CHAPTER

Task 2 & 3 Key ideas and conventions

CONTENTS

Simplification - hierarchical zooming

> In the previous chapter Task 2 -- Introduction we looked at the main ideas of minimalist coding:

- we code links between simple propositions which we do not necessarily conceive of as variables which can take different states at different times.
- we let factor labels do most of the work, so we start off by distinguishing between, say `wealth` and `poverty`.
- we don't use separate metadata columns for factors. In fact, we don't even have a table for factors at all: our coding results only in a table of links.

That approach has general applicability. In this chapter we get down to some specific suggested conventions about how to make this approach more useful for answering concrete questions. These suggestions are all implemented in the Causal Map app, but of course other approaches are possible.

Our approach is algorithmic

All the pioneers of causal mapping used different algorithms to simplify, query and synthesise their data.

What we add: TODO

Simplification - co-terminal link bundles

In most use cases, the data contains many "co-terminal links": links with the same cause and effect. We call these "bundles" of links. It is often bewildering to show all of those links in a map so by default we condense them all into a single visual arrow, usually printing the number of sources and/or citations on the arrow and making the width depend on the number of citations.

Simplification - hierarchical zooming

✅ Simplifying causal maps with hierarchical coding

Summary

You can use the special separator ; to create nested factor labels, like this:

New intervention; midwife training ➔ Healthy behaviour; hand washing

We can read this separator as “in particular” or “for example”:

New intervention, in particular the midwife training,

Or we can read it in reverse like this:

The midwife training, which is an example of / part of the new intervention

Factor labels can be nested to any number of levels, e.g.

New intervention; midwife training; hand washing instructions

The higher level factors can, within the same coding scheme, themselves be used for coding.

So as well as creating links to and from New intervention; midwife training; hand washing instructions, you can always also use New intervention; midwife training and New intervention as factors too.

e.g. we could code “this whole new intervention has also led to happier health providers” like this:

New intervention ➜ Happier health providers

We can “zoom out” of a causal map containing nested factors to show a simpler, higher-level structure as a summary. This is done by applying an algorithm which re-routes links to and from the lower-level factors into their higher-level parents.

So then, loosely yet informatively and with certain caveats, accepting a loss of detail but affirming that the overall meaning is not distorted, this algorithm can deduce for us, from the first example above, the following causal map:

New intervention ➜ Healthy behaviour

Usually each higher-level factor will each be a summary of *many different* lower-level factors.

Introduction

An analyst coding text to create a causal map is confronted with the same challenge as any qualitative researcher: identifying recurring themes in such a way as to help a larger picture emerge, while retaining important detail. Expressing factor labels in a hierarchical fashion can help solve this problem. But hierarchical labelling also enables a particular strength of causal mapping: it lets us “zoom out” to view a whole causal map from a higher-level perspective, merging causally similar concepts to give a simpler map with far fewer components. Formally, the process of zooming out produces a map which logically *follows from*, is *implied by*, the original map. This chapter also introduces a smarter way to “zoom out” from a causal map, and explains how these features are implemented in the Causal Map app.

When conducting qualitative coding of any text, there is always a tension between wanting to keep the detail (e.g. hand washing) but also to code in such a way that general themes emerge (e.g. health behaviour). One way to do this is to organise the codes into a hierarchical structure, so that “Hand washing” is nested as part of “Health behaviour”. This can be done (see e.g. Dedoose, saturateapp.com) by using labels in which the hierarchy is directly expressed, for example Hand washing; health behaviour – using semi-colons or some other convenient character to separate the levels of the hierarchy.

This approach is convenient for several reasons:

- A search for “Health behaviour” will find Health behaviour; Hand washing as well as Health behaviour; vaccinating children and other combinations.

- It can be arbitrarily extended to any number of levels, e.g. Health behaviour; Hand washing; Before meals
- Related items appear next to each other when they are listed alphabetically
- The hierarchical structure does not require that the analyst (whether using paper-and-pencil or software) maintains a separate set of "parent" codes; the higher-level codes are simply whatever is visible before the semi-colons. Higher-level codes can be created and changed on the fly without having to open a separate codebook or software interface.
- It is possible to code directly at higher levels, for example using the code Health behaviour where no more details are given.

When reading a nested factor label aloud, the semi-colons could be substituted with "... and in particular ....", e.g. "Health behaviour, and in particular Hand washing, and in particular Before meals".

The way factor (labels) emerge during causal mapping is just a special case of the way codes emerge in any qualitative coding process, and nested coding is useful in ordinary qualitative data analysis as well as in causal mapping. However, hierarchical coding in causal mapping is particularly exciting because it allows us to do things like simplify a causal map to give a higher-level version of it with far fewer components.

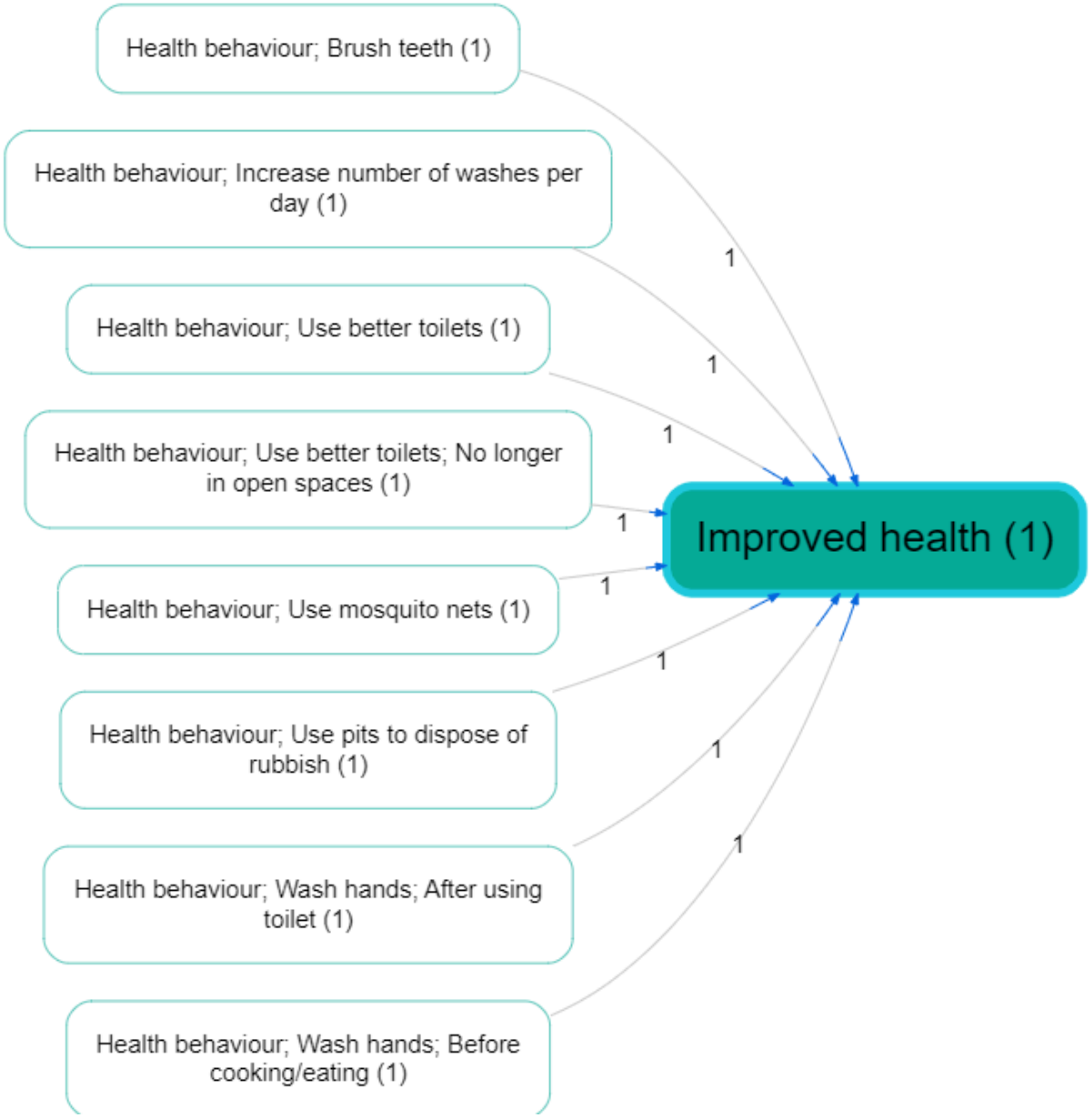
A factor can't belong to two different hierarchies

One limitation to this way of expressing the hierarchy as part of the factor label is that you can't make one factor belong to two different higher-level concepts. For example, you could understand a particular midwife training both as causally part of a new intervention but also perhaps as causally part of an institution's in-service training programme or an individual's workload, but you can't code it as both "New intervention; midwife training" and "In-service training; midwife training" at the same time.

This limitation is because of the meaning of the semicolon: the ; in Y; X means that this label can be replaced as needed with just Y , accepting a loss of detail but affirming that the overall causal story is not essentially distorted. If a hierarchical label had more than one parent, we wouldn't know which parent to "roll up" the factor into.

If you find yourself wanting to put a factor into more than one hierarchy, try using Factor label tags -- coding factor metadata within its label instead.

Zooming out



Assuming we have a causal map which has used hierarchical coding, as in the small map shown above, how do we take advantage of this coding to “zoom out”?

If we define the “level” of a factor as the number of semicolons in its label plus 1, here is the same map, zoomed out to level 2 (i.e. a maximum of one semi-colon per factor).

image.png

Here is the same map, zoomed out to level 1 (i.e. there are no semi-colons at all).

A warning: causal mapping as described here is a *qualitative* process. While zooming in and out can be very useful, it should never be used mechanically.

Zooming out is like making deductions with the ; separator

A causal map coded using a hierarchical separator can be "zoomed out" given a specific interpretation of the **;** separator, as follows.

If we know

New intervention; midwife training ➜ Healthy behaviour; hand washing

then, loosely yet informatively and with certain caveats, accepting a loss of detail but affirming that the overall meaning is not distorted, we can deduce:

New intervention ➜ Healthy behaviour; hand washing

and

New intervention; midwife training ➜ Healthy behaviour

and even

New intervention ➜ Healthy behaviour

This actually reflects the dilemma of the analyst often referred to as *granularity*: with how much detail should I code the beginning (or the end) of this causal story? Expressing a factor as Health behaviour; Hand washing; Before meals shows that this is indeed to be understood as a kind of health behaviour, although of course not the whole of it. By using this approach, the analyst says: if you are just looking for the global picture, I am happy for this factor to be understood as Health behaviour.

When factors are nested like this within one another as part of a hierarchy, it is possible to give an overview and 'zoom out' of the detailed data. This helps to simplify some of the analysis, enabling the user to focus on the links between the top-level groups rather than all the details. It follows that two factors like Y; X and Y; Z are *causally similar enough to one another to merge into Y* at a more general level.

Expressing a factor in a form like Y; X **means** it can be replaced as needed with just Y, accepting a loss of detail but affirming that the overall meaning is not essentially distorted. If you wouldn't be happy to accept this replacement, don't use the ";" to code this factor.

Semi-quantitative formulations work best

We already saw that causal mapping often works best when the factors are semi-quantitative. The hierarchical approach also works best when the higher-level factors are themselves labelled such

that also they are *semi-quantitative*, *causal* factors which could be used on their own – in a way which themes or categories see here could not. Good examples would be:

- Social problems
- Social problems; Unemployment
- Social problems; Addiction
- Psychosocial stressors
- Psychosocial stressors; Fear of job losses
- Psychosocial stressors; Pre-existing mental health issues

Here, Social problems and Psychosocial stressors are higher-level causal factors in their own right; they are not just themes or boxes to put factors into.

So we might have:

“The problem of unemployment is a psychosocial stress for many”

Social problems; Unemployment ➜ Psychosocial stressors

“When people get stressed they often turn to drugs“

Psychosocial stressors ➜ Social problems; Addiction

These could be combined into this story:

Social problems; Unemployment ➜ Psychosocial stressors ➜ Social problems; Addiction

If we zoom out of the above story, we could focus in on the higher-level factors and in this case we would get a vicious cycle:

Social problems ➜ Psychosocial stressors ➜ Social problems

Higher-level factors are *generalisations*

Usually, we don’t use higher levels merely to organise factors into themes which are not causally relevant.

Health; vaccinations law is passed

Health; mortality rate

These two items can be grouped into a *theme* “health” but can hardly be understood as special cases of a more general causal factor, so it would be a mistake to use the semi-colon. Instead, it would be more suitable to include the word “Health” just as a **hashtag**, without the semi-colon:

Vaccinations law is passed #health

Mortality rate #health

In other words, **causal factors in hierarchies should usually be semi-quantitative**.

Don’t mix desirability!

All the factors within one hierarchy should be desirable, or undesirable, but not both.

Generally speaking, make sure that the **sentiment of a more detailed factor is interpretable in the same way as the factor higher up in the hierarchy**. Ideally *all* the detailed factors within a hierarchy should be broadly *desirable,* or all *undesirable*, but not both. For example, you don’t want to nest something undesirable into something desirable. E.g. you don’t want to formulate a factor like this:

Stakeholder capacity; Lack of skills.

Because capacity would normally be understood as something desirable, and lack of skills would not. If you zoom out to level 1, this factor will be counted as an instance of Stakeholder capacity which is surely not what you want.

Try to use opposites coding and the ~ symbol to reformulate as:

~Stakeholder capacity; ~Presence of skills.

If you zoom out to level 1, this factor will *not* be counted as an instance of Stakeholder capacity.

Using higher level factors for “Mixed bags”

In spite of what we just said, sometimes you find you *have* use higher-level factors just to group a mixed bag, like this:

Politics; increase in populism

Politics; shift to the left

Politics; shift to the right

Politics; more engagement from younger voters.

The higher-level factor Politics is not in any sense a generalisation of these very disparate factors. However, we can at least think of it as a ‘mixed bag’. If we roll the map containing these stories up to level 1, we’ll see this ‘mixed bag’ factor Politics as a cause and effect of other factors. It will be a bit hard to interpret, but we can live with it as long as we remember that it is a mixed bag rather than a semi-quantitative summary.

Hierarchical coding as a way of coping with a large number of factors

Usually analysts are faced with the quandary of either having too many factors which they lose track of, or merging them into a smaller number of factors and losing information. With hierarchies, you can have your cake and eat it; it is similar to the process of recoding an unwieldy number of factors into a smaller number of less granular items, but with the advantage that the process is reversible; the information can be viewed from the new higher level but also viewed from the original, more granular level. For example, we can count that the higher-level factor

component “Health behaviour” was mentioned ten times, while retaining the information about the individual mentions of its more granular components.

Don’t use a hierarchy when a hashtag will do

When the analyst wants to group certain factors into a theme (like “health”) which is not itself a causal factor, hierarchical coding should not be used. Instead, text hashtags like “#Environment” or “[Environment]” or just “Environment” can be used to create themes simply by adding the text hashtag to the factor label, e.g.

Soil loss (Environment)

Eco training courses for NGOs (Environment)

Re-usable factor components as hashtags

Sometimes your factors relate to each other in ways which are not just hierarchical. For example:

- Activities completed; Training; Health
- Activities completed; Distribution; Health; First-aid kits
- Outcomes achieved; Health; First-aid skills

These are three (hierarchical) factors in which “health” appears in different places, and at different levels of the hierarchy.

This is not ideal, but sometimes it’s just the best way to organise a tricky set of factors.

In this example, “Health” appears only as a “component” of other factors. Although on its own it might look like a mere theme rather than a causal factor, it plays a role in differentiating the causal factors in which it participates, e.g. “Activities completed, in particular training, in particular on health”; and because “Health” is used across hierarchies, it can *also* be treated as a hashtag and can be used as part of searches, lists and counts of factors, etc.

Isn’t that a contradiction? Didn’t we just say that “Health” is not to be used on its own as a factor because it is just a theme and is not expressed in a semi-quantitative way? No, because here the word “Health” does not function as only a theme but as a way of differentiating causal factors: all the actual factor labels in which it participates are correctly expressed in a semi-quantitative way. So Activities completed; Training; Health is intended as a causal factor about the extent of completion of activities, in particular training activities, and in particular health training activities.

Causal mapping looks for linearity first

Causal mapping most often looks for linearity first, while of course being on the lookout for feedback loops and circular shapes. Whereas most systems approaches do the opposite.

Can you spot a complex system when you see one?

Version 1

The network pictured above, even though it is quite small, looks pretty tangled. We're not going to fully understand it, so we'd better get out our tools for dealing with complexity? But wait, look at the boring, old-fashioned hierarchy below.

Version 2

Did you spot that they have exactly the same structure? Now it is easier to see that it is just a hierarchy. D, E and F have one contributor each, whereas G and H share I, J and K as contributors, and feed only into B, whereas D, E and F all feed into both B and C, which feed into A. Easy. Nothing which should be too hard to predict, no balancing feedback loops.

"Complex" and "System" are very buzzy buzz-words at the moment. We should check we don't throw them around too much without thinking. I'm just reading Moore, Parsons and Jessop in the American Journal of Evaluation. They quote Magee and de Weck (2004) who define complex systems as systems "with numerous components and interconnections, interactions or interdependence that are difficult to describe, understand, predict, manage, design, and/or

change." Well yes, kinda. But what if you find a system difficult to describe, etc, just because you didn't look hard enough?

Yes, causal maps are just concept maps with only one type of connector, and that connector means "... causes....". Whereas concept maps can have any type of connector you like. Historically, causal maps come from concept maps.
Laying out causal maps is a challenge! Most folks from the systems tradition like swirly circular layouts which make them look like everything is one big feedback loop. If there is a more linear structure, we recommend showing that linear structure.

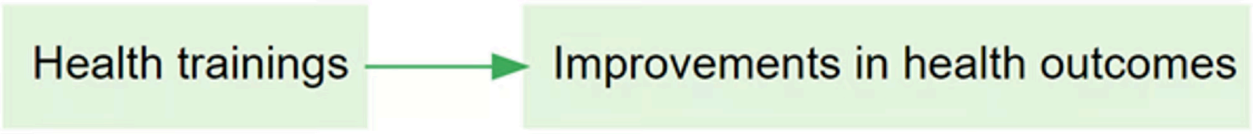
➕➖ Opposites

Introduction

In the first part of this Guide we have dealt only with undifferentiated links which simply say “C causally influences/influenced E” or more precisely “Source S claims/believes that C causally influences/influenced E.” We call this “barebones style” causal mapping. There is nothing more to this kind of causal map than links between factors. No other features are used.

Barebones-style mapping can be interpreted in two different ways:

- As in QuIP, to show causal influences between past events. It is an open question to what extent these causal claims can be generalised. It is possible to parse a link from C to E as saying not only that C is something which has the causal power to influence E but also that in some sense C happened and did in fact influence E, i.e. made a difference to it.
- To show only causal influences between factors, without recording what did or does happen.



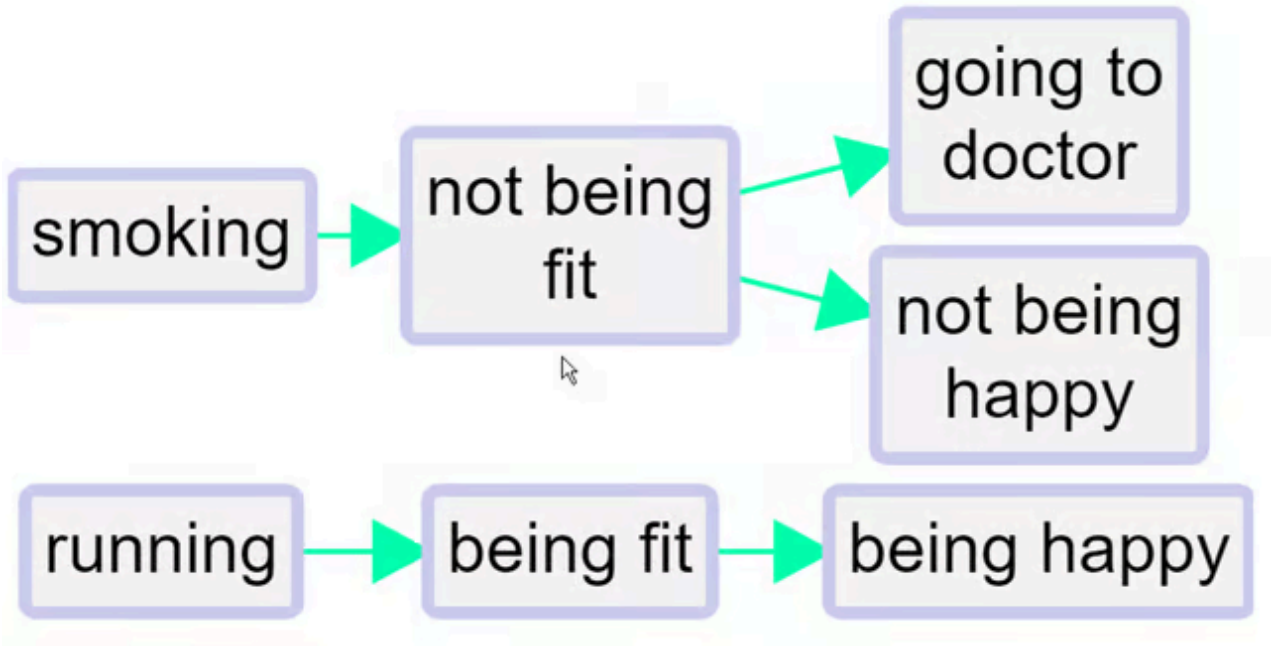
In Part 1 of this Guide we did in fact also introduce an additional convention, hierarchical coding, in which the “;” separator is used to encode the idea that C; D can be read “D, an example of C” and that at a suitable level of abstraction we can approximate C; D as C.

Combining opposites

This section presents a simple and powerful way of dealing with “negative” factors such as `poor health` which are in some sense the opposite of existing “positive” factors such as `good health`.

In some kinds of causal mapping and systems diagramming, more sophisticated approaches are used in which the factors are considered to be variables and the links between them can have positive or negative strength. The approach we present here is a simpler alternative.

Here is an example of quite minimalist QuIP-style coding. There are the beginnings of some ideas about (and issues with) polarity: for example, we have `fit`and `not fit`.



We've all done this kind of coding, with classic examples being coding for both employment and unemployment or for both health and illness. This could for example be two different stories about two different people; or it could be different aspects of or periods within one person's life.

This minimalist style on its own is unsatisfactory. We haven't told the app that being fit is represented with a somehow positive and somehow negative factor. So can't join them up. We can't compare the way that being fit leads to happiness and on the other hand not being fit leads to unhappiness (and to visiting the doctor). **We can't for example deduce that running might make visits to the doctor less likely.** Also, if we produce a table or do other analyses focused on healthy habits, we might miss data on the closely related unhealthy habits.

The first step forwards is to follow this convention:

> To signal that two factors are opposites we formalise the idea we already instinctively used in the above example, where we used the word "not" for one of each pair. Formally, we will code them in the form "~Y" and "Y." The ~ may appear at the start of a factor label. This already ensures that when we search for "Y" we will also find "~ Y."In the Edit Multiple Factors panel, these two factors will be listed next to each other - the alphabetical listing will ignore the ~.

We talk about *opposites* rather than positive/negative or plus/minus because that frees us from any implications about valence or sentiment: smoking is the opposite of not smoking, health is the opposite of not health / ill health / illness.

Where there is some kind of valence or sentiment involved, we do suggest using the ~ sign for the negative member of the pair. But it wouldn't make any difference to the app.

So the same map would look like this, using

```
~
```

instead of

```
not
```

.

Non-hierarchical coding with opposites is easy:

- Eating vegetables
- ~Eating vegetables
- Smoking
- ~Smoking

When you use the “combine opposites” filter (switched on),

the app tries to combine any pairs of factors which are opposites. It looks at all the factors which begin with a ~ and takes off the ~ where there was one. But it only does this if there is in fact such an opposite already coded in the file as currently filtered, otherwise there wouldn’t be any point.

So now there are for example two factors combined into the “fit” factor and two into the “happy” factor. The “not” factors have their incoming and outgoing links preserved, but when a factor is flipped to match up with its opposite, the part of the link next to that factor are now coloured pink. So the lower link from fit to happy is pink because the factor at each end of the link has been flipped from “~Y” form to “Y” form; the influence factor was originally *not fit* and the consequence factor was originally *not happy*. So there is no danger of thinking that this is really just another case of the other link, i.e. of fitness leading to happiness.

So, a link has two polarities: a *from* polarity and a *to* polarity. If the signs of the two polarities are opposite, then the effect of the influence factor on the consequence factor is reversed.

Both links from fit to happy have the same overall polarity (normal, not reversed) but they do not represent the same information.

No information is lost when you press the "combine opposites" button; you can still always read off the original map from it.

Opposites coding within a hierarchy

When using hierarchical coding, the sign "~" may appear at the start of a factor label *and/or at the start of any component in a factor label.*

Here is a similar story, now coded hierarchically. In this example, we only see ~ at the beginning of the factors, not yet within them.

When you press "combine opposites," the app tries to combine any pairs of factors which are opposites. It looks at all the factors which begin with a ~ and *flips each component*, taking off the ~ where there was one, and inserting one where there was not. But again it only does this if there is in fact such an opposite already coded in the file as currently filtered, otherwise there wouldn't be any point, because there is nothing to combine.

Here is the same example, but also "zoomed out" to the top level.

A quantitative social scientist might solve this problem by flipping the polarity of the negative examples, coding them as positive but using minus strengths for the connections. So smoking influences good health but with a minus strength. However this always seems somehow unsatisfactory and is complicated to do. It is particularly unsatisfactory when *both* ends of the arrow are flipped in this way so that we code the influence of being unfit on being unhappy as a green arrow from fitness to health!

By default in print view, links in the same bundle, i.e. with the same from and two factors, are no longer always displayed as one, with the frequency noted as a label. If we were using the quantitative approach, some of the links in the bundle may have minus rather than plus strength, etc, and we would have to somehow form some kind of average to arrive at an overall strength score, which is not at all satisfactory. Now, the links are only counted together if they have the same *from* and *to* polarities. So there can be up to four different links from one factor to another in Print view.

We are deliberately **not** falling Into the trap of somehow trying to aggregate the different *strengths* to say for example “there are 6 plus links from advocacy to compliance and 1 minus link so this is like 5 plus links because 6 - 1 = 5.” We don’t have evidence for an aggregated strength; we have aggregated evidence for a strength. Aggregating different pieces of *evidence* for links with different strengths is not the same as aggregating links with different strengths. So our more conservative approach preserves information.

It’s also possible that someone says “I know this intervention works not only because the intervention made me happier but also because I saw the people who didn’t get it and they are definitely not happier as a result.” In this case, we might code both arrows, intervention ➜ happy and not intervention ➜ not happy.

Opposites coding within components of a hierarchy

Sometimes we need to use the ~ sign within the components of a hierarchy.

- ~Healthy habits; ~eating vegetables

is the opposite of

- Healthy habits; eating vegetables

Not eating vegetables, which is an example of unhealthy behaviour, is the opposite of eating vegetables, an example of healthy behaviour.

- ~Healthy habits; smoking

is the opposite of

- Healthy habits; ~smoking

Smoking, which is an example of unhealthy behaviour, is the opposite of not smoking, an example of healthy behaviour.

So here we add one more causal claim to our above example, at the bottom:

The healthy habit of not smoking leads to being fit.

So the app correctly detects that not smoking is the opposite of smoking:

The two arrows at bottom left (one all green, one all pink) show that there is one example of this particular healthy habit leading to fitness, and the complementary example in which the opposite

of this habit leads to the opposite of fitness.

Zoomed out to the top level:

Bivalent variables?

Note that this is the mutually exclusive condition from classical logic, but we don't have any mention of an exhaustive condition: for us, it is not the case that everything has to be either wealthy or poor.

We could imagine that in a causal map, even before we even think about which specific factors might be opposites of other specific factors, each and every factor is really a kind of two-value factor like wealthy (as opposed to non-wealthy) or receiving tuition (as opposed to not receiving tuition). This is the quantitative way of thinking, in which every factor is really a variable which takes at least two values. But this is an unnecessary complication for us. The important point is that usually people don't think of the absence of something as having causal powers (though there are exceptions).

Suppose we are trying to code someone's understanding of income. They tell us that if when people are wealthy, they tend do certain things, and when they are poor they tend to do other things. To some extent, these two sets of things are themselves the reverse or contrary of one another. For example wealthy people will probably have the best education and poor people the least. Sometimes the sets of things we associate with the two opposite poles are not obviously a simple reflection of one another. For example, poor people are often hungry whereas non-poor people are not. But in an affluent country, not being hungry is probably not an important feature we would think of to describe being wealthy as opposed to not-wealthy, if we can assume that non-wealthy people are *overall* rarely hungry. We are more likely to think of things like eating often in posh restaurants.

So being wealthy and being poor are a good example of factors which we *should* consider coding as opposites. We can call the resulting factor, when we press the combine opposites button, *bivalent*.

, we could imagine there was a middle point too:

- Wealthy
- Not-wealthy / Not-poor - a kind of midpoint
- Poor

This is the usual case.

Which pairs of factors should we consider for opposites coding?

Short answer: use opposites coding for a pair of factors X and Y (i.e. recode Y as ~X or recode X as ~Y) if both X and Y naturally occur, separately, in the coding, but they can be considered, broadly speaking, as opposites of one another:

- in particular, it wouldn't normally make sense to apply both of them at the same time in the same situation (you can't be both wealthy and poor in the same sense at the same time)

If in doubt about which of the pair to recode, we usually pick X as the primary member of the pair if it is:

- usually considered as positive / beneficial / valuable
- and/or usually associated with "more" of something rather than "less" of something.

⚒️ Colour of factor borders (not currently implemented)

But there is more: the border colour could reflect the overall polarity of the factor.

One possibility is: The more *plus* incoming and outgoing arrows there are, the greener it is, and the more *minus* arrows there are, the redder it is. If the balance is equal, the border is light grey. If a factor has a red border, that means that at least mostly, its *opposite* was mentioned. So in this example, happy has a grey border because it was mentioned once and its opposite was mentioned once. Fit has a slightly pink border because it has one incoming and one outgoing plusses, and one incoming and two outgoing minuses.

However this reflects information which is actually already visible in the diagram. A second possibility is to use the same green-red scale to reflect the proportion of factors contributing to a factor which have been flipped: this information can not otherwise be deduced from the map. It becomes particularly relevant when coding opposites within a hierarchy, see below. However this colour scheme might be misinterpreted, as it is only affected by the number and polarity of factors which have been collapsed and not by the number of links.

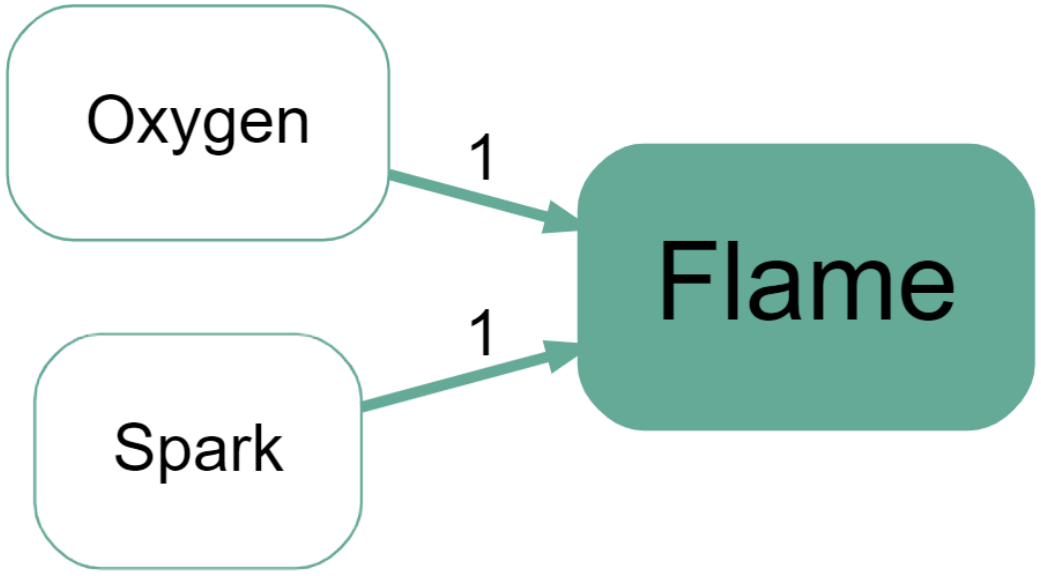
The total of the numbers on the arrows (here all the numbers are 1 so they are not shown) into and out of a factor is its citation count. But now we have additional information and its citation count is also equal to the number of times it was mentioned in plus form and the number of times it was mentioned in the opposite form.

Context

In this section, we'll look at "Context" as it appears to us at Causal Map when we do actual (qualitative) causal mapping: taking causal claims which real-life stakeholders actually make and trying to encode them in as systematic a way as possible. We believe that doing qualitative causal mapping is a really good testbed for theoretical ideas in evaluation and social science: do they fit with what people actually say?

So, how do we code this in causal terms: "When enough oxygen is present, a spark will always cause a flame"? (How) can we distinguish between a context like Oxygen and just some causal factor like Flame? How do we encode a context in a causal map?

We can draw a line from Spark to Flame, but what do we do about Oxygen? Drawing a line also from Oxygen to Flame doesn't seem to capture the context-ness of Oxygen.



Version 1: context is just a causal factor

In general terms this diagram is ok.

But we know the relationship between Spark and Oxygen to Flame is a *partial* function:

Oxygen	**Spark**	**Flame**
Yes	Yes	Yes
Yes	No	No
No	Yes	?
No	No	?

We know how Spark controls Flame only *given* Oxygen, but not when there is no Oxygen. This suggests that we cannot deal with a contextual factor as an ordinary causal factor. It is a special one which causally *enables* causal link(s) between other factors.

This *lack* of information about what happens in the *absence* of the contextual factor means that it acts like a *sufficient* condition for the causal relationship. (The defining characteristic of a sufficient condition is that no claim is made about what happens in its absence, just as the defining characteristic of a necessary condition is that no claim is made about what happens when it is present (only when it is absent).

Inside the context Oxygen, Spark is a necessary and sufficient cause of Flame (or so our respondent tells us). Outside the context C, Spark never makes any difference to Flame (or, we have no information about the effect, which isn't the same thing, but it doesn't matter here.)

Perhaps it is this very absence of information about what happens with no Oxygen which makes Oxygen feel more like a context rather than an ordinary causal factor. You can't see that in this first diagram.

Of course there are other things which you could mention as part of the context: there is fuel, it's dry, etc.

There are many ways to encode contextual information in a Causal Map; here we suggest storing the information inside each link.

Using context in the Causal Map app

At the moment, you can encode context in the Causal Map app simply by creating a hashtag for it. You can use a family of hashtags by using some characters in common, like "Context:", like this:

You can also search by and filter for contexts ✨ Transforms Filters: Include or exclude hashtags, so you can show a map only in the context of oxygen or without that context.

Plain coding

Summary

Causal mapping doesn't usually deal with the kind of non-causal themes which are the focus of ordinary QDA (like in NVivo!). However sometimes it can be really useful to be able to simply note the presence of something without any causal connection.

We call this "plain coding". You can use it for:

1. Noting the presence of something which is not mentioned as part of a causal link in the statement you are coding but does appear *elsewhere* as a causal factor as part of a causal link.
2. ...Or noting the presence of something which is "nothing but" a theme and never appears in causal coding.

Causal Map makes this possible in a simple way: we define a "plain coding" as simply a link from a factor to itself with has the hashtag `#plain_coding`.

Factors involved in plain codings (whether some of the time or all of the time) can of course still be involved in hierarchies and opposites coding just like any other factor.

So by default, plain codings will still appear in maps, as self-loops from the factor to itself, although it is possible to exclude such links by using the appropriate transforms filter.

Doing plain coding like this has the big advantage that factors coded with plain codings will still appear in the tables in the ordinary way, so for example the plain coding to (and from) the factor increased income will still count towards its source and citation counts.

It also means that you can easily delete a plain coding from a map by clicking on the link.

The hashtag `#plain_coding` distinguishes plain codings from actual causal links where a factor indeed influences itself, for example if someone says that increased income led to even more increased income – in this case the hashtag is not used. This hashtag can also be used together with other hashtags for the same link in the usual way.

How the Causal Map app implements this:

See [[#causal-overlay]]

Table features -- Statistical tests of group differences

Summary

How can we compare different groups like districts, gender or questionnaire sections, within maps and tables?

Reporting global and local network statistics

TODO

Simplification - factor and link frequency

Simplification - hierarchical zooming

TODO

