A draft outcomes table

6 Sep 2026

A small companion to Outcome harvesting, which argues that the harvest belongs to people. It does, and you cannot always get the people in a room about everything for as long as it takes. So here is the workflow that reads interview narratives and comes back with a draft outcomes table, along with what such a table is for and what it leaves out.

Work in progress. A first workflow rather than a method. It has run three times on the app's own example project, most recently on 5 September 2026, and never on an evaluation. One column in it is wrong in every row, which is reported below rather than fixed first. Comments welcome.

In short

A draft table is something for a workshop to argue with. It is not a harvest.

- **The workflow is three steps and already exists.** `frame` takes the sources, `harvest_outcomes` codes them, `tabulate_outcomes` counts. It is ordinary Rubicon, with nothing in it specific to this method beyond the wording of one instruction.
- **The coding uses Outcome Harvesting's own four fields**: who changed (actor), what changed (outcome description), why it matters (significance), and how the intervention contributed (contribution). Each row quotes the passage it came from, located in the document.
- **One run, on nineteen household interviews, produced 272 outcome rows.** Between four and thirty-two per document, naming 138 distinct actors, every quotation matching its source exactly.
- **It answers neither of the two questions the parent paper picks out.** Nothing declares the programme objectives in advance. An objective nothing was harvested against therefore cannot show up at all, which is the whole of the coverage question.
- **A harvest cannot supply a denominator.** The parent paper insists that a figure from a harvested set states its base. Nothing in this workflow does.
- **One column is wrong in all 272 rows.** The model was asked to write down which document each outcome came from and invented an answer 39 different ways, while the correct answer was on the row all along.

See also: Outcome harvesting, the parent paper, which is about which questions a machine should touch at all; Rubicon; Theory-based evaluation.

Intended audience: evaluators who have interview transcripts, a harvest to prepare for, and less workshop time than the method assumes.

What a draft table is for

The parent paper's position stands. Deciding what changed, and which changes matter, is a judgement by people who were there. A machine reading documents afterwards does not replace the room.

Drafting is a different act. A room that starts from a blank page spends its first hour remembering things. A room that starts from 272 candidate statements pulled out of the transcripts spends its first hour disagreeing, which is the work you wanted from it. The statements are proposals. Some will be activities rather than outcomes, some will be the same outcome twice, some will belong to nobody in the room. Striking those out is the harvest doing its job.

Two conditions make that a defence rather than an excuse. The table has to arrive marked as a draft, with the passages behind every row. Anybody can then see what the machine read and disagree with the reading. And the drafting cannot decide anything the room was going to decide: which changes are significant, whose contribution counts, and what the set as a whole says are all downstream of it.

The workflow

Three steps. Nothing in the engine knows this is outcome harvesting.

- `frame` takes the sources, in this case all nineteen in the project, and records that it took all of them.
- `harvest_outcomes` is a `code` step over those documents, chunked at 3,000 characters, with a floor of one row per numbered segment that a segment can fall below only by refusing in writing. Its instruction defines an outcome in the method's own terms, as a change in the behaviour, relationships, activities, policies or practices of an individual, group, community or organisation. Its columns are the method's own four fields.
- `tabulate_outcomes` is a `compute` step counting the rows, then grouping them by attributes of the source document.

The four coding columns are the part worth copying. `actor` is who changed. `outcome_description` is what changed in their behaviour, relationships or practices. `significance` is why that matters locally. `contribution` is how the project or its partners influenced it. Those are the fields a harvest fills in for each outcome, so a draft that fills in the same four can be read straight into the method rather than translated first.

What makes it a Rubicon workflow rather than a prompt is what surrounds it. The frame records that all nineteen documents were read, which stops the table being reported later as though it covered more or fewer. Every row quotes the passage that produced it. The quotation is located in the document rather than taken on trust. The whole thing is registered and dated before it runs.

What one run produced

Run `b3e03b11`, version 2 of the workflow, against `example-original-rubicon` on 5 September 2026. The corpus is the app's own example project: nineteen household interview transcripts, anonymised, from two provinces.

It produced **272 outcome rows** from all nineteen documents, between four and thirty-two per document, naming **138 distinct actors**. Every one of the 272 quotations matched its source document exactly. The two earlier runs, on version 1 of the workflow, produced 248 rows each.

The counting step grouped by attributes of the source rather than of the outcome: 163 outcomes came from interviews whose main respondent was a woman and 109 from interviews whose main respondent was a man, 132 from one province and 140 from the other.

Those two figures are worth pausing on, because they are the right arithmetic answering a question nobody asked. They say how the outcomes are distributed across the documents. A harvest wants to know how they distribute across objectives, actor types and years, which are properties of the outcome. The columns holding those properties were coded and are sitting in the table. Nothing in this workflow groups by them yet.

What a draft table does not carry

- **It does not declare any objectives, so it cannot show a gap.** The parent paper's second question, which objectives have nothing against them, is the one an evaluator cannot get any other way. It works only because the objectives are a declared list fixed in advance. This workflow declares none. An objective nothing was harvested against is therefore absent from the table rather than showing as a nought. Adding the list is most of the work of answering that question.
- **It does not state any base.** A harvest is a search rather than a sample. 272 means 272 statements found by one pass over nineteen documents. It does not mean that 272 outcomes happened. The parent paper insists every figure from a harvested set says that much. Nothing in this workflow says it, though Rubicon has a `note` step whose whole job is to.
- **It counts passages rather than outcomes.** Two rows describing the same change in two interviews are two rows. Deduplication is a judgement about whether two accounts are of one event, which is the room's work rather than the machine's.
- **It cannot tell an outcome from an activity.** Some of the 272 are descriptions of a service being delivered. The method's own discipline is that a change in what somebody does is an outcome and a thing the programme did is not. Nothing in this run enforces that.
- **It substantiates nothing**, has no drafting loop with the people described, and records no sense-making. Those are the parent paper's human steps and they are unchanged.

The column that is wrong in every row

Between version 1 and version 2 the workflow gained a `source_id` column, because a table of outcomes that does not say which interview each came from is hard to use. It was added to the coding instruction, which meant asking the model to write it.

It got it wrong 272 times out of 272. Across the run it invented 39 different values for a corpus of nineteen documents: 74 rows say `1`, 49 say `N/A`, 29 say `Document 1`. The rest are variations on `source_1`, `Doc1`, `document_1` and `unknown`. Not one row matches the identifier of the document it was actually drawn from.

The failure is not the model's. It was shown one chunk of text at a time and asked which document it was from, which is a question the chunk does not answer. Meanwhile the ledger already held the right answer on every row, because a quotation is stored against the source it was located in. That is how the province and respondent-sex figures above were computed at all.

So the fault is a design one, of a kind worth naming: a field was asked of the model when it was already a fact about the record. It is cheap to notice on this run and would be expensive to notice on a real one, because the column looks filled. Every row has something in it, and a table where a fifth of the entries say `N/A` reads as a table with some missing data rather than as a column that is entirely fabricated.

Next steps

- Take the source identifier from the record instead of asking for it, and take the same care with any other column that is a fact about the document rather than about the passage.
- Group by the coded columns as well as by the document, so the table answers questions about outcomes rather than about interviews.
- Add a declared list of objectives and the coverage question that goes with it, which is the point at which this workflow starts answering the parent paper's second question rather than only drafting.
- State the base in the output rather than in a paper, so a figure from a harvested set cannot be reported as though it had one.

- Put a draft table in front of people who know the setting, and record what they struck out. That number is the only real measure of whether drafting helped.