



Geo-X training course

Hands-on 2

Please use the following citation for:

- **This exercise**

Miguel Sanchez-Lopez, Michaela Rychetska, 2026, Hands-on 2: Software Setup – Geo-X Training Course (Version 1.0.). Climate Compatible Growth. DOI: [to be assigned].

- **Geo-X software**

Sanchez-Lopez, M., Naqvi, S., Leonard, A., Schirren, L., & Hirmer, S. (2026). Geo-X: Generalised geospatial software for production of green commodities. *SoftwareX*, 34, 102712. <https://doi.org/10.1016/j.softx.2026.102712>

Learning outcomes

By the end of this exercise, you will be able to:

- Understand how the two Geo-X repositories fit together, and what each one is responsible for.
- Apply the fork → branch → commit → push workflow to manage your own Geo-X workspace.
- Install a working Geo-X environment, including the required optimisation solver.
- Verify the installation with a dry run of the Snakemake workflow.

Activity 1 – The two Geo-X repositories

Geo-X is built around two separate GitHub repositories that work together:

- **Geo-X-data-prep** builds the spatial input Geo-X needs: it turns raw GIS layers (land cover, slope, water bodies, roads, etc.) into a set of H3 hexagons covering the country of interest.



- **Geo-X** is the main repository. It takes that hexagon file, together with techno-economic parameters and a configuration file, and runs the commodity production and transport cost optimisation.

Note: In this class, we will only set up Geo-X. Geo-X-data-prep, and the spatial data it produces, is covered in Class 3.

Activity 2 – Fork and clone Geo-X

Before installing anything, it is worth being clear on three terms we will use throughout the course:

- **Upstream repository:** the original code, maintained by CCG at github.com/ClimateCompatibleGrowth/Geo-X.
- **Fork:** your own online copy of that repository, under your GitHub account. Changes here do not affect the upstream repository.
- **Clone:** the local copy of your fork, downloaded onto your machine, where you will actually do the work.

Try it: let's set up your own copy of Geo-X.

- Go to <https://github.com/ClimateCompatibleGrowth/Geo-X> and click 'Fork' (top-right of the page). GitHub creates your own copy at <https://github.com/your-username/Geo-X>.
- Clone your fork to your machine: `git clone https://github.com/your-username/Geo-X.git`

Note: Use your fork URL when cloning, not the original CCG repository – this keeps your changes independent and traceable.

Activity 3 – Set up the environment and install a solver

Geo-X's dependencies are managed through a conda/mamba environment, and the optimisation itself requires a solver compatible with PyPSA.

Try it:



- From the root of your cloned Geo-X folder, create the environment: `mamba env create -f environment.yaml`
- Activate it: `mamba activate geox`. Remember to reactivate this every time you open a new terminal session.
- Install a solver. For this course we recommend Cbc, which is free and open source: `mamba install -c conda-forge coincbc`. Gurobi is also supported (commercial, with a free academic licence) – only install it if you already hold a licence.

Note: Geo-X-data-prep uses two further environments of its own (prep and spider). You do not need these yet – they are covered in Class 3.

Activity 4 – Verify your installation

A full optimisation run can take anywhere from several minutes to several hours, so we will not run one now. Instead, we will use Snakemake’s dry-run mode, which resolves the full chain of rules – checking that your repository, environment and solver are correctly linked – without actually executing any computation.

Try it:

- Confirm Snakemake is available: `snakemake --version`
- From the root of the Geo-X folder, run a dry run of the full optimisation workflow: `snakemake -j 1 -n optimise_all`
- Check the output: Snakemake should list the chain of rules it would run (e.g. `get_weather_data`, `optimize_transport`, `calculate_water_costs`, `optimize_plant`, `calculate_total_cost`) with no errors such as `MissingInputException`.

Note: The `-n` flag means ‘dry run’: nothing is actually computed. This is enough to confirm your setup is correct. We will run the workflow for real from Class 4 onwards.

Activity 5 – Start your own workspace with Git

From Class 3 onwards, we will build a case study for Ghana – one of the countries where CCG carries out analysis – covering both hydrogen and ammonia production. Rather than



working directly on the main branch, you will keep this work in its own branch, following the same fork → branch → commit → push habit you will use for the rest of the course.

Try it:

- Create a new branch for your case study: `git checkout -b ghana-h2-nh3`
- Make a small first commit, just to confirm the workflow: for example, add a short note to a text file describing what you plan to build: `git add .` then `git commit -m "Start Ghana H2/NH3 workspace"`
- Push your branch to your fork: `git push origin ghana-h2-nh3`

Note: Before every exercise from here on, check: Am I working in my fork? (the repo URL should show your username, not ClimateCompatibleGrowth). Am I on the right branch? (`git branch`). Did I commit my last change before moving on? (`git status`).

Wrap-up

You have completed this exercise. Before moving on, check that:

- You have forked and cloned Geo-X.
- Your geox environment is active, with a solver installed.
- Your dry run of `optimise_all` completed without errors.
- Your `ghana-h2-nh3` branch has been created, committed to, and pushed to your fork.

Well done! In Class 3, you will prepare the spatial and techno-economic data Geo-X needs. Using this Ghana branch as your working space.