Computational & AI-Assisted Methods for Social Sciences

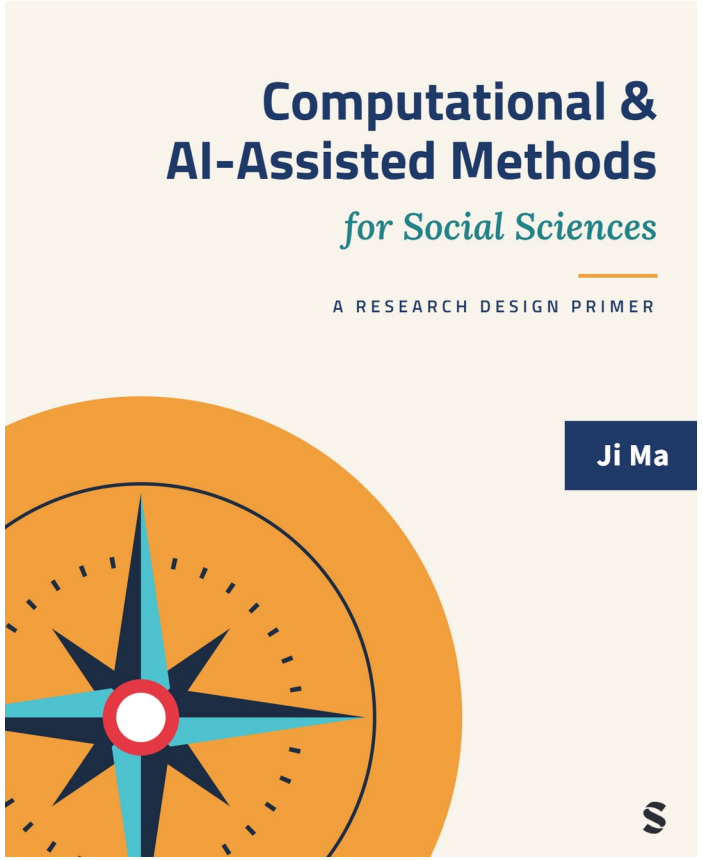
A Research Design Primer

Chapter 2

AI-Assisted Research Framework: Rules, Memory, and Knowledge

Ji Ma

cssprimer.org



Learning objectives

By the end of this chapter, you can:

- Distinguish rules, memory, and knowledge, and explain why collapsing them undermines agents
- Recognize the disorganized environment, not the weak model, as the usual cause of poor agent performance
- Understand the epistemological rationale: essential opacity makes structure necessary, not merely convenient
- Translate the framework into a concrete project directory
- Prepare the research environment the later exercises assume

From using a tool to working with an agent

- Chapter 1: ask, answer, verify — AI as a tool
- This chapter: organize the project so agents work consistently, session after session
- The framework is infrastructure, not a method of analysis

§2.1

Why a framework: disorganized research environments

When agents disappoint, check the environment before blaming the model.

A familiar scenario: the scattered project

- Data files scattered; three codebook versions, none marked current
- Key articles locked in Zotero, unreadable to the agent
- Ask the agent to refine a coding dictionary…
- …it returns a plausible but generic list that ignores the project's actual state

Takeaway: Each iteration feels productive but does not accumulate — an environment problem.

The more common cause of poor agent performance is something you control entirely: the state of the research environment itself.

Trust is thick; reliance is thin

§2.1

- Scientific trust is “thick”: responsibility, accountability, checkable procedure (Koskinen 2024)
- A colleague can explain and justify their coding decisions
- An agent gives you *outputs* but not *accountability*

Takeaway: You can use an agent's outputs; you cannot trust them in the thick sense collaboration requires.

Essential, not accidental, epistemic opacity

- A mass spectrometer is opaque to *you*; someone understands it
- No one, developers included, can fully grasp how an LLM reaches its outputs
- Opacity is a structural property of the technology, not a training gap

Three requirements for responsible reliance

§2.1

- Configure what the agent may and may not do — “agencing” (Lee 2025)
- Keep a traceable record of what happened (Clark & Khosrowi 2022)
- Ground the agent's reasoning in explicit project materials (Feng et al. 2026)

Takeaway: Rules configure, memory records, knowledge grounds — the framework follows from the epistemology.

A digital ba: persistence across sessions

- AI as a “theoretical workshop” — but a workshop must be organized (Olenick et al. 2025)
- Nonaka's *ba*: the live, shared space of knowledge creation
- The live context evaporates when the session ends and the context window clears

§2.2

The framework: rules, memory, and knowledge

Three pillars, each guarding against a different kind of failure.

Three pillars, one line each

- **Rules** = how the agent works
- **Memory** = where the project currently stands
- **Knowledge** = what the project knows

Takeaway: Without rules: inconsistency. Without memory: repetition. Without knowledge: polished shallowness.

Rules: the working discipline of the project

- Scope of authority: where the agent may act, where it must ask
- File and folder conventions; where each output belongs
- Verification and documentation expectations
- Conduct under uncertainty: flag ambiguity instead of forcing a conclusion

Takeaway: Rules convert a general-purpose model into a project-specific assistant.

Verification belongs inside the rules

- Save interim outputs before finalizing
- Run checks before treating results as usable
- Inspect random samples of coded data; compare against prior versions
- Flag ambiguity instead of forcing certainty

Takeaway: Without checkpoints, fast workflows produce polished nonsense.

“Be careful” is a wish, not a rule

- Vague: “be careful with the data”
- Specific: “never modify data/raw/; date-prefix everything in data/interim/”
- “After coding, inspect 20 random records; report class-level precision and recall”

Takeaway: A specific rule is a configuration of agency; a vague one delegates it silently.

Memory: where the project currently stands

§2.2

- Prior attempts and their outcomes; decisions and their rationale
- Failure patterns to avoid repeating
- Current direction, open questions, priorities
- Why abandoned paths were abandoned

Takeaway: Without memory, every session reconstructs context from scratch.

Research directions are orientations, not specifications

§2.2

- “We want to understand how AI is discussed in prosocial contexts”
- Part of the work *is* figuring out what the question really is
- Unrecorded refinements leave the agent optimizing a superseded framing

Takeaway: Social science memory must track a moving question, not execute a fixed spec.

Memory has layers that move at different speeds

- Research direction — revised over months
- Working questions — revised over weeks
- Task priorities — revised each session
- Speed is diagnostic: a direction changing weekly is not yet a direction

Takeaway: Separated layers let an agent tell a settled commitment from a working guess.

Selective memory enables retrospective control

- Not a transcript: record key decisions and direction changes
- Opacity hides the *how*; memory preserves the *what* and the *why*
- Retrospective control: the audit trail that responsible reliance requires (Koskinen 2024)

Takeaway: If AI-assisted discovery is collective work, memory is the collective's shared record.

Knowledge: what the project knows

- Literature, theory notes, concept definitions
- Codebooks and promptbooks: the measurement instruments in force
- Data documentation, source provenance, methodological references

Takeaway: LLMs have broad general knowledge but shallow project-specific knowledge.

Pin contested concepts to your definition

§2.2

- This project: trust = perceived competence, benevolence, and integrity
- Without the pin: psychology, computer science, and philosophy each offer a different “trust”
- Each borrowed definition produces a different codebook

Takeaway: Knowledge anchors the agent to your theoretical commitments, not its training data.

One instrument, two records: promptbook and promptlog

Promptbook — in kb/

- The current prompt set for coding or extraction (Stuhler et al. 2025)
- Includes system prompt and output-format instructions
- Always the version in force, one version at a time

Promptlog — in memory/

- Why each version replaced the last
- Dated audit findings and affected batches
- Accumulates instead of replacing

Comparing rules, memory, and knowledge

	Rules	Memory	Knowledge
Core question	How should I work here?	What has happened so far?	What should I reason with?
Core role	Governs conduct	Tracks trajectory	Grounds reasoning
Rate of change	Stable	Changes often	Accumulates
What it is not	Project history	Chat transcripts; knowledge	A log of attempts
Failure if missing	Messy files, skipped checks	Repeated failures, lost trajectory	Generic, shallow work
Typical artifacts	AGENTS.md, naming conventions, validation protocols	Run logs, decision records, promptlogs	Codebooks, concept definitions, promptbooks

Collapsing the three is the common failure

- One chat thread or notebook holding instructions, notes, and materials
- A failed earlier attempt read as a current instruction
- A verification requirement buried in a run log
- A key reference lost among outdated notes

Takeaway: The agent cannot tell constraint from history from evidence.

§2.3–2.4

From framework to folders

The directory should reveal the framework at a glance.

Rules can be hierarchical; memory and knowledge stay central

- Root AGENTS.md: project-wide expectations
- analysis/AGENTS.md and writing/AGENTS.md: local refinements
- memory/ and kb/: one copy, visible to every session

Takeaway: Global rules give consistency; local rules give precision; central memory prevents contradiction.

The framework as a directory

- AGENTS.md — rules: how the agent works
- memory/ — CURRENT, DECISIONS, OPEN_QUESTIONS, PROMPTLOG, runs/
- kb/ — concepts, literature, data-docs, codebooks and promptbooks
- data/, analysis/, writing/, outputs/ — the work itself, one home per kind of file

↗ **Chapter 2 templates** · cssprimer.org/ch02/

Takeaway: A person or an agent sees the project's logic before opening a single file.

data/: the separation that makes pipelines auditable

- raw/ — untouched source data, never modified
- interim/ — checkpoint outputs you can inspect
- processed/ — what analysis actually reads
- external/ — datasets from outside the project's own workflow

Takeaway: Auditable beats merely repeatable: every transformation leaves an inspectable trace.

Anatomy of a rules file

§2.4

Section	Question it settles	Failure it prevents
Scope of authority	Where may the agent act without asking?	Silent edits to raw data or memory files
File conventions	Where does each kind of output belong?	Outputs under invented names nothing downstream reads
Verification	What must be checked before an output counts?	Polished results no one inspected
Documentation	What must be written down, and where?	Decisions that cannot be reconstructed later
Handling uncertainty	What happens when the agent does not know?	A default assumption buried in a finished result

↗ **Root rules template** · cssprimer.org/ch02/templates/project-skeleton/AGENTS.md

The memory files, one page each

§2.4

- CURRENT.md: what is true now — no history, no definitions
- DECISIONS.md: what, why, rejected alternatives, downstream implications
- PROMPTLOG.md: entries that mark real changes to the instrument
- kb/README.md: an annotated map naming which files are authoritative

Takeaway: A memory file earns its keep by being read, not by being exhaustive.

§2.5–2.6

Tools, and getting started

Tool-agnostic files; a deliberate, minimal beginning.

The framework is deliberately tool-agnostic

§2.5

- Plain Markdown under self-explaining names; any tool can read it
- Planning, coding, and writing may each suit a different tool
- Context lives in the directory, not in any tool's chat history

Takeaway: Switching tools costs nothing when the project itself carries the context.

A minimal starting point: three files

§2.6

- AGENTS.md: file conventions, verification expectations, conduct under uncertainty
- memory/CURRENT.md: direction, open questions, immediate next tasks
- kb/README.md: a map of the materials you have
- Already using LLM coding? Add a promptbook and a promptlog

↗ **Project skeleton download** · cssprimer.org/ch02/templates/project-skeleton.zip

Takeaway: Three files answer the three core questions; that is enough to begin.

Keep it current, or it misleads

§2.6

- A three-week-old CURRENT.md is worse than no file at all
- Brief, regular updates beat comprehensive, infrequent ones
- Delegate the upkeep: agents maintain memory files well

Takeaway: The framework is a living system, not a one-time setup.

What the later chapters deposit into the framework

§2.6

- Ch. 4: codebook and data-source evaluation → kb/; workflow documentation → memory
- Ch. 6: run logs, promptlog entries, reconciliation memos → memory; promptbooks, linkage tables → kb/
- Ch. 8: the theory behind a centrality choice → knowledge; construction decisions → memory

Takeaway: Every exercise adds to the framework; the framework makes the additions cumulative.

Discussion

- In your own words, distinguish rules, memory, and knowledge. Why does collapsing them into one document undermine agent performance?
- Koskinen says we can rely on AI but not trust it. How does this framework make reliance responsible — and what risks remain even in a well-organized project?
- What verification requirements would you write into the AGENTS.md for your own project?
- If you revise an LLM coding prompt after inspecting false positives, what goes in the promptbook and what goes in the promptlog?

Key concepts from Chapter 2

Rules: Procedural constraints on how the agent works; stable; govern conduct

Memory: The evolving project state: decisions, direction, open questions; tracks trajectory

Knowledge: Substantive materials the agent reasons with: literature, concepts, codebooks

Thick trust vs. reliance: Trust requires accountability and explanation; opaque tools permit only reliance

Essential epistemic opacity: No one, developers included, can fully grasp how an LLM produces its outputs

Agencing: Actively configuring the boundaries of human and machine action in a project

Promptbook: The current prompt set operationalizing an LLM coding or extraction task; knowledge

Promptlog: The selective revision history of the promptbook, with audit evidence; memory

Materials on the companion site

Chapter 2 hub and templates cssprimer.org/ch02/
The downloadable project skeleton with the rules, memory, and knowledge files prefilled

Project skeleton (direct download) cssprimer.org/ch02/templates/project-skeleton.zip
The zip students unpack to start their own project

Getting started cssprimer.org/setup/
The tool installation this chapter's framework assumes

Where to go next

- Set up: the three core files for your own project — before next session
- Download: the complete project skeleton, every folder and starter file, at cssprimer.org/ch02
- Watch: the recorded session working a small study through this structure in VS Code
- Read: Chapter 3, the nature of data — source, structure, standardization