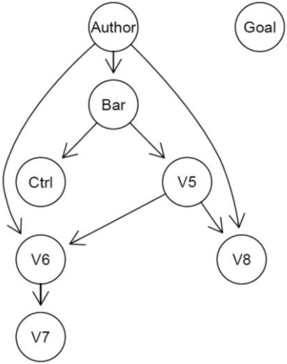
# Discovering Causal Structure with the PC-algorithm

## CRG and MSDSlab Meeting

Discussant: Oisín Ryan

February 23, 2018

# What is a DAG?




- DAG = **D**irected **A**cyclic **G**raph
- **Nodes** or **Vertices** = {Author, Bar, V5.. }
- Directed **Edges** $\rightarrow$
- No cycles
  - Cannot have $A \rightarrow B \rightarrow C \rightarrow A$

# Why DAGs?

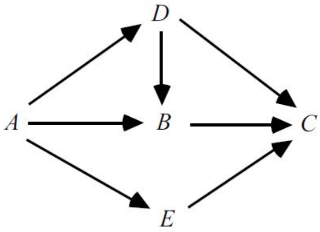
1. DAGs can be used to represent joint probability distributions
   - Often called **Bayesian networks**
   - Nodes represent **variables**
   - Edges represent **dependencies** between pairs of variables
     - $A \to B$ means $A \not\perp\!\!\!\perp B$
   - Read off **conditional dependency** relationships

# Why DAGs?

1. DAGs can be used to represent joint probability distributions
   - Often called **Bayesian networks**
   - Nodes represent **variables**
   - Edges represent **dependencies** between pairs of variables
     - $A \rightarrow B$ means $A \not\perp\!\!\!\perp B$
   - Read off **conditional dependency** relationships

2. DAGs + probability distribution used for **causal inference**
   - Edges represent **direct causal links**
     - $A \rightarrow B$ means $A$ causes $B$
   - **Counterfactual** causality (Pearl, Rubin, Spirtes & Glymour)
   - Account for typical ideas about causality
     - Forward in time - acyclical
     - Explains "paradoxes" - Simpsons, Lords

# Some graph terminology




- **Parents**(B) $= \{A, D\}$
- **Children**(B) $= \{C\}$
- A is an **ancestor** of C
- C is a **descendant** of A
- **Path** $=$ sequence of edges

# DAGs and Probability distributions

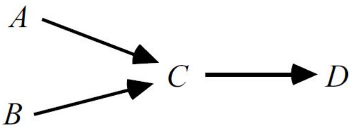
DAGs can be used to represent a joint density function using the **Markov Condition**

$$P(\boldsymbol{V}) = \prod_{V \in \boldsymbol{V}} P\big(V|\textbf{Parents}(V)\big) \tag{1}$$

# DAGs and Probability distributions

DAGs can be used to represent a joint density function using the **Markov Condition**

$$P(\boldsymbol{V}) = \prod_{V \in \boldsymbol{V}} P\big(V | \textbf{Parents}(V)\big) \tag{1}$$

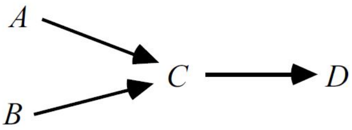



$P(A, B, C, D) =$
$P(A)P(B)P(C|A, B)P(D|C)$

# DAGs and Probability distributions

DAGs can be used to represent a joint density function using the **Markov Condition**

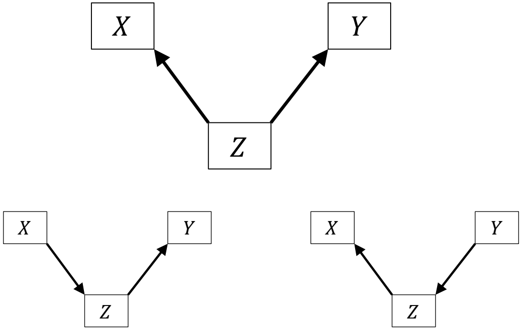
$$P(\boldsymbol{V}) = \prod_{V \in \boldsymbol{V}} P\big(V | \textbf{Parents}(V)\big) \tag{1}$$




$P(A, B, C, D) =$
$P(A)P(B)P(C|A, B)P(D|C)$

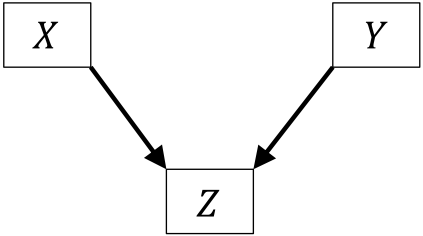
We can also read off **conditional (in)dependence** relationships not directly implied by the Markov Condition

# DAGs and conditional dependencies



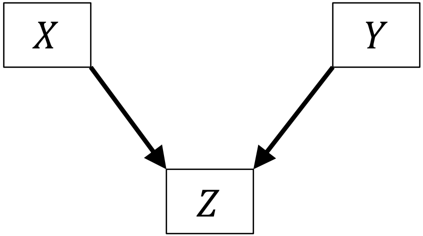

$$X \not\perp\!\!\!\perp Y$$
$$X \perp\!\!\!\perp Y \mid Z$$

# DAGs and conditional dependencies




$$X \perp\!\!\!\perp Y$$
$$X \not\perp\!\!\!\perp Y \mid Z$$

# DAGs and conditional dependencies




$$X \perp\!\!\!\perp Y$$
$$X \not\perp\!\!\!\perp Y \mid Z$$

General rules to read off conditional (in)dependencies from DAGs are known as **d-seperation** rules

# What use is having a DAG?

- Representation of causal relations amongst variables
- Estimation of causal effects
- Identify sufficient conditioning set to control for confounding
  - I may not need to condition on **all** possible ancestors
  - I shouldn't condition on colliders

# Discovering DAGs from Data

**PC** algorithm (**P**eter Spirtes & **C**lark Glymour)

Assuming:

- The set of observed variables is sufficent
  - No common causes not present in the dataset
  - Extensions that account for **latent variables** do exist!
- The distribution of the observed variables is faithful to a DAG

# PC algorithm

Two simple principles

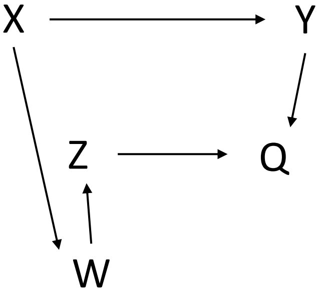
1. There is an edge $X - Y$ if and only if $X$ and $Y$ are dependent conditional on **every possible subset** of the other variables
   - For a graph $G$ with vertex set $\boldsymbol{V}$:
   - $X - Y$ iff $X \not\perp\!\!\!\perp Y | \boldsymbol{S}$, for all $\boldsymbol{S} \subseteq \boldsymbol{V} \setminus \{X, Y\}$

# PC algorithm

Two simple principles

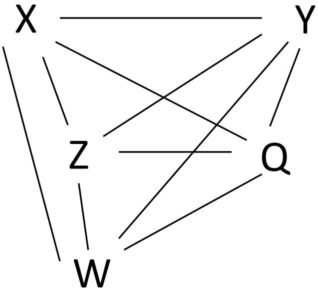
1. There is an edge $X - Y$ if and only if $X$ and $Y$ are dependent conditional on **every possible subset** of the other variables
   - For a graph $G$ with vertex set $\boldsymbol{V}$:
   - $X - Y$ iff $X \not\perp\!\!\!\perp Y | \boldsymbol{S}$, for all $\boldsymbol{S} \subseteq \boldsymbol{V} \setminus \{X, Y\}$
2. If $X - Y - Z$, we can orientate the arrows as $X \rightarrow Y \leftarrow Z$ if and only if $X$ and $Z$ are dependent conditional on every set containing $Y$
   - We only orientate arrows if we can identify a collider

# PC algorithm example




True DAG

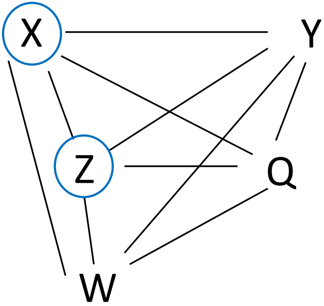
# PC algorithm example




**Step 0**

- Start with a fully connected undirected graph

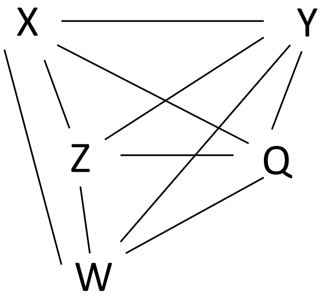
# PC algorithm example




**Step 1a**

- Test marginal dependencies for each pair
- E.g. correlation between X and Z
- If not significant, delete the edge
- repeat for all pairs

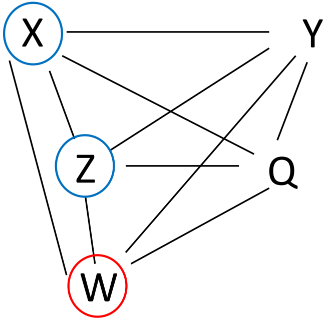
# PC algorithm example




**Step 1a**

- Test marginal dependencies for each pair
- E.g. correlation between X and Z
- If not significant, delete the edge
- repeat for all pairs

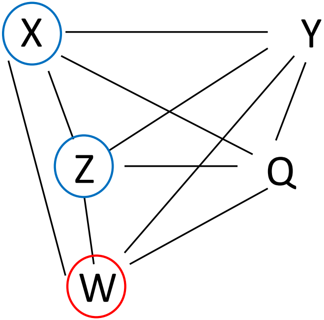
# PC algorithm example




**Step 1b**

- Take the result of step 1a as input
- For each **adjacent pair** in this graph (e.g., $A, B$)
- Form the **adjacency set** of $A$ and $B$: set of variables connected to **either** $A$ or $B$, $\boldsymbol{Adj}(A, B)$
- Test CI of $A$ and $B$ for each size=1 subset of $\boldsymbol{Adj}(A, B)$

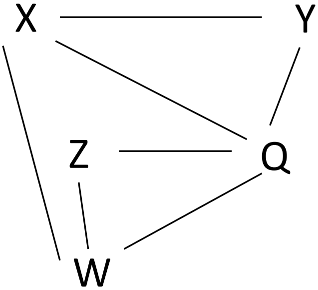
# PC algorithm example




**Step 1b**

- Take the result of step 1a as input
- For each **adjacent pair** in this graph (e.g., $A, B$)
- Form the **adjacency set** of $A$ and $B$: set of variables connected to **either** $A$ or $B$, $\boldsymbol{Adj}(A, B)$
- Test CI of $A$ and $B$ for each size=1 subset of $\boldsymbol{Adj}(A, B)$
- Record the variables that seperate $A$ from $B$

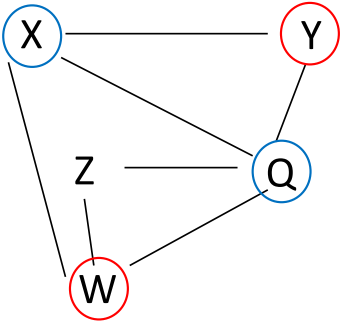
# PC algorithm example




**Step 1c**

- Take the result of step 1b as input
- Repeat previous step but for subset of size =2

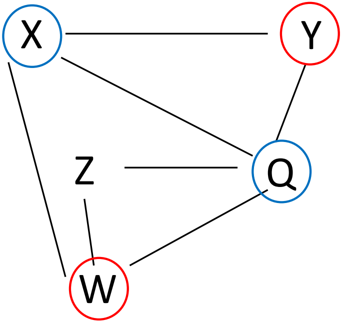
# PC algorithm example




**Step 1c**

- Take the result of step 1b as input
- Repeat previous step but for subset of size $=2$

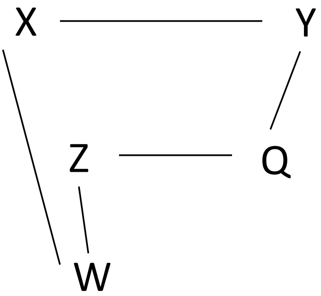
# PC algorithm example




**Step 1c**

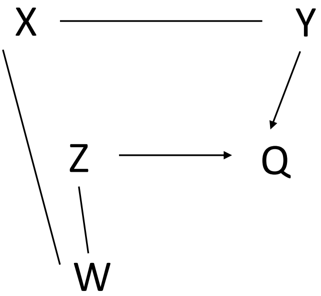
- Take the result of step 1b as input
- Repeat previous step but for subset of size =2
- In general, keep increasing the number of variables you condition on until this is larger than the size of $\boldsymbol{Adj}(A, B)$

# PC algorithm example




- This is an estimate of the **skeleton** of the DAG
- An estimate of the undirected version of the DAG

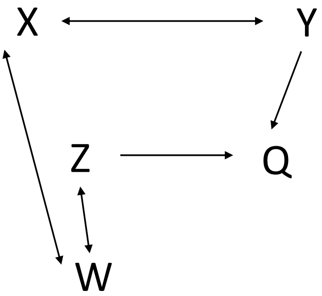
# PC algorithm example




**Step 2**

- For each triplet (open triangle) $A - B - C$
- Orient $A \rightarrow B \leftarrow C$ if we did not condition on $B$ to seperate $A$ and $C$
- We now have a Complete Partially-Oriented DAG (CPDAG)

# PC algorithm example




**Step 2**

- For each triplet (open triangle) $A - B - C$
- Orient $A \rightarrow B \leftarrow C$ if we did not condition on $B$ to seperate $A$ and $C$
- We now have a Complete Partially-Oriented DAG (CPDAG)

Let's test the PC algorithm out