COMP5339: Data Engineering

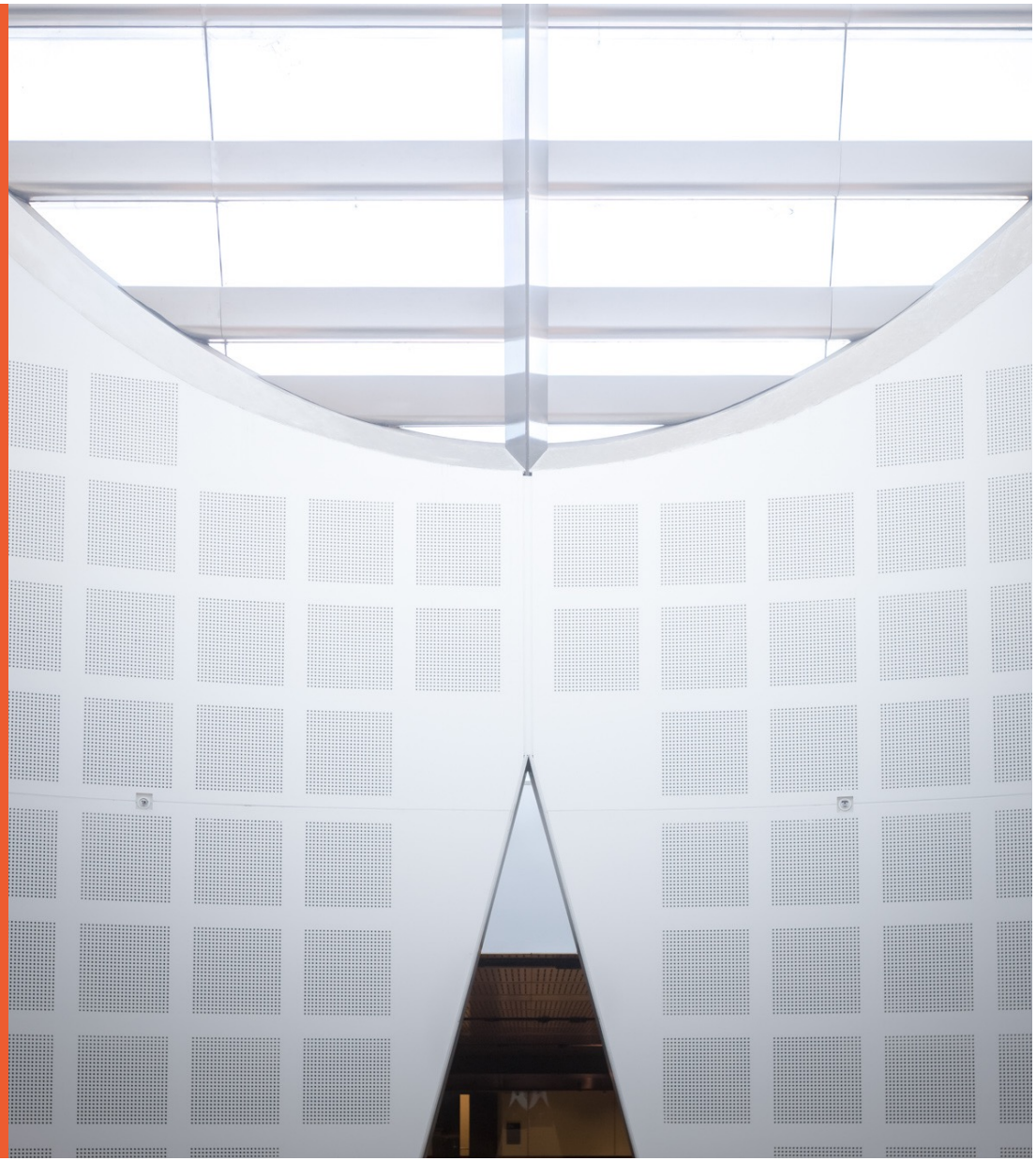
W6: Spatial Data Engineering

Presented by

Alan Fekete

School of Computer Science

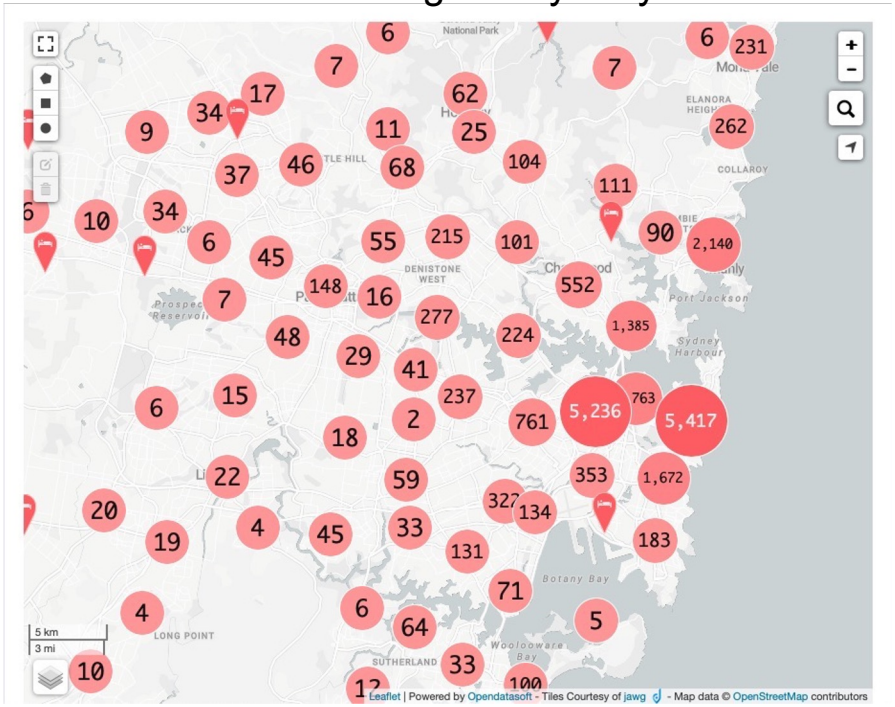
THE UNIVERSITY OF SYDNEY



Motivation

- Data often have spatial attributes
  - For example, address of listings or GPS location of photos
  - Important for spatial data analysis
  - Great for data visualisation
- Spatial query examples
  - Show the property listings within ten kilometres of your current location
  - List all organisations from Victoria and its adjoining states
- Data engineering for spatial data?
  - How to efficiently load, transform and serve spatial data?

Airbnb Listings in Sydney



[Source: public.opendatasoft.com]

Example Applications of Spatial Data

- Location-aware services
  - Location-aware mobile phone apps (eg. food finder, ride-/car-/bike-sharing apps, etc.)
  - Web APIs for geo-data (eg. location APIs)
- Geographic Information Systems (GIS)
  - E.g., ESRI's ArcInfo; OpenGIS Consortium
  - Geospatial information
  - All classes of spatial queries and data are common
- Multimedia Databases and data analysis
  - Images, video, text, etc. stored and retrieved by content
  - First converted to feature vector form; high dimensionality
  - Nearest-neighbor queries are the most common
- …

Overview of Spatial Data

Spatial Data

- Spatial data is about objects and entities which have a location and/or a geometry
- A special form is **geospatial data** which refers to data or information that identifies the geographic location of features and boundaries on Earth (such as localities, cities, suburbs etc)

Example: OpenStreetMap API

[Doc: https://wiki.openstreetmap.org/wiki/Nominatim]

https://nominatim.openstreetmap.org/search?city=Sydney&country=Australia&format=json


```
[ { "place_id": 198993911,
    "licence": "Data © OpenStreetMap contributors, ODbL 1.0. https://osm.org/copyright",
    "osm_type": "relation", "osm_id": 5750005,
    "boundingbox": [
      "-34.1732416",
      "-33.3641481",
      "150.260825",
      "151.3427428"
    ],
    "lat": "-33.8548157",
    "lon": "151.2164539",
    "display_name": "Sydney, New South Wales, Australia",
    "class": "place",
    "type": "city",
    "importance": 0.9623536703831199,
    "icon": "https://nominatim.openstreetmap.org/images/mapicons/poi_place_city.p.20.png"
  },
  { "place_id": 198840448,
     "osm_type": "relation", "osm_id": 5729534,
    "boundingbox": [
      "-33.8797564",
      "-33.8561096",
      "151.196999",
      "151.223011"
    ],
    "lat": "-33.8679574", "lon": "151.210047",
    "display_name": "Sydney, Council of the City of Sydney, New South Wales, 2000, Australia",
    "type": "administrative",
    …
  }
]
```

SDBMS vs GIS

空间数据

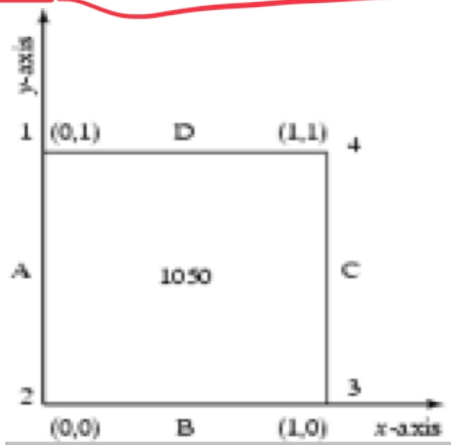
- Spatial Database Management System (SDBMS)
  - Handle a large amount of spatial data stored in secondary storage.
  - Spatial semantics built into query language
  - Specialized index structure to access spatial data
- Geographic Information System (GIS)
  - SDBMS Client
  - Characterized by a rich set of geographic analysis functions
  - SDBMS allows GIS to scale to large datasets, which are now becoming the norm
  - Information in GIS is typically organized in "layers".
    - For example, a map has a layer of "roads", "train stations", "suburbs", …
    - GIS allows data exploration and integration across layers.

Example: Census Block Database

一种空间数据类型

```
CREATE TABLE census_blocks (
    name          text,
    area          float,
    population    number,
    boundary      polyline );
```

- Note: Polyline is not a built-in datatype in non-spatial databases



example

Census_blocks

Name	Area	Population	Boundary
1050	1	1839	Polyline ((0,0),(0,1),(1,1),(1,0))

Example (cont'd): Mapping Spatial Data into Relations (if no spatial data type available)

Census_blocks

Name	Area	Population	boundary-ID
340	1	1839	1050

Polygon

boundary-ID	edge-name
1050	A
1050	B
1050	C
1050	D

Edge

edge-name	endpoint
A	1
A	2
B	2
B	3
C	3
C	4
D	4
D	1

Point

endpoint	*x*-coor	*y*-coor
1	0	1
2	0	0
3	1	0
4	1	1

Possible, but not ideal: Even simple queries now involve many joins.
Hence: SDBMS features are very important when dealing with spatial data.

Types of Spatial Data

- Point Data
  - Points in a multidimensional space
  - E.g., geo-location of some data entities from location-aware apps
  - E.g., ***raster data*** such as satellite imagery, where each pixel stores a measured value
  - E.g., Feature vectors extracted from text

point by point within image

- Region Data
  - Objects have spatial extent with location and boundary
  - DB typically uses geometric approximations constructed using line segments, polygons, etc., called ***vector data***.

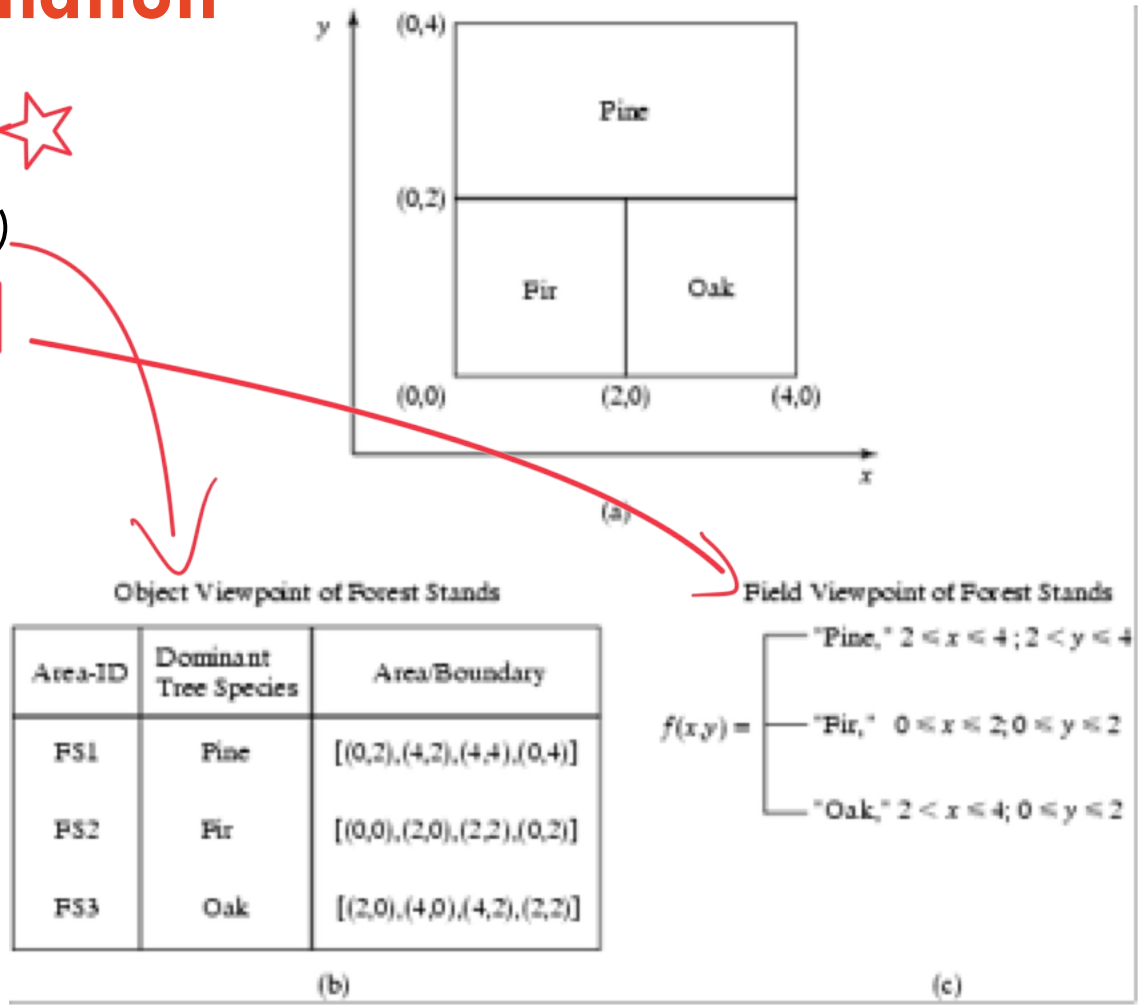
Quiz

- Are all images classified as raster data?

No

Models of Spatial Information

- Two common models
  - Object based *(also: entity-based)*
  - Field based *(also: space-based)*
- Example: Forest stands
  - (a) forest stand map
  - (b) Object view has 3 polygons
  - (c) Field view has a function



Area-ID	Dominant Tree Species	Area/Boundary
FS1	Pine	[(0,2),(4,2),(4,4),(0,4)]
FS2	Fir	[(0,0),(2,0),(2,2),(0,2)]
FS3	Oak	[(2,0),(4,0),(4,2),(2,2)]

Summary

- Spatial data
  - is about objects and entities which have a location and/or a geometry
  - point versus region data
  - can have different representations: Object based and Field based
- SDBMS vs GIS
- We will concentrate on object-based models and SDBMS

Quiz

- Is GIS a data storage system? No

↓ it is a client

SDBMS store the spatial data

OGC Spatial Data Model

- Open Geospatial Consortium

Object Model

- Object model concepts
  - **Objects**: distinct identifiable things relevant to an application
    - Objects have attributes and operations
  - **Attribute**: a simple (e.g. numeric, string) property of an object
  - **Operations**: function maps object attributes to other objects
- Example from a road map
  - Objects: roads, landmarks, ...
  - Attributes of road objects:
    - spatial: location, e.g. polygon boundary of land-parcel
    - non-spatial: name (e.g. Route 66), type (e.g. motorway, residential street), number of lanes, …
  - Operations on road objects: determine length, determine intersection with other roads, ...

Understand how to distinguish datatype

Classifying Spatial Objects

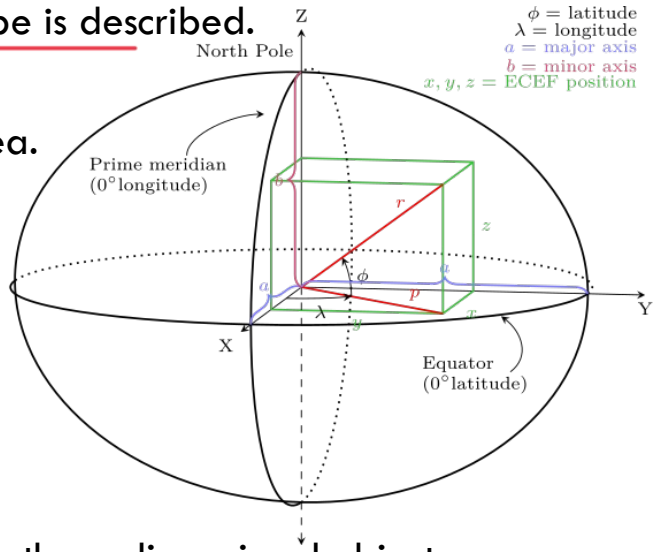
- Spatial objects are spatial attributes of general objects
- Spatial objects are of many types
  - Simple
    - 0- dimensional (points), 1 dimensional (curves), 2 dimensional (surfaces)

Spatial Object Types	Example Object	Dimension
Point	Road intersection	0
Curve	River	1
Surface	Country	2

  - Collections
    - Polygon collection (e.g. boundary of Japan or Hawaii), …

Coordinate Systems

- Each instance of a spatial type has a *spatial reference identifier* (SRID)
  - SRID is an integer that identifies the coordinate system in which the type is described.
  - Two instances with different SRIDs are incompatible.
  - Different SRIDs mean different results for some operations, such as area.
- Typical types of coordinate systems
  - **Cartesian Coordinates**
    - point positions from a defined origin along axes on a plane
  - **Geographic Coordinates (Geodetic)**
    - angular coordinates (longitude and latitude)
  - **Geocentric Coordinates**
    - three-dimensional Cartesian coordinates that model the earth as a three-dimensional object.
  - **Projected Coordinates**
    - planar Cartesian coordinates that result from performing a mathematical mapping from Earth surface to a plane



Quiz

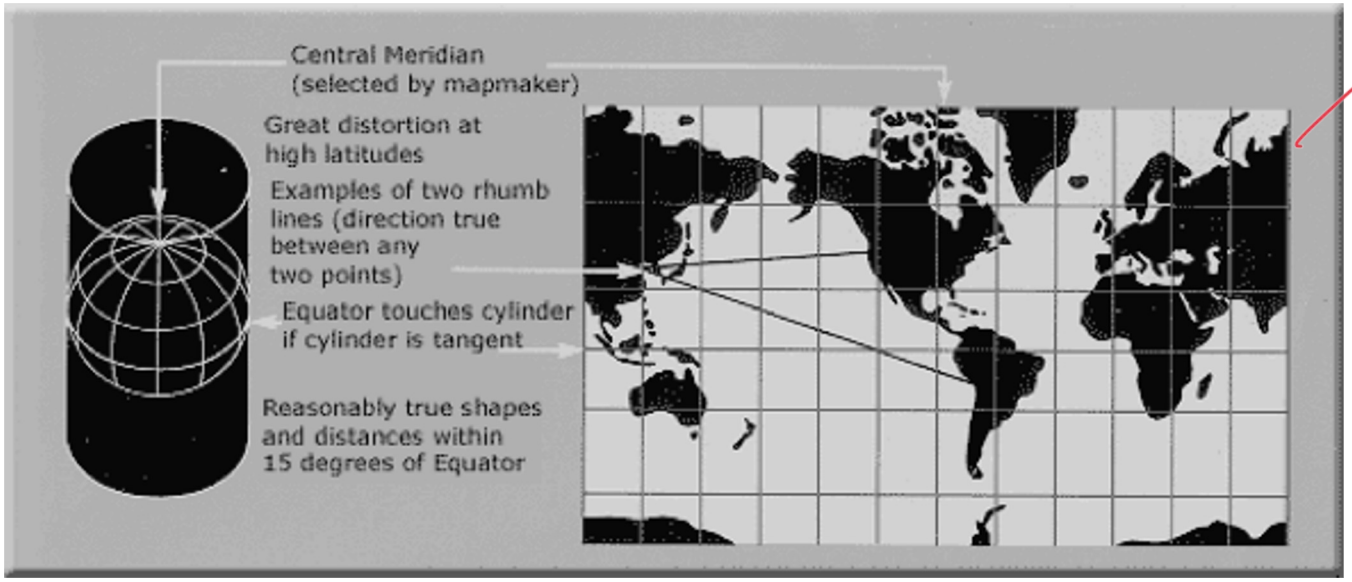
- Is transformation between Geographic coordinates (3-dim) and Geocentric coordinates (lon. and lat.) lossless?

No

North pole cannot be resolved

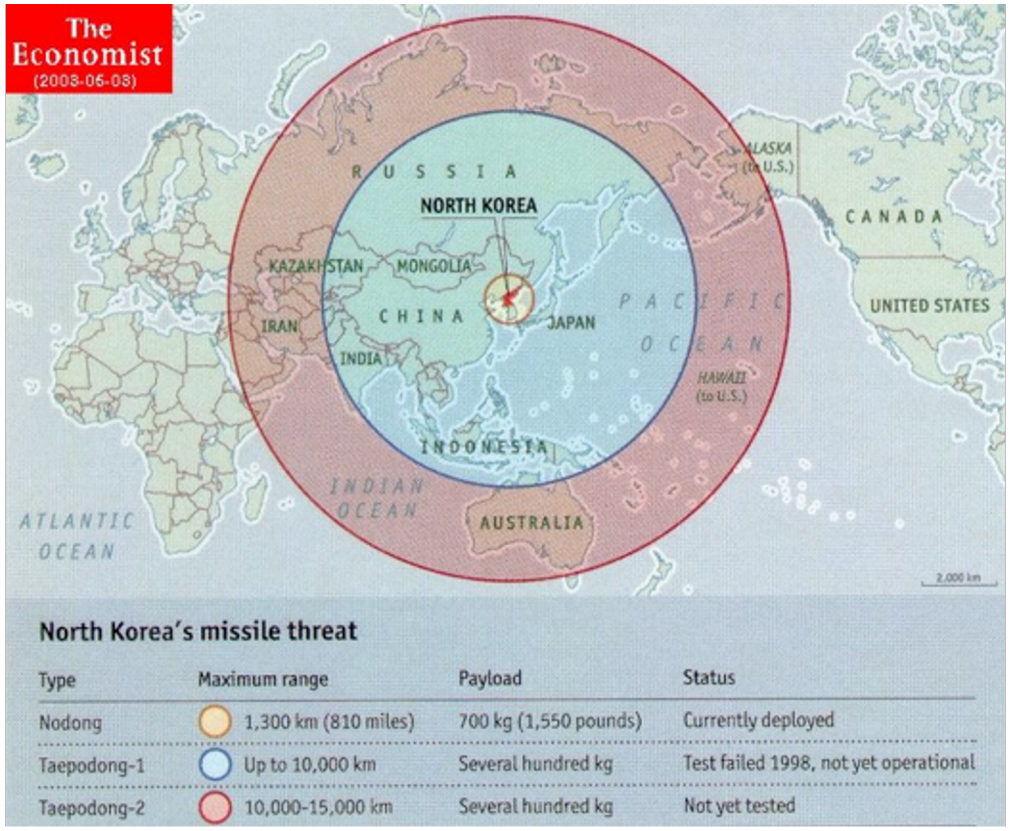
Mercator Projection

- Projections try to fit something round, like the earth, into something flat, like a map.
- This process introduces distortion
  - For Mercator projection, most of the distortion happens at high latitude relative to the equator
  - E.g., in reality, Greenland is 14 times smaller than Africa



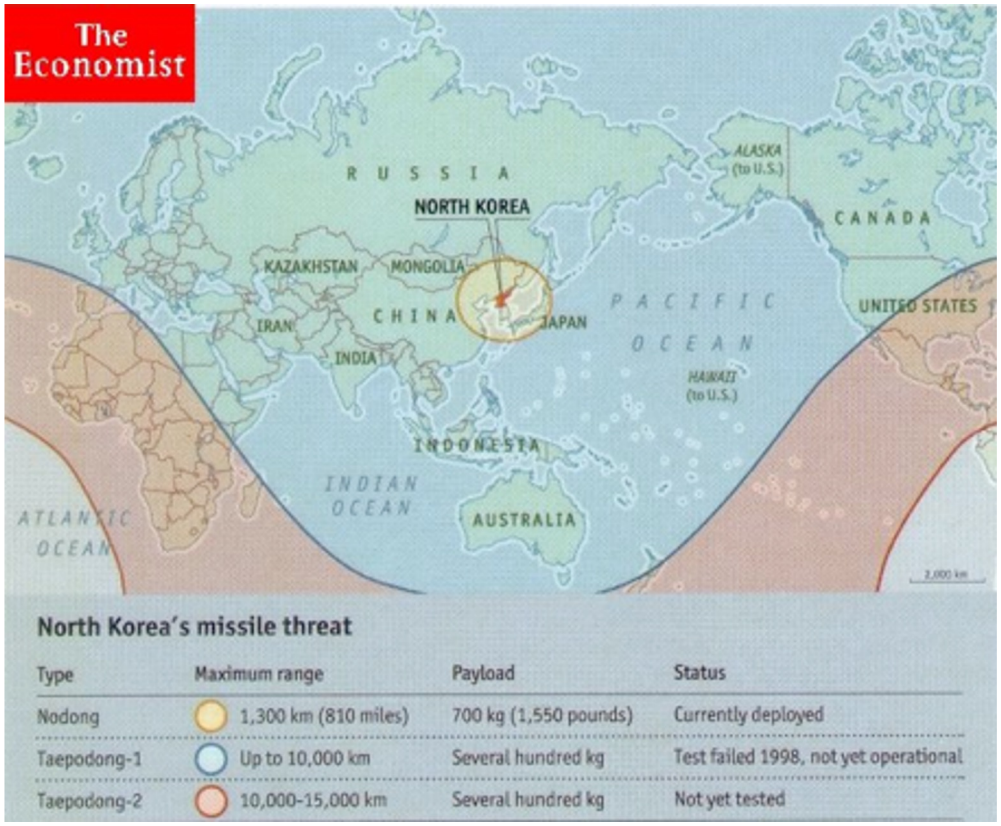
Example: Coordinate Systems & Projections Matter

Planar Computation Model



Published on May 3rd 2003 - **wrong**!

Correct Spherical Computation Model



Corrected on May 15th, 2003

WGS84 versus Australian GDA94

- WGS84 is used by the official GPS system
- The official geodetic datum (coordinate system) for Australia is GDA94 ("Geocentric Datum of Australia")
  - Based on IERS Terrestrial Reference Frame (ITRF), but fixed to a number of reference points in Australia.
- Difference between WGS84 and GDA94:
  - "The spheroids used for WGS84 and GDA84 are almost identical, and both systems are geocentric. Thus for most mapping, exploration and GIS uses, WGS84 and GDA94 coordinates will be the same. [...] For precise surveys, however, the difference between WGS84 and GDA94 may be significant, and changes slowly over time. [...] The difference between GDA94 and WGS84 is approximately 45cms in 2000."
    [http://www.geoproject.com.au/gda.faq.html]

Summary

- OpenGIS Consortium (OGC) Data Model
  - Standardised object-based representation of (geo-)spatial data

- Coordinate Systems
  - Important to interpret and relate geo-spatial data correctly

DB Support and Spatial Querying for (Geo-)Spatial Data: PostGIS

THE UNIVERSITY OF SYDNEY

Support for Spatial Data in PostgreSQL

- Postgresql supports **spatial data** out of the box
- Geometric types
  - Point, line, box, path, polygon, circle
  - https://www.postgresql.org/docs/current/static/datatype-geometric.html
- Geometric functions and operators
  - Intersect, contains, overlaps, distance etc
  - https://www.postgresql.org/docs/current/static/functions-geometry.html
- Indexing of geometric types
  - GiST
  - https://www.postgresql.org/docs/current/static/gist.html

```
CREATE INDEX census_idx ON CensusData USING GIST(geom);
```

- For **geographic data**: PostGIS extension (open source – https://postgis.net)

PostGIS

[http://postgis.net/documentation/]

- Spatial database extension for PostgreSQL supporting geographic objects
  - **Geometry types** for Points, LineStrings, Polygons, MultiPoints, etc.
    - including import/export from standard formats such as GeoJSON or KML
  - **Geography type** for geodetic data
  - Support for **spatial reference systems** and transformations between them
  - **Spatial predicates** on geometries
  - Spatial operators for determining **geospatial measurements** like area, distance, length and perimeter, and **geospatial set operations**, like union, difference etc.
  - **R-Tree indexing** (over GiST)

PostGIS Example

```
INSERT INTO superhero VALUES ('Catwoman',ST_SetSRID(ST_MakePoint(41.87,-87.634),4326));
…

SELECT superhero.name
  FROM city, superhero
 WHERE ST_Contains(city.geom, superhero.location)
   AND city.name = 'Gotham';
```

using **WGS**84

PostGIS: Geometry vs. Geography Type

See also: https://postgis.net/docs/using_postgis_dbmanagement.html#PostGIS_Geography

- **Geometry** type:
  - shapes on a *plane*; shortest path between two points is a straight line
- **Geography** type
  - Basis is a *sphere;* shortest path between two points is a circle arc
- This difference affects all calculations of geometries/geographies (areas, distances, lengths, intersections, …)
  - Because of more complex computations, less operations defined in PostGIS on **geographies** than on geometries
    - ST_Intersects(), ST_Area(), ST_Distance(), ST_Perimeter(), ST_CoveredBy()
    - those functions, that are defined, take more CPU time to execute than for geometries
  - Large-scale geodetic data (spanning countries or even continents) ideally represented and compared as geographies, in smaller scale geometry is fine
    - If needed, a geometry type can be converted on demand using `::geometry`

Operations on Spatial Objects in the Object Model

- Classifying operations
  - ***Set based***: 2-dimensional spatial objects (e.g. polygons) are sets of points
    - a set operation (e.g. intersection) of 2 polygons produce another polygon
  - ***Topological operations***: Boundary of USA touches boundary of Canada
  - ***Directional***: Brisbane is north of Sydney
  - ***Metric***: Brisbane is about 730 km from Sydney (by plane).

Set based	Union, Intersection, Containment
Toplogical	Touches, Disjoint, Overlap, etc.
Directional	East,North-West, etc.
Metric	Distance

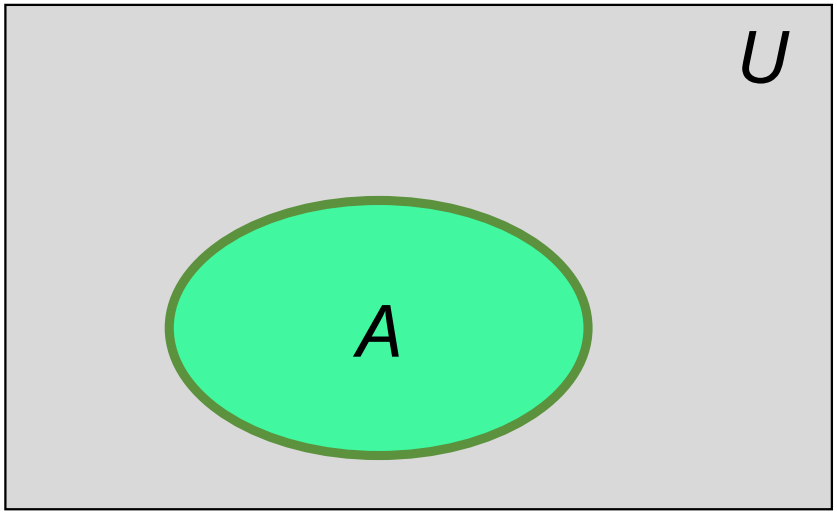
3种描述方式

Topological Relationships

- Topological Relationships
  - invariant under elastic deformation (without tear, merge).
  - Two countries that touch each other in a planar paper map will continue to do so in spherical globe maps.
- Topology is the study of topological relationships
- Example queries with topological operations
  - What is the topological relationship between two objects A and B ?
  - Find all objects which have a given topological relationship to object A ?

Topological Concepts

- Interior, boundary, exterior
  - Let A be an object in a "Universe" U.



Green is *A* **interior** (A°)

Blue is **boundary** of *A* (∂A)

Grey is *A* **exterior** (A^-)

  - Exterior is also referred to as the complement of an object

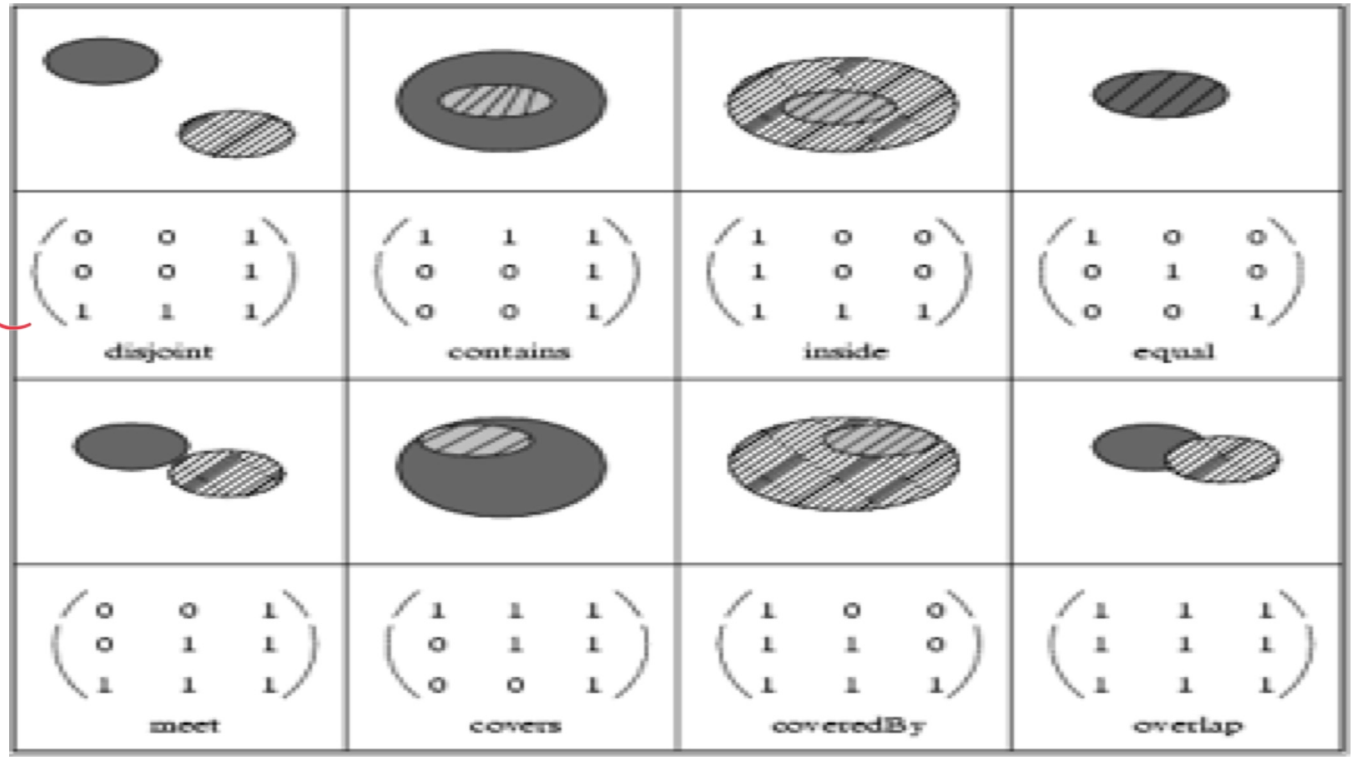
Nine-Intersection Model of Topological Relationships

- Many topological relationships can be specified using the **9-Intersection Model**
  - Eight possible 2D topological relationships for objects without holes
- Nine intersections between interior, boundary, exterior of A, B
  - A and B are spatial objects in a two-dimensional plane.
  - It can be arranged as a 3 by 3 matrix
  - Matrix elements take a value of 0 (false) or 1 (true).

$$\Gamma_9(A,B) = \begin{pmatrix} A^\circ \cap B^\circ & A^\circ \cap \partial B & A^\circ \cap B^- \\ \partial A \cap B^\circ & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^\circ & A^- \cap \partial B & A^- \cap B^- \end{pmatrix}$$

Nine-Intersection Model of Topological Relationships

Eight possible topological operations on 2D objects without holes



0 corresponds to = Ø, and 1 to ≠Ø

PostGIS Example

- Example query using topological relationship:

```
SELECT superhero.name
  FROM city, superhero
 WHERE ST_Contains(city.geom, superhero.location)
   AND city.name = 'Gotham';
```

Spatial Data with DuckDB

- **DuckDB** is a modern in-process SQL database with a portable design that targets analytical workloads and data science use cases
  - more lightweight than PostgreSQL, more powerful and faster than sqlite
- DuckDB has a spatial extension ("INSTALL spatial")
  - supports **geo-spatial data** following the OGC model
  - standard **spatial functions and predicates**, mostly compatible to PostGIS
    - DuckDB has no separate *geography* data type, but only **GEOMETRY** which by default assumes 'EPSG:4326' (WGS84) and which has some special functions to correctly calculate spherical distances and areas such as ST_Area_Spheroid()
    - R-Tree indexing support for faster spatial querying
  - Documentation: https://duckdb.org/docs/stable/core_extensions/spatial/overview
    https://duckdb.org/docs/stable/core_extensions/spatial/functions

Summary

- Querying spatial data with various operations, most importantly: Topological operations
- Nine-Intersection Model
- Supported via spatial extensions in many databases such as PostGIS or in SQL Server – also in Python via GeoPandas

GeoPandas

THE UNIVERSITY OF SYDNEY

Processing of GeoData in Python

- GeoPandas
  - http://geopandas.org
- Builds on and internally uses a number of other Python frameworks
  - **Shapely**: Manipulation and analysis of geometric objects in the Cartesian plane.
  - **GeoPandas**: geographic data
  - **Geos**: geospatial measures, functions and predicates (same core as PostGIS)
  - **Fiona**: library for reading&writing geo-formats, such as ESRI Shapefiles, Mapinfo TAB files, …
  - Libspatialite
  - Hdf5
  - …
- Might need post-installation with Anaconda: `conda install geopandas`

GeoPandas Data Types

- Provides two main geo data types
  - **GeoSeries** and **GeoDataFrame**
- **GeoSeries**
  - Main geometry building block on which GeoDataFrame is build
  - Series of GeoPandas **geometry** objects, which are an extension of a shapely.geometry object
  - Six classes, closely resembling the OGC Data Model
    - Single entity (core, basic types):
      - Point
      - Line
      - Polygon
    - Homogeneous entity collections:
      - Multi-Point, Multi-Line, Multi-Polygon

Example GeoSeries

Example from GeoHackWeek by UW [https://geohackweek.github.io/vector/]

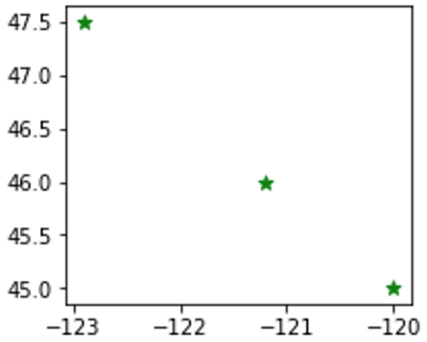
```
import geopandas as gpd
from shapely.geometry import Point
import matplotlib.pyplot as plt

# create a GeoSeries object from 3 Points
gs = gpd.GeoSeries([Point(-120,45), Point(-121.2,46), Point(-122.9,47.5)])

# check new object
Print(gs)
Print(type(gs), len(gs))

# set WGS84 reference system
gs = gs.set_crs('epsg:4326')

# visualise using MatplotLib.PyPlot
gs.plot(marker='*', color='green', markersize=100, figsize=(3, 3))
plt.xlim([-123, -119.8])
plt.ylim([44.8, 47.7]);
```



Quiz

- How many columns does gs contain?

GeoSeries 的设计和 pandas.Series 一样，本质上就是 一维数据结构，所以它 永远只有 1 列（geometry 列）。

GeoSeries Attributes and Methods

http://geopandas.org/data_structures.html

Attributes

- **area**: shape area (in units of current CRS projection)
- **bounds**: tuple of max and min coordinates on each axis for each shape
- **total_bounds**: tuple of max and min coordinates on each axis for entire GeoSeries
- **geom_type**: type of geometry.

Basic Methods

- **distance**(other): returns Series with minimum distance from each entry to other
- **centroid**: returns GeoSeries of centroids
- **to_crs()**: change coordinate reference system
- **plot**(): plot GeoSeries

Relationship Tests

- **contains**(other): is shape contained within other
- **intersects**(other): does shape intersect other
- …

CRS – Coordinate Reference System

http://geopandas.org/projections.html

- Similar to the SRID from OGC, GeoPandas uses a CRS to tell Python how coordinates relate to places on the Earth.
  - codes for most commonly used projections: www.spatialreference.org
  - one of the most commonly used CRS is WGS84
    - can be referred to by EPSG codes, e.g., for WGS84: "epsg:4326"
- Setting a Projection:
  - Can manually set a projection by assigning the correct codes to **crs** attribute:

```
my_geoseries.set_crs('epsg:4326')
```

  - You can see an object's current CRS through the crs attribute: **my_geoseries.crs**
  - Note: Data loaded from a reputable source (using the from_file() command) *should* always include projection information.
- Re-Projecting: geo-objects support a `to_crs(...)` function

```
Example: my_geoseries.to_crs(3395)
```

GeoDataFrame

- GeoDataFrame allows to combine normal data features and values with spatial data coming from GeoSeries

- The most important property of a GeoDataFrame is that it always has one GeoSeries column that holds a special status.
- This GeoSeries is referred to as the GeoDataFrame's "**geometry**".
- When a spatial method is applied to a GeoDataFrame (or a spatial attribute like **area** is called), this command will always act on the "**geometry**" column.

Example GeoDataFrame

Example from GeoHackWeek by UW [https://geohackweek.github.io/vector/]

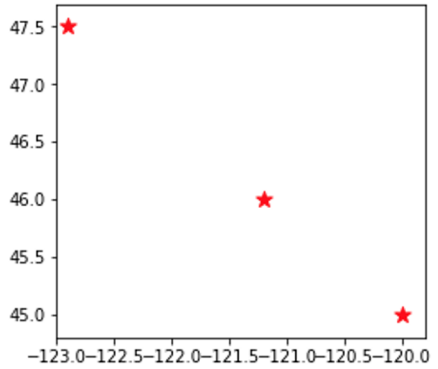
```
import pandas as pd
import geopandas as gpd
from shapely.geometry import Point

# create a standard DataFrame from some data including coordinates
data = {'name': ['a', 'b', 'c'],
        'lat': [45, 46, 47.5],
        'lon': [-120, -121.2, -122.9]}
df = pd.DataFrame(data)
print(df)
```

	name	lat	lon
0	a	45.0	-120.0
1	b	46.0	-121.2
2	c	47.5	-122.9

```
# use above data to create a GeoDataFrame with 'list-of-shapes'
geometry = [Point(xy) for xy in zip(df['lon'], df['lat'])]
gdf = gpd.GeoDataFrame(df, geometry=geometry)

# visualise using built-in plot functionality
gdf.plot(marker='*', color='red',
         markersize=50, figsize=(3, 3));
```



Plotting & Mapping

http://geopandas.org/mapping.html

- *geopandas* provides a high-level interface to matplotlib library for making maps.
- Mapping shapes is as easy as using the **plot()** method on a GeoSeries or GeoDataFrame.
  - Without parameters gives simple plot with random colors
  - Can be customized to liking
    - E.g. colors, legend, titles…
  - Maps (with the same CRS!) can be overlayed in one plot

Reading GeoSpatial Data From a Shape File + Plotting

Example from GeoHackWeek by UW [https://geohackweek.github.io/vector/]

GeoPandas.read_file() uses internally the Fiona library for reading and writing all kinds of geo-formats, such as ESRI Shapefiles, Mapinfo TAB files, …

```
import geopandas as gpd

# read some standard example dataset that come with geopandas
world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
cities= gpd.read_file(gpd.datasets.get_path('naturalearth_cities'))

world.plot()
```

Coordinate Reference System（坐标参考系统）

```
# make sure cities are in same CRS as world data
cities = cities.to_crs(world.crs)

# plot cities on top of base world map (2-layer plot)
base = world.plot(color='white', edgecolor='black')
cities.plot(ax=base, marker='o', color='red', markersize=5)
```

Reading GeoSpatial Data From PostGIS

Example from GeoHackWeek by UW [https://geohackweek.github.io/vector/]

- GeoPandas can read geometry data from PostGIS using **read_postgis()** command

```
import psycopg2
conn = psycopg2.connect(db_conn_dict)
…
seas = gpd.read_postgis("SELECT * FROM world_seas", conn, coerce_float=False)
conn.close()

seas.crs
seas.head()
seas.plot(column='oceans', categorical=True, legend=True, figsize=(14, 6));
```

Joining GeoPandas Data

http://geopandas.org/mergingdata.html

- Attribute Joins
  - Two GeoDataFrames can be joined on matching attribute values using the **merge()** function
  - Example: add country names to country_shapes using ISO codes

```
country_shapes = country_shapes.merge(country_names,on='iso_a3')
```

- Spatial Joins
  - **sjoin():** Merge two geometry objects based on their spatial relationship
  - Example:

```
cities_with_country = gpd.sjoin(cities, countries,
                                how="inner", op='intersects')
```

Summary

- Various support frameworks available in Python to import, represent or visualize spatial data
- Geo-Spatial data support in Python: GeoPandas
- Supports the OGC model including coordinate systems, standard topological relationships, and spatial joins
- Can import spatial data from various sources

Spatial Data Exchange using JSON and XML

THE UNIVERSITY OF SYDNEY

Geo-aware Web API

- Web APIs are very important for modern/mobile applications
  - RESTful APIs allow for low-latency dynamic applications
  - return results as JSON or XML
- Location-aware services nowadays are very popular
- Need to be able to represent spatial data in JSON or XML formats
  - **GeoJSON**
  - **KML**
  - many more…

Spatial Data Exchange Formats

- **GeoJSON**
- **KML**
- **OSM XML**
  - Open Street Map XML exchange format
  - https://wiki.openstreetmap.org/wiki/OSM_XML
- WMS
  - **Web Map Service** (**WMS**) by the Open Geospatial Consortium (OGC)
  - Other OGC standards for web geo-APIs:
    Web Feature Service (WFS) and Web Coverage Service (WCS) data
- **MapInfo TAB** format
- **Esri Shapefile** Format

GeoJSON

- GeoJSON is a format for encoding geographic data structures as JSON
  - Define spatial data as features of standard JSON objects
  - supports various geometry types
    - Point, LineString, Polygon, MultiPoint, MultiLineString and MultiPolygon
  - and additional properties
- Example:

```
{
   "type": "Feature",
   "geometry":  {
       "type": "Point",
       "coordinates": [125.6, 10.1]
   },
   "properties": {
       "name": "Dinagat Islands”
   }
}
```

[source: geojson.org]

- GeoJSON Specification: RFC7946 (Aug 2016)

GeoJSON Examples

- https://data.nsw.gov.au/data/dataset/nsw-rural-fire-service-current-incidents-feed

- http://geojson.io - website for authoring and sharing GeoJSON files
- geojsonlint.com - website that checks validity of GeoJSON data
- Python support library: geopandas

- Utility service:
  http://jsonprettyprint.net

Reading GeoSpatial Data From GeoJSON

```
import requests
import geojson
headers_dict = {'accept': 'application/text', 'authorization': '*****'}
opendata_url = "https://api.transport.nsw.gov.au/v1/life/hazards/incident/open"
r = requests.get(opendata_url, headers=headers_dict)

response = r.text.replace('POINT', 'Point') # quick hack to fix data for geojson
incidents_geo = geojson.loads(response)

print(type(incidents_geo))
print(incidents_geo.keys())

incidents_gdf = GeoDataFrame.from_features(incidents_geo)
incidents_gdf.head()
incidents_gdf.plot(ax=world.plot(cmap='Set3',figsize=(10,6)), marker='o', color='red', markersize=15);
bounds=incidents_gdf.geometry.bounds
plt.xlim([bounds.minx.min()-5, boiunds.maxx.max()+5]);
plt.ylim([bounds.miny.min()-5, boiunds.maxy.max()+5]);
```

GeoJSON as Output Format

- You can also use GeoJSON to display geometries on maps
- Example: Google Maps API
  - https://developers.google.com/maps/documentation/javascript/datalayer

```
var map;
function initMap() {
  map = new google.maps.Map(document.getElementById('map'), {
    zoom: 4,
    center: {lat: -28, lng: 137}
  });

  // NOTE: This uses cross-domain XHR, and may not work on older browsers.
  map.data.loadGeoJson('https://storage.googleapis.com/mapsdevsite/json/google.json');
}
```

- Alternatively on OpenStreetMap: https://www.openstreetmap.org/

KML – the Keyhole Markup Language

- Spatial file format originally developed for Google Earth
  - Since 2008 international standard of the Open Geospatial Consortium
- XML variant, more widely used now
  - KML files specify a set of features (place marks, images, polygons, 3D models, textual descriptions, etc.) for display in spatial software
- Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>
 <Placemark>
   <name>New York City</name>
   <description>New York City</description>
   <Point>
      <coordinates>-74.006393,40.714172,0</coordinates>
   </Point>
 </Placemark>
</Document>
</kml>
```

- Documentation:
  - https://developers.google.com/kml/documentation/kml_tut

Example Usages

- https://www.ga.gov.au/data-pubs
- https://www.data.gov.au/search?format=KML
  - Eg. “City of Gold Coast Road Closures”
  - or “Ipswich City Boundary”

    => also available as GeoJSON

    => Possible to visualise with eg., http://geojson.io
    (there is also a Python library for this called **geojsonio**)

- There are currently already hundreds of open spatial data sets available via data.gov.au which use either KML or GeoJSON

Summary

- Ingest Data with spatial features
  - Such as geographic location (point data) or a geometry (spatial object)
  - Store and index in database for later processing
    - **Spatial indexing** is important
    - make sure coordinate systems (SRID) are compatible
  - Be aware of **spatial query types** and **spatial joins**

- Use Web APIs with support for GeoJSON to lookup further data
  - Eg. Boundaries of localities
  - Parsing GeoJSON required => semi-structured data techniques of last week

- Use GeoJSON or KML to export or visualize spatial data