Assignment 2 – Scientific Programming for Geospatial Sciences

Raian V. Maretto, Mahdi Farnaghi, Rosa Aguilar Bolivar

12th January 2026

Group size: 2 students

Workload: 14 hours per student (≈28 hours per pair)

Deliverables:

1. **Source code** (repository link or zip file)
2. **Technical report** (max 4 pages)

Deadlines:

- Pairs to be submitted by Wednesday, **14th of January, by 17hs**.
- Proposed geospatial problem to be solved to be presented at the feedback session on the **20th of January from 13:45 to 14:30**. Please note that no slide is needed for this, you just need to describe it quickly orally.
- Final project to be submitted **1st of February, 23:59**

1. Description

During the second part of the course, the more advanced and essential concepts of scientific programming for geospatial sciences have been addressed. During the lectures, you were exposed to several different concepts; during the exercises, you learned to apply the gained knowledge. Modern geospatial analysis increasingly relies on integrating raster and vector data, handling large multidimensional datasets, and applying tensor-based computation for scalable analysis and machine learning.

In this assignment, you will design and implement a reproducible geospatial analysis pipeline that **integrates raster and vector data to solve a real-world geospatial problem**. You will work, in pair, with Earth Observation (satellite, drones, airborne sensors, etc.) raster data, vector features, and spatio-temporal data cubes, and demonstrate scientific programming best practices.

Each pair must choose a real-world geospatial problem**,** acquire or generate relevant datasets, implement algorithms, build a small geodata service, and produce visualizations and documentation.

2. Real-World Problem (Choose **one**)

Each pair must choose **one** of the following application scenarios:

Option A – Urban Heat Island Analysis

Analyze how land cover and urban morphology influence land surface temperature.

- Raster data: Land Surface Temperature (e.g. MODIS, ECOSTRESS)
- Vector data: Urban boundaries, land use polygons
- Output: Zonal statistics, spatio-temporal trends, and hotspot identification

Example Datasets

- MODIS Land Surface Temperature (LST)
    - Single-band raster
    - Good for normalization, thresholding, statistics

Sentinel-2 NDVI (single date)

- One band or computed index

Option B – Flood Exposure Assessment

Estimate population or infrastructure exposure to flooding.

- Raster data: Flood extent, satellite images, or elevation-derived flood probability
- Vector data: Buildings, roads, water bodies, or administrative boundaries
- Output: Exposure metrics per administrative unit

Option C – Deforestation Risk Mapping

Assess deforestation dynamics and proximity to infrastructure.

- Raster data: Forest cover / forest loss time series
- Vector data: Roads, protected areas
- Output: Risk indicators and temporal summaries

Option D – Environmental Suitability Mapping

Combine environmental rasters and vector constraints.

- Raster data: Climate or vegetation indices
- Vector data: Species observations or land management zones
- Output: Suitability map (e.g. for windmills, solar panels, bioenergy, forest services, ect.) and zonal summaries

You may propose your own problem **with instructor approval**, provided it meaningfully integrates raster and vector data.

Examples of data sources:

- **For Satellite Images:**
  - Copernicus Data Space Ecosystem: https://dataspace.copernicus.eu/
  - USGS Earth Explorer: https://earthexplorer.usgs.gov/
  - Brazil Data Cube: https://data.inpe.br/bdc/en/home-page-2/
  - Euro Data Cube: https://eurodatacube.com/
- **For Vectorial data and maps:**
  - OpenStreetMaps: https://www.openstreetmap.org
    - Regional downloads (Geofabrik): https://download.geofabrik.de
    - Overpass (REST/WFS) API: https://overpass-api.de
    - Python tools: https://osmnx.readthedocs.io
  - FAO GeoNetwork (Global and regional datasets on Land Use, Agriculture, Soils, Protected areas - REST API available): https://www.fao.org/geonetwork
  - Natural Earth (Global cartographic dataset on Administrative boundaries, rivers, coastlines, populated places): https://www.naturalearthdata.com
  - PDOK (Dutch public services on mapping): https://www.pdok.nl/
  - GeoDa (Spatial Data Science library which contains some example datasets): https://geodacenter.github.io/
  - Global Human Settlement Layer (GHSL): https://human-settlement.emergency.copernicus.eu/datasets.php#urban

3. Required Technical Components

Your solution **must explicitly demonstrate proficiency in** the following topics.

3.1. NumPy Arrays

- Raster data loaded as NumPy arrays
- Element-wise operations (e.g. masking, normalization, band algebra)
- Efficient array manipulation (no Python loops where avoidable)

Please note that, if you do that in XArray, it is also fine, then you don't need to do it also in NumPy.

Examples:

- Thresholding raster values
- Computing indices (e.g. NDVI-like metrics)
- Applying vector-derived masks to rasters

3.2. Tensors with TensorFlow and PyTorch

You must also experiment with Tensors. In this case, perform the operations below, or repeat the operations performed with arrays using now Tensors, and compare the performance.

Required:

- Convert raster data to tensors
- Perform tensor-based operations (e.g. convolution, aggregation, or regression)
- Demonstrate GPU awareness (even if run on CPU)

Examples:

- PyTorch tensor for raster classification
- TensorFlow tensor for spatial filtering or regression
- Comparison of NumPy vs tensor-based performance (brief)

3.3. Vector Processing (GeoPandas, GDAL/OGR, Shapely)

- Read/write vector data using Fiona
- Geometry operations using Shapely
- Attribute and spatial operations using GeoPandas
- You are required to perform **at least 3 geospatial operations.**

3.4. Raster and Vector Data Cubes (Xarray)

- Load raster data as `Dataset`
- Handle at least **one additional dimension** (time, band, or scenario)

Examples:

- Time-series analysis per polygon
- Band-wise statistics
- Spatio-temporal slicing

3.5. Raster–Vector Integration (Core Requirement)

This is the **core of the assignment**.

You must demonstrate:

- Raster sampling within vector geometries
- Zonal statistics using vector features
- Bidirectional interaction (vector → raster and raster → vector)

4. Deliverables

4.1. Source Code Repository (can be submitted with a zip file)

Must contain:

- Full Python source code
- README.md with installation and execution instructions
- Tests
- Dataset references or scripts for downloading them
- API documentation (OpenAPI docs count)

4.2. Technical Report (max 4 pages)

Please note that the report is **mandatory**, and without it the grade will be **zero**. The report should follow the following structure:

1. **Problem statement**
2. **Datasets** (source, description, spatial extent)
    - Raster sources
    - Vector sources
    - Spatial and temporal resolution
3. **Methodology**
    - Describe your choices and methods used.
    - math formulations
    - algorithms
    - geospatial algorithms and how did you apply the algorithms learnt in the geospatial context
    - NumPy array operations
    - Tensor operations (TensorFlow or PyTorch)
    - Raster/vector cube representation (Xarray/Xvec)
    - Raster–vector integration approach
4. **Results**
    - Maps, charts, etc
5. **Discussion**
    - Discuss on the problem, how did you solve it and why that solution was appropriate. Limitations, computational considerations.
6. **Conclusion (lessons learned)**
    - Main conclusions on the work, and reflections on what did you learn
7. **Workload distribution (Individual contributions)**
    - Describe here how each student contributed to the work. Who developed what
8. **References**

5. Rubric

Item	Focus Area	What Is Assessed	Points	Performance levels
1	**Raster–Vector Integration & Correctness**	Correct and meaningful integration of raster and vector data (e.g. zonal statistics, raster masking, raster sampling at vector locations, temporal aggregation). Clear and explicit implementation, not a black box.	**25**	Excellent: 26–30 Good: 20–25 Satisfactory: 10–19 Poor: 0–9
2	**Arrays, Tensors & Data Cubes**	Explicit and correct use of: * NumPy arrays (indexing, masking, aggregation) * TensorFlow or PyTorch tensors (creation, manipulation, NumPy ↔ tensor conversion) * Raster/vector data cubes using Xarray and Xvec, including alignment and aggregation across dimensions	**25**	Excellent: 26–30 Good: 20–25 Satisfactory: 10–19 Poor: 0–9
3	**Raster & Vector Processing**	Correct use of GeoPandas, Fiona, and Shapely for vector processing; correct raster handling; proper CRS management; appropriate spatial operations	**15**	Excellent: 13–15 Good: 9–12 Satisfactory: 5–8 Poor: 0–4
4	**Code Quality & Reproducibility**	Clean and modular code structure; separation of data loading, processing, analysis, and visualization; reproducible execution with clear README and environment specification	**15**	Excellent: 13–15 Good: 9–12 Satisfactory: 5–8 Poor: 0–4
5	**Report Quality & Scientific Explanation**	Clear problem definition; data description; explanation of methods (arrays, tensors, cubes, integration); correct interpretation of results; clarity and structure of writing	**20**	Excellent: 9–10 Good: 6–8 Satisfactory: 3–5 Poor: 0–2
		Total	**100**	