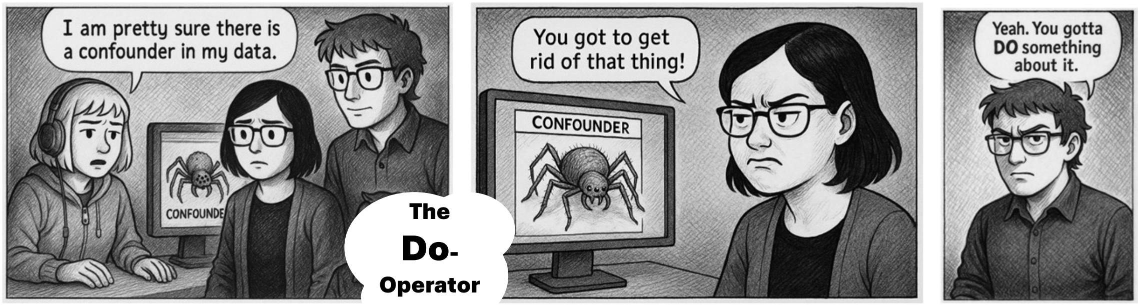


The **Do-Operator (do(X = x))** represents an **intervention**: it forces the variable X to take the value x, **breaking its natural causes**.

The Do-Operator allows to test out counter factuals:

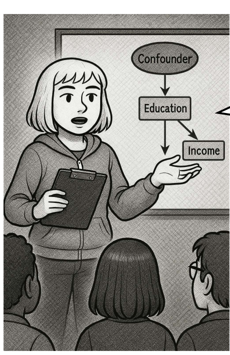
- For this historical figure, what if they had received a classical education instead of a religious one? What if it had been a man instead of a woman?
- For this manuscript, what if it had been printed with a movable type press rather than hand-copied?
- For this region, what if the dominant language in the 13th century had been Cech instead of German? What if **Přemysl Otakar II** had kicked Rudolf of Habsburg's butt?

To understand the do-Operator:

- In simulate_data(), a causal graph with a confounder is simulated.
- In estimate_adjusted_model(data) we control the confounder by regression.
- In predict_do_intervention() we use the regressed values, to produce input-data (as if the confounder was fixed). We are now able to predict the effect of education on income, as if there was no confounder.

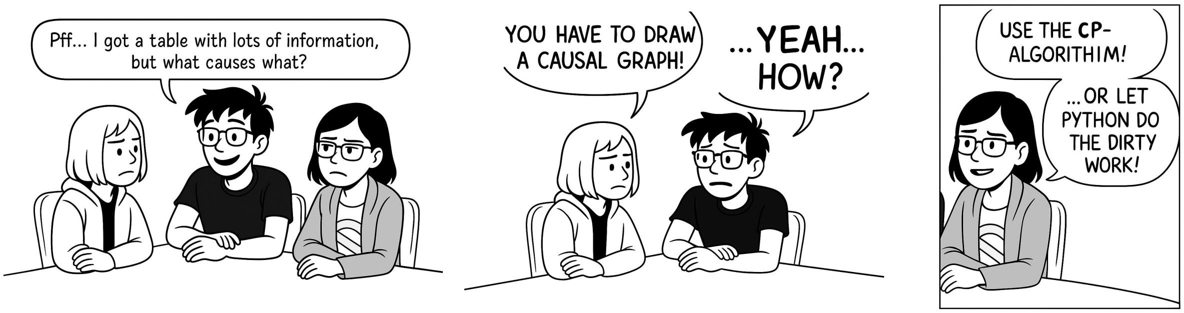
Do-Operator notation: $P(a_i|do(B = b_i), \beta, \gamma, c_i)$

$b_i = controlled\ variable$
$\beta, \gamma = other\ variables$
$c_i = treatment\ variable$



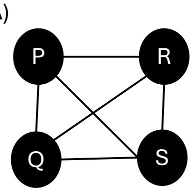
- The do-Operator can simulate a world, where a confounder has no influence on the data.
- We can also simulate and then compare two different worlds: One, where something happens – and one where it does not (ATE).
- We can simulate two versions of an individual (ITE).
- We can average impact of a treatment on those that are treated (ATT)

- Run doOperator.py.
- Explain the visualization.
- In your own words: What is the do-Operator *doing*?

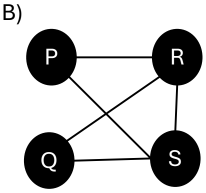


- File causalLearn.py simulates a collider graph with Nodes Q,P,R,S and stores information in a pandas data frame (a table).
- The PC-algorithm is applied using significance test, to find depended nodes
- It calculates the resulting graph.

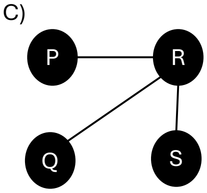
A)



B)

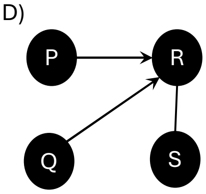


C)

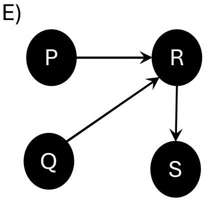


1. **Start with a fully connected undirected graph** (between all observed variables)
2. **Remove edges for unconditional independence** (i.e., if two variables are marginally independent)
3. **Remove more edges using conditional independence**
   1. For each pair A–B, check all subsets C of adjacent variables
   2. If $A \perp\!\!\!\perp B \mid C$, remove the edge between A and B
4. **Orient v-structures** (colliders):
   1. If A–B–C, and A and C are not connected, and A is **not** independent of C given B → then orient as A → B ← C
5. **Propagate orientation** using **Meek's rules**
   1. For example: avoid cycles, prevent new colliders, etc.

D)



E)



- Run causalLearn.py.
- Identify nodes of the "real" graph in the resulting graph.
- Imagine running PC-algorithm on real world data. What could be the benefits?