Scottish Graduate School of Social Science

Sgoil Cheumnaichean Saidheans Sòisealta na h-Alba

Collecting Digital Data

LLMs as Coding Assistants

Dr Diarmuid McDonnell

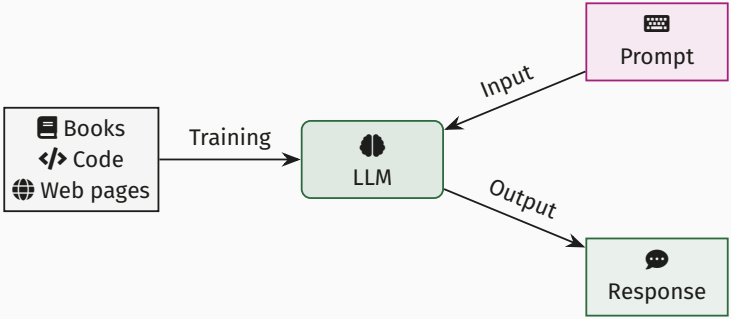
Braw Data Ltd · Gradel Institute of Charity, University of Oxford

24 February 2026

LLMs as Coding Assistants

What Are LLMs?

A **Large Language Model** is a statistical model trained on vast amounts of text and code.



You provide a **prompt** (a question or instruction), and the model generates a **response**.

Paid vs Open Models

Paid (ChatGPT, Claude)	Open (Llama, Mistral)
Hosted by provider	Run locally or self-host
Subscription / API fees	Free to use
Often more capable	Improving rapidly
Data sent to provider	Data stays with you
Easy to use (chat UI)	Requires some setup

Tip

For this course, any chat-based LLM will work — use whichever you have access to.

How to Prompt for Code

Good prompt

"Write a Python script using `requests` to download stop-and-search data from the UK Police API for the City of London force. Save the results as a JSON file."

Bad prompt

"Get me some police data."

What makes a good prompt?

- Specify the **programming language** and **libraries**
- Provide the **exact URL** or API endpoint
- State the **desired output** format
- Include relevant **parameters** (e.g., force name, date range)

What LLMs Are Good At

- **Boilerplate code** — standard request/response patterns, file I/O, data wrangling
- **Syntax and library usage** — "How do I use `pandas.json_normalize`?"
- **Explaining error messages** — paste a traceback and ask what went wrong
- **Converting between languages** — turn a Python script into R (or vice versa)
- **Generating documentation** — docstrings, comments, README files

Key insight

LLMs excel at tasks that are **well-documented** and **widely practised** online.

What LLMs Are Bad At

- **Up-to-date API endpoints/parameters** — training data has a cut-off date
- **Complex multi-step logic** — long workflows with many dependencies
- **Understanding your specific research context** — it doesn't know your data
- **Guaranteeing correctness** — output looks plausible but may be wrong
- **Hallucinated function names** — confidently invents functions that don't exist

Remember

LLMs are **confident**, not **correct**. Always verify their output.

Critical Evaluation of LLM-Generated Code

Golden rule

You **must** understand the code yourself. If you can't explain what it does, don't use it.

- **Always test** LLM-generated code — run it, inspect the output
- **Check API docs** match what the LLM suggests — endpoints, parameters, authentication
- **Be wary of hallucinated libraries/functions** — verify they actually exist
- **Consider reproducibility** — LLM outputs vary between sessions and models
- **Security risks** — never paste API keys, passwords, or sensitive data into a chat

Practical 4: LLM Showdown

- Compare different LLMs on the same coding task
- Evaluate the quality, correctness, and usability of generated code
- Apply the critical evaluation criteria from this lecture

Use your preferred LLM (ChatGPT, Claude, Llama, etc.)
to complete the challenge tasks.

Thank You!

Dr Diarmuid McDonnell

Director, Braw Data Ltd

Visiting Fellow, Gradel Institute of Charity, University of Oxford

www.brawdata.co.uk Gradel Institute of Charity Course materials on GitHub

Scottish Graduate School of Social Science

Sgoil Cheumnaichean Saidheans Sòisealta na h-Alba

Thank you for attending the **SGSSS Collecting Digital Data** course.

Good luck with your research!