Computational & AI-Assisted Methods for Social Sciences

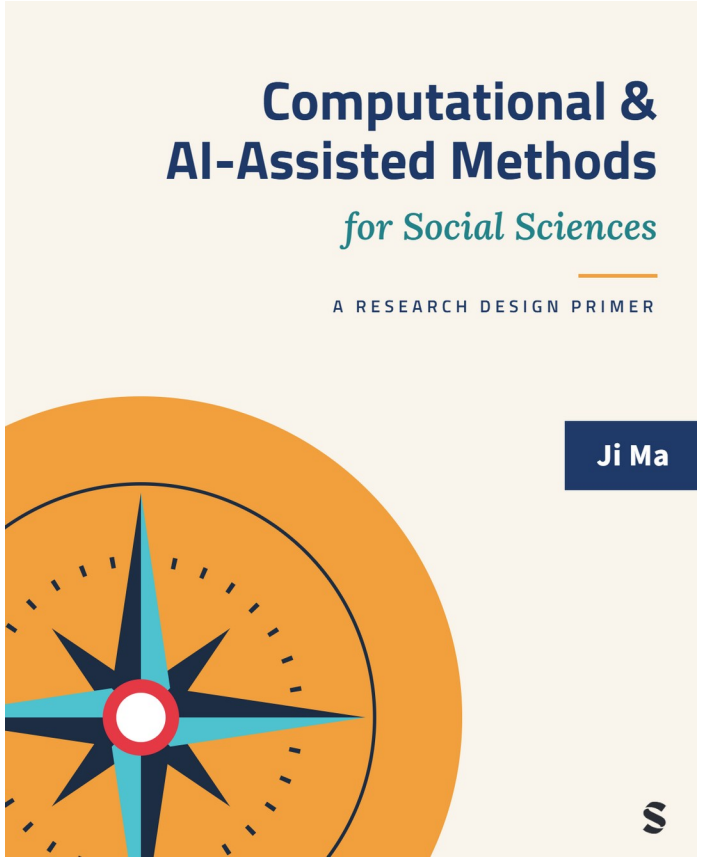
A Research Design Primer

Chapter 4

Gathering Literature in Your Field

Ji Ma

cssprimer.org



Learning objectives

By the end of this chapter, you can:

- Find candidate data sources through repositories, published studies, and informal networks
- Evaluate a source at four levels: raw data, preprocessing, content, retrieval
- Normalize retrieved records into tidy, relational tables documented by an ERD and a codebook
- Screen a corpus for relevance with an LLM and verify the screening with an agreement check
- Design and document a reproducible retrieval-and-preprocessing workflow

This chapter builds the dataset you will keep

- Exercises are built around literature in your own field
- The corpus returns in later chapters — and in your own projects
- Mirrors Chapter 3's structure: source → structure → quality → workflow

§4.1

Source: finding and evaluating data

Where data live, and the four-level inspection before you commit.

Where researchers actually find data

§4.1

- Formal channels: publications and data repositories (Gregory et al. 2020)
- Informal channels: colleagues, mentors, conference conversations
- Consider sources early; align questions with data you can realistically access

Takeaway: We practice on scholarly literature; the strategies transfer to any data type.

Data from existing studies

- Every empirical article has a data-and-methods section
- Open science mandates: journals and funders increasingly require shared data and code
- A peer-reviewed use helps you judge fitness for your purpose
- Read the methodology and documentation before reusing anything

Repositories, catalogs, and search engines

- Research repositories: ICPSR, Dataverse, OSF — documentation and quality control
- Open catalogs: Zenodo, Kaggle, GitHub
- Search engines: Google Dataset Search, Data Commons, re3data

Five ways repurposed data mislead (Twidale & Marty)

§4.1

- Labels lie: “Urban Residence” differs across regions and years
- Collection practices evolve — race questions shift from one box to many
- “Cleanup” overwrites historical values
- Automated systems silently alter records, erasing address histories
- The original purpose decided what was recorded at all

Takeaway: Understand why and how the data were collected before drawing conclusions.

Personal ties and informal sources

- Conference conversations surface little-known datasets
- Mentors and collaborators share unpublished data — and hard-won context
- Verify provenance, ethics clearance, and usage conditions first

Takeaway: Unique access arrives with documentation gaps; clarify terms before you build on it.

Evaluate a source at four levels

§4.1

- **Raw data**: who gathered it, for what purpose, by what method
- **Preprocessing**: what was imputed, recoded, merged — and by whom
- **Content**: which variables, in which formats
- **Retrieval**: API or scraping, volume, rate limits

Takeaway: Weeks of analysis can die on a sampling choice you never checked.

OpenAlex under the four-level lens

§4.1

- Raw: metadata from Microsoft Academic, Crossref, and others — each with its own history
- Preprocessing: duplicates removed, author names harmonized
- Content: bibliographic records, citation links, concept tags, in JSON
- Retrieval: free API and bulk snapshots; rate limits apply

Takeaway: Does its sourcing overlook the journals most relevant to your field? Always trace the roots.

Exercise: inspecting a data source

- Read the OpenAlex help documentation — no retrieval yet
- Work through the four levels: raw, preprocessing, content, retrieval
- Classify the source on Chapter 3's axes: format, security, interface
- Ground every claim in concrete examples from the documentation

You will need

- A browser and the OpenAlex documentation at help.openalex.org
- No data retrieval yet — this exercise decides whether the source is worth retrieving

Deliverable

A one-page source review answering the five prompts, ending with the three-axis classification and a stated decision on whether to build your corpus from OpenAlex.

↗ **Chapter 4 hub** · cssprimer.org/ch04/

§4.2

Structure: from JSON to relational tables

Normalization in practice: three normal forms, one diagram, one dictionary.

Two tools capture your data's organization

§4.2

- JSON is built for transmission, not analysis
- Strategy: tidy within each table, relational across tables
- ERD shows how tables relate; codebook says what variables mean

Takeaway: The diagram and the dictionary together make a dataset legible to others.

Normalization: store each fact exactly once

- User details live in one Users table; other tables hold only the foreign key
- Update a screen name in one place, not everywhere it appears
- Three normal forms steer the design: 1NF, 2NF, 3NF

Takeaway: Less duplication means fewer chances for your tables to disagree.

First normal form: one value per cell

- A Hashtags cell holding “#Healthcare, #Reform” cannot be queried or updated safely
- Fix: a PostHashtags table, one row per post–hashtag pair
- Now “which posts used #Policy?” is a simple filter

Takeaway: Atomic values are what make filtering and joining possible at all.

Second normal form: depend on the whole key

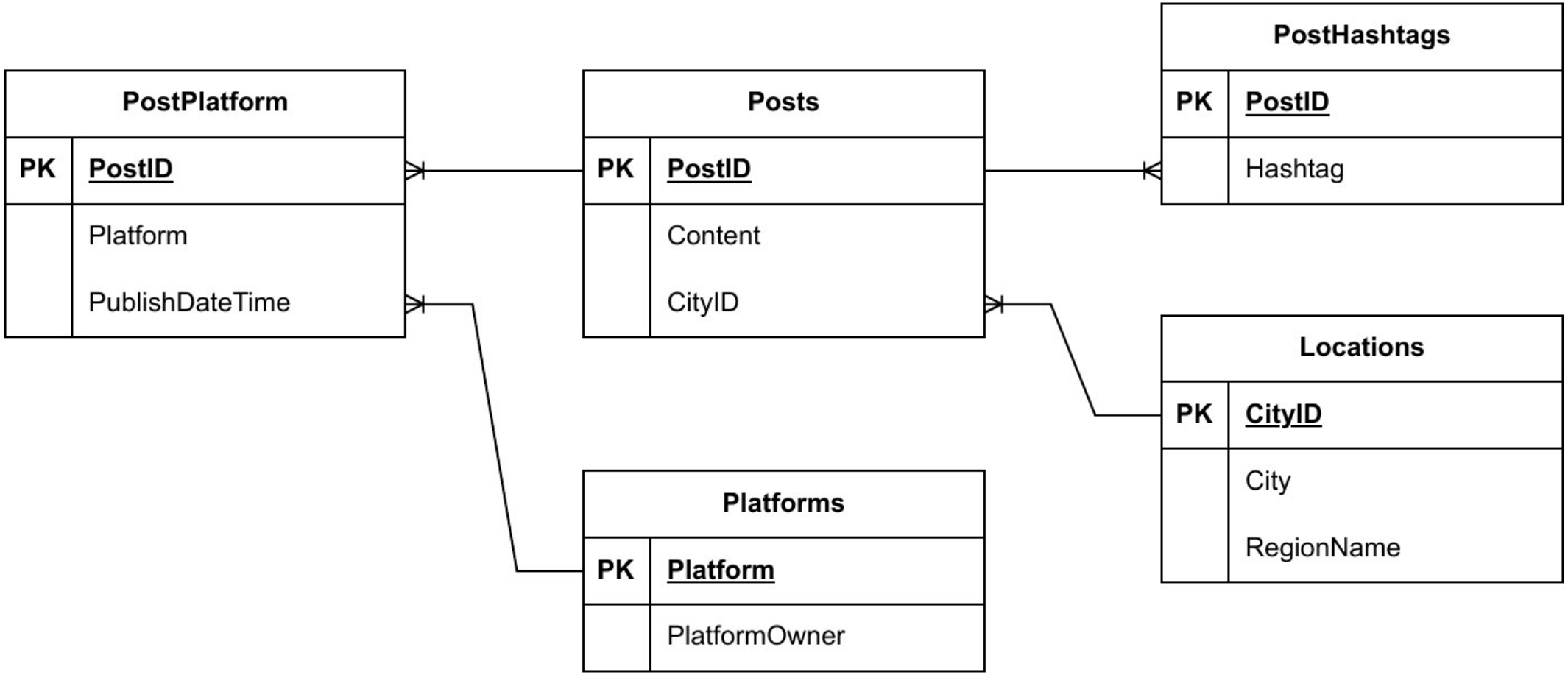
- Composite key {PostID, Platform}: publish time genuinely needs both
- PlatformOwner depends on Platform alone — a partial dependency
- Fix: a separate Platforms table keyed by Platform

Third normal form: no transitive dependencies

- PostID → City, and City → RegionName: a chain through a non-key attribute
- A spelling fix to “Seattle” should not touch many rows
- Fix: a Locations table keyed by CityID; Posts stores only the foreign key

Takeaway: City–region facts change once, in one table — and the payoff is integrity plus simpler queries.

The normalized tables, assembled into an ERD



Each box is a table with its columns; lines mark one-to-many links through primary and foreign keys.

Takeaway: One glance shows how to query the database for any analysis.

Exercise: drawing an entity relationship diagram

- Understand your JSON's content and structure — a JSON viewer helps
- Normalize into tables against the three normal forms
- Draw the ERD: MySQL Workbench, draw.io, dbdiagram.io, or pen and paper
- Optional: have an LLM draft the schema, then audit it against your own

You will need

- Your retrieved OpenAlex records in JSON
- A JSON viewer
- A diagramming tool — or pen and paper

Deliverable

An ERD of your normalized tables, plus a short note naming each table's unit of analysis and any structure you deliberately did not build.

Codebook — and promptbook

- The ERD shows relations; the codebook defines each variable's meaning
- Per variable: name, type, range or categories, coding rules, missing-value codes
- LLM-built variables get a promptbook entry: prompts, model version, settings, audit plan

Takeaway: Clear enough that another researcher needs no questions answered.

Exercise: creating a codebook

- Variable name and data type
- Range or valid categories
- Definitions and coding rules
- Missing values and special codes
- Promptbook entry wherever an LLM contributed

You will need

- Your normalized tables from the ERD exercise
- Any prompts you used if an LLM helped build a variable

Deliverable

A codebook covering every variable you intend to analyze, saved to kb/codebooks/ in the Chapter 2 project structure. Later chapters read from it, so build it carefully once.

§4.3

Quality: checking and improving data

Garbage in, garbage out — and the screening pass that keeps your corpus on topic.

Garbage in, garbage out — compounded at scale

- Errors propagate through millions of intricately linked records
- OpenAlex refreshes records and takes community-driven updates
- Responsibility for fitness stays with you, question by question

Exercise: inspecting data quality in OpenAlex

- Source: which APIs, dumps, and fields did you retrieve — and what do those choices exclude?
- Accuracy: missing DOIs, empty abstracts, authors without affiliations?
- Consistency: date formats and journal names standardized across records?
- Timeliness and cleaning: current enough? What did your cleaning change?

You will need

- Your retrieved OpenAlex dataset
- Your codebook — it tells you which fields matter enough to check

Deliverable

A one-page data-quality assessment covering source, accuracy and completeness, and consistency — each with a concrete example from your own records and a stated plan for handling it.

Keyword queries over-collect by design

§4.3

- “Civic participation” sweeps in patient participation and participatory sensing
- Systematic reviews screen titles and abstracts against explicit criteria
- An LLM can run the exhausting first pass — under the same discipline

Takeaway: Vague criteria produce confident, arbitrary screening; the model fills gaps with its own defaults.

Exercise: LLM screening with an agreement check

- Write inclusion and exclusion criteria, plus an “uncertain” option
- The model labels each record — include, exclude, uncertain — quoting its evidence
- Hand-screen 30 records blind; compute agreement; close-read every disagreement
- Accept, revise and rerun, or fall back to hand screening — and document the decision

You will need

- OpenAlex records with titles and abstracts
- Access to a language model
- A written statement of what your corpus is about; prompt templates at cssprimer.org/primers/prompting

Deliverable

The screened corpus plus a half-page screening note: criteria, agreement rate, disagreement diagnosis, decision. The note is part of your dataset's provenance.

↗ Prompting and its failure modes · cssprimer.org/primers/prompting

§4.4

Standardization with workflows

Make the whole pipeline explicit, documented, and runnable.

A workflow makes the process explicit

- Define query terms → retrieve → parse and store → clean and unify → merge with other data
- Each arrow is a documented, repeatable step
- Shared goals: effectiveness, reproducibility, reusability, traceability

Takeaway: If a step lives only in your head, the workflow does not contain it.

Document at two levels of detail

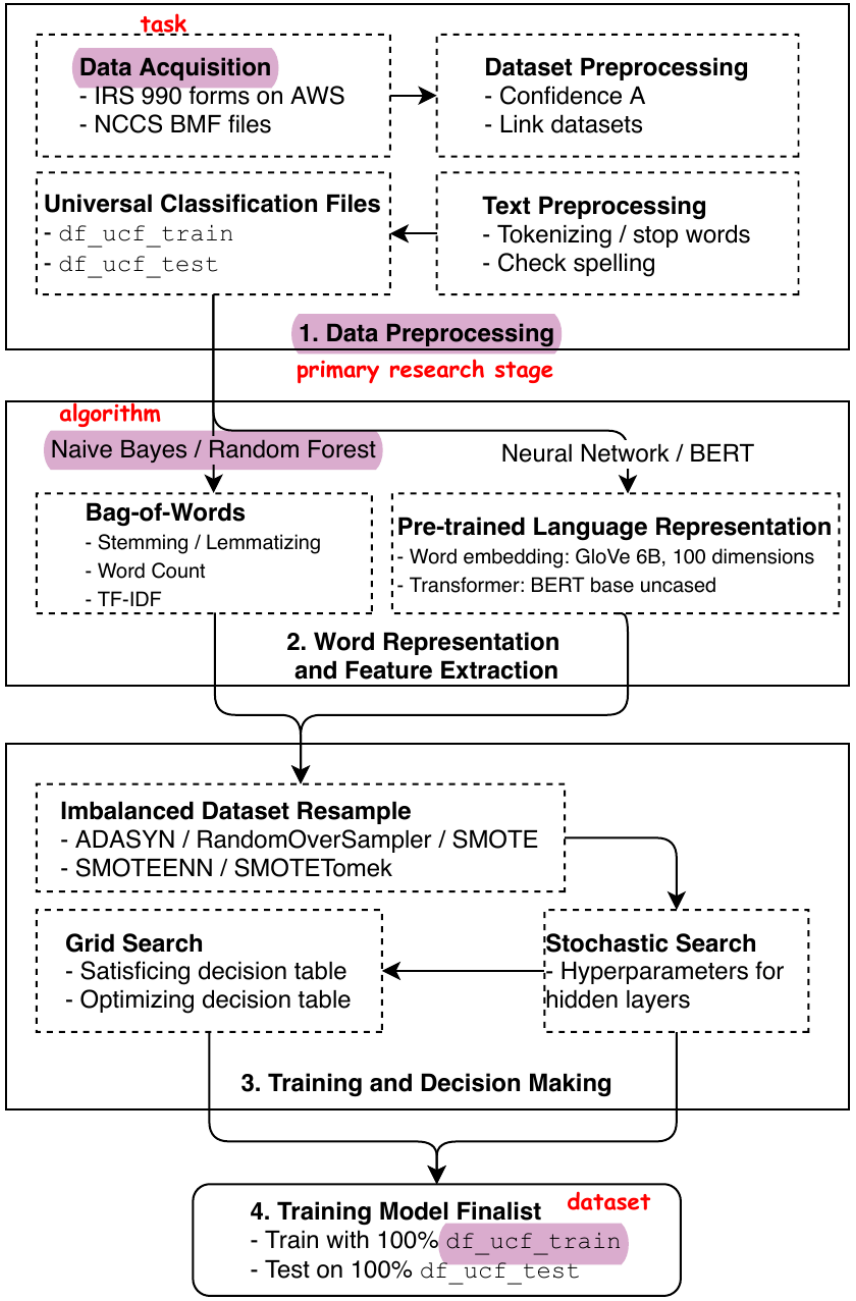
§4.4

- Macro: a diagram from OpenAlex to local database to analysis
- Micro: endpoints, filters like publication_date:>2020-01-01, scripts, cleaning rules, version control
- Quick orientation for readers; full depth for reproducers

Takeaway: A diagram without notes is decoration; notes without a diagram are a maze.

A real multi-stage workflow, published

Automated classification of U.S. nonprofits: stages, task-level detail, and component types in one diagram.



Source: Ma (2021), Fig. 1, Nonprofit and Voluntary Sector Quarterly. © SAGE Publications. Included for instructional use.

Takeaway: Every stage names its inputs, tools, and outputs.

Exercise: your workflow

- Define your scope: keywords, journals, or subject headings
- Draw the workflow with a short narrative for each component
- Name and organize: 01_OpenAlexQuery.R, raw_data/, cleaned_data/, a README
- Add a promptbook if LLMs helped with coding or validation

You will need

- The retrieval and cleaning steps you actually performed in this chapter
- A diagramming tool

Deliverable

A workflow diagram with a short narrative per stage, plus the named and organized project directory it describes — the documentation later chapters assume you kept.

Five workflow habits (Gentzkow & Shapiro)

- Make the directory runnable: one script rebuilds everything
- Make changes safe: version control; run the full pipeline before committing
- Let structure communicate: input, code, output; keyed, normalized tables
- Abstract repeated code; comment strategically, not extensively
- Track tasks outside email: owner, status, definition of done

Tools that carry the load

§4.4

- OpenRefine: bulk cleaning with a replayable operation history
- Spreadsheets: quick inspection — beware silent type conversions
- draw.io, dbdiagram.io, MySQL Workbench for ERDs; some generate the SQL
- GitHub and OSF for documentation, versioning, collaboration

↗ **Data-management tools** · cssprimer.org/tools/data

Takeaway: An updated tool list and repository catalog live at cssprimer.org/tools/data.

Discussion

- What did inspecting OpenAlex change about how you now read the data section of published papers?
- How much structure was enough? Where did normalization serve your research, and where did it become engineering for its own sake?
- What did the agreement check reveal? Would you trust the same workflow on a corpus you could not hand-verify?
- Could a classmate reproduce your dataset from your documentation alone? Name your weakest-documented step.

Key concepts from Chapter 4

Entity relationship diagram: Visual blueprint of how tables link through keys in a relational database

Normalization: Restructuring tables to store each fact once, steered by the three normal forms

Three normal forms: Atomic values; whole-key dependency; no transitive dependencies

Codebook: Defines how each variable is structured, coded, and interpreted

Promptbook: Records the prompts, model settings, and audits behind LLM-built variables

Agreement check: Blind hand-screening of a sample to estimate concordance with model labels

Rate limit: A cap on API requests that shapes retrieval time and design

Version control: One authoritative history of what changed, when, and by whom

Materials on the companion site

Chapter 4 hub cssprimer.org/ch04/

The exercise chapter's materials, prerequisites, and deliverable in one place

Data-management tools cssprimer.org/tools/data

OpenAlex, reference managers, and the rest of this chapter's toolkit

Getting started cssprimer.org/setup/

Where to send students whose environment breaks mid-exercise

Where to go next

- Revisit: Chapter 3's 3S framework — your corpus is now a worked case of all three
- Browse: the chapter hub — retrieval notebook, prompt templates, tools catalog at cssprimer.org/ch04
- Next: Stage 2, concept representation — Chapters 5 and 6 turn this corpus into measures