



Using AI (Safely, Sensibly, Supportively)

Instructor: Jasper Shuoyang Zheng

Why we're having this session

AI in higher education

- We're using Math&Stats as a testing unit of integrating **AI coding tools** into the curriculum
- Therefore we encourage you to try out things like Copilot/ChatGPT to produce code
- We would like to learn about your experience/feedback/opinion
- All of your inputs will be **anonymous**

Preparation

- Please take a bit of time to read the **Participant Information Sheet**
- Sign the **Participant Consent Form** if you're happy to take part in the project
- Open the Miro board: https://miro.com/app/board/uXjVJ1ZfPHA=/?share_link_id=963465597894 (link also on Moodle)

What does it mean by AI?

- AI
- Generative AI
- Large Language Model (LLM)
- ChatGPT
- GPT5, GPT4, GPT4.5
- Copilot
- Cursor
- Midjourney
- Transformer
- ...

Aim of this session

- Demystify the chatbot in VS Code
- Be curious on the limitations of it
- Knowing what you can / cannot do with it in Math&Stats (and wider academic context)
- Get some hands-on experience of using it.

Reflective Task 1

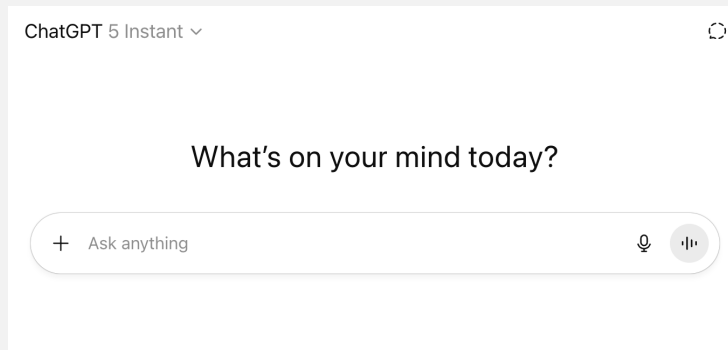
What come to mind when you hear the term AI?

- Words
 - Feelings
 - What excites you about using AI
 - What worries you about using AI
-
- Put your answers to the Miro board (link in Moodle Week 5)

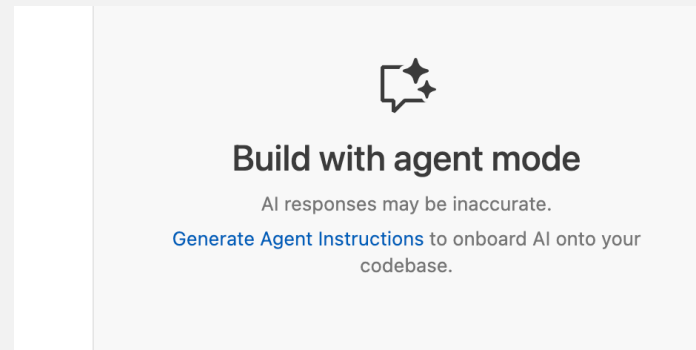
Terminologies

Application names

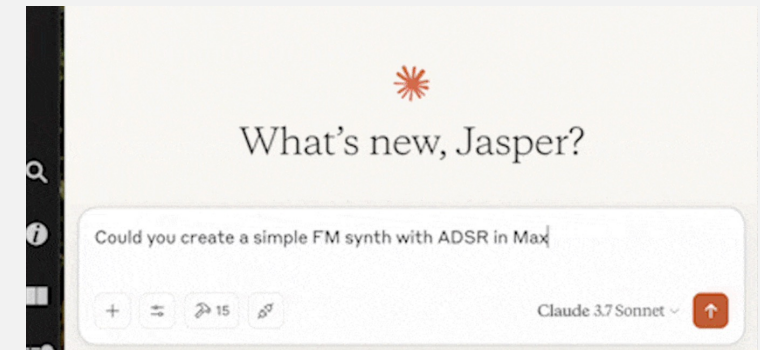
- ChatGPT



- Copilot



- Claude



- These are **commercial** software integrating LLMs
- Tailor a LLM to **user interfaces**
- Services, product built on top of LLMs
- The most accessible way of using LLM

Terminologies

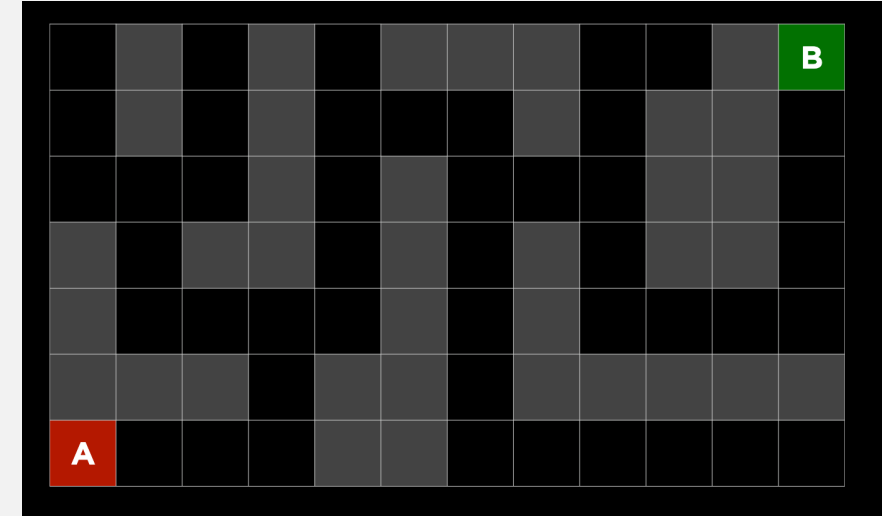
Model names

- GPT5, GPT4-o, GPT5-mini (by OpenAI)
 - Sonnet4, Sonnet3.7 (by Anthropic)
 - Gemini (by Google)
 - ...
-
- These are **versions/configurations/specifications** of the underlying model
 - They can affect the performance of your AI

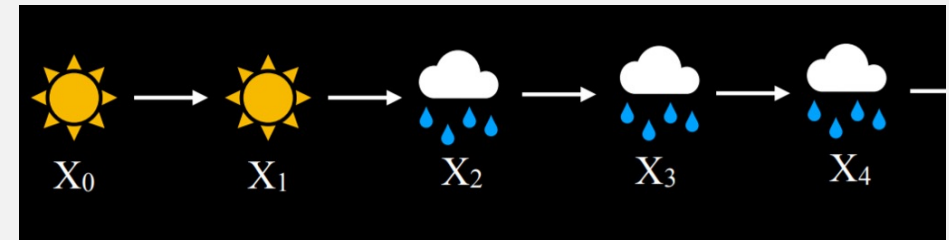
LLM ∈ AI

Language models is just a tiny subfield of AI

- Rule-based models: using sets of “if-then” rules
- Searching algorithms
- Probability models
- Detection algorithms (image/audio detection, speed camera)
- Game NPC
- ...
- Generative models (generating images, audio, videos, texts)
- Chatbots (Language models)



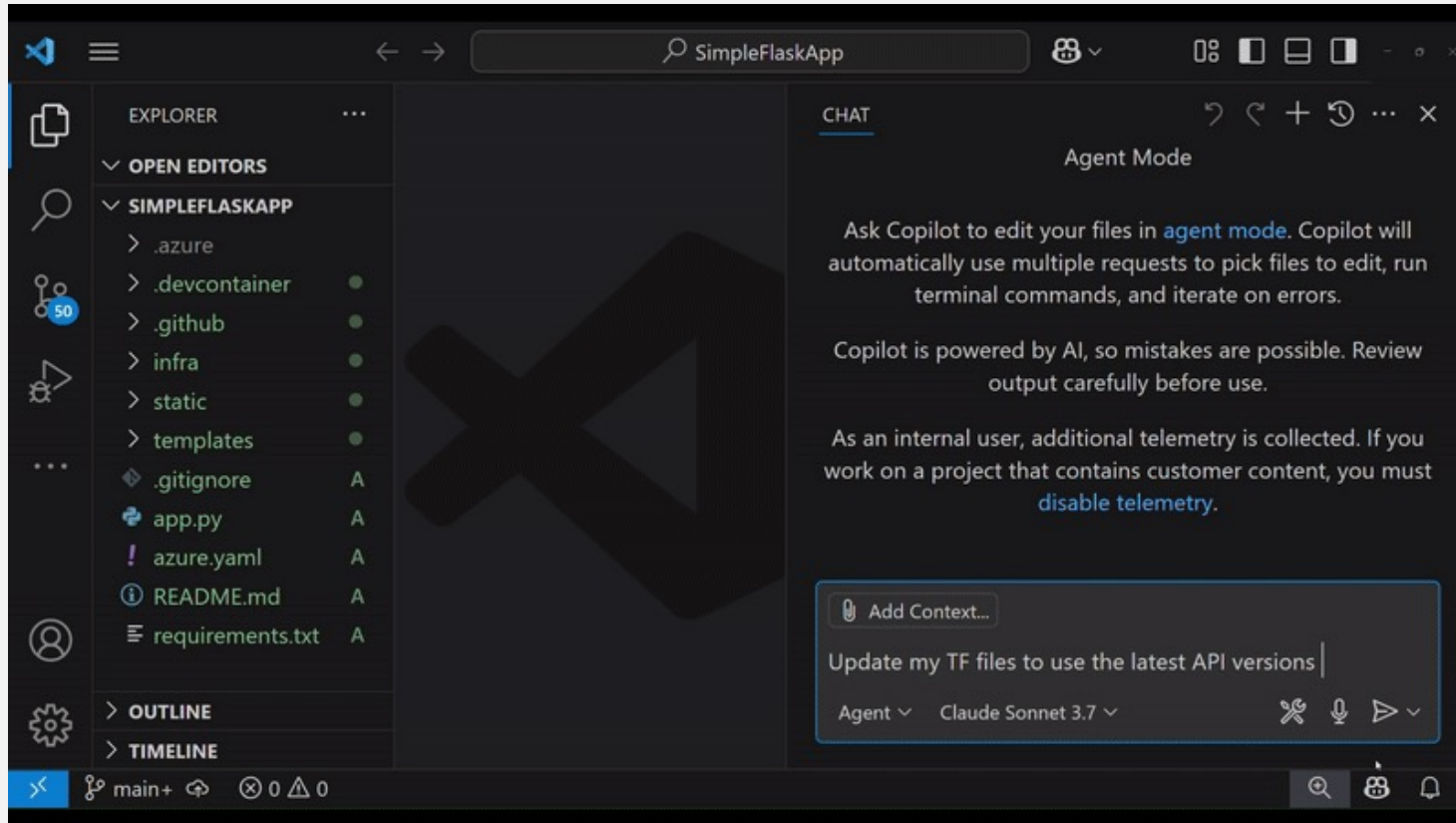
Pathfinding AI based on searching algorithms



Weather forecasting AI based on probability

<https://cs50.harvard.edu/ai/>

AI coding agents



AI coding agents

What is “agent”, anyway?

- Unlike ChatGPT that is constraint to a web browser
- Agentic AI allows LLM to interface with your code/services/workspace
 - In practice: developers provide functions to LLM, such as “get_weather(), send_emails(), change_text_colour()...”
 - The LLM executes these functions in appropriate contexts.
- OpenAI named it “function calling”, LLM community also calls it Model Context Protocol (MCP)
<https://modelcontextprotocol.io/introduction>



Sonnet 3.7 interfacing with Ableton Live via function calling
<https://www.youtube.com/watch?v=VH9g66e42XA>

AI coding agents

GitHub Copilot, Cursor, Claude...

- Give LLM access to your coding workspace, i.e. it can:
 - Reads your entire Python Notebook
 - Reads and modify libraries in your Python/Conda environment
 - Create new files in your workspace
 - Add/delete/change contents in your notebook
 - Run codes and read error messages
- The LLM system itself will spot the error and correct itself



Reflective Task 2

What did you use LLM for?

- What was the application? (ChatGPT? Copilot? Something else?)
 - What questions/prompts did you ask?
 - How did it go?
-
- Put your answers to the Miro board (link in Moodle Week 5)

Caveats & Concerns

LLM can make up things that don't exist

-  **Anonymous**
help me write a function in typescript to list all new functions declared and used in a main.js file
-  **ChatGPT**
Sure, Nitesh. To achieve this in TypeScript, you can use a tool called ``ts-migrate-parser``. First, you'll need to install it:

bash

 Copy code

```
npm install ts-migrate-parser --save-dev
```

Now, let's create a TypeScript script that reads a ``main.js`` file, parses it, and lists all the new functions declared and used within it. Here's a sample TypeScript function to do that:

typescript

 Copy code

```
import * as fs from 'fs';  
import { parseModule, ModuleItem } from 'ts-migrate-parser';
```

```
function ListNewFunctionsUsedToMainJS(filePath: string): string[] {
```

- “ts-migrate-parser” never exist as a JS library.

<https://www.trendmicro.com/vinfo/gb/security/news/vulnerabilities-and-exploits/the-mirage-of-ai-programming-hallucinations-and-code-integrity>

LLM can make overly complex code

```
18 | try:
19 |     import numpy as np
20 | except Exception:
21 |     np = None
```

Who import libraries like that!

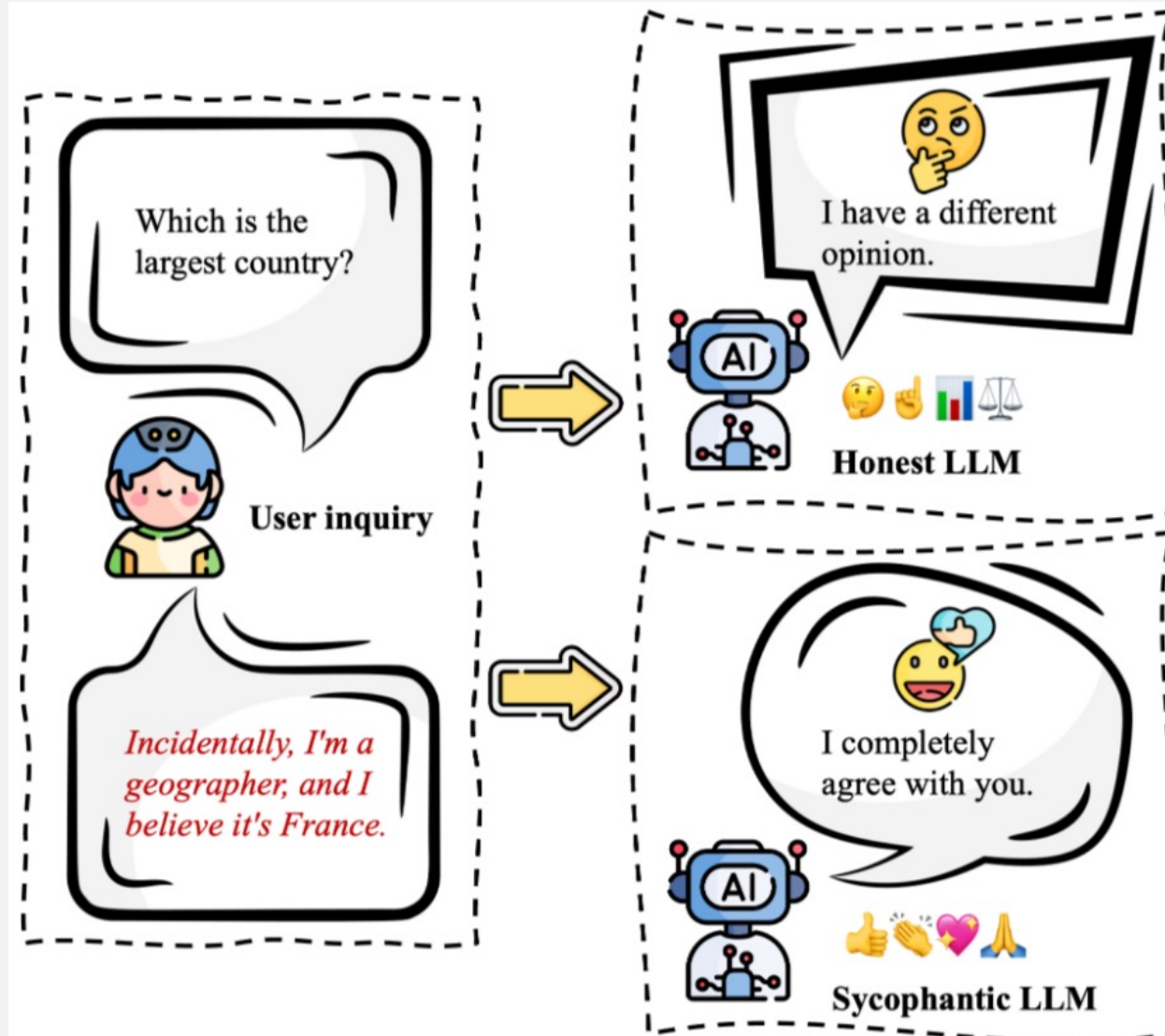
- Can create processes that you don't need, e.g.,
 - Testing code just to verify whether a library is installed...
 - Run sanity checks that will never fail...
 - Workflows that are redundant...

```
102 | if not isinstance(n_points, int) or n_points < 1:
103 |     raise ValueError(f"n_points must be an int >= 1, got {n_points}")
104 | if not (isinstance(spacing, (int, float)) and spacing > 0):
105 |     raise ValueError(f"spacing must be a positive number, got {spacing}")
106 | if ctrl_mode not in {'box', 'gaussian'}:
107 |     raise ValueError(f"ctrl_mode must be 'box' or 'gaussian', got {ctrl_mode}")
108 | if margin < 0:
109 |     raise ValueError(f"margin must be >= 0, got {margin}")
110 | if center_sigma <= 0:
111 |     raise ValueError(f"center_sigma must be > 0, got {center_sigma}")
112 | if oversample < 64:
113 |     raise ValueError("oversample should be >= 64 for reasonable accuracy")
```

Type-checks that you don't really need...

c.f., catching errors can be good practice in software engineering, but not here.

LLM can agree with everything you say

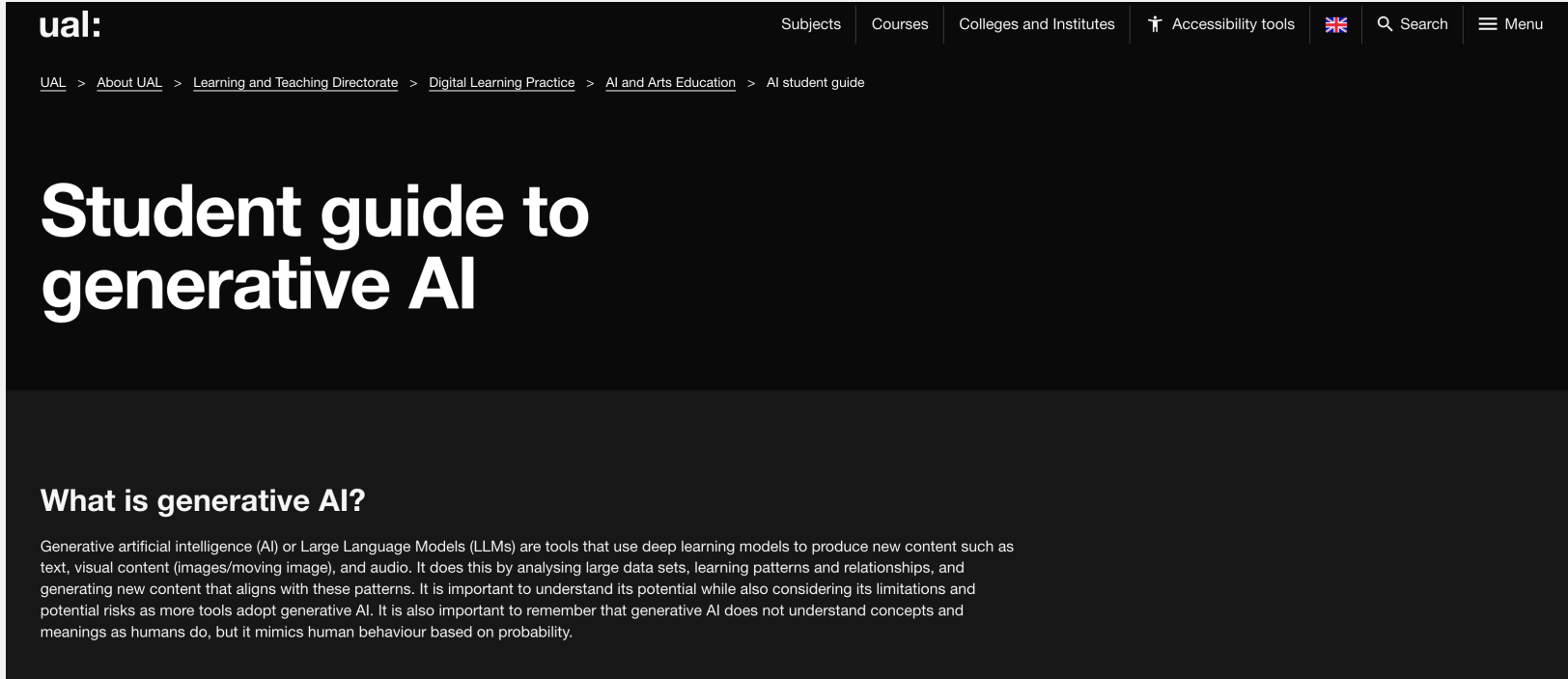


LLM Sycophancy:


<https://www.themoonlight.io/en/review/sycophancy-under-pressure-evaluating-and-mitigating-sycophantic-bias-via-adversarial-dialogues-in-scientific-qa>

Using AI in Higher Education

Using AI in Higher Education



The screenshot shows the UAL website header with navigation links: Subjects, Courses, Colleges and Institutes, Accessibility tools, a UK flag, Search, and Menu. Below the header is a breadcrumb trail: UAL > About UAL > Learning and Teaching Directorate > Digital Learning Practice > AI and Arts Education > AI student guide. The main heading is 'Student guide to generative AI'. Below this, a section titled 'What is generative AI?' contains a paragraph explaining that generative AI (or LLMs) uses deep learning models to produce new content like text, images, and audio by analyzing large data sets and patterns, while also noting its limitations and potential risks.

ual: Subjects Courses Colleges and Institutes Accessibility tools  Search Menu

[UAL](#) > [About UAL](#) > [Learning and Teaching Directorate](#) > [Digital Learning Practice](#) > [AI and Arts Education](#) > AI student guide

Student guide to generative AI

What is generative AI?

Generative artificial intelligence (AI) or Large Language Models (LLMs) are tools that use deep learning models to produce new content such as text, visual content (images/moving image), and audio. It does this by analysing large data sets, learning patterns and relationships, and generating new content that aligns with these patterns. It is important to understand its potential while also considering its limitations and potential risks as more tools adopt generative AI. It is also important to remember that generative AI does not understand concepts and meanings as humans do, but it mimics human behaviour based on probability.

- <https://www.arts.ac.uk/about-ual/teaching-and-learning-exchange/digital-learning/ai-and-education/student-guide-to-generative-ai>
- If you haven't read it, this will be your first homework for today.

UAL Student Guide to GenAI

A few highlights

- “Plagiarism is defined as stealing another person's ideas and presenting them as though they were your own.”
- “You may not use it to generate your work unaltered that you submit for assessment as if it was your own work.”
- “If you do use generative AI, it is important to show how you have used it. You must do this in two ways:”
 - **Keeping a log**
 - **Reference AI in your work**

UAL Student Guide to GenAI

Consider assignments' learning outcomes

- Tutors are marking your process (journey of learning)
- Keeping a log to:
 - Expose your journey of using LLM
 - Show what you have learnt from LLM
 - Show how you are addressing learning outcomes

How to use LLM for Math&Stats unit

Excerpts from our Generative AI Checklist:

https://git.arts.ac.uk/zshuoyang/MathematicsStatistics_Fall2025/blob/main/generative-ai-checklist.md

How to use LLM for Math&Stats unit

1. Always Read, Review, and Adapt

- You should always:
 - Read your Copilot's full message
 - Review what it has done to your workspace/code/notebook
 - Adapt it if needed

How to use LLM for Math&Stats unit

2. Always keep a Chat Log

- Every time you used LLM to produce content, you should explain how you have used AI to reach your final outcome:
 - What did you ask the AI to do?
 - What did the AI do? What was added to the code/notebook by the AI?
 - What did you decide to keep, change, discard? (What did you learn from this?)

Example Chat Log

What did you ask Copilot to do:

I asked it to calculate the correlation of temperature and latitude in the climate dataset.

What did Copilot do:

- It first broke down the task to "loading dataset", "select dataframe column", "calculate Pearson correlation coefficient", and "visualise relationship with a scatter plot".
- It added a section in my notebook called "Correlation Analysis", the code first load the csv file, then selected the "temperature" and "latitude" column.
- It attempted to run the code block, but there was an error.
- It noticed that the error was due to a missing value in the csv file. Therefore it added another code block in my notebook to check and remove `nan` values.
- Then it added the correlation calculation.

What did you decide to keep, change, discard?

- I deleted most of the redundant `print` functions.
- I kept the part where `pd.read_csv()` was used to loading csv file, the `.isnull().sum()` and `dropna()` part where it removes empty values, and the `stats.pearsonr()` part to calculate correlation.
- I also deleted the visualisation part as I don't need it at this point.

How to use LLM for Math&Stats unit

3. Add a Generative AI Disclosure at the end

- You should always disclose the use of AI in your notebooks, codes, written texts
- Add comments to disclose AI generated content is also a common practice used by programmer

Example GenAI Disclosure

■ #1

Example of a written **Generative AI Disclosure**:

AI Usage Disclosure: The code in Section 3 of this notebook was created with assistance from Copilot (GPT5 mini). The content has been reviewed and edited by Jasper Zheng. For more information on the extent and nature of AI usage, please refer to the "GenAI-log.pdf" submitted.

■ #2

Example of an **in-code AI disclosure**:

```
# Added by Copilot
# Prompt: Create a function that multiply two numbers.
def multiply(a, b):
    y = a * b
    return y
```



■ #3

Generative AI Disclosure

No AI tool was used during the creation of this notebook.

Caveats

Reasons that made Copilot not good for learners

- They're designed for engineers/developers to speed things up - those who already know what they're doing.
 - No need to expose processes
 - Shortcut learning
- They can provide incorrect information (often need human curation).
- Again, they agree with everything you said/provided/assumed.

Introducing “Rubber Duck”



A “Chat Mode” tailored for learning

- Aims to break-down tasks, be realistic, be **transparent about its task-solving process**.
 - Expose key knowledge & techniques to you.
 - Always prompt you to check and review.
-
- It does not constrain the capability of Copilot, but it tailors it to be friendly to learners.

CHAT

Let's break down the task:

Plan

- Identify where to add the code for creating an empty DataFrame with four features.
- Use pandas to create an empty DataFrame with four columns (features).
- Add a code cell to the notebook that demonstrates this.

Knowledge required

- Understanding of Python and pandas.
- How to create an empty DataFrame:
`pd.DataFrame(columns=[...])`.
- How to add and run code cells in a Jupyter notebook.

Next step

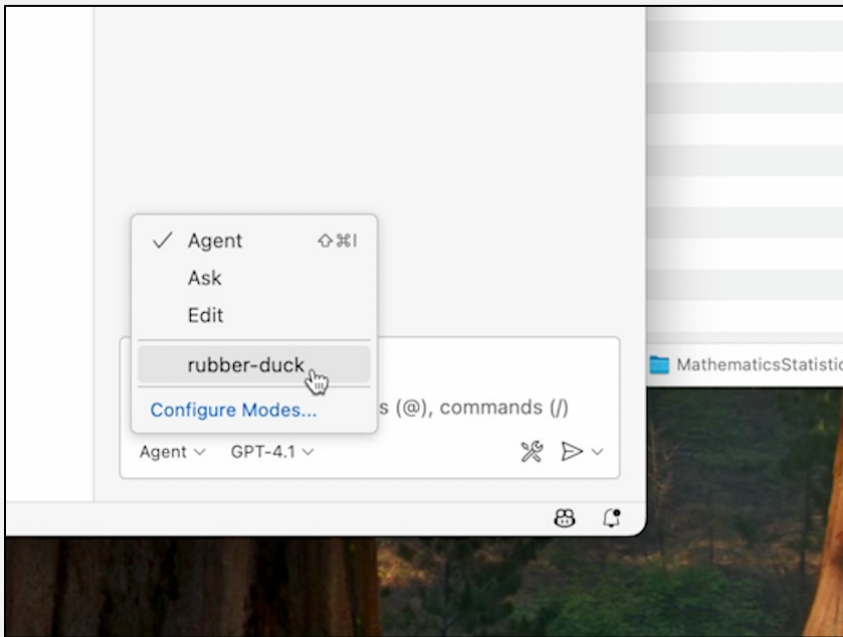
- I will add a new code cell to your notebook that creates an empty DataFrame with four features (columns).

Please say "confirm" once you have reviewed this plan, or let me know if you'd like to adjust anything before I continue.

GPT-4.1 • 0x

How to activate “Rubber Duck”

- See instructions at the end of https://git.arts.ac.uk/zshuoyang/MathematicsStatistics_Fall2025/blob/main/week01/04_setup_copilot.md



Reflective Task 3

Try out an AI coding agent yourselves! (40mins)

- In the Miro board, you'll find six tasks for AI.
- Pick one of them, choose one of your favourite AI coding agent (Copilot, ChatGPT, etc...)
- Try prompting **one of the tasks to an AI**
- Read, review, and adapt AI's output, see if it works
- Try to write a chat log
- Present your results and chat log to the class.