

# UCL-UoG International Workshop

## Day 1: Introduction to Geoprocessing in RStudio

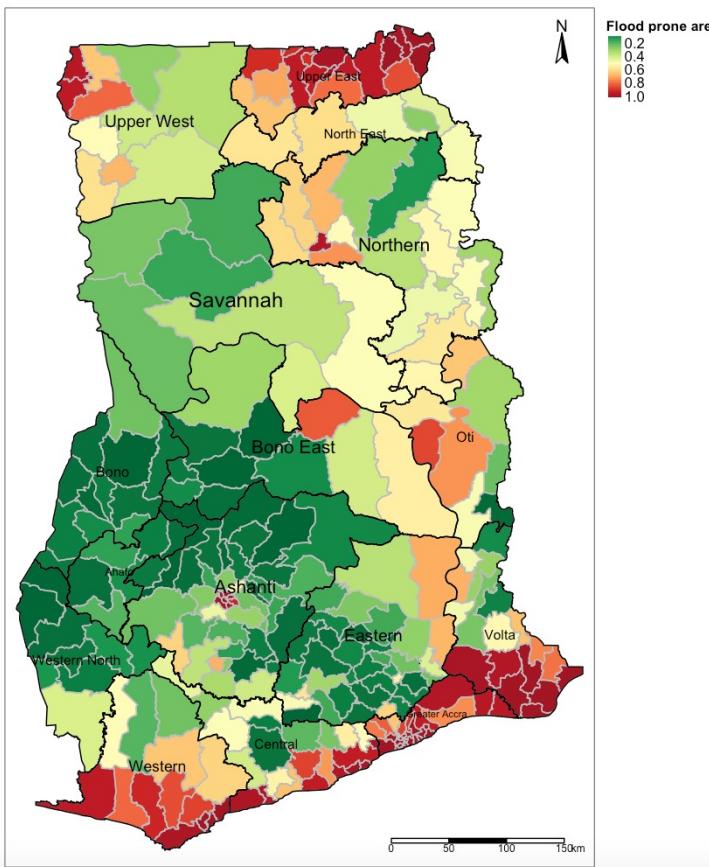
Team: Dr. Anwar Musah, Prof. Luiza Campos, Dr. Mumuni Abu, Dr. Stephen Law

# What will we learn?

## What will we learn?

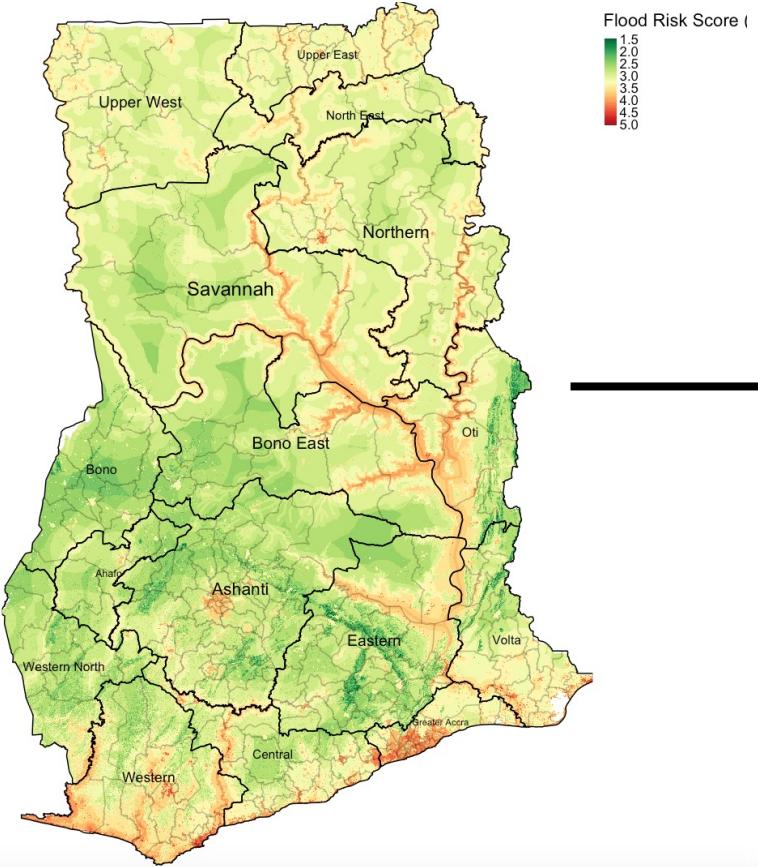
- To introduce you to etiquettes of RStudio and its programming environment
- You will be introduced to the various types of spatial data
- For the day 1 practical – you will be shown how to merge non-spatial attribute data with that this spatial. The focus will be on “vector data”
- You will be shown how to use the “tmap” and “sf” packages to perform various geoprocessing to visualise areal data from a shapefile. This particular dataset will focus on flood hazards across Ghana.

## Day 1 (Vector)

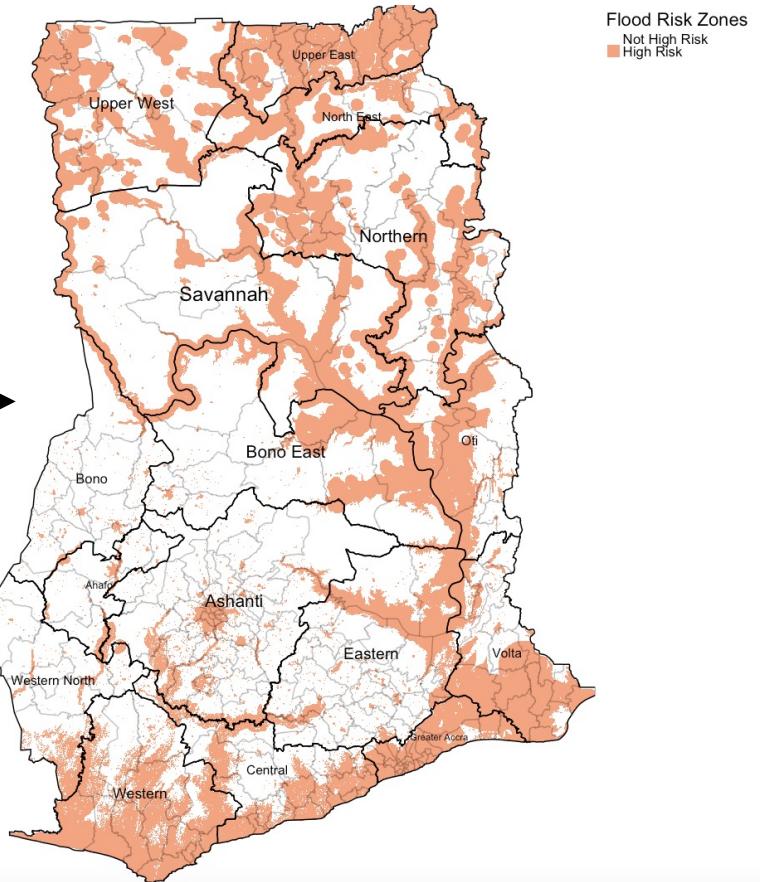


Mapping burden of surface affected by floods in districts in Ghana

## Day 2 (Raster)

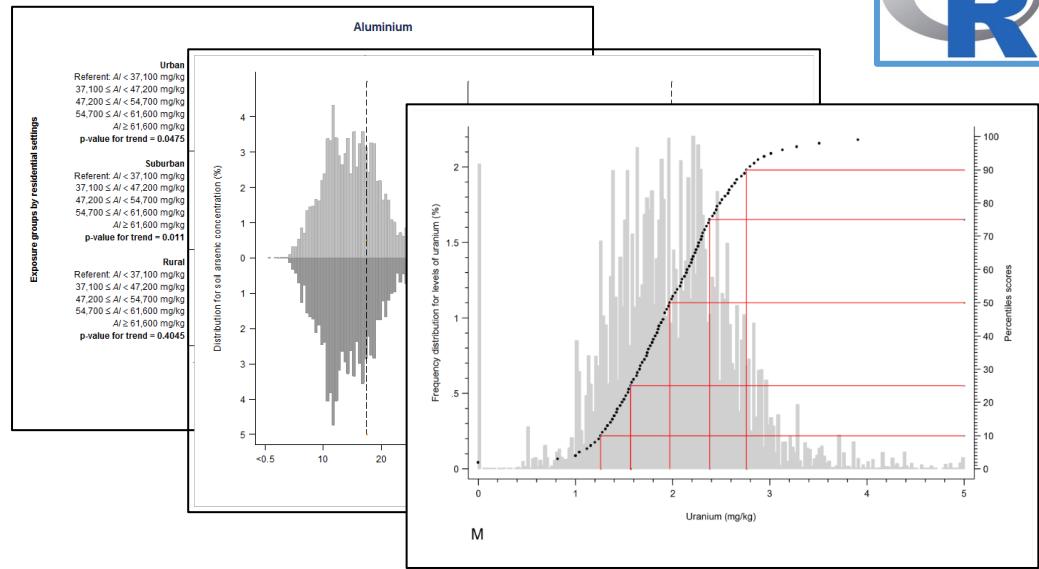
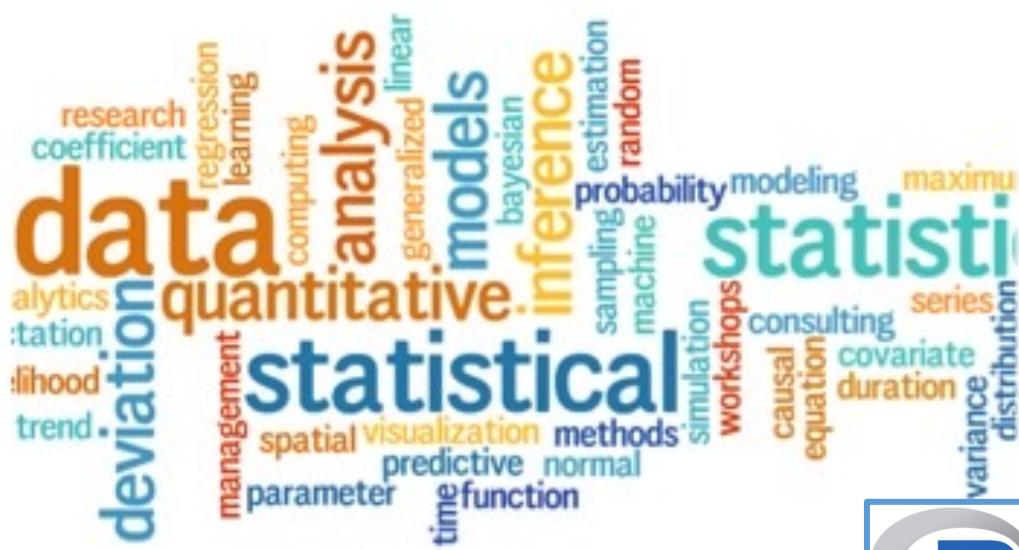


Compute risk score for pixels across Ghana based on a set of environmental variables with decisions



Reclassify risk scores based on a threshold (> 4, high risk and above). Zone the areas in Ghana

# What is RStudio?

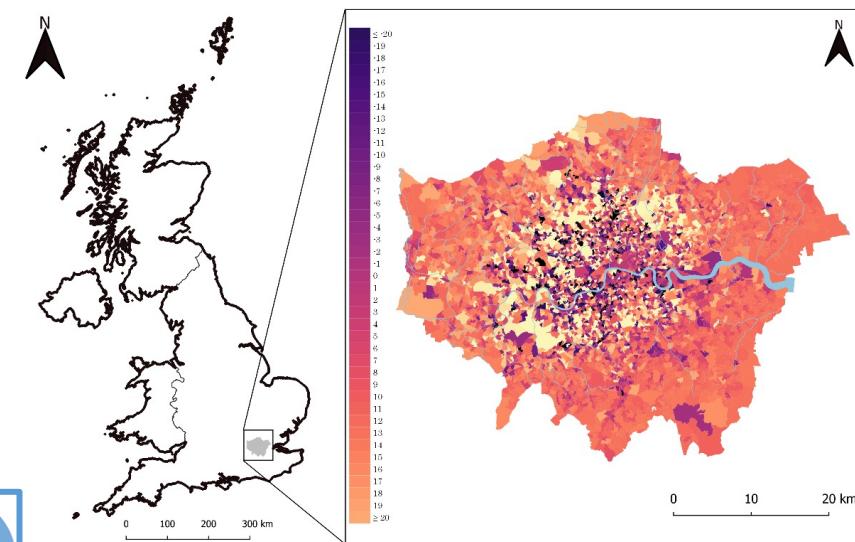


```

181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
  
```

```

(i == 2010 & j == 1) {
  file <- paste0("/Users/anwarmusah/Desktop/AM_Zika2019/Data/Brazil/Climatic/Temperat
raster_file <- raster(file)
recife_temperature_cropped <- crop(raster_file, recife_extent)
recife_temperature_masked <- mask(recife_temperature_cropped, bra_recife_outline)
recife_temperature_masked <- projectRaster(recife_temperature_masked, crs=proj4)
recife_temp_aggr <- extract(recife_temperature_masked, bra_recife_areas, fun=mean, d
colnames(recife_temp_aggr)[1] <- "fid"
colnames(recife_temp_aggr)[2] <- "temperature"
colnames(recife_temp_aggr)[3] <- "district_id"
recife_temp_aggr$year <- i
recife_temp_aggr$month <- j
recife_temperature <- recife_temp_aggr[,c(1,3,4,5,2)]
}
else {
  file <- paste0("/Users/anwarmusah/Desktop/AM_Zika2019/Data/Brazil/Climatic/Temperat
raster_file <- raster(file)
recife_temperature_cropped <- crop(raster_file, recife_extent)
recife_temperature_masked <- mask(recife_temperature_cropped, bra_recife_outline)
recife_temperature_masked <- projectRaster(recife_temperature_masked, crs=proj4)
recife_temp_aggr <- extract(recife_temperature_masked, bra_recife_areas, fun=mean, d
colnames(recife_temp_aggr)[1] <- "fid"
colnames(recife_temp_aggr)[2] <- "temperature"
colnames(recife_temp_aggr)[3] <- "district_id"
recife_temp_aggr$year <- i
recife_temp_aggr$month <- j
recife_temperature <- recife_temp_aggr[,c(1,3,4,5,2)]
}
  
```





PROJ.4





PROJ.4





**R (Standard)**

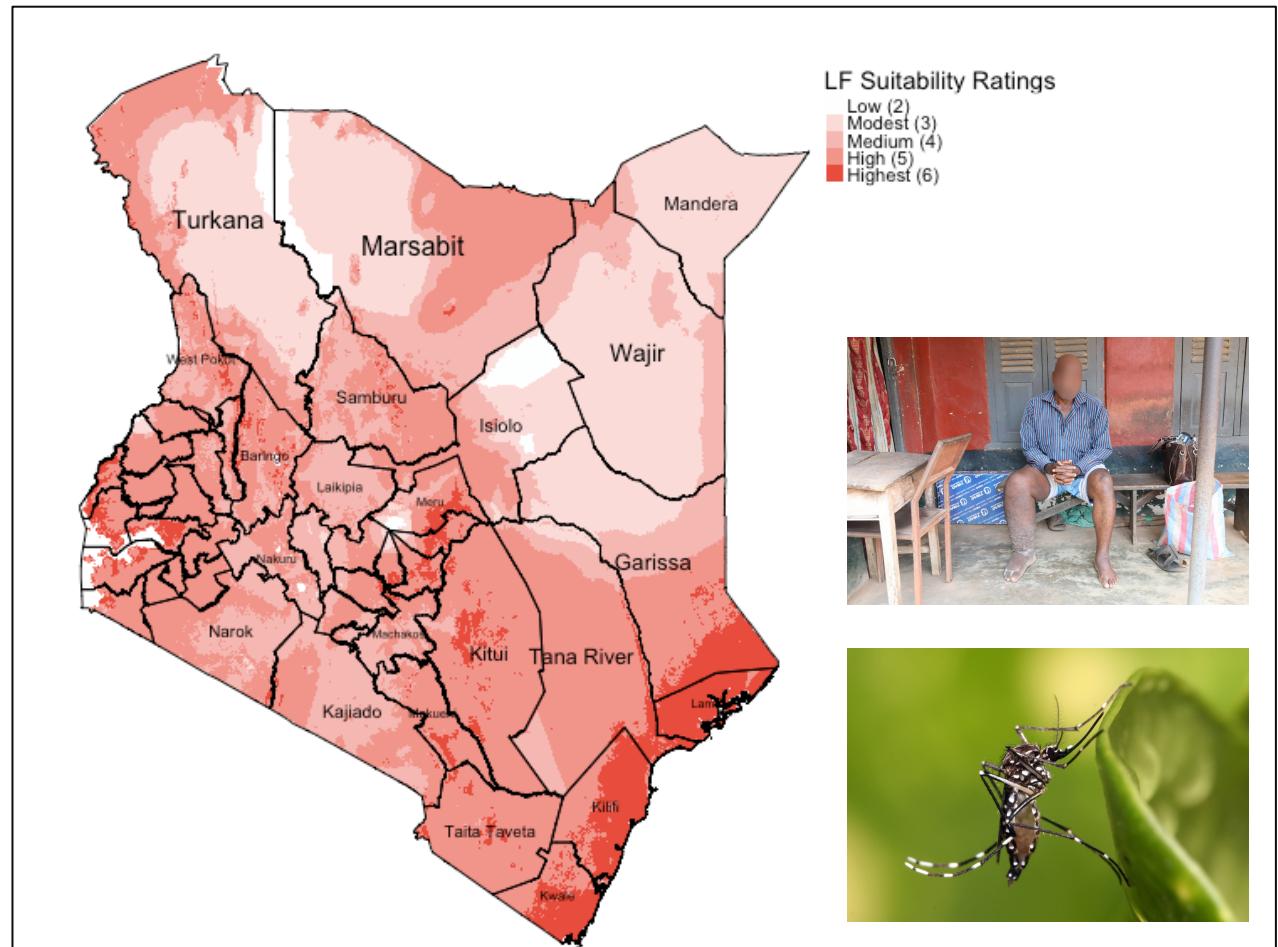


**RStudio (Best)**

There are two version of the software: 1.) R, and 2.) **RStudio**; The second is much preferred as its straightforward and intuitive.

# Why are we teaching RStudio?

1. Flexible and provides access to powerful packages for analysis
2. Impressive graphs, visualizations and maps
3. Excellent statistical capabilities too



**Example:** Map generated in R to illustrate areas that are environmentally suitable for the spread of neglected tropical disease called 'Lymphatic Filariasis (LF)' in Kenya.

## Sources:

1. Global Atlas for Helminths Infection (<http://www.thiswormyworld.org>)
2. ESPEN (<https://espen.afro.who.int>)

# ... and why learn how to code in RStudio?

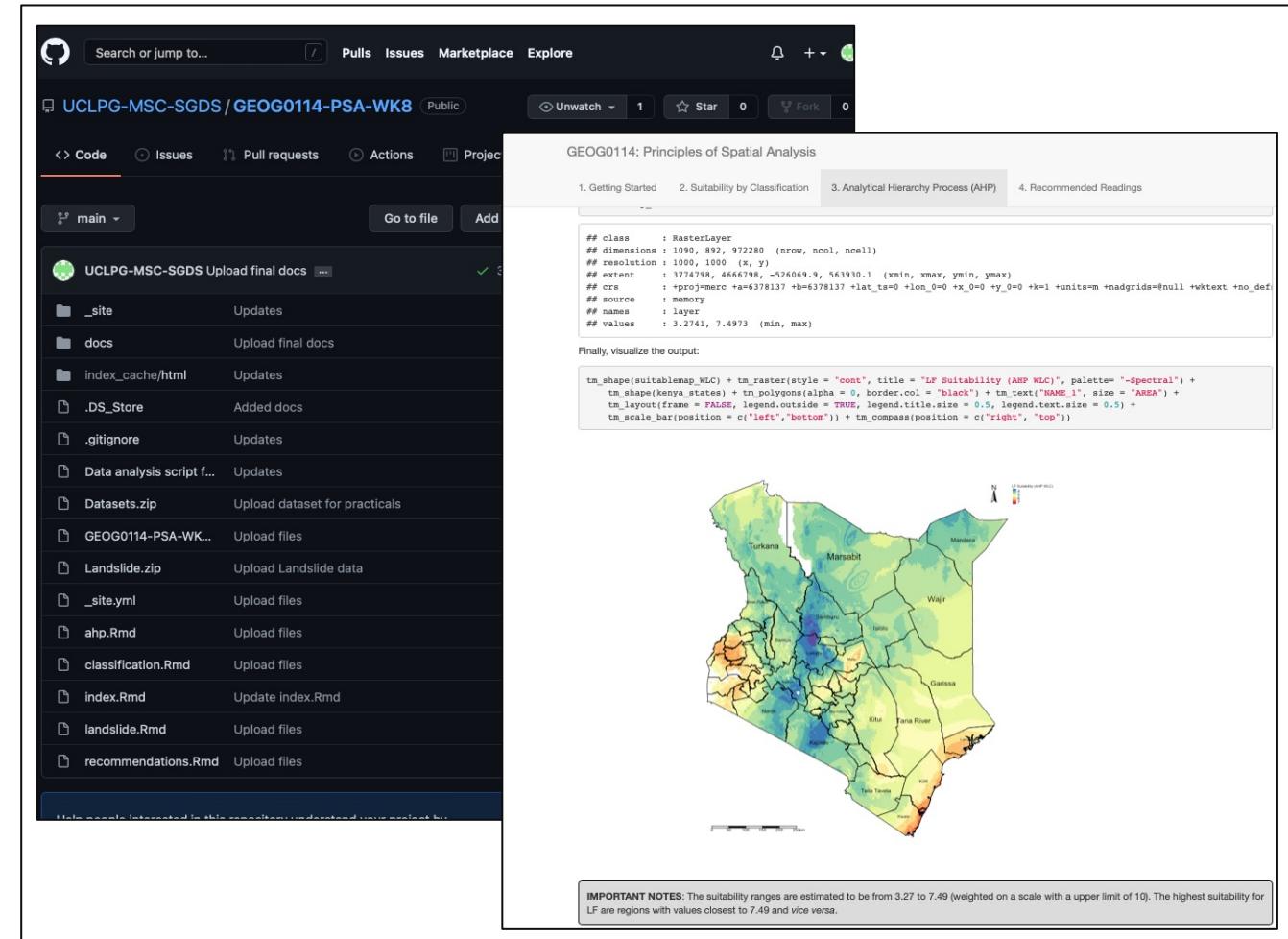
## 1. Efficiency

- Automated tasks and data managing
- Can recycle & reuse code scripts for new projects

## 2. Fosters good scientific practice

- Transparency and replication (AKA reproducible research)
- Creates log so anyone can follow in your footstep (i.e., github, gitlab etc.,)

You can literally pull-off some really creative stuff like generating websites, accessing tools via APIs etc.



**Example: Working in RStudio and synchronising it with GitHub to not only use as a cloud back-up, but to generate a website through RStudio and GitHub for teaching MSc Students.**

# What is Vector Spatial Data & its Features?

Suppose we want to map the following from this landscape:

**1. Physical objects:**

- Location of buildings
- Farm plots
- Locations of trees
- Road network
- Block areas (divided by the road)

**2. Levels of soil moisture across the landscape**



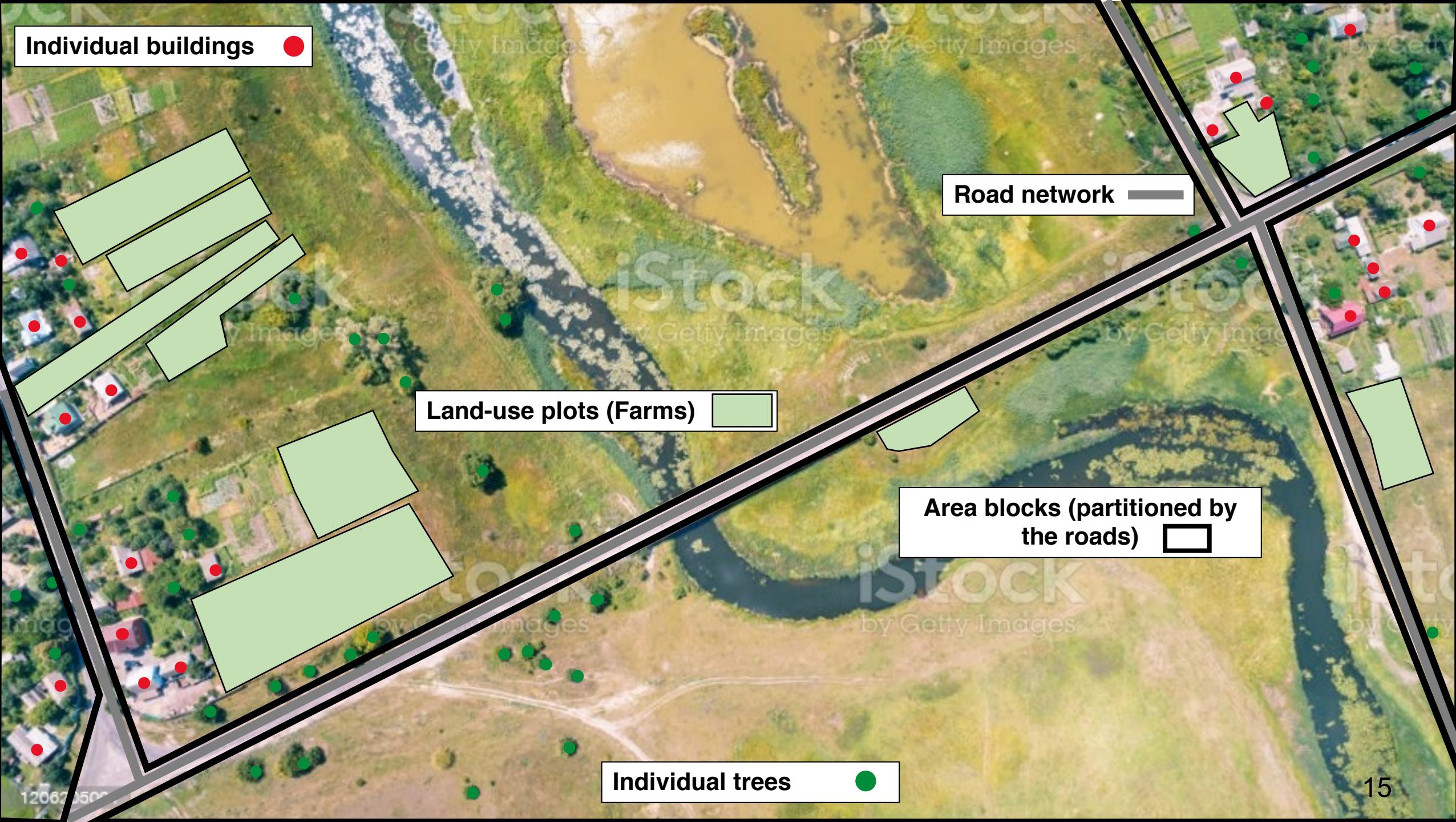
Individual buildings

Road network

Land-use plots (Farm)

Area blocks (partitioned by  
the roads)

Individual trees



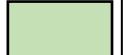
Individual buildings



Road network



Land-use plots (Farms)

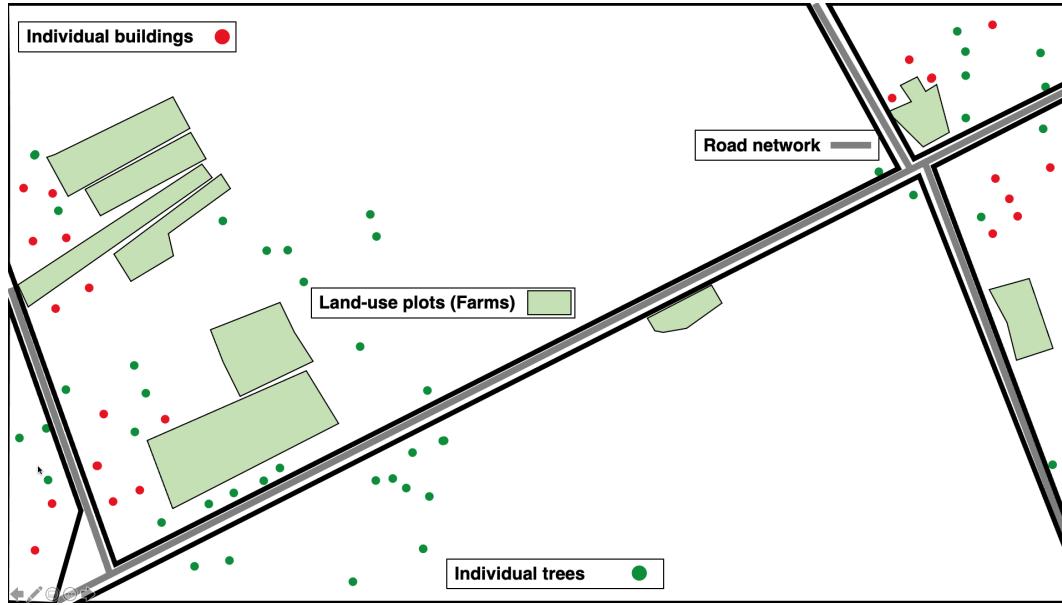


Area blocks (partitioned by  
the roads)



Individual trees



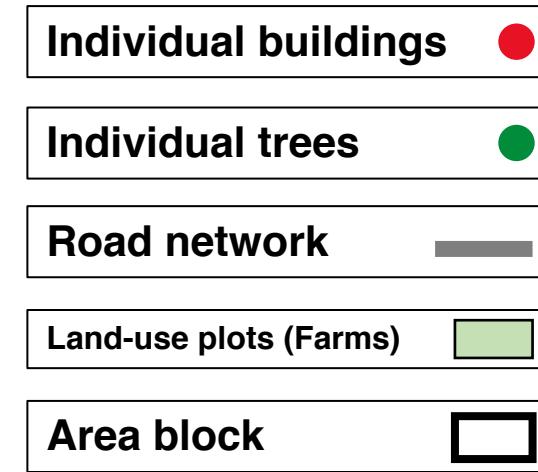
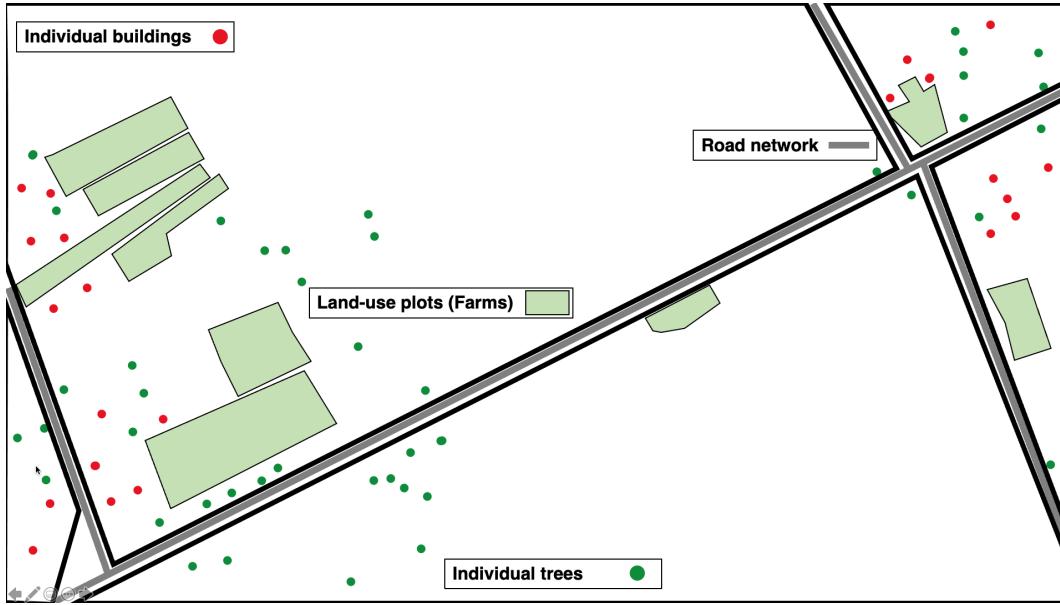


<b>Individual buildings</b>	●
<b>Individual trees</b>	●
<b>Road network</b>	—
<b>Land-use plots (Farms)</b>	■
<b>Area block</b>	□

The above objects listed are called “**Features**”. A feature can be described according to its characteristics which is termed an “**Attribute**” in GIS. The attribute of a feature can be a **numeric** or **text** observation.

**For example:**

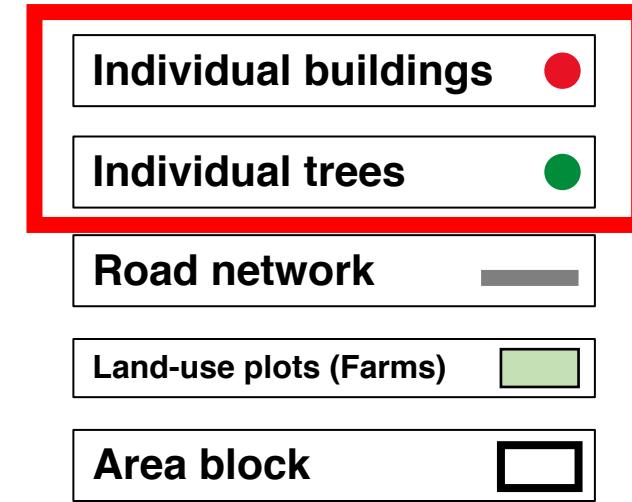
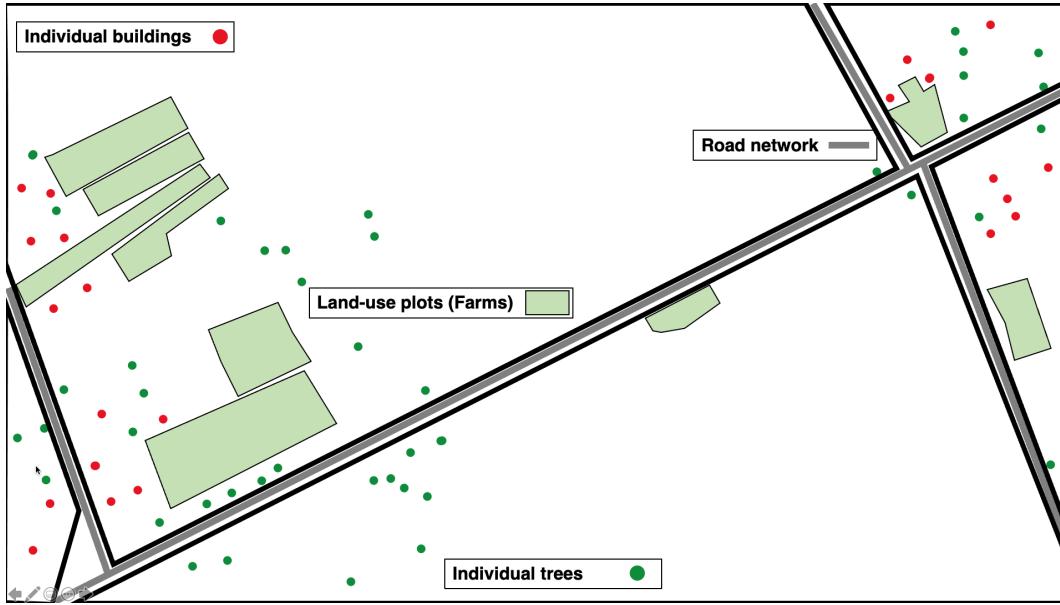
- A building is a **point feature** on this map, the number of people living a building is a **numeric attribute** describing this **feature**. Type of building (i.e., Victorian or modern) is a **text attribute**
- The road network is a **polyline feature**, the length (or distance (m)) of the road is a **numeric attribute** describing the road
- Land-use plot is a **polygon (or area) feature**, the type of land-use (i.e., farming) is the **text attribute** describing what that polygon is etc.



Now, the above objects are typically **Discrete** features, and therefore classed a **Vector Data**

There are three main types of **Vector Data**:

1. **Point vector**
2. **Polyline or Line vector**
3. **Polygon vector**



Now, the above objects are typically **Discrete** features, and therefore classed a **Vector Data**

There are three main types of **Vector Data**:

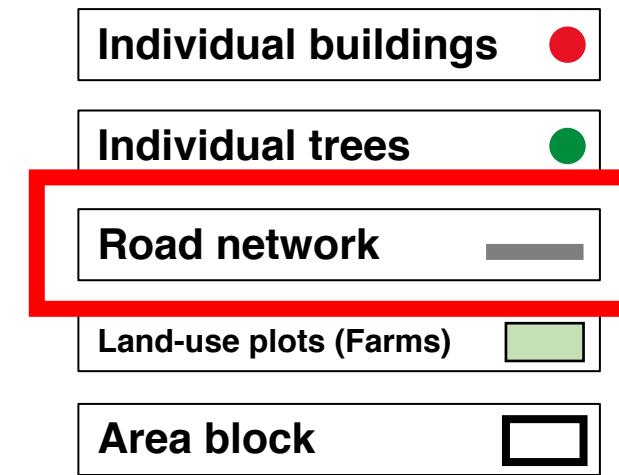
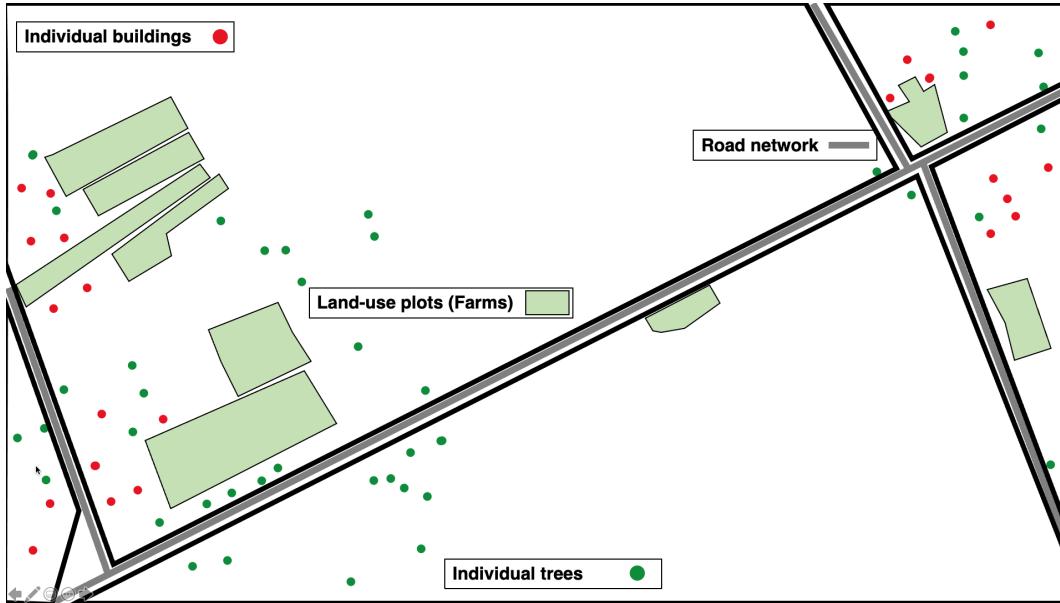
### 1. Point vector

### 2. Polylines or Line vector

### 3. Polygon vector

#### Characteristics of a point vector

- X, Y location characterize as a coordinate
- Has no area
- Has no length
- This applies to discrete features of sample points
- Its geometry consists of a single **node** or **vertex**



Now, the above objects are typically **Discrete** features, and therefore classed a **Vector Data**

There are three main types of **Vector Data**:

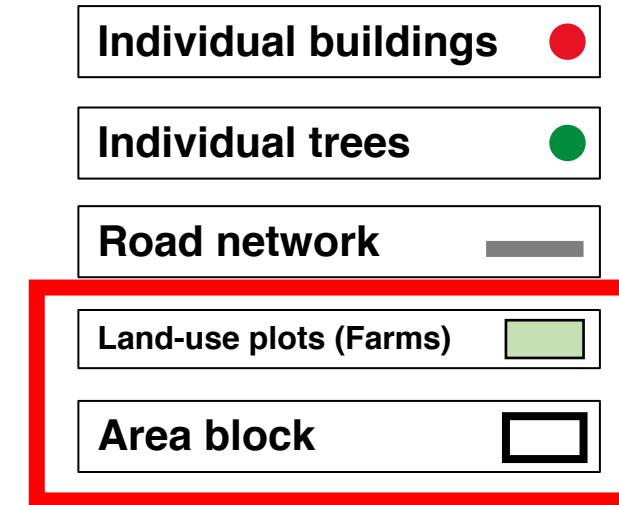
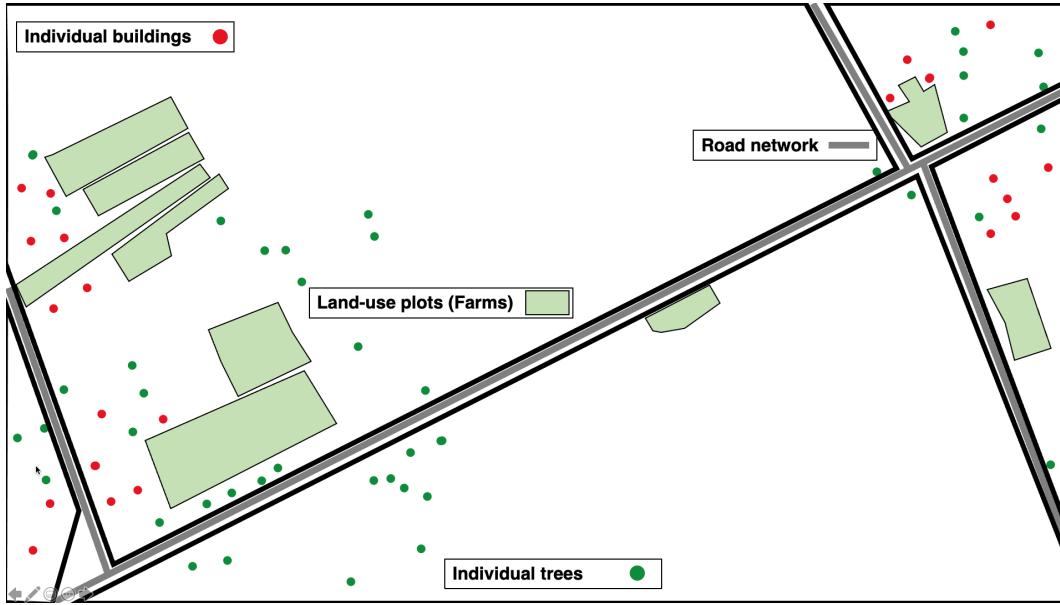
**1. Point vector**

**2. Polylines or Line vector**

**3. Polygon vector**

#### Characteristics of a polyline or line vector

- These are a series of X, Y points characterized by coordinates to form a line
- Has no area
- They have a length
- They have a direction (important for visualizing rivers, streams & roads)
- Connectivity (it connects to other line segments in the network)
- This applies to features without an area but with a length – roads, rivers, railway tracks or migration flows between two or more locations.
- Its geometry consists of **2 nodes** (i.e., beginning and end of point of line) & can have more than one **vertex** (i.e., point(s) that connect different lines together)
- It is never enclosed



Now, the above objects are typically **Discrete** features, and therefore classed a **Vector Data**

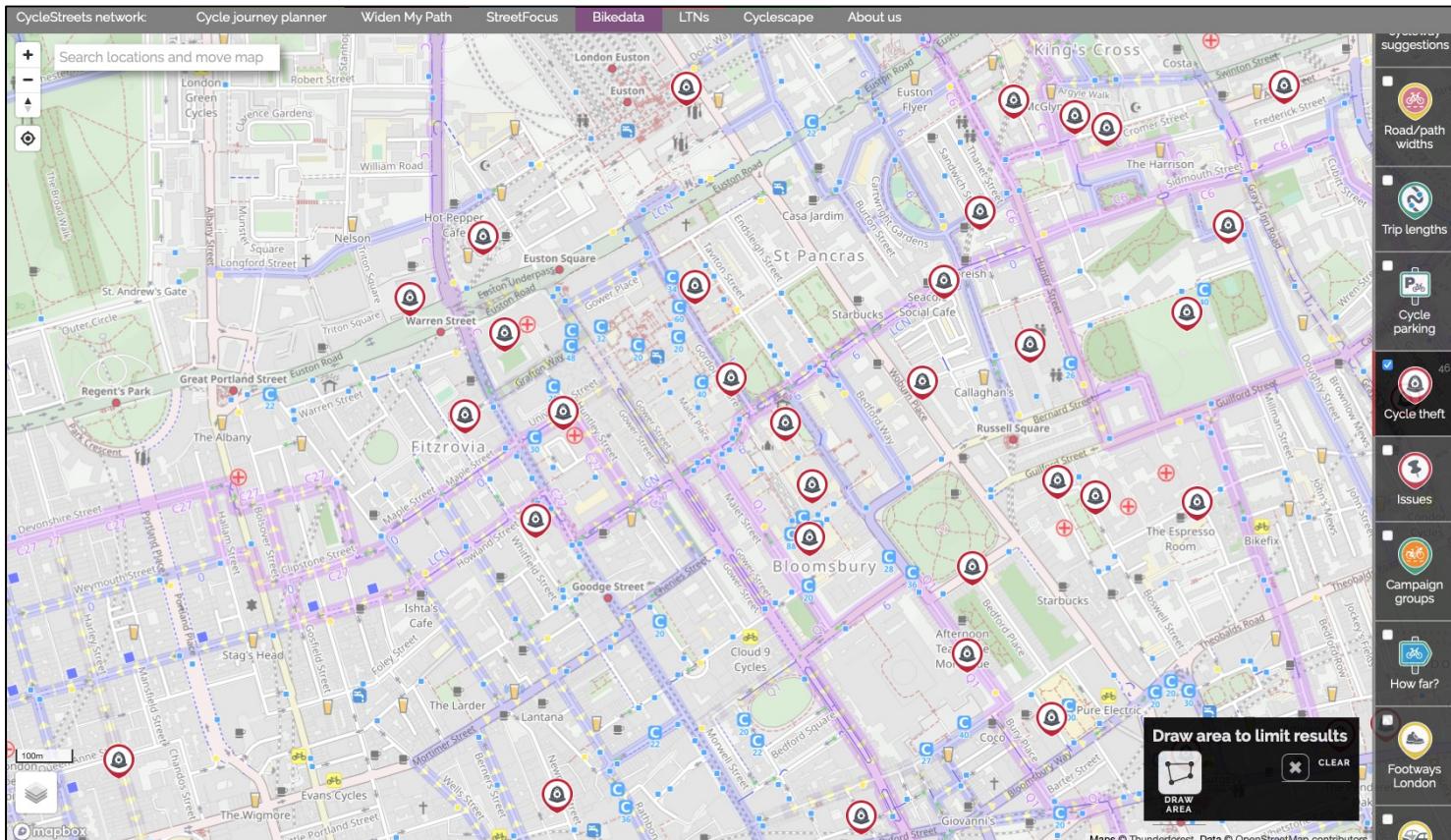
There are three main types of **Vector Data**:

1. **Point vector**
2. **Polylines or Line vector**
3. **Polygon vector**

#### **Characteristics of a polygon vector**

- These are a series of X, Y points characterize by coordinates to form an enclosed region
- It has an area
- It has no length but rather a perimeter instead
- This applies to features with enclosed regions – e.g., postcode areas, area of residential premise, counties (other administrative boundaries) etc.
- Polygons have **three vertices** or more each connecting sequentially where the first vertex connects with the last vertex.

## Point Pattern Data (PPD)



Represents point locations of bicycle thefts in Central London area

Source: BikeData.CycleStreets network <https://bikedata.cyclestreets.net/>

## Key Characteristics

The main interest is the occurrences of an event at a points (or points). These events occur at “random” at any given geographic space and time.

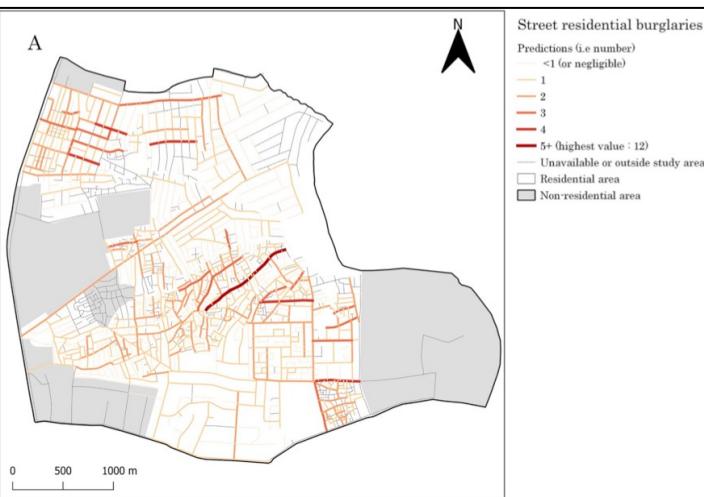
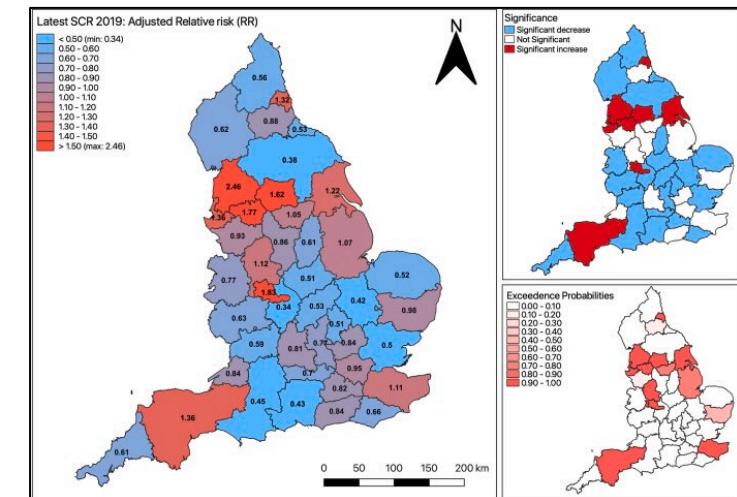
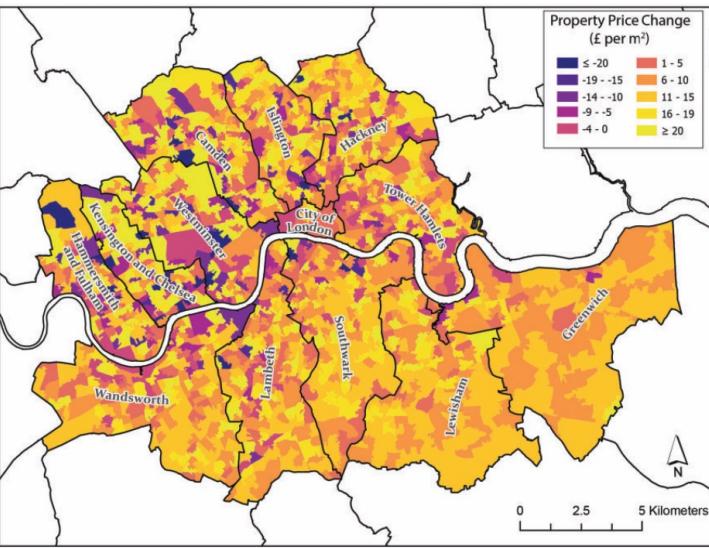
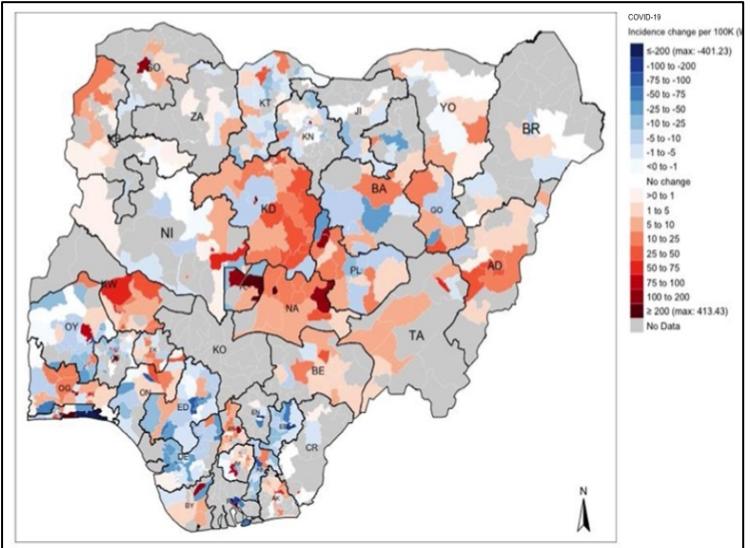
Examples of PPDs events (or outcomes):

- Point locations of burglaries
- Riots
- Locations of car collisions etc.,
- Locations of where an adult tree needs to be replanted

Some PPDs events may carry additional information that may describe the occurrence of an observed event (or outcome):

- Burglary: Type of premise that was burgled, time of day the burglary occurred etc.,
- Car collision: type of road, weather condition etc.

## Aggregated Data



## Key Characteristics

The main interest the quantity of interest defined for line segments, areas, or regions.

Events (or outcomes) that are aggregated measures to areas:

- Prevalence of a disease in areas
- Population density in a county
- Regional unemployment rates
- Risk of an outcome

### Sources:

- Musah, A., et al. (2020): <https://doi.org/10.1016/j.apgeog.2019.102126>  
 Todd, J., et al. (2021): <https://doi.org/10.1177/23998083211001836>  
 Li, L., et al. (2022): <https://doi.org/10.1016/j.apgeog.2022.102718>  
 Elimian, K., et al. (2022): <http://dx.doi.org/10.1136/bmjopen-2022-063703>

## Non-spatial context & data structure:

	ATTRIBUTE			
	Variable 1	Variable 2	...	Variable n
Entity 1	$attribute_{11}$	$attribute_{12}$	...	$attribute_{1n}$
Entity 2	$attribute_{21}$	$attribute_{22}$	...	$attribute_{2n}$
:	:	:	..	:
Entity m	$attribute_{m1}$	$attribute_{m2}$	...	$attribute_{mn}$

To apply some spatial analysis to data – you must have some variable that defines the entity's geographic location. This can be **GPS coordinates, spatially referenced geometries** of an area

NOTES: It is not enough to have just the name of the area(s). It must be some geometric entry!

- Attributes that defines an entity's location are typically excluded from the analysis
- The conventional statistical methods, that assumes independence, are used for analyzing such dataset
- Results churned from this dataset are completely independent from “**spatial arrangement**” of the entities.

# Spatial context & data structure [1]:

	Geographical Coordinate		ATTRIBUTE			
	X	Y	Variable 1	Variable 2	...	Variable n
Entity 1	$X_1$	$Y_1$	$attribute_{11}$	$attribute_{12}$	...	$attribute_{1n}$
Entity 2	$X_2$	$Y_2$	$attribute_{21}$	$attribute_{22}$	...	$attribute_{2n}$
:	:	:	:	:	..	:
Entity m	$X_m$	$Y_m$	$attribute_{m1}$	$attribute_{m2}$	...	$attribute_{mn}$

In this example, what defines the entity's geographic location are **X, Y GPS coordinates**.

Definition of an entity's location are not limited to coordinates, you can have spatially reference areas with their associated boundaries with geometries.

This instance illustrates an example of geostatistical data.

- Attributes that defines an entity's location are typically explicitly incorporated into analysis
- Spatial statistical methods, that assumes dependence, are used for analyzing such geographically referenced dataset
- Results churned from this dataset are completely dependent from “**spatial arrangement**” of the entities.

## Spatial context & data structure [2]:

**Flood data**

	A	B	C	D	E	F	G	H
1	NAME_2							
2	Asunafo North	61670000						
3	Asunafo South	213260000						
4	Asutifi North	161440000						
5	Asutifi South	71170000						
6	Tano North	63980000						
7	Tano South	20640000						
8	Adansi Akrofoum	217430000						
9	Adansi Asokwa	18610000						
10	Adansi North	41710000						
11	Adansi South	351740000						
12	Afigya-Kwabre North	5420000						
13	Afigya-Kwabre South	69180000						
14	Ahafo-Ano North	37420000						
15	Ahafo-Ano South East	24810000						
16	Ahafo-Ano South West	6690000						
17	Amansie Central	363880000						
18	Amansie South	133670000						
19	Amansie West	143890000						
20	Asante-Akim Central	53720000						
21	Asante-Akim North	246520000						
22	Asante-Akim South	51220000						
23	Asokore-Mampong	28120000						
24	Asokwa	31330000						
25	Atwima-Kwanwoma	135380000						
26	Atwima-Mponua	519430000						
27	Atwima-Nwabiagya North	38350000						
28	Atwima-Nwabiagya South	114530000						
29	Bekwai	115700000						
30	Bosome Freho	42320000						
31	Bosomtwe	147590000						
32	Ejisu	113670000						
33	Ejura-Sekyedumase	41350000						
34	Juaben	8340000						
35	Kumasi	25360000						
36	Kwabre East	69630000						

	A	B	C	D	E	F	G	H
1	NAME_2							
2	Asunafo North	1417524579						
3	Asunafo South	27404248.1						
4	Asutifi North	940135547.9						
5	Asutifi South	599759691.4						
6	Tano North	41071135.4						
7	Tano South	491213456.4						
8	Adansi Akrofoum	70859238.8						
9	Adansi Asokwa	98651435.7						
10	Adansi North	259045178.4						
11	Adansi South	63690339.9						
12	Afigya-Kwabre North	187259028.7						
13	Afigya-Kwabre South	224011317.5						
14	Ahafo-Ano North	96239737.9						
15	Ahafo-Ano South East	689293224.9						
16	Ahafo-Ano South West	106804879.4						
17	Amansie Central	53540387.7						
18	Amansie South	595698321.6						
19	Amansie West	40866658.5						
20	Asante-Akim Central	501678682.3						
21	Asante-Akim North	1131089538						
22	Asante-Akim South	158879729						
23	Asokore-Mampong	24027505.73						
24	Asokwa	5826532.77						
25	Atwima-Kwanwoma	52682133.8						
26	Atwima-Mponua	1891514638						
27	Atwima-Nwabiagya North	13895929.1						
28	Atwima-Nwabiagya South	567959143.2						
29	Bekwai	537771810.5						
30	Bosome Freho	571471566						
31	Bosomtwe	424465104						
32	Ejisu	13252411.3						
33	Ejura-Sekyedumase	346444084						
34	Juaben	171971426.3						
35	Kumasi	1720232.91						
36	Kwabre East	123537780.8						

The join is performed on a common column

To calculate the proportion of land surface that are flood prone, we will need to join the Flood data with the total surface area data by “NAME\_2” column using the merge() function

# Workflow of the online handbook

## Format and learning flow [1]

To generate `bmi` into our data frame, we would need to access the `height` (m) and `weight` (kg) columns using the `$` from the data frame its stored to, and apply the above formula as a code to generate the new `bmi` column:

```
# Create 'bmi' in the data frame i.e., 'dataset' and calculate 'bmi'  
# using the $weight and $height  
dataset$bmi <- dataset$weight/((dataset$height)^2)  
# View the data frame 'dataset' and you will see the new bmi variable inside  
View(dataset)
```

You can overwrite the `height` (m) column to change its units into centimeters by multiplying it to 100; equally, the `weight` (kg) column can be overwritten and converted from units of kilograms to grams by multiplying it to 1000.

```
# using $height and *100  
dataset$height <- dataset$height*100  
# using $weight and *100  
dataset$weight <- dataset$weight*1000  
# use View() the data frame 'dataset' and you will see the updated variables  
View(dataset)
```

The screenshot shows a video player interface with a slide titled "What is R/RStudio". The slide contains the R Studio logo, a photograph of a computer monitor displaying RStudio's interface with various panels like the code editor and data viewer, and three hyperlinks: "The r-project for statistical computing: <https://www.r-project.org>", "Open-source RStudio: <https://www.rstudio.com>", and "The R Journal: <https://journal.r-project.org>". A small video thumbnail in the bottom right corner shows a person speaking. The video player has a progress bar at the bottom left indicating it is at 0:26 / 9:16.

There will always be a video to explain the necessary theory of the statistical method, as well as an explanation of the code

## Format and learning flow [2]

To generate `bmi` into our data frame, we would need to access the `height` (m) and `weight` (kg) columns using the `$` from the data frame its stored to, and apply the above formula as a code to generate the new `bmi` column:

```
# Create 'bmi' in the data frame i.e., 'dataset' and calculate 'bmi'  
# using the $weight and $height  
dataset$bmi <- dataset$weight/((dataset$height)^2)  
# View the data frame 'dataset' and you will see the new bmi variable inside  
View(dataset)
```

You can overwrite the `height` (m) column to change its units into centimeters by multiplying it to 100; equally, the `weight` (kg) column can be overwritten and converted from units of kilograms to grams by multiplying it to 1000.

```
# using $height and *100  
dataset$height <- dataset$height*100  
# using $weight and *100  
dataset$weight <- dataset$weight*1000  
# use View() the data frame 'dataset' and you will see the updated variables  
View(dataset)
```

There will always be a text to explain the necessary code and what the code is also doing, and the steps for coding – so here, you will have to read through the text.

## Format and learning flow [3]

To generate `bmi` into our data frame, we would need to access the `height` (m) and `weight` (kg) columns using the `$` from the data frame its stored to, and apply the above formula as a code to generate the new `bmi` column:

```
# Create 'bmi' in the data frame i.e., 'dataset' and calculate 'bmi'  
# using the $weight and $height  
dataset$bmi <- dataset$weight/((dataset$height)^2) # Comment  
# View the data frame 'dataset' and you will see the new bmi variable inside  
View(dataset)
```

Actual code

You can overwrite the `height` (m) column to change its units into centimeters by multiplying it to 100; equally, the `weight` (kg) column can be overwritten and converted from units of kilograms to grams by multiplying it to 1000.

```
# using $height and *100  
dataset$height <- dataset$height*100  
# using $weight and *100  
dataset$weight <- dataset$weight*1000  
# use View() the data frame 'dataset' and you will see the updated variables  
View(dataset)
```

There will always be code chunks for you to replicate and study in your own RStudio.

The code chunks will be annotated with a `#` (hash tag) with a comment to give context. So, this is not actual code

The real code is the code chuck with no `#` (hash tag) at the beginning.

Let's begin the walkthrough

Go to this website:

<https://uclpg-msc-sgds.github.io/UCL-UoG-Workshop/>