



GROUPING THE SECOND LEVEL

📅 1 Aug 2026

Grouping the second level when only the first level has a codebook

A common way to code is to fix the first level of the label with a codebook and leave the second level free. The coder, or the AI, picks a first-level factor from an agreed list, then writes underneath it what the respondent actually said:

```
Farm production; switched to drought-tolerant seed
Farm production; started irrigating the lower plot
Farm production; planted earlier after the training
Farm production; better yields from the new seed
```

This works well while you are coding. A codebook fixes the top level across coders and across sources, and the free second level lets you stay close to what people said. It stops working when you come to report. Forty second levels have appeared under one first-level factor, most of them said once, several of them the same thing in different words.

Zooming to level 1 does not solve this. It replaces the forty with one node, which is readable and almost empty. You usually want something in between: a handful of second-level groups under each first-level factor, each with enough citations behind it to be worth a sentence in the report.

Each first-level factor is a separate problem

The useful assumption is that the grouping problem is independent for each first-level factor. Take all the factors that start with **Farm production**;; group their second levels among themselves, and never let a factor move to a different first level. Then do the same for the next first-level factor.

The codebook has already done the hard part of deciding what belongs with what at the top, so re-opening that during a second-level tidy-up would undo it. And the second levels only make sense relative to their parent: **started irrigating the lower plot** groups with the other production changes under **Farm production**, and would group with something else entirely under **Household spending**.

The practical effect is that a big project becomes a set of small problems. Thirty first-level factors give you thirty groupings of a dozen or two items each, and each one is small enough to check by eye.

How many groups

Five groups per first-level factor is about the most that helps, however large the bundle. The reason is interpretation rather than tidiness. A group with fifteen citations behind it supports a sentence in a report. A group with two supports nothing, and eight thin groups are no easier to read than the forty labels they replaced. So aim for few groups with plenty of members each, and treat the pull towards more groups as the thing to resist.

Use fewer groups still on a smaller bundle. Ten distinct second levels support two or three, and a group needs at least two members, so a bundle of four labels supports two at the very most. A group of one is an ungrouped factor with extra steps; where something really will not join anything, leave it as it is and say so.

These are rules of thumb and they should give way when you have a reason. If the ten labels under a factor really describe five separate things, five groups is the right answer and the heuristic is wrong. The numbers are there to stop you splitting hairs, and they should never make you merge two findings into one.

What makes a good second-level group

The same test as for hierarchical labels generally, applied one level down. Writing **Farm production; irrigation** over four different specific changes licenses a reader to treat all four as evidence for irrigation. So the group label has to be something you would be content to report on its own.

The group label should describe a change or an event, in the same register as the labels it replaces. **Irrigation practices** is a heading. **Started irrigating** is a factor.

Groups should not mix desirability by accident. Where some of the second levels describe an improvement and others describe the same thing getting worse, either separate them or mark the polarity explicitly with **opposites coding** and a ~. Merging them quietly loses the finding.

Sentiment as an input, on the effect side only

If you have coded sentiment, it records something the labels do not: whether the respondent treated this outcome as good or bad. Use it, with one restriction. Sentiment belongs to the link and values the effect, so read it only from the links where the label appears as an **effect**. Where the label appears as the cause, the sentiment is somebody's valuation of something further downstream, and feeding that in mixes two different judgements.

Mixed sentiment inside a group is not automatically a fault. Sometimes it is the finding: some respondents approve of **Access to consumer goods; more smartphones** and others deplore it, so one group containing both records a real disagreement about value. Splitting it would invent a distinction nobody made.

Splitting is right in the other case, where the label is not expressed in even a semi-quantitative way and the sentiment is doing work the label should have done. **Level of education; maths education** names a topic rather than a change. If half its links are positive and half negative, the respondents were talking about two things: good maths education and poor maths education. Here splitting recovers a distinction the coding lost.

Which of the two you are looking at depends on the study and what it is for, so the grouping needs the aim of the study in front of it, not only the labels and the counts. In an evaluation of whether a programme improved teaching, split the maths group. In a study of what people value in their community, leave the smartphone disagreement standing.

This all assumes a project without much opposites coding. Where ~ is already doing the polarity work, sentiment and the marker can disagree, and that is a separate problem.

Checking what happened

Grouping is a recode, so look at a table of what became what: the original label down the side, the new group label across the top, the number of links in each cell. Did anything end up in a group you would not defend to a client? Did most of the bundle end up in one group, which usually means the grouping was too coarse?

Read it bundle by bundle, since that is how it was produced. A wrong assignment under **Farm production** is no guide to whether **Household spending** came out well.

Doing it in the app

Nothing here needs an AI. Grouping second levels is an ordinary recode, and on a small project you can read the bundle, decide the groups yourself and type them in. What the AI buys you is the drudgery: thirty bundles instead of three, each with twenty labels in it, is an afternoon by hand and a few minutes with a model.

Recode into a label set rather than over the top of your labels. A label set is a second cause and effect for every link, stored alongside the first, so the original labels stay where they are and you can see both versions at once. That gives you the before-and-after table. It also means an unconvincing grouping costs you nothing. When you are happy with it, promote the set to become the default labels. See [recoding labels temporarily](#).

Ask MapCat to group the second level under each top-level factor. It takes one first-level bundle at a time by itself, so you no longer narrow the view factor by factor. It applies the rules on this page: a few groups per bundle, with group labels that read as causes rather than headings. Name a label set for it to write into. Say also whether it should read the quotes. That costs more, because the price then follows the number of links instead of the number of distinct labels, but it is what lets **Level of education; maths**

education come apart into the good version and the poor one. Leave reading the quotes off for a cheap tidy-up of labels that are already clear.

The manual routes work too, and for a small project they are faster than explaining yourself to an AI. Narrow to one first-level factor with the factor label filter, so the AI sees one bundle and cannot move a factor across the boundary, then run the recode over what is left with an instruction saying how many groups you want and that the part before the semicolon must not change. Or sort the Factors table by label so the members of a bundle are next to each other, switch on Bulk Edit, then overwrite the second levels you want merged with the same text. See [bulk relabelling factors](#).

See also

- [Hierarchical coding](#) for the semicolon convention, what a roll-up licenses, and the guardrails this page inherits.
- [Recoding labels temporarily](#) for label sets.
- [Bulk relabelling factors](#) for the manual routes.