Selections of vine structures and their applications

Zhu, K.

DOI
10.4233/uuid:84ba9f5e-2c5c-4280-9789-35fd650fc617

Publication date
2022

Document Version
Final published version

Citation (APA)
Zhu, K. (2022). *Selections of vine structures and their applications*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:84ba9f5e-2c5c-4280-9789-35fd650fc617

SELECTIONS OF VINE STRUCTURES AND THEIR APPLICATIONS

SELECTIONS OF VINE STRUCTURES AND THEIR APPLICATIONS

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus Prof.dr.ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates
to be defended publicly on
Wednesday 6 July 2022 at 15:00 o'clock

by

Kailun ZHU

Master of Science in Financial Engineering,
National University of Singapore, Singapore,
born in Huzhou, Zhejiang, China.

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Dr. D. Kurowicka	Delft University of Technology, promotor
Dr. ir. G.F. Nane	Delft University of Technology, copromotor
Independent members:	
Prof. dr. ir. G. Jongbloed	Delft University of Technology
Prof. dr. T.J. Bedford	University of Strathclyde, Glasgow
Prof. dr. I. van Keilegom	Katholieke Universiteit Leuven
Prof. dr. T.W. Nagler	Ludwig-Maximilians-Universität München
Prof. dr. A. Papapantoleon	Delft University of Technology, reserve member

An electronic version of this dissertation is available at
http://repository.tudelft.nl/.

Science is a wonderful thing
if one does not have to earn one's living at it.

Albert Einstein

CONTENTS

SUMMARY

Copulas are important models that allow to capture the dependence among variables. There are many types of bivariate parametric copula families, which allow to model data sets with different properties: symmetric and asymmetric dependence, upper (lower) tail dependence. In higher dimensions popular families of copulas, e.g., Gaussian, Student-t and canonical Archimedean are not sufficiently flexible in representing different types of dependence that they can realize. By decomposing the multivariate copula into a sequence of bivariate (conditional) copulas, based on a graph called vine (which is a nested set of trees), one is able to construct a n dimensional copula with the bivariate copulas that can have different types of dependence (e.g., tail behavior and asymmetries). The model constructed this way is called the vine copula model.

In order to estimate a vine copula model, it is required to specify 1) a vine structure; 2) the bivariate copula family for each conditional copula assigned to edges of the vine; 3) the copula parameters. The estimation is performed tree-by-tree in order to reduce computational complexity, and often the vine is considered in the simplified form, where all the conditional copulas are assumed not to depend directly on the conditioning variables. Even if theoretically any vine structure used in the density decomposition represents the same density, different vine structures will lead to different performance in practice, due to, the estimation procedure of the vine copula model and the simplifying assumption. There are $\binom{n}{2}(n-2)!2^{\binom{n-2}{2}}$ vine structures on n variables, hence it is not possible to estimate all of them in high dimensions.

In this thesis, contributions to the problem of vine structure selection are presented. At first, different vine representations are discussed in Chapter 2. The tree-wise, the vine triangular array and the vine matrix representations have been used in the literature. A new vine representation, called vine binomial tree, is proposed. With this representation a vine structure can be determined by any given order of variables and a sequence of binary numbers. Then for the remaining part of the thesis, different ways of vine structure constructions are proposed in various applications.

The main topic in Chapter 3 is the search for a vine structure with the best performance for given data. The most popular heuristic of vine structure construction is the Dißmann's heuristic. It constructs the vine structure tree-wise, starting from the first tree, along with the copula family selection and copula parameter estimation. Each tree structure is determined by a maximum spanning tree with weights being the absolute Kendall's tau. However, there is no guarantee that this heuristic will result in a structure with the optimal performance. Hence, it is proposed to search for a number of additional vines that are 'sufficiently' different from the initial one (the one constructed based on the heuristic), to see if a better model can be found. The number of common sampling orders has been shown to be a good proxy to measure the difference between two vine structures. We proposed to search for better performing structures within vines having 2 common sampling orders with the initial structure. Several algorithms are proposed to

construct vines having 2 sampling orders in common with a given initial vine structure. They have been applied in synthetic patient data generation in Chapter 4, when mixed continuous and discrete data set is considered.

For regression problems in estimating a conditional distribution of response variables given covariates, studied in Chapter 5, vine copula model can be applied by constructing a vine structure so that it contains the response variable in the conditioned set of the top node. Then the conditional distribution is given in analytic form. Additional constraints for the vine structure should be included when the joint conditional distribution of two response variables is required in analytic form. The vine based regression approach has been applied in a stress analysis of the manufacturing industry.

A vine is in principle a fully connected graph. It can be simplified by introducing conditional independence, which is to assign independence copula to edges of the vine. In Chapter 6, a relationship between an m-saturated vine (a vine with some nodes removed, or assigned with independence copula) and a strongly chordal graph (a special type of undirected graph) is proved. Several algorithms are proposed to construct an m-saturated vine corresponding to a strongly chordal graph. Due to the specification of conditional independence, it becomes possible to assess all vine structures for the subvines in the m-saturated vine and/or estimate non-simplified vines.

SAMENVATTING

Copulas zijn belangrijke modellen voor het vastleggen van afhankelijkheden tussen variabelen. Er zijn vele soorten bivariate parametrische copula families die het mogelijk maken om datasets met uiteenlopende eigenschappen te modelleren. In het bijzonder symmetrisch en asymmetrische afhankelijkheid, rechter (linker) staart afhankelijkheid. Voorbeelden van populaire families van copulas in hogere dimensies, zoals de Gaussische, Student-t en canonieke Archimedes zijn niet voldoende flexibel genoeg in het weergeven van alle potentieel realiseerbare afhankelijkheden. Door middel van het ontbinden van de multivariabele copula in een rij van bivariate (conditionele) copulas, gebaseerd op een graaf genaamd vine (een geneste verzameling van bomen), is men in staat om een n-dimensionale copula te construeren met de bivariate copulas, met verschillende soort afhankelijkheden (zoals staart en asymmetrische afhankelijkheid). Het model dat op deze manier is geconstrueerd wordt ookwel het vine copula model genoemd.

Voor het schatten van een vine copula model is het vereist het volgende te specificeren, 1) een vine structuur; 2) de bivariate copula familie voor iedere conditionele copula toegewezen aan de takken van de vine structuur; 3) de copula parameters. De schatting is boom per boom uitgevoerd om de computationele complexiteit te reduceren, bovendien is de vine meestal beschouwd in z'n meest simplistische vorm, hier wordt aangenomen dat alle conditionele copulas niet direct conditioneel afhangen van de variabelen. Zelfs als theoretisch gezien iedere vine structuur, die gebruikt wordt in de decompositie van de verdeling, dezelfde dichtheid representeert, kan het zijn dat de vine structuren leiden tot verschillende prestaties in de praktijk. Reden hiervan is bijvoorbeeld de schattingsprocedure van het vine copula model en de aannames die zijn versimpeld. Er zijn $\binom{n}{2}(n-2)!2^{\binom{n-2}{2}}$ vine structuren op n variabelen, en dus is het niet mogelijk om ze allemaal te schatten in hogere dimensies.

In deze scriptie worden de contributies in vine structuur selectie gepresenteerd. Om mee te beginnen worden de verschillende vine representaties besproken in hoofdstuk 2. The boom methode, de driehoekige matrix en de vine matrix reprensentaties zijn al eerder gebruik in de bekende literatuur. Een nieuwe vine representatie, genaamd vine binomiaal boom, wordt voorgesteld. Met behulp van deze representatie kan een vine structuur bepaald worden door elke willekeurig gegeven volgorde van variabelen en een rij van binaire getallen. In het resterende deel van de scriptie worden er diverse technieken voorgesteld voor vine structuur constructies op uiteenlopende toepassingen.

Het voornaamste onderwerp, aan bod komend in hoofdstuk 3, is hoe er gezocht moet worden naar vine structuren met de beste prestaties voor de gegeven data. De meest populaire heuristiek voor het construeren van vine structuren is de Dißmann heuristiek. De Dißmann heuristiek maakt de vine structuur met per boom, beginnend vanaf de eerste boom, samen met de copula familie selectie en de copula parameter schatting. Iedere boom structuur wordt vastgesteld door een maximaal opspannende boom met

factoren gelijk aan de absolute Kendall tau. Echter is er geen garantie dat deze heuristiek de structuur met optimale prestaties zal verschenken. Daarom is het aangedragen om te zoeken naar een tal additionele vines die 'genoeg' verschillen van de initiële (diegene geconstrueerd op basis van de heuristiek), om te checken of er een beter presterend model kan worden gerealiseerd. Het aantal gemeenschappelijke sampling volgordes is gebleken een goede indicator te zijn voor het aanduiden van de ongelijksoortigheid van twee vine structuren. We stellen voor om te zoeken naar beter presterende structuren binnen vines die 2 sampling volgordes gemeen hebben binnen de initiële structuur. Verschillende algoritme worden voorgesteld voor het samenstellen van vines die 2 sampling volgordes gemeen hebben. Het zoeken naar vines met 2 gemeenschappelijke sampling volgordes is toegepast voor het genereren van synthetische patiëntgegevens in hoofdstuk 4, waar een dataset met zowel continue als discrete data wordt bekeken.

Voor regressie problemen waar de conditionele verdeling van responsvariabelen gegeven de covariaten wordt geschat, onderzocht in hoofdstuk 5, kan men een vine copula model toepassen door het samenstellen van de vine structuur zodanig dat de structuur de responsvariabelen bevat in de geconditioneerd verzameling in de bovenste node. Dan wordt de voorwaardelijke kansverdeling geven in analytische vorm. Additionele eisen voor de vine structuur moeten worden toegevoegd wanneer de gemeenschappelijke voorwaardelijke kansverdeling van twee responsvariabelen een analytische vorm heeft. De op vine gebaseerde regressie methode is al toegepast in stress analyse voor de fabrieksindustrie.

Een vine is in principe een complete graaf. Het kan versimpeld worden door het introduceren van voorwaardelijke onafhankelijkheid, wat betekend het toekennen van copula onafhankelijk aan de takken van de vine. In hoofdstuk 6, wordt een relatie tussen de m-verzadigde vine (een vine met een aantal takken verwijderd, of met toegekende copula onafhankelijkheid) en de sterk chordale graaf (een speciaal soort ongerichte graaf) bewezen. Diverse algoritmen zijn voorgesteld voor het construeren van een m-verzadigde vine overeenkomstig met een sterk chordale graaf. Vanwege de specificatie van de voorwaardelijke onafhankelijkheid, wordt het mogelijk om alle vine structuren te beoordelen voor de subvines in de m-verzadigde vine en of het schatten niet versimpelde vines.

1

INTRODUCTION

Dependence modeling can be carried out by applying copula functions, where the dependence is estimated separately from the marginal distributions. However, dependence structures represented by a single copula family are limited, and this problem becomes severe in high dimensions. By decomposing the multivariate copula function into a sequence of bivariate copulas based on a graph called vine, one is able to capture flexibly different kinds of dependence structures.

1

1.1. COPULAS

Modeling dependencies is very important in many applications. A simple example from the area of finance could be an investment problem, where ignoring the dependence among stock returns might lead to a not optimal investment strategy and ultimately loss of money. In health studies, due to legal and privacy restrictions, data of real patients cannot be disseminated freely. Synthetic data generation provides a way to allow broad access to patient data. The dependence among patients' characteristics have to be modeled correctly so the generated data recovers the properties of real patients. Finally, in engineering application, one might want to evaluate the safety of a new car design and evaluate its structural behavior during crashing. In this case, the dependence among the so called firewall points (metal plates that suppose to protect passengers when accident happens) is of interest. In all these examples the dependence structure of the characteristics of interest has to be modeled. Analyzing and modeling dependence structure can be achieved in many different ways. A good overview can be found, e.g., in Joe (1997).

More formally the problem to be solved is to specify a joint distribution $F_{12\ldots n}$ for the observations of n random variables $X_i, i = 1,\ldots,n$. The most popular approach is to fit a multivariate Gaussian distribution. Obviously, if one observes from the data properties as, asymmetry, heavy tails or tail dependence, the normality assumption is violated and different types of models have to be considered. For example in finance, it has been observed that the daily log-returns of stocks are heavy-tailed distributed and their dependence exhibits asymmetries and tail dependence (McNeil et al. (2010)). In medicine similar reasons can also be found for suggesting not to model with multivariate Gaussian distribution for different characteristics of patients, such as age, weight, sex and etc..

The Sklar's theorem (Sklar (1959)) below provides a more flexible alternative to model the joint distribution.

Theorem 1.1.1 (Sklar's theorem)**.** *For n random variables $X_i, i = 1,\ldots,n$ with joint distribution function $F_{12\ldots n}$ and univariate distributions $F_i(x_i)$, their joint distribution can be expressed as*

$$F_{12\ldots n}(x_1, x_2, \ldots, x_n) = C_{12\ldots n}(F_1(x_1), F_2(x_2), \ldots, F_n(x_n)),$$

where $C_{12\ldots n}$ is a distribution function on the unit hypercube $[0,1]^n$, called a copula.

When all the variables are continuous the copula is unique whereas for discrete variables the copula is uniquely determined on the support of the variables. Denote the univariate density of the continuous variable X_i as $f_i(x_i)$, and the joint density of variables $X_1,\ldots,X_n$ as $f_{12\ldots n}(x_1, x_2, \ldots, x_n)$. If the copula density $c_{12\ldots n}(u_1, u_2, \ldots, u_n)$ (where $u_i = F_i(x_i)$) exists then the joint density is

$$f_{12\ldots n}(x_1, x_2, \ldots, x_n) = c_{12\ldots n}(u_1, u_2, \ldots, u_n) f_1(x_1) f_2(x_2) \cdots f_n(x_n).$$

In Figure 1.1 examples of contour plots of six different bivariate copula families with standard normal margins are shown. Each bivariate copula (except for the independence copula) represents the same overall dependence measured by Kendall's tau equal

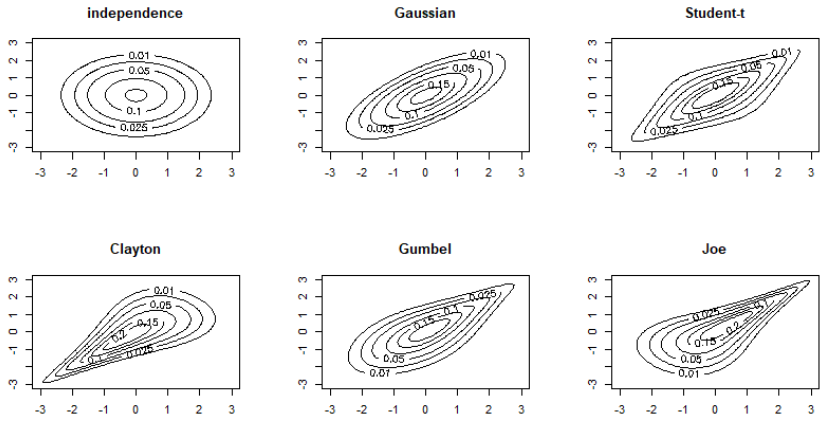

Figure 1.1: Contour plot of different bivariate copula families (marginal distributions are transformed into standard normal distribution for the purpose of visualization) where the Kendall's tau is always 0.5 (except for the independence copula).

to 0.5, where Kendall's tau is a rank correlation measure of dependence and can be calculated for the variables with copula C_{12} as:

$$\tau = 4 \int \int_{[0,1]^2} C_{12}(u_1, u_2) dC_{12}(u_1, u_2) - 1.$$

Even if the overall dependence, of the shown copula families, is the same, some of their properties are different. All the copula families shown in Figure 1.1 capture symmetric dependence but the dependence represented by Clayton, Gumbel and Joe copulas is not radial symmetric. Gumbel copula has upper tail dependence and lower tail independence, and Clayton copula, in contrast, exhibits lower tail dependence and upper tail independence. There are many parametric copula families that are used to capture different properties observed in data (Nelsen (2006), Joe (2014), Hofert et al. (2018)). In this thesis we will use the following parametric copulas, Gaussian (N), Student-t (t), Clayton (C), Gumbel (G), Joe (J), BB1, BB6, BB7 and BB8 copulas, and also their rotated versions which are referred to using the following convention: G180 represents a Gumbel copula rotated by 180 degrees in anti-clockwise direction. Hence this copula can capture lower tail dependence and upper tail independence.

Copulas have been proved to be a useful tool in many applications, e.g. in finance (Cherubini et al. (2004), McNeil et al. (2010)). In higher dimensions, however, copulas corresponding to known multivariate distributions are limited in types of dependence structures that they can represent. For example, the Gaussian copula does not allow modeling tail dependence or asymmetric dependence. By decomposing the multivariate copula function into a series of bivariate (conditional) copulas one can construct a multivariate copula with very rich set of properties. This approach is called the pair copula constructions and was introduced at first in Joe (1997). It became known as a vine copula model in Cooke (1997), Bedford & Cooke (2001, 2002). This thesis contributes to

the theory and applications of vine copula model. In the next sections the basic definitions and properties of the vine copula model are presented.

1.2. VINES

In Cooke (1997), Bedford & Cooke (2001, 2002), a decomposition based on a graphical structure called regular vine (composed of a set of trees) has been introduced. This model allows to construct a high dimensional distribution by combining bivariate and conditional bivariate copulas arranged according to a vine structure. Any type of bivariate copula can be used in this construction and there are no algebraic constrains on the parameters of these copulas. Moreover, non-parametric copulas are also allowed in the construction. A detailed introduction to the vine copula model can be found in Kurowicka & Joe (2011), Czado (2019), Czado & Nagler (2022). The flexibility of the vine copula models led to an enormous volume of applications, e.g., in finance (see Aas (2016) and the reference therein), in engineering (e.g., Kurowicka & Joe (2011), Schepsmeier & Czado (2016), Li et al. (2021)) and in health studies (e.g., Stöber et al. (2015), Cooke et al. (2019), Hasler et al. (2018)).

In this section, the main definitions concerning vines and their basic properties are presented. Then it is explained how a joint distribution is represented with the vine copula model.

1.2.1. REGULAR VINES

A regular vine on n elements ($V(n)$) or on element set $\mathcal{V}_n = \{v_1, \ldots, v_n\}$ ($V(\mathcal{V}_n)$), is a nested set of connected trees $V = \{T_1, ..., T_{n-1}\}$ where the edges of tree T_j are the nodes of tree T_{j+1}, $j = 1, \ldots, n-2$ and such that two edges in tree T_j are joined by an edge in tree T_{j+1} only if these edges share a common node, $j = 1, ..., n-2$. The formal definition is as follows.

Definition 1.2.1 (Regular vine)**.** *V is a regular vine if*

1. $V = \{T_1, \ldots, T_{n-1}\}$,

2. *T_1 is a connected tree with nodes $N_1 = \mathcal{V}(T_1)$, and edges $E_1 = \mathcal{E}(T_1)$; for $i = 2, \ldots, n-1$ T_i is a tree with nodes $N_i = E_{i-1}$.*

3. *(proximity) For $i = 2, \ldots, n-1$, $\{a, b\} \in E_i, \#a \triangle b = 2$ where $\triangle$ denotes the symmetric difference.*

The proximity condition can be interpreted using graph theory terminology as follows: a node in tree T_i that connects two nodes in tree T_{i-1} is called a **m-parent** (parent in short) node and these two nodes are called the **m-children** (children in short) of the parent node or **siblings**. Since all trees in a vine are connected then each node in T_i, $2 \leq i \leq n-1$ has a sibling, and the proximity condition implies that each node has a common child with its sibling. The node in tree T_i of the vine is the edge of T_{i-1}. Hence for the purpose of completeness the single edge in tree T_{n-1} is referred to as a single node of tree T_n, and $V(n)$ is a set of n trees $V(n) = \{T_1, \ldots, T_n\}$ where T_n is just one node called the **top node** of $V(n)$.

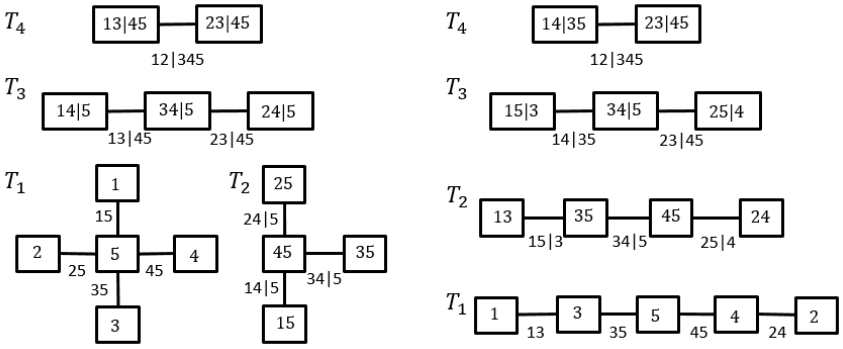

Figure 1.2: Regular vines with 5 elements (left - $V_1(5)$; right - $V_2(5)$).

Regular vines are used as graphical structures that allow the choice of a set of algebraically independent (conditional) bivariate copulas to factorize the joint density. This is done via conditional bivariate constraints associated with each edge in each tree, by using *constraint, conditioned* and *conditioning sets*, which are defined below.

Definition 1.2.2 (Constraint, Conditioned, and Conditioning sets)**.**

1. *For $e \in E_i, i \leq n-1$, the **constraint set** associated with e is the **complete union** U_e^* of e, that is, the subset of $\{1,\dots,n\}$ reachable from e by the membership relation.*

2. *For $i = 1,\dots,n-1, e \in E_i$, if $e = \{j,k\}$ then the **conditioning set** associated with e is*

$$D_e = U_j^* \cap U_k^*$$

*and the **conditioned set** associated with e is*

$$\{C_{e,j}, C_{e,k}\} = \{U_j^* \setminus D_e,\ U_k^* \setminus D_e\}.$$

To illustrate the concepts introduced above, we consider the vine $V_1(5)$ in Figure 1.2 (left) with its trees $T_1,\dots,T_4$. T_1 has nodes $N_1 = \{1,2,3,4,5\}$ and edges $E_1 = \{(1,5),(2,5),(3,5),(4,5)\}$. In tree T_2, we can see that $N_2 = E_1$. Moreover, the constraint sets of the three edges of E_2 are $\{1,4,5\}, \{3,4,5\}, \{2,4,5\}$. The conditioning sets are all $\{5\}$ and the conditioned sets are $\{1,4\},\{3,4\},\{2,4\}$. Nodes in the third tree are named with the convention used for E_2. Next we see that the constraint sets for two edges in E_3 are $\{1,3,4,5\}$ and $\{2,3,4,5\}$, which gives the conditioning sets for both edges equal to $\{4,5\}$ and conditioned sets are $\{1,3\}$ and $\{2,3\}$, respectively. There is always one edge in the last tree, which in the case of $V_1(5)$ has conditioning and conditioned sets equal to $\{3,4,5\}$ and $\{1,2\}$, respectively.

We can see that for node $e \in N_2$ the conditioning set is empty. Often a node e in a regular vine is associated with its conditioned and conditioning sets and write $e = \{C_{e,j}, C_{e,k} | D_e\}$ or simply with its constraint set U_e^*. According to the proximity condition, the cardinality of the conditioned set is always 2 and $C_{e,j}, C_{e,k}$ are called each other **partners**.

The m-descendant/m-ancestor (descendant/ancestor in short) relationship in regular vine is defined as follows,

Definition 1.2.3 (descendant; ancestor)**.** *If a node e is reachable from a node f via the membership relation: $e \in e_1 \in \cdots \in f$, we say that e is an descendant of f and f is an ancestor of e.*

For $V_1(5)$ in Figure 1.2 (left), node 15 is a descendant of node 13|45 and 13|45 is an ancestor of 15 through node 14|5. However 13|45 is an ancestor of neither node 25 nor node 2. We can now introduce a notion of a subvine of $V(\mathcal{V}_n)$. Given a node e with constraint set U_e^* in a regular vine $V(\mathcal{V}_n), U_e^* \subseteq \mathcal{V}_n$, the regular vine $V(U_e^*)$ is called a **subvine** of $V(\mathcal{V}_n)$ induced by the elements U_e^*, if it is the case that the top node of $V(U_e^*)$ is node e and all descendants of e in $V(\mathcal{V}_n)$ belong to $V(U_e^*)$.

The following properties of regular vines have been proved (see, e.g., Kurowicka & Joe (2011)) and can be checked for the two vines in Figure 1.2.

Properties of regular vines:

1. There are $n-1$ trees and $\binom{n}{2}$ edges in a regular vine on n elements.
2. Conditioned sets are doubletons.
3. Each pair appears once as a conditioned set of an edge.
4. There are $i-1$ and $i+1$ elements in the conditioning and constraint sets of an edge of the ith tree, respectively.
5. If two nodes have the same constraint sets, they are the same node.
6. If element i is a member of a conditioned set of an edge e of a regular vine then i is a member of the conditioned set of exactly one of the children of e and the conditioning set of an child is a subset of D_e.
7. If $N_1 = \{x_1, y_1 | A \setminus \{y_1\}\}$ and $N_2 = \{x_2, y_2 | A \setminus \{y_2\}\}$, where $A \subset \{1, 2, ..., n\}$, $x_1, x_2 \notin A$ and $x_1 \neq x_2$ are nodes of tree T_i of regular vine on n elements then N_1 and N_2 have a common child. Moreover if $y_1 \neq y_2$ then this child is: $\{y_1, y_2 | A \setminus \{y_1, y_2\}\}$.

1.2.2. VINE COPULA MODEL

In Bedford & Cooke (2002), the following representation theorem for the joint density as the product of bivariate (conditional) copulas assigned to the nodes of a regular vine on n elements and the marginal densities is proved (arguments of the functions have been suppressed to simplify the notation):

Theorem 1.2.1. *Let (F, V, B) be a copula vine specification where: $F = (F_1, \ldots, F_n)$ and each F_i has density f_i, $i = 1, \ldots, n$, $V(n) = (T_1, \ldots, T_{n-1})$ is a regular vine on n elements and $B = \{C_{jk;D_e} \mid e(j,k) \in \bigcup_{i=1}^{n-1} E_i$, where $e(j,k)$ is the unique edge with conditioned set $\{j,k\}$, and $C_{jk;D_e}$ is a copula for $\{X_j, X_k\}$ conditional on variables in D_e with density $c_{jk;D_e}\}$.*

Then the vine-dependent distribution for (F, V, B) *is uniquely determined and has a density given by*

$$f_{1\ldots n} = f_1 \cdots f_n \prod_{i=1}^{n-1} \prod_{e(j,k)\in E_i} c_{jk;D_e}(C_{j|D_e}, C_{k|D_e}|D_e),$$

where

$$C_{i|A} = \frac{\partial C_{ij;A\setminus\{j\}}(C_{i|A\setminus\{j\}}, C_{j|A\setminus\{j\}}; A\setminus\{j\})}{\partial C_{j|A\setminus\{j\}}}.$$

The conditional distribution $C_{i|A}$ is called the h-function (introduced in Joe (1997)) and can be denoted as $h_{i|j;A\setminus\{j\}}$.

Conversely, it has been shown in Czado (2010) that for continuous random variables $X_1, \ldots, X_n$ with positive joint density $f_{1,\ldots,n}$, marginal densities $f_1, \ldots, f_n$ and conditional densities of the variables $X_k|X_1, \ldots, X_j$ denoted as $f_{k|1,\ldots,j}$, the standard factorization (not unique)

$$f_{1\ldots n} = f_1 f_{2|1} \cdots f_{n-1|1,\ldots,n-2} f_{n|1,\ldots,n-1}$$

can be further rewritten by repeated application of the recursive formula

$$f_{i|A} = c_{ij;A\setminus\{j\}}\left(C_{i|A\setminus\{j\}}, C_{j|A\setminus\{j\}}; A\setminus\{j\}\right) f_{j|A\setminus\{j\}}.$$

where $A \subset \{1, \ldots, n\}$ and $i \notin A$, into products of conditional bivariate copulas on a regular vine.

One usually assumes that the bivariate conditional copula $c_{jk;D_e}(C_{j|D_e}, C_{k|D_e}|D_e)$ does not depend directly on the variables X_{D_e}, hence it becomes $c_{jk;D_e}(C_{j|D_e}, C_{k|D_e})$. This assumption is called the simplifying assumption, which will be explained briefly in Section 1.2.4.

Then, for example, a joint density for five variables can be represented by the vine $V_1(5)$ in Figure 1.2 (left) as follows, where the copula density $c_{jk;D_e}(C_{j|D_e}, C_{k|D_e})$ are abbreviated as $c_{jk;D_e}$,

$$\begin{aligned} f_{1\ldots n} &= f_1 f_5 c_{15} f_4 c_{45} c_{14;5} f_3 c_{35} c_{34;5} c_{13;45} f_2 c_{25} c_{24;5} c_{23;45} c_{12;345} \\ &= f_1 f_2 f_3 f_4 f_5 c_{15} c_{25} c_{35} c_{45} c_{14;5} c_{34;5} c_{24;5} c_{13;45} c_{23;45} c_{12;345}. \end{aligned}$$

For the other vine structure $V_2(5)$, in Figure 1.2 (right), one gets a different density decomposition, which is as follows,

$$\begin{aligned} f_{1\ldots n} &= f_1 f_3 c_{13} f_5 c_{35} c_{15;3} f_4 c_{45} c_{34;5} c_{14;35} f_2 c_{24} c_{25;4} c_{23;45} c_{12;345} \\ &= f_1 f_2 f_3 f_4 f_5 c_{13} c_{35} c_{45} c_{24} c_{15;3} c_{34;5} c_{25;4} c_{14;35} c_{23;45} c_{12;345}. \end{aligned}$$

There are many decompositions of a positive density and theoretically all are equivalent. In practice, however, due to the estimation errors, simplifying assumption etc., one decomposition might lead to a better performance. This causes one of the main challenge of the vine copula models, which is the vine structure selection problem. In the next section we explain in detail the estimation procedure of the vine copula model.

1

1.2.3. ESTIMATION

Estimation of the vine copula model requires that the estimation of each univariate marginal distribution and the dependence structure represented by the sequence of bivariate (conditional) copulas. This involves 1) estimation of the marginal distributions; 2) selection of a vine structure; 3) choice of the copula families for each bivariate (conditional) copula; 4) estimation of the copula parameters. First univariate margins are estimated, then the observations are transformed through probability integral transformation (PIT) to get pseudo observations. These pseudo observations are then used to estimate the vine copula model. This estimation procedure is called the inference functions for margins (IFM) method introduced in Joe (1997).

Let us consider N independent and identically distributed realizations of variables $(X_1,\dots,X_n)$, $(x_1^m, x_2^m,\dots,x_n^m)$, where $m = 1,\dots,N$, which we refer to as being in **x-scale**. The univariate parametric distributions $F_i, i = 1,\dots,n$ are fitted or the empirical cumulative distribution functions $\hat{F}_i(x_i) = \frac{1}{N+1}\sum_{m=1}^{N} \mathbb{1}_{x_i^m \le x_i}$ (other non-parametric methods e.g. kernel smoothing, splines can also be applied) are used to transform data to uniform scale. Non-parametric approach is often preferred Joe (1997), as it reduces estimation errors for parameters of copulas when parametric models of margins are misspecified. The data transformed using PIT called pseudo observations $(u_1^m, u_2^m,\dots,u_n^m)$ is often referred to as data in the **u-scale**. Sometimes it is also advantageous to transform the pseudo observations into data following standard normal distribution, which is referred to as data in **z-scale**.

The pseudo observations, $(u_1^m, u_2^m,\dots,u_n^m)$ are used to find parameters $\boldsymbol{\theta}$ of vine copula model through the maximum likelihood method. The log-likelihood function is as follows,

$$\ell(\boldsymbol{u};\boldsymbol{\theta}) = \sum_{m=1}^{N} \ln\Big(\prod_{i=1}^{n-1}\prod_{e(j,k)\in E_i} c_{jk;D_e}\big(C_{j|D_e}(u_j^m|\boldsymbol{u}_{D_e}^m), C_{k|D_e}(u_k^m|\boldsymbol{u}_{D_e}^m)|\boldsymbol{\theta}\big)\Big).$$

To penalize models with larger number of parameters, we use Akaike information criterion (AIC) (Akaike (1998)) and Bayesian information criterion (BIC) (Schwarz (1978)), which are:

$$\begin{aligned} AIC(\boldsymbol{u}) &= -2\ell(\boldsymbol{u};\boldsymbol{\theta}) + 2|\boldsymbol{\theta}|, \\ BIC(\boldsymbol{u}) &= -2\ell(\boldsymbol{u};\boldsymbol{\theta}) + \ln(N)|\boldsymbol{\theta}|. \end{aligned}$$

where $|\boldsymbol{\theta}|$ denotes the number of parameters of the parametric copulas in the vine copula model. In this thesis, most of the time these two information measures of models' performance are used. Other options could be the generalized information criterion (Konishi & Kitagawa (1996)) or the modified BIC (Nagler, Bumann & Czado (2019)).

To test whether one vine copula model performs significantly better than another, a likelihood ratio type test called the Vuong test Vuong (1989) will be applied. Suppose two regular vine copula models with densities $f^A_{1,\dots,n}(;\boldsymbol{\theta_1})$ and $f^B_{1,\dots,n}(;\boldsymbol{\theta_2})$, respectively, are considered. The ratio of log likelihoods of these models on pseudo observations $\boldsymbol{u}^m$ is

$p_m = \ln \frac{f^A_{1,\dots,n}(\boldsymbol{u}^m;\boldsymbol{\theta_1})}{f^B_{1,\dots,n}(\boldsymbol{u}^m;\boldsymbol{\theta_2})}$, then the test statistic of the Vuong test is

$$v = \frac{\frac{1}{N}\sum_{m=1}^{N} p_m}{\sqrt{\sum_{m=1}^{N}(p_m - \bar{p})^2}}.$$

The test statistic v is shown in Vuong (1989) to follow a standard normal distribution under the null hypothesis that the two models are not significantly different. One can further correct this statistic (in order to choose a more parsimonious model) by applying a Schwarz correction $\ln(N)(\frac{|\boldsymbol{\theta_1}|}{2} - \frac{|\boldsymbol{\theta_2}|}{2})$.

Although the IFM approach reduces the complexity of MLE for parameters of copulas, optimizing the likelihood over all the parameters $\boldsymbol{\theta}$ of the bivariate copulas in vine copula model is still very expensive. Hence, estimation of the vine copula model is implemented tree-by-tree (Aas et al. (2009)). This is to estimate at first each bivariate copula in the first tree based on the pseudo observations. Then the data is transformed using conditional distributions computed from the copulas in the first tree and will be used to estimate the bivariate conditional copulas in the second tree etc.. This iterative procedure is implemented in R in the **VineCopula** (Nagler, Schepsmeier, Stoeber, Brechmann, Graeler & Erhardt (2019)) package and in the **rvinecopulib** (Nagler & Vatter (2021)) package.

The vine copula model can also be applied in the case of continuous-discrete and discrete data sets. The adjustments that need to be made when discrete variables are added to continuous variables are shown in Appendix 1.A.

1.2.4. SIMPLIFYING ASSUMPTION

Vine copulas are applied usually in simplified form (called simplified vine). All conditional copulas are assumed not to depend directly on the variables in the conditioning set. A detailed discussion of the simplifying assumption can be found in Stöber et al. (2013), Spanhel & Kurz (2015), Kurz & Spanhel (2017). The problem of using vines without simplifying assumption stems from the difficulty to observe how these conditional copulas might depend on the conditioning variables. Some attempts to consider non-simplified vines are to fix the copula family but to allow a relationship between the copula parameter (or the corresponding conditional Kendall's tau) and the conditioning variables. This could be non-parametric relationship (Acar et al. (2011)), or described with generalized additive models (Nagler & Vatter (2020)). A discussion about the performance of these two methods can be found in Acar et al. (2019).

Testing the simplifying assumption can also be a difficult task. A test introduced in Kurz (2019), implemented in the **pacotest** package, provides the way to examine if simplified vine copula is appropriate for the data. The simplifying assumption for the whole regular vine copula model is tested by performing separate tests for each conditional copula in this vine. Different partitions of data set based on the conditioning variables are considered and the equality of correlations of conditioned variables on these partitions is checked. For a partition Γ of size L and the pseudo observations $(u_1^m, u_2^m, \dots, u_n^m), m = 1, \dots, N$, the test statistic is

$$T_N(\Gamma) = N(A\hat{R}_\Gamma)^\intercal (A\hat{\Sigma}_\Gamma A^\intercal)^{-1}(A\hat{R}_\Gamma),$$

where $\hat{R}_\Gamma = (\hat{r}_1, \ldots, \hat{r}_L)^\top$ is the vector of sample correlations in each partition, $\hat{\Sigma}_\Gamma$ is a $L \times L$ diagonal matrix with elements $\hat{\Sigma}_\Gamma(l,l) = \hat{\sigma}^2(\hat{r}_l)$ ($\hat{\sigma}^2(\hat{r}_l)$ is the estimate of the asymptotic variance $\sqrt{N}(\hat{r}_l - r_l)$) and A is a $(L-1) \times L$ matrix such that $(A\hat{R}_\Gamma)^\top(A\hat{R}_\Gamma) = \sum_{l=1}^{L-1}(\hat{r}_l - \hat{r}_{l+1})^2$.

It has been shown $T_N(\Gamma)$ follows a χ^2_{L-1} distribution under the null hypothesis that the correlations in each partition are equal. In this thesis only simplified vine copula models are considered.

1.3. IMPORTANCE OF VINE STRUCTURE

Any vine structure can be used to decompose a density. Theoretically it does not matter which vine structure is used. In practice, however, due to the limited choice of parametric bivariate copula families, tree-wise estimation procedure (explained in Section 1.2.3) as well as the simplifying assumption (explained in Section 1.2.4), the performance of different vine copula models might vary. There are $\binom{n}{2}(n-2)!2^{\binom{n-2}{2}}$ vine structures for n variables, as proved in Nápoles (2010), hence it is not possible to estimate all vine structures and choose the best one in high dimensions. A heuristic called the Dißmann's heuristic introduced in Dißmann et al. (2013) is often used. It constructs the vine structure tree-wise, starting from the first tree, along with the copula family selection and copula parameter estimation. Each tree structure is determined by a maximum spanning tree with weights being the absolute Kendall's tau.

To illustrate the importance of the choice of vine structure, a 4 dimensional example is presented below. In Table 1.1 information about the vine structure is listed (see conditioning and conditioned sets (column 1), copula families (column 2) and the corresponding Kendall's tau used to compute the parameter of the copula (column 3)). A sample of 1000 points is simulated from this model. This data set is used to estimate all possible vine structures (there are 24 structures) in 4 dimensions.

True vine model			Dißmann's heuristic		
Structure	Copula family	τ	Structure	Copula family	τ
Tree-3			Tree-3		
32|41	J270	-0.61	42|13	t	0.06
Tree-2			Tree-2		
13|4	G	0.32	12|3	t	0.12
42|1	J	0.45	41|3	t	-0.07
Tree-1			Tree-1		
41	C90	-0.54	32	F	-0.59
43	C270	-0.73	31	BB8_180	0.55
12	J90	-0.36	43	C270	-0.72

Table 1.1: The specified vine copula model (True vine model) in dimension 4 with its vine structure, the copula families and the Kendall's tau of the copulas; The estimated vine copula model by Dißmann's heuristic (Dißmann's heuristic) with its vine structure, the estimated copula families and the estimated Kendall's tau of the copulas.

In Figure 1.3 a box plot of AIC values for each estimated vine structure (out of 24 vine structures in dimension 4) is shown. Obviously the true vine model has the smallest

AIC. The *AIC* of the vine model obtained using the Dißmann's heuristic is shown with the dotted horizontal line. One can compare that its structure is quite different from the true model (see columns 4,5,6 in Table 1.1 to examine the vine structure, the chosen copula families and the Kendall's tau corresponding to their estimated parameters). The performance of the vine chosen by the Dißmann's heuristic is worse than the performance of 16 other vine structures (significantly worse than 12 of them according to a Vuong test with 5% significance level). The vine structure used to simulate the data is a simplified vine, however, when a different structure is taken, namely one obtained with the Dißmann's heuristic, the simplified assumption is rejected at 5% significance level in tree T_3.

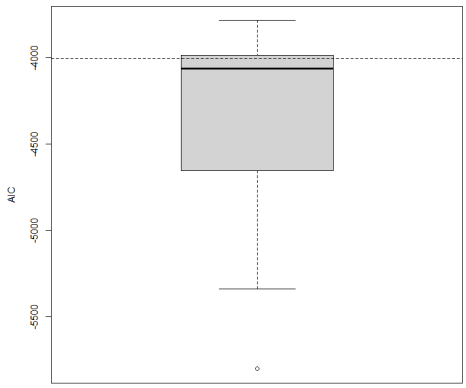

Figure 1.3: Box plot of *AIC* for all the vine structures (24 different vine structures) in dimension 4 for the simulated data. The *AIC* represented by horizontal line is the AIC of the vine copula model estimated by the Dißmann's heuristic.

As can be already observed in the 4 dimensional example the Dißmann's heuristic might not perform well, and considering that the number of vine structures grows exponentially with dimension, there is a need for finding better methods to choose a vine structure in vine copula models.

1.4. OUTLINE OF THE THESIS

The main focus of this thesis is to propose new methods of vine structure selection. The thesis is organized as follows,

- In Chapter 2, an introduction to vine structure representation is given. A few ways of representing regular vines graphically, from the literature, are included. They all have their merits. Figure 1.2 is the most commonly used representation by listing each tree structure of a vine. In order to have a clearer visualization of the parent-child relationship between nodes of a vine, a vine triangular array is proposed. In order to have a

more compact representation that can ease programming, a vine matrix representation can be used. A new vine representation, called the vine binomial tree, is proposed. Vine binomial tree, in contrast to the vine triangular array, is not very compact but allows the parent-child relationships to be easily manipulated, which is useful when one wants to construct very different vine structures; in contrast to the vine matrix, vine structure constructed through vine binomial tree does not rely on the specified order, which allows more flexible ways of vine structure construction.

- Chapter 3 is based on the paper, Zhu et al. (2020). A new method is proposed to search for a vine structure with better performance for data than an initial structure. The initial structure can be any structure from existing vine structure constructions, e.g., the Dißmann's heuristic. The proposed search method will examine a set of structures that are 'significantly' different from the initial one. It has been shown that one can take the number of sampling orders these structures have in common as a proxy indicating the differences between them. This is because vines that have more common sampling orders will have more common bivariate (conditional) copulas, which will lead to similar performance. Several algorithms are proposed in Chapter 3 to construct vine structures with any given number of common sampling orders. Finally, it is suggested to search for vines having 2 sampling orders in common with the initial structure because, 1) it is known how many vines having 2 common sampling orders with an initial vine are, and this number is not very large as compared with the number of all vine structures; 2) it is also relatively easy to construct vines having 2 common sampling orders with an initial structure.

- Chapter 4 is based on the paper, Zhu et al. (n.d.). The method designed in Chapter 3 to find an 'optimal' vine copula model is applied for acute ischemic stroke patients data set including mixed continuous and discrete variables. This model is built to generate synthetic patients data. It has been shown that the obtained vine copula model for the patients data set has a better performance as compared with other method that has been proposed for this purpose in the literature: the fully conditional specification (by specifying a sequence of conditional distributions).

- Chapter 5 is based on the paper, Zhu et al. (2021). In this chapter R-vine based forward selection regression method is discussed. We showed that the vine structure in the vine based regression does not need to be limited to any particular vine structure. An algorithm has been proposed to consider all possible vine structures in the process. When a vine structure is constructed in an appropriate way, the conditional distribution of the response variable(s) given the covariates can be estimated in analytic form.

- Chapter 6 is based on the paper, Zhu & Kurowicka (2022). The main idea in this chapter is to consider vine copula models with certain pattern of conditional independence. This is achieved by working with incomplete vines, called m-saturated vines, where certain bivariate copulas are fixed to be the independence copula. This assumption reduces the complexity of vine copula model. The main result of this chapter is the proof of an equivalence between m-saturated vines and strongly chordal graphs (which are undirected graphs with special properties). Due to the specified conditional independence, it becomes possible to work with subvines with smaller number of variables.

This allows to examine all possible structures for the subvines and/or estimate non-simplified vines.

1.A. APPENDIX TO CHAPTER 1

Vine copula model can also be applied in the case when both continuous and discrete variables exist. Detailed discussion can be found in Panagiotelis et al. (2012), Joe (2014) and we follow the notation in Joe (2014) and assuming a simplified vine.

Theorem 1.2.1 still holds for mixed variables, but in order to take into account discrete variables, the bivariate copula density $c_{jk;D_e}(C_{j|D_e}, C_{k|D_e})$ is now generalized to $\tilde{c}_{jk;D_e}(C_{j|D_e}, C_{k|D_e})$. Denote $C^+_{j|D_e} = C_{j|D_e}(u_j|\boldsymbol{u}_{D_e})$ and $C^-_{j|D_e} = C_{j|D_e}(u_j^-|\boldsymbol{u}_{D_e})$, where u_j^- is the left-sided limit to u_j hence for continuous variable it's u_j and for integer valued discrete variable it is $u_j - 1$. Conditional density for continuous variable j and conditional probability mass function for discrete variable j given variables in D_e are both denoted as $f_{j|D_e}$. Then the copula density can be represented as,

- When both variable j and k are continuous, $\tilde{c}_{jk;D_e}(C_{j|D_e}, C_{k|D_e}) = c_{jk;D_e}(C_{j|D_e}, C_{k|D_e})$.
- When only variable j is discrete, $\tilde{c}_{jk;D_e}(C_{j|D_e}, C_{k|D_e}) = \frac{C_{j|k;D_e}(C^+_{j|D_e}|C_{k|D_e}) - C_{j|k;D_e}(C^-_{j|D_e}|C_{k|D_e})}{f_{j|D_e}}$.
- When only variable k is discrete, $\tilde{c}_{jk;D_e}(C_{j|D_e}, C_{k|D_e}) = \frac{C_{k|j;D_e}(C^+_{k|D_e}|C_{j|D_e}) - C_{k|j;D_e}(C^-_{k|D_e}|C_{j|D_e})}{f_{k|D_e}}$.
- When both variable j and k are discrete,
  $\tilde{c}_{jk;D_e}(C_{j|D_e}, C_{k|D_e}) = \frac{C_{jk;D_e}(C^+_{j|D_e}, C^+_{k|D_e}) - C_{jk;D_e}(C^-_{j|D_e}, C^+_{k|D_e}) - C_{jk;D_e}(C^+_{j|D_e}, C^-_{k|D_e}) + C_{jk;D_e}(C^-_{j|D_e}, C^-_{k|D_e})}{f_{j|D_e} f_{k|D_e}}$.

where the condition distribution $C_{j|k;D_e}$ or $C_{k|j;D_e}$ is differentiated by the corresponding variable in $C_{jk;D_e}$ and the argument $C_{j|D_e}$ or $C_{k|D_e}$ is computed recursively from copulas in lower trees. Both of them can be represented similarly as follows,

$$C_{j|D_e\cup\{k\}} = \begin{cases} C_{j|k;D_e}(C_{j|D_e}|C_{k|D_e}), & X_k \text{ continuous}, \\ \frac{C_{jk;D_e}(C^+_{k|D_e}, C_{j|D_e}) - C_{jk;D_e}(C^-_{k|D_e}, C_{j|D_e})}{F^+_{k|D_e} - F^-_{k|D_e}}, & X_k \text{ discrete}. \end{cases}$$

2

VINE REPRESENTATIONS AND STRUCTURE CONSTRUCTIONS

As explained in Chapter 1, different vine structures might have better or worse performance on given data set. In this chapter a few ways of representing vine structures are shown. Each one of them has its advantages and disadvantages. We present in detail the vine binomial tree representation, which has been shown very useful in the vine structure construction.

Parts of this chapter have been published in Zhu et al. (2020, 2021).

2.1. INTRODUCTION

There are $\binom{n}{2}(n-2)!2^{\binom{n-2}{2}}$ vine structures on n elements (Nápoles (2010)). Theoretically, each one of these structures can be used to represent a density. However, when vine copulas are estimated from data, due to the tree-wise estimation, the limited choice of copula families and the simplifying assumption, some vine structures might have better performance than others. Different heuristic methods to choose the 'best' structure have been introduced in the literature. The most popular one is the Dißmann's heuristic, presented in Dißmann et al. (2013). The idea of this method is to start building a vine structure from the first tree that is chosen to maximize the dependence. Copulas in the first tree are estimated and data is transformed using these estimated copulas, so the second tree structure can be constructed similarly, and etc.. Originally, Kendall's tau measure of dependence was used as weights in Dißmann's heuristic. Other choices have been also considered, for example, p-value of goodness-of-fit tests for the bivariate copulas (Czado et al. (2013)), or combination of Kendall's tau and p-value of test of simplifying assumption (Kraus & Czado (2017*b*)). Simulation studies in these papers show that these methods often lead to improved vine models. In Kurowicka (2011), the author proposes to construct a vine structure starting from the top node of the vine, which is chosen as the one with the smallest absolute partial correlation estimated from data. The idea is quite similar to the Dißmann's heuristic, namely one wants to capture the least dependence in higher level trees, hence most dependence in lower levels. This heuristic, however, has not been studied so extensively. In Brechmann & Joe (2014), the authors propose to search for tree structures based on so called *1-neighbor* trees. Another approach is to examine random trees obtained through a Monte Carlo simulation Chang et al. (2019). This method is computationally expensive and can only be applied in lower dimensions. Bayesian approaches to vine structure selection are proposed in Gruber & Czado (2015, 2018).

Instead of considering each tree separately in a vine, Cooke et al. (2015) proposes to search for vine structures having 0 common sampling orders with an initial estimated vine structure. This is a very different idea as compared to the previous attempts. The proposal in Cooke et al. (2015) is based on the intuition that the number of common sampling orders might be used as a measure of similarity of vine structures. This intuition was not supported by extensive investigation. An extension of this idea is worked out in Zhu et al. (2020), where the authors show that the number of common sampling orders is a good proxy to measure similarity of vine structures. However, they decided to search for vines having 2 common sampling orders with the initial structure. This is the topic discussed in Chapter 3.

This chapter gives an overview of different representations of vine structures, which will be used throughout this thesis. Then we illustrate how a vine structure can be constructed using each of these representations.

2.2. VINE REPRESENTATIONS

In this section four different vine representations are discussed. These include tree by tree representation which we have already used in Chapter 1, vine triangular array which first appeared in Cooke et al. (2015), vine matrix representation introduced in

Nápoles (2010) and finally, vine binomial tree (VBT) introduced in (Zhu et al. (2020)) is explained. Compared with the first three existing representations, VBT provides a more flexible way of vine structure construction.

2.2.1. TREE-WISE REPRESENTATION AND VINE TRIANGULAR ARRAY

A vine is a set of trees hence the most intuitive representation for a vine structure is the tree by tree representation shown in Figure 2.1 for two vines on 5 elements.

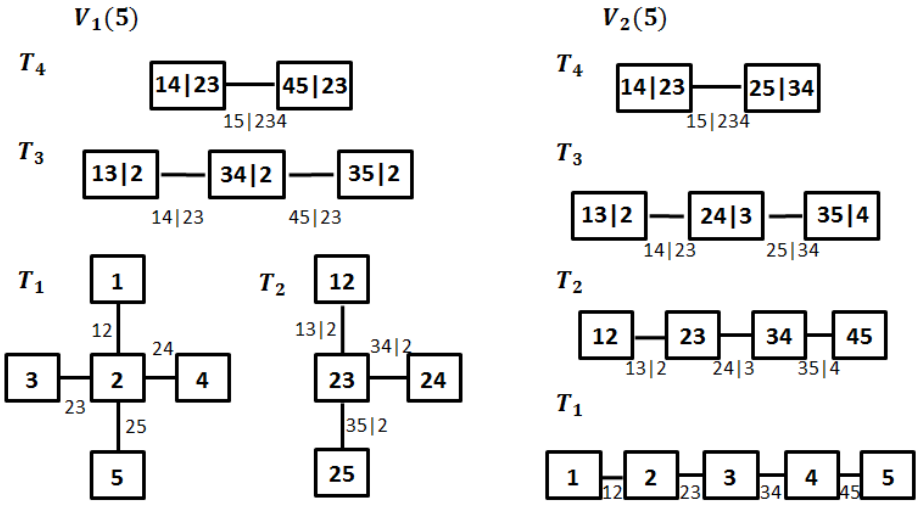

Figure 2.1: Tree by tree representation of two vines $V_1(5)$ (left) and $V_2(5)$ (right).

These vines are two special types of vine structures. $V_1(5)$ is called a **C-vine**. This vine has a root node (one node connected to all remaining nodes in this tree) in every tree. In Figure 2.1 (left), node 2 is the root node in the first tree, node 23 is the root node in the second tree and node 34|2 is the root node in the third tree. In the fourth tree there are only two nodes 14|23, 45|23 so each can be chosen as a root node; $V_2(5)$ is called a **D-vine**. For this type of vine, all nodes in the first tree have degree at most two (where degree of a node is the number of edges connected to this node). Its structure is thus determined by the order of nodes in the first tree, as due to proximity, there are no more choices for nodes in higher trees.

In order to emphasize the parent-child relationship of nodes in a vine structure, another representation called **vine triangular array** was introduced in Cooke et al. (2015). Figure 2.2 shows the vine triangular array representations for the two vines $V_1(5)$ and $V_2(5)$ in Figure 2.1.

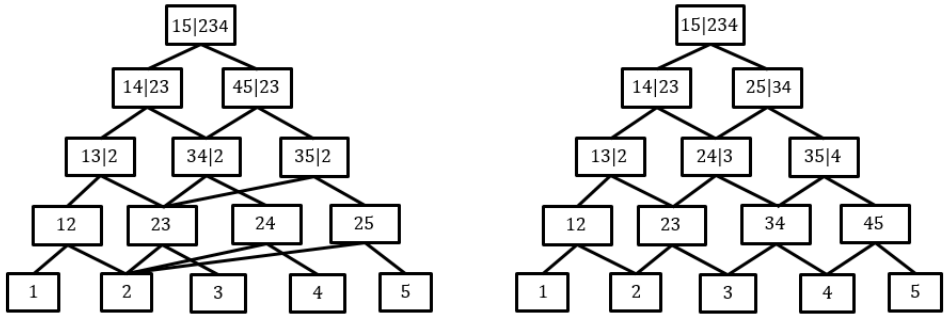

Figure 2.2: Triangular array representation for $V_1(5)$ (left) and $V_2(5)$ (right) in Figure 2.1

From properties of regular vines we know that each node in tree T_i has two children in tree T_{i-1}. Note that the conditioned and conditioning sets of a given node are already sufficient to determine which two nodes have been combined in the lower tree. However, in the vine triangular array, we draw lines joining children and parents to make them immediately visible. Then, starting from the top node we can follow one of two lines leading to its children in tree T_{n-1} and again we choose one of two lines until we reach a node in T_1. Then we can refer to a **path** [1] in the vine triangular array.

Both tree by tree and triangular array are not so easy and compact to be used in computer implementations. In the next two sections vine matrix and vine binomial tree representations are introduced.

2.2.2. VINE MATRIX

Vine matrix representation introduced in Nápoles (2010) is very popular (note that in Dißmann et al. (2013) slightly adjusted vine matrix approach was used) and compact way to store information needed to represent a vine structure. The structure of a vine on n elements is stored in a lower triangular n by n matrix. Additional matrices are be used to store information about copula family and copula parameters. This is implemented in the `VineCopula` package (using lower triangular matrix) and in the `rvinecopulib` package (using upper triangular matrix). In our presentation we follow the result in Nápoles (2010), hence the vine matrix is shown in the form of a lower triangular matrix.

The matrix representation requires the specification of a natural order of variables. A **natural order** NO is a permutation of elements $\{1,\dots,n\}$, which is determined as follows: we start with either element in the conditioned set of the top node, and the remaining elements $j, j = 2,\dots,n-1$ in NO are then determined such that $j-1, j$ is a conditioned set of a node in tree T_{n-j+2} (Joe et al. (2011)). Then the matrix representation of a vine is defined as follows:

Definition 2.2.1 (regular vine matrix)**.** *A regular vine matrix $M(V(n)) = \{m_{i,j}\}$ for $i, j = 1,\dots,n$ and $i \le j$ is a lower triangular matrix with $m_{i,i}$ equals the element in position i in NO and $m_{i+1,i}$ equals to the element in position $i+1$ in NO. Furthermore, it represents a*

[1] We do not allow a path in the vine triangular array to go up and down. Hence, in this sense the defined path is directional.

regular vine $V(n)$ if and only if for all $i \geq n-1$, element $m_{i,j} = m_{h,h}$ or $m_{i,j} = m_{i+1,h}$ for some h such that $j < h \leq i$ and $\{m_{j,j}, \ldots, m_{i-1,j}\} \cap \{m_{i+1,h}, \ldots, m_{n,h}\} = \emptyset$.

There are always two natural orders for a regular vine. For the regular vine $V_1(5)$ in Figure 2.2 (left), the natural order $NO = (1,5,4,3,2)$ leads to the matrix $M_1(V_1(5))$ and the vine matrix $M_2(V_1(5))$ is obtained when $NO = (5,1,4,3,2)$ is used.

$$M_1(V_1(5)) = \begin{pmatrix} 1 & & & & \\ 5 & 5 & & & \\ 4 & 4 & 4 & & \\ 3 & 3 & 3 & 3 & \\ 2 & 2 & 2 & 2 & 2 \end{pmatrix} \qquad M_2(V_1(5)) = \begin{pmatrix} 5 & & & & \\ 1 & 1 & & & \\ 4 & 4 & 4 & & \\ 3 & 3 & 3 & 3 & \\ 2 & 2 & 2 & 2 & 2 \end{pmatrix}$$

Observing that a natural order forms elements on the diagonal of each matrix, and conditioned and conditioning sets of nodes in a vine are represented in the vine matrix by $m_{j,j}, m_{k,j} | m_{k+1,j}, \ldots, m_{n,j}$, $j < k \leq n$. For instance, when we look at $M_1(V_1(5))$, we can see nodes that have element 1 in the conditioned set are coded in the first column. First, element 1 and 5 form a conditioned set of a node whose conditioning set is composed of elements lying in the first column underneath 5, hence is {4,3,2}. Then, element 1 is in the conditioned set of a node together with element 4, and the conditioning set lies underneath 4 which is {3,2}, and etc..

In Section 2.3.1 we show how a vine structure $V(n)$ can be constructed for a given NO by using a binary sequence of length $\binom{n-2}{2}$.

2.2.3. VINE BINOMIAL TREE

We will now present a **vine binomial tree** (VBT) representation of regular vines, which has been introduced in Zhu et al. (2020). Although VBT is not, strictly speaking, a binomial tree since it does not have a root and in the last split each node has only one child, most of the concepts used in binomial trees, such as sub-trees, depth, path (root-to-leaf path) are inherited by VBT. Depth-i, for $i = 1, \ldots, n$ represents the level of the binomial split. It is in principle not necessary to list the last split in depth-n. For completeness, however, we show it in Figure 2.3. Each node of the VBT is referred to as an **element** in VBT and is denoted by VBT$[i][j]$, where $i < n$ is the depth of this element and $j = 1, \ldots, 2^i$ represents its position in depth-i, from left to right.

VBT is just another representation of information contained in the vine triangular array. A pair of elements (VBT$[i][2k-1]$, VBT$[i][2k]$), $k = 1, ..., 2^{i-1}$ represents the conditioned set of a node in tree T_{n-i+1} in the vine triangular array. We call VBT$[i][2k]$ a **partner element** of VBT$[i][2k-1]$. Furthermore, the conditioning set of the node in the vine triangular array contains elements that have not yet appeared on the paths in VBT through this pair of elements up to and including depth-i. For example in Figure 2.3, the conditioned set of node 43|2 in $V_1(5)$ appears in (VBT[3][1], VBT[3][2]), where 4 and 3 are partner elements and element 2 that has not appeared on the paths up to and including this pair of elements is the conditioning set. Note that the same sub-trees appear underneath the elements VBT[2][1] and VBT[2][3], which is because 34|2 is the common child of 54|23 and of 14|23. The **corresponding elements** in VBT of the element in the conditioned set of a node in the vine triangular array can be found in the pair of elements in

VBT representing this node, e.g. corresponding elements of 4 in node 43|2 are VBT[3][1] and VBT[3][5]. The part of VBT in the shaded area denotes a common subvine, where the top node of this subvine is represented by the same pair of elements (the order of the elements can be different).

The proximity condition is also encoded in the elements of VBT. If we consider a pair of elements in depth-i, this node needs to have two children, which are nodes represented by the pair of elements in depth-$(i+1)$. For example in Figure 2.3, the pair of elements (VBT[2][1],VBT[2][2])=(5,4) representing node 54|23 whose two children are (VBT[3][1], VBT[3][2])=(4,3) and (VBT[3][3], VBT[3][4])=(5,3) representing nodes 43|2 and 53|2, respectively. Moreover, since the initial segments of paths through VBT[1][1], VBT[2][1] and VBT[1][2], VBT[2][3] contain the same elements, the sub-trees underneath VBT[2][1] and VBT[2][3] are the same (see the shaded elements in Figure 2.3).

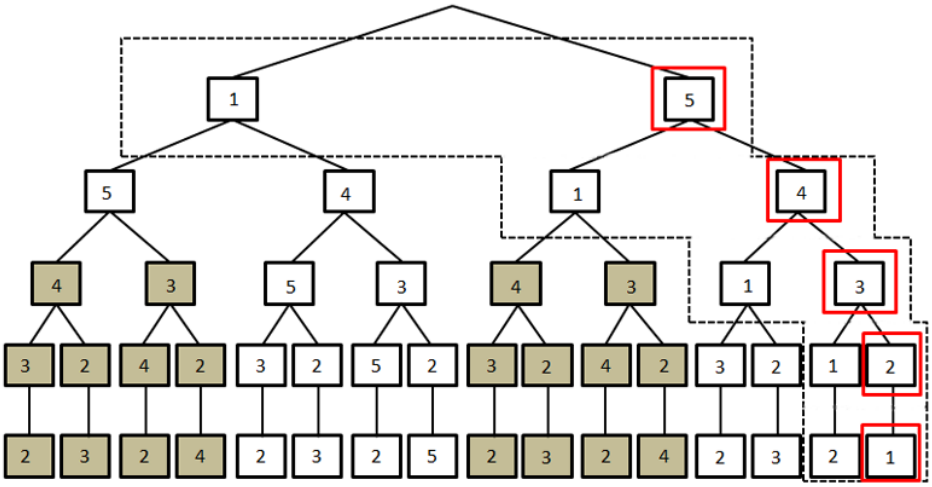

Figure 2.3: VBT representation of $V_1(5)$. The dashed area denotes a substructure and the order is represented in red squares. The shaded area denotes a common subvine in dimension three.

One path in the vine triangular array gives us a **substructure** in corresponding VBT. The root-to-leaf path in VBT is given and partner elements to elements of this path are easily found. Moreover, we define the **ordering** $(l_1,\dots,l_n)$ of elements in substructure by taking l_i to be an element in the pair of elements in the substructure in depth-$(n-i+1)$, but not in depth-$(n-i+2)$ for $i>2$ and l_2, l_1 are the most left elements in the pairs of elements in depth-$(n-1)$ and depth-n, respectively. For example, a path through nodes 15|234, 14|23, 13|2 and 12 in triangular array of $V_1(5)$ corresponds to the substructure in dashed area in Figure 2.3 and the ordering $(2,1,3,4,5)$ is obtained. This ordering is just a sampling order of regular vine which will be introduced in Chapter 3. In Chapter 5 a different order is used, referred to as **order of elements in VBT**. Elements are ordered by choosing always the right elements in the pair of elements (from the last depth to depth-1) in this substructure, hence order $(1,2,3,4,5)$ is obtained. Since we can always exchange the first two elements in the order, these two approaches are the same.

The reverse NO in the vine matrix representation is a special order in VBT. One NO of the vine matrix is determined in the substructure constituting pair of elements (VBT[1][1], VBT[1][2) in depth-1, (VBT[2][1], VBT[2][2]) in depth-2, ..., (VBT[i][1], VBT[i][2]) in depth-i, ... (VBT[n][1], VBT[n][2]) in depth-n; the other NO is then determined in the substructure of pair of elements (VBT[1][1], VBT[1][2) in depth-1, (VBT[2][3], VBT[2][4]) in depth-2, ..., (VBT[i][$2^{i-1}+1$], VBT[i][$2^{i-1}+2$]) in depth-i, ..., (VBT[$n-1$][$2^{n-2}+1$], VBT[$n-1$][$2^{n-2}+2$]) in depth-$n-1$ and (VBT[n][$2^{n-2}+1$], VBT[n][$2^{n-2}+2$]) in depth-n.

To represent a regular vine, a VBT needs to satisfy conditions presented in Proposition 2.2.1.

Proposition 2.2.1. *VBT represents a regular vine if and only if it satisfies conditions (1,2,3,4).*

1. *Each path in VBT is a unique permutation of* $\{1,...,n\}$.
2. *All partner elements in VBT are distinct. That is,* VBT[i][$2k-1$] $\neq$ VBT[i][$2k$], *for* $i = 1,\ldots,n-1$, $k = 1,\ldots,2^{i-1}$.
3. *For* $i = 1,\ldots,n-2$ *and* $j = 1,\ldots,2^i$ *if* j *is odd then* VBT[i][j] *is either* VBT[$i+1$][$2j+1$] *or* VBT[$i+1$][$2j+2$], *and if* j *is even* VBT[i][j] *is either* VBT[$i+1$][$2j-3$] *or* VBT[$i+1$][$2j-2$].
4. *If the initial segments of* i ($i \geq 2$) *elements of two paths contain the same elements (irrespective the order) and* VBT[i][j] *and* VBT[i][k] *are the* i*-th element in each path respectively,* $1 \leq j,k \leq 2^i$, *then the sub-trees underneath root* VBT[i][j] *and underneath root* VBT[i][k] *are the same.*

Proof. By construction, a VBT representing a regular vine satisfies conditions (1,2,3,4). Conversely, by property (2), elements in (VBT[i][$2k-1$], VBT[i][$2k$]), for each $i = 1,\ldots,n$, with $k = 1,...,2^{i-1}$ are distinct and they constitute the conditioned set of one node in echelon-$(n-i+1)$ in the vine triangular array. Since the VBT inherits properties of a binomial tree, then the initial segments up to depth-i of paths through (VBT[i][$2k-1$], VBT[i][$2k$]) are the same. Hence, by properties (1) and (3), the common elements of paths in subtrees underneath roots VBT[i][$2k-1$] and VBT[i][$2k$] form the conditioning set of the node in echelon-$(n-i+1)$ of vine triangular array.

To conclude the proof, we show that, under conditions (1,2,3,4), the proximity condition is preserved. Consider the nearby pairs of elements (VBT[i][$4l-3$], VBT[i][$4l-2$]) and (VBT[i][$4l-1$], VBT[i][$4l$]), with $i \geq 2$ and $l = 1,\ldots,2^{i-2}$. By conditions (1,2,3), the symmetric difference of these two pairs contains 2 elements, that is VBT[$i-1$][$2l-1$] and VBT[$i-1$][$2l$]. Thus they are children of node in echelon-$(n-i+2)$ with conditioned set (VBT[$i-1$][$2l-1$], VBT[$i-1$][$2l$]) (they are siblings). Because condition (3), VBT[$i-1$][$2l-1$] will be either VBT[i][$4l-1$] or VBT[i][$4l$] and VBT[$i-1$][$2l$] will be either VBT[i][$4l-3$] or VBT[i][$4l-2$]. Hence by condition (4) these two nodes have a common child. □

It is possible to formalize a standard form of a VBT as follows,

Definition 2.2.2 (Standard form of VBT)**.** *VBT is in its* ***standard form*** *if* VBT[$i+1$][$2j+1$] *is equal to* VBT[i][j], *when* j *is odd, and* VBT[$i+1$][$2j-3$] *is equal to* VBT[i][j] *otherwise, for* $i = 1,\ldots,n-2$ *and* $j = 1,\ldots,2^i$.

The VBT in Figure 2.3 is a standard VBT. Note that there are two standard VBT representations for one regular vine structure, which depend on the order of elements in depth-1.

In the next section we show how VBT is used to construct a vine structure for a given order.

2.3. VINE STRUCTURE CONSTRUCTION

There are different ways of constructing a vine structure. According to the definition of regular vine, an intuitive way is to construct the vine structure tree-wise as proposed in Dißmann et al. (2013) following the so called bottom-up approach, or using top-down approach advised in Kurowicka (2011). Rather than applying tree-wise construction, in this section we show that a vine structure can be determined by an order (a permutation of elements $\{1,\dots,n\}$) and a sequence of binary number of length $\binom{n-2}{2}$.

It has been discussed in Brechmann & Schepsmeier (2013) that a C-vine can be determined by an order $(l_0, l_1, \cdots, l_k)$, where l_1 is the root node in the first tree, node $l_1 l_2$ is the root node in the second tree, node $l_{i-1} l_i | l_1 \dots l_{i-2}$ is the root node in tree-i for $i > 3$ and finally we have an edge $l_0 l_k | l_1 \dots l_{k-1}$ in the last tree. This leads to order of the C-vine $V_1(5)$ in Figure 2.1 (left) equals to $(1,2,3,4,5)$. For a D-vine, when the first tree is $l_0 - l_1 - l_2 - \cdots - l_k$, then the order can be taken as $(l_0, l_1, \dots, l_k)$ (or reverse order), which for the vine in Figure 2.1 (right) is $(1,2,3,4,5)$. These types of vines correspond to a particular choice of the binary sequence, which will be discussed later on in this section.

2.3.1. THROUGH VINE MATRIX

In Joe et al. (2011), it is explained how a regular vine can be constructed in the form of a vine matrix for a given natural order $NO = \{l_1, \dots, l_n\}$ and a binary sequence (which is placed in a lower triangular matrix $B = \{b_{i,j}\}, i, j = 1, \dots, n$ and $i \leq j$. Matrix B satisfies additionally that elements $b_{n,j} = 1$, $b_{i,i} = 1$ and $b_{i+1,i} = 1$. In total there are $\binom{n-2}{2}$ elements in matrix B that can be specified. Algorithm 1 describes this procedure.

When all elements that can be chosen in matrix B are set to be equal to 0, the constructed vine is a D-vine. In contrast if all elements in B are 1 then the constructed vine is a C-vine. In this sense D-vine and C-vine are the most 'extreme' structures of vines. Other combinations of numbers in B lead to different types of vines, which has been discussed by the notion of vine equivalence class (Joe et al. (2011)).

The algorithm to construct a vine structure can also be presented as a combination of extensions of $V(j)$ by extra element j; as shown in Nápoles (2010) there are always 2^{j-2} possible extensions.

2.3.2. THROUGH VINE BINOMIAL TREE

We show in this section how a vine structure can be constructed based on VBT. This lays the foundation of methods introduced in Chapter 3 and Chapter 5. An example is given at first. Suppose the given order of elements is $(l_0, l_1, l_2, l_3, l_4)$.

The construction is done by the successive extension by next element in the given order. The extension process starts with extending the vine with order (l_0, l_1, l_2) by l_3. This is due to the fact that the vine structure in dimension 3 is fully determined by the

Algorithm 1 Generation algorithm of regular vine structure on vine matrix.

Input: A natural order NO and lower triangular matrix B.
Output: A regular vine matrix M.

Initialize a lower triangular matrix $M = \{m_{i,j}\}$ for $i, j = 1, \ldots, n$ and $i \leq j$.
Assign elements $m_{i,i}$ with l_i in NO and $m_{i+1,i}$ with l_{i+1}.
Assign $m_{n,n-2}$ with l_n.
for $j = n-3$ to 1 **do**
    Initialize ac (active column) being $j+2$.
    for $i = j+2$ to n **do**
        if $b_{i,j} = 1$ **then**
            Assign $m_{i,j}$ with $m_{ac,ac}$.
            ac is updated by $k(j < k \leq i)$ where l_k is the first element in NO which doesn't appear in column j.
        else
            Assign $m_{i,j}$ with $m_{i+1,ac}$.
        end if
    end for
end for

given order. Two possible extensions for the vine structure with given order (l_0, l_1, l_2, l_3) are shown in Figure 2.4. They depend on the choice of element VBT[2][2] (indicated by the black circle). According to Proposition 2.2.1 parts 3) and 4), this element can be taken as either VBT[3][5] or VBT[3][6] (marked by red circle in both figures). Once its value is determined, the remaining elements in VBT (red elements) can be filled by following the properties of VBT. Since there are two choices available for VBT[2][2], we assign to it an **indicator**, where 1 denotes that its value is given by the left element VBT[3][5] and 0 denotes the right element VBT[3][6] in this pair of elements. By the properties of standard VBT, if indicator is 1 then the root in the first tree (element l_1) will remain unchanged and this leads to the C-vine in Figure 2.4 (left). By contrast, element l_1 loses the status of the root node in tree 1 when indicator is 0 and the first tree is now $l_0 - l_1 - l_2 - l_3$. Hence a D-vine is obtained, as shown in Figure 2.4 (right).

Once a choice of indicator is made for the vine structure with order (l_0, l_1, l_2, l_3), we can extend this vine by l_4, as depicted in Figure 2.5. The subvine on elements (l_0, l_1, l_2, l_3) gives the right hand side of the VBT and the elements in the left side need to be filled in. Two choices are available for VBT[2][2] (see the black circle), either element VBT[3][5] or VBT[3][6]. Next, the choice for element VBT[3][4] (shown in black circle) depends on the choice for VBT[2][2]. If the indicator for VBT[2][2] is 1 then the value of VBT[3][4] can be taken to be equal to either VBT[4][9] or VBT[4][10], otherwise its value is either VBT[4][11] or VBT[4][12]. Note that in every step this process finds a partner element of the newly added element, whose value is determined by an indicator. Hence the final vine structure for a given order can be determined by a sequence of indicators.

The detailed algorithm is given as follows by specifying the elements that need to be filled in as **0 elements** (to be consistent with the notation in Chapter 3) during extension, and the extension procedure is implemented from depth-$(n-2)$ to depth-2.

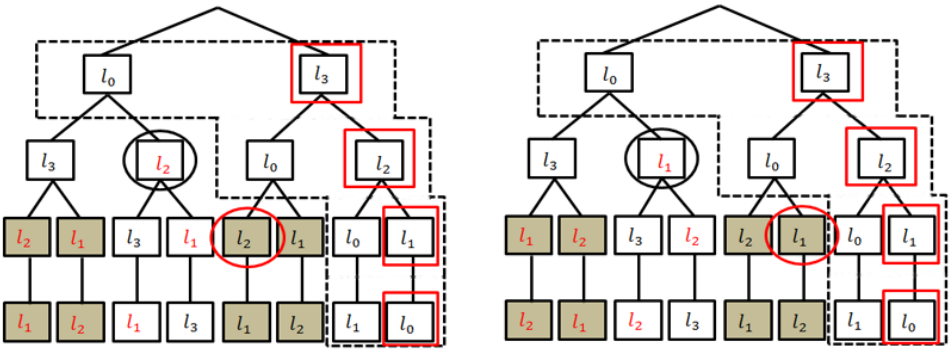

Figure 2.4: Standard form of VBTs and possible extensions where two choices of VBT[2][2] (in black circle) lead to the C-vine (left) and the D-vine (right) with order (l_0, l_1, l_2, l_3). (Elements in black are already known or can be filled in using definition of the standard form of a VBT, whereas elements in red depend on previous choices.) The shaded area denotes a common subvine.

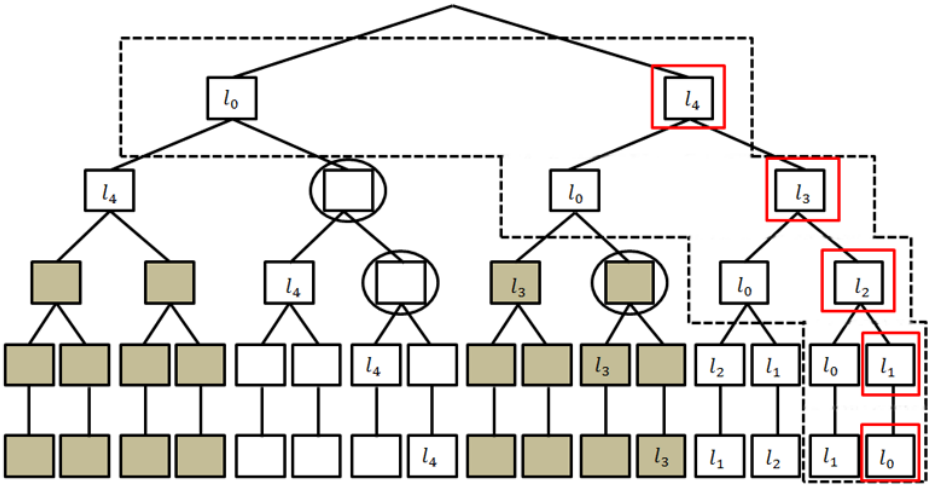

Figure 2.5: VBT with given order $(l_0, l_1, l_2, l_3, l_4)$ in the extension process where the sub-tree underneath VBT[1][2] is already filled. The elements where choices are possible are indicated by a black circle. The shaded area denotes a common subvine.

Algorithm 2 Generation algorithm of regular vine structure on VBT.

Input: A given order and a sequence of binary indicators.
Output: A fully specified VBT representing a regular vine.

1: Specify a substructure in VBT for the given order according to the definition of the ordering in VBT.
2: Fill in 0 elements in VBT that can be determined by the definition of a standard VBT.
3: **for** $i = n-2$ to 2 **do**
4: Find the 0 element in depth-i.
5: Find the pair of elements in depth-$(i+1)$ according to Proposition 2.2.1 parts 3) and 4).
6: **if** Indicator 1 is given **then**
7: The 0 element in depth-i is filled by the left element in the found pair of elements.
8: **else**
9: The 0 element in depth-i is filled by the right element in the found pair of elements.
10: Exchanging the orders of the pairs of elements in the sub-tree underneath the partner element of the 0 element in depth-i, such that the VBT is always in its standard form.
11: **end if**
12: Repeat the above procedures (Steps 4-11) for the 0 elements in depth-j for $j = i+1$ to $n-2$.
13: **end for**
14: Fill in 0 elements in VBT that can be determined by the definition of a standard VBT.

Following the example above and the Algorithm 2, we notice that if in every step of the extension process indicators are chosen to be 1, the partner element of the newly added element l_j $(j = 3,4)$ in depth-i $(2 \le i \le j)$ is equal to l_{j-i+1}. By choosing always elements corresponding to 1, a C-vine is obtained. In the case when each time 0 is picked, the partner of the newly added element in depth-i is l_{i-1} and we get a D-vine structure. This relationship holds true for a general order $(l_0, l_1, \cdots, l_k)$ which is presented in Proposition 2.3.2.

We start with the proof of one step of the extension process.

Proposition 2.3.1. *During extension when a new element is added, the partner element of the newly added element in depth-i $(2 \le i)$, VBT$[i][2^{i-1}]$, is VBT$[i+1][2^i+1]$ if all indicators in the extension are chosen to be 1, or is VBT$[i+1][3\cdot 2^{i-1}]$ if indicators are all 0.*

Proof. The proof is by induction on the depth in VBT. In depth-2, VBT[2][2] is the first partner element where choice can be made when a new element is added. Due to the definition of the standard form of a VBT, the initial segment of the path through VBT[1][1] and VBT[2][1] contains the same elements as the one through VBT[1][2] and VBT[2][3]. Then by Proposition 2.2.1 parts 3) and 4), the element VBT[2][2] can be chosen to be one element in the pair of elements (VBT[3][5], VBT[3][6]).

If indicator 1 is chosen, then VBT[2][2] = VBT[3][5]. Assume that our claim holds for any partner element up to depth-j, hence that the partner element VBT$[l][2^{l-1}]$ = VBT$[l+1][2^l+1]$ $(l \le j)$ when all indicators are 1. In depth-$(j+1)$, the partner element is VBT$[j+1][2^j]$. By the definition of the standard form of a VBT, the initial segments of the paths through VBT[1][1], VBT[2][2], VBT[3][4], ..., VBT$[j][2^{j-1}]$, VBT$[j+1][2^j-1]$ and VBT[1][2], VBT[2][3], VBT[3][5], ..., VBT$[j+1][2^j+1]$ contain the same elements. Thus according to Proposition 2.2.1 part 3) and 4), the value of VBT$[j+1][2^j]$ can be one of the pair of elements (VBT$[j+2][2^{j+1}+1]$, VBT$[j+2][2^{j+1}+2]$). If indicator is 1, then this partner is VBT$[j+2][2^{j+1}+1]$.

Similarly if indicator is chosen as 0, VBT[2][2] = VBT[3][6]. Assume the proposition holds for any partner element up to depth-j, which means the partner element VBT$[l][2^{l-1}]$ in depth-l $(l \le j)$ is VBT$[l+1][3\cdot 2^{l-1}]$ when all indicators are 0. In depth-$(j+1)$, the value of the partner element VBT$[j+1][2^j]$ can be chosen from the pair of elements (VBT$[j+2][3\cdot 2^j-1]$, VBT$[j+2][3\cdot 2^j]$) as the initial segment on the path through VBT[1][1], VBT[2][2], VBT[3][4], ..., VBT$[j][2^{j-1}]$, VBT$[j+1][2^j-1]$ is the same as (irrespective the order) the initial segment on the path through VBT[1][2], VBT[2][3], VBT[3][6], ..., VBT$[j+1][3\cdot 2^{j-1}-1]$. Since the indicator is chosen to be 0, VBT$[j+1][2^j-1]$ = VBT$[j+2][3\cdot 2^j]$.

□

Using the above proposition we show how the C-vine and D-vine for a given order can be obtained through two special sequences of indicators.

Proposition 2.3.2. *Suppose a VBT with given order $(l_0, l_1, \cdots, l_k)$, $(k \ge 3)$ is constructed through the extension process. If at every step of the extension indicator 1 is chosen, the partner element of the newly added element l_j $(3 \le j \le k)$ in depth-i $(2 \le i \le j)$ is l_{j-i+1} and a C-vine with the given order is obtained. If every choice is determined by indicator 0 then the partner element of the newly added element l_j in depth-i is l_{i-1} and a D-vine with this given order is constructed.*

Proof. The proof is by induction following the extension process for a VBT with given order $(l_0, l_1, \cdots, l_k)$. The extension process starts in dimension 4 $(k = 3)$ where the VBT with order (l_0, l_1, l_2) is given and l_3 is added. The partner element of l_3, VBT[2][2], is the only element where choice can be made. Its value can be taken as l_2 when indicator is 1 which gives us a C-vine with order (l_0, l_1, l_2, l_3) or l_1 when indicator is 0 which leads to a D-vine with the same order. We have discussed this case in detail in the example above.

Assume the proposition is true for $k = n$, hence all indicators are 1 and a C-vine with order $(l_0, l_1, \cdots, l_n)$ is constructed. The partner element of l_j $(3 \le j \le n)$ in depth-i $(2 \le i \le j)$ is l_{j-i+1}. Similarly, a D-vine with order $(l_0, l_1, \cdots, l_k)$ is obtained when all indicators are chosen to be 0 and the partner element of the newly added element l_j in depth-i is l_{i-1}.

When l_{n+1} is added then by Proposition 2.3.1, the partner element in depth-i, the element VBT$[i][2^{i-1}]$, is equal to VBT$[i+1][2^i+1]$ if all indicators are 1. Elements VBT$[i+1][2^i+1]$ in the current VBT (the VBT extended by l_{n+1}) are equal to VBT$[i][1]$ in the VBT before extension that represents a C-vine with order $(l_0, l_1, \cdots, l_n)$. The value of elements VBT$[i][1]$ in the VBT before extension is already known. The first one is VBT[2][1] = l_n and its partner element is VBT[2][2] = l_{n-1} in the VBT before extension. Element l_{n-1} will

appear in VBT[3][1] according to the definition of the standard form of a VBT. Then the partner element VBT[3][2] = l_{n-2} as $l_{n-1}l_{n-2}|D_e$ is the root node in tree-$(n-1)$. The value of VBT[4][1] is l_{n-2} and its partner element VBT[4][2] = l_{n-3} which is equal to VBT[5][1] because $l_{n-2}l_{n-3}|D_e$ is the root node in tree-$(n-2)$. Similarly we then have VBT[i][1] being $l_n, l_{n-1}, \ldots, l_2$ which are the partner elements of l_{n+1} up to depth-n in the current VBT. In depth-$(n+1)$ the partner element is l_1 according to Proposition 2.2.1 part 1) and 2). Thus the claim follows and the current VBT keeps the root nodes of the C-vine with order $(l_0, l_1, \cdots, l_n)$ unchanged. This leads to an extended C-vine with order $(l_0, l_1, \cdots, l_n, l_{n+1})$.

As for the D-vine case, by Proposition 2.3.1, the partner element of l_{n+1} in depth-i is equal to VBT[$i+1$][$3\cdot 2^{i-1}$] if all indicators are 0. Elements VBT[$i+1$][$3\cdot 2^{i-1}$] in the current VBT (the VBT extended by l_{n+1}) are the same as VBT[i][2^{i-1}] in the VBT before extension representing a D-vine with order $(l_0, l_1, \cdots, l_n)$. Notice that these elements are just the partner elements of l_n in each depth in the VBT before extension hence they are l_{i-1} in depth-i. Then the partner elements of l_{n+1} up to depth-n are $l_1, l_2, \ldots, l_{n-1}$. According to Proposition 2.2.1 part 1) and 2), the partner element in depth-$(n+1)$ is then l_n. Thus the new edge in the first tree of the extended by l_{n+1} vine is $l_{n+1}l_n$, which gives us a D-vine with order $(l_0, l_1, \cdots, l_n, l_{n+1})$. □

Proposition 2.3.2 generates a C-vine or a D-vine on VBT similarly with the way of how they are constructed on vine matrix as explained in Section 2.3.1. However if the given order is not set to be the natural order in VBT, then different C-vine or D-vine is generated. The elements are permuted but general type of structure might be unchanged.

APPLICATIONS

3

COMMON SAMPLING ORDERS OF REGULAR VINES WITH APPLICATION TO MODEL SELECTION

The selection of vine structure to represent dependencies in a data set with a regular vine copula model is still an open question. Up to date, the most popular heuristic to choose the vine structure is to construct consecutive trees by capturing largest correlations in lower trees. However, this might not lead to the optimal vine structure. A new heuristic based on sampling orders implied by regular vines is investigated. The idea is to start with an initial vine structure, that can be chosen with any existing procedure and search for a regular vine copula representing the data better within vines having 2 common sampling orders with this structure. Several algorithms are proposed to support the new heuristic. Both in the simulation study and real data analysis, the potential of the new heuristic to find a structure fitting the data better than the initial vine copula model, is shown.

Parts of this chapter have been published in Zhu et al. (2020).

3.1. INTRODUCTION

In this chapter, an extension of the idea proposed in Cooke et al. (2015) is discussed. In Cooke et al. (2015), the authors introduce the concept of sampling orders implied by a regular vine and notice that vines having more sampling orders in common have more common elements in their density decomposition. The idea proposed in Cooke et al. (2015) is that, if one wants to find a better fitting structure than the initial one, for a given data, the most promising way is to search for structures having none or a small number of common sampling orders with the initial vine. However, this claim has been evaluated only on one 4-dimension example. We extend their work and examine whether this heuristic search for a vine structure based on common sampling orders is a valuable method, also in higher dimensions.

This chapter is organized as follows. In Section 3.2, we briefly review and discuss sampling orders implied by a regular vine. Main contribution of this chapter, where algorithms allowing generation of all regular vines having given number of common sampling orders with the initial structure is shown in Section 3.3; Then we propose our heuristic search method and test its performance via simulation study in Section 3.4; In Section 3.5, our new heuristic is applied to real data sets to see if we can find a vine structure that performs better than ones obtained so far; Finally, we give conclusions in Section 3.6. Algorithms are presented in Appendix 3.A.2.

3.2. SAMPLING ORDERS

In this section, the definition of sampling orders will be introduced and algorithms to find the sampling orders implied by a vine will be explained.

3.2.1. DEFINITION

Sampling orders are related to density decomposition. There are many possibilities to factorize a density as introduced in Section 1.2.2 in Chapter 1. To give an example for 5 random variables $X_1, \ldots, X_5$, one possibility for the density f_{12345} is

$$f_{12345} = f_5 f_{4|5} f_{3|45} f_{2|345} f_{1|2345} \tag{3.2.1}$$

which can be further decomposed as

$$\begin{aligned} f_{12345} \quad = \quad & f_5 \cdot \\ & c_{45} \cdot f_4 \cdot \\ & c_{34;5} \cdot c_{35} \cdot f_3 \cdot \\ & c_{23;45} \cdot c_{24;5} \cdot c_{25} \cdot f_2 \cdot \\ & c_{12;345} \cdot c_{13;45} \cdot c_{14;5} \cdot c_{15} \cdot f_1 \end{aligned}$$

Notice that f_{12345} is decomposed according to the vine $V_1(5)$ represented by vine triangular array shown in Figure 3.1 [1]. Moreover, the decomposition above includes in each

[1] Note that the first tree in the vine triangular array does not need to be listed, hence in this chapter we do not show T_1 in vine triangular array for simplicity. The ordering of nodes in each tree is different in Cooke et al. (2015). We will introduce a standard form of vine triangular array later on in Section 3.2.2.

row, the five factors in Equation (3.2.1). The product over densities in the first four rows gives the density f_{2345}. If one removes all densities involving variable 1 from the decomposition, one gets margin of f_{12345}, f_{2345}. This process is referred to as **marginalization** with respect to variable 1.

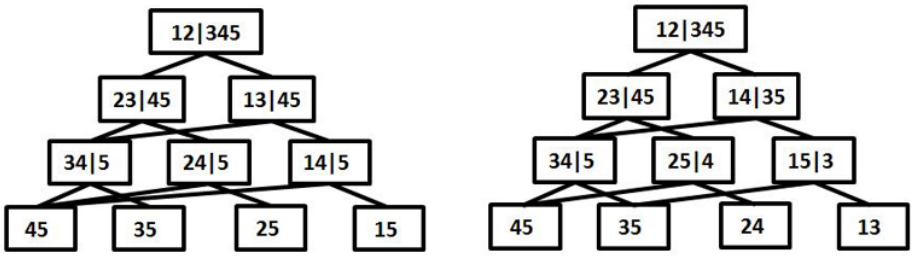

Figure 3.1: Vine triangular arrays for two regular vines on 5 elements (left - $V_1(5)$; right - $V_2(5)$)

Factorization (3.2.1) implies the sampling order $X_5 \to X_4 \to X_3 \to X_2 \to X_1$, which we simplify to $(5,4,3,2,1)$. The regular vine specification of $V_1(5)$ can be also obtained from another standard factorization, for example based on $(5,3,4,1,2)$. The following definition of a sampling order implied by a regular vine has been introduced in Cooke et al. (2015).

Definition 3.2.1. *(Sampling order implied by a Regular Vine) A **sampling order** for n variables is a sequence of conditional densities in which the first density is unconditional, and the density for other variables is conditioned on the preceding variables in the ordering. A **sampling order is implied by a regular vine** representation of the density if each conditional density can be written as a product of copula densities in the vine and one dimensional margins.*

We denote a sampling order as $(p_1, p_2, ..., p_n)$, where $p_i \in \{1, \ldots, n\}$ in this chapter.

There exist sampling orders which are not implied by a given regular vine. For example, vine $V_1(5)$ in Figure 3.1 will not imply the sampling order $(1,2,3,4,5)$, since representing the conditional density $f_{5|1234}$ would require, amongst others, variable 5 to be in the conditioned set of the top node in its triangular array.

In the next section, we will explain how all the sampling orders implied by a vine can be found.

3.2.2. SAMPLING ORDERS IMPLIED BY A VINE

Sampling orders implied by a regular vine are not difficult to find. We will at first generate them using vine triangular array, and extend slightly the terminology for this representation.

Since a node in tree T_i is an edge in tree T_{i-1} for $i > 1$, it is sometimes more effective to talk about **echelon-i** that denotes the set of nodes whose constraint sets have cardinality i, where $i = 1, \ldots, n$ (see, for example Cooke et al. (2015)). For a vine on n elements

there are n echelons. Nodes of the first tree $\{1,...,n\}$ belong to echelon-1 and are omitted in Figure 3.1. The top echelon, echelon-n, containing the single edge in tree T_{n-1} is referred to as the **top node** of vine $V(n)$. Later on in this chapter, the reference to nodes of regular vine $V(n)$ means the nodes in the triangular array of this vine. We call the two nodes in each echelon whose conditioned set contains element in the conditioned set of the top node as the **outside nodes** (these nodes are also called the leaf nodes in Chang & Joe (2019)).

It is possible to define a standard form of vine triangular array as follows. The marginalization by variable 1 in vine triangular array requires to remove all outside nodes whose conditioned set contains variable 1. This process is graphically represented in Figure 3.2. A subvine representing f_{2345} with top node 23|45 emerges from the marginalization and will be denoted as $V_1(4)$. Using concept of marginalization allows us to specify the **standard form of vine triangular array**. In the top node either element can be set to be the left element (thus there are always two standard forms for one vine). Then the elements of the conditioned sets of other nodes are ordered through consecutive marginalization with respect to the left element in the conditioned set of current node and setting the left element in the conditioned set of child node (after marginalization) as the right element in its conditioned set.

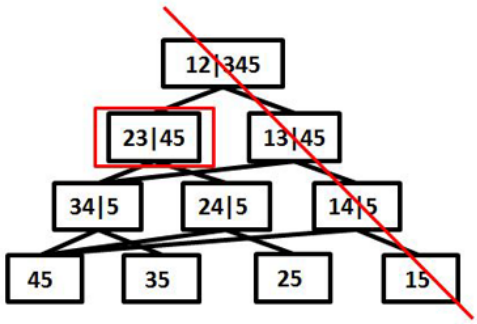

Figure 3.2: Graphical representation of marginalization for $V_1(5)$ by variable 1 to get $V_1(4)$.

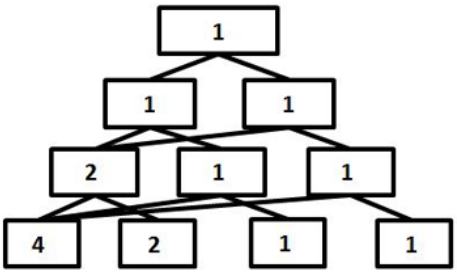

Figure 3.3: Number of subvines in triangular array of $V_1(5)$.

To find sampling orders two algorithms, bottom-up and top-down algorithms, have been proposed by Cooke et al. (2015). In this chapter the top-down algorithm will be considered. It starts by choosing the last element, p_n in the sampling order and performs successive marginalization steps. p_n is chosen from one of the elements in the conditioned set of the top node in vine triangular array. When the last element of the sampling order is chosen, say n, one can remove $f_{n|1,2,...,n-1}$ from the standard factorization, which is equivalent with marginalizing $f_{1,...,n}$ by n to get $f_{1,...,n-1}$ and removing all outside nodes in the triangular array representation of regular vine whose conditioned set containing variable n.

For $V_1(5)$ in Figure 3.1 the top-down algorithm will start by choosing an element from the conditioned set 12 of the top node. If we choose $p_5 = 1$, marginalization by variable

1 gives the subvine $V_1(4)$ with top node 23|45. Succeedingly, p_4 can be either 2 or 3. Setting $p_4 = 2$ and marginalizing $V_1(4)$ by 2 we obtain $V_1(3)$ with top node 34|5 and p_3 can be either 3 or 4. If p_3 is chosen to be 3, $V_1(2)$ is obtained which has top node 45 and p_2 is either 4 or 5. The first element p_1 in the sampling order is equal to the variable not chosen so far. This gives the sampling order $(5,4,3,2,1)$. The Top-down algorithm is presented in Algorithm 3.

Algorithm 3 Top-down algorithm

Input: a given regular vine $V(n)$
Output: a sampling order of $V(n)$

1: Set $i = n$. Choose one variable in the conditioned set of the top node of $V(n)$, say k, $p_n = k$.
2: **repeat**
3: Marginalize $V(i)$ by p_i to obtain a subvine $V(i-1)$.
4: Choose one variable in the conditioned set of top node of $V(i-1)$ and set it as p_{i-1}.
5: $i \leftarrow i-1$.
6: **until** Reach echelon-3, $i = 3$.
7: Choose one of two remaining variables as p_2 and assign the other to p_1.

Note that, at every step in echelon-i of the top-down algorithm, we have two choices for the value of p_i, thus the following remark holds:

Remark 3.2.1. *There are* 2 *possible values for* p_n, 2^2 *choices for* (p_{n-1}, p_n),..., 2^{n-2} *possibilities for* $(p_3, p_4, \ldots, p_n)$ *and* 2^{n-1} *ways of choosing* $(p_2, p_3, \ldots, p_n)$.

This is, in fact, the main idea of the proof of the next theorem, which is presented in Cooke et al. (2015).

Theorem 3.2.1. *Any regular vine on n elements implies* 2^{n-1} *sampling orders.*

The following remark holds.

Remark 3.2.2. *For a particular choice of* $(p_{n-i+1}, \ldots, p_n)$, *with* $1 \leq i \leq n-2$, *there are* $2^{n-1}/(2^i)$ *sampling orders that end with* $(p_{n-i+1}, \ldots, p_n)$.

The graphical representation of the number of subvines with given top node obtained by consecutive marginalization has been introduced in Cooke et al. (2015) and is depicted in Figure 3.3. There is just one vine with the top node 12|345 in $V_1(5)$, hence 1 in the top node. After marginalizing with respect to variable 1 (see Figure 3.2), we get one subvine in echelon-4 with the top node 23|45. Similarly the marginalization with respect to variable 2 gives one subvine with the top node 13|45, hence both nodes in echelon-4 in Figure 3.3 are assigned value 1. Furthermore, if we marginalize first with respect to 1 and then with respect to 2, or the other way around, we obtain the same subvine with the top node 34|5 in echelon−3, which leads to value 2. The other nodes are quantified similarly, e.g., the subvine with top node 45 can be obtained by marginalizing first by 1, then by 2 and by 3. However, three other options are also available, these are marginalizing by {1,3,2}, {2,1,3} and {2,3,1}. We observe that the number of subvines with node

$node_j$ in echelon-i as top node after marginalization, denoted as nsv_j, is equal to the sum of the number of subvines with top nodes corresponding to the parents of node $node_j$. The number of sampling orders can then be read from the number of subvines in the triangular array (see Figure 3.3), by doubling the sum over all values in echelon-2.

Sampling orders implied by a vine can also be easily found on VBT. Below is the VBT of $V_1(5)$.

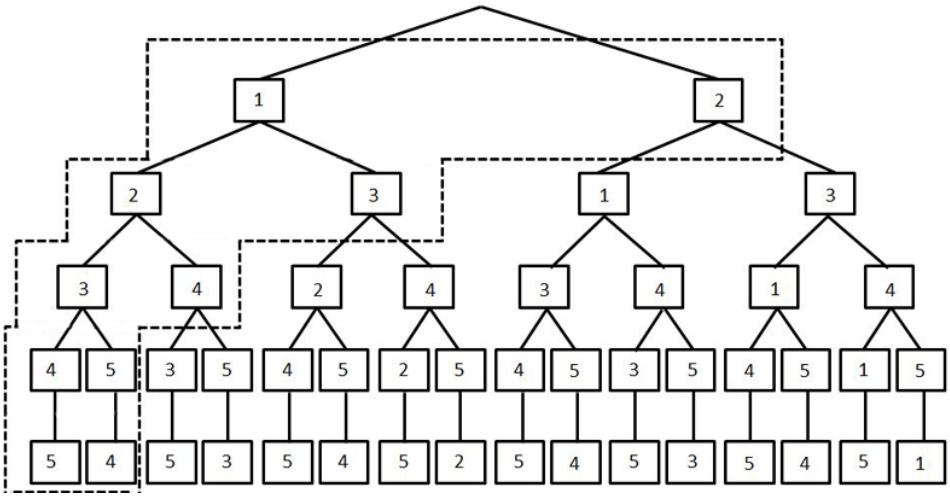

Figure 3.4: Standard VBT representation of $V_1(5)$. The dashed area denotes a substructure representing a path in the vine triangular array.

Note that the sampling orders found by Algorithm 3 are just paths in reverse order in the VBT in Figure 3.4, where element VBT$[i][j]$ in one path is equal to p_{n-i+1} in the sampling order. The VBT thus depicts all the sampling orders and we see immediately that vine $V_1(5)$ entails 16 sampling orders. The VBT of $V_2(5)$ reveals also 16 sampling orders, and, by comparing the two VBTs, we see that 10 of those sampling orders are common.

Obviously, if for any two vines all sampling orders are the same then we are dealing with the same vine. The more sampling orders two vines have in common the more their vine triangular array representation looks alike. This similarity can be observed from the density decomposition of these two vines. If we expand the density according to the factorization $(5,4,3,2,1)$, for vine $V_1(5)$ we get

$$f_{12345} = \boldsymbol{f_1 f_2 f_3 f_4 f_5 c_{45} c_{35} c_{34|5}} c_{25} c_{24|5} \boldsymbol{c_{23|45}} c_{15} c_{14|5} c_{13|45} \boldsymbol{c_{12|345}} \tag{3.2.2}$$

while for $V_2(5)$, the decomposition is

$$f_{12345} = \boldsymbol{f_1 f_2 f_3 f_4 f_5 c_{45} c_{35} c_{34|5}} c_{24} c_{25|4} \boldsymbol{c_{23|45}} c_{13} c_{15|3} c_{14|35} \boldsymbol{c_{12|345}}, \tag{3.2.3}$$

where common terms of the density are printed bold.

Theoretically, it does not matter which vine structure we select, since all structures represent the same density. However, in practice, the structure might play a crucial role.

For example, in the estimation process described e.g. in Dißmann et al. (2013), the limited choice of copula families, the errors due to tree-wise estimation and the simplifying assumption, lead to differences in performance of estimated vines with different structures for given data set. In Cooke et al. (2015), the authors proposed to search for vines that have none or a small number of common sampling orders with an initial structure, in order to improve the data fit. The main contribution of this chapter is to extend and test the performance of this heuristic, in simulations, as well as for real data sets.

3.3. COMMON SAMPLING ORDERS

In this section, we will show two algorithms, one to find only common sampling orders of two given regular vines and the other to generate all regular vines having a given number of common sampling orders with an initial regular vine.

3.3.1. FINDING COMMON SAMPLING ORDERS OF TWO GIVEN REGULAR VINES

We can find the common sampling orders of two vines by listing all sampling orders of each of the two vines and inspecting how many of those are common. This is however not efficient as the number of sampling orders grows exponentially with the dimension of the vine. Hence in this section, we present an algorithm to find only common sampling orders of two given regular vines.

Since paths in the VBT depict all sampling orders, finding common sampling orders requires choosing only common paths in VBTs. These common paths also appear in the vine triangular array, hence finding common sampling orders can also be carried out by finding common paths in the vine triangular array by comparing, echelon by echelon, elements in the conditioned set of **corresponding nodes** (with the same constraint set) of these vines. These common paths are then retained in VBT. If the conditioned sets in both triangular arrays representing two vines are the same, all paths through this node in one vine triangular array can be common, otherwise some paths are blocked, where a blocked path is represented by a 0 indicator of a line from the node in vine triangular array.

As an example, we show the procedure of finding common sampling orders between $V_1(5)$ and $V_2(5)$ in Figure 3.1. The top nodes of $V_1(5)$ and $V_2(5)$ are the same, which means we can marginalize them by either variable 1 or 2 (p_5 can be either 1 or 2 in common sampling orders), so the lines connecting the top node with its children are available. In echelon-4 the node 23|45 is the same in both vines hence both lines leading to m-children are available.

However, in the conditioned set for 13|45 of $V_1(5)$ and 14|35 of $V_2(5)$ only variable 1 is common thus variable 1 has to be fourth (p_4) in common sampling orders that ends with $p_5 = 2$. Other paths in $V_1(5)$ where $p_4 = 3$ are blocked. This is illustrated in Figure 3.5, where 3 is replaced by 0 and the indicator of the line connecting the child whose conditioned set contains 1 is set to 0. If all paths through a node in the next echelon are blocked then 0 is assigned to that node to indicate that this node does not need to be compared anymore. This procedure is called the **re-evaluation** of the vine triangular array. Similarly, the number of subvines in triangular array can also be re-evaluated, as shown in Figure 3.6. For $node_j$ in echelon-i, the number of subvines with this node as top node,

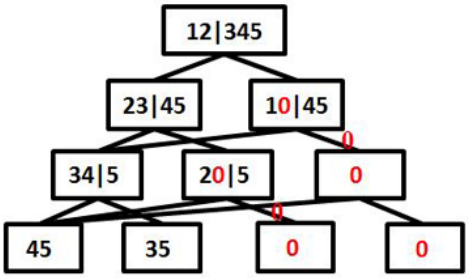

Figure 3.5: The triangular array of $V_1(5)$ after re-evaluation when comparing with $V_2(5)$.

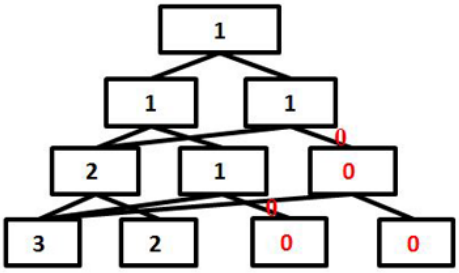

Figure 3.6: The number of subvines in triangular array of $V_1(5)$ after re-evaluation when comparing with $V_2(5)$.

nsv_j, is now the sum of numbers assigned to the parents of this node in echelon-$(i+1)$ but without those parents whose one line connecting this node is assigned 0 indicator.

In echelon-3, the nodes not assigned 0 till now are compared. Hence, as both elements in the conditioned sets of $V_1(5)$ and $V_2(5)$ for node 34|5 are the same both lines from node 34|5 stay available, and we block line between 25 and 24|5 as variable 2 has to be p_3 in the common paths crossing this node. We replace 4 by 0 in node 24|5, assign 0 to node 25 by re-evaluation.

The procedure is reflected in VBT, where corresponding elements in VBT of zero elements in the conditioned sets in vine triangular array are replaced by 0 thus each path in VBT through these elements is blocked and we can set all elements in sub-trees with these elements as root to be 0. This is graphically presented in Figure 3.7 where we can see directly how many and which common sampling orders $V_1(5)$ and $V_2(5)$ have by all paths in VBT that do not contain 0.

In general, when comparing the conditioned sets of corresponding nodes in echelon-i in triangular arrays representing two vines, $V_1(n)$ and $V_2(n)$, we distinguish three cases which will be of importance in the description of a general algorithm for finding vine with given number of common sampling orders.

Case1: No common element in the conditioned set.
This implies that the two vines cannot have a common value for p_{n-i+1} in paths through this node in triangular array. Thus all paths crossing this node are blocked and the indicators for both lines from this node are set to 0.

Case2: There is one common element in the conditioned set.
In this case the common element has to be chosen as p_{n-i+1} in paths through this node in triangular array. The other element cannot be p_{n-i+1} so the indicator of lines connecting the child whose conditioned set containing the common element is 0.

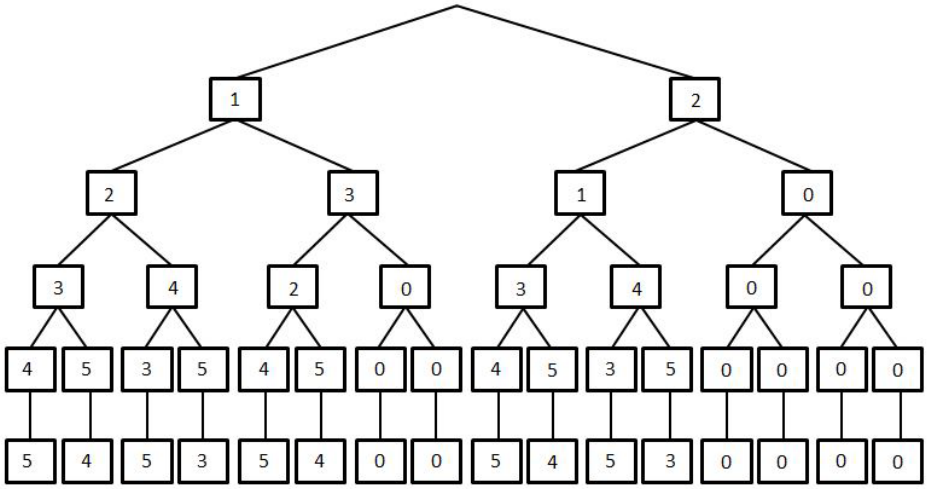

Figure 3.7: VBT representation of $V_1(5)$, when compared with $V_2(5)$.

Case3: Both elements of the conditioned sets are the same.
We have two choices for the value of p_{n-i+1} in paths through this node in triangular array.

We can observe that due to proximity few simple results can be stated about possible cases that can happen when two regular vines are compared. These results are presented below.

Proposition 3.3.1. *When nodes of two regular vines are compared the following properties are satisfied:*

a) Nodes in echelon-3 cannot be Case1.

b) The child of a node in Case3 cannot be in Case1.

c) For a node with a parent in Case3, the line to its child whose conditioned set does not include any element in the conditioned set of its parent, cannot be assigned 0.

d) There is at most one node in Case1. A Case1 node can appear if and only if these vines have 0 common sampling orders.

Proof. Parts a), b) and c) follow immediately from the proximity condition.

Part d) follows from the observation that if in a vine with top node in Case3 there is one node in Case1 in echelon-i, $3 < i < n-1$, then all paths, that are not blocked up to echelon-i, go through that Case1 node.

We prove this claim via induction on the dimension of the vine. When $n = i+2$ the only situation in which we can get Case1 node in echelon-i, according to b) and c), is the case when the two children of the top node are in Case2 and the Case1 node is their

common child. This means the two paths in the triangular array will go through this Case1 node and are then blocked.

Assume that the claim is true for $n \leq k$, $k \geq i+2$. We will show that the claim is true for $n = k+1$. We consider the following situations:

1) Assume that the top node is the only Case3 node, then its two children in echelon-k are in Case2 and their common child and other nodes in lower echelons are also Case2, thus the two paths from the top node will intersect in the only one node in echelon-$(k-1)$, due to c). So after in echelons lower than $(k-1)$ there is only one path which will be blocked after crossing the Case1 node in echelon-i.

2) Suppose that there are more nodes in Case3 and assume that the highest echelon containing a node in Case3 is echelon-j, $i+2 \leq j \leq k$, then

i) if $j < k$, then all nodes in echelons higher than j but lower than $k+1$ are in Case2. Hence both paths from the top node will go through the Case3 node in echelon-j, as in part 1). Since $j < k$ applying inductive step all paths in that subvine in dimension j will be blocked after the Case1 node.

ii) if $j = k$ and there is only one node out of two in echelon-k that is Case3, which means the other node is Case2. By c), the path through that Case2 node will go through the common child in echelon-$k-1$ and this common child is in the subvine with the Case3 node in echelon-k as top node. Applying induction, all paths in that sub-vine will finally be blocked after the Case1 node.

iii) if $j = k$ and both nodes in echelon-k are in Case3, due to c, the Case1 node can only be in the common part of these two k-dimensional subvines. Applying the inductive step proves the result.

Part d) follows from the above by noticing that if there were two different nodes in Case1, then paths through the nodes should meet in these nodes (since there must be one subvine that has Case3 top node and includes these two Case1 nodes). Then these two nodes will either be the same node or one of them is not Case1 but is assigned to 0. The second part of d) is immediate.

□

Another simple result for vines having 2 sampling orders in common is proved below.

Proposition 3.3.2. *Let $V_1(n)$ and $V_2(n)$, with $n \geq 3$, be two regular vines with 2 common sampling orders. Then all nodes compared in the procedure of finding common sampling orders are in Case2.*

Proof. By Proposition 3.3.1(d), all nodes in the triangular array are either Case3 or Case2. If all nodes are Case2, then one path remains available in the re-evaluation of the triangular array, which results in 2 common sampling orders. Assume there is one node in Case3. If this node is in echelon-3, then there are at least 4 common sampling orders. If this node is in a higher echelon, then two lines from this node to its children would be available hence there is an extra path through the triangular array, which would lead to more common sampling orders. □

The general algorithm to find the common sampling orders is presented in Algorithm 4 in Appendix 3.A.2.

If we denote the indicators for lines that are not 0 as 1, accounting for these indicators in a descending order of echelons, from left node to right in one echelon and from left to right line for the same node, leads to a sequence of indicators. The sequence of indicators of $V_1(5)$, when compared to $V_2(5)$ in the above example is $I = (11|0111|111110)$, where consecutive echelons are separated by vertical line $|$. As we have explained in this section the common sampling orders are determined by this sequence of indicators which is in fact the main idea how to generate all regular vines having given number of common sampling orders with the initial vine presented in Section 3.3.2.

3.3.2. GENERATING VINES HAVING A GIVEN NUMBER OF COMMON SAMPLING ORDERS WITH AN INITIAL VINE

We denote the number of common sampling orders of two vines $V_1(n)$ and $V_2(n)$ as $nComSO$, where $nComSO \in [0, 2^{n-1}]$. Recall that $nComSO$ can be determined from the re-evaluation of the number of subvines in the triangular array of $V_1(5)$, when compared with $V_2(5)$ shown in Figure 3.6. More specifically, $nComSO$ is twice the sum of values in echelon-2. Finding all possible vine triangular arrays after re-evaluations of $V_1(n)$ leads to finding all possible vines having $nComSO$ common sampling orders with $V_1(n)$. This will be carried out in three main steps.

Step 1) find all possible indicator sequences and apply re-evaluation in vine triangular array;

Step 2) choose values for zero elements in the conditioned sets of the non 0 nodes in triangular array after re-evaluation;

Step 3) choose values for the 0 nodes in triangular array after re-evaluation.

The choices in all above steps have to be made such that the proximity condition is satisfied.

FINDING ALL POSSIBLE INDICATORS

Indicators assigned to lines in the triangular array determine the number of common sampling orders two regular vines will have. They uniquely determine which nodes are assigned 0 and which cases appear in nodes that are not 0. Not all assignments of indicators to lines in the triangular array, however, are consistent. Constraints on the assignments follow from Proposition 3.3.1.

The number of sampling orders is related to the number of paths in the triangular array. The number of paths crossing a given node, $node_j$, in echelon-i is equal to the product 2^{i-1} and the number, nsv_j, of subvines with this node as top node obtained by marginalization (as explained in Section 3.2.2 this information is contained in Figure 3.3).

Suppose that we want to generate indicator sequences of vines on n elements that have $nComSO$ sampling orders in common with the initial vine. This means that we have to assign some 0 indicators to lines in the triangular array to remove $A_n = 2^{n-1} -$

$nComSO$ sampling orders. The assignment of 0 indicator to a line from $node_j$, in vine triangular array, blocks half of paths through this node. As shown in Remark 3.2.2, the number of sampling orders that we remove by assigning 0 indicator to one line from $node_j$ in echelon-i is equal to $2^{i-1}/2$ times nsv_j, which we call the **product** corresponding to $node_j$. Assigning both lines 0 indicators blocks all paths leads to double the product. However, this option is available only if vines with $nComSO = 0$ common sampling orders with the initial vine are required according to Proposition 3.3.1 (d).

The algorithm of finding possible indicator sequences is recursive and starts by finding a node in the highest echelon of the triangular array, whose product is smaller than or equal to A_n. We will subtract from A_n the number of sampling orders that can be removed by different assignments of 0 indicators to lines from this node. If we assign one line in $node_j$ to 0 then

$$L = 2^{i-2} \cdot nsv_j \tag{3.3.1}$$

sampling orders could be removed. However, if we were to assign none of the lines 0 the upper bound on the possible removals later on in the algorithm, denoted as U, can be set as

$$\begin{aligned} U = \{&\text{the sum of the products over non 0 nodes not considered in echelon-}i \\ &\text{with one line assigned 0 and products of all non 0 nodes in} \\ &\text{echelon-}(i-1)\text{ after re-evaluation with both lines assigned 0.}\} \end{aligned} \tag{3.3.2}$$

Note that when algorithm is in echelon-4, nodes in echelon-3 can be assigned only one 0 indicator. If at any step of the procedure $A_n \geq U$ then option of not assigning any 0 to lines from this node is not available. Each choice of the assignments of 0 indicators has to be consistent with Proposition 3.3.1.

The example of generating indicator sequences of vines having 10 sampling orders in common with $V_1(5)$ is as follows: since $nComSO = 10$, assigning both lines 0 is not possible. Initially $A_n = 2^4 - 10 = 6$. The product of the node 13|45 is equal to $2^2 \cdot 1$ which is the first one smaller than 6, hence we start the algorithm in this node. In this case $L = 2^2 \cdot 1 = 4$ and only the line to node 14|5 can be assigned 0 indicator due to Proposition 3.3.1 (c). $A_n = 6 - L = 2$ and re-evaluation is required to include assignment of 0 which is shown in Figure 3.6. The next node in the highest echelon with largest product smaller than 2 is 24|5. We can only set 0 indicator to line to node 25, where $L = 2$ and get $A_n = 2 - L = 0$. This choice is consistent with Proposition 3.3.1 and leads to the indicator sequence $I_1^{(10)} = (11|0111|111110)$. If we consider not assigning any 0 to lines from the node 25, we get $U = 0 < A_n$ which means that this choice is not available.

At node 13|45 we could have considered not assigning 0. To decide if not assigning any 0 would be possible we compute $U = 2^2 \cdot 1 + 2 \cdot 1 + 2 \cdot 1 + 2 \cdot 1 = 10$. Since $U > A_n$, there are enough possibilities to remove sampling orders later on in the process. Hence not assigning any 0 is possible and we choose not to assign 0. The algorithm leads us to the next node in echelon-4 with product equal to $L = 4 < A_n$, hence to node 23|45. Assigning one 0 to line to node 24|5, leads to $A_n = 2$ and 0 can be assigned to line connecting 14|5 and 15 to get $A_n = 2 - 2 = 0$ which gives another indicator sequence

$I_2^{(10)}$ = (11|1110|011111). Similarly last two indicator sequences can be obtained $I_3^{(10)}$ = (11|1111|011011) and $I_4^{(10)}$ = (11|1111|111010).

Having a possible indicator sequence allows us to see directly which case happens in which node in vine triangular array. For example, $I_1^{(10)}$ = (11|0111|111110) denotes Case2 happens in the node 13|45 and 24|5, thus they become 10|45 and 20|5. The rest of nodes in echelons larger than 3 are in Case3. Together with re-evaluation this leads to the triangular array in Figure 3.5.

A general algorithm to find possible indicators is presented in Algorithm 5 in 3.A.2. Following the algorithm, we can get that all possible indicator sequences for regular vines having 0 sampling orders in common with $V_1(5)$ are: $I_1^{(0)}$ = (01|1100|111111), $I_2^{(0)}$ = (10|0011|111111) and $I_3^{(0)}$ = (00|1111|111111). Sequence $I_1^{(0)}$ will be used later on in the following subsections when we introduce an extra example.

CHOICE FOR THE ZERO ELEMENTS IN THE CONDITIONED SETS OF THE NON 0 NODES IN TRIANGULAR ARRAY AFTER RE-EVALUATION

In this subsection we show how to replace the zero elements in the conditioned sets of non 0 nodes by the elements in the conditioning set. The procedure starts with nodes in Case2 from lowest echelons where fewer choices for zero elements are possible. Then it considers the zero elements in the Case1 node if such node exists. All choices are made such that proximity condition is satisfied. We present the idea using two examples and then proceed with the general algorithm.

The zero element in node 20|5 in Figure 3.5 can only be 5 thus we will get 25|4, whereas we have two choices for the zero element in node 10|45, one is 4 and the other is 5. However, due to proximity, the chosen element must appear in the conditioned set of node 34|5, hence the node can only be 14|35. All nodes with Case2 have been handled and there is no node in Case1 in this example, the procedure stops and the result is shown in Figure 3.8. Note that possible choices of zero elements in the conditioned sets of the nodes lead to 14|35 and 25|4 as in case when V_1 is compared with V_2. However, V_2 is not the only vine that has 10 common sampling orders with V_1 for this indicator sequence. There are still some choices for 0 nodes in the triangular array. They will be considered in the next subsection.

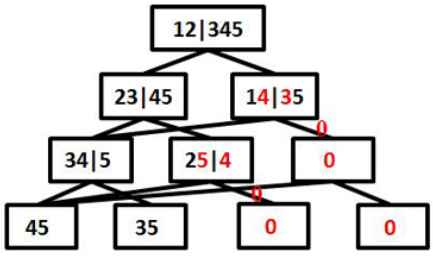

Figure 3.8: The triangular array of $V_1(5)$ after re-evaluation with choices of zero elements in the conditioned sets satisfying proximity.

Next, we present one more example, where vines with 0 common sampling orders with $V_1(5)$ are of interest. We consider the indicator sequence $I_1^{(0)} = (01|1100|111111)$ that denotes Case2 in the top node and Case1 in node 23|45, which are set to 10|345 and 00|45 and the triangular array after re-evaluation is shown in Figure 3.9 (left). The zero element in 10|345 can be chosen out of elements of its conditioning set $\{3,4,5\}$ and has to be in the conditioned set of its child by marginalization with 1. This child is a node with Case1. Its zero elements can only be chosen as $\{4,5\}$ and it becomes 45|23. This means that, in order to replace zero in the top node, only two choices are available, that is $\{4,5\}$. When 4 is chosen, we get a triangular array as in Figure 3.9 (right).

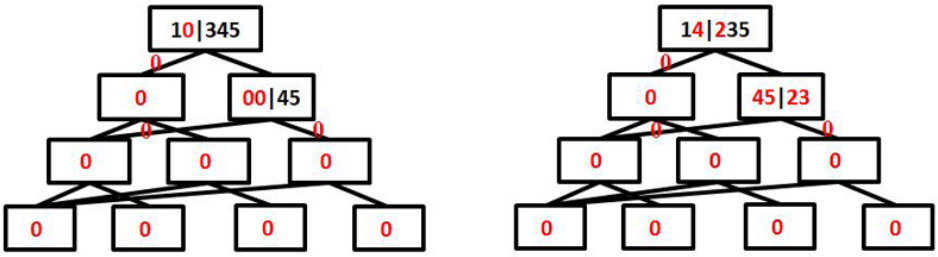

Figure 3.9: The triangular array of $V_1(5)$ with indicators $I_1^{(0)}$ after re-evaluation (left); with choices of zero elements in the conditioned sets satisfying the proximity condition (right).

A general algorithm to choose for the zero elements in the conditioned sets of the triangular array is presented in Algorithm 6 in Appendix 3.A.2.

CHOICE OF THE 0 NODES IN TRIANGULAR ARRAY AFTER RE-EVALUATION

The procedure of this subsection basically follows Algorithm 2 in Chapter 2. However, in this section, we will introduce another algorithm that is a bit different. This is because the largest completely specified subvine in VBT may not always be the substructure determined by a given order as in Algorithm 2 in Chapter 2.

Nodes assigned to 0 in the triangular array after re-evaluation are nodes that can be simply chosen such that the proximity condition is satisfied. The main idea is similar to extending regular vines based on natural order presented in Nápoles (2011). It uses properties of regular vines which ensure that two vines on $1,\ldots,i$ and $2,\ldots,i+1$ elements, denoted by $V([1,i])$ and $V([2,i+1])$, respectively, can be combined into a vine on $1,\ldots,i+1$ elements, denoted by $V([1,i+1])$, with top node $1,i+1|2,...,i$ and the overlapping part which is $V([2,i])$. Hence if $V([1,i])$ is the largest completely specified (has no 0 node) and there are some unspecified elements (0 nodes) in $V([1,i+1])$ then these elements can be only in the part of $V([1,i+1])$ that is not common with $V([1,i])$. Since node $1,i+1|2,...,i$ is specified then unspecified elements are in $V([2,i+1])$ but not in $V([2,i])$. If the top node of $V([2,i+1])$ is unspecified, since top nodes of $V([1,i])$ and $V([2,i+1])$

are siblings, then one element in the conditioned set of the top node of $V([2,i+1])$ has to be in the conditioned set of the top node of $V([1,i+1])$ (has to be $i+1$) and the other element should be either elements in the conditioned set of the common child with its sibling. Similarly choices can be made if top nodes of smaller subvines of $V([2,i+1])$ are unspecified.

A vine triangular array contains implicit parent/child relationships of the initial vine, which complicates the discussion of possible vine structures having $nComSO$ sampling orders in common. VBT contains all information in the vine triangular array, as 0 nodes in the triangular array become subtrees of 0 elements in VBT, and the idea of choosing unspecified elements of a vine, as explained above can be presented more naturally in VBT.

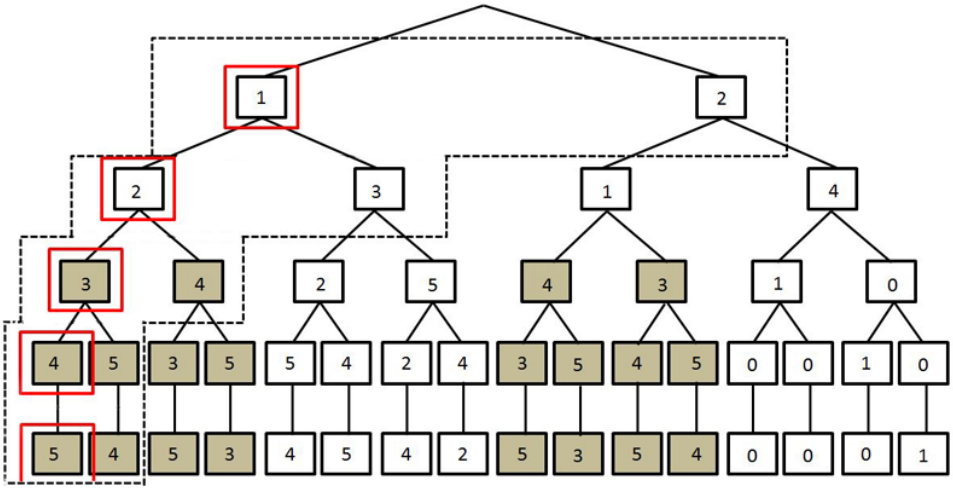

Figure 3.10: Standard VBT corresponding to the triangular array in Figure 3.8.

Our procedure requires an ordering of elements of the vine, which will be used to perform consecutive extensions of vines. We are considering first the situation when $nComSO > 0$, whereas vines having 0 common sampling orders with the initial vine will be handled later on. If $nComSO > 0$, there exist paths in the vine triangular array and we start with one such path. This path, as we mentioned in Section 2.2.3 in Chapter 2, corresponds to a substructure in VBT. For instance the path through nodes: 12|345, 23|45, 34|5 and 45 in the triangular array in Figure 3.5 corresponds to the substructure in VBT in Figure 3.10, indicated by pair of elements in the dashed area. This path gives us an ordering of variables $\{5,4,3,2,1\}$ (elements in red squares).

We see that the largest completely specified subvine in Figure 3.10 is on elements $\{5,4,3,2\}$. The procedure will extend this vine with the next element in the ordering. When using the VBT representation, the algorithm starts in VBT[2][1] = 2, the most left element in pair of elements that represents the top node of the completely specified vine, and makes an **Ascend step** to VBT[1][1] = 1 in order to find a subvine on elements $\{5,4,3,1\}$, whose top node is a sibling of the top node of the vine on $\{5,3,4,2\}$. In VBT

the top nodes of these subvines are nearby pairs of element (VBT[2][1], VBT[2][2]) and (VBT[2][3], VBT[2][4]). The algorithm makes then a **Descend step** to the subvine corresponding to the top node (VBT[2][3],VBT[2][4]). Due to (4) in Properties of VBT, sub-trees underneath VBT[2][1] = 2 and VBT[2][3] = 1 are the same, except possible ordering to keep the VBT in the standard form (see shaded nodes in Figure 3.10). Element VBT[2][4] is not 0 then the descend step is followed to VBT[3][7], the most left element in pair of elements represents top node of vine on $\{1,3,5\}$, where another subvine on $\{3,4,5\}$ has no 0 elements. Due to property (4), the sub-tree underneath VBT[3][5] = 4 is the same as the sub-tree underneath VBT[3][7] = 1 and according to property (3), VBT[3][8] can be chosen to be equal to VBT[4][13] = 3 or VBT[4][14] = 5. We choose VBT[3][8] to be equal to 3 and other elements in VBT can be filled in according to VBT properties (1,2). The resulting VBT is shown in Figure 3.11 and the corresponding triangular array is in Figure 3.14. Notice that the vine we obtained is not $V_2(5)$.

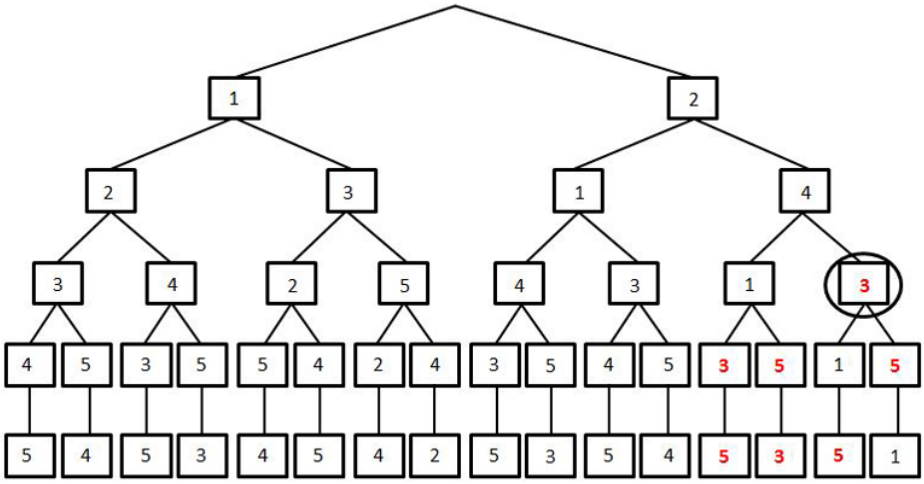

Figure 3.11: VBT of a vine with 10 common sampling orders with $V_1(5)$, where the choice made in the algorithm is shown in the black circle.

In order to specify vines having 0 sampling orders in common with an initial vine, an extra step is needed to start the algorithm presented above. This step involves the construction of a substructure in VBT. The example shown in the triangular array after re-evaluation in Figure 3.9(right) will be used to introduce the idea. We start with the longest specified path, one that contains the most specified nodes (in the example this is a path containing nodes 14|235, 45|23). We get an incompletely specified substructure on elements 5,4,1 in red squares in Figure 3.12. The remaining elements of the substructure can be chosen such that the VBT satisfies the properties and is in standard form. The ordering required for consecutive extensions can be $\{3,2,5,4,1\}$ as shown in Figure 3.12, or $\{2,3,5,4,1\}$.

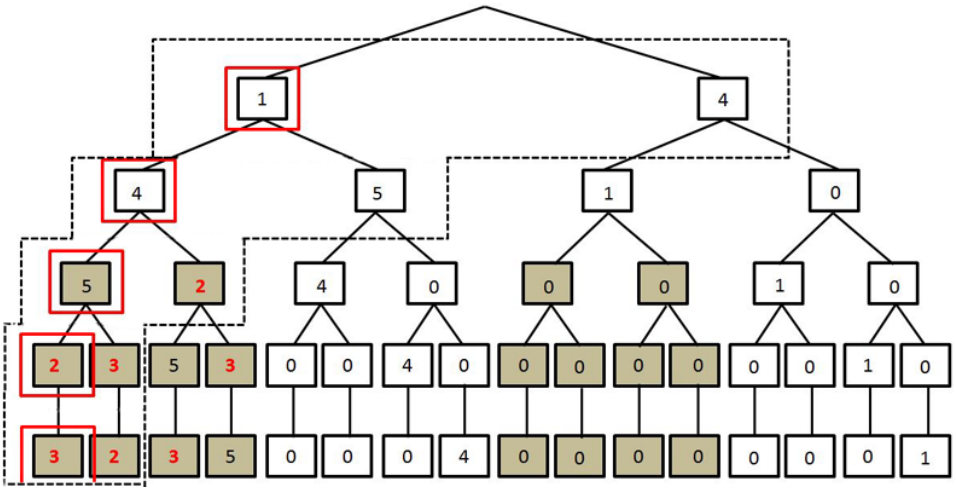

Figure 3.12: Standard VBT with chosen substructure of a vine with 0 common sampling orders with $V_1(5)$.

We can observe that the largest fully specified part of the VBT is a subvine on $\{5,2,3\}$ and, in order to extend it by element 4, an ascend step from VBT[3][1] = 5 to VBT[2][1] = 4 is performed. We find another subvine on $\{4,2,3\}$. We make a descend step to the vine on $\{4,2,3\}$ and since VBT[3][4] = 0, we can choose it to be equal to 2 or 3. When 2 is chosen, by applying property (1), we then have a completely specified vine on $\{4,5,2,3\}$. Next, another ascend step from VBT[2][1] = 4 to VBT[1][1] = 1 is performed to extend the subvine by 1. The subvine where there are 0 elements is on $\{3,2,5,1\}$. Performing a descend step, the sub-tree underneath VBT[2][3] = 1 is the same as sub-tree underneath VBT[2][1] = 4 (with possibly different ordering). The element VBT[2][4] can be chosen to be either 2 or 5 and we set VBT[2][4] = 2. From VBT[2][3] = 1, the next descend step is made to the subvine on $\{3,5,1\}$, where element VBT[3][8] can be chosen to be equal to 5 or 3. The result is shown in Figure 3.13 and the corresponding triangular array in Figure 3.15. The general algorithm is presented in Algorithm 7 in Appendix 3.A.2.

3.3.3. THE NUMBER OF VINES HAVING *nComSO* COMMON SAMPLING ORDERS WITH AN INITIAL VINE

In the previous subsections we have shown how to generate vines with *nComSO* sampling orders in common with the initial vine. We have followed three steps to generate such vines: generating indicator sequences, choosing zero elements in the conditioned sets of non 0 nodes and making choices for 0 nodes in the triangular array after re-evaluation. In this section, a few results about the number of regular vines having different *nComSO* sampling orders in common are presented.

In general the number of vines having *nComSO* sampling orders with initial vine depends on the structure of the initial vine. However, we show below two cases in which *nComSO* does not depend on the initial structure.

Proposition 3.3.3. *The number of vines on n elements having* $2^{n-1}-2$ *sampling orders*

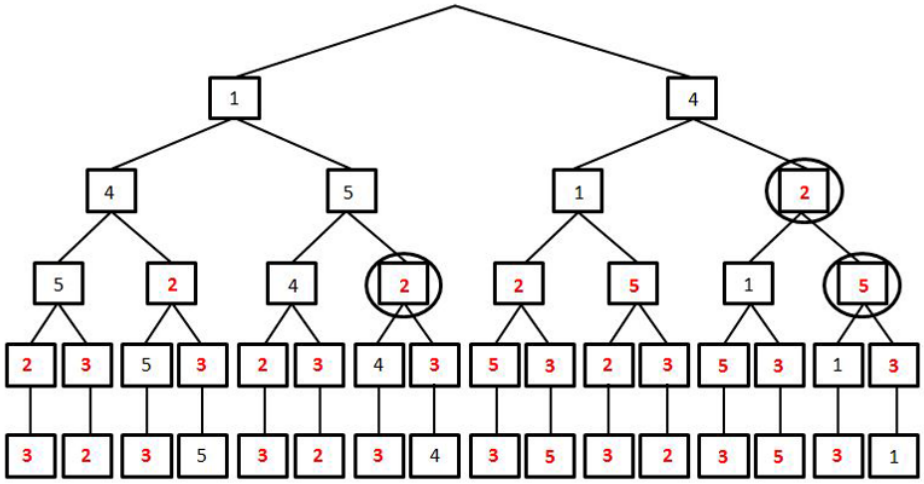

Figure 3.13: VBT of a vine with 0 common sampling orders with V_1 where choices made are indicated by black circles.

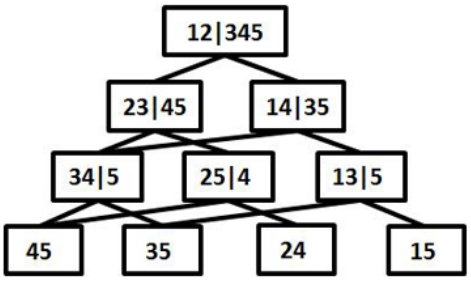

Figure 3.14: Vine triangular array of a vine having 10 common sampling orders with $V_1(5)$, corresponding to VBT in Figure3.11.

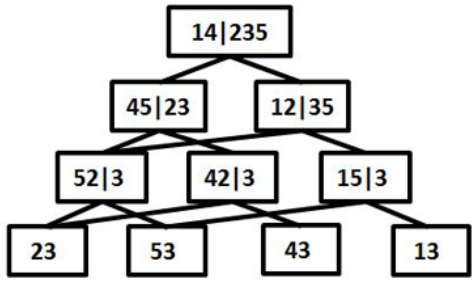

Figure 3.15: Vine triangular array of a vine having 0 common sampling orders with $V_1(5)$, corresponding to VBT in Figure3.13.

in common with an initial vine is equal to 2.

Proof. We consider the three steps to generate regular vines having $nComSO = 2^{n-1}-2$ sampling orders in common with an initial vine. In the first step, since $A_n = 2^{n-1}-(2^{n-1}-2) = 2$, there are only two nodes in the triangular array whose product is smaller than or equal to A_n. These are the outside nodes in echelon-3. To obtain A_n finally equal to 0, we have to assign 0 indicator to one line from either of these two nodes by Proposition 3.3.1 (c). This leads to only two possible indicator sequences. For each indicator sequence, the zero element in the conditioned set can only be the element in the conditioning set since the node is in echelon-3. There is no node assigned to 0, so the triangular array can be fully specified by the first two steps. Thus the number of vines having $2^{n-1}-2$ sampling orders in common is always equal to 2. □

Proposition 3.3.4. *The number of vines on n elements having* 2 *sampling orders in common with an initial vine is equal to* $2\cdot 3^{n-3}\cdot 2^{\binom{n-2}{2}}$.

Proof. According to Proposition 3.3.2 for vines having 2 common sampling orders, Case2 has to happen in each node during comparison. Thus one line from each Case2 node is assigned 0 indicator and re-evaluation is applied such that all remaining nodes are set to 0. This gives us particular choices for the possible indicator sequences which together with the choices for the zero elements in the conditioned sets of Case2 nodes will allow us to count the number of ways one path through the triangular array can be obtained in steps 1) and 2) discussed in Section 3.3.2.

We have 2 choices for the element to be common in the top node. The other one will be set to 0 and we can choose this zero element, due to proximity condition, to be either element in the conditioned set of its child. However, if the common element in the conditioned set of its child is the non-common element in its conditioned set, then we only have one choice (otherwise we get the same node). Moreover, there is only one choice for the Case2 node in echelon-3. Hence, our path through the triangular array can be such that for all $n-3$ nodes (except the node in echelon-3) on this path there are two choices for the zero element in the conditioned set: one such a path with 2^{n-3} choices. Or there can be one node out of $n-3$ with only one choice and the rest with two choices: $\binom{n-3}{n-4}$ paths with 2^{n-4} choices, or two such nodes etc., which gives

$$\sum_{j=0}^{n-3}\binom{n-3}{j}2^j = 3^{n-3}$$

choices for the path. A path in the triangular array after one of these choice corresponds to a substructure in VBT. By Proposition 3.3.4 in Appendix 3.A.1, we can see that step 3 in our algorithm corresponds to consecutive extensions of vines and leads to $2^{\binom{n-2}{2}}$ regular vines after step 3).

Combining all above we get: 2 choices of common element in the top node, 3^{n-3} ways to form a path through the triangular array and $2^{\binom{n-2}{2}}$ choices in step 3), which gives that the total number of vines with 2 sampling orders in common with an initial vine is equal to $2\cdot 3^{n-3}\cdot 2^{\binom{n-2}{2}}$. □

From simulations we observe that the number of vines having 0 common sampling orders does not depend on the initial structure but we are not able to provide a proof. In Proposition 3.3.5 we show a lower bound for the number of vines having 0 common sampling orders with an initial vine.

Proposition 3.3.5. *A lower bound for the number of vines on n elements having* 0 *sampling orders in common with an initial vine is* $\binom{n-2}{2} \cdot (n-2)! \cdot 2^{\binom{n-2}{2}}$.

Proof. Consider the situation where Case1 occurs in the top node. This means that any 2 variables out of $n-2$ in the conditioning set, hence $\binom{n-2}{2}$ possibilities, can be chosen. To construct the substructure in VBT, any permutation of remaining variables is available, which gives $(n-2)!$ choices. Moreover, we can generate $2^{\binom{n-2}{2}}$ regular vines from each substructure. Hence the lower bound for the number of vines having 0 common sampling orders is $\binom{n-2}{2} \cdot (n-2)! \cdot 2^{\binom{n-2}{2}}$. □

Using this crude lower bound we can observe that the proportion of vines having 0 sampling orders in common with given initial vine and the number of all regular vines converges to 1 as n goes to infinity.

3.4. SIMULATION STUDY

As we have explained in the end of Section 3.2, the number of common sampling orders of two vines indicates 'similarity' in the density decomposition of these vine structures. In this section, we will test how this 'similarity' relates to the performance of different vine structures on a data set in terms of the Akaike Information Criterion (AIC). We start with a detailed discussion of an example using a 5 dimensional regular vine where we will explore the differences in $AICs$ in various modeling setups and for different number of common sampling orders with an initial structure. Furthermore, we propose a heuristic search procedure, which starts with an initial structure and searches for a vine structure with an AIC lower than that of the initial one.

3.4.1. SIMULATION OF A 5 DIMENSIONAL REGULAR VINE

An accurate estimation of a vine copula requires specification of the vine structure, as well as a good fit of the (conditional) copulas. We want firstly to investigate how a misspecification of the vine structure, as well as of copula (families) does affect the goodness of fit of the model. The goodness of fit of models will be evaluated in terms of the AIC and a goodness-of-fit test based on probability integral transformation (PIT) test [2]. A structure is hence randomly selected using the function `RVineMatrixSample` in the **VineCopula** package, with copula families chosen randomly out of Gumbel(G) and Clayton(C) copulas, as well as their rotated versions. Note that, for consistency reasons, we keep the same notation of families of copula as in the **VineCopula** package. Kendall's correlations, chosen to find parameters of these copulas, were simulated from a *Beta(2,2)* distribution, and are allowed to take negative values with probability 0.5. The

[2] PIT test is introduced in Breymann et al. (2003) which is based on the Rosenblatt's probability integral transform Rosenblatt (1952). It is known to perform very well when data does not need to be ranked (as is the case in our simulation study)

chosen structure, as well as the chosen set of copula families, is denoted by $\mathscr{C}_0$, and the corresponding Kendall's correlations τ_0 are depicted in Table 3.1.

This regular vine, with uniform margins will be denoted by V_0. Furthermore, 1000 observations are sampled from vine V_0 and two cases will be considered. The data is firstly fitted using the same vine structure and fixing copula families $\mathscr{C}_0$. We denote this fitted vine as V_1, and the estimated Kendall's correlations ($\hat{\tau}_1$), as well as goodness of fit measures (AIC_1 and p_1) are reported in Table 3.1. Finally, we keep fixed vine structure of V_0, but choose copula families that minimize AIC over all available copulas in the **VineCopula** package. The copula families might not coincide with $\mathscr{C}_0$ and will be denoted by $\mathscr{C}_2$. This provides a regular vine denoted as V_2, and the estimated Kendall's correlations ($\hat{\tau}_2$) and measures of fit (AIC_2 and p_2) are also reported in Table 3.1.

Vine Structure	V_0 $\mathscr{C}_0$	τ_0	V_1 $\hat{\tau}_1$	AIC_1	p_1	V_2 $\mathscr{C}_2$	$\hat{\tau}_2$	AIC_2	p_2
echelon-5									
54|123	G270	-0.74	-0.71	-1725.60	0.956	G270	-0.70	-1661.86	0.903
echelon-4									
34|12	G270	-0.49	-0.47	-651.72	0.960	G270	-0.48	-654.03	0.945
25|13	G	0.69	0.68	-1526.08	0.438	G	0.68	-1497.84	0.454
echelon-3									
24|1	C270	-0.56	-0.56	-1071.77	0.902	C270	-0.56	-1070.78	0.841
35|1	C270	-0.13	-0.13	-59.6726	0.401	BB8_90	-0.13	-63.22	0.702
32|1	C	0.57	0.57	-1129.16	0.409	C	0.57	-1129.16	0.409
echelon-2									
14	C180	0.80	0.80	-2570.13	0.947	J	0.80	-2570.35	0.921
15	C90	-0.79	-0.80	-2560.66	0.743	J270	-0.80	-2561.96	0.819
12	G	0.40	0.42	-495.39	0.399	G	0.42	-495.39	0.399
13	C180	0.41	0.43	-616.08	0.308	C180	0.43	-616.08	0.308
$AIC_{V_1} = -12406.26$						$AIC_{V_2} = -12320.65$			

Table 3.1: An example of a 5 dimensional regular vine, V_0, represented via a vine triangular array, echelon by echelon, along with copula families $\mathscr{C}_0$ and Kendall's correlations τ_0. 1000 observations are sampled from V_0, and data are fitted using the same vine structure and the same copula families $\mathscr{C}_0$, (V_1), and by using the fixed structure and different copula families $\mathscr{C}_2$, which minimize AIC over all available families in the **VineCopula** package, (V_2). Kendall's correlation estimates and goodness of fit measures for bivariate copulas in V_1 (τ_1, AIC_1 and p_1) and V_2 (τ_2, AIC_2 and p_2) are presented. The AIC for regular vine in V_1 and V_2 are AIC_{V_1} and AIC_{V_2}, respectively.

We further explore how the misspecification of the structure affects the goodness of fit of the vine. The data generated from V_0 are now fitted using the heuristic method presented in Dißmann et al. (2013). A vine structure is chosen, tree by tree, by maximizing the sum of absolute values of Kendall's correlations for pairs of variables, and we denote this vine structure as V_{Diss}. This heuristic has been implemented into the built in function `RVineStructureSelect`. Table 3.2 reports the results of fitting.

Vine Structure	Copula Family	$\hat{\tau}_{Diss}$	AIC_{Diss}	PIT test p_{Diss}
echelon-5				
42|153	t	0.59	-419.20	0.585
echelon-4				
12|53	BB1	0.61	-1159.29	0.0822
34|51	Tawn90	-0.74	-1947.66	0.186
echelon-3				
52|3	BB8	0.10	-46.61	0.740
54|1	Tawn90	-0.06	-31.01	0.740
13|5	Tawn2_180	0.04	-16.62	0.375
echelon-2				
32	BB1	0.66	-1431.90	0.640
14	J	0.80	-2570.35	0.921
53	$BB7\_270$	-0.46	-655.23	0.620
51	J90	-0.80	-2561.96	0.957
$AIC_{Diss} = -10839.81$				

Table 3.2: Vine triangular array, echelon by echelon, of a regular vine V_{Diss}, obtained by using the method in Dißmann et al. (2013) to fit the 1000 observations sampled from V_0 in Table 3.1. Kendall's correlations ($\hat{\tau}_{Diss}$), along with goodness of fit measures (AIC and p_{Diss}). The AIC_{Diss} depicts the overall goodness of fit of V_{Diss}.

We can draw a few conclusions from the results presented in Table 3.1 and Table 3.2.

- When the structure and the copula families are fixed (V_1), the quality of Kendall's correlation estimates deteriorates slightly for nodes in higher echelons. This is caused by the tree-wise estimation.

- When the structure is kept fixed and the copula families are chosen such that they minimize the AIC over all available copula families in the **VineCopula** package (V_2), different copula families than the ones used for simulation might be chosen. These copulas have smaller AIC and they fit the data well according to the PIT goodness-of-fit test(e.g. p-values for copulas 15 and 14 are 0.743 and 0.947, when copula families are fixed and 0.819 and 0.921 when they are chosen to minimize AIC, respectively). Additionally, we checked that we cannot reject the hypothesis that V_2 is a simplified vine for the data at 5% significance level by using the `pacotest` (or called CCC test) in **pacotest** package (Kurz (2019)). However, V_1 is significantly better than V_2 according to Vuong test (Vuong (1989)) at 5% level. Hence, this example illustrates that choosing copula families by minimizing AIC cannot ensure the global minimum AIC for the whole regular vine.

- When the vine structure is chosen by constructing trees having the largest sum of absolute empirical Kendall's τ, a different structure is selected to best fit the data. Even though the sum of AIC for the first tree is much larger than the corresponding value for the true structure, the overall fit is much worse. Note that $AIC_{Diss} = -10839.81$, as compared to $AIC_{V_1} = -12406.26$. Moreover, note that the p-values of PIT test for copulas 32 and 53 which are not directly specified in

the true structure are 0.640 and 0.620, respectively, which implies these bivariate margins are well modeled by the available parametric copulas in the package. We cannot reject the hypothesis that V_{Diss} is a simplified vine with `pacotest` at 5% level. Even though we can fit the bivariate margins by a parametric copulas well according to the PIT test and this vine structure is a simplified vine by `pacotest`, the final AIC is significantly larger than the initial vine according to the Vuong test. This emphasizes the importance of choosing a proper regular vine structure.

To compare the performance of different vine structures in fitting this data set, we fitted all 480 vine structures in dimension 5 and computed their $AICs$. The vine structures can be grouped by the number of common sampling orders $nComSO$ with V_{Diss}. Recall that $nComSO$ for a 5 dimensional vine can be $0, 2, 4, \ldots, 16$. Figure 3.16 shows the box plots of the differences in AIC between all 5 dimensional regular vines and V_{Diss} with respect to $nComSO$ sampling orders in common with V_{Diss}.

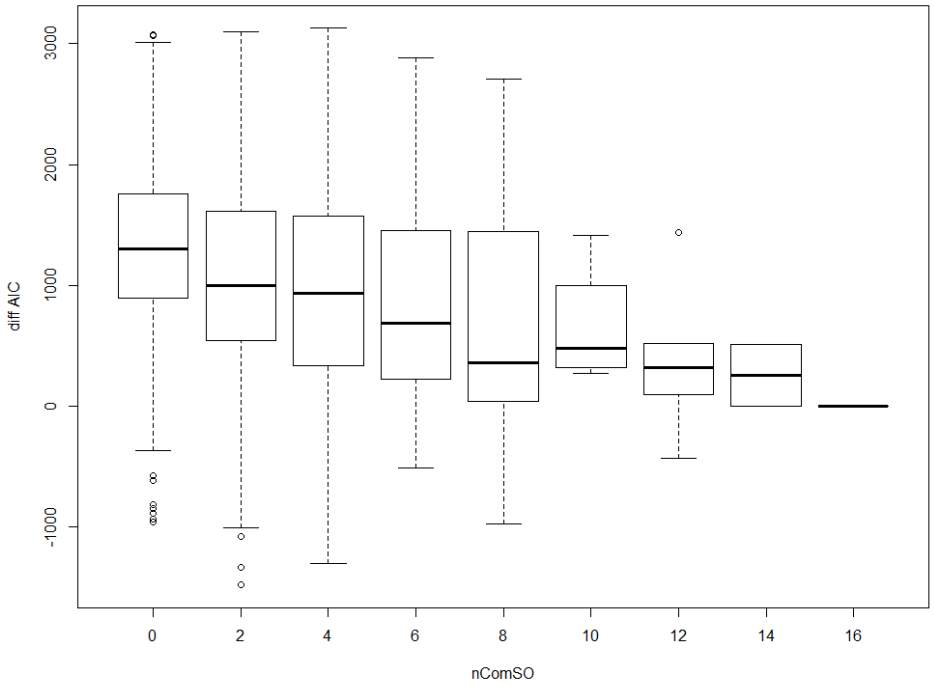

Figure 3.16: Boxplot of differences of $AICs$ of vines having different $nComSO$ common sampling orders with V_{Diss} in Table 3.2 where positive difference mean higher AIC than of V_{Diss}.

The maximum number of common sampling orders with V_{Diss} is 16 and only one vine has such many sampling orders with V_{Diss}, that is V_{Diss} itself. Thus the difference in AIC is 0. Additionally, there are 2 regular vines that have 14 sampling orders in common, 6 regular vines having 12 common sampling orders and 6 regular vines having 10

sampling orders in common with V_{Diss}. We can observe that the performance, in terms of AIC, of vines having a large number of common sampling orders with V_{Diss} is quite similar. This confirms the claim in Cooke et al. (2015) that vines having more common sampling orders will behave similarly in terms of AIC. Hence, regular vines having none or a small number of common sampling orders might behave differently in terms of AIC, and could therefore give improvements in terms of AIC, as compared to the initial structure.

Out of 208 vines that have 0 sampling orders in common with V_{Diss}, 51 have lower AIC. From these 45 have significantly lower AIC, according to the Vuong test at 5% significance level. From the 144 vines with 2 sampling orders in common with V_{Diss}, 27 have better AIC, out of which 20 have significantly better AIC.

Obviously, if the initial structure is already close to the true structure, there will not be much room for improvement. However, if the initial structure is not yet optimal, as in our example, we will be able to find better performing vine structures by considering vines with none or a small number of sampling orders in common with the initial vine. By closely examining the vine structures with low and high AIC in the subset of the vines with none or small $nComSO$ with V_{Diss}, we can identify two reasons for large AIC : i) a bad fit of copulas in lower trees that are specified in the true structure via conditional copulas (e.g., 25 with p-value 0.00977, according to the PIT test) and ii) the assumption of simplified vine structure including conditional copula, e.g., for 34|2, 25|3 and 13|2, which is rejected by using the `pacotest` at the significance level of 5%. There could be other factors that will affect the estimation.

3.4.2. HEURISTIC SEARCH OF A STRUCTURE WITH LOWER AIC THAN OF THE INITIAL VINE

As observed in Figure 3.16 vines having 0 sampling orders in common with a given structure might perform better (have smaller AIC) when compared to the initially chosen structure. Searching for a better vine in the set of vines having 0 common sampling orders with the initial vine was advocated in Cooke et al. (2015). However, we showed in Section 3.3.3, the set of vines having 0 common sampling orders with the initial structure is very large indeed. So we adapt the heuristic method in Cooke et al. (2015) and propose to search for vine structures with possible improved performance in the set of vines having 2 sampling orders in common with the initial structure. As shown in Section 3.3.3, the size of this set does not depend on the initial structure and is equal to $2 \cdot 3^{n-3} \cdot 2^{\binom{n-2}{2}}$. Moreover, these vines can be easily randomly sampled with Algorithm 8 in Appendix 3.A.2 (Note that this algorithm is only asymptotically uniform as explained in Appendix 3.A.2). We outline our proposed heuristic below.

Heuristic

To find a vine structure with a smaller AIC than that of an initial vine, generate and fit k random vines having 2 sampling orders in common with the initial vine. Choose the vine structure with the smallest AIC.

To decide how large k could be set to in our heuristic, we will investigate computational complexity of fitting and generating vines for different dimensions and data size.

We consider data sets in dimension 5, 10, 15 and 20, and evaluate sample size 300 and 1000. For each data set we obtain an initial vine according to Dißmann's algorithm in Dißmann et al. (2013), generate one random vine structure having 2 common sampling orders with it by Algorithm 8 and fit this random vine to the data. This process is repeated 100 times and the average time needed to generate a random vine structure and to fit it is reported in Table 3.3.

Sample size 300			Sample size 1000		
Dimension	*genTime*	*fitTime*	Dimension	*genTime*	*fitTime*
5	0.00053	4.7575	5	0.00051	15.5065
10	0.00328	22.6965	10	0.00331	74.8740
15	0.0744	56.1900	15	0.0790	182.9340
20	3.6771	106.5196	20	3.4869	349.7468

Table 3.3: Average generating time (*genTime*), in seconds, of one random vine having 2 sampling orders in common with the initial vine using Algorithm 8. Average fitting time (*fitTime*) of one vine, in seconds, for sample sizes of 300 and 1000.

From the results in Table 3.3, we observe that generating vines having 2 sampling orders in common with the initial structure is quite fast up to dimension 15. As compared to the estimation, the time for generating structures is negligible. However, it increases significantly in higher dimensions. This concern can certainly be answered with professional implementation which can be naturally carried out in parallel environment.

3.4.3. TESTING THE HEURISTIC SEARCH METHOD

The simulation study to test the performance of the proposed heuristic is similar to the one proposed in Kraus & Czado (2017*b*). We generate 100 times data with sample size 300 and 1000 from a randomly chosen regular vine copula model in dimensions 5, 10, 15 and 20, denoted by V_{true}. Vine structures are obtained using `RVineMatrixSample` in the **VineCopula** package. The copula families are restricted in Clayton (C) and Gumbel (G) copulas and their rotated versions. The copula parameters are obtained from Kendall's tau simulated from a *Beta(2,2)* distribution, multiplied by -1 with probability 0.5, as in the previous section.

An initial vine structure V_{Diss} is obtained by using the method in Dißmann et al. (2013)[3]. Due to computational concerns, we only choose k linearly increasing with dimension. Thus $k = 10$ regular vines that have 2 sampling orders in common with V_{Diss} in dimension 5, $k = 20$ regular vines in dimension 10, $k = 30$ in dimension 15 and $k = 40$ in dimension 20 are randomly sampled and fitted, and the AIC of each of these vines is obtained. In case we have found a structure with smaller AIC, a Vuong test is applied to see whether this improvement is significant, at the significance level of 5%. The vine with the smallest AIC, if it is significant, will be denoted by V_{min}. Furthermore, we apply an out of sample validation for our result when our heuristic finds any improvement. In

[3] In this part of the simulation study we used `vinecop` in **rvinecopulib** package where the heuristic in Dißmann et al. (2013) is implemented. In this implementation computations are much faster but smaller number of copula families are available. The *fitTime* is now approximately 8 times smaller than the results shown in Table 3.3 which were obtained on a computer with Intel Core i5-6500 3.2GHz (4 cores)

this case we generate another sample with the same size from the true vine model and calculate AIC for V_{Diss} and the vine having the smallest AIC in sample. If the out of sample AIC is still smaller than of V_{Diss}, we recognize this vine as an improvement and apply a Vuong test. In Table 3.4, the results of the simulation study are shown.

Sample size 300							
	Dim	k	$\Delta AIC(ini)$	$\#imp$	$\Delta AIC(imp)$	$\Delta AIC(sim)$	proportion
In Sample	5	10	258.17	68(54)	236.95	130.26	181.91%
	10	20	1426.35	86(77)	449.94	1039.90	43.27%
	15	30	2418.53	73(63)	543.09	1909.75	28.44%
	20	40	3655.65	67(61)	687.40	3046.90	22.56%
Out of Sample	5		388.22	63(48)	332.52	172.90	192.32%
	10		1844.74	82(73)	645.96	1270.40	50.85%
	15		2966.55	64(53)	867.62	2155.55	40.25%
	20		4434.31	61(48)	1208.32	3236.12	37.34%
Sample size 1000							
	Dim	k	$\Delta AIC(ini)$	$\#imp$	$\Delta AIC(imp)$	$\Delta AIC(sim)$	proportion
In Sample	5	10	1109.64	73(70)	832.87	537.27	155.02%
	10	20	6152.95	87(84)	1824.15	4457.53	40.92%
	15	30	10823.41	77(77)	2045.96	9059.54	22.58%
	20	40	16683.22	65(59)	2313.61	14604.09	15.84%
Out of Sample	5		1489.05	72(70)	954.49	688.00	138.73%
	10		7784.70	87(83)	2235.44	5519.68	40.50%
	15		13612.53	76(74)	2685.56	10950.72	24.52%
	20		20652.11	64(59)	3005.77	17541.24	17.14%

Table 3.4: Results of performance of the proposed heuristic for 100 data sets of size 300 and 1000, from vines in dimensions 5, 10, 15 and 20. Average difference of AIC between the initial V_{Diss} and V_{true} is denoted as $\Delta AIC(ini)$. Average difference of AIC between the V_{Diss} and the vine chosen by our heuristic V_{min} is denoted as $\Delta AIC(imp)$ and average difference of AIC between V_{min} and the true vine V_{true} structure is $\Delta AIC(sim)$. The number of times an improvement in terms of AIC with respect to V_{Diss} has been obtained is shown in column $\#imp$, additionally the number of times the improvement was statistically significant, according to Vuong test at 5% significance level is shown in brackets. The proportion is calculated as $\frac{\Delta AIC(imp)}{\Delta AIC(sim)} * 100\%$.

.

We can see that the heuristic procedure works well in improving the AIC of a given initial vine structure, V_{Diss}. Out of 100 data sets, about 70 lead to a significantly better vine copula as compared to V_{Diss}. Since our heuristic is based on randomly generated vine structures having 2 common sampling orders, the number of obtained improvements changes from simulation to simulation[4]. We compare the result of our heuristic with randomly selecting k structures out of possible vine structures in Appendix 3.A.3. That validates that our heuristic can beat the random choice.

Moreover, the average improvement in AIC is quite good. However, there is still much more room for improvement. When considering the proportion in Table 3.4, we see that this proportion becomes smaller as dimension grows (from 155.02% in dimension 5 to 15.84% in dimension 20 for 1000 sample size). By randomly choosing k vine

[4]We have repeated these simulations few times and observed that the number of improvements stays stable around 70 out of 100.

structures from vines having 2 sampling orders in common there is no guarantee that we can always choose those vines which result in an improvement in model fitting. This becomes worse when dimension becomes larger since the number of vines having 2 common sampling orders increases exponentially with dimension, whereas we allowed k to grow only linearly. More testing is needed to determine the optimal in terms of computation time and accuracy number of k.

3.5. REAL DATA ANALYSIS

To further test our heuristic method, we implement it for few real data sets, that are available and have been already analyzed in Kraus & Czado (2017*b*). We will examine whether using our heuristic method provides a structure with better performance in terms of AIC than that of the initial structure V_{Diss}. We use two non financial data sets: `uranium` Cook & Johnson (1986) and `concrete` Yeh (1998) and four financial data sets: 20 country portfolio monthly returns (from 01-1991 to 12-2017) and industry portfolio daily returns in dimension 5, 10 and 17 (all from 02-01-1997 to 31-01-2017), from $Kenneth\,R.French - Datalibrary$ [5].

We apply an $ARMA(1,1) - GARCH(1,1)$ filter, with student-t residual for the financial data sets to remove the time dependence. Then all data sets are transformed by probability integral transformation based on re-scaled empirical distribution function.

As in the simulation study, the initial structure V_{Diss} for each data set has been chosen using the heuristic in Dißmann et al. (2013). To see if we can find a better performing structure, we employ our heuristic search method.

Data	concrete	uranium	industry5	industry10	industry17	country20
Dim	4	7	5	10	17	20
Data size	1030	655	5054	5054	5054	324
k	5	10	10	20	30	40
AIC_{Diss}	-522.87	-1759.98	-23056.49	-47805.99	-89399.42	-6372.86
$\#imp$	2(0)	2(1)	1(0)	2(0)	0(0)	0(0)
ΔAIC	13.33	50.12(50.12)	18.65	29.81	$N.A.$	$N.A.$

Table 3.5: Results of performance of the proposed heuristic method for real data sets. ΔAIC denotes the largest difference in AIC of the structure obtained by our heuristic, V_{min}, as compared to V_{Diss}. $\#imp$ is the number of times the structure obtained by our heuristic leads to lower AIC than V_{Diss}. The number of significant improvements is shown in the brackets.

The following can be observed from Table 3.5:

- Our heuristic finds 2 structures that have smaller AIC than V_{Diss} both for the `uranium` and `concrete` data sets. However, only one significantly better structure is found for the `uranium` and none for the latter data according to Vuong test. It has been shown the simplifying assumption doesn't hold for the initial vine structure from these data sets in Kurz & Spanhel (2017). In Kraus & Czado (2017*b*), two algorithms have been applied to this data sets to find vine structure that is better

[5] $http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html$

than V_{Diss}. The final structure obtained in Kraus & Czado (2017*b*) has lower AIC than the one obtained by our heuristic (with ΔAIC equal to 28.7 for `concrete` data and 57.3 for `uranium` data set). We examined all (24) possible vine structures for the `concrete` data set, we found that the structure found in Kraus & Czado (2017*b*) is the best simplified vine. This structure has 4 sampling orders in common with V_{Diss} which explains why our heuristic was not able to discover this structure.

- For the financial data sets our heuristic does not lead to any (significant) improvements especially in high dimensions. It has been observed that this kind of data obtained by applying $ARMA-GARCH$ filter can be modeled well by t-copula and the simplifying assumption holds (see Kurz & Spanhel (2017)). This might be the reason why for financial data the effect of the choice of the structure is weaker and we can hardly improve the performance of V_{Diss} in terms of AIC[6]. Another reason might be our choice of k. As we mentioned before, the number of vines having 2 sampling orders in common grows exponentially as dimension becomes higher, thus we might need to have larger k in higher dimensions in order to increase the probability of finding a better structure.

3.6. CONCLUSION

In this chapter we presented theoretical background for the introduction and test of a heuristic process for structure selection of vine copula model. The idea of this method is to pick randomly and evaluate few structures which are not 'similar' to the initial structure. To measure this 'similarity', we adopt the idea in Cooke et al. (2015) to use number of common sampling orders. We build a procedure including Algorithm 5, 6 and 7 to generate all regular vines having given number of common sampling orders with an initial vine. Both in the simulation and real data study we have chosen to search for a structure performing better in terms of AIC[7] than the structure chosen by Dißmann et al. (2013), V_{Diss}, in the subset of vines having 2 common sampling orders with V_{Diss}. This choice was motivated both by theoretical and practical considerations. Our new heuristic is shown to perform well in the simulation study. Its performance, however, on the real data is not overwhelming. We can hardly make any improvement using our heuristic especially for the financial data set after filtering. This may be due to too small number of (k) random regular vines we choose, or because the initial structure is already very good. Certainly one can consider to generate more random vines having 2 common sampling orders with the initial structure but this has to be supported with more efficient implementation of generating algorithm.

[6] It is also observed that in financial data sets mostly positive correlations are present. To test if the results of our heuristic would not change if only positive dependencies were allowed in the simulation study performed in Section 3.4.3, we have conducted the simulation for 5 dimensional vines again with only one difference that the correlations are all positive sampled from a *Beta(2,2)* distribution. We showed that also in this setup about 70 out of 100 simulated vines can be improved using the heuristic.

[7] On the request of the reviewer of this chapter we have compared performance of our heuristic based on improvements of AIC with one using KL divergence between two fitted vines for 5 dimensional vines. The conclusions did not change but the differences in computation efficiency (KL divergence computed with Monte Carlo with 10^6 samples) are prohibitive to perform simulation study for of performance of our heuristic using KL distance in higher dimensions.

The choice of V_{Diss} as an initial structure is not necessary. Whatever algorithm is used to find an initial vine our heuristic can be applied to obtain possibly better performing structure for the data.

3.A. APPENDIX TO CHAPTER 3

3.A.1. PROOF

Proposition 3.A.1. *There are* $2^{\binom{n-2}{2}}$ *regular vines corresponding to one completely specified substructure in VBT.*

Proof. This can be proven by observing that a substructure gives an ordering of variables that is in fact the natural order in Nápoles (2011) where it has been proven that there are $2^{\binom{n-2}{2}}$ regular vines with this natural order.

However, similarly the proof can be shown following choices in VBT. If we only have a substructure, the largest known subvine is in dimension 3, thus we need to do $n-3$ ascend steps to fill in 0 elements in VBT following the ordering given by substructure. For each ascend step, we have to do descend steps until depth-$(n-2)$. In each descend step we have zero element which can be taken to be one of 2 choice. Thus we have 2 choice for the first extension (to extend into a 4 dimension subvine), 2^2 for the second extension and 2^{n-3} for the last extension. In total, we have $\sum_{i=1}^{n-3} 2^i = 2^{\binom{n-2}{2}}$ choices which gives the number of possible regular vines. □

3.A.2. ALGORITHMS

The algorithms used in Chapter 3 are presented.

Algorithm 4 Finding common sampling orders of two vines $V_1(n)$ and $V_2(n)$

Input: two given regular vines $V_1(n)$ and $V_2(n)$.
Output: VBT form of $V_1(n)$ compared to $V_2(n)$.

Draw VBT form of $V_1(n)$ and set $i = n$
while $i > 2$ **do**
  repeat
    For corresponding nodes of $V_1(n)$ and $V_2(n)$ in echelon-i in vine triangular array,
    - if Case1 then the indicators of both lines are 0, both elements in condition set become 0,
    - if Case2 then the indicator of the line to the child containing common variable is 0, the other variable is replaced by 0,
    - if Case3 then nothing changes in conditioned set.

  until We exhaust all nodes in echelon-i that are not set to be 0.
  if indicator of any line is set to be 0 **then**
    Reflect those changes to the corresponding elements in depth-$(n-i+1)$ in VBT by setting these elements and all elements in sub-trees with these elements as root to be 0.
    Perform Re-evaluation of vine triangular array.
  end if
  $i = i - 1$
end while

Algorithm 5 Finding all possible indicator sequences for $nComSO$ and initial vine

Input: Initial vine and $nComSO$.
Output: All possible indicator sequence.

Set indicators of all lines to 1.
Set $A_n = 2^{n-1} - nComSO$
Find the first node in highest echelon whose product is smaller than or equal to A_n.
repeat
    Find L and U using (3.3.1) and (3.3.2), respectively.
    Three possible options of assignments of 0 to lines from this node due to different conditions,

- If $nComSO = 0$ and $A_n - 2L = 0$, assign both lines 0 indicators.
- If $U > A_n$, assign neither of lines 0 indicator.
- Assign 0 indicator to one line such that Proposition 3.3.1 is satisfied and set $A_n = A_n - L$.

    Choose the first option that meets condition.
    if any 0 indicator has been assigned **then**
        Do Re-evaluation of the number of subvines in triangular array.
    end if
    if $A_n = 0$ **then**
        Store the current indicator sequence.
        Choose the next option for current node if it's possible. Otherwise go back to previous considered nodes in the recursive process and choose next possible options.
    else
        Find the next non 0 node in highest echelon whose product is smaller or equal to A_n.
    end if
until no more non 0 nodes in vine triangular array

Algorithm 6 Choice of zero elements in conditioned sets of nodes that are not 0 in vine triangular array

Input: triangular array of initial vine and indicator sequence.
Output: triangular array with 0 nodes.

$i = n$
while $i > 2$ **do**
    repeat
        if indicator of any line 0 **then**
            Set the element in the conditioned set of node to be 0
        end if
    until We exhaust all indicators in echelon-i
    Re-evaluation of vine triangular array if there is 0 indicator in this echelon
    $i = i - 1$
end while
for $j = 2$ to 1 **do**
    Start from the lowest echelon node to the highest one where (Case j) happens
    Choose variable from conditioning set for the zero element in conditioned set if this choice satisfies proximity condition
end for

Algorithm 7 Choice of nodes that are 0 in vine triangular array

Input: vine triangular array after choice for zero elements in conditioned set of non 0 nodes.

Output: vine triangular array without zero elements that represents a regular vine.

Construct VBT corresponding to input vine triangular array.
Choose or construct a substructure in VBT and set the order of vine extensions.
Start from the largest fully specified subvine in VBT.
for Each Ascend step to extend subvine **do**
    Find the nearby pair of elements, which constitutes the conditioned set of a subvine.
    if there are still 0 elements in this subvine before depth-$(n-1)$ **then**
        Do consecutive Descend steps until depth-$(n-2)$
        for Each time we do Descend step **do**
            Duplicate common sub-tree without root if there are 0 elements.
            if there is 0 in the nearby pair of elements **then**
                Choose one value for this element by property (3).
                Keep VBT to be in standard form
            end if
        end for
    end if
end for
Fill in remaining 0 elements in depth-$(n-1)$ and n by property (1,2)
Get vine triangular array from the VBT.

According to the proof in Proposition 3.3.4, we need to generate a binary sequence with length $(n-2)+(n-3)+\binom{n-2}{2}$ to sample one regular vine having 2 common sampling orders with the given initial vine structure. The first $n-2$ binary numbers are to determine which nodes are in Case2 in vine triangular array and which elements in their conditioned set are set to be 0. The middle $n-3$ binary numbers are to assign those zero elements in conditioned set of Case2 nodes according to proximity. The last $\binom{n-2}{2}$ binary numbers are to fill in 0 elements in VBT.

Algorithm 8 Randomly generating one regular vines having 2 common sampling orders

Input: initial structure

Output: one random regular vines having 2 sampling orders in common.

1: Generating one random binary sequences I with length $(n-2)+(n-3)+\binom{n-2}{2}$.
2: **for** $i = 1$ to $(n-2)$ **do**
3: from the top node to node in echelon-3,
- If $I[i]$ is 0, assign the left line 0 indicator.
- If $I[i]$ is 1, assign the right line 0 indicator.

4: **end for**
5: Set zero elements and 0 nodes in triangular array according to the indicator sequence.
6: **for** $i = (n-1)$ to $(2n-5)$ **do**
7: for non 0 nodes in echelon-4 to echelon-n, the zero element in the conditioned set becomes,
- If choice possible and $I[i]$ is 0, the zero element is chosen as the left element in the conditioned set of its child.
- If choice possible and $I[i]$ is 1, the zero element is chosen as the right element in the conditioned set of its child.
- If no choice, fill in the value for the zero element and move to next element of I.

8: **end for**
9: **for** $i = (2n-4)$ to $\binom{n-2}{2}$ **do**
10: in each descend step, the 0 element in VBT becomes,
- If $I[i]$ is 1, zero element of VBT is the left element in pair of elements.
- If $I[i]$ is 0, zero element of VBT is the right element in pair of elements.

11: After each assignment for 0 element in VBT, keep VBT in its standard form.
12: **end for**
13: Get vine triangular array from VBT.

Since we may ignore elements in the random binary sequences I in line 7 in Algorithm 8, we will get the same vine structure if only these elements are different for two binary sequences. However the number of possible ignored elements is small as compared to the total length of the binary sequence (at most $\frac{n-3}{2n-5+\binom{n-2}{2}}$) as dimension becomes larger. Thus we can regard Algorithm 8 as an asymptotically uniform sampling procedure.

3.A.3. SIMULATION RESULT BY RANDOM CHOICE

We compare performance of our heuristic with results obtained by sampling k vine structures chosen uniformly from the set of possible regular vines and present the results in Table 3.6.

Sample size 300							
	Dim	k	$\Delta AIC(ini)$	$\#imp^{rnd}$	$\Delta AIC(imp)^{rnd}$	$\Delta AIC(sim)^{rnd}$	proportion
In Sample	5	10	258.17	73(63)	194.50	154.65	125.77%
	10	20	1426.35	62(58)	440.09	1166.80	37.72%
	15	30	2418.53	55(42)	478.64	2118.51	22.59%
	20	40	3655.65	59(49)	530.41	3199.69	16.58%
Out of Sample	5	10	388.22	68(56)	268.00	209.60	127.86%
	10	20	1844.74	60(56)	599.76	1413.20	42.44%
	15	30	2966.55	52(43)	820.94	2368.12	34.67%
	20	40	4434.31	51(39)	1127.52	3413.37	33.03%
Sample size 1000							
	Dim	k	$\Delta AIC(ini)$	$\#imp^{rnd}$	$\Delta AIC(imp)^{rnd}$	$\Delta AIC(sim)^{rnd}$	proportion
In Sample	5	10	1109.64	77(72)	772.80	595.62	129.75%
	10	20	6152.95	80(78)	1646.71	4825.00	34.13%
	15	30	10823.41	67(61)	2062.20	9286.02	22.21%
	20	40	16683.22	57(56)	2343.68	15134.95	15.48%
Out of Sample	5	10	1489.05	71(67)	952.21	726.88	131.00%
	10	20	7784.70	77(74)	2046.57	6020.40	33.99%
	15	30	13612.53	63(58)	2699.06	11195.33	24.11%
	20	40	20652.11	56(53)	3243.98	17867.82	18.16%

Table 3.6: Results of performance by randomly choosing k random vines for 100 data sets of size 300 and 1000, from vines in dimensions 5, 10, 15 and 20. Average difference of AIC between the initial V_{Diss} and V_{true} is denoted as $\Delta AIC(ini)$. Average difference of AIC between the V_{Diss} and the vine by random choice having the best improvement in AIC V_{min}^{rnd} is denoted as $\Delta AIC(imp)^{rnd}$ and average difference of AIC between V_{min}^{rnd} and the true vine V_{true} structure is $\Delta AIC(sim)^{rnd}$. The number of times an improvement in terms of AIC with respect to V_{Diss} has been obtained is shown in column $\#imp^{rnd}$, additionally the number of times the improvement was statistically significant, according to Vuong test at 5% significance level is shown in brackets. The proportion is calculated as $\frac{\Delta AIC(imp)^{rnd}}{\Delta AIC(sim)^{rnd}}$

We observe (see Table 3.6, Figure 3.17 and Figure 3.18) that our heuristic behaves better than randomly choosing k vine structures in aspect of the frequency and amount of improvement in AIC. This confirms that our heuristic to search in the space of vines having 2 common sampling orders is a good choice.

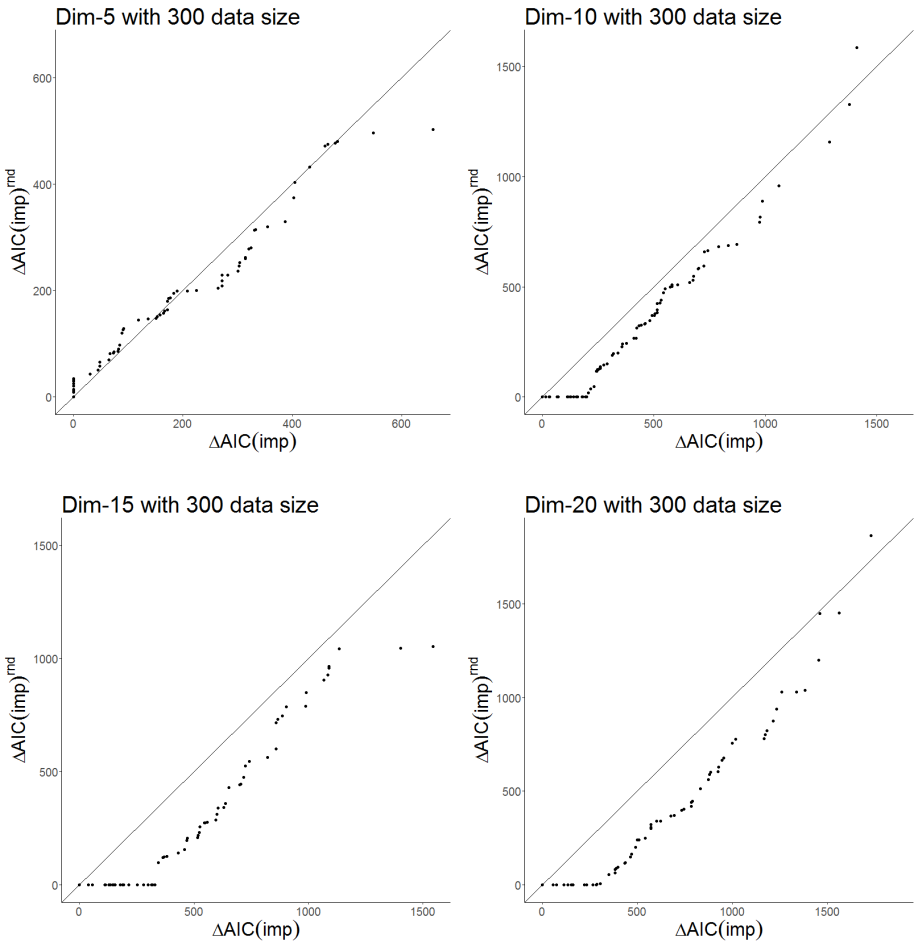

Figure 3.17: Comparison of AIC improvement to V_{Diss} in different dimensions for 300 sample size. $\Delta AIC(imp)$ means the improvement of our heuristic whereas $\Delta AIC(imp)^{rnd}$ is the improvement by random choosing vines. If no structures having improvement were found, we denote the improvement by 0.

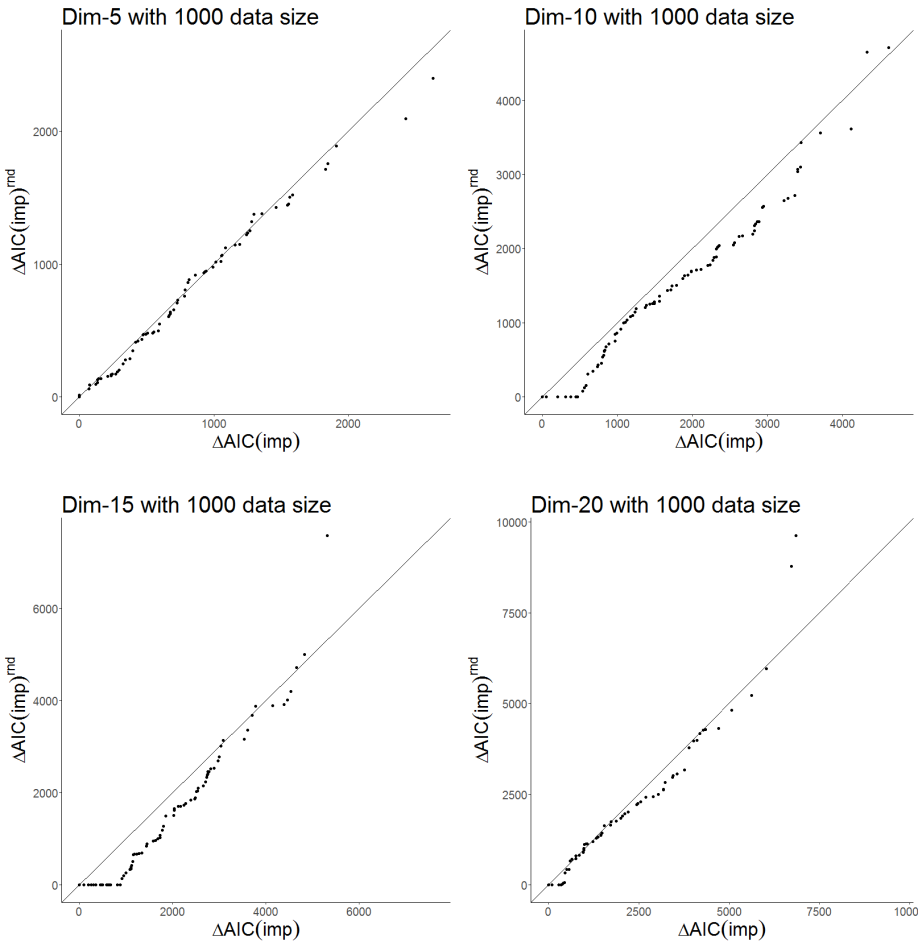

Figure 3.18: Comparison of AIC improvement to V_{Diss} in different dimensions for 1000 sample size. $\Delta AIC(imp)$ means the improvement of our heuristic whereas $\Delta AIC(imp)^{rnd}$ is the improvement by random choosing vines. If no structures having improvement were found, we denote the improvement by 0.

4

VINE COPULA BASED GENERATION OF SYNTHETIC POPULATION OF ACUTE ISCHEMIC STROKE PATIENTS

Synthetic data generation provides a way to allow broad access of patient data for researchers and public without violating the patients' privacy. Such generated data has to recover the properties of the real patients, hence distributions and statistics like mean, variance, range etc. of patients' characteristics as well as dependencies among them should be well represented. Various methods have been applied in the literature for this purpose. The vine copula model is used to fit the real patient data and to generate synthetic cohorts of patients. Furthermore, the vine copula model is compared with a popular approach in synthetic data generation, called the fully conditional specification. The data set of acute ischemic stroke patients from the MR CLEAN Registry collected in the INSIST project is used to perform the comparison of these models. We show, that on this data set, the vine copula model outperforms the fully conditional specification approach.

Parts of this chapter have been published in Zhu et al. (n.d.).

4.1. INTRODUCTION

There is a growing interest in the use of synthetic patient data within health care systems. Since the introduction of General Data Protection Regulation (Goodman & Flaxman (2017)) the access to real patient data sets is hindered by legal, privacy and intellectual property restrictions. The use of synthetic patient populations allows researchers, industry, policy makers to get access to large patient cohorts free of protected health information and personally identifiable information constraints. However, these synthetic populations should capture all distributions and dependencies observed in real-patient populations (Walonoski & et al. (2018), Buczak & Gifford (2010)).

In this study, we used data from the MR CLEAN Registry, the largest nationwide Registry of ischemic stroke patients in the Netherlands to generate synthetic cohorts. Stroke is the second leading cause of disability and death worldwide, affecting over 12.2 million patients each year (GBD 2019 Stroke Collaborators (2021)). Approximately 85% of all strokes are ischemic strokes, caused by a thrombus occluding an intracranial artery, in turn causing a sudden interruption of blood flow and therefore brain ischemia. Acute treatment of ischemic stroke is vital, since every hour of delay in treatment is associated with a $3-5\%$ decreased probability of achieving functional independence (Saver & et al. (2016), Mulder & et al. (2018)).

The data consists of 15 continuous and discrete variables, including continuous parameters: age, baseline National Institutes of Health Stroke Scale (NIHSS)[1], systolic blood pressure on hospital admission, diastolic blood pressure on hospital admission, time between stroke onset and arrival at the first hospital, time between arrival at the first hospital and start of EVT; dichotomous parameters: medical history of ischemic stroke, diabetes mellitus, and atrial fibrillation, presence of hyperdense artery sign on baseline NCCT; ordinal parameters: pre-stroke modified Ranking Scale score (mRS), Alberta Stroke Program Early Tomography Score at baseline (ASPECTS), location of occlusion, and collateral score.

The synthetic stroke population will be integrated in the in silico framework as part of the INSIST project. INSIST-In Silico trials for treatment of acute ischemic stroke, www.insist-h2020.eu) is a European project which aims to develop a platform that enables the execution of in silico trials for acute ischemic stroke. The proposed in silico trial platform aims to be a proof-of-concept to assess the extent to which in silico modeling can accurately simulate stroke treatment and estimate outcome for a synthetic cohort of ischemic stroke patients.

There are several methods in the literature to generate a synthetic population of patients. In Teutonico & et al. (2015) the authors propose a bootstrap procedure to randomize the real patient data. Using this method, however, one is not able to generate any combinations of patients' characteristics that have not been observed in the data. A parametric approach introduced in Tannenbaum & et al. (2006) requires fitting a multivariate Gaussian distribution to the data. Similarly, in Tucker et al. (2020) a Gaussian Bayesian network is constructed based on the observed patients data. Both these methods might not represent data well when Gaussian assumption is not satisfied. Moreover they can lead to unrealistic samples in simulated population of synthetic patients.

[1] NIHSS is not a continuous variable (having 43 categories) but it will be treated as such in this chapter.

Rather than specifying a parametric joint distribution, the authors in Smania & Jonsson (2021) propose to estimate a sequence of conditional distributions. This method was originally developed to perform missing data imputation introduced in van Buuren (2012), and is referred to as fully conditional specification. The comparison study therein shows that when the assumption of multivariate Gaussian distribution does not hold the fully conditional specification approach works better in capturing properties of the data.

In this chapter we consider a vine copula based approach to estimate the joint distribution. The vine structure is chosen based on the method in Chapter 3. Furthermore, we will compare the vine copula model with the fully conditional specification method to find out which one is better in generating synthetic data of acute ischemic stroke patients.

The chapter is organized as follows. In Section 4.2 the models used to generate synthetic data are shortly introduced and compared, and the performance measures used for comparison are presented. In Section 4.3 the data set is introduced. The performance of the compared models on the real patient data is shown in Section 4.4, and finally conclusions are included in Section 4.5.

4.2. MODELS

Simulating virtual patients' population requires sampling from a joint distribution of variables $\mathbf{X} = (X_1, \ldots, X_n)$. This joint distribution can be specified in different ways and in this chapter, we will compare two approaches: 1) through a sequence of conditional distributions of $X_i|X_{-i}$, where X_{-i} denotes the conditional distribution of all variables except X_i; 2) through estimating each marginal distributions $F_i, i = 1, \ldots, n$ and a vine copula representing the dependence among variables.

In this section we shortly introduce two methods based on conditional distributions: a fully conditional specification (FCS), which is the method of multiple imputation by chained equations, and a conditional distribution method, denoted as CA, which is a special case of the FCS satisfying assumptions of monotone missing data pattern. We also introduce a copula based approach called vine copula (VC).

4.2.1. FULLY CONDITIONAL SPECIFICATION

Different models can be used to specify the conditional distributions [2]. In this chapter, we present the model based on multiple imputation by chained equations (van Buuren (2012), Gravesteijn & et al. (2021)), referred to as FCS. This model is constructed by successive specification of the conditional distributions in the tth iteration,

$$F^{(t)}(x_i^{mis,t}|X_1^{(t)}, X_2^{(t)}, \ldots, x_i^{obs}, X_{i+1}^{(t-1)}, \ldots, X_n^{(t-1)}),$$

where x_i^{mis} and x_i^{obs} are the set of missing and observed values for the ith variable respectively, and the conditional distributions are estimated following the imputation order from X_1 to X_n. Since the data, which needs to be simulated, can be regarded as missing then FCS can be applied for synthetic data generation.

[2] Note that the specifications have to be consistent. In general, however, there is no guarantee that such a joint distribution with sequentially specified conditional distributions exists.

The FCS method is easy to implement and it allows different regression type methods to specify the conditional distributions. Some disadvantages and model setting in this method are listed below (also discussed in van Buuren (2012), Li et al. (2012)):

- *Mis-specification of the conditional distributions.* For large number of variables it is not possible to specify each conditional distribution in detail. In this chapter we restrict the conditional distributions in the FCS model to be as follows: for continuous variables Bayesian linear regression is chosen, for binary variables the logistic regression is used and for ordinal data polytomous regression is taken. FCS method is implemented in the **mice** package.

- *Slow convergence.* A large number of iterations, t, might be needed for the good performance of this method when large number of missing data is presented. In van Buuren (2012), the authors concluded that the rather small number of iterations is enough to get good results and proposed to use as default 5 iterations in the **mice** package). In the synthetic data generation, because the number of simulated data is usually larger than the observed data the issue of slow convergence may be present. We follow suggestions in Samuels & et al. (n.d.) and increase the number of iterations to $t = 20$ in this chapter.

- *Order of conditional distributions.* The order of specified conditional distributions matters (see, e.g., in Goncalves & et al. (2020)). The default order in the **mice** package is from the left to the right of the input data in a dataframe. We will consider several different orders when data is analyzed and choose the best one.

In synthetic data generation, since we are generating new data, it then follows monotone missing data pattern (Drechsler (2011)). In this case the FCS method can be simplified by estimating the conditional distributions of variable $i+1$ conditional only on the variables estimated earlier in the order, which is denoted as $F_{i+1|1,\dots,i}$. Moreover, in this setup, one iteration is sufficient for the model. The conditional distributions for different types of variables are specified the same way as in the FCS method.

4.2.2. VINE COPULA APPROACH

As mentioned in Section 1.2.3 in Chapter 1, the estimation of VC model requires to determine 1) the vine structure (graph); 2) the copula families for each bivariate copula; 3) the copula parameters. For n variables, there are $\frac{n!}{2}2^{\binom{n-2}{2}}$ vine structures (Nápoles (2010)), hence it is not possible to estimate all of them and choose the best one. In practice a heuristic introduced in Dißmann et al. (2013) is used, which suggests to construct a vine structure that captures most dependence in lower trees of the vine. The copula families along with their parameters are selected from a set of copula families according to a given criterion (in this chapter we consider information metric AIC (Akaike (1998))). This approach is implemented in the **rvinecopulib** package (Nagler & Vatter (2021)).

VC has been shown to be a flexible tool to capture dependence in high dimensions. This model also allows an efficient simulation process to generate synthetic data, but one should be aware of some issues that can affect the performance of the VC model:

- *Choice of vine structures.* The number of possible vine structures grows exponentially with dimension. In practice the heuristic in Dißmann et al. (2013) is applied and works well in general, but there is no guarantee that it will lead to the best model. In Zhu et al. (2020) the authors suggest to search for several extra vine structures having 2 sampling orders in common with the initial vine structure. This allows to search for a better model that is sufficiently different from the initial choice. In this chapter we follow this approach to select the vine structure for the VC model.

- *Choice of bivariate copula families.* In this chapter we use parametric copula families implemented in the **rvinecopulib** package: Gaussian, Student-t, Clayton, Gumbel, Frank, Joe, BB1, BB6, BB7, BB8 copulas and their rotated versions. A nonparametric approach is also considered (see e.g Nagler & Czado (2016)).

- *The simplifying assumption.* In practice to make estimation efficient it is assumed that the bivariate conditional copulas in the vine decomposition (assigned to higher level trees in the vine graph) do not directly depend on the conditioning variables (a detailed discussion about this assumption can be found in Stöber et al. (2013), Spanhel & Kurz (2015), Kurz & Spanhel (2017)). This assumption might not be satisfied by the data. In this chapter we only consider simplified VC model.

4.2.3. DATA PREPARATION AND PERFORMANCE MEASURES

To fit the VC, FCS and CA models, data needs to be pre-prepared. The marginal cumulative distribution function (cdf) are estimated by the empirical cumulative distribution function (edf), defined as $\hat{F}_i(x_i) = \frac{1}{N+1}\sum_{m=1}^{N} \mathbb{1}_{x_i^m \leq x_i}$ for variable X_i on the observations $x_i^m, m = 1,\ldots,N$. Then the data is transformed into pseudo observations by PIT. The VC model is estimated based on the pseudo observations. The conditional distributions in the FCS and the CA models are estimated through a regression method, hence the pseudo observations of each variable are transformed further into z-scale before estimating FCS and CA models.

To compare the performance of the three models the following measures will be used:

- *Univariate margins.* The absolute difference between the chosen statistics computed from the observed data and those from the generated samples from a model. The chosen statistics are: the mean, the standard deviation (sd), the range, the 5% 50% and 95% quantiles of each continuous variable. For discrete variables the χ^2 test statistic is computed as follows, $\chi^2 = \sum_i \frac{(O_i - E_i)^2}{E_i}$ where for category i, O_i is the observed frequency from the generated data and E_i is the expected frequency from the observed data.

- *Marginal and joint distribution.* The mean absolute error between the marginal (joint) cdf of the simulated data and the cdf of the observations, evaluated on N observations $(\boldsymbol{x}^m, m = 1,...,N)$. This measure can be defined for any subset of variables. Let $A \subset \{1,\ldots,n\}$, and F_A^{syn} and F_A^{obs} are the cdf of the simulated and of the observed data over variables in the set A, respectively. The MAE_A for the variables in set A is denoted as,

$$MAE_A = \frac{1}{N}\sum_{m=1}^{N} |F_A^{syn}(\boldsymbol{x}_A^m) - F_A^{obs}(\boldsymbol{x}_A^m))|. \tag{4.2.1}$$

The cdf F_A^{syn} and F_A^{obs} are estimated by empirical distributions. For dimension larger than one, the edf of variables X_A is estimated by using the function `F.n` in the **copula** package (Hofert et al. (2020)).

In this chapter we will generate 10 synthetic data sets for each estimated model, compute the performance measures listed above for each generated data set and present the average results (Rubin (1987)).

4

4.3. DATA DESCRIPTION

The data used in this chapter is from the MR CLEAN Registry, a prospectively collected database including all patients treated with EVT in the Netherlands. More information about the data set can be found in Jansen et al. (2018), Compagne & et al. (n.d.). We selected patients' and imaging characteristics obtained before the start of EVT that are expected to interact with stroke treatment or to predict outcome after stroke, based on expert opinion. This resulted in a set of 15 mixed variables. These characteristics are required as input for the in silico stroke treatment models developed and validated within the INSIST project (Venema & et al. (2017)).

- Continuous variables [3],
  Age: age of the patient in years;
  Systolic BP: systolic blood pressure measured on hospital admission (emergency department) in mmHg (millimeters of mercury);
  Diastolic BP: diastolic blood pressure measured on hospital admission (emergency department) in mmHg;
  NIHSS: National Institutes of Health stroke scale (NIHSS, indicating stroke severity based on scoring neurologic symptoms) on hospital admission, ranging from 0 = no symptoms to 42 = severe neurologic deficit); [4]
  Time to ER: time between stroke onset and arrival hospital (in minutes);
  Time ER groin: time between arrival hospital and start of EVT (in minutes).

- Dichotomous variables,
  Sex: sex of the patient (Male:M /Female:F);
  Prev stroke: Medical history of stroke (Yes:Y /No:N);
  Prev diabetes: Medical history of diabetes mellitus (Yes:Y /No:N);
  Prev fibril: Medical history of atrial fibrillation (Yes:Y /No:N);
  HAS: Presence of hyperdense artery sign on non-contrast computed tomography scan performed on hospital admission (Yes:Y /No:N).

[3] Variables *Systolic BP, Diastolic BP, NIHSS, Time to ER, Time ER groin* include only integer values hence they are in principle not continuous variables. In this chapter we treat them as continuous variables as in Samuels & et al. (n.d.).

[4] This variable has only finite number of levels but will be treated as continuous in our analysis.

- Ordinal variables,
  Premrs: Modified Rankin Scale (mRS) score indicating the level of functional dependence in daily life before the stroke happened, assessed on hospital admission (ranging from 0, no symptoms to 5, severe disability) (6 levels);
  Collaterals: collateral status assessed with computed tomography angiography scan performed on hospital admission. The collateral circulation refers to the subsidiary network of vascular channels that stabilize cerebral blood flow when principal conduits fail (alternative route to provide blood to the affected brain region). The score ranges from 0 = absent collaterals [unfavorable], 1 = filling < 50% of occluded area, 2 = filling > 50% but less < 100% of occluded area, to 3 = filling 100% of occluded area [favorable] (4 levels);
  Occlusion loc: Intracranial artery occlusion location causing the stroke on computed tomography angiography scan performed on hospital admission (listed from proximal= in a vessel closer to the heart, most often a larger brain vessel to distal= further away from the heart, most often a smaller brain vessel: 1 = Other:M3,A1,A2; 2 = intracranial ICA; 3 = ICA-T; 4 = M1; 5 = M2;) (5 levels);
  ASPECTS: Alberta Stroke Program Early Tomography Score (ASPECTS) at baseline (hospital admission) non-contrast CT scan (ranging from 10 = no early ischemic changes in the brain [favorable] to 0 = early ischemic changes in all 10 brain regions [unfavorable]; 1 point is subtracted from the score of 10 for early ischemic change for each of the defined regions) (11 levels).

This patient data set consists of 3082 data points and the detailed description about the data set can be found in Appendix 4.A.1.

In the next section, we will show how the synthetic data is generated by the VC, FCS and CA models, and a comparison of performances for these methods is presented.

4.4. ANALYSIS

In this section we apply the VC, the FCS and the CA methods to the acute ischemic stroke patients data to see which method has better performance in synthetic data generation.

We divide the observations randomly into an evaluation data and a test data set. The evaluation data consists of two thirds (2055) of data points and the test data contains 1027 data points. The synthetic data size is chosen to be 10000 and the generation process is repeated 10 times.

We show, in Section 4.4.1, at first how the three models are estimated. Then the comparison of the results for the marginal and the joint distributions of the variables is performed based on the measures in Section 4.2.3. The MAE is calculated between the marginal (joint) cdf of the observations (the evaluation data or the test data) and the cdf of the generated data, evaluated on the observations. In Section 4.4.2 we select four most important variables (according to experts) and present a detailed comparison study of the three models based on these four variables.

4.4.1. MARGINAL AND JOINT DISTRIBUTIONS

As discussed in Section 4.2 the continuous variables are at first transformed into pseudo observations using PIT, and the cdf of each margin is estimated by the empirical distribution. The pseudo observations are then further transformed into their z-scale in order to satisfy the Gaussian assumption required by both FCS and CA methods.

The estimation of the VC model requires to specify a vine structure. For 15 dimensional data set it is not possible to estimate all structures. Hence, the vine structure for the data is chosen by evaluating an initial vine constructed by the heuristic in Dißmann et al. (2013) and then examining 50 extra vines having 2 sampling orders in common with the initial structure (as explained in Chapter 3). The VC model is then chosen to be the best performing structure. The detailed discussion of how to choose a proper vine structure for the VC model is presented in Appendix 4.A.2.

The FCS and the CA model require a choice of the order of variables. There are 15! orders, hence as in the case of vine structures, it is not possible to verify all of them. One option is to order variables according to their medical importance as judged by experts, which in this study has been: (*Age, Occlusion loc, NIHSS, ASPECTS, Premrs, Collaterals, Prev stroke, Prev diabetes, Prev fibril, Sex, HAS, Systolic BP, Diastolic BP, Time to ER, Time ER groin*). We can further randomly generate extra 20 orders [5] for the FCS and the CA model and pick one with the best performance. The best order for the FCS model is different than the best order for the CA model. For the FCS model the order is (*Prev fibril, Sex, Prev stroke, Premrs, Time ER groin, NIHSS, Time to ER, ASPECTS, Collaterals, Prev diabetes, Age, Systolic BP, HAS, Diastolic BP, Occlusion loc*), whereas for the CA model it is (*Age, Time to ER, Systolic BP, Prev diabetes, Prev stroke, Prev fibril, HAS, Diastolic BP, Premrs, NIHSS, Sex, ASPECTS, Occlusion loc, Collaterals, Time ER groin*).

The MAE (averaged MAE for the 10 simulated data sets) for the joint cdf for each of the three methods is shown in Table 4.1 for the evaluation data set and the test data set.

	eval			test		
	VC	FCS	CA	VC	FCS	CA
$100 \times MAE$	0.0807	0.0781	**0.0774**	0.1191	**0.1156**	0.1162

Table 4.1: The MAE (multiplied by 100) between the joint cdf of the observations (evaluation data (eval) and test data (test)) and the cdf of the synthetic data generated with VC, FCS and CA model, respectively.

The differences in MAE are quite small for all the three methods. For the evaluation data set the CA method performs the best, while for the test data set the FCS method has the best performance. The larger MAE of the VC model can be caused by the lack of fit of the parametric bivariate copula families available in the software or the simplifying assumption we used when estimating the VC model.

Although the VC model does not outperform the other two methods according to the MAE when all variables are considered, it might still perform better for some subsets of variables. In Table 4.11 in the Appendix 4.A.3, MAE for 1-dimensional margins is presented. In Table 4.12 and Table 4.13 in the Appendix 4.A.3, mean absolute difference between the chosen statistics discussed in Section 4.2.3 is presented. We can observe

[5] Note that the number of possible vine structures is larger than the number of possible orders. Hence we decided to examine 20 orders (which is comparable to the 50 vine structures we searched for the VC model.

that the VC model outperforms the other two models in estimating the univariate distributions of all continuous variables according to both the MAE and the chosen statistics. Furthermore, VC model gives better results than the other two methods in modeling tails of distributions (high and low quantiles) for almost all the continuous variables (exceptions are, in the evaluation data set for variables *Time ER groin* and *NIHSS*, the CA method has the best performance at 5% quantile; in the test data for *Systolic BP* the FCS is the best at 5% quantile and for *Diastolic BP* the FCS model outperforms other two at 95% quantile). Since margins are handled separately from the dependence in VC model, its performance on the univariate margins is very good.

For higher dimensional distributions, due to computational complexity, the analysis is performed only in dimension two and three. In Table 4.2 the results indicates the number of times a given method outperforms the others according to MAE.

eval										test								
Dim		#OPF			Dim		#OPF			Dim		#OPF			Dim		#OPF		
		VC	FCS	CA			VC	FCS	CA			VC	FCS	CA			VC	FCS	CA
2	VC		13	9	3	VC		35	39	2	VC		1	0	3	VC		11	4
	FCS	92		66		FCS	420		328		FCS	104		63		FCS	444		308
	CA	96	39			CA	416	127			CA	105	42			CA	451	147	

Table 4.2: Result of MAE for high dimensional marginal distributions, where #OPF - the number of times the model in the column outperforms the model in the row; eval - the evaluation data set and test - the test data set.

The results presented in Table 4.2 reveal that the VC model has the best performance in most of the bivariate and trivariate marginal distributions. It is not possible to analyze relationships between all variables in detail, hence in the next section we concentrate on four selected variables.

4.4.2. PERFORMANCE OF THE METHODS ON THE SELECTED VARIABLES

This section presents a detailed analysis of the VC, FCS and CA methods in generating samples from the four important variables (as judged by experts): X_1 = *Age*, X_2 = *NIHSS*, X_3 = *Occlusion loc* and X_4 = *Collaterals*. First MAE for all subsets of these variables is presented in Table 4.3.

Except for the marginal distributions of variables X_3 (*Occlusion loc*) and X_4 (*Collaterals*) and their joint distribution, the VC model is always the best model according to the MAE both on the evaluation and the test data. The MAE of the VC method for X_3 and X_4 is larger than the MAE of the FCS and CA methods on the evaluation data but smaller on the test data. This can indicate the tendency of overfitting for the FCS and the CA methods.

Next in Table 4.4 we examine the performance of all three methods in realizing the chosen statistics on the evaluation and the test data.

The performance of all the three models on the statistics of variable X_1 (*Age*) and X_2 (*NIHSS*) is quite comparable. This result coincides with the conclusions from the MAE of the 1-dimensional marginal distribution of continuous variables. For the discrete variables, the VC model works better than the others according to the χ^2 test statistic. This is

		$100 \times MAE$									
		F_1	F_2	F_3	F_4	F_{12}	F_{13}	F_{14}	F_{23}	F_{24}	F_{34}
eval	VC	**0.3259**	**0.3393**	0.4491	0.3816	**0.3680**	**0.4449**	**0.3072**	**0.4012**	**0.4229**	0.4537
	FCS	0.5614	0.8289	**0.2794**	**0.3260**	0.6454	0.5285	0.5812	0.7056	0.7423	**0.3242**
	CA	0.6645	0.7897	0.4538	0.3403	0.6637	0.6049	0.5653	0.6485	0.6120	0.5322
		F_{123}	F_{124}	F_{134}	F_{234}	F_{1234}					
	VC	**0.3708**	**0.3694**	**0.3619**	**0.4016**	**0.3331**					
	FCS	0.5922	0.6040	0.5179	0.6677	0.5543					
	CA	0.5464	0.5406	0.5363	0.5587	0.4841					
		F_1	F_2	F_3	F_4	F_{12}	F_{13}	F_{14}	F_{23}	F_{24}	F_{34}
test	VC	**0.3267**	**0.3395**	**0.5068**	**1.0899**	**0.4440**	**0.5183**	**0.7716**	**0.4400**	**0.5287**	**0.9769**
	FCS	0.5699	0.8190	0.8472	1.5548	0.6950	0.7451	1.2687	0.7334	1.0846	1.2083
	CA	0.6662	0.7941	0.9119	1.4656	0.7441	0.8335	1.1609	0.7539	0.8133	1.2694
		F_{123}	F_{124}	F_{134}	F_{234}	F_{1234}					
	VC	**0.4523**	**0.4909**	**0.6646**	**0.5092**	**0.4271**					
	FCS	0.6791	0.8053	1.0221	0.8802	0.6565					
	CA	0.6852	0.6944	0.9987	0.7356	0.5953					

Table 4.3: MAE (the MAE is multiplied by 100) for marginal and joint cdfs for the selected variables, X_1 = *Age*, X_2 = *NIHSS*, X_3 = *Occlusion loc* and X_4 = *Collaterals*.

eval			mean	sd	range	5%	50%	95%			χ^2
	X_1	VC	**0.1154**	0.1091	**0.1982**	**0.2710**	**0.1525**	**0.1248**	X_3	VC	**14.6297**
		FCS	0.2145	0.1873	0.2050	0.4653	0.2128	0.3622		FCS	34.7652
		CA	0.3130	**0.0972**	0.2724	0.5043	0.3822	0.2571		CA	18.3822
	X_2	VC	**0.0528**	**0.0329**	0.1000	0.3000	0.1000	**0.1000**	X_4	VC	6.1441
		FCS	0.1562	0.1010	0.1000	0.3000	**0**	**0.1000**		FCS	7.3615
		CA	0.1617	0.1234	**0**	**0.2000**	0.4000	0.3000		CA	**5.3883**
test			mean	sd	range	5%	50%	95%			χ^2
	X_1	VC	**0.1126**	0.1221	0.5303	**0.2597**	**0.1486**	**0.1772**	X_3	VC	**22.4813**
		FCS	0.2169	0.1659	0.9564	0.3539	0.1727	0.6730		FCS	56.7458
		CA	0.3088	**0.0866**	**0.4322**	0.5193	0.3425	0.3494		CA	52.5757
	X_2	VC	**0.0519**	**0.0368**	**0**	0.1000	0.1000	**0**	X_4	VC	**35.9456**
		FCS	0.1478	0.0862	0.2000	0.1000	**0**	0.1000		FCS	203.2500
		CA	0.1574	0.1052	0.1000	**0**	0.3000	0.3000		CA	73.5746

Table 4.4: Mean absolute difference between the observations (evaluation (eval) and test data set) and the synthetic data of the statistics (mean, sd, range, 5%, 50% and 95% quantile for continuous variables and the χ^2 test statistic for the discrete variables.) for the VC, FCS and CA methods on the selected variables, X_1 = *Age*, X_2 = *NIHSS*, X_3 = *Occlusion loc* and X_4 = *Collaterals*.

in contrast to the results obtained by MAE, which can be explained on the distributions of these variables X_3 (*Occlusion loc*) and X_4 (*Collaterals*) shown in Table 4.5.

		level-1	level-2	level-3	level-4	level-5			level-1	level-2	level-3	level-4
X_3	Obs_{eval}	0.0068	0.0564	0.2151	0.5742	0.1474	X_4	Obs_{eval}	0.0608	0.3572	0.3830	0.1990
	Obs_{test}	0.0097	0.0380	0.2132	0.5852	0.1538		Obs_{test}	0.0604	0.3710	0.3934	0.1753
	VC	0.0075	0.0495	0.2140	0.5781	0.1510		VC	0.0597	0.3605	0.3878	0.1920
	FCS	0.0113	0.0564	0.2141	0.5696	0.1486		FCS	0.0650	0.3521	0.3823	0.2006
	CA	0.0094	0.0585	0.2160	0.5661	0.1499		CA	0.0642	0.3549	0.3806	0.2003

Table 4.5: The frequency table of variable X_3 (*Occlusion loc*) and X_4 (*Collaterals*) for the observations (Obs_{eval} on the evaluation data, Obs_{test} on the test data), the stacked simulated data from the VC, FCS and CA models.

We can see that the VC model recovers the best the frequencies of levels with small probabilities, e.g. level-1 of X_3 (*Occlusion loc*) and level-1 of variable X_4 (*Collaterals*). For some levels with larger frequencies, e.g. level-2 of X_3 and level-4 of X_4, the difference between the observed and simulated frequencies of the FCS and CA models is smaller. Since MAE is a measure on the absolute error whereas χ^2 test statistic is a measure on the relative error, one can explain a smaller MAE for the FCS model (or the CA model) than the one for the VC model, but a larger χ^2 test statistic.

To explain the better performance of the VC model on bivariate marginal distributions, we show the quantile-quantile plot (Q-Q plot) of X_1 (*Age*) given that X_4 (*Collaterals*) is in level-4 (category "100% of occluded area") in Figure 4.1 (left) and the Q-Q plot of X_2 (*NIHSS*) given X_3 (*Occlusion loc*) is in level-5 (category "M2") in Figure 4.1 (right) on the evaluation data. These two figures are quite representative to explain why the VC model can capture the dependence between the variables well.

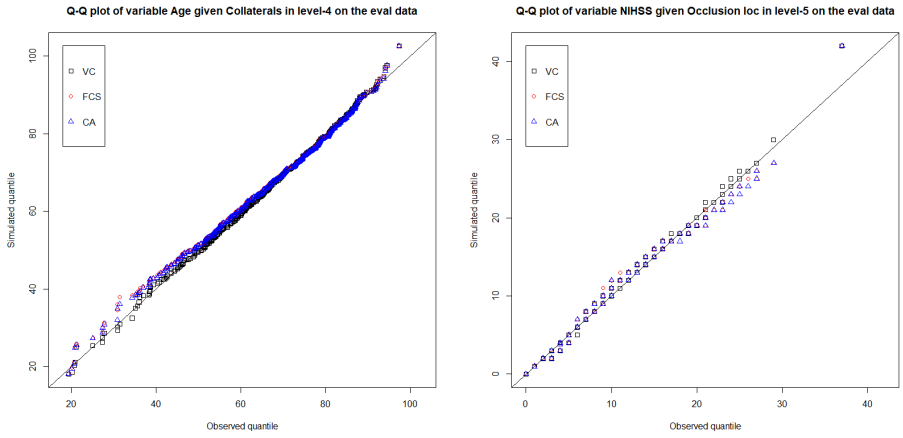

Figure 4.1: Q-Q plot (left) for the variable X_1 (*Age*) given that X_4 (*Collaterals*) is in level-4 (category "100% of occluded area"); Q-Q plot (right) for the variable X_2 (*NIHSS*) given that X_3 (*Occlusion loc*) is in level-5 (category "M2"). Simulated quantile for the VC model is shown by black square, for the FCS model it is shown by red circle and for the CA model it is shown by blue triangle.

For the bivariate distributions of the discrete variables, we explain the performance of the three methods by their contingency table in Table 4.6.

	Obs_{eval}; Obs_{test}; VC; FCS; CA	
X_3 \ X_4	level-1	level-2
level-1	0.0000; 0.0000; 0.0005; 0.0002; 0.00002	0.0010; 0.0010; 0.0032; 0.0019; 0.0013
level-2	0.0010; 0.0010; 0.0045; 0.0012; 0.0012	0.0165; 0.0088; 0.0197; 0.0162; 0.0167
level-3	0.0229; 0.0253; 0.0156; 0.0232; 0.0240	0.0929; 0.0993; 0.0849; 0.0925; 0.0931
level-4	0.0311; 0.0331; 0.0327; 0.0340; 0.0323	0.2039; 0.2152; 0.2058; 0.1979; 0.2001
level-5	0.0058; 0.0010; 0.0064; 0.0064; 0.0067	0.0428; 0.0467; 0.0469; 0.0435; 0.0438
X_3 \ X_4	level-3	level-4
level-1	0.0015; 0.0010; 0.0027; 0.0030; 0.0018	0.0044; 0.0078; 0.0011; 0.0062; 0.0063
level-2	0.0204; 0.0156; 0.0174; 0.0210; 0.0213	0.0185; 0.0127; 0.0081; 0.0180; 0.0194
level-3	0.0725; 0.0701; 0.0787; 0.0717; 0.0722	0.0268; 0.0185; 0.0348; 0.0267; 0.0267
level-4	0.2229; 0.2288; 0.2270; 0.2215; 0.2181	0.1163; 0.1081; 0.1126; 0.1162; 0.1157
level-5	0.0657; 0.0779; 0.0622; 0.0652; 0.0672	0.0331; 0.0282; 0.0355; 0.0336; 0.0323

Table 4.6: The contingency table of variable X_3 (*Occlusion loc*) and X_4 (*Collaterals*) for the observations (Obs_{eval} on the evaluation data, Obs_{test} on the test data), the stacked simulated data from the VC, FCS and CA models.

We see that for the evaluation data, the VC model on the bivariate distribution of X_3 (*Occlusion loc*) and X_4 (*Collaterals*) is not so good. This is not surprising in view of the worse performance of this method on the univariate margins of X_3 and X_4. For example, when the level of X_3 is in level-2 (category "Intracranial ICA") and the level of X_4 is in level-4 (category "100% of occluded area"), the joint frequency for the VC model is only 0.0081 (compared to 0.0185 for the observed data; 0.0180 for the FCS model; 0.0194 for the CA model). However, on the test data set the frequency in that region of the joint distribution is 0.0127, and then the frequency of the data from the FCS and CA model is not as close to the observed frequency.

Finally we consider the trivariate distribution, F_{134} and F_{234}. We select the Q-Q plot of the conditional distribution of X_1 (*Age*) given that X_3 (*Occlusion loc*) is in level-4 (category "M1") and X_4 (*Collaterals*) is in level-4 (category "100% of occluded area") in Figure 4.2 (left) and the Q-Q plot of X_2 given that X_3 is in the level-5 (category "M2") and X_4 in level-3 (category "> 50% but less < 100%") in Figure 4.2 (right).

The region in F_{34} where X_3 (*Occlusion loc*) is in level-2 (category "Intracranial ICA") and X_4 (*Collaterals*) is in level-4 (category "100% of occluded area") contains very little data points for variable X_1 (*Age*). Although the VC model has the worst performance in that region, its contribution to MAE of the joint distribution of F_{134} and F_{234} is small. This is confirmed graphically in the Q-Q plots in Figure 4.2. However, if we look at the conditional distribution in Figure 4.2 (right), even though the VC model has the best performance it can generate values which are higher than the maximum of the observation of X_2 (*NIHSS*) given X_3 and X_4. This indicates possible mis-specification of the tail dependence in the VC model.

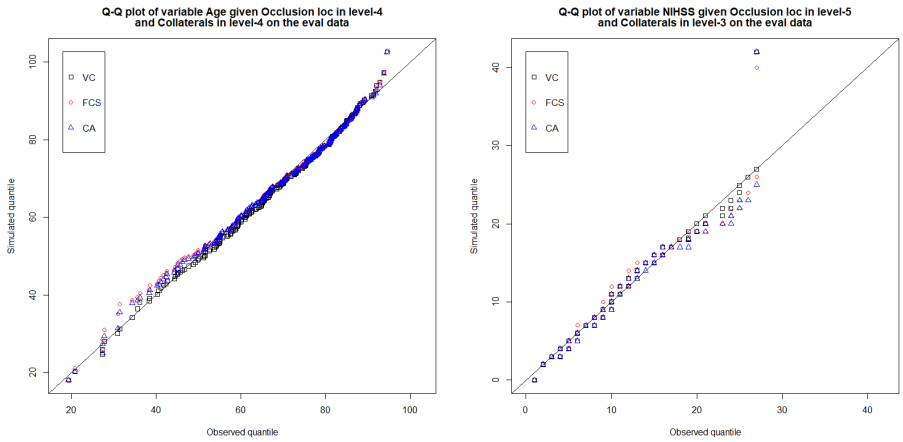

Figure 4.2: Q-Q plot (left) for the variable X_1 (*Age*) given that X_3 (*Occlusion loc*) is in level-4 (category "M1") and X_4 (*Collaterals*) is in level-4 (category "100% of occluded area") ; Q-Q plot (right) for the variable X_2 (*NIHSS*) given that X_3 is in level-5 (category "M2") and X_4 in level-3 (category "> 50% but less < 100%"). Simulated quantile for the VC model is shown by black square, for the FCS model it is shown by red circle and for the CA model it is shown by blue triangle.

4.5. CONCLUSIONS

In this chapter we compare the VC model with two fully conditional specification approaches, the FCS and CA models, based on their performance in generating synthetic acute ischemic stroke patients' population. Similar comparison has been done in Samuels & et al. (n.d.). However, we consider not only the heuristic vine structure selection in Dißmann et al. (2013) but also more vine structures by the method in Zhu et al. (2020). Similarly, more orders of variables are taken into account in the estimation of the FCS and CA models. By the above model setting a better VC model and also better FCS and CA models are found, according to the MAE of the joint distribution of all the variables.

Although the VC model does not have an overall best performance (the difference among the three models in MAE is small), in most of the margins especially for the selected four variables, the VC model performs the best in modeling the data set whereas the FCS and CA models show tendency of overfitting. The worse performance of the VC model in some margins is due to the mis-specification of the parametric copula, where the estimated copula captures well the tail dependence (e.g. for categories with low frequency of discrete variables) but mis-specifies other dependencies.

Finally we enumerate also some limitations of our research, which will lead to possible future improvement of the models.

- The VC model estimates the dependence structure separately from the 1-dim marginal distributions. The marginal distribution of continuous variable is estimated by its empirical distribution in this chapter, hence one cannot generate data points that are out of the range of the observations. This also applies to the FCS and CA models. Hence, model fitting can be improved by considering other univariate distributions in PIT, for

example, a spline based survival model in Royston & Parmar (2002).

- It has been discussed in Appendix 4.A.2 that, by using non-parametric bivariate copulas in estimating the VC model, the MAE of the joint distribution is larger when compared with the parametric VC model. For some margins, non-parametric bivariate copulas can be preferred as they are more flexible in representing dependence structures especially for discrete variables. However, due to the lack of likelihood based criterion in comparing parametric and non-parametric bivariate copulas, it is not available at this moment to take both of them into account in the estimation of the VC model.
- Only simplified VC model is considered in this chapter. Possible model fitting improvement can be made by assuming a relationship between the copula parameter and the variables conditional on like in Acar et al. (2011), Nagler & Vatter (2020).
- In this chapter we use MAE to compare the three models (the VC, FCS and CA models). The difference in MAE of the three models is quite small, and we are not able to make a decision on which model is significantly better than the others.

As has been mentioned in the beginning of this chapter, synthetic data generation requires to capture all distributions and dependencies observed in real-patient population. Considering the comparison result in this chapter the constructed VC model is a better choice in synthetic data generation. By this conclusion, one can apply the VC model to generate large patient cohorts free of privacy to support descriptive, predictive research or to provide causal application in the INSIST project as discussed in Samuels & et al. (n.d.).

4.A. APPENDIX TO CHAPTER 4

4.A.1. PATIENTS DATA DESCRIPTION

Selected variables were described using mean, standard deviation (sd), skewness, and kurtosis for continuous variables, and frequencies for discrete variables, shown in Table 4.7. Skewness and kurtosis describe the shape of the distribution for a variable, by which one can judge how the distribution is different from a normal distribution (with skewness equal to 0 and kurtosis equal to 3).

Age	mean (sd)	70.0586 (14.1647)
	skewness (kurtosis)	-0.6722 (3.3115)
	range	18.0835~102.7077
Systolic BP	mean (sd)	150.0 (25.02)
	skewness (kurtosis)	0.3991 (3.2805)
	range	68~255
Diastolic BP	mean (sd)	82.4 (15.57)
	skewness (kurtosis)	0.2582 (3.3664)
	range	34~155
Time to ER	mean (sd)	79.0 (61.08)
	skewness (kurtosis)	1.6376 (5.2647)
	range	1~329
Time ER groin	mean (sd)	110.7 (55.05)
	skewness (kurtosis)	0.6465 (3.4717)
	range	1~347
NIHSS	mean (sd)	15.3 (6.17)
	skewness (kurtosis)	0.1062 (3.3184)
	range	0~42
		frequency
Sex	F(0.4783); M(0.5217)	
Prev stroke	N(0.8329); Y(0.1671)	
Prev diabetes	N(0.8397); Y(0.1603)	
Prev fibril	N(0.7602); Y(0.2398)	
HAS	N(0.4513); Y(0.5487)	
Premrs	1(0.6788); 2(0.1301); 3(0.0740); 4(0.0672); 5(0.0415); 6(0.0084)	
Collaterals	1(0.0607); 2(0.3618); 3(0.3864); 4(0.1911)	
Occlusion loc	1(0.0078); 2(0.0503); 3(0.2145); 4(0.5779); 5(0.1496)	
ASPECTS	1(0.0006); 2(0.0042); 3(0.0068); 4(0.0117); 5(0.0221); 6(0.0357); 7(0.0584); 8(0.1113); 9(0.1639); 10(0.2252); 11(0.3602)	

Table 4.7: Basic statistics (mean, standard deviation (sd), skewness, kurtosis and range) for the continuous variables of the acute ischemic stroke patients data.

According to the skewness and kurtosis, except for the variable *Time to ER* other continuous variables are likely to be normally distributed. However, large amount of ties are presented in the data: in 3082 data points, only 2822 different values for *Age*, 140 values for *Systolic BP*, 96 values for *Diastolic BP*, 271 values for *Time to ER*, 270 values for *Time*

ER groin and 43 values for *NIHSS*. Hence in PIT, we transform all the continuous variables by their empirical distribution for the VC model and further transform them into standard normal distribution for the FCS and CA models.

The bivariate dependence, measured by Kendall's tau, for most pairs of variables is not strong (the absolute value of the Kendall's tau is under 0.10). The maximum Kendall's tau is 0.33 between *Systolic BP* and *Diastolic BP*.

4.A.2. BEST VC MODEL SELECTION

We start with pseudo observations in estimating a VC model, as explained in Section 4.4. As discussed in Section 4.2.2, the vine structure of the VC model is at first constructed by the Dißmann's heuristic in Dißmann et al. (2013) and then some additional vine structures having 2 sampling orders in common will be generated and estimated to see whether a better fit can be achieved. For 15 variables it have been suggested in Zhu et al. (2020) to explore 30 additional vine structures. This choice is dictated by a trade-off between computational efficiency and possible improvements. In this chapter we decided to generate at first 20 vines having 2 common sampling orders with the initial one and then each time extra 10 vines are added to see whether the improvement is stable until at a maximum 100 vines are searched. The improvement is judged by the AIC and BIC (without taking into account margins). The result we obtained is shown in Table 4.8.

	improvement			
$\#2SO$	$\#imp(AIC)$	$\Delta AIC(imp)$	$\#imp(BIC)$	$\Delta BIC(imp)$
20	1	2.56	2	12.15
30	4	4.18	3	12.61
40	7	5.19	6	17.97
50	7	5.19	6	17.97

Table 4.8: The improvement as compared with the initial vine by searching $\#2SO$ additional vines having 2 common sampling orders with the initial structure. $\#imp(AIC)$ denotes the number of vines having lower AIC and $\Delta AIC(imp)$ means the absolute average difference in AIC between the initial vine and the vines having lower AIC. Similarly $\#imp(BIC)$ and $\Delta BIC(imp)$ are defined.

We can observe that after 40 vines have been considered 7 of them had better AIC. Additional 10 extra vines, however, did not lead to any improvement in AIC or BIC, so the process stopped. There are in total 10 different vines that are either have better AIC or BIC. In Table 4.9 we show the AIC and BIC of these 10 vines, and the result from a Vuong test (Vuong (1989)) (without or with Schwarz correction), both on the evaluation and test data.

When the Vuong test (also the Vuong test with Schwarz correction) is applied on the evaluation data, one can conclude that, at 5% significance level, it cannot be rejected all these 10 vines perform not significantly different than the initial vine. However, for the test data set, the first, sixth, seventh and the eighth vines having 2 sampling orders in common perform significantly better than the initial vine according to the Vuong test with Schwarz correction. Out of those vines we chose the one having the smallest AIC and also BIC. Hence, the seventh vine having 2 sampling orders in common with the

	Initial			vines2SO		
AIC_{in}	-2422.72	**-2437.28**	-2430.27	-2427.04	-2426.70	-2425.27
BIC_{in}	-1961.22	**-1987.03**	-1957.52	-1948.65	-1948.32	-1958.15
p-value		0.5992	0.5916	0.6435	0.6508	0.8331
		0.1989	0.8636	0.5732	0.5586	0.8871
				vines2SO		
AIC_{in}		-2424.99	-2423.81	-2420.79	-2418.70	-2406.01
BIC_{in}		-1974.74	-1984.83	-1981.81	-1979.71	-1967.02
p-value		0.9364	0.7605	0.6305	0.5935	0.2509
		0.5329	0.2972	0.3184	0.4114	0.7874
AIC_{out}	-890.02	-913.85	-889.15	-898.25	-907.18	-903.13
BIC_{out}	-485.40	-519.10	-474.66	-478.83	-487.76	-493.57
p-value		0.1929	0.8476	0.3826	0.1324	0.3365
		0.0270	0.5106	0.6870	0.8781	0.6030
				vines2SO		
AIC_{out}		-917.26	**-920.52**	-918.37	-890.60	-890.94
BIC_{out}		-522.51	**-535.63**	-533.48	-505.72	-506.05
p-value		0.1245	0.1400	0.1733	0.6425	0.6372
		0.0142	0.0010	0.0013	0.2035	0.1689

Table 4.9: Result for the initial vine (Initial) and the other 10 vines having 2 common sampling orders (vines2SO). AIC_{in} (BIC_{in}) denotes the AIC (BIC) of the model for the estimated data set. AIC_{out} (BIC_{out}) are results for the test data set. The p-value of the Vuong test (with and without [underneath], Schwarz correction) of equality of the initial vine and each one vine having 2 common sampling orders for different data sets are shown.

initial constitutes the final structure.

It is suggested in Nagler (2018), that when discrete variables are modeled by the VC method, non-parametric copulas should be preferred. To investigate this claim, we run a similar procedure as in the case of parametric VC model, where the bivariate copulas are modeled non-parametrically (by 2-dimensional Gaussian kernel smoothing method). In this analysis we take the MAE of the joint distribution on the evaluation and test data sets as the measure of performance. An initial vine is estimated by the Dißmann's heuristic and several vines having 2 common sampling orders are estimated. The process is shown in Table 4.10,

	improvement			
$\#2SO$	$\#imp(MAE)_{eval}$	$\Delta MAE(imp)_{eval}$	$\#imp(MAE)_{test}$	$\Delta MAE(imp)_{test}$
20	7	3.612E-7	9	1.324E-5
30	13	6.107E-6	16	1.792E-5
40	15	3.432E-6	19	2.204E-5
50	20	2.032E-6	23	1.861E-5
60	23	1.957E-7	27	2.034E-5

Table 4.10: The improvement as compared to the initial vine by searching for other $\#2SO$ vines. $\#imp(MAE)_{eval}$ ($\#imp(MAE)_{test}$) denotes the number of vines having smaller MAE on the evaluation (test) data, and $\Delta MAE(imp)_{eval}$ ($\Delta MAE(imp)_{test}$) means the absolute average difference in MAE between the initial vine and the vines having smaller MAE on the evaluation (test) data.

Although the number of vines having smaller MAE increases when more vines having 2 common sampling orders are constructed, the average improvement in MAE becomes smaller after 30 vines have been considered on the evaluation data set and remains stable on the test data. We stop the search after 60 vines having 2 common sampling orders have been examined. The best vine structure in the non-parametric VC model is different than the one in the best parametric VC model. This structure has MAE (times 100) of the joint distribution equal to 0.0832 on the evaluation data set and 0.1208 on the test data set. When compared with the MAE (times 100) for the best parametric VC model with MAE (times 100) of 0.0807 on the evaluation data and 0.1191 on the test data, we decide to use the parametric VC model in this chapter.

4.A.3. MAE AND STATISTICS OF THE SYNTHETIC DATA OF 1-DIM MARGINS

In this section results concerning MAE and the mean absolute difference of statistics (the mean, the standard deviation (sd), the range, the 5% quantile, the 50% quantile and the 95% quantile) are shown. The MAE is given in Table 4.11 and the statistics is in Table 4.12 and Table 4.13 (In the test data for the variable *ASPECTS*, level 0 does not appear hence there is no test statistic for this variable on the test data set).

	Dim		100 × *MAE*					
eval	1		*Age*	*Systolic BP*	*Diastolic BP*	*Time to ER*	*Time ER groin*	*NIHSS*
		VC	**0.3259**	**0.3576**	**0.3393**	**0.3150**	**0.3564**	**0.3393**
		FCS	0.5614	0.6252	0.7816	0.5214	0.7722	0.8289
		CA	0.6645	0.6178	0.5935	0.7254	0.4400	0.7897
			Sex	*Prev stroke*	*Prev diabetes*	*Prev atrial fibril*	*HAS*	*Premrs*
		VC	0.7518	**0.3380**	**0.4080**	**0.5382**	0.4412	**0.5449**
		FCS	**0.4918**	1.6683	1.0080	0.9054	**0.2681**	1.6540
		CA	0.6463	0.8188	0.5091	1.0237	0.5395	0.6562
			Collaterals	*Occlusion loc*	*ASPECTS*			
		VC	0.3816	0.4491	0.2929			
		FCS	**0.3260**	**0.2794**	**0.2886**			
		CA	0.3403	0.4538	0.3668			
test	1		*Age*	*Systolic BP*	*Diastolic BP*	*Time to ER*	*Time ER groin*	*NIHSS*
		VC	**0.3267**	**0.3616**	**0.3272**	**0.3207**	**0.3558**	**0.3395**
		FCS	0.5699	0.6235	0.7723	0.5153	0.7668	0.8190
		CA	0.6662	0.6179	0.5923	0.7257	0.4363	0.7941
			Sex	*Prev stroke*	*Prev diabetes*	*Prev atrial fibril*	*HAS*	*Premrs*
		VC	**1.5395**	**0.3792**	0.8476	**0.5465**	**0.7987**	**1.2730**
		FCS	1.8257	1.9945	**0.6375**	1.8478	0.9874	3.4772
		CA	2.0070	0.8863	1.2787	1.0289	1.2230	1.9687
			Collaterals	*Occlusion loc*	*ASPECTS*			
		VC	**1.0899**	**0.5068**	**0.6029**			
		FCS	1.5548	0.8472	0.9459			
		CA	1.4656	0.9119	1.0002			

Table 4.11: Result of MAE (the MAE is multiplied by 100) for each variable for the VC, FCS and CA methods, where eval denotes the evaluation data set and test denotes the test data set.

eval			*Age*	*Systolic BP*	*Diastolic BP*	*Time to ER*	*Time ER groin*	*NIHSS*
	mean	VC	**0.1154**	**0.2319**	**0.1289**	**0.5022**	0.5615	**0.0528**
		FCS	0.2145	0.4760	0.3923	0.9000	1.1475	0.1562
		CA	0.3130	0.4686	0.3070	1.4937	**0.4721**	0.1617
	sd	VC	0.1091	**0.1861**	**0.1069**	**0.4163**	**0.4501**	**0.0329**
		FCS	0.1873	0.3735	0.3126	1.0766	1.1546	0.1010
		CA	**0.0972**	0.3511	0.1194	1.4989	0.8201	0.1234
	range	VC	**0.1982**	3.1000	4.0000	0	2.4000	0.1000
		FCS	0.2050	1.8000	**2.1000**	0	3.4000	0.1000
		CA	0.2724	**1.7000**	3.4000	0	**1.6000**	**0**
	5%	VC	**0.2701**	**0.8000**	**0.2000**	**0.4000**	0.3000	0.3000
		FCS	0.4653	0.9000	0.7000	0.6000	0.2000	0.3000
		CA	0.5043	0.9000	0.6000	1.3000	**0.2000**	**0.2000**
	50%	VC	**0.1525**	**0.1000**	**0.2000**	**0.4000**	0.9000	0.1000
		FCS	0.2128	0.3000	0.7000	0.8000	2.4000	**0**
		CA	0.3822	**0.1000**	0.7000	1.4000	**0.6000**	0.4000
	95%	VC	**0.1248**	**1.3000**	0	**0.7000**	**1.5000**	**0.1000**
		FCS	0.3622	2.2000	0	2.4000	3.9000	**0.1000**
		CA	0.2571	1.9000	0	3.1000	3.1000	0.3000
			Sex	*Prev stroke*	*Prev diabetes*	*Prev fibril*	*HAS*	*Premrs*
	χ^2	VC	11.3321	**1.7791**	**2.3784**	**4.0022**	5.1540	8.7561
		FCS	**6.7920**	32.8880	15.3236	12.0676	**2.8553**	41.8390
		CA	12.2931	9.2619	4.1972	13.5430	8.2438	**8.2380**
			Collaterals	*Occlusion loc*	*ASPECTS*			
		VC	6.1441	**14.6297**	25.9651			
		FCS	7.3615	34.7652	27.3516			
		CA	**5.3883**	18.3822	**22.4397**			

Table 4.12: Mean absolute difference of the statistics (mean, sd, range, 5%, 50% and 95% quantile for continuous variables and the χ^2 test statistic for the discrete variables, between the evaluation data set (eval) and the synthetic data.) for the VC, FCS and CA methods.

test			*Age*	*Systolic BP*	*Diastolic BP*	*Time to ER*	*Time ER groin*	*NIHSS*
	mean	VC	**0.1126**	**0.2393**	**0.1231**	**0.4969**	0.5521	**0.0519**
		FCS	0.2169	0.4936	0.3970	0.8874	1.1384	0.1478
		CA	0.3088	0.4797	0.3019	1.5056	**0.4771**	0.1574
	sd	VC	0.1221	**0.2598**	0.1355	**0.4302**	**0.4651**	**0.0368**
		FCS	0.1659	0.3730	0.3309	1.0018	1.0047	0.0862
		CA	**0.0866**	0.4135	**0.1319**	1.6012	0.7602	0.1052
	range	VC	0.5303	2.2000	0.5000	0.7000	0.7000	**0**
		FCS	0.9564	**0.8000**	**0.1000**	**0.8000**	0.7000	0.2000
		CA	**0.4322**	1.9000	0.4000	1.6000	0.7000	0.1000
	5%	VC	**0.2597**	1.1000	**0.2000**	**0**	**0.8000**	0.1000
		FCS	0.3539	**0.8000**	1.1000	0.5000	0.9000	0.1000
		CA	0.5193	0.9000	0.7000	0.2000	0.9000	**0**
	50%	VC	**0.1486**	**0.3000**	**0**	**0.5000**	**0.6000**	0.1000
		FCS	0.1727	0.5000	**0**	0.7000	1.4000	**0**
		CA	0.3425	0.4000	0.1000	1.1000	1.1000	0.3000
	95%	VC	**0.1772**	**0.5000**	0.4000	**0.9000**	**1.3000**	**0**
		FCS	0.6730	2.0000	**0.2000**	2.3000	2.1000	0.1000
		CA	0.3494	1.8000	0.2000	3.5000	1.7000	0.3000
			Sex	*Prev stroke*	*Prev diabetes*	*Prev fibril*	*HAS*	*Premrs*
	χ^2	VC	**37.2790**	**2.3704**	8.7518	**3.6054**	**12.7387**	**35.9456**
		FCS	53.6656	45.6811	**5.9420**	38.8341	19.1755	203.2500
		CA	72.3792	11.4726	20.5623	16.2526	35.3935	73.5746
			Collaterals	*Occlusion loc*	*ASPECTS*			
		VC	**22.4813**	**46.4045**				
		FCS	56.7458	102.4522				
		CA	52.5757	125.0656				

Table 4.13: Mean absolute difference of the statistics (mean, sd, range, 5%, 50% and 95% quantile for continuous variables and the χ^2 test statistic for the discrete variables, between the test data set (test) and the synthetic data.) for the VC, FCS and CA methods.

5

SIMPLIFIED R-VINE BASED FORWARD REGRESSION

An extension of the D-vine based forward regression procedure to a R-vine forward regression is proposed. In this extension any R-vine structure can be taken into account. Moreover, a new heuristic is proposed to determine which R-vine structure is the most appropriate to model the conditional distribution of the response variable given the covariates. It is shown in the simulation that the performance of the heuristic is comparable to the D-vine based approach. Furthermore, it is explained how to extend the heuristic into a situation when more than one response variable are of interest. Finally, the proposed R-vine regression is applied to perform a stress analysis on the manufacturing sector which shows its impact on the whole economy.

Parts of this chapter have been published in Zhu et al. (2021).

5.1. INTRODUCTION

Linear regression is one of the most popular models used in many applications. The conditional distribution of the response variable(s) is modeled by describing its conditional expectation as a linear function of covariates and the error term. The strong assumptions of this model, which may not be appropriate for some data sets, can be alleviated by applying transformations, for example in the generalized linear models McCullagh & Nelder (1989).

The problem of modeling the relationship between the response variable(s) and the covariates can also be approached by estimating their joint distribution, from which the conditional distribution (or its expectation) can be computed. The joint distribution can be estimated by estimating the marginal distributions and a copula separately. In Parsa & Klugman (2011) a multivariate Gaussian copula is used for which the conditional distribution is given in closed form. Multivariate copula families other than Gaussian (e.g. Archimedean families) might give a better approximation of the data and have also been applied in the context of regression Noh et al. (2013). However, the conditional density, in the above approach, has to be computed as a quotient of a joint density of all variables and a joint density of the covariates, where the joint density of the covariates is computed by integrating out the response variable(s). This approach is computationally expensive. Moreover, the dependence structure represented by the copula is restricted to few multivariate copula families which might not be sufficiently flexible in higher dimensions.

Vine copula model can also be applied in regression analysis. For some vine structures the conditional distribution of the response variable given the covariates can be estimated directly from the vine rather than through integration. This is possible when the response variable appears in the conditioned set of the edge in the last tree of the vine. Under this requirement, the regular vine copula model has been used in Cooke et al. (2019) to estimate the conditional distribution of IQ given duration of breastfeeding and other covariates.

In Kraus & Czado (2017*a*), a D-vine based forward regression procedure has been introduced. The vine structure is fixed to be a D-vine and the order of variables in the D-vine, as introduced in Brechmann & Schepsmeier (2013), is chosen via the forward selection procedure fixing the response variable as the first element in this order. These choices reduce computational burden of vine copula regression method and at the same time allow variable selection. The D-vine copula model used in Kraus & Czado (2017*a*) is the so called simplified vine where all conditional copulas do not directly depend on the conditioning variables. More about the simplifying assumption can be found in Haff et al. (2010) and Stöber et al. (2013). In Herrmann (2018) and Chang & Joe (2019) different heuristics based on computing some correlation measures to choose the vine structure in the regression setting have been explored.

The contribution of this chapter is an extension of the method presented in Kraus & Czado (2017*a*) to allow other structures than the D-vine in estimation of the conditional distribution. Our goal is to test whether more flexibility in the choice of vine structure (which is more computationally intensive) can lead to significantly better copula regression model. We propose a general approach of how the vine structure can be obtained during the forward selection such that the conditional distribution can be estimated in

the analytic form. The approach is motivated by the results and algorithms presented in Zhu et al. (2020). Furthermore, we propose a new heuristic R-vine based forward selection procedure which is different than the tree-wise maximization of correlations in Herrmann (2018) during extension. Our new heuristic maximizes the correlations 'globally' when constructing the vine structure.

Moreover, we consider in this chapter the case when more than one response variables are of interest. Our heuristic of choosing a vine structure has been extended and allows to build the joint conditional distribution of multiple response variables given the covariates. The conditional distribution of two response variables obtained by our heuristic is given in an analytic form but in the case of more than two requires integration.

In this chapter we estimate only simplified vine copulas. To take the effect of the simplifying assumption into account, a nonparametric method to estimate the relationship between the parameter in the conditional copula and the covariates is introduced in Acar et al. (2011) (in 3 dimensional setting). Moreover a fully nonparametric estimation of the joint conditional distribution is proposed in Gijbels et al. (2011), Veraverbeke et al. (2011). However, none of these lends itself to be applied in the case where many covariates are of interest. They are simply too computationally intensive and not scalable to higher dimensions. It is still an open question how to alleviate the effect of the simplifying assumption.

This chapter is organized as follows: Section 5.2 explains how the conditional distribution can be represented by regular vines and which requirement is necessary such that the conditional distribution can be estimated without integration. A review of the D-vine based forward regression is presented in Section 5.3.1. Furthermore, we propose our heuristic for the R-vine based forward selection procedure in Section 5.3.2. A comparison of the performance for different regression models based on one example data set is shown in Section 5.3.3. In Section 5.4 a simulation study to compare the D-vine and R-vine forward selection methods is presented. Moreover, in Section 5.5, our heuristic is generalized to allow estimating joint conditional distribution of more than one response variable. To improve the R-vine regression model in Section 5.6 a method introduced in Zhu et al. (2020), that proposes to search for several vines having 2 sampling orders in common with the initial one, is modified and applied. The real data analysis is in Section 5.7 and the conclusion can be found in Section 5.8.

5.2. CONDITIONAL DENSITY BASED ON REGULAR VINES

In Figure 5.1 examples of two special vine structures, C-vine $V_1(5)$ and D-vine $V_2(5)$ are shown.

Regression analysis requires modeling of the conditional density. One of the advantages of adopting the regular vine decomposition for the density is that the conditional density can be easily available. The conditional density of the response variable is equal to the product of its marginal density and all copula densities assigned to the edges of the vine whose conditioned set includes the response variable Cooke et al. (2015). For example, the conditional density $f_{1|2345}$ can be obtained from the decomposition of $V_1(5)$ as

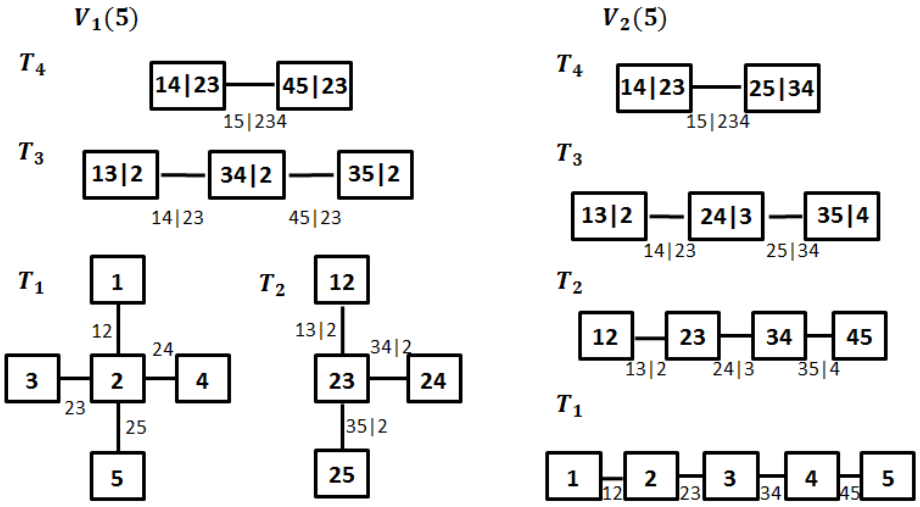

Figure 5.1: Tree-wise representation for two vines $V_1(5)$ (left) and $V_2(5)$ (right).

follows:

$$f_{1|2345} = f_1 c_{12}(F_1,F_2) c_{13;2}(C_{1|2},C_{3|2}) c_{14;23}(C_{1|23},C_{4|23}) c_{15;234}(C_{1|234},C_{5|234}).$$

Then the conditional distribution $F_{1|2345}$ can be represented as follows (the arguments of the copulas can be calculated in the same way, and the h-function has been discussed in Section 1.2.2 in Chapter 1),

$$\begin{aligned} F_{1|2345} &= h_{1|5;234}(C_{1|234}|C_{5|234}) \\ &= h_{1|5;234}\left(h_{1|4;23}\left(C_{1|23}|C_{4|23}\right)|h_{5|4;23}\left(C_{5|23}|C_{4|23}\right)\right) \\ &= h_{1|5;234}(h_{1|4;23}(h_{1|3;2}(h_{1|2}(F_1|F_2)|h_{3|2}(F_3|F_2))|h_{4|3;2}(h_{4|2}(F_4|F_2)|h_{3|2}(F_3|F_2)))| \\ &\quad h_{5|4;23}(h_{5|3;2}(h_{5|2}(F_5|F_2)|h_{3|2}(F_3|F_2))|h_{4|3;2}(h_{4|3}(F_4|F_3)|h_{4|2}(F_4|F_2)))). \end{aligned}$$

The conditional expectation of the response variable is in general not available in closed form and has to be computed by integration or by simulation. However, the quantile function at quantile α of the conditional distribution $F_{1|2345}$ with density $f_{1|2345}$ can be computed as follows:

$$\begin{aligned} F^{-1}_{1|2345}(\alpha) = {} & F_1^{-1} h^{-1}_{1|2}(h^{-1}_{1|3;2}(h^{-1}_{1|4;23}(h^{-1}_{1|5;234}(\alpha| \\ & h_{5|4;23}(h_{5|3;2}(h_{5|2}(F_5|F_2)|h_{3|2}(F_3|F_2))|h_{4|3;2}(h_{4|3}(F_4|F_3)|h_{4|2}(F_4|F_2)))))| \\ & h_{4|3;2}(h_{4|2}(F_4|F_2)|h_{3|2}(F_3|F_2)))|h_{3|2}(F_3|F_2))|F_2). \end{aligned}$$

The estimation of the vine copula model is a two-step approach called the Inference Functions for Margins (IFM) introduced in Joe (1997). Suppose we have a data set

$(x_k^m, x_1^m, \ldots, x_j^m)$ with $m = 1,2,\ldots,N$, the first step is to estimate each marginal distribution F_i, respectively. In Joe (1997), it has been advised to use nonparametric approaches to get F_i in IFM method. The marginal data is transformed into u-scale by applying probability integral transformation $u_i^m = F_i(x_i^m)$. The vine copula model is estimated based on the transformed data set u_i^m called the pseudo observations. The log-likelihood expression for the conditional distribution $F_{k|1,\ldots,j}(x_k^m|x_1^m,\ldots,x_j^m)$ denoted after transformation of margins as $C_{k|1,\ldots,j}(u_k^m|u_1^m,\ldots,u_j^m)$ is as follows,

$$cll_{k|1,\ldots,j}(u_k^m, u_1^m, \ldots, u_j^m; \hat{\boldsymbol{\theta}}) = \sum_{m=1}^{N} \ln\Big(\prod_{i=1}^{j} c_{ki;D_e}\big(C_{k|D_e}(u_{k|D_e}^m; \hat{\boldsymbol{\theta}}_{\boldsymbol{k}}), C_{i|D_e}(u_{i|D_e}^m; \hat{\boldsymbol{\theta}}_{\boldsymbol{i}}); \hat{\theta}_{ki}\big)\Big), \tag{5.2.1}$$

where D_e represents the conditioning set in (un)conditional bivariate copulas and $\hat{\theta}_{ki}$ denotes their parameters. Pseudo observations $u_{k|D_e}^m$ and $u_{i|D_e}^m$ are calculated by the h-function. The vectors of parameters $\hat{\boldsymbol{\theta}}_{\boldsymbol{k}}$ and $\hat{\boldsymbol{\theta}}_{\boldsymbol{i}}$ contain parameters in the arguments of the copula obtained by the h-functions. Moreover, $\hat{\boldsymbol{\theta}}$ is a vector containing all parameters of the vine model. We can further penalize the cll by the number of parameters to get cll^{AIC} and cll^{BIC},

$$\begin{aligned} cll^{AIC}_{k|1,\ldots,j}(u_k^m, u_1^m, \ldots, u_j^m; \hat{\boldsymbol{\theta}}) &= -2cll_{k|1,\ldots,j}(u_k^m, u_1^m, \ldots, u_j^m; \hat{\boldsymbol{\theta}}) + 2|\hat{\boldsymbol{\theta}}|, \\ cll^{BIC}_{k|1,\ldots,j}(u_k^m, u_1^m, \ldots, u_j^m; \hat{\boldsymbol{\theta}}) &= -2cll_{k|1,\ldots,j}(u_k^m, u_1^m, \ldots, u_j^m; \hat{\boldsymbol{\theta}}) + \ln(N)|\hat{\boldsymbol{\theta}}|. \end{aligned} \tag{5.2.2}$$

However, not every regular vine structure will represent the given conditional density directly. To get this easy representation it is necessary that the response variable is in the conditioned set of the edge in the last tree. As seen from Figure 5.1, it will not be possible to obtain $F_{3|1245}$ based on $V_1(5)$ or $V_2(5)$ as variable 3 does not appear in the conditioned set of the edge 15|234. Though in Cooke et al. (2015) a method of computing conditional density in such case was proposed (called plug-in conditionalization), it will be very computationally expensive especially in high dimensions.

Nonetheless, there are still many vine structures that can decompose the conditional density easily and one needs to decide which structure is the best to use. Theoretically, it does not matter as all these structures model the same conditional distribution. When estimated from data however, some structures may perform better than others due to 1) limited choice of copula families in the **VineCopula** package; 2) numerical errors in the tree-wise estimation; 3) the simplifying assumption which assumes that the conditional copula does not depend directly on the conditioning variables (which simplifies estimation but can have an impact on the performance of different vine structures).

In the next section we discuss the forward selection procedure for vine regression with one response variable.

5.3. VINE REGRESSION - FORWARD SELECTION

In this section we first explain briefly the D-vine based forward selection procedure introduced in Kraus & Czado (2017*a*). Then the extension allowing the choice of different

regular vine structures will be presented. We will end this section with a detailed analysis of an example data set to compare the performance of different regression methods.

5.3.1. D-VINE FORWARD SELECTION

The vine copula forward selection in Kraus & Czado (2017*a*) is a two-step approach where the marginal distributions are estimated first by kernel smoothing. Then the dependence structure in the form of a D-vine copula model is estimated. The D-vine structure is uniquely determined by the order of variables as discussed in Section 2.3 in Chapter 2. This order will be determined during the forward selection procedure. Suppose that l_0 is the index of the response variable and $(l_1,\dots,l_k)$ are the set of indices of the covariates. Each step of the procedure consists of picking a new variable from the set of covariates that contributes the most to the conditional distribution computed from a D-vine model extended by this variable. The contribution is measured by the conditional penalized likelihood cll^{AIC} shown in Equation 5.2.2. As we discussed in Section 2.3.2 in Chapter 2, once a new variable is added all the new copulas are the copulas whose conditioned set contains the new variable. Thus we can further measure the contribution by an improvement Δcll^{AIC}, which is the AIC for the new copula $C_{l_0 l_i;l_1,\dots,l_{i-1}}$ plus a penalty of the number of parameters in other new copulas. In case of no improvement, $\Delta cll^{AIC} \geq 0$ and the procedure will stop.

The general algorithm for the D-vine based forward regression presented in Kraus & Czado (2017*a*) is in Algorithm 9.

Algorithm 9 D-vine based forward selection

Input: a data set including variables $(Y,\mathbf{X})$ where $\mathbf{X}=\{X_1,X_2,\dots\}$
Output: a conditional distribution $F_{Y|\mathbf{D}}$ where $\mathbf{D}\subseteq\mathbf{X}$

Regard l_0 as the index of Y
Initialize Δcll^{AIC} to be 0, $\mathbf{D}=\emptyset$ and $I=\{l_0\}$
repeat
  for each new variable candidate X_{new} in $\mathbf{X}$ in the ith step of forward procedure **do**
    Extend the previous D-vine with order I by adding the new variable X_{new} to have a D-vine with order (I,j) where j denotes the index of X_{new}
    Estimate the new copulas whose conditioned set contains j
    Calculate the Δcll^{AIC} from the new conditional distribution $F_{Y|\mathbf{D}\cup\{X_{new}\}}$
  end for
  Choose X_{new}^{best} that gives us the smallest Δcll^{AIC} (Δcll_{min}^{AIC})
  if $\Delta cll_{min}^{AIC}<0$ **then**
    Exclude X_{new}^{best} from $\mathbf{X}$ and add it into $\mathbf{D}$
    Add l_i in the end of I where l_i denotes the index of X_{new}^{best}
  end if
until $\Delta cll_{min}^{AIC}\geq 0$

If the structure of the vine is known (D-vine, C-vine or other structures) then the estimation of the conditional distribution can be carried out as above. We will see in

Section 5.3.2 how to choose the R-vine structure for a given order of variables which we get from the forward selection.

5.3.2. R-VINE FORWARD SELECTION - A HEURISTIC METHOD

We have explained in Section 2.3.2 in Chapter 2, that the vine structure is determined by the substructure for a given order and a sequence of indicators. Similar to the D-vine based process the order for the R-vine structure is chosen by computing Δcll^{AIC} for each new variable candidate. In contrast to the D-vine based procedure where the vine structure is determined for each new variable candidate in the ith step of the forward selection, we have to check 2^{i-2} regular vine structures and choose the one that gives us the best fitting conditional distribution. This process is carried out through extending the VBT with order $(l_0, l_1, ..., l_{i-1})$ by variable l_i hence specifying values of $i-2$ indicators. Estimating all 2^{i-2} regular vines is computationally infeasible (though could be carried out in lower dimensions) so to make it applicable we propose a heuristic to choose the vine structure in the forward selection process when a new variable is added.

Suppose the order of variables in the forward selection is $(l_0, l_1, l_2, l_3, l_4)$ as in Figure 5.2.

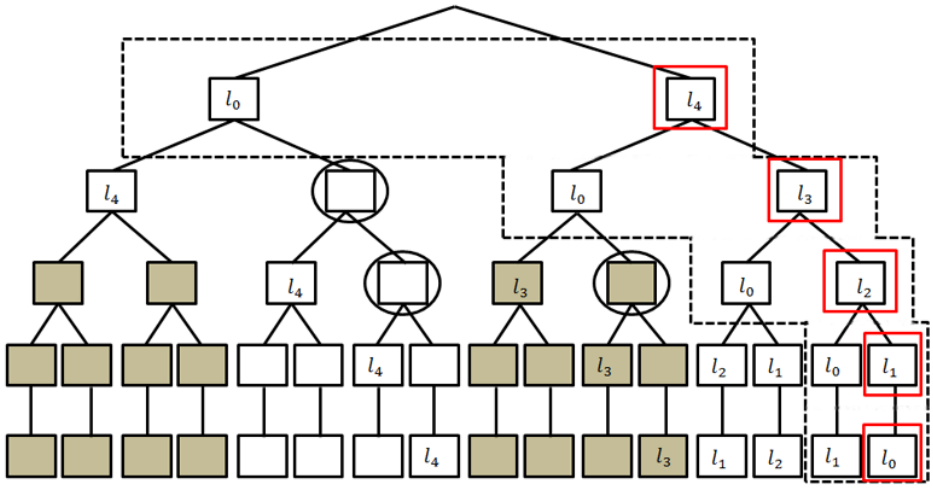

Figure 5.2: VBT with given order $(l_0, l_1, l_2, l_3, l_4)$ in the extension process where the sub-tree underneath VBT[1][2] is already filled. The elements where choices are possible are indicated by a black circle.

The response variable l_0 is placed in VBT[1][1] thus the substructure determined by the order will be the most right one. Unspecified elements indicated by a circle are determined by values of a sequence of indicators and the remaining elements can be filled in by the properties of VBT. When l_3 is added into the R-vine model with order (l_0, l_1, l_2), we have 2 choices in VBT[3][6]. The new copulas we need to estimate are either $C_{l_0 l_3;l_1 l_2}$, $C_{l_3 l_2;l_1}$, $C_{l_3 l_1}$ if indicator is 1 or $C_{l_0 l_3;l_1 l_2}$, $C_{l_3 l_1;l_2}$, $C_{l_3 l_2}$ if indicator is 0. To each choice of structure determined by indicators we can assign weight $\omega_{ij;D_e}$, which leads to a binomial tree like structure as shown in Figure 5.3 (left). We name this structure **weighted**

binomial tree (wBT). If we set VBT[3][6] = l_2, we can similarly have the wBT in Figure 5.3 (right) when l_4 is added.

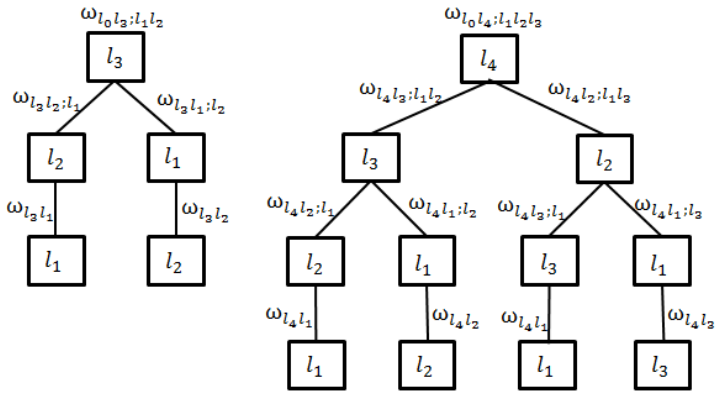

Figure 5.3: Weighted Binomial Tree (wBT) in the third step (left) and the fourth step (right) in the forward selection.

Each path in wBT corresponds to one indicator sequence thus one vine structure. Instead of estimating copulas for all possible vine structures our heuristic proposes to choose the structure determined by the path of wBT with the largest sum of weights. Different weights can be chosen at this step. They could be based on correlations, estimates of tail dependence and likelihood related measures etc. In this chapter we propose to compute them as follows:

Heuristic - 1 response variable

1. *Assume all new copulas $C_{ij;D_e}$ emerging from any path in wBT are Gaussian copulas.*

2. *Set the weight $\omega_{ij;D_e}$ to each Gaussian copula $C_{ij;D_e}$ as its parameter, hence the partial correlation $\rho_{ij;D_e}$.*

3. *Choose the path in wBT which gives the largest sum of the absolute value of the weights (partial correlations).*

4. *Construct the new vine structure according to the chosen path.*

The algorithm of the R-vine based forward regression differs from the one presented for D-vine only at line (4) in Algorithm 9 which now becomes:

As mentioned in the introduction the D-vine based forward selection regression method has been already extended to R-vine based regression in Herrmann (2018) and in Chang & Joe (2019). In Herrmann (2018) two heuristic methods have been introduced. The vine structure is constructed by first choosing an order using 1) D-vine forward selection, or

Algorithm 10 R-vine based forward selection

⋮

5: Extend the previous heuristic R-vine with order I by adding the new variable X_{new} to have a R-vine with order (I, j) where j denotes the index of X_{new}

⋮

2) partial correlations (variable with the largest absolute value of correlation with the response variable becomes second in the order, then all partial correlations with the response variable given the second in order are examined and the largest one becomes the third element in the order etc). Given the order the process is restarted and the vine structure is extended by adding one covariate at a time according to the chosen order and picking the new edge in each tree that corresponds to the largest correlation (empirical Kendall's tau). In contrast to this tree-wise structure selection for the given order our heuristic chooses the structure which gives us the largest sum of the absolute value of partial correlations globally in each step of extension. The method introduced in Chang & Joe (2019) is not a forward selection procedure. It constructs the vine copula of the covariates using the heuristic in Dißmann et al. (2013) and extends this vine by the response variable such that this structure captures the largest correlation in each tree.

It has been shown in both these papers that new heuristics behave quite similarly to the D-vine based forward regression. Thus, in this chapter we mainly compare our heuristic with the D-vine based regression. The comparison is done via the illustration example in Section 5.3.3 and also via a simulation study in Section 5.4.

5.3.3. ANALYSIS OF AN EXAMPLE DATA SET

In this section a comparison of different forward selection regression methods will be analyzed based on one example data set.

We simulate 1000 observations from $\mathbf{X} = (X_1, \ldots, X_8)$ where $(X_1, X_2, X_3, X_4, X_5)$ each has exponential distribution with parameters $(0.5, 1, 1.5, 2, 2.5)$ respectively, and X_6, X_7, X_8 are standard normally distributed. The dependence structure of $\mathbf{X}$ is from a D-vine copula with order $(1, 4, 8, 7, 6, 5, 3, 2)$ (see detailed information in Table 5.14 in Appendix 5.A.1). The bivariate copulas are chosen out of Gumbel(G), Clayton(C) and Joe(J) copulas as well as their rotated versions. The copula parameters are transformed from Kendall's tau which are simulated from a *Beta(2,2)* distribution, and are allowed to take negative values with probability 0.5.

The cll^{AIC} for the true conditional distribution $C_{1|2345678}$ when margins are transformed into u-scale is -5263.45. Moreover, the pseudo observations of the true model, $C_{1|2345678}(u_1^m | u_2^m, \ldots, u_8^m)$, should be realizations of the standard uniform distribution. Hence, after applying to them the inverse standard normal cdf we can observe in Figure 5.4 the performance of the true model on the data via Q-Q plot. Similarly, when other models are estimated the conditional distribution of the response variable in each model is obtained and the in sample performance of these models can be visualized by the Q-Q plot as well, which we call in sample validation Q-Q plot.

Estimated Models:

1. Linear regression

A linear forward regression model is estimated by the function `lm` in R with forward selection based on AIC implemented by the function `step`. Estimation details for this model are listed in Table 5.15 in Appendix 5.A.1. The in sample validation Q-Q plot for the linear regression model is shown in Figure 5.4 (black line). It is clear that this model does not preform well and is especially poor in the tails. This is not surprising as the example was designed to make it difficult for the linear regression model.

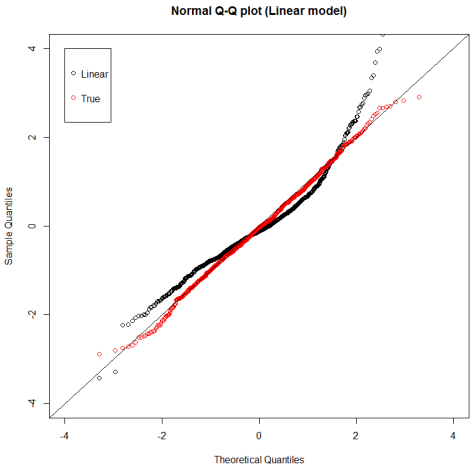

Figure 5.4: In sample validation Q-Q plot for the linear regression (black) in comparison with the true model (red).

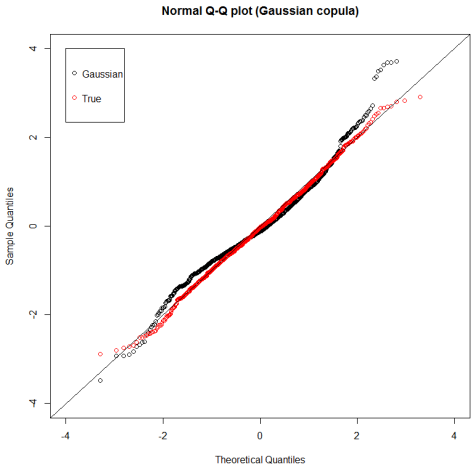

Figure 5.5: In sample validation Q-Q plot for the Gaussian copula model (black) in comparison with the true model (red).

The results for linear regression model are shown for completeness. The main point of this chapter is to compare the copula based models. For all copula based regression models we first transform each margin into their u-scale by applying kernel smoothing through the function `pkde` in the package **ks** (Duong (2019)). Then the conditional distribution $C_{1|D}$ (where D denotes the set of covariates) is estimated based on Δcll^{AIC} through forward selection. We point out that the AIC of the conditional distribution is computed only on the pseudo observations, hence the contribution of margins is not included. For copula based models this comparison is fair.

2. Gaussian copula regression

The Gaussian copula regression is performed by first transforming marginal pseudo observations to z-scale and then fitting a linear forward regression like above. None of the covariates are removed during the forward selection and the AIC of the conditional distribution (without the marginal distribution) is -2384.95. The details of this model are listed in Table 5.16 in Appendix 5.A.1. The in sample validation

Q-Q plot for the Gaussian copula model is in Figure 5.5. As we can see the Gaussian copula regression performs better than the linear model, however, there are still discrepancies especially in the tails.

3. D-vine regression

The details of the D-vine based forward selection are shown in Table 5.1. In the first step, variable X_8 is chosen as it has the smallest $\Delta cll^{AIC} = -1931.48$. In the second step, we choose variable X_2 to be added into the D-vine (a D-vine with order $(1,8)$). Next X_7 is added and the process chooses $l_4 = 6$, $l_5 = 4$, $l_6 = 3$. In the seventh step, the smallest Δcll^{AIC} is 1.68 which is larger than 0, hence the process stops and the final D-vine copula model with order $(1,8,2,7,6,4,3)$ is shown in Table 5.17 in Appendix 5.A.1. The AIC of the conditional distribution is -2461.48.

Forward Selection (D-vine copula)				
step	order of variables	Δcll^{AIC}	order of variables	Δcll^{AIC}
1	$(1,2)$	-196.62	$(1,3)$	-410.33
	$(1,4)$	-329.23	$(1,5)$	-255.28
	$(1,6)$	-35.83	$(1,7)$	-859.10
	(1,8)	-1931.48		
2	**(1,8,2)**	-186.21	$(1,8,3)$	-46.43
	$(1,8,4)$	-112.00	$(1,8,5)$	-71.46
	$(1,8,6)$	-40.45	$(1,8,7)$	-50.76
3	$(1,8,2,3)$	-60.71	$(1,8,2,4)$	-102.19
	$(1,8,2,5)$	-34.46	$(1,8,2,6)$	-27.51
	(1,8,2,7)	-122.36		
4	$(1,8,2,7,3)$	-15.47	$(1,8,2,7,4)$	-30.03
	$(1,8,2,7,5)$	-42.09	**(1,8,2,7,6)**	-59.39
5	$(1,8,2,7,6,3)$	-57.98	**(1,8,2,7,6,4)**	-148.31
	$(1,8,2,7,6,5)$	-2.33		
6	**(1,8,2,7,6,4,3)**	-13.72	$(1,8,2,7,6,4,5)$	4.80
7	$(1,8,2,7,6,4,3,5)$	1.68		

Table 5.1: Forward selection by D-vine copula model. In each step the chosen variable is printed in bold. The final D-vine structure has the order of variables $(1,8,2,7,6,4,3)$.

The in sample validation Q-Q plot for D-vine copula model is shown in Figure 5.6. We can observe that the performance is improved as compared to the Gaussian copula model especially in the tails. This is because in the true model most of the copulas are copulas with tail dependence which cannot be captured properly by the Gaussian copula.

4. R-vine regression

We apply our heuristic to choose the vine structure and the order of variables is determined during the forward selection. The details of the forward selection are shown in Table 5.2. The wBTs to determine the vine structure for each possible

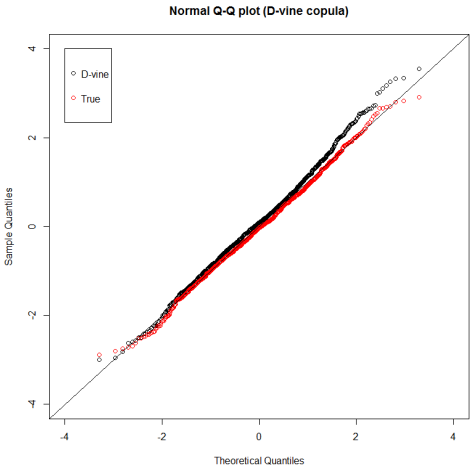

Figure 5.6: In sample validation Q-Q plot for the D-vine regression model (black) in comparison with the true model (red).

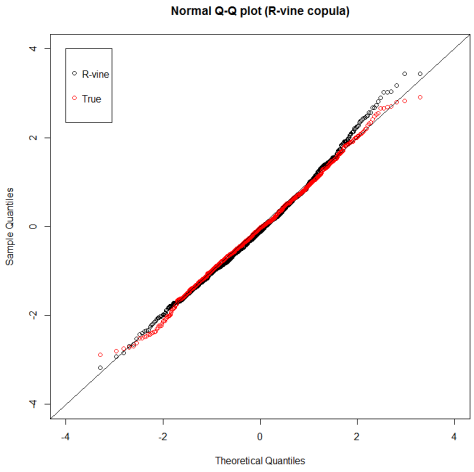

Figure 5.7: In sample validation Q-Q plot for the R-vine regression model (black) in comparison with the true model (red).

5

new variable candidate in each step of the forward selection are in Appendix 5.A.1 and the final R-vine copula model is in Table 5.22 in Appendix 5.A.1. Note that the first 2 steps of the R-vine and the D-vine based forward selection are the same since there is no choice for the vine structure for a given order.

The in sample validation Q-Q plot for the R-vine model is shown in Figure 5.7. In the R-vine model variable X_5 that is removed in the previous D-vine model is added in the fourth step in the forward selection. The final AIC of the conditional distribution is -2677.72 which is smaller than the AIC from the D-vine model. The validation plot is comparable with the one obtained with the D-vine copula model in Figure 5.6. The deviation in the upper tail is a bit smaller.

To not only visually compare the performance with respect to the in sample validation Q-Q plots we also apply the AD (Anderson-Darling) and Shapiro tests of normality to the transformed pseudo observations of the obtained conditional distributions for different models. The results are collected in Table 5.3. All models do not estimate the conditional distribution well enough. At the significance level 5% all these models would be rejected. This could be caused by the forward selection process, estimation error, misspecification of copula families as well as the assumption of simplified vine copula (both the D-vine and the R-vine model reject the hypothesis of being simplified at the 5% significance level using the 'CCC' test implemented in the function `pacotest` introduced in Kurz (2019)).

Forward Selection (R-vine copula)				
step	order of variables	Δcll^{AIC}	order of variables	Δcll^{AIC}
1	(1,2)	-196.62	(1,3)	-410.33
	(1,4)	-329.23	(1,5)	-255.28
	(1,6)	-35.83	(1,7)	-859.10
	(1,8)	-1931.48		
2	**(1,8,2)**	-186.21	(1,8,3)	-46.43
	(1,8,4)	-112.00	(1,8,5)	-71.46
	(1,8,6)	-40.45	(1,8,7)	-50.76
3	(1,8,2,3)	-60.71	(1,8,2,4)	-100.87
	(1,8,2,5)	-22.96	(1,8,2,6)	-27.51
	(1,8,2,7)	-122.36		
4	(1,8,2,7,3)	-15.47	(1,8,2,7,4)	-21.15
	(1,8,2,7,5)	-57.35	(1,8,2,7,6)	-56.93
5	(1,8,2,7,5,3)	-23.91	**(1,8,2,7,5,4)**	-119.50
	(1,8,2,7,5,6)	-19.21		
6	(1,8,2,7,5,4,3)	-10.04	**(1,8,2,7,5,4,6)**	-131.03
7	**(1,8,2,7,5,4,6,3)**	-129.79		

Table 5.2: Forward selection by R-vine copula model. In each step the chosen variable is printed in bold. The final R-vine structure has the order of variables (1,8,2,7,5,4,6,3).

	True	Linear	Gaussian	D-vine	R-vine
AD					
p-value	0.2714	< 2.2E-16	< 2.2E-16	2.471E-3	3.241E-4
Shapiro					
p-value	0.1853	< 2.2E-16	< 2.2E-16	8.303E-3	2.257E-4

Table 5.3: The result of AD and Shapiro tests for normality of the transformed pseudo observation from the true model, linear model, Gaussian copula model, D-vine copula model and R-vine copula model.

We also compare the quantile prediction performance of all presented models. To measure the predictive performance at the α quantile of the response variable l_0 given the covariates (denoted as set D) of the models we use the measure introduced in Koenker & Machado (1999) which we call the mean quantile error (MQE):

$$MQE_{l_0|D}(\alpha) = 1 - \frac{\sum_{m=1}^{N} \rho_\alpha(x_{l_0}^m - \hat{q}_{l_0|D}(\alpha))}{\sum_{m=1}^{N} \rho_\alpha(x_{l_0}^m - \hat{q}_{l_0}(\alpha))},$$

where $\rho_\alpha(v) = (\alpha - \mathbb{1}_{v\leq 0})v$ is the check function and $\hat{q}_{l_0|D}(\alpha)$ is the estimated quantile for the given covariates values $x_{l_i}^m$ $(i \geq 1)$ in D whereas $\hat{q}_{l_0}(\alpha)$ is the empirical quantile for the response variable.

In Table 5.4 the results of MQE calculated at quantiles $(0.01, 0.05, 0.5, 0.95, 0.99)$ for the true conditional distribution and the estimated conditional distributions are presented. We notice an asymmetry of performance of the five models with respect to MQE which is caused by the fact that the marginal distribution of the response variable follows

an exponential distribution. The linear regression model is the worst one among these models. The Gaussian copula model behaves a bit better than the linear but worse than the other two vine models especially at low quantiles. The D-vine based forward regression model and our heuristic R-vine perform quite similarly, though the R-vine model gives a smaller cll^{AIC}.

	0.01	0.05	0.5	0.95	0.99
True	0.7483	0.8015	0.9007	0.9311	0.9404
Linear	-0.2557	0.0838	0.5213	0.5319	0.4318
Gaussian	0.3002	0.4470	0.6776	0.6883	0.6682
D-vine	0.4761	0.5459	0.7019	0.7014	0.6731
R-vine	0.3272	0.5049	0.7149	0.7283	0.7147

Table 5.4: The in sample mean quantile error (MQE) at different quantiles (0.01,0.05,0.5,0.95,0.99) for the true conditional distribution and the estimated conditional distribution including the linear model, Gaussian copula model, D-vine copula model, and R-vine copula model.

5

Furthermore we apply an out of sample validation where another 10 data sets are generated from the true model and the range and the average (denoted in the brackets) of the MQE at different quantiles are calculated for all examined models. The results are presented in Table 5.5.

		0.01	0.05	0.5	0.95	0.99
True	$rng.$	0.7521~0.7835	0.8052~0.8266	0.8959~0.9112	0.9264~0.9455	0.9209~0.9527
	$avg.$	(0.7644)	(0.8148)	(0.9044)	(0.9374)	(0.9418)
Linear	$rng.$	-0.4837~-0.1331	-0.0298~0.1399	0.4985~0.5584	0.5223~0.6017	0.3223~0.5047
	$avg.$	(-0.2469)	(0.0894)	(0.5243)	(0.5521)	(0.4138)
Gaussian	$rng.$	-0.1746~0.2767	0.3300~0.4607	0.6560~0.7066	0.6661~0.7742	0.5120~0.7706
	$avg.$	(0.1165)	(0.4103)	(0.6852)	(0.7192)	(0.6692)
D-vine	$rng.$	0.1671~0.4548	0.4556~0.5421	0.6750~0.7104	0.6434~0.7622	0.5086~0.7485
	$avg.$	(0.2916)	(0.5028)	(0.6941)	(0.7061)	(0.6522)
R-vine	$rng.$	0.0447~0.4277	0.4137~0.5598	0.6847~0.7280	0.6724~0.7811	0.5590~0.8094
	$avg.$	(0.2702)	(0.4946)	(0.7095)	(0.7305)	(0.6920)

Table 5.5: Range and average of mean quantile error (MQE) at different quantiles (0.01,0.05,0.5,0.95,0.99) in 10 out of sample validation for the true conditional distribution and the estimated conditional distribution including the linear model, Gaussian copula model, D-vine copula model and R-vine copula model.

The predictive performance for the estimated models in the out of sample validation is comparable. The variability of the MQE at quantile 0.01 out of sample is larger than for other quantiles.

A further comparison for the D-vine and R-vine based regression model is shown in the simulation study in Section 5.4. We finish this section by stating the advantages of using vine based forward regression, improvements made by taking a R-vine structure in the forward selection as compared to D-vine, and some disadvantages which could be addressed in the future.

Advantages and improvements

- The vine based regression can capture different types of tail behaviors in the conditional distribution with the help of different bivariate copula families.
- Some covariates can be removed in the forward selection if they do not improve the performance of the conditional distribution.
- The conditional distribution is relatively easy to estimate since the response variable always appears in the conditioned set of the edge in the last tree.
- Each time when adding in a new variable one only needs to estimate those new copulas whose conditioned set contains the new variable.
- R-vine based regression allows flexibility in choosing the vine structure. This is helpful in 1) improving the estimation of the conditional distribution; 2) correctly removing variables that do not improve the conditional distribution.

Disadvantages

- Forward section does not guarantee that the true model will be found. In the example, the true model is a D-vine and the D-vine based procedure finds one with different order.
- Though forward selection is a good approach because of the easy application and of allowing removing variables that do not contribute to the conditional distribution, there is no guarantee that all important covariates are included.
- During the tree-wise estimation of the vine model, mis-specification of bivariate copula may occur due to the limited choice of parametric copula families. This could be improved by using non-parametric copulas in our approach.
- D-vine as well as R-vine regression procedure is based on simplified vines to ease the parameter estimation complexity of the vine model. However, this assumption might be too restrictive to lead to a good estimation as shown in the example. It is still an open question how to alleviate the effect of this assumption.

5.4. SIMULATION

In this section we perform a further comparison of D-vine and R-vine forward selection procedures via a simulation study. Similarly to the example in Section 5.3.3, a random regular vine copula model is generated by randomly choosing a vine structure. Its bivariate copula families are chosen from Gumbel(G), Clayton(C) and Joe(J) copulas as well as their rotated versions. The copula parameters are transformed from Kendall's tau which are simulated from a *Beta(2,2)* distribution, and are allowed to take negative values with probability 0.5. Since we only compare our heuristic with the D-vine model all margins are set to be standard uniformly distributed. The dimension of the random regular vine model is chosen as 10, 15 and 20, respectively.

We estimate the conditional distribution based on the training data set of size N_{train} and compare their behaviour in sample, hence using the training data, and out of sample on a validation data set of size $N_{val} = N_{train}$. The data size taken into account is 300 or 1000.

To see if there is a difference in the performance of the fitted models when the response variable is in the conditioned set or the conditioning set of the true model two situations are examined. We denote as Resp1 the case when the response variable is chosen randomly from the conditioned set and as Resp2 when is picked at random from the conditioning set of the edge in the last tree of the vine structure.

The comparison of the R-vine and the D-vine model is made using the following criteria: the MQE at quantiles (0.01,0.05,0.5,0.95,0.99) and the cll^{AIC} of the conditional distribution both in the training and validation data sets. Due to the complexity of simulations we only use one validation data set for one random vine copula model. This procedure is repeated 100 times. The results are shown in Table 5.6[1] for cll^{AIC} and in Table 5.7 for MQE. In the case of Resp1, the MQE for the true model is also given in Table 5.23 in Appendix 5.A.1. As for Resp2, the true conditional distribution is not available in an analytic form hence MQE results for the true model are not available.

			Resp1				Resp2			
			In Sample		Out of Sample		In Sample		Out of Sample	
Size	Dim		$\#imp$	ΔAIC_{imp}	$\#imp$	ΔAIC_{imp}	$\#imp$	ΔAIC_{imp}	$\#imp$	ΔAIC_{imp}
300	10	Dv	51	34.09	39	48.65	47	53.53	46	56.15
		Rv	49	48.89	61	42.44	53	60.35	54	65.91
	15	Dv	50	38.89	48	42.19	45	36.49	42	55.82
		Rv	50	45.53	51	59.92	55	49.56	58	58.39
	20	Dv	42	35.11	44	43.12	45	32.61	45	43.80
		Rv	58	43.17	56	57.92	55	47.98	55	54.34
1000	10	Dv	43	190.80	48	176.61	44	189.15	40	189.66
		Rv	57	175.40	52	207.31	56	218.96	60	220.38
	15	Dv	55	138.62	53	133.79	53	181.96	55	183.63
		Rv	45	183.40	47	194.22	47	163.30	45	161.49
	20	Dv	43	156.33	42	154.67	44	125.60	45	136.01
		Rv	57	192.77	58	189.99	56	198.06	55	202.00

Table 5.6: The simulation results for the D-vine based model (Dv) and the R-vine based model (Rv). The response variable for the conditional distribution is chosen by either Resp1 or Resp2. $\#imp$ denotes the number of times the given model has smaller cll^{AIC} than the other model and ΔAIC_{imp} is the average of the difference of cll^{AIC} when the given model has smaller cll^{AIC} then the other model. The results are shown for 300 and 1000 data sizes and for dimensions 10, 15 and 20.

[1] In Table 5.6 for 300 data size in dimension 15, it happened once that the cll^{AIC} was not available for out of sample validation. This is because the function `BiCopPDF` in the package **VineCopula** only allows Gumbel copula parameter to be in (0,50] but in the estimation it is 63.3.

			Resp1					Resp2				
Size	Dim		0.01	0.05	0.5	0.95	0.99	0.01	0.05	0.5	0.95	0.99
300	10	Dv	0.5820	0.6778	0.7782	0.6732	0.5694	0.6574	0.7344	0.8116	0.7189	0.6285
			(0.5024)	(0.6435)	(0.7566)	(0.6349)	(0.4989)	(0.5531)	(0.6984)	(0.7998)	(0.7001)	(0.5873)
		Rv	0.5852	0.6844	0.7785	0.6739	0.5678	0.6577	0.7349	0.8125	0.7201	0.6299
			(0.5241)	(0.6528)	(0.7568)	(0.6273)	(0.4814)	(0.5877)	(0.7056)	(0.8023)	(0.6993)	(0.5865)
	15	Dv	0.6037	0.7105	0.8060	0.7096	0.6106	0.6868	0.7665	0.8415	0.7589	0.6774
			(0.5451)	(0.6825)	(0.7924)	(0.6789)	(0.5378)	(0.6341)	(0.7433)	(0.8272)	(0.7271)	(0.6165)
		Rv	0.6134	0.7146	0.8038	0.7108	0.6054	0.6972	0.7683	0.8399	0.7592	0.6783
			(0.5596)	(0.6833)	(0.7908)	(0.6880)	(0.5504)	(0.6300)	(0.7422)	(0.8264)	(0.7299)	(0.6163)
	20	Dv	0.6339	0.7200	0.8074	0.7135	0.6177	0.6708	0.7480	0.8291	0.7452	0.6597
			(0.5569)	(0.6813)	(0.7890)	(0.6773)	(0.5395)	(0.6001)	(0.7172)	(0.8123)	(0.7073)	(0.5734)
		Rv	0.6458	0.7254	0.8101	0.7215	0.6292	0.6708	0.7481	0.8299	0.7481	0.6641
			(0.5831)	(0.6905)	(0.7947)	(0.6940)	(0.5727)	(0.5975)	(0.7168)	(0.8138)	(0.7116)	(0.5797)
1000	10	Dv	0.5995	0.6965	0.7854	0.6847	0.5885	0.7068	0.7816	0.8512	0.7732	0.6937
			(0.5897)	(0.6906)	(0.7830)	(0.6772)	(0.5736)	(0.6813)	(0.7701)	(0.8448)	(0.7614)	(0.6715)
		Rv	0.6021	0.6943	0.7839	0.6826	0.5868	0.7099	0.7817	0.8483	0.7733	0.6989
			(0.5900)	(0.6890)	(0.7804)	(0.6767)	(0.5730)	(0.6869)	(0.7731)	(0.8438)	(0.7650)	(0.6820)
	15	Dv	0.6442	0.7319	0.8151	0.7304	0.6399	0.6733	0.7518	0.8261	0.7465	0.6637
			(0.6154)	(0.7174)	(0.8061)	(0.7151)	(0.6162)	(0.6533)	(0.7422)	(0.8230)	(0.7417)	(0.6502)
		Rv	0.6390	0.7276	0.8128	0.7263	0.6347	0.6685	0.7449	0.8213	0.7452	0.6693
			(0.6178)	(0.7164)	(0.8057)	(0.7119)	(0.6101)	(0.6431)	(0.7357)	(0.8176)	(0.7398)	(0.6572)
	20	Dv	0.6535	0.7391	0.8211	0.7370	0.6464	0.6756	0.7633	0.8407	0.7646	0.6845
			(0.6311)	(0.7285)	(0.8143)	(0.7249)	(0.6246)	(0.6545)	(0.7522)	(0.8348)	(0.7523)	(0.6554)
		Rv	0.6484	0.7388	0.8188	0.7334	0.6451	0.6795	0.7622	0.8397	0.7649	0.6855
			(0.6312)	(0.7304)	(0.8134)	(0.7227)	(0.6306)	(0.6561)	(0.7496)	(0.8339)	(0.7541)	(0.6643)

Table 5.7: The simulation results for the average MQE at quantiles (0.01,0.05,0.5,0.95,0.99) over 100 repetitions. The results are shown for D-vine model (Dv) and R-vine model (Rv), for data sets of size 300 and 1000, in sample and out of sample (in brackets), for dimensions of 10, 15 and 20.

In general the conditional distribution estimated by our heuristic R-vine based forward regression behaves quite similarly to the one estimated by the D-vine procedure. The number of times where one model behaves better than the other is around 50 and the average improvement in cll^{AIC} is also quite similar, though a bit better for R-vine. The MQE results for these two models do not show significant differences for all examined quantile levels. We do not observe significantly different behaviour of the models for different response variables (cases Resp1 and Resp2). Additionally we have collected information about the number of covariates used in each model (picked by the forward selection). Also from this perspective these two models do not differ much as can be observed in Table 5.8.

Few reasons for the similarities of the behaviour for both models could be named. Our heuristic is based on the partial correlations hence an assumption that all new copulas are Gaussian copulas, which could be too restrictive. However, the real problem is that the first two steps of the R-vine and D-vine forward selection procedures are the same. Different vine structures can be chosen only when the third covariate is added. The forward selection picks the 'most important' covariates earlier in the process. We observed that the first two steps cover at least 71.47% and 64.81% of the whole cll^{AIC}

Size	Dim		Resp1		Resp2	
			$avg.\#cov$	$\#cov_{more}$	$avg.\#cov$	$\#cov_{more}$
300	10	Dv	5.66	28	5.88	28
		Rv	5.82	39	5.87	33
	15	Dv	6.81	45	6.65	31
		Rv	6.58	29	6.86	47
	20	Dv	6.99	34	6.98	27
		Rv	7.02	39	7.16	34
1000	10	Dv	7.42	31	7.52	35
		Rv	7.47	33	7.49	34
	15	Dv	9.66	47	9.25	33
		Rv	9.06	25	9.30	37
	20	Dv	10.26	37	10.45	33
		Rv	10.10	39	10.35	34

Table 5.8: The number of covariates in the forward selection for the D-vine (Dv) and R-vine (Rv) based model. The average number of included covariates is denoted as $avg.\#cov$ and $\#cov_{more}$ represents the number of times where one model contains more covariates than the other. The result is shown for data sizes 300 and 1000 and for dimensions 10, 15 and 20.

for 300 and 1000 data size, respectively. Hence the benefit of allowing for different vine structures is limited. This was also concluded in the simulation study in Herrmann (2018).

In Section 5.6, we will explain how the method introduced in Zhu et al. (2020) can provide a way to further improve the estimation once the model is given by the forward selection.

5.5. HEURISTIC FORWARD SELECTION FOR MORE RESPONSE VARIABLES

In Section 5.3.2 a forward selection procedure to obtain the conditional distribution of one response variable based on a R-vine structure has been presented. In the case of multiple response variables, the joint conditional distribution will be of interest. A naive solution could be to first estimate the conditional margins respectively using the heuristic forward procedure and then to estimate the conditional copula for the pseudo observations obtained by transforming the data through the estimated conditional margins. There are two problems with this approach 1) it can happen that different covariates are chosen by the forward selection for each margin (though we can regard those missing covariates as being (conditionally) independent of the response variable); 2) the variable selection is applied to each response variable separately rather than to the joint conditional distribution.

In this section we show how the R-vine based forward selection procedure for one response variable can be extended to allow more response variables. The detail procedure will be discussed for two response variables and a short discussion of extending this method to more than two variables will be given at the end of the section.

The process of R-vine based forward selection for two response variables is similar

to the case of one response variable. The R-vine structure for a given order can be determined by a sequence of indicators and a substructure in VBT where the order is the order of newly added variables from the forward selection. However, for two response variables, to be able to obtain the joint conditional distribution of the response variables in analytic form additional constraints on the vine structures are needed. For instance, suppose we want to estimate the joint conditional density $f_{14|235}$ for the variables X_1 and X_4 given X_2, X_3, X_5 using $V_1(5)$ in Figure 5.1 (left). Then the conditional density can be decomposed as the product of copula densities assigned to the edges of $V_1(5)$ and is equal:

$$f_{14|235} = f_{1|2345} f_{4|235} = f_1 c_{12} c_{13;2} c_{14;23} c_{15;234} f_4 c_{24} c_{34;2} c_{45;23}.$$

However, the conditional distribution $F_{14|235}$ has to be computed through integration when decomposition based on $V_1(5)$ is used. This can be achieved either by integration of the conditional density or by computing the conditional distributions $F_{1|2345}$ and $F_{4|235}$ (which are $C_{1|2345}$ and $C_{4|235}$ when margins are transformed into u-scale, and they are represented by the corresponding h-functions, hence are directly computed from copulas in $V_1(5)$ model), then integrating up to the given value x_4 of argument in the function $F_{1|2345}$ weighted with density $f_{4|235}$.

In order to get the conditional distribution of the two response variables directly, we will construct a vine structure such that the two response variables are exactly the conditioned set of the edge in the last tree. For example, the conditional distribution $F_{15|234}$ can be easily estimated based on $V_1(5)$,

$$F_{15|234} = C_{15;234}(F_{1|234}, F_{5|234}).$$

In terms of our procedure this requirement boils down to ensuring that one response variable is in the first place in the given order and the other response variable is the last element in the order in each forward step. A detail example for the construction of the vine structure is as follows. Suppose the order of variables is $(l_0^1, l_1, l_2, l_3, l_0^2)$ where l_0^1 and l_0^2 represent the two response variables and l_1, l_2, l_3 are the covariates. Then in each step of the forward selection, we have a VBT as shown in Figure 5.8 and Figure 5.9. Since the forward selection concerns only the dependence structure the margins are transformed into their u-scale.

In the first step when l_1 is added the vine structure is completely determined by the order and the conditional density is $f_{l_0^1 l_0^2|l_1} = c_{l_0^1 l_1} c_{l_0^1 l_0^2;l_1} c_{l_0^2 l_1}$. In the second step, we have a choice for the value in VBT[2][2] (in black circle in Figure 5.8) which can be either l_2 if indicator is 1 or l_1 if indicator is 0. The conditional density $f_{l_0^1 l_0^2|l_1 l_2}$ can then be decomposed as either $c_{l_0^1 l_1} c_{l_0^1 l_2;l_1} c_{l_0^1 l_0^2;l_1 l_2} c_{l_0^2 l_1} c_{l_0^2 l_2;l_1}$ or $c_{l_0^1 l_1} c_{l_0^1 l_2;l_1} c_{l_0^1 l_0^2;l_1 l_2} c_{l_0^2 l_2} c_{l_0^2 l_1;l_2}$. Note that if we choose l_2, then we do not need to re-estimate the copula $c_{l_0^2 l_1}$ and only copulas $c_{l_0^2 l_2;l_1}$ and $c_{l_0^1 l_0^2;l_1 l_2}$ have to be estimated. If l_1 is chosen then $c_{l_0^2 l_2}$ and $c_{l_0^2 l_1;l_2}$ and of course $c_{l_0^1 l_0^2;l_1 l_2}$ have to be estimated.

Suppose in the second step we choose l_2, then in the third step in Figure 5.9, we have two unspecified elements in VBT where 2 choices are possible, which are VBT[2][2] and VBT[3][4]. Again choices of VBT[2][2] to be l_3 and VBT[3][4] to be l_2 will not need

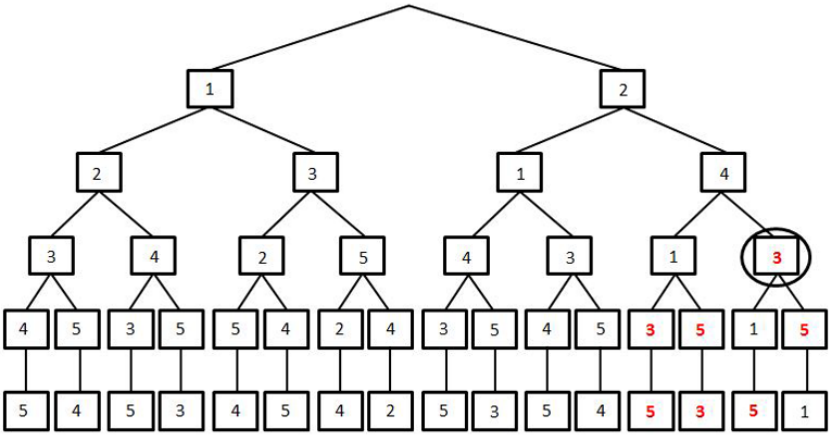

Figure 5.8: The first step (left) and the second step (right) in the forward selection for two response variables. The substructure in the left VBT corresponding to the given order (l_0^1, l_1, l_0^2) and the one in the right corresponding to the given order (l_0^1, l_1, l_2, l_0^2) are in dashed area. The order of variables in these substructures are in red squares. The element in VBT where we have 2 choices is included in the black circle in the right VBT.

to re-estimate copulas that have been previously estimated. The conditional density decomposition is then

$$f_{l_0^1 l_0^2 | l_1 l_2 l_3} = c_{l_0^1 l_1} c_{l_0^1 l_2; l_1} c_{l_0^1 l_3; l_1} c_{l_0^1 l_0^2; l_1 l_2 l_3} c_{l_0^2 l_1} c_{l_0^2 l_2; l_1} c_{l_0^2 l_3; l_1 l_2}.$$

We can see that only $c_{l_0^1 l_3; l_1}$, $c_{l_0^1 l_0^2; l_1 l_2 l_3}$ and $c_{l_0^2 l_3; l_1 l_2}$ have to be additionally estimated. Though in terms of computational complexity the above choice is beneficial but it leads to fixing the structure to a C-vine with order $(l_0^1, l_1, l_2, l_3, l_0^2)$ according to Proposition 2.3.2 (the indicators are always chosen to be 1). In order to have more flexibility of possible vine structures, we will not adopt this computationally efficient construction in this chapter.

In the forward selection procedure above we first pick the covariate l_1 to form a vine with order (l_0^1, l_1, l_0^2), which is equivalent with the choice based on subvine (l_0^1, l_1) as for one response variable, extended by response variable l_0^2. Similarly, to pick l_2 vine structures based on subvine (l_0^1, l_1, l_2) are constructed and extended by l_0^2 etc. Thus the construction of vine structure in each step of forward selection is as follows: for each new variable candidate we construct a vine structure similarly to the case when l_0^1 is the single response variable shown in Section 5.3.2, then extend this vine by adding l_0^2 and unspecified elements in VBT are determined by the heuristic presented in Section 5.3.2.

Note that in the procedure above two response variables are treated a bit differently. In a sense the subvine structure not containing variable l_0^1 is more constrained by the forward selection of newly added covariates while when extending the structure with l_0^2 more choices of different vine structures are possible. This leads to a certain asymmetry in treatment of the response variables and can lead to different performance of the regression model when the order of response variables is exchanged. This behaviour can be observed in the numerical example presented later on where both orders of response

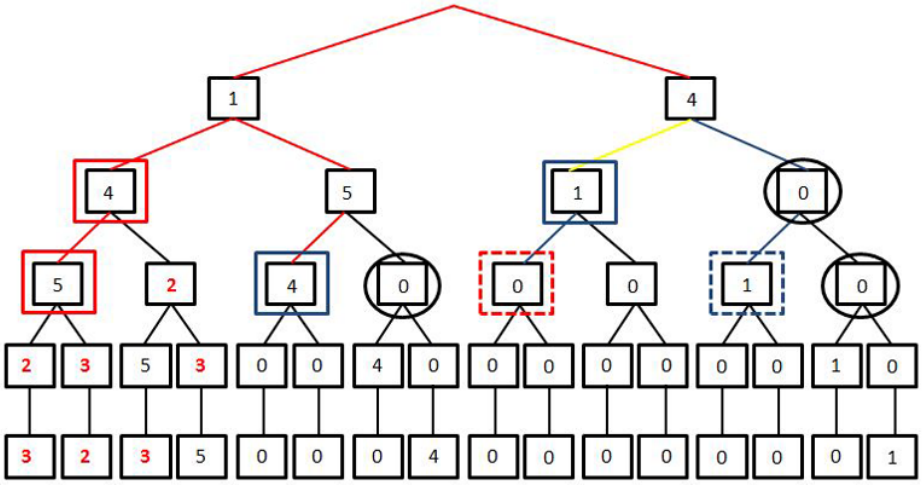

Figure 5.9: The third step in the forward selection. The substructure corresponding to the given order $(l_0^1, l_1, l_2, l_3, l_0^2)$ is in the dashed area and the order is in red squares. The elements in VBT where we have 2 choices are denoted by the black circle (the value in VBT[3][6] has been determined in the second step).

variables are used to build a regression model. For just two response variables, when not too many covariates are of interest, it is possible to fit both models. In general, however, checking all possible combinations is prohibitive. Hence we propose to determine the order of response variables in our procedure by picking l_0^1 as the response variable which is the most correlated, measured by empirical multiple correlation Yule & Kendall (1965), with the remaining response variables and covariates.

Heuristic - multiple response

The order of the response variables is determined by the decreasing order of empirical multiple correlations[2] *for a response variable based on the remaining response variables and covariates.*

For the same data set analyzed in Section 5.3 where margins are transformed into u-scale by kernel smoothing, we apply our heuristic to model the conditional distribution $F_{12|D}$ by a forward selection approach. The estimated multiple correlations are: $R^2_{1\{23...8\}} = 0.8952$ and $R^2_{2\{13...8\}} = 0.7805$ (the empirical correlation matrix of normal scores for the data is in Table 5.24 in Appendix 5.A.1). Thus (l_0^1, l_0^2) (order of response variables) is (1,2). However, for this example we have estimated models when both orders are used for comparison. The detailed steps of our approach are shown in Table 5.9 and the final vine copula model is in Table 5.25 in Appendix 5.A.1.

[2]The empirical multiple correlation $R^2_{i\{P\}}$ is computed from empirical correlation matrix P of normal scores by $R^2_{i\{P\}} = 1 - \frac{det(P)}{det(P^{ii})}$, where P^{ii} denotes the matrix P without the ith row and the ith column.

Forward Selection (R-vine copula with response variables 1,2)				
step	order of variables	Δcll^{AIC}	order of variables	Δcll^{AIC}
1	(1,3,2)	-1241.33	(1,4,2)	-661.74
	(1,5,2)	-792.52	(1,6,2)	-473.01
	(1,7,2)	-1161.53	**(1,8,2)**	-2249.03
2	**(1,8,3,2)**	-875.78	(1,8,4,2)	-179.38
	(1,8,5,2)	-348.15	(1,8,6,2)	-275.56
	(1,8,7,2)	-275.84		
3	(1,8,3,4,2)	-597.40	(1,8,3,5,2)	-796.46
	(1,8,3,6,2)	-930.52	(1,8,3,7,2)	-7.68
4	**(1,8,3,6,4,2)**	-235.87	(1,8,3,6,5,2)	-67.37
	(1,8,3,6,7,2)	-143.28		
5	**(1,8,3,6,4,5,2)**	-103.68	(1,8,3,6,4,7,2)	-34.21
6	**(1,8,3,6,4,5,7,2)**	-11.60		

Table 5.9: Forward selection R-vine copula regression model for two response variables ordered as (1,2). In each step the chosen variable is printed in bold. The final R-vine structure has the order of variables (1,8,3,6,4,5,7,2).

5

Details for the model with the order of the response variables being (2,1) and R-vine models for one response variable (either variable 1 or variable 2) are given in the 5.A.1. Next, we compare the performance of the R-vine based forward approach with both orders of response variables to a naive approach, called Separate or SeparateAll. In model Separate the forward selection process is carried out for both margins separately, hence different variables could have been removed by the selection process. The joint conditional distribution of the response variables is then estimated by a copula from the pseudo observations obtained via applying estimated conditional margins. As for the SeparateAll model all the covariates are kept during the forward selection to the marginal conditional distributions. The cll^{AIC} (excluding margins) for the estimated joint conditional distribution from these four methods and the true model are given in Table 5.10.

Furthermore, we compare the performance of these four models using the Brier skill score. The Brier score introduced in Brier (1950) is simply the average of the squared difference between the probability of an event occurrence estimated by a model and whether this event happens (which is 1 if it happens otherwise it is 0). Hence we compute the average of the squared difference between the probability of an event occurring conditioned on the given value of covariates and whether this event happens, and then divide by the average of the squared difference of the unconditional probability and the dichotomous outcome. If this ratio is smaller than one then the estimated conditional model predicts the required event better than the unconditional model (taken as reference model). Subtracting this ratio from one gives the Brier skill score presented below for response variables (l_0^1, l_0^2) with covariates set D and required event $P(l_0^1 \leq t_1, l_0^2 \leq t_2)$

$$BS_{l_0^1 l_0^2}(t_1,t_2) = 1 - \frac{\sum_{m=1}^{N} (\mathbb{1}_{x_{l_0^1}^m \leq t_1, x_{l_0^2}^m \leq t_2} - P_{l_0^1 l_0^2|D}(t_1,t_2))^2}{\sum_{m=1}^{N} (\mathbb{1}_{x_{l_0^1}^m \leq t_1, x_{l_0^2}^m \leq t_2} - P_{l_0^1 l_0^2}(t_1,t_2))^2},$$

where $P_{l_0^1 l_0^2}$ and $P_{l_0^1 l_0^2|D}$ denote the probability $P(l_0^1 \leq t_1, l_0^2 \leq t_2)$ computed from unconditional and conditional model, respectively. t_1 is the threshold for the response variable l_0^1 and t_2 is the threshold for l_0^2 in the event.

In Table 5.10 the Brier skill score results are presented for the four estimated models as well as the true model. The thresholds t_1 and t_2 are chosen to be the empirical quantiles of the marginal distributions of l_0^1 and l_0^2 (transformed by kernel smoothing) at level $(0.1, 0.2, 0.3, \ldots, 0.9)$, respectively (the same quatiles for both thresholds are chosen).

		Threshold levels								
	cll^{AIC}	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
True	-8256.87	0.7658	0.8216	0.7782	0.8138	0.7797	0.8024	0.7807	0.7655	0.7406
R-vine(1,2)	-4406.48	0.6664	0.6993	0.6976	0.7008	0.6589	0.6853	0.6618	0.5941	0.5467
R-vine(2,1)	-3999.70	-0.8492	-0.1105	0.0597	0.2123	0.2767	0.3529	0.3539	0.3152	0.2434
Separate	-4386.60	0.6186	0.6878	0.7024	0.7212	0.6632	0.6935	0.6520	0.5956	0.5688
SeparateAll	-4371.36	0.5981	0.6873	0.6958	0.7209	0.6686	0.6960	0.6541	0.6006	0.5702

Table 5.10: The AIC for the conditional distribution $C_{12|D}$ and the Brier skill score for different threshold levels are listed. They are estimated by different approach: R-vine (1,2) represents the heuristic approach with (1,2) as the order of response variables whereas R-vine (2,1) is with the order of (2,1). Separate is the model where the conditional margins are found using the heuristic in Section 5.3.2 and then the conditional copula was estimated from pseudo-observations. SeparateAll in contrast to Separate does not allow any variable to be removed in estimating conditional margins.

The R-vine model obtained with our heuristic, R-vine(1,2), has the smallest cll^{AIC}. Its Brier skill score is much larger than other estimated models in the case of threshold corresponding to 0.1 quantiles of the response variables. For other quantile levels results of Brier skill score are comparable for R-vine(1,2) as well as Separate and SeparateAll. The R-vine model with different order of the response variables, R-vine(2,1), shows the worst performance of all estimated models which confirms the choice made by our heuristic. As compared to the true model all four estimated models perform poorly. This is also confirmed by the results of the bivariate KS goodness-of-fit test Fasano & Franceschini (1987), where the assumption that the data is a random sample from the estimated model is strongly rejected for all these four methods (with a maximum p-value $3.9649E-10$ for the R-vine model from our heuristic approach).

The extension of the forward selection regression to more than two response variables leads to a few complications. In the case of $s > 2$ response variables, $(l_0^1, \ldots, l_0^s)$, there are $s!$ different orders for the response variables that could be considered. The heuristic based on the empirical multiple correlation explained above can be employed to find such an order.

Though the conditional density can be computed analytically for subsets of vine structures, the analytic form of the conditional distribution is hardly available and has to be computed by integration. For example, suppose we have three response variables (l_0^1, l_0^2, l_0^3) and we have already estimated a vine structure where we have copula $C_{l_0^1 l_0^3; l_0^2 \cup D}$ for the edge in the last tree, then the joint conditional distribution can only be computed through the integration up to given value $x_{l_0^2}$ of argument in $F_{l_0^1 l_0^3 | l_0^2 \cup D}$ weighted with density $f_{l_0^2|D}$.

If there are s response variables then at least $s-2$ dimensional integral will need

to be solved. In this case characteristics of interest from the required joint conditional distribution of the response variables can be obtained via simulations.

5.6. EXTENSION STUDY BY SEARCHING FOR VINES HAVING 2 SAMPLING ORDERS IN COMMON

As shown in the example in Section 5.3.3, the R-vine based forward regression procedure is not able to overcome some disadvantages caused by the application of forward selection and the assumption of simplified vine copula. Rather than adopting other variable selection methods or allowing conditional copulas in our model to depend on the conditioning variables, we opt here to use the heuristic studied in Zhu et al. (2020) (also presented in Chapter 3) to improve the estimation of the conditional distribution. This heuristic suggests to start with an estimated initial vine structure and then to search for several regular vines that have 2 common sampling orders with the initial structure. The number of common sampling orders of two vine structures correlates with the number of common bivariate (un)conditional copulas in their copula decomposition. Hence the idea of the heuristic presented in Zhu et al. (2020) is to consider vine structures that are not 'similar' to the initially estimated vine. The set of vines having 2 common sampling orders have been chosen because it is sufficiently large (but not too large compared to the number of all vines) and contains vine structures that are in principle quite different than the initial vine. Another reason to choose this set of vines is the existence of an algorithm to generate random structures from this set, see Algorithm 6 in the Appendix 3.A.2.

Once we have estimated the regular vine model from the R-vine based forward regression, we can search for several regular vine structures that have 2 common sampling orders with this vine and choose the one that gives us the best fitting conditional distribution. The algorithm to find vines having 2 sampling orders in common has to be modified slightly as in the R-vine based forward regression we require the response variable to appear in the conditioned set of the edge in the last tree of the vine.

The algorithm developed in Zhu et al. (2020) can be easily adapted to include this requirement and one can calculate that there are $3^{(n-3)}2^{\binom{n-2}{2}}$ regular vines having 2 sampling orders in common with the initial vine and one common variable (the response variable) in the conditioned set of the edge in the last tree.

As suggested in Zhu et al. (2020), we have chosen to search for 10 random vines having 2 sampling orders in common with the vine obtained in the R-vine based forward regression estimated in Section 5.3.3. Out of 10 vines having 2 sampling orders in common only one is better than the estimated R-vine model with cll^{AIC} equal to -2720.32 [3]. The true vine has 0 common sampling order with the R-vine based model hence it cannot be found by the above search.

Similarly, as in Section 5.3.3, we test the performance of the obtained model. The in sample validation Q-Q plot is shown in Figure 5.10. The p-values of AD test and Shapiro test are 0.1178 and 0.00217, and the in sample and out of sample results of MQE at quan-

[3] Additionally we also looked at 10 random vines generated with the algorithm in Joe et al. (2011) modified such that the response variable is in the conditioned set of the edge in the last tree. This procedure led to a vine with the smallest cll^{AIC} equal to -2508.34. Hence it validates the result shown in Zhu et al. (2020).

tiles (0.01,0.05,0.5,0.95,0.99) can be found in Table 5.11.

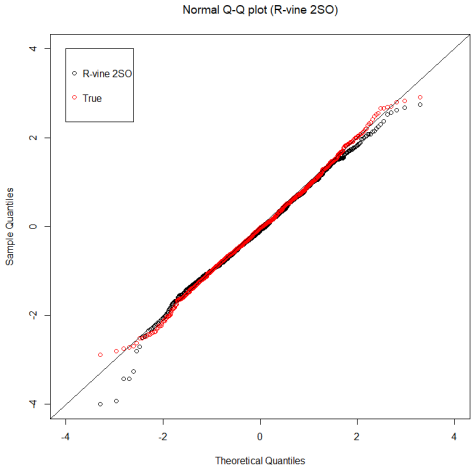

Figure 5.10: In sample validation Q-Q plot for the best model (smallest cll^{AIC}) we find in vines having 2 sampling orders (SO) in common with the R-vine model (black) in comparison with the true model (red).

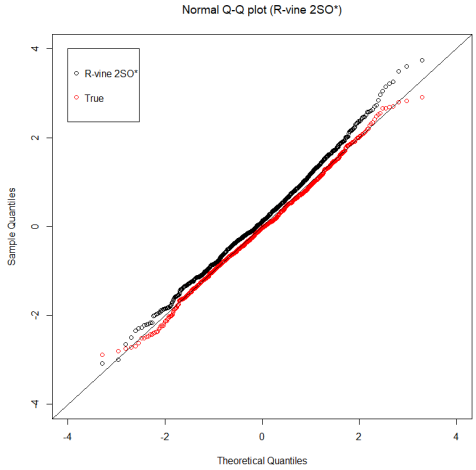

Figure 5.11: In sample validation Q-Q plot for the best model (largest MQE at quantile 0.01) we find in vines having 2 sampling orders (SO) in common with the R-vine model (black) in comparison with the true model (red).

		0.01	0.05	0.5	0.95	0.99
R-vine 2SO		0.2370	0.5237	0.7120	0.7612	0.7811
	$rng.$	-0.2276~0.4161	0.3563~0.5506	0.6581~0.7177	0.7136~0.7616	0.6601~0.7580
	$avg.$	(0.0780)	(0.4409)	(0.6934)	(0.7396)	(0.7282)
R-vine 2SO*		0.4513	0.5380	0.7013	0.7154	0.6881
	$rng.$	0.0253~0.3544	0.4116~0.5059	0.6717~0.7146	0.6581~0.7815	0.5066~0.8047
	$avg.$	(0.2375)	(0.4686)	(0.6944)	(0.7163)	(0.6689)

Table 5.11: In sample mean quantile error (MQE) and the range and average (in brackets) of out of sample MQE at different quantiles (0.01,0.05,0.5,0.95,0.99). The result is shown for R-vine 2SO which is the vine found by searching for vines having 2 sampling orders in common based on smallest cll^{AIC}, and for R-vine 2SO* which is found by largest MQE at quantile 0.01.

The vine structure obtained by searching for 2 common sampling orders provides a better model of the conditional distribution for the data. Predictions are good in the median and upper tails. However, the in-sample validation Q-Q plot in Figure 5.10 shows more deviations in lower tail. This behaviour is also visible in the range of MQE at quantile 0.01 for out of sample data sets which varies from -0.2276 to 0.4161.

Note that so far we have used cll^{AIC} as a measure of performance to decide which structure performs best. If the performance at a certain quantile, say 0.01, is of importance we can adjust our procedure and search for a vine that gives the best performance at this quantile level. This can be implemented in both R-vine regression forward selection as well as in the procedure that searches vines having 2 sampling orders in common.

The vine with the best performance at 0.01 quantile level in the 10 vines having 2 common sampling orders is denoted as R-vine 2SO* in Table 5.11 with $cll^{AIC} = -2456.75$ and MQE = 0.4513 at 0.01 quantile. We can also observe in Figure 5.11 that the in sample validation Q-Q plot for this new vine structure has much smaller deviation in lower tail.

5.7. REAL DATA ANALYSIS

In this section the techniques presented so far in this chapter are applied to perform a stress test analysis on the manufacturing industry. The manufacturing sector is a key sector in many economies and is involved in creating sustainable economic growth (see Behun et al. (2018)). This sector also faces challenges such as shifting and fragmenting demand, natural disasters (e.g. Thailand's flooding), government's action (e.g. tariff policy), shortage of high-skill workers etc. Hence it is important to assess the impact of the manufacturing industries on other sectors in economy in order to prevent a potential future crisis.

In our analysis we will use the data set from $KennethR.French - Datalibrary$ [4] which consists of 1122 monthly returns of 30 industry portfolios from 07-1926 to 12-2019. As in Kraus & Czado (2017*a*) we apply a stress scenario to the indices in the manufacturing sector to see how indices in other industries are affected. The industry sectors are formed by four digit SIC code according to Fama and French 30 industries classification, which are

- Manufacturing industry [5]: Chemicals (**Chems**), Construction &Construction Materials (**Cnstr**), Steel Works Etc (**Steel**), Fabricated Products &Machinery (**FabPr**), Electrical Equipment (**ElcEq**), Automobiles &Trucks (**Autos**), Aircraft, Ships &Railroad Equipment (**Carry**), Business Supplies &Shipping Containers (**Paper**).

- Remaining industries: Food Products (**Food**), Beer &Liquor (**Beer**), Tobacco Products (**Smoke**), Recreation (**Games**), Printing &Publishing (**Books**), Consumer Goods (**Hshld**), Apparel (**Clths**), Healthcare, Medical Equipment &Pharmaceutical Products (**Hlth**), Textiles (**Txtls**), Precious Metals, Non-Metallic &Industrial Metal Mining (**Mines**), Coal (**Coal**), Petroleum &Natural Gas (**Oil**), Utilities (**Util**), Communication (**Telcm**), Personal &Business Services (**Servs**), Business Equipment (**BusEq**), Transportation (**Trans**), Wholesale (**Whlsl**), Retail (**Rtail**), Restaurants, Hotels &Motels (**Meals**), Banking, Insurance Real Estate &Trading (**Fin**) and Everything Else (**Other**).

We follow the steps taken in Kurz (2019) and first filter the data by a ARMA(1,1)-GARCH(1,1) model with student-t innovation. The residuals are fitted by kernel smoothing and the fitted margins are then transformed to data in u-scale. The stress test can be applied on the transformed data only as discussed in Brechmann et al. (2013).

[4] *http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html*

[5] According to Fama and French 10 industries classification, most of the stocks in the manufacturing industry sector can be covered by the indices, Chems, Cnstr, Steel, FabPr, ElcEq, Autos, Carry and Paper, based on the four digit SIC code.

5.7.1. ONE RESPONSE VARIABLE CASE

In this section, we concentrate on the case where only one response variable is taken into account in the stress test. For each index in industries other than the manufacturing, we apply our heuristic R-vine regression to estimate the distribution of this index conditioned on the stressed indices in the manufacturing sector. The estimated quantiles of the conditional distribution when the manufacturing industry is stressed is shown as a bar plots in Figure 5.12 for stress level 0.05 (red) and for level 0.01 (black). Each bar represents the range between 0.1 and 0.9 quantile of the conditional distribution and the median of the conditional distribution is denoted as hollow dot (0.05 level) or square (0.01 level).

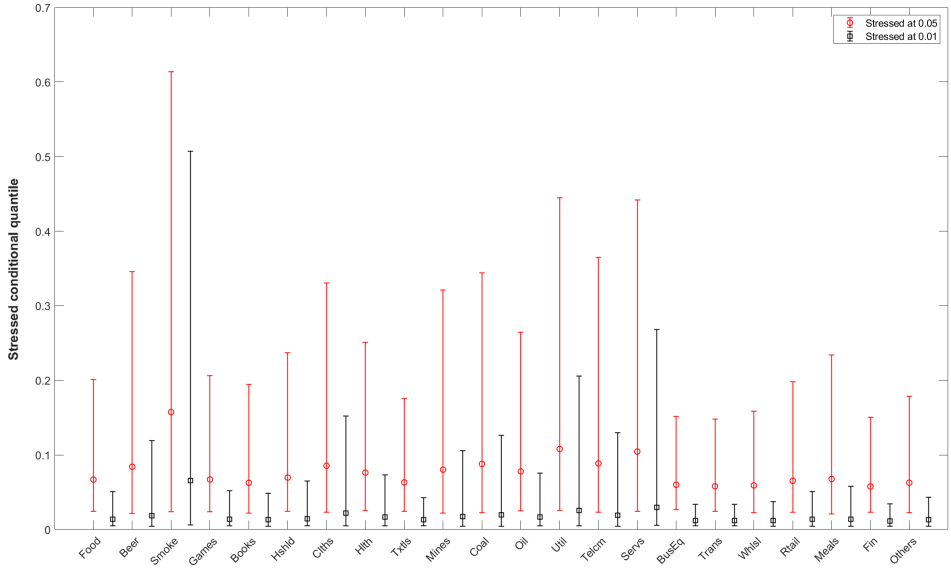

Figure 5.12: Results of the stress analysis per index when manufacturing industries are stressed at 0.05 (red) or 0.01 (black) level. The median of the conditional distribution is denoted as hollow dot (0.05 level) or square (0.01 level), and bars represent the range between its 0.1 and 0.9 quantile.

A few observations are worth pointing out. We see that the medians of most industries are close to the stress level of the manufacturing industries. Moreover, the quantile interval for stress level 0.01 is narrower as compared to the stress level 0.05. This indicates a strong impact of the manufacturing sector on the other industries.

We also observe that in our R-vine based forward procedure for each index the first chosen covariate is always the index Cnstr or the FabPr, except for the conditional distribution of Hlth (Paper being the first) and Mines, Coals (Steel being the first). This makes it clear that Cnstr and the FabPr are very important within the manufacturing sector for the well being of the economy. By contrast, the index Chems is retained in the regression model only for the following response variables: Hlth, Oil and Telcm. Hence problems in chemical industries do not affect the economy as much as the other manufacturing indices.

Smoke index is the least affected one by the stressed manufacturing sector as we see

in Figure 5.12. This conclusion could have been drawn also from exploring its empirical multiple correlation with manufacturing indices which was the smallest and equal to 0.2847. The Smoke industry includes companies which grow, deliver, advertise and sell tobacco and tobacco-related products. It is a rather self-contained industry hence the effect of the crisis in the manufacturing sector is limited.

Conclusions that can be drawn from exploring empirical multiple correlations do not always align with ones obtained from our regression analysis. For instance, the empirical multiple correlation for the Clths (0.5697) is larger than the one for the Coal (0.4455), but results plotted in Figure 5.12 show that when manufacturing sector is stressed at level 0.01 Coal is more affected than Cloth. This shows the importance of more thorough analysis.

In the next section, we explore how different indices are jointly affected by the stressed manufacturing sector.

5.7.2. TWO RESPONSE VARIABLES CASE

Our analysis can be carried out also to explore the joint behaviour of different indices when stress in manufacturing sector occurs by following the procedures explained in Section 5.5. We present results of bivariate response variables only, where we explore the behaviour of index Food with other indices when manufacturing sector is stressed. For each pair Food and Y index in remaining industry, their joint distribution conditioned on the manufacturing sector is computed. From these joint distributions we then compute the probability that the return of the Food index and Y index are both smaller or equal to given thresholds t (denoted as $P^t_{Food,Y}$) when the manufacturing industry is stressed at 0.05 or 0.01 level. We chose the threshold to be equal to their conditional median or 0.1 quantile ($t = 0.5, 0.1$) obtained in the stress test analysis in Section 5.7.1.

If the Food index is conditionally independent with the other index Y, then this probability should be 0.25 (0.5^2) when both thresholds are the median or $0.01 (0.1^2)$ when the thresholds are 0.1 quantile. The degree of the conditional dependence of the Food index and Y index when the manufacturing sector is stressed can be assessed by the quotient of the probability $P^t_{Food,Y}$ and t^2. The result is shown in Table 5.12 when $t = 0.5$ and in Table 5.13 when $t = 0.1$.

	0.05	0.01		0.05	0.01		0.05	0.01
Food, Beer	1.2434	1.2441	Food, Smoke	1.3494	1.4076	Food, Games	1.1296	1.1593
Books, Food	1.1415	1.1449	Hshld, Food	1.3456	1.3545	Food, Clths	1.3849	1.5012
Food, Hlth	1.1681	1.1599	Txtls, Food	1.1819	1.1926	Food, Mines	0.9750	0.9491
Food, Coal	0.9841	0.9551	Food, Oil	0.9801	0.9798	Food, Util	1.2836	1.2722
Food, Telcm	1.2841	1.2903	Food, Servs	1.3192	1.4505	BusEq, Food	1.0247	1.0261
Trans, Food	1.1618	1.1951	Whlsl, Food	1.1563	1.1645	Rtail, Food	1.2299	1.2305
Food, Meals	1.2903	1.2964	Fin, Food	1.0942	1.0978	Other, Food	1.1024	1.1060

Table 5.12: $P^t_{Food,Y}/0.5^2$ where the thresholds are chosen to be the median of the two indices in the corresponding stress scenario in Section 5.7.1. The order of the response variables based on our heuristic is also indicated.

	0.05	0.01		0.05	0.01		0.05	0.01
Food, Beer	2.4837	2.4938	Food, Smoke	3.7568	3.9389	Food, Games	1.2088	1.3182
Books, Food	1.3260	1.3779	Hshld, Food	2.7433	2.7826	Food, Clths	2.1766	2.3104
Food, Hlth	2.4821	2.4774	Txtls, Food	1.3381	1.3612	Food, Mines	0.8548	0.8506
Food, Coal	1.1780	1.1538	Food, Oil	1.2385	1.2405	Food, Util	3.3179	3.3004
Food, Telcm	2.2938	2.2967	Food, Servs	1.7698	2.0227	BusEq, Food	1.6004	1.6239
Trans, Food	1.2507	1.3422	Whlsl, Food	2.5230	2.5505	Rtail, Food	2.3369	2.3671
Food, Meals	2.1548	2.1603	Fin, Food	1.9666	2.0073	Other, Food	1.1164	1.1317

Table 5.13: $P^t_{Food,Y}/0.1^2$ where the thresholds are chosen to be the 0.1 quantile of the two indices in the corresponding stress scenario in Section 5.7.1. The order of the response variables based on our heuristic is also indicated

We see that most quotients in Tables 5.12 and Table 5.13 are larger than 1 hence most indices show positive conditional correlation to the Food index when the manufacturing sector is stressed. This relationship becomes stronger as the stress level of the manufacturing sector changes from 0.05 to 0.01 though the difference is small. Furthermore as expected, the quotient is always larger when the threshold is chosen to be the 0.1 quantile than of the median.

In the forward selection procedure the first added covariate is always the Cnstr index and the second one is mostly (16 out of 21) the Paper index. The Chems index is now included in the conditional distribution when Y is Hlths, Coal, Oil, Meals and Other index. As the forward selection is now applied to both the response variables, more covariates are included than in the case of only one response variable.

The Smoke index, which was the least affected industry by the stressed manufacturing sector in the case of one response variable analysis, now has the largest joint probability of being below the threshold while Food is also below the threshold. When the thresholds are the median the quotients for the energy industry (indices Mines, Coal and Oil) are smaller than 1, which indicates their negative conditional correlation with Food when the manufacturing sector is stressed. However, this behaviour is not visible when partial correlation are examined (0.03627 for Food,Mines, 0.09411 for Food,Coal and 0.09044 for Food,Oil).

Similarly, the analysis concerning the joint behaviour of different indices other than Food conditioned on the stressed manufacturing sector can be carried out.

5.8. CONCLUSION

In this chapter results of an effort to extend the simplified D-vine based forward regression method, introduced in Kraus & Czado (2017*a*), by allowing different types of regular vine structures have been presented. However, when combined with the forward selection procedure (which determines the order of the variables for the vine structure) the benefits of such a extension, in terms of possible improvements in estimated models, are shown to be limited. Other than the forward selection procedures to choose the order of the variables in the construction of the vine structures can be easily combined with the methods proposed in this chapter.

Even if computationally expensive, R-vine based regression methods are shown to provide a better alternative to the traditional linear regression. Moreover, the R-vine

based regression can be tailored to give an improved performance in particular regions of the conditional distribution by choosing a specific selection criterion for this region (e.g. specific quantile level). R-vine regression can also be extended to the case of more than one response variables. This is of great importance if the joint behaviour of the response variables is required as we have illustrated in the stress test analysis.

5.A. APPENDIX TO CHAPTER 5

5.A.1. MODEL SPECIFICATION

RANDOM DIM-8 D-VINE

				V_0				
structure	$\mathscr{C}_0$	τ_0	structure	$\mathscr{C}_0$	τ_0	structure	$\mathscr{C}_0$	τ_0
Tree-7								
12|345678	G	0.68						
Tree-6								
42|35678	G90	-0.46	31|45678	C270	-0.26			
Tree-5								
82|3567	G	0.25	51|4678	G270	-0.35	43|5678	G270	-0.35
Tree-4								
72|356	C	0.19	61|478	G270	-0.27	83|567	G	0.31
54|678	C270	-0.67						
Tree-3								
62|35	G270	-0.06	71|48	J90	-0.03	73|56	J270	-0.60
64|78	C180	0.80	85|67	J90	-0.29			
Tree-2								
52|3	C270	-0.56	81|4	J270	-0.77	63|5	J90	-0.18
74|8	G270	-0.38	75|6	C180	0.63	68|7	C270	-0.15
Tree-1								
32	J270	-0.52	41	C270	-0.33	53	G90	-0.22
84	G180	0.43	65	C90	-0.59	78	J270	-0.65
76	C270	-0.08						

Table 5.14: A random D-vine copula model with order $(1,4,8,7,6,5,3,2)$. $\mathscr{C}_0$ represents the set of copula families assigned to each edge and τ_0 is the set of kendall's tau of the corresponding copula.

LINEAR FORWARD REGRESSION MODEL

Linear forward regression model				$AIC = -91.15$			$\sigma = 0.9517$	
coefficient	intercept	X_2	X_3	X_4	X_5	X_6	X_7	X_8
	0.26	0.42	0.23	0.88	1.79	-0.09	-0.03	-1.53
p-value	8.87E-3	< 2E-16	4.76E-3	4,94E-9	< 2E-16	0.21	0.60	< 2E-16

Table 5.15: The linear regression model based on the AIC forward selection where X_1 is the response variable. σ is the standard deviation of the residuals. The p-values refers to the test with null hypothesis stating that the coefficient is zero.

GAUSSIAN COPULA FORWARD SELECTION MODEL

The Gaussian copula model is estimated by estimating the marginal distributions and a multivariate Gaussian copula separately. In order to include the forward selection in this model the distributions of both the response variable and the covariates are estimated by kernel smoothing and they are transformed into u-scale. Then we transform them into z-scale and apply a linear forward regression approach.

Gaussian copula by linear approach				$AIC = -2384.95$			$\sigma = 0.3023$	
coefficient	intercept	X_2	X_3	X_4	X_5	X_6	X_7	X_8
	0.02	0.26	0.09	0.46	0.30	-0.13	-0.09	-1.09
p-value	0.11	< 2E-16	2.77E-5	< 2E-16	< 2E-16	1.64E-7	1.42E-4	< 2E-16

Table 5.16: The linear regression model based on the AIC forward selection for the data transformed into z-scale where X_1 is the response variable. σ is the standard deviation of the residuals. The p-values refers to the test with null hypothesis stating that the coefficient is zero.

5

D-VINE BASED FORWARD SELECTION MODEL

				V_2			
structure	$\mathscr{C}_2$	$\hat{\tau}_2$	logLik	structure	$\mathscr{C}_2$	$\hat{\tau}_2$	logLik
Tree-6							
31|24678	Tawn180	0.11	18.86				
Tree-5							
41|2678	BB1_180	0.27	84.15	83|2467	BB7_180	0.16	39.60
Tree-4							
61|278	Tawn2_270	-0.17	35.70	23|467	BB8_90	-0.55	398.63
48|267	BB7	0.13	45.50				
Tree-3							
71|28	BB8_90	-0.18	65.18	73|46	Tawn2_270	-0.36	177.60
68|27	F	-0.18	35.78	24|67	Tawn2	0.07	21.52
Tree-2							
21|8	BB1_180	0.29	96.11	63|4	Tawn90	-0.20	88.03
78|2	J270	-0.62	672.31	74|6	BB8_270	-0.70	860.47
62|7	Tawn	0.29	151.53				
Tree-1							
81	t	-0.77	967.74	43	BB7_180	0.37	172.79
28	J90	-0.21	65.67	64	BB7	0.47	335.30
72	C	0.26	88.61	76	J90	-0.07	13.18

Table 5.17: Final D-vine copula model through the forward selection. $\mathscr{C}_2$ represents the set of copula families assigned to each edge and $\hat{\tau}_2$ is the set of estimated kendall's tau of the corresponding copula.

WBTS IN R-VINE FORWARD SELECTION

(1,8,2,3)		(1,8,2,4)		(1,8,2,5)		(1,8,2,6)		(1,8,2,7)	
$\omega_{32;8}$	$\omega_{38;2}$	$\omega_{42;8}$	$\omega_{48;2}$	$\omega 52;8$	$\omega 58;2$	$\omega_{62;8}$	$\omega_{68;2}$	$\omega_{72;8}$	$\omega_{78;2}$
-0.69	**0.60**	**0.23**	0.64	**-0.43**	-0.58	0.34	**0.05**	0.19	**-0.79**
ω_{38}	ω_{32}	ω_{48}	ω_{42}	ω_{58}	ω_{52}	ω_{68}	ω_{62}	ω_{78}	ω_{72}
0.58	**-0.68**	**0.61**	0.04	**-0.48**	-0.26	-0.03	**0.33**	-0.80	**0.30**
(0)		(1)		(1)		(0)		(0)	

Table 5.18: The wBTs in the 3^{rd} step for each possible new variable candidate in the forward selection. The selected path in wBT is denoted in bold and the result sequence of indicators is listed below each wBT.

5

(1,8,2,7,3)				(1,8,2,7,4)			
$\omega_{37;28}$		$\omega_{38;27}$		$\omega_{47;28}$		$\omega_{48;27}$	
-0.24		**0.27**		**-0.59**		0.06	
$\omega_{38;2}$	$\omega_{32;8}$	$\omega_{37;2}$	$\omega_{32;7}$	$\omega_{48;2}$	$\omega_{42;8}$	$\omega_{47;2}$	$\omega_{42;7}$
0.60	-0.69	-0.59	**-0.66**	0.64	**0.23**	-0.78	0.40
ω_{32}	ω_{38}	ω_{32}	ω_{37}	ω_{42}	ω_{48}	ω_{42}	ω_{47}
-0.68	0.58	-0.68	**-0.61**	0.04	**0.61**	0.04	-0.73
(0,0)				(1,0)			
(1,8,2,7,5)				(1,8,2,7,6)			
$\omega_{57;28}$		$\omega_{58;27}$		$\omega_{67;28}$		$\omega_{68;27}$	
0.51		-0.04		-0.35		**-0.25**	
$\omega_{58;2}$	$\omega_{52;8}$	$\omega_{57;2}$	$\omega_{52;7}$	$\omega_{68;2}$	$\omega_{62;8}$	$\omega_{67;2}$	$\omega_{62;7}$
-0.58	**-0.43**	0.71	-0.55	0.05	0.34	**-0.25**	0.39
ω_{52}	ω_{58}	ω_{52}	ω_{57}	ω_{62}	ω_{68}	ω_{62}	ω_{67}
-0.26	**-0.48**	-0.26	0.58	0.33	-0.03	**0.33**	-0.13
(1,0)				(0,1)			

Table 5.19: The wBTs in the 4^{th} step for each possible new variable candidate in the forward selection.The selected path in wBT is denoted in bold and the result sequence of indicators is listed below each wBT.

$(1,8,2,7,5,3)$							
$\omega_{35;278}$ **-0.50**				$\omega_{37;258}$ 0.05			
$\omega_{37;28}$ 0.24		$\omega_{38;27}$ **0.27**		$\omega_{35:28}$ -0.54		$\omega_{32;58}$ -0.79	
$\omega_{38;2}$ 0.60	$\omega_{32;8}$ -0.69	$\omega_{37;2}$ -0.59	$\omega_{32;7}$ **-0.66**	$\omega_{32;8}$ -0.69	$\omega_{38;2}$ 0.60	$\omega_{35;8}$ -0.06	$\omega_{38;5}$ 0.52
ω_{32} -0.68	ω_{38} 0.58	ω_{32} -0.68	ω_{37} **-0.61**	ω_{38} 0.58	ω_{32} -0.68	ω_{38} 0.58	ω_{35} -0.32
$(1,0,0)$							
$(1,8,2,7,5,4)$							
$\omega_{45;278}$ **-0.87**				$\omega_{47;258}$ -0.34			
$\omega_{47;28}$ **-0.59**		$\omega_{48;27}$ 0.06		$\omega_{45:28}$ -0.90		$\omega_{42;58}$ -0.38	
$\omega_{48;2}$ 0.64	$\omega_{42;8}$ **0.23**	$\omega_{47;2}$ -0.78	$\omega_{42;7}$ 0.40	$\omega_{42;8}$ 0.23	$\omega_{48;2}$ 0.64	$\omega_{45;8}$ -0.89	$\omega_{48;5}$ 0.49
ω_{42} 0.04	ω_{48} **0.61**	ω_{42} 0.04	ω_{47} -0.73	ω_{48} 0.61	ω_{42} 0.04	ω_{48} 0.61	ω_{45} -0.91
$(1,1,0)$							
$(1,8,2,7,5,6)$							
$\omega_{65;278}$ -0.85				$\omega_{67;258}$ **0.22**			
$\omega_{67;28}$ -0.35		$\omega_{68;27}$ -0.25		$\omega_{65:28}$ -0.86		$\omega_{62;58}$ **-0.10**	
$\omega_{68;2}$ 0.05	$\omega_{62;8}$ 0.34	$\omega_{67;2}$ -0.25	$\omega_{62;7}$ 0.39	$\omega_{62;8}$ 0.34	$\omega_{68;2}$ 0.05	$\omega_{65;8}$ -0.88	$\omega_{68;5}$ **-0.68**
ω_{62} 0.33	ω_{68} -0.03	ω_{62} 0.33	ω_{67} -0.13	ω_{68} -0.03	ω_{62} 0.33	ω_{68} -0.03	ω_{65} **-0.75**
$(0,0,0)$							

Table 5.20: The wBTs in the 5^{th} step for each possible new variable candidate in the forward selection.The selected path in wBT is denoted in bold and the result sequence of indicators is listed below each wBT.

For simplicity we don't show the wBTs in the 6^{th} and the 7^{th} step in forward selection but only the sequence of indicators in the below table,

	sequence of indicators
$(1,8,2,7,5,4,3)$	$(1,1,0,0)$
$(1,8,2,7,5,4,6)$	$(0,0,0,0)$
$(1,8,2,7,5,4,6,3)$	$(1,0,1,0,0)$

Table 5.21: Sequence of indicators in the 6^{th} and the 7^{th} step in the forward selection.

R-VINE BASED FORWARD SELECTION MODEL

			V_3				
structure	$\mathscr{C}_3$	$\hat{\tau}_3$	logLik	structure	$\mathscr{C}_3$	$\hat{\tau}_3$	logLik
Tree-7							
31|245678	F	0.26	77.89				
Tree-6							
61|24578	F	-0.25	76.52	63|24578	Tawn2	0.22	80.77
Tree-5							
41|2578	BB7_180	0.22	68.75	43|2578	t	0.01	48.63
56|2478	Twan2_90	-0.26	117.30				
Tree-4							
51|278	J180	0.14	34.67	53|278	BB8_90	-0.33	163.01
76|248	BB8	0.33	161.45	45|278	F	-0.69	647.13
Tree-3							
71|28	BB8_90	-0.18	65.18	83|27	Tawn2	0.14	45.73
26|48	Tawn2	0.20	62.28	75|28	Tawn	0.29	145.28
74|28	BB8_270	-0.45	252.14				
Tree-2							
21|8	BB1_180	0.29	96.11	23|7	BB8_90	-0.49	343.00
86|4	t	-0.53	397.76	25|8	Tawn270	-0.24	120.00
24|8	Tawn2	0.16	55.69	87|2	J90	-0.62	672.31
Tree-1							
81	t	-0.77	967.74	73	t	-0.45	241.27
46	BB7	0.47	335.30	85	C270	-0.33	182.38
84	BB1	0.45	262.70	27	C	0.26	88.61
28	J90	-0.21	65.67				

Table 5.22: Final R-vine copula model through the forward selection. $\mathscr{C}_3$ represents the set of copula families assigned to each edge and $\hat{\tau}_3$ is the set of estimated kendall's tau of the corresponding copula.

MQE OF THE TRUE CONDITIONAL DISTRIBUTION IN SIMULATION

			Resp1				
Size	Dim		0.01	0.05	0.5	0.95	0.99
300	10	True	0.9649	0.9731	0.9810	0.9698	0.9586
			(0.9645)	(0.9721)	(0.9804)	(0.9691)	(0.9586)
	15	True	0.9934	0.9959	0.9972	0.9953	0.9920
			(0.9930)	(0.9957)	(0.9973)	(0.9959)	(0.9942)
	20	True	0.9970	0.9991	0.9996	0.9994	0.9986
			(0.9987)	(0.9994)	(0.9997)	(0.9996)	(0.9993)
1000	10	True	0.9620	0.9705	0.9779	0.9630	0.9485
			(0.9618)	(0.9707)	(0.9779)	(0.9633)	(0.9500)
	15	True	0.9948	0.9961	0.9972	0.9953	0.9930
			(0.9948)	(0.9961)	(0.9972)	(0.9956)	(0.9939)
	20	True	0.9984	0.9993	0.9996	0.9992	0.9984
			(0.9986)	(0.9994)	(0.9996)	(0.9991)	(0.9980)

Table 5.23: The simulation results for the MQE of the true conditional distribution of Resp1 at quantiles (0.01,0.05,0.5,0.95,0.99). The result is shown for 300 evaluation data set and 1000 evaluation data set, also for dimensions in 10, 15 and 20 respectively.

R-VINE BASED FORWARD SELECTION MODEL FOR TWO RESPONSE VARIABLES

The correlation matrix of the sample data set is as follows,

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
X_1	1							
X_2	0.3281	1						
X_3	-0.5727	-0.6779	1					
X_4	-0.4725	0.0398	0.5003	1				
X_5	0.3746	-0.2593	-0.3206	-0.9141	1			
X_6	0.0879	0.3332	0.1190	0.6583	-0.7538	1		
X_7	0.6876	0.2957	-0.6138	-0.7347	0.5784	-0.1280	1	
X_8	-0.9216	-0.2288	0.5841	0.6130	-0.4815	-0.0313	-0.7999	1

Table 5.24: The correlation matrix for the sample data set in Section 5.3.3.

				V_4			
structure	$\mathscr{C}_4$	$\hat{\tau}_4$	logLik	structure	$\mathscr{C}_4$	$\hat{\tau}_4$	logLik
Tree-7							
21|345678	BB1_180	0.37	189.63				
Tree-6							
71|34568	Tawn180	0.12	18.05	72|34568	t	0.08	10.23
Tree-5							
51|3468	t	0.04	9.11	52|3468	BB8_90	-0.15	29.82
47|3568	BB8_270	-0.39	196.57				
Tree-4							
41|368	BB8	0.27	82.33	82|346	G	0.27	107.82
87|356	BB7_270	-0.11	18.33	54|368	t	-0.42	232.33
Tree-3							
61|38	BB8	0.12	24.42	42|36	Tawn2_180	0.24	78.36
37|56	BB8_90	-0.55	419.81	84|36	N	0.56	436.93
85|36	J90	-0.56	585.81				
Tree-2							
31|8	Tawn2_270	-0.15	27.22	62|3	C180	0.42	296.16
67|5	J	0.44	346.61	34|6	Tawn	0.33	203.53
35|6	BB7_90	-0.23	63.13	68|3	Tawn2_90	-0.12	18.11
Tree-1							
81	t	-0.77	967.74	32	BB8_270	-0.53	413.34
57	Tawn2	0.36	242.61	64	BB7	0.47	335.30
65	BB8_270	-0.59	533.74	38	t	0.42	215.33
36	t	0.07	35.60				

Table 5.25: Final R-vine copula model for two response variables (1,2) through the forward selection. $\mathscr{C}_4$ represents the set of copula families assigned to each edge and $\hat{\tau}_4$ is the set of estimated kendall's tau of the corresponding copula.

V_5							
structure	$\mathscr{C}_5$	$\hat{\tau}_5$	logLik	structure	$\mathscr{C}_5$	$\hat{\tau}_5$	logLik
Tree-5							
12|3458	BB1	0.37	174.59				
Tree-4							
42|358	t	-0.12	47.64	41|358	t	0.13	33.57
Tree-3							
52|38	BB8_90	-0.48	346.16	81|35	BB8_270	-0.66	668.90
54|38	t	-0.72	804.43				
Tree-2							
32|8	BB8_270	-0.51	347.82	31|5	t	-0.43	213.86
84|3	BB8_180	0.28	134.88	85|3	J90	-0.22	126.67
Tree-1							
82	J270	-0.21	65.67	51	J	0.25	128.64
34	BB7_180	0.37	172.79	35	BB1_90	-0.24	62.28
38	t	0.42	215.33				

Table 5.26: Final R-vine copula model for two response variables (2,1) through the forward selection. $\mathscr{C}_5$ represents the set of copula families assigned to each edge and $\hat{\tau}_5$ is the set of estimated kendall's tau of the corresponding copula.

V_6							
structure	$\mathscr{C}_6$	$\hat{\tau}_6$	logLik	structure	$\mathscr{C}_6$	$\hat{\tau}_6$	logLik
Tree-4							
31|468	F	-0.18	35.64				
Tree-3							
61|48	F	-0.22	52.60	83|64	BB1_180	0.20	56.17
Tree-2							
41|8	t	0.22	60.00	63|4	Tawn90	-0.20	88.03
68|4	t	-0.53	397.76				
Tree-1							
81	t	-0.77	967.74	43	BB7_180	0.37	172.79
48	BB1	0.45	262.70	46	BB7	0.47	335.30

Table 5.27: Final R-vine copula model for the conditional margins of variable 1 in the naive approach (Separate). $\mathscr{C}_6$ represents the set of copula families assigned to each edge and $\hat{\tau}_6$ is the set of estimated kendall's tau of the corresponding copula.

			V_7				
structure	$\mathscr{C}_7$	$\hat{\tau}_7$	logLik	structure	$\mathscr{C}_7$	$\hat{\tau}_7$	logLik
Tree-6							
72|34568	t	0.07	17.02				
Tree-5							
62|3458	t	0.09	28.37	67|3458	BB8	0.32	139.20
Tree-4							
82|345	t	0.15	35.53	87|345	BB8_90	-0.30	182.64
86|345	BB8_90	-0.47	333.49				
Tree-3							
42|35	BB8_270	-0.13	32.35	47|35	t	-0.41	222.57
36|54	J270	-0.13	38.72	38|45	BB8	0.27	91.88
Tree-2							
52|3	BB8_90	-0.53	416.47	37|5	F	-0.46	248.31
56|4	Tawn_90	-0.38	256.70	48|5	BB7	0.38	219.07
43|5	BB1	0.36	160.93				
Tree-1							
32	BB8_270	-0.53	413.34	57	Tawn2	0.36	242.61
46	BB7	0.47	335.30	58	C90	-0.33	182.38
53	BB1_270	-0.24	62.28	54	F	-0.77	908.24

Table 5.28: Final R-vine copula model for the conditional margins of variable 2 in the naive approach (Separate). $\mathscr{C}_7$ represents the set of copula families assigned to each edge and $\hat{\tau}_7$ is the set of estimated kendall's tau of the corresponding copula.

6

REGULAR VINES WITH STRONGLY CHORDAL PATTERN OF (CONDITIONAL) INDEPENDENCE

Multivariate statistical models can be simplified by assuming that a pattern of conditional independence is present in the given data. A popular way of capturing the (conditional) independence is to use probabilistic graphical models. The relationship between strongly chordal graphs and m-saturated vines is proved. Moreover, an algorithm to construct an m-saturated vine structure corresponding to strongly chordal graph is provided. This allows the reduction of regular vine copula models complexity. When the underlying data is sparse our approach leads to a model with better performance as compared with current heuristic methods. Furthermore, due to reduction of model complexity it is possible to evaluate all vine structures as well as to fit non-simplified vines. These advantages have been shown in the simulated and real data examples.

Parts of this chapter have been published in Zhu & Kurowicka (2022).

6.1. INTRODUCTION

Probabilistic graphical models are very useful in representing the joint distribution of random variables. In these models the variables are associated with the vertices of the graph and the (conditional) independence in the joint distribution of these variables are determined by the absence of edges. It suffices to study the structure of the graph in order to assess whether variables are (conditionally) independent. The corresponding joint density can then be built with additional information provided about the types of dependence encoded in the edges. For example, the joint probability density represented by a chordal graph is known to be equal to the product of densities of the variables in maximal cliques divided by the product of densities over the variables in the separators (Pearl (1988)), and the joint density of a Bayesian network is the product of conditional densities of variables given their parents in the graph (Lauritzen (1996)). However, for continuous random variables, these models are not easy to be estimated or/and are not simple to be sampled in high dimensions unless we assume that the distributions are multivariate Gaussian or a Gaussian copula.

The regular vine model, which is a set of nested trees, is another example of graphical model to represent a joint distribution. In principle regular vines are fully connected graphs and the (conditional) independence can be specified by setting the bivariate copulas to be independence copula. This can be seen as removing some nodes in the nested set of trees. In general, however, it is not known how the structure of a regular vine with some nodes missing can be used to describe all (conditional) independence of the random variables in the joint distribution. There are cases where the conditional independence can be easily specified in a regular vine model. In Brechmann et al. (2012), truncated regular vines are introduced where all bivariate copulas above certain tree level are set to be the independence copula. The level of truncation is chosen during the tree-wise estimation process (estimation stops when the absolute AIC of the new copulas in the tree is smaller than a predetermined threshold or the improvement in likelihood by adding the next tree is not sufficient (determined by the Vuong test (Vuong (1989))). Determining (conditional) independence during estimation is convenient but is in general not optimal. The chosen vine structure, the simplifying assumption and the tree-wise estimation procedure can result in wrongly specified conditional (in)dependence in the estimated model.

The idea of simplifying regular vine copula models by assuming a pattern of conditional independence has been already discussed in the literature. Both Müller & Czado (2018) and Hobæk Haff et al. (2016) present conditions under which a directed acyclic graph (DAG) in the former and a chordal graph in the latter corresponds to a truncated vine. Then heuristics are provided to construct the vine copula model based on a graph obtained from data. A different heuristic construction of the vine structure is proposed in Müller & Czado (2019*b*), where conditional independence result from structural equation models. In Müller & Czado (2019*a*) the conditional independence is found using graphical Lasso (Friedman et al. (2007)), where variables are partitioned into homogeneous groups and subvines corresponding to each group are estimated at first. Then these subvines are combined together in a similar way as by using the concept of merging discussed in Cooke et al. (2015). This method requires to specify a pre-determined level for a maximum number of dimensions for subvines as well as a truncation level.

In this chapter we show that the set of conditional independence in the form of a special chordal graph, called strongly chordal graph (Farber (1983)), can be represented by regular vine copulas. Our main result establishes the equivalence between strongly chordal graphs and m-saturated vines (abbreviated as m-vine, introduced in Kurowicka & Cooke (2006)). A special case of this relationship has been studied in Hobæk Haff et al. (2016), where the authors showed that the truncated vines (which are special cases of m-vines) correspond to chordal graphs satisfying extra conditions.

The chapter is organized as follows: we start with basic notations concerning graph theory and m-vines in Section 6.2; In Section 6.3 the main theorem of this chapter describing the relationship between a strongly chordal graph and an m-vine is proved and an algorithm to construct an m-vine corresponding to a given graph is proposed. Simulation results are presented in Section 6.4, where two examples showing the advantages of the m-vine approach are listed in Section 6.4.1. A general heuristic of choosing the m-vine structure for a given data set is proposed in Section 6.4.2, which is tested by a simulation study in Section 6.4.3. Finally, a real data analysis is implemented in Section 6.5 and the conclusions finalize the chapter in Section 6.6.

6.2. BACKGROUND

In this section, we present basic definitions from graph theory and a brief introduction to m-vines as well as the notations we will use in the chapter.

6.2.1. GRAPHS

Let $\mathcal{G}$ be a graph with vertices $V(\mathcal{G}) = \{v_1, \ldots, v_n\}$ and edges $E(\mathcal{G})$. A graph is **complete** if every pair of vertices is connected by a unique edge. Two vertices are **adjacent** if there is an edge between them, whereas the adjacency of edges means two edges share a common vertex. We say that $\mathcal{G}(\mathcal{U})$ is a **subgraph** of graph $\mathcal{G}$ induced by the vertex set $\mathcal{U}$ if $\mathcal{G}(\mathcal{U})$ is the graph with vertex set $\mathcal{U} \subseteq V(\mathcal{G})$ and with edge set $\{(u,v) \in E(\mathcal{G}) | u, v \in \mathcal{U}\}$. $\mathcal{C} \subseteq V(\mathcal{G})$ is a **clique** of $\mathcal{G}$ when $\mathcal{G}(\mathcal{C})$ is a complete subgraph. If $\mathcal{G}(\mathcal{C})$ is a maximal complete subgraph of graph $\mathcal{G}$ then $\mathcal{C}$ is called **maximal clique**.

A **path** of length k between vertices α and β in graph $\mathcal{G}$ is a sequence $\alpha = \alpha_0, ..., \alpha_k = \beta$ of distinct vertices of $\mathcal{G}$ such that $(\alpha_{i-1}, \alpha_i) \in E(\mathcal{G})$ for all $i = 1, ..., k$. A **cycle** of length k is a path of length k in which the end points are identical ($\alpha = \beta$).

A vertex set $\mathcal{S} \subseteq V(\mathcal{G})$ is said to be (α, β)-**separator** if all paths from α to β intersect $\mathcal{S}$. Furthermore the set $\mathcal{S}$ is said to **separate** vertex set $\mathcal{A}$ from vertex set $\mathcal{B}$ if $\mathcal{S}$ is an (α, β)-separator for every $\alpha \in \mathcal{A}$ and $\beta \in \mathcal{B}$.

A connected graph T is called a **tree** if it has no cycle.

Graph $\mathcal{G}$ is said to be **chordal** if every cycle of $\mathcal{G}$ with length larger than 3 has a chord (where chord of a cycle is an edge joining two nonconsecutive vertices in the cycle). Hence, a tree is an example of a chordal graph. A **strongly chordal** graph is a chordal graph with the property that every cycle of even length of at least six contains a chord. This chord combines with the edges of the cycle to form two shorter even-length cycles.

In this chapter we will discuss also graphs whose vertices correspond to sets of elements (and vertices will be referred to as nodes) and edges correspond to intersections of sets. A clique tree (or junction tree), $\mathcal{T}$, of graph $\mathcal{G}$ is an example of such a graph. If

$\mathscr{C}(\mathscr{G}) = (\mathscr{C}_1, ..., \mathscr{C}_k)$ is a set of maximal cliques of graph $\mathscr{G}$ then $\mathscr{T}$ with nodes $\mathscr{C}(\mathscr{G})$ is said to be a **clique tree** if any nonempty intersection $\mathscr{C}_i \cap \mathscr{C}_j$ is contained in every node on the unique path in $\mathscr{T}$ between $\mathscr{C}_i$ and $\mathscr{C}_j$. This property is called the running intersection property (RIP). The set of separators $\mathscr{S}(\mathscr{T})$ in a clique tree is the nonempty intersections of adjacent nodes. For a vertex v of $\mathscr{G}$, $\mathscr{T}_{\{v\}}$ denotes the subgraph of $\mathscr{T}$ induced by nodes containing v. If $\mathscr{T}$ is a clique tree, then for any $v \in \mathscr{V}(\mathscr{G})$, $\mathscr{T}_{\{v\}}$ is always connected according to RIP.

We show below an example (from Mukhopadhyay & Rahman (2021)) to illustrate the above definitions.

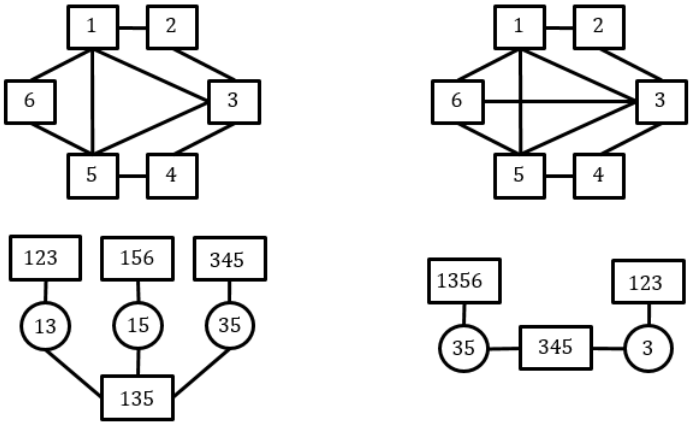

Figure 6.1: The graphs below are the clique trees of chordal graphs above. The left graph is a chordal graph but not strongly chordal while the right one is a strongly chordal graph.

In Figure 6.1, two graphs with six nodes (above) and their corresponding clique trees (underneath) are presented. Maximal cliques are shown in squares and separators are shown in circles in the clique tree. The left graph contains four cliques $\{1,2,3\}$, $\{1,5,6\}$, $\{3,4,5\}$ and $\{1,3,5\}$ (this can be seen also in its clique tree). This graph is chordal but it is not strongly chordal. This graph can be made strongly chordal when an extra edge 36 is added. Alternatively, edge 14 or edge 25 instead of edge 36 could be added. The graph constructed by adding edge 36 is shown in Figure 6.1 (right). We can observe that the addition of edge 36 leads to the existence of clique $\{1,3,5,6\}$, which is composed of two cliques $\{1,5,6\}$ and $\{1,3,5\}$.

In the next section, a graphical model called m-vine in presented.

6.2.2. VINES

Vines introduced in Section 1.2.1 in Chapter 1 are in principle fully connected graph. Figure 6.2 shows two examples of regular vine on 5 elements, denoted as $V_1(5)$ (left) and $V_2(5)$ (right).

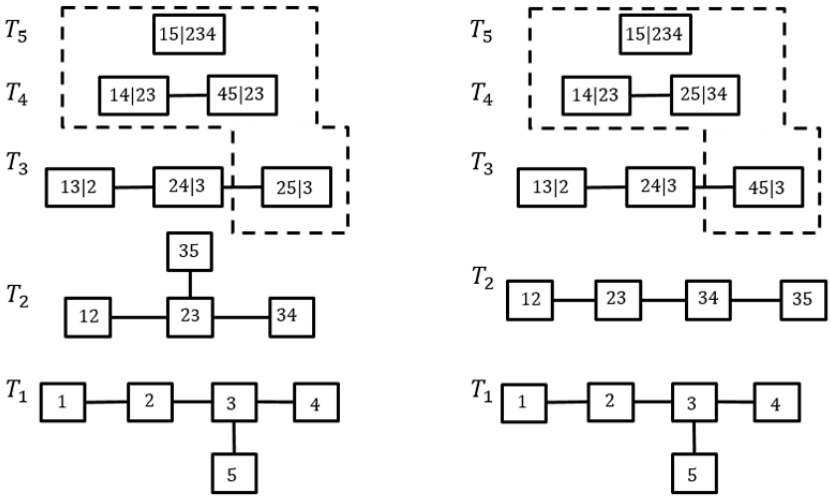

Figure 6.2: Two regular vines on 5 elements with conditioned and conditioning sets. The left vine is denoted as $V_1(5)$ and the right one is $V_2(5)$. The nodes included in the dashed areas can be removed to form incomplete vines.

The vine triangular arrays of the vines $V_1(5)$ and $V_2(5)$ in Figure 6.2 are shown in Figure 6.3.

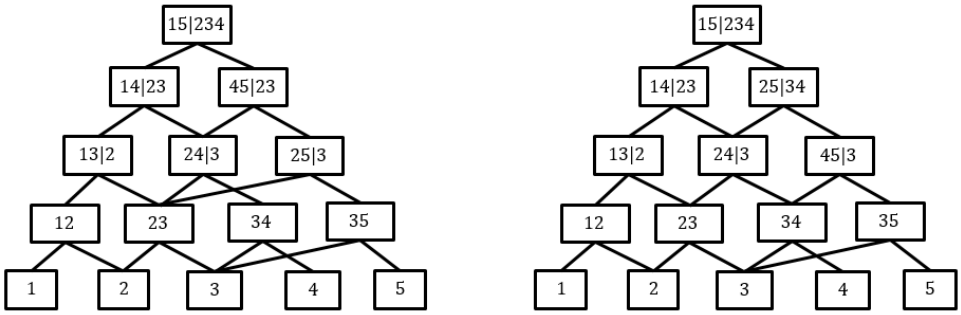

Figure 6.3: Vine triangular array for $V_1(5)$ (left), and $V_2(5)$ (right) in Figure 6.2.

Incomplete vines are vines from which some nodes have been removed. In this chapter two kinds of incomplete vines: truncated vines (Brechmann et al. (2012)) and m-saturated vines (Kurowicka & Cooke (2006)), are considered.

Definition 6.2.1 (Truncated vine)**.** *An incomplete vine is a k-truncated vine of a regular vine $V(n)$, $1 \leq k \leq n$, if all nodes in trees $T_i, k+1 \leq i \leq n$ have been removed.*

Definition 6.2.2 (m-saturated vine (for short m-vine))**.** *An incomplete vine is a m-saturated vine of a regular vine $V(n)$ if all descendants of a node in its node set belong to the node set of the incomplete vine.*

It is easy to observe that a truncated vine is a special type of m-vine.

Regular vines have only one top node, which is the node in the highest tree level, however, for truncated vine or m-vine there are several "top" nodes, which we call **maximal nodes**. Maximal nodes are nodes that do not have ancestor. For example, the maximal nodes in a 3-truncated vine of $V_1(5)$, shown in Figure 6.2 (left), are $13|2, 24|3, 25|3$. The maximal nodes of m-vine can appear in different tree levels. For $V_2(5)$, shown in Figure 6.2 (right), the maximal nodes in an m-vine constructed by removing nodes in the dashed area are $13|2, 24|3, 35$.

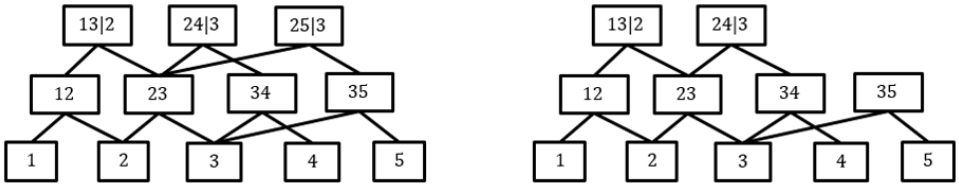

Figure 6.4: Vine triangular array for 3-truncated vine (left) of $V_1(5)$, and an m-vine (right) by removing the nodes in the dashed area in $V_2(5)$ in Figure 6.2.

We will represent an m-vine as $V(U^*_{e_1}, \ldots, U^*_{e_k})$ where $U^*_{e_1}, \ldots, U^*_{e_k}$ are the constraint sets of its maximal nodes $e_1, \ldots, e_k$. Note that this representation is not unique as both m-vines in Figure 6.2 obtained by removing nodes in dashed areas have the same maximal nodes, hence the same representation: $V(\{1,2,3\},\{2,3,4\},\{3,5\})$. We will call two maximal nodes **adjacent** if these two nodes share common nodes in their descendants (a common subvine). For example, maximal nodes $24|3$, $13|2$ and 35 are adjacent to each other.

In Section 6.3, we will study the correspondence of incomplete vines and chordal graphs.

6.3. RELATIONSHIP BETWEEN M-VINE AND STRONGLY CHORDAL GRAPH

In Kurowicka & Cooke (2006), Hobæk Haff et al. (2016), it has been shown that there exists a relationship between chordal graphs and m-vines (or truncated vines). We discuss it briefly in Section 6.3.1 and conclude that m-vines correspond to chordal graphs, but not all chordal graphs correspond to m-vines (this has been shown in Hobæk Haff et al. (2016) for truncated vines). A more general discussion on this topic follows, leading to the main result of this chapter and its proof. We show the equivalence of m-vines and strongly chordal graphs.

6.3.1. CHORDAL GRAPH AND M-VINE

One of the many characterizations of chordal graphs is the existence of a corresponding clique tree (Gavril (1974)). To show that an m-vine corresponds to a chordal graph, it is sufficient to observe that constraint sets of maximal nodes in an m-vine form maximal cliques and the separators are the intersections of constraint sets of adjacent maximal

nodes. A tree obtained in this way is a clique tree of a chordal graph (Kurowicka & Cooke (2006), Hobæk Haff et al. (2016)).

As an example, let us consider the m-vine obtained by removing nodes in dashed areas shown in Figure 6.2. For both m-vines, the corresponding clique tree (maximal cliques shown in squares and separators are in circles) and the chordal graphs are shown in Figure 6.5 (left panel - the chordal graph for the m-vines when the nodes in the dashed area are removed for both vines; middle panel - the chordal graph for the 3-truncated vine in Figure 6.2 (left); right panel - the chordal graph for the 3-truncated vine in Figure 6.2 (right)).

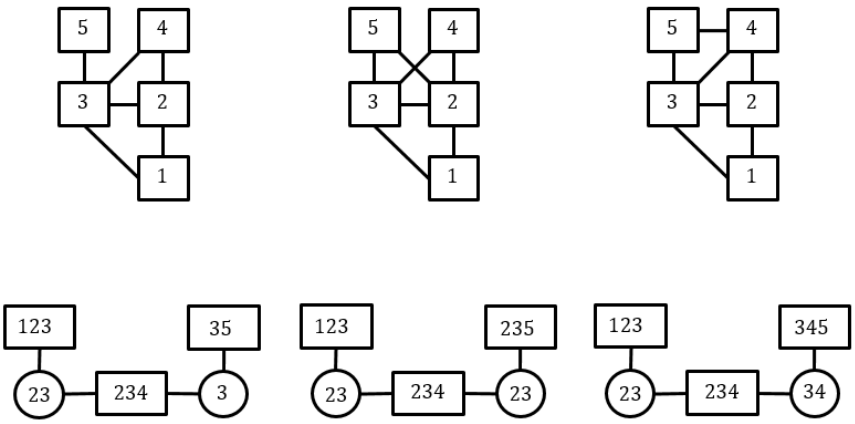

Figure 6.5: Examples of clique trees (below) of chordal graphs (above). The graph on the left corresponds to the m-vines in Figure 6.2 where the nodes in the dashed area have been removed. The middle and the right graphs correspond to the 3-truncated vines for $V_1(5)$ and $V_2(5)$ in Figure 6.2, respectively.

It is not, however, the case that all chordal graphs correspond to m-vines. We call those clique trees to which a corresponding m-vine exists as **regular clique trees**.

Definition 6.3.1 (Regular clique tree)**.** *A clique tree $\mathcal{T}$ is regular if there exists an m-vine $V(U^*_{e_1},\dots,U^*_{e_k})$ such that the constraint sets of the maximal nodes $e_1,\dots,e_k$ of the m-vine are the nodes $\mathcal{V}(\mathcal{T}) = \{\mathscr{C}_1,\dots,\mathscr{C}_k\}$ of the clique tree, respectively.*

In Figure 6.6, we show an example of a regular and a non-regular clique tree. The 'black' tree is a regular clique tree and the non-regular clique tree is obtained when node $\{1,3,7\}$ is replaced by $\{2,3,7\}$, and we print it in red.

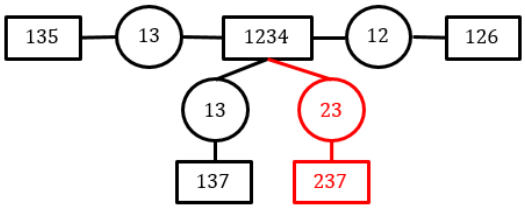

Figure 6.6: A regular (black, $\mathcal{T}$) and non-regular (red, $\mathcal{T}'$) clique tree.

For the 'black' clique tree $\mathcal{T}$ in Figure 6.6, there are four cliques shown in squares and three separators shown in circles. In this example, the corresponding m-vine can be constructed as follows: the clique with cardinality (number of elements) equal to 4 is the constraint set of the maximal node in tree T_4 of the m-vine, whereas the cliques with cardinality equal to 3 are the constraint sets of maximal nodes in T_3. The node with constraint set $\{1,2,3,4\}$ has two children in T_3 and they share the common child 13 with node $\{1,3,5\}$ and $\{1,3,7\}$. Moreover, they share the common child 12 with node $\{1,2,6\}$. One possible m-vine can then have maximal nodes 26|1, 15|3, 17|3 and 23|14 (it has two children 24|1 and 34|1).

When we look at the 'red' clique tree $\mathcal{T}'$, where node $\{1,3,7\}$ is replaced by $\{2,3,7\}$, following the arguments above, we can observe that the children of node $\{1,2,3,4\}$ should share the common node 13 with node $\{1,3,5\}$, the common node 12 with node $\{1,2,6\}$ and the common node 23 with node $\{2,3,7\}$. This is not possible for a regular vine as 13, 12 and 23 constitutes a cycle.

6.3.2. STRONGLY CHORDAL GRAPH AND M-VINE

As proved in McKee (2003), strongly chordal graphs are characterized by the existence of a strong clique tree, defined below:

Definition 6.3.2 (Strong clique tree)**.** *Clique tree $\mathcal{T}$ is a strong clique tree of graph $\mathcal{G}$ if there exists the sequence of trees $\mathcal{T}^{(0)}, \mathcal{T}^{(1)}, ..., \mathcal{T}^{(k)}$, where $\mathcal{T}^{(0)} = \mathcal{T}$, $\mathcal{T}^{(k)}$ is edgeless and $\mathcal{T}^{(i)}$ is a maximum spanning tree of the separators $\mathcal{S}(\mathcal{T}^{(i-1)})$ with edge weight being the cardinality of separator such that $\mathcal{T}^{(i)}_{\{v\}}$ is connected for any vertex v in $\mathcal{G}$.*

In Figure 6.7, the sequence of trees $\mathcal{T}^{(i)}, i = 1, \ldots$ for clique trees in Figure 6.6 are shown. We can observe that in $\mathcal{T}^{(1)}$ (right) node (1,2) and (2,3) are not connected.

In McKee (2003) another characterization of a strong clique tree was proposed.

Theorem 6.3.1. *A clique tree $\mathcal{T}$ of a graph $\mathcal{G}$ is strong if and only if there do not exist distinct vertices $v_1, ..., v_k \in \mathcal{V}(\mathcal{G})$ and distinct nodes $\mathcal{C}_1, \ldots, \mathcal{C}_k \in \mathcal{V}(\mathcal{T})$ where $k \geq 3$ and, for each i, $\mathcal{C}_i \cap \{v_1, ..., v_k\} = \{v_i, v_{i+1}\}$ (computing subscripts modulo k).*

Using the above theorem, we see immediately that the red clique tree in Figure 6.6 is not strong as for cliques $\mathcal{C}_1 = \{1,3,5\}$, $\mathcal{C}_2 = \{1,2,6\}$, $\mathcal{C}_3 = \{2,3,7\}$ and a set of vertices $\{v_1 = 3, v_2 = 1, v_3 = 2\}$ the condition in the theorem is violated.

In the remaining part of this section, we prove that a strong clique tree corresponds to an m-vine. We only consider the situation when the strong clique tree is connected,

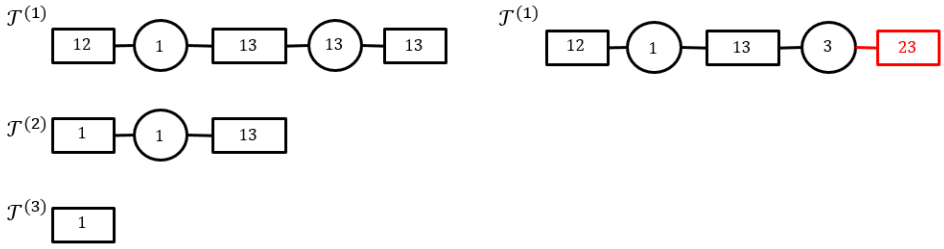

Figure 6.7: Sequence of trees $\mathcal{T}^{(i)}$, $i = 1,\ldots$ for the clique trees in Figure 6.6 (black (left) and red (right)), respectively.

otherwise the proof can be applied to each connected subgraph of the strong clique tree, separately.

Theorem 6.3.2. *Clique tree $\mathcal{T}$ is regular if and only if $\mathcal{T}$ is strong.*

Proof.
Necessity: Assume that $\mathcal{T}$ is regular but $\mathcal{T}$ is not strong. By theorem 6.3.1, there exist distinct vertices $v_1,\ldots,v_k \in V(\mathcal{G})$ and distinct nodes $\mathcal{C}_1,\ldots,\mathcal{C}_k \in V(\mathcal{T})$ where $k \geq 3$ and, for each i, $\mathcal{C}_i \cap \{v_1,...,v_k\} = \{v_i, v_{i+1}\}$ (Note that $\mathcal{C}_k \cap \{v_1,...,v_k\} = \{v_k, v_1\}$). Each $\mathcal{C}_i$ corresponds to the constraint set of a maximal node in the m-vine (a subvine of this m-vine), and they intersect by forming a path through vertices $\{v_1, v_2,\ldots, v_k, v_1\}$. This leads to a cycle in the first tree of m-vine which is not allowed by the definition of a vine.
Sufficiency: The proof of sufficiency is by construction and is presented in Section 6.3.4. □

Before illustrating the algorithm to prove sufficiency, we show in the next section the algorithm of merging vines. This lays the foundation for the algorithm of constructing an m-vine corresponding to a strongly chordal graph.

6.3.3. MERGING VINES

In this section, an algorithm to combine two regular vines $V(n)$ and $V(m)$ into a larger vine, such that $V(n)$ and $V(m)$ are its subvines is presented. This algorithm lays the foundation for the procedure to construct an m-vine corresponding to a strongly chordal graph.

We call the procedure of combining two vines **merging**. This procedure has been introduced in Cooke et al. (2015) for vines without overlapping elements. In that case merging two vines is always possible. If we want to combine two vines that share the same subset of elements, then merging is possible only if they share a common subvine on common elements. A formal definition is as follows,

Definition 6.3.3 (Merger)**.** *A regular vine $V(\mathcal{V}_n \cup \mathcal{V}_m)$ is a merger of two regular vines $V(n)$ and $V(m)$ where $\mathcal{V}_n \nsubseteq \mathcal{V}_m$ and $\mathcal{V}_m \nsubseteq \mathcal{V}_n$ if $V(n)$ and $V(m)$ are subvines of $V(\mathcal{V}_n \cup \mathcal{V}_m)$.*

Construction of the merger for two vines $V(n)$ and $V(m)$ with common subvine $V(p)$ can be done following a bottom-up approach, where each tree structure starting from

tree T_{p+1} in the merger is constructed sequentially. This can be achieved following Algorithm 11.

Algorithm 11 General approach of merging two regular vines $V(n)$ and $V(m)$

Input: regular vine $V(n)$ and $V(m)$ where $\mathcal{V}_n \cap \mathcal{V}_m = \mathcal{V}_p$ (possibly empty)
Output: A merger $V(\mathcal{V}_n \cup \mathcal{V}_m)$

1: **for** i in $p+1$ to $n+m-p$ **do**
2: Construct the tree structure T_i of the merger satisfying three conditions:
- the proximity condition;
- if $i \leq \max(n,m)$, nodes in tree T_i of the merger should include the nodes in T_i of $V(n)$ and $V(m)$;
- if $i \leq \max(n,m)$, connections between nodes of the merger have to be consistent with connections that these nodes have in $V(n)$ and $V(m)$.

3: **end for**

In Cooke et al. (2015), the merging algorithm has been presented in the case of two vines that do not overlap, hence when $\mathcal{V}_n \cap \mathcal{V}_m = \mathcal{V}_p = \emptyset$. The possible merger of two vines without overlap was shown to be determined by the choice of **sampling order** of $V(n)$ and $V(m)$. This gives a more efficient way of constructing the merger when compared to the method in Algorithm 11, and allows to prove that there are always 2^{n+m-2} possible mergers of $V(n)$ and $V(m)$. The procedure developed to merge vines without overlap can be adapted to work also in the case of vines with overlap.

Suppose we have a sampling order $(a_1, a_2, \ldots, a_n)$ for $V(n)$ and a sampling order $(b_1, b_2, \ldots, b_m)$ for $V(m)$, the new partners of a_n in nodes from T_{n+m} to T_{n+1} in the merger are the elements in reverse sampling order, $(b_m, \ldots, b_1)$. After marginalizing element a_n from the merger, the new partners of a_{n-1} in nodes from T_{n-1+m} to T_n in the merger will be the elements $(b_m, \ldots, b_1)$. In general, the new partners of a_i will be chosen following the reverse sampling order, $(b_m, \ldots, b_1)$, from T_{i+m} to T_{i+1}. Similarly the partners of b_j will be $(a_n, \ldots, a_1)$ from T_{j+n} to T_{j+1} in the merger. A general algorithm is as follows (which appears in the proof of Theorem 5.2 in Cooke et al. (2015)),

Algorithm 12 Merging two regular vines $V(n)$ and $V(m)$ without overlapping elements

Input: regular vine $V(n)$ and $V(m)$ where $\mathcal{V}_n \cap \mathcal{V}_m = \emptyset$
Output: A merger $V(\mathcal{V}_n \cup \mathcal{V}_m)$

1: Determine a sampling order $SO_n = (a_1, \ldots, a_n)$ for $V(n)$.
2: Determine a sampling order $SO_m = (b_1, \ldots, b_m)$ for $V(m)$.
3: The merger is constructed based on SO_n and SO_m where the new partners of a_i from T_{i+m} to T_{i+1} will be $(b_m, \ldots, b_1)$, and the new partners of b_j from T_{j+n} to T_{j+1} will be $(a_n, \ldots, a_1)$.

The example to illustrate Algorithm 12 is presented in Appendix 6.A.1. Extending a regular vine by just one distinct element, introduced in Nápoles (2010), is a special case of merging two vines without overlap. In this case, one vine is just a single element and its sampling order is this element itself.

In the case when $V(n)$ and $V(m)$ have a common subvine $V(p)$, $\mathcal{V}_p \neq \emptyset$, the tree structures up to and including T_p of the merger are fixed, hence only the segment $(a_{p+1},\dots,a_n)$ and $(b_{p+1},\dots,b_m)$ will contribute to the construction of the merger when Algorithm 12 is applied. However, not all sampling orders lead to a valid merger. One extra constraint that both the initial segments $(a_1,\dots,a_p)$ and $(b_1,\dots,b_p)$ are sampling orders of the common subvine $V(p)$ should be satisfied (a detailed proof can be seen in Appendix 6.A.2). This constraint guarantees that the construction of the merger from T_{n+m-p} to T_{p+2} follows a vine structure construction for merger without overlap, and a valid tree structure construction for T_{p+1}, hence a valid merger can be constructed according to Algorithm 12.

As an example, we show how to merge vine $V_1(5) = V(\{1,2,3,4,5\})$ and another vine denoted as $V'(5) = V(\{2,3,6,7,8\})$, which share a common subvine on elements $\{2,3\}$. These two vines are shown in Figure 6.8 (black).

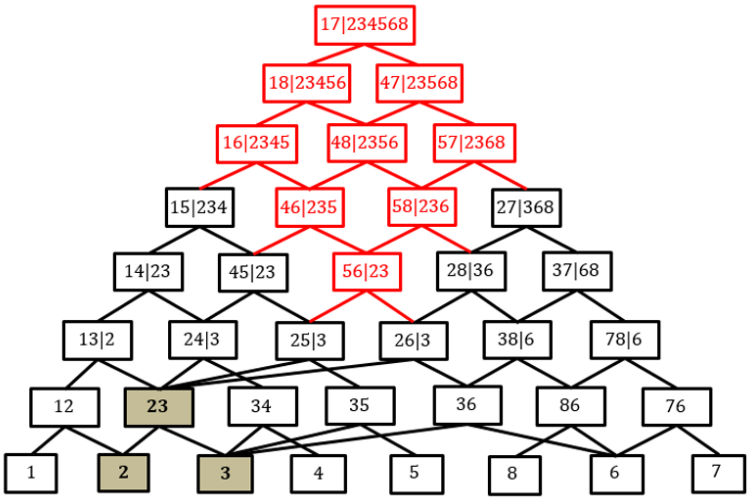

Figure 6.8: A merger of vine $V_1(5)$ ($V(\{1,2,3,4,5\})$) (black) and $V'(5)$ ($V(\{2,3,6,7,8\})$ (black). The nodes in the common subvine $V(\{2,3\})$ is denoted in shaded area and the new nodes and lines in the vine triangular array are marked in red.

Following Algorithm 12 with extra conditions on the sampling orders, one possible merger of $V_1(5)$ and $V'(5)$ is shown in Figure 6.8, where the sampling order is $(2,3,5,4,1)$ or $(3,2,5,4,1)$ in $V_1(5)$ and $(2,3,6,8,7)$ or $(3,2,6,8,7)$ in $V'(5)$. Other choice for the segment (a_3,a_4,a_5) in $V_1(5)$ can be $(4,5,1)$, $(4,1,5)$ or $(1,4,5)$. There is no other choice for the segment (b_3,b_4,b_5) in $V'(5)$. Hence in total there are four possible mergers of $V_1(5)$ and $V'(5)$.

In general it is unknown how many mergers can be constructed. The number of mergers depends on the structures of both vines and the overlapping elements. In this chapter, we follow the bottom-up approach in Algorithm 11 when we want to merge two vines with overlap (we expect differences in the performance of both algorithms but this problem will not be researched here). The merger given in Figure 6.8 is constructed by

connecting nodes 25|3 and 26|3 in T_3. Other choice can be to connect 13|2 and 26|3 (leading to another merger) or to connect 24|3 with 26|3 (leading to two extra mergers).

In the examples above we have explained how two vines can be merged. However, when there are more overlapping vines that need to be merged, the order of merging should be specified so the merging is done consistently. We show in Figure 6.9 one example where three vines $V(\{1,2,3\})$, $V(\{1,3,5\})$ and $V(\{1,4,5\})$ (denoted as $V_1(3)$, $V_2(3)$ and $V_3(3)$, respectively) are to be merged.

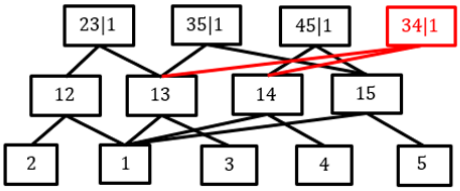

Figure 6.9: Three regular vines $V(\{1,2,3\})$ ($V_1(3)$), $V(\{1,3,5\})$ ($V_2(3)$) and $V(\{1,4,5\})$ ($V_3(3)$) (black) and one possible way of merging $V_1(3)$ and $V_3(3)$ (red).

If we were to merge $V_1(3)$ and $V_3(3)$ first, setting the new node in T_3 of $V\{1,2,3,4,5\}$ to be 34|1 (in red) instead of 35|1, as shown in Figure 6.9, then their merger will not have a common subvine with $V_2(3)$. By choosing right a correct order of merging these vines the problem above can be avoided. In the example above, we should merge $V_1(3)$ and $V_2(3)$ first and then $V_3(3)$, or to merge $V_2(3)$ and $V_3(3)$ first and then $V_1(3)$. In Section 6.3.4, we will provide a procedure to order the merging process when an m-vine corresponding to a strong clique tree is constructed.

6.3.4. CONSTRUCTION OF AN M-VINE CORRESPONDING TO A STRONG CLIQUE TREE

In this section, an algorithm to construct an m-vine corresponding to a strong clique tree is presented. Regular vines on elements of each node in the strong clique tree $\mathcal{T}$ are constructed and they are constructed consistently such that they contain common subvines. This is done by following, in reverse order, the sequence of trees $\mathcal{T}^{(i)}, i = 0,\dots,k$ obtained for $\mathcal{T}$. The nodes of these trees determine constraint sets of nodes that have to appear at different trees in a regular vine. The elements in a node in $\mathcal{T}^{(i)}$ are either in its separator(s) or do not appear in any node of $\mathcal{T}^{(j)}, j \geq i$, which we refer to as **extra elements**.

First, an example of the approach will be presented. The m-vine corresponding to the strong clique tree $\mathcal{T}$ in Figure 6.6 (black) can be constructed as follows:

The sequence of trees $\mathcal{T}^{(i)}, i = 1,\dots,3$ are in Figure 6.7 (left). We follow these trees in reverse order starting with $\mathcal{T}^{(3)}$.

Step 1: In $\mathcal{T}^{(3)}$ there is only one node, which has no separator (the process is just started) and has an extra element 1. Hence, a vine $V(\{1\})$ is constructed.

Step 2: In $\mathcal{T}^{(2)}$ we have $\mathcal{C}_1^{(2)} = \{1\}$ and $\mathcal{C}_2^{(2)} = \{1,3\}$. Node $\mathcal{C}_1^{(2)}$ has a separator $\{1\}$ and no extra element, hence the vine is $V(\{1\})$. Node $\mathcal{C}_2^{(2)}$ has a separator $\{1\}$ and an

extra element 3, then the vine for this node is obtained through extending $V(\{1\})$ by element 3, to get $V(\{1,3\})$.

Step 3: In $\mathcal{T}^{(1)}$, there are three nodes $\mathscr{C}_1^{(1)} = \{1,2\}, \mathscr{C}_2^{(1)} = \{1,3\}$ and $\mathscr{C}_3^{1} = \{1,3\}$. Node $\mathscr{C}_1^{(1)}$ has separator $\{1\}$ and an extra element 2, hence we obtain $V(\{1,2\})$ by extending $V(\{1\})$ with element 2. In the case of node $\mathscr{C}_2^{(1)}$, we see that it has separators $\{1\}$ and $\{1,3\}$. Separator $\{1\}$ is a subset of $\{1,3\}$, hence only separator $\{1,3\}$ will be considered. Since there is no extra element its corresponding vine is $V(\{1,3\})$. Node $\mathscr{C}_3^{(1)}$ has separator $\{1,3\}$ and no extra element so its corresponding vine is $V(\{1,3\})$.

Step 4: In $\mathcal{T}^{(0)}$ corresponding vines for nodes $\{1,3,5\}$, $\{1,2,6\}$ and $\{1,3,7\}$ can be constructed similarly as above. Node $\{1,2,3,4\}$ has two separators, $\{1,2\}$ and $\{1,3\}$, with their corresponding vines $V(\{1,2\})$ and $V(\{1,3\})$, respectively, and an extra element 4. A vine $V(\{1,2,3\})$ can be constructed by at first merging $V(\{1,2\})$ and $V(\{1,3\})$ and then by extending with element 4 to get $V(\{1,2,3,4\})$. The last step is to combine the regular vines in $\mathcal{T}^{(0)}$ to get the m-vine $V(\{1,2,6\},\{1,3,5\},\{1,2,3,4\},\{1,3,7\})$.

The general algorithm to construct an m-vine corresponding to a strong clique tree $\mathcal{T}$ is presented in Algorithm 13.

Algorithm 13 Construction of an m-vine for a strong clique tree $\mathcal{T}$

Input: A tree sequence $\mathcal{T}^{(i)}, i = 0,\ldots,k$ where $\mathcal{T}^{(0)} = \mathcal{T}$
Output: An m-vine corresponding to the strong clique tree $\mathcal{T}$

1: **for** i from k to 0 **do**
2: **repeat**
3: Find the separators of $\mathscr{C}_j^{(i)}$ which are not subsets of the others.
4: **if** no separator found **then**
5: Construct a regular vine $V(\mathscr{C}_j^{(i)})$ for the node $\mathscr{C}_j^{(i)}$.
6: **else**
7: Find the vine structures corresponding to the separators.
8: Combine the vines corresponding to separators and the extra elements (if applicable).
9: The vine after merging and extending by extra elements is $V(\mathscr{C}_j^{(i)})$.
10: **end if**
11: **until** all nodes in $\mathcal{T}^{(i)}$ have been considered
12: **end for**
13: Combine the vine structures corresponding to the nodes in $\mathcal{T}^{(0)}$ to get the m-vine.

As discussed in the end of Section 6.3.3, a correct order of merging more than two regular vines is needed. This can be implemented in line 8 in Algorithm 13 where more than two separators of node $\mathscr{C}_j^{(i)}$ are found. These separators are nodes in tree $\mathcal{T}^{(i+1)}$. We propose to merge them following the known depth-first search algorithm of trees. That is, to choose randomly one separator as a root of the tree and then order separators by applying a depth-first algorithm in $\mathcal{T}^{(i+1)}$. The detailed algorithm can be found in Algorithm 15 in the Appendix 6.A.5. Due to the property of strong clique tree, the merging of the separators this way is always consistent.

Theorem 6.3.3 below ensures that Algorithm 13 produces a consistent m-vine. We show that subvines corresponding to each node in $\mathcal{T}^{(i)}, i = 0,\dots,k$ are constructed such that if they have overlapping elements they must share a common subvine on the overlap.

Theorem 6.3.3. *Let $\mathscr{C}_m$ and $\mathscr{C}_n$ be nodes in the tree sequence $\mathcal{T}^{(i)}, i = 0,\dots,k$ and let $\mathscr{C}_p = \mathscr{C}_m \cap C_n \neq \emptyset$. Regular vines $V(\mathscr{C}_m)$ and $V(\mathscr{C}_n)$ constructed according to Algorithm 13 share a common subvine $V(\mathscr{C}_p)$.*

Proof. Suppose nodes $\mathscr{C}_m$ and $\mathscr{C}_n$ are in tree $\mathcal{T}^{(i_m)}$ and $\mathcal{T}^{(i_n)}$ respectively, where $i_m \leq i_n$. It suffices to find a node with elements $\mathscr{C}_p$ in the tree sequence. If $i_m = i_n$, then there is a path through nodes $\mathscr{C}_m$ and $\mathscr{C}_n$ and the overlapping elements $\mathscr{C}_p$ appear in at least one separator of $\mathscr{C}_m$ or $\mathscr{C}_n$ (if the separator is $\mathscr{C}_p$ then node is found). These separators are the nodes in tree $\mathcal{T}^{(i_m+1)}$ and there must be a path through these separators. Similarly, their separators containing elements $\mathscr{C}_p$ are the nodes in $\mathcal{T}^{(i_m+2)}$ and a path through them exists as well. Applying this argument iteratively, we can finally find a node in a certain tree level containing only elements $\mathscr{C}_p$. The common subvine $V(\mathscr{C}_p)$ of $V(\mathscr{C}_m)$ and $V(\mathscr{C}_n)$ is built at that stage and will not change anymore during the construction. When $i_m < i_n$, the search procedure starts with the separator(s) of $\mathscr{C}_m$ that contains elements $\mathscr{C}_p$. This separator will finally result in node(s) that includes the elements $\mathscr{C}_p$ in $\mathcal{T}^{(i_n)}$, otherwise the property that $\mathcal{T}^{(i)}_{\{v\}}$ is connected is violated. Similarly, as in the case when $i_m = i_n$, a node $\mathscr{C}_p$ and the corresponding common subvine $V(\mathscr{C}_p)$ can be found. □

In the next section we show how the theoretical results presented above can be applied in practice.

6.4. APPLICATIONS OF THE EQUIVALENCE BETWEEN M-VINES AND STRONGLY CHORDAL GRAPHS

The theoretical result presented in the previous section provides a flexible method that estimates a distribution with strongly chordal pattern of conditional independence. Using our result one construct a variety of distributions having a specified set of conditional independence and different properties, e.g. correlation structures, asymmetries, tail dependence. This can be valuable, e.g. in simulation studies that evaluate the performance of conditional independence tests in a non-Gaussian environment. Moreover one can take an advantage of existing software implementations of vine copula models.

However, in this section we concentrate on a different application of our result. It is known that the number of regular vine structures on n elements grows exponentially with dimensions. Hence estimating all vine structures for the given data is infeasible. Simplifying the model by assuming a pattern of conditional independence to be presented in the data, represented by a strongly chordal graph, makes the estimation effort manageable. The number of m-vine structures constructed following Algorithm 13 is much smaller (however the exact number of m-vine structures is unknown). For clique trees containing a small enough number of elements in maximal nodes, one can assess all vine structures for the subvines corresponding to the maximal nodes. Furthermore one can even estimate a non-simplified regular vine.

In this section, we first analyze in detail two simulated examples where the advantages of simplifying the vine copula model are highlighted. In high dimensions when it is not possible to estimate all possible m-vines, a heuristic method to choose an m-vine structure for a given data is needed. We propose a general heuristic of constructing m-vine structures from data, along with a simulation study to evaluate its performance.

The following procedure describes how to obtain a strongly chordal graph from data. For a given data transformed into its z-scale, the strongly chordal graph is constructed as follows: a graph is chosen at first using function `selectFast` in the **GGMselect** package (Giraud et al. (2009)) (the best graph in a family of graphs generated by the method introduced in Meinshausen & Bühlmann (2006), Wille & Bühlmann (2006) is chosen), if this graph is not chordal, necessary edges are added to make it chordal; if this graph is not strongly chordal (according to Theorem 6.3.1), more edges are added based on the algorithm in Mukhopadhyay & Rahman (2021).

6.4.1. SIMULATED EXAMPLES

An m-vine on 10 elements is used in simulations . The maximal nodes are {2}, {6}, {1,3,7,8} and {1,4,5,9,10}. Details about the simulated vine are presented in Table 6.1 (in this example, we include commas to separate elements in the nodes of the vine and in the nodes of the clique tree), where we show the m-vine structure, the copula family C and the corresponding Kendall's tau τ. In order to amplify the effect of the vine structure, the copula families are chosen from Clayton copula (C), Gumbel copula (G) and their rotated versions. The vine triangular array for this m-vine is shown in Appendix 6.A.3.

structure	C	τ	structure	C	τ	structure	C	τ
Tree-4								
9,5|1,4,10	C270	-0.60						
Tree-3								
5,10|1,4	C180	0.48	1,7|3,8	G270	-0.34	9,10|1,4	C270	-0.67
Tree-2								
4,5|1	G180	0.74	3,7|8	C90	-0.68	1,9|4	C90	-0.50
1,3|8	C90	-0.70	1,10|4	G270	-0.88			
Tree-1								
1,5	G270	-0.33	7,8	C	0.61	4,9	G180	0.49
3,8	G	0.50	4,10	G	0.54	1,4	G180	0.56
1,8	C180	0.64						

Table 6.1: An m-vine $V(\{2\},\{6\},\{1,3,7,8\},\{1,4,5,9,10\})$.

ABILITY TO ASSESS ALL POSSIBLE VINE STRUCTURES

The number of vines grows exponentially as dimension grows. However in the m-vine approach, we may be able to find the best vine structure for a maximal clique with relatively small number of variables. We show below an example.

We simulate 1000 samples from the m-vine. The strong clique tree from the data is as follows,

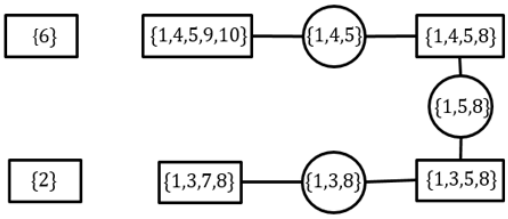

Figure 6.10: The strong clique tree estimated by `selectFast` for the data simulated from the m-vine with 1000 samples.

The graph given by the function `selectFast` is already a strongly chordal graph and we observe that it does not exactly recover the clique tree of the true m-vine (This is not surprising as the methods GGMselect assumes gaussian data.). Extra maximal cliques appear in the estimated tree, $\{1,4,5,8\}$ and $\{1,3,5,8\}$. However, we show below that one can still find the true structure (copula families can be different) for subvine $V(\{1,4,5,9,10\})$ and $V(\{1,3,7,8\})$.

According to Algorithm 13, $V(\{1,4,5,9,10\})$ is constructed by extending $V(\{1,4,5\})$ with variable 9 and 10, and $V(\{1,3,7,8\})$ is constructed by extending $V(\{1,3,8\})$ with variable 7. Vine $V(\{1,4,5\})$ requires to contain node 1,5, hence only two possible structures are available. The best structure is the true structure which has the smallest $AIC = -3207.56$ (also smallest $BIC = -3183.03$). When we fix the best structure for $V(\{1,4,5\})$, there are 48 possible vine structures for $V(\{1,4,5,9,10\})$ and it is easy to estimate all of them. The one that has the smallest $AIC = -12394.17$ (also smallest $BIC = -12330.37$) is the true structure for these variables. Similarly, for $V(\{1,3,7,8\})$, its subvine $V(\{1,3,8\})$ has two possible structures with node 1,8 included. The best $V(\{1,3,8\})$ is the one we simulated from and has $AIC = -4202.58$ ($BIC = -4182.95$). Then extending $V(\{1,3,8\})$ by variable 7 leads to estimating four possible structures and the best one is the true one $V(\{1,3,7,8\})$ ($AIC = -7638.84$ and $BIC = -7599.58$).

The remaining maximal cliques $\{1,4,5,8\}$ and $\{1,3,7,8\}$ are determined once $V(\{1,4,5\})$ and $V(\{1,3,8\})$ are fixed (In general finding the best vine for each maximal clique does not guarantee a globally optimal m-vine). We compare the m-vine estimated following the procedure above with the true m-vine and observe that the contribution to AIC and BIC of the extra nodes $5,8|1,3,5|1,8$ and $4,8|1,5$ in the constructed m-vine is very small. Following Dißmann's heuristic in Dißmann et al. (2013), Brechmann et al. (2012), a 5-truncated vine is constructed. We see in Table 6.2 that the m-vine we constructed is significantly better than the truncated vine. However, even when the true structure is retrieved, due to estimation error from the tree-wise estimation procedure and the limited choice of parametric copula families, the constructed m-vine performs worse than the true m-vine for the data.

				Vuong test p-value	
	AIC	*BIC*	true	m-vine	5-trun
true	-20438.98	-20360.46		2.83E-15	3.02E-117
m-vine	-20036.67	-19913.97	4.48E-20		1.77E-110
5-trun	-18134.05	-18016.27	7.03E-123	1.08E-109	

Table 6.2: The *AIC*, *BIC* and the p-value from a Vuong test for the true m-vine (true), the m-vine we constructed (m-vine) corresponding to the strong clique tree in Figure 6.10 and a 5-truncated vine constructed by Dißmann's heuristic (5-trun). The upper diagonal in the Vuong test shows the result without Schwarz correction while the lower diagonal shows the result with Schwarz correction.

ABILITY TO ASSESS NON-SIMPLIFIED MODELS FOR SUBVINES

Non-simplified vines are difficult to construct and estimate. There are a few attempts to handle this problem, which propose to estimate the relationship between the copula parameter (or the corresponding conditional Kendall's tau) and the conditioning variables non-parametrically (Acar et al. (2011)) or to use generalized additive models (GAM) (Nagler & Vatter (2020)). A discussion about the performance of these two methods can be found in Acar et al. (2019). These approaches are in principle feasible only in low dimensions. In this chapter, we show an example using the latter method which is implemented in the **gamCopula** package.

The non-simplified regular vine is constructed based on the same m-vine as in Table 6.1. We replace the constant conditional copula by assuming a relationship between the conditional Kendall's tau and the conditioning variables in the subvines $V(\{1,4,5\})$, $V(\{1,4,10\})$ and $V(\{1,4,9\})$ as follows: $\tau = \frac{e^z-1}{e^z+1}$, where $z = -\beta_0 + \beta_1 u$ and u denotes the conditioning variable. The copula families are adjusted to include their rotated versions (the copula family for node $4,5|1$ ($\beta_0 = -2.77, \beta_1 = 5.70$) includes G180 and G270, for node $1,10|4$ ($\beta_0 = -1.63, \beta_1 = 3.28$) includes G and G270 and for node $1,9|4$ ($\beta_0 = -1.27, \beta_1 = 2.34$) includes C180 and C90). Again 1000 samples are simulated from this m-vine and the resulting strong clique tree presented in Figure 6.11.

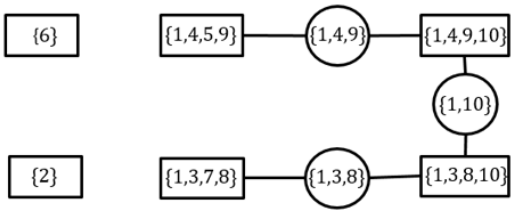

Figure 6.11: The strong clique tree for the data simulated from a non simplified m-vine of size 1000 by adding edges 19 and 1,10 to the estimated graph from `selecFast`.

Edges 1,9 and 1,10 are added in order to make the graph strongly chordal. We see that in the estimated tree nodes 5 and 10 as well as 4 and 10 are not connected. Hence the true structure for $V(\{1,4,5,9,10\})$ cannot be found. Similar, as in the previous example, we can still find the best vine for each maximal clique. The maximal clique $\{1,4,5,9\}$ requires to construct $V(\{1,4,9\})$ first. There are three possible vine structures for $V(\{1,4,9\})$ and we first estimate a simplified vine for each structure. Then we apply the `pacotest`

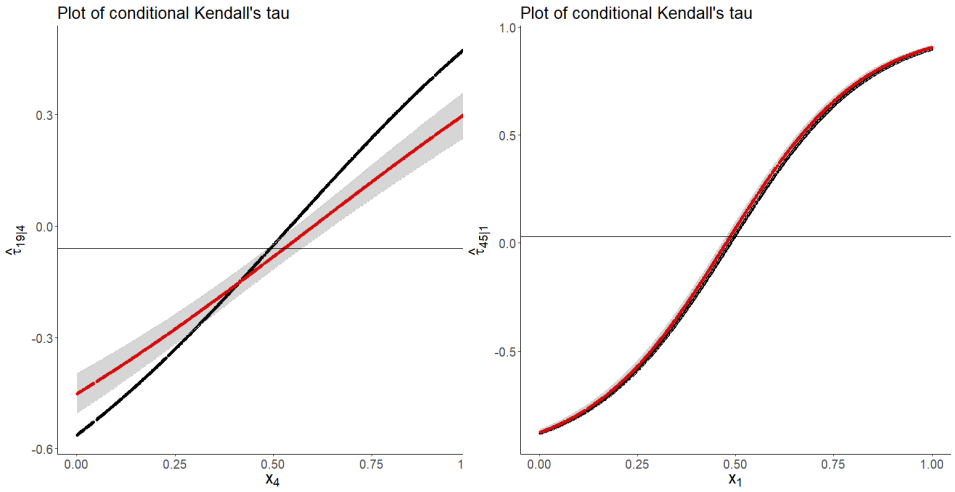

Figure 6.12: Kendall's tau for the conditional copula in vine $V(\{1,4,9\})$ (left) and in vine $V(\{1,4,5\})$ (right). The true conditional tau is marked in black points and the estimated tau is marked in red points with its 95% confidence interval. The horizontal line denotes the conditional tau estimated by a simplified vine ($\tau = -0.06$ (left) and $\tau = 0.02$ (right)).

6

(Kurz (2019)) to check whether the simplifying assumption is rejected. If it is rejected, we fit a non-simplified vine with the GAM method instead. All the three structures are rejected to be a simplified vine, and the best non-simplified vine ($AIC = -1977.96$ and $BIC = -1953.42$) is the one with the true vine structure (the number of parameters in a non-simplified vine includes, 1) the number of parameters of the simplified bivariate (conditional) copulas, 2) the number of degrees of freedom in the GAM model contained in the non-simplified bivariate conditional copulas. 3) the number of the second parameters in the non-simplified bivariate conditional copulas if there are any). The relationship between the conditional tau and the conditioning variable is shown in Figure 6.12 (left). The estimated non-simplified conditional copula is a Student-t copula with degrees of freedom equal to 15.33 and the coefficients are $\hat{\beta}_0 = -0.97, \hat{\beta}_1 = 1.62$.

Fixing the best non-simplified vine $V(\{1,4,9\})$ gives four possible structures for $V(\{1,4,5,9\})$. The best structure is to connect 5 and 1 in the first tree. A non-simplified vine is estimated which has $AIC = -3588.35$ and $BIC = -3517.44$. Furthermore, the non-simplified conditional copula $C_{45;1}$ can be estimated quite well, as shown in Figure 6.12 (right). The vine structure for the maximal clique $\{1,4,9,10\}$ is fixed once $V(\{1,10\})$ [1] is determined, which leads to a non-simplified vine $V(\{1,4,9,10\})$ with $AIC = -3116.49$ and $BIC = -3023.53$. The remaining two maximal cliques require to construct $V(\{1,3,8\})$ at first. Similarly there are three possible structures and for each structure a simplified vine is estimated or a non-simplified vine is estimated instead decided by the `pacotest`. Two of them are non-simplified vines but the best one is a simplified vine having the

[1] the **gamCopula** package does not allow the $BB6$ family copula ($AIC = -506.61$, $BIC = -496.79$) to be chosen which for this data has better AIC than the Gumbel copula ($AIC = -505.73$, $BIC = -500.82$). Hence we fix this copula to be Gumbel also in the later simplified vine $V(\{1,3,8,10\})$

true structure ($AIC = -3908.47$ and $BIC = -3893.75$). Once $V(\{1,3,8\})$ is determined $V(\{1,3,8,10\})$ is fixed. $V(\{1,3,7,8\})$ is constructed by extending $V(\{1,3,8\})$ with variable 7 and there are four possibilities to consider. Two of them are rejected to be simplified vines hence non-simplified vines are estimated instead. The best $V(\{1,3,7,8\})$ is a simplified vine with its true structure ($AIC = -7139.28$ and $BIC = -7104.93$).

The Dißmann's heuristic gives a 4-truncated vine. Information about the performance of the models is provided in Table 6.3 and we see that the m-vine approach is significantly better.

			Vuong test p-value		
	AIC	*BIC*	true	m-vine	4-trun
true	-11865.60	-11772.35		0.775	2.02E-13
m-vine	-11865.28	-11688.48	0.494		1.64E-45
4-trun	-10812.19	-10714.04	1.40E-13	2.36E-37	

Table 6.3: The *AIC*, *BIC* and the p-value from a Vuong test for the true m-vine model (true), the m-vine we constructed (m-vine) corresponding to the strong clique tree in Figure 6.11 and a 4-truncated vine constructed by Dißmann's heuristic (4-trun). The upper diagonal in the Vuong test shows the result without Schwarz correction while the lower diagonal shows the result with Schwarz correction.

Since we are not able to recover the true vine structure due to the restriction caused by the constructed strongly chordal graph, the dependence between X_1 and X_{10} conditional on X_4 cannot be estimated well, which is shown in Appendix 6.A.4.

In the two simulated examples above, we have explained that by applying the m-vine approach we are able to focus on estimating its subvines on small number of variables and can improve the model estimation. Furthermore when some maximal cliques share no separators or a separator with one element only, their corresponding vine structure can be estimated separately. This leads to a great reduction of computational effort when data from a sparse model is considered.

However, it is also very clear that if our assumptions of conditional independence structure present in the data are not correct the performance of our method will not be optimal.

6.4.2. HEURISTICS

Algorithm 13 explains how the m-vine corresponding to a given strong clique tree can be constructed. There are still many possible m-vine structures and, in general, it is not known how many there are. In this section, we will design a heuristic search for the 'best' m-vine structure for the data. This will be achieved by specifying information on how choices should be made in Algorithm 13 in every step where a choice is possible. We discuss all these choices below.

- *In line 5 in algorithm 13, the method of constructing a regular vine on given elements is required.*
  At this point we will use Dißmann's heuristic where a vine structure is constructed tree by tree and each tree structure is determined by the maximum spanning tree with absolute value of Kendall's tau being the edge weight. Hence in the case of only one maximal clique, the m-vine approach coincides with Dißmann's heuristic.

- *In line 8 in algorithm 13, the way of combining vines corresponding to separators and the extra elements needs to be provided.*
  First we merge the vines for the separators and then extend the merger by the extra elements. The order of merging separators is determined by Algorithm 15. We require two choices at this point.

  1. The choice of the sampling order for the regular vines in Algorithm 12 when vines without overlap are merged and tree structures for trees higher than T_p when merging vines with overlap.

     When vines do not overlap, the sampling order is chosen based on empirical multiple correlation (Yule & Kendall (1965)). For n variables and their empirical correlation matrix Σ of normal scores, the empirical multiple correlation of variable i is $R^2_{i\{\Sigma\}} = 1 - \frac{det(\Sigma)}{det(\Sigma^{ii})}$, where Σ^{ii} denotes the matrix Σ without ith row and ith column and det is the determinant.

     When merging $V(n)$ and $V(m)$ where $\mathcal{V}_n \cap \mathcal{V}_m = \emptyset$, the sampling order $SO_n = \{a_1, \ldots, a_n\}$ is determined in reverse. From the top node of $V(n)$ there are two variables in the conditioned set denoted as v_n, v_{n-1}. The empirical multiple correlation for these two variables are denoted by $R^2_{v_n\{\Sigma\}}$ and $R^2_{v_{n-1}\{\Sigma\}}$ respectively, where Σ is the empirical correlation matrix for variables $\mathcal{V}_n \cup \mathcal{V}_m$. Since a_n will appear in the top node of the merger, it's more reasonable to choose a_n with smaller correlation in higher trees. When the value of a_n is determined, $V(n)$ is marginalized by a_n and we get a subvine with conditioned set $v_{n-1}v_{n-2}$ in its top node. The variable a_n will be removed from the matrix Σ. By successively applying this approach we get a reverse sampling order $(a_n, \ldots, a_1)$. The sampling order of $V(m)$ can be similarly determined. The algorithm below concludes the above procedure.

Algorithm 14 Determine a sampling order of $V(n)$ during merging with $V(m)$

Input: A regular vine $V(n)$ and empirical correlation matrix $\Sigma(\mathcal{V}_n \cup \mathcal{V}_m)$
Output: A sampling order $SO_n = (a_1, \ldots, a_n)$

1: **for** i from n to 2 **do**
2: Calculate the empirical multiple correlation $R^2_{v_i\{\Sigma\}}$ and $R^2_{v_{i-1}\{\Sigma\}}$ for the variables v_i, v_{i-1} in the conditioned set of the top node of $V(n)$.
3: a_i is chosen to be the variable with smaller empirical multiple correlation.
4: $V(n)$ is updated by marginalizing a_i.
5: Σ is updated by removing the row and the column of a_i.
6: **end for**

     When merging $V(n)$ and $V(m)$ with overlapping elements $\mathcal{V}_p \neq \emptyset$, the heuristic to determine the tree structure higher than T_p is the same as Dißmann's heuristic. A weight (absolute value of Kendall's tau) is assigned to the possible edges and the tree structure is chosen to be the maximum spanning tree.

  2. When there is only one extra element, then vine structure can be constructed by extending a vine by this element as explained in Algorithm 12. However, when there are more extra elements, we must decide in which order they should be added. We

propose to construct a vine on extra elements using Dißmann's heuristic and merge these vines using the approach explained in the previous point.

Note that due to the choices we made, not all m-vine structures are considered. This is mainly due to the choice we made in the second point. For example in Figure 6.10, the vine $V(\{1,4,5,9,10\})$ is constructed by at first fixing a vine structure for $V(\{1,4,5\})$ and then merging with $V(\{9,10\})$, which accounts for only 8 possibilities. This choice eliminates 40 possible vine structures for $V(\{1,4,5,9,10\})$.

6.4.3. SIMULATION

A simulation study is presented in this section to show the performance of the proposed heuristic when compared to the Dißmann's heuristic. The data is simulated from a regular vine copula model. The vine structure is randomly constructed by the function `RVineMatrixSample`. Its bivariate copula families are chosen from Gumbel(G), Clayton(C) and Joe(J) copulas as well as their rotated versions. The copula parameters are obtained from Kendall's tau which are simulated from a *Beta(2,2)* distribution. In order to distinguish from the independence copula, all Kendall's tau smaller than 0.2 are rounded off to 0.2 (according to the test in Genest & Favre (2007) this 0.2 Kendall's tau is rejected to be 0 for 300 or 1000 sample size). The Kendall's tau is also allowed to take negative values with probability 0.5.

The conditional independence are introduced by at first determining a truncation level k and then replacing randomly the bivariate copulas in tree T_k with independence copula. The dimension of the simulated model is 10, 20 and 30 respectively and the sampled data size is 300 or 1000. The proportions of the independent copulas in vine models are chosen to be 30% (leading to a not very sparse model) and 70% (sparse model). Hence, for dimension 10 there are 13 ($\lfloor 45 * 0.3 \rfloor$) or 31 ($\lfloor 45 * 0.7 \rfloor$) independence copulas assigned, for dimension 20 there are 57 (190*0.3) or 133 (190*0.7) independence copulas, and for dimension 30 there are 130 ($\lfloor 435 * 0.3 \rfloor$) or 304 ($\lfloor 435 * 0.7 \rfloor$) independence copulas.

For each simulated data, we apply 1) **method-1**, the Dißmann's heuristic to get a truncated vine; 2) **method-2**, the procedure introduced in the previous section (estimate a graph by `selectFast`, make it strongly chordal if needed) and the heuristic to get an m-vine from the estimated strongly chordal graph; 3) **method-3**, the heuristic directly on the strongly chordal graph from the simulated m-vine model (which is a graph by connecting all the variables in the constraint set of each maximal node of the m-vine respectively). We then calculate the AIC and BIC for the vines estimated by these three approaches. Furthermore, we apply a Vuong test (without or with Schwarz correction) to the estimated vines to see whether one vine is significantly better than the other. This procedure is repeated 100 times. The results are shown in Table 6.4.

Dim	%	size		$\#imp$	ΔAIC	ΔBIC	$\#imp_{Schwarz}$	$\Delta AIC_{Schwarz}$	$\Delta BIC_{Schwarz}$	$\#OVLP$
10	30	300	M-2	71(33)	267.88	262.94	71(25)	345.71	342.30	17(18)
			M-3	80(67)	1129.93	1175.32	84(70)	1088.23	1133.58	
		1000	M-2	81(34)	586.96	585.51	81(32)	622.37	621.91	7(5)
			M-3	84(76)	4067.88	4162.23	86(80)	3877.10	3970.65	
	70	300	M-2	49(30)	352.33	354.67	49(30)	352.33	354.67	7(7)
			M-3	96(84)	448.61	467.48	96(86)	439.47	458.55	
		1000	M-2	56(39)	901.21	908.88	54(39)	904.02	912.58	8(7)
			M-3	92(89)	1545.87	1588.94	96(89)	1545.87	1588.94	
20	30	300	M-2	84(73)	406.15	193.32	74(29)	609.99	380.35	15(16)
			M-3	59(40)	1604.54	1744.54	64(50)	1330.61	1474.39	
		1000	M-2	72(30)	441.84	436.44	77(30)	450.86	450.86	8(11)
			M-3	61(50)	4408.98	4781.87	64(56)	3983.06	4356.40	
	70	300	M-2	31(18)	925.64	883.16	29(18)	926.89	894.39	3(0)
			M-3	99(96)	2943.62	3249.96	100(97)	2915.97	3222.12	
		1000	M-2	81(71)	842.96	641.15	75(37)	1341.46	1170.62	1(0)
			M-3	99(98)	12268.93	12979.95	99(99)	12147.05	12857.28	
30	30	300	M-2	82(73)	599.97	189.19	59(8)	997.39	420.53	9(18)
			M-3	47(25)	2036.46	2226.09	58(34)	1604.84	1813.34	
		1000	M-2	25(13)	747.70	678.93	22(14)	701.45	663.88	16(19)
			M-3	67(45)	442.24	302.50	41(11)	1141.46	897.68	
	70	300	M-2	49(42)	574.41	231.53	34(3)	1039.37	491.21	10(7)
			M-3	64(57)	3158.75	3957.02	82(71)	2592.34	3376.34	
		1000	M-2	94(90)	883.60	390.24	84(32)	1341.17	771.41	2(6)
			M-3	73(72)	12479.21	14267.61	79(73)	12312.03	14101.68	

Table 6.4: Simulation result in dimension 10, 20 and 30 respectively with proportion of independence copula being 30% or 70%, with 300 or 1000 data size. M-2 denotes method-2 and M-3 denotes method-3. $\#imp$ denotes the number of times the estimated m-vine and the truncated vine from method-1 are not significantly different according to the Vuong test or the estimated m-vine is significantly better (the number for this case is shown in brackets). ΔAIC and ΔBIC denotes the average absolute AIC and BIC improvement when the estimated m-vine is significantly better than the truncated vine. $\#imp_{Schwarz}$, $\Delta AIC_{Schwarz}$ and $\Delta BIC_{Schwarz}$ correspond to the same results of the Vuong test with Schwarz correction. $\#OVLP$ denotes the number of times the m-vines from method-2 and method-3 are not significantly different according to the Vuong test (the number in brackets correspond to the Vuong test with Schwarz correction).

We draw several conclusions from the simulation results.

- The performance of method-2 is comparable to method-1 when the simulated data is not very sparse (30% proportion of independence copula) in dimension 10 and 20. However, when the simulated data is sparse (70% proportion of independence copula) method-2 performs worse than method-1 especially when the dimension is high and the data size is small. This is because the data is simulated from a non-Gaussian distribution which violates the Gaussian assumption used in the function `selectFast`. Moreover, when smaller data size is generated, this function is less likely to capture the conditional independence well especially in high dimensions. According to the Vuong test, the m-vines from method-2 and method-3 are less likely to have similar performance when the underlying data is sparse;

- when the underlying strongly chordal graph is correctly captured in method-3, the

performance is much better than method-1 when the simulated data is sparse. However in dimension 20 when the underlying data is not very sparse or in dimension 30, the performance of method-3 is not as good as in other cases. This is due to the choice of our heuristic vine structure constructions, where lots of vine structures are not considered (as discussed in the example in Section 6.4.2). Even though the vine structures are not well chosen in general, one can still find some vine structures with much better AIC or BIC, as shown in Table 6.4 where the average improvement in AIC and BIC is large.

6.5. REAL DATA ANALYSIS

In this section, we analyze two data sets.

Data set 1 The first data set we consider is the uranium data set (Cook & Johnson (1986)). It consists of 655 data points for 7 variables, which are denoted as X_1 (Uranium), X_2 (Lithium), X_3 (Cobalt), X_4 (Potassium), X_5 (Cesium), X_6 (Scandum) and X_7 (Titanium). The data is transformed into u-scale using their empirical distributions. The function `selectFast` is applied to the data transformed to z-scale and we get that the output graph is not chordal. After two extra edges 16 and 27 are added into the graph, the final strong clique tree is obtained, which is shown in Figure 6.13.

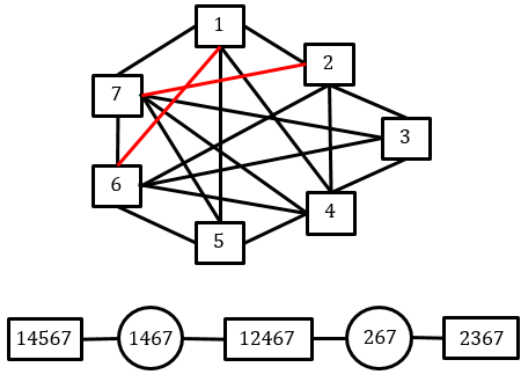

Figure 6.13: The strongly chordal graph (by adding the extra edges marked in red) and the strong clique tree for uranium data set.

We can observe that rather than estimating a dim-7 regular vine one can analyze this data by estimating two dim-5 vines and one dim-4 vine with overlapping elements. Following our heuristic m-vine approach we obtain a simplified m-vine with $AIC = -1655.24$ and $BIC = -1534.16$. The Dißmann's heuristic gives a 3-truncated vine with $AIC = -1572.59$ and $BIC = -1496.35$. The m-vine approach is better (the p-value of the Vuong test is equal to 8.19E-03 but 0.3302 with Schwarz correction) and it can be improved further by taking into account non-simplified vines. This data set was already extensively studied in Acar et al. (2012), Killiches et al. (2017) and there has been shown

that the conditional copula $C_{36;7}$ does depend on the conditioning variable.

In our further analysis we concentrate on the maximal clique $\{2,3,6,7\}$ and the vine structures for other maximal cliques can be constructed similarly. The best vine structure for the maximal clique $\{2,3,6,7\}$ can be found similarly as in Section 6.4.1, which is to search for all possible structures, estimate simplified vine at first and non-simplified vine instead if it's rejected according to the `pacotest`. Node 67 has to be included in the first tree as the restriction in the graph. By extending $V(\{6,7\})$ with variable 2 and 3 there are 12 possible vine structures. For each of them, we apply the mentioned estimation procedure and find the best one according to AIC ($AIC = -939.50$) is a non-simplified vine. This vine has the same vine structure as the simplified subvine on $\{2,3,6,7\}$ in the heuristic m-vine approach (with $AIC = -891.88$ and $BIC = -847.03$). It contains nodes 26, 67 and 36 in the second tree, nodes 27|6 and 37|6 in the third tree and node 23|67 in the last tree. We see that non-simplified vines are better models for the data. The Kendall's tau for the conditional copula in the third tree for both simplified and non-simplified copula are shown in Figure 6.14. The best non-simplified vine according to BIC ($BIC = -863.40$) is the one with nodes 26, 67, 23 in the second tree, 27|6, 36|2 in the third tree and 37|26 in the fourth tree.

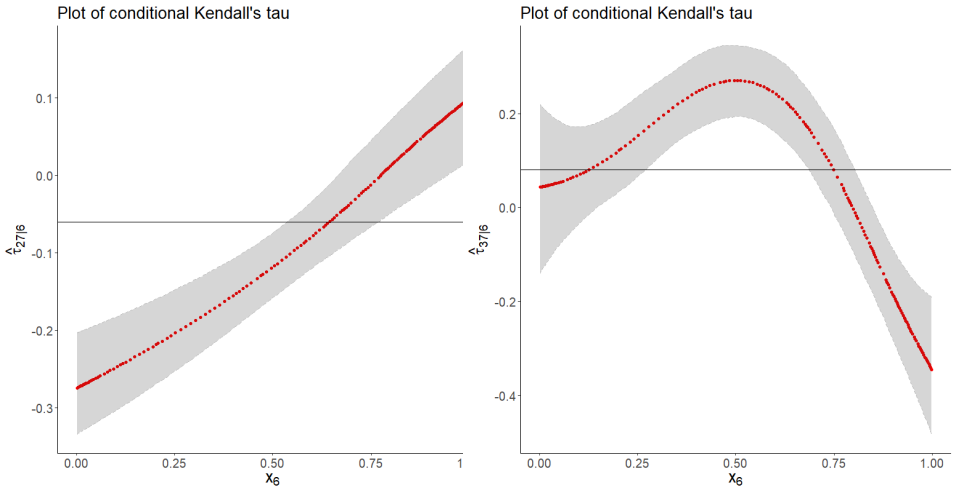

Figure 6.14: Kendall's tau for the conditional copula $C_{27;6}$ (left) and $C_{37;6}$ (right), marked in red points with its 95% confidence interval. The horizontal line denotes the conditional Kendall's tau of the simplified copula.

Data set 2 The second data set is the financial data in $KennethR.French-Datalibrary$ [2] which consists of 1142 monthly returns of 49 industry portfolios from 07-1926 to 08-2021. Detailed explanation for the industries can be found in Appendix 6.A.6. We follow the steps taken in Kurz (2019) and first filter the data by a ARMA(1,1)-GARCH(1,1) model with Student's t innovation. The residuals are fitted by their empirical distributions, and the fitted margins are then transformed to u-scale. The graph for the data in z-scale given by the function `selectFast` consists of 220 edges and is not chordal. To make this graph

[2] *http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html*

chordal extra 358 edges are added, and furthermore in order to make it strongly chordal 101 more edges are added. The final strongly chordal graph is shown in Figure 6.15 by the function `tkplot` (Csardi & Nepusz (2006)) and the added extra edges are marked in red. The obtained twenty maximal cliques in the corresponding clique tree are shown in Appendix 6.A.6.

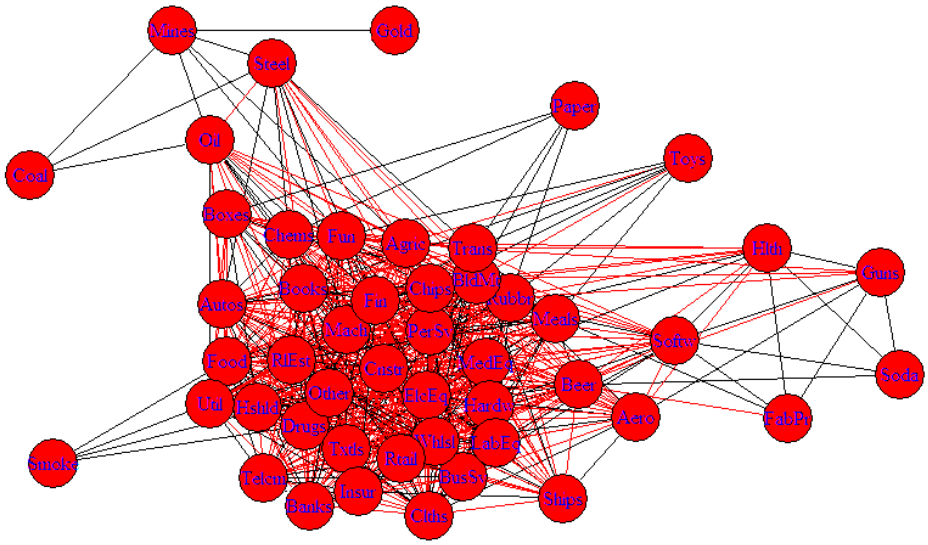

Figure 6.15: The final constructed strongly chordal graph. The extra edges added to make the graph strongly chordal is marked as red.

Although there were many edges added to make the graph strongly choral, for some indices their relationship to other indices stays the same, for example the index **Mines** for which no extra edges are added to have it connected with other vertices. Furthermore not all the added edges are meaningless to represent the relationship between indices. For example, the index **Oil** was connected with indices **Chems**, **Cnstr**, **Mach**, **Mines**, **Coal**, **Util** and **Fin**, and thirteen more edges between **Oil** and other indices are added like with **Fun** and **Steel**. In our m-vine approach, the relationship between **Oil** and **Fun**, and between **Oil** and **Steel** are estimated by a G180 copula ($\tau = 0.33$, lower tail dependence $\lambda_L = 0.41$) and a BB1 copula ($\tau = 0.44$, lower tail dependence $\lambda_L = 0.48$ and upper tail dependence $\lambda_U = 0.30$) respectively. Discussion about how the **Oil** industry affects the **Fun** industry can be found in Dahlquist & Vonderau (2022). The relationship between **Oil** industry and **Steel** industry seems more intuitive. There is a stronger lower tail dependence than the upper tail dependence for these variables.

The resulting m-vine has $AIC = -52749.33$ and $BIC = -49362.09$. Compared with the estimated 28-truncated vine from Dißmann's heuristic ($AIC = -52217.30$ and $BIC = -48930.87$), the m-vine approach is significantly better (the p-value in Vuong test equals to 2.10E-4, and 5.20E-3 with Schwarz correction). In the constructed strong clique tree, there are some maximal cliques with five or six variables for which we can further im-

prove the estimation by assessing all vine structures corresponding to those maximal cliques.

6.6. CONCLUSION

In this chapter we studied m-vines introduced in Kurowicka & Cooke (2006) and proved their relationship to the strongly chordal graphs. We proposed to use this result to simplify the regular vine copula model by assuming that a pattern of conditional independence represented by strongly chordal graph is appropriate for the data. This makes the estimation of the model much less expensive. When modeling a sparse data it allows to consider all possible vine structures corresponding to nodes in strong clique tree, and furthermore it makes feasible to consider non-simplified vines.

We showed that when the assumptions of the pattern of conditional independence is correct our method provides obvious advantages. However, it is not surprising that when the model is simplified incorrectly, our method might miss important dependencies and its performance might be poor.

We summarize several deficiencies of the m-vine approach as follows:

- The estimation accuracy of the m-vine approach relies heavily on the initial graph generated from the data as shown in the simulation study in Section 6.4.3. The m-vine approach will not consider more complex vine structures than the information about conditional (in)dependencies given by the graph. In this chapter, we use the Gaussian graphical model to generate the graph. When non-Gaussian dependence appears, methods in Bauer et al. (2012), Bauer & Czado (2016), Hobæk Haff et al. (2016) (where the conditional independence is captured by pair copula constructions including vines) can be used. This issue is however still an open problem.

- The m-vine approach benefits from the sparsity of given data to reduce computations. However, in some cases many edges may be added into the estimated graph in order to make it strongly chordal, for example when the estimated graph is sparse but contains a large cycle. This problem so far cannot be mitigated.

- Even though we can assess all possible structures for the subvines, the true m-vine structure may not be found. This is due to the restrictions in the strongly chordal graph or the heuristics we proposed. However, we can assess all possible subvine structures and choose the best if maximal cliques with small number of variables are obtained.

- To the best of the authors' knowledge, it is not known in general how specified conditional independence can be represented by a regular vine with some nodes removed. In this chapter we only discussed the conditional independence represented by strongly chordal graphs.

6.A. APPENDIX TO CHAPTER 6

6.A.1. EXAMPLE OF MERGER WITHOUT OVERLAP

Below we show an example of how to merge two regular vines without overlap. One vine is $V_1(5)$ ($V(\{1,2,3,4,5\})$) and the other vine is $V(3)$ ($V(\{6,7,8\})$), both of them shown in black in Figure 6.16. The sampling order we have chosen for $V(5)$ is $(2,3,5,4,1)$ and for $V(3)$ it is $(6,8,7)$.

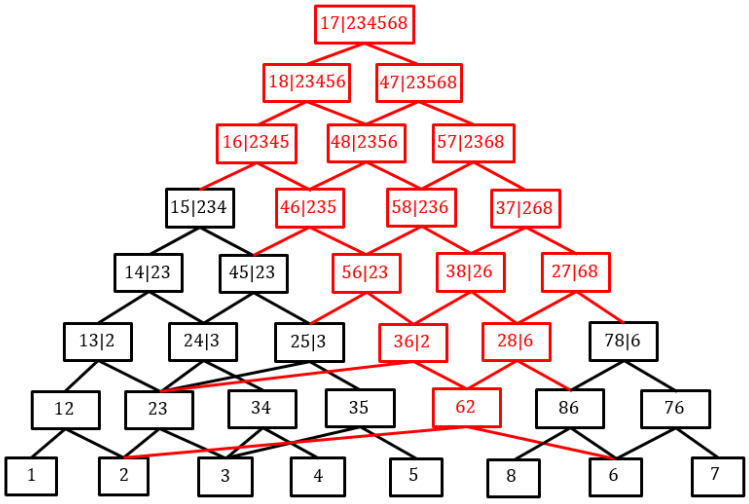

Figure 6.16: A merger of vine $V_1(5)$ ($V(\{1,2,3,4,5\})$) (black) and $V(3)$ ($V(\{6,7,8\})$ (black). The new nodes and lines in the vine triangular array are marked in red.

We see that the partners of $a_5 = 1$ are 7 in T_8, 8 in T_7 and 6 in T_6. Then marginalizing by 1 we have the partners of $a_4 = 4$ are 7 in T_7, 8 in T_6 and 6 in T_5. Through marginalizing by $a_3 = 5$, $a_2 = 3$ and $a_1 = 2$, we obtain their partners and the structure of joint vine with new nodes printed in red is shown in Figure 6.16.

6.A.2. THEOREM OF MERGER WITH OVERLAP

The theorem in this section concerns constraints on sampling orders that will lead to consistent mergers of two vines with overlap using the method in Algorithm 12.

Theorem 6.A.1. *Given two regular vines $V(n)$ and $V(m)$ with $\mathcal{V}_n \cap \mathcal{V}_m = \mathcal{V}_p \neq \emptyset$, the merger $V(\mathcal{V}_n \cup \mathcal{V}_m)$ constructed by two sampling orders $SO_n = (a_1,\ldots,a_n)$ and $SO_m = (b_1,\ldots,b_m)$ according to Algorithm 12 is valid (leads to a regular vine on elements $\mathcal{V}_n \cup \mathcal{V}_m$) if and only if $(a_1,\ldots,a_p)$ and $(b_1,\ldots,b_p)$ are sampling orders of $V(p)$.*

Proof. The structure of the merger is fixed up to and including tree T_p hence only the segment $(a_{p+1},\ldots,a_n)$ in SO_n and $(b_{p+1},\ldots,b_m)$ in SO_m affect the construction of the merger. According to Algorithm 12, in the merger, the new partners of $a_i, p+1 \leq i \leq n$ are $(b_m,\ldots,b_{p+1})$ from tree T_{i+m-p} to T_{i+1}, and the new partners of $b_j, p+1 \leq j \leq m$ are $(a_n,\ldots,a_{p+1})$ from tree T_{j+n-p} to T_{j+1}. Denote the top node of $V(p)$ as t_p.

Sufficiency: The proof of sufficiency is by construction according to Algorithm 12. Because $(a_1,\ldots,a_p)$ and $(b_1,\ldots,b_p)$ are sampling orders of the common subvine $V(p)$, the segment $(a_{p+1},\ldots,a_n)$ and $(b_{p+1},\ldots,b_m)$ do not contain elements in $\mathcal{V}_p$. Hence elements $(a_{p+1},\ldots,a_n)$ are not in $\mathcal{V}_m$ and $(b_{p+1},\ldots,b_m)$ are not in $\mathcal{V}_n$. This means that construction of the merger up to and including tree T_{p+2} is valid as it follows a vine structure construction for merger of two vines without overlap. A new node with conditioned set $a_{p+1}b_{p+1}$ is chosen in T_{p+2} of the merger, which has one child in $V(n)$ (whose conditioned set contains a_{p+1}) and the other child in $V(m)$ (whose conditioned set contains b_{p+1}). Since $(a_1,\ldots,a_p)$ and $(b_1,\ldots,b_p)$ are sampling orders of $V(p)$, these two children are both a parent node of t_p. Hence the tree construction in T_{p+1} is also valid, and a consistent merger is obtained.

Necessity: Let us assume that only $(a_1,\ldots,a_p)$ is not a sampling order of $V(p)$. Hence at least one element in the conditioned set $\{s,t\}\in\mathcal{V}_p$ of t_p, is in $(a_{p+1},\ldots,a_n)$. Let $a_k=s$ (the case that $a_k=t$ is similar), $p+1\le k\le n$, and the new partners of a_k in the merger are $(b_m,\ldots,b_{p+1})$ from tree T_{k+m-p} to tree T_{k+1}. We also have that a_k is either in the conditioning or conditioned set of parents of t_p in tree T_{p+1} of $V(m)$. If a_k is in the conditioning set, then by proximity condition, it cannot be in the conditioned sets of nodes in higher trees. If a_k is in the conditioned set, there must be a parent node where a_k is partnered with b_{p+1}, hence node with conditioned set a_kb_{p+1} already exists in $V(m)$. This leads to contradiction as each pair of elements exists in the conditioned set of a node only once in a regular vine. □

6

6.A.3. EXAMPLE VINE TRIANGULAR ARRAY

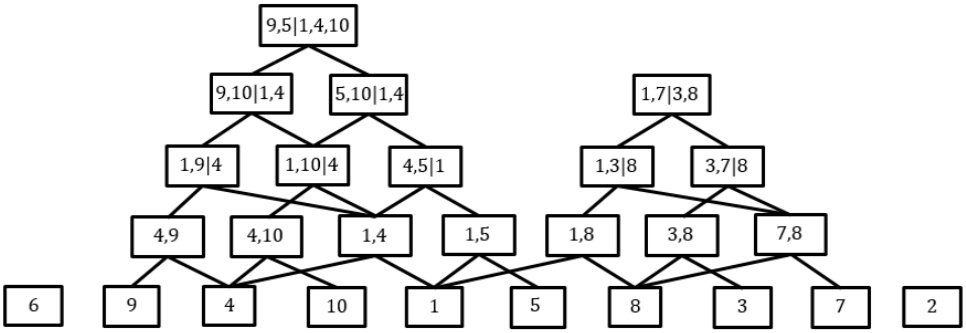

Figure 6.17: The vine triangular array representation for the m-vine in Section 6.4.1.

6.A.4. NON SIMPLIFIED COPULA EXAMPLE

To show the dependence between X_1 and X_{10} conditional on X_4, we simulated data from the estimated m-vine. The relationship between the conditional Kendall's tau of $C_{1,10|4}$ (copula family is C180 or C270) and variable X_4, estimated by GAM model is shown in the figure below. The estimated non-simplified m-vine where this relationship is not modeled directly cannot recover the relationship quite well.

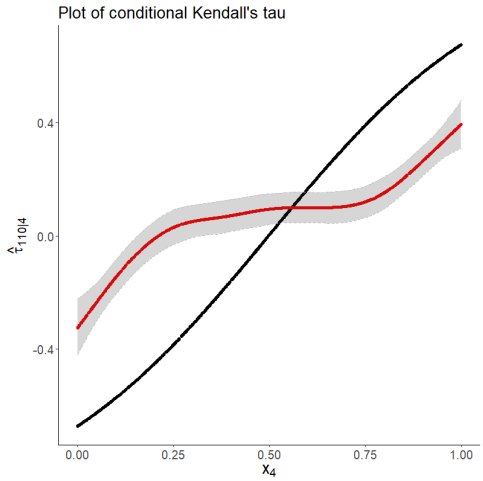

Figure 6.18: Kendall's tau for the conditional copula $C_{1,10|4}$ with respect to X_4. The true conditional tau is marked in black points and the estimated tau is marked in red points with its 95% confidence interval.

6.A.5. ALGORITHM

Algorithm 15 Determine the order of separators to be merged for node $\mathscr{C}_j^{(i)}$

Input: the separators $\mathscr{S}$ and the clique tree $\mathscr{T}^{(i+1)}$
Output: the order of separators $order$

1: $index = 0$.
2: **while** not all separators in $\mathscr{S}$ are considered **do**
3: Randomly pick one separator from the separators that haven't been considered, denoted as $\mathscr{S}'_{index}$.
4: Set $index = index + 1$ and $order[index]$ = the index of $\mathscr{S}'_{index}$ in $\mathscr{S}$.
5: Find the neighbors of $\mathscr{S}'_{index}$ in $\mathscr{T}^{(i+1)}$, denoted as $neigh_{index}$.
6: Apply the below recursive procedure.
7: **procedure** RECURSE($neigh_{index}$)
8: **if** there are at least one neighbor in $neigh_{index}$ **then**
9: **for** each neighbor in $neigh_{index}$ **do**
10: **if** this neighbor is in $\mathscr{S}$ **then**
11: **if** this neighbor has overlap with $\mathscr{S}'_{index}$ **then**
12: Denote this neighbor as $\mathscr{S}''_{index}$.
13: Set $index = index + 1$, $order[index]$ = the index of $\mathscr{S}''_{index}$ in $\mathscr{S}$.
14: **end if**
15: **end if**
16: Find the neighbors of this neighbor in $\mathscr{T}^{(i+1)}$ and choose ones which haven't been considered before, denoted as $neigh_{index}$.
17: RECURSE($neigh_{index}$)
18: **end for**
19: **end if**
20: **end procedure**
21: **end while**

In line 13 in Algorithm 13 when all the vines corresponding to the nodes $\mathscr{T}^{(0)}$ are combined, the order of combining these vines should follow Algorithm 15 as well. This is achieved by setting $\mathscr{S}$ to be the maximal cliques $\mathscr{C}$ of the strongly chordal graph and the underlying tree structure is $\mathscr{T}^{(0)}$.

6.A.6. REAL DATA VARIABLES SPECIFICATION

The industries in the 49 industry portfolios are, Agriculture (**Agric**, X_1), Food Products (**Food**,X_2), Candy & Soda (**Soda**, X_3), Beer &Liquor (**Beer**, X_4), Tobacco Products (**Smoke**, X_5), Recreation (**Toys**, X_6), Entertainment (**Fun**, X_7), Printing &Publishing (**Books**, X_8), Consumer Goods (**Hshld**, X_9), Apparel (**Clths**, X_{10}), Healthcare (**Hlth**, X_{11}), Medical Equipment (**MedEq**, X_{12}), Pharmaceutical Products (**Drugs**, X_{13}), Chemicals (**Chems**, X_{14}), Rubber and Plastic Products (**Rubbr**, X_{15}), Textiles (**Txtls**, X_{16}), Construction Materials (**BldMt**, X_{17}), Construction (**Cnstr**, X_{18}), Steel Works Etc (**Steel**, X_{19}), Fabricated Products &Machinery (**FabPr**, X_{20}), Machinery (**Mach**, X_{21}), Electrical Equipment (**ElcEq**, X_{22}), Automobiles &Trucks (**Autos**, X_{23}), Aircraft (**Aero**, X_{24}), Ships &Railroad Equipment

(**Ships**, X_{25}), Defense (**Guns**, X_{26}), Precious Metals (**Gold**, X_{27}), Non-Metallic &Industrial Metal Mining (**Mines**, X_{28}), Coal (**Coal**, X_{29}), Petroleum &Natural Gas (**Oil**, X_{30}), Utilities (**Util**, X_{31}), Communication (**Telcm**, X_{32}), Personal Services (**PerSv**, X_{33}), Business Services (**BusSv**, X_{34}), Computers (**Hardw**, X_{35}), Computer Software (**Softw**, X_{36}), Electrical Equipment (**Chips**, X_{37}), Measuring & Control Equipment (**LabEq**, X_{38}), Business Supplies (**Paper**, X_{39}), Shipping Containers (**Boxes**, X_{40}), Transportation (**Trans**, X_{41}), Wholesale (**Whlsl**, X_{42}), Retail (**Rtail**, X_{43}), Restaurants, Hotels &Motels (**Meals**, X_{44}), Banking (**Banks**, X_{45}), Insurance (**Insur**, X_{46}), Real Estate (**RlEst**, X_{47}), Trading (**Fin**, X_{48}) and Almost nothing (**Other**, X_{49}).

The maximal cliques in the strong clique tree for the financial data set in Section 6.5.
{27,28}
{19,28,29,30}
{3,4,11,26,36}
{1,7,19,28,30}
{2,5,13,31,49}
{14,15,17,39,40,41}
{4,11,15,20,26,36}
{4,11,15,17,24,26,36,41}
{1,6,7,8,15,17,37,41,44}
{1,4,11,12,15,17,24,33,36,37,41}
{1,7,8,14,15,17,19,21,23,30,33,37,41,48}
{1,2,7,8,13,14,15,17,18,21,23,30,31,33,37,41,48,49}
{1,4,12,13,15,17,18,21,22,24,33,34,35,36,37,38,41,43,48,49}
{1,4,10,12,13,15,17,18,21,22,24,25,33,34,35,37,38,41,42,43,48,49}
{1,2,7,8,9,12,13,14,15,17,18,21,22,23,31,33,37,40,41,43,48,49}
{1,4,10,12,13,15,17,18,21,22,25,33,34,35,37,38,41,42,43,46,47,48,49}
{1,2,7,8,9,12,13,14,15,17,18,21,22,23,31,33,34,35,37,38,41,42,43,46,47,48,49}
{1,2,7,8,9,12,13,15,16,17,18,21,22,23,31,33,34,35,37,38,41,42,43,45,46,47,48,49}
{1,2,4,7,8,9,10,12,13,15,16,17,18,21,22,33,34,35,37,38,41,42,43,44,46,47,48,49}
{1,2,4,7,8,9,10,12,13,15,16,17,18,21,22,31,32,33,34,35,37,38,41,42,43,45,46,47,48,49}

7

CONCLUSION

The main topic of this thesis is to discuss the problem of selecting vine structures for the vine copula model. This problem is important as the choice of a structure is shown to have strong influence on the performance of the vine copula model for a given data. This topic is quite challenging as the number of possible vine structures grows exponentially with the number of variables. In this chapter, the main conclusions of the works in the thesis will be presented and a few future research directions will be discussed.

- In order to construct a vine structure, a new vine representation called vine binomial tree (VBT) is proposed in this thesis. VBT has been shown to be a useful tool in constructing vine structures. We have applied this representation in a few problems discussed in the thesis. However, VBT is not as compact as other representations such as the vine matrix representation. Due to computational inefficiency of the VBT representation, some of the developed algorithms could not be implemented in very high dimensions. Hence, it will be of interest to find a relationship between VBT and vine matrix, by which efficient implementation of the algorithms can be achieved. Such an extension might also lead to a new package in R for the developed algorithms.
- The idea of trying to find a vine structure with a better performance for a given data than an initial structure is very intuitive. One can examine a fixed number of randomly chosen structures, but that can be inefficient as the space of possible vine structures in higher dimensions is large. In this thesis, it is proposed to search for vines having 2 common sampling orders with an initial vine. The approach was motivated by the observation that vines having large number of sampling orders in common with the initial vine have many repeated copulas in the density decomposition. It has been shown both in the simulation study and the real data analysis that vine copula models with better performance can be found with this method. The number of common sampling orders is not the only measure to explore the space of vine structures. Different types of measures of similarity between two vine structures could be investigated in the future, which might allow an improved performance of the search procedure presented in this thesis.

- A choice of structure in R-vine based regression has been investigated in this thesis. We have developed algorithms that allow the construction of all possible vine structures that lead to a conditional distribution of the response variable given covariates in an analytic form. However, when the construction is combined with the forward selection approach, the importance of the structure selection to the performance of the regression model is not as strong as expected. This is because the variables contributing the most to the performance are fixed at first, when not many vine structures can be constructed. One of the possible research directions could be to combine the proposed vine based regression with other types of variable selection methods.

- Vine copula model is flexible but certainly not parsimonious. It can be simplified by assuming a pattern of conditional independence presented in the given data. In this thesis, it is shown that for a strongly chordal pattern of conditional independence, an m-vine can be constructed by assigning independence copula to the corresponding edges of the vine based on the given strongly chordal graph. There are still many m-vine structures corresponding to the given strongly chordal graph, and it is computationally expensive to examine each of them. However, the assumed strongly chordal pattern of conditional independence allows one to work with models composed of smaller subvines, where it becomes possible to estimate all possible subvine structures and/or even consider non-simplified vines. It has been shown in a simulation study that when the assumed conditional independence is not correct, the proposed m-vine model might not perform well. Hence, improving the efficiency of our method under different graph selection methods, especially for non-Gaussian method, would be of interest. Moreover, more research is needed to investigate how in general a specified pattern of conditional independence can be represented by incomplete vines.

- In this thesis only simplified vines are considered. We have shown in the example in the beginning of this thesis that, when the vine structure is different than the structure of the simulated model, the simplifying assumption may be violated. It is, in general, very difficult to judge if the given data can be modeled by a simplified vine, because in order to answer this question one needs to consider all structures. This problem is very difficult but we hope that some progress can be made in the future to extend our knowledge of this issue.

ACKNOWLEDGEMENTS

At first I would like to thank to the financial support from the China Scholarship Council. Then I want to thank my supervisors Dorota Kurowicka and Gabriela Florentina Nane. Thank you very much for your novel ideas and suggestions directing my research projects, your patient guidance on writing papers, your generous attitude towards my faults, e.g. my disappearance. Without your supports, I would not have been able to succeed in the world of vine copula and finished this thesis. I would also like to thank the support from my colleagues in the Applied Probability and Statistics group.

CURRICULUM VITÆ

Kailun ZHU

Kailun Zhu was born in Huzhou, Zhejiang Province, China, on 01-Sep-1992. He received his bachelor's degree in Zhejiang University in 2015 major in Mathematics & Applied Mathematics. Then he went to Singapore to study Financial Engineering and received his master's degree in National University of Singapore in July, 2017. In October, 2017, he moved to the Netherlands and start his PhD study working with vine copula model, in the group of Applied Probability in Delft University of Technology.

BIBLIOGRAPHY

Aas, K. (2016), 'Pair-copula constructions for financial applications: A review', *Econometrics* **4**(4).
URL: *https://www.mdpi.com/2225-1146/4/4/43*

Aas, K., Czado, C., Frigessi, A. & Bakken, H. (2009), 'Pair-copula constructions of multiple dependence', *Insurance: Mathematics and Economics* **44**(2), 182 – 198.

Acar, E. F., Craiu, R. V. & Yao, F. (2011), 'Dependence calibration in conditional copulas: A nonparametric approach', *Biometrics* **67**(2), 445–453.

Acar, E. F., Czado, C. & Lysy, M. (2019), 'Flexible dynamic vine copula models for multivariate time series data', *Econometrics and Statistics* **12**, 181–197.
URL: *https://www.sciencedirect.com/science/article/pii/S2452306219300206*

Acar, E. F., Genest, C. & Nešlehová, J. (2012), 'Beyond simplified pair-copula constructions', *Journal of Multivariate Analysis* **110**, 74 – 90.

Akaike, H. (1998), Information theory and an extension of the maximum likelihood principle, *in* P. E., T. K. & K. G., eds, 'Selected Papers of Hirotugu Akaike', Springer, pp. 199–213.

Bauer, A. & Czado, C. (2016), 'Pair-copula bayesian networks', *Journal of Computational and Graphical Statistics* **25**(4), 1248–1271.

Bauer, A., Czado, C. & Klein, T. (2012), 'Pair-copula constructions for non-gaussian dag models', *Canadian Journal of Statistics* **40**(1), 86–109.

Bedford, T. & Cooke, R. (2001), 'Probability density decomposition for conditionally dependent random variables modeled by vines', *Ann. Math. Artif. Intell.* **32**, 245– 268.

Bedford, T. & Cooke, R. (2002), 'Vines - a new graphical model for dependent random variables', *Annals of Statistics* **30**(4), 1031–1068.

Behun, M., Gavurova, B., Tkacova, A. & Kotaskova, A. (2018), 'The impact of the manufacturing industry on the economic cycle of european union countries', *Journal of Competitiveness* **10**(1), 23–39.

Brechmann, E. C., Czado, C. & Aas, K. (2012), 'Truncated regular vines in high dimensions with application to financial data', *The Canadian Journal of Statistics / La Revue Canadienne de Statistique* **40**(1), 68–85.
URL: *http://www.jstor.org/stable/41724516*

Brechmann, E. C., Hendrich, K. & Czado, C. (2013), 'Conditional copula simulation for systemic risk stress testing', *Insurance: Mathematics and Economics* **53**(3), 722 – 732.

Brechmann, E. C. & Joe, H. (2014), 'Parsimonious parameterization of correlation matrices using truncated vines and factor analysis', *Computational Statistics & Data Analysis* **77**, 233 – 251.

Brechmann, E. & Schepsmeier, U. (2013), 'Modeling dependence with c- and d-vine copulas: The r package cdvine', *Journal of Statistical Software, Articles* **52**(3), 1–27.

Breymann, W., Dias, A. & Embrechts, P. (2003), 'Dependence structures for multivariate high-frequency data in finance', *Quantitative Finance* **3**(1), 1–14.

Brier, G. W. (1950), 'Verification of forecasts expressed in terms of probability', *Monthly Weather Review* **78**(1), 1–3.

Buczak, A. L. & Gifford, C. M. (2010), Fuzzy association rule mining for community crime pattern discovery, *in* 'ACM SIGKDD Workshop on Intelligence and Security Informatics', ISI-KDD '10, Association for Computing Machinery, New York, NY, USA.
URL: *https://doi.org/10.1145/1938606.1938608*

Chang, B. & Joe, H. (2019), 'Prediction based on conditional distributions of vine copulas', *Computational Statistics & Data Analysis* **139**, 45 – 63.

Chang, B., Pan, S. & Joe, H. (2019), Vine copula structure learning via monte carlo tree search, *in* K. Chaudhuri & M. Sugiyama, eds, 'Proceedings of Machine Learning Research', Vol. 89 of *Proceedings of Machine Learning Research*, PMLR, pp. 353–361.
URL: *http://proceedings.mlr.press/v89/chang19a.html*

Cherubini, U., Luciano, E. & Vecchiato, W. (2004), *Copula Methods in Finance*, The Wiley Finance Series, Wiley.
URL: *https://books.google.nl/books?id=0dyagVg20XQC*

Compagne, K. C. & et al. (n.d.), 'Improvements in endovascular treatment for acute ischemic stroke: A longitudinal study in the mr clean registry', *Stroke* **0**(0), STROKEAHA.121.034919.
URL: *https://www.ahajournals.org/doi/abs/10.1161/STROKEAHA.121.034919*

Cook, R. D. & Johnson, M. E. (1986), 'Generalized burr-pareto-logistic distributions with applications to a uranium exploration data set', *Technometrics* **28**(2), 123 – 131.

Cooke, R. (1997), 'Markov and entropy properties of tree- and vine-dependent variables', *Proceedings of the ASA Section on Bayesian Statistical Science* pp. 166–175.

Cooke, R., Kurowicka, D. & Wilson, K. (2015), 'Sampling, conditionalizing, counting, merging, searching regular vines', *Journal of Multivariate Analysis* **138**, 4 – 18.

Cooke, R. M., Joe, H. & Chang, B. (2019), 'Vine copula regression for observational studies', *AStA Advances in Statistical Analysis* .
URL: *https://doi.org/10.1007/s10182-019-00353-5*

Csardi, G. & Nepusz, T. (2006), 'The igraph software package for complex network research', *InterJournal* **Complex Systems**, 1695.
URL: *https://igraph.org*

Czado, C. (2010), Pair-copula constructions of multivariate copulas, *in* P. Jaworski, F. Durante, W. Härdle & T. Rychlik, eds, 'Copula Theory and Its Applications: Proceedings of the Workshop Held in Warsaw, 25-26 September 2009', Springer Berlin Heidelberg, pp. 93– 109.

Czado, C. (2019), *Analyzing dependent data with vine copulas*, Lecture Notes in Statistics, Springer.

Czado, C., Brechmann, E. C. & Gruber, L. (2013), Selection of vine copulas, *in* P. Jaworski, F. Durante & W. Härdle, eds, 'Copulae in Mathematical and Quantitative Finance', Springer Berlin Heidelberg, pp. 17– 37.

Czado, C. & Nagler, T. (2022), 'Vine copula based modeling', *Annual Review of Statistics and Its Application* **9**(1), null.
URL: *https://doi.org/10.1146/annurev-statistics-040220-101153*

Dahlquist, M. & Vonderau, P. (2022), *Petrocinema: Sponsored Film and the Oil Industry*, Bloomsbury Academic.
URL: *https://books.google.nl/books?id=bUBBEAAAQBAJ*

Dißmann, J., Brechmann, E., Czado, C. & Kurowicka, D. (2013), 'Selecting and estimating regular vine copulae and application to financial returns', *Computational Statistics & Data Analysis* **59**, 52 – 69.

Drechsler, J. (2011), *Synthetic datasets for statistical disclosure control, theory and implementation*, Lecture Notes in Statistics, Springer.

Duong, T. (2019), *ks: Kernel Smoothing*. R package version 1.11.6.
URL: *https://CRAN.R-project.org/package=ks*

Farber, M. (1983), 'Characterizations of strongly chordal graphs', *Discrete Mathematics* **43**(2), 173–189.

Fasano, G. & Franceschini, A. (1987), 'A multidimensional version of the kolmogorov-smirnov test', *Mon Not R astr* **225**, 155–170.

Friedman, J., Hastie, T. & Tibshirani, R. (2007), 'Sparse inverse covariance estimation with the graphical lasso', *Biostatistics* **9**(3), 432–441.

Gavril, F. (1974), 'The intersection graphs of subtrees in trees are exactly the chordal graphs', *Journal of Combinatorial Theory, Series B* **16**(1), 47–56.
URL: *https://www.sciencedirect.com/science/article/pii/009589567490094X*

GBD 2019 Stroke Collaborators (2021), 'Global, regional, and national burden of stroke and its risk factors, 1990-2019: a systematic analysis for the global burden of disease study 2019', *Lancet Neurol* **20**(10), 795–820.

Genest, C. & Favre, A.-C. (2007), 'Everything you always wanted to know about copula modeling but were afraid to ask', *Journal of Hydrologic Engineering* **12**(4), 347–368.

Gijbels, I., Veraverbeke, N. & Omelka, M. (2011), 'Conditional copulas, association measures and their applications', *Computational Statistics & Data Analysis* **55**(5), 1919 – 1932.

Giraud, Christophe, Huet, Sylvie, Verzelem & Nicolas (2009), 'Graph selection with ggmselect', arXiv:0907.0619.
URL: *http://fr.arxiv.org/abs/0907.0619*

Goncalves, A. & et al. (2020), 'Generation and evaluation of synthetic patient data', *BMC medical research methodology* **108**(1), null.

Goodman, B. & Flaxman, S. (2017), 'European union regulations on algorithmic decision-making and a "right to explanation"', *AI Magazine* **38**(3), 50–57.
URL: *https://ojs.aaai.org/index.php/aimagazine/article/view/2741*

Gravesteijn, B. & et al. (2021), 'Missing data in prediction research: A five-step approach for multiple imputation, illustrated in the center-tbi study', *Journal of neurotrauma* **38**(13), 1842–1857.

Gruber, L. & Czado, C. (2015), 'Sequential Bayesian Model Selection of Regular Vine Copulas', *Bayesian Analysis* **10**(4), 937 – 963.
URL: *https://doi.org/10.1214/14-BA930*

Gruber, L. F. & Czado, C. (2018), 'Bayesian Model Selection of Regular Vine Copulas', *Bayesian Analysis* **13**(4), 1111 – 1135.
URL: *https://doi.org/10.1214/17-BA1089*

Haff, I. H., Aas, K. & Frigessi, A. (2010), 'On the simplified pair-copula construction — simply useful or too simplistic?', *Journal of Multivariate Analysis* **101**(5), 1296 – 1310.

Hasler, C., Craiu, R. V. & Rivest, L.-P. (2018), 'Vine copulas for imputation of monotone non-response', *International Statistical Review* **86**(3), 488–511.
URL: *https://onlinelibrary.wiley.com/doi/abs/10.1111/insr.12263*

Herrmann, J. S. (2018), Regular vine copula based quantile regression, Master's thesis, Technische Universität München, Germany.

Hobæk Haff, I., Aas, K., Frigessi, A. & Lacal, V. (2016), 'Structure learning in bayesian networks using regular vines', *Computational Statistics & Data Analysis* **101**, 186–208.
URL: *https://www.sciencedirect.com/science/article/pii/S0167947316300457*

Hofert, M., Kojadinovic, I., Maechler, M. & Yan, J. (2018), *Elements of Copula Modeling with R*, Springer.

Hofert, M., Kojadinovic, I., Maechler, M. & Yan, J. (2020), *copula: Multivariate Dependence with Copulas*. R package version 1.0-1.
URL: *https://CRAN.R-project.org/package=copula*

Jansen, I. G. H., Mulder, M. J. H. L. & Goldhoorn, R.-J. B. (2018), 'Endovascular treatment for acute ischaemic stroke in routine clinical practice: prospective, observational cohort study (mr clean registry)', *BMJ* **360**.
URL: *https://www.bmj.com/content/360/bmj.k949*

Joe, H. (1997), *Multivariate Models and Multivariate Dependence Concepts*, Taylor & Francis.

Joe, H. (2014), *Dependence Modeling with Copulas*, Chapman and Hall/CRC.

Joe, H., Cooke, R. & Kurowicka, D. (2011), Regular vines: Generation algorithm and number of equivalence classes, *in* H. Joe & D. Kurowicka, eds, 'Dependence Modeling: Vine Copula Handbook', World Scientific, pp. 219– 231.

Killiches, M., Kraus, D. & Czado, C. (2017), 'Examination and visualisation of the simplifying assumption for vine copulas in three dimensions', *Australian & New Zealand Journal of Statistics* **59**(1), 95–117.
URL: *https://onlinelibrary.wiley.com/doi/abs/10.1111/anzs.12182*

Koenker, R. & Machado, J. A. F. (1999), 'Goodness of fit and related inference processes for quantile regression', *Journal of the American Statistical Association* **94**(448), 1296–1310.

Konishi, S. & Kitagawa, G. (1996), 'Generalised information criteria in model selection', *Biometrika* **83**(4), 875–890.
URL: *http://www.jstor.org/stable/2337290*

Kraus, D. & Czado, C. (2017*a*), 'D-vine copula based quantile regression', *Computational Statistics & Data Analysis* **110**, 1 – 18.

Kraus, D. & Czado, C. (2017*b*), 'Growing simplified vine copula trees: improving Dißmann's algorithm', *ArXiv e-prints* .

Kurowicka, D. (2011), Optimal truncation of vines, *in* H. Joe & D. Kurowicka, eds, 'Dependence Modeling: Vine Copula Handbook', World Scientific, pp. 243– 257.

Kurowicka, D. & Cooke, R. (2006), 'Completion problem with partial correlation vines', *Linear Algebra and its Applications* **418**(1), 188–200.
URL: *https://www.sciencedirect.com/science/article/pii/S0024379506000619*

Kurowicka, D. & Joe, H. (2011), *Dependence Modeling: Vine Copula Handbook*, World Scientific.

Kurz, M. S. (2019), *pacotest: Testing for Partial Copulas and the Simplifying Assumption in Vine Copulas*. R package version 0.3.1.

Kurz, M. S. & Spanhel, F. (2017), 'Testing the simplifying assumption in high-dimensional vine copulas', *ArXiv e-prints* .

Lauritzen, S. (1996), *Graphical Models*, Oxford Statistical Science Series, Clarendon Press.

Li, F., Yu, Y. & Rubin, D. B. (2012), 'Imputing missing data by fully conditional models: Some cautionary examples and guidelines'.

Li, H., Huang, G., Li, Y., Sun, J. & Gao, P. (2021), 'A c-vine copula-based quantile regression method for streamflow forecasting in xiangxi river basin, china', *Sustainability* **13**(9).
URL: *https://www.mdpi.com/2071-1050/13/9/4627*

McCullagh, P. & Nelder, J. (1989), *Generalized Linear Models, Second Edition*, Chapman & Hall/CRC Monographs on Statistics & Applied Probability, Taylor & Francis.

McKee, T. A. (2003), 'Subgraph trees in graph theory', *Discrete Mathematics* **270**(1), 3 – 12.

McNeil, A., Embrechts, P. & Frey, R. (2010), *Quantitative Risk Management: Concepts, Techniques and Tools*, Princeton series in finance, New Age International.
URL: *https://books.google.nl/books?id=lc7PlwEACAAJ*

Meinshausen, N. & Bühlmann, P. (2006), 'High-dimensional graphs and variable selection with the Lasso', *The Annals of Statistics* **34**(3), 1436 – 1462.

Müller, D. & Czado, C. (2018), 'Representing sparse gaussian dags as sparse r-vines allowing for non-gaussian dependence', *Journal of Computational and Graphical Statistics* **27**(2), 334–344.

Müller, D. & Czado, C. (2019*a*), 'Dependence modelling in ultra high dimensions with vine copulas and the graphical lasso', *Computational Statistics & Data Analysis* **137**, 211–232.
URL: *https://www.sciencedirect.com/science/article/pii/S0167947319300568*

Müller, D. & Czado, C. (2019*b*), 'Selection of sparse vine copulas in high dimensions with the lasso', *Statistics and Computing* **29**(2), 269 – 287.

Mukhopadhyay, A. & Rahman, M. Z. (2021), Algorithms for generating strongly chordal graphs, *in* M. L. Gavrilova & C. K. Tan, eds, 'Transactions on Computational Science XXXVIII', Springer Berlin Heidelberg.

Mulder, M. J. & et al. (2018), 'Time to endovascular treatment and outcome in acute ischemic stroke', *Circulation* **138**(3), 232–240.
URL: *https://www.ahajournals.org/doi/abs/10.1161/CIRCULATIONAHA.117.032600*

Nagler, T. (2018), 'A generic approach to nonparametric function estimation with mixed data'.

Nagler, T., Bumann, C. & Czado, C. (2019), 'Model selection in sparse high-dimensional vine copula models with an application to portfolio risk', *Journal of Multivariate Analysis* **172**, 180–192. Dependence Models.
URL: *https://www.sciencedirect.com/science/article/pii/S0047259X18300630*

Nagler, T. & Czado, C. (2016), 'Evading the curse of dimensionality in nonparametric density estimation with simplified vine copulas', *Journal of Multivariate Analysis* **151**, 69–89.
URL: *https://www.sciencedirect.com/science/article/pii/S0047259X16300471*

Nagler, T., Schepsmeier, U., Stoeber, J., Brechmann, E. C., Graeler, B. & Erhardt, T. (2019), *VineCopula: Statistical Inference of Vine Copulas*. R package version 2.2.0.
URL: *https://CRAN.R-project.org/package=VineCopula*

Nagler, T. & Vatter, T. (2020), *gamCopula: Generalized Additive Models for Bivariate Conditional Dependence Structures and Vine Copulas*. R package version 0.0-7.
URL: *https://CRAN.R-project.org/package=gamCopula*

Nagler, T. & Vatter, T. (2021), *rvinecopulib: High Performance Algorithms for Vine Copula Modeling*. R package version 0.5.5.1.1.
URL: *https://CRAN.R-project.org/package=rvinecopulib*

Nápoles, O. (2010), Bayesian Belief Nets and Vines in Aviation Safety and Other Applications, PhD thesis, Delft University of Technology.

Nápoles, O. (2011), Counting vines, *in* H. Joe & D. Kurowicka, eds, 'Dependence Modeling: Vine Copula Handbook', World Scientific, pp. 199– 228.

Nelsen, R. (2006), *An Introduction to Copulas*, Springer.

Noh, H., Ghouch, A. E. & Bouezmarni, T. (2013), 'Copula-based regression estimation and inference', *Journal of the American Statistical Association* **108**(502), 676–688.

Panagiotelis, A., Czado, C. & Joe, H. (2012), 'Pair copula constructions for multivariate discrete data', *Journal of the American Statistical Association* **107**(499), 1063–1072.

Parsa, R. A. & Klugman, S. A. (2011), 'Copula regression', *Variance* **5**(1), 45–54.

Pearl, J. (1988), *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann series in representation and reasoning, Elsevier Science.
URL: *https://books.google.nl/books?id=AvNID7LyMusC*

Rosenblatt, M. (1952), 'Remarks on a multivariate transformation', *Ann. Math. Statist.* **23**(3), 470–472.

Royston, P. & Parmar, M. K. B. (2002), 'Flexible parametric proportional-hazards and proportional-odds models for censored survival data, with application to prognostic modelling and estimation of treatment effects', *Statistics in Medicine* **21**(15), 2175–2197.
URL: *https://onlinelibrary.wiley.com/doi/abs/10.1002/sim.1203*

Rubin, D. B. (1987), *Multiple Imputation for Nonresponse in Surveys*, Wiley.

Samuels, N. & et al. (n.d.), A synthetic stroke population: rationale, comparison of methods, and overview of applications. unpublished.

Saver, J. L. & et al. (2016), 'Time to treatment with endovascular thrombectomy and outcomes from ischemic stroke: A meta-analysis', *JAMA* **316**(12), 1279–1288.

Schepsmeier, U. & Czado, C. (2016), 'Dependence modelling with regular vine copula models: a case-study for car crash simulation data', *Journal of the Royal Statistical Society. Series C (Applied Statistics)* **65**(3), 415–429.
URL: *http://www.jstor.org/stable/24773029*

Schwarz, G. (1978), 'Estimating the Dimension of a Model', *The Annals of Statistics* **6**(2), 461 – 464.
URL: *https://doi.org/10.1214/aos/1176344136*

Sklar, M. (1959), *Fonctions de Répartition À N Dimensions Et Leurs Marges*, Université Paris 8.
URL: *https://books.google.nl/books?id=nreSmAEACAAJ*

Smania, G. & Jonsson, E. N. (2021), 'Conditional distribution modeling as an alternative method for covariates simulation: Comparison with joint multivariate normal and bootstrap techniques', *pharmacometrics & systems pharmacology* **10**(4), 330–339.

Spanhel, F. & Kurz, M. S. (2015), 'The partial vine copula: A dependence measure and approximation based on the simplifying assumption', *ArXiv e-prints* .

Stöber, J., Hong, H. G., Czado, C. & Ghosh, P. (2015), 'Comorbidity of chronic diseases in the elderly: Patterns identified by a copula design for mixed responses', *Computational Statistics & Data Analysis* **88**, 28–39.
URL: *https://www.sciencedirect.com/science/article/pii/S0167947315000304*

Stöber, J., Joe, H. & Czado, C. (2013), 'Simplified pair copula constructions—limitations and extensions', *Journal of Multivariate Analysis* **119**, 101 – 118.

Tannenbaum, S. J. & et al. (2006), 'Simulation of correlated continuous and categorical variables using a single multivariate distribution', *Journal of Pharmacokinetics and Pharmacodynamics* **33**(6), 773–794.
URL: *https://doi.org/10.1007/s10928-006-9033-1*

Teutonico, D. & et al. (2015), 'Generating virtual patients by multivariate and discrete resampling techniques', *Pharmaceutical Research* **32**(10), 3228–3237.
URL: *https://doi.org/10.1007/s11095-015-1699-x*

Tucker, A., Wang, Z., Rotalinti, Y. & Myles, P. (2020), 'Generating high-fidelity synthetic patient data for assessing machine learning healthcare software', *npj Digital Medicine* **3**(1), 147.
URL: *https://doi.org/10.1038/s41746-020-00353-9*

van Buuren, S. (2012), *Flexible Imputation of Missing Data*, Chapman & Hall/CRC Interdisciplinary Statistics, CRC Press.
URL: *https://books.google.nl/books?id=elDNBQAAQBAJ*

Venema, C. & et al. (2017), '18f-fes pet has added value in staging and therapy decision making in patients with disseminated lobular breast cancer', *Clinical nuclear medicine* **42**(8), 612–614.

Veraverbeke, N., Omelka, M. & Gijbels, I. (2011), 'Estimation of a conditional copula and association measures', *Scandinavian Journal of Statistics* **38**(4), 766–780.

Vuong, Q. H. (1989), 'Likelihood ratio tests for model selection and non-nested hypotheses', *Econometrica: Journal of the Econometric Society* pp. 307–333.

Walonoski, J. & et al. (2018), 'Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record', *Journal of the American Medical Informatics Association* **25**(3), 230–238.

Wille, A. & Bühlmann, P. (2006), 'Low-order conditional independence graphs for inferring genetic networks', *Statistical Applications in Genetics and Molecular Biology* **5**(1).

Yeh, I. (1998), 'Modeling of strength of high-performance concrete using artificial neural networks', *Cement and Concrete Research* **28**(12), 1797 – 1808.

Yule, G. & Kendall, M. (1965), *An Introduction to the Theory of Statistics*, C. Griffin.
URL: *https://books.google.nl/books?id=CM1WAAAAYAAJ*

Zhu, K. & Kurowicka, D. (2022), 'Regular vines with strongly chordal pattern of (conditional) independence', *Computational Statistics & Data Analysis* **172**, 107461.
URL: *https://www.sciencedirect.com/science/article/pii/S016794732200041X*

Zhu, K., Kurowicka, D. & Nane, G. F. (2020), 'Common sampling orders of regular vines with application to model selection', *Computational Statistics & Data Analysis* **142**, 106811.

Zhu, K., Kurowicka, D. & Nane, G. F. (2021), 'Simplified r-vine based forward regression', *Computational Statistics & Data Analysis* **155**, 107091.
URL: *https://www.sciencedirect.com/science/article/pii/S0167947320301821*

Zhu, K., Samuels, N., Kurowicka, D. & Nieboer, D. (n.d.), Vine copula based generation of synthetic population of acute ischemic stroke patients. unpublished.