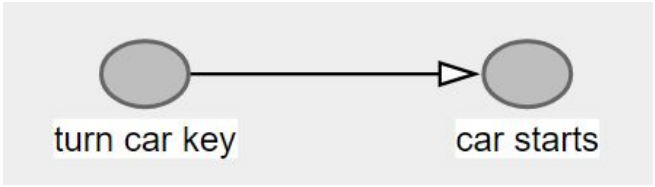
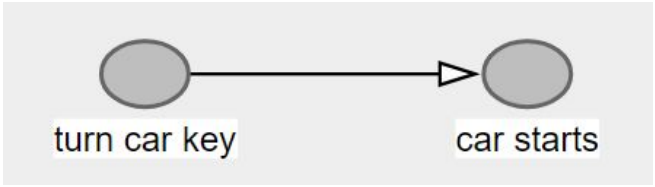
How I form causal conclusions

- Make ultimate conclusion

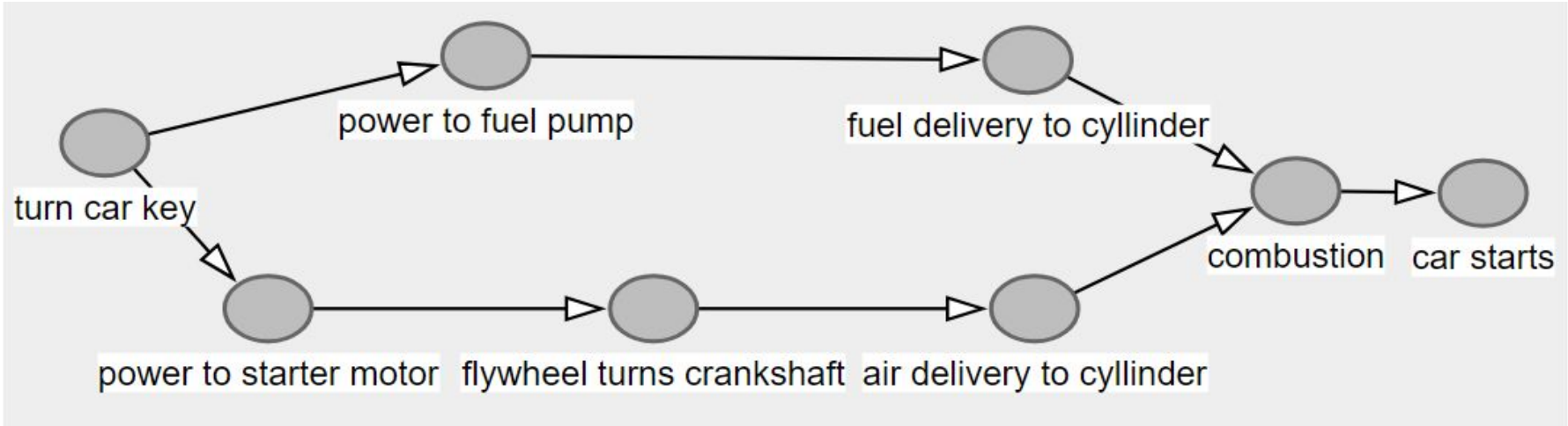


How I form causal conclusions

- Make ultimate conclusion



- Investigate intermediary steps to see if initial conclusion is true



How I form causal conclusions

- This approach is helpful in day to day life when not all variables are not immediately known.
  - I do not need to know how a microwave works in order to conclude that pressing the minute button will heat up my food.
- This approach is not helpful in the context of this project. If we have all variables involved in a process, how can we determine their relationships?

How my algorithm forms causal conclusions

Part 1: forming undirected skeleton

- Start by connecting every node to each other through an undirected edge
- Initialize **n** to 0
- For each pair of adjacent vertices (variables) **a** and **b** in the graph:
  - Create set of all neighbors to **a** and **b.** If this set is larger than **n….**
    - Check whether **a** and **b** should be direct neighbors (dependent) or if they are connected through a longer path (**a** → **c** → **d** → **b**) by using conditional independence tests.
    - If a test shows **a** and **b** are independent, remove that edge
    - Increment **n**
  - End loop when no edges are removed during an iteration

How my algorithm forms causal conclusions

Part 2: forming directed edges

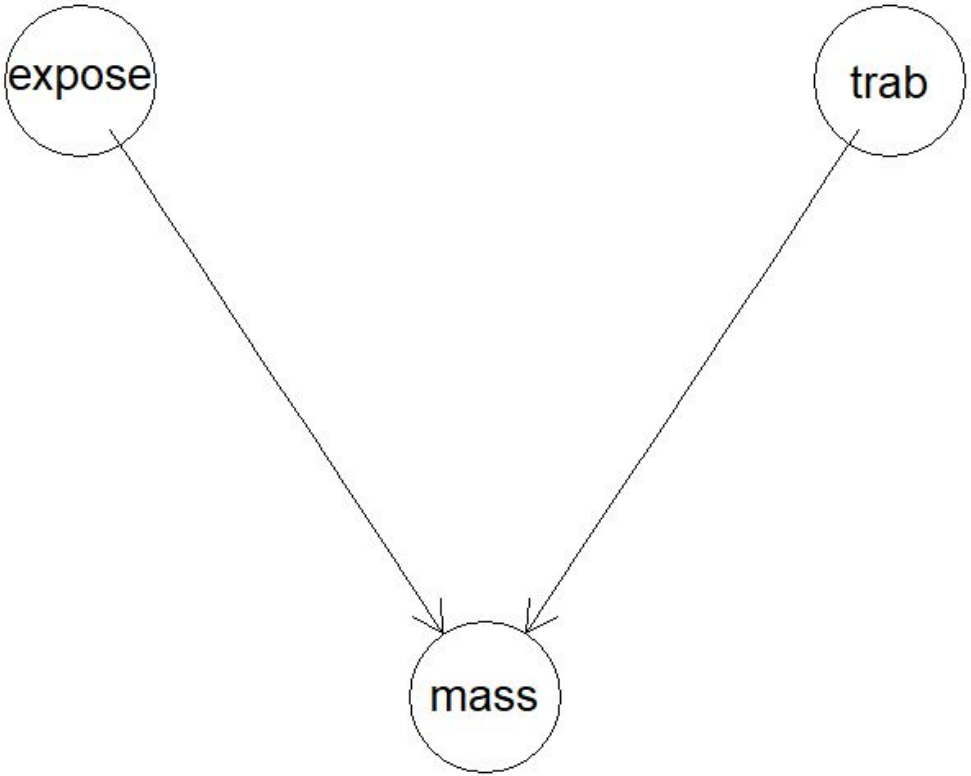
- Create list of all triplets (**a**, **b**, **c**) such that **a** and **c** have a common neighbor **b** but are not directly connected to each other.
- For each triplet (**a**, **b**, **c**):
  - Determine if if there exists a direct causal relationship between **a** and **c**, mediated by **b**.
    - If **a** and **c** are dependent through conditional independence tests
      - Orient edges as **a** → **b** ← **c**
    - If **a** and **c** are independent
      - Orient edges as **a** ← **b** → **c**
  - End loop when all triplets are processed

How my algorithm forms causal conclusions

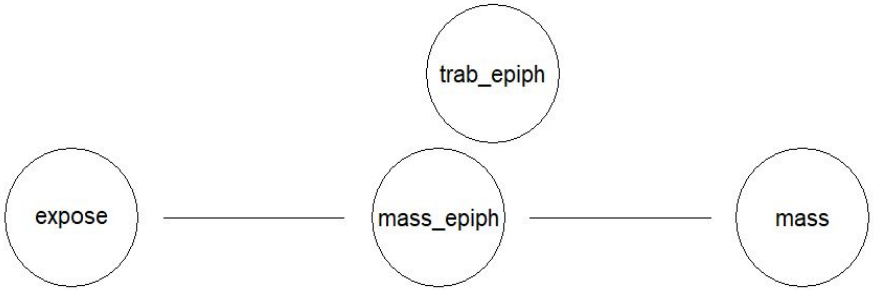
Shortcoming: conflicting edges?

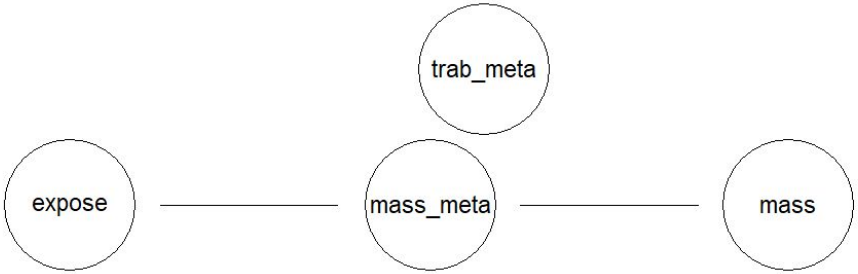
- What if we obtain two triplets **a** → **b** ← **c** and **b** → **c** ← **d**.
  - Of course this means that at least one of the statistical tests is wrong
  - While in some implementations the conflicting edge is overwritten, I believe this to be the reason why some edges are left undirected after running bnlearn's pc.stable().
- What if we have only one triplet, and the expected output is **a** → **b** → **c**
  - This is not possible within the algorithms design

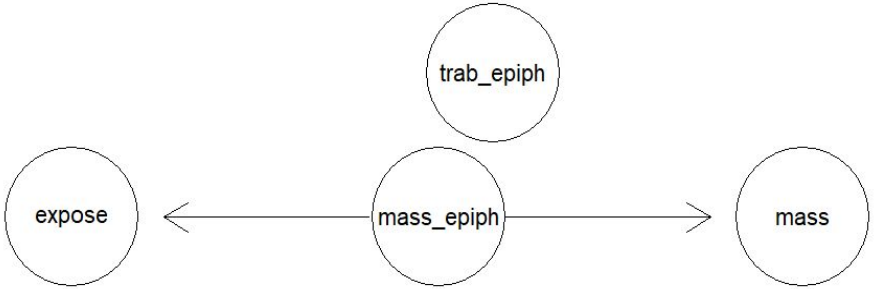
Presentation of results: Dubee / Alwood

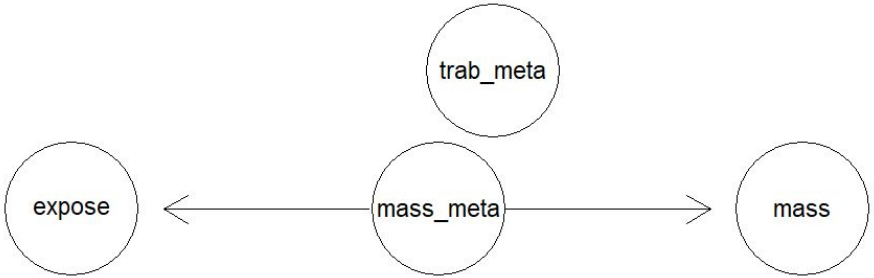


Presentation of results: GDLS / Kuene 2015

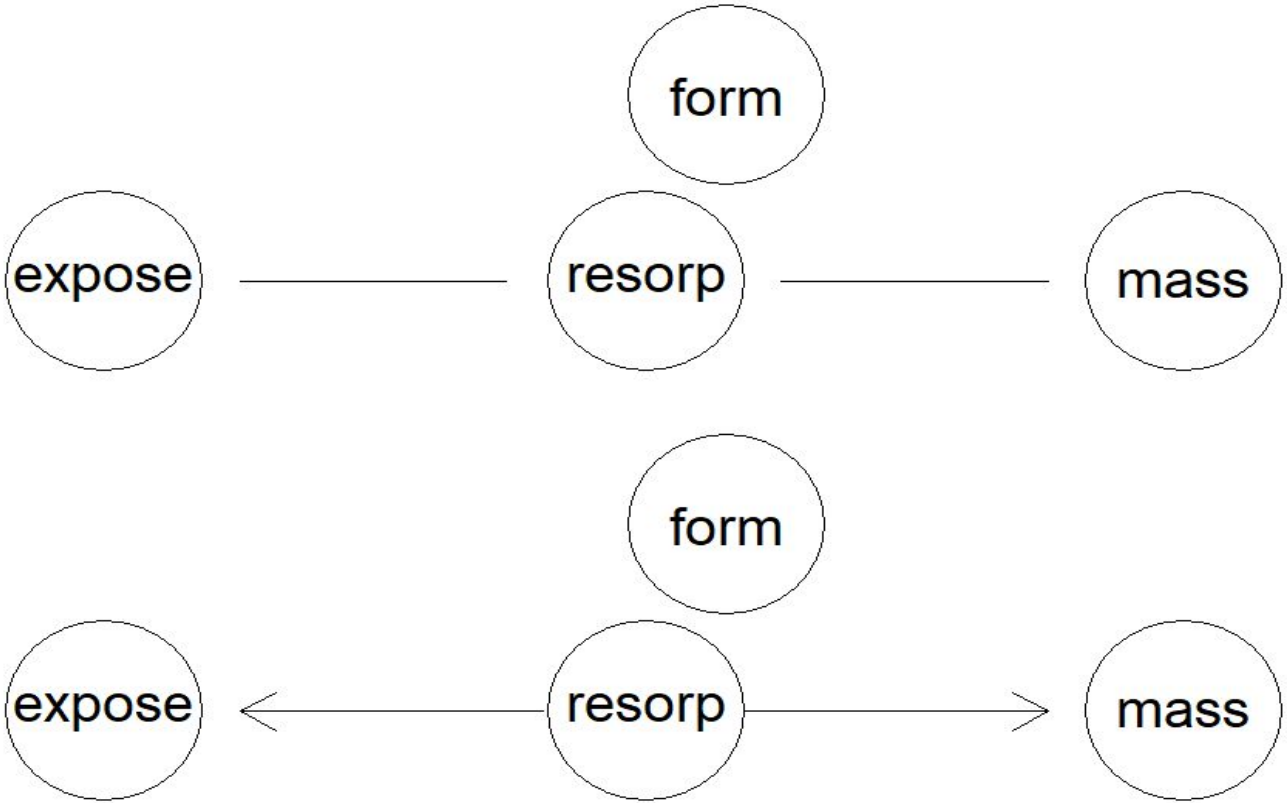








Presentation of results: Keune 2016 / Turner



Presentation of results: Ko