# Generative AI Tools for Quantitative Research

## Worksheet

See full resource: <https://www.ncrm.ac.uk/resources/online/all/?id=20859>

### How to use LLMs in VS Code (Stata or R coding)

**Step 1**. Install an extension to enable VS Code to understand either .r (R) or .do (stata) script files. VS code natively understands other languages (e.g., python, html). In video 1, the [R extension](https://marketplace.visualstudio.com/items?itemName=REditorSupport.r) and [Stata MCP extension](https://github.com/hanlulong/stata-mcp) was used. Once installed, VS Code should recognise the scripts and render the contents (e.g., comments will appear in green).

**Step 2.** Setup Copilot within VS Code. Note that [Copilot Pro](https://docs.github.com/en/copilot/how-tos/manage-your-account/getting-free-access-to-copilot-pro-as-a-student-teacher-or-maintainer) is currently freely offered to those in the education sector.

**Step 3.** Use Copilot for code suggestions such as autocomplete when typicing, or writing/inserting code.

Note: [CoPilot](https://docs.posit.co/ide/user/ide/guide/tools/copilot.html) is also available within RStudio for autocomplete, and the [chattr](https://mlverse.github.io/chattr/) package enables a chat interface.

### How to use LLMs locally (ollama):

If you want to use a stand-alone app with a chat interface, and link with your IDE, use [ollama](https://ollama.com/library) (Steps below).

* Software such as [LM Studio](https://lmstudio.ai/), [AI Navigator](https://www.anaconda.com/download) are also other straightforward stand-alone applications which offer similar features.

Installing and using Ollama (stand-alone chatbot interface and connected to IDE - VS Code).

**Step 1.** Download and install [ollama](https://ollama.com/library).

**Step 2.** Select which LLM you would like to download from [ollama’s library](https://ollama.com/library).

**Step 3.** Download the LLM using the

* Ollama app
* Or via terminal (Mac) or cmd (windows) e.g., the following command will download the Deepseek-r1 8 billion paramater LLM (a deepseek LLM 5.2gb in size): ollama run deepseek-r1:8b

**Step 4.** Use the LLM via chat interfaces, either:

1. Using the Ollama app
2. inside the terminal / cmd (after typing `ollama run deepseek-r1:8b’)
3. via a plugin—improved UI versus the terminal e.g., [Page Assist](https://chromewebstore.google.com/detail/page-assist-a-web-ui-for/jfgfiigpkhlkbnfnbobbkinehhfdhndo?hl=en)
4. via IDEs such as VS Code (in the copilot panel select Manage models…. Ollama… then the local LLM of choice).

Note that you can also download the latest models from huggingface (e.g., ollama run [hf.co/](http://hf.co/path)unsloth/DeepSeek-R1-GGUF will download this model: <https://huggingface.co/unsloth/DeepSeek-R1-GGUF>)

### How to use LLMs in Positron

[Positron](https://positron.posit.co/) is an IDE developed by Posit, the creators of RStudio, and is built off VS Code. It is particularly well suited to using R. At the time of writing, Positron has support for two LLM integrations: Anthropic’s Claude, accessible via a chat interface, and GitHub Copilot for inline code completions. GitHub Copilot Pro is available for individuals in the education sector (teachers and students) while Claude requires an API key and is pay-per-use, separate to Anthropic’s more usual monthly subscription model. The API is cheap to use, and prices have gone down over time (interactions with Claude in Video 4 cost less than 20 pence). The Claude integration can be accessed by clicking the chat robot icon in the sidebar. GitHub Copilot line completions are automatic, once set up.

Claude is especially well integrated into Positron workflows. Information from a session (e.g., console output, plots, and metadata from objects loaded into memory) is automatically sent to Claude as context and users are able to add files or folders (e.g., open scripts) as further context to prompts. Claude can be operated in Ask, Edit or Agent modes. Ask mode is self-explanatory: you ask a question and get an answer. In Edit mode, Claude can directly add or change code in open scripts, while in Agent mode, Claude can directly run the code and read the generated outputs. Hovering over code in Claude’s output, buttons appear that can be clicked to run, copy, insert, or add the code to a new file.

Given some data is sent to Claude along with prompts, care must be taken ensure sensitive information is not sent. More detail on LLM integrations into Positron is provided on the [Positron website](https://positron.posit.co/assistant.html). Note, there are also a number of R packages (e.g., [ellmer](https://ellmer.tidyverse.org/)) that enable use of LLMs, though there are typically not directly integrated into the IDE and are instead accessed via R function calls. This can be useful for, e.g., performing analyses where the LLM generates data to be analysed (for example, classifying text).

### Creating visualisation tools (‘vibe coding’)

LLMs now let researchers build interactive apps and visualisers simply by describing what they want—a conversational workflow known as [vibe coding](https://en.wikipedia.org/wiki/Vibe_coding). The term captures a fast, intuitive development process in which the user proposes an idea and the LLM generates or adjusts the underlying code.

Take a visualisation of selection‑bias in survey research. A first prompt such as

“Create an HTML visualiser that illustrates selection bias” gives the model freedom to choose its own design. A tighter prompt—for example, “Build an HTML page that plots a scatter‑plot, adds a slider to remove data points, and overlays two trend lines (unbiased vs biased estimate)”—should yield a more predictable output. Iterating with follow‑up prompts (“limit the slider to 0–100 %”, “colour the biased line red”, “show tooltips with sample size”) steadily refines the tool. See [this](https://dbann.github.io/selection/) end product, which was created in this way.

LLMs sometimes hallucinate, make mistakes, or add unwanted features. Good “vibe coders” tackle these glitches by switching models, tightening prompts, or keeping the code base small and modular. Frontier systems with canvas previewscan speed up debugging by showing live renders, though these may mis‑compile.

Possible visualisation tools researchers can build with LLMs:

* Interactive teaching aids (e.g., [hand‑tracking demos](https://dbann.github.io/regression_visualizer/) of regression).
* Public‑engagement widgets that let users explore policy scenarios.
* Meta‑research dashboards such as calculators exposing the [hidden cost of peer review](https://dbann.github.io/grantreview/).

Entire static sites can be scaffolded the same way—try prompting a frontier LLM to *“mock‑up a website on quantitative research”* and iterate from there.

Vibe coding guides: <https://www.youtube.com/watch?v=BJjsfNO5JTo>

National Centre for Research Methods (NCRM)
Social Sciences
Murray Building (Bldg 58)
University of Southampton
Southampton SO17 1BJ
United Kingdom

**Web** www.ncrm.ac.uk
**Email** info@ncrm.ac.uk
**Tel** +44 23 8059 4539
**Twitter** @NCRMUK