Claude Code for Economics RAs

Rules, Checklists, and Skills From My Setup

Ida Grigoryeva

UC San Diego

BFI PREP, University of Chicago
September 23, 2026

The kit for today's tasks
idagri.github.io/files/claude-code-predoc-kit.zip

How the room works today: servers, languages, and Claude

Hands up for each option as I read it

1. Where does your code run?

(a) My laptop (b) A university server or cluster
(c) A secure data enclave (d) Several of these

2. Your main language?

(a) Stata (b) R (c) Python (d) Something else

3. Where do you use Claude today?

(a) The chat window (b) Cowork
(c) The Code tab or the terminal (d) Not yet

4. What do you use it for most?

(a) Writing code (b) Fixing errors
(c) Editing text (d) Literature and citations

5. Where do your PI's drafts live?

(a) Overleaf (b) Word or Google Docs
(c) LaTeX files in a shared folder (d) No draft yet

6. What data do you work with?

(a) Public data (b) Data under a use agreement
(c) Confidential data on a secure server (d) A mix

Learning objectives

- Set up a project folder: a rules file, Auto mode, a `ping` check
- Check citations against publisher records, with fabricated entries kept apart from unverifiable ones
- Turn a coding mistake into a numbered lesson and a checklist line
- Close a chat with a verdict, a handoff file, and a two-sentence blurb

Three rule layers, loaded at different costs

Layer	When it loads	What goes there	Mine today
Global rules file `~/.claude/CLAUDE.md`	Every reply, in every project	Reply shape, the markers, the close-off	About 370 lines
Project rules file `<folder>/CLAUDE.md`	Every session opened in that folder	Data rules, file layout, the code gate	One per project
Skills `.claude/skills/<name>/`	Only when a task calls for one	Long manuals: lessons, checklists, the citation check	20 skills, the Stata one about 8,200 lines

One question for every new rule: where it binds

- On every reply: the global file
- Only in one project: that folder's `CLAUDE.md`
- Only for one task: a skill

The kit: a project rules file and seven starter skills, with your global file as the first step at home

Eight of my 20 skills sit behind today's tasks

Skill	What it does	Where today
`bib-check`	checks each reference against Crossref and the publisher record, keeps a fabricated entry apart from an unverifiable one, and checks the claim as well	Task 1
`ida-stata-lessons`, `ida-python-lessons`	every past mistake as a numbered lesson, plus a checklist run before code is handed back	Task 2
`close-chat`	verifies the work, files every finding, and ends on what the chat accomplished and a verdict	Task 3
`referee-code`	audits a paper's text against its replication code, claim by claim	Task 4
`efficient-ocr`	sets the plan and the cost before any scanned page is read, including the split across helpers	Parallel helpers
`ida-tex-lessons`	LaTeX mistakes already paid for, and a checklist before any compile	Exhibits
`slide-deck`	a storyboard with a timing table, then the frames; the skill that built this deck	This deck

In the kit: basic versions of bib-check, close-chat, and the lessons skills, plus code-to-text-audit and reduce-ai

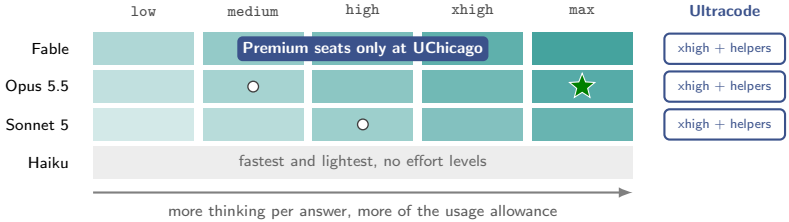
Auto mode by default, because every change has an undo path

- **Manual:** a prompt before every change, which by the third I approve without reading
- **Auto:** each action still judged, and destructive or irreversible ones brought to me
- **Never delete:** a superseded file moved to an archive folder, `_archive_claude/`
- **Version history:** git and Dropbox, so a bad run gets reverted
- **Outside every undo path:** identified data and credentials, so they stay out of any folder Claude can see

The condition behind my default

- Manual review: worth its cost only when I cannot roll back
- My answer: different if that condition changes

Thinking models & effort levels



○ the model's default effort ★ my usual: 76% of my handoffs at max effort, most of them on Opus

- Sonnet for most coding, Opus for hard multi-step reasoning: Anthropic's own advice
- `ultrathink` anywhere in a prompt: deeper thinking on that one turn

Tip: start the 5-hour window early

A scheduled "hi" early in the morning, so the window is already running when you sit down

Through Chrome, Claude can scrape pages that a plain download misses

- Claude in Chrome: opening pages in my own browser, clicking, scrolling, and reading what renders
- Within reach: tables built by JavaScript, results behind a search form, and pages behind my login
- My setup: the Code tab driving Chrome and saving what it reads as a CSV in the project

Example prompt, on my screen

```
Open the district population table on the statistics office's website in Chrome. First tell me whether the site's terms of use allow automated collection. Then page through every district and save district, year, and population to data/raw/district_population.csv, with the page address and today's date in two more columns. Stop and ask me if a page wants a sign-in, and end with how many rows you saved and which pages failed.
```

On a UChicago seat

Nothing to install today, since this runs on my screen; later, the extension from the Chrome Web Store, on by default for Enterprise seats since September 10, 2026, unless your administrators switched it off

Setup: open the kit in the Code tab and switch to Auto

1. Download and unzip the kit
2. Claude Desktop app, **Code** tab: open the folder `claude-code-predoc-kit`
3. Mode picker: switch from **Manual** to **Auto**, since your sessions start in Manual
4. Type `ping`
5. Hands up when the reply starts with `pong: kit rules loaded`

The kit

idagri.github.io/files/claude-code-predoc-kit.zip

If a step fails

- Pong missing: open `claude-code-predoc-kit` itself, then start a new session
- Auto missing from the picker: removed by your administrators, so pick Accept edits
- Nothing else to install today: no Stata, no Python, no browser extension

Task 1: check each citation against its publisher record

Risks in a literature summary: a cited paper that does not exist, and a real paper cited for a claim it never makes

Laptop prompt, task 1 in `prompt_card.md`

```
Use the citation-check skill on the first paragraph of 1_citation_check/lit_summary.tex and the entries it cites in
1_citation_check/lit.bib. For each entry: is it a real paper with the right authors, year, and journal, and does the paper
support the sentence that cites it? Use only the saved records in 1_citation_check/publisher_records/ and stay off the web.
Report only, as a numbered table: entry, status (VERIFIED, WRONG DETAIL, FABRICATED, UNVERIFIABLE, or CLAIM MISMATCH), the
evidence, the proposed fix. Do not edit any file and do not delete any entry.
```

What to look for in the reply

- One row per entry, each with a status and its evidence
- A fabricated entry and an unverifiable one with different statuses
- Every file left as it was, with no entry deleted
- A reply that ends on a numbered question for you

Task 1 debrief: the claim gets checked as well as the title

- **Fabricated:** a DOI belonging to a different paper
- **Unverifiable:** a working paper with no public record yet
- **Claim mismatch:** the summary's sentence beside what the paper reports

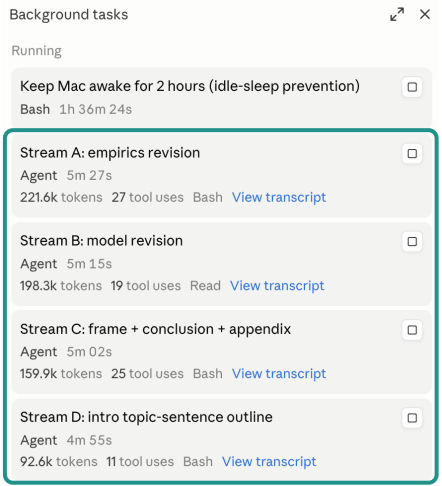
Projector follow-up

```
Apply the WRONG DETAIL fix in 1_citation_check/lit.bib. Keep the UNVERIFIABLE entry as it is: I saw that paper presented and it is not public yet. For the CLAIM MISMATCH, show me the current sentence and your proposed rewording side by side before you change anything. Tell me what the FABRICATED entry claimed to be, and do not delete it until I say so.
```

The tip

Different verdicts for an unverifiable citation and a fabricated one, and a check on the claim as well as the title

One prompt can run several helper agents in parallel



- **Ultracode**, next to the model picker: my pick when a job splits into parts that can run at the same time
- In the screenshot: helpers launched by one chat for separate parts of a paper, each editing only its own files
- Background tasks panel: each helper's run time, tokens, and tool calls, with its transcript one click away
- Each helper: 90,000 to 220,000 tokens, with a Standard seat at about a tenth of my allowance

Parallel helpers fit jobs that split into independent pieces

Job	One helper per	The separate check
OCR of scanned PDFs, page by page	batch of pages	another helper re-reading a sample of pages against the images
Literature review verification	section of the bibliography	another helper re-checking every citation flagged as wrong
PhD programs with a specific focus	program	another helper opening each faculty page the table cites

- Each result saved to a file as its helper finishes, so a run that stops midway keeps what it found
- The checker: always a different helper from the one that produced the result

Example prompt, on my screen

```
Split this job across 4 helper agents, one per batch of 25 pages of scans/survey.pdf. Each helper transcribes its pages to its own CSV and saves it as soon as it finishes. Then launch one more helper that re-reads 10 random pages against the images and lists every mismatch. Tell me how many helpers you will launch before you start.
```

Reply format

- The answer first, the reasoning after
- Bullets and tables for every list and comparison
- File paths listed below a table, one per line
- Verdict words: "Fixed", "NOT verified: no log"
- Every edit as a numbered old/new table
- Each number with the file or log line it came from
- Numbered markers at the end of the reply, on the next two slides

Two lines from a reply in this format

Fixed: the merge keeps all 500 villages
NOT verified: the log ends on `r(601)`, and table 1 did not change

My replies end on numbered markers that I answer in one line

The end of Claude's reply after a merge

S1 **Lost from the sample:** 30 of 500 villages lost in the merge
Can we get them back: yes, fix the ID width

D1 **Decided:** kept the later-dated row of each duplicate pair, because the pair differs only in the enumerator field

Q1 **Your call:** base year 2000 or 2010?

My whole answer

```
Q1 2000, S1 yes
```

- **S** a smaller sample I did not ask for
- **D** a decision Claude made without asking me
- **Q** something Claude needs from me
- Numbered, so I answer by tag
- An unanswered ask: back under **Still open** with its number
- The tags in Claude: shown with emoji markers
- Your replies today: the same ending, from these rules in the kit's `CLAUDE.md`

In the app, the markers print as circles and hearts

> 🔷 **D1 - Decided:** kept the later-dated row of each duplicate pair, because the pair differs only in the enumerator field.

> 🟡 **F1 - Issue:** column 5 has N = 994 while the other columns have 1,000.
> **Next step:** restrict every column to the common sample, or report each N.

> 💗 **S1 - Lost from the sample:** 30 of 500 villages failed the 2010 match.
> **Can we get it back:** yes, all 30: the 2010 file stores village_id as zero-padded text.

> 🟢 **Q1 - Your call:** base year 2000 or 2010?

The example lines from the kit's `CLAUDE.md`, as the app prints them, so today's replies end this way

Task 2: a do-file review that starts from a lessons skill

A `cap drop` on a prefix: the risk of deleting a different variable through Stata's name abbreviation, while the do-file still runs with no error

Laptop prompt, task 2 in `prompt_card.md`

```
Review 2_dofile_review/build_district_totals.do for silent bugs before I run it. Do not run it.
```

Python users: swap in `build_district_totals.py`

What to look for in the reply

- Claude naming the lessons skill and running its checklist before it reads the code
- Each silent bug listed with its line number
- Claude not running the file

The silent bugs in the file

- `cap drop` on a prefix: the per-capita variable deleted through name abbreviation
- Name abbreviation still on, since the file has no `set varabbrev off`
- No log
- `collapse (sum)`, which has no `missing` option: an all-missing district turned into a hard zero

Projector follow-up 1

```
Add what you just found as a numbered lesson in my lessons skill, with a
one-line Rule and the incident, and add a text search to
references/checklist.md that catches it next time.
```

Projector follow-up 2

```
Now fix the file.
```

What lands on disk

- Lesson 1.1 with its one-line Rule, plus one new line in `references/checklist.md`
- The fixed file, reported as `build_district_totals.do` ✓

Task 3: a close-off that ends on a verdict

A chat that ends on "let me know if you need anything else": its half-done items left to a new chat with no memory of them

Laptop prompt, task 3 in `prompt_card.md`

```
Read TODO.md and notes/ first. Are we done? What is outstanding?
```

What to look for in the reply

- The files it re-read before answering
- What is done, and what is NOT done
- A verdict line: "This chat is NOT closed"
- A second line: "Safe to close this window: no"

Research chats opened from a blurb like this one: 57% of mine

The handoff blurb starts the next chat where this one stopped

Projector follow-up

```
Hand the open item to a fresh chat: write the handoff file and print the blurb. If you can offer it as a task chip, do that too.
```

What comes back

- A dated handoff file in `notes/`
- A two-sentence blurb in its own block: what to execute, plus the one fact the new chat needs
- A task chip on my screen: the same blurb as a one-click button

Your turn

- Open a new session in the kit folder and paste only the blurb
- The new chat: reads the handoff file, picks up the open item

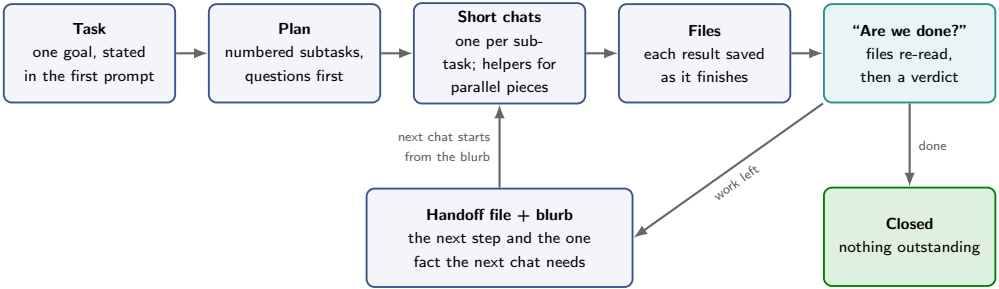
Context

- The whole chat resent with every message: more usage per question in a long chat
- Compaction near the limit: older history replaced by a summary, with details lost
- Usage: a large request for compaction itself, none for `/clear`

Closing chats

- One task per chat, findings written to files as they appear
- A long chat closed with a handoff file and a blurb, before compaction
- 57% of my research chats opened from a handoff or a blurb
- Unavoidable compaction: `/compact` plus what to keep, for example `/compact keep the file names and the open questions`

Approaching a task: split, hand off, close



Your Standard seat differs from my plan in four ways

	My Max plan	Your UChicago Standard seat
Usage allowance	20 times a Pro plan	About twice a Pro plan, so about a tenth of mine
Models	Every current model	Fable 5 for Premium seats only
Mode at session start	Auto	Manual until you switch to Auto
Administrator layer	Only my own settings	Managed settings that nothing you set overrides

University data rules

- Today: simulated data only
- A PI's unpublished project: additional steps required by the university, through the Data Stewardship Council
- Health information: never

Recap

- ✓ Set up a project folder: a rules file, Auto mode, a `ping` check
- ✓ Check citations against publisher records, with fabricated entries kept apart from unverifiable ones
- ✓ Turn a coding mistake into a numbered lesson and a checklist line
- ✓ Close a chat with a verdict, a handoff file, and a two-sentence blurb

Three steps at home turn the kit into your own setup

1. Paste the marker and close-off blocks from the kit's `CLAUDE.md` into your global rules file, `~/.claude/CLAUDE.md`
2. Copy a starter skill from the kit's `.claude/skills/` into your personal skills folder, `~/.claude/skills/`
3. Write one lesson the day something breaks: a one-line Rule, the incident, and one checklist line

Windows paths: `README_setup.md` inside the kit

The kit



idagri.github.io/files/claude-code-predoc-kit.zip

Questions?

Every prompt from today: `prompt_card.md` in the kit, which stays online

The kit
idagri.github.io/files/claude-code-predoc-kit.zip

Ida Grigoryeva · idaliya.gri@gmail.com

▸ Code to text ▸ Plot variants ▸ Separate verifier ▸ Predoc questions ▸ Sources

Task 4: audit the draft's sentences against the do-file

The risk: a draft describing a sample or a specification that the code no longer runs, with a referee or a replicator finding it first

Laptop prompt, task 4 in `prompt_card.md`

```
Audit the draft section in 4_code_to_text/ against the do-file in 4_code_to_text/. For every claim in the text about the sample,
the specification, the fixed effects, the clustering, and the variable definitions: quote the sentence with its line number,
cite the code line that implements it, say whether they agree, and for each disagreement say which side you would change and
why. Do not edit either file and do not run anything. End with two short lists: what checks out, and what you could not verify
from the files.
```

What to look for in the reply

- A ranked table: each sentence with its line, the code line beside it, and the side to change
- A list of what checks out, and a list of what the files cannot settle
- Both files left as they were

▸ Debrief ◂ back

Task 4 debrief: four places where the text and the code disagree

Claim	The text says	The code does
Clustering	By district	By village
Distance cutoff	5 km	10 km
Fixed effects	District and year	District only
Sample size	N = 1,000	A smaller N in the log
Checks out	Outcome: log employment	
Cannot be verified	Winsorizing, and what urban exposure measures: not settled by any file	

Projector follow-up

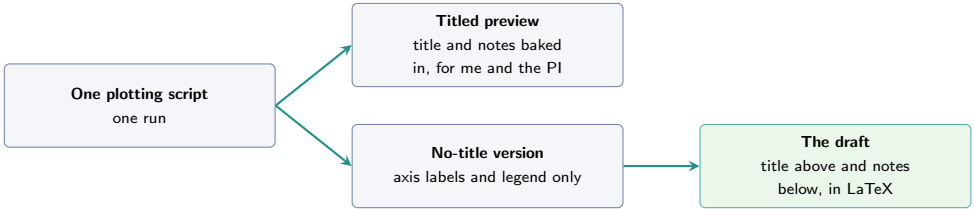
```
Apply fix 2 to the text only, and show me the old and new sentence side by side.
```

The tip: an audit of the text against the code before anyone outside does one

back

One script writes the plot twice: a titled preview and a no-title file

The same fixes every round: white space, the legend over the data, raw variable names, a title printed twice



- The rendered image viewed by Claude before it says the plot is done
- Only the no-title file embedded in the draft: the title printed once

The tip

A correction given twice, turned into a standing rule

back

The helper that produced a number never verifies it

Checking a 40-page draft against its log: an afternoon of reading

- Five helper sessions launched by Claude, one per section of the draft
- A sixth helper that recomputes the key numbers without seeing the other five
- Each helper's result written to a file as it finishes: findings kept if a run dies midway
- My part: reading the disagreements between the sixth helper and the other five

Kept on my screen because of its usage cost

- Six helpers: six sessions' worth of usage
- A Standard seat: about a tenth of my allowance

back

Questions a sharp predoc will ask

Question	My answer
Auto mode, when my data has names in it?	A sandboxed setup for identified data, and Manual or Accept edits for a PI's shared repository
Is this over-engineered?	Each line of mine added after an incident; for a first setup, ten lines and one skill
Who maintains all those lessons?	Each lesson written by Claude the same day, with one line for me to read
Does Claude follow a long rules file?	Not always: kept on track by an example inside each rule, a `ping` check, and a checklist
Will this get my statement past a detector?	Zero detector movement from phrase bans, and movement only from whole blocks of my own words
Why would my PI want your markers?	What transfers: the contract (numbered asks, every drop in N reported, a verdict at the close), under names of your own

back

Sources

- The kit: https://idagri.github.io/files/claude-code-predoc-kit.zip
- Models, effort levels, Ultracode: https://code.claude.com/docs/en/model-config
- Usage windows, context, compaction: https://code.claude.com/docs/en/costs
- Permission modes: https://code.claude.com/docs/en/permission-modes
- Five-hour session limit: https://support.claude.com/en/articles/9797557-usage-limit-best-practices
- Claude in Chrome on Enterprise seats: https://support.claude.com/en/articles/13065128-claude-in-chrome-admin-controls
- UChicago seats and data rules: https://intranet.uchicago.edu/tools-and-resources/tools-and-applications/claude/frequently-asked-questions

back