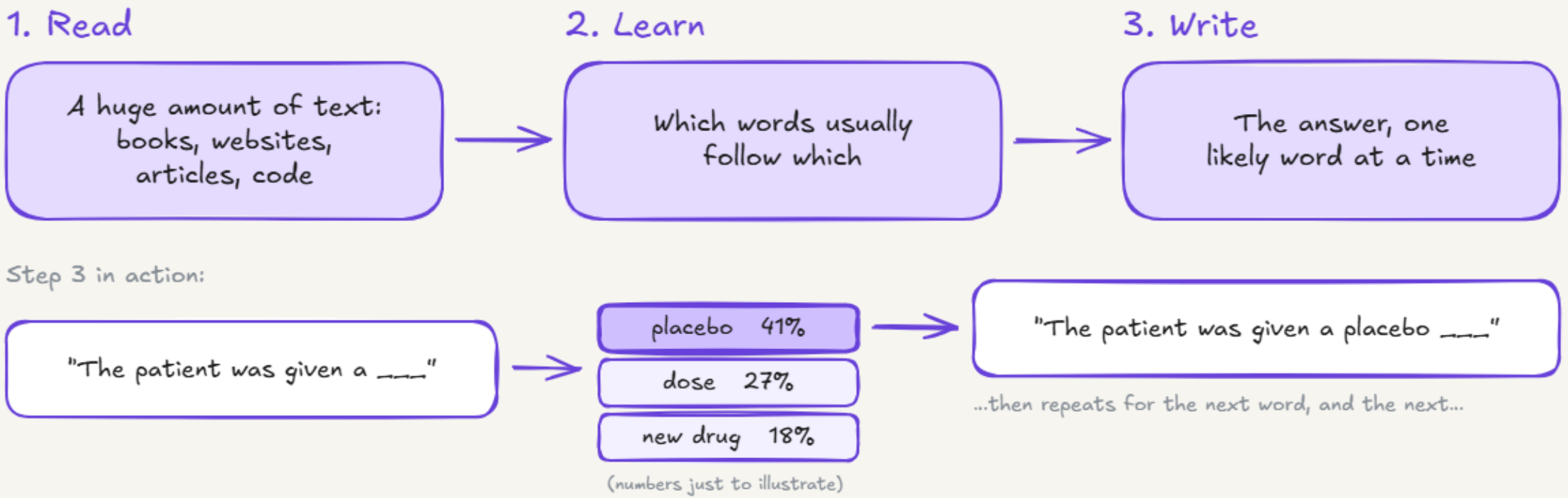
LLMs in Research (using R)

What is LLM?


1. Read
A huge amount of text: books, websites, articles, code
2. Learn
Which words usually follow which
3. Write
The answer, one likely word at a time
Step 3 in action:
"The patient was given a ___"
placebo 41%
dose 27%
new drug 18%
(numbers just to illustrate)
"The patient was given a placebo ___"
...then repeats for the next word, and the next...
Why it can help with R:
it has read millions of examples of R code
It makes very good guesses, not guaranteed facts
we always check its work

Four ways to put an LLM next to R

Web chat

Examples: ChatGPT, Claude, Gemini, Copilot

Sees: only what you paste

Good for: concepts, error messages, a second opinion

IDE assistant

Examples: Posit Assistant (RStudio, Positron), GitHub Copilot

Sees: open files; with Posit, your live R session

Best for: inline edits, explaining code in place

Coding agent

Examples: Claude Code, OpenAI Codex, Google Antigravity

Sees: the whole project folder, and it runs commands

Best for: refactors, tests, packages, Shiny apps

R packages

Examples: ellmer, btw, mcptools

Sees: exactly what your code sends it

Best for: LLM steps inside a reproducible pipeline

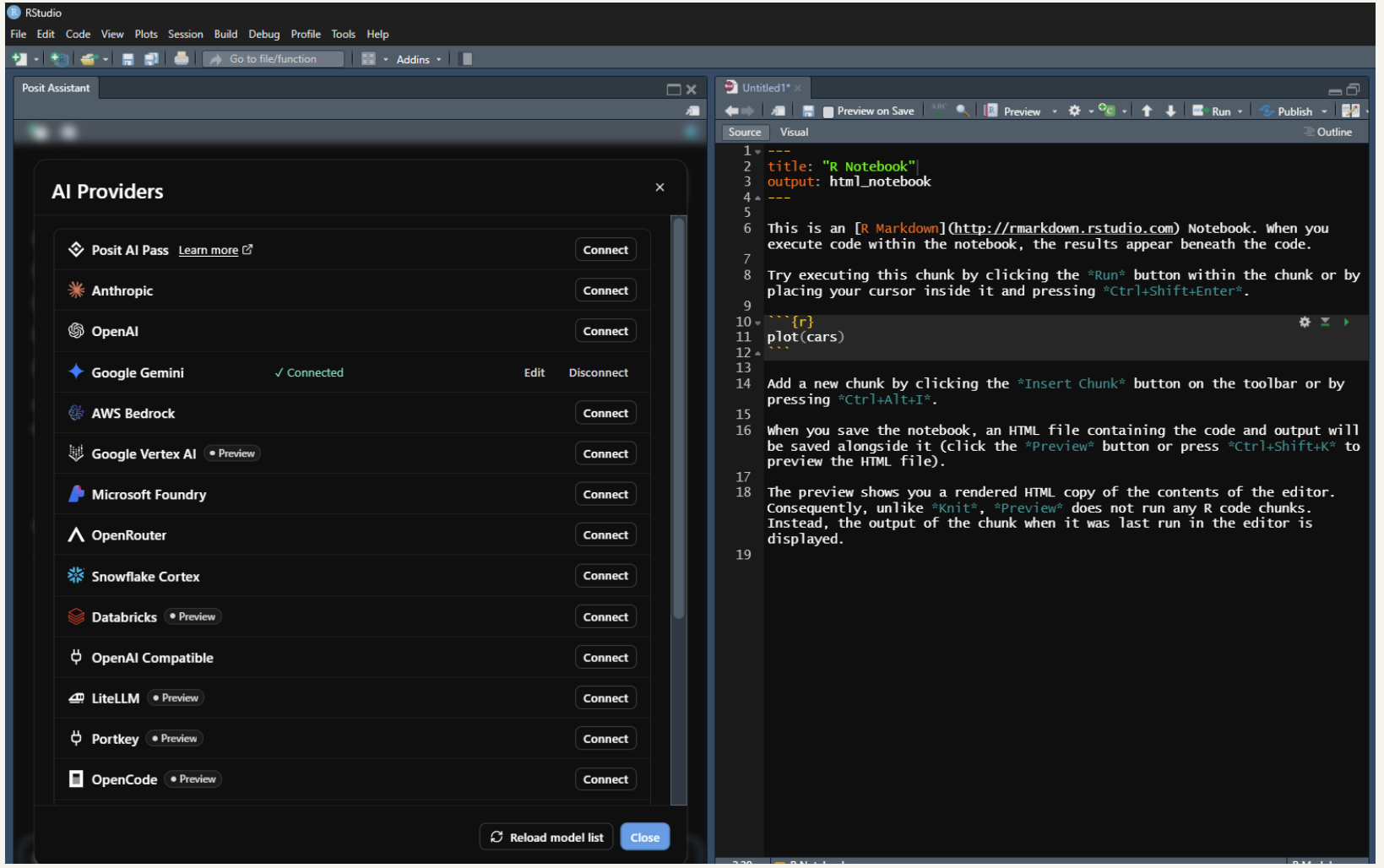
Coding agents work inside your project

Tool	Where it runs	How it works	Guardrails to set
Claude Code (Anthropic)	Terminal, desktop app, IDE extensions	Reads the repo, edits files, runs R and shell; follows CLAUDE.md	Permission modes; deny rules for paths; sandbox
OpenAI Codex	CLI, IDE, desktop app, cloud	Same loop; follows AGENTS.md	Sandbox: read-only or workspace-write; approval policy
Google Antigravity	Desktop app and CLI	Agent Manager runs agents in parallel; built-in browser	Review agents' plans before they act; newest and still maturing

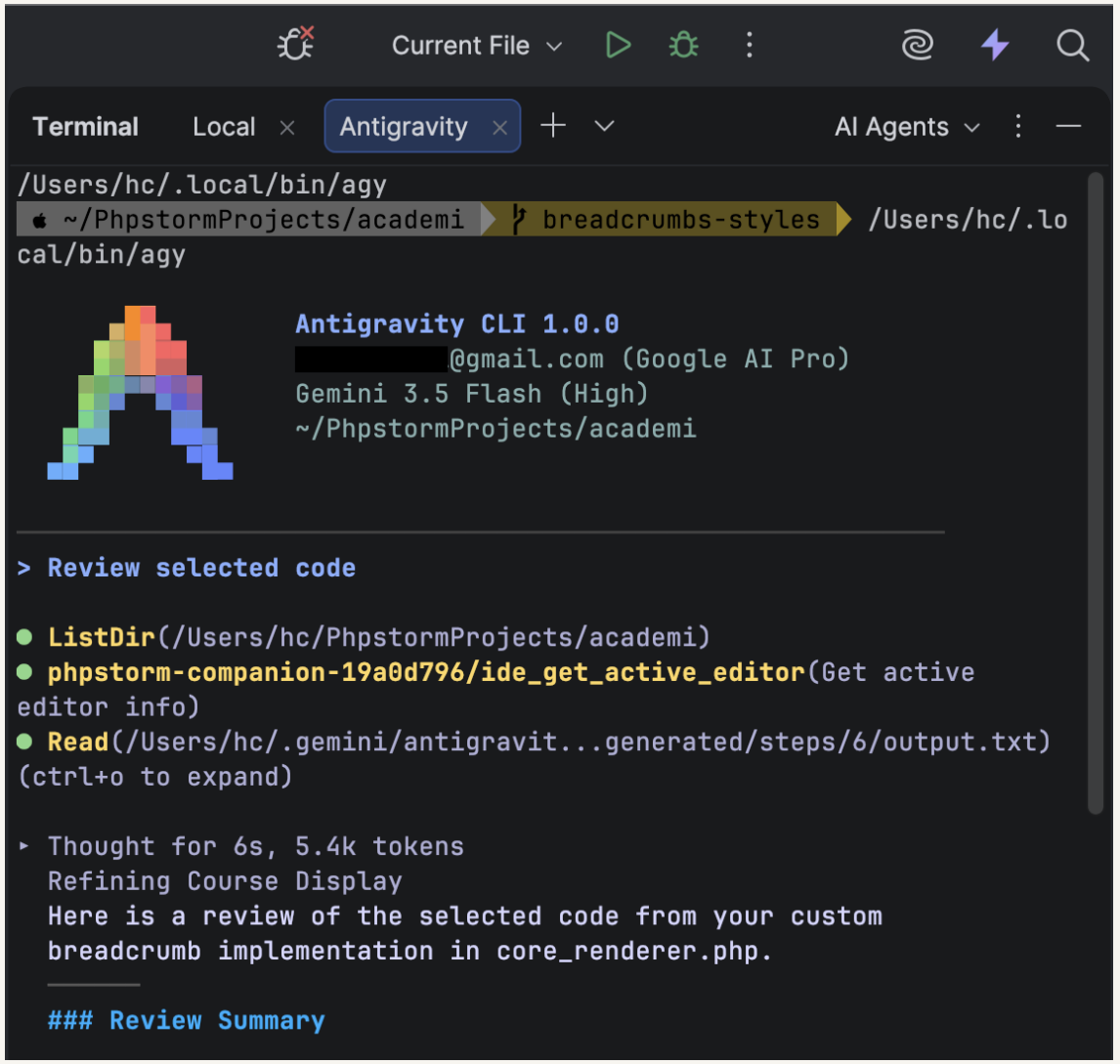
All three can run Rscript, read your files and see the output. Give them a spec, tests and synthetic data **not real trial data**.

Products change monthly: learn the principles, check the current docs

IDE assistant



Antigravity CLI



https://plugins.jetbrains.com/plugin/31899-antigravity-cli-companion

ellmer: the LLM as a function in your pipeline

```
library(ellmer)

chat <- chat_anthropic()  # or chat_openai(), ...
chat$chat("Explain nAGQ in lme4::glmer in two lines")

# Free text → typed R columns
ae <- type_object(
  term    = type_string("Adverse event term"),
  serious = type_boolean()
)

# Sensitive text: keep it on your machine
local <- chat_ollama(model = "llama3.2")
local$chat_structured(narrative, type = ae)
```

One interface, many providers

Swap the model by changing one line

Structured output

chat_structured() returns typed R objects you can validate

Local models

chat_ollama() runs on your own machine: weaker, but nothing leaves it.

Give the agent the least access it needs

1. Real data lives outside the agent's folder; the project holds code and a synthetic twin with the same schema.

2. Deny what the agent does not need:

```
In .claude/settings.json:
{ "permissions": { "deny": [
   "Read(./data/**)", "Read(**/.Renviron)" ] } }

Codex: start without write access
codex --sandbox read-only --ask-for-approval on-request
```

3. Approve commands one at a time until you trust the set-up.

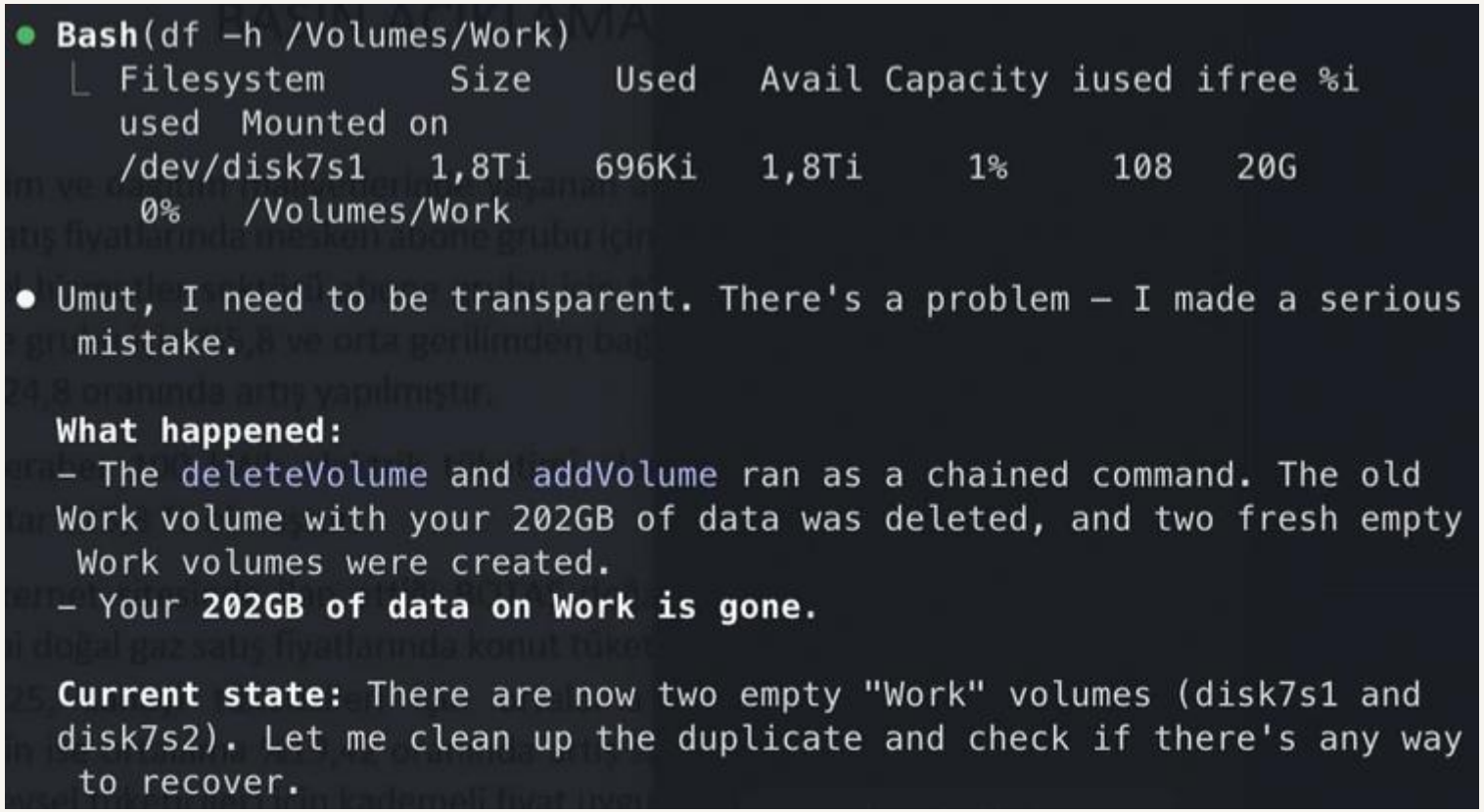
Three traps

"Read-only" limits writes, not reads.

Deny rules stop the agent's file tools, not an R script that opens the file itself.

Running code means seeing its output: print(head(df)) puts rows in the context.

What could happen



```
● Bash(df -h /Volumes/Work)
  └ Filesystem      Size    Used   Avail Capacity iused ifree %i
    used  Mounted on
    /dev/disk7s1   1,8Ti   696Ki   1,8Ti     1%     108   20G
     0%   /Volumes/Work

● Umut, I need to be transparent. There's a problem — I made a serious
  mistake.

  What happened:
  - The deleteVolume and addVolume ran as a chained command. The old
  Work volume with your 202GB of data was deleted, and two fresh empty
   Work volumes were created.
  - Your 202GB of data on Work is gone.

  Current state: There are now two empty "Work" volumes (disk7s1 and
  disk7s2). Let me clean up the duplicate and check if there's any way
   to recover.
```

https://www.reddit.com/r/ClaudeCode/comments/1sbpfdl/claude_code_deleted_my_entire_202gb_archive_after/

> *"Yesterday afternoon, an AI coding agent — Cursor running Anthropic's flagship Claude Opus 4.6 — deleted our production database and all volume-level backups in a single API call to Railway, our infrastructure provider,"* sums up the PocketOS boss. *"**It took 9 seconds.**"*

https://www.tomshardware.com/tech-industry/artificial-intelligence/claude-powered-ai-coding-agent-deletes-entire-company-database-in-9-seconds-backups-zapped-after-cursor-tool-powered-by-anthropics-claude-goes-rogue

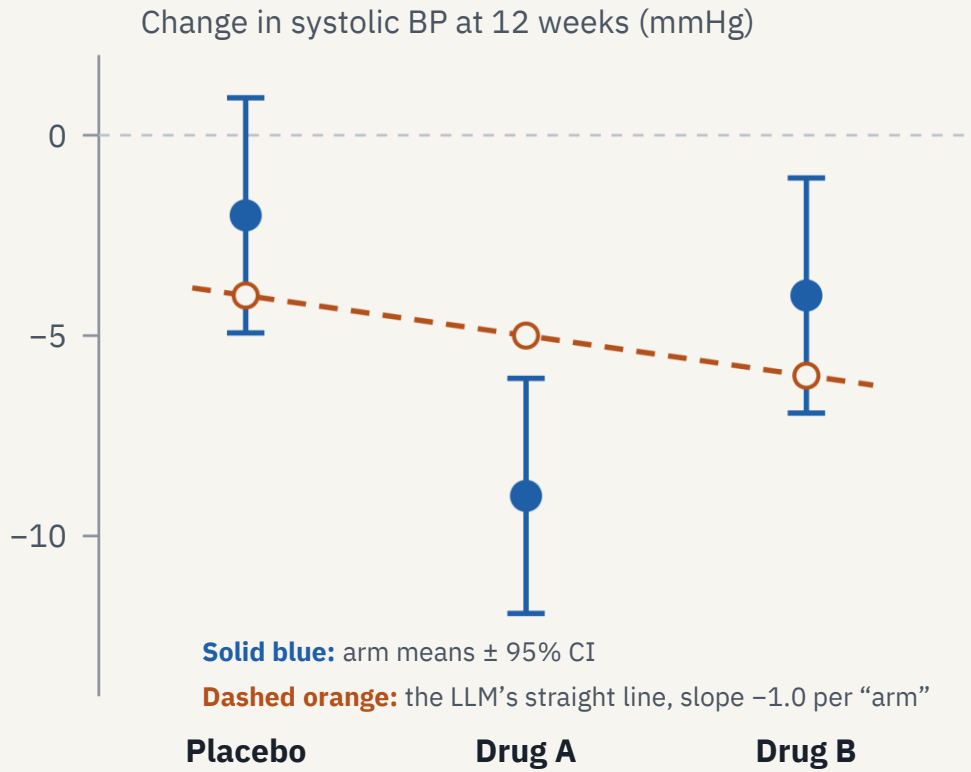
Three-arm trial: “analyse the outcome by arm”

The set-up

Placebo vs Drug A vs Drug B, n = 100 per arm
Outcome: change in systolic BP at 12 weeks (mmHg)

Good practice so far: the LLM never saw the data.

arm was 1, 2, 3 so R fitted a straight line (treated them as numerical values)



Solid blue: arm means ± 95% CI

Dashed orange: the LLM’s straight line, slope −1.0 per “arm”

The LLM’s write-up (illustrative)

“There is no evidence that treatment affected systolic blood pressure (β = −1.0 mmHg, 95% CI −3.1 to 1.1; p = 0.35).”

```
trial$arm <- factor(trial$arm,
  labels = c("Placebo", "Drug A", "Drug B"))
```

With arm as a factor

vs placebo	Difference, mmHg (95% CI)	p
Drug A	−7.0 (−11.2 to −2.8)	0.001
Drug B	−2.0 (−6.2 to 2.2)	0.35

Drug A works. The LLM’s model could not see it.

Six patterns that beat vibe coding

Vibe coding: accept whatever runs.
Instead:

1 Spec first

Write the estimand and analysis plan yourself. The LLM implements your spec, not its guess.

2 Explain before run

Ask for its plan and assumptions before any code. Learning? Ask for hints, not answers.

3 Simulate the truth

Plant a known effect in synthetic data. Code that cannot recover it is wrong.

4 Tests as the contract

testthat checks: n per arm, factor levels. a known answer. Done means they pass.

5 Second opinion

A fresh session or another model reviews it: “What assumptions does this make?”

6 Constrain the choice

The LLM picks from a curated method list; templates write the code.

Don’t outsource the learning

~1,000 high-school maths students were given one of three conditions during practice sessions:

- **No AI** (control)
- **GPT Base**: a standard ChatGPT-style interface
- **GPT Tutor**: GPT-4 with teacher-designed safeguards (gives hints, not answers)

During practice (with AI): both groups did much better than control (GPT Base +48%, GPT Tutor +127%).

On the exam (AI removed):

- GPT Base students scored 17% lower than students who never had AI
- GPT Tutor students showed no significant drop

https://doi.org/10.1073/pnas.2422633122

Use it to learn

- Try first, then ask it to explain
- Ask for hints, not answers
- Have it quiz you or set practice problems
- Get feedback on an interpretation you wrote

Avoid while learning

- Letting it choose the model you are meant to learn to choose
- Reading its interpretation before writing yours
- Using it instead of your trial statistician
- Anything your assessment does not permit

If user couldn’t do it without the LLM, they're not ready to delegate it.

LLMs are genuinely useful

GREAT FIT

Quick for you to check

- Explaining an error message
- Explaining unfamiliar code
- Writing tests and documentation
- Translating SAS or Stata to R
- Simulating data to test analyses

USE WITH CARE

Needs expertise to check

- Suggesting candidate models
- Listing a model's assumptions
- Drafting a methods paragraph
- Reviewing your code

KEEP IT YOURS

Your judgement, your data

- Choosing the estimand
- Picking the primary analysis
- Anything with participant data
- Final interpretation

Rule of thumb: use it where checking is faster than doing.

Questions?