Swiss Federal Institute of Technology Zurich

Department of Mathematics

Seminar for
Statistics

Master Thesis Summer 2018

Nicola Gnecco

Causality in Heavy-Tailed Data

Submission Date: August 3rd, 2018

Adviser: Prof. Dr. Nicolai Meinshausen

To my Family and to Ninna

Preface

I would like to thank Professor Meinshausen, for guiding me during this thesis. He dedicated a lot of his time to help me. He always gave me the right insights to proceed and gently corrected me when I was going off the path. It has been a privilege to have him as a supervisor.

I am grateful to Professor Engelke, for the several insightful discussions and his constant and constructive feedbacks.

Abstract

We introduce a novel method to estimate a causal order from heavy-tailed data. We start with a binary coefficient Γ, to detect causal directions between heavy-tailed variables. This coefficient, proposed by Engelke, Meinshausen, and Peters (2018), can also detect the presence of a hidden confounder. We investigate the population properties of the Γ coefficient in a linear SEM, with an arbitrary number of variables. Besides, we prove that it is possible to identify the source node of a linear SEM, under some assumptions. Based on this result, we build four competing algorithms to recover the causal order of a graph from observational data. We compare and test the sample properties of the algorithms on simulated data. We show that our algorithms perform equally well in the large sample limit when the heavy-tail assumption is fulfilled. Finally, we test the algorithms in the presence of hidden confounders.

Contents

1 Introduction 1
1.1 Problem and Motivation . . . 1
1.2 Outline . . . 2
1.3 Contributions . . . 2

2 Causality 3
2.1 Graphs . . . 3
2.2 Bayesian Networks . . . 8
2.3 Causal Bayesian Networks . . . 10
2.3.1 Causal Bayesian Network and *do-calculus* . . . 11
2.3.2 Structural Equation Models (SEM) . . . 12
2.4 Causal Structure Learning . . . 14
2.4.1 Class 1: Learning the Markov Equivalence Class . . . 14
2.4.2 Class 2: Leveraging the non-Gaussian noise . . . 16

3 Heavy-Tailed Distributions 19
3.1 Intuition . . . 19
3.2 Scale invariance and power-law . . . 21
3.3 Approximately scale invariance and regularly varying distributions . . . 22
3.4 Formalizing the catastrophe principle . . . 24

4 Method 27
4.1 Intuition . . . 27
4.2 Properties of Γ_{ij} . . . 27
4.2.1 Setup . . . 27
4.2.2 Results . . . 28
4.3 Examples . . . 31
4.4 Structure Learning . . . 35

5 Simulations 41
5.1 Learning a Topological Order . . . 41
5.1.1 Q-performance . . . 41
5.1.2 Topological matrix . . . 41
5.1.3 Simulation 1 - Simple chain . . . 42
5.1.4 Simulation 2 - Polytree . . . 43
5.1.5 Simulation 3 - Non-Polytree . . . 44
5.1.6 Simulation 4 - Graph with confounders . . . 44
5.2 Some comments . . . 45

6 Conclusions 59

Appendix 59

A Proofs 61
A.1 Proof of Proposition 2.1.0.9 . . . 61
A.2 Proof of Proposition 2.3.2.2 . . . 61
A.3 Proof of Proposition 3.2.0.3 . . . 61
A.4 Proof of Proposition 3.3.0.3 . . . 62

A.5 Proof of Proposition 3.3.0.5 63
A.6 Proof of Lemma 3.4.0.1 63
A.7 Proof of Lemma 3.4.0.2 64

B Probability Theory **65**

List of Figures

2.2 Parents of B (red) $-$ Children of B (green) 5
2.11 Left: Joint distribution $P(X_1, X_2)$ generated by the SEM 2.4.2.1. The linear regression of X_2 on X_1 (respectively of X_1 on X_2) is shown as a blue (resp. red) line. Middle: Residuals $X_2 - f(X_1)$ are independent of X_1. Right: Residuals $X_1 - g(X_2)$ depend on X_2. 18

3.1 Exponential tail (red) and Pareto tail (blue) 20

4.1 X_1 causes X_2 28
4.2 Causal Bayesian Network with a confounder X_C 37

5.1 Causal Bayesian Network - Simple chain 42
5.2 Causal Bayesian Network - Polytree 43
5.3 Causal Bayesian Network - Non-Polytree 44
5.4 Causal Bayesian Network - Graph with confounders 44
5.5 **Simple chain**. Topological matrices for $n = 1000$, $\alpha = 1$, $b = 2$ 48
5.6 **Polytree**. Topological matrices for $n = 10000$, $\alpha = 5$, $b = 2$ 51
5.7 **Non-Polytree**. Topological matrices for $n = 1000$, $\alpha = 1$, $b = 3$ 54
5.8 **Graph with confounders**. Topological matrices for $n = 10000$, $\alpha = 5$, $b = 1$ 57

List of Tables

5.1 **Simple chain**. Q-performance for ***Drop*** algorithm 46
5.2 **Simple chain**. Q-performance for ***Regress*** algorithm 46
5.3 **Simple chain**. Q-performance for ***Remove*** algorithm 47
5.4 **Simple chain**. Q-performance for ***Fast*** algorithm 47
5.5 **Polytree**. Q-performance for ***Drop*** algorithm 49
5.6 **Polytree**. Q-performance for ***Regress*** algorithm 49
5.7 **Polytree**. Q-performance for ***Remove*** algorithm 50
5.8 **Polytree**. Q-performance for ***Fast*** algorithm 50
5.9 **Non-Polytree**. Q-performance for ***Drop*** algorithm 52
5.10 **Non-Polytree**. Q-performance for ***Regress*** algorithm 52
5.11 **Non-Polytree**. Q-performance for ***Remove*** algorithm 53
5.12 **Non-Polytree**. Q-performance for ***Fast*** algorithm 53
5.13 **Graph with confounders**. Q-performance for ***Drop*** algorithm 55
5.14 **Graph with confounders**. Q-performance for ***Regress*** algorithm 55
5.15 **Graph with confounders**. Q-performance for ***Remove*** algorithm 56
5.16 **Graph with confounders**. Q-performance for ***Fast*** algorithm 56

Chapter 1

Introduction

1.1 Problem and Motivation

Consider a random vector $X = (X_1, \ldots, X_p)$ with a heavy-tailed distribution. Suppose you observe some *iid* copies of X and you want to reconstruct the causal relations between the X_i, $i = 1, \ldots, p$. We will show a method to estimate a causal order for the variables by exploiting their heavy-tailed distribution. Inferring causal relations is one of the primary goals of the scientific method. Randomized experiments have always been the gold standard when it comes to detecting causal effects between variables. In the last decades, Judea Pearl has introduced a framework to establish when it is possible to infer causality from observational data, i.e., without doing any experiment. This aspect has many practical advantages whenever it is not possible to run randomized controlled trials (for ethical of practical reasons). Some domains which deal with these issues are medicine and genetics for example. Potential areas of application of causality combined with heavy-tailed data are, for example, financial markets or environmental catastrophes.

Establishing causal relations from simple observations is hard. There are situations in which two different causal structures can generate the same observational data, for example in the case of Gaussian noise in a linear Structural Equation Model (SEM). In this setup, it is impossible to recover the causal structure just by observing the data. Many approaches to solving this problem aim to estimate a class of plausible causal structures (called Markov Equivalence class) rather than zeroing in on a unique causal order. Spirtes, Glymour, Scheines, Heckerman, Meek, Cooper, and Richardson (2000) proposed the first algorithms in this direction.

A different approach to the problem is to assume that the noise of the SEM in non-Gaussian and to exploit this "asymmetry" to detect causal relations in the data. Among these methods, we recall LiNGAM-algorithm of Shimizu, Hoyer, Hyvärinen, and Kerminen (2006). Based on the ideas of Independent Component Analysis (ICA) (Comon, Jutten, and Herault, 1991), the algorithm manages to estimate the causal ordering and the causal effects between the variables by assuming non-Gaussian noise in the SEM.

This thesis introduces a method tailored on heavy-tailed data, to estimate the causal ordering of a set of variables. We start with the idea of Engelke et al. (2018), who propose a binary coefficient Γ to detect the causal direction for pairs of variables. Under certain conditions, the Γ coefficient can also detect the presence of a hidden confounder, by exploiting the heavy-tail assumption. We build on this idea, and we prove some properties of the Γ coefficient in a linear SEM with an arbitrary number of variables. We also prove that, under certain assumptions, it is possible to identify the exogenous variable of a linear SEM. Based on these results, we build four competing algorithms to recover the causal order from observational data.

1.2 Outline

In Chapter 2 we present the background material of Causal Inference, reviewing the primary results in the field. Chapter 3 is devoted to motivate and introduce the required background for heavy-tailed distributions and in particular regularly varying distributions. The central part starts with Chapter 4, where we present a novel method to estimate a causal order from heavy-tailed data. In this chapter, we prove the main results for the Γ coefficient, and we construct four competing algorithms. In Chapter 5 we compare the four algorithms and assess their performance in simulations. Also, we test empirically whether the algorithms can deal with hidden confounding variables.

1.3 Contributions

This thesis builds upon the idea of Engelke et al. (2018). In their paper, they present a binary coefficient Γ to detect the causal direction for pairs of heavy-tailed variables. Also, they show how to compute the Γ coefficient for an SEM of two variables and one hidden confounder. We extend this result to the case of p variables, under the assumption of regularly varying distribution for the noise (proposition 4.2.2.2). Based on this, we prove that it is possible to detect the source node of a linear SEM with no hidden confounders (lemma 4.3.0.5 and lemma 4.4.0.1). We introduce four competing algorithms to recover the causal order from observational data. We test these algorithms on some simulated SEMs to assess their sample properties.

To summarise our contributions:

1. We prove the population properties of the Γ coefficient for a random vector $X = (X_1, \dots, X_p)$ modeled by a linear SEM. We assume that the β's are positive and the noise distribution is regularly varying (a subclass of heavy-tailed data).
2. In the population case, we prove that it is possible to identify the source node of a linear SEM (under the same assumptions of point 1). In our proof, we do not allow for hidden confounders.
3. We build four algorithms to estimate the causal order of a random vector $X = (X_1, \dots, X_p)$ from observational data (under the same assumptions of point 1)[1].

---

[1] All results are fully reproducible in R. The code is available upon request, and we plan to publish it on Github.

Chapter 2

Causality

In this chapter, we review the necessary background in Causal Inference to better the results of Chapter 4. The presentation of the material is based on the Causality lectures held by Professor Maathuis (2017) and Professor Meinshausen (2018) at ETH Zurich.

2.1 Graphs

We start by reviewing some definitions in Graph Theory.

- A **graph** $G = (V, E)$ is a pair that consists of a set of vertices (nodes) V and a set of edges $E \subseteq V \times V$.
- An edge $(i, j) \in E$ is called **undirected** if $(i, j) \in E$ and $(j, i) \in E$. We denote it with $i \sim j$
- An edge $(i, j) \in E$ is called **directed** if $(i, j) \in E$ and $(j, i) \notin E$. We denote it with $i \rightarrow j$
- A graph G is called **undirected** if all its edges are undirected
- A graph G is called **directed** if all its edges are directed
- For each pair of nodes $i, j \in V$, there is **at most one edge**
- For two nodes $i, j \in V$, i is a **parent** of j if $i \rightarrow j$
- For two nodes $i, j \in V$, i is a **child** of j if $i \leftarrow j$
- Two nodes $i, j \in V$ are said to be **adjacent** if $i \rightarrow j$ or $i \leftarrow j$
- Given a node $i \in V$, we call $pa(i)$ the set of its **parents**, $ch(i)$ the set of its **children** and $adj(i)$ the set of its **adjacent** nodes[1]
- A node $i \in V$ with no parents is called **root** node (or **source** node)
- A **path** in G is a sequence of (at least two) distinct vertices $i_1, ..., i_n$, such that there is an edge between i_k and i_{k+1}, for $k = 1, ..., n-1$. [2]
- A **directed path** is a path in which $i_k \rightarrow i_{k+1}$ for $k = 1, ..., n-1$
- A (**directed**) **n-cycle** is a (directed) path of length n in which $i_1 = i_n$
- A **directed acyclic graph** (**DAG**) is a directed graph that contains no directed cycles
- A **fully connected DAG** is a DAG where all pairs of vertices are joined by a directed edge

---

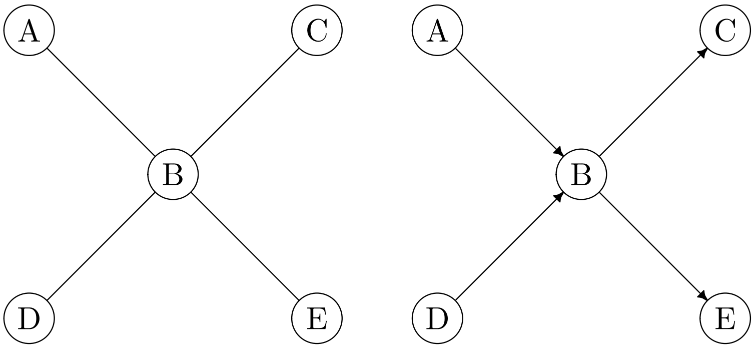
[1] Sometimes, we write $pa(i, G)$, $ch(i, G)$ and $adj(i, G)$ to emphasize the dependence on G

[2] By abuse of notation, we denote that an edge $(l, k) \in E$ belongs to a path $\boldsymbol{p}$ by writing $(l, k) \in \boldsymbol{p}$.

- For two nodes $i, j \in V$, i is an **ancestor** of j if there exists a directed path between i and j
- For two nodes $i, j \in V$, i is a **descendant** of j if there exists a directed path between j and i [3]
- Given a node $i \in V$, we call $an(i)$ the set of its **ancestors** and $desc(i)$ the set of its **descendants**
- Given a node $i \in V$, we define $nonan(i) := V \setminus an(i)$ the set of its **non-ancestors** and $nondesc(i) := V \setminus desc(i)$ the set of its **non-descendendant**
- Three nodes are called a **v-structure** if one node, say $i \in V$, has two parents that are themselves not adjacent. In this case, we say that i is a **collider**
- A path between i and j is **blocked** by $S \subset V$ (not containing i or j) if at least one of the following holds:
  - There is a non-collider on the path that is in S
  - There is a collider on the path and neither this collider (nor any of its descendants) is in S
- Two nodes $i, j \in V$ are **d-separated** by $S \subset V$, d-sep$(i, j; S)$, if all paths between i and j are blocked by S. Otherwise they are **d-connected**, d-conn$(i, j; S)$.
- An undirected graph G is a **tree** if between any pair of nodes $i, j \in V$ there exists one and only one path
- The **skeleton** of a DAG G is the undirected graph obtained by removing the edge directions
- A DAG G is a **polytree** if its skeleton is a tree

Below we present some examples to describe the most important properties of a graph.

Example 2.1.0.1 (Undirected and Directed Graphs).



(a) Undirected graph (b) Directed graph

---

[3] By definition, we say that each node is an ancestor and a descendant of itself.

Example 2.1.0.2 (Parents and Children)**.**

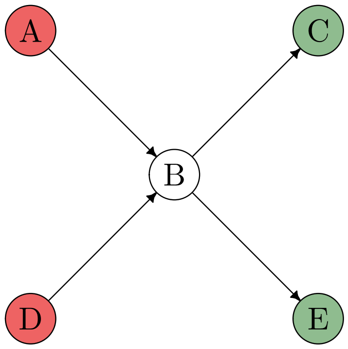
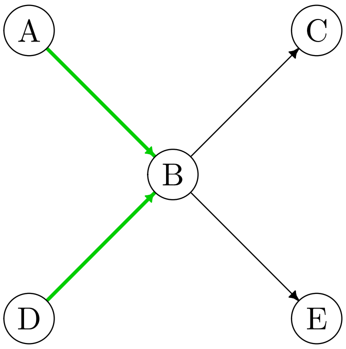
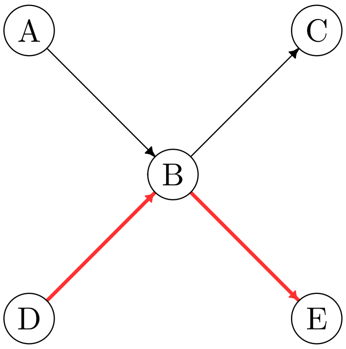


Figure 2.2: Parents of B (red) – Children of B (green)

Example 2.1.0.3 (Path and Directed Path)**.**

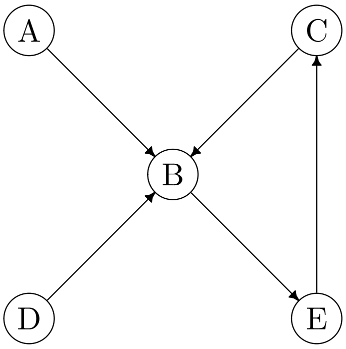


(a) Path (green)

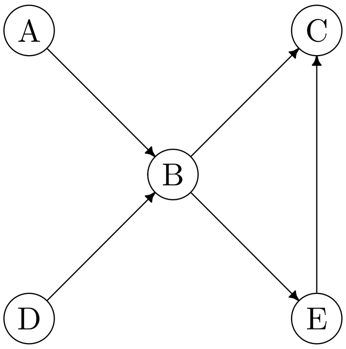


(b) Directed path (red)

Example 2.1.0.4 (Directed Cyclic and Acyclic Graph)**.**

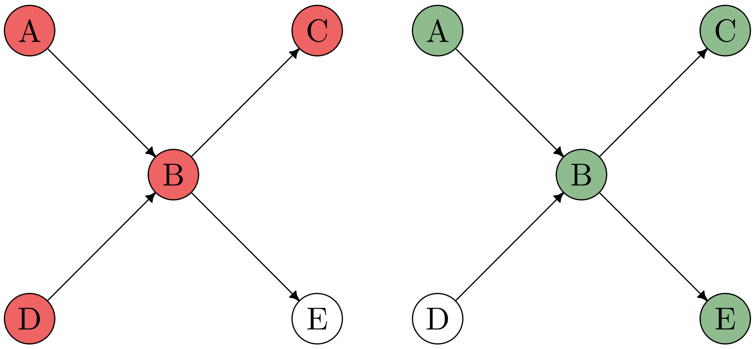


(a) Directed Cyclic Graph



(b) Directed Acyclic Graph (DAG)

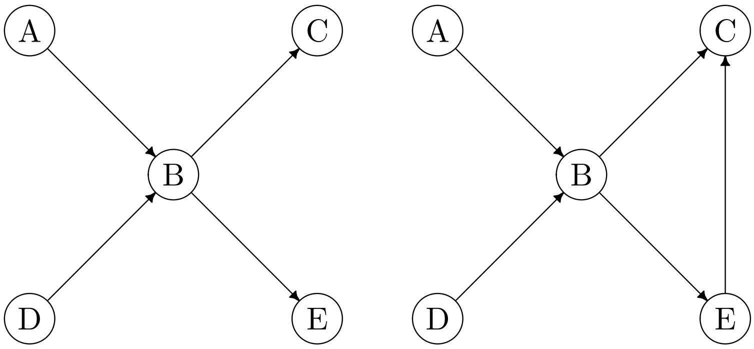
Example 2.1.0.5 (Ancestors and Descendants)**.**


A
C
B
D
E
A
C
B
D
E

(a) Ancestors of C (red)

(b) Descendants of A (green)

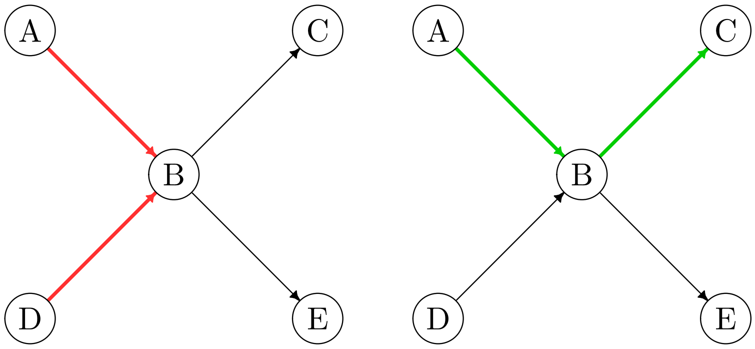
Example 2.1.0.6 (Polytree and Non-Polytree)**.**


A
C
B
D
E
A
C
B
D
E

(a) Polytree

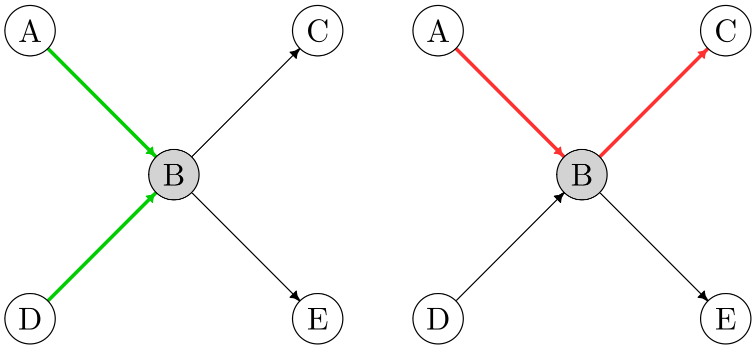
(b) Non-Polytree

Example 2.1.0.7 (Blocked Paths and d-separation)**.**



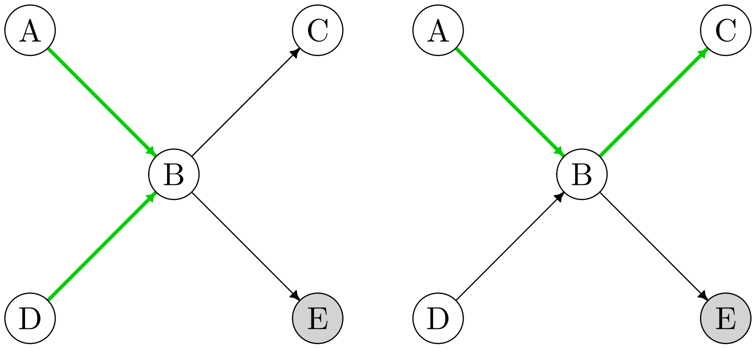
(a) Path A to D (red) is blocked; d-sep$(A, D; \emptyset)$

(b) Path A to C (green) is open; d-conn$(A, C; \emptyset)$



(c) Observing B, path A to D (green) is open; d-conn$(A, D; \{B\})$

(d) Observing B, path A to C (red) is blocked; d-sep$(A, C; \{B\})$



(e) Observing E, path A to D (green) is open; d-conn$(A, D; \{E\})$

(f) Observing E, path A to C (green) is open; d-conn$(A, C; \{E\})$

At this point, we introduce an important definition for a DAG, called **topological** (or **causal**) **order**, as presented in Peters, Janzing, and Schölkopf (2017), Appendix B, Definition B.1.

Definition 2.1.0.8. *Given a DAG G, we call a permutation, that is, a bijective mapping,*

$$\pi : \{1, \ldots, p\} \to \{1, \ldots, p\},$$

*a **topological** (or **causal**) **order** if it satisfies*

$$\pi(i) < \pi(j) \text{ for all } i \in an(j)$$

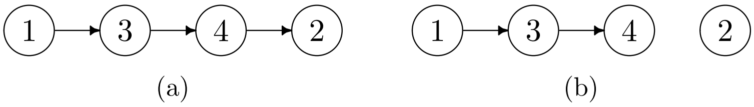
Due to the acyclicity of a DAG, we can always find a **topological order**[4]. This result is given in the following proposition.

Proposition 2.1.0.9. *For each DAG $G = (V, E)$, there is always a* ***topological order****.*

The proof can be found in Appendix A.1.

It is also interesting to note that for a DAG, the **topological order** need not be unique. We can see this in the following example.

Example 2.1.0.10. Here we show two DAGs. The one on the left has only one valid topological order, namely $\pi = (1, 4, 2, 3)$. The one on the right has four valid topological orders, which are $\pi_1 = (1, 4, 2, 3)$, $\pi_2 = (1, 3, 2, 4)$, $\pi_3 = (1, 2, 3, 4)$ and $\pi_4 = (2, 1, 3, 4)$.



Most of the times, we think of a graph as a set of nodes connected by some edges. However, it is often useful to represent a graph using a special matrix called **adjacency matrix**. We will encounter this matrix (and some variations of it) again in Chapter 5.

Definition 2.1.0.11. *Let $G = (V, E)$ be a DAG with p vertices. We define the matrix $A \in \{0, 1\}^{p \times p}$ an* ***adjacency matrix*** *if it satisfies*

$$A_{ij} = \begin{cases} 1, & \text{if } i \to j \\ 0, & \text{else} \end{cases}$$

2.2 Bayesian Networks

When we combine DAGs, random variables and probabilities, we come up with an object that plays a central role in Causal Inference, namely the **Bayesian Network**.

Definition 2.2.0.1. *Let $G = (V, E)$ be a DAG with p nodes. Let $X = (X_i)$, $i \in V$ be a set of random variables indexed by V*[5] *that follow a joint distribution P with density f. Then, we call the pair (G, P) a* ***Bayesian Network*** *if the distribution P factorizes*[6]*,*

$$f(x_1, \ldots, x_p) = \prod_{i=1}^{p} f(x_i | \, x_{pa(i)})$$

---

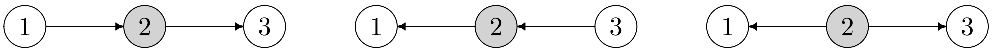
[4] For convenience, we denote $\pi = (1, 3, 2, 4)$ the **topological order** $\pi : \{1, 2, 3, 4\} \to \{1, 3, 2, 4\}$

[5] Sometimes, with abuse of notation, we write X_i to refer to node i.

[6] We use "P factorizes as ..." to say "Given a joint distribution P with density f, f factorizes as ...".

It is important to note that each joint distribution P factorizes according to a fully connected DAG G'. In other words, any distribution P with density f can be written as $f(x_1, \dots, x_p) = \prod_{i=1}^{p} f(x_i|\ x_{pa(i,G')})$, where G' is a fully connected DAG. Also, for some distribution P it is possible to find more than one valid **Bayesian Network**, as we show in the following example.

Example 2.2.0.2. Let $X = (X_1, X_2, X_3)$ be a random vector that follows a joint distribution P with density f. Assume that X_1 is conditionally independent of X_3 given X_2, i.e., $f(x_1, x_3|\ x_2) = f(x_1|\ x_2)\ f(x_3|\ x_2)$. We see that P can factorize according to three Bayesian Networks



(a) $BN_1 = (G_1, P)$ (b) $BN_2 = (G_2, P)$ (c) $BN_3 = (G_3, P)$

For BN_1 we have:

$$\begin{aligned} f(x_1, x_2, x_3) &= f(x_1)\ f(x_2|\ x_1)\ f(x_3|\ x_1, x_2) \\ &= f(x_1)\ f(x_2|\ x_1)\ f(x_3|\ x_2) \end{aligned}$$

For BN_2 we have:

$$\begin{aligned} f(x_1, x_2, x_3) &= f(x_3)\ f(x_2|\ x_3)\ f(x_1|\ x_3, x_2) \\ &= f(x_3)\ f(x_2|\ x_3)\ f(x_1|\ x_2) \end{aligned}$$

For BN_3 we have:

$$\begin{aligned} f(x_1, x_2, x_3) &= f(x_2)\ f(x_1|\ x_2)\ f(x_3|x_2, x_1) \\ &= f(x_2)\ f(x_1|\ x_2)\ f(x_3|\ x_2) \end{aligned}$$

The following definitions are particularly relevant, as they put in relation the world of graph theory with the world of probability theory.

Definition 2.2.0.3. *Let $X \in \mathbb{R}^p \sim P$ be a random vector and let $G = (V, E)$ be a DAG. Then, the joint distribution P is said to satisfy the* ***Local Markov property*** *with respect to G if and only if*

$$X_i \perp\!\!\!\perp X_{nondesc(i) \setminus pa(i)}|\ X_{pa(i)}, \quad \forall i \in V$$

In words, every node is independent of its non-descendants given its parents.

Definition 2.2.0.4. *Let $X \in \mathbb{R}^p \sim P$ be a random vector and let $G = (V, E)$ be a DAG. Then, the joint distribution P is said to satisfy the **Global Markov property** with respect to G if and only if*

$$\text{If } A \text{ and } B \text{ are d-separated by } S\text{, then } X_A \perp\!\!\!\perp X_B |\ X_S$$

*where $A \subset V$, $B \subset V$, and $S \subset V$. In words, any d-separation that we read from the graph G implies a conditional independence in the distribution P. For this reason we also say that G is an **Independence Map** (I-Map) of P.*

Definition 2.2.0.5. *Let $X \in \mathbb{R}^p \sim P$ be a random vector and let $G = (V, E)$ be a DAG. Then, the joint distribution P is said to satisfy the **Markov factorization property** with respect to G if and only if*

$$f(x_1, \ldots, x_p) = \prod_{i=1}^{p} f(x_i |\ x_{pa(i)})$$

where we assume that P has a valid density f.

A significant result states that the three definitions above are equivalent if the joint distribution P has a positive and continuous density f (see Theorem 5, Section 3.2.1., Pearl (1988)).

We can see that the Global Markov property works only in one direction. More precisely, if two nodes in the graph are d-separated, then we know that the corresponding random variables are conditionally independent but not the other way round. It would be nice to have a one-to-one correspondence between the d-separation from the graph and the conditional independence from the distribution. For this reason, we introduce the concept of **Faithfulness**.

Definition 2.2.0.6. *Let $X \in \mathbb{R}^p \sim P$ be a random vector and let $G = (V, E)$ be a DAG. Then, the distribution P is said to be **faithful** to the graph G if and only if*

$$\text{If } X_A \perp\!\!\!\perp X_B |\ X_S\text{, then } A \text{ and } B \text{ are d-separated by } S$$

for $A \subset V$, $B \subset V$, and $S \subset V$. In words, any d-connection that we read from the graph G implies a conditional dependence in the distribution P.

When we assume both the **Global Markov property** and **faithfulness**, we are sure that there is a one-to-one correspondence between the d-separations in the graph and the conditional independences in the distribution. In this case, we say that the DAG G is a **Perfect Map** of P. This is very useful when we want to infer the graph structure from observational data.

2.3 Causal Bayesian Networks

Suppose we observe $(X_1, Y_1), \ldots, (X_n, Y_n)$ i.i.d. copies of (X, Y). When we want to study the association between X and Y, we try to model:

- $P(Y|\ X = x)$ for a classification problem
- $\mathrm{E}[Y|\ X = X]$ for a regression problem

However, this is not enough when we want to talk about causation. In fact, we should remember the adage: "association does not imply causation". If we want to make causal statements, we must add new toolkits to our arsenal. We start with a provisional definition of **causal effect**.

Definition 2.3.0.1 (Provisional definition)**.** *We say that X has a* ***causal effect*** *on Y if intervening on X has an effect on the distribution of Y.*

Here, we are trying to answer a more ambitious question than simply modeling a conditional probability (or expectation). We want to understand how the system changes under specific actions/interventions. Most of the times, however, we are not allowed to actively manipulate the system (e.g., for ethical or practical reasons), and we only have access to observations generated by the system. The three tools used to tackle these kinds of issues are:

1. Causal Bayesian Networks and *do-calculus*
2. Structural Equation Models (SEM)
3. Counterfactuals and potential outcomes

In this thesis, we present only the first two points.

2.3.1 Causal Bayesian Network and *do-calculus*

We introduce the notation $\mathrm{do}(X = \tilde{x})$ to represent a hypothetical intervention where X is set to the value $\tilde{x}$ uniformly over the entire population.

- $f(y|\ \mathrm{do}(X = \tilde{x}))$ denotes the marginal density of Y after we intervene on X and set it to $\tilde{x}$, i.e., $\mathrm{do}(X = \tilde{x})$.

Definition 2.3.1.1 (Formal definition)**.** *Let (X, Y) be a random vector that follows a joint distribution P with density f. We say that X has a* ***causal effect*** *on Y if $f(y|\ do(X = \tilde{x}))$ depends on $\tilde{x}$.*

In other words, there exists $x' \neq \tilde{x}$ such that

$$f(y|\ do(X = x')) \neq f(y|\ do(X = \tilde{x}))$$

Definition 2.3.1.2. *Let $G = (V, E)$ be a DAG. Let $X = (X_i)$, $i \in V$, be a random vector that follows a joint distribution P with density f. Let P factorize according to G. We say that (G, P) is a* ***Causal Bayesian Network*** *if*

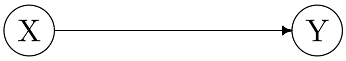
$$f(x_1, \dots, x_p | do(X_W = \tilde{x}_w)) = \begin{cases} \prod_{i \notin V \setminus W} f(x_i|\ x_{pa(i,G)}), & \textit{if}\ \ X_W = \tilde{x}_w \\ 0, & \textit{else} \end{cases}$$

$$= \prod_{i \notin V \setminus W} f(x_i|\ x_{pa(i,G)})\ \mathbb{1}[X_W = \tilde{x}_w]$$

where $W \subset V$.

From definition 2.3.1.2, we see that manipulating X_W only modifies the terms $f(x_j|\, x_{pa(j,G)})$ into $\mathbb{1}[X_j = \tilde{x}_j]$, for $j \in W$. All other terms remain unchanged. In this regard, we can say that the intervention acts "locally" on the data generating process and the system is modular (see for example Haavelmo (1944)). Some authors, such as Peters et al. (2017), describe this property as the *principle of independent mechanisms.*

Example 2.3.1.3. Consider the causal Bayesian Network (CBN) shown in the figure below. Let (X, Y) be a discrete random vector that follows a joint distribution P with probability mass function (pmf) f.



From definition 2.3.1.2, we know that $f(x, y|\, \mathrm{do}(X = \tilde{x})) = f(y|x)\ \mathbb{1}[X = \tilde{x}]$. Hence, we can compute the marginal of Y when we intervene on X and the marginal of X when we intervene on Y.

- Marginal of Y: $f(y|\, \mathrm{do}(X = \tilde{x})) = \sum_x f(x, y|\, \mathrm{do}(X = \tilde{x})) = \sum_x f(y|x)\ \mathbb{1}[x = \tilde{x}] = f(y|\tilde{x})$. In this case, we see that the (observational) conditional pmf $f(y|\tilde{x})$ is equal to the (interventional) pmf $f(y|\, \mathrm{do}(X = \tilde{x}))$. In this situation, if X and Y are correlated, then X causes Y. In fact, if $f(y|\, \tilde{x})$ depends on $\tilde{x}$ so does $f(y|\, \mathrm{do}(X = \tilde{x}))$ (and vice-versa).
- Marginal of X: $f(x|\, \mathrm{do}(Y = \tilde{y})) = \sum_y f(x, y|\, \mathrm{do}(Y = \tilde{y})) = \sum_y f(x)\ \mathbb{1}[y = \tilde{y}] = f(x) \sum_y \mathbb{1}[y = \tilde{y}] = f(x)$. Where the second equality follows from the definition of CBN. In this case, we observe that $f(x|\, \mathrm{do}(Y = \tilde{y}))$ does not depend on $\tilde{y}$ and hence Y does not cause X.

Note that in general, the interventional distribution is different from the conditional distribution. Regarding the modularity, we note that every time we intervene on one variable, we only affect a limited part of the mechanism. Recall that the joint pmf in the example equals $f(x, y) = f(x)\ f(y|x)$. When we manipulate $X = \tilde{x}$, we obtain a modified system in which only the term $f(x)$ is affected: $f(x, y|\, \mathrm{do}(X = \tilde{x})) = \mathbb{1}[X = \tilde{x}]f(y|x)$. In this regard, we say that the mechanisms $f(x)$ and $f(y|x)$ are independent (see Peters et al. (2017), Section 2.1).

2.3.2 Structural Equation Models (SEM)

Another way to represent a causal system is by using **Structural Equation Models** (SEMs). SEMs describe a generating mechanism by modeling each variable as a function of its parental graph and some noise.

Definition 2.3.2.1. *Let $G = (V, E)$ be a DAG. Let $X = (X_i)$, $i \in V$, be a random vector. Assume that each X_i is generated by some function of its parents and noise Y_i. An SEM is a pair $\mathbb{S} = (S, P_Y)$,*

$$S_i : \qquad X_i \leftarrow h_i(X_{pa(i,G)}, Y_i), \qquad \textit{for all } i \in V$$

where $S = \{S_1, \ldots, S_p\}$ *is a set of equations and* P_Y *is the joint distribution of the independent noise* Y_i*, for* $i \in V$*.*

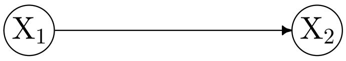
Here, we interpret the assignment operator "$\leftarrow$" as we do in programming. Hence, we can generate each variable only after its ancestors are assigned a value. The graph of an SEM is obtained by drawing direct edges from each variable occurring on the right-hand side of the equations to the variables on the left-hand side. In this way, we hand up with a DAG.

Proposition 2.3.2.2. *Let* $G = (V, E)$ *be a DAG. Let* $X = (X_i)$*,* $i \in V$*, be a random vector. Assume that there is an SEM related to* G*. Due to the acyclic structure, an SEM defines a unique distribution over* X_V *such that* $X_i = h_i(X_{pa(i,G)}, Y_i)$*, in distribution, for all* $i \in V$*.*

The proof can be found in Appendix A.2.

In case of an SEM, it is very straightforward to account for intervention: all we need to do is reassigning the value of the variables that we manipulate.

Example 2.3.2.3. Consider the following DAG G,



and the related SEM,

$$\begin{aligned} X_1 &\leftarrow Y_1 \\ X_2 &\leftarrow h_2(X_1, Y_2) \end{aligned}$$

for some function h_2. Suppose we make an intervention and we set $X_1 \leftarrow \tilde{x}_1$. We reflect this change in the SEM as follows:

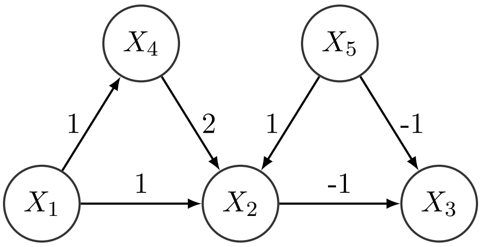
$$\begin{aligned} X_1 &\leftarrow \tilde{x}_1 \\ X_2 &\leftarrow h_2(X_1, Y_2) \end{aligned}$$

This is equivalent to the operator $\text{do}(X_1 = \tilde{x}_1)$ that we saw in the section 2.3.1. Also, we note here the modularity of the system.

To summarise, the connection between SEM and CBN goes as follows. We start with a DAG $G = (V, E)$ and a set of random variables $X = (X_i)$, where $i \in V$. We assign a value to each random variable, using the SEM $X_i \leftarrow h_i(X_{pa(i,G)}, Y_i)$. This set of equations defines a unique distribution P over X_V that factorizes according to G, so we have a Bayesian network (G, P). When we consider interventions, as shown in example 2.3.2.3, we obtain a CBN.

In the particular case where all the equations are linear and additive in the noise, we can quickly read off from the SEM (or equivalently from the CBN) the causal effects between the variables. Below we present an example of a **linear** SEM.

Example 2.3.2.4. Consider the DAG G,



With related SEM,

$$
\begin{aligned}
&X_1 \leftarrow Y_1\\
&X_5 \leftarrow Y_5\\
&X_4 \leftarrow (1)\cdot X_1 + Y_4\\
&X_2 \leftarrow (1)\cdot X_1 + (2)\cdot X_4 + (1)\cdot X_5 + Y_2\\
&X_3 \leftarrow (-1)\cdot X_2 + (-1)\cdot X_5 + Y_3
\end{aligned}
$$

We see that X_1 is cause of X_2, X_3 and X_4, since they are connected by a directed path starting from X_1. To compute the *direct* causal effect, it suffices to read off the coefficients on the directed edges. For example, X_1 has a direct causal effect of 1 on both X_2 and X_4. To compute the *total* causal effect of X_i on X_j, we must:

a. multiply the edge coefficients along each directed path from i to j

b. sum up the results over all paths

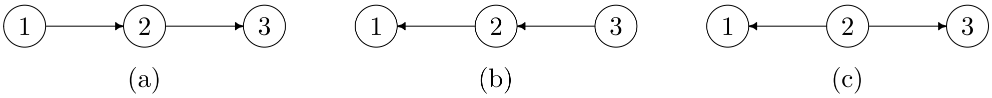
For example, the total causal effect of X_1 on X_2 equals to $1 \cdot 2 + 1 = 3$. We see that $1 \cdot 2$ is the causal effect along the directed path X_1, X_4, X_2, whereas 1 is the direct causal effect of X_1 on X_2. If we want to compute the total causal effect of X_1 on X_3 we obtain $1\cdot(-1)+1\cdot 2\cdot(-1) = -3$. In this case, $1\cdot(-1)$ is the causal effect along the directed path X_1, X_2, X_3, whereas $1 \cdot 2 \cdot (-1)$ is the causal effect along the directed path X_1, X_4, X_2, X_3.

2.4 Causal Structure Learning

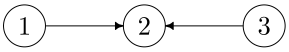
In this section, we review the main methods to infer the causal structure from observational data. As noted by Goudet, Kalainathan, Caillou, Lopez-Paz, Guyon, Sebag, Tritas, and Tubaro (2017), these methods can be divided into two classes: those based on the Markov properties of a DAG and those based on the non-Gaussianity of the noise distribution.

2.4.1 Class 1: Learning the Markov Equivalence Class

In example 2.2.0.2, we have seen that different Bayesian Networks can describe the same distribution P. Thus, from the knowledge of P alone, often we cannot reconstruct an exact DAG. What we can do is to identify a class of DAGs that represent a plausible causal structure of the given data. Recalling example 2.2.0.2, if $(X_1, X_2, X_3) \sim P$ such that $X_1 \perp\!\!\!\perp X_3 |\, X_2$, then we know that the plausible DAGs are:



On the other hand, if we know that $X_1 \perp\!\!\!\perp X_3$ then we can infer that the unique DAG G that is an I-Map for P is:



This example leads to the following definition.

Definition 2.4.1.1. *Two DAGs G_1 and G_2 are said* ***Markov Equivalent*** $(G_1, G_2) \in ME$ *if they describe the same set of d-separations.*

The Markov Equivalence property describes an equivalence relation ME over DAGs. Hence, given a DAG G, it is possible to define the Markov Equivalence class $[G] := \{g | \, (g, G) \in ME\}$. An important result from Vermat and Pearl (1990) states that all DAGs in a Markov Equivalence class share the **same skeleton** and the **same v-structures**. To represent equivalence classes of DAGs pictorially we introduce the concept of CPDAG.[7]

Definition 2.4.1.2. *Let $G = (V, E)$ be a DAG and $[G]$ its Markov Equivalence class.*

We define ***CPDAG*** *a graph with the following properties:*

- *There is an edge between i and j if and only if i and j are d-connected given S, $S \subset V \setminus \{i, j\}$, for all $g \in [G]$*
- *There is a directed edge from i to j if and only if i is a parent of j for all $g \in [G]$*
- *There is an unidentified edge between i and j if and only if there is a DAG $g \in [G]$ with $i \rightarrow j$ and a DAG $g' \in [G]$ with $i \leftarrow j$*

where $i, j \in V$

Constraint-based methods are all those algorithms that exploit conditional independences between variables to learn the CPDAG. One of the first constraint-based methods is the SGS-algorithm of Spirtes, Glymour, and Scheines (1990). SGS-algorithm starts from a fully connected graph. First, it tries to recover the skeleton of the graph, running d-separation tests between all pair of adjacent nodes given some subset of the remaining nodes. Secondly, it identifies v-structures. Finally, it orients as many of the remaining nodes as possible. The problem with this method, however, is the algorithmic complexity. As stated by Spirtes et al. (2000), in the worst case, the SGS-algorithm requires a number of d-separation tests that increases exponentially with the number of vertices.

An improvement over the SGS-algorithm is given by the PC-algorithm of Spirtes et al. (2000), which reduces the number of d-separation tests. The idea is the following. Starting from a fully connected graph, for each pair of adjacent nodes (i, j) the algorithm removes an edge if X_i and X_j are conditionally independent given some subset of size $k = 0, 1, \ldots$ of $adj(i)$ or $adj(j)$. Once the skeleton is recovered, PC-algorithm determines the v-structures and finally directs as many edges of the remaining ones as possible.

---

[7] The definition is based on the Lecture Notes of the Causality course held by Professor Maathuis (2017).

The FCI-algorithm of Spirtes, Meek, and Richardson (1995) extends PC-algorithm to the case of latent variables. A further improvement is given by RFCI-algorithm of Colombo, Maathuis, Kalisch, and Richardson (2012), which deals with both latent variables and high-dimensional DAGs.

Score-based methods are all those algorithms that learn the CPDAG by minimizing a particular score. Some methods aim at minimizing a Likelihood score. Other methods, like the Greedy Equivalent Search (GES) algorithm of Chickering (2002), add a penalization term to the score using the Bayesian Information Criterion, in order to keep the CPDAG as sparse as possible. An improvement is given by the Fast Equivalent Search (FGES) algorithm of Ramsey (2015), which speeds up the computation using different data structures compared to GES-algorithm.

2.4.2 Class 2: Leveraging the non-Gaussian noise

These classes of algorithm exploit the non-Gaussianity of the noise to infer the causal structure of the observational data.

The first approach in this direction is given by Shimizu et al. (2006), with the LiNGAM-algorithm, which stands for Linear non-Gaussian Acyclic Models. The presentation is based on the Lecture Notes of the Causality course held by Professor Maathuis (2017). The starting point is a linear SEM related to a DAG $G = (V, E)$.

$$X_i \leftarrow \sum_{j \in V} \beta_{ij} X_j + Y_i, \quad \forall i \in V$$

Since the SEM specifies a DAG G, there exists a topological order π (see proposition 2.1.0.9), such that we can write:

$$X_i \leftarrow \sum_{\pi(j) < \pi(i)} \beta_{ij} X_j + Y_i, \quad \forall i, j \in V$$

If we write this in matrix form, we obtain a linear system with a strictly lower triangular matrix:

$$X \leftarrow BX + Y$$

where $X, Y \in \mathbb{R}^p$ and $B \in \mathbb{R}^{p \times p}$. The non-zero entries in the B matrix are equivalent to the edges in the DAG. Moreover, due to acyclicity, there is an ordering of the variables (a topological order) such that B is strictly lower triangular. Regarding the noise Y_i, $i \in V$, LiNGAM makes the following assumptions:

- Mutually independent (no hidden variables)
- Mean zero
- Non-zero variance
- **Non-Gaussian** distribution

Example 2.4.2.1. Consider the following SEM,

$$\begin{aligned} X_3 &\leftarrow Y_3 \\ X_1 &\leftarrow 3 \cdot X_3 + Y_1 \\ X_2 &\leftarrow -5 \cdot X_1 + Y_2 \end{aligned}$$

Considering the topological order $\pi = (2, 3, 1)$, the SEM can write in matrix form,

$$X := \begin{bmatrix} X_3 \\ X_1 \\ X_2 \end{bmatrix} \leftarrow \begin{bmatrix} 0 & 0 & 0 \\ 3 & 0 & 0 \\ 0 & -5 & 0 \end{bmatrix} \begin{bmatrix} X_3 \\ X_1 \\ X_2 \end{bmatrix} + \begin{bmatrix} Y_3 \\ Y_1 \\ Y_2 \end{bmatrix} =: BX + Y$$

We see that matrix B is strictly lower triangular once the variables are arranged according to a topological order.

Example 2.4.2.2. Consider the following SEMs,

$$\begin{aligned} X_1 &\leftarrow Y_1 \\ X_2 &\leftarrow 0.8 \cdot X_1 + Y_2 \end{aligned} \tag{2.4.2.1}$$

$$\begin{aligned} X_2 &\leftarrow Y_2 \\ X_1 &\leftarrow 0.8 \cdot X_2 + Y_1 \end{aligned} \tag{2.4.2.2}$$

Let the covariance matrix of $X = (X_1, X_2)$ in *both* models be

$$\Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

If we assume that Y_i is Gaussian with mean zero, for $i = 1, 2$, then $X = (X_1, X_2) \sim N(0, \Sigma)$, in both models. We see that both models generate the same Gaussian distribution, even if they are causally different. We have a *non-identifiability* problem.

Instead, if we assume that Y_i is non-Gaussian, for $i = 1, 2$, then we have a chance to identify the causal structure from the data. Consider figure 2.11.

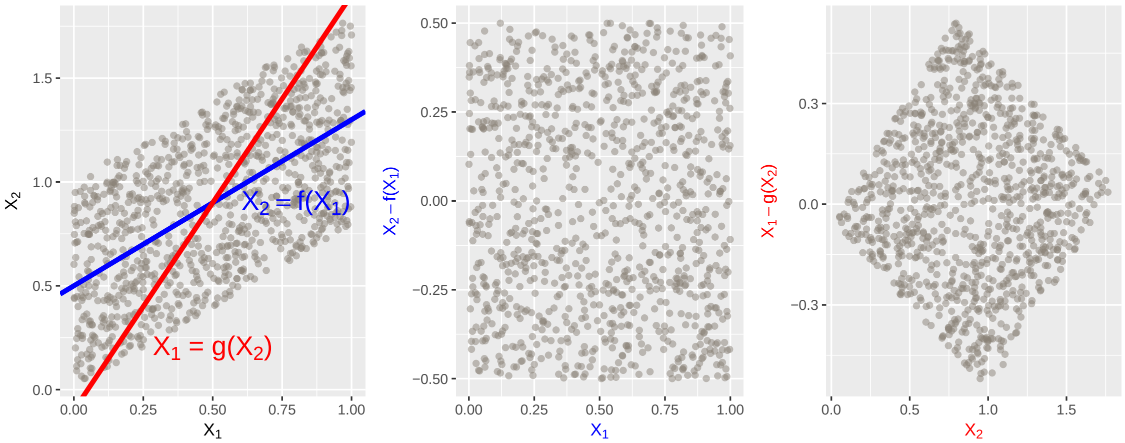


Figure 2.11: Left: Joint distribution $P(X_1, X_2)$ generated by the SEM 2.4.2.1. The linear regression of X_2 on X_1 (respectively of X_1 on X_2) is shown as a blue (resp. red) line. Middle: Residuals $X_2 - f(X_1)$ are independent of X_1. Right: Residuals $X_1 - g(X_2)$ depend on X_2.

Suppose that X_1 causes X_2 as shown in the SEM 2.4.2.1. If we assume that the noise $Y_i \sim U[0,1]$ is uniformly distributed, for $i = 1, 2$, then we can infer the causal direction from the data. Indeed, if we regress X_2 on X_1, the residuals look independent of X_1. On the other hand, if we regress X_1 on X_2, then the residuals do depend on X_2. This asymmetry can be exploited to infer the causal direction $X_1 \to X_2$.

The first version of LiNGAM (Shimizu et al., 2006), called *ICA*-LiNGAM, is closely related to *Independent Component Analysis* (ICA) (Comon et al., 1991). This method has the advantage of exploiting the well-known techniques for ICA. For this reason, it is computationally efficient. As a downside, however, the algorithm may converge to local optima, as it happens in most ICA-based algorithms. An improvement over ICA-LiNGAM is *Direct*-LiNGAM of Shimizu, Inazumi, Sogawa, Hyvärinen, Kawahara, Washio, Hoyer, and Bollen (2011). The steps are the following:

1. Find an exogenous variable (i.e., a *root* node) based on its independence from the residuals of all pairwise regressions between the variables.
2. Remove the effect of the *root* node from the other variables using least square regression.
3. Repeat points 1. and 2. on the residuals of the remaining variables, until no variables are left.
4. Return a causal order π of the variables.
5. Build a strictly lower triangular matrix B, following the causal order π.
6. Run sparse regression (see Tibshirani (1996)) to determine the zero coefficients in the B matrix. More precisely, regress each variable on its predecessors in the causal order π

Before closing the chapter, we want to highlight the following. The algorithms that we present in Chapter 4 were inspired in part by the reading of Direct-LiNGAM of Shimizu et al. (2011).

Chapter 3

Heavy-Tailed Distributions

In this chapter, we will talk about heavy-tailed distributions and in particular regularly varying distributions. We start by giving some intuition and real-life properties related to heavy-tailed distributions. Next, we formalize these ideas and present some meaningful results that will be used in Chapter 4.

3.1 Intuition

In order to understand heavy-tailed distributions, we think it is a good idea to compare them to the light-tailed distributions. Light-tailed distributions comprise most of the ones usually taught in introductory Statistics courses, e.g., Gaussian and Exponential.

A first property that differentiates heavy-tailed distributions to light-tailed distributions is the **scale invariance**. More precisely, regularly varying distributions have tails that decay more slowly than exponential tails. Consider the Exponential distribution, which is light-tailed, and the Pareto distribution, which has regularly varying tails. Denoting $\overline{F}$ the right-tail cumulative distribution function, we have:

- Exponential: $\overline{F}(x) = e^{-\lambda x}, \quad x \geq 0$
- Pareto: $\overline{F}(x) = c^{\alpha} x^{-\alpha}, \quad x \geq c$

To understand why we say scale invariant, note that, if we scale up the x to $y = 1000 \cdot x$, we obtain $\overline{F}(y) = \overline{F}(1000 \cdot x) = c^{\alpha}(1000 \cdot x)^{-\alpha} = 1000^{-\alpha} c^{\alpha} x^{-\alpha} = 1000^{-\alpha}\overline{F}(x) = k \cdot \overline{F}(x)$, for $x \geq c/1000$. Thus, changing the scale of x does not change the appearance of $\overline{F}(x)$. It just gets rescaled by a constant factor (in our case $k = 1000^{-\alpha}$). In figure 3.1, we see the difference between the Exponential and the Pareto tail.

The second aspect we want to discuss is the **catastrophe** and the **conspiracy principle**, two definitions proposed by Nair, Wierman, and Zwart (2018). To understand the two ideas, we give an example based on Nair et al. (2018). Suppose you are told that there are five people in a room that you have never seen before. Also, you are given the following information: the total height of the five people is ten meters, and the total number of their followers on Instagram is ten million. Without even seeing the group, you will presume that the total height of ten meters is due to most of the people being around two meters tall. It could be a basketball team. Most likely, you would never think that the sum of ten meters stems from four people 1.8 meters tall and one giant 2.8 meters tall. You expect the sum is

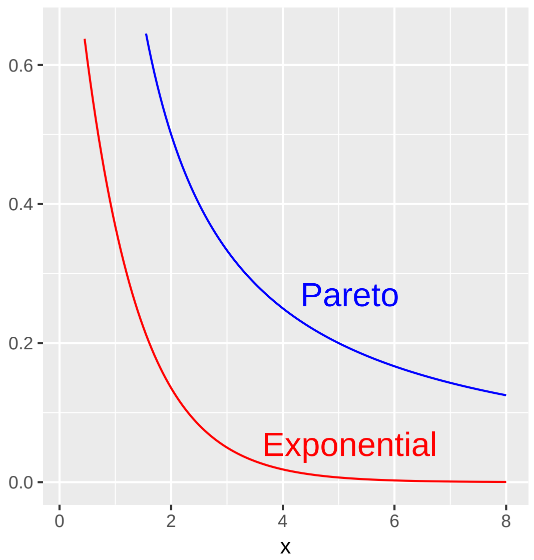


Figure 3.1: Exponential tail (red) and Pareto tail (blue)

generated by most of the people being quite tall, but nobody being a giant. In other words, you suppose that the group average is a reasonable approximation of the samples you will encounter. We call this phenomenon **conspiracy principle**. Many people slightly taller than usual, "conspire" together to generate a rather extreme overall height. Light-tailed distributions share this property. For example, height is usually modeled as a Gaussian random variable.

Let us consider now the Instagram followers. Without even seeing the five people, you will assume that there is a celebrity among them, having around ten million followers. There is only one cause of the unusually high number of followers. In this case, the sample average is not a good approximation of the samples you will encounter. We define this phenomenon **catastrophe principle**. The fundamental concept of the catastrophe principle is that the maximum observation in the group "behaves" as the total sum.

To summarise with formulas,

1. **Conspiracy principle**: if $X_1 + \cdots + X_p$ is large, it is probably due to *many* slightly bigger than average observations. Light-tailed distributions "obey" to this principle.

$$\underbrace{P(\max(X_1,\dots,X_p) > t)}_{\text{tail of the max observation}} = o[\underbrace{P(X_1,\dots,X_p > t)}_{\text{tail of the sum}}], \qquad \text{as } t \to \infty$$

2. **Catastrophe principle**: if $X_1 + \cdots + X_p$ is large, it is probably due to *one* unexpectedly large observation. Regularly varying distributions "obey" to this principle.

$$\underbrace{P(\max(X_1,\dots,X_p) > t)}_{\text{tail of the max observation}} \sim \underbrace{P(X_1,\dots,X_p > t)}_{\text{tail of the sum}}, \qquad \text{as } t \to \infty$$

The third property that differentiates light-tailed from regularly varying distributions is the **residual waiting time**. Suppose that you have already waited x minutes and you want to know what is the distribution of the *residual* waiting time. Let us define $R(t, x)$ the probability of waiting additional t minutes given that we have already waited x minutes.

$$
\begin{aligned}
R(t,x) := P[X > x+t |\ X > x] &= \frac{P[X > x+t, X > x]}{P[X > x]} \\
&= \underbrace{P[X > x |\ X > x+t]}_{=1} \frac{P[X > x+t]}{P[X > x]} \\
&= \frac{P[X > x+t]}{P[X > x]} = \frac{\overline{F}(x+t)}{\overline{F}(x)}
\end{aligned}
$$

where $\overline{F} = 1-F$ is the right-tail cumulative distribution function. If we model the residual waiting time with the Exponential distribution $\overline{F}(x) = e^{-\lambda x}$ (i.e., light-tailed), with $\lambda > 0$ and $x \geq 0$,

$$R(t,x) = e^{-\lambda(x+t)} \cdot e^{-\lambda x} = e^{-\lambda t}$$

which is independent of x. Hence we can say that the residual waiting time does not depend on how long we have already waited[1]. Suppose now that we model the residual waiting time with the Pareto distribution $\overline{F}(x) = c^{\alpha} x^{-\alpha}$ (i.e., regularly varying), with $c \geq 0$, $\alpha > 0$ and $x \geq c$. The distribution of the residual waiting time is:

$$R(t,x) = c^{\alpha}(x+t)^{-\alpha} \cdot c^{\alpha} x^{-\alpha} = (1 + \frac{t}{x})^{-\alpha}$$

which is increasing in x. The more we have waited, the higher the probability of waiting t additional minutes. A real-life example of this phenomenon is the waiting time for an email response: if someone does not answer in a short amount of time, it is quite likely that will not answer at all.

To summarise, we have seen three properties that distinguish light-tailed distributions from regularly varying distributions.

1. Scale invariance
2. Conspiracy and catastrophe principle
3. Memorylessness and increasing residual life

In the rest of the chapter, we reintroduce these concepts more formally.

3.2 Scale invariance and power-law

The presentation that follows is based on Nair et al. (2018). Let us start by defining the concept of scale invariance and power-law for a distribution.

---

[1] This is the well-known *memorylessness* property of the Exponential distribution.

Definition 3.2.0.1. *A distribution function F is* ***scale invariant*** *if there exists a $c \geq 0$ and a continuous function g such that*

$$\overline{F}(\lambda x) = g(\lambda)\ \overline{F}(x)$$

for all λ, x such that $\lambda x \geq c$, where $\overline{F} = 1 - F$.

From definition 3.2.0.1 we observe that the Pareto distribution is scale invariant. Recall that its right-tail cumulative distribution function is $\overline{F}(x) = c^{\alpha}x^{-\alpha}$, for $x \geq c$. Thus, $\overline{F}(x) = c^{\alpha}(\lambda x)^{-\alpha} = \lambda^{-\alpha}c^{\alpha}x^{-\alpha} = \lambda^{-\alpha}\overline{F}(x) = g(\lambda)\ \overline{F}(x)$, for $\lambda x \geq c$.

Definition 3.2.0.2. *A distribution function F is* ***power-law*** *if there exists a $c \geq 0$, $k \geq 0$ and $\alpha > 0$ such that*

$$\overline{F}(x) = kx^{-\alpha}$$

for $x \geq c$.

From definition 3.2.0.2 we see that the Pareto distribution is also power-law. If we set $k = c^{\alpha} \geq 0$, we obtain its right-tail cumulative distribution function $\overline{F}(x) = c^{\alpha}x^{-\alpha}$, for $x \geq c$.

Scale invariance, though nice, is a special property. Indeed, as the next proposition states, all scale invariant distributions have power-law tails, and thus only the Pareto distribution belongs to this class.

Proposition 3.2.0.3. *A distribution F is scale invariant if and only if $\overline{F}$ has a power-law tail.*

The proof can be found in Appendix A.3.

3.3 Approximately scale invariance and regularly varying distributions

We have seen that a distribution is scale invariant if and only if it has power-law tails. However, this class is extremely restricted, as it comprises only the Pareto distribution. For this reason, we relax the definition and look at all those distributions that are scale invariant only in the tails. We call them *asymptotically scale invariant*[2].

Definition 3.3.0.1. *A non-negative random variable X and its distribution F are said to be* ***asymptotically scale invariant*** *if there exists a strictly positive, finite, and continuous function g such that for any $\lambda > 0$,*

$$\lim_{x\to\infty} \frac{\overline{F}(\lambda x)}{\overline{F}(x)} = g(\lambda)$$

---

[2] An important distribution that belongs to this class is the Student's t.

As we introduced the concept of asymptotically scale invariant distributions, it seems reasonable to present those distributions that have *approximately* power-law tails. This leads to the following definitions.

Definition 3.3.0.2. *A positive (measurable) function f is called* ***regularly varying*** *(at infinity) with index $\alpha \in \mathbb{R}$, $f(x) \in RV(\alpha)$, if and only if:*

- *it is defined on some neighborhood of infinity $[x_0, \infty)$*
- $\lim_{x\to\infty} \frac{f(\lambda x)}{f(x)} = \lambda^{\alpha}$, *for $\lambda > 0$*

  If $\alpha = 0$, f is said to be ***slowly varying*** *(at infinity), $f(x) \in RV(0)$.*

From the definition 3.3.0.2, if $\alpha = 0$ we have a *slowly varying* function. These functions can be thought to behave almost as constant at infinity. Also, they are the basic building block to express any *regularly varying* function. The next result shows how to write any regularly varying function as a product of a slowly varying and a power-law function.

Proposition 3.3.0.3. *A function f is regularly varying with index $\alpha \in \mathbb{R}$, if and only if it can be written as*

$$f(x) = \ell(x)x^{\alpha}$$

where ℓ is a slowly varying function.

The proof can be found in Appendix A.4.

The result of proposition 3.3.0.3 highlights the fact that regularly varying functions can be thought, approximately, as power-law functions, as they differ from them only by ℓ (which can be treated as a "constant").

Given this setup, we are ready to apply the concept of regularly varying functions to random variables.

Definition 3.3.0.4. *A non-negative random variable X and its distribution F are said to be* ***regularly varying*** *with index $\alpha > 0$, if the right-tail cumulative distribution function $\overline{F}(x) = 1 - F(x)$ writes*

$$\overline{F}(x) = \ell(x)x^{-\alpha}, \qquad \text{as } x \to \infty$$

where $\ell \in RV(0)$ be a slowly varying function. In other words $\overline{F}(x) \in RV(-\alpha)$ as $x \to \infty$.

In the previous section, we saw that there is a one-to-one correspondence between scale invariant and power-law distributions. Therefore, we also expect that asymptotically scale invariant distributions are *approximately* power-law. This intuition is stated in the next result (see Nair et al. (2018), Chapter 3.2).

Proposition 3.3.0.5. *A distribution F is asymptotically scale invariant if and only if $\overline{F}$ is regularly varying.*

The proof can be found in Appendix A.5.

It is straightforward to integrate and differentiate the Pareto distribution because it is a power-law. It would be nice if we could integrate and differentiate regularly varying distributions as if they were power-laws. The next theorems 3.3.0.6 and 3.3.0.7 (for which we do not give a proof) allow us to do so.

Theorem 3.3.0.6 (Karamata's theorem)**.** *Supoose that $\ell \in RV(0)$ and that ℓ is bounded on every compact subsets of $[c,\infty)$ for some $c \geq 0$. Let $f(x) = \ell(x)x^\alpha$ be $RV(\alpha)$. Then*

(a) for $\alpha > -1$

$$\int_c^x f(u)\ du \sim \frac{x \cdot f(x)}{\alpha+1}, \qquad \text{as } x \to \infty$$

(b) for $\alpha < -1$

$$\int_x^\infty f(u)\ du \sim -\frac{x \cdot f(x)}{\alpha+1}, \qquad \text{as } x \to \infty$$

What the theorem says, intuitively, is the following. When $x \to \infty$, a regularly varying function f with index α behaves like a power function of degree α. For example, consider the function $f(x) = x^\alpha$. If we let $c = 0$ and $\alpha \neq -1$, we have:

(a) for $\alpha > -1$

$$\int_0^x f(u)\ du = \frac{x^{\alpha+1}}{\alpha+1} = \frac{x \cdot f(x)}{\alpha+1}, \qquad \text{as } x \to \infty$$

(b) for $\alpha < -1$

$$\int_x^\infty f(u)\ du = -\frac{x^{\alpha+1}}{\alpha+1} = -\frac{x \cdot f(x)}{\alpha+1}, \qquad \text{as } x \to \infty$$

Theorem 3.3.0.7 is the corresponding version of Karamata's theorem for differentiation (see Embrechts, Klüppelberg, and Mikosch (2013), Theorem A3.7).

Theorem 3.3.0.7 (Monotone density theorem)**.** *Let $U(x) = \int_0^x u(y)\ dy$ (or $\int_x^\infty u(y)\ dy$) where u is ultimately monotone (i.e., u is monotone on (z,∞) for some $z > 0$). If*

$$U(x) \sim c \cdot x^\alpha \ell(x), \qquad \text{as } x \to \infty$$

with $c \geq 0$, $\alpha \in \mathbb{R}$ and $\ell \in RV(0)$, then

$$u(x) \sim c \cdot \alpha x^{\alpha-1} \ell(x), \qquad \text{as } x \to \infty$$

For $c = 0$ the above relations are interpreted as $U(x) = o(x^\alpha \ell(x))$ and $u(x) = o(x^{\alpha-1}\ell(x))$.

3.4 Formalizing the catastrophe principle

In the previous sections, we have mentioned that the *catastrophe principle* is a quintessential property of regularly varying distributions. This property states that big sums of random variables tend to be driven by only one cause. In other words, the distribution of the maximum is equal to the distribution of the sum, when $x \to \infty$. The next two lemmas 3.4.0.1 and 3.4.0.2 formalize these ideas.

Lemma 3.4.0.1. *Let X and Y be two independent, regularly varying, non-negative random variables, with index $\alpha > 0$. Then $X + Y$ is regularly varying with index α and*

$$P(X + Y > u) \sim P(X > u) + P(Y > u), \qquad \textit{as } u \to \infty$$

The proof can be found in Appendix A.6.

Lemma 3.4.0.2. *Let X and Y be two independent, regularly varying, non-negative random variables, with index $\alpha > 0$. Then*

$$P(\max(X, Y) > u) \sim P(X + Y > u), \qquad \textit{as } u \to \infty$$

The proof can be found in Appendix A.7.

We can extend the result of lemma 3.4.0.1 via induction on n (see Embrechts et al. (2013), Section 1.3.1, Corollary 1.3.2)

Corollary 3.4.0.3. *Let $X, X_1, \ldots, X_n$ be iid non-negative regularly varying random variables with index $\alpha > 0$ and*

$$S_n = X_1 + \cdots + X_n, \qquad \textit{for } n \geq 1$$

Then

$$P(X_1 + \cdots + X_n > u) \sim n \; P(X > u), \qquad \textit{as } u \to \infty$$

Another consequence of lemma 3.4.0.1 is the following (see Mikosch (1999), Section 1.3.1, Corollary 1.3.8).

Corollary 3.4.0.4. *Let $X, X_1, \ldots, X_n$ be iid non-negative regularly varying random variables with index $\alpha > 0$ and $\psi_1, \ldots, \psi_n$ be non-negative constants. Then*

$$P(\psi_1 X_1 + \cdots + \psi_n X_n > u) \sim P(X > u)(\psi_1^\alpha + \cdots + \psi_n^\alpha), \qquad \textit{as } u \to \infty$$

These corollaries will be particularly useful in Chapter 4 when we develop an algorithm to learn the causal order of a linear SEM with regularly varying noise.

Chapter 4

Method

In this chapter, we present a method to learn the causal order from observational data, with the assumption that the data is heavy-tailed. In particular, we focus our attention on the subclass of regularly varying distributions.

4.1 Intuition

We say that X_1 causes X_2 if we observe the following:

- When X_1 takes extreme high values so does X_2
- When X_2 takes extreme high values, X_1 "behaves" normally

Pictorially, we see this happening in figure 4.1.

We can see that whenever X_1, on the x-axis, is large (i.e., beyond the red line), then X_2 is large too. However, when X_2, on the y-axis, is large (i.e., above the green line), then X_1 is not always large. In order to measure the causal effect in the extremes, we define the coefficient proposed by Engelke et al. (2018),

Definition 4.1.0.1. *Let X_1 and X_2 be two independent, regularly varying, non-negative random variables, with index $\alpha > 0$. Let F_i denote the cumulative distribution, for $i = 1, 2$. We define the coefficient:*

$$\Gamma_{12}(q) = \mathrm{E}[F_2(X_2)|\ F_1(X_1) > q], \qquad \textit{where } 0 < q < 1$$

4.2 Properties of Γ_{ij}

4.2.1 Setup

Before presenting the properties of Γ_{ij}, we lay down the setup that we will use for the rest of the chapter.

1. Let $X = (X_1, ..., X_p)$ be a random vector that follows a joint distribution P with density f. Let F_i be the marginal cumulative distribution of X_i, for $i = 1, ..., p$

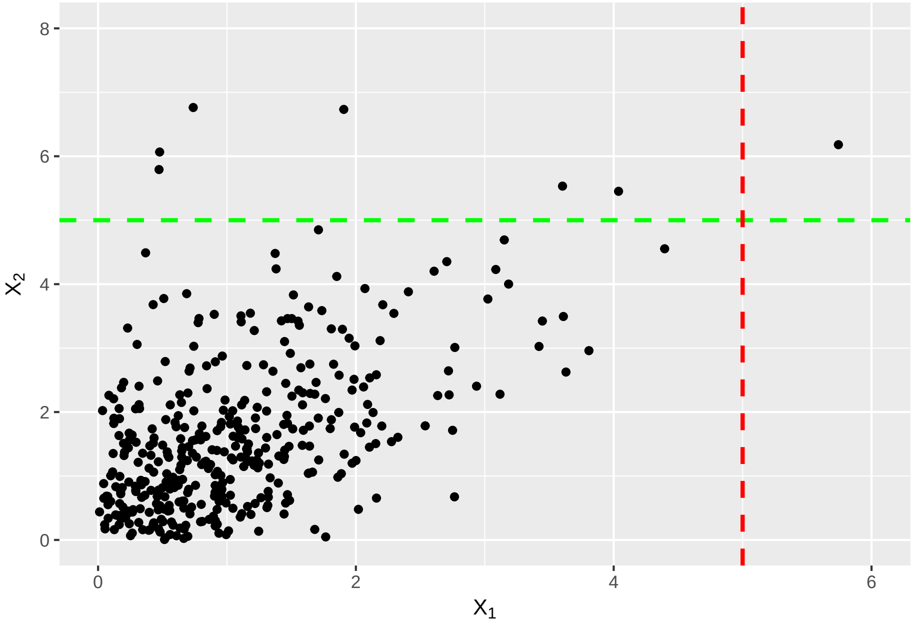


Figure 4.1: X_1 causes X_2

2. Let $G = (V, E)$ be a Directed Acyclic Graph (DAG), with $V = \{1, \dots, p\}$ and $E \subset V \times V$
3. Let (G, P) be a Causal Bayesian Network with related Structural Equation Model (SEM):
    a. $X_i \leftarrow \sum_{j \in pa(i)} \beta_{ij} X_j + Y_i$ with $\beta_{is} > 0$, $i, j \in V$
    b. Y_i are independent random noise on $[0, \infty)$ with regularly varying distribution, for $i \in V$. Their tails have the form $P(Y_i > u) \sim \ell(u) u^{-\alpha}$ as $u \to \infty$, where ℓ is a slowly varying function, and $\alpha > 0$ is the tail index

4.2.2 Results

Recall that F_1 and F_2 are regularly varying distributions. Therefore, from lemma 3.4.0.2 it holds,

$$P(\max(Y_1, Y_2) > u) \sim P(Y_1 + Y_2 > u), \qquad \text{as } u \to \infty$$

Under this assumption it makes sense to define for $i, j \in \{1, 2\}$ and $i \neq j$

$$p_{ij} = \lim_{u \to \infty} \frac{\overline{F_i}(u)}{\overline{F_i}(u) + \overline{F_j}(u)} \in [0, 1]$$

The following proposition and proof are taken from Engelke et al. (2018).

Proposition 4.2.2.1. *Let Y_1 and Y_2 be two independent random variables on $[0, \infty)$ with distribution F_1 and F_2 respectively. Assume further that $P(Y_i > u) \sim \ell(u) u^{-\alpha}$ as $u \to \infty$, $i = 1, 2$. Then:*

$$\lim_{u\to\infty} \mathrm{E}[F_1(Y_1)|\; Y_1+Y_2>u] = 1-\frac{1}{2}\; p_{21}$$

Proof. For u > 0 large enough, we have

$$\begin{aligned} P(F_1(Y_1)\le x, Y_1+Y_2>u) = &\; P(F_1(Y_1)\le x, Y_1+Y_2>u, Y_1>u) \\ &+ P(F_1(Y_1)\le x, Y_1+Y_2>u, Y_2>u) \\ &- P(F_1(Y_1)\le x, Y_1+Y_2>u, Y_1>u, Y_2>u) \\ &+ P(F_1(Y_1)\le x, Y_1+Y_2>u, Y_1<u, Y_2<u) \end{aligned} \tag{4.2.2.1}$$

We see that

- $P(F_1(Y_1)\le x, Y_1+Y_2>u, Y_1>u) \to 0$ as $u\to\infty$
- $P(F_1(Y_1)\le x, Y_1+Y_2>u, Y_2>u) = P(F_1(Y_1)\le x, Y_2>u)P(Y_1+Y_2>u|\; F_1(Y_1)\le x, Y_2>u) = P(F_1(Y_1)\le x, Y_2>u)\cdot 1 = P(F_1(Y_1)\le x)P(Y_2>u)$ due to independence
- $P(F_1(Y_1)\le x, Y_1+Y_2>u, Y_1>u, Y_2>u) \to 0$ as $u\to\infty$
- $P(F_1(Y_1)\le x, Y_1+Y_2>u, Y_1<u, Y_2<u) \le P(Y_1+Y_2>u, Y_1<u, Y_2<u) = P(Y_1+Y_2>u) - P(\max(Y_1,Y_2)>u) = o(\overline{F_1}(u)+\overline{F_2}(u))$ by lemma 3.4.0.2

Putting everything together and dividing equation (4.2.2.1) by $P(Y_1+Y_2>u)\sim P(Y_1>u)+P(Y_2>u)$ we obtain, for $0\le x<1$,

$$\lim_{u\to\infty} P(F_1(Y_1)\le x|\; Y_1+Y_2>u) = P(F_1(Y_1)\le x)\; p_{21}$$

Consequently, the random variable $F_1(Y_1)$, conditional on the event that $\{Y_1+Y_2>u\}$, converges weakly as $u\to\infty$ to a random variable with distribution function $B_{p_{21}}U+(1-B_{p_{21}})$, where U is a uniform distribution on $[0,1]$ and, independently, $B_{p_{21}}$ is Bernoulli with success probability p_2. By weak convergence we conclude

$$\lim_{u\to\infty} \mathrm{E}[F_1(Y_1)|\; Y_1+Y_2>u] = 1-\frac{1}{2}\; p_{21}$$

□

We extend the result above to the case of a linear SEM with p variables.

Proposition 4.2.2.2. *Let (G,P) be a causal Bayesian Network with related SEM as shown in 4.2.1. Let $i,j\in V$. Then it holds:*

$$\Gamma_{ji} = \lim_{u\to\infty} \mathrm{E}[F_i(X_i)\mid X_j>u] = 1-\frac{1}{2}p_{AB}$$

where

- $p_{AB} = \lim_{u\to\infty} \dfrac{P(Z_A>u)}{P(Z_A>u)+P(Z_B>u)}$

- $Z_A := h_1(Y_A)$, $Z_B := h_2(Y_B)$, *where* h_1 *and* h_2 *are monotonic increasing linear functions.*[1]
- $A := an(j) \cap an(i)^c$
- $B := an(j) \cap an(i)$

Proof. In a linear SEM, for monotonic increasing linear functions h, h_1, h_2, we can write

$$X_j \leftarrow h(Y_{an(j)}) = h_1(Y_{an(j)\cap an(i)^c}) + h_2(Y_{an(j)\cap an(i)}) = Z_A + Z_B$$

where the second equality holds due to linearity. For corollary 3.4.0.4, Z_A and Z_B are regularly varying random variables, hence we can write, for u large enough,

$$\begin{aligned} P(F_i(X_i) \leq x, X_j > u) = P(F_i(X_i) \leq x, Z_A + Z_B > u) = \\ = P(F_i(X_i) \leq x, Z_A + Z_B > u, Z_A > u) \\ + P(F_i(X_i) \leq x, Z_A + Z_B > u, Z_B > u) \\ - P(F_i(X_i) \leq x, Z_A + Z_B > u, Z_A > u, Z_B > u) \\ + P(F_i(X_i) \leq x, Z_A + Z_B > u, Z_A < u, Z_B < u) \end{aligned} \tag{4.2.2.2}$$

We see that

- $P(F_i(X_i) \leq x, Z_A + Z_B > u, Z_A > u) = P(F_i(X_i) \leq x, Z_A > u)P(Z_A + Z_B > u |\ F_i(X_i) \leq x, Z_A > u) = P(F_i(X_i) \leq x, Z_A > u) \cdot 1 = P(F_i(X_i) \leq x)P(Z_A > u)$ since we can write $X_i \leftarrow g(Y_{an(i)})$ which is independent of Z_A
- $P(F_i(X_i) \leq x, Z_A + Z_B > u, Z_B > u) \to 0$ as $u \to \infty$
- $P(F_i(X_i) \leq x, Z_A + Z_B > u, Z_B > u, Z_A > u) \to 0$ as $u \to \infty$
- $P(F_i(X_i) \leq x, Z_A + Z_B > u, Z_B < u, Z_A < u) \leq P(Z_A + Z_B > u, Z_B < u, Z_A < u) = P(Z_A + Z_B > u) - P(\max(Z_A, Z_B) > u) = o(P(Z_A > u) + P(Z_B > u))$ by lemma 3.4.0.2

Putting everything together and dividing equation (4.2.2.2) by $P(Z_A + Z_B > u) \sim P(Z_A > u) + P(Z_B > u)$ we obtain, for $0 \leq x < 1$,

$$\lim_{u\to\infty} P(F_i(X_i) \leq x |\ Z_A + Z_B > u) = P(F_i(X_i) \leq x)p_{AB}$$

and thus,

$$\lim_{u\to\infty} \mathrm{E}[F_i(X_i) \mid X_j > u] = 1 - \frac{1}{2}p_{AB}$$

□

---

[1] By abuse of notation, we write $Z_\emptyset = 0$ and $P(Z_\emptyset > u) = 0$, for $u > 0$.

4.3 Examples

In this section, we look at some examples of proposition 4.2.2.2, while we keep all the assumptions made in 4.2.1.

Example 4.3.0.1. Suppose we have the following SEM, where X_1 is independent of X_2:

$$\begin{aligned} X_1 &\leftarrow Y_1 \\ X_2 &\leftarrow Y_2 \end{aligned}$$

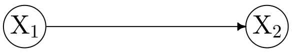
In this case, we see that $\Gamma_{12} = \Gamma_{21} = \frac{1}{2}$. This follows from the definition of Γ. Indeed,

$$\Gamma_{12} = \lim_{q \to 1} \mathrm{E}[F_2(X_2) |\ F_1(X_1) > q] = \mathrm{E}[F_2(X_2)] = \frac{1}{2}$$

where the second equality holds due to independence, and the third equality holds because $F_2(X_2) \sim U[0,1]$. Similarly, we observe that $\Gamma_{21} = \frac{1}{2}$.

Example 4.3.0.2. Suppose we have the following SEM, where X_1 causes X_2:

$$\begin{aligned} X_1 &\leftarrow Y_1 \\ X_2 &\leftarrow \beta_{21} X_1 + Y_2 \end{aligned}$$



We want to look at the causal relation between X_1 and X_2. The ancestors of node 1 are $an(1) = \{1\}$, the ancestors of node 2 are $an(2) = \{1, 2\}$.

To compute Γ_{12}, we define:

- $A := an(1) \cap an(2)^c = \emptyset$
- $B := an(1) \cap an(2) = \{1\}$
- $X_1 \leftarrow Z_A + Z_B = h_1(Y_A) + h_2(Y_B) = 0 + Y_1$

Putting everything together, we obtain:

$$\begin{aligned} p_{AB} &= \lim_{u \to \infty} \frac{P(Z_A > u)}{P(Z_A > u) + P(Z_B > u)} = \frac{0}{0 + P(Y_1 > u)} = 0 \\ \Gamma_{12} &= 1 - \frac{1}{2} p_{AB} = 1 \end{aligned}$$

To compute Γ_{21}, we define:

- $A := an(2) \cap an(1)^c = \{2\}$
- $B := an(2) \cap an(1) = \{1\}$
- $X_2 \leftarrow Z_A + Z_B = h_1(Y_A) + h_2(Y_B) = Y_2 + \beta_{21} Y_1$

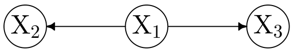
Putting everything together, we obtain:

$$
\begin{aligned}
p_{AB} &= \lim_{u \to \infty} \frac{P(Z_A > u)}{P(Z_A > u) + P(Z_B > u)} \\
&= \frac{\ell(u) u^{-\alpha}}{\ell(u) u^{-\alpha}(1 + \beta_{21}^{\alpha})} = \frac{1}{1 + \beta_{21}^{\alpha}} \in (0,1) \text{ as } \beta_{21} > 0 \\
\Gamma_{21} &= 1 - \frac{1}{2} p_{AB} < 1
\end{aligned}
$$

From this example we see that the Γ coefficient is asymmetric, in the sense that if X_1 causes X_2, then $\Gamma_{12} = 1 > \Gamma_{21}$.

Example 4.3.0.3. Suppose we have the following linear SEM, where X_1 causes both X_2 and X_3. In this setup, X_2 and X_3 have a positive correlation even if none of them causes the other one. It is the classic case of confounding, where a common (hidden) variable causes two or more variables. We will show that the Γ coefficient can detect the confounding phenomenon in this simple setting.

$$
\begin{aligned}
X_1 &\leftarrow Y_1 \\
X_2 &\leftarrow \beta_{21} X_1 + Y_2 \\
X_3 &\leftarrow \beta_{31} X_1 + Y_3
\end{aligned}
$$



We want to look at the causal relationship between X_2 and X_3, keeping X_1 unobserved. The ancestors of node 2 are $an(2) = \{2, 1\}$, the ancestors of node 3 are $an(3) = \{3, 1\}$.

To compute Γ_{23}, we define:

- $A := an(2) \cap an(3)^c = \{2\}$
- $B := an(2) \cap an(3) = \{1\}$
- $X_2 \leftarrow Z_A + Z_B = h_1(Y_A) + h_2(Y_B) = Y_2 + \beta_{21} Y_1$

Putting everything together, we obtain:

$$
\begin{aligned}
p_{AB} &= \lim_{u \to \infty} \frac{P(Y_2 > u)}{P(Y_2 > u) + P(\beta_{21} Y_1 > u)} \\
&= \frac{\ell(u) u^{-\alpha}}{\ell(u) u^{-\alpha}(1 + \beta_{21}^{\alpha})} = \frac{1}{1 + \beta_{21}^{\alpha}} \in (0,1) \text{ as } \beta_{21} > 0 \\
\Gamma_{23} &= 1 - \frac{1}{2} p_{AB} < 1
\end{aligned}
$$

To compute Γ_{32}, we define:

- $A := an(3) \cap an(2)^c = \{3\}$
- $B := an(3) \cap an(2) = \{1\}$
- $X_3 \leftarrow Z_A + Z_B = h_1(Y_A) + h_2(Y_B) = Y_3 + \beta_{31} Y_1$

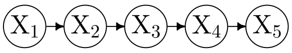
Hence, we obtain:

$$\begin{aligned} p_{AB} &= \frac{1}{1+\beta_{31}^{\alpha}} \in (0,1) \text{ as } \beta_{31} > 0 \\ \Gamma_{32} &= 1 - \frac{1}{2} p_{AB} < 1 \end{aligned}$$

In this case, we see that neither Γ_{23} nor Γ_{32} are equal to 1, indicating that X_2 and X_3 are not causally related (even if they are correlated).

Example 4.3.0.4. Consider the following DAG, a chain with five nodes. In this case the SEM writes:

$$\begin{aligned} X_1 &\leftarrow Y_1 \\ X_2 &\leftarrow \beta_{21} X_1 + Y_2 \\ X_3 &\leftarrow \beta_{32} X_2 + Y_3 \\ X_4 &\leftarrow \beta_{43} X_3 + Y_4 \\ X_5 &\leftarrow \beta_{54} X_4 + Y_5 \end{aligned}$$



We show two properties of the Γ coefficient. In particular:

i.) If $i \in an(j)$ then $\Gamma_{ij} = 1$

ii.) If $i \in an(j)$ then $\Gamma_{ki} \leq \Gamma_{kj}$, for $k \in V \setminus \{i, j\}$

Consider case i.). Let $i = 1$. For $j = 2, \ldots, 5$, we have:

- $A := an(1) \cap an(j)^c = \emptyset$
- $B := an(1) \cap an(j) = an(1) = \{1\}$
- $X_1 \leftarrow Z_A + Z_B = 0 + Y_1$

Then, it follows that $p_{AB} = 0$, and so $\Gamma_{1j} = 1$, for $j = 2, \ldots, 5$.

Consider case ii.). We start with a trivial case, where $k = 1$, $i = 2$, $j = 3$. In this case, from case i.), we have that $\Gamma_{12} = \Gamma_{13} = 1$. Therefore case ii.) holds, too.
Now, let $i = 1$, $j = 2$, $k = 3$. We want to compare Γ_{31} with Γ_{32}. For Γ_{31} we define:

- $A := an(3) \cap an(1)^c = \{2, 3\}$
- $B := an(3) \cap an(1) = \{1\}$

- $X_3 \leftarrow Z_A + Z_B = (Y_3 + \beta_{32} Y_2) + (\beta_{32}\beta_{21} Y_1)$

Then, it follows:

$$p_{AB} = \frac{1 + \beta_{32}^{\alpha}}{1 + \beta_{32}^{\alpha} + (\beta_{32}\beta_{21})^{\alpha}}$$

For Γ_{32} we define:

- $A := an(3) \cap an(2)^c = \{3\}$
- $B := an(3) \cap an(2) = \{1, 2\}$
- $X_3 \leftarrow Z_A + Z_B = (Y_3) + (\beta_{32} Y_2 + \beta_{32}\beta_{21} Y_1)$

Then, it follows:

$$p^*_{AB} = \frac{1}{1 + \beta_{32}^{\alpha} + (\beta_{32}\beta_{21})^{\alpha}}$$

We can see that $p_{AB} \geq p^*_{AB}$ and thus, $\Gamma_{31} \leq \Gamma_{32}$.

We can generalize case i.) and ii.) in the following lemma.

Lemma 4.3.0.5. *Let (G, P) be a causal Bayesian Network with related SEM as shown in 4.2.1. Let $i, j, k \in V$ with $i \in an(j)$. Then, the following holds:*

i.) $\Gamma_{ij} = 1$

ii.) $\Gamma_{ki} \leq \Gamma_{kj}$

Proof. i.) Since i is ancestor of j, it holds that $an(i) \subset an(j)$. This implies that the intersection between $an(i)$ and $an(j)^c$ is empty. If we define:

- $A := an(i) \cap an(j)^c = \emptyset$
- $B := an(i) \cap an(j) = an(i)$

we observe that $p_{AB} = 0$ and thus $\Gamma_{ij} = 1$.

ii.) Since $i \in an(j)$ it holds that $an(i) \subset an(j)$. Equivalently, we can write $an(i)^c \supset an(j)^c$. Hence it follows that

$$\begin{aligned} A &:= an(k) \cap an(i)^c \supset an(k) \cap an(j)^c =: A^* \\ B &:= an(k) \cap an(i) \subset an(k) \cap an(j) := B^* \end{aligned}$$

Therefore,

$$\begin{aligned}\Gamma_{ki} &= 1-\frac{1}{2}p_{AB} = 1-\frac{1}{2}\lim_{u\to\infty}\frac{P(Z_A>u)}{P(Z_{an(k)}>u)} = 1-\frac{1}{2}\lim_{u\to\infty}\frac{P(Z_{A^*}>u)+P(Z_{A\setminus A^*}>u)}{P(Z_{an(k)}>u)}\\ &\leq 1-\frac{1}{2}\lim_{u\to\infty}\frac{P(Z_{A^*}>u)}{P(Z_{an(k)}>u)} = 1-\frac{1}{2}p_{A^*B^*} = \Gamma_{kj}\end{aligned}$$

where the third equality holds because Z_A is a linear combination of regular varying random variables, with positive coefficients, and hence corollary 3.4.0.4 applies. ☐

4.4 Structure Learning

We want to find an algorithm to discover the topological order of a causal graph, given a set of observations generated by an SEM, as described in 4.2.1. First, we recall the main facts that we discovered in the previous section. Namely, given a causal Bayesian network (G, P), if the node i has no parents (also called *root* node) we have

$$\Gamma_{ij} = \begin{cases} 1, & \text{if } j \in desc(i) \\ \frac{1}{2}, & \text{if } j \notin desc(i) \end{cases}$$
$$\Gamma_{ji} = \begin{cases} [\frac{1}{2}, 1), & \text{if } j \in desc(i) \\ \frac{1}{2}, & \text{if } j \notin desc(i) \end{cases}$$

From this facts, we see that if i is a *root* node, the quantity $\Delta_{ij} := \Gamma_{ij} - \Gamma_{ji} \geq 0$ for $j \in V \setminus \{i\}$. Intuitively, we can expect that a *source* node i (not necessarily unique), maximizes the quantity $S_i = \sum_{j\neq i}\Delta_{ij}$, in the population case. The following lemma establishes this result and gives us a tool to detect a *root* node from a set of observed variables.

Lemma 4.4.0.1. *Let (G, P) be a Bayesian Network with related SEM, as defined in 4.2.1. Suppose that P satisfies the global Markov property and that there are no hidden confounders in the SEM. Define*

$$S_i := \sum_{j\neq i}\Delta_{ij}, \quad \forall i, j \in V$$

where $\Delta_{ij} := \Gamma_{ij} - \Gamma_{ji}$. Let $k \in V$. If $k \in \arg\max_i S_i$, then k is a root node.

Proof. Suppose $k \in \arg\max_i S_i$ and that k is not a root node. This means that $\{k\} \subset an(k)$. Since G is a DAG, there exists a node $m \in an(k)$ such that $pa(m) = \emptyset$. Since $m \in an(k)$, it holds that $desc(k) \subset desc(m)$. Hence, we can write $desc(m) := \{m\} \cup desc(k) \cup A$ for some (possibly empty) set A. Define $N := non\text{-}desc(m)$. Since m is d-separated from all nodes $j \in N$, using the Markov assumption, we know that $X_m \perp\!\!\!\perp X_N$. In other words, $\Gamma_{mj} = \Gamma_{jm} = \frac{1}{2}$ for $j \in N$. At the same time, it holds that $\Gamma_{kj} - \Gamma_{jk} = 0$ for the nodes $j \in N$ that are d-separated from k, and $\Gamma_{kj} - \Gamma_{jk} < 0$ for the nodes $j \in N$ that are ancestors of k. It follows that $\sum_{j\in N}(\Gamma_{kj} - \Gamma_{jk}) \leq 0$.

Let us decompose $V = desc(m) \cup non\text{-}desc(m) := \{m\} \cup desc(k) \cup A \cup N$. Thus,

$$\begin{aligned}
S_k &= \sum_{j \neq k} \Delta_{kj} = \sum_{j \neq k} (\Gamma_{kj} - \Gamma_{jk}) \\
&= \sum_{j \in desc(m) \setminus \{k\}} (\Gamma_{kj} - \Gamma_{jk}) + \sum_{j \in non\text{-}desc(m)} (\Gamma_{kj} - \Gamma_{jk}) \\
&= (\Gamma_{km} - \Gamma_{mk}) + \sum_{j \in desc(k) \setminus \{k\}} (\Gamma_{kj} - \Gamma_{jk}) + \sum_{j \in A} (\Gamma_{kj} - \Gamma_{jk}) + \sum_{j \in N} (\Gamma_{kj} - \Gamma_{jk}) \\
&\leq (\Gamma_{km} - \Gamma_{mk}) + \sum_{j \in desc(k) \setminus \{k\}} (1 - \Gamma_{jk}) + \sum_{j \in A} (\Gamma_{kj} - \Gamma_{jk}) \\
&< (\Gamma_{mk} - \Gamma_{km}) + \sum_{j \in desc(k) \setminus \{k\}} (1 - \Gamma_{jm}) + \sum_{j \in A} (1 - \Gamma_{jm}) \\
&= \sum_{j \neq m} \Delta_{mj} = S_m
\end{aligned}$$

where the inequality holds because $\Gamma_{mk} = 1 > \Gamma_{km}$ and $\Gamma_{jm} \leq \Gamma_{jk}$ for $j \in V \setminus \{m, k\}$, according to lemma 4.3.0.5.

But this is a contradiction, since we assumed that $k \in \arg\max_i S_i$. □

The result of lemma 4.4.0.1 tells us that, given a set of variables satisfying the assumptions 4.2.1, it is possible to identify one variable with no parents, when $n \to \infty$. Intuitively, we could identify one variable as the *root* node, remove its effect from the remaining ones, and repeat the procedure until no variables are left. In this way, we would be able to construct a topological order for the set of variables. Hence, this procedure would involve two steps:

1. Identify the *root* node
2. Remove impact of *root* node from the remaining nodes

Regarding point 1., we define first an estimator for Γ, as proposed by Engelke et al. (2018),

Definition 4.4.0.2 (A non-parametric estimator). *Let X_1 and X_2 be two heavy-tailed random variables. Let (X_{i1}, X_{i2}) be an independent observation of (X_1, X_2) for $i = 1, \dots, n$. Let R^n_{ij} be the rank of X_{ij} among the $X_{1j}, \dots, X_{nj}$ for $j = 1, 2$. Replacing F_1 and F_2 in definition 4.1.0.1 by the empirical counterparts, and the threshold u by $u_n = 1 - c/n$ for some integer $1 \leq c \leq n$, we obtain*

$$\hat{\Gamma}_{12}(u_n) = \frac{1}{c} \sum_{i=1}^{n} \frac{R_{i2}}{n+1} \mathbb{1}\{R_{i1} > n + 1/2 - c\}$$

For this estimator to be consistent, a classical assumption in extreme value theory is that $c = c_n$ depends on the sample size n such that $c_n \to \infty$ and $c_n/n \to 0$ as $n \to \infty$. In our case, we choose $c_n = \sqrt{n}/2$. The estimator $\hat{\Gamma}_{21}(u_n)$ is defined analogously.

Using the estimated coefficients $\hat{\Gamma}_{ij}$, we can identify a *root* node by computing $k = \arg\max_i \hat{S}_i = \arg\max_i \sum_{j \neq i} \hat{\Delta}_{ij}$, for $i, j \in V$. Once we have such maximizer, we want to remove its effect from the other variables to be able to use result of lemma 4.4.0.1

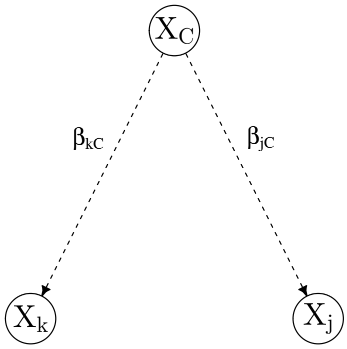


Figure 4.2: Causal Bayesian Network with a confounder X_C

recursively. We present here 4 competing algorithms that are based, to some extent, on points 1. and 2. above. We name the algorithms *Drop*, *Regress*, *Remove* and *Fast*. All four algorithms have in common the first step, namely finding the root node. To do that, we estimate the matrix $\hat{\Gamma}$ from a dataset, we compute the scores $\hat{S}_i = \sum_{j \neq i} \hat{\Delta}_{ij}$, for all $i \in V$, and choose the variable $k = \arg\max_i \hat{S}_i$. The four algorithms differ regarding the second point, i.e., how to remove the effect of the root nodes. **Fast** algorithm does not remove any node from the dataset. It computes $\hat{S}_i$, for all $i \in V$, and builds the topological order by ranking the variables by $\hat{S}_i$, in decreasing order. **Remove** algorithm iteratively removes the root nodes from the dataset, which is equivalent to recompute at each iteration $\hat{S}_i$ considering only the remaining variables in the graph. **Regress** algorithm regresses out the impact of the root node from the other variables.[2] For example, suppose we have p variables $X_1, \ldots, X_p$, and we have identified X_k as the source node. We can transform the remaining variables in the following way:

$$\widetilde{X}_i \leftarrow X_i - \hat{\beta}_{ik} X_k, \quad i \in V \setminus \{k\}$$

where $\hat{\beta}_{ik}$ is the ordinary least squares (OLS) estimated coefficient of the *total* effect of X_k on X_i. This method has two shortcomings. First, it applies only to linear SEMs. This problem, however, is easily solved if we implement non-linear regression techniques (e.g., Generalized Additive Model). The second issue, which is more problematic, arises in the case of a confounder. Consider, for example, the Causal Bayesian Network shown in 4.2 where the variable X_C is hidden. In this case, we see that the direct effect of X_k on X_j is zero since X_k does not cause X_j. However, the OLS estimated coefficient for $X_j = \hat{\beta}_{jk} X_k + \hat{\varepsilon}_j$ is biased and in particular $\mathrm{E}[\hat{\beta}_{jk}] \propto \beta_{kC} \cdot \beta_{jC} \neq 0$.

To avoid this issue, we propose the algorithm named **Drop**. The idea behind the algorithm relies on the definition of the estimator for Γ, 4.4.0.2. In particular, we note that $\hat{\Gamma}$ depends only on the c_n greatest observations, where $c_n = \sqrt{n}/2$. Assuming that X_k causes its descendant through extremes observations, by removing the c_n-greatest observations from the data set, we should be able to filter out the effect of the root node from the other variables. Using an example, suppose that we identify X_k as a *source* node. Let n be the number of observations and $c_n = \sqrt{n}/2$ the order statistic. Denote the c_n^{th}-greatest observation of X_k as $X_{c_n,k}$. We remove from the dataset the variable X_k and all the rows where $X_k \geq X_{c_n,k}$.

---

[2] We credit the idea of regressing out the impact of the root node to Shimizu et al. (2011).

Below, we present the pseudocode of the four algorithms.

Algorithm 1 *Drop*

```
function DROP ALGORITHM($X$)
    ▷ Input: data set $X \in \mathbb{R}^{n \times p}$
    ▷ Output: vector representing a topological order $\pi = [\pi(1), \dots, \pi(p)]$
    $V \leftarrow [1, \dots, p]$
    $X_0 \leftarrow X$
    $V_0 \leftarrow V$
    $\pi \leftarrow [\ ]$
    for iter $\in \{1, \dots, p-1\}$ do
        ▷ find root node
        $\Gamma_{ij} \leftarrow$ compute gamma($X_0[, i], X_0[, j]$) for $i, j \in V_0$, $i \neq j$
        $\Delta_{ij} \leftarrow \Gamma_{ij} - \Gamma_{ji}$
        $S_i \leftarrow \sum_{j \neq i} \Delta_{ij}$ for $i \in V_0$
        $root \leftarrow \arg\max_i S_i$
        ▷ remove impact of the root node
        $X_0 \leftarrow$ remove extreme observations($X_0$)
        $X_0 \leftarrow$ remove $root$ variable($X_0$)
        $V_0 \leftarrow$ remove $root$ node($V_0$)
        $\pi \leftarrow$ update topological order($\pi, root$)
    end for
    $root \leftarrow V_0[1]$
    $\pi \leftarrow$ update topological order($\pi, root$)
    return $\pi$
end function
```

Algorithm 2 *Regress*

```
function REGRESS ALGORITHM($X$)
    ▷ Input: data set $X \in \mathbb{R}^{n \times p}$
    ▷ Output: vector representing a topological order $\pi = [\pi(1), \dots, \pi(p)]$
    $V \leftarrow [1, \dots, p]$
    $X_0 \leftarrow X$
    $V_0 \leftarrow V$
    $\pi \leftarrow [\ ]$
    for iter $\in \{1, \dots, p-1\}$ do
        ▷ find root node
        $\Gamma_{ij} \leftarrow$ compute gamma($X_0[,i], X_0[,j]$) for $i, j \in V_0$, $i \neq j$
        $\Delta_{ij} \leftarrow \Gamma_{ij} - \Gamma_{ji}$
        $S_i \leftarrow \sum_{j \neq i} \Delta_{ij}$ for $i \in V_0$
        $root \leftarrow \arg\max_i S_i$
        ▷ remove impact of the root node
        $X_0 \leftarrow$ regress out all variables vs $root(X_0)$
        $X_0 \leftarrow$ remove $root$ variable($X_0$)
        $V_0 \leftarrow$ remove $root$ node($V_0$)
        $\pi \leftarrow$ update topological order($\pi, root$)
    end for
    $root \leftarrow V_0[1]$
    $\pi \leftarrow$ update topological order($\pi, root$)
    return $\pi$
end function
```

Algorithm 3 *Remove*

```
function REMOVE ALGORITHM($X$)
    ▷ Input: data set $X \in \mathbb{R}^{n \times p}$
    ▷ Output: vector representing a topological order $\pi = [\pi(1), \dots, \pi(p)]$
    $V \leftarrow [1, \dots, p]$
    $X_0 \leftarrow X$
    $V_0 \leftarrow V$
    $\pi \leftarrow [\ ]$
    for iter $\in \{1, \dots, p-1\}$ do
        ▷ find root node
        $\Gamma_{ij} \leftarrow$ compute gamma($X_0[,i], X_0[,j]$) for $i, j \in V_0$, $i \neq j$
        $\Delta_{ij} \leftarrow \Gamma_{ij} - \Gamma_{ji}$
        $S_i \leftarrow \sum_{j \neq i} \Delta_{ij}$ for $i \in V_0$
        $root \leftarrow \arg\max_i S_i$
        ▷ remove impact of the root node
        $X_0 \leftarrow$ remove $root$ variable($X_0$)
        $V_0 \leftarrow$ remove $root$ node($V_0$)
        $\pi \leftarrow$ update topological order($\pi, root$)
    end for
    $root \leftarrow V_0[1]$
    $\pi \leftarrow$ update topological order($\pi, root$)
    return $\pi$
end function
```

Algorithm 4 *Fast*

```
function FAST ALGORITHM($X$)
    ▷ Input: data set $X \in \mathbb{R}^{n \times p}$
    ▷ Output: vector representing a topological order $\pi = [\pi(1), \ldots, \pi(p)]$
    $V \leftarrow [1, \ldots, p]$
    ▷ compute scores $S_i$
    $\Gamma_{ij} \leftarrow$ compute gamma($X[, i], X[, j]$) for $i, j \in V$, $i \neq j$
    $\Delta_{ij} \leftarrow \Gamma_{ij} - \Gamma_{ji}$
    $S_i \leftarrow \sum_{j \neq i} \Delta_{ij}$ for $i \in V$
    $S \leftarrow [S_1, \ldots, S_p]$
    ▷ compute topological order based on $S$
    $\tilde{S} \leftarrow$ sort in decreasing order($S$)
    $\pi \leftarrow$ extract indices($\tilde{S}$)
    return $\pi$
end function
```

Chapter 5

Simulations

In this chapter, we present some simulations to assess the performance of the four algorithms introduced in Chapter 4.

5.1 Learning a Topological Order

The goal is to estimate a topological order from a heavy-tailed dataset (in particular with regularly varying tails). We test the algorithms on different linear SEMs, assuming t-Student distributed noise and a grid of three parameters: sample size n, tail-index α and the average magnitude of the SEM coefficients b. For each combination (n, α, b) we assess the performance of the four methods (i.e., *Drop*, *Regress*, *Remove* and *Fast* algorithm) over 200 simulations. We define two metrics to measure the performance,

- Q-performance
- Topological matrix

5.1.1 Q-performance

Suppose we have a Causal Bayesian Network with graph $G = (V, E)$, where $V \in \{1, \dots, p\}$ and $E \subset V \times V$. Let $\Pi_G = \Big\{(\pi(1), \dots, \pi(p)) : \;\; i \in V, \quad i \in an(j) \implies \pi(i) < \pi(j)\Big\}$ be the collection of valid topological orders of G. Given an estimated topological order $\hat{\pi} = [\hat{\pi}(1), \dots, \hat{\pi}(p)]$ we can verify whether $\hat{\pi}$ belongs to the "true" collection Π_G. If we run m-simulations on a fixed combination of (n, α, b), we obtain $\hat{\pi}_1, \dots, \hat{\pi}_m$. We define:

$$Q = \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}[\hat{\pi}_i \in \Pi_G]$$

as the proportion of correctly estimated topological sorts.

5.1.2 Topological matrix

Consider a Causal Bayesian Network with graph $G = (V, E)$, where $V \in \{1, \dots, p\}$ and $E \in V \times V$. We define the *true topological matrix* $M \in \{0, 1\}^{p \times p}$ such that

$$M_{ij} = \begin{cases} 1, & \text{if } \exists \pi \in \Pi_G \text{ s.t. } \pi(i) < \pi(j) \\ 0, & \text{else} \end{cases}$$

for $i, j \in V$. Similarly, we can estimate a topological matrix from a dataset in the following way:

1. Estimate a topological order $\hat{\pi} = [\hat{\pi}(1), \dots, \hat{\pi}(p)]$
2. Define $\hat{M} \in \{0,1\}^{p \times p}$, where,

$$\hat{M}_{ij} = \begin{cases} 1, & \text{if } \hat{\pi}(i) < \hat{\pi}(j) \\ 0, & \text{else} \end{cases}$$

for $i, j \in V$. Over m-simulations, we can compute an "average" topological matrix $\overline{M}_{ij} = \frac{1}{m}\sum_{\ell=1}^{m} \hat{M}_{ij}^{\ell}$, for $i, j \in V$. Each entry $\overline{M}_{ij}$ tells how many times (in percentage) variable X_i comes before variable X_j in the estimated topological sorts. Comparing M with $\overline{M}$ is a useful visualization aid to assess the strengths and weaknesses of the algorithms.

In the following, we present the results for different simulations. For each linear SEM, we consider a grid of three parameters. The parameters are:

- sample size: $n \in \{100, 1000, 10000\}$
- tail-index: $\alpha \in \{1, \dots, 5\}$
- average magnitude of SEM coefficients: $b \in \{0.5, 1, 1.5\}$, where $\beta's \sim U[0, 2b]$

We assume that the noise is distributed as a t-Student, where the degrees of freedom correspond to the tail-index α. For each combination of (n, α, b) we simulate 200 datasets, and we compare the Q-performance of the four algorithms. Besides, we compare the "true" topological matrix M with the estimated one $\overline{M}$ for some configurations of (n, α, b).

5.1.3 Simulation 1 - Simple chain

Consider the Causal Bayesian Network depicted in figure 5.1. We assume a linear SEM with related DAG $G = (V, E)$, where $V = \{1, \dots, 5\}$ and $E = \{(1,2), (2,3), (3,4), (4,5)\}$.

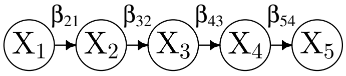


Figure 5.1: Causal Bayesian Network - Simple chain

Each entry in tables 5.1, 5.2, 5.3, 5.4 represent the Q-performance over 200 simulations for a particular algorithm and parameter configuration (n, α, b). A score of zero means that all estimated causal orders are wrong. A score of one means that all estimated causal orders are correct. To give a benchmark performance, note that for this graph there is only one valid topological order, that is $\pi = (1, 2, 3, 4, 5)$. At the same time, since the graph has five nodes, there are $5! = 120$ possible ways to sort the vertices. Hence, on average, a random algorithm would guess the correct order one out of 120 times.

Looking at tables 5.1, 5.2, 5.3, 5.4, we can observe that all the four methods perform in a similar way when the sample size is large ($n = 10,000$), and the tails of the noise distribution are heavy ($\alpha \approx 1, 2$). For smaller values of n and higher values of α, *Fast* method is the one that performs worst in most cases. Consider figure 5.5, with the true topological matrix and the estimated ones, for $n = 1,000, \alpha = 1, b = 1$. Recall that the entries (i, j) of the estimated matrices count how many times (in percentage) $\hat{\pi}(i) < \hat{\pi}(j)$ over 200 simulations, where $i, j \in V$. For example, in *Drop* and *Remove* algorithm $\hat{\pi}(5) < \hat{\pi}(4)$ only 4% of the times. On the other hand, *Fast* algorithm makes more mistakes since $\hat{\pi}(5) < \hat{\pi}(4)$ in 12% of the simulations. It seems that *Fast* algorithm tends to make mistakes for the variables that come "late" in the correct topological order (in this case X_4 and X_5). To understand why this happens, recall that *Fast* method does not remove the causal impact of the ancestors from its descendants. More precisely, one can show that:

- $\Gamma_{45} = 1$, since X_4 causes $X_5 \leftarrow \beta_{54} X_4 + Y_5$
- $\Gamma_{54} \approx 1$ (by using proposition 4.2.2.2)

Hence, both coefficients are very similar $\Gamma_{45} \approx \Gamma_{54}$ and it is difficult for *Fast* method to detect the causal order between X_4 and X_5. In contrast, the other three methods (*Drop*, *Regress* and *Remove*) iteratively remove the impact of the root nodes from the other vertices. This allows estimating more precisely the causal order of the variables that come "late" with respect to π. For example, if we remove the causal effect of X_1, X_2, X_3 from X_4 and X_5, we obtain $\widetilde{X}_4 \leftarrow Y_4$ and $\widetilde{X}_5 \leftarrow \beta_{54} \widetilde{X}_4 + Y_5$. Using proposition 4.2.2.2 on the transformed variables $\widetilde{X}_4$ and $\widetilde{X}_5$ we obtain $\Gamma_{45} = 1 > \Gamma_{54} = 1 - \frac{1}{2} \frac{1}{1+\beta_{54}^{\alpha}}$. Therefore, it is easier to identify the causal direction between nodes 4 and 5.

5.1.4 Simulation 2 - Polytree

Consider the Causal Bayesian Network shown in figure 5.2. We assume a linear SEM with DAG $G = (V, E)$, where $V = \{1, \ldots, 6\}$ and $E = \{(1, 3), (2, 3), (2, 4), (3, 5), (5, 6)\}$.

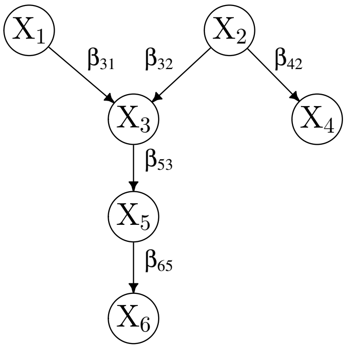


Figure 5.2: Causal Bayesian Network - Polytree

Tables 5.5, 5.6, 5.7, 5.8 show the Q-performance of the four algorithms. In this case, a random algorithm would guess the correct order one time out of 80 trials, on average. In fact, there are nine valid topological orders for the graph and $6! = 720$ ways to permute six nodes. As it happens in Simulation 1, the four algorithms perform almost identically for large sample sizes ($n = 10,000$) and heavy-tailed noise ($\alpha \approx 1$). In the other cases, the algorithms *Drop*, *Regress* and *Remove* outperform *Fast* method.

Looking at a specific set of parameters, $n = 10000$, $\alpha = 5$ and $b = 2$, we observe again that *Fast* algorithm tends to make more mistakes for the variables that come "late" in the true causal order. As we can see in matrix 5.6, *Fast* algorithm incorrectly estimates $\hat{\pi}(6) < \hat{\pi}(5)$ 40% of the times, $\hat{\pi}(6) < \hat{\pi}(3)$ 23% of the times and $\hat{\pi}(5) < \hat{\pi}(3)$ 30% of the times. In comparison, the error rates for the other algorithms are 11% for $\hat{\pi}(6) < \hat{\pi}(5)$, 4% for $\hat{\pi}(6) < \hat{\pi}(3)$ and 9% for $\hat{\pi}(5) < \hat{\pi}(3)$, approximately.

5.1.5 Simulation 3 - Non-Polytree

Consider a slightly more complex Causal Bayesian Network, depicted in figure 5.3. There are five valid causal orders out of 720 possible permutations of the nodes. A random algorithm, on average, would guess the correct order less than seven times out of 1000 trials.

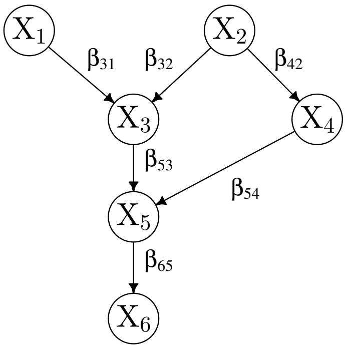


Figure 5.3: Causal Bayesian Network - Non-Polytree

As already shown in the previous simulations, *Fast* algorithm's Q-performance is lower compared to the other methods especially when $n = 1,000$ and $\alpha \approx 4, 5$ (see tables 5.9, 5.10, 5.11, 5.12). Looking at the topological matrices 5.7, we note that *Drop*, *Regress* and *Remove* algorithm perform almost equally, for $n = 1000$, $\alpha = 1$ and $b = 3$. As before, *Fast* algorithm has a higher error rate for variables X_3, X_5, X_6 compared to the other algorithms. However, it is interesting to note that for some pairs of variables $((X_5, X_1), (X_6, X_1), (X_6, X_4))$ it makes no mistakes, while *Drop*, *Regress* and *Remove* algorithm do.

5.1.6 Simulation 4 - Graph with confounders

In this last simulation, we make a step further and test the four algorithms on a Causal Bayesian Network with two confounders (figure 5.4).

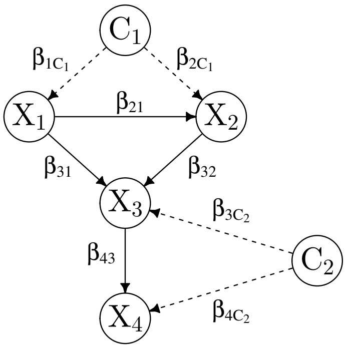


Figure 5.4: Causal Bayesian Network - Graph with confounders

Even if lemma 4.4.0.1 does not cover the case of confounding variables, we think it is useful to study empirically how the methods perform in this setting. In this case, there is only one valid topological sort, $\pi = (1, 2, 3, 4)$, and 24 possible permutations of the four observed variables. Thus, a random algorithm would make a correct guess, on average, less than 5% of the times. Even in the presence of confounding variables, when $n = 10,000$ and $\alpha \approx 1, 2$, all algorithms guess the correct topological order in around 90% of the simulations (see tables 5.13, 5.14, 5.15, 5.16). If we keep the sample size large ($n = 10,000$) and make the tails lighter ($\alpha = 5$), we see from the topological matrices 5.8 that there is a slight loss in performance, especially for *Fast* algorithm. The results in this setting are a good starting point. In fact, they suggest that lemma 4.4.0.1 might also hold in the case of confounding variables.

5.2 Some comments

In the previous simulations, we have observed that for large sample size ($n = 10,000$) and heavy-tailed noise ($\alpha \approx 1, 2$) the four methods are similar concerning their Q-performance and topological matrices. However, when we depart from this ideal setup (i.e., when n is smaller and $\alpha \approx 4, 5$), *Drop*, *Regress* and *Remove* algorithm are preferable than *Fast* algorithm. Also, we have studied the impact of the parameters n, α, b. In particular:

1. The sample size n seems to be crucial for the performance of the algorithms. For example, if we look at table reftab:chain5Drop, when $\alpha = 1$, the performance improves from 30% ($n = 100$) to 95% ($n = 10000$) roughly. Similar behaviour happens across the board for all algorithms and simulations. Therefore, we conclude that the results proven in Chapter refmethod tend to hold mainly in the population case (i.e., when $n \to \infty$).

2. The tail-index α (in our case the degrees of freedom) also plays an important role. As we depart from the heavy-tailed assumption, the algorithms start to "break". This should be no surprise, however. Indeed, the four algorithms are built upon the Γ coefficient, which relies on the heavy-tailed distribution of the noise (see definition refdef:def-gamma).

3. It is not clear the effect of b. Recall that b determines the average size of the β coefficients in the SEM. Looking at tables with the Q-performance it seems that there is an interaction between the parameters α and b. We conjecture that it is not b alone, rather the ration b/α to impact the algorithms' performance. If we consider a linear SEM, each equation is made of two parts: signal and noise. The parameter b regulates the signal, whereas the parameter α regulates the noise. From the simulations, it seems that the algorithms fail when the signal is high ($b \approx 1.5$) compared to the noise ($\alpha \approx 5$). This phenomenon is in line with the theoretical results of Chapter refmethod. Consider, for example, the following SEM:

$$
\begin{aligned}
X_1 &\leftarrow Y_1 \\
X_2 &\leftarrow \underbrace{\beta X_1}_{\text{signal}} + \underbrace{Y_1}_{\text{noise}}
\end{aligned}
$$

where $P(Y_i > u) \sim \ell(u)u^{-\alpha}$ as $u \to \infty$, $i = 1, 2$. Since X_1 causes X_2, then $\Gamma_{12} = 1$. On the other hand, one can show that $\Gamma_{21} = 1 - \frac{1}{2}\frac{1}{1+\beta^{\alpha}}$. If we take β large enough (i.e., b is large) then the signal dominates the noise and $\Gamma_{21} \approx 1$. Therefore, $\Gamma_{12} \approx \Gamma_{21}$ and we cannot detect anymore the causal order between the variables.

Table 5.1: **Simple chain**. Q-performance for ***Drop*** algorithm

$n = 100$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.220	0.145	0.065	0.070	0.035
$b = 1$	0.365	0.185	0.115	0.075	0.060
$b = 1.5$	0.400	0.220	0.105	0.115	0.090
$n = 1000$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.805	0.505	0.405	0.335	0.300
$b = 1$	0.905	0.700	0.485	0.435	0.315
$b = 1.5$	0.910	0.625	0.395	0.310	0.190
$n = 10000$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.945	0.805	0.710	0.63	0.595
$b = 1$	0.965	0.910	0.850	0.75	0.720
$b = 1.5$	0.995	0.880	0.685	0.60	0.465

Table 5.2: **Simple chain**. Q-performance for ***Regress*** algorithm

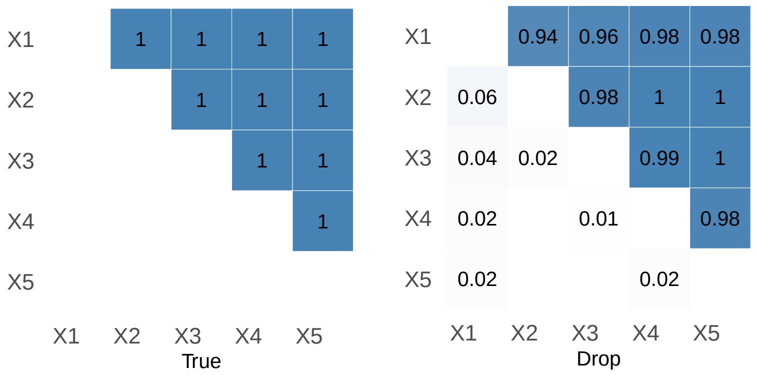
$n = 100$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.210	0.100	0.07	0.06	0.055
$b = 1$	0.345	0.165	0.18	0.09	0.060
$b = 1.5$	0.485	0.285	0.16	0.13	0.115
$n = 1000$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.805	0.635	0.445	0.390	0.255
$b = 1$	0.905	0.715	0.630	0.475	0.350
$b = 1.5$	0.945	0.760	0.655	0.460	0.390
$n = 10000$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.935	0.740	0.680	0.67	0.580
$b = 1$	0.985	0.945	0.855	0.78	0.765
$b = 1.5$	0.985	0.935	0.900	0.84	0.755

Table 5.3: **Simple chain**. Q-performance for ***Remove*** algorithm

$n = 100$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.270	0.175	0.115	0.055	0.065
$b = 1$	0.405	0.215	0.125	0.085	0.100
$b = 1.5$	0.475	0.345	0.215	0.210	0.150
$n = 1000$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.775	0.535	0.455	0.395	0.295
$b = 1$	0.895	0.745	0.470	0.395	0.295
$b = 1.5$	0.860	0.575	0.410	0.285	0.260
$n = 10000$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.96	0.805	0.73	0.605	0.645
$b = 1$	0.94	0.910	0.82	0.795	0.640
$b = 1.5$	0.99	0.770	0.67	0.505	0.445

Table 5.4: **Simple chain**. Q-performance for ***Fast*** algorithm

$n = 100$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.165	0.075	0.070	0.035	0.055
$b = 1$	0.185	0.085	0.110	0.050	0.055
$b = 1.5$	0.195	0.130	0.115	0.135	0.115
$n = 1000$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.655	0.415	0.210	0.180	0.120
$b = 1$	0.745	0.510	0.305	0.240	0.205
$b = 1.5$	0.630	0.330	0.255	0.225	0.125
$n = 10000$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.91	0.575	0.495	0.45	0.375
$b = 1$	0.97	0.830	0.615	0.54	0.410
$b = 1.5$	0.92	0.625	0.395	0.31	0.305

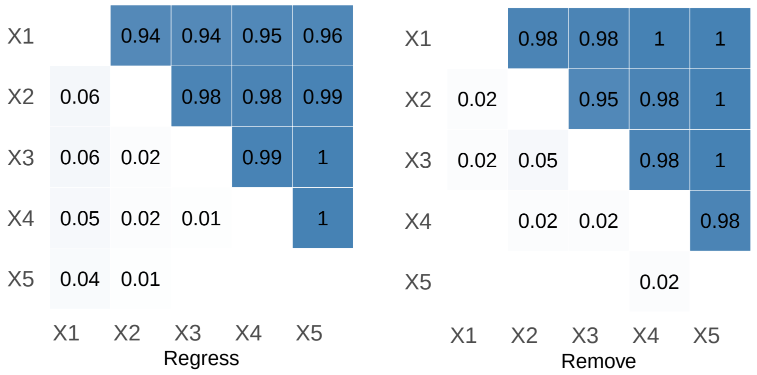


	X1	X2	X3	X4	X5
X1		1	1	1	1
X2			1	1	1
X3				1	1
X4					1
X5					

True

	X1	X2	X3	X4	X5
X1		0.94	0.96	0.98	0.98
X2	0.06		0.98	1	1
X3	0.04	0.02		0.99	1
X4	0.02		0.01		0.98
X5	0.02			0.02	

Drop

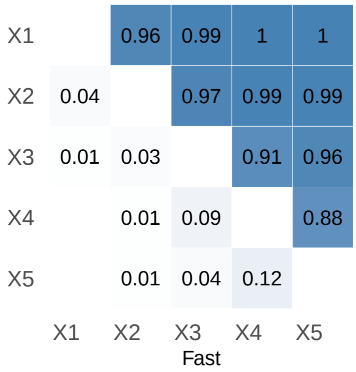


	X1	X2	X3	X4	X5
X1		0.94	0.94	0.95	0.96
X2	0.06		0.98	0.98	0.99
X3	0.06	0.02		0.99	1
X4	0.05	0.02	0.01		1
X5	0.04	0.01			

Regress

	X1	X2	X3	X4	X5
X1		0.98	0.98	1	1
X2	0.02		0.95	0.98	1
X3	0.02	0.05		0.98	1
X4		0.02	0.02		0.98
X5				0.02	

Remove



	X1	X2	X3	X4	X5
X1		0.96	0.99	1	1
X2	0.04		0.97	0.99	0.99
X3	0.01	0.03		0.91	0.96
X4		0.01	0.09		0.88
X5		0.01	0.04	0.12	

Fast

Figure 5.5: **Simple chain**. Topological matrices for $n = 1000$, $\alpha = 1$, $b = 2$

Table 5.5: **Polytree**. Q-performance for ***Drop*** algorithm

$n = 100$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.285	0.115	0.115	0.090	0.05
$b = 1$	0.320	0.165	0.090	0.045	0.05
$b = 1.5$	0.355	0.150	0.125	0.105	0.05
$n = 1000$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.825	0.63	0.53	0.335	0.22
$b = 1$	0.920	0.66	0.47	0.315	0.22
$b = 1.5$	0.865	0.56	0.34	0.225	0.16
$n = 10000$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.960	0.845	0.825	0.750	0.600
$b = 1$	0.995	0.920	0.745	0.675	0.600
$b = 1.5$	0.980	0.815	0.685	0.475	0.325

Table 5.6: **Polytree**. Q-performance for ***Regress*** algorithm

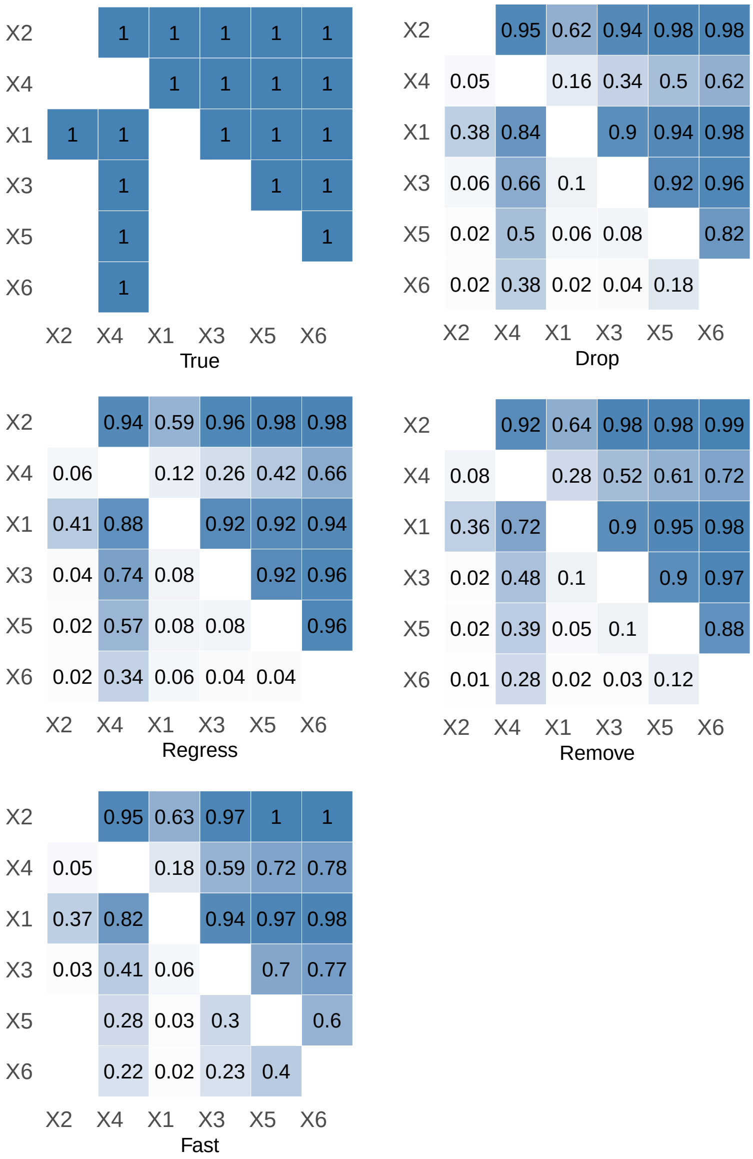
$n = 100$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.245	0.155	0.05	0.085	0.060
$b = 1$	0.390	0.220	0.16	0.105	0.070
$b = 1.5$	0.370	0.235	0.16	0.115	0.075
$n = 1000$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.875	0.67	0.575	0.410	0.385
$b = 1$	0.930	0.75	0.615	0.415	0.370
$b = 1.5$	0.925	0.73	0.585	0.405	0.295
$n = 10000$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.98	0.88	0.82	0.755	0.665
$b = 1$	0.99	0.92	0.88	0.855	0.755
$b = 1.5$	1.00	0.97	0.80	0.700	0.585

Table 5.7: **Polytree**. Q-performance for ***Remove*** algorithm

$n = 100$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.300	0.190	0.065	0.065	0.035
$b = 1$	0.405	0.225	0.085	0.105	0.060
$b = 1.5$	0.445	0.230	0.105	0.100	0.085
$n = 1000$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.855	0.630	0.495	0.345	0.295
$b = 1$	0.870	0.700	0.440	0.330	0.280
$b = 1.5$	0.850	0.655	0.365	0.280	0.255
$n = 10000$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.975	0.835	0.705	0.690	0.695
$b = 1$	0.955	0.910	0.800	0.715	0.645
$b = 1.5$	0.970	0.835	0.540	0.500	0.375

Table 5.8: **Polytree**. Q-performance for ***Fast*** algorithm

$n = 100$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.235	0.100	0.065	0.09	0.040
$b = 1$	0.195	0.095	0.080	0.06	0.055
$b = 1.5$	0.200	0.065	0.040	0.03	0.080
$n = 1000$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.78	0.47	0.330	0.305	0.225
$b = 1$	0.76	0.49	0.295	0.250	0.215
$b = 1.5$	0.63	0.35	0.215	0.170	0.150
$n = 10000$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.935	0.730	0.685	0.560	0.465
$b = 1$	0.985	0.845	0.665	0.470	0.325
$b = 1.5$	0.940	0.675	0.460	0.265	0.215



	X2	X4	X1	X3	X5	X6
X2		1	1	1	1	1
X4			1	1	1	1
X1	1	1		1	1	1
X3		1			1	1
X5		1				1
X6		1				

True

	X2	X4	X1	X3	X5	X6
X2		0.95	0.62	0.94	0.98	0.98
X4	0.05		0.16	0.34	0.5	0.62
X1	0.38	0.84		0.9	0.94	0.98
X3	0.06	0.66	0.1		0.92	0.96
X5	0.02	0.5	0.06	0.08		0.82
X6	0.02	0.38	0.02	0.04	0.18	

Drop

	X2	X4	X1	X3	X5	X6
X2		0.94	0.59	0.96	0.98	0.98
X4	0.06		0.12	0.26	0.42	0.66
X1	0.41	0.88		0.92	0.92	0.94
X3	0.04	0.74	0.08		0.92	0.96
X5	0.02	0.57	0.08	0.08		0.96
X6	0.02	0.34	0.06	0.04	0.04	

Regress

	X2	X4	X1	X3	X5	X6
X2		0.92	0.64	0.98	0.98	0.99
X4	0.08		0.28	0.52	0.61	0.72
X1	0.36	0.72		0.9	0.95	0.98
X3	0.02	0.48	0.1		0.9	0.97
X5	0.02	0.39	0.05	0.1		0.88
X6	0.01	0.28	0.02	0.03	0.12	

Remove

	X2	X4	X1	X3	X5	X6
X2		0.95	0.63	0.97	1	1
X4	0.05		0.18	0.59	0.72	0.78
X1	0.37	0.82		0.94	0.97	0.98
X3	0.03	0.41	0.06		0.7	0.77
X5		0.28	0.03	0.3		0.6
X6		0.22	0.02	0.23	0.4	

Fast

Figure 5.6: **Polytree**. Topological matrices for $n = 10000$, $\alpha = 5$, $b = 2$

Table 5.9: **Non-Polytree**. Q-performance for ***Drop*** algorithm

$n = 100$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.185	0.105	0.090	0.080	0.025
$b = 1$	0.260	0.120	0.085	0.040	0.030
$b = 1.5$	0.240	0.170	0.075	0.065	0.030
$n = 1000$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.815	0.565	0.380	0.320	0.230
$b = 1$	0.855	0.540	0.295	0.225	0.155
$b = 1.5$	0.740	0.350	0.245	0.140	0.075
$n = 10000$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.965	0.80	0.75	0.70	0.585
$b = 1$	0.965	0.85	0.68	0.54	0.415
$b = 1.5$	0.990	0.64	0.42	0.24	0.145

Table 5.10: **Non-Polytree**. Q-performance for ***Regress*** algorithm

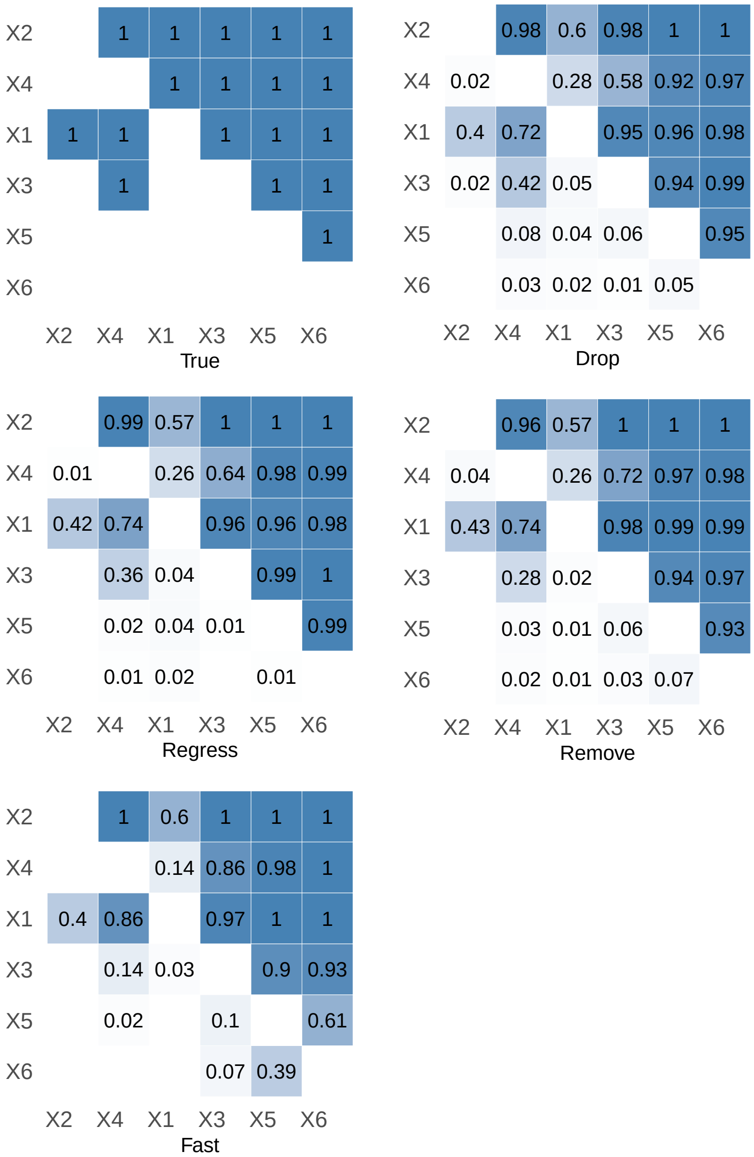
$n = 100$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.300	0.140	0.085	0.065	0.075
$b = 1$	0.385	0.250	0.180	0.095	0.095
$b = 1.5$	0.460	0.195	0.150	0.110	0.050
$n = 1000$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.840	0.66	0.515	0.415	0.365
$b = 1$	0.915	0.76	0.675	0.550	0.415
$b = 1.5$	0.915	0.75	0.550	0.480	0.280
$n = 10000$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.99	0.820	0.775	0.735	0.720
$b = 1$	0.99	0.970	0.875	0.845	0.800
$b = 1.5$	0.99	0.965	0.880	0.780	0.705

Table 5.11: **Non-Polytree**. Q-performance for ***Remove*** algorithm

$n = 100$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.300	0.10	0.06	0.040	0.055
$b = 1$	0.325	0.19	0.10	0.045	0.030
$b = 1.5$	0.450	0.19	0.11	0.080	0.090
$n = 1000$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.84	0.625	0.440	0.360	0.255
$b = 1$	0.88	0.650	0.420	0.315	0.195
$b = 1.5$	0.81	0.570	0.325	0.185	0.240
$n = 10000$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.975	0.825	0.745	0.690	0.665
$b = 1$	0.975	0.840	0.740	0.585	0.460
$b = 1.5$	0.980	0.805	0.460	0.395	0.290

Table 5.12: **Non-Polytree**. Q-performance for ***Fast*** algorithm

$n = 100$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.195	0.055	0.05	0.015	0.030
$b = 1$	0.205	0.080	0.05	0.025	0.010
$b = 1.5$	0.215	0.060	0.04	0.020	0.025
$n = 1000$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.76	0.555	0.365	0.230	0.105
$b = 1$	0.71	0.465	0.235	0.160	0.125
$b = 1.5$	0.49	0.305	0.165	0.155	0.085
$n = 10000$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.955	0.805	0.700	0.550	0.525
$b = 1$	0.965	0.735	0.505	0.325	0.275
$b = 1.5$	0.885	0.510	0.270	0.270	0.135



	X2	X4	X1	X3	X5	X6
X2		1	1	1	1	1
X4			1	1	1	1
X1	1	1		1	1	1
X3		1			1	1
X5						1
X6						

True

	X2	X4	X1	X3	X5	X6
X2		0.98	0.6	0.98	1	1
X4	0.02		0.28	0.58	0.92	0.97
X1	0.4	0.72		0.95	0.96	0.98
X3	0.02	0.42	0.05		0.94	0.99
X5		0.08	0.04	0.06		0.95
X6		0.03	0.02	0.01	0.05	

Drop

	X2	X4	X1	X3	X5	X6
X2		0.99	0.57	1	1	1
X4	0.01		0.26	0.64	0.98	0.99
X1	0.42	0.74		0.96	0.96	0.98
X3		0.36	0.04		0.99	1
X5		0.02	0.04	0.01		0.99
X6		0.01	0.02		0.01	

Regress

	X2	X4	X1	X3	X5	X6
X2		0.96	0.57	1	1	1
X4	0.04		0.26	0.72	0.97	0.98
X1	0.43	0.74		0.98	0.99	0.99
X3		0.28	0.02		0.94	0.97
X5		0.03	0.01	0.06		0.93
X6		0.02	0.01	0.03	0.07	

Remove

	X2	X4	X1	X3	X5	X6
X2		1	0.6	1	1	1
X4			0.14	0.86	0.98	1
X1	0.4	0.86		0.97	1	1
X3		0.14	0.03		0.9	0.93
X5		0.02		0.1		0.61
X6				0.07	0.39	

Fast

Figure 5.7: **Non-Polytree**. Topological matrices for $n = 1000$, $\alpha = 1$, $b = 3$

Table 5.13: **Graph with confounders**. Q-performance for ***Drop*** algorithm

$n = 100$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.490	0.250	0.17	0.120	0.130
$b = 1$	0.475	0.240	0.19	0.185	0.135
$b = 1.5$	0.480	0.345	0.22	0.280	0.190
$n = 1000$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.890	0.750	0.605	0.50	0.335
$b = 1$	0.900	0.680	0.465	0.33	0.290
$b = 1.5$	0.805	0.505	0.345	0.22	0.215
$n = 10000$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.990	0.920	0.88	0.830	0.715
$b = 1$	0.995	0.895	0.73	0.535	0.495
$b = 1.5$	0.980	0.805	0.51	0.420	0.315

Table 5.14: **Graph with confounders**. Q-performance for ***Regress*** algorithm

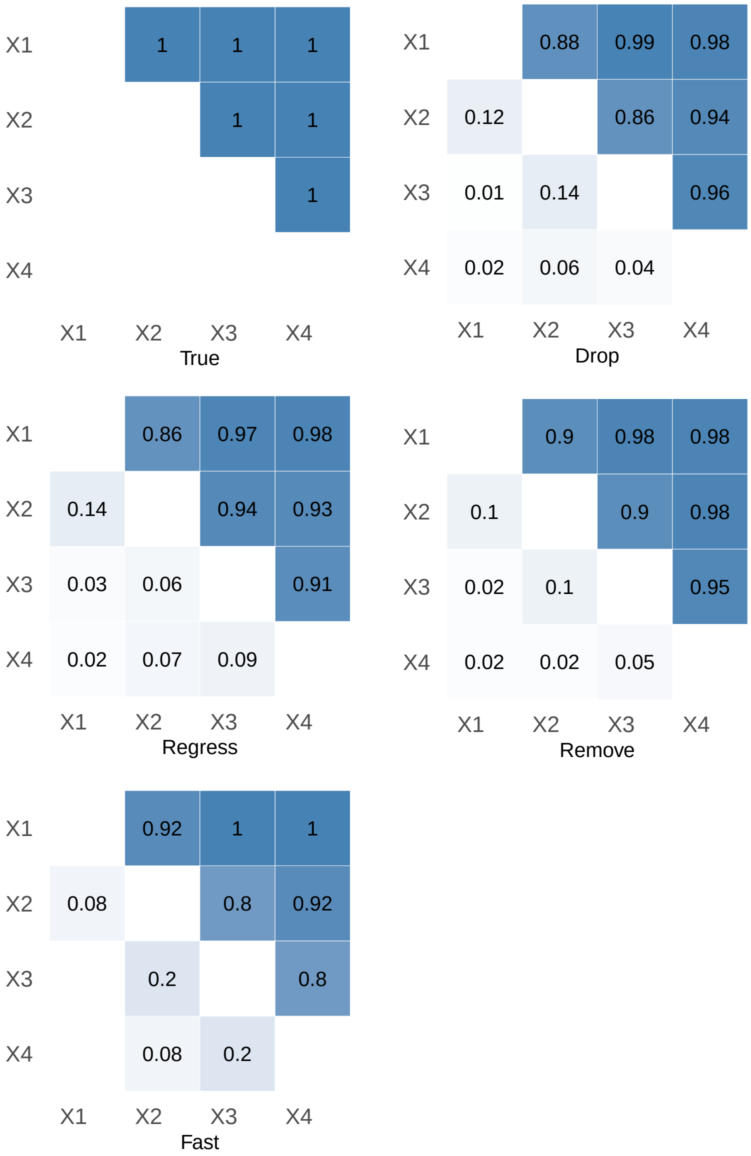
$n = 100$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.480	0.28	0.18	0.155	0.130
$b = 1$	0.465	0.28	0.23	0.180	0.120
$b = 1.5$	0.505	0.29	0.27	0.155	0.125
$n = 1000$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.895	0.800	0.595	0.500	0.420
$b = 1$	0.910	0.700	0.465	0.340	0.265
$b = 1.5$	0.835	0.535	0.310	0.275	0.205
$n = 10000$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.985	0.925	0.860	0.840	0.715
$b = 1$	0.990	0.900	0.735	0.615	0.455
$b = 1.5$	0.990	0.805	0.465	0.320	0.235

Table 5.15: **Graph with confounders**. Q-performance for ***Remove*** algorithm

$n = 100$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.435	0.280	0.200	0.140	0.175
$b = 1$	0.560	0.335	0.225	0.210	0.195
$b = 1.5$	0.485	0.340	0.300	0.265	0.285
$n = 1000$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.92	0.755	0.590	0.495	0.420
$b = 1$	0.82	0.640	0.445	0.280	0.295
$b = 1.5$	0.83	0.500	0.325	0.280	0.220
$n = 10000$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	1.000	0.905	0.885	0.785	0.745
$b = 1$	0.985	0.890	0.645	0.520	0.490
$b = 1.5$	0.945	0.740	0.475	0.310	0.280

Table 5.16: **Graph with confounders**. Q-performance for ***Fast*** algorithm

$n = 100$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.375	0.235	0.150	0.175	0.100
$b = 1$	0.360	0.200	0.205	0.120	0.120
$b = 1.5$	0.410	0.220	0.260	0.200	0.235
$n = 1000$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.875	0.685	0.560	0.39	0.295
$b = 1$	0.825	0.500	0.425	0.26	0.215
$b = 1.5$	0.650	0.375	0.315	0.23	0.220
$n = 10000$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
$b = 0.5$	0.990	0.850	0.750	0.630	0.565
$b = 1$	0.995	0.825	0.605	0.525	0.380
$b = 1.5$	0.940	0.715	0.405	0.305	0.285



True

	X1	X2	X3	X4
X1		1	1	1
X2			1	1
X3				1
X4				

Drop

	X1	X2	X3	X4
X1		0.88	0.99	0.98
X2	0.12		0.86	0.94
X3	0.01	0.14		0.96
X4	0.02	0.06	0.04	

Regress

	X1	X2	X3	X4
X1		0.86	0.97	0.98
X2	0.14		0.94	0.93
X3	0.03	0.06		0.91
X4	0.02	0.07	0.09	

Remove

	X1	X2	X3	X4
X1		0.9	0.98	0.98
X2	0.1		0.9	0.98
X3	0.02	0.1		0.95
X4	0.02	0.02	0.05	

Fast

	X1	X2	X3	X4
X1		0.92	1	1
X2	0.08		0.8	0.92
X3		0.2		0.8
X4		0.08	0.2	

Figure 5.8: **Graph with confounders**. Topological matrices for $n = 10000$, $\alpha = 5$, $b = 1$

Chapter 6

Conclusions

We introduced a novel method to estimate a causal order from observational data. Built upon the ideas of Engelke et al. (2018), we first proved the population properties of the Γ coefficient for a random vector $X = (X_1, \ldots, X_p)$ modeled by a linear SEM. We assume positive β's coefficients and regularly varying distribution for the noise. In the population case, we proved that it is possible to identify the source node of a linear SEM. Based on these results, we built four algorithms to estimate the causal order of a random vector $X = (X_1, \ldots, X_p)$ from observational data. We tested the sample properties of the algorithms on simulated data. From the simulations results we concluded the following points:

- The four algorithms perform equally well when the sample size n is large, and when the tails of the distribution α are heavy. When we depart from these assumptions, we see that *Fast*-algorithm underperforms compared to the others (i.e., *Drop, Regress* and *Remove*-algorithm).
- All four algorithms' performance highly depends on the sample size n. In some simulations, we observe more than a five-fold improvement in the performance when we increase the number of observations from 100 to $10,000$.
- The tail-index α, which determines the thickness of the tails, is also very important in the performance of the algorithms. The performance decreases more than 50% when α goes from one to five. This effect is sharper when the sample size is small (e.g., $n = 100$).
- It is not clear the effect of b, i.e., the average size of the β coefficients. We conjecture that the ratio b/α affects the performance of the algorithms.
- Overall, the algorithms perform well when the assumptions of Chapter 4 are met (i.e., when $n \to \infty$ and when $\alpha \approx 1$). When we depart from the assumptions, the algorithms estimates get worse.

There are still many open questions, but we believe that the following points can be the first ones to address in future related works:

1. Extend lemma 4.4.0.1 to the case of hidden confounders
2. Study and prove the asymptotic properties of the Γ coefficient defined in 4.1.0.1
3. Make the algorithms more robust to the sample size n
4. Test the algorithms on real data from different fields (e.g., finance and genetics)

Appendix A

Proofs

In this appendix, we present the proofs for some of the propositions and lemmas stated in Chapter 2 and Chapter 3.

A.1 Proof of Proposition 2.1.0.9

The proof is based on Peters et al. (2017), Appendix B, Proposition B.2.

Proof. We proceed by induction. We start by showing that in each DAG there is a root node. Start with any node and move to one of its parents (if there is any). You will never visit a parent that you have seen before (if you did there had been a directed cycle). After at most $p-1$ steps you reach a node without any parent (a *root* node). Let's call this node i. Since i is a root node it can be put first in a topological order, $i = \pi^{-1}(1)$. Consider the subgraph G' obtained by removing the root node i, i.e. $V' = V \setminus \{i\}$. We see that G' is still a DAG, hence we can find a root node, say j, and put it in second position in the topological order, $j = \pi^{-1}(2)$. If we repeat this procedure until no more vertices are left, we end up with a valid topological order π for G. □

A.2 Proof of Proposition 2.3.2.2

The proof is taken from Meinshausen (2018), Section 2.1, Proposition 2.1.2.

Proof. Using a topological ordering π we can write each node i as a function of the noise terms Y_j with $\pi(j) \leq \pi(i)$ (using the SEM iteratively). That is:

$$X_i \leftarrow g_i((Y_j)_{j:\pi(j)\leq\pi(i)})$$

□

A.3 Proof of Proposition 3.2.0.3

The proof is taken from Nair et al. (2018), Section 3.1.

Proof. ($\Leftarrow$) Suppose $\overline{F}$ has power-law tail. Then, there exists a $c \geq 0$, $k \geq 0$ and $\alpha > 0$ such that $\overline{F}(x) = kx^{-\alpha}$, for $x \geq c$. Then, for λ, x such that $\lambda x \geq c$ we have

$$\overline{F}(\lambda x) = k(\lambda x)^{-\alpha} = \lambda^{-\alpha} k x^{-\alpha} = \lambda^{-\alpha}\overline{F}(x)$$

where $g(\lambda) = \lambda^{-\alpha}$ is continuous for $\lambda > 0$.

($\Rightarrow$) First, note that the case where $\overline{F}$ is identically zero over $[c, \infty)$ trivially satisfies the conditions of the proposition (this corresponds to $k = 0$). Excluding the trivial case, we can see that $\overline{F}(x) > 0$ for all $x \geq c$. Suppose, in fact, that there is $x' \geq c$ such that $\overline{F}(x') = 0$. Then $\overline{F}(x) = \overline{F}(x' \cdot \frac{x}{x'}) = g(\frac{x}{x'})\overline{F}(x') = 0$ for all $x \geq c$. This means that we are back to the trivial case.

Now, consider $x, y > 0$. Pick a z large enough such that $z,\ z \cdot x,\ z \cdot x \cdot y \geq c$. It follows that $\overline{F}(z \cdot x \cdot y) = \overline{F}(z)g(x \cdot y)$. We could also write $\overline{F}(z \cdot x \cdot y) = \overline{F}(x \cdot z)g(y) = \overline{F}(z)g(x)g(y)$. Since $\overline{F}(z) > 0$, we can conclude that

$$g(x \cdot y) = g(x)g(y), \qquad \text{for all } x, y > 0 \tag{A.3.0.1}$$

It can be shown that the only non-zero function satisfying A.3.0.1 is $g(x) = x^{-\alpha}$, for some $\alpha \in \mathbb{R}$. Hence

$$\overline{F}(x) = \overline{F}(c \cdot x/c) = g(x/c)\overline{F}(c) = c^{\alpha}x^{-\alpha}\overline{F}(c) = kx^{-\alpha}, \qquad \text{for } x \geq c$$

where $k = c^{\alpha}\overline{F}(c) > 0$. Finally, since we are dealing with a distribution, we want that $\lim_{x\to\infty} \overline{F}(x) = 0$, so we impose $\alpha > 0$.

□

A.4 Proof of Proposition 3.3.0.3

The proof is taken from Nair et al. (2018), Section 3.3.

Proof. ($\Leftarrow$) Suppose $f(x) = \ell(x)x^{\alpha}$, where $\ell \in RV(0)$. Then

$$\frac{f(\lambda x)}{f(x)} = \frac{\ell(\lambda x)}{\ell(x)}\frac{(\lambda x)^{\alpha}}{x^{\alpha}} \sim 1 \cdot \lambda^{\alpha}, \qquad \text{as } x \to \infty$$

since $\lim_{x\to\infty} \frac{\ell(\lambda x)}{\ell(x)} = \lambda^0 = 1$ by definition 3.3.0.2. Thus, $f \in RV(\alpha)$

($\Rightarrow$) Suppose $f \in RV(\alpha)$. Then, $\lim_{x\to\infty} \frac{f(\lambda x)}{f(x)} = \lambda^{\alpha}$ for $\lambda > 0$. Define $\ell(x) = f(x)/x^{\alpha}$. We see that $\ell(x)$ is slowly varying

$$\frac{\ell(\lambda x)}{\ell(x)} = \frac{f(\lambda x)}{f(x)}\frac{1}{\lambda^{\alpha}} \sim \lambda^{\alpha}\frac{1}{\lambda^{\alpha}} = 1, \qquad \text{as } x \to \infty$$

Hence, we can write $f(x) = \ell(x)x^{\alpha}$, where $\ell \in RV(0)$. □

A.5 Proof of Proposition 3.3.0.5

The proof is taken from Nair et al. (2018), Section 3.2. It follows the same line as the proof of proposition 3.2.0.3.

Proof. ($\Leftarrow$) Suppose $\overline{F} \in RV(-\alpha)$ where $\alpha > 0$. Then we can be write as $\overline{F}(x) = \ell(x)x^{-\alpha}$, where ℓ is a slowly varying function. We have that

$$\lim_{x\to\infty} \frac{\overline{F}(\lambda x)}{\overline{F}(x)} = \lim_{x\to\infty} \frac{\lambda^{-\alpha}\overline{F}(x)}{\overline{F}(x)} = \lambda^{-\alpha} = g(\lambda)$$

We see that g is a strictly positive, finite and continuous, for $\lambda > 0$ and $\alpha > 0$. Hence $\overline{F}$ is asymptotically scale invariant.

($\Rightarrow$) Fix $x, y > 0$. The asymptotic scale-free property implies that

$$\lim_{z\to\infty} \frac{\overline{F}(xyz)}{\overline{F}(z)} = g(xy)$$

We can also compute the same limit by writing $\frac{\overline{F}(xyz)}{\overline{F}(z)} = \frac{\overline{F}(xyz)}{\overline{F}(xz)}\frac{\overline{F}(xz)}{\overline{F}(z)}$. Note that $\frac{\overline{F}(xyz)}{\overline{F}(xz)} \to g(y)$ and $\frac{\overline{F}(xz)}{\overline{F}(z)} \to g(x)$ as $z \to \infty$. This implies that

$$\lim_{z\to\infty} \frac{\overline{F}(xyz)}{\overline{F}(z)} = g(x)g(y)$$

Thus, we conclude that the function g satisfies

$$g(x \cdot y) = g(x)g(y), \qquad \text{for all } x, y > 0 \tag{A.5.0.1}$$

It can be shown that the only non-zero function satisfying A.5.0.1 is $g(x) = x^{-\alpha}$, for some $\alpha \in \mathbb{R}$. Hence

$$\lim_{x\to\infty} \frac{\overline{F}(\lambda x)}{\overline{F}(x)} = g(\lambda) = \lambda^{-\alpha}$$

where $\lambda > 0$. Finally, since we are dealing with a distribution, we want that $\lim_{x\to\infty} \overline{F}(x) = 0$, so we impose $\alpha > 0$. Hence $\overline{F} \in RV(-\alpha)$.

□

A.6 Proof of Lemma 3.4.0.1

The proof is taken from Embrechts et al. (2013), Section 1.3.1, Lemma 1.3.1.

Proof. Using $\{X+Y>x\} \supset \{X>x\} \cup \{Y>x\}$ one easily checks that

$$P(X+Y>x) \geq [P(X>x)+P(Y>x)](1-o(1))$$

If $0<\delta<1/2$, then from

$\{X+Y>x\} \subset \{X>(1-\delta)x\} \cup \{Y>(1-\delta)x\} \cup \{X>\delta x, Y>\delta x\}$

it follows that

$$\begin{aligned}
&P(X+Y>x)\\
&\leq\ P(X>(1-\delta)x)+P(Y>(1-\delta)x)+P(X>\delta x)P(Y>\delta x)\\
&=\ [P(X>(1-\delta)x)+P(Y>(1-\delta)x)](1+o(1))
\end{aligned}$$

Hence

$$\begin{aligned}
1 \leq\ & \liminf_{x\to\infty} \frac{P(X+Y>x)}{P(X>x)+P(Y>x)}\\
\leq\ & \limsup_{x\to\infty} \frac{P(X+Y>x)}{P(X>x)+P(Y>x)}\\
\leq\ & (1-\delta)^{-\alpha}
\end{aligned}$$

which proves the result as $\delta \downarrow 0$. □

A.7 Proof of Lemma 3.4.0.2

Proof. We can write

$$\begin{aligned}
P(\max(X,Y)>u) &= 1-P(\max(X,Y)<u) = 1-P(X<u, Y<u)\\
&= 1-F(u)^2 = 1-(1-\overline{F}(u))^2 = 1-(1-2\overline{F}(u)+\overline{F}(u)^2)\\
&= 2\overline{F}(u)-\overline{F}(u)^2 \sim 2\ell(u)u^{-\alpha}-(\ell(u)u^{-\alpha})^2
\end{aligned}$$

where the third equality follows from the independence of X and Y. It follows that

$$\frac{P(\max(X,Y)>u)}{P(X+Y>u)} \sim 1-\frac{(\ell(u)u^{-\alpha})^2}{2\ell(u)u^{-\alpha}} = 1-o(1), \qquad \text{as } u\to\infty$$

since $\lim_{u\to\infty}\overline{F}(u) = \lim_{u\to\infty}\ell(u)u^{-\alpha} = 0$. □

Appendix B

Probability Theory

In this Appendix we review some definitions from probability theory that are used in this thesis. The presentation is based on Peters et al. (2017), Appendix A.1.

Definition B.0.0.1. *We denote with* P *the* ***distribution*** *of the random vector* $X = (X_1, \ldots, X_p)$

Definition B.0.0.2. *We denote with* $x \mapsto f(x)$ *the* ***density*** *of the random vector* $X = (X_1, \ldots, X_p)$. *We (sometimes implicitly) assume its existence or continuity.*

Definition B.0.0.3. *We denote with* $x \mapsto F(x)$ *the* ***cumulative distribution*** *of the random vector* $X = (X_1, \ldots, X_p)$. *We denote with* $\overline{F} = 1 - F$ *the* ***right-tail*** *cumulative distribution.*

Definition B.0.0.4. *We call* X ***independent*** *of* Y *and write* $X \perp\!\!\!\perp Y$ *if and only if*

$$f(x, y) = f(x)f(y)$$

for all x, y.

Definition B.0.0.5. *We call* $X_1, \ldots, X_p$ ***jointly (or mutually) independent*** *if and only if*

$$f(x_1, \ldots, x_p) = f(x_1) \cdot \cdots \cdot f(x_p)$$

for all $x_1, \ldots, x_p$. *If* $X_1, \ldots, X_p$ *are jointly independent then all pairs* X_i *and* X_j, *with* $i \neq j$, *are independent too. The converse does not hold in general.*

Definition B.0.0.6. *We say that* X *is* ***conditionally independent*** *of* Y *given* Z, *and write* $X \perp\!\!\!\perp Y|\ Z$, *if and only if*

$$f(x, y|\ z) = f(x|\ z)\ f(y|\ z)$$

for all x, y, z *such that* $f(z) > 0$.

Bibliography

Chickering, D. M. (2002). Optimal structure identification with greedy search. *Journal of machine learning research 3*(Nov), 507–554.

Colombo, D., M. H. Maathuis, M. Kalisch, and T. S. Richardson (2012). Learning high-dimensional directed acyclic graphs with latent and selection variables. *The Annals of Statistics*, 294–321.

Comon, P., C. Jutten, and J. Herault (1991). Blind separation of sources, part ii: Problems statement. *Signal processing 24*(1), 11–20.

Embrechts, P., C. Klüppelberg, and T. Mikosch (2013). *Modelling extremal events: for insurance and finance*, Volume 33. Springer Science & Business Media.

Engelke, S., N. Meinshausen, and J. Peters (forthcoming 2018). Causality in heavy-tailed models.

Goudet, O., D. Kalainathan, P. Caillou, D. Lopez-Paz, I. Guyon, M. Sebag, A. Tritas, and P. Tubaro (2017). Learning functional causal models with generative neural networks. *arXiv preprint arXiv:1709.05321*.

Haavelmo, T. (1944). The probability approach in econometrics. *Econometrica: Journal of the Econometric Society*, iii–115.

Maathuis, M. (2017, February-May). Course in causality - lecture notes.

Meinshausen, N. (2018, February-May). Course in causality - lecture notes.

Mikosch, T. (1999). *Regular variation, subexponentiality and their applications in probability theory.* Eindhoven University of Technology.

Nair, J., A. Wierman, and B. Zwart (in press 2018). The fundamentals of heavy tails: Properties, emergence, and identification. `http://users.cms.caltech.edu/~adamw/heavytails.html`.

Pearl, J. (1988). Probabilistic reasoning in intelligent systems: networks of plausible inference.

Peters, J., D. Janzing, and B. Schölkopf (2017). *Elements of Causal Inference: Foundations and Learning Algorithms.* Cambridge, MA, USA: MIT Press.

Ramsey, J. D. (2015). Scaling up greedy causal search for continuous variables. *arXiv preprint arXiv:1507.07749*.

Shimizu, S., P. O. Hoyer, A. Hyvärinen, and A. Kerminen (2006). A linear non-gaussian acyclic model for causal discovery. *Journal of Machine Learning Research 7*(Oct), 2003–2030.

Shimizu, S., T. Inazumi, Y. Sogawa, A. Hyvärinen, Y. Kawahara, T. Washio, P. O. Hoyer, and K. Bollen (2011). Directlingam: A direct method for learning a linear non-gaussian structural equation model. *Journal of Machine Learning Research 12*(Apr), 1225–1248.

Spirtes, P., C. N. Glymour, R. Scheines, D. Heckerman, C. Meek, G. Cooper, and T. Richardson (2000). *Causation, prediction, and search.* MIT press.

Spirtes, P., C. Meek, and T. Richardson (1995). Causal inference in the presence of latent variables and selection bias. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pp. 499–506. Morgan Kaufmann Publishers Inc.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 267–288.

Vermat, T. and J. Pearl (1990). Equivalence and synthesis of causal models.

Declaration of Originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor .

Title of work (in block letters):

CAUSALITY IN HEAVY-TAILED DATA

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
GNECCO	NICOLA

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the Citation etiquette information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work .
- I am aware that the work may be screened electronically for plagiarism.
- I have understood and followed the guidelines in the document *Scientific Works in Mathematics*.

Place, date:	**Signature(s):**
ZÜRICH, 03-08-2018	Nicola Gnecco

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.