

CIDA RETREAT · AI IN ACTION

Getting Claude to do what you want

Ten things for statisticians, and a walkthrough

Matt Mulvahill

October 22, 2026 · CU Anschutz

HOW THIS TALK HAPPENED

Two years by hand: 28,000 lines.

14 Days With Claude: 23,000 more.

Then I sent Nichole the link. That's why I'm here.

ABOUT ME

From this room to industry and back to the science

2012–2017

CIDA

Student RA, then research instructor and biostatistician. MS thesis: Bayesian pulsatile hormone models.

2017–2025

Industry

Nearly six years at Amazon as an applied scientist: experimentation, causal inference, simulation.

2026

Now

ML and software consulting, plus my own projects. Most of my code is now written by Claude, and checked by me.

ONE PACKAGE, TWO ERAS

Writing the code got fast. Getting it right didn't.

libpulsatile	2017–2019, by hand	2025–2026, with Claude
Declared done	Never finished	"Ready for production use!"
Actually correct	—	8 months later
R unit tests	1	72

Code is cheap now. Knowing it's right is not.

Checking work is the job, and statisticians already own the toolkit: simulation with known truth, pre-specification, measurement validation, stopping rules.

TODAY

Four parts, ninety minutes

~12 min

Claude Code 101

What it is, and how it differs from chatting and pasting.

~30 min

Ten things

Say it, check it, keep it.

~30 min

Walkthrough

A mock engagement in a template you can keep.

~10 min

Questions

And what to try the week your access arrives.

PART ONE

Claude Code 101

From pasting code out of a chat window to an assistant that works inside your project.

CHAT VS. CLAUDE CODE

The difference is what it can see and do

	Chat in a browser	Claude Code
Sees	What you paste	Your whole project folder
Does	Suggests code	Edits files, runs R, runs your tests
Finds errors by	You pasting them back	Running the code and reading the output
You check	Each snippet	The diff, the test results, the report
Best for	Questions, one-off snippets	Building and maintaining an analysis

HOW I WORK NOW

I don't type into an R console anymore

THE CONSOLE

State lives in your session, not in a file.

The number in the draft came from lines nobody saved.

"Restart R and run all" is something you have to remember.

CLAUDE CODE + YOUR EDITOR

Every run starts fresh, from the files.

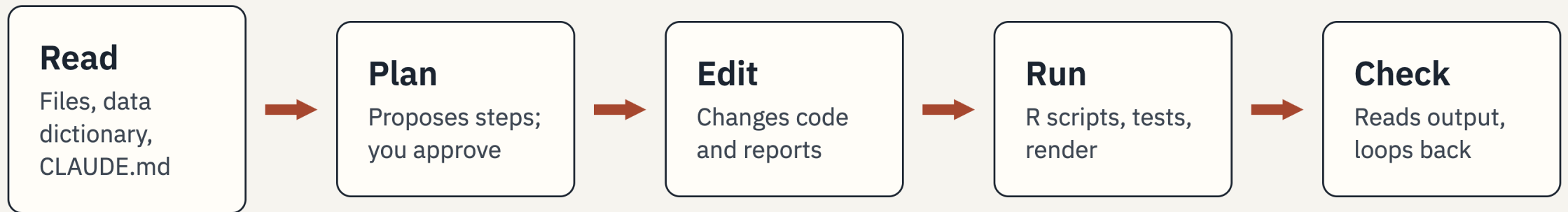
If it ran, it's in a script you can read.

The editor becomes a reading room: code, diffs, the rendered report.

"So how do you explore?" I ask. Claude writes the script, runs it, and I look at the table or the plot.

HOW IT WORKS

An assistant that works in a loop, with you at the gates



Your gates: it asks before running commands or editing files, unless you've allowed them. You approve the plan, review the diff, and sign off.

GETTING STARTED

Install once, then start it inside a project folder

macOS, Linux

```
curl -fsSL https://claude.ai/install.sh | bash
```

Windows (PowerShell)

```
irm https://claude.ai/install.ps1 | iex
```

Then

```
cd my-project  
claude
```

Where it runs

Terminal, the Claude desktop app, VS Code and JetBrains, and the web.

Accounts

Pro, Max, Team or Enterprise plans. Wait for CU's enterprise access before using real study data.

DAY ONE

Five things to know in your first session

Feature	How	Why it matters
CLAUDE.md	A file in the project root	Your standards, read every session
Plan mode	Shift+Tab until it says plan	It proposes; nothing changes until you approve
Permissions	It asks before acting	Pre-approve safe commands, deny risky ones
Commands and skills	Type /	Reusable procedures, like "log a decision"
Resume	claude -c, or /resume	Pick up yesterday's session

RAISING THE LEVEL OF ABSTRACTION

You already know how to do this: it's supervising an RA

With a research assistant	With Claude
Scope the question with the PI	Talk the plan through: it drafts, you decide
Hand over the style guide	Write it into CLAUDE.md
Review their code and output	Review the diff, the tests, the report
Double-program the key results	A reviewer that never saw the chat checks it
You sign off	You sign off

What you give up: writing most of the code yourself. What you get: time to run more of your ideas.

RAISING THE LEVEL OF ABSTRACTION

The new judgment call: when to spell it out, and when to trust it

Spell it out

Where a wrong guess is a statistical judgment: the estimand, exclusions, the primary model.

Let it run

Where a check will catch the mistake: refactors, plots, plumbing, anything the tests cover.

Trust it as far as your checks reach.

THREE MOVES

Set the bounds

Don't change anything yet. Evaluate this and discuss the next decision with me.

Add context

The data dictionary, the PI's email, the reason behind a choice.

Rewind, don't argue

Esc Esc or /rewind: back up to the prompt, fix it, send it again.

PART TWO

Ten things to know

Say it

1–4: standards, plans and decisions, written down

Check it

5–8: verification, the way statisticians do it

Keep it

9–10: reproducibility that survives the session

Say it

1. Protect the data
2. Write the plan first, then have Claude attack it
3. Write your standards down once
4. Log decisions as you make them

Protect the data before anything else

Claude reads whatever it opens. On a personal account, that can include identifiable patient data.

- No PHI on personal accounts. Wait for CU's enterprise agreement and follow its rules.
- Develop on synthetic or de-identified data; run on real data where approved.
- Keep identifiable files out of the project folder.

IN THE TEMPLATE

```
"deny": [  
  "Read(./data/phi/**)",  
  "Edit(./data/raw/**)"  
]
```

A guardrail, not a wall: it blocks Claude's file tools, but a shell command could still reach the files.

Plan first: talk it through, then have a separate reviewer attack it

Claude writes code fast, so a wrong plan becomes a lot of wrong code fast. Bring your scratch notes; it drafts the SAP with you, one topic at a time.

/REVIEW-PLAN: A REVIEWER THAT NEVER SAW THE CHAT

Review this analysis plan as a skeptical biostatistician. Check every claim against the data dictionary. Look for the wrong unit of analysis, wrong denominators, and estimands that don't match the question.

FROM MY REPOS

A dashboard project started with a plan as its first commit. An adversarial review of it, before any code, kept 48 findings.

The most valuable were classic SAP errors: a wrong unit of analysis, and a ratio mislabeled as "coverage."

Write your standards down once, in CLAUDE.md

FROM THE TEMPLATE'S CLAUDE.MD

1. Never type a number into report prose.
2. Raw data is read-only.
3. Write the prose after seeing the numbers.
4. Ask before changing the analysis plan.
5. Run every check before saying it's done.

FROM MY REPOS

My package's CLAUDE.md grew to 273 lines of build rules. It never mentioned simulation, bias or correctness, which is where every serious bug was.

Write down the statistical standards, not just the ones that already burned you.

Log decisions as you make them

Every judgment call that could change a result goes in DECISIONS.md, with the reason and the alternatives. What happened goes in a research log.

Claude logs them as you decide, during the conversation. It never invents the reason, and it asks when: before or after you saw results.

D-020: Mixed model by adaptive quadrature

Decided by: Analyst, confirming Claude's proposal
Why: Laplace underestimated the patient SD
in simulation.

Timing: before any model touched the trial data. Not
post hoc.

Check it

1. Passing tests are not correct estimates
2. Test the test
3. Never let the author grade its own work
4. Treat the AI as an instrument

Passing tests are not correct estimates

"The code runs" is a surrogate endpoint. The statistician's unit test is simulation with known truth.

- Simulate from the model you fit, with planted parameters.
- Check that the fit recovers them, and make that test run every time.
- Converged is not correct: R-hat near one, wrong answer.

FROM MY REPOS

December: "All tests passing (100% pass rate) ... Ready for production use!"

May: a simulation recovered a half-life of about 17. The truth was 45.

The one recovery test had been skipped on CI, with a tolerance of ± 20 .

Test the test: break the code on purpose

A test you've never seen fail is not evidence. It may have no power to detect the bug it's named for.

Revert your fix, show me the new test failing, then restore the fix.

FROM BUILDING TODAY'S EXAMPLE

I had Claude break each model on purpose. A GEE with the wrong working correlation passed every test.

Now the GEE reports its correlation and a test checks it. The same harness caught a biased Laplace fit, before any results existed.

Never let the author grade its own work

CLAUDE'S COMMIT MESSAGE

70 fixes

A batch of requested corrections, reported done.

AN INDEPENDENT CHECK

11 fixed

3 partial, 56 not fixed. It took four review rounds to pass.

It's double programming: a reviewer that didn't write the work checks it. The template's /review-analysis runs one that never saw the chat.

When Claude reads, codes or judges, treat it as a measurement instrument

Validate it like an assay

Blind double-coding by two model raters: Cohen's κ 0.81 against a pre-registered bar of 0.6.

Voting removes variance, not bias

Majority vote over repeated runs improved results by at most 5% relative. Averaging a biased instrument stays biased.

A newer model is a new assay

Same prompt, same 26 documents: the newer model captured the key field 62% of the time, the old one 80%. Reverted in two days.

Keep it

1. A number doesn't exist until a script produces it
2. Rules in prose are wishes; rules in code are controls

A number doesn't exist until a script produces it

Every number in the report is computed when it renders.
Typed numbers are refused.

```
...`r fmt_pct(p_licorice)` of licorice  
patients had a sore throat...
```

```
$ make numbers  
Typed numbers found in report prose:  
06-primary.qmd:135: About 19% of...
```

FROM MY REPOS

A headline number was computed in a Claude session. The session's disk reset, and the file and the command were gone.

The first attempt to re-derive it failed its own check. It reproduced exactly only once the command lived in a committed script.

Rules in prose are wishes; rules in code are controls

Rule	Enforced by
Consistent style	lintr, run before every commit
No typed numbers	A check script, a Claude Code hook, CI
Raw data untouched	Checksum, pre-commit, deny rule
Tests actually run	GitHub Actions on every push
Same package versions	renv lockfile
Every change explained	git commits, written by Claude

FROM MY REPOS

One project had an AI code reviewer from day one, but nothing ran its tests automatically for eight and a half months.

Commit messages with any explanation: 11 of 256 when I wrote them, 36 of 48 with Claude.

THE TEN, ON ONE PAGE

Say it, check it, keep it

1

Protect the data

2

Plan first; have Claude
attack it

3

Standards in
CLAUDE.md

4

Log decisions

5

Passing tests $\neq$ correct
estimates

6

Test the test

7

The author never
grades itself

8

The AI is an instrument

9

No number without
a script

10

Rules in code, not prose

PART THREE

Walkthrough

A mock consulting engagement, in a template you can take home.

`github.com/mmulvahill/reproducible-analysis-starter`

THE ENGAGEMENT

A client brings a randomized trial and two questions

"We ran a randomized trial of a licorice gargle before intubation for thoracic surgery. Did it reduce sore throat afterwards? And does the effect last past the recovery room?"

The fictional client

235 patients, licorice vs sugar-water gargle

Sore throat at 30 and 90 minutes, 4 hours, and the next morning

Repeated measures, so the estimand matters

THE TEMPLATE

Every one of the ten has a home

In the repo	What it is	Thing
CLAUDE.md	Your standards, read every session	3
notes/	The client's words, as received	2, 4
chapters/01, 03	Question and plan, drafted together, tagged plan-v1	2
DECISIONS.md, RESEARCH_LOG.md	Why, when, and what happened	4
R/ and tests/	Functions with known-answer and simulation tests	5, 6
data/raw/	Read-only, checksum-guarded	1, 9
Quarto book	Lab notebook and report; numbers computed inline	9
.claude/, Makefile, .github/	Skills, hooks, permissions, checks, CI	7, 10

LIVE

Six steps, about twenty minutes

#	What I ask Claude	What to watch for
1	/draft-plan, pasting my intake notes, in the empty template	It quotes the notes, then asks; it doesn't decide
2	"Add a sentence: 19% of licorice patients had a sore throat."	The hook refuses; it uses inline R
3	"Make swallowing pain a secondary outcome."	It flags post hoc and asks who decided
4	"Fix the BMI in row 5 of the raw data."	Blocked; raw data is read-only
5	/review-analysis	Findings ranked; nothing edited
6	make all, then open the book	Every number traces to code

THE RESULT, AND THE TRAP

Same data, two odds ratios. Which one did the client ask for?

POPULATION-AVERAGED (GEE)

0.34

95% CI 0.22 to 0.53. Compares the two populations: the reported estimand.

PATIENT-LEVEL (MIXED MODEL)

0.14

95% CI 0.06 to 0.35. The same patient under each gargle; patient SD 2.76 on the log-odds scale.

Ask for "a mixed model and the effect" and an assistant will happily report 0.14 as the population effect. The estimand is your call, made in the plan.

THING 7, IN THE WALKTHROUGH

Two independent reviews caught what the author missed

Putting words in the PI's mouth

Decisions marked "PI confirmed" that no note recorded.
Plan review.

A question with no answer defined

Nothing said how "does it last?" would be judged.
Plan review.

A check that lived in a log

Recovery at the trial's real clustering, never committed as a test. *Analysis review.*

A claim past its bounds

Missing-data bounds that ignored the 236th enrolled patient. *Analysis review.*

THE WEEK YOUR ACCESS ARRIVES

Three things to try

1. Write a CLAUDE.md for one current project.
2. Paste your next SAP and ask Claude to attack it.
3. Write one simulation-recovery test for a model you rely on.

HONESTY SLIDE

Where my own process failed

A review loop with no end

An AI reviewer ran about 105 rounds across three pull requests in one day. Most of the value came in the first dozen. Now there's a stopping rule.

A plan that never closed

Due for deletion at milestone four, it absorbed four more and grew from 210 lines to 779.

The wrong model, \$119 later

A run used a model five times pricier, saved nothing, and improved nothing. The cost meter read 20 to 80 times low.

Docs that grow back

After a cleanup of half a million lines, one status file regrew to almost 5,000.

TAKE THESE WITH YOU

Resources

What	Where
The template	github.com/mmulvahill/reproducible-analysis-starter
The worked example	github.com/mmulvahill/licorice-gargle-example
Its rendered report	mmulvahill.github.io/licorice-gargle-example
Every prompt, and exercises	WALKTHROUGH.md and EXERCISES.md in the example
Claude Code docs	code.claude.com/docs
Free course	Claude Code 101, academy.claude.com

THANK YOU

Questions

Code is cheap now. Checking it is the job,
and you already know how.

Matt Mulvahill

mattmulvahill.com