University of Genoa
Department of Mathematics
PhD in Mathematics and Applications
Curriculum Mathematical Methods for Data Analysis

Stratified Staged Trees: Modelling, Software and Applications

Supervisor: Professor Eva Riccomagno
Department of Mathematics

PhD Student: Federico Carli
Freshman Number 3657085

Academic Year 2020/2021

Contents

List of Figures

List of Tables

List of Algorithms

Declarations

The author declares that, within all the projects in which he collaborated during the PhD, he mainly carried out in first person what follows:

- a state of the art review of Probabilistic Graphical Models, with particular attention to Bayesian Networks. An in-depth detailed estimation method for PGMs is that based on neighborhood regression. An analysis on a pediatric dentistry framework was personally conducted, then performing a reading of the results together with Doctor Alessandro Ugolini, which is the expert of the reference sector. An Undirected Graph is used for this data analysis;

- the neighborhood regression estimation criterion is adopted to perform a variable selection in an insurance application. In particular, this approach is the core of the implemented method *mixed Vector Autoregressive Graphical Models*. In this joint analysis carried out with Elena Pesce, the author contributed most to the application of this method and to the prediction of the response variables through regressive models. Furthermore, R Shiny code was personally developed in order to provide an interactive app usable also from the business experts;

- a review of the theory about Staged Trees and Chain Event Graphs. The author also proposed new estimation methods for this class of models, based on the penalized likelihood and on the concept of distance or divergence existing between pairs of discrete probability distributions;

- the importance of the ordering with which the variables are placed in the strata of the Staged Tree and the need to know how to deal with structural and/or observed zeros have been addressed throughout the PhD. This has led to personally provide proposals with which to manage both situations;

- for the study carried out on Chapter 4 which highlights the relation existing between Staged Trees and Bayesian Networks, the author worked mostly on the formulation of the Staged Tree corresponding to a BN, while his colleagues

Gherardo Varando and Manuele Leonelli contributed mainly to develop the correspondence between the Asymmetry-labeled DAG and the Staged Tree;

- regarding the *R* package **stagedtrees** implemented with Manuele Leonelli, Eva Riccomagno and Gherardo Varando, the author worked extensively to the code for the structure learning algorithms and functions for providing inference information according to estimated Staged Trees. Instead, the other authors have worked more on the outputs, such as prints and plots, and the correct assembly of the functions to create a valid and functioning package. The author carried out also the simulation study in Section 5.5 to show the usage of the package on a set of datasets;

- the author has since the beginning of the PhD found a strong similarity between Staged Trees and Classification Trees. For this reason, thanks to the help of Manuele Leonelli and Gherardo Varando, he proposed the use of Staged Trees as a classification tool. Personally, the author conducted a study on various datasets, showing results comparable with those of other well-known classifiers;

- the author found a relationship existing between Relational Models and Staged Trees. In particular, Stratified Staged Trees are relational models with an overall effect, that means they are members of regular exponential family. This implies that the theory about maximum likelihood estimates of Staged Trees parameters are analogous to those computed under traditional log-linear models.

Introduction

This thesis is focused on Probabilistic Graphical Models (PGMs), which are a rich framework for encoding probability distributions over complex domains. In particular, joint multivariate distributions over large numbers of random variables that interact with each other can be investigated through PGMs and conditional independence statements can be succinctly represented with graphical representations. These representations sit at the intersection of statistics and computer science, relying on concepts mainly from probability theory, graph algorithms and machine learning. They are applied in a wide variety of fields, such as medical diagnosis, image understanding, speech recognition, natural language processing, and many more.

The fundamental and universal applicability of PGMs is due to a number of factors. Firstly, the graphs can visually represent the scientific content of a given model and facilitate communication between researcher and statistician. Secondly, statistical models supported on PGMs are naturally modular so that complex problems can be described and handled by careful combination of simple elements. Thirdly, graphs are natural data structures for modern digital computers. Thus, models can be efficiently communicated to these and the road is paved for exploiting their computational power [77].

Over the years theory and methodology have developed and been extended in a multitude of directions. In particular, in this thesis different aspects of new classes of PGMs called Staged Trees and Chain Event Graphs (CEGs) are studied. In some sense, Staged Trees are a generalization of Bayesian Networks (BNs). Indeed, BNs provide a transparent graphical tool to define a complex process in terms of conditional independent structures. This facilitates the identification of relevant structural components of the process, allows the factorization of the joint probability distribution and optimises the computational costs and time for inferences. Despite this and their obvious strengths in allowing for the reduction in the dimensionality of joint probability distributions of the statistical model and in providing a transparent framework for causal inference, BNs are not optimal GMs in all situ-

ations. The biggest problems with their usage mainly occur when the event space is not a simple product of the sample spaces of the random variables of interest, and when conditional independence statements are true only under certain values of variables. This happens when there are context-specific conditional independence structures [13, 109].

Some extensions to the BN framework have been proposed to handle these issues. For instance, a context-specific BN [13] that uses supplementary trees to represent the conditional probability tables that show context-specific information. Alternatively, the standard BN can be reorganized in order to depict context-specific independences using multiple vertices associated with a single variable. Another proposal is to use Bayesian Multinets or Similarity Networks [49]. These adopt a hypothesis variable to encode the context-specific statements over a particular set of random variables. For each value taken by the hypothesis variable the graphical modeller has to construct a particular BN model called local network. The collection of these local networks constitute a Bayesian Multinet or a Similarity Network. However, in both these approaches, a process is described by a set of networks instead of a single graph. The natural consequence is that the modelling procedure becomes more complicated and the computational complexity to encode these models increases substantially compared to a standard BN. These problems only get worse when the hypothesis variable has to represent context-specific hypotheses associated with different states of the process. The corresponding drawbacks become more pronounced and the computational complexities increase dramatically.

Probabilistic Decision Graph [64] were originally proposed for automated check of probabilistic expert systems under context-specific information and are based on ordered binary decision diagrams [18]. This enables the user to perform efficient probabilistic inference especially in models with context-specific structures. Since a Probabilistic Decision Graph has an underlying tree graph it comes close to Staged Trees and CEGs models, which are the main focus of this thesis. Smith and Anderson [107] showed that CEG models encompass all discrete BN models and its discrete variants described above as a special subclass and they are also richer than Probabilistic Decision Graphs whose semantics is actually somewhat distinct [112].

Unlike most of its competitors, Chain Event Graphs can capture all (also context-specific) conditional independences in a unique graph, obtained by a coalescence over the vertices of an appropriately constructed probability tree, called Staged Tree [27, 99, 107]. CEGs have been developed for categorical variables and have been used for cohort studies [4], causal analysis [111, 115] and case-control studies [68, 69]. Structure learning algorithms have been defined in the literature [5, 26, 28, 105].

The user's toolbox to efficiently and effectively perform uncertainty reasoning with CEGs further includes methods for inference and probability propagation [53, 116], the exploration of equivalence classes [55] and robustness studies [80, 126]. The model class of CEGs and Staged Trees have been further extended to model dynamic problems with recursively updated probabilities [6, 45], decision problems under the framework expected utility maximization [113] and Bayesian games [114].

The main contributions of this thesis to the literature on Staged Trees are related to Stratified Staged Trees with a keen eye of application. Few observations are made on non-Stratified Staged Trees in the last part of the thesis. A core output of the thesis is an *R* software package which efficiently implements a host of functions for learning and estimating Staged Trees from data, relying on likelihood principles. Also structural learning algorithms based on distance or divergence between pair of categorical probability distributions and based on the clusterization of probability distributions in a fixed number of stages for each stratum of the tree are developed. This is in contrast with the only other method for learning Staged Trees available in the literature which follows a Bayesian approach.

The thesis consists of seven chapters. Chapter 1 reviews the essential theory on Probabilistic Graphical Models and presents in detail an estimation criterion based on Neighborhood Regression for time-varying DAGs. It is instrumental to Chapter 2 which deals with a case study provided by Swiss Re Group, the sponsor of this thesis. Chapter 1 includes also the analysis of a dataset on pediatric tooth cavities using Undirected Graphical Models, which will be refined in Chapter 5 using Staged Trees. In Chapter 2 a pilot study in an insurance context is presented. PGMs are used to develop a variables selection based on estimated significant relations between response variable and covariates. PGMs for time series are introduced and a bootstrap approach is adopted in order to produce robust estimates and be careful with data sensitivity, crucial topic in an insurance framework. Chapter 3 presents Staged Trees and Chain Event Graphs, giving details on their theoretical formulations and estimation criteria based on Bayesian or Frequentist approaches or based on distance/divergence between pair of categorical probability distributions. It also gives a review of the Dynamic Programming algorithm, which guarantees a global optimum of one of the estimation criteria above over the set of all Stratified Staged Trees; this assumes an ordering of the components of a random vector $\boldsymbol{X}$. Next, the importance of the order of variables used to build the Staged Tree is highlighted, providing also three different algorithms to infer such variables ordering.

In Chapter 4 the relation existing between Bayesian Networks and Staged Trees is studied in details. It is proposed a conversion algorithm to obtain the Strati-

fied Staged Tree structure corresponding to a given BN. This is presented as an asymmetry-labeled DAGs (ALDAGs), which is a minimal DAG such that the statistical model embedded in the given Staged Tree is contained in the one associated to that DAG. Two applications of ALDAGs conclude the chapter.

Chapter 5 introduces the *R* package **stagedtrees**, which includes several algorithms for learning the structure of Staged Trees and CEGs from data. Score-based, distance-based and clustering-based algorithms are implemented, as well as various functionalities to provide inferential, visualization, descriptive and summary statistics tools for such models and about their graph structure. A simulation study for all the learning algorithms is performed on nine datasets as well as the analysis on the pediatric dentistry dataset is carried on through Staged Trees, which are able to manage its structural zeros.

Chapter 6 considers Staged Trees as classification models. Firstly, an overview about the theory of Bayesian Network Classifiers is given and Staged Tree Classifiers are presented. A classification experiment involving 14 datasets is conducted, comparing the performances of 9 Staged Tree Classifiers implemented through the *R* package **stagedtrees** with respect to those obtained with the state of the art classification algorithms.

In Chapter 7 the importance of structural or observed zero counts in contingency tables is widely investigated. A method to overcome this issue in the context of Staged Trees is proposed: remove vertices associated to zero counts from the model search space. Indeed, it is shown as computational times of considering or not these vertices for learning algorithms are highly different. To conclude the chapter, using the fact that Stratified Staged Trees are relational models [72, 73] containing an overall effect, it is shown that they are characterized by the usual properties of log-linear models.

The following manuscripts are based on this thesis:

- Carli et al. [20], submitted to *Journal of Statistical Software*;
- Carli et al. [21], submitted to *Pattern Recognition*;
- Varando et al. [121], submitted to *International Journal of Intelligent Systems*;
- Carli et al. [22], submitted to *Insurance: Mathematics and Economics*;
- Ugolini et al. [117], submitted to *Clinical Oral Investigations*;
- Varando et al. [120], an *R* package available on the Comprehensive R Archive Network (CRAN) at *https://CRAN.R-project.org/package=stagedtrees*.

The PhD was funded by the reinsurance company Swiss Re Group. Chapter 2 and Carli et al. [22] collect some of the work done on that project jointly with Dr Elena Pesce. The specific contribution which finds place in this thesis is related to the application of the estimation method based on neighborhood regression on time series data.

Chapter 1

Probabilistic Graphical Models

1.1 Introduction

Probabilistic Graphical Models (PGMs) refer to statistical models supported on a graph or network consisting of vertices or nodes connected by undirected or directed edges or arcs, where the vertices are associated to random variables and edges indicate some kind of dependence relationship between these variables. Probability distributions are associated to the graph which satisfies the conditional independence structure implied by missing edges in the graph. Indeed, a missing edge between a pair of nodes denotes that the two corresponding random variables are independent, conditionally on the other variables.

PGMs vast applicability is due to a number of factors. Firstly, graphs can visually represent the scientific content of a given model and facilitate communication between researcher and statistician. Secondly, the models are modular so that complex problems can be described and hadled by combinations of simpler elements. This has many advantages in terms of representation, inference, learning and predictions [74]. The joint distribution can be decomposed according to the factorization theorem [77]: each factor of this factorization relates to a subset of the random variables considered, each subset can be modelled separately (and thus reducing the dimensionality of the multivariate problem) and the marginal probability distribution functions can be multiplied together to return the full joint probability [10, 125]. Also Pearl [95] reasons that in real world problems it is often easier for a group of experts to identify conditional independence statements among a small number of variables instead of the joint distribution and that therefore judgements are usually made only on a subset of variables.

PGMs belong to both the fields of statistics and computer science, relying on concepts mainly from probability theory, graph algorithms, model selection and

machine learning. There are many areas on which PGMs are applied, such as medical diagnosis, image understanding, speech recognition, natural language processing, economics, ecology, biology, insurance and many others.

A PGM estimated from a dataset can be useful to confirm known independence relationships, to validate the dataset and mainly to identify unexpected relationships among the collected variables. PGMs are quick and easy to be applied in many fields, they can be estimated also in presence of latent/unobserved variables or missing values and they supply a compact and intuitive representation of conditional independences and correlations among problem factors. Furthermore, they do not only allow the encoding of probability distributions but also provide a very clear interface to interpret the model and to perform predictions. A really appealing characterization of PGMs is that they do not require necessarily the definition of a response variable, since they estimate a joint distribution and not the conditional distribution of the response variable given the covariates, as usually done in statistics or machine learning.

PGMs have been widely studied in Højsgaard et al. [62], Koller and Friedman [74], Lauritzen [77], Smith [106], Studeny [110], Whittaker [125] and many more. They are mainly distinguished in three classes: Undirected Graphs (UGs), Directed Acyclic Graphs (DAGs) or Bayesian Networks (BNs) [32, 42] and Chain Graphs (CGs) [110]. DAGs have only directed edges between vertices in the graph and do not admit cycles. Any type of discrete/continuous distribution may be associated to each variable of a DAG; however, usually for continuous variables a Gaussian distribution is assumed. BNs are one of the most commonly used Graphical Models and will be described in detail throughout this chapter. In contrast to this, UGs or Markov Fields describe conditional independence statements between the variables in the graph only through undirected edges. Examples of these are the Gaussian GMs with continuous variables or log-linear GMs with discrete variables [77]. Finally, graphs may have a mixture of directed and undirected edges, leading to the definition of the chain graph [78].

PGMs have been implemented in various softwares. In this work R is used, which is a free software environment for statistical computing and graphics. The R packages **bnlearn** by Scutari [103] and **gRain** by Højsgaard et al. [63] implement BNs. In Chapter 5 we present the R package **stagedtrees** for a novel class of PGMs which is the topic of the second part of this thesis.

Formally, a graph can be defined as a pair $G = (V, E)$, where V is a finite set of *vertices* or *nodes* of G and E is the finite set of *edges* or *arcs*. Edges can be undirected ($\alpha - \beta$), directed ($\alpha \to \beta$) or bidirected ($\alpha \leftrightarrow \beta$). Each edge $e \in E$ is

related to a pair of nodes $(\alpha, \beta) \in V$ x V and for any pair of nodes there may be more edges. The nodes of a graph represent continue or discrete random variables and a missing edge indicates a conditional independence statement as detailed in next section.

The chapter is organized as follows: in Section 1.2 the main definitions and propositions about graph theory are set out, while Section 1.3 recalls how to read independence statements from the graph of a PGM. Section 1.4 considers an application for UGMs (see Ugolini et al. [117]) and Section 1.5 presents an estimation criterion for Graphical Models based on a neighborhood regression approach which will be used in Chapter 2.

1.2 Graph Theory

In this section and in Section 1.3 we follow Lauritzen [77].

Definition 1. *(Adjacent). Two vertices α and β are said to be adjacent or neighbours, $\alpha \sim \beta$, if there is an edge between α and β in G. The set of neighbours of a vertex α is denoted as $adj(\alpha)$ or $ne(\alpha)$. Otherwise, if there is not an edge between α and β, $\alpha \not\sim \beta$, then α and β are said to be non-adjacent.*

Definition 2. *(Type of Graph). A graph can be:*

- *undirected, if all its edges are undirected;*
- *directed, if all its edges are directed;*
- *bidirected, if all its edges are bidirected;*
- *mixed, if its edges are at least of two types.*

Definition 3. *(Complete & Clique). A subset $A \subseteq V$ is complete if all pairs of vertices in A are adjacent. A graph $G = (V, E)$ is complete if the vertex set V is complete. Furthermore, a clique of a graph G is a maximal complete subset not contained in a larger complete subset. The set of cliques of a graph G is denoted by $\mathcal{C}(G)$.*

Definition 4. *(Path & Separate). A path of length n between two vertices α and β in a graph G is a set of vertices $\alpha = \alpha_0, \alpha_1, \ldots, \alpha_n = \beta$, where $\alpha_{i-1} \sim \alpha_i$ for $i = 1, \ldots, n$. If there is a path from α to β we say that α leads to β, $\alpha \mapsto \beta$.*

A subset $D \subset V$ in a graph $G = (V, E)$ is said to separate $A \subset V$ from $B \subset V$ if any path between a vertex in A and a vertex in B contains a vertex from D.

Definition 5. *(Subgraph). The graph $G_0 = (V_0, E_0)$ is said to be a subgraph of $G = (V, E)$ if $V_0 \subseteq V$ and $E_0 \subseteq E$. For $A \subseteq V$, let E_A denote the set of edges in E between vertices in A. Then $G_A = (A, E_A)$ is the subgraph induced by A.*

Definition 6. *(Boundary & Closure). The boundary of $\alpha \in V$, $bd(\alpha) = adj(\alpha)$, is the set of vertices adjacent to α. So, for undirected graph the boundary is equal to the set of neighbours $ne(\alpha)$. The closure of α, $cl(\alpha) = bd(\alpha) \cup \{\alpha\}$, is the union of the set of vertices adjacent to α and α itself.*

Definition 7. *(Parents & Children). The parents $pa(\beta)$ of a node $\beta \in V$ are those nodes $\alpha \in V$ for which $\alpha \to \beta$. The children $ch(\alpha)$ of a node $\alpha \in V$ are those nodes $\beta \in V$ for which $\alpha \to \beta$.*

The following nine definitions are useful to understand the characteristics of an estimated PGM, such for instance if it is acyclic, triangulated or decomposable.

Definition 8. *(Acyclic). A directed graph is acyclic if it has no directed cycles, that is, cycles with the arrows pointing in the same direction all the way around. In particular, it does not exist any vertex $v \in V$ that leads to itself: $v \not\to v$. So, a directed acyclic graph is called DAG.*

Definition 9. *(Chord, Chordless & Triangulated). A cycle $\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_n) = \alpha$ with adjacent elements $\alpha_i \sim \alpha_j$, with $j \notin \{i-1, i+1\}$, is said to have a chord. If it has no chords it is said to be chordless.*

A graph with no chordless of length ≥ 4 is called triangulated or chordal.

Definition 10. *(Decompose & Decomposable). A triple (A, B, D) of non-empty disjoint subsets of V is said to decompose G into $G_{A \cup D}$ and $G_{B \cup D}$ if $V = A \cup B \cup D$ and D is a complete subset of V and separates A from B.*

A graph is said decomposable if and only if:

- *it is complete;*
- *it can be decomposed into decomposable subgraphs;*
- *it is triangulated.*

Definition 11. *(Node Perfect Ordering). An ordering of nodes in a graph is called perfect ordering if $bd(i) \cap \{1, \ldots, i-1\}$ is complete for all $i \in V$. A perfect ordering of nodes in a graph exists if and only if the graph is triangulated.*

Definition 12. *(Topological Ordering). A topological order associated to a directed graph is a linear ordering of its vertices such that for every directed edge $e = (u, v)$, u comes before v in the node ordering. A topological ordering is possible if and only if the graph has no directed cycles, that is, if it is a Directed Acyclic Graph. Furthermore, any DAG has at least one topological ordering.*

Definition 13. *(Maximal Prime Subgraph Decomposition). The maximal prime subgraph decomposition of an undirected graph is the smallest subgraphs into which the graph can be decomposed.*

Definition 14. *(Markov Blanket). The Markov blanket of a vertex $v \in V$ in a DAG $G = (V, E)$ is defined as the minimal set that separates v from the remaining vertices. So, the markov blanket of v is the union of v's parents, v's children and the parents of v's children.*

1.3 Conditional Independence

Let A, B and C be three disjoint sets of vertices of a graph $G = (V, E)$. Throughout all the thesis $\boldsymbol{X}_A$ stands for the random vector associated to the indices of random variables contained in the set A and $f(\boldsymbol{x}_A)$ stands for the corresponding probability density function. Furthermore, let $\mathbb{X}_A$ be the sample space of $\boldsymbol{X}_A$ built as the product of the marginal sample spaces of variables in A. Let's indicate with q the cardinality of the set A, then:

$$
\begin{aligned}
X_A &= (X_{A_1}, \ldots, X_{A_q}) \\
f(\boldsymbol{x}_A) &= f\Big(X_{A_1}, \ldots, X_{A_q}\Big) \\
\mathbb{X}_A &= \prod_{i=1}^{q} \mathbb{X}_{A_i}.
\end{aligned}
$$

Now, two different characterizations of conditional independence (factorization criterion), indicated with $A \perp\!\!\!\perp B \mid C$, can be considered:

- $f(\boldsymbol{x}_A, \boldsymbol{x}_B \mid \boldsymbol{x}_C) = f(\boldsymbol{x}_A \mid \boldsymbol{x}_C)\, f(\boldsymbol{x}_B \mid \boldsymbol{x}_C)$,
- $f(\boldsymbol{x}_A, \boldsymbol{x}_B, \boldsymbol{x}_C) = \frac{g(\boldsymbol{x}_A, \boldsymbol{x}_C)}{h(\boldsymbol{x}_B, \boldsymbol{x}_C)}$, with the functions f, g and h positive-definite distributions.

Theorem 1 gives properties of conditional independence among sets of variables, while Theorem 2 provides a decomposition of the joint distribution of a random vector $\boldsymbol{X}_V$ according to the clique set $\mathcal{C}(G)$ of its undirected graph $G = (V, E)$.

Theorem 1. *Let A, B, C and D be four sets of variables. Then, the following fundamental properties of conditional independence can be defined:*

- *symmetry:* $A \perp\!\!\!\perp B \mid C \;\rightarrow\; B \perp\!\!\!\perp A \mid C$;
- *reduction:* $A \perp\!\!\!\perp B \mid C$ *and* $D \subseteq B \;\rightarrow\; A \perp\!\!\!\perp D \mid C$;
- *weak union:* $A \perp\!\!\!\perp (B \cup C) \mid D \;\rightarrow\; A \perp\!\!\!\perp B \mid (C \cup D)$;
- *contraction:* $(A \perp\!\!\!\perp C \mid B)$ *and* $(A \perp\!\!\!\perp D \mid B \cup C) \;\rightarrow\; A \perp\!\!\!\perp (C \cup D) \mid B$.

Theorem 2. *Let $G = (V, E)$ be an undirected graph and $\mathcal{C}(G)$ its clique set. Then, assuming that the joint distribution $f(\boldsymbol{X}_V)$ is strictly positive, it factorizes w.r.t. G if*

$$f(\boldsymbol{X}_V) \;=\; \prod_{c \in \mathcal{C}(G)} \psi_c(\boldsymbol{X}_c), \tag{1.1}$$

where ψ_c depends on $\boldsymbol{X}_V$ only through $\boldsymbol{X}_c$.

Independence statements can be read from UGMs using Definition 15, 16 and 17.

Definition 15. *(Pairwise Markov Property). Let α and β be two vertices of an undirected graph $G = (V, E)$. If α and β are not adjacent, then the variables corresponding to α and β are said to be independent conditionally on all the other variables in V.*
In symbols, $\alpha \not\sim \beta \;\rightarrow\; \alpha \perp\!\!\!\perp \beta \mid V \setminus \{\alpha, \beta\}$.

Definition 16. *(Local Markov Property). Let $\alpha \in V$ be a vertex of an undirected graph $G = (V, E)$. For any $\alpha \in V$, α is independent from all other variables except the closure of α, conditionally on the boundary of α.*
In symbols, $\forall\, \alpha \in V : \alpha \perp\!\!\!\perp V \setminus cl(\alpha) \mid bd(\alpha)$.

Definition 17. *(Global Markov Property). If two sets of variables $A \subset V$ and $B \subset V$ are separated by a set $C \subset V$ in an undirected graph $G = (V, E)$, then A is said to be independent from B given C.*
In symbols, A and B are separated by $C \;\rightarrow\; A \perp\!\!\!\perp B \mid C$.

Proposition 1 allows to rewrite that decomposition for Directed Acyclic Graphs $G = (V, E)$ by using topological orders of variables induced by the edges structure E of that graph. Note that a DAG can have more than one topological ordering of the variables on which it is estimated.

Proposition 1. *(Factorization Criterion for DAGs). The set of variables $\boldsymbol{X}_V$ corresponding to nodes in V may be ordered in such way that the joint distribution $f(\boldsymbol{X}_V)$ factorizes as*

$$f(\boldsymbol{X}_V) \ = \prod_{v \in V} \ f(X_v \mid \boldsymbol{x}_{pa(v)}) \tag{1.2}$$

for some sets of variables $\{pa(v)\}_{v \in V}$ such that the variables in $pa(v)$ precede v in the node ordering. This order is known as topological order, as introduced in Definition 12.

Proposition 2 states that if two disjoint sets of variables A and B are separated by a third one C in a graph G, then they are independent conditionally on C according to the underlying statistical model.

Proposition 2. *(D-separation for DAG). Two sets of variables $A \subset V$ and $B \subset V$ are d-separated by a set $C \subset V$ if and only if they are separated in the graph formed by moralizing the anterior graph of $A \cup B \cup C$. If A and B are d-separated by the set C, then $A \perp\!\!\!\perp B \mid C$ under the model.*

1.4 An Application in a Pediatric Dentistry Framework

This section presents an application of UGM to model complex interactions among factors affecting early childhood caries (ECC) development. The dataset is provided by Dr. Alessandro Ugolini from the Department of Integrated Surgical and Diagnostic Sciences of the University of Genoa. "Early childhood caries is defined as the presence of one or more decayed (non-cavitated or cavitated lesions), missing or filled surfaces, in any primary tooth of a child under six years of age" [98]. ECC is the most common chronic infectious disease of childhood with the prevalence up to 65% and is becoming a serious public health problem in both developing and industrialized countries. ECC can begin early in life at around three years old and progresses rapidly in those who are at high risk, and often goes untreated. Its consequences can affect the immediate and long-term quality of life of the child and family and can have significant social and economic consequences.

The dataset is from an observational study and it enrolls all children born in 2008 and 2009, attending one of the 10 kindergartens in Chiavari (Genoa). The design study, data collection, error of methods and data preparation are described in Ugolini et al. [117]. The dataset considered here has 234 observations and 8 variables described in Table 1.1, while in Table 1.2 the marginal distributions of the

eight variables are reported in percentage. Two variables are binary, four ternary and two variables have four levels for a total of $4^2 3^4 2^2$ possible combinations of values.

The UGM approach is adopted with the purpose of validating existing knowledge on the interaction of the risk factors associated with ECC, of investigating the existence of further interactions and of assessing the overall relationship among ECC risk factors. In Section 1.4.1 the results obtained through the UGM are commented in detail, while in Section 1.4.2 an exhaustive final discussion about the analysis carried out on this study is reported.

Variable Name	Variable Type	Variable Definition
Oral hygiene status	Binary	Is the Oral hygiene status of the child adequate or not adequate?
Caries variation	Binary	Has the child increased the number of caries at age five, with respect to age three?
Breastfeeding type	Ternary	*Which type of breastfeeding has the child received? No breastfeeding, exclusive or mixed (also with feeding bottle) breastfeeding?*
Breastfeeding time	Quaternary	**For how many months has the child been breastfed? This is a categorization of the collected variable.**
Use of Pacifier	Quaternary	**For how many months has the child used the pacifier? This is a categorization of the original variable.**
Frequency of toothbrushing	Ternary	*How many times per day does the child brush his/her teeth 1, 2 or more than 2?*
Consumption of sugared beverages	Ternary	Does the child drink sugary or carbonated sodas 1 time at day, 1 time at week or occasionally?
Consumption of vegetables/fruits	Ternary	Does the child eat vegetables 1 time at day, 1 time at week or occasionally?

Table 1.1: Variables for the UG modelling constructed following associative and explorative data analysis. The construction of the variables from the original ones in Table 7.5 in the appendix is illustrated in the text. Here, in italics the same variables as in Table 7.5 and in bold a categorization of the original ones. The others are calculated starting from different original variables in Table 7.5.

1.4.1 Data Analysis

The UGM in Figure 1.1 gives the factorization of the joint probability distribution of $\boldsymbol{X}$ shown in Equation (1.3), where $P(\boldsymbol{X} = \boldsymbol{x})$ is the probability that the eight variables take the value $\boldsymbol{x}$. The UGM is obtained through a stepwise procedure wich checks at each step of the algorithm if an addition or a deletion of an edge can be carried out in order to minimize the BIC (Bayesian Information Criterion).

Hence it follows that, under the UGM, the eight variables are decomposed into three macro factors. The factor $P(X_V = x_V)$ is the probability that the variable Consumption of vegetables/fruits takes value x_V, with x_V that can be equal to 1, 2 or 3 (Table 1.2). Similarly, $P(X_B = x_B)$ is the estimated probability of the variable Frequency of toothbrushing (X_B). The remaining six factors are collected in $\boldsymbol{X}_R$.

Caries variation	Percentage	Oral hygiene status	Percentage
0 (no variation)	80.34	1 (adequate)	84.19
1 (variation)	19.66	2 (not adequate)	15.81
Breastfeeding type		Frequency of toothbrushing	
0 (not breastfed)	18.38	1 (once a day)	27.35
1 (exclusive breastfed)	58.12	2 (twice a day)	60.68
2 (breast and bottled fed)	23.50	3 (more than twice a day)	11.97
Consumption of sugared beverages		Consumption of vegetables/fruits	
1 (daily)	29.91	1 (daily)	41.45
2 (weekly)	56.84	2 (weekly)	53.42
3 (occasionally)	13.25	3 (occasionally)	5.13
Breastfeeding time		Use of Pacifier	
0 (0 months)	18.38	0 (0 months)	31.62
1 (1-6 months)	30.34	1 (1-36 months)	29.06
2 (7-12 months)	36.32	2 (36-48 months)	24.79
3 (>12 months)	14.96	3 (>48 months)	14.53

Table 1.2: Relative/percentage univariate distributions of the eight selected variables used for the UG modelling.

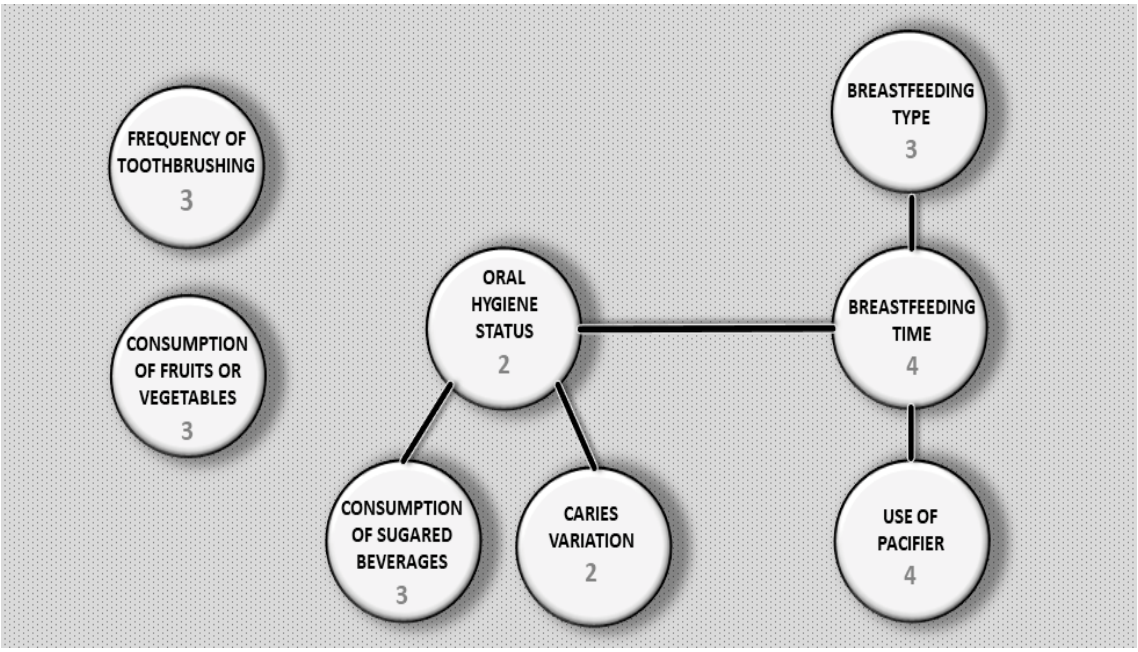


Figure 1.1: Estimated UGM on the eight selected variables. Inside the node representing each variable, the number of levels that it takes is reported.

$$P(\boldsymbol{X} = \boldsymbol{x}) \;=\; P(X_V \;=\; x_V)\; P(X_B \;=\; x_B)\; P(\boldsymbol{X}_R \;=\; \boldsymbol{x}_R) \tag{1.3}$$

In a framework with only categorical variables and having assumed the probability of each instance of $\boldsymbol{X}$ strictly positive, in its simplest form an UGM could be seen as a visualization of the more popular log-linear model [62, 77]. The assumption of strict positivity is very strong and motivates the development of the statistical

model classes called Staged Trees and Chain Event Graphs (see Chapters 3 and 7). The parameters of the log-linear model for the six connected variables in Figure 1.1 correspond to the intercept, one for each node (that is, each variable or main effect), one for each arc (that is, each two-way interaction). No three-way or higher level of interaction among variables has to be included in the log-linear model because in the estimated UGM's graph in Figure 1.1 there are no triangles nor higher order of fully connected nodes, equivalently not higher order interaction was detected by the estimated UGM model (see e.g. Højsgaard et al. [62]). In formulae, the log-linear model associated to the best fitting UGM for the considered dataset can be expressed as in Equation (1.4), where S = Consumption of sugared beverages, O = Oral hygiene status, C = Caries variation, TIME = Breastfeeding time, TYPE = Breastfeeding type and P = Use of Pacifier. Furthermore, a generic instance $\boldsymbol{x}$ is defined according to the product space defined by the product of the sample spaces of the six connected variables, i.e. $\boldsymbol{x} = (i,\ j,\ k,\ l,\ m,\ n)$, with $i = 1, 2, 3,\ j = 1, 2,\ k = 0, 1,\ l = 0, 1, 2, 3,\ m = 0, 1, 2$ and $n = 0, 1, 2, 3$ the levels taken by S, O, C, TIME, TYPE and P, respectively. $P(\boldsymbol{x})$ is the joint probability of the occurrence of a generic instance $\boldsymbol{x}$ and u is a constant.

$$\begin{aligned} \log P(\boldsymbol{x}) = u &+ u_i^{\mathrm{S}} + u_j^{\mathrm{O}} + u_k^{\mathrm{C}} + u_l^{\mathrm{TIME}} + u_m^{\mathrm{TYPE}} + u_n^{\mathrm{P}} + \\ &u_{ij}^{\mathrm{S\ O}} + u_{jk}^{\mathrm{O\ C}} + u_{jl}^{\mathrm{O\ TIME}} + u_{lm}^{\mathrm{TIME\ TYPE}} + u_{ln}^{\mathrm{TIME\ P}} \end{aligned} \tag{1.4}$$

Model Verification

Statistical hypothesis tests can be carried out in order to evaluate if a simpler model with respect to the fitted UGM can be estimated, more precisely if any arc in the graph in Figure 1.1 can be deleted. The F-test tests whether a single variable or an interaction term can be removed from the model without information loss. The null hypothesis is that the UG model and a simpler model are statistically equivalent, so the latter is preferred because it has fewer parameters. Instead, the alternative hypothesis states that the removal of a term from the original UG model produces a significant information loss, so the UG model is preferable.

A preliminary study on the eight selected variables is also carried out within the framework of contingency table analysis and is summarized with the Fisher's exact statistical hypothesis test for count data [87]. This test is used to examine whether an association between two nominal variables in a sample is unlikely to reflect the same association in the population from which the sample is extracted.

Table 1.3 summarises the p-values for the Fisher's exact tests for all pairs of the

eight selected variables. Five marginal dependences are detected by the test and highlighted in Table 1.3. The strongest ones are four: Breastfeeding type and time, Oral hygiene status and Consumption of sugared beverages, Oral hygiene status and Caries variation, Caries variation and Consumption of sugared beverages. The p-values for the Fisher's exact test are carried out in order to examine the most interesting three-way associations and are reported in Table 1.4 and Table 1.5. The first refers to independence of some pairs of problem variables conditional on Oral hygiene status, while the second is related to conditioning on Breastfeeding time. In contrast to the marginal associations detected in Table 1.3, these specific analyses on conditional sub-populations do not support any associations. Instead, they suggest independence of the three pairs of variables in the rows of Table 1.4 and Table 1.5 conditional on Oral hygiene status and Breastfeeding time, respectively.

	X_B	X_O	X_S	X_V	X_{TIME}	X_O	X_C
X_{TYPE}	0.586	0.436	0.374	0.875	<0.001***	0.126	0.104
X_B		0.837	0.613	0.095	0.453	0.067	0.787
X_O			<0.001***	0.515	0.057	0.164	<0.001***
X_S				0.718	0.473	0.361	<0.001***
X_V					0.429	0.122	0.760
X_{TIME}						0.008**	0.110
X_P							0.824

Table 1.3: P-values of the Fisher's exact tests for marginal independences. P-values smaller than 0.001 suggest the four strongest relations. * Significant ($p \leq 0.050$), ** Significant ($p \leq 0.010$) and *** Significant ($p \leq 0.001$).

		Oral hygiene status	
		1 (adequate)	2 (not adequate)
Consumption of sugared beverages	Caries variation	0.101	1.000
Breastfeeding time	Caries variation	0.505	0.108
Breastfeeding time	Consumption of sugared beverages	0.928	1.000

Table 1.4: P-values of Fisher's exact tests for the most interesting conditional independences given the Oral hygiene status. Not Significant p-values suggest independence relationships among pair of variables (left) given a specific level of Oral hygiene status. Since for all three pairs of variables the p-values are not significant for both the levels of the oral hygiene, the conclusion is that these pairs of variables are conditional independent.

The F-test statistics for all sub-models of the UGM in Figure 1.1 are reported in Table 1.6. Only the main effect of the use of pacifier could be removed, due to a p-value greater than the reference level 0.05. But Use of Pacifier is statistically significant in the interaction with Breastfeeding time. Hence, Use of Pacifier is kept in the model, which is taken to be hierarchical, so as main effects are included in

		Breastfeeding time			
		0 (0 months)	1 (1-6 months)	2 (7-12 months)	3 (>12 months)
Oral hygiene status	Use of Pacifier	0.069	0.326	0.052	0.933
Oral hygiene status	Breastfeeding type	1.000	0.677	1.000	0.708
Use of Pacifier	Breastfeeding type	1.000	1.000	1.000	1.000

Table 1.5: P-values of Fisher's exact tests for the most interesting conditional independence given Breastfeeding time. Not Significant p-values suggest independence relationships among pair of variables (left) given a specific level of Breastfeeding time. Since for all three pairs of variables the p-values are not significant for all the four levels of the time of the breastfeeding, the conclusion is that these pairs of variables are conditional independent.

the model for any two-way interaction in the model. In conclusion, the estimated UGM is also supported by the F-tests and the Fisher's exact tests.

	Df	Deviance	AIC	F-value	P-value
UGM	-	135.82	419.57	-	-
Oral hygiene status	1	252.43	534.18	465	<0.001***
Consumption of sugared beverages	2	313.52	593.27	355	<0.001***
Breastfeeding time	3	177.68	455.43	56	<0.001***
Breastfeeding type	2	230.30	510.05	189	<0.001***
Use of pacifier	3	137.00	414.76	2	0.194
Caries variation	1	329.82	611.58	774	<0.001***
Oral hygiene status - Consumption of sugared beverages	2	287.02	566.77	302	<0.001***
Oral hygiene status - Caries variation	1	279.48	561.24	573	<0.001***
Oral hygiene status - Breastfeeding time	3	143.22	420.98	10	<0.001***
Breastfeeding time - Breastfeeding type	6	361.07	632.82	150	<0.001***
Breastfeeding time - Use of pacifier	9	158.62	424.37	10	<0.001***

Table 1.6: F-tests for the estimated UGM. Each row corresponds to a test that tests if a single variable or an interaction between pair of variables can be removed from the estimated UGM without losing information. Significant p-values suggest that the preferable model is the UGM.

The main findings implied by the UGM in Figure 1.1 and the factorization in Equation (1.3) can be listed:

1. Consumption of vegetables/fruits and Frequency of toothbrushing are marginally independent with respect to the other six variables. This implies that further analyses can be focused on the six connected variables. This factorization establishes that X_V, X_B and $\boldsymbol{X}_R$ are mutually independent random vectors, i.e. for the children we analysed, brushing teeth frequency has limited relevance for the oral hygiene status and the variation of number of caries. The joint probability distribution of the six connected variables $\boldsymbol{X}_R$ can be further factorized according to the structure of the graph in Figure 1.1 [77]. This factorization is not reported here. It is sufficient for the purpose of this study to note that

each factor corresponds to an arc, nodes connecting arcs play the special role of separators and that there are no cycles. Consequences of these facts are illustrated next.

2. Main conclusions can be drawn from the analysis of the six remaining variables are: Consumption of sugared beverages and Caries variation are conditionally independent given Oral hygiene status, that is the effect of the consumption of sugared beverages on caries development is mediated by the oral hygiene status. Furthermore, for caries development the quality of oral hygiene is more relevant than the frequency of toothbrushing. Conditionally on Breastfeeding time, the three variables Oral hygiene status, Breastfeeding type and Use of Pacifier are independent. In particular, Use of Pacifier is independent from Breastfeeding type given Breastfeeding time, that is the type of breastfeeding is a risk factor for the development of caries but simply mediated by the duration of breastfeeding. In the same way, the role of the use of pacifier in the caries development is mediated by the duration of breastfeeding and it resulted statistically significant only when considered in combination with breastfeeding time.

3. Five two-way contingency tables suffice to the understanding of the relationships among the six connected variables and no higher order table is needed. This is because only five arcs and no cycle are present in the UGM in Figure 1.1. The table matching type and time of the breastfeeding is reported on Table 1.7. It is the most troublesome for estimation because it includes five structural zeros: indeed, Breastfeeding time is zero for children that have not been breastfed.

4. Hard to estimate parameters are associated with cells with zero counts: the R package **gRim** used to carry out this study returns parameters estimation and the model diagnostics are satisfactory. This is the weak point in the analysis with UGM (see Chapters 3 and 7).

5. The UGM in Figure 1.1 has 38 parameters: 21 for two-way interactions (arcs), 16 for the linear terms (nodes) and 1 for the constant term. This is a drastic reduction with respect to the $4^2 3^4 2^2$ (5184) possible combinations of the eight selected variable levels. As already stated in point 1 above, the analysis can be carried out without the two variables Consumption of vegetables/fruit and Frequency of toothbrushing.

		Breastfeeding type		
		0	1	2
Breastfeeding time	0	18.38	0.00	0.00
	1	0.00	23.08	7.26
	2	0.00	25.64	10.68
	3	0.00	9.40	5.56

Table 1.7: Relative/percentage distribution of joint distribution of Breastfeeding type and time. It highlights structural zeros.

Focus on Caries Variation

To investigate the relevance of the seven collected variables on the caries variation, an odds-ratio analysis was carried out. The odds-ratios and the corresponding p-values are reported in Table 1.8. It shows a strong relation between caries variation and oral hygiene status (p-value $<$ 0.001 and odds-ratio $>$ 50) and between caries variation and consumption of sugared beverages (p-value $<$ 0.001 and odds-ratio $>$ 50). However, unlike what most would expect, no statistical direct evidence has been found to support a relation between the variation of caries and toothbrushing frequency.

1.4.2 Discussion

This is the first caries study in which dental caries risk factors are assessed by Probabilistic Graphical Models. UGM analysis has provided useful insights regarding specific interactions between risk factors in caries development and valuable biological information that could be missed otherwise.

Firstly, in this study Consumption of sugared beverages variable, a proxy for sugar intake, results to be conditional independent from Caries variation given Oral hygiene status and also that Frequency of toothbrushing is marginally independent from the variable Caries variation. This means that the quality of oral hygiene (and therefore the plaque index) and not the toothbrushing frequency is a primary factor that modulates (+ or -) the sugar consumption in his primary role of the ECC developer.

Secondly, in Kumar et al. [75] the authors reported that individuals who state that they brush their teeth infrequently are at a greater risk for the incidence or increment of new carious lesions than those brushing more frequently. But they considered only toothbrushing frequency as the main outcome because their interest was on whether toothbrushing frequency was predictive of the development of carious lesions, without any consideration for the correctness and efficiency of the

Breastfeeding type	odds-ratio	left-end confidence interval	right-end confidence interval	p-value
1 VS 0	2.528	0.918	8.939	0.101
2 VS 0	3.329	1.086	12.539	0.048*
2 VS 1	1.317	0.619	2.718	0.463
Breastfeeding time				
1 VS 0	2.395	0.791	8.937	0.148
2 VS 0	2.437	0.832	8.929	0.132
3 VS 0	4.469	1.360	17.608	0.019*
2 VS 1	1.018	0.462	2.269	0.965
3 VS 1	1.866	0.732	4.703	0.185
3 VS 2	1.833	0.740	4.446	0.182
Frequency of toothbrushing				
2 VS 1	0.800	0.390	1.694	0.550
3 VS 1	0.974	0.311	2.782	0.962
3 VS 2	1.217	0.415	3.151	0.700
Oral hygiene status				
2 VS 1	673.200	126.338	12581.880	0.001***
Consumption of sugared beverages				
2 VS 1	0.653	0.233	1.904	0.419
3 VS 1	270.000	47.232	5196.620	0.001***
3 VS 2	413.333	75.672	7796.548	0.001***
Consumption of vegetables/fruits				
2 VS 1	0.868	0.445	1.704	0.678
3 VS 1	1.283	0.267	4.774	0.726
3 VS 2	1.478	0.310	5.413	0.580
Use of Pacifier				
1 VS 0	1.113	0.495	2.507	0.794
2 VS 0	0.722	0.282	1.768	0.483
3 VS 0	1.020	0.355	2.721	0.970
2 VS 1	0.649	0.252	1.594	0.353
3 VS 1	0.916	0.317	2.453	0.865
3 VS 2	1.412	0.458	4.215	0.537

Table 1.8: Odds-ratios of Caries variation with respect to the other seven selected variables. Significant p-value associated to an odds-ratio highlights a useful variable in order to predict the variation of the number of caries. 95% Confidence intervals for odds-ratios are also calculated.

oral hygiene. This introduced an important bias in the evaluation of the role of oral hygiene in caries development. The present studies linked toothbrushing frequency, the quality of oral hygiene (checked by a senior trained dentist) and the caries development also with diet habits and made evident that the quality of oral hygiene plays a pivotal role in ECC development and that it can act as a protective factor in the ECC development. These results suggested that the community- and school-based oral health programs should focus also on the quality of oral hygiene and not only on the toothbrushing frequency. Indeed, here the odds-ratio study confirmed the findings reported in the literature: a direct relation has been found between ECC development and oral hygiene and the consumption of sugared beverages. Instead, no statistical evidence has been found to support a relation between ECC development and toothbrushing frequency. It is widely believed that effective removal of dental biofilm by toothbrushing can reduce the development of new carious lesions, but the evidence base is weak, especially because most of the results come from studies that considered the frequency of brushing and not the quality of oral hygiene [75].

Based on the results, the oral health education of children and parents in dentist's clinical practice and the community- and school-based oral health programs for ECC prevention should be improved with supervised toothbrushing program.

Thirdly, another group of interactions identified by the UGM is that ECC development (Caries variation) is conditionally independent from Type of breastfeeding given Oral hygiene status and also from the Breastfeeding time and Use of Pacifier. In the present study the odds-ratio analysis found moderated correlation between ECC development, type of breastfeeding (p-value = 0.048 and odds-ratio = 3.329) and length of the period of breastfeeding (p-value = 0.019 and odds-ratio = 4.469, Table 1.8. Therefore, it appears that the type of breastfeeding and the use of pacifier are risk factors for the development of caries but mediated by the duration of breastfeeding and oral hygiene status, resulting both statistically significant only when considered in combination with breastfeeding time. These are interesting findings on a very debated topic in the literature. Among the ECC risk factors not only the type, but also the duration of feeding represents a critical issue in literature. The UGM again displayed on this topic the pivotal role of the quality of oral hygiene (this information is lost in commonly used statistical models) and how it is able to act as a protective factor in ECC development (Figure 1.1). Thus, in order to decrease the ECC incidence, the antenatal and postnatal educational interventions to mothers for breastfeeding practices (focusing on the duration of the breastfeeding period, which has been shown to be one of the most important risk factors), need to be supported by incorporating mother and child oral health promotion to reduce caries experience, improve oral hygiene and dietary habits.

1.5 Estimation through Neighborhood Regression

In this chapter only the estimation criterion based on neighborhood regression is explained in detail since it has been used for a real application in Chapter 2.

The neighborhood regression approach is described in Haslbeck and Waldorp [57] and is implemented in the *R* package **mgm**. Haslbeck and Waldorp studied four different scenarios:

- Mixed Graphical Models (MGMs) on stationary time series.
- Mixed Graphical Models (MGMs) on time-varying time series.
- Mixed Vector Autoregressive Graphical Models (mVAR GMs) on stationary time series.

- Mixed Vector Autoregressive Graphical Models (mVAR GMs) on time-varying time series.

Consider a p-dimensional random vector $\boldsymbol{X}$ with each variable X_s taking values in a potentially different set $\mathbb{X}_s$, and let $G = (V, E)$ be an Undirected Graph over p nodes corresponding to the p variables. Assuming that each variable $X_s \in \mathbf{X}$ belongs to an exponential family, $s = 1, \ldots, p$, the factorization of the joint distribution seen in Equation (1.1) can be rewritten as displayed in Equation (1.5), where $\boldsymbol{\theta}_c$ are parameters associated to the clique functions, $\phi_c(\boldsymbol{X}_c) = \log \psi_c(\boldsymbol{X}_c)$ are sufficient statistics and $\Phi(\boldsymbol{\theta})$ the log-normalization constant [57].

$$\mathbb{P}(\boldsymbol{X}) = \exp\left[\sum_{c \in \mathcal{C}(G)} \boldsymbol{\theta}_c \phi_c(\boldsymbol{X}_c) - \Phi(\boldsymbol{\theta})\right] \tag{1.5}$$

Denote with $N(s) = \{t \in V \mid (s, t) \in E\}$ the neighborhood of a generic node $s \in V$ in a graph $G = (V, E)$, and with $\boldsymbol{X}_{\setminus\{s\}} = \boldsymbol{X}_{V \setminus \{s\}}$ the set of all variables in $\mathbf{X}$ excepted the one corresponding to the vertex s.

Definition 18. *(Node-conditional Distribution). The node-conditional distribution of each node $X_s \in \boldsymbol{X}$ given $\boldsymbol{X}_{\setminus\{s\}}$ is given by an arbitrary univariate exponential family distribution and is shown in Equation (1.6), where $\Phi_s(\cdot)$ is the sufficient statistic, $B_s(\cdot)$ is the base measure and $E_s(\boldsymbol{X}_{\setminus s})$ the canonical parameter.*

$$\mathbb{P}(X_s | \boldsymbol{X}_{\setminus s}) = exp\left[E_s(\boldsymbol{X}_{\setminus s})\ \phi_s(X_s)\ +\ B_s(X_s)\ -\ \Phi(\boldsymbol{X}_{\setminus s})\right] \tag{1.6}$$

Note that $\Phi_s(\cdot)$ and $B_s(\cdot)$ are characterized by the exponential family to which the node s belongs and $E_s(\boldsymbol{X}_{\setminus s})$ is a function of all variables in $\boldsymbol{X}$ excepted X_s.

Definition 19. *(Mixed Graphical Model). A Graphical Model $G = (V, E)$ estimated on a random vector $\boldsymbol{X}$ with components X_s that can be continuous, count or categorical variables, for $s \in V$, is called a mixed Graphical Model (MGM).*

Definition 20. *(k-order Mixed Graphical Model). A MGM $G = (V, E)$ is said to have order k if each clique c in the clique set $\mathcal{C}(G)$ contains at most k nodes.*

Proposition 3. *The node-conditional distributions in Equation (1.6), for each $X_s \in \boldsymbol{X}$, are consistent with the joint distribution in Equation (1.5), which is Markov with respect to the graph $G = (V, E)$ with corresponding clique set $\mathcal{C}(G)$ of maximal dimension k for each clique $c \in \mathcal{C}(G)$, if and only if the canonical parameters $\{E_s(\cdot)\}_{s \in V}$ are a linear combination of products of univariate sufficient statistic functions $\{\phi(X_r)\}_{r \in N(s)}$ of maximal order k, that is:*

$$E_s(\boldsymbol{X}_{\setminus s}) = \theta_s + \sum_{r \in N(s)} \theta_{sr}\phi_r(X_r) + \ldots\ldots + \sum_{r_1,\ldots,r_{k-1} \in N(s)} \theta_{r_1,\ldots,r_{k-1}} \prod_{j=1}^{k-1} \phi_{r_j}(X_{r_j}), \quad (1.7)$$

where $\boldsymbol{\theta}_{s\cdot} = \{\theta_s,\ \theta_{sr},\ \ldots,\ \theta_{s,r_1,\ldots,r_{k-1}}\}$ *are the canonical parameters and* $N(s)$ *is the set of neighborhoods of node* s *according to* G*.*

Finally, by multiplying p conditional distributions, the joint distribution of $\mathbf{X}$ can be factorizes as in Equation (1.8).

$$\begin{aligned} \mathbb{P}(\mathbf{X}) = \exp \Bigg[& \sum_{s \in V} \theta_s \phi_s(X_s) \ + \ \sum_{s \in V} \sum_{r \in N(s)} \theta_{sr} \phi_s(X_s)\phi_r(X_r) \ + \ \ldots \\ & + \ \sum_{r_1,\ldots,r_k \in C} \theta_{r_1,\ldots,r_k} \ \prod_{j=1}^{k} \phi_{r_j}(X_{r_j}) \ + \ \sum_{s \in V} B_s(X_s) \ - \ \Phi(\boldsymbol{\theta}) \Bigg] \end{aligned} \quad (1.8)$$

The canonical parameters $\boldsymbol{\theta}_{s\cdot}$ for each node $s \in V$ could have different dimensions according to the exponential family of the corresponding variable X_s or according to whether X_s is a discrete or continuous variable. For instance, let's discuss the simplest version of the generical k dimensional clique set: $k = 2$, that is only cliques with at most two nodes are considered. Only three different scenarios can occur between a generic pair of variables X_s and X_r:

1. X_s and X_r both continuous variables: only $R = 1$ parameter θ_{sr} is sufficient to monitor their interaction relation;

2. X_s and X_r both categorical variables with m and u levels, respectively: $R = (m-1) \times (u-1)$ parameters are needed to monitor their interaction;

3. X_s continuous variable and X_r discrete variable with m levels: $R = (m-1)$ parameters are needed to estimate their interaction.

All this reasoning can be formalized denoting with θ_{sr}^z the parameter defining the interaction between nodes s and r indexed by $z \in \{1, \ldots, R\}$.

Proposition 4. *(Presence of an edge). An edge is present between* s *and* r *if and only if there exists at least one parameter* θ_{sr}^z *different from zero:* $(s,r) \in E \iff \exists\ z\ \ s.t.\ |\theta_{sr}^z| > 0$.

For a k-order MGM, an edge between s and r is a function of all cliques of size at most k that include both s and r. In particular, for pairwise ($k = 2$) MGM is

sufficient to build an undirected graph to represent the two-way interactions, while for a $k > 2$ MGM a factor graph (see Definition 21) is needed to represent all two-way and higher-order interactions, since with an undirected graph one can not understand if an interaction relationship is present due to a two-way or an higher-order relation.

Definition 21. *(Factor Graph). A Factor Graph is a bipartite graph representing the factorization of the joint distribution of* $\mathbf{X}$*. A bipartite graph is a graph whose vertices can be divided into two disjoint and independent sets* A *and* B *such that every edge connects a vertex in* A *to one in* B*.*

1.5.1 Parameters Estimation

The joint distribution in Equation (1.8) is the product of univariate conditional distributions. Since all univariate conditional distributions in Equation (1.6) are members of exponential families, it is possible to estimate the joint distribution by a series of regressions, one for each variable, as in the Generalized Linear Model (GLM) framework [93]. Indeed, the internal sums in Equation (1.6) are related to the neighborhood $N(s)$ for each $s \in V$; first, they are estimated and then they are combined to provide an estimation of the joint distribution of $\boldsymbol{X}$.

The lasso estimates of $\boldsymbol{\theta}$ are computed as

$$\hat{\boldsymbol{\theta}} \;=\; \arg\min_{\boldsymbol{\theta}} \Big[-\, l(\boldsymbol{\theta}, \mathbf{X}) \;+\; \lambda \, \|\boldsymbol{\theta}\|_1 \Big], \tag{1.9}$$

where $l(\boldsymbol{\theta}, \mathbf{X})$ is the log-likelihood function, $\|\boldsymbol{\theta}\|_1 = \sum_{j=1}^{J} \|\theta_j\|$ is the l_1 norm of the parameter vector $\boldsymbol{\theta}$, J is its length and λ is the penalization hyper-parameter [59]. The l_1 penalty ensures that the model is identified also in high-dimensional settings where the number of variables p is greater than the number of observations n [60].

The *design matrix* $\boldsymbol{X}$ is defined with respect to the node-conditional distribution of s and the maximal interaction order k allowed by the Mixed Graphical Model. For instance, for each node $s \in V$, for $k = 2$ the design matrix contains all the other variables $\boldsymbol{X}_{\setminus s}$ or their respective indicator functions if they are categorical; for $k = 3$ the same structure described for $k = 2$ plus the interaction terms among all the variables in $V_{\setminus s}$ or their respective indicator functions.

In Proposition 4 the criterion to use for establishing whether an edge is present or not has been described. Now it remains to detail how the estimates of parameters are combined, since multiple parameters are generally required to monitor an interaction between non-continuous variables. Hence, when the interaction is among a pair of continuous variables the estimation of the corresponding parameter is taken, while

when the interaction involves categorical variables the mean of the absolute value of the estimates is taken. For example, for a pairwise interaction between nodes s and r, one parameter θ_{sr} is estimated from the regression on s and another one (θ_{rs}) from the regression on r. To obtain a conditional dependence graph G, these values has to be combined in some way to produce a final *weight*. This can be done in two different ways:

- **OR-rule:** arithmetic mean between the two estimates of parameters:

$$\hat{\theta}(sr) = \frac{\hat{\theta}_{sr} + \hat{\theta}_{rs}}{2}$$

- **AND-rule:** arithmetic mean between the two estimates of parameters if they are both different from zero, otherwise the estimate of their relation is set to be zero:

$$\hat{\theta}(sr) = \begin{cases} \frac{\hat{\theta}_{sr}+\hat{\theta}_{rs}}{2}, & \text{if } \hat{\theta}_{sr} \neq 0 \text{ and } \hat{\theta}_{rs} \neq 0 \\ 0, & \text{otherwise} \end{cases}$$

The formulations of the OR/AND rules can be intuitively generalized for a generic k MGM.

The penalized parameter λ can be selected using the cross validation or a model selection criterion as the *Extended Bayesian Information Criterion (EBIC)*, which is shown in Equation (1.10), where $\|\boldsymbol{\theta}\|_0$ is the number of parameters estimated from the proposed model, i.e. the nonzero neighbors, and γ a tuning parameter. Note that if $\gamma = 0$ the EBIC corresponds to the classical BIC [102].

$$\text{EBIC}_\gamma(\boldsymbol{\theta}) \;=\; -\,2\; l(\hat{\boldsymbol{\theta}}, \mathbf{X}) \;+\; \|\boldsymbol{\theta}\|_0 \log n \;+\; 2\;\gamma\;\|\boldsymbol{\theta}\|_0 \log p \tag{1.10}$$

Pseudo-Algorithm 1. (MGMs estimation via neighborhood regression).

1. *for each node $s \in V$:*
   (a) *build the design matrix defined by the order k of the MGM;*
   (b) *solve the lasso problem in Equation (1.9) with penalization hyper-parameter λ.*
   (c) *aggregate interactions with several parameters into a single edge-weight.*
2. *Combine the estimates of parameters with the OR or AND rule obtaining $\hat{\boldsymbol{\theta}}$.*
3. *Define G according to $\hat{\boldsymbol{\theta}}$.*

1.5.2 Mixed Vector Autoregressive Graphical Models

This neighborhood regression can be applied also in a framework with time dependent variables, i.e. time series. In particular, in mixed Vector Autoregressive (mVAR) Graphical Models, each variable $X_s \in \mathbf{X}$ detected at a temporal instant t is modelled as a linear combination of all variables in $\mathbf{X}$ (s included) detected at a previous temporal instant $t-l$, where l stands for the temporal lag in which one is interested in.

The theoretical assumptions and reasonings done throughout the section are also valid for the autoregressive context, except for one big difference: here, the canonical parameter $E_s(X_{\setminus s})$ in Equation (1.6) is not a function of parameters associated to interactions between variables detected at the same current time t, but of a previous instant $t-l$. This is formularized below, where L indicates the set of lags one wants to investigate:

$$E_s^t(\mathbf{X}) \;=\; \theta_s \;+\; \sum_{j \in L} \sum_{r \in N(s)} \theta_{sr}^{t-j}\, \phi_r(X_r^{t-j}).$$

Two changes can be applied to the Algorithm 1 in order to obtain the procedure to use in the context of mVAR GMs. Firstly, the design matrix $\boldsymbol{X}$ has to be a function also of L, i.e. its structure is the same as in Algorithm 1, but there are as many design matrices as lag $l \in L$. Secondly, the OR or AND rules have not to be applied, since the effect of X_r^{t-1} on X_s^t is conceptually different from the effect of X_s^{t-1} on X_r^t. Algorithm 2 gives a pseudo-code for the estimation of an mVAR model via neighborhood regression.

Pseudo-Algorithm 2. (mixed VAR models estimation via neighborhood regression).

1. *for each node $s \in V$:*
   (a) *build the design matrix according to the set of lags L;*
   (b) *solve the lasso problem in Equation (1.9) with penalization hyper-parameter λ obtaining $\hat{\boldsymbol{\theta}}$.*
2. *Define the DAG D_j according to $\hat{\boldsymbol{\theta}}$, for each lag $j \in L$.*

An application of Algorithm 2 is shown in Chapter 2. There exist a variant of Algorithms 1 and 2 for time-varying variables, but, since in the application framework of Chapter 2 we assume to have stationary time series, their details are not given here [57].

1.6 Conclusion

The chapter provides several definitions and propositions about theory on PGMs, graphs and conditional independence. In Section 1.5 an estimation approach for PGMs based on neighborhood regression is described. This criterion will be empirically used in Chapter 2 in an insurance framework to estimate the dependence structure among thirty covariates and two response variables. The chapter gives details about the consolidated theory of PGMs and does not propose new theoretical contributions. It includes the analysis of a dataset via UGM; for further details see also Ugolini et al. [117]. This study presents problems related to structural zeros, as shown in Table 1.7, leading to approximated results according to the theory of PGMs. This is because the model estimation criteria for PGMs have the assumption that the probability associated to any possible event in the sample space is strictly greater then zero. This problem can be overcome with Staged Trees, since they permit to encode detailed information about conditional probability distributions, as it will be shown in Chapter 3. In Chapter 5 the analysis on this pediatric dentistry dataset is carried on with Staged Trees.

Chapter 2

An Insurance Application: Swiss Re Project

2.1 Introduction

This chapter uses Carli et al. [22] as a guideline. The context of this work is marine insurance and the objective is to forecast the trend of marine losses at a global scale in upcoming years in order to constantly update an insurance company costing model. The proposed procedure starts from a potentially large number of indicators collected at different time frequencies, selects the most relevant ones through Graphical Models and uses regressive models to forecast loss trends. The use of GMs makes the variable selection more understandable and interpretable even for not statisticians. Indeed, they supply a compact representation of the dependence structure among problem factors and do not only allow the encoding of probability distributions but also provide a very clear interface to interpret the model and to perform predictions. Furthermore, GMs estimated from a dataset can be useful to confirm known independence relationships, to validate the dataset and mainly to identify unexpected relationships. Robustness of estimates and data sensitivity, which are crucial in an insurance context, are dealt with bootstrap. Goodness of fit and of estimates is expressed via the percentage error and the mean square error. Functions available in a number of *R* packages are used and the whole process is collected in an ad-hoc Shiny App for making the analysis replicable and the proposed procedure usable also by business experts.

2.1.1 Motivation

The objective of this project is to forecast the trend of marine losses in upcoming years, that is the loss or damage of ships. Marine losses are divided into two categories, *total* and *partial*. A total loss occurs when damage to a vessel is beyond repair or salvage, while a partial loss occurs when a damage can be repaired.

As a matter of fact, the global trend of marine losses is not constant, so that insurance companies are interested in evaluating not only which are the factors that may have an influence on marine losses, but also in predicting the future trend in order to adjust baseline cost produced by the in-house costing model. For example, if the trend is predicted to be increasing, an increase in the baseline cost of the insurance policy would be recommended, and vice versa. The aim of this work is to derive a model based on some covariates which can anticipate the trend of marine losses at a global scale; that is, the focus is not on the single vessel rather consider all the vessels together, not considering single characteristics of a specific ship but worldwide indicators. Furthermore, the derived model has to be constantly evaluated and updated as soon as new data arrives.

Two different models have been implemented for the two response variables, total and partial losses. These two models are derived with the same procedure outlined in this chapter but they are different in the involved covariates. In the case study of Section 2.5, as proposed by Swiss Re Group, these models are used for adjusting baseline cost of the in-house costing model, but the output of the procedure suggested here can be employed in the costing model in other ways, e.g. for improving its predictability.

2.1.2 Marine Context

Marine traffic is a very complex system that includes ships, ports, routes, equipment, people, cargo material and environmental issues, but also political regulations. The volume of ship transportation has increased rapidly in the last few years due to the growth of the economic development and trade among different countries with 6480 millions of tons in 2003 to 11005 millions of tons in 2018. Furthermore, despite nowadays the safety of ships and equipment have reached a very high technological level, the number of amount of claims is not significantly decreased as expected. Careful analysis of the causes of accidents carried out through accident causation theory/waterborne transport research shows that most accidents are not caused by a single event, but by a series of interacting factors.

It is reasonable to study the relationship between marine losses and sources

of risks, some of which are known in the business and some have been identified with the procedure of this work. One can consider four different types of factors: those that are certainly influential whose effect only needs to be measured, those for which one wants to evaluate if their measured effect is statistically significant, not measurable factors that could possibly be relevant or influential (e.g. crew composition) and hidden/undetected factors. Here only factors of the first two types are included in order to obtain an easily interpretable model. This choice has been made because we were interested in working with known and/or measurable variables, as the application on which we tested the procedure in the following of the chapter required it.

The number of covariates that can influence marine losses at a global scale is huge, for example various indicators related to economical, social, political, technological and environmental global phenomena are used. Both marine losses and factors/indicators taken into considerations in this work are time series. A first difficulty is then to reduce this huge number of factors to the most influential ones. Another difficulty is to understand the causal temporal structure of these indicators and take into account the temporal shift among marine losses and some of them. The derivation of these predictive models is empirical in part, in the sense that when new time points become available, the model can be automatically updated, carrying out some checks guided by insurance domain experts if needed.

The indicators are available from different sources, both public and private databases. They may be collected at different time frequencies and they may be incomplete, particularly on the most recent time points. In Section 2.2 different methods for dealing with indicators at different time frequencies and for imputing missing values in time series are considered.

It is highly plausible that, considering a large number of indicators, multicollinearity problems may arise. To deal with this, in Section 2.3 mixed Vector Autoregressive (mVAR) Graphical Models and their usage in this framework in order to select the most influential factors are briefly presented. The methodologies outlined in Sections 2.2 and 2.3 not only allow to deal with multi-frequency and multi-collinearity, but also provide a visualization of the relationships among factors which can advise domain experts. These are a sort of preprocessing of the data.

Once these preliminary results have been validated with the help of business experts, multivariate autoregressive models to forecast marine losses trends have been implemented in Section 2.4.1. They use information from the mVAR Graphical Models such as lags at which each selected indicator has an influence. Section 2.4.2

is dedicated to the robustness of the models estimates to deal with data sensitivity.

Finally, in Section 2.5 the case study that motivates the procedure proposed in this work is presented.

2.1.3 Process Flow of the Procedure

Figure 2.1 illustrates the process flow of the whole procedure. This follows the main steps that allowed us to address the difficulties emerging during this work. Starting from a set of indicators from public and private databases gathered with the help of business experts, firstly, it is necessary to ensure that all the indicators collected, whose trend could influence marine losses, have the same sampling frequency (quarterly in the case study of Section 2.5). This is because the interest is in performing a multivariate analysis for the reasons mentioned in Section 2.1.2. Secondly, this reduces the number of indicators to the most influential ones. Thirdly and finally, we build the model of interest.

As soon as new data arrives, if the sampling frequency is already the one desired, the output of the procedure is simply updated; otherwise, if the new observation is on a lower sampling frequency, it is necessary to proceed with temporal disaggregation, carrying out also some checks guided by insurance domain experts. A review of the entire procedure is instead required if major events that may change dependence relationships occur.

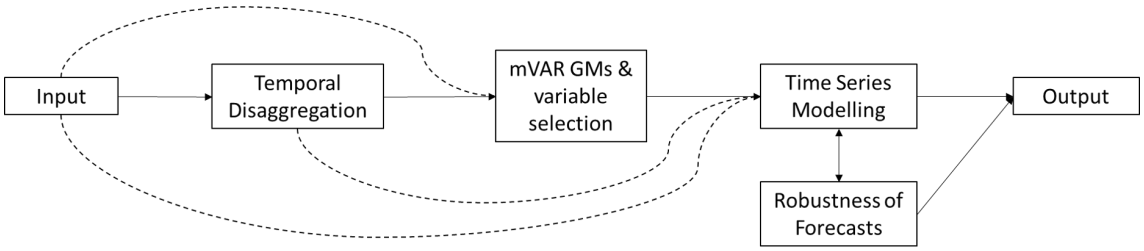


Figure 2.1: Process flow.

2.2 Temporal Disaggregation and Interpolation

Important economic indicators are taken from official sources such as official national and EU statistics. Often they are observed only on a yearly time window (low sampling time frequency) and are calculated after the end of the year. But if one is interested in constantly evaluating the decisions in order to identify preventively a

potential new trend and, eventually, update the business strategy, it is reasonable to adjust economic indicators to the sampling time frequency of the marine loss variables, i.e. quarterly. From now on, the term *time frequency* is used instead of *sampling time frequency* for simplicity.

The problem of reliably disaggregating low frequency to high frequency time series is known as *temporal disaggregation*. Temporal disaggregation methods are widely used in official statistics and their main objective is to construct a new high frequency series which is somehow consistent with the low frequency ones. Consistency could be of different types and depends on the nature of the data: for example, in France, Italy and other European countries, quarterly values of Gross Domestic Product (GDP) are computed using disaggregation such that, for each year, the sum of quarterly values of GDP must be equal to the annual value. For the case study in this work, yearly time series are disaggregated into quarterly time series.

Furthermore, estimating a multivariate autoregressive model requires all variables to have the same frequency. Since there is no way to fully make up for the missing data, the accuracy of the resulting high frequency series may be low, but, despite this, having one bad high frequency series could still be preferable to the switch to a lower frequency.

There are two general approaches to temporal disaggregation [124, 128]: (a) fitting a smooth and continuous curve through the lower frequency benchmark points [23], (b) using proxy indicators [101], which are high frequency indirect measures that approximate a phenomenon measured by the low frequency indicator to be disaggregated. Approach (a) includes smoothing methods based on e.g. cubic splines [7] and the Boot, Feibes and Lisman (BFL) method [11]. Important methods for approach (b) are introduced in Denton [34], Dagum and Cholette [30], Chow and Lin [24], Fernandez [43] and Litterman [84].

The Denton and Denton-Cholette methods use a single proxy indicator; as a special case a constant (e.g. a series consisting of only ones in each quarter) can be embodied allowing for temporal disaggregation without high frequency indicator series. Regression methods on the low frequency series based on several proxy indicators are described in Chow and Lin [24], Fernandez [43] and Litterman [84].

A problem with the smoothing methods in approach (a) is that while the values of the low frequency series are often interpolated or well estimated by the disaggregated series, the other estimated values may not even be in a reasonable range. For instance, if there are change points within a year then this approach will not work. In the same way, the use of an additional indicator may not always be the best

solution because two time series that are strongly correlated at a lower frequency, may not be strongly correlated at a higher frequency, so that also the choice of indicators may be critical. In order to guarantee more reasonable estimates, the validation by business experts is essential.

Some of the above mentioned methods are employed, in particular using the *R* packages **tempdisagg** and **imputeTS**, and then, following standard practice with forecasting in machine learning, a linear combination of the estimates of the high frequency time series is computed with pre-specified weights: this could lead to more robust estimates than each method taken separately [59]. The weights can be assumed equal in case of non prior knowledge on their value, or can be elicited by experts or through some preliminary analysis.

The details of this procedure for the case study are presented in Section 2.5.

2.3 Variable Selection Through Mixed Vector Autoregressive Graphical Models

In the past twenty years there has been an explosion in the use of Graphical Models to represent relationships in a random vector as conditional independence statements among its components [77]. GMs are helpful to perform inference which takes advantage of the underlying graphical representations and can also be used as a tool for (a sort of) variable selection. GMs provide a useful visualization of dependence relationships through their graphical representation and this greatly benefits the identification of which variables are relevant (i.e. dependent) for one or more response variables of your choice.

Here, for the variable selection an approach based on mixed Vector Autoregressive (mVAR) Graphical Models is adopted [57]. These are an extension of GMs for time-varying random vectors in which the underlying model changes over time under the assumption that change is a smooth function of time and they can be implemented in *R* through the package **mgm**. Furthermore, a univariate exponential function can be associated to each component of the random vector. mVAR GMs are able not only to define the set of influential factors for the response variable, but also to identify the time lag ℓ on which each selected time series is influential (see below).

Model estimation in mVAR GMs is based on a neighborhood regression approach and it can be explained through Example 1.

Example 1. *Consider a random vector* $\boldsymbol{X}$ *with four stationary and continuous*

yearly time series: $\boldsymbol{X} = (X_1, X_2, X_3, X_4)$*. Here let assume that the four time series are already on the same time frequency and in the appropriate unit measure (important point for the application in Section 2.5). In Figure 2.2 a path of each of the four components of* $\boldsymbol{X}$ *is shown.*

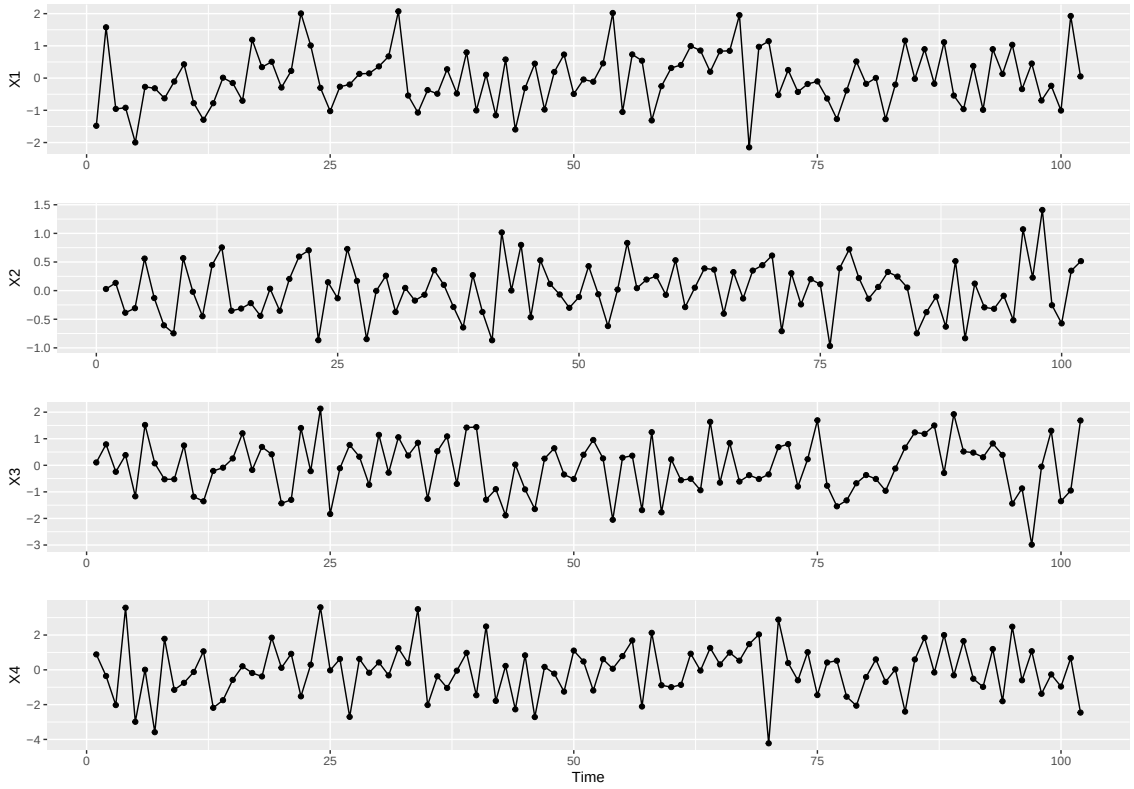


Figure 2.2: Plot of the four components of $\mathbf{X}$.

For each time lag ℓ *of interest (here* $\ell = 1, 2$ *and in the case study in Section 2.5* $\ell = 1, \ldots, 12$*), a GM is estimated by regressing each component of* $\boldsymbol{X}$ *on all of* $\boldsymbol{X}$ *according to Equation (2.1), where* t *stands for time and random errors are centered in zero,* $\mathbb{E}(\varepsilon_{i\ell}) = 0$*, for* $i = 1, \ldots, 4$*.*

$$\begin{aligned}
X_{1t} &= \beta_{01\ell} + \beta_{11\ell} X_{1t-\ell} + \beta_{21\ell} X_{2t-\ell} + \beta_{31\ell} X_{3t-\ell} + \beta_{41\ell} X_{4t-l} + \epsilon_{1\ell} \\
X_{2t} &= \beta_{02\ell} + \beta_{12\ell} X_{1t-\ell} + \beta_{22\ell} X_{2t-\ell} + \beta_{32\ell} X_{3t-\ell} + \beta_{42\ell} X_{4t-\ell} + \epsilon_{2\ell} \\
X_{3t} &= \beta_{03\ell} + \beta_{13\ell} X_{1t-\ell} + \beta_{23\ell} X_{2t-\ell} + \beta_{33\ell} X_{3t-\ell} + \beta_{43\ell} X_{4t-\ell} + \epsilon_{3\ell} \\
X_{4t} &= \beta_{04\ell} + \beta_{14\ell} X_{1t-\ell} + \beta_{24\ell} X_{2t-\ell} + \beta_{34\ell} X_{3t-\ell} + \beta_{44\ell} X_{4t-\ell} + \epsilon_{4\ell}
\end{aligned} \tag{2.1}$$

The estimated parameters $\hat{\beta}_\ell$ can be efficiently stored in ℓ square matrices with dimension the length of $\mathbf{X}$ plus one, say $p + 1$. The strength of the effect of each covariate on the components of $\mathbf{X}$ can be displayed in a GM considered for each

lag, obtaining an intuitive and helpful representation. The color of edges in the graphs denotes the sign of the corresponding $\hat{\beta}_\ell$ between the two time series and the thickness of edges is proportional to the strength of that relationship. The selection of the dependence structure of $\mathbf{X}$ and of the important lags is based on these matrices. The presence of an edge and its thickness are the information needed to perform a variable selection.

Notice that one time series lagged at a specific lag can have an effect on another time series and the latter can have an effect at the same time lag on that time series. This is represented in a GM with two nodes with two edges pointing each other (this does not occur in Example 1 but occurs in the case study of Section 2.5). Mathematically, this occurs when for a generic pair of time series X_i and X_j, the corresponding $\hat{\beta}_{ij\ell} > 0$ and also $\hat{\beta}_{ji\ell} > 0$, for a specific lag ℓ.

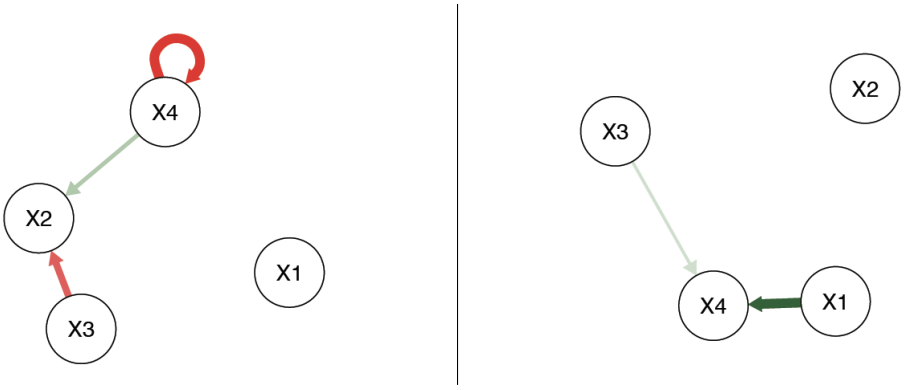


Figure 2.3: Possible GMs for Equation (2.1) showing covariates significance for time series lagged at one year (left) or two years (right).

Example 1. ***(continued).*** *A possible outcome with $\ell = 1, 2$ is given in Figure 2.3. According to the GM in Figure 2.3 (left) X_3 lagged at one year has a negative effect on X_2, while X_4 has a positive effect on X_2. The self-loop on X_4 denotes an autoregressive time series. For time series lagged at two years (Figure 2.3 right) X_1 and X_3 have a positive influence on anticipate X_4. In this example, assuming that X_4 is the variable of interest, X_2 can be excluded from further analysis since it has no edge pointing in or out of it for both considered lags, while X_1 and X_3 have to be retained because at their effect at lag 2.*

The variable selection procedure outlined above is a thoroughly multivariate method, computationally efficient, and, as expected, has produced more accurate results than those obtained from a simple univariate analysis as shown below.

Example 1. ***(continued).*** *The forecasts for the last two yearly values of the response variable X_4 provided by an univariate autoregressive model ($\hat{X}_{4t} = \hat{\beta}_1 X_{4t-1}$)*

and by the multivariate autoregressive model estimated with the procedure outlined in this section (relations displayed in Figure 2.3), that is reported in Equation (2.2), are compared.

$$\hat{X}_{4t} = \hat{\beta}_{441} X_{4t-1} + \hat{\beta}_{142} X_{1t-2} + \hat{\beta}_{342} X_{3t-2} \tag{2.2}$$

In Figure 2.4 these forecasted values are displayed: red is related to univariate approach and green to the multivariate one. The estimates obtained with the multivariate method are more accurate than the ones with the univariate method.

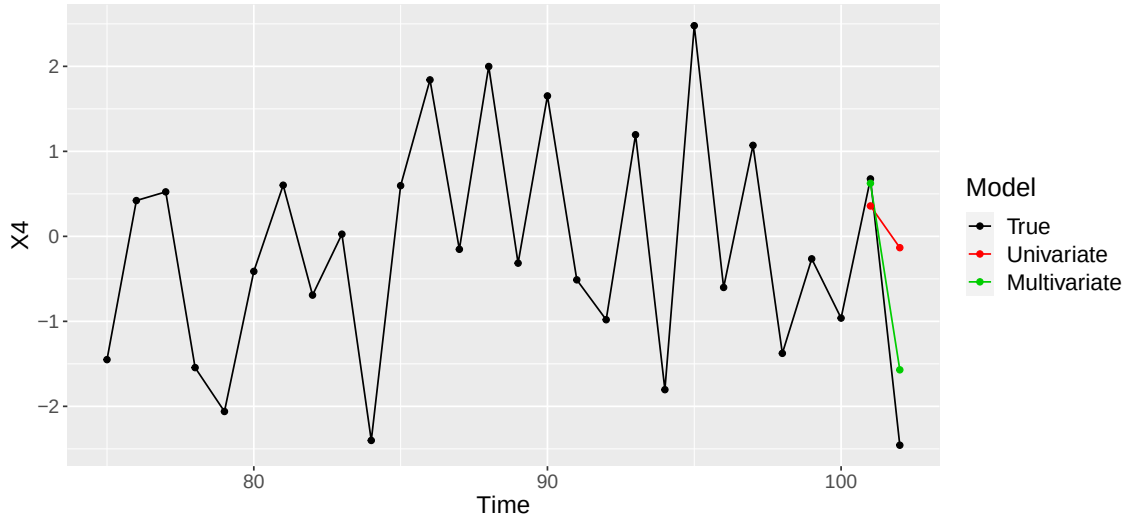


Figure 2.4: Plot of predicted values obtained by univariate (*red*) and multivariate (*green*) analysis.

In the case study of Section 2.5 two response variables are considered, but the variable selection is performed reasoning one variable at a time, in analogy to X_4 in Example 1. This is because for each response variable one has to select the corresponding influential covariates, which can be different with respect to the influential factors for the other response variable.

2.3.1 Parameters Estimation

The log-likelihood associated to the estimated neighborhood regression model can be calculated and it can be penalized with respect to the number of estimated parameters [57]. For variable selection purpose, the *Lasso Penalization* is used [59]. It has the property to shrink the less important variable's coefficients to zero by setting, a priori, a penalization hyper-parameter which often is unknown or hard to estimate. Equation (2.3) shows how the parameters vector $\boldsymbol{\beta}_\ell$ is estimated in this work, where λ is the penalization hyper-parameter, $l(\boldsymbol{\beta}_\ell, \mathbf{X})$ the log-likelihood of the model and $||\boldsymbol{\beta}_\ell||_1$ the ℓ_1 norm of the parameters vector $\boldsymbol{\beta}_\ell$.

$$\hat{\boldsymbol{\beta}}_\ell = \arg\min_{\boldsymbol{\beta}_\ell} \left\{ l(\boldsymbol{\beta}_\ell, \mathbf{X}) + \lambda \, ||\boldsymbol{\beta}_\ell||_1 \right\} \tag{2.3}$$

For a robust estimation of $\boldsymbol{\beta}_\ell$ many different hyper-parameters (one hundred hyper-parameters values suffice for the case study in Section 2.5) are used and then the average of all the estimates of the $\boldsymbol{\beta}_\ell$ is computed. With this procedure one can be more confident that the most significant relationships between time series have been selected, with respect to the estimation of a neighborhood regression model made only once. An approach based on the choice of a specific value of λ that leads to a a particular level of sparsity in the graph could be adopted. Also a study using a cross validation method to select the best lambda could be carried out. The choice made in this thesis is motivated by the fact that, at first, we wanted to distinguish the variables that were most likely not influencing the response variable. Then, through an analysis of these first results together with the business experts, the threshold below which to consider an effect as not significant has been selected. We are aware that this approach is not usual in a machine learning context, but it was considered the most appropriate together with the business experts, in order to have a conservative intermediate result and to be able to discuss the continuation of the work together.

To achieve a more conservative dependence structure for $\mathbf{X}$, relationships among variables are considered significant if the corresponding estimated $\hat{\boldsymbol{\beta}}_\ell$ is not only different from zero, but also greater, in absolute value, of a fixed threshold, say π_0:

$$|\hat{\boldsymbol{\beta}}_\ell| > \pi_0.$$

Example 1. ***(continued).*** *According to the thickness of edges in Figure 2.3, it can concluded that also X_3 is not relevant for X_4, due to the thin edge between them. This leads to a further reduction of the set of selected variables.*

Note that normalize or standardize the explanatory variables, if they are not on the same scale, is advisable for applying this shrinking method on the parameters.

2.4 Time Series Modelling

2.4.1 Data Modelling

After the temporal disaggregation and the selection of time series of interest for the prediction of one or more response variables, the next step is the modelling of the data in order to produce a forecast for future time points. Thus, from the

original variables in $\mathbf{X}$, a response Y is chosen (below Y is unidimensional for sake of exposition), and a subset, say $\tilde{\boldsymbol{X}}$, of p variables is selected.

In this work four different methods chosen for their predictive performances, their simplicity of use and/or their recognized usefulness in the application context of this work and their forecast values are compared. These models are the following:

- Vectorized Autoregressive Moving Average with Exogenous Variables (VARMAX) (for details see Lütkepohl [86]);
- Generalized Linear Autoregressive Moving Average Model (GLARMA) (e.g. Dunsmuir et al. [39]);
- Generalized Linear Regression Model (GLM) (e.g. Hastie et al. [59]);
- Linear Regression Model (LM) (e.g. Hastie et al. [59]).

The VARMAX(k, q) model has the formulation shown in Equation (2.4), where Y_t is the response variable at time t, $\alpha_0, \ldots, \alpha_k$ the $k+1$ coefficients related to the autoregressive part of the process, $\boldsymbol{X}_t, \ldots, \boldsymbol{X}_{t-k}$ the $k+1$ covariate variables $\tilde{\boldsymbol{X}}$ at lagged values from 0 to k, $\boldsymbol{\beta}_0, \ldots, \boldsymbol{\beta}_k$ the $k+1$ coefficients vectors of the regressive component of the model, $\varepsilon_t, \ldots, \varepsilon_{t-q}$ the $q+1$ white noise error term and $\theta_1, \ldots, \theta_q$ the q coefficients vectors related to the moving average part of the process.

$$\begin{aligned} Y_t = {} & \alpha_0 + \alpha_1 Y_{t-1} + \cdots + \alpha_k Y_{t-k} + \boldsymbol{X}_t\boldsymbol{\beta}_0 + \boldsymbol{X}_{t-1}\boldsymbol{\beta}_1 + \cdots + \boldsymbol{X}_{t-k}\boldsymbol{\beta}_k \\ & + \ \varepsilon_t + \ \theta_1\varepsilon_{t-1} + \cdots + \theta_q\varepsilon_{t-q} \end{aligned} \tag{2.4}$$

The GLARMA model in Equation (2.5) is expressed as the expected value of the response variable at time t, where g is the link function (e.g. log), $Z_t = \sum_{i=1}^{\infty} \phi_i e_{t-i}$, e_{t-i} , $i = 1, \ldots, \infty$, the residual term and ϕ_i , $i = 1, \ldots, \infty$, the coefficients related to the residuals.

$$\mathbb{E}(Y_t) = \mu_t = g^{-1}(\boldsymbol{X}_t\boldsymbol{\beta}_0 + \boldsymbol{X}_{t-1}\boldsymbol{\beta}_1 + \cdots + \boldsymbol{X}_{t-k}\boldsymbol{\beta}_k + Z_t) \tag{2.5}$$

If the terms Z_t are not present in Equation (2.5), the GLARMA model is the standard GLM in Equation (2.6).

$$\mathbb{E}(Y_t) = \mu_t = g^{-1}(\boldsymbol{X}_t\boldsymbol{\beta}_0 + \boldsymbol{X}_{t-1}\boldsymbol{\beta}_1 + \cdots + \boldsymbol{X}_{t-k}\boldsymbol{\beta}_k) \tag{2.6}$$

Furthermore, if the link function g in Equation (2.6) is the identity function, then it is the standard LM. In the case study in Section 2.5 the link function g

is the logarithm and the expected value is with respect to a Poisson distribution. GLM and LM are estimated to study whether a model without autoregressive and moving average components provides competitive forecast's performances. They are considered also for their minor complexity with respect to VARMAX and GLARMA models and because they are already adopted by the company. The forecasting performances of these four models can be compared, for instance, using some measures of the quality of the forecasts (e.g. Mean Square Error). For the case study, also a Weighted Combination (WC) of the forecasts of the four models is performed in order to build more robust estimates compared to each forecasts obtained with the four models taken separately (see e.g. Box et al. [15]). Note that the WC is not a statistical model itself.

Example 1. ***(continued).*** *According to the variable selection performed in Example 1 of Section 2.3 and denoting the response variable X_4 with Y_t, Equation (2.4) becomes*

$$Y_t = \alpha_0 + \alpha_1 Y_{t-1} + \boldsymbol{X}_{t-2}\boldsymbol{\beta}_2 + \varepsilon_t + \ \theta_1\varepsilon_{t-1} + \theta_2\varepsilon_{t-2},$$

with $\boldsymbol{X}_{t-2}$ the matrix with relevant covariates at lag $\ell = 2$, i.e. X_1 and X_3. Hence, the $\boldsymbol{X}\boldsymbol{\beta}$ term that holds for also the other three estimated models is simply $\boldsymbol{X}_{t-2}\boldsymbol{\beta}_2$, since at lag $\ell = 1$ no variable was influential for the response one.

2.4.2 Robustness of Forecasts

In Section 2.2 some methods for temporal disaggregation are presented in order to obtain high frequency values from low frequency ones and it has been commented on the fact that the accuracy may be low since it is impossible to fully make up for the missing data. It has been proposed also to employ different methods of disaggregation and then compute their weighted linear combination. In this section, a bootstrap approach is adopted in order to deal with the data sensitivity derived from disaggregation, to achieve robustness of the model estimates in Section 2.4.1 and improve forecasting accuracy.

In the last decade, many developments have taken place in bootstrapping time series in order to apply a risk preventive approach. The bootstrap method is a powerful tool, especially when only a small data set (e.g. a short-in-time series) is available to predict the behaviour of complex systems or processes and obtain robust results. The main idea is to generate bootstrap replicates by resampling with replacement the original time series according to one of the methods outlined below and, on these bootstrap samples, calculate statistics, i.e. parameter estimates in Section 2.4.1, and obtain their confidence intervals. In this work, this is carried

out only on the response variables.

Efron's original bootstrap procedure [40] requires i.i.d. data; a way to overcome the issue of the dependence structure of time series is to use the *naive bootstrap*, that is to compute the first order differences and, if those are i.i.d., resample the differences and sum these differences to construct many bootstrap samples of the original time series.

Since the first order differences in many cases are not i.i.d., one can instead use the *block bootstrap* [19, 122], which tries to replicate the dependence structure of a time series by resampling i.i.d. blocks of data instead of individual time points. This requires us to choose a large enough block length b_ℓ according to the autocorrelation on the original time series, so that observations more than b_ℓ time units apart will be nearly independent. This method, clearly, works only if the blocks are i.i.d..

If neither i.i.d blocks can be identified or if one can make some assumptions on the modelling of the dependence structure underlying the data, then it is possible to apply the *residual bootstrap* [19]. Once one of the models in Section 2.4.1 (e.g. VARMAX) is estimated on the observed time series, the residuals which embody the remaining uncertainty can be calculated. If they are i.i.d., one can resample them and generate bootstrap samples by adding these resampled residuals to the fitted time series (according to the estimated model). For example, considering an estimate of the LM in Equation (2.6) with g the identity function as $Y_t = \hat{Y}_t + \varepsilon_t$, the residual bootstrap consists in resampling with replacement the residuals ε_t, obtaining the new sampled residuals ε'_t and the new bootstrap samples $Y'_t = \hat{Y}_t + \varepsilon'_t$. Extensions of residual bootstrap to not i.i.d. residuals can handle other dependency structures [36].

There is no universal method that works well for every case, but it must be selected according to the type of time series and their dependence structure. For the case study, the residual bootstrap approach is adopted and it is employed on the response variables, i.e. total and partial losses.

The application of one of the above bootstrap methods leads to the construction of many bootstrap samples on which, being them plausible variations of the original time series, is reasonable to estimate one of the models in Section 2.4.1. With this procedure the objective is trying to overcome the data sensitivity problem: indeed, instead of having a single forecast, as many forecasts as the number of bootstrap samples are obtained. Hence, a distribution of the predicted values and the corresponding confidence intervals are derived in order to evaluate their robustness, i.e. the wider the confidence intervals, the less robust the estimates.

2.5 Case Study

In this work 30 time dependent indicators, modelled as $X_1, \ldots, X_{30}$, are considered in addition to the two response variables, total and partial losses (Y_1 and Y_2 respectively), collected from 2006 to 2017. Among these 30 indicators, thirteen require disaggregation as their collected values are only on a yearly scale while the time series for Y_1 and Y_2 are on a quarterly scale. Let $\underline{\boldsymbol{X}} = (X_1, \ldots, X_{30})$ and $\underline{\boldsymbol{Y}} = (Y_1, Y_2)$ be respectively the random vectors of either Poisson or Gaussian distributed covariates and of two Poisson distributed responses, defining $\mathbf{X} = (\underline{\boldsymbol{X}}, \underline{\boldsymbol{Y}}) = (X_1, \ldots, X_{30}, Y_1, Y_2)$. The time series in $\underline{\boldsymbol{X}}$ are divided into five macro areas: *technological* (e.g. seaborne trade), *political* (e.g. piracy), *social* (e.g. death), *economical* (e.g. steel price, GDP) and *environmental* (e.g. CO_2 maritime, sea ice extent). These indicators are mainly collected from public sources but also from private databases; for this reason the data reported in this section are masked or altered, if needed, to overcome the problem of violation of sensitive data.

2.5.1 Temporal Disaggregation

The steps for performing temporal disaggregation in Section 2.2 are outlined on the covariate *CO_2 Maritime*, i.e. the CO_2 emissions by vessels. Different methods in Denton [34], Dagum and Cholette [30] and Baxter [7] are used, in particular the `na_interpolation` function of the *R* package **imputeTS** [90] for simple linear and cubic spline interpolation and the `td` function of **tempdisagg** [101] for Denton and Denton-Cholette methods. The results of these four methods are shown in Figure 2.5. The linear interpolation and the Denton method (red curve) give exactly the same estimates of the quarterly values, indeed it can be shown that the Denton method without a proxy indicator corresponds to the linear interpolation. Although the cubic spline interpolation (green) and the Denton-Cholette method (blue) produce very similar estimates, some differences can be observed especially in the last quarters of 2016 and 2017. As already mentioned at the end of Section 2.2, a linear combination of the estimates (pink line) is computed assuming, in this case, equal weights for each method.

2.5.2 mVAR Graphical Models and Variable Selection

After temporal disaggregation, all thirty-two time series are on a quarterly frequency scale. In Equation (2.1) lags ℓ are taken from 1 to 12, due to the company's interest in modelling losses by monitoring the dependence structure up to at most three previous years (i.e. lag 12 with quarterly time series). As the random vector $\mathbf{X}$

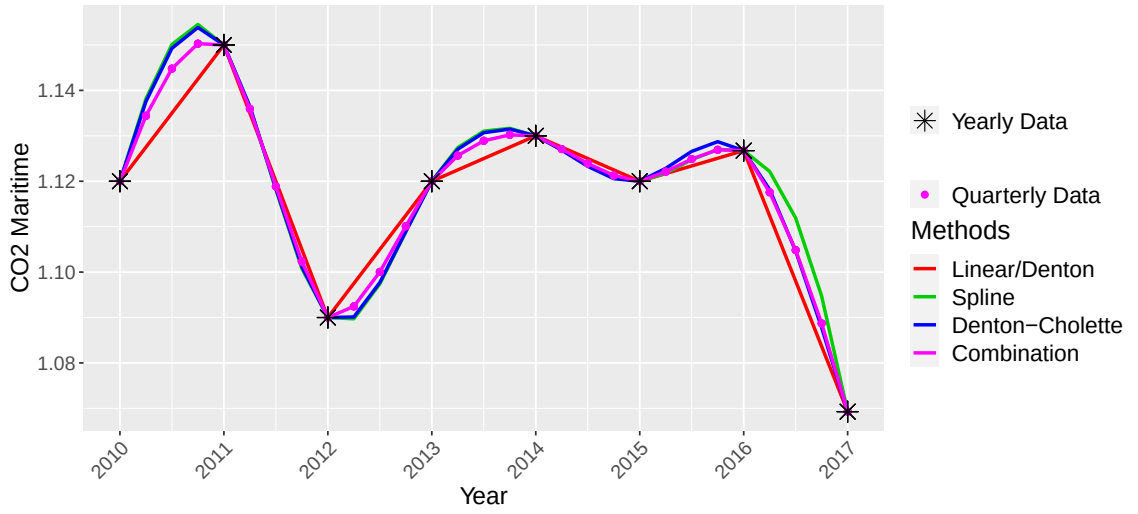


Figure 2.5: Temporal Disaggregation methods applied on the covariate CO_2 *Maritime*; time window from 2010 to 2017.

has thirty two components, the neighborhood regression scheme in Equation (2.1) of Section 2.3 is composed by thirty two regression equations for each considered lag, leading to the definition of twelve square matrices of size thirty three for the estimated parameters $\hat{\boldsymbol{\beta}}_\ell$.

Estimates of the mVAR GMs were performed also with a penalty parameter λ selected through cross validation. However, for many lags few edges have been estimated, resulting this approach not reliable and too sensitive to the hyper-parameter value. From a practical point of view, this was also not useful since we want to understand the causal relationship among the variables taken into consideration and we cannot rely on such unstable estimates. In Table 2.1 the results obtained through the optimal lambda selected with a cross validation approach are summarized. It highlights that for many lags the variable selection is too restrictive. Indeed, for 6 lags out of the 12 considered, no significant covariates were found for Total and Partial losses. In contexts like this, it is usual to use an approach based on cross validation, but the poor stability of the results shown in Table 2.1 led us to adopt a different method.

For this reason, the neighborhood regression procedure is repeated one hundred times. The range of values chosen for λ by the software's default is a linear sequence (on logarithmic scale) from the minimum value for which no parameters are estimated to be zero to the smallest maximum value for which the model would set all parameters to zero [58]. Therefore, hundred parameters estimates $\hat{\beta}_\ell^r$ are obtained, one for each chosen penalization hyper-parameter, $\lambda \in [0.0001, 0.5500]$, and from now on:

Time Lag	λ	# edges for Total Loss	# edges for Partial Loss
$\ell = 1$	0.1705	2	2
$\ell = 2$	0.2291	0	0
$\ell = 3$	0.2133	1	3
$\ell = 4$	0.2620	0	0
$\ell = 5$	0.2485	1	1
$\ell = 6$	0.2467	0	0
$\ell = 7$	0.2384	0	0
$\ell = 8$	0.2375	2	1
$\ell = 9$	0.2483	1	1
$\ell = 10$	0.2849	0	0
$\ell = 11$	0.3365	0	0
$\ell = 12$	0.2181	0	2

Table 2.1: Summary of variable selection procedure applied to Total and Partial losses with penalization parameter lambda selected through cross validation.

$$\hat{\boldsymbol{\beta}}_\ell \;=\; \frac{1}{100}\sum_{r=1}^{100} \hat{\beta}_\ell{}^r.$$

In Figure 2.6 the estimated mVAR GM for $\ell = 3$ is displayed, where the edges give a graphical representation of the matrix of estimated parameters $\hat{\boldsymbol{\beta}}_\ell$ for $\ell = 3$.

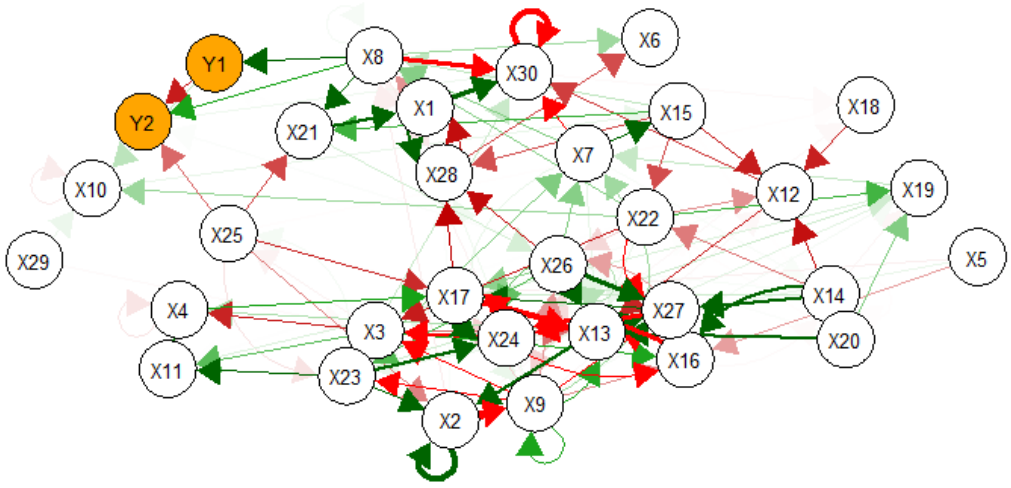


Figure 2.6: Graphical representation of estimated mVAR GM for lag $\ell = 3$.

This provides an estimate of the dependence structure among the components of $\mathbf{X}$ at lag $\ell = 3$. However, the main focus is in identifying the time series that have a direct effect on the two response variables (orange nodes in Figure 2.6). Following the steps outlined in Example 1, the graphical representation can be considerably

simplified, first by drawing only the edges pointing the nodes (one or both) related to the two response variables (Y_1, Y_2) and second by eliminating nodes that do not have an edge pointing to Y_1 or Y_2. The result of these two simplifications for lag $\ell = 3$ is shown in Figure 2.7: *Demolition* (X_8) has a large positive effect on both the responses, while *Population Growth* (X_{25}) and *Total Losses* have a negative effect on *Partial Losses*. Note for instance that in the mVAR GM in Figure 2.6 the effect of X_7 on Y_1 is mediated by X_8; indeed in the graph there is a direct path from X_7 to Y_1 through X_8. This is the power of Graphical Models: probably the exclusion of X_8 from the study would lead to the definition of a significant effect of X_7 on Y_1, but this effect is mediated from X_8 and then X_7 can be excluded for further analyses.

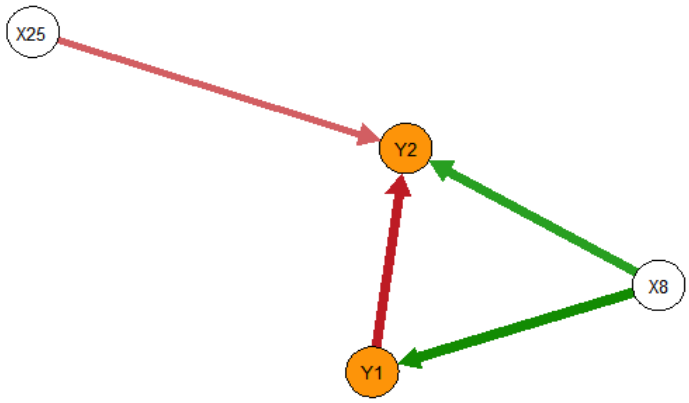


Figure 2.7: Restricted graphical representation of estimated mVAR GM for lag $\ell = 3$.

After the estimation of mVAR GM for lags ℓ from 1 to 12, one can select the covariates that have a larger significant effect at different lags on total or partial losses, separately. A summary of the variable selection made in the case study is reported in Table 2.2. Here, the color of a time series indicates the strength, in absolute value, of its effect on predicting the losses. Three ranges are considered: red is for small intensity, i.e. $|\hat{\boldsymbol{\beta}}_\ell| \in (0.001, 0.01)$, orange for medium, i.e. $|\hat{\boldsymbol{\beta}}_\ell| \in (0.01, 0.1)$ and green for large $|\hat{\boldsymbol{\beta}}_\ell| \geq 0.1$. For completeness, in Table 2.2 all the time series with a corresponding statistically significant $\hat{\boldsymbol{\beta}}_\ell \neq 0$ are reported, but, for a conservative estimate of the dependence structure as suggested in Section 2.3, an effect on the losses is considered significant only if $|\hat{\boldsymbol{\beta}}_\ell| > 0.01$, that is only the colors orange and green. Note that at this level of the process flow it is important to identify only the set of most influential covariates, without the sign of their effect on the losses. Conversely, the signs will be really crucial in the next step for forecasting losses with regressive models.

Time Lag	Total Losses	Partial Losses
$\ell = 1$	Sea Ice Extent, Demolition, Steel Price	Sea Ice Extent, Demolition, Port Detention, Globalization, Size Ship
$\ell = 2$	Laidup	Port Detention, Sea Temperature
$\ell = 3$	Demolition	Demolition, Population Growth, Total Losses
$\ell = 4$	Sea Ice Extent, Weather Incidents, Orderbook	Sea Ice Extent
$\ell = 5$	Sea Ice Extent	Sea Ice Extent
$\ell = 6$	Crude Oil, Safety Route, Sea Temperature	Laidup, Sea Temperature
$\ell = 7$	Port Detention	Demolition, Steel Price, Partial Losses
$\ell = 8$	Commodity, Partial Losses	Partial Losses, Orderbook, Population Growth, Average Age
$\ell = 9$	Sea Ice Extent	Sea Ice Extent, Population Growth, Weather Incidents
$\ell = 10$	Partial Losses, Piracy, Sea Temperature, Weather Incidents	Death, Laidup, Weather Incidents
$\ell = 11$	Demolition	Death, Demolition, Total Losses
$\ell = 12$	Average Age, Safety Route	Average Age, Total Losses, Laidup, Globalization

Table 2.2: Summary of variable selection procedure applied to total and partial losses.

At this step, the process has identified the most influential variables for total and partial losses and also the lags at which the influence occurs. This is another key information for the next step of the process flow, namely the estimation of regressive models of Section 2.4.1. From Table 2.2 it can observed that many time series are influential at more than one lag; if all the indicators are considered at all the lags at which they are influential, there would be more parameters to be estimated in the regressive models than observed time points. Then, influential indicators are included only at lag ℓ for which the strongest dependence relationship (largest intensity) is estimated, for example *Demolition* for total losses is selected as relevant at lag $\ell = 3$. Furthermore, if for an indicator the same intensity (e.g. medium) is observed at multiple lags, the indicator is included in the model only at the smallest lag; for instance, *Port detention* for partial losses is selected as relevant at lag $\ell = 1$. Finally, partial and total losses are not considered as relevant covariates even if an edge pointing the losses is estimated in the corresponding mVAR GM, since we want to have always available data for covariates and, therefore, this may not be verified with the losses, because their real values are published with a time delay of several quarters.

The result of these steps is reported in Table 2.3. For both losses it happens to select 11 covariates. Either total and partial losses at $\ell = 5, 11$ do not have influential indicators, while they share at the same lag *Sea Ice Extent* ($\ell = 1$), *Demolition* ($\ell = 3$) and *Average Age* ($\ell = 12$). The overall number of shared indicators is seven. Results in Table 2.3 were confirmed with business experts. Some variables at the estimated lag were expected in the model. For other variables the process flow allowed the identification of the lags at which they are most influential. Furthermore, for indicators for which a strong influence was not expected but were included in the regressive models by the variable selection, the business experts agreed on their relevance thanks to insights given by this pilot study.

Time Lag	$\boldsymbol{X}_{t-\ell}$	Total Losses	Partial Losses
$\ell = 1$	$\boldsymbol{X}_{t-1}$	Sea Ice Extent, Steel Price	Port Detention, Sea Ice Extent
$\ell = 2$	$\boldsymbol{X}_{t-2}$		Sea Temperature
$\ell = 3$	$\boldsymbol{X}_{t-3}$	Demolition	Demolition, Population Growth
$\ell = 4$	$\boldsymbol{X}_{t-4}$	Weather Incidents	
$\ell = 5$	$\boldsymbol{X}_{t-5}$		
$\ell = 6$	$\boldsymbol{X}_{t-6}$	Crude Oil	Laidup
$\ell = 7$	$\boldsymbol{X}_{t-7}$	Port Detention	Steel Price
$\ell = 8$	$\boldsymbol{X}_{t-8}$	Commodity	Orderbook
$\ell = 9$	$\boldsymbol{X}_{t-9}$		Weather Incidents
$\ell = 10$	$\boldsymbol{X}_{t-10}$	Piracy, Sea Temperature	Death
$\ell = 11$	$\boldsymbol{X}_{t-11}$		
$\ell = 12$	$\boldsymbol{X}_{t-12}$	Average Age, Safety Route	Average Age

Table 2.3: Final results of variable selection procedure applied to total and partial losses.

2.5.3 Time Series Modelling and Robustness of Forecasts

For the prediction of future time points for total and partial losses, the last step of the process flow in Figure 2.1 is to estimate a regressive model, based on the temporal dependence structure estimated in the previous subsection. The resulting VARMAX models are reported in Equations (2.7) and (2.8), respectively for total and partial losses.

$$\begin{aligned} Y_{1t} &= \alpha_0 + \boldsymbol{X}_{t-1}\boldsymbol{\beta}_1 + \boldsymbol{X}_{t-3}\boldsymbol{\beta}_3 + \boldsymbol{X}_{t-4}\boldsymbol{\beta}_4 + \boldsymbol{X}_{t-6}\boldsymbol{\beta}_6 + \boldsymbol{X}_{t-7}\boldsymbol{\beta}_7 + \boldsymbol{X}_{t-8}\boldsymbol{\beta}_8 \\ &+ \boldsymbol{X}_{t-10}\boldsymbol{\beta}_{10} + \boldsymbol{X}_{t-12}\boldsymbol{\beta}_{12} + \varepsilon_t + \theta_1\varepsilon_{t-1} + \ldots + \theta_q\varepsilon_{t-q} \end{aligned} \tag{2.7}$$

$$\begin{aligned} Y_{2t} &= \alpha_0 + \boldsymbol{X}_{t-1}\boldsymbol{\beta}_1 + \boldsymbol{X}_{t-2}\boldsymbol{\beta}_2 + \boldsymbol{X}_{t-3}\boldsymbol{\beta}_3 + \boldsymbol{X}_{t-6}\boldsymbol{\beta}_6 + \boldsymbol{X}_{t-7}\boldsymbol{\beta}_7 \\ &+ \boldsymbol{X}_{t-8}\boldsymbol{\beta}_8 + \boldsymbol{X}_{t-9}\boldsymbol{\beta}_9 + \boldsymbol{X}_{t-10}\boldsymbol{\beta}_{10} + \boldsymbol{X}_{t-12}\boldsymbol{\beta}_{12} + \varepsilon_t \\ &+ \theta_1\varepsilon_{t-1} + \ldots + \theta_q\varepsilon_{t-q} \end{aligned} \tag{2.8}$$

With $\boldsymbol{X}_{t-\ell}$ are denoted the covariate variables $\boldsymbol{X}$ at lagged values for $\ell = 1, \ldots, 12$ as reported in Table 2.3. For instance, for total losses at lag $\ell = 1$, only indicators *Sea Ice Extent* and *Steel Price* are considered. Clearly, the covariates that were not influential on the response variables at a certain lag (white lines in Table 2.3) were not included in the model and the corresponding $\boldsymbol{\beta}$ parameters were not estimated, e.g $\boldsymbol{\beta}_5$. Furthermore, the term $\boldsymbol{X}_t\boldsymbol{\beta}_0$ is not considered because one would always like to have available data for covariates and, therefore, observed in the past quarters and not at the current time point t.

The other three predictive models outlined in Section 2.4.1 confirm the VARMAX models and their $\boldsymbol{X\beta}$ structure follows the structure of the VARMAX equations. For the GLARMA and GLM, the link function g is set to be the logarithm since the two response variables belong to the Poisson family. Moreover, here the error term is intended as the predictive residual, as outlined for Equation (2.5).

In order to measure the goodness of fit of the regressive models, the observed sample is divided into a train set (from 2006 to 2016) for their estimation and a test set (2017) to evaluate their predictive performances.

The results below refer to the residual bootstrap procedure outlined in Section 2.4.2 for partial losses, with 500 bootstrap iterations; if more bootstrap iterations are considered the forecasts are analogous. The results for total losses are similar and they are not reported here.

In Table 2.4 the true and forecasted values of partial losses for year 2017 are shown. These forecasts are provided by the five models seen in Section 2.4.1, that are VARMAX, GLARMA, GLM, LM and their Weighted Combination (WC).

Quantity	True	VARMAX	GLARMA	GLM	LM	WC
2017 - Q1	507.00	528.00	539.00	539.00	545.00	538.00
2017 - Q2	464.00	511.00	470.00	474.00	471.00	481.00
2017 - Q3	459.00	493.00	478.00	484.00	483.00	485.00
2017 - Q4	574.00	559.00	563.00	573.00	573.00	567.00
2017	2004.00	2091.00	2050.00	2070.00	2072.00	2071.00
2017 % Error	-	4.34	2.30	3.29	3.39	3.34
2017 MSE	-	4031.00	1542.00	1815.00	2070.00	1959.00

Table 2.4: True and forecasted values of partial losses for year 2017. The forecasts are provided by the five models seen in Section 2.4.1.

The yearly estimated value of 2017 is the sum of the four quarterly predicted values and the global percentage error (% Error) is computed as the percentage of the difference between the yearly predicted value and the true one divided by the latter, e.g. for GLARMA $2.87\% = \frac{1650-1604}{1604}$. The best performing model according to the % Error and the Mean Square Error (MSE) is the GLARMA, but also the other four models provide competitive performances since the % Error for all cases is less than 5%. The first three quarters are overestimated, while the last is underestimated; hence, the overall yearly estimate results to be overestimated and this is a positive aspect as, from an insurance point of view, it is better to consider overestimates in order to be more provident. Note also that the WC in this case study does not provide the best estimate, but in another context where not all models overestimate the true values, the WC could achieve more robust estimates.

In Table 2.5 the 90% bootstrap confidence intervals are reported, which contain the corresponding true values in all cases considered here.

Quantity	True	VARMAX		GLARMA		GLM		LM		WC	
		5%	95%	5%	95%	5%	95%	5%	95%	5%	95%
2017 - Q1	507.00	446.95	607.10	474.00	611.00	473.00	612.05	476.95	614.00	471.68	607.09
2017 - Q2	464.00	427.30	696.10	402.00	550.00	405.90	551.05	387.75	557.05	405.71	559.76
2017 - Q3	459.00	387.95	686.10	396.00	572.00	401.95	573.05	384.95	580.00	398.21	573.62
2017 - Q4	574.00	418.70	997.10	426.00	730.00	436.00	737.05	426.75	712.00	440.70	711.01
2017	2004.00	1691.80	2445.45	1706.95	2442.05	1729.90	2455.15	1709.40	2446.40	1729.86	2430.55
2017 % Error	-	-15.58	22.03	-14.82	21.86	-13.68	22.51	-14.70	22.08	-13.68	21.28

Table 2.5: True values and 90% confidence intervals of forecasted values for partial losses and year 2017. Confidence interval is not calculated for MSE since it is hardly interpretable.

As an additional tool for the goodness of fit of the estimated models, Figure 2.8 depicts the true values of partial losses from 2010 to 2017 and the corresponding estimated values. For a better visualization of forecasts, in Figure 2.9 only the time window related to 2017 is displayed. The GLARMA seems to be again the most accurate, but all models catch the correct trend and match roughly the true one, except for the second quarter of 2017 that is highly overestimated by the VARMAX.

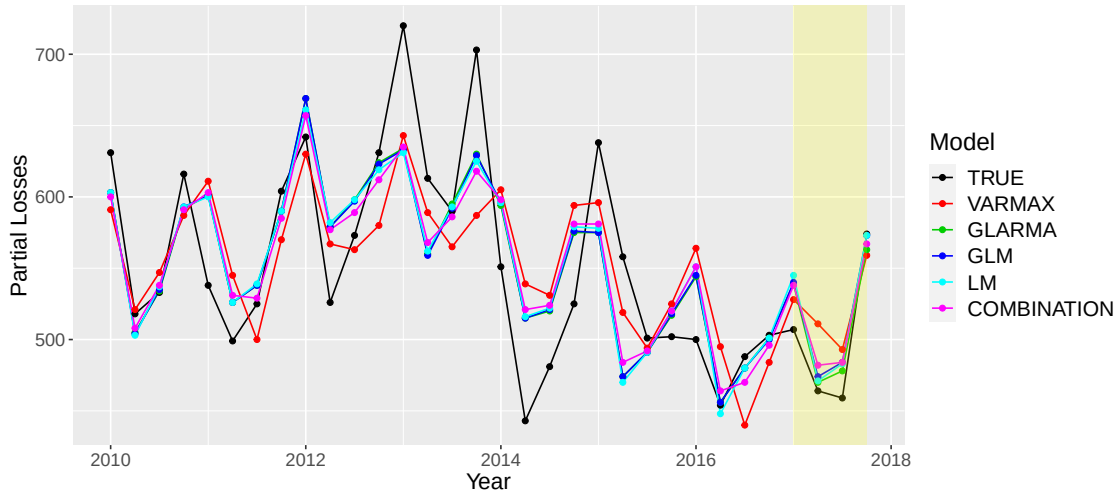


Figure 2.8: Graphical representation of partial losses from 2010, its forecast for quarters of 2017 obtained through 5 models and fitted values for time window from 2010 to 2016.

In Figure 2.10 and Figure 2.11 the histograms of the bootstrap estimates of, respectively, quarterly and yearly values for 2017 obtained with the GLARMA model are shown. The vertical dashed black lines corresponding to the true values in the observed sample and the red ones representing the bootstrap mean for the considered value are quite close in each histogram. This suggests that the estimated model is robust.

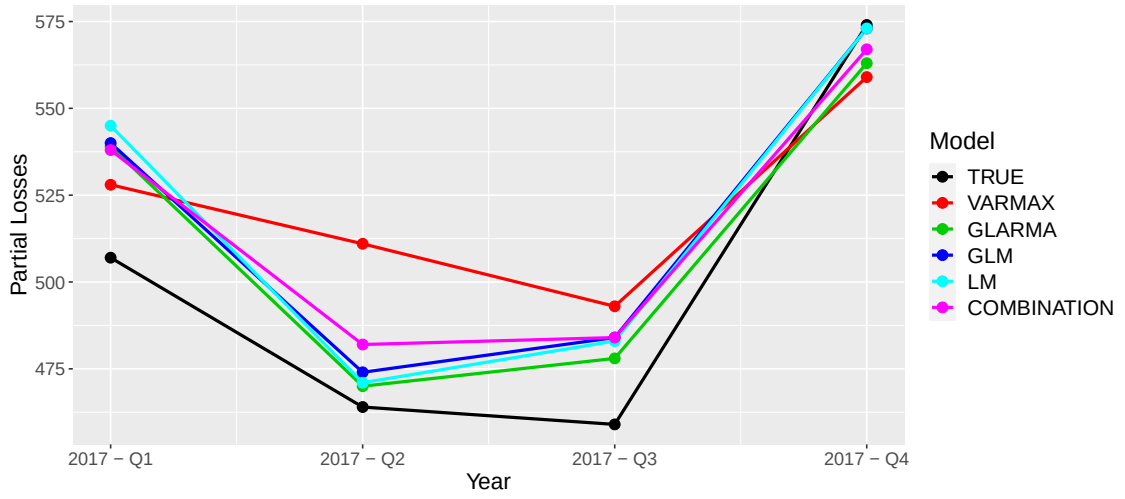


Figure 2.9: Zoom of graphical representation of partial losses and its forecast obtained through 5 models.

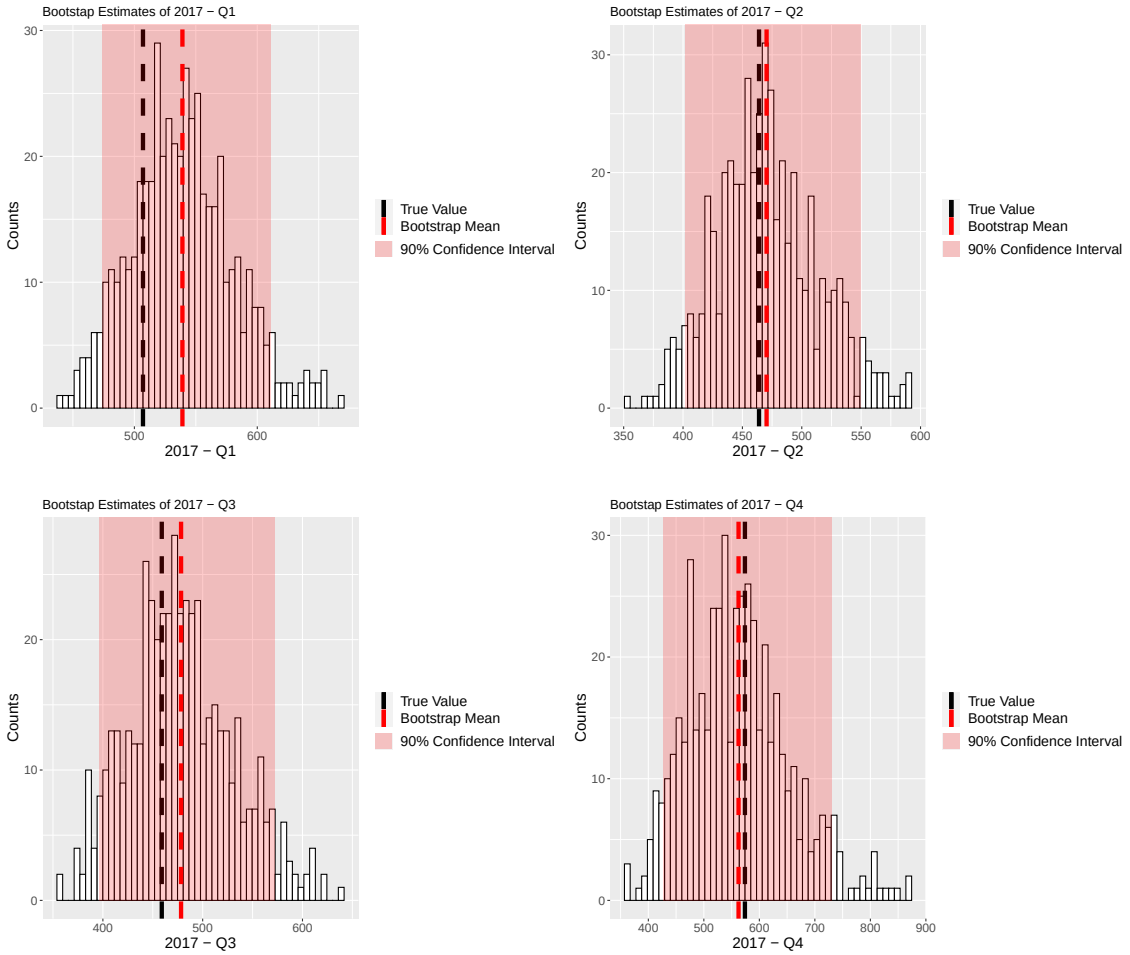


Figure 2.10: Histograms of bootstrap estimates of quarterly values obtained through GLARMA.

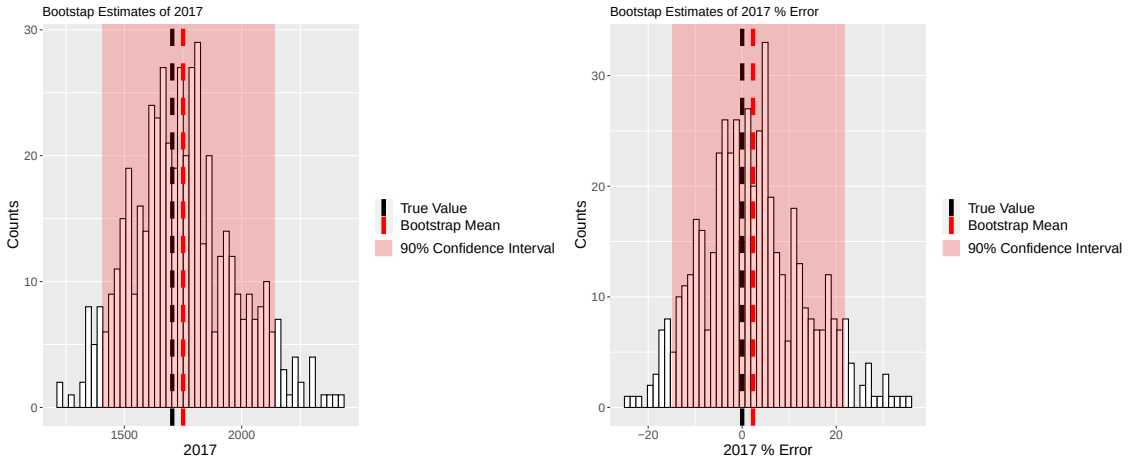


Figure 2.11: Bootstrap distributions of yearly estimates obtained through GLARMA.

The bootstrap samples (red lines) obtained by the residual bootstrap procedure of Section 2.4.2 are displayed in Figure 2.12. For the four quarters of 2017, also the whole bootstrap predictions (green lines), the mean bootstrap prediction of Table 2.4 (yellow line) and the 90% confidence intervals of Table 2.5 (blue vertical lines) obtained with the GLARMA are plotted. Further confirmation that the residual bootstrap procedure is appropriate derives from the fact that the bootstrap samples are plausible variations of the observed time series (black line).

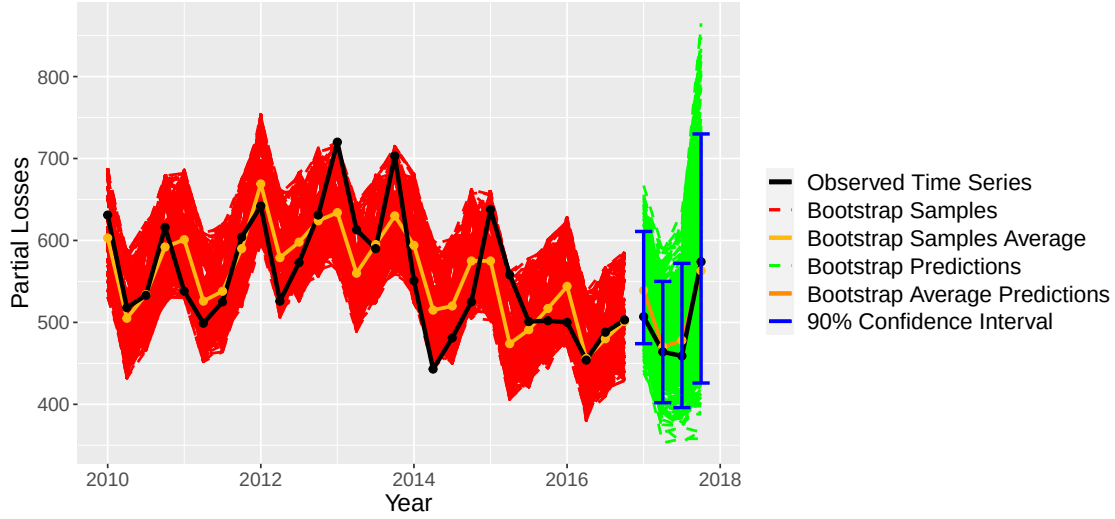


Figure 2.12: Bootstrap replicates of the GLARMA model for partial losses from 2010 to 2017.

An ad-hoc Shiny App has been designed and implemented in *R* for carrying on the steps in the process flow in Figure 2.1, except for the temporal disaggregation

step that is done a priori. The App in input receives quarterly time series and in output returns estimates of future time points of losses, their confidence intervals based on residual bootstrap and plots of both the estimated mVAR GMs and forecasts such as Figure 2.12. The App is completely interactive, in the sense that the user can select how many and which lags ℓ for the mVAR GMs to consider (e.g. Figure 2.6), if only the restricted graphical representation (e.g. Figure 2.7) should be displayed or also the tables (e.g. Table 2.4) and plots (e.g. Figure 2.12) of forecasts obtained by regressive models. Furthermore, the user can choose to visualize also bootstrap samples, bootstrap predictions with their 90% confidence intervals and the corresponding histograms (e.g. Figure 2.11).

2.6 Conclusion

In this pilot study the problem of forecasting the trend of marine losses at global scale up to three years has been considered. Since the collected time series were on different time frequencies, in Section 2.2 different methods of temporal disaggregation are outlined in order to obtain for all indicators the same high frequency of the response variables, i.e. quarterly data. In Section 2.3 the number of considered variables is reduced by selecting the most influential ones for the prediction of total and partial losses with mixed Vector Autoregressive Graphical Models.

After carrying out temporal disaggregation and variable selection, regressive models have been estimated in order to forecast future time points of marine losses. Data sensitivity and robustness of regressive model estimates were achieved and verified through a residual bootstrap procedure. In this context, the best performing model among those estimated, according to MSE and % Error, was GLARMA, providing a percentage error equal to 2.30%. To make the procedure illustrated in this chapter usable also by business experts, an ad-hoc Shiny App has been designed and implemented in *R*.

The whole procedure summarized in Figure 2.1 is a complex combination of different approaches, which are often used separately to solve smaller problems. Indeed, the state of the art of methods used in the chapter, for instance time series disaggregation, variable selection and bootstrap approach, is consolidated in the machine learning context. On the other hand, the sequentially application of these criteria may be view as an innovative idea.

Chapter 3

Stratified Staged Trees and Chain Event Graphs: Theory and Estimation

3.1 Why Staged Trees?

In this chapter recently introduced statistical models for categorical data called Staged Trees and Chain Event Graphs (CEGs) are presented [27, 107]. Also a number of algorithms for learning Stratified Staged Trees from data is introduced; they are at the core of the *R* package **stagedtrees** [120].

All discrete Bayesian Networks from Chapter 1 can be seen as Stratified Staged Trees (see Smith and Anderson [107] and Chapter 4). BNs provide a transparent graphical tool to define a complex process in terms of conditional independent local structures. This facilitates the identification of relevant structural components of the process being modelled, allows the factorization of the joint probability distribution and optimises the computational costs and time for inferences. Despite this and their obvious strengths in allowing for the reduction in the dimensionality of joint probability distributions of the model and in providing a transparent framework for causal inference, BNs may be not suitable models in certain context, for example:

- when the event space is not a product space, i.e. the event space can not be obtained as the product of the sample spaces of each considered variable. This may occur also when the nature of the hypothesis underlying the process is asymmetric; for instance, with many survey data;
- when conditional independence statements are true only under certain values of the considered random variables. This happens when there are context-

specific conditional independence structures [13, 109], which are detailed next.

The first scenario can be overcome through non-Stratified Staged Trees, which are described in Section 3.2. Some notes about the second scenario are discussed in Chapter 7. We recall the definition of context-specific independence for a random vector $\boldsymbol{X}$ from Pensar et al. [96]; this will be developed in Chapter 4.

Some extensions to the BN framework have been proposed to handle context-specific conditional independences and not product sample spaces. For instance, a context-specific BN proposed by Boutilier et al. [13] uses supplementary trees to represent the conditional probability tables that show context-specific information. Also the standard BN can be reorganized in order to depict context-specific independences using multiple vertices associated with a single variable. Another proposal is to use Bayesian Multinets or Similarity Networks [49]. These adopt a hypothesis variable to encode the context-specific statements over a set of random variables Z. For each value taken by the hypothesis variable the graphical modeller has to construct a particular BN model called local network. The collection of these local networks constitute a Bayesian Multinet or a Similarity Network.

In all the above approaches, a process is described by a set of networks instead of a single graph. The natural consequence is that the modelling procedure becomes more complicated and the computational complexity to encode these models increases substantially compared to a standard BN. These problems get worse when variables have to be intended as context-specific hypothesis associated with different states of the process. The corresponding drawbacks become more pronounced and the computational complexities increase dramatically.

Another class of models that enables to handle context-specific information is the Probabilistic Decision Graph [64]. These models were originally proposed for automated check of probabilistic expert systems and are based on ordered binary decision diagrams [18]. They allow efficient probabilistic inference especially in models with context-specific structures. Although there is a considerable overlap between Probabilistic Decision Graphs and BNs, the Probabilistic Decision Graph model class does not constitute a superclass of BNs. However, Probabilistic Decision Graphs have an underlying treegraph and, in this sense, are similar to Staged Trees and CEGs models.

Smith and Anderson [107] showed that CEG models encompass all discrete BN models and its discrete variants described above as a special subclass and they are also richer than Probabilistic Decision Graphs [112]. Unlike most of its competitors, CEGs [27, 99] can capture all conditional independences (also context-specific), compatible with the order of vertices, in a unique graph. Any CEG is obtained by a

coalescence over the vertices of an appropriately constructed probability tree, called a Staged Tree. Indeed, the Staged Tree and its associated CEG defined the same statistical model, but the CEG offers a more compact graphical representation of that statistical model. CEGs have been used for cohort studies [4], causal analysis [111, 115] and case-control studies [68, 69]. Structure learning algorithms have been defined in the literature [5, 26, 28, 105]. There are methods for inference and probability propagation [53, 116], the exploration of equivalence classes [55] and robustness studies [80, 126]. The model class of CEGs and Staged Trees have been further extended to model dynamic problems with recursively updated probabilities [6, 45], decision problems under the framework expected utility maximization [113] and Bayesian games [114].

The chapter is organized as follows: Section 3.2 introduces Staged Trees and CEGs and provides graphical examples. Sections 3.3, 3.4 and 3.5 outline the theory inherent to algorithmic implementations of estimation criteria for Stratified Staged Trees based on bayesian, frequentist and distance/divergence between probability distributions approaches, respectively. Section 3.6 gives a review of the Dynamic Programming algorithm, which guarantees a global optimum of one of the estimation criteria above over the set of all Stratified Staged Trees given an ordering of the components of a random vector $\boldsymbol{X}$. It also provides an intuitive and empirical proof of its non-applicability in real contexts even with few random variables. Section 3.7 highlights the importance of the order of variables in the construction of a Staged Tree, and gives three different algorithms to infer such variables order. Section 3.8 shows how confidence intervals can be calculated for transition probabilities of a Staged Tree; four different methods are provided. At last, Section 3.9 offers a brief conclusion of the chapter, highlighting the salient results.

All the above is implemented in the *R* package **stagedtrees** [120], whose usage will be detailed in Chapter 5.

3.2 Staged Trees

Staged Trees are directed trees equipped with probabilities where atomic events coincide with root-to-leaf paths. A CEG is constructed starting from a Staged Tree by coalescence of some vertices and it represent the same statistical model. Definition 22 refers to a general Staged Tree, while Defintion 23 to **X**-compatible Staged Trees and Defintion 24 to Stratified Staged Trees. The *R* package **stagedtrees** [120] is currently available for Stratified Staged Trees and Carli et al. [20] offers a guideline for its usage. The implementation of non-Stratified Staged Trees is still currently

under investigation, since they do not present any structure, as for instance they are not **X**-compatible; Görgen et al. [54] provides an algorithm for their estimation.

A directed tree $\mathcal{T} = (V, E)$ is a tree with vertex set V and edge set E, where each vertex except for the root has one parent only, all non-leaf vertices have at least two children and all edges point away from the root. Denote with $\boldsymbol{l}_{\mathcal{T}}$ the leaves of $\mathcal{T}$. In this framework directed trees are also called Event Trees and their vertices are also called situations. For $v, v' \in V$ let $e = (v, v') \in E$ be the edge pointing from v to v'. For a non-leaf v, let $E(v) = \{v' \in V : (v, v') \in E\}$ and call $\mathcal{F}(v) = (v, E(v))$ a floret of the tree. Let Θ be a non-empty set of labels and $\theta : E \to \Theta$ be a function such that for any non-leaf $v \in V$ the labels in $\theta(E(v))$ are all distinct. The set $\theta(E(v))$ is denoted by θ_v and is called the set of floret labels. Next assume $\Theta \subseteq [0, 1]$. If $\sum_{e \in E(v)} \theta(e) = 1$ for all non-leaf v, then $\mathcal{T}$ together with the θ_v's is called a *probability tree* and $\theta(e)$ is the probability of the edge $e \in E$. Given a vertex $v \in V$, there is a unique path in $\mathcal{T}$ from the root of the tree to v, denoted as $\lambda(v)$. Each root-to-leaf path $\lambda(v)$ in $\mathcal{T}$, for each leaf vertex $v \in \boldsymbol{l}_{\mathcal{T}}$, is associated to an atom in a discrete probability space and the corresponding atomic probabilities can be defined as $\prod_{e \in E(\lambda)} \theta(e)$, where $E(\lambda) = \{e \in E : e \in \lambda(v)\}$ denotes the set of edges in the path $\lambda(v)$. Throughout this thesis, edges on root-to-leaf paths $\lambda(\boldsymbol{l}_{\mathcal{T}})$ are ordered from the closest to the root to the closest to the leaf. The atomic probabilities together with Θ give the statistical model associated to the tree.

Definition 22. *A Probability Tree where for some $v, v' \in V$ $\theta_v = \theta_{v'}$, is called a Staged Tree. The vertices v and v' are said to be in the same stage.*

Although not strictly necessary, a probability tree can represent the joint probability distribution of a discrete random vector $\mathbf{X} = (X_1, \ldots, X_p)$ taking values in a product space $\mathbb{X} = \times_{i=1}^{p} \mathbb{X}_i$, where $\mathbb{X}_i$ is the finite sample space of X_i, $i = 1, \ldots, p$. Indeed, for $\mathbf{x} = (x_1, \ldots, x_p) \in \mathbb{X}$, the joint probability can be factorized according to the chain rule of probabilities

$$p(\mathbf{x}) = \prod_{i=2}^{p} p(x_i \mid \mathbf{x}^{i-1})\, p(x_1), \tag{3.1}$$

where $\mathbf{x}^{i-1} = (x_1, \ldots, x_{i-1}) \in \times_{j=1}^{i-1} \mathbb{X}_j$. This sequential factorization can be represented by a probability tree as the one in Figure 3.1b where the probabilities on the right-hand-side of Equation (3.1) are associated to the edges emanating from the non-leaf vertices.

Definition 23. *A Probability Tree $\mathcal{T}$ is called **X**-compatible if:*

- *it has as many leaves as elements in $\mathbb{X}$;*

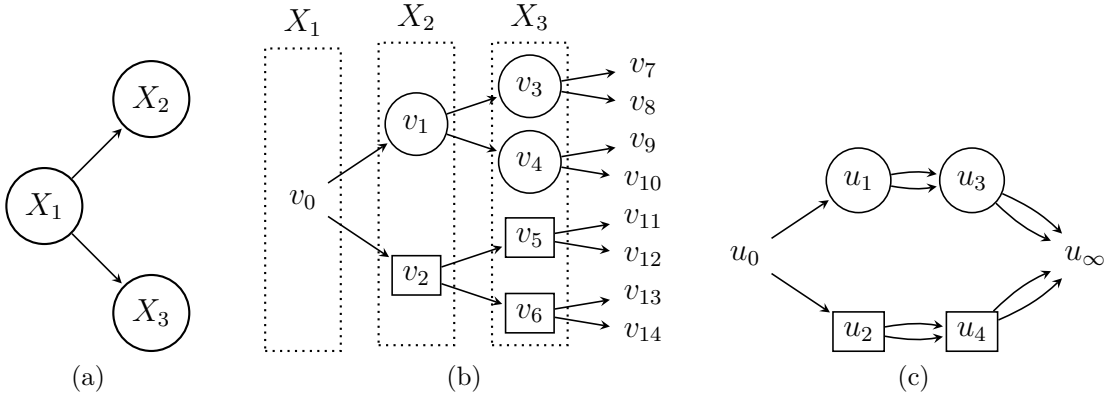


Figure 3.1: Illustration of the construction of Stratified Staged Tree and CEG starting from a BN for three binary random variables. The BN in (a) is represented like the **X**-compatible tree in (b) where the edges emanating from v_0 represent the outcomes of X_1, the edges emanating from v_1 and v_2 represent the outcomes of X_2 conditionally on the outcome of X_1 and the edges emanating from $v_3, \ldots, v_6$ represent the outcomes of X_3 conditionally on X_1 and X_2. The conditional independence encoded by the BN coincides with the staging $\{v_3, v_4\}$ and $\{v_5, v_6\}$. The Stratified Staged Tree in (b) is transformed into the CEG in (c), with positions $u_0 = \{v_0\}$, $u_1 = \{v_1\}$, $u_2 = \{v_2\}$, $u_3 = \{v_3, v_4\}$, $u_4 = \{v_5, v_6\}$ and $u_\infty = \{v_7, \ldots, v_{14}\}$.

- *edges on a root-to-leaf path λ are ordered from the closest to the root to the one closest to the leaf: $\lambda = (e_1, \ldots, e_p)$;*
- *for each $x \in \mathbb{X}$ there exists a unique root-to-leaf path $\lambda = (e_1, \ldots, e_p)$ such that $\theta(e_1) = \theta(x_1)$ and $\theta(e_i) = p(x_i \mid x^{i-1})$, for $i = 2, \ldots, p$.*

For an **X**-compatible tree, all vertices at distance i from the root are associated to the same random variable X_{i+1}, $i = 1, \ldots, p-1$, and are said to be in the same *stratum*.

Conditional independence statements embedded in BNs correspond to equalities between probabilities on the right-hand-side of Equation (3.1). This can be captured in Probability Trees by identifying some of the floret probability values.

For example, the BN in Figure 3.1a implies that X_3 is conditionally independent of X_2 given X_1, i.e. $p(x_3 \mid x_2, x_1) = p(x_3 \mid x_1)$ for all $x_i \in \mathbb{X}_i$, $i = 1, 2, 3$. The same conditional independence is embedded in the Staged Tree in Figure 3.1b by the staging $\{v_3, v_4\}$ and $\{v_5, v_6\}$ so that $\theta_{v_3} = \theta_{v_4}$ and $\theta_{v_5} = \theta_{v_6}$. By construction, all BNs have a Staged Tree representation such that situations in the same stage must be in the same stratum as in Figure 3.1. Details on the estimation of the Staged Tree corresponding to a BN are given in Section 4.

Definition 24. *An **X**-compatible Staged Tree is called stratified if all non-leaf vertices in the same stage are in the same stratum.*

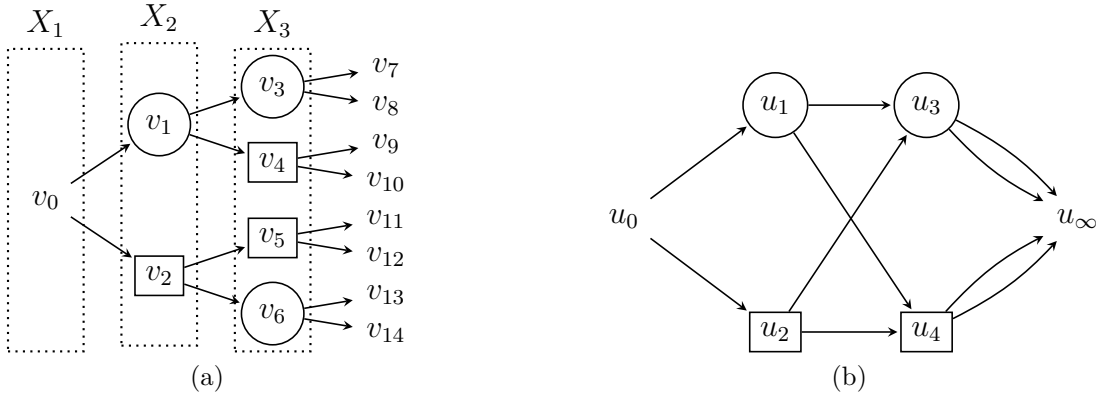


Figure 3.2: Stratified Staged Tree and CEG for three binary random variables with stages $\{v_0\}, \{v_1\}, \{v_2\}, \{v_3, v_6\}, \{v_4, v_5\}$ and positions $u_0 = \{v_0\}$, $u_1 = \{v_1\}$, $u_2 = \{v_2\}$, $u_3 = \{v_3, v_6\}$, $u_4 = \{v_4, v_5\}$ and $u_\infty = \{v_7, \ldots, v_{14}\}$.

Consider again a discrete random vector $\mathbf{X} = (X_1, \ldots, X_p)$ taking values in a product space $\mathbb{X} = \times_{i=1}^{p} \mathbb{X}_i$. Each root-to-leaf path λ in a Stratified Staged Tree corresponds to a possible realization $\mathbf{x} = (x_1, \ldots, x_p) \in \mathbb{X}$. The number of leaves in a Stratified Staged Tree is equal to the cardinality of the product space $\mathbb{X}$.

The class of Stratified Staged Trees is much larger than the one of BNs over the same variables. This is so because we demand that the BN is built on the random variables themselves and not on functions of them. Indeed, there is always a BN representation of a Staged Tree or a CEG if we allow the BN to be on functions of $\boldsymbol{X}$.

For instance, the Stratified Staged Tree with staging $\{v_3, v_6\}$ and $\{v_4, v_5\}$ in Figure 3.2a does not have a BN representation over the same X variables. The conditional independence structure embedded in the Stratified Staged Tree in Figure 3.2a is asymmetric. The stage structure for the stratum related to the variable X_3 is totally asymmetric, one can not even write a conditional independence statement in mathematical formulas. This Stratified Staged Tree states that the probability distribution of X_3 conditional on the path $v_0 \to v_1 \to v_3$ is equal to the one given the path $v_0 \to v_2 \to v_6$. The same conclusion can be drawn for the stage $\{v_4, v_5\}$, saying that the probability distribution of X_3 conditional on the path $v_0 \to v_1 \to v_4$ is equal to the one given the path $v_0 \to v_2 \to v_5$.

3.2.1 Chain Event Graphs

Staged Trees are very expressive and flexible but, as the number of variables increases, they can not succinctly visualize their staging. For this reason, Smith and Anderson [107] devised a coalescence of the tree by merging some of its vertices

in the same stage and therefore reducing the size of the graphical representation. The resulting graph is called Chain Event Graph (CEG), which represents the exact same probability model as the original Staged Tree [27]. The construction of a CEG starting from a Staged Tree is illustrated next.

Given a probability tree $\mathcal{T}$, a subtree $\mathcal{T}(v)$ rooted at $v \in V$ is the tree with v-to-leaf paths of $\mathcal{T}$ and the same edge probabilities. Two vertices $v, v' \in V$ in the same stage are said to be in the same position if the subtrees $\mathcal{T}(v)$ and $\mathcal{T}(v')$ are equal. For instance, the vertices v_3 and v_4 in Figure 3.1b are in the same stage but also in the same position. Therefore, for situations in the same position the full downstream stage structure is identical, and not only the immediate floret probabilities as for situations in the same stage. Positions give a coarser partition $\boldsymbol{u}$ of the vertex set of a Staged Tree than stages do. Hereby, all leaves are trivially in the same position denoted by $\boldsymbol{u}_\infty$.

Definition 25. *The Chain Event Graph (CEG) is the graph obtained from a Staged Tree $\mathcal{T} = (V, E)$ having a vertex for each set in U and edge set F so constructed: if there exist edges $e = (v, v')$, $e' = (w, w') \in E$ and v, w are in the same position then there exist corresponding edges $f, f' \in F$. If also v', w' are in the same position then the labels associated to f and f' are equal and are probabilities inherited from $\mathcal{T}$.*

The process of constructing a CEG is illustrated in Figure 3.1. Intuitively CEGs are very appealing, highly "viable and straightforward new tools for statistical inference" [27]. Indeed, the greater the number of levels of categorical variables involved, the more difficult it will be to have conditional independence that are valid for all the levels. A price one has to pay for the flexibility that CEGs offer is that the search space for model selection can be huge. Nevertheless, it is possible to counter this problem by restricting the search space using constraints. For example, if there are collections of Staged Trees that correspond to different people hypotheses, then model selection and comparison can be trivial. This is because usually only a subset of models would make much sense to domain experts.

3.2.2 Notations

Some simple facts and notation on Staged Trees CEGs follow, which are important for the development of the **stagedtrees** package. They are expressed for Stratified Staged Trees but analogue facts can be considered for any Staged Tree.

Vertices

Let $\mathbf{X} = (X_1, \ldots, X_p)$ be a discrete random vector taking values in a product space $\mathbb{X} = \times_{i=1}^{p} \mathbb{X}_i$. Denote with $|\mathbb{X}_i| = s_i$ the cardinality of $\mathbb{X}_i$, for $i = 1, \ldots, p$, and with $|\mathbb{X}| = \prod_{i=1}^{p} s_i$ the cardinality of the product space $\mathbb{X}$. The number of vertices/situations associated to each stratum of the tree is given in Table 3.1.

Variable	Stratum Number	Number of Situations
X_1	1 (root)	1
X_2	2	s_1
X_3	3	$s_1 \times s_2$
X_4	4	$s_1 \times s_2 \times s_3$
...	...	...
...	...	...
X_p	p	$\prod_{i=1}^{p-1} s_i$
-	$p+1$ (leaves)	$\prod_{i=1}^{p} s_i$

Table 3.1: Number of situations in each stratum of a Stratified Staged Tree.

The cardinality of the vertex set V is the sum of the number of situations for each stratum, i.e. the sum of values in the third column of Table 3.1:

$$\begin{aligned} |V| &= 1 + s_1 + (s_1 \times s_2) + (s_1 \times s_2 \times s_3) + \ \ldots \ + (s_1 \times \ldots \times s_p) = \\ &= 1 + s_1(1 + s_2(1 + s_3(\ldots(1 + s_p)))) = \bar{s} \end{aligned}$$

Stages

Let $\mathbf{q} = (\mathbf{q}_1, \ldots, \mathbf{q}_k)$ be the vector containing the stage structure, with k the number of distinct stages in the Stratified Staged Tree. Each $\boldsymbol{q}_i$, for $i = 1, \ldots, k$, is the set of vertices in the i-th stage. Since leaves are terminal vertices without a probability distribution associated, it does not make sense to define stages for that stratum of the tree. The number of possible stages associated to a Stratified Staged Tree can range between the number of stages of an independence model and a full-dependence model. The first corresponds to a tree where for each stratum all situations are in the same stage, while in the second one vertices in the same stratum are in a own stage (see Figure 3.3). Hence, the smallest number of distinct stages is p, i.e. the number of variables, and is achieved by the independence model: all vertices in a stratum are in the same stage. On the other hand, the largest number of stages is obtained through the full-dependence model and is equal to $\bar{s} - \prod_{i=1}^{p} s_i$, i.e. the cardinality of the vertex set V minus the number of leaves. In the example in Figure 3.3 three binary variables are considered, leading to the definition of 7 not-leaf vertices: the number of stages can be included between 3 (independence model) and 7 (full-

dependence model). For convenience and shortage of symbols, in a Stratified Staged Tree stages located in different strata are depicted with the same shape, as circle or rectangle in Figure 3.3b. Throughout the thesis as already mentioned, if not specified, we will always talk about Stratified Staged Trees. Then, the assumption is that only situations belonging to the same stratum, i.e. associated to the same random variable, can be enclosed in the same stage.

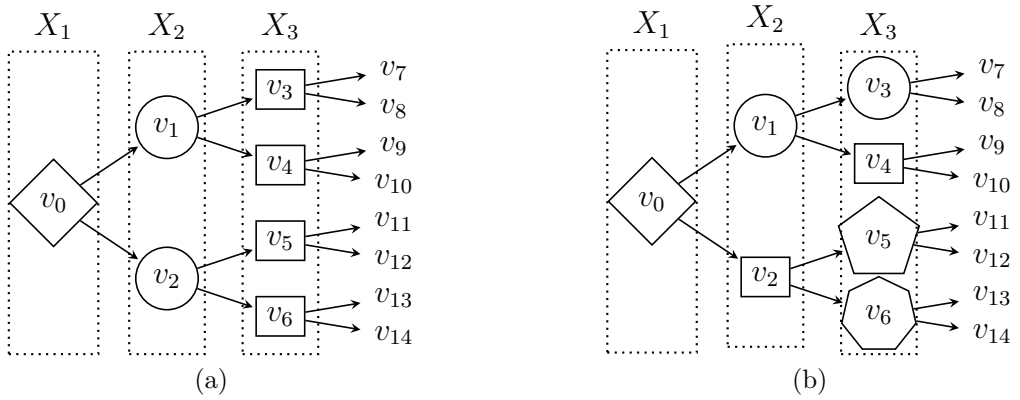


Figure 3.3: (a): Stratified Staged Tree for the independence structure on three binary variables X_1, X_2 and X_3, i.e. $X_1 \perp\!\!\!\perp X_2 \perp\!\!\!\perp X_3$. $\mathbf{q} = (\boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{q}_3)$, with $\boldsymbol{q}_1 = \{v_0\}$, $\boldsymbol{q}_2 = \{v_1, v_2\}$, $\boldsymbol{q}_3 = \{v_3, v_4, v_5, v_6\}$. (b): Stratified Staged Tree embedding a full-dependence structure among the three binary variables X_1, X_2 and X_3. $\mathbf{q} = (\boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{q}_3, \boldsymbol{q}_4, \boldsymbol{q}_5, \boldsymbol{q}_6, \boldsymbol{q}_7)$, with $\boldsymbol{q}_1 = \{v_0\}$, $\boldsymbol{q}_2 = \{v_1\}$, $\boldsymbol{q}_3 = \{v_2\}$, $\boldsymbol{q}_4 = \{v_3\}$, $\boldsymbol{q}_5 = \{v_4\}$, $\boldsymbol{q}_6 = \{v_5\}$, $\boldsymbol{q}_7 = \{v_6\}$.

Stages Probabilities

Let again $\boldsymbol{q} = (\boldsymbol{q}_1, \dots, \boldsymbol{q}_k)$ be the vector of stages and $E(\boldsymbol{q}_i) = \{e_{i_1}, \dots, e_{i_{k_i}}\}$ be the edges emanating from the stage $\boldsymbol{q}_i$, for $i = 1, \dots, k$. Then, we denote with $\mathbf{p} = (\mathbf{p}_{q_1}, \dots, \mathbf{p}_{q_k})$ the parameters vector associated to that stage structure $\mathbf{q}$ and, without loss of generality, we indicate $\mathbf{p}_{q_i}$ with $\mathbf{p}_i$. The latter can be written as $\boldsymbol{p}_i = p_{ij} = p(e_{ij})$, for $j = 1, \dots, k_i$, and corresponds to transition probabilities associated to edges emanating from the stage $\boldsymbol{q}_i$, $E(\boldsymbol{q}_i)$. Hence, the parameters vector can be succintly written as $\mathbf{p} = (p_{ij},\ i = 1, \dots, k;\ j = 1, \dots, k_i)$. Each component $\boldsymbol{p}_i$ of $\boldsymbol{p}$ is subject to the constraint $\sum_{j=1}^{k_i} p_{ij} = 1$, which guarantees that each stage parameters define a probability distribution.

Note that the number k_i of edges and the corresponding parameters associated to the stage $\boldsymbol{q}_i$ is determine with respect to the stratum in which that stage is located. More precisely, this number corresponds to the cardinality of the sample space of the variable related to the stratum in which the stage is. For example, in Figure 3.3a

situations in the stage $\boldsymbol{q}_3 = \{v_3, v_4, v_5, v_6\}$ have two outgoing edges, since they are related to X_3, which is a binary variable ($|\mathbb{X}_3| = s_3 = 2$).

Stages Sample Sizes

The same reasoning is done for the observations in the sample, using the notations as above. Let $\mathbf{y} = (\mathbf{y}_{q_1}, \ldots, \mathbf{y}_{q_k}) = (\mathbf{y}_1, \ldots, \mathbf{y}_k) = (y_{ij},\ i = 1, \ldots, k;\ j = 1, \ldots, k_i)$ be the sample sizes vector associated to the stages. y_{ij} denotes the number of observations that arrive at the stage $\boldsymbol{q}_i$, starting from the root, and pass along the edge e_{ij}. Let n be the number of observations collected in the sample: $n = \sum_{i=1}^{k_1} y_1$, indicating with the vector y_1 the sample size of the stage related to the root of the tree. Note that in a Stratified Staged Tree the sample size n corresponds to the sum of all the sample sizes of all the stages associated to a stratum of the tree. This holds for any stratum of the tree.

Prior Distribution, Likelihood Function and Posterior Distribution for Staged Trees

A Bayesian or a Frequentist approach can be adopted for the estimation of the stages structure and the probability distributions associated to each stage. In Bayesian probability theory, if the posterior distribution of the parameters of interest belongs to the same probability distribution family of its prior distribution, then the prior and posterior are called *conjugate distributions* and the prior is said to be a *conjugate prior* for the likelihood function. The choice made in this thesis for the prior distribution for the components $\boldsymbol{p}_i$ of the parameters vector $\mathbf{p}$ is the Dirichlet distribution, for which the Multinomial distribution is the conjugate: $P_i \sim Dir(\boldsymbol{\alpha}_i),\ \boldsymbol{\alpha}_i = (\alpha_{i_1}, \ldots, \alpha_{i_{k_i}})$, with $\alpha_{ij} > 0$, for $i = 1, \ldots, k$ and $j = 1, \ldots, k_i$. Other choices for the prior distribution can be done, especially according to conjugate distributions [27]. The prior Dirichlet distribution of $\mathbf{p}$ is shown in Equation (3.2).

$$\pi(\mathbf{p}) = \prod_{i=1}^{k} \pi(P_i = \mathbf{p}_i) = \prod_{i=1}^{k} \frac{\Gamma(\sum_{j=1}^{k_i} \alpha_{ij})}{\prod_{j=1}^{k_i} \Gamma(\alpha_{ij})} \prod_{j=1}^{k_i} {p_{ij}}^{\alpha_{ij}-1} \tag{3.2}$$

The likelihood function of parameters vector $\mathbf{p}$ given the observed sample $\mathbf{y}$ is factorized according to the stage structure $\boldsymbol{q}$ and by assuming for each Y_i, $i = 1, \ldots, k$, a Multinomial distribution: $Y_i \mid P_i = \mathbf{p}_i \sim Multinom(y_i = \sum_{j=1}^{k_i} y_{ij}, \mathbf{p}_i)$. Equation (3.3) exhibits the likelihood function in details.

$$L(\mathbf{p} \mid \mathbf{y}) = f(\mathbf{y} \mid \mathbf{p}) = \prod_{i=1}^{k} f_i(\mathbf{y}_i \mid \mathbf{p}_i) = \frac{n!}{\prod_{i=1}^{k} \prod_{j=1}^{k_i} y_{ij}!} \prod_{i=1}^{k} \prod_{j=1}^{k_i} p_{ij}^{\ y_{ij}} \tag{3.3}$$

The posterior distribution is shown in Equation (3.4), which, unless a rewrite of the constant C and a normalization coefficient dependent only on the sample size $\mathbf{y}$ ($L(\mathbf{y})$), is still a Dirichlet distribution. The posterior distribution of the components $\boldsymbol{p}_i$ of $\boldsymbol{p}$ is $P_i \mid Y_i = \mathbf{y}_i \ \sim \ Dir(\boldsymbol{\alpha}_{i+})$, with $\boldsymbol{\alpha}_{i+} = \boldsymbol{\alpha}_i + \mathbf{y}_i, \ \ for\, i = 1, \ldots, k$.

$$\begin{aligned} \pi(\mathbf{p} \mid \mathbf{y}) \ &= \ \frac{\pi(\mathbf{p})\ L(\mathbf{p} \mid \mathbf{y})}{L(\mathbf{y})} \ \propto \ \pi(\mathbf{p})\ L(\mathbf{p} \mid \mathbf{y}) \ = \ \prod_{i=1}^{k} \pi(\mathbf{p}_i)\ L(\mathbf{p}_i \mid \mathbf{y}_i) \\ &= \ n! \prod_{i=1}^{k} \frac{\Gamma(\sum_{j=1}^{k_i} \alpha_{ij})}{\prod_{j=1}^{k_i} \Gamma(\alpha_{ij})\ y_{ij}!} \prod_{j=1}^{k_i} p_{ij}^{\ \alpha_{ij} \ + \ y_{ij} \ - \ 1} \\ &= \ C \prod_{j=1}^{k_i} p_{ij}^{\ \alpha_{ij} \ + \ y_{ij} \ - \ 1} \end{aligned} \tag{3.4}$$

A big advantage of an analysis closed under sampling is that the marginal likelihood (Equation (3.5)) of the corresponding model can be written in closed form.

$$\begin{aligned} L(\mathbf{y}) \ &= \ f(\mathbf{y}) \ = \ \int_{\mathbf{p}} f(\mathbf{y}, \mathbf{p})\ d\mathbf{p} \ = \ \int_{\mathbf{p}} \pi(\mathbf{p})\ L(\mathbf{p} \mid \mathbf{y})\ d\mathbf{p} \\ &= \ \int_{\mathbf{p}} n! \prod_{i=1}^{k} \frac{\Gamma(\sum_{j=1}^{k_i} \alpha_{ij})}{\prod_{j=1}^{k_i} \Gamma(\alpha_{ij})\ y_{ij}!} \prod_{j=1}^{k_i} p_{ij}^{\ \alpha_{ij} \ + \ y_{ij} \ - \ 1}\ d\mathbf{p} \\ &= \ \prod_{i=1}^{k} \frac{\Gamma(\sum_{j=1}^{k_i} \ \alpha_{ij})}{\Gamma(\sum_{j=1}^{k_i} \ \alpha_{ij+})} \prod_{j=1}^{k_i} \frac{\Gamma(\alpha_{ij+})}{\Gamma(\alpha_{ij})} \end{aligned} \tag{3.5}$$

Finally, the marginal log-likelihood can be expressed as

$$\log L(\mathbf{y}) \ = \ \sum_{i=1}^{k} \left[\Big(\log \Gamma(\bar{\alpha}_i) \ - \ \log \Gamma(\bar{\alpha}_{i+}) \Big) \ - \ \Big(\sum_{j=1}^{k_i} \ \log \Gamma(\alpha_{ij}) \ - \ \log \Gamma(\alpha_{ij+}) \Big) \right],$$

with $\bar{\alpha}_i \ = \ \sum_{j=1}^{k_i} \ \alpha_{ij}$, for $i = 1, \ldots, k$.

In the framework introduced in this section, the estimation of a component p_{ij} of the parameters vector $\mathbf{p}$ can be done mainly with a Bayesian approach according to the Maxium A Posterior (MAP) criterion and with a Frequentist approach through the Maximum Likelihood Estimation (MLE). The formulations of these two criteria are reported in Equation (3.6) and (3.7), respectively.

$$\hat{p}_{ij}^{MAP} \;=\; \frac{\alpha_{ij} \;+\; y_{ij}}{\sum_{j=1}^{k_i} \alpha_{ij} + y_{ij}},\ \text{for } i = 1, \ldots, k;\ j = 1, \ldots, k_i. \tag{3.6}$$

$$\hat{p}_{ij}^{MLE} \;=\; \frac{y_{ij}}{\sum_{j=1}^{k_i} y_{ij}},\ \text{for } i = 1, \ldots, k;\ j = 1, \ldots, k_i. \tag{3.7}$$

The posterior distribution is hence maximizes through the MAP estimates $\hat{\mathbf{p}}^{\mathbf{MAP}} = (\hat{\mathbf{p}}_1^{MAP}, \ldots, \hat{\mathbf{p}}_k^{MAP}) = (\hat{p}_{ij}^{MAP},\ i = 1, \ldots, k;\ j = 1, \ldots, k_i)$, leading to the definition of the **SCORE** associated to the estimated model based on the observed sample $\mathbf{y}$:

$$\mathbf{SCORE} \;=\; \pi(\hat{\mathbf{p}}^{\mathbf{MAP}} \mid \mathbf{y})$$

On the other hand, the estimation of $\mathbf{p}$ with the MLE $\hat{\mathbf{p}}^{\mathbf{MLE}} = (\hat{\mathbf{p}}_1^{MLE}, \ldots, \hat{\mathbf{p}}_k^{MLE}) = (\hat{p}_{ij}^{MLE},\ i = 1, \ldots, k;\ j = 1, \ldots, k_i)$ allows to maximize the likelihood function in a Frequentist approach:

$$\max L(\mathbf{p} \mid \mathbf{y}) \;=\; L(\hat{\mathbf{p}}^{\mathbf{MLE}} \mid \mathbf{y})$$

3.3 Agglomerative Hierarchical Clustering Estimation

The main challenges for implementing the MAP algorithm for Stratified Staged Trees are:

- setting the prior distribution parameters $\alpha_{ij} \in \boldsymbol{\alpha}$ even for a single Stratified Staged Tree often needs a considerable amount of care. Furthermore, the number of possible models one can estimate according to the observed sample is huge. This is because the estimation of the stage structure $\boldsymbol{q}$ and the corresponding stage probabilities $p_{ij} \in \boldsymbol{p}$ require to set a priori the parameters $\boldsymbol{\alpha}$.

- The number of possible stage structures $\mathbf{q}$ of a Stratified Staged Tree increases exponentially with the number of situations/vertices in the underlying Event Tree.

- When a dataset has millions of observations, the MAP method tends to assign higher posterior probability than it should to models that contain less structure than the true data generating process. This means that the corresponding Stratified Staged Tree tends to have too many stages.

For these reasons, instead of a straightforward MAP method, Freeman and Smith [46] designed an Agglomerative Hierarchical Clustering (AHC) algorithm which is also implemented in the *R* package **ceg** [25] and was one of the few available algorithms for learning Stratified Staged Trees and CEGs from data, before the work of this thesis. Despite the algorithm comes from the literature and is not a novelty proposed with this thesis work, it is necessary to introduce it in detail for the understanding of the algorithms that will be proposed in the following. To explain the details of this algorithm, it is necessary to introduced some extra information.

Throughout this chapter suppose that the set of atomic event Ω is fixed. From a statistical point of view, a model associated to a Stratified Staged Tree is identified through the stage structure $\mathbf{q}$ and stage parameters $\mathbf{p}$ inherent to the tree. The model search space can be defined as $\mathcal{T} = \{\mathcal{M}_{\mathcal{T}} \mid \mathcal{T} \text{ is a Stratified Staged Tree in accordance with the established context}\}$. Using the strengths of Graphical Models, rather than searching over statistical models $\mathcal{M}_{\mathcal{T}} \in \mathcal{T}$, the model search is done over stage structures representing these models.

Let $\mathcal{M}_1, \mathcal{M}_2 \in \mathcal{T}$ be two models one wants to compare with $\mathcal{T}_1$ and $\mathcal{T}_2$ the corresponding Stratified Staged Trees with stage parameters vectors $\mathbf{p}_1$ and $\mathbf{p}_2$, respectively. To compare these two models, the *posterior Bayes Factor (pBF)* shown in Equation (3.8) can be adopted, with $L_i(\mathbf{y})$, $\pi_i(\mathbf{p}_i \mid \mathbf{y})$ and $\pi_i(\mathbf{p}_i)$ the marginal likelihood function, the posterior and prior distributions, respectively, for the model $\mathcal{M}_i,\ i = 1, 2$.

$$\text{pBF}(\mathcal{T}_1, \mathcal{T}_2) \;=\; \frac{\pi_1(\mathbf{p}_1 \mid \mathbf{y})}{\pi_2(\mathbf{p}_2 \mid \mathbf{y})} \;=\; \frac{\pi_1(\mathbf{p}_1)\ L_1(\mathbf{y})}{\pi_2(\mathbf{p}_2)\ L_2(\mathbf{y})} \tag{3.8}$$

Computing the logarithm on both sides of Equation (3.8) leads to the definition of the *log-posterior Bayes Factor (lpBF)* reported in Equation (3.9), where k^1 and k^2 are the number of distinct stages in $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively, and $\boldsymbol{p^1}$ and $\boldsymbol{p^2}$ the stage parameters vectors associated to $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively.

$$
\begin{aligned}
\text{lpBF}(\mathcal{T}_1, \mathcal{T}_2) &= \log \pi_1(\mathbf{p}_1 \mid \mathbf{y}) \; - \; \log \pi_2(\mathbf{p}_2 \mid \mathbf{y}) \\
&= \log \pi_1(\mathbf{p}_1) \; + \; \log L_1(\mathbf{y}) \; - \; \log \pi_2(\mathbf{p}_2) \; - \; \log L_2(\mathbf{y}) \\
&= \log \pi_1(\mathbf{p}_1) \; - \; \log \pi_2(\mathbf{p}_2) \; + \\
&\sum_{i=1}^{k^1} \left[\left(\log \Gamma(\bar{\alpha}_i^1) - \log \Gamma(\bar{\alpha}_{i+}^1) \right) - \left(\sum_{j=1}^{k_i} \log \Gamma(\alpha_{ij}^1) - \log \Gamma(\alpha_{ij+}^1) \right) \right] - \\
&\sum_{i=1}^{k^2} \left[\left(\log \Gamma(\bar{\alpha}_i^2) - \log \Gamma(\bar{\alpha}_{i+}^2) \right) - \left(\sum_{j=1}^{k_i} \log \Gamma(\alpha_{ij}^2) - \log \Gamma(\alpha_{ij+}^2) \right) \right]
\end{aligned}
\tag{3.9}
$$

Finally, the AHC algorithm can be described in detail. In a Stratified Staged Tree framework, the algorithm starts from the saturated (full-dependence model) Stratified Staged Tree $\mathcal{T}_0$ in which each situation is in a own stage, whose underlying tree is the event tree $\mathcal{T}$. Then, it merges sequentially stages at each iteration until the best stage structure according to a criterion is reached. This greedy search strategy is based on the log-posterior probability of each constructed model. In particular, assume that the AHC algorithm has chosen the Stratified Staged Tree $\mathcal{T}_i$ as the best local model at the end of the iteration i. Let $\mathbf{N}(\mathcal{T}_i) \subseteq \mathcal{T}$ be the family of possible models at the iteration $i+1$, which is composed by all Stratified Staged Tree $\mathcal{T}_{i+1}$ having the same underlying event tree as the saturated model $\mathcal{T}_0$ but with one stage less than $\mathcal{T}_i$. Hence, if a couple of distinct stages $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ in $\mathcal{T}_i$ are merged together in $\mathcal{T}_{i+1}$ forming $\boldsymbol{q}_{1-2}$, without any other changes over the tree, then $\mathcal{T}_{i+1}$ is *1-nested* in $\mathcal{T}_i$. An example of 1-nested Stratified Staged Tree will be reported in next section (Figure 3.4).

Setting a uniform prior distribution over the model space $\mathcal{T}$, the lpBF between the current model $\mathcal{T}_i$ at the iteration i and a candidate $\mathcal{T}_{i+1} \in \mathbf{N}(\mathcal{T}_i)$ can be calculated in a closed form as shown in Equation (3.10), where α_{ij}^1 and α_{ij}^2 are the parameters associated to $\mathcal{T}_i$ and $\mathcal{T}_{i+1}$, respectively. Note that since the Stratified Staged Trees $\mathcal{T}_i$ and $\mathcal{T}_{i+1}$ are 1-nested for definition of the model seach of the AHC algorithm, the parameters vectors $\boldsymbol{p^1}$ and $\boldsymbol{p^2}$ are different in only one component. More precisely, assuming that stages q_1 and q_2 in $\mathcal{T}_i$ are merged together in $\mathcal{T}_{i+1}$ forming q_{1-2}, $\boldsymbol{p^2}$ has one less element with respect to $\boldsymbol{p^1}$. In Equation (3.10), k_1^1 and k_{1-2}^2 denote the number of emanating edges from stages q_1 and q_2 in $\mathcal{T}_i$ and q_{1-2} in $\mathcal{T}_{i+1}$, respectively. In the Stratified Staged Tree framework, these values are equal since only stages located in the same stratum, i.e. associated to the same variable, can be assigned to the same new stage: $k_1^1 = k_{1-2}^2$.

$$
\begin{aligned}
\text{lpBF}(\mathcal{T}_i, \mathcal{T}_{i+1}) \; = & \log \Gamma(\bar{\alpha}^1_1) \; - \; \log \Gamma(\bar{\alpha}^1_{1+}) \; + \; \log \Gamma(\bar{\alpha}^1_2) \; - \; \log \Gamma(\bar{\alpha}^1_{2+}) \; - \\
& \sum_{j=1}^{k^1_1} \Bigg(\log \Gamma(\alpha^1_{1j}) \; - \; \log \Gamma(\alpha^1_{1j+}) \; + \; \log \Gamma(\alpha^1_{2j}) \; - \\
& \log \Gamma(\alpha^1_{2j+}) \Bigg) \; - \; \Bigg(\log \Gamma(\bar{\alpha}^2_{1-2}) \; - \; \log \Gamma(\bar{\alpha}^2_{1-2+}) \; - \\
& \sum_{j=1}^{k^2_{1-2}} \Big(\log \Gamma(\alpha^2_{1-2j}) \; - \; \log \Gamma(\alpha^2_{1-2j+}) \Big) \Bigg)
\end{aligned}
\tag{3.10}
$$

Algorithm 1 below reports the pseudo-code for the AHC algorithm.

3.4 Penalized Log-Likelihood Estimation

The maximum likelihood estimator of a Stratified Staged Tree is always the full-dependence model, that is the one which assigns each node of the tree to a own stage (e.g. Figure 3.3b). Clearly this has a big downside, because the full-dependence model in the context of this thesis corresponds to the most overfitted model one can estimate, since the probability distribution associated to each node of the tree is set to be different from others. Consequently, even any inference that one wishes to make on such a model would be of poor significance, as each variable is marginally dependent on all the others: this is an implausible situation and other estimation criteria are considered next.

An estimation method for Stratified Staged Tree based on penalized log-likelihood enables us to reduce the number of degrees of freedom of the model through the joining in the same stage of vertices of the tree which provides an improvement of the penalized log-likelihood function. Denote with $\mathcal{M}_0$ the statistical model associated to a Stratified Staged Tree $\mathcal{T}_0$ with k stages, using all the notations introduced in Section 3.2. Suppose that for a stratum of the tree one wants to merge together two stages that in $\mathcal{M}_0$ were separated, creating an alternative model $\mathcal{M}_1$ with $k-1$ stages.

The penalized log-likelihood associated to $\mathcal{M}_0$ is reported in Equation (3.11), where λ is the penalization hyper-parameter and $|\mathcal{M}_0| = |\mathbf{p}| = \sum_{i=1}^{k} |\mathbf{p}_i|$, with $|\mathbf{p}_i|$ the cardinality of the parameters associated to the stage q_i, is the number of parameters estimated under $\mathcal{M}_0$. The likelihood function of $\mathcal{M}_0$ is the logarithm of Equation (3.3), from which adding a penalty term leads to the penalized log-likelihood in Equation (3.11).

Algorithm 1: AHC Algorithm

Input: A complete dataset, an event tree $\mathcal{T}$ and a prior parameters vector $\boldsymbol{p}$.

Output: The best scoring Stratified Staged Tree found.

Initialize an array $\mathbf{q}$ with the saturated stage structure of $\mathcal{T}_0 = \mathcal{T}$.
Initialize an array $\mathbf{y}$ with the conditional frequencies tables for each stage of $\mathcal{T}_0$ (stages sample sizes).
Obtain the maximum a posterior estimate $\hat{\mathbf{p}}^{\mathbf{MAP}}$ of the stage parameters vector $\mathbf{p}$ (Equation (3.6)).
Initialize the *score* with the log-posterior probabilities $\pi_0(\hat{\mathbf{p}}^{\mathbf{MAP}} \mid \mathbf{y})$ according to $\hat{\mathbf{p}}^{\mathbf{MAP}}$, $\mathbf{y}$ and $\mathbf{q}$ associated to $\mathcal{T}_0$.
stop $\leftarrow$ *FALSE*
while *stop = FALSE* **do**
    for *every pair of stages* $\boldsymbol{q}_i$ *and* $\boldsymbol{q}_j$ *with the same number* $k_i = k_j$ *of outgoing edges* **do**
        Calculate the log-posterior bayes factor (lpBF) (Equation (3.10)) between the stage structure that merges the stages $\boldsymbol{q}_i$ and $\boldsymbol{q}_j$ into the same stage $\boldsymbol{q}_{i-j} = \boldsymbol{q}_i \cup \boldsymbol{q}_j$ keeping all other stages invariant and the current stage structure $\mathbf{q}$.
        if *there does not exist any pair* $\boldsymbol{q}_i$ *and* $\boldsymbol{q}_j$ **then**
            stop $\leftarrow$ *TRUE*
    Take the pair of stages $\boldsymbol{q}_i^\star$ and $\boldsymbol{q}_j^\star$ that provides the largest lpBF.
    if $lpBF[\boldsymbol{q}_i^\star, \boldsymbol{q}_j^\star] > 0$ **then**
        score $\leftarrow$ *score* + $lpBF[\boldsymbol{q}_i^\star, \boldsymbol{q}_j^\star]$
        Update $\mathbf{q}$ gathering $\boldsymbol{q}_i^\star$ and $\boldsymbol{q}_j^\star$ into a single stage $\boldsymbol{q}_{i-j}^\star$. Let $k_{i-j}^\star$ be the number of outgoing edges from that stage.
        Update $\mathbf{y}$ gathering $y_i^\star$ and $y_j^\star$ into the new stage sample size $y_{i-j}^\star = y_i^\star + y_j^\star$.
        Update $\alpha_{i-j}^\star$ summing $\alpha_i^\star$ and $\alpha_j^\star$: $\alpha_{i-j}^\star = \alpha_i^\star + \alpha_j^\star$.
        Update the maximum a posterior estimate $\hat{p}_{i-j}^{\star\, MAP} = \frac{\alpha_{i-j}^\star + y_{i-j}^\star}{\sum_{t=1}^{k_{i-j}^\star} \alpha_{i-jt}^\star + y_{i-jt}^\star}$ of the unique component of stage parameters vector $\mathbf{p}$ that has to be updated.
    else
        stop $\leftarrow$ *TRUE*
return *score,* $\mathbf{q}$, $\mathbf{y}$ *and* $\boldsymbol{p}$.

$$l^{\mathcal{M}_0}(\mathbf{p} \mid \mathbf{y}, \lambda) \;=\; \log n! \;-\; \sum_{i=1}^{k}\sum_{j=1}^{k_i} \log y_{ij}! \;+\; \sum_{i=1}^{k}\sum_{j=1}^{k_i} y_{ij} \log p_{ij} \;-\; \lambda\, |\mathcal{M}_0| \quad (3.11)$$

For clarity of exposure, let q_1 and q_2 be the two separated stages in $\mathcal{M}_0$ that are

merged together in $\mathcal{M}_1$, defining the new stage q_{1-2} with the following properties:

- $k_{1-2} = k_1 = k_2$ number of outgoing edges from the florets of $\boldsymbol{q}_{1-2}$;
- $y_{1-2j} = y_{1j} + y_{2j}$, for $j = 1, \ldots, k_{1-2}$, number of observations within $\boldsymbol{q}_{1-2}$;
- $\mathbf{p}_{1-2}$ the $\boldsymbol{q}_{1-2}$ stage parameter vector.

The likelihood function associated to $\mathcal{M}_1$ is given in Equation (3.12).

$$L^{\mathcal{M}_1}(\mathbf{p} \mid \mathbf{y}) = \frac{n!}{\prod_{j=1}^{k_{1-2}} y_{1-2j}! \ \prod_{i=3}^{k} \prod_{j=1}^{k_i} y_{ij}!} \prod_{j=1}^{k_{1-2}} p_{1-2j}^{y_{1-2j}} \prod_{i=3}^{k} \prod_{j=1}^{k_i} p_{ij}{}^{y_{ij}} \tag{3.12}$$

Note that the same notation about the sample sizes and parameters associated to stages, which are stored respectively in $\mathbf{y}$ and $\mathbf{p}$, is adopted for both models $\mathcal{M}_0$ and $\mathcal{M}_1$. This is because these vectors represent the same quantities of interest, since the two models are nested, that is they are equivalent over all the tree except the new stage q_{1-2} in $\mathcal{M}_1$, involving a small difference in notation only for this stage.

The penalized log-likelihood of $\mathcal{M}_1$ is given in Equation (3.13), where $|\mathcal{M}_1| = |\mathbf{p}| = |\mathbf{p}_{1-2}| + \sum_{i=3}^{k_i} |\mathbf{p}_i|$.

$$\begin{aligned} l^{\mathcal{M}_1}(\mathbf{p} \mid \mathbf{y}, \lambda) \ = \ & \log n! \ - \ \sum_{j=1}^{k_{1-2}} \log y_{1-2j}! \ - \ \sum_{i=3}^{k} \sum_{j=1}^{k_i} \log y_{ij}! \\ & + \ \sum_{j=1}^{k_{1-2}} y_{1-2j} \log p_{1-2j} \ + \ \sum_{i=3}^{k} \sum_{j=1}^{k_i} y_{ij} \log p_{ij} \ - \ \lambda \, |\mathcal{M}_1| \end{aligned} \tag{3.13}$$

To test if the proposed alternative model $\mathcal{M}_1$ is preferable with respect to the original $\mathcal{M}_0$ the (log) **likelihood ratio test (LRT)** can be used [82]:

$$W(\mathbf{p}) = -2 \left(l_p^{\mathcal{M}_1}(\mathbf{p} \mid \mathbf{y}, \lambda) \ - \ l_p^{\mathcal{M}_0}(\mathbf{p} \mid \mathbf{y}, \lambda) \right) \ \sim \ \chi^2_{|\mathcal{M}_0| - |\mathcal{M}_1|},$$

testing the hypothesis:

$$\begin{aligned} H_0 : \ & \mathcal{M}_0 \ \text{ preferable model} \\ H_1 : \ & \mathcal{M}_1 \ \text{ preferable model.} \end{aligned}$$

The computations of the LRT in this framework are the following:

$$
\begin{aligned}
LRT = & -2 \left(\left(\log n! - \sum_{j=1}^{k_{1-2}} \log y_{1-2j}! - \sum_{i=3}^{k} \sum_{j=1}^{k_i} \log y_{ij}! \right.\right. \\
& \left. + \sum_{j=1}^{k_{1-2}} y_{1-2j} \log p_{1-2j} + \sum_{i=3}^{k} \sum_{j=1}^{k_i} y_{ij} \log p_{ij} - \lambda |\mathcal{M}_1| \right) \\
& - \left(\log n! - \sum_{i=1}^{k} \sum_{j=1}^{k_i} \log y_{ij}! + \sum_{i=1}^{k} \sum_{j=1}^{k_i} y_{ij} \log p_{ij} - \lambda |\mathcal{M}_0| \right) \Bigg) \\
= & -2 \left(\left(\sum_{j=1}^{k_{1-2}} y_{1-2j} \log p_{1-2j} - \sum_{j=1}^{k_{12}} \log y_{1-2j}! - \lambda |\mathcal{M}_1| \right) \right. \\
& \left. - \left(\sum_{j=1}^{2} \sum_{j=1}^{k_i} y_{ij} \log p_{ij} - \sum_{j=1}^{2} \sum_{j=1}^{k_i} \log y_{ij}! - \lambda |\mathcal{M}_0| \right) \right) \\
= & -2 \left(\sum_{j=1}^{k_{12}} y_{1-2j} \log p_{1-2j} - \sum_{j=1}^{2} \sum_{j=1}^{k_i} y_{ij} \log p_{ij} \right. \\
& \left. - \sum_{j=1}^{k_{12}} \log y_{1-2j}! + \sum_{j=1}^{2} \sum_{j=1}^{k_i} \log y_{ij}! - \lambda \left(|\mathcal{M}_1| - |\mathcal{M}_0| \right) \right) \sim \chi^2_{|\mathcal{M}_0|-|\mathcal{M}_1|},
\end{aligned}
$$

where the term $|\mathcal{M}_1| - |\mathcal{M}_0|$ associated to the penalization hyper-parameter λ is a negative number, since $\mathcal{M}_1$ is 1-nested in $\mathcal{M}_0$.

3.4.1 A Detailed Example

Let $\mathbf{X} = (X_1, X_2)$ be a random vector composed by two Bernoulli random variables. Suppose that the aim is to establish if X_2 is independent from X_1 or not: this can be done also using a Stratified Staged Tree. In this framework, the Probability Tree has one vertex related to the variable X_1, i.e. the root of the tree, two vertices in the second stratum for conditional distributions of X_2 given X_1 and four situations in the leaves stratum. The product space $\mathbb{X}$ has four elements since it is determined by all possible combinations of two Bernoulli variables, and this value corresponds to the number of leaves in the associated tree. In Figure 3.4 the Stratified Staged Trees corresponding to a dependence (Figure 3.4a) and an independence (Figure 3.4b) model are displayed. The stage structures associated to these two trees are the only possible ones, as the unique choice one can make is whether to put the situations v_2 and v_3 in the same stage or not. If they are set together, it means that the conditional probability distributions of X_2 given the two possible outcomes of X_1 are equal, i.e. X_1 and X_2 are independent. Denote this independence model with $\mathcal{M}_1$ and the dependence model with $\mathcal{M}_0$.

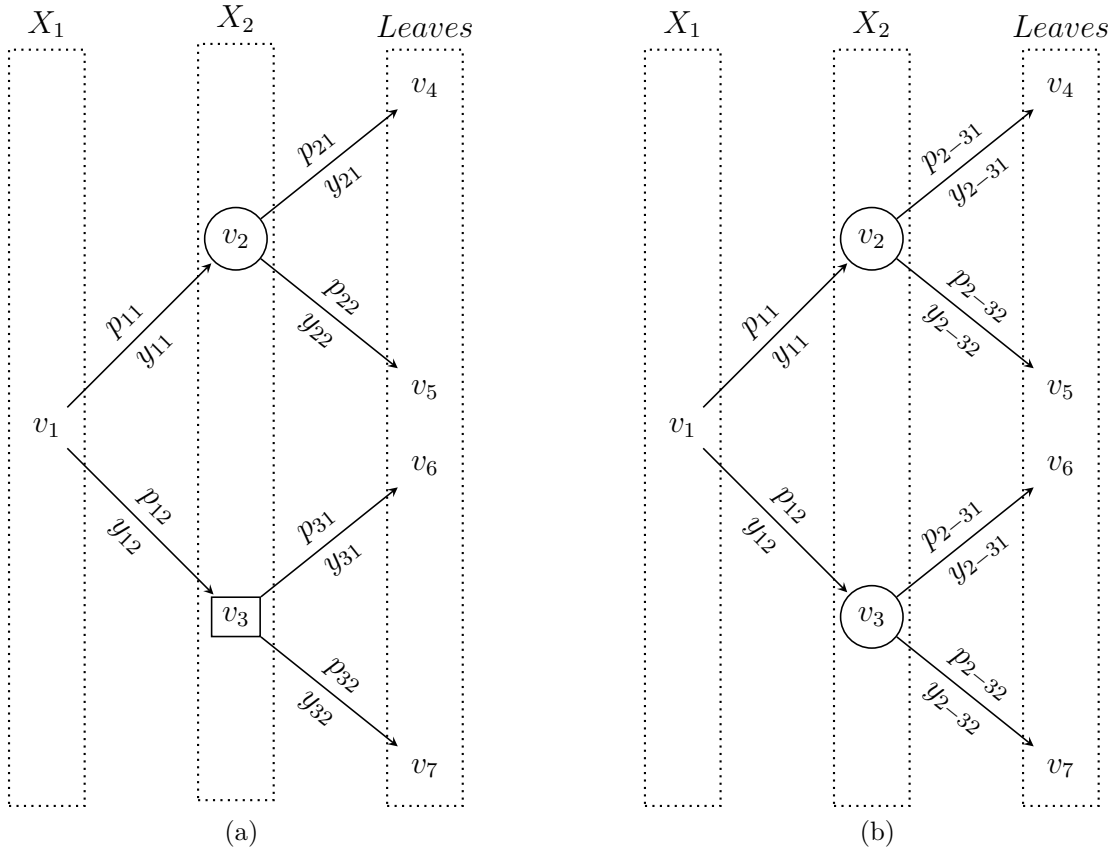


Figure 3.4: Stratified Staged Trees corresponding to $\mathcal{M}_0$ (a) and $\mathcal{M}_1$ (b). Stage structure for $\mathcal{M}_0$: $q_1 = \{v_1\},\ q_2 = \{v_2\},\ q_3 = \{v_3\}$. Stage structure for $\mathcal{M}_1$: $q_1 = \{v_1\},\ \boldsymbol{q}_{2-3} = \{v_2, v_3\}$.

The penalized log-likelihood function associated to the dependence model $\mathcal{M}_0$ with corresponding Stratified Staged Tree in Figure 3.4a is the following:

$$\begin{aligned} l_p^{\mathcal{M}_0}(\mathbf{p} \mid \mathbf{y}, \lambda) = {} & \log n! - \log y_{11}! - \log y_{12}! - \log y_{21}! - \log y_{22}! - \log y_{31}! \\ & - \log y_{32}! + y_{11} \log p_{11} + y_{12} \log p_{12} + y_{21} \log p_{21} \\ & + y_{22} \log p_{22} + y_{31} \log p_{31} + y_{32} \log p_{32} - 3\ \lambda, \end{aligned}$$

where the penalization term is $3\ \lambda$ because the parameters that have to be estimated are only $p_{11},\ p_{21},\ p_{31}$, since a Stratified Staged Tree has the property that characterized a Probability Tree introduced in the Section 3.2, namely it has the constraint to sum up to 1 for each floret distribution: $p_{12} = 1 - p_{11}$, $p_{22} = 1 - p_{21}$ and $p_{32} = 1 - p_{31}$.

The penalized log-likelihood for the independence model $\mathcal{M}_1$ with corresponding Stratified Staged Tree in Figure 3.4b is the following:

$$
\begin{aligned}
l_p^{\mathcal{M}_1}(\mathbf{p} \mid \mathbf{y}, \lambda) = \; & \log n! \; - \; \log y_{11}! \; - \; \log y_{12}! \; - \; \log y_{2-31}! \; - \; \log y_{2-32}! \\
& + \; y_{11} \log p_{11} \; + \; y_{12} \log p_{12} \; + \; y_{2-31} \log p_{2-31} \\
& + \; y_{2-32} \log p_{2-32} \; - \; 2 \; \lambda.
\end{aligned}
$$

The objective of this simple example is to determine if X_1 and X_2 are dependent or independent, which is reflected in identifying the preferable model among $\mathcal{M}_0$ and $\mathcal{M}_1$. This can be done through the log LR test as explained above, which in this framework has the following theoretical formulation:

$$
\begin{aligned}
LRT = \; & -2 \left(l_p^{\mathcal{M}_1} \; (\mathbf{p} \mid \mathbf{y}, \lambda) \; - \; l_p^{\mathcal{M}_0} \; (\mathbf{p} \mid \mathbf{y}, \lambda) \right) \\
= \; & -2 \left(- \log y_{2-31}! \; - \; \log y_{2-32}! \; + \; \log y_{21}! \; + \; \log y_{22}! \right. \\
& + \; \log y_{31}! \; + \; \log y_{32}! \; + \; y_{2-31} \log p_{2-31} \; + \; y_{2-32} \log p_{2-32} \\
& \left. - \; y_{21} \log p_{21} \; - \; y_{22} \log p_{22} \; - \; y_{31} \log p_{31} \; - \; y_{32} \log p_{32} \; + \; \lambda \right) \; \sim \; \chi_1^2.
\end{aligned}
$$

In a practical example, the log-likelihood ratio test can be computed returning a value. This is done by calculating the penalized log-likelihood function of the models $\mathcal{M}_0$ and $\mathcal{M}_1$ in the maximum likelihood estimation of the parameters vector $\mathbf{p}$: $\hat{\mathbf{p}}^{\mathbf{MLE}}$. This leads to the following formulation, in which by replacing all the sample sizes stored in $\mathbf{y}$ returns a value (fixing also a λ value) to be compared with the theoretical value of a Chi-Square distribution with one degree of freedom, which is, for instance at a significance level of 5%, 3.84.

$$
\begin{aligned}
L\hat{R}T = \; & -2 \left(l_p^{\mathcal{M}_1} \; (\hat{\mathbf{p}}^{\mathbf{MLE}} \mid \mathbf{y}, \lambda) \; - \; l_p^{\mathcal{M}_0} \; (\hat{\mathbf{p}}^{\mathbf{MLE}} \mid \mathbf{y}, \lambda) \right) \\
= \; & -2 \left(- \log y_{2-31}! \; - \; \log y_{2-32}! \; + \; \log y_{21}! \; + \; \log y_{22}! \; + \; \log y_{31}! \right. \\
& + \; \log y_{32}! \; + \; y_{2-31} \log \frac{y_{2-31}}{y_{2-31} \; + \; y_{2-32}} \; + \; y_{2-32} \log \frac{y_{2-32}}{y_{2-31} \; + \; y_{2-32}} \\
& - \; y_{21} \log \frac{y_{21}}{y_{21} \; + \; y_{22}} \; - \; y_{22} \log \frac{y_{22}}{y_{21} \; + \; y_{22}} \; - \; y_{31} \log \frac{y_{31}}{y_{31} \; + \; y_{32}} \\
& \left. - \; y_{32} \log \frac{y_{32}}{y_{31} \; + \; y_{32}} \; + \; \lambda \right) \; \sim \; \chi_1^2.
\end{aligned}
$$

Hence, if $L\hat{R}T$ is less than 3.84 the dependence model $\mathcal{M}_0$ is preferable with

respect to the independence one, $\mathcal{M}_1$, otherwise the latter is the best fitting model.

To conclude the example, the Chain Event Graphs associated to the Stratified Staged Trees in Figure 3.4 are reported in Figure 3.5.

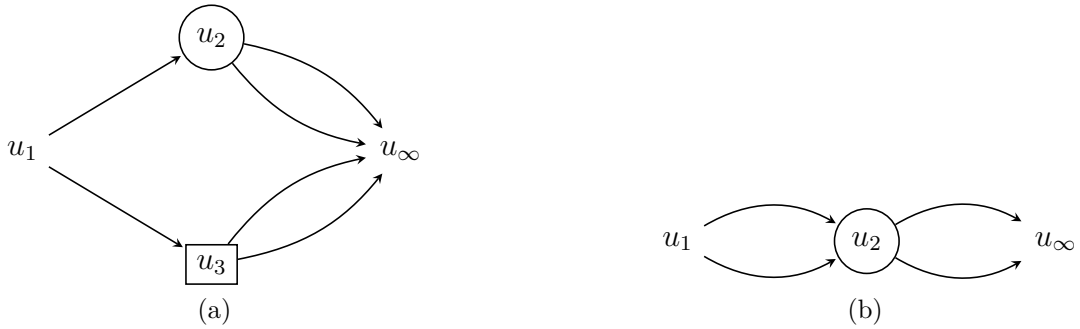


Figure 3.5: Corresponding Chain Event Graphs for Stratified Staged Trees in Figure 3.4. (a): positions for $\mathcal{M}_0$: $u_1 = \{v_1\}$, $u_2 = \{v_2\}$, $u_3 = \{v_3\}$ and $\boldsymbol{u}_\infty = \{v4, \ldots, v_7\}$. (b): Positions for $\mathcal{M}_1$: $u_1 = \{v_1\}$, $\boldsymbol{u}_2 = \{v_2, v_3\}$ and $\boldsymbol{u}_\infty = \{v4, \ldots, v_7\}$.

□

3.4.2 Penalized Log-Likelihood Algorithm

From an algorithmic point of view, the algorithm inherent to a model search based on penalized log-likelihood approach can be outlined along the lines of what was done for Agglomerative Hierarchical Clustering in the Section 3.3. Instead of using the likelihood ratio test, which is a test that returns simply a p-value of how much one model is likely preferable over another, the algorithm introduced in this thesis is based on a log-likelihood related quantity that can be updated on each its step. In particular, this quantity indicates if two different stages $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ in the Stratified Staged Tree $\mathcal{T}_i$ at a generic iteration i can be merged together, obtaining the new stage $\boldsymbol{q}_{1-2} = \boldsymbol{q}_1 \cup \boldsymbol{q}_2$ in the proposed Stratified Staged Tree $\mathcal{T}_{i+1}$. This is denoted with *Difference between Penalized Log-Likelihood (DPLL) functions*, it is a function of trees $\mathcal{T}_i$ and $\mathcal{T}_{i+1}$ and, using the notations already introduced previously, its formulation is reported in Equation (3.14). Also with this approach, since the Stratified Staged Tree $\mathcal{T}_{i+1}$ is 1-nested in $\mathcal{T}_i$, the corresponding stage structures $\mathbf{q^1}$ ($\mathcal{T}_i$) and $\mathbf{q^2}$ ($\mathcal{T}_{i+1}$) are different only in one component, then the DPPL is composed by a limited number of terms. Note also that DPLL corresponds to the quantity between the round brackets in the log-likelihood ratio test.

$$
\begin{aligned}
DPLL(\mathcal{T}_i, \mathcal{T}_{i+1}) \;&=\; l_p^{\mathcal{T}_{i+1}}(\mathbf{p} \mid \mathbf{y}, \lambda) \;-\; l_p^{\mathcal{T}_i}(\mathbf{p} \mid \mathbf{y}, \lambda) \\
&=\; \sum_{i=1}^{2}\sum_{j=1}^{k_i} \log y_{ij}! \;-\; \sum_{j=1}^{k_{1-2}} \log y_{1-2j}! \\
&+\; \sum_{j=1}^{k_{1-2}} y_{1-2j} \log p_{1-2j} \;-\; \sum_{i=1}^{2}\sum_{j=1}^{k_i} y_{ij} \log p_{ij} \;+\; \lambda\, k_{1-2}
\end{aligned}
\tag{3.14}
$$

Proposition 5. *Let $\mathcal{T}_i$ and $\mathcal{T}_{i+1}$ be two Stratified Staged Trees with stages structure $\mathbf{q^1}$ and $\mathbf{q^2}$ respectively. Assume that $\mathcal{T}_{i+1}$ is 1-nested in $\mathcal{T}_i$. Then, the configuration of the stages $\mathbf{q^2}$ in $\mathcal{T}_{i+1}$ is preferable with respect to $\mathbf{q^1}$ in $\mathcal{T}_i$ if:*

$$
DPLL(\mathcal{T}_i, \mathcal{T}_{i+1}) \;>\; 0
$$

$$
\Updownarrow
$$

$$
\sum_{i=1}^{2}\sum_{j=1}^{k_i} \log y_{ij}! - \sum_{j=1}^{k_{1-2}} \log y_{1-2j}! + \sum_{j=1}^{k_{1-2}} y_{1-2j} \log p_{1-2j} - \sum_{i=1}^{2}\sum_{j=1}^{k_i} y_{ij} \log p_{ij} \;>\; -\lambda\, k_{1-2}.
$$

Furthermore, the difference between the not penalized likelihood of the Stratified Staged Trees $\mathcal{T}_{i+1}$ and $\mathcal{T}_i$ (left-side of the previous inequality) is always less than zero, since $\mathcal{T}_{i+1}$ is 1-nested in $\mathcal{T}_i$, implying this that they have exactly the same stage structure except the union of the stages $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ in $\mathbf{q^2}$.

Therefore, at the i-th step of the algorithm the pair of stages $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ related to the same variable is merged into $\boldsymbol{q}_{1-2}$ if and only if:

$$
-\lambda\, k_{1-2} < \sum_{i=1}^{2}\sum_{j=1}^{k_i} \log y_{ij}! - \sum_{j=1}^{k_{1-2}} \log y_{1-2j}! + \sum_{j=1}^{k_{1-2}} y_{1-2j} \log p_{1-2j} - \sum_{i=1}^{2}\sum_{j=1}^{k_i} y_{ij} \log p_{ij} \;< 0.
$$

At each step of the algorithm, among all possible unions of stages that satisfy this inequality, the bigger one is chosen, i.e. the one that is closest to zero.

Finally, in Algorithm 2 a pseudo-code for the implementation of the algorithm based on penalized log-likelihood is reported.

Other interesting estimation criteria based on a penalization version of the log-likelihood are **Akaike Information Criterion (AIC)** and **Bayesian Information Criterion (BIC)**: their formulations are reported in Equation (3.15) and (3.16), respectively, where $|\mathcal{M}|$ is the number of estimated parameters in the model $\mathcal{M}$, n the number of observations in the collected sample and $\hat{l}$ the maximum value

Algorithm 2: Penalized Log-Likelihood Algorithm

Input: A complete dataset, an event tree $\mathcal{T}$ and a penalization hyper-parameter λ.

Output: The best scoring Stratified Staged Tree found.

Initialize an array $\mathbf{q}$ with the saturated stage structure of $\mathcal{T}_0 = \mathcal{T}$.
Initialize an array $\mathbf{y}$ with the conditional frequencies tables for each stage of $\mathcal{T}_0$ (stages sample sizes).
Obtain the maximum likelihood estimate $\hat{\mathbf{p}}^{\mathbf{MLE}}$ of the stage parameters vector $\mathbf{p}$ (Equation (3.7)).
Initialize the *score* with the maximum penalized log-likelihood $l_p^{\mathcal{T}_0}(\hat{\mathbf{p}}^{\mathbf{MLE}} \mid \mathbf{y}, \lambda)$ according to $\hat{\mathbf{p}}^{\mathbf{MLE}}$, $\mathbf{y}$ and $\mathbf{q}$ associated to $\mathcal{T}_0$.
stop $\leftarrow$ *FALSE*
while *stop = FALSE* **do**
    for *every pair of stages* q_i *and* q_j *with the same number* $k_i = k_j$ *of outgoing edges* **do**
        Calculate the difference between penalized log-likelihood (Equation (3.14)) functions between the proposal model with the stage structure that merges the stages q_i and q_j into the same stage $q_{i-j} = q_i \cup q_j$ keeping all other stages invariant and the model with the current stage structure $\mathbf{q}$.
        if *there does not exist any pair* q_i *and* q_j **then**
            stop $\leftarrow$ *TRUE*
    Take the pair of stages $q_i^\star$ and $q_j^\star$ that provides the largest DPLL.
    if $DPLL[q_i^\star, q_j^\star] > 0$ **then**
        score $\leftarrow$ *score* $+ DPLL[q_i^\star, q_j^\star]$
        Update $\mathbf{q}$ gathering $q_i^\star$ and $q_j^\star$ into a single stage $q_{i-j}^\star$. Let $k_{i-j}^\star$ be the number of outgoing edges from that stage.
        Update $\mathbf{y}$ gathering $y_i^\star$ and $y_j^\star$ into the new stage sample size $y_{i-j}^\star = y_i^\star + y_j^\star$.
        Update the maximum likelihood estimate $\hat{p}_{i-j}^{\star\ MLE} = \frac{y_{i-j}^\star}{\sum_{t=1}^{k_{i-j}^\star} y_{i-jt}^\star}$ of the unique component of stage parameters vector $\mathbf{p}$ that has to be updated.
    else
        stop $\leftarrow$ *TRUE*
return *score,* $\mathbf{q}$ *and* $\mathbf{y}$.

of the log-likelihood function for $\mathcal{M}$. The lower AIC and BIC values, the better the corresponding model $\mathcal{M}$.

$$\text{AIC} \;=\; 2\,|\mathcal{M}| \;-\; 2\,\hat{l} \tag{3.15}$$

$$\text{BIC} \;=\; |\mathcal{M}| \log\,(n) - \;2\,\hat{l} \tag{3.16}$$

When fitting models, it is possible to increase the likelihood corresponding to the model by adding parameters, but this may lead in the introduction of overfitting. AIC, BIC and DPLL attempt to resolve this problem by introducing a penalty term for the number of parameters estimated in the model. Note that the penalty term is larger in BIC than in AIC, so in general the best fitting model according to one of the two criteria is different from the best fitting according to the other one.

An ad-hoc algorithm can be implemented also for the minimization of AIC and BIC index: it is enough to change the equation used in line 8 of the pseudo-code in Algorithm 2. For this reason, this is not explicitly reported here.

3.5 Distance and Divergence Based Estimation

In this section an algorithm for estimating Stratified Staged Trees based on distance or divergence between discrete probability distributions is introduced. In particular, two situations located in the same stratum of the tree are merged together in the same stage if their probability distributions are close enough: how to establish if these distributions are close enough is detailed throughout this section.

Eight different distances/divergences are defined to compare conditional distributions associated to situations related to the same random variable, i.e. in the same stratum of the tree, in order to build the stage structure of the Stratified Staged Tree. Furthermore, relations that exist among some of these distances and their properties are discussed and the pseudo-code for the estimation of the Stratified Staged Tree based on these distances/divergences is reported in Algorithm 3. Throughout the section, a generic pair of discrete distributions over the same underlying set of events is denoted with $p(x)$ and $q(x)$. The metrics implemented in the *R* package **stagedtrees** are the following:

1. **Euclidean Distance**:

$$d(p,q) \;=\; \sqrt{\sum_{x\in\chi} \Big(p(x) \;-\; q(x)\Big)^2}\,. \tag{3.17}$$

2. **Manhattan Distance**, also known as *taxicab geometry* or l_1 *norm*:

$$d_1(p,q) \;= \sum_{x\in\chi} \Big|p(x) \;-\; q(x)\Big|. \tag{3.18}$$

3. **Kullback-Leibler Divergence**, also known as *Relative Entropy*:

$$D_{KL}(p \parallel q) \;=\; \sum_{x\in\chi} p(x)\,\log\frac{p(x)}{q(x)} \;=\; -\sum_{x\in\chi} p(x)\,\log\frac{q(x)}{p(x)}. \tag{3.19}$$

4. **Bhattacharyya Distance**:

$$D_B(p,q) \;=\; -\ln\Big(BC(p,q)\Big) \;=\; -\,ln\bigg(\sum_{x\in\chi}\sqrt{p(x)\;q(x)}\bigg), \tag{3.20}$$

   where $BC(p,q) \;=\; \sum_{x\in\chi}\sqrt{p(x)\;q(x)}$ is the *Bhattacharyya coefficient* and measure the amount of overlap between two statistical samples or populations.

5. **Rényi Divergence** generalizes the Kullback–Leibler divergence.
   For $0 \;\leq\; \alpha \;\leq\; \infty$ and $\alpha \;\neq\; 1$, the Rényi Divergence of order α (or α-*divergence*) of a distribution p from a distribution q is:

$$D_\alpha(p \parallel q) \;=\; \frac{1}{\alpha-1}\,\log\sum_{x\in\chi}\frac{p(x)^\alpha}{q(x)^{\alpha-1}} \tag{3.21}$$

6. **Hellinger Distance** is closely related to the Bhattacharyya distance:

$$H(p,q) \;=\; \frac{1}{\sqrt{2}}\sqrt{\sum_{x\in\chi}\Big(\sqrt{p(x)}-\sqrt{q(x)}\Big)^2} \tag{3.22}$$

7. **Total Variation Distance** is an example of a statistical distance metric and it is sometimes called *statistical distance* or *variational distance*:

$$\delta(p,q) \;=\; \frac{1}{2}\sum_{x\in\chi}|p(x) \;-\; q(x)| \tag{3.23}$$

8. **Chan-Darwiche Distance**:

$$CD(p,q) \;=\; \log R \;-\; \log r \;=\; \log\max_{x\in\chi}\frac{q(x)}{p(x)} \;-\; \log\min_{x\in\chi}\frac{q(x)}{p(x)}, \tag{3.24}$$

   where $R = \max\limits_{x\in\chi}\;\frac{q(x)}{p(x)}$ and $r = \min\limits_{x\in\chi}\;\frac{q(x)}{p(x)}$.

The main relations among these distances are reported in Appendix.

3.5.1 Distance and Divergence based Algorithm

From an algorithmic point of view, at each step of the algorithm, for a fixed stratum of the Stratified Staged Tree, the idea is to select the pair of stages that provides the smallest distance/divergence among the corresponding conditional probability distributions of these two stages. If this difference is smaller than a fixed threshold ϵ, then the two stages are merged together in the same stage and all the quantities of interest as for instance the penalized log-likelihood and the stage structure are updated. If for a stratum of the tree the smallest distance among probability distributions of stages is greater than ϵ, the algorithm moves on the next variable, i.e. stratum. When no union of stages for any stratum can be carried out, the algorithm ends.

There is no guarantee that at each step of this algorithm the penalized log-likelihood or some score based on it such as AIC or BIC is optimized. However, intuitively also the penalized log-likelihood intends to merge together stages with similar probability distributions (e.g. with a low distance), since from a penalized log-likelihood point of view it is convenient to reduce the number of parameters only in this setting. For this reason, in many of its steps also the distance based algorithm will provide an improvement of the penalized log-likelihood.

In Algorithm 3 the pseudo-code of the distance or divergence based algorithm for estimating the stage structure of a Stratified Staged Tree, where $f(p_i^\star, p_j^\star)$ represents any distance between a pair of probability distributions one wants to implement, for instance one of the eight presented in this section. In Section 5.5 an exaustive study using also these distances is carried out through the **stagedtrees** package.

Algorithm 3: Distance or Divergence Based Algorithm

Input: A complete dataset, an event tree $\mathcal{T}$, a penalization hyper-parameter λ and a threshold ϵ to evaluate distances between probability distributions of pairs of stages.

Output: The best scoring Stratified Staged Tree found.

Initialize an array $\mathbf{q}$ with the saturated stage structure of $\mathcal{T}_0 = \mathcal{T}$.
Initialize an array $\mathbf{y}$ with the conditional frequencies tables for each stage of $\mathcal{T}_0$ (stages sample sizes).
Obtain the maximum likelihood estimate $\hat{\mathbf{p}}^{\mathbf{MLE}}$ of the stage parameters vector $\mathbf{p}$, as shown in Equation (3.7).
Initialize the *score* with the maximum penalized log-likelihood $l_p^{\mathcal{T}_0}(\hat{\mathbf{p}}^{\mathbf{MLE}} \mid \mathbf{y}, \lambda)$ according to $\hat{\mathbf{p}}^{\mathbf{MLE}}$, $\mathbf{y}$ and $\mathbf{q}$ associated to $\mathcal{T}_0$.
stop $\leftarrow$ *FALSE*
while *stop = FALSE* **do**
    for *every pair of stages* q_i *and* q_j *with the same number* $k_i = k_j$ *of outgoing edges* **do**
        Calculate the Distance or Divergence between their probability distributions using f (Equations from (3.17) to (3.24)): $f(p_i, p_j)$.
        if *there does not exist any pair* q_i *and* q_j **then**
            stop $\leftarrow$ *TRUE*
    Take the pair of stages $q_i^\star$ and $q_j^\star$ that provides the smallest value of f.
    if $f[q_i^\star, q_j^\star] = f(p_i^\star, p_j^\star) < \epsilon$ **then**
        Calculate the difference between penalized log-likelihood (Equation (3.14)) functions between the proposal model with the stage structure that merges the stages q_i and q_j into the same stage $q_{i-j} = q_i \cup q_j$ keeping all other stages invariant and the model with the current stage structure $\mathbf{q}$.
        score $\leftarrow$ *score* + $DPLL[q_i^\star, q_j^\star]$
        Update $\mathbf{q}$ gathering $q_i^\star$ and $q_j^\star$ into a single stage $q_{i-j}^\star$. Let $k_{i-j}^\star$ be the number of outgoing edges from that stage.
        Update $\mathbf{y}$ gathering $y_i^\star$ and $y_j^\star$ into the new stage sample size $y_{i-j}^\star = y_i^\star + y_j^\star$.
        Update the maximum likelihood estimate $\hat{p}_{i-j}^{\star\ MLE} = \frac{y_{i-j}^\star}{\sum_{t=1}^{k_{i-j}^\star} y_{i-jt}^\star}$ of the unique component of stage parameters vector $\mathbf{p}$ that has to be updated.
    else
        stop $\leftarrow$ *TRUE*
return *score,* $\mathbf{q}$ *and* $\mathbf{y}$.

3.6 A Dynamic Programming Algorithm

Given a dataset and an order of its variables, an algorithm which returns the best fitting Stratified Staged Tree is described in Collazo et al. [27]. It performs an exhaustive and dynamic search in a very large model space. Its idea is to decompose the estimation of the global stage structure associated to the Staged Tree into many local estimates, more exactly one for each stratum of the tree. It explores all possible configurations of stages structures associated to each variable and selects the one that maximizes the optimization criterion, for instance the log-posterior Bayes Factor or the penalized log-likelihood function. From a theoretical point of view, this algorithm ensures that the global optimum according to a score-optimization criterion is achieved. Unfortunately, in practice it is unusable as soon as the number of the variables is greater than a very small number, such as 4. Indeed, the number of possible configurations has an highly exponential growth with respect to the number of vertices.

Figure 3.6 shows all possible configurations with five situations. Below each configuration the number of possible arrangements according to that type of configuration is reported. Considering only five situations, seven different configurations of stages can be set: the simplest is the one where all situations are in the same stage (independence), while when each situation is in a own stage characterizes the most complex one (full-dependence). We also have five intermediate configurations that can be obtained with a number of arrangements. For instance, the configuration where 4 situations are in the same stage and one is in a own stage can be achieved in five different ways, excluding each vertex one at a time from the stage composed by the remaining four.

In general, for a fixed combination, the number of possible settings for the stage structure is given by the *partitions* [92] of the number of vertices into at most a number of stages equal to that number, i.e. the limit case with each vertex in a own stage. The total number of partitions of n elements is the *Bell number* B_n, which satisfies the recursive formula displayed in Equation (3.25). Bell numbers can be calculated as in Equation (3.26), where $\left\{ {n \atop k} \right\}$ is the *Stirling partition number*. The first Bell numbers are the following: $B_0 = 1,\ B_1 = 1,\ B_2 = 2,\ B_3 = 5,\ B_4 = 15,\ B_5 = 52,\ B_6 = 203,\ B_7 = 877,\ B_8 = 4140,\ B_9 = 21147,\ B_{10} = 115975$. They exhibit an exponential growth, indeed B_{20} is already on the order of trillions.

$$B_{n+1} \ = \ \sum_{k=0}^{n} \binom{n}{k} B_k \tag{3.25}$$

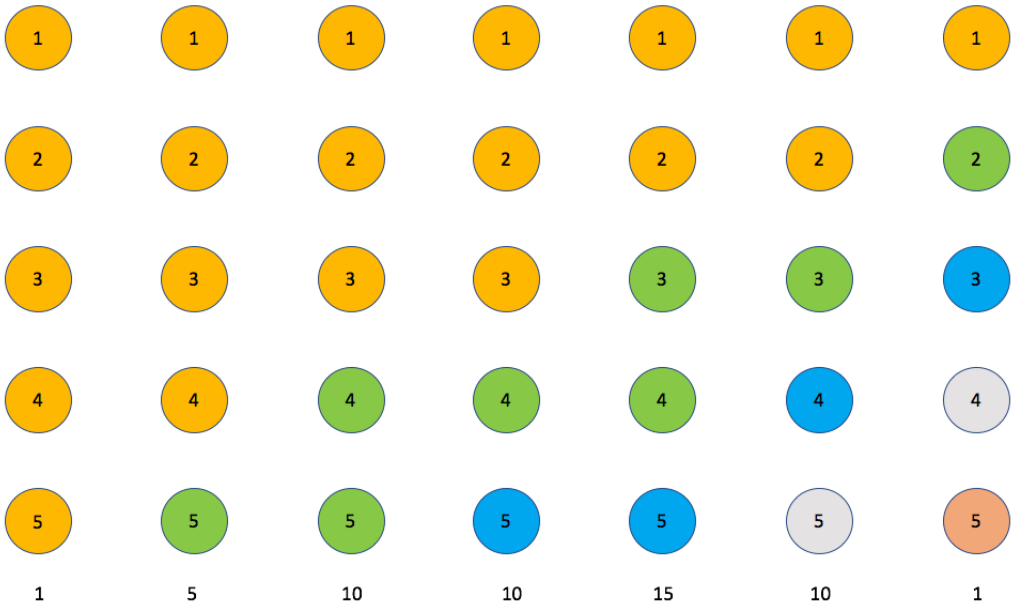


Figure 3.6: Possible configurations with five situations.

$$B_n = \sum_{k=0}^{n} \left\{ {n \atop k} \right\} = \sum_{k=0}^{n} \frac{1}{k!} \sum_{i=0}^{k} (-1)^i \binom{k}{i} (k-i)^n \tag{3.26}$$

In Table 3.2 the exact values obtained with a number of situations between one and twelve are shown. From a computational point of view, with only twelve situations the calculation of the best stage structure is already very expensive, since more than four millions of different stage structures have to be compared. Moreover, adding a situation leads to an increase of about five times the cardinality of partitions obtained with one less situation. This implies that with only twenty vertices about six thousand of billions of stage structures have to be compared and this results clearly intractable. Note that, even just with few variables, the corresponding Staged Tree has tens and tens of situations. In conclusion, this algorithm is really appealing from a theoretical point of view, but it is unfortunately totally unusable in application frameworks. For this, only score-optimizations based on local optimum can be implemented, as for instance the hill-climbing model search adopted in previous algorithms.

3.7 Order of Variables in Stratified Staged Trees

It is well known that the ordering of the variables in a Bayesian Network affects the quality of the inference that can be done through it. Also for Stratified Staged Trees, the order of the variables plays a crucial role for the model learning. This is

# Situations	# Configurations	# Partitions
1	1	1
2	2	2
3	3	5
4	5	15
5	7	52
6	11	203
7	15	877
8	22	4140
9	30	21147
10	42	115975
11	56	678570
12	77	4213597

Table 3.2: Cardinality of possible configurations with a given number of situations.

because contingency tables associated to nodes of the tree, from which conditional probability distributions are estimated, are constructed according to the order in which variables are placed in the strata of the tree. At the end of this section an exhaustive simulation over two datasets is performed, estimating the stage structures associated to Stratified Staged Trees based on each possible ordering of the variables. This study shows that changing the order of the variables leads to quite different performances of learning algorithms according to the accuracy of the predicted values they provide. The outcome of interest, on which the accuracy is computed as the proportion of corrected predictions for the whole dataset, is a Bernoulli variable for both studies; details will be given in Section 3.7.1. Accuracy was chosen since one of the contexts in which it is very useful to select the order of the variables is that of the prediction of a categorical variable.

In this section three different fundamentals of probability or information theory are proposed to be used to infer a variables ordering. Let X, Y and Z be three categorical random variables with sample spaces $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$, respectively. Then, three criteria can be introduced as follows:

- *Mutual Information* between X and Y:

$$I(X,Y) \;=\; D_{KL}(p(x,y) \,\|\, p(x)\,p(y)) \;=\; \sum_{y\in\mathcal{Y}} \sum_{x\in\mathcal{X}} p(x,y) \log \frac{p(x,y)}{p(x)\;p(y)}, \quad (3.27)$$

  where $D_{KL}(p(x,y) \;\|\; p(x)\;p(y))$ is the Kullback-Leibler Divergence between the joint distribution of X and Y and the product of the two marginal ones.

- *Conditional Mutual Information* between X and Y given Z:

$$I(X;Y|Z) \;=\; \sum_{z\in\mathcal{Z}} \sum_{y\in\mathcal{Y}} \sum_{x\in\mathcal{X}} p(x,y,z)\, \log \frac{p(x,y,z)\; p(z)}{p(x,z)\; p(y,z)}. \tag{3.28}$$

- *Conditional Entropy* of X given Y:

$$H(X|Y) \;=\; -\sum_{y\in\mathcal{Y}} \sum_{x\in\mathcal{X}} p(x,y)\, \log \frac{p(x,y)}{p(y)}, \tag{3.29}$$

  remembering that the marginal entropy of X is $H(X) = -\sum_{x\in X}\; p(x)\log p(x)$.

These quantities have many properties, including some that make it possible to express one of these fundamentals as a function of other. The ten main ones are reported in the appendix.

Each variable ordering algorithm assumes that the variable associated to the root of the tree is the response variable. This is suggested for a classification usage of Stratified Staged Trees and for a clearer interpretation in a sense explained in Carli et al. [21]. The requirement of the variable of interest being the root of the tree follows from the idea that one may want that it has no parents, in order to maximize the information provided by the features. This is because, otherwise, features not connected to the response variable, i.e. variables located in previous strata with respect to the response one, would not provide any information for classification. Details on this choice are also given in Section 6.6, where the usage of the R package **stagedtrees** for a classification purpose is demonstrated.

Note that not for a classification purpose, once a Staged Tree has been fixed, if the response variable is placed in internal strata, then the learning carried out on its ancestors within the tree explains relationships about the response variable (through Bayesian rules).

The pseudo-code of the algorithm based on the minimization of conditional entropy is reported in Algorithm 4. Let $\mathbf{X} = (X_1, \ldots, X_p)$ be the random vector of interest and X_1 be the response variable. The first step of Algorithm 4 consists in identifying the variable $X_i \in \mathbf{X}$, $i = 2, \ldots, p$, such that the entropy of its probability distribution given the response variable X_1 is minimal. The choice of the minimum guarantees that the most unbalanced probability distribution, and then the most influential variable according to the conditional entropy, is selected. At the $i-$th step of the algorithm, the variable which provides the smallest entropy of its distribution conditionally on the previous $i-1$ variables is chosen.

The general idea is that the minimization of the entropy of the probability distribution of a variable conditionally on all the variables that precede it in the ordering

is associated to the assignment of more importance to the distributions of those variables that have not a uniform behavior. Indeed, the entropy is maximized in the presence of maximum indecision (e.g. uniform distribution) and is minimized where there is no indecision (indicator variable). For example, the outcome of the toss of a fair coin represents the maximum uncertainty (maximum entropy), while the outcome of the throw of a rigged die with all the faces equal corresponds to an exact event (minimum entropy).

Algorithm 4: Variables Ordering according to Conditional Entropy

Input: A complete dataset.
Output: The best variable ordering for Stratified Staged Tree.

```
Initialize an array order with the name of the response variable. For ease
  of exposure, let X_1 be the response variable.
Initialize an array names with the name of all variables in the dataset,
  except X_1.
Initialize an array score with the entropy of the marginal probability
  distribution of X_1.
while names ≠ ∅ do
    for every variable's name nam ∈ names do
        Calculate the conditional entropy H of the probability distribution
          of nam given order (Equation (3.29)).
    Take the variable's name nam* which provides the smallest value of
      conditional entropy H*.
    Update order by adding nam*.
    Update names by removing nam*.
    score = score + H*.
return order and score.
```

The ordering of variables attained with mutual information or conditional mutual information is based on the maximization of these quantities, since both measure the information of the system. The idea of the algorithm based on conditional mutual information is the same as for conditional entropy, with the exception of the maximization of this quantity instead of its minimization, as just mentioned; the pseudo-code in reported in Algorithm 5. At each step of the algorithm the mutual information between the response variable and the proposed variable, conditionally on variables placed in the ordering between these two, is computed.

At last, the pseudo-code of the mutual information based algorithm is displayed in Algorithm 6.

These three algorithms do not find a global optimum according to some performance criterion. This is because at each step of algorithms the best variable is chosen accordingly, but we have no guarantee that the final result is the best over-

Algorithm 5: Variables Ordering according to Conditional Mutual Information

Input: A complete dataset.
Output: The best variable ordering for Stratified Staged Tree.

Initialize an array **order** with the name of the response variable. For ease of exposure, let X_1 be the response variable.
Initialize an array **names** with the name of all variables in the dataset, except X_1.
for *every variable's name* `nam` $\in$ ***names*** **do**
    Calculate the mutual information I of response variable X_1 and `nam` (Equation (3.27)).
Take the variable's name `nam`* which provides the greatest value I^* of mutual information with the response variable.
Update **order** by adding `nam`*.
Update **names** by removing `nam`*.
Initialize **score** with the mutual information I^*.
while ***names*** $\neq \emptyset$ **do**
    for *every variable's name* `nam` $\in$ ***names*** **do**
        Calculate the conditional mutual information I (Equation (3.28)) of X_1 and `nam` given **order** (excluding X_1).
    Take the variable's name `nam`* which provides the largest value of conditional mutual information I^*.
    Update **order** by adding `nam`*.
    Update **names** by removing `nam`*.
    score = **score** $+ I^*$.
return **order** and **score**.

all. Their usage is useful to produce a variables ordering that can be justified as it optimizes, locally step by step, foundations of information or probability theory.

3.7.1 An Application: Staged Tree Classifiers

A brief empirical study is conducted to shown that classification accuracies of Staged Tree Classifiers estimated according to the conditional mutual information algorithm for ordering the variables are better than the average computed on those of all possible ordering. Staged Tree Classifiers will be discussed in detail in Section 6.6, showing that different orders of the variables lead to different classification. They are Staged Trees seen as classification tool.

Two datasets with a small amount of features, namely `puffin` and `monks3` (see Table 6.2 for details) are considered. In particular, using the whole dataset, we compute the accuracy in classifying the response variable, located at the root of the tree, according to all the possible variables ordering. The datasets are not divided in train

Algorithm 6: Variables Ordering according to Mutual Information

Input: A complete dataset.

Output: The best variable ordering for Stratified Staged Tree.

1 Initialize an array **order** with the name of the response variable. For ease of exposure, let X_1 be the response variable.

2 Initialize an array **names** with the name of all variables in the dataset, except X_1.

3 Initialize **score** $= 0$.

4 **for** *every variable's name* `nam` $\in$ ***names*** **do**

5 Calculate the mutual information $I(X_1,$ `nam`$)$ of response variable X_1 and `nam` (Equation (3.27)).

6 Update **order**: order the variable's name `nam` $\in$ **names** in decreasing ordering according to the value of mutual information $I(X_1,$ `nam`$)$ they provide jointly with X_1.

7 **score** $= \sum_{\texttt{nam} \in \textbf{names}} I(X_1,$ `nam`$)$.

8 Update **names** with the empty set $\emptyset$.

9 **return order** and **score**.

and test set, since the purpose of this section is not to produce inference according to the forecasted values but is only to show how the variables ordering algorithms perform. The accuracy is computed as the proportion of corrected predictions for the whole dataset.

Two learning algorithms are designed:

- `ST_Naive`, the Naive Staged Tree Classifier learnt with hierarchical clustering of probability distributions in two different stages for each stratum of trees (details in Chapter 6);
- `ST_FBHC`, the Staged Tree Classifier learnt with the fast backward hill-climbing algorithm, maximizing the negative BIC score (see Section 5.2).

Figure 3.7 shows the *violin plots* of the distributions of accuracies for `monks3` and `puffin`, achieved with `ST_Naive` and `ST_FBHC` and using all possible orders of variables; the accuracy obtained through the CMI variables ordering is indicated with a cross symbol. The figure shows that, as expected, the order of the features is highly relevant with respect to classification performance. For instance, both learning algorithms exhibit a variety of accuracies (from 0.5 to 1) for the `monks3` dataset.

It is therefore critical to couple any algorithm to learn a Staged Tree with an appropriate method to select a good ordering of the variables. Here the use of the conditional mutual information (CMI) criterion is adopted as a representative for

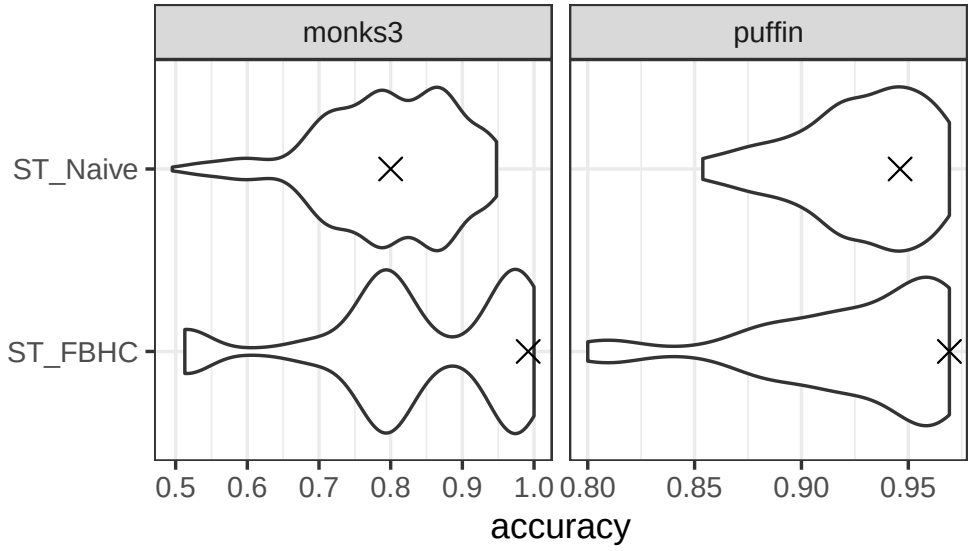


Figure 3.7: Violin plots of distribution of accuracy for all possible orders of features. Accuracies obtained with the CMI variables ordering are shown with a cross. Results using two learning algorithms for datasets `monks3` and `puffin`.

the three algorithms proposed in this section, since the other two methods provide comparable results, showing accuracies better than average and similar to the one obtained through CMI.

It can be observed that in both `monks3` and `puffin` datasets, the CMI order leads to a Staged Tree which performs better than the majority of all possible orders. Remember that high accuracy values correspond to preferable models. In particular, the fast backward hill-climbing algorithm provides almost the global optimum according to the accuracy of its estimates for both the analyzed datasets.

3.8 Confidence Intervals for Stages Probability Distributions

In this section multiple confidence intervals for stages probabilities are considered. The probability distribution associated to the i-th stage belongs to the Multinomial family:

$$Y_i \mid P_i = \boldsymbol{p}_i \sim Multinom(y_i, \boldsymbol{p}_i),$$

where $y_i = \sum_{j=1}^{k_i} y_{ij}$ is the number of observations in the i-th stage and $\boldsymbol{p}_i = (p_{i1}, \ldots, p_{ik_i})$, for $i = 1, \ldots, k$. Let $\boldsymbol{y}_i = (y_{i1}, \ldots, y_{ik_i})$ be the vector of observed cell counts for the k_i outgoing edges from stage i, for $i = 1, \ldots, p$. Assuming the

sample size y_i is fixed, the vector $\boldsymbol{y}_i$ is an observation from a Multinomial distribution with parameters $\boldsymbol{p}_i = (p_{i1}, \ldots, p_{ik_i})$. The vector $\hat{\boldsymbol{p}}_i = (\hat{p}_{i1}, \ldots, \hat{p}_{ik_i})$, with $\hat{p}_{ij} = \frac{y_{ij}}{y_i}$, is the maximum likelihood estimation of $\boldsymbol{p}_i$ and is unbiased. The variance of the estimation $\hat{p}_{ij}$ is $\frac{p_{ij}(1-p_{ij})}{y_{ij}}$ and is usually estimated by $\frac{\hat{p}_{ij}(1-\hat{p}_{ij})}{y_{ij}}$.

Four different methods to compute confidence intervals are outlined in this work: *Goodman* [51], *Wald* [123], *Waldcc* [123] and *Wilson* [127].

For ease of exposure, we consider two-sided confidence intervals and we focus on a single stage. Clearly the procedure can be generalized for all the stages and also for one-sided intervals.

The Goodman confidence intervals meet the requirement that the corresponding confidence statement about all the p_{ij}, for $j = 1, \ldots, k_i$, will be correct with probability at least equal to $1 - \alpha$, with α the desired confidence level. The confidence interval for each parameter p_{ij} is computed as $L < p_{ij} < U$, with L and U the lower and upper bounds of that interval, respectively.

In Equations (3.30) and (3.31) the computations to obtain L and U according to Goodman method are displayed, where B is the upper $\frac{\alpha}{k_i}$ percentile of a Chi Square distribution with 1 degree of freedom: $B = \chi^2_1\ (1 - \alpha + \frac{\alpha}{k_i})$.

$$L = \frac{B + 2\ y_{ij} - \sqrt{B\ \left(B + \frac{4\ y_{ij}\ (y_i\ -\ y_{ij})}{y_i}\right)}}{2\ (B + y_i)} \tag{3.30}$$

$$U = \frac{B + 2\ y_{ij} + \sqrt{B\ \left(B + \frac{4\ y_{ij}\ (y_i\ -\ y_{ij})}{y_i}\right)}}{2\ (B + y_i)} \tag{3.31}$$

A Bonferroni correction is used in order to obtain simultaneously k_i confidence intervals for $\boldsymbol{p}_i$ [2]. This is reflected in the selected percentile of the Chi Square distribution for the value of B, which is more conservative ($\frac{\alpha}{k_i}$) with respect to the original one (α). This implies that the probability that the confidence interval thus obtained will be incorrect for a single given parameter p_{ij} is ($\frac{\alpha}{k_i}$) and, therefore, the probability is α (or less) that at least one of the k_i confidence intervals will be incorrect.

The endpoints L and U of the confidence interval according to the Wald method are

$$L = \hat{p}_{ij} - \sqrt{\frac{\chi^2_1(1-\alpha)\ \ \hat{p}_{ij}(1-\hat{p}_{ij})}{y_i}} \tag{3.32}$$

$$U = \hat{p}_{ij} + \sqrt{\frac{\chi^2_1(1-\alpha)\ \ \hat{p}_{ij}(1-\hat{p}_{ij})}{y_i}}. \tag{3.33}$$

The Waldcc method is a continuity correction of the Wald confidence intervals and lower and upper values can be implemented through Equations (3.34) and (3.35) respectively. This correction expands the interval by $\frac{1}{y_i}$ with respect to the one obtained through Wald, which represents a significant alteration in the presence of small sample sizes y_i, while it is negligible as y_i grows.

$$L = \hat{p}_{ij} - \sqrt{\frac{\chi^2_1(1-\alpha)\ \ \hat{p}_{ij}(1-\hat{p}_{ij})}{y_i}} - \frac{1}{2\ y_i} \tag{3.34}$$

$$U = \hat{p}_{ij} + \sqrt{\frac{\chi^2_1(1-\alpha)\ \ \hat{p}_{ij}(1-\hat{p}_{ij})}{y_i}} + \frac{1}{2\ y_i} \tag{3.35}$$

At last, the Wilson method is similar to the Goodman method, with a modification on B: $B^* = \chi^2_1(1-\alpha)$.

$$L = \frac{B^* + 2\ y_{ij} - \sqrt{B^{*2} + 4\ y_{ij}\ B^*\ (1 - \frac{y_{ij}}{y_i})}}{2\ (B^* + y_i)} \tag{3.36}$$

$$U = \frac{B^* + 2\ y_{ij} + \sqrt{B^{*2} + 4\ y_{ij}\ B^*\ (1 - \frac{y_{ij}}{y_i})}}{2\ (B^* + y_i)} \tag{3.37}$$

The main problems that can be encountered in estimating confidence intervals on Stratified Staged Trees are the presence of very small sample sizes in some nodes of the tree and the presence of highly unbalanced, almost degenerate, distributions. The first problem can also be the cause of the second, especially in the strata of the tree next to the leaves. Confidence intervals that have been estimated in one of these two scenarios may be therefore biased. For details, see Glaz and Sison [50] and Möstel et al. [91]. Clearly, a credible interval using a Bayesian approach through the posterior distribution of parameters could be defined, which would also help to overcome the above problems. But, since the thesis is mainly based on a frequentist approach, these intervals are not explored here.

Consider a simple example with two variables, `Class` and `Sex`, on which a Stratified Staged Tree has been estimated using a backward hill-climbing search starting from the full dependence model. The resulting tree is depicted in Figure 3.8, showing the partial conditional independence

$$\texttt{Sex} \perp\!\!\!\perp \texttt{Class} \mid \texttt{Class} = \{1st, 2nd\},$$

which corresponds to merging the two nodes corresponding to the paths `Class` = $1st$ and `Class` = $2nd$ into the "black" stage for the `Sex` stratum.

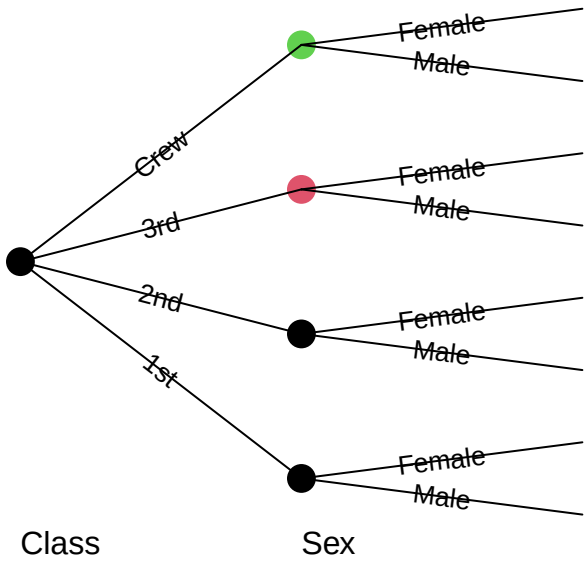


Figure 3.8: Example of a Stratified Staged Tree based on 2 variables: `Class` and `Sex`.

The estimated transition probabilities and the corresponding confidence intervals obtained through the four methods described in this section are shown in Table 3.3. In particular, the distribution associated to the root vertex is a Multinomial with 4 possible outcomes, while the distributions associated to stages for `Sex` are Binomial distributions.

The confidence intervals obtained through the Goodman method have a greater width because of the Bonferroni correction, which implies that the probability that the true parameter lies within the interval, for each parameter of the Multinomial distribution of a fixed stage, is $1 - \alpha + \frac{\alpha}{k_i}$. This guarantees that the simultaneous k_i confidence intervals have $1 - \alpha$ as coverage.

			Wald		Waldcc		Goodman		Wilson	
Stage	Event	MLE	2.5 %	97.5 %	2.5 %	97.5 %	2.5 %	97.5 %	2.5 %	97.5 %
1	`Class` $= 1st$	0.1477	0.1328	0.1625	0.1326	0.1627	0.1298	0.1675	0.1335	0.1631
1	`Class` $= 2nd$	0.1295	0.1155	0.1435	0.1152	0.1437	0.1127	0.1484	0.1161	0.1442
1	`Class` $= 3rd$	0.3208	0.3013	0.3403	0.3010	0.3405	0.2964	0.3461	0.3016	0.3406
1	`Class` $= Crew$	0.4020	0.3816	0.4226	0.3814	0.4228	0.3763	0.4284	0.3818	0.4227
1	`Sex` $= Male$	0.5885	0.5495	0.6276	0.5487	0.6284	0.5433	0.6323	0.5490	0.6269
1	`Sex` $= Female$	0.4115	0.3724	0.4505	0.3716	0.4513	0.3677	0.4567	0.3731	0.4510
2	`Sex` $= Male$	0.7224	0.6893	0.7554	0.6886	0.7561	0.6831	0.7585	0.6882	0.7541
2	`Sex` $= Female$	0.2776	0.2446	0.3107	0.2439	0.3114	0.2415	0.3169	0.2459	0.3118
3	`Sex` $= Male$	0.9740	0.9635	0.9845	0.9630	0.9851	0.9591	0.9836	0.9613	0.9826
3	`Sex` $= Female$	0.0260	0.0155	0.0365	0.0149	0.0370	0.0164	0.0409	0.0174	0.0387

Table 3.3: Confidence intervals obtained with *Wald*, *Waldcc*, *Goodman* and *Wilson* methods for the estimated parameters of the Stratified Staged Tree in Figure 3.8.

3.9 Conclusion

The chapter provides the main definitions of Staged Trees and Chain Event Graphs. In Section 3.3 the Agglomerative Hierarchical Clustering [46] is presented. It is the first algorithm for learning a Staged Tree from data. Sections 3.4 and 3.5 show two innovative estimation criteria for the learning of stages, which are based on penalized log-likelihood and distance/divergence between pair of probability distributions of stages, respectively. Section 3.6 empirically demonstrates that the Dynamic Programming algorithm, which is appealing from a theoretical point of view since it searches the global optimum of a score function, can not be used in practical applications. Section 3.7 suggests three algorithms to infer an order of variables for the building of the Staged Tree structure. These algorithms are based on conditional entropy and conditional mutual information. Finally, how to calculate confidence intervals in the context of Staged Trees is presented in Section 3.8. Four methods are developed, one (*Goodman*) resulting the most appropriate for transition probabilities for stages with Multinomial distributions with more than two possible outcomes, since it uses a Bonferroni correction. The other three methods proposed are *Wald, Waldcc* and *Wilson*, which nevertheless represent suitable alternatives to *Goodman*.

Chapter 4

Asymmetry-labeled DAGs

This chapter follows Varando et al. [121] and explains in detail the relation existing between Bayesian Networks and Staged Trees. Every BN can be represented as a Staged Tree, as demonstrated by Smith and Anderson [107] and by Duarte and Solus [38]. In this chapter, first an algorithm for the Staged Tree representation of a given BN is provided following Smith and Anderson [107] and, second, the minimal BN representation of a Staged Tree which embeds all its conditional independences is introduced. It allows the introduction of a criterion to identify all conditional independences implied by a Staged Tree model.

The presence or absence of edges in a BNs encodes either (conditionally) full dependence or independence between two variables. The flexibility of the Staged Tree enables to model and consequently identify intermediate relationships among variables, namely *context-specific*, *partial* or *local* [97]. Here, it is proposed to address this problem by defining classes of dependences among variables and introducing methods to identify the appropriate class from the Staged Tree. This leads to the definition of a new class of DAGs, called **asymmetry-labeled DAGs (ALDAGs)**, which is a DAG whose edges are colored according to the type of relationship existing between the variables corresponding to the edge nodes. Learning algorithms for ALDAGs, which use any structural learning algorithm for Staged Trees, are discussed below and applied to various datasets. Some definitions may be repeated from previous sections for making the chapter more self-contained.

4.1 Bayesian Networks and Conditional Independence

Let $\boldsymbol{X} = (X_1, \ldots, X_p) = (X_i)_{i\in[p]}$, with $[p] = \{1, \ldots, p\}$, be a categorical random vector with joint distribution probability P and sample space $\mathbb{X} = \times_{i\in[p]}\mathbb{X}_i$. For $A \subset [p]$, let be $\boldsymbol{X}_A = (X_i)_{i\in A}$ and $\boldsymbol{x}_A = (x_i)_{i\in A}$, where $\boldsymbol{x}_A \in \mathbb{X}_A = \times_{i\in A}\mathbb{X}_i$. Then, a directed acyclic graph (DAG) for $\boldsymbol{X}$ can be written as $G = ([p], E)$, indicating with $V = [p]$ the vertex set and E the edge set.

We say that the joint distribution P is Markov with respect to G if, for $\boldsymbol{x} \in \mathbb{X}$,

$$P(\boldsymbol{x}) = \prod_{k\in[p]} P(x_k \mid \boldsymbol{x}_{pa(k)}), \tag{4.1}$$

where $pa(k)$ is the parent set of k in G and $P(x_k \mid \boldsymbol{x}_{pa(k)})$ is a shorthand for $P(X_k = x_k \mid \boldsymbol{X}_{pa(k)} = \boldsymbol{x}_{pa(k)})$. The Markov condition implies conditional independences of the form

$$X_i \perp\!\!\!\perp \boldsymbol{X}_{[i-1]} \mid \boldsymbol{X}_{pa(i)}, \tag{4.2}$$

which are equivalent to

$$P(x_i \mid \boldsymbol{x}_{[i-1]\setminus pa(i)}, \boldsymbol{x}_{pa(i)}) \;=\; P(x_i \mid \boldsymbol{x}_{pa(i)}), \tag{4.3}$$

for all $\boldsymbol{x} \in \mathbb{X}$. Henceforth, P is assumed to be strictly positive.

Definition 26. *The Bayesian Network model (associated to G) is*

$$\mathcal{M}_G = \{P \in \Delta^\circ_{|\mathbb{X}|-1} \mid P \text{ is Markov to } G\},$$

where $\Delta^\circ_{|\mathbb{X}|-1}$ is the $(|\mathbb{X}|-1)$-dimensional open probability simplex.

It is customary to label the vertices of a BN to respect a topological order of G (see Definition 12). In the sequel we assume that the vertices of the DAG are ordered according to a given topological order.

To illustrate the methodology, throughout the section the `Titanic` dataset [33] will be used, which provides information on the fate of the Titanic passengers and available from the **datasets** package bundled in *R*. `Titanic` includes four categorical variables: `Class` (C) has four levels whilst `Gender` (G), `Survived` (S) and `Age` (A) are binary. The BN learned using the hill-climbing algorithm implemented in the *R* package **bnlearn** is reported in Figure 4.1 and embeds only the following conditional

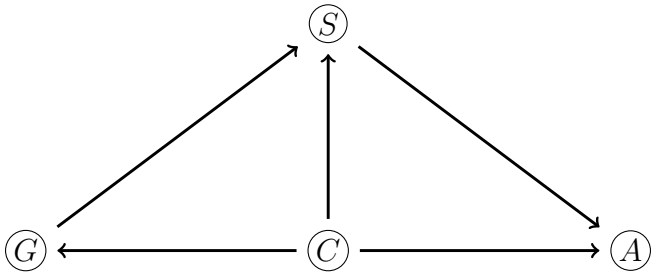


Figure 4.1: Learned BN on the `Titanic` dataset.

independence statement:

$$\texttt{Age} \perp\!\!\!\perp \texttt{Gender} \mid \texttt{Class}, \texttt{Survived}. \tag{4.4}$$

There is only one topological order of the variables: C, G, S, A.

4.1.1 Non-Symmetric Conditional Independence

Three types of non-symmetric conditional independences can be considered, according to Pensar et al. [97]. Let $\boldsymbol{X}_A$, $\boldsymbol{X}_B$ and $\boldsymbol{X}_C$ be three disjoint subsets of $(X_i)_{i\in[p]}$.

Definition 27. *(Context-specific Conditional Independence). We say that $\boldsymbol{X}_A$ is context-specific independent of $\boldsymbol{X}_B$ given context $\boldsymbol{x}_C \in \mathbb{X}_C$ if*

$$P(\boldsymbol{x}_A \mid \boldsymbol{x}_B, \boldsymbol{x}_C) \;=\; P(\boldsymbol{x}_A \mid \boldsymbol{x}_C) \tag{4.5}$$

holds for all $(\boldsymbol{x}_A, \boldsymbol{x}_B) \in \mathbb{X}_{A\cup B}$ and write $\boldsymbol{X}_A \perp\!\!\!\perp \boldsymbol{X}_B \mid \boldsymbol{x}_C$.

The condition in Equation (4.5) reduces to standard conditional independence in Equation (4.2) if it holds for all $\boldsymbol{x}_C \in \mathbb{X}_C$.

Definition 28. *(Partial Conditional Independence). We say that $\boldsymbol{X}_A$ is partially conditionally independent of $\boldsymbol{X}_B$ in the domain $\mathcal{D}_B \subseteq \mathbb{X}_B$ given context $\boldsymbol{X}_C = \boldsymbol{x}_C$ if*

$$P(\boldsymbol{x}_A \mid \boldsymbol{x}_B, \boldsymbol{x}_C) \;=\; P(\boldsymbol{x}_A \mid \tilde{\boldsymbol{x}}_B, \boldsymbol{x}_C) \tag{4.6}$$

holds for all $(\boldsymbol{x}_A, \boldsymbol{x}_B), (\boldsymbol{x}_A, \tilde{\boldsymbol{x}}_B) \in \mathbb{X}_A \times \mathcal{D}_B$ and write $\boldsymbol{X}_A \perp\!\!\!\perp \boldsymbol{X}_B \mid \mathcal{D}_B, \boldsymbol{x}_C$.

Clearly, Equation (4.5) and Equation (4.6) coincide if $\mathcal{D}_B = \mathbb{X}_B$. Furthermore, the sample space $\mathbb{X}_B$ must contain more than two elements for a non-trivial partial conditional independence to hold.

Definition 29. *(Local Conditional Independence). For $i \in [p]$ and $A \subset [p]$ such that $A \cap \{i\} = \emptyset$, local conditional independence expresses identifications of probabilities*

of the form

$$P(x_i \mid \boldsymbol{x}_A) \;=\; P(x_i \mid \tilde{\boldsymbol{x}}_A) \tag{4.7}$$

for all $x_i \in \mathbb{X}_i$ *and two* $\boldsymbol{x}_A, \tilde{\boldsymbol{x}}_A \in \mathbb{X}_A$.

Notice that in terms of generality, Equation (4.5) $\preceq$ Equation (4.6) $\preceq$ Equation (4.7). Condition in Equation (4.7) simply states that some conditional probability distributions are identified, where no discernable patterns as in Equations (4.5) and (4.6) can be detected.

Differently to any other Probabilistic Graphical Model, the class of Staged Trees is capable of graphically represents and formally encodes any of the types of conditional independences defined in Equations from (4.3) to (4.7).

4.2 Notation

Let $\mathcal{T} = (V, E)$ be a directed, finite, rooted tree with vertex set V, root node v_0 and edge set E. As introduced in Chapter 3, let Θ be a non-empty set of labels and $\theta : E \to \Theta$ be a function such that for any non-leaf $v \in V$ the labels in $\theta(E(v))$ are all distinct. Then, the equivalence classes induced by $\theta(E(v))$ form a partition $\boldsymbol{q}$ of the internal vertices of the tree in stages. We also have that, for all $v \in V$, θ satisfies $\theta(v, \mathbf{x}_{[i]}) = (\kappa(v), x_i)$ for some function $\kappa : V \to \mathcal{C}$, where $\mathcal{C}$ is a set of labels.

As an illustration, Figure 4.2 depicts a small Staged Tree: the labeling θ is given by the colored text labels in the edges. The Staged Tree represents a possible model for sequential testing and isolation policies for individuals in an infectious disease scenario. In particular, an individual can get tested or not with some probability; if tested, the result is positive or negative and this affects the probability of being isolated or not. Similarly, if an individual is not tested at first, he/she could get tested or not in a second time and again the result of the test influences if isolation is applied. The staging of the Staged Tree in Figure 4.2 is given by the partition $\boldsymbol{q} = (\boldsymbol{q}_1 = \{v_0,\ v_1\},\ \boldsymbol{q}_2 = \{v_2, v_4\},\ \boldsymbol{q}_3 = \{v_3\},\ \boldsymbol{q}_4 = \{v_6, v_{10}\},\ \boldsymbol{q}_5 = \{v_5, v_9\})$. The associated model describes equalities between conditional distributions of events, for example the fact that v_0 and v_1 are in the same stage indicates that the probability of being tested is equal to the probability of being tested in a second time given that in a first time the individual has not been tested. Similarly, the stages $\boldsymbol{q}_4$ and $\boldsymbol{q}_5$ represent the fact that the probability of being isolated depends only on the results of the test for an individual who has been tested.

The parameter space Θ_T associated to a Staged Tree $\mathcal{T} = (V, E)$ is defined as in Equation (4.8), which defines a class of probability mass functions over the edges

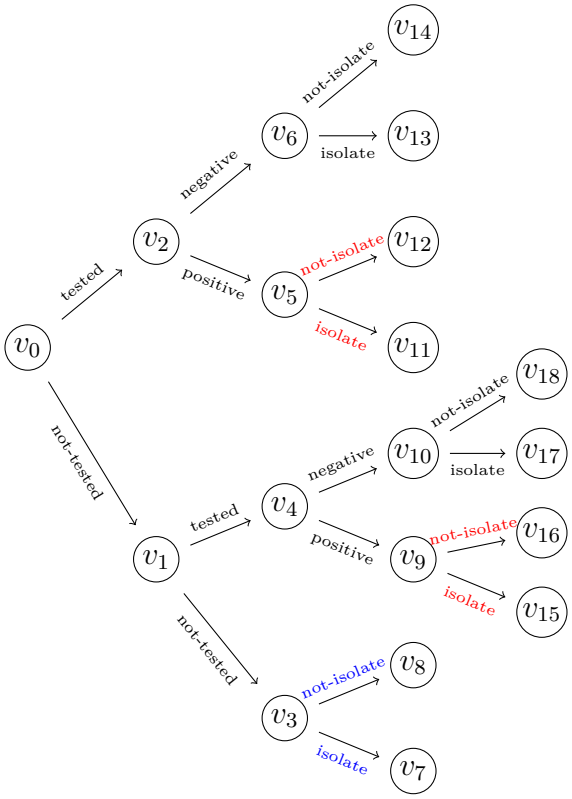


Figure 4.2: A small Staged Tree representing sequential testing and isolation policies in an infectious disease scenario.

emanating from any non-leaf vertex.

$$\Theta_T = \left\{ \boldsymbol{x} \in \mathbb{R}^{|\theta(E)|} \mid \forall\ e \in E, x_{\theta(e)} \in (0,1) \text{ and } \sum_{e \in E(v)} x_{\theta(e)} = 1 \right\} \quad (4.8)$$

Definition 30. *The* Staged Tree model $\mathcal{M}_{\mathcal{T}}$ *associated to the Staged Tree* $\mathcal{T}$ *with labeling* θ *is the image of the map*

$$\begin{aligned} &\phi_{\mathcal{T}} : \Theta_T \to \Delta^{\circ}_{|l_{\mathcal{T}}|-1}; \\ &\phi_{\mathcal{T}} : x \mapsto p_l = \left(\prod_{e \in E(\lambda(l))} x_{\theta(e)} \right)_{l \in \boldsymbol{l}_{\mathcal{T}}}. \end{aligned} \quad (4.9)$$

The definition of Staged Trees model in Definition 30 is based on the topology of the underlying directed tree. Although not strictly required, it is customary for such a tree to represent the sample space of a categorical random vector $\boldsymbol{X}$ and its labeling to represent non-symmetric conditional independences over the components of $\boldsymbol{X}$. These Staged Trees are called **X**-compatible, as already mentioned. Henceforth, and as common in practice, here the focus is on **X**-compatible Staged Trees, since they naturally model distributions of categorical random vectors. From now on, denote

with $\boldsymbol{x}_{-i}$ the vector $\boldsymbol{x}$ without its i-th entry.

Figure 4.3 reports an example of an **X**-compatible Staged Tree model for the Titanic dataset learned with the *R* package **stagedtrees**. The *coloring* given by the function κ is shown in the vertices and each edge $(\cdot, (x_1, \ldots, x_i))$ is labeled with x_i. The edge labeling θ can be read from the graph combining the text label and the color of the emanating vertex. For example, $\theta(v_1, v_5) \neq \theta(v_1, v_6) = \theta(v_2, v_8) \neq \theta(v_3, v_{10}) \neq \theta(v_5, v_{14})$. This representation of the labeling θ over vertices is equivalent to that over edges, whilst being more interpretable. There are 29 non-leaf vertices and the stage structure is $\boldsymbol{q} = (\boldsymbol{q}_1 = \{v_0\}, \boldsymbol{q}_2 = \{v_1, v_2\}, \boldsymbol{q}_3 = \{v_3\}, \boldsymbol{q}_4 = \{v_4\}, \boldsymbol{q}_5 = \{v_5, v_{10}\}, \boldsymbol{q}_6 = \{v_6\}, \boldsymbol{q}_7 = \{v_7, v_9\}, \boldsymbol{q}_8 = \{v_8, v_{12}\}, \boldsymbol{q}_9 = \{v_{11}\}, \boldsymbol{q}_{10} = \{v_{14}, v_{21}, v_{22}\}, \boldsymbol{q}_{11} = \{v_{13}, v_{15}, v_{16}, v_{17}, v_{19}, v_{25}, v_{26}, v_{27}, v_{28}\}, \boldsymbol{q}_{12} = \{v_{18}\}, \boldsymbol{q}_{13} = \{v_{20}, v_{22}\}, \boldsymbol{q}_{14} = \{v_{23}, v_{24}\})$.

Definition 31. *Let $\mathcal{T} = (V, E)$ be a Staged Tree. The* depth *of a vertex $v \in V$ is equal to the number of edges in $\lambda(v)$ and denote with V_k the set of vertices in $\mathcal{T}$ with depth k.*

Using Definition 31, Stratified Staged Trees are Staged Trees where vertices in the same stage have the same depth; see for instance the Staged Tree in Figure 4.3, that is equal colors at different depths can not correspond to same staging.

Conditional independence is formally modeled and represented in Staged Trees via the labeling θ. As an illustration consider the Staged Tree in Figure 4.3 for the `Titanic` dataset. The fact that v_1 and v_2 are in the same stage represents the partial independence $\texttt{Gender} \perp\!\!\!\perp \texttt{Class} \mid \{\texttt{1st}, \texttt{2nd}\}$. Considering vertices at depth two, $\boldsymbol{q}_7$ and $\boldsymbol{q}_8$ again represents partial conditional independences. More interesting is the stage $\boldsymbol{q}_5$, which implies

$$P(S = s \mid \texttt{Female}, \texttt{3rd}) = P(S = s \mid \texttt{Male}, \texttt{1st}),$$

i.e. the probability of survival for females travelling in third class is the same as that of male travelling in first class. Such a statement is a generic local conditional independence. Considering the last level, it can be noticed a very non-symmetric staging structure. For instance, the fact that the top four vertices v_{25}, v_{26}, v_{27} and v_{28} belong to the same stage implies the context-specific independence

$$\texttt{Age} \perp\!\!\!\perp \texttt{Survived}, \texttt{Gender} \mid \texttt{Class} = \texttt{Crew}.$$

The Staged Tree in Figure 4.3, embedding the above non-symmetric conditional independences, gives a better representation of the data than the BN in Figure 4.1. Indeed, the BIC of the Staged Tree can be computed as 10440.39, whilst the one of the BN is larger and equal to 10502.28.

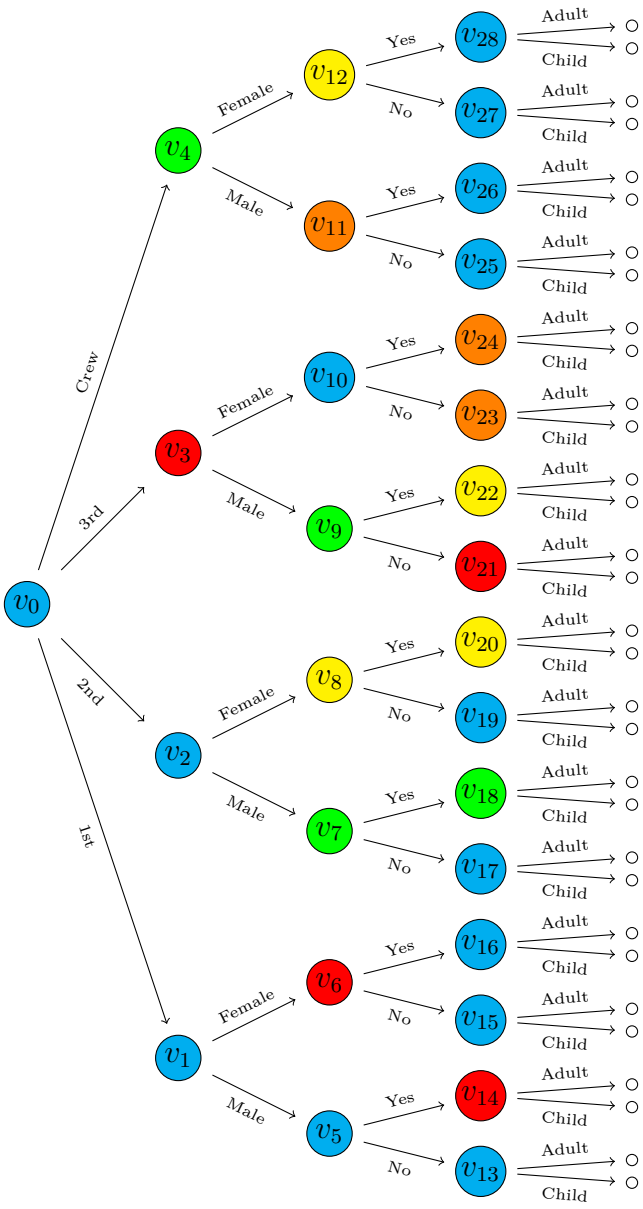


Figure 4.3: A Staged Tree compatible with (`Class`, `Gender`, `Survived`, `Age`), learned over the `Titanic` dataset.

4.2.1 Staged Trees and Bayesian Networks

Although the relationship between BNs and Staged Trees was already formalized by Smith and Anderson [107], a formal procedure to represent a BN as a Staged Tree has been only recently introduced in Duarte and Solus [38]. This is reviewed next.

Assume $\boldsymbol{X}$ is topologically ordered with respect to a DAG G and consider an $\mathbf{X}$-compatible Staged Tree with vertex set V, edge set E and labeling θ defined via the coloring $\kappa(\boldsymbol{x}_{[i]}) = \boldsymbol{x}_{pa(i)}$ of the vertices. The Staged Tree $\mathcal{T}_G$, with vertex set V, edge set E and labeling θ so constructed, is called the *Staged Tree model of* G. Importantly, $\mathcal{M}_G = \mathcal{M}_{\mathcal{T}_G}$, i.e. the two models are exactly the same, since they entail exactly the same factorization of the joint probability. Clearly, $\mathcal{T}_G$ is stratified and its staging represents the Markov conditions associated to the graph G.

As an illustration, Figure 4.4 reports the tree $\mathcal{T}_G$ associated to the BN in Figure 4.1. Since the variables `Class`, `Gender` and `Survived` are fully connected in the BN, the associated Staged Tree representation is such that vertices with depth one and two are in their own individual stages. The only symmetric conditional independence embedded in the BN given in Equation (4.4) is represented by joining pairs of vertices with depth three in the same stage. Clearly, the structure of the Staged Tree representing a BN in Figure 4.4 exhibits a lot more symmetry than the one in Figure 4.3, whose staging can represent a wide array of non-symmetric independences.

Another contribution is the solution of the following inverse problem: given an $\mathbf{X}$-compatible Stratified Staged Tree $\mathcal{T} = (V, E)$ with labeling θ, find the corresponding DAG G. The resulting DAG can not represent, in general, the same model as the Staged Tree, since BNs can not represent non-symmetric conditional independences. Nevertheless, in Varando et al. [121] the authors prove that a minimal DAG can be retrieved, in a sense that is formalized next.

Lemma 1. *Let $\mathcal{T} = (V, E)$ with labeling θ and $\mathcal{T}' = (V, E)$ with labeling θ' be two Staged Trees with same tree (V, E). We have that $\mathcal{M}_T \subseteq \mathcal{M}_{T'}$ if and only if for each $e, f \in E$, $\theta'(e) = \theta'(f) \;\Rightarrow\; \theta(e) = \theta(f)$.*

Proof. The proof follows directly from the definition of $\mathcal{M}_{\mathcal{T}}$. □

Proposition 6. *Let $\mathcal{T} = (V, E)$ with labeling θ be an $\mathbf{X}$-compatible Stratified Staged Tree, with $\kappa : V \to \mathcal{C}$ the vertex labeling that defines θ. Let $G_{\mathcal{T}} = ([p], F_T)$ be the DAG with vertex set $[p]$ and whose edge set F_T includes the edge $(k, i), k < i$, if and only if there exist $\mathbf{x}_{[i-1]}, \mathbf{x'}_{[i-1]} \in \mathbb{X}_{[i-1]}$ such that $x_j = x'_j$ for all $j \neq k$ and*

$$\kappa(\mathbf{x}_{[i-1]}) \neq \kappa(\mathbf{x'}_{[i-1]}). \tag{4.10}$$

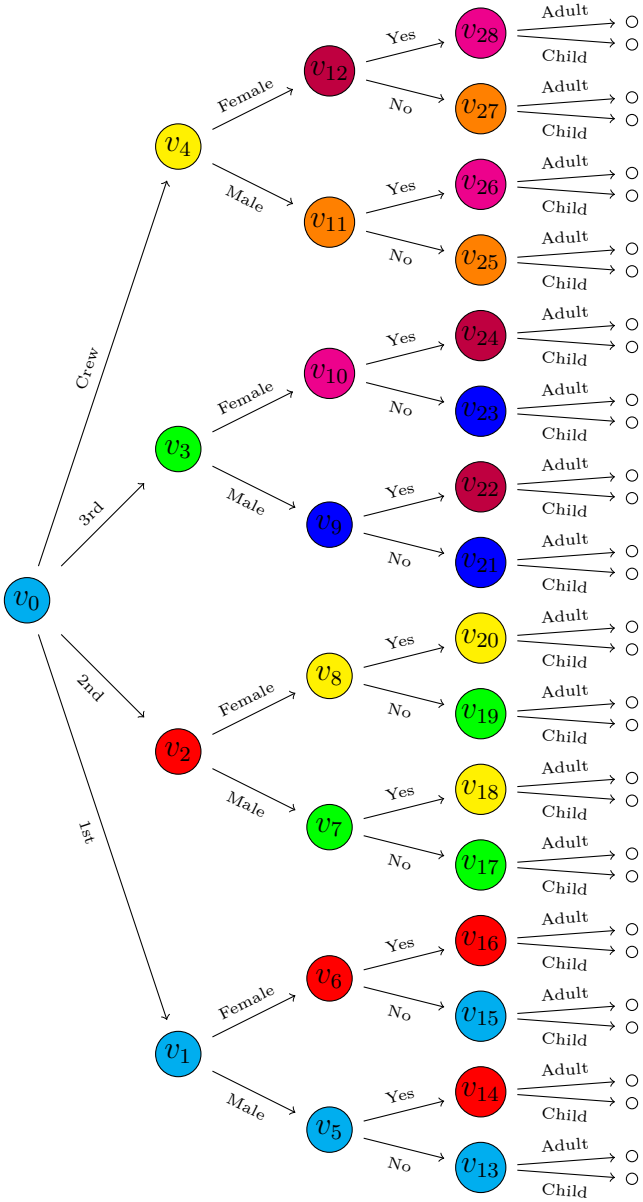


Figure 4.4: The Staged Tree representation of the BN in Figure 4.1 for the `Titanic` dataset.

Then $G_{\mathcal{T}} = ([p], F_T)$ is the minimal DAG such that $\mathcal{M}_{\mathcal{T}} \subseteq \mathcal{M}_{G_{\mathcal{T}}}$, in the sense that for every DAG $G = ([p], F)$ such that $1, \ldots, p$ is a topological order, if $\mathcal{M}_{\mathcal{T}} \subseteq \mathcal{M}_G$ then $F_T \subseteq F$.

In particular $X_A \perp\!\!\!\perp X_B \mid X_C$ holds in $\mathcal{M}_{\mathcal{T}}$ if and only if A and B are d-separated by C in $G_{\mathcal{T}}$.

Proof. Let start by proving that $\mathcal{M}_{\mathcal{T}} \subseteq \mathcal{M}_{G_{\mathcal{T}}}$. We have that $\mathcal{M}_{G_{\mathcal{T}}} = \mathcal{M}_{\mathcal{T}_{G_{\mathcal{T}}}}$ and let θ' be the labeling of $T_{G_{\mathcal{T}}}$. Both $\mathcal{T}$ and $\mathcal{T}_{G_{\mathcal{T}}}$ are $\mathbf{X}$-compatible Staged Tree and thus share the same vertex and edge set. Suppose that for $\mathbf{x}_{[i-1]}, \mathbf{y}_{[i-1]} \in \mathbb{X}_{[i-1]}$,

$\kappa(\mathbf{x}_{[i-1]}) \neq \kappa(\mathbf{y}_{[i-1]})$. We define now a sequence of nodes $\mathbf{x}_{[ii-1]} = \mathbf{x}^0_{[i-1]}, \ldots, \mathbf{x}^{i-1}_{[i-1]} = \mathbf{y}_{[i-1]}$ in $\mathbb{X}_{[i-1]} \subseteq V$ as following,

$$\mathbf{x}^0_{[i-1]} = \mathbf{x}_{[i-1]}, \quad x^h_j = \begin{cases} x^{h-1}_j & j > h \\ y_j & j \leq h \end{cases} .$$

Define $h_0 = \min\{h \text{ s.t. } \kappa(\mathbf{x}^h_{[i-1]}) \neq \kappa(\mathbf{x}^{h-1}_{[i-1]})\}$. We have that $k(\mathbf{x}_{[i-1]}) = \ldots = \kappa(\mathbf{x}^{h_0-1}_{[i-1]}) \neq \kappa(\mathbf{x}^{h_0}_{[i-1]})$ and thus, by construction of $G_\mathcal{T}$, $(h_0, i) \in F_T$ that in turn implies $\theta'(\mathbf{x}_{[i-1]}, x_i) \neq \theta'(\mathbf{y}_{[i-1]}, x_i)$ for all $x_i \in \mathbb{X}_i$. We thus have $\mathcal{M}_\mathcal{T} \subseteq \mathcal{M}_{\mathcal{T}_{G_\mathcal{T}}} = \mathcal{M}_{G_\mathcal{T}}$ by Lemma 1.

Assume now $G = ([p], F)$ is a DAG with $1, \ldots, p$ as a topological ordering and such that $\mathcal{M}_\mathcal{T} \subseteq \mathcal{M}_G$, then $\mathcal{M}_\mathcal{T} \subseteq \mathcal{M}_{\mathcal{T}_G}$ and if γ is the labeling of $\mathcal{T}_G$ we have that $\gamma(e) = \gamma(f) \Rightarrow \theta(e) = \theta(f)$ for each $e, f \in E$ by Lemma 1. Let $(k, i) \in F_T$ be an edge of $G_\mathcal{T}$, then by construction there exist $\mathbf{x}_{[i-1]}, \mathbf{x}'_{[i-1]} \in \mathbb{X}_{[i-1]}$ with $x_j = x'_j, j \neq k$ such that $\kappa(\mathbf{x}_{[i-1]}) \neq \kappa(\mathbf{x}'_{[i-1]})$. Hence, $\gamma(\mathbf{x}_{[i-1]}, (\mathbf{x}_{[i-1]}, x_i)) \neq \gamma(\mathbf{x}'_{[i-1]}, (\mathbf{x}'_{[i-1]}, x_i)))$ and it is easy to see that this implies $(h, i) \in F$ by construction of $\mathcal{T}_G$.

□

Corollary 1. *In the setup of Proposition 6,* $\mathcal{M}_{\mathcal{T}_G} = \mathcal{M}_{G_{\mathcal{T}_G}}$.

A Staged Tree $\mathcal{T}$ is therefore a sub-model of the resulting $G_\mathcal{T}$ which embeds the same set of symmetric conditional independences. The BN $G_\mathcal{T}$ is minimal in the sense that it includes the smallest number of edges among all possible BNs that include $\mathcal{M}_\mathcal{T}$ as a sub-model. The models $\mathcal{M}_\mathcal{T}$ and $\mathcal{M}_{G_\mathcal{T}}$ are exactly equal if and only if $\mathcal{T}$ embeds only symmetric conditional independences. As an illustration consider the Staged Tree in Figure 4.3. It can be shown using Proposition 6 that the associated BN $G_\mathcal{T}$ is complete and, since $G_\mathcal{T}$ is the saturated model, it must be that $\mathcal{M}_\mathcal{T} \subseteq \mathcal{M}_{G_\mathcal{T}}$. Conversely, if Proposition 6 is used to transform the Staged Tree in Figure 4.4 into a BN, using Corollary 1 the resulting BN must be the one in Figure 4.1.

Importantly, Proposition 6 gives a novel criterion to read symmetric conditional independence statements from a Staged Tree, by transforming it into a BN whose structure represents the same equalities of the form in Equation (4.3). Conditional independence statements in the Staged Tree can then be read from the associated BN using the d-separation criterion (see Proposition 2 and [74]). For instance, the Staged Tree in Figure 4.3 does not embed any symmetric conditional independence, since the associated BN is complete. In next sections how to identify non-symmetric conditional independences in Staged Trees is discussed.

4.3 Non-Symmetric Dependence and DAGs: AL-DAGs

Proposition 6 identifies if there is a dependence between two random variables in a $\mathbf{X}$-compatible Staged Tree $\mathcal{T}$ and in such a case draws an edge in $G_{\mathcal{T}}$. However, the Staged Tree carries a lot more information about the type of relationship existing between the two variables. In this section methods to label the edges of $G_{\mathcal{T}}$ are introduced in such a way that the labeling provides some of the information stored in the Staged Tree $\mathcal{T}$ about the underlying non-symmetric independences.

4.3.1 Classes of Statistical Dependence

First, it is needed to characterize the type of dependence existing between two random variables that are joined by an edge in a DAG G.

Definition 32. *Let P be the joint probability distribution of $\mathbf{X}$ taking values in $\mathbb{X}$ and assume P is Markov with respect to a DAG $G = ([p], E)$. For each $(j, i) \in E$ we say that the dependence of X_i from X_j is of class*

- context*, if X_i and X_j are context-specific independent given some context $\mathbf{x}_C$ with $C = pa(i) \setminus \{j\}$.*
- partial*, if X_i is partially conditionally independent of X_j in a domain $\mathcal{D}_j \subset \mathbb{X}_j$ given a context $\mathbf{x}_C$ with $C = pa(i) \setminus \{j\}$; and X_i and X_j are not context-specific independent given the same context $\mathbf{x}_C$.*
- local*, if none of the above hold and a local independence of the form $P(x_i \mid \mathbf{x}_{pa(i)}) = P(x_i \mid \tilde{\mathbf{x}}_{pa(i)})$ is valid where $x_j \neq \tilde{x}_j$.*
- total*, if none of the above hold.*

Notice that if the class of dependence between X_i and X_j is context or partial then there may also be local independence statements as in Equation (4.7) involving these two variables. Similarly, the dependence between X_i and X_j can be both context and partial with respect to two different contexts. On the other hand if their class of dependence is local, by definition there are no context-specific or partial equalities.

Proposition 6 paves the way to assess the class of dependence existing between X_i and X_j. In particular, one has to check if there are equalities of the form in Equation (4.10) for all or some $\boldsymbol{x}_{[i]} \in \mathbb{X}_{[i]}$ and, if so, to which class they correspond.

4.3.2 Asymmetry-Labeled DAGs

An edge in a BN represents, by construction, a total dependence between two random variables. However, the flexibility of Staged Trees enables us to assess if such a dependence is of any other of the classes introduced in Definition 32. This leads us to define a new graphical representation, called *asymmetry-labeled DAG* (ALDAG), whose edges are colored depending on the type of relationship among variables.

Formally, let G be the DAG and E its edge set. Let

$$\mathcal{L}^A = \{\text{'context'}, \text{'partial'}, \text{'context/partial'}, \text{'local'}, \text{'total'}\}$$

be the set of edge labels marking the type of dependence.

Definition 33. *An ALDAG is a pair (G, ψ) where $G = ([p], E)$ is a DAG and ψ is a function from the edge set of G to $\mathcal{L}^A$, i.e. $\psi : E \to \mathcal{L}^A$. We say that a joint probability P is compatible with an ALDAG (G, ψ) if P is Markov to G and additionally P respects all the edge labels given by ψ; that is, for each $(j, i) \in E$, X_i is $\psi(i, j)$ dependent from X_j.*

Henceforth, the labeling is represented via a coloring of the edges of the ALDAG. Note that BNs have an ALDAG representation where all edges have label 'total'. Standard features of BNs are also valid over ALDAGs: for instance, the already-mentioned d-separation criterion as well as fast probability propagation algorithms [77].

General structural learning algorithms for ALDAGs can be easily defined using the following routine: (i) learn a Staged Tree model $\mathcal{T}$ from data using for instance any of the algorithms in **stagedtrees**; (ii) transform the learned Staged Tree $\mathcal{T}$ into $G_{\mathcal{T}}$ as in Proposition 6; (iii) assign a label to each edge of $G_{\mathcal{T}}$ by checking the equalities in Equation (4.10).

An ALDAG can also be seen as a refinement of a DAG by the addition of edge labels indicating the class of dependence. Given a DAG G, the following steps implement such a refinement: (i) transform G into the Staged Tree $\mathcal{T}_G$; (ii) run a backward structural learning algorithm, which can only join stages together, using $\mathcal{T}_G$ as starting model (see for detail about computational implementation Chapter 5 and Carli et al. [20]); (iii) transform the resulting tree into $G_{\mathcal{T}_G}$ and apply the edge-labeling. The resulting ALDAG has an edge set which is either equal to or a subset of the edge set of the original DAG. Furthermore, the edge set is now labeled and denoting the class of dependence existing between any pair of random variables.

As an illustration of ALDAGs, consider the Staged Tree for the `Titanic` data in Figure 4.3 which, using Proposition 6, is transformed into the ALDAG in Figure 4.5.

Although the ALDAG does not carry all the information stored in the Staged Tree, which is quite complex to read, it intuitively describes the class of dependence existing among the random variables. The blue edges denote that there is partial dependence between `Class` and any other variable. The red edges denote that `Age` has a context dependence with `Gender` and `Survived`. Notice importantly that in the standard BN, which does not have the flexibility to embed non-symmetric independences, the variables `Age` and `Gender` were considered conditionally independent. Lastly, the green edge between `Gender` and `Survived` implies that there is only a local dependence between these two variables. Such extended forms of dependence better describe the fate of the Titanic passengers since, as already noticed, the BIC of the associated Staged Tree is smaller than the one of the best scoring BN.

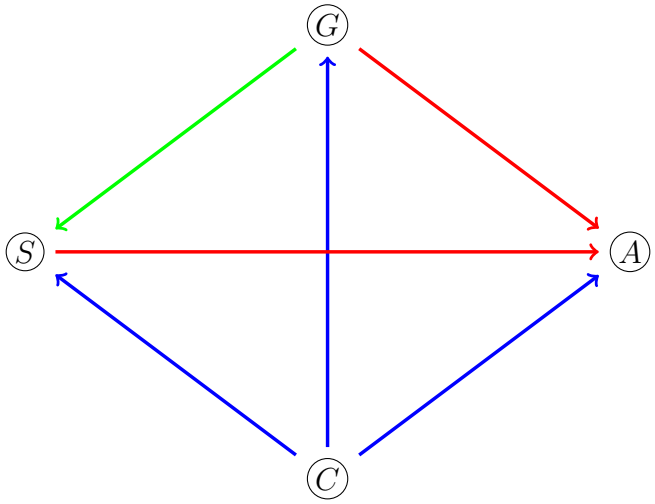


Figure 4.5: An ALDAG for the `Titanic` dataset constructed from the Staged Tree in Figure 4.3. The edge coloring is: red - context; blue - partial; green - local.

4.4 Conversion Algorithms

The implemented algorithms work with $\boldsymbol{X}$-compatible Stratified Staged Trees. Let $\boldsymbol{X}$ be a random vector taking values in $\mathbb{X} = \times_{i\in[p]}\mathbb{X}_i$ and assume there are total orders over each $\mathbb{X}_i$; then, $\mathbb{X}_{[i]}$ has an induced lexicographic total order. The node labeling (or coloring) κ can be represented by a vector of vectors of symbols $\boldsymbol{s} = (\boldsymbol{s}^1, \dots, \boldsymbol{s}^{p-1})$, where $\boldsymbol{s}^j \in \mathcal{C}^{|\mathbb{X}_{[j+1]}|}$ is the vector of coloring $(\kappa(v))_{v\in\mathbb{X}_{[j]}}$ and $\mathbb{X}_{[j+1]}$ is the set of nodes of $\mathcal{T}$ with depth j. Since the tree is stratified, the vector $\boldsymbol{s}$ represents the stages of the tree. Indeed, the original vector of stages $\boldsymbol{q} = (\boldsymbol{q}_1, \dots, \boldsymbol{q}_k)$ can be rewritten as a vector of vectors $\boldsymbol{s}$, where each $\boldsymbol{s}^j$, for $j = 1, \dots, p-1$, encloses all the components of $\boldsymbol{q}$ related to stages with depth j from the root.

Algorithm 7 describes the pseudo-code for the conversion algorithm that takes as input a DAG over $[p]$ and outputs an $\boldsymbol{X}$-compatible Stratified Staged Tree. Let assume that $1, \ldots, p$ is a topological ordering of the DAG nodes.

Algorithm 7: DAG to $\boldsymbol{X}$-compatible Stratified Staged Tree

Input: A DAG $G = ([p], F)$ such that $1, \ldots, p$ is a topological order according to G.

Output: The stage vector $\boldsymbol{s} = (\boldsymbol{s}^1, \ldots, \boldsymbol{s}^{p-1})$ encoding an $\mathbf{X}$-compatible Stratified Staged Tree $\mathcal{T}$ such that $\mathcal{M}_{\mathcal{T}} = \mathcal{M}_G$.

```
for i = 1 to p − 1 do
    s^i = [1]
    for j = 1 to i do
        if j ∈ pa(i + 1) then
            s^i = [s × X_j : s ∈ s^i]
        else
            s^i = [s × [1, . . . , 1] : s ∈ s^i]
return s.
```

Theorem 3 shows two important characteristics of the Stratified Staged Tree representation of a BN. It uses the notation $|pa(i)|$ for the cardinality of the parent set of X_i, $pa(i)[j]$ for the j-th element of the parent set of X_i and $|pa(i)[j]|$ for the dimension of the sample space of that element.

Theorem 3. *Let $G = ([p], E)$ be a DAG with vertex set $[p] = \{1, \ldots, p\}$ and edge set E. Let $\mathbb{X} = (X_i)_{i \in [p]}$ be a categorical random vector and O a topological order of variables. Then, the number of stages in the Stratified Staged Tree $\mathcal{T}_G = (V, F)$ built according to O and which embeds the same statistical model $\mathcal{M}_{\mathcal{T}_G}$ as the one associated to G, $\mathcal{M}_G$, can be calculated in a closed form:*

$$\#\ stages \ = \ \sum_{i=1}^{p} \mathbb{1}\{pa(i) = \emptyset\} \ + \ \sum_{i=1}^{p} \prod_{j=1}^{|pa(i)|} \Big|pa(i)[j]\Big|. \tag{4.11}$$

Also the number of degrees of freedom of $\mathcal{M}_{\mathcal{T}_G}$ can be computed:

$$\#\ df \ = \ \sum_{i=1}^{p} \mathbb{1}\{pa(i) = \emptyset\} \ \Big(|\mathbb{X}_i| \ - \ 1\Big) \ + \ \sum_{i=1}^{p} \Big(|\mathbb{X}_i| \ - \ 1\Big) \ \prod_{j=1}^{|pa(i)|} \Big|pa(i)[j]\Big|. \tag{4.12}$$

Algorithm 8 is an implementation of the conversion from a Stratified Staged Tree to the corresponding DAG. It records also additionally information on the type of dependence among variables that could be used to define the corresponding ALDAG.

In the pseudo-code for Algorithm 8 the following notation will be used: $\text{vec}(A)$ is the column-wise vectorization of a matrix A and $\text{mat}^{m,n}(\boldsymbol{a})$ is the column-wise (m,n)-matrix-filling such that $\text{vec}(\text{mat}(a)) = a$ and $\text{mat}^{m,n}(\text{vec}(A)) = A$. At last, a is a vector of symbols of length mn and A is a matrix of symbols of dimensions (m,n).

Algorithm 8: $\boldsymbol{X}$-compatible Stratified Staged Tree to ALDAG

Input: An $\mathbf{X}$-compatible Stratified Staged Tree T encoded with stage vector $\boldsymbol{s} = (\boldsymbol{s}^1, \ldots, \boldsymbol{s}^{p-1})$.

Output: A minimal DAG $G = ([p], F)$ such that $\mathcal{M}_{\mathcal{T}} \subseteq \mathcal{M}_G$ and a labeling of its edges ψ defining an ALDAG.

```
G = (V, E = ∅)
for i = 1 to p − 1 do
    a = s^i
    # compute the number of distinct symbols in a
    N = length(a) = ∏_{j∈[i]} |X_j|
    c = |{a_k : k ∈ [N]}|
    for j = i to 1 do
        m = |X_j|
        n = N/m
        A = mat^{m,n}(a)
        c_k = |{A_{u,k} : u ∈ [m]}|, k ∈ [n]
        r_u = |{A_{u,k} : k ∈ [n]}|, u ∈ [m]
        if max{c_k : k ∈ [n]} > 1 then
            # j is a parent of i + 1
            E = E ∪ {(j, i + 1)}
            a = vec(A^t)
            # check now the type of dependence
            if min{c_k : k ∈ [n]} = m then
                r = ∑_{i∈[m]} |{A_{u,k} : k ∈ [n]}|
                d = |{A_{u,k} : u ∈ [m], k ∈ [n]}|
                if r ≠ d then
                    ψ(j, i + 1) = local
            if min{c_k : k ∈ [n]} = 1 then
                ψ(j, i + 1) = context
                if ∃ k ∈ [n] s.t. 2 < c_k < m then
                    ψ(j, i + 1) = partial
            else
                ψ(j, i + 1) = partial
```

4.5 Applications

Two additional real-world applications are analyzed to further illustrate the capabilities of Staged Trees and ALDAGs.

4.5.1 Cancer-Associated Muscle Wasting

First a subset of the dataset of Eisner et al. [41] including metabolomic information of 77 individuals is studied: 47 of them suffering of cachexia, whilst the remaining do not. Cachexia is a metabolic syndrome characterized by loss of muscle with or without loss of fat mass. Although the study of Eisner et al. [41] included 71 different metabolics which could possibly distinguish individuals who suffer of Cachexia from those who do not, for the illustrative purposes the focus is on only six of them: Adipate (A), Betaine (B), Fumarate (F), Glucose (GC), Glutamine (GM) and Valine (V). These metabolics are measured in a continuous scale and have been recently investigated in the context of Gaussian BNs [52]. The variables are discretized into three levels using the equal frequency method (see e.g. Ropero et al. [100]). A BN over these variables together with the binary variable Muscle Loss (ML) is learnt using an hill-climbing algorithm and reported in Figure 4.6 (left). The learned BN is quite sparse: it only includes six edges and has 35 free parameters.

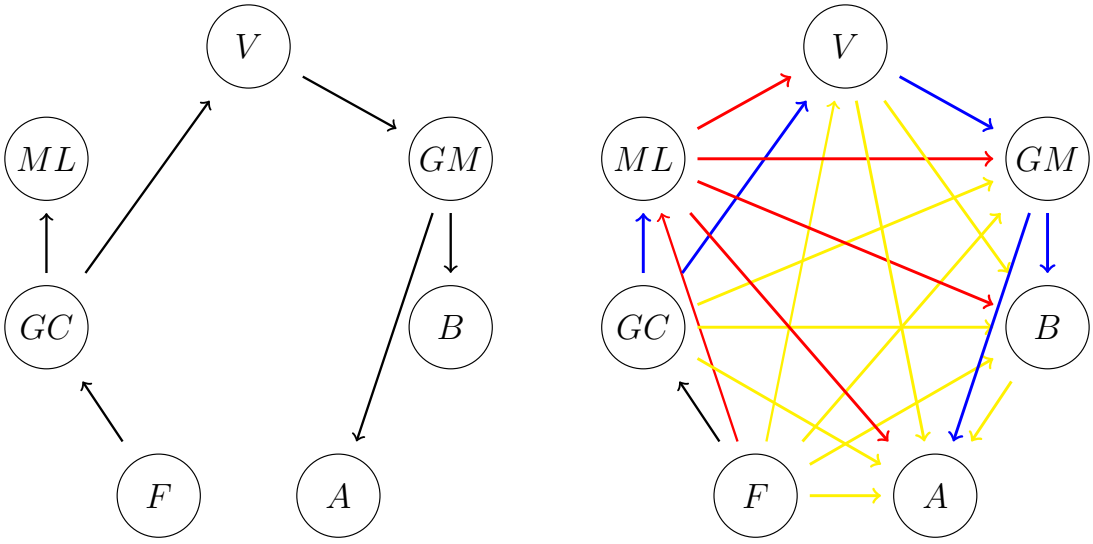


Figure 4.6: Learned BN (left) and ALDAG (right) for the muscle wasting application. The edge coloring is: red - context; blue - partial; yellow - context/partial; black - total.

The learned BN is then used as starting point of a hill-climbing structural search over the space of Stratified Staged Trees. The final Staged Tree model, with a

BIC score of 793.36, provides a big improvement compared to the starting BN whose BIC is 963.97, without increasing the complexity since the number of free parameters has not changed. The associated tree has 1458 leaves and therefore its staging can not be visually investigated. On the other hand, the ALDAG gives a concise summary of the underlying dependence structure and is reported in Figure 4.6 (right). First, one can notice that the resulting graph is complete and that there are various types of dependence between random variables. Only the arc between F and GC is total, meaning that it encodes the same dependence as in the BN. All other edges are labeled with non-symmetric dependences. Therefore the presence of an edge in the BN is too generic and does not flexibly represent local equalities in the probability tables. Similarly, the absence of an edge is actually too restrictive since the equalities associated to symmetric conditional independence do not hold over the corresponding whole sample spaces. The latter suggests that the BN is not a good representation of this data; the chosen BN is selected only because it is the best of a poorly fitting collection.

4.5.2 Body Fat and Body Measurements

Next the body fat data first studied in Johnson [66] reporting percentage of Body Fat (BF), Age (A), Weight (W), Height (H), and ten body-circumference measurements for 252 men is analyzed. As in Lauritzen and Zwiernik [79], 11 individuals are removed from the study for various problems associated to their observations. The variables are again discretized into two levels using the equal frequency method.

A BN is learned over this data using hill-climbing and reported in Figure 4.7 by assuming all edges to represent total dependence. In this case a refinement of this BN using a backward hill-climbing algorithm which can only join stages together (as discussed in Section 4.3.2) is considered. The resulting Staged Tree, with a BIC score of 3419.11, better represents the data than the discrete BN which has a BIC of 3468.32. The associated ALDAG is in Figure 4.7. The same pairs of nodes are connected by edges, due to the algorithm chosen to learn the Staged Tree, but a variety of dependence classes now exist. For some random variables the BN represented the actual relationship between them since there are quite a few edges of class "total" in the ALDAG. However, it can be also seen that there are multiple edges of class "context" and "local". Notice that since the variables are binary there can not be partial dependences.

One interesting conclusion that can be drawn from this network is that, conditionally on the Abdomen (AB) measurement, Body Fat is independent of all other measurements. One can be interested in assessing if this conclusion is due to the

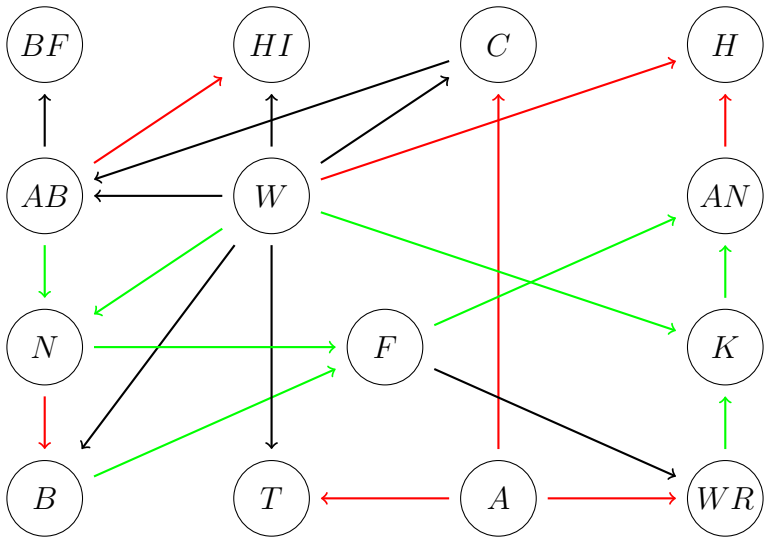


Figure 4.7: Learned BN (without considering coloring) and ALDAG (with coloring) for the body fat application.

fact that structural learning of BNs is restricted to symmetric relationships. For this reason, starting from the learned Staged Tree, an hill-climbing algorithm is run, which can both join and split stages, only over the vertices associated to the variable BF. The resulting Staged Tree has a better score, BIC = 3403.01, and, more importantly, the associated ALDAG (not reported here) is such that the parents of Body Fat are now Age, Weight and measurements of Abdomen, Thigh and Chest: many of these variables were conditionally independent of Body Fat in the original BN.

4.6 Conclusion

Staged Trees are a flexible class of models that can represent highly non-symmetric relationships. This richness has the drawback that independences are often difficult to assess and visualize intuitively through its graph. In this chapter, methods that summarize both the symmetric and non-symmetric relationships learned from data via structural learning by transforming the tree into a DAG are outlined. As a result, a novel class of graphs which extend DAGs by labeling their edges is introduced. The data applications showed the superior fit to data of such models as well as the information they can provide in real domains.

The new DAG edge labeling is based on the identification of the class of dependence. A different possibility would be to define a dependence numerical index between any two variables which measures how different their relationship is from

total dependence/independence. By learning a Staged Tree from data, one could then label the edges of a BN with such indexes. The definition of such models is the focus of current research.

This work further provides a first criterion to read any symmetric conditional independence from a Staged Tree. Algorithms to assess if generic non-symmetric conditional independence statements hold still need to be developed. Here an intermediate solution to this problem has been provided by characterizing if a non-symmetric independence exists or not. The plan is to provide a conclusive solution to non-symmetric independence queries in future work.

The chapter exhibits the relationship between Staged Trees and Bayesian Networks. In particular, the first two sections give a quick review of interesting results from the state of the art on BNs, Staged Trees and symmetric conditional independence. In Section 4.3 non-symmetric independences are introduced: total, context-specific, partial and local. These can not be represented through Bayesian Networks, but they can in a succinctly way and in a unique graph with a Staged Tree. Furthermore, a new class of DAGs, called Asymmetry-labeled DAGs (ALDAGs) [121], is proposed: this is a minimal DAG such that the statistical model embedded in the given Staged Tree is contained in the one associated to that DAG. Also two conversion algorithms are presented with the pseudo-code: the first starts from a BN and a topological order of variables and returns the corresponding Staged Tree structure embedding the same statistical model; the second takes in input a Staged Tree and returns the corresponding ALDAG. In the last part of the chapter two small applications are carried out in order to show the strength and usefulness of this innovative DAG.

Chapter 5

The R Package stagedtrees

In this chapter the *R* package **stagedtrees** [120] is presented. The chapter follows [20] closely. **stagedtrees** is freely available from the Comprehensive R Archive Network (CRAN) at *https://CRAN.R-project.org/package=stagedtrees* and includes several algorithms for learning the structure of Staged Trees and CEGs from data. Score-based and clustering-based algorithms are implemented, as well as various functionalities to provide inferential, visualization, descriptive and summary statistics tools for such models and about their graph structure. There is another *R* package called **ceg** [25] for the implementation of Staged Trees, which, however, includes a single learning algorithm (*Agglomerative Hierarchical Clustering*, [46]) and few functionalities to produce graphs and/or inference starting from an estimated model. The project of **stagedtrees** was born from the need to build new estimation algorithms for Staged Trees and Chain Event Graphs capable of making applicable in real contexts all the theoretical knowledge developed in the last decade regarding these Graphical Models. The capabilities of **stagedtrees** are illustrated throughout this chapter using mainly two datasets both included in the package, as done similarly in Carli et al. [20] where this package is introduced. Currently, the package works for $\boldsymbol{X}$-compatible Staged Trees.

Section 5.1 explains how to initialize a Staged Tree starting from a dataset and how to build the corresponding CEG. In Section 5.2 the implemented estimation algorithms are listed, while Section 5.3 presents the main functions for producing graphs, outputs and summaries of the models estimated using the algorithms in Section 5.2. More details are available in the *R* manual and the *R* help page of **stagedtrees**. In Section 5.4 a detailed example of the usage of the package is conducted on the Titanic dataset [33]. In Section 5.5 a simulation study is carried out by comparing the performance of different algorithms implemented in **stagedtrees** on a number of datasets, in order to verify accuracy and robustness to hyper-parameters

that has to be set a priori. Section 5.6 carried on the study performed in Chapter 1 on the pediatric dentistry dataset using UGMs, with the usage of Staged Trees and ALDAGs.

5.1 Initialize Staged Trees and Create CEGs

The main object class implemented in the *R* package **stagedtrees** is `sevt`, which represents a Staged Tree model. Given a dataset, either in `data.frame`, `table` or `list` format, a Staged Tree which is compatible with the variables in the dataset can be constructed using the functions `full` or `indep`. The function `full` returns a `sevt` object which defines in *R* a Staged Tree where each vertex is in a different stage. It corresponds to the saturated statistical model, where the number of free parameters equals the number of edges minus the number of non leaf vertices, equivalently the number of leaves minus one. Conversely, `indep` returns a tree where all vertices in the same stratum are in the same stage, corresponding to a model where all variables are marginally independent of each other.

Worth-mentioning arguments of these two functions are:

- `order`, which fixes the order of the variables in the tree. It can be used also to take a subset of the collected variables in the dataset (`data`) given in input: if an order not considering all the variables in `data` is provided, then the estimated Staged Tree will have strata related only to that subset; see e.g. Section 5.4.4;

- `join_unobserved` which enables vertices of the tree where no observations are observed according to given dataset to be collapsed (by default set to `TRUE`). If this is `TRUE`, `name_unobserved` can be used to specified the name to assign to these unobserved situations (by default `"UNOBSERVED"`); see e.g. Section 7.1;

- `lambda`, which implements a Laplace smoothing to address possible zero counts in the case where `join_unobserved` is set to `FALSE`. If one wants to use a Bayesian approach, it can be used also as a uniform prior distribution over the tree, even if `join_unobserved` is `TRUE`.

Furthermore, a `bn.fit` object created with the **bnlearn** package could be turned into a `sevt` object modelling the same conditional independences with `as_sevt`. A Staged Tree can be converted into a CEG model using the `ceg` function. The usual `print`, `summary` and `plot` functions provide basic information, more detailed information and the graphical representation of the model, respectively.

5.2 Structure Learning Algorithms

stagedtrees implements a variety of structure learning algorithms. These can be grouped into three macro-categories:

- score-based algorithms using various heuristics to maximize a score function. The default value of `score` is the negative BIC, but any other can be defined by the user. Four different score optimizers are developed in the package:
  - an *hill-climbing* score optimization `stages_hc` which, for each stratum, at each iteration, searches for the stage that has to move either to a different or a new stage maximizing a score, until no score improvement is found. The theoretical formulations are shown in detail in Section 3.4, where the function that has to be optimize at line 11 of Algorithm 2 is by default the BIC index;
  - a *backward hill-climbing* `stages_bhc` which searches the joining of two stages maximizing a score until no score improvement is found;
  - a *fast backward hill-climbing* `stages_fbch` which joins two stages whenever the joining improves the score, until no improvement is possible or the maximum number of iterations `max_iter` is reached;
  - a *random backward hill-climbing* `stages_bhcr` which at each iteration randomly selects a stratum and two stages and joins the stages if the score is increased. The procedure is repeated until the number of iterations reaches `max_iter`.
- Distance-based algorithm using different distances or divergences for comparing discrete probability distributions of stages:
  - *backward joining* of stages `stages_bj` which iteratively joins stages if the distance between their floret probabilities is less then a given threshold value `thr`. The distance can be chosen with the `distance` argument: the default is the symmetrized Kullback-Leibler divergence `"kullback"`, but also the other seven described in details in Section 3.5 are implemented, i.e. Manhattan (`"manhattan"`), Euclidean (`"euclidean"`), Renyi Divergence (`"reny"`), Total Variation (`"totvar"`), Hellinger (`"hellinger"`), Bhattacharyya (`"bhatt"`) and the Chan-Darwiche (`"chandarw"`). Note that the higher `thr`, the more vertices will be agglomerated in the same stage using any distance or divergence, converging asymptotically to the independence model.

- Clustering-based algorithms, where stages are identified according to the clusterization of the discrete probability distributions of florets:

  - *hierarchical clustering* of stages `stages_hclust` which estimates a user-defined number `k` of stages in each stratum. The function inherits all arguments of the standard `hclust` function from the **stats** package. For instance, the `distance` used for the clustering can be choosen from the ones implemented in `dist` of **stats**, with the addition of `"totvar"` and `"hellinger"`;
  - *clustering* of stages using the *kmeans* algorithm `stages_kmeans`, creating a user-defined number `k` of stages in each stratum. The function inherits all arguments of the standard `kmeans` function from the **stats** package.

The starting model of any structure learning algorithm has to be a Staged Tree which, for instance, may be constructed directly from a dataset using `full` or `indep`. The function `stages_hc` embeds both split and join of stages, that is a combination of forward and backward procedure as done similarly in stepwise regression with an approach based on bidirectional elimination [59]. All other score, distance and clustering based algorithms have implemented only a joining step of previously defined stages, i.e. a backward approach. For this reason, it is appropriate that the independence model `indep` is used as starting model only for the algorithm `stages_hc`. Indeed, any other structure learning algorithm executed starting from the independence model will end after zero iterations, since no joining step can be performed. Instead, the stage structure related to the dependence model `full` can be used as starting point for any algorithm developed in the package, including `stages_hc`. Intuitively, `stages_hc` is the slowest algorithm due to its more detailed model search. The computational costs of each algorithm in **stagedtrees** will be discussed in Section 7.2.

Different structure learning algorithms can be easily combined since the starting model for any algorithm could be also an already estimated model with another structure learning algorithm. Furthermore, model search can be performed only over a subset of strata specified by the argument `scope`; stages in other strata will not be touched. With the option `ignore` one can specifies a vector of stages that will be ignored and left untouched by the model search; by default it is the set of the unobserved stages stored in `name_unobserved`, if it exists. This can be useful both to left untouched some probability distributions that one knows a priori that does not want to merge together with others and to reduce the computational burden. The option `trace` is used to print via message amount of info about the composition

of stages during the iterations of the algorithm.

Even if not properly a structure learning algorithm, it is important to re-emphasize the usefulness of isolating in separate stages those situations in which no observation arrives starting from the root of the Staged Tree. They are the analogue of empty cells (zero counts) in contingency tables. As mentioned above, this can be done with the option `join_unobserved` of functions `full` and `indep`. Moreover, an ad-hoc function `join_unobserved` for doing this is implemented in the package. Indeed, not-observed situations do not contribute to the log-likelihood of the model and thus algorithms based on penalized log-likelihood scores, as AIC and BIC indices, will always join unobserved situations with other stages in a randomly manner. This is a coherent behaviour from a score-optimization point of view, which implies a reduction of the number of parameters but, nevertheless, the user would probably prefer to isolate unobserved situations for interpretation's sake.

All above algorithms work with a fixed ordering of the variables, which can be set with the argument `order`. For learning a Staged Tree from data with an optimal variable ordering, the function `order_dynamic` can used, which implements the dynamic programming ordering algorithm of Silander and Leong [105] and Cowell et al. [28]. The search of the optimal order can be coupled with any of the model search algorithms mentioned above which can be set with the argument `alg`.

5.3 Main Functions for Inference on Staged Trees

stagedtrees provides an array of functions to explore and perform inference over a learned model, among which the main ones are the following:

- `stndnaming` renames stages in a standard manner. It assigns them increasing numbers from 1 to the number of different stages, for each stratum in the tree. Its main usage is for greater visual clarity for print, plot and summary of the estimated model;
- `subtree` enables for the construction of a subtree having as root any vertex of the tree. This can be achieved specifying the `path` starting from the root and ending at that vertex;
- `summary` returns for each stratum of the tree all the estimated stages, the number of paths and observations starting from the root that arrives to each stage and their corresponding probability distributions;
- `compare_stages` compares the stage structure of two $\boldsymbol{X}$-compatible Staged Trees with the same order of variables, given in input as `object1` and `object2`.

It returns also a plot where nodes in different stages are colored in red if `plot` is set to `TRUE`. Three methods (`method`) are available:

  - `"naive"` first applies `stndnaming` to both objects and then simply compares the resulting stage names;
  - `"hamming"` uses the `hamming_stages` function that finds a minimal subset of nodes which stages must be changed to obtain equal stage structures;
  - `"stages"` uses the `diff_stages` function that checks whether exactly the same stage structure is present in both models.

  Intuitively, the second method finds less differences between stage structures, while the third being more restrictive leads to the identification of many diversities. Setting the option `return_tree` to `TRUE` will return the stages differences obtained according to the selected method as a list of numerical vectors with same lengths and structure as the list `stages` in `object1` and `object2`, where values are 1 if the corresponding vertex has different (with respect to `method`) associated stage, and 0 otherwise. For more details see the *R* help page of `compare_stages`;

- `sample_from` generates `nsim` observations according to the probability distributions encoded in the Staged Tree given in input via `object`. This can be used to perform simulation studies over a learned model, exploiting also the option `seed` for replicable analysis;

- `get_stage` retrieves the stage associated to a given `path` from the root. To be used in combination with `summary` and/or `plot` for a more helpful use;

- `get_path` gives all the paths that starting from the root arrive to a given `stage` for a given stratum (`var`);

- `prob` computes the probability (or its logarithm if `log` = TRUE) of any event of interest (`x`) according to the probability distributions encoded in the Staged Tree given in input through `object`. It can be used to derive all atomic probabilities; see theory in Section 3.2;

- `confint` provides confidence intervals for Staged Tree parameters. By default, it computes the confidence intervals for the probability parameters for all the stages in the Staged Tree given in input; intervals can be computed only for a single variable by specifying it in `parm`. Five methods are available: `wald`, `waldcc`, `wilson`, `goodman` and `quesenberry-hurst`;

- `LR_test` performs a likelihood ratio test to assess if two nested Staged Trees are significantly different from each other.

- `plot` is a dependencies-free plotting function for Staged Trees and Chain Event Graphs; users can specify stage-coloring, node and edge size and labels appearance for the graphical representation of the model given in input as `x`. An useful argument of the function especially in contexts with many variables is `limit`, which permits to specify the number of stratum to be plotted; by default 10. The user can also choose if some situations have not to be plotted by setting `ignore` to the name of those stages that must not be displayed (default `ignore = x$name_unobserved`);

- `barplot` generates barplots to visualize the floret probabilities for each stage of a specified variable (`var`). Many functionalities of the original `barplot` function of **graphics** package are inherited;

- `predict` forecasts the most probable level for the class variable (`class`), according to an estimated Staged Tree (`object`), given all the other variables in the model. If specified `prob = TRUE`, it returns the probabilities related to each class levels for each observation in `newdata`.

5.4 Usage of stagedtrees

The well-known Titanic dataset [33], which provides information on the fate of the Titanic passengers and available from the **datasets** package bundled in *R*, is used to exemplify the usage of **stagedtrees**. **stagedtrees** and its dependencies (the *R* packages **graphics** and **stats**) are available from CRAN, as is the package **bnlearn**.

5.4.1 Learning the Stage Structure from a Dataset

The `Titanic` dataset can be loaded into a `table` of the same name with the call to `data`.

```
R> data("Titanic")
R> str(Titanic)
```

```
 'table' num [1:4, 1:2, 1:2, 1:2] 0 0 35 0 0 0 17 0 118 154 ...
 - attr(*, "dimnames")=List of 4
  ..$ Class    : chr [1:4] "1st" "2nd" "3rd" "Crew"
  ..$ Sex      : chr [1:2] "Male" "Female"
```

```
..$ Age      : chr [1:2] "Child" "Adult"
..$ Survived: chr [1:2] "No" "Yes"
```

`Titanic` includes four categorical variables: `Sex`, `Age` and `Survived` are binary and `Class` has four levels. Initial Staged Trees where all vertices within a stratum are either in the same or in different stages can be constructed using the `indep` and `full` functions, respectively. For both models, for each stratum of the tree, situations where no observations are observed in `Titanic` are isolated in an own stage called `"na"`.

```
R> library(stagedtrees)
R> m.full <- full(Titanic, name_unobserved = "na")
R> m.indep <- indep(Titanic, name_unobserved = "na")
R> m.full

Staged event tree (fitted)
Class[4] -> Sex[2] -> Age[2] -> Survived[2]
'log Lik.' -5151.517 (df=30)

R> m.indep

Staged event tree (fitted)
Class[4] -> Sex[2] -> Age[2] -> Survived[2]
'log Lik.' -5773.349 (df=7)
```

The printing of `m.full` and `m.indep` gives information about the order of the variables in the tree, the value of the log-likelihood function and the number of free parameters, whilst `plot` displays the Stratified Staged Tree with stages colored within each stratum as shown in Figure 5.1. The plot of `m.full` is depicted using the `Dynamic` palette from the **colorspace** package [129], since the default palette has only 8 colors and thus stages for the last variable would be impossible to graphically distinguish.

```
R> library(colorspace)
R> plot(m.full, col = function(s) qualitative_hcl(length(s),
        "Dynamic"))

R> plot(m.indep)
```

Notice that there are no crew members, either male or female, who are children and this is correctly reflected in the trees in Figure 5.1 since the subtree associated to such events are collapsed (by default the argument `join_unobserved` is set to

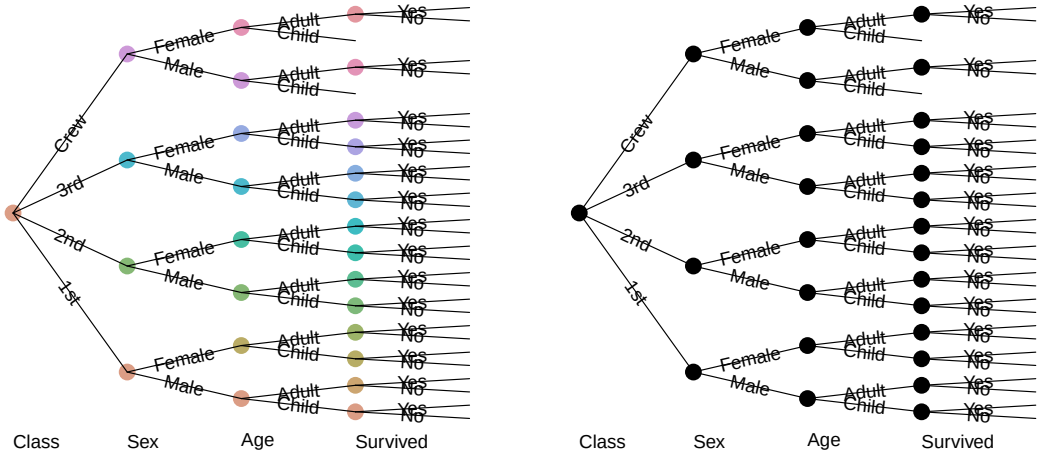


Figure 5.1: Left: Staged Tree `m.full` where all vertices in the same stratum are in a different stage, except for unobserved situations: there are 28 different stages considering also the `"na"` stage in the stratum related to `Survived`. Colors in different strata can be equal. Right: Staged Tree `m.indep` where all vertices in the same stratum are in the same stage: there are 5 different stages. The labels at the bottom denote the variable associated to a stratum.

`TRUE`). The name of these collapsed vertices was set to `"na"` with the argument `name_unobserved` as seen before.

Using the Staged Tree `m.full` or `m.indep` as starting point, structural learning algorithms can be used to infer the staging structure from the data. The hill-climbing algorithm implemented in `stages_hc` can receive in input both `m.full` and `m.indep`, whilst backward score-based (`stages_bhc`, `stages_fbhc` and `stages_bhcr`), distance-based (`stages_bj`) and clustering algorithms (`stages_hclust` and `stages_kmeans`) start from the `m.full` tree. For illustration purposes, the `stages_hc` function is used with the `m.indep` tree, whilst `stages_bj` is used with `m.full`.

```
R> mod1 <- stages_hc(m.indep)
R> mod2 <- stages_bj(m.full, thr = 0.1)
```

The `stages_hc` function has the negative BIC as a default score to be maximize, while the default distance for `stages_bj` is the symmetrized Kullback-Leibler divergence, with threshold 0.1 in this example. The learned `mod1` and `mod2` are plotted in Figure 5.2. Both Staged Trees suggest that the variables are dependent in a non-symmetric fashion and thus suggest context-specific independences. The stage structures of the two trees are quite different and may be affected by the choice of

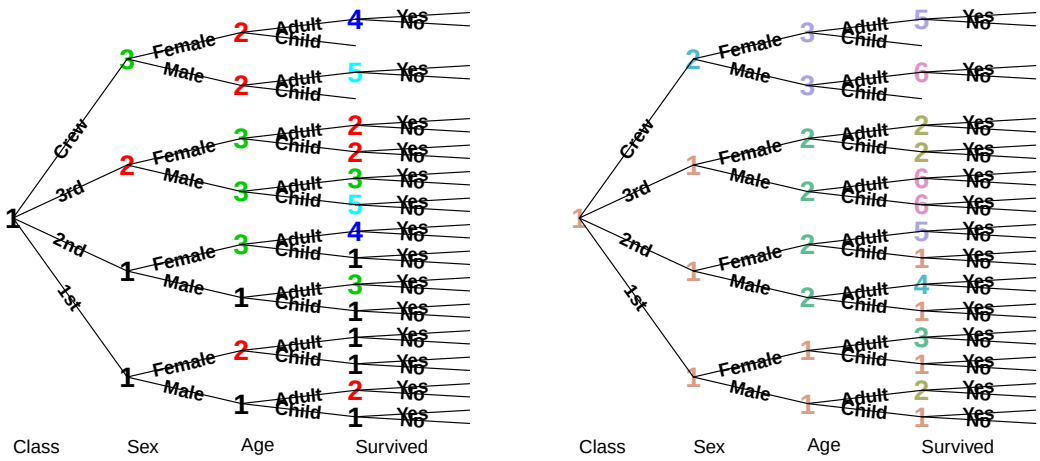


Figure 5.2: Staged Trees `mod1` (left) and `mod2` (right) learned using the `stages_hc` and the `stages_bj` algorithms, respectively.

threshold in `mod2`. However, they also share some common features: for instance, both state that the distribution of Male/Female is the same for passengers in the first and second class.

Since all structural learning algorithms take as input a Staged Tree, it is possible to refine a learned model: for instance the model `mod2` learned using a backward algorithm may be refined using a standard hill-climbing algorithm.

```
R> mod3 <- stndnaming(stages_hc(mod2))
```

```
R> plot(mod3, ignore = NULL, cex_label_nodes = 1.5,
        cex_nodes = 0, font = 2)
```

The resulting Staged Tree is reported in Figure 5.3. For illustrative purpose, the full tree (by setting `ignore = NULL`) and the numbering of the stages after renaming them with the function `stndnaming` are reported. The two Staged Tree structures in `mod1` and `mod3` are compared through the `compare_stages` function, whose output highlights in red the nodes in different stages. Different methods can be used to compare two Staged Tree structures, here the `"stages"` method is used: it checks if the same exact stages are present in both models.

```
R> compare_stages(mod1, mod3, method = "stages", plot = TRUE)
```

```
[1] FALSE
```

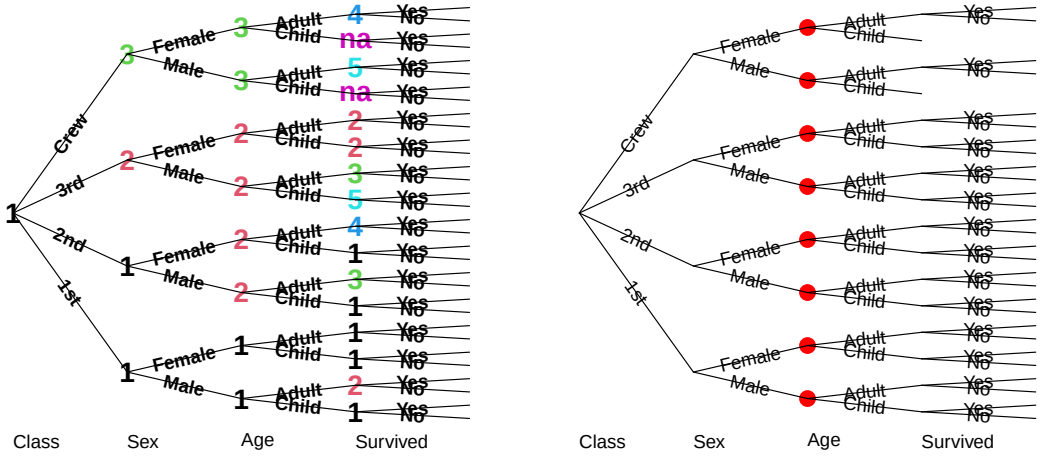


Figure 5.3: Staged Event Tree `mod3` (left) and output of the `compare_stages` function between models `mod1` and `mod3` (right). Vertices depicted by a red dot in the right plot correspond to vertices for which the staging structure differs.

Figure 5.3 shows that the two models have the same stage structure over the `Sex` and `Survived` variables, but they completely differ over `Age`.

The model selection criteria `AIC` and `BIC` can be used to choose the best fitting model.

```
R> cbind(AIC(mod1, mod2, mod3),
         BIC = BIC(mod1, mod2, mod3)$BIC)
```

```
      df      AIC      BIC
mod1 15 10364.49 10449.94
mod2 15 10390.37 10475.82
mod3 15 10365.02 10450.47
```

According to both criteria, `mod1` is the best fitting model among those tried. It is not surprising that `mod2` obtains the worst BIC scores since it was estimated with the `stages_bj` function that joins stages following a distance based heuristic and thus not the minimization of the BIC score.

5.4.2 Bayesian Networks as Staged Trees

stagedtrees has the capability of translating a BN learned with the **bnlearn** package into a Staged Tree. To use **bnlearn** the dataset `Titanic` needs to be converted into a data frame.

```
R> titanic.df <- as.data.frame(Titanic)
R> titanic.df <- titanic.df[rep(row.names(titanic.df),
                                titanic.df$Freq), 1:4]
```

The `hc` function of **bnlearn** can be used to learn the graph of the BN reported in Figure 5.4 left.

```
R> library(bnlearn)
R> mod.bn <- bnlearn::hc(titanic.df)
R> plot(mod.bn)
```

`bn.fit` returns an object of class `bn.fit` which can be turned into an object of class `sevt` using the `as_sevt` function. `sevt_fit` is used to compute also the stage probability distributions. Below the *R* code.

```
R> mod.bn <- bn.fit(mod.bn, titanic.df)
R> bn.tree <- sevt_fit(as_sevt(mod.bn),
                       data = titanic.df, lambda = 0)
R> plot(bn.tree)
```

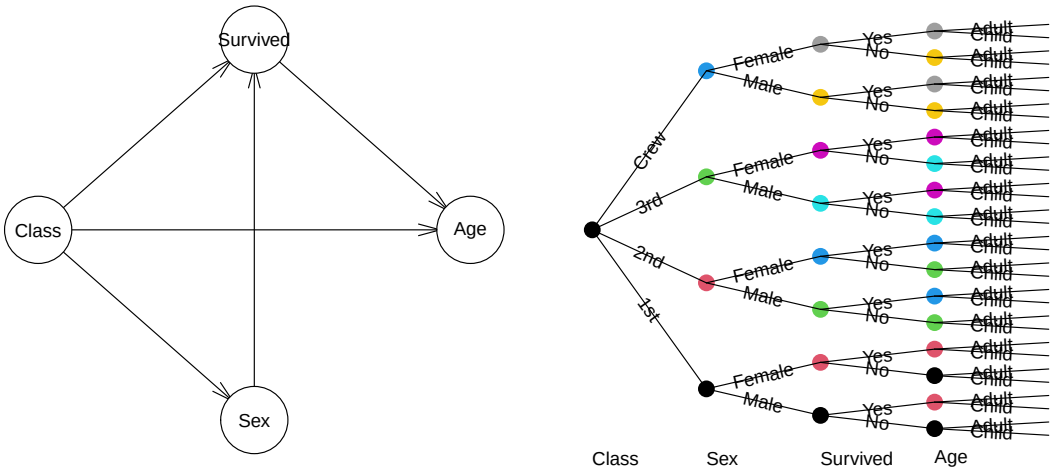


Figure 5.4: Left: BN model learned using the `hc` function of **bnlearn**. Right: associated Staged Event Tree.

The learned BN embeds only one conditional independence statement: `Age` and `Sex` are conditionally independent given `Class` and `Survived`. This is represented in Figure 5.4 right by the highly symmetric staging structure over the variable `Age`. Notice that this tree, since it is representing the associated BN, does not collapses

subtrees where there are no associated observations in the dataset. However, this can be achieved by using the function `join_unobserved`. It is also worth noticing that the order of the variables chosen by **bnlearn** to build the corresponding Stratified Staged Tree is different to the one used for `mod1`, `mod2` and `mod3`. Therefore, it is not possible to use `compare_stages` to compare `bn.tree` with `mod1`, `mod2` or `mod3`.

The Staged Tree corresponding to the associated learned BN could be used as the starting point of any structure learning algorithms (see also Barclay et al. [4]). As an illustration, the `stages_hclust` function is used specifying that in each stratum there should be 2 stages.

```
R> mod4 <- stages_hclust(bn.tree, k = 2)
R> plot(mod4, col = function(x) c("red3", "blue3"))
```

The Staged Tree `mod4`, which is displayed in Figure 5.5 left, is coalesced into the more compact CEG representation shown in Figure 5.5 right. This can be achieved by the `ceg` function which takes as input mod4 and the `plot` method for `ceg` objects, which requires the recommended **igraph** package [29].

```
R> library(igraph)
R> plot(ceg(mod4), col = function(x) c("red3", "blue3"))
```

For `mod4`, vertices in the last stratum are coalesced into two positions, whilst vertices in the penultimate stratum are coalesced into four positions, thus reducing the overall number of vertices of the underlying graphical representation. We refer to the Conclusions for a discussion of the CEG plotting capabilities.

5.4.3 Querying the Model

Having chosen a model, the focus is on using it to perform inference and understanding the relationship among the problem variables. Here `mod1` is chosen since it was the best scoring model according to AIC and BIC.

The dataset in this simple example only includes four variables and its Staged Tree can be easily investigated by eye. For more complex applications the function `subtree` is useful as it enables the construction of a subtree having as root any vertex of the tree. The subtree can be achieved specifying the path starting from the root and ending at that vertex. For instance, it is possible to construct the subtree for the crew members of the Titanic.

```
R> subtree.crew <- subtree(mod1, c(Class = "Crew"))
R> subtree.crew
```

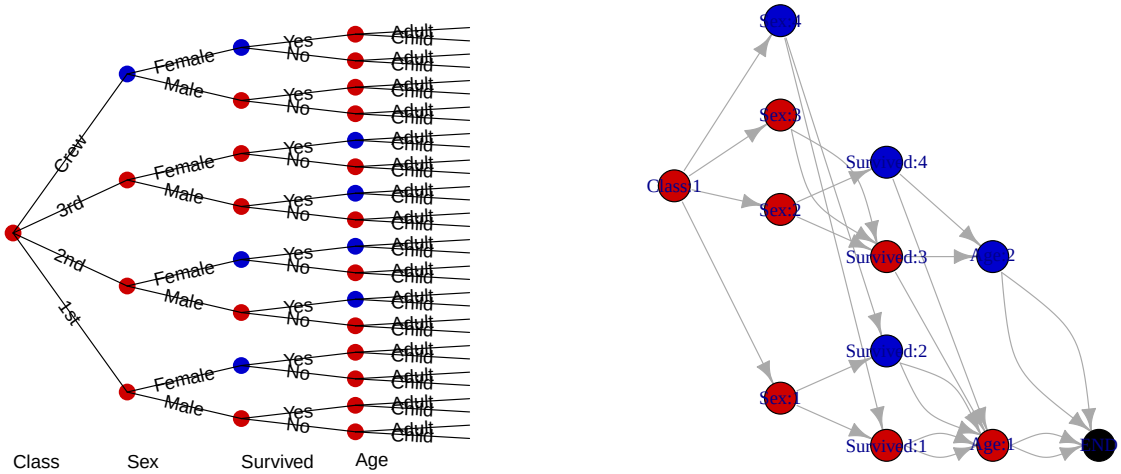


Figure 5.5: Staged Event Tree `mod4` (left) and its corresponding CEG representation (right).

```
Staged event tree (fitted)
Sex[2] -> Age[2] -> Survived[2]
```

```
R> plot(subtree.crew)
```

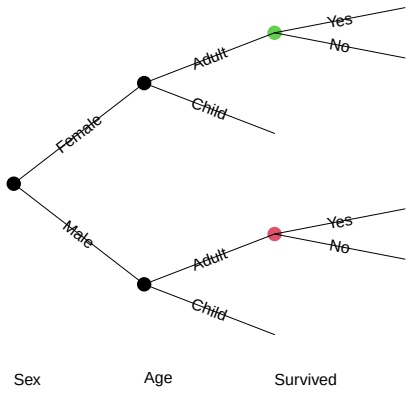


Figure 5.6: Subtree of the Staged Tree `mod1` representing `Sex`, `Age` and `Survived` of `Crew` passengers only.

`subtree.crew` is still formally a Staged Tree over three variables. It is displayed in Figure 5.6 and clearly its stage structure coincides with the one in the upper half of `mod1` reported on the left of Figure 5.2. The colors of the stages are different

in the two plots since two different colors palettes have been used, but the stage structure is the same.

A detailed model summary of `mod1` can be obtained by the `summary` function. For ease of exposition the stages are also renamed with `stndnaming`.

```
R> mod1 <- stndnaming(mod1)
R> summary(mod1)

Call:
stages_hc(m.indep)
lambda:  0
Stages:
  Variable:  Class
 stage  npaths  sample.size       1st       2nd       3rd      Crew
     1       0         2201 0.1476602 0.1294866 0.3207633 0.40209
  ------------
  Variable:  Sex
 stage  npaths  sample.size      Male    Female
     1       2          610 0.5885246 0.4114754
     2       1          706 0.7223796 0.2776204
     3       1          885 0.9740113 0.0259887
  ------------
  Variable:  Age
 stage  npaths  sample.size       Child     Adult
     1       2          359 0.0445682451 0.9554318
     2       3         1030 0.0009708738 0.9990291
     3       3          812 0.1133004926 0.8866995
  ------------
  Variable:  Survived
 stage  npaths  sample.size          No       Yes
     1       5          174 0.02298851 0.9770115
     2       3          371 0.60377358 0.3962264
     3       2          630 0.85873016 0.1412698
     4       2          116 0.13793103 0.8620690
     5       2          910 0.77472527 0.2252747
    na       2            0          NA        NA
  ------------
```

The output of `summary` together with the function `get_stage` allow to determine

the estimated survival probabilities of the passengers of the Titanic. Stage 1 for `Survived` has the highest survival probability and it includes children from the first two classes and adult women from the first class, as shown by the following code.

```
R> get_path(mod1, var = "Survived", stage = "1")
```

```
  Class    Sex   Age
1   1st   Male Child
3   1st Female Child
4   1st Female Adult
5   2nd   Male Child
7   2nd Female Child
```

Stage 3 has the lowest survival probability and includes adult males of second and third class. These considerations about stage 1 and 3 are in agreement also with what is displayed in Figure 5.7.

```
R> get_path(mod1, var = "Survived", stage = "3")
```

```
   Class  Sex   Age
6    2nd Male Adult
10   3rd Male Adult
```

The package **stagedtrees** also includes the function `get_stage` to get the stage associated to a given path.

```
R> get_stage(mod1, path = c("Crew", "Female"))
```

```
[1] "2"
```

The function `prob` allows for the computation of the probability of any event of interest.

```
R> prob(mod1, c(Survived = "Yes"))
```

```
[1] 0.3236376
```

```
R> prob(mod1, c(Survived = "Yes"),
        conditional_on = c(Age = "Adult"))
```

```
[1] 0.3165252
```

```
R> prob(mod1, c(Survived = "Yes"),
        conditional_on = c(Age = "Child"))
```

```
[1] 0.4584954
```

For instance, the probability of survival of any passenger is 0.3236, but this decreases to 0.3165 or increases to 0.4585 given that the passenger was an adult or a child, respectively. Similarly the conditional probability of survival of a male child traveling in first class can be computed.

```
R> cond <- c(Age = "Child", Class = "1st", Sex = "Male")
R> prob(mod1, c(Survived = "Yes"), conditional_on = cond)
```

```
[1] 0.9770115
```

Notice that this exactly the same probability shown with the `summary` function for vertices in stage 1 of the variable `Survived`. This probability is also estimated to be the same for the conditioning events shown above with the `get_path` function.

All atomic probabilities related to the leaves of the Staged Tree can be also obtained as follows:

```
R> obs <- expand.grid(mod1$tree[4:1])[, 4:1]
R> cbind(obs, p = round(prob(mod1, obs), 6))
```

```
   Class    Sex   Age Survived        p
1    1st   Male Child       No 0.000089
2    1st   Male Child      Yes 0.003784
3    1st   Male Adult       No 0.050130
4    1st   Male Adult      Yes 0.032898
5    1st Female Child       No 0.000001
6    1st Female Child      Yes 0.000058
7    1st Female Adult       No 0.001395
8    1st Female Adult      Yes 0.059304
9    2nd   Male Child       No 0.000078
10   2nd   Male Child      Yes 0.003318
11   2nd   Male Adult       No 0.062524
12   2nd   Male Adult      Yes 0.010286
13   2nd Female Child       No 0.000139
14   2nd Female Child      Yes 0.005898
15   2nd Female Adult       No 0.006516
16   2nd Female Adult      Yes 0.040727
17   3rd   Male Child       No 0.020339
18   3rd   Male Child      Yes 0.005914
19   3rd   Male Adult       No 0.176434
```

```
20    3rd    Male  Adult      Yes 0.029025
21    3rd  Female  Child       No 0.006092
22    3rd  Female  Child      Yes 0.003998
23    3rd  Female  Adult       No 0.047675
24    3rd  Female  Adult      Yes 0.031286
25   Crew    Male  Child       No 0.000000
26   Crew    Male  Child      Yes 0.000000
27   Crew    Male  Adult       No 0.303119
28   Crew    Male  Adult      Yes 0.088141
29   Crew  Female  Child       No 0.000000
30   Crew  Female  Child      Yes 0.000000
31   Crew  Female  Adult       No 0.001440
32   Crew  Female  Adult      Yes 0.009000
```

It shows that around 30% of the observations follows the root-to-leaf path `Crew`, `Male`, `Adult`, `No`.

As another possible usage of the `prob` function, it can be used to retrieve the estimated parameters for any vertex of the tree. This can be performed through the option `conditional_on` by specifying a path starting from the root and arrives to the desired vertex. For instance, the probability parameters associated to the event `Age` = "Adult" conditioned on the eight paths defined by the stratified Staged Tree in Figure 5.2 for the vertices in the stratum of `Age` can be computed as below. The output retrieves the stage parameters shown through the `summary` of `mod1`.

```
R> grid <- expand.grid(mod1$tree[2:1])[, 2:1]
R> cbind(Stage = mod1$stages$Age, grid,
         p = prob(mod1, data.frame(Age = rep("Adult", 8)),
         conditional_on = grid))
```

```
Stage  Class     Sex         p
    1    1st    Male 0.9554318
    2    1st  Female 0.9990291
    1    2nd    Male 0.9554318
    3    2nd  Female 0.8866995
    3    3rd    Male 0.8866995
    3    3rd  Female 0.8866995
    2   Crew    Male 0.9990291
    2   Crew  Female 0.9990291
```

In the package also a function (`confint`) for the estimation of confidence intervals

for Staged Tree parameters is provided. For instance, confidence intervals for stages related to `Age` with the `goodman` method can be computed as follows:

```
R> confint(mod1, "Age", method = "goodman")
```

```
                          2.5 %       97.5 %
Age=Child|1  0.0258101180  0.075897181
Age=Adult|1  0.9241028192  0.974189882
Age=Child|2  0.0001411609  0.006645046
Age=Adult|2  0.9933549540  0.999858839
Age=Child|3  0.0907102352  0.140646387
Age=Adult|3  0.8593536135  0.909289765
```

Finally, barplots can be created to give a visual representation of the estimated probabilities associated to a stratum of the tree as reported in Figure 5.7.

```
R> barplot(mod3, "Survived", ylab = "Survived", horiz = TRUE,
          args.legend = list(x = 1), legend.text = TRUE)
```

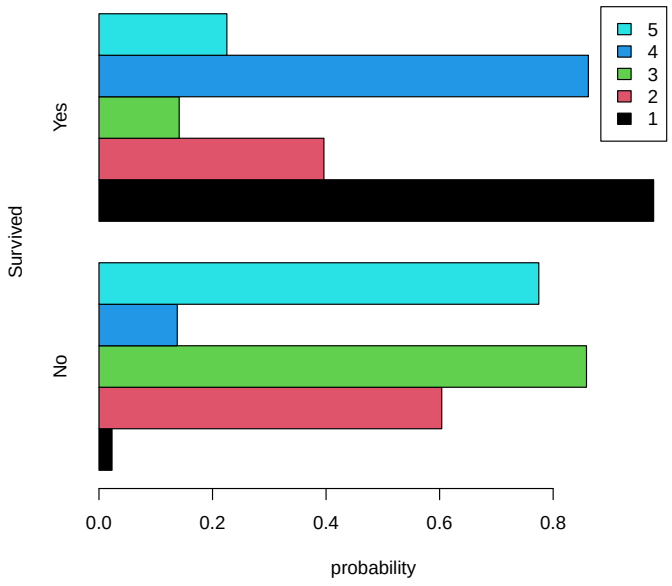


Figure 5.7: Output of the `barplot` function for the variable `Survived` according to the stage structure of `mod3` depicted in Figure 5.3 left.

5.4.4 A Dataset analysis: PhDArticles

The `data.frame` `PhDArticles` includes information regarding the number of publications of 915 PhD biochemistry students during the 1950s and 1960s [85] and it is available in the **stagedtrees** package.

The pipe operator from the **magrittr** package [3] is also used. Even if it is not essential for the **stagedtrees** implementations and it is not one of the dependencies, the use of the pipe operator improves readability of the code and simplifies the user experience.

```
R> library(magrittr)
R> data("PhDArticles")
R> str(PhDArticles)

'data.frame':   915 obs. of  6 variables:
 $ Articles: Factor w/ 3 levels "0","1-2",">2": 1 1 1 ...
 $ Gender  : Factor w/ 2 levels "male","female": 1 2 2 ...
 $ Kids    : Factor w/ 2 levels "yes","no": 2 2 2 ...
 $ Married : Factor w/ 2 levels "no","yes": 2 1 1 ...
 $ Mentor  : Factor w/ 3 levels "low","medium","high": 2 2 2 ...
 $ Prestige: Factor w/ 2 levels "low","high": 1 1 2  ...

R> bn <- bnlearn::hc(PhDArticles)
R> plot(bn)

R> order <- c("Gender", "Kids", "Married", "Articles")
R> bn.as.tree <- as_sevt(bn.fit(bn, data = PhDArticles),
     order = order)
R> plot(bn.as.tree)
```

The learned BN model in Figure 5.8 left states that the number of publications (`Articles`) is marginally independent of `Gender`, `Married` and `Kids` and states that the prestige of the University is conditionally independent of the number of publications of the student given the number of publications of the mentor. The strength of the marginal independence between `Articles` and (`Gender`, `Kids`, `Married`) is investigated in Figure 5.8 right.

On these four variables, a Staged Tree starting from the independence tree (`phd.mod1`) and from the full tree (`phd.mod2`) is learned using the hill-climbing algorithm and they are reported in Figure 5.9.

```
R> phd.mod1 <- PhDArticles %>% indep(order = order) %>%
```

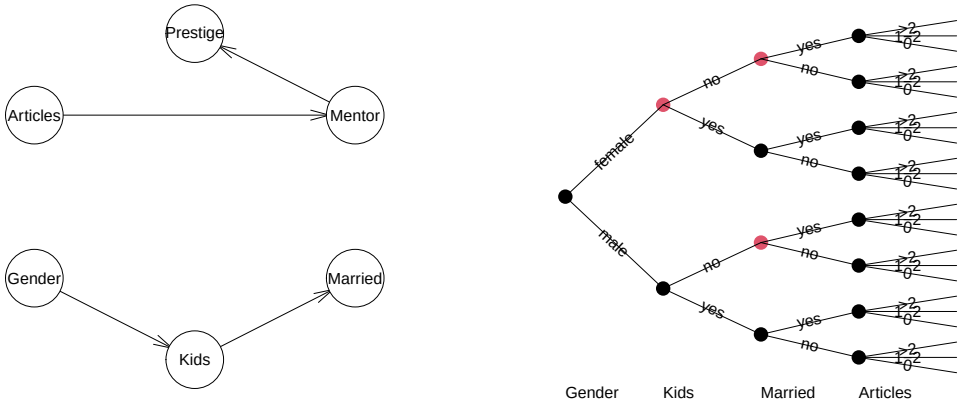


Figure 5.8: BN model learned over the `PhDArticles` dataset and equivalent Staged Tree over `Gender`, `Kids`, `Married` and `Articles`.

```
      stages_hc
R> phd.mod2 <- PhDArticles %>% full(order = order) %>%
      stages_hc
```

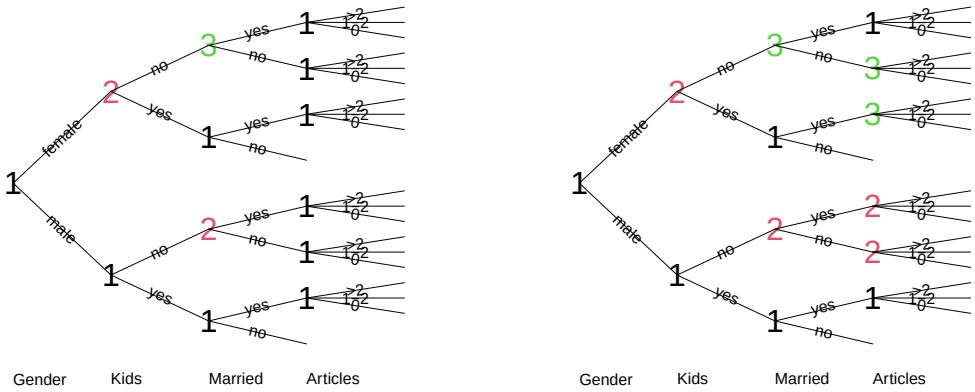


Figure 5.9: Staged tree models learned over the variables `Gender`, `Kids`, `Married` and `Articles` of `PhDArticles`. Left: Staged Event Tree `phd.mod1`. Right: Staged Event Tree `phd.mod2`.

```
R> compare_stages(phd.mod1, phd.mod2, plot = TRUE,
      method = "stages")
[1] FALSE
```

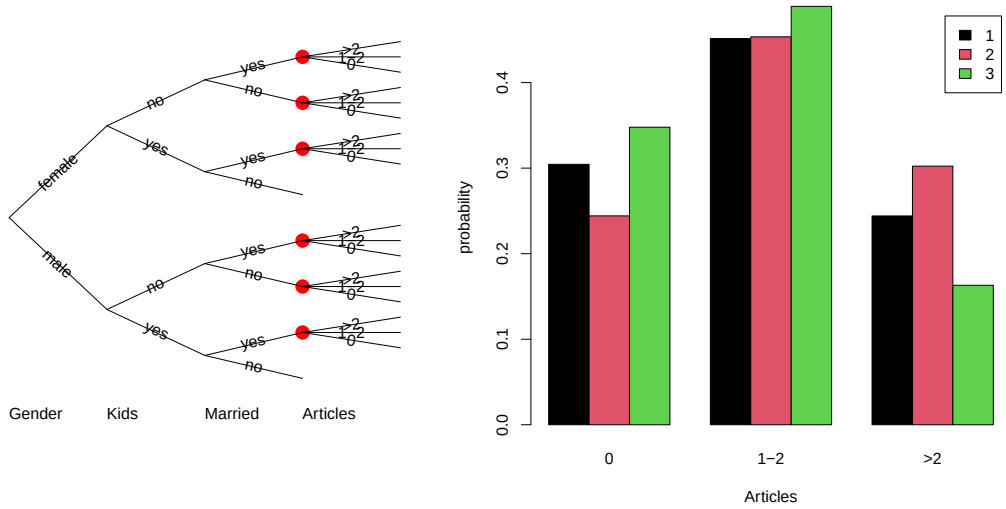


Figure 5.10: Left: Comparison between `phd.mod1` and `phd.mod2` over the variables `Gender`, `Kids`, `Married` and `Articles` of `PhDArticles`. Right: Conditional probability of `Articles` given `Gender`, `Kids` and `Married` for the stages in `phd.mod2`.

The estimated staging structures of the two Staged Trees in Figure 5.9 show that for the first three variables they are exactly equal, according also to the comparison depicted in Figure 5.10 left. Conversely, for the variable `Articles` in `phd.mod1` only one stage distribution is estimated and in `phd.mod2` three stages distributions are obtained (apart from the unobserved situations in the `"UNOBSERVED"` stage). To further explore the different conditional probabilities associated to the stages for `Articles` in `phd.mod2`, the `barplot` function can be used (Figure 5.10 right).

```
R> barplot(phd.mod2, "Articles", legend.text = TRUE,
     xlab = "Articles")
```

From the output in Figure 5.10 right together with the Staged Tree in Figure 5.9 right, it can be noted that unmarried women without kids as well as married women with kids (stage 3) have the lowest estimated probability of a high number of articles. The population with the highest probability of a high number of publications consists of men with no kids (stage 2).

A likelihood-ratio test can be carried out with `LR_test` to test if the simpler `phd.mod1` model describes the data sufficiently well compared to the more complex `phd.mod2`. The function automatically checks if the two input models are nested.

```
R> LR_test(phd.mod1, phd.mod2)
p-value = 0.001972608
```

The small p-value obtained (< 0.05) confirms that the asymmetric structure described by `phd.mod2` is indeed supported by the data.

Finally, a Staged Tree over all the variables in `PhDArticles` is built by using the backward-joining algorithm implemented in `stages_bj`. In Figure 5.11 the plot of the resulting model is displayed together with the barplot associated to `Articles` conditional probabilities.

```
R> order <- c("Prestige", "Mentor", order)
R> phd.all <- PhDArticles %>% full(order = order) %>%
     stages_bj(thr = + 0.5) %>% stndnaming
```

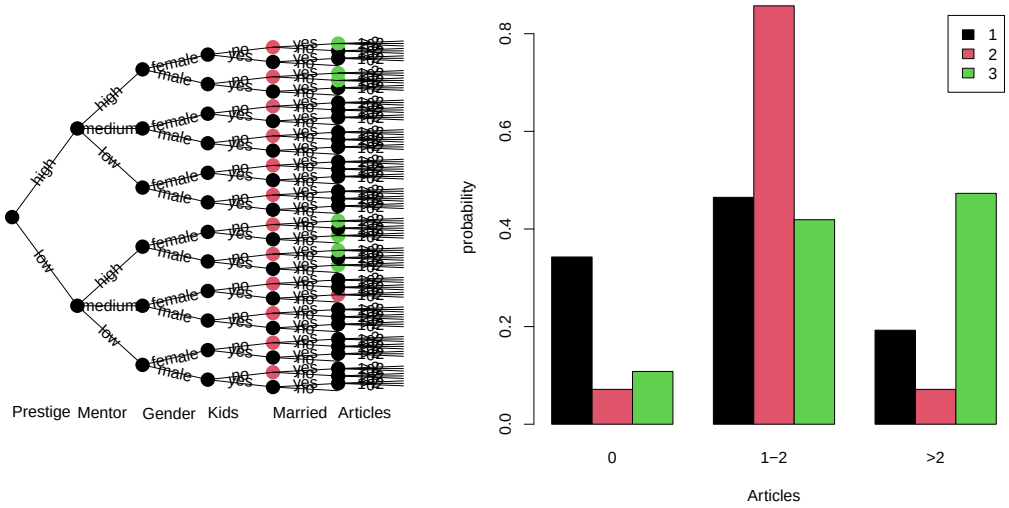


Figure 5.11: Staged Tree `phd.all` (left) over all the variables of `PhDArticles` and corresponding estimated conditional probabilities for stages related to variable `Articles` (right).

The stage with highest probability of a large number of articles (stage 3) includes now the following paths:

```
R> get_path(phd.all, "Articles", "3")
```

	Prestige	Mentor	Gender	Kids	Married
18	low	high	male	yes	yes
20	low	high	male	no	yes
22	low	high	female	yes	yes
24	low	high	female	no	yes
43	high	high	male	no	no
44	high	high	male	no	yes
48	high	high	female	no	yes

So, PhD students with a high number of publications all have a mentor with a high number of publications and most of them are married and with no kids.

In all previous analyses we were interested in assessing how the number of articles were affected by the other factors. For this reason, the variable `Articles` was chosen to be the last in the order, whilst the others were arbitrarily fixed according to one of the topological orders of the learned BN. However, the function `search_dynamic` implements the dynamic programming ordering algorithm of Cowell et al. [28] to search an optimal order of the variables from data.

```
R> phd.order <- search_dynamic(PhDArticles)

Staged event tree (fitted)
Articles[3] -> Married[2] -> Mentor[3] -> Gender[2] ->
Prestige[2] -> Kids[2]
'log Lik.' -4076.919 (df=19)
```

In the optimal order `Articles` is chosen as the root of the tree and therefore its staging can not be studied to assess how it depends on the other variables. However, according to the BIC, this new Staged Tree provides a great improvement compared to `phd.all`.

```
R> BIC(phd.all, phd.order)

             df      BIC
phd.all      15 8504.529
phd.order    19 8283.398
```

5.5 A Simulation Study

A simulation analysis of structural learning algorithms implemented in **stagedtrees** is performed on nine datasets, chosen mostly from the literature on CEGs and PGMs for contingency tables. The main features of these datasets are summarized in Table 5.1, which for each dataset gives the number of observations, variables, root-to-leaf paths, cells with zero counts, non-leaf nodes and edges in the corresponding Stratified Staged Tree. Some dataset is available directly in the *R* software, in particular from packages **stagedtrees**, **gRbase** [35], **MBCbook** [14] and **datasets**.

It is not the purpose of this section to show how to model these datasets and interpret the estimated stage structure in each specific context. For how to perform inference on Staged Tree see Section 5.4, while for more details about the datasets, its

features and main references refer to Table 5.2 or the help function in *R*. Indeed, here only indices relating to the performances of the estimated algorithms are reported, without going into details of their interpretation.

Dataset	# observations	# variables	# root-to-leaf paths λ	# non-leaf nodes	# 0 cells	# edges
`Asym`	1000	4	16	15	1	30
`chestSim500`	500	8	256	255	182	510
`FallEld`	50000	4	64	27	0	90
`PhDArticles`	915	6	144	136	0	279
`Pokemon`	999	5	32	31	0	62
`puffin`	69	6	768	343	284	1110
`reinis`	1841	6	64	63	0	126
`selfy`	2804	4	72	34	4	105
`Titanic`	2201	4	32	27	0	58

Table 5.1: Summary information about the datasets considered for the simulation study.

Dataset	References	*R* Package
`Asym`	simulated dataset	**stagedtrees**
`chestSim500`	Højsgaard et al. [62]	**gRbase**
`FallEld`	Shenvi et al. [104]	
`PhDArticles`	Long [85]	**stagedtrees**
`Pokemon`	Gabbiadini et al. [48]	**stagedtrees**
`puffin`	Bouveyron et al. [14]	**MBCbook**
`reinis`	Højsgaard et al. [62]	**gRbase**
`selfy`	Dalla Zuanna et al. [31]	
`Titanic`	Dawson [33]	**datasets**

Table 5.2: Main references and *R* packages related to the analyzed datasets.

For all nine datasets and all algorithms from the **stagedtrees** package, the simulations respect the following characteristics:

- all the algorithms are estimated after the isolation of unobserved situations in own stages;
- score-based algorithms, namely the ones based on a hill-climbing optimization, use the maximization of the negative BIC as `score`;
- for distance-based algorithms, all the distances and divergences implemented in **stagedtrees** are adopted, through the option `distance`, each one with four different threshold values, i.e. 0.01, 0.05, 0.20 and 0.50;
- a Bayesian Network learnt with **bnlearn** package has been refined through the backward hill-climbing algorithm (Refined BN);

- the algorithm based on the clustering of the probability distributions associated to vertices of the Staged Tree is estimated three times, i.e. defining two, three or four different stages in each stratum of the Staged Tree. Stages embedding unobserved situations are not counted, since they do not require the estimation of any parameter. The clustering method is based on Total Variation as distance between pair of probability distributions (HClust);
- the kmeans algorithm is based on the *Hartigan-Wong* approach [56] and it is estimated three times, defining two, three or four different stages in each stratum of the Staged Tree, without counting stages corresponding to unobserved situations.

For more details see Table 5.3.

Name	Function
Indep	`indep`
Full	`full`
HC - Indep	`stages_hc`
BHC	`stages_bhc`
Fast BHC	`stages_fbhc`
Random BHC	`stages_bhcr`
Kullback-Leibler	`stages_bj`
Manhattan	`stages_bj`
Euclidean	`stages_bj`
Renyi	`stages_bj`
Total Variation	`stages_bj`
Hellinger	`stages_bj`
Bhattacharyya	`stages_bj`
Chan-Darwiche	`stages_bj`
HClust	`stages_hclust`
Kmeans	`stages_kmeans`
Refined BN	`stages_bhc(as_sevt(bn.fit( )))`

Table 5.3: List of algorithms from the *R* package **stagedtrees** used to estimate stage structures of Stratified Staged Trees for the nine datasets in Table 5.2.

For each dataset, each algorithm is run 10 times on 80% of the data randomly selected and the estimated model is tested on the remaining part (20%). The average of all the investigated quantities over the 10 runs is then computed. The collected indices of performance for each estimated model are the number of degrees of freedom, log-likelihood, AIC and BIC values, classification accuracy (the response variable is the one in the first stratum of the Stratified Staged Tree) and the computational cost (in seconds).

From Table 5.4 reported in this section as guideline and from Table 7.6 to Table 7.13 in Appendix, the results obtained with all algorithms on the nine datasets are shown. The following general conclusions can be made:

- Full and Indep are the starting models in order to compare the performances of all the structural learning algorithms implemented. The first fits a full-dependence structure to the dataset, by providing one of the best results according to the log-likelihood, due to the overfitting introduced. The Indep model fits a full-independence structure to the dataset, estimating always the smallest log-likelihood, due to its underfitting.

- The number of estimated parameters (df) is highly variable, according to the criterion and the starting stage structure (dependence or independence model). As expected, for backward algorithms with a distance-based joining, the higher is the threshold below which the distance between the probability distributions of two stages are set to be equal, the lower will be the number of estimated parameters.

- Most often, the higher the number of degrees of freedom a model has, the higher is the correspoding log-likelihood value.

- The minimum values of the AIC and BIC indices are attained with hill-climbing algorithms. This is intuitive, because the implemented score-based algorithms have as optimization default the minimization of the BIC index. However, even if the distance-based algorithms not aiming to minimize these indices, their performances according to AIC and BIC values are satisfactory and comparable with the score-based methods.

- The hill-climbing algorithms are slower than others. In particular, the hill-climbing starting from the independence model (HC - Indep) is the slowest, because it both joins and splits stages. Conversely, distance-based methods, fast or random backward hill-climbing and HClust are the fastest.

- The accuracy of all models is comparable, the lowest scoring models being Indep, HClust and Kmeans due to their simplicity. Occasionally, some distance-based algorithm provides low accuracy with threshold 0.50. This is because many situations are merged together in the same stage, being 0.50 often a too high threshold under which two discrete distributions are set to be equal.

Model	df	logLik	AIC	BIC	Accuracy	Computational Time
Indep	10.00	-7892.51	15805.03	15862.19	0.76	0.21
Full	64.60	-6251.36	12631.92	13001.18	0.85	0.22
HC - Indep	31.00	-6277.94	12617.89	12795.08	0.85	1.05
BHC	32.60	-6271.70	12608.60	12794.94	0.85	0.34
Fast BHC	31.00	-6301.09	12664.18	12841.38	0.85	0.22
Random BHC	37.60	-6284.50	12644.21	12859.13	0.85	0.23
Kullback-Leibler - 0.01	60.40	-6250.38	12621.56	12966.80	0.85	0.21
Kullback-Leibler - 0.05	50.20	-6250.88	12602.17	12889.11	0.85	0.22
Kullback-Leibler - 0.20	38.00	-6262.82	12601.65	12818.86	0.85	0.23
Kullback-Leibler - 0.50	28.20	-6308.05	12672.49	12833.68	0.84	0.22
Manhattan - 0.01	64.40	-6251.30	12631.40	12999.51	0.85	0.22
Manhattan - 0.05	62.20	-6250.53	12625.45	12980.99	0.85	0.21
Manhattan - 0.20	46.40	-6253.90	12600.61	12865.83	0.85	0.22
Manhattan - 0.50	29.40	-6331.54	12721.88	12889.93	0.84	0.21
Euclidean - 0.01	64.20	-6251.16	12630.72	12997.69	0.85	0.22
Euclidean - 0.05	59.00	-6250.15	12618.31	12955.55	0.85	0.21
Euclidean - 0.20	40.50	-6263.78	12608.56	12840.06	0.85	0.21
Euclidean - 0.50	20.40	-6595.01	13230.83	13347.43	0.84	0.22
Renyi - 0.01	63.00	-6251.07	12628.13	12988.24	0.85	0.21
Renyi - 0.05	54.80	-6250.66	12610.91	12924.15	0.85	0.22
Renyi - 0.20	44.80	-6254.68	12598.96	12855.04	0.85	0.22
Renyi - 0.50	35.00	-6274.63	12619.25	12819.31	0.85	0.23
Total Variation - 0.01	64.40	-6251.30	12631.40	12999.51	0.85	0.22
Total Variation - 0.05	62.20	-6250.53	12625.45	12980.99	0.85	0.21
Total Variation - 0.20	46.40	-6253.90	12600.61	12865.83	0.85	0.22
Total Variation - 0.50	29.40	-6331.54	12721.88	12889.93	0.84	0.22
Hellinger - 0.01	51.40	-6250.56	12603.91	12897.72	0.85	0.22
Hellinger - 0.05	38.00	-6262.60	12601.20	12818.41	0.85	0.21
Hellinger - 0.20	21.00	-6560.98	13163.95	13283.99	0.83	0.22
Hellinger - 0.50	15.00	-6904.43	13838.87	13924.61	0.77	0.22
Bhattacharyya - 0.01	45.60	-6253.13	12597.46	12858.11	0.85	0.22
Bhattacharyya - 0.05	28.40	-6307.20	12671.20	12833.53	0.84	0.22
Bhattacharyya - 0.20	18.00	-6826.32	13688.63	13791.52	0.80	0.22
Bhattacharyya - 0.50	13.00	-7333.95	14693.90	14768.21	0.80	0.22
Chan-Darwiche - 0.01	64.40	-6251.30	12631.40	12999.51	0.85	0.22
Chan-Darwiche - 0.05	64.40	-6251.30	12631.40	12999.51	0.85	0.22
Chan-Darwiche - 0.20	62.80	-6250.97	12627.53	12986.50	0.85	0.22
Chan-Darwiche - 0.50	54.80	-6250.40	12610.39	12923.63	0.85	0.22
HClust k = 2	16.00	-6724.99	13481.97	13573.43	0.80	0.22
HClust k = 3	22.00	-6530.52	13105.03	13230.78	0.83	0.22
HClust k = 4	27.00	-6373.04	12800.09	12954.42	0.83	0.22
Kmeans k = 2	16.00	-6689.69	13411.39	13502.85	0.81	0.21
Kmeans k = 3	22.00	-6497.92	13039.83	13165.58	0.81	0.22
Kmeans k = 4	27.00	-6377.29	12808.58	12962.91	0.83	0.22
Refined BN	28.60	-6286.85	12630.89	12794.37	0.85	0.31

Table 5.4: Mean results for **stagedtrees** algorithms over 10 replications for `selfy` dataset. Experiments performed on a standard laptop with 8 GB of RAM and an i5 3.1 GHz CPU.

5.6 Application on Pediatric Dentistry Framework

The analysis done in Section 1.4 on the Pediatric Dentistry dataset is carried forward in this section with Stratified Staged Trees and ALDAGs. The need to implement these new types of Graphical Models arises from the fact that in this pediatric study there were structural zeros (Table 1.7), which led to obtain approximate estimates through the UGM proposed in Section 1.4.1. These structural zeros can be managed efficiently through Staged Trees, as it can be seen in Figure 5.12. Indeed, it shows that for children which have not been breastfed, i.e. type of breastfeeding equal to 0, the breastfeeding time is clearly equal to 0 months.

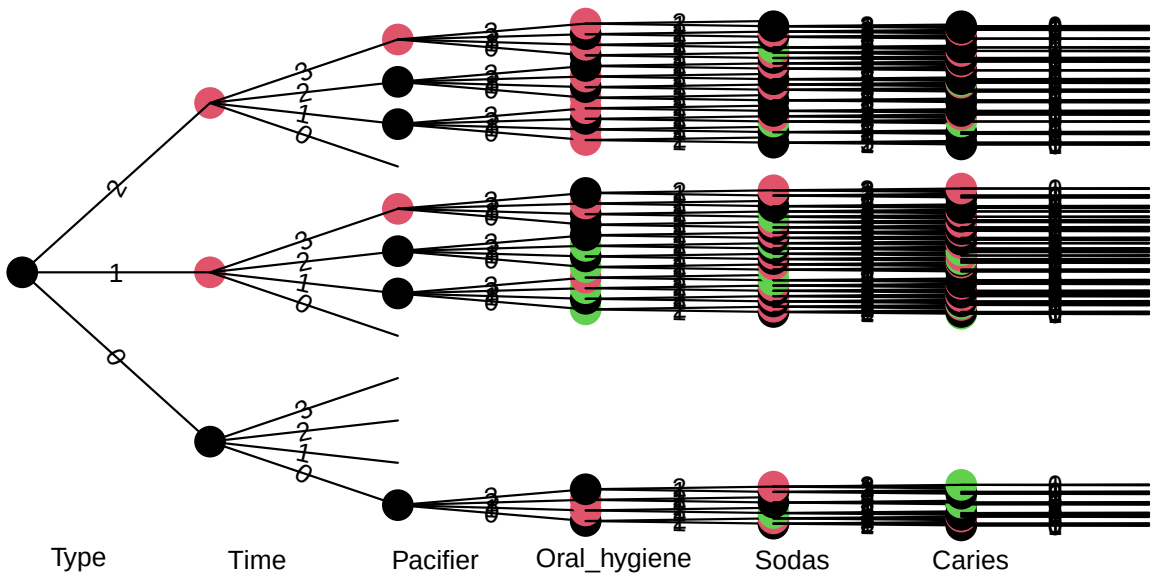


Figure 5.12: Stratified Staged Tree estimated on the six connected variables in the UG in Figure 1.1.

The construction of the Staged Tree needs the choice of a variables ordering. For this, first a Bayesian Network is estimated on the six connected variables in Figure 1.1 in order to obtain the set of possible topological orders of variables. Among them, it is chosen an ordering that can be interpretable for a practical point of view for the dentists: type of breastgeeding, time of breastfeeding, use of pacifier,

oral hygiene status, consumption of sugared beverages (Sodas) and caries variation. The Stratified Staged Tree in Figure 5.12 is obtained with a backward hill-climbing algorithm, starting from the full-dependence stage structure, isolating structural and observed zeros in own stages, for each stratum of the tree. Figure 5.12 gives the plot of the tree with the removal of these unobserved vertices, in order to have a more straightforward graphical representation. For instance, the stage composed by the two red nodes for breastfeeding time shows the partial conditional independence

$$\text{Time} \perp\!\!\!\perp \text{Type} \mid \text{Type} = \{1, 2\}.$$

For a better visualization of the estimated stage structure, since the tree is not very readable for the last two strata, the subtrees conditioned on the three levels of the root variable, i.e. the type of breastfeeding, are shown in Figures 5.13, 5.14 and 5.15. Figure 5.13 shows only the tree associated to the time of breastfeeding equal to 0, being it the only category assumed by children who have not been breastfed (structural zeros).

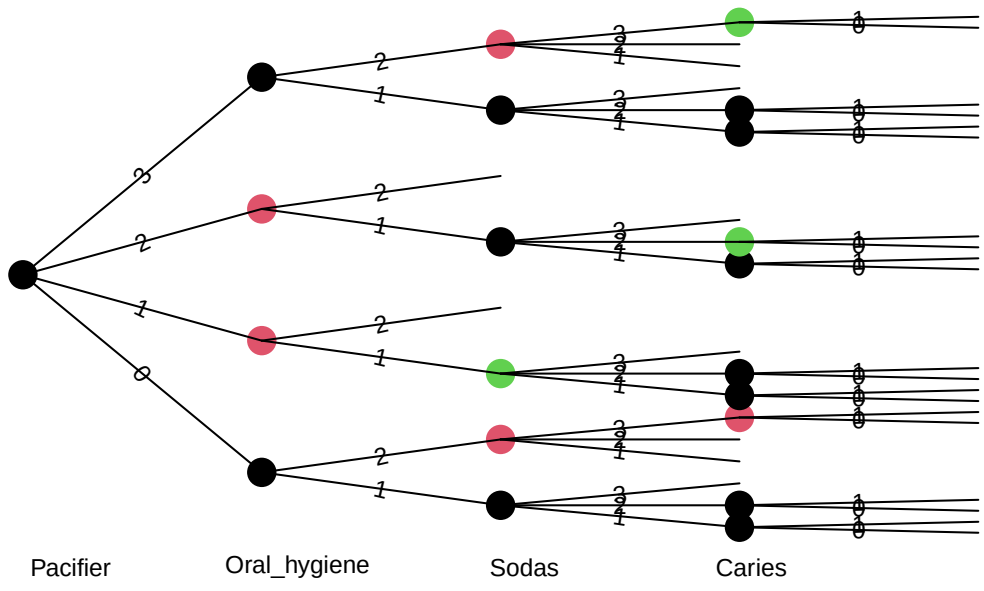


Figure 5.13: Subtree considering only "0" for type of breasfeeding and time of breastfeeding.

Also from Figure 5.14 and 5.15 an asymmetric dependence structure is high-

lighted, proving the usefulness in this context of an approach based on Staged Trees. Since the complete tree and the three subtrees have many vertices, a reading of the conditional independences according to the corresponding stage structure is difficult. An attempt was therefore made to simplify the graphical representation by constructing the CEG, which however did not lead to a clear improvement. For this reason his graph is not shown here.

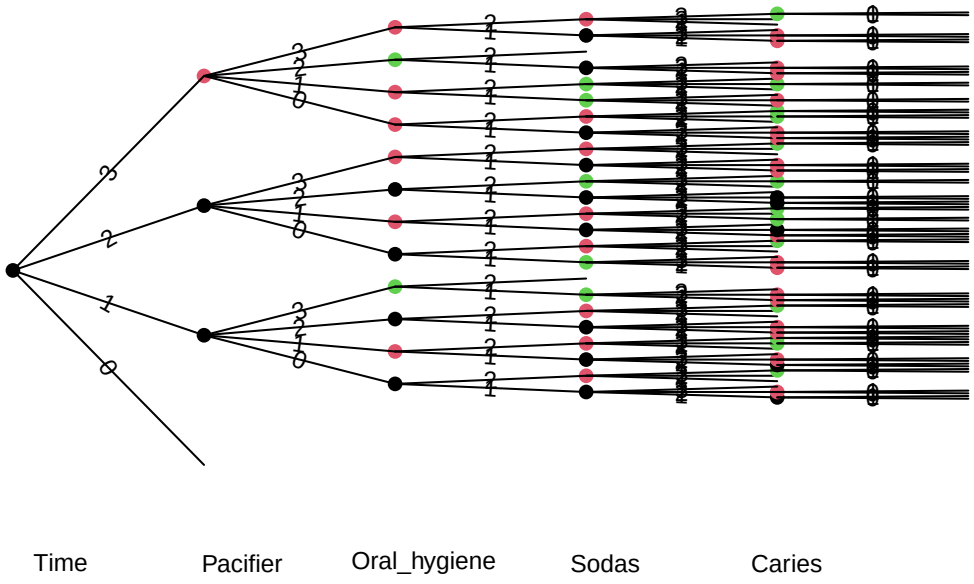


Figure 5.14: Subtree considering only type of breasfeeding equal to "1".

Context-specific, partial and local independences can be read all in a unique graph through the ALDAG (Figure 5.16) corresponding to the Stratified Staged Tree in Figure 5.12. This type of graph is very useful when the interpretation of the Staged Tree is difficult or impossible for its dimension. The ALDAG highlights many dependences, since its graph is complete. Note for instance the partial independence (blue edge) between type and time of breastfeeding, as observed studying the stage coloring in the second stratum of the Stratified Staged Tree. It is a current work the deepening of this asymmetric dependence structure in order to provide useful clinical insights. Indeed, this is not an easy task, since the ALDAG has a complex structure. A possible choice in order to simplify the structure Stratified Staged Tree and consequently also that of the ALDAG is to use another learning algorithm for

the estimation of the stage structure. For instance we can use the distance-based one with a threshold that is not too low, so as not to be too restrictive and therefore in order to place an higher number of vertices of a stratum of the tree in the same stage.

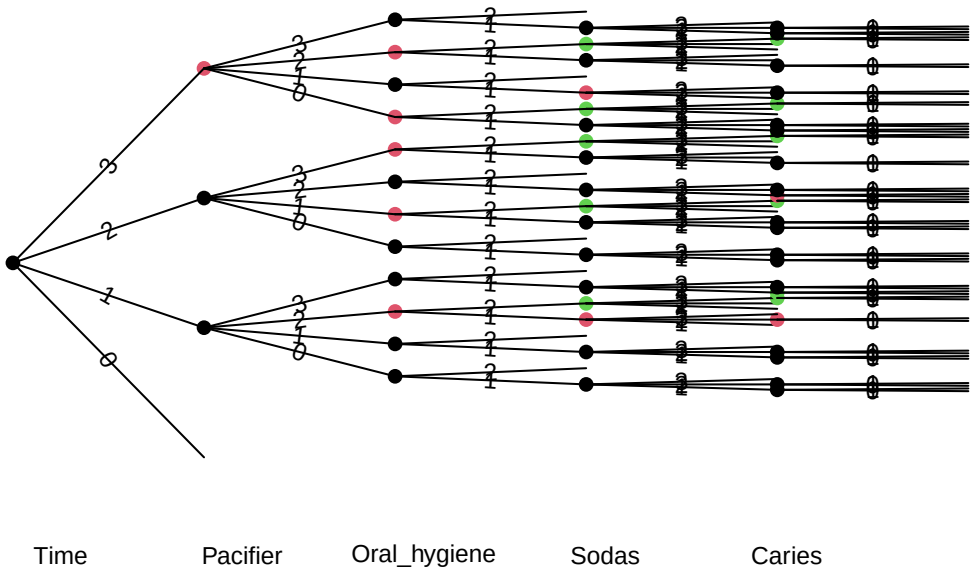


Figure 5.15: Subtree considering only type of breasfeeding equal to "2".

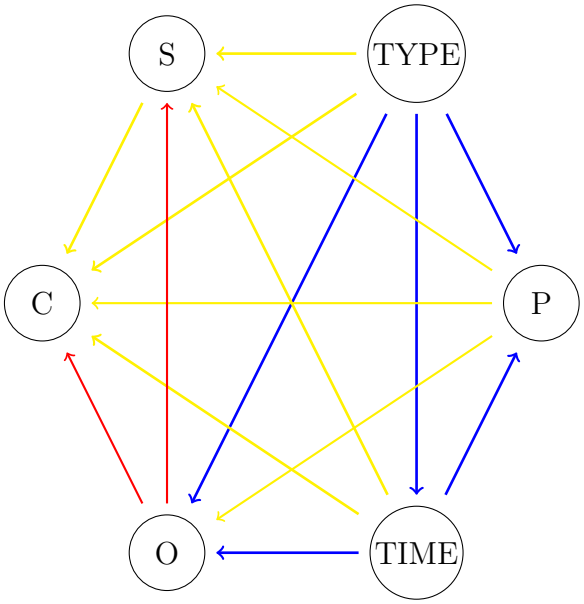


Figure 5.16: Asymmetry-labeled DAG estimated on the Pediatric Dentistry dataset. TYPE: Breastfeeding type, TIME: Breastfeeding time, P: Use of pacifier, C: Caries Variation, O: Oral hygiene status, S: Consumption of sugared beverages. The edge coloring is: red - context; blue - partial; yellow - context/partial.

5.7 Conclusion

The chapter presents the *R* package **stagedtrees** [120] for estimating Staged Trees and Chain Event Graphs from a dataset. Its functionalities are showed and exemplified and a simulation study is carried out to check the efficiency of its different structural learning algorithms. The package is also supported by Carli et al. [20], which explains its main features along the lines of what was done throughout this chapter.

The analysis of the pediatric dentistry dataset with Staged Trees carried out in Section 5.6 highlights many asymmetric dependences, suggesting that an UGM is not suitable in this context. Currently, we are studying this asymmetric dependence structure among the six connected variables of interest in order to provide useful insights for the experts.

Chapter 6

Staged Trees as a Classification Tool

Generative models for classification purpose employ the joint probability distribution of the response variable and the covariates to construct a decision rule. Among generative models, Bayesian Networks and Naive Bayes Classifiers are the most commonly used and they provide a clear graphical representation of the relationship among all variables. However, they have the disadvantage of highly restricting the type of relationships that could exist, by not allowing for context-specific independences. In Carli et al. [21] a new class of generative classifiers is introduced, called **Staged Tree Classifiers (STCs)**, which formally account for asymmetric and context-specific independences. The Naive Staged Tree Classifier is also defined, which extends the Naive Bayes Classifier whilst retaining the same complexity. A simulation study in Section 6.6 shows that the classification accuracies of Staged Tree Classifiers implemented using **stagedtrees** are competitive with those obtained through classifiers of the state of the art of machine learning.

The aim of statistical classification is to assign labels to instances described by a vector of explanatory/feature variables. The classification task is guided by a statistical model learnt using data containing labeled instances. The array of models now designed to perform classification is constantly increasing and includes, among others, Random Forests [61], Recursive Partitioning [17] and Neural Networks [108].

Bayesian Network Classifiers (BNCs) [9, 47] are special types of BNs designed for classification problems. These have been applied to a wide array of real-world applications with competitive classification performance against classifiers from the state of the art [44]. There are many advantages associated to BNCs. First, they provide an explicit and intuitive representation of the relationship among features represented by a graph. Second, they are a fully coherent probabilistic model thus

giving uncertainty measures about the chosen labels. Third, many of the methods and algorithms developed for general BNs can be simply adapted and used for BNCs (e.g. Benjumeda et al. [8]). Last, they are implemented in various pieces of user-friendly software, for instance in the *R* package **bnclassify** of Mihaljevic et al. [88]. More generally, BNCs are generative classifiers which give an estimate of the joint probability distribution of both features and class.

As discussed in Chapter 4, one of the main limitations of BNs is that they can only explicitly represent symmetric conditional independences among variables of interest. However, in many applied domains, conditional independences are context-specific, meaning that they only hold for specific instantiations of the conditioning variables. For this reason, extensions of BNs have been proposed that can take into account asymmetric independences [12, 65, 97]. With the exception of the Probabilistic Decision Graph [65], all these models somehow lose the intuitiveness of BNs since they can not represent all the models' information into a unique graph.

In this chapter a novel class of generative classifiers based on Staged Trees is considered. Indeed, although Staged Trees have been used in a variety of applications, including the modelling of health problems [4, 69] and criminal activities [26], their specific use for classification problems has been limited.

STCs are generative classifiers which, whilst extending the class of BNCs to deal with asymmetric conditional independences, share the same advantages of BNCs: first, the relationship among random variables is still intuitively depicted in a unique graph; second, they are fully coherent probabilistic model; third, learning algorithms already defined for Staged Trees [20, 26, 46] can simply be adapted for classification purposes; fourth, the freely-available *R* package **stagedtrees** gives an implementation of a variety of learning algorithms and inferential routines to apply the methods in practice.

Mirroring the theory of BNCs, Staged Tree Classifiers of different complexity are discussed and their properties investigated. Experimental studies demonstrate that they have comparable classification rates to state of the art classifiers, with the added benefit that they are highly expressive in a meaning that is formalized below.

Section 6.1 gives an overview about the theory of Bayesian Network Classifiers, while in Section 6.2 the Staged Tree Classifiers are presented. Section 6.3 shows the relationship between BNCs and STCs and Section 6.4 provides two propositions for the reading on STCs of dependences between the class variable and the freatures. Section 6.5 proposes the corresponding of the Naive Bayes Classifier in the context of Staged Trees, which is called Naive Staged Tree Classifier. In Section 6.6 a classification experiment involving 14 datasets is conducted, comparing the performances

of 9 STCs implemented through **stagedtrees** with respect to those obtained with BNCs, classification tree, logistic model, neural network, random forest and naive bayes. Section 6.7 provides a conclusion of the chapter with the main insights and results obtained.

6.1 Bayesian Network Classifiers

Let $\mathbf{X} = (X_1, \ldots, X_p)$ be a vector of discrete feature variables with sample space $\mathbb{X} = \times_{i=1}^{p} \mathbb{X}_i$, where $\mathbb{X}_i$ is the sample space of X_i, and $\mathbf{x} = (x_1, \ldots, x_p) \in \mathbb{X}$. Let C be the discrete class variable with sample space $\mathbb{C}$ and $c \in \mathbb{C}$. Given a training set of labeled observations $\mathcal{D} = \{(\mathbf{x}^1, c^1), \ldots, (\mathbf{x}^N, c^N)\}$, where $\mathbf{x}^i \in \mathbb{X}$ and $c^i \in \mathbb{C}$, the aim of a generative classifier is to learn a joint probability $p(c, \mathbf{x})$, shorthand of $p(C = c,\ \boldsymbol{X} = \mathbf{x})$, and assign a non-labeled instance $\mathbf{x}$ to the most probable a posterior class found as

$$\arg\max_{c \in \mathbb{C}}\ p(c \mid \mathbf{x}) = \arg\max_{c \in \mathbb{C}}\ p(c, \mathbf{x}).$$

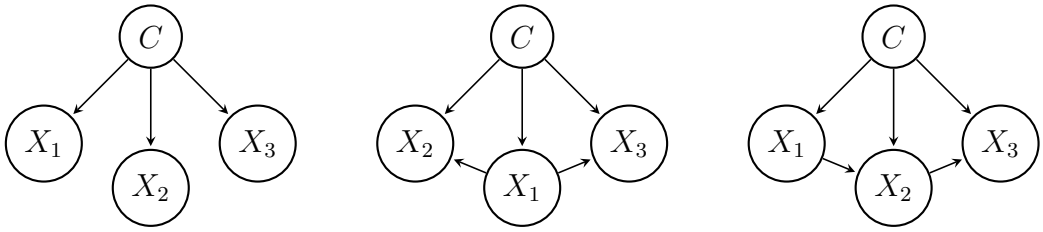
Such a classifier is referred to as *Bayes Classifier*.

BNCs are Bayes Classifiers that factorize $p(c, \boldsymbol{x})$ according to a BN over the variables $X_1, \ldots, X_p$ and C as shown in Equation (6.1), where $\boldsymbol{\pi}(x_i) \in \times_{j \in pa(i)} \mathbb{X}_j$, $\boldsymbol{\pi}(c) \in \times_{j \in pa(c)} \mathbb{X}_i$, $pa(i)$ and $pa(c)$ are the parent sets of X_i and C in the BN, respectively.

$$p(c, \mathbf{x}) \ = \ p(c \mid \boldsymbol{\pi}(c)) \ \prod_{i=1}^{n} \ p(x_i \mid \boldsymbol{\pi}(x_i)) \tag{6.1}$$

Therefore, the learning of a BNC consists of two steps: first the learning of its DAG; second, given the DAG structure, the learning of the probabilities in Equation (6.1). Notice that, given a DAG, by taking advantage of the probability factorization associated to a BNC, the number of parameters to be learnt is usually a lot smaller than $|\mathbb{C}| \times |\mathbb{X}|$, the total number of atoms in the sample space. Furthermore, these probabilities can be easily learnt using the methods designed for generic BNs, both in a frequentist or a bayesian approach.

Although any BN model could be used for classification purposes, most often the underlying DAG is restricted so that the class variable C has no parents, that is $pa(c) = \emptyset$. Therefore, in BNCs the class variable is the root of the DAG and there is a direct link from C to X_i, for $i = 1, \ldots, p$, since otherwise features not connected to the class would not provide any information for classification. The simplest possible model is the so-called Naive Bayes Classifier [89] which assumes



(a) Naive BNC (b) SPODE BNC (c) TAN BNC

Figure 6.1: Examples of BNCs with three features and one class.

the features are conditionally independent given the class (Figure 6.1a). BNCs of increasing complexity can then be defined by adding dependences between the feature variables. For instance, the super-parent-one-dependence-estimator (SPODE) BNC [70] assumes there is a feature parent of all others (Figure 6.1b). Another commonly used classifier is the tree-augmented naive (TAN) BNC [47] for which each feature has at most two parents: the class and possibly another feature (Figure 6.1c).

Although BNCs of any complexity can be learnt and used in practice, empirical evidence demonstrates that model complexity does not necessarily implies better classification accuracy [9]. Despite of their simplicity, Naive BNCs have been shown to lead to good accuracy in classification problems [9, 44]. For such models the inclusion of redundant variables worsen the performance of the classifier [76]. It is therefore critical to include in BNCs a small number of variables with a high predictive association with the class.

Alongside these empirical evaluations of BNCs, theoretical studies about the expressiveness of such models have appeared. Recently, Varando et al. [118, 119] fully characterized the decision functions induced by various BNCs and consequently derived bounds for their expressive power. They built on the work of Ling and Zhang [83] that demonstrated that any BNC whose vertices have at most k parents can not represent any decision function containing $(k+1)$-XORs (also known as parity functions [94]). Thus, Naive BNCs are not capable of capturing any 2-XORs. On the positive side, Domingos and Pazzani [37] demonstrated that Naive BNCs are optimal under a $0-1$ loss even when the assumption of conditional independence among features does not hold.

6.2 Staged Tree Classifiers

The technology of Staged Trees and CEGs has been refined over the years and methods to investigate causal relationships [115], perform statistical inference [53], and check model's robustness [81] are now available. However, the specific use of

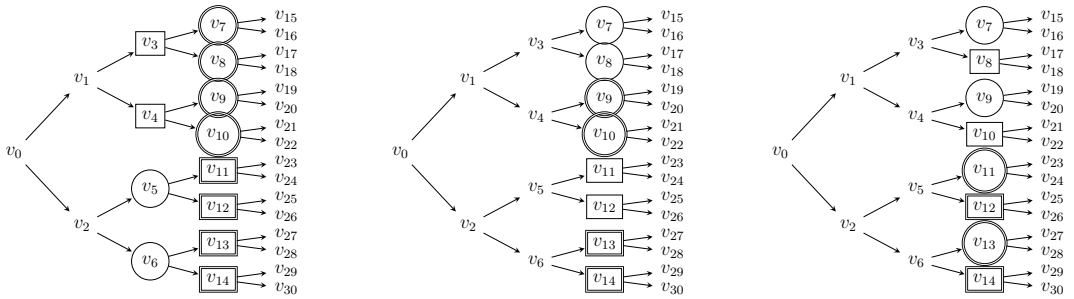


Figure 6.2: Representation of BNCs as Staged Trees Classifiers. (a): Naive BNC in Figure 6.1a as a Staged Tree Classifier. (b): SPODE BNC in Figure 6.1b as a Staged Tree Classifier. (c): TAN BNC in Figure 6.1c as a Staged Tree Classifier.

Staged Trees for classification has not been investigated in the literature. Thus, just as BNCs have been defined as a specific subclass of BNs whose graph entertains some properties, the class of Staged Tree Classifiers is defined here.

Definition 34. *A Staged Tree Classifier for the class C and features $\mathbf{X}$ is a $(C, \mathbf{X})$-compatible Staged Tree.*

The requirement of C being the root of the tree follows from the idea that in most BNCs the class has no parents, so to maximize the information provided by the features for classification. Although classifiers associated to non-product spaces could be considered, here, for comparison to BNCs, the assumption of compatibility is made.

6.3 Relationship between BNCs and Staged Tree Classifiers

The BNCs reviewed in Section 6.1 can now be represented as Staged Tree Classifiers. For instance Naive BNCs (Figure 6.2a), SPODE BNCs (Figure 6.2b) and TAN BNCs (Figure 6.2c) can concisely be represented as Staged Tree Classifiers. It is henceforth assumed that STCs are constructed using a topological order of the features which is compatible to the one embedded in the corresponding BNC.

From the existence of an equivalent Staged Tree compatible with the topological order of a given BN [27], it follows that every BNC such that $pa(c) = \emptyset$ can be directly translated into a Staged Tree.

Proposition 7. *All BNCs such that $pa(c) = \emptyset$ are Stratified Staged Tree Classifiers.*

The proof of Proposition 7 follows by observing that, since $pa(c) = \emptyset$, there exists a topological sorting in which the class variable C is in the first position. In particular Naive Bayes, SPODE, and TAN Classifiers can all be represented by STCs.

Recall that with V_k we indicate the set of vertices at depth k from the root of the tree. Then, Proposition 8 is useful to understand the structure of BNCs and consequently propose extensions of these based on Staged Trees which entertain similar properties.

Proposition 8. *A Naive BNC is equivalent to a Staged Tree Classifier built with an arbitrary ordering of features and where, for all $v \in V_1$, the subtree $\mathcal{T}(v)$ is a Stratified Staged Tree where all nodes at the same distance from the root are in the same stage. In particular, the Naive Bayes equivalent Stratified Staged Tree has $|\mathcal{C}|$ stages per each feature.*

6.4 Conditional Independence in Staged Tree Classifiers

For the specific task of classification, it is possible to derive two results about the dependence between the features and the class in Staged Tree Classifiers.

Proposition 9. *If all $v \in V_k$ of a Staged Tree Classifier are in the same stage then $(C, X_1, X_2, \ldots, X_{k-1})$ and X_k are marginally independent, i.e. $(C, X_1, \ldots, X_{k-1}) \perp\!\!\!\perp X_k$.*

Proposition 10. *For any $v \in V_1$, let $\mathcal{T}(v)$ be the subtree with root v. If all $\mathcal{T}(v)$ have the same stage structure over the vertices at distance $k-1$ from the root, then X_k is independent of C conditionally on $X_1, \ldots, X_{k-1}$, i.e. $C \perp\!\!\!\perp X_k \mid X_1, \ldots, X_{k-1}$.*

These two results are illustrated in Figure 6.3. For instance, consider the features associated to the last random variable in Figure 6.3b. The vertices in the upper half are framed as the vertices in the bottom half, thus implying that the class variable is conditionally independent of the last feature given all others.

6.5 Naive Staged Tree Classifiers

The class of STCs is extremely rich and for any classification task the number of candidate models that could explain the relationship between class and features increases exponentially. One first common assumption that can be made here is to

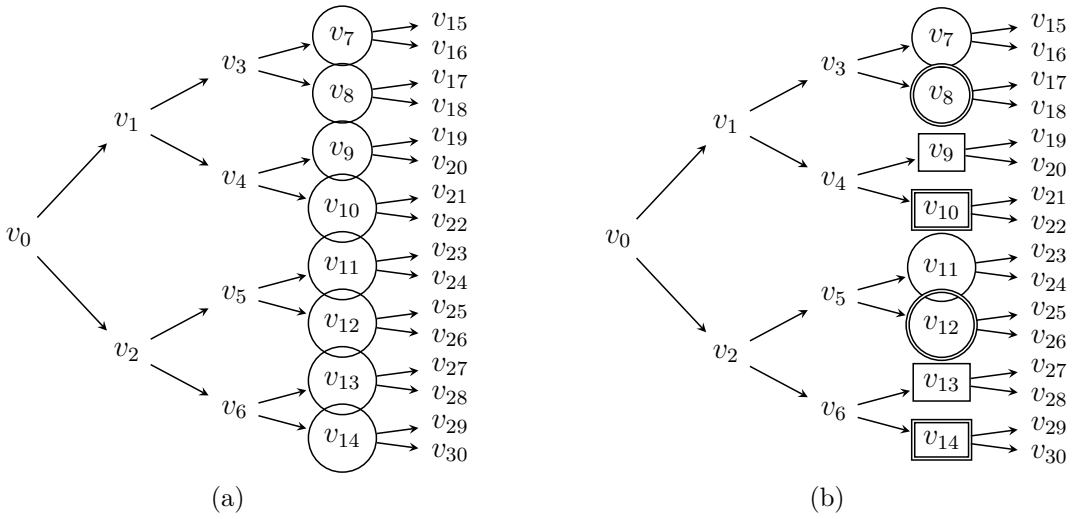


Figure 6.3: Staged Trees Classifiers embedding conditional independence statements among features and the class. (a): $X_3 \perp\!\!\!\perp C$. (b): $X_3 \perp\!\!\!\perp C \mid X_1, X_2$.

consider only Stratified Staged Trees, ones where only vertices at the same distance from the root can be in a same stage. However, even with this assumption the model class of STCs is still much richer than that of BNCs. Therefore, just like BNCs whose DAGs have restricted topologies (as SPODE and TAN classifiers) have been studied, next a class of simpler Staged Tree Classifiers is introduced.

In many practical applications the Naive BNC has very good classification performance despite of its simplicity. A Naive BNC has a total of $\sum_{j=1}^{p} |\mathbb{C}| \cdot (|\mathbb{X}_j| - 1) + |\mathbb{C}| - 1$ free parameters that need to be learnt, whilst its DAG is always fixed. Similarly, a class of STCs which has the constraint of having the same number of free parameters as naive BNCs, and therefore has the same complexity, whilst being a much richer class of models then the naive BNC, is introduced.

Definition 35. *A Stratified Staged Tree Classifier such that, for every $k \leq p$, the set V_k is partitioned into $|\mathbb{C}|$ stages is called naive.*

It straightforwardly follows from the definition that Naive STCs have the same number of free parameters as Naive Bayes Classifiers.

Despite of the strict constraint on the number of parameters, the class of Naive STCs is still rich and extends naive BNCs in a non-trivial way. Differently to naive BNCs, it is not sufficient to simply learn the probabilities of the Naive Staged Tree Classifier, but also the staging structure has to be discovered. However, because of the strict restriction on the number of parameters, fast algorithms can be devised to efficiently explore the model space. Notice that in a binary classification problem the set V_k must be partitioned into two subsets.

	ST_Naive		RFor		NB	
	-1	1	-1	1	-1	1
-1	0.4623	0.0713	0.3972	0.3123	0.4314	0.3982
1	0.0347	0.4317	0.0998	0.1907	0.0656	0.1048

Table 6.1: Proportion of predicted instances in the simulated XOR example for the Naive Staged Tree Classifier (ST_Naive), Random Forest (RFor) and Naive Bayes (NB).

Critically and differently to Naive Bayes Classifiers, Naive Staged Trees are capable of representing complex decision rules. For instance, consider the simplest scenario of a binary class with two binary features. The Naive STC in Figure 6.4, which does not have a naive BNC representation [118], is capturing the only 2-XOR present.

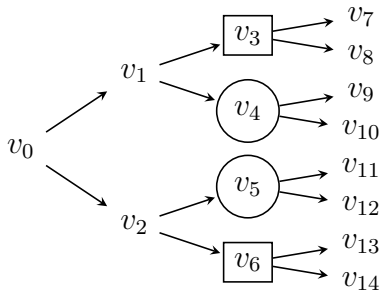


Figure 6.4: Example of Naive Staged Tree Classifier capturing a 2-XOR.

To investigate further the capabilities of Naive STCs in expressing complex decision rules, $N_{train} = 200$ observations are simulated from $p = 10$ binary variables $X_1, \ldots, X_p$ taking values in $\mathbb{X} = \{-1, +1\}^p$. The class variable is defined as the parity (or XOR) function $C = \prod_{i=1}^{p} X_i$ and Naive Bayes, Random Forest and Naive Staged Tree Classifiers are compared over $N_{test} = 10000$ test instances, obtaining the results in Tables 6.1. As expected, the Naive Bayes Classifier is unable to represent the parity function [118] and wrongly classifies the class in more than 40% of the test data. Similar performances are obtained by Random Forests (implemented with the *R* package **randomForest** using 500 trees), even if theoretically they have much larger expressive power. Conversely, the Naive Staged Tree correctly learns the parity function with an accuracy of 90%.

6.6 Classification Study

An experiment based on classification problem is conducted. The learning of the stage structure of a Staged Tree Classifier is obtained through the *R* package **stagedtrees**. The classification accuracy for binary classification of Staged Tree Classifiers is investigated in a comprehensive simulation study involving 14 datasets, whose details are given in Table 6.2. Each dataset is randomly divided ten times in train set (80% of the data) to learn the classifiers and test set (remaining 20%) to predict the response. The reported performance measures, area under the curve (AUC) and balanced accuracy, are computed as the mean over these ten replications.

Dataset	# observations	# variables	# atomic events
`Asym`	100	4	16
`BreastCancer`	683	10	1024
`chestSim500`	500	8	256
`energy1`	768	9	1728
`energy2`	768	9	1728
`FallEld`	5000	4	64
`fertility`	100	10	15552
`monks1`	432	7	864
`monks2`	432	7	864
`monks3`	432	7	864
`puffin`	69	6	768
`tic-tac-toe`	958	10	39366
`Titanic`	2201	4	32
`voting`	435	17	131072

Table 6.2: Details about the 14 datasets included in the classification study.

First, 9 model search algorithms for learning Staged Tree Classifiers are compared, namely:

- ST_Full, each vertex is in its own stage;
- ST_HC_Indep, hill-climbing algorithm starting from independence model;
- ST_HC_Full, hill-climbing algorithm starting from ST_Full;
- ST_BHC, backward hill-climbing;
- ST_FBHC, fast backward hill-climbing;
- ST_BJ_01, backward joining of vertices that have Kullback-Leibler divergence between their floret probability distributions less than 0.01;

- ST_BJ_20, as ST_BJ_01 but with threshold at 0.20;
- ST_Naive_HC, Naive Staged Tree learnt with hierarchical clustering;
- ST_Naive_KM, Naive Staged Tree learnt with kmeans.

Further details about these algorithms can be found in Section 5.2 or in Carli et al. [20].

Due to computational restrictions, for ST_HC_Full the model search is restricted to the first five features according to the variable ordering chosen through Conditional Mutual Information criterion (Algorithm 5), whilst for ST_BHC and ST_HC_Indep only the first seven are considered. The vertices corresponding to the remaining variables are still used for classification but left as in the starting tree of the model search.

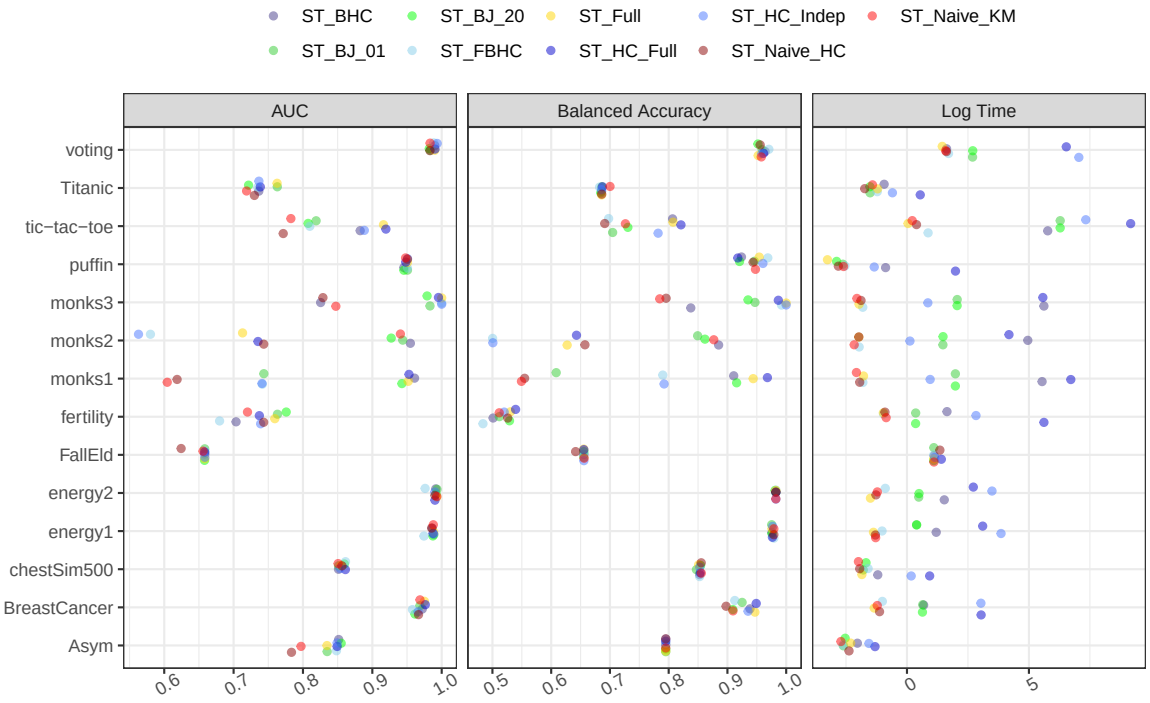


Figure 6.5: AUC, balanced accuracy and logarithm of time spent for structure learning for nine STCs algorithms over fourteen datasets.

The results of the experiment are reported in Figure 6.5, which suggests the following conclusions:

- The ST_Full model (in yellow), which does not require any model search and has the largest number of parameters, has in general lower AUC and balanced

accuracy than other Staged Trees. This highlights the need of a model-based search of simpler models;

- Models based on hill-climbing (in blue) overall perform better than others (in particular ST_HC_Full). This is expected since these are the most refined learning algorithms and, as a consequence, they are also the slowest.

- Models based on backward joining (in green) have a satisfactory performance, often comparable to that of hill-climbing models, whilst being much quicker to learn.

- Naive Staged Trees (in red) can be learnt extremely quickly and whilst often they have a lower performance, there are cases where they are comparable to the one of much more complex trees (see e.g. the balanced accuracy for the `Titanic` dataset)

Let mention that AUC is the area under the ROC curve [16] and the balanced accuracy is the average between the sensitivity and the specificity of predicted values. In turn, the sensitivity and specificity are calculated as the proportion of true positive and negative predicted values by the considered model, respectively. In particular, the balanced accuracy is useful when the outcomes of the binary response variable are imbalanced, i.e. one of the two categories appears a lot more often than the other.

Next, STCs are compared with their competitor generative classifier, namely BNCs. For ease of exposition, three representative Staged Trees are selected (ST_BJ_01, ST_HC_Full and ST_Naive_KM) and three BNCs are fitted: (i) the TAN BNC (BNC_TAN); (ii) the 3-dependence BNC (BNC_KDB); (iii) the Naive Bayes (BNC_NB). The results are reported in Figure 6.6. It can be seen that for most datasets there is one STC (in red) that outperforms BNCs (in blue). Due to the complexity of the models, Staged Trees are in general slower to learn, but the ST_Naive_KM, due to its simplicity, has learning times comparable to those of generic BNCs.

Figure 6.7 reports the results of the simulation experiments for three STCs (ST_BJ_01, ST_HC_Full and ST_Naive_KM) as well as other state of the art generative and discriminative classifiers, namely: (i) the Naive Bayes (BNC_NB); (ii) the TAN BNC (BNC_TAN); (iii) Classification Trees (CTree) (iv) Logistic Regression (Logistic); (v) Neural Networks with 20 hidden layers and 0.01 as weight decay (NNet); (vi) Random Forests combining 100 classification trees (RFor). Although in some cases discriminative classifiers (in green) outperform Staged Trees (in

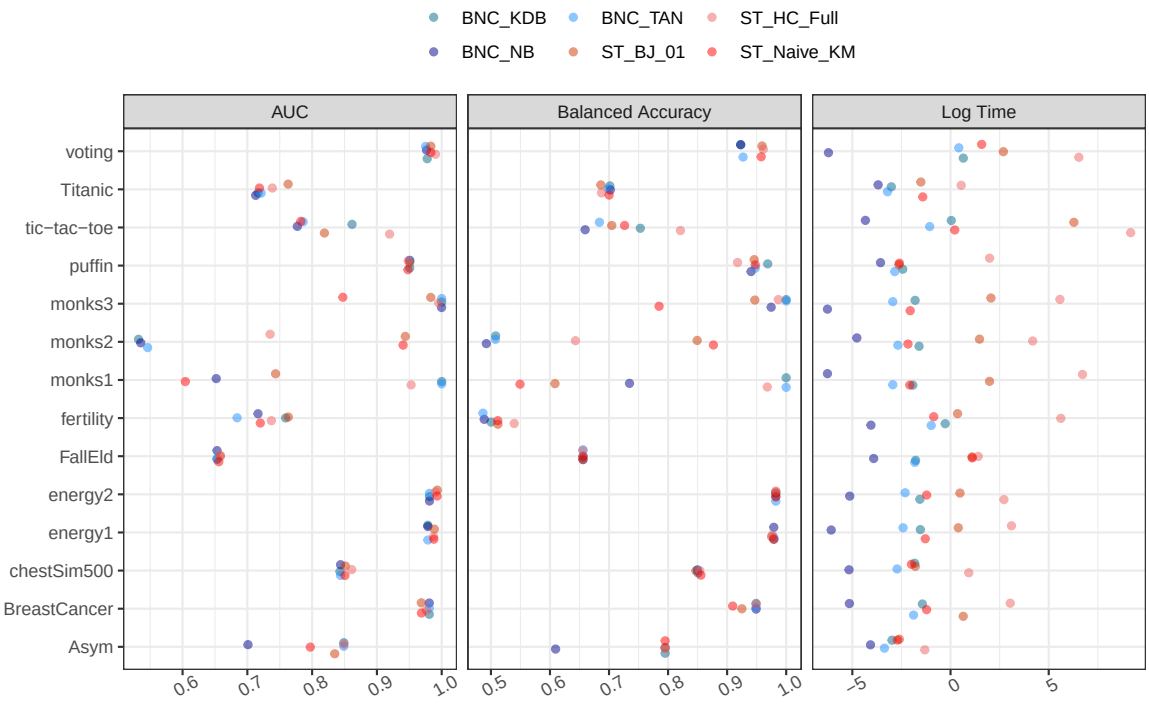


Figure 6.6: AUC, balanced accuracy and logarithm of time spent for structure learning for three STCs (in red) and three BNCs (in blue) over fourteen datasets.

red), in many others they have comparable AUC and balanced accuracy. However, as shown in the next section, STCs have the capability of producing an understanding of the relationship between the class and the features, since they are generative models. As already noticed, Staged Trees have an advantage over BNCs (in blue). Although the learning time for generic Staged Trees is larger, the learning time for Naive Staged Tree Classifiers is comparable to those of state of the art classifiers.

Last, the performance of Naive Bayes Classifiers are compared with the one of Naive Staged Tree Classifiers in Figure 6.8. The overall conclusion is that in most cases Naive Staged Trees (in red) outperform Naive Bayes in terms of AUC and balanced accuracy. Furthermore, although Naive Staged Trees require more learning time since their structure has to be discovered, these can be learnt very quickly and most times in less than one second.

6.6.1 An applied classification analysis

To illustrate the inference capabilities of STCs, next an applied classification analysis over the freely available `Titanic` dataset is developed. It has three binary variables (Survived, Sex and Age) and a categorical variable Class taking four levels (1st,

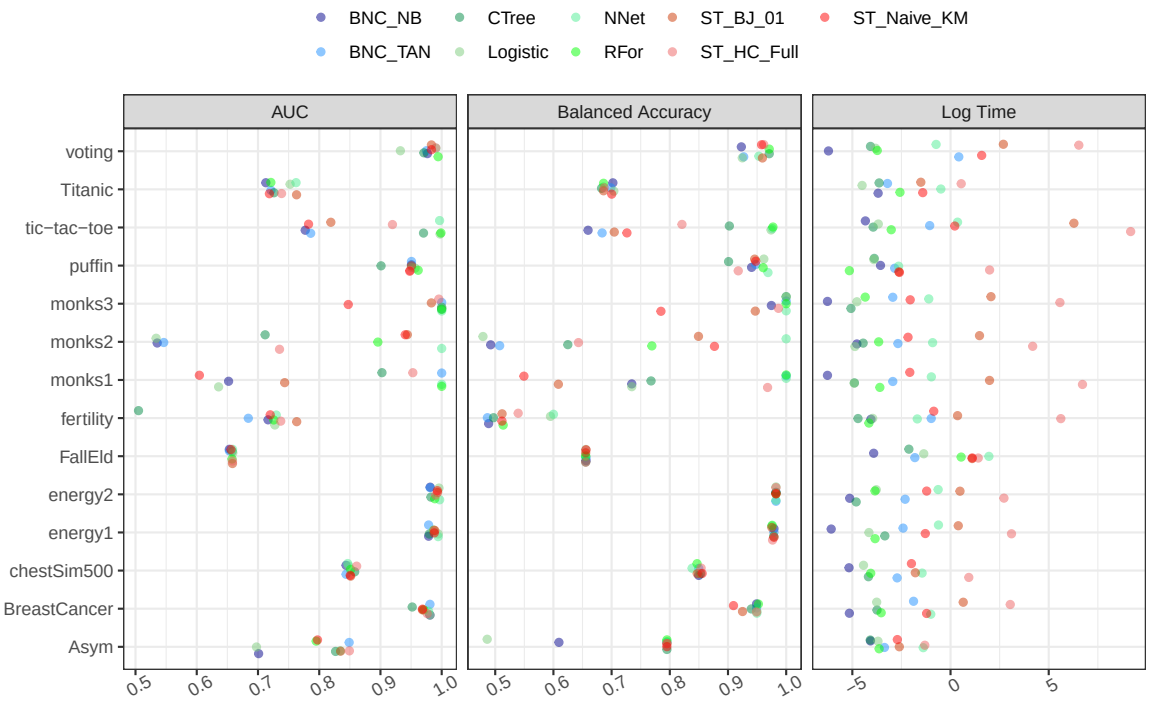


Figure 6.7: AUC, balanced accuracy and logarithm of time spent for structure learning for three STCs (in red), two BNCs (in blue) and other discriminative models (in green) over fourteen datasets.

2nd, 3rd and Crew). The aim is to correctly classify whether the Titanic passengers survived or not based on their gender, age and travelling class.

In Figure 6.5 it can be seen that one of the best STCs is the ST_BJ_01 learnt using a backward joining of the vertices based on the Kullback-Leibler divergence and a threshold of 0.1. Therefore, in Figure 6.9 this STC learnt over the full Titanic dataset using the *R* package **stagedtrees** is reported. By investigating the staging structure, one can deduce conditional independence statements relating to the classification variable (Survived) and the features. From stages associated to Class: $P(\text{Class} \mid \text{Sex} = \text{Male}, \text{Survived}) = P(\text{Class} \mid \text{Sex} = \text{Male})$ since the second and the fourth vertices (starting from the top) are in the same stage. This implies the context-specific conditional independence

$$\text{Class} \perp\!\!\!\perp \text{Survived} \mid \text{Sex} = \text{Male}.$$

The complex staging structure over the Age variable also implies asymmetric conditional independences. It can be noticed that all paths going through an edge labeled Crew are in the same stage for the variable Age. This implies the context-specific

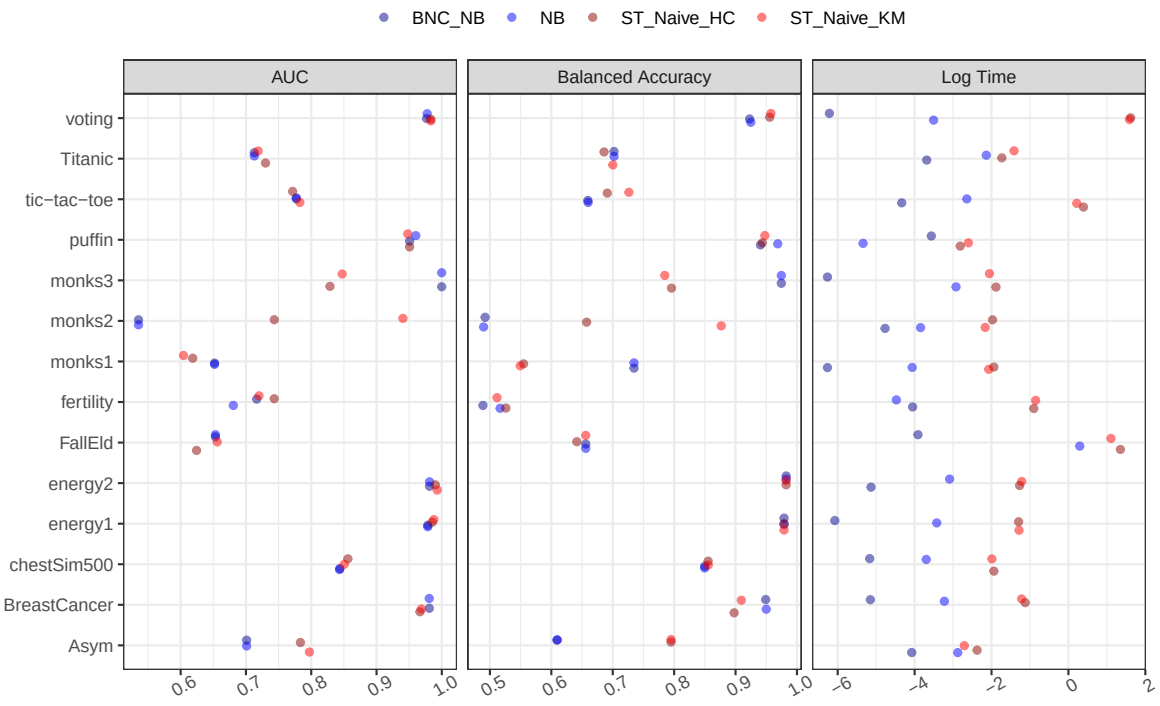


Figure 6.8: AUC, balanced accuracy and logarithm of time spent for structure learning for two Naive Staged Tree Classifiers (in red) and two implementations of Naive Bayes Classifiers (in blue) over fourteen datasets.

independence

$$\text{Age} \perp\!\!\!\perp \text{Survived} \mid \text{Class} = \text{Crew}.$$

The same statement can also be drawn for Class = 3rd.

As an additional illustration in Figure 6.10 the Naive STC learnt over the full Titanic dataset using the kmeans hierarchical clustering algorithm is reported. The staging structure over the variables Sex and Class implies that Sex and Survived are not independent and that Class is conditionally independent of Survived given Sex. The staging structure over the Age variable is a lot more complex describing highly asymmetric constraints on the associated probabilities. Whilst imposing much more flexible dependence structures, the Naive STC has the same complexity of the Naive Bayes Classifier, meaning they have the same number of independent parameters.

BNCs of different complexity are also learnt over the full `Titanic` dataset using the *R* package **bnclassify**. Irrespective of the complexity chosen, the model selection search always returns the simple Naive Bayes Classifier. The fact that Staged Tree Classifiers outperfom BNCs in classification measures for the Titanic dataset (see Figure 6.6), as well as for other datasets, highlights the need of asymmetric and

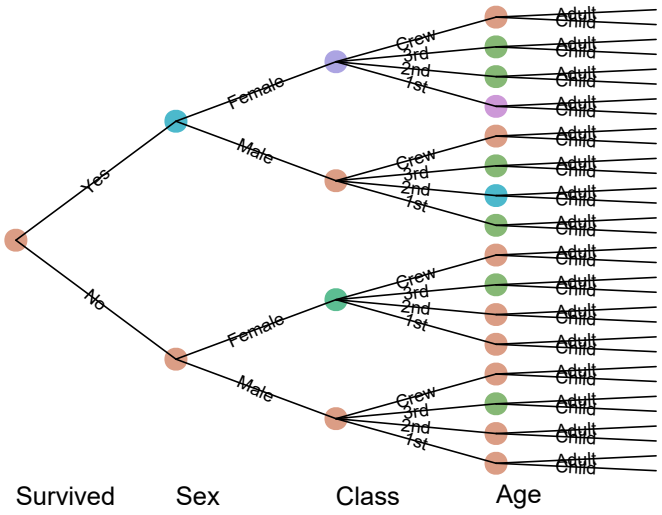


Figure 6.9: STC ST_BJ_01 learnt over the full `Titanic` dataset.

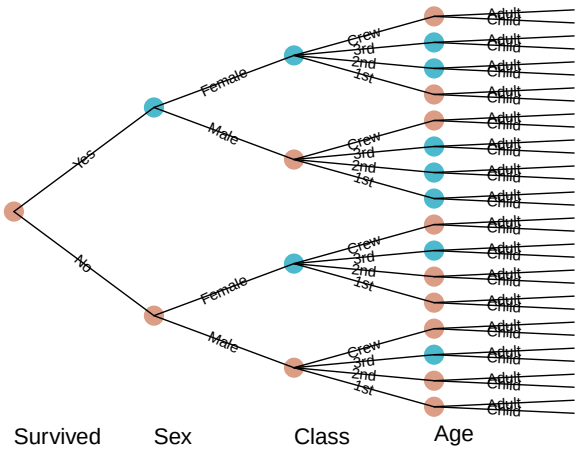


Figure 6.10: Naive STC learnt over the full `Titanic` dataset using the `stages_kmeans` algorithm.

context-specific generative classifiers that can more flexibly model the dependence structure between the classification variable and the features.

6.7 Conclusion

The idea to use Staged Tree for a classification purpose is an innovative proposal done in Carli et al. [21]. Staged Trees Classifiers are a highly-expressive new class of generative classifiers with classification performance comparable to that of state of the art classifiers. They embed context-specific conditional independence statements which can be easily read by the stage structure of the tree.

A special STC is the Naive Staged Tree Classifier which, whilst having the same complexity as the Naive Bayes Classifier, can flexibly represents sophisticated classification rule. Naive Staged Trees not only relax the assumption of conditional independence of the features as in Naive Bayes Classifiers, but also have better performances in classification, as highlighted by the simulation study.

Naive Staged Trees are learnt from data using a clustering algorithm of the probability distributions over the non-leaf vertices of the tree. Such algorithms divide the vertices at the same distance from the root in $|\mathbb{C}|$ stages. More generally, one could devise clustering algorithms were, for each variable, the number of stages is selected according to an optimality criterion. Furthermore, many model search algorithms implemented in **stagedtrees** are based on the maximization of a model score. A personal research currently under investigation is an algorithm based on the minimization of the classification error, as commonly implemented for BNCs.

Chapter 7

Handle Zero Counts with Staged Trees

This chapter presents some thoughts on one of the motivations behind Staged Trees, that is the presence of structural and observed zero counts in contingency tables. This is also one of the main problems that we encountered estimating Staged Trees on real data. This problem is investigated proposing a method to overcome this issue in the context of Staged Trees.

Section 7.1 shows through an example how predominant is the presence of vertices where no observation arrive starting from the root of the tree; this issue corresponds to observe a zero count in the corresponding contingency table. In Section 7.2 computational costs of algorithms in **stagedtrees** for estimating stage structure are investigated, by comparing their execution times if unobserved situations are left untouched or not. This is done with an analysis on four datasets. In Section 7.3 relational models [72, 73] are introduced, since they are defined on not Cartesian product sample spaces, thus permitting to model data with structural and observed zeros. The connection with Staged Trees is due to how relational models can efficiently manage conditional independence relationships. Indeed, when the sample space is an incomplete Cartesian product, the relational model can specify different variants of conditional independence in the parts of the sample space, depending on whether or not the part is or is not affected by the missing cells. In Section 7.3.2 it is shown that the Stratified Staged Trees are relational models with the overall effect.

7.1 Structural and Observed Zero Counts in Contingency Tables

One of the major problems in working on categorical variables is related to the possibility of finding contingency tables with structural and/or observed zero counts. This problem is extensively addressed in Agresti [1], where theory on categorical analysis about model's parameters estimation in presence of zeros is outlined. Instead, regarding the classical PGMs, as Undirected Graphs or Bayesian Networks, or Staged Trees and Chain Event Graphs, this topic is not handled in depth.

For all these reasons, in this section an intuitive idea on how to treat them in a Staged Trees framework is pointed out. It is also shown how much relevant is this problem for Staged Trees, which have the advantage of being capable to detect any asymmetric or context-specific conditional independence statements, with the disadvantage that the search space for model selection can be huge also with a not large number of variables. This is because, for a fixed order of variables, any probability distribution of a variable given any possible combination of the first ones that precede it in the tree is defined. This leads to the definition of a huge number of conditional contingency tables, which could have many cells with zero count since, adding each stratum of the tree a variable on which the distribution of the next one has to be conditioned on, it is even more probable to not observe any units for many combinations of levels of the variables.

The problem of zero count, also denoted in this framework with unobserved vertices/situations, has been extensively addressed also in **stagedtrees**, as already shown in previous chapters. Indeed, this is an interesting topic from both theoretical and computational points of view. It is reasonable that the model search space is restricted only to vertices in which, starting from the root of the tree, at least one observation of the collected data gets. This is because, if this restriction was not supposed, learning algorithms for Staged Trees would agglomerate randomly unobserved vertices to any other stage, being this step convenient from a modelling point of view, since the number of parameters that should be estimated would decrease. This is because the contribution to the likelihood function of a degenerate probability distribution associated to not observed vertices is null. Then, the randomly joining of unobserved situations to other stages is a coherent behaviour from a score-optimization (e.g. likelihood, AIC, BIC) point of view, which implies a reduction of the number of parameters but, nevertheless, one would probably prefer to isolate unobserved vertices for interpretation's sake. The researcher can choose whether to separate unobserved vertices from others. In Carli et al. [20] the recommendation is

to isolate them also in order to perform fast estimations of learning algorithms. This leads to a a huge reduction of the dimension of the model search space, especially with a large number of variables.

In the *R* package **stagedtrees** the possibility for the user to specify if he/she wants to separate in a own stage the unobserved vertices of the Staged Tree is included. A simple example can be seen in Figure 7.1, where these particular vertices are denoted with `"UNOBSERVED"`. In addition, all the implemented learning algorithms allow the specification of whether this set of vertices has to be untouched and excluded from the model search space. This is discussed also in Chapter 5. Another way in which zeros can be managed is to implement a Laplace smoothing for the probability distributions associated with the vertices of the tree. This can be done in the package by defining a lambda greater than zero, which permits to not have any degenerate probability distributions over the tree. For details see also Section 5.1.

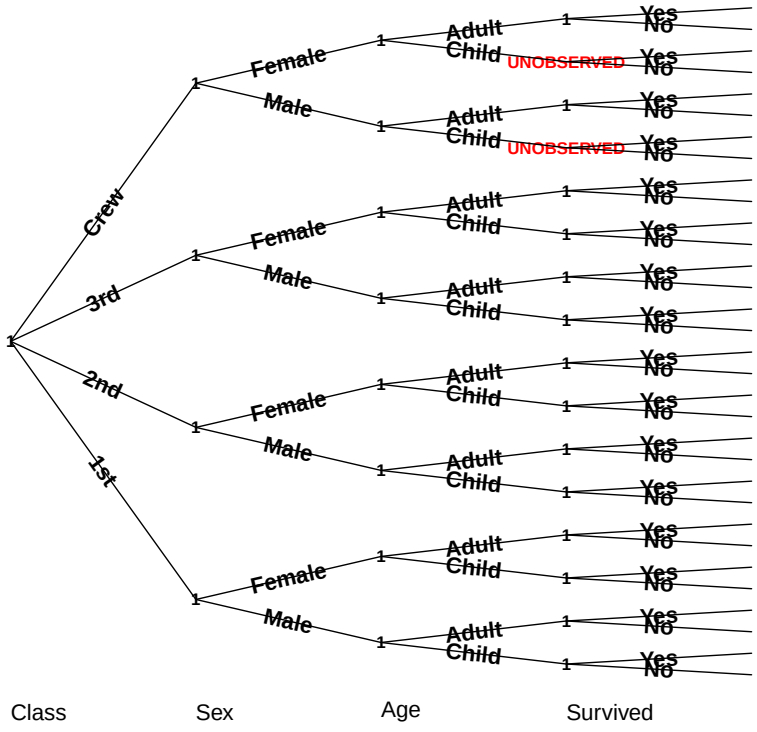


Figure 7.1: Stratified Staged Tree with two unobserved situations.

To clarify what introduced in this section, the number of unobserved vertices

on Stratified Staged Trees associated to twenty-two datasets, mostly of which are analyzed in Chapter 5, are calculated. In particular, the percentage of zero count vertices among all the vertices in the tree is computed. This is done stratum by stratum of the tree in order to show the behavior of this process as the number of involved variables increases, as shown in Figure 7.2. The structure of the trees is created according to the order of the variable in each dataset. The order of the variables is another important feature that is discussed in Section 3.7. The features of all datasets are not exhibited here since the aim of this section is only related to properties of unobserved situations in Staged Trees and not on performing inference on estimated models. The number of unobserved situations is clearly also related to the number of collected observations and the number of levels associated to the considered categorical variables. As a guideline, the datasets are mainly characterized by binary and ternary variables, while the units are mostly not more then one thousand. This example wants to be a simple empirical demonstration of the relevant presence of zero counts in contingency tables. Intuitively, from an algorithmic point of view, the computational burden of structural learning algorithms would extremely decrease lefting isolated the unobserved situations. In Section 7.2 an analysis on computational costs of algorithms in **stagedtrees** for estimating stage structure is carried out, by comparing their execution times if unobserved situations are left untouched or not.

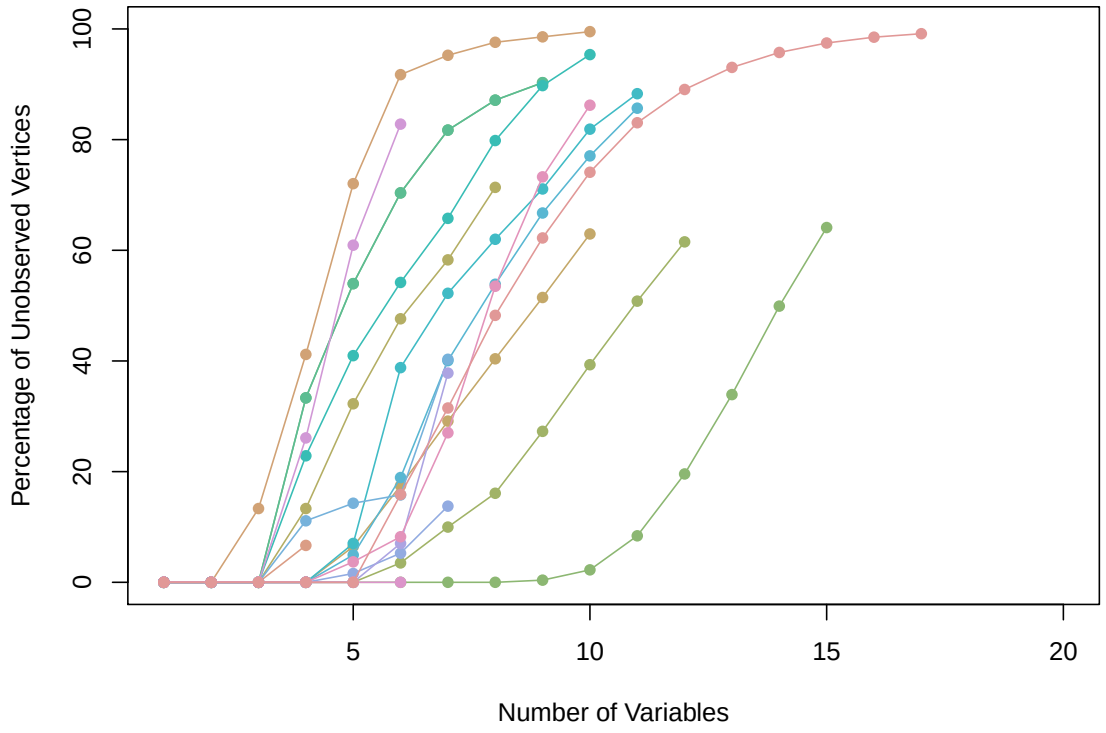


Figure 7.2: Percentage of unobserved vertices with respect to the number of involved variables. Each line corresponds to the behavior of the percentage of unobserved situations in a dataset, by adding one variable/stratum at time. Twenty-two datasets are considered.

7.2 Computational Burden of Algorithms whether or not Unobserved Situations are Isolated

In this section the computational costs of 14 algorithms from **stagedtrees** estimated on four datasets are recorded. The algorithms are mostly the whole of those implemented in **stagedtrees**, except the hill-climbing (`stages_hc`) and backward hill-climbing (`stages_bhc`) which are the slowest. The computational costs of each algorithm are recorded starting from considering two variables, adding one at time up to the number of variables in the corresponding dataset. Furthermore, the order of variables is chosen according to the conditional mutual information algorithm (Algorithm 5).

The whole experiment is repeated twice for each dataset, isolating or not the unobserved situations in a own stage for each stratum of the corresponding Stratified Staged Tree and keeping them untouched. Indeed, the aim of this section is to

show the extreme importance of the isolation of vertices associated to contingency tables with zero counts. The results are reported as ratio (R) between the cost of estimate the algorithm not separating the unobserved situations from others (M) and by isolating them ($M_{\text{UNOBSERVED}}$); its formulation is displayed in Equation (7.1). Hence, values of R larger than 1 correspond to how many times it takes the algorithm that does not treat zeros to estimate the stage structure of the tree with respect to the algorithm that isolates them.

$$R = \frac{\text{TIME}(M)}{\text{TIME}(M_{\text{UNOBSERVED}})} \tag{7.1}$$

Results obtained on `chestSim500`, `energy1`, `energy2` and `puffin` are reported in Table 7.1, 7.2, 7.3 and 7.4, respectively, where empty values on first columns stand for ratios that are zero or infinite due to extremely fast execution of algorithms on few strata of the tree. From these tables it can be seen that computational burden related to algorithms Full, Indep, Fast BHC, Random BHC, HClust with k = 2 and Kmeans is not really affected by the isolation of unobserved situations, since the corresponding R index is near to 1 for each dataset and number of considered variables. This is because these algorithms are extremely fast and they are not affected by the difference in executing M or $M_{\text{UNOBSERVED}}$, since the four datasets are also characterized by a not high number of variables. Conversely, the remaining algorithms, e.g. the distance-based ones with the eight distances/divergences implemented in the package, prove to be really slow if the zero counts are not investigated, showing computational times from 10 to 450 times higher than those obtained by isolating these vertices. The behavior of the R value for these algorithms has an exponentially trend with respect to the number of considered variables. This suggests that considering a dataset with few more variables than those considered in this study, as a dozen, the M algorithm would not be able to provide an estimate of the stage structure in a reasonable time. Instead, the $M_{\text{UNOBSERVED}}$ version of algorithms in **stagedtrees** can be applied in frameworks with more variables, for instance in Section 6.6 a dataset with 17 binary variables is analyzed (`voting`).

Furthermore, note that `stages_hc` and `stages_bhc` algorithms are not estimated here as the M algorithm take too long to run. Indeed, the difference on computational times would have been seen on these algorithms in an even more marked way than what happens for distance-based ones.

		# of Variables						
		2	3	4	5	6	7	8
Algorithm	Full	0.50	0.50	1.50	1.00	1.33	0.86	0.87
	Indep		1.00	0.50	2.00	0.75	0.67	1.00
	Fast BHC			0.67	0.60	0.57	0.87	1.27
	Random BHC		2.00	1.33	1.25	0.83	1.00	1.11
	Kullback-Leibler - 0.1		2.00	1.00	1.00	1.80	4.87	44.22
	Manhattan - 0.1		1.50	5.00	1.75	2.40	4.75	16.25
	Euclidean - 0.1		1.00	1.00	1.00	1.40	2.71	9.50
	Renyi - 0.1			1.50	1.50	1.80	5.38	27.44
	Total Variation - 0.1			3.00	3.00	1.40	2.67	7.80
	Hellinger - 0.1	0.50	1.00	1.00	1.00	1.00	3.17	11.33
	Bhattacharyya - 0.1			1.00	1.33	0.80	2.83	9.00
	Chan-Darwiche - 0.1			1.00	0.67	1.50	5.00	21.78
	HClust k = 2		1.00	0.25	1.00	1.25	1.33	1.25
	Kmeans		1.00	1.00	1.00	0.75	0.75	0.92

Table 7.1: Computational costs of 14 algorithms estimated on `chestSim500` dataset by adding one variable at a time. All the algorithms are estimated according to the variables ordering given by the conditional mutual information.

		# of Variables							
		2	3	4	5	6	7	8	9
Algorithm	Full		1.00	1.00	0.50	1.00	1.00	0.90	1.21
	Indep		1.00	0.33	1.33	0.60	0.83	0.62	1.14
	Fast BHC		1.00	0.50	1.00	1.33	1.86	1.85	2.56
	Random BHC	3.00	0.67	1.00	1.00	0.88	1.00	1.00	1.12
	Kullback-Leibler - 0.1	2.00		2.00	1.25	14.67	33.80	138.42	407.92
	Manhattan - 0.1			2.00	0.75	3.40	12.50	48.13	326.67
	Euclidean - 0.1			1.00	1.67	5.50	11.00	84.33	368.84
	Renyi - 0.1		2.00	0.40	1.40	17.00	43.89	156.53	448.05
	Total Variation - 0.1		1.00	0.33	1.00	3.00	13.25	65.62	243.83
	Hellinger - 0.1		1.00	1.00	2.50	5.40	21.00	78.21	335.82
	Bhattacharyya - 0.1	0.50		1.50	1.00	3.00	10.50	52.83	343.41
	Chan-Darwiche - 0.1			0.50	5.00	7.50	31.71	122.29	382.06
	HClust k = 2		0.50	1.33	1.00	0.80	1.83	1.27	1.56
	Kmeans		2.00	0.33	0.75	1.00	1.33	1.10	1.59

Table 7.2: Computational costs of 14 algorithms estimated on `energy1` dataset by adding one variable at a time. All the algorithms are estimated according to the variables ordering given by the conditional mutual information.

		# of Variables							
		2	3	4	5	6	7	8	9
Algorithm	Full			1.00	0.67	1.67	1.00	1.14	1.00
	Indep		2.00	0.67	1.00	0.83	0.67	1.44	0.68
	Fast BHC	2.00	1.00	1.67	0.56	2.43	1.50	1.57	2.21
	Random BHC	1.00	0.33	1.00	0.80	1.00	1.00	0.92	0.80
	Kullback-Leibler - 0.1			0.50	1.67	8.80	40.80	134.20	156.02
	Manhattan - 0.1		0.67	0.67	0.50	1.56	4.53	18.76	93.03
	Euclidean - 0.1			1.00	0.67	2.14	14.71	44.00	224.96
	Renyi - 0.1		2.00	0.33	1.50	9.60	28.14	116.72	302.85
	Total Variation - 0.1			1.00	1.00	9.33	10.45	57.60	267.21
	Hellinger - 0.1			2.00	0.75	4.20	18.00	82.67	281.00
	Bhattacharyya - 0.1		1.00	0.67	1.33	4.33	14.00	63.46	219.59
	Chan-Darwiche - 0.1		2.00	2.00	1.25	11.00	51.00	170.93	292.25
	HClust k = 2			0.50	1.00	1.25	0.86	0.92	0.96
	Kmeans			0.20	1.00	0.60	1.00	1.10	1.22

Table 7.3: Computational costs of 14 algorithms estimated on `energy2` dataset by adding one variable at a time. All the algorithms are estimated according to the variables ordering given by the conditional mutual information.

		# of Variables				
		2	3	4	5	6
Algorithm	Full				0.67	0.75
	Indep			0.50	0.50	1.00
	Fast BHC			2.00	0.75	1.67
	Random BHC		1.00	0.67	0.60	0.71
	Kullback-Leibler - 0.1		0.50	2.50	5.33	68.60
	Manhattan - 0.1			3.00	2.20	29.83
	Euclidean - 0.1		0.50		2.00	34.67
	Renyi - 0.1			5.00	5.17	84.38
	Total Variation - 0.1			0.67	3.67	30.00
	Hellinger - 0.1			4.00	4.00	45.20
	Bhattacharyya - 0.1			3.00	2.75	30.67
	Chan-Darwiche - 0.1			2.50	7.33	83.67
	HClust k = 2			1.00	1.00	0.80
	Kmeans			2.00	0.67	1.20

Table 7.4: Computational costs of 14 algorithms estimated on `puffin` dataset by adding one variable at a time. All the algorithms are estimated according to the variables ordering given by the conditional mutual information.

7.3 Relational Models

In this section relational models are described. After introducing definitions, propositions and theorems that help to understand this class of models, in the final part of the section the relationships that have been found between these models and the Staged Trees will be presented. This link represents a novelty in the state of the

art. In this regard, the research carried out on this topic is certainly still at the beginning and many other things will be done in the near future.

Relational models are log-linear models defined on arbitrary discrete sample spaces by specifying effects associated with any subsets of their cells, even if these spaces have not a Cartesian product structure [72, 73]. Let $X_1, \ldots, X_p$ be random variables taking values in finite sets $\mathcal{X}_1, \ldots, \mathcal{X}_p$ and denote with $\mathcal{I}$ the sample space, defined as a non-empty subset of $\mathcal{X}_1 \times \cdots \times \mathcal{X}_p$ and written as a sequence of length $I = |\mathcal{I}|$ in the lexicographic order.

Definition 36. *A statistical model is called relational model if it can be written as*

$$log\ \boldsymbol{\delta} \ = \ \boldsymbol{A}^T\ \boldsymbol{\theta}, \tag{7.2}$$

where:

- $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_I)$*, with* $\delta_i \in (0, 1)$*, for* $i = 1, \ldots, I$*, the parameter associated to cell* $i \in \mathcal{I}$ *and* $\sum_{i=1}^{I} \delta_i = 1$*;*
- $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_J)^T$ *are the log-linear parameters of the model;*
- *the model design matrix* $\boldsymbol{A}$ *with dimension* $J \times I$ *is a* $0-1$ *full row rank matrix with at least one* 1 *in each column.*

Then, we say that the model design matrix $\boldsymbol{A}$ *defines the relational model* $RM(\boldsymbol{A})$*.*

The parameters $\boldsymbol{\delta}$ can be understood as probability parameters of Multinomial distributions, $\delta_i = p_i$, or, for count data, as parameters of Poisson distributions, $\delta_i = \lambda_i$.

Proposition 11. *The number* J *of rows of the model design matrix* $\boldsymbol{A}$ *is less or equal of its number* I *of columns.*

Proof. For the definition of a design matrix of a relational model, $\boldsymbol{A}$ is a full row rank matrix (J). But, for the definition of the rank of a matrix we also have that

$$J \ = \ rank(\boldsymbol{A}) \ \leq \ min(J,\ I),$$

which implies that $J \leq I$. □

Proposition 11 states that super-saturated (over parameterized) models can not be written as relational models, unless they are reparameterized.

Definition 37. *Relational models which have a normalizing constant/grand mean/intercept for all the cells* $i \in \mathcal{I}$ *are said relational models with the overall effect.*

Proposition 12. *Let* $\mathbf{1}^T = (1, \ldots, 1)$ *be the row of* 1*'s of length* I*. Then, if* $\mathbf{1}^T$ *belongs to the space spanned by the rows of* $\boldsymbol{A}$*, the relational model* $RM(\boldsymbol{A})$ *is said to be a model with an overall effect.*

From Proposition 12 it follows that if $\mathbf{1}^T$ does not belong to the space spanned by the rows of $\boldsymbol{A}$, the $RM(\boldsymbol{A})$ is said to be a model without the overall effect. This occurs only if $2 \leq J = rank(\boldsymbol{A}) < I - 1$.

Theorem 4. *If the sum of the rows of the design matrix* $\boldsymbol{A}$ *is a constant vector* $\boldsymbol{c}^T = (c, \ldots, c)$*, with* $c \neq 0$*, then* $\mathbf{1}^T$ *belongs to the row space of* $\boldsymbol{A}$ *and the relational model* $RM(\boldsymbol{A})$ *contains the overall effect.*

Proof. Let a_i^T be the i-th row of $\boldsymbol{A}$. If $\sum_{i=1}^J a_i^T = \boldsymbol{c}^T$, then it follows that

$$\sum_{i=1}^{J} a_i^T = c\, \mathbf{1}^T \qquad \Longrightarrow \qquad \frac{1}{c} \sum_{i=1}^{J} a_i^T = \mathbf{1}^T,$$

that is the 1's vector can be obtain as a linear combination of the rows of $\boldsymbol{A}$. □

Theorem 4 gives only a sufficient condition for verifying whether an overall effect in a relational model is included or not. Consider for instance the following design matrix $\boldsymbol{A}$

$$\boldsymbol{A} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix},$$

which does not satisfy that theorem, since its rows sum is the vector $(2, 1, 1)$, but its row space contains the 1's vector.

Theorem 5. *Let* $\boldsymbol{A}$ *be a* $J \times I$ *matrix as in Definition 36 and* $\boldsymbol{D}$ *a basis of* $Ker(\boldsymbol{A})$*. Then, a dual representation of the relational model* $RM(\boldsymbol{A})$ *can be obtained through* $\boldsymbol{D}$*: log* $\boldsymbol{\delta} = \boldsymbol{A}^T\boldsymbol{\theta}$ *holds if and only if* $\boldsymbol{D}$ *log* $\boldsymbol{\delta} = 0$*.*

Proof. It follows from the definition of the kernel of a matrix. □

The number of the degrees of freedom K of the relational model is equal to *dim(Ker(*$\boldsymbol{A}$*))*. Denote with $d_1^T, \ldots, d_K^T$ the rows of $\boldsymbol{D}$. The dual representation can be expressed in terms of generalized odds ratios or cross-product differences, displayed respectively in Equation (7.3) and (7.4), where $\boldsymbol{d}^+$ and $\boldsymbol{d}^-$ stands for the positive and the negative parts of a vector $\boldsymbol{d}$. By convention, $\boldsymbol{p}^{\boldsymbol{d}} = p_1^{d_1} p_2^{d_2} \ldots p_I^{d_I}$.

$$\frac{\boldsymbol{p}^{\boldsymbol{d_1^+}}}{\boldsymbol{p}^{\boldsymbol{d_1^-}}} = 1, \qquad \frac{\boldsymbol{p}^{\boldsymbol{d_2^+}}}{\boldsymbol{p}^{\boldsymbol{d_2^-}}} = 1, \quad \ldots, \quad \frac{\boldsymbol{p}^{\boldsymbol{d_K^+}}}{\boldsymbol{p}^{\boldsymbol{d_K^-}}} = 1 \tag{7.3}$$

$$\boldsymbol{p}^{\boldsymbol{d_1^+}} - \boldsymbol{p}^{\boldsymbol{d_1^-}} = 0, \qquad \boldsymbol{p}^{\boldsymbol{d_2^+}} - \boldsymbol{p}^{\boldsymbol{d_2^-}} = 0, \quad \ldots, \quad \boldsymbol{p}^{\boldsymbol{d_K^+}} - \boldsymbol{p}^{\boldsymbol{d_K^-}} = 0 \tag{7.4}$$

A generalized odds ratio in Equation (7.3) is called homogeneous if the sum of the components of $\boldsymbol{d}^+$ is the same as the sum of the components of $\boldsymbol{d}^-$, and non-homogeneous otherwise.

Proposition 13. *A relational model $RM(\boldsymbol{A})$ does not contain the overall effect if and only if*

1. *among the generalized odds ratios defining $RM(\boldsymbol{A})$, there is at least one non-homogeneous odds ratios;*

2. *there exists a kernel basis matrix $\boldsymbol{D}$ whose rows satisfy:*

$$d_1^T\ 1 \neq 0, \qquad d_2^T\ 1 = 0, \quad \ldots, \quad d_K^T\ 1 = 0.$$

Proof. According to Definition 13, $\boldsymbol{D}$ has at least one row, say $\boldsymbol{d}_1^T$, that is not orthogonal to 1: $C_1 = \boldsymbol{d}_1^T \neq 0$. Suppose there exists another row, say $\boldsymbol{d}_2^T$, that is not orthogonal to 1 and thus $C_2 = \boldsymbol{d}_2^T \neq 0$. Then, the vectors $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ are linear independent as the vectors $\boldsymbol{d}_1$ and $C_2\boldsymbol{d}_1 - C_1\boldsymbol{d}_2$. Note that $(C_2\boldsymbol{d}_1 - C_1\boldsymbol{d}_2)^T 1 = 0$; substitute the row $\boldsymbol{d}_2^T$ with the row $C_2\boldsymbol{d}_1^T - C_1\boldsymbol{d}_2^T$ and apply the same transformation with appropriate C's to the remaining rows, if needed. □

7.3.1 Relational Model Structure with and without the Overall Effect

This section deals with adding the overall effect to a relational model which does not include it and with removing it when it is included. The consequences of adding or removing the overall effect is studied separately.

Include the Overall Effect in a Relational Model

Let $RM(\boldsymbol{A})$ be a relational model without the overall effect and $RM(\bar{\boldsymbol{A}})$ be the corresponding augmented model, with $\bar{\boldsymbol{A}}$ the augmented matrix:

$$\bar{\boldsymbol{A}} = \begin{pmatrix} \mathbf{1}^T \\ \boldsymbol{A} \end{pmatrix}.$$

Then, the following two Theorems hold. For their proofs, see Klimova et al. [73].

Theorem 6. *The dual representation of $RM(\bar{\boldsymbol{A}})$ can be obtained from the dual representation of $RM(\boldsymbol{A})$ by removing the constraint specified by the non-homogeneous odds ratio from the latter.*

Theorem 7. *The following holds:*

1. $RM(\bar{\boldsymbol{A}})$ *is a regular exponential family;*

2. $RM(\boldsymbol{A}) \subseteq RM(\bar{\boldsymbol{A}})$. *In particular,* $RM(\boldsymbol{A}) = \{\boldsymbol{\delta} \in RM(\bar{\boldsymbol{A}}) : \ \theta_0(\boldsymbol{\delta}) = 0\}$, *with* $\theta_0(\boldsymbol{\delta})$ *the overall effect of* $\boldsymbol{\delta}$*;*

3. $RM(\bar{\boldsymbol{A}})$ *is minimal, in the sense that any regular exponential family containing* $RM(\boldsymbol{A})$ *also contains* $RM(\bar{\boldsymbol{A}})$.

Example 2. *The relational models generated by the matrices*

$$\boldsymbol{A} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \qquad \bar{\boldsymbol{A}} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

consist of positive probability distribution which can be written in the following parametric forms:

$$\begin{cases} p_1 = \alpha_1 \\ p_2 = \alpha_1 \\ p_3 = \alpha_1 \ \alpha_2 \\ p_4 = \alpha_2 \end{cases} \qquad \begin{cases} p_1 = \beta_0 \ \beta_1 \\ p_2 = \beta_0 \ \beta_1 \\ p_3 = \beta_0 \ \beta_1 \ \beta_2 \\ p_4 = \beta_0 \ \beta_2, \end{cases}$$

where β_0 *is the overall effect. The dual representation can be written in the log-linear form, using* $d_1 = (-1, 0, 1, -1)^T \in Ker(\boldsymbol{A})$, *and* $d_2 = (-1, 1, 0, 0)^T \in Ker(\boldsymbol{A}) \cap Ker(\bar{\boldsymbol{A}})$*:*

$$\begin{cases} d_1^T \ log \ p = 0 \\ d_2^T \ log \ p = 0 \end{cases} \qquad \begin{cases} d_2^T \ log \ p = 0. \end{cases}$$

By Theorem 6, after that the overall effect is added, the model specification does not include the non-homogeneous constraint anymore. In terms of the generalized odds ratios:

$$\begin{cases} \frac{p_3}{p_1 \ p_4} = 1 \\ \frac{p_1}{p_2} = 1 \end{cases} \qquad \begin{cases} \frac{p_1}{p_2} = 1 \end{cases}$$

Remove the Overall Effect in a Relational Model

A relational model with the overall effect can be reparameterized such that its design matrix has a row of 1's, and because of full row rank, this vector is not spanned by the other rows.

The implication of the removal of the overall effect is investigated using a design matrix of that structure, call it $\bar{\boldsymbol{A}}_1$. By the removal of the row $\mathbf{1}^T$, one may obtain a different design matrix on the same sample space, but it may happen that there exists a cell i_0, whose only parameter is the overall effect, and after its removal, the i_0-th column contains zeros only. In such cases, to have a proper design matrix, such columns, that is such cells, need to be removed. Denote with $\mathcal{I}_0$ the set of all cells i_0 and let $I_0 = |\mathcal{I}_0|$. Then, the reduced design matrix $\mathbf{A}_1$ is obtained from $\bar{\boldsymbol{A}}_1$ after removing the row of 1's and deleting the columns which, after this, contain only zeros. This is a design matrix on $\mathcal{I} \setminus \mathcal{I}_0$. Using this terminology, the matrix $\bar{\boldsymbol{A}}_1$ can be written as

$$\bar{\boldsymbol{A}}_1 = \begin{pmatrix} \mathbf{1}^T_{(I \setminus I_0)} & \mathbf{1}^T_{I_0} \\ \boldsymbol{A}_1 & \boldsymbol{O}_{(J-1)\times I_0} \end{pmatrix}.$$

If the sample spaces of $RM(\bar{\boldsymbol{A}}_1)$ and $RM(\boldsymbol{A}_1)$ are the same, i.e. when $\mathcal{I}_0$ is empty, the reduced model is the subset of the original one, consisting of the distributions whose overall effect is zero (see Theorem 7). If the sample space is reduced, the relationship between the kernel basis matrices is described in Theorem 8. For its proof see Klimova et al. [73].

Theorem 8. *The following holds:*

1. $dim(Ker(\boldsymbol{A}_1)) = dim(Ker(\bar{\boldsymbol{A}}_1)) - I_0 + 1$;

2. *the kernel basis matrix $\boldsymbol{D}_1$ of $\boldsymbol{A}_1$ may be obtained from the kernel basis matrix $\bar{\boldsymbol{D}}_1$ of $\bar{\boldsymbol{A}}_1$ by deleting the columns in I_0 and then leaving out the redundant rows.*

Example 3. *Let $RM(\bar{\boldsymbol{A}}_1)$ be the relational model generated by*

$$\bar{\boldsymbol{A}}_1 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}.$$

The corresponding system of equations are:

$$\begin{cases} p_1 = \alpha_0\ \alpha_1 \\ p_2 = \alpha_0\ \alpha_2 \\ p_3 = \alpha_0\ \alpha_1\ \alpha_2 \\ p_4 = \alpha_0\ \alpha_2 \\ p_5 = \alpha_0 \\ p_6 = \alpha_0 \end{cases} \Longrightarrow \begin{cases} log\ \delta_1 = \theta_0 + \theta_1 \\ log\ \delta_2 = \theta_0 + \theta_2 \\ log\ \delta_3 = \theta_0 + \theta_1 + \theta_2 \\ log\ \delta_4 = \theta_0 + \theta_2 \\ log\ \delta_5 = \theta_0 \\ log\ \delta_6 = \theta_0 \end{cases}.$$

Hence, $\mathcal{I}_0 = \{5, 6\}$*, since the* 5*-th and* 6*-th columns of* $\bar{\boldsymbol{A}}_1$ *have only zeros, after the elimination of the first row which has only* 1*'s, corresponding to the overall effect. In terms of the generalized odds ratios, the model can be written as:*

$$\begin{cases} \frac{p_3\ p_5}{p_1\ p_2} = 1 \\ \frac{p_3\ p_6}{p_1\ p_2} = 1 \\ \frac{p_2}{p_4} = 1 \end{cases}$$

Removing the row $\mathbf{1}^T$ *and the last two columns corresponding to* $\mathcal{I}_0$ *leads to the definition of the reduced design matrix* $\boldsymbol{A}_1$*:*

$$\boldsymbol{A}_1 = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}.$$

The model $RM(\boldsymbol{A}_1)$ *does not have the overall effect and can be specified by two generalized odds ratios:*

$$\begin{cases} \frac{p_3}{p_1\ p_4} = 1 \\ \frac{p_2}{p_4} = 1 \end{cases}$$

These odds ratios are defined on the smaller probability space and may be obtained by removing p_5 *and* p_6 *and the redundant odds ratio from the ones specified from the original model characterized by* $\bar{\boldsymbol{A}}_1$*.*

7.3.2 Relationship between Relational Models and Staged Trees

In this section the relationship between Staged Trees and relational models is investigated. These two classes of models for categorical data are intersected but are not identical, as shown in the following two examples, where the first shows that not all

relational models can be seen as Staged Trees and the second that not all Staged Trees are relational models.

Example 4. ***Relational Model which is not a Staged Tree.*** *In Kawamura et al. [67], three types of baits were used in traps to catch crabs: fish alone, sugarcane alone, fish-sugarcane combination. The sample space consists of three cells,* $\mathcal{I} = \{(0,1),(1,0),(1,1)\}$*, and the cell* $(0,0)$ *is absent by design, because there were no traps without bait. Under the AS independence (Klimova & Rudas, 2015), the cell parameter associated with the other two cells. This is a relational model without the overall effect, generated by the matrix*

$$\boldsymbol{A} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}.$$

The overall effect is not included in the model defined by $\boldsymbol{A}$*. We also have that there does not exist a Staged Tree corresponding to the model defined by that matrix.*

Example 5. ***Staged Tree which is not a Relational Model.*** *Example of coin tossing: suppose that a coin is flipped once and only if the outcome is heads it is flipped again. Let be* ρ_1 *the probability of heads and* $\rho_2 = 1 - \rho_1$ *the probability of tails. This experiment can be represented by the Staged Tree in Figure 7.3.*

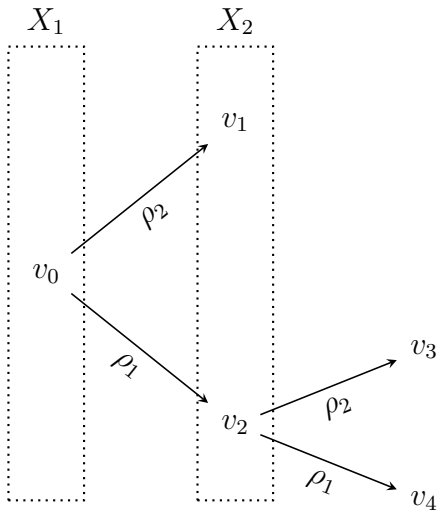


Figure 7.3: Staged Tree which is not a relational model.

$$\begin{cases} p_1 = \rho_2 \\ p_2 = \rho_1\ \rho_2 \\ p_3 = \rho_1^2 \end{cases} \quad \Longrightarrow \quad \begin{cases} log\ \delta_1 = \theta_2 \\ log\ \delta_2 = \theta_1 + \theta_2 \\ log\ \delta_3 = 2\ \theta_2 \end{cases}.$$

The design matrix $\boldsymbol{A}$ *corresponding to this model is:*

$$\boldsymbol{A} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 2 \end{pmatrix},$$

which is not a $0-1$ *matrix.*

Definition 38. *The model design matrix* $\boldsymbol{A}$ *associated to a Staged Tree is built according to its root-to-leaf paths. The number of root-to-leaf paths corresponds to the number of columns in* $\boldsymbol{A}$*, while the number of distinct probability parameters associated to floret distributions of the whole tree characterizes the number of rows in* $\boldsymbol{A}$*.*

For an example of the construction of the design matrix $\boldsymbol{A}$ associated to a Staged Tree see Example 6. Note that the probability distribution associated to each root-to-leaf path of the tree can be obtained as the product of the floret probability parameters along the considered root-to-leaf path. Then, Definition 39 gives a characterization of a type of Staged Trees we are interested on.

Definition 39. *Staged Trees are said multilinear if and only if they do not have repeated parameters in each root-to-leaf path.*

Proposition 14. *Multilinear Staged Trees can be formalized through a design matrix* $\boldsymbol{A}$ *with* $0-1$ *values.*

Proof. The proof follows from the definition of a design matrix for a multilinear Staged Tree. Each root-to-leaf path is the product of distinct parameters and this implies that each row of $\boldsymbol{A}$ has only zeros or ones. A value greater than 1 would be present in $\boldsymbol{A}$ only if a floret probability parameter were repeated in a root-to-leaf path, but, for definition, this is not possible for multilinear Staged Trees. □

For instance, Staged Tree in Figure 7.3 is not multilinear, since p_3 is obtained as the square of ρ_1. From this it follows that Staged Trees which are not multilinear have a corresponding design matrix $\boldsymbol{A}$ which does not contain only 0 or 1, leading then to a not relational model. For this reason, next only multilinear Staged Trees will be considered. Note that Stratified Staged Trees are multilinear, since probability distributions can be defined through the same parameters vector only for vertices in the same stratum of the tree.

Proposition 15. *Stratified Staged Trees are relational model* $RM(\boldsymbol{A}^*)$*, with* $\boldsymbol{A}^*$ *a basis of the vector space generated by the rows of the matrix* $\boldsymbol{A}$ *built according the root-to-leaf paths of the tree.* $\boldsymbol{A}^*$ *has a number of rows smaller or equal of the number of columns* ($J \leq I$)*.*

Proof. The original model design matrix $\boldsymbol{A}$ corresponding to the parametrization deduced from the Stratified Staged Tree defines a saturated model. This is for the sum up to one constraints for each floret distributions, which are not explicitly considered in $\boldsymbol{A}$. This issue is overcome by obtaining a new design matrix $\boldsymbol{A}^*$ as a basis of the vector space generated by the rows of the original $\boldsymbol{A}$. The so-built $\boldsymbol{A}^*$ has a number of rows smaller or equal of the number of its columns ($J \leq I$). □

Corollary 2. *Let $\mathcal{T} = (V, E)$ be a multilinear Staged Tree. Then, $\mathcal{T}$ is a relational model containing an overall effect.*

Proof. If the model design matrix $\boldsymbol{A}$ built according to root-to-leaf paths in $\mathcal{T}$ is not full row rank, through Proposition 15 a full row rank matrix $\boldsymbol{A}^*$ can be always obtained. Then, the model defined by $\boldsymbol{A}^*$ is the relational model $RM(\boldsymbol{A}^*)$.

Denote now with v_0 the root vertex of $\mathcal{T}$ and $\mathcal{F}(v_0) = (v_0, E(v_0))$ the corresponding floret, with $k_0 = |E(v_0)|$ the number of outgoing edges from v_0. Then, the space spanned by the first k_0 rows of the original design matrix $\boldsymbol{A}$ contains the 1's vector. This is because each root-to-leaf path of a multilinear Staged Tree contains exactly one $e_0 \in E(v_0)$. But, for construction, also the row space spanned by $\boldsymbol{A}^*$ contains that vector, since $\boldsymbol{A}^*$ is a basis of the vector space generated by the rows of the original $\boldsymbol{A}$. □

Example 6. *Consider the Stratified Staged Tree associated to an independence model between two variables, X_1 and X_2, whose graphical representation is depicted in Figure 7.4.*

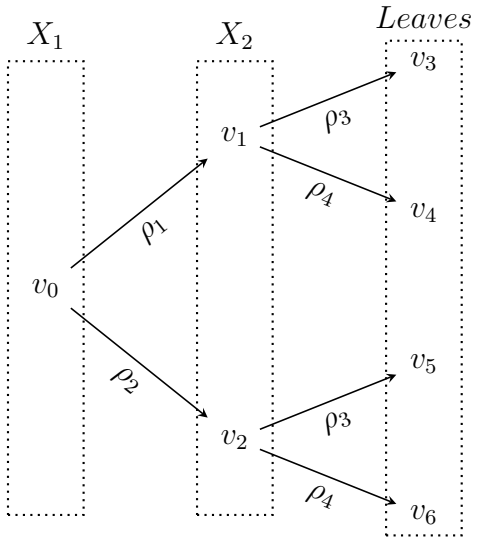


Figure 7.4: Stratified Staged Tree representing marginal independence between X_1 and X_2.

Denote with 0 *and* 1 *the possible outcomes for* X_1 *and* X_2*, then the sample space is* $\mathcal{I} = \{0,1\}^2$*. Vertices* v_3, v_4, v_5 *and* v_6 *are the leaves of the tree and they correspond to the joint probability* $\mathbb{P}(X_1 = x_1, X_2 = x_2)$*, for* $(x_1, x_2) \in \mathcal{I}$*. Let* p_1, p_2, p_3 *and* p_4 *be the probabilities associated to the leaves, respectively from the top to the bottom of the tree. Hence, the corresponding joint probabilities and log-linear model probabilities can be computed as:*

$$\begin{cases} p_1 = \rho_1\ \rho_3 \\ p_2 = \rho_1\ \rho_4 \\ p_3 = \rho_2\ \rho_3 \\ p_4 = \rho_2\ \rho_4 \end{cases} \quad \Longrightarrow \quad \begin{cases} log\ \delta_1 = \theta_1 + \theta_3 \\ log\ \delta_2 = \theta_1 + \theta_4 \\ log\ \delta_3 = \theta_2 + \theta_3 \\ log\ \delta_4 = \theta_2 + \theta_4 \end{cases},$$

with $\delta_i = p_i$ *and* $\theta_i = log\ \rho_i$*, for* $i = 1, 2, 3, 4$*. As a consequence, the design matrix* $\boldsymbol{A}$ *embedding this model is:*

$$\boldsymbol{A} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}.$$

$\boldsymbol{A}$ *has 4 rows since the florets of the tree have four distinct parameters* (ρ_1, ρ_2, ρ_3 *and* ρ_4*) and 4 columns for the four root-to-leaf paths with probabilities* p_1, p_2, p_3 *and* p_4*. Note that* $\boldsymbol{A}$ *is not full rank, since the first row can be obtain as a linear combination of the others:* $a_1^T = a_3^T + a_4^T - a_2^T$*, with* a_i^T *the i-th row of* $\boldsymbol{A}$*. Deleting the first row, a full row rank matrix* $\boldsymbol{A}^*$ *is obtained:*

$$\boldsymbol{A}^* = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix},$$

whose row space contains $\mathbf{1}^T$ *and then* $RM(\boldsymbol{A}^*)$ *contains the overall effect. The new system of equations is*

$$\begin{cases} p_1 = \rho_2 \\ p_2 = \rho_3 \\ p_3 = \rho_1\ \rho_2 \\ p_4 = \rho_1\ \rho_3 \end{cases},$$

which can be represented with a Staged Tree structure only as in Figure 7.5, which unfortunately is not a Staged Tree. This is because the sum up to one for each floret must be valid, but setting $\rho_2 + \rho_3 = 1$ *leads to define* $\rho_1 = 0$*. This suggests that there exists always a design matrix* $\boldsymbol{A}^*$ *representing the same statistical model embedded in the Stratified Staged Tree with design matrix* $\boldsymbol{A}$*, but there is not always a Staged Tree representation of* $\boldsymbol{A}^*$*.*

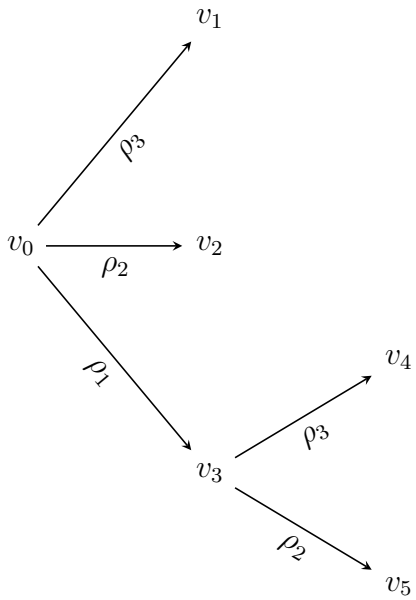


Figure 7.5: Model parametrization which can not be seen as a Staged Tree.

In conclusion, a multilinear Staged Tree model can be always reparameterized in order to be written as a relational model in the form of Equation (7.2) through a design matrix $\boldsymbol{A}^*$. Furthermore, it always includes an overall effect (see Corollary 2). Figure 7.6 summarizes these relations existing between relational models and Staged Trees.

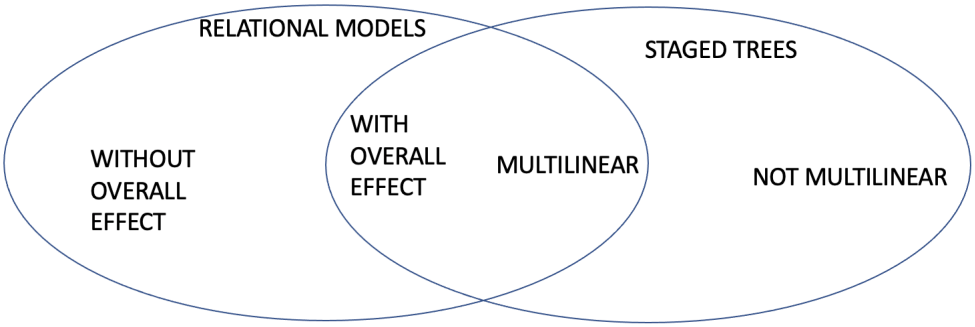


Figure 7.6: Graphical representation of relation between relational models and Staged Trees.

An important consequence for multilinear Staged Trees of being relational models with an overall effect is that they are members of regular exponential family, according to Theorem 7. This implies that the theory about maximum likelihood estimates (MLEs) of model parameters and the goodness of fit tests are analogous to those computed under traditional log-linear models and already introduced in Chapter 3; for details see Klimova et al. [72, 73] and Klimova and Rudas [71].

Another interesting result shown in Klimova et al. [73] is that if the overall effect is missing in the model, the latter belongs to a curved exponential family, while if the overall effect is added in the model, the exponential family becomes the smallest regular containing the curved one (Theorem 7).

7.4 Conclusion

In Section 7.1 a method is proposed to isolate vertices of the tree corresponding to zero counts in contingency tables in unobserved stages and, not considering them in the model search space, it drastically reduces the computational burden of estimation algorithms. Section 7.2 exhibits computational times for algorithms from **stagedtrees** applied to four datasets to monitor the effect of zero observations in some vertices. In particular, this computational burden is calculated as a ratio between the time of algorithms where vertices of the Staged Tree are not isolated in a own stage and those where are isolated. It shows that it is crucial to not consider unobserved vertices in the model search space.

Section 7.3 gives the theory of relational models. These models are introduced in this thesis since they are strongly related to Stratified Staged Trees, which are for construction multilinear trees. Proposition 15 and Corollay 2 states that a Stratified Staged Tree can be always seen as a relational model as in Equation (7.2), with a reparameterization of its transition probabilities. The theory on relational models guarantees also that the maximum likelihood estimation and statistical tests for Stratified Staged Trees are the traditional ones for log-linear models. An important area of application of relational models is the case when several binary variables are observed, but the combination that no variable is present is either logically impossible (*structural zero*) or is possible but was left out from the study design (*observed zero*). Section 7.3.2 gives details on the relationship existing between relational models and Staged Trees: Stratified Staged Trees are relational models with the overall effect. This section is only a first result of the research carried out on this topic, which can be used as a starting point for future works.

Conclusion

This work provides several new contributions to the state of the art on Staged Trees and Chain Event Graphs, among which the main ones are the following:

- score-based and distance-based algorithms for learning the stage structure of Staged Trees. The first are based on the minimization of AIC or BIC index or on the maximization of the penalized log-likelihood, while the second can be implemented using any distance or divergence that can be calculate between two discrete probability distributions [20];
- algorithms based on conditional entropy, mutual information and conditional mutual information to establish a variables ordering to use for the construction of a Stratified Staged Tree. This is because the ordering of strata in the tree plays a crucial role [21];
- how to treat vertices of tree where no data is observed. It is empirically shown how learning algorithms are appealing also from a computational point of view when these vertices are excluded from the model search space. Otherwise, considering also them, they result too slow with respect to those from the state of the art;
- algorithm to obtain the Staged Tree representation embedding the same statistical model of a given Bayesian Network and a topological order of the considered variables [121];
- new class of Directed Acyclic Graph, named Asymmetric-labeled DAG (ALDAG), which gives a BN representation of a given Staged Tree. The ALDAG is a minimal DAG such that the statistical model embedded in the Staged Tree is contained in the one associated to the ALDAG. This is possible thanks to the use of colored edges, so that each color indicates a different type of conditional dependence: total, context-specific, partial or local [121];
- Staged Trees view as a statistical tool for classification purpose. Staged Tree Classifiers are introduced, which exhibit comparable predictive results based

on accuracy with respect to algorithms from state of the art of machine learning such as neural networks and random forests [21];

- *R* package **stagedtrees** which permits to implement structural learning algorithms for Staged Trees and Chain Event Graphs for a set of categorical variables. Various functionalities to provide inferential, visualization, descriptive and summary statistics tools for such models and about their graph structure are developed [120].

In particular, this work offers in its first chapter an overview of the state of the art on Probabilistic Graphical Models. It exhibits also the usage of Undirected Graphs in a pediatric dentistry framework, providing interesting results and conclusions about the mechanism according to which dental caries form in children. The details of an estimation criterion based on neighborhood regression is reported, since it is useful for a real application in Chapter 2. Indeed, Chapter 2 proposes a complex procedure for an insurance analysis, which starts from a potentially large number of indicators collected at different time frequencies, selects the most relevant ones through Probabilistic Graphical Models and uses regressive models to forecast trends for two response variables. The use of PGMs makes the variable selection more understandable and interpretable even for not statisticians. The estimated PGMs turned out to be useful to confirm known independence relationships, to validate the used dataset and mainly to identify unexpected relationships. Robustness of estimates and data sensitivity, which are crucial in an insurance context, are dealt with a bootstrap approach. Goodness of fit and of estimates is expressed via the percentage error and the mean square error, resulting the GLARMA model the best fitting model for this analysis, among those tried.

Chapter 3 gives the motivations for which Staged Trees and Chain Event Graphs have been introduced in the last decade in the state of the art of PGMs. Shortly, in real application with the increase of available data, methods that do not allow to identify asymmetric or context-specific conditional independence relationships, as for instance Bayesian Networks and Undirected Graphs, are increasingly unusable. Innovative learning algorithms for the estimation of the stage structure of Staged Trees are proposed. Three algorithms based on fundamentals of probability and information theory, i.e. mutual information, conditional mutual information and conditional entropy, are introduced in order to obtain a variables ordering to use for the construction of the Staged Tree for a set of variables of interest. The chapter ends providing methods for computing confidence intervals for probability parameters of Staged Trees. If one is interested to confidence intervals for many

stages probability distributions, the Goodman method results to be more robust, since it uses a Bonferroni correction. A considerable amount of care must be used in calculating confidence intervals in presence of very small sample sizes for some nodes of the tree and with highly unbalanced, almost degenerate, distributions.

A particular Directed Acyclic Graph encoding also these types of conditional independence statements is proposed in Chapter 4: Asymmetry-Labeled DAG (ALDAG). It is able to represent in a unique graph total, context-specific, partial and local conditional independences by depicting edges between nodes with different colors. In the chapter the close relationship that exists between BNs and Staged Trees is outlined, proposing an algorithm which takes as input a BN and a topological order of variables and returns the corresponding Stratified Staged Tree embedding the same statistical model. Also the opposite algorithm is implemented: it takes in input a Staged Tree and returns the corresponding ALDAG.

Chapter 5 exhibits the *R* package **stagedtrees** which permits to estimate a Staged Tree from a given dataset using structure learning algorithms based on score optimization or clustering of probability distributions or distance/divergence between pair of probability distributions. An exhaustive simulation study on nine datasets is conducted to show the performances of each algorithm as the parameters that have to be set change. The application on the pediatric dentistry dataset carried out in Chapter 1 is continued through Staged Trees, which are able to manage the presence of structural zeros in this dataset. Furthermore, many asymmetric conditional independences which are material on the current research are detected.

Chapter 6 of this work shows as Staged Trees can be view as a classification tool. In particular, Staged Tree Classifiers are defined, with a focus on the Naive Staged Tree Classifier which is the Staged Tree representation of the Naive Bayes Classifier. It appears that Staged Tree Classifiers have predictive performances comparable with those obtained with the state of the art classifiers. This is the results of a classification analysis carried out on fourteen datasets, on which accuracy, area under the ROC curve and computational time of algorithms from **stagedtrees** and the most famous classifiers in the state of the art are compared. The great advantage of the algorithms from **stagedtrees** is that they provide a graphical representation of the results, making them more understandable to anyone has to interpret them.

In the last chapter the importance of manage vertices in Staged Trees associated to zero counts in contingency tables is explained in detail from theoretical, practical and computational point of view. Indeed, computational times of all algorithms in **stagedtrees** have been compared when unobserved vertices, i.e. vertices associated to contingency tables with structural or observed zero counts, are left untouched

from the model search or not. It turns out that not considering these vertices in the analysis reduces considerably the computational times. The chapter offers also a review of the state of the art on relational models as well as the relationship existing between these models and Staged Trees. It follows that Stratified Staged Trees are relational models with the overall effect, ensuring that the maximum likelihood estimates of its parameters are unbiased.

Acknowledgment

The PhD has been funded by Swiss Re Group. Swiss Re support and collaboration are gratefully acknowledged, with a special mention to Dr Patrizia Ferretti-Kern, Andrea Mazza, Nicola Linguerri and Ermir Qeli. Patrizia made the PhD position available in Genoa and hosted me for a week at the Swiss Re headquarter in Zurich; Andrea always provided support and direct contact, making himself available for any clarification regarding the projects carried out with Swiss Re; Nicola, my Company tutor, kept me updated on various Company projects with the aim of getting me to know the Company better; Ermir gave me a much appreciated advice: "study, read and experiment new and different things during the PhD, that's a luxury time when you can do that".

Also the Department of Mathematics of the University of Genoa and Professor Eva Riccomagno are thanked for the opportunity that I was granted to work in the research sector in a cutting-edge topic. Eva has always supported me with her great experience and extensive knowledge during these PhD years, stimulating interesting discussions that have helped to provide interesting insights for the research carried out.

The works and projects inherent to Swiss Re have been developed together with the PhD student Elena Pesce. She is thanked for the understanding and collaboration that have led to working well and in harmony.

Last but surely not least, I am grateful to Dr Gherardo Varando and Manuele Leonelli for the many exchanges of ideas and opinions they had with me about the central topic of my PhD, the Staged Trees. Indeed, thanks to this close and fruitful collaboration started at the end of the first PhD year, I have got interested into Staged Trees.

Appendix

Chapter 1

4 unit identification variable	Id; Progressive number; Gender; Ethnicity
18 general questionnaire variables	Binary variables. If the child was born in time; has been breastfed, if yes, if he/she has been breastfed in an exclusive or mixed way; has used the feeding bottle; has used the pacifier; has sucked his thumb; has received instructions of oral hygiene; has suffered tooth or face trauma; takes fluoride. Other variables: how many times each day the child brushes his teeth. For how many months the child has been breastfed; used the feeding bottle; used the pacifier; sucked his thumb.
29 variables measured at the three timepoints age 3, 4 and 5	Binary variables. If the child has had tonsils or adenoids surgery; allergies; good oral hygiene; only milk teeth; an infantile or normal swallowing; lip competence; normal or oral type of breathing; recurrent otitis; wears a cross bite; drinks juices, tea or sugary drinks; drinks coca cola, fanta or other carbonated drinks. Ternary variables. If the child eats fruit or vegetables daily, weekly or occasionally; drinks juices, tea or sugary drinks daily, weekly or occasionally; drinks coca cola, fanta or other carbonated drinks daily, weekly or occasionally; has mesial, head to head or distal step right occlusion; has mesial, head to head or distal step left occlusion; has first, second or third class of right canine key; has first, second or third class of left canine key; has a bilateral, right unilateral or left unilateral cross bite; has a right deflected, centered or left deflected upper midline; has a right deflected, centered or left deflected lower midline; has a normal, low or interposed posture of tongue. Other variables: child age; weight; height; number of caries and of fillings; mm of overbite and of overjet.

Table 7.5: Measured variables in the longitudinal study.

Chapter 3

The main properties and relations among the distances introduced in Chapter 3 are the following:

- $\delta(p,q) \;=\; \frac{1}{2}\, d_1(p,q)$
- $D_{KL}(p \parallel q) \;\geq\; 0$
- $\lim\limits_{\alpha \to 1} \; D_\alpha(p \parallel q) \;=\; D_{KL}(p \parallel q)$
- $\lim\limits_{\alpha \to \frac{1}{2}} \; D_\alpha(p \parallel q) \;=\; -2 \log \mathrm{BC}(p,q) \;=\; -2 \log \sum_{x \in X} \sqrt{p(x)\; q(x)}$
- $0 \;\leq\; \mathrm{BC}(p,q) \;\leq\; 1 \qquad \Rightarrow \qquad 0 \;\leq\; D_B(p,q) \;\leq\; \infty$
- **Pinsker's inequality:** $\delta(p,q) \;\leq\; \sqrt{\frac{1}{2}\; D_{KL}(p \parallel q)}$
- $H(p,q) \;= \sqrt{1 \;-\; \mathrm{BC}(p,q)} \;=\; \sqrt{1 \;-\; e^{-D_B(p,q)}}$
- $H^2(p,q) \;\leq\; \delta(p,q) \;\leq\; \sqrt{2}\; H(p,q)$

Mutual information, conditional mutual information and conditional entropy have many properties, among which the main ones are the following:

1. $I(X;Y|Z) \;=\; H(X,Z) \;+\; H(Y,Z) \;-\; H(Z) \;-\; H(X,Y,Z) \;=$
   $\qquad = \; H(X|Z) \;-\; H(X|Y,Z) \;=$
   $\qquad = \; H(X|Z) + \; H(Y|Z) \;-\; H(X,Y|Z)$
2. $I(X;Y|Z) \;=\; D_{KL}(p(x,y,z) \parallel p(x|z)\; p(y|z)\; p(z))$
3. $I(X;Y;Z) \;=\; I(X;Z) \;+\; I(X;Y|Z)$
4. $I(X;Y|Z) \;\geq\; 0$ always
5. $I(X;Y) \;=\; I(Y;X)$
6. $I(X;Y) \;=\; H(X) \;-\; H(X|Y) \;=$
   $\qquad = \; H(Y) \;-\; H(Y|X) \;=$
   $\qquad = \; H(X) + \; H(Y) \;-\; H(X,Y) \;=$
   $\qquad = \; H(X,Y) \;-\; H(X|Y) \;-\; H(Y|X)$
7. $I(X;Y) \;=\; \mathbb{E}_Y\Big[\; D_{KL}(p(x,y) \parallel p(x))\Big]$
8. $H(X) \;\geq\; H(X|Y)$

9. $H(Y) = I(Y,Y)$

10. $H(Y|X) = 0 \iff X$ and Y completely dependent

Chapter 5

Model	df	logLik	AIC	BIC	Accuracy	Computational Time
Indep	5.00	-2151.47	4312.94	4336.36	0.70	0.05
Full	15.00	-1914.68	3859.36	3929.63	0.85	0.05
HC - Indep	10.30	-1919.84	3860.27	3908.52	0.85	0.12
BHC	10.30	-1919.85	3860.30	3908.55	0.85	0.06
Fast BHC	10.10	-1921.83	3863.85	3911.17	0.85	0.06
Random BHC	9.10	-1921.83	3861.85	3904.48	0.85	0.06
Kullback-Leibler - 0.01	13.30	-1914.88	3856.36	3918.67	0.85	0.05
Kullback-Leibler - 0.05	11.50	-1916.79	3856.57	3910.45	0.85	0.05
Kullback-Leibler - 0.20	9.40	-1927.55	3873.90	3917.93	0.85	0.06
Kullback-Leibler - 0.50	8.00	-1947.56	3911.12	3948.59	0.85	0.05
Manhattan - 0.01	14.80	-1914.68	3858.96	3928.30	0.85	0.05
Manhattan - 0.05	13.80	-1914.74	3857.09	3921.73	0.85	0.06
Manhattan - 0.20	11.10	-1917.55	3857.30	3909.29	0.85	0.05
Manhattan - 0.50	8.40	-1940.09	3896.98	3936.34	0.85	0.05
Euclidean - 0.01	14.70	-1914.68	3858.77	3927.63	0.85	0.05
Euclidean - 0.05	13.60	-1914.76	3856.73	3920.44	0.85	0.05
Euclidean - 0.20	10.00	-1921.27	3862.53	3909.38	0.85	0.06
Euclidean - 0.50	7.00	-1956.18	3926.35	3959.14	0.85	0.06
Renyi - 0.01	13.60	-1914.76	3856.73	3920.44	0.85	0.06
Renyi - 0.05	11.90	-1916.18	3856.16	3911.91	0.85	0.05
Renyi - 0.20	10.10	-1920.78	3861.77	3909.08	0.85	0.06
Renyi - 0.50	8.60	-1937.35	3891.90	3932.19	0.85	0.06
Total Variation - 0.01	14.80	-1914.68	3858.96	3928.30	0.85	0.05
Total Variation - 0.05	13.80	-1914.74	3857.09	3921.73	0.85	0.05
Total Variation - 0.20	11.10	-1917.55	3857.30	3909.29	0.85	0.06
Total Variation - 0.50	8.40	-1940.09	3896.98	3936.34	0.85	0.06
Hellinger - 0.01	11.90	-1916.18	3856.16	3911.91	0.85	0.06
Hellinger - 0.05	9.40	-1927.55	3873.90	3917.93	0.85	0.06
Hellinger - 0.20	7.00	-1956.18	3926.35	3959.14	0.85	0.05
Hellinger - 0.50	6.00	-2058.07	4128.13	4156.24	0.70	0.06
Bhattacharyya - 0.01	10.80	-1918.35	3858.30	3908.89	0.85	0.05
Bhattacharyya - 0.05	8.20	-1943.60	3903.59	3942.01	0.85	0.05
Bhattacharyya - 0.20	6.00	-2058.07	4128.13	4156.24	0.70	0.06
Bhattacharyya - 0.50	5.00	-2151.47	4312.94	4336.36	0.70	0.06
Chan-Darwiche - 0.01	15.00	-1914.68	3859.36	3929.63	0.85	0.05
Chan-Darwiche - 0.05	14.50	-1914.69	3858.38	3926.31	0.85	0.05
Chan-Darwiche - 0.20	13.30	-1914.88	3856.36	3918.67	0.85	0.05
Chan-Darwiche - 0.50	11.80	-1916.36	3856.33	3911.61	0.85	0.05
HClust k = 2	8.00	-1957.83	3931.65	3969.13	0.85	0.05
HClust k = 3	10.00	-1929.41	3878.82	3925.67	0.85	0.05
HClust k = 4	12.00	-1919.89	3863.77	3919.99	0.85	0.05
Kmeans k = 2	8.00	-1956.07	3928.14	3965.62	0.85	0.06
Kmeans k = 3	10.00	-1929.40	3878.80	3925.65	0.85	0.06
Kmeans k = 4	12.00	-1920.99	3865.98	3922.19	0.85	0.05
Refined BN	9.00	-1916.46	3850.92	3893.08	0.85	0.08

Table 7.6: Mean results for **stagedtrees** algorithms over 10 replications for `Asym` dataset. Experiments performed on a standard laptop with 8 GB of RAM and an i5 3.1 GHz CPU.

Model	df	logLik	AIC	BIC	Accuracy	Computational Time
Indep	13.40	-1170.78	2368.35	2421.84	0.54	0.08
Full	73.20	-897.04	1940.49	2232.66	0.85	0.08
HC - Indep	20.40	-886.90	1814.60	1896.03	0.85	2.17
BHC	20.50	-886.57	1814.14	1895.96	0.85	0.32
Fast BHC	20.10	-908.36	1856.92	1937.14	0.85	0.09
Random BHC	36.10	-912.44	1897.07	2041.16	0.84	0.08
Kullback-Leibler - 0.01	44.50	-886.20	1861.40	2039.02	0.84	0.11
Kullback-Leibler - 0.05	32.10	-884.94	1834.08	1962.21	0.85	0.10
Kullback-Leibler - 0.20	24.10	-897.33	1842.85	1939.04	0.84	0.10
Kullback-Leibler - 0.50	19.80	-912.40	1864.40	1943.43	0.84	0.10
Manhattan - 0.01	54.80	-889.05	1887.70	2106.43	0.84	0.09
Manhattan - 0.05	45.40	-887.57	1865.94	2047.15	0.84	0.09
Manhattan - 0.20	29.80	-889.01	1837.62	1956.57	0.84	0.09
Manhattan - 0.50	20.00	-908.69	1857.38	1937.21	0.84	0.09
Euclidean - 0.01	53.60	-888.34	1883.88	2097.82	0.84	0.08
Euclidean - 0.05	42.40	-887.28	1859.35	2028.59	0.85	0.09
Euclidean - 0.20	26.00	-894.07	1840.14	1943.91	0.84	0.09
Euclidean - 0.50	18.00	-914.10	1864.21	1936.05	0.83	0.09
Renyi - 0.01	49.40	-887.82	1874.44	2071.61	0.84	0.09
Renyi - 0.05	38.00	-885.15	1846.31	1997.98	0.85	0.09
Renyi - 0.20	29.00	-885.63	1829.25	1945.00	0.84	0.09
Renyi - 0.50	22.80	-902.83	1851.26	1942.26	0.84	0.10
Total Variation - 0.01	54.80	-889.05	1887.70	2106.43	0.84	0.08
Total Variation - 0.05	45.40	-887.57	1865.94	2047.15	0.84	0.08
Total Variation - 0.20	29.80	-889.01	1837.62	1956.57	0.84	0.09
Total Variation - 0.50	20.00	-908.69	1857.38	1937.21	0.84	0.09
Hellinger - 0.01	33.50	-884.68	1836.35	1970.07	0.85	0.09
Hellinger - 0.05	24.00	-898.07	1844.14	1939.93	0.84	0.09
Hellinger - 0.20	18.00	-914.89	1865.79	1937.64	0.84	0.09
Hellinger - 0.50	17.50	-925.12	1885.24	1955.09	0.81	0.09
Bhattacharyya - 0.01	30.00	-885.16	1830.33	1950.07	0.84	0.09
Bhattacharyya - 0.05	21.10	-908.78	1859.77	1943.99	0.84	0.09
Bhattacharyya - 0.20	17.70	-915.17	1865.74	1936.39	0.84	0.09
Bhattacharyya - 0.50	15.40	-1021.94	2074.68	2136.15	0.54	0.09
Chan-Darwiche - 0.01	59.60	-892.09	1903.37	2141.27	0.85	0.09
Chan-Darwiche - 0.05	58.10	-891.83	1899.86	2131.77	0.84	0.09
Chan-Darwiche - 0.20	51.60	-889.92	1883.03	2088.99	0.84	0.09
Chan-Darwiche - 0.50	38.90	-885.24	1848.28	2003.54	0.84	0.09
HClust k = 2	20.40	-904.68	1850.17	1931.59	0.84	0.08
HClust k = 3	26.40	-894.35	1841.50	1946.87	0.84	0.08
HClust k = 4	32.00	-887.79	1839.58	1967.31	0.84	0.08
Kmeans k = 2	20.40	-905.75	1852.30	1933.72	0.84	0.08
Kmeans k = 3	26.40	-890.17	1833.13	1938.51	0.84	0.08
Kmeans k = 4	32.00	-887.50	1839.00	1966.73	0.84	0.08
Refined BN	18.50	-892.87	1822.73	1896.58	0.85	1.41

Table 7.7: Mean results for **stagedtrees** algorithms over 10 replications for `chestSim500` dataset. Experiments performed on a standard laptop with 8 GB of RAM and an i5 3.1 GHz CPU.

Model	df	logLik	AIC	BIC	Accuracy	Computational Time
Indep	8.00	-69495.80	139007.61	139076.38	0.69	2.68
Full	63.00	-54797.79	109721.58	110263.17	0.77	2.68
HC - Indep	24.50	-54780.72	109610.44	109821.05	0.77	3.06
BHC	24.50	-54780.72	109610.44	109821.05	0.77	2.74
Fast BHC	24.40	-54797.71	109644.22	109853.98	0.77	2.72
Random BHC	27.20	-54792.90	109640.20	109874.03	0.77	2.67
Kullback-Leibler - 0.01	45.20	-54783.11	109656.61	110045.18	0.77	2.69
Kullback-Leibler - 0.05	29.50	-54882.73	109824.47	110078.07	0.77	2.69
Kullback-Leibler - 0.20	24.70	-54921.32	109892.05	110104.38	0.76	2.76
Kullback-Leibler - 0.50	16.00	-54950.54	109933.09	110070.63	0.76	2.68
Manhattan - 0.01	49.50	-54784.34	109667.68	110093.21	0.77	2.67
Manhattan - 0.05	40.30	-54778.43	109637.47	109983.91	0.77	2.67
Manhattan - 0.20	21.30	-54887.62	109817.84	110000.95	0.77	2.67
Manhattan - 0.50	16.00	-54950.54	109933.09	110070.63	0.76	2.66
Euclidean - 0.01	44.70	-54779.78	109648.96	110033.23	0.77	2.73
Euclidean - 0.05	38.10	-54778.18	109632.57	109960.10	0.77	2.86
Euclidean - 0.20	19.80	-54897.22	109834.04	110004.25	0.77	2.79
Euclidean - 0.50	15.00	-57872.08	115774.16	115903.11	0.70	2.71
Renyi - 0.01	50.10	-54787.43	109675.07	110105.76	0.77	2.73
Renyi - 0.05	38.50	-54871.66	109820.32	110151.29	0.77	2.70
Renyi - 0.20	25.80	-54906.76	109865.13	110086.92	0.77	2.71
Renyi - 0.50	21.70	-54942.22	109927.84	110114.38	0.76	2.69
Total Variation - 0.01	49.50	-54784.34	109667.68	110093.21	0.77	2.69
Total Variation - 0.05	40.30	-54778.43	109637.47	109983.91	0.77	2.71
Total Variation - 0.20	21.30	-54887.62	109817.84	110000.95	0.77	2.71
Total Variation - 0.50	16.00	-54950.54	109933.09	110070.63	0.76	2.71
Hellinger - 0.01	30.90	-54872.56	109806.93	110072.56	0.77	2.73
Hellinger - 0.05	21.70	-54918.43	109880.27	110066.82	0.76	2.74
Hellinger - 0.20	15.00	-55342.04	110714.09	110843.04	0.76	2.67
Hellinger - 0.50	14.00	-59134.06	118296.11	118416.47	0.69	2.66
Bhattacharyya - 0.01	26.00	-54899.67	109851.33	110074.85	0.77	2.68
Bhattacharyya - 0.05	16.00	-54950.54	109933.09	110070.63	0.76	2.67
Bhattacharyya - 0.20	14.00	-59134.06	118296.11	118416.47	0.69	2.67
Bhattacharyya - 0.50	11.00	-59926.86	119875.71	119970.27	0.69	2.68
Chan-Darwiche - 0.01	63.00	-54797.79	109721.58	110263.17	0.77	2.67
Chan-Darwiche - 0.05	63.00	-54797.79	109721.58	110263.17	0.77	2.67
Chan-Darwiche - 0.20	62.00	-54799.82	109723.65	110256.64	0.77	2.68
Chan-Darwiche - 0.50	55.70	-54795.84	109703.09	110181.92	0.77	2.67
HClust k = 2	15.00	-57814.20	115658.40	115787.35	0.72	2.67
HClust k = 3	19.00	-54872.81	109783.62	109946.96	0.77	2.67
HClust k = 4	23.00	-54804.19	109654.38	109852.10	0.77	2.66
Kmeans k = 2	15.00	-58550.45	117130.89	117259.84	0.70	2.67
Kmeans k = 3	19.00	-55017.23	110072.46	110235.79	0.77	2.85
Kmeans k = 4	23.00	-54844.36	109734.71	109932.44	0.77	2.77
Refined BN	21.00	-54778.87	109599.74	109780.27	0.77	2.94

Table 7.8: Mean results for **stagedtrees** algorithms over 10 replications for `FallEld` dataset. Experiments performed on a standard laptop with 8 GB of RAM and an i5 3.1 GHz CPU.

Model	df	logLik	AIC	BIC	Accuracy	Computational Time
Indep	8.70	-3525.04	7067.49	7107.47	0.46	0.15
Full	142.40	-3264.92	6814.65	7469.09	0.46	0.14
HC - Indep	15.30	-3282.52	6595.64	6665.96	0.44	12.22
BHC	18.80	-3260.57	6558.73	6645.13	0.42	7.30
Fast BHC	14.70	-3361.31	6752.02	6819.58	0.44	0.17
Random BHC	84.80	-3314.66	6798.92	7188.65	0.45	0.14
Kullback-Leibler - 0.01	42.10	-3244.74	6573.68	6767.16	0.45	0.46
Kullback-Leibler - 0.05	26.70	-3251.73	6556.85	6679.56	0.44	0.47
Kullback-Leibler - 0.20	16.30	-3286.16	6604.92	6679.84	0.44	0.45
Kullback-Leibler - 0.50	10.80	-3364.85	6751.29	6800.93	0.43	0.47
Manhattan - 0.01	93.80	-3247.62	6682.84	7113.92	0.46	0.21
Manhattan - 0.05	55.30	-3242.87	6596.34	6850.48	0.46	0.23
Manhattan - 0.20	25.90	-3252.61	6557.02	6676.05	0.44	0.24
Manhattan - 0.50	13.30	-3313.82	6654.23	6715.36	0.43	0.24
Euclidean - 0.01	86.40	-3245.93	6664.66	7061.73	0.46	0.23
Euclidean - 0.05	46.40	-3243.19	6579.19	6792.43	0.45	0.25
Euclidean - 0.20	21.80	-3262.60	6568.80	6668.99	0.44	0.25
Euclidean - 0.50	9.90	-3394.15	6808.10	6853.60	0.43	0.26
Renyi - 0.01	50.30	-3244.01	6588.62	6819.79	0.46	0.46
Renyi - 0.05	31.20	-3248.15	6558.71	6702.10	0.45	0.47
Renyi - 0.20	22.10	-3263.11	6570.43	6671.99	0.44	0.46
Renyi - 0.50	15.00	-3294.17	6618.34	6687.28	0.44	0.47
Total Variation - 0.01	93.80	-3247.62	6682.84	7113.92	0.46	0.21
Total Variation - 0.05	55.30	-3242.87	6596.34	6850.48	0.46	0.23
Total Variation - 0.20	25.90	-3252.61	6557.02	6676.05	0.44	0.24
Total Variation - 0.50	13.30	-3313.82	6654.23	6715.36	0.43	0.24
Hellinger - 0.01	27.50	-3250.69	6556.39	6682.77	0.44	0.26
Hellinger - 0.05	16.20	-3286.77	6605.94	6680.39	0.44	0.27
Hellinger - 0.20	9.70	-3393.49	6806.39	6850.97	0.44	0.26
Hellinger - 0.50	8.90	-3498.45	7014.70	7055.60	0.46	0.27
Bhattacharyya - 0.01	24.10	-3255.77	6559.75	6670.51	0.44	0.24
Bhattacharyya - 0.05	13.30	-3307.05	6640.69	6701.82	0.45	0.24
Bhattacharyya - 0.20	9.70	-3393.49	6806.39	6850.97	0.44	0.24
Bhattacharyya - 0.50	8.70	-3525.04	7067.49	7107.47	0.46	0.25
Chan-Darwiche - 0.01	106.30	-3252.36	6717.32	7205.85	0.46	0.33
Chan-Darwiche - 0.05	85.10	-3251.69	6673.58	7064.69	0.46	0.35
Chan-Darwiche - 0.20	46.20	-3245.81	6584.02	6796.34	0.45	0.39
Chan-Darwiche - 0.50	27.30	-3252.79	6560.17	6685.64	0.44	0.39
HClust k = 2	14.70	-3320.17	6669.74	6737.30	0.44	0.14
HClust k = 3	20.70	-3283.48	6608.35	6703.48	0.43	0.14
HClust k = 4	25.70	-3260.73	6572.86	6690.97	0.44	0.14
Kmeans k = 2	14.70	-3301.21	6631.83	6699.38	0.45	0.14
Kmeans k = 3	20.70	-3274.58	6590.57	6685.70	0.45	0.14
Kmeans k = 4	25.70	-3261.63	6574.66	6692.77	0.44	0.14
Refined BN	18.80	-3302.69	6642.98	6729.38	0.48	0.39

Table 7.9: Mean results for **stagedtrees** algorithms over 10 replications for `PhDArticles` dataset. Experiments performed on a standard laptop with 8 GB of RAM and an i5 3.1 GHz CPU.

Model	df	logLik	AIC	BIC	Accuracy	Computational Time
Indep	5.00	-2628.05	5266.10	5289.52	0.72	0.07
Full	31.00	-2577.11	5216.22	5361.44	0.72	0.07
HC - Indep	6.60	-2589.86	5192.93	5223.85	0.72	0.14
BHC	7.90	-2583.28	5182.37	5219.38	0.72	0.15
Fast BHC	6.10	-2594.10	5200.41	5228.99	0.72	0.07
Random BHC	6.30	-2593.02	5198.65	5228.16	0.72	0.08
Kullback-Leibler - 0.01	12.40	-2578.31	5181.43	5239.52	0.72	0.08
Kullback-Leibler - 0.05	8.20	-2583.64	5183.68	5222.10	0.72	0.08
Kullback-Leibler - 0.20	6.10	-2594.25	5200.70	5229.28	0.72	0.09
Kullback-Leibler - 0.50	5.00	-2628.05	5266.10	5289.52	0.72	0.08
Manhattan - 0.01	26.70	-2577.09	5207.59	5332.67	0.72	0.07
Manhattan - 0.05	17.90	-2577.30	5190.40	5274.25	0.72	0.07
Manhattan - 0.20	8.50	-2582.53	5182.06	5221.88	0.72	0.08
Manhattan - 0.50	6.00	-2594.47	5200.93	5229.04	0.72	0.08
Euclidean - 0.01	25.60	-2577.09	5205.38	5325.31	0.72	0.07
Euclidean - 0.05	15.30	-2577.64	5185.89	5257.56	0.72	0.08
Euclidean - 0.20	7.40	-2587.89	5190.57	5225.24	0.72	0.08
Euclidean - 0.50	5.00	-2628.05	5266.10	5289.52	0.72	0.09
Renyi - 0.01	15.40	-2577.62	5186.03	5258.18	0.72	0.08
Renyi - 0.05	10.00	-2579.85	5179.70	5226.55	0.72	0.08
Renyi - 0.20	6.90	-2589.91	5193.62	5225.94	0.72	0.08
Renyi - 0.50	6.10	-2594.25	5200.70	5229.28	0.72	0.08
Total Variation - 0.01	26.70	-2577.09	5207.59	5332.67	0.72	0.07
Total Variation - 0.05	17.90	-2577.30	5190.40	5274.25	0.72	0.07
Total Variation - 0.20	8.50	-2582.53	5182.06	5221.88	0.72	0.09
Total Variation - 0.50	6.00	-2594.47	5200.93	5229.04	0.72	0.08
Hellinger - 0.01	8.80	-2582.10	5181.81	5223.03	0.72	0.08
Hellinger - 0.05	6.10	-2594.25	5200.70	5229.28	0.72	0.08
Hellinger - 0.20	5.00	-2628.05	5266.10	5289.52	0.72	0.08
Hellinger - 0.50	5.00	-2628.05	5266.10	5289.52	0.72	0.08
Bhattacharyya - 0.01	7.50	-2587.14	5189.28	5224.42	0.72	0.08
Bhattacharyya - 0.05	5.20	-2620.32	5251.04	5275.40	0.72	0.08
Bhattacharyya - 0.20	5.00	-2628.05	5266.10	5289.52	0.72	0.08
Bhattacharyya - 0.50	5.00	-2628.05	5266.10	5289.52	0.72	0.08
Chan-Darwiche - 0.01	29.00	-2577.10	5212.20	5348.05	0.72	0.07
Chan-Darwiche - 0.05	23.10	-2577.11	5200.42	5308.64	0.72	0.08
Chan-Darwiche - 0.20	12.50	-2578.25	5181.49	5240.05	0.72	0.08
Chan-Darwiche - 0.50	8.10	-2583.96	5184.11	5222.06	0.72	0.08
HClust k = 2	9.00	-2583.35	5184.71	5226.87	0.72	0.07
HClust k = 3	12.00	-2579.27	5182.54	5238.76	0.72	0.07
HClust k = 4	15.00	-2577.93	5185.86	5256.13	0.72	0.07
Kmeans k = 2	9.00	-2582.97	5183.94	5226.10	0.72	0.07
Kmeans k = 3	12.00	-2579.42	5182.84	5239.06	0.72	0.07
Kmeans k = 4	15.00	-2578.09	5186.18	5256.45	0.72	0.07
Refined BN	6.20	-2593.74	5199.89	5228.93	0.72	0.09

Table 7.10: Mean results for **stagedtrees** algorithms over 10 replications for `Pokemon` dataset. Experiments performed on a standard laptop with 8 GB of RAM and an i5 3.1 GHz CPU.

Model	df	logLik	AIC	BIC	Accuracy	Computational Time
Indep	21.00	-250.43	542.86	585.39	0.69	0.02
Full	139.40	-203.66	686.12	968.45	0.94	0.02
HC - Indep	27.00	-200.98	455.96	510.64	0.93	0.33
BHC	31.40	-181.33	425.46	489.05	0.90	0.26
Fast BHC	27.00	-210.83	475.67	530.35	0.96	0.02
Random BHC	19.60	-211.89	462.98	502.68	0.95	0.02
Kullback-Leibler - 0.01	91.10	-185.96	554.12	738.63	0.95	0.03
Kullback-Leibler - 0.05	75.70	-180.13	511.67	664.99	0.94	0.03
Kullback-Leibler - 0.20	54.50	-175.48	459.96	570.34	0.92	0.03
Kullback-Leibler - 0.50	41.20	-175.51	433.41	516.86	0.89	0.03
Manhattan - 0.01	99.00	-191.48	580.96	781.47	0.94	0.02
Manhattan - 0.05	95.40	-188.65	568.11	761.33	0.94	0.02
Manhattan - 0.20	72.70	-179.17	503.73	650.97	0.94	0.02
Manhattan - 0.50	45.90	-175.96	443.73	536.69	0.88	0.02
Euclidean - 0.01	98.80	-191.32	580.23	780.34	0.94	0.02
Euclidean - 0.05	90.00	-185.42	550.84	733.12	0.94	0.02
Euclidean - 0.20	56.60	-176.16	465.52	580.16	0.92	0.02
Euclidean - 0.50	32.80	-180.67	426.95	493.38	0.90	0.03
Renyi - 0.01	94.50	-188.26	565.51	756.91	0.95	0.03
Renyi - 0.05	84.60	-183.42	536.04	707.38	0.95	0.03
Renyi - 0.20	62.10	-177.51	479.22	604.99	0.93	0.03
Renyi - 0.50	49.50	-175.71	450.41	550.67	0.92	0.03
Total Variation - 0.01	99.00	-191.48	580.96	781.47	0.94	0.02
Total Variation - 0.05	95.40	-188.65	568.11	761.33	0.94	0.02
Total Variation - 0.20	72.70	-179.17	503.73	650.97	0.94	0.02
Total Variation - 0.50	45.90	-175.96	443.73	536.69	0.88	0.02
Hellinger - 0.01	79.70	-181.40	522.20	683.62	0.95	0.02
Hellinger - 0.05	54.50	-175.48	459.96	570.34	0.92	0.02
Hellinger - 0.20	35.30	-178.71	428.02	499.52	0.90	0.03
Hellinger - 0.50	29.50	-189.40	437.80	497.54	0.90	0.03
Bhattacharyya - 0.01	65.20	-177.81	486.01	618.07	0.93	0.02
Bhattacharyya - 0.05	44.40	-175.42	439.64	529.57	0.88	0.02
Bhattacharyya - 0.20	30.80	-184.24	430.08	492.46	0.90	0.03
Bhattacharyya - 0.50	23.10	-230.14	506.48	553.27	0.68	0.03
Chan-Darwiche - 0.01	99.00	-191.48	580.96	781.47	0.94	0.03
Chan-Darwiche - 0.05	99.00	-191.48	580.96	781.47	0.94	0.03
Chan-Darwiche - 0.20	92.50	-187.04	559.08	746.42	0.95	0.03
Chan-Darwiche - 0.50	80.00	-182.02	524.04	686.07	0.95	0.03
HClust k = 2	33.00	-189.24	444.49	511.32	0.92	0.02
HClust k = 3	44.00	-182.11	452.22	541.34	0.93	0.02
HClust k = 4	55.00	-181.86	473.72	585.11	0.92	0.02
Kmeans k = 2	33.00	-187.08	440.16	506.99	0.93	0.02
Kmeans k = 3	44.00	-180.79	449.59	538.70	0.92	0.02
Kmeans k = 4	55.00	-182.57	475.14	586.54	0.90	0.02
Refined BN	30.10	-200.56	461.31	522.28	0.95	1.57

Table 7.11: Mean results for **stagedtrees** algorithms over 10 replications for `puffin` dataset. Experiments performed on a standard laptop with 8 GB of RAM and an i5 3.1 GHz CPU.

Model	df	logLik	AIC	BIC	Accuracy	Computational Time
Indep	6.00	-5651.15	11314.30	11346.08	0.86	0.17
Full	63.00	-5303.06	10732.12	11065.71	0.85	0.17
HC - Indep	10.20	-5330.93	10682.27	10736.28	0.86	1.34
BHC	12.20	-5317.72	10659.84	10724.44	0.85	0.79
Fast BHC	9.50	-5343.53	10706.05	10756.35	0.86	0.18
Random BHC	18.20	-5341.70	10719.81	10816.18	0.86	0.18
Kullback-Leibler - 0.01	23.50	-5305.60	10658.19	10782.63	0.85	0.20
Kullback-Leibler - 0.05	14.30	-5320.88	10670.36	10746.08	0.85	0.20
Kullback-Leibler - 0.20	8.70	-5357.79	10732.97	10779.04	0.86	0.21
Kullback-Leibler - 0.50	7.00	-5374.01	10762.01	10799.08	0.86	0.21
Manhattan - 0.01	49.10	-5303.03	10704.26	10964.25	0.85	0.18
Manhattan - 0.05	32.10	-5303.28	10670.77	10840.74	0.85	0.18
Manhattan - 0.20	14.90	-5318.15	10666.11	10745.00	0.85	0.18
Manhattan - 0.50	8.00	-5362.16	10740.32	10782.68	0.86	0.19
Euclidean - 0.01	45.90	-5303.04	10697.88	10940.92	0.85	0.18
Euclidean - 0.05	27.10	-5304.08	10662.36	10805.86	0.85	0.18
Euclidean - 0.20	11.50	-5336.74	10696.49	10757.38	0.86	0.19
Euclidean - 0.50	7.00	-5374.01	10762.01	10799.08	0.86	0.19
Renyi - 0.01	28.20	-5303.78	10663.97	10813.29	0.85	0.20
Renyi - 0.05	17.60	-5312.40	10660.00	10753.19	0.85	0.20
Renyi - 0.20	11.30	-5336.83	10696.26	10756.10	0.85	0.21
Renyi - 0.50	8.00	-5362.16	10740.32	10782.68	0.86	0.20
Total Variation - 0.01	49.10	-5303.03	10704.26	10964.25	0.85	0.18
Total Variation - 0.05	32.10	-5303.28	10670.77	10840.74	0.85	0.18
Total Variation - 0.20	14.90	-5318.15	10666.11	10745.00	0.85	0.19
Total Variation - 0.50	8.00	-5362.16	10740.32	10782.68	0.86	0.19
Hellinger - 0.01	15.40	-5317.05	10664.89	10746.44	0.85	0.19
Hellinger - 0.05	8.60	-5358.06	10733.31	10778.85	0.86	0.19
Hellinger - 0.20	7.00	-5374.01	10762.01	10799.08	0.86	0.19
Hellinger - 0.50	6.00	-5651.15	11314.30	11346.08	0.86	0.19
Bhattacharyya - 0.01	12.90	-5325.47	10676.75	10745.05	0.85	0.18
Bhattacharyya - 0.05	7.30	-5373.22	10761.03	10799.68	0.86	0.19
Bhattacharyya - 0.20	7.00	-5374.01	10762.01	10799.08	0.86	0.19
Bhattacharyya - 0.50	6.00	-5651.15	11314.30	11346.08	0.86	0.19
Chan-Darwiche - 0.01	53.90	-5303.04	10713.88	10999.28	0.85	0.18
Chan-Darwiche - 0.05	41.00	-5303.05	10688.11	10905.21	0.85	0.19
Chan-Darwiche - 0.20	24.70	-5305.42	10660.23	10791.02	0.85	0.20
Chan-Darwiche - 0.50	14.30	-5321.64	10671.88	10747.60	0.85	0.20
HClust k = 2	11.00	-5341.77	10705.55	10763.79	0.86	0.17
HClust k = 3	15.00	-5324.98	10679.95	10759.38	0.86	0.18
HClust k = 4	19.00	-5310.91	10659.82	10760.43	0.85	0.18
Kmeans k = 2	11.00	-5336.97	10695.94	10754.19	0.86	0.17
Kmeans k = 3	15.00	-5321.69	10673.37	10752.80	0.86	0.18
Kmeans k = 4	19.00	-5311.20	10660.39	10761.00	0.85	0.18
Refined BN	10.90	-5336.92	10695.63	10753.35	0.86	0.28

Table 7.12: Mean results for **stagedtrees** algorithms over 10 replications for `reinis` dataset. Experiments performed on a standard laptop with 8 GB of RAM and an i5 3.1 GHz CPU.

Model	df	logLik	AIC	BIC	Accuracy	Computational Time
Indep	6.00	-4610.44	9232.88	9265.72	0.68	0.13
Full	31.00	-4119.90	8301.80	8471.48	0.79	0.12
HC - Indep	15.10	-4121.59	8273.38	8356.03	0.79	0.46
BHC	15.60	-4119.42	8270.04	8355.42	0.79	0.19
Fast BHC	14.90	-4123.30	8276.40	8357.96	0.79	0.12
Random BHC	15.10	-4126.49	8283.18	8365.83	0.79	0.13
Kullback-Leibler - 0.01	20.70	-4117.32	8276.04	8389.34	0.79	0.12
Kullback-Leibler - 0.05	16.20	-4122.03	8276.46	8365.13	0.79	0.12
Kullback-Leibler - 0.20	13.40	-4143.84	8314.48	8387.83	0.78	0.12
Kullback-Leibler - 0.50	8.10	-4327.43	8671.06	8715.40	0.77	0.12
Manhattan - 0.01	26.60	-4117.63	8288.47	8434.07	0.79	0.12
Manhattan - 0.05	20.40	-4118.27	8277.33	8388.99	0.79	0.12
Manhattan - 0.20	14.30	-4163.65	8355.91	8434.18	0.77	0.12
Manhattan - 0.50	10.20	-4284.46	8589.31	8645.14	0.77	0.12
Euclidean - 0.01	25.70	-4117.42	8286.25	8426.92	0.79	0.12
Euclidean - 0.05	18.90	-4119.17	8276.14	8379.59	0.79	0.12
Euclidean - 0.20	13.10	-4209.33	8444.87	8516.57	0.78	0.12
Euclidean - 0.50	7.90	-4338.84	8693.48	8736.72	0.77	0.13
Renyi - 0.01	22.20	-4117.23	8278.87	8400.38	0.79	0.12
Renyi - 0.05	19.20	-4117.77	8273.94	8379.04	0.79	0.12
Renyi - 0.20	15.50	-4125.59	8282.18	8367.03	0.79	0.12
Renyi - 0.50	13.10	-4147.19	8320.58	8392.28	0.78	0.14
Total Variation - 0.01	26.60	-4117.63	8288.47	8434.07	0.79	0.12
Total Variation - 0.05	20.40	-4118.27	8277.33	8388.99	0.79	0.12
Total Variation - 0.20	14.30	-4163.65	8355.91	8434.18	0.77	0.12
Total Variation - 0.50	10.20	-4284.46	8589.31	8645.14	0.77	0.12
Hellinger - 0.01	17.10	-4120.38	8274.97	8368.56	0.79	0.12
Hellinger - 0.05	13.40	-4143.84	8314.48	8387.83	0.78	0.12
Hellinger - 0.20	7.90	-4338.84	8693.48	8736.72	0.77	0.12
Hellinger - 0.50	6.00	-4610.44	9232.88	9265.72	0.68	0.12
Bhattacharyya - 0.01	15.50	-4125.03	8281.07	8365.91	0.79	0.12
Bhattacharyya - 0.05	8.80	-4310.75	8639.10	8687.27	0.77	0.12
Bhattacharyya - 0.20	7.00	-4353.48	8720.96	8759.28	0.77	0.12
Bhattacharyya - 0.50	6.00	-4610.44	9232.88	9265.72	0.68	0.12
Chan-Darwiche - 0.01	30.60	-4119.81	8300.82	8468.31	0.79	0.12
Chan-Darwiche - 0.05	28.70	-4119.71	8296.82	8453.91	0.79	0.12
Chan-Darwiche - 0.20	24.90	-4119.34	8288.47	8424.76	0.79	0.12
Chan-Darwiche - 0.50	19.70	-4119.45	8278.30	8386.13	0.79	0.12
HClust k = 2	11.00	-4266.11	8554.22	8614.43	0.77	0.12
HClust k = 3	13.00	-4216.91	8459.83	8530.98	0.76	0.12
HClust k = 4	15.00	-4153.64	8337.29	8419.39	0.78	0.12
Kmeans k = 2	11.00	-4218.70	8459.41	8519.62	0.77	0.12
Kmeans k = 3	13.00	-4165.42	8356.84	8427.99	0.78	0.12
Kmeans k = 4	15.00	-4131.36	8292.72	8374.82	0.78	0.12
Refined BN	16.80	-4128.07	8289.74	8381.69	0.79	0.17

Table 7.13: Mean results for **stagedtrees** algorithms over 10 replications for `Titanic` dataset. Experiments performed on a standard laptop with 8 GB of RAM and an i5 3.1 GHz CPU.

Bibliography

[1] Alan Agresti. *Categorical data analysis*, volume 482. John Wiley & Sons, 2003.

[2] Richard A Armstrong. When to use the b onferroni correction. *Ophthalmic and Physiological Optics*, 34(5):502–508, 2014.

[3] Stefan Bache and Hadley Wickham. ***magrittr**: A Forward-Pipe Operator for R*, 2014. URL `https://CRAN.R-project.org/package=magrittr`. `R` package version 1.5.

[4] Lorna M Barclay, Jane L Hutton, and Jim Q Smith. Refining a Bayesian Network Using a Chain Event Graph. *International Journal of Approximate Reasoning*, 54:1300–1309, 2013. doi: 10.1016/j.ijar.2013.05.006.

[5] Lorna M Barclay, Jane L Hutton, and Jim Q Smith. Chain event graphs for informed missingness. *Bayesian Analysis*, 9(1):53–76, 2014. doi: 10.1214/13-BA843.

[6] Lorna M Barclay, Rodrigo A Collazo, Jim Q Smith, Peter A Thwaites, Ann E Nicholson, et al. The dynamic chain event graph. *Electronic Journal of Statistics*, 9(2):2130–2169, 2015.

[7] Michael A Baxter. *Interpolating annual data into monthly or quarterly data*. Number 6. Office for National Statistics, 1998.

[8] Marco Benjumeda, Sergio Luengo-Sanchez, Pedro Larrañaga, and Concha Bielza. Tractable learning of bayesian networks from partially observed data. *Pattern Recognition*, 91:190–199, 2019.

[9] Concha Bielza and Pedro Larranaga. Discrete bayesian network classifiers: A survey. *ACM Computing Surveys (CSUR)*, 47(1):1–43, 2014.

[10] Yvonne M Bishop, Stephen E Fienberg, and Paul W Holland. *Discrete multivariate analysis: theory and practice*. Springer Science & Business Media, 2007.

[11] John CG Boot, Walter Feibes, and Johannes Hubertus Cornelius Lisman. Further methods of derivation of quarterly figures from annual data. *Applied Statistics*, pages 65–75, 1967.

[12] C Boutilier, N Friedman, M Goldszmidt, and D Koller. Context-Specific Independence in Bayesian Networks. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, pages 115–123, 1996.

[13] Craig Boutilier, Nir Friedman, Moises Goldszmidt, and Daphne Koller. Context-specific independence in bayesian networks. *arXiv preprint arXiv:1302.3562*, 2013.

[14] Charles Bouveyron, Gilles Celeux, Brendan Murphy, and Adrian Raftery. *Model-Based Clustering and Classification for Data Science: With Applications in* `R`. Cambridge University Press, 2019. doi: 10.1017/9781108644181.

[15] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[16] Andrew P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

[17] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[18] Randal E Bryant. Graph-based algorithms for boolean function manipulation. *Computers, IEEE Transactions on*, 100(8):677–691, 1986.

[19] Peter Bühlmann. Bootstraps for time series. *Statistical science*, pages 52–72, 2002.

[20] Federico Carli, Manuele Leonelli, Eva Riccomagno, and Gherardo Varando. The r package stagedtrees for structural learning of stratified staged trees. *arXiv preprint arXiv:2004.06459*, 2020.

[21] Federico Carli, Manuele Leonelli, and Gherardo Varando. A new class of generative classifiers based on staged tree models. *arXiv preprint arXiv:2012.13798*, 2020.

[22] Federico Carli, Elena Pesce, Eva Riccomagno, and Andrea Mazza. Combination of autoregressive graphical models and time series bootstrap methods for marine losses forecast: a pilot study. 2021.

[23] Graeme Chamberlin. Methods explained: Temporal disaggregation. *Economic & Labour Market Review*, 4(11):106–121, 2010.

[24] Gregory C Chow and An-loh Lin. Best linear unbiased interpolation, distribution, and extrapolation of time series by related series. *The review of Economics and Statistics*, pages 372–375, 1971.

[25] Rodrigo A Collazo and Pier-Giovanni Taranti. ***ceg**: Chain Event Graph*, 2017. URL `https://CRAN.R-project.org/package=ceg`. `R` package version 0.1.0.

[26] Rodrigo A Collazo, Jim Q Smith, et al. A new family of non-local priors for chain event graph model selection. *Bayesian Analysis*, 11(4):1165–1201, 2016.

[27] Rodrigo A Collazo, Christiane Görgen, and Jim Q Smith. *Chain event graphs*. CRC Press, 2018.

[28] Robert G Cowell, Jim Q Smith, et al. Causal discovery through map selection of stratified chain event graphs. *Electronic Journal of Statistics*, 8(1):965–997, 2014.

[29] Gabor Csardi and Tamas Nepusz. The **igraph** software package for complex network research. *InterJournal*, Complex Systems:1695, 2006. URL `http://igraph.org`.

[30] Estela Bee Dagum and Pierre A Cholette. *Benchmarking, temporal distribution, and reconciliation methods for time series*, volume 186. Springer Science & Business Media, 2006.

[31] Gianpiero Dalla Zuanna, Marcantonio Caltabiano, Alessandra Minello, and Daniele Vignoli. Catching up! the sexual opinions and behaviour of italian students (2000-2017). Technical report, DISIA Working Paper, 2019, 2019.

[32] Adnan Darwiche. *Modeling and reasoning with Bayesian networks*. Cambridge university press, 2009.

[33] Robert J MacG Dawson. The "unusual episode" data revisited. *Journal of Statistics Education*, 3(3), 1995.

[34] Frank T Denton. Adjustment of monthly or quarterly series to annual totals: an approach based on quadratic minimization. *Journal of the American Statistical Association*, 66(333):99–102, 1971.

[35] Claus Dethlefsen and Søren Højsgaard. A common platform for graphical models in `R`: The **gRbase** package. *Journal of Statistical Software*, 14(17): 1–12, 2005. URL `http://www.jstatsoft.org/v14/i17/`.

[36] Holger Dette and Rafael Weißbach. A bootstrap test for the comparison of nonlinear time series. *Computational statistics & data analysis*, 53(4):1339–1349, 2009.

[37] Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine learning*, 29(2):103–130, 1997.

[38] Eliana Duarte and Liam Solus. Algebraic geometry of discrete interventional models. *arXiv:2012.03593*, 2020.

[39] William TM Dunsmuir, David J Scott, et al. The glarma package for observation-driven time series regression of counts. *Journal of Statistical Software*, 67(7):1–36, 2015.

[40] Bradley Efron. Bootstrap methods: another look at the jackknife. In *Breakthroughs in statistics*, pages 569–593. Springer, 1992.

[41] Roman Eisner, Cynthia Stretch, Thomas Eastman, Jianguo Xia, David Hau, Sambasivarao Damaraju, Russell Greiner, David S Wishart, and Vickie E Baracos. Learning to predict cancer-associated skeletal muscle wasting from 1 h-nmr profiles of urinary metabolites. *Metabolomics*, 7(1):25–34, 2011.

[42] Norman Fenton and Martin Neil. *Risk assessment and decision analysis with Bayesian networks*. Crc Press, 2012.

[43] Roque B Fernandez. A methodological note on the estimation of time series. *The Review of Economics and Statistics*, 63(3):471–476, 1981.

[44] M Julia Flores, José A Gámez, and Ana M Martínez. Supervised classification with bayesian networks: A review on models and applications. *Intelligent data analysis for real-life applications: theory and practice*, pages 72–102, 2012.

[45] Guy Freeman and Jim Q Smith. Dynamic staged trees for discrete multivariate time series: Forecasting, model selection and causal analysis. *Bayesian Analysis*, 6(2):279–305, 2011. doi: 10.1214/11-BA610.

[46] Guy Freeman and Jim Q Smith. Bayesian map model selection of chain event graphs. *Journal of Multivariate Analysis*, 102(7):1152–1165, 2011.

[47] Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine learning*, 29(2):131–163, 1997.

[48] Alessandro Gabbiadini, Christina Sagioglou, and Tobias Greitemeyer. Does pokémon go lead to a more physically active life style? *Computers in Human Behavior*, 84:258–263, 2018. doi: 10.1016/j.chb.2018.03.005.

[49] Dan Geiger and David Heckerman. Knowledge representation and inference in similarity networks and bayesian multinets. *Artificial Intelligence*, 82(1-2): 45–74, 1996.

[50] Joseph Glaz and Cristina P Sison. Simultaneous confidence intervals for multinomial proportions. *Journal of Statistical Planning and Inference*, 82(1-2): 251–262, 1999.

[51] Leo A Goodman. On simultaneous confidence intervals for multinomial proportions. *Technometrics*, 7(2):247–254, 1965.

[52] Christiane Görgen and Manuele Leonelli. Model-preserving sensitivity analysis for families of gaussian distributions. *arXiv preprint arXiv:1809.10794*, 2018.

[53] Christiane Görgen, Manuele Leonelli, and Jim Q Smith. A differential approach for staged trees. In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 346–355. Springer-Verlag, 2015.

[54] Christiane Görgen, Anna Bigatti, Eva Riccomagno, and Jim Q Smith. Discovery of statistical equivalence classes using computer algebra. *International Journal of Approximate Reasoning*, 95:167–184, 2018.

[55] Christiane Görgen, Jim Q Smith, et al. Equivalence classes of staged trees. *Bernoulli*, 24(4A):2676–2692, 2018.

[56] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108, 1979.

[57] Jonas Haslbeck and Lourens Waldorp. mgm: Estimating time-varying mixed graphical models in high-dimensional data. *Journal of Statistical Software,*

Articles, 93(8):1–46, 2020. ISSN 1548-7660. doi: 10.18637/jss.v093.i08. URL `https://www.jstatsoft.org/v093/i08`.

[58] Trevor Hastie and Junyang Qian. Glmnet vignette. *Retrieved June*, 9(2016): 1–30, 2014.

[59] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[60] Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical learning with sparsity: the lasso and generalizations*. CRC press, 2015.

[61] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

[62] Søren Højsgaard, David Edwards, and Steffen Lauritzen. *Graphical models with R*. Springer Science & Business Media, 2012.

[63] Søren Højsgaard et al. Graphical independence networks with the gRain package for R. *Journal of Statistical Software*, 46(10):1–26, 2012.

[64] Manfred Jaeger. Probabilistic decision graphs - combining verification and ai techniques for probabilistic inference. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 12(supp01):19–42, 2004.

[65] Manfred Jaeger, Jens D Nielsen, and Tomi Silander. Learning probabilistic decision graphs. *International Journal of Approximate Reasoning*, 42(1-2): 84–100, 2006.

[66] Roger W Johnson. Fitting percentage of body fat to simple body measurements. *Journal of Statistics Education*, 4(1), 1996.

[67] Gunzo Kawamura, Tatsuro Matsuoka, Takayuki Tajiri, Masataka Nishida, and Mitsuru Hayashi. Effectiveness of a sugarcane-fish combination as bait in trapping swimming crabs. *Fisheries Research*, 22(1-2):155–160, 1995.

[68] Claire Keeble, Peter A Thwaites, Stuart Barber, Graham Law, and Paul Baxter. Adaptation of chain event graphs for use with case-control studies in epidemiology. *The International Journal of Biostatistics*, 13(2), 2017. doi: 10.1515/ijb-2016-0073.

[69] Claire Keeble, Peter A Thwaites, Paul Baxter, Stuart Barber, Roger Parslow, and Graham Law. Learning through chain event graphs: The role of maternal factors in childhood type 1 diabetes. *American Journal of Epidemiology*, 186 (10):1204–1208, 2017. doi: 10.1093/aje/kwx171.

[70] Eamonn J Keogh and Michael J Pazzani. Learning the structure of augmented bayesian classifiers. *International Journal on Artificial Intelligence Tools*, 11 (04):587–601, 2002.

[71] Anna Klimova and Tamás Rudas. Testing the fit of relational models. *Communications in Statistics-Theory and Methods*, pages 1–20, 2021.

[72] Anna Klimova, Tamás Rudas, and Adrian Dobra. Relational models for contingency tables. *Journal of multivariate analysis*, 104(1):159–173, 2012.

[73] Anna Klimova, Tamás Rudas, et al. On the role of the overall effect in exponential families. *Electronic Journal of Statistics*, 12(2):2430–2453, 2018.

[74] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[75] Sourab Kumar, Jyothi Tadakamadla, and Newell Johnson. Effect of toothbrushing frequency on incidence and increment of dental caries: a systematic review and meta-analysis. *Journal of dental research*, 95(11):1230–1236, 2016.

[76] Pat Langley and Stephanie Sage. Induction of selective bayesian classifiers. In *Uncertainty Proceedings 1994*, pages 399–406. Elsevier, 1994.

[77] Steffen L Lauritzen. *Graphical models*, volume 17. Clarendon Press, 1996.

[78] Steffen L Lauritzen and Nanny Wermuth. Graphical models for associations between variables, some of which are qualitative and some quantitative. *The annals of Statistics*, pages 31–57, 1989.

[79] Steffen L Lauritzen and Piotr Zwiernik. Locally associated graphical models. *arXiv preprint arXiv:2008.04688*, 2020.

[80] Manuele Leonelli. Sensitivity analysis beyond linearity. *International Journal of Approximate Reasoning*, 113:106–118, 2019.

[81] Manuele Leonelli, Christiane Görgen, and Jim Q Smith. Sensitivity analysis in multilinear probabilistic models. *Information Sciences*, 411:84–97, 2017.

[82] Fraser Lewis, Adam Butler, and Lucy Gilbert. A unified approach to model selection using the likelihood ratio test. *Methods in Ecology and Evolution*, 2 (2):155–162, 2011.

[83] Charles X Ling and Huajie Zhang. The representational power of discrete bayesian networks. *Journal of Machine Learning Research*, 3(Dec):709–721, 2002.

[84] Robert B. Litterman. A random walk, markov model for the distribution of time series. *Journal of Business Economic Statistics*, 1(2):169–173, 1983.

[85] J Scott Long. The origins of sex differences in science. *Social forces*, 68(4): 1297–1316, 1990.

[86] Helmut Lütkepohl. *New introduction to multiple time series analysis*. Springer Science & Business Media, 2005.

[87] Cyrus R Mehta and Nitin R Patel. A network algorithm for performing fisher's exact test in r× c contingency tables. *Journal of the American Statistical Association*, 78(382):427–434, 1983.

[88] Bojan Mihaljevic, Concha Bielza, and Pedro Larrañaga. *bnclassify: learning discrete Bayesian network classifiers from data*, 2018. R package version 0.4.0.

[89] Marvin Minsky. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30, 1961.

[90] Steffen Moritz and Thomas Bartz-Beielstein. imputets: time series missing value imputation in r. *R J.*, 9(1):207, 2017.

[91] Linda Möstel, Marius Pfeuffer, and Matthias Fischer. Statistical inference for markov chains with applications to credit risk. *Computational Statistics*, pages 1–26, 2020.

[92] Allan H Murphy. A new vector partition of the probability score. *Journal of applied Meteorology*, 12(4):595–600, 1973.

[93] John Ashworth Nelder and Robert WM Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society: Series A (General)*, 135(3):370–384, 1972.

[94] Ryan O'Donnell. *Analysis of boolean functions*. Cambridge University Press, 2014.

[95] Judea Pearl. Fusion, propagation, and structuring in belief networks. *Artificial intelligence*, 29(3):241–288, 1986.

[96] Johan Pensar, Henrik Nyman, Timo Koski, and Jukka Corander. Labeled directed acyclic graphs: a generalization of context-specific independence in directed graphical models. *Data mining and knowledge discovery*, 29(2):503–533, 2015.

[97] Johan Pensar, Henrik Nyman, Jarno Lintusaari, and Jukka Corander. The role of local partial independence in learning of bayesian networks. *International journal of approximate reasoning*, 69:91–105, 2016.

[98] Nigel B Pitts, Ramon J Baez, Carolina Diaz-Guillory, Kevin J Donly, Carlos Alberto Feldens, Colman McGrath, Prathip Phantumvanit, W Kim Seow, Nikolai Sharkov, Yupin Songpaisan, et al. Early childhood caries: Iapd bangkok declaration. *Journal of dentistry for children (Chicago, Ill.)*, 86(2): 72, 2019.

[99] Eva Riccomagno and Jim Q Smith. Identifying a cause in models which are not simple bayesian networks. In *Proceedings of the 10th Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 1315–1322, 2004.

[100] Rosa F Ropero, Silja Renooij, and Linda C Van der Gaag. Discretizing environmental data for learning bayesian-network classifiers. *Ecological Modelling*, 368:391–403, 2018.

[101] Christoph Sax and Peter Steiner. tempdisagg: Methods for temporal disaggregation and interpolation of time series, 2013. *URL: http://CRAN. R-project. org/package= tempdisagg. R package version 0.22.[p80]*, 2013.

[102] Gideon Schwarz et al. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

[103] Marco Scutari. Learning bayesian networks with the bnlearn r package. *arXiv preprint arXiv:0908.3817*, 2009.

[104] Aditi Shenvi, Jim Q Smith, Robert Walton, and Sandra Eldridge. Modelling with non-stratified chain event graphs. In *Bayesian Statistics and New Generations*, pages 155–163. Springer International Publishing, 2019.

[105] Tomi Silander and Tze-Yun Leong. A dynamic programming algorithm for learning chain event graphs. In *International Conference on Discovery Science*, pages 201–216. Springer, 2013.

[106] Jim Q Smith. *Bayesian decision analysis: principles and practice*. Cambridge University Press, 2010.

[107] Jim Q Smith and Paul E Anderson. Conditional independence and chain event graphs. *Artificial Intelligence*, 172(1):42–68, 2008.

[108] Donald F Specht. Probabilistic neural networks. *Neural networks*, 3(1):109–118, 1990.

[109] David J Spiegelhalter and Steffen L Lauritzen. Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 20(5):579–605, 1990.

[110] Milan Studeny. *Probabilistic conditional independence structures*. Springer Science & Business Media, 2006.

[111] Peter A Thwaites. Causal identifiability via chain event graphs. *Artificial Intelligence*, 195:291–315, 2013.

[112] Peter A Thwaites and Jim Q Smith. A separation theorem for chain event graphs. *arXiv preprint arXiv:1501.05215*, 2015.

[113] Peter A Thwaites and Jim Q Smith. A new method for tackling asymmetric decision problems. *International Journal of Approximate Reasoning*, 88:624–639, 2017.

[114] Peter A Thwaites and Jim Q Smith. A graphical method for simplifying bayesian games. *Reliability Engineering & System Safety*, 179:3–11, 2018.

[115] Peter A Thwaites, Jim Q Smith, and Eva Riccomagno. Causal analysis with chain event graphs. *Artificial Intelligence*, 174(12-13):889–909, 2010.

[116] Peter A Thwaites, Jim Q Smith, and Robert G Cowell. Propagation using chain event graphs. *arXiv preprint arXiv:1206.3293*, 2012.

[117] Alessandro Ugolini, Federico Carli, Paola Agostino, Armando Silvestrini-Biavati, and Eva Riccomagno. Probabilistic graphical modelling of early childhood caries development. 2021.

[118] Gherardo Varando, Concha Bielza, and Pedro Larrañaga. Decision boundary for discrete Bayesian network classifiers. *Journal of Machine Learning Research*, 16:2725–2749, 2015.

[119] Gherardo Varando, Concha Bielza, and Pedro Larrañaga. Decision functions for chain classifiers based on Bayesian networks for multi-label classification. *International Journal of Approximate Reasoning*, 68:164–178, 2016.

[120] Gherardo Varando, Federico Carli, Manuele Leonelli, and Eva Riccomagno. *stagedtrees: Staged Event Trees*, 2020. URL `https://github.com/gherardovarando/stagedtrees`. R package version 2.2.0.

[121] Gherardo Varando, Federico Carli, and Manuele Leonelli. Staged trees and asymmetry-labeled dags. 2021.

[122] Richard M Vogel and Amy L Shallcross. The moving blocks bootstrap versus parametric time series models. *Water Resources Research*, 32(6):1875–1882, 1996.

[123] Abraham Wald. Tests of statistical hypotheses concerning several parameters when the number of observations is large. *Transactions of the American Mathematical society*, 54(3):426–482, 1943.

[124] William WS Wei and Daniel O Stram. Disaggregation of time series models. *Journal of the Royal Statistical Society: Series B (Methodological)*, 52(3):453–467, 1990.

[125] Joe Whittaker. *Graphical models in applied multivariate statistics*. Wiley Publishing, 2009.

[126] Rachel L Wilkerson and Jim Q Smith. Bayesian diagnostics for chain event graphs. *arXiv preprint arXiv:1910.04679*, 2019.

[127] Edwin B Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927.

[128] Sebastian Wójcik. Temporal disaggregation of time series with regularization term. *Barometr Regionalny. Analizy i prognozy*, 3:183–188, 2016.

[129] Achim Zeileis, Jason C. Fisher, Kurt Hornik, Ross Ihaka, Claire D. McWhite, Paul Murrell, Reto Stauffer, and Claus O. Wilke. colorspace: A toolbox for

manipulating and assessing colors and palettes. arXiv 1903.06490, arXiv.org E-Print Archive, March 2019. URL http://arxiv.org/abs/1903.06490.