GenAi usage guidelines – Lila Wagenaar

GenAi tools used

During the course of Complex system simulation, I used Microsoft Copilot and ChatGPT, especially during the development of our project. I used these tools mainly for:

- Deepening my understanding of some of the concepts learned in class, as well as concepts related to our project (Scalon paper about vegetation cluster)
- Helping me understand heavy scientific papers
- As my background for coding is very light, I used to give me the main structure of codes, and help me understand each line, and the different parameters' role in the codes
- Understanding Git workflows, and LaTex Beamer
- Giving me overall “backbone” of the project by pointing the direction to follow, the structure, and good practice for this kind of project (readMEe files, all the requirements from the slides, …)
- Improving code readability (structure, docstrings)
- Debug and understand the output of my codes

Prompting strategy

- Context prompting: To use these tools, I first always provide a lot of contexts (what is expected/required, for which projects, code snippets, plot descriptions…) and always use the same conversation, so I only have to do this once.
- Goal: Then I explain what I’m not understanding (concepts, output of the code), or what I would like to have instead, … just basically asking what I need
- Reprompting if not happy with the answer: if the tool did not really answer my request, I’ll add even more context and explain why it’s not what I meant, or what I need. I try to be more specific for it to understand.
- Critical evaluation: Before adopting the answer, I always think about the logic behind it, and compare it against my course or my own knowledge. If I think it makes sense, then I’ll keep it.

Specific prompts provided

After pasting the project's requirements, codes, or notes, my prompts were :

- “Do you think this is a good project proposal? What could we skip or pay more attention to?”
- “Help me start the implementation of a 2D CA model without giving me the full code. What should the structure look like.” The same for percolation
- “I’ll past an academic paper, please explain the main concepts and important vocabulary as if I have no prior background in the topic. Focus on the intuition and purpose for me to understand more easily.”
- “Is this code correct if I my goald is to add a parameter that adds a weight to the local rule?”

- “Does mixing the local and global rules like this make sense, and how do I ensure probabilities stay between 0 and 1?”
- “What extensions could we add to our model (simulations, analysis, plots) to answer the research questions more completely”?
- “Why is this code not working?”
- “What should a README file contain? What else could I add?”
- “From this animation, what should I change to generate multiple CA evolutions side by side at the same time?”
- “Why does our model do not percolate when the rainfall value is higher than the percolation threshold?” “Why are my plots not straight lines when I plot the inverse cumulative distribution?” “And why are we using an inverse cumulative distribution instead of a normal distribution?”
- “This is pretty much what our presentation is about/what I’m going to say, are there things that are incorrect? Things to add?”
- “What does this error message mean? What should I do to fix it?”

Output generated

Depending on the prompts I used, it would generate either

- Some explanatory paragraphs about concepts, model behaviour, …
- Debugging explanations for code, Github or LaTex Beamer errors
- Main structure of code with explanation of different functions

What was adopted, modified, or discarded

From the prompts I generated, I think I adopted the majority of them, whether it was coding, explanations, … but I was always checking if it was producing what I asked for, and if a line of code didn’t make sense in my opinion, I would remove it and try it for myself.

For explanations, I understood the idea that the AI was explaining to me, but rewrote things with my words to make sure I understood completely. I also modified some code feedback to match it more with the results I wanted.

I discarded some suggestions for the projects that I thought were not really in the scope of the projects, or were a bit out of reach for my coding skills (different kinds of analysis, or complicated plots).

Observation of effectiveness and limitations

In my opinion, GenAi is a very good tool for me to understand concepts differently. I think its ability to explain by giving a lot of examples, and simplifying, was very useful to understand the core ideas of many topics. I also think it’s a great tool to debug coding issues, and saved me a lot of time.

However, the tool is made to be enjoyable, and does not necessarily contradict the user, even when needed. I think it’s important to phrase your needs in a way that is not biased, for the tool to be more objective.

When generating some code, it’s also important to understand the different steps to ensure that the final output will correspond to what was initially asked. And when interpreting or clarifying results, the user should take a step back to think if the explanation actually makes sense.

Finally, even if I was explaining the context quite deeply, the tool cannot know everything that’s happening to my computer. For example, when fixing issues with GitHub merging issues, I sometimes shouldn’t have executed what he told me right away, and try to understand the error messages' real meaning.