The submission "Making, Updating, and Querying Causal Models using CausalQueries" is a revised version submitted after initial screening. The authors were asked to add an illustrative example, provide complete replication material and improve the implementation, in particular to avoid internal structures as well as to ease post-processing the results.

Overall we feel that the submission did not suitably address the issues raised, in particular regarding improving the implementation by including a better exploitation of R's classes and methods system which should enable users to easily adopt use of the package but also inspect the fitted object and in this way contribute to improving data analysis and statistical modeling.

We decided not to send the current submission into review because the chances for successful publication were still deemed too low.

Specific comments:

o The main class of the package is "causal\_model", which has print, summary and plot methods. However, all entries of these objects have their own class but not method to query them. Hence, the internal structure of causal\_model objects is exposed and a number of methods included in the package are dedicated to parts of this internal structure, which is very difficult to follow in practice.

@all there is a core thing here that I don’t understand: why do they go into the internal structure at all when they are not required to

Note I do NOT think they have a problem with grab; they don’t mention it

@Till what’s weird is you never really have to pull anything from this model as you work directly with update\_model and query\_model for most actions. These objects are really all internal in a sense.

For instance:

These two do not have any methods: "parents" "stan\_objects"

@Lily please add print methods to these two

> class(lipids\_model$P) [1] "parameter\_matrix" "data.frame" > methods(class = "parameter\_matrix") [1] print see '?methods' for accessing help and source code

I don’t know what the problem is but wonder should we have a summary method for all df objects?

@Till can you see if there’s a way to do that compactly?

> class(lipids\_model$parameters\_df) [1] "parameters\_df" "data.frame" > methods(class = "parameter\_df") no methods found

This is a typo by the reviewer

> class(lipids\_model$dag) [1] "dag" "data.frame" > methods(class = "dag") [1] print

For the last one, at least, a plot method would make a lot of sense.

@Till

We have an issue here that "dag" does not contain the information we need for plot\_model

I wonder should we remove or rename model$dag.

We only use it for accessing the list of parents. So we could just have a list of parents as an attribute on model$parents\_df

Here’s a function to generate the list and replace get\_parents

generate\_node\_parents\_list <- function(df) {

# Initialize an empty list

node\_parents\_list <- list()

# Iterate over each row of the data frame

for (i in 1:nrow(df)) {

node <- df$node[i]

parent\_nodes <- df$parent\_nodes[i]

# Split the parent nodes by comma and remove any whitespace

parent\_nodes\_list <- unlist(strsplit(parent\_nodes, ",\\s\*"))

# Assign the parent nodes to the corresponding node in the list

node\_parents\_list[[node]] <- parent\_nodes\_list

}

node\_parents\_list[df$parent\_nodes != ""]

}

Need to be sure it goes in the right order and does not break anything

Since these classes all inherit from data.frame, trying to plot or summary can results in errors or irrelevant outputs.

@Gosha can you see what breaks

I saw no problems with printing here

lipids\_model <-

make\_model("Z -> X -> Y; X <-> Y") |>

update\_model(lipids\_data)

lipids\_model |> lapply(print)

however plotting problems here:

plot(lipids\_model[[2]])

Finally, for a model class, causal\_model clearly needs to have more methods (at least confint or alike for instance).

@all

OK so this is not like other model classes which are for estimation outputs not for models in our sense.

Confint makes no sense for ussomething similar could be applied to a query output but it’s a bit silly

So say people expect this. Should we provide a \*message\* as part of summary(model) that advises people that for estimates they should use query(model): “This is a summary of the model structure. To generate estimates for queries of interest please use query\_model” or something similar

Or we give more informative messages here:

|  |
| --- |
| > confint(model)  Error in UseMethod("vcov") :  no applicable method for 'vcov' applied to an object of class "causal\_model"  > coef(model)  NULL |
|  |
| |  | | --- | | > | |

Alteranative (but maybe silly) we could provide a kind of default query. E.g the effect of node 1 on node n under priors. But that could be slow as well as silly.

o Loading the package gives

> library("CausalQueries") Loading required package: dplyr

Attaching package: ‘dplyr’

The following objects are masked from ‘package:stats’:

filter, lag

The following objects are masked from ‘package:base’:

intersect, setdiff, setequal, union

Loading required package: Rcpp

This raises the question why it is not sufficient to have dplyr in Imports and thus avoid masking base R functions.

@Lily can you clean up these imports?

o The output of

> help(package = "CausalQueries")

lists also internal functions. This can be avoided by adding keyword internal to the respective man pages.

@ Till can you look at this

o The manuscript lacks a conclusion / summary section at the end.

@Macartan add