The many ways of GIS for digital humanities

Summer School on Digital Humanities

Web site: https://bit.ly/dt4h-gis

Augusto Ciuffoletti

9 giugno 2025

Plan of the Tutorial

- The tutorial is divided into seven sessions.
- The first session is lecture-style, introducing basic concepts, terminology, and background.
- Each of the following six sessions focuses on a specific tool or concept and includes hands-on exercises:
  - QGIS, OpenStreetMap, UMap, GaiaGPS, Georeferencing with QGIS, and Leaflet.
- Only minimal prior experience is expected, so everyone can participate.
- All learning materials are available online, so you can revisit and try exercises at home if needed.
- You can keep the slides on your phone and practice on your PC

What is digital cartography (aka GIS)

- Digital cartography shares fundamental principles with classical cartography:
  - It records the geographical position of objects or reference points
  - It represents the morphological features of the landscape
  - It maps travel routes and pathways
  - It associates specific attributes and characteristics with mapped objects
  - It can depict imaginary landscapes or reconstruct past and future territorial scenarios

Why do we use digital cartography?

- Digital and conventional cartography share similar purposes
- Both serve as essential tools for:
  - Measuring geometric dimensions of objects and areas
  - Defining and recording state and property boundaries
  - Planning and navigating routes to specific destinations
  - Documenting journeys and various forms of travel
  - Geographically situating human or natural events to analyze relationships
  - Depicting and teaching about distant or inaccessible places
- These applications can relate to the present, as well as to past or future scenarios

The advantages of digital cartography

- Digital and traditional cartography differ primarily in the medium used to store maps
  - Digital maps are recorded on various types of digital media and accessed via suitable devices
- This distinction brings several key advantages:
  - Easy sharing due to the dematerialization of maps
  - Automatic acquisition of positions and movements
  - Ability to merge data from multiple maps
  - Integration of multimedia information
  - Simplified creation and reuse of maps

Cartography and public history

- History and cartography are deeply interconnected
  - History records events in relation to places
- The way we represent the world reflects our perspectives and values
- Was a medieval geographer creating maps for his king a public historian?
  - Engaging the public with the past
  - Applying history to practical use
  - Encouraging critical reflection
- Can the T-and-O map be considered a public history document?
- Will today's maps become public history documents in the future?
- Who has the capability to create such historical records?
  - Digital cartography opens new perspectives on this sense
  - The answer depends on accessibility and widespread use

Diffusion of digital cartography

- Digital cartography relies on:
  - Powerful graphics processors
  - High-definition displays
- In the Pentium era, these were largely inaccessible to PCs
  - ...limiting the advantages previously mentioned
- Digital cartography became widely affordable around 2005
- Today, nearly everyone carries a pocket-sized GIS engine
- Despite advancements, multiple representation standards still exist (standardization is ongoing)
- Cartography is now technically accessible to anyone
- Current challenges:
  - Simplifying access to cartographic tools
  - Harmonizing representation to enable data integration
- Future directions:
  - Developing autonomous devices to continuously record environmental features
  - Enhancing the communication of historical narratives

Web Mapping

- The Web is a powerful medium for sharing resources
- Web mapping technology emerged a few years after the creation of the WWW in 1989
- The evolution of the Web paralleled the advancement of Web mapping
- In the early ’90s, maps were primarily static, offering limited interaction or layering
- By the late ’90s, users gained the ability to manipulate maps and create new ones
  - ...with computationally intensive tasks handled on the server side
- Between 2000 and 2005, advancements in Web technologies facilitated the rise of Web mapping services
  - ...enabling seamless integration with other services via standardized interfaces
  - ...making the definition of standard representations and protocols increasingly important

Web Mapping in Web 2.0

- More powerful personal computing devices enable real-time interaction with Web mapping servers
  - ...allowing maps to be generated as mashups from multiple databases
- The advent of Web 2.0 (2005) introduces crowd-sourced geospatial data
- Increased computing power enables client-side manipulation of map features
  - ...with cloud storage and servers facilitating authentication and data sharing

Access: open vs closed digital cartography

- A fundamental choice in online content:
  - Data can be publicly accessible or restricted to private use
- The same distinction applies to digital cartography

Examples:

- **Open-source cartography**: OpenStreetMap
  - Maps are freely available in the public domain
  - Anyone can contribute by adding features
  - Maps can be reused without restrictions
- **Freely accessible but proprietary cartography**: Google Maps
  - Access is provided through a private service
  - Users can create and overlay their own maps
- **Commercial/private cartography**: Mapbox
  - Maps are provided as a paid service
  - Costs scale with usage (e.g., number of views)

Fundamental Core Concepts

- Concepts that **simplify access** to geographic data
- **Coordinates**: Latitude and Longitude
- **Geographic Features**:
  - Point – Defined by a single coordinate pair
  - Segment – A straight line connecting two points
  - Line – A sequence of connected segments
  - Area – A closed shape formed by a continuous line
- **Data Models**:
  - Vector Model – A collection of features with attributes
  - Raster Model – A grid of cells storing attributes
    - Often derived from graphic formats like JPEG
- **Additional Core Elements**:
  - Attributes – Data linked to features and cells
  - Layers – Organized sets of maps for structured visualization
- A suite of tools supports the manipulation and visualization of these concepts

Geographic Coordinate Systems

- **Harmonizing representation** requires the existence of standards for data models
- A Geographic Coordinate System (GCS) defines how a point is represented on the Earth's surface
- A standard GCS plays a crucial role in sharing meaningful information about positions, paths, and distances
- The standard evolves over time to accommodate changing needs and advances in technology
  - Originally, latitude was computed based on the maximum duration of daylight

World Geodetic System of 1984

- A widely adopted Geographic Coordinate System (GCS) today is `WGS84` (World Geodetic System 1984)
- The label `EPSG:4326` refers to its "non-projected" version
  - For example, `EPSG:3856` represents its Pseudo-Mercator projection on a square surface
- `WGS84 EPSG4326` is used by the Global Positioning System (GPS) and for data storage formats such as GeoJSON
- `WGS84 EPSG3856` is used by Google Maps and computer visualization tools
- Key features of `WGS84 EPSG4326`:
  - Coordinates are expressed in latitude (north) and longitude (east) (in this order)
  - Coordinates are expressed in degrees (decimal format)

Storing a digital map

- A digital map usually includes:
  - raster tiles as a visual background
  - a collection of vector features
- **Raster tiles** are available from various providers like OpenStreetMap (free) or Mapbox (paid)
- Tiles are accessed by specifying the zoom level and the tile's position in a grid
  - e.g.: `http://tile.openstreetmap.org/<zoom>/<x>/<y>.png`
  - try `https://tile.openstreetmap.org/7/67/46.png`
- **Vector features** are stored in a database, with tools for searching and updating similar to those in conventional databases
- As with traditional databases, you can choose between relational and non-relational models

PostGIS: a relational GIS database

- A sample query that creates a new feature:

```
INSERT INTO places (name, coord)
    VALUES ('Pisa', ST_GeographyFromText('SRID=4326;POINT(10.41 43.72)'));
```

- Legend:
  - places is a table created beforehand
  - It contains two columns: one for the name of a place and one for its coordinates
  - The INSERT command adds a new row to the table
  - The new point is named *Pisa*
  - Coordinates are provided using the ST_GeographyFromText function from PostGIS
  - The input string includes an SRID to define the coordinate system
  - `4326` refers to the `WGS 84` standard (EPSG:4326)
  - Coordinates follow the format: longitude first, then latitude — note the order is reversed from `WGS 84` standard

GeoJSON: maps as JavaScript objects

- GeoJSON is a GIS extension of the JSON object description language
- A `map_layer` variable hosting a collection of features is initialized as

```
map_layer = { "type": "FeatureCollection",
              "features": [] }
```

- A new point feature is defined with

```
new_feature = { "type": "Feature",
                "properties": {"name":"Pisa"},
                "geometry": {"type": "Point", "coordinates": [ 10.41, 43.72 ] } }
```

- And the JavaScript statement to insert the new feature in the empty collection is:

```
map_layer["features"].append(new_feature)
```

GeoJSON and noSQL databases

- The previous example refers to variables in the scope of a Javascript program
- Using a noSQL database service, the service provides an API based on JavaScript objects
- The following snippet connects to a MongoDB server, selects a collection and inserts a new feature

```
client = MongoClient("mongodb://localhost:27017") # Connect to DB
db = client["gis_database"] # Select a database
collection = db["map_layer"] # Select a collection
# insert the feature
collection.insert_one(
  { "type": "Feature",
    "properties": {"name":"Pisa"},
    "geometry": {"type": "Point", "coordinates": [ 10.41, 43.72 ] } } )
```

- Note: the `insert_one` call corresponds to the SQL `INSERT` query seen above

Going deeper

The rest of this tutorial is divided into six introductory hands-on sessions:

- Fundamentals of QGIS
- Working with OpenStreetMap
- Creating Maps with uMap
- Using GaiaGPS for Field Data
- Georeferencing in QGIS
- Introduction to the Leaflet Library

QGIS: a local application

Summer School on Digital Humanities

Web site: https://bit.ly/dt4h-gis

Augusto Ciuffoletti

10 giugno 2025

QGIS: a local application

- The user installs a GIS application on the PC

User

- In this scenario, the Web is a tool for exchanging data
  - but it is not directly involved
- Quantum GIS (QGIS) is an open-source GIS application
  - Developed and maintained by volunteers
  - First released in 2002
  - Here we use version 3.34 (Prizren)
- Runs on Windows, Linux, and macOS

QGIS Operation

- Acquires and aggregates layers from various formats
  - Includes both local data and remote databases
- Enables creation of new layers
  - Populated with customized features
- Among final output options:
  - Produce a graphic file (JPG, PNG, etc.)
  - Save in QGIS format
  - Publish on the *QGIS Cloud* (plugin needed)

Hands-on QGIS - Load a raster

Create a New Project

- Open QGIS and select Project -> New

Add a Raster Background Layer

- Layer -> Data Source Manager -> XYZ Tiles
  - You can also use the Ctrl-L instead of using the menu
- Double-click on OpenStreetMap
- Use the control pad to zoom in on a specific region

Understanding the Raster Layer

- The map is now displayed as a raster layer
  - Composed of multiple tiles, similar to an image
  - Cannot be modified within QGIS
- Various providers offer raster layers
  - *OpenStreetMap* is a free, open-source provider

Hands-on QGIS - Add a Vector Layer

Define a Vector Layer

- Layer -> Create Layer -> New Shapefile Layer
  - Or use the *New Shapefile* icon in the toolbar (third icon in the second row)
- Choose:
  - A filename to save the layer (e.g., Demo)
  - The feature type: *Point*, *Multipoint*, *LineString*, *Polygon*
    - In this example, use *Point*
  - A coordinate system (`EPSG:4326 WGS84`)
- Add new fields for the features in the layer
  - e.g. *Last visit* with type *Date* and click *Add to Fields List*
  - When finished, click Ok
- The new layer appears in the *Layers Panel*
  - To view the layers panel, View -> Panels and tick Layers
- Two layers shown, Demo and OpenStreetMap
- We can edit the Demo vector layer

Hands-on QGIS - Refine the layer definition

Further Configuration of a Layer

- Double-click on the Demo layer to set its properties
  - In *Symbology*, choose the graphic symbol and adjust its properties
  - In *Fields*, update feature attributes
    - you may want to add a new *last visit* field
    - for this enable editing with the pencil and add (or delete) a field
  - In *Labels*, select *Single label* and choose the field for labeling the points (e.g., select the *name* field)

Hands-on QGIS - Working with points

Populate a Vector Layer (with Points)

- Select the Demo layer and Layer -> Toggle editing
  - Or the pencil in the second toolbar
- Then select Edit -> Add Point feature
  - or the `Ctrl+.` shortcut
  - The mouse pointer changes to a crosshair
- Click on the map to add a new point
  - A box appears to set feature fields
- Repeat as you like
- To move a point feature,
  - menu Edit -> Edit geometry -> Move Feature
  - right click on the point to move
  - drag to the new position
  - left click to displace the selected point
- To exit edit mode, right-click on the Demo layer and select Layer -> Toggle Editing

Hands-on QGIS - Edit fields

Update Feature Attributes

- Right-click on the Demo layer and select Open Attribute Table
  - Use the bottom-right icons to adjust the view style
- Press `Ctrl+E` to enable table editing (or click the *Pencil* icon)
- Modify attribute values as needed
- Press `Ctrl+S` to save

Add an Attribute ("desc") to the Features

- Right-click on the Demo layer and select Open Attribute Table
  - Enable editing
  - Press `Ctrl+W` to add a new field (or find the "New Field" button in the toolbar)
  - Set the name and type (e.g., "desc" of type Text)
  - Click OK

Hands-on QGIS - Process fields

For each point compute a new field with distance from Rome in degrees

- Select a layer and click the Open Attribute Table button in the toolbar
- Click `CTRL+I` or the abacus icon in the attribute table window
- Input a name for the new field (e.g., *Lat*)
- Choose a type for the field (e.g., *Decimal Number*)
- Enter the following formula in the *Expression* box

```
distance(@geometry, make_point(12.5, 41.9))
```

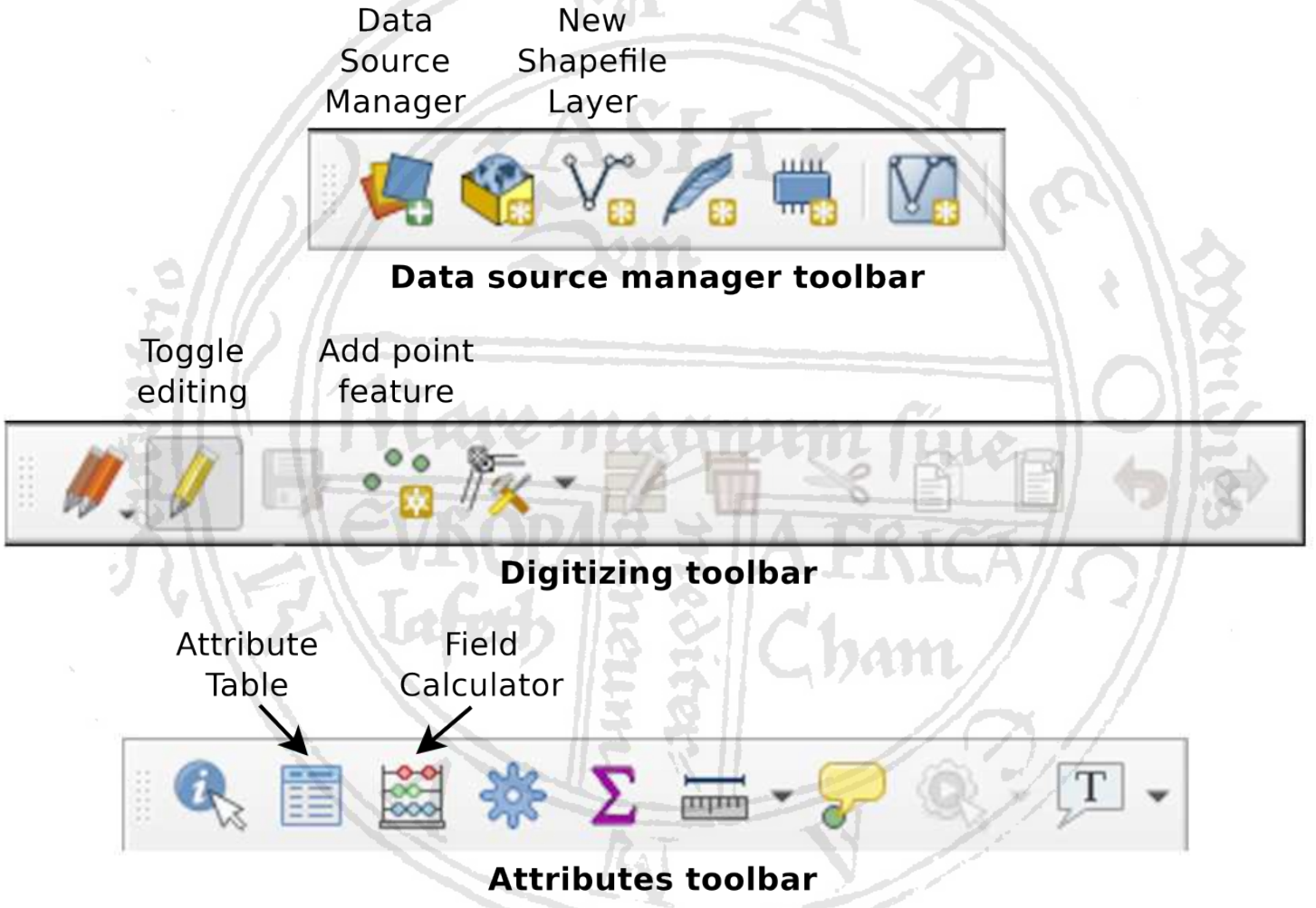
- The `distance` function takes two points
  - `@geometry` is the one corresponding to the row in the table
  - `make_point(12.5, 41.9)` corresponds to Rome (long,lat)
- Note: to see meters conversion is needed, from `EPSG:4326` to `EPSG:3857`, using the `transform` function

Hands-on QGIS - Save or export

Save Your Work

- Save the project in QGIS native format (Ctrl+S or Project -> Save)
- Export as an image (Project -> Import/Export -> Export Map to Image)
- Export in a portable vector format (Project -> Export DXF)

GUI Toolbar Icons (Quick Reference)



Lab Activity

- (Basic) North of La Spezia, there is a region called "Cinque Terre". The name comes from five fishing villages: Corniglia, Manarola, Vernazza, Monterosso, and Riomaggiore. Set a Point for each village and display a label with its name on the map.
- (Intermediate) Draw a sea route visiting all the villages, starting from Levanto (another small town to the north). For this create a new LineString vector, enable editing, select `Add Linear Element` and mark waypoints with the left button. Right button to close the LineString.
- (Intermediate) Convert the line to a new layer of vertices using Vector -> Geometry Tools -> Extract Vertices
- (Advanced) Compute the longitude and latitude of these points, and label each one with a string "(long, lat)" using the concat function in the calculator.

Dynamic Web Map Services

Summer School on Digital Humanities

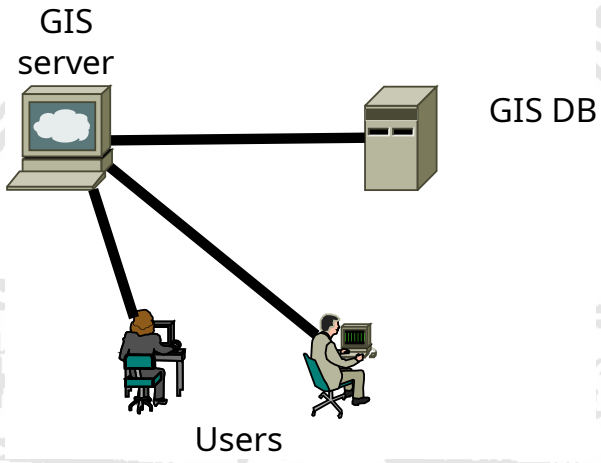
Web site: https://bit.ly/dt4h-gis

Augusto Ciuffoletti

11 giugno 2025

Dynamic Web Map Services

- A local application does not facilitate map sharing
- We need an interactive web-based map service



- Web Mapping enables cartographers to maintain a shared map
  - The cartographer accesses the mapping service via a web browser
  - The server generates a web page integrating the map
  - Embedded code connects to a remote database to retrieve and update data
  - The cartographer can modify the view or input new data

Web GIS vs. Desktop GIS Applications

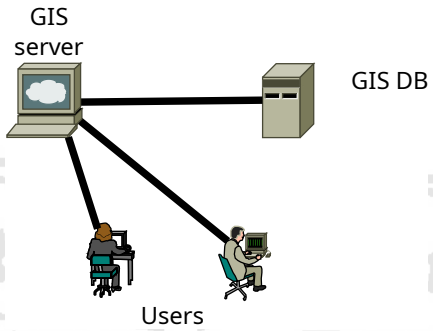
- Compared to a desktop GIS application (like QGIS):
  - No installation required
  - Platform-independent (works on any OS)
  - Responsive design for different devices (PC, tablet, smartphone)
  - Designed for sharing — requires access control mechanisms
- Developing such a dynamic application requires a specialized JavaScript library

Tools for Web Maps: JavaScript Libraries

- JavaScript enables complex functionalities in web pages
- The Leaflet library allows web pages to interact with GIS servers and store user data
- Users can modify and update the map interactively
- This setup creates a complex architecture:
  - The user downloads a web page (designed by the cartographer)
  - The page interacts with a PostGIS server and a raster data repository
- We will explore OpenStreetMap, which is implemented using the *Leaflet* library
  - ...and explore *Leaflet* in the last session

Example of an Open Web Map Service: OpenStreetMap

- The OpenStreetMap server (`www.openstreetmap.org`) renders a dynamic map in the browser, drawing data from a public database



- Public Collaboration:
  - Anyone with write access can update the database—all changes are publicly visible
  - There is no option for a private workspace
- Using the built-in **Id** editor:
  - Easily create features like a bar, swimming pool, or street
  - **Save** changes cautiously—they might become visible to everyone

Getting Started with OpenStreetMap

- Open a browser and visit OpenStreetMap
- To access the service:
  - **Sign in** with an existing account or a third-party service (e.g. Google, Microsoft, Facebook) or
  - **Register** a new account

Creating a Point Feature in OpenStreetMap

- To start editing press the Edit button top left
- To add a point feature (but do not press **Save**):
  - Zoom in using the trackpad until **Edit** is enabled
  - Select the **Edit** option (opens the *iD* editor)
  - Zoom until the "Zoom in to edit" banner disappears
  - Click the **Point** tool in the top toolbar (it turns blue)
  - Click on the map to place the point
  - Choose a feature type (e.g., **Café**) from the left sidebar
  - Fill in relevant attributes
- To stop editing click the logo icon top left (or reload the page)

Additional Editing in OpenStreetMap

- To draw a Line or Area:
  - Click to place each vertex
  - Press *Esc* or double-click to finish
- To edit an existing feature:
  - Right-click to access transformation options:
    - Convert to a circle
    - Convert to a point
    - Align angles to 90°
    - Flip or rotate
- Keyboard shortcuts:
  - Ctrl+C / Ctrl+V to copy and paste
  - Ctrl+Z to undo changes
- Pressing **Save** commits changes to OpenStreetMap—please refrain from saving test edits

Lab Activity

- Scenario: South of Pescara lies "Francavilla al Mare," a seaside resort town
  - Locate "Lido Merope"
  - Add an Area for the beach
  - Set Beach Resort as the feature type
  - Set the Name field to "Spiaggia del Lido Merope"
  - Undo...

Sharing Maps: The GIS Cloud

Summer School on Digital Humanities

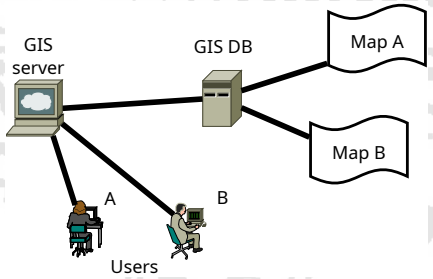
Web site: https://bit.ly/dt4h-gis

Augusto Ciuffoletti

11 giugno 2025

Sharing Maps: The GIS Cloud

- OpenStreetMap (OSM) does not support adding custom layers
  - It lacks infrastructure for proprietary data storage
- Desktop GIS applications (e.g., QGIS) allow users to overlay OSM data with their own layers
- Alternative services provide infrastructure for collaborative map sharing



- In these services, the server acts as an intermediary between users and data storage
- Unlike OSM, access and sharing restrictions are enforced

The uMap Service

- A stable project (launched in 2014)
- Open-source (can be installed on a private server)
- Uses OpenStreetMap as a background raster layer
- Users can:
  - Import and display vector layers
  - Connect to external databases
  - Create new maps stored within the uMap infrastructure
- Supports *popup* feature annotations with:
  - Text, images, and web links
- Maps can be shared and collaboratively edited by a restricted user group

A uMap Tour (Sect. 5 in the Booklet)

- Open a browser and visit https://umap.openstreetmap.fr
- Log in using OpenStreetMap credentials (or via Bitbucket, or GitHub)
  - It is also possible to explore uMap without an account (with limited functionality)
- Click "Create a map"
  - An OpenStreetMap raster background will appear

Editing Your Shared Map

- Upon map creation, you are in Edit Mode
  - To exit, click the blue Save draft button, then View
  - To re-enter edit mode, click the pencil icon Edit in the top-right corner
- Two toolboxes appear on the right:
  - Upper box: feature creation tool
  - Lower box: map management tools
  - On the left panel, click  to search for a town (e.g., ”Pisa”)
- If town names are displayed in French, switch the raster layer:
  - Click  and select "OpenStreetMap"

Adding a Feature

- Choose a feature type from the right-hand menu:
  -  Point
  -  Line
  -  Area
- Click on the desired feature type, then interact with the map:
  - A single click places a point
  - Multiple clicks define a line or area
  - A double-click ends the line or closes the area
- Once the feature is created, a form appears:
  - Enter the *name* and *description* of the feature
  - Explore additional customization options
- To edit an existing feature, click it and select the pencil icon
  - Click the question mark near *Description* for formatting tips

Special Content in the **Description** Field

The *Description* field can contain interactive elements:

(Note: these appear only after Save and View)

- Add an image by inserting its link within double braces:

```
{{http://gardenersnet.com/flower/pics/daffodil04.jpg}}
```

- Add a hyperlink using double brackets:

```
[[http://www.google.com]]
```

- Embed a video (e.g., YouTube) using triple braces:

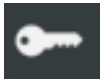
```
{{{https://player.vimeo.com/video/56810854}}}
```

Creating a Slideshow

- A slideshow cycles through the map features sequentially
  - Features are displayed in the order they were added
- To enable the slideshow:
  - Click the  gear icon and select *Slideshow*
  - Enable *Slideshow mode*
  - Set the time interval between slides
  - Optionally, enable auto-play on map load
  - Click Save
- Familiar playback controls ("Play," "Pause," "Next," "Previous") appear in the bottom right corner
- If auto-play is enabled, the slideshow starts automatically upon loading the map
- Not fully functional, but may be useful

How to Share, Embed, and Export

To share your map with specific users or anyone

- Click  (right panel)
- Adjust editor permissions and visibility settings

To embed your map in a webpage

- Click  (left panel)
- Export features in various formats
  - GeoJSON – JavaScript-compatible format
  - KMZ – Google Maps format
  - GPX – GPS-compatible format
  - CSV – spreadsheet compatible format
- Copy the embed code (”iframe” tag) for use in web pages
- Configure a custom URL with query

Time for a uMap *Challenge*! (Step 1)

Geolocate Yourself

- Open https://umap.openstreetmap.fr/it/map/dhss_2025_616270
- Add a *point* feature at a location you like (e.g., your town, favorite hike, birthplace)
- Enter your nickname in the *Description* field
- Save the point as a visible feature
- Warning: the map is public: avoid personal data

Time for a uMap *Challenge* (Step 2)

The Map to the Tower

- Help a friend visiting Pisa for the first time:
  - Add points of interest (leaning tower, train station, whatever...)
    - Fill "Description" fields as you like
    - try with an image of the tower,

      https://upload.wikimedia.org/wikipedia/commons/6/66/The_Leaning_Tower_of_Pisa_SB.jpeg
  - Create a map showing a route from the train station to the Leaning Tower
    - Draw a path from the station to an imaginary bus stop, then the bus route, and finally the walk to the tower
    - At the bus stop, enter details: bus number, direction, stop name, and estimated travel time

On the field: Gaia GPS

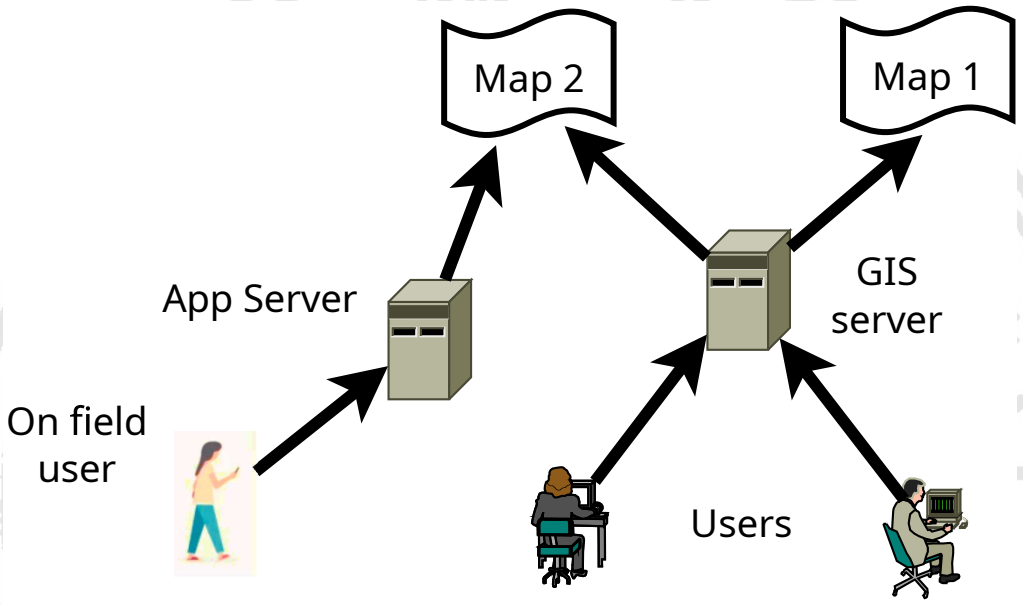
Summer School on Digital Humanities

Web site: https://bit.ly/dt4h-gis

Augusto Ciuffoletti

11 giugno 2025

On the field: Gaia GPS



- The system now includes a user on the field
  - as well as autonomous devices like drones
- Gaia GPS is one of the many useful apps for field GIS
  - ...and one of the few available on both Android and iPhone

Using the Gaia GPS App

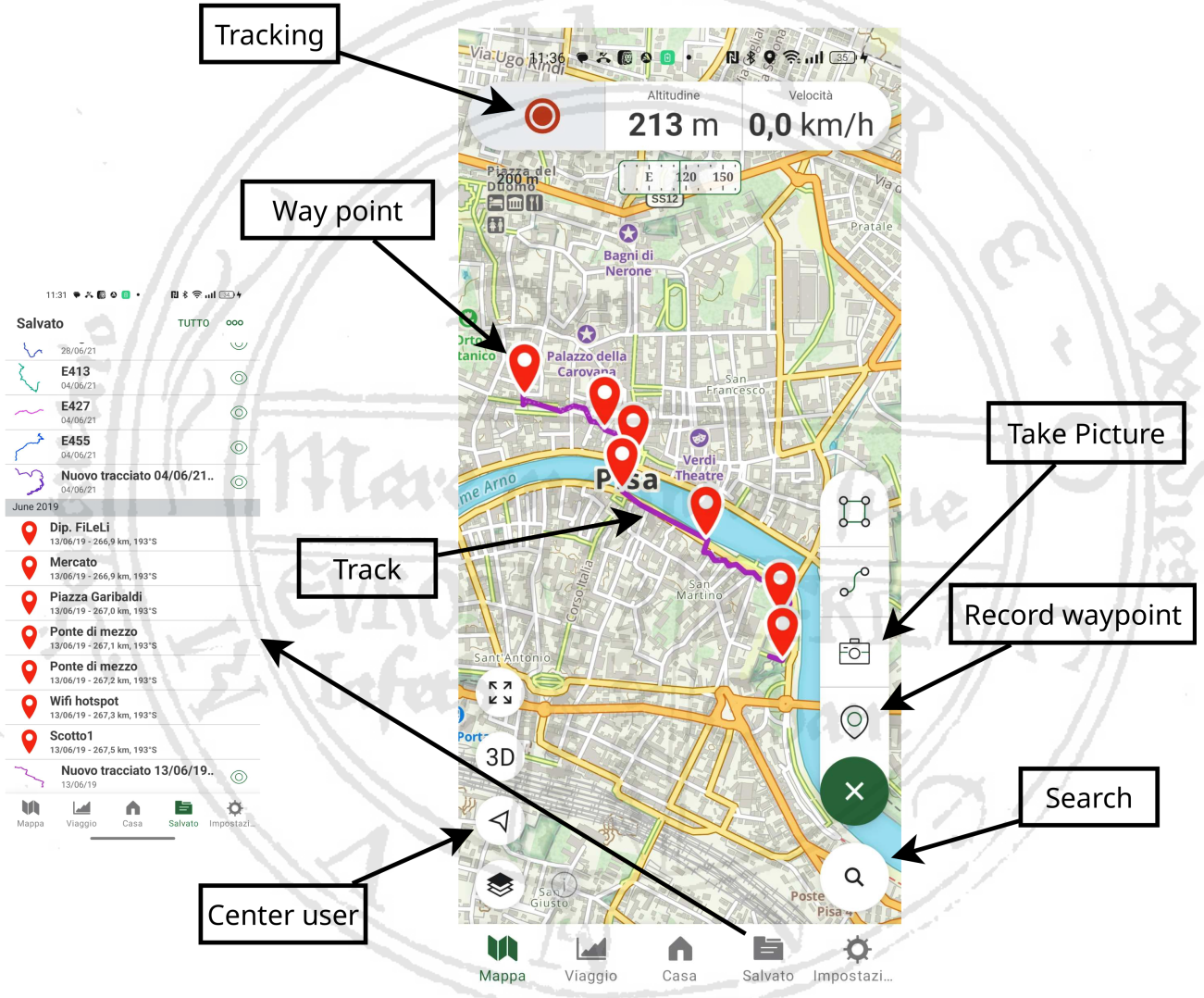
- Register to the service (use a free account for practice)
- Key activities:
  - Record a live track of an itinerary (line feature)
  - Record waypoints (point features)
  - Add written annotations (as point attributes)
  - Attach pictures to points (as point attributes)
- Gaia GPS supports all these activities
- Features can be exported as GPX, KMZ, or CSV local files
  - or shared with users through socials
- A companion Web service assists in processing features
  - here GeoJSON export is also available
- Features can be imported too

Import a sample map

- On your smartphone, use the *Google lens* to reach the Google Drive hosting the course material:

- Enter the GPX folder, and tap the file "PisaPonteDiMezzo.gpx"
- Select to "Open with" the GaiaGPS App
- The map is going to be used in the next steps

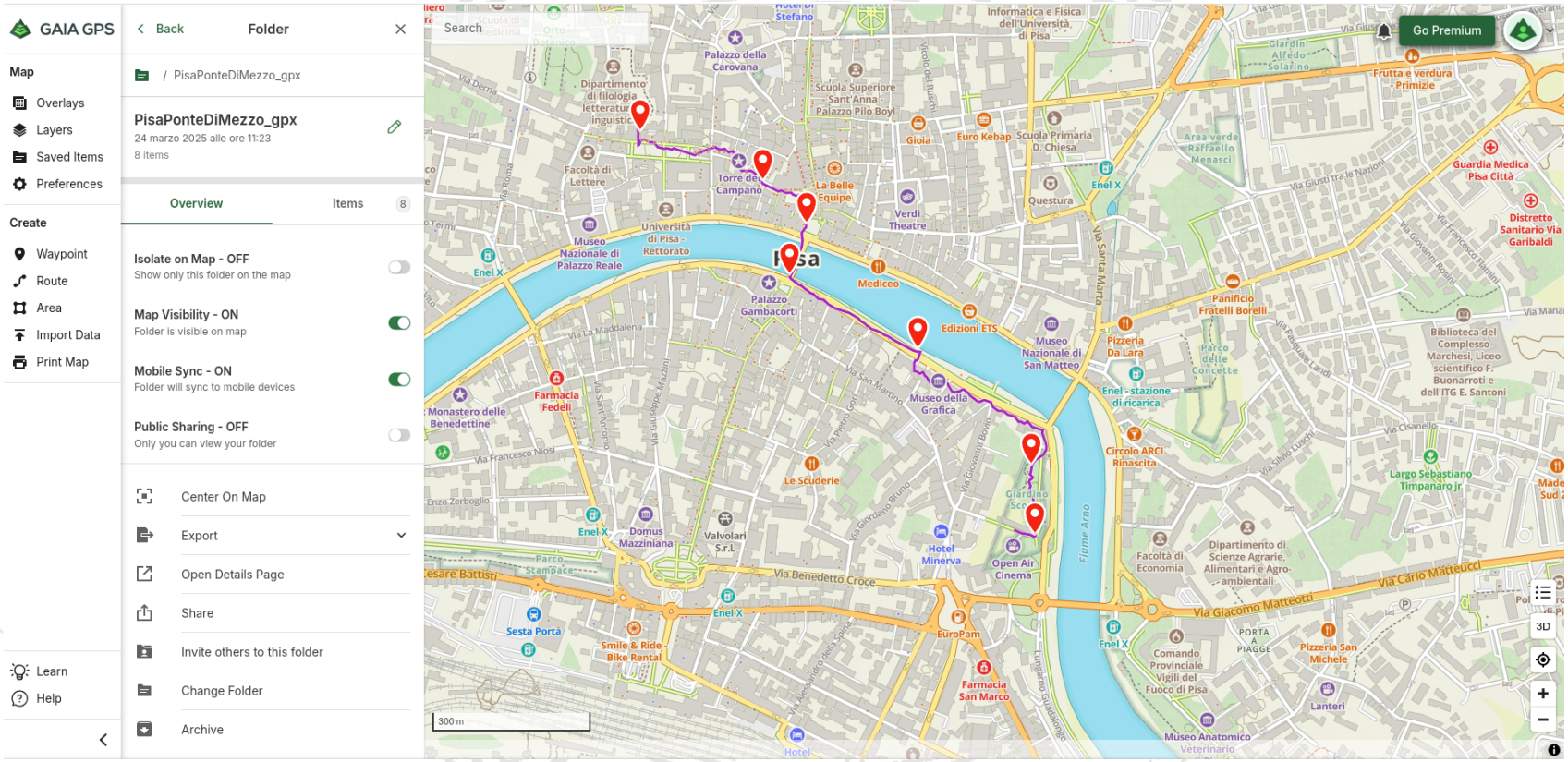
Gaia GPS: the *main* screen



Processing the track on the Web service

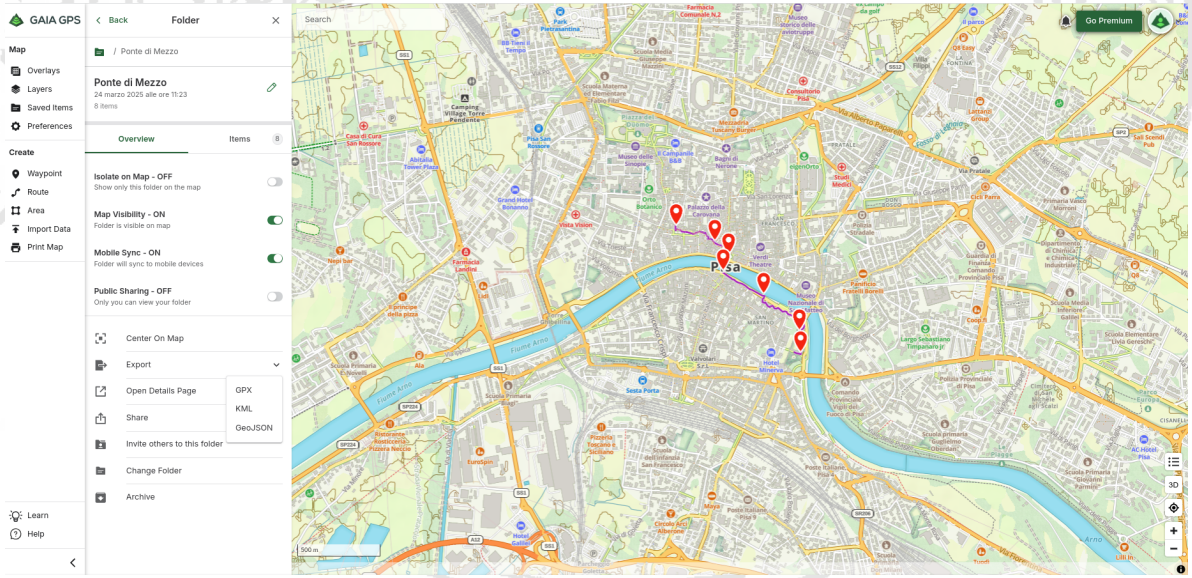
- Sharing your GPX via GaiaGPS cloud storage allows easy track viewing on a computer
- Access the service at https://www.gaiagps.com/ with your Gaia GPS credentials
- Click on the Saved Items icon on the left menu
- Select the track you uploaded in the previous step to see the map

Gaia GPS Web Service



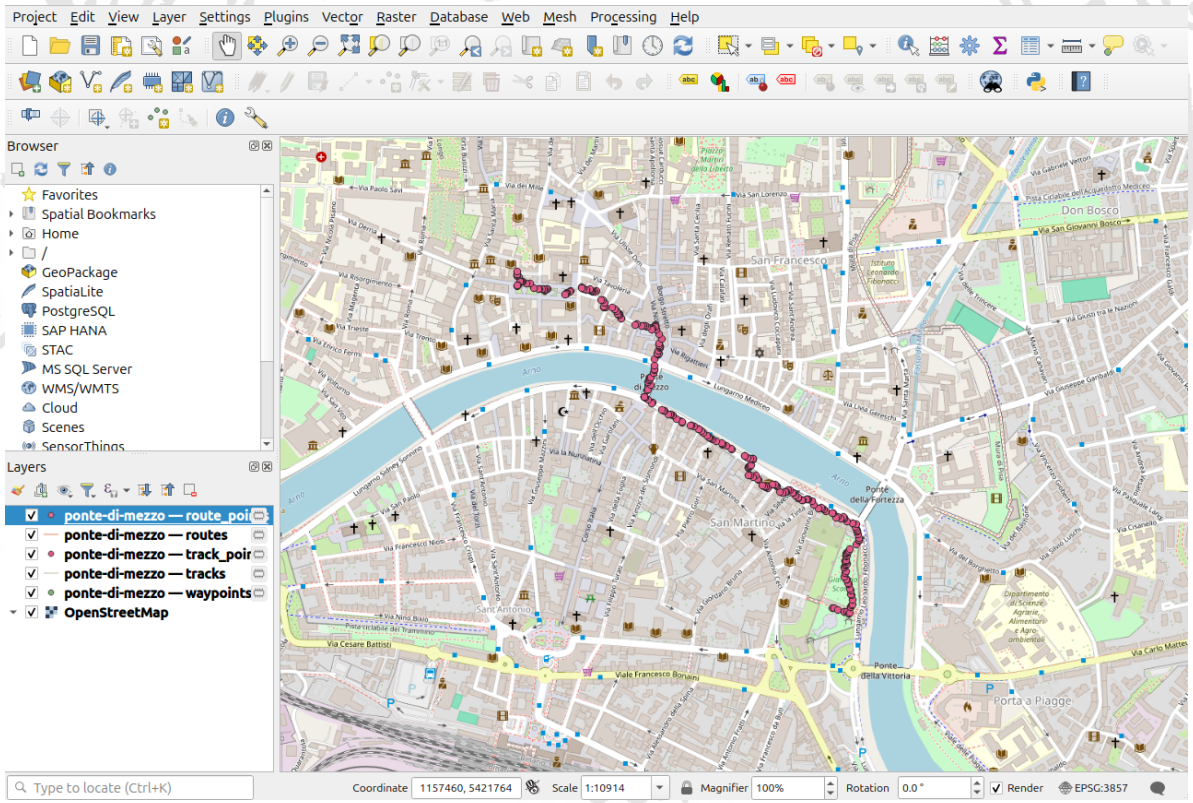
Downloading the data to a PC

- In the left toolbar, select *Saved Items*
- Click on the feature to export
- Scroll the feature frame to find the *Export* menu
- Choose an export format from the list
- Save the file to your computer



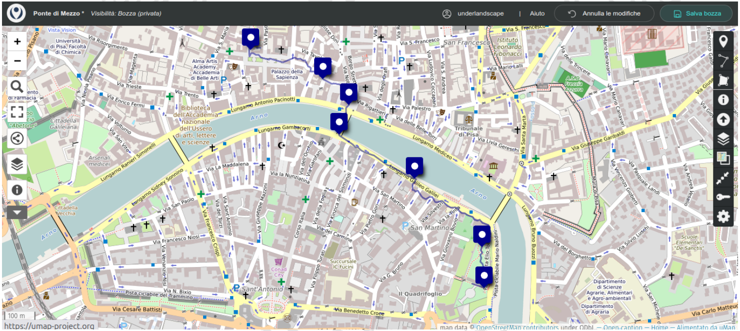
Import a Gaia GPS track into QGIS

- After downloading the track to your PC:
  - Open QGIS and create a new project
  - Using the Data Source Manager add an OpenStreetMap layer
  - Layer -> Add layer -> Add vector and navigate to your GPX
  - Add, select all, OK, and close the form



Import a Gaia GPS track into uMap

- After downloading the sample track on your PC:
  - Access the uMap Web service and create a new map
  - Press the import button on the right
  - Locate and select your GPX file



Recording a track with Gaia GPS

- Select "Map" view with the bottom bar
- Center map at your position with the a button in the left corner toolbar
- Start recording with the top/left red button
  - Running time is displayed
- While walking, record waypoints and take pictures using the + button
  - Add notes to waypoints
  - Associate pictures with waypoints
- Tap on the running time to stop and save
- Access saved tracks via the Saved button
  - Left screen in figure
- Tap a feature to export it in GPX, KML, or CSV format
- The track is now available through the GaiaGPS Web Service seen above

Track your walk: a Gaia GPS *'challenge'*

- Select a nearby point of interest (bridge, pub, or tower)
- Start track recording in Gaia GPS
- Walk to your destination, capturing photos and annotating points of interest in the app
- Upon returning, visit the Gaia GPS website and download your track as a GPX file
- Import your track into a new uMap project and/or
- Add it to the shared map at `https://umap.openstreetmap.fr/it/map/dhss_2025_616270` (or to your own map)
- Include your photos in the waypoint descriptions:
  - Use the image link found on the GaiaGPS Web Site
    - Select a waypoint with an image
    - Scroll down and "Open Details Page"
    - Click on the image and right click again to select "Copy image link"

Back to QGis: Georeferencing

Summer School on Digital Humanities

Web site: https://bit.ly/dt4h-gis

Augusto Ciuffoletti

12 giugno 2025

Back to QGis: Georeferencing

- Georeferencing involves transforming an image into a map
  - assigning geographic coordinates to each pixel in the image
- To achieve this, match points on the image with corresponding locations on an accurate reference raster (e.g., OSM)
- A georeferencing tool then calculates the coordinates for all pixels
  - Accuracy improves with the number of reference points
  - The image may need morphing (non-linear transformation)
  - Optimal reference points are distant and non-aligned
- QGIS provides tools for this task

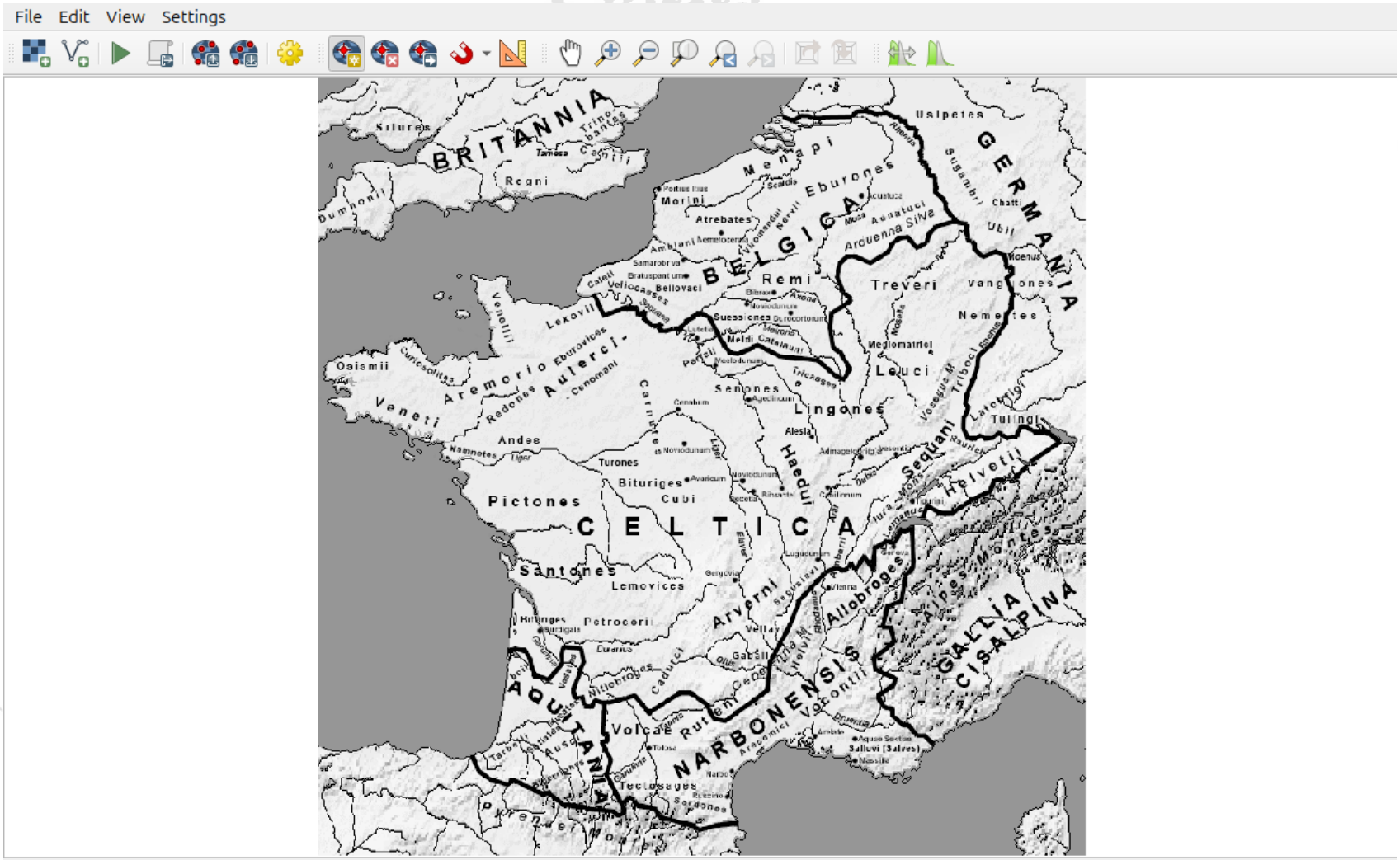
Georeferencing: Preparation

- We want to georeference the map avaliable at
  - Download the *png* file

```
https://it.m.wikipedia.org/wiki/File:Map_Gallia_Tribes_Towns.png
```

- Create a new project and load the reference raster (OSM)
- Adjust the scale to match the area covered in the map
- Observe the code in the bottom right corner: EPSG:3857 (WGS84 - Pseudo Mercator)
- Open the Georeferencer tool:
  - Select Layer -> Georeferencer... to open a new window
- In the *Georeferencer* window:
  - Select File -> Open Raster
  - Locate and open the image file you downloaded

Unreferenced image loaded



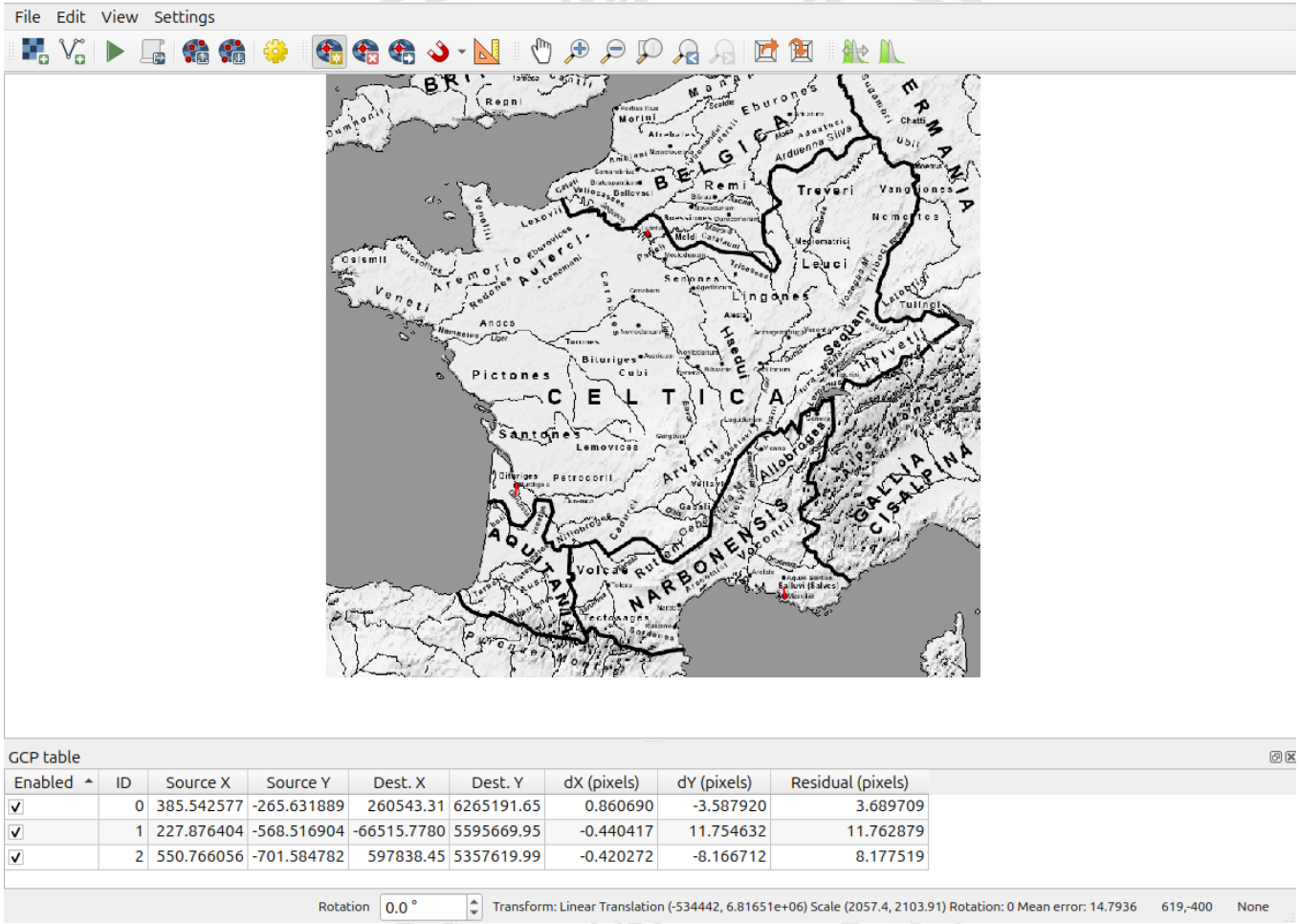
Setup the transformation type

- Configure transformation settings:
  - Select Settings -> Transformation Settings
  - Choose a transformation type (TPS is generally suitable)
  - Ensure the SR is set to EPSG:3857 - WGS84/Pseudo-Mercator
  - Specify a target file for the result
  - Enable "Load in project when done"
  - Click OK to apply the settings and return to the Georeferencer window

Matching Points

- Repeat the following steps for at least three (distant, non-aligned) points on your map image:
  - Identify a recognizable detail on the map image that also appears on the reference raster
    - e.g., in an ancient map of France, *Lutetia* corresponds to modern Paris
    - Use arrow keys to move and the mouse wheel to zoom, or select the *Hand* icon to use the mouse
  - Click when the crosshair is positioned over the reference detail (e.g., *Lutetia*)
  - A window appears to input the coordinates
  - Click the From Map Canvas button
  - The map and dialog disappear, and you return to the OSM raster with a crosshair pointer
  - Locate the corresponding point (e.g., *Paris*) on the raster and click
  - QGIS extracts the geographic coordinates from the OSM raster
  - The map reappears with the coordinates filled in
  - Click OK and repeat for at least three points

The map before georeferencing



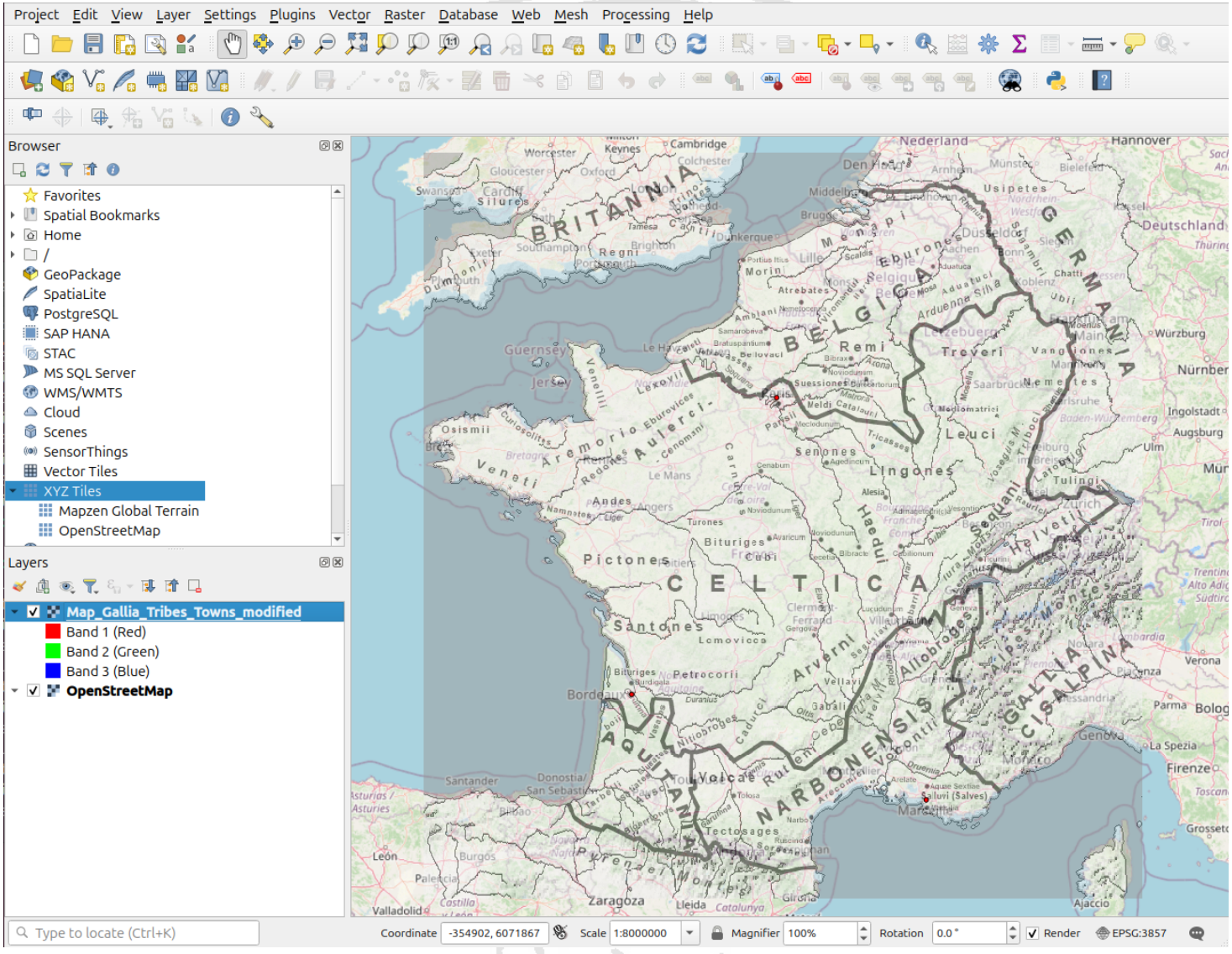
Running the Georeferencer

- Once all reference points are set, apply the georeferencing algorithm
- Click the green triangle in the *Georeferencer* toolbar to start the process
- A pop-up will confirm completion
- Keep the Georeferencer window open and switch to the main QGIS window to inspect the result

Inspecting the Result

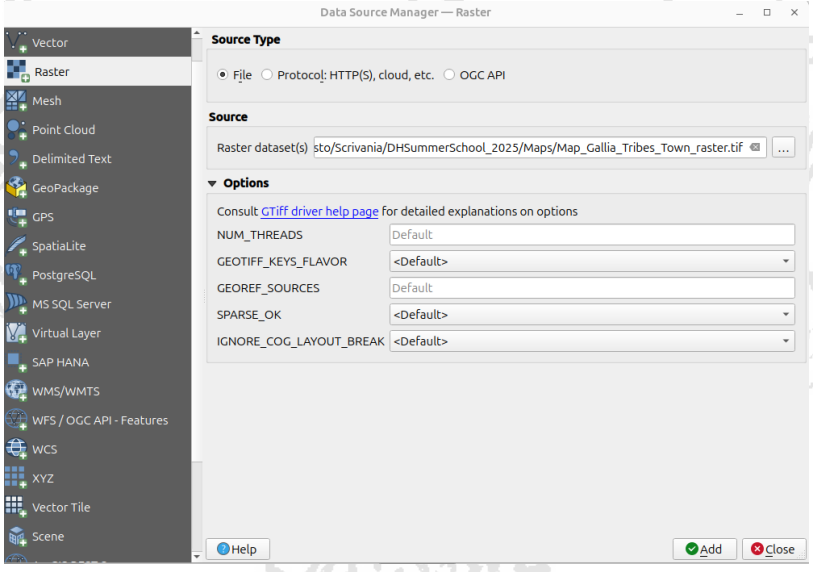
- The image appears as a new raster layer in the *Layers* panel
- To assess the georeferencing accuracy, adjust transparency:
  - Right-click the new layer in the Layers panel and select Properties -> Transparency
  - Set Global Opacity to approximately 50
- The next slide illustrates an OSM raster of France with a georeferenced historical map of ancient tribes
- The three reference points used: Paris, Marseille, and Bordeaux
- Observe how the northern coastline differs between the maps
- If the result is unsatisfactory:
  - Remove the layer
  - Return to the *Georeferencer* window to add more points
  - Repeat the georeferencing process
    - Hint: use small islands instead of towns

Referenced image generated



Use Your New Raster in QGIS

- During the georeferencing process, you specified a location to save the new raster
- To load it in QGIS, open a new project and access the Data Source Manager
  - Select Raster as the data source type
  - Click File to choose the raster format
  - Browse your filesystem and set the Source field to the path of your new raster



More Resources

- Find in-depth QGIS tutorials at
  https://www.qgistutorials.com/en/
- Access geographic data (such as OpenStreetMap) from regional and global sources:
  - https://earthexplorer.usgs.gov/ (Explore available *datasets*)
  - http://wms.pcn.minambiente.it/mattm/servizi-di-scaricamento/ for downloading *WFS* resources to import into QGIS
- Try an engaging tutorial: https://www.qgistutorials.com/en/docs/3/working_with_terrain.html
  - Learn to add contour lines to QGIS maps

Make your own GIS service

Summer School on Digital Humanities

Web site: https://bit.ly/dt4h-gis

Augusto Ciuffoletti

13 giugno 2025

How to proceed

- The tutorial consists in the step-by-step creation of a simple app that:
  - Displays a map
  - Allow the user to add markers to the map
  - Exports the markers as a GeoJSON string
- The tool we are going to use to practice the *leaflet* library is Stackblitz (`https://stackblitz.com/`), an online IDE for JavaScript
- The code for each step can be viewed, tested, and modified as a Stackblitz project
- The link to each project is in the title of each slide, and in the course website page dedicated to this topic.

Using the Stackblitz IDE

- Follow the project link for the first step
- In the right frame you see the preview of your service, showing a map
  - the URL on top of the frame is functional: try it...
- In the left frame there is the project content
  - The *README.md* describes the step
  - The *index.html* is the HTML code for the page
  - The *index.js* file is the javascript code using the leaflet library
  - The other files are not of interest
- The selected file is shown in the center frame
  - You can edit the code and see what happens
  - For instance, try to change the string in line 10 in index.html and notice the preview change
  - Your edits remain local. To save your project you should register on Stackblitz

Step 1: the background (project)

- The first step in our tutorial consists of using the *Leaflet* library to display an OpenStreetMap raster
- How to:
  - The reference to the library is in the *package-lock* file
  - In the HTML file:
    - a `head` element with the CSS for the *Leaflet* library
    - a `div` element for the map (its `id` is `mapid`)
  - The index.js file contains the JavaScript code of our App
  - The capital `L` stands for the *Leaflet* class
  - So we create a map with two parameters
    - the id of the DOM element hosting the raster (our `mapid`)
    - A JavaScript object that describes position of map center and zoom level
  - Next we add a background raster, which is *OpenStreetMap*

Step 1: Lab activity

- Browse the web to find the coordinates of a place at your choice as the center of the raster
- Modify/remove the zoom factor

IMPORTANT:

- relax: you **cannot** damage my repo (you'd need my credentials)
- you may *Fork* (button on top-left corner) a branch in a repo of your own (recommended not strictly needed)
- you can undo unsaved updates with Ctrl-z
- after forking and signing up you can save your work

Step 2: show the coordinates (project)

- When the user clicks on the map an alert appears with the coordinates of the click
- How to:
  - The HTML file is identical, we added management of a click event in the JavaScript
  - We apply the *on* method to the map to catch *click* events
    - the first parameter is the name of the event we want to capture
    - the second parameter is a callback that takes the event description as a parameter
    - the callback displays an alert containing data extracted from the event descriptor `e`
    - the event descriptor is an object
    - we extract the `lat` and `lng` fields in the `latlng` field.

Step 2: Lab activity

- create a named and write within the coordinates. Use

```
document.getElementById("myDiv").textContent = ...
```

Step 3: collect coordinates (project)

- Each click on the map adds a marker, and their coordinates are shown on the page
- How to:
  - We add a *div* for the coordinates in the html
  - In the JavaScript we add to the event callback the creation of the new marker
    - its position is computed using the `latlng` field in the event descriptor
  - The coordinates are appended to the list in a `div` element of the DOM

Step 3: Lab activity

- Reverse the order of coordinates: longitude is shown first in the bottom list

Step 4: enumerated markers (project)

- A progressive index is assigned to each new point
- The index is shown in the list and added as a *title* field in the marker definition
  - the title field is automatically displayed when the mouse hovers on the marker
- How to:
  - HTML is the same
  - We add a new global variable `n` in the JavaScript
  - The event callback increments the variable each time it is run
  - The value of `n` is displayed on each line in the list
  - The marker constructor now takes a second parameter containing the marker options
    - among which the `title` option

Step 4: Lab activity

- Configure the marker as draggable (ignore that the displayed coordinates become inconsistent)
  - hint: in the `marker` variable definition add a `draggable: true` property after the title, separated by a ","

Step 5: all markers in an array (project)

- Record the markers in an array to have them accessible
  - in the previous steps the marker was a local variable in the callback
- How to:
  - Create an array for the markers
  - Push markers in the array
  - `n` index corresponds to array length
    - no need to increment it

Step 5: Lab activity

- Create a button that fades-out the markers
- Hint
  - loop through all items in the array with a for loop

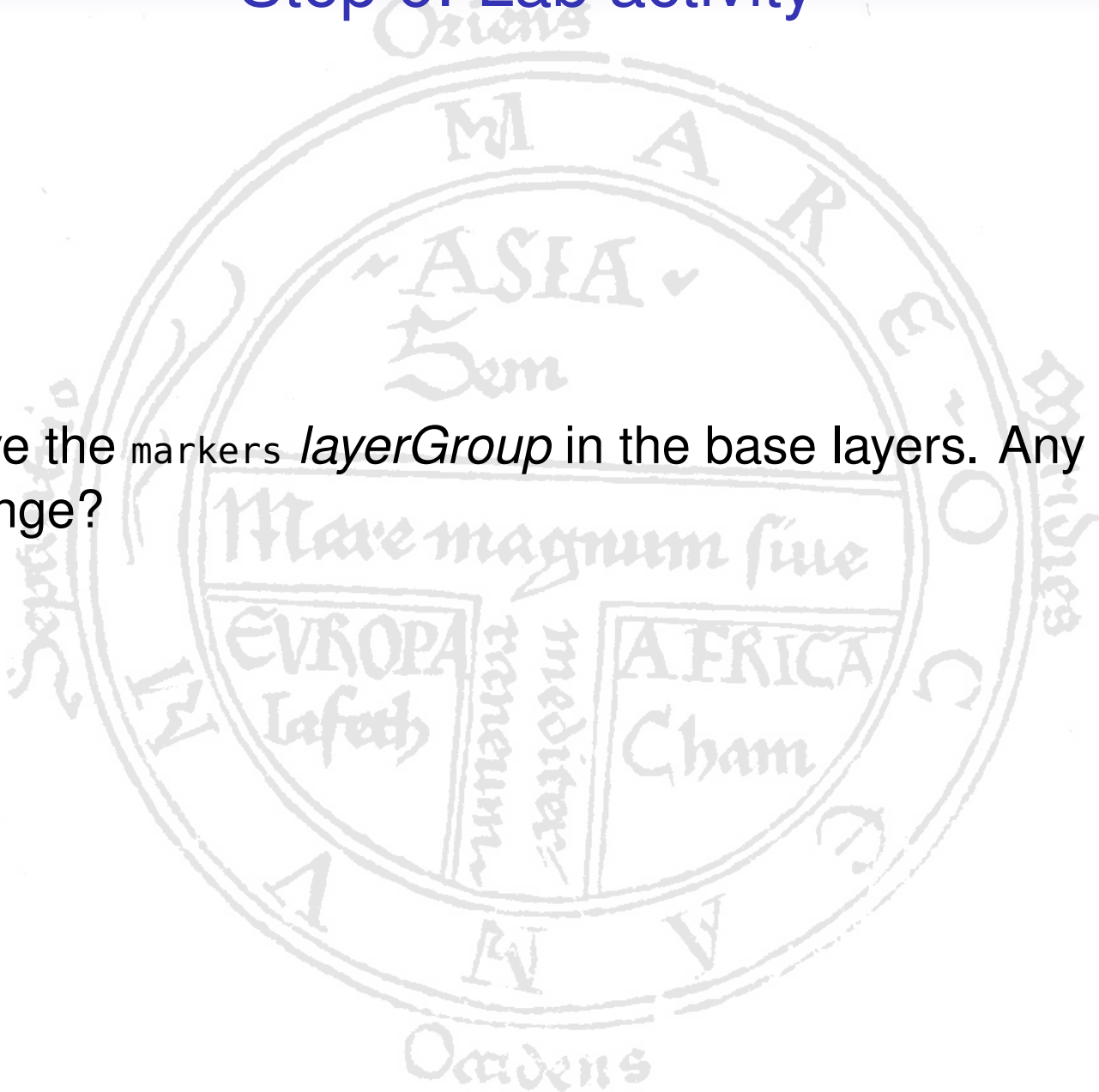
```
for (let m in markers) {...}
```

  - use the `setOpacity(0.5)` on each marker

Step 6: all markers in a layer (project)

- Having all markers in a layer is more practical than in an array
- How to:
  - HTML is always the same
  - Replace the array with a *layerGroup* object added to the map (`markers`)
  - Replace the *push* operation with an *addLayer* applied to the layerGroup
  - Add a layer control icon to the map to toggle layer visibility
  - The control creation takes two object arguments
    - One for the base layers (radio button, just one)
    - One for the overlay layers (multiple choice)
  - See the effect on the layers button top-right in the map

Step 6: Lab activity



- move the `markers` *layerGroup* in the base layers. Any change?

Step 7: GeoJSON serialization (project)

- It is handy to have a standard string representation of a piece of data (serialization)
  - e.g. to store the data in a file
- The GeoJSON representation can be easely transformed into a JSON string, and viceversa
- We want to print in the console the JSON string for our markers
- The `toGeoJSON` method converts the markers layer into a JavaScript object with the GeoJSON format
  - alas, in this way we lose the *title* field
- The `stringify` method serializes the object as a String object
- The string is finally recorded put on the display

Step 7: Lab activity

- Copy the generated GeoJSON and feed it to an online viewer, like https://www.geometrymapper.com/
- Is there any way to record the *title* field in the JSON string?
- Study the GeoJSON format in the console and find a solution