

BACHELOR THESIS

Learning causal relations from observations: A method and its application to the spread of the ‘fear factor’ in financial markets

FACULTY OF MATHEMATICS, COMPUTER SCIENCE
AND STATISTICS

AUTHOR: Jonas Gottal

MENTOR: Kilian Rückschloß

SUPERVISOR: Prof. Dr. François Bry

SUBMISSION DATE: 15th September 2022



I, Jonas Gottal, hereby confirm that I have written the accompanying thesis by myself, without contributions from any sources other than those cited in the text and acknowledgements.

Munich, the 15th September 2022

Jonas Gottal


.....
(Signature of the candidate)

Learning causal relations from observations: A method and its application to the spread of the ‘fear factor’ in financial markets

Abstract

This thesis describes how to learn Bayesian networks from observations and how they provide optimal predictions, a measure to avoid overfitting and a resilient approach for incomplete data. Ultimately, their potential for causal interpretations leads to the topic of causal structure discovery. These are only some reasons why they are predominantly utilised in many areas – from medical research to finance. The latter will also be the domain of our use case with focus on the twelve most influential countries for financial markets: we apply the theory to implied volatility data, which reflects the expected market fluctuations. We show how this indicator, also referred to as the *fear factor*, spreads across the global financial markets with Switzerland as epicenter.

Contents

1 Introduction	5
2 Basics	7
2.1 Bayes theorem	7
2.2 Bayesian networks	8
2.3 Exact inference	10
2.4 Interventions as foundation for causality	11
2.5 Parameter and structure learning	12
2.6 Receiver Operating Curves (ROC) for evaluation	13
3 Related work	14
4 Learning with complete data	15
4.1 Optimal parameter learning	15
4.2 Maximum-a-posteriori (MAP) learning	18
4.3 Minimum Description Length (MDL)	19
4.4 Maximum Likelihood (ML) learning	20
4.5 Naive Bayes classifier	22
4.6 Structure learning and the Bayesian Information Criterion (BIC)	26
4.7 Inferred causation	32
4.8 Conclusion	36
5 Approach for real-life problems: incomplete data	37
5.1 Expectation Maximisation (EM) algorithm	38
5.2 EM algorithm for Bayesian networks	42
6 Use case in finance: Implied Volatility	45
6.1 Terminology: financial markets and derivatives	45
6.2 Data management: preprocessing and validation	47
6.3 Learning networks from implied volatility data	49
6.4 Evaluating networks from implied volatility data	50
6.5 Causal networks and final result	60
6.6 Critique	63
6.7 Conclusion	63
List of Figures	65
List of Tables	66
References	91

1 Introduction

Implied volatility has always rippled in the face of crises – when the dot-com bubble burst in 2000, in the midst of the 2008 financial crisis and with the onset of the Covid-19 pandemic in 2020. Since implied volatility reflects the expected fluctuation of the stock market, it is also referred to as the *fear factor*. So, how does this fear factor spread in the global financial markets, and how does one country impact other countries? In this thesis, we want to learn visual graphical models that show causal relations between the twelve most important countries for financial markets and how their dedicated expected volatilities affect each other. To do so, we need directed graphical models that represent cause-effect relationships with dedicated probability distributions – causal Bayesian networks. For this reason, we show in detail how one can learn these networks from data and infer the causal structure.

We thoroughly explain how one can learn from data through probability theory by reformulating the problem as exact inference and maximum likelihood and how Bayesian networks provide a robust foundation for learning by combining existing knowledge (a-priori probabilities) with observed data (evidence). Furthermore, we address why Bayesian networks provide optimal predictions and a measure to avoid overfitting. Considering that the topology of our graphical model is crucial, we illustrate in particular the process for the structure learning of Bayesian networks. Specifically, we cover the causal structure in great detail and demonstrate an algorithm to infer causation from our networks – this will reveal insight into the actual impact of one country on another, not just correlations. Since our obtained volatility data contains missing values and to further strengthen our approach, we show how Bayesian networks even handle decision making with incomplete data and explain the expectation maximisation algorithm in detail.

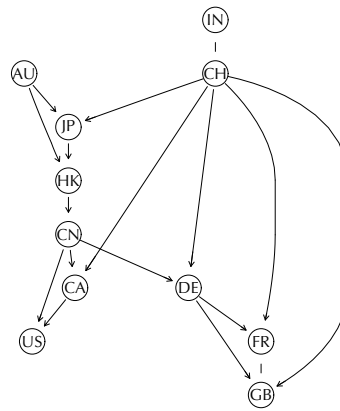


Figure 1: Our ultimate conclusion reveals Switzerland’s role as the epicentre and shows how the fear factor spreads across the global financial markets.

Our final causal Bayesian network shows thoroughly the role of Switzerland as the financial hub in the global financial market. Switzerland divides the network in Figure [4](#) into a western and an eastern hemisphere with dedicated sub-clusters. Thus, the time zones and consequently the tradable hours could be an important generating process for the implied volatility data. Although we cannot draw many conclusions, Switzerland's role as a global financial epicenter seems very intuitive.

2 Basics

We begin with an introduction to Bayesian statistics to gain a better understanding of Bayesian networks and the corresponding learning methods, explaining the terminology and basics of probability theory and networks. Then we cover the topic of exact inference – calculating a desired probability from other known probabilities. Followed by a brief example of parameter learning and structure learning. [36] As we divide our overall approach in two sections – complete and incomplete data – we need to explain those categories: We classify *complete data*, when each data point describes the values for every variable. *Incomplete data* ranges from missing values to entire hidden (*latent*) variables – which are not observable in our data. For incomplete data we have to further distinguish by means of a small running example – collecting information in a hospital: Data are *missing completely at random* (MCAR), if the absence is neither dependent on the original variable nor on other variables, e.g., a doctor losing his notes rendering that days data corrupted. *Missing at random* (MAR), meaning the missing is independent of the variable itself but the absence is dependent on other variables in the data set e.g., differences in frequency of data between genders, due to more women receiving care in this specific hospital section in general. The last category is *missing not at random* (MNAR) – meaning the the reason of a value’s absence is directly related to that variable, e.g., if a patient’s vital signs cannot be measured because the doctor felt he was too weak. [13] [35] [20] [46]

2.1 Bayes theorem

Thomas Bayes’s ‘*An essay towards solving a problem in the doctrine of chances*’ was published two years after his death in 1763 in the Philosophical Transactions of the Royal Society of London. The intended title ‘*A Method of Calculating the Exact Probability of All Conclusions founded on Induction*’ [67] gives a more reasonable insight on the topic: The first formulation of conditional probabilities and therefore the foundation of today’s *Bayes theorem*. [39]

Conditional probabilities are also called *a-posteriori probabilities* as they describe a probabilistic statement of belief *after* seeing an event. The counterpart – an unconditional probability – is called *a-priori probability*. The a-posteriori probability is defined as follows (for $P(b) > 0$): [59]

$$P(a | b) = \frac{P(a \wedge b)}{P(b)}$$

which can be rewritten as the *product rule*:

$$P(a \wedge b) = P(a | b)P(b)$$

And Bayes theorem can be derived from this product rule for hypothesis h and data d : [59]

$$P(d \wedge h) = P(d | h)P(h) \quad \text{and} \quad P(d \wedge h) = P(h | d)P(d)$$

by combining both to

$$P(h | d) = \frac{P(d | h)P(h)}{P(d)}$$

where $P(h)$ is the a-priori probability of h before seeing d and $P(h | d)$ the a-posteriori probability of h after seeing d . The following example from RUSSELL may better illustrate the relevance: While the a-priori probability of having a cavity is $P(\text{cavity}) = 0.2$, the a-posteriori probability of having a cavity, if the patient also suffers from toothache, is $P(\text{cavity} | \text{toothache}) = 0.6$. Therefore, Bayes theorem can be used for updating probabilities after obtaining new evidence – or *learning from data*. [5] [59]

2.2 Bayesian networks

From a syntactic point of view, a Bayesian network is a **D**irected **A**cyclic **G**raph (DAG) $\mathbf{G} = \langle \mathbf{V}, \mathbf{E} \rangle$ with a finite set of vertices \mathbf{V} and a set of directed edges \mathbf{E} between those vertices – meaning there are no undirected or bidirected edges. [39] Two vertices are called *adjacent* if there is an edge between them and *non-adjacent* if not. Each vertex (or node) as a representation of a discrete or continuous random variable holds some local probability information. In order to reduce complexity, this graphical representation assumes the Markov property of each node being conditionally independent of its non-descendants given its parents – thus called the *Markov assumption*. Therefore, each vertex V_i has a conditional probability distribution $P(V_i | \text{Parents}(V_i))$, quantifying the effect of the parent node and the edges between them represent conditional dependence. From a semantic point of view, a Bayesian network is a representation of the joint probability distribution: [59] [46] [35] [56]

$$P(v_1, \dots, v_n) = \prod_{i=1}^n P(v_i | \text{Parents}(V_i))$$

“[...] a Bayesian network can be viewed as a collection of probabilistic classification/regression models, organized by conditional-independence relationships.”

– David Heckerman, *A tutorial on learning with Bayesian networks* [36], p. 14]

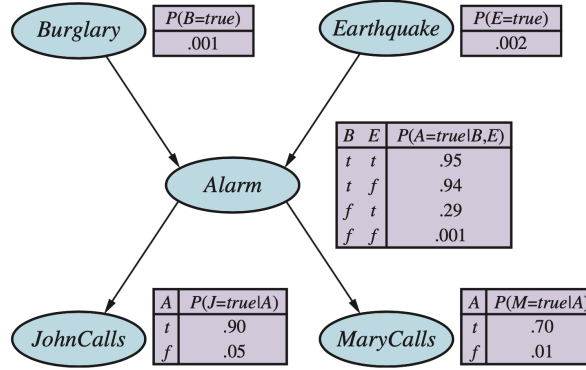


Figure 2: Exemplary Bayesian network with dedicated conditional probability tables (CPT). [59, p. 432]

This Bayesian network is based on an alarm system that reliably detects burglaries with a probability of 0.94 – as described in the CPT. However, it is also occasionally triggered by earthquakes. Additionally, there are two neighbours (John and Mary), who also call when they hear the alarm. The entries in the conditional probability tables are our *parameters*, describing the probability distribution of the Bayesian network. Thus, for this exemplary network with binary variables (either true or false) 10 parameters are necessary to describe the joint probability distribution. [59]

Although different Bayesian networks can mimic the same underlying distribution, the topological ordering of the nodes is not trivial. We aim for minimal, compact networks, which can be achieved by placing the causes before effects. [56] [59] [36]



(a) Similar network with 13 parameters. (b) Similar network with 31 parameters.

Figure 3: While all three represent the same joint distribution, the resulting network is more compact if causes are placed before effects. The required parameters of each binary node are calculated by 2^n , where n denotes the number of parents. [59, p. 436]

2.3 Exact inference

We described inference as calculating a desired probability from other known probabilities. There are two categories – exact and approximate inference. Due to the complexity of the calculations for exact inference, it is intractable to apply for large networks. Therefore, the latter uses sampling algorithms to provide approximate solutions, but these will not be further discussed. We solely focus on exact inference to calculate the a-posteriori probability \mathbf{P} after an observed event \mathbf{d} with $d_1, d_2, \dots, d_j \in \mathbf{D}$ (set of evidence variables/observable data). Where the bold \mathbf{P} is used for a vector of numbers or a conditional distribution: $\mathbf{P}(\mathbf{a} \mid \mathbf{b})$ returns the values of $P(\mathbf{a} = a_i \mid \mathbf{b} = b_j)$ for each pair of i, j . [59]

The most intuitive approach of inference is by enumeration or *marginalisation* – the summation of ‘the probabilities for each possible value of the other variables’ [59, p. 414] solely to eliminate them from the equation.

As an example for the variables \mathbf{A} and \mathbf{B} :

$$\mathbf{P}(\mathbf{A}) = \sum_{\mathbf{b}} \mathbf{P}(\mathbf{A}, \mathbf{B} = \mathbf{b})$$

So, a query for the a-posteriori probability distribution $\mathbf{P}(X \mid \mathbf{d})$ can be calculated as follows with X as query variable and \mathbf{y} as hidden (unobserved) variable.

$$\mathbf{P}(X \mid \mathbf{d}) = \frac{\mathbf{P}(X, \mathbf{d})}{P(\mathbf{d})} \stackrel{[1]}{=} \alpha \mathbf{P}(X, \mathbf{d}) = \alpha \sum_{\mathbf{y}} \mathbf{P}(X, \mathbf{d}, \mathbf{y})$$

In Bayesian networks, we already have the complete joint distribution as products of conditional probabilities, which simplifies such queries significantly – by summation of those products of conditional probabilities. Due to the constant α , there is still the need for normalisation to attain the actual probabilities (this will be explained in more detail in the following example as well as in Section 4). [59] [5]

But for now, the focus will be on our exemplary Bayesian network from RUSSELL [59], introduced in the previous section. In order to calculate the probability of a *Burglary*, given *John* and *Mary* called, $P(\text{Burglary} \mid \text{JohnCalls} = \text{true}, \text{MaryCalls} = \text{true})$, our latent variable \mathbf{y} would include the nodes *Earthquake* and *Alarm*. For more compact calculations, we simply denote the initials, e.g., instead of *JohnCalls* = *true*, we refer to j and instead of *JohnCalls* = *false*, we write $\neg j$. With the above expression this yields:

$$\mathbf{P}(\text{Burglary} \mid j, m) = \alpha \mathbf{P}(\text{Burglary}, j, m) = \alpha \sum_e \sum_a \mathbf{P}(\text{Burglary}, j, m, e, a)$$

We focus on the case of *Burglary* = *true*:

$$P(b \mid j, m) = \alpha \sum_e \sum_a P(b)P(e)P(a \mid b, e)P(j \mid a)P(m \mid a)$$

¹due to $\alpha = \frac{1}{P(\mathbf{d})}$ being a constant.

For our two binary variables e and a , this means that we calculate and then sum four individual products of probabilities. With our conditional probability tables from Figure 2, we receive these values:

Table 1: The initial results of the individual products.

	e	$\neg e$
a	$0.70 \cdot 0.90 \cdot 0.95 \cdot 0.002 \cdot 0.001 = 1.20 \times 10^{-6}$	$0.70 \cdot 0.90 \cdot 0.94 \cdot 0.998 \cdot 0.001 = 5.91 \times 10^{-4}$
$\neg a$	$0.01 \cdot 0.05 \cdot 0.05 \cdot 0.002 \cdot 0.001 = 5.00 \times 10^{-11}$	$0.01 \cdot 0.05 \cdot 0.06 \cdot 0.998 \cdot 0.001 = 2.99 \times 10^{-8}$

The sum of these four values is $P(b \mid j, m) = \alpha 5.92 \times 10^{-4}$. With the same procedure for *Burglary* = *false*, we obtain $P(\neg b \mid j, m) = \alpha 1.49 \times 10^{-3}$. Thus, to lose α and normalise these probabilities, we simply divide them by the sum of both and yield:

$$\mathbf{P}(B \mid j, m) = \alpha \langle 5.92 \times 10^{-4}, 1.49 \times 10^{-3} \rangle \approx \langle 0.28, 0.72 \rangle$$

This means, if the neighbours call, the probability of a burglary is approximately 28%.

There is further potential for improvement in terms of complexity by pruning the redundant calculations e.g., with application of the variable elimination algorithm, but this will not be part of our scope. [59]

2.4 Interventions as foundation for causality

Since different Bayesian networks can represent the same underlying distribution as described in Section 2.2, they are not necessarily portraying causal relationships. However, there are advantages in Bayesian networks, when the topological ordering of the nodes is causal. Since correlation stems from causation, causal relations are more reliable and furthermore causal networks can represent external changes – *interventions*. Thus, we use interventions to define the causal order of Bayesian networks. By intervening on individual variables, we change the underlying graph and recalculate the resulting distribution: all incoming arcs to the individual target variable are removed, the variable itself is set to 1 (or **True** for boolean variables), and no other relations than the target variable are changed. Using a small example from PEARL [56], we show how these modular configurations in causal networks provide insights into predictions from interventions. [56] [3]

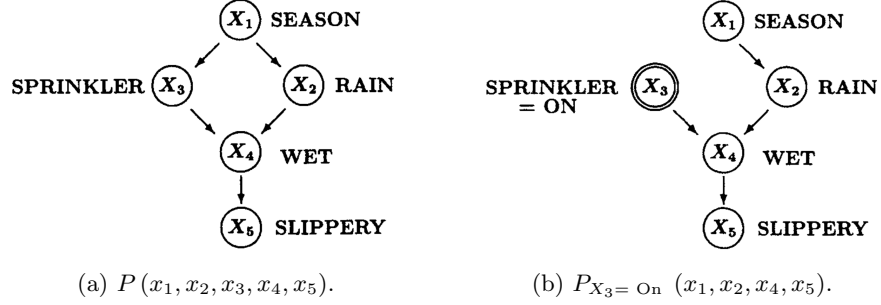


Figure 4: The joint probability distribution of the causal network is given by $P(x_1, x_2, x_3, x_4, x_5) = P(x_1)P(x_2 | x_1)P(x_3 | x_1)P(x_4 | x_2, x_3)P(x_5 | x_4)$. With one *atomic* change of setting the variable X_3 to a constant (SPRINKLER = ON) and thus removing the incoming arc from its parent (X_1), we can calculate the result of this intervention: $P_{X_3=\text{On}}(x_1, x_2, x_4, x_5) = P(x_1)P(x_2 | x_1)P(x_4 | x_2, X_3 = \text{On})P(x_5 | x_4)$. [56] p. 15, 23]

This is inherently different from *observing* $X_3 = \text{On}$, which can be calculated by $P(x_1, x_2, x_4, x_5 | X_3 = \text{On})$. The removal of an arc shows the difference between observing and doing, which is the reason the latter is called the *do*-Operator for our intervention. [56] [3]

2.5 Parameter and structure learning

As we have seen in Section 2.2 there are two components of Bayesian networks, which are crucial: The overall structure of the network and the local probability information each node holds. Therefore, we will start by analysing each part on its own before putting it together as a comprehensive learning problem. The first part will cover parameter learning: The calculation of the conditional probabilities formulated as a problem of inference and maximum likelihood given a fixed structure. The second part will focus on structure learning: Optimising the networks structure based on three different approaches: (1) constraint-based algorithms, applying conditional independence tests, (2) score-based algorithms, optimising a score function and (3) hybrid algorithms which combine both. As constraint-based and hybrid algorithms are shown to be less accurate and very rarely faster than score-based ones, we will refrain from implementing them in the thesis at hand. Since structure learning is based on concepts of parameter learning, we start with the latter. [59] [36] [62] [61]

2.6 Receiver Operating Curves (ROC) for evaluation

In order to validate learned models – here the Bayesian networks – we compare predictions made from the model with previously separated test data. In this way, we can objectively evaluate them according to their predictive qualities. For this comparison, we utilise these fundamental principles:

Sensitivity or true positive rate (TPR) is derived from the true positives TP, i.e., the correctly identified positives P from the test set: $TPR = \frac{TP}{P}$

Specificity or true negative rate (TNR) is derived from the true negatives TN, i.e., the correctly identified negatives N from the test set: $TNR = \frac{TN}{N}$

By plotting both the sensitivity and specificity in relation, we obtain the so called Receiver Operating Curve (ROC). For the results of both measures 1 or 100% is the optimum, and if the curve is the diagonal, we observed a random process. In Figure 5, we can see some common examples of curves. In order to further summarise these evaluations, we can calculate the Area Under the Curve (AUC) to rank the models. Again, a value of 0.5 indicates a random process. [35] [17] [46] [59]

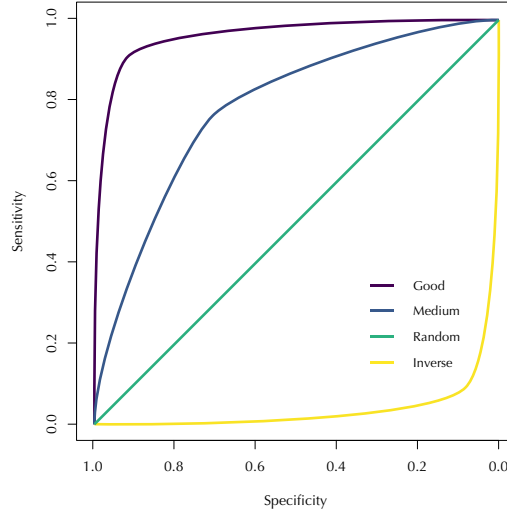


Figure 5: Exemplary receiver operating curves for a *good*, *medium*, *random* and *inverse* model fit. The perfect score would be 1.0 on each dimension – specificity and sensitivity.

3 Related work

The article ‘*A Guide to the Literature on Learning Probabilistic Networks from Data*’ [8] provides an useful overview of the complete landscape of learning networks from data. In ‘*Bayesian Theory*’ [5] the foundations in statistics and in depth discussions for our topic are covered. Complexity and the algorithmic point of view, but also great explanations of basic techniques for Bayesian learning of network structures with complete data are discussed in ‘*Learning Bayesian networks: The combination of knowledge and statistical data*’ [37]. In LAURITZEN, S. L., AND SPIEGELHALTER [47] a great and versatile example of application is presented, that we also reproduce later in this thesis. Furthermore, in the tutorial from HECKERMAN [36] the basics on learning – especially MAP learning are explained in depth. For the EM algorithm in the last section FRIEDMAN [20] is recommended. PEARL [57] and MEEK [49] are the foundation for the inferred causal models.

Beyond the research papers, the textbooks from MITCHELL [50], RUSSELL [59] and HASTIE [35] are comprehensive introductions to the basics in statistics, and PEARL [56] for causality. For the applied section of this thesis – the implementation in R – the work from NAGARAJAN [52] and SCUTARI [61] is helpful. And for the niche topic of Information Criteria the publications from SCHWARZ [60] and NEATH [54] provide useful overviews. For the fear factor itself, the implied volatility, HULL [42] is the recommended literature, as well as for options, futures, and derivatives in addition to JOSHI [45].

4 Learning with complete data

For the section of parameter learning with complete data, we will introduce a running example from RUSSELL [59]: Candy comes in two flavours (*cherry* and *lime*) – wrapped in the same paper and sold in large indistinguishable bags. There are five differently composed bags of candy, represented by h_i (*hypothesis*) generating the hypothesis space H : [59]

h_1 : 100% *cherry*
 h_2 : 75% *cherry* + 25% *lime*
 h_3 : 50% *cherry* + 50% *lime*
 h_4 : 25% *cherry* + 75% *lime*
 h_5 : 100% *lime*

Our objective is to assign the correct hypothesis to a randomly chosen bag. For the hypotheses h_1, \dots, h_5 the a-priori distribution is already given: (0.10, 0.20, 0.40, 0.20, 0.10). Nonetheless, as we cannot detect from mere observation which hypothesis h_i holds true, we need to gather evidence (data \mathbf{d}), by unwrapping one candy at a time: d_j being either *cherry* or *lime*. Our objective is to predict the next candy flavour – by application of Bayesian learning.

4.1 Optimal parameter learning

Optimal parameter learning means calculating the probability of each hypothesis, considering all observable data $d_j \in \mathbf{d}$. This means, predictions are made with all hypotheses independently, considering their probabilities – the process of inference from Section 2.3. In order to make a forecast $\mathbf{P}(X \mid \mathbf{d})$ about a random variable X , we require the a-priori probability of each hypothesis $P(h_i)$ and the likelihood of the data under each hypothesis $P(\mathbf{d} \mid h_i)$ as given. With the assumption² of observations being independent and identically distributed (i.i.d.), we can derive by application of Bayes theorem: [59] [58]

$$\begin{aligned}
 \mathbf{P}(X \mid \mathbf{d}) &= \sum_i \mathbf{P}(X \mid \mathbf{d}, h_i) \mathbf{P}(h_i \mid \mathbf{d}) \\
 &\stackrel{[3]}{=} \sum_i \mathbf{P}(X \mid h_i) \mathbf{P}(h_i \mid \mathbf{d}) \\
 &\stackrel{\text{Bayes theorem}}{=} \sum_i \mathbf{P}(X \mid h_i) \frac{P(\mathbf{d} \mid h_i) P(h_i)}{P(\mathbf{d})} \\
 &\stackrel{\text{const.}}{=} \sum_i \mathbf{P}(X \mid h_i) \alpha P(\mathbf{d} \mid h_i) P(h_i) \\
 &\stackrel{\text{i.i.d.}}{=} \sum_i \mathbf{P}(X \mid h_i) \alpha \prod_j P(d_j \mid h_i) P(h_i)
 \end{aligned}$$

²In our example this assumption is true because the bags of candy are (infinitely) large.

In order to give the equation $\mathbf{P}(X | \mathbf{d}) = \sum_i \mathbf{P}(X | h_i) \alpha \prod_j P(d_j | h_i) P(h_i)$ more insight, we go through our example in two steps and explain the intuition behind the math. Our final prediction consists of weighted averages of each hypothesis's individual prediction. 'The hypotheses themselves are essentially *intermediaries* between the raw data and the predictions.' [59], p. 773]

In the first step, we calculate the likelihood of each hypothesis under the data $\mathbf{P}(h_i | \mathbf{d}) = \alpha_j \prod_j P(d_j | h_i) P(h_i)$. In our simplified example, we selected a bag associated to h_5 : Making predictions while only drawing one flavour (*lime*) – updating the probability with each new *lime* candy up to ten times (d_1, \dots, d_{10}). So, with the composition of candy from Section 4, we can make the first calculations and determine $P(h_i) \prod_j P(d_j | h_i)$ and the current normalisation constant α_j : [59]

Table 2: $\frac{1}{\alpha} \mathbf{P}(h_i | \mathbf{d}) = P(h_i) \prod_j P(d_j | h_i)$.

Hypothesis	h_1	h_2	h_3	h_4	h_5	Marginalisation for Normalisation
$P(h_i)$	0.10	0.20	0.40	0.20	0.10	$\frac{1}{\alpha_j} = \sum_i P(h_i) \prod_j \dots$
$P(\text{lime})$	0.00	0.25	0.50	0.75	1.00	
$\mathbf{d} = d_1$	$0.10 \cdot 0.00^1$	$0.20 \cdot 0.25^1$	$0.40 \cdot 0.50^1$	$0.20 \cdot 0.75^1$	$0.10 \cdot 1.00^1$	$\frac{1}{\alpha_1} = 0.50$
$\mathbf{d} = d_2$	$0.10 \cdot 0.00^2$	$0.20 \cdot 0.25^2$	$0.40 \cdot 0.50^2$	$0.20 \cdot 0.75^2$	$0.10 \cdot 1.00^2$	$\frac{1}{\alpha_2} = 0.33$
$\mathbf{d} = d_3$	$0.10 \cdot 0.00^3$	$0.20 \cdot 0.25^3$	$0.40 \cdot 0.50^3$	$0.20 \cdot 0.75^3$	$0.10 \cdot 1.00^3$	$\frac{1}{\alpha_3} = 0.24$
$\mathbf{d} = d_4$	$0.10 \cdot 0.00^4$	$0.20 \cdot 0.25^4$	$0.40 \cdot 0.50^4$	$0.20 \cdot 0.75^4$	$0.10 \cdot 1.00^4$	$\frac{1}{\alpha_4} = 0.19$
$\mathbf{d} = d_5$	$0.10 \cdot 0.00^5$	$0.20 \cdot 0.25^5$	$0.40 \cdot 0.50^5$	$0.20 \cdot 0.75^5$	$0.10 \cdot 1.00^5$	$\frac{1}{\alpha_5} = 0.16$
$\mathbf{d} = d_6$	$0.10 \cdot 0.00^6$	$0.20 \cdot 0.25^6$	$0.40 \cdot 0.50^6$	$0.20 \cdot 0.75^6$	$0.10 \cdot 1.00^6$	$\frac{1}{\alpha_6} = 0.14$
$\mathbf{d} = d_7$	$0.10 \cdot 0.00^7$	$0.20 \cdot 0.25^7$	$0.40 \cdot 0.50^7$	$0.20 \cdot 0.75^7$	$0.10 \cdot 1.00^7$	$\frac{1}{\alpha_7} = 0.13$
$\mathbf{d} = d_8$	$0.10 \cdot 0.00^8$	$0.20 \cdot 0.25^8$	$0.40 \cdot 0.50^8$	$0.20 \cdot 0.75^8$	$0.10 \cdot 1.00^8$	$\frac{1}{\alpha_8} = 0.12$
$\mathbf{d} = d_9$	$0.10 \cdot 0.00^9$	$0.20 \cdot 0.25^9$	$0.40 \cdot 0.50^9$	$0.20 \cdot 0.75^9$	$0.10 \cdot 1.00^9$	$\frac{1}{\alpha_9} = 0.12$
$\mathbf{d} = d_{10}$	$0.10 \cdot 0.00^{10}$	$0.20 \cdot 0.25^{10}$	$0.40 \cdot 0.50^{10}$	$0.20 \cdot 0.75^{10}$	$0.10 \cdot 1.00^{10}$	$\frac{1}{\alpha_{10}} = 0.11$

After multiplying each value by the dedicated normalisation constant α_j – the reciprocal value of $\sum_i P(h_i) \prod_j P(d_j | h_i)$ – we obtain the a-posteriori likelihoods $\mathbf{P}(h_i | \mathbf{d})$ of the hypotheses.

Hypothesis	h_1	h_2	h_3	h_4	h_5
$P(h_i)$	0.10	0.20	0.40	0.20	0.10
$P(\text{lime})$	0.00	0.25	0.50	0.75	1.00
$\mathbf{d} = d_0$	0.10	0.20	0.40	0.20	0.10
$\mathbf{d} = d_1$	0.00	0.10	0.40	0.30	0.20
$\mathbf{d} = d_2$	0.00	0.04	0.31	0.35	0.31
$\mathbf{d} = d_3$	0.00	0.01	0.21	0.36	0.42
$\mathbf{d} = d_4$	0.00	0.00	0.13	0.33	0.53
$\mathbf{d} = d_5$	0.00	0.00	0.08	0.30	0.62
$\mathbf{d} = d_6$	0.00	0.00	0.04	0.25	0.70
$\mathbf{d} = d_7$	0.00	0.00	0.02	0.21	0.77
$\mathbf{d} = d_8$	0.00	0.00	0.01	0.16	0.82
$\mathbf{d} = d_9$	0.00	0.00	0.01	0.13	0.86
$\mathbf{d} = d_{10}$	0.00	0.00	0.00	0.10	0.90

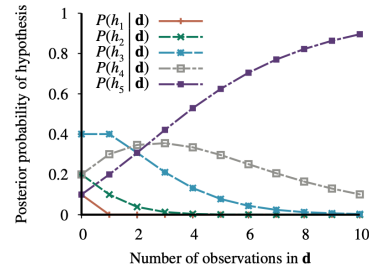


Figure 6: Posterior probabilities by the number of observations in \mathbf{d} being a *lime* candy. [59], p. 774]

Table 3: $\mathbf{P}(h_i | \mathbf{d}) = \alpha P(h_i) \prod_j P(d_j | h_i)$.

³Under the assumption that each hypothesis specifies a probability distribution over X . [58]

As we can see in the results from Table 3 in Figure 6, the likelihood of hypothesis h_5 (all *lime*) being true, increases with each *lime* candy. While this is very intuitive, the rapid decline in likelihood for hypothesis h_3 (50:50 composition), and the peak of h_4 at d_2 – two *lime* candies – is quite surprising. Only a small sample of evidence is necessary for the hypotheses to vastly distinguish themselves [59].

Finally, for our prediction of our next candy being *lime*, $\mathbf{P}(X = \text{lime} \mid \mathbf{d})$ we sum the product of the predictions within each hypothesis $\mathbf{P}(X = \text{lime} \mid h_i)$ and the previously normalised values from Table 2 $\mathbf{P}(h_i \mid \mathbf{d})$. Now we have the likelihoods over all hypotheses marginalised over the hypotheses in Table 4

Table 4: $\mathbf{P}(X \mid \mathbf{d}) = \sum_i \mathbf{P}(X \mid h_i) \mathbf{P}(h_i \mid \mathbf{d})$.

Hypothesis	h_1	h_2	h_3	h_4	h_5	Marginalisation for
$P(h_i)$	0.10	0.20	0.40	0.20	0.10	Final result
$P(\text{lime})$	0.00	0.25	0.50	0.75	1.00	
$\mathbf{d} = d_1$	$P(h_1 \mid d_1) \cdot 0.00$	$P(h_2 \mid d_1) \cdot 0.25$	$P(h_3 \mid d_1) \cdot 0.50$	$P(h_4 \mid d_1) \cdot 0.75$	$P(h_5 \mid d_1) \cdot 1.00$	$\sum_i \mathbf{P}(\dots) \mathbf{P}(\dots \mid d_1)$
$\mathbf{d} = d_2$	$P(h_1 \mid d_2) \cdot 0.00$	$P(h_2 \mid d_2) \cdot 0.25$	$P(h_3 \mid d_2) \cdot 0.50$	$P(h_4 \mid d_2) \cdot 0.75$	$P(h_5 \mid d_2) \cdot 1.00$	$\sum_i \mathbf{P}(\dots) \mathbf{P}(\dots \mid d_2)$
$\mathbf{d} = d_3$	$P(h_1 \mid d_3) \cdot 0.00$	$P(h_2 \mid d_3) \cdot 0.25$	$P(h_3 \mid d_3) \cdot 0.50$	$P(h_4 \mid d_3) \cdot 0.75$	$P(h_5 \mid d_3) \cdot 1.00$	$\sum_i \mathbf{P}(\dots) \mathbf{P}(\dots \mid d_3)$
$\mathbf{d} = d_4$	$P(h_1 \mid d_4) \cdot 0.00$	$P(h_2 \mid d_4) \cdot 0.25$	$P(h_3 \mid d_4) \cdot 0.50$	$P(h_4 \mid d_4) \cdot 0.75$	$P(h_5 \mid d_4) \cdot 1.00$	$\sum_i \mathbf{P}(\dots) \mathbf{P}(\dots \mid d_4)$
$\mathbf{d} = d_5$	$P(h_1 \mid d_5) \cdot 0.00$	$P(h_2 \mid d_5) \cdot 0.25$	$P(h_3 \mid d_5) \cdot 0.50$	$P(h_4 \mid d_5) \cdot 0.75$	$P(h_5 \mid d_5) \cdot 1.00$	$\sum_i \mathbf{P}(\dots) \mathbf{P}(\dots \mid d_5)$
$\mathbf{d} = d_6$	$P(h_1 \mid d_6) \cdot 0.00$	$P(h_2 \mid d_6) \cdot 0.25$	$P(h_3 \mid d_6) \cdot 0.50$	$P(h_4 \mid d_6) \cdot 0.75$	$P(h_5 \mid d_6) \cdot 1.00$	$\sum_i \mathbf{P}(\dots) \mathbf{P}(\dots \mid d_6)$
$\mathbf{d} = d_7$	$P(h_1 \mid d_7) \cdot 0.00$	$P(h_2 \mid d_7) \cdot 0.25$	$P(h_3 \mid d_7) \cdot 0.50$	$P(h_4 \mid d_7) \cdot 0.75$	$P(h_5 \mid d_7) \cdot 1.00$	$\sum_i \mathbf{P}(\dots) \mathbf{P}(\dots \mid d_7)$
$\mathbf{d} = d_8$	$P(h_1 \mid d_8) \cdot 0.00$	$P(h_2 \mid d_8) \cdot 0.25$	$P(h_3 \mid d_8) \cdot 0.50$	$P(h_4 \mid d_8) \cdot 0.75$	$P(h_5 \mid d_8) \cdot 1.00$	$\sum_i \mathbf{P}(\dots) \mathbf{P}(\dots \mid d_8)$
$\mathbf{d} = d_9$	$P(h_1 \mid d_9) \cdot 0.00$	$P(h_2 \mid d_9) \cdot 0.25$	$P(h_3 \mid d_9) \cdot 0.50$	$P(h_4 \mid d_9) \cdot 0.75$	$P(h_5 \mid d_9) \cdot 1.00$	$\sum_i \mathbf{P}(\dots) \mathbf{P}(\dots \mid d_9)$
$\mathbf{d} = d_{10}$	$P(h_1 \mid d_{10}) \cdot 0.00$	$P(h_2 \mid d_{10}) \cdot 0.25$	$P(h_3 \mid d_{10}) \cdot 0.50$	$P(h_4 \mid d_{10}) \cdot 0.75$	$P(h_5 \mid d_{10}) \cdot 1.00$	$\sum_i \mathbf{P}(\dots) \mathbf{P}(\dots \mid d_{10})$

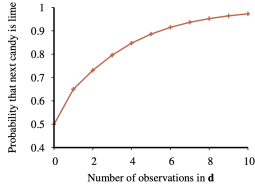


Figure 7: Bayesian prediction $P(d_{N+1} = \text{lime} \mid d_1, \dots, d_N)$ for the next $(N + 1)$ candy being *lime*. [59, p. 774]

As we can see in the results from Table 19 (Appendix) in Figure 7, the likelihood over all hypotheses h_i of the next candy being *lime*, increases with each *lime* candy. After the 8th observation in \mathbf{d} the likelihood grows to over 0.95.

The essential aspect of Bayesian learning is that the prediction is ultimately consistent with the true hypothesis. Due to the likelihood of observing data not corresponding to the real-world is shrinking with each evidence in \mathbf{d} , the a-posteriori probability of any false hypothesis will eventually vanish. [59] However, most importantly this prediction is optimal and any other method is less accurate – given the same hypothesis space and a-priori probabilities and the data. [50] In this example, the hypothesis space was $i = 5$ and for a more realistic application of an unknown composition of a bag of candy ($k = 100$ pieces) and only two flavours ($n = 2$), there would already be $i = 101$ hypotheses⁴.

$$\left(\binom{n}{k} \right) = \binom{n+k-1}{n-1} = 101$$

So, this optimality of Bayesian learning is very resource consuming and for many cases even intractable. Therefore, we need to explore more simplified methods.

⁴ h_1 : (100 *lime* | 0 *cherry*), h_2 : (99 *lime* | 1 *cherry*), ..., h_{101} : (0 *lime* | 100 *cherry*).

4.2 Maximum-a-posteriori (MAP) learning

The first simplification of optimal Bayesian parameter learning is Maximum-a-posteriori (MAP) learning. The core idea is not to make predictions by calculating weighted averages of each hypothesis’s individual prediction, but to make the prediction solely based on *the* most probable hypothesis h_{MAP} – the one maximising $P(h_i \mid \mathbf{d})$: [59]

$$h_{MAP} = \arg \max_{h_i \in H} P(h_i \mid \mathbf{d})$$

MAP predictions are approximately Bayesian with $\mathbf{P}(X \mid \mathbf{d}) \approx \mathbf{P}(X \mid h_{MAP})$, but neither optimal nor always flawless: In our running example after three *limes* (d_3), the most likely hypothesis is h_5 (100% *lime*) as seen in Figure 6. Based on that hypothesis, the prediction of another *lime* candy would be 1.0, while the optimal Bayesian prediction is 0.8 according to Figure 7. With more data the results of both approaches converge, as the a-priori distributions are quickly overridden by the data and the a-posteriori probability of any false hypothesis vanishes. Thus, for small data sets caution is advised. For suitable data sets, this approach brings significant advantages in complexity as optimisation problems are less resource draining than marginalisation of large hypothesis spaces. [59]

As mentioned in the introduction, the Bayesian approach to learn from observations, as described in Section 4.1, offers a solution to *overfitting* – the unwanted excessive adaption to training data, leading to poor performance on unobserved data. Bayesian learning penalises complexity: While often only the most complex hypothesis can reproduce the data *exactly*, the a-priori probability will usually decrease significantly with the increasing complexity of the hypothesis. This compromise between complexity of hypotheses and their adaptability to data is why MAP learning is the probabilistic implementation of Ockham’s razor: ‘Prefer the simplest hypothesis that fits the data.’ [50, p. 65]

“Pluralitas non est ponenda sine necessitate”
“Plurality [of entities] should not be posited without necessity”

– William of Ockham, (1280–1349) [59, p. 733]

4.3 Minimum Description Length (MDL)

Another perspective to view MAP Learning and Ockham's razor is the Minimum Description Length (MDL) principle. [50]

$$\begin{aligned}
h_{MAP} &= \operatorname{argmax}_{h \in H} P(h \mid \mathbf{d}) \\
&= \operatorname{argmax}_{h \in H} \frac{P(\mathbf{d} \mid h)P(h)}{P(\mathbf{d})} \\
&= \operatorname{argmax}_{h \in H} P(\mathbf{d} \mid h)P(h) \\
&\stackrel{[5]}{=} \operatorname{argmax}_{h \in H} \log_2 P(\mathbf{d} \mid h) + \log_2 P(h) \\
&\stackrel{[6]}{=} \operatorname{argmin}_{h \in H} \underbrace{-\log_2 P(\mathbf{d} \mid h)}_{\substack{\text{Number of bits} \\ \text{to specify the} \\ \text{data } \mathbf{d} \text{ in } h}} - \underbrace{\log_2 P(h)}_{\substack{\text{Number of bits} \\ \text{to specify the} \\ \text{hypothesis } h \in H}} \\
&= \operatorname{argmin}_{h \in H} \underbrace{-\log_2 P(h)}_{\substack{\text{Number of bits} \\ \text{to specify the} \\ \text{hypothesis } h \in H}} \underbrace{-\log_2 P(\mathbf{d} \mid h)}_{\substack{\text{Number of bits} \\ \text{to specify the} \\ \text{data } \mathbf{d} \text{ in } h}}
\end{aligned}$$

As explained in Section [4.2] the expression of h_{MAP} states, that short hypotheses are preferred – in this representation the minimisation of each term is evident: One for encoding the hypotheses and one for the data within the hypothesis. [59] From a perspective of information theory ‘optimal code (i.e., the code that minimizes the expected message length) assigns $-\log_2 p_i$ bits to encode message i ’ [50, p. 172] where p_i is the probability of encountering that message. This embodies the concept of (Shannon) entropy, quantifying the expected (im)purity for a random variable's possible outcomes. The description length (number of bits) L required to encode message i using code R is denoted as $L_R(i)$. With this notation we can rewrite $L_{R_h}(h) = -\log_2 P(h)$ and $L_{R_{\mathbf{d}|h}}(\mathbf{d} \mid h) = -\log_2 P(\mathbf{d} \mid h)$ with R_h and $R_{\mathbf{d}|h}$ being optimal encodings. So, h_{MAP} can be rewritten as minimisation of both description lengths – the hypothesis and the data given the hypothesis. [50]

$$h_{MAP} = \operatorname{argmin}_{h \in H} L_{R_h}(h) + L_{R_{\mathbf{d}|h}}(\mathbf{d} \mid h)$$

In order to show that MDL and MAP are equivalent ($h_{MDL} = h_{MAP}$), we choose R_1 as optimal encoding for the hypothesis R_h and R_2 as optimal encoding for the data given the hypothesis $R_{\mathbf{d}|h}$. Thus, for the Minimum Description Length principle, we can choose h_{MDL} so, [50]

$$h_{MDL} = \operatorname{argmin}_{h \in H} L_{R_1}(h) + L_{R_2}(\mathbf{d} \mid h)$$

⁵which can be expressed as maximising the \log_2 .

⁶which can be expressed as minimising the negative of the expression.

4.4 Maximum Likelihood (ML) learning

Our last simplification can be made under the assumption of every hypothesis $h \in H$ being equally probable a-priori $\forall h_i, h \in H : P(h_i) = P(h)$. [50]

$$\begin{aligned} h_{MAP} &= \operatorname{argmax}_{h \in H} P(h \mid \mathbf{d}) \\ &= \operatorname{argmax}_{h \in H} \frac{P(\mathbf{d} \mid h)P(h)}{P(\mathbf{d})} \\ &= \operatorname{argmax}_{h \in H} P(\mathbf{d} \mid h)P(h) \\ h_{ML} &= \operatorname{argmax}_{h_i \in H} P(\mathbf{d} \mid h_i) \\ \forall h_i, h \in H : P(h_i) &= P(h) \end{aligned}$$

So, we can reduce MAP learning to maximising $P(\mathbf{d} \mid h_i)$ – the maximum likelihood (ML) hypothesis h_{ML} .

Obviously, the assumption of a uniform prior over H is quite bold, but given that many hypotheses in science are set up subjectively and without any criterion to distinguish the probabilities, it is a reasonable simplification. More importantly, as stated in Section 4.2 with larger data sets, the a-priori probabilities are quickly overridden by the data and the a-posteriori probability of any false hypothesis vanishes. [59]

For our running example of candy bags from RUSSELL [59], we assume we do not know the composition of flavours (*cherry* and *lime*) – resulting in a uniform a-priori probability over hypotheses space H . For the random variable *Flavour* the probability of *cherry* is θ and for *lime* therefore $1 - \theta$. After N candies (of which c are *cherry* and $\ell = N - c$ are *lime*) we get the likelihood: [59]

$$P(\mathbf{d} \mid h_\theta) = \prod_{j=1}^N P(d_j \mid h_\theta) = \theta^c \cdot (1 - \theta)^\ell$$

h_{ML} is determined by the value θ maximising the expression above. In order to reduce the complexity of the optimisation ($\operatorname{argmax} P(\mathbf{d} \mid h_\theta)$), we maximise the log function [7] and therefore the *log likelihood*: [59]

$$L(\mathbf{d} \mid h_\theta) = \log P(\mathbf{d} \mid h_\theta) = \sum_{j=1}^N \log P(d_j \mid h_\theta) = c \log \theta + \ell \log(1 - \theta)$$

After differentiating the likelihood function L with respect to θ and setting the result to zero, we obtain: [59]

$$\frac{dL(\mathbf{d} \mid h_\theta)}{d\theta} = \frac{c}{\theta} - \frac{\ell}{1 - \theta} = 0 \quad \Rightarrow \quad \theta = \frac{c}{c + \ell} = \frac{c}{N}$$

⁷This is legitimate because $\log P$ is a monotonic function of P .

So, as we can see, the maximum likelihood hypothesis h_{ML} predicts the true composition of flavours to be equal to the candies unwrapped and observed.

As this is the default approach for maximum likelihood learning – and therefore broadly implemented – a brief recap from RUSSELL: [59]

- I Find expression for the likelihood of the data as function of the parameters.
- II Differentiate the log likelihood with respect to each parameter.
- III Set the derivative to zero to find the parameters of h_{ML} .

However, as discussed in previous sections, if the data set is too small, the results may vary: An event that has not been observed can not be predicted – meaning after seeing zero *lime* candies after N observations, h_{ML} assigns the probability 0 to ever seeing a *lime* candy. In Section [4.5] we present a way to avoid this. [59]

4.5 Naive Bayes classifier

“If it looks like a duck, swims like a duck, and quacks like a duck, then it probably is a duck.”

- The duck test, a form of abductive reasoning.^[8]

Before we start with the general form of structure learning, we introduce a simplified version: The naive Bayes classifier – structure learning of fixed (star shaped) structures. The *duck test* gives a basic introduction to the concept: If the attributes fit a certain class, we assign this class. So, ‘[i]f it looks like a duck, swims like a duck, and quacks like a duck’, we can assign with some certainty P the class variable $C = c_{duck}$. We therefore want to classify (or predict) the random variable $C \in \mathbf{C}$ under the data (attributes), where \mathbf{C} is the space of all classes. The structure of naive Bayes networks is always the same star shape with attributes as leaves and C as root – because of the assumption making the model naive: All observed attributes x_j are conditionally independent. If this assumption holds true, the approach is equivalent to MAP learning, because we choose the most probable hypothesis of classes – or in short the most probable class $C = c_{MAP}$ given the k attributes x_1, x_2, \dots, x_k : ^[59] ^[17] ^[50]

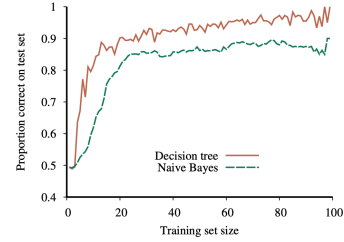


Figure 8: Efficiency of naive Bayes classifier. ^[59] p. 779]

$$\begin{aligned}
 c_{MAP} &= \operatorname{argmax}_{C \in \mathbf{C}} P(C \mid x_1, x_2 \dots x_k) \\
 &= \operatorname{argmax}_{C \in \mathbf{C}} \frac{P(x_1, x_2 \dots x_k \mid C) P(C)}{P(x_1, x_2 \dots x_k)} \\
 &\stackrel{[9]}{=} \operatorname{argmax}_{C \in \mathbf{C}} P(x_1, x_2 \dots x_k \mid C) P(C) \\
 c_{NB} &\stackrel{[10]}{=} \operatorname{argmax}_{C \in \mathbf{C}} P(C) \prod_j P(x_j \mid C)
 \end{aligned}$$

Nonetheless, even if the attributes are not conditional independent, the classifier is highly efficient, as it does not require an explicit search in the hypothesis space, and is also robust as described in Figure ^[8] ^[59]

⁸Original source unknown – early uses from James Whitcombe Riley (1916) and Ronald Reagan (1967). ^[71]

⁹Simplification due to $\alpha = \frac{1}{P(x_1, x_2, \dots, x_k)}$ being a constant.

¹⁰Simplification due to conditional independence.

As derived, the probability of each class is determined by: [59](#)

$$\mathbf{P}(C \mid x_1, x_2, \dots, x_k) = \alpha \mathbf{P}(C) \prod_j \mathbf{P}(x_j \mid C)$$

And $\mathbf{P}(C)$ and $\mathbf{P}(x_j \mid C)$ are estimated based on the observed relative frequency in the training data. [17](#)

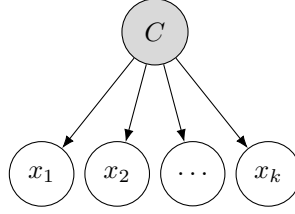


Figure 9: The Bayesian network of class C for the k attributes x_1, x_2, \dots, x_k .

As our in depth example, we focus on an everyday task: Emails – and unfortunately *spam mail*. How can we calculate as reliably and efficiently as possible the probability of the email being spam, given the attributes x_1, x_2, \dots, x_k of this new email?

$$\mathbf{P}(\text{Spam} \mid x_1, x_2, \dots, x_k)$$

We are going to implement the naive Bayes classifier to predict the class C of our incoming emails (either *Spam* or $\neg\text{Spam}$) by using their content – k words x_1, x_2, \dots, x_k – for prediction and training. At first, we have to build our histograms with the total number of words already encountered, visualised in Figure [10](#). [17](#)

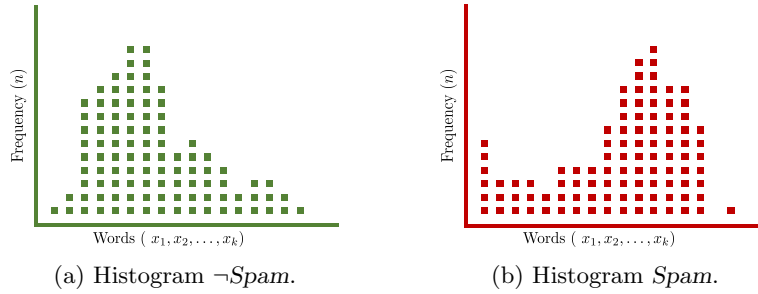


Figure 10: Both histograms show the word frequency n of each word x_1, x_2, \dots, x_k observed in either $\neg\text{Spam}$ or *Spam*.

The overall likelihood of emails being either $\neg Spam$ or $Spam$ results from their frequencies – in our example we received 12 emails, 4 of which are spam: $\mathbf{P}(\neg Spam) = \frac{8}{8+4}$ and $\mathbf{P}(Spam) = \frac{4}{8+4}$. For the words x_1, x_2, \dots, x_k we get the total number of words N for each class c_j with $\sum_{i=1}^k n_i$. 59

By application of $\alpha \mathbf{P}(C) \prod_j \mathbf{P}(x_j | C)$ in Table 5, we obtain for the words “Dear” and “Friend” the (normalised) likelihoods.

Table 5: Probability $\mathbf{P}(C | x_1, x_2, \dots, x_k)$ of classification C – either $\neg Spam$ or $Spam$ – based on frequency of words x_1, x_2, \dots, x_k (here: “Dear” and “Friend”).

\mathbf{P}	$\mathbf{P}(x_j C) = \frac{\text{Frequency}}{\text{Words } N \text{ in } C}$	$\mathbf{P}(C x_1, x_2, \dots, x_k) = \alpha \mathbf{P}(C) \prod_j \mathbf{P}(x_j C)$
$\mathbf{P}(\neg Spam) = \frac{8}{12}$	$\mathbf{P}(\text{“Dear”} \neg Spam) = \frac{8}{17}$ $\mathbf{P}(\text{“Friend”} \neg Spam) = \frac{5}{17}$	$\mathbf{P}(\neg Spam) \mathbf{P}(\text{“Dear”} \neg Spam) \mathbf{P}(\text{“Friend”} \neg Spam)$ $= \alpha \mathbf{P}(\neg Spam \text{“Dear”, “Friend”}) = 0.0922 \rightarrow \mathbf{0.93}$
$\mathbf{P}(Spam) = \frac{4}{12}$	$\mathbf{P}(\text{“Dear”} Spam) = \frac{1}{7}$ $\mathbf{P}(\text{“Friend”} Spam) = \frac{1}{7}$	$\mathbf{P}(Spam) \mathbf{P}(\text{“Dear”} Spam) \mathbf{P}(\text{“Friend”} Spam)$ $= \alpha \mathbf{P}(Spam \text{“Dear”, “Friend”}) = 0.0068 \rightarrow \mathbf{0.07}$

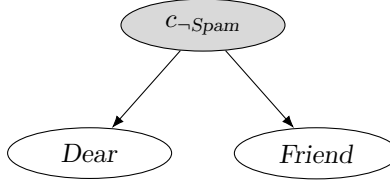


Figure 11: The dedicated Bayesian network of the naive Bayes classifier for the words “Dear” and “Friend” with the normalised probability 0.93.

We follow the same approach for the words “Money” and “Bayes”, but encounter a problem already mentioned in section 4.4:

Table 6: Probability $\mathbf{P}(C | x_1, x_2, \dots, x_k)$ of classification C – either $\neg Spam$ or $Spam$ – based on frequency of words x_1, x_2, \dots, x_k (here: “Money” and “Bayes”).

\mathbf{P}	$\mathbf{P}(x_j C) = \frac{\text{Frequency}}{\text{Words } N \text{ in } C}$	$\mathbf{P}(C x_1, x_2, \dots, x_k) = \alpha \mathbf{P}(C) \prod_j \mathbf{P}(x_j C)$
$\mathbf{P}(\neg Spam) = \frac{8}{12}$	$\mathbf{P}(\text{“Money”} \neg Spam) = \frac{1}{17}$ $\mathbf{P}(\text{“Bayes”} \neg Spam) = \frac{3}{17}$	$\mathbf{P}(\neg Spam) \mathbf{P}(\text{“Money”} \neg Spam) \mathbf{P}(\text{“Bayes”} \neg Spam)$ $= \alpha \mathbf{P}(\neg Spam \text{“Money”, “Bayes”}) = 0.0069 \rightarrow \mathbf{1.00}$
$\mathbf{P}(Spam) = \frac{4}{12}$	$\mathbf{P}(\text{“Money”} Spam) = \frac{5}{7}$ $\mathbf{P}(\text{“Bayes”} Spam) = \frac{0}{7}$	$\mathbf{P}(Spam) \mathbf{P}(\text{“Money”} Spam) \mathbf{P}(\text{“Bayes”} Spam)$ $= \alpha \mathbf{P}(Spam \text{“Money”, “Bayes”}) = 0.0000 \rightarrow \mathbf{0.00}$

Since the word “Bayes” is not yet in the vocabulary for spam, the probability will always be 0 regardless of any other words. To avoid this situation – since there is no more significance in this value – we bypass it by adding a fictitious occurrence of each word: $n = n + 1$. The total number thus increases to a whole

of 11 words for spam (+1 for each word in our list – i.e., “Dear”, “Friend”, “Money” and “Bayes”) and 21 total words for $\neg\text{spam}$. This results in the following final table: [59](#)

Table 7: Probability $\mathbf{P}(C \mid x_1, x_2, \dots, x_k)$ of classification C – either $\neg\text{Spam}$ or Spam – based on frequency of words x_1, x_2, \dots, x_k (here: “Money” and “Bayes”).

\mathbf{P}	$\mathbf{P}(x_j \mid C) = \frac{\text{Frequency}}{\text{Words in } C}$	$\mathbf{P}(C \mid x_1, x_2, \dots, x_k) = \alpha \mathbf{P}(C) \prod_j \mathbf{P}(x_j \mid C)$
$\mathbf{P}(\neg\text{Spam}) = \frac{8}{12}$	$\mathbf{P}(\text{“Money”} \mid \neg\text{Spam}) = \frac{2}{21}$ $\mathbf{P}(\text{“Bayes”} \mid \neg\text{Spam}) = \frac{4}{21}$	$\mathbf{P}(\neg\text{Spam}) \mathbf{P}(\text{“Money”} \mid \neg\text{Spam}) \mathbf{P}(\text{“Bayes”} \mid \neg\text{Spam})$ $= \alpha \mathbf{P}(\neg\text{Spam} \mid \text{“Money”}, \text{“Bayes”}) = 0.0121 \rightarrow \mathbf{0.42}$
$\mathbf{P}(\text{Spam}) = \frac{4}{12}$	$\mathbf{P}(\text{“Money”} \mid \text{Spam}) = \frac{6}{11}$ $\mathbf{P}(\text{“Bayes”} \mid \text{Spam}) = \frac{1}{11}$	$\mathbf{P}(\text{Spam}) \mathbf{P}(\text{“Money”} \mid \text{Spam}) \mathbf{P}(\text{“Bayes”} \mid \text{Spam})$ $= \alpha \mathbf{P}(\text{Spam} \mid \text{“Money”}, \text{“Bayes”}) = 0.0165 \rightarrow \mathbf{0.58}$

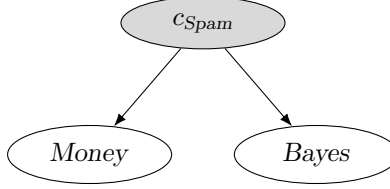


Figure 12: The dedicated Bayesian network of the naive Bayes classifier for the words “Money” and “Bayes” with the normalised probability 0.58.

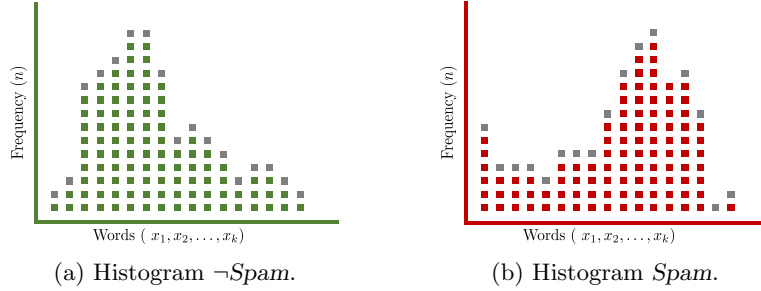


Figure 13: Updated histograms after adjusting the frequency by $n = n + 1$.

With this simple trick – visualised in Figure [13](#) – we can handle situations, where we have not seen certain data yet. Either a *lime* candy from Section [4.4](#), or the frequency $n = 0$ of certain words in our spam filter (otherwise, it could be bypassed by inventing new words).

4.6 Structure learning and the Bayesian Information Criterion (BIC)

After the special case of structure learning (the naive Bayes classifier), we can now continue with the general model. Until now, the structure was either already given, or provided through some expertise. Unfortunately, in many situations neither is an option: from medical research (e.g., “smoking can cause cancer”) to environmental studies (e.g., “effect of CO₂ concentrations on climate”) assumptions need to be backed with data.

If we cannot rely on any prior knowledge to start from a partial correct Bayesian network (and optimise from there), we begin with an empty DAG – containing no links – and start adding parents for each node and iterate for all possible orderings. Additionally, we can reverse, add, or delete links as long the graph remains acyclical as visualised in Figure 16. However, for each change while iterating, we need to test whether an appropriate structure has been found. At the very least we have to verify if the conditional independence statements – implied by the structure (Markov assumption) – are actually satisfied in our data.

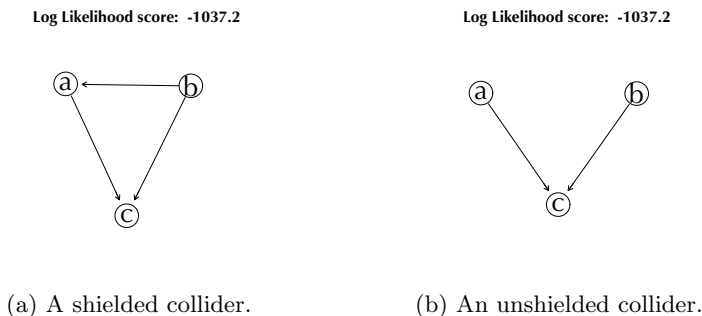


Figure 14: Comparison of Log Likelihood scores between a shielded and unshielded collider for three random binary variables a, b and c with $n = 500$.

Nonetheless, this approach will lead us to a fully connected (and almost always overfitted) network – as simplified in Figure 14. Therefore, to find a reliable model, we need to penalise complexity while maximising the likelihood of the structure generating our data. Avoiding the unwanted excessive adaption to training data and finding a compromise between complexity and the adaptability to data, sounds like MDL from Section 4.3: the information-theoretic implementation of Ockham’s razor. [59] [60] Thus, we introduce the Bayesian Information Criterion (BIC) or Schwarz Criterion, derived by SCHWARZ (1978) [60] as an asymptotic approximation to the posterior probability of a candidate model M_k , which does not depend on the prior. So, the a-posteriori most probable model (the one which seems most plausible from the available data) is preferred. [54] RISSANEN (1987) derived a minimum description length (MDL) criterion equi-

valent^[11] (the additive inverse to be precise) to the BIC. [52] Thus, its two terms will be very familiar: One measuring how well the model fits the data and one that punishes the complexity. With n observations in data \mathbf{d} for k -dimensional candidate model M_k , there is a density function $\mathcal{F}(k)$, where $L(\theta_k | \mathbf{d})$ – the likelihood function defined in Section 4.4 – indicates the likelihood corresponding to this density. By maximising that likelihood $L(\theta_k | \mathbf{d})$, we obtain the estimate $\hat{\theta}_k$ and thus $L(\hat{\theta}_k | \mathbf{d})$ for expression of the entire criterion: [36] [54]

$$\text{BIC} = \underbrace{-2 \ln L(\hat{\theta}_k | \mathbf{d})}_{\text{Model fit under data}} + \underbrace{k \ln(n)}_{\text{Complexity punishment}}$$

So, in short the maximised value of the likelihood function of the model given the data on the one hand and the number of parameters (dimensions) for the model on the other hand.

Due its simplicity, efficiency and consistency, the Bayesian Information Criterion is predominantly used for model selection and implemented in the thesis at hand.

For our example of the general case of structure learning, we use a synthetic data set – generated by a constructed Bayesian network from LAURITZEN and SPIEGELHALTER (1988). [47] With this approach, we know what the end result should look like as seen in Figure 15, and can evaluate our results appropriately.

“Shortness-of-breath (dyspnoea) may be due to tuberculosis, lung cancer or bronchitis, or none of them, or more than one of them. A recent visit to Asia increases the chances of tuberculosis, while smoking is known to be a risk factor for both lung cancer and bronchitis. The results of a single chest X-ray do not discriminate between lung cancer and tuberculosis, as neither does the presence or absence of dyspnoea. [...] either [...] is an economical way [...] of expressing our judgement that [X-ray] and [dyspnoea] do not discriminate between [tuberculosis] and [lung cancer].”

– Lauritzen and Spiegelhalter (1988) [47], p. 163], *A fictitious example*

¹¹We mention this equivalence because unfortunately the derivation of the BIC is out of scope, due to its length and it provides background for the following intuition.

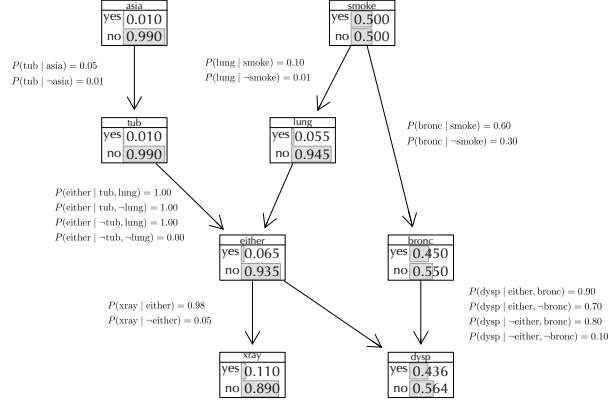


Figure 15: Original Bayesian network generating our data set ($n=5000$) with conditional probability tables (CPT) and the final probabilities as nodes.

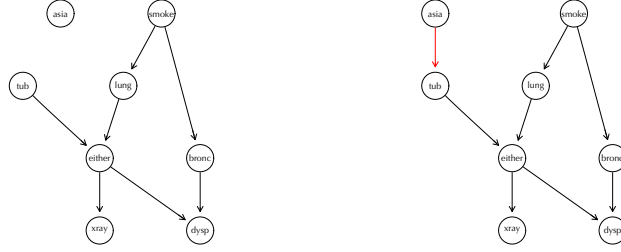
Table 8: Head of our binary sampled synthetic data set. [47](#) [61](#)

asia	smoke	tub	lung	bronc	either	xray	dysp
no	yes	no	no	yes	no	no	yes
no	yes	no	no	no	no	no	no
no	no	yes	no	no	yes	yes	yes
no	no	no	no	yes	no	no	yes
no	no	no	no	no	no	no	yes
...

Due to the number of each possible structure growing more than exponential in its number of variables, we have to resort to numerical methods like sampling (*Monte Carlo*) and/or convex optimisation (local search).

In order to find the best possible value of BIC, we implemented two greedy search algorithms to explore the search space until the global minimum is reached. As described above we start with an empty graph and reverse, add, or delete one arc at a time. The first one is called “*Hill-Climbing*” – named after an analogy to a mountaineer looking for the summit, blinded by dense fog, and directing as steeply uphill as possible. If it only goes down in all directions, he has reached the summit. With random restarts the algorithm tries to avoid local optima. The other is “*Tabu search*” – a modified version of Hill-Climbing – able to even escape local optima. This algorithm maintains a tabu list of already visited states to not revisit, which can also be leveraged to switch from a local minimum to a known better state. In our implementation, Tabu search also performs additional iterations after an optimum is found to verify a global optimum. [59](#) [61](#) [52](#)

As depicted in Figure 16, starting with an empty graph, the BIC score is recalculated for every change. The arc alteration increasing the score the most is incorporated. In step 7, the additional arc would increase the score, but since there is another possible arc (8) that would increase the score even more, this one is accepted. In step 10 – 12 every possible arc to the last node *asia* (only two of them are illustrated) decreases the BIC score.¹²



(a) Our learned Bayesian network. (b) The intended Bayesian network.

Figure 17: As the BIC punishes complexity, the arc between *asia* and *tub* decreases the overall score and thus cannot be learned as its likelihoods – 0.05 and 0.01 as depicted in Figure 15 – are too small for the sample size $n = 5000$ (thus marked red).

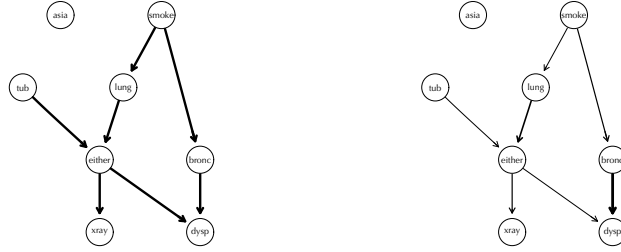
With both algorithms we are able to learn the same Bayesian network from our synthetic data. According to intuition, if we increase the sample size (here: $n > 7500$), we can also learn the missing arc (marked red in Figure 17b) and therefore the complete network as intended. However, to further distinguish both algorithms, we need to visualise the BIC itself. Thus, we introduce two additional scores and visualise them as arc strength in our DAGs. At first, we calculate the Pearson Chi-Squared (χ^2) statistic – testing the null hypothesis of conditional independence. This test provides insight into whether the relationships in the network occurred by chance. The p-value indicates significance, where lower values imply stronger relationships. And the delta of BIC denotes the change of the network score caused by the removal of the arc. So, the difference in the overall BIC score with and without the arc.¹³ 52 35 61 46

¹²In **bnlearn** the BIC is rescaled by -2 : $\text{BIC} = + \ln L(\hat{\theta}_k | \mathbf{d}) - \frac{1}{2}k \ln(n)$, thus higher values are better. 61

¹³As previously mentioned, in **bnlearn** higher BIC values correspond to better networks. 61

Table 9: Results from learning the Bayesian networks from data: Both algorithms performed equivalently and both reached the global optimum.

Arcs		Hill-Climbing algorithm		Tabu search algorithm	
from	to	p-value (χ^2)	Δ BIC value	p-value (χ^2)	Δ BIC value
<i>bronc</i>	<i>dysp</i>	0.00	-1382.14	0.00	-1382.14
<i>either</i>	<i>dysp</i>	9.67×10^{-94}	-119.61	9.67×10^{-94}	-119.61
<i>either</i>	<i>xray</i>	0.00	-768.17	0.00	-768.17
<i>lung</i>	<i>either</i>	0.00	-1084.90	0.00	-1084.90
<i>smoke</i>	<i>bronc</i>	4.22×10^{-105}	-236.87	4.22×10^{-105}	-236.87
<i>smoke</i>	<i>lung</i>	4.39×10^{-44}	-106.93	4.39×10^{-44}	-106.93
<i>tub</i>	<i>either</i>	0.00	-231.55	0.00	-231.55



(a) p-values (χ^2) ranging from 0 to 1. (b) Δ BIC values ($\in \mathbb{R}$).

Figure 18: As both algorithms result in the same scores, we only portray one graph each.

We can see that both algorithms yield the same result and the significance of all arcs indicates the network fits the distribution. Thus, we can trust our learned Bayesian network. The Δ BIC values show, that the arcs with the highest conditional probabilities also have the biggest influence in the overall network scores – which is quite intuitive. As discussed previously, only the arc with the largest positive impact on the BIC score is implemented in each iteration. Therefore, the order of arcs learned in Figure 16 corresponds to descending Δ BIC scores.

In order to verify that not just the structure, but also the parameters fit our original Bayesian network, we conduct conditional probability queries with `cpquery` – a `bnlearn` function estimating the conditional probabilities by sampling (further specified in detail in Code 1). While we could retrieve the conditional probabilities directly, this function provides the possibility for more complex and precise queries later on in Section 6.4. As we can see in Table 20 (Appendix) for the most values only minor differences can be determined. Nonetheless, for very small probabilities there are higher deviations and of course for our missing arc the value is enormous: Instead of sampling a conditional probability $P(tub | asia) = 0.050$ it obviously estimated $P(tub) = 0.010$ due to the missing link. Therefore, our parameters are as expected. 61

4.7 Inferred causation

*“I would rather discover one causal law
than be King of Persia.”*

– Democritus, (460 – 370 B.C.) [56], p. 41]

Bayesian networks provide the potential for causal interpretations, but up to this point, we have focused on probability distributions and conditional probabilities – in other words, correlation, not causation. Thus, the field of causal discovery will be of essence and we now introduce and apply methods to infer the actual causal structure. [56] [12] There are several options to learn a causal model directly, but we aim for a more comparable approach (here ROC from Section 6.4) and thus derive causality from networks by causal structure search. [33] Prior to our causal model, we introduce *d-separation* defined by PEARL: [56]

Definition. A path p is *d-separated* by a set of variables S if:

1. p contains either $i \rightarrow n \rightarrow j$ or $i \leftarrow n \rightarrow j$ such that node n is in S , or
2. p contains a collider $i \rightarrow n \leftarrow j$ such that neither node n nor a descendant of n is in S .

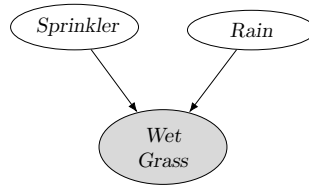
A is d-separated from B by S , if S blocks *every* undirected path from a node in A to a node in B .

Theorem. The *d-separation* of A and B by S implies the conditional independence $P(A, B \mid S) = P(A \mid S)P(B \mid S)$.

Proof. See VERMA [69]. □

This is a fundamental definition for the independence of nodes and the flow of information within the network. In Section 2.4 we already developed some preliminary concepts, but a further small example might provide more intuition: Knowing that it rained does not tell us whether the sprinkler was on. However, knowing that it did not rain after seeing the wet grass tells us, that the sprinkler must have been on.

Figure 19: Exemplary illustration of the significance of d-separation.



For our further approach, we broaden our assumptions and in addition to acyclicity we define the following: [56] [57] [66] [16] [52]

- I *Causal Markov* assumption – an extension to our previous Markov assumption and followed directly from d-separation: instead of conditional probabilities, a deterministic relationship is implied and nodes are conditionally independent of its non-descendants, given its direct causes.
- II *Causal faithfulness* or stability assumption – as it assumes all independencies to be stable and thus for any change in the parameters they will remain. So, only the d-separated nodes are independent.
- III *Causal sufficiency* assumption – of no common confounders: all the common causes of our variables are already present in our network.

Theorem. Let G and G' be two directed acyclic graphs with a finite set of vertices V . A distribution that is faithful to G is also faithful to G' , if and only if they both share the same adjacencies and unshielded colliders.

Proof. See VERMA [70]. □

Thus, for learning causal relations from observations, these equivalences in graphs are of particular interest: all structures with identical unshielded colliders and adjacencies – derived from the Markov assumption of Section 2.2 – form a *Markov equivalence class*. Since all causal structures are described by Markov equivalence classes, it is fundamental to specify them further, here exemplarily for three nodes in Figure 20 in green. The two cascading arrangements and the pattern of a common parent are all equivalent – only the v-structure or unshielded collider in red does not belong to the equivalence class. Most causal discovery approaches learn exclusively these equivalence classes. [56]

Our causal structure search is based on the *Inductive Causation* (IC) algorithm from PEARL and MEEK. The gist is the inductive reasoning of Ockham’s razor (see Sections 4.3 and 4.6) to discard any model for which we find a more compact *minimal* model that represents our data equally. This *inferred causation* means in essence, that a is supposed to have a causal influence on b if there is $a \rightarrow b$ ‘in every minimal structure consistent with the data’ [56, p. 45]. Thus, the topology of our Bayesian network – the underlying DAG – is sufficient for our causal structure search. [49] [57] [56] [16]

As we learn a causal Markov equivalence class, only unshielded colliders present in our underlying Bayesian network are to be adapted. Any further v-structures must be avoided, as illustrated in Figure 21. These rules are sound, as any other orientation in these patterns lead to further unshielded colliders or acyclicity.



Figure 20: Markov equivalence.

This approach is illustrated as pseudo code in Figure 22 and implemented in R (Appendix). 49 57

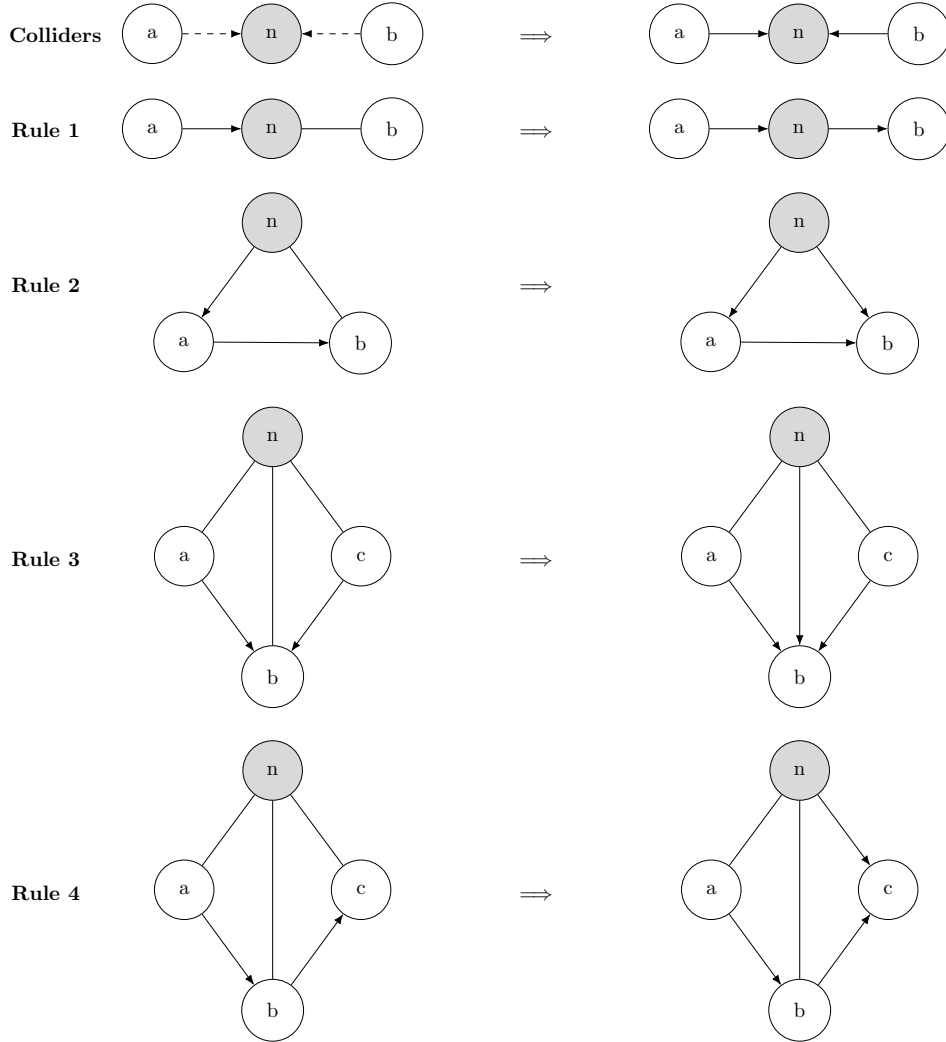


Figure 21: After orienting the unshielded colliders present in our original Bayesian network once, we follow these four rules of the Inductive Causation algorithm repeatedly until convergence to obtain a maximally oriented Partially Directed Acyclical Graph (PDAG).

Algorithm 1 Inductive Causation based on PEARL and MEEK

Input: Bayesian network (BN) from **bnlearn**
Output: Partially Directed Acyclic Graph (PDAG) in **bnlearn**

```

1: procedure INDUCTIVE CAUSATION( $BN$ )
2:    $PDAG \leftarrow \text{skeleton}(BN)$ 
3:   for nodes  $n$  in  $PDAG$  do
4:     if  $\text{neighbours}(n) > 1$  then
5:       for permutations  $(a, b)$  of  $\text{neighbours}(n)$  do
6:         if arcs  $a \rightarrow n$  and  $b \rightarrow n$  exist in  $BN$  then
7:           if no edge between  $a, b$  then
8:             add arcs  $a \rightarrow n$  and  $b \rightarrow n$  ▷ C
9:           end if
10:        end if
11:      end for
12:    end if
13:  end for
14:  new.arcs = TRUE
15:  while new.arcs = TRUE do
16:    new.arcs = FALSE
17:    for nodes  $n$  in  $PDAG$  do
18:      if  $\text{neighbours}(n) > 1$  then
19:        for permutations  $(a, b)$  of  $\text{neighbours}(n)$  do
20:          if no edge between  $a, b$  then
21:            if  $(a \rightarrow n)$  and not  $(b \rightarrow n)$  then
22:              add arc  $n \rightarrow b$  and new.arcs = TRUE ▷ R1
23:            end if
24:          end if
25:          if  $b$  in  $\text{descendants}(n)$  and not  $(b \rightarrow n)$  then
26:            add arc  $n \rightarrow b$  and new.arcs = TRUE ▷ R2
27:          end if
28:        end for
29:      end if
30:      if  $\text{neighbours}(n) > 2$  then
31:        for permutations  $(a, b, c)$  of  $\text{neighbours}(n)$  do
32:          if  $(a \rightarrow b)$  and  $(c \rightarrow b)$  then
33:            add arc  $n \rightarrow b$  and new.arcs = TRUE ▷ R3
34:          end if
35:          if  $(a \rightarrow b)$  and  $(b \rightarrow c)$  then
36:            add arc  $n \rightarrow c$  and new.arcs = TRUE ▷ R4
37:          end if
38:        end for
39:      end if
40:    end for
41:  end while
42: end procedure
  
```

Figure 22: The algorithm based on PEARL and MEEK uses **bnlearn** methods “**skeleton**”, “**neighbours**” and “**descendants**” and only adds arcs if the PDAG remains acyclical. The implementation in Code [5](#) is also extended by tests in Code [6](#).

4.8 Conclusion

Decision making based on data is a central component of all scientific disciplines – from medicine to finance and economics – but also in many industries. Either to explain or explore data or to make predictions.

In this thesis we have introduced optimal Bayesian predictions by leveraging exact inference and marginalisation and shown how to avoid overfitting with MAP learning. We examined Bayesian statistics from a perspective of information theory before deriving our final simplification – ML learning – as predominantly used in science and industry. Our everyday example of the naive Bayes classifier provided an intuitive introduction to structure learning and demonstrated how to avoid probabilities of 0. With the general case of structure learning we introduced the Bayesian Information Criterion for model selection and illustrated its use by learning a Bayesian network from data. Overall Bayesian networks provide a robust foundation for learning by combining existing knowledge (a-priori probabilities) with observed data (evidence) and offer optimal predictions and elegant solutions to overfitting. Since Bayesian networks provide the potential for causal interpretations, we concluded with causal structure search and showed how one can elegantly infer causality from Bayesian networks.

5 Approach for real-life problems: incomplete data

In reality, the data almost always looks different from our synthetic problem: *Incomplete data* – composed of missing values or hidden (*latent*) variables, which can not be observed. Therefore, in the next section we focus on more realistic assumptions for learning from data and present an approach that aims to overcome these obstacles.

a	b	c	d
NA	0	1	1
1	0	NA	0
1	NA	1	NA
NA	0	1	0
...

a	b	c	d	?
0	0	1	1	?
1	0	0	0	?
1	1	1	1	?
1	0	1	0	?
...

(a) Example for missing values – often labeled “NA” (not available).

(b) Example of a latent variable that cannot be observed.

Figure 23: Small example of cases of incomplete data.

The desire to be able to learn in the presence of missing values is obvious, but we need to address the importance of latent variables: through leveraging expert knowledge to place hidden variables – explicitly not occurring in the original network – we can often learn simpler models. In medical research for example, only the symptoms (and often the doctor’s diagnosis, resulting treatment, and the final outcome after the treatment) can be observed – but never the disease itself. So, if we try to model a Bayesian network containing behaviours and symptoms (values either *none*, *moderate*, or *severe*), the resulting structure could end in a nearly fully connected network with a total number of parameters of 708 as seen in Figure 24a. In order to reduce parameters and therefore complexity we apply our expert knowledge and place the node *heart disease*. The number of parameters could be reduced to 78 and thus less data are necessary to learn these parameters. 59 20

While it looks simple, hidden variables significantly complicate learning from a point of inference. For example in Figure 24b for the recently added node, we must find a new way to learn the conditional distribution, because we have no data. 59

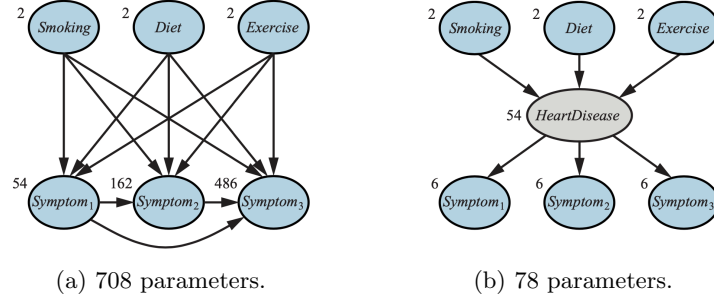


Figure 24: Reduction of variables by introducing unknown hidden variable through expert knowledge. [59, p. 789]

Both problems – missing values and latent variables – can be addressed by one common approach: The Expectation Maximisation (EM) algorithm introduced by DEMPSTER [13]. In the following section we will discuss the general form of the EM algorithm, further explain it by means of two examples and finally apply it to our synthetic data set.

5.1 Expectation Maximisation (EM) algorithm

How does the Expectation Maximisation (EM) algorithm even work? The gist is to estimate expected values for the missing data – the expectation step (*E-step*). Thereupon, recalculate the parameters by using the expected values as if they were observed values – the maximisation step (*M-step*). The algorithm iterates through both steps until it converges to a maximum likelihood hypothesis rendering the estimated values for the latent variables.

First, we need to establish new definitions: We want to estimate a set of parameters θ , describing the underlying probability distribution. For j independently occurring events, the observable data $\mathbf{d} = d_1, d_2, \dots, d_j$ and the unobservable data $\mathbf{z} = z_1, z_2, \dots, z_j$ combine to the complete data $\mathbf{y} = \mathbf{d} \cup \mathbf{z}$. The probability distribution of random variable \mathbf{z} depends on the observed data and the parameters θ . We distinguish between h as current hypothesis for the values of θ , and h' as updated hypothesis estimated for each iteration. The fundamental principle of the EM algorithm is similar to Section 4.4: Finding the maximum likelihood hypothesis h' that maximises $E[\ln P(\mathbf{y} | h')]$ – describing the expected value of our random variable \mathbf{y} . As usual we maximise the logarithm $\ln P(\mathbf{y} | h')$ instead. Thus, we define the function $Q(h' | h)$: [13] [50]

$$Q(h' | h) = E[\ln P(\mathbf{y} | h') | h, \mathbf{d}]$$

For h determined by θ , $Q(h' | h)$ returns the expected value of $P(\mathbf{y} | h')$ as a function of h' given the observed data.

The heart of the algorithm is the iteration of these two steps until it converges: [50]

- **E-step:** The Expectation step estimates the *expected* distribution over all data \mathbf{y} with observed data \mathbf{d} and current h and assigns it to $Q(h' | h)$.

$$Q(h' | h) \leftarrow E[\ln P(\mathbf{y} | h') | h, \mathbf{d}] \quad (1)$$

- **M-step:** The Maximisation step replaces the current hypothesis h with the updated hypothesis h' that *maximises* the Q function.

$$h \leftarrow \arg \max_{h' \in H} Q(h' | h) \quad (2)$$

Concisely, the E-step computes the expected log likelihood of assumed complete data – being the a-posteriori over the hidden variables, given the data. Thereupon, the M-step maximises this expected log likelihood in regard to the parameters.

Thus, to further illustrate how the algorithm works, and how to, for example, learn the distribution of *heart disease* from Figure 24b, we follow the example of DO [15] and revise our initial problem of candies from Section 4. There are two indistinguishable bags of candy with dedicated probability distributions h_1 to h_5 (but since h_1 and h_5 are too distinct, they are excluded). Thus, the two bags of candy contain either 25, 50 or 75 percent *lime* candy and the rest *cherry* candy. After drawing $N = 10$ candies from each bag three times, there are six draws of ten candies each with no indication of the two bags as illustrated in Figure 25b. In order to determine their true underlying distributions, the EM algorithm is applied. [50] [15] [59]

Bag A	Bag B	Observable counts
l l c l l l l l c l	l c c l c l l l c c	l l c l l l l l c l
l c l l l c l l l l	c c l c c l l l c c	l c c l c l l l c c
l c l c l c l l l l	l c l c l c l c c c	l c l l l c l l l l
		c c l c c l l l c c
		l c l c l c l l l l
		l c l c c l c l c c
77%	43%	60%

(a) True distributions of candies.

(b) Observable counts of candies.

Figure 25: Arbitrary sample data for the small example of the EM algorithm with the dedicated percentage of *lime* candy, where each row represents a draw of ten candies and l indicates *lime* and c *cherry* candy.

Despite the incomplete data, we attempt to calculate the true distributions based on only the six individual draws and the counts of candies assigned to them. θ_A denotes the probability of a *lime* candy from bag *A* and $1 - \theta_A$ the probability of a *cherry* candy. Furthermore, n denotes the count of *lime* candies per draw, $N = 10$ the number of total candies each and i the iteration of the algorithm. In order to estimate the true parameters, we calculate the relative frequency of the expected counts of *lime* candies. For the first *E-step*, we assume arbitrary distinct initial values for the hidden parameters $\theta_A^{(0)}$ and $\theta_B^{(0)}$ to then calculate the likelihood of each draw for each bag.

Table 10: $(\theta_A^{(i)})^n \cdot (1 - \theta_A^{(i)})^{(N-n)}$ and $(\theta_B^{(i)})^n \cdot (1 - \theta_B^{(i)})^{(N-n)}$.

For $\theta_A^{(0)} = 0.65$	For $\theta_B^{(0)} = 0.55$
$0.65^8 \cdot (1 - 0.65)^{(10-8)}$	$0.55^8 \cdot (1 - 0.55)^{(10-8)}$
$0.65^5 \cdot (1 - 0.65)^{(10-5)}$	$0.55^5 \cdot (1 - 0.55)^{(10-5)}$
$0.65^8 \cdot (1 - 0.65)^{(10-8)}$	$0.55^8 \cdot (1 - 0.55)^{(10-8)}$
$0.65^4 \cdot (1 - 0.65)^{(10-4)}$	$0.55^4 \cdot (1 - 0.55)^{(10-4)}$
$0.65^7 \cdot (1 - 0.65)^{(10-7)}$	$0.55^7 \cdot (1 - 0.55)^{(10-7)}$
$0.65^4 \cdot (1 - 0.65)^{(10-4)}$	$0.55^4 \cdot (1 - 0.55)^{(10-4)}$

Table 11: Normalisation of the likelihoods from Table 10 by marginalisation as described in Section 4.1.

For $\theta_A^{(0)} = 0.65$	For $\theta_B^{(0)} = 0.55$
0.70	0.30
0.40	0.60
0.70	0.30
0.30	0.70
0.60	0.40
0.30	0.70

To finalise the E-Step, after normalising the likelihoods, we calculate the expected counts of *lime* and *cherry* by multiplying the observed counts with the values from Table 11.

Table 12: E-step results – the expected counts of *lime* and *cherry* for each draw and bag, including the total counts for the calculation of relative frequencies.

Bag <i>A</i>		Bag <i>B</i>	
<i>lime</i>	<i>cherry</i>	<i>lime</i>	<i>cherry</i>
5.6	1.4	2.4	0.6
2.0	2.0	3.0	3.0
5.6	1.4	2.4	0.6
1.2	1.8	2.8	4.2
4.2	1.8	2.8	1.2
1.2	1.8	2.8	4.2
19.8	10.2	16.2	13.8

For the *M-step*, we derive the updated parameters by calculating the relative frequency of *lime* candy with the expected values from the Table [12](#):

$$\theta_A^{(1)} = \frac{19.8}{19.8 + 10.2} = 0.6597 \quad \text{and} \quad \theta_B^{(1)} = \frac{16.2}{16.2 + 13.8} = 0.5405$$

This procedure is repeated until convergence, as shown in the following table:

Table 13: After several iteration the algorithm converges to our desired results.

Iteration <i>i</i>	$\theta_A^{(i)}$	$\theta_B^{(i)}$
0	0.65	0.55
1	0.66	0.54
2	0.67	0.53
3	0.68	0.52
4	0.69	0.51
5	0.70	0.50
6	0.70	0.50
7	0.71	0.49
8	0.71	0.49
9	0.71	0.49
10	0.72	0.49
...

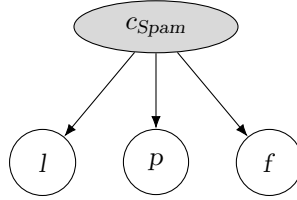
Thus, the EM algorithm converges to one distribution that is 72% and one that is 49%, which represents closest the bags h_2 and h_3 .

This very general procedure to learn from incomplete data has one major downfall: It can terminate in local optima. Furthermore, the estimation of the E-step may turn out to be intractable, as for example in large Bayesian networks. So, this algorithm often needs to be assisted by sampling procedures. As a note of caution, this approach is intended for data *missing (completely) at random* (MCAR/MAR), but should not be implemented as sole solution for data *missing NOT at random* (MNAR). [46] [59] [50] [15]

5.2 EM algorithm for Bayesian networks

In Section [4.5] we introduced the naive Bayes classifier and an example for spam emails. We will base our following adapted example from RUSSELL [59] for the EM algorithm on this and refine it even further. While their content – k words x_1, x_2, \dots, x_k – will be ignored due to privacy in this scenario, emails also often include *links* or additional *pictures*. Furthermore, we distinguish between a *formal* and a non-formal salutation. While those additional features are independent, given its category c_{Spam} , the conditional probability distribution for each feature depends on that category.

Figure 26: The Bayesian network of class *Spam* for the 3 attributes *links* l , *pictures* p and *formal* salutation f .



The notation for the parameters is as follows: θ is the a-priori probability for an email being spam. θ_{S-l} and $\theta_{\neg S-l}$ are the probabilities, that the email contains *links*, given it is spam or not spam. The same notation applies for θ_{S-p} , $\theta_{\neg S-p}$, θ_{S-f} and $\theta_{\neg S-f}$ for the conditional probabilities of *picture* and *formal salutation*. This describes a mixture model – containing a mixture of different distributions – with the category *Spam* being a hidden variable. In order to recover our classification labels, we apply the EM algorithm for data generated by the *true* model with ($n = 1000$) observations and the following parameters: [59]

$$\theta = 0.5, \quad \theta_{S-l} = \theta_{S-p} = \theta_{S-f} = 0.8, \quad \theta_{\neg S-l} = \theta_{\neg S-p} = \theta_{\neg S-f} = 0.3$$

Due to $\theta = 0.5$, the emails are equally likely to be spam or no spam. One category contains mostly a formal salutation, links and pictures and the other more informal greetings, no links nor pictures. The discrete data generated for the eight possible kinds of emails are as described below:

Table 14: The result of $n = 1000$ sampled binary observations for our parameters (1 for *true* and 0 for *false*).

	<i>formal salutation</i> = 1		<i>formal salutation</i> = 0	
	<i>links</i> = 1	<i>links</i> = 0	<i>links</i> = 1	<i>links</i> = 0
<i>picture</i> = 1	273	93	104	90
<i>picture</i> = 0	79	100	94	167

At first we need to initialise our parameters (randomly). Again, the exponent denotes the algorithms iteration:

$$\theta^{(0)} = 0.6, \quad \theta_{S-l}^{(0)} = \theta_{S-p}^{(0)} = \theta_{S-f}^{(0)} = 0.6, \quad \theta_{\neg S-l}^{(0)} = \theta_{\neg S-p}^{(0)} = \theta_{\neg S-f}^{(0)} = 0.4$$

As we cannot calculate θ for our latent variable of emails being spam directly, we estimate the expected counts instead – for $\hat{n}(Spam)$ being the sum of probability that the email is spam:

$$\theta^{(1)} = \frac{\hat{n}(Spam)}{n} = \frac{\sum_{j=1}^n P(Spam | link_j, picture_j, formal_j)}{n}$$

As explained in Section 4.5 we can calculate the inference for our naive Bayes model manually by means of conditional independence:

$$\theta^{(1)} = \frac{1}{n} \sum_{j=1}^n \frac{P(l_j | S) P(a_j | S) P(f_j | S)}{P(l_j | S) P(a_j | S) P(f_j | S) + P(l_j | \neg S) P(a_j | \neg S) P(f_j | \neg S)}$$

This formula is applied to all eight categories of emails, beginning with the 273 emails with links, pictures and formal greetings (l, p, f):

$$\frac{273}{1000} \cdot \frac{\theta_{S-l}^{(0)} \theta_{S-p}^{(0)} \theta_{S-f}^{(0)}}{\theta_{S-l}^{(0)} \theta_{S-p}^{(0)} \theta_{S-f}^{(0)} + \theta_{\neg S-l}^{(0)} \theta_{\neg S-p}^{(0)} \theta_{\neg S-f}^{(0)} (1 - \theta^{(0)})} \approx 0.22797$$

Combined with the other seven kinds of emails, we obtain $\theta^{(1)} = 0.6124$. For our individual parameters we now estimate the expected counts, starting with θ_{S-l} :

$$\sum_{j: link_j=1} P(Spam | link_j = 1, picture_j, formal_j)$$

finally obtaining for our first iteration:

$$\begin{aligned} \theta_{S-l}^{(1)} &= 0.6684, & \theta_{S-p}^{(1)} &= 0.6483, & \theta_{S-f}^{(1)} &= 0.6558, \\ \theta_{\neg S-l}^{(1)} &= 0.3887, & \theta_{\neg S-p}^{(1)} &= 0.3817, & \theta_{\neg S-f}^{(1)} &= 0.3827 \end{aligned}$$

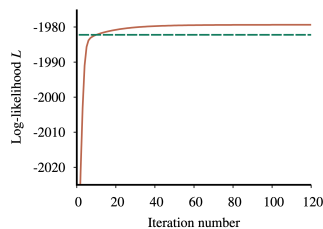


Figure 27: Likelihood of the data given the two models. [59, p. 792]

of spam emails remains: identifiability. While we could separate spam and non spam emails, we have no further way of labelling both classes correctly. Depending on the initialisation of parameters, the algorithm can converge to either one them. [59]

For our final example we implemented the *structural EM*, that combines the EM algorithm from above to optimise parameters with a graph search from Section 4.6 to also update the structure of our Bayesian network. This algorithm conducts its search in the joint space of structure and parameters. So, for each iteration, not only the parameters of the current structure can be optimised, but also a new structure can be selected as announced in Section 5. [20]

Contrary to the introductory example of this Section, expert knowledge can be lacking or be insufficient. Even if we knew a hidden variable existed, we still need to place it adequately in the network. As described in Section 4.6 we can modify all arcs as long as the graph remains acyclical and iterate for all possible orderings of our nodes. However, for the structural EM we can also add, remove or reposition new (hidden) variables. As previously described for each iteration only the structure maximising the likelihood given the current parameters is accepted and applied. [59]

[20]

So, to test the structural EM, we fragmented our synthetic data set by removing large sections of multiple variables to obtain an incomplete data set. We additionally whitelisted the missing arc from Section 4.6 because we already established, that even for complete data ($n = 5000$ observations) this link cannot be learned. [61]

While the computational run time increased significantly, the final result in Figure 28 is quite promising. The structure and the parameters could still be approximated as intended.

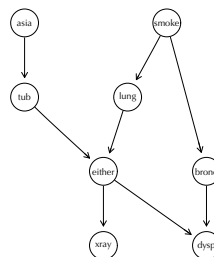


Figure 28: Final implementation of the structural EM algorithm for manually and intentionally corrupted synthetic data set in **bnlearn**.

6 Use case in finance: Implied Volatility

In order to apply our obtained knowledge, we propose a use case in finance with real world data. Our interest targets the so called *implied volatility* data and the relationships of this indicator between different countries. Since implied volatility reflects the expected fluctuation of the share price and is often used to capture the market’s sentiment towards the developments of a particular asset, it is also referred to as the *fear factor* and used to predict crises. The approach we propose is structured as follows: first, we discuss the terminology of financial markets and then analyse and process our raw data. After learning our Bayesian networks, we validate their predictive properties and compare them appropriately. Finally, we apply methods of causal discovery to infer the actual impact between countries. [45](#) [42](#) [48](#)

6.1 Terminology: financial markets and derivatives

Besides dark pools, there are two different ways to conclude a transaction: over-the-counter (OTC) between two firms and exchange trades. At exchanges for every transaction the order book is filled with the relevant (and thus for us useful) information, while OTC and dark pool trades remain invisible to the public eyes. We further separate between the stock market and derivatives market, which is significantly bigger in terms of underlying assets. By exploiting minuscule price differences on two or more exchanges to their own benefit, arbitrageurs balance these markets. Thus, we can utilise the information from the derivative exchanges to gain insights into the stock market and vice versa. But what are derivatives? As an umbrella term, it is vaguely defined as an agreement between two parties on a future transaction, where the value can be *derived* from a number of underlying variables. The two most popular derivatives are futures and options. While the future is a firm agreement to buy or sell an underlying asset *at* a specified time in the *future* at a specified price, the option gives the holder the right – the *option* – to exercise the trade *up to* the specified time. Since options are the foundation for most implied volatility calculations, we will examine them in more detail with a small example.

Table 15: In this arbitrary example we consider three *call* options – entailing the right to *buy* the underlying stock – with similar prices on the stock market (spot price). The strike price of an option is the price at which the underlying asset can be purchased until the date of maturity. We chose strike prices similar to spot prices (*at-the-money* options), so the options have no *intrinsic* value. For example, a call option with a strike price of 105.00 and a spot price of 125.00 has an intrinsic value of 20.00, due to immediate arbitrage opportunities.

Ticker	Stock name	Spot price	Strike price	Option price	Time to maturity
AAPL	Apple Inc.	167.92	170.00	1.62	30 days
META	Meta Platforms, Inc.	168.53	170.00	3.45	30 days
NVDA	NVIDIA Corporation	169.86	170.00	1.92	30 days

These options can now be used to calculate the so called *implied volatility* – volatility *implied* by the option’s prices. While calculating historical volatility – via standard deviation – is straight forward, the calculation of implied volatility is more complicated and not part of this thesis. For example, the famous Black-Scholes approach aims to approximate the price of an option by modelling the geometric brownian motion – taking into account for volatility, time to maturity, spot and strike prices among other variables. If applied in the opposite direction, this model can calculate the implied volatility for the underlying asset, using the current option prices. Since the exact calculations are not scope of this thesis, we choose to provide more intuition with our example: while all three stocks are traded roughly at the same price and the options also have the same strike price and time to maturity, they differ significantly in their option prices – prices that arise directly from the assessments of all market participants. Thus, a higher option price (minus the intrinsic value) is the result of greater uncertainty about the whereabouts of the underlying asset until maturity. This uncertainty implied by option prices is our implied volatility and this also shows why it is often called the fear factor. [45] [42]

Implied volatilities are calculated for countless financial products – but our focus is on national equity market indices, replicating a countries stock market performance in one instrument. We gathered a data set of implied volatility indices from the twelve most important countries for financial markets in terms of market capitalisation¹⁴ of listed companies (summarised in Table [21]). [72] Intraday data would be optimal to analyse the simultaneous changes on each exchange, but unfortunately, we were only able to collect daily data (end of day). However, for a long time horizon of up to 10 years. [42]

¹⁴Product of the stock price and the number of available shares. [42]

6.2 Data management: preprocessing and validation

Our data set (Table 21, Appendix) contains twelve different implied volatility indices, each replicating the implied volatility of national stock market indices. For reasons of readability, we refer to them in the following by the name of the respective country. The raw time series of daily implied volatility in percent points (Figure 29) demonstrates an important property for our further analyses: stationarity. Thus, our means, variances and correlations do not vary excessively and we do not have to address the influence of a global trend within our time series. Also within non-dynamic Bayesian networks the underlying generating process can not change over time and the predictive accuracy of our models would differ at different points in time. Obviously, our data shows some outliers (especially Russia in 2014, due to oil prices) and trends at more granular level, but in general our data provides sufficient quality. 33 64

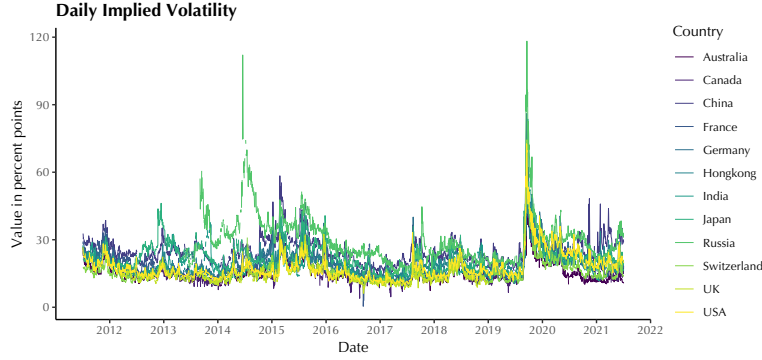


Figure 29: Stationary time series: an index value of 30 means that the implied volatility of the underlying is considered to be at 30%.

However, we care mostly about the daily changes while the different markets interact. Therefore, we calculated them as daily *log returns* for multiple reasons: the most important one is, that a stock market can be approximated as a geometric Brownian motion. Thus, we can apply the famous *Itô's lemma*¹⁵ to derive a log-normal distribution and since we are aiming for normally distributed variables, we chose the natural logarithm to achieve this. Furthermore, this approach yields a normalised data set (Figure 30). 45 42

¹⁵Itô calculus is beyond the scope of this thesis, but is a recommendation for interested readers and enthusiasts for stochastic calculus.

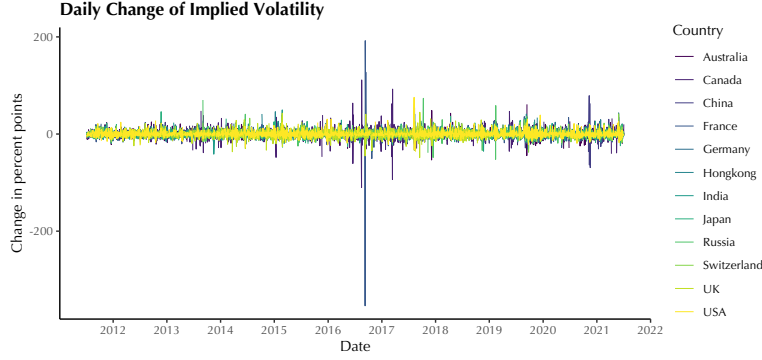


Figure 30: Daily log returns.

Since we want to analyse the impact of one country on another, but do not need to predict the exact value of the index, we are satisfied with the up-and downward movements. Thus, we discretise our data, with 1 representing positive and -1 negative changes. Due to the immense activity in these large financial instruments, there are no sideways movements and therefore no values of 0. As in Figure 31 illustrated, the values are balanced and not skewed in either direction.

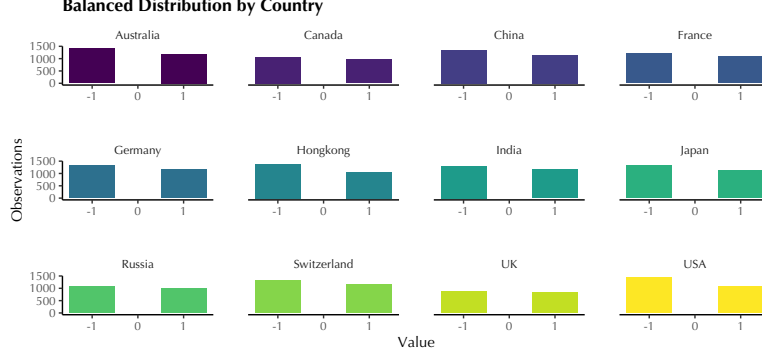


Figure 31: A balanced distribution between -1 and 1.

For validation purposes, we randomly split our dataset row-wise into a test and training set with the ratio $\frac{1}{3}$ and $\frac{2}{3}$, leaving still enough data to train reliable models. For the EM algorithm, we use the incomplete data as is, but for the score based approaches we omit the rows of the train set with missing values – resulting in the EM algorithm seeing more data. The test set for both is without rows of missing values and therefore the same to be comparable for subsequent validation. [46] [35]

6.3 Learning networks from implied volatility data

With our processed and discretised data, we can now learn our Bayesian network’s structure and parameters. Due to the diversity of algorithms and score functions in **bnlearn**, we have to make some restrictions. As previously discussed in Section 2.5, hybrid and constraint-based approaches perform less accurately. [62] Therefore, we only implemented three hybrid algorithms (*Restricted Maximization*, *Hybrid HPC*, *Max-Min Hill-Climbing*) to verify this hypothesis for our data, but mostly focus on the score functions in Table 16 optimised by Hill-Climbing and Tabu from Section 4.6. Furthermore, we implement the EM algorithm from Section 5.2, with the maximisation step conducted by each Hill-Climbing and Tabu.

Table 16: Overview of all scoring functions available using **bnlearn** and implemented with Hill-Climbing and Tabu. [9] [63] [61]

Abbreviation	Name	Category	Score equivalent
AIC	Akaike Information Criterion	Information-theoretic	✓
BIC	Bayesian Information Criterion	Information-theoretic	✓
fNML	factorised Normalized Maximum Likelihood	Information-theoretic	✗
qNML	quotient Normalized Maximum Likelihood	Information-theoretic	✓
Loglik	Log-likelihood	Information-theoretic	✓
Pred-Loglik	Predictive Log-likelihood	Information-theoretic	✓
BDe	Bayesian Dirichlet equivalent	Bayesian	✓
BDj	Bayesian Dirichlet equivalent (Jeffrey’s prior)	Bayesian	✗
BDLa	locally averaged Bayesian Dirichlet	Bayesian	✗
BDs	Bayesian Dirichlet sparse	Bayesian	✗
mBDe	modified Bayesian Dirichlet equivalent	Bayesian	✗
K2	K2	Bayesian	✗

As the name suggests, information-theoretic scoring functions are based on information theory and often equivalent to a well established concept (e.g., $BIC \Leftrightarrow MDL$, $Loglik \Leftrightarrow$ (cross) entropy), and Bayesian scoring functions utilise Bayes theorem and maximise the a-posteriori probability distribution of the networks. [9] [61] *Score-equivalence* means, the scoring function assigns the same score to all networks in one Markov equivalence class, denoted by identical adjacencies and unshielded colliders (illustrated in Figure 32). We already covered them in Section 4.7. [52] [61] [10]

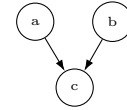


Figure 32: Unshielded collider or v-structure

Thus, to verify – or exclude – the presented scoring functions, as well as hybrid- and score-based algorithms, we run a k -fold cross validation to measure the loss of our data. This means we partition our data in k equal-sized parts, and learn the model with $k - 1$ parts. In order to validate them, we calculate the negated expected log-likelihood – *negative entropy* – of the remaining partition k , where lower values are better. [52] [35] [61] The results in Figure 33 show, that we can not exclude certain combinations, as the overall loss is very similar.

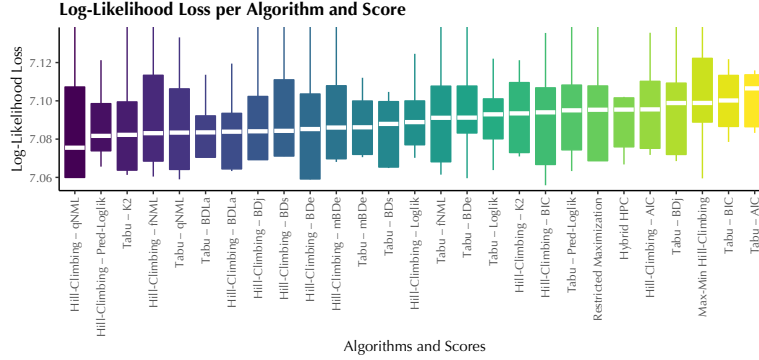


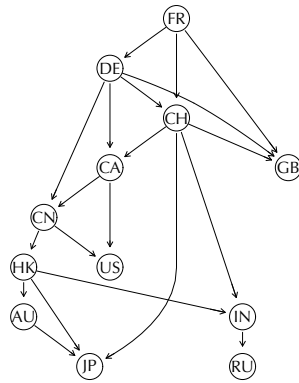
Figure 33: k -fold cross validation with $k = 10$ folds to verify the fit of the algorithms and scoring functions to our data as implemented in Code 2.

We apply both, the score-based and hybrid approaches by averaging $n = 100$ networks of a bootstrap approach: while resampling from the given dataset, we generate a small amount of noise, in order to superimpose spurious correlations. By estimating the confidence in individual arcs and setting their minimum threshold – here 0.7 [4] – we obtain an overall more robust structure with rarely any false positives, as implemented in Code 3. The following networks in Figure 34 are the results, but as they differ in many arcs and orderings, we need to define a way to objectively validate and compare of our different models. [52] [61] [43] [21]

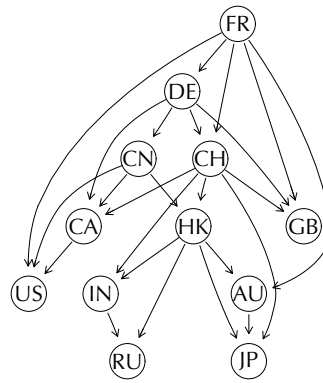
6.4 Evaluating networks from implied volatility data

In order to validate our learned Bayesian networks, we exploit the binary representation of our data and follow the approach described in Section 2.6. We make predictions from our networks using `predict()` and compare them with the test data. In Figure 35, we can see the ROC for each country – or node in our network in this case – depicted for each algorithm for comparability. Thus, to further summarise these evaluations, we calculate the Area Under the Curve (AUC) and populate Table 22 (Appendix), ordered by the sum of all values to rank our models. [35] [17] [46] [59]

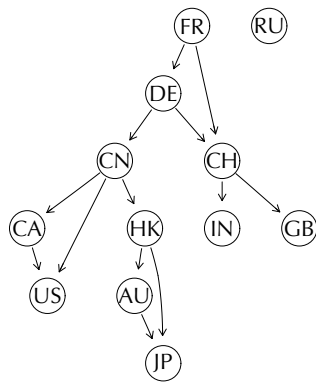
Hill-Climbing – AIC



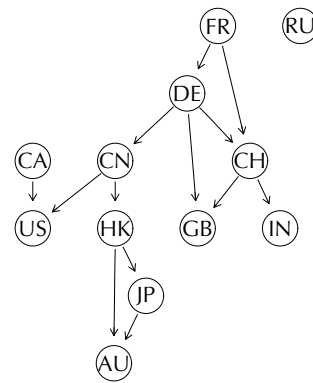
Tabu – AIC



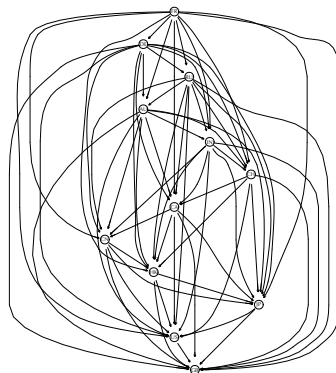
Hill-Climbing – BIC



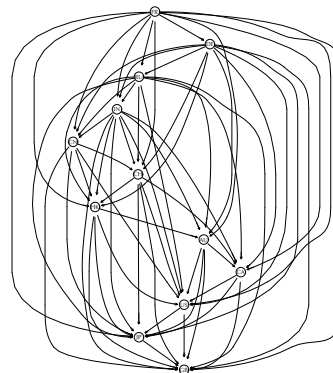
Tabu – BIC



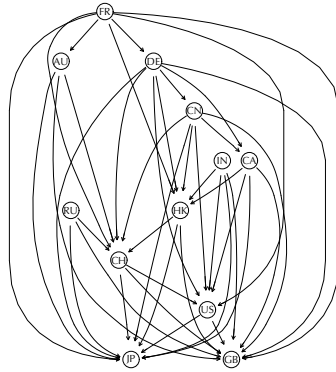
Hill-Climbing – Loglik



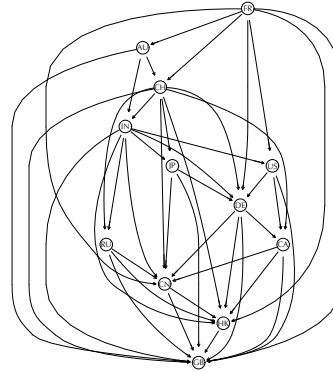
Tabu – Loglik



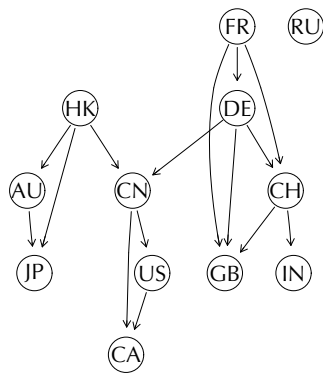
Hill-Climbing – Pred-Loglik



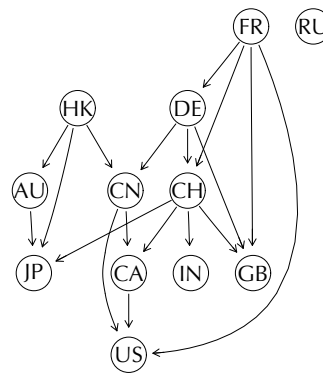
Tabu – Pred-Loglik



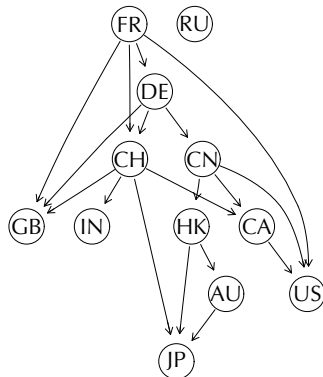
Hill-Climbing – fNML



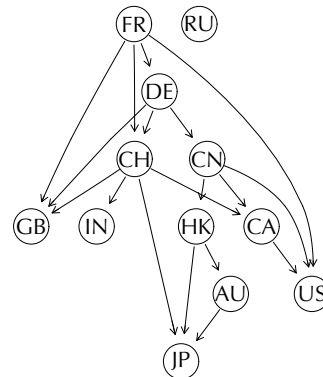
Tabu – fNML



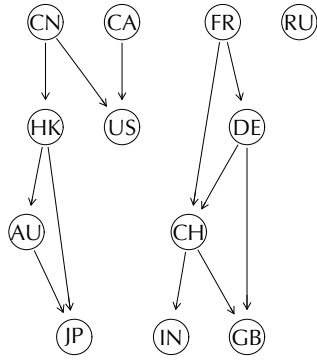
Hill-Climbing – qNML



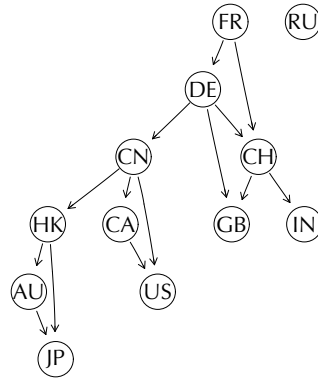
Tabu – qNML



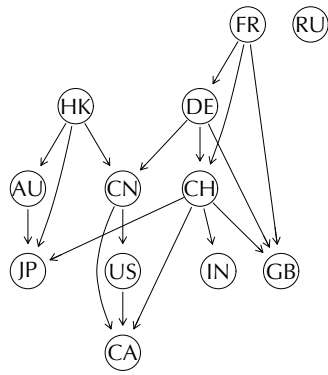
Hill-Climbing – BDe



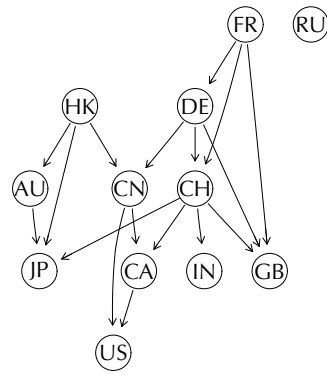
Tabu – BDe



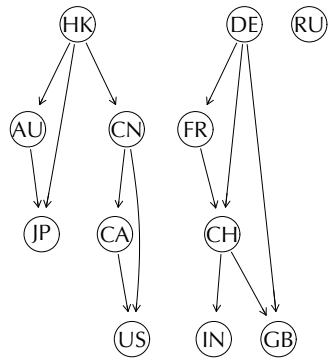
Hill-Climbing – BDj



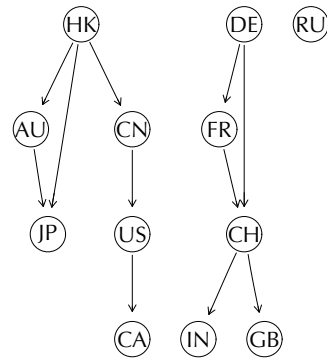
Tabu – BDj



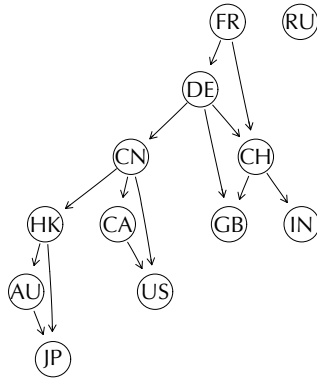
Hill-Climbing – BDLa



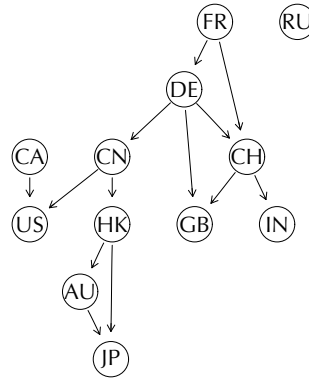
Tabu – BDLa



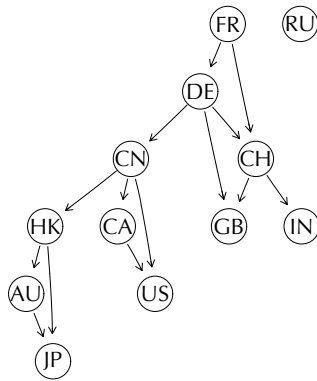
Hill-Climbing - BDs



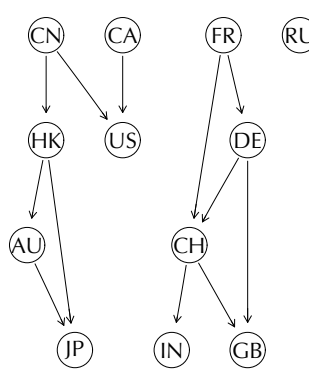
Tabu - BDs



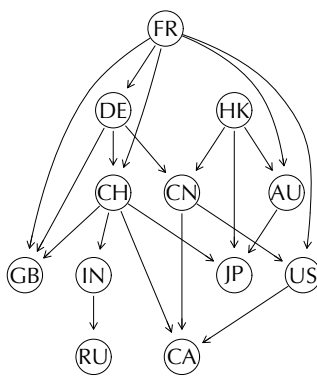
Hill-Climbing - mBDs



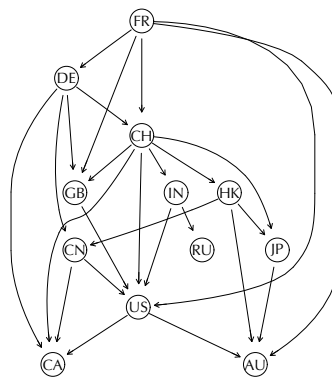
Tabu - mBDs



Hill-Climbing - K2



Tabu - K2



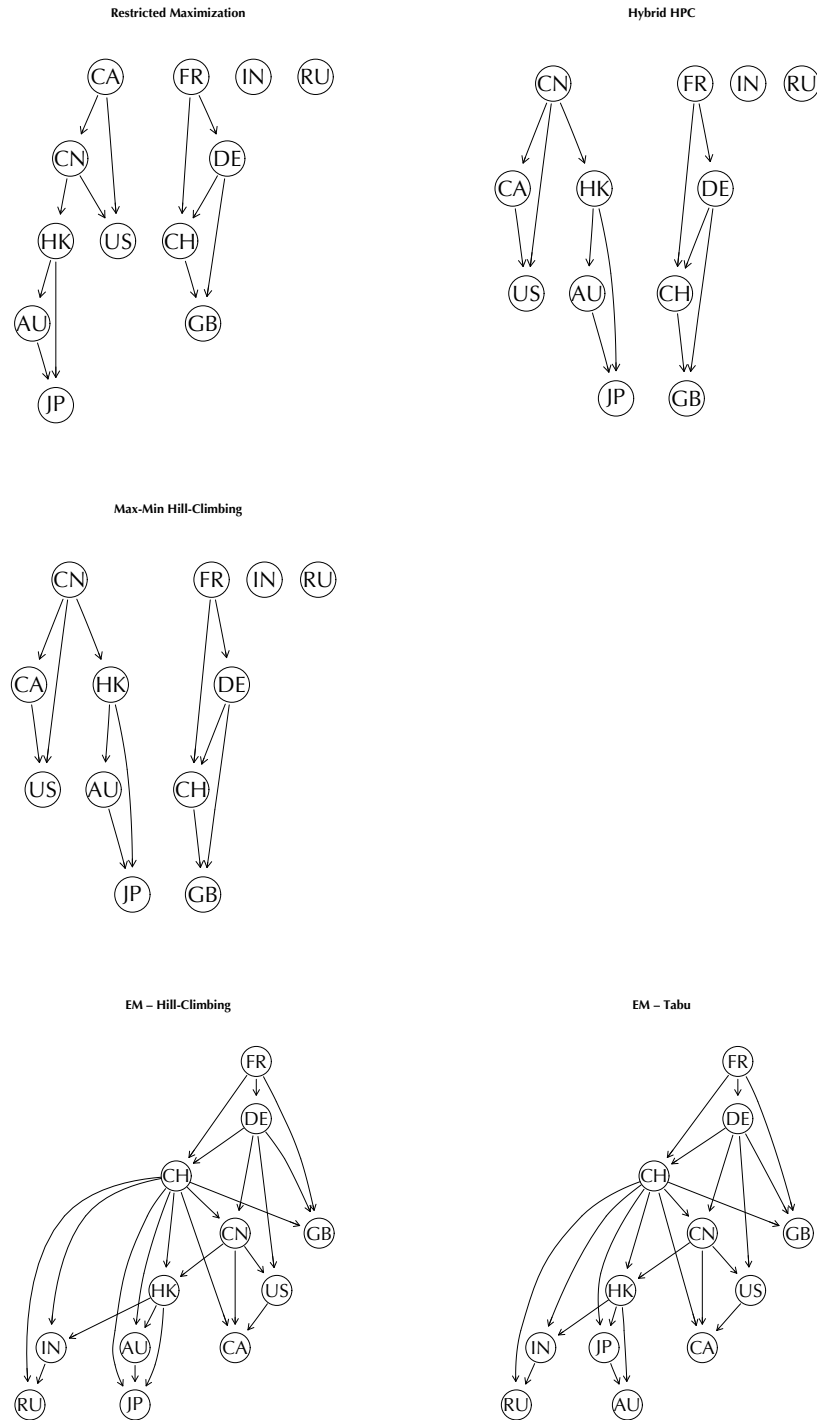
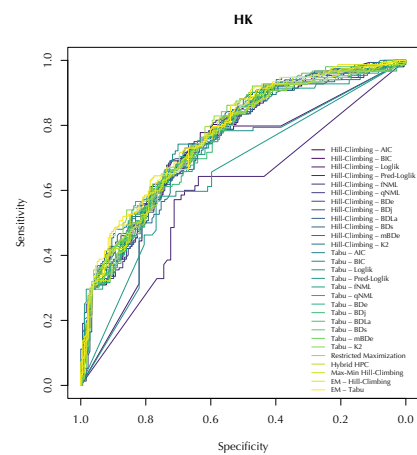
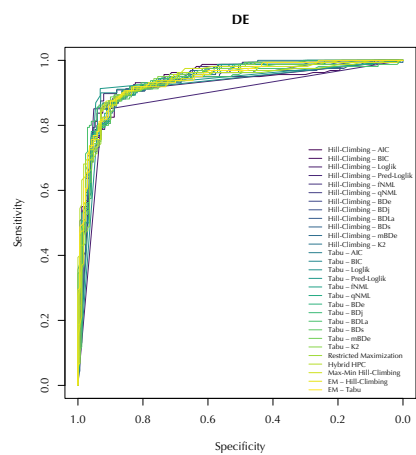
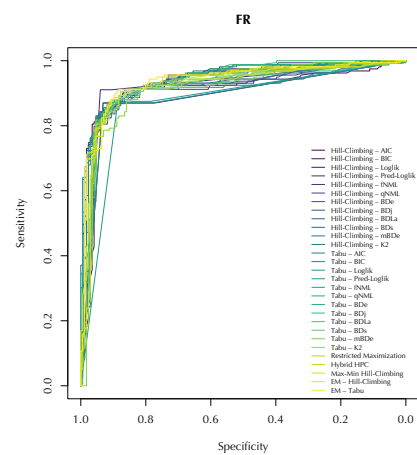
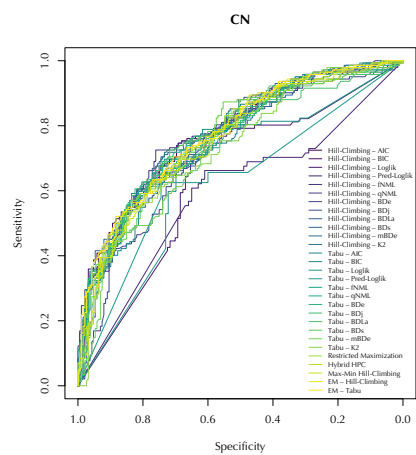
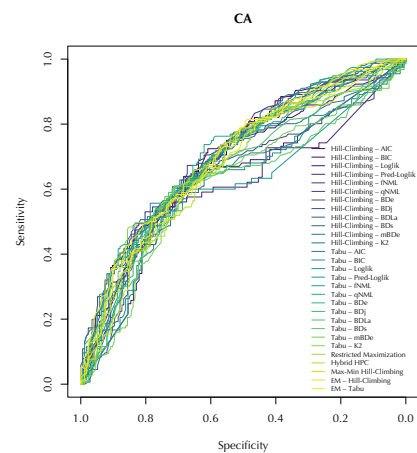
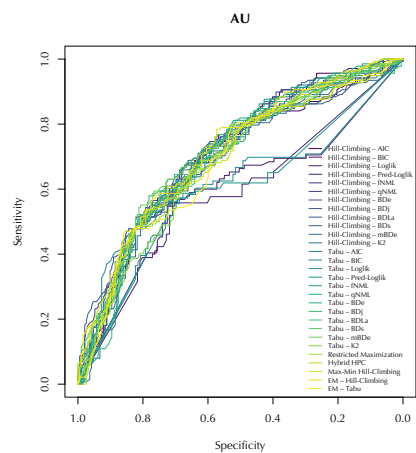


Figure 34: Resulting learned Bayesian networks with dedicated algorithms and scoring functions.



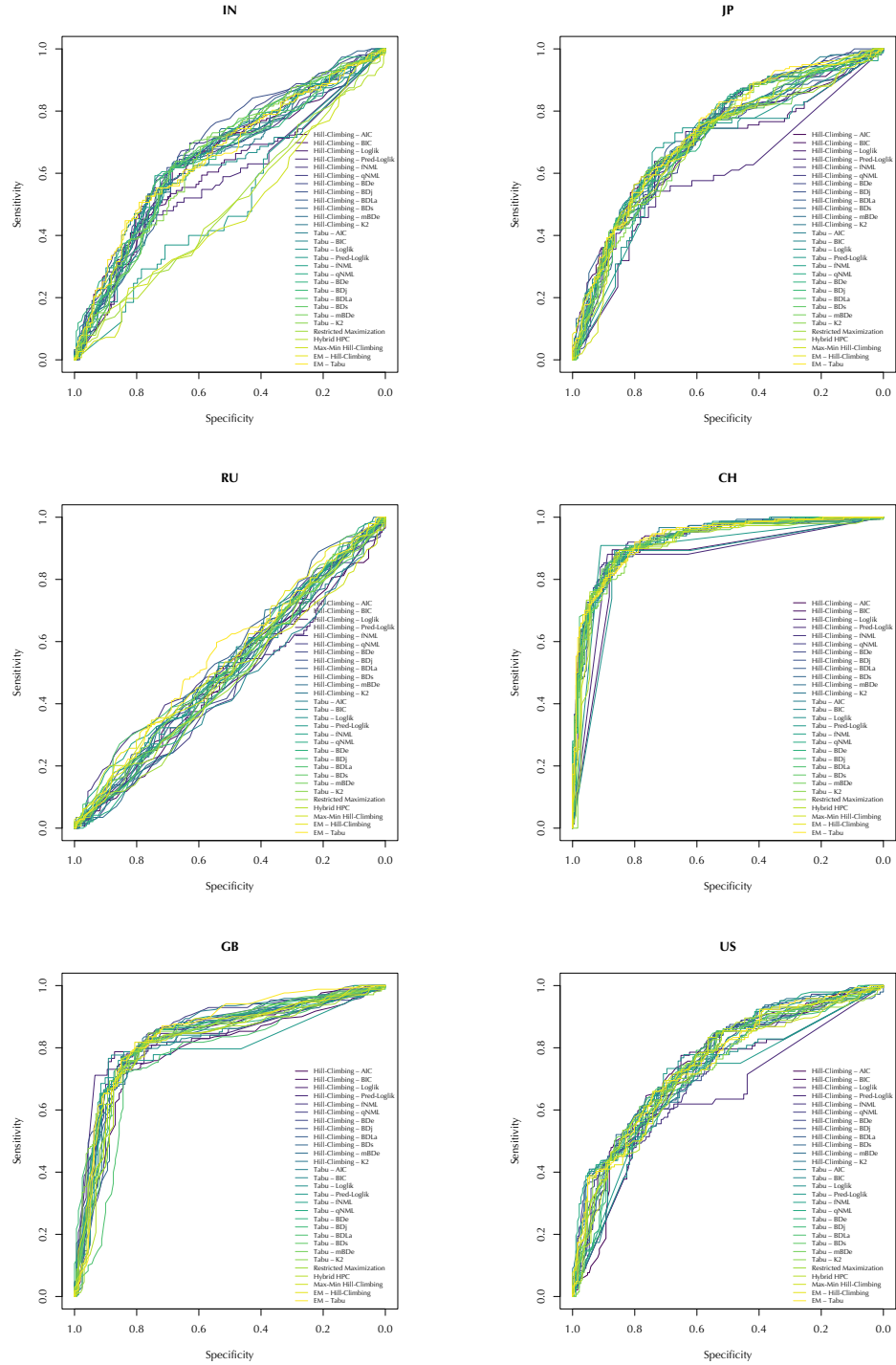


Figure 35: ROC curves of all implemented algorithms per country (node).

As illustrated in Figure 35 and Table 22 (Appendix), most countries demonstrated significant predictive qualities. However, *Canada*, *India*, and especially *Russia* are very close to a random process. We continue our analysis, considering the possibility of dropping them from our data set and thus Bayesian networks.

We proceed by comparing the best performing networks (in terms of probability distributions) and illustrating their overlap, as shown in Figure 36.

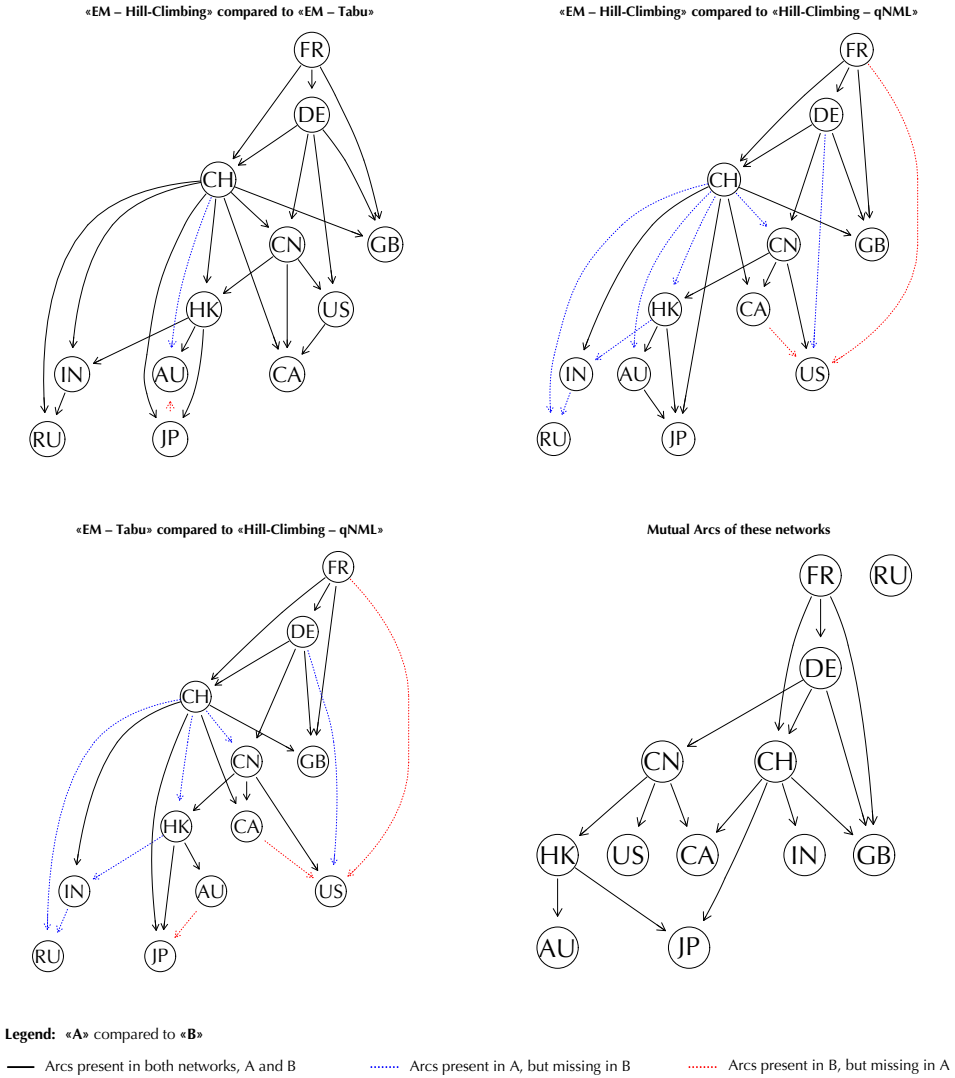


Figure 36: Comparison of the three networks with the highest total AUC scores.

After comparing the Bayesian networks in terms of structural similarities, we now examine the reliability of the predictions and the corresponding conditional probabilities. In Figure 37, we illustrate this confidence using arc strength and show that we can expect reliable distributions for *France*, *Germany*, and *Switzerland* in particular. Table 17 provides an excerpt of some recognisable conditional probabilities.

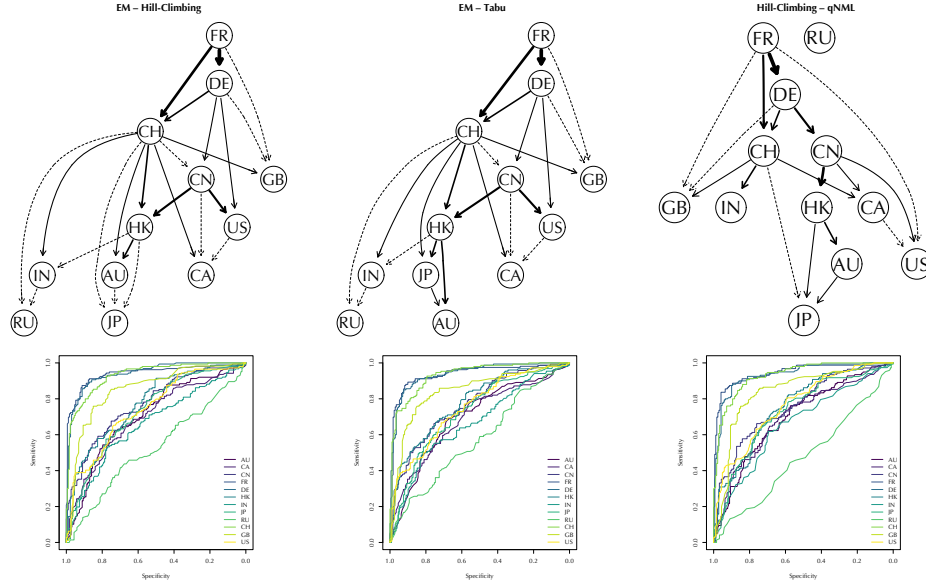


Figure 37: Comparison of the three networks with the highest total AUC scores – confidence depicted as arc strength by Δ BIC scores (see Section 4.6).

Table 17: Excerpt of some recognisable conditional probabilities, conducted with `cpquery`: since the structures are not identical, these values cannot be retrieved directly from the conditional probability tables.

Conditional Probability	EM - Hill-Climbing	EM - Tabu	Hill-Climbing - qNML
$P(\text{Germany} \mid \text{France})$	0.83	0.83	0.86
$P(\text{Switzerland} \mid \text{France}, \text{Germany})$	0.85	0.85	0.83
$P(\neg \text{UK} \mid \neg \text{France}, \neg \text{Germany}, \neg \text{Switzerland})$	0.85	0.85	0.85
$P(\text{Hongkong} \mid \text{China}, \text{Switzerland})$	0.69	0.69	0.59
$P(\text{China} \mid \text{Switzerland}, \text{USA})$	0.74	0.74	0.68
$P(\text{USA} \mid \text{China}, \text{Germany})$	0.69	0.69	0.66

6.5 Causal networks and final result

When we introduced this use case, we talked about the impact of the countries on each other. However, up to this point, we have merely identified significant conditional probabilities – in other words, correlation, not causation. For this reason, we now apply methods of causal discovery from Section 4.7 to infer the actual cause – the desired impact. 56 12 We do not attempt to identify the best causal Bayesian network, but rather a small set of plausible causal Bayesian networks that fit our data. While there are several methods to learn a causal model directly, we aim for a more comparable approach (here ROC from Section 6.4) and thus derive causality from our networks by causal structure search as depicted in our algorithm in Figure 22 and implemented in Code 5 (Appendix). 49 57 33

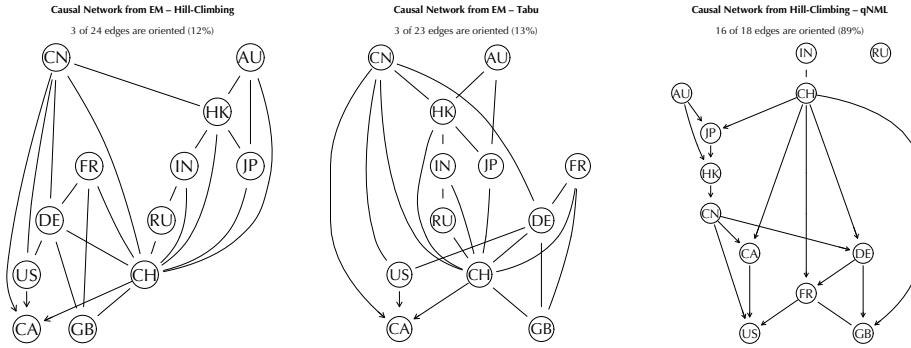


Figure 38: The three causal networks with the highest total AUC scores.

Now we can apply our algorithm from Figure 22 and Code 5 and discuss the results. In Figure 38 our two best performing networks could barely be oriented, only the Hill-Climbing – qNML network has significant orientations. Since we are interested in causal networks with both, many orientations and reliable predictive qualities, we expand our focus to the best quantile in terms of AUC. Thus, the additional causal networks are illustrated in Figure 39. Also for Tabu – qNML we can orient most of the edges and only *India – Switzerland* and *UK – France* is missing a directed arc. Furthermore, as *Russia* is no part of the network and thus no cause for any effects, we can now remove it from our data set, due to its low significance in our ROC analysis. *Canada* and *India* will remain in our data set, as their AUC scores are not necessarily random for our best performing networks with values slightly below or above 0.7 41.

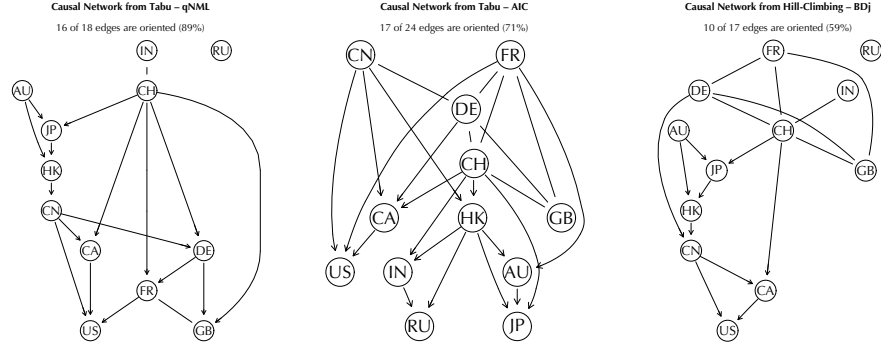


Figure 39: The remaining three causal networks from the highest quantile in terms of AUC scores.

For the two causal networks with the most orientations in Figure 40, it can be argued that *Switzerland* is the financial hub. As described in the Appendix, *China's* underlying is traded in the USA, but tracks Chinese equities traded on the Hongkong stock exchange. This explains the connection between *China*, *USA*, and *Hongkong*. So, without *China* – or with more reliable data on *China's* implied volatility – *Switzerland* divides the network into a western and an eastern hemisphere with dedicated sub-clusters. Thus, the time zones and consequently the tradable hours could be an important generating process for the data. Although we cannot draw many conclusions, *Switzerland's* role as a global financial epicentre seems very intuitive.

«Causal Hill-Climbing – qNML» compared to «Causal Tabu – qNML»

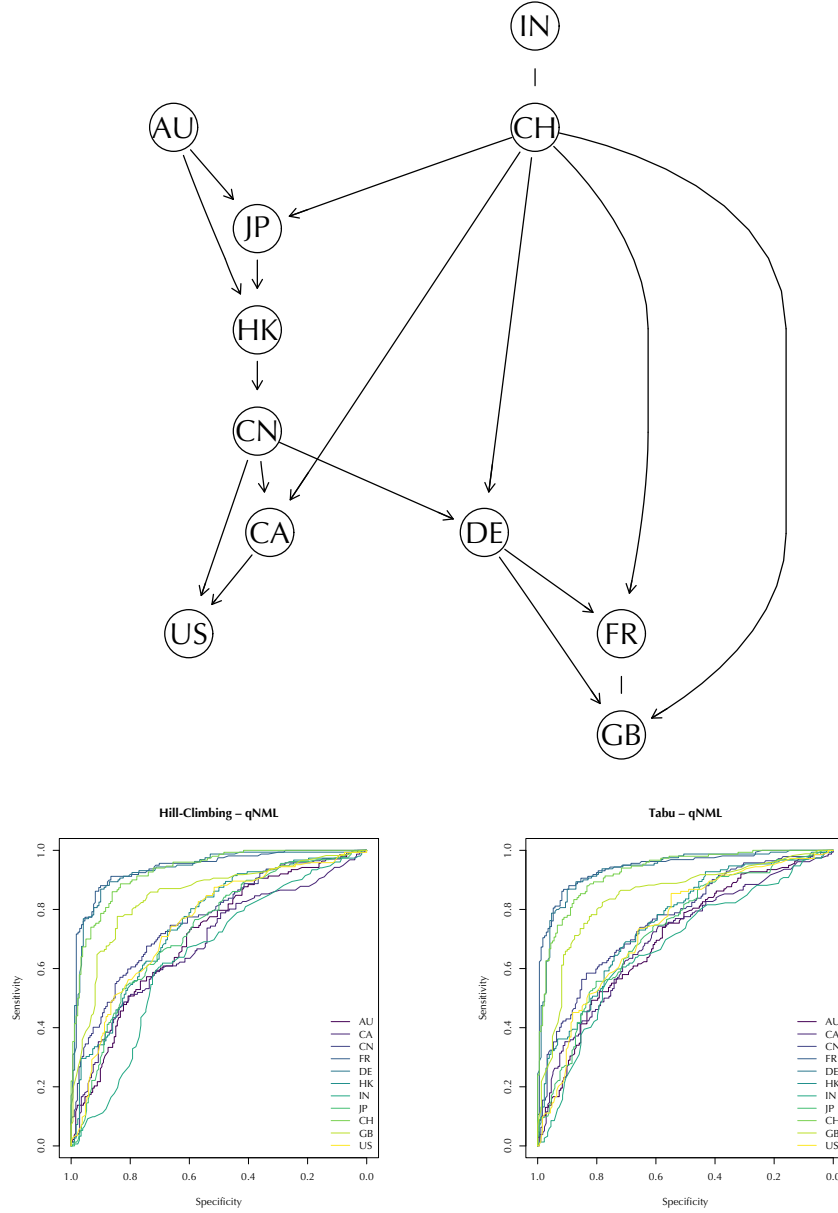


Figure 40: Final causal structure: overlap of our networks with the most orientations in the highest quantile of total AUC.

6.6 Critique

Even if impacts can be discovered without chronological temporal information, causal discovery can only benefit from it. With intraday data this approach would be even more reliable. Also, with respect to *measurement errors*, with more granular data our models would be more resilient, e.g., to the influence of high-frequency traders and other speculators. Currently, we have to assume that our multivariate time series of ten years is invariant. However, since the underlying generating process changes frequently, this likely results in an underfit for our averaged static model. Thus, *dynamic* Bayesian networks could be implemented with a shorter (but denser) time frame and validated by rolling windows and tested against recent unseen data. [57] [33] [52]

In Section [2.4], we established causality on the foundation of intervention and introduced PEARL’s *do*-Operator, but as a thought experiment, this leads to the question of who intervenes in the financial market. Overall, our approach of inferred causation is based on assumptions that are unlikely to apply to our use case and need to be carefully evaluated. We included only the largest global financial players, no other countries, and most importantly, no other variables. On the one hand, the *selection bias* and, on the other hand, the assumption that there are no common confounding factors are very fragile: why should implied volatility not also be influenced by completely different processes? Even if the IC algorithm works with latent variables, we need to include more detailed and comprehensive data from more potential influencers and then implement a dynamic approach. [7] [33] [57]

6.7 Conclusion

Decision making based on observations is an essential element of all scientific disciplines – from the explanation of the generating process behind the data to profound predictions. With two brief examples, we formalised and implemented the EM algorithm for both parameter and structure learning, and provided an approach to realistic problems. We have shown, that even with incomplete or ambiguous data we can reason under uncertainty. Some of our probabilistic models and theories performed better and more efficiently than others. Overall Bayesian networks provide a robust foundation for learning by combining existing knowledge (a-priori probabilities) with observed data (evidence) and offer optimal predictions and an elegant measure to avoid overfitting. In our use case, we showed how to process real-world financial data, applied our obtained knowledge to train models, and utilised their predictions to benchmark the performance. With methods of causal structure discovery, we obtained more knowledge than through mere correlations and we were able to infer the cause – the actual impact of one country on another. Even if more granular data would be desirable, we gained valuable insights into the global financial markets, especially we unveiled the role of Switzerland as financial epicentre.

List of Symbols

Probabilities

P	Probability
\mathbf{P}	Distribution of probabilities
p	Probability density for a continuous variable

Data sets

\mathbf{D}	Data space of observable data (D_1, D_2, \dots, D_j)
\mathbf{d}	Distribution of observed/training data (evidence)
d_j	Observable data point j
\mathbf{Z}	Data space of unobservable data (Z_1, Z_2, \dots, Z_j)
\mathbf{z}	Distribution of unobserved data (latent variables)
z_j	Unobservable data point j
\mathbf{Y}	Data space of all data $\mathbf{Y} = \mathbf{D} \cup \mathbf{Z} (Y_1, Y_2, \dots, Y_j)$
\mathbf{y}	Distribution of all data $\mathbf{y} = \mathbf{d} \cup \mathbf{z}$
x_j	Attribute j

Hypotheses

H	Hypothesis space (containing candidate target functions)
h_i	Hypotheses i
X	Arbitrary random variable
\mathbf{C}	Class space
C	Random variable for classes
c_i	Class i

Other Symbols

μ	Mean
σ	Standard deviation

List of Figures

1	Final results	5
2	Exemplary Bayesian network	9
3	Relevance of topological ordering of nodes	9
4	Interventions in causal Bayesian networks	12
5	ROC explanation	13
6	Posterior probabilities for observations of lime candies	16
7	Prediction for observations of lime candies	17
8	Efficiency of naive Bayes classifier	22
9	Structure of exemplary Bayesian network	23
10	Histograms for naive Bayes classifier	23
11	Bayesian network for the words “ <i>Dear</i> ” and “ <i>Friend</i> ”	24
12	Bayesian network for the words “ <i>Money</i> ” and “ <i>Bayes</i> ”	25
13	Adjusted histograms for naive Bayes classifier	25
14	Comparison of Log Likelihood scores	26
15	Original Bayesian network generating our data with CPTs	28
16	Visualisation of BIC scores	29
17	Comparison of learned and original network	30
18	Visualisation of p-values and Δ BIC values	31
19	d-separation example	32
20	Markov Equivalence classes	33
21	Rules of orientation	34
22	Inductive Causation (IC) algorithm	35
23	Example for incomplete data	37
24	Parameter reduction through additional hidden variable	38
25	Arbitrary data for EM algorithm example	39
26	Structure of exemplary Bayesian network	42
27	Likelihood of the data given the model	44
28	Final Bayesian network learned through EM algorithm	44
29	Daily Implied Volatility	47
30	Change of daily Implied Volatility	48
31	Distribution of binary data	48
32	Unshielded collider	49
33	k -fold cross validation	50
34	Overview of all learned Bayesian networks	55
35	ROC curves of all implemented algorithms per country (node)	57
36	Comparison of networks with highest AUC scores	58
37	Comparison of the impact of the networks with highest AUC scores	59
38	The three causal networks with highest AUC scores	60
39	The remaining three causal networks	61
40	Final causal structure	62

List of Tables

1	Results of CPT products	11
2	Probability of each hypothesis under the data	16
3	Final results of probability of each hypothesis under the data	16
4	Marginalised prediction over the data	17
5	Probability of classification of words “ <i>Dear</i> ” and “ <i>Friend</i> ”	24
6	Probability of classification of words “ <i>Money</i> ” and “ <i>Bayes</i> ”	24
7	Probability of classification of words “ <i>Money</i> ” and “ <i>Bayes</i> ”	25
8	Head of sampled synthetic data set	28
9	Comparison of both algorithms for synthetic data	31
10	Results of likelihood calculations	40
11	Normalised likelihood calculations	40
12	E-step results	41
13	EM Algorithm results per iteration	41
14	Sampled probability table	43
15	Example for options	46
16	Comparison of scoring functions	49
17	Selected conditional probabilities from learned networks	59
18	Results of probability of each hypothesis under the data	68
19	Results of marginalised predictions under the data	68
20	Results of conditional probability queries	69
21	Overview of data set	71
22	Results of AUC – Area under the ROC curves	72
23	Version specification	73

List of Code

1	Conditional probability query sampling	73
2	k -fold cross validation with $k = 10$ folds	75
3	Bootstrap pipeline for averaged models with $n = 100$ runs	77
4	Model evaluation with ROC plots	81
5	Inductive Causation algorithm	83
6	Inductive Causation test	87

Appendix

Excel spreadsheet calculations

Table 18 shows the intermediate results from Section 4.1 of Table 2 calculated in Excel.¹⁶ First, the probability \mathbf{P} of each hypothesis under the data \mathbf{d} was calculated.

Table 18: $\frac{1}{\alpha} \mathbf{P}(h_i | \mathbf{d}) = P(h_i) \prod_j P(d_j | h_i)$

Hypothesis	h_1	h_2	h_3	h_4	h_5	Marginalisation for
$P(h_i)$	0.10	0.20	0.40	0.20	0.10	Normalisation
$P(lime)$	0.00	0.25	0.50	0.75	1.00	
$\mathbf{d} = d_1$	0.00	0.05	0.20	0.15	0.10	0.50
$\mathbf{d} = d_2$	0.00	0.01	0.10	0.11	0.10	0.33
$\mathbf{d} = d_3$	0.00	0.00	0.05	0.08	0.10	0.24
$\mathbf{d} = d_4$	0.00	0.00	0.03	0.06	0.10	0.19
$\mathbf{d} = d_5$	0.00	0.00	0.01	0.05	0.10	0.16
$\mathbf{d} = d_6$	0.00	0.00	0.01	0.04	0.10	0.14
$\mathbf{d} = d_7$	0.00	0.00	0.00	0.03	0.10	0.13
$\mathbf{d} = d_8$	0.00	0.00	0.00	0.02	0.10	0.12
$\mathbf{d} = d_9$	0.00	0.00	0.00	0.02	0.10	0.12
$\mathbf{d} = d_{10}$	0.00	0.00	0.00	0.01	0.10	0.11

In Table 19 we obtain the final result – after multiplying each value by the dedicated normalisation constants – by calculating the prediction X over the data \mathbf{d} and marginalising as described in Table 4.

Table 19: $\mathbf{P}(X | \mathbf{d}) = \sum_i \mathbf{P}(X | h_i) \mathbf{P}(h_i | \mathbf{d})$

Hypothesis	h_1	h_2	h_3	h_4	h_5	Marginalisation for
$P(h_i)$	0.10	0.20	0.40	0.20	0.10	Final result
$P(lime)$	0.00	0.25	0.50	0.75	1.00	
$\mathbf{d} = d_0$	0.00	0.05	0.20	0.15	0.10	0.50
$\mathbf{d} = d_1$	0.00	0.03	0.20	0.23	0.20	0.65
$\mathbf{d} = d_2$	0.00	0.01	0.15	0.26	0.31	0.73
$\mathbf{d} = d_3$	0.00	0.00	0.11	0.27	0.42	0.80
$\mathbf{d} = d_4$	0.00	0.00	0.07	0.25	0.53	0.85
$\mathbf{d} = d_5$	0.00	0.00	0.04	0.22	0.62	0.89
$\mathbf{d} = d_6$	0.00	0.00	0.02	0.19	0.70	0.92
$\mathbf{d} = d_7$	0.00	0.00	0.01	0.15	0.77	0.94
$\mathbf{d} = d_8$	0.00	0.00	0.01	0.12	0.82	0.95
$\mathbf{d} = d_9$	0.00	0.00	0.00	0.10	0.86	0.96
$\mathbf{d} = d_{10}$	0.00	0.00	0.00	0.08	0.90	0.97

¹⁶The significant figures in this and all other calculations are based on HIGHAM [38] and on our underlying information about a-priori probabilities and sample sizes.

Conditional probability queries

Table 20: The conditional probability queries were conducted with `cpquery`, with $n = 100.000$ observations for each the original Bayesian network (BN) and the learned one. The query values are an average from 25 individual samples (executions). Deviation is calculated as follows: $\frac{\text{learned} - \text{original}}{\text{original}}$

Conditional Probability	Original BN	Learned BN	Deviation (in %)
$P(\text{tub} \mid \text{asia})$	0.048	¹⁷ 0.010	-80.28%
$P(\text{tub} \mid \neg \text{asia})$	0.010	¹⁸ 0.009	-13.77%
$P(\text{lung} \mid \text{smoke})$	0.100	0.102	1.89%
$P(\text{lung} \mid \neg \text{smoke})$	0.010	0.011	13.43%
$P(\text{bronc} \mid \text{smoke})$	0.600	0.606	1.07%
$P(\text{bronc} \mid \neg \text{smoke})$	0.300	0.301	0.33%
$P(\text{either} \mid \text{tub}, \text{lung})$	1.000	1.000	0.00%
$P(\text{either} \mid \text{tub}, \neg \text{lung})$	1.000	1.000	0.00%
$P(\text{either} \mid \neg \text{tub}, \text{lung})$	1.000	1.000	0.00%
$P(\text{either} \mid \neg \text{tub}, \neg \text{lung})$	0.000	0.000	0.00%
$P(\text{dysp} \mid \text{either}, \text{bronc})$	0.900	0.895	-0.53%
$P(\text{dysp} \mid \text{either}, \neg \text{bronc})$	0.702	0.705	0.48%
$P(\text{dysp} \mid \neg \text{either}, \text{bronc})$	0.800	0.816	2.02%
$P(\text{dysp} \mid \neg \text{either}, \neg \text{bronc})$	0.100	0.091	-9.51%
$P(\text{xray} \mid \text{either})$	0.981	0.979	-0.16%
$P(\text{xray} \mid \neg \text{either})$	0.050	0.053	5.49%

¹⁷Due to the missing arc, instead of sampling a conditional probability $P(\text{tub} \mid \text{asia}) \approx 0.050$ `cpquery` estimated $P(\text{tub}) \approx 0.010$.

¹⁸See footnote above. Deviation is less significant due to $P(\text{tub} \mid \neg \text{asia}) \approx P(\text{tub}) \approx 0.010$.

Raw data

The raw data has been acquired from investing.com and onvista.de [accessed 27-April-2022]. Unfortunately, only daily time frames were available. The utilised data is 30-day implied volatility, calculated based on near-term and next-term option prices. [1] [19] As expected the VXFXI is implicit in the prices of its underlying FXI (iShares Trust FTSE China 25 Index Fund) options. However, the anomaly here is, that FXI – an instrument traded in NYSE Arca, New York – tracks Chinese equities traded on the Hongkong Stock Exchange and the VXFXI is traded at CBOE, Chicago. This will most certainly not lead to perfect portrayal of China’s implied volatility. [6] [19] Also the tradable hours should be treated with caution. Many derivatives exchanges offer night sessions or OTC trades to their customers. However, since most of the (transparent) total volume is traded during the opening and closing periods of regular trading hours, our modelling assumption is therefore supported. [34] The data from Table [21] give a total of 2724 observations, and after omitting rows with NA values, 1017 remain. Since the data is split with the ratio $\frac{1}{3}$ and $\frac{2}{3}$, the training set for the EM algorithm consists of 1816 rows and the remaining algorithms are trained with a set of 687 rows. Both are tested with a set of 330.

¹⁸Tradable hours pulled from dedicated website of exchange.

¹⁹Sometimes referred to as AXVI.

²⁰Sometimes referred to as Nikkei 225 VI.

Table 21: Mapping of implied volatility data and their underlying asset to dedicated countries, including stock exchanges, tradable hours and available time frame with sources.

Index ticker	Associated country	Short form	Underlying assets	Traded exchange	Tradable hours (UTC) ¹⁹	Available timeframe	Data source [accessed 27-April-2022]
A-VIX ^a	Australia	AU	S&P/ASX 200	ASX (Sydney)	20:00 – 02:00	Jan 2012 – Dec 2021	https://www.investing.com 22
VIXI	Canada	CA	S&P/TSX 60	TSX (Toronto)	04:30 – 11:00	Jan 2012 – Dec 2021	https://www.investing.com 31
VXFXI	China	CN	FXI	CBOE (Chicago)	04:30 – 11:00	Jan 2012 – Dec 2021	https://www.investing.com 32
VCAC	France	FR	CAC 40	Euronext Paris	11:01 – 19:30	Jan 2012 – Dec 2021	https://www.investing.com 26
VDAX-NEW	Germany	DE	DAX	Deutsche Börse (Frankfurt)	10:00 – 18:30	Jan 2012 – Dec 2021	https://www.investing.com 27
VHSI	Hongkong	HK	HSI	HKEX (Hongkong)	17:30 – 00:00	Jan 2012 – Dec 2021	https://www.investing.com 29
NIFVIX	India	IN	NIFTY	NSE (Mumbai)	14:45 – 21:00	Jan 2012 – Dec 2021	https://www.investing.com 24
JNIV ^b	Japan	JP	Nikkei	Osaka Exchange	18:00 – 00:15	Jan 2012 – Dec 2021	https://www.investing.com 23
RVI	Russia	RU	RTSI	MOEX (Moscow)	13:00 – 21:45	Nov 2013 – Dec 2021	https://www.investing.com 25
VSMI	Switzerland	CH	VMI	SIX Swiss Exchange (Zurich)	11:00 – 19:30	Jan 2012 – Dec 2021	https://www.onvista.de/ 55
VFTSE	United Kingdom	GB	FTSE 100	Euronext Amsterdam	11:01 – 19:30	Aug 2012 – Jun 2019	https://www.investing.com 28
VIX	United States	US	S&P500	CBOE (Chicago)	04:30 – 11:00	Jan 2012 – Dec 2021	https://www.investing.com 30

Table 22: Area under the ROC curves – ranked highest to lowest.

Algorithm	Australia	Canada	China	France	Germany	Hongkong	India	Japan	Russia	Switzerland	UK	USA	Σ AUC
EM – Hill-Climbing	0.72	0.70	0.78	0.94	0.94	0.78	0.66	0.73	0.55	0.93	0.84	0.74	9.31
EM – Tabu	0.69	0.69	0.77	0.94	0.94	0.78	0.66	0.73	0.56	0.93	0.85	0.75	9.30
Hill-Climbing – qNML	0.71	0.71	0.77	0.95	0.93	0.76	0.67	0.73	0.49	0.93	0.86	0.76	9.28
Tabu – qNML	0.71	0.71	0.77	0.95	0.94	0.76	0.67	0.72	0.50	0.93	0.84	0.77	9.26
Tabu – AIC	0.71	0.71	0.76	0.94	0.94	0.77	0.65	0.73	0.51	0.93	0.85	0.77	9.26
Hill-Climbing – BDj	0.71	0.70	0.77	0.94	0.94	0.75	0.68	0.73	0.51	0.93	0.84	0.74	9.25
Tabu – FNML	0.71	0.71	0.76	0.95	0.94	0.75	0.64	0.73	0.53	0.93	0.84	0.76	9.23
Hill-Climbing – K2	0.69	0.70	0.77	0.94	0.94	0.74	0.66	0.73	0.53	0.93	0.85	0.75	9.23
Hill-Climbing – AIC	0.71	0.71	0.78	0.94	0.94	0.76	0.66	0.73	0.50	0.93	0.84	0.72	9.21
Hill-Climbing – FNML	0.70	0.70	0.78	0.94	0.94	0.76	0.65	0.70	0.55	0.93	0.85	0.72	9.21
Tabu – BDj	0.71	0.69	0.77	0.95	0.94	0.75	0.65	0.72	0.50	0.93	0.85	0.72	9.18
Hill-Climbing – BDs	0.70	0.69	0.77	0.91	0.94	0.77	0.67	0.71	0.51	0.92	0.82	0.74	9.14
Hill-Climbing – BDLa	0.71	0.67	0.76	0.92	0.93	0.76	0.63	0.71	0.54	0.93	0.83	0.75	9.13
Tabu – K2	0.68	0.70	0.74	0.92	0.94	0.76	0.65	0.71	0.52	0.91	0.84	0.73	9.11
Tabu – BDs	0.71	0.63	0.76	0.92	0.94	0.77	0.64	0.72	0.51	0.92	0.82	0.76	9.10
Tabu – BDe	0.69	0.67	0.77	0.92	0.94	0.77	0.64	0.71	0.51	0.93	0.82	0.73	9.09
Hill-Climbing – mBDe	0.68	0.66	0.78	0.92	0.93	0.77	0.64	0.71	0.48	0.92	0.82	0.75	9.06
Tabu – BDLa	0.70	0.65	0.75	0.92	0.93	0.76	0.66	0.70	0.52	0.93	0.76	0.75	9.05
Tabu – mBDe	0.71	0.62	0.74	0.91	0.94	0.77	0.66	0.69	0.50	0.93	0.83	0.74	9.03
Hill-Climbing – BIC	0.69	0.67	0.77	0.91	0.93	0.76	0.63	0.69	0.48	0.93	0.78	0.76	9.01
Hill-Climbing – BDe	0.70	0.65	0.73	0.91	0.94	0.76	0.67	0.69	0.47	0.92	0.82	0.74	8.99
Tabu – BIC	0.71	0.63	0.76	0.91	0.94	0.77	0.65	0.70	0.45	0.92	0.82	0.72	8.97
Hybrid HPC	0.70	0.67	0.76	0.93	0.94	0.76	0.49	0.70	0.51	0.91	0.81	0.73	8.91
Restricted Maximization	0.69	0.68	0.76	0.92	0.94	0.76	0.50	0.69	0.48	0.91	0.81	0.74	8.90
Max-Min Hill-Climbing	0.69	0.68	0.76	0.92	0.94	0.76	0.48	0.69	0.49	0.92	0.82	0.74	8.90
Tabu – Loglik	0.60	0.63	0.66	0.89	0.92	0.69	0.60	0.66	0.50	0.86	0.82	0.70	8.53
Hill-Climbing – Loglik	0.59	0.63	0.65	0.89	0.91	0.69	0.59	0.65	0.50	0.87	0.82	0.70	8.48
Tabu – Pred-Loglik	0.59	0.61	0.64	0.87	0.93	0.66	0.51	0.68	0.52	0.90	0.79	0.69	8.40
Hill-Climbing – Pred-Loglik	0.59	0.62	0.59	0.92	0.90	0.59	0.59	0.61	0.48	0.87	0.83	0.64	8.23
Average	0.69	0.67	0.74	0.92	0.93	0.75	0.63	0.70	0.51	0.92	0.83	0.73	9.03

R Code (RStudio)

Table 23: Utilised packages and their versions.

Package	Version
		kableExtra	1.3.4
Base R	4.2.0	pROC	1.18.0
BiocGenerics	0.41.2	purrr	0.3.4
bnlearn	4.7.1	readr	2.1.2
dplyr	1.0.8	reshape2	1.4.4
forcats	0.5.1	Rgraphviz	2.39.1
ggplot2	3.3.5	scales	1.2.0
gRain	1.3.9	stringr	1.2.0
graph	1.73.0	tibble	3.1.6
gRbase	1.8.7	tidyr	1.2.0
gttools	3.9.2	tidyverse	1.3.1
...	...	viridis	0.6.2

Code 1: The conditional probability queries were conducted with `cpquery`, with $n = 100.000$ observations for each the original Bayesian network and the learned one, and then averaged from 25 individual samples executions. Finally the deviation is calculated (see Table 20).

```

1 arcstr.t.p <- arc.strength(tabu(asia, score = "bic"), asia,
   criterion = "x2")
2 strength.plot(tabu(asia, score = "bic"), arcstr.t.p)
3
4 arcstr.t.b <- arc.strength(tabu(asia, score = "bic"), asia)
5 strength.plot(tabu(asia, score = "bic"), arcstr.t.b)
6
7
8 # Merge all information and output to LaTeX
9 mergedtables <-
10   merge(
11     merge(arcstr.h.p, arcstr.h.b, by = c("from", "to")),
12     merge(arcstr.t.p, arcstr.t.b, by = c("from", "to")),
13     by = c("from", "to")
14   )
15 colnames(mergedtables) <-
16   c("from", "to", "p-value HC", "BIC HC", "p-value Tabu", "
   BIC Tabu")
17 kable(mergedtables,
18       digits = 4,
19       booktabs = T,
```

```

20     format = "latex")
21
22 # Compare with intended Graph
23 dag_learned <- tabu(asia, score = "bic")
24 compare(dag_learned, bn.net(original))
25 #true positive (tp) arcs — appearing both in target and in current
26 #false positive (fp) arcs — appearing in current but not in target
27 #false negative (fn) arcs — appearing in target but not in current
28 graphviz.compare(dag_learned, bn.net(original))
29
30 # Conditional Probabilities by Query Sampling — 100000 observations
    for each of the 25 individual samples (executions)
31
32 df <-
33     data.frame(
34         event = c(
35             '(tub=="yes")',
36             '(tub=="yes")',
37             '(lung=="yes")',
38             '(lung=="yes")',
39             '(bronc=="yes")',
40             '(bronc=="yes")',
41             '(either=="yes")',
42             '(either=="yes")',
43             '(either=="yes")',
44             '(either=="yes")',
45             '(dysp=="yes")',
46             '(dysp=="yes")',
47             '(dysp=="yes")',
48             '(dysp=="yes")',
49             '(xray=="yes")',
50             '(xray=="yes")',
51         ),
52         evidence = c(
53             '(asia=="yes")',
54             '(asia=="no")',
55             '(smoke=="yes")',
56             '(smoke=="no")',
57             '(smoke=="yes")',
58             '(smoke=="no")',
59             '(tub=="yes" & lung=="yes")',
60             '(tub=="yes" & lung=="no")',
61             '(tub=="no" & lung=="yes")',
62             '(tub=="no" & lung=="no")',
63             '(either=="yes" & bronc=="yes")',
64             '(either=="yes" & bronc=="no")',
65             '(either=="no" & bronc=="yes")',
66             '(either=="no" & bronc=="no")',
67             '(either=="yes")',
68             '(either=="no")',
69         )
70     )
71 # loop through all rows of our df with events and evidence and
    sample each the original and the learned BN for best comparison
72 for (i in 1:nrow(df)) {
73     # use text for the function cpquery, as it has some bugs
74

```

```

75     # original DAG first
76     pastetext = paste("cpquery(original, ",
77                       df$event[i],
78                       ", ",
79                       df$evidence[i],
80                       ",n=100000",
81                       ")")
82     # Average the result of 100.000 observations with 25
      executions to gather a stable final result
83     my.sum = 0
84     number.nodes = 25
85
86     for (j in 1:number.nodes) {
87         current = eval(parse(text = pastetext))
88         my.sum = my.sum + current
89     }
90     # new column for results
91     df$original[i] = my.sum / number.nodes
92
93     # learned DAG second (tabusearch)
94     pastetext = paste("cpquery(tabusearch, ",
95                       df$event[i],
96                       ", ",
97                       df$evidence[i],
98                       ",n=100000",

```

Code 2: k -fold cross validation with $k = 10$ to verify the fit of the algorithms and scoring functions to our data. Due to the runtime, a progress bar is incorporated.

```

1  validate_scores <-
2    function(complete.data,
3             k,
4             score.algorithms,
5             scores,
6             hybrid.algorithms) {
7      #empty global lists
8      name.list <- list()
9      loss.list <- list()
10
11     #Progress bar as the function takes some time
12     p <- 0
13     pb <- txtProgressBar(
14       min = 0,
15       max = length(score.algorithms) * length(scores) +
16         length(hybrid.algorithms),
17       style = 3,
18       width = 50,
19       char = "="
20     )
21
22     #iterate through all algorithms and scores
23     n <- 0
24     for (a in score.algorithms) {

```

```

25 n <- n + 1
26 for (s in scores) {
27   #pred log like score on separated test set, so it needs a
    separated case
28   if (s == "Pred-Loglik") {
29     l11 <-
30       sample(1:nrow(complete.data),
31             nrow(complete.data) / 3)
32     l12 <- -l11
33     l11_data <-
34       as.data.frame(lapply(complete.data[l11,], as.factor))
35     l12_data <-
36       as.data.frame(lapply(complete.data[l12,], as.factor))
37     #new test train split
38     out <- paste0(score.algorithms.names[n], " - ", s)
39     name.list <- append(name.list, out)
40     #permutation list
41     #k-fold cross validation
42     cv.res <-
43       bn.cv(
44         l11_data,
45         bn = tolower(a),
46         loss = "logl",
47         fit = "mle",
48         runs = k,
49         algorithm.args = list(
50           score = tolower(s),
51           newdata = l12_data,
52           optimized = TRUE
53         )
54       )
55     #save the loss as a list
56     Log.Like.Loss <- list(c(loss(cv.res)))
57     loss.list <- append(loss.list, Log.Like.Loss)
58     p <- p + 1
59     setTxtProgressBar(pb, p)
60
61   }
62   else{
63     out <- paste0(score.algorithms.names[n], " - ", s)
64     name.list <- append(name.list, out)
65     #permutation list
66     #k-fold cross validation
67     cv.res <-
68       bn.cv(
69         complete.data,
70         bn = tolower(a),
71         loss = "logl",
72         fit = "mle",
73         runs = k,
74         algorithm.args = list(score = tolower(s))
75       )
76     #save the loss as a list
77     Log.Like.Loss <- list(c(loss(cv.res)))
78     loss.list <- append(loss.list, Log.Like.Loss)
79     p <- p + 1
80     setTxtProgressBar(pb, p)

```

```

81     }
82   }
83 }
84 }
85
86 #Hybrid
87 n <- 0
88 for (a in hybrid.algorithms) {
89   n <- n + 1
90   out <- paste0(hybrid.algorithms.names[n])
91   name.list <- append(name.list, out)
92   #permutation list
93   #k-fold cross validation
94   cv.res <- bn.cv(
95     complete.data,
96     bn = tolower(a),
97     loss = "logl",
98     fit = "mle",
99     # "bayes"
100    runs = k
101  )
102  #save the loss as a list
103  Log.Like.Loss <- list(c(loss(cv.res)))
104  loss.list <- append(loss.list, Log.Like.Loss)
105  p <- p + 1
106  setTxtProgressBar(pb, p)
107
108 }
109 #save the lists of loss as dataframe and return
110 names <- name.list[seq_len(length(name.list))]
111 df <- data.frame(unlist(names), unlist(loss.list))
112 names(df) <- c("Algorithm", "Loss")
113 close(pb)
114 return(df)
115 }

```

Code 3: Bootstrap approach to implement pipeline for averaged models with $n = 100$ runs including a progress bar, due to the long runtime.

```

1 build_models <-
2   function(incomplete.training_data,
3     complete.training_data,
4     n,
5     threshold,
6     score.algorithms,
7     scores,
8     hybrid.algorithms) {
9
10    #empty list to save all possible permutations
11    permutation.list <- list()
12    #Progressbar as this function takes significant time
13    p <- 0
14    t <-

```

```

15 pb <- txtProgressBar(min = 0,
16                       max = length(score.algorithms)*length(
                                scores)+length(hybrid.algorithms)+
                                length(score.algorithms),
17                       style = 3,
18                       width = 50,
19                       char = "=")
20 i <- 0
21 for (a in score.algorithms) {
22   i <- i + 1
23   #iterate through all algorithms and scores
24   for (s in scores) {
25     #pred log like score on separated test set, so it needs a
      separated case
26     if (s == "Pred-Loglik") {
27       ll1 <-
28         sample(1:nrow(complete.training_data),
29               nrow(complete.training_data) / 3)
30       ll2 <- -ll1
31       ll1_data <-
32         as.data.frame(lapply(complete.training_data[ll1, ], as.
                                factor))
33       ll2_data <-
34         as.data.frame(lapply(complete.training_data[ll2, ], as.
                                factor))
35       #new test train split
36       out <- paste0(score.algorithms.names[i], " - ", s)
37       #name of plot
38       permutation.list <- append(permutation.list, out)
39       #permutation list
40       #boot strap approach
41       res.boot.strength = boot.strength(
42         ll1_data,
43         R = n,
44         algorithm = tolower(a),
45         algorithm.args = list(
46           score = tolower(s),
47           newdata = ll2_data,
48           optimized = TRUE
49           #optimized for faster results (reuse sampled scores)
50         ),
51         cpdag = FALSE
52         # we do not care about the equivalence class YET. Thus
           we want a bayesian network
53       )
54       #averaged network from bootstrapped approach
55       res <-
56         averaged.network(res.boot.strength, threshold =
           threshold)
57       arcs(res) <- directed.arcs(res)
58       #assign arc strength and network
59       assign(paste0("fitted.", gsub("[:space:]", "", out)),
60             bn.fit(res, ll1_data),
61             envir = parent.frame())
62       assign(
63         paste0("arcstr.", gsub("[:space:]", "", out)),
64         arc.strength(res, ll1_data, criterion = "bic"),

```



```

65     envir = parent.frame()
66   )
67   p <- p+1
68   setTxtProgressBar(pb, p)
69
70 }
71
72 else{
73   out <- paste0(score.algorithms.names[i], " - ", s)
74   permutation.list <- append(permutation.list, out)
75   #permutation list
76   #boot strap approach
77   res.boot.strength = boot.strength(
78     complete.training_data,
79     R = n,
80     algorithm = tolower(a),
81     algorithm.args = list(score = tolower(s), optimized =
      TRUE),
82     #optimized for faster results (reuse sampled scores)
83     cpdag = FALSE
84     # we do not care about the equivalence class YET. Thus
      we want a bayesian network
85   )
86   #averaged network from bootstrapped approach
87   res <-
88     averaged.network(res.boot.strength, threshold =
      threshold)
89   #assign arc strength and network
90   arcs(res) <- directed.arcs(res) # ignore undirected arcs
91   assign(paste0("fitted.", gsub("[:space:]", "", out)),
92     bn.fit(res, complete.training_data),
93     envir = parent.frame())
94   assign(
95     paste0("arcstr.", gsub("[:space:]", "", out)),
96     arc.strength(res, complete.training_data, criterion = "
      bic"),
97     envir = parent.frame()
98   )
99   p <- p+1
100   setTxtProgressBar(pb, p)
101 }
102 }
103 }
104
105 #Hybrid
106 i <- 0
107 for (a in hybrid.algorithms) {
108   i <- i + 1
109   out <- paste0(hybrid.algorithms.names[i])
110   permutation.list <- append(permutation.list, out)
111   #permutation list
112   #boot strap approach
113   res.boot.strength = boot.strength(
114     complete.training_data,
115     R = n,
116     algorithm = tolower(a),
117     cpdag = FALSE

```

```

118         # we do not care about the equivalence class YET. Thus we
119         # want a bayesian network
120     )
121     #averaged network from bootstrapped approach
122     res <- averaged.network(res.boot.strength, threshold =
123     threshold)
124     arcs(res) <- directed.arcs(res) # ignore undirected arcs
125     #assign arc strength and network
126     assign(paste0("fitted."), gsub("[:space:]", "", out)),
127     bn.fit(res, complete.training_data),
128     envir = parent.frame())
129     assign(paste0("arcstr."), gsub("[:space:]", "", out)),
130     arc.strength(res, complete.training_data, criterion =
131     "bic"),
132     envir = parent.frame())
133     p <- p+1
134     setTxtProgressBar(pb, p)
135 }
136
137 #EM
138 i <- 0
139 for (a in score.algorithms) {
140     i <- i + 1
141     start = bn.fit(empty.graph(names(incomplete.training_data)),
142     incomplete.training_data)
143     out <- paste0("EM - ", score.algorithms.names[i])
144     #permutation list
145     permutation.list <- append(permutation.list, out)
146     # structural EM
147     res <-
148     structural.em(incomplete.training_data,
149     maximize = tolower(a),
150     start = start)
151     #assign arc strength and network
152     assign(paste0("fitted."), gsub("[:space:]", "", out)),
153     bn.fit(res, incomplete.training_data),
154     envir = parent.frame())
155     assign(
156     paste0("arcstr."), gsub("[:space:]", "", out)),
157     arc.strength(res, complete.training_data, criterion = "bic"
158     ),
159     envir = parent.frame()
160 )
161     p <- p+1
162     setTxtProgressBar(pb, p)
163 }
164 #prepare the dataframe with the permutation list
165 df <- data.frame(as.data.frame(do.call(rbind, permutation.list)
166 ))
167 colnames(df) <- "Algorithm"
168 close(pb)
169 return(df)
170 }

```

Code 4: Evaluation of our learned models by comparing predictions with test data and exporting the dedicated ROC plots. Due to the runtime, a progress bar is incorporated.

```

1 roc_plots <- function(complete.testing_data) {
2   # progress bar as the function takes some time
3   p <- 0
4   pb <- txtProgressBar(min = 0,
5                         max = ncol(complete.testing_data),
6                         style = 3,
7                         width = 50,
8                         char = "=")
9   for(i in 1:ncol(complete.testing_data)) {
10    #iterate through the columns — here countries
11    AUROC.list <- list()
12    filename <-
13      paste0("Exports/ROC/", colnames(complete.testing_data[i]), ".
14            pdf")
15    cairo_pdf(file = filename)
16    par(pty = "s", family = "Optima")
17    #start plot with style, axis and titles
18    plot(
19      NA,
20      main = colnames(complete.testing_data[i]),
21      ylim = c(0, 1),
22      xlim = c(1, 0),
23      xlab = "Specificity",
24      ylab = "Sensitivity",
25      family = "Optima"
26    )
27    #color palatte and legend
28    mypal <- viridis(length(unlist(permutation.list)))
29    legend(
30      "bottomright",
31      legend = unlist(permutation.list),
32      col = mypal,
33      bty = "n",
34      lty = 1,
35      cex = 0.63
36    )
37    z <- 0
38    #iterate through our models and plot the curve for each
39    for (j in permutation.list) {
40      z <- z + 1
41      out <- paste0(j)
42      fitted <- get(paste0("fitted.", gsub("[:space:]", "", out))
43    )
44    #make prediction with model
45    prediction <-
46      predict(
47        fitted,
48        colnames(complete.testing_data[i]),
49        complete.testing_data,
50        #bayesian prediction with all the available nodes
51        method = "bayes-lw",
52        prob = TRUE
53      )

```

```

52     #assign the probs
53     prediction.attributes <- attributes(prediction)$prob[1, ]
54     #plot the roc curve
55     roc.curve <-
56     roc(
57         response = complete.testing_data[[i]],
58         predictor = prediction.attributes,
59         levels = c(-1, 1),
60         direction = ">"
61     )
62     par(pty = "s", family = "Optima")
63     lines(
64         main = colnames(complete.testing_data[i]),
65         roc.curve,
66         xlim = c(1, 0),
67         ylim = c(0, 1),
68         col = mypal[z],
69         lwd = 0.8
70     )
71     #AUROC value
72     app <- auc(roc.curve)
73     ## save AUROC in list
74     AUROC.list <- append(AUROC.list, app)
75
76 }
77 ## save AUROC in column of country and add new row for each
78     ## model
79     new.data.frame <-
80     data.frame(as.data.frame(do.call(rbind, AUROC.list)))
81     colnames(new.data.frame) <- colnames(complete.testing_data[i])
82     AUROC <-<- cbind(AUROC, new.data.frame)
83
84     ## Finish PDF
85     dev.off()
86     p <- p+1
87     setTxtProgressBar(pb, p)
88 }
89 #close progress bar
90 close(pb)
91 }

```

Code 5: Inductive Causation algorithm based on PEARL and MEEK from Figure 22 – implemented on top of `bnlearn` package with try blocks to avoid acyclicity. In order to get a better insight into the algorithm, we provide exports with information about the applied rule and the associated nodes after each change in the structure (like a *flip-book*). In addition, the final export provides the ratio of oriented arcs in percent, as this is one of our main interests.

```

1 inductive_causation <- function(my_bn) {
2   #Start with skeleton of BN — our PDAG
3   bn <- skeleton(bn.net(my_bn))
4   #underlying network of the Bayesian network for the unshielded
      colliders
5   my_bn <- bn.net(my_bn)
6   #local variables to indicate whether something changed and thus a
      new plot needs to be generated
7   r <- ""
8   co <- FALSE
9   # for nodes n in PDAG
10  for (n in nodes(bn)) {
11    # if neighbours(n) > 1
12    if (length(nbr(bn, n)) > 1) {
13      #list of neighbors
14      x <- nbr(bn, n)
15      #permutation of all neighbors of length 2
16      perm.list <- permutations(length(x), 2, x, repeats = FALSE)
17      #iterate through all permutations a, b
18      for (z in 1:(length(perm.list) / 2)) {
19        a <- perm.list[z, 1]
20        b <- perm.list[z, 2]
21        # if no edge/arc between a and b && only edge between n-a
          and n-b
22        # the approach doesnt seem intuitive, but there is no
          faster way to check for arcs than dropping or adding
          those in a copy of the network and then compare them
23        updated <- drop.edge(bn, a, b, )
24        updated <- drop.arc(updated, a, b)
25        updated <- set.edge(updated, a, n)
26        updated <- set.edge(updated, b, n)
27        res <- paste0(all.equal(updated, bn))
28        check <- "TRUE"
29        # and only if old bn has these orientations
30        updated_bn <- set.arc(my_bn, a, n, check.cycles = FALSE)
31        updated_bn <-
32          set.arc(updated_bn, b, n, check.cycles = FALSE)
33        res_bn <- paste0(all.equal(updated_bn, my_bn))
34        if ((check == res) & (check == res_bn)) {
35          # if no edge/arc between a and b && only edge between n-a
            and n-b
36          res1 <-
37            try(set.arc(bn, a, n, check.cycles = TRUE), silent =
              TRUE)
38          res2 <-
39            try(set.arc(bn, b, n, check.cycles = TRUE), silent =
              TRUE)
40          #try blocks to make sure our PDAG remains acyclical
41          if (!(class(res1) == "try-error") &
42              !(class(res2) == "try-error")) {

```

```

43     bn <- set.arc(bn, a, n)
44     bn <- set.arc(bn, b, n)
45     # collider has been added and thus we can plot the
      change
46     co <- TRUE
47   }
48 }
49 if (co == TRUE) {
50   # Only for visualisations
51   out <- paste0("Colliders - ", n, " - ", a, " - ", b)
52   filename <-
53     paste0("Exports/Networks/Causal/Iteration/", out, ".pdf
      ")
54   bn_plot(bn, out, filename, "")
55   # so we can set the variable "co" to FALSE again
56   co <- FALSE
57 }
58 }
59 }
60
61 }
62
63 new_arcs <- TRUE
64 i <- 0
65 # while loop: only if no more arcs can be oriented, the loop
      terminates
66 while (new_arcs == TRUE) {
67   i <- i + 1
68   new_arcs <- FALSE
69   for (n in nodes(bn)) {
70     # if length(nbr(n))>1:
71     if (length(nbr(bn, n)) > 1) {
72       x <- nbr(bn, n)
73       perm.list <- permutations(length(x), 2, x, repeats = FALSE)
74       #for all permutations (a, b) of nbr(n) with l=2
75       for (z in 1:(length(perm.list) / 2)) {
76         a <- perm.list[z, 1]
77         b <- perm.list[z, 2]
78         updated <- drop.edge(bn, a, b)
79         updated <- drop.arc(updated, a, b)
80         # if no edge/arc between a,b then
81         res <- paste0(all.equal(updated, bn))
82         check <- "TRUE"
83         # the approach doesnt seem intuitive, but there is no
            faster way to check for arcs than dropping or adding
            those in a copy of the network and then compare them
84         if ((check == res)) {
85           updated <- drop.edge(bn, n, b)
86           # if not (b->n or n->b), so if edge not arc
87           res <- paste0(all.equal(updated, bn))
88           if (!(check == res)) {
89             updated <- set.arc(bn, a, n, check.cycles = FALSE)
90             # if a->n R1
91             res <- paste0(all.equal(updated, bn))
92             if ((check == res)) {
93               # try n -> b and new_arcs = TRUE || R1
94               res <-

```

```

95         try(set.arc(bn, n, b, check.cycles = TRUE),
96             silent = TRUE)
97         #try blocks to make sure our PDAG remains acyclical
98         if (!(class(res) == "try-error")) {
99             bn <- set.arc(bn, n, b)
100             new_arcs <- TRUE
101             #set dedicated rule for the plots
102             r <- " - R1"
103         }
104     }
105 }
106 # to make sure only one change per iteration happens for
107 # better visualisations
108 if (r == "") {
109     updated <- drop.edge(bn, n, b)
110     # if not (b->n or n->b), so if edge not arc
111     res <- paste0(all.equal(updated, bn))
112     if (!(check == res)) {
113         # if b in descendants(bn, "n") R2
114         des <- paste0(is.element(b, descendants(bn, n)))
115         if ((check == des)) {
116             # try n -> b and new_arcs = TRUE || R2
117             res <-
118                 try(set.arc(bn, n, b, check.cycles = TRUE),
119                     silent = TRUE)
120             #try blocks to make sure our PDAG remains acyclical
121             if (!(class(res) == "try-error")) {
122                 bn <- set.arc(bn, n, b)
123                 new_arcs <- TRUE
124                 #set dedicated rule for the plots
125                 r <- " - R2"
126             }
127         }
128     }
129 }
130 #rules 3 and 4 only for 3 nodes or more
131 if (length(nbr(bn, n)) > 2) {
132     x <- nbr(bn, n)
133     perm.list <- permutations(length(x), 3, x, repeats = FALSE)
134     #for all permutations (a, b) of nbr(n) with l=2
135     for (z in 1:(length(perm.list) / 3)) {
136         a <- perm.list[z, 1]
137         b <- perm.list[z, 2]
138         c <- perm.list[z, 3]
139         # if a -> b and c -> b and n - b for r3
140         # if a -> b and b -> c and n - c for r4
141         # check for common rules
142         updated1 <- drop.edge(bn, n, a)
143         updated2 <- drop.edge(bn, n, b)
144         updated3 <- drop.edge(bn, n, c)
145         updated <- set.arc(bn, a, b, check.cycles = FALSE)
146         res1 <- paste0(all.equal(updated1, bn))
147         res2 <- paste0(all.equal(updated2, bn))
148         res3 <- paste0(all.equal(updated3, bn))

```

```

149     res <- paste0(all.equal(updated, bn))
150     check <- "TRUE"
151     if (((check == res1) |
152         (check == res2) |
153         (check == res3))) & (check == res)) {
154         # if c->b then n->b
155         updated <- set.arc(bn, c, b, check.cycles = FALSE)
156         res <- paste0(all.equal(updated, bn))
157         if ((check == res)) {
158             res <- try(set.arc(bn, n, b, check.cycles = TRUE),
159                 silent = TRUE)
159             #try blocks to make sure our PDAG remains acyclical
160             if (!(class(res) == "try-error")) {
161                 bn <- set.arc(bn, n, b)
162                 new_arcs <- TRUE
163                 #set dedicated rule for the plots
164                 r <- " - R3"
165             }
166         }
167         #if b->c then n->c
168         updated <- set.arc(bn, b, c, check.cycles = FALSE)
169         res <- paste0(all.equal(updated, bn))
170         if ((check == res)) {
171             res <- try(set.arc(bn, n, c, check.cycles = TRUE),
172                 silent = TRUE)
172             #try blocks to make sure our PDAG remains acyclical
173             if (!(class(res) == "try-error")) {
174                 bn <- set.arc(bn, n, c)
175                 new_arcs <- TRUE
176                 #set dedicated rule for the plots
177                 r <- " - R4"
178             }
179         }
180     }
181 }
182 }
183 if (!(r == "")) {
184     # Only for visualisations while iterating
185     out <- paste0("Orientation - ", n, r, " - ", i)
186     filename <-
187     paste0("Exports/Networks/Causal/Iteration/", out, ".pdf")
188     bn_plot(bn, out, filename, "")
189     #set dedicated rule back to empty
190     r <- ""
191 }
192 }
193
194
195 }
196
197 return(bn)
198 }
199
200 #Final export of plots with indicator of how many arcs can be
    oriented in the subtitle
201 causal_plots <- function(AUROC, n) {
202     x <- 0

```



```

203 for (j in AUROC[1:n, 1]) {
204   x <- x + 1
205   out <- paste0(j)
206   fitted <-
207     get(paste0("fitted.", gsub("[:space:]", "", out)))
208   bn <- inductive_causation(fitted)
209   undirected <- as.numeric(length((undirected.arcs(bn))) / 4)
210   directed <- as.numeric(length((directed.arcs(bn))) / 2)
211   total <- undirected + directed
212   result <- round(((directed / total) * 100), digits = 0)
213   # directed of total edges are oriented (result %)
214   out <- paste0("Causal Network from ", out)
215   filename <- paste0("Exports/Networks/Causal/Causal", x, ".pdf")
216   submain <-
217     paste0(directed, " of ", total, " edges are oriented (",
218           result, "%)")
219   bn_plot(bn, out, filename, submain)
220 }

```

Code 6: Tests for the unshielded colliders and the orientation rules of the Inductive Causation algorithm in Code 5

```

1 causal_test <- function(AUROC, i, n) {
2   x <- 0
3   for (j in AUROC[i:n, 1]) {
4     x <- x + 1
5     out <- paste0(j)
6     fitted <-
7       get(paste0("fitted.", gsub("[:space:]", "", out)))
8     bn <- bn.net(fitted)
9     #compare the unshielded colliders of the Bayesian network with
10      the resulting causal plots
11     expression <-
12       unshielded.colliders(bn, arcs = FALSE, debug = FALSE)
13     return(expression)
14   }
15 }
16 # orientation rules are the same as the inductive causation
17   algorithm without the colliders
18 orientation_rules <- function(my_bn) {
19   bn <- my_bn
20   r <- ""
21   new_arcs <- TRUE
22   i <- 0
23   # while loop: only if no more arcs can be oriented, the loop
24     terminates
25   while (new_arcs == TRUE) {
26     i <- i + 1
27     new_arcs <- FALSE
28     for (n in nodes(bn)) {
29       # if length(nbr(n))>1:

```

```

28   if (length(nbr(bn, n)) > 1) {
29     x <- nbr(bn, n)
30     perm.list <- permutations(length(x), 2, x, repeats = FALSE)
31     #for all permutations (a, b) of nbr(n) with l=2
32     for (z in 1:(length(perm.list) / 2)) {
33       a <- perm.list[z, 1]
34       b <- perm.list[z, 2]
35       updated <- drop.edge(bn, a, b)
36       updated <- drop.arc(updated, a, b)
37       # if no edge/arc between a,b then
38       res <- paste0(all.equal(updated, bn))
39       check <- "TRUE"
40       # the approach doesnt seem intuitive, but there is no
         faster way to check for arcs than dropping or adding
         those in a copy of the network and then compare them
41       if ((check == res)) {
42         updated <- drop.edge(bn, n, b)
43         # if not (b->n or n->b), so if edge not arc
44         res <- paste0(all.equal(updated, bn))
45         if (!(check == res)) {
46           updated <- set.arc(bn, a, n, check.cycles = FALSE)
47           # if a->n R1
48           res <- paste0(all.equal(updated, bn))
49           if ((check == res)) {
50             # try n -> b and new_arcs = TRUE || R1
51             res <-
52               try(set.arc(bn, n, b, check.cycles = TRUE),
                 silent = TRUE)
53             #try blocks to make sure our PDAG remains acyclical
54             if (!(class(res) == "try-error")) {
55               bn <- set.arc(bn, n, b)
56               new_arcs <- TRUE
57               #set dedicated rule for the plots
58               r <- " - R1"
59             }
60           }
61         }
62       }
63       # to make sure only one change per iteration happens for
         better visualisations
64       if (r == ""){
65         updated <- drop.edge(bn, n, b)
66         # if not (b->n or n->b), so if edge not arc
67         res <- paste0(all.equal(updated, bn))
68         if (!(check == res)) {
69           # if b in descendants(bn, "n") R2
70           des <- paste0(is.element(b, descendants(bn, n)))
71           if ((check == des)) {
72             # try n -> b and new_arcs = TRUE || R2
73             res <-
74               try(set.arc(bn, n, b, check.cycles = TRUE),
                 silent = TRUE)
75             #try blocks to make sure our PDAG remains acyclical
76             if (!(class(res) == "try-error")) {
77               bn <- set.arc(bn, n, b)
78               new_arcs <- TRUE
79               #set dedicated rule for the plots

```

```

80         r <- " - R2"
81     }
82 }
83 }
84 }
85 }
86 }
87 #rules 3 and 4 only for 3 nodes or more
88 if (length(nbr(bn, n)) > 2) {
89     x <- nbr(bn, n)
90     perm.list <- permutations(length(x), 3, x, repeats = FALSE)
91     #for all permutations (a, b) of nbr(n) with l=2
92     for (z in 1:(length(perm.list) / 3)) {
93         a <- perm.list[z, 1]
94         b <- perm.list[z, 2]
95         c <- perm.list[z, 3]
96         # if a -> b and c -> b and n - b for r3
97         # if a -> b and b -> c and n - c for r4
98         # check for common rules
99         updated1 <- drop.edge(bn, n, a)
100        updated2 <- drop.edge(bn, n, b)
101        updated3 <- drop.edge(bn, n, c)
102        updated <- set.arc(bn, a, b, check.cycles = FALSE)
103        res1 <- paste0(all.equal(updated1, bn))
104        res2 <- paste0(all.equal(updated2, bn))
105        res3 <- paste0(all.equal(updated3, bn))
106        res <- paste0(all.equal(updated, bn))
107        check <- "TRUE"
108        if (((check == res1) |
109             (check == res2) |
110             (check == res3))) & (check == res)) {
111            # if c->b then n->b
112            updated <- set.arc(bn, c, b, check.cycles = FALSE)
113            res <- paste0(all.equal(updated, bn))
114            if ((check == res)) {
115                res <- try(set.arc(bn, n, b, check.cycles = TRUE),
116                           silent = TRUE)
117                #try blocks to make sure our PDAG remains acyclical
118                if (!(class(res) == "try-error")) {
119                    bn <- set.arc(bn, n, b)
120                    new_arcs <- TRUE
121                    #set dedicated rule for the plots
122                    r <- " - R3"
123                }
124            }
125            #if b->c then n->c
126            updated <- set.arc(bn, b, c, check.cycles = FALSE)
127            res <- paste0(all.equal(updated, bn))
128            if ((check == res)) {
129                res <- try(set.arc(bn, n, c, check.cycles = TRUE),
130                           silent = TRUE)
131                #try blocks to make sure our PDAG remains acyclical
132                if (!(class(res) == "try-error")) {
133                    bn <- set.arc(bn, n, c)
134                    new_arcs <- TRUE
135                    #set dedicated rule for the plots
136                    r <- " - R4"

```

```

135     }
136   }
137 }
138 }
139 }
140 if (!(r == "")) {
141   # Only for visualisations while iterating
142   out <- paste0("Orientation - ", n, r, " - ", i)
143   filename <-
144     paste0("Exports/Networks/Causal/Iteration/", out, ".pdf")
145   bn_plot(bn, out, filename, "")
146   #set dedicated rule back to empty
147   r <- ""
148 }
149 }
150
151
152 }
153
154 return(bn)
155 }
156
157 ## Test cases to be applied in main class ##
158
159 # r1 = model2network("[a][b][n|a:b]")
160 # r1 <- set.edge(r1, "n", "b")
161 #
162 # r2 = model2network("[a|n][b|a:n][n]")
163 # r2 <- set.edge(r2, "n", "b")
164 #
165 # r3 = model2network("[a|n][b|a:c:n][c|n][n]")
166 # r3 <- set.edge(r3, "n", "a")
167 # r3 <- set.edge(r3, "n", "b")
168 # r3 <- set.edge(r3, "n", "c")
169 #
170 # r4 = model2network("[a|n][b|a:n][c|b:n][n]")
171 # r4 <- set.edge(r4, "n", "a")
172 # r4 <- set.edge(r4, "n", "b")
173 # r4 <- set.edge(r4, "n", "c")
174 #
175 #
176 # #repeat for r1 to r4
177 # graphviz.plot(r1, layout = "dot")
178 # test <- orientation_rules(r1)
179 # graphviz.plot(test, layout = "dot")

```

References

- [1] ALEXANDER, C., KAPRAUN, J., AND KOROVILAS, D. Trading and investing in volatility products. *Financial Markets, Institutions & Instruments* 24 (11 2015).
- [2] ASX GROUP. Asx – cash market trading hours. <https://www2.asx.com.au/markets/market-resources/trading-hours-calendar/cash-market-trading-hours>, 2022. [Online; accessed 01-July-2022].
- [3] BAREINBOIM, E., BRITO, C., AND PEARL, J. Local characterizations of causal bayesian networks. In *Graph Structures for Knowledge Representation and Reasoning* (Berlin, Heidelberg, 2012), M. Croitoru, S. Rudolph, N. Wilson, J. Howse, and O. Corby, Eds., Springer Berlin Heidelberg, pp. 1 – 17.
- [4] BATES, B. C., DOWDY, A. J., AND MCCAW, L. A bayesian approach to exploring the influence of climate variability modes on fire weather conditions and lightning-ignited wildfires. *Climate Dynamics* 57, 3 (2021), 1207 – 1225.
- [5] BERNARDO, J. M., AND SMITH, A. F. M. *Bayesian Theory*. John Wiley & Sons, LTD, 2000.
- [6] BLACKROCK. Fact sheet as of 03/31/2022 – iShares China Large-Cap ETF. <https://www.ishares.com/us/literature/fact-sheet/fxi-ishares-china-large-cap-etf-fund-fact-sheet-en-us.pdf>, March 2022. [Online; accessed 11-June-2022].
- [7] BORGELT, C., AND KRUSE, R. A critique of inductive causation. In *Symbolic and Quantitative Approaches to Reasoning and Uncertainty* (Berlin, Heidelberg, 1999), A. Hunter and S. Parsons, Eds., Springer Berlin Heidelberg, pp. 68–79.
- [8] BUNTINE, W. A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering* 8 (1996), 195–210.
- [9] CARVALHO, A. M. Scoring functions for learning bayesian networks. Tech. rep., Instituto Superior Técnico, Technical University of Lisbon, 2009.
- [10] CASTELLETTI, F., CONSONNI, G., VEDOVA, M. L. D., AND PELUSO, S. Learning Markov Equivalence Classes of Directed Acyclic Graphs: An Objective Bayes Approach. *Bayesian Analysis* 13, 4 (2018), 1235 – 1260.
- [11] CBOE EXCHANGE. Cboe – hours & holidays. <https://www.cboe.com/about/hours>, 2022. [Online; accessed 01-July-2022].
- [12] CHICHARRO, D., AND PANZERI, S. Algorithms of causal inference for the analysis of effective connectivity among brain regions. *Frontiers in Neuroinformatics* 8 (2014).

- [13] DEMPSTER, A. P., LAIRD, N. M., AND RUBIN, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)* 39, 1 (1977), 1–38.
- [14] DEUTSCHE BÖRSE AG. Xetra – trading calendar and trading hours. <https://www.xetra.com/xetra-en/trading/trading-calendar-and-trading-hours>, 2022. [Online; accessed 01-July-2022].
- [15] DO, C. B., AND BATZOGLOU, S. What is the expectation maximization algorithm? *Nature Biotechnology* 26 (2008), 897 – 899.
- [16] DRUZDZEL, M. J. The role of assumptions in causal discovery. In *8th Workshop on Uncertainty Processing (WUPES-09)* (September 2009), pp. 57 – 68.
- [17] ERTEL, W. *Grundkurs Künstliche Intelligenz - Eine praxisorientierte Einführung*, 5th ed. Springer-Verlag, 2020.
- [18] EURONEXT GROUP. Euronext – trading hours (derivatives markets). <https://live.euronext.com/en/media/295>, 2022. [Online; accessed 01-July-2022].
- [19] FASSAS, A., AND SIRIOPOULOS, C. Implied volatility indices – a review. *The Quarterly Review of Economics and Finance* 79, C (2021), 303–329.
- [20] FRIEDMAN, N. The bayesian structural EM algorithm. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence* (San Francisco, CA, 1998), Morgan Kaufmann Publishers Inc., pp. 129–138.
- [21] FRIEDMAN, N., GOLDSZMIDT, M., AND WYNER, A. Data analysis with bayesian networks: A bootstrap approach. In *UAI '99: Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence* (1999), Morgan Kaufmann, pp. 196 – 205.
- [22] FUSION MEDIA. AXVI – historical rates. <https://www.investing.com/indices/s-p-asx-200-vix-historical-data>, 2022. [Online; accessed 27-April-2022].
- [23] FUSION MEDIA. JNIV – historical rates. <https://www.investing.com/indices/nikkei-volatility-historical-data>, 2022. [Online; accessed 27-April-2022].
- [24] FUSION MEDIA. NIFVIX – historical rates. <https://www.investing.com/indices/india-vix-historical-data>, 2022. [Online; accessed 27-April-2022].
- [25] FUSION MEDIA. RVI – historical rates. <https://www.investing.com/indices/russian-vix-historical-data>, 2022. [Online; accessed 27-April-2022].

- [26] FUSION MEDIA. VCAC – historical rates. <https://www.investing.com/indices/cac-40-vix-historical-data>, 2022. [Online; accessed 27-April-2022].
- [27] FUSION MEDIA. VDAX – historical rates. <https://www.investing.com/indices/vdax-historical-data>, 2022. [Online; accessed 27-April-2022].
- [28] FUSION MEDIA. VFTSE – historical rates. <https://www.investing.com/indices/ftse-100-vix-historical-data>, 2022. [Online; accessed 27-April-2022].
- [29] FUSION MEDIA. VHSI – historical rates. <https://www.investing.com/indices/hsi-volatility-historical-data>, 2022. [Online; accessed 27-April-2022].
- [30] FUSION MEDIA. VIX – historical rates. <https://www.investing.com/indices/volatility-s-p-500-historical-data>, 2022. [Online; accessed 27-April-2022].
- [31] FUSION MEDIA. VIXI – historical rates. <https://www.investing.com/indices/s-p-tsx-60-vix-historical-data>, 2022. [Online; accessed 27-April-2022].
- [32] FUSION MEDIA. VXFXI – historical rates. <https://www.investing.com/indices/cboe-china-etf-volatility-historical-data>, 2022. [Online; accessed 27-April-2022].
- [33] GLYMOUR, C., ZHANG, K., AND SPIRITES, P. Review of causal discovery methods based on graphical models. *Frontiers in Genetics* 10 (2019).
- [34] GRACZYK, M. B., AND DUARTE QUEIRÓS, S. M. Intraday seasonalities and nonstationarity of trading volume in financial markets: Individual and cross-sectional features. *PLOS ONE* 11, 11 (11 2016), 1 – 25.
- [35] HASTIE, T., TIBSHIRANI, R., AND FRIEDMAN, J. *The Elements of Statistical Learning*, 2nd ed. Springer New York, 2009.
- [36] HECKERMAN, D. A tutorial on learning with bayesian networks. In *Innovations in Bayesian Networks: Theory and Applications*, D. E. Holmes and L. C. Jain, Eds. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 33–82.
- [37] HECKERMAN, D., GEIGER, D., AND CHICKERING, D. M. Learning bayesian networks: The combination of knowledge and statistical data. *Machine Learning* 20, 3 (1995), 197–243.
- [38] HIGHAM, N. J. *Accuracy and Stability of Numerical Algorithms*, second ed. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002.

- [39] HOLMES, D. E., AND JAIN, L. C. Introduction to bayesian networks. In *Innovations in Bayesian Networks: Theory and Applications*, D. E. Holmes and L. C. Jain, Eds. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 1–5.
- [40] HONG KONG EXCHANGES AND CLEARING. Hkex – securities market trading hours. https://www.hkex.com.hk/Services/Trading-hours-and-Severe-Weather-Arrangements/Trading-Hours/Securities-Market?sc_lang=en, 2022. [Online; accessed 01-July-2022].
- [41] HOSMER JR., D. W., LEMESHOW, S., AND STURDIVANT, R. X. *Applied Logistic Regression*, 3rd ed. John Wiley & Sons, 2013.
- [42] HULL, J. C. *Options, Futures, and other Derivatives*, 11th ed. Pearson, Boston, 2021.
- [43] IMOTO, S., KIM, S., SHIMODAIRA, H., ABURATANI, S., TASHIRO, K., KUHARA, S., AND MIYANO, S. Bootstrap analysis of gene networks based on bayesian networks and nonparametric regression. *Genome Informatics* 13 (2002), 369 – 370.
- [44] JAPAN EXCHANGE GROUP. Jpx – trading hours. <https://www.jpx.co.jp/english/derivatives/rules/trading-hours/>, 2022. [Online; accessed 01-July-2022].
- [45] JOSHI, M. S. *The Concepts and Practice of Mathematical Finance*, 2nd ed. Mathematics, Finance and Risk. Cambridge University Press, 2008.
- [46] KAUERMANN, G., KÜCHENHOFF, H., AND HEUMANN, C. *Statistical Foundations, Reasoning and Inference*. Springer International Publishing, 2021.
- [47] LAURITZEN, S. L., AND SPIEGELHALTER, D. J. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)* 50, 2 (1988), 157–224.
- [48] MALZ, A. Crises and volatility. *Risk* 14 (November 2001), 105 – 108.
- [49] MEEK, C. Causal inference and causal explanation with background knowledge. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence* (San Francisco, CA, USA, 1995), UAI’95, Morgan Kaufmann Publishers Inc., pp. 403–410.
- [50] MITCHELL, T. M. *Machine Learning*. McGraw-Hill, New York, 1997.
- [51] MOSCOW EXCHANGE. Moex – trading schedule. <https://www.moex.com/n45650/?nt=201>, 2022. [Online; accessed 01-July-2022].
- [52] NAGARAJAN, R., SCUTARI, M., AND LÈBRE, S. *Bayesian Networks in R*. Springer New York, 2014.

- [53] NATIONAL STOCK EXCHANGE OF INDIA. Nse – market timings & holidays. <https://www.nseindia.com/resources/exchange-communication-holidays>, 2022. [Online; accessed 01-July-2022].
- [54] NEATH, A. A., AND CAVANAUGH, J. E. The bayesian information criterion: Background, derivation, and applications. *WIREs Computational Statistics* 4, 2 (2012), 199–203.
- [55] ONVISTA MEDIA. VSMI – historical rates. <https://www.onvista.de/index/VSMI-VOLATILITAETS-Index-7911812>, 2022. [Online; accessed 27-April-2022].
- [56] PEARL, J. *Causality*, 2nd ed. Cambridge University Press, Cambridge, UK, 2009.
- [57] PEARL, J., AND VERMA, T. A theory of inferred causation. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning* (San Francisco, CA, USA, 1991), KR’91, Morgan Kaufmann Publishers Inc., pp. 441–452.
- [58] RUSSELL, S. J., AND NORVIG, P. *Künstliche Intelligenz – ein moderner Ansatz*, 3rd ed. Pearson, Higher Education, 2012.
- [59] RUSSELL, S. J., AND NORVIG, P. *Artificial Intelligence – A Modern Approach*, 4th ed. Pearson, 2021.
- [60] SCHWARZ, G. Estimating the dimension of a model. *The Annals of Statistics* 6, 2 (1978), 461–464.
- [61] SCUTARI, M. Package ‘bnlearn’. Tech. rep., CRAN, 2022.
- [62] SCUTARI, M., GRAAFLAND, C. E., AND GUTIÉRREZ, J. M. Who learns better bayesian network structures: Accuracy and speed of structure learning algorithms. *International Journal of Approximate Reasoning* 115 (2019), 235–253.
- [63] SCUTARI, M., VITOLO, C., AND TUCKER, A. Learning bayesian networks from big data with greedy search: computational complexity and efficient implementation. *Statistics and Computing* 29, 5 (2019), 1095 – 1108.
- [64] SHUMWAY, R. H., AND STOFFER, D. S. *Time Series Analysis and Its Applications*, 4th ed. Springer International Publishing, 2017.
- [65] SIX GROUP. Six – handelszeiten. <https://www.six-group.com/de/products-services/the-swiss-stock-exchange/trading/trading-provisions/trading-hours.html#scrollTo=trading-hours-overview>, 2022. [Online; accessed 01-July-2022].
- [66] SPIRTEs, P., AND GLYMOUR, C. An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review* 9, 1 (1991), 62 – 72.

- [67] STIGLER, S. M. The true title of bayes’s essay. *Statistical Science* 28, 3 (2013).
- [68] TMX GROUP. Tsx – trading hours. <https://www.tsx.com/trading/calendars-and-trading-hours/trading-hours>, 2022. [Online; accessed 01-July-2022].
- [69] VERMA, T., AND PEARL, J. Causal networks: Semantics and expressiveness. In *Proceedings of the Fourth Annual Conference on Uncertainty in Artificial Intelligence* (NLD, 1990), UAI ’88, North-Holland Publishing Co., pp. 69 – 78.
- [70] VERMA, T., AND PEARL, J. Equivalence and synthesis of causal models. In *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence* (USA, 1990), UAI ’90, Elsevier Science Inc., pp. 255 – 270.
- [71] WIKIPEDIA CONTRIBUTORS. Duck test – Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Duck_test&oldid=1069942509, 2022. [Online; accessed 13-February-2022].
- [72] WORLD FEDERATION OF EXCHANGES. WFE – market statistics. <https://focus.world-exchanges.org/issue/september-2022/market-statistics>, 2022. [Online; accessed 11-September-2022].