

# How to use these materials

AI Agents for Empirical Research — Cobra Seminar companion

Paul Seidel, University of Mannheim | May 2026

The session recording is distributed separately by the seminar organizers. This document covers everything else: what we built, how the configuration files fit together, and how to reproduce the results.

## 1. What we did

A live, end-to-end empirical research replication driven by Claude Code. We replicated Bernard & Thomas (1989), *Journal of Accounting Research* — the post-earnings-announcement drift (PEAD) anomaly — from raw WRDS pull through to a referee-reviewed SSRN-style working paper, in one session.

The point of the session was not the replication itself; it was to show that every role on a typical research team maps cleanly onto a Claude Code primitive. The full mapping is below.

Human role in the team	Claude Code primitive
<b>YOU's standing preferences</b> <i>voice, methodology, defaults across all projects</i>	Global CLAUDE.md (~/.claude/CLAUDE.md)
<b>Project Brief</b> <i>target journal, sample, clustering, citation style</i>	Local CLAUDE.md (project root)
<b>Notepad</b> <i>what you noticed about this project</i>	Memory (Claude-written, per-project facts)
<b>3 Coauthors</b> <i>data, code, writing</i>	Sub-agents: <b>data-downloader</b> + <b>analyst</b> (writing handled by the Skill below)
<b>Writing Playbook</b> <i>style, citations, review checklist</i>	Skill: <b>paper-writing</b> (SSRN voice, AFA citations, mandatory review)
<b>Editor + 2 Referees</b> <i>parallel review with synthesis</i>	Sub-agent team: <b>referee-econometrics</b> , <b>referee-theory</b> , <b>referee-exposition</b> (parallel) + <b>referee-chair</b> (synthesizer)
<b>Colleague who shares code</b> <i>bounded access to an external system</i>	MCP servers (GitHub, WRDS, Drive, Slack, ...)

Each primitive runs in its own isolated context, with its own instruction prompt and tool list. Hand-offs between agents go through plain files (a YAML manifest for raw data, parquet panels, .tex tables, Markdown referee reports), so the work is auditable and re-runnable.

## 2. Installation

1. Install **Visual Studio Code**.
2. Install the **Claude Code** extension from the VS Code marketplace and sign in with an Anthropic account that has Claude Code access.
3. Verify Python 3.11 or newer is available on the system path (`python -version`).
4. Optional: install MiKTeX (Windows) or TeX Live (macOS/Linux) if you intend to compile the LaTeX outputs.

## 3. Configuration — where each file goes

1. Copy **Configuration/GLOBAL\_CLAUDE.md** to your user-level directory:  
Windows: C:\Users\<you>\.claude\CLAUDE.md  
macOS/Linux: ~/.claude/CLAUDE.md  
This file sets your personal voice and methodological defaults across every project.
2. Copy **Configuration/Project\_CLAUDE.md** to the root of your project folder as **CLAUDE.md**. This file sets project-specific conventions (sample window, clustering choice, citation style).
3. Copy **Configuration/.claude/** (containing **agents/** and **skills/**) into the project root. These provide the sub-agents and the Skill the session used.

## 4. Operating Claude Code

- **Sub-agents.** Invoke by name in natural language: “use the **data-downloader** agent to pull WRDS”. Each sub-agent runs in its own context window with its own tool permissions.

- **Skills.** Triggered automatically when your request matches the Skill's description, or explicitly via `/skill-name`. The `paper-writing` Skill enforces the SSRN/AFA style and forces every numerical claim to be re-verified against the source files before it is written.
- **Prompts.** The 11 prompts we typed during the live session are in `03_Demo_Prompts.docx`, alongside the verbatim configuration files we pasted into Claude Code each time the prompts asked for one. The prompts themselves are short and terse on purpose — Claude infers the rest from the configuration files. Copy and paste them in order.

## 5. Reproducing the results

The `Outputs/` folder already contains every table, figure, referee report, and the compiled `Replication_Paper.pdf` from the live session, so no WRDS access is required to inspect them. To re-run the pipeline from scratch, obtain the underlying CRSP, IBES, and Compustat tables yourself from [WRDS](#) under your own subscription terms, then step through `03_Demo_Prompts.docx` in order.

## 6. Package contents (full index)

File / Folder	What it shows
<code>01_Slides.pdf</code>	Conceptual deck of the session. Pipeline mapping, foundations, and feature tour.
<code>02_How_To_Use.pdf</code>	This document.
<code>03_Demo_Prompts.docx</code>	Two parts: the verbatim configuration files we pasted into Claude Code, and the 11 prompts we typed during the live session, numbered and in order.
<code>04_Replication_Paper.pdf</code>	The PEAD replication working paper, drafted by the <code>paper-writing</code> Skill.
<code>05_Referee_Report.pdf</code>	The referee report produced by the four-agent panel.
<code>06_PEAD_Project_README.md</code>	The replication project README with the targets-to-output mapping.
<code>Configuration/</code>	Global and project <code>CLAUDE.md</code> , six sub-agents, and the <code>paper-writing</code> Skill.
<code>Source_Code/</code>	Python pipeline scripts.
<code>Outputs/</code>	Generated tables ( <code>.tex</code> ), figures ( <code>.pdf</code> ), referee reports ( <code>.md</code> ), logs.
<code>requirements.txt</code> , <code>.env.example</code>	Python dependencies and credential template.

## 7. Further reading & watching

**Official Claude Code docs** (each primitive has its own page; start at the overview):

<https://code.claude.com/docs/> — [VS Code setup](#) | [Memory & CLAUDE.md](#) | [Sub-agents](#) | [Skills](#) | [MCP](#).

**Video — overall introduction.**

- English, end-to-end walkthrough (~35 min) — [https://www.youtube.com/watch?v=vLwfguLz\\_qQ](https://www.youtube.com/watch?v=vLwfguLz_qQ)
- Deutsch (Alternative auf Deutsch) — <https://www.youtube.com/watch?v=Lu95f1ZBIos>

**Video — per primitive (English).**

- Memory and `CLAUDE.md` — *Claude Code's Memory System: The Full Guide* — <https://www.youtube.com/watch?v=FRwZg6V0jvQ>
- Sub-agents — *Claude Code Sub-Agents: Step-by-Step Beginner Tutorial* — <https://www.youtube.com/watch?v=MbKUeufUZIY>
- Skills — *7 Claude Code skills I use every single day* — <https://www.youtube.com/watch?v=UpgjJdQJShWg>
- MCP servers — *Claude Code MCP: How To Add To Servers* — <https://www.youtube.com/watch?v=4fFPGvZ3gEQ>

**Research-workflow references** (what other empirical researchers are doing).

- Pedro H. C. Sant'Anna (Emory), *My Claude Code workflow* — <https://psantanna.com/claude-code-my-workflow/workflow-guide.html>. Most complete public template for an academic Claude Code setup; the parallel-referee pattern in this session is adapted from his orchestrator design.
- Paul Goldsmith-Pinkham (Yale), *Claude Code for Applied Economists* mini-series at Markus' Academy — <https://bcf.princeton.edu/events/paul-goldsmith-pinkham-mini-series-on-claude-code-for-applied-economists/> (companion: <https://paulgp.substack.com/p/research-in-the-time-of-ai>).

Questions: Paul Seidel, University of Mannheim — [Paul.seidel@uni-mannheim.de](mailto:Paul.seidel@uni-mannheim.de). Adapt the configuration files to your own project — they are deliberately minimal.