Claude Code for Accounting Research

Module 5 — Writing, Safety, and the Verification Capstone

John Barrios

Yale School of Management

Today's Arc

- **0:00–0:15** Mini-lecture: writing as thinking, the banal/meaningful line
- **0:15–0:40** Demos: editor-not-rewriter, style skills, referee-DAG
- **0:40–1:05** Mini-lecture: verification debt, sandboxes, credential hygiene, paper trails
- **1:05–1:15** Demo: git as the operating system
- **1:15–1:25** Break
- **1:25–2:45** Capstone lab: harden a Module 3 or Module 4 pipeline
- **2:45–3:05** Debrief + exit ticket + Module 6 handout

Writing as Thinking

- The sentence you can't quite make work usually stands in for an argument you haven't worked out yet
- Vague prompt $\rightarrow$ model does the intellectual work $\rightarrow$ you adopt it — the paragraph reads fine; the thinking never happened in your head
- **The test:** can you discuss it, out loud, without the document or the model in front of you?
- Rushed writing handed to an LLM comes back undifferentiated — a tired, vague prompt is exactly its generic default

Common Failure

Mistaking fluency for having thought it through

A paragraph that reads smoothly is not evidence the argument inside it is yours. “This sounds right” and “I could defend this in a seminar” are not the same test.

The Banal/Meaningful Line

Banal — delegate

- Variable-definition tables
- Data appendices
- Table notes, robustness headers
- Content already determined by the data

Meaningful — protect

- Identification argument
- Statement of contribution
- Limitations section
- The argument itself, not a transcription of one

The line is yours to draw. Commit to one rule: one category you always delegate, one you never let an agent originate.

Common Failure

Treating the line as obvious in advance

The trap is the middle ground — a robustness-check narration that looks banal but hides a judgment call. If a sentence requires you to *decide* something rather than report something, it's on the meaningful side even if it's short.

Anti-Homogenization

- LLMs raise the floor reliably — and compress voice, flattening distinctive writing toward the same competent, generic register
- House styles (JAR vs. TAR) are real and partially mitigate this — but don't eliminate the pull within a journal's own register
- Today's skills are countermeasures, not shortcuts — they exist because the pull is real

Demo 1: Editor, Not Rewriter

Ask for comments, not rewrites — the exact prompt:

```
Please review this Identification section. I want
feedback in the style of a New York Times editor --
sharp on argument and clarity. Do not directly edit
my writing. Instead, insert inline <!-- comments -->
at the specific places where the argument is weak,
unclear, or could be tightened. Do not rewrite any
sentences yourself; leave the prose exactly as
written and let the comments carry the feedback.
```

The two words doing all the work: “do not.” You do the rewriting — which means you do the thinking.

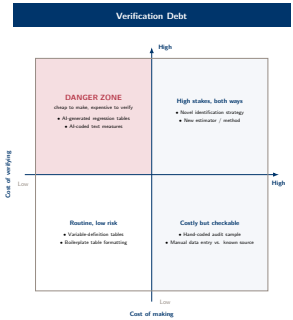
Demo 2: Style Skills as SOPs

- `barrios-voice` — trained on published prose; gets real patterns right, and sometimes reads like a caricature (a style guide is yesterday's you)
- `econ-humanizer` — flags AI-typical tells: inflated significance language, hedge-stacking, the "this paper contributes to the growing literature" sentence
- `latex-tables` — the banal-side workhorse, mentioned in passing
- Module 3's lesson, applied to writing: a skill is an SOP, not a power-up

Demo 3: Referee-DAG

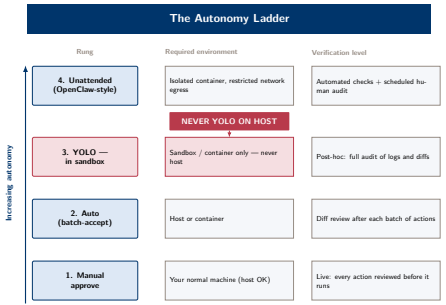
- `strategic-revision` treats four referee reports like a project manager treats a stack of tickets
- Parses comments into tasks, infers blocking dependencies, batches what runs in parallel, flags the critical path and contradictions
- Take the *pattern*, not the machinery — decompose, map dependencies, batch, surface conflicts before you start writing

Verification Debt: the 2×2



Cost to make × cost to verify. Danger zone: cheap to make, costly to verify. More autonomy → debt arrives faster. Exercise: sort two of your own week's tasks into the grid with a partner, three minutes.

Environment First: the Autonomy Ladder



YOLO-in-a-sandbox and YOLO-on-host are different exposure categories, not the same risk scaled down. Proof: `safehouse cat ~/.ssh/id_ed25519` → "Operation not permitted."

Installing a Skill Is Installing Software

- A skill is someone else's instructions running in your environment — read it before installing it, the way you'd read a script before running it on your data
- "It's just a markdown file" does not mean it can't instruct an agent to do something you wouldn't have approved yourself
- A skill from Anthropic or your own group is one thing; a two-star repo from a forum post is another

Common Failure

Treating sandboxing as a substitute for data governance

A container controls what an agent can *do*; it does not control what it can *see* once you mount a folder in, and it does nothing to stop mounted contents from reaching the model. Minimize what gets mounted — the sandbox didn't already handle it.

Credential Hygiene: the Audit

Run from the root of any project before calling it clean:

```
git log --all --full-history -- .env
grep -r "WRDS_PASSWORD" .
```

- Both should return nothing — if either finds something, the credential is compromised
- Fix: **rotate it**, then remove it from history — deleting the current copy alone fixes nothing

Paper Trails: a Map, Not a Verdict

- Requested up front: `DECISIONS.md`, `LOG.md`, sample-attrition table
- An LLM will write a `DECISIONS.md` entry claiming work was done when it wasn't — confident, specific prose, no matching commit anywhere
- Treat every specific claim as a hypothesis to check against the commit history, not a fact to file away

The line to remember

Documentation is a map of what supposedly happened, not a verdict on whether it did.

No Hand-Typed Numbers

- A pipeline script computes a result, writes it to `results.tex`
- The paper's `.tex` never states the number directly — it pulls it in with `\input{results/table2_main.tex}`
- Rerun the pipeline, the paper updates at the next compile — no step where a person copies a number by hand
- An LLM asked to draft a results paragraph will happily invent a plausible number without this constraint

Demo 4: Git as the Operating System

- Five commands and one habit — handout carries the reference; this is the demo
- Live: `git diff HEAD 1` after a real Claude session — a 12-line reviewable diff
- One rollback, on camera
- **A commit is the unit of verification** — tell Claude at project start to commit at every meaningful checkpoint

The Reviewable Diff

capstone — zsh

```
$ git diff HEAD~1 --stat
 clean_fsds.py | 12 +++++-------
 1 file changed, 5 insertions(+), 7 deletions(-)

$ git diff HEAD~1 -- clean_fsds.py
-df = df.dropna()
-df['assets'] = df['at']
+df = df.dropna(subset=['at', 'sale'])
+df['assets'] = df['at'].astype(float)
```

Stylized session transcript (not a screenshot).
A commit is the unit of verification: a 12-line diff you can actually review.

A 12-line commit, with a message that says why the change was made — the unit of verification, on screen.

Into the Capstone

- Choose a Module 3 (EDGAR panel) or Module 4 (WRDS/fallback funda) pipeline
- Harden it against the rubric: reproducibility, verification, hygiene, environment (stretch)
- Before starting: 30 seconds, write down one task you'll always delegate, one you'll never let an agent originate

Capstone Rubric

- **Reproducibility** — clean-clone rerun from the README alone; meaningful commits at every checkpoint
- **Verification** — DECISIONS.md/LOG.md/attrition cross-checked against commits; two sourced sanity checks; one number traced raw data → `results.tex` → output, zero hand-typing
- **Hygiene** — both audit commands run and pasted; `.gitignore` correct; tunnel down; one banal artifact delegated, one meaningful artifact written by hand

Stretch: rerun inside a container, paste the blocked-access proof. Self-graded; instructor spot-checks 3–4 live.

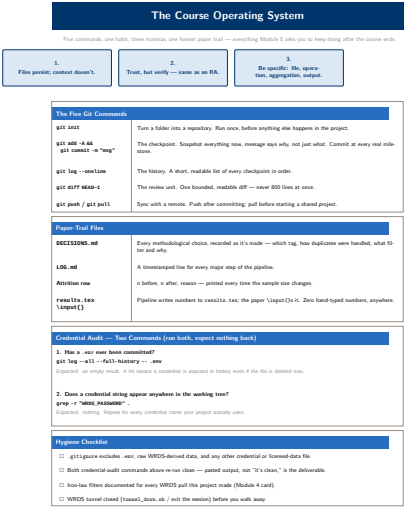
Module 6: Readings, Handed Out Today

- Required: Korinek (2023) excerpts; Bloom, Jones, Van Reenen, Webb (2020) excerpts; Macnamara et al. (2024); *Nature* editorial (2023); Lusher, Yang, Carrell (2023)
- Accounting-specific option: swap in Kim, Muhn, Nikolaev (2024) or Dong, Stratopoulos, Wang (2024)
- Read overnight — position memo prompt: “What I will still be paid for in 2036”
- Debate positions assigned in class

Debrief and Exit Ticket

- Spot-check 3–4 projects on screen — audit one DECISIONS.md claim against the commit log together
- Collect delegation rules: one task always delegated, one never — coauthor-handoff discussion folds in here
- Exit ticket: standard three + your banal/meaningful delegation rule, in writing

The Course Operating System



The Course Operating System

Five commands, one habit, three mantras, one honest paper trail — everything Module 5 asks you to keep doing after the course ends.

1. Files persist; context doesn't.
2. Trust, but verify — same as an RA.
3. Be specific: file, operation, aggregation, output.

The Five Git Commands

`git init`	Turn a folder into a repository. Run once, before anything else happens in the project.
`git add -A && git commit -m "msg"`	The checkpoint. Snapshot everything now, message says *why*, not just what. Commit at every real milestone.
`git log --oneline`	The history. A short, readable list of every checkpoint in order.
`git diff HEAD~1`	The review unit. One bounded, readable diff — never 800 lines at once.
`git push` / `git pull`	Sync with a remote. Push after committing; pull before starting a shared project.

Paper-Trail Files

`DECISIONS.md`	Every methodological choice, recorded as it's made — which tag, how duplicates were handled, what filter and why.
`LOG.md`	A timestamped line for every major step of the pipeline.
`Attrition row`	n before, n after, reason — printed every time the sample size changes.
`results.tex` `\input{}`	Pipeline writes numbers to `results.tex`; the paper `\input{}`s it. Zero hand-typed numbers, anywhere.

Credential Audit — Two Commands (run both, expect nothing back)

1. Has a `.env` ever been committed?

`git log --all --full-history -- .env`

Expected: an empty result. A hit means a credential is exposed in history even if the file is deleted now.

2. Does a credential string appear anywhere in the working tree?

`grep -r "WRDS_PASSWORD" .`

Expected: nothing. Repeat for every credential name your project actually uses.

Hygiene Checklist

- ☐ `.gitignore` excludes `.env`, raw WRDS-derived data, and any other credential or licensed-data file.
- ☐ Both credential-audit commands above re-run clean — pasted output, not "it's clean," in the deliverable.
- ☐ Iron-law filters documented for every WRDS pull this project made (Module 4 card).
- ☐ WRDS tunnel closed (`tunnel_down.sh` / exit the session) before you walk away.