

# A Brief Overview of GIS

Masoomali Fatehkia  
Science Monday  
14 February 2022

# Geometric objects

# Geometric object types

## Vector data

- Point
- Polygon/Multipolygon

## Raster data

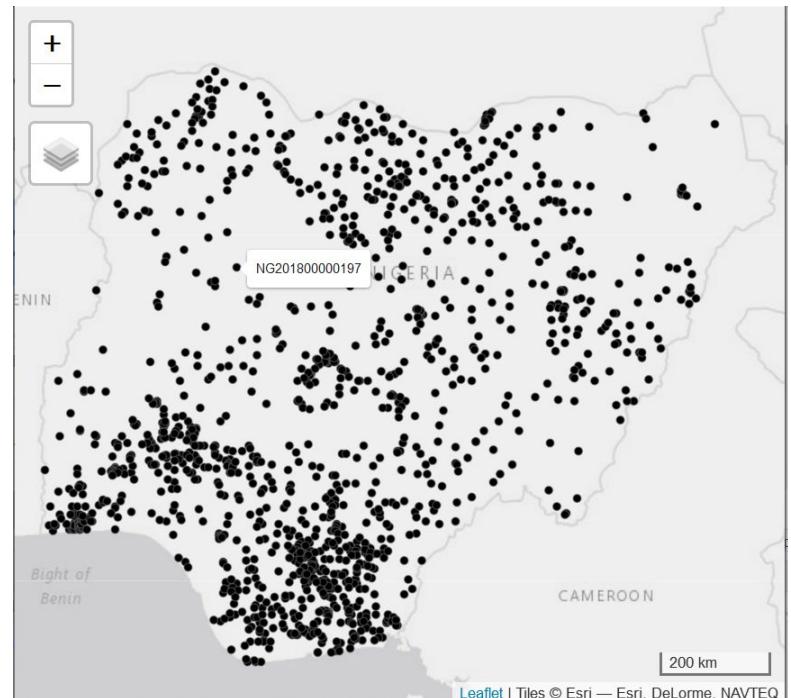
- Raster

# Point

Latitude and longitude pairs

*Examples:*

- Surveyed locations
- Reported crime incidents
- Geo-tagged tweets

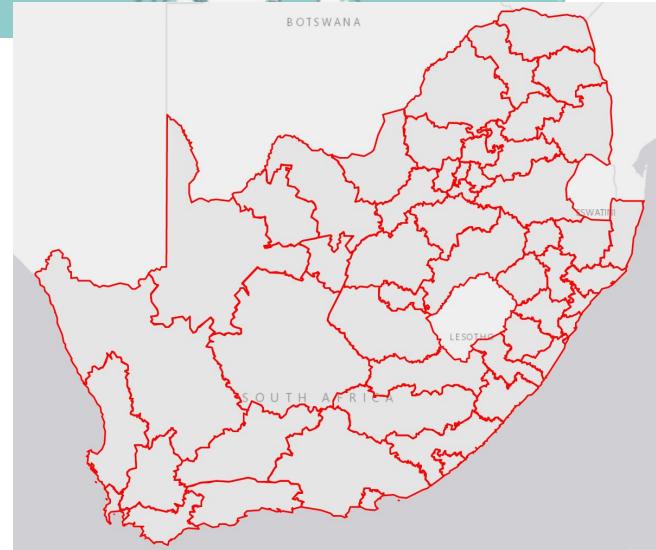


# Polygon/Multipolygon

A collection of points

*Examples:*

- Countries
- Subnational admin. regions in a country



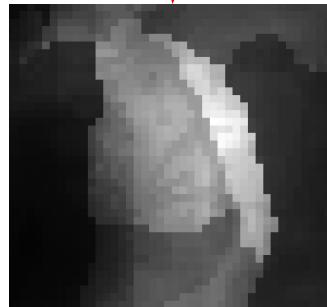
# Raster

Grid (rows and columns) where each cell has a value.

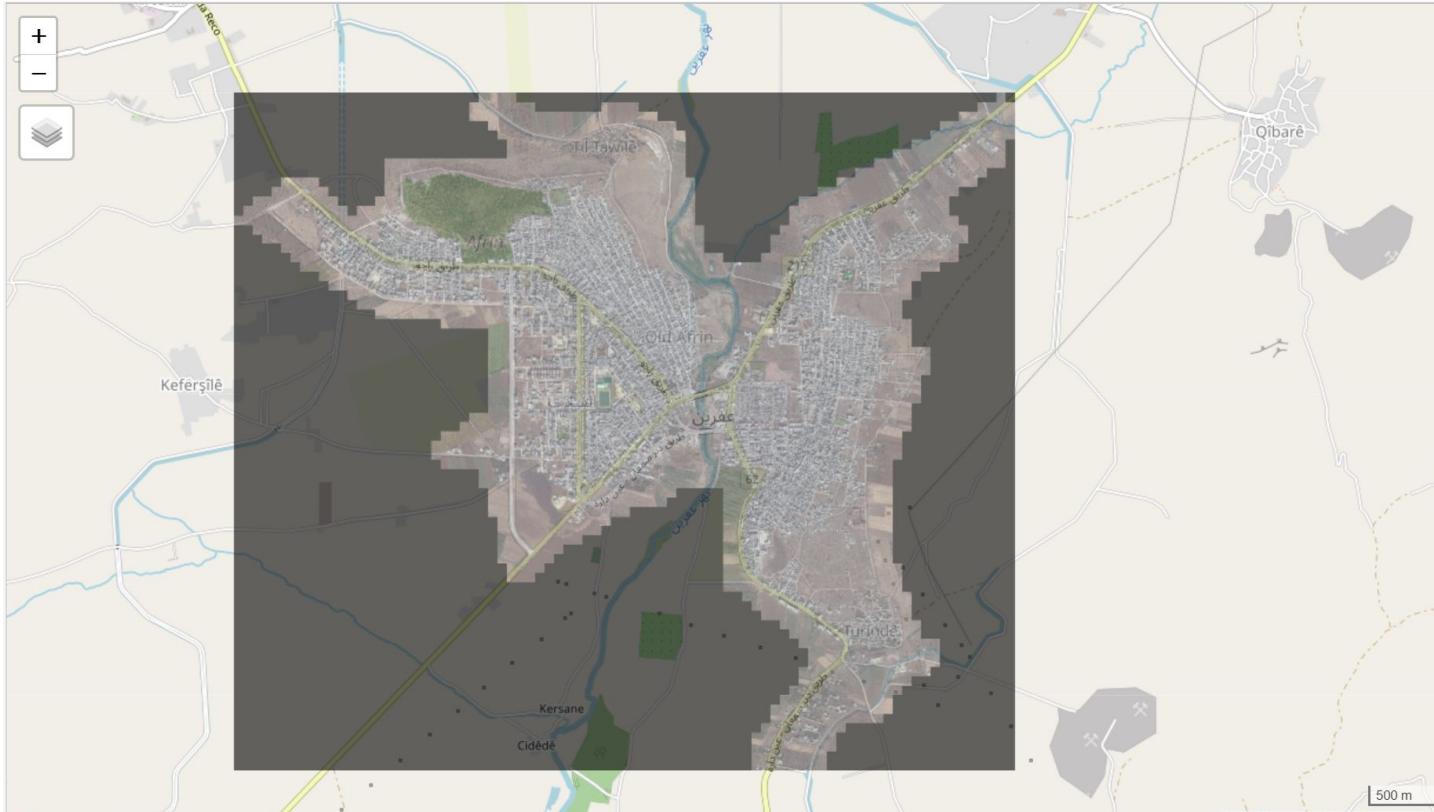
Origin, extent (number of rows/columns & cell size resolution)

*Examples:*

- Gridded population data (wpop, FB-HRSL)
- Nightlights
- Satellite imagery



# Raster: Satellite Imagery



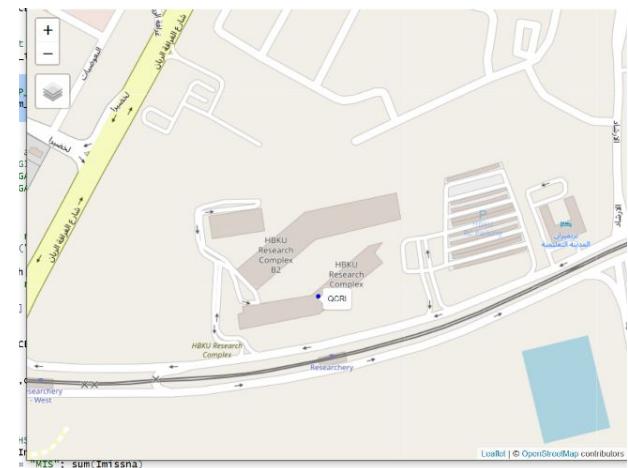
# Coordinate Reference System (CRS)

What do these coordinates (lat,long) mean?

1. (25.32155, 51.42576)
2. (2800622, 542851.6)

They are the same point (QCRI) in different CRS:

1. WGS84 CRS -> *geographic coordinates*
2. UTM (zone 39) CRS -> *projected coordinates*



# Coordinate Reference System (CRS)

CRS gives details about the model of the earth/reference points used and what type of projection is used if projecting from 3D to 2D.

Why do we care?

- Different sources may use different CRS.
- Affects how spatial operations are performed. May need to transform your data for certain analysis.

*Two categories:*

- Geographic coordinate reference system
- Projected coordinate reference system

# Coordinate Reference System (CRS): Geographic

Coordinate pair in degrees (Latitude & longitude) and

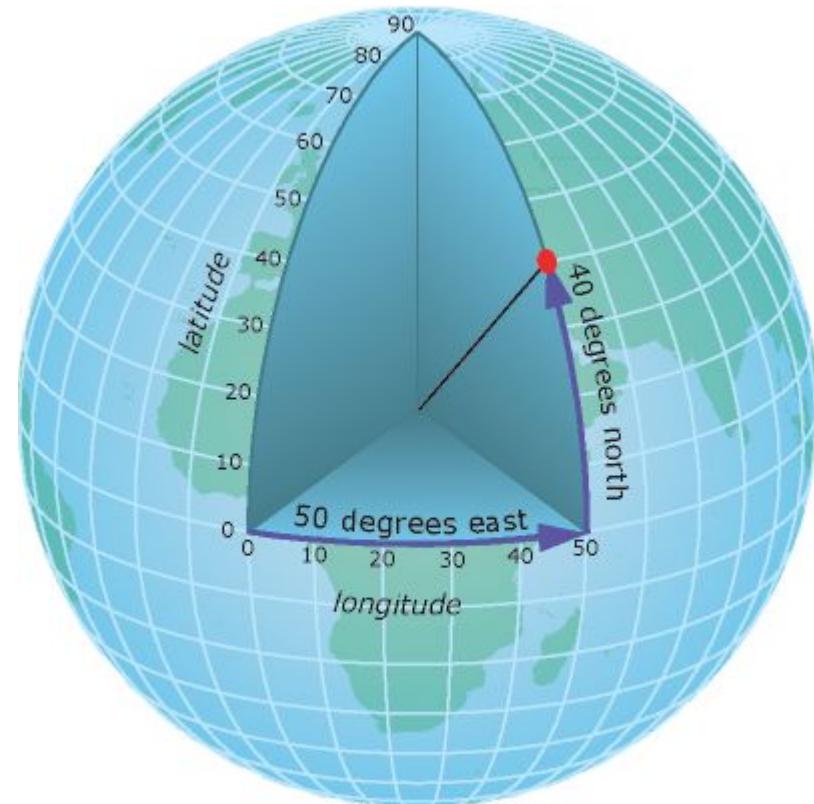
Datum (model of the earth used)

*Examples:*

WGS84 - World Geodesic System

1984

NAD83 - North American Datum 1983



# Coordinate Reference System (CRS): Projected

Represent a 3D earth as a 2D plane (map)

Different projections exist.

Each projection preserves some properties at the cost of distorting others:  
distance, area, shape.

Need to choose best one for your use case.

*Examples*

Universal Transverse Mercator (UTM), Azimuthal equal area, Azimuthal equidistant, ....

# Coordinate Reference System (CRS) Formats

How are CRS information stored?

Multiple formats:

- **Well-known Text (WKT)**: compact representation of the elements of a CRS
- Proj.4
- Identifying code: 'AUTHORITY:CODE' eg: EPSG:4326
- Others...

# Coordinate Reference System (CRS) Formats

Search for different CRS and their various formats:

<https://spatialreference.org/>

The screenshot shows the Spatial Reference website with the projection set to 4326. The page includes a map of the world, navigation links (Home, Upload Your Own, List user-contributed references, List all references), and a search bar. Below the navigation is a detailed JSON representation of the WGS 84 CRS definition:

```
GEOGCS["WGS 84",
    DATUM["WGS_1984",
        SPHEROID["WGS 84",6378137,298.257223563,
            AUTHORITY["EPSG","7030"]],
        AUTHORITY["EPSG","6326"]],
    PRIMEM["Greenwich",0,
        AUTHORITY["EPSG","8901"]],
    UNIT["degree",0.01745329251994328,
        AUTHORITY["EPSG","9122"]],
    AUTHORITY["EPSG","4326"]]
```

+proj=longlat +ellps=WGS84 +datum=WGS84 +no\_defs

The screenshot shows the Spatial Reference website for EPSG:4326. It includes a map, navigation links, and a large section titled "EPSG:4326". Below it is a "WGS 84 (Google it)" link. To the right is a list of "Well Known Text" formats, each with a red arrow pointing to the "Well Known Text as HTML" link.

- [Well Known Text as HTML](#)
- [Human-Readable OGC WKT](#)
- [Proj4](#)
- [OGC WKT](#)
- [JSON](#)
- [GML](#)
- [ESRI WKT](#)
- [.PRJ File](#)
- [USGS](#)
- [MapServer Mapfile | Python](#)
- [Mapnik XML | Python](#)
- [GeoServer](#)
- [PostGIS spatial\\_ref\\_sys INSERT statement](#)
- [Proj4js format](#)

# File types: ESRI shapefiles

Consists of multiple files (.shp, .shx, .dbf, .prj etc.)

Partially-open

Limitations on data column names, file sizes



Name	Date modified	Type	Size
Ibn_admbnda_adm0_cdr_20160822.cpg	7/12/2020 2:41 PM	CPG File	
Ibn_admbnda_adm0_cdr_20160822.dbf	7/12/2020 2:41 PM	DBF File	
Ibn_admbnda_adm0_cdr_20160822.prj	7/12/2020 2:41 PM	PRJ File	
Ibn_admbnda_adm0_cdr_20160822.sbn	7/12/2020 2:41 PM	SBN File	
Ibn_admbnda_adm0_cdr_20160822.sbx	7/12/2020 2:41 PM	SBX File	
<input checked="" type="checkbox"/> Ibn_admbnda_adm0_cdr_20160822.shp	7/12/2020 2:41 PM	SHP File	
Ibn_admbnda_adm0_cdr_20160822.shp	7/12/2020 2:41 PM	XML Document	
Ibn_admbnda_adm0_cdr_20160822.shx	7/12/2020 2:41 PM	SHX File	

# File types: GeoJSON

Extension of the JSON format

File extension: .geojson

```
1  [ { "type": "FeatureCollection",
2    "features": [
3      { "type": "Feature",
4        "geometry": { "type": "Point", "coordinates": [102.0, 0.5] },
5        "properties": { "prop0": "value0" }
6      },
7      { "type": "Feature",
8        "geometry": {
9          "type": "Polygon",
10         "coordinates": [
11           [ [100.0, 0.0], [101.0, 0.0], [101.0, 1.0],
12             [100.0, 1.0], [100.0, 0.0] ]
13         ]
14       },
15       "properties": {
16         "prop0": "value0",
17         "prop1": { "this": "that" }
18       }
19     }
20   ]
21 }
22 }
```

# File types: Many others

- KML
  - .kml file (XML based format)
  - Developed for use with Google Earth
- GPX: .gpx
- FlatGeobuf: .fgb
- SQLite/SpatialLite: .sqlite
- ESRI FileGDB: .gdb
- GeoPackage: .gpkg

# File types: Raster

## GeoTIFF:

- .tif/.tiff file
- A TIFF file containing additional spatial metadata

## ArcASCII:

- .asc
- Text format data

# Non-standard files/inputs

Text file with list of coordinates? CSV file with latitude and longitude columns?

- Write your own scripts/functions to parse the data.
- Then pass these to object constructors of libraries in R/python. Also include the CRS of the data.
- Save output as geojson/shapefile.

[Python] shapely, GeoPandas

[R] sp/sf, rgdal

# Some useful online tools

Create your own polygons/shapes

<https://geojson.io>

```
1 {  
2   "type": "FeatureCollection",  
3   "features": [  
4     {  
5       "type": "Feature",  
6       "properties": {},  
7       "geometry": {  
8         "type": "Polygon",  
9         "coordinates": [  
10            [  
11              [51.40348434448242,  
12                25.309269760067775],  
13              [51.406660079956055,  
14                25.307795501346572],  
15              [51.45343780517578,  
16                25.317571829456185],  
17              [51.452579498291016,  
18                25.309269760067775]  
19            ]  
20          ]  
21        }  
22      }  
23    ]  
24  }  
25 }
```

Quickly view/validate a geojson

<https://geojsonlint.com/>

```
{  
  "type": "FeatureCollection",  
  "features": [  
    {  
      "type": "Feature",  
      "properties": {},  
      "geometry": {  
        "type": "Polygon",  
        "coordinates": [  
          [  
            [51.40348434448242,  
             25.309269760067775],  
            [51.406660079956055,  
             25.307795501346572],  
            [51.45343780517578,  
             25.317571829456185],  
            [51.452579498291016,  
             25.309269760067775]  
          ]  
        ]  
      }  
    }  
  ]  
}
```

GeoJSONLint   Point ▾   LineString ▾   Polygon ▾   Feature ▾   GeometryCollection

Use this site to validate and view your GeoJSON. For details about GeoJSON, [read the spec](#).

Clear Current Features

# Some sources of geospatial data

- Country administrative regions: [gadm.org](http://gadm.org)
- Nightlights: <https://eogdata.mines.edu/products/vnl/>
- High-res gridded population maps:
  - Worldpop (<https://worldpop.org/>)
  - FB HRSL (<https://research.facebook.com/downloads/high-resolution-settlement-layer-hrsl/>)
- Many others, eg: Individual country agencies/open data portals
- Compilations of links to various free online GIS datasets:
  - <http://freegisdata.rtwilson.com/>
  - <http://bit.ly/1c3yZCs>

# Manipulating spatial data

# Polygon simplification

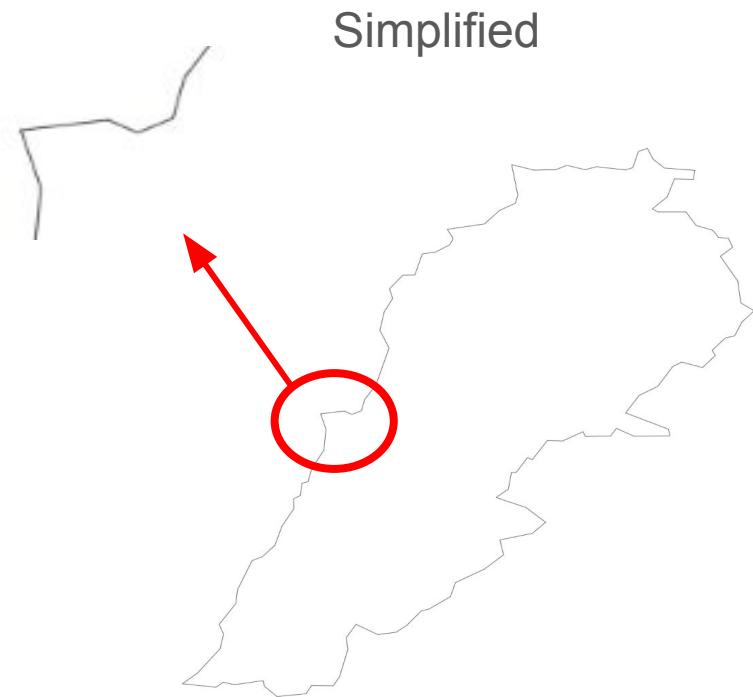
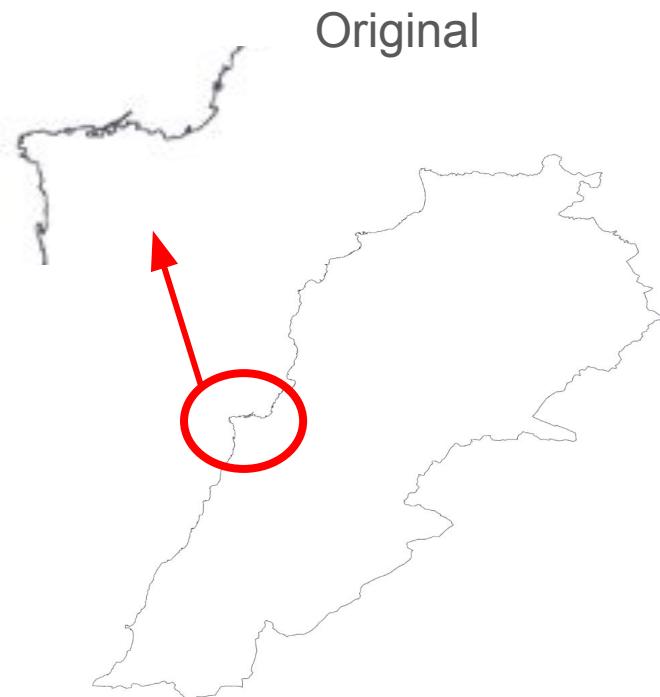
Reduce the number of vertices in a polygon.

Makes the object simpler, takes less memory and can be faster to load/visualize (eg: in online app).

Can be done by libraries in R/Python.

Online tools: <https://mapshaper.org/>

# Polygon simplification



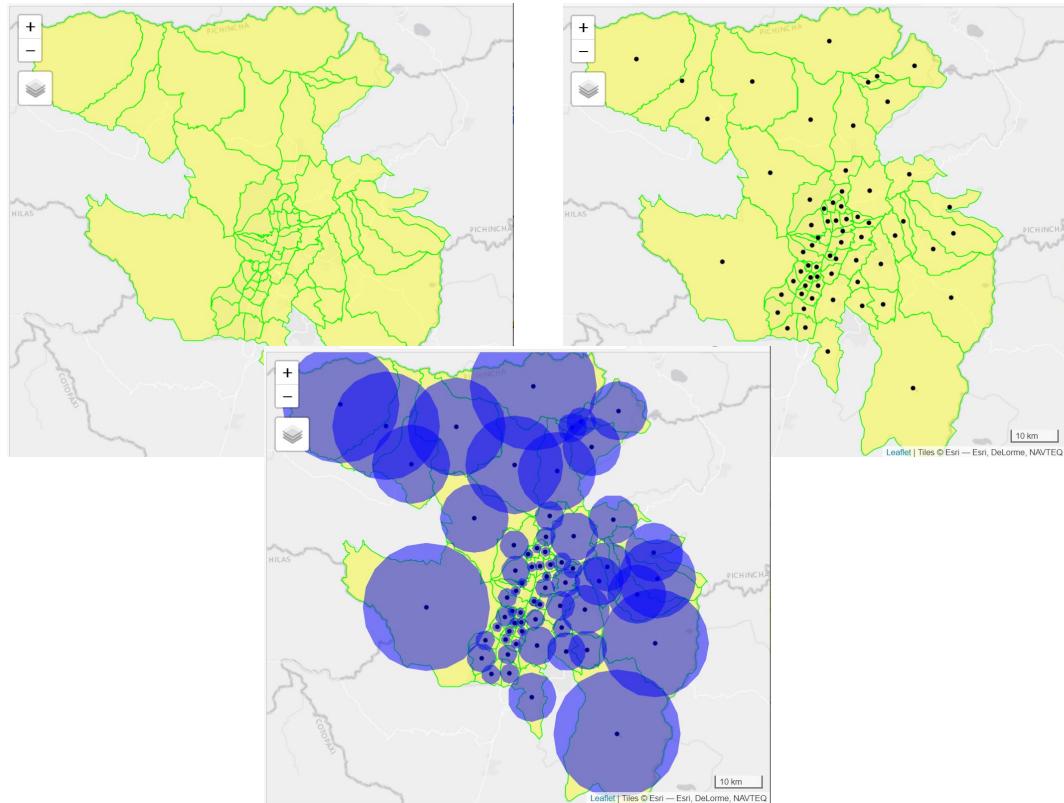
# Centroids

Identify the center of a geographic object.

Represents a more complex object as a single point.

Different techniques exist.

Maybe needed for some analysis,  
eg: estimating distances between polygons

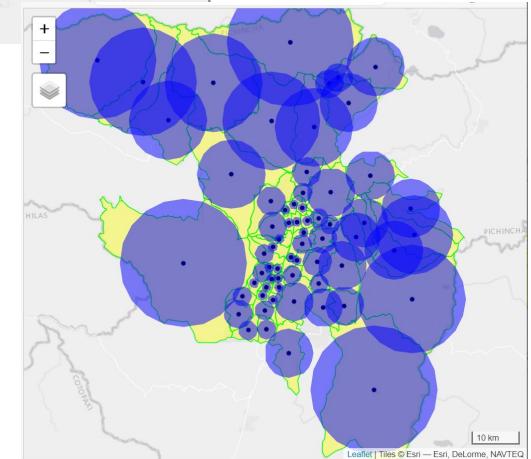
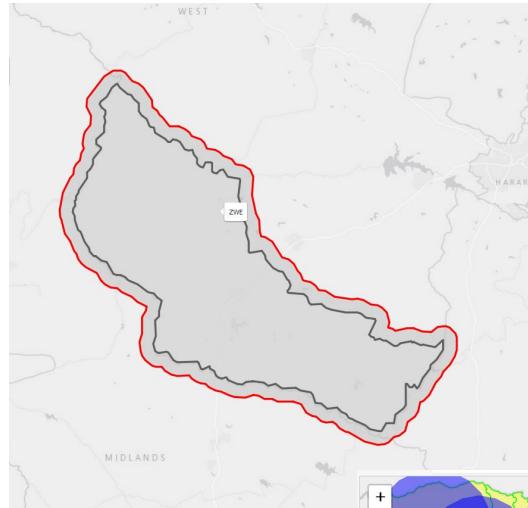


# Buffers

Area within given distance of geometric object

Get points within a given distance of a polygon

DHS spatial noise. Draw buffer around survey data when merging with other datasets.

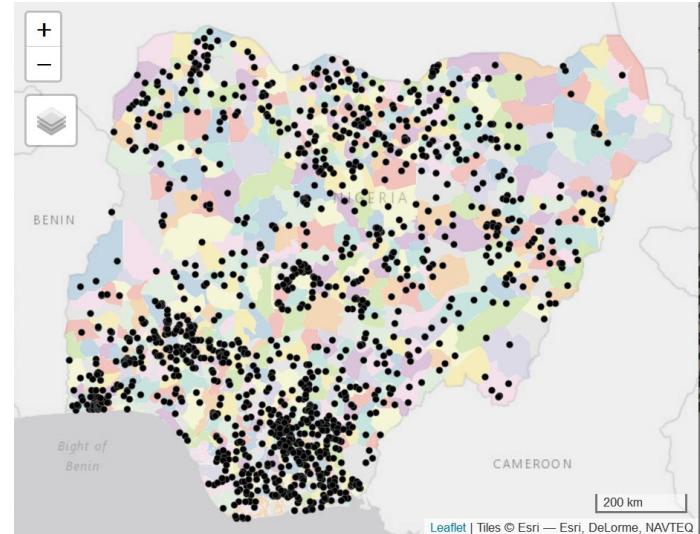


# point-in-polygon

Aggregating points data to higher geographic units:

- Aggregating counts from geotagged tweets
- Aggregating reported incidents
- Aggregating from survey data for each region in a country

Merging data from higher level admin units with the points data

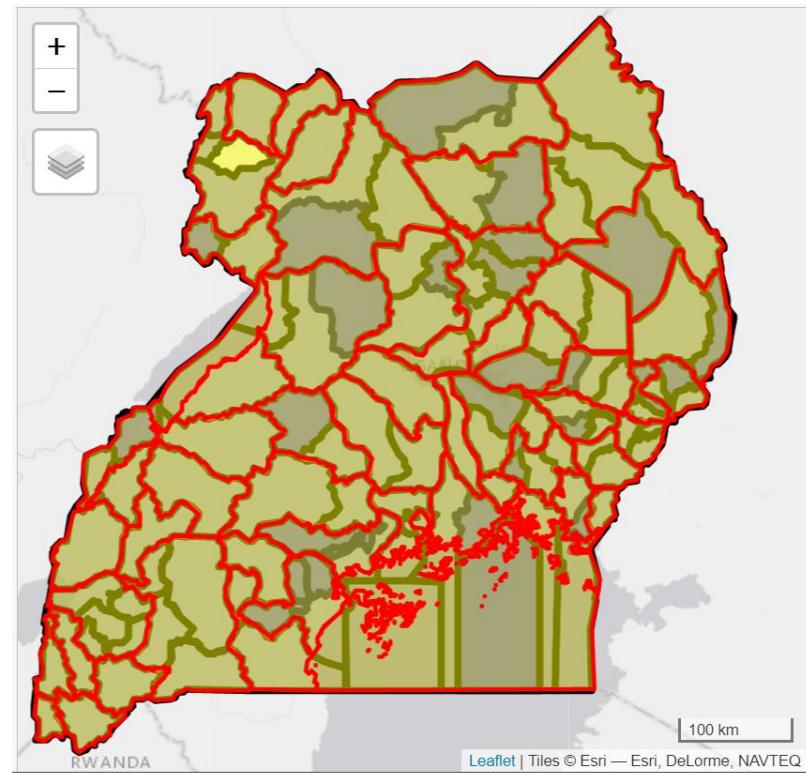


# polygon intersections

Merging data from different sources at different spatial granularity.

Common operations:

- intersection
- Union
- Difference



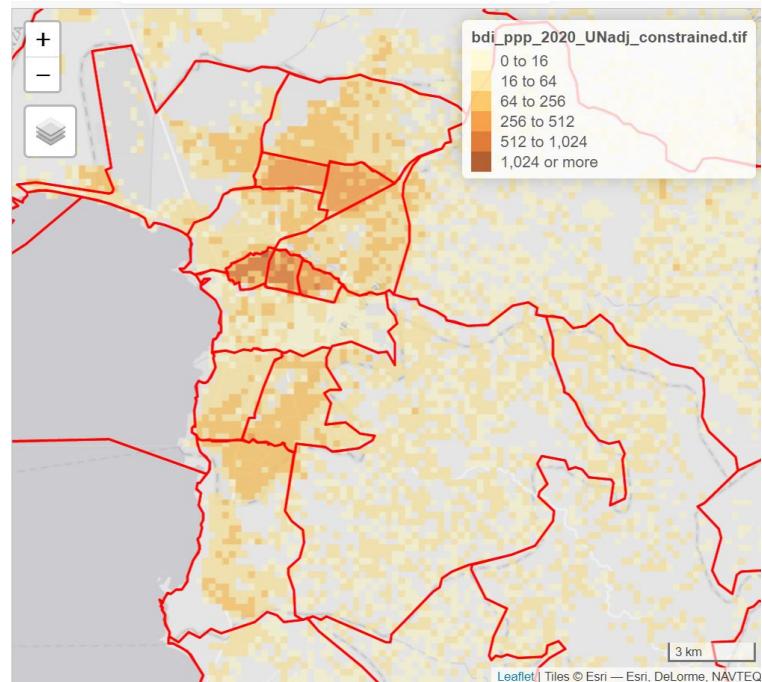
# Raster aggregation

Get **total** population for subnational regions from high-res population maps

Compute **mean** nightlight intensity.

[Python] [rasterio](#)

[R] raster, exactextractr



# Raster aggregation/disaggregation & resampling

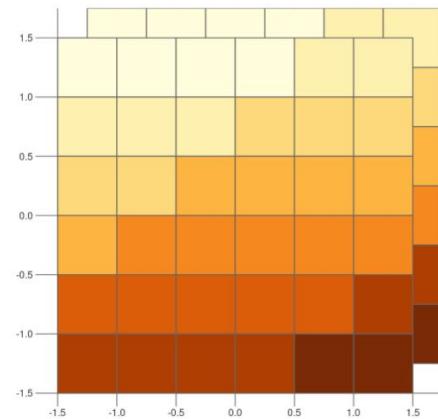
Combining data from multiple rasters. Eg: **mean pop. weighted nightlights**:

- nightlights raster (500m res)
- population raster (100m res)

Need rasters to be aligned.

Aggregate pop. raster & then resample.

Different techniques exist. Check documentation for library you are using.



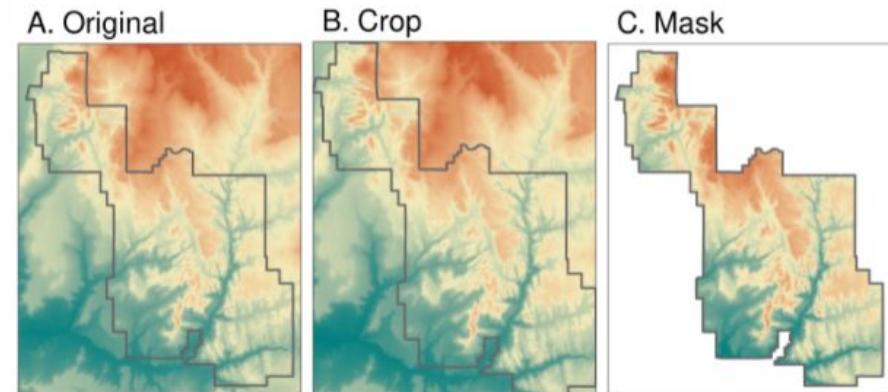
# Raster cropping/masking

## Cropping

- Exclude parts of the raster than are not needed
- Eg: nightlights intensity for the World but our analysis is for Qatar only

## Masking

- ‘Delete’ values outside your polygon  
(set them to 0)



# Creating map visualizations

Interactive visualizations/exploratory analysis

Maps for publications

[**R**] tmap, ggplot

[**Python**] Folium

Open-source software: QGIS

# Some useful resources

Geospatial processing in R: <https://geocompr.robinlovelace.net/>

Geospatial analysis with python:

<https://kodu.ut.ee/~kmoch/geopython2018/index.html>