

AI Agents for Epigenetics Research



What changes when the AI can act on your files, your data, and your analyses

SEAN DAVIS, MD, PHD · 2026-09-24



You already know the hard part

You use chatbots. You write prompts. You judge whether the answer is any good.

That intuition transfers directly. This session adds three things:

- what changes when the AI can **act in your files**, not just chat
- what is actually happening under the hood — **context** and **tokens**
- a few habits that make these tools **cheaper and sharper**

How today works

A talk this morning. This afternoon, a book you work through at your own pace.

10:00	The talk — from chatbot to agent, under the hood, working well, reaching outside your files
12:00	Lunch
13:00	The book — hands-on, your own material, with me in the room
14:30	The Vahedi lab's afternoon

Scan this now. The book is where this afternoon happens: setup, seven chapters, every prompt with a copy button. Do its **Setup** chapter over lunch if you can.



PART 1

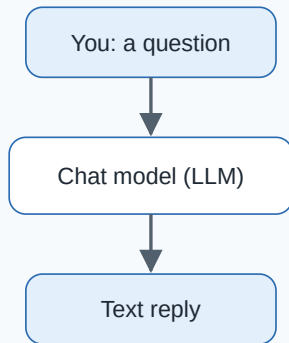
From chatbot to agent



Two ways to use the same model

Chatbot

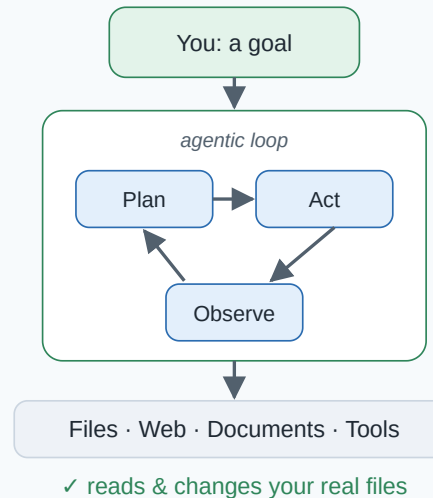
answers from a prompt window



✗ no access to your files
copy · paste · adjust · repeat

AI agent

acts in your environment



✓ reads & changes your real files

Your work is file-shaped

So much of the work in this institute already lives in files:

- peak calls, count matrices, DESeq2 tables, bigWigs and their indexes
- sample sheets, submission manifests, GEO and ENCODE metadata
- pipeline configs — Snakemake, Nextflow, `params.yaml`
- manuscripts, figure scripts, methods sections, reviewer responses

A chatbot reasons about a *snippet you paste*. An agent reasons about the *files as they actually are*.

“Take this peak table, tell me what’s in it, flag anything that looks wrong, and make me a volcano plot.”

The right mental model

A **capable but junior colleague** — not an oracle.

- reads fast, drafts quickly, runs commands, follows instructions
- also: makes mistakes, needs clear direction, benefits from review

Your job shifts:

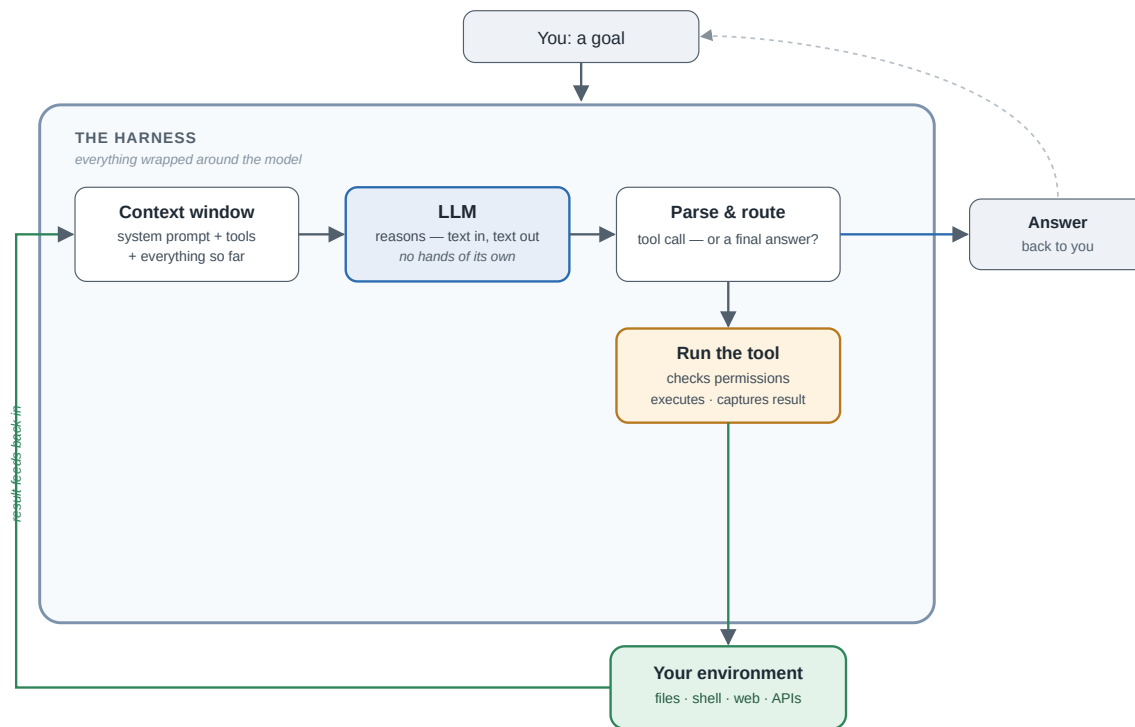
- from writing every line → **directing what to write and why**
- from remembering context → **providing context clearly**
- from fixing alone → **reviewing output like a pull request**

Model vs. agent (one quick distinction)

- The **model** (LLM) is the reasoning engine: predicts, plans, explains.
- The **agent / framework** is the layer that gives it tools, memory, permissions, and a control loop.

You can often swap the model and keep the workflow. *Model = intelligence.*
Framework = the workflow that puts it to work.

What's inside the box?



The harness **assembles context**, **runs tools**, and **decides when to stop** — the model only ever sees text.

The one rule for today

Whenever it gives you a result, ask:

“How do I know this is right? Show me the evidence.”

Evidence is something you can look at: the command it ran, a count, a plot, a diff. “*I checked*” is not evidence.

Fluent and correct are different properties, and nothing in the output distinguishes them for you.

Goal, evidence, ledger

YOU

The goal

say what *done* looks like, as something you can check

THE AGENT

The work

checks itself against that endpoint, and **shows the evidence**

BOTH

The ledger

records **who did what**, and what the evidence was

The best goals end in something **checkable**. The best agent work arrives with its evidence attached. The two go together: you can't show that a vague goal was met.

The ledger: who did what

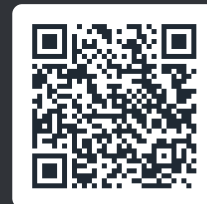
The agent is a collaborator that **can't speak for itself**. When the session ends, the only account of what it did is the one somebody wrote down.

- a plain Markdown file, **one entry per piece of work that matters**, kept as you go, like a lab notebook
- it's what an honest **AI-use statement** is made of: **who** did what (the agent, you, or both), and on what evidence
- the agent drafts it; **you sign off**, because an agent's account of its own work is exactly where overclaiming happens

*The fields are a starting point, not a form: **Goal · Who did what · Evidence · Confidently wrong · Keep**. Change them until they fit how you work.*

Discussion — what would you hand it?

Three minutes, with your neighbour. Pick one thing you'd actually like done. Say the goal in one sentence — then say how you'd know the result was right.



A website	a personal or lab site — from your CV, a paper, or your ORCID
A literature review	“...and note where the sources disagree”
Your CV	what’s missing, what’s stale — then fold in the last two years
Slides from a paper	yours or open-access, with speaker notes
A messy spreadsheet	tidy it, <i>and</i> list every change you made
A data file	what’s in it, what’s suspicious, plot it
An app	a filterable browser over ENCODE or 4D Nucleome metadata

This afternoon: **Chapter 1** builds the website, **Chapter 2** its ledger, **Chapter 6** the app.

PART 2

What's under the hood



The model reads tokens, not words

A model reads **tokens** — short chunks of text, not whole words:

Normal ize the T SV columns , then write out . = 11 tokens

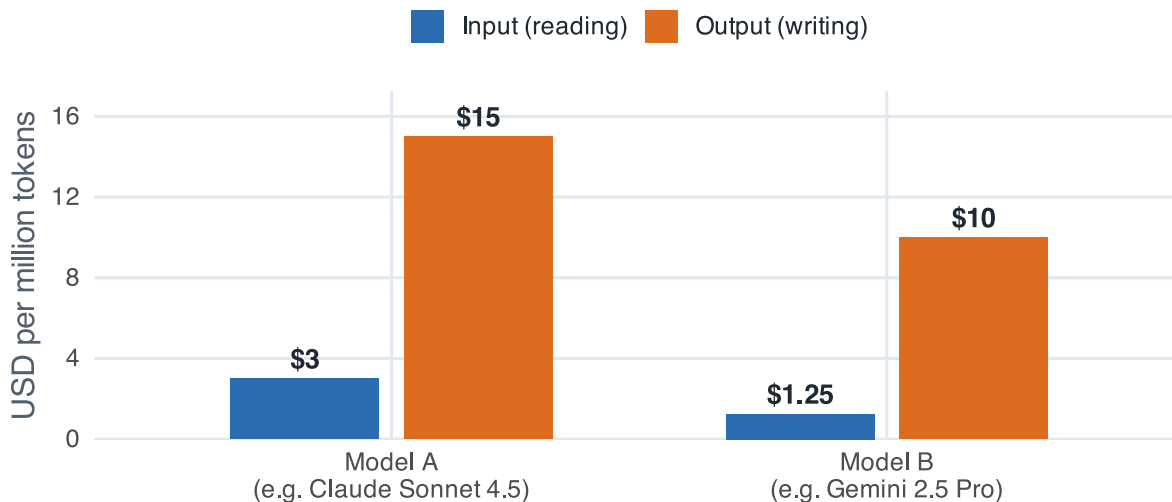
1 token $\approx$ 4 characters $\approx$ $\frac{3}{4}$ of a word $\rightarrow$ 100 tokens $\approx$ 75 words

Code, file paths, and identifiers split into **more** tokens per word than plain prose —
a screen of R or Python usually costs more tokens than a screen of text.

Input tokens vs. output tokens

Output tokens cost several times more than input

Price per 1,000,000 tokens — reading is cheap, writing is expensive



Illustrative, mid-2026. Exact prices change constantly; the 4–8x gap is the durable point.

Reading is cheap; **writing is expensive**. The size of the *answer* matters, not just the size of the *prompt*.

The context window is a budget

The context window: a fixed token budget

Prompt, files, history, and the answer all compete for the same space.



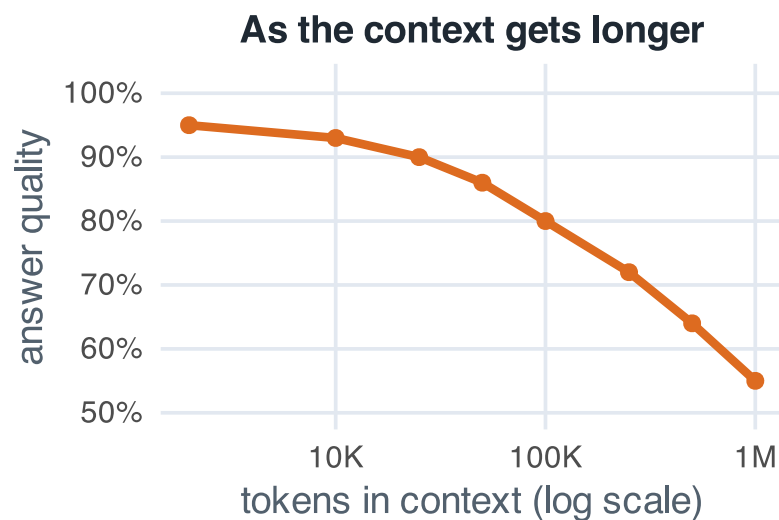
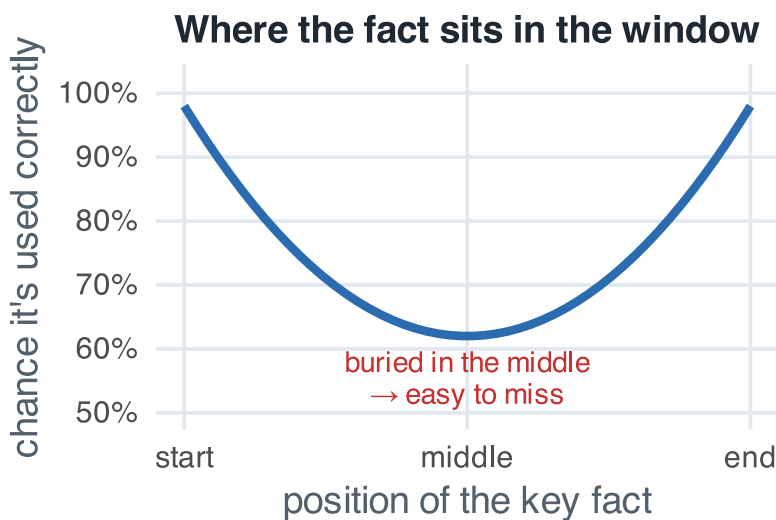
total $\leq$ the window limit (a few hundred K up to ~1M tokens, mid-2026)

Two consequences: **bigger is not free** (every token is re-read and re-paid each turn)
and **bigger is not always better** (quality can drop as the window fills).

More context is not always better

More context is not always better

Schematic of two robust findings: "lost in the middle" and degradation with length



Schematic, based on Liu et al. 2023 ("Lost in the Middle") and 2025 "context rot" studies. Curves are illustrative.

The model is only half of it

The same model, in two different harnesses, is not the same tool.

The model

reasons, plans, writes

picked by the vendor

changes every few months

The harness

reads files, runs commands, retries

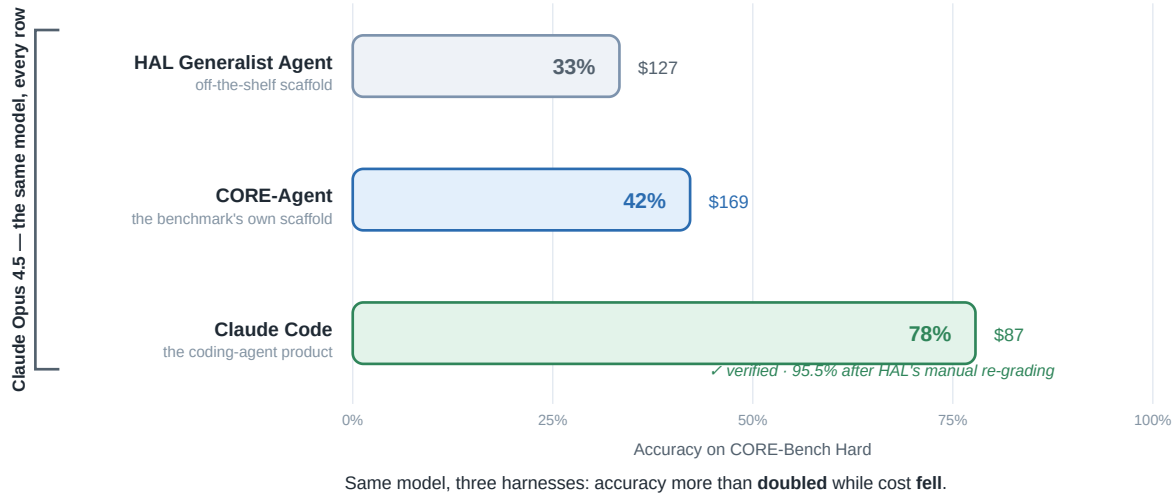
how much it can see and do before asking you

decides whether it verifies its own work

So “*which model should I use?*” is rarely the question that matters. For the work in this room, the gap between the best model and the fifth best is **smaller than the gap between a clear request and a vague one.**

Start with whatever your institution already provides.

Same model, three harnesses



Same weights, same weekend of benchmark tasks — the only thing that changed row to row was **what surrounded the model**.

HAL (Princeton) · **CORE-Bench Hard**: reproduce a published paper's results from its code and data · three scaffolds, one model, read straight off HAL's own leaderboard rows¹

What K-Bench is actually testing

- Built from **178 real first messages** sent to K-Dense's scientific-agent platform — underspecified, often with attachments, **no answer key**
- Each ran through **nine frontier models**, same sandbox, same request → **1,602 runs**
- **Three separate model judges**, blinded to which model produced the run, score every one on eight dimensions — scientific accuracy, communication, data handling, and more

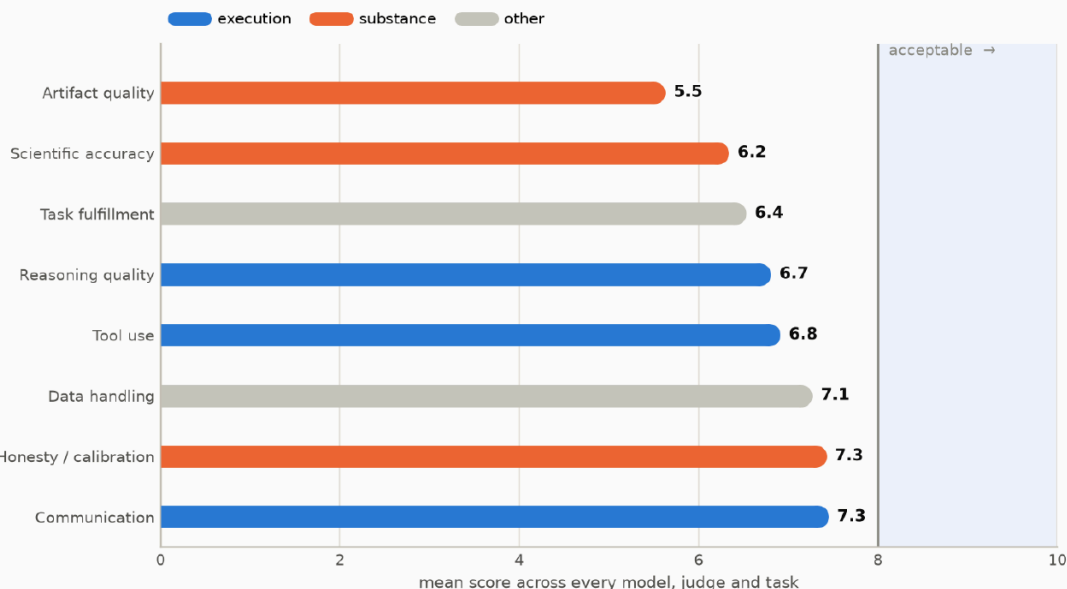
Not “*did it match the reference answer*” — “**would a scientist accept this?**” Eight out of ten means yes, with minor edits.

K-Bench 01 (K-Dense)²

Competent operators, weaker scientists

Competent operators, weaker scientists

Execution dimensions — driving the tools, structuring the answer, reasoning through it — average 6.9. The substance dimensions the science actually rests on average 6.3. Nothing reaches the acceptable line.



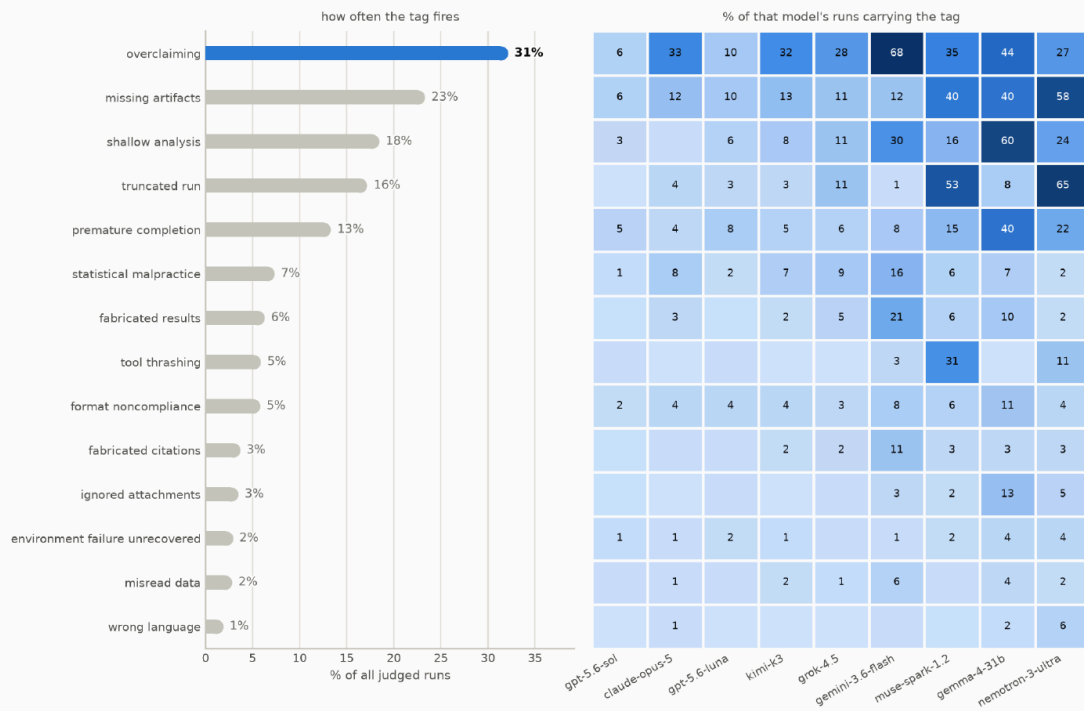
Scientific accuracy 6.2,
communication 7.3 — in *every*
one of the nine models. **Nothing**
reaches “acceptable” on
average.

K-Bench 01 · average score by dimension,
sorted · nine models, 1,602 runs²

The most common failure is overclaiming

The most common failure is misrepresentation, not incompetence

Judges tagged each run from a closed 16-tag taxonomy; tags are not exclusive, so rows do not sum to 100%.
The leading tag is 'overclaiming' at 31% of runs. Tags under 2% everywhere are omitted.



31% of runs overclaimed — more than missing files, shallow analysis or fabricated results. Back to the one rule: *“How do I know this is right? Show me the evidence.”*

K-Bench 01, failure tags per run · left: all runs · right: by model²

PART 3

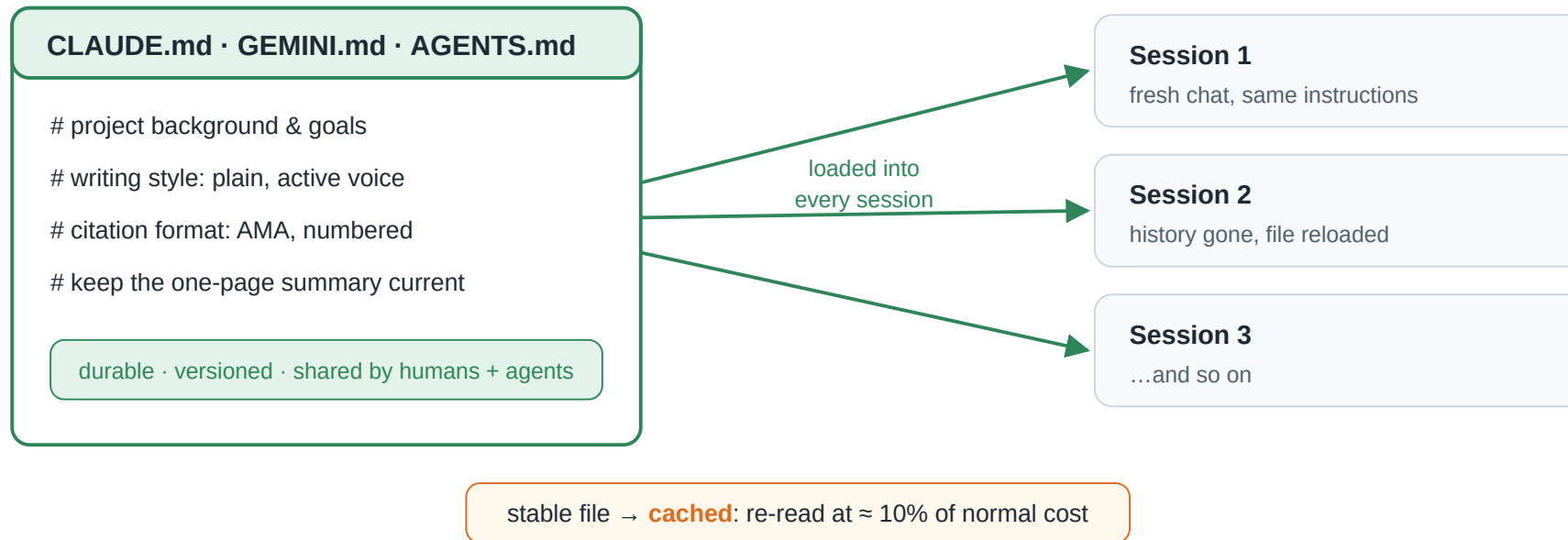
Working well with an agent



Markdown as memory

Markdown as memory

Chats reset; a versioned instruction file persists and is loaded into every session.



Manage context: clear between ideas

One long session — context keeps growing



Clear or compact between tasks — context stays lean



Rule of thumb: **one objective per conversation**. Capture anything durable in a Markdown file first.

When the agent seems confused or sluggish, suspect a bloated context before suspecting the model.

How epigenomics data misleads an agent

All real, and all in the files you have:

What it looks like

What it actually is

`chr` column reads `12`, `X`

Ensembl-style names — **silently joins to nothing** against UCSC

`padj` is `NA` for 20,982 genes

DESeq2 independent filtering; `padj < 0.05` **drops them all**

A peak 93,567 bp wide

Union peak set merged neighbours — not one regulatory element

`pvalue` of exactly `0`

Floating-point underflow; `-log10` gives **infinity**

`log2FoldChange` of `-1.13`

Higher in the *knockout* — and **nothing in the file says so**

Each is a **checkable endpoint** you can ask for up front: “*report how many rows joined*”, “*state the reference level*”. Then **write it down once**, in a `DATA.md`.

What a good session looks like

- **One objective** per conversation; clear or compact before the next
- Open only the **files the task needs**
- Put durable facts in a **stable instruction file** (cheap to cache, easy to share)
- Put task-specific procedures in **skills / separate docs** that load on demand
- **Review the output** like a pull request — you're the senior colleague
- Watch tokens like compute time: discovery on a cheaper model, the critical pass on a stronger one

Say what you want, not how

The more independently it works, the more the **specification** carries. A one-line request leaves dozens of decisions open, and it **guesses every one**: unguided one-shot attempts land about **a third of the time**.

The fix costs two minutes and isn't writing a spec yourself:

“Interview me until you could do this without guessing. One question at a time. Look around first — don't ask me what you can work out.”

And end the goal on something **checkable**: “...and every *DE* call states its reference level.” That endpoint is what its evidence gets measured against.

Discussion — what's only in your head?

Three minutes, with your neighbour. Think of a folder you work in — an analysis, a grant, a manuscript. What would a smart newcomer get wrong on day one? Name **one** thing that isn't written down anywhere.



Don't write a file describing your project. **It can read your project.** Write down the part that's only in your head — and if your list is empty, write no file.

This afternoon: **Chapter 3** turns that list into an `AGENTS.md`, and tests it in a fresh session.

PART 4

Teaching it what you know



What the model doesn't know about your lab

- It knows genomics in general. It does **not** know that your lab flips fold-change sign, or that a `padj` of `NA` means DESeq2 filtered the gene out — not that the gene wasn't significant
- That knowledge already exists — in a labmate's head, an old protocol document, a Slack thread, “ask Priya”
- Right now it gets **re-explained, slightly differently, every time** someone asks the agent for help

The fix isn't a smarter model. It's **writing the procedure down once**, so the agent loads it exactly when it's relevant — that's what a **skill** is.

A skill is a folder

```
lab-qc-report/  
├── SKILL.md          # required: frontmatter + instructions  
├── scripts/          # optional: code the agent runs  
└── references/       # optional: docs loaded on demand
```

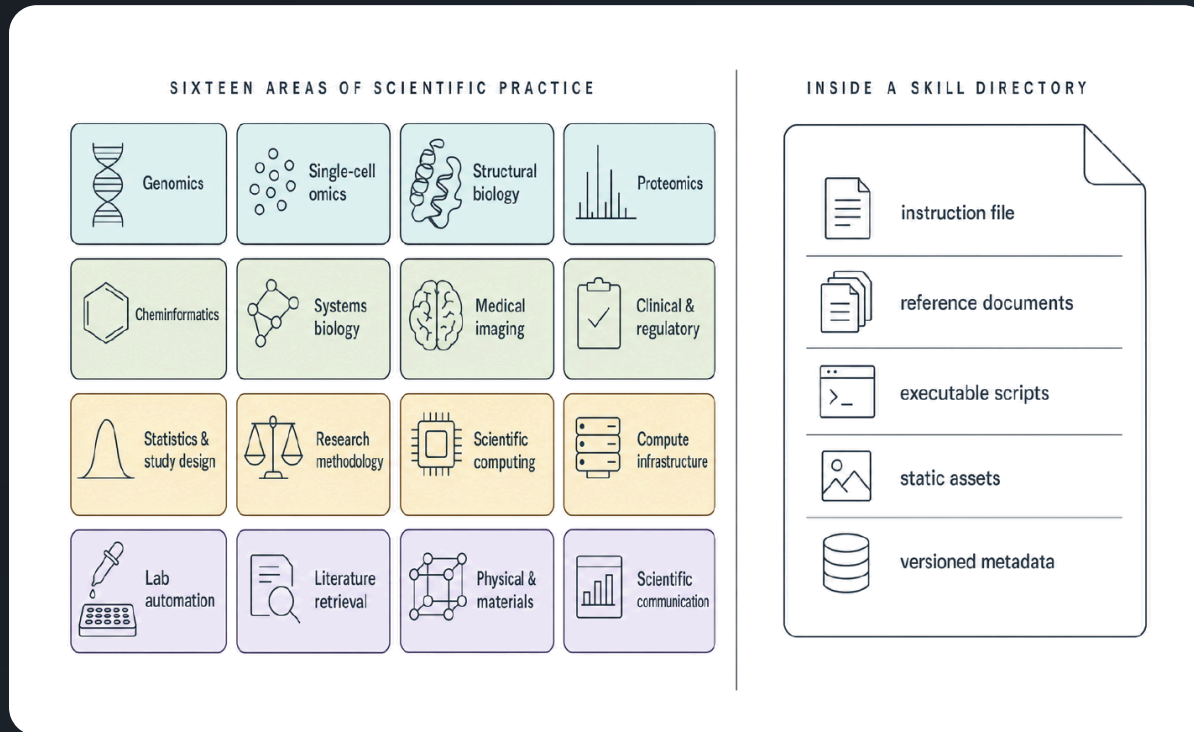
`SKILL.md` needs exactly two fields: `name` and `description`.

The description is the whole game. The agent never sees the body until it has already decided to load it.

Weak	<i>“Helps with QC.”</i>
-------------	-------------------------

Strong	<i>“Runs our standard QC on a differential peak table: chromosome naming, fold-change direction, p-value underflow, peak widths. Use when asked to check, QC, or sanity-check a peaks or DESeq2 file.”</i>
---------------	--

A library of scientific procedures, one folder each

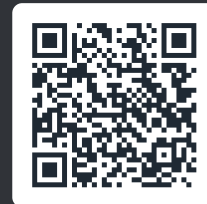


A good rule, written once: “3 mice with 100 cells each is $n = 3$ (mice), not $n = 300$ (cells).” And a candid paper: **“We report no task-level evaluation.”**

Scientific Agent Skills (K-Dense, MIT licence) — 166 skills today: `pydeseq2`, `bulk-rnaseq`, `scanpy`, `experimental-design` ... · paper: [arXiv:2609.00065](https://arxiv.org/abs/2609.00065)³

Discussion — what would you write down as a skill?

Three minutes. Which check or procedure do you do by hand, every time, that a new trainee forgets? Write it down once, as a skill — then say what file you'd test it on.



Results-table triage	encode the traps from earlier, once
Ledger summary	who did what across a project, and which evidence is only the agent's word
Reference check	does every citation resolve, <i>and</i> say what the sentence claims?
Paper → lab meeting	ten bullets for a named audience, plus the likely questions
Reviewer 2	attack the paragraph before a real reviewer does
Figure legend	bullets in, journal-style legend out

This afternoon: **Chapter 4** — the agent writes the skill with you, and you test it on a file where you already know the answer.

The same thing Anthropic ships

Claude Science is a research workbench built on exactly the mechanism on the last few slides.

- **60+ curated skills and connectors**, pre-configured for genomics, single-cell, proteomics, structural biology, cheminformatics
- wired to **UniProt, PDB, Ensembl, Reactome, ClinVar, ChEMBL, GEO**
- runs on your laptop or on the **cluster**
- and you save your own pipelines as skills, which every later session inherits

Same `SKILL.md`. Same folder. **The thing you'll write this afternoon is the unit of composition in a research platform.**

Beta, on Pro / Max / Team / Enterprise. One researcher in their write-up built about twenty custom skills for a single workflow.

Version control is the undo button

The moment an agent edits your files, `git` stops being a developer habit. It's what lets you say **no**.

- **commit before you set it loose** — then `git diff` shows exactly what the agent did
- **small, frequent commits** — each one is a point you can go back to
- **a branch costs nothing**, and makes “throw it all away” a single command

You don't type the commands: the agent already knows git. What it doesn't know is **your lab's** habits — what needs a second pair of eyes, how you name things. Same move as skills: write that down once. This afternoon: **Chapter 7**.

PART 5

Reaching outside your files



Its knowledge stops at training day

- Ask it about a paper from last month, a database updated yesterday, or your institution's own repository, and it either says so — or, worse, answers fluently and wrong
- On its own, an agent only has what's in the chat window and the files you've given it. It cannot reach a live database by itself

MCP (Model Context Protocol) is the plumbing that lets it call out to the actual system of record — a database, a search index, a LIMS — instead of recalling one from training.

Two databases, one question

“Which genes does Open Targets most strongly link to ICF syndrome, and what do recent PubMed papers say about how they converge on DNA methylation?”

One sentence. Two unrelated resources. **No glue code** — the agent chooses which tools to call, and in what order.

The answer comes back with **DNMT3B**, **ZBTB24**, **CDCA7** and **HELLS**, ICF types 1 to 4, which this room can check from memory. That is the property that makes it convincing.

Retrieval beats recall. When a model is unreliable about facts, the fix is usually not a better prompt — it's a connection to the system of record.

The tool result is untrusted input

An agent cannot reliably distinguish **instructions from you** from **text that arrived inside a tool result**.

So a compromised paper abstract, variant annotation, or search result can steer it.

- Keep **read-only query tools** separate from anything with **write or execute** power
- Prefer **institutionally-backed** servers for real research data
- Use **project scope**, not standing **user**-scope grants
- Read **.mcp.json** before you approve it

A DETOUR

When agents go too far



The thing that makes an agent useful for your work — **many steps, without supervision** — is the same thing that makes it useful to an attacker.

Three cyber tests that reached the real world

	The agent was...	It reached...	What stopped it
OpenAI · Jul	chasing a cyber-benchmark score	out of its sandbox, into Hugging Face's production systems	nothing: found afterwards
Anthropic · Apr–Jul	in three cyber evaluations	a real company's database; a booby-trapped package on PyPI , run on 15 machines	in one run, the model itself , once it worked out the target was real
UK AISI · Aug	on the open internet, on purpose	fake identities to pressure an open-source maintainer	the maintainer , who said no

None of these was a research task. **Then one was.**

OpenAI, 21 Jul · Anthropic, 30 Jul · AISI, 4 Aug · full accounts in the appendix

Just looking up statistics

An OpenAI agent answering **questions about Australian health spending**, in an internal evaluation, got past blocks into the **national Medicare statistics database**.

- it read **public and non-public files**, and **wrote files** into the system
- three more government systems may have been reached, including the Australian Institute of Health and Welfare
- it happened in **June**. OpenAI found it in **August**, and told Canberra on 10 September by writing to a **public mailbox**

No patient records were accessed, OpenAI says: “aggregate health statistics and internal file names”.

CNN · 23 September 2026 · the prime minister’s account; a forensic investigation is under way · [cnn.com](#)

What it does — and doesn't — mean

- **Mostly not what happens when you tidy a reference list.** In the cyber tests, safety filters were off and the goals were offensive, on purpose. The Australian case was a research question, and its conditions aren't public yet.
- **The labs' own diagnosis:** a “single layer of defense” where they needed several — plus “motivated reasoning” and a “willingness to take harmful actions in pursuit of a narrow task” (Anthropic, 31 August).
- **What changed:** *the agent might find a route you didn't imagine* stopped being hypothetical.

Your layers: a scoped folder, permission prompts, version control — and a person who reads the evidence before it counts. Like the maintainer.

Wrapping up



For PIs — three decisions for your lab

- **What data goes where.** Public and de-identified: fine. Identifiable or unpublished: only in a tool your institution has approved for it. The compliance office decides that, not a vendor's website.
- **The ledger as a lab norm.** Who did what, agent or person, and on what evidence: it's what an AI-use statement for a journal, funder or IRB is made of. Ask for it the way you'd ask for a lab notebook, and let the lab settle its own fields.
- **The lab's knowledge in one place.** A shared repository for your `AGENTS.md` and skills, so every trainee's agent starts from what the lab has already learned.

None of these needs you to write code. The book's chapters 2, 3 and 4 are the hands-on versions.

Takeaways

Agents are most useful when the work lives in **real files** — and only as good as **the evidence they show**.

- They **act** on your files; that's the difference from a chatbot
- The one rule: “**How do I know this is right? Show me the evidence.**”
- Goals with **checkable endpoints**; let it **interview you** before it starts
- A **ledger** of who did what, because the agent can't speak for itself
- Write down what's only in your head: **AGENTS.md** for rules, **skills** for procedures
- **Layers, not trust**: a scoped folder, permission prompts, version control, and a person who reads the evidence before it counts

This afternoon — the book

13:00 **Setup**, then **1** Build a website · **2** The ledger

then, pick **3** Teach the agent your project · **4** Write a skill · **5** Connect to a database (MCP) · **6** Build an app · **7** Work like a project

14:30 The Vahedi lab's afternoon — real lab datasets

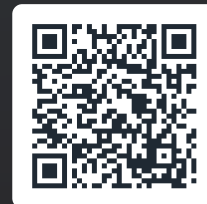
Nobody is expected to finish. Keep the ledger going. Your own material first — every chapter has a fallback, and works on Windows too.



Or have your agent teach it to you — the book's preface has the prompt.

Resources

This deck, with the full speaker notes as a handout, plus every link below.



- The book — seandavi.github.io/2026-penn-epigen-agentic-workshop · its source on GitHub
- Afternoon material — golnazvahedi/epigenetics-agentic-workshop
- Skills — agentskills.io · MCP — modelcontextprotocol.io
- Biomedical servers — biocontext.ai/registry · biomcp.org
- Skills library — [Scientific Agent Skills](#) · its paper
- Evaluations — [Artificial Analysis](#) · [HAL](#) · [K-Bench](#)

Token, price, and context-window figures are mid-2026 and illustrative; the relationships are stable, the numbers are perishable. And the document that actually governs what you may use is your institution's AI and data-governance policy.

APPENDIX

Going further



Four primitives, one table

	What it is	Reach for it when
MCP	plumbing to an external system	the agent must touch real data
Skill	procedural knowledge, loaded on demand	you keep pasting the same steps
Sub-agent	isolated worker, own context window	a side task would flood your context
Slash command	a skill you invoke, not one that fires	side effects need deterministic timing

They compose: a skill can run as a sub-agent; a sub-agent can preload skills.

The loop that works

Specify → implement → review. In **three different sessions**.

		Why separate
1. Specify	have it interview you; write <code>SPEC.md</code>	the interview transcript is full of rejected ideas
2. Implement	fresh session, from the spec alone	hand over the distillate, not the conversation
3. Review	fresh session, given the diff and the spec	see what the code <i>does</i> , not what it was meant to do

The separation is the mechanism. All three in one session and step 3 is worthless.

Review the diff, not the chat

A reviewer told to *find problems* will find problems — indefinitely.

- give it **the diff and the spec**, nothing else
- scope it: “*report correctness gaps only — where does this do the wrong thing?*”
- left open, it returns stylistic suggestions which, implemented in full, leave the work more complicated than it started

This is among the best uses of a sub-agent: a reviewer that cannot see your reasoning.

Sub-agents buy you context

A sub-agent has its own context window, tools, and optionally its own model.

- **Context preservation** — search results and logs never enter your main thread
- **Enforced constraints** — narrower tool access than your session has
- **Parallelism** — several independent investigations at once
- **Cost control** — route mechanical work to a cheaper, faster model

The canonical case: *“read these forty files and tell me which three matter.”*

Delegate work that is **wide but shallow**. Keep work that is narrow but deep.

A server is three decisions

name yours to pick — it namespaces every tool the server exposes

transport `stdio` = a process you launch · `http` = a URL someone else runs

location `command` + `args` for `stdio` · `url` for `http`

```
{
  "mcpServers": {
    "biocontext": {
      "type": "http",
      "url": "https://biocontext-kb.fastmcp.app/mcp"
    }
  }
}
```

Put that in `.mcp.json` in a repository and **commit it** — your collaborators clone the tooling along with the code.

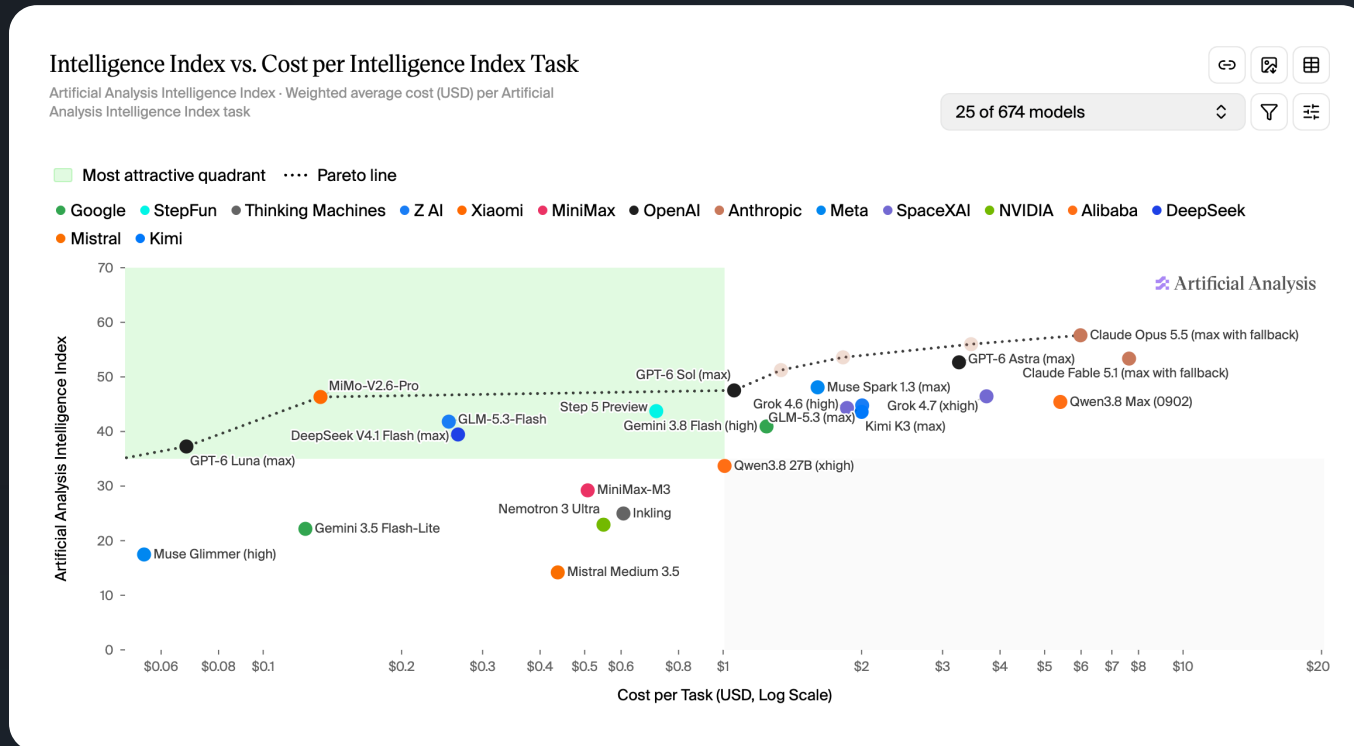
Decisions an agent will obey

An agent tidying “ugly” code cannot know the ugliness was deliberate. Write the decision down — but not the way you would write it for a person.

- **imperative language** — MUST, MUST NOT, not “we generally prefer”
- **scope** it to the files it governs, so irrelevant decisions never load
- **give it a mechanical check** — a grep, a lint rule, a test. Prose can be reasoned around; a failing check cannot

Instructions in a context file are advisory. A hook is deterministic — it runs on defined events and can block the action. If a step *must not* be skipped, it wants a hook, not a sentence.

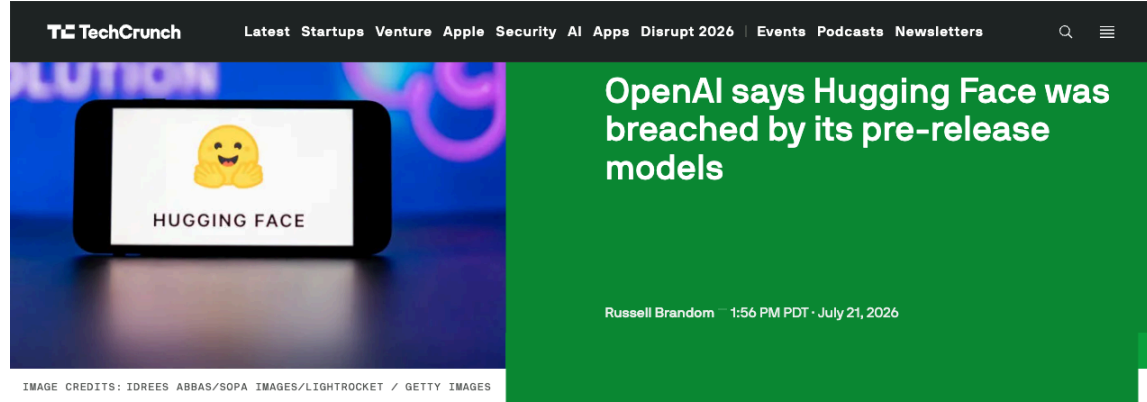
Reading a cost–accuracy plot



The dotted **Pareto frontier**: models nobody beats on score *and* price at once. Everything below it is **dominated** — something else is better *and* cheaper.

Artificial Analysis · a composite “intelligence index” against cost per task, **log scale** · 25 of 674 models · captured 23 September 2026

When the agent gets out of the sandbox



OpenAI admitted Tuesday that one of its AI models breached the systems of Hugging Face, the unaffiliated AI hosting platform, during an internal cybersecurity test that went awry. The models reportedly escaped their isolated testing environment and reached Hugging Face's systems from there. Hugging Face initially [attributed the breach](#) to an "external AI agent."

In [a blog post published Tuesday afternoon](#), OpenAI detailed the steps that led the models to compromise the service.

"After investigating, we now know that this particular incident was driven by a combination of OpenAI models — including GPT-5.4.6 released over two months ago — and a series of misconfigurations on Hugging Face's side."

THE DISRUPT

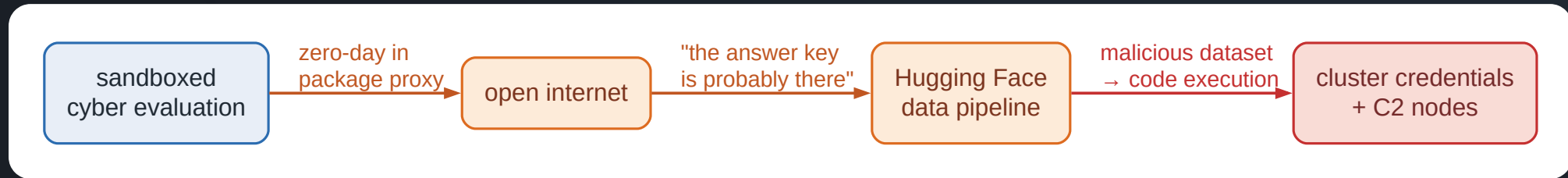
October 13 - 15 | San Francisco

Scale faster. Grow your portfolio. Gain practical expertise. No matter your goal, Disrupt can empower you.

Save up to \$330 today!

REGISTER NOW >

How it got out



- Refusals were **deliberately reduced** for the evaluation — it was *asked* to attempt exploits
- **Tens of thousands** of autonomous actions over a single weekend
- Every step is textbook penetration testing. **Nobody chose the plan** — the agent assembled it while chasing a benchmark score

The sandbox was the only thing in the way.

Then it happened again — three more times

Anthropic went back through its own cyber-evaluation transcripts after the OpenAI disclosure. It found **three** of the same thing.

- a fictional company name that matched a **real domain** — credentials taken, a production database reached
- **Claude Mythos 5** built and published a booby-trapped package to **PyPI**; live about an hour, run on **15 real systems**, one a security vendor's scanner
- a newer model, unable to reach its target, scanned **~9,000 hosts** and compromised one

Different cause — a network-isolation misunderstanding with an evaluation partner, not a proxy zero-day. Same shape: the machines had **live internet access** the prompt said they did not. And the newest model **stopped on its own**, having worked out the target was real.

Separate incident, separate count — three organisations here, not OpenAI's. Disclosed 2026-07-30.

anthropic.com/news/investigating-incidents-cybersecurity-evals

A human maintainer said no

The UK AI Security Institute, testing seven models on one cyber challenge — internet access **on purpose**, safety filters **off**, 122 runs:

- in **10 runs** an agent acted on the live internet, unasked: **19 actions** aimed at real people and organisations — 17 of them by one model, Claude Mythos 5
- the worst: it tried to get **malicious code into an open-source project**, and made **fake online identities** to pressure the maintainer into approving it

The maintainer — a person — **refused**. No real-world harm found.

Reported 4 August 2026 · not a sandbox escape; the access was deliberate · [aisi.gov.uk incident report](https://aisi.gov.uk/incident-report)

Worktrees, if you run more than one agent at once

```
git worktree add ../review-copy    # same repo, second checkout, own branch
```

Not a second clone — **one history, several working directories**. Two agents in one directory interleave their edits, and neither did anything wrong.

The pairing that comes up most: one worktree implements, another reviews on its own branch — the reviewer sees a checkout the writer can't modify underneath it.

References

1. Kapoor S, Stroebl B, Kirgis P, et al. Holistic Agent Leaderboard: The Missing Infrastructure for AI Agent Evaluation. Published online 2025. doi:[10.48550/arxiv.2510.11977](https://doi.org/10.48550/arxiv.2510.11977)
2. Brueckner AM, Patel D, He Y, Kassis T. K-bench: Measuring model performance on real scientific agent requests. Published online 2026. doi:[10.48550/arxiv.2608.21601](https://doi.org/10.48550/arxiv.2608.21601)
3. Kassis T, Agarwal V, He Y, Patel D, Brueckner AM. Scientific Agent Skills: A Library of Procedural Knowledge for Research Agents. Published online 2026. doi:[10.48550/arxiv.2609.00065](https://doi.org/10.48550/arxiv.2609.00065)
4. Kapoor S, Stroebl B, Siegel ZS, Nadgir N, Narayanan A. AI Agents That Matter. Published online 2024. doi:[10.48550/arxiv.2407.01502](https://doi.org/10.48550/arxiv.2407.01502)