ICT for Civic Data — Turin University 2025-26

Enrichment and Cleaning

Crash Course — Day 3

civicliteraci.es

Day 3

> Today's plan

Morning:

- Finish your individual data search and map from yesterday
- Shared tutorial: enrich the flood map with health facility data from OpenStreetMap

Afternoon:

- Verification and cleaning exercise in Google Sheets using the FloodArchive data

Finish Your Map

> Continue from yesterday

Pick up where you left off:

- If you have an angle but no data yet: **Find** a source, **Get** it, put it on the map
- If you have data but it's not on the map yet: ask Gemini CLI to add it as a layer
- If your map works: improve it (popups, colours, legend) or add a second dataset

Use the **chatbot** for questions and discovery, the **CLI** for file work.

Remember: ask for a **"simple Leaflet map"** to avoid the agent over-engineering.

> Reminder: common issues and solutions

- **Agent publishes to wrong branch?** Tell it "publish on the main branch, not gh-pages" — or change Settings → Pages to use the gh-pages branch
- **Agent thinking too long?** Press **Escape** (maybe multiple times), then ask it why it's taking so long
- **Map not updating?** Hard refresh: **Ctrl+Shift+R** (Windows) or **Cmd+Shift+R** (Mac)
- **Agent fails to download a file?** Download it yourself and drag-and-drop it into the Codespace
- **Agent builds something too complex?** Ask for a **"simple Leaflet map"**

From Case Study to Demonstration

> The case study

Health facilities in flood-prone areas are at risk. When a flood hits, the facilities that should serve the community may themselves be damaged, understaffed, or inaccessible.

This is a **concrete case study**: specific, grounded, observable.

> From case study to angle to proposal

The angle

Knowing which facilities are at risk helps with:

- **Prevention**: strengthen facilities *before* the next event
- **Staff safety**: contingency plans for health workers in flood zones
- **Response**: knowing which facilities are safe supports evacuation and shelter decisions

The proposal

A **facility risk dashboard** that field staff can use on the go during response, showing which facilities are in flood zones and their status.

Built on data. Grounded, realistic, approachable, impactful.

> What do we build to demonstrate this?

The proposal says: "we can help you identify at-risk facilities."

The demonstration says: **here, look — we already did it for one country.**

We will build a map that shows flood events and health facilities, with a filter to highlight facilities in flood risk zones. That map is the "show don't tell" artifact.

> What we will build

A map with:

- Flood events from FloodArchive (filtered to one country)
- Health facilities from OpenStreetMap
- A **filter** (dropdown or checkbox) to show only facilities within 1km of a flood event

To get there, we need to go through several pipeline steps. Let's walk through them.

> Before we start: how to prompt for complex tasks

By now you have used your AI tool for many tasks. Some worked well. Some didn't.

When a task has multiple steps, the prompt needs **structure**. Five elements:

1. **Objective**: what are you trying to achieve?
2. **Steps**: what should the AI do, in what order?
3. **Reasoning**: why each step? (so the AI doesn't skip or improvise)
4. **Output shape**: what should the result look like?
5. **Human verification**: how will you check the result?

A prompt without these elements is an invitation for the AI to fill in the blanks.

> Trust the code, not the reasoning

Trust	Don't trust
The AI's **technical capability**: it knows how to write code, query APIs, convert formats.	The AI's **reasoning about data content**: it does not understand what the data means.
Scripts are **deterministic**: the same script on the same data produces the same result every time.	Never ask the AI to directly modify the content of your data. Ask it to write a script that does it. You review the script.

> AI gets lazy with large files

When processing large datasets, AI models tend to:

- > Read the first 50 rows carefully
- > **Assume** the rest is similar
- > Make up or skip data after that

This is why we break work into steps and verify at each stage. If the AI handles your whole dataset in one pass, the end of the file is less reliable than the beginning.

> Step 1: Get the flood data for one country

If you don't already have the FloodArchive file, upload it to your repo.

Ask Gemini CLI to **write a script** that filters it to your chosen country and saves the result.

> « Write a script that reads FloodArchive_clustered.csv and keeps only the rows where Country is [your country]. Save the result as a new CSV file. I'm a beginner, please explain what you're doing in simple terms. »

This is a **Clean** step: reducing the dataset to what you need.

> Step 2: Create a map for your country

Ask Gemini CLI to build a simple Leaflet map from the filtered data.

« Create a simple Leaflet map that can be published on GitHub Pages. Display the flood events from [your filtered CSV] as circles. Add popups with date, cause, and severity. Center the map on [your country]. Publish it on GitHub Pages from the main branch. I'm a beginner, please explain what you're doing in simple terms. »

Push and check the live page. This is a **Verify** step: does the data look right on the map?

> Step 3: Get health facilities from OpenStreetMap

OpenStreetMap has health facility locations worldwide via the **Overpass API**:

- **No registration, no API key**
- Queries can take **30+ seconds** for larger countries. That is normal.

« Query the Overpass API for all hospitals and clinics in [your country]. Save the result as a GeoJSON file. I'm a beginner, please explain what you're doing in simple terms. »

This is a **Get** step. Check: how many facilities? Does the number seem reasonable?

> Step 4: Add facilities to the map

« Add the health facilities from [your GeoJSON file] as a new layer on my Leaflet map. Use a different colour from the flood events. Add popups with the facility name and type. Push and publish on GitHub Pages from the main branch. »

Push and check. This is another **Verify** step: do facilities appear where you expect them? Do flood events and facilities overlap in some areas?

> Step 5: separate at-risk facilities

Now the key step. Ask Gemini to **write a script** that identifies which facilities are near flood events.

« Write a script that takes the health facilities GeoJSON and the filtered flood CSV. For each facility, check if any flood event occurred within 1km. Produce two output files: one with facilities within 1km of a flood event, one with the rest. Explain your approach before running the script. »

Objective | Steps | Output shape | Reasoning

> Step 6: add a filter to the map

The final step: an interactive filter.

« Update my Leaflet map to load both facility files (at-risk and not-at-risk). Show at-risk facilities in red and others in blue. Add a checkbox or dropdown filter that lets the user show only at-risk facilities. Publish on GitHub Pages from the main branch. Keep it simple and explain your choices. »

Push and check. You now have an interactive map that answers the question: **which health facilities are in flood risk zones?**

> Exercise: build the enriched map

Individual work — follow the six steps above

Work through each step in order. At each step:

- Read what the agent proposes before approving
- Check the output before moving on
- If something goes wrong, go back one step

The goal is the interactive map with the filter. But getting through steps 1–4 is already valuable.

> The five elements in practice

Look at what we just did:

Element	How we used it
Objective	"Which facilities are in flood risk zones?"
Steps	Filter → Map → Get facilities → Add layer → Separate → Filter
Reasoning	Each step has a clear pipeline role (Get, Clean, Verify)
Output shape	Two GeoJSON files + an interactive map with filter
Verification	Check the map at every step. "Explain your approach" before scripts.

When you give the AI all five, the result is **structured, verifiable, and documented**.

> Build data intuition before visualising

Before asking the AI to make charts or maps, **look at the data yourself** in a spreadsheet.

- Sort columns. Scroll through. What stands out?
- What are the min and max values? Do they make sense?
- How many unique values are in each column?

This is how you develop intuition: "oh, I could visualise it like this." The intuition comes from **seeing the data**, not from asking the AI what to do with it.

> When data sources disagree

You may find two sources that show **different numbers** for the same thing.

Be careful: they may have been built with **a different objective in mind**. Different collection methods, different definitions, different time periods.

They may **both be true**, even though they show a different reality. Document the discrepancy, explain which source you chose and why.

> What we covered this morning

- Finished individual map work from yesterday
- Learned the five-element prompt structure for complex AI tasks
- Built an enriched map: flood events + health facilities + interactive filter
- Practiced Get, Clean, and Verify as separate steps with clear outputs

This afternoon: we look at the FloodArchive data up close and clean it in Google Sheets.