



Technische Universität München
Zentrum Mathematik
Lehrstuhl für Mathematische Statistik

Selection of Sparse Vine Copulas in Ultra High Dimensions

Dominik Thomas Müller

Vollständiger Abdruck der von der Fakultät für Mathematik der Technischen Universität München zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)
genehmigten Dissertation.

Vorsitzender: Prof. Dr. Matthias Scherer
Prüfer der Dissertation: 1. Prof. Claudia Czado, Ph.D.
2. Prof. Harry Joe, Ph.D.
(University of British Columbia, Kanada)
3. Prof. Gal Elidan, Ph.D.
(Hebrew University of Jerusalem, Israel)
(nur schriftliches Gutachten)

Die Dissertation wurde am 17.10.2017 bei der Technischen Universität München eingereicht und durch die Fakultät für Mathematik am 29.11.2017 angenommen.

Zusammenfassung

Das Ziel dieser Arbeit ist die Anwendbarkeit von sogenannten Vine Copula Modellen in ultrahohe Dimensionen zu erweitern. Vine Copulas stellen eine neuartige flexible Modellklasse dar, welche Verteilungen in endlichen Dimensionen durch die Kombination von beliebigen Marginalverteilungen und bivariaten Copulas beschreibt. Die Konstruktion eines solchen Modells ist nicht eindeutig und beruht auf einem graphischem Modell, der Vine Baumstruktur. In Dimensionen, welche ein Dutzend Variablen übersteigen, wird diese Struktur unumgänglich komplex und vollständige Modelle leiden unter Überparametrisierung. Daher besteht der Bedarf in hohen Dimensionen gut beschreibende Modelle zu finden, welche außerdem sparsam parametrisiert sind.

Das erste Ziel in dieser Arbeit ist es, alternative Methoden für aktuelle "gierige" Suchverfahren zu finden. Bei diesen ist nicht sichergestellt, dass sie in moderaten Dimensionen in denen sie noch funktionieren, auch gute Lösungen finden. Das zweite Ziel besteht darin, neue Ansätze für Daten in ultrahohen Dimensionen, d.h. einigen Tausend, zu entwickeln und Modelle in guter Qualität und annehmbarer Zeit zu finden. Hierzu werden sowohl gerichtete als auch ungerichtete Gaußsche graphische Modelle sowie Methoden der univariaten Modellselektion benutzt.

Der Hauptbeitrag dieser Arbeit ist nun das Entwickeln mehrerer Methoden, um Gaußsche (graphische) Modelle mit Gaußschen Vine Copulas zu verbinden. Dadurch wird erstens die Strukturschätzung in Vine Copulas erleichtert. Zweitens, erlaubt es uns zudem, den Gaußschen Modellen Parametersparsamkeit aufzuerlegen, um diese in den zugehörigen Vine Copula Modellen zu erhalten. Die vorgestellten Ansätze werden theoretisch begründet und durchgehend illustriert. Da theoretische Vergleiche in hohen Dimensionen schwierig sind, führen wir sowohl Simulationsstudien als auch Anwendungen in hohen bzw. ultrahohen Dimensionen durch, um unsere Ansätze mit dem Goldstandard zu vergleichen. Wir sehen nicht nur, dass wir die aktuelle Standardmethode bzgl. Gütemaßen als auch bzgl. Berechnungszeit schlagen, sondern auch dass wir Bereiche nicht-Gaußscher ultrahoch-dimensionaler Datenmodellierung mit mehr als 2.000 Variablen erstmals erreichen können. Alle vorgestellten Ansätze werden abschließend im Kontext eines hochdimensionalen Finanz-Datensatzes angewandt und verglichen.

Abstract

This thesis extends the usability of vine copulas into ultra high dimensions. Vine copulas are a quite novel flexible statistical model class, which allows to model distributions in finite dimensions by combining arbitrary marginal distributions and bivariate copulas. The construction of such a model is not unique, but relies on a graphical model, the vine tree structure. In dimensions exceeding a dozen variables, the structure becomes inevitably complex, and full models suffer from over-parametrization. Hence, a need exists for finding good fitting parsimonious models in high dimensions.

The first goal in this thesis is to find alternative methods for current greedy-search algorithms. For these, it is not ensured that they obtain suitable solutions in moderate dimensions, where they are still applicable. The second goal is to develop new approaches for data in ultra high dimensions, i. e. several thousand dimensions, to find models of good quality in reasonable time. To this end, both directed and undirected Gaussian graphical models as well as univariate model selection techniques are used.

The main contribution of this thesis is now to develop several methods linking Gaussian (graphical) models to Gaussian vine copulas. Thus, first we make structural estimation in vine copulas more feasible. Secondly, we impose sparsity in the Gaussian models to obtain model parsimony in the associated vine copulas. All developed approaches are theoretically backed and illustrated throughout the thesis. Since theoretical comparisons in high dimensions are hard, we perform simulation studies and both high and ultra high dimensional data applications to compare our approaches to the current benchmark. We see that we not only outperform the current benchmark in terms of goodness of fit and computation time, but we are also able to first enter domains of non-Gaussian modelling of ultra high dimensional data with more than 2,000 variables. All presented approaches are finally applied and compared in the context of a high dimensional financial data set.

Acknowledgements

First and foremost, I want to express my sincere gratitude to my supervisor Prof. Claudia Czado. I am particularly thankful for the opportunity she offered me to return to university after several years outside of academia. She introduced me to vine copulas and always had an open ear for discussions and questions regarding the directions of my research. Furthermore, she steadily encouraged me through the entire time of my dissertation and gave me the freedom to explore different approaches and ideas I had. I am also very grateful for the possibility to present my work at several conferences and to attend a variety of meetings and workshops, which were very valuable experiences.

Secondly, I want to thank Prof. Harry Joe for accepting to act as a referee for my thesis. I truly benefited from his huge experience in dependence modelling and the talks we had during his times in Munich.

I also want to express my gratitude to Prof. Gal Elidan for taking his time and accepting to act as referee for my thesis.

Additionally, I am very thankful for financial support offered by Allianz Deutschland AG as well as TUM Graduate School and the International School of Applied Mathematics (ISAM).

Furthermore, I am happy to remember such an enjoyable time at the Chair of Mathematical Statistics, and thus, I also want to thank my colleagues Nicole, Matthias, Dani, Thomas, Alex and Tobias for inspiring discussions in the office and good times at the conferences we attended.

Finally, I am deeply indebted to my parents for encouraging and enabling me to follow the path I want to pursue. Last, I thank my wonderful wife Christine for her loving support through all the recent years.

Contents

Zusammenfassung	iii
Abstract	v
Acknowledgements	vii
1 Introduction	1
2 Dependence Modelling with Vine Copulas	9
2.1 Copulas	9
2.1.1 Sklar’s Theorem	9
2.1.2 Copula Families	11
2.1.3 Dependence Measures	14
2.1.4 Estimation of Pair Copulas	15
2.2 Vine Copulas	16
2.2.1 Model Assessment	19
2.2.2 Model Selection	20
2.2.3 Model Simplification	21
3 Mathematical Foundations and Sparse Modelling	23
3.1 Mathematical Foundations	23
3.1.1 Time Series Models	24
3.1.2 Conditional Independence	26
3.1.3 Properties of the Multivariate Gaussian Distribution	27
3.2 The Lasso in Linear Regression	28
3.3 Graphical Models	30
3.3.1 Graph Theory	31
3.3.2 Undirected Graphical Models	33
3.3.3 Estimation of Undirected Graphical Models	34
3.3.4 Directed Graphical Models	37
3.3.5 Estimation of Graphical Models on DAGs	40
3.4 Structural Equation Models (SEMs)	42
4 Representing Sparse Gaussian DAGs as Sparse R-vines	45
4.1 Motivation: Combinatorial Example	46
4.2 Representing Truncated R-vines as DAGs	46

4.3	Representing DAGs as Truncated R-vines under Sufficient Conditions	48
4.3.1	Representing 1-DAGs as 1-Truncated R-vines	48
4.3.2	Representing k -DAGs under Sufficient Conditions	48
4.3.3	Special k -DAGs with R-vine Representations	52
4.3.4	A Necessary Condition for R-vine Representations	53
4.4	Representing Arbitrary k -DAGs	56
4.4.1	Algorithmic Implementation	56
4.4.2	Toy Example	58
4.5	Application	59
5	Selection of Sparse Vine Copulas in High Dimensions with the Lasso	63
5.1	Motivation: SEM Representation of R-vines	63
5.2	Vine Copula Structure Selection with the Lasso	69
5.2.1	Calculation of the Ordering Function	70
5.2.2	Sparse R-vine Structure Selection	73
5.2.3	R-vine Regularization Paths	76
5.2.4	Selection of the Tuning Parameter	79
5.3	Application	81
6	Dependence Modeling in Ultra High Dimensions with Vine Copulas and the Graphical Lasso	83
6.1	A Divide-And-Conquer-Strategy	83
6.1.1	Motivation: Considering Computational Complexity	83
6.1.2	Clustering High Dimensional Data	84
6.1.3	Improving Estimation Accuracy	87
6.2	Algorithmic Implementation	89
6.2.1	Computational Implementation	89
6.2.2	Toy Example	90
6.3	Application	97
7	Comparison Studies Utilizing the Proposed Methods	105
7.1	Set up of the Comparison	106
7.1.1	Data Preparation	106
7.1.2	Numerical Implementation	107
7.2	Simulation Study	107
7.3	Runtime Comparison	129
7.4	Value at Risk (VaR) Forecasting and Backtesting in 400 Dimensions	137
7.4.1	Data Preparation and Modelling	137
7.4.2	Value at Risk (VaR) One Day Ahead Forecasting	140
7.4.3	Value at Risk (VaR) Backtesting	145
8	Conclusion and Outlook	161
8.1	Contribution	161
8.2	Future Research Directions	162

A	Supplementary Material to Chapter 4	173
A.1	DAGs Estimated on the Euro Stoxx 50 Data Set	174
A.2	Numerical Results of Fitted Models	180
A.3	Algorithms to Chapter 4	182
B	Algorithms to Chapter 5	185
C	Algorithm <code>RVineClusterSelect</code>	189

Chapter 1

Introduction

In recent years and decades, the amount of data which is available worldwide for analysis increased exponentially, see e.g. the work of Han et al. (2011). The purpose of the analysis is not self-evident, but to improve the understanding of phenomena in many domains of science and applications. Furthermore, it should also enable us to predict future developments. Thus, the complexity imposed by data sets becoming larger must be offset by either more sharply increasing computational power, or more sophisticated statistical methods. The goal of this thesis is to add the latter to the body of research, more precisely, new and fast approaches to find *non-Gaussian dependence models in ultra high dimensions*.

We start with an illustrating example in the context of financial risk management. It should however be noted upfront, that our methods are also applicable to any other area of research, for example engineering, biology or meteorology.

Let us consider a company which owns $d = 100$ stocks S_1, \dots, S_{100} which are composed in an equally weighted portfolio P , and whose value *today* is

$$\mu_t^P = \sum_{j=1}^{100} S_t^j.$$

Let the *distribution function* of μ_t^P be given by F_P . Additionally, assume that modelling the individual stocks behaviour is easy, for example using methods of Taylor (2008). Now, our company wants to control the risk associated with this portfolio. Hence, it wants to have an idea in what range the value of the portfolio will lie *tomorrow*, i. e. if we can find *upper and lower bounds* such that tomorrows portfolio value can be bounded with high probability. Of course, lower bounds for *losses* are far more important than upper bounds for exceeding *gains*. This is the concept of the *Value at Risk (VaR)* at level α , defined by

$$\text{VaR}_{1-\alpha}(P) = \inf \{x | F_P(x) \geq 1 - \alpha\}. \quad (1.1)$$

This quantity defines how low tomorrows portfolio value can be at worst with probability $1 - \alpha$. In most cases and also demanded for example by bank regulation in the European Union under the Basel Accord (Basel Committee on Banking Supervision, 2006), $\alpha = 0.01$ over a 10 day time horizon. Having models for the univariate marginal time series, it is straightforward to calculate their individual VaR. However, the portfolio VaR can not be

easily calculated based on the marginal Value at Risk, see McNeil et al. (2015). This is because the stocks in the portfolio are *dependent* and influence each other. On the one hand side, this is beneficial because of the *diversification*, i. e. a decrease in one stock may be associated to an increase in another stock and offset a loss. However, it has been observed that in times of market turmoil, diversification is lost as a *contagion* effect occurs (Kenourgios et al., 2011). This means, losses of one stock are no longer offset by other gains but accompanied by other losses. If this is not covered in the portfolio VaR model, the associated risk is dramatically underestimated and might endanger the financial stability of the entire company owning the stocks. Hence, a joint modelling approach must be undertaken to account for such behaviour. In recent times, the *multivariate Gaussian distribution* has been the method of choice. However, even though the model is able to describe multivariate behaviour, it lacks other favourable features. Most importantly, the Gaussian distribution is not able to model *fat tails*. This means, highly unlikely but severe events modelled in the *tail of the distribution function* can not be described by the Gaussian distribution as it has no probability mass in the tails. However, this is explicitly the behaviour which is most challenging in terms of stock markets moves and also of interest in other domains of science, where extremal events are most harmful, for example, weather applications, see Yu et al. (2017). Furthermore, it also assumes symmetric dependence, which might not be justified empirically all the time. Most prominently, the financial crisis starting in 2007 was also directly linked to the use of Gaussian methods, see Salmon (2012). From that point, it is clear that more flexible models are demanded, especially in this domain.

A first method to overcome the strict assumptions using multivariate distributions is achieved by using *copulas*, see e.g. Joe (2014) or Nelsen (2006). This very appealing framework allows to separate *univariate marginal distributions* from the *joint distribution*, and thus, the *dependence behaviour*, according to the famous Theorem of Sklar (1959). Thus, modelling a d -dimensional distribution decomposes into modeling of the d marginal distributions and a d -dimensional copula. Considering the latter, there exists a vast number of *parametric copula functions* in $d = 2$ dimensions, i. e. in the *bivariate* case, see Joe (2014, Chapter 3), Furthermore, also non-parametric estimation of copulas is feasible (Chen and Huang, 2007). Going to higher dimensions however, copula functions are becoming more complex to describe in the parametric case. Furthermore, non-parametric estimation suffers from the *curse of dimensionality*, i. e. data becoming sparse in higher dimensions and slowing down the estimators. Based on earlier work by Joe (1996) and Bedford and Cooke (2001, 2002), the *pair copula construction (PCC)* of Aas et al. (2009), develops a d -dimensional copula using only bivariate copulas as building-blocks. In a nutshell, $d(d - 1) / 2$ pair copulas are combined to obtain a joint distribution in d dimensions. This transforms the problem of estimating a d -dimensional copula into $d(d - 1) / 2$ pair copulas. However, the pair copula construction is not unique but relies on a graphical tool, the *regular vine (R-vine)*. Starting with a tree on d nodes, $d - 2$ trees follow subsequently chosen according to a specific rule set called the *proximity condition*, determining the structure how the $d(d - 1) / 2$ pair copulas are composed. As shown by Kurowicka

and Joe (2011, Chapter 9), in d dimensions there exist

$$\frac{d!}{2} \times 2^{\binom{d-2}{2}},$$

R-vine structures and we are challenged by a complex combinatorial model selection problem. This super exponentially growing number makes it infeasible to evaluate all models, even in small dimensions. For intermediate dimensions, the algorithm of Dißmann et al. (2013) has proven quite successfully using a *greedy search strategy*. Yet, this strategy can not be proven to be optimal and might lead to sub-optimal models, given that its applicable at all. As we will see later on, it will fail to work in several hundred dimensions. Of course the question arises whether this is an important problem. We argue by reconsidering our previous example of a company safeguarding its portfolio. To carry on the argument, it is also quite realistic for such a company to not only own 100 but $d_S = 1000$ stocks and also d_B *bonds*, i. e. fixed income financial instruments, which are also interlinked to the stock markets. Furthermore, assume it also holds d_{FX} *foreign exchange financial instruments* which should be considered as well, as the change in value of a currency pair always affects the corresponding stock markets of the countries. Finally, assume our portfolio also contains d_C *commodities* as for example precious metals or agricultural products. Their price moves could also affect the stocks our company is holding because the companies associated to these stocks either produce or use the corresponding commodities. Thus, the total dimension of such a portfolio risk model would scale to

$$d = d_S + d_B + d_{FX} + d_C \gg 1000.$$

For such a purpose, a very high dimensional dependence model is needed. However, this model should also reflect some parsimony. For example there might be strong dependence of stocks in some industry sectors with some of the associated commodities these industries produce or use. Additionally, companies having business activities in specific regions can also be more exposed to specific foreign exchange pairs related to these regions, i. e. having regional effects. It is realistic to assume that having modelled these *intra-sectoral* dependencies, the remaining dependencies are quite negligible and can be omitted to obtain parsimonious models in the end. The same considerations also apply to several other financial applications described by Aas (2016), which one can easily imagine to extend to several hundreds or thousands of dimensions.

All these considerations, measuring and modelling dependence, posed in the context of financial services can also be extended to either thousands of weather measurements (Yu et al., 2017), water discharge stations (Pereira et al., 2017) or metabolites measured in the human system (Krumstiek et al., 2011). We see not only that there is a huge number of problems demanding for flexible dependence models, but they are also high dimensional, if not *ultra high dimensional*, i. e. with $d > 1000$. This thesis aims to contribute to the research in this domain.

The first contribution of this thesis is as following. First, we introduce copulas, vine copulas and models based on the multivariate Gaussian distribution, either in terms of

graphical models or so called *structural equation models (SEM)*. These models serve as a bridge which allows us to relate the tree structure of an R-vine to a specific multivariate Gaussian distribution. Since the structural inference for the latter can be solved much faster, we then use our theoretical results relating both models classes, to use the Gaussian models as *proxy* for the tree structure in the R-vine. All of the mentioned Gaussian models induce some kind of model parsimony or *sparsity*, which is related to *(conditional) independence* in these models. Fortunately, this is the same kind of model parsimony which can be expressed in a vine copula, since (conditional) independence assumptions make our vine model more parsimonious. As we can regulate the sparsity in the Gaussian proxy models, we also have a natural tool to handle sparsity in the vine copula models. We want to stress that we believe that Gaussian modelling approaches are not sufficient for a variety of real world phenomena. However, because of their unmatched simplicity, allowing to describe high dimensional models in closed form expressions and the accompanying computational tractability, we consider them as very useful intermediate step from the data to more complex dependence models. Furthermore, we want to exploit the manifold of research results which have been obtained with respect to Gaussian models in recent decades. The second main contribution is based on the practical usability of the presented approaches. Since the ultimate goal of our work is not only to find methods to develop high dimensional dependence models, but also make them usable in the scientific community for practical applications, all our algorithms are implemented in the statistical software R (R Core Team, 2017) using some C++ code for more computationally demanding tasks. Thus, the thesis is also containing a large chapter which contains numerical results, showcasing that our approaches do not only work well in practical applications. Even more, they are also to the authors knowledge the only feasible approaches to estimate high dimensional vine copulas with a high degree of model parsimony. Thus, we extend the usability of vine copula models from about $d = 50$ to $d > 2000$, paving the way for widespread application of very promising dependence models in many possible areas of research.

Outline of the Thesis

In Chapter 2, we will first of all give a brief introduction into dependence modelling with vine copulas. To this end, we will first consider *copulas* in general. We will present the Theorem of Sklar (1959) and introduce properties of copulas and some of the most prominent pair copula families, the so called *elliptical* and *Archimedean* copulas. We will briefly discuss *parameter estimation* and then continue to *measures of bivariate dependence* such as the *Pearson correlation* or *Kendall's τ* . We then move on and introduce vine copulas. We show how the pair copula construction (PCC) works in an explicit example in six dimensions and discuss how the corresponding models can be stored and treated algorithmically. We will next discuss how to compare different vine copula models and then recapitulate the *benchmark model selection strategies* which have been in place for several years now, but also shed light on new developments. We discuss their properties especially with respect to high dimensional problems. Finally, we also consider the topic

of *model simplification* for vine copulas and how parsimonious models can be obtained. Chapter 3 introduces the methods we are going to use subsequently to first find structures for high dimensional copula models and second, to impose sparsity in these. Starting in Section 3.1, we will describe *time series models* since it is vital to understand the marginal behaviour of the considered financial data sets before continuing to the multivariate modelling. We introduce properties of *conditional independence* since this will be the mathematical equivalent of *sparsity* in terms of this thesis. We also shed light on the *multivariate Gaussian distribution* and its properties we are going to exploit later. Section 3.2 introduces the well-known *Lasso* invented by Tibshirani (1994), solving linear regression equations in a parsimonious way by introducing a penalization parameter for too complex models. It is basically a univariate method, but can also be applied to a set of equations, jointly defining a multivariate distribution. Section 3.3 then introduces *graphical models*, both on *undirected* and *directed* graphs. We provide the necessary *graph theory* allowing us to describe these objects and their properties. Also, we discuss how estimation of these models works given some underlying Gaussian distribution. The chapter is finished by introducing *structural equation models (SEMs)*. These are sets of linear equations, describing a joint multivariate distribution, and will come into play together with the Lasso later on.

The first approach is introduced in Chapter 4, where we connect graphical models on *directed acyclic graphs (DAGs)* to R-vines. We motivate our approach by considering how first of all an R-vine can be represented by a DAG. This then leads to inverting the consideration and defining a *R-vine representation of a DAG* which is our mathematical formalization of the *proxy* we referred to previously. Starting with a particular special case of a so called 1-DAG, we will present sufficient conditions under which a general k -DAG can be represented by a *k-truncated R-vine*, which is a specific parsimonious R-vine. We will prove a main representation theorem and illustrate the representation thoroughly. Next, we will derive explicit special cases of k -DAGs for which general representations exist and introduce also a necessary condition for our results. We are then considering a general algorithmic implementation, allowing us to represent arbitrary k -DAGs by R-vines. We illustrate our algorithm and finish the chapter using a real world data example in $d = 52$ dimensions. We denote this method the *DAG approach*.

In Chapter 5, we use a connection between R-vines and the previously introduced structural equation models (SEMs). Using the *Lasso*, these structural equations can be solved parsimoniously to resemble an R-vine structure. We will first show the theoretical connection and illustrate how sparsity in these equations refers to sparsity in the R-vine. Then, we present approaches to determine both the so called *order* of the equations as well as the *non-zero* regression coefficients. Both these can be linked to the structure of R-vines captured in so called *R-vine matrices*. By transferring concepts of the Lasso to R-vines, we propose a method to obtain various parsimonious models based on a single R-vine structure, called the *R-vine regularization path*. We close with an example in $d = 222$ dimensions. We refer to this method as the *Lasso approach*, based either on a *single* or *adaptive thresholding*.

Finally, these methods are already much faster than the benchmark and allow more parsi-

mony and flexibility. Yet they are similar in how they work to the benchmark algorithm of Dißmann et al. (2013), and hence also suffer from quickly increasing dimensions. The third approach, referred to as *cluster method* and described in Chapter 6 rethinks the problem entirely. Instead of solving a difficult d dimensional problem, we consider a set of problems in $d_T \ll d$ dimensions, where T indicates a *threshold dimension*. Because of the quadratic nature of the numbers of required parameters, solving these smaller problems and then recombining at a latter stage is much more computationally favourable. Of course, this demands a sound method for *clustering* the original problems into feasible sub-problems. This is again attained using graphical models. Moreover, these methods not only allow to cluster the original problem, but also estimate the *sub-R-vines* with increased accuracy compared to the standard benchmark algorithm developed by Dißmann et al. (2013), by exploiting conditional independence, inherent in the graphical model. This approach then elevates R-vines into a completely novel range of *ultra high dimensional data sets* with more than $d > 2000$ variables, which is demonstrated by a corresponding example in this section.

After the theoretical part, we evaluate our methods in practical applications in Chapter 7. First, we use simulated financial data in $d = 85$ dimensions of differently sparse scenarios to make assertions about the quality of our approaches in domains where also the benchmark is generally working well. We see comparable results, however already more in our favour with respect to looking for sparsity by considering *penalized goodness of fit measures*. Next, we carry out a runtime comparison, where we show that the benchmark algorithm is quite limited in terms of models that can be selected because of complexity. Furthermore, we demonstrate that our methods are numerically tractable even when we consider thousands of dimensions, without sacrificing model accuracy, i. e. goodness of fit. Finally, we return to the initially described example of safeguarding portfolio risk, i. e. calculating *Value at Risk* (1.1) of a portfolio of 400 stocks and use real world data to compare our proposed approaches to the benchmark and towards each other. We want to stress again that even though the models are fitted on financial data, this is neither the sole purpose nor the only possible application, but happens because of data availability and easy interpretability of the models compared to other domains of science.

In Chapter 8, we summarize our contribution and recapitulate the advantages and caveats of our proposed methods. We describe the current state of the research and give outlooks for further research, to enable applicability of even higher dimensional and versatile models.

For a better overview, Figure 1.1 positions the content of the thesis in the current body of research and shows connections between the corresponding chapters and references. Hereby, directed acyclic graphs (DAGs) are the main tool we use in Chapter 4, structural equation models (SEMs) are used in Chapter 5 and Gaussian graphical models (GGMs) in Chapter 6.

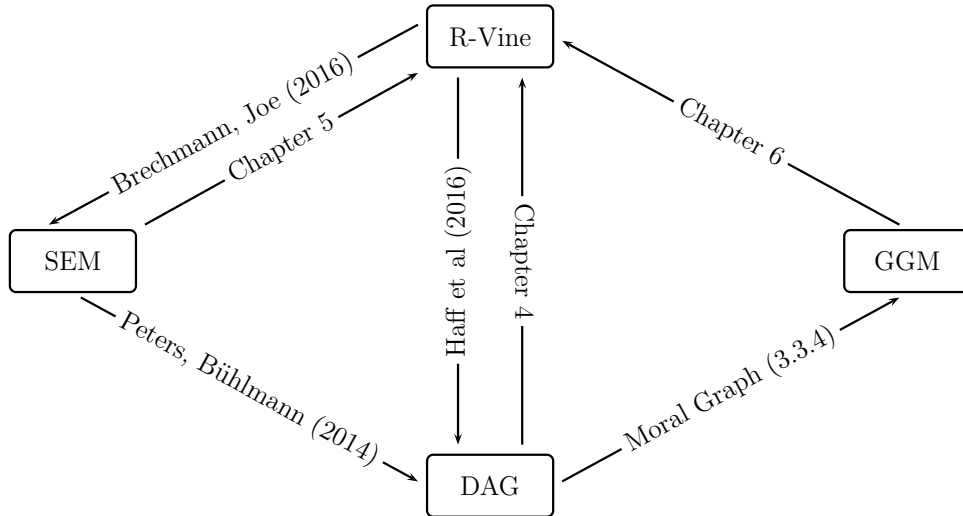


Figure 1.1: Connections between the current body of research and the results of the thesis.

The work in this thesis contains, summarizes and extends results described in previously published manuscripts.

- Chapter 4 is derived in part from an article published in the *Journal of Computational and Graphical Statistics* on 04.08.2017, available online using the link <http://amstat.tandfonline.com/doi/full/10.1080/10618600.2017.1366911>, and referenced as Müller and Czado (2017b).
- Chapter 5 is based on an article submitted for publication in *Statistics & Computing* and publicly available online using the link <https://arxiv.org/abs/1705.05877>, and referenced as Müller and Czado (2017c).
- Chapter 6 is based on an article submitted for publication the *Journal of Machine Learning Research* and publicly available online using the link <https://arxiv.org/abs/1709.05119>, and referenced as Müller and Czado (2017a).

Chapter 2

Dependence Modelling with Vine Copulas

We will now introduce the methods we are using to model multivariate data. First, we are going to introduce *copulas* in Section 2.1 which allow us to flexibly model random vectors by separating marginal behaviour from dependence. We discuss different so called *copula families* and their properties and estimation in the bivariate case. Second, we will scale the use of copulas to arbitrary dimensions introducing *vine copulas* in Section 2.2. After their introduction, we discuss several methods to compare different vine copula models and describe previous model selection approaches, as for example the current benchmark of Dißmann et al. (2013). This section is based on Müller and Czado (2017a).

2.1 Copulas

We will use the following conventions in the remainder of this thesis. Upper case letters X denote random variables, and lower case letters x their realizations. We use bold letters \mathbf{X} for random vectors and \mathbf{x} for the vector of realizations. Matrices M are identified by upper case letters, but are clearly distinguished from random variables in the text. We write x_i for the i -th entry of the vector \mathbf{x} . Denote sub vectors of $\mathbf{x} = (x_1, \dots, x_d)^T$ by $\mathbf{x}_D := (x_j)_{j \in D}$. When considering matrices, we denote $m_{i,j}$ the j -th entry in the i -th row of the matrix $M = (m_{i,j})_{i=1, \dots, n, j=1, \dots, d} \in \mathbb{R}^{n \times d}$. For rows or columns of a $d \times d$ matrix M , we write $M_{,j} = (m_{1,j}, \dots, m_{d,j})$ for the j -th column and $M_i = (m_{i,1}, \dots, m_{i,d})$ for the i -th row of M , respectively.

2.1.1 Sklar's Theorem

For a random vector $\mathbf{X} = (X_1, \dots, X_d)$ we denote the joint distribution function and density by $F(x_1, \dots, x_d)$ and $f(x_1, \dots, x_d) = \frac{\partial F}{\partial x_1, \dots, \partial x_d}(x_1, \dots, x_d)$, respectively. Additionally, denote the corresponding marginal distribution functions by $F_1(x_1), \dots, F_d(x_d)$ with marginal density functions $f_1(x_1), \dots, f_d(x_d)$, for $i = 1, \dots, d$. The marginal distributions can be modelled by a wide array of parametric probability distributions or also estimated non-parametrically. This is however not necessarily easy for the joint distribution in d dimensions. For example, multivariate extensions of the univariate Gaussian

distribution or the Student's-t distribution can be used in arbitrary dimensions. However, these also imply that each marginal distribution has a univariate Gaussian or Student's-t distribution, respectively. Since this is a very strong assumption, especially in high dimensions, more flexible models are needed, overcoming these restrictions. A promising way is to use *copulas*. We define a *copula* $C : [0, 1]^d \rightarrow [0, 1]$ as a multivariate distribution function with uniform marginal distributions. Since copulas are defined on the unit interval, we introduce now three data scales which are important in the remainder of this thesis.

- The *x-scale*, given as the original scale of X_i with density $f_i(x_i)$, $i = 1, \dots, d$.
- The *u-scale* or *copula-scale*, given by $U_i = F_i(X_i)$, where F_i is the cumulative distribution function (cdf) of X_i and $U_i \sim \mathcal{U}[0, 1]$, $i = 1, \dots, d$ where $\mathcal{U}[0, 1]$ denotes a standard uniform distribution on $[0, 1]$.
- the *z-scale* or *marginally normalized scale*, given by $Z_i = \Phi^{-1}(U_i)$, where Φ is the cumulative distribution function of a standard Gaussian (normal) distribution denoted by $\mathcal{N}(0, 1)$, thus $Z_i \sim \mathcal{N}(0, 1)$, $i = 1, \dots, d$.

The *z-scale* will also occur when considering models assuming that the data arises from a multivariate Gaussian distribution.

The connection between the multivariate distribution F we want to model and the copula is now the famous Theorem of Sklar (1959). It separates the marginal distribution functions from the joint distribution function. More precisely, for each multivariate distribution function F with univariate marginals F_i , $i = 1, \dots, d$, there exists a copula C such that

$$F(x_1, \dots, x_d) = C(F_1(x_1), \dots, F_d(x_d)). \quad (2.1)$$

If all marginal distributions F_i , $i = 1, \dots, d$ are continuous, C is unique. The corresponding density f with respect to the copula is obtained by taking derivatives in (2.1) to obtain

$$f(x_1, \dots, x_d) = c(F_1(x_1), \dots, F_d(x_d)) \prod_{i=1}^d f_i(x_i), \quad (2.2)$$

with $c(u_1, \dots, u_d) = \frac{\partial C}{\partial u_1 \dots \partial u_d}(u_1, \dots, u_d)$ the *copula density* and $(u_1, \dots, u_d) \in [0, 1]^d$. The important fact is now, that given multivariate data, we can model the univariate, i. e. idiosyncratic behaviour and the dependence part completely isolated from each other. This is clearly a benefit over, e. g. the multivariate Gaussian distributions, which require also Gaussian marginals.

A very favourable feature of copulas is their ability to model *extremal behaviour*, or in other words *tail dependence*, which describes dependence in the tails of the corresponding bivariate distribution. More formally, let (X_1, X_2) random variables with marginal distributions F_1, F_2 and copula C , i. e.

$$F(x_1, x_2) = C(F_1(x_1), F_2(x_2)).$$

We define the *upper tail dependence* between X_1 and X_2 by

$$\lambda_{upper} := \lim_{u \nearrow 1} \mathbb{P}(X_2 > F_2^{-1}(u) \mid X_1 > F_1^{-1}(u)) = \lim_{u \nearrow 1} \frac{C(u, u)}{u},$$

and the *lower tail dependence* by

$$\lambda_{lower} := \lim_{u \searrow 0} \mathbb{P}(X_2 \leq F_2^{-1}(u) \mid X_1 \leq F_1^{-1}(u)) = \lim_{u \searrow 0} \frac{1 - 2u + C(u, u)}{1 - u},$$

given the existence of the limits in the unit interval, i.e. being a probability. Tail dependence greater than zero then describes that X_2 tends to an extremal value once X_1 does. Of course, tail dependence equals to zero means no tail dependence. Clearly, tail dependence is a property of the copula C . We will now present some of the most often used parametric copula models, i.e. parametric forms for the copula function C .

2.1.2 Copula Families

Since a copula is a function $C : [0, 1]^d \mapsto [0, 1]$, describing it explicitly can become complex in the general case. Thus, we will only introduce the two most prominent, the *Gaussian* and *Student's-t* copula in arbitrary dimensions and then continue with the bivariate case of $d = 2$ which will be required in the remainder of the thesis. First note, that from inversion of (2.1) we have

$$C(u_1, \dots, u_d) = F(F_1^{-1}(u_1), \dots, F_d^{-1}(u_d)),$$

for $u_1, \dots, u_d \in [0, 1]^d$. Thus, for any choice of a joint distribution function F and univariate marginal distributions F_i , $i = 1, \dots, d$, a copula is specified. Two representatives of the *elliptical distributions*, the Gaussian and Student's-t distribution give rise to the Gaussian and Student's-t copula. For the Gaussian distribution in d dimensions we obtain

$$C_{\Sigma}^{Gauss}(u_1, \dots, u_d) = \Phi_{\Sigma}(\Phi^{-1}(u_1), \dots, \Phi^{-1}(u_d)),$$

where Φ denotes the cdf of $\mathcal{N}(0, 1)$ and Φ_{Σ} denotes the distribution function of a d -dimensional multivariate Gaussian distribution with zero mean, unit variances and correlation matrix Σ . We examine the density in two dimensions more precisely.

Example 2.1 (Gaussian copula density in two dimensions). *Let $d = 2$ and assume the correlation matrix*

$$\Sigma = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \text{ with } \rho \in [-1, 1].$$

Then, the Gaussian copula density in two dimensions is given by

$$c(u_1, u_2) = \frac{1}{\sqrt{1 - \rho^2}} \exp\left(-\frac{\rho^2(u_1^2 + u_2^2) - 2\rho u_1 u_2}{2(1 - \rho^2)}\right).$$

The same procedure can be applied to the Student's-t distribution to obtain

$$C_{\Sigma, \nu}^{Student's-t}(u_1, \dots, u_d) = t_{\Sigma, \nu}(t_{\nu}^{-1}(u_1), \dots, t_{\nu}^{-1}(u_d)),$$

where Σ is a positive semi-definite $d \times d$ matrix and $\nu > 0$. Additionally, let t_{ν} denote the cdf of a random variable following a Student's-t distribution with ν degrees of freedom and $t_{\Sigma, \nu}$ be the distribution function of a d -dimensional random variable following a multivariate Student's-t distribution with correlation matrix Σ and degrees of freedom ν . Contrary to the Gaussian copula, the Student's-t copula can feature upper and lower tail dependence. This makes it suitable for financial applications, where tail dependence is often monitored in times of market turmoil.

Another class of copulas is described by the *Archimedean copulas*, see Nelsen (2006, Chapter 4). Let $\phi : [0, 1] \rightarrow [0, \infty]$ a continuous, strictly monotonic decreasing and convex function. Additionally, let $\phi(1) = 0$ and define ϕ^{-1} as the generalized inverse of ϕ by

$$\phi^{-1}(y) = \begin{cases} \phi^{-1}(y), & 0 \leq y \leq \phi(0), \\ 0, & \phi(0) < y \leq \infty. \end{cases}$$

With this *generator* ϕ , we can define a bivariate copula by

$$C_{\phi}(u_1, u_2) = \phi^{-1}(\phi(u_1) + \phi(u_2)).$$

Note that we omit higher dimensional Archimedean copulas here since they will not be used in the remainder of the thesis. Depending on the generator function ϕ , we obtain different well known copula families. These are for example the *Clayton*, *Gumbel*, *Frank* or *Joe* copula family, see Joe (2014, Chapter 3). We summarize these families briefly.

- The *Clayton* copula with lower tail dependence has the generator $\phi(t) = \frac{1}{\theta} \left(\frac{1}{t^{\theta}} - 1 \right)$ and

$$C(u_1, u_2) = (u_1^{-\theta} + u_2^{-\theta} - 1)^{\frac{1}{\theta}} \text{ for } \theta \geq 1.$$

- The *Gumbel* copula with upper tail dependence has the generator $\phi(t) = (-\log t)^{\theta}$ and

$$C(u_1, u_2) = \exp \left(- \left((-\log u_1)^{\theta} + (-\log u_2)^{\theta} \right)^{\frac{1}{\theta}} \right) \text{ for } \theta > 0.$$

- The *Frank* copula with no tail dependence has the generator $\phi(t) = -\log \left(\frac{\exp(-\theta t) - 1}{\exp(-\theta) - 1} \right)$ and

$$C(u_1, u_2) = -\frac{1}{\theta} \log \left(1 + \frac{(\exp(-\theta u_1) - 1)(\exp(-\theta u_2) - 1)}{\exp(-\theta) - 1} \right) \text{ for } \theta \in \mathbb{R} \setminus \{0\}.$$

- The *Joe* copula with upper tail dependence has the generator

$\phi(t) = -\log(1 - (1 - t)^\theta)$ and

$$C(u_1, u_2) = 1 - \left((1 - u_1)^\theta + (1 - u_2)^\theta - (1 - u_1)^\theta (1 - u_2)^\theta \right)^{\frac{1}{\theta}} \text{ for } \theta > 1.$$

For a detailed discussion, see the books of Joe (2014, Chapter 4) or Nelsen (2006, Chapter 3). All of these families can also be rotated by 90° , 180° and 270° by considering

- $c_{90}(u_1, u_2) = c(1 - u_2, u_1)$,
- $c_{180}(u_1, u_2) = c(1 - u_1, 1 - u_2)$,
- $c_{270}(u_1, u_2) = c(u_2, 1 - u_1)$.

For example, a 180° rotation of a copula with lower tail dependence yields a copula with upper tail dependence and vice versa. Additionally, there also exist two parametric copula families, the *BB-copula* families, see Joe (1997, Chapter 5), which will not be covered here. It suffices to acknowledge that there exists a large number of parametric pair copula families, including rotations, which can describe different features of bivariate data, e. g. tail dependence or asymmetric dependence.

All these pair copula families have in common that the strength of the dependence they describe is governed not only by their functional form, but also by a *copula parameter* θ . Consider for example the Gaussian copula again.

Example 2.2 (Example 2.1 cont.). *Consider the bivariate Gaussian copula generated by a two-dimensional Gaussian distribution with correlation matrix*

$$\Sigma = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \text{ with } \rho \in [-1, 1].$$

For $\rho = 1$, we have perfect positive dependence and for $\rho = -1$ we have perfect negative dependence. Note also that $\rho = 0$ describes independence.

Similarly for the Archimedean copulas, the parameter θ determines the corresponding strength of dependence. Estimation of this parameter is an additional aspect which we consider in Section 2.1.4. Finally, if we want to model no dependence or *independence*, we use the *independence copula* given by

$$C^\perp(u_1, \dots, u_d) = \prod_{i=1}^d u_i \text{ with copula density } c^\perp(u_1, \dots, u_d) = 1.$$

In addition to the parametric pair copula families, also *non-parametric* estimation of copulas is possible, see for example Nagler (2014). Hence, pair copulas are very flexible tools for modelling bivariate dependence.

However, a multivariate extension of these dependence models is not straightforward. While the elliptical copulas follow very strict parametric assumptions, Archimedean copulas are not as flexible as their behaviour in d dimensions is governed only by a single

parameter. This limits the applicability in practical considerations by not accounting for more complex dependence patterns. Furthermore, also non-parametric estimation fails in higher dimensions because of the *curse of dimensionality* (Bellman, 2015). This means roughly, that multivariate data becomes sparse in high dimensions, which leads to very slow convergence rates for non-parametric estimators, see also Nagler (2014). Hence, we will only work with bivariate copula families in the remainder of the thesis.

2.1.3 Dependence Measures

An important topic and the main reason to perform dependence modelling, is describing and modelling association between two or more random variables. First, we need to *quantify* the strength of dependence by *scalar dependence measures*. The most well known of these is the *Pearson product-moment correlation*, which is defined as follows. Consider two random variables X, Y and define

$$\rho_{X,Y} := \rho(X, Y) = \frac{\text{Cov}(X, Y)}{\sqrt{\text{Var}(X)}\sqrt{\text{Var}(Y)}},$$

with

$$\text{Cov}(X, Y) = \mathbb{E}(XY - \mathbb{E}(X)\mathbb{E}(Y)) \text{ and } \text{Var}(X) = \text{Cov}(X, X).$$

By the Cauchy-Schwarz inequality, we have that $\rho_{X,Y} \in [-1, 1]$. In our considerations, we will also need the dependence of two random variables given other random variables. An extension of the product moment correlation is the *partial correlation* of X and Y given a k -dimensional random vector \mathbf{Z} , denoted $\rho_{XY;\mathbf{Z}}$ which is the correlation of X and Y after having removed the effect of \mathbf{Z} (Kurowicka and Joe, 2011, p. 47). It is recursively defined on basis of correlations and lower order partial correlations. More precisely, let $\mathbf{Z} \in \mathbb{R}^{d-1}$, and $W \in \mathbb{R}$, then the partial correlation of X and Y given \mathbf{Z} can be recursively defined by

$$\rho_{X,Y;\mathbf{Z} \cup W} = \frac{\rho_{X,Y;\mathbf{Z}} - \rho_{X,W;\mathbf{Z}}\rho_{W,Y;\mathbf{Z}}}{\sqrt{(1 - \rho_{X,W;\mathbf{Z}}^2)(1 - \rho_{W,Y;\mathbf{Z}}^2)}}. \quad (2.3)$$

A drawback of the Pearson correlation is the fact that it is not invariant with respect to monotone transformations. Instead, we will most often consider a dependence measure which satisfies this important property, *Kendall's τ* . It is defined as the probability of concordance minus the probability of discordance of two random variables. For this, let (X_1, Y_1) and (X_2, Y_2) be i.i.d. copies of (X, Y) and define

$$\tau_{X,Y} = \tau(X, Y) := \mathbb{P}((X_1 - X_2)(Y_1 - Y_2) > 0) - \mathbb{P}((X_1 - X_2)(Y_1 - Y_2) < 0).$$

As for the Pearson correlation, $\tau_{X,Y} \in [-1, 1]$. If the dependence of the pair (X, Y) is governed by a copula, Kendall's τ of (X, Y) can be directly calculated based on the copula and is independent of the corresponding marginal distributions. More precisely, let (X, Y)

be samples from a copula C , then

$$\tau_{X,Y} = \int_{[0,1]^2} C(u_1, u_2) dC(u_1, u_2) - 1. \quad (2.4)$$

If C belongs to the class of elliptical or Archimedean copulas, C also depends on some copula parameter θ . Thus, also $\tau_{X,Y}$ depends on θ in this case. Hence, assuming a specific copula gives an association between the copula parameter and Kendall's τ . We examine this dependence for the Gaussian copula.

Example 2.3 (Example 2.2 cont.). *Let $(X, Y) \sim C$ where C is a Gaussian copula with correlation $\rho \in [-1, 1]$. Thus, we have for Kendall's τ*

$$\tau(X, Y) = \frac{2}{\pi} \arcsin(\rho), \quad (2.5)$$

see Joe (2014, p. 58).

We will use relationships like these in the next section when we continue with methods for the estimation of bivariate copulas based on data.

2.1.4 Estimation of Pair Copulas

Assume that we have a sample of data on the x -scale, $\{(x_{i,1}, x_{i,2}), i = 1, \dots, n\}$ which we want to describe using copulas. Thus, we first have to obtain u -scale data. Using the corresponding marginal distribution functions F_1 and F_2 , we obtain data

$$\{(u_{i,1} = F_1(x_{i,1}), u_{i,2} = F_2(x_{i,2})), i = 1, \dots, n\},$$

on the *copula scale*. In a data-driven set up, the marginal distribution functions F_1 and F_2 are not necessarily known and hence subject to estimation. If the corresponding marginal distribution functions are parametrically estimated, we call this the *inference for margins (IFM)* approach, see Joe (2005) and if they are non-parametrically estimated, we speak of a semi-parametric approach (Genest et al., 1995). To estimate a pair copula on the u -scale data, we can now either fit any of the described parametric pair copula families using a *maximum likelihood estimation (MLE)* to infer a parameter estimate $\hat{\theta}$. More precisely, the estimator which maximizes the log-likelihood is

$$\hat{\theta} = \arg \max_{\theta \in \Theta} \sum_{i=1}^n \log(c(u_{i,1}, u_{i,2}, \theta)),$$

where c is the corresponding copula density, parametrized by some copula parameter $\theta \in \Theta$, which is a parameter domain, depending on the corresponding pair copula family. This optimization problem can be solved with standard numerical optimization algorithms. Another method to estimate the corresponding copula parameter θ is by *inversion* of the empirical Kendall's $\hat{\tau}$. With (2.4), we have $\tau = g(\theta)$ for some function g which depends

on the concrete choice of parametric copula family. Hence, we can use an empirical estimator for τ to calculate $\hat{\theta} = g^{-1}(\hat{\tau})$. For most of the parametric copula families we introduced, the corresponding functions g are well known. However, the inversion method is only possible for copula families with one parameter, while the MLE approach works for arbitrary families. We consider this in the example of the Gaussian copula.

Example 2.4 (Example 2.3 cont.). *Let $(X, Y) \sim C$ where C is a Gaussian copula with correlation $\rho \in [-1, 1]$. Thus, we have as copula parameter estimate in terms of empirical Kendall's $\hat{\tau}$*

$$\hat{\rho} = \sin\left(\frac{\pi}{2}\hat{\tau}\right). \quad (2.6)$$

Thus, we can fit bivariate copulas to data and estimate the corresponding parameters $\hat{\theta}$ for each arbitrary copula density c . Assume we have M candidate families with pair copula densities c_m and corresponding parameter estimates $\hat{\theta}_m$, $m = 1, \dots, M$. A natural selection of the best fitting copula is then to choose the model m^* such that

$$m^* = \arg \max_{m=1, \dots, M} \sum_{i=1}^n \log\left(c_m\left(u_{i,1}, u_{i,2}, \hat{\theta}_m\right)\right).$$

However, this does not take into account that different families can have different number of parameters. Since ultimately, we want to avoid over-parametrized models, the complexity of a copula model should be taken into account as well as its goodness of fit, expressed by the log-likelihood. To adjust for this, we consider for example the *Akaike information criterion (AIC)* (Akaike, 1973) given by

$$m_{AIC}^* = \arg \min_{m=1, \dots, M} -2\left(\sum_{i=1}^n \log\left(c_m\left(u_{i,1}, u_{i,2}, \hat{\theta}_m\right)\right)\right) + 2 \log(n).$$

In the remainder of the thesis, whenever we optimize over pair copulas for a given pair of copula data, we choose the pair copula family and parameter which minimizes AIC.

This section contained every aspect of copulas, especially bivariate copulas, which we need for the remainder of the thesis. Next, we will continue with *vine copulas* which allow constructions of a multivariate copula only using (conditional) bivariate copulas. By also taking into account marginal distributions, we obtain flexibility to model multivariate distributions.

2.2 Vine Copulas

As we have seen, bivariate copulas are a very flexible model class. However, apart from the elliptical copulas, these families can not be easily extended to arbitrary dimensions. To overcome this, Joe (1996) constructed distributions in d dimensions modelled by $d(d-1)/2$ bivariate distributions. Extending his work, Aas et al. (2009) developed the *pair-copula-construction (PCC)*, which builds up a d -dimensional distribution using d marginals and $d(d-1)/2$ (conditional) bivariate copulas. These building blocks can be chosen entirely

independent from each other and thus provide a very flexible modelling approach. For example, pair copulas with heavy tails or asymmetric dependence can be used. Yet, the construction of a d -dimensional distribution with $d(d-1)/2$ (conditional) pairs is not unique. More precisely, there exists an exponentially growing number of valid constructions, see Kurowicka and Joe (2011, p. 190). With the work of Bedford and Cooke (2001, 2002), introducing *regular vines*, a framework was developed which allowed to organize the possible constructions by *vine trees*. In total, $d-1$ of these trees are required to define a d -dimensional distribution and are given by $\mathcal{V} = (T_1, \dots, T_{d-1})$ such that

- (i) T_1 is a tree with nodes $V_1 = \{1, \dots, d\}$ and edges E_1 ,
- (ii) for $i \geq 2$, T_i is a tree with nodes $V_i = E_{i-1}$ and edges E_i ,
- (iii) if two nodes in T_{i+1} are joined by an edge, the corresponding edges in T_i must share a common node (proximity condition).

By (ii), edges become nodes and are connected with new edges recursively. For a node, e.g. $\{1, 2\} \in V_2$, we define the two nodes $\{1\}, \{2\} \in V_1$ of which the node in V_2 is combined, as *m-children*. For some node in T_k , define the *m-family* as the union of all its m-children and their m-children in trees T_1, \dots, T_{k-1} . Each edge in one of the R-vine trees consists of a bivariate *conditioned set* and a *conditioning set*, ranging from the empty set to a set containing $d-2$ variables. To specify how an edge represents a specific (conditional) pair, let the *complete union* of an edge e be $A_e := \{j \in V_1 | \exists e_1 \in E_1, \dots, e_{i-1} \in E_{i-1} : j \in e_1 \in \dots \in e_{i-1} \in e\}$. The conditioning set of an edge $e = \{a, b\}$ is then given by $D_e := A_a \cap A_b$. The conditioned set is given by $\mathcal{C}_e := \mathcal{C}_{e,a} \cup \mathcal{C}_{e,b}$ with $\mathcal{C}_{e,a} := A_a \setminus D_e$ and $\mathcal{C}_{e,b} := A_b \setminus D_e$. For all edges $e \in E_i$, $1 \leq i \leq d-1$, we define the set of bivariate copula densities by $\mathcal{B}(V) = \{c_{j(e), \ell(e); D(e)} | e \in E_i, 1 \leq i \leq d-1\}$ with the conditioned set $j(e)$, $\ell(e)$ and the conditioning set $D(e)$. Hence, with the PCC, Equation (2.2) can be written as

$$f(x_1, \dots, x_d) = \left(\prod_{i=1}^d f_i(x_i) \right) \times \left(\prod_{i=1}^{d-1} \prod_{e \in E_i} c_{j(e), \ell(e); D(e)} \left(F_{j|D}(x_{j(e)} | \mathbf{x}_{D(e)}), F_{\ell|D}(x_{\ell(e)} | \mathbf{x}_{D(e)}) \right) \right). \quad (2.7)$$

In (2.7), we implicitly took into account the *simplifying assumption*. It imposes that a two-dimensional conditional copula density, e.g.

$$c_{13;2}(F_{1|2}(x_1|x_2), F_{3|2}(x_3|x_2); x_2)$$

is independent of the conditioning value $X_2 = x_2$. A detailed discussion can be found in Stöber et al. (2013). We define the parameters of the bivariate copula densities $\mathcal{B}(V)$ by $\Theta(\mathcal{B}(V))$. This determines the R-vine copula $\mathcal{W} = (V, \mathcal{B}(V), \Theta(\mathcal{B}(V)))$. An intuitive representation of vine copulas is given by lower triangular $d \times d$ matrices, see Dißmann et al. (2013). Such an R-vine matrix $M = (m_{i,j})_{i=1, \dots, d; j=1, \dots, d}$ has to satisfy three properties.

- (i) $\{m_{d,i}, \dots, m_{i,i}\} \subset \{m_{d,j}, \dots, m_{j,j}\}$ for $1 \geq i \geq j \geq d$.
- (ii) $m_{i,i} \notin \{m_{i+1,i+1}, \dots, m_{d,i+1}\}$ for $1 \geq i \geq d - 1$.
- (iii) For all $j = d - 2, \dots, 1, i = j + 1, \dots, d$, there exist (k, ℓ) with $k < j$ and $\ell < k$ such that

$$\begin{aligned} \{m_{i,j}, \{m_{d,j}, \dots, m_{i+1,j}\}\} &= \{m_{k,k}, \{m_{1,k}, \dots, m_{\ell,k}\}\} \text{ or} \\ \{m_{i,j}, \{m_{d,j}, \dots, m_{i+1,j}\}\} &= \{m_{\ell,\ell}, \{m_{1,k}, \dots, m_{\ell-1,k}, m_{k,k}\}\}. \end{aligned} \quad (2.8)$$

The last property reflects the proximity condition. We now give an example R-vine.

Example 2.5 (R-vine in six dimensions). *The R-vine tree sequence in Figure 2.1 is described by the R-vine matrix M as follows. Edges in T_1 are pairs of the main diagonal and the lowest row, e. g. (2,1), (6,2), (3,6). T_2 is given by the main diagonal and the second last row conditioned on the last row, e. g. 6,1|2; 3,2|6. Higher order trees are characterized similarly. M -children of e. g. 6,1|2 are (2,1) and (6,2) and its m -family comprises additionally 1,2,6. For a column p in M , only entries of the main-diagonal right of p , i. e. values in $(m_{p+1,p+1}, \dots, m_{d,d})$ are allowed and no entry occurs more than once in a column.*

$$M = \begin{pmatrix} 4 & & & & & \\ 1 & 5 & & & & \\ 3 & 1 & 3 & & & \\ 6 & 3 & 1 & 6 & & \\ 2 & 6 & 2 & 1 & 2 & \\ 5 & 2 & 6 & 2 & 1 & 1 \end{pmatrix}$$

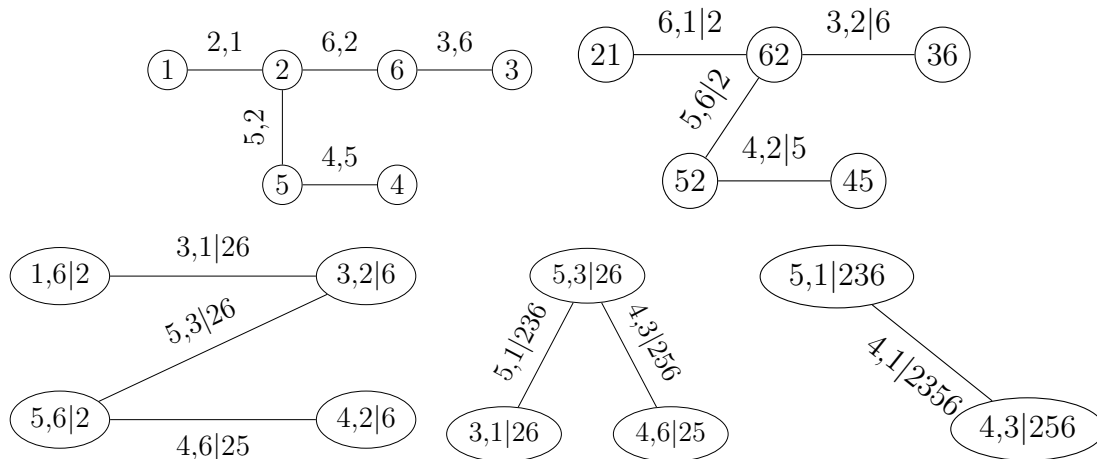


Figure 2.1: Example 2.5: R-vine trees T_1, T_2 (top), T_3, T_4, T_5 (bottom), left to right.

We abbreviate $c_{j,\ell;D} := c_{j,\ell;D}(F_{j|D}(x_j|\mathbf{x}_D), F_{\ell|D}(x_\ell|\mathbf{x}_D))$ for the conditioning vector \mathbf{x}_D . Additionally, let $\mathbf{x} = (x_1, \dots, x_6)$ and $f_i := f_i(x_i)$. The density in (2.7) becomes

$$\begin{aligned} f(\mathbf{x}) &= f_1 \times f_2 \times f_3 \times f_4 \times f_5 \times f_6 \times c_{2,1} \times c_{6,2} \times c_{3,6} \times c_{5,2} \times c_{4,5} \times c_{6,1;2} \times c_{3,2;6} \\ &\quad \times c_{5,6;2} \times c_{4,2;5} \times c_{3,1;26} \times c_{5,3;26} \times c_{4,6;25} \times c_{5,1;236} \times c_{4,3;256} \times c_{4,1;2356}. \end{aligned}$$

The pair copula families and parameters are also described by lower triangular family and parameter matrices $\Gamma = (\gamma_{i,j})_{i=1,\dots,d;j=1,\dots,d}$ and $P = (p_{i,j})_{i=1,\dots,d;j=1,\dots,d}$. Thus, the family and parameters of the edge $6, 1|2$ are given by $\gamma_{5,4}$ and $p_{5,4}$, respectively. When two-parametric pair copulas are considered, we use an additional parameter matrix P_2 .

This concludes our description of R-vine models. We continue on how to assess the goodness of fit of several different R-vine models given data.

2.2.1 Model Assessment

We consider an R-vine model in d dimensions with specification $\mathcal{W} = (\mathcal{V}, \mathcal{B}(\mathcal{V}), \Theta(\mathcal{B}(\mathcal{V})))$. Additionally, assume we have n replications of d -dimensional data $X = (\mathbf{x}_1, \dots, \mathbf{x}_n)^T \in \mathbb{R}^{n \times d}$ with $\mathbf{x}_k \in \mathbb{R}^d$ for $k = 1, \dots, n$. Including the marginal distributions $f_i(x_i)$, $i = 1, \dots, d$, the log-likelihood for the model \mathcal{W} on the x -scale is

$$\mathcal{L}(\mathcal{W}, X) = \sum_{k=1}^n \left(\sum_{i=1}^d \log \left(f_i(x_{k,i}) \right) + \sum_{i=1}^{d-1} \sum_{e \in E_i} \log \left(c_{j(e), \ell(e); D(e)} \left(F_{j|D}(x_{k,j(e)} | \mathbf{x}_{k,D(e)}), F_{\ell|D}(x_{k,\ell(e)} | \mathbf{x}_{k,D(e)}) \right) \right) \right).$$

The log-likelihood will always increase whenever more parameters are included in a model. Thus, in high dimensional setups where the number of significant parameters grows slower than the total number of possible parameters, it is not feasible to use log-likelihood for model discrimination. Because of this, *penalized goodness of fit* measures as the (AIC) and the *Bayesian information criterion (BIC)* (Schwarz, 1978) were developed. For $n \geq 8$, BIC will penalize more than AIC. If the number of possible parameters in an R-vine $q(d) = 2 \times d(d-1)/2$ is greater or equal than the sample size and the model is relatively small, the penalization of BIC will be too weak. For these setups, we use the *modified BIC (mBIC)* of Frommlet et al. (2011) or the *generalized information criterion (GIC)*, see Fan and Tang (2013). More precisely, we have

$$\begin{aligned} AIC(\mathcal{W}, X) &= -2\mathcal{L}(\mathcal{W}, X) + 2p \\ BIC(\mathcal{W}, X) &= -2\mathcal{L}(\mathcal{W}, X) + \log(n)p, \\ mBIC(\mathcal{W}, X) &= -2\mathcal{L}(\mathcal{W}, X) + p \log(nq^2) - 2 \log(p!) \\ &\quad - \sum_{j=1}^p \log(\log(nq^2/j)) \\ GIC(\mathcal{W}, X) &= -2\mathcal{L}(\mathcal{W}, X) + \log(\log(n)) \log(p)p, \end{aligned} \tag{2.9}$$

where p equals the number of parameters in the model Θ . Finally note that the number of parameters corresponds to the the number of one parametric copulas and two times the number of two parametric copulas in the model.

2.2.2 Model Selection

Since the space of all R-vine structures is too large to explore explicitly each model, the standard estimation method relies on using a search heuristic such as the *Dißmann algorithm* (Dißmann et al., 2013). Initially, for each pair $(j, \ell) \in \binom{d}{2}$, Kendall's τ of the pair (U_j, U_ℓ) is calculated. The intuition is that variable pairs with high dependence should contribute significantly to the model fit and should be included in the first trees. Since T_1 must be a tree, the $d - 1$ edges with highest sum of absolute value of Kendall's τ are chosen based on a maximum spanning tree algorithm, e. g. the one developed by Prim (1957). Afterwards, on the selected edges either maximum likelihood estimation for all desired pair copula types is performed or the corresponding copula parameters are estimated by inversion of the empirical Kendall's τ . From these estimators, *pseudo-observations* are generated. More precisely, assume we want to estimate the pair copula density $c_{j(e), \ell(e); D(e)}$. Then, we use the pseudo-observations generated from $\hat{F}_{j|D}(x_{j(e)} | \mathbf{x}_{D(e)})$ and $\hat{F}_{\ell|D}(x_{\ell(e)} | \mathbf{x}_{D(e)})$ to estimate pair copula families and parameters on this pair. The corresponding $\hat{F}_{j|D}$ and $\hat{F}_{\ell|D}$ are explicitly given by derivatives of the pair copula distribution functions in lower trees evaluated at the estimated parameters. After taking into account the proximity condition, Kendall's τ is calculated on all admissible pairs of pseudo-observations and again, a maximum spanning tree is determined. After $d - 1$ iterations, the R-vine structure is determined. This proceeding has some drawbacks. First, it is not ensured that for each tree the maximum spanning tree in terms of Kendall's τ leads to a structure with e. g. optimal log-likelihood or any other penalized goodness of fit measure as in (2.9). Second, as lower order trees influence higher order trees, sub-optimal choices lead to error-propagation. Finally, in each step a tree is fitted over all remaining (conditional) pairs, and clusters with strong within dependence are not treated any different from structures with less dependence. Overall the effort is of order d^2 since $d(d - 1)/2$ pair copulas are estimated.

As mentioned, using Kendall's τ as edge weights is only heuristically backed by the goal to model the strongest dependency first. However, this approach is not ensured to optimize e. g. log-likelihood or AIC in a given tree and thus, also not in the entire R-vine. We can instead also estimate pair copula densities $c_{j,\ell}$ for each edge (j, ℓ) on all $d(d - 1)/2$ edges in the first tree and then calculate a maximum spanning tree with respect to an edge weight $\mu_{j,\ell}$, e. g. log-likelihood or any other penalized goodness of fit measure as in (2.9), based on the actual fit. Thus, this choice would optimize the corresponding goodness of fit measure in a given tree. For higher trees, this can be done similarly. This approach has first been discussed by Czado et al. (2013). However, it was considered having unacceptable computational complexity and hence, computation time. We will however come back to it later for an improved version in Chapter 6.

Another promising proposal has been made by Kraus and Czado (2017), explicitly searching for simplified R-vine structures. One can expect that the true structure of the R-vine can be correctly identified if the data is originating from a simplified R-vine. However, it works similar to Dißmann's algorithm and hence, has the same computational complexity. There have been attempts to relate undirected graphical models to R-vines for structure

selection. Hobæk Haff et al. (2016) showed that a k -truncated R-vine can be expressed as chordal graph with maximum clique size $k + 1$ and vice versa. However, this chordal graph needs to adhere to some other non-trivial properties. These are in practice not met when a graphical model is fitted. Thus, finding a sparse undirected graph and translating it into a sparse R-vine is hard and computationally infeasible. Kovács and Szántai (2016) propose an algorithm to calculate a k -truncated R-vine from a chordal graph with maximum clique size $k + 1$ and show that taking into account the aforementioned property only leads to a chordal graph with maximum clique size $k + 2$. The later can then be used to estimate a $k + 2$ truncated R-vine. However, the problem is only deferred. Where there exist several methods for finding sparse undirected graphical models in high dimensions, these will not be chordal in most cases. Finding a so called *chordal cover* with clique size at most $k + 1$ is known to be *NP-complete*, see Arnborg and Corneil (1987). Hence, for dimensions $d > 1000$ where k can not be assumed too small, the problem is also intractable.

Finally, there also exist Bayesian methods for R-vine estimation (Gruber and Czado, 2015b,a). However, these do require even more computational effort and are hence not feasible in more than $d > 20$ dimensions.

2.2.3 Model Simplification

All of the proposed approaches may be modified by testing each pair copula for the independence copula with density $c^\perp(u_1, u_2) = 1$. Thus, a type-1 error $\alpha \in (0, 1)$ is specified and each pair-copula is tested for the null hypothesis to be the independence copula. Only if this hypothesis can be rejected at the level $1 - \alpha$, an estimation of pair copula family and parameter is performed. Similarly, entire R-vine trees can be tested for only containing the independence copula. However, this also requires that an additional tree is fitted before it can be tested. Contrary to this, a *truncation level* $k \in \{1, \dots, d - 2\}$ can be specified upfront. By doing so, only the first k R-vine trees are estimated and independence is assumed for all higher trees. If a k -truncation is imposed, (2.7) becomes

$$f(x_1, \dots, x_d) = \left(\prod_{i=1}^d f_i(x_i) \right) \times \left(\prod_{i=1}^k \prod_{e \in E_i} c_{j(e), \ell(e); D(e)} \left(F_{j|D}(x_{j(e)} | \mathbf{x}_{D(e)}), F_{\ell|D}(x_{\ell(e)} | \mathbf{x}_{D(e)}) \right) \right).$$

In Example 2.5, a k -truncated R-vine is given by

$$c_{j(e), \ell(e); D(e)} = c^\perp = 1 \text{ if } |D(e)| \geq k.$$

More details are given in the work of Brechmann et al. (2012). Finally, testing for independence does not explicitly decrease the computational effort. However, a k -truncation leads to estimation of only about kd pair copulas. Yet, identifying a sensible truncation level k prior to estimation is hard and not intuitively clear.

Chapter 3

Mathematical Foundations and Sparse Modelling

We will now introduce the frameworks and concepts that we are going to use to estimate high dimensional vine copula models in the next chapters of the thesis. First, we briefly describe basic mathematical concepts as univariate *time series models* and *conditional independence* of random vectors in Section 3.1. Additionally, we recall properties of the multivariate Gaussian distribution as we will make use of well-established Gaussian methods. Another important aspect we exploit in our work is the the well-known *Lasso*, which we will use for structure selection in Chapter 5. We continue with *probabilistic graphical models* in Section 3.3, which will be extensively used in the Chapters 4 and 6. Finally, we introduce a similar concept, *structural equation models (SEMs)* in Section 3.4, required in Chapters 4 and 5.

Parts of this chapter are slightly adapted from previously submitted manuscripts. Conditional independencies (Section 3.1.2) as well as the introduction to (directed) graphical models, Section 3.3, are described in Müller and Czado (2017b). Section 3.2 covering the Lasso and Section 3.4 introducing structural equation models are based on Müller and Czado (2017c). Parts of the introduction to the multivariate Gaussian distribution (3.1.3) and undirected graphical models in Sections 3.3.2 and 3.3.3 are based on Müller and Czado (2017a).

3.1 Mathematical Foundations

As explained in Section 2.1, a necessary prerequisite for working with copulas is the adequate modelling of the marginal distribution functions. Because of the availability of this data, we will consider financial data sets, i. e. stock data in the remainder of the thesis. For these, there exists a rich literature of models to describe the univariate behaviour. We will briefly introduce the corresponding *time series models* and give references. Next, we recapitulate properties of conditional independence in multivariate random vectors, and finally, the multivariate Gaussian distribution which will be vital for later estimation of graphical models.

3.1.1 Time Series Models

All of the data we will consider in this thesis are financial *time series*. Thus, before we can model the corresponding dependence behaviour, we have to take into account the marginal distributions, see Section 2.1.4. Since we want to model i.i.d. data, we need to remove trends and serial dependence off the corresponding time series. This will be done with the time series models we introduce now. A standard reference for these models is Brockwell and Davis (2016). When we speak of time series, we understand a sequence of random variables $(X_t)_{t \in T}$, i. e. a *stochastic process* with some *time domain* $T = \mathbb{Z}$ in our case. A describing feature of a time series is its *auto-covariance function*. Let $(X_t)_{t \in T}$ be a time series such that $\mathbb{E}(X_t^2) < \infty$ for all $t \in T$. The auto-covariance function γ is given by

$$\gamma(s, t) = \text{Cov}(X_s, X_t) \text{ for } s, t \in T.$$

Clearly, a desirable property of a time series is for γ to depend only on the distance between the time points $|t - s|$ rather than on a specific value of s and t . This is the concept of *stationarity*, which is a regularity assumption for time series. More precisely, we say $(X_t)_{t \in T}$ is a stationary time series if

- (i) $\mathbb{E}(X_t^2) < \infty$ for all $t \in T$,
- (ii) $\mathbb{E}(X_t) = \text{const}$ for all $t \in T$,
- (iii) $\gamma(s, t) = \gamma(r + s, r + t)$ for all $r, s, t \in T$.

A very simple stationary process is a *white-noise* process, which is a time series equivalent to a random error. More precisely, let $(X_t)_{t \in T}$ with $\mathbb{E}(X_t) = 0$ for all $t \in T$ and auto-covariance function $\gamma(s - t) = \sigma^2 \mathbf{1}_{\{s-t=0\}}$ for some $\sigma > 0$. This means, the process does not contain any auto-covariance and hence, observing today does not give information about any future time points of the process. We denote it as $X \sim WN(0, \sigma^2)$. Using this simple process, we can define more elaborate classes of time series models. Those are the *autoregressive (AR)*, the *moving average (MA)* and a composition of both, the *autoregressive moving average (ARMA)* process, see (Brockwell and Davis, 2016, Chapters 2,3).

Definition 3.1 (AR(p) process). *A time series $(X_t)_{t \in T}$ is called AR(p) process if X_t is stationary and if for every t*

$$X_t = \sum_{i=1}^p \phi_i X_{t-i} + Z_t$$

with $p \in \mathbb{N} \cup \{0\}$ and $\phi_1, \dots, \phi_p \in \mathbb{R}$ and a white noise process $(Z_t)_{t \in T}$.

As we see, the model assumes that X_t can be described as a weighted sum of its previous values and a white noise error term.

Definition 3.2 (MA(q) process). *A time series $(X_t)_{t \in T}$ is called MA(q) process if X_t is stationary and if for every t*

$$X_t = \sum_{i=1}^q \theta_j Z_{t-i} + Z_t,$$

$q \in \mathbb{N} \cup \{0\}$, $\theta_1, \dots, \theta_q \in \mathbb{R}$ and a white noise process $(Z_t)_{t \in T}$.

Contrary to the autoregressive model, the moving average process does not take into account previous values of the time series X_t but a weighted sum of white noise error terms. These both can be combined into one joint model, the *ARMA* process.

Definition 3.3 (ARMA(p, q) process). *A time series $(X_t)_{t \in T}$ is called ARMA(p, q) process if X_t is stationary and if for every t*

$$X_t = \sum_{i=1}^p \phi_i X_{t-i} + \sum_{i=1}^q \theta_j Z_{t-i} + Z_t$$

$p, q \in \mathbb{N} \cup \{0\}$, $\phi_1, \dots, \phi_p, \theta_1, \dots, \theta_q \in \mathbb{R}$ and a white noise process $(Z_t)_{t \in T}$. If $(X_t - \mu)_{t \in T}$ for $\mu \in \mathbb{R}$ is an ARMA(p, q) process, $(X_t)_{t \in T}$ is an ARMA(p, q) process with mean μ .

These time series models have proven very important in modelling financial data. However, the assumption that the error term follows a simple white noise with constant variance σ^2 may be too restrictive. Especially in times of market distress, increased volatility is observed and it is vital to account for this. To overcome this, Bollerslev (1986) introduced the *GARCH* model, allowing for more elaborate modelling of the volatility.

Definition 3.4 (GARCH(p, q) process). *Let $(\epsilon_t)_{t \in T}$ be a stochastic process and Ψ_t the information set (sigma algebra) containing the information until time t . Then, $(\epsilon_t)_{t \in T}$ is called a GARCH (p, q) process if for every t*

$$\begin{aligned} \epsilon_t | \Psi_t &\sim \mathcal{N}(0, \sigma_t^2), \\ \sigma_t^2 &= \omega + \sum_{i=1}^q \alpha_i \epsilon_{t-i}^2 + \sum_{i=1}^p \beta_i \sigma_{t-i}^2, \end{aligned}$$

where

$$\begin{aligned} p &\in \mathbb{N} \cup \{0\} & q &\in \mathbb{N}, \\ \omega &> 0, & \alpha_i &\geq 0, \quad i = 1, \dots, q, \\ & & \beta_i &\geq 0, \quad i = 1, \dots, p. \end{aligned}$$

Thus, in the GARCH model, the volatility is not constant but is a weighted sum of an intercept, previously realized variances σ_t^2 and a stochastic process ϵ_t . The ARMA and the GARCH process can now be combined to the very flexible ARMA-GARCH process. From Definition 3.4 we see that the errors are assumed to have a Gaussian distribution. Especially for financial applications, this assumption may be too strong, since e. g. a sharp decrease in a stock price can not be explained by a Gaussian distribution which has no fat tails. This leads to a generalization using the Student's-t distribution, see Bollerslev (1987) or a skewed Student's-t distribution, see Hansen (1994). Fortunately, these models are also easy to fit using computational methods.

3.1.2 Conditional Independence

Our task is to simplify high-dimensional vine copula models. Since an R-vine models conditional distributions, a logical step is to exploit conditional independence to set many pair copulas to the independence copula, making the expression for the joint distribution much more parsimonious. On the other hand side, graphical models as we will introduce later on are models which enable us to identify *conditional independencies* using graph theoretic arguments and to exploit these. As we need to work with these models extensively, we need some calculation rules for conditional independencies in random vectors. We assume throughout the thesis that all occurring densities are strictly positive, which eases considerations. As we work on conditional independencies induced by models with underlying multivariate Gaussian distribution, this assumption can be made without any restrictions. First, we define conditional independence, see Lauritzen (1996, p. 29).

Definition 3.5 (Conditional independence). *Consider three random variables X, Y, Z with corresponding probability density functions f_X, f_Y, f_Z and assume throughout that all densities are strictly positive. We say that X is conditionally independent of Y given Z , denoted*

$$X \perp\!\!\!\perp Y \mid Z,$$

if and only if

$$f_{X,Y|Z}(x,y|z) = f_{X|Z}(x|z) f_{Y|Z}(y|z).$$

This means that knowing Z renders X and Y independent. There exists a number of results involving conditional independence. We will use the following from Whittaker (1990, p. 33) in Chapter 4 repeatedly.

Proposition 3.6 (Calculation rules for conditional independencies). *If $(\mathbf{X}, \mathbf{Y}, \mathbf{Z}_1, \mathbf{Z}_2)$ is a partitioned random vector with joint density $f_{X,Y,Z_1,Z_2} > 0$, then the following expressions are equivalent:*

$$(i) \mathbf{Y} \perp\!\!\!\perp (\mathbf{Z}_1, \mathbf{Z}_2) \mid \mathbf{X},$$

$$(ii) \mathbf{Y} \perp\!\!\!\perp \mathbf{Z}_2 \mid (\mathbf{X}, \mathbf{Z}_1) \text{ and } \mathbf{Y} \perp\!\!\!\perp \mathbf{Z}_1 \mid \mathbf{X}.$$

Proof. To show (i) \Rightarrow (ii), we refer to the block independence Lemma in Whittaker (1990, p. 33) for the first assertion. To show the second part of (ii) with $\mathbf{Z}_2 \in \mathbb{R}^p$ we have

$$\begin{aligned} f_{\mathbf{Y},\mathbf{Z}_1|\mathbf{X}}(\mathbf{y}, \mathbf{z}_1; \mathbf{x}) &= \int_{\mathbb{R}^p} f_{\mathbf{Y},\mathbf{Z}_1,\mathbf{Z}_2|\mathbf{X}}(\mathbf{y}, \mathbf{z}_1, \mathbf{z}_2; \mathbf{x}) d\mathbf{z}_2 \\ &= \int_{\mathbb{R}^p} f_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}; \mathbf{x}) f_{\mathbf{Z}_1,\mathbf{Z}_2|\mathbf{X}}(\mathbf{z}_1, \mathbf{z}_2; \mathbf{x}) d\mathbf{z}_2 \\ &= f_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}; \mathbf{x}) \int_{\mathbb{R}^p} f_{\mathbf{Z}_1,\mathbf{Z}_2|\mathbf{X}}(\mathbf{z}_1, \mathbf{z}_2; \mathbf{x}) d\mathbf{z}_2 = f_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}; \mathbf{x}) f_{\mathbf{Z}_1|\mathbf{X}}(\mathbf{z}_1; \mathbf{x}), \end{aligned}$$

using (i) in the second line. Direction (ii) \Rightarrow (i) is given in Whittaker (1990, p. 35). \square

This result allows us to draw several conclusions from conditional independence statements.

3.1.3 Properties of the Multivariate Gaussian Distribution

We will briefly review the most important properties of the multivariate Gaussian distribution which we will use later on. Recall that a random vector \mathbf{X} obeys a multivariate Gaussian distribution in d dimensions with mean vector $\boldsymbol{\mu}$ and covariance matrix Σ , denote $\mathbf{X} \sim \mathcal{N}_d(\boldsymbol{\mu}, \Sigma)$ if and only if its density can be written as

$$f(\mathbf{x}; \boldsymbol{\mu}, \Sigma) = (2\pi)^{-\frac{d}{2}} \det(\Sigma)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu})\right). \quad (3.1)$$

Here, Σ is a $d \times d$ positive definite matrix. For our considerations we assume throughout the thesis $\boldsymbol{\mu} = \mathbf{0}$. A very useful property of the multivariate Gaussian distribution is that for a vector $\mathbf{X} \sim \mathcal{N}_d(\mathbf{0}, \Sigma)$, we can draw inference about conditional independence in this random vector from the inverse of the covariance matrix $\Omega = \Sigma^{-1}$. More precisely, let $j, \ell \in \{1, \dots, d\}$ with $j \neq \ell$. Then,

$$\Omega_{j\ell} = 0 \Leftrightarrow X_j \perp\!\!\!\perp X_\ell \mid \{X_1, \dots, X_d\} \setminus \{X_j, X_\ell\}, \quad (3.2)$$

see Whittaker (1990, p. 164). Thus, conditional independence can be read directly off the inverse covariance matrix. Note that also the corresponding partial correlations, see (2.3) are related to the conditional correlations for the multivariate Gaussian distribution. More precisely, let $\mathbf{X} \sim \mathcal{N}_d(\mathbf{0}, \Sigma)$, $j, \ell \in \{1, \dots, d\}$ with $j \neq \ell$ and $D \subseteq \{1, \dots, d\} \setminus \{j, \ell\}$, then

$$\rho_{X_j, X_\ell; \mathbf{X}_D} = \text{Cov}(X_j, X_\ell | \mathbf{X}_D), \quad (3.3)$$

as shown by Baba et al. (2004). Thus, we also have that conditional independence is equivalent to zero partial correlation for a multivariate Gaussian distribution.

If we consider two d -dimensional Gaussian distributions with densities $f(\mathbf{x}; \mathbf{0}, \Sigma_1)$ and $f(\mathbf{x}; \mathbf{0}, \Sigma_2)$, a natural task is to compare these two distributions. A well known method is the *Kullback-Leibler information divergence*, see Kullback and Leibler (1951). Let us assume two probability distributions P and Q with corresponding real valued densities p and q . Then, the Kullback-Leibler information divergence is defined by

$$\mathcal{D}(P, Q) = \int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right) dx.$$

Where this integral may be hard to evaluate in practical for arbitrary distributions, for two multivariate Gaussian distributions it is a closed form expression. More precisely, we have in terms of the corresponding covariance matrices Σ_1 and Σ_2 that

$$\mathcal{D}(\Sigma_1, \Sigma_2) = \frac{1}{2} \text{tr}(\Sigma_1 \Sigma_2^{-1}) - \frac{1}{2} \log(\det(\Sigma_1 \Sigma_2^{-1})) - \frac{d}{2}, \quad (3.4)$$

see Whittaker (1990, p. 167). Finally, note that $\mathcal{D}(P, Q) \geq 0$ and $\mathcal{D}(P, Q) = 0$ if and only if $P = Q$.

3.2 The Lasso in Linear Regression

As sketched in Section 2.2, we have to solve a high dimensional model selection problem for finding parsimonious vine copulas in high dimensions. Among conditional independencies in multivariate Gaussian distributions, we also use model selection approaches for univariate problems to approach this goal. A well known method for model selection in the regression domain is the *Lasso*, as acronym for *Least Absolute Shrinkage and Selection Operator*, which we will introduce now, and apply later on in Chapter 5.

In the most general case, consider a sample of n observations $\{\mathbf{x}_k, y_k\}$, $k = 1, \dots, n$, where $\mathbf{x}_k = (x_{k,1}, \dots, x_{k,p}) \in \mathbb{R}^p$. We want to approximate y_k given a set of linear predictors $x_{k,j}$

$$y_k = \varphi_0 + \sum_{\ell=1}^p \varphi_\ell x_{k,\ell},$$

with unknown regression coefficients φ_0 and $\boldsymbol{\varphi} = (\varphi_1, \dots, \varphi_p)$. This is most often solved by minimizing the quadratic error with respect to φ_0 and $\boldsymbol{\varphi} = (\varphi_1, \dots, \varphi_p)$:

$$\min_{(\varphi_0, \boldsymbol{\varphi}) \in \mathbb{R}^{p+1}} \left(\frac{1}{2n} \sum_{k=1}^n \left(y_k - \varphi_0 - \sum_{\ell=1}^p \varphi_\ell x_{k,\ell} \right)^2 \right). \quad (3.5)$$

The solution to this optimization problem often contains many coefficients $\varphi_\ell \neq 0$, $\ell = 1, \dots, p$. Thus, for p large, the model becomes overly parametrized and hard to interpret. Yet, solving (3.5) under the additional constraint

$$\sum_{\ell=1}^p |\varphi_\ell| \leq t, \quad t \geq 0, \quad (3.6)$$

yields a parsimonious model. This *regularization technique* is called the *Lasso* and since its invention by Tibshirani (1994), proved very useful in many applications. By shrinking coefficients exactly to zero, it combines both parameter estimation and model selection in one step. It also works in cases where $p > n$, which are hard to solve otherwise. The Lasso is hence the method of choice when dealing with many possible predictors, of which only some contribute significantly to the model fit. For convenience, we will consider the following Lagrangian form of the optimization problem (3.7), which is equivalent to (3.5) under the constraint (3.6):

$$\min_{(\varphi_0, \boldsymbol{\varphi}) \in \mathbb{R}^{p+1}} \left(\frac{1}{2n} \sum_{k=1}^n \left(y_k - \varphi_0 - \sum_{\ell=1}^p \varphi_\ell x_{k,\ell} \right)^2 + \lambda \sum_{\ell=1}^p |\varphi_\ell| \right), \quad (3.7)$$

for some $\lambda \geq 0$. One can show that a solution $(\hat{\varphi}_0^\lambda, \hat{\boldsymbol{\varphi}}^\lambda)$ of (3.7) minimizes the problem in (3.5) under the condition (3.6) with $t = |\hat{\boldsymbol{\varphi}}^\lambda| = \sum_{\ell=1}^p |\hat{\varphi}_\ell^\lambda|$ (Hastie et al., 2015). We do not include an intercept in our considerations and thus set $\varphi_0 \equiv 0$ for the remainder of the paper. If we consider the problem (3.7) and set $\lambda = \infty$, all coefficients $\hat{\varphi}_\ell$, $\ell = 1, \dots, p$ will be set to zero because of the penalization. Decreasing $\lambda > 0$, more and more coefficients

become non-zero. This relationship between $\lambda > 0$ and $\hat{\varphi}_\ell^\lambda$, $\ell = 1, \dots, p$ is called the *regularization path*. We formalize it by a set $\Lambda(\lambda)$ such that

$$\Lambda(\lambda) = \{\ell : \hat{\varphi}_\ell^\lambda \neq 0 \text{ in } \hat{\varphi}_\lambda\}.$$

Thus, for each $\lambda > 0$ we are given the non-zero regression coefficients. How to choose $\lambda > 0$ is not obvious. Most often, *k-fold cross-validation* is employed. We divide the total data set of n observations into $m > 1$ randomly chosen subsets M_1, \dots, M_m such that $\bigcup_{j=1}^m M_j = n$. We obtain m training data sets $S_{training}^j = \{1, \dots, n\} \setminus M_j$ and corresponding test data sets $S_{test}^j = M_j$, $j = 1, \dots, m$. Then, the coefficient vector $\hat{\varphi}_r = (\hat{\varphi}_1^r, \dots, \hat{\varphi}_p^r) \in \mathbb{R}^p$ is estimated for various λ_r , for $r = 1, \dots, R$ on each of the m training sets $S_{training}$. Now we use these R coefficient vectors to perform for each test data set S_{test} an *out of sample prediction* of the values

$$\hat{y}_k^r = \sum_{\ell=1}^p \hat{\varphi}_\ell^r x_{k,\ell}, \quad k \in M_j, \quad j = 1, \dots, m, \quad r = 1, \dots, R.$$

For these values, we also know the true values y_k , $k \in M_j$, $j = 1, \dots, m$. Thus, we can calculate the *mean squared prediction error* over all training and test data sets:

$$\delta_j^r = \frac{1}{|M_j|} \sum_{k \in M_j} (y_k - \hat{y}_k^r)^2, \quad j = 1, \dots, m.$$

Since we have m pairs of training and test data, we obtain an estimate for the prediction error for each of the R values of λ_r for $r = 1, \dots, R$ by averaging:

$$\Delta_r = \frac{1}{m} \sum_{j=1}^m \delta_j^r, \quad r = 1, \dots, R.$$

Next, consider the dependence between λ_r , $r = 1, \dots, R$ and the corresponding error Δ_r . A natural choice is to select

$$\lambda_{min}^{CV} = \arg \min_{r=1, \dots, R} \Delta_r.$$

Alternatively, we choose λ_ℓ such that it is at least in within one-standard error of the minimum, denote λ_{1se}^{CV} . For both types of cross validation methods, see Friedman et al. (2010) or the work by Hastie et al. (2015, p. 13). We conclude with a brief example, introducing the concept of regularization paths.

Example 3.7 (Lasso, regularization path, cross validation). *We use the `worldindices` data set, included in the `CDVine` package, see Brechmann and Schepsmeier (2013) comprising $d = 6$ variables with $n = 396$ observations on the u -scale. More precisely, these are the stocks indices \hat{GSPC} , $\hat{N225}$, \hat{SSEC} , \hat{GDAXI} , \hat{FCHI} , \hat{FTSE} of the US, Japanese, Chinese, German, French and British stock markets. We transform our observations to the z -scale using the Gaussian quantile function and denote them by Z_i , $i = 1, \dots, 6$ where $Z_1 \equiv \hat{GSPC}$, $Z_2 \equiv \hat{N225}$, and so on. Let us assume that we want to model the in-*

denote $Z_3 = \hat{SSEC}$ by the regressors \hat{GSPC} , \hat{GDAXI} , \hat{FCHI} , \hat{FTSE} , Z_1 , Z_4 , Z_5 , Z_6 , respectively. We write the regression equation

$$Z_{k,3} = \varphi_0 + \sum_{\ell=1,4,5,6} \varphi_\ell Z_{k,\ell}, \quad k = 1, \dots, n,$$

with unknown regression coefficients φ_0 and $\boldsymbol{\varphi} = (\varphi_1, \dots, \varphi_p)$. We set $\varphi_0 \equiv 0$ and want to solve the regression problem with the Lasso. Thus we obtain the optimization problem

$$\min_{\boldsymbol{\varphi} \in \mathbb{R}^4} \left(\frac{1}{2n} \sum_{k=1}^n \left(Z_{k,3} - \sum_{\ell=1,4,5,6} \varphi_\ell Z_{k,\ell} \right)^2 + \lambda \sum_{\ell=1,4,5,6} |\varphi_\ell| \right). \quad (3.8)$$

The solution to this optimization problem is a regularization path, either along $\lambda > 0$ or $\sum_{i=1,4,5,6} |\hat{\varphi}_i|$, i. e. the L_1 norm of the regression vector. We use the R-package `glmnet` of Friedman et al. (2010) to calculate the regularization paths with respect to the L_1 norm, see Figure 3.1, left panel, and $\log(\lambda)$, see Figure 3.1, right panel.

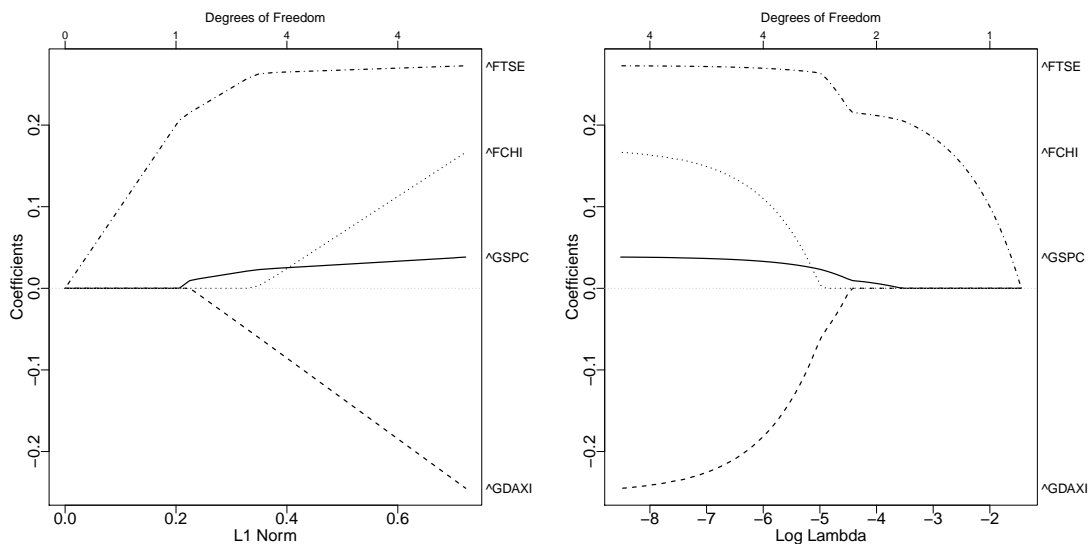


Figure 3.1: Example 3.7: Regularization path of the coefficients with respect to the L_1 norm of coefficients (left) and $\log(\lambda)$.

We see that $Z_6 \equiv \hat{FTSE}$ is the first non-zero coefficient along the regularization path. Additionally, we obtain that coefficients can of course also be negative and the regularization paths of different regressors may intersect. We denote the path by $\Lambda(0) = \{6, 1, 4, 5\}$. Note that $\log(\lambda) < 0$ must not necessarily be the case as in this example. Above the plot, the corresponding number of non-zero parameters is indicated.

3.3 Graphical Models

As mentioned previously, we want to exploit conditional independence to reduce complexity in high dimensional vine copula models. This conditional independence, or *sparsity*

is the key component of another prominent class of models, so called *graphical models*. They are obtained by combining graph theory and statistical models and have proven as very useful visual display of information obtained from data. Among the most popular sources are the books by Whittaker (1990), Lauritzen (1996) and Koller and Friedman (2009). We will first introduce the *graph theory* which is necessary to describe the different concepts. Next, we introduce two major classes of different graphical models. The first uses *undirected graphs* to describe (conditional) independencies in the data and the second uses *directed graphs*, more precisely *directed acyclic graphs*. We will introduce both the different models and standard algorithms developed to estimate these models. The models itself will be extensively used in the Chapters 4 and 6.

3.3.1 Graph Theory

The necessary graph theory is taken from Lauritzen (1996, pp. 4-7). Let $V \neq \emptyset$ be a finite set, the *node set* and let $E \subseteq \{(v, w) \mid (v, w) \in V \times V \text{ with } v \neq w\}$ be the *edge set*, to define a graph $\mathcal{G} = (V, E)$ as a pair of node set and edge set. An edge (v, w) is undirected if $(v, w) \in E \Rightarrow (w, v) \in E$, and (v, w) is *directed* if $(v, w) \in E \Rightarrow (w, v) \notin E$. A directed edge (v, w) is called an *arrow* and denoted by $v \rightarrow w$ with v the *tail* and w the *head*. An arrow between v and w without specifying the orientation is denoted by $v \leftrightarrow w$ and no arrow between v and w regardless of orientation is denoted by $v \nleftrightarrow w$. An *undirected* graph contains only undirected edges and a *directed* graph only directed edges. Replacing all arrows in a directed graph \mathcal{G} by undirected edges, we obtain the *skeleton* \mathcal{G}^s of \mathcal{G} . If a graph only contains undirected edges, it is an *undirected* graph and if it contains only directed edges, it is a *directed* graph. We will not consider graphs with both directed and undirected edges. A weighted graph is defined by a *weight function* $\mu : E \rightarrow \mathbb{R}$. Define a *path* of length k from nodes α to β by a sequence of distinct nodes $\alpha = \alpha_0, \dots, \alpha_k = \beta$ such that $(\alpha_{i-1}, \alpha_i) \in E$ for $i = 1, \dots, k$. This applies to both undirected and directed graphs. A *cycle* is defined as a path with $\alpha = \beta$ and graph without cycles is called *acyclic*. In a directed graph, a *chain* of length k from α to β is a sequence of distinct nodes $\alpha = \alpha_0, \dots, \alpha_k = \beta$ with $\alpha_{i-1} \rightarrow \alpha_i$ or $\alpha_i \rightarrow \alpha_{i-1}$ for $i = 1, \dots, k$. Thus, a directed graph may contain a chain from α to β but no path from α to β , see Example 3.8.

Example 3.8 (Paths and chains in directed graphs). *Consider the following two directed graphs \mathcal{G}_1 and \mathcal{G}_2 .*

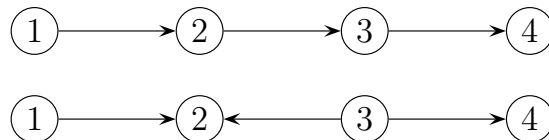


Figure 3.2: Example graphs \mathcal{G}_1 (upper panel) and \mathcal{G}_2 (lower panel).

For each of the two graphs, we consider the question whether a path or chain from 1 to 4 exists. In \mathcal{G}_1 , clearly a path from 1 to 4 along 2 and 3 exists. Additionally, also a

chain from 1 to 4 exists, as well as a chain from 4 to 1 since for the existence of a chain, the edge orientation is not relevant. With the same argument, there exists a chain in \mathcal{G}_2 between 1 and 4. However, no path between 1 and 4 exists as there is no edge $2 \rightarrow 3$.

A graph $\mathcal{H} = (W, F)$ is a *subgraph* of $\mathcal{G} = (V, E)$ if $W \subseteq V$ and $F \subseteq E$. We speak of an *induced subgraph* $\mathcal{H} = (W, F)$ if

$$W \subseteq V \text{ and } F = \{(v, w) \mid (v, w) \in W \times W \text{ with } v \neq w\} \cap E,$$

i. e. \mathcal{H} contains a subset of nodes of \mathcal{G} and all the edges of \mathcal{G} between these nodes, see Example 3.9.

Example 3.9 (Subgraphs and induced subgraphs). *Consider the following three graphs where \mathcal{G}_2 and \mathcal{G}_3 are subgraphs of \mathcal{G}_1 . \mathcal{G}_3 is an induced subgraph of \mathcal{G}_1 , whereas \mathcal{G}_2 is not since the edges $(2, 3)$ and $(1, 5)$ are present in \mathcal{G}_1 on the subset of nodes $\{1, 2, 3, 5\}$ but are missing in \mathcal{G}_2 .*

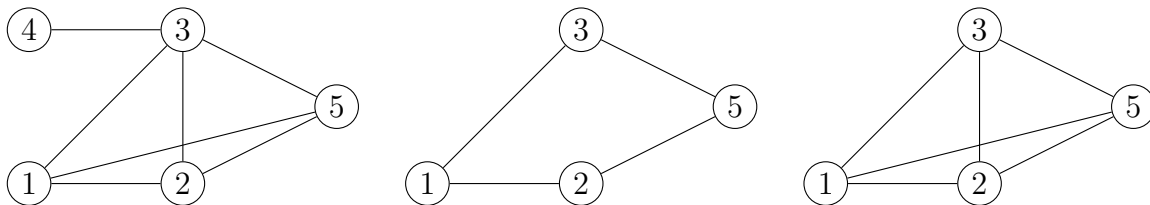


Figure 3.3: Example 3.9: Graphs $\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3$, from left to right.

If $\mathcal{G} = (V, E)$ is undirected and a path from v to w exists for all $v, w \in V$, we say that \mathcal{G} is *connected* and if $\mathcal{G} = (V, E)$ is directed we say that \mathcal{G} is weakly connected if a path from v to w exists for all $v, w \in V$ in the skeleton \mathcal{G}^s of \mathcal{G} . If an undirected graph is connected and acyclic, i. e. has no cycles, it is a *tree* and has $d - 1$ edges on d nodes. Whenever for a graph $\mathcal{G} = (V, E)$ we have that there exists a disjoint partition of $V = \bigcup_{i=1}^p V_i$ such that the p subgraphs \mathcal{H}_i induced by V_i for $i = 1, \dots, p$ are connected subgraphs, we speak of *connected components* of \mathcal{G} . For \mathcal{G} undirected and $\alpha, \beta \in V$, a set $S \subseteq V$ is said to be an (α, β) *separator* in \mathcal{G} if all paths from α to β intersect S . S is said to *separate* A from B in \mathcal{G} if it is an (α, β) separator in \mathcal{G} for every $\alpha \in A, \beta \in B$ and we denote it by $\alpha \perp \beta \mid S$ [\mathcal{G}] and $A \perp B \mid S$ [\mathcal{G}], respectively, see Example 3.10

Example 3.10 (Graphical separation). *Consider the graph \mathcal{G} in Figure 3.4.*

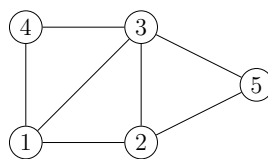


Figure 3.4: Example 3.10: Undirected graph \mathcal{G} in five dimensions.

By virtue of the graphical separation, we have for example $1, 4 \perp 5 \mid 2, 3$ [\mathcal{G}].

We will now combine graph theory and probability theory which together form *probabilistic graphical models*. We start with undirected models and then continue to models on directed acyclic graphs.

3.3.2 Undirected Graphical Models

We consider a random vector $\mathbf{X} = (X_1, \dots, X_d)$ and assume $\mathbf{X} \sim \mathcal{N}_d(\mathbf{0}, \Sigma)$ throughout the rest of the section. Let $\mathcal{G} = (V, E)$ be an undirected graph on d nodes $V = \{1, \dots, d\}$. We say that \mathbf{X} is *Markov* with respect to \mathcal{G} , if

$$X_j \perp\!\!\!\perp X_\ell \mid \{X_1, \dots, X_d\} \setminus \{X_j, X_\ell\} \Leftrightarrow (j, \ell) \notin E. \quad (3.9)$$

This is the definition of an undirected graphical model \mathcal{G} for the distribution of \mathbf{X} . More precisely, each missing edge in the graph \mathcal{G} corresponds to random variables which are conditionally independent given the rest. For the multivariate Gaussian distribution we assumed, this is also directly expressed via the inverse of the correlation matrix, as we have seen in (3.2). More precisely, recall that for $\Omega = \Sigma^{-1}$ we have

$$\Omega_{j,\ell} = 0 \Leftrightarrow X_j \perp\!\!\!\perp X_\ell \mid \{X_1, \dots, X_d\} \setminus \{X_j, X_\ell\}.$$

Thus, estimating the *sparsity pattern* of Ω is equivalent to estimation of the graph \mathcal{G} . From the graph however, we can also extract more information. If the underlying distribution is Gaussian, we have the *global Markov property*, see Lauritzen (1996, Chapter 6). It means that for $S \subseteq \{1, \dots, d\} \setminus \{j, \ell\}$ we have that

$$X_j \perp\!\!\!\perp X_\ell \mid X_S \Leftrightarrow j \perp \ell \mid S [\mathcal{G}], \quad (3.10)$$

which is a more favourable property than (3.9) and which we will exploit later. The following example showcases how conditional independencies can be learned from a graph.

Example 3.11 (Graphical model). *Consider the graph \mathcal{G} in Figure 3.5.*

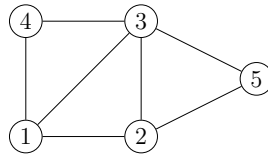


Figure 3.5: Example 3.11: Graphical model in five dimensions.

By virtue of the graphical separation, we have $1, 4 \perp 5 \mid 2, 3 [\mathcal{G}] \Rightarrow 1, 4 \perp\!\!\!\perp 5 \mid 2, 3$.

Of course, before a graphical model, i.e. a graph can be used to draw conditional independencies from it, we have to determine how the graph looks for an underlying data set. Thus, it has to be estimated using numerical methods. We give a quick review and introduce one of the most well known methods to solve this estimation problem, the *graphical Lasso*.

3.3.3 Estimation of Undirected Graphical Models

The first work in this direction was undertaken by Dempster (1972) where it was called *covariance selection* and an algorithmic implementation was given. In recent years, attention went to algorithms which allow to estimate undirected graphical models in very high dimensional setups, i. e. $d > 100$. Furthermore, these problems are often accompanied by the fact that $n < d$, i. e. the sample size falls below the number of dimension, which leads to additional challenges with respect to estimation of such a graphical model. As we have seen, it is equivalent to estimate the sparsity pattern of the precision matrix or the graph. A method to estimate the graph directly is the approach of Meinshausen and Bühlmann (2006), which is called *neighbourhood selection*. The proposal is to solve a linear regression for each node onto all other nodes, but subject to a $\|\cdot\|_1$ penalty, i. e. the Lasso, see Section 3.2. The authors show that neighbourhood selection consistently estimates the graph \mathcal{G} . Another approach is to maximize the log-likelihood with respect to the precision matrix directly with a $\|\cdot\|_1$ penalty, to enforce zero entries in the estimator for the precision matrix. This is the motivation for the *graphical Lasso*, see Friedman et al. (2008), which we will describe more precisely now.

Estimating Sparse Inverse Covariance Matrices with the Graphical Lasso

Denote the sample covariance matrix by $S = X^T X/n \in \mathbb{R}^{d \times d}$ where we have $X = (x_{i,j})_{i=1,\dots,n,j=1,\dots,d} \in \mathbb{R}^{n \times d}$ is the observed and centered data matrix with $\mathbf{X} \sim \mathcal{N}_d(\mathbf{0}, \Sigma)$. Then, the graphical Lasso calculates a sparse undirected graphical model by finding a solution for Ω . Considering the logarithm of (3.1) and taking derivatives with respect to Σ , we obtain as optimization problem

$$\max_{\Omega \in \mathbb{R}^{d \times d}} \log(\det(\Omega)) + \text{tr}(S\Omega), \quad (3.11)$$

with solution $\hat{\Omega} = S^{-1}$. However, this solution will in general have no zero entries and hence, induce no sparse graph. Furthermore, in high dimensional data sets we often have $d > n$, which leads to a singular matrix S and the inverse of S does not exist. The graphical Lasso overcomes this by introducing a penalty in (3.11) and solving

$$\max_{\Omega \in \mathbb{R}^{d \times d}} \log(\det(\Omega)) + \text{tr}(S\Omega) + \lambda \sum_{i=1}^d \sum_{j=1}^d |\Omega_{i,j}|. \quad (3.12)$$

depending on some *regularization parameter* $\lambda \geq 0$. For this optimization problem, many efficient numerical solvers also for thousands of dimensions exist. For $\lambda = 0$, there is no penalization and the solutions of (3.11) and (3.12) coincide. For fixed $\lambda > 0$, denote the solution of (3.12) by $\hat{\Omega}^\lambda$ and define $\mathcal{G}_\lambda = (V, E^\lambda)$ by

$$(i, j) \in E^\lambda \Leftrightarrow \hat{\Omega}_{i,j}^\lambda \neq 0. \quad (3.13)$$

Varying $\lambda > 0$, we obtain a piecewise constant *solution path* of graphs with different levels of sparsity. Letting $\lambda \rightarrow 0$, the solutions \mathcal{G}_λ will become more and more dense. For $\lambda \rightarrow \infty$, the number of *connected components* will increase but their individual sizes decreases. In practice, the solution path is calculated along a vector $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_J)$ with $\lambda_j > 0$ and $J \in \mathbb{N}$. Several modifications and improvements for the graphical Lasso have been proposed. For instance, Witten et al. (2011) demonstrates that the search for (3.12) can be carried out in terms of block-diagonal matrices, breaking apart the large problem into smaller ones.

Vital for us is another property, namely that the connected components with respect to some $\lambda > 0$ can also be calculated directly from the sample covariance matrix S . More precisely, consider a fixed $\lambda > 0$ and a solution $\hat{\Omega}^\lambda$ of (3.12) with $\mathcal{G}_\lambda = (V, E^\lambda)$ as defined in (3.13). Thus, $\hat{\Omega}^\lambda$ and equivalently E^λ induce a vertex partition

$$V = \bigcup_{i=1}^p V_i^\lambda,$$

where each V_i^λ is a connected component for $i = 1, \dots, p$, based on the edge set E^λ . Alternatively, define a graph $\mathcal{H} = (V, \mathcal{F}^\lambda)$ based on the sample covariance matrix S with edge set \mathcal{F}^λ and adjacency matrix $\Pi^{\mathcal{H}} = (\pi_{i,j}^{\mathcal{H}})_{i=1, \dots, d, j=1, \dots, d}$ such that

$$\pi_{i,j}^{\mathcal{H}} = 1 \Leftrightarrow (i, j) \in \mathcal{F}^\lambda \Leftrightarrow |S_{i,j}| \geq \lambda, \quad (3.14)$$

This way of assigning the edge set is called *screening*. The graph \mathcal{H} has now, say q connected components we denote by W_i for $i = 1, \dots, q$ and consider the associated partition $V = \bigcup_{i=1}^q W_i$. It has now been shown by Mazumder and Hastie (2012), that $p = q$ and moreover, $V_i = W_i$ for all $i = 1, \dots, p$. This makes a decomposition of the entire graphical Lasso problem more tractable as we can split it into p parallel tasks which can be performed entirely independent from each other. Furthermore, we have a very easy screening rule for intractably high dimensional datasets to decompose their dependence behaviour in multiple smaller parts which now are tractable. Inside the connected components, we can then use the graphical Lasso to obtain non-dense graphs. We give a brief example.

Example 3.12 (Screening). *Assume we have a dataset on the z-scale in six dimensions with the following empirical covariance matrix*

$$S = \begin{pmatrix} 1.0000 & 0.2058 & 0.1794 & 0.7340 & 0.7298 & 0.7167 \\ 0.2058 & 1.0000 & 0.3212 & 0.2643 & 0.3158 & 0.2848 \\ 0.1794 & 0.3212 & 1.0000 & 0.1895 & 0.2105 & 0.2327 \\ 0.7340 & 0.2643 & 0.1895 & 1.0000 & 0.9606 & 0.9089 \\ 0.7298 & 0.3158 & 0.2105 & 0.9606 & 1.0000 & 0.9378 \\ 0.7167 & 0.2848 & 0.2327 & 0.9089 & 0.9378 & 1.0000 \end{pmatrix}.$$

We use the *huge* R-package (Zhao et al., 2015) to calculate a sequence of $J = 4$ values

for λ given by

$$\lambda = (0.9607, 0.7438, 0.3452, 0.2070).$$

Solving (3.12) for these values we obtain graphical models $\mathcal{G}_{\lambda_1}, \dots, \mathcal{G}_{\lambda_4}$ as shown in Figure 3.6. Initially, there are only isolated nodes and hence $|V| = 6$ connected components.

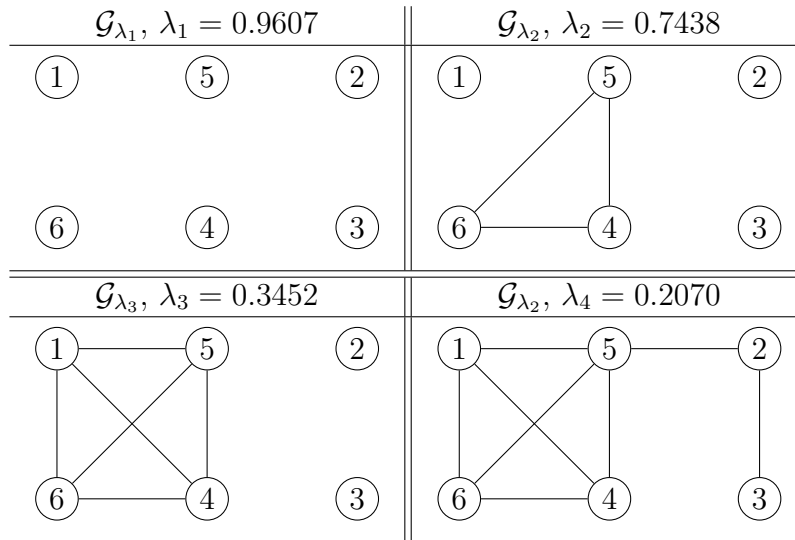


Figure 3.6: Example 3.12: Sequence of estimated graphical models $\mathcal{G}_{\lambda_1}, \dots, \mathcal{G}_{\lambda_4}$.

In the second and third graph, we have one connected component of size 3 together with isolated nodes and size 4 with isolated nodes. In the fourth graph, only one connected component of size $|V| = 6$ exists. If we consider $\lambda_2 = 0.7438$ and apply the screening rule (3.14), we obtain the following adjacency matrix based on S

$$\Pi^{\mathcal{G}_{\lambda_2}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}.$$

This adjacency matrix constitutes the same connected components as in the corresponding graph of the graphical Lasso solution path. It indicates that the nodes 4, 5, 6 are all connected to each other. However, the result by Mazumder and Hastie (2012) only assures that the connected components of the screening rule and the graphical Lasso are the same for a specific $\lambda > 0$, but no knowledge about the edges in the connected components can be drawn from the screening rule.

We continue this section with the consideration of directed graphical models, more precisely, *directed acyclic graphs (DAGs)*.

3.3.4 Directed Graphical Models

Let $\mathcal{G} = (V, E)$ be a directed acyclic graph (DAG). We will now introduce some special graph theoretic definitions, properties and notations for these graphs. If there exists a path from w to v , we write $w >_{\mathcal{G}} v$. Denote a disjoint union by $\dot{\cup}$, and define

- the *parents* $\text{pa}(v) := \{w \in V \mid w \rightarrow v\}$,
- the *ancestors* $\text{an}(v) := \{w \in V \mid w >_{\mathcal{G}} v\}$,
- the *descendants* $\text{de}(v) := \{w \in V \mid v >_{\mathcal{G}} w\}$,
- the *non-descendants* $\text{nd}(v) := V \setminus (\text{de}(v) \dot{\cup} \text{pa}(v) \dot{\cup} v)$.

Then, $V = v \dot{\cup} \text{pa}(v) \dot{\cup} \text{de}(v) \dot{\cup} \text{nd}(v)$ for all $v \in V$. $A \subseteq V$ is *ancestral* if $\text{pa}(v) \subseteq A$ for all $v \in A$, with $\text{An}(A)$ the smallest ancestral set containing A . We consider in the following DAGs with a limited number of parents.

Definition 3.13 (k-DAGs). *Let $\mathcal{G} = (V, E)$ be a DAG. We define*

$$k_v := |\text{pa}(v)|,$$

and speak of a *k-DAG* \mathcal{G} if and only if

$$k_v < k \text{ for all } v \in V.$$

Each DAG has a *topological ordering* (Andersson and Perlman, 1998), specified by an ordering function η . More precisely, let $V = \{v_1, \dots, v_d\}$ and $\eta : V \rightarrow \{1, \dots, d\}$ such that for each pair $v_i, v_j \in V$ we have

$$\eta(v_i) < \eta(v_j) \Rightarrow v_j \not>_{\mathcal{G}} v_i,$$

i. e. there is no path from v_j to v_i in \mathcal{G} . An ordering η always exists, but it is not necessarily unique. By $\{\eta^{-1}(1), \dots, \eta^{-1}(d)\}$, we refer to V ordered increasingly according to η and by $\{\eta^{-1}(d), \dots, \eta^{-1}(1)\}$ we refer to V ordered decreasingly according to η . We showcase an example of a DAG illustrating the previous definitions

Example 3.14 (DAG in six dimensions). *Consider the DAG $\mathcal{G} = (V, E)$ in Figure 3.7. Table 3.1 displays the topological ordering function, parents, descendants and non-descendants for all $v \in V$.*

v	$\eta(v)$	$\eta^{-1}(v)$	$\text{pa}(v) = \{w_1^v, w_2^v\}$	$\text{de}(v)$	$\text{nd}(v)$
1	1	1	-	2,3,4,5,6	-
2	2	2	1	3,4,5,6	-
3	4	6	6,2	-	1,4,5
4	6	3	5,2	-	1,3,6
5	5	5	6,2	4	1,3
6	3	4	2	3,4,5	1

Table 3.1: Example 3.14: Properties of the DAG \mathcal{G} in Figure 3.7.

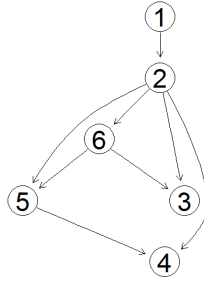


Figure 3.7: Example 3.14: DAG \mathcal{G} .

We see that a high value of $\eta(v)$ corresponds to more non-descendants. Note that η is not unique since $3 \leftrightarrow 5$ in \mathcal{G}_1 . Hence, a topological ordering for \mathcal{G}_1 is also given by $\{\eta^{-1}(1), \dots, \eta^{-1}(6)\} = \{1, 2, 6, 5, 3, 4\}$.

A *v-structure* in \mathcal{G} is a triple of nodes $(u, v, w) \in V$ where $u \rightarrow v$ and $w \rightarrow v$ but $u \not\leftrightarrow w$. The *moral graph* \mathcal{G}^m of a DAG \mathcal{G} is the skeleton \mathcal{G}^s of \mathcal{G} with an additional undirected edge (u, w) for each *v-structure* (u, v, w) . As for undirected graphs, separation can also be defined for DAGs, called *d-separation*.

Definition 3.15 (d-separation). Let $\mathcal{G} = (V, E)$ be a directed acyclic graph. A chain π from a to b in \mathcal{G} is said to be blocked by a set of nodes S , if it contains a node $\gamma \in \pi$ such that either

- (i) $\gamma \in S$ and arrows of π do not meet head-to-head at γ , or
- (ii) $\gamma \notin S$ nor has γ any descendants in S , and arrows of π do meet head-to-head at γ .

A chain that is not blocked by S is said to be active. Two subsets A and B are now said to be *d-separated* by S in \mathcal{G} if all chains from A to B are blocked by S in \mathcal{G} and we denote it by $A \perp_d B \mid S [\mathcal{G}]$.

Example 3.16 (d-separation). We give an example of the *d-separation* based on a DAG \mathcal{G} in five dimensions, see Figure 3.8.

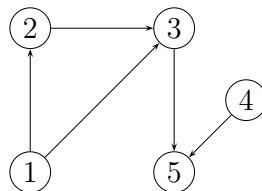


Figure 3.8: Example 3.16: DAG \mathcal{G} in five dimensions.

First, we want to consider whether $3 \perp_d 4 \mid 5 [\mathcal{G}]$ holds. By application of the *d-separation*, we see that $S = 5$ can not block a chain from 3 to 4 as arrows do meet head-to-head at 5, hence $3 \perp_d 4 \mid 5 [\mathcal{G}]$ does not hold.

Second, we consider whether $1 \perp_d 5 \mid 23 [\mathcal{G}]$. The chain from 1 to 5 via 3 is blocked

as arrows meet not head-to-head at 3. Secondly, also the chain from 1 to 5 via $\{2, 3\}$ is blocked as arrows meet not head-to-head. Hence, we conclude $1 \perp_d 5 \mid 23$ $[\mathcal{G}]$. For another example, see the example by Lauritzen (1996, p. 50).

In the case of undirected graphs, we related graphical separation to conditional independence. For graphical models on DAG we can also relate conditional independence to the corresponding graph theoretic properties.

Markov Properties on DAGs

As for undirected graphs, we can also define *Markov properties* on directed acyclic graphs. Sometimes, these graphical models on DAGs are also referred to as *Bayesian Networks*. We consider again a random vector $\mathbf{X} = (X_1, \dots, X_d)$ and assume $\mathbf{X} \sim \mathcal{N}_d(\mathbf{0}, \Sigma)$. Let $\mathcal{G} = (V, E)$ be a DAG. We say that \mathbf{X} obeys the *directed local Markov property* according to \mathcal{G} if

$$v \perp\!\!\!\perp \text{nd}(v) \mid \text{pa}(v) \text{ for all } v \in V. \quad (3.15)$$

Example 3.17 (Example 3.14 cont.). Consider again the DAG \mathcal{G} in six dimensions from Example 3.14.

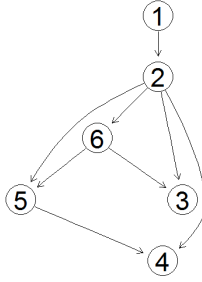


Figure 3.9: Example 3.17: DAG \mathcal{G} in six dimensions.

The directed local Markov property (3.15) for the DAG \mathcal{G} in Figure 3.9 gives the conditional independencies

$$\begin{aligned} 4 &\perp\!\!\!\perp 1, 3, 6 \mid 2, 5, \\ 5 &\perp\!\!\!\perp 1, 3 \mid 2, 6, \\ 3 &\perp\!\!\!\perp 1, 4, 5 \mid 2, 6, \\ 6 &\perp\!\!\!\perp 1 \mid 2. \end{aligned}$$

Furthermore, the joint density of \mathbf{X} can be directly written in terms of the DAG by

$$f(\mathbf{x}) = \prod_{i=1}^d f_{v_i \mid \text{pa}(v_i)}(x_{v_i} \mid \mathbf{x}_{\text{pa}(v_i)}), \quad (3.16)$$

see Lauritzen (1996, p. 46). Another important set when speaking about DAG models are the so called *Markov blankets*. The Markov blanket of a node $v \in V$ is the minimal subset $S \subseteq \{V \setminus v\}$ such that

$$v \perp\!\!\!\perp V \setminus \{S \cup v\} \mid S.$$

In other words, the Markov blanket S of a node v are all the nodes which are necessary to render any other node in $V \setminus \{S \cup v\}$ conditionally independent from v if conditioned on S . Finally, we relate DAG models to the previously introduced undirected graphical models, see Section 3.3.2.

Correspondence to Undirected Graphical Models

Given a graphical model on a DAG, we can easily find the corresponding undirected graphical model by considering the moral graph. More precisely, Lauritzen (1996, p. 48) showed that for a DAG $\mathcal{G} = (V, E)$ and disjoint sets $A, B, S \subseteq V$ it holds

$$A \perp_d B \mid S [\mathcal{G}] \Leftrightarrow A \perp B \mid S [(\mathcal{G}_{An(A \cup B \cup S)})^m].$$

The latter property is known as the *global directed Markov property*, which can be shown to be equivalent to the local directed Markov property, (3.15) (Lauritzen, 1996, p. 51). This also shows that the *d-separation* can be conveniently used to check conditional independence in a DAG.

Where estimating undirected graphs was directly related to identifying zero entries in the inverse covariance matrix, estimation of directed models is more complex. We will discuss approaches to estimate these models next.

3.3.5 Estimation of Graphical Models on DAGs

The task of estimating a graphical model on a DAG can be separated in two steps. The first step is concerned with the *structure learning* where the graph structure is identified. Several approaches to this goal have been proposed and they can be split into *score-based* and *constraint-based* methods. Another category which combines both are the *hybrid approaches*. Because of the intuitive appearance of DAG models, also learning from expert knowledge is possible. After the structure is obtained, a second step with *parameter learning* follows. Because of the parsimonious structure in (3.16), this task can be split into the corresponding factors $f_{v_i | \text{pa}(v_i)}(x_{v_i} | \mathbf{x}_{\text{pa}(v_i)})$, $i = 1, \dots, d$. We will first describe constraint based methods very briefly since they will not play an essential role in the remainder of the thesis. Afterwards, more attention is given to score-based methods, which will be used in Chapter 4.

Constraint-Based Methods

Given data, constraint-based methods perform several tests for conditional independence to find Markov blankets for all the nodes and afterwards, identify v-structures. Since the number of possible sets to test for conditional independence is growing exponentially in dimensions, a straightforward application is not possible. The first practically usable implementation was the *PC-algorithm*, see Spirtes et al. (2000, Section 5.4.2), where multiple conditional independence tests carried out in a specific order ensured identification of the graphical structure. The next step in this direction was the *Grow-Shrink (GS)*

algorithm by Margaritis (2003). It targets at identifying the Markov blanket for each node directly and afterwards orienting the edges. Subsequent recent extensions of that algorithm aimed at reducing the number of necessary conditional independence tests or avoiding false positives, see Nagarajan et al. (2013, p. 18). Because of the particular way these algorithms work, it is not directly possible to restrict the maximum number of parents for a node in the structure selection process to obtain a k -DAG, see Definition 3.13. Since we will need this property later on, we introduce score-based methods and in particular the *Hill-Climbing algorithm*, which allow for such a restriction.

Score-Based Methods

Recall that we assume multivariate Gaussianity of the random vector \mathbf{X} generating the data. To avoid double-subscripts, we identify the nodes in the node set $v_i \equiv i$ for $i = 1, \dots, d$. The idea behind score-based method is to find a DAG $\mathcal{G} = (V = \{1, \dots, d\}, E)$ whose corresponding probability distribution has highest score for the given data. More precisely, assume a factorization

$$f(\mathbf{x}) = \prod_{i=1}^d f_{i|\text{pa}(i)}(x_i | \mathbf{x}_{\text{pa}(i)}),$$

and a sample of n observations of d -dimensional data $(\mathbf{x}_1, \dots, \mathbf{x}_n)^T \in \mathbb{R}^{n \times d}$ with $\mathbf{x}_k \in \mathbb{R}^d$ for $k = 1, \dots, n$. Using the log-likelihood as score to be optimized for, we have

$$\mathcal{L}(\mathcal{G}, (\mathbf{x}_1, \dots, \mathbf{x}_n)) = \sum_{k=1}^n \sum_{i=1}^d \log \left(f_{i|\text{pa}(i)}(x_{k,i} | \mathbf{x}_{k,\text{pa}(i)}) \right),$$

which depends on the DAG \mathcal{G} via the sets $\text{pa}(i)$ for $i = 1, \dots, d$. Since the log-likelihood increases whenever new parameters enter the model, maximizing it leads often to overly parametrized and complicated models and the corresponding graph would contain the maximum number of edges such that it is still a DAG. Hence, a penalty is introduced to consider a different score, in our case, the *Bayesian Information Criterion (BIC)*

$$\text{BIC}(\mathcal{G}, (\mathbf{x}_1, \dots, \mathbf{x}_n)) = -2 \left(\sum_{k=1}^n \sum_{i=1}^d \log \left(f_{i|\text{pa}(i)}(x_{k,i} | \mathbf{x}_{k,\text{pa}(i)}) \right) \right) + \log(n) p, \quad (3.17)$$

where p is the number of parameters. For these, we have

$$p = 2|V| + |E|$$

as for each node $i = 1, \dots, d$, we have a mean and variance parameter for the marginal models and $|E|$ edges in the graph total. Neglecting the mean and assuming unit variance, it reduces to $p = |E|$ in (3.17).

Hill-Climbing Algorithm

The *Hill-Climbing algorithm*, see Nagarajan et al. (2013, p. 19) aims to optimize (3.17). Initially, we start with some graph \mathcal{G}_0 . Most often, this is the empty graph. Assume now we are in step $\ell - 1$ for $\ell \geq 1$ with a graph

$$\mathcal{G}_{\ell-1} = (V, E_{\ell-1}) \text{ with score } \text{BIC}(\mathcal{G}_{\ell-1}, (\mathbf{x}_1, \dots, \mathbf{x}_n)), \text{ for } \ell \geq 1.$$

Then, we can perform one of the following operations:

- addition of an edge $\mathcal{G}' = \mathcal{G}_{\ell-1} \cup (j, \ell)$ for some $(j, \ell) \notin E_{\ell-1}$,
- removal of an edge: $\mathcal{G}' = \mathcal{G}_{\ell-1} \setminus (j, \ell)$ for some $(j, \ell) \in E_{\ell-1}$,
- reversion of an edge: $\mathcal{G}' = (\mathcal{G}_{\ell-1} \setminus (j, \ell)) \cup (\ell, j)$ for some $(j, \ell) \in E_{\ell-1}$,

where we have to ensure that the resulting graph \mathcal{G}' is still a DAG, i. e. does not contain any directed cycles. Then, we have a new graph

$$\mathcal{G}' \text{ with score } \text{BIC}(\mathcal{G}', (\mathbf{x}_1, \dots, \mathbf{x}_n)).$$

We consider now whether

$$\text{BIC}(\mathcal{G}', (\mathbf{x}_1, \dots, \mathbf{x}_n)) < \text{BIC}(\mathcal{G}_{\ell-1}, (\mathbf{x}_1, \dots, \mathbf{x}_n))$$

and if this is the case, we set

$$\mathcal{G}_\ell := \mathcal{G}'.$$

and continue until no improvement can be attained by adding, removing or reversing any edges. Imposing now to have a k -DAG is easy by additionally checking to add or reverse only edges such that for no node in the graph, the number of maximum parents k is exceeded. Most often, this algorithm attains only a local optimizer but not a local one. To overcome this, one can run M random restarts and take the best score of all the M random restarts. In our practical considerations, we rely on the *Hill-Climbing* algorithm implementation in the `bnlearn` R-package, see Scutari (2010). This concludes the part describing graphical models.

3.4 Structural Equation Models (SEMs)

Structural equation models (SEMs) are a widely used class of models in the social sciences to model multivariate data. In their most general form, they allow to model actual observations and latent variables, see e. g. Kaplan (2009), Hoyle (1995) or Bollen (1989). This property is necessary, for example in psychology, where specific properties of individuals can not be measured directly. We want to stress that we are not considering latent variables in this thesis and are thus only dealing with actual observations. In the most general

form that we use, a SEM in d dimensions is defined by a system of linear equations

$$\begin{aligned} X_1 &= \psi_1 \epsilon_1, \\ X_2 &= \varphi_{2,1} X_1 + \psi_2 \epsilon_2, \\ X_j &= \sum_{i=1}^{j-1} \varphi_{j,\kappa_i(j)} X_{\kappa_i(j)} + \psi_j \epsilon_j, \quad j \in \{3, \dots, d\}, \end{aligned} \tag{3.18}$$

with $\epsilon_j \sim \mathcal{N}(0, 1)$ i.i.d. and ψ_j is such that $\text{Var}(X_j) = 1$ for $j = 1, \dots, d$. Additionally, ϵ_j is independent of X_1, \dots, X_{j-1} for $j = 2, \dots, d$ and $X_j \sim \mathcal{N}(0, 1)$, $j = 1, \dots, d$. The function $\kappa_i(j) \in \{1, \dots, j-1\}$ is used to specify the non-zero right hand sides of X_j . In the social sciences, a SEM is used to draw causal inference from data. In our work, this will not be the case. However, we use it to properly define a specific multivariate Gaussian distribution. More precisely, the SEM (3.18) leads to a parsimonious parametrization of the correlation matrix of the d -dimensional random vector \mathbf{X} , see Brechmann and Joe (2014). For example, the corresponding correlations and partial correlations can be expressed in terms of φ . Additionally, we can directly impose *conditional independence* in the multivariate Gaussian distribution by setting specific $\varphi_{j,\kappa_i(j)} = 0$ in the SEM for $j = 1, \dots, d$ and $i = \{1, \dots, j-1\}$. Consider the following example.

Example 3.18 (Conditional independence in SEMs). *Assume we have the following SEM in four dimensions.*

$$\begin{aligned} X_1 &= \psi_1 \epsilon_1, \\ X_2 &= \varphi_{2,1} X_1 + \psi_2 \epsilon_2, \\ X_3 &= \varphi_{3,1} X_1 + \varphi_{3,2} X_2 + \psi_3 \epsilon_3, \\ X_4 &= \varphi_{4,1} X_1 + \varphi_{4,2} X_2 + \psi_4 \epsilon_4. \end{aligned} \tag{3.19}$$

First, we monitor that X_3 and X_4 are jointly only governed by X_1 and X_2 and their corresponding random errors. This leads to the conclusion that the SEM induces $X_3 \perp\!\!\!\perp X_4 \mid X_1, X_2$. This can be seen as in following. X_3 and X_4 can be rewritten in terms of X_1, X_2 and their corresponding error terms. Each covariance of either X_1 or X_2 conditioned on X_1, X_2 is zero. The only remaining term is

$$\text{Cov}(\epsilon_3, \epsilon_4 \mid X_1, X_2) = \text{Cov}(\epsilon_3, \epsilon_4) = 0,$$

which can be concluded because of the mutual independence of ϵ_j for $j = 1, \dots, d$ and the assumption that ϵ_j is independent of X_1, \dots, X_{j-1} for $j = 2, \dots, d$.

This can be seen also even more easily in terms of graphical models and works as following. Based on a SEM in (3.18), define a graph $\mathcal{G} = (V = \{1, \dots, d\}, E = \emptyset)$ and then add arrows $X_{\kappa_i(j)} \rightarrow X_j$ for each $j \in 2, \dots, d$ and $i = 1, \dots, j-1$. In other words, each non-zero right hand side term $X_{\kappa_i(j)}$ yields an arrow $X_{\kappa_i(j)} \rightarrow X_j$. Thus, clearly, \mathcal{G} is a DAG since in a SEM we have that $\kappa_i(j) \in \{1, \dots, j-1\}$. Now, by Peters and Bühlmann (2014), the joint distribution of $\mathbf{X} = (X_1, \dots, X_d)$ is uniquely determined by \mathcal{G} and it is Markov with respect to \mathcal{G} . Additionally, if the SEM has at most k summands

on the right hand side of (3.18), we obtain a k -DAG. Furthermore, \mathcal{G} has a topological ordering $1, \dots, d$.

Example 3.19 (Example 3.18 cont.). *Consider the SEM of (3.19) again. Then we can define a DAG \mathcal{G} by drawing arrows from each right hand side term to its left hand side term and obtain the DAG \mathcal{G} in Figure 3.10.*

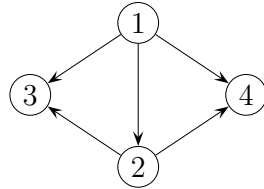


Figure 3.10: Example 3.19: DAG \mathcal{G} in four dimensions.

We can use the directed local Markov property (3.15) to directly conclude

$$3 \perp 4 \mid 1, 2 [\mathcal{G}] \Rightarrow 3 \perp\!\!\!\perp 4 \mid 1, 2.$$

Finally, recall that these considerations can be made since we assume multivariate Gaussianity for our models at the moment. For more general cases including strictly positive densities, see Lauritzen (1996, Chapter 2).

Chapter 4

Representing Sparse Gaussian DAGs as Sparse R-vines

We will now introduce the first of three methods for estimation of high dimensional vine copulas. The basic idea is to use a Gaussian graphical model, more precisely, a Gaussian k -DAG as a *proxy* to exploit on the one hand side structural information about the corresponding R-vine structure. On the other hand side, conditional independence information can be drawn from the DAG which can be used to make the R-vine more parsimonious and reduce estimation effort. There have been previous efforts to relate Gaussian DAGs or *Bayesian Networks* to copulas and vines. The work by Elidan (2010) exchanges Gaussian densities in (3.16) by copula densities. However, this leads to the consideration of copula densities in arbitrary dimensions and which is cumbersome as laid out in Section 2.1.4. The work by Bauer et al. (2012) and Bauer and Czado (2016) develops a pair copula construction for a DAG. Yet, this construction does not have the desirable property of an R-vine that all arguments of higher order copula functions are explicitly given in terms of lower order trees. Hence, it involves numerical integration with respect to arbitrary dimensions and is thus not feasible in high dimensional setups. Finally, Pircalabelu et al. (2017) estimate the factors in (3.16) by a quotient of C- and D-vine densities to obtain conditional densities. Yet, this approach does not lead to a joint multivariate distribution in general.

The outline of the chapter is as follows. First, we motivate our approach by illustrating the opposite transformation, i. e. how sparse R-vines can be represented by DAGs. Then, we tackle the original problem in the case of a 1-DAG which is particularly simple. Next, we consider the general case of a k -DAG and give sufficient conditions under which a k -DAG can be represented as a k -truncated R-vine. We will rigorously prove the statement and illustrate it throughout with examples. Afterwards, we give classes of k -DAGs for which we can generally prove the existence of a k -DAG representation and derive some necessary conditions along with some further information. As we see, the assumptions we have to impose on a k -DAG to obtain a k -truncated R-vine representation are quite strong. So, we motivate a more heuristic procedure which can be carried out in any case. We sketch an algorithmic implementation and finally give a data example.

The material presented in this chapter is a slight extension of Müller and Czado (2017b).

4.1 Motivation: Combinatorial Example

A brief motivation can be given in terms of a combinatorial consideration. We show that, given a k -DAG \mathcal{G} on d vertices, see Definition 3.13, the number of arrows in \mathcal{G} is less or equal than the number of edges in the first k R-vine trees of an R-vine in d dimensions. Thus, we get an idea to relate the edges in the DAG with the non-independence copulas in a k -truncated R-vine.

Lemma 4.1 (Correspondence between the number of arrows in a DAG and edges in an R-vine). *Let $\mathcal{G} = (V, E)$ be a k -DAG with $|V| = d$ and let T_1, \dots, T_k be an arbitrary R-vine tree sequence where T_1 has d nodes. Then, the number of arrows in \mathcal{G} is less or equal the number of edges in the first k R-vine trees, T_1, \dots, T_k .*

Proof. We label the nodes of the DAG \mathcal{G} according to its topological ordering and count the maximum number of parents for each node. Node v_1 has no parent, and node v_2 can only have one parent, i. e. v_1 . Node v_3 might have two parents, v_4 might have three parents and so on. This can be continued to maximumly k parents for node v_{k+1} and then remains constant with k parents until v_d . Thus, an upper bound for the number of arrows in the DAG \mathcal{G} is

$$\left(\sum_{i=1}^{k-1} i \right) + k \cdot (d - k) = kd - k^2 + \frac{k(k-1)}{2} = kd - \frac{k(k+1)}{2}. \quad (4.1)$$

In a k -truncated R-vine, the R-vine tree sequence can be divided in the first k trees and the last $d - 1 - k$ trees. The number of edges in all R-vine trees is $d(d-1)/2$ and tree T_i has $d - i$ edges for $i = 1, \dots, d - 1$. Thus, the number of edges in the first k trees is

$$\sum_{i=1}^k (d - i) = kd - \frac{k(k+1)}{2},$$

which is equal to the number of arrows in a DAG with at most k parents, see (4.1). \square

Thus, it is quite natural to try resembling k -DAGs as k -truncated R-vines by matching the corresponding edges in the DAG to unique conditioned sets of non-independence copulas in the R-vine. We will sketch this proceeding more in the next section, where we show how k -truncated R-vines can be represented by DAGs.

4.2 Representing Truncated R-vines as DAGs

To establish a connection between k -truncated Gaussian R-vines and DAGs, we follow Brechmann and Joe (2014) using *structural equation models (SEMs)*. Let $\mathcal{V} = (T_1, \dots, T_{d-1})$ be an R-vine tree sequence and assume without loss of generality $\{1, 2\} \in T_1$ and for $j = 3, \dots, d$ denote the edges in T_1 by $\{j, \kappa_1(j)\}$ with $\kappa_1(j) \in \{1, \dots, j-1\}$. The higher order trees contain edges $j, \kappa_i(j) | \kappa_1(j), \dots, \kappa_{i-1}(j) \in T_i$ for $i = 2, \dots, d-1$ with

$\kappa_i(j) \in \{1, \dots, j-1\}$. Based on this R-vine, define the associated SEM $\mathcal{S}(\mathcal{V})$ by

$$\begin{aligned} X_1 &= \psi_1 \epsilon_1, \\ X_2 &= \varphi_{2,1} X_1 + \psi_2 \epsilon_2, \\ X_j &= \sum_{i=1}^{j-1} \varphi_{j, \kappa_i(j)} X_{\kappa_i(j)} + \psi_j \epsilon_j, \end{aligned} \tag{4.2}$$

with $\epsilon_j \sim \mathcal{N}(0, 1)$ i.i.d. and ψ_j such that $\text{Var}(X_j) = 1$ for $j = 1, \dots, d$. Additionally, recall that ϵ_j is independent of X_1, \dots, X_{j-1} for $j = 2, \dots, d$ and $X_j \sim \mathcal{N}(0, 1)$, $j = 1, \dots, d$. In other words, given the R-vine matrix M , the diagonal of M can be extracted to be the inverse ordering on the left hand side of the SEM. The right hand side can then be written down using the columns of the R-vine matrix M . Thus, equation j in the SEM is given by $X_{M_{d-j+1, d-j+1}}$ on the left hand side and $X_{M_{d, d-j+1}}, \dots, X_{M_{d-j+2, d-j+1}}$ as non-zero coefficients on the right hand side for X_j with $j = 1, \dots, d$. By definition of the function κ , we implicitly imposed that the main diagonal of the R-vine matrix is $(d, \dots, 1)$. This is an easy way to represent a Gaussian truncated R-vine by an SEM. Leading to DAGs, we have that based on $\mathcal{S}(\mathcal{V})$, we define a graph $\mathcal{G} = (V = \{1, \dots, d\}, E = \emptyset)$ and then add arrows $X_{\kappa_i(j)} \rightarrow X_j$ for each $j \in 2, \dots, d$ and $i = 1, \dots, j-1$. In other words, each conditioned set of the R-vine yields an arrow. As seen previously in Section 3.4, \mathcal{G} is a DAG and the joint distribution of $\mathbf{X} = (X_1, \dots, X_d)$ is uniquely determined by \mathcal{G} and it is Markov with respect to \mathcal{G} . Additionally, if the R-vine is k -truncated, we have at most k summands on the right hand side of (4.2) and thus, obtain a k -DAG. Furthermore, \mathcal{G} has a topological ordering $1, \dots, d$ which is the inverse of the R-vine matrix main diagonal. We now show that if there is a truncation, then two different R-vines may have the same DAG representation.

Example 4.2 (Different 2-truncated R-vines with same DAG representation in four dimensions). Consider the following two 2-truncated R-vines in Figure 4.1 and Figure 4.2 and their 2-DAG representation, Figure 4.3. Since the conditioned sets of \mathcal{V}_1 and \mathcal{V}_2 in the non-truncated trees are the same, both R-vines have the same DAG representation \mathcal{G}_2 . The coefficients on associated edges of DAGs are however different, so correlation matrices are different even if the DAG structure is the same. Yet, both correlation matrices are belonging to distributions which are Markov with respect to \mathcal{G}_2 . Thus, the conditional independencies in the R-vine imposed by the truncation are reflected by the directed local Markov properties in the DAG.

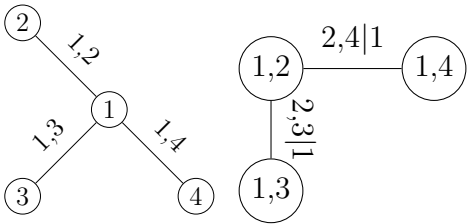


Figure 4.1: Example 4.2: R-vine \mathcal{V}_1 .

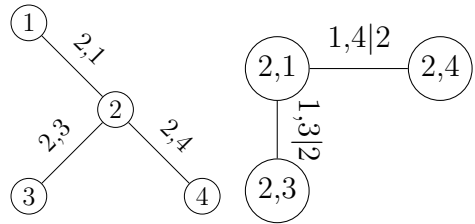


Figure 4.2: Example 4.2: R-vine \mathcal{V}_2 .

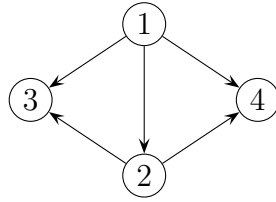


Figure 4.3: Example 4.2: DAG \mathcal{G}_2 associated to \mathcal{V}_1 and \mathcal{V}_2 .

This demonstrates how conditional independencies in R-vines are represented by DAGs. Our goal is to find R-vines representing the conditional independencies given by a DAG which will be covered next.

4.3 Representing DAGs as Truncated R-vines under Sufficient Conditions

First, we formalize our understanding of an R-vine representation of a DAG. The idea is that the R-vine tree sequence is based on the k -DAG \mathcal{G} which imposes a k -truncation.

Definition 4.3 (R-vine representation of a DAG). *Let $\mathcal{G} = (V, E)$ be a k -DAG. A k -truncated R-vine representation of \mathcal{G} is an R-vine $\mathcal{W}(\mathcal{G}) = (\mathcal{V}, \mathcal{B}(\mathcal{V}), \Theta(\mathcal{B}(\mathcal{V})))$ with R-vine tree sequence $\mathcal{V} = (T_1, \dots, T_{d-1})$ such that the R-vine trees T_{k+1}, \dots, T_{d-1} contain edges $[j(e), \ell(e); D(e)]$ where $j(e) \perp\!\!\!\perp \ell(e) \mid D(e)$ by \mathcal{G} .*

We will now find R-vine representations for 1-DAGs, continued by representations for general k -DAGs afterwards.

4.3.1 Representing 1-DAGs as 1-Truncated R-vines

A representation of a 1-DAG, i. e. a *Markov-tree*, is particularly easy to find.

Proposition 4.4 (Representing Markov Trees). *Let $\mathcal{G} = (V, E)$ be a 1-DAG. There exists a 1-truncated R-vine representation $\mathcal{W}(\mathcal{G})$ of \mathcal{G} . If $|E| = d - 1$, $T_1 = \mathcal{G}^s = \mathcal{G}^m$.*

The proof will be made using results from the general part and thus, deferred to a later section. For the moment, it suffices to see that in the special case of a 1-DAG, the skeleton is an undirected graph and is the first R-vine tree T_1 of an R-vine representation. We will now consider the general case.

4.3.2 Representing k -DAGs under Sufficient Conditions

Let $\mathcal{G} = (V, E)$ be an arbitrary k -DAG. We evaluate sufficient conditions such that an incomplete R-vine tree sequence $(T_1 = (V, E_1), \dots, T_k = (V_k, E_k))$ can be extended to a k -truncated R-vine representation $\mathcal{W}(\mathcal{G})$ of \mathcal{G} .

For the proof, we first present two lemmas. The first makes assertions about the conditioned sets in higher order R-vine trees with respect to DAG properties.

Lemma 4.7. *Let \mathcal{G} be a k -DAG and T_1, \dots, T_k an R-vine tree sequence satisfying the assumptions A1 and A2. For each $[j(e), \ell(e); D(e)]$ with $e \in E_{k+1}, \dots, E_{d-1}$, we have $\ell(e) \in \text{nd}(j(e))$.*

Proof. Consider an arbitrary edge $[j(e), \ell(e); D(e)]$ for $e \in E_{k+1}, \dots, E_{d-1}$. We have $\ell(e) \notin \text{pa}(j(e))$, since conditioned sets in an R-vine tree sequence are unique and all conditioned sets of the form $j(e), \ell(e)$ with $\ell(e) \in \text{pa}(j(e))$ occurred already in the first k trees by A1. Additionally, $\ell(e) \notin \text{de}(j(e))$, since otherwise would violate A2, as $\ell(e) >_{\mathcal{G}} j(e)$. Finally, $\ell(e) \neq j(e)$, since the two elements of a conditioned set must be distinct. Thus, we have $\ell(e) \notin (\text{pa}(j(e)) \dot{\cup} \text{de}(j(e)) \dot{\cup} j(e)) = V \setminus \text{nd}(j(e))$ and hence $\ell(e) \in \text{nd}(j(e))$. \square

We will use the previous example to showcase this property.

Example 4.8 (Example 4.5 cont.). *Illustrating Lemma 4.7, consider the R-vine matrix M of Example 4.5 and column 3. To complete M , we need to fill in e.g. $M_{4,3}$. Valid entries can come from the main diagonal of M right of 3, i.e. $\{M_{4,4}, M_{5,5}, M_{6,6}\} = \{6, 2, 1\}$. Since $\text{pa}(3) = \{2, 6\}$ and by A1, the edges in the first two R-vine trees are $\{3, 6\}$ and $3, 2|6$, the only remaining entry is $M_{4,3} = 1$. This can only be a non-descendant of 3 because of A2.*

The next lemma is necessary to precisely define the contents of the conditioning set in higher order trees.

Lemma 4.9. *Let \mathcal{G} be a k -DAG and T_1, \dots, T_k an R-vine tree sequence satisfying the assumptions A1 and A2. For each $[j(e), \ell(e); D(e)]$ with $e \in E_{k+1}, \dots, E_{d-1}$, we have $D(e) \subseteq \{\text{pa}(j(e)) \dot{\cup} \text{nd}(j(e))\}$.*

Proof. Consider $j(e'), \ell(e'); D(e')$ for $e' \in E_{k+1}$. We have the following two cases.

First case: $|\text{pa}(j(e'))| = k$. All parents of $j(e')$ occurred in the conditioned set of edges together with $j(e')$ in the first k R-vine trees. Hence, $\text{pa}(j(e')) = D(e')$ and $|D(e')| = k$.

Second case: $|\text{pa}(j(e'))| =: k_{j(e')} < k$. Similarly to the first case, we conclude $\text{pa}(j(e')) \subset D(e')$. Let $D(e') \setminus \text{pa}(j(e')) = D_1$ with $|D_1(e')| = k - k_{j(e')} > 0$. To obtain the elements of $D(e')$, recall A2 and consider the column of the R-vine matrix M in which $j(e')$ is in the diagonal, say column p . The entries $\{M_{d-k,p}, \dots, M_{d,p}\}$ describe the elements which occurred in conditioned sets together with $j(e')$ in the first k trees. As these entries may only be taken from the right of $M_{p,p} = j(e')$, these must be non-descendants of $j(e')$. To conclude the statement for the R-vine trees T_{k+2}, \dots, T_{d-1} , we use an inductive argument. Let $e'' \in E_{k+2}$ and $j(e'')$ is in the diagonal of the R-vine matrix M in column p . Then, for the conditioning set of e'' we have $D(e'') = M_{d-k-1,p} \dot{\cup} \{M_{d-k,p}, \dots, M_{d,p}\}$. For the set $\{M_{d-k,p}, \dots, M_{d,p}\}$ we have shown that it can only consist of parents and non-descendants of $j(e'')$. As $M_{d-k-1,p}$ can only have a value occurring in the main diagonal of the R-vine matrix to the right of column p , it must be a non-descendant of $j(e'')$. The same argument holds inductively for the trees T_{k+3}, \dots, T_{d-1} . Thus, we have shown that for each edge $[j(e), \ell(e); D(e)]$ with $e \in E_{k+1}, \dots, E_{d-1}$ we have $D(e) \subseteq \{\text{pa}(j(e)) \dot{\cup} \text{nd}(j(e))\}$. \square

Again, we continue with our previous example and see how the aforementioned property affects the R-vine structure.

Example 4.10 (Example 4.8 cont.). *Consider the first column of M with $M_{1,1} = 4$. Since $\text{pa}(4) = \{2, 5\}$, $\{4, 5\} \in E_1$ and $4, 2|5 \in E_2$, independently of the values in $M_{2,1}, \dots, M_{4,1}$, $\text{pa}(4) = \{2, 5\}$ is in the conditioning set for each of the edges associated to these entries of M . There are more nodes in the conditioning set, which are non-descendants of 4 by A_2 .*

We now conclude with the proof of Theorem 4.6 using the Lemmas 4.7 and 4.9. The main idea is to use the general expression for pair copulas in higher order trees and to relate it to the directed local Markov property in the corresponding DAG.

Proof. Abbreviate $j_e \equiv j(e)$, $\ell_e \equiv \ell(e)$, $D_e \equiv D(e)$ and set $j_e, \ell_e | D_e \equiv [j(e), \ell(e); D(e)]$ with $e \in E_{k+1}, \dots, E_{d-1}$ arbitrary but fixed. For the node j_e in the DAG \mathcal{G} we have by the directed local Markov property (3.15) that $j_e \perp\!\!\!\perp \text{nd}(j_e) \mid \text{pa}(j_e)$ and thus with Lemma 4.7,

$$j_e \perp\!\!\!\perp \ell_e \dot{\cup} (\text{nd}(j_e) \setminus \ell_e) \mid \text{pa}(j_e). \quad (4.4)$$

Set $\hat{\text{nd}}(j_e) := D_e \setminus \text{pa}(j_e)$ with $\hat{\text{nd}}(j_e) \subseteq \text{nd}(j_e)$ by Lemma 4.9, plug it into (4.4) obtaining

$$j_e \perp\!\!\!\perp \left(\ell_e \dot{\cup} \left((\text{nd}(j_e) \setminus \ell_e) \setminus \hat{\text{nd}}(j_e) \right) \dot{\cup} \hat{\text{nd}}(j_e) \right) \mid \text{pa}(j_e), \quad (4.5)$$

exploiting $\ell_e \cap \hat{\text{nd}}(j_e) = \emptyset$, i. e. a node can not be part of the conditioning and the conditioned set of the same edge. Applying Proposition 3.6 on (4.5) yields $j_e \perp\!\!\!\perp \ell_e \dot{\cup} \hat{\text{nd}}(j_e) \mid \text{pa}(j_e)$ by dropping $\left((\text{nd}(j_e) \setminus \ell_e) \setminus \hat{\text{nd}}(j_e) \right)$ in (4.5). $\ell_e \dot{\cup} \hat{\text{nd}}(j_e)$ is a disjoint union on which Proposition 3.6 can be applied to conclude $j_e \perp\!\!\!\perp \ell_e \mid \text{pa}(j_e) \dot{\cup} \hat{\text{nd}}(j_e)$. By definition of $\hat{\text{nd}}(j_e)$, we have $D_e = \text{pa}(j_e) \dot{\cup} \hat{\text{nd}}(j_e)$ and obtain the final result $j_e \perp\!\!\!\perp \ell_e \mid D_e$ for $e \in E_{k+1}, \dots, E_{d-1}$. Since each edge is assigned a pair copula density, we can now choose the independence copula density c^\perp for these edges in E_{k+1}, \dots, E_{d-1} backed by the conditional independence properties of the DAG. The resulting R-vine is thus a k -truncated R-vine. \square

Finally, we illustrate how the previous results all come together to draw conditional independence statements for higher order R-vine trees.

Example 4.11 (Example 4.10 cont.). *We illustrate Theorem 4.6 using the previous Examples 4.8 and 4.10. Consider column 1 of M and edge $4, 3|256 \in E_4$. From the conditional independence $4 \perp\!\!\!\perp 1, 3, 6 \mid 2, 5$ obtained from the DAG \mathcal{G} , we select the non-descendants of 4 to neglect, i. e. 1, to yield $4 \perp\!\!\!\perp 3, 6 \mid 2, 5$ by application of Proposition 3.6 and finally $4 \perp\!\!\!\perp 3 \mid 2, 5, 6$ by second application of Proposition 3.6.*

With the main result in Theorem 4.6, we can now also easily provide the proof for Proposition 4.4.

Proof. We assume $|E| = d - 1$. If not, the argument can be applied to each weakly connected subgraph of \mathcal{G} . Since $k = 1$, there are no v -structures, hence, the moral graph \mathcal{G}^m is the skeleton \mathcal{G}^s of \mathcal{G} and \mathcal{G}^m is connected. Since there are $d - 1$ arrows in \mathcal{G} , there are $d - 1$ undirected edges in \mathcal{G}^m . Since each connected graph on d nodes with $d - 1$ edges is a tree, \mathcal{G}^m is a tree. Additionally, each edge in \mathcal{G}^m corresponds to an arrow $w \leftrightarrow v$ in the DAG \mathcal{G} , satisfying Assumption A1. The main diagonal of the R-vine matrix can be chosen to be a decreasing topological ordering of \mathcal{G} by starting with a node which has no descendants but one parent, say v_d and let the corresponding R-vine matrix M be such that $M_{1,1} = v_d$. Thus, its parent and all other nodes must occur on the diagonal to the right of it. Next, take a node which has either one descendant, i. e. v_d or no descendant, denote v_{d-1} and set $M_{2,2} = v_{d-1}$. This can be repeated until $v_1 = M_{d,d}$ and determines the R-vine matrix main diagonal which is a decreasing topological ordering of \mathcal{G} , satisfying Assumption A2 onto which Theorem 4.6 applies. \square

This describes the main theoretical part of our work. Under quite strong assumptions, we were able to relate Gaussian k -DAGs to Gaussian k -truncated R-vines. However, we see that our assumptions imposed that we already had an incomplete R-vine tree sequence which was then extended. Finding the R-vine representation $\mathcal{W}(\mathcal{G})$ of an arbitrary k -DAG \mathcal{G} is a complex combinatorial problem as the existence of an incomplete R-vine tree sequence satisfying A1 and A2 is not clear. We now show special classes of k -DAGs where we can prove the existence of their k -truncated R-vine representations.

4.3.3 Special k -DAGs with R-vine Representations

Corollary 4.12 (k -DAGs with R-vine representation). *Let $\mathcal{G} = (V, E)$ be a k -DAG such that $V = \{v_1, \dots, v_d\}$ is an increasing topological ordering of \mathcal{G} . If, for all $v_i \in V$, (a) $\text{pa}(v_i) \subseteq \{v_{i-k}, \dots, v_{i-1}\}$ for all $i > k$, then an R-vine representation $\mathcal{W}(\mathcal{G})$ of \mathcal{G} exists; (b) $\text{pa}(v_i) \subseteq \{v_1, \dots, v_k\}$ for all $i > k$, an R-vine representation $\mathcal{W}(\mathcal{G})$ of \mathcal{G} exists.*

For these special k -DAGs, we can prove their R-vine representations in terms of k -truncated D-vines and k -truncated C-Vines.

Proof. Case (a): Let $\text{pa}(v_i) \subseteq \{v_{i-k}, \dots, v_{i-1}\}$. The R-vine representation $\mathcal{W}(\mathcal{G})$ is given by T_1 being a path from v_1 to v_d according to the topological ordering of \mathcal{G} , i. e. a D -vine. Because of the proximity condition, T_2, \dots, T_{d-1} are uniquely determined by T_1 . In tree T_j , the edges have the form $v_i, v_{i-j} | v_{i-j+1}, \dots, v_{i-1}$ and each conditioned set in the first k R-vine trees represents an arrow of \mathcal{G} , satisfying A1. A2 also holds since in a D-vine, the main diagonal of the R-vine matrix can be written as ordering v_d, \dots, v_1 of the path in T_1 .

Case (b): If $\text{pa}(v_i) \subseteq \{v_1, \dots, v_k\}$, T_1 is given a $star$ with central node v_1 . T_2 is a star with central node $\{v_1, v_2\}$ and so on, giving rise to a C -vine. In tree T_{k-j} , the edges have the form $v_i, v_{k-j} | v_1, \dots, v_{k-j-1}$ for $i \geq k$, satisfying A1. The main diagonal of the R-vine matrix of a C-vine is ordered according to the central nodes in the C-vine, satisfying A2. To both, Theorem 4.6 applies. Examples of a 2-DAG with D-vine and a k -DAG with C-vine representation are shown in Figure 4.4. \square

We show graphical examples of the aforementioned special k -DAG classes in Corollary 4.12.

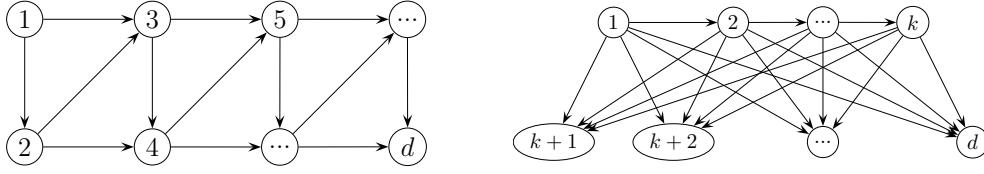


Figure 4.4: Examples of DAGs with D-Vine (left) and C-vine (right) representation.

In practical applications, given a k -DAG \mathcal{G} , it is most important to find a suitable first R-vine tree T_1 of an R-vine representation $\mathcal{W}(\mathcal{G})$ since this tree puts significant restrictions on the higher order R-vine trees by the proximity condition. Hence, given some DAG, it makes sense to impose necessary conditions on the first R-vine tree which have to hold for an R-vine representation to exist.

4.3.4 A Necessary Condition for R-vine Representations

Proposition 4.13. *Assume a k -DAG $\mathcal{G} = (V, E)$, recall $k_v = |\text{pa}(v)| \leq k$ and*

$$V^v := \{v, \text{pa}(v)\} = \{v, w_1^v, \dots, w_{k_v}^v\}, v \in V.$$

For $v \in V$, denote the induced subgraphs $T_1^v := (V^v, E^v) \subseteq T_1$ of $T_1 = (V, E_1)$ on V^v . Thus, E^v contains all edges in T_1 between nodes of V^v . If there exists an R-vine representation $\mathcal{W}(\mathcal{G})$ with incomplete R-vine tree sequence $\mathcal{V} = (T_1, \dots, T_k)$ such that A1 and A2 hold, then, $T_1 = (V, E_1)$ must be such that

- (i) for all $v \in V$ with $\text{pa}(v) = k$, T_1^v contains a path involving all nodes of V^v ,
- (ii) the union of the induced subgraphs $\bigcup_{i \in \mathcal{I}} T_1^{v_i} := (\bigcup_{i \in \mathcal{I}} V^{v_i}, \bigcup_{i \in \mathcal{I}} E^{v_i}) \subseteq T_1$ is acyclic for $\mathcal{I} := \{i \in V : |\text{pa}(v_i)| = k_{v_i} = k\}$.

We prove the result using the proximity condition and graphical arguments.

Proof. To show (i), assume (T_1, \dots, T_k) of $\mathcal{W}(\mathcal{G})$ satisfies A1 and A2. Choose $v \in V$ with $k_v = k$ arbitrary but fixed. Order the set $\text{pa}(v) = \{w_1^v, \dots, w_k^v\}$ such that v, w_i^v is the conditioned set of an edge $e \in E_i$, $i = 1, \dots, k$, ensured by A2. Then, by the proof of Theorem 4.6, each edge $e \in E_i$, $i = 1, \dots, k$ corresponding to $v \in V$ must have the form $v, w_i^v | w_1^v, \dots, w_{i-1}^v$. By the set formalism, see Example 2.5, and the proximity condition, see (iii) on page 17, we have $\{\{v, w_1^v\}, \{w_1^v, w_2^v\}\} \in E_2$ requiring $\{v, w_1^v\} \in E_1$ and $\{w_1^v, w_2^v\} \in E_1$. For $\{\{\{v, w_1^v\}, \{w_1^v, w_2^v\}\}, \{\{w_1^v, w_2^v\}, \{w_2^v, w_3^v\}\}\} \in E_3$ we can conclude in a first step $\{\{w_1^v, w_2^v\}, \{w_2^v, w_3^v\}\} \in E_2$ and in a second step $\{w_2^v, w_3^v\} \in E_1$. This can be extended to E_k and yields $\{w_i^v, w_{i+1}^v\} \in E_1$ for $i = 1, \dots, k-1$. Thus, v and its parents, i. e. V^v represent a path in T_1^v . Showing (ii), for each $i \in \mathcal{I}$ the graph $T_1^{v_i}$ is a subgraph of T_1 by (i). Thus, the union of $T_1^{v_i}$ over all $i \in \mathcal{I}$ must be a subgraph of T_1 . Since T_1 is a tree, it is acyclic, hence, each of its subgraphs must be, and so the graph in (ii). \square

Whereas the proof of (i) is a direct consequence of the proximity condition, the proof of (ii) is less intuitive. Hence, we show a graphical example illustrating it.

Example 4.14 (DAG in six dimensions). *Consider the DAG \mathcal{G} in Figure 4.5. By Proposition 4.13, we need to find an R-vine tree $T_1 = (V, E_1)$ such that the induced subgraphs $T_1^v = (V^v, E^v) \subseteq T_1$ contain paths involving all nodes of V^v for $V^v \in \{\{4, 1, 2\}, \{5, 1, 3\}, \{6, 2, 3\}\}$.*

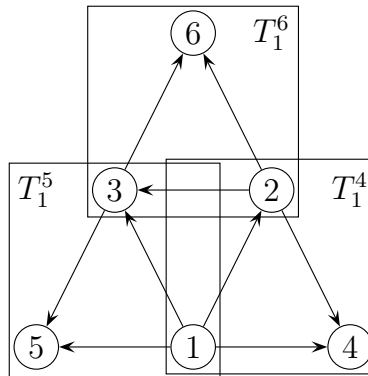


Figure 4.5: Example 4.14: DAG \mathcal{G}

Then, use the path T_1^4 from 1 to 2, T_1^6 from 2 to 3 and finally T_1^5 from 3 to 1. However, this creates a cycle in T_1 , which is a contradiction. Removing any edge which closes the cycle yields an induced subgraph which is not connected. Thus, the DAG \mathcal{G}_2 can not be represented by a 2-truncated R-vine. 3-truncated R-vines are given in the Figures 4.6 and 4.7.

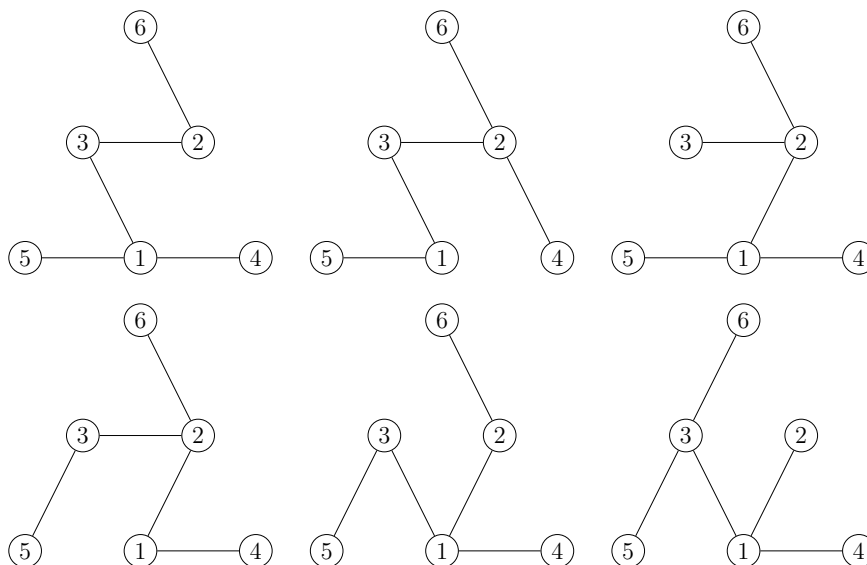


Figure 4.6: Example 4.14: Six admissible choices of first R-vine trees T_1 leading to a 3-truncated R-vine.

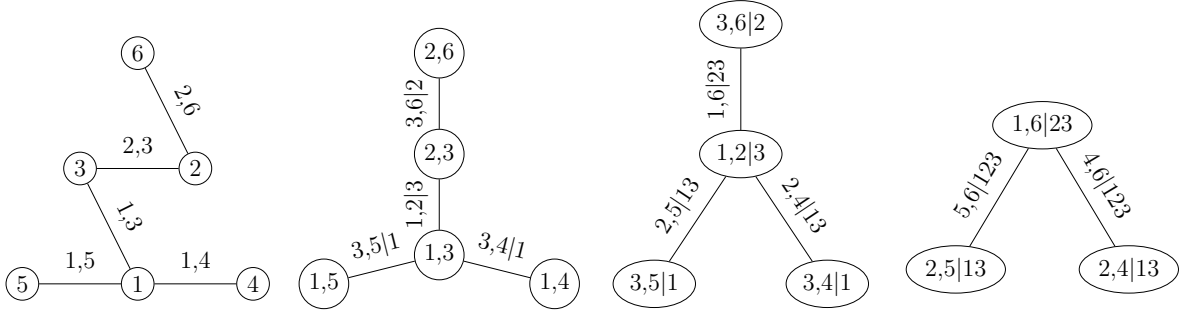


Figure 4.7: Example 4.14: First four R-vine trees T_1, T_2, T_3, T_4 (from left to right), showing the 3-truncation as given by the conditional independence properties in the DAG \mathcal{G} . Note that $24|13$ can not be set to the independence copula since 2 is a parent of 4.

Based on Proposition 4.13, we are given an intuition how to construct an admissible first R-vine tree T_1 of an R-vine representation $\mathcal{W}(\mathcal{G})$ of a DAG \mathcal{G} . Moreover, it also yields a best possible truncation level $k' > k$ for which a k' -truncated R-vine representation exists.

Corollary 4.15 (Best possible truncation level k'). *Consider a k -DAG $\mathcal{G} = (V, E)$. Let $T_1 = (V, E_1)$ be a tree and for each $v, w \in V$ let δ_v^w be the length of the unique path from v to w in T_1 . If T_1 is extended by successive R-vine trees $T_i, i \in \{2, \dots, d-1\}$, then the truncation level k' can be bounded from below by*

$$k' \geq \max_{v \in V} \max_{w \in \text{pa}(v)} \delta_v^w.$$

Proof. Since T_1 is a tree, all paths are unique. If not, there exist two distinct paths between v and w and both paths together are a cycle from v to v . Consider an arbitrary node $v \in V$ with parents $\text{pa}(v) = \{w_1^v, \dots, w_{k_v}^v\}$ in \mathcal{G} such that $w_{k_v}^v := \arg \max_{w \in \text{pa}(v)} \delta_v^w$, then there exists a unique path from v to $w_{k_v}^v$, $v = \alpha_0, \dots, \alpha_{\delta_v^w} = w_{k_v}^v$. From Theorem 4.6, our goal is to obtain edges with conditioned sets v, w with $w \in \text{pa}(v)$ in an R-vine tree T_i with lowest possible order i . Similar to the proof of Proposition 4.13, we try to obtain an edge

$$v, w_{k_v}^v | \alpha_1, \dots, \alpha_{\delta_v^w - 1} \in T_{\delta_v^w}, \quad (4.6)$$

with $\delta_v^w - 1$ entries in the conditioning set. The conditioned set of 4.6 can not occur in a tree T_i with $i < \delta_v^w$ because of the proximity condition and since the path between v and $w_{k_v}^v$ is unique. By the d-separation, page 38, two nodes in a DAG connected by an arrow, i.e. v and its parent $w_{k_v}^v$, can not be d-separated by any set S . Thus, the pair copula density associated to the edge (4.6) in $T_{\delta_v^w}$ is not the independence copula density c^\perp . This tree $T_{\delta_v^w}$ is characterized by a path distance in T_1 and the maximum path distance over all parents of $v \in V$ yields the highest lower bound. As it has to hold for all $v \in V$, we obtain a lower bound for the truncation level k' by the maximum over all $v \in V$. \square

We present a brief example for the Corollary.

Example 4.16 (Example for Corollary 4.15). *Consider the R-vine tree T_1 in Figure 4.8.*

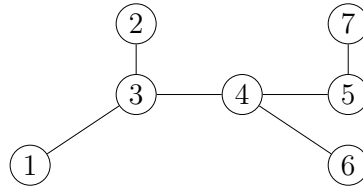


Figure 4.8: Example 4.16: R-vine tree T_1

Assume an underlying DAG \mathcal{G} with $1 \in \text{pa}(7)$. We have a lower bound for the truncation level $k' \geq 4$ since the path in T_1 from 7 to 1 is $7 - 5 - 4 - 3 - 1$ with a path length $\ell_7^1 = 4$. Not earlier as in tree T_4 , i. e. not in the trees T_1, T_2, T_3 an edge with conditioned set $7, 1$ can be obtained which can not be represented by the independence copula. The corresponding edges in the trees T_1, \dots, T_4 are as follows.

$T_1 :$	1, 3	3, 4	4, 5	5, 7
$T_2 :$	1, 4 3	3, 5 4	4, 7 5	
$T_3 :$	1, 5 3, 4	3, 7 4, 5		
$T_4 :$	1, 7 3, 4, 5			

Summing up, A1 and A2 are strong assumptions and hence only rarely satisfied for arbitrary DAGs. This motivates an heuristic approach for arbitrary k -DAGs to find a sparse R-vine representation exploiting their conditional independencies, even if no exact representation in the sense of Definition 4.3 exists.

4.4 Representing Arbitrary k -DAGs

From the more theoretical arguments in the previous section, we will now consider more explicitly k -DAGs fitted on data. Our goal is to find an R-vine representation $\mathcal{W}(\mathcal{G}_k)$ of an arbitrary k -DAG \mathcal{G}_k for $k \geq 2$. This R-vine representation \mathcal{W} can then equipped with non-Gaussian pair copulas for each non-independence copula. For the first R-vine tree T_1 , we have d^{d-2} candidates. Considering all these and checking Proposition 4.13 is not feasible. Additionally, A2 is hard to check upfront since it is not fully understood how a certain R-vine matrix diagonal relates to specific R-vines. Fixing the main diagonal may also result in suboptimal models. Hence, Theorem 4.6 can not be applied directly. With the motivation, that we have data on which we can fit k -DAGs using e. g. the *Hill-Climbing* algorithm, see Section 3.3.5, we want to sketch an algorithm allowing first of all to learn the R-vine structure from the structure of the DAGs. Second, as outlined, the goal is to incorporate conditional independence from the DAG models to regulate for sparsity.

4.4.1 Algorithmic Implementation

By A1, arrows in \mathcal{G}_k shall be modelled as conditioned sets in R-vine trees T_i for $i \in \{1, \dots, k\}$. Yet, for $k \geq 2$, there may be up to $kd - (k(k+1))/2$ candidate edges for T_1

which is limited to $d-1$ edges. Hence, it is crucial to find the most important arrows of \mathcal{G}_k for T_1 . An heuristic measure for the importance of an arrow $v \rightarrow w$ in \mathcal{G}_k , fitted to data, is how often the arrow $v \rightarrow w$ exists in $1, \dots, k-1$ -DAGs $\mathcal{G}_1, \dots, \mathcal{G}_{k-1}$, also fitted to data. However, also an arrow $w \rightarrow v$ is possible. Since R-vines are undirected graphical models, we neglect the orientation of arrows in the DAGs by considering their skeletons. Thus, for each edge (v, w) in the skeleton \mathcal{G}_k^s of the DAG \mathcal{G}_k we estimate DAGs $\mathcal{G}_1, \dots, \mathcal{G}_{k-1}$ using the data, obtain their skeletons $\mathcal{G}_1^s, \dots, \mathcal{G}_{k-1}^s$ and count how often the edge (v, w) exists in these graphs. An edge $(v, w) \in \mathcal{G}_i^s$ might be more important than an edge $(v, w) \in \mathcal{G}_j^s$ with $i < j$, which we describe by a non-increasing function of the maximum number of parents $g(i)$. Formally, consider i -DAGs \mathcal{G}_i for $i = 1, \dots, k$ estimated using the data. Denote $\mathcal{G}_i^s = (V, E_i^s)$ the skeleton of \mathcal{G}_i for $i = 1, \dots, k$ and define an undirected graph $\mathcal{H} = (V, E_1^{\mathcal{H}}) := \bigcup_{i=1}^k \mathcal{G}_i^s$ with edge weights μ_1 for $(v, w) \in E_1^{\mathcal{H}}$ given by

$$\mu_1(v, w) := \sum_{i=1}^k g(i) \mathbb{1}_{\{(v,w) \in E_i^s\}}(v, w), \quad (4.7)$$

with $g(i) > 0$ non-increasing for $i = 1, \dots, k$. Since we have for the weights $g(i) > 0$, we obtain $\mu_1 \geq 0$. In the remainder of this section, assume $g(i) \equiv 1$ for simplicity. This is also chosen in our algorithmic implementation. Now, on \mathcal{H} , find a maximum spanning tree T_1 by, e. g. Prim (1957), maximizing the sum of μ_1 . The higher order trees are built iteratively. Define a full graph $T_2 = (V_2, E_2)$ on $V_2 = E_1$ and delete each edge in E_2 not allowed by the proximity condition. Additionally, let $g(k) > \mu_0 > 0$. Denote the edges by $[j(e), \ell(e); D(e)]$ for $e \in E_2$ and set

$$\begin{aligned} \mu_2(e) &= \mu_1(j(e), \ell(e)) && \text{if } \mu_1(j(e), \ell(e)) > 0, \\ \mu_2(e) &= \mu_0 && \text{if } j(e) \perp_d \ell(e) \mid D(e) [\mathcal{G}_k], \\ \mu_2(e) &= 0 && \text{if } \mu_1(j(e), \ell(e)) = 0 \text{ and } j(e) \not\perp_d \ell(e) \mid D(e) [\mathcal{G}_k]. \end{aligned} \quad (4.8)$$

Thus, $\mu_2(e) > 0$ if its conditioned set is an edge in at least one of the skeletons $\mathcal{G}_1^s, \dots, \mathcal{G}_k^s$. Since we can not ensure A2, and thus not use the directed local Markov property as in Theorem 4.6. We overcome this using *d-separation*. For $[j(e), \ell(e); D(e)]$, $e \in E_i$, $2 \leq i \leq d-1$, we check $j(e) \perp_d \ell(e) \mid D(e) [\mathcal{G}_k]$. To facilitate conditional independence, i. e. sparsity, for $e \in E_2$ we assign $\mu_0 > 0$. In the remainder, $\mu_0 := g(k)/2 = 1/2$ since $g(\cdot)$ is constant, i. e. it will not exceed the weight of an edge $[j(e), \ell(e); D(e)]$ with $j(e) \leftrightarrow \ell(e)$ in any of the DAGs $\mathcal{G}_1, \dots, \mathcal{G}_k$ as we want to model relationships in the DAGs prioritized. All other weights are zero and a maximum spanning tree algorithm is applied on E_2 . If an edge with weight μ_0 is chosen, we can directly set the independence copula by virtue of the d-separation, and thus, conditional independence. We repeat this for T_3, \dots, T_{d-1} . Since each pair of variables occurs exactly once as conditioned set in an R-vine, each weight μ_1 in \mathcal{H} is used exactly once. The actual truncation level k' is such that the R-vine trees $T_{k'+1}, \dots, T_{d-1}$ contain only the independence copula. The corresponding algorithm is given in Appendix A.3 and we will describe it using a subsequent toy example. Note that the d-separation can be easily checked for example by an implementation of the

Bayes-Ball algorithm, see e. g. Shachter (1998).

4.4.2 Toy Example

Example 4.17 (Heuristics for transformation). Consider the DAGs \mathcal{G}_k , $k = 1, 2, 3$, with at most k parents, see Figure 4.9, from left to right.

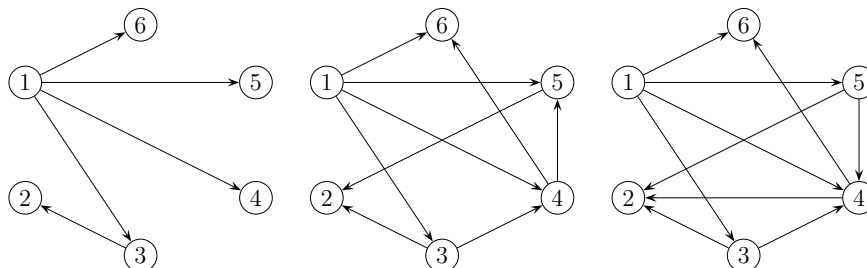


Figure 4.9: Example 4.17: DAGs \mathcal{G}_k for $k = 1, 2, 3$ with at most k parents (from left to right).

Applying a maximum spanning tree algorithm on \mathcal{H} to find the first R-vine tree T_1 , we obtain the skeleton \mathcal{G}_1^s , see Figure 4.10, first figure. This is however not in general the case. We sketch the intermediate step of building T_2 , where we already removed edges not allowed by the proximity condition and assigned weights according to Equation (4.8), see Figure 4.10, second to fourth figures.

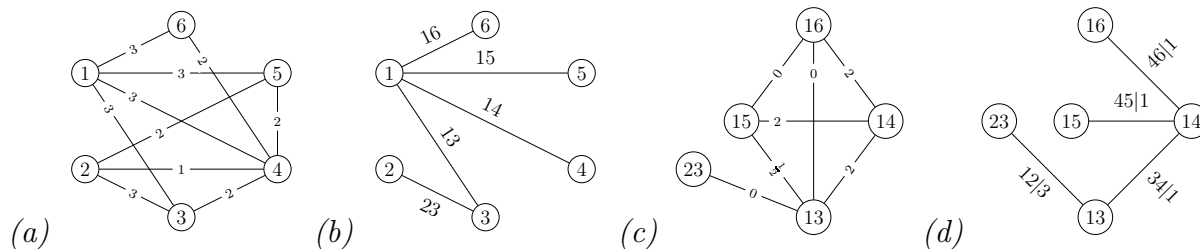


Figure 4.10: Example 4.17: Weighted graph \mathcal{H} with weight function $g(i) \equiv 1$ and $\mu_0 = \frac{g(1)}{2} = \frac{1}{2}$ (a) and R-vine representation of the DAG \mathcal{G}_3 with R-vine trees T_1 (b), intermediate step for building tree T_2 (c) and final T_2 (d). Note that $3 \perp\!\!\!\perp 5 \mid 1$ by the d-separation in \mathcal{G}_3 and hence the weight of the corresponding edge assigned is $\mu_0 = \frac{1}{2}$. However, this edge is not chosen by the maximum spanning tree algorithm. The corresponding

We see that T_3 has the form of a so called D-vine, i. e. the R-vine tree is a path. Thus, the structure of higher order trees T_4 and T_5 is already determined, see Figure 4.11.

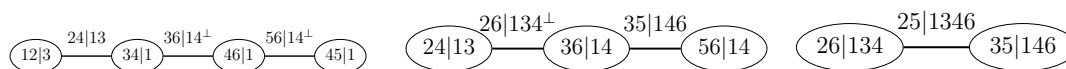


Figure 4.11: Example 4.17: R-vine representation of the DAG \mathcal{G}_3 . Trees T_3, T_4, T_5 (from left to right). Edges with superscript \perp are associated with the independence copula by the d-separation in \mathcal{G}_3 .

Based on the first R-vine tree T_1 and Corollary 4.15 we infer the lower bound for the truncation level. We consider the sets $V^v = \{v, \text{pa}(v)\}$ for $v \in V$ based on \mathcal{G}_3 . For example, the node 2 has the parents $\text{pa}(2) = \{3, 4, 5\}$ in \mathcal{G}_3 . Based on the first R-vine tree T_1 we check the lengths of shortest paths between 2 and its parents and obtain $\ell_2^3 = 1$, $\ell_2^4 = 3$ and $\ell_2^5 = 3$. By application of Corollary 4.15, this gives a lower bound for the truncation level $k' \geq 3$. The lengths of the shortest paths in T_1 for all nodes $v \in V$ can be found in Table 4.1.

v	$\text{pa}(v) = \{w_1^v, w_2^v, w_3^v\}$	$\ell_v^{w_1}, \ell_v^{w_2}, \ell_v^{w_3}$	$\max_{w \in \text{pa}(v)} \ell_v^w$
1	-	-	-
2	3,4,5	1,3,3	3
3	1	1	1
4	1,3,5	1,2,2	2
5	1	1	1
6	1,4	1,2	2

Table 4.1: Example 4.17: Shortest path distances in T_1 between nodes v and their parents $\text{pa}(v)$ in DAG \mathcal{G}_3 .

We obtain $k' = \max_{v \in V} \max_{w \in \text{pa}(v)} \ell_v^w = 3$. Note that this lower bound is not attained as we have the conditioned set $\{2, 5\}$ in the R-vine Tree T_5 which can not be represented by the independence copula as this conditioned set is associated to an edge in the DAG \mathcal{G}_3 . However, several edges with superscript \perp can be associated with the independence copula by the d -separation. The trees \mathcal{G}_1 and \mathcal{G}_2 are only used to obtain the weights for the corresponding trees, but not with respect to check for d -separation.

4.5 Application

In Brechmann and Czado (2013), the authors analyzed the Euro Stoxx 50 and collected time series of daily log returns of $d = 52$ major stocks and indices from May 22, 2006 to April 29, 2010 with $n = 985$ observations. For the chosen marginal models, see Brechmann and Czado (2013). To learn k -DAGs for $k = 1, \dots, 10$ from the z -scale data, we use the *Hill-Climbing* algorithm of the R-package `bnlearn`, see Scutari (2010) since it allows to limit the maximum number of parents as described in Section 3.3.5. These k -DAGs are shown in Appendix A.1. All pair copula families of the R-package `VineCopula` of Schepsmeier et al. (2017) were allowed. As laid out initially, the approach has two goals. The first was to find truncated R-vines related to Gaussian DAGs which overcome the restriction of Gaussian distributions. Thus, we compare the goodness of fit of the k -DAGs \mathcal{G}_k to their R-vine representations $\mathcal{W}(\mathcal{G}_k)$. If our approach represents the structure of the DAGs well and there is non-Gaussian dependence, the variety of pair copula families of an R-vine should improve the fit notably. Second, we want to check whether our approach can compete with Dißmann's algorithm. Using their algorithm, we calculate a sequence of t -truncated R-vines for $t = 1, \dots, 51$, using an level $\alpha = 0.05$ independence test. Hence, we consider three models in terms of number of parameters vs. log-likelihood and GIC.

Comparing the log-likelihood of DAGs and R-vines, we have to bear in mind that the

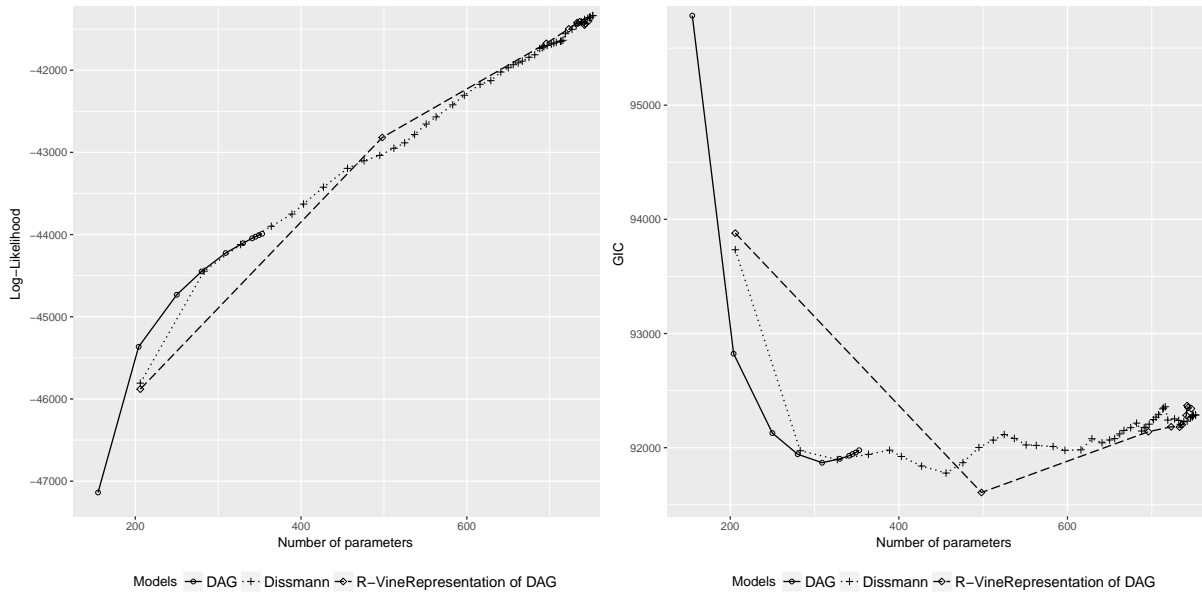


Figure 4.12: Comparison of k -DAGs \mathcal{G}_k , R-vine representations $\mathcal{W}(\mathcal{G}_k)$, $k = 1, \dots, 10$ and t -truncated Dißmann's algorithm, $t = 1, \dots, 51$ on z -scale, number of parameters vs. log-likelihood (left), number of parameters vs. GIC (right).

marginals in the DAG are assumed to be standard Gaussian and we also have to assume the same marginals for the R-vines, as done in e.g. Høbæk Haff et al. (2016). Yet, an advantage of vine copulas are independently chosen marginals. Thus, there is additional upside potential for the R-vine model. The results are given in Figure 4.12 and Tables A.1, A.2 in Appendix A.2. The DAG models have the least parameters which makes them sparse, but their goodness of fit falls behind the two competitors in case of non-Gaussian dependence. Comparing Dißmann's approach to our algorithm, we see a very similar behaviour when it comes to log-likelihood. However, the 2-DAG outperforms all other models in terms of GIC. The computation time for our algorithm ranges from 125 sec. for a 1-DAG to 270 sec. for a 10-DAG. Dißmann's algorithm needs more than 600 sec. for a first R-vine tree and up to 760 sec. for a full estimation. Thus, our approach is about three to five times faster. This is also what we inferred from the simulation study. Our approach is significantly faster, since given a specific edge $[j(e), \ell(e); D(e)]$, Dißmann's algorithm first carries out an independence test for the pair copula. If the hypothesis is rejected, a maximum likelihood fit of the pair copula is carried out. Our approach however checks $j(e) \perp\!\!\!\perp \ell(e) \mid D(e)$ based on the $j(e) \perp_d \ell(e) \mid D(e)$ $[\mathcal{G}_k]$, which is significantly faster once we know the DAG. The actual truncation levels k' of the R-vine representations are given in Table A.1. They are relatively high given the number of parents of these DAGs. However, this is because of very few non independence copulas in higher trees. For example, in the R-vine representation of the 2-DAG, T_{19}, \dots, T_{51} contain 56 non-independence copulas of 561 edges, i. e. about 10%, see also Figure 4.13. To visualize the actual truncation levels of the R-vine representations of a 2-DAG, consider the distribution of independence copulas. Thus, we plot a 52×52 matrix indicating which pair copulas

are the independence copula in the R-vine representation, see the lower triangular region of Figure 4.13. The upper triangular region encodes which pair copulas are set to the independence copula when we use an additional level $\alpha = 0.05$ independence test. Thus, each independence copula in the lower triangular is also in the upper triangular.

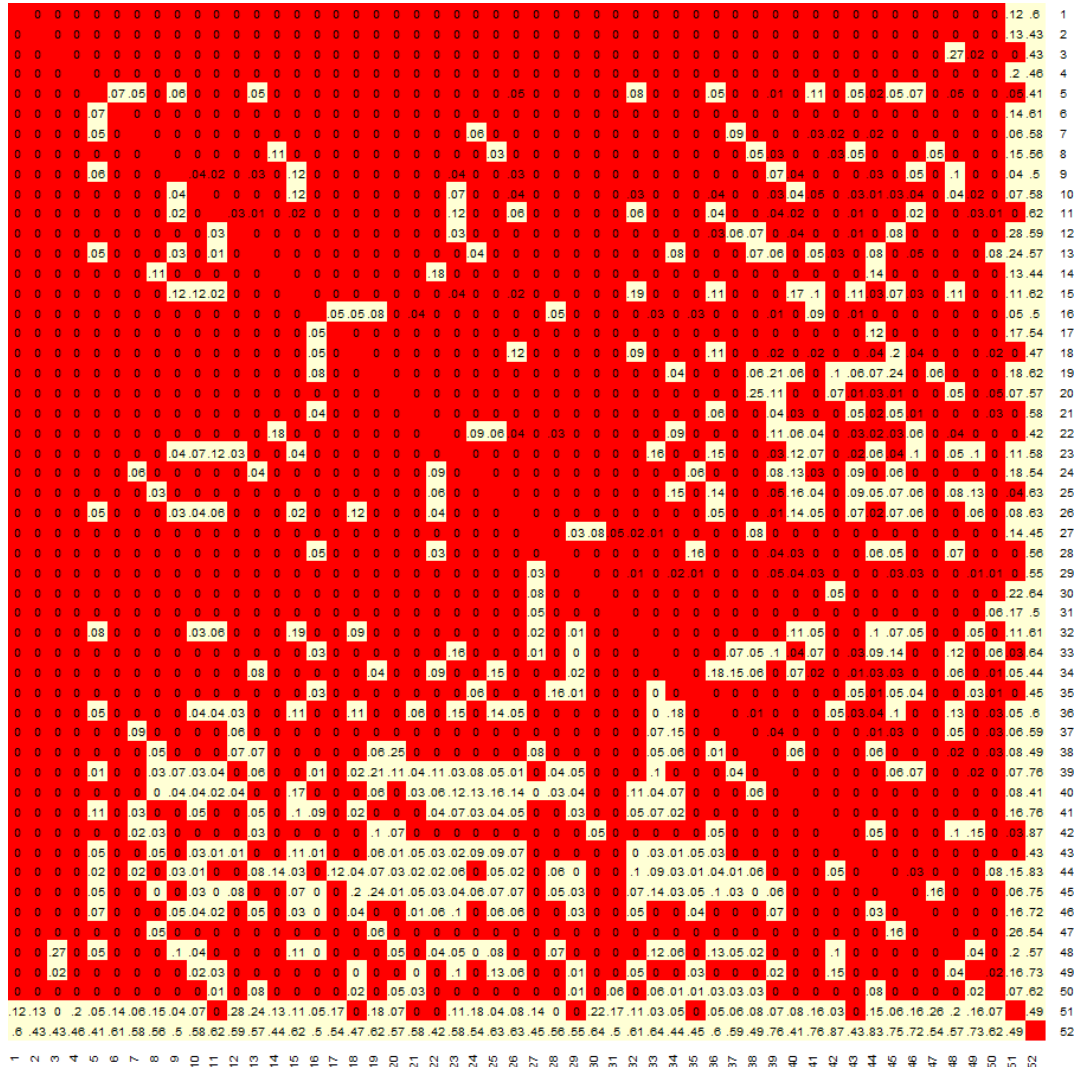


Figure 4.13: Distribution of independence copulas in the R-vine calculated by `RepresentDAGRVine` for a 2-DAG and absolute values of Kendall's τ of the corresponding pair copulas. Bright white color indicates a non independence copula whereas dark red color indicates an independence copula.

This sparsity pattern is not negatively influencing the computation times or GIC as demonstrated. It is also not intuitively apparent that a specific truncation level is more sensible to describe the data compared to a generally chosen sparse structure.

With this, the first approach to select vine copulas in high dimensions is presented. We remark that first, Gaussian DAG model help to fit parsimonious R-vine models with a sound theoretical reasoning. Second, we see that our algorithm works quite similarly to Dißmann's, i. e. we use a maximum spanning tree approach. The next approach, discussed in Chapter 5, will present a different view.

Chapter 5

Selection of Sparse Vine Copulas in High Dimensions with the Lasso

As in Chapter 4, we use the correspondence between Gaussian R-vines and their SEM expression, see Sections 3.4 and 4.2. There however, we used the SEMs only as intermediate step before final consideration of the corresponding DAGs which were also related to the SEMs. Now, the SEMs act as starting point for the R-vines. More precisely, we identify a set of linear equations with a parametrization for the correlation matrix in the same way as a Gaussian R-vine parametrizes the correlation matrix Σ of a multivariate Gaussian distribution. The intuition is now that each SEM equation contains both structural information about the R-vine structure and also information about non-independence copulas, identified by non-zero coefficients in the SEM. The idea is then to use a regularization approach to solve the SEM and set specific parameters exactly to zero, thus simplifying the structure of the corresponding R-vine and also imposing conditional independence. The regularization approach used here is the Lasso, see Section 3.2.

We will first show the correspondence between general SEMs and Gaussian R-vines, sketch how zero coefficients in the SEM can relate to independence copulas in the R-vine and then propose a method allowing us to shrink SEMs for regularization of R-vines. However, as in the previous chapter, the *proximity condition* as main ingredient for R-vine structures comes into play and must be taken into consideration for our approach. After having proposed a method to use the Lasso for structure estimation, we show how it can also be used to regulate for sparsity in the corresponding R-vine model. A data application in $d = 222$ dimensions concludes the chapter.

This chapter is a slightly altered and extended version of the submitted manuscript by Müller and Czado (2017c).

5.1 Motivation: SEM Representation of R-vines

Our approach connects the R-vine structure to *structural equation models (SEMs)*, see Section 3.4. For this, we utilize again the approach of Brechmann and Joe (2014), who give a representation of k -truncated Gaussian R-vines in terms of SEMs. Given a Gaussian R-vine with structure \mathcal{V} , we define a SEM corresponding to \mathcal{V} denoted by $\mathcal{S}(\mathcal{V})$. Let $\mathcal{V} = (T_1, \dots, T_{d-1})$ be an R-vine tree sequence and assume without loss of generality

$\{1, 2\} \in T_1$. For $j = 3, \dots, d$ denote the edges in T_1 by $\{j, \kappa_1(j)\}$ using an *assignment function* $\kappa_1(j)$, $j = 2, \dots, d$. For higher trees, we generalize κ_i for $i = 2, \dots, i-1$. Thus, the trees T_i contain edges $j, \kappa_i(j) | \kappa_1(j), \dots, \kappa_{i-1}(j) \in T_i$ for $i = 2, \dots, d-1$. Based on the R-vine structure V , define $\mathcal{S}(\mathcal{V})$ by

$$\begin{aligned} X_1 &= \psi_1 \epsilon_1, \\ X_2 &= \varphi_{2,1} X_1 + \psi_2 \epsilon_2, \\ X_j &= \sum_{i=1}^{j-1} \varphi_{j, \kappa_i(j)} X_{\kappa_i(j)} + \psi_j \epsilon_j, \quad j = 3, \dots, d, \end{aligned} \tag{5.1}$$

with $\epsilon_j \sim \mathcal{N}(0, 1)$ i.i.d. and ψ_j such that $\text{Var}(X_j) = 1$ for $j = 1, \dots, d$. Additionally, ϵ_j is independent of X_1, \dots, X_{j-1} , for $j = 2, \dots, d$. Brechmann and Joe (2014) assume a k -truncated R-vine and restate the SEM in (5.1) with

$$X_j = \sum_{i=1}^{\max(j-1, k)} \varphi_{j, \kappa_i(j)} X_{\kappa_i(j)} + \psi_j \epsilon_j, \quad j = 3, \dots, d.$$

Thus, we have for each edge $e \in E_i$, $i = k+1, \dots, d-1$ that for $j = 3, \dots, d$:

$$c_{j(e), \kappa_i(j(e)); \kappa_1(j(e)), \dots, \kappa_{i-1}(j(e))} \equiv 1 \Rightarrow \varphi_{j(e), \kappa_i(j(e))} = 0. \tag{5.2}$$

The first step to generalize this implication is that we not only allow for a specific truncation level $k = 1, \dots, d-1$. Furthermore, we want to set specific regression coefficients $\varphi_{j, \kappa_i(j)}$ to zero, also for $i < k$. Additionally, we generalize the ordering of the equations from first to last using an ordering function $\eta : \{1, \dots, d\} \rightarrow \{1, \dots, d\}$. Thus, $X_{\eta(j)}$ is on the left hand side of the j -th equation and has at most j right hand summands, including the error term, i. e. we obtain a triangular structure. We rewrite (5.1) as

$$\begin{aligned} X_{\eta(1)} &= \psi_{\eta(1)} \epsilon_{\eta(1)}, \\ X_{\eta(2)} &= \varphi_{\eta(2), \kappa_1(\eta(2))} X_{\kappa_1(\eta(2))} + \psi_{\eta(2)} \epsilon_{\eta(2)}, \\ X_{\eta(j)} &= \sum_{i=1}^{j-1} \varphi_{\eta(j), \kappa_i(\eta(j))} X_{\kappa_i(\eta(j))} + \psi_{\eta(j)} \epsilon_{\eta(j)}, \quad j = 3, \dots, d. \end{aligned} \tag{5.3}$$

We define some additional terminology to deal with zero regression coefficients.

Definition 5.1 (SEM regressor sets). *Consider a SEM as in (5.3) with ordering function η . Then, $X_{\eta(j)}$ has at most $j-1$ potential regressors $\kappa_i(\eta(j))$ for $i = 1, \dots, j-1$. We define the set of potential regressors of $X_{\eta(j)}$ by $\mathcal{R}(\eta(j)) = \{\eta(1), \dots, \eta(j-1)\}$, i. e. the left hand side indices of the previous $j-1$ structural equations. Define the set $\mathcal{R}_1(\eta(j)) = \{\kappa_i(\eta(j)), i = 1, \dots, j-1 : \varphi_{\eta(j), \kappa_i(\eta(j))} \neq 0\}$, the set of actual regressors of $X_{\eta(j)}$. $\mathcal{R}_0(\eta(j)) = \mathcal{R}(\eta(j)) \setminus \mathcal{R}_1(\eta(j))$ is the set of unused regressors.*

We visualize the concepts in the following example. Recall that j refers to the j -th row in the SEM and $\eta(j)$ to the corresponding left hand side index of the j -th row.

Example 5.2 (Example 2.5 cont.). *Following our previous example, the R-vine matrix M gives rise to the following values of the ordering function η and the assignment function κ . Considering η , we have the main diagonal $\text{diag}(M) = (4, 5, 3, 6, 2, 1) = (\eta(d), \dots, \eta(1))$, see also Table 5.1, left two columns. Since R-vine matrices are most often denoted as lower-diagonal matrices in the literature, we have $m_{1,1} = \eta(d), \dots, m_{d,d} = \eta(1)$. The values of the assignment function κ can be read column-wise from M . For example, consider $M_{\cdot,j}$, the j -th column of M with $m_{j,j} = \eta(d - j + 1)$. Then, $\kappa_1(\eta(d - j + 1)) = m_{d,j}$ and $\kappa_2(\eta(d - j + 1)) = m_{d-1,j}$. Generally, we obtain for $i = 1, \dots, d - j$:*

$$\kappa_i(\eta(d - j + 1)) = m_{d-i+1,j}, \quad j = 1, \dots, d - 2.$$

The values of κ can also be written in tabular form, see Table 5.1. The i -th row of this table corresponds to column $d - i + 1$ of the R-vine matrix M . For example, consider the first column of M , i. e. $j = 1$ with $m_{j,j} = \eta(6 - 1 + 1) = \eta(6) = 4$, according to Table 5.1. Correspondingly $\kappa_1(\eta(6 - 1 + 1)) = \kappa_1(\eta(6)) = \kappa_1(4) = 5 = m_{6,1}$, see also Table 5.1.

j	$\eta(j)$	$\kappa_1(\eta(j))$	$\kappa_2(\eta(j))$	$\kappa_3(\eta(j))$	$\kappa_4(\eta(j))$	$\kappa_5(\eta(j))$
1	1	-	-	-	-	-
2	2	$1 = m_{6,5}$	-	-	-	-
3	6	$2 = m_{6,4}$	$1 = m_{5,4}$	-	-	-
4	3	$6 = m_{6,3}$	$2 = m_{5,3}$	$1 = m_{4,3}$	-	-
5	5	$2 = m_{6,2}$	$6 = m_{5,2}$	$3 = m_{4,2}$	$1 = m_{3,2}$	-
6	4	$5 = m_{6,1}$	$2 = m_{5,1}$	$6 = m_{4,1}$	$3 = m_{3,1}$	$1 = m_{2,1}$

Table 5.1: Example 2.5: Inverse of ordering function η and assignment function κ .

The R-vine Matrix M is given by

$$\begin{pmatrix} m_{1,1} & & & & & & \\ m_{2,1} & m_{2,2} & & & & & \\ m_{3,1} & m_{3,2} & m_{3,3} & & & & \\ m_{4,1} & m_{4,2} & m_{4,3} & m_{4,4} & & & \\ m_{5,1} & m_{5,2} & m_{5,3} & m_{5,4} & m_{5,5} & & \\ m_{6,1} & m_{6,2} & m_{6,3} & m_{6,4} & m_{6,5} & m_{6,6} & \end{pmatrix} = \begin{pmatrix} 4 & & & & & & \\ 1 & 5 & & & & & \\ 3 & 1 & 3 & & & & \\ 6 & 3 & 1 & 6 & & & \\ 2 & 6 & 2 & 1 & 2 & & \\ 5 & 2 & 6 & 2 & 1 & 1 & \end{pmatrix}$$

We now want to evaluate the correspondence between independence copulas in the R-vine and zero coefficients in the SEM. Assume the following lower triangular family matrix $\Gamma = (\gamma_{i,j})_{i=1, \dots, d; j=1, \dots, d}$ with 0 representing independence and 1 indicating a Gaussian copula.

$$\Gamma = \begin{pmatrix} - & & & & & & \\ \gamma_{2,1} & - & & & & & \\ \gamma_{3,1} & \gamma_{3,2} & - & & & & \\ \gamma_{4,1} & \gamma_{4,2} & \gamma_{4,3} & - & & & \\ \gamma_{5,1} & \gamma_{5,2} & \gamma_{5,3} & \gamma_{5,4} & - & & \\ \gamma_{6,1} & \gamma_{6,2} & \gamma_{6,3} & \gamma_{6,4} & \gamma_{6,5} & - & \end{pmatrix} = \begin{pmatrix} - & & & & & & \\ 0 & - & & & & & \\ 1 & 1 & - & & & & \\ 0 & 0 & 1 & - & & & \\ 1 & 0 & 0 & 1 & - & & \\ 1 & 1 & 1 & 1 & 1 & - & \end{pmatrix}$$

The zeros in the family matrix Γ , i. e. independence copulas, are reflected by zero coefficients in the SEM. For $j = 1, \dots, d-2$ and $i = 1, \dots, d-j$ we have

$$\gamma_{d-i+1,j} = 0 \Rightarrow \varphi_{\eta(d-j+1), \kappa_i(\eta(d-j+1))} = 0.$$

We emphasize that only the parameter value φ is set to zero. The assignment function κ is unchanged since it is necessary to determine a valid R-vine structure. This way, we impose independence, i. e. sparsity in the R-vine which is reflected by the corresponding SEM. We now illustrate how this choice affects \mathcal{R} , \mathcal{R}_0 and \mathcal{R}_1 .

$$\begin{aligned} X_1 &= \psi_1 \epsilon_1, \\ X_2 &= \varphi_{2,1} X_1 + \psi_2 \epsilon_2, \\ X_6 &= \varphi_{6,2} X_2 + \varphi_{6,1} X_1 + \psi_6 \epsilon_6, \\ X_3 &= \varphi_{3,6} X_6 + \varphi_{3,1} X_1 + \psi_3 \epsilon_3, \\ X_5 &= \varphi_{5,2} X_2 + \varphi_{5,1} X_1 + \psi_5 \epsilon_5, \\ X_4 &= \varphi_{4,5} X_5 + \varphi_{4,2} X_2 + \varphi_{4,3} X_3 + \psi_4 \epsilon_4. \end{aligned} \tag{5.4}$$

$\eta(j)$	$\mathcal{R}(\eta(j))$	$\mathcal{R}_1(\eta(j))$	$\mathcal{R}_0(\eta(j))$
1	\emptyset	\emptyset	\emptyset
2	$\{1\}$	$\{1\}$	\emptyset
6	$\{2, 1\}$	$\{2, 1\}$	\emptyset
3	$\{6, 2, 1\}$	$\{6, 1\}$	$\{2\}$
5	$\{2, 6, 3, 1\}$	$\{2, 1\}$	$\{6, 3\}$
4	$\{5, 2, 6, 3, 1\}$	$\{5, 2, 3\}$	$\{6, 1\}$

Table 5.2: Example 2.5: Sets \mathcal{R} , \mathcal{R}_1 , \mathcal{R}_0 .

In other words, the non-zero coefficients in the SEM (5.4) are drawn from the corresponding columns of the R-vine structure matrix M where the family matrix Γ is non-zero. Consider an arbitrary column $j = 1, \dots, 5$ in the matrix M . The non-zero entries $(\gamma_{d,j}, \dots, \gamma_{j+1,j})$ correspond to $\mathcal{R}_1(\eta(d-j+1))$. For example, if we consider again the first column of M , $M_{\cdot,j}$ for $j = 1$ and $(\gamma_{6,1}, \dots, \gamma_{2,1}) = (1, 1, 0, 1, 0)$. Using this vector to obtain the non-zero entries from the R-vine structure matrix M , we have the first column $M_{d:2,1} = (5, 2, 6, 3, 1)$ and thus the non-zero entries $(5, 2, 3)$ as in (5.4) and Table 5.2 for $\mathcal{R}_1(\eta(d-1+1)) = \mathcal{R}_1(\eta(6)) = \mathcal{R}_1(4)$.

Having characterized the connection between R-vines and SEMs, our goal is now to find an inverse transformation. More precisely, given high dimensional data, we want estimate a SEM where many of the coefficients are zero. For simplicity, assume $\eta(j) = j$ for $j = 1, \dots, d$. For each structural equation, we obtain a set $\mathcal{R}_0(j)$ with $|\mathcal{R}_0(j)| > 0$. This leaves us with a sparse SEM as in (5.1),

$$\begin{aligned} X_1 &= \psi_1 \epsilon_1, \\ X_2 &= \varphi_{2,1} X_1 + \psi_2 \epsilon_2, \\ X_j &= \sum_{i \in \mathcal{R}_1(j)} \varphi_{j, \kappa_i(j)} X_{\kappa_i(j)} + \psi_j \epsilon_j, \quad j = 3, \dots, d. \end{aligned}$$

Under additional assumptions to be specified later on, this SEM can also be written as an R-vine with structure matrix \hat{M} and family matrix $\hat{\Gamma}$. Because of the zero coefficients $\mathcal{R}_0(j)$, entries in the family matrix $\hat{\Gamma}$ can be set to zero, i. e. representing the independence copula. This means, we want to generalize the implication (5.2) in such a way that we have for each edge $e \in E_i$, $i = 1, \dots, d-1$ and $j = 3, \dots, d$:

$$\varphi_{j, \kappa_i(j)} = 0 \Rightarrow c_{j(e), \kappa_i(j(e)); \kappa_1(j(e)), \dots, \kappa_{i-1}(j(e))} = 1. \quad (5.5)$$

Thus, we obtain a sparse R-vine model. This model is not restricted to a joint Gaussian probability distribution as our SEM is. We can estimate the marginal distributions entirely independent of the dependence behaviour and use vast numbers of parametric and non-parametric pair copulas to describe the joint distribution. To describe more precisely what is motivated by (5.5), we now introduce an R-vine representation of SEM.

Definition 5.3 (R-vine representation of a SEM). *Consider a SEM \mathcal{S} in d dimensions, where we assume without loss of generality $\eta(j) \equiv j$ for $j = 1, \dots, d$.*

$$\begin{aligned} X_1 &= \psi_1 \epsilon_1, \\ X_2 &= \varphi_{2,1} X_1 + \psi_2 \epsilon_2, \\ X_j &= \sum_{i=1}^{j-1} \varphi_{j, \kappa_i(j)} X_{\kappa_i(j)} + \psi_j \epsilon_j. \end{aligned} \quad (5.6)$$

The SEM (5.6) has an R-vine representation $\mathcal{W}(\mathcal{S})$ with R-vine tree sequence (T_1, \dots, T_{d-1}) if for $j = 2, \dots, d$ and $i = 1, \dots, j-1$ we have

$$j, \kappa_i(j) | \kappa_1(j), \dots, \kappa_{j-1}(i) \in T_j.$$

To put it in a nutshell, the j -th row of the SEM corresponds to column $d-j+1$ of the R-vine matrix for $j = 1, \dots, d$. This definition connects SEMs and R-vines. Based on this, we can consider setting specific regressors in the SEM to zero to obtain a sparse R-vine model. We note two caveats of this approach. First, of all, not every SEM with specific coefficients set to zero reflects a R-vine structure, since the proximity condition has to hold for the R-vine structure. Second, a SEM does not necessarily determine the

R-vine structure uniquely. We give examples for these assertions and move on to sketch the general approach.

Example 5.4 (SEM without R-vine representation). *Consider the following SEM in five dimensions.*

$$\begin{aligned} X_1 &= \psi_1 \epsilon_1 \\ X_2 &= \varphi_{2,1} X_1 + \psi_2 \epsilon_2 \\ X_3 &= \varphi_{3,1} X_1 + \varphi_{3,2} X_2 + \psi_3 \epsilon_3 \\ X_4 &= \varphi_{4,1} X_1 + \varphi_{4,2} X_2 + \psi_4 \epsilon_4 \\ X_5 &= \varphi_{5,3} X_3 + \varphi_{5,4} X_4 + \psi_5 \epsilon_5 \end{aligned}$$

If we now want to find a representing R-vine structure, the R-vine trees T_1 and T_2 must have edges in terms of the assignment function κ as we saw from Definition 5.3. Since we have at most two right hand side summands, we need to find values for $\kappa_i(j)$ for $j = 3, 4, 5$ and $i = 1, 2$ such that the following holds:

$$\begin{aligned} i, \kappa_1(i) &\in T_1, \\ i, \kappa_2(i) | \kappa_1(i) &\in T_2. \end{aligned}$$

Assume that the following edges are chosen in the first tree T_1 :

$$\{\{2, \kappa_1(2)\}, \{3, \kappa_1(3)\}, \{4, \kappa_1(4)\}, \{5, \kappa_1(5)\}\} = \{\{2, 1\}, \{3, 1\}, \{4, 1\}, \{5, 3\}\} \in E_1.$$

Now, we can not set $\kappa_2(5) = 4$ to obtain $5, 4 | 3 \in T_2$ as required. This is since $5, 4 | 3 = \{\{5, 3\}, \{4, 3\}\}$, but $\{3, 4\} \notin T_1$. Note additionally that we can not have more than four edges in T_1 , since otherwise, it would not be a tree. If $\kappa_1(5) = 4$, it can similarly be shown that the same argument holds.

Next, we show an example of that two R-vines with identical SEM representations.

Example 5.5 (Different 2-truncated R-vines with identical SEM representation in four dimensions). *Consider the following two 2-truncated R-vines and their SEM representations.*

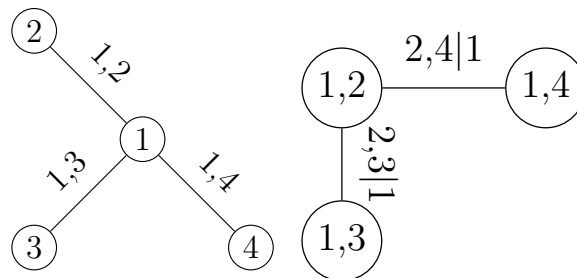


Figure 5.1: Example 5.5: R-vine structure \mathcal{V}_1 , corresponding to SEM $\mathcal{S}(\mathcal{V}_1)$ in (5.7).

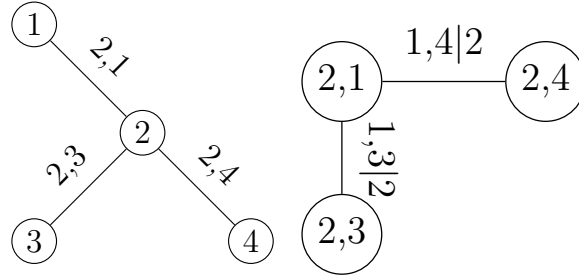


Figure 5.2: Example 5.5: R-vine structure \mathcal{V}_2 , corresponding to SEM $\mathcal{S}(\mathcal{V}_2)$ in (5.8).

Both have identical SEM representations, i. e. only looking at the corresponding equations without knowing exactly the assignment function κ and thus, which regressor belongs to which R-vine tree, we are not able to distinguish between those two SEMs.

$$\begin{aligned}
 X_1 &= \psi_1 \epsilon_1, \\
 X_2 &= \varphi_{2,1} X_1 + \psi_2 \epsilon_2, \\
 X_3 &= \varphi_{3,1} X_1 + \varphi_{3,2} X_2 + \psi_3 \epsilon_3, \\
 X_4 &= \varphi_{4,1} X_1 + \varphi_{4,2} X_2 + \psi_4 \epsilon_4.
 \end{aligned} \tag{5.7}$$

$$\begin{aligned}
 X_1 &= \psi_1 \epsilon_1, \\
 X_2 &= \varphi_{2,1} X_1 + \psi_2 \epsilon_2, \\
 X_3 &= \varphi_{3,1} X_2 + \varphi_{3,2} X_1 + \psi_3 \epsilon_3, \\
 X_4 &= \varphi_{4,1} X_2 + \varphi_{4,2} X_1 + \psi_4 \epsilon_4.
 \end{aligned} \tag{5.8}$$

We recognize the same identification issues here as in Example 4.2.

We will develop an approach which overcomes the restrictions sketched in the Examples 5.4 and 5.5. First, we will need to determine the R-vine structure based on the assignment function κ before we consider the sets of zero coefficients. The method we are going to use for this is the Lasso, introduced in Section 3.2.

5.2 Vine Copula Structure Selection with the Lasso

This part now focusses on the first part of our agenda, the identification of a suitable R-vine structure. We will do this using the Lasso and the SEM representation of R-vines. Subsequently, we will also use the Lasso for the second main part, setting numerous pair copulas to the independence copula.

Finding a suitable R-vine from a SEM decomposes in two parts. First, we have to find the ordering function η introduced in the previous section. This resembles the main diagonal of the corresponding R-vine matrix. Next, we have to find the actual entries for the R-vine matrix or, in the SEM view, the regressors.

5.2.1 Calculation of the Ordering Function

The first two of the three methods we propose for calculating the ordering function η are based on empirical dependence measures. The third takes directly into account the Lasso.

Maximum linear and rank correlation

Definition 5.6 (Maximum linear correlation ordering). *Let $\hat{\Sigma} = (\hat{\rho}_{i,j})_{i=1,\dots,d;j=1,\dots,d}$ be the empirical correlation matrix of the data on the z -scale. Calculate the row sums of the absolute value of $\hat{\Sigma}$ and order the results decreasingly to obtain η . In other words, $\eta_{\Sigma}(i)$ is a permutation of $i = 1, \dots, d$ such that*

$$\sum_{j=1}^d \left| \hat{\rho}_{\eta_{\Sigma}^{-1}(1),j} \right| \leq \dots \leq \sum_{j=1}^d \left| \hat{\rho}_{\eta_{\Sigma}^{-1}(d),j} \right|.$$

The intuition behind this approach is to put the variables which have most linear dependence with the remaining variables in the first equations such that they can act as regressors for variables in later equations of the SEM. Since we ultimately want to work with non-linear and non-Gaussian dependence, we can also use a rank based measure like Kendall's τ in the same manner.

Definition 5.7 (Maximum Kendall's τ ordering). *Let $\hat{Y} = (\hat{\tau}_{i,j})_{i=1,\dots,d;j=1,\dots,d}$ be the empirical Kendall's τ matrix of the data on the x -scale. Calculate the row sums of the absolute value of \hat{Y} and order the results decreasingly to obtain η . In other words, $\eta_{\tau}(i)$ is a permutation of $i = 1, \dots, d$ such that*

$$\sum_{j=1}^d \left| \hat{\tau}_{\eta_{\tau}^{-1}(1),j} \right| \leq \dots \leq \sum_{j=1}^d \left| \hat{\tau}_{\eta_{\tau}^{-1}(d),j} \right|.$$

The motivation is equivalent to in definition 5.6. Considering Kendall's τ as dependence measure on rank data is sensible as finally, copulas often are estimated on the empirical ranks of the data. We now present a third approach which directly involves the application of the Lasso and the later development of the SEMs.

Maximum Lasso regressor approach

Assume for the moment we already have found an ordering η and that it coincides with the ordering of the variables, i. e. $\eta(j) = j$ for $j = 1, \dots, d$. A SEM in the form of (5.1), X_j can have regressors X_i for $i < j$, based on our model assumption. Thus, if we compute solutions for the d equations

$$X_j = \sum_{i=1, i \neq j}^d \beta_{i,j} X_i + \psi_j \epsilon_j, \quad j = 1, \dots, d,$$

we end up with a list of regression coefficients for each X_j , $j = 1, \dots, d$. Moreover, if we solve these equations with the Lasso and some suitably chosen $\lambda \geq 0$, specific regression coefficients are set to zero. Considering all equations, some X_i will occur more often with non-zero coefficients than others. Based on the SEM structure we have, it is beneficial to assign the regressors which occur often a low value of the ordering function η . In a SEM with such a structure, these X_i which occurred often as regressors can then be chosen as regressors by the assignment function κ for many SEM rows.

Definition 5.8 (Lasso Ordering). *Consider n samples from $\mathbf{X} = (X_1, \dots, X_d) \in \mathbb{R}^d$ and let $B \in \mathbb{R}^{(d-1) \times d}$ with the columns β_j , $j = 1, \dots, d$, such that we have $\beta_j = (\beta_{j,1}, \dots, \beta_{j,j-1}, \beta_{j,j+1}, \dots, \beta_{j,d})$ are the Lasso solutions to the d minimization problems*

$$\min_{\beta_j \in \mathbb{R}^{d-1}} \left(\frac{1}{2n} \sum_{k=1}^n \left(x_{k,j} - \sum_{\ell=1, \ell \neq j}^d \beta_{j,\ell} x_{k,\ell} \right)^2 + \lambda_j \sum_{\ell=1, \ell \neq j}^d |\beta_{j,\ell}| \right).$$

For each possible regressor $j = 1, \dots, d$, calculate the number of $\beta_{j,\ell} = 0$ over all ℓ and assign the ones with highest occurrence the lowest number in the ordering function η_L . More precisely,

$$\sum_{\ell=1}^d \mathbb{1}_{\{\beta_{\eta_L(1),\ell} \neq 0\}} \leq \dots \leq \sum_{\ell=1}^d \mathbb{1}_{\{\beta_{\eta_L(d),\ell} \neq 0\}}$$

The corresponding λ_ℓ are calculated via k -fold cross-validation. In case of ties, i. e. two or more variables are occurring equally often as regressors for the remaining variables, we choose the ordering of these variables randomly.

The intuition is similar to a method proposed by Meinshausen and Bühlmann (2006) to find undirected graphical models. They use the Lasso to find neighbourhoods of nodes which are exactly the non-zero coefficient regressors calculated by the Lasso. We give a brief numerical example.

Example 5.9 (Calculation of ordering function η). *We consider the `worldindices` data set, included in the `CDvine` package of Brechmann and Schepsmeier (2013) comprising $d = 6$ variables with $n = 396$ observations on the u -scale. We transform our observations to the z -scale using the Gaussian quantile function. For the maximum linear correlation approach we have Table 5.3.*

	\hat{GSPC}	$\hat{N225}$	\hat{SSEC}	\hat{GDAXI}	\hat{FCHI}	\hat{FTSE}	row sum	$\eta_\Sigma(i)$
\hat{GSPC}	1.00	0.20	0.18	0.75	0.74	0.73	3.60	4
$\hat{N225}$	0.20	1.00	0.33	0.26	0.31	0.28	2.39	5
\hat{SSEC}	0.18	0.33	1.00	0.19	0.21	0.23	2.14	6
\hat{GDAXI}	0.75	0.26	0.19	1.00	0.96	0.91	4.07	3
\hat{FCHI}	0.74	0.31	0.21	0.96	1.00	0.94	4.18	1
\hat{FTSE}	0.73	0.28	0.23	0.91	0.94	1.00	4.09	2

Table 5.3: Example 5.9: Empirical correlation matrix $\hat{\Sigma}$ on z -scale, row sums and ordering function η_Σ based on $\hat{\Sigma}$.

For the maximum rank correlation approach we obtain the ordering as given in Table 5.4.

	\hat{GSPC}	$\hat{N225}$	\hat{SSEC}	\hat{GDAXI}	\hat{FCHI}	\hat{FTSE}	row sum	$\eta_\tau(i)$
\hat{GSPC}	1.00	0.10	0.10	0.50	0.50	0.49	2.70	4
$\hat{N225}$	0.10	1.00	0.20	0.16	0.18	0.16	1.81	5
\hat{SSEC}	0.10	0.20	1.00	0.12	0.14	0.15	1.71	6
\hat{GDAXI}	0.50	0.16	0.12	1.00	0.82	0.73	3.34	2
\hat{FCHI}	0.50	0.18	0.14	0.82	1.00	0.77	3.42	1
\hat{FTSE}	0.49	0.16	0.15	0.73	0.77	1.00	3.30	3

Table 5.4: Example 5.9: Empirical Kendall's tau matrix $\hat{\Upsilon}$ on z -scale, row sums and ordering function η_τ based on $\hat{\Upsilon}$.

For the maximum Lasso regressor approach, we calculate Lasso regression coefficients of Z_j on Z_{-j} for $j = 1, \dots, 6$, see Table 5.5. Of course, the number of non-zero regression coefficients depends on the choice of the penalization coefficient λ_j for each regression on Z_j . Experiments showed that it is feasible to choose λ_j according to k -fold cross-validation.

variable	id j	# occurrence	$\eta_L(j)$	λ_j
\hat{GSPC}	1	2	4	0.170
$\hat{N225}$	2	2	5	0.129
\hat{SSEC}	3	1	6	0.171
\hat{GDAXI}	4	3	3	0.065
\hat{FCHI}	5	4	1	0.049
\hat{FTSE}	6	4	2	0.053

Table 5.5: Example 5.9: Variable name, id j , number of occurrence as regressors, ordering function η_L based on maximum Lasso Ordering and 5-fold cross-validated λ_j .

If two or more variables have the same number of occurrences as regressors for other variables, we choose randomly to determine a unique ordering. If one or more variables do not occur as regressors at all, we assign them the last ranks and break ties by choosing randomly.

It is to be noted that the first two heuristics can be failed using simple counterexamples. Hence, we will always use the *maximum Lasso regressor approach* in the remainder. To provide an intuition for the ordering, recall the correspondence between SEMs and DAGs illustrated in the Examples 3.18 and 3.19. Hence, specifying the ordering of an SEM is equivalent to specifying the topological ordering of a DAG. Since cycles must be omitted in a DAG, it is sensible to give nodes which are directly influencing other nodes a low order, such that they can act as parents of these nodes. This concludes the part considering the SEM ordering function η .

5.2.2 Sparse R-vine Structure Selection

Knowing the ordering function η , we can write a SEM as in (5.1). Assume for notational convenience that the ordering $1, \dots, d$ already reflects the ordering η as chosen in Section 5.2.1, i. e. $\eta(j) \equiv j$. The first two equations of the SEM are trivially described. However, we can not directly use the Lasso to solve the $d - 2$ later SEM equations stepwise or simultaneously. If we do, we might end up with non zero coefficients, which cannot be translated into a valid R-vine matrix as in Example 5.2. It is much more likely that we obtain a sparse SEM as in Example 5.4, which does not have a representation as R-vine in the sense of Definition 5.3 because of the restrictions imposed by the proximity condition. Additionally, we have to keep in mind that the solution to our SEM is also dependent on the choice of the penalization parameter λ . Thus, for different values of λ , different R-vine representations with different levels of sparsity result. We will now present an approach which computes an R-vine structure matrix M . We consider the first R-vine tree and all higher order trees separately.

Selection of the First R-vine Tree

Let M be a $d \times d$ matrix with $\text{diag}(M) = (m_{1,1}, \dots, m_{d,d})$. To obtain a valid R-vine matrix, we trivially set the entry $m_{d,d-1} = m_{d,d}$ and we are left to determine Lasso regularization paths for the remaining $d - 2$ columns of M . Thus, we have the regression problems for $j = 3, \dots, d$:

$$\min_{\varphi \in \mathbb{R}^{j-1}} \left(\frac{1}{2n} \sum_{k=1}^n \left(x_{i,j} - \sum_{\ell=1}^{j-1} \varphi_{j,\ell} x_{k,\ell} \right)^2 + \lambda_j \sum_{\ell=1}^{j-1} |\varphi_{j,\ell}| \right), \quad (5.9)$$

and denote the solutions as $\hat{\varphi}_j^\lambda = (\hat{\varphi}_{j,1}^\lambda, \dots, \hat{\varphi}_{j,j-1}^\lambda) \in \mathbb{R}^{j-1}$. To formalize how we process these solutions, recall the definition of the regularization path by the set Λ , returning the non-zero coefficients in the regression of X_j for each value of $\lambda > 0$:

$$\Lambda(\lambda, j) = \{ \ell : \hat{\varphi}_{j,\ell}^\lambda \neq 0 \text{ in } \hat{\varphi}_j^\lambda \}, \text{ with } k(\lambda, j) = |\Lambda(\lambda, j)|.$$

Clearly, $k(\lambda_1, j) \geq k(\lambda_2, j)$ for $\lambda_1 \leq \lambda_2$. If $\varphi_1 \in \Lambda(\lambda_1, j)$ and $\varphi_2 \in \Lambda(\lambda_1, j)$ but $\varphi_1 \in \Lambda(\lambda_2, j)$ and $\varphi_2 \notin \Lambda(\lambda_2, j)$ for $\lambda_1 < \lambda_2$, we say $\varphi_1 \succ \varphi_2$. This terminology is necessary to obtain an ordering on the set $\Lambda(\lambda, j)$. It is motivated by the fact that we want to obtain the coefficients which are non-zero for the largest penalization values of λ . Thus, assume we have two coefficients for the problem (5.9), $\hat{\varphi}_1 = \hat{\varphi}_2 = 0$ for some $\lambda > 0$. Now, letting $\lambda \rightarrow 0$, both coefficients will become non-zero in the end, as the penalization shrinks to zero. However, if there exists a $\lambda' > 0$ such that (5.9) is solved with $\lambda_j = \lambda'$ and we obtain $\hat{\varphi}_1 \neq 0$ but $\hat{\varphi}_2 = 0$, we consider φ_1 the more important coefficient and denote $\varphi_1 \succ \varphi_2$. The set $\Lambda(\lambda, j)$ contains all non-zero regressors for the penalization value λ of the regression problem (5.9), ordered according to their first non-zero occurrence, i. e. the regularization path. In the case of two or more $\hat{\varphi}_j$ which are simultaneously non-zero on the regularization path, we take the one with the highest absolute value of the coefficient

the first R-vine tree T_1 , encoded by the d -th row of the partial R-vine matrix M' .

$$M' = \begin{pmatrix} 3 & & & & & & \\ & 2 & & & & & \\ & & 1 & & & & \\ & & & 4 & & & \\ \square & & & & 6 & 6 & \\ 2 & 5 & 4 & 5 & 5 & 5 & \end{pmatrix} \quad M = \begin{pmatrix} 3 & & & & & & \\ 6 & 2 & & & & & \\ 1 & 6 & 1 & & & & \\ 4 & 1 & 6 & 4 & & & \\ 5 & 4 & 5 & 6 & 6 & & \\ 2 & 5 & 4 & 5 & 5 & 5 & \end{pmatrix}$$

We need to determine the second tree, i. e. M'_{d-1} . First, we note that $m_{5,4} = 6$ is the only valid choice. For the general case, consider the missing entry $m_{5,1}$, marked by \square . First, we check whether the second entry in the regularization path, $\Lambda(0,3)_2 = 6$ is valid. By checking the proximity condition (2.8), this is not the case as 2 and 6 are not connected in T_1 . Thus, we recompute the regularization path such that $\Lambda(0,3)_1 = m_{6,1} = 2$ and $\Lambda(0,3)_2$ adheres to the proximity condition. The set of possible regressors are the entries on the main diagonal to the right of the first column 2, 1, 4, 6, 5, where 2 is already occurring. This leaves us with 1, 4, 6, 5. From these, only 5 is a possible entry according to the proximity condition. Thus, the remaining 1, 4, 6 are set on a blacklist set for the entry $m_{5,1}$ by $B(5,1) = \{1, 4, 6\}$. Next, we re-run the penalized regression to find a new regularization path reflecting the blacklist. However, we also have to include that there are regressors we want to include on the regularization path before the second regressor, i. e. $m_{6,1} = 2$. We will call it the whitelist set $\mathcal{W}(5,1) = \{m_{6,1}\} = \{2\}$. Since we can set individual penalties for each variable, we set $\lambda_{3,2} = 0$. The optimization problem for the entry $m_{5,1}$ is given by:

$$\min_{\varphi \in \mathbb{R}^1} \left(\frac{1}{2n} \sum_{k=1}^n \left(X_{k,3} - \sum_{\ell \in \{1,2,4,5,6\} \setminus \{1,4,6\}} \varphi_{3,\ell} X_{k,\ell} \right)^2 + \sum_{\ell \in \{1,2,4,5,6\} \setminus \{1,4,6\} \setminus 2} \lambda_{3,\ell} |\varphi_{3,\ell}| \right).$$

Thus, we obtain a new sequence $\Lambda(0,3)$ such that $\Lambda(0,3)_1 = m_{6,1}$ and $\Lambda(0,3)_2 = 5$ adheres to the proximity condition. Whenever we have to start a new regression since the next regressor on the regularization path does not adhere to the proximity condition as described previously, we denote this as a proximity condition failure (pcf). In the end, we obtain the complete R-vine matrix M . Additionally, we yield the corresponding λ entries for each entry, based either on an already computed regularization path or a new computation. We store it together with the R-vine matrix.

Using this approach, we complete a partial R-vine matrix column-wise from right to left in $d-1$ steps. However, since each lower order tree put restrictions on higher order trees by the proximity condition, we have j iterations in the $d-j$ -th column for $j = 1, \dots, d-1$. From a computational point of view, it is more favourable to complete the matrix row-by-row, i. e. tree by tree. Thus, the structure estimation, i. e. computation of regularization paths, can be done in parallel. Because of the particular importance, we restate the optimization leading to the higher order tree estimates in the general form.

Definition 5.11 (Higher order tree selection). *Let M be a partial R-vine matrix and*

assume without loss of generality the main diagonal $(m_{1,1}, \dots, m_{d,d}) = (d, \dots, 1)$. For each matrix entry $m_{i,j}$ with $i > j$, define the set of potential regressors

$$\mathcal{H}(i, j) = \{m_{j+1, j+1}, \dots, m_{d,d}\},$$

the whitelist

$$\mathcal{W}(i, j) = \{m_{d,j}, \dots, m_{i+1,j}\},$$

and the blacklist

$$B(i, j) = \{\ell \in \mathcal{H}(i, j) \setminus \mathcal{W}(i, j) : \ell \text{ does not satisfy the proximity condition}\}.$$

For $\varphi \in \mathbb{R}^{j-1-|B(i,j)|}$, we solve the optimization problem:

$$\min_{\varphi} \left(\frac{1}{2n} \sum_{k=1}^n \left(X_{k,j} - \sum_{\ell \in \mathcal{H}(i,j) \setminus B(i,j)} \varphi_{j,\ell} X_{k,\ell} \right)^2 + \sum_{\ell \in (\mathcal{H}(i,j) \setminus B(i,j)) \setminus \mathcal{W}(i,j)} \lambda_{j,\ell} |\varphi_{j,\ell}| \right), \quad (5.10)$$

to obtain a regularization path $\Lambda(\lambda, j)$ such that

- $\Lambda(0, j)_{\ell} = m_{d-\ell+1, j}$ for $\ell \in 1, \dots, |\mathcal{W}(m_{i,j})|$,
- $\Lambda(0, j)_{d-i+1}$ adheres to the proximity condition.

To check whether a specific regressor $m_{i,j}$ is in the blacklist or not, we can use the partial R-vine matrix to see if (2.8) holds for this value. Recall that we know the entire regularization path for each column already for the calculation of the first tree, T_1 . A simpler approach compared to solving the equation (5.10) is the following. Given the regularization path $\Lambda(0, j)$ assume that $(m_{d,j}, \dots, m_{d-i,j}) = (\Lambda(0, j)_1, \dots, \Lambda(0, j)_i)$, i. e. the first i values of the regularization path have been used for the first i trees. If now $\Lambda(0, j)_{i+1}$ is not a valid entry for $m_{d-i-1, j}$ according to the proximity condition, we could also just use the next possible value $\Lambda(0, j)_{i+\ell}$ for $\ell > 1$ which is allowed according to the proximity condition. However, this value might not coincide with the solution of the optimization problem in Definition 5.11, but save computation time.

This concludes the part where we deal with the structure selection of the R-vine. We continue with considering the sparsity, i. e. how to use the Lasso to not only calculate a feasible structure but also perform model selection. Thus, we aim to make our R-vine model sparse by setting independence copulas.

5.2.3 R-vine Regularization Paths

From the previous calculations, we obtain an R-vine structure together with a regularization path, i. e. a functional relationship between $\lambda > 0$ and the non-zero regression coefficients as in Section 3.2, more specifically in Figure 3.1. Now, we use this information to define the entire regularization path of the regression of $X_{m_{j,j}}$ onto X_{ℓ} , $\ell = m_{j+1, j}, \dots, m_{d,j}$ where M denotes the R-vine structure matrix. This path will be called *column regularization path*. For notational convenience, we reverse the order of the

rows of the matrix to obtain a new matrix M^* . By this convention, the corresponding i -th entry in column j corresponds to the i -th R-vine tree and we have $m_{i,j}^* = m_{d-i+1,j}$ for $j = 1, \dots, d-1, i = 1, \dots, d-j$. This is also the notation introduced as *vine-array* in Joe (2014). For example, the first column of the R-vine Matrix M from Example 5.10 is $M_{\cdot,1} = (3, 6, 1, 4, 5, 2)$. Thus, $M_{\cdot,1}^* = (2, 5, 4, 1, 6, 3)$. Finally, note that the j -th column in M and M^* have exactly $d-j$ non-zero entries.

Definition 5.12 (Column regularization path). *Let M be an R-vine structure matrix in d dimensions. We define a column regularization path of the reversed j -th column $M_{\cdot,j}^* = (m_{1,j}^*, \dots, m_{d-j,j}^*)$ by a vector $\boldsymbol{\lambda}_j = (\lambda_{1,j}, \dots, \lambda_{d-j,j}) \geq 0$ for $j = 1, \dots, d-2$ such that*

$$\{i : \lambda_{i,j} < \lambda'\} = \{i : c_{j, \kappa_i(j) | \kappa_1(j), \dots, \kappa_{i-1}(j)} = 0 \text{ in R-vine tree } T_i\}$$

for some $\lambda' > 0$.

Thus, each column $j = 1, \dots, d-1$ of the R-vine matrix is assigned a vector $\boldsymbol{\lambda}_j \in \mathbb{R}^{d-j}$ which contains *threshold* values. These values are a by-product of the penalized regressions we ran and specify for which threshold of penalization, the corresponding SEM coefficients are set to zero, and hence, pair copulas are set to independence copulas. Thus, only by comparing component-wise $\boldsymbol{\lambda}_j > \lambda'$ for some $\lambda' > 0$, the column regularization path helps to set pair copulas to the independence copula to reflect a specific degree of sparsity associated to λ' . For the column $d-1$ where we only have one value, we perform a single regression, so called *soft thresholding* to calculate the corresponding value of $\lambda_{d-1} > 0$. The advantage of this path is now that we are able to regularize each column of the R-vine matrix independently based on a solid theoretical reasoning, i. e. the Lasso. In practice, we consider the R-vine family matrix Γ and fix a specific threshold of $\lambda' > 0$. We consider the column regularization path $\boldsymbol{\lambda}_j$ and calculate component-wise the j -th column of the R-vine family matrix Γ as

$$(\Gamma_{d,j}, \dots, \Gamma_{d-j,j}) = \left(\mathbb{1}_{\{\lambda_{1,j} \geq \lambda'\}}, \dots, \mathbb{1}_{\{\lambda_{d-j,j} \geq \lambda'\}} \right)$$

Note that we reverse the ordering to work solely with lower triangular matrices, i. e. $\Gamma_{d-i+1,j}$ corresponds to $\lambda_{i,j}$ for $i = 1, \dots, d-j$. Thus, all coefficients which are on the regularization path associated to a value of $\lambda < \lambda'$, are set to zero, and hence, the corresponding pair copula is set to the independence copula. The remaining pair copulas are then subject to further estimation. We can not only calculate single column regularization paths, but the entire *regularization path of the R-vine*.

Definition 5.13 (Regularization path of an R-vine). *Let M be an R-vine structure matrix in d dimensions. The regularization path of the R-vine is a matrix $\mathbf{A} \in \mathbb{R}^{d \times d}$ such that its columns $\mathbf{A}_{\cdot,j}$ are column regularization paths of the corresponding R-vine matrix columns $M_{\cdot,j}$, $j = 1, \dots, d-1$.*

Summarizing, we obtain an R-vine structure \mathcal{V} which is not only entirely independent of pseudo-observations of lower level trees as compared to Dißmann's algorithm. It is

also independent of a specific penalization level λ since it is built stepwise by considering the non-zero regressors along the entire regularization paths for the R-vine tree sequence. This allows us to calculate one specific R-vine structure and then consider it under arbitrary many penalization levels λ , obtaining different levels of sparsity for one generally valid R-vine structures. Based on this structure and regularization path matrix, we can then perform maximum likelihood estimation on the non-independence copulas to obtain a general R-vine $\mathcal{W} = (\mathcal{V}, \mathcal{B}(\mathcal{V}), \Theta(\mathcal{B}(\mathcal{V})))$. Next, we consider the regularization path matrix of the six-dimensional example R-vine from Example 5.14 before proposing methods for choosing λ .

Example 5.14 (Example 5.10 cont.). Consider the R-vine matrix M from Example 5.10. For comparison, we also display the R-vine matrix generated from Dißmann’s algorithm, calculated with *VineCopula* R-package (Schepsmeier et al., 2017).

$$M_{Lasso} = \begin{pmatrix} 3 & & & & & & \\ 6 & 2 & & & & & \\ 1 & 6 & 1 & & & & \\ 4 & 1 & 6 & 4 & & & \\ 5 & 4 & 5 & 6 & 6 & & \\ 2 & 5 & 4 & 5 & 5 & 5 & \end{pmatrix} \quad M_{Dißmann} = \begin{pmatrix} 3 & & & & & & \\ 6 & 2 & & & & & \\ 1 & 6 & 1 & & & & \\ 4 & 1 & 6 & 4 & & & \\ 5 & 4 & 5 & 6 & 5 & & \\ 2 & 5 & 4 & 5 & 6 & 6 & \end{pmatrix}$$

Note that these two R-vine matrices give rise to the exactly same R-vine tree structure. We compute the following regularization path matrix $\mathbf{\Lambda}$ for the R-vine structure M .

$$\mathbf{\Lambda} = \begin{pmatrix} 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0072 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0082 & 0.0039 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0005 & 0.0091 & 0.4993 & 0.0000 & 0.0000 & 0.0000 \\ 0.0538 & 0.0210 & 0.6601 & 0.1344 & 0.0000 & 0.0000 \\ 0.3171 & 0.3117 & 0.7244 & 0.9481 & 0.9378 & 0.0000 \end{pmatrix}$$

We observe that the values are column-wise monotonically decreasing if there is no proximity condition failure (pcf). For example, in the first column, the original regularization path $\Lambda(0, 3) = (2, 6, 1, 4, 5)$ did not meet the proximity condition and was recalculated. Thus, the values of λ are not necessarily decreasing. We visualize the column regularization path of column 2 with a step function, see Figure 5.3, indicating the corresponding entries in the R-vine matrix. As $\lambda \rightarrow 0$, more and more pair copulas are set to the independence copula, starting from higher order trees to lower order trees. The matrix $\mathbf{\Lambda}$ can now be used to regulate the sparsity, i. e. the number of independence copulas in our R-vine model. We will outline different approaches in Section 5.2.4 and examine the corresponding effects on the matrix in a subsequent example. Imposing now, say, a threshold

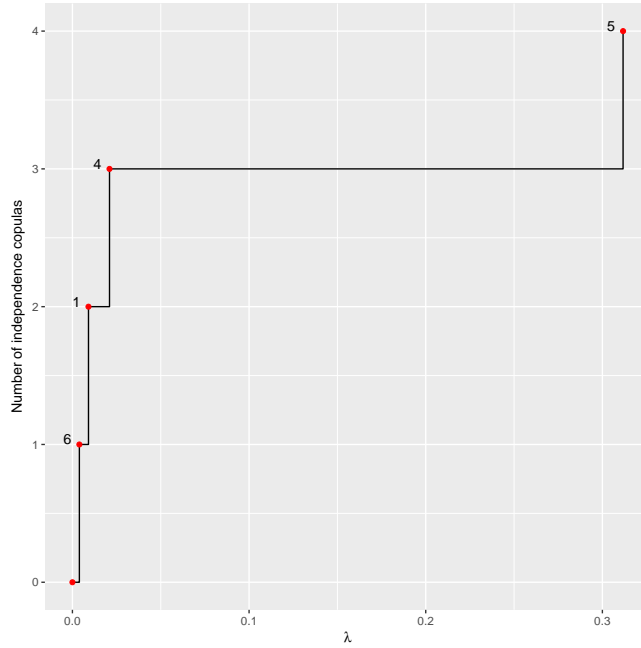


Figure 5.3: Column regularization path of column 2.

$\lambda' = 0.3$, we have the following R-vine family matrix:

$$\Gamma = \begin{pmatrix} - & & & & & \\ 0 & - & & & & \\ 0 & 0 & - & & & \\ 0 & 0 & 1 & - & & \\ 0 & 0 & 1 & 0 & - & \\ 1 & 1 & 1 & 1 & 1 & \end{pmatrix}.$$

The non-zero entries reflect Gaussian pair copulas which are then subject to maximum likelihood estimation and to be exchanged for any other pair copula family. Another possible way to choose the degree of sparsity is to use the cross-validated values λ_{CV} we obtained while calculating the full regularization paths for each X_j with $j = m_{1,1}, \dots, m_{d-2,d-2}$. These values are in our example as follows:

$$\lambda_{CV} = (0.1342, 0.1441, 0.1110, 0.1370).$$

Using this penalization, we obtain an almost full family matrix with less sparsity, as all values in Λ except one are greater or equal these values if considered column-wise. This is however a behaviour we would expect for such low dimensional data sets. We expect higher dimensional data sets to be much more sparse.

5.2.4 Selection of the Tuning Parameter

We propose two approaches how we can utilize the regularization path matrices to obtain sparse R-vine models. A high value in these matrices means a significant contribution

to the model fit, where a low values means the opposite. Introducing now a threshold value λ_T and checking whether or not entries in \mathbf{A} are below or above this value, the corresponding entries in Γ are set to the independence copula or left for estimation of the pair copula type and parameter. Denote the regularization path of the R-vine by $\mathbf{A} = (\lambda_{i,j})_{i=1,\dots,d;j=1,\dots,d}$ with family matrix $\Gamma = (\gamma_{i,j})_{i=1,\dots,d;j=1,\dots,d}$.

Single Threshold Approach

The first approach is to specify some threshold $\lambda_T > 0$ and calculate the family matrix entries according to

$$\gamma_{i,j} = \mathbb{1}_{\{\lambda_{i,j} \geq \lambda_T\}}, \quad j = 1, \dots, d-1, \quad i = 1, \dots, d-j.$$

Pair copulas corresponding to unit entries in the family matrix are then subject to, e.g. maximum likelihood estimation. Such an approach can easily be evaluated using a grid of $\lambda_T \in (0, 1)$. Recall that we only need to compute the structure and regularization matrix once upfront and then evaluate the corresponding threshold. In the data example, we consider a grid of threshold parameters.

Adaptive Threshold Approach

A second approach is to specify not a threshold value itself, but to calculate the threshold such that a specified share of the entries in \mathbf{A} fall below the threshold. Recall that \mathbf{A} has $\binom{d}{2}$ entries as lower diagonal matrix. Our intention is to grab the highest $100\mu\%$ of the values in \mathbf{A} for $\mu \in (0, 1)$. Thus, we solve the following equation for a threshold λ_μ :

$$\lfloor \mu \cdot \binom{d}{2} \rfloor = \sum_{j=1,\dots,d-1, i=1,\dots,d-j}^d \mathbb{1}_{\{\lambda_{i,j} \geq \lambda_\mu\}}$$

This threshold can easily be found by sorting all entries of \mathbf{A} decreasingly and stop once $\lfloor \mu \cdot \binom{d}{2} \rfloor$ entries have been found.

Example 5.15 (Example 5.14 cont.). *We consider the regularization path matrix \mathbf{A} as in Example 5.14. For the single threshold approach, we choose $\lambda_T = 0.1$ to obtain Γ_{λ_T} and for the adaptive threshold approach, we use $\mu = 0.5$. With $\binom{d}{2} = \binom{6}{2} = 15$, we have $\lfloor \mu \cdot \binom{6}{2} \rfloor = 7$, i. e. we select the entries with the highest seven values in \mathbf{A} , obtaining Γ_{λ_μ} .*

$$\Gamma_{\lambda_T} = \begin{pmatrix} - & & & & & \\ 0 & - & & & & \\ 0 & 0 & - & & & \\ 0 & 0 & 1 & - & & \\ 0 & 0 & 1 & 1 & - & \\ 1 & 1 & 1 & 1 & 1 & - \end{pmatrix} \quad \Gamma_{\lambda_\mu} = \begin{pmatrix} - & & & & & \\ 0 & - & & & & \\ 0 & 0 & - & & & \\ 0 & 0 & 1 & - & & \\ 0 & 0 & 1 & 0 & - & \\ 1 & 1 & 1 & 1 & 1 & - \end{pmatrix}$$

5.3 Application

We scale our approach to higher dimensions compared to Section 4.5. Because of the availability of data, we again consider a financial data set. Thus, we obtain data from the S&P500 constituents from January 1, 2013 to December 31, 2016. We isolate $d = 222$ stocks which fall into the sectors Financial Services (70), Health Care (40), Industrials (54), Information Technology (52) and Telecommunication Services (6). We apply suitable *ARMA-GARCH* models to remove trends and seasonality from the time series, see Section 7.1.1. The residuals are then transformed using the empirical cumulative distribution function to the *u-scale*. To obtain models, we use Dißmann’s algorithm with a level $\alpha = 0.05$ independence test and $1, \dots, 221$ -truncation, i. e. we fit a full model and the split it into the first k trees for $k = 1, \dots, 221$ to obtain sub-models. We consider only one-parametric pair copula families and the *t-copula*. The same pair copula selection also applies for our approach where we calculated models along a grid of single threshold values $\lambda_T \in \{0.05^4, 0.1^4, \dots, 0.45^4, 0.5^4\}$. We additionally also test for independence using a significance level $\alpha = 0.05$. As a comparison, we also include the merely Gaussian SEM. We plot the corresponding BIC and mBIC values of both models, see Figure 5.4. For the Lasso approach, the BIC and mBIC of the models is decreasing with decreasing λ_T as less pair copulas get penalized and we obtain more and more parameters. We see that BIC and mBIC attain a minimum for both Lasso and Dißmann approach different from the full models, where $\lambda_T^{opt} = (1/4)^4$ and $t^{opt} = 16$ for Dißmann. Both our Lasso approach and Dißmann’s algorithm outperform the Gaussian SEM significantly which indicates non-Gaussian dependence. Additionally, we see that the Lasso outperforms the Dißmann method in both criteria as it attains smaller values. For computation times, we report that one fit of the Lasso approach took approximately 30 minutes. The entire Dißmann fit for the full model took over 5.5 hours, all times on a Linux Cluster with 32 cores. Thus, all of the Lasso models on the grid were fitted before the full Dißmann fit was complete. The Gaussian SEM is much faster compared to both non-Gaussian approaches. We see that in terms of mBIC, the optimal models have around 2,000 parameters out of total $222 \times (221) / 2 \approx 25,000$ parameters. We observe most often Student’s-t (386) and Frank copulas (810). Our expectation is that for higher dimensional models, the ratio of significant parameters to total number of parameters becomes even smaller, making sparse model selection key for high dimensional setups.

To briefly recapitulate, we have presented a novel approach which uses a representation of an R-vine in terms of a system of linear equations. The solution to these equations, depending on some regularization parameter λ was then translated into a sparse R-vine, without using maximum spanning trees, as the previous algorithms. As we have seen, this tool works very fast and attains good results to obtain parsimonious models. We will now proceed to the third approach, which allows us to tap into even higher dimensions.

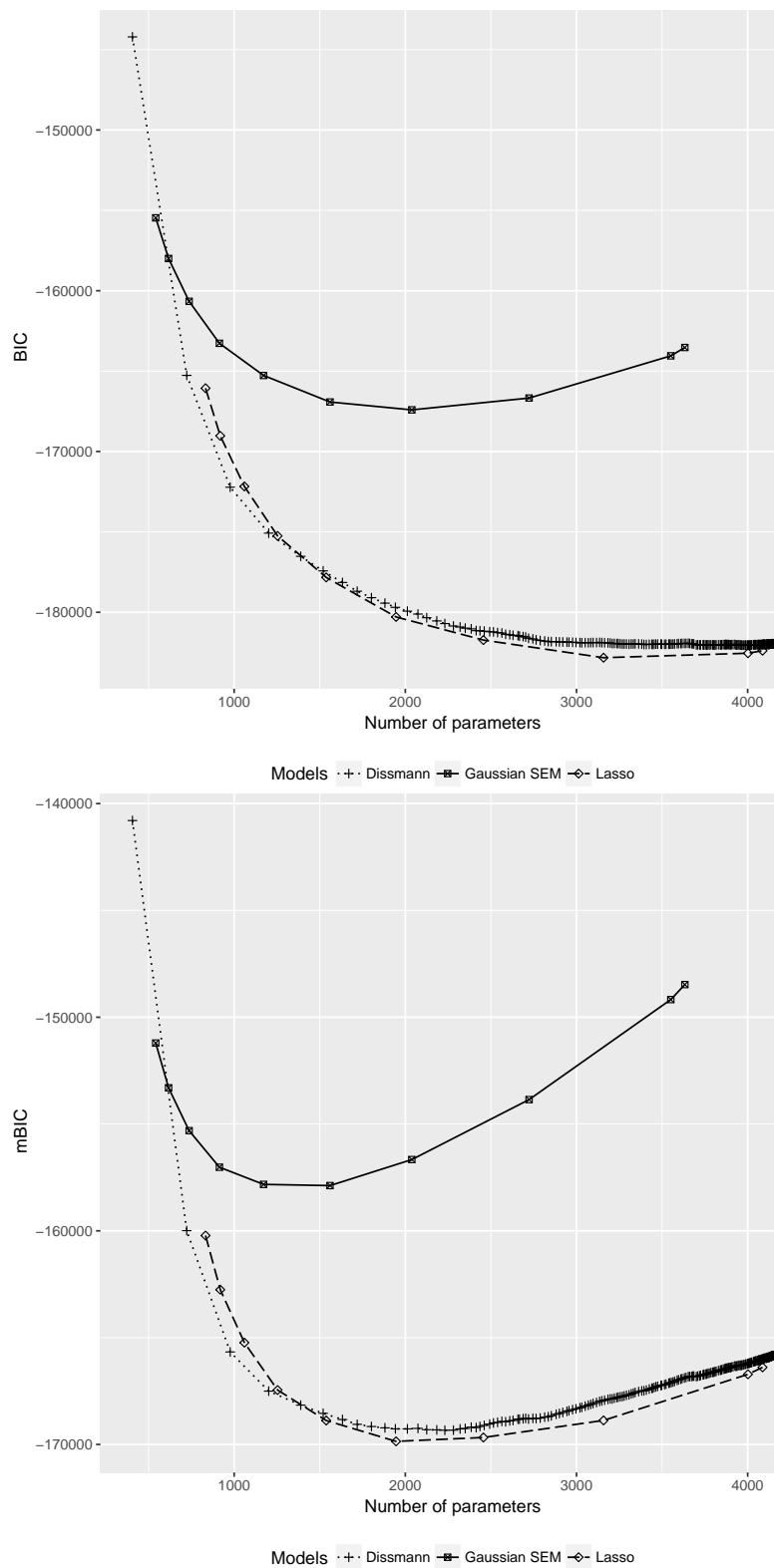


Figure 5.4: Comparison of Lasso approach with single threshold $\lambda_T \in \{0.05^4, 0.1^4, \dots, 0.45^4, 0.5^4\}$ vs. Gaussian SEM model with same threshold vs. t -truncated Dißmann's algorithm, $t = 1, \dots, 221$ on u -scale by number of parameters vs. BIC (top) and number of parameters vs. mBIC (bottom).

Chapter 6

Dependence Modeling in Ultra High Dimensions with Vine Copulas and the Graphical Lasso

We will now continue our third approach, using a *divide-and-conquer* strategy for estimation of R-vines in ultra high dimensions. Since the entire pair copula construction builds on constructing a distribution in d dimensions by $d(d - 1)/2$ components, the problem has an inherently quadratic complexity. We will now sketch how we will firstly use undirected Gaussian graphical models, see Section 3.3.2 to cluster high dimensional data sets in smaller subsets. Next, we will use properties of the Gaussian distribution associated to this graphical model, to make model estimation in R-vines more efficient. The approach resembles partly our method in Chapter 4, where we first draw structural inference from a graphical model, and then use the inherent sparsity to make our R-vine model more parsimonious. After describing these two parts, we will discuss the numerical implementation in more detail. The material in this chapter is taken from Müller and Czado (2017a).

6.1 A Divide-And-Conquer-Strategy

We will briefly motivate our approach in Section 6.1.1. Next, we describe more precisely the *divide* part, i. e. how to cluster ultra-high dimensional problems in smaller sub-problems in Section 6.1.2. After that, the *conquer* part where we show how to estimate R-vines with increased precision in Section 6.1.3 follows.

6.1.1 Motivation: Considering Computational Complexity

First of all, we note that the driving factor in the complexity of a vine copula model is the number of pair copulas to be estimated. Even though the estimation itself can be carried out quickly, the effort grows very fast since in d dimensions, we have in total $d(d - 1)/2$ pair copulas. If we recall the approaches in the Chapters 4 and 5, we notice the following. First, using DAGs, the original problem remained unchanged in a sense, that we were still calculating maximum spanning trees as in the algorithm of Dißmann et al. (2013). Even though the edge weights are calculated differently and many pair copulas are set

to the independence copula upfront, we have to calculate $d - 1$ spanning trees. Hence, it is reasonable to assume that the algorithm we developed in Chapter 4 has a similar computational complexity as Dißmann's.

Using the Lasso method in Chapter 5, we can expect to benefit from the fact that we only solve linear regressions, which is a problem that can be carried out with high efficiency and thus, very fast. Alas, even though the number of equations grows linear in d , we need to solve significantly more of these problems as the proximity condition requires us to re-run the calculations. A nice feature is that given a structure estimate with regularization path matrix, we can conveniently vary λ and obtain the number of non-independence copulas in the resulting model for a specific λ . This allows to control the effort for the estimation of pair copula families and parameters. A disadvantage of this approach is however that it works global and allows not for clusters of variables which are more dependent.

Now, for dimensions $d > 1000$, it is however sensible to assume that there exist clusters of variables which have stronger dependence within the cluster and weaker dependence outside. Hence, a global approach which considers all pairwise dependencies equally has two drawbacks. First, we model dependencies which are not significantly contributing to the model fit and thus, over parametrize our model. Second, in clusters where stronger dependence exists, we may be too imprecise because we oversee these clusters in the magnitude of variables, and hence, have a model with inferior fit. Thus, a divide-and-conquer strategy which

- clusters the data in clusters of stronger dependence,
- models the within-cluster dependence with increased precision,

is expected to outperform other models which work globally on the data set.

6.1.2 Clustering High Dimensional Data

Recall that our goal is ultimately to estimate R-vines in ultra high dimensions. Thus, instead of estimating an R-vine as described in the Chapters 2, 4 or 5 on data in d dimensions, we first cluster the data set into a partition and then perform estimation on the elements of the partition. The partition is provided by the methods presented in Section 3.3.2, i. e. the graphical Lasso. Inside these clusters, we can then estimate R-vines in smaller dimensions with improved accuracy as a consequence of exploiting the global Markov property, (3.10). Assume that we are given data $X \in \mathbb{R}^{n \times d}$, where $d \approx 1000$ or more. We will consider a sequence of J disjoint partitions of $V = \{1, \dots, d\}$ into p_j connected components $V^j := \bigcup_{i=1}^{p_j} V_i^j$, $j = 1, \dots, J$. This is expressed by a sequence of graphical models

$$\mathcal{G}_1 = \left(V^1 = \bigcup_{i=1}^{p_1} V_i^1, \bigcup_{i=1}^{p_1} E_i^1 \right), \dots, \mathcal{G}_J = \left(V^J = \bigcup_{i=1}^{p_J} V_i^J, \bigcup_{i=1}^{p_J} E_i^J \right). \quad (6.1)$$

In most practical applications, $J = 15$ or $J = 30$, see Zhao et al. (2015). If partition V^j is only a single connected component, we have $p_j = 1$. The sequence \mathcal{G}_j for $j = 1, \dots, J$

can be identified as solution path corresponding to the graphical Lasso for J different penalization values of $\lambda > 0$. To identify the size of connected components in these graphs, define for each partition V^j , $j = 1, \dots, J$,

$$\delta_j = \max_{i=1, \dots, p_j} |V_i^j|. \quad (6.2)$$

Instead of considering the entire data set in d dimensions, we consider subsets of lower dimensions on the connected component with maximum dimension $\delta_j < d$ dimensions. In practical applications, we will have some *threshold dimension* $0 < d_T < d$ and calculate the solution path of the graphical Lasso for a sequence $\{\lambda_1, \dots, \lambda_J\}$ based on the screening property (3.14). This works very fast and we can select the corresponding graphical model and associated partition V^T by

$$T = \arg \max_{j=1, \dots, J} \delta_j \text{ such that } \delta_j \leq d_T.$$

Finally, we denote the chosen partition T by V^T and the corresponding graph by $\mathcal{G}_T = (V^T = \bigcup_{i=1}^{p_T} V_i^T, \bigcup_{i=1}^{p_T} E_i^T)$.

Example 6.1 (Example 3.12 cont.). *Consider the sequence of graphs in Example 3.12. Using the notation of (6.1) and defining $\mathcal{G}_j = \mathcal{G}^{\lambda_j}$, we have $J = 4$ and p_j, δ_j for $j = 1, \dots, 4$ as follows:*

j	1	2	3	4
p_j	6	4	3	1
δ_j	1	3	4	6

Table 6.1: Example 6.1: Number of connected components p_j and maximum component sizes δ_j for graphs in Figure 3.6.

Assume $d_T = 4$, then $T = \arg \max_{j=1, \dots, 4} \delta_j$ such that $\delta_j \leq 4$, thus $T = 3$. Hence, the graph \mathcal{G}_3 is selected with partition $V^3 = \{1, 4, 5, 6\} \cup \{2\} \cup \{3\}$ with $p_T = 3$ and $\delta_T = 4$.

Now, we consider the problem of estimating *sub-R-vines* on these connected components induced by a partition V^T . Thus, with respect to this partition we estimate R-vines on the connected components of \mathcal{G}_T , i.e. the elements of the partition V_i^T , $i = 1, \dots, p_T$ with at most dimension δ_T . Each of these *sub-R-vines* is then stored in an R-vine matrix and the corresponding matrices can be combined non-uniquely to an R-vine matrix of dimension $d \times d$. This is however an incomplete R-vine matrix as it does not contain information how the connected components are connected to each other. Additionally, connected components of size 1, i.e. isolated nodes are not yet included. Both the missing connections and the isolated nodes can however be easily connected afterwards as we will show in a subsequent example. For this, we also introduce the *fill-level* $k_F \geq 0$ which determines how many R-vine trees *outside* the connected components should be estimated.

Example 6.2 (Clustering example). Assume we have the graphical model \mathcal{G}_T as in Figure 6.1.

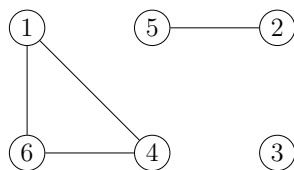


Figure 6.1: Example 6.2: Graphical model \mathcal{G}_T in six dimensions.

Given this graphical model, we estimate sub-R-vines on the components $\{1, 4, 6\}$ and $\{2, 5\}$, respectively. These two components give rise to the following R-vine matrices, where we assume that the estimate in M_1 is optimal with respect to some edge weight.

$$M_1 = \begin{pmatrix} 1 & & \\ 6 & 4 & \\ 4 & 6 & 6 \end{pmatrix} \quad M_2 = \begin{pmatrix} 2 & \\ 5 & 5 \end{pmatrix}.$$

Together with the isolated node 3, these two R-vines can be arranged into a joint R-vine on six dimensions described by the matrix M_{123} .

$$M_{123} = \begin{pmatrix} 3 & & & & & \\ & 2 & & & & \\ & & 5 & & & \\ & & & 1 & & \\ \hline & & & 6 & 4 & \\ \square_1 & 5 & \square_2 & 4 & 6 & 6 \end{pmatrix}, \quad M'_{123} = \begin{pmatrix} 3 & & & & & \\ & 2 & & & & \\ & & 5 & & & \\ & & & 1 & & \\ \hline \Delta_1 & \Delta_2 & \Delta_3 & 6 & 4 & \\ 2 & 5 & 1 & 4 & 6 & 6 \end{pmatrix}.$$

This would correspond to the graphical model in \mathcal{G}_T . Now, we can connect the connected components with each other in the first k_F trees, i. e. to the fill-level. This works as following for the example of $k_F = 2$. The entries in M_{123} marked by \square describe the pair copulas between the connected components and are chosen from $\square_1 \in \{2, 5, 1, 4, 6\}$ and $\square_2 \in \{1, 4, 6\}$. For example, $\square_2 \notin \{3, 2\}$ since only diagonal entries from the right of the corresponding column may be used. To select the entries in the last row, we consider all admissible pairs $(3, \square_1)$ with $\square_1 \in \{2, 5, 1, 4, 6\}$ and $(5, \square_1)$ with $\square_1 \in \{1, 4, 6\}$. Recall that a pair copula in the first tree is fitted on the pair of diagonal entry and the entry in the last row of the R-vine matrix. We fit pair copulas for each of these pairs and then select the best according to some edge weight μ . After having completed the last row in M_{123} using $\square_1 = 2$ and $\square_2 = 1$, we obtain M'_{123} and are to fill the second tree, i. e. the fifth row of M'_{123} consisting of the entries $\Delta_1, \Delta_2, \Delta_3$. However, we have to take into account the proximity condition. By checking (2.8), this leaves admissible entries $\Delta_1 \in \{5\}$, $\Delta_2 \in \{1\}$ and $\Delta_3 \in \{4\}$. Finally, these pair copulas are fitted and the matrix is finalized, see M''_{123} with associated family matrix Γ with \star denoting pair copulas which are not the independence

copula by virtue of the graphical model.

$$M''_{123} = \begin{pmatrix} 3 \\ 6 & 2 \\ 4 & 6 & 5 \\ 1 & 4 & 6 & 1 \\ \hline 5 & 1 & 4 & 6 & 4 \\ 2 & 5 & 1 & 4 & 6 & 6 \end{pmatrix}, \Gamma = \begin{pmatrix} \\ \\ \\ \hline * & * & * & * \\ * & * & * & * & * \end{pmatrix}.$$

We note that this particular R-vine is a D-vine, i. e. the first R-vine tree is given by a path through the nodes $3 - 2 - 5 - 1 - 4 - 6$ and determines all subsequent trees. This is however not necessarily the case in general.

Motivated by the previous example, we define the *R-vine representation* of an undirected graphical model \mathcal{G} with fill level $k_F \geq 0$ by $\mathcal{W}(\mathcal{G}, k_F)$. This R-vine representation can then again be equipped with non-Gaussian pair copulas. Thus, we are left to estimate R-vines in the connected components of \mathcal{G}_T which can be combined into one R-vine. We will not use the standard algorithm for estimation but exploit also the graphical structure *within the connected component*. Hence, consider an arbitrary connected component within \mathcal{G}_T of size ν and denote it by $\mathcal{H} = (W, \mathcal{F})$. This is again a graphical model with respect to the vertices in W . We will describe an approach for estimating R-vines on W with improved accuracy.

6.1.3 Improving Estimation Accuracy

We are now considering a connected component of $\mathcal{G}_T = (V^T = \bigcup_{i=1}^{p_T} V_i^T, \bigcup_{i=1}^{p_T} E_i^T)$ and denote it by $\mathcal{H} = (W, \mathcal{F})$. We are to estimate an R-vine on the variables in W and denote $\nu = |W|$. We consider the computational complexity in terms of pair copulas to estimate. In total, these are $\nu(\nu - 1)/2$ pair copulas to be estimated for a R-vine on W with ν variables. Denote the corresponding R-vine tree sequence $\mathcal{V} = (T_1, \dots, T_{\nu-1})$. To find the first R-vine tree T_1 , start with a full graph on W . Dißmann's algorithm would now Kendall's τ on all pairs $(j, \ell) \in W \times W$ and use the weights $\mu_{j,\ell} = |\tau_{j,\ell}|$ to find a *maximum spanning tree*. As discussed in Section 2.2.2, we can also estimate pair copula densities $c_{j,\ell}$ for each edge (j, ℓ) on all $\nu(\nu - 1)/2$ edges and calculate a maximum spanning tree with respect to an edge weight $\mu_{j,\ell}$, e. g. log-likelihood or a penalized goodness of fit measure as in (2.9), based on the *actual fit*. However, this increases the required effort for an R-vine tree sequence significantly. As we have $\nu - 1$ trees on $\nu - i + 1$ nodes for $i = 2, \dots, \nu - 2$ and consider in the worst case all possible pairs in each tree, this sums up to

$$\sum_{i=1}^{\nu-2} \frac{(\nu - i + 1)(\nu - i)}{2} = \frac{\nu^3 - \nu}{6} - 1, \quad (6.3)$$

pair copulas, where the leading term can only be bounded from above by ν^3 . Note that this is a worst case complexity since in higher trees, the proximity condition can exclude

some edges, and hence, no pair copula needs to be estimated for these. In the particular case of a D -vine, where the first tree is a path, i. e. each node except for the two end nodes has degree 2, the remaining R-vine tree sequence is completely determined and the effort collapses to order $\nu(\nu - 1)/2$. Whereas in the case of a C -vine, where each tree is a star, the upper bound in (6.3) is attained. Recall that the method of Dißmann et al. (2013), only $\nu(\nu - 1)/2$ pair copulas have to be estimated. Hence, for general R-vines, estimation of ν^3 pair copulas is not admissible even if $\nu \ll d$. However, we can take into account the (conditional) independence information by the graph \mathcal{H} to set a huge number of pair copulas to independence copulas upfront, leading to a significant decrease of computational effort.

- Recall that \mathcal{F} is the edge set of \mathcal{H} and define $\mathcal{H}_0 = (W, W \times W)$, i. e. a full graph on W and assign pair copulas and weights

$$c_{j,\ell} = 1 \Leftrightarrow \mu_{j,\ell} = 0 \Leftrightarrow (j, \ell) \notin \mathcal{F}.$$

For all remaining edges $(j, \ell) \in W \times W$, we perform maximum likelihood estimation on the pair copulas and obtain weights $\mu_{j,\ell} = |\mu(c_{j,\ell})|$ based on the actual fit. Thus, we have weights for all edges and can now calculate a maximum spanning tree $T_1 = (W, E_1)$.

- We define $T_k = (E_{k-1}, E_{k-1} \times E_{k-1})$ for $k = 2, \dots, \nu - 1$, i. e. a full graph on E_{k-1} . We remove all edges not allowed by the proximity condition. For the remaining edges, we assign pair copulas and weights

$$c_{j,\ell;D} = 1 \Leftrightarrow \mu_{j,\ell;D} = 0 \Leftrightarrow j \perp \ell \mid D \text{ } [\mathcal{H}].$$

For all remaining edges $(j, \ell; D) \in E_{k-1} \times E_{k-1}$ we perform maximum likelihood estimation on the pair copulas and obtain weights $\mu_{j,\ell;D} = |\mu(c_{j,\ell;D})|$ based on the actual fit. Thus, we have weights for all edges and can now calculate a maximum spanning tree T_k for $k = 2, \dots, \nu - 1$.

If the connected component \mathcal{H} is not too dense, i. e. $|\mathcal{F}| \sim \nu$ instead of $|\mathcal{F}| \sim \nu^2$, overall about ν^2 pair copulas are estimated for the entire R-vine in ν dimensions. From this point of view, it is more beneficial to neglect the edges which are not chosen anyway because of (conditional) independence and perform a more thorough analysis on the remaining edges. The algorithm of Dißmann might miss important edges, especially if all estimated Kendall's τ values are similar. Furthermore, our approach fosters sparsity by setting pair copulas to independence straight away. This is not possible in Dißmann's algorithm, except for truncation, which however has to be specified upfront in a very inflexible manner. We will show that our approach is able to capture significantly more dependence compared to Dißmann's. An example illustrating this approach finalizes the section.

Example 6.3 (Example 6.1 cont.). Assume the graphical model \mathcal{H} as in Figure 6.2 (left).

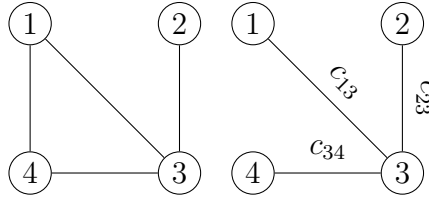


Figure 6.2: Example 6.3: Graphical model \mathcal{H} (left) and first R-vine tree T_1 (right).

Using our approach, we fit pair copulas $c_{1,3}$, $c_{1,4}$, $c_{2,3}$, $c_{3,4}$ and consider the corresponding goodness of fit values $\mu_{1,3}$, $\mu_{1,4}$, $\mu_{2,3}$, $\mu_{3,4}$, given for example by the AIC. The other missing edges are neglected, i. e. no pair copula is estimated. Assume additionally that the optimal choice with respect to AIC is given by the tree given in the right panel of Figure 6.2. Then, the second R-vine tree T_2 may contain for example the possible edge $c_{2,4|3}$. However, we can see from the left panel of Figure 6.2 that $2 \perp 4 \mid 3$ [\mathcal{H}] and thus, the pair copula $c_{2,4|3}$ is set to the independence copula upfront in T_2 .

6.2 Algorithmic Implementation

We will now consider more deeply the algorithmic implementation of the approach, first going into the computational details in Section 6.2.1 and then giving an exhaustive toy example in Section 6.2.2.

6.2.1 Computational Implementation

We combine the two previous steps in one algorithm, allowing to estimate high dimensional vine copulas based on a clustering by the graphical Lasso. For this, we need to specify a threshold dimension $d_T < d$ and a fill-level $k_F \geq 0$ describing until which tree we will estimate pair-copulas *outside* the connected components. The later is beneficial since we only use the connected components to break our original problem into tractable sub-problems. However, if there is dependence outside the connected components, from a computational point of view, estimating within the connected components of dimension at most $d_T < d$ first and then connect these components afterwards is much more beneficial than estimating an R-vine on d dimensions. We use the **huge** R-package, see Zhao et al. (2015) to generate high dimensional undirected graphical models and use the default settings there. Since we normally operate on *copula-data* (U_1, \dots, U_n) , i. e. data with uniform marginals, we transform our observations to the *z-scale*, i. e. consider $(Z_1 = \Phi^{-1}(U_1), \dots, Z_d = \Phi^{-1}(U_d)) \sim \mathcal{N}_d(0, \Sigma)$ in a Gaussian set up. The only change we perform in the default setting of **huge** is that we always want to obtain a sequence of 30 graphs, this is regulated by the number of $\lambda_j \geq 0$ to be evaluated. In terms of the previous notation, we set $J = 30$. Our algorithm then selects a partition such that the maximum component size is less or equal d_T as shown in Section 6.1.2 and then performs on each of the components an improved R-vine selection based on Section 6.1.3. The edge metric μ we use is the absolute value of the AIC of the associated pair copula term

in our considerations. Afterwards, we combine these *sub-R-vines* into one *joint R-vine* matrix on which we operate further. Finally, we estimate the pair copulas in the first $k_F \geq 0$ trees of the joint R-vine. From this point on, we only operate on the R-vine matrix M . Thus, we have to take into account that in a certain column j of M , only values of the main diagonal right of j , i. e. in $M_{j+1,j+1}, \dots, M_{d,d}$ can occur. Thus, for a given entry in $M_{i,j}$, we check which entries in $M_{j+1,j+1}, \dots, M_{d,d}$ are valid according to the proximity condition. For those, we fit pair copulas and select the best choice according to the same metric, i. e. absolute value of AIC. The entire algorithm, which we will refer to as *RVineClusterSelect*, is given Appendix, Chapter C.

6.2.2 Toy Example

For a better understanding, we provide an exhaustive end-to-end toy example which covers all aspects of our algorithm using an example in eight dimensions.

Example 6.4 (End-to-End Toy Example). *We will now sketch an end-to-end toy example to illustrate precisely how our algorithm works. Therefore, we use an example in $d = 8$ dimensions for illustration and refer to the **code lines** in Algorithm 8.*

Line Input: $X \in \mathbb{R}^{n \times 8}$, $d_T = 4$, $k_F = 2$.

Line 1 Assume we have a sequence of graphs as given in Figure 6.3.

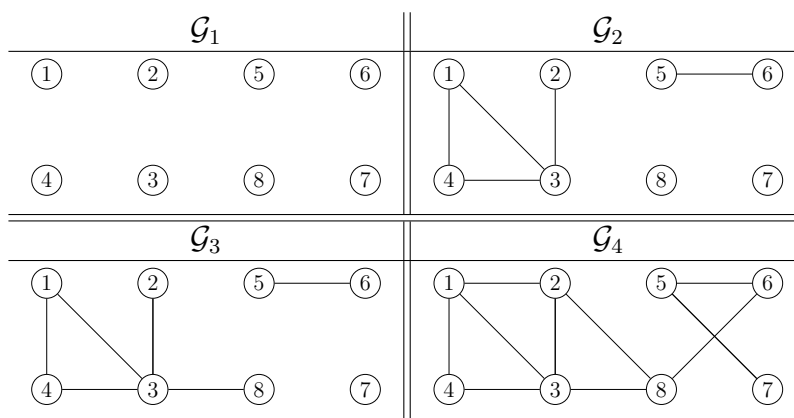


Figure 6.3: Example 6.4: Sequence of estimated graphical models $\mathcal{G}_1, \dots, \mathcal{G}_4$ on X .

Line 2 We calculate the corresponding number of connected components p_j and maximum component sizes δ_j , $j = 1, \dots, 4$ as in Table 6.2.

j	1	2	3	4
p_j	8	4	3	1
δ_j	1	4	5	8

Table 6.2: Example 6.4: Number of connected components p_j and maximum component sizes δ_j for graphs $\mathcal{G}_1, \dots, \mathcal{G}_4$ in Figure 6.3.

Given the threshold dimension $d_T = 4$, see **Line Input**, we find

$$T = \arg \max_{j=1,\dots,4} \delta_j \leq 4 \Rightarrow T = 2.$$

More precisely, only for this clustering, i. e. graph \mathcal{G}_2 , we still obtain a number of dimensions less or equal to the threshold dimension $d_T = 4$. This is the main ingredient to control the estimation effort in the sub R -vines. Hence, we have four connected components in \mathcal{G}_2 , i. e. $p_T = 4$. With the ordering we assume in Algorithm 8, **Line 1** we have the node and edge sets

$$\begin{aligned} V_1^2 &= \{1, 2, 3, 4\}, & E_1^2 &= \{(1, 3), (1, 4), (2, 3), (3, 4)\}, \\ V_2^2 &= \{5, 6\}, & E_2^2 &= \{(5, 6)\}, \\ V_3^2 &= \{7\}, & E_3^2 &= \emptyset, \\ V_4^2 &= \{8\} & E_4^2 &= \emptyset. \end{aligned}$$

Line 3 We start into the **for** loop over all $p_T = 4$ connected components with $i = 1$ and consider the first connected component with four nodes, V_1^2 .

Line 4 Set $\nu_1 = |V_1^2| = 4$, $\mathcal{H} = (\mathcal{W} = V_1^2, \mathcal{F} = E_1^2)$ and $\mathcal{H}_0 = (\mathcal{W}, \mathcal{W} \times \mathcal{W})$.

Line 5 We check $\nu_i \geq 3$. This is necessary to determine if there are less than three nodes in this connected component. If this is the case, i. e. for example only one node, then there is no pair copula to estimate. If the connected component contains two nodes, there is only one pair copula to estimate, and this case will be covered later on. Now, since $\nu_1 = |V_1^2| = |\{1, 2, 3, 4\}| = 4$, this **if** condition is fulfilled. We continue by calculating the first R -vine tree on this connected component.

Line 6 We consider first all possible edges in the full graph \mathcal{H}_0 , i. e.

$$\begin{aligned} &(1, 2), (1, 3), (1, 4), \\ &(2, 3), (2, 4), \\ &(3, 4). \end{aligned}$$

Lines 7–14 We check for each of the edges in the full graph \mathcal{H}_0 , if it is contained in the edge set \mathcal{F} of \mathcal{H} . If this is not the case, we assign the independence copula to these directly. Otherwise, we perform pair copula estimation. More precisely, we have

$$\begin{aligned} c_{1,2}(u_1, u_2) &= 1 \\ c_{2,4}(u_2, u_4) &= 1, \end{aligned}$$

since these edges are not contained in \mathcal{H} . For the remaining edges, we carry out maximum likelihood estimation or inversion of Kendall's τ . Assume we have estimated pair copulas and calculated their positive AIC values

$$\mu_{j,\ell} = |\mu(c_{j,\ell})|,$$

and

$$\mu_{3,4} > \mu_{1,3} > \mu_{2,3} > \mu_{1,4} > 0.$$

We determine a maximum spanning tree, and hence the first R-vine tree on this partition T_1^{1234} , see Figure 6.4 (right panel).

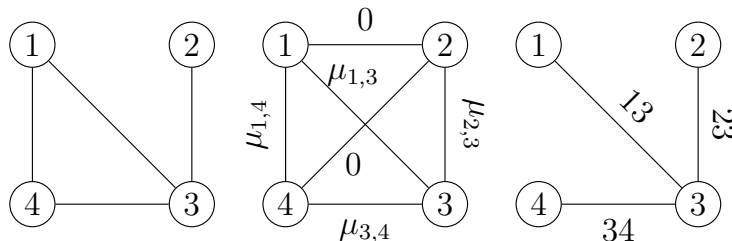


Figure 6.4: Example 6.4: Graphical model \mathcal{H} (left), weighted graph \mathcal{H}_0 (middle) and first R-vine tree T_1^{1234} (right).

Lines 15–26 We start into the loop with $k = 2$. Thus, we build the second R-vine tree T_2^{1234} . Since T_1^{1234} is a star, the proximity condition does not put any restrictions on edges in T_2^{1234} . Recall the first R-vine tree is depicted in Figure 6.4, right panel. We find the second R-vine tree by considering the full graph on the nodes E_1 , see Figure 6.5. Now,

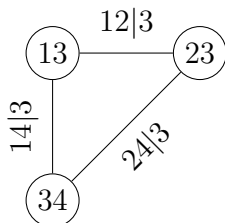


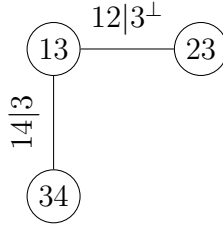
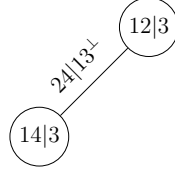
Figure 6.5: Example 6.4: Intermediate step towards the second R-vine tree T_2^{1234} .

for each of the edges in Figure 6.5 we check, according to code line *Line 18*

$$\begin{aligned} 1 \perp 2 \mid 3 [H] &\Rightarrow c_{12;3}(u_{1|3}, u_{2|3}) = 1 \\ 1 \not\perp 4 \mid 3 [H] & \\ 2 \perp 4 \mid 3 [H] &\Rightarrow c_{24;3}(u_{2|3}, u_{4|3}) = 1. \end{aligned}$$

Thus, the two pair copulas $c_{12;3}$ and $c_{24;3}$ are set the independence copula directly and assigned weights $\mu_{12;3} = \mu_{24;3} = 0$. Since $c_{14;3}$ can not be set to the independence copula by virtue of the graphical separation, we perform maximum likelihood estimation or estimation by inversion of Kendall's τ . Hence, a maximum spanning tree and thus, the second R-vine tree T_2^{1234} is given by Figure 6.6. Edges with the independence copula by virtue of the graphical separation are indicated with \perp . Note however, that it is also possible that the maximum likelihood estimate for $c_{14;3}$ is the independence copula.

Lines 15–26 We rerun the loop for $k = 3$. The third and last R-vine tree T_3^{1234} is already


 Figure 6.6: Example 6.4: Second R-vine tree T_2^{1234} .

 Figure 6.7: Example 6.4: Third R-vine tree T_3^{1234} .

fully determined and given in Figure 6.7. For the pair copula $c_{24;13}$, we see with the graph \mathcal{H} in Figure 6.4 (left panel), we have

$$2 \perp 4 \mid 13 [H] \Rightarrow c_{24;13}(u_{2|13}, u_{4|13}) = 1.$$

Hence, it is also assigned the independence copula. This determines the sub-R-vine on the nodes

$$V_1^2 = \{1, 2, 3, 4\}$$

with R-vine matrix

$$M_{1234} = \begin{pmatrix} 2 & & & \\ 4 & 4 & & \\ 1 & 1 & 1 & \\ 3 & 3 & 3 & 3 \end{pmatrix}, \Gamma_{1234} = \begin{pmatrix} - & & & \\ 0 & - & & \\ 0 & * & - & \\ * & * & * & - \end{pmatrix},$$

and associated parameter and family matrix Γ_{1234} , P_{1234} . Note that a 0 in the family matrix Γ_{1234} denotes the independence copula and * represents a pair copula family chosen according to maximum likelihood estimation.

Line 3 We have hence the second iteration in the outer loop with $i = 2$ and the second partition

$$V_2^2 = \{5, 6\} \quad E_2^2 = \{(5, 6)\}.$$

Line 4 Set $\nu_2 = |V_2^2| = 2$, $\mathcal{H} = (\mathcal{W} = V_2^2, \mathcal{F} = E_2^2)$ and $\mathcal{H}_0 = (\mathcal{W}, \mathcal{W} \times \mathcal{W})$.

Line 5 We see that there are only two nodes and hence, we go to the **else** case in **Line 29** and estimate the pair copula $c_{5,6}$. We find the R-vine matrix easily to be

$$M_{56} = \begin{pmatrix} 6 & 0 \\ 5 & 5 \end{pmatrix}.$$

The remaining loops are empty since the condition in **Line 5** or **Line 28** are not satisfied. Hence, we continue and combine the sub R -vines.

Line 33 We link the two sub- R -vines and the two isolated nodes in one joint R -vine matrix. For this, we start with the largest sub- R -vine in the lower right corner of the joint R -vine matrix.

$$M_0 = \begin{pmatrix} 8 \\ 0 & 7 \\ 0 & 0 & 6 \\ 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 5 & 0 & 3 & 3 & 3 & 3 \end{pmatrix}.$$

Additionally, keep in mind that also the corresponding family and parameter matrices are combined this way. We note that the 0 entries in the lowest row indicate the connections outside the connected components. Since we have fill-level $k_F = 2$, we need to find optimal connections for the 0 entries in the lowest two rows. Hence, we consider all possible combinations. Denote

$$M_1 = \begin{pmatrix} 8 \\ 0 & 7 \\ 0 & 0 & 6 \\ 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ \Delta_1 & \Delta_2 & 5 & \Delta_3 & 3 & 3 & 3 & 3 \end{pmatrix}.$$

In an R -vine matrix M , for an entry $M_{i,j}$, only values out of $M_{j+1,j+1}, \dots, M_{d,d}$ can occur. Thus, we have

$$\begin{aligned} \Delta_1 &\in \{1, 2, 3, 4, 5, 6, 7\}, \\ \Delta_2 &\in \{1, 2, 3, 4, 5, 6\}, \\ \Delta_3 &\in \{1, 2, 3, 4\}. \end{aligned} \tag{6.4}$$

Next, we carry out maximum likelihood estimation or pair copula estimation via inversion of Kendall's τ for each possible pair. Hence, we estimate pairs between the corresponding diagonal entry and the admissible entries in (6.4),

$$\begin{aligned} &(1, 8), (2, 8), (3, 8), (4, 8), (5, 8), (6, 8), (7, 8), \\ &(1, 7), (2, 7), (3, 7), (4, 7), (5, 7), (6, 7), \\ &(1, 5), (2, 5), (3, 5), (4, 5), \end{aligned} \tag{6.5}$$

calculate the AIC of each pair in (6.5) and choose the optimal pair copulas for each open

to optimal values of μ is carried out. An admissible choice for M with final family matrix Γ is for example

$$M = \begin{pmatrix} 8 \\ 7 & 7 \\ 6 & 6 & 6 \\ 5 & 4 & 4 & 5 \\ 4 & 1 & 1 & 4 & 2 \\ 1 & 3 & 3 & 1 & 4 & 4 \\ 3 & 2 & 2 & 3 & 1 & 1 & 1 \\ 2 & 5 & 5 & 2 & 3 & 3 & 3 & 3 \end{pmatrix}, \Gamma = \begin{pmatrix} - \\ 0 & - \\ 0 & 0 & - \\ 0 & 0 & 0 & - \\ 0 & 0 & 0 & 0 & - \\ 0 & 0 & 0 & 0 & 0 & - \\ * & * & * & * & 0 & * & - \\ * & * & * & * & * & * & * & - \end{pmatrix}.$$

This finishes our example.

6.3 Application

If we want to compare the Gaussian modelling approach with the non-Gaussian R-vine approach, we need to set the fill-level $k_F = 0$, i. e. only model the edges present in the Gaussian model by pair copulas and not include additional edges between the connected components. However, for illustration of the effect, we will also include a fit using fill-level $k_F = 1$ in our considerations later on. Secondly, we have to assume the same marginal distributions. This time, we obtained data on $d = 2131$ stocks with $n = 999$ observations. The stocks are based in the US (1866), Sweden (154) and Australia (111). The following industry sectors are covered, see Table 6.3.

ID	Sector Description	Total	USA	Sweden	Australia
1	Materials and Energy	218	209	3	6
2	Industry Conglomerates	4	4	0	0
3	Consumer Staples	205	189	12	4
4	Financial Services and Real Estate	524	449	49	26
5	Healthcare and Chemicals	232	182	26	24
6	Manufacturing, Industrials and Defense	195	173	12	10
7	Business Management and Services	325	295	21	9
8	IT, Telecommunication and Software	360	301	28	31
9	Utilities	68	64	3	1
Total		2131	1866	154	111

Table 6.3: Distribution of stocks over industry sectors and geographies in the data application.

Define S_i the i -th stock for $i = 1, \dots, 2131$, denote the industry sector assignment by

$$\mathcal{I}_\ell : \{i : \text{Stock } S_i \text{ belongs to industry } \ell\} \text{ for } \ell = 1, \dots, 9,$$

and the node set $V = \{1, \dots, 2131\}$ for all graphical models $\mathcal{G}_1, \dots, \mathcal{G}_{30}$. We estimate a

path of 30 graphical Lasso solutions $\mathcal{G}_1, \dots, \mathcal{G}_{30}$ calculated by the huge R-package without setting a threshold dimension. To obtain also the log-likelihoods, we evaluate the corresponding covariance matrices of the solutions. We report the first ten corresponding maximum connected component sizes of $\mathcal{G}_1, \dots, \mathcal{G}_{10}$ in Table 6.4.

δ_1	δ_2	δ_3	δ_4	δ_5	δ_6	δ_7	δ_8	δ_9	δ_{10}
1	2	42	84	128	215	453	804	1027	1231

Table 6.4: Maximum connected component sizes for $\mathcal{G}_1, \dots, \mathcal{G}_{10}$.

The remaining 20 cluster sizes for $\mathcal{G}_{11}, \dots, \mathcal{G}_{30}$ are significantly higher than 1200 and thus not considered for our method, as these subs-dimensions are too large to fit an R-vine model onto them. In fact, we only fit R-vines on the graphs $\mathcal{G}_2, \dots, \mathcal{G}_9$ since the first partition with $\delta_1 = 1$ is an empty graph bearing no information and the largest \mathcal{G}_{10} with $\delta_{10} = 1231$ is also too large. Hence, we consider eight graphical models $\mathcal{G}_2, \dots, \mathcal{G}_9$ in the following. An interesting property we can observe is the *industry sector concentration* within each connected component of the graphical models $\mathcal{G}_2, \dots, \mathcal{G}_9$. In the notation of Section 6.1.2, consider graphical models $\mathcal{G}_j = (V^j = \bigcup_{i=1}^{p_j} V_i^j, \bigcup_{i=1}^{p_j} E_i^j)$ with $j = 2, \dots, 9$. Let us fix j and consider the k -th component V_k^j of V^j . Then, denote $|V_k^j| = \nu_k^j$, i. e. ν_k^j stocks are contained in this connected component. Recall that each node represents a stock associated to one industry sector. We define the industry sector occurring most often in the component V_k^j by

$$b_k^j = \arg \max_{\ell \in \{1, \dots, 9\}} \sum_{i \in V_k^j} \mathbb{1}_{\{i \in \mathcal{I}_\ell\}}(i).$$

A natural measure for the sector concentration in V_k^j is given by

$$\varrho_k^j = \frac{b_k^j}{\nu_k^j}.$$

In other words, we count the occurrence of all different sectors within each connected component and divide the number of the most often occurring sector per connected component by the total number of nodes in this connected component. We do this for all connected components within each of the graphs in the sequence $\mathcal{G}_2, \dots, \mathcal{G}_9$. The results, see Table 6.5, demonstrate a very high sector concentration in the connected components over the entire sequence of considered graphs. We see first of all, that a large portion of connected components have sector concentration $\varrho_j = 1$, second and third column of Table 6.5. Thus, at least 80% of the connected components are dominated by a single sector. Additionally, we see from the last column that also the connected components where more than one sector are present, have quite a high sector concentration. This backs our assumption that the graphical Lasso works very well to isolate highly dependent subsets from each other. The motivation for this is the assumption that especially in high dimensions, there exist *clusters of dependence*. As we have e. g. geographical or

\mathcal{G}_j	p_j	$ \{k \in 1, \dots, p_j : \varrho_k^j = 1\} $	$ \{k \in 1, \dots, p_j : \varrho_k^j = 1\} / p_j$	mean of $\{\varrho_k^j : \varrho_k^j \neq 1\}$
\mathcal{G}_2	3	3	1	-
\mathcal{G}_3	20	19	0.95	0.83
\mathcal{G}_4	42	39	0.93	0.64
\mathcal{G}_5	67	62	0.93	0.74
\mathcal{G}_6	69	60	0.87	0.66
\mathcal{G}_7	61	46	0.75	0.69
\mathcal{G}_8	47	41	0.87	0.55
\mathcal{G}_9	31	26	0.84	0.58

Table 6.5: Number of connected components p_j in \mathcal{G}_j for $j = 2, \dots, 9$, number of connected components $k = 1, \dots, p_j$ with sector concentration $\varrho_k^j = 1$, percentage of connected components with sector concentration $\varrho_k^j = 1$ compared to all connected components and mean of the sector concentrations ϱ_k^j over the remaining connected components where sector concentration $\varrho_k^j \neq 1$.

industry-sectoral dependency in a high dimensional stocks data set, we can expect the model to make use of *conditional independencies* or, in other words, *sparsity*. The idea is that after all intra-geographical or sectoral dependencies are described, cross geographical or sectoral dependencies are weak and can be neglected outside the connected components. The same is often monitored in biology, where only a small set of genes affects each other. Finally, we compare the Gaussian model and the R-vine model in terms of log-likelihood and GIC on the *z-scale* for corresponding numbers of parameters, see Figure 6.10. In the Gaussian model, the number of parameters is equal to the number of edges in the graph whereas for the R-vine, the number of parameters is equal to the number of pair copula parameters. In both cases, we add d parameters for the estimated variance of the marginal distributions. This is because we need to add Gaussian marginals with the same variance as estimated by the graphical Lasso to our R-vine model, to make the two models comparable.

We clearly see that the flexibility of the R-vine compared to the Gaussian model leads to significant out-performance with respect to log-likelihood and even more with respect to GIC. This stems from the fact that the R-vine gains much more exploratory power with adding additional parameters. This also true for the corresponding model with $k_F = 1$. The most parsimonious model in terms of GIC is the R-vine given by \mathcal{G}_7 . This is also similar for the Gaussian model. We analyse this model more precisely.

First of all, we check both the *intra*class and *inter*class correlations based on \mathcal{G}_7 . To this end, we compute the sums of the absolute values of the corresponding correlations. As a proxy for the interclass correlations, we calculate the mean of all off-diagonal entries in the correlation matrix.

$$\rho^{inter} = \frac{1}{\binom{2131}{2}} \sum_{i=1}^{2131} \sum_{j=1, j \neq i}^{2131} |\rho_{i,j}| \approx 0.187$$

Next, we iterate over all connected components of \mathcal{G}_7 which contain more than one node and denote them by

$$\mathcal{K}_7 = \{i \in \{1, \dots, p_7\} : |V_i^7| > 1\}.$$

For each $k \in \mathcal{K}_7$, we calculate the mean of the intraclass correlations by

$$\rho_k^{intra} = \frac{1}{\binom{|V_k^7|}{2}} \sum_{i \in V_k^7} \sum_{j \in V_k^7, j \neq i} |\rho_{i,j}| \text{ for all } k \in \mathcal{K}_7.$$

In total, we have $|\mathcal{K}_7| = 61$ connected components. We average over these mean values to obtain

$$\rho_{intra} = \frac{1}{61} \sum_{k=1}^{61} \rho_k^{intra} \approx 0.406,$$

which is significantly higher than the proxy of the interclass correlation ρ_{inter} .

Next, we analyse more precisely the pair copula families in the final model based on the graph \mathcal{G}_7 . We split our considerations between the first R-vine tree T_1 and all the higher order trees T_2, \dots, T_{2130} . The five most occurring pair copula families in T_1 are given in Table 6.6. We see that more than half of the pair copulas are Student's-t copulas as we

Copula family	absolute number	relative number
Student's-t	1078	0.51
Frank	382	0.18
Survival Gumbel	186	0.09
Survival BB1	143	0.07
Gaussian	126	0.06

Table 6.6: Most occurring pair copula families in R-vine tree T_1 .

expect for financial datasets because of their feature to accommodate tail dependence. We also can evaluate the mean of the degrees of freedom df for these Student's-t copulas, measuring their tail dependence, and find it to be about seven. Hence, we have strong non-Gaussian dependence. Furthermore, also Gumbel and BB1 families are present for asymmetric dependence. We also see a high number of Frank copulas which are symmetric and have no tail dependence. Considering this more precisely, evaluate the pairs in the first R-vine tree which are described by Frank copulas. Of these pairs, only 47% contain stocks which share the same economic sector. In contrary, if we consider the pairs which are not linked by Frank copulas, we obtain more than 67% sharing the same economic sector. Hence, more than half of the Frank copulas describe inter-class dependences which are expected to be weaker, where more than two third of the other pair copulas describe intra-class dependence which is found to be stronger, see the above considerations.

When considering the higher order trees, we find the following distribution of most occurring pair copula families, see Table 6.7. We do not give any relative proportions in this case since the widely dominating copula is the independence copula because of the sparsity induced by the graphical model. Finally, we also find the R-vine to be empirically truncated at $k = 405$, i. e. only the first 405 trees do contain pair copulas different from

Copula family	absolute number
Frank	2439
Student's-t	846
Gaussian	699
Survival BB8	427
Survival Gumbel	215

Table 6.7: Most occurring pair copula families in R-vine trees T_2, \dots, T_{2131} .

the independence copula, and the higher 1726 trees do not, thus, less than 20% of the trees contain non-independence copulas. When considering the Student's-t copulas in the R-vine and their parameters, i. e. the degrees of freedom df , we have that for lower df , we have more tail dependence and for higher $df > 30$, the distribution becomes quite similar to the Gaussian distribution. We consider box plots of the degrees of freedom over the R-vines computed with the graphs $\mathcal{G}_3, \dots, \mathcal{G}_9$, see Figure 6.9. Note that in the R-vine computed based on \mathcal{G}_2 , we do not have any Student's-t copulas.

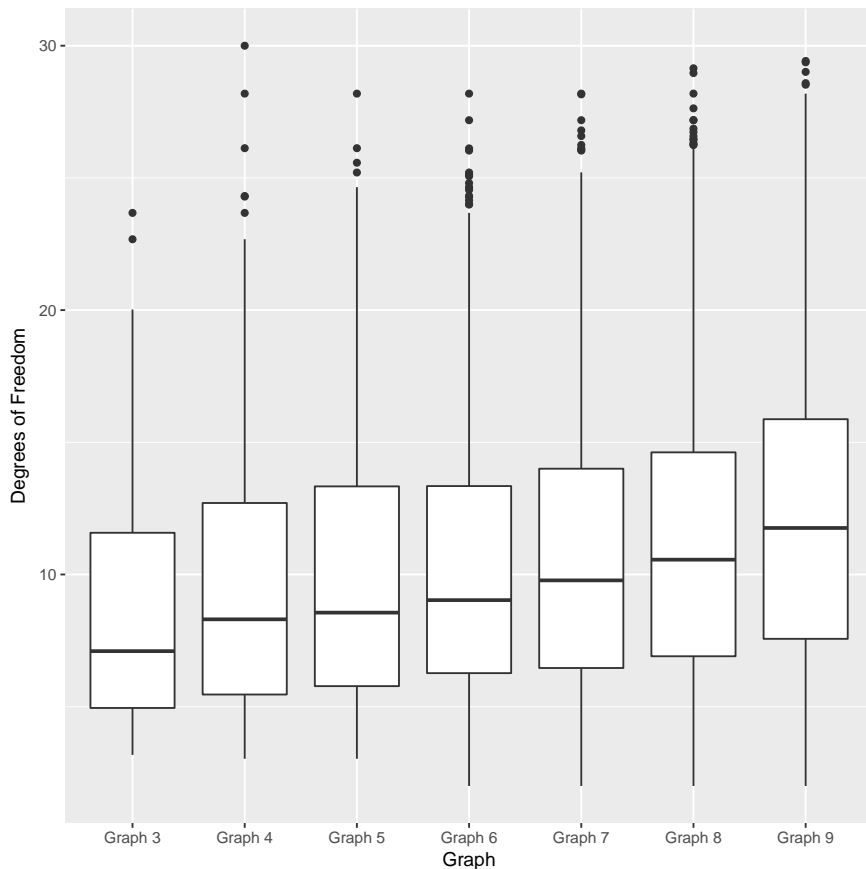


Figure 6.9: Distribution of the degrees of freedom as parameter for the Student's-t copula.

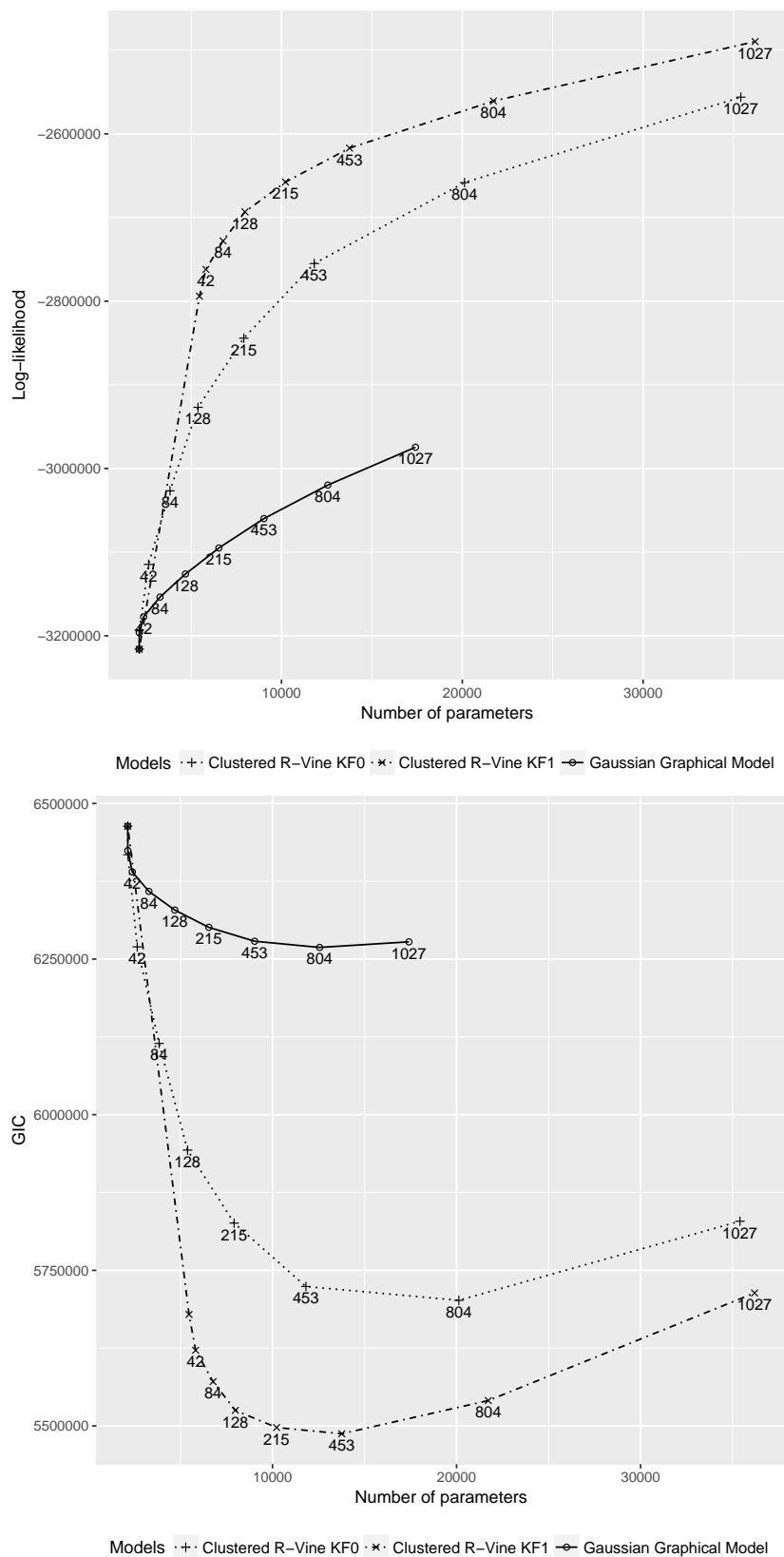


Figure 6.10: Comparison of clustered R-vines with fill-levels $k_F = 0, 1$ and Gaussian graphical model fitted with the graphical Lasso $\mathcal{G}_1, \dots, \mathcal{G}_9$: log-likelihood (top), GIC (bottom).

Since most of the degrees of freedom vary around 10 to 15, we clearly monitor and model heavy tailed data, which is quite important considering financial returns. If this property is not adequately taken into account, risk models are deemed to fail in times of heavy market turmoil when assets become highly correlated, leading to a significant reduction of diversification when it is most needed.

Finally, our models based on the graphs $\mathcal{G}_2, \dots, \mathcal{G}_9$ took between 12 hours and 2 days for estimation on a Linux Cluster with 32 cores. The Gaussian graphical model needs only several minutes for estimation in these dimensions. It is worth noting that other methods for estimating R-vines as the one of Dißmann et al. (2013) failed in these dimensions because of exceeding time or memory consumption.

This finishes the third chapter presenting the last of the three novel approaches. We have seen that this method decomposes the d -dimensional problem entirely different as the two previous approaches in the Chapters 4 and 5. We are now interested in comparing all the three presented approaches towards each other and also with the current benchmark algorithm of Dißmann et al. (2013).

Chapter 7

Comparison Studies Utilizing the Proposed Methods

We will now perform comparison studies of the proposed methods, applied to both simulated and real world data. The goal is to show the following assertions.

- (i) Our approaches are competitive with, or outperform the benchmark selection algorithm by Dißmann et al. (2013) as implemented in the `VineCopula` R-package of Schepsmeier et al. (2017). We show this for scenarios with moderate number of dimensions, i. e. $d \sim 100$ with respect to
 - goodness of fit measures as log-likelihood, AIC, BIC, mBIC and GIC, see (2.9),
 - computation time,
 - model parsimony in terms of number of parameters and non-independence copulas.
- (ii) Our approaches are the *only* numerically tractable option once the dimension exceeds $d \sim 500$, since the standard algorithm is becoming infeasible due to the required computation time, memory consumption, overly parametrized models or some or all of these issues.

In Section 7.1.1, we will discuss data preparation and numerical considerations regarding the used algorithms. Next, we will show a simulation study in $d = 85$ dimensions in Section 7.2, where we compare our three proposed approaches to the current benchmark algorithm. We will consider three differently sparse scenarios. Since we work on simulated data, we also know the *true* model parameters which we can compare ourselves to. In Section 7.3 we will compare the computational complexity of our proposed algorithms and the benchmark over data sets with up to $d = 1750$ dimensions to study the effect of increasing dimensions on the time complexity. Finally, in Section 7.4, we perform Value at Risk forecasting and backtesting for portfolios of $d = 400$ stocks using our proposed methods and the standard approach, to assess the effect on real world data.

The results in Section 7.2 and Section 7.3 have already been reported in Müller and Czado (2017b,c,a). The description of the data preparation follows Müller and Czado (2017a).

7.1 Set up of the Comparison

7.1.1 Data Preparation

For the subsequent data applications, we consider financial daily data. First of all, this is motivated by the fact that this kind of data is widely available, also in high dimensions. Furthermore, we need to carry out marginal modelling upfront, which is well understood for financial time series. More precisely, we consider closing prices of shares adjusted for dividends and splits S_t^j for $i = 1, \dots, n$ observations for $j = 1, \dots, d$ shares. In the next step, we calculate daily log returns by

$$R_i^j = \log \left(\frac{S_i^j}{S_{i-1}^j} \right) \text{ for } i = 2, \dots, n.$$

These log returns are then filtered for trend and seasonality, i. e. idiosyncratic behaviour. We use ARMA-GARCH(p, q) models, see Section 3.1.1, with $(p, q) \in \{0, 1\} \times \{0, 1\}$, i. e. four different specifications and allow for residuals distributed according to three different distributions, normal, Student's-t or skewed Student's-t, denoting the distribution by G_{ξ} . Here, the vector ξ denotes the corresponding parameters of the distribution G . Thus, in total we consider $4 \times 3 = 12$ models which we fit for each marginal time series and choose the best in terms of log-likelihood. Next, we compute standardized residuals by

$$x_{i,j} = \frac{R_i^j - \hat{R}_i^j}{\hat{\sigma}_j} \text{ for } j = 1, \dots, d \text{ and } i = 1, \dots, n - 1.$$

based on the estimated time series models, where $\hat{\sigma}_j^2, j = 1, \dots, d$ is the estimated variance of the error distribution. From these standardized residuals, we have two possible ways to proceed to ultimately obtain copula data. Either, we calculate the *empirical cumulative distribution function* \hat{F}_j of $X_{1,j}, \dots, X_{n-1,j}$ for $j = 1, \dots, d$ to obtain *copula data*

$$u_{i,j}^{np} = \hat{F}_j(x_{i,j}) \tag{7.1}$$

for $i = 1, \dots, n - 1$ and $j = 1, \dots, d$. Note that np stands for *nonparametric*. On the other hand side, we can also use the parametric distribution function of the residuals G_{ξ} and calculate

$$u_{i,j}^p = G_{\hat{\xi}_{j,j}}(x_{i,j})$$

for $i = 1, \dots, n - 1$ and $j = 1, \dots, d$, where the p stands for *parametric* and both the function G and also its parameter estimate depend on the corresponding time series $j = 1, \dots, d$. In the Sections 7.2 and 7.3 we will use non-parametrically estimated residuals and for the application in the Section 7.4 we will use parametrically estimated. The later is necessary since only parametrically estimated marginals allow us to re-transform simulated copula data back to the original marginal distribution. We will describe this proceeding more precisely in Section 7.4.

7.1.2 Numerical Implementation

We will briefly describe details of the numerical implementation concerning our analysis. This is to discuss settings we have made in the correspondingly used R-packages and give references. For obtaining the data, we used the R-package `quantmod` of Ryan and Ulrich (2017) to obtain financial time series in high dimensions accessing `https://www.yahoo.com`. Based on these, time series models were fitted using the `rugarch` R-package of Ghalanos (2015). Whenever R-vines are estimated using the algorithm of Dißmann et al. (2013), the R-package `VineCopula` of Schepsmeier et al. (2017) and more precisely, the function `RVineStructureSelect` was used. To estimate Gaussian DAGs as introduced in Chapter 4, we refer to the `bnlearn` R-package of Scutari (2010), and more precisely, the `hc` function for calculating DAGs using the *Hill-Climbing* algorithm. The default optimization criterion is Gaussian BIC. To calculate *Lasso regularization paths* as introduced in Chapter 5, we use the R-package `glmnet` of Friedman et al. (2010) and for undirected graphical models as used in Chapter 6, we refer to the `huge` R-package of Zhao et al. (2015). We note that all these packages are designed to work with Gaussian data, hence we transform the *copula data* to the *z-scale* before we apply the corresponding functions.

The plots are generated by the `ggplot2` R-package of Wickham (2009) and the `igraph` R-package of Csardi and Nepusz (2006), which was also used for analysis of graphs.

All numerical computations were performed on a Linux Cluster with 32 cores.

The algorithms we developed are contained in our R-package `VineCopulaHD` for high dimensional selection of copula models. Since our functions are based on and make use of the `VineCopula` package, all pair copula families implemented in the `VineCopula` package can be used in our package. Additionally, also a test for pairwise independence as described in the `VineCopula` package is possible to combine with our approaches and is used by default, both for Dißmann’s method as well as for our approaches.

7.2 Simulation Study

First of all, we are interested in evaluating our approaches for data with moderate dimensions, i. e. $d \sim 100$. In this case, the benchmark algorithm for vine copula selection of Dißmann et al. (2013) can also still be carried out and is reasonably fast. Since our approaches are however designed to work well in set-ups with sparse data, we want to account for this. Hence, we define three scenarios with different levels of sparsity. In accordance to the data preparation described in Section 7.1.1, we accumulated stock quotes of the S&P100 constituents from 01.01.2013 to 31.12.2016 and process them as previously described. This leads to observation size $n = 1007$ and $d = 85$ dimensions since 15 stocks dropped out of the index by the end of the observation period. After preprocessing the data and calculating copula data using the non-parametric transformation (7.1), we obtain $U \in [0, 1]^{n \times d}$.

We will use the following notational conventions to easy comparison of the different model selection algorithms we presented in this thesis. Since a vine copula model consists of a

structure, pair copulas and the corresponding pair copula parameters, we abbreviate

$$\mathcal{W}_p^m := (\mathcal{V}_p^m, \mathcal{B}_p^m(\mathcal{V}_p^m), \Theta_p^m(\mathcal{B}_p^m(\mathcal{V}_p^m))),$$

where m denotes a specific model selection algorithm. More precisely, we have

- for $m = \mathbf{Diss}$, the benchmark of Dißmann et al. (2013),
- for $m = \mathbf{DAG}$, the model selection described in Chapter 4,
- for $m = \mathbf{Lasso}$, the model selection described in Chapter 5, and more precisely,
 - for $m = \mathbf{Lasso-ST}$, using single thresholding,
 - for $m = \mathbf{Lasso-AT}$, using adaptive thresholding,
- for $m = \mathbf{Cluster}$, the model selection described in Chapter 6.

By p , we denote additional parameters which the corresponding model selection algorithm may have. If we want to make clear that this model is estimated using data U , we write

$$\mathcal{W}_p^m(U).$$

Considering again the data set $U \in [0, 1]^{1007 \times 85}$, we use Dißmann's algorithm to fit three different vine copula models

$$\mathcal{W}_2^D, \mathcal{W}_5^D, \mathcal{W}_{10}^D,$$

to the data, imposing a 2, 5 and 10-truncation. We allow for all parametric pair copula families implemented in the `VineCopula` R-package of Schepsmeier et al. (2017) and find the pair copulas in the first tree as given in Table 7.1. Additionally, we perform an independence test for each pair copula at the significance level $\alpha = 0.05$. Clearly, the 2-truncation is the most parsimonious model, however, also the 5 and 10-truncation are quite sparse models compared to a full model with 84 trees. From these fitted models, we simulate $M = 50$ replications

$$\begin{aligned} &U_1^2, \dots, U_M^2, \\ &U_1^5, \dots, U_M^5, \\ &U_1^{10}, \dots, U_M^{10}, \end{aligned}$$

with $n = 1000$ observations each. Onto these data sets, we fit vine copula models using the different approaches presented in this thesis. We use $m = D$ and $p = 2, 5, 10$ to denote the *true* models with the corresponding truncation levels in the corresponding scenarios. This proceeding has the advantage that we can now consider the models $\mathcal{W}_2^D, \mathcal{W}_5^D, \mathcal{W}_{10}^D$ as *true models* from which the simulated data arise. We briefly summarize the pair copula families on the first tree of the true models. These pair copulas indicate non-Gaussian, i. e. heavy tailed and asymmetric dependence. For the degrees of freedom of the Student's-t copula, we see a mean of 6.4, which indicates heavy tails.

These data sets serve now as basis to compare our approaches. More precisely, for each scenario $s = 2, 5, 10$ and each replication $i = 1, \dots, M$ we compare the following models.

Copula family	absolute number
Student's-t	58
Survival BB1	15
Survival BB8	6
Survival Gumbel	3
BB1	1
Frank	1
Total	84

Table 7.1: Most occurring pair copula families in R-vine tree T_1 of $\mathcal{W}_2^D, \mathcal{W}_5^D, \mathcal{W}_{10}^D$.

Model name	Notation	Parameters	Description
True	$\mathcal{W}_s^D(U_i^s)$	-	true R-vine model we simulate from
Dißmann	$\mathcal{W}^{Diß}(U_i^s)$	-	R-vines fitted using Dißmann's method, see Section 2.2
k-DAG	$\mathcal{W}_k^{DAG}(U_i^s)$	$k = 2, 3, 4$	R-vine representations of k -DAGs, see Chapter 4
Lasso-ST-λ	$\mathcal{W}_\lambda^{Lasso-ST}(U_i^s)$	$\lambda = 0.2^4, 0.25^4$	R-vine representations of SEMs calculated with the Lasso, see Chapter 5 using the single threshold approach
Lasso-AT-μ	$\mathcal{W}_\mu^{Lasso-AT}(U_i^s)$	$\mu = 0.1, 0.2$	R-vine representations of SEMs calculated with the Lasso, see Chapter 5 using the adaptive threshold approach
Cluster-d_T-k_F	$\mathcal{W}_{d_T-k_F}^{Cluster}(U_i^s)$	$d_T = 25, 50, 75$	R-vines based on clusterings, see Chapter 6 using $J = 15$ undirected graphical models generated with the graphical Lasso and considering threshold dimensions d_T and fill-level $k_F = \lceil \log(d) \rceil = \lceil \log(85) \rceil = 5$

Table 7.2: Overview of compared models in simulation study.

Since we compare $M = 50$ values for each model, for example log-likelihood, we use *box plots* to visualize the results. We will consider the goodness of fit measures log-likelihood, AIC, BIC, mBIC and GIC see (2.9), as well as the computation time in minutes and the number of parameters and non-independence copulas in the corresponding models. The simulation scheme is visualized in Figure 7.1.

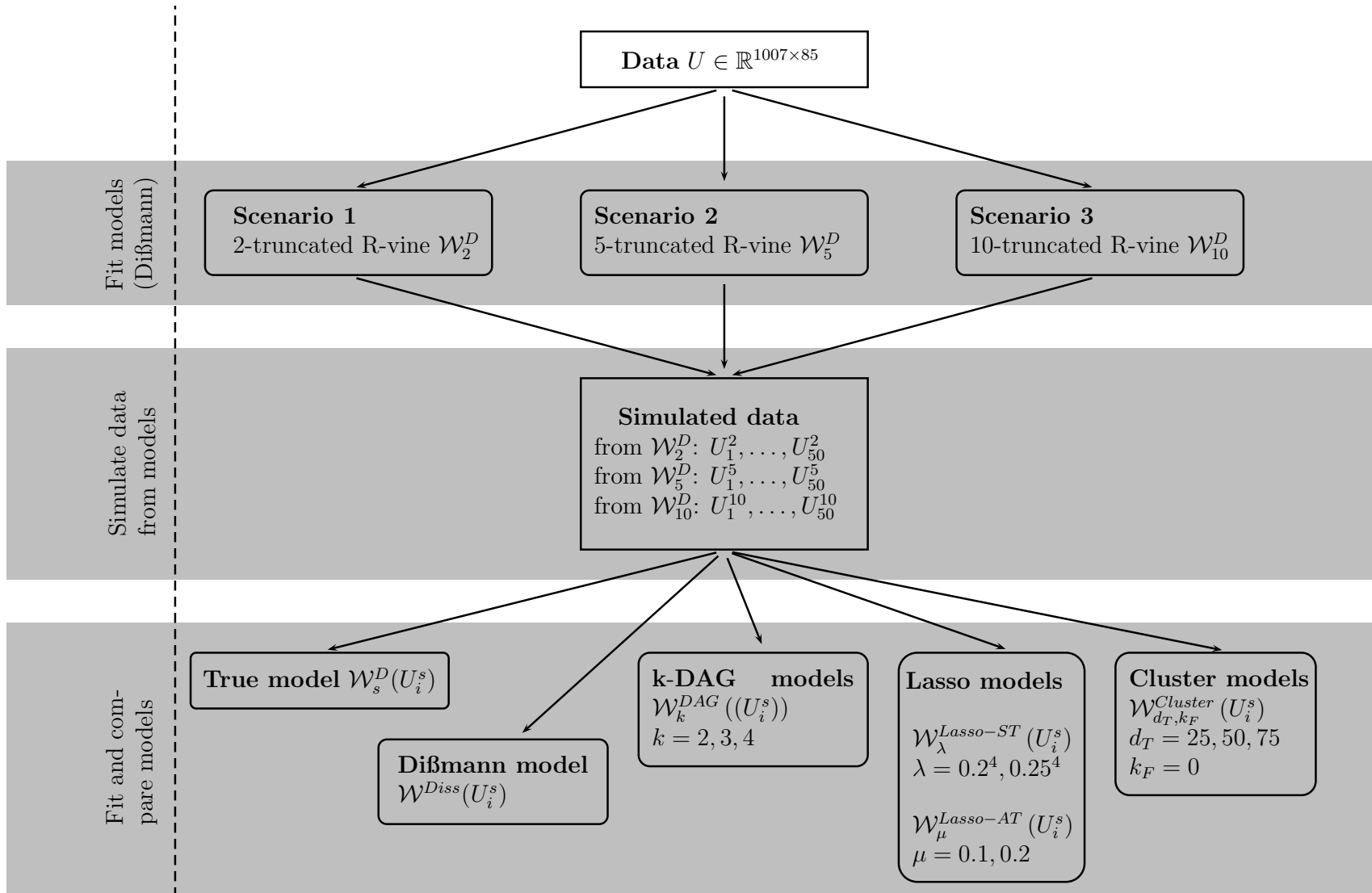


Figure 7.1: Scheme of simulation study.

Scenario \mathcal{W}_2^D : 2-truncation

We start with the most parsimonious scenario \mathcal{W}_2^D based on a 2-truncated R-vine. First we consider the log-likelihood.

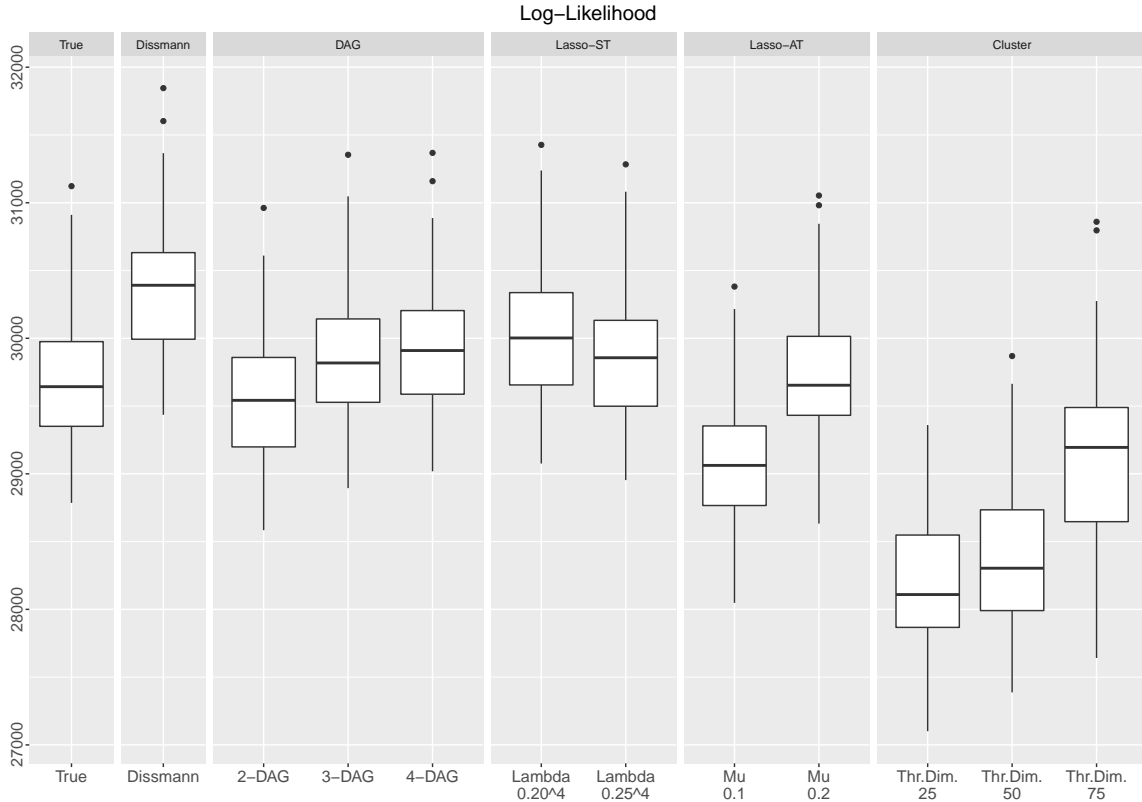


Figure 7.2: Log-likelihood for Scenario \mathcal{W}_2^D : Comparison of **True model**, **Dißmann**, **k-DAG** for $k = 2, 3, 4$, **Lasso-ST- λ** for $\lambda = 0.2^4, 0.25^4$, **Lasso-AT- μ** for $\mu = 0.1, 0.2$ and **Cluster- d_T -5** for **Threshold Dimension (Thr. Dim.)** $d_T = 25, 50, 75$.

First of all, we see that **Dißmann's** algorithm outperforms all other competitors with respect to log-likelihood, see Figure 7.2. It is worth noting that the attained log-likelihood is also higher than the likelihood of the true model from which we simulate. This indicates over-fitting. The closest competitors are the **DAG** and the **Lasso-ST** models, where the other algorithms attain significantly lower values in log-likelihood. However, it is interesting that the **Lasso-ST** and the **Cluster** method are either shrinking the number of used parameters quite inflexibly or perform a not very sensible clustering. For such a small dimension, breaking up the dependence structure in sub-R-vines is not yet necessary. We next consider penalized goodness of fit values as AIC and BIC in Figure 7.4 and mBIC and GIC in Figure 7.5. We see that in terms of AIC, **Dißmann's** algorithm is still ahead, but the distances are decreasing. This is even more the case in terms of BIC where only the **Lasso-AT** and the **Cluster** method fall behind. Considering mBIC, the picture is quite similar to BIC. However, once we look at GIC, the picture changes and **Dißmann's** method is outperformed by the **DAG** models, the **Lasso-ST** and **Lasso-AT-0.1** and the

Cluster-75-5 method. This becomes even more intriguing considering the number of parameters and the number of non independence copulas as reported in Figure 7.6. We see that **Dißmann’s** model has a significantly higher number of parameters than all other competitor methods, which leads to the decline in goodness of fit with respect to strongly penalizing measures as GIC. Apart from the **Cluster-50-5** and **Cluster-75-5** method, all other approaches have a higher number of parameters compared to the true model. This approach and parametrization leads to quite parsimonious models, which can also be seen in terms of computation time, Figure 7.3. **Dißmann’s** method uses by far the most time. Especially the **Lasso-ST** is very fast. The computation time of the **Cluster** method is not as low, however, the strength of this approach will come into place in much higher dimensions. We also clearly see that for the **k-DAGs**, with increasing number of k , we have increasing computation time. We will now compare more in depth the associated correlation matrices of the corresponding **k-DAG** models.

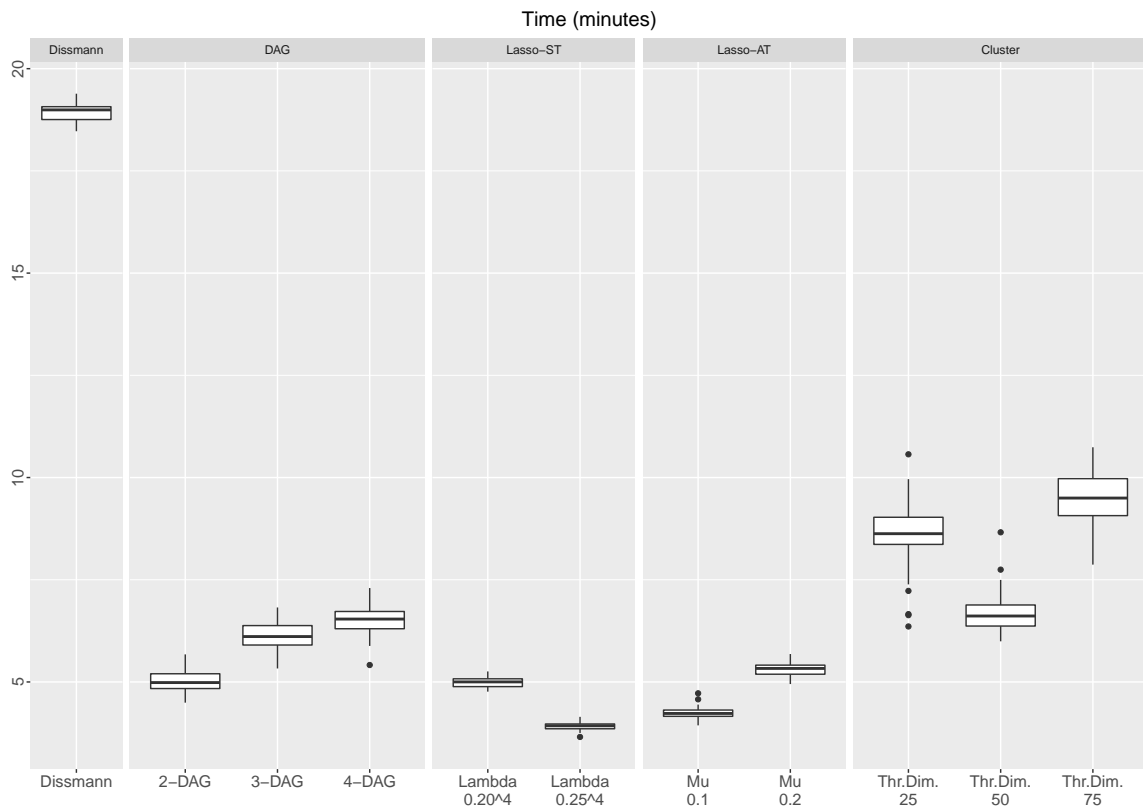


Figure 7.3: Computation time in minutes for Scenario \mathcal{W}_2^D with 2-truncation: Comparison of **True model**, **Dißmann**, **k-DAG** for $k = 2, 3, 4$, **Lasso-ST- λ** for $\lambda = 0.2^4, 0.25^4$, **Lasso-AT- μ** for $\mu = 0.1, 0.2$ and **Cluster- d_T -5** for **Threshold Dimension (Thr. Dim.)** $d_T = 25, 50, 75$.

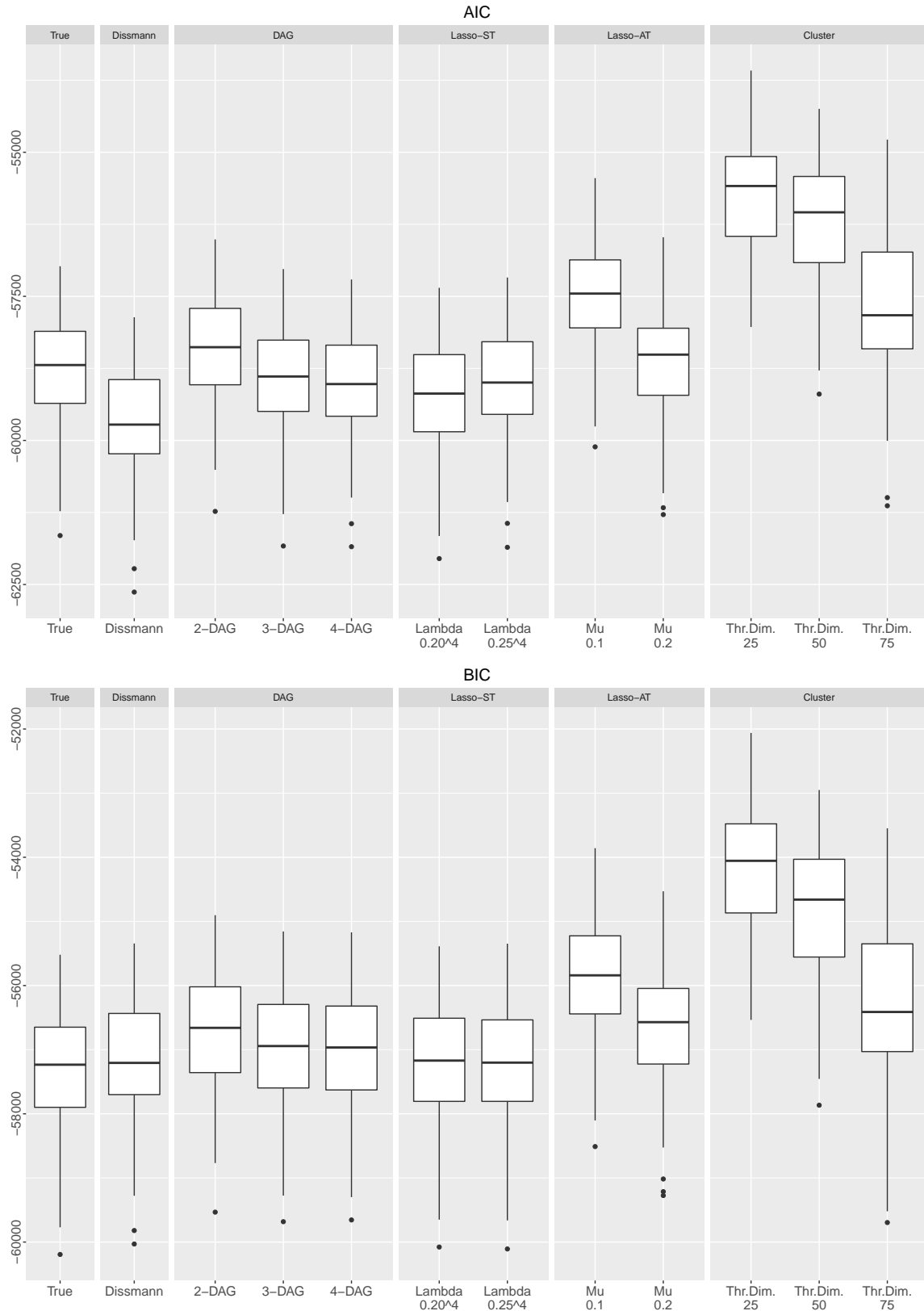


Figure 7.4: AIC (upper panel) and BIC (lower panel) for Scenario \mathcal{W}_2^D with 2-truncation: Comparison of **True model**, **Dißmann**, **k-DAG** for $k = 2, 3, 4$, **Lasso-ST- λ** for $\lambda = 0.2^4, 0.25^4$, **Lasso-AT- μ** for $\mu = 0.1, 0.2$ and **Cluster- d_T -5** for **Threshold Dimension (Thr. Dim.)** $d_T = 25, 50, 75$.

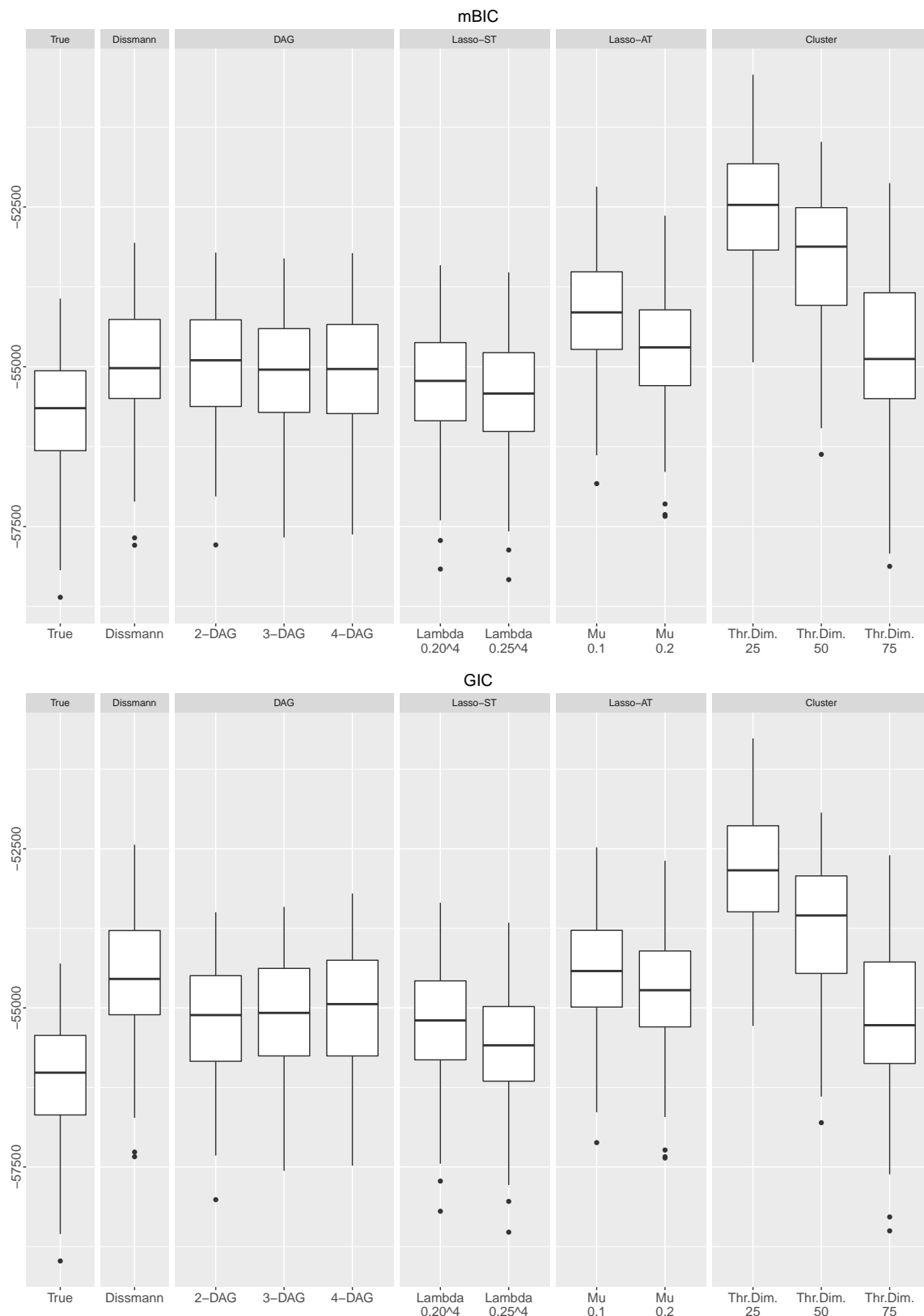


Figure 7.5: mBIC (upper panel) and GIC (lower panel) for Scenario \mathcal{W}_2^D with 2-truncation: Comparison of **True model**, **Dißmann**, **k-DAG** for $k = 2, 3, 4$, **Lasso-ST- λ** for $\lambda = 0.2^4, 0.25^4$, **Lasso-AT- μ** for $\mu = 0.1, 0.2$ and **Cluster- d_T -5** for **Threshold Dimension (Thr. Dim.)** $d_T = 25, 50, 75$.

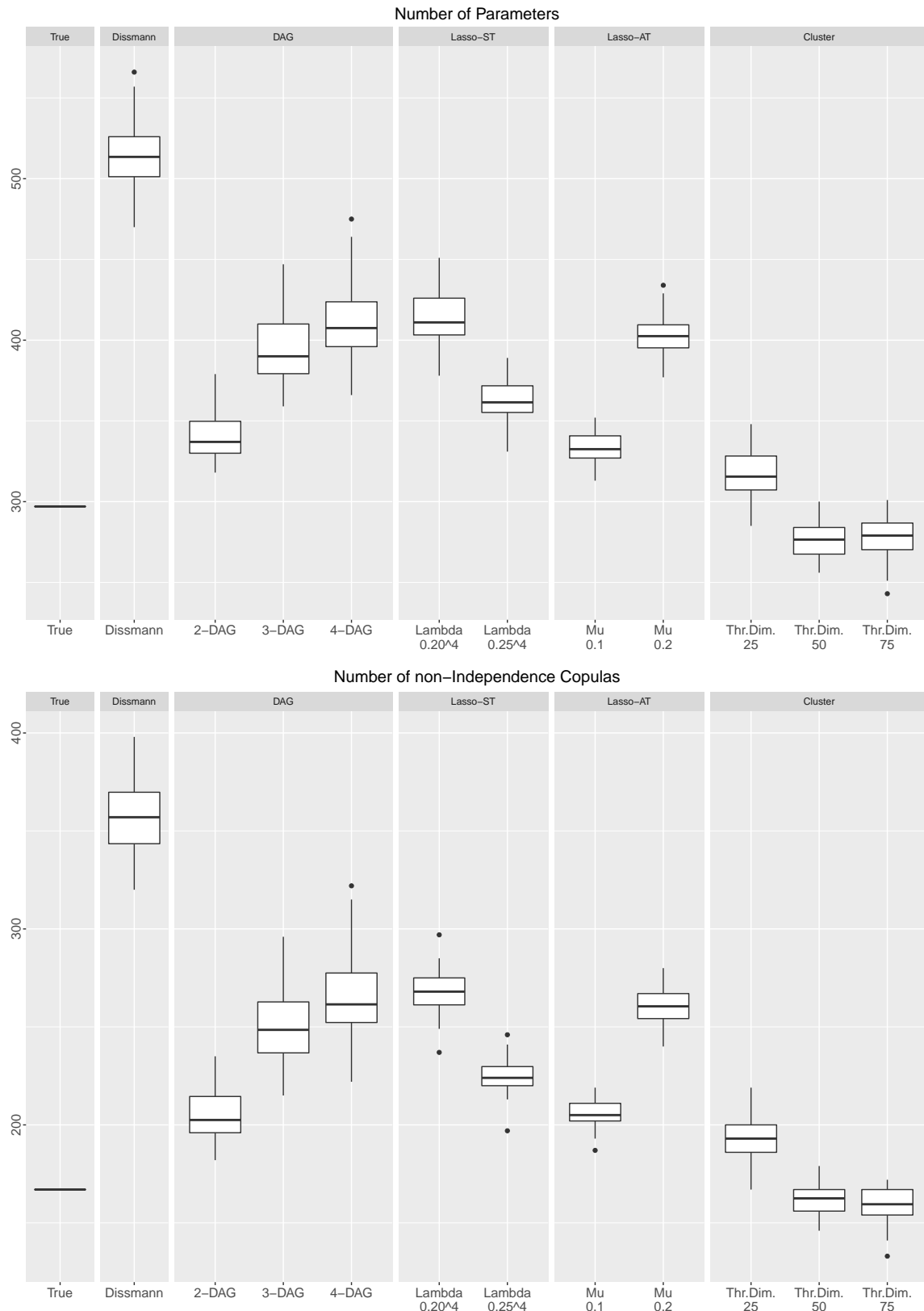


Figure 7.6: Number of parameters (upper panel) and number of non-independence copulas (lower panel) for Scenario \mathcal{W}_2^D with 2-truncation: Comparison of **True** model, **Dißmann**, **k-DAG** for $k = 2, 3, 4$, **Lasso-ST- λ** for $\lambda = 0.2^4, 0.25^4$, **Lasso-AT- μ** for $\mu = 0.1, 0.2$ and **Cluster- d_T -5** for **Threshold Dimension (Thr. Dim.)** $d_T = 25, 50, 75$.

Comparison of Associated Correlation Matrices

Another interesting fact about the DAG models can be seen comparing the correlation matrices. First, we have the simulated data on the z -scale, denote $Z_i = \phi^{-1}(U_i^s)$ and calculate its corresponding correlation matrix $\hat{\Sigma}$. Recalling the approach from Chapter 4, we first estimate a Gaussian DAG with at most k parents on this data, denote it by

$$\mathcal{G}_k \text{ for } k = 2, 3, 4.$$

Each of these DAGs induces a structured correlation matrix, denote by $\Sigma_{\mathcal{G}_k}$, related to the graphical structure of the DAG. An algorithm how to construct a covariance matrix induced by a DAG is given in Appendix A.3, Algorithm 4 and based on Rütimann et al. (2009). The covariance matrix can then easily be standardized to obtain a correlation matrix. Next, based on this Gaussian DAG, we find an R-vine representation $\mathcal{W}(\mathcal{G}_k)$ and equip this R-vine with arbitrary pair copulas based on maximum likelihood estimation, but also taking into account the conditional independencies as in the DAG. This R-vine is then transformed into the *nearest Gaussian R-vine*. For this, the Kendall's τ of each edge in the R-vine is transformed into a (partial) correlation, using the formula in Example 2.4, and the corresponding pair copula families are exchanged to Gaussian. Denote this Gaussian R-vine by $\mathcal{W}^G(\mathcal{G}_k)$. Based on the correlations and partial correlations, we obtain the correlation matrix described by this Gaussian R-vines and denote it by $\Sigma_{\mathcal{W}^G(\mathcal{G}_k)}$. Thus, we have three different correlation matrices and can compare these using the Kullback-Leibler divergence \mathcal{D} given by (3.4) between the associated correlation matrices under normality, see Section 3.1.3. These divergence values describe the following.

- (a) $\mathcal{D}(\hat{\Sigma}, \Sigma_{\mathcal{G}_k})$ describes how well the correlation matrix $\hat{\Sigma}$ is modelled by the Gaussian DAG \mathcal{G}_k for $k = 2, 3, 4$. This divergence is expected to decrease for increasing k as more edges are allowed.
- (b) $\mathcal{D}(\Sigma_{\mathcal{G}_k}, \Sigma_{\mathcal{W}^G(\mathcal{G}_k)})$ describes how well our algorithm finds an approximating R-vine $\mathcal{W}(\mathcal{G}_k)$ for the Gaussian DAG \mathcal{G}_k for $k = 2, 3, 4$. This divergence is expected to be small and vary only slightly with increasing k since our algorithm should find feasible representations for any k . For $k = 1$, a perfect representation is possible and hence, there the divergence should be zero. However, for $k \geq 2$, the divergence should decrease as more edges in the DAG give also more flexibility for the R-vine.
- (c) $\mathcal{D}(\hat{\Sigma}, \Sigma_{\mathcal{W}^G(\mathcal{G}_k)})$ describes how well the sample correlation matrix $\hat{\Sigma}$ is modelled by the Gaussian R-vine representation. As for (a), we expect this divergence to decrease since for more available edges, the model fit should increase. Of course, this is based on the assumption that our algorithm for representing Gaussian DAGs as R-vines works sufficiently well. Additionally, we also consider the resulting R-vine estimated by Dißmann's algorithm, calculate the nearest Gaussian R-vine and the corresponding correlation matrix, denote Σ_D .

The connection between these divergence measures is also depicted in Figure 7.7.

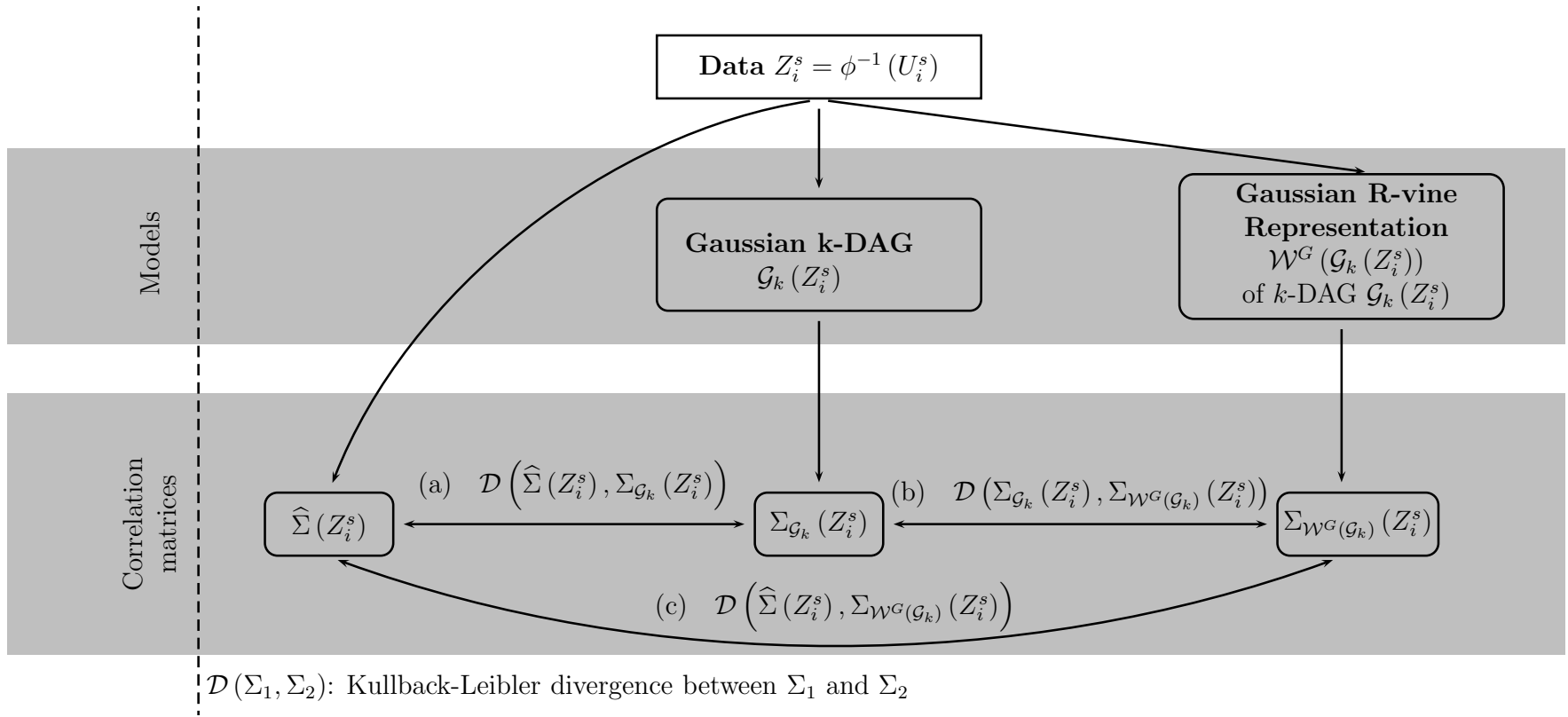


Figure 7.7: Scheme of the comparison of associated correlation matrices.

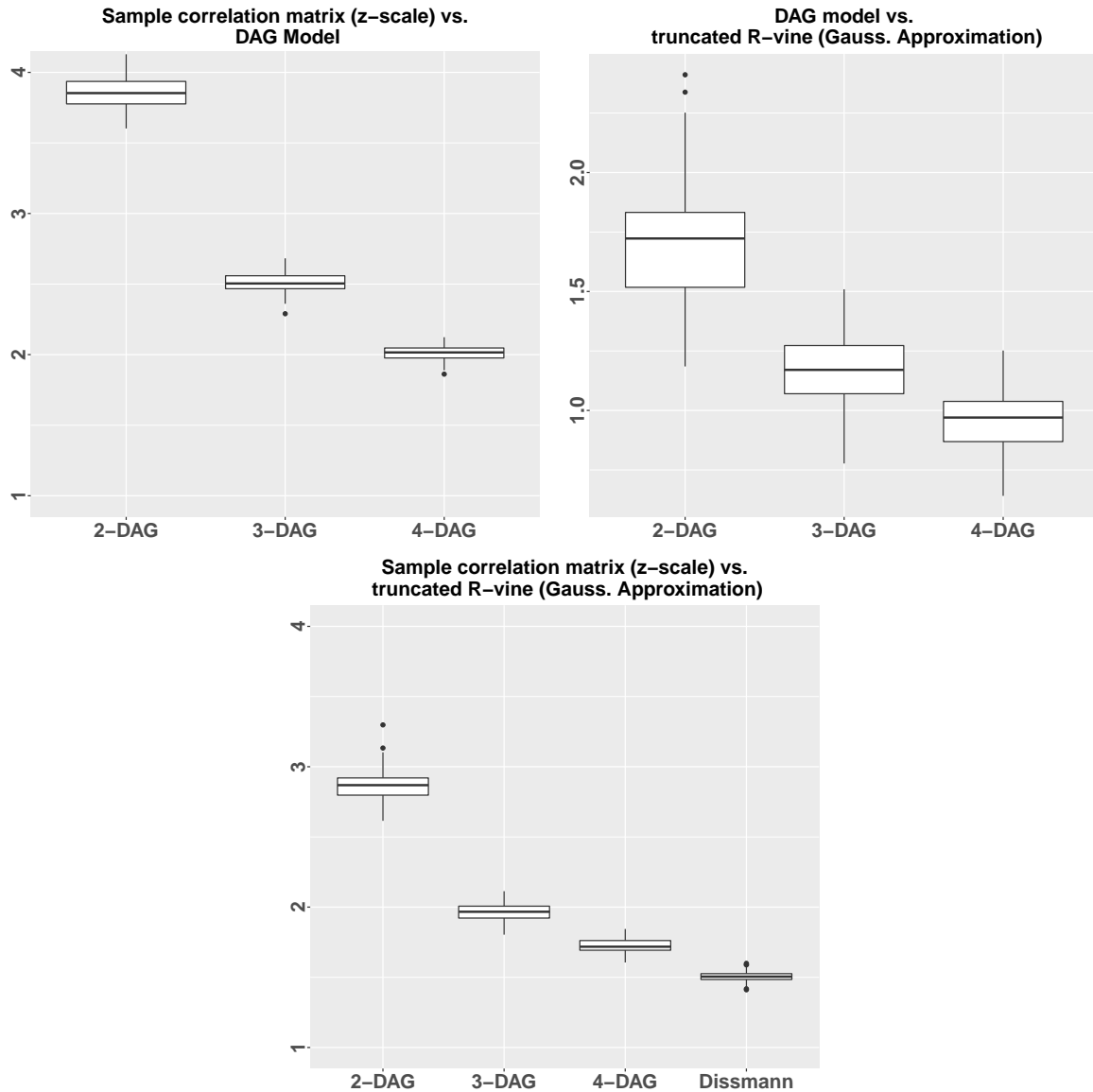


Figure 7.8: Kullback-Leibler divergence for Scenario \mathcal{W}_2^D with 2-truncation between (a) $\hat{\Sigma}$ and $\Sigma_{\mathcal{G}_k}$ (upper left), (b) $\Sigma_{\mathcal{G}_k}$ and $\Sigma_{\mathcal{W}^G(\mathcal{G}_k)}$ (upper right), (c) $\hat{\Sigma}$ and $\Sigma_{\mathcal{W}^G(\mathcal{G}_k)}$, Σ_D (lower center) for $k = 2, 3, 4$.

We draw the conclusion that a **2-DAG** is not a good approximation of the sample correlation matrix, but we obtain better fit with **3** and **4-DAG**. The low values in the center plot indicate that our approach maps the structure between DAG and R-vine representation well. In the right plot, **Dißmann's** algorithm obtains a smaller divergence to the sample correlation matrix on the *z-scale*, however, for higher k , the divergence decreases also for ***k*-DAGs**.

Next, we will move on and compare for a specific simulated data set the first R-vine trees estimated by the models in our comparison.

Comparison of Associated First R-vine Trees

We will also consider more explicitly the graphical models chosen by our algorithms, i. e. the R-vine trees. To this end, we consider the first data set U_1^2 sampled from the 2–truncated R-vine \mathcal{W}_2^D and the following **selection algorithms** with their corresponding first R-vine trees.

- **Dißmann:** T_1^{Diss}
- **4-DAG:** T_1^{DAG}
- **Lasso-ST-0.25⁴:** T_1^{Lasso}
- **Cluster-75-5:** $T_1^{Cluster}$

The choice of these models is based on the fact that they attained best fit in their corresponding model classes with respect to GIC.

First note that we have ten industry sectors in the S&P100 data set on $d = 85$ dimensions. These are listed in Table 7.3. Secondly, we consider how each first R-vine tree selected by











Sector name	Sector color	Sector size
Consumer Discretionary		11
Consumer Staples		8
Energy		6
Financials		15
Health Care		13
Industrials		15
Information Technology		8
Materials		3
Telecommunications Services		2
Utilities		4
Total		85

Table 7.3: Simulation study: Overview of industry sectors in the S&P100 subset with $d = 85$ dimensions.

the aforementioned four models differs from the first R-vine tree in the true model. For this, we plot the corresponding first tree and indicate for each edge with a different color, whether it is chosen in the true model, by the corresponding **selection algorithm** or in both. More precisely,

- the edge is selected in the **true model** but not in the **selection algorithm**,
- the edge is selected in both **true model** and in the **selection algorithm**,
- the edge is not selected in the **true model** but in the **selection algorithm**.

The corresponding results are shown in Figure 7.9 and Figure 7.10. Additionally, in Table 7.4, we illustrate the corresponding numbers of red, blue and green edges for the four graphs. The table is read as following. For example, the **4-DAG** shares 64 edges out of 84 edges, i. e. about 76 % of the edges in the true model. Thus, the remaining 20 edges which are contained in the true model are replaced by different edges selected by the **4-DAG** to obtain a tree with in total 84 edges. We see that in all the first R-vine trees,

Comparison between true model and	selection algorithm but not true model	both true model and selection algorithm
Dißmann	15 (18 %)	69 (82 %)
4-DAG	20 (24 %)	64 (76 %)
Lasso-ST-0.25⁴	18 (21 %)	66 (79 %)
Cluster-75 - 5	11 (13 %)	73 (87 %)

Table 7.4: Comparison of edge occurrence in scenario \mathcal{W}_D^2 with 2-truncation between **true model** T_1 of \mathcal{W}_2^D and **Dißmann** T_1^{Diss} , **4-DAG**: T_1^{DAG} , **Lasso-ST-0.25⁴**: T_1^{Lasso} and **Cluster-75**: $T_1^{Cluster}$.

the intra-sectoral dependence is strong, depicted by the blue edges. The largest overlap between a **selection algorithm** and the **true model** exists for the **Cluster-75** $T_1^{Cluster}$ model, where we have only 11 edges which are not present in both models. The smallest overlap exists for the **4-DAG** T_1^{DAG} with 20 edges which are not present in both models. We see this for example in the *Financials* sector located in the bottom left of the plots. There, the corresponding plot for the **4-DAG**: T_1^{DAG} , see Figure 7.9, right panel, does not exhibit many blue edges, i. e. not many common edges. On the contrary, considering the **Cluster-75** $T_1^{Cluster}$ model, we see that most non-blue edges are connecting different sectors, see Figure 7.10, right panel.

Finally, we also examine the properties of the graphical models estimated by the graphical Lasso and used by the **Cluster** method.

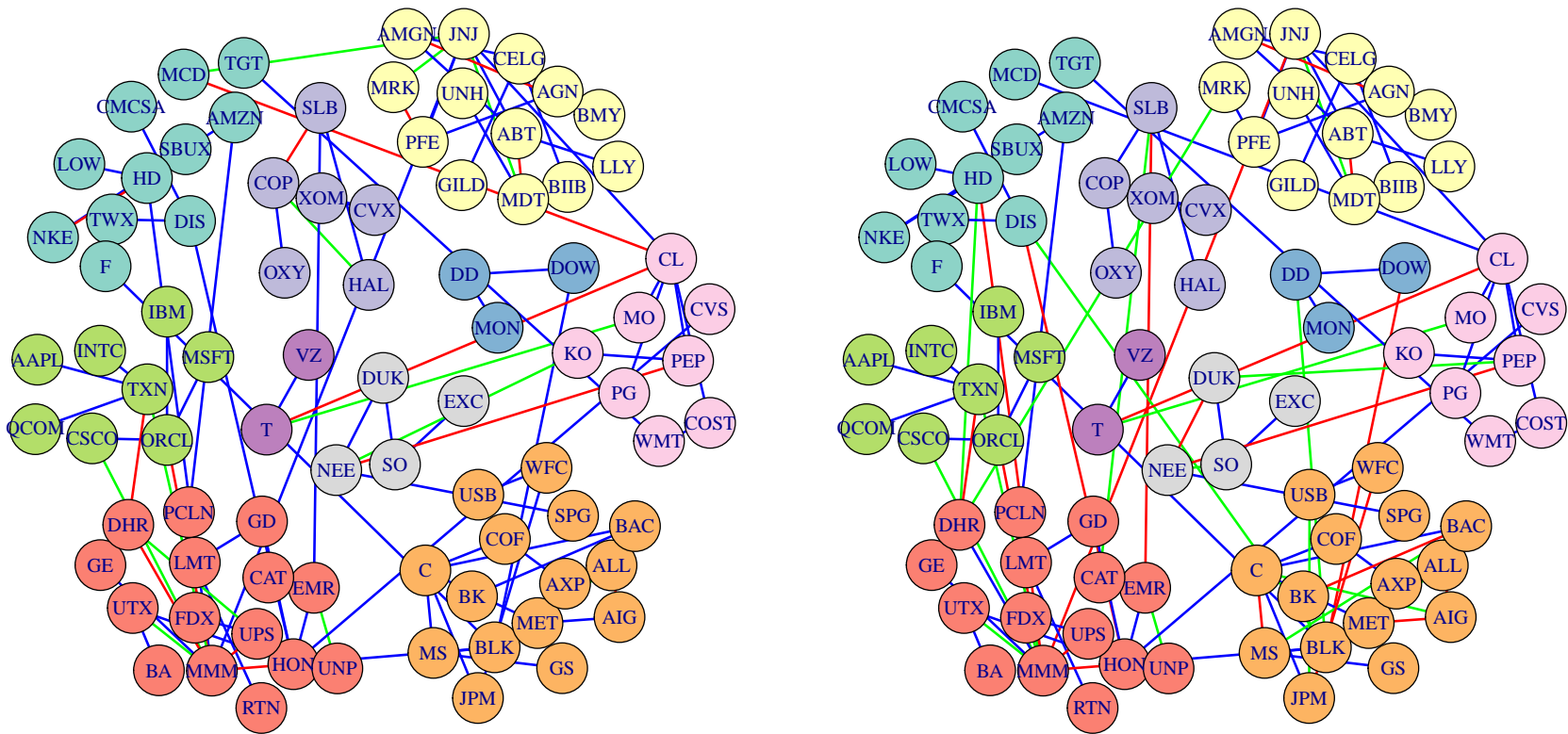


Figure 7.9: Comparison of edge occurrence in scenario \mathcal{W}_2^D with 2-truncation between **true model** T_1 of \mathcal{W}_2^D and **Dißmann** T_1^{Diss} (left panel) and between **true model** T_1 of \mathcal{W}_2^D and **4-DAG**: T_1^{DAG} (right panel).

Color coding:

- the edge is selected in the **true model** but not in the **selection algorithm**
- the edge is selected in both **true model** and in the **selection algorithm**
- the edge is not selected in the **true model** but in the **selection algorithm**

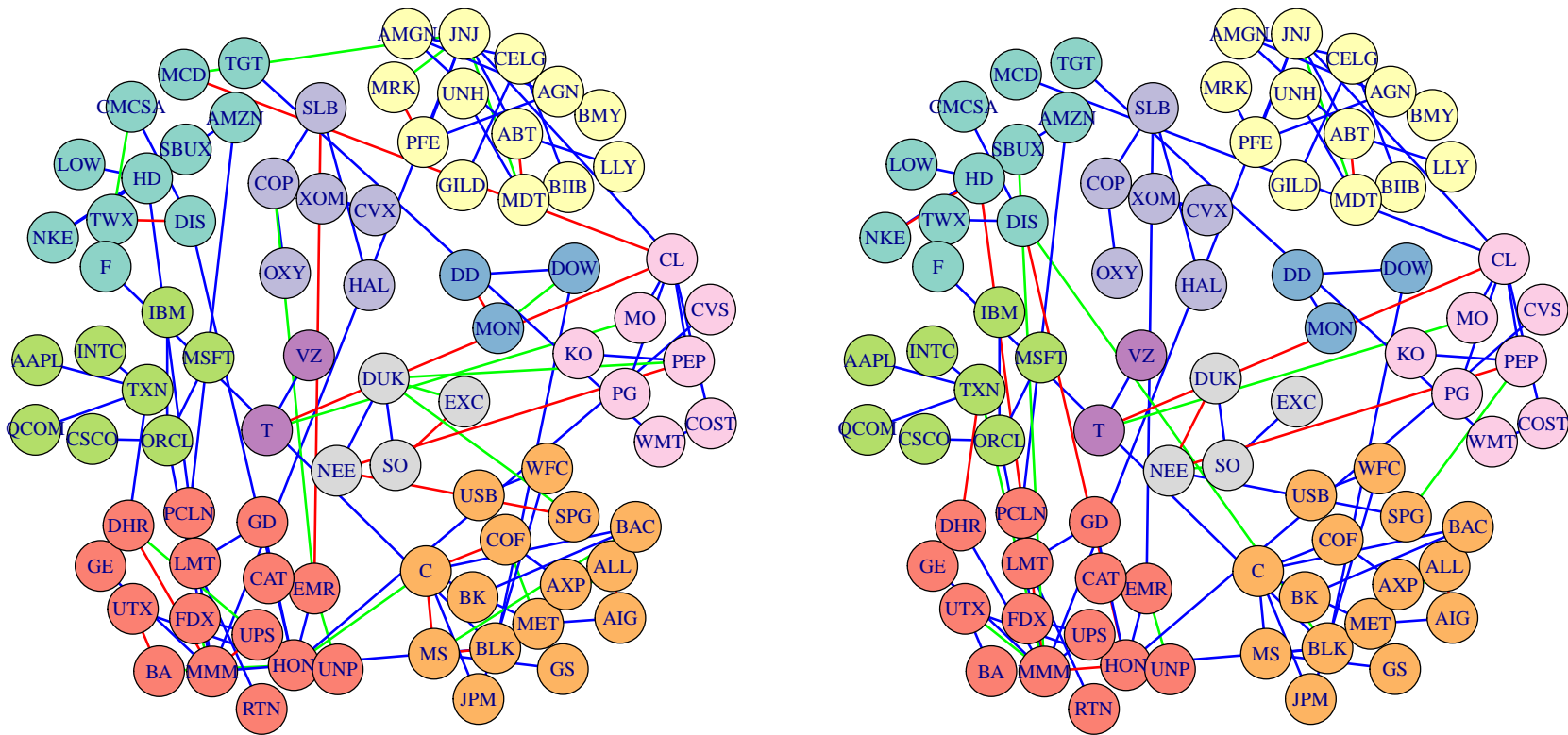


Figure 7.10: Comparison of edge occurrence in scenario \mathcal{W}_2^D with 2-truncation between **true model** T_1 of \mathcal{W}_2^D and **Lasso-ST-0.25⁴**: T_1^{Lasso} (left panel) and between **true model** T_1 of \mathcal{W}_2^D and **Cluster-75 - 5**: $T_1^{Cluster}$ (right panel).

Color coding:

- the edge is selected in the **true model** but not in the **selection algorithm**
- the edge is selected in both **true model** and in the **selection algorithm**
- the edge is not selected in the **true model** but in the **selection algorithm**

Comparison of Associated Graphical Models

We can also more thoroughly inspect the underlying graphical models which are used by the

- **Cluster-25-5,**
- **Cluster-50-5,**
- **Cluster-75-5,**

models. More precisely, we evaluate the maximum component sizes δ_T^i and the number of connected components p_T^i of the chosen partitions T_i for all 50 replications $i = 1, \dots, 50$. There, we observe that for higher threshold dimension, the number of the connected components decrease while the maximum component sizes naturally increases, see Figure 7.11. We will now continue considering the other two scenarios, \mathcal{W}_5^D and \mathcal{W}_{10}^D , however

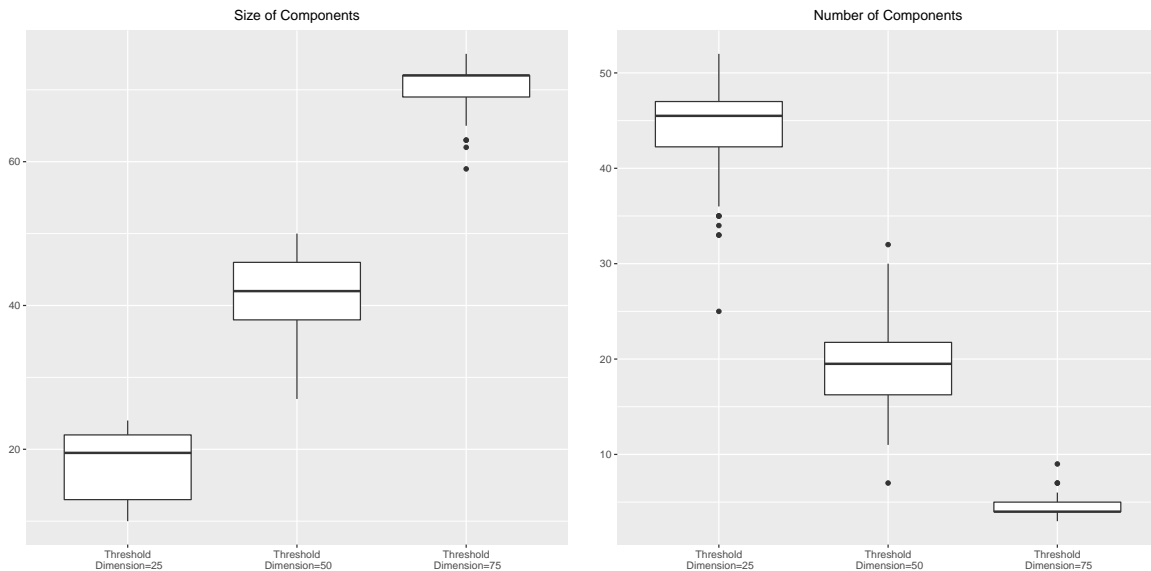


Figure 7.11: Distribution of maximum component sizes δ_T^i (left) and distribution of the number of connected components p_T^i (right) for each of the 50 replications, $i = 1, \dots, 50$ for different threshold dimensions $d_T = 25, 50, 75$ in scenario \mathcal{W}_D^2 with 2-truncation.

only comparing the corresponding goodness of fit values, number of parameters and non-independence copulas and the computation time.

Scenario \mathcal{W}_5^D : 5-truncation

We see very similar results as in the scenario \mathcal{W}_2^D . More precisely, the **Lasso-ST** models as well as the **3** and **4-DAG** models can compete with **Dißmann's** algorithm to attain the **true** model. The most significant difference is that now all competitors fit models with more parameters, as also the **true** model has more parameters, compared to Scenario 2, see Figure 7.12. However, **Dißmann's** method is still prone to over-fitting as we see

in terms of mBIC and GIC, Figure 7.13. It is however also to be noted, that in scenarios with such moderate dimension, this effect is not so problematic as **Dißmann's** approach also yields high log-likelihood to set this large number of parameters off. Yet, we see from the computation time that the large difference between **Dißmann's** method and for example, the **Lasso** approaches, will become tremendous in several hundreds or thousands of dimensions. We will now finally consider the scenario \mathcal{W}_{10}^D before we continue with the next section.

Scenario \mathcal{W}_{10}^D : 10-truncation

The same statements as made for the previous scenarios can also be made for this scenario, however, we see the differences between the algorithms diminishing. This can be explained since this scenario is not really sparse any more, and hence, there is no sparsity to exploit. Additionally, in terms of GIC, we would choose the **Lasso-ST-0.2⁴** model. This is also the most accurate model in terms of number of parameters and also quite beneficial with respect to computation time.

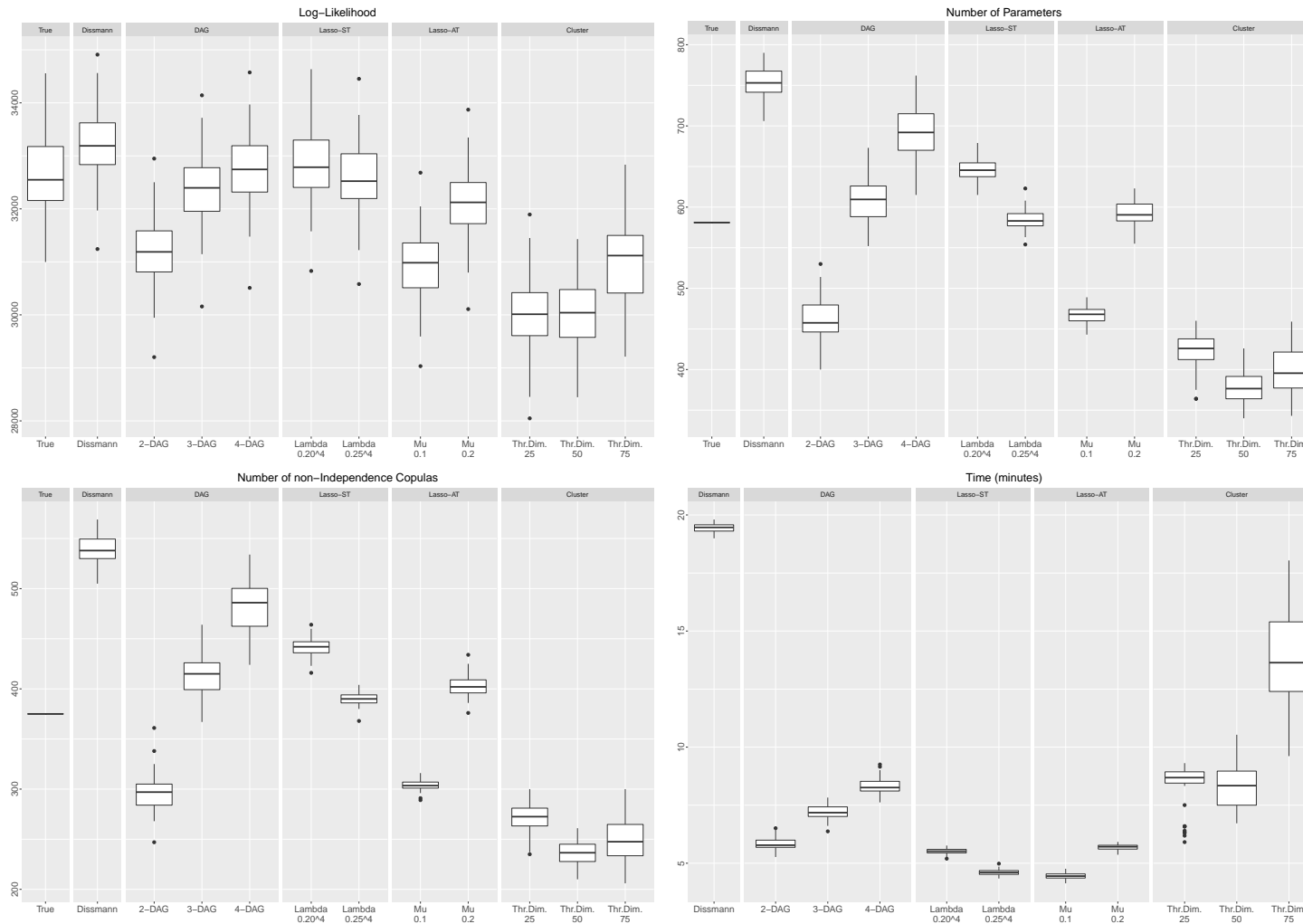


Figure 7.12: Log-likelihood (upper left), number of parameters (upper right), number of non-independence copulas (lower left) and computation time in minutes (lower right) for scenario \mathcal{W}_5^D : Comparison of **True model**, **Dißmann**, **k-DAG** for $k = 2, 3, 4$, **Lasso-ST- λ** for $\lambda = 0.2^4, 0.25^4$, **Lasso-AT- μ** for $\mu = 0.1, 0.2$ and **Cluster- d_T -5** for **Threshold Dimension (Thr. Dim.)** $d_T = 25, 50, 75$.

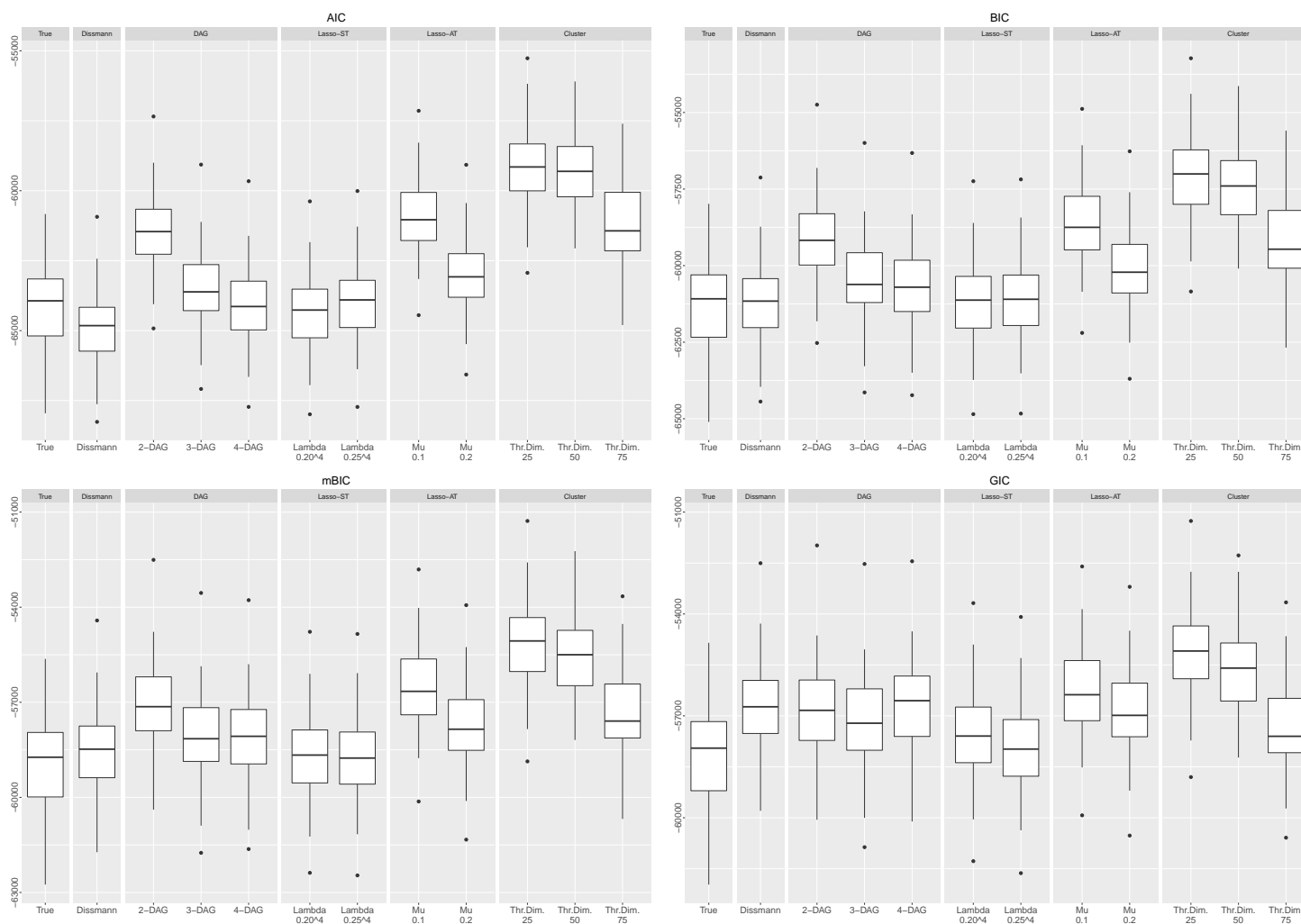


Figure 7.13: AIC (upper left), BIC (upper right), mBIC (lower left) and GIC (lower right) for scenario \mathcal{W}_5^D : Comparison of **True model**, **Dissmann**, **k-DAG** for $k = 2, 3, 4$, **Lasso-ST- λ** for $\lambda = 0.2^4, 0.25^4$, **Lasso-AT- μ** for $\mu = 0.1, 0.2$ and **Cluster- d_T -5** for **Threshold Dimension (Thr. Dim.)** $d_T = 25, 50, 75$.

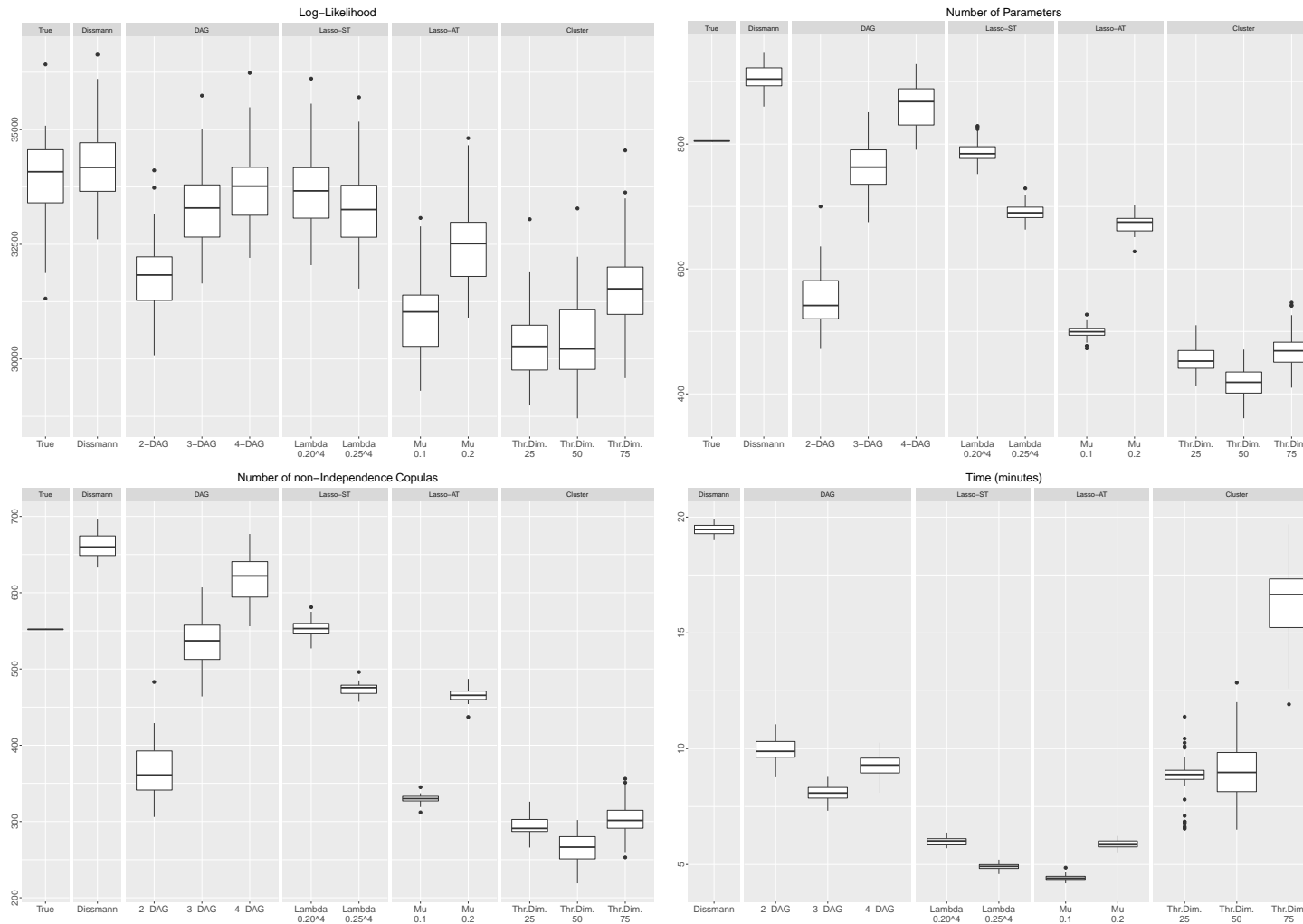


Figure 7.14: Log-likelihood (upper left), number of parameters (upper right), number of non-independence copulas (lower left) and computation time in minutes (lower right) for scenario \mathcal{W}_{10}^D : Comparison of **True model**, **Dißmann**, **k-DAG** for $k = 2, 3, 4$, **Lasso-ST- λ** for $\lambda = 0.2^4, 0.25^4$, **Lasso-AT- μ** for $\mu = 0.1, 0.2$ and **Cluster- d_T -5** for **Threshold Dimension (Thr. Dim.)** $d_T = 25, 50, 75$.

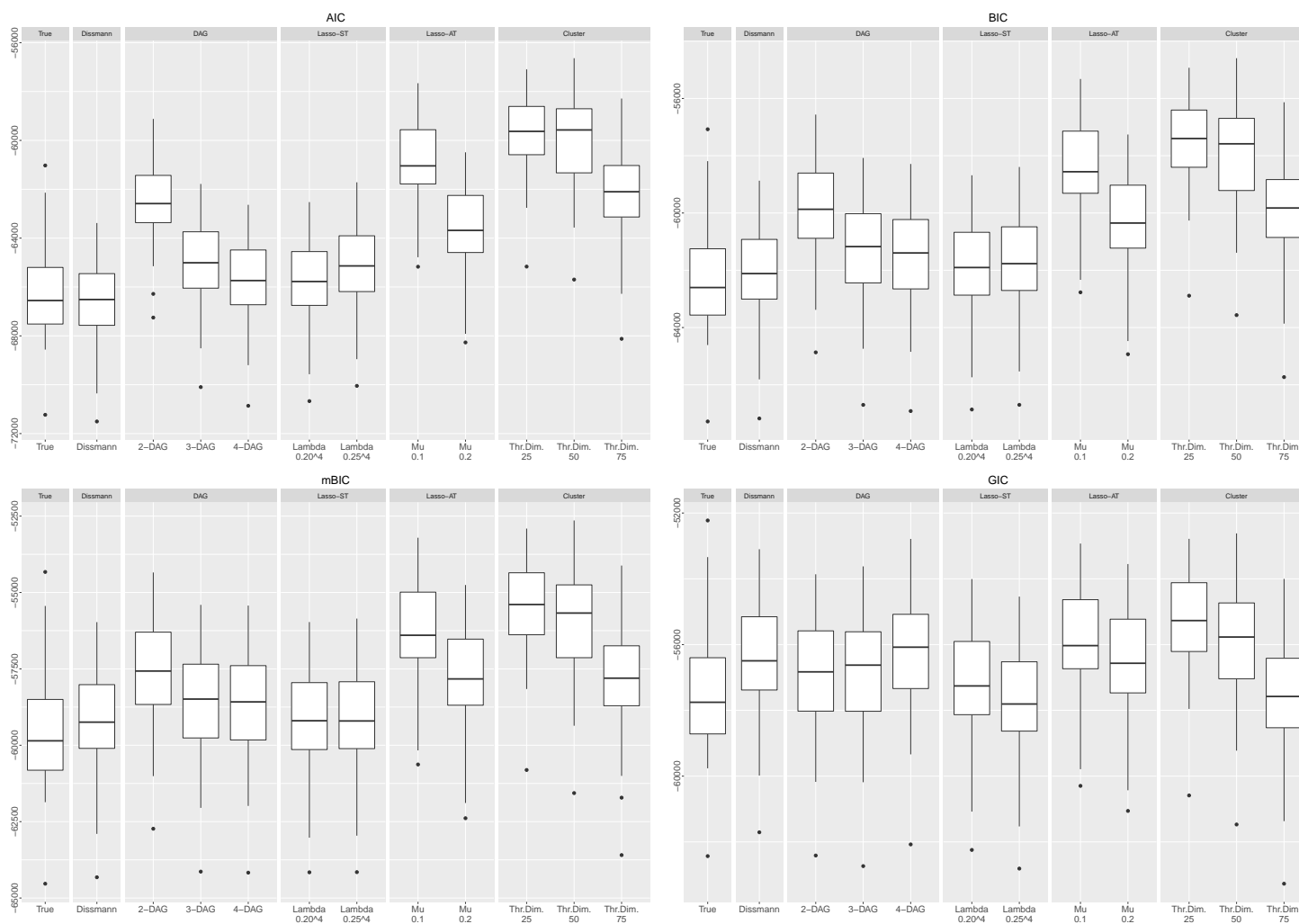


Figure 7.15: AIC (upper left), BIC (upper right), mBIC (lower left) and GIC (lower right) for scenario \mathcal{W}_{10}^D : Comparison of **True model**, **Dißmann**, **k-DAG** for $k = 2, 3, 4$, **Lasso-ST- λ** for $\lambda = 0.2^4, 0.25^4$, **Lasso-AT- μ** for $\mu = 0.1, 0.2$ and **Cluster- d_T -5** for **Threshold Dimension (Thr. Dim.)** $d_T = 25, 50, 75$.

7.3 Runtime Comparison

A crucially important aspect when comparing models is not only their goodness of fit, but also their complexity. This is not only related to the number of parameters describing the model, but also how complex it is to fit the model in terms of time and memory consumption. As we have seen, models generated with Dißmann’s algorithm in $d = 85$ dimensions have quite a large number of parameters compared to our competitors proposed in this thesis. Additionally, also the computation time is multiple times longer. We want to study the latter effect in a more dynamical environment, i. e. considering various dimensions. For this, we obtained data from $d = 1757$ stock listed companies from the entire globe in the financial services industry and preprocessed them according to Section 7.1.1. The data contains $n = 470$ trading days, i. e. about two years of data. Thus, we have a data matrix $U \in [0, 1]^{n \times d}$ and split it into 18 nested subsets

$$U_\ell \in [0, 1]^{n \times d_\ell} \text{ with } d_\ell \in \{50, 150, 250, \dots, 1750\},$$

by only considering the first d_ℓ columns. Our goal is to compare all the approaches proposed in the Chapters 4, 5 and 6 as well as Dißmann’s method considering the computation times with respect to the dimension. Based on what we have seen in Section 7.2, we expect most of our approaches to be much faster than the benchmark since they use significantly less parameters, especially in high dimensions. Hence, we compare the models given in Table 7.5.

Model name	Notation	Parameters	Description
Dißmann	$\mathcal{W}^{Diss}(U_\ell)$	-	R-vines fitted using Dißmann’s method, see Section 2.2
k-DAG	$\mathcal{W}_k^{DAG}(U_\ell)$	$k = 1, 2, 3, 4$	R-vine representations of k -DAGs, see Chapter 4
Lasso-ST-λ	$\mathcal{W}_\lambda^{Lasso-ST}(U_\ell)$	$\lambda = 0.2^4, 0.25^4$	R-vine representations of SEMs calculated with the Lasso, see Chapter 5 using the single threshold approach
Lasso-AT-μ	$\mathcal{W}_\mu^{Lasso-AT}(U_\ell)$	$\mu = 0.1, 0.2$	R-vine representations of SEMs calculated with the Lasso, see Chapter 5 using the adaptive threshold approach
Cluster-j-k_F	$\mathcal{W}_{j-k_F}^{Cluster}(U_\ell)$	$j = 1, 2, 3, 4$ $k_F = 0, 1$	R-vines based on clusterings, see Chapter 6 using $J = 30$ undirected graphical models generated with the graphical Lasso and considering component sizes $\delta_1^\ell, \dots, \delta_4^\ell$ according to Table 7.7 and fill-levels $k_F = 0, 1$

Table 7.5: Overview of the compared models in runtime comparison.

For the **Cluster** models, this means that for each dimension d_ℓ , we obtain four **Cluster**

models with fill level $k_F = 0$ and four **Cluster** models with fill level $k_F = 1$. We denote them as stated in Table 7.6. The corresponding maximum component sizes $\delta_1^\ell, \dots, \delta_4^\ell$ in

j	$k_F = 0$	$k_F = 1$
1	Cluster-1-$kF0$	Cluster-1-$kF1$
2	Cluster-2-$kF0$	Cluster-2-$kF1$
3	Cluster-3-$kF0$	Cluster-3-$kF1$
4	Cluster-4-$kF0$	Cluster-4-$kF1$

Table 7.6: Runtime comparison: Overview of **Cluster** models.

the underlying undirected graphical models are given in Table 7.7.

Data matrix	Dimension d_ℓ	δ_1^ℓ	δ_2^ℓ	δ_3^ℓ	δ_4^ℓ
U_1	50	3	4	10	18
U_2	150	9	20	46	57
U_3	250	18	36	61	84
U_4	350	26	52	90	121
U_5	450	24	64	107	152
U_6	550	17	67	127	175
U_7	650	20	84	154	210
U_8	750	20	105	187	248
U_9	850	21	75	197	278
U_{10}	950	22	82	220	307
U_{11}	1050	12	88	224	435
U_{12}	1150	12	93	238	458
U_{13}	1250	12	93	238	470
U_{14}	1350	12	92	239	483
U_{15}	1450	3	81	224	465
U_{16}	1550	3	81	228	477
U_{17}	1650	3	81	228	490
U_{18}	1750	3	81	228	439

Table 7.7: Maximum component sizes $\delta_1^\ell, \dots, \delta_4^\ell$ as defined in (6.2) of undirected graphs $\mathcal{G}_1^\ell, \dots, \mathcal{G}_4^\ell$ for data matrices U_ℓ with dimension d_ℓ , $\ell = 1, \dots, 18$.

To prevent that we only compare computation times and possibly prefer models with respect to time which are however significantly worse in terms of goodness of fit, we also consider log-likelihood, BIC, mBIC and GIC as well. The results are summarized in the Figures 7.16 - 7.20.

First of all, note that missing nodes in the Figures 7.16 - 7.20 are caused by time outs on the computer cluster or caused by memory problems. We note that **Dißmann's** algorithm was not able to fit models with $d = 450, \dots, 1750$ dimensions because we received a time out. For computation time, see Figure 7.16, the slope of **Dißmann's** curve relating dimension and computation time is the steepest.

We see a similar slope for the **k -DAGs** for $k = 1, 2, 3, 4$, since overall, these algorithms

work in a similar way using maximum spanning trees. However, there have to be fewer calculations of edge weights for the corresponding trees. Additionally, many pair copulas are set to the independence copula upfront because of the d-separation in the corresponding **DAG** models. However, going to thousands of dimensions, this algorithm is not feasible as well. The **Lasso** approach has a much less steep slope, however different between using a **Lasso-ST** and the **Lasso-AT**. Recalling the methods, the **Lasso-ST** method sets all pair copulas which are associated with λ values lower than a specified threshold λ_T to the independence copula. The **Lasso-AT** method sets the pair copulas associated to the $(1 - \mu)100\%$ lowest percent of λ values to the independence copula. Hence, $\mu \times d(d - 1) / 2$ non-independence copulas will be kept for further estimation, which is dependent on the dimension, and hence, grows linearly with d . As we see in the mBIC and GIC, Figures 7.19 and 7.20, this is not a feasible strategy for very high dimensions $d > 900$. However, the **Lasso-ST** approach performs particularly well.

For the **Cluster** method, we see that we attain very slowly increasing computation time, slightly more for $k_F = 1$ than $k_F = 0$. Yet, we also see that in terms of GIC, it is still beneficial to set fill level $k_F = 1$. Also, the models are parsimonious because only non-independence copulas are estimated where actual dependence by the graphical Lasso is detected. For models in dimensions exceeding $d > 1500$ and $k_F = 1$, i. e. fitting pair copulas outside connected components, we perform a different pair copula estimation. While we use maximum likelihood estimation for dimensions $d \leq 1500$, we only use estimation of pair copulas using inversion of Kendall's τ , see Section 2.1.4 because of computational reasons. However, this is not negatively influencing the goodness of fit, and one can expect to go to even higher dimensions using the **Cluster** method.

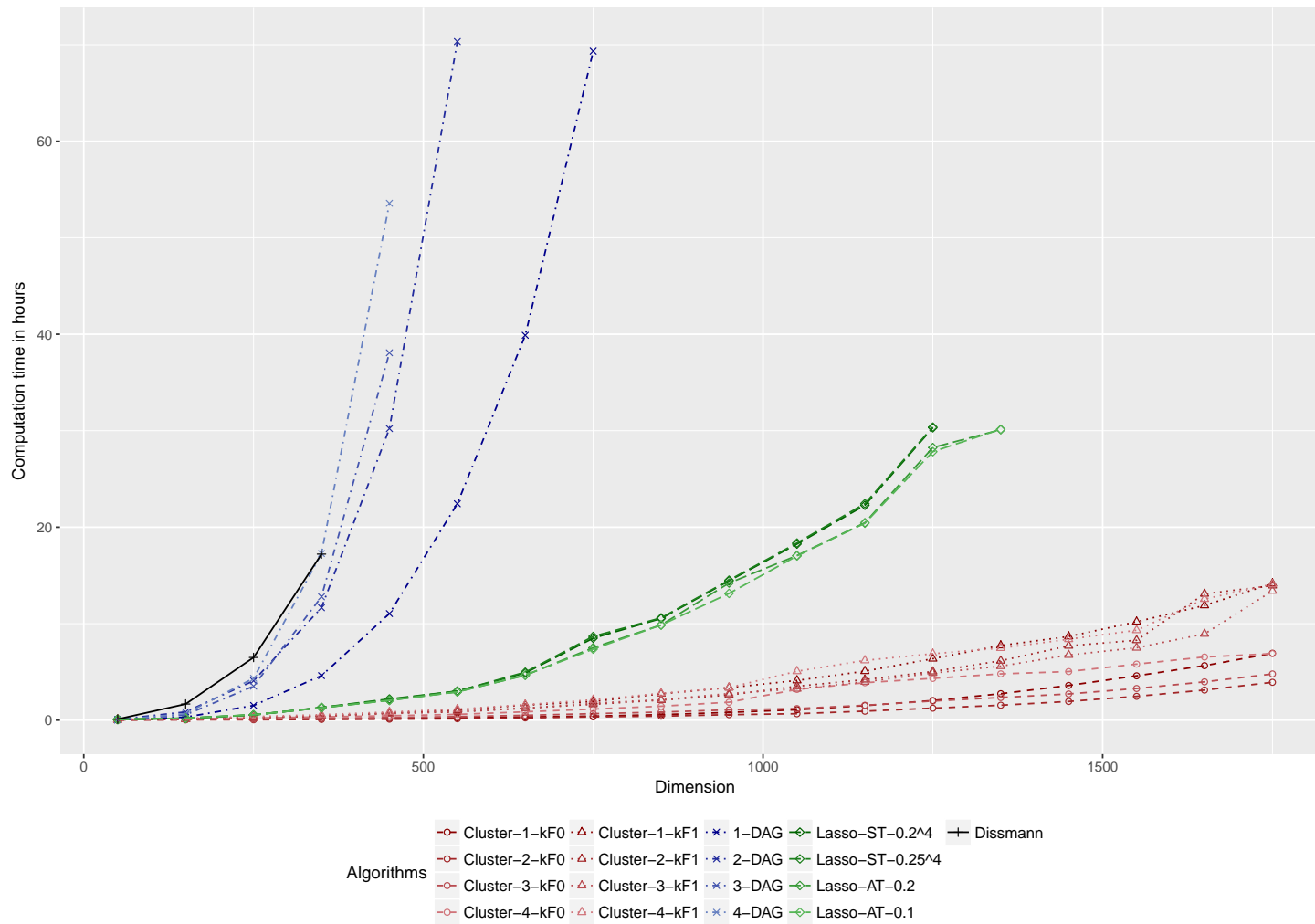


Figure 7.16: Runtime analysis: Comparison of computation time along dimensions $50, \dots, 1750$ for **Dißmann** (solid black line), **k-DAG** for $k = 1, 2, 3, 4$ (point-dashed blue lines), **Lasso-ST- λ** for $\lambda = 0.2^4, 0.25^4$ (dark dashed green lines), **Lasso-AT- μ** for $\mu = 0.1, 0.2$ (bright dashed green lines) and **Cluster- j - k_F** for $j = 1, \dots, 4$ and $k_F = 0$ (dashed red line with circles) and $k_F = 1$ (dashed red line with triangles).

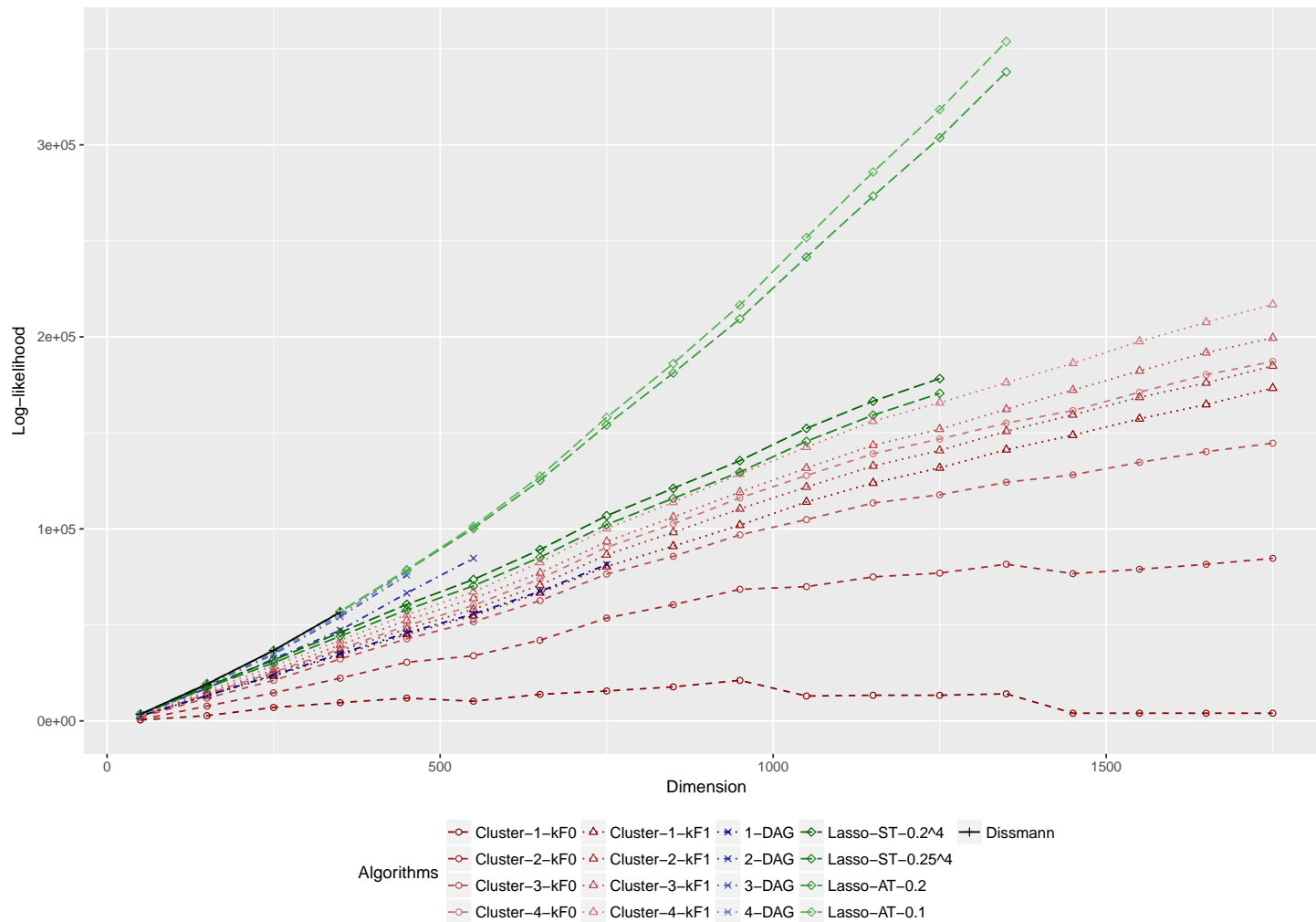


Figure 7.17: Runtime analysis: Comparison of log-likelihood along dimensions $50, \dots, 1750$ for **Dißmann** (solid black line), **k-DAG** for $k = 1, 2, 3, 4$ (point-dashed blue lines), **Lasso-ST- λ** for $\lambda = 0.2^4, 0.25^4$ (dark dashed green lines), **Lasso-AT- μ** for $\mu = 0.1, 0.2$ (bright dashed green lines) and **Cluster- j - k_F** for $j = 1, \dots, 4$ and $k_F = 0$ (dashed red line with circles) and $k_F = 1$ (dashed red line with triangles).

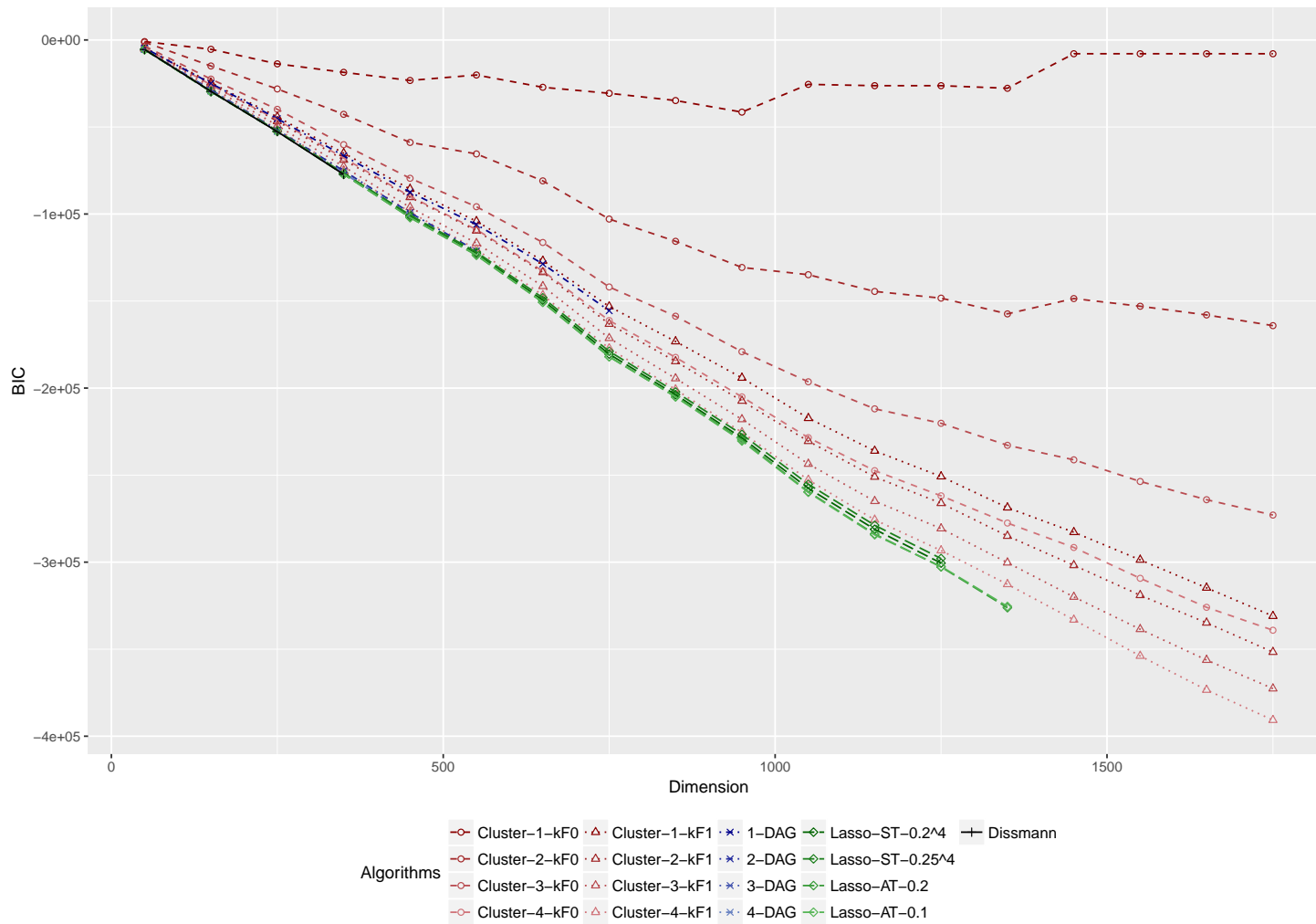


Figure 7.18: Runtime analysis: Comparison of BIC along dimensions 50, . . . , 1750 for **Dißmann** (solid black line), **k-DAG** for $k = 1, 2, 3, 4$ (point-dashed blue lines), **Lasso-ST- λ** for $\lambda = 0.2^4, 0.25^4$ (dark dashed green lines), **Lasso-AT- μ** for $\mu = 0.1, 0.2$ (bright dashed green lines) and **Cluster- j - k_F** for $j = 1, \dots, 4$ and $k_F = 0$ (dashed red line with circles) and $k_F = 1$ (dashed red line with triangles).

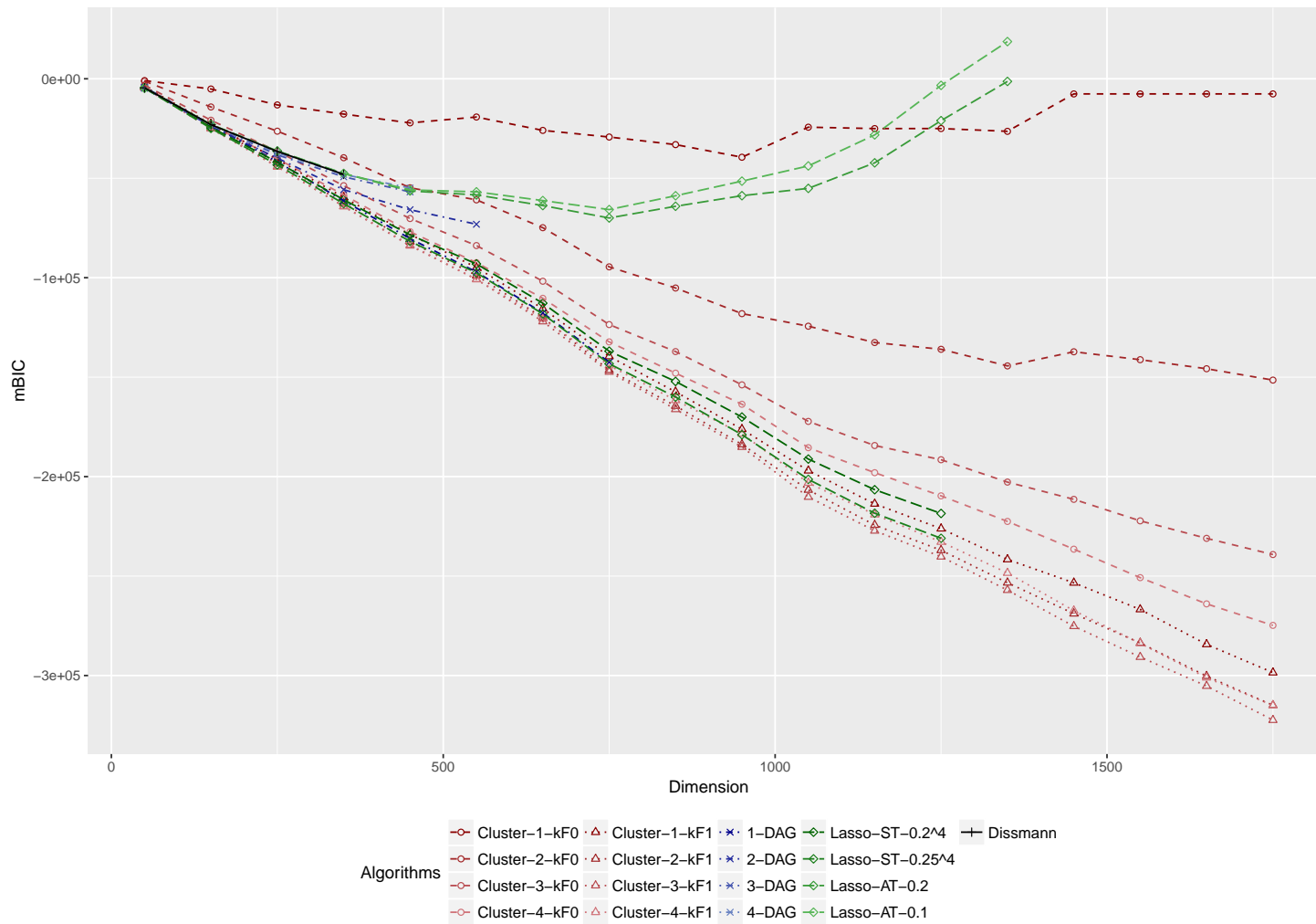


Figure 7.19: Runtime analysis: Comparison of mBIC along dimensions $50, \dots, 1750$ for **Dißmann** (solid black line), **k-DAG** for $k = 1, 2, 3, 4$ (point-dashed blue lines), **Lasso-ST- λ** for $\lambda = 0.2^4, 0.25^4$ (dark dashed green lines), **Lasso-AT- μ** for $\mu = 0.1, 0.2$ (bright dashed green lines) and **Cluster- j - k_F** for $j = 1, \dots, 4$ and $k_F = 0$ (dashed red line with circles) and $k_F = 1$ (dashed red line with triangles).

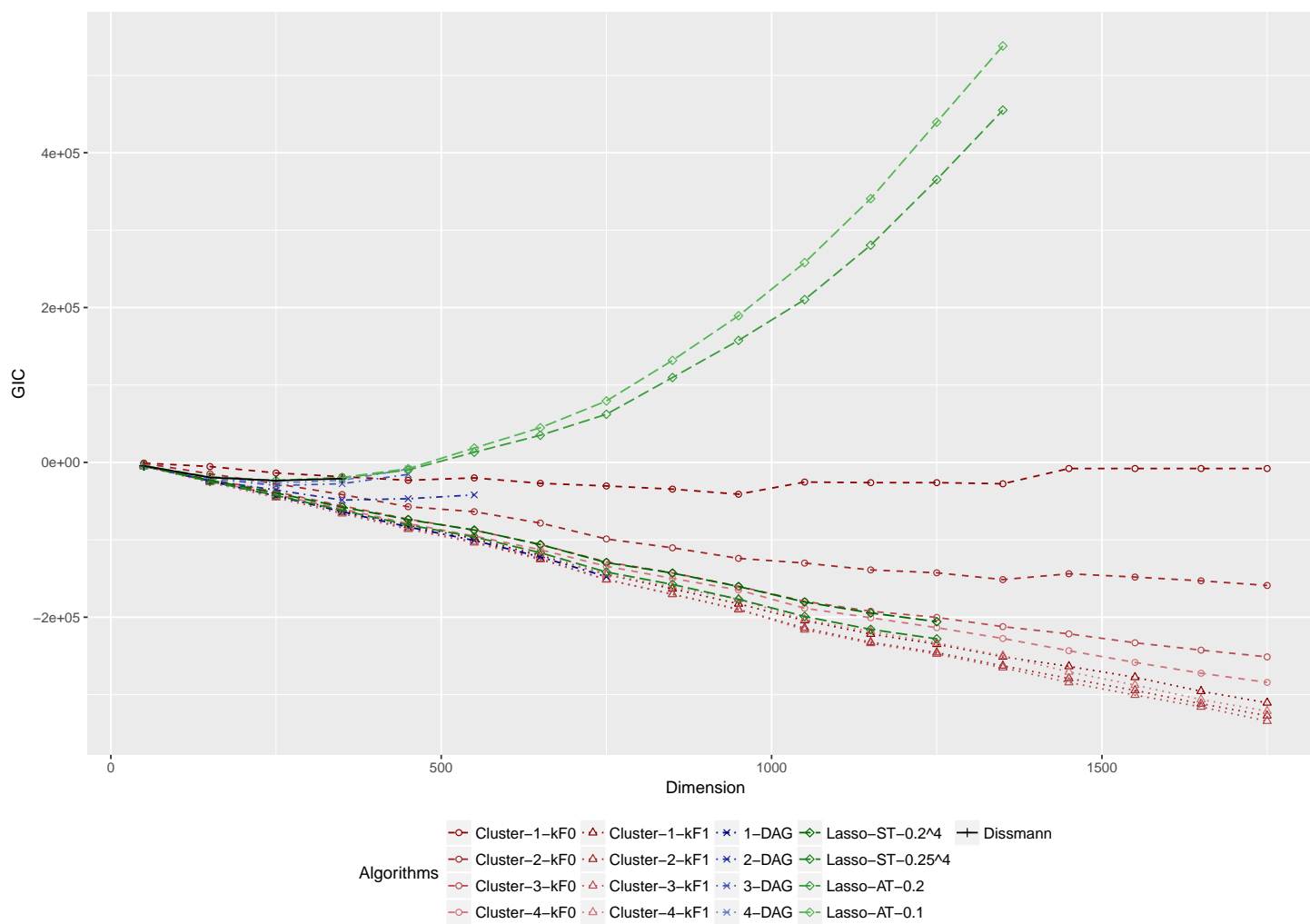


Figure 7.20: Runtime analysis: Comparison of GIC along dimensions 50, \dots , 1750 for **Dißmann** (solid black line), **k-DAG** for $k = 1, 2, 3, 4$ (point-dashed blue lines), **Lasso-ST- λ** for $\lambda = 0.2^4, 0.25^4$ (dark dashed green lines), **Lasso-AT- μ** for $\mu = 0.1, 0.2$ (bright dashed green lines) and **Cluster- j - k_F** for $j = 1, \dots, 4$ and $k_F = 0$ (dashed red line with circles) and $k_F = 1$ (dashed red line with triangles).

7.4 Value at Risk (VaR) Forecasting and Backtesting in 400 Dimensions

We conclude the thesis with an application in the financial context, returning to the motivating example in the Introduction, Chapter 1. Again, this is mostly because of the wide availability of financial data and because of the particular knowledge to estimate marginal models in this case. We consider three different scenarios, based on the data previously presented and analyzed in Section 6.3. More precisely, we take three subsets, each of dimension 400 from the original data set in dimension $d = 2131$. The dimension of the subsets is chosen such that the method of Dißmann et al. (2013) is still able to estimate models, as we want to compare the outcomes. These subsets are given by

- a random draw of 400 stocks out of all $d = 2131$ stocks,
- the sectors *Consumer Staples (ID 3)* and *Manufacturing, Industrials and Defense (ID 6)* combined, see Table 6.3, yielding exactly 400 stocks,
- a random draw of 400 stocks out of the 524 stocks in the *Financial Services and Real Estate (ID 4)* sector.

The motivation is that we have three different portfolio compositions. The first is just a random sample out of all industry sectors. The second is the union of two sectors where we would expect strong intra-sectoral dependency for both the two sectors but not much inter-sectoral dependence. The latter is due to the fact that consumer goods and industrial goods are suspected to follow rather different economy cycles and market dynamics. Finally, the third portfolio only considers *Financial Services* companies since these are expected to be strongly dependent, as seen for example in the contagion effects in the aftermath of the 2007 financial crisis. Note that since most of the stocks are based in the US, we consider this market as the leader and do not adapt for maybe different time zones between geographies.

We will now firstly describe how we set up the different models and then consider the subsequent *VaR Forecasting* and *VaR Backtesting*, where we compare the models.

7.4.1 Data Preparation and Modelling

First, we will not directly model stock prices S^j for $j = 1, \dots, d$, but instead model log-returns. Denote these by

$$R_t^j = \log \left(\frac{S_t^j}{S_{t-1}^j} \right), \text{ for } j = 1, \dots, d \text{ and } t = 2, \dots, T.$$

These time series of log returns need then to be filtered for trend and serial dependence using ARMA(p, q) models with $(p, q) \in \{0, 1\} \times \{0, 1\}$, i. e. four different specifications and allow for residuals distributed according to three different distributions, normal, Student's-t or skewed Student's-t, denoting the distribution by G_ξ , see Section 3.1.1. We estimate

all these $3 \times 4 = 12$ different models and proceed for each $j = 1, \dots, d$ by choosing the one with the highest log-likelihood. More precisely, we find coefficients $\mu_j, \varphi_j, \theta_j$ for the corresponding ARMA(1,1) model equations

$$R_t^j = \mu_j + \varphi_j (R_{t-1}^j - \mu_j) + \theta_j \epsilon_{t-1}^j + \epsilon_t^j, \text{ for } j = 1, \dots, d \text{ and } t = 2, \dots, T.$$

For an ARMA(1,0) process, i. e. an auto-regressive AR(1) process, we have $\theta_j = 0$. For an ARMA(0,1) process, i. e. a moving average MA(1) process, we have $\phi_j = 0$. For the error term, we assume a GARCH(1,1) model. Based on Definition 3.4, we can decompose the error in a so called *innovations* distribution and a scaling factor. More precisely, we have

$$\begin{aligned} \epsilon_t^j &= \sigma_t^j Z_t^j && \text{for } j = 1, \dots, d \text{ and } t = 2, \dots, T, \\ Z_t^j &\sim G_{\xi_j} && \text{for } j = 1, \dots, d \text{ and } t = 2, \dots, T, \\ (\sigma_t^j)^2 &= \omega_j + \alpha_j (\epsilon_{t-1}^j)^2 + \beta_j (\sigma_{t-1}^j)^2 && \text{for } j = 1, \dots, d \text{ and } t = 2, \dots, T, \end{aligned}$$

with G_{ξ_j} the distribution function of the normal, Student's-t or skewed Student's-t distribution with corresponding parameter vector ξ_j for $j = 1, \dots, d$. We denote the corresponding maximum likelihood estimators we proceed with by

$$\hat{\mu}_j, \hat{\varphi}_j, \hat{\theta}_j, \hat{\omega}_j, \hat{\alpha}_j, \hat{\beta}_j, \hat{\xi}_j \text{ for } j = 1, \dots, d \text{ and } t = 2, \dots, T.$$

With the estimated parameters, we can calculate *raw residuals* by

$$\hat{\epsilon}_t^j = R_t^j - \hat{R}_t^j = R_t^j - \left(\hat{\mu}_j + \hat{\varphi}_j (R_{t-1}^j - \hat{\mu}_j) + \hat{\theta}_j \epsilon_{t-1}^j \right) \text{ for } j = 1, \dots, d \text{ and } t = 2, \dots, T.$$

Since we have

$$\text{Var}(\epsilon_t^j) = (\sigma_t^j)^2,$$

we need to *standardize* the errors to obtain *standardized residuals*

$$\hat{z}_t^j = \frac{\hat{\epsilon}_t^j}{\hat{\sigma}_t^j},$$

where $\hat{\sigma}_t^j$ is the estimated standard deviation. These standardized residuals now follow the error distribution $G_{\hat{\xi}_j}$ and can be transformed to the *copula scale* using the probability integral transform (PIT) and the estimated parameters $\hat{\xi}_j$. More precisely, we have

$$u_j^t = G_{\hat{\xi}_j}(\hat{z}_t^j) \text{ for } j = 1, \dots, d \text{ and } t = 2, \dots, T. \quad (7.2)$$

Based on this data, we obtain a sample $U \in [0, 1]^{(T-1) \times d}$. We apply the algorithm of Dißmann et al. (2013) as well as our approaches described in the Chapters 4, 5 and 6, to obtain vine copula models. To recall, these will be the same as in Section 7.3, see Table 7.8.

Model name	Notation	Parameters	Description
Dißmann	$\mathcal{W}^{Diss}(U)$	-	R-vines fitted using Dißmann's method, see Section 2.2
k-DAG	$\mathcal{W}_k^{DAG}(U)$	$k = 1, 2, 3, 4$	R-vine representations of k -DAGs, see Chapter 4
Lasso-ST-λ	$\mathcal{W}_\lambda^{Lasso-ST}(U)$	$\lambda = 0.2^4, 0.25^4$	R-vine representations of SEMs calculated with the Lasso, see Chapter 5 using the single threshold approach
Lasso-AT-μ	$\mathcal{W}_\mu^{Lasso-AT}(U)$	$\mu = 0.1, 0.2$	R-vine representations of SEMs calculated with the Lasso, see Chapter 5 using the adaptive threshold approach
Cluster-d_T-k_F	$\mathcal{W}_{d_T-k_F}^{Cluster}(U)$	$d_T = 50, 100, 150, 200$	R-vines based on clusterings, see Chapter 6 using threshold dimensions $d_T = 50, 100, 150, 200$ and fill-level $k_F = \lceil \log(400) \rceil = 6$.

Table 7.8: Overview of compared models in VaR application.

In total, we have 13 models we fit to the data set U . Thus, we can now simulate from these models to obtain 13 samples of copula data, see Table 7.9. The samples of

Model name	Notation	Sample name
Dißmann	\mathcal{W}^{Diss}	U_{Diss}
1-DAG	\mathcal{W}_1^{DAG}	U_{1-DAG}
2-DAG	\mathcal{W}_2^{DAG}	U_{2-DAG}
3-DAG	\mathcal{W}_3^{DAG}	U_{3-DAG}
4-DAG	\mathcal{W}_4^{DAG}	U_{4-DAG}
Lasso-ST-0.2⁴	$\mathcal{W}_{0.25^4}^{Lasso-ST}$	$U_{Lasso-ST-0.2^4}$
Lasso-ST-0.25⁴	$\mathcal{W}_{0.25^4}^{Lasso-ST}$	$U_{Lasso-ST-0.25^4}$
Lasso-AT-0.1	$\mathcal{W}_{0.1}^{Lasso-AT}$	$U_{Lasso-AT-0.1}$
Lasso-AT-0.2	$\mathcal{W}_{0.2}^{Lasso-AT}$	$U_{Lasso-AT-0.2}$
Cluster-50-6	$\mathcal{W}_{50-6}^{Cluster}$	$U_{Cluster-50-6}$
Cluster-100-6	$\mathcal{W}_{100-6}^{Cluster}$	$U_{Cluster-100-6}$
Cluster-150-6	$\mathcal{W}_{150-6}^{Cluster}$	$U_{Cluster-150-6}$
Cluster-200-6	$\mathcal{W}_{200-6}^{Cluster}$	$U_{Cluster-200-6}$

Table 7.9: Overview of models and samples in Value at Risk (VaR) applications.

copula data can then be transformed back to the original scale. How we deal with this is described in the next section.

7.4.2 Value at Risk (VaR) One Day Ahead Forecasting

For illustration, denote a sample by $\mathbf{u}_{T+1} \in [0, 1]^d$. For notational convenience, and since the further processing is the same for all models, we do not annotate the model to the sample. Given a simulated observation u_{T+1}^j for $j = 1, \dots, d$, we perform the following operations to transform the *copula sample* to a *one day ahead portfolio value forecast*. First, we invert (7.2) by calculating

$$z_{T+1}^j = G_{\xi_j}^{-1}(u_{T+1}^j) \text{ for } j = 1, \dots, d. \quad (7.3)$$

Next, we update the (conditional) variance using the GARCH(1,1) model equation by

$$(\sigma_{T+1}^j)^2 = \hat{\omega}_j + \hat{\alpha}_j (\epsilon_T^j)^2 + \hat{\beta}_j (\sigma_T^j)^2. \quad (7.4)$$

Using the updated variance from (7.4) and the simulated data z_{T+1}^j from (7.3), we calculate the return forecast using the ARMA(1,1) equation

$$R_{T+1}^j = \hat{\mu}_j + \hat{\varphi}_j (R_T^j - \hat{\mu}_j) + \hat{\theta}_j \epsilon_T^j + z_{T+1}^j \sigma_{T+1}^j. \quad (7.5)$$

The *portfolio return* is now calculated by transforming the log-returns to

$$r_{T+1} = \frac{1}{d} \sum_{j=1}^d (\exp(R_{T+1}^j) - 1),$$

when assuming an equally split portfolio. We repeat this calculation for m iterations to obtain m one-day ahead portfolio returns $r_{T+1}^1, \dots, r_{T+1}^m$. If we are now interested in the *Value at Risk (VaR)* at confidence level α , we order the simulated returns increasingly

$$r_{T+1}^{[1]} \leq r_{T+1}^{[2]} \leq \dots \leq r_{T+1}^{[m-1]} \leq r_{T+1}^{[m]}$$

and choose the empirical $(1 - \alpha)$ -quantile $i^* = 1, \dots, m$ such that

$$r_{T+1}^{[i^*]} \text{ with } i^* \leq (1 - \alpha)m \leq i^* + 1.$$

We apply this framework now in the three scenarios described previously. We have the following parameter settings.

$$T = 500, \quad m = 100,000, \quad \mathbf{q}_\alpha = (0.005, 0.01)$$

Thus, we use the first 500 data points to predict the $T+1 = 501$ -st observation, i. e. obtain a one-day ahead forecast. We do this using $m = 100,000$ simulations and we evaluate the corresponding $\mathbf{q}_\alpha = (0.005, 0.01)$ quantiles, i. e. the 99% and the 99.5% VaR.

Scenario 1: Random Sample

First, we consider Scenario 1, which is composed by drawing a random sample of 400 stocks. More precisely, we examine how the models behave in terms of goodness of fit as described by log-likelihood, based on the first $T = 500$ observations. Note that we only compare the log-likelihood of the dependence models, since the marginal models are the always identical. Thus, we have an idea which VaR forecasts are associated to models obtaining superior goodness of fit. Next, we check which models predict the same VaR values but have less parameters.

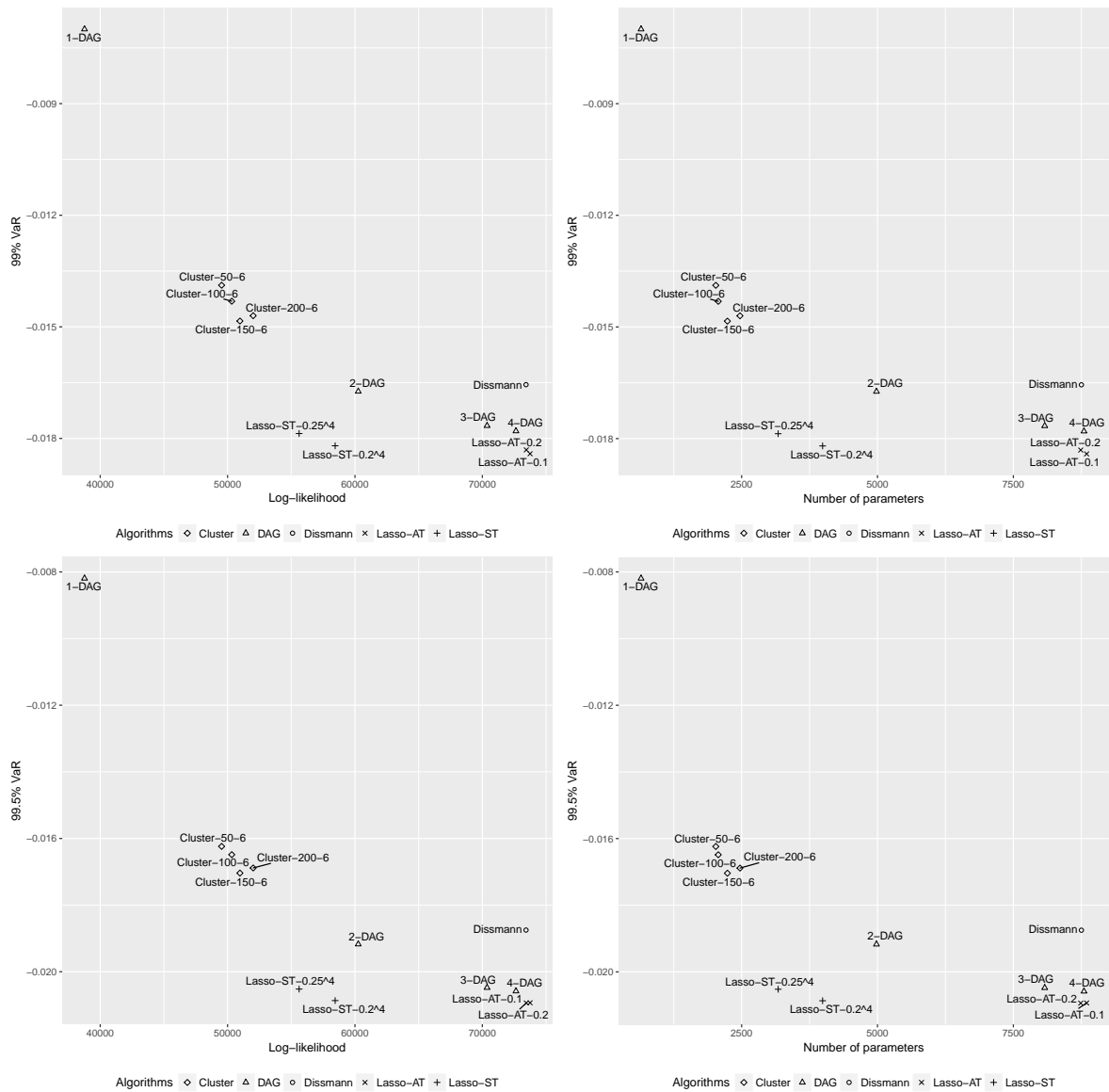


Figure 7.21: One day ahead forecasting results for $T = 501$: 99% VaR (top) and 99.5% VaR (bottom) in Scenario 1 (Random Sample) in terms of log-likelihood vs. Value at Risk (left) as well as number of parameters vs. Value at Risk (right): Comparison of **Dissmanno** (\circ), **k-DAG** for $k = 1, 2, 3, 4$ (Δ), **Lasso-ST- λ** for $\lambda = 0.2^4, 0.25^4$ ($+$), **Lasso-AT- μ** for $\mu = 0.1, 0.2$ (\times) and **Cluster- d_T -6** for $d_T = 50, 100, 150, 200$ (\diamond).

Thus, we first compare the models while considering log-likelihood with VaR and start off with 99% VaR, see Figure 7.21, upper left panel. Note that we do not explicitly consider penalized goodness of fit measures as BIC or GIC since we consider the number of parameters isolated in a second step. We see that **Dißmann's** model as well as the **3-DAG**, **4-DAG** and **Lasso-AT- μ** models have highest log-likelihood. Additionally, we see that models with high explanatory power, i. e. high log-likelihood prefer VaR values around -0.018 . Next, we compare the models which predict this VaR in terms of number of parameters, see Figure 7.21, upper right panel. There, we see that even though the log-likelihood is not as high, the **Lasso-ST- λ** models achieve equally suitable quite conservative Value at Risk values. We monitor the same behaviour for the higher VaR at level 99.5%, see Figure 7.21, lower panel. Thus, we can reduce the number of parameters by more than 60% using the **Lasso-ST- λ** models while obtaining a similar and hence, suitable VaR forecast.

We move to Scenario 2, where we consider stocks of the union of two industry sectors.

Scenario 2: Sectors 3 & 6

Next, we consider the sectors *Consumer Staples (ID 3)* and *Manufacturing, Industrials and Defense (ID 6)*, together 400 stocks. We see a quite similar behaviour compared to the previous scenario, see Figure 7.22. However, we see that the 99% VaR is higher with about -0.017 compared to -0.018 for the models with highest log-likelihood, which might indicate higher diversification effects. This is reasonable as losses in e. g. consumer staples are offset by gains in manufacturing or vice versa. Furthermore, we also monitor again that the **Lasso-ST- λ** models are sufficient in terms of significantly less parameters than the larger models, but having the same VaR bounds. We also see this behaviour for the 99.5% VaR. The **Cluster- d_T -6** models for $d_T = 50, 100, 150, 200$ are too risk affine, but have also the least absolute number of parameters. Where a **2-DAG** model was still acceptable in Scenario 1, this is not the case in Scenario 2 as the risk is severely underestimated, which also applies to the **1-DAG** model. Note that the heavy spike in the middle of the time around June 2016 can be related to Great Britain deciding to leave the European Union, as the referendum took place at that time.

7.4 Value at Risk (VaR) Forecasting and Backtesting in 400 Dimensions

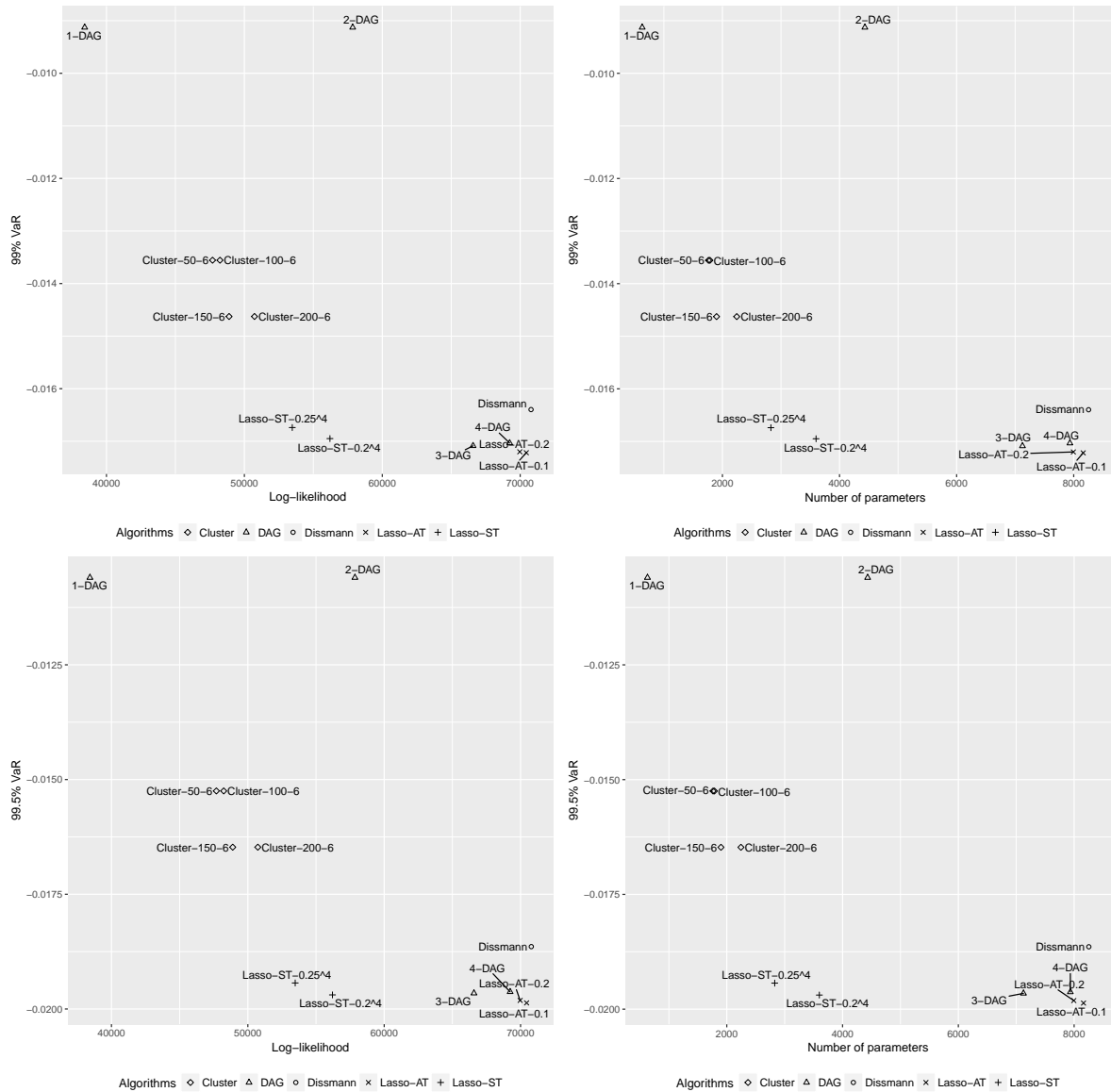


Figure 7.22: One day ahead forecasting results for $T = 501$: 99% VaR (top) and 99.5% VaR (bottom) in Scenario 2 (Sectors 3 & 6) in terms of log-likelihood vs. Value at Risk (left) as well as number of parameters vs. Value at Risk (right): Comparison of **Dißmann** (\circ), **k-DAG** for $k = 1, 2, 3, 4$ (Δ), **Lasso-ST- λ** for $\lambda = 0.2^4, 0.25^4$ ($+$), **Lasso-AT- μ** for $\mu = 0.1, 0.2$ (\times) and **Cluster- d_T -6** for $d_T = 50, 100, 150, 200$ (\diamond).

Scenario 3: Financial Services Sector

Finally, we consider 400 stocks from the *Financial Services* sector and expect strong interactions among these stocks. With respect to the properties of the models, we draw the same conclusions with respect to parsimony as in the previous two scenarios, i. e. the same algorithms provide the most parsimonious models as for the second scenario. However, we also see that 99% VaR attained by the best fitting models is higher compared to the previous two other portfolios with -0.016 . This can be explained by the general

upward movement of the Financial Services sectors fostered by very low interest rates set by the central banks. Additionally, we also see that **1-DAG** model is not sufficient. This approach gave us an intuition about the properties of the different scenarios and the associated models. Yet, we can also perform a more quantitatively driven analysis using *VaR backtesting*, which follows now.

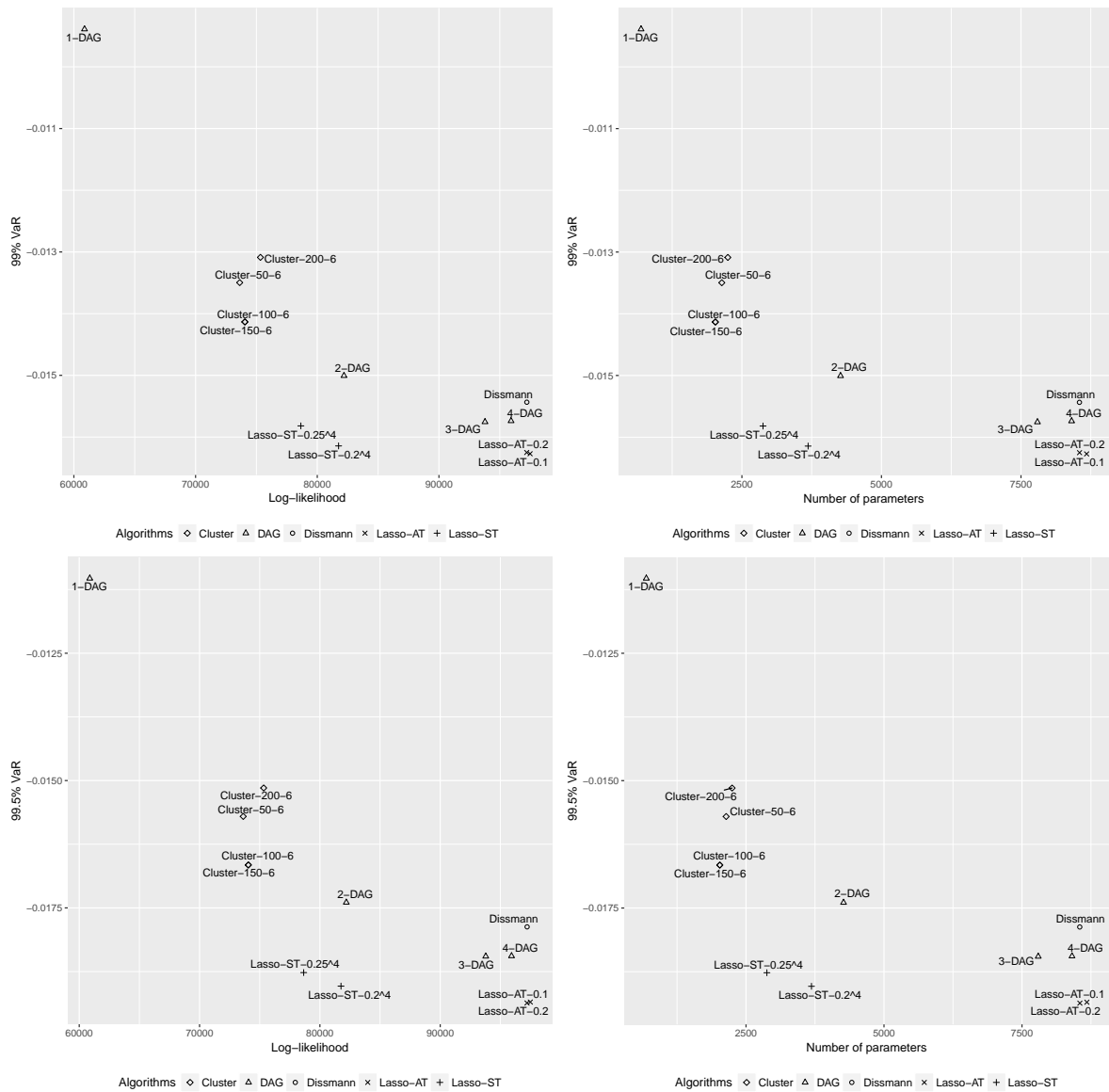


Figure 7.23: One day ahead forecasting results for $T = 501$: 99% VaR (top) and 99.5% VaR (bottom) in Scenario 3 (Financial Services) in terms of log-likelihood vs. Value at Risk (left) as well as number of parameters vs. Value at Risk (right): Comparison of **Dissmann** (\circ), **k-DAG** for $k = 1, 2, 3, 4$ (Δ), **Lasso-ST- λ** for $\lambda = 0.2^4, 0.25^4$ ($+$), **Lasso-AT- μ** for $\mu = 0.1, 0.2$ (\times) and **Cluster- d_T -6** for $d_T = 50, 100, 150, 200$ (\diamond).

7.4.3 Value at Risk (VaR) Backtesting

The basic idea of backtesting is the following. Assume we have historical data from $t = 1, \dots, T$. Then we can develop a model based on the data, but only using the first T' time points with $T' < T$, denote the corresponding model by $M_1^{T'}$. Next, we can compute *out-of-sample* predictions using the model for the time point $T' + 1$, i.e. a one day ahead VaR forecast as in Section 7.4.2. However, for this time point, we also have historical data and can now compare how our model performed against reality i.e. if the VaR forecast was exceeded or not. This procedure is called *backtesting* and is also required by regulators, see Basel Committee on Banking Supervision (2006, 2011, 2013). Normally, we would then re-estimate the model on time point $T' + 1$ using data on the time points $2, \dots, T' + 1$, denote $M_2^{T'+1}$ and compute a one day ahead forecast to predict time point $T' + 2$, compare it to the real value and so on, see Figure 7.24. However, this

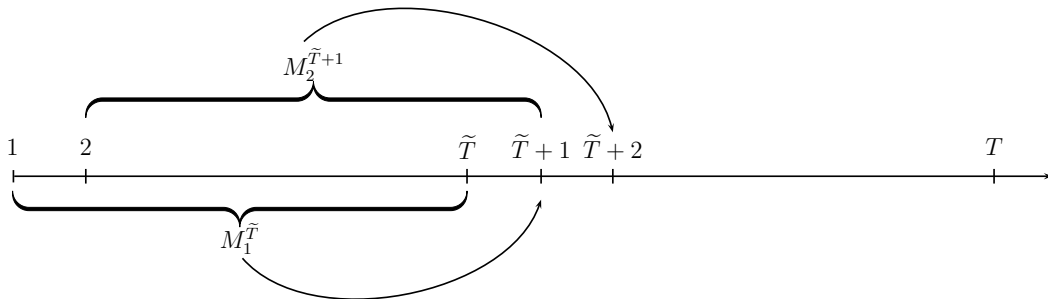


Figure 7.24: Schematic view of out of sample prediction with rolling window model estimation.

requires for each new time point to fit a new model. This is not feasible if models are as complex as in our case and take a huge effort to estimate and simulate from. Thus, we proceed as follows to obtain a dynamic forecast.

- We fit ARMA-GARCH models as described in Section 7.4.1 on the time points $t = 1, \dots, 500$.
- Based on these, we obtain copula data as described in Section 7.4.1 and fit 13 different dependence models, see Table 7.8 on the time points $t = 1, \dots, 500$, denote M_1^{500} .
- We use M_1^{500} to obtain *VaR forecasts*, using the one day ahead methodology as described in Section 7.4.2 for the 250 time points $t = 501, \dots, 750$. More precisely, we carry out the following steps.
 - Simulate $\mathbf{m} = \mathbf{1,000}$ replications from the dependence model \mathcal{W} and denote the sample by $\mathbf{u}_{501} \in [0, 1]^d$.
 - Transform this sample from the *copula-scale* to the standardized residuals' *x-scale* as in 7.3 by calculating

$$z_{501}^j = G_{\xi_j}^{-1}(u_{501}^j) \text{ for } j = 1, \dots, d.$$

- Next, we update the *conditional variance* in the GARCH(1,1) model, i. e. the variance forecast

$$(\hat{\sigma}_{501}^j)^2 = \hat{\omega}_j + \hat{\alpha}_j (\epsilon_{500}^j)^2 + \hat{\beta}_j (\sigma_{500}^j)^2. \quad (7.6)$$

- This conditional variance forecast is then used in the ARMA(1,1) model equation together with the simulated and transformed residual z_{501}^j to obtain a *return forecast*

$$\hat{R}_{501}^j = \hat{\mu}_j + \hat{\varphi}_j (R_{500}^j - \hat{\mu}_j) + \hat{\theta}_j \epsilon_{500}^j + z_{501}^j \sigma_{501}^j. \quad (7.7)$$

- Since we assume an equally split portfolio, the *portfolio return forecast* is then calculated by

$$\hat{r}_{501} = \frac{1}{d} \sum_{j=1}^d \left(\exp \left(\hat{R}_{501}^j \right) - 1 \right).$$

- Now, having $m = 1,000$ replications, we obtain a sequence of m one day ahead portfolio return forecasts for day $t = 501$:

$$\hat{r}_{501}^1, \dots, \hat{r}_{501}^m.$$

- Based on these m replications, we calculate
 - the empirical $\alpha = 0.1$ -quantile, i. e. 90% VaR,
 - the empirical $\alpha = 0.05$ -quantile: i. e. 95% VaR,
 - the return forecast

$$\hat{r}_{501} = \frac{1}{1000} \sum_{i=1}^{1000} \hat{r}_{501}^i.$$

- These three values can then be compared to the real portfolio return r_{501} .
- Now, we want to calculate the portfolio return forecast \hat{r}_{502} for $t = 502$. For this, we calculate the standardized residuals of the individual stocks

$$\epsilon_{501}^j = \frac{\hat{R}_{501}^j - R_{501}^j}{\sigma_{501}^j}, \quad j = 1, \dots, d.$$

Recall that in the backtesting set up, the true values R_{501}^j , $j = 1, \dots, d$ are available at time point $t = 501$.

- We update the conditional variance as in 7.6,

$$(\hat{\sigma}_{502}^j)^2 = \hat{\omega}_j + \hat{\alpha}_j (\epsilon_{501}^j)^2 + \hat{\beta}_j (\sigma_{501}^j)^2.$$

- Next, we simulate again $m = 1,000$ replications of $\mathbf{u}_{502} \in [0, 1]^d$ from the

dependence model \mathcal{W} and carry out the prediction as in 7.7,

$$\hat{R}_{502}^j = \hat{\mu}_j + \hat{\varphi}_j (R_{501}^j - \hat{\mu}_j) + \hat{\theta}_j \epsilon_{501}^j + z_{502}^j \sigma_{502}^j.$$

- As in the prediction for day $t = 501$, we calculate the return forecast and the corresponding quantiles and continue with the subsequent days, $t = 503, \dots, 750$.
- Thus, we obtain a sequence of predictions for the time points $t = 501, \dots, 750$.
- We fit ARMA-GARCH models as described in Section 7.4.1 on the time points $t = 251, \dots, 750$.
- Based on these, we obtain copula data as described in Section 7.4.1 and fit 13 different dependence models, see Table 7.8 on the data points $t = 251, \dots, 750$, denote M_{251}^{750} .
- We use M_{251}^{750} to obtain *VaR forecasts*, using the one day ahead methodology as described in Section 7.4.2 for the 249 time points $t = 751, \dots, 999$, as described more precisely above for the time interval $t = 501, \dots, 750$.

By this, we only refit our model once, see Figure 7.25. In total, we obtain $250 + 249 = 499$

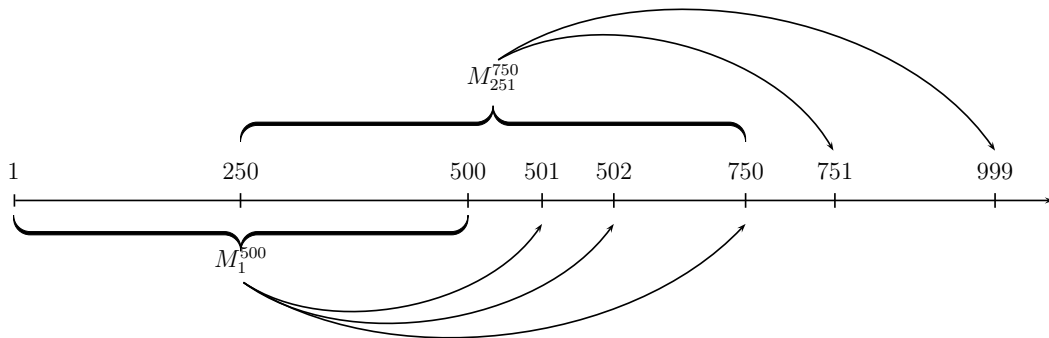


Figure 7.25: Schematic view of out of sample prediction with one refitting.

one day ahead forecasts for the next days return and corresponding Value at Risk values. As we use $m = 1,000$ replications for each day, we only calculate the corresponding 90% and 95% VaR and omit the higher 99% and 99.5% VaR which were feasible with $m = 100,000$ simulated values in the forecasting example. To assess the quality of our predictions, we can then perform a *backtest* and compare the 499 VaR forecasts with the actually attained values of the portfolio. The methodology to assess the quality of our predictions is presented next.

VaR Backtesting Methodology

We can now either compare for example graphically how often the actual returns were less than predicted by the 90% or 95% VaR. However, there is a more sound statistical procedure for this, described by Christoffersen (2012, Chapter 13). More precisely, let

$\alpha \in (0, 1)$ and assume that we have for each day $t = 1, \dots, T = 499$ a portfolio VaR forecast at level α , denote VaR_t^α . Assume additionally that we have the information, whether or not the actual portfolio return r_t on each day fell below the VaR forecast. This leads to the *hit statistics*, defined by

$$\mathcal{I}_t = \begin{cases} 1, & \text{if } r_t < VaR_t^\alpha \\ 0, & \text{if } r_t \geq VaR_t^\alpha. \end{cases}$$

Hence, we obtain a hit sequence $(\mathcal{I}_t)_{t=1, \dots, T}$. If our VaR model works correctly, this sequence should have two properties. First, it should be unpredictable, because if there were a pattern, we could use it to improve our VaR model. Second, the *exceedance*, i. e. the event $\mathcal{I}_t = 1$ should happen with probability α because of the construction of our VaR model. This means, we want to have

$$(\mathcal{I}_t)_{t=1, \dots, T} \sim \text{Bernoulli}(\alpha) \text{ i.i.d.}$$

First, we consider whether the hit sequence has mean α . To this end, let

$$n_i = \sum_{t=1}^T \mathbb{1}_{\{\mathcal{I}_t=i\}} \text{ for } i = 0, 1.$$

If we assume $(\mathcal{I}_t)_{t=1, \dots, T} \sim \text{Bernoulli}(p)$, then the so called *unconditional coverage (UC) test* checks the hypothesis

$$H_0 : p = \alpha \text{ vs. } H_1 : p \neq \alpha.$$

To derive a test for this hypothesis, we calculate the likelihood of an i.i.d. *Bernoulli* (p) sequence

$$L(p) = \prod_{t=1}^T (1-p)^{1-\mathcal{I}_t} p^{\mathcal{I}_t} = (1-p)^{T_0} p^{T_1},$$

with T_0 the number of zeros and T_1 the number of ones in the hit sequence. With the maximum likelihood estimator

$$\hat{p} = T_1/T,$$

we have

$$L(\hat{p}) = \left(1 - \frac{T_1}{T}\right)^{T_0} \left(\frac{T_1}{T}\right)^{T_1}.$$

Under the null hypothesis, we have $p = \alpha$, and thus

$$L(\alpha) = (1 - \alpha)^{T_0} \alpha^{T_1}.$$

and use a *likelihood ratio test*, considering the test statistics

$$\begin{aligned} LR_{UC} &= -2 \log \left(\frac{L(\alpha)}{L(\hat{p})} \right) \\ &= -2 \log \left((1-\alpha)^{T_0} \alpha^{T_1} / \left((1-T_1/T)^{T_0} (T_1/T)^{T_1} \right) \right) \sim \chi_1^2, \end{aligned} \quad (7.8)$$

with χ_k^2 denoting a *Chi-squared-distribution* with k degrees of freedom. The *p-value* can be calculated to be

$$\eta = 1 - F_{\chi_1^2}(LR_{UC}).$$

A *p-value* of η says that for any *significance level* $\eta' \geq \eta$, the null-hypothesis H_0 would be rejected.

Next, we consider the question whether the hit sequence $(\mathcal{I}_t)_{t=1,\dots,T}$ is *independent*. More precisely, we test

$$H_0 : (\mathcal{I}_t)_{t=1,\dots,T} \text{ are independent vs. } H_1 : (\mathcal{I}_t)_{t=1,\dots,T} \text{ are not independent .}$$

For this, we need another test statistic, measuring whether a *hit* is succeeded by other hits, i. e. whether they occur in clusters or at random. We define

$$n_{ij} = \sum_{t=1}^{T-1} \mathbb{1}_{\{\mathcal{I}_t=i, \mathcal{I}_{t+1}=j\}} \text{ for } i, j = 0, 1,$$

and

$$\begin{aligned} \hat{\pi}_{01} &= \frac{n_{01}}{n_{00} + n_{01}}, \\ \hat{\pi}_{11} &= \frac{n_{11}}{n_{10} + n_{11}}, \\ \hat{\pi} &= \frac{n_{01} + n_{11}}{T}. \end{aligned}$$

The test statistics for the independence test can be written as

$$LR_{Ind} = -2 \log \left(\frac{(1-\hat{\pi})^{n_{00}+n_{10}} \hat{\pi}^{n_{01}+n_{11}}}{(1-\hat{\pi}_{01})^{n_{00}} \hat{\pi}_{01}^{n_{01}} (1-\hat{\pi}_{11})^{n_{10}} \hat{\pi}_{11}^{n_{11}}} \right) \sim \chi_1^2, \quad (7.9)$$

see Schamberger (2015). Finally, we can also perform a joint test for both exceedance and independence, called the *conditional coverage (CC) test*. The hypotheses are

$$H_0 : (\mathcal{I}_t)_{t=1,\dots,T} \sim \text{Bernoulli}(\alpha) \text{ i.i.d. vs. } H_1 : \text{not } H_0,$$

and can be tested using (7.8) and (7.9) with

$$LR_{CC} = LR_{UC} + LR_{Ind} \sim \chi_2^2.$$

Since these test statistics all follow asymptotically a closed form distribution, we can use the theoretical quantile to reject the null-hypothesis or not. However, if we do not

have a large number of observations and especially not a large number of hits, using the asymptotic results can be misleading. Hence, we can also use Monte Carlo simulations for the p-values. More precisely, since we know that under the null hypothesis H_0 our *hit sequence* is *Bernoulli* (α) i.i.d. with length T , we can simulate a large number, for example 999 of such *Bernoulli* (α) i.i.d. sequences of length T . Then, we calculate the corresponding likelihood ratio test statistics on these sequences, denote $(\widetilde{LR}_i)_{i=1,\dots,999}$. Finally, the Monte-Carlo p-value η_{MC} of a test statistic LR can be calculated by

$$\eta_{MC} = \frac{1}{1000} \left(1 + \sum_{i=1}^{999} \mathbb{1}_{\{\widetilde{LR}_i \geq LR\}} \right).$$

We will consider both the p-values with respect to the asymptotic distribution of the test statistics and the p-values drawn with Monte Carlo simulation. More precisely, the following p-values as shown in Table 7.10 are considered.

Abbreviation	Test for	Null hypothesis	p-value type
UC	Unconditional coverage	$(\mathcal{I}_t)_{t=1,\dots,T} \sim \text{Bernoulli}(\alpha)$	asymptotic χ^2 test
UC-MC	Unconditional coverage	$(\mathcal{I}_t)_{t=1,\dots,T} \sim \text{Bernoulli}(\alpha)$	Monte Carlo approximation
Ind	Independence	$(\mathcal{I}_t)_{t=1,\dots,T}$ independent	asymptotic χ^2 test
Ind-MC	Independence	$(\mathcal{I}_t)_{t=1,\dots,T}$ independent	Monte Carlo approximation
CC	Conditional coverage	$(\mathcal{I}_t)_{t=1,\dots,T} \sim \text{Bernoulli}(\alpha)$ i.i.d.	asymptotic χ^2 test
CC-MC	Conditional coverage	$(\mathcal{I}_t)_{t=1,\dots,T} \sim \text{Bernoulli}(\alpha)$ i.i.d.	Monte Carlo approximation

Table 7.10: Summary of hypothesis tests, test purpose, null hypotheses and test type in the Value at Risk backtesting.

We consider again the three scenarios separated. We will use a $\beta = 0.05$ significance level for the hypothesis test whether we can reject the null hypotheses stated in Table 7.10. Note that this is not to be confused with the Value at Risk level α . Since we want to not rejected the null-hypothesis, a p-value larger than 0.05 is good for the corresponding model.

Scenario 1: Random Sample

First, we consider the random sample scenario and VaR level $\alpha = 0.90$. The p-values for the hypothesis tests as described in Table 7.10 are given in Table 7.11.

7.4 Value at Risk (VaR) Forecasting and Backtesting in 400 Dimensions

Model	UC	UC-MC	Ind	Ind-MC	CC	CC-MC
Dißmann	0.108	0.109	0.004	0.002	0.004	0.004
1-DAG	0.000	0.001	0.224	0.223	0.000	0.001
2-DAG	0.021	0.020	0.001	0.001	0.000	0.001
3-DAG	0.186	0.187	0.045	0.043	0.056	0.048
4-DAG	0.299	0.304	0.003	0.002	0.007	0.007
Lasso-ST-0.2⁴	0.186	0.187	0.006	0.004	0.010	0.012
Lasso-ST-0.25⁴	0.371	0.376	0.019	0.016	0.042	0.040
Lasso-AT-0.1	0.453	0.415	0.014	0.009	0.036	0.033
Lasso-AT-0.2	0.299	0.304	0.003	0.002	0.007	0.007
Cluster-50-6	0.000	0.001	0.009	0.006	0.000	0.001
Cluster-100-6	0.000	0.001	0.000	0.001	0.000	0.001
Cluster-150-6	0.000	0.001	0.001	0.001	0.000	0.001
Cluster-200-6	0.000	0.001	0.001	0.001	0.000	0.001

Table 7.11: Backtesting results for VaR Scenario 1 (Random Sample) and $\alpha = 0.90$: p-values based on the χ^2 distribution (UC, Ind, CC) and Monte Carlo approach (UC-MC, Ind-MC, CC-MC) for the *unconditional coverage*, *independence* and *conditional coverage* test. Values exceeding 0.05 are marked in bold.

Additionally, we present the raw number of VaR hits since a model can be rejected if it is either too conservative or too risk affine, see Table 7.12. For example, for a 90% VaR, the hit rate $\hat{p} \sim 0.1$ in the ideal case.

Model	Hit rate \hat{p} for 90% VaR
Dißmann	0.124
1-DAG	0.275
2-DAG	0.134
3-DAG	0.120
4-DAG	0.116
Lasso-ST-0.2⁴	0.120
Lasso-ST-0.25⁴	0.114
Lasso-AT-0.1	0.112
Lasso-AT-0.2	0.116
Cluster-50-6	0.176
Cluster-100-6	0.156
Cluster-150-6	0.154
Cluster-200-6	0.160

Table 7.12: Hit rates $\hat{p} = T_1/T$ for 90% VaR in Scenario 1 (Random Sample).

Next, we give the corresponding p-values for VaR level $\alpha = 0.95$ in Table 7.13.

Model	UC	UC-MC	Ind	Ind-MC	CC	CC-MC
Dißmann	0.004	0.005	0.123	0.186	0.005	0.005
1-DAG	0.000	0.001	0.027	0.017	0.000	0.001
2-DAG	0.012	0.010	0.078	0.081	0.009	0.006
3-DAG	0.020	0.022	0.060	0.058	0.012	0.011
4-DAG	0.020	0.022	0.060	0.058	0.012	0.011
Lasso-ST-0.2⁴	0.033	0.037	0.046	0.035	0.014	0.013
Lasso-ST-0.25⁴	0.051	0.058	0.035	0.022	0.016	0.014
Lasso-AT-0.1	0.077	0.074	0.026	0.017	0.018	0.016
Lasso-AT-0.2	0.051	0.058	0.035	0.022	0.016	0.014
Cluster-50-6	0.000	0.001	0.001	0.001	0.000	0.001
Cluster-100-6	0.000	0.001	0.069	0.076	0.000	0.001
Cluster-150-6	0.000	0.001	0.135	0.202	0.000	0.001
Cluster-200-6	0.000	0.001	0.108	0.162	0.000	0.001

Table 7.13: Backtesting results for VaR Scenario 1 (Random Sample) and $\alpha = 0.95$: p-values based on the χ^2 distribution (UC, Ind, CC) and Monte Carlo approach (UC-MC, Ind-MC, CC-MC) for the *unconditional coverage*, *independence* and *conditional coverage* test. Values exceeding 0.05 are marked in bold.

The corresponding hit rates are given in Table 7.14.

Model	Hit rate \hat{p} for 95% VaR
Dißmann	0.082
1-DAG	0.226
2-DAG	0.078
3-DAG	0.074
4-DAG	0.074
Lasso-ST-0.2⁴	0.072
Lasso-ST-0.25⁴	0.070
Lasso-AT-0.1	0.068
Lasso-AT-0.2	0.070
Cluster-50-6	0.118
Cluster-100-6	0.102
Cluster-150-6	0.100
Cluster-200-6	0.098

Table 7.14: Hit rates $\hat{p} = T_1/T$ for 95% VaR in Scenario 1 (Random Sample).

Finally, we give the number of parameters, log-likelihood and GIC values for all the models, split in the first fit interval $t = 1, \dots, 500$ and the second fit interval, $t = 251, \dots, 750$.

7.4 Value at Risk (VaR) Forecasting and Backtesting in 400 Dimensions

	$t = 1, \dots, 500$			$t = 251, \dots, 750$		
Model	No. par.	Log-lik.	GIC	No. par.	Log-lik.	GIC
Dißmann	8752	73429	-1725	9199	86634	-19885
1-DAG	649	38769	-69861	667	48918	-89913
2-DAG	4981	60257	-43044	5195	72130	-63062
3-DAG	8077	70371	-7986	8634	83532	-24102
4-DAG	8799	72638	722	9337	85588	-15238
Lasso-ST-0.2⁴	3990	58444	-56449	4068	70805	-79844
Lasso-ST-0.25⁴	3172	55617	-64514	3164	67568	-88548
Lasso-AT-0.1	8848	73749	-596	9014	86409	-22855
Lasso-AT-0.2	8739	73453	-2012	8977	86211	-23141
Cluster-50-6	2025	49533	-70900	2047	60715	-92918
Cluster-100-6	2070	50330	-71785	1986	60533	-93514
Cluster-150-6	2238	50971	-70406	1986	60533	-93514
Cluster-200-6	2470	52011	-68772	2231	62147	-92868

Table 7.15: Summary of dependence models in Scenario 1 (Random Sample): number of parameters, log-likelihood and GIC for both fitting time periods $t = 1, \dots, 500$ and $t = 251, \dots, 750$.

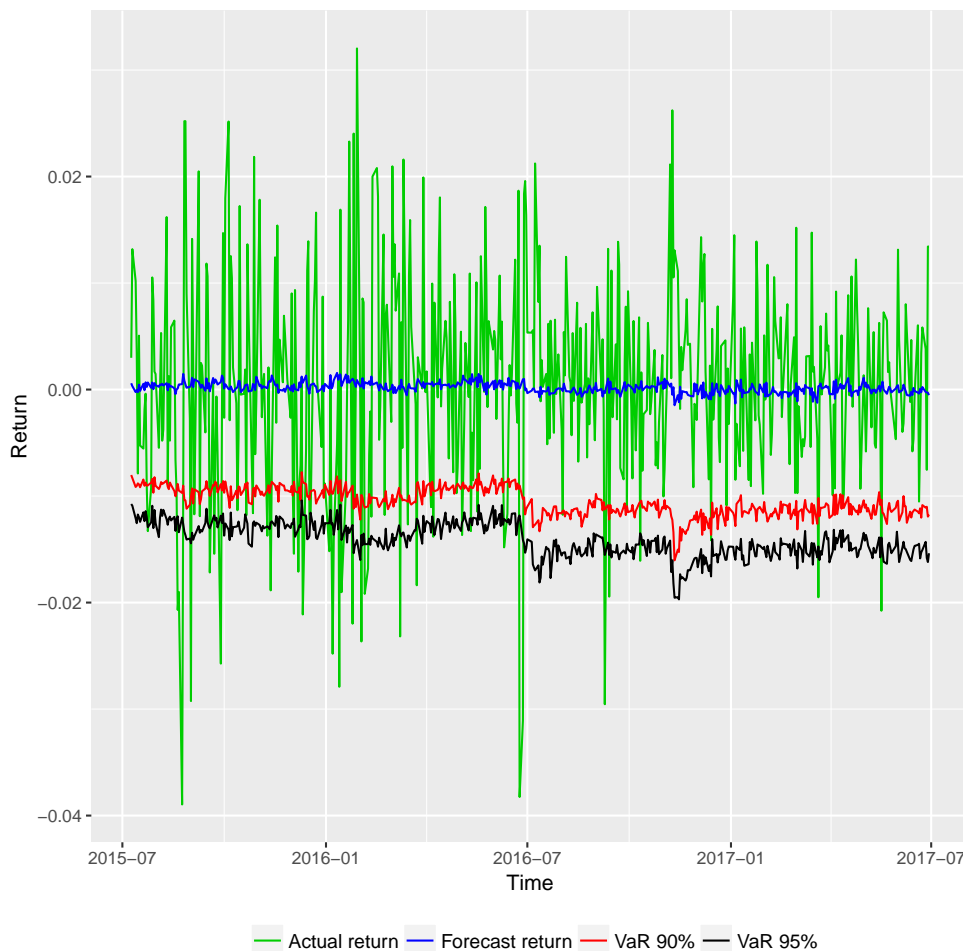


Figure 7.26: Backtest results for VaR Scenario 1 (Random sample): **3-DAG** model with return forecasts, 90% and 95% VaR and actual returns.

We can clearly see that almost all of our models are not conservative enough since most values in Table 7.12 exceed 0.1 and in Table 7.14 exceed 0.05. However, we have to take into account that only 500 observations were used for training of both marginal and dependence models. If we analyse more precisely, only the **3-DAG** model was not rejected for VaR level $\alpha = 0.9$ in the joint conditional coverage test considering a significance level of $\beta = 0.05$, since the p-value exceeds this value. Additionally, we see that for models which capture the exceedance rate rather good, we have to reject the hypothesis of independent hits. This can also be accounted in part to the short test period of 499 observations. However, we also see that for example a **1-DAG** is a too strong simplification and thus, not a good model. The most promising results are obtained for the **Lasso-AT** and **Lasso-ST** models, which were also very conservative in the forecasting. Especially the **Lasso-ST** models are also quite sparse as we can see in Table 7.15. **Cluster** models are presumably too sparse and thus, have too many hits to be considered feasible. Yet, this can be corrected with different threshold dimensions and fill-levels. The **3-DAG** model can be seen in Figure 7.26. We continue with the second scenario.

Scenario 2: Sectors 3 & 6

We consider again the VaR level $\alpha = 0.90$ first. The p-values of the hypothesis tests are given in Table 7.16.

Model	UC	UC-MC	Ind	Ind-MC	CC	CC-MC
Dißmann	0.545	0.501	0.029	0.026	0.076	0.070
1-DAG	0.000	0.001	0.243	0.240	0.000	0.001
2-DAG	0.000	0.001	0.243	0.240	0.000	0.001
3-DAG	0.870	0.811	0.033	0.033	0.103	0.106
4-DAG	0.545	0.501	0.029	0.026	0.076	0.070
Lasso-ST-0.2⁴	0.453	0.415	0.038	0.036	0.088	0.085
Lasso-ST-0.25⁴	0.545	0.501	0.029	0.026	0.076	0.070
Lasso-AT-0.1	0.647	0.603	0.007	0.006	0.024	0.023
Lasso-AT-0.2	0.453	0.415	0.014	0.009	0.036	0.033
Cluster-50-6	0.043	0.039	0.031	0.029	0.012	0.015
Cluster-100-6	0.043	0.039	0.031	0.029	0.012	0.015
Cluster-150-6	0.015	0.012	0.030	0.029	0.005	0.006
Cluster-200-6	0.015	0.012	0.030	0.029	0.005	0.006

Table 7.16: Backtesting results for VaR Scenario 2 (Sectors 3 & 6) and $\alpha = 0.90$: p-values based on the χ^2 distribution (UC, Ind, CC) and Monte Carlo approach (UC-MC, Ind-MC, CC-MC) for the *unconditional coverage*, *independence* and *conditional coverage* test. Values exceeding 0.05 are marked in bold.

The raw number of VaR hits is given by Table 7.17.

7.4 Value at Risk (VaR) Forecasting and Backtesting in 400 Dimensions

Model	Hit rate \hat{p} for 90% VaR
Dißmann	0.110
1-DAG	0.214
2-DAG	0.214
3-DAG	0.104
4-DAG	0.110
Lasso-ST-0.2⁴	0.112
Lasso-ST-0.25⁴	0.110
Lasso-AT-0.1	0.108
Lasso-AT-0.2	0.112
Cluster-50-6	0.130
Cluster-100-6	0.130
Cluster-150-6	0.136
Cluster-200-6	0.136

Table 7.17: Hit rates $\hat{p} = T_1/T$ for 90% VaR in Scenario 2 (Sectors 3 & 6).

For the p-values for VaR level $\alpha = 0.95$, see Table 7.18 and for the corresponding hit rates, see Table 7.19.

Model	UC	UC-MC	Ind	Ind-MC	CC	CC-MC
Dißmann	0.114	0.105	0.019	0.011	0.018	0.018
1-DAG	0.000	0.001	0.078	0.081	0.000	0.001
2-DAG	0.000	0.001	0.078	0.081	0.000	0.001
3-DAG	0.165	0.152	0.014	0.007	0.018	0.017
4-DAG	0.165	0.152	0.014	0.007	0.018	0.017
Lasso-ST-0.2⁴	0.077	0.074	0.026	0.017	0.018	0.016
Lasso-ST-0.25⁴	0.077	0.074	0.026	0.017	0.018	0.016
Lasso-AT-0.1	0.165	0.152	0.014	0.007	0.018	0.017
Lasso-AT-0.2	0.077	0.074	0.026	0.017	0.018	0.016
Cluster-50-6	0.002	0.004	0.016	0.008	0.001	0.001
Cluster-100-6	0.002	0.004	0.016	0.008	0.001	0.001
Cluster-150-6	0.007	0.007	0.031	0.020	0.003	0.003
Cluster-200-6	0.007	0.007	0.031	0.020	0.003	0.003

Table 7.18: Backtesting results for VaR Scenario 2 (Sectors 3 & 6) and $\alpha = 0.95$: p-values based on the χ^2 distribution (UC, Ind, CC) and Monte Carlo approach (UC-MC, Ind-MC, CC-MC) for the *unconditional coverage*, *independence* and *conditional coverage* test. Values exceeding 0.05 are marked in bold.

Model	Hit rate \hat{p} for 95% VaR
Dißmann	0.066
1-DAG	0.160
2-DAG	0.160
3-DAG	0.064
4-DAG	0.064
Lasso-ST-0.2⁴	0.068
Lasso-ST-0.25⁴	0.068
Lasso-AT-0.1	0.064
Lasso-AT-0.2	0.068
Cluster-50-6	0.084
Cluster-100-6	0.084
Cluster-150-6	0.080
Cluster-200-6	0.080

Table 7.19: Hit rates $\hat{p} = T_1/T$ for 95% VaR in Scenario 2 (Sectors 3 & 6).

Finally, the model overview is given in Table 7.20. The most promising models in this scenario are the method of **Dißmann**, as well as **3-DAG** and **4-DAG** and **Lasso-ST** models. In practical applications, we would choose the later because of their particular sparsity with only about 40% of the parameters of the more complex models. They also have the highest p-values considering 90% VaR and are among the best for the 95% VaR. The predicted returns for the **Lasso-ST-0.25⁴** are shown in Figure 7.27. We finish the backtesting section with consideration of the last scenario.

Model	$t = 1, \dots, 500$			$t = 251, \dots, 750$		
	No. par.	Log-lik.	GIC	No. par.	Log-lik.	GIC
Dißmann	8254	70787	-5583	8326	76636	-15963
1-DAG	630	38421	-69423	619	42497	-77724
2-DAG	4433	57853	-.47703	619	42497	-77724
3-DAG	7125	66582	-17689	7498	72341	-22462
4-DAG	7935	69235	-8305	8316	75112	-13096
Lasso-ST-0.2⁴	3599	56205	-58572	3656	61331	-67865
Lasso-ST-0.25⁴	2828	53473	-65887	2838	58342	-75461
Lasso-AT-0.1	8162	70444	-6579	8285	75810	-15062
Lasso-AT-0.2	7994	69969	-8698	8169	75514	-16590
Cluster-50-6	1763	47693	-71311	1900	53148	-80090
Cluster-100-6	1786	48232	-72033	1900	53148	-80090
Cluster-150-6	1900	48893	-71580	1944	53806	-80718
Cluster-200-6	2245	50734	-69820	1944	53806	-80718

Table 7.20: Summary of dependence models in Scenario 2 (Sectors 3 & 6): number of parameters, log-likelihood and GIC for both fitting time periods $t = 1, \dots, 500$ and $t = 251, \dots, 750$.

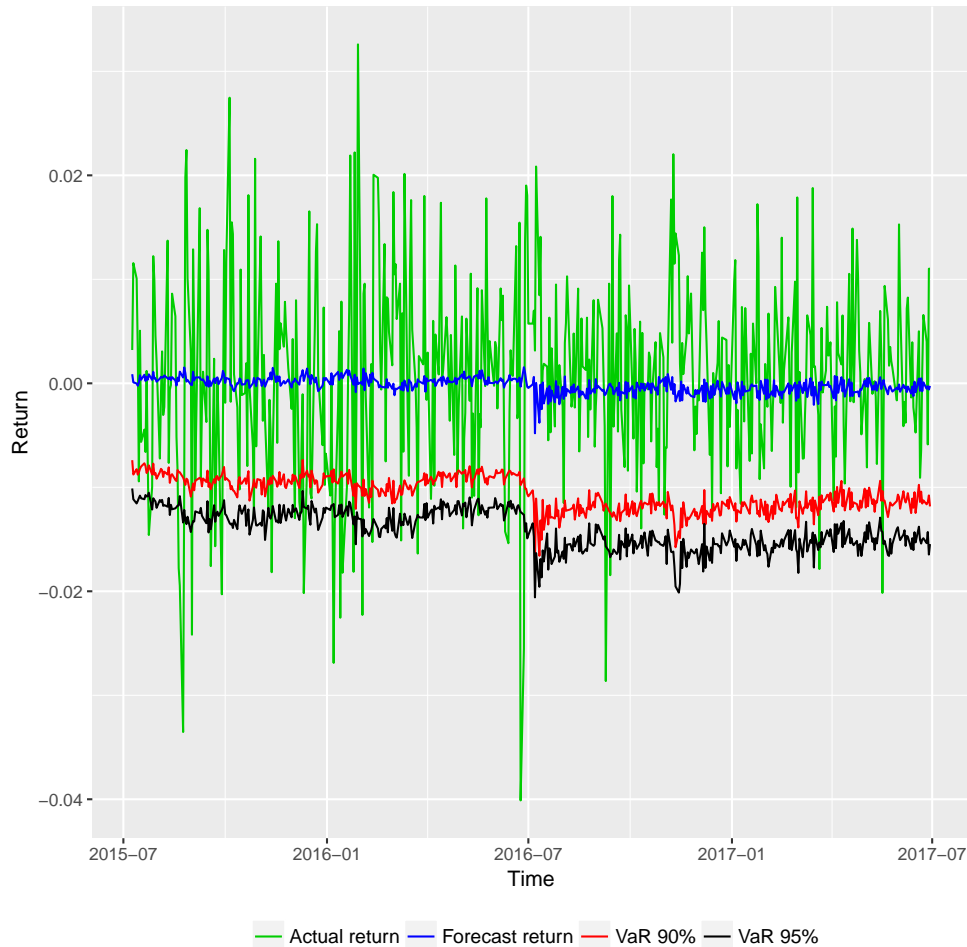


Figure 7.27: Backtest results for VaR Scenario 2 (Sectors 3 & 6): **Lasso-ST-0.25⁴** model with return forecasts, 90% and 95% VaR and actual returns.

Scenario 3: Financial Services Sector

We start with considering the VaR level $\alpha = 0.9$, for which the p-values are given in Table 7.21 and the hit rates in Table 7.22. The corresponding p-values for VaR level $\alpha = 0.95$ are given in Table 7.23, the hit rates can be found in Table 7.24. The number of parameters as well as goodness of fit measures log-likelihood and GIC are displayed in Table 7.25. Summarizing, we see that no model is not rejected for either 90% or 95% VaR at a $\beta = 0.05$ significance level, i. e. all the corresponding p-values in the conditional coverage test are lower for both the asymptotic and the Monte-Carlo thresholds. However, the hypothesis of independent hits is not rejected quite often for a variety of models. Yet, the **Lasso-AT-0.1** model has the most promising results and the least number of VaR exceedances, i. e. hits. To give a reason why the models perform rather poorly, see the log-likelihood values attained by the models in Scenario 3 in Table 7.25 in the first fit period $t = 1, \dots, 500$, and compare it to the other two scenarios in Tables 7.15 and 7.20. There, we see that we obtain much higher log-likelihood values in Scenario 3 compared to the previous two scenarios with about 95,000 compared to about 70,000. Thus, we expect much more dependence between the Financial Services companies which is modelled here.

However, with the rather short time periods of 500 time points for model fitting, we can expect that this low number of observations is not sufficient to grasp the variety of effects present in the Financial Services sector. Another possible explanation is that the Financial Services sector behaves more dynamically and hence, the models would be required to be updated more often and not only after 250 days, which is approximately one trading year. However, especially this gives rise to the need for fast algorithms which are provided in this thesis, as for example the **Lasso** methods, which proved very feasible in this application. Finally, we show the forecast returns and VaR for the **Lasso-AT-0.1** algorithm in Figure 7.28

Model	UC	UC-MC	Ind	Ind-MC	CC	CC-MC
Dißmann	0.003	0.004	0.088	0.083	0.003	0.003
1-DAG	0.000	0.001	0.014	0.009	0.000	0.001
2-DAG	0.000	0.001	0.050	0.046	0.000	0.001
3-DAG	0.001	0.003	0.033	0.033	0.000	0.001
4-DAG	0.001	0.002	0.090	0.090	0.001	0.001
Lasso-ST-0.2⁴	0.001	0.002	0.090	0.090	0.001	0.001
Lasso-ST-0.25⁴	0.004	0.004	0.139	0.137	0.006	0.007
Lasso-AT-0.1	0.007	0.006	0.111	0.106	0.007	0.007
Lasso-AT-0.2	0.007	0.006	0.111	0.106	0.007	0.007
Cluster-50-6	0.000	0.001	0.092	0.091	0.000	0.001
Cluster-100-6	0.000	0.001	0.055	0.054	0.000	0.001
Cluster-150-6	0.000	0.001	0.049	0.044	0.000	0.001
Cluster-200-6	0.000	0.001	0.064	0.057	0.000	0.001

Table 7.21: Backtesting results for VaR Scenario 3 (Financial Services) and $\alpha = 0.90$: p-values based on the χ^2 distribution (UC, Ind, CC) and Monte Carlo approach (UC-MC, Ind-MC, CC-MC) for the *unconditional coverage*, *independence* and *conditional coverage* test. Values exceeding 0.05 are marked in bold.

Model	Hit rate \hat{p} for 90% VaR
Dißmann	0.144
1-DAG	0.218
2-DAG	0.162
3-DAG	0.148
4-DAG	0.150
Lasso-ST-0.2⁴	0.150
Lasso-ST-0.25⁴	0.142
Lasso-AT-0.1	0.140
Lasso-AT-0.2	0.140
Cluster-50-6	0.182
Cluster-100-6	0.178
Cluster-150-6	0.182
Cluster-200-6	0.184

Table 7.22: Hit rates $\hat{p} = T_1/T$ for 90% VaR in Scenario 3 (Financial Services).

7.4 Value at Risk (VaR) Forecasting and Backtesting in 400 Dimensions

Model	UC	UC-MC	Ind	Ind-MC	CC	CC-MC
Dißmann	0.004	0.005	0.123	0.186	0.005	0.005
1-DAG	0.000	0.001	0.044	0.032	0.000	0.001
2-DAG	0.000	0.001	0.067	0.062	0.000	0.001
3-DAG	0.001	0.004	0.022	0.014	0.000	0.001
4-DAG	0.001	0.004	0.070	0.076	0.001	0.001
Lasso-ST-0.2⁴	0.002	0.004	0.054	0.043	0.002	0.001
Lasso-ST-0.25⁴	0.004	0.005	0.041	0.026	0.002	0.003
Lasso-AT-0.1	0.020	0.022	0.060	0.058	0.012	0.011
Lasso-AT-0.2	0.012	0.010	0.023	0.014	0.003	0.004
Cluster-50-6	0.000	0.001	0.131	0.202	0.000	0.001
Cluster-100-6	0.000	0.001	0.074	0.077	0.000	0.001
Cluster-150-6	0.000	0.001	0.217	0.333	0.000	0.001
Cluster-200-6	0.000	0.001	0.110	0.163	0.000	0.001

Table 7.23: Backtesting results for VaR Scenario 3 (Financial Services) and $\alpha = 0.95$: p-values based on the χ^2 distribution (UC, Ind, CC) and Monte Carlo approach (UC-MC, Ind-MC, CC-MC) for the *unconditional coverage*, *independence* and *conditional coverage* test. Values exceeding 0.05 are marked in bold.

Model	Hit rate \hat{p} for 95% VaR
Dißmann	0.082
1-DAG	0.186
2-DAG	0.094
3-DAG	0.086
4-DAG	0.086
Lasso-ST-0.2⁴	0.084
Lasso-ST-0.25⁴	0.082
Lasso-AT-0.1	0.074
Lasso-AT-0.2	0.078
Cluster-50-6	0.122
Cluster-100-6	0.110
Cluster-150-6	0.120
Cluster-200-6	0.106

Table 7.24: Hit rates $\hat{p} = T_1/T$ for 95% VaR in Scenario 3 (Financial Services).

Model	$t = 1, \dots, 500$			$t = 251, \dots, 750$		
	No. par.	Log-lik.	GIC	No. par.	Log-lik.	GIC
Dißmann	8547	97221	-53078	8868	113586	-79901
1-DAG	693	60875	-113468	708	75302	-142116
2-DAG	4267	82195	-99232	4719	98696	-124462
3-DAG	7795	93782	-59950	7875	109558	-90046
4-DAG	8407	95927	-53059	8501	111727	-82934
Lasso-ST-0.2⁴	2878	78652	-115427	2822	93437	-145913
Lasso-ST-0.25⁴	3686	81752	-108204	3593	96551	-139363
Lasso-AT-0.1	8676	97490	-51245	8996	113794	-77957
Lasso-AT-0.2	8549	97213	-53025	8795	113393	-80862
Cluster-50-6	2140	73629	-117277	2073	87542	-146162
Cluster-100-6	2025	74054	-119944	2012	88252	-148543
Cluster-150-6	2025	74054	-119944	1856	87049	-148579
Cluster-200-6	2248	75331	-118966	2761	92731	-145497

Table 7.25: Summary of dependence models in Scenario 3 (Financial Services): number of parameters, log-likelihood and GIC for both fitting time periods $t = 1, \dots, 500$ and $t = 251, \dots, 750$.

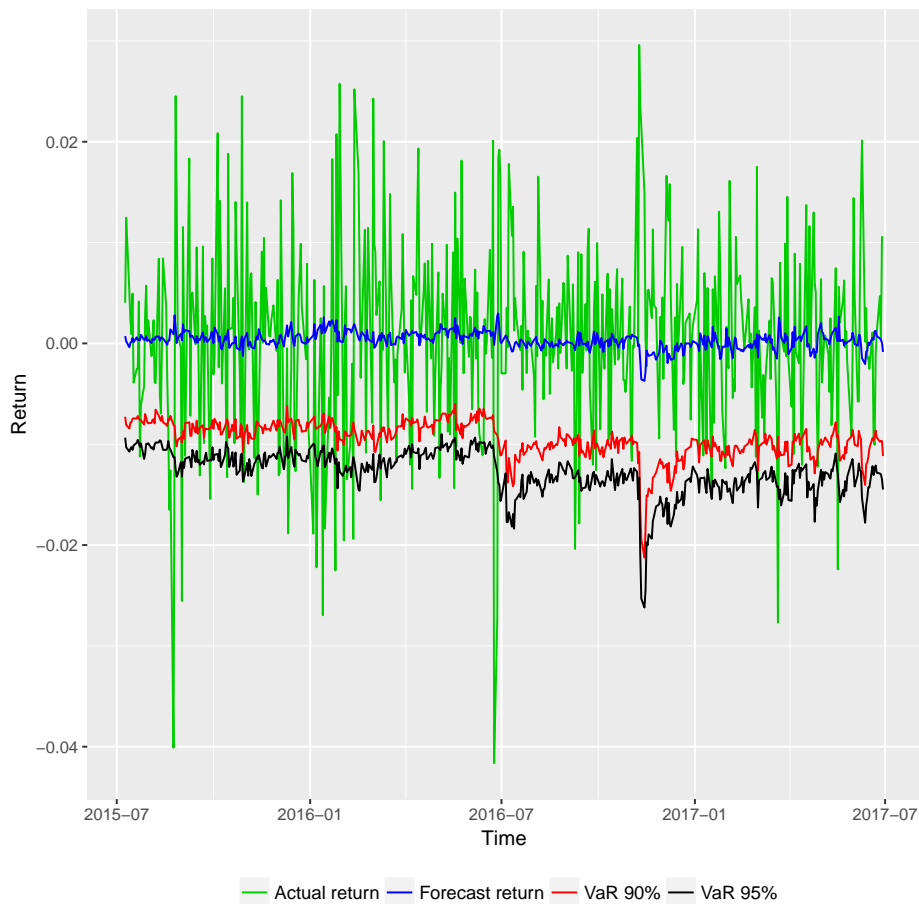


Figure 7.28: Backtest results for VaR Scenario 3 (Financial Services): **Lasso-AT-0.1** model with return forecasts, 90% and 95% VaR and actual returns.

Chapter 8

Conclusion and Outlook

We will now summarize the contribution of this thesis and afterwards point out additional research directions which might be possible based on the work presented here.

8.1 Contribution

First of all, we recapitulated both copulas and vine copulas as basis for our work in Chapter 2, also shedding light on the current benchmark algorithm. Afterwards, we introduced statistical prerequisites with respect to graphical models, the Lasso and structural equation models in Chapter 3.

In Chapter 4, we first laid out the connection between k -truncated R-vines and k -DAGs for $k \geq 1$, based on structural equation modes. This was the basis and motivation for finding an inverse transformation, allowing us to use algorithms for the fast estimation of k -DAGs, also in high dimensions, to fit R-vine structures. We proved under sufficient conditions when such a transformation can be applied based on strong assumptions. For specific k -DAGs, we were able to prove that corresponding k -truncated R-vine representations exist and we also succeeded to give necessary conditions for such a representation. Yet, based on the very strong assumptions, we motivated a more heuristic approach to find R-vine representations of arbitrary k -DAGs. These are however not k -truncated any more. We showed how conditional independences in a DAG can be exploited by an R-vine and gave a data application showing that our approach is able to outperform the benchmark method on the one hand side and Gaussian models on the other hand side.

Chapter 5 used the correspondence between R-vines and structural equation models. Exploiting the well-known Lasso, we showed how the concept of regularization paths can lead over to R-vines. Thus, we first have a proxy for the R-vine structure based on the sparse solution of a system of linear equations. Secondly, the regularization path concept allowed us to flexibly account for sparsity in the SEM by varying the degree of penalization and thus, induce sparsity in the R-vine. We showed that our approach is multiple times faster than current benchmark spanning-tree algorithms and provided different ways for finding specific thresholds.

The concept introduced in Chapter 6 takes another different approach by implementing a *divide-and-conquer* strategy. By using the graphical Lasso, we showed how clustering of ultra high dimensional data sets can be attained flexibly to obtain tractable data sub-

sets. Similar to using the *d-separation* for DAGs in Chapter 4, we used the *graphical separation* in undirected graphical models to estimate sparse models. However, this time it was not only a method to obtain parsimony, but also to actually improve the goodness of fit by lowering computational complexity for a more precise optimization which was not feasible before. These two approaches together founded a powerful strategy to make R-vines firstly applicable for data sets with several thousands of dimensions. We introduced a data set with more than 2,000 variables on which we accomplished to estimate joint non-Gaussian dependence models with different degrees of sparsity and outperform pure Gaussian methods drastically, all this in feasible time.

Chapter 7 then provides the numerical experiments to justify our assertions with respect to quality and efficiency of our proposed methods. In Section 7.2, we considered differently sparse scenarios in 85 dimensions to simulate from and compare our methods with the benchmark algorithm and demonstrate the advantages of our novel procedures. Considering higher dimensions, we modelled up to 1750 dimensions for a joint runtime and goodness of fit analysis based on real world data in Section 7.3. These results confirmed our findings in previous data applications and simulations, showing that our methods are very well designed to be used for high dimensional problems. Finally, in Section 7.4, we tackled a real world problem and estimated Value at Risk bounds for three different portfolios of 400 stocks each, using our recently developed algorithms and the benchmark. We saw that for each portfolio composition with different characteristics, i. e. more or less dependence or cross-sectoral relationships, our algorithms provided good Value at Risk bounds. Furthermore, they did so using much less parameters compared to the benchmark algorithm.

8.2 Future Research Directions

We now want to mention some aspects which can be seen as starting points for future research, improving both quality but also efficiency of next generation model selection algorithms for R-vines.

With respect to Chapter 4, the natural question arises for which $k \geq 1$, k -DAGs shall be represented by R-vines. A natural choice is to estimate DAGs for increasing number of k until no new edges are introduced. This works if the corresponding function to optimize is using a penalization, as for example *BIC*, which we use. Another option is to calculate the Kullback-Leibler divergences (3.4) between k -DAGs for an increasing sequence of $k \geq 1$. If the divergence between two subsequently estimated k -DAGs falls below a certain threshold, no sufficient improvement is attained by higher k , and the corresponding k -DAG shall be represented. Another question is whether DAGs estimated with any algorithm and not only Hill-Climbing style search algorithms can be used. For example, also *constraint based* methods can be employed. However, they do not directly adhere to the concept of k -DAGs because of their design. Yet, an option would be to exchange conditional independence tests relying on Gaussian data, with tests which do not assume this property. Those might possibly be more specifically designed to *copula-*

data.

The Lasso method proposed in Chapter 5 relied on an association between zero coefficients in the SEM and independence copulas in the R-vine. However, in the Gaussian case, a zero coefficient in the SEM is not equivalent to a zero partial correlation and hence, an independence copula. This could however be included with additional constraints on the corresponding *blacklist* and *whitelist* sets. Yet, this would increase computation time significantly in large systems. Additionally, the approach is based on the selection of a threshold value λ for the corresponding penalization. Normally, in linear models this is achieved via cross-validation, which is however not feasible in high dimensional R-vines because repeated model estimation on different test data sets requires too much time. Also, the proceeding of thresholding every equation in the SEM with the same λ is only a rule of thumb and can be improved upon. A promising idea is to use not the $\|\cdot\|_1$ penalty as in the Lasso, but the *elastic net* of Zou and Hastie (2005). More precisely, the penalty

$$\lambda \sum_{\ell=1}^p \|\varphi_{\ell}\|_1$$

is exchanged by

$$\alpha \left(\lambda \sum_{\ell=1}^p \|\varphi_{\ell}\|_1 \right) + (1 - \alpha) \left(\lambda \sum_{\ell=1}^p \|\varphi_{\ell}\|_2 \right),$$

which is alleged to work better when regressors show dependency, which is what we expect from our data sets most often.

The method proposed in Chapter 6 also relies heavily on the choice of a penalization parameter λ as in Chapter 5. However, in both cases there is currently only the option to evaluate models for a grid of threshold values. We would consider using special techniques for choosing λ in the context of the graphical Lasso, see for example the approach of Liu et al. (2010). Additionally, huge gains could be attained given that the graphs in a solution path only change slightly such that previously estimated pair copulas for a specific partition generated by \mathcal{G}_T might also be occurring in the partition generated by \mathcal{G}_{T+1} . However, this would in some sense imply to impose the proximity condition on the solution path for the graphical models, which is a combinatorial restriction and hence, might lead to non-optimal results in terms of the achieved log-likelihoods of the models. Furthermore, going to even higher dimensional models, more elaborate search strategies for the fill level k_F need to be employed, as currently only values which proved as working well in our experiments are considered. However, this value must be handled with care as it can also lead again to unnecessarily over-parametrized models. Alternatively, one could consider using *sub-clusterings* for extremely ultra high dimensional datasets. Hence, we would initially cluster a large dataset in dimension d using a clustering algorithm, as for example the *graphical Lasso* to obtain a specific intermediate *threshold dimension* $d_T^1 \ll d$. If the resulting sub-sets are still too large to be handled, the *connected components* of maximum size d_T^1 could be clustered again using a suitable threshold dimension $d_T^2 \ll d_T^1$. These clusters could then be reassembled into a joint model using two step procedure.

However, one has to be very careful with specifying the corresponding threshold dimensions. Another option is to use different clustering algorithms, and not only the graphical Lasso.

Considering high dimensions, also the possibility of introducing *latent factors* should be taken into account, see Krupskii and Joe (2013). This can make models much more parsimonious by removing the effect of a latent factor, influencing the dependence structure of the observed variable. All the proposed methods can also be considered together with the effects of latent factors. Note furthermore, that even when a common latent factor is not sensible to be included in a model, there are several ways of model diagnostics as *principal component analysis* which can be ran as a starting point for the analysis.

Additionally, even though all of the applications in this thesis cover financial data, this is not a restriction but happens because of the wide availability of this data and well known estimation of the marginals. Especially the domain of *computational biology* exhibits large data sets which require different handling of the marginal distributions and the dependence part. Yet, these data sets offer highly interesting insights and large possibilities for high dimensional dependence modelling with sparse vine copulas, and should be elaborated more in the future.

Bibliography

- Aas, K. (2016). Pair-copula constructions for financial applications: A review. *Econometrics* 4(4), 43.
- Aas, K., C. Czado, A. Frigessi, and H. Bakken (2009). Pair-copula constructions of multiple dependence. *Insurance, Mathematics and Economics* 44, 182–198.
- Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In B. N. Petrov and F. Csaki (Eds.), *Proceedings of the Second International Symposium on Information Theory Budapest*, Akademiai Kiado, pp. 267–281.
- Andersson, S. A. and M. D. Perlman (1998). Normal linear regression models with recursive graphical markov structure. *Journal of Multivariate Analysis* 66, 133–187.
- Arnborg, S. and P. A. Corneil, Derek G. (1987). Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic and Discrete Methods* 8(2), 277–284.
- Baba, K., R. Shibata, and M. Sibuya (2004). Partial correlation and conditional correlation as measures of conditional independence. *Australian & New Zealand Journal of Statistics* 46(4), 657–664.
- Basel Committee on Banking Supervision (2006). International Convergence of Capital Measurement and Capital Standards. <http://www.bis.org/publ/bcbs128.pdf>. Accessed: 08.09.2017.
- Basel Committee on Banking Supervision (2011). Basel III: A global regulatory framework for more resilient banks and banking systems. <http://www.bis.org/publ/bcbs189.pdf>. Accessed: 08.09.2017.
- Basel Committee on Banking Supervision (2013). Fundamental review of the trading book: A revised market risk framework. <http://www.bis.org/publ/bcbs265.pdf>. Accessed: 08.09.2017.
- Bauer, A. and C. Czado (2016). Pair-Copula Bayesian networks. *Journal of Computational and Graphical Statistics* 25(4), 1248–1271.
- Bauer, A., C. Czado, and T. Klein (2012). Pair-copula constructions for non-Gaussian DAG models. *Canadian Journal of Statistics* 40, 86–109.
- Bedford, T. and R. Cooke (2001). Probability density decomposition for conditionally dependent random variables modeled by vines. *Annals of Mathematics and Artificial Intelligence* 32, 245–268.

Bibliography

- Bedford, T. and R. Cooke (2002). Vines - a new graphical model for dependent random variables. *The Annals of Statistics* 30(4), 1031–1068.
- Bellman, R. E. (2015). *Adaptive control processes: a guided tour*. Princeton University Press.
- Bollen, K. A. (1989). *Structural Equations with Latent Variables* (1st ed.). Chichester, United Kingdom: John Wiley & Sons,.
- Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics* 31, 307–327.
- Bollerslev, T. (1987). A conditionally heteroscedastic time series model for speculative prices and rates of return. *Review of Economics and Statistics* 69, 542–547.
- Brechmann, E. and C. Czado (2013). Risk management with high-dimensional vine copulas: An analysis of the Euro Stoxx 50. *Statistics & Risk Modeling* 30, 307–342.
- Brechmann, E., C. Czado, and K. Aas (2012). Truncated regular vines in high dimensions with application to financial data. *Canadian Journal of Statistics* 40, 68–85.
- Brechmann, E. C. and H. Joe (2014). Parsimonious parameterization of correlation matrices using truncated vines and factor analysis. *Computational Statistics & Data Analysis* 77, 233–251.
- Brechmann, E. C. and U. Schepsmeier (2013). Modeling Dependence with C- and D-Vine Copulas: The R package CDVine. *Journal of Statistical Software* 52(3), 1–27.
- Brockwell, P. J. and R. A. Davis (2016). *Introduction to Time Series and Forecasting*. New York: Springer.
- Chen, S. X. and T.-M. Huang (2007). Nonparametric estimation of copula functions for dependence modelling. *Canadian Journal of Statistics* 35(2), 265–282.
- Christoffersen, P. F. (2012). *Elements of financial risk management*. London, United Kingdom: Academic Press.
- Csardi, G. and T. Nepusz (2006). The igraph software package for complex network research. *InterJournal Complex Systems*, 1695.
- Czado, C., S. Jeske, and M. Hofmann (2013). Selection strategies for regular vine copulae. *Journal de la Société Française de Statistique* 154, 174–191.
- Dempster, A. P. (1972). Covariance selection. *Biometrics* 28(1), 157–175.
- Dißmann, J., E. Brechmann, C. Czado, and D. Kurowicka (2013). Selecting and estimating regular vine copulae and application to financial returns. *Computational Statistics & Data Analysis* 52(1), 52–59.

- Elidan, G. (2010). Copula Bayesian Networks. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta (Eds.), *Advances in Neural Information Processing Systems 23*, pp. 559–567. Curran Associates, Inc.
- Fan, Y. and C. Y. Tang (2013). Tuning parameter selection in high dimensional penalized likelihood. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 75(3), 531–552.
- Friedman, J., T. Hastie, and R. Tibshirani (2008). Sparse inverse covariance estimation with the graphical Lasso. *Biostatistics* 9(3), 432.
- Friedman, J., T. Hastie, and R. Tibshirani (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software* 33(1), 1–22.
- Frommlet, F., A. Chakrabarti, M. Murawska, and M. Bogdan (2011). Asymptotic Bayes optimality under sparsity for generally distributed effect sizes under the alternative. Technical report.
- Genest, C., K. Ghoudi, and L. Rivest (1995). A semi-parametric estimation procedure of dependence parameters in multivariate families of distributions. *Biometrika* 82, 543–552.
- Ghalanos, A. (2015). *rugarch: Univariate GARCH models*. R package version 1.3-6.
- Gruber, L. and C. Czado (2015a). Bayesian model selection of regular vine copulas. *Preprint*.
- Gruber, L. and C. Czado (2015b). Sequential bayesian model selection of regular vine copulas. *Bayesian Analysis* 10, 937–963.
- Han, J., J. Pei, and M. Kamber (2011). *Data mining: concepts and techniques*. Elsevier.
- Hansen, B. E. (1994). Autoregressive conditional density estimation. *International Economic Review* 35(3), 705–730.
- Hastie, T., R. Tibshirani, and M. Wainwright (2015). *Statistical Learning with Sparsity The Lasso and Generalizations*. Boca Raton, Florida: CRC Press.
- Hobæk Haff, I., K. Aas, A. Frigessi, and V. L. Graziani (2016). Structure learning in Bayesian Networks using regular vines. *Computational Statistics & Data Analysis* 101, 186–208.
- Hoyle, R. H. (1995). *Structural Equation Modeling* (1st ed.). Thousand Oaks, California: SAGE Publications.
- Joe, H. (1996). Families of m -variate distributions with given margins and $m(m - 1)/2$ bivariate dependence parameters. In L. Rüschendorf, B. Schweizer, and M. D. Taylor (Eds.), *Distributions with fixed marginals and related topics*, pp. 120–141. Hayward: Institute of Mathematical Statistics.

Bibliography

- Joe, H. (1997). *Multivariate Models and Dependence Concepts*. London, United Kingdom: Chapman & Hall.
- Joe, H. (2005). Asymptotic efficiency of the two stage estimation method for copula-based models. *Journal of Multivariate Analysis* 94, 401–419.
- Joe, H. (2014). *Dependence Modeling with Copulas*. Boca Raton, Florida: CRC Press.
- Kaplan, D. (2009). *Structural Equation Modeling: Foundations and Extensions* (2nd ed.). Thousand Oaks, California: SAGE Publications.
- Kenourgios, D., A. Samitas, and N. Paltalidis (2011). Financial crises and stock market contagion in a multivariate time-varying asymmetric framework. *Journal of International Financial Markets, Institutions and Money* 21(1), 92–106.
- Koller, D. and N. Friedman (2009). *Probabilistic Graphical Models: Principles and Techniques* (1st ed.). Cambridge, Massachusetts: MIT Press.
- Kovács, E. and T. Szántai (2016). On the connection between cherry-tree copulas and truncated R-vine copulas. *arXiv preprint arXiv:1604.03269*.
- Kraus, D. and C. Czado (2017). Growing simplified vine copula trees: improving Dißmann’s algorithm. *arXiv preprint arXiv:1703.05203*.
- Krumsiek, J., K. Suhre, T. Illig, J. Adamski, and F. J. Theis (2011). Gaussian graphical modeling reconstructs pathway reactions from high-throughput metabolomics data. *BMC systems biology* 5(1), 21.
- Krupskii, P. and H. Joe (2013). Factor copula models for multivariate data. *Journal of Multivariate Analysis* 120, 85–101.
- Kullback, S. and R. Leibler (1951). On information and sufficiency. *Annals of Mathematical Statistics* 22, 79–86.
- Kurowicka, D. and R. Cooke (2006). *Uncertainty Analysis and High Dimensional Dependence Modelling* (1st ed.). Chichester, United Kingdom: John Wiley & Sons,.
- Kurowicka, D. and H. Joe (2011). *Dependence Modeling - Vine Copula Handbook*. Singapore: World Scientific Publishing Co.
- Lauritzen, S. L. (1996). *Graphical Models* (1st ed.). Oxford, United Kingdom: University Press.
- Liu, H., K. Roeder, and L. Wasserman (2010). Stability approach to regularization selection (stars) for high dimensional graphical models. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems, NIPS’10, USA*, pp. 1432–1440. Curran Associates Inc.

- Margaritis, D. (2003). Learning Bayesian network model structure from data. Technical report, Carnegie-Mellon University Pittsburgh, PA School of Computer Science.
- Mazumder, R. and T. Hastie (2012). Exact Covariance Thresholding into Connected Components for Large Scale Graphical Lasso. *Journal of Machine Learning Research* 13, 723–736.
- McNeil, A. J., R. Frey, and P. Embrechts (2015). *Quantitative risk management: Concepts, techniques and tools*. Princeton university press.
- Meinshausen, N. and P. Bühlmann (2006, 06). High-dimensional graphs and variable selection with the Lasso. *The Annals of Statistics* 34(3), 1436–1462.
- Müller, D. and C. Czado (2017a). Dependence Modeling in Ultra High Dimensions with Vine Copulas and the Graphical Lasso. *arXiv preprint arXiv:1709.05119*.
- Müller, D. and C. Czado (2017b). Representing Sparse Gaussian DAGs as Sparse R-vines Allowing for Non-Gaussian Dependence. *To appear in the Journal of Computational and Graphical Statistics*.
- Müller, D. and C. Czado (2017c). Selection of Sparse Vine Copulas in High Dimensions with the Lasso. *arXiv preprint arXiv:1705.05877*.
- Nagarajan, R., M. Scutari, and S. Lebre (2013). *Bayesian Networks in R with Applications in Systems Biology*. New York: Springer. ISBN 978-1461464457.
- Nagler, T. (2014). Kernel Methods for Vine Copula Estimation. Master’s thesis, Technische Universität München.
- Nelsen, R. B. (2006). *An Introduction to Copulas*. New York: Springer.
- Pereira, G. A., A. Veiga, T. Erhardt, and C. Czado (2017). A periodic spatial vine copula model for multi-site streamflow simulation. *Electric Power Systems Research* 152, 9 – 17.
- Peters, J. and P. Bühlmann (2014). Identifiability of gaussian structural equation models with equal error variances. *Biometrika* 101, 219–228.
- Pircalabelu, E., G. Claeskens, and I. Gijbels (2017). Copula directed acyclic graphs. *Statistics and Computing* 27(1), 55–78.
- Prim, R. C. (1957). Shortest connection networks and some generalizations. *Bell System Technical Journal* 36, 1389–1401.
- R Core Team (2017). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing.
- Rütimann, P., P. Bühlmann, et al. (2009). High dimensional sparse covariance estimation via directed acyclic graphs. *Electronic Journal of Statistics* 3, 1133–1160.

Bibliography

- Ryan, J. A. and J. M. Ulrich (2017). *quantmod: Quantitative Financial Modelling Framework*. R package version 0.4-10.
- Salmon, F. (2012). The formula that killed wall street. *Significance* 9(1), 16–20.
- Schamberger, B. (2015). Bayesian Analysis of the One-Factor Copula Model with Applications to Finance. Master’s thesis, Technische Universität München.
- Schepsmeier, U., J. Stöber, E. C. Brechmann, B. Graeler, T. Nagler, and T. Erhardt (2017). *VineCopula: Statistical Inference of Vine Copulas*. R package version 2.1.2.
- Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics* 6(2), 461–464.
- Scutari, M. (2010). Learning Bayesian Networks with the bnlearn R package. *Journal of Statistical Software* 35(3), 1–22.
- Shachter, R. D. (1998). Bayes-ball: Rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams). In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pp. 480–487. Morgan Kaufmann Publishers Inc.
- Sklar, A. (1959). Fonctions de répartition à n dimensions et leurs marges. *Publ. Inst. Stat. Univ. Paris* 8, 229–231.
- Spirites, P., C. N. Glymour, and R. Scheines (2000). *Causation, prediction, and search*. Cambridge, Massachusetts: MIT Press.
- Stöber, J., H. Joe, and C. Czado (2013). Simplified pair copula constructions—limitations and extensions. *Journal of Multivariate Analysis* 119(0), 101 – 118.
- Taylor, S. J. (2008). *Modelling financial time series*. Singapore: World Scientific.
- Tibshirani, R. (1994). Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society, Series B* 58, 267–288.
- Whittaker, J. (1990). *Graphical Models in Applied Multivariate Statistics* (1st ed.). Chichester, United Kingdom: John Wiley & Sons,.
- Wickham, H. (2009). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York.
- Witten, D. M., J. H. Friedman, and S. Noah (2011). New Insights and Faster Computations for the Graphical Lasso. *Journal of Computational and Graphical Statistics* 20(4), 892–900.
- Yu, H., W. I. T. Uy, and J. Dauwels (2017, Feb). Modeling spatial extremes via ensemble-of-trees of pairwise copulas. *IEEE Transactions on Signal Processing* 65(3), 571–586.

Zhao, T., X. Li, H. Liu, K. Roeder, J. Lafferty, and L. Wasserman (2015). *huge: High-Dimensional Undirected Graph Estimation*. R package version 1.2.7.

Zou, H. and T. Hastie (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67(2), 301–320.

Appendix A

Supplementary Material to Chapter 4

A.1 DAGs Estimated on the Euro Stoxx 50 Data Set

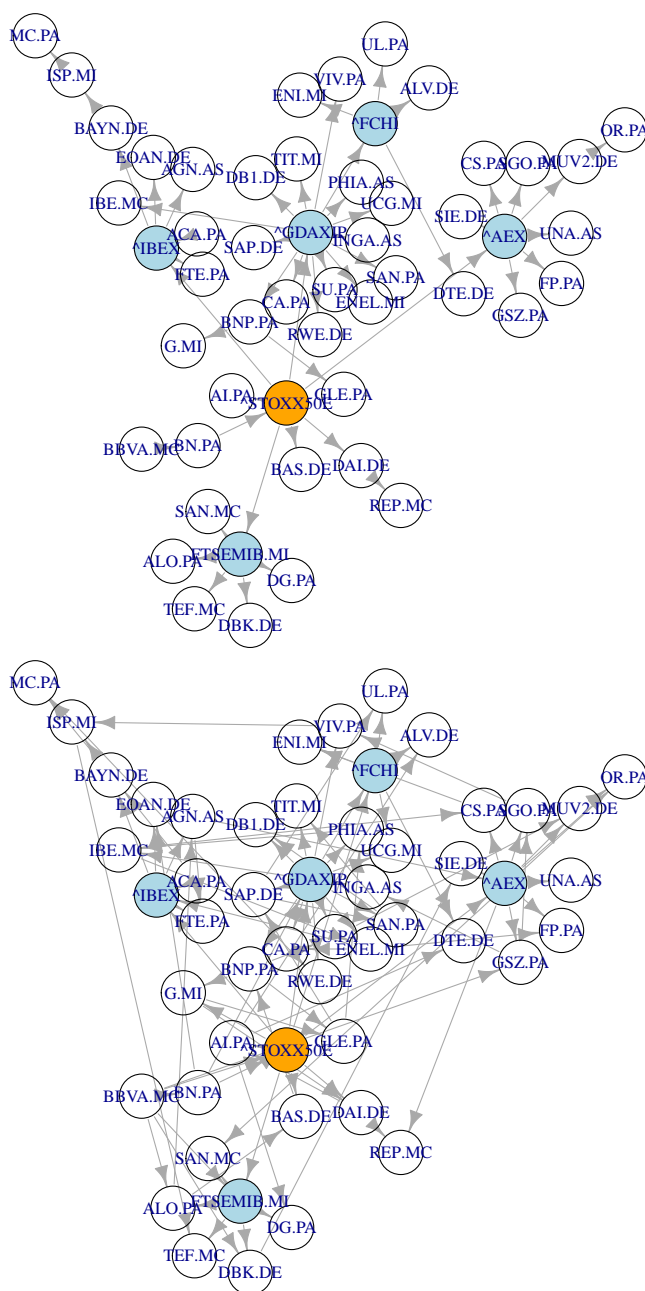


Figure A.1: DAGs estimated on the Euro Stoxx 50 data set with at most $k = 1, 2$ parents (upper, lower).

A.1 DAGs Estimated on the Euro Stoxx 50 Data Set

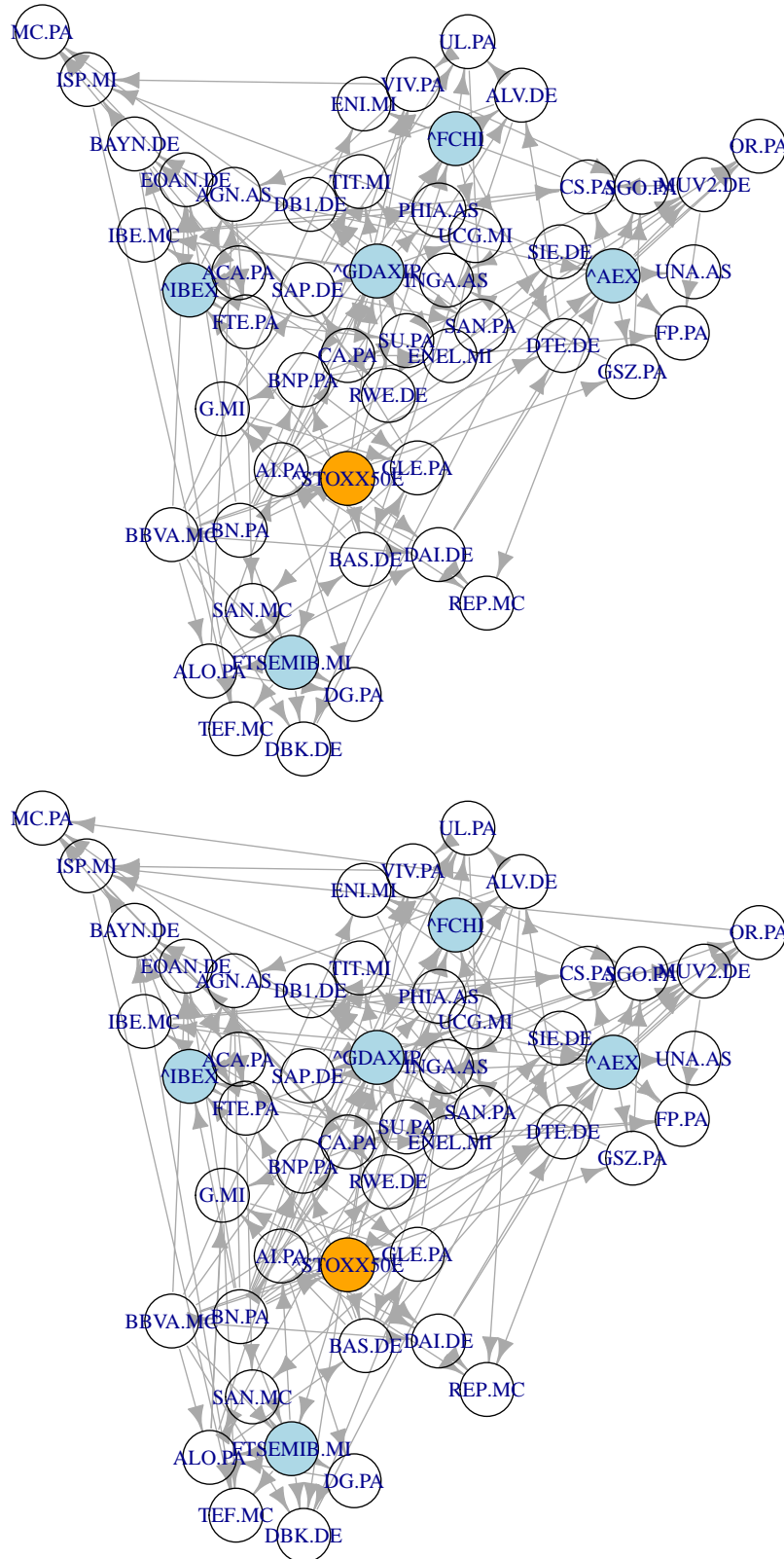


Figure A.2: DAGs estimated on the Euro Stoxx 50 data set with at most $k = 3, 4$ parents (upper, lower).

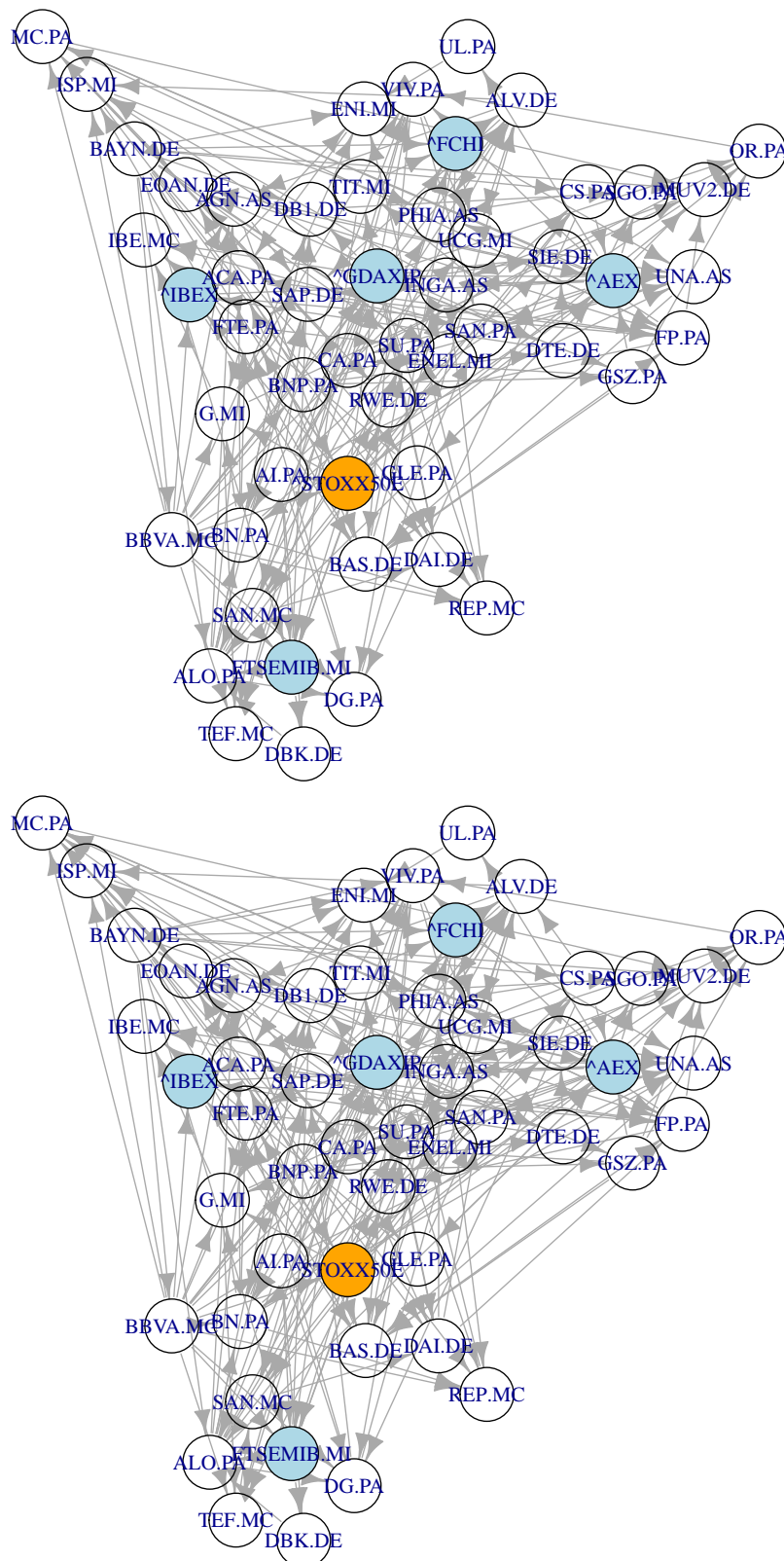


Figure A.3: DAGs estimated on the Euro Stoxx 50 data set with at most $k = 5, 6$ parents (upper, lower).

A.1 DAGs Estimated on the Euro Stoxx 50 Data Set

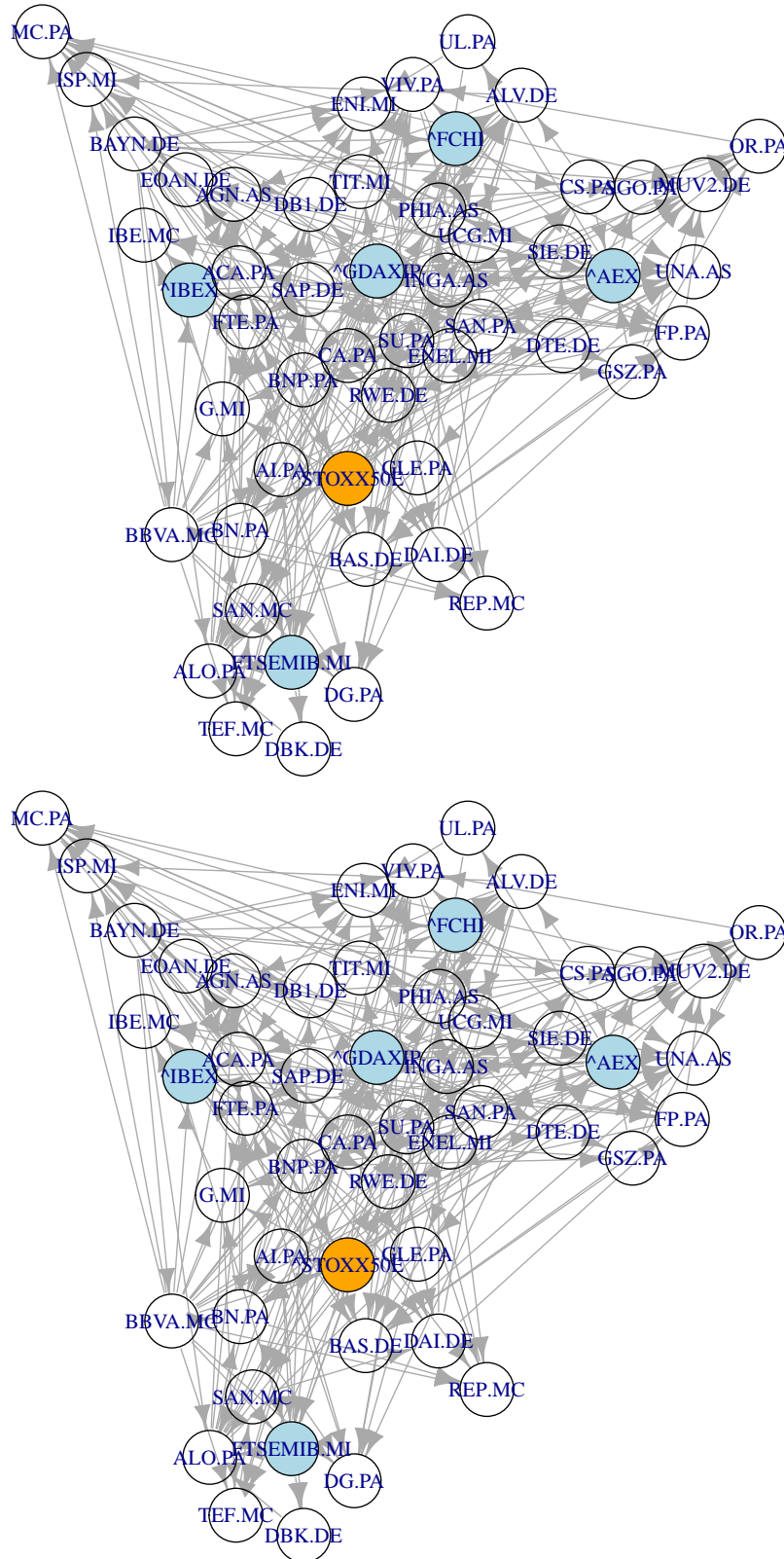


Figure A.4: DAGs estimated on the Euro Stoxx 50 data set with at most $k = 7, 8$ parents (upper, lower).

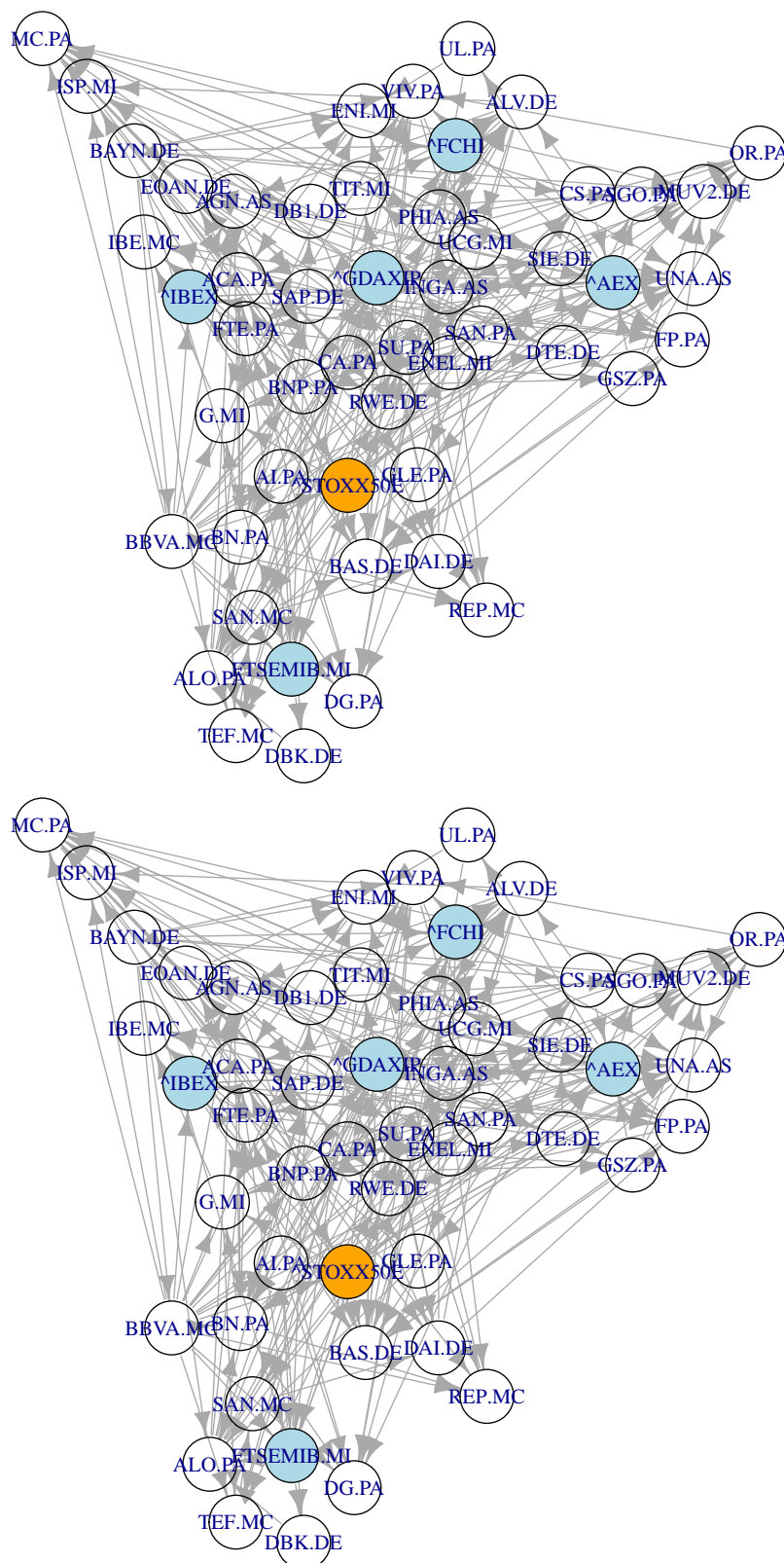


Figure A.5: DAGs estimated on the Euro Stoxx 50 data set with at most $k = 9, 10$ parents (upper, lower).

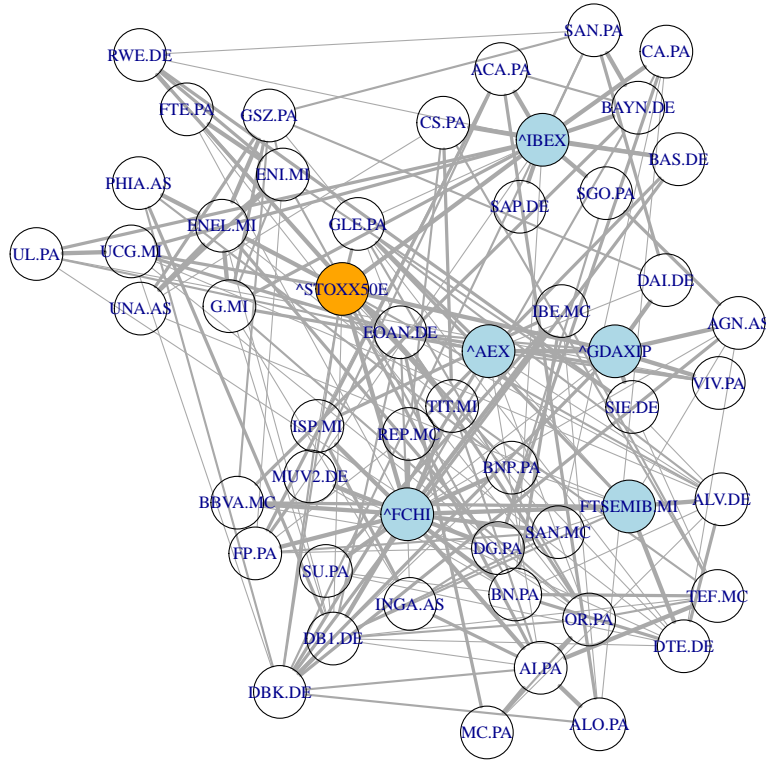


Figure A.6: Undirected weighted graph \mathcal{H} defined by the union of $\mathcal{G}_1^s, \dots, \mathcal{G}_4^s$ with weights calculated by (4.7), see Section 4.4. The weight of the edges is represented by the line width and illustrates how often these edges occurred in the skeletons $\mathcal{G}_1^s, \dots, \mathcal{G}_4^s$. We observe strong dependence especially among and between the national stocks indices and the Euro Stoxx 50 index itself. This confirms our expectation that our approach captures the most important relationships in the data.

A.2 Numerical Results of Fitted Models

DAG				R-vine representation of DAG							
Max. par-ents	No. par.	Log-lik.	GIC	No. par	No. ni-pc	No. G-pc	No. non-G-pc	Log-lik.	GIC	k'	time (sec.)
1	155	-47138	95784	206	51	0	51	-45880	93879	1	118
2	204	-45365	92824	498	249	17	232	-42818	91608	47	190
3	250	-44732	92128	696	401	49	352	-41673	92140	48	210
4	280	-44448	91942	723	428	59	369	-41497	92183	49	233
5	309	-44224	91868	742	435	54	381	-41451	92369	51	249
6	330	-44104	91903	743	439	54	385	-41435	92352	48	260
7	341	-44045	91930	747	437	53	384	-41401	92342	48	259
8	345	-44026	91944	741	434	58	376	-41415	92283	51	263
9	349	-44007	91958	736	432	59	373	-41411	92201	49	265
10	353	-43990	91978	733	427	61	366	-41423	92180	49	263

Table A.1: Numerical results for DAG and DAG representations. Calculations based on z -scale, abbreviations *ni-pc* for *non independence pair copula*, *G-pc* for *Gaussian pair copula*.

A.2 Numerical Results of Fitted Models

trunc. level	No. par	No. ni-pc	No. G-pc	No. non- G-pc	Log- lik.	GIC	time (sec.)
1	206	51	0	51	-45808	93734	600
2	283	99	2	97	-44445	91973	613
3	327	126	5	121	-44121	91896	653
4	364	152	8	144	-43899	91942	698
5	389	170	13	157	-43750	91978	723
6	403	182	14	168	-43628	91923	716
7	427	197	14	183	-43423	91839	715
8	456	217	17	200	-43194	91778	705
9	476	231	18	213	-43102	91869	708
10	495	245	19	226	-43037	92002	704
11	512	259	20	239	-42950	92067	705
12	525	269	23	246	-42884	92115	708
13	537	279	25	254	-42782	92081	702
14	551	290	25	265	-42655	92024	704
15	563	299	27	272	-42568	92020	713
16	583	313	28	285	-42421	92010	706
17	597	321	30	291	-42305	91977	704
18	616	335	31	304	-42172	91981	708
19	629	344	32	312	-42127	92078	708
20	641	353	34	319	-42023	92043	704
21	650	361	34	327	-41970	92068	710
22	656	365	35	330	-41933	92080	705
23	662	370	37	333	-41910	92121	715
24	667	375	39	336	-41889	92151	710
25	675	381	42	339	-41843	92175	709
26	682	386	44	342	-41812	92215	706
27	688	391	46	345	-41734	92146	711
28	691	393	46	347	-41727	92176	712
29	693	395	46	349	-41714	92179	709
30	697	397	46	351	-41699	92206	719
31	702	400	47	353	-41682	92246	718
32	705	403	47	356	-41671	92267	715
33	708	405	47	358	-41660	92289	716
34	713	408	47	361	-41649	92339	715
35	714	409	48	361	-41645	92348	725
36	716	411	49	362	-41636	92359	730
37	719	413	50	363	-41556	92242	731
38	727	418	50	368	-41503	92252	723
39	732	422	50	372	-41459	92239	732

40	734	423	50	373	-41427	92205	724
41	738	425	50	375	-41409	92227	724
42	742	428	50	378	-41383	92232	721
43	746	432	52	380	-41368	92262	723
44	748	434	52	382	-41359	92273	730
45	748	434	52	382	-41359	92273	731
46	748	434	52	382	-41359	92273	744
47	749	435	52	383	-41356	92283	738
48	749	435	52	383	-41356	92283	732
49	749	435	52	383	-41356	92283	730
50	752	437	52	385	-41336	92286	731
51	752	437	52	385	-41336	92286	739

Table A.2: Numerical results for Dißmann algorithm. Calculations based on *z-scale*, abbreviations *ni-pc* for *non independence pair copula*, *G-pc* for *Gaussian pair copula*.

A.3 Algorithms to Chapter 4

input : DAG $\mathcal{G} = (V = (v_1, \dots, v_d), E)$ with topological ordering $v_i >_{\mathcal{G}} v_j$, truncation level $k = \max_{v \in V} |\text{pa}(v)| = 1$.

output: R-vine tree sequence T_1, \dots, T_{d-1} given by a R-vine matrix M and an independence matrix $F \in \{0, 1\}^{d \times d}$, indicating which pair copula families can be set to the independence copula.

```

1 set  $M = \text{diag}(d, \dots, 1)$ ;
2 set  $F = (0)^{d \times d}$ ;
3 for  $i = d - 1$  to 1 do
4   | set  $M_{d,i} = \text{pa}(M_{i,i})$ ;
5   | set  $F_{d,i} = 1$ ;
6 end
7 complete  $M$  according to the proximity condition;
8 return  $M, F$ ;
```

Algorithm 1: RepresentMarkovTreeRVine: Construction of a R-vine tree matrix M and independence matrix F obtained from a DAG $\mathcal{G} = (V, E)$ with at most one parent.

```

input : DAGs  $\mathcal{G}_i, i = 1, \dots, k$  with at most  $i$  parents, weighting function  $g(i)$ .
output: R-vine matrix  $M$  and independence matrix  $F$  indicating which pair
        copulas are the independence copula, truncation level  $k'$ .

1 calculate skeletons  $\mathcal{G}_i^s$  of DAGs  $\mathcal{G}_i$  for  $i = 1, \dots, k$ ;
2 create  $\mathcal{H} = (V, E_1^{\mathcal{H}}) := \bigcup_{i=1}^k \mathcal{G}_i^s$ ;
3 set weights  $\mu_1(v, w) = \sum_{i=1}^k g(i) \mathbb{1}_{(v,w) \in E_i^s}(v, w)$  for each edge  $(v, w) \in E_1^{\mathcal{H}}$ ;
4 calculate maximum spanning tree  $T_1 = (V, E_1^T)$  on  $\mathcal{H}$ ;
5 for  $i = 2$  to  $d - 1$  do
6     create full undirected graph  $\mathcal{H}_i = (V_i = E_{i-1}, E_i^{\mathcal{H}})$ ;
7     delete edges not allowed by the proximity condition;
8     for  $e \in E_i^{\mathcal{H}}$  do
9         if  $\mu_1(j(e), \ell(e)) \neq 0$  then
10            assign DAG weights  $\mu_i(e) = \mu_1(j(e), \ell(e))$ ;
11        else
12            if  $j(e) \perp\!\!\!\perp \ell(e) \mid D(e)$  according to  $d$ -separation in  $\mathcal{G}_k$  then
13                assign independence weight  $\mu_i(e) = \mu_0$ ;
14            else
15                end
16            end
17        end
18    calculate maximum spanning tree  $T_i = (V_i, E_i^T)$  on  $\mathcal{H}_i$ ;
19 end
20 create R-vine matrix  $M$  from R-vine trees  $T_1, \dots, T_{d-1}$ ;

```

Algorithm 2: RepresentDAGRVine: Calculation of an R-vine tree matrix M obtained from DAGs $\mathcal{G}_1, \dots, \mathcal{G}_k$.

```

input : R-vine matrix  $M$  and DAG  $\mathcal{G}_k$ .
output: Independence matrix  $F$  indicating which pair copulas are the
         independence copula and truncation level  $k'$ .
1 set  $F = (0)^{d \times d}$ ;
2 for  $i = d$  to 2 do
3   for  $j = i - 1$  to 1 do
4     if  $M_{i,j} \perp\!\!\!\perp M_{j,j} \mid M_{i+1,j}, \dots, M_{d,j}$  according to  $d$ -separation in  $\mathcal{G}_k$  then
5       set  $F_{i,j} = 0$ ;
6     else
7       set  $F_{i,j} = 1$ ;
8     end
9   end
10 end
11 set  $k'$  such that in the R-vine trees  $T_{k'+1}, \dots, T_{d-1}$  only the independence copula
    occurs;

```

Algorithm 3: RepresentDAGRVine: Calculation of an independence matrix F and truncation level k' for an R-vine matrix obtained from Algorithm 3.

```

input : DAG  $\mathcal{G} = (V = (v_1, \dots, v_d), E)$ , empirical covariance matrix  $\hat{\Sigma}$ .
output: Model induced covariance matrix  $\hat{\Sigma}_{\mathcal{G}}$ .
1 set  $A = \{0\}^{d \times d}$ ;
2 for  $i = 1$  to  $d$  do
3   for  $j = 1$  to  $d$  do
4     if  $j \in \text{pa}(i)$  then
5        $\hat{A}_{ij} = - \left( \hat{\Sigma}_{i, \text{pa}(i)} \left( \hat{\Sigma}_{\text{pa}(i), \text{pa}(i)} \right)^{-1} \right)_j$ 
6     else
7       if  $i = j$  then
8          $\hat{A}_{ij} = 1$ 
9       else
10         $\hat{A}_{ij} = 0$ 
11       end
12     end
13   end
14 end
15 define  $\widehat{\text{Cov}}(\epsilon) = \text{diag} \left( \hat{\Sigma}_{1|\text{pa}(1)}, \dots, \hat{\Sigma}_{d|\text{pa}(d)} \right)$ ;
16 solve  $\hat{\Sigma}_{\mathcal{G}} = \hat{A}^{-1} \widehat{\text{Cov}}(\epsilon) \hat{A}$ ;
17 return  $\hat{\Sigma}_{\mathcal{G}}$ ;

```

Algorithm 4: DAGtoCovMatrix: Covariance matrix estimation based on a DAG model.

Appendix B

Algorithms to Chapter 5

```

input : Data  $X \in \mathbb{R}^{n \times d}$ , ordering function  $\eta_L$ .
output: R-vine structure  $\mathbf{M}$  and regularization path matrix  $\mathbf{A}$ .
1 order Columns of data  $X \in \mathbb{R}^{n \times d}$  according to ordering function  $\eta_L$ , i. e.
    $\tilde{X}_{\cdot,j} = X_{\cdot,\eta_L(j)}$  for  $j = 1, \dots, d$ ;
2 create  $\mathbf{M}$  with  $\text{diag}(\mathbf{M}) = (\eta_L(d), \dots, \eta_L(1))$ ;
3 create  $\mathbf{A} = \{0\} \in \mathbb{R}^{d \times d}$ ;
4 set  $m_{d,d-1} = m_{d,d}$ ;
5 for  $j = 3$  to  $d$  do
6     solve  $\min_{\varphi \in \mathbb{R}^{j-1}} \left( \frac{1}{2n} \sum_{i=1}^n \left( x_{i,j} - \sum_{\ell=1}^{j-1} \varphi_{j,\ell} x_{i,\ell} \right)^2 + \lambda_j \sum_{\ell=1}^{j-1} |\varphi_{j,\ell}| \right)$ ;
7     store  $\Lambda(\lambda, j) = \{\ell : \hat{\varphi}_{j,\ell}^\lambda \neq 0 \text{ in } \hat{\varphi}_j^\lambda\}$ , with  $k(\lambda, j) = |\Lambda(\lambda, j)|$ ;
8     set  $m_{d,j} = \Lambda(0, j)_1$ ;
9     calculate  $\tilde{\lambda} > 0$  such that  $k(\tilde{\lambda}, j) - k(\tilde{\lambda} - \epsilon, j) = 1$  and  $k(\tilde{\lambda}, j) = 1$  where
         $\epsilon > 0$ ; // search  $\tilde{\lambda}$  where  $\varphi_{j,\ell}$  vanishes
10    set  $\Lambda_{d,j} = \tilde{\lambda}$ ;
11 end
12 for  $k = 2$  to  $d - 1$  do
13     for  $j = k + 1$  to  $d$  do
14         if  $\Lambda(0, j)_k$  adheres to the proximity condition then
15             set  $m_{d-k+1,d-j+1} = \Lambda(0, j)_k$ ;
16             calculate  $\tilde{\lambda} > 0$  such that  $k(\tilde{\lambda}, j) - k(\tilde{\lambda} - \epsilon, j) = 1$  and  $k(\tilde{\lambda}, j) = k$ 
                where  $\epsilon > 0$ ; // search  $\tilde{\lambda}$  where  $\varphi_{j,\ell}$  vanishes
17             set  $\Lambda_{d-k+1,d-j+1} = \tilde{\lambda}$ ;
18         else
19             define the set of potential regressors  $\mathcal{H}(k, j) = \{m_{j+1,j+1}, \dots, m_{d,d}\}$ ;
20             define the whitelist  $\mathcal{W}(k, j) = \{m_{d,j}, \dots, m_{k+1,j}\}$ ;
21             define the blacklist
                 $\mathcal{B}(k, j) = \{\ell \in \mathcal{H}(k, j) \setminus \mathcal{W}(k, j) : \ell \text{ does not satisfy the pc.}\}$ ;
22             solve  $\min_{\varphi \in \mathbb{R}^{j-1-|\mathcal{B}(k,j)|}} \left( \frac{1}{2n} \sum_{i=1}^n \left( X_{i,j} - \sum_{\ell \in \mathcal{H}(k,j) \setminus \mathcal{B}(k,j)} \varphi_{j,\ell} X_{i,\ell} \right)^2 + \right.$ 
                 $\left. \sum_{\ell \in (\mathcal{H}(k,j) \setminus \mathcal{B}(k,j)) \setminus \mathcal{W}(k,j)} \lambda_{j,\ell} |\varphi_{j,\ell}| \right)$ ;
23             set  $m_{d-k+1,d-j+1} = \Lambda(0, j)_k$ ;
24             calculate  $\tilde{\lambda} > 0$  such that  $k(\tilde{\lambda}, j) - k(\tilde{\lambda} - \epsilon, j) = 1$  and  $k(\tilde{\lambda}, j) = k$ 
                where  $\epsilon > 0$ ; // search  $\tilde{\lambda}$  where  $\varphi_{j,\ell}$  vanishes
25             set  $\Lambda_{d-k+1,d-j+1} = \tilde{\lambda}$ ;
26         end
27     end
28 end
29 return R-vine matrix  $\mathbf{M}$  and regularization path matrix  $\mathbf{A}$ ;
    
```

Algorithm 5: RVineLassoSelect: Calculation of an R-vine structure \mathbf{M} and regularization path matrix \mathbf{A}

input : Data $X \in \mathbb{R}^{n \times d}$, $\lambda \in \mathbb{R}^d$.
output: Ordering function η .

- 1 **define** $\mathbf{B} = \{0\} \in \mathbb{R}^{d \times d}$;
- 2 **for** $j = 1$ to d **do**
- 3 **solve** $\min_{\varphi_j \in \mathbb{R}^{d-1}} \left(\frac{1}{2n} \sum_{i=1}^n \left(x_{i,j} - \sum_{\ell=1, \ell \neq j}^d \varphi_{j,\ell} x_{i,\ell} \right)^2 + \lambda_\ell \sum_{\ell=1, \ell \neq j}^d |\varphi_{j,\ell}| \right)$;
- 4 **set** $B_{j,j} = (\varphi_{j,1}, \dots, \varphi_{j,j-1}, 0, \varphi_{j,j+1}, \dots, \varphi_{j,d})$;
- 5 **end**
- 6 **calculate** ordering function η_L such that

$$\sum_{\ell=1}^d \mathbb{1}_{\{\varphi_{\eta_L(1),\ell} \neq 0\}} \geq \dots \geq \sum_{\ell=1}^d \mathbb{1}_{\{\varphi_{\eta_L(d),\ell} \neq 0\}}$$
- 7 **return** ordering function η_L ;

Algorithm 6: CalcOrdering: Calculation of an ordering function η

```

input : Data  $U \in [0, 1]^{n \times d}$ , R-vine matrix  $\mathbf{M}$ , regularization path matrix  $\mathbf{A}$ ,
        single thresholding parameter  $\lambda_T > 0$  or adaptive thresholding parameter
         $\mu \in [0, 1]$ .
output: Sparse R-vine with structure  $\mathbf{M}$ , family matrix  $\mathbf{\Gamma}$  and parameter matrix
         $\mathbf{\Theta}$ .

1 if Single thresholding mode then
2   for  $k = 2$  to  $d$  do
3     for  $j = 1$  to  $k - 1$  do
4       if  $\Lambda_{k,j} \geq \lambda_T$  then
5          $\gamma_{k,j} = 1$ ;
6       else
7         do nothing;
8       end
9     end
10  end
11 else
12  solve  $[\mu \cdot \binom{d}{2}] = \sum_{j=1, \dots, d-1, k=1, \dots, d-j}^d \mathbb{1}_{\{\lambda_{k,j} \geq \lambda_\mu\}}$ ; for  $k = 2$  to  $d$  do
13    for  $j = 1$  to  $k - 1$  do
14      if  $\Lambda_{k,j} \geq \lambda_\mu$  then
15         $\gamma_{k,j} = 1$ ;
16      else
17        do nothing;
18      end
19    end
20  end
21  for  $k = 1$  to  $d - 1$  do
22    for  $j = 1$  to  $d - k$  do
23      if  $\gamma_{k,j} = 1$  then
24        estimate pair copula using maximum likelihood or inversion of
        Kendall's  $\tau$ ;
25      else
26        do nothing;
27      end
28    end
29  end
30 end
31 return R-vine with structure  $\mathbf{M}$ , family matrix  $\mathbf{\Gamma}$  and parameter matrix  $\mathbf{\Theta}$ ;

```

Algorithm 7: RVineThresholdSelect: Single or adaptive thresholding of an R-vine and estimation of pair copulas

Appendix C

Algorithm `RVineClusterSelect`

```

input : Data  $X \in \mathbb{R}^{n \times d}$ ,  $d_T \leq d$ ,  $k_F < d$ .
output: R-vine in  $d$  dimensions.

1 calculate  $(\mathcal{G}_1 = (\bigcup_{i=1}^{p_1} V_i^1, \bigcup_{i=1}^{p_1} E_i^1), \dots, \mathcal{G}_J = (\bigcup_{i=1}^{p_J} V_i^J, \bigcup_{i=1}^{p_J} E_i^J))$  and
   assume  $V_i^j$  are ordered such that  $|V_i^j| \leq |V_{i-1}^j|$  for  $i = 2, \dots, p_j$  and  $j = 1, \dots, J$ ;
2 select  $\mathcal{G}_T$  such that  $T = \arg \max_{j=1, \dots, J} \delta_j \leq d_T$  with  $\delta_j = \max_{i=1, \dots, p_i} |V_i^j|$ ;
3 for  $i = 1$  to  $p_T$  do
4     set  $\nu_i = |V_i^T|$ ,  $\mathcal{H} = (\mathcal{W} = \mathcal{V}_i^T, \mathcal{F} = E_i^T)$ ,  $\mathcal{H}_0 = (\mathcal{W}, \mathcal{W} \times \mathcal{W})$ ;
5     if  $\nu_i \geq 3$  then
6         for  $(j, \ell) \in \mathcal{W} \times \mathcal{W}$  do
7             if  $(j, \ell) \notin \mathcal{F}$  then
8                 set  $c_{j, \ell} = 1$ ;
9             else
10                estimate pair-copula  $c_{j, \ell}$ ;
11            end
12        end
13        calculate weights  $\mu_{j, \ell} = |\mu(c_{j, \ell})|$ ; // e. g. Log-Lik., AIC, BIC
14        calculate optimal spanning tree  $T_1 = (\mathcal{W}, E_1)$  on  $\mathcal{H}_0$  w.r.t.  $\mu$ ;
15        for  $k = 2$  to  $\nu_i - 1$  do
16            set  $T_k = (E_{k-1}, E_k)$  with  $E_k$  admissible edges by proximity condition;
17            for  $(j, \ell; D) \in E_k$  do
18                if  $j \perp \ell \mid D [\mathcal{H}]$  then
19                    set  $c_{j, \ell; D} = 1$ ;
20                else
21                    estimate pair-copula  $c_{j, \ell; D}$ ;
22                end
23            end
24            calculate weights  $\mu_{j, \ell; D} = |\mu(c_{j, \ell; D})|$ ;
25            calculate optimal spanning tree  $T_k = (V_k, E_k)$  w.r.t.  $\mu$ ;
26        end
27    else
28        if  $\nu_i = 2$  then
29            estimate pair-copula  $c_{j, \ell}$  for pair  $(j, \ell) \in E_i^T$ ;
30        end
31    end
32 end
33 combine sub-R-vines and isolated nodes into R-vine matrix  $M \in \{0, \dots, d\}^{d \times d}$ ;
34 for  $k = 1$  to  $k_F$  do
35     estimate pair-copulas between sub-R-vines in tree  $k$  of in R-vine matrix  $M$ ;
36 end
    
```

Algorithm 8: *RVineClusterSelect*: Selection of an R-vine in d dimensions.