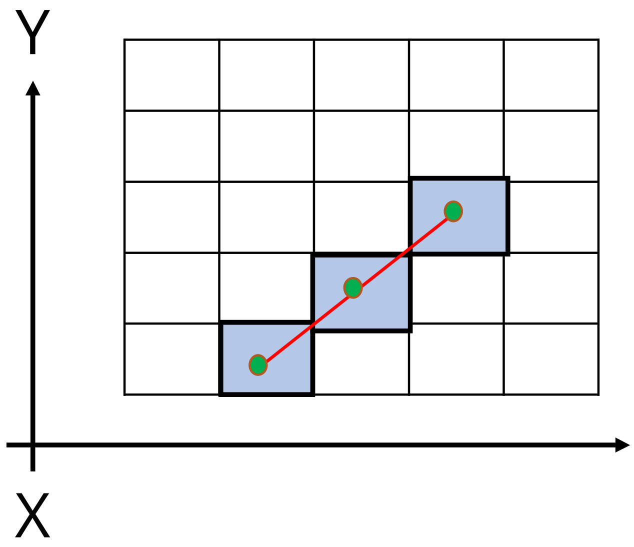
## Vectorization

- Converting raster (binary) images to vector counterparts
    - Find edge pixels
    - Draw a line passing through edge pixels & map their center points to the corresponding cartesian coordinate system (e.g., x, y)



# Rasterization

- **Point**
  - Find the **pixel** with a center that is closest to vector point
- **Line**
  - Find pixels intersecting with the original line
  - Bresenham algorithm
- **Polygon**
  - For every pixel, find if it is inside the polygon (point in polygon, PIP)
  - Polygon based fill algorithm

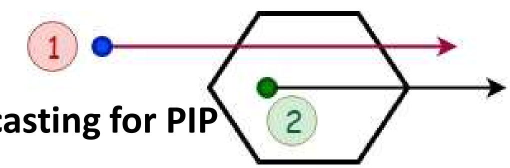


## point in polygon

- Point-in-polygon (PIP)
  - **Raw casting algorithm**

  (1) Pass a ray out from the test point

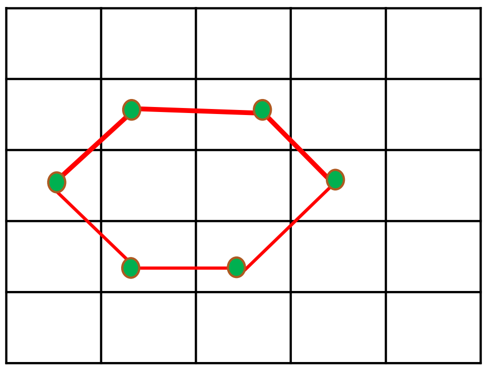
  (2) Count the number of times that the ray intersects with the boundaries of the polygon
   - Even → outside
   - Odd → inside

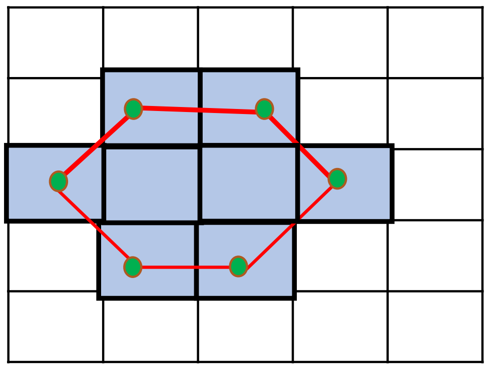



Raw casting for PIP

# Rasterization

- **Polygon based fill algorithm**
  - For each row in the grid
    - find the intersection points between the row and polygon edges
    - Sort the intersection points with reference to x-axis
    - All pixels that are located between an intersection point with an odd position and its successor are part of the polygon

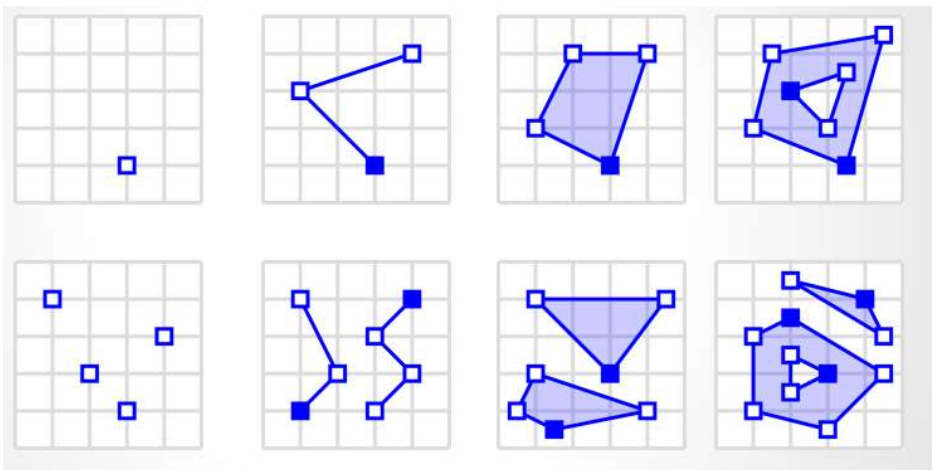




# JTS data types

- **Java Topology Suit (JTS)** is an open-source library of spatial predicates and functions for processing geometries
  - creating and manipulating vector geometry

- **Point**
  LineString
  LinearRing
  **Polygon**
  MultiPoint
  MultiLineString
  **MultiPolygon**
  GeometryCollection



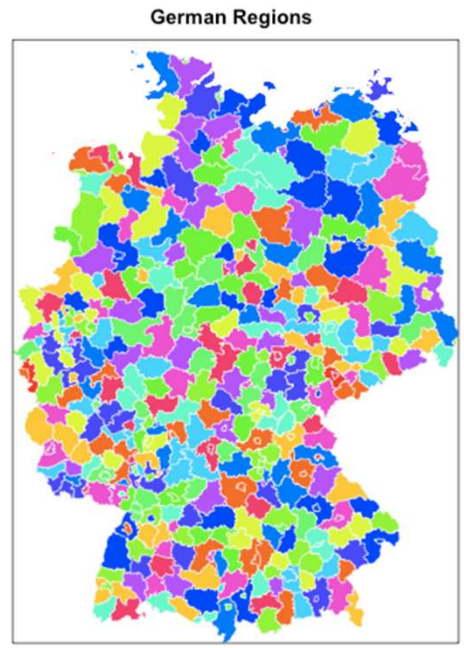
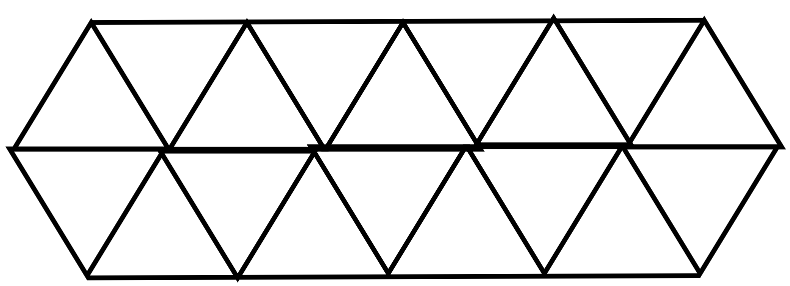
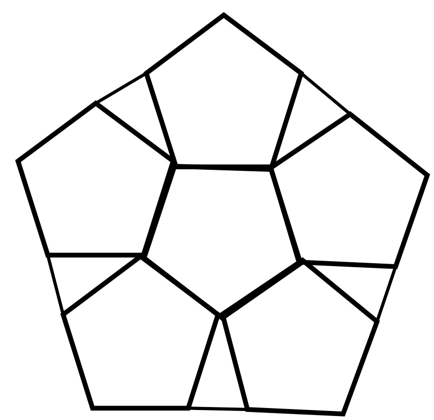



Image source

# Spatial framework

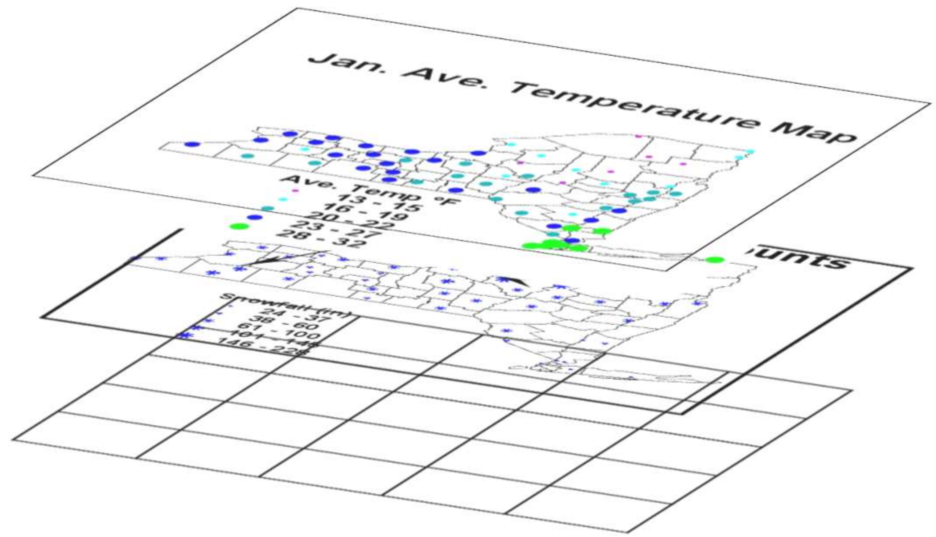
- Spatial framework: a division of a space region
  - tessellation of spatial objects





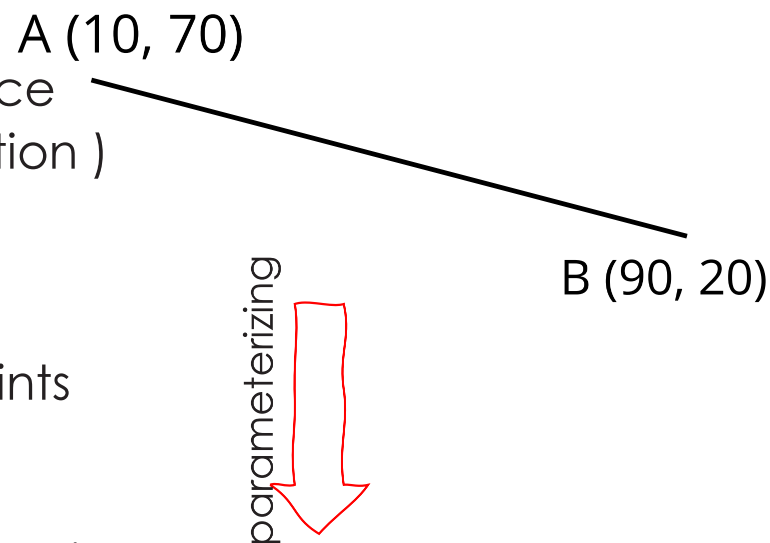
# Layer

- A spatial framework in addition to the field that assigns values for each location in the framework



# Data models

- Storing a line from 2-D space ?
  - Endpoints coordinates can be stored in 4-D space
    - Transformation (i.e., mapping, parameterization ) from 2-D embedding space to a 4-D space
    - 2-D space: the space from where lines geometrically reside
    - 4-D space: the space containing the endpoints representing the lines
  - Fine for just retrieving the data.
    - However, the inherent geometry and relationship to the embedding space are ignored




| X1 | X2 | Y1 | Y2 |
|---|---|---|---|
| 10 | 90 | 70 | 20 |

Spatial object loses its shape!

# Why not parameterizing?

- Are the two lines close to each other?
  - Difficult to tell from the 4-D space
  - Spatial proximity in 4-D space are not necessarily preserved!
- We could reconstruct into a 2-D space,
  - However, why was the transformation used!


| X1 | X2 | Y1 | Y2 |
|---|---|---|---|
| 10 | 90 | 70 | 20 |
| . | . | . | . |
| . | . | . | . |
| 40 | 60 | 44 | 32 |

Spatial object loses its shape!

# Geospatial vector file formats

- Vector files are GIS data files that represent **point**, **line**, or **polygon** data
- Common
  - Esri **Shapefile**
  - Geographic JavaScript Object Notation (**GeoJSON**)
  - OpenStreetMap **OSM XML**
  - And many others (outside the scope of discussion)
- For points vector data
  - **CSV**, **TSV**

# Esri Shapefile

- most common geospatial file type, the industry standard.
- three files that are mandatory to make up a shapefile
  - SHP is the feature geometry.
  - SHX is the shape index position.
  - DBF is the attribute data.
- optionally
  - PRJ is the projection system **metadata**
  - XML is the associated metadata.
  - SBN is the spatial index for optimizing queries.
  - SBX optimizes loading times.

Map shaper

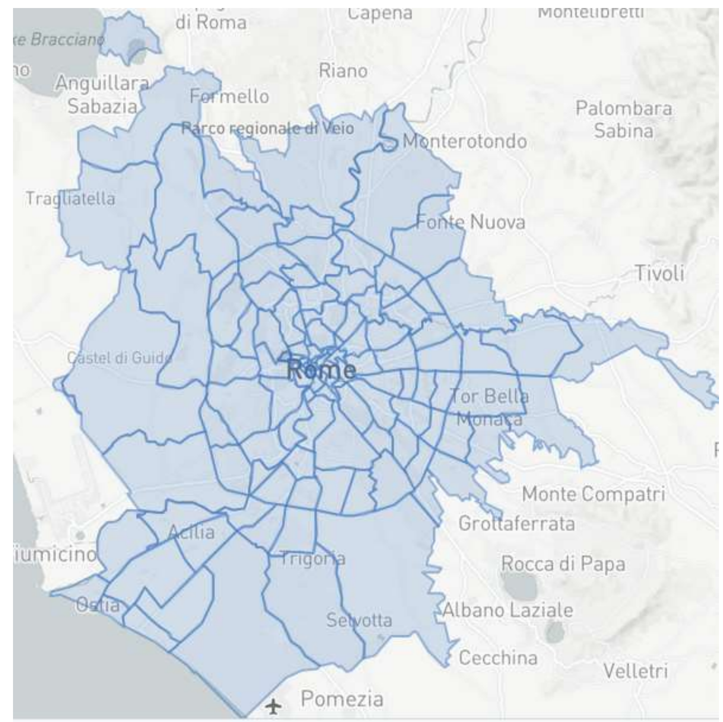
SF neighborhood

SF neighborhood shape files

# Geographic JavaScript Object Notation (GeoJSON)

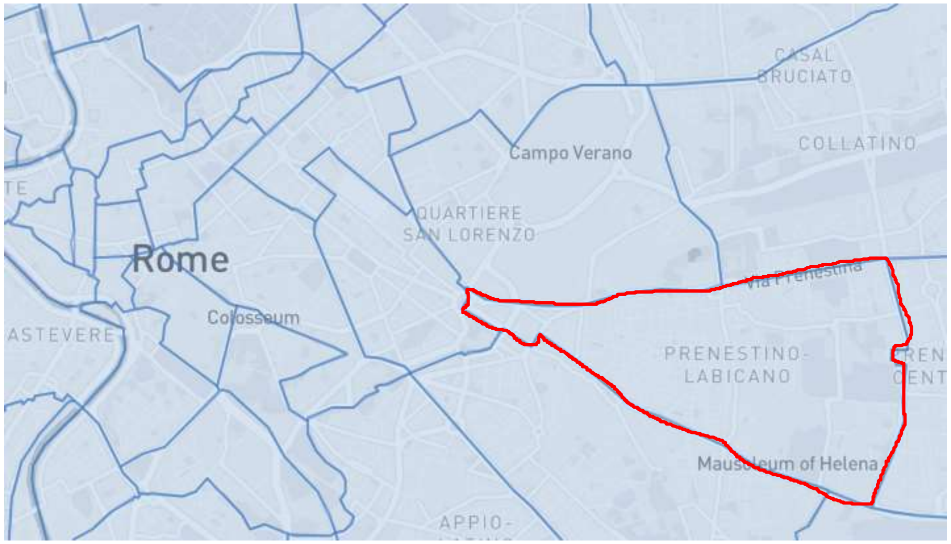
- Mostly web-based mapping
- Stores coordinates as text in JavaScript Object Notation (JSON) form
  - Vector points, lines and polygons as well as tabular information

Neighborhoods Rome, Italy



# Example GeoJSON

- { "type": "Feature", "properties": { "IDquartiere": "Q07", "TIPOLOGIA": "Quartiere", "quartiere": "Prenestino-Labicano", "CODICE_SUD": 207.0, "PERIMETRO": 10505.3598993, "AREA": 4291955.5175200002, "CODICE_NOM": "Q_07", "IDENTIFICA": 23.0 }, "geometry": { "type": "Polygon", "coordinates": [ [ [ 12.559051, 41.8948005 ], [ 12.5598259, 41.8926515 ], …….. ] ] } }

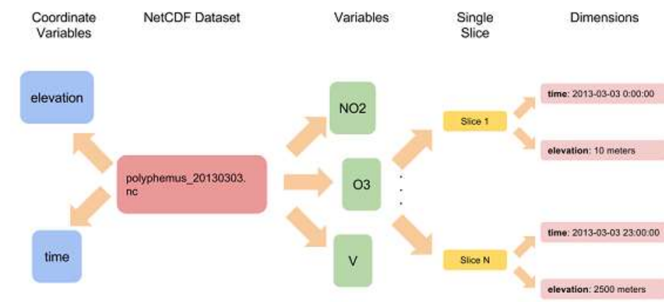


# Advanced geospatial file formats

- A lot of weather data uses **temporal GIS** data formats because of how important time is related to weather
  - **multi-temporal geospatial** data has time & geographic components
  - **weather and climate data** track temperature and meteorological changes in a geographical context across time
- Network Common Data Form (**NetCDF**)
- GRIdded Binary or General Regularly-distributed Information in Binary (**GRIB**)

# Network Common Data Form (NetCDF)

- NetCDF array-based for storing multi-dimensional data
  - A **multidimensional** array, having various variables many dimensions for every variable
- An example: **temperature**, **precipitation** or **wind speed** across **time** (**space time-series** data)
- Typical in scientific data (oceanic and atmospheric) for storing **spatial time series** data
  - Storing meteorology & **remote sensing** data
- Python tool to convert **NetCDF** to **CSV**




source

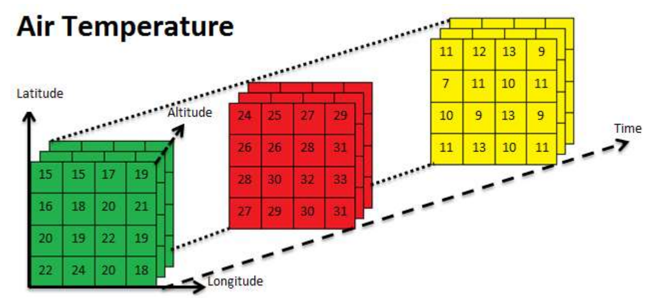



Image source

| Latitude | Longitude | Value | dataDate | time | shortName |
|---|---|---|---|---|---|
| 46.229 | 8.207 | 6.8078549464e-09 | 20141021 | 1200 | pm10 |
| 46.229 | 8.957 | 4.4633872154e-09 | 20141021 | 1200 | pm10 |
| 46.229 | 9.707 | 5.2217103974e-09 | 20141021 | 1200 | pm10 |

# Example output **NetCDF** to **CSV**

# GRIB

- Typical in **meteorology** for representing **weather** data (historical & forecast)
  - Defined by the **World Meteorological Organization (WMO)**
- **Multidimensional** files storing **meteorological** data in the form of **sequential byte array**
- Python tool to convert **GRIB** to **CSV**

Example extracting **lat, lon, 2t** (2m temperature) at time = 12:00 from a GRIB file.

```
Latitude, Longitude, Value
   90.000     0.000 2.7346786499e+02
   90.000     0.250 2.7346786499e+02
   90.000     0.500 2.7346786499e+02
   90.000     0.750 2.7346786499e+02
...
```

source

## Detour: advanced scenario

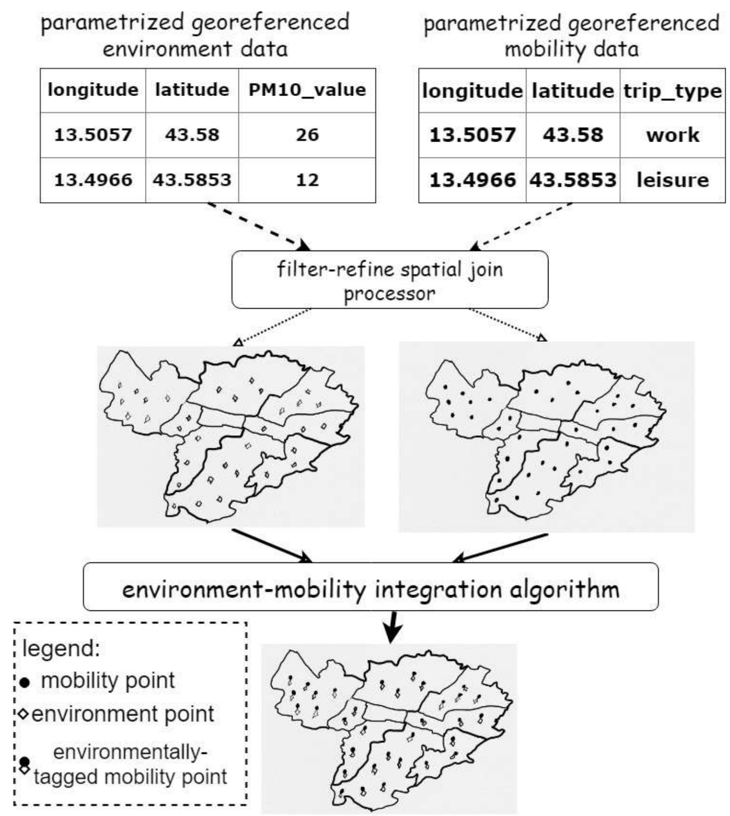
## Thinking geospatially ahead!

# Spatial multidomain analysis

- Studying the **correlation** between **vehicle pollutant emissions** and the **health of dwellers** in metropolitan cities
  - Requires **joining** geo-referenced **mobility** and **meteorological** data (**spatial join**)
- Requires regular analytics of the relationships between **mobility** patterns and **climate change**
  - e.g., Interactive **heatmap** visualization
- Helps municipalities and city officials in making strategic decisions for the benefit of the health of citizens
- We need to **join** georeferenced **meteorological** & **mobility** data

# Heuristic overview

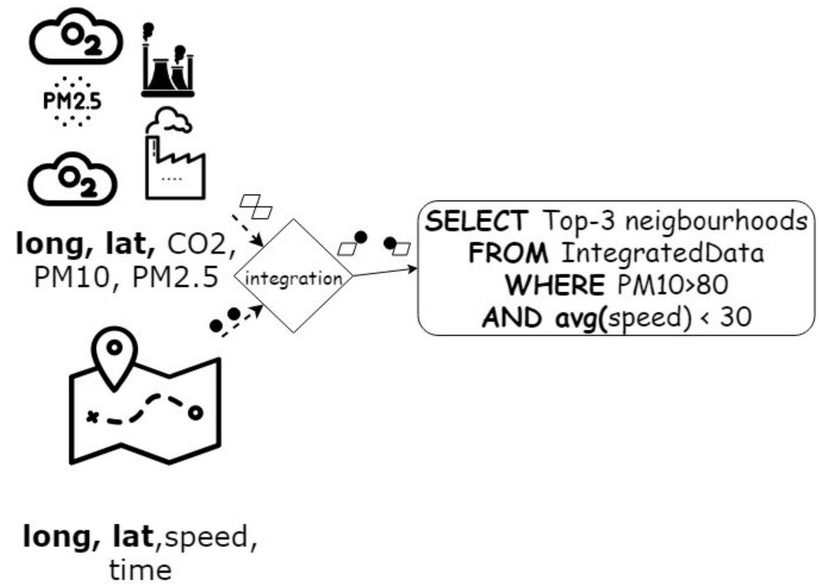
- Our method is equivalent to the heuristic overview shown in figure
- It resorts to **overlaying** corresponding maps of both datasets with a cheap **equijoin** operation
- We will discuss an efficient distributed method to perform this kind of join, at scale, with QoS guarantees in **part 3** of this course.

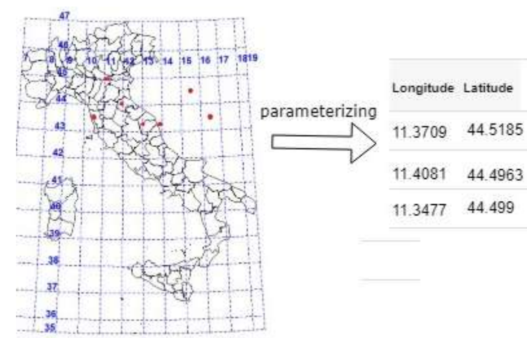


# Advanced spatial join

- A joint analysis on **location** and **time** dimensions in **series** data.
  - We need to apply **spatial join**.

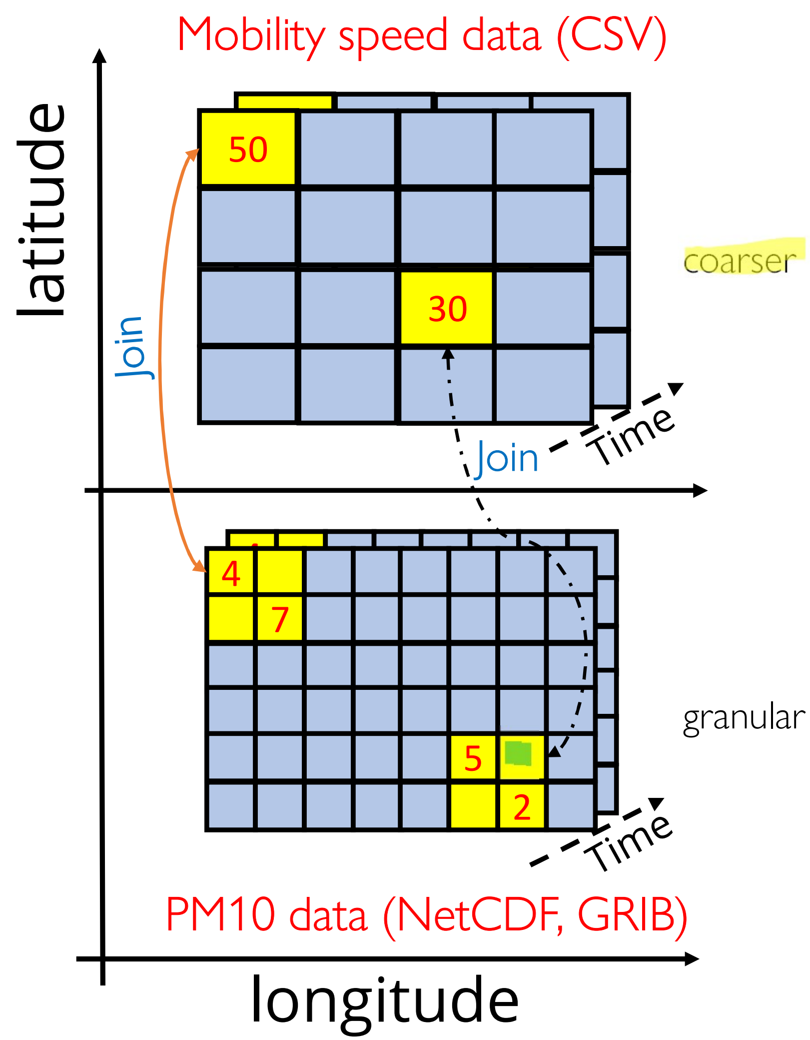
- However, **spatial join** is computationally **expensive**.
  - Spatial data is parametrized (**longitudes** and **latitudes**)
  - Objects **loses** their **geometrical** information by this **transformation**.
  - Bringing parametrized tuples back into real geometries is expensive



## Challenges

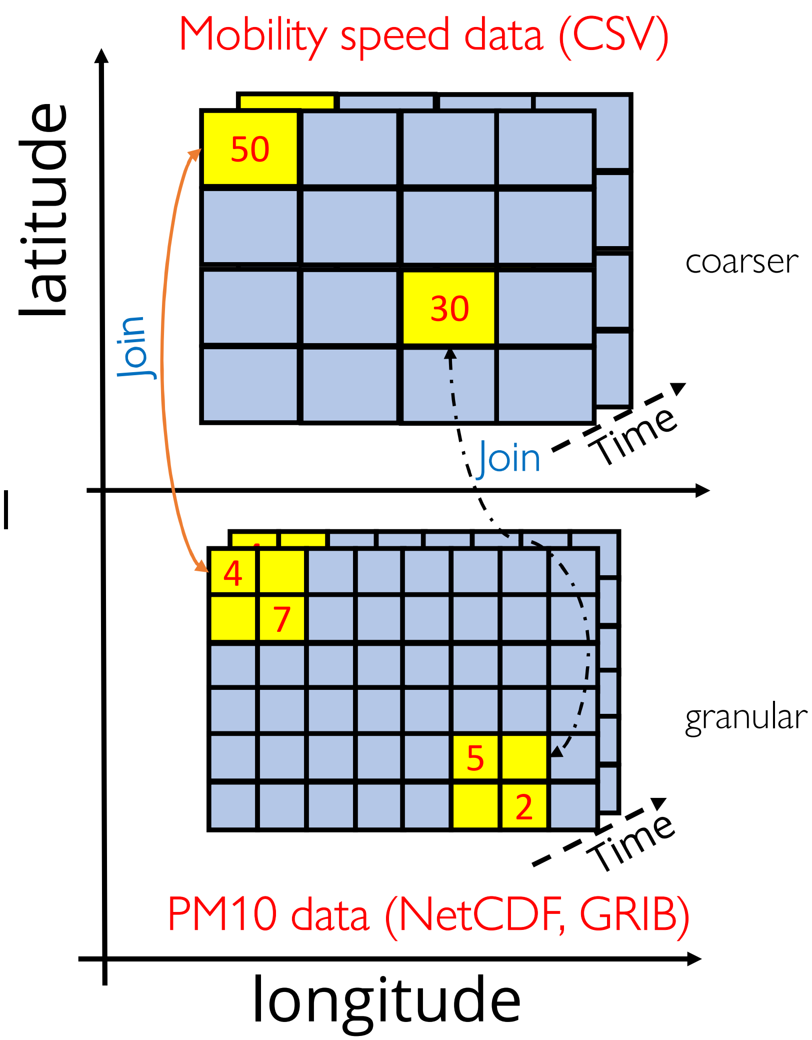
- Different **formats**. **mobility** data in tabular (e.g., **CSV**) and georeferenced **meteorological** data in **NetCDF** or **GRIB**
- Selecting the right data from a **constellation** of heterogeneous sources
  - GPS data is **not 100% accurate**
  - Loss of accuracy during data collection
  - GPS coordinates can be inaccurate when the handset is moving quickly, such as in a car or airplane
  - **Meteorological** data may have been collected with differing set of spatial **granularity** (**granular** & **coarser**)



# Challenges

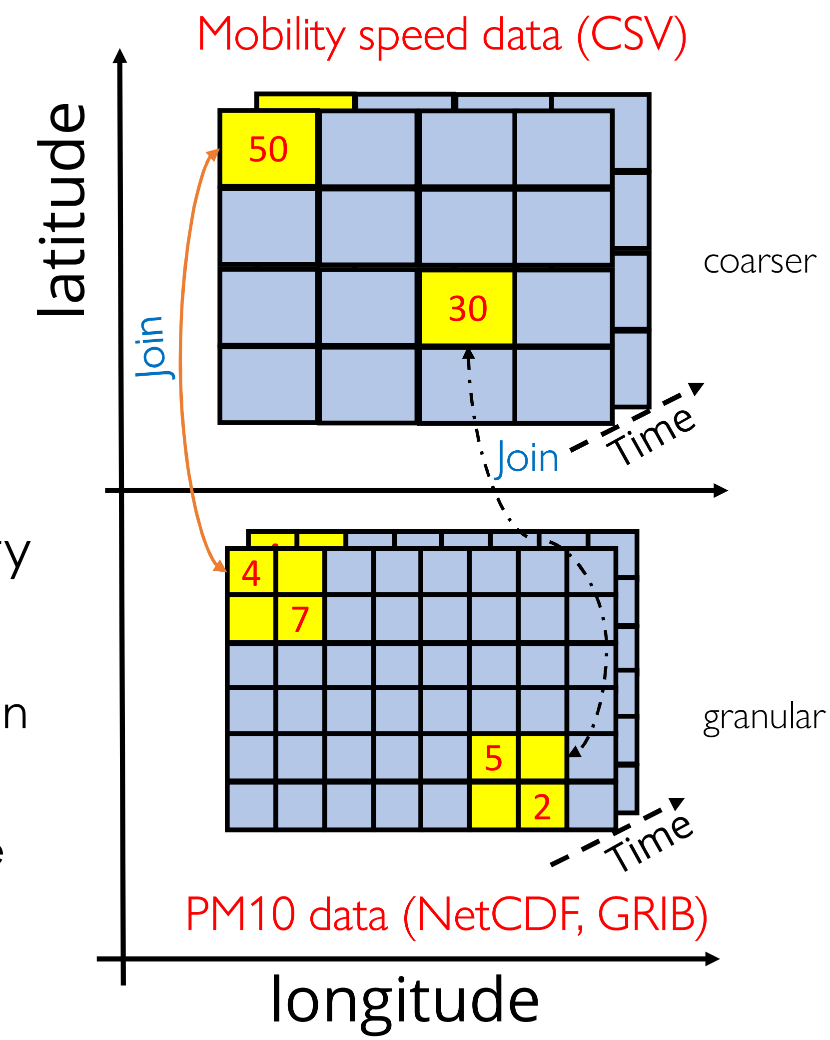
- **Interoperability** is a key
  - **Spatial interoperability**. Do data match up in the spatial dimension?
  - **Temporal interoperability**. Do **weather** and **mobility** data match up in the **temporal** space?
- What is the **spatial** & **temporal** scale for **weather** & **mobility** data
- Imagine the earth flattened and gridded, what is the **size** of each grid cell for which meteorological data is aggregated?

- What distributed **data management methods** can be used to **store** and **process** such **georeferenced multidomain** data, at scale?



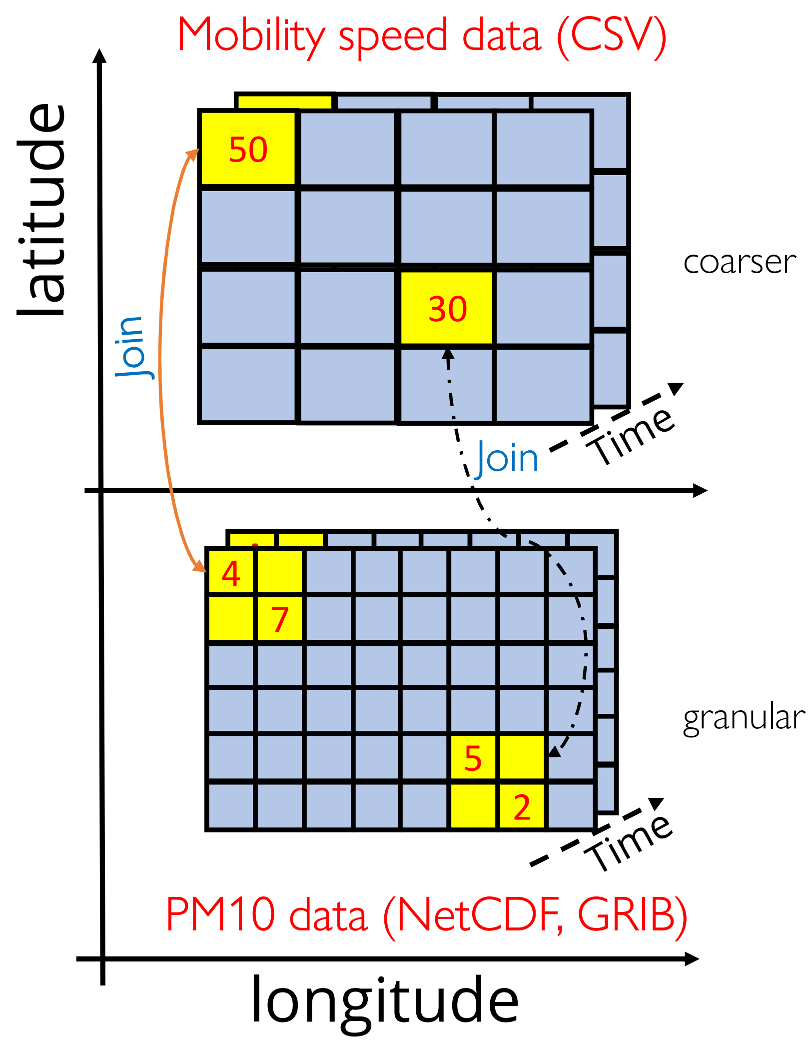
# Challenges

- **Spatial resolution**
  - "What is the smallest unit of area measured?"
  - We obtain a lower resolution by **aggregating** the data over a **greater area**, which makes it more difficult to reason about the data on a **smaller scale**
  - Whether there had been increasing median PM10 caused by vehicles density on a granular level?
    - It would be hard to establish the pattern
    - Mobility resolution are substantially lower than weather data resolution
    - Changes of median vehicle density in other parts of the coarser resolution might obscure or falsely enhance what is happening on granular level



# Challenges

- Temporal resolution
  - The frequency at which spatial data has been collected
    - How often the measurements are taken
  - We may collect mobility data on a daily basis, whereas meteorological data is taken every few hours for the same geography
  - It is not easy to discover the correlation between mobility and meteorological data on an hourly basis, different temporal resolutions

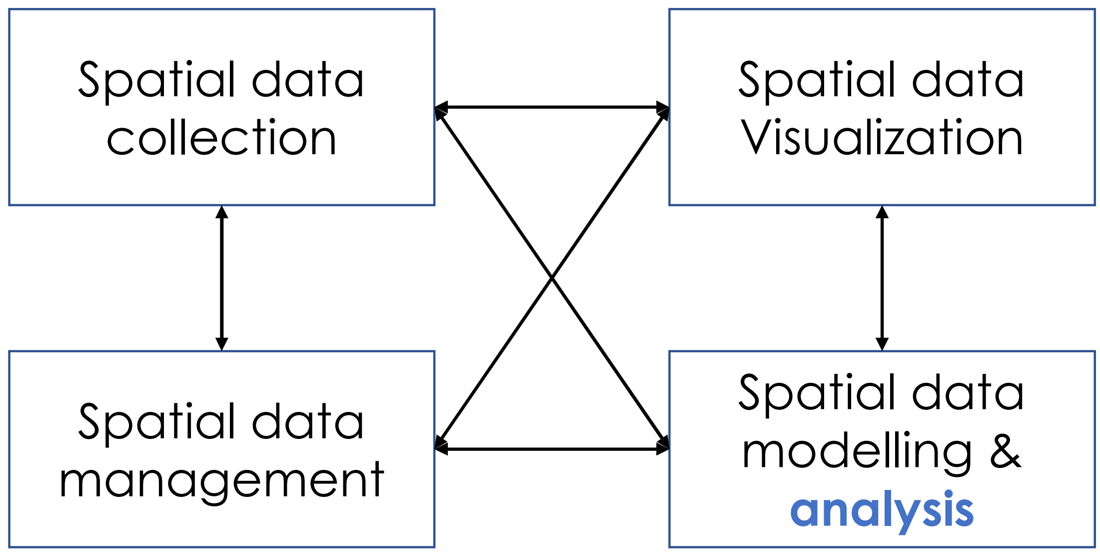


# Spatial Analysis

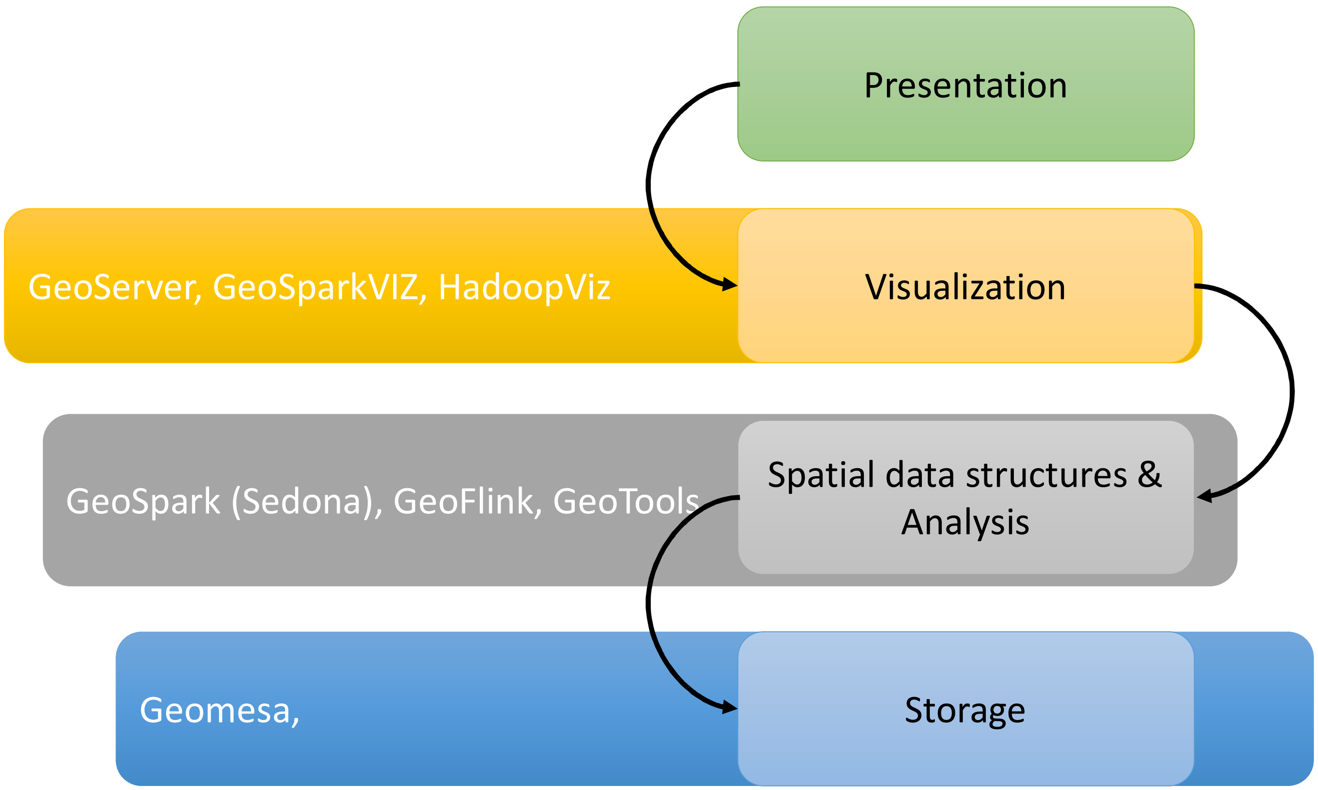
# Tasks in Geographic Information Systems (GIS)

- A geographic information system (GIS) is a fusion of computer hardware and software for
  - Collecting
  - Managing
  - Analyzing
  - Displaying

  geo-referenced data (geospatial, spatial)



# Spatial analysis stack



## Geospatial analysis

- Four kinds of analysis:
  - Point pattern
  - Autocorrelation
  - Proximity
  - Correlation
- Looking at **location** alone, or location and attributes at the same time.
- Differ in whether they scan **points** *and* **areas**, or just points *or* areas
- Looking at just one **theme** (for example, only population) or several themes at a time (like two maps of districts, one showing population density per district and the other one of average income).

# Example analysis

- *how many* rides are taken each day (in each district),
  - and we can identify the days of the week and month in which the most rides take place.

```
-[ RECORD 1 ]--------+------------------
vendor_id              | 1
pickup_datetime        | 2016-01-01 00:00:01
dropoff_datetime       | 2016-01-01 00:11:55
trip_distance          | 1.20
pickup_longitude       | -73.979423522949219
pickup_latitude        | 40.744613647460938
dropoff_longitude      | -73.992034912109375
dropoff_latitude       | 40.753944396972656
fare_amount            | 9
```

How many rides on New Year's morning originated from **within** 400m of Times Square

```
day                 | count
--------------------+--------
2016-01-01 00:00:00 | 345037
2016-01-02 00:00:00 | 312831
2016-01-03 00:00:00 | 302878
2016-01-04 00:00:00 | 316171
```

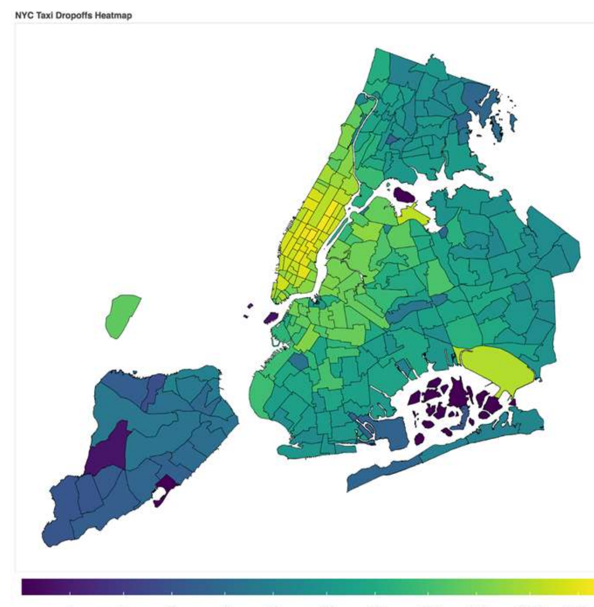



Image source

# Geodata Themes

- **Cultural**
  - Administrative (**Boundaries**, cities and planning)
  - Socioeconomic data (Demographics, economy and crime)
  - Transportation (**Roads**, **railways** and airport)
- **Physical**
  - Environmental data (Agriculture, soils and **climate**)
  - Hydrography data (Oceans, lakes and rivers)
  - Elevation data (Terrain and relief)
  - **Urban** and **regional** planning

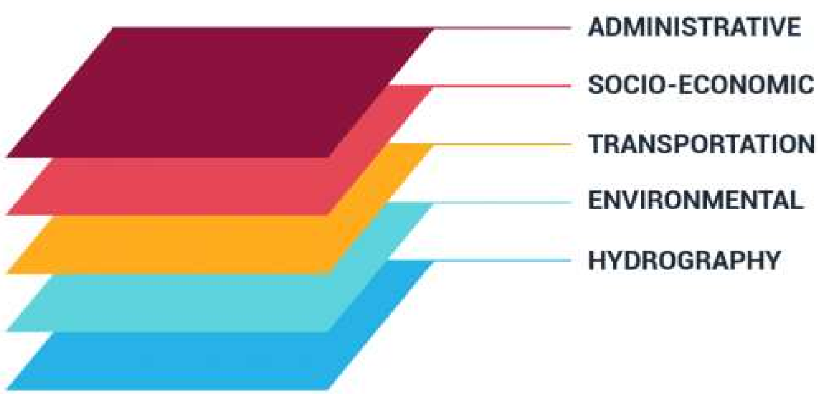


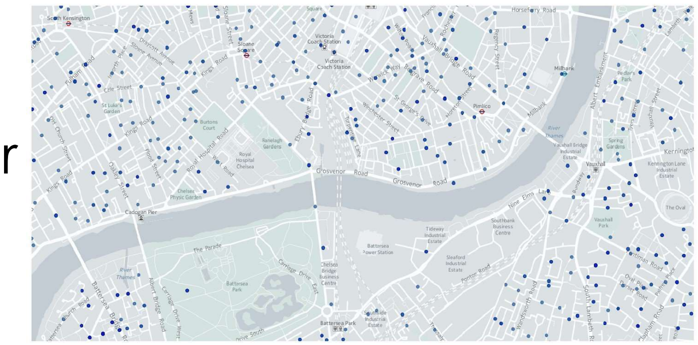

Image source

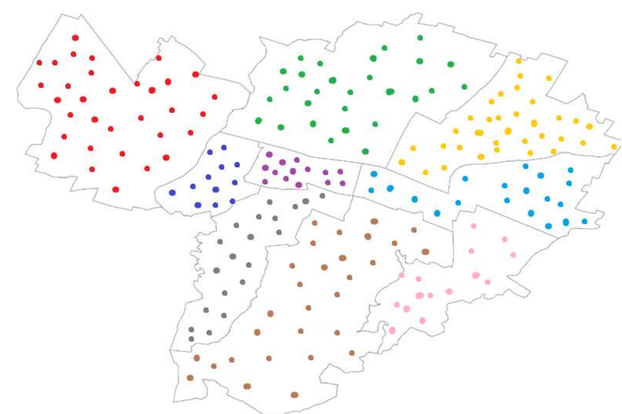
# Point Pattern Analysis

- Locational **distribution** of **objects** or **events** within one theme
  - spatial **distributions** of **locations** of **objects** or events
  - **relationships** between the locations of objects in **space** relative to the locations of other objects
- Describing the pattern of a single **theme** of interest—locations of cars —over the embedding area

Image source

London Burglary Locations (2015)





# Point Pattern Analysis

- Three types
  - **Random**
  - **Uniform**
    - Objects are roughly evenly distributed across the embedding space
    - **Sensors** for measuring the pollution levels in a city
  - **Clustered**
    - Objects are located in groups forming clusters
      - Cars in city centers during rush hours

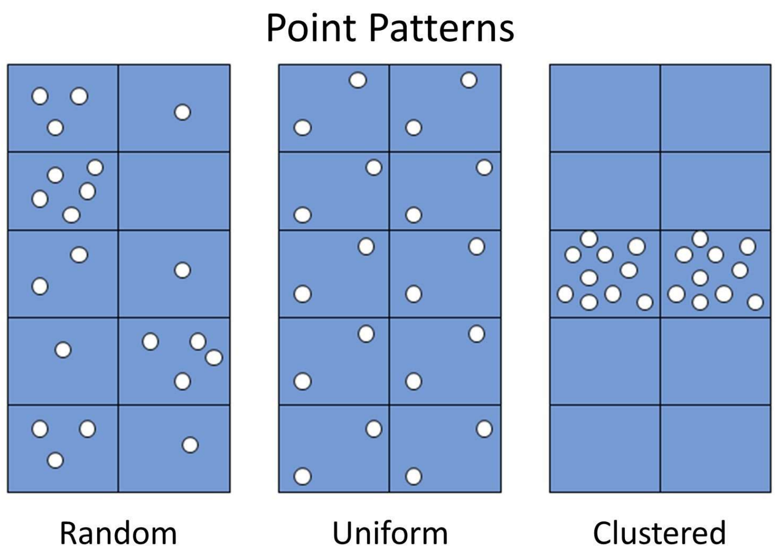



Image source

# Autocorrelation Analysis

- Spatial distribution of **location** and **attributes** over an area
  - **Mobility** data is collected at the level of individual cars, it can then be **aggregated** and **mapped** over an area, rather than tied to one specific location
  - **Autocorrelation** → the relationship of **one attribute to itself**
  - Reports these data as **mobility rates** for specific **neighborhoods**, which allows us to compare mobility among neighboring areas.
  - To make decisions regarding measures to reduce the impact of vehicle **mobility** on **traffic congestion** levels on neighborhood levels, autocorrelation analysis is the best.

# Tobler's First Law of Geography

- "Everything is related to everything else, but near things are more related than distant things."
  - ***Negative*** *autocorrelation* → nearby things are unrelated
  - ***Positive*** *autocorrelation* → *nearby things display similar pattern of the attribute*
  - ***No autocorrelation*** → no discernible pattern in the distribution of the attribute

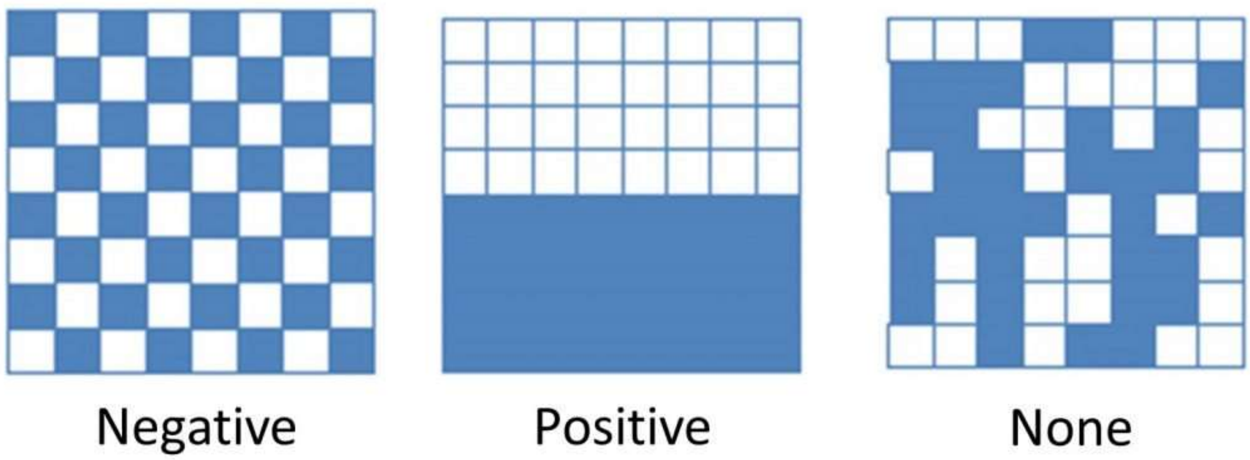



Image source

- London burglary rates aggregated by **borough**
- **positive autocorrelation** → Those with high burglary rates are generally located nearer to other boroughs with high or above-average burglary rates.

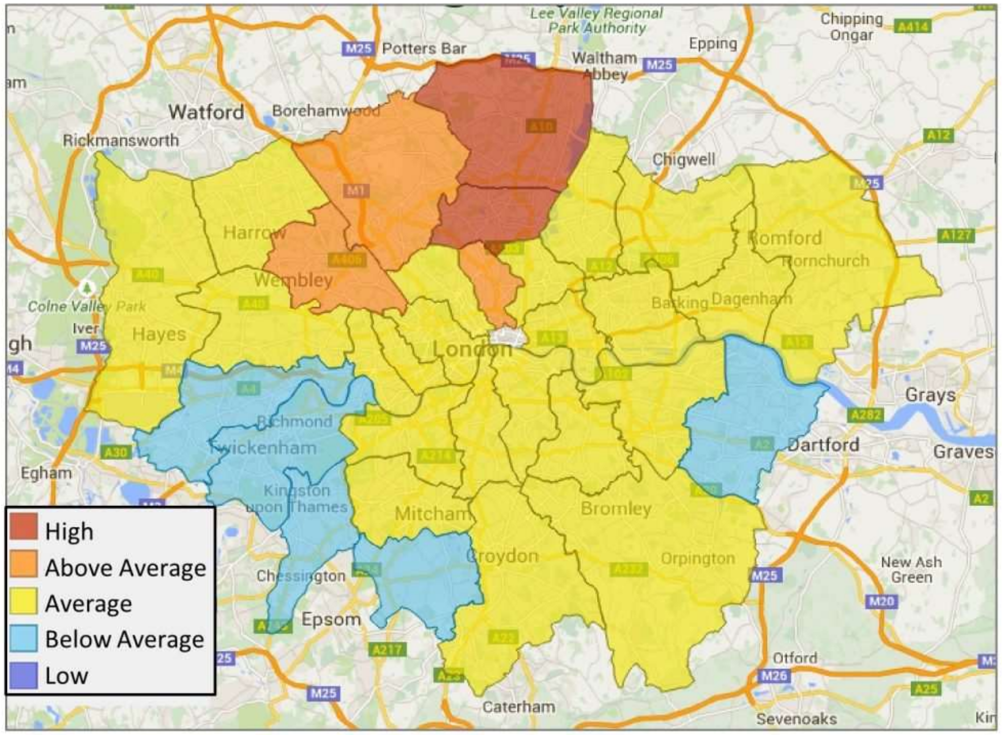



Image source

# Proximity Analysis

- Describes the spatial relationships and patterns between locations across **two themes**
  - **point pattern analysis** with two different kinds of objects or events.
  - Relationship between vehicle **mobility** and air **pollutions**
  - help us to make sense of the world across time and locational distance
  - tremendously useful for public health → determining how diseases spread (outbreaks), predicting vulnerability to disease, and how and where interventions are essential

## Correlation Analysis

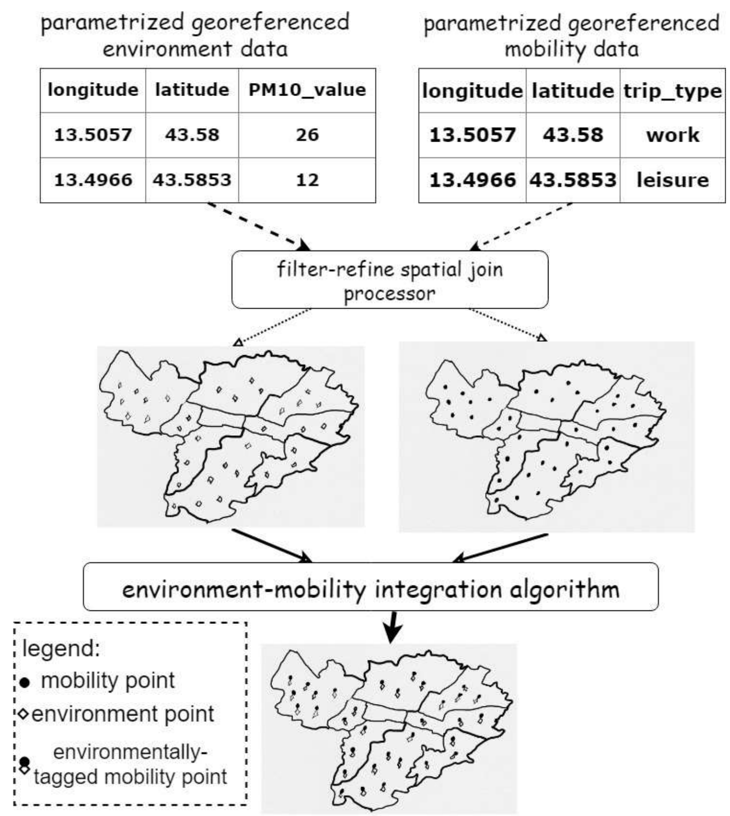
- Analyzing the spatial relationship between **multiple attributes** or **themes**
  - Relationship between mobility and pollution rates
  - the degree or extent to which **two** or more different **attributes** are **spatially** related
  - relationship between an **aggregated** attribute and a specific **point**
  - overlaps between proximity and correlation analysis

# Spatial multidomain analysis

- Studying the **correlation** between **vehicle pollutant emissions** and the **health of dwellers** in metropolitan cities
  - Requires **joining** geo-referenced **mobility** and **meteorological** data (**spatial join**)
- Requires regular analytics of the relationships between **mobility** patterns and **climate change**
  - e.g., Interactive **heatmap** visualization
- Helps municipalities and city officials in making strategic decisions for the benefit of the health of citizens
- We need to **join** georeferenced **meteorological** & **mobility** data

# Heuristic overview

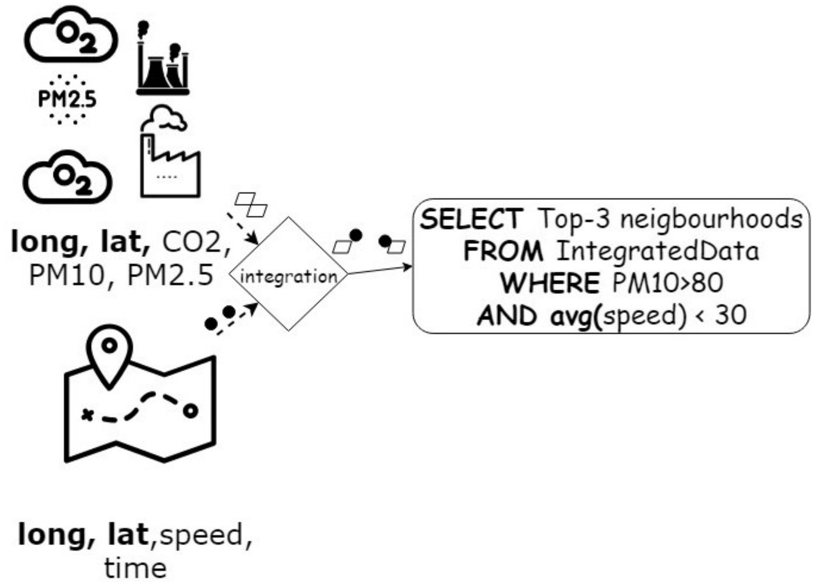
- Looking at the **two maps** side by side, we can figure a general **correlation** between **neighborhoods** with relatively high **mobility** rates and higher levels PM 10 **pollutions**
- It resorts to overlaying corresponding maps of both datasets with a cheap **equijoin** operation
- We will discuss an efficient distributed method to perform this kind of **join**, at scale, with QoS guarantees in **part 3** of this course.

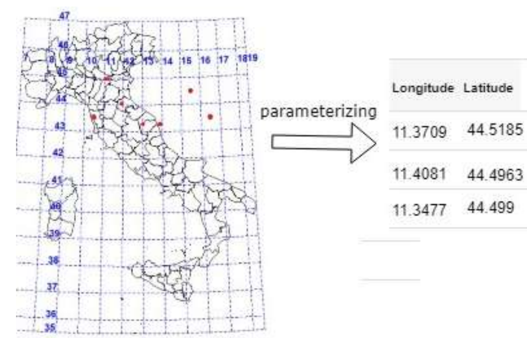


# Advanced spatial join

- A joint analysis on **location** and **time** dimensions in **series** data.
  - We need to apply **spatial join**.

- However, **spatial join** is computationally **expensive**.
  - Spatial data is parametrized (**longitudes** and **latitudes**)
  - Objects **loses** their **geometrical** information by this **transformation**.
  - Bringing parametrized tuples back into real geometries is expensive



# Geospatial analysis

| Analysis | concentration | Geometries | Themes |
|---|---|---|---|
| Point pattern | location | point | 1 |
| Proximity | Location | Point/region | 2+ |
| Correlation | Location and attribute values | Point/region | 2+ |
| Autocorrelation | Location and attribute values | Region | 1 |

# Querying Geospatial Data

- What is the spatial relationship between neighborhoods
  - For example, joining a table to itself and measuring distances between neighborhoods

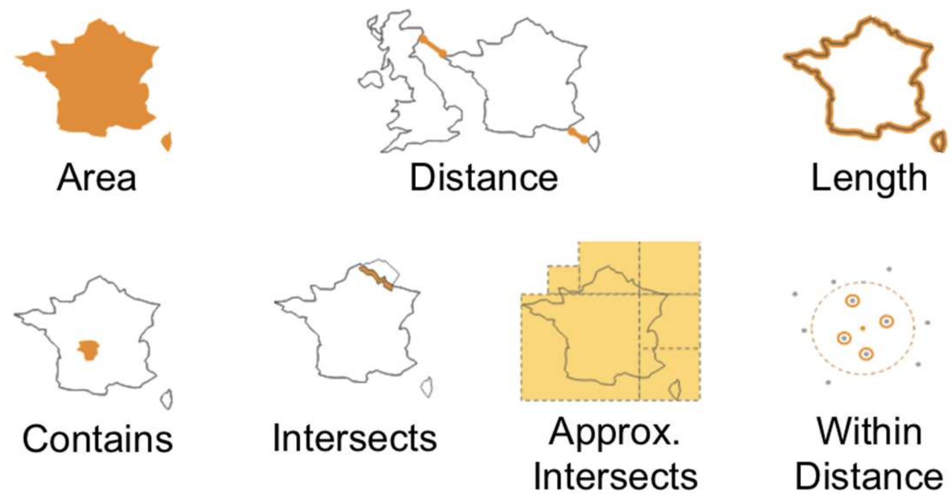
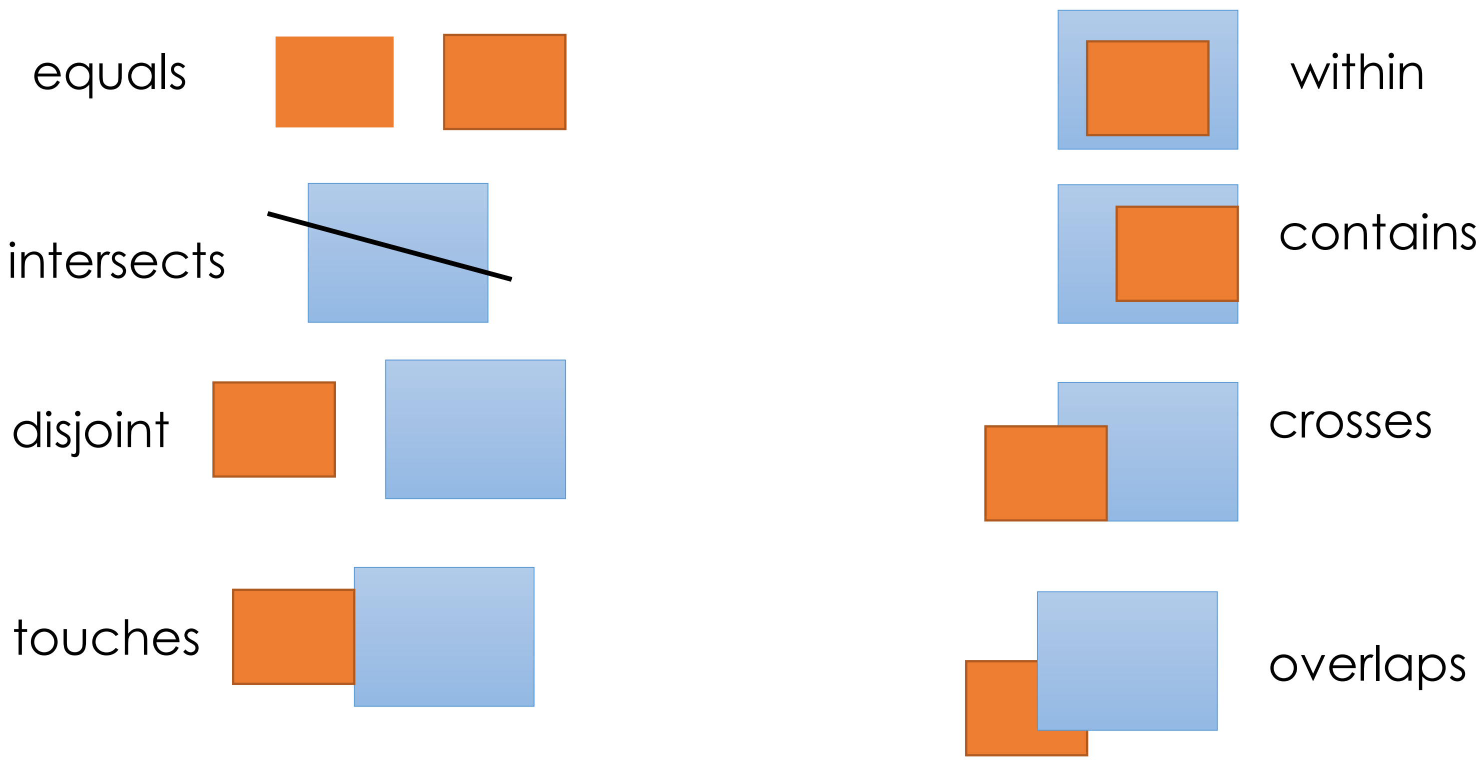



Image source

# Spatial relationships (predicates)

**the most important operations**



# Spatial relationships examples

- ST_WITHIN(**x**, **y**)
  - no point of **x** is outside of **y**
- ST_INTERSECTS(**x**, **y**)
- ST_EQUALS (**x** , **y**)
  - **X** and **y** represent the same
- ST_TOUCHES(**x** , **y**)
  - **X** intersects **y**. The interior of **x** and the interior of **y** are disjoint