Ten Ways to Use Agentic AI in Academic Research

REDES-IA — Herramientas de IA para ciencias sociales

Concrete uses of AI coding agents — Claude Code, Cursor, and similar — in academic research. For grad students, postdocs, and faculty; each item is something the workshop demonstrates on real artifacts: code, manuscripts, .bib files, slide decks.

Two things separate an agent from a chatbot. First, *context*: it sits inside your project folder and reads every file in it — manuscript, data, codebook, .bib, slides, replication code, prior emails — simultaneously, not the few snippets you would paste into a chat window. Second, *action*: it edits files, writes new ones, runs code, reads the output, and iterates. The ten uses below are instances of the same underlying capability — one collaborator with your whole project loaded that can do things, not just suggest them. Working in ten separate chat windows loses most of this.

1. Literature triage and synthesis

Drop a stack of PDFs on the agent and ask it to extract each paper's core claim, method, and relevance to your project. The point is not to replace close reading; it's to drain the "maybe relevant" pile down to the papers you actually need to read, and to build a first-pass map of an unfamiliar literature before you commit to a deep dive. Useful for lit reviews, grant background sections, and the moment you realize you should have been reading some adjacent literature two years ago.

2. Learning new methods or unfamiliar literatures

When a statistical technique, archival approach, or body of scholarship outside your field becomes relevant, an agent will walk you through it with a worked example on your own data, point to canonical references, and flag the likely critics. Especially valuable for solo researchers, people between institutions, and anyone whose nearest peer group doesn't happen to cover the method they need this month. Best for the questions you'd never ask a colleague because they're too basic, or you've already asked twice.

3. Data wrangling and harmonization

Fuzzy-matching author names across datasets, OCR'ing scanned tables, deduping records, reconciling country codes between WDI and Penn World Table, assembling a panel from twenty messy raw files. The agent does not just produce plausible-looking code — it runs the merge, inspects what comes out, and iterates until the output passes the sanity checks you specify. This is where the productivity gap with the old StackOverflow workflow is largest.

4. Scrapers, APIs, and data pipelines

Pulling from Census, FRED, OpenAlex, Crossref, archival catalogs, or .gov sites that ship data as ugly HTML tables. The agent writes the request, parses the response, looks at what came back, and adjusts when the schema is not what the docs promised. For projects that need periodic refresh, it can also schedule the pipeline and diff successive runs so you notice when an upstream source quietly changes.

5. Writing, editing, and running empirical code

The full loop — pandas/R/Stata code written, run, error read, fixed, re-run — without you mediating every step. Most useful for the tedious-but-mechanical parts: reshaping panels, recoding values, building descriptives, formatting tables, and the matplotlib legend that refuses to stay where you put it. The difference from a chat-window workflow is that the agent does not stop at "here's what to try"; it tries, watches the output, and adjusts.

6. Verifying consistency and catching errors

Two angles on the same skill. First, code review — agents are unusually good at catching silent errors in pandas merges, off-by-one slicing, miscoded missing values, and the dropped observations you didn't notice. Second, cross-document consistency — numbers in the abstract matching numbers in Table 1, variable definitions matching the codebook, claims in the conclusion matching what the regressions actually show. The agent reads across files in a way a co-author skimming the PDF can't.

7. Project documentation and upkeep

The highest-leverage habit on this list, and the one most users do not realize is possible. As the project evolves, the agent maintains a README, codebook, and decision log — every variable construction, sample restriction, and specification choice recorded the moment it is made, with the reasoning attached. The same agent keeps the artifacts tidy: cleaning duplicates from the .bib, archiving stale notebooks, flagging dead code, normalizing file names. Reconstructing all of this a year later from memory during the R&R is the part of empirical work that ages researchers fastest, and with an agent it is optional.

8. Manuscript writing and production

The agent has the whole manuscript open — .tex source, tables, figures, .bib, data, codebook — and can write across them. This means prose drafted from the project rather than from thin air: the abstract built from the actual numbers in your tables, the methods section written against the variable definitions in your codebook, the response letter pointing to the specific edits you made. The same agent handles production: LaTeX table formatting to journal specs, Beamer, tikz, the 4am compile error the morning of submission, and the surprising fraction of writing time most academics lose to ggplot/matplotlib legend placement, axis labels, and color schemes, most of which now goes away.

9. Custom skills, subagents, and memory

Out of the box, an agent knows nothing about your project, your conventions, or the workflows you keep repeating. The compounding payoff comes from building the scaffolding: CLAUDE.md files that encode your project's conventions, persistent memory notes about who you are and how you work, custom skills (knowledge: how to audit an abstract, format a regression table, validate a specification before estimation), and subagents (roles: a referee, a copyeditor, a methodologist).

After a few months of this, the agent is no longer a generic helper but a customized team that knows your projects and your preferences.

10. Teaching materials

Slides, problem sets, exam questions, TA guides, syllabi. The most concrete payoff: variants of the same problem so students cannot copy last year's solution set. Beyond that, agents adapt the same content across difficulty levels (PhD methods → MA methods → undergrad), draft worked solutions you can edit, and produce the marginal slide or worked example that turns a rough class into a polished one — especially when you are building a course from scratch.