Day 1 — August 27, Aarhus

Practical AI for Academics

Day 1 — an introduction to Generative AI

Claes Bäckman

Leibniz Institute for Financial Research SAFE

Let's start with an example – building a website for a paper

Three messages: AI is here, the workflow matters more than the model, and verification is key.

1 The landscape has shifted

AI tools have moved from **chat** to **reasoning** to **agentic** systems that read files, run code, and iterate.

Implication: tools you tried in 2023 are not the same tools today.

2 Value is in the workflow

The agent's leverage comes from **context** (your files), **standing instructions** (`CLAUDE.md`), and **reusable skills** — not from raw model IQ.

Implication: a one-hour setup compounds.

3 Verification is the new skill

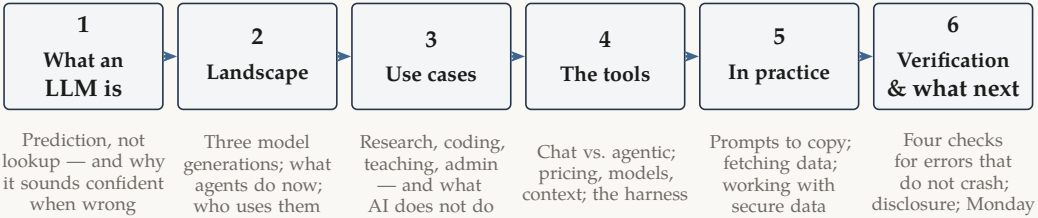
Output volume is no longer the constraint — **judging it** is. Silent errors and confident statements with no backing are what you should look out for.

Implication: the researcher's edge is critical reading

Source: Workshop synthesis based on my work, Mollick (2026), Velikov (2026), Korinek (2025).

Key message today: **Context improves AI tools tremendously, and the goal is to provide the right context**

Day 1 agenda



This is more than three hours of material. We stop where we stop, and pick it up tomorrow morning.

Tomorrow (Day 2). Hands-on with your own materials: a referee report on your draft, a `voice.md`, admin tasks, slides and websites.

Slides, skills, and guides for both days: `claesbackman.com/aarhus`
Guides: AI guides, Claude Code vs. Codex, VS Code setup, researcher tooling, verification, linking to Overleaf.

Who has used ChatGPT or Claude inside an IDE or terminal? Who has used Claude Code or the Claude desktop app?

Section 1

What an LLM actually is

What is an LLM, really?

A large language model is trained **predict the next token**, over and over, very well.

AI tools are prediction machines, not lookup machines.

- It is **not** a database — it does not look facts up, it generates plausible text
- It is **not** a calculator — it predicts what an answer *looks like*
- It is **not** connected to your data or the live web (unless the tool adds that)

A useful mental model

A very well-read, very fast colleague who has read almost everything but remembers none of it precisely, has no access to your files, will never admit to not knowing, and is unapologetic about being wrong.

Claude-powered AI agent's confession after deleting a firm's entire database: 'I violated every principle I was given'

Claude-powered AI agent's confession after deleting a firm's entire database: 'I violated every principle I was given'

AI agent hacks gym to get its user a spot in pilates class

Why it sounds so confident when it's wrong

The model is trained to produce **fluent, plausible** text based on **the context it has available**.

Imagine that you are asking an LLM about a paper. If the model does not have the paper available, it may come up with something based on the training data

- It is not looking anything up (unless it has a search tool). It manufactures (predicts) what the paper is likely to say
- If the paper is well known, it may recall real facts but garble the details — coauthors, year, magnitudes
- This is a major source of hallucinations

Why it sounds so confident when it's wrong

The model is trained to produce **fluent, plausible** text based on **the context it has available**.

Imagine that you are asking an LLM about a paper. If the model does not have the paper available, it may come up with something based on the training data

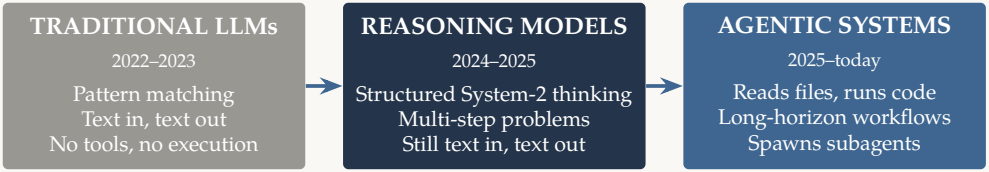
- It is not looking anything up (unless it has a search tool). It manufactures (predicts) what the paper is likely to say
- If the paper is well known, it may recall real facts but garble the details — coauthors, year, magnitudes
- This is a major source of hallucinations

If you instead provide the paper in a readable form, the model reads the text and makes predictions based on **the context it has available — the paper**.

Section 2

The landscape

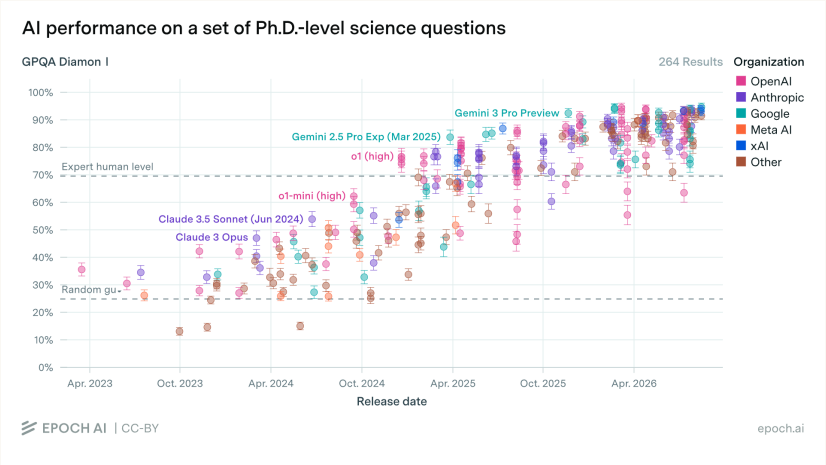
Three generations of AI models



Key takeaway. *A 2023 mental model ("ChatGPT is a chatbot that hallucinates") no longer describes what the tools can do.*

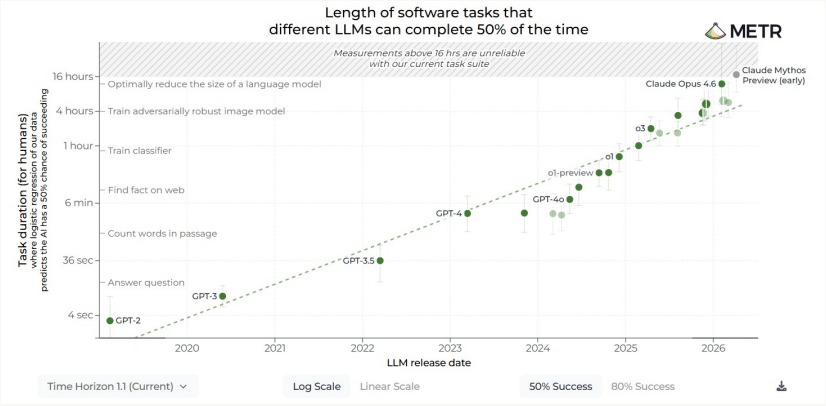
Source: Mollick (2026); Korinek (2025); Velikov (2026).

GPQA Diamond - AI performance on a set of Ph.D.-level science questions



Source: Mollick (2026); Korinek (2025); Velikov (2026).

Exponential progress

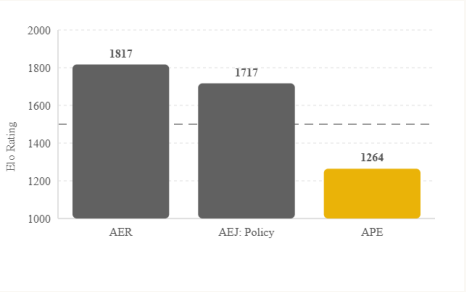


Source: Mollick (2026); Korinek (2025); Velikov (2026).

The pace is incredible

- Novy-Marx & Velikov (2026) describe how you could automate academic research using LLMs
- **Paul Novosad** wrote a paper with AI in ~3 hours, and then another one with the opposite result
- **Project APE** has automated 1,000 policy-evaluation papers
- **López-Lira** has developed system that discovers a problem, generates a theory, verifies it adversarially, and writes a publication-ready paper.

Papers are not that bad on average



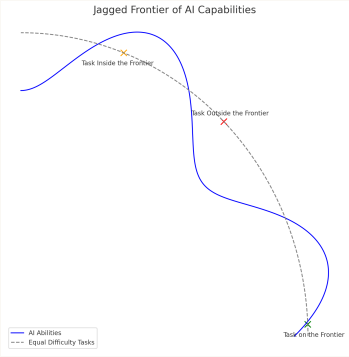
ELO ratings of papers by source.

Source: Mollick (2026); Korinek (2025); Velikov (2026).

Agents are world-class at some tasks, but not at all of them

The jagged frontier: Ethan Mollick, Dell'Acqua *et al.* (2026)

- **Inside the frontier:** draft prose, code, lit-review summaries, structured critique, table re-derivation
- **Outside the frontier (?):** anything where the answer is not implicit in the training data, like judgment calls on how to frame a paper



The frontier is uneven — map it before you trust it.

Source: Mollick (2026); Korinek (2025); Velikov (2026).

AI is especially good at some parts of the work

Coding is largely solved – Boris Cherny, head of Claude Code at Anthropic

Source: Mollick (2026); Korinek (2025); Velikov (2026).

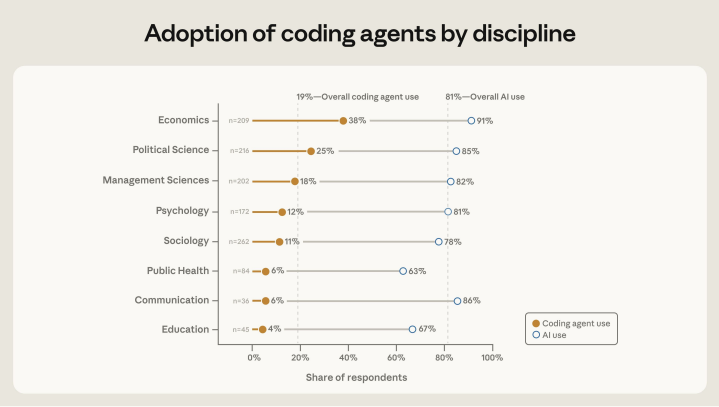
Seemingly also for econometric coding

> We study how large language models write code for econometric analysis. We compare three dimensions of AI-assisted coding: statistical software (Stata, R, or Python), prompting (zero-shot versus few-shot), and the degree of agency, from a chatbot that writes a single script to an agent that executes and revises its own code. On a benchmark of applied econometric and statistical tasks, moving from the chatbot to the constrained agent raises task success from 74 to 96 percent, at about eight additional cents per run. For Claude Sonnet 4.6 and GPT-5.4 through Codex, few-shot prompting improves the chatbot far more than the constrained agent, indicating that prompting and agency act as substitutes. For these models, differences across statistical software are sizeable under the chatbot but largely disappear under the constrained agent.

AI Agents and Prompt Engineering in Econometric Coding, Galiani *et al.* (2026), NBER working paper 35588,

Source: Mollick (2026); Korinek (2025); Velikov (2026).

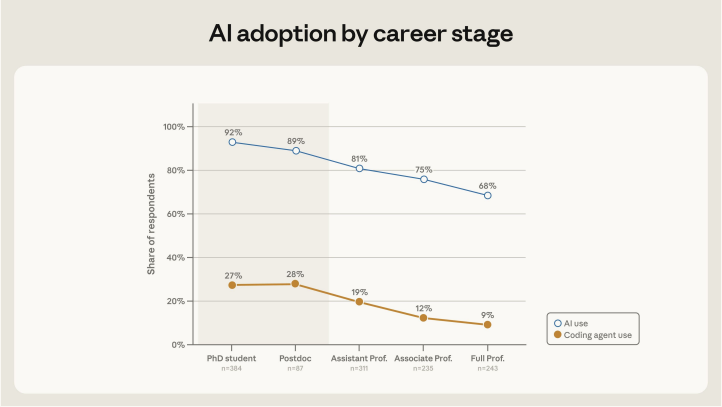
AI adoption of chatbots is high, but not adoption of coding agents



(Lyttelton *et al.* , 2026), survey of 1,260 social scientists about AI and coding agent use, fielded in February and March 2026.

Source: Mollick (2026); Korinek (2025); Velikov (2026).

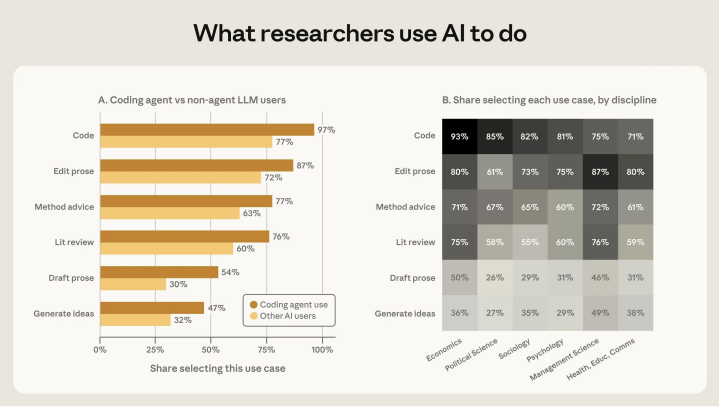
Adoption also differs by career stage



(Lyttelton *et al.* , 2026), survey of 1,260 social scientists about AI and coding agent use, fielded in February and March 2026.

Source: Mollick (2026); Korinek (2025); Velikov (2026).

Common use case is coding and writing



(Lyttelton *et al.* , 2026), survey of 1,260 social scientists about AI and coding agent use, fielded in February and March 2026.

Source: Mollick (2026); Korinek (2025); Velikov (2026).

Section 3

Use cases for AI

Agentic AI pays off across three distinct use cases

1 Research

- Lit-review summaries
- Pre-submission referee reports
- Feedback on your own drafts
- Responses to referees

Mental model: *AI as a feedback machine.*

2 Coding

- Stata / R / Python scripts
- Debugging error logs
- Robustness checks
- Replication packages

Mental model: *AI as a talented RA.*

3 Practical / admin

- Update slides and materials
- Mock exams/problem sets
- Conference budgets, CVs
- Create websites

Mental model: *AI as a personal assistant.*

Admin is an undervalued example

Agents are great at reducing cost of administrative tasks

- **Friction is highest** — the starting cost dominates the doing
- **Quality bar is lower** — no referee, no audience
- **Nobody else to delegate to** — Small tasks not worth giving to someone else

Example — a CV for research grants

Different funders, different required formats.

Without an agent. Reformat, re-order, drop sections, rewrite the bio.

With an agent. Drop the master CV and the funder's template in the folder. The agent reformats, you check.

You can save time on a task which is easy to verify and that no one pays you for.

Resources are unequally distributed

The feedback gap

- ~40% of NBER papers acknowledge RAs (Caro, 2024); among papers with RAs, the average is over 3
- Elite departments run internal seminars, lunch talks, drafts circulated to senior faculty
- Most researchers do not have access to that infrastructure
- Otherwise, feedback comes from referees — slow, sparse, end-of-process

What AI changes

- The same critic available for $20/month, anywhere
- Pre-submission referee reports, identification critique, line-by-line edits on demand
- A more equal access to quick and relevant feedback

AI is going to change how we do research

The obvious gains

Same research, done faster and better.

- **Faster coding** — Stata/R/Python drafts, debugging error logs, robustness checks, replication packages
- **Better writing** — structured critique, voice-matched edits, pre-submission referee reports

New ways of working

New outputs and projects that were infeasible before.

- **Different ways of communicating research** — interactive websites, policy briefs, dashboards, audio summaries from one paper
- **Different types of projects** — mass automation (Project APE), automated pipelines (López-Lira), text-heavy work that used to be too costly

AI rewards expertise!

1. **Better questions.** Being an expert means that you ask better questions
2. **Steering.** Being an expert makes it easier to ask follow up questions and steer the conversation where you want it to go
3. **Validation.** Being an expert makes it easier to validate the output you get

Three things AI does not do

1. **It does not supply the question.** It lowers the cost of asking a question seriously, but it does not tell you which question is worth asking.
2. **It does not replace judgment.** You need to be in the loop to decide what to do and whether it is worth doing.
3. **It doesn't know about institutional context.** It will write a referee report on a paper if asked, but it will not know the journal policy on AI use.

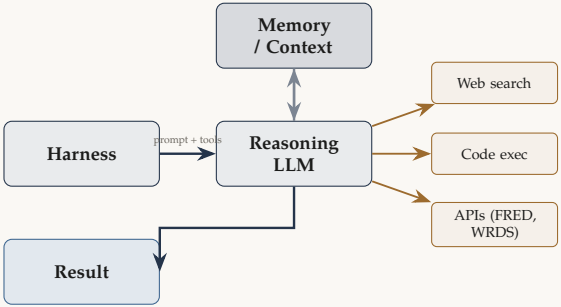
KEY TAKEAWAY. *AI are highly capable tools that can answer any question confidently, but you are still in charge of the output.*

Let's check in on the website

Section 4

How the tools work

The agent is the harness around the model



What happens when you type a prompt

1. Your prompt goes to the model with `CLAUDE.md`, recent files, and history
2. The model decides which **tools** to call — read, edit, shell, web
3. Each tool returns a result back into the conversation
4. The loop continues until the task is done — or the agent asks you a question

---

KEY TAKEAWAY. *ChatGPT, Claude.ai, Claude Code, Codex, and Cursor are all harnesses around **similar** models. The harness is the source of capability.*

Source: Korinek (2025); figure adapted from Velikov (2026).

Models like Claude Code work through six core capabilities

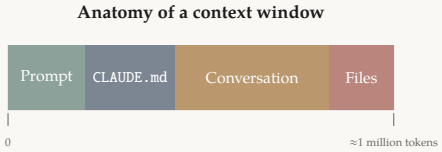
Capability	What it lets the agent do
Project files (`Read`)	Read files, inspect folders, understand project structure
Editing (`Edit`, `Write`)	Propose and apply diffs; create or modify files
Shell (`Bash`)	Run commands such as `Rscript`, `pdflatex`, tests, linters
Search (`Grep`, `Glob`)	Search filenames and file contents in the project
Web (`WebSearch`)	Search or fetch web pages when web access is enabled
MCP / plugins	Connect to external tools: GitHub, databases, custom servers

Source: Korinek (2025); figure adapted from Velikov (2026).

Context is the budget that you have available

Habits that protect it

- Point at specific files with `@filename` — don't let Claude read everything
- Start a new session for a new task (`/clear` or the + button)
- Watch the context indicator — `/context` shows the breakdown
- Convert long PDFs to markdown once, up front



Source: Korinek (2025); figure adapted from Velikov (2026).

Five prompting rules: specific, iterate, correct early, verify, restart.

1. **Specific.** “Feedback on my paper” is a worse request than “Give me feedback on the identification strategy in Section 4.”
2. **Iterate.** 10–20 turns, not one.
3. **Correct early.** If the first answer is off, fix it now — or restart the chat.
4. **Trust but verify.** Run the code. Re-read the passage. Check the citation.
5. **Open new sessions.** 30+ turns or the task has shifted — `/clear` and restart.

Source: Korinek (2025); figure adapted from Velikov (2026).

Four failure modes — flattery, generic feedback, fabrication, drift.

Failure mode	Symptom	Fix
Flattery	*"This is a well-structured piece."*	Ignore. Ask for the strongest objection a hostile referee would raise.
Generic feedback	*"Consider strengthening the motivation."*	Push back: "Give me a specific sentence that would make it stronger."
Confident fabrication	Claude describes a passage that is not in the draft, or cites a paper that does not exist.	Always re-read the passage. Always check the citation.
Session drift	Repeats mistakes, invents files, gives circular answers, forgets the last three turns.	`/clear` and start fresh. Cost: 5 seconds.

Source: Korinek (2025); figure adapted from Velikov (2026).

File formats fall into three buckets

Native — no friction	`.md`, `.tex`, `.txt`, `.bib`, `.csv`, `.json`, `.yaml`
Code — native	`.py`, `.R`, `.do`, `.ipynb`
Usable — convert first	`.docx`, born-digital `.pdf`, `.xlsx`
Hard — OCR or skip	Scanned PDFs, `.dta` / `.rds` binaries, `.qsf`

KEY TAKEAWAY. *Convert once, commit the markdown alongside the original, point the agent at the markdown. Re-converting on every session is the single biggest waste of context.*

Source: Korinek (2025); figure adapted from Velikov (2026).

Converting files is as easy as asking Claude to do it

Section 5

Choosing your tool

Which provider to use?

The "leading" provider seem to differ month by month as new models are released

- Claude was ahead, now ChatGPT seem to be back in business
- Gemini is Google's version, Grok is from xAI, now part of SpaceX (who owns X and Cursor), Llama is the model from Meta; all are competing and improving

Open-weight models are cheaper to run

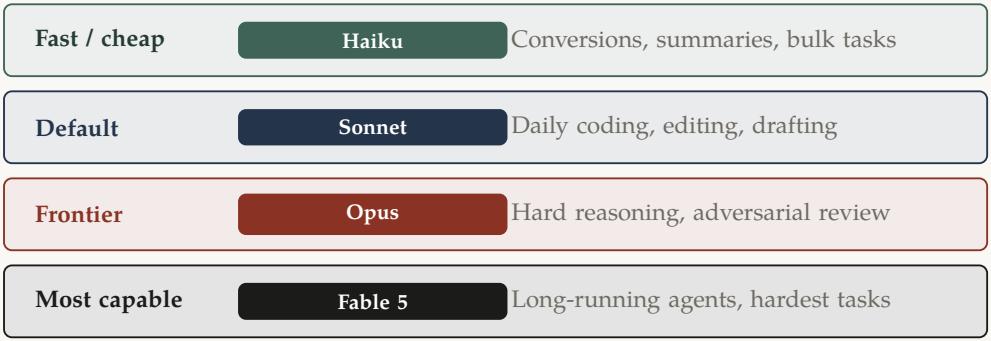
- Many open-weight models are not that far behind on performance
- Mistral is an European model, which may be good for regulatory compliance

In the end, switching costs are low and it is more important to just start.

Three pricing tiers for Claude

Free	**$0**	Casual chat on claude.ai; no real Claude Code usage
Pro	**$20 / mo**	Light daily Claude Code use, projects, research ▶ Start here
Max	**$100–200 / mo**	Heavy Claude Code use (5x / 20x); the frontier model all day
API	**Pay per token**	Automation, batching, custom-built tooling

Four model tiers



You can also choose effort level

Effort determines how many tokens Claude uses when responding

- Trade off between response thoroughness and token efficiency
- You can adjust the effort parameter up when you need more thorough reasoning and deeper analysis.
- Use low effort if you want to have more speed

Tokens are the unit models bill and reason in — about 3–4 characters of English. A short paper is ≈15–30k tokens. Higher effort means more tokens used.

A structural difference between chatbots and agentic tools

Chat: ChatGPT, Claude.ai

Runs in a sandbox.

- You describe the problem in words
- Paste text in, get text out
- No access to your files or your code
- Output is generic — surface-level

When to use. Brainstorming, draft polish on a single paragraph, quick lookups.

Agentic: Claude Code, Codex

Runs on your computer.

- Reads every file in your project folder
- Edits files, runs code, calls shell commands
- Calls tools: search, web, MCP servers
- Output is grounded in **your** files

When to use. Any task that touches more than one file or needs to execute code.

... but the difference between chat and agentic AI getting smaller

Claude has Cowork and Claude Code in the main Claude app

ChatGPT app includes Codex, the agentic tool akin to Claude Code

And much of the workflow is converging between chat and agentic AI tools

Claude and ChatGPT have desktop apps for Mac and Windows

You open the app and point it at a folder — the *Code* tab is Claude Code, the *Cowork* tab is for non-coding work

The agent has the same context and does the same tasks as in e.g. VS Code

Drawback is that you cannot see what the edits are as easily

Key takeaway. *Switching costs are small once you understand the pattern. These two days use **Claude Code**, but everything transfers.*

Claude and ChatGPT both have "projects", which are closer to agentic AI tools

With projects you can provide context to Claude by uploading files

You can put in instructions that the models will always follow

Each project will have a "Memory"

The key is that context and memory makes AI work better – use whatever tool you like

You can use whatever program you like

Surface	What it is good for
VS Code	**What I use.** File tree, diff view, integrated terminal, side-by-side PDF preview.
Terminal / CLI	`claude` (or `codex`) in any shell. Good for servers and quick one-shots.
Desktop app	Claude Desktop *Cowork* / *Code* tabs, or the Codex desktop app.
Web / cloud	claude.ai/code or chatgpt.com/codex. Point at a GitHub repo.
Mobile	Capture ideas; monitor or redirect cloud agents. Not for editing.
Other editors	Cursor, JetBrains, Zed — keep your editor, add the agent.

More details at claesbackman.com/agentic-ai-overview.

Claude Code vs. Codex: the workflow is the same.

Both will: read your files, run code, edit drafts, iterate, spawn subagents.

Claude Code (Anthropic)

Project memory: `CLAUDE.md`
Skills trigger: `/review-paper`
Backed by: Opus / Sonnet / Haiku
Surfaces: VS Code, CLI, web, desktop

Codex (OpenAI)

Project memory: `AGENTS.md`
Skills trigger: `$review-paper`
Backed by: GPT-5 family
Surfaces: VS Code, CLI, web, mobile

Key message: **Context improves AI tools tremendously, and the goal is to provide the right context**

Key difference: **how you provide the useful context and how easy it is**

Quick example of the difference - a conference budget

Section 6

What you can use AI for

Different use cases

Explain a paper and how it is relevant for your work

Let agents fetch data from public APIs

Write code for a paper

Update slides and materials for teaching

Help with writing and editing

Make mock exams or problem sets (students might already be doing this)

And more?

Two directions: ask it to explain and then ask it to implement.

Explain

A paper. *"Explain the identification strategy in Section 4 to a second-year PhD student. Which assumption does it rest on, and where could it fail?"*

Code. *"Walk through this do-file block by block: what does each merge do, and where could observations be dropped?"*

Instructions. *"Here are the journal's submission guidelines. Give me a checklist of what my manuscript must satisfy and flag what is missing."* Same for grant calls and data-access rules.

Also: error messages, referee letters — *"what is this actually asking for?"*

Implement

From the paper. *"Implement the estimator in Section 4 in Stata, following the paper's notation. Ask me about anything the paper leaves ambiguous before you write code."*

From the instructions. *"Reformat the manuscript to these guidelines and list every change you made."*

Chat vs. agent. In a chatbot you paste the code into Stata and run it yourself. An agent runs it and shows you the output.

Explain first, then implement: you check the code against an explanation you already understood.

Let the agent fetch the data — public APIs such as Statistics Denmark's StatBank.

What an API is. A URL that returns data as CSV or JSON instead of a web page.

Statistics Denmark. `api.statbank.dk` list tables, read a table's variables and codes, download the data.

Same pattern elsewhere. Eurostat, ECB, FRED, World Bank, OECD, Nationalbanken.

Verify. Agents invent table and variable codes. You need to verify, e.g. compare numbers with the website or more thorough checks.

Have the model write code to fetch the data

Prompt

"Using the Statistics Denmark API, download table FOLK1A — population by municipality — for every quarter from 2010 onwards, all 98 municipalities, totals over sex and age. Look up the table's variables and codes first and show me what you plan to request before downloading.
Save the result as `data/raw/folk1a.csv` *and write* `data/README.md` *describing the variables and codes. Use Python.*
Then report the number of municipalities and quarters you got, and the population of Aarhus in 2020Q1 so I can check it."

Update or create teaching materials based on your own course

Updating slides. *"These slides use 2015 examples. Suggest current replacements — recent policy episodes, updated figures to look up — keeping the structure."*

Explaining hard concepts. *"Three ways to explain instrumental variables: intuitive, graphical, formal."*

Slides from a source. *"Draft a 12-slide lecture outline on money demand for a second-year macro course based on Chapter 12 in @course-book, with a worked example on each concept."*

Mock exams or problem sets

Practice problems. *"Five problems on consumer choice at increasing difficulty, with full worked solutions, based on slides in @chapter6-slides.tex."*

Variants. *"Three numerically different versions of this problem, to rotate across years."*

Coverage and timing. *"Map each question to a learning objective. Which syllabus topics are not tested? How long does a prepared student need?"*

KEY TAKEAWAY. ***Verify every number.*** *A next-token predictor is not a calculator: solve every quantitative problem yourself, or have it write code to check.*

Create new ways of doing outreach

Ask for a website. Claude is great at making interactive websites that can showcase your paper.

Policy briefs or blog posts. Get a quick draft of a shorter version of your paper based on your own writing and edit it.

Social media posts. Draft a LinkedIn post to promote your paper.

All will require editing, but can easily be based on your own writing to speed up outreach activities.

Section 7

Working with secure data

You should never paste any sensitive data or results from DST servers into Claude

Files or chats that Claude read go to Anthropic servers

Confidentiality and data

- **Never paste** confidential microdata, register data, or unpublished co-authored work.
- **"Training on your data."** Some consumer tiers may use your chats by default. Know the setting; prefer business/team tiers for anything sensitive.
- **When in doubt, abstract it.** Ask about a made-up example or use a synthetic dataset.
- **Check institutional policy** before putting any work material into any AI tool.

Secure servers split the workflow

What Claude can still help with on your laptop

- Your `.do` files
- Writing and iterating on empirical strategy

Claude cannot run the files or read the logs on the server, but can still be very helpful in a number of ways

Two artifacts to keep local

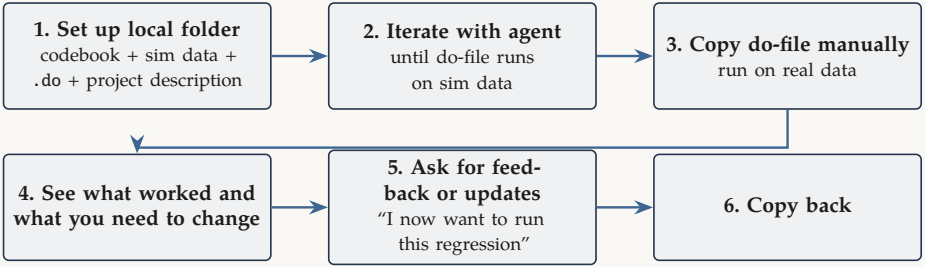
1 Codebook in markdown

- Convert once (`/pdf-to-markdown`)
- Variable names, labels, value ranges, missing-value codes
- Agent reads it before writing `recode`, `label define`, `merge`

2 Simulated dataset

- Same variable names, types, ranges — not real values
- AER/JF/ReStud replication packages often ship with these
- Or generate from the codebook

A workflow for secure data on servers



KEY TAKEAWAY. *The agent never sees the real data* ***(and you should never paste any result from the server into Claude)****— but it sees everything that explains the data. That is usually enough to write correct Stata code.*

Section 8

Verifying LLM output

Verification is the new bottleneck — you are looking for errors that do not crash.

Agents are fast, cheap, and competent at code, including in languages you do not know.

- They are able to produce a paper in 15 minutes and a new codebase in 20
- Or a lot more material if you allow it to run for longer. The sky is the limit

Verification is the new bottleneck — you are looking for errors that do not crash.

Agents are fast, cheap, and competent at code, including in languages you do not know.

- They are able to produce a paper in 15 minutes and a new codebase in 20
- Or a lot more material if you allow it to run for longer. The sky is the limit

But you are responsible for the output, and so you have to make sure that it is right

Verification is the new bottleneck — you are looking for errors that do not crash.

Agents are fast, cheap, and competent at code, including in languages you do not know.

- They are able to produce a paper in 15 minutes and a new codebase in 20
- Or a lot more material if you allow it to run for longer. The sky is the limit

But you are responsible for the output, and so you have to make sure that it is right

The errors that matter are not code crashes. Agents are very good at finding and fixing what crashes. What you get back is a table that runs and looks plausible — and may be wrong because the agent made the wrong decision or used the wrong fixed effect.

Source: claesbackman.com/verifying-llm-output.html; Litt (2026); Goldsmith-Pinkham (2026); Scott Cunningham.

First principle: you have to read the code and the writing

The code is the analysis, and you are responsible once a result is in the paper.

- But you can automate some checks
- And not all code is created equal – data cleaning and regressions require more oversight than website code

Git makes this easier

Focusing on what **changed in the code** reduces the burden somewhat

`git diff` shows what changed, for you and for the agent.

- An agent that sees the whole codebase spreads its attention thin, gives generic answers, and costs tokens.
- A reviewer that sees a diff knows exactly what is new and compares it against code you already checked

Four checks that you can run

Check	What you do	What it catches	Cost
Fresh agent attacks the diff	Fresh chat or subagent reviews the diff.	Errors invisible to the agent that wrote the code	Minutes
Different model	Same diff, model from another developer (Codex vs. Claude)	Shared habits: two copies of one model make the same mistakes	Minutes
Make the agent quiz you	Fresh session explains the change, then quizzes you	Your own understanding gap	10–15 min
Reimplement in another language	Fresh folder, `spec.md` + `README.md` only, other language and model, you run it	Code that faithfully implements something other than the paper	Hours

Source: claesbackman.com/verifying-llm-output.html; Litt (2026); Goldsmith-Pinkham (2026); Scott Cunningham.

Checks 1: a fresh agent has no stake in the code

Two ways to get independence:

- Ask the current agent: *"Use a subagent to review the code . . . "*
- Open a new chat and point it at the diff

Have the reviewer report, not fix. A reviewer that edits hands you a second diff to verify. Confident false positives are common: make it prove the findings that matter, then fix those yourself.

Check 2: Use a different model

Two instances of the same model share training data and habits of thought, so they tend to make the same mistakes.

Use a model from a different develop to break that dependency

Check 3: make the agent quiz you

The thirty-second version. Open a fresh session and ask:

> *"Read the last commit and the files it touches, then ask me three questions about what it changed. Do not tell me the answers until I have answered. If my answer is wrong, say so directly — do not tell me I was close."*

Always a new session, not the one that made the change. Starting fresh forces the agent to read the files instead of recalling the conversation.

A skill: `/explain-diff`

`/explain-diff` checks what changed within a Github repo and generates a html site that you can open

- **Verification.** It diffs generated tables, compares N and headline coefficients across logs, and quotes numbers only from output files it has opened.
- **Output.** One offline HTML page in five sections: what the code did before, what changed and why, consequences for the results, a code walkthrough grouped by purpose, and a five-question quiz. Saved outside the repo with a plain-text summary in the chat.

Check 4: reimplement the analysis in another language (This is an idea from Scott Cunningham)

What to do. If you write code in Stata, have the model code up results in Stata or R.

Why it is strong. If errors across languages are independent, differences in the output reveal them.

Isolate the second implementation. Enforce independence: new folder, fresh session, a `spec.md` with the specification copied from the paper (results deleted), and a `README.md` describing the raw data.

Reserve it for the main tables and for steps where a mistake is invisible in the output: a panel construction, an event-time definition, an index built from several sources.

Section 9

Implications and getting started

What this means for academic work

- **Friction drops an order of magnitude** in writing, editing, and admin
- **The bottleneck shifts** from producing to judging
- **Verification becomes the differentiator** — not typing speed, not LaTeX fluency
- **Disclosure norms are forming** — expect them to harden over the next year
- **Tooling will keep moving** — the principles (specificity, iteration, verification) will not

---

Key takeaway. *The researchers who get the most out of these tools are not the most technical. They are the ones who treat AI as a colleague to argue with.*

Source: Workshop synthesis; verification checklist adapted from Korinek (2025).

Verification is going to be increasingly important (and in the end also automated?)

Check	What to look for	How to do it
Run the code	Claude writes plausible Stata / R / Python that does not always run.	Run the code yourself.
Check the numbers	Statistics, dates, quotes get invented.	Re-derive from the table or paper.
Read the citations	Citations to papers that do not exist are common.	Check if you can find it on Google Scholar.
Re-read the passage	Claude confidently describes text not in the draft.	Open the file, read the section in question.

KEY TAKEAWAY. *You are faster with AI, not absent.*

Source: Workshop synthesis; verification checklist adapted from Korinek (2025).

Separate publishing from research quality

Publishing gets noisier

- Journals fill with polished, competent-looking mediocrity
- Signal-to-noise drops and reviewing gets harder
- Editors and referees absorb the cost

Research can still get better

- Ideas that could not be written before reach readers
- The language barrier falls
- Feedback, formerly a privilege of elite departments, becomes universal

Key takeaway. *The open question is whether we update how we **evaluate** research now that competent-looking writing is free. Because research is about to get better.*

Source: Workshop synthesis; verification checklist adapted from Korinek (2025).

Disclosure and data norms are still in flux.

Disclosure

- **Journals** mostly require acknowledging AI assistance but specifics vary
- **Policy briefs, teaching, internal docs** have no strong norm yet (?)
- **Default rule:** disclose where your name is on it

Data handling

- Agentic tools run *locally*, but prompts and file excerpts may be sent to the model provider
- **Do not** paste confidential microdata, HR-sensitive material, or DUA-covered material into any AI tool without checking institutional policy
- Web chat ≠ local agent — different tools, different policies

KEY TAKEAWAY. *Check before pasting. Once is enough — the policy almost never changes mid-project.*

Source: Workshop synthesis; verification checklist adapted from Korinek (2025).

End of Day 1

Tomorrow: hands-on with your own materials. Bring a draft you care about and 4–5 PDFs of your own writing.

Slides, skills, guides: `claesbackman.com/aarhus`

Appendix: `CLAUDE.md` starter template

```
# About me
I am a researcher in [field].

# How I want you to work with me
- Ask clarifying questions before generating long output.
- Critical, skeptical tone in feedback. Do not flatter.
- When editing, preserve my voice (see voice.md). No generic LLM phrasing.
- Avoid bullet lists and passive voice in formal writing.
- Cite the specific line or section you are commenting on.

# Things to avoid
- Do not fabricate citations.
- Do not over-claim causality.
- Do not insert emoji or markdown decorations in formal docs.
```

Appendix: My skill library

Skill	Use
`/review-paper`	Pre-submission referee report, 8 agents
`/review-paper-light`	Fast 2-agent version, ≈ 1 minute
`/review-paper-code`	Paper + code alignment review
`/review-grant`	Pre-submission grant review, 6 agents
`/review-pap`	Pre-analysis-plan review
`/audit-analysis`	Adversarial audit of changed analysis code
`/explain-diff`	Explain a code change as an HTML page with a quiz
`/paper-version`	Paper → policy brief / 1-page / 5-page
`/pdf-to-markdown`	Convert PDFs to readable markdown

All at github.com/claesbackman/AI-research-feedback (MIT), linked from `claesbackman.com/aarhus`. Fork them, modify them, make them yours.

Appendix: signs of session drift

- Claude ignores `CLAUDE.md`
- Repeats mistakes you already corrected
- Invents files that do not exist
- Gives circular answers
- Forgets the last three turns

Fix. Start a fresh session or close and reopen. Re-state the task. Re-attach the relevant file.

30+ turns or the task has shifted → start fresh. Always.

References I

Dell'Acqua, Fabrizio, McFowland III, Edward, Mollick, Ethan, Lifshitz, Hila, Kellogg, Katherine C, Rajendran, Saran, Krayer, Lisa, Candelon, François, & Lakhani, Karim R. 2026. Navigating the jagged technological frontier: Field experimental evidence of the effects of artificial intelligence on knowledge worker productivity and quality. *Organization Science*, **37**(2), 403–423.

Galiani, Sebastian, López, Federico Ariel, & Sosa, Raul A. 2026. *AI Agents and Prompt Engineering in Econometric Coding*. Tech. rept. National Bureau of Economic Research.

Lyttelton, Thomas, Massenkoff, Maxim, & Wilmers, Nathan. 2026. *Coding Agents in the Social Sciences*. Anthropic Research.

Novy-Marx, Robert, & Velikov, Mihail. 2026. Artificial intelligence–powered (finance) scholarship. *Journal of Economic Literature*, **64**(1), 5–37.