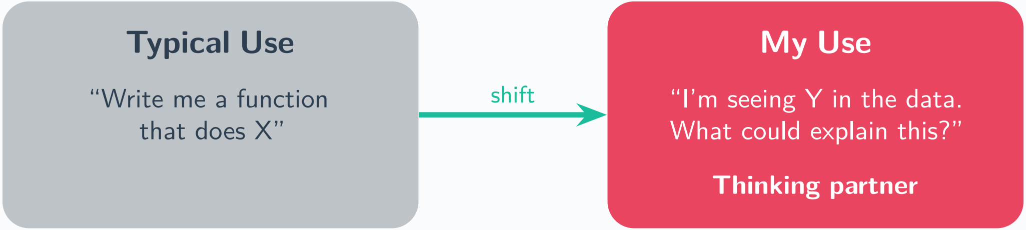
# My Claude Code Workflow

A Framework for AI-Assisted Empirical Research

Scott Cunningham

# The Core Insight



**Typical Use**

“Write me a function that does X”

shift

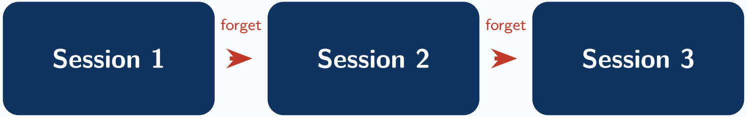
**My Use**

“I’m seeing Y in the data. What could explain this?”

**Thinking partner**

The hard part of empirical research isn’t writing code—
it’s **figuring out what code to write** and
whether **results mean what you think**.

# The Fundamental Problem: Claude Has Amnesia

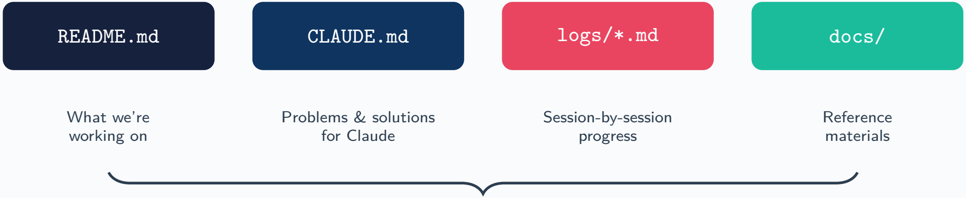


Every session starts from **zero context**.

Most people just re-explain everything verbally.

**I build external memory in markdown files.**

# External Memory via Markdown



`README.md`

What we're working on

`CLAUDE.md`

Problems & solutions for Claude

`logs/*.md`

Session-by-session progress

`docs/`

Reference materials

**Institutional memory** that persists even though Claude's doesn't

# Part I

The Daily Practices

# The Socratic Method for Alignment

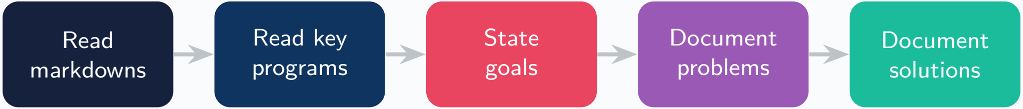
*"Do you see the issue with this specification?"*

*"That's not it. The problem is the standard errors."*

*"Guess at what I'm about to ask you to do."*

**Why?** If Claude guesses wrong,
that reveals a misunderstanding
that needs correcting **before** proceeding.

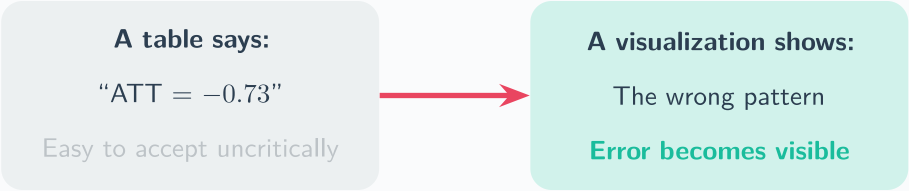
# Session Startup Routine



Read markdowns → Read key programs → State goals → Document problems → Document solutions

Takes **2 minutes**. Claude starts each session **informed** rather than ignorant.

# Verification Through Visualization



**A table says:**

“$\text{ATT} = -0.73$”

Easy to accept uncritically

**A visualization shows:**

The wrong pattern

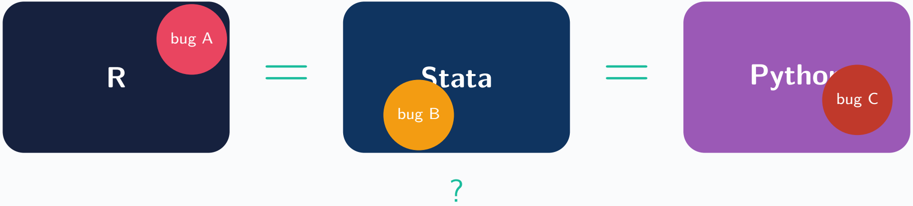
**Error becomes visible**

**Trust pictures over numbers.** Always ask for figures.

# Part II

Cross-Software Replication

# Bugs Are Orthogonal Across Languages



A bug in `dplyr` code is **unlikely** to be the same bug in Stata.

If all three produce the **same answer**, you have **high confidence**.

# Why Cross-Language Validation Works

**Syntax errors are language-specific**

Both must navigate their own syntax

**Default behaviors differ**

NA propagation, factor ordering, etc.

**Implementation paths differ**

`group_by()` vs. `collapse`

Same answer ⇒ Same concept ⇒ **Correct**

# The Validation Table

| Check | R | Stata | Match? |
|---|---|---|---|
| Sample rows | 5,234 | 5,234 | ✓ |
| Mean age (treatment) | 31.7 | 31.7 | ✓ |
| Mean hourly wage | $5.43 | $5.43 | ✓ |
| N missing earnings | 412 | 412 | ✓ |

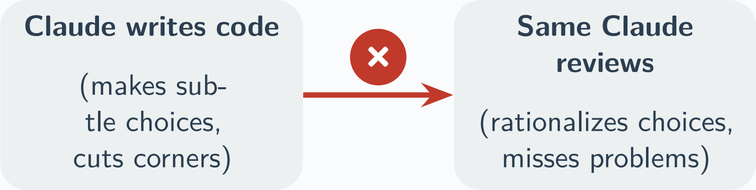
If any differ ⇒ **investigate before proceeding**.

This isn't paranoia. It's **methodology**.

# Part III

Adversarial Review

# You Can't Grade Your Own Homework



**Claude writes code**

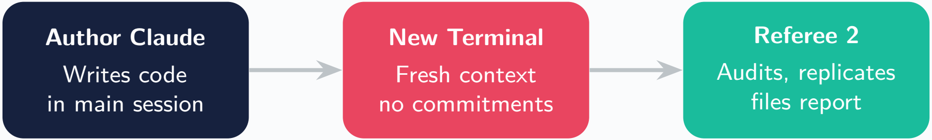
(makes subtle choices, cuts corners)

**Same Claude reviews**

(rationalizes choices, misses problems)

Asking the same Claude to review its own code is like asking a student to grade their own exam.

# The Solution: Referee 2 Protocol



**Author Claude**

Writes code
in main session

**New Terminal**

Fresh context
no commitments

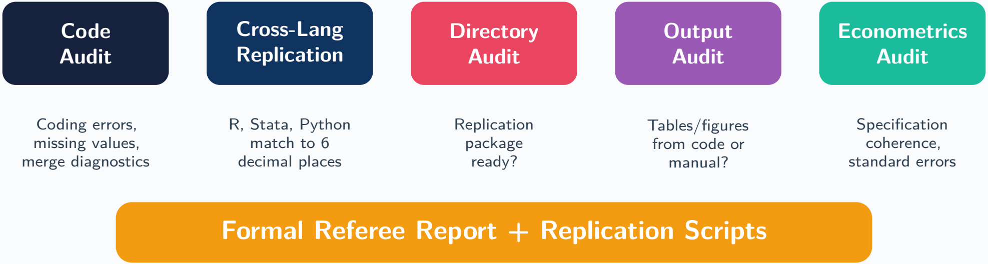
**Referee 2**

Audits, replicates
files report

True adversarial review requires **separation**.

Fresh context. No prior commitments. Formal protocol.

# The Five Audits



**Code Audit**

Coding errors, missing values, merge diagnostics

**Cross-Lang Replication**

R, Stata, Python match to 6 decimal places

**Directory Audit**

Replication package ready?

**Output Audit**

Tables/figures from code or manual?

**Econometrics Audit**

Specification coherence, standard errors

**Formal Referee Report + Replication Scripts**

# What Referee 2 Catches

- ✓ **Unstated assumptions** — “Did you verify X or just assume it?”
- ✓ **Alternative explanations** — “Could the pattern come from something else?”
- ✓ **Documentation gaps** — “Where does it explicitly say this?”
- ✓ **Logical leaps** — “You concluded A, but evidence only supports B”
- ✓ **Missing verification** — “Have you actually checked the raw data?”

# The Philosophy

Referee 2 isn't about being negative.

It's about **earning confidence**.

A conclusion that survives rigorous challenge
is stronger than one never questioned.

Find weak points and either **fix them** or **accept them knowingly**.

# Part IV

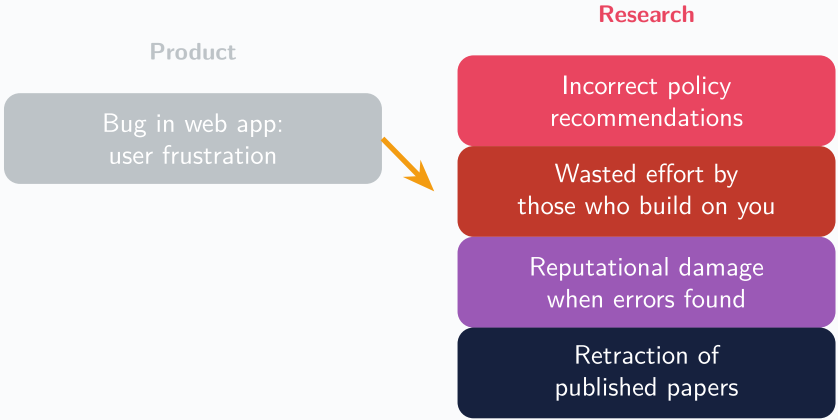
Why This Works

# Research Is Not Product Development

| Aspect | Product Dev | Research |
|---|---|---|
| Goal | Ship working code | Understand correctly |
| Error cost | Bug in production | Wrong conclusion |
| Iteration | Fast ship fix later | Slow careful right |
| Testing | Unit tests CI | Visual inspection |
| Success | Does it run? | Does it mean what we think? |

The difference between “the code runs” and “**the code is correct**.”

# The Stakes Are Different



Product

Bug in web app:
user frustration

Research

Incorrect policy recommendations

Wasted effort by those who build on you

Reputational damage when errors found

Retraction of published papers

# The Workflow in Summary

| **Dimension** | My Approach |
|---|---|
| **Philosophy** | Thinking partner, not code generator |
| **Memory** | External via markdown (Claude has amnesia) |
| **Verification** | Cross-software: R = Stata = Python |
| **Review** | Referee 2 protocol in fresh terminal |
| **Documentation** | First-class output, not afterthought |
| **Visualization** | Trust pictures over numbers |
| **Speed** | Correctness over velocity |

# The Key Insight

Claude Code isn't just a code generator—

it's a **thinking partner**.

Ask it questions. Make it explain.

Verify visually. Document everything.

And when stakes are high, spawn **Referee 2**.

# That's How I Use AI for Research

Scott Cunningham

`scunning.com · causalinf.substack.com`