Claude Code for Accounting Research

Module 1 — Foundations: Tools, Context, and the Research Map

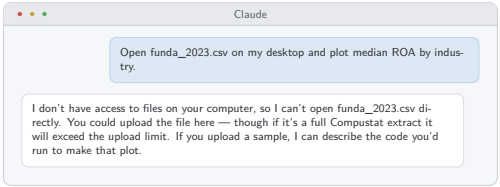
John Barrios

Yale School of Management

Today's Arc

- **0:00–0:40** Lecture: tools, context, verification, research map
- **0:40–1:15** Live demo: one task, start to finish
- **1:15–1:25** Break
- **1:25–2:45** Lab: install, `CLAUDE.md`, one verified task
- **2:45–3:05** Debrief + exit ticket

“AI Can't Do Real Research Work”?

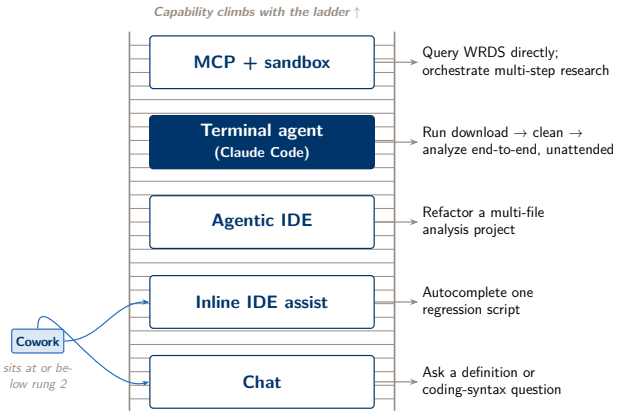


Stylized session transcript (not a screenshot).
Chat-level tool: no filesystem, no execution — rung 1 of the ladder.

Stylized session transcript (not a screenshot).
Terminal agent: reads files, writes a named script, runs it locally — rung 4.

Same task, different rung. Half right — about the tool tried, not the ceiling.

The Tool Ladder



Claude Code sits at rung 4 — terminal, filesystem, real execution.

Common Failure

Dismissing AI after a Cowork-level disappointment

The tool did exactly what it was built to do — it wasn't built to touch your filesystem. Right tool, different rung.

Claude Code as a Terminal-Resident RA

Stylized session transcript (not a screenshot). Anatomy of a Claude Code turn.

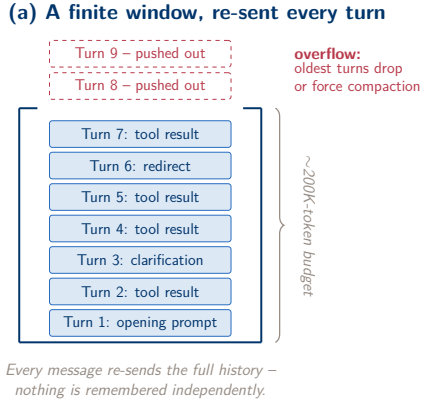
Your existing R/Python/Stata, your files — manage it like an RA, not a search engine.

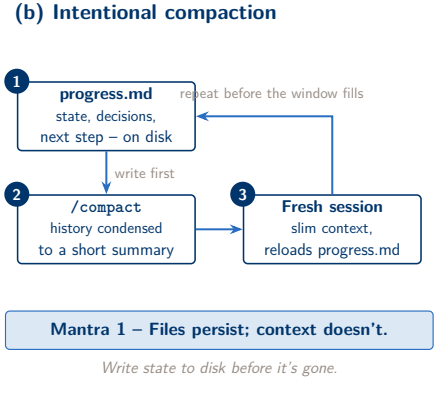
Common Failure

Assuming the runtime is handled for you

No bundled Python or R — it uses whatever is installed locally. “Command not found” means check your environment, not the agent.

Context Window Mechanics





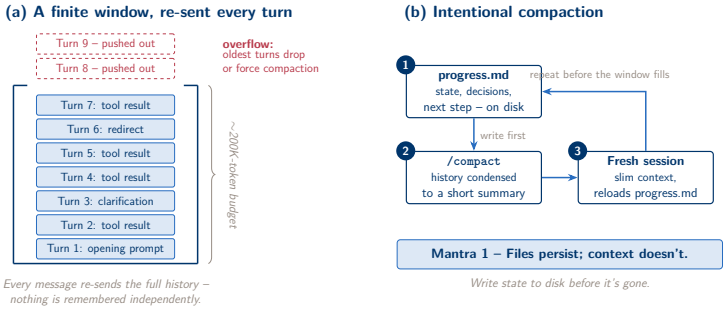
Every turn re-sends the whole conversation so far; ∼200K tokens fills fast.

Common Failure

Importing your Dropbox mental model

Nothing persists across sessions unless it was written to a file a future session can read.

Degradation and Intentional Compaction



Long sessions: the window fills, and quality can drift before it's full.

Mantra 1

Files persist; context doesn't.

Common Failure

Treating compaction as forgetting everything

Compaction condenses; it doesn't erase. Write the file, then compact.

Trust, But Verify --- The RA Standard

- Delegate freely; verify before you build on it
- Same standard you'd apply to a real RA's work
- Pull one case you know well; check it by hand

Verify this

One number, one independent source — every time, before you trust the rest.

Goal-First Prompting: Four Components

- **File** — what you're operating on
- **Operation** — the transformation to perform
- **Aggregation** — the level the result lives at
- **Output** — name and format, in advance

Vague vs. Specific

Vague:

"Look at this Compustat file and tell me something about leverage."

Specific:

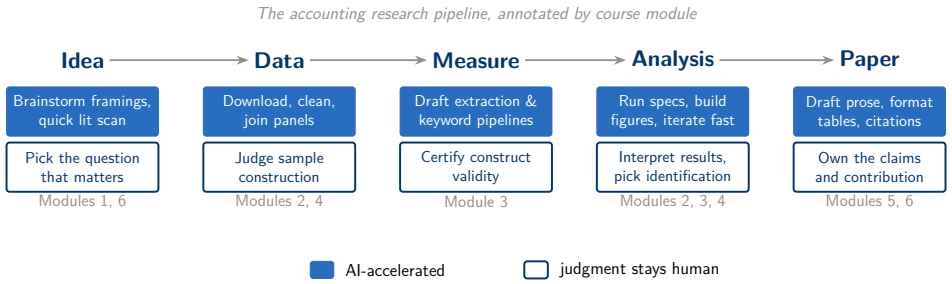
"Read funda_1990_2023.csv. For each gvkey-fiscal year, compute book leverage as long-term debt / total assets, winsorize at 1st/99th pctile, mean by fiscal year x 2-digit SIC. Save leverage_by_industry_year.csv and leverage_trends.png."

Common Failure

Describing the goal instead of the task

"Analyze this and tell me what's interesting" hands the agent every decision. Check file, operation, aggregation, output before you hit enter.

The Accounting Research Workflow Map



idea → data → measure → analysis → paper — some stretches accelerate, some stay human.

The Three Module 1 Mantras

Carry these all week

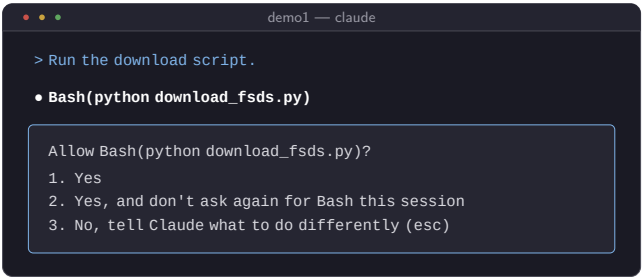
1. Files persist; context doesn't.
2. Trust, but verify — same as an RA.
3. Be specific: file, operation, aggregation, output.

On your verification protocol card. Module 6 asks the harder version of this question.

Next: Live Demo (0:40--1:15)

- One task, start to finish, in a fresh `~/demo1/`
- Launch, permission modes, goal-first prompt (SEC FSDS)
- Watch the loop; redirect, don't argue (Esc)
- Plan Mode preview; `CLAUDE.md`; verify the count

What You'll See: The Permission Ask

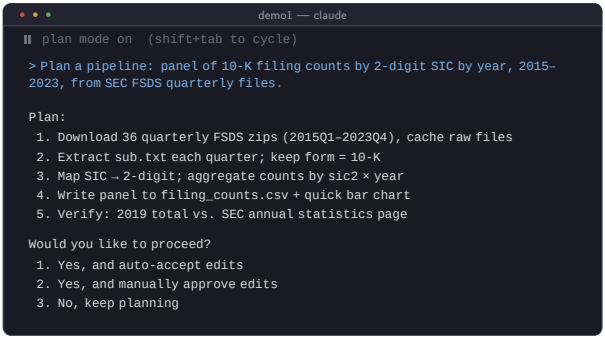


Manual: ask every time — safest while trust is new

Auto: approve the session — pair with git (Module 5)

Stylized session transcript (not a screenshot).

What You'll See: Plan Mode



Stylized session transcript (not a screenshot).
Design before code: review the plan, then execute — or don't.

Next: The Lab (1:25--2:45)

- **Part A (25 min):** environment — `claude` launches, `~/lab1/`
- **Part B (15 min):** write `CLAUDE.md`, restart, confirm it loads
- **Part C (40 min):** tiered task — base / standard / stretch
- Verify checklist: one output checked against an outside source

Debrief, Exit Ticket, and Homework

- Debrief: what broke? Esc or argue? Whose prompt was vaguer than expected?
- Exit ticket: what clicked, what broke, what to verify next time
- **Homework:** confirm your WRDS account + Duo enrollment; test `ssh wrds` before Module 2