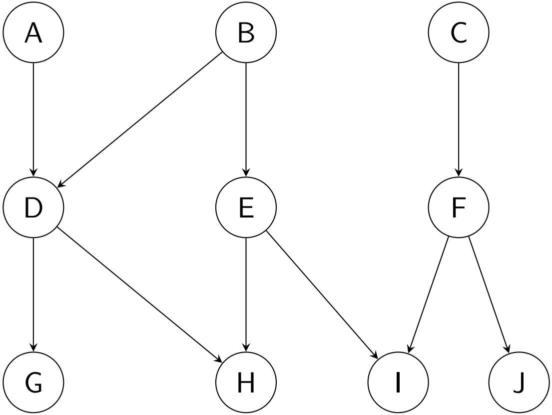
Causal Discovery for Earth System Sciences

Section 2 – Graphical Models

Emiliano Díaz Salas Porras

Image and Signal Processing @ Universitat de València

Example DAG: Structure and Terminology



Notation:

- Children of A, Parents of I, ancestors of H?
- Paths between D and I, directed path between B and H??
- V-structues? Skeleton of DAG?

Independence Maps and d-Separation

Goal: Use a DAG to represent conditional independence (CI) structure of a distribution P.

d-Separation: $X \perp Y \mid Z$.

- A path between X and Y is **blocked** (info. flow) by a set Z if:
  - It contains a chain or fork: $X \to M \to Y$ or $X \leftarrow M \to Y$ with $M \in Z$
  - It contains a collider: $X \to M \leftarrow Y$ and $M \notin Z$ and no descendant of M is in Z
- $X \perp Y \mid Z$ if all paths between X and Y are blocked by Z

Key Idea: We want to relate d-separation in the DAG to conditional independence in the joint probability distribution (in the data). What do we need to ask of the DAG to do this?

Independence Map (I-map):

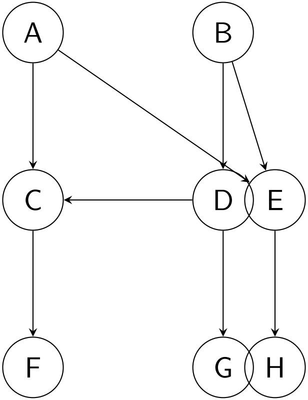
- A DAG $\mathcal{G}$ is an **I-map** of distribution P if:

$$X \perp Y \mid Z \quad \Rightarrow \quad X \perp\!\!\!\perp Y \mid Z \text{ in } P$$

Exercise: d-Separation on a DAG

Given the DAG below, answer the following:

1. What are the non-descendants of node D?
2. Which variables are d-separated from A given $\{B, E\}$?
3. List all variable sets that d-connect A and G



Markov Property of a DAG

Let $\mathcal{G}$ be a DAG over random variables $X_1, \ldots, X_n$ and P a joint probability distribution (with a density!) over them.We say that P is **Markov with respect to** $\mathcal{G}$ if it satisfies any of the following equivalent conditions:

1. **Global Markov Property** ($\mathcal{G}$ an I-map of P, $I(P) \subseteq I(\mathbf{G})$)
   - For any disjoint sets X, Y, and Z:

$$X \perp_d Y \mid Z \text{ in } \mathcal{G} \quad \Rightarrow \quad X \perp\!\!\!\perp Y \mid Z \text{ in } P$$

2. **Local Markov Property**
   - Each variable is conditionally independent of its non-descendants given its parents:

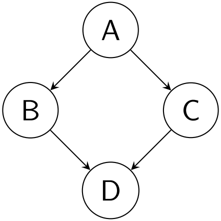
$$X_i \perp\!\!\!\perp \mathrm{ND}(X_i) \mid \mathrm{Pa}(X_i)$$

3. **Markov Factorization Property**
   - The joint distribution factors according to the DAG:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \mathrm{Pa}(X_i))$$

Example: DAG Satisfying the Markov Properties

DAG:



1. **Global Markov Property:**
   - $B \perp C \mid A \Rightarrow B \perp\!\!\!\perp C \mid A$ in P
   - $B \perp C$ unconditionally is false (connected through A)
2. **Local Markov Property:**
   - $B \perp\!\!\!\perp \{C, D\} \mid A$
   - $C \perp\!\!\!\perp \{B, D\} \mid A$
   - $D \perp\!\!\!\perp \{A\} \mid \{B, C\}$
3. **Factorization:**

Why is markov not enough for learning graph? What joint distributions is a fully conneced DAG an I-map for?

Minimality

Definition (Minimality): P satisfies the **causal minimality condition** with respect to $\mathcal{G}$ if:

- $\mathcal{G}$ is an I-map of P (i.e., $I(P) \subseteq I(\mathcal{G})$), and
- There is no strict subgraph $\mathcal{G}' \subset \mathcal{G}$ such that $I(P) \subseteq I(\mathcal{G}')$

Intuition: Every edge in $\mathcal{G}$ is necessary to maintain right independencies in P. No edge is superfluous.

Constructing a Minimal I-map (Procedure):

1. Take a topological ordering of the variables.
2. Write the Markov factorization:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \text{Parents}(X_i))$$

3. Simplify each term using known conditional independencies.
   - Drop variables from the conditioning set if $X_i \perp\!\!\!\perp V \mid \text{Rest}$.

Exercise: Constructing Minimal I-maps

Given: Three random variables X_1, X_2, X_3 **Assumption:** The only conditional independence in the distribution P is:

$$X_1 \perp\!\!\!\perp X_3 \mid X_2$$

Task: Construct all possible **minimal I-maps** (DAGs) that are:

- Markov with respect to P
- Minimal (i.e., have no removable edges without violating the Markov condition)

Hint: Use the construction procedure:

1. Choose a node ordering
2. Write the full factorization
3. Use the given conditional independence to remove unnecessary edges

Minimal causal maps no unique? Why?

Faithfulness

Faithfulness (Definition): A distribution P is **faithful** to a DAG $\mathcal{G}$ if:

$$X \perp\!\!\!\perp Y \mid Z \text{ in } P \quad \Rightarrow \quad X \perp_d Y \mid Z \text{ in } \mathcal{G}$$

Interpretation:

- No conditional independence in P arises accidentally — all are reflected in the DAG structure.
- All d-connected pairs in $\mathcal{G}$ are dependent in P
- $I(\mathcal{G}) \subseteq I(P)$

Perfect I-map (Definition): An Graph is a perfect I-map of P if P is markov and faithful wrt $\mathcal{G}$
or Equivalently:

$$I(P) = I(\mathcal{G})$$

Stronger than causal minimality!

Exercise: find aminimal causal map that is not faithful from previous exercise

A Bayesian Network (BN) is a pair $(\mathcal{G}, P)$ where:

- $\mathcal{G}$ is a **DAG** over variables $X_1, \ldots, X_n$
- P is a joint distribution that satisfies the **Markov property** with respect to $\mathcal{G}$

Key Properties:

- $\mathcal{G}$ is an **I-map** of P
- P factorizes as:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \mathrm{Pa}(X_i))$$

- The independencies implied by $\mathcal{G}$ via d-separation are valid in P

BNs are useful for:

- Efficient representation of high-dimensional distributions (sparsity)
- Reading conditional independencies directly from the graph (global independencies)

Why Are Bayesian Networks Useful?

1. Efficient Probabilistic Inference

- Factorization reduces the complexity of computing marginals and conditionals.
- Enables efficient algorithms:
  - Variable elimination
  - Belief propagation
  - Ancestral sampling / MCMC
- Supports reasoning under uncertainty: $P(Y \mid X = x)$

2. Compact Representation of Joint Distributions

- A full joint distribution grows exponentially with number of variables
- BNs capture structure via sparse graphs:

$$P(X_1, \ldots, X_n) = \prod_i P(X_i \mid \mathrm{Pa}(X_i))$$

3. Causality

Exercise: Parameters in Bayesian Networks

Assume:

- $X_1, \dots, X_7$ are Bernoulli random variables
- For each DAG, compute the number of parameters required to specify the full joint distribution

Graph structures:

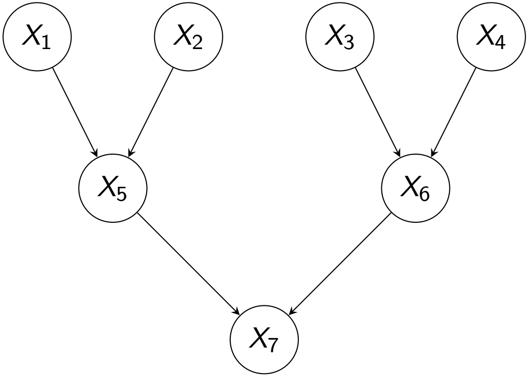
- **(A) Empty DAG:** No edges
- **(B) Chain DAG:** $X_1 \to X_2 \to \cdots \to X_7$
- **(C) Layered DAG:**

$$\{X_1 \to X_5,\ X_2 \to X_5,\ X_3 \to X_6,\ X_4 \to X_6,\ X_5 \to X_7,\ X_6 \to X_7\}$$

- **(D) Fully Connected DAG:** Each X_i has all earlier X_j as parents

Question: How many parameters are needed in each case?

Layered DAG Example: 7 Nodes



Solution: Parameters in Bayesian Networks

Solution: Parameters in Bayesian Networks

(A) Empty DAG:

- Each X_i is independent $\rightarrow$ 1 parameter each
- Total: $7 \times 1 = \boxed{7}$

(B) Chain DAG:

- X_1: 1 param
- X_2–X_7: 6 nodes with 1 parent $\rightarrow 6 \times 2$ params
- Total: $1 + 6 \times 2 = \boxed{13}$

(C) Layered DAG:

- X_1–X_4: 1 param each (no parents)
- X_5, X_6: each has 2 parents $\rightarrow 2^2 = 4$ params
- X_7: 2 parents $(X_5, X_6) \rightarrow 4$ params
- Total: $4 \times 1 + 2 \times 4 + 4 = \boxed{20}$

(D) Fully Connected DAG:

- Total: $\sum_{i=1}^{7} 2^{i-1} = 2^7 - 1 = \boxed{127}$ parameters

Key Question: Can different DAGs represent the same set of conditional independencies?

Yes. Two DAGs $\mathcal{G}_1$ and $\mathcal{G}_2$ are **Markov equivalent** if:

$$I(\mathcal{G}_1) = I(\mathcal{G}_2)$$

Markov Equivalence Class (MEC):

- The set of all DAGs that encode the same CI structure.
- Denoted as the collection of DAGs with:
  - Same skeleton (i.e., same undirected edges)
  - Same set of v-structures ($X \to Z \leftarrow Y$, with no edge between X and Y)

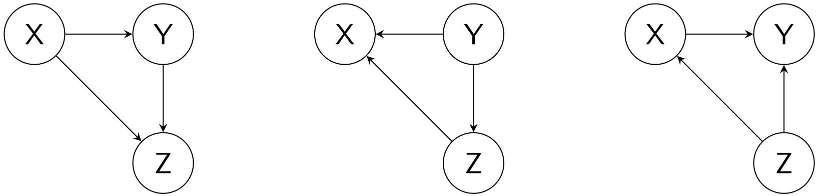
CPDAG: A **Completed Partially Directed Acyclic Graph** represents a MEC:

- Directed edges: common to all DAGs in the MEC
- Undirected edges: direction varies between DAGs in the class

Example: Markov Equivalent DAGs and their CPDAG

Three DAGs with same skeleton and no v-structures:

DAG 1 **DAG 2** **DAG 3**



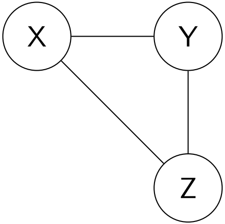
All have:

- Same skeleton: edges $\{X - Y, Y - Z, X - Z\}$
- No v-structures (no collider of form $X \to Z \leftarrow Y$)
- Same set of d-separation relations $\Rightarrow$ same independencies

CPDAG Representing the Equivalence Class

CPDAG: Represents all DAGs in the equivalence class — edges that are **undirected** can vary in orientation, edges that are **directed** are common to all DAGs in the class.

CPDAG for previous DAGs:

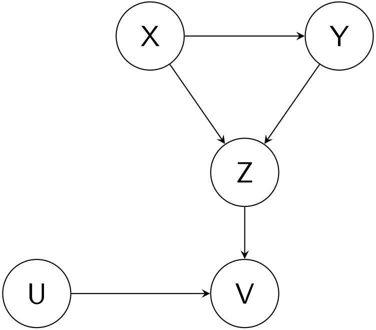


Interpretation:

- No directed edges $\Rightarrow$ no v-structures in the equivalence class
- All three DAGs are valid orientations of this CPDAG

Exercise: Find the CPDAG for the Given DAG

Given DAG:



Task: Draw the CPDAG representing the Markov Equivalence Class of this DAG.

Solution: CPDAG for the Given DAG

Step 1 – Skeleton: Undirected edges between:

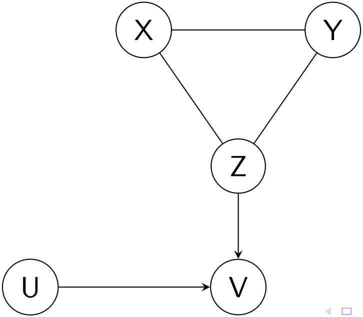
$$X - Y, \quad X - Z, \quad Y - Z, \quad Z - V, \quad U - V$$

Step 2 – Identify v-structures (immoralities):

$$X \to Z \leftarrow Y \quad \text{(not a v-structure: X and Y are connected)}$$

Z $\to V \leftarrow U$ **is a v-structure (immorality)**

CPDAG:



Why Naïve Structure Learning Fails

Naïve idea: Learn the full set of conditional independence relations $I(P)$ from data. Then search for a DAG $\mathcal{G}$ (for example with minimal-map search prodcedure) such that:

$$I(\mathcal{G}) = I(P)$$

Problem:

- **Not every set of independencies is representable** by a DAG.
- A DAG must be **faithful** to P:

$$I(P) \subseteq I(\mathcal{G}) \quad \text{and} \quad I(\mathcal{G}) \subseteq I(P)$$

- But some independencies arise from **accidental cancellation** or functional symmetries, not from d-separation.

Why Naïve Structure Learning Fails

Conclusion: Without additional assumptions, there may be no DAG $\mathcal{G}$ such that $I(\mathcal{G}) = I(P)$. Additionally, it is too expensive to obtain exhaustive list of independencies. We need to exploit markov equivalences to reduce number of inependence tests.

We need:

- The **faithfulness assumption**, and
- A structured approach to search the space of DAGs

Non-Adjacency Result: towards PC algorithm

Assume:

- The true distribution P is **faithful** to some DAG $\mathcal{G}$
- We have access to an oracle (or algorithm) for testing CI

Non-Adjacency Theorem: In a DAG $\mathcal{G}$, two variables X and Y are **not adjacent** (i.e., no edge between them) if and only if there exists a set $Z \subseteq \text{Pa}(X) \cup \text{Pa}(Y)$ such that:

$$X \perp\!\!\!\perp Y \mid Z$$

Implications:

- We can detect absence of edges via conditional independence tests
- If we assume faithfulness, d-separation and CI align:

$$X \perp_d Y \mid Z \quad \Leftrightarrow \quad X \perp\!\!\!\perp Y \mid Z$$

- This allows us to reconstruct the skeleton (undirected graph) of $\mathcal{G}$. **This result forms the backbone of the PC algorithm.**

The PC Algorithm: Pseudocode

Step 1: Initialize Fully Connected Undirected Graph

- Create complete undirected graph over all variables

Step 2: Remove Edges Based on Conditional Independence

- For each pair (X, Y), test for independence conditioned on subsets of adjacent nodes
- Start with conditioning sets of size 0, increase size iteratively
- If $X \perp\!\!\!\perp Y \mid Z$, remove edge $X - Y$
- Record separating set $\text{Sep}(X, Y) = Z$

Step 3: Orient Edges to Identify V-structures

- For all triples $X - Z - Y$ with X and Y non-adjacent:

$$\text{If } Z \notin \text{Sep}(X, Y)\text{, orient as } X \rightarrow Z \leftarrow Y$$

Step 4: Apply Orientation Rules

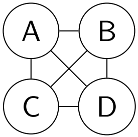
- Use logical rules to propagate orientations while avoiding cycles
- Example: If $X \rightarrow Y$ and $Y - Z$, then orient $Y \rightarrow Z$ (Meek's rules)

PC Algorithm Example – Step 1: Setup

Ground truth DAG (not known to learner):
Assume: The CI oracle returns:

- $A \not\perp B$, $A \not\perp C$, $B \not\perp D$, $C \not\perp D$
- $A \perp D \mid \{B, C\}$
- $B \perp C$

Step 1: Initialize a fully connected undirected graph over A, B, C, D



PC Algorithm Example – Step 2: Remove Edges

Apply conditional independence tests:

- $B \perp C \Rightarrow$ remove edge $B - C$
- $A \perp D \mid \{B, C\} \Rightarrow$ remove edge $A - D$

Remaining skeleton:

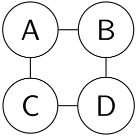
$$A - B, \quad A - C, \quad B - D, \quad C - D$$

Separating sets:

- $\text{Sep}(B, C) = \emptyset$
- $\text{Sep}(A, D) = \{B, C\}$

PC Algorithm Example – Step 3: Orient V-Structures

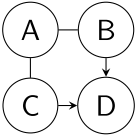
Skeleton after edge removal:



V-Structure Rule: If $X - Z - Y$ and X and Y are not adjacent, and $Z \notin \text{Sep}(X, Y)$, then orient as: $X \rightarrow Z \leftarrow Y$

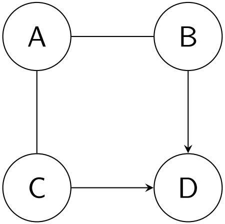
Apply to: $B - D - C$, and $B \nsim C$, and $D \notin \text{Sep}(B, C) = \emptyset \Rightarrow$ **Orient:** $B \rightarrow D \leftarrow C$

Current partially directed graph:



PC Algorithm Example – Step 4: Apply Orientation Rules

Current partial graph:



Apply orientation rules (Meek's rules):

- Rule 1 (No new v-structures): If $A \rightarrow B$, $B - C$, and A and C not connected, then orient $B \rightarrow C$
- Rule 2 (Avoid cycles): Orient any edge that would otherwise create a directed cycle

In this example: No further orientations are possible without introducing cycles or v-structures.

Python notebook with SeasFire + pybnesian

Structure Learning: Motivation and Overview

Goal: Learn a DAG $\mathcal{G}$ from data such that:

$$I(P) \approx I(\mathcal{G})$$

General Approach:

1. Suppose we have an independence oracle.
2. Find all local independencies (PC algoirthm) and trust that the dag(s, cpdag) that this defines is faithful - there are no extra ones .
3. Decide between the dags in MEC (need another criterion!)

What could go wrong?

Causal Sufficiency and Hidden Variables

Causal Sufficiency Assumption: All common causes of the observed variables are themselves observed.

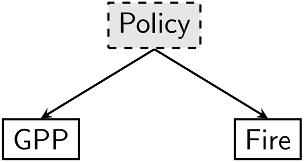
What if this assumption fails?

- Unobserved (latent) variables can act as hidden confounders
- Conditional independence tests may no longer reflect d-separation in any DAG over the observed variables
- DAG-based methods (e.g., PC) can fail or produce incorrect conclusions

Two solutions:

1. Use the **projected graph** over observed variables:
   - May include bidirected edges to represent unobserved common causes
2. Use a **Maximal Ancestral Graph (MAG):**
   - Graph that includes directed and bidirected edges
   - Represents conditional independencies among observed variables, even when latent confounders exist

Spurious GPP–Fire Association from Unobserved Policy



- Gross Primary Productivity (GPP) and fire occurrence are both influenced by an unobserved land management policy.
- Only GPP and Fire are observed; ignoring Policy leads to a spurious GPP–Fire association.
- Violates causal sufficiency: not all common causes are observed, so the GPP–Fire correlation can be misinterpreted as a direct causal link.