Technical University of Munich

Department of Mathematics

Vine Copula Based Synthetic Data Generation for Classification

Master's Thesis

by

Elisabeth Griesbauer

Supervisor:	Prof. Claudia Czado, Ph.D.
	Prof. Arnoldo Frigessi, Ph.D. (external)
	Prof. Ingrid Hobæk Haff, Ph.D. (external)
Submission Date:	July 11, 2022

I hereby declare that this thesis is my own work and that no other sources have been used except those clearly indicated and referenced.

Oslo, July 11, 2022

Acknowledgments

I want to sincerely thank Prof. Claudia Czado, Prof. Arnoldo Frigessi and Prof. Ingrid Hobæk Haff for their great supervision and support and for giving me the opportunity to come to Oslo for my thesis.
I also want to thank my colleagues and friends who accompanied me the past years of study and who made them a wonderful time: thank you for the shared laughter and the mutual encouragement.
Finally I want to express my deep gratitude to my partner and my family: Your unconditional love and support are my foundation and have brought me to where I am today.

Abstract

Generating synthetic data enables research on data, which due to privacy protection must not be published, such as patient data. Synthetic data can also augment existing (real) data, if they are limited. Commonly used methods for synthetic data generation, such as generative adversarial networks (GANs) (Goodfellow et al. (2014)) or variational auto encoders (VAEs) (Kingma and Welling (2013)), are based on neural networks. This makes their training intensive and, especially in the case of GANs, difficult (Arjovsky and Bottou (2017)). We use vine copulas as a synthetic data generator and focus on a setting, where the analysis to be done on true and synthetic data is classification. The synthetic data produced should allow the user to estimate a classification rule, which is similar to the classification rule, that would be estimated on the true data. We compare three different vine estimation methods in a simulation study, as well as on real-data applications in astronomy and cancer genomics. We find, the best vine estimation method depends on the properties of the true data, but the differences are overall not very large.

Zusammenfassung

Synthetische Daten ermöglichen Forschung dort, wo aus Datenschutzgründen Daten nicht veröffentlicht werden können, wie beispielsweise Patientendaten aus medizinischen Studien. Ebenso können synthetische Daten einen bestehenden Datensatz vergrößern, wenn dieser wenige Beobachtungen beinhaltet. Gängige Methoden zur Erzeugung synthetischer Daten, wie Generative Adversarial Networks (GANs) (Goodfellow et al. (2014)) oder Variational Auto Encoders (VAEs) (Kingma and Welling (2013)), basieren auf neuronalen Netzen. Das macht ihr Training intensiv und insbesondere bei GANs schwierig (Arjovsky and Bottou (2017)). Wir verwenden Vine Copulas als Erzeugungsmethode von synthetischen Daten und konzentrieren uns auf Klassifikation der echten und synthetischen Daten als Anwendung. Dabei sollen die erzeugten synthetischen Daten es dem Benutzer ermöglichen, eine Klassifizierungsregel zu schätzen, die der auf den wahren Daten geschätzten Klassifizierungsregel ähnlich ist. Wir vergleichen drei verschiedene Methoden zur Schätzung der Vine Copula in einer Simulationsstudie sowie an realen Daten aus der Astronomie und Krebsgenomik. Dabei hängt die beste Vine-Schätzungsmethode von den Eigenschaften der wahren Daten ab, wobei die Unterschiede eher gering ausfallen.

Notation

Sets

[d]	set of natural numbers $\{1, 2, ..., d\}$ with $d \in \mathbb{N}$
$\|A\|$	cardinality of set A
$\mathbb{N}$	the set of natural numbers $\{1, 2, 3, ...\}$
$\emptyset$	the empty set

Vectors and Matrices

Vectors are denoted by small letters, matrices by capital letters.

$\boldsymbol{x} = \begin{pmatrix} \vert \\ \boldsymbol{x} \\ \vert \end{pmatrix}$	d-dimensional vector $(x_1, x_2, ..., x_d) \in \mathbb{R}^d$, visual indication of the vector
$\boldsymbol{x}_A$	subvector of $\boldsymbol{x} \in \mathbb{R}^d$ with components of indices $i \in A \subset [d]$
$\boldsymbol{x}_{-j}$	subvector of $\boldsymbol{x} \in \mathbb{R}^d$ without the component of index $j \in [d]$
M	matrix in $\mathbb{R}^{m \times n}$, denoted a by capital, non-bold letter
$(m_{ij})_{i \in [m], j \in [n]}$	matrix $M \in \mathbb{R}^{m \times n}$ with ijth component $m_{ij} \in \mathbb{R}$
M_A	submatrix of matrix $M \in \mathbb{R}^{m \times n}$ with rows and columns $i, j \in A \subset [m] \times [n]$
M_{-i}	submatrix of matrix $M \in \mathbb{R}^{m \times n}$ with row $i \in [m]$ removed
M^{-j}	submatrix of matrix $M \in \mathbb{R}^{m \times n}$ with column $j \in [n]$ removed
m_j	column vector of column $j \in [n]$ for a given matrix $M \in \mathbb{R}^{m \times n}$ with $m_j \in \mathbb{R}^n$
m_i^T	row vector of row $i \in [m]$ for a given matrix $M \in \mathbb{R}^{m \times n}$ with $m_i^T \in \mathbb{R}^m$

Probabilities and Random Variables

X	random variable with $X : \Omega \to \mathbb{R}$
$\boldsymbol{X}$	d-dimensional random vector $(X_1, X_2, ..., X_d) : \Omega \to \mathbb{R}^d$

Contents

1 Introduction

Making data sets publicly available enables research. However, this is in many cases not possible in order to protect confidential data. For example think of patient data in medical studies. Synthetic data generated from the original true data can be a solution to this problem. Instead of analysing the relation between features $\boldsymbol{X}_{true}$ and response Y_{true} on the true data, researches can estimate the relation between synthetic features $\boldsymbol{X}_{synth}$ and synthetic response Y_{synth} on the synthetic data. Therefore the synthetic data generating method must reproduce the dependencies between true $\boldsymbol{X}_{true}$ and true Y_{true} in the synthetic data. If this is given, the relation estimated on the synthetic data will be similar to the one estimated on the true data. If at the same time the synthetic data does not disclose observations from the true data, it will not be necessary to publish the true confidential data. In other situations we encounter the problem, that data are available but limited. Training data intense methods on them, such as neural nets, becomes difficult. Here synthetic data can augment the existing data.

A commonly used method to generate synthetic data are generative adversarial networks (GANs) by Goodfellow et al. (2014). They are based on the idea of pitting a generative model against a discriminative model in training. GANs enjoy popularity in high dimensional data applications such as image data, see for example Frid-Adar et al. (2018) and Han et al. (2018), as well as in synthetic data generation for tabular data, see Xu et al. (2019) and Choi et al. (2017). Furthermore, variational auto encoders (VAEs) by Kingma and Welling (2013) are used for the generation of synthetic data. Wan et al. (2017) for example apply them to generate synthetic data for imbalanced learning in high dimensions. Both GANs and VAEs are based on neural nets, which is why their training is intensive and especially for GANs can be difficult, Arjovsky and Bottou (2017).

Instead of neural networks, other generative models can be used as synthetic data generators. Vine copulas introduced by Aas et al. (2009) and based on works by Sklar (1959), Bedford and Cooke (2001), Bedford and Cooke (2002) are flexible generative models constructed from bivariate building blocks. Tagasovska et al. (2019) generate synthetic data by combining them with a pre-trained autoencoder for lower dimensional representation to overcome intense training. They restrict themselves to non-parametric vines with truncation after 5 trees. Similarly, Kamthe et al. (2021) exploit the features of vine copulas to obtain a synthetic data generator, which can deal with mixed data and is interpretable. They use normalizing flows to learn the multivariate copula density and marginals.

In this thesis, we use vine copulas as a synthetic data generator and focus on the case when the task is classification. We assess the quality of the synthetic data by its capability to let a classifier learn a similar rule from them, that the same classifier would estimate on the true data. Through our analysis, we provide a benchmark of how well vines are generally suited to generate synthetic data for classification. In addition, we compare three different vine estimation methods: vine estimation with non-parametric, parametric and mixed pair copula estimation. The comparison is done with respect to classification and variable selection performance of the classifier trained on the synthetic data obtained by each method. We find that the best vine estimation method depends on properties of the true data and whether the intended goal is best classification performance or best variable selection performance. Overall, our approach works well when comparing the results of the synthetic classifier to the one trained on the true data. The differences in performance between the vine estimation methods are not very large and increase with increasing difficulty of the classification task.

The remainder of this thesis is organized as follows: In Sections 2 and 3 we introduce the most important theory on vine copulas and classification. Section 4 combines the previously reviewed concepts to the methodology of vine copula based synthetic data generation for classification. In Section 5 we apply the latter in a simulation study, Sections 6 and 7 we assess vines as synthetic data generator on real data in astronomy and cancer genomics and compare the three vine estimation methods. Finally Section 8 concludes this thesis and points to further work.

2 An Introduction to Vine Copulas

2.1 An Intuition for Non-Statisticians

One of the main goals of statistics is to understand the joint behavior of various (random) quantities, called random variables. The mood of a person for example seems to be varying so let us view it as a random variable. When I am visiting my family I can observe that my father's mood is somehow linked to the mood of my mother. Her being happy has also an effect on how my two siblings are feeling. It is not only interesting to learn the marginal distribution of each random variable, such as if it is more likely that my brother is in a happy rather than a grumpy mood. It is especially interesting to know how the moods of the different family members influence each other. So the goal is to find out the dependence structure among the random variables. The joint distribution of the random variables, the moods, contains all the information, i.e. all dependencies as well as all marginal distributions, and can be modeled with the help of data.
However, the information I receive about mutual dependence when observing my father's and my sister's mood together is always influenced by both their marginal distributions. We will see later that the joint density - which is, if it exists, derived from the joint distribution - can be split into the copula density and marginal densities. The copula captures the mere dependence between all the random variables and therefore enables us to understand their joint behavior.

2.2 Copulas

For introducing the concept of copulas first their formal definition is given. After stating Sklar's central theorem a method of constructing multivariate probability distributions with the help of bivariate copulas is given. Further we cover different dependence measures and families of copulas. In Czado (2019) for example the reader can find more details.

Definition 2.2.1. Let $d \in \mathbb{N}$. The function $C : [0,1]^d \to [0,1]^d$ is a *d-dimensional copula* if it is a d-dimensional cumulative distribution function with uniform marginal distributions $U[0,1]$.

So for the random vector $(U_1, \dots U_d)$ taking on values $(u_1, \dots, u_d) \in [0,1]^d$ it is:

$$C(u_1, \dots, u_d) = P(U_1 \leq u_1, \dots, U_d \leq u_d) \ . \tag{2.1}$$

As for all distributions, the copula distribution has a density if it is absolutely continuous. For $(u_1, \dots, u_d) \in [0,1]^d$ the copula density is obtained by:

$$c(u_1, ..., u_d) = \frac{\partial^d \mathbb{C}(u_1, \dots, u_d)}{\partial u_1 \dots \partial u_d} \ . \tag{2.2}$$

Before stating Sklar's Theorem we remind ourselves of the probability integral transform.

Definition 2.2.2 (Probability integral transform, PIT)**.** Let X be a continuous random variable with distribution function F. Then $u := F(x)$ is defined as the *probability integral transform* of X at $x \in \mathbb{R}$.

Remark 2.2.1. The distribution of $U := F(X)$ can easily be obtained:

$$P(U \leq u) = P(F(X) \leq u) = P(X \leq F^{-1}(u)) = F(F^{-1}(u)) = u \ , \tag{2.3}$$

using that $X \sim F$. As this holds for any $u \in [0,1]$ we find that $U \sim Unif[0,1]$.

Theorem 2.2.1 (Sklar's Theorem)**.** *Let $\boldsymbol{X}$ be a d-dimensional random vector with distribution function F and marginal distributions $F_1, \dots F_d$. Then F can be expressed as:*

$$F(x_1, \dots, x_d) = C(F_1(x_1), \dots, F_d(x_d)) \ , \quad (x_1, \dots, x_d) \in \mathbb{R}^d \ . \tag{2.4}$$

where C is a copula. If F is absolutely continuous, the copula C is unique. We then say that the copula C is corresponding to the distribution F. In the case of absolute continuity all densities exist and we can express the joint density f of $\boldsymbol{X}$ as:

$$f(x_1, \dots, x_d) = c(F_1(x_1), \dots, F_d(x_d)) \cdot f_1(x_1) \cdot ... \cdot f_d(x_d) \ . \tag{2.5}$$

Conversely, let C be the d-dimensional copula corresponding to the joint distribution function F of $\boldsymbol{X}$ with marginal distributions $F_1, \dots F_d$. Then we can express C as:

$$C(u_1, \dots, u_d) = F(F_1^{-1}(u_1), \dots, F_d^{-1}(u_d)) \tag{2.6}$$

with copula density:

$$c(u_1, \dots, u_d) = \frac{f(F_1^{-1}(u_1), \dots, F_d^{-1}(u_d))}{f_1(F_1^{-1}(u_1)) \cdot ... \cdot f_d(F_d^{-1}(u_d))} \ . \tag{2.7}$$

Sklar's Theorem, Sklar (1959) provides us with the link between the copula on the d-dimensional hypercube and the probability distribution of the random vector $\boldsymbol{X}$. Some remarks on the theorem:

- Given any copula C and some marginal distributions $F_1, \dots F_d$, the expression in (2.4) allows us to construct an arbitrary joint distribution function F. The same is true for densities.
- The equation in (2.5) illustrates how the joint density f of a random vector $\boldsymbol{X}$ can be split into the joint copula density, which captures the dependence structure of $\boldsymbol{X}$, and the marginal densities $f_1, \dots f_d$.
- The copula C is by definition again a distribution function with uniform density. Therefore Sklar's Theorem can be applied to the copula itself. This might not be of interest now, as for $u \in [0,1]$ we have that $C(u) = u$ and $c(u) = 1$ trivially. However having this in mind will be of use later.

The following *bounds* allow us to later describe certain properties of copulas:

Theorem 2.2.2 (Fréchet-Hoeffding bounds)**.** *Let C be a d-dimensional copula. Then for any $\boldsymbol{u} \in [0,1]^d$:*

$$W^d(\boldsymbol{u}) \leq C(\boldsymbol{u}) \leq M^d(\boldsymbol{u}) \ ,$$

with:

$$\begin{aligned} W^d(\boldsymbol{u}) &:= \max\{u_1 + \cdots + u_d - d + 1, 0\} \ , \\ M^d(\boldsymbol{u}) &:= \min\{u_1, \dots, u_d\} \ . \end{aligned}$$

A proof of this theorem can for example be found in Nelsen (2007).

Remark 2.2.2. It can be shown, that the upper bound $M^d(\boldsymbol{u})$ is a copula. The lower bound $W^d(\boldsymbol{u})$ is only a copula for $d = 2$. In the following we use the notation:

$$\begin{aligned} W(u_1, u_d) &:= W^2(u_1, u_2) = \max\{u_1 + u_2 - 1, 0\} \ , \\ M(u_1, u_2) &:= M^2(u_1, u_2) = \min\{u_1, u_2\} \ . \end{aligned}$$

Following Nelsen (2007), we say that for two uniformly distributed random variables $U_1, U_2 \in [0, 1]$ "U_1 is an almost surely increasing function of U_2" if and only if their copula is $C(U_1, U_2) = M(U_1, U_2)$. Then $P(U_1 = U_2) = 1$ and we call this *complete positive dependence*. Similarly, we say that "U_1 is an almost surely decreasing function of U_2" if and only if their copula is $C(U_1, U_2) = W(U_1, U_2)$. Then $P(U_1 + U_2 = 1) = 1$ and we call this *complete negative dependence*.

Additionally, it is useful to look at the following 2-dimensional case:

Remark 2.2.3. For two independent random variables the corresponding bivariate copula distribution function is by (2.5) of Sklar's Theorem:

$$C(u_1, u_2) = F(F_1^{-1}(u_1), F_2^{-1}(u_2)) \overset{ind.}{=} F_1(F_1^{-1}(u_1)) \cdot F_2(F_2^{-1}(u_2)) = u_1 \cdot u_2 \ .$$

It is known as the *independence copula* and denoted by $\Pi(u_1, u_2)$.

In the same manner that we have different classes of probability distributions with their respective explicit formulation, we have different classes of copula distributions with their corresponding formulation. The inverse Sklar's Theorem 2.2.1 gives the construction of one class of copulas, namely the *elliptical copulas*. The two most prominent members of the class are the Gauss copula and the Student's t copula. We define the bivariate cases here. The multivariate cases are constructed in the same way.

Definition 2.2.3 (bivariate Gauss copula)**.** Let $\Phi_2(\cdot, \cdot; \rho)$ be the 2-dimensional standard normal distribution with mean vector $\boldsymbol{\mu} = 0$ and correlation parameter $\rho \in (0, 1)$, and let $\Phi^{-1}(\cdot)$ be the quantile function of the univariate standard normal distribution. Then by Sklar's Theorem 2.2.1 we obtain the *bivariate Gauss copula* by:

$$C(u_1, u_2; \rho) = \Phi_2(\Phi^{-1}(u_1), \Phi^{-1}(u_2); \rho) \ . \tag{2.8}$$

Definition 2.2.4 (bivariate Student's t copula)**.** In the same manner, the *bivariate Student's t copula* is given by:

$$C(u_1, u_2; \nu, \rho) = \int_0^{u_1} \int_0^{u_2} \frac{t(T_\nu^{-1}(v_1), T_\nu^{-1}(v_2); \nu, \rho)}{t_\nu(T_\nu^{-1}(v_1)) t_\nu(T_\nu^{-1}(v_2))} dv_1 dv_2 \ , \tag{2.9}$$

with $t(\cdot, \cdot; \nu, \rho)$ the bivariate Student's t density with $\nu > 0$ degrees of freedom of two random variables with correlation $\rho \in (-1, 1)$.

Remark 2.2.4. For the bivariate Gauss copula and the bivariate Student's t copula it holds, that:

$$C(u_1, u_2; \rho) = \begin{cases} \Pi(u_1, u_2), & \text{for} \quad \delta = 0 \ , \\ W(u_1, u_2), & \text{for} \quad \delta \to -1 \ , \\ M(u_1, u_2), & \text{for} \quad \delta \to 1 \ . \end{cases}$$

Verbally expressed, they show complete positive dependence for $\delta \to -1$, independence $\delta = 0$ and complete negative dependence for $\delta \to 1$.

An overview of commonly used distribution functions and their notation can be found in the Appendix 9.

Bivariate Copula Families

Beside the *elliptical copulas* such as the *bivariate Gauss copula* and the *bivariate Student's t copula*, which we have seen are constructed from known multivariate distributions, we want to introduce further bivariate copula families. They can all be extended to any dimension $d \in \mathbb{N}$. Let u_1, $u_2 \in [0,1]$:

- **Clayton copula:** The *bivariate Clayton copula* is given by:
$$C(u_1,u_2) = (u_1^{-\delta} + u_2^{-\delta} - 1)^{-\frac{1}{\delta}} \ . \tag{2.10}$$
For the Clayton copula the parameter δ can take on values in $[-1,\infty) \setminus \{0\}$. However, for $\delta \in [-1,0)$ there arise numerical problems as stated in Dißmann (2010). Therefore we use the Clayton copula to model only positive dependence and thus $\delta \in (0,\infty)$: for $\delta \to 0$ we have independence, for $\delta \to \infty$ we have complete positive dependence.

- **Gumbel copula:** The *bivariate Gumbel copula* is given by:
$$C(u_1,u_2) = exp\big\{-[(-ln(u_1))^{\delta} + (-ln(u_1))^{\delta}]^{\frac{1}{\delta}}\big\} \ , \tag{2.11}$$
where $\delta \geq 1$. For $\delta = 1$ we have independence and for $\delta \to \infty$ we have complete positive dependence.

- **Frank copula:** The *bivariate Frank copula*, which can model positive as well as negative dependence, is defined as:
$$C(u_1,u_2) = -\frac{1}{\delta} ln\Big(\frac{1}{1-e^{-\delta}}[(1-e^{-\delta}) - (1-e^{-\delta u_1})(1-e^{-\delta u_2})]\Big) \ , \tag{2.12}$$
with $\delta \in (-\infty,\infty) \setminus \{0\}$. For $\delta \to 0$ we obtain independence, for $\delta \to -\infty$ complete negative dependence and for $\delta \to \infty$ complete positive dependence.

- **Joe copula:** The *bivariate Joe copula* is given by:
$$C(u_1,u_2) = 1 - \Big((1-u_1)^{\delta} + (1-u_2)^{\delta} - (1-u_1)^{\delta}(1-u_2)^{\delta}\Big)^{\frac{1}{\delta}} \ , \tag{2.13}$$
where the parameter δ takes on values $\delta > 1$. For $\delta \to 1$ we have independence and for $\delta \to \infty$ we have complete positive dependence.

The pair copulas listed above all have one parameter. They can be extended to $d \in \mathbb{N}$ dimensions and belong to the copula class called *Archimedean copulas*, which arise from generator functions[1]. In Nelsen (2007), which the list of pair copulas above is based on,

[1]These generator functions $\varphi : [0,1] \to [0,\infty)$ are continuous, strictly monotone decreasing and convex with $\varphi(1) = 0$. For a chosen generator function φ we obtain the Archimedean copula C by:
$$C(u_1,u_2) = \varphi^{[-1]}(\varphi(u_1) + \varphi(u_2)) \ ,$$
where $\varphi^{[-1]}(t) := \begin{cases} \varphi^{-1}(t) & \text{for } 0 \leq t \leq \varphi(0) \ , \\ 0 & \text{for } \varphi(0) \leq t \leq \infty \ . \end{cases}$

the reader can find more details. The *BB1* and *BB7* family are examples of Archimedean copula families with two parameters. The Archimedean class is covered in more detail for example in Nelsen (2007).
Further there exists the class of *extreme-value copulas*, which are based on a generalization of extreme-value theory to higher dimensions. Representatives of this class are the *Tawn copula*, *BB5* and *Extended Joe copula (BB8)*. In Gudendorf and Segers (2010) the reader can find more details.

Dependence and Association Measures

The possibly complex dependence structure between two random variables X_1 and X_2 is fully captured in their joint distribution function F and their corresponding pair copula C respectively. We introduce several one-number summaries of the pairwise association between two random variables, which give information about strength and direction of the association present, or the occurrence of joint extreme events.

Definition 2.2.5 (Pearson correlation)**.** Let the two random variables X_1 and X_2 have finite second moments. The *Pearson correlation coefficient* ρ between X_1 and X_2 is defined as:

$$\rho(X_1, X_2) := Cor(X_1, X_2) = \frac{Cov(X_1, X_2)}{\sqrt{Var(X_1)}\sqrt{Var(X_2)}} \in [-1, 1] \,. \tag{2.14}$$

If it is clear from the context, $\rho(X_1, X_2)$ will be denoted as ρ. For a set of observations $\{(x_{i1}, x_{i2}) \mid i \in [n]\}$ of (X_1, X_2) the *estimated Pearson correlation coefficient* is given by:

$$\hat{\rho} = \hat{\rho}(X_1, X_2) := \frac{\sum_{i=1}^{n}(x_{i1} - \bar{x}_1)(x_{i2} - \bar{x}_2)}{\sqrt{\sum_{i=1}^{n}(x_{i1} - \bar{x}_1)^2}\sqrt{\sum_{i=1}^{n}(x_{i2} - \bar{x}_2)^2}} \,, \tag{2.15}$$

with $\bar{x}_1 := \sum_{i=1}^{n} x_{i1}$ and $\bar{x}_2 := \sum_{i=1}^{n} x_{i2}$ the sample means.

The Pearson correlation is a measure of linear dependence and not "scale-invariant", i.e. not invariant under monotone increasing transformations of the random variables. Kurowicka and Cooke (2006) also showed, that the value of (2.14) depends on the marginal distributions of X_1 and X_2.

We wish to have an association measure, which is not influenced by whether we compute it on the original-scale or on the copula scale. Therefore it has to remain unaltered when applying the marginal distribution F_1 to X_1 and F_2 to X_2. In other words: we wish to have a scale-invariant measure.
Additionally, we would like to capture not only linear dependence, but also a general association between the values of two random variables X_1 and X_2. One aspect of this general association is the so-called *concordance*: Two random variables are said to be concordant, if "large" values of one variable tend to be associated with "large" values of the other random variable and equivalently "small" values of one tend to be associated with "small" values of the other. In a mathematical formulation, let (x_{11}, x_{12}) and (x_{21}, x_{22}) be a pair of observations of (X_1, X_2). It is said to be *concordant*, if $(x_{11} - x_{21})(x_{12} - x_{22}) > 0$ or *discordant*, if $(x_{11} - x_{21})(x_{12} - x_{22}) < 0$.

The *Kendall's* τ introduced in the following is a scale-invariant association measure for two continuous random variables X_1 and X_2 based on the comparison of concordant with discordant behavior:

Definition 2.2.6 (Kendall's τ)**.** For two continuous random variables X_1 and X_2 the *Kendall's* τ is defined as:

$$\tau(X_1, X_2) := P\big((X_{11} - X_{21})(X_{12} - X_{22}) > 0\big) - P\big((X_{11} - X_{21})(X_{12} - X_{22}) < 0\big) \ , \tag{2.16}$$

where (X_{11}, X_{12}) and (X_{21}, X_{22}) are i.i.d. copies of (X_1, X_2). Let $\{(x_{i1}, x_{i2}) \mid i \in [n]\}$ be n observations of (X_1, X_2). Then we have $\binom{n}{2}$ distinct pairs of observations of (X_1, X_2) being either concordant or discordant in the continuous case.[2] In the discrete case there can also exist so-called *extra* x_1 *pairs* for $x_{11} = x_{21}$ or *extra* x_2 *pairs* for $x_{12} = x_{22}$. Then the *estimated Kendall's* τ *(discrete case)* is given by:

$$\hat{\tau}_{discr.} := \frac{N_c - N_d}{\sqrt{N_c + N_d + N_1}\sqrt{N_c + N_d + N_2}} \ , \tag{2.17}$$

with N_c the number of concordant pairs, N_d the number of discordant pairs, N_1 the number of extra x_1 pairs and N_2 the number of extra x_2 pairs. The *estimated Kendall's* τ *(continuous case)* is defined as:

$$\hat{\tau}_{cont.} := \frac{N_c - N_d}{\binom{n}{2}} \ . \tag{2.18}$$

From the definition of Kendall's τ it becomes obvious, that with the help of Sklar's Theorem it can be expressed in terms of the copula associated with the distribution of (X_1, X_2):

Theorem 2.2.3 (Kendall's τ expressed in terms of copula)**.** *Let* (X_1, X_2) *be a continuous random vector. Then:*

$$\tau = 4 \int_{[0,1]^2} C(u_1, u_2) dC(u_1, u_2) - 1 \ . \tag{2.19}$$

Proof. Following Czado (2019), let $(X_{11}, X_{12}) \sim G$ and $(X_{21}, X_{22}) \sim G$ be two i.i.d. copies of (X_1, X_2). Let F_1 be the marginal distribution of X_1 and F_2 of X_2 respectively and let C be the bivariate copula distribution associated with G. We know that $\tau(X_1, X_2)$ is defined as:

$$\tau(X_1, X_2) = P\big((X_{11} - X_{21})(X_{12} - X_{22}) > 0\big) - P\big((X_{11} - X_{21})(X_{12} - X_{22}) < 0\big) \ .$$

With:

$$P\big((X_{11} - X_{21})(X_{12} - X_{22}) > 0\big) = 1 - P\big((X_{11} - X_{21})(X_{12} - X_{22}) < 0\big)$$

and thus:

$$\tau(X_1, X_2) = 2P\big((X_{11} - X_{21})(X_{12} - X_{22}) > 0\big) - 1 \ ,$$

[2]For (X_1, X_2) continuous the case $(x_{11} - x_{21})(x_{12} - x_{22}) = 0$ occurs with probability 0.

it remains to compute $P\big((X_{11}-X_{21})(X_{12}-X_{22})>0\big)$. It holds, that:

$$P\big((X_{11}-X_{21})(X_{12}-X_{22})>0\big)=P(X_{11}>X_{21},\,X_{12}>X_{22})+P(X_{11}<X_{21},\,X_{12}<X_{22})\ ,$$

and hence the proof is complete, once we evaluated $P(X_{11}>X_{21},\,X_{12}>X_{22})$ and $P(X_{11}<X_{21},\,X_{12}<X_{22})$. This can be done by integrating over either (X_{11},X_{12}) or (X_{21},X_{22}). We integrate over the former:

$$\begin{aligned}P(X_{11}>X_{21},\,X_{12}>X_{22}) &= \iint_{\mathbb{R}^2} P(X_{21}<x_1,\,X_{22}<x_2)dG(x_1,x_2)\\ &= \iint_{\mathbb{R}^2} G(x_1,\,x_2)dG(x_1,x_2)\\ &\overset{Sklar}{=} \iint_{\mathbb{R}^2} C(F_1(x_1),F_2(x_2))dC(F_1(x_1),F_2(x_2))\\ &= \iint_{\mathbb{R}^2} C(u_1,u_2)dC(u_1,u_2)\ ,\end{aligned}$$

with $u_1=F_1(x_1)$ and $u_2=F_2(x_2)$. By the same reasoning we have:

$$\begin{aligned}P(X_{11}<X_{21},\,X_{12}<X_{22}) &= \iint_{\mathbb{R}^2} P(X_{21}>x_1,\,X_{22}>x_2)dG(x_1,x_2)\\ &= \iint_{\mathbb{R}^2} \big[1-F_1(x_1)-F_2(x_2)+P(X_{21}<x_1,\,X_{22}<x_2)\big]dG(x_1,x_2)\\ &\overset{Sklar}{=} \iint_{\mathbb{R}^2} \big[1-F_1(x_1)-F_2(x_2)+C(F_1(x_1),\,F_2(x_2))\big]dC(F_1(x_1),\,F_2(x_2))\\ &= \iint_{\mathbb{R}^2} \big[1-u_1-u_2+C(u_1,\,u_2)\big]dC(u_1,\,u_2)\\ &= 1-\frac{1}{2}-\frac{1}{2}+\iint_{\mathbb{R}^2} C(u_1,\,u_2)dC(u_1,\,u_2)\\ &= \iint_{\mathbb{R}^2} C(u_1,\,u_2)dC(u_1,\,u_2)\ ,\end{aligned}$$

which concludes the proof. □

For the Gauss and Student's t copulas and the copula families introduced in Section 2.2 the expression for the Kendall's τ is given in Table 1. We notice, that for these copulas the Kendall's τ stands in one-to-one correspondence to the copula family parameter. The estimation method called *inversion of Kendall's* τ will later use this fact to estimate the parameter of the chosen copula family, i.e. ρ or δ respectively, from an estimated $\hat{\tau}$.

family	**Kendall's τ**	**range of Kendall's τ**
Gauss	$\tau = \frac{2}{\pi} arcsin(\rho)$	$[-1, 1]$
Student's t	$\tau = \frac{2}{\pi} arcsin(\rho)$	$[-1, 1]$
Gumbel	$\tau = 1 - \frac{1}{\delta}$	$[0, 1]$
Clayton	$\tau = \frac{\delta}{\delta+2}$	$[0, 1]$
Frank	$\tau = 1 - \frac{4}{\delta} + 4\frac{D_1(\delta)}{\delta}$ where $D_1(\delta) = \int_0^\delta \frac{x/\delta}{e^x-1} dx$ is the Debye function	$[-1, 1]$
Joe	$\tau = 1 + \left(\frac{-2+2\gamma+2ln(2)+\Psi(\frac{1}{\delta})+\Psi(\frac{1}{2}\frac{2+\delta}{\delta})+\delta}{-2+\delta}\right)$ where $\gamma \approx 0.57721$ is the Euler constant and $\Psi(x) = \frac{d}{dx} ln(\Gamma(x))$ the digamma function	$[0, 1]$

Table 1: Kendall's τ computed for elliptical copulas and Archimedean copulas with one parameter.

Association measures, which give information about the occurrence of joint extreme behavior, are the *upper and lower tail dependence coefficient:*

Definition 2.2.7 (Upper and lower tail dependence coefficients)**.** Let $X_1 \sim F_1$ and $X_2 \sim F_2$ be two continuous random variables. The *upper tail dependence coefficient* λ^{upper} is defined as:

$$\lambda^{upper} := \lim_{t \to 1^-} P\big(X_2 > F_2^{-1}(t) \mid X_1 > F_1^{-1}(t)\big) = \lim_{t \to 1^-} \frac{1 - 2t + C(t,t)}{1 - t} \ , \tag{2.20}$$

and the *lower tail dependence coefficient* λ^{lower} is given by:

$$\lambda^{lower} := \lim_{t \to 0^+} P\big(X_2 < F_2^{-1}(t) \mid X_1 < F_1^{-1}(t)\big) = \lim_{t \to 0^+} \frac{C(t,t)}{t} \ . \tag{2.21}$$

Explicitly calculated, we obtain the following upper and lower tail dependence coefficients for the different copula families mentioned in Subsection 2.2 and before:

family	**upper tail dependence** λ^{upper}	**lower tail dependence** λ^{lower}
Gauss	0	0
Student's t	$2\ \mathrm{t}_{\nu+1}\Big(-\sqrt{\nu+1}\sqrt{\frac{1-\rho}{1+\rho}}\Big)$	$2\ \mathrm{t}_{\nu+1}\Big(-\sqrt{\nu+1}\sqrt{\frac{1-\rho}{1+\rho}}\Big)$
Gumbel	$2-2^{1/\delta}$	0
Clayton	0	$2^{-1/\delta}$
Frank	0	0
Joe	$2-2^{1/\delta}$	0

Table 2: Upper and lower tail dependence coefficients of selected copula families.

Both the BB1 and BB7 copula model upper and lower tail dependence asymmetrically. The Tawn copula has a lower tail dependence coefficient equal to 0.

Rotation of Bivariate Copulas

We have already seen that some copulas, e.g. the Gumbel copula, have $\lambda^{upper} \neq 0$ and $\lambda^{lower} = 0$, i.e. they are only suited to model data that show strong dependence in the upper tail, some have $\lambda^{upper} = 0$ and $\lambda^{lower} \neq 0$, i.e. they are only suited to model lower tail dependence in the data and other none at all because of $\lambda^{upper} = \lambda^{lower} = 0$. When fitting a bivariate copula to data, which for example show a clear dependence in the lower tail, the Gumbel copula among all other copula families, which do not model lower tail dependence, would not be a reasonable choice. The concept of rotating the Gumbel copula, such that instead of upper tail dependence it can handle lower tail dependence, seems very natural here. Rotations of pair-copulas are indeed used to extend their modelling capability.

We rotate the copula density $c(\cdot,\cdot)$ counterclockwise by:

- 90° and define $c_{90}(u_1, u_2) := c(1-u_2, u_1)$,
- 180° and define $c_{180}(u_1, u_2) := c(1-u_1, 1-u_2)$ and
- by 270° and define $c_{270}(u_1, u_2) := c(u_2, 1-u_1)$.

For the Gumbel, Clayton and Joe copula the range of Kendall's τ can be extended to take on values in the whole interval $[-1, 1]$ by rotations of 90° and 270° respectively.
Figure 1 illustrates the rotation of a Gumbel copula. The plots displayed are so-called *normalized contour plots*, which are commonly used to visually assess the characteristic shape of bivariate copulas and the tail dependence behavior. They show the contours of the density:

$$f(z_1, z_2) = c(\Phi(z_1), \Phi(z_2))\phi(z_1)\phi(z_2)\ ,$$

of the random vector $(Z_1, Z_2) := \big(\Phi^{-1}(F_1(X_1)), \Phi^{-1}(F_2(X_2))\big)$.

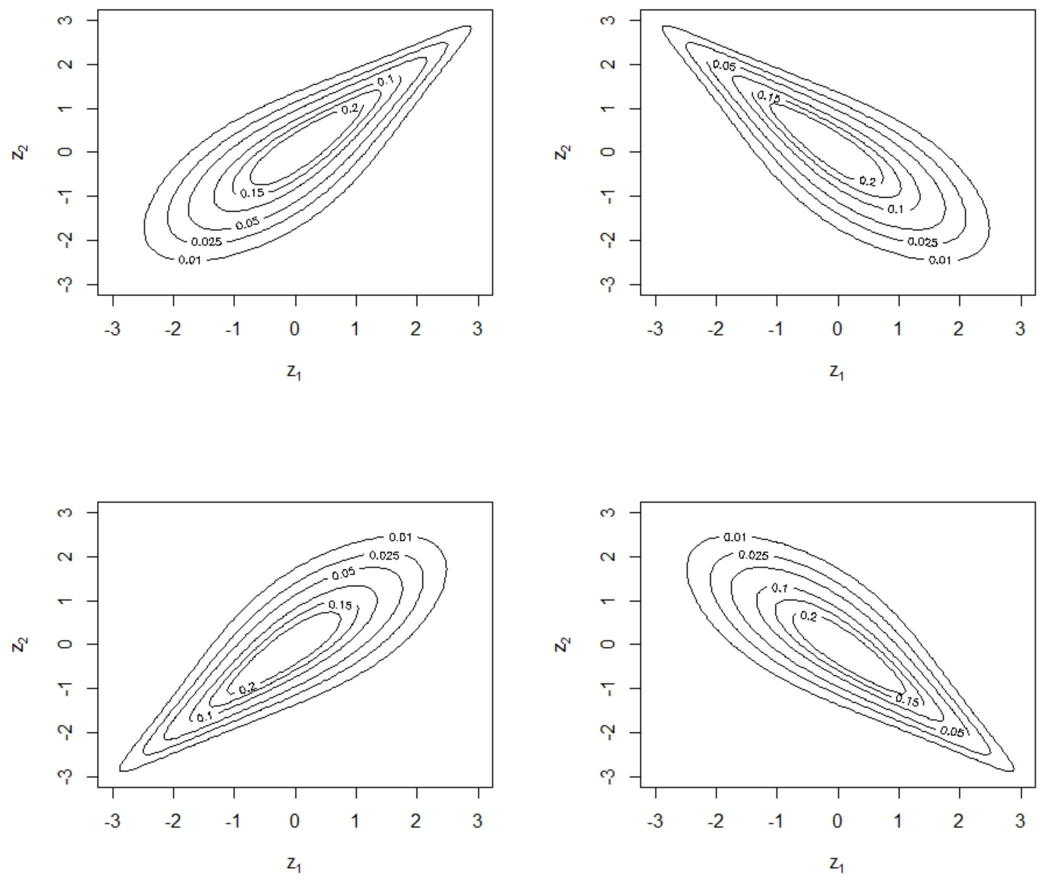

Figure 1: Normalized contour plots of rotated Gumbel copula: top left: rotation of 0°, $\tau = 0.6$, top right: rotation of 90°, $\tau = -0.6$, bottom left: rotation of 180°, $\tau = 0.6$ and bottom right: rotation of 270°, $\tau = -0.6$.

Pair Copula Construction (PCC)

We have seen, that with the help of Sklar's Theorem 2.2.1 a multivariate distribution can be decomposed into or constructed from its corresponding multivariate copula and its marginal distributions. Especially from the construction point of view, the set of multivariate copulas to choose from, i.e. elliptical and Archimedean copulas, is rather limited so far and constrained in modeling flexibility. However, complex high-dimensional dependence structures call for more flexible multivariate copulas.
An answer can be found when we look into the decomposition of multivariate distributions: Aas et al. (2009), which the following subsection is based on, decompose a multivariate density by using a cascade of bivariate building blocks: pair copulas. Knowing how to decompose a multivariate distribution, we can reverse this approach in order to construct multivariate copulas and distribution functions respectively. These are flexible and their construction is simple. This is the idea of *pair copula construction.*

Before we start with a three dimensional example, we define the following notation:

Definition 2.2.8. Let $\boldsymbol{X}_D \in \mathbb{R}^d$ be a random vector and $\boldsymbol{x}_D \in \mathbb{R}^d$, let $i, j, d \in \mathbb{N}$ and $D \subset \mathbb{N}$ with $i, j \notin D$ and $|D| = d$. Let $F_{ij\,|\,D}(\cdot, \cdot \,|\, \boldsymbol{X}_D = \boldsymbol{x}_D)$ be the conditional distribution of (X_i, X_j) given that $\boldsymbol{X}_D = \boldsymbol{x}_D$. The copula distribution associated with $F_{ij\,|\,D}(\cdot, \cdot \,|\, \boldsymbol{X}_D = \boldsymbol{x}_D)$ is denoted by:

$$C_{ij;D}(\cdot, \cdot; \boldsymbol{x}_D) \; .$$

If existing, its corresponding density is denoted by:

$$c_{ij;D}(\cdot, \cdot; \boldsymbol{x}_D) \; .$$

Example 2.2.1. Let $\boldsymbol{X} = (X_1, X_2, X_3)$ be a random vector with joint density function f_{123} and marginal density functions f_1, f_2 and f_3. Using conditioning we can rewrite the joint density function:

$$f_{123}(x_1, x_2, x_3) = f_{1|23}(x_1 \,|\, x_2, x_3) f_{2|3}(x_2 \,|\, x_3) f_3(x_3) \; , \tag{2.22}$$

with:

$$f_{2|3}(x_2 \,|\, x_3) = \frac{f_{23}(x_2, x_3)}{f_3(x_3)} \; , \tag{2.23}$$

$$f_{1|23}(x_1 \,|\, x_2, x_3) = \frac{f_{123}(x_1, x_2, x_3)}{f_{23}(x_2, x_3)} = \frac{f_{13|2}(x_1, x_3 \,|\, x_2)}{f_{3|2}(x_3 \,|\, x_2)} \; . \tag{2.24}$$

By Sklar's Theorem 2.2.1 we know, that:

$$f_{23}(x_2, x_3) = c_{23}(F_2(x_2), F_3(x_3)) f_2(x_2) f_3(x_3) \; , \tag{2.25}$$

and thus (2.23) becomes:

$$f_{2|3}(x_2 \,|\, x_3) := \frac{f_{23}(x_2, x_3)}{f_3(x_3)} = c_{23}(F_2(x_2), F_3(x_3)) f_2(x_2) \; . \tag{2.26}$$

In the same manner we obtain (2.24):

$$\begin{aligned} f_{1|23}(x_1 \,|\, x_2, x_3) &= \frac{f_{13|2}(x_1, x_3 \,|\, x_2)}{f_{3|2}(x_3 \,|\, x_2)} \\ &= \frac{c_{13;2}(F(x_1 \,|\, x_2), F(x_3 \,|\, x_2); x_2) f_{1|2}(x_1 \,|\, x_2) f_{3|2}(x_3 \,|\, x_2)}{f_{3|2}(x_3 \,|\, x_2)} \\ &= c_{13;2}(F(x_1 \,|\, x_2), F(x_3 \,|\, x_2); x_2) f_{1|2}(x_1 \,|\, x_2) \\ &= c_{13;2}(F(x_1 \,|\, x_2), F(x_3 \,|\, x_2); x_2) c_{12}(F_1(x_1), F_2(x_2)) f_1(x_1) \; . \end{aligned} \tag{2.27}$$

Combining (2.26) and (2.27) we can decompose (2.22) into a product of pair copulas and marginal distributions:

$$\begin{aligned} f_{123}(x_1, x_2, x_3) = \; & c_{13;2}(F(x_1 \,|\, x_2), F(x_3 \,|\, x_2); x_2) \\ & c_{12}(F_1(x_1), F_2(x_2)) \, c_{23}(F_2(x_2), F_3(x_3)) \\ & f_1(x_1) \, f_2(x_2) \, f_3(x_3) \; . \end{aligned} \tag{2.28}$$

Remark 2.2.5. (a) The decomposition with conditioning in (2.22) is not unique. Neither is therefore (2.28). In general we could reorder (X_1, X_2, X_3) in $3! = 6$ ways. However, in a pair copula there is no distinction made between the first and the second argument, i.e. $c_{ij}(u_i, u_j) = c_{ji}(u_j, u_i)$. That is why we end up with three distinct decompositions in the form of (2.28).

(b) We see, that $c_{13;2}(\cdot, \cdot; x_2)$, the pair copula associated with the conditional distribution of (X_1, X_3) given $X_2 = x_2$ depends on the value x_2 of X_2. We stick to the terminology in Czado (2019) and speak of a *pair copula decomposition*, if the copulas associated with conditional distributions are allowed to depend on the value of the conditioning variable, i.e. here $X_2 = x_2$. If we ignore this dependence, which in our case would be equivalent to:

$$\forall x_2 \in \mathbb{R}: \quad c_{13;2}(u_1, u_3; x_2) = c_{13;2}(u_1, u_3), \quad u_1 \in [0,1],\ u_3 \in [0,1] \ ,$$

we make the *simplifying assumption* in three dimensions. In general it assumes, that copulas associated with conditional distributions do not depend on the value(s) of the conditioning variable(s). More details on whether this assumption is too simplistic can be found in Haff et al. (2010). If we assume the simplifying assumption, we can reverse the decomposition approach and view the simplified version of (2.28) as the construction of the three dimensional density f_{123} from pair copula densities, conditional distributions and marginal densities. In this case we speak of *pair copula construction*.

(c) For the rest of the thesis we will work with the simplifying assumption, even if not mentioned explicitly, and therefore speak of pair copula construction.

(d) Obviously the construction in Example 2.2.1 can be generalized to higher dimensions. There we encounter conditional marginal densities, which can be expressed as:

$$f(x \mid \boldsymbol{v}) = c_{xv_j;\boldsymbol{v}_{-j}}(F(x \mid \boldsymbol{v}_{-j}), F(v_j \mid \boldsymbol{v}_{-j})) \cdot f(x \mid \boldsymbol{v}_{-j}) \ , \tag{2.29}$$

with $\boldsymbol{v} \in \mathbb{R}^d$ and $\boldsymbol{v}_{-j}$ the sub-vector of $\boldsymbol{v}$ with the jth component left out. The second factor of (2.29) can again be factorized with (2.29). This illustrates the iterative nature of the construction. Finally, with the result of Joe (1996), that:

$$\begin{aligned}\forall j: \quad F(x \mid \boldsymbol{v}) &= \left.\frac{\partial C_{x,v_j;\boldsymbol{v}_{-j}}\big(F(x \mid \boldsymbol{v}_{-j}), u\big)}{\partial u}\right|_{u=F(v_j \mid \boldsymbol{v}_{-j})} \\ &=: \frac{\partial C_{x,v_j;\boldsymbol{v}_{-j}}\big(F(x \mid \boldsymbol{v}_{-j}), F(v_j \mid \boldsymbol{v}_{-j})\big)}{\partial F(v_j \mid \boldsymbol{v}_{-j})} \ ,\end{aligned} \tag{2.30}$$

the construction or decomposition respectively is completed. Here *h-functions* help to simplify the notation of conditional distributions and copulas.

Definition 2.2.9. For a bivariate copula C_{uv} the corresponding *h-function* is defined for all $(u, v) \in [0,1]^2$ as:

$$h_{u \mid v}(u \mid v) := \frac{\partial}{\partial v} C_{uv}(u, v) \ . \tag{2.31}$$

Clearly (2.30) holds for any continuous distribution F and thus also for the bivariate copula distribution C_{uv}. With $C(u) = u$ for any $u \in [0,1]$ and the copula C it follows that:

$$C_{u\,|\,v}(u\,|\,v) \overset{(2.30)}{=} \frac{\partial}{\partial v} C_{uv}(u,v) \overset{2.2.9}{=} h_{u\,|\,v}(u\,|\,v)\;. \tag{2.32}$$

2.3 Regular Vines

In Section 2.2 we saw that a d-dimensional probability distribution function can be constructed from or decomposed into bivariate building-blocks, pair copulas. For a specific d-dimensional probability distribution there exist several pair copula constructions, a subset of them satisfying a condition, which we will later get to know as the *proximity condition.* Bedford and Cooke (2001) and Bedford and Cooke (2002) introduced *regular vines (R-vines)* and the *R-vine specification* to efficiently represent the pair copula constructions satisfying the proximity condition. The *R-vine specification* captures the structure of the pair copula construction: each bivariate copula is associated with an edge in a sequence of nested trees, the *R-vine.* The families, rotations and parameters of the bivariate copulas may be stored in matrices. This compact notation facilitates the estimation and sampling procedures on R-vines. Bedford and Cooke (2001) and Bedford and Cooke (2002) also show, that each R-vine specification stands for a unique d-dimensional distribution F.

Regular vines rely on concepts of graph theory, to which a brief introduction will be given next.

A Brief Introduction to Graph Theory

For more details the reader is directed to Diestel (2018). A *graph* consists of *nodes*, which are connected with *edges.* It is therefore very suitable to capture and illustrate for example the relationship (represented be an edge) between several random variables (represented by nodes). After introducing the basic concept we give special types of graphs, namely paths, cycles and trees, whose name already gives a good intuition about their structure.

Definition 2.3.1 (graph, adjacent, subgraph)**.**

- A graph $G = (V,E)$ is a pair of sets with $E \subset \{\{v,w\}\,|\,v,w \in V\}$. For $E \subset \{(v,w)\,|\,v,w,\in V\}$, G is called *directed.*
- The elements of the set V are called *vertices* or *nodes*, the elements of E are called *edges.*
- Two nodes $v,w \in V$ are called *adjacent*, if they are joined by an edge: $\{v,w\} \in E$. For $e = \{v,w\} \in E$ the nodes v and w are called *endpoints.*
- Let $V' \subset V$ and $E' \subset E$. Then $G' := (V',E')$ is a *subgraph.*

From now on we will speak about undirected graphs, even if not explicitly specified.
Now that we can depict the dependence structure of several random variables in a graph we want to additionally capture the strength of association of each pair of random variables,

which are related to each other, i.e. are connected by an edge. In other words we want to assign each edge a *weight*, i.e. the value of the association measure of the two random variables connected. For this reason we introduce the concept of a *weighted graph*.

Definition 2.3.2 (weighted graph)**.** Let $G = (V, E)$ be a graph. G is called *weighted*, if there exists a function $f : E \to \mathbb{R}$ assigning a weight $f(e)$ to each edge $e \in E$. Then G can be denoted as $G = (V, E, f)$.

Random variables, which are endpoints of many edges in the graph, are in relation with many other random variables: they are influential. For being able to describe this from the graph, we introduce the *degree of a node*.

Definition 2.3.3 (degree of a node)**.** Let $G = (V, E)$ be a graph. The degree of a node $v \in V$ is defined as:

$$deg(v) := \big|\{w \in V \mid \{v, w\} \in E\}\big| \,.$$

It is the number of neighbors, which are connected to v via an edge.

Graphs can itself take on different structures, by which they can be characterized.

Definition 2.3.4 (path, cycle, connected, acyclic)**.** Let $G = (V, E)$ and $P = (W, F)$ be two graphs.

- P is a *path*, if it is a non-empty and:

  $$W = \{v_0, v_1, \dots, v_k\} \,, \quad F = \{\{v_0, v_1\}, \{v_1, v_2\}, \dots, \{v_{k-1}, v_k\}\} \,,$$

  for some $k \in \mathbb{N}$ and with $v_i \neq v_j$ for all $i \neq j,\ i, j \in [k]$. The number of edges of the path is its *length*.

- G is called *connected*, if there exists a path between any two nodes $v, w \in V$.

- If $v_0 = v_k$, then P is called a *cycle*. G is called *acyclic*, if it does not contain any cycles as subgraphs.

Graphs, which are so-called *trees*, will become central for regular vines. Their name already suggests their structure.

Definition 2.3.5 (tree, leaf)**.** Let $T = (V, E)$ be a graph. Then T is a *tree*, if it is acyclic and connected. A node $v \in V$ is called a *leaf*, if $deg(v) = 1$.

The following properties closer characterize trees.

Theorem 2.3.1. *The following assertions are equivalent for a graph $T = (V, E)$:*

(i) T is a tree.

(ii) Any two vertices of T are linked by a unique path in T.

(iii) T is minimally connected, i.e. T is connected but $T' := (V, E \setminus \{e\})$ is not connected for any edge $e \in E$.

(iv) T is maximally acyclic, i.e. T contains no cycle but $T' := (V, E \cup \{v, w\})$ does, for any two non-adjacent vertices $v, w \in V$.

For a graph, which is not a tree, it can be possible to find a subgraph *spanning* the whole graph while fulfilling the characteristics of a tree. This is then called a *spanning tree.*

Definition 2.3.6 (spanning tree)**.** A subgraph $T = (V_T, E_T)$ of $G = (V, E)$ is called a *spanning tree* of G, if T is a tree and $V = V_T$.

Regular Vines

Definition 2.3.7 (Vine, regular vine, regular vine tree sequence)**.** A set of trees $\mathcal{V} = (T_1, ..., T_{d-1})$ is a *vine* on d elements if:

(i) T_1 is a tree with edge set E_1 and node set $V_1 = \{1, ..., d\}$.

(ii) For $i \in \{2, ..., (d-1)\}$ it holds that T_i is a tree with edge set E_i and node set $V_i = E_{i-1}$.

$\mathcal{V}$ is an *regular vine* (*R-vine*) or *regular vine tree sequence* (*R-vine tree sequence*) if additionally the so called *proximity condition* holds:

(iii) For $i \in \{2, ..., (d-1)\}$ and $\{a, b\} \in E_i$ with $a = \{a_1, a_2\}$ and $b = \{b_1, b_2\}$ we have that $|a \cap b| = 1$.

Remark 2.3.1. The *proximity condition* makes sure that nodes a and b are only then joined by an edge in tree T_i if they share a common node in tree T_{i-1}, where $a, b \in E_{i-1}$.

Among the R-vines we have two sub-classes: the C-vines and the D-vine. They distinguish themselves through a special structure each tree in $\mathcal{V}$ takes on.

Definition 2.3.8 (C-vine, D-vine)**.** An R-vine tree sequence $\mathcal{V}$ on d elements is called:

(i) *D-vine*, if for each node v of each tree $T_i \in \mathcal{V}$, $i \in [d-1]$ it holds that $deg(v) \leq 2$,

(ii) *C-vine*, if in each tree $T_i \in \mathcal{V}$, $i \in [d-1]$ there is one unique node v with $deg(v) = d-i$ which is called *root node.*

Example 2.3.1 (C-vine). Below we see the tree sequence of a C-vine on 5 elements without node and edge labels. The root node in each star-shaped tree is coloured. The trees (T3) and (T4) are stars and paths at the same time.

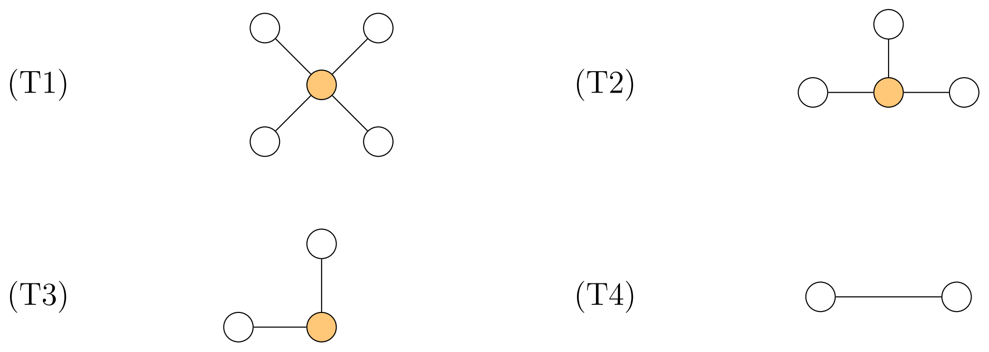

Figure 2: A C-vine on 5 elements.

In order to illustrate the concept of an R-vine tree sequence and its nested structure we give another example.

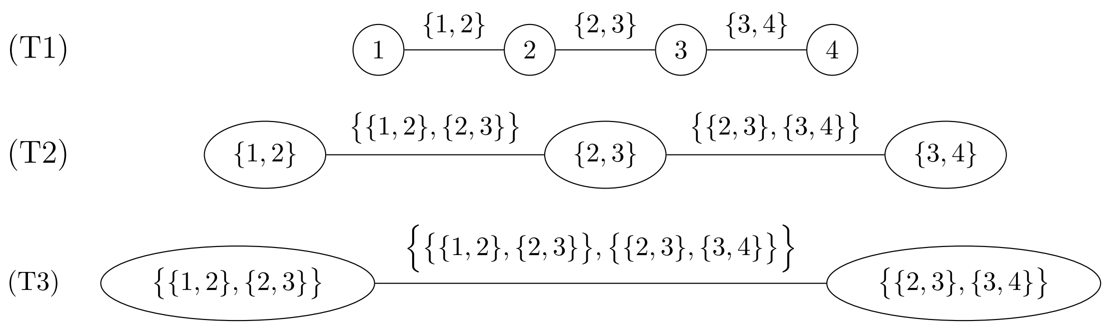

Figure 3: A D-vine on 4 elements.

Example 2.3.2 (R-vine tree sequence on 4 elements)*.* In Figure 3 an R-vine tree sequence $\mathcal{V}$ on $d = 4$ elements is displayed. We notice that $\mathcal{V}$ is actually a D-vine. This is recognizable by the path structure of the graphs, which is the geometric consequence of $deg(v) \leq 2 \; \forall v \in V_i$ with $(V_i, E_i) = T_i \in \mathcal{V}$ and connectedness. It is obvious from the trees that conditions (i) and (ii) of Definition 2.3.7 are satisfied: (i) the node set $V_1 = \{1, 2, 3, 4\}$ and (ii) $V_2 = E_1$ as well as $V_3 = E_2$. That the proximity condition (iii) holds here is only slightly harder to check:

- In tree T_2 we have the edge set:

$$E_2 = \Big\{\{\{1,2\},\{2,3\}\},\{\{2,3\},\{3,4\}\}\Big\} .$$

According the proximity the proximity condition 2.3.7 we check for edge $\big\{\{1,2\},\{2,3\}\big\}$ and set:

$$\{a,b\} := \big\{\{1,2\},\{2,3\}\big\} , \quad a := \{1,2\} , \quad b := \{2,3\} .$$

Then:

$$|a \cap b| = |\{1,2\} \cap \{2,3\}| = |\{2\}| = 1 .$$

Similarly, for the second edge we can set:

$$\{a,b\} := \big\{\{2,3\},\{3,4\}\big\} , \quad a := \{2,3\} , \quad b := \{3,4\} .$$

and obtain:

$$|a \cap b| = |\{2,3\} \cap \{3,4\}| = |\{3\}| = 1 .$$

- In tree T_3 the edge set E_3 contains only of one edge:

$$E_3 = \Bigg\{\Big\{\{\{1,2\},\{2,3\}\},\{\{2,3\},\{3,4\}\}\Big\}\Bigg\} .$$

If we set:

$$\{a,b\} := \Big\{\{\{1,2\},\{2,3\}\},\{\{2,3\},\{3,4\}\}\Big\}$$
$$a := \{\{1,2\},\{2,3\}\}\ , \quad b := \{\{2,3\},\{3,4\}\}\ ,$$

we obtain:

$$|a \cap b| = |\{\{1,2\},\{2,3\}\} \cap \{\{2,3\},\{3,4\}\}| = |\{\{2,3\}\}| = 1\ .$$

Thus in total the proximity condition holds.

This notation for edges and nodes is hard to read and use. By Bedford and Cooke (2002) and results of Kurowicka and Cooke (2003) the edges of each tree can be uniquely identified by two *conditioned nodes* and a set of *conditioning nodes*. With the following definition we introduce this concept of representation.

Definition 2.3.9 (Complete union, conditioning set, conditioned set)**.** Let $\mathcal{V}$ be an R-vine tree sequence. The *complete union* U_e of the edge $e \in E_i$ is defined as:

$$U_e := \{j \in V_1 \mid \exists e_1 \in E_1, ..., e_{i-1} \in E_{i-1} \quad \text{s.th.} \quad j \in e_1 \in ... \in e_{i-1} \in e\}. \tag{2.33}$$

The set:

$$D_e := U_a \cap U_b \tag{2.34}$$

is called *conditioning set* D_e of an edge $e = \{a,b\}$ and the *conditioned sets* $\mathcal{C}_{e,a}$, $\mathcal{C}_{e,b}$ and $\mathcal{C}_e$ are given by:

$$\mathcal{C}_{e,a} := U_a \setminus D_e\ , \quad \mathcal{C}_{e,b} := U_b \setminus D_e \quad \text{and} \quad \mathcal{C}_e{}^1 := \mathcal{C}_{e,a} \cup \mathcal{C}_{e,b}\ . \tag{2.35}$$

Example 2.3.2 (continued). If we use the notation introduced above we obtain the following conditioning sets:

$$\begin{aligned}
T_1: &\quad D_{\{1,2\}} = \emptyset\ , \quad D_{\{2,3\}} = \emptyset\ , \quad D_{\{3,4\}} = \emptyset\ ,\\
T_2: &\quad D_{\{\{1,2\},\{2,3\}\}} = \{2\}\ , \quad D_{\{\{2,3\},\{3,4\}\}} = \{3\}\ ,\\
T_3: &\quad D_{\{\{\{1,2\},\{2,3\}\},\{\{2,3\},\{3,4\}\}\}} = \{2,3\}\ .
\end{aligned}$$

and the following conditioned sets:

$$\begin{aligned}
T_1: &\quad \mathcal{C}_{\{1,2\}} = \{1,2\}\ , \quad \mathcal{C}_{\{2,3\}} = \{2,3\}\ , \quad \mathcal{C}_{\{3,4\}} = \{3,4\}\ ,\\
T_2: &\quad \mathcal{C}_{\{\{1,2\},\{2,3\}\}} = \{1,3\}\ , \quad \mathcal{C}_{\{\{2,3\},\{3,4\}\}} = \{2,4\}\ ,\\
T_3: &\quad \mathcal{C}_{\{\{\{1,2\},\{2,3\}\},\{\{2,3\},\{3,4\}\}\}} = \{1,4\}\ .
\end{aligned}$$

[1]The set $\mathcal{C}_e$ will be called *conditioned set* as well. This might seem confusing as it is actually the union of the two conditioned sets $\mathcal{C}_{e,a}$ and $\mathcal{C}_{e,b}$. However, in the pair copula construction we deal with pair copulas of two random variables X_i and X_j with $\{i,j\} = \mathcal{C}_e$. So most of the time we speak about $\mathcal{C}_e$, which will therefore be called *conditioned set* for simplicity.

Figure 4 displays a tree sequence with the new notation which is more readable. It still describes the R-vine tree sequence uniquely up to permutation and order of the elements of the set $\mathcal{C}_e$.[2]

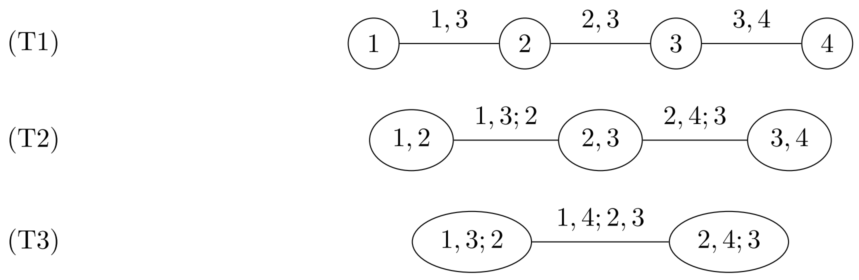

Figure 4: A D-vine on 4 elements with the notation of Definition 2.3.9.

Definition 2.3.10 (Constraint set)**.** The *constraint set* $\mathcal{CV}$ for the R-vine tree sequence $\mathcal{V}$ is defined as:

$$\mathcal{CV} := \left\{ (\mathcal{C}_{e,a}, \mathcal{C}_{e,b}; D_e) \mid e = \{a, b\},\ e \in E_i \quad \text{for} \quad i \in [d-1] \right\} . \tag{2.36}$$

Here the edge $e = (\mathcal{C}_{e,a}, \mathcal{C}_{e,b}; D_e)$ of the R-vine tree sequence will often be abbreviated by $e = (e_a, e_b; D_e)$.

Example 2.3.2 (continued). Here the constraint set is:

$$\left\{ (1,2), (2,3), (3,4), (1,3;2), (2,4;3), (1,4;2,3) \right\} .$$

Note that the curly braces of the conditioned and conditioning sets are left out. This is not completely precise nor consistent to Definition 2.3.10, but facilitates notation. In Figure 4 the round braces are left out as well.

The following results, which are stated and proved in Kurowicka and Cooke (2003) on pages 228 to 231, show that any edge of an R-vine tree sequence can be identified by its conditioning or conditioned sets. Let $\mathcal{V}$ be an R-vine tree sequence on d elements and and $a, b \in E_i \in T_i \in \mathcal{V}$ with $i \in [d-1]$:

- If $U_a = U_b$, then $a = b$.
- If the conditioned sets of edges a, b in $\mathcal{V}$ are equal, then $a = b$.

Definition 2.3.11 (R-vine specification)**.** The triple $(\boldsymbol{F}, \mathcal{V}, B)$ is called *R-vine specification* if:

(i) $\boldsymbol{F} = (F_1, ..., F_d)$ is a vector of continuous and invertible distribution functions,

(ii) $\mathcal{V}$ is an R-vine tree sequence on d elements and

(iii) $B := \left\{ C_e \mid e \in E_i \quad \text{for} \quad i \in [d-1] \right\}$ is the set of bivariate copulas C_e with E_i the edge set of tree T_i of the R-vine tree sequence $\mathcal{V}$.

[2]As a convention we order the elements of $\mathcal{C}_e$ in ascending as done in Figure 4.

By this definition each edge $e \in E_i$ of a tree T_i in $\mathcal{V}$ corresponds to a bivariate copula C_e.

Definition 2.3.12 (Realizing an R-vine specification)**.** A joint distribution F of the random vector $\boldsymbol{X} = (X_1, ..., X_d)$ is said to *realize an R-vine specification* $(\boldsymbol{F}, \mathcal{V}, B)$ or *exhibit an R-vine dependence*, if C_e is the bivariate copula of $X_{\mathcal{C}_{e,a}}$ and $X_{\mathcal{C}_{e,b}}$ given $\boldsymbol{X}_{D_e}$ for each edge $e = \{a, b\} \in E_i$ and the marginal distribution of X_i is F_i for $i \in [d]$.

Remark 2.3.2 (Simplifying assumption). The assumption that for each edge e of $\mathcal{V}$ the bivariate copula C_e does not depend on the value $\boldsymbol{x}_{D_e}$ the conditioning random vector $\boldsymbol{X}_{D_e}$ takes on is called *simplifying assumption.*

Theorem 2.3.2. *Let $(\boldsymbol{F}, \mathcal{V}, B)$ be an R-vine specification on d elements where all pair-copulas $C_e \in B$ satisfy the simplifying assumption and have densities c_e. There is a unique distribution F that realizes this R-vine specification with density:*

$$f_{1,...d}(x_1, ..., x_d) = \prod_{i=1}^{d} f_i(x_i) \cdot \tag{2.37}$$

$$\prod_{i=1}^{d-1} \prod_{e \in E_i} c_{\mathcal{C}_{e,a},\mathcal{C}_{e,b};D_e}\left(F_{\mathcal{C}_{e,a}|D_e}(x_{\mathcal{C}_{e,a}}|\boldsymbol{x}_{D_e}), F_{\mathcal{C}_{e,b}|D_e}(x_{\mathcal{C}_{e,b}}|\boldsymbol{x}_{D_e})\right) , \tag{2.38}$$

where f_i denote the densities of F_i.

Proof. The proof of theorem can be found in Bedford and Cooke (2001) and Bedford and Cooke (2002). □

Representing Regular Vines: Regular Vine Matrices

In order to efficiently store an R-vine specification $(\boldsymbol{F}, \mathcal{V}, B)$, we introduce *R-vine matrices* for the R-vine structure. *R-vine family matrices* and *R-vine parameter matrices* specify the pair copulas of the pair copula construction. R-vine matrices were first introduced by Dißmann (2010), based on Kurowicka (2009). We start with some notation:

- Let $M = (m_{i,j})_{i,j \in \{1,2,...,d\}} \in \{1, 2, ..., d\}^{d \times d}$ be a lower triangular matrix.
- Let the set of non-zero entries of the ith column in M be:

$$L_M(i) := \{m_{i,i}, m_{i+1,i}, ..., m_{d,i}\} \ . \tag{2.39}$$

- Additionally, we define:

$$B_M(i) := \left\{(m_{i,i}, D) \mid D = \{m_{k,i}, ...m_{d,i}\},\ k \in \{i+2, ..., d\}\right\} \tag{2.40}$$

$$\tilde{B}_M(i) := \left\{(m_{k,i}, D) \mid D = \{m_{i,i}\} \cup \{m_{k+1,i}, ...m_{d,i}\},\ k \in \{i+2, ..., d\}\right\} . \tag{2.41}$$

Now we can give the following definition:

Definition 2.3.13 (R-vine matrix)**.** Let $M \in \{1, 2, ..., d\}^{d \times d}$ be a lower triangular matrix. M is called an *R-vine matrix*, if it satisfies the three conditions:

(i) $L_M(i) \subset L_M(j)$ for all $1 \leq j < i \leq d$,

(ii) $m_{i,i} \notin L_M(i+1)$ for all $i \in [d-1]$,

(iii) for all $i \in [d-1]$ it holds that for all $k \in \{i+1, ..., d-1\}$:

$$\big(m_{k,i}, \{m_{k+1,i}, ..., m_{d,i}\}\big) \in B_M(i+1) \cup ... \cup B_M(d-1) \cup \tilde{B}_M(i+1) \cup ... \cup \tilde{B}_M(d-1) \; .$$

We want to interpret the conditions given in Definition 2.3.13.

Remark 2.3.3. Condition (i) makes sure, that every column i contains the entries of all columns right of column i. Condition (ii) requires, that all elements of $[d]$ occur exactly once as a diagonal entry of M. After being a diagonal entry in column i, $m_{i,i}$ will not occur in a column right of column i. It can be shown, that (i) and (ii) follow from condition (iii). However, they are kept for better understanding. Condition (iii) is hard to check for a given R-vine matrix, but essentially, it assures that the R-vine tree sequence constructed from the R-vine matrix satisfies the proximity condition.

Dißmann (2010) shows, that there are 2^{d-1} R-vine matrices specifying the same R-vine tree sequence on d elements. An algorithm to compute an R-vine matrix from an R-vine tree sequence and to construct an R-vine tree sequence from an R-vine matrix can be found in Stöber and Czado (2017)[3].

Remark 2.3.4. Generally we can read off the edges of the respective trees in the R-vine tree sequence from the R-vine matrix M as follows:

1. The nodes of T_1 are $V_1 = [d]$, which is clear without looking at the R-vine matrix.

2. The edges of T_1 are given as the set:

$$E_1 = \big\{\{m_{i,i}, m_{d,i}\} \mid i \in [d-1]\big\} \; . \tag{2.42}$$

3. For all trees T_j with $j \in \{2, ..., d-1\}$ the edges are given by:

$$E_j = \big\{\{m_{i,i}, m_{(d-j+1),i} \mid m_{(d-j+2),i}, ..., m_{d,i}\} \mid i \in [d-j]\big\} \; . \tag{2.43}$$

We want to illustrate Definition 2.3.13 and this construction procedure by an example.

Example 2.3.3. We consider the R-vine matrix M:

$$M = \begin{pmatrix} 1 & & & & \\ 5 & 4 & & & \\ 4 & 5 & 2 & & \\ 3 & 2 & 5 & 3 & \\ 2 & 3 & 3 & 5 & 5 \end{pmatrix} \; . \tag{2.44}$$

[3]Note, that Stöber and Czado (2017) use upper triangular matrices instead of lower triangular matrices as R-vine matrices. Both are common choices in the literature. To read and construct upper triangular R-vine matrices, one can just "reflect" the methods shown here. In the R-package `rvinecopulib` by Nagler and Vatter (2021) anti-diagonal upper triangular matrices are used as R-vine matrices to ease the use of indices in the algorithms.

Checking Definition 2.3.13, we immediately note, that (i) each column contains all elements of columns right of it. Also (ii) holds, namely that we have each element of [5] once as a diagonal element. After that it does not occur again in a column to the right. Condition (iii) can also be checked. Reading off of M, the edges of T_1 are:

$$E_1 = \left\{ \boxed{(1,2)}\,\boxed{(3,4)}\,\boxed{(2,3)}\,\boxed{(3,5)} \right\}. \tag{2.45}$$

This makes sense, if we have a look at 2. of Remark 2.3.4 and M:

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

Then the edges of T_2 are:

$$E_2 = \left\{ \boxed{(1,3;2)}\,\boxed{(2,4;3)}\,\boxed{(2,5;3)} \right\}. \tag{2.46}$$

Considering 3. of remark 2.3.4 and M, this is also correct:

$$M = \begin{pmatrix} \textcircled{1} & & & & \\ 5 & \textcircled{4} & & & \\ 4 & 5 & \textcircled{2} & & \\ \textcircled{3} & \textcircled{2} & \textcircled{5} & 3 & \\ \boxed{2} & \boxed{3} & \boxed{3} & 5 & 5 \end{pmatrix}.$$

By the same reasoning we get:

$$E_3 = \left\{ \boxed{(1,4;2,3)}\,\boxed{(4,5;2,3)} \right\}, \tag{2.47}$$

$$E_4 = \left\{ \boxed{(1,5;2,3,4)} \right\}, \tag{2.48}$$

which can be seen from M:

$$M = \begin{pmatrix} \textcircled{1} & & & & \\ 5 & \textcircled{4} & & & \\ \textcircled{4} & \textcircled{5} & 2 & & \\ \boxed{3} & \boxed{2} & 5 & 3 & \\ \boxed{2} & \boxed{3} & 3 & 5 & 5 \end{pmatrix} \quad \text{and} \quad M = \begin{pmatrix} \textcircled{1} & & & & \\ \textcircled{5} & 4 & & & \\ \boxed{4} & 5 & 2 & & \\ \boxed{3} & 2 & 5 & 3 & \\ \boxed{2} & 3 & 3 & 5 & 5 \end{pmatrix}.$$

For each column we marked the components of M belonging to the conditioning set with rectangles and the two conditioned nodes with circles. The R-vine tree sequence given by M consequently looks like Figure 5.

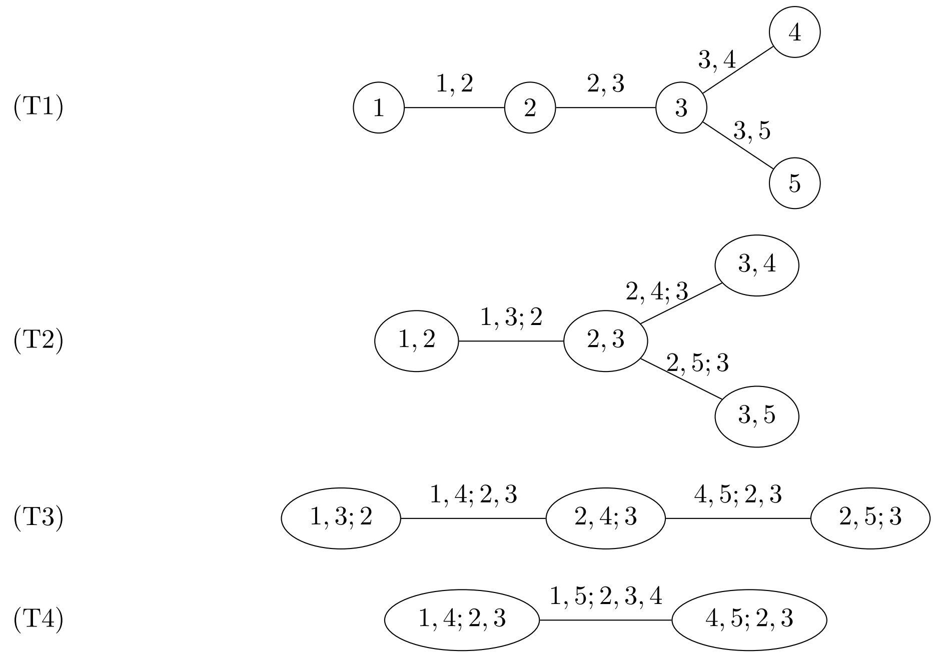

Figure 5: R-vine tree sequence corresponding to the R-vine matrix M.

Recall, that there are two sub-classes of R-vines, namely the C- and D-vines. Their special graphical structure is reflected in the structure of their R-vine matrix.

Remark 2.3.5. After reordering of the nodes, each C-vine on d elements can be represented by an R-vine matrix of the form:

$$M_C = \begin{pmatrix} 1 & & & & \\ 2 & 2 & & & \\ \vdots & \vdots & \ddots & & \\ d-1 & d-1 & \dots & d-1 & \\ d & d & \dots & d & d \end{pmatrix}. \tag{2.49}$$

The root node of T_1 is d, the root node of T_2 is $\big((d-1), d\big)$ and for $j \in \{3, ..., d-1\}$ the root node of T_j is $\big((d-j+1), (d-j+2)\ ;\ (d-j+3),\ ...,\ d\ \big)$.
Similarly, each D-vine on d elements can be represented by an R-vine matrix of the following form, if we reorder the nodes:

$$M_D = \begin{pmatrix} d & & & & & & \\ 1 & d-1 & & & & & \\ 2 & 1 & d-2 & & & & \\ 3 & 2 & 1 & d-3 & & & \\ \vdots & \vdots & \vdots & \ddots & \ddots & & \\ d-2 & d-3 & d-4 & \dots & 1 & 2 & \\ d-1 & d-2 & d-3 & d-4 & \dots & 1 & 1 \end{pmatrix}. \tag{2.50}$$

After introducing R-vine matrices, which capture $\mathcal{V}$, it remains to define *R-vine family matrices* and *R-vine parameter matrices* to fully describe the pair copulas in the pair copula construction. Together with the marginal distributions in $\boldsymbol{F}$ we then have a full representation of the R-vine specification $(\boldsymbol{F}, \mathcal{V}, B)$.
So let $(\boldsymbol{F}, \mathcal{V}, B)$ be an R-vine specification and M be an R-vine matrix for $\mathcal{V}$. For each pair copula $C_e \in B$ its family can be stored as an entry of a lower triangular family matrix. The location of this entry depends on the location of the edge e in $\mathcal{V}$. In the same manner each parameter determining C_e can be stored as an entry of a lower triangular parameter matrix. If C_e belongs to a copula family with $k > 1$ parameters $\theta_1, \dots, \theta_k$, then for each $j \in [k]$ the parameter θ_j will be stored in a separate parameter matrix P_j at a location again corresponding to the location of e in $\mathcal{V}$. In more detail:

Definition 2.3.14 (R-vine family matrix)**.** Let $(\boldsymbol{F}, \mathcal{V}, B)$ be an R-vine specification with M an R-vine matrix corresponding to $\mathcal{V}$. Let $F = (f_{ij})_{i,j \in [d]} \in \mathbb{R}^{d \times d}$ be a matrix with:

(i) $f_{ij} \in \{0\} \cup \{I, N, C, G, F, J, t, BB1, BB7, \dots\}$ for all $i, j \in [d]$,

(ii) $f_{ij} = 0$ for all $i \leq j \in [d]$,

(iii) f_{ij} gives the family of the pair copula C_e corresponding to the edge e given in M:

$$e = \{m_{j,j}, m_{i,j} \mid m_{(i+1),j}, \dots, m_{d,j}\} .$$

Then F is an *R-vine family matrix* for $(\boldsymbol{F}, \mathcal{V}, B)$.

Similarly to Definition 2.3.14 we define *R-vine parameter matrices.*

Definition 2.3.15 (R-vine parameter matrices)**.** Let $(\boldsymbol{F}, \mathcal{V}, B)$ be an R-vine specification. Let M be an R-vine matrix corresponding to $\mathcal{V}$ and F be the R-vine family matrix encoding the families of the pair copulas $C_e \in B$. Let:

$$l := \max\{\#\text{ parameters of } C_e \mid C_e \in B\}$$

be the maximal number of parameters among all families of pair copulas specified in F. Let $P_1 = (p_{ij}^{(1)})_{i,j \in [d]}, \dots, P_l = (p_{ij}^{(l)})_{i,j \in [d]} \in \mathbb{R}^{d \times d}$ be matrices with:

(i) $p_{ij}^{(k)} \in \mathbb{R}$ for all $i, j \in [d],\ k \in [l]$,

(ii) $p_{ij}^{(k)} = 0$ for all $i \leq j \in [d],\ k \in [l]$,

(iii) $p_{ij}^{(k)}$ gives the kth parameter of the pair copula C_e corresponding to the edge e given in M:

$$e = \{m_{j,j}, m_{i,j} \mid m_{(i+1),j}, \dots, m_{d,j}\} .$$

If this parameter does not exist, i.e. the family f_{ij} has less than k parameters, $p_{ij}^{(k)} = 0$.

Then $P_1, \dots, P_l$ are *R-vine parameter matrices* for $(\boldsymbol{F}, \mathcal{V}, B)$.

Example 2.3.3 (continued). To fully represent the pair copulas, that are given by the R-vine matrix M, we have a look at the corresponding R-vine family matrix and R-vine parameter matrices of M:

$$M = \begin{pmatrix} 1 & & & & \\ 5 & \textcircled{4} & & & \\ 4 & 5 & 2 & & \\ 3 & \textcircled{2} & 5 & 3 & \\ 2 & \boxed{3} & 3 & 5 & 5 \end{pmatrix}, \quad F = \begin{pmatrix} & & & & \\ I & & & & \\ N & I & & & \\ J & \textcircled{J} & I & & \\ G & C & BB1 & F & \end{pmatrix},$$

$$P_1 = \begin{pmatrix} & & & & \\ 0 & & & & \\ 0.45 & 0 & & & \\ 2.24 & \textcircled{1.72} & 0 & & \\ 1.58 & 3.69 & 1.47 & -5.71 & \end{pmatrix}, \quad P_2 = \begin{pmatrix} & & & & \\ 0 & & & & \\ 0 & 0 & & & \\ 0 & \textcircled{0} & 0 & & \\ 0 & 0 & 3.21 & 0 & \end{pmatrix}.$$

We notice, that the pair copula $(2, 4; 3)$ is of the copula type J, which stands for the Joe copula. This copula has one parameter with value 1.72. The pair copula of the edge $(2, 3)$ is of the family type $BB1$ and has two parameters 1.47 and 3.21. The second parameter is encoded in a second parameter matrix P_2 at the corresponding component.

Estimating Regular Vines

We estimate the joint distributions $F_{1\dots d}$ of the random vector $\boldsymbol{X} = (X_1, \dots, X_d) \in \mathbb{R}^d$ from data by estimating an R-vine specification $(\boldsymbol{F}, \mathcal{V}, B)$[4] from them. So let $(x_{i1}, \dots, x_{id})_{i\in[n]}$ be $n \in \mathbb{N}$ i.i.d. realizations of $\boldsymbol{X}$. Estimating $(\boldsymbol{F}, \mathcal{V}, B)$ we perform the following steps:

(1) estimate the marginal distributions $F_j(\cdot)$ of X_j with $j \in [d]$ stored in $\boldsymbol{F}$,

(2) select the R-vine tree sequence $\mathcal{V}$ and choose the pair copula families of the pairs of random variables corresponding to $\mathcal{V}$ and (if applicable) estimate their parameter(s). They are stored in B.

For **Step** (1) we can decide to non-parametrically estimate the univariate marginal distribution functions. In case we have prior information, e.g. on the parametric distribution family, this can also be done parametrically, i.e. estimating the respective parameter.

We introduce two non-parametric estimators of univariate distribution functions:

[4]We estimate one out of several R-vine specifications, which estimate $F_{1\dots d}$.

- **empirical distribution function:** For realizations x_{ij} with $i \in [n]$ of X_j it is defined as:

$$\hat{F}_j^{emp}(x) := \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{\{x \,|\, x_{ij} \leq x\}}(x) \; , \tag{2.51}$$

  where $\mathbb{1}(\cdot)$ denotes the indicator function.

- **kernel density estimator:** Let again x_{ij} with $i \in [n]$ be n realizations of X_j. A non-negative, symmetric function $K : \mathbb{R} \to \mathbb{R}$ with $\int_{-\infty}^{\infty} K(x)dx = 1$ is called a *kernel*. It could for example be:

$$K(x) := \begin{cases} \frac{1}{2} & \text{for} -1 \leq x \leq 1 \\ 0 & \text{else}\,, \end{cases} \; ,$$

  or any kind of probability density function. Rosenblatt (1956) first investigated in the *kernel density estimator* defined as:

$$\hat{F}_j^{kde}(x) := \frac{1}{n}\sum_{i=1}^{n} K(x - x_{ij}) \; . \tag{2.52}$$

Then we select the R-vine tree sequence $\mathcal{V}$ together with the pair copulas in B in **Step** (2). The tree sequence $\mathcal{V}$ determines of which two random variables the bivariate (conditional) dependence structure is modeled through a pair copula in B. "It is more important to model the dependence structure between random variables that have high dependencies correctly, because most copula families can model independence and the copulae distribution functions for parameters close to independence are very similar." This is one of the reasons Dissmann et al. (2013) give for constructing their *Dißmann's algorithm* in the following way: they use a maximum spanning tree algorithm to select the R-vine tree sequence $\mathcal{V}$ while maximizing the sum of the edge weights - the absolute Kendall's τ value of the two adjacent random variables[5]. The *Dißmann's algorithm* is given in Algorithm 1.

We want to outline lines 3 and 7 of Dißmann's Algorithm shown in Algorithm 1 more closely. This is, how in **Step** (2) the pair copula family is chosen and the corresponding parameters are estimated.

- **Selecting pair-copula families and estimating parameters:** More details to this can be found in Dissmann et al. (2013) and Czado (2019), on which the following is based on. For fitting in `R` we use the package `rvinecopulib` by Nagler and Vatter (2021).

  - We first consider the pair copulas corresponding to edges in tree T_1. Let $\mathcal{B}_e$ be the set of possible pair copula families to choose from for an edge $e =$

[5]For tree levels 2 to $d-1$ these are additionally conditioned on a set of random variables.

Algorithm 1: Dißmann's algorithm of Dissmann et al. (2013)

```
    Input: $n \in \mathbb{N}$ i.i.d. realizations of the random vector $(X_1, \dots, X_d)$, i.e.
           $(x_{i1}, \dots, x_{id})_{i \in [n]}$
    Output: $\mathcal{V}$ and $B$ of R-vine copula specification
1   Calculate the empirical Kendall's $\tau$ value $\hat{\tau}_{j,k}$ for all possible variable pairs $(j,k)$,
      $1 \leq j < k \leq d$.
2   Select the spanning tree that maximizes the sum of absolute empirical Kendalls's
      $\tau$ values, i.e.:
      $$T_1 = \underset{T=(V,E) \text{ in spanning tree}}{\arg\max} \sum_{e=(j,k) \in E} \left|\hat{\tau}_{j,k}\right| .$$
3   For each edge $(j,k)$ in the selected spanning tree, select a copula ad estimate the
      corresponding parameter(s). Then generate pseudo-observations
      $\hat{u}_{i,j\,|\,k} := \hat{F}_{j\,|\,k}(x_{ij}\,|\,x_{ik})$ and $\hat{u}_{i,k\,|\,j} := \hat{F}_{k\,|\,j}(x_{ik}\,|\,x_{ij})$, $i \in [n]$ using Equation (2.30)
      with the fitted copula $\hat{C}_{jk}$.
4   for $l \in \{2, \dots, d-1\}$ do
5       For all conditional variable pairs $(j,\, k\,;\, D)$ that can be part of tree $T_l$, i.e. all
          edges fulfilling the proximity condition (iii) of Definition 2.3.7: calculate the
          empirical Kendall's $\tau$ value $\hat{\tau}_{j,\,k\,;\,D}$ by applying the estimator of Definition
          2.2.6 to the pseudo-observations $\left(\hat{u}_{i,j\,|\,k \cup D}, \hat{u}_{i,k\,|\,j \cup D}\right)$. Denote these edges in
          the set $E_l^*$.
6       Among these edges, select the spanning tree that maximizes the sum of
          absolute empirical Kendall's $\tau$ values, i.e.:
          $$T_l = \underset{T=(V,E) \text{ in spanning tree with } E \subset E_l^*}{\arg\max} \sum_{e=(j,\,k\,;\,D) \in E} \left|\hat{\tau}_{j,\,k\,;\,D}\right| .$$
7       For each edge $(j,\, k\,;\, D)$ in the selected spanning tree $T_l$, select a conditional
          copula and estimate the corresponding parameter(s). Then generate
          pseudo-observations $\hat{u}_{i,j\,|\,k \cup D} := \hat{F}_{j\,|\,k \cup D}(x_{ij}\,|\,x_{ik}, x_{iD})$ and
          $\hat{u}_{i,k\,|\,j \cup D} := \hat{F}_{k\,|\,j \cup D}(x_{ik}\,|\,x_{ij}, x_{iD})$, $i \in [n]$ using Equation (2.30) with the fitted
          copula $\hat{C}_{jk;D}$.
8   end
```

$(j,k) \in E_1$. The set $\mathcal{B}_e$ can for example contain all Archimedean copulas given in **Bivariate Copula Families** on page 6, all bivariate copulas with at most two parameters or any other choice of parameteric pair copula families. In the function `vinecop` of the `R`-package `rvinecopulib` by Nagler and Vatter (2021), which we use for R-vine estimation, $\mathcal{B}_e$ is specified in the function parameter `family_set`. It allows for different family collections, such as `"onepar"` for pair copula families with only one parameter, `"elliptical"` for elliptical pair copula families or `"archimedean"` for Archimedean pair copula families.

– For each element $C^B \in \mathcal{B}_e$ we fit C^B with density c^B to the copula data

$(u_{i,j}, u_{i,k})_{i\in[n]}$ with $u_{i,j} := \hat{F}_j(x_{i,j})$ and $u_{i,j} := \hat{F}_j(x_{i,j})$, $i \in [n]$. Here $\hat{F}(\cdot)$ is for example one of the estimators of univeriate distribution functions introduced on page 27.

– Fitting each $C^B \in \mathcal{B}_e$ is equivalent to estimating the parameter(s) of each C^B. This can be done with maximum likelihood estimation (mle). The maximum likelihood estimator $\hat{\boldsymbol{\theta}}_{mle}$ maximizes the likelihood:

$$l(\boldsymbol{\theta}^B; \boldsymbol{u}) = \prod_{i=1}^{n} c^B(u_{i,j}, u_{i,k}; \boldsymbol{\theta}^B) \ .$$

In our analyses we always use maximum likelihood estimation for estimation parametric pair copulas.

For the pair copula families with a one-to-one relation between the copula parameter θ and Kendall's τ^B[6]:

$$\tau = k^B(\theta^B) \ ,$$

estimation through *inversion of Kendall's* τ can also be used:

$$\hat{\theta}^B_\tau := k^{B^{-1}}(\hat{\tau}) \ . \tag{2.53}$$

In (2.53) we use the empirical estimator $\hat{\tau}$ of τ defined in Definition 2.2.6.

In the `R`-function `vinecop` of the package `rvinecopulib` the parameter estimation method is specified in the parameter `par_method` as `"itau"` representing inversion of Kendall's τ or `"mle"` fpr maximum likelihood estimation. We use maximum likelihood estimation as default in our analyses.

– Then for each pair copula $C^B \in \mathcal{B}_e$ the corresponding AIC value, Akaike (1998), is calculated:

$$AIC(C^B, \hat{\boldsymbol{\theta}}^B; \boldsymbol{u}_e) := -2\sum_{i=1}^{n} ln\big(c^B(u_{i,j}, u_{i,k}; \hat{\boldsymbol{\theta}}^B)\big) + 2 \cdot k^B \ , \tag{2.54}$$

where k^B is the dimension of the parameter vector $\hat{\boldsymbol{\theta}}^B$ and $\boldsymbol{u}_e := (u_{i,j}, u_{i,k})_{i\in[n]}$.

– The pair copula C_e with parameter $\boldsymbol{\theta}_e$ is selected for edge e, which minimizes $AIC(C^B, \hat{\boldsymbol{\theta}}^B; \boldsymbol{u}_e)$. In `vinecop` the selection criterion can be specified in `selcrit` with `"aic"` as default.

– For an edge $e = (j, k\,;\, D) \in E_i$ of tree T_i with $i > 1$ we proceed similarly. Pseudo-observations are computed:

$$\hat{u}_{i,j\,|\,k\cup D} := \hat{F}_{j\,|\,k\cup D}(x_{ij}\,|\,x_{ik}, x_{iD}) \ ,$$
$$\hat{u}_{i,k\,|\,j\cup D} := \hat{F}_{k\,|\,j\cup D}(x_{ik}\,|\,x_{ij}, x_{iD}) \ ,$$

for $i \in [n]$ using Equation (2.30) with the fitted copula $\hat{C}_{jk;D}$. Applying the estimator of Definition 2.2.6 to these pseudo-observations, we obtain $\hat{\tau}_{j,k\,;\,D}$.

[6]These are all pair copulas with at most one parameter: independence, Gauss, Student's t, Clayton, Gumbel, Frank and Joe.

Again we can estimate the parameter(s) $\boldsymbol{\theta}^B$ of pair copula $C^B \in \mathcal{B}_e$ by inversion of Kendall's τ, if applicable, or by maximum likelihood estimation. We again select the pair copula C_e, which minimizes the AIC.

Instead of estimating the pair copulas completely parametrically in **Step** (2) as outlined above, we can also decide to estimate some of them non-parametrically, if this yields a better fit in terms of AIC of Equation 2.54. There the number of parameters k^B in Equation 2.54 is substituted with a non-parametric analogue, the *effective number of parameters*, Nagler (2014). An introduction to the effective number of parameters can be found in Friedman et al. (2001). The `R`-package `kdecopula` by Nagler (2016), which implements kernel density estimation for copulas, such as the transformation estimator, uses the definition of the effective number of parameters from Section 5.3.2. of Loader (2006).
To allow parametric as well as non-parametric pair copula estimation we have to augment $\mathcal{B}_e$ with a non-parametric pair copula. In the `R`-function `vinecop` of the package `rvinecopulib`, which we use for R-vine estimation, this is done by adding `"tll"` (Transformation Kernel) to `family_set` and setting the non-parametric estimation method in `nonpar_method` to either `"constant"`, `"linear"` or `"quadratic"`. In the following we illustrate, how a non-parametric pair copula can be estimated with the *transformation kernel estimator* corresponding to the choice `nonpar_method = "constant"` in the `vinecop`-function, which we use in our analyses.

- **Estimation of non-parametric bivariate copula - Transformation Kernels:** For details beyond this short introduction the reader is referred to Charpentier et al. (2007), Nagler (2014) and Nagler (2016), which the following is based on.

  Let $K : \mathbb{R} \to \mathbb{R}$ be a non-negative, symmetric function with $\int_{-\infty}^{\infty} K(x)dx = 1$. We call K a *kernel* and for now define it to be:

$$K(x) := \begin{cases} \frac{1}{2} & \text{for} -1 \leq x \leq 1 \\ 0 & \text{else}\,, \end{cases}$$

  In general K could be any probability density function. For $b > 0$ we define the notation:

$$K_b(x) := \frac{K(\frac{x}{b})}{b}\ .$$

  Given n samples $\{(x_{i1}, x_{i2}) \mid i \in [n]\}$ of the bivariate random vector (X_1, X_2) the bivariate *kernel density estimator* with bandwidth $b_n > 0$ is given by:

$$\hat{f}_n(y_1, y_2) = \frac{1}{n}\sum_{i=1}^{n} K_{b_n}(y_1 - x_{i1})K_{b_n}(y_2 - x_{i2})\ . \tag{2.55}$$

  In order to use this bivariate kernel density estimator to obtain a non-parametric pair-copula estimate, we need to overcome the boundedness of the support of the

copula, which is restricted to the unit cube. Therefore the copula data are transformed, such that their distribution is supported on $\mathbb{R}^2$. Then kernel estimation methods can be applied before the estimate is back-transformed to $[0,1]^2$.
Let (X_1, X_2) be a random vector with marginal distributions F_1 and F_2 and $(U_1, U_2) := (F_1(X_1), F_2(X_2))$ and G be a continuous distribution function on $\mathbb{R}$ with differentiable strictly positive density g. A common choice is $G = \Phi$, the normal distribution, which we also use in our analyses. We obtain a new random vector $(Z_1, Z_2) := (\Phi^{-1}(U_1), \Phi^{-1}(U_2))$, which has the density:

$$f(z_1, z_2) = \phi(z_1)\phi(z_2)\, c(\Phi(z_1), \Phi(z_2)) \,. \tag{2.56}$$

Given a sample $\{(u_{i1}, u_{i2}) \mid i \in [n]\}$ of (U_1, U_2), which can be transformed to a sample $\{(z_{i1}, z_{i2}) \mid (z_{i1}, z_{i2}) = (\Phi^{-1}(u_{i1}), \Phi^{-1}(u_{i2})),\ i \in [n]\}$, this joint density can be estimated with the kernel estimator:

$$\hat{f}_n(y_1, y_2) = \frac{1}{n} \sum_{i=1}^{n} K_{b_n}(y_1 - z_{i1}) K_{b_n}(y_2 - z_{i2}) \,.$$

If we then rearrange (2.56) and use the estimator $\hat{f}_n$ for f, then for any $(v_1, v_2) \in [0,1]^2$ we obtain the *transformation kernel estimator* of the copula density $c(v_1, v_2)$ with bandwidth b_n:

$$\hat{c}_n^{(T)}(v_1, v_2) = \frac{\sum_{i=1}^{n} K_{b_n} \Phi^{-1}(v_1) - \Phi^{-1}(u_{i1})) K_{b_n}(\Phi^{-1}(v_2) - \Phi^{-1}(u_{i2}))}{n\phi(\Phi^{-1}(v_1))\phi(\Phi^{-1}(v_2))} \tag{2.57}$$

We do not have use the same bandwidth b_n for both components of the random vector (U_1, U_2). Instead we can specify two different bandwidths: b_{1n} for U_1 and b_{2n} for U_2. Then we can store them in a diagonal matrix:

$$B_n := \begin{pmatrix} b_{1n} & * \\ * & b_{2n} \end{pmatrix} ,$$

with $\det(B_n) = b_{1n} b_{2n}$. With this and with setting $\boldsymbol{v} := (v_1, v_2)$ and $\boldsymbol{u}_i := (u_{i1}, u_{i2})$ we can rewrite the transformation kernel estimator:

$$\hat{c}_n^{(T')}(\boldsymbol{v}) = \frac{\sum_{i=1}^{n} K_{B_n}\big(\Phi^{-1}(\boldsymbol{v}) - \Phi^{-1}(\boldsymbol{u}_i)\big)}{n\phi(\Phi^{-1}(v_1))\phi(\Phi^{-1}(v_2))} ,$$

where Φ^{-1} is applied componentwise and short notation:

$$K_{B_n}\big(\Phi^{-1}(\boldsymbol{v}) - \Phi^{-1}(\boldsymbol{u}_i)\big) = \frac{K\Big(\big(B_n^{-1}[\Phi^{-1}(\boldsymbol{v}) - \Phi^{-1}(\boldsymbol{u}_i)]\big)_1\Big)\Big(\big(B_n^{-1}[\Phi^{-1}(\boldsymbol{v}) - \Phi^{-1}(\boldsymbol{u}_i)]\big)_2\Big)}{\det(B_n)} \,.$$

We even do not have to restrict ourselves to diagonal matrices but introduce a third bandwidth parameter b_{3n} below the diagonal:

$$B_n = \begin{pmatrix} b_{1n} & 0 \\ b_{3n} & b_{2n} \end{pmatrix}$$

such that $B_n B_n^T$ is a symmetric and positive definite matrix. Nagler (2014) call the resulting transformation estimator the *fully parametrized transformation estimator* with bandwidth matrix B_n.

How is this bandwidth matrix selected in practice? In the `vinecop`-function of their `R`-package `rvinecopulib` Nagler and Vatter (2021) select the bandwidth matrix B_n with a rule of thumb suggested in Nagler (2016) and introduced in Nagler (2014):

$$B_n = n^{-1/6} \hat{\Sigma}_{\boldsymbol{Z}}^{1/2} \ ,$$

with $\hat{\Sigma}_{\boldsymbol{Z}}$ the empirical covariance matrix of $\boldsymbol{Z} := \left(\Phi^{-1}(u_{i1}), \Phi^{-1}(u_{i2})\right)_{i \in [n]}$.

Besides the transformation kernel estimator a *local likelihood transformation estimator* could be used to estimate the pair copula. Nagler (2014) sketches the underlying idea in the following way: "The idea behind the local likelihood method [...] is to assume that the log-density $\log f(x, y)$ [...] can locally be approximated by a polynomial [...] of some order p." Its definition can be found in Definition 3.6. of Nagler (2014).
In the `vinecop`-function of `rvinecopulib` by Nagler and Vatter (2021) the local likelihood transformation estimator of order 1 (linear) corresponds to the option `nonpar_method = "linear"`, the local likelihood transformation estimator of order 2 (quadratic) to `nonpar_method = "quadratic"`.

Simulating from Regular Vines

Simulation algorithms can be found in Czado (2019).

Regular Vines with Discrete Components

So far we have only dealt with copulas and R-vines for *continuous* random vectors $\boldsymbol{X} = (X_1, \dots, X_d) \in \mathbb{R}^d$. Suppose now, that without loss of generality the last component of $\boldsymbol{X}$ takes on only discrete values: $X_d \in \{1, \dots, K\}$ with $K \in \mathbb{N}$. How would a pair copula construction in this mixed[7] case look like? We illustrate this with two examples:

Example 2.3.4. (2-dimensional mixed cases) Let $X_1 \in \mathbb{R}$ and $X_2 \in \{1, \dots, K\}$ be two random variables, X_1 being continuous and X_2 being discrete. For $x_1 \in \mathbb{R}$ and $x_2 \in \{1, \dots, K\}$ we compute the joint density $f_{12}(x_1, x_2)$. For clarity we color the discrete random variable X_2 and its corresponding observations with x_2 in purple.

[7]Using the term "mixed" we mean, that the random vector $\boldsymbol{X}$ consists of continuous and discrete random variables.

(i) In the first case we decompose:

$$f_{12}(x_1, x_2) = f_{1\,|\,2}(x_1 \,|\, x_2) \cdot f_2(x_2) \ ,$$

where $f_{1\,|\,2}(x_1 \,|\, X_2 = x_2)$ is abbreviated by $f_{1\,|\,2}(x_1 \,|\, x_2)$. We will use this notation in the subsequent. Therefore we compute the conditional distribution:

$$\begin{aligned} F_{1\,|\,2}(x_1 \,|\, x_2) &= \frac{P(X_1 \leq x_1, X_2 = x_2)}{P(X_2 = x_2)} \\ &= \frac{P(X_1 \leq x_1, X_2 \leq x_2) - P(X_1 \leq x_1, X_2 \leq x_2 - 1)}{P(X_2 \leq x_2) - P(X_2 \leq x_2 - 1)} \\ &= \frac{F_{1,2}(x_1, x_2) - F_{1,2}(x_1, x_2 - 1)}{F_2(x_2) - F_2(x_2 - 1)} \\ &\overset{(2.4)}{=} \frac{C_{1,2}(F_1(x_1), F_2(x_2)) - C_{1,2}(F_1(x_1), F_2(x_2 - 1))}{F_2(x_2) - F_2(x_2 - 1)} \ . \end{aligned} \tag{2.58}$$

Computing the corresponding probability mass function yields:

$$\begin{aligned} f_{1\,|\,2}(x_1 \,|\, x_2) &= \frac{\partial}{\partial x_1} F_{1\,|\,2}(x_1 \,|\, x_2) \\ &= \frac{\partial}{\partial x_1}\left[\frac{C_{1,2}(F_1(x_1), F_2(x_2)) - C_{1,2}(F_1(x_1), F_2(x_2 - 1))}{F_2(x_2) - F_2(x_2 - 1)}\right] \\ &= \frac{\frac{\partial}{\partial x_1} C_{1,2}(F_1(x_1), F_2(x_2)) - \frac{\partial}{\partial x_1} C_{1,2}(F_1(x_1), F_2(x_2 - 1))}{F_2(x_2) - F_2(x_2 - 1)} \\ &= \frac{\frac{\partial}{\partial F_1(x_1)} C_{1,2}(F_1(x_1), F_2(x_2)) - \frac{\partial}{\partial F_1(x_1)} C_{1,2}(F_1(x_1), F_2(x_2 - 1))}{F_2(x_2) - F_2(x_2 - 1)} \cdot \frac{\partial F_1(x_1)}{\partial x_1} \\ &\overset{2.2.9}{=} \frac{h_{2\,|\,1}(F_2(x_2) \,|\, F_1(x_1)) - h_{2\,|\,1}(F_2(x_2 - 1) \,|\, F_1(x_1))}{f_2(x_2)} \cdot f_1(x_1) \ . \end{aligned} \tag{2.59}$$

This gives in total:

$$\begin{aligned} f_{12}(x_1, x_2) &= f_{1\,|\,2}(x_1, \,|\, x_2) \cdot f_2(x_2) \\ &= \left[h_{2\,|\,1}(F_2(x_2) \,|\, F_1(x_1)) - h_{2\,|\,1}(F_2(x_2 - 1) \,|\, F_1(x_1))\right] \cdot f_1(x_1) \ . \end{aligned}$$

(ii) Obviously, choosing a different decomposition of the joint density $f_{12}(x_1, x_2)$ should not change the result, which we illustrate quickly. This time we decompose:

$$f_{12}(x_1, x_2) = f_{2\,|\,1}(x_2, \,|\, x_1) \cdot f_1(x_1) \ .$$

Therefore we compute:

$$
\begin{aligned}
F_{2\,|\,1}(x_2\,|\,x_1) &= \sum_{\substack{z\in\{1,\dots K\},\\ z\le x_2}} \frac{f_{12}(x_1,z)}{f_1(x_1)}\\
&= \sum_{\substack{z\in\{1,\dots K\},\\ z\le x_2}} \frac{\frac{\partial}{\partial x_1}F_{12}(x_1,z)-\frac{\partial}{\partial x_1}F_{12}(x_1,z-1)}{f_1(x_1)}\\
&= \frac{1}{f_1(x_1)}\left[\sum_{\substack{z\in\{1,\dots K\},\\ z\le x_2}} \frac{\partial}{\partial x_1}F_{12}(x_1,z) - \sum_{\substack{z\in\{1,\dots K\},\\ z\le x_2-1}} \frac{\partial}{\partial x_1}F_{12}(x_1,z)\right]\\
&= \frac{\frac{\partial}{\partial x_1}F_{12}(x_1,x_2)}{f_1(x_1)}\\
&\overset{(2.4)}{=} \frac{\frac{\partial}{\partial x_1}C_{12}(F_1(x_1),F_2(x_2))}{f_1(x_1)}\\
&= \frac{\frac{\partial}{\partial F_1(x_1)}C_{12}(F_1(x_1),F_2(x_2))\cdot\frac{\partial F_1(x_1)}{\partial x_1}}{f_1(x_1)}\\
&\overset{2.2.9}{=} \frac{h_{2\,|\,1}(F_2(x_2)\,|\,F_1(x_1))\cdot f_1(x_1)}{f_1(x_1)}\\
&= h_{2\,|\,1}(F_2(x_2)\,|\,F_1(x_1))\ . \qquad (2.60)
\end{aligned}
$$

The corresponding probability mass function is:

$$
\begin{aligned}
f_{2\,|\,1}(x_2\,|\,x_1) &= F_{2\,|\,1}(x_2\,|\,x_1) - F_{2\,|\,1}(x_2-1\,|\,x_1)\\
&= h_{2\,|\,1}(F_2(x_2)\,|\,F_1(x_1)) - h_{2\,|\,1}(F_2(x_2-1)\,|\,F_1(x_1))\ ,
\end{aligned}
$$

which in total again gives:

$$
\begin{aligned}
f_{12}(x_1,x_2) &= f_{2\,|\,1}(x_2,\,|\,x_1)\cdot f_1(x_1)\\
&= \left[h_{2\,|\,1}(F_2(x_2)\,|\,F_1(x_1)) - h_{2\,|\,1}(F_2(x_2-1)\,|\,F_1(x_1))\right]\cdot f_1(x_1)\ .
\end{aligned}
$$

Before we proceed with the 3-dimensional example, we introduce the following notation, which is based on Czado (2019):

Definition 2.3.16. (general notation of h-functions of bivariate copulas) For the bivariate copula $C_{ij;D}(u_i,u_j)$ in a simplified regular vine we use the notation:

$$
\begin{aligned}
h_{i\,|\,j;D}(u_i\,|\,u_j) &:= \frac{\partial}{\partial u_j}C_{ij;D}(u_i,u_j)\ ,\\
h_{j\,|\,i;D}(u_j\,|\,u_i) &:= \frac{\partial}{\partial u_i}C_{ij;D}(u_i,u_j)\ .
\end{aligned}
$$

We also give the following identity:

Lemma 2.3.1. For the bivariate distribution F_{12} of the continuous random variables $X_1, X_2 \in \mathbb{R}$ with corresponding copula distribution C_{12} and densities f_{12} and c_{12} it holds that:

$$f_{1\,|\,2}(x_1\,|\,x_2) = c_{12}(F_1(x_1), F_2(x_2))f_1(x_1) \tag{2.61}$$

$$\begin{aligned} F_{1\,|\,2}(x_1\,|\,x_2) &= \frac{\partial}{\partial u_2} C_{12}(F_1(x_1), u_2)\Big|_{u_2=F_2(x_2)} \\ &=: \frac{\partial}{\partial F_2(x_2)} C_{12}(F_1(x_1), F_2(x_2)) \end{aligned} \tag{2.62}$$

Proof. Following Czado (2019) we have:

$$\begin{aligned} f_{1\,|\,2}(x_1\,|\,x_2) &= \frac{f_{12}(x_1, x_2)}{f_2(x_2)} \overset{(2.5)}{=} \frac{c_{12}(F_1(x_1), F_2(x_2))f_1(x_1)f_2(x_2)}{f_2(x_2)} \\ &= \left[\frac{\partial^2 C_{12}(u_1, u_2)}{\partial u_1 \partial u_2} \cdot \frac{\partial u_1}{\partial x_1}\right]\Bigg|_{\substack{u_1=F_1(x_1)\\ u_2=F_2(x_2)}} \\ &= \frac{\partial}{\partial u_2}\left[\frac{\partial}{\partial x_1} C_{12}(F_1(x_1), u_2)\right]\Bigg|_{u_2=F_2(x_2)} \end{aligned}$$

$$\begin{aligned} F_{1\,|\,2}(x_1\,|\,x_2) &= \int_{-\infty}^{x_1} f_{1\,|\,2}(z\,|\,x_2)\,dz \\ &= \int_{-\infty}^{x_1} \frac{\partial}{\partial u_2}\left[\frac{\partial}{\partial x_1} C_{12}(F_1(z), u_2)\right]\Bigg|_{u_2=F_2(x_2)} dz \\ &= \frac{\partial}{\partial u_2}\left[\int_{-\infty}^{x_1} \frac{\partial}{\partial x_1} C_{12}(F_1(z), u_2)\,dz\right]\Bigg|_{u_2=F_2(x_2)} \\ &= \frac{\partial}{\partial u_2} C_{12}(F_1(x_1), u_2)\Big|_{u_2=F_2(x_2)} \end{aligned}$$

□

Example 2.3.5. (3-dimensional mixed cases) Let $(X_1, X_2, X_3) \in \mathbb{R}^2 \times \{1, \dots, K\}$ be a random vector and let $x_1, x_2 \in \mathbb{R},\ x_3 \in \{1, \dots, K\}$. As in Example 2.3.4 we will use the abbreviation $f_{i\,|\,D}(x_i\,|\,\boldsymbol{x}_D)$ for $f_{i\,|\,D}(x_i\,|\,\boldsymbol{X}_D = \boldsymbol{x}_D)$. For clarity we again color discrete X_3 and x_3 in purple.

(i) We can factorize the joint probability mass function in the following way:

$$f_{123}(x_1, x_2, x_3) = f_{3\,|\,12}(x_3\,|\,x_1, x_2) \cdot f_{1\,|\,2}(x_1\,|\,x_2) \cdot f_2(x_2)\ .$$

Therefore we derive:

$$\begin{aligned}
F_{3\,|\,12}(x_3\,|\,x_1,x_2) &= \sum_{\substack{z\in\{1,\dots K\},\\ z\le x_3}} \frac{f_{13\,|\,2}(x_1,z\,|\,x_2)}{f_{1\,|\,2}(x_1\,|\,x_2)} \\
&= \sum_{\substack{z\in\{1,\dots K\},\\ z\le x_3}} \frac{\frac{\partial}{\partial x_1}F_{13\,|\,2}(x_1,z\,|\,x_2) - \frac{\partial}{\partial x_1}F_{13\,|\,2}(x_1,z-1\,|\,x_2)}{f_{1\,|\,2}(x_1\,|\,x_2)} \\
&= \frac{1}{f_{1\,|\,2}(x_1\,|\,x_2)}\left[\sum_{\substack{z\in\{1,\dots K\},\\ z\le x_3}} \frac{\partial}{\partial x_1}F_{13\,|\,2}(x_1,z\,|\,x_2) - \sum_{\substack{z\in\{1,\dots K\},\\ z\le x_3-1}} \frac{\partial}{\partial x_1}F_{13\,|\,2}(x_1,z\,|\,x_2)\right] \\
&= \frac{\frac{\partial}{\partial x_1}F_{13\,|\,2}(x_1,x_3\,|\,x_2)}{f_{1\,|\,2}(x_1\,|\,x_2)} \\
&\overset{(2.4)}{=} \frac{\frac{\partial}{\partial x_1}C_{13;2}(F_{1\,|\,2}(x_1\,|\,x_2),F_{3\,|\,2}(x_3\,|\,x_2))}{f_{1\,|\,2}(x_1\,|\,x_2)} \\
&= \frac{\frac{\partial}{\partial F_{1\,|\,2}(x_1\,|\,x_2)}C_{13;2}(F_{1\,|\,2}(x_1\,|\,x_2),F_{3\,|\,2}(x_3\,|\,x_2))\cdot\frac{\partial F_{1\,|\,2}(x_1\,|\,x_2)}{\partial x_1}}{f_{1\,|\,2}(x_1\,|\,x_2)} \\
&\overset{2.3.16}{=} \frac{h_{3\,|\,1;2}(F_{3\,|\,2}(x_3\,|\,x_2)\,|\,F_{1\,|\,2}(x_1\,|\,x_2))\cdot f_{1\,|\,2}(x_1\,|\,x_2)}{f_{1\,|\,2}(x_1\,|\,x_2)} \\
&= h_{3\,|\,1;2}(F_{3\,|\,2}(x_3\,|\,x_2)\,|\,F_{1\,|\,2}(x_1\,|\,x_2))\;.
\end{aligned}$$

This yields the probability mass function:

$$\begin{aligned}
f_{3\,|\,12}(x_3\,|\,x_1,x_2) &= F_{3\,|\,12}(x_3\,|\,x_1,x_2) - F_{3\,|\,12}(x_3-1\,|\,x_1,x_2) \\
&= h_{3\,|\,1;2}(F_{3\,|\,2}(x_3\,|\,x_2)\,|\,F_{1\,|\,2}(x_1\,|\,x_2)) - h_{3\,|\,1;2}(F_{3\,|\,2}(x_3-1\,|\,x_2)\,|\,F_{1\,|\,2}(x_1\,|\,x_2))\;.
\end{aligned}$$

For continuous X_1 and X_2 we know by Lemma 2.2.9 that:

$$f_{1\,|\,2}(x_1,\,|\,x_2) = c_{12}(F_1(x_1),F_2(x_2))\cdot f_1(x_1)\;,$$

which gives in total:

$$\begin{aligned}
f_{123}(x_1,x_2,x_3) &= f_{3\,|\,12}(x_3\,|\,x_1,x_2)\cdot f_{1\,|\,2}(x_1,\,|\,x_2)\cdot f_2(x_2) \\
&= \left[h_{3\,|\,1;2}(F_{3\,|\,2}(x_3\,|\,x_2)\,|\,F_{1\,|\,2}(x_1\,|\,x_2)) - h_{3\,|\,1;2}(F_{3\,|\,2}(x_3-1\,|\,x_2)\,|\,F_{1\,|\,2}(x_1\,|\,x_2))\right] \\
&\quad\cdot c_{12}(F_1(x_1),F_2(x_2))\cdot f_1(x_1)\cdot f_2(x_2)\;.
\end{aligned}$$

Here we can re-express $F_{1\,|\,2}(x_1\,|\,x_2)$, $F_{3\,|\,2}(x_3\,|\,x_2)$ and $F_{3\,|\,2}(x_3-1\,|\,x_2)$ as h-functions with the help of Lemma 2.3.1 for continuous X_1 and X_2 and (2.60):

$$\begin{aligned}
F_{1\,|\,2}(x_1\,|\,x_2) &= \frac{\partial}{\partial F_2(x_2)}C_{12}(F_1(x_1),F_2(x_2)) \\
&\overset{2.2.9}{=} h_{1\,|\,2}(F_1(x_1)\,|\,F_2(x_2))\;, \\
F_{3\,|\,2}(x_3\,|\,x_2) &\overset{(2.60)}{=} h_{3\,|\,2}(F_3(x_3)\,|\,F_2(x_2))\;.
\end{aligned}$$

Note, that we can reorder continuous X_1 and X_2 as we wish.

(ii) This time we factorize the joint probability mass function in the following way:

$$f_{123}(x_1, x_2, x_3) = f_{1\,|\,23}(x_1\,|\,x_2, x_3) \cdot f_{3\,|\,2}(x_3\,|\,x_2) \cdot f_2(x_2)\ .$$

Therefore we derive:

$$\begin{aligned}
F_{1\,|\,23}(x_1\,|\,x_2, x_3) &= \frac{P(X_1 \le x_1, X_3 = x_3\,|\,X_2 = x_2)}{P(X_3 = x_3\,|\,X_2 = x_2)} \\
&= \frac{F_{13\,|\,2}(x_1, x_3\,|\,x_2) - F_{13\,|\,2}(x_1, x_3 - 1\,|\,x_2)}{f_{3\,|\,2}(x_3\,|\,x_2)} \\
&\overset{(2.4)}{=} \frac{C_{13;2}(F_{1\,|\,2}(x_1\,|\,x_2), F_{3\,|\,2}(x_3\,|\,x_2)) - C_{13;2}(F_{1\,|\,2}(x_1\,|\,x_2), F_{3\,|\,2}(x_3 - 1\,|\,x_2))}{f_{3\,|\,2}(x_3\,|\,x_2)}\ .
\end{aligned}$$

The probability mass function is:

$$\begin{aligned}
f_{1\,|\,23}(x_1\,|\,x_2, x_3) &= \frac{\partial}{\partial x_1} F_{1\,|\,23}(x_1\,|\,x_2, x_3) \\
&= \frac{\frac{\partial}{\partial x_1}\left[C_{13;2}(F_{1\,|\,2}(x_1\,|\,x_2), F_{3\,|\,2}(x_3\,|\,x_2)) - C_{13;2}(F_{1\,|\,2}(x_1\,|\,x_2), F_{3\,|\,2}(x_3 - 1\,|\,x_2))\right]}{f_{3\,|\,2}(x_3\,|\,x_2)} \\
&= \frac{\partial}{\partial F_{1\,|\,2}(x_1\,|\,x_2)}\big[C_{13;2}(F_{1\,|\,2}(x_1\,|\,x_2), F_{3\,|\,2}(x_3\,|\,x_2)) \\
&\qquad - C_{13;2}(F_{1\,|\,2}(x_1\,|\,x_2), F_{3\,|\,2}(x_3 - 1\,|\,x_2))\big] \cdot \frac{\partial F_{1\,|\,2}(x_1\,|\,x_2)}{\partial x_1} \cdot \frac{1}{f_{3\,|\,2}(x_3\,|\,x_2)} \\
&\overset{2.3.16}{=} \frac{h_{3\,|\,1;2}(F_{3\,|\,2}(x_3\,|\,x_2)\,|\,F_{1\,|\,2}(x_1\,|\,x_2)) - h_{3\,|\,1;2}(F_{3\,|\,2}(x_3 - 1\,|\,x_2)\,|\,F_{1\,|\,2}(x_1\,|\,x_2))}{f_{3\,|\,2}(x_3\,|\,x_2)} \\
&\qquad \cdot f_{1\,|\,2}(x_1\,|\,x_2)\ .
\end{aligned}$$

This gives in total:

$$\begin{aligned}
f_{123}(x_1, x_2, x_3) &= f_{1\,|\,23}(x_1\,|\,x_2, x_3) \cdot f_{3\,|\,2}(x_3\,|\,x_2) \cdot f_2(x_2) \\
&= \frac{h_{3\,|\,1;2}(F_{3\,|\,2}(x_3\,|\,x_2)\,|\,F_{1\,|\,2}(x_1\,|\,x_2)) - h_{3\,|\,1;2}(F_{3\,|\,2}(x_3 - 1\,|\,x_2)\,|\,F_{1\,|\,2}(x_1\,|\,x_2))}{f_{3\,|\,2}(x_3\,|\,x_2)} \\
&\quad \cdot f_{1\,|\,2}(x_1\,|\,x_2) \cdot f_{3\,|\,2}(x_3\,|\,x_2) \cdot f_2(x_2) \\
&\overset{2.3.1}{=} \left[h_{3\,|\,1;2}(F_{3\,|\,2}(x_3\,|\,x_2)\,|\,F_{1\,|\,2}(x_1\,|\,x_2)) - h_{3\,|\,1;2}(F_{3\,|\,2}(x_3 - 1\,|\,x_2)\,|\,F_{1\,|\,2}(x_1\,|\,x_2))\right] \\
&\quad c_{12}(F_1(x_1), F_2(x_2)) \cdot f_1(x_1) \cdot f_2(x_2)\ .
\end{aligned}$$

Again $F_{3\,|\,2}(x_3\,|\,x_2)$, $F_{3\,|\,2}(x_3 - 1\,|\,x_2)$ and $F_{1\,|\,2}(x_1\,|\,x_2)$ can be re-expressed as h-functions as above.

(iii) Finally, we decompose the joint probability mass function in the following way:

$$f_{123}(x_1, x_2, x_3) = f_{1\,|\,23}(x_1\,|\,x_2, x_3) \cdot f_{2\,|\,3}(x_2\,|\,x_3) \cdot f_3(x_3)\ .$$

That is why we compute:

$$\begin{aligned}
F_{1\,|\,23}(x_1\,|\,x_2,x_3) &= P(X_1 \le x_1\,|\,X_2=x_2, X_3=x_3)\\
&= \frac{\frac{\partial}{\partial x_2}F_{12\,|\,3}(x_1,x_2\,|\,x_3)}{f_{2\,|\,3}(x_2\,|\,x_3)}\\
&\overset{(2.4)}{=} \frac{\frac{\partial}{\partial x_2}C_{12;3}(F_{1\,|\,3}(x_1\,|\,x_3),F_{2\,|\,3}(x_2\,|\,x_3))}{f_{2\,|\,3}(x_2\,|\,x_3)}\\
&= \frac{\frac{\partial}{\partial F_{2\,|\,3}(x_2\,|\,x_3)}C_{12;3}(F_{1\,|\,3}(x_1\,|\,x_3),F_{2\,|\,3}(x_2\,|\,x_3))}{f_{2\,|\,3}(x_2\,|\,x_3)} \cdot \frac{\partial F_{2\,|\,3}(x_2\,|\,x_3)}{\partial x_2}\\
&= \frac{\partial}{\partial F_{2\,|\,3}(x_2\,|\,x_3)}C_{12;3}(F_{1\,|\,3}(x_1\,|\,x_3),F_{2\,|\,3}(x_2\,|\,x_3))\\
&\overset{2.3.16}{=} h_{1\,|\,2;3}(F_{1\,|\,3}(x_1\,|\,x_3)\,|\,F_{2\,|\,3}(x_2\,|\,x_3))
\end{aligned}$$

Hence:

$$\begin{aligned}
f_{1\,|\,23}(x_1\,|\,x_2,x_3) &= \frac{\partial}{\partial x_1}F_{1\,|\,23}(x_1\,|\,x_2,x_3)\\
&= \frac{\partial}{\partial x_1}\left[\frac{\partial}{\partial F_{2\,|\,3}(x_2\,|\,x_3)}C_{12;3}(F_{1\,|\,3}(x_1\,|\,x_3),F_{2\,|\,3}(x_2\,|\,x_3))\right]\\
&= \frac{\partial^2}{\partial F_{1\,|\,3}(x_1\,|\,x_3)\partial F_{2\,|\,3}(x_2\,|\,x_3)}C_{12;3}(F_{1\,|\,3}(x_1\,|\,x_3),F_{2\,|\,3}(x_2\,|\,x_3))\\
&\quad \cdot \frac{\partial F_{1\,|\,3}(x_1\,|\,x_3)}{\partial x_1}\\
&= c_{12;3}(F_{1\,|\,3}(x_1\,|\,x_3),F_{2\,|\,3}(x_2\,|\,x_3)) \cdot f_{1\,|\,3}(x_1\,|\,x_3)\ .
\end{aligned}$$

Together with (2.59) this gives:

$$\begin{aligned}
f_{123}(x_1,x_2,x_3) &= c_{12;3}(F_{1\,|\,3}(x_1\,|\,x_3),F_{2\,|\,3}(x_2\,|\,x_3)) \cdot f_{1\,|\,3}(x_1\,|\,x_3)\\
&\quad \cdot f_{2\,|\,3}(x_2,\,|\,x_3) \cdot f_3(x_3)\\
&\overset{(2.59)}{=} c_{12;3}(F_{1\,|\,3}(x_1\,|\,x_3),F_{2\,|\,3}(x_2\,|\,x_3))\\
&\quad \cdot \frac{h_{3\,|\,1}(F_3(x_3)\,|\,F_1(x_1)) - h_{3\,|\,1}(F_3(x_3-1)\,|\,F_1(x_1))}{f_3(x_3)} \cdot f_1(x_1)\\
&\quad \cdot \frac{h_{3\,|\,2}(F_3(x_3)\,|\,F_2(x_2)) - h_{3\,|\,2}(F_3(x_3-1)\,|\,F_2(x_2))}{f_3(x_3)} \cdot f_2(x_2)\\
&\quad \cdot f_3(x_3)\\
&= c_{12;3}(F_{1\,|\,3}(x_1\,|\,x_3),F_{2\,|\,3}(x_2\,|\,x_3)) \cdot \frac{f_1(x_1)\cdot f_2(x_2)}{f_3(x_3)}\\
&\quad \cdot \left[h_{3\,|\,1}(F_3(x_3)\,|\,F_1(x_1)) - h_{3\,|\,1}(F_3(x_3-1)\,|\,F_1(x_1))\right]\\
&\quad \cdot \left[h_{3\,|\,2}(F_3(x_3)\,|\,F_2(x_2)) - h_{3\,|\,2}(F_3(x_3-1)\,|\,F_2(x_2))\right]\ ,
\end{aligned}$$

where with the help of (2.58) we could re-express the conditional distributions $F_{1\,|\,3}(x_1\,|\,x_3)$ and $F_{2\,|\,3}(x_2\,|\,x_3)$.

A few remarks here:

Remark 2.3.6. (a) Genest and Nešlehová (2007) illustrate, that Sklar's Theorem 2.2.1 still guarantees the existence of a copula representation as is (2.4) of the bivariate distribution $F_{12}(X_1, X_2)$ for the case, that at least one random variables is discrete:

$$F_{12}(X_1, X_2) = C_{12}(F_1(X_1), F_2(X_2)) \ . \tag{2.63}$$

Due to jumps in the distribution function of a discrete random variable and plateaus in their corresponding inverse, the copula in (2.63) is no longer unique on the entire unit cube, but unique on the product of the ranges of the marginal distribution functions.

(b) In Example 2.3.5 we have seen, how the joint probability mass function of the 3-dimensional random vector $(X_1, X_2, X_3) \in \mathbb{R}^2 \times \{1, \ldots, K\}$ containing one discrete random variable X_3 can be factorized.

(c) Again we notice, that we can decompose the joint probability mass function of a mixed random vector into pair copulas and marginal distributions. Examples 2.3.4 and 2.3.5 already give an idea, how the decomposition of the mixed joint probability mass function can be generalized to higher dimensions. Section 3 of Schallhorn (2017) illustrates this in detail. This motivates fitting an R-vine in a similar way as in Section 2.3.

(d) In Examples 2.3.4 and 2.3.5 we gave the decomposition of the joint probability mass function of a mixed random vector. Panagiotelis et al. (2012) introduce pair copula constructions for a (purely) discrete random vector:

$$\boldsymbol{X} = (X_1, \ldots, x_d) \in \{1, \ldots, K\}^d \ .$$

For $(x_1, \ldots, x_d) \in \{1, \ldots, K\}^d$ the joint probability mass function of $\boldsymbol{X}$ can be factorized as:

$$\begin{aligned} P(X_1 = x_1, \ldots, X_d = x_d) = P(&X_1 = x_1 \,|\, X_2 = x_2, \ldots, X_d = x_d) \\ &\cdot P(X_2 = x_2 \,|\, X_3 = x_3, \ldots, X_d = x_d) \cdot \ldots \\ &\cdot P(X_d = x_d) \ . \end{aligned}$$

Here each factor can be re-expressed with the discrete analogue of (2.30), which Panagiotelis et al. (2012) give as:

$$\begin{aligned} &P(X_j = x_j \,|\, \boldsymbol{V} = \boldsymbol{v}) \\ &= \frac{P(X_j = x_j, V_h = v_h \,|\, \boldsymbol{V}_{-h} = \boldsymbol{v}_{-h})}{P(V_h = v_h \,|\, \boldsymbol{V}_{-h} = \boldsymbol{v}_{-h})} \\ &= \frac{1}{P(V_h = v_h \,|\, \boldsymbol{V}_{-h} = \boldsymbol{v}_{-h})} \cdot \Big\{ \sum_{i_j=0,1} \sum_{i_h=0,1} (-1)^{i_j + i_h} \\ &\qquad \cdot P(X_j \leq x_j - i_j, V_h \leq v_h - i_h \,|\, \boldsymbol{V}_{-h} = \boldsymbol{v}_{-h}) \Big\} \ , \end{aligned} \tag{2.64}$$

where $\boldsymbol{V}$ is a sub-vector of $\boldsymbol{X}$. With Sklar's Theorem 2.2.1 this yields:

$$\begin{aligned}
&P(X_j = x_j \,|\, \boldsymbol{V} = \boldsymbol{v}) \\
&= \frac{1}{P(V_h = v_h \,|\, \boldsymbol{V}_{-h} = \boldsymbol{v}_{-h})} \cdot \Big\{ \sum_{i_j=0,1} \sum_{i_h=0,1} (-1)^{i_j+i_h} \\
&\quad \cdot C_{X_j,V_h;\boldsymbol{V}_{-h}}(F_{X_j \,|\, \boldsymbol{V}_{-h}}(x_j - i_j \,|\, \boldsymbol{V}_{-h} = \boldsymbol{v}_{-h}), F_{V_h \,|\, \boldsymbol{V}_{-h}}(v_h - i_h \,|\, \boldsymbol{V}_{-h} = \boldsymbol{v}_{-h}))\Big\} \,,
\end{aligned}$$

where the arguments of the copula can be expressed with the help of the following identity:

$$\begin{aligned}
&F_{X_j \,|\, V_h, \boldsymbol{V}_{-h}}(x_j \,|\, V_h = v_h, \boldsymbol{V}_{-h} = \boldsymbol{v}_{-h}) \\
&= \Big[C_{X_j,V_h;\boldsymbol{V}_{-h}}(F_{X_j \,|\, \boldsymbol{V}_{-h}}(x_j \,|\, \boldsymbol{V}_{-h} = \boldsymbol{v}_{-h}), F_{V_h \,|\, \boldsymbol{V}_{-h}}(v_h \,|\, \boldsymbol{V}_{-h} = \boldsymbol{v}_{-h})) \\
&\quad - C_{X_j,V_h;\boldsymbol{V}_{-h}}(F_{X_j \,|\, \boldsymbol{V}_{-h}}(x_j \,|\, \boldsymbol{V}_{-h} = \boldsymbol{v}_{-h}), F_{V_h \,|\, \boldsymbol{V}_{-h}}(v_h - 1 \,|\, \boldsymbol{V}_{-h} = \boldsymbol{v}_{-h}))\Big] \\
&\quad \cdot \frac{1}{P(V_k = v_k \,|\, \boldsymbol{V}_{-\{h,k\}})} \,.
\end{aligned}$$

Here $k \neq h$ indexes another component of the sub-vector $\boldsymbol{V}$.

As mentioned in (c) we want to estimate an R-vine specification $(\boldsymbol{F}, \mathcal{V}, B)$, that the joint distributions $F_{1 \ldots d}$ of the mixed random vector $\boldsymbol{X} = (X_1, \ldots, X_d)$ realizes, from $n \in \mathbb{N}$ i.i.d. realizations of $\boldsymbol{X}$, i.e. $(x_{i1}, \ldots, x_{id})_{i \in [n]}$.
Panagiotelis et al. (2017) propose two R-vine selection algorithms with only parametric pair copula families, that adapt Dißmann's Algorithm 1 to the discrete case. They use maximum likelihood estimation and an in-sample modified version of the AIC[8] by Akaike (1998) or an out-of-sample log predictive score based on Gneiting and Raftery (2007), respectively, as edge weights.
In the R-package `rvinecopulib` by Nagler and Vatter (2021), which we will later use for our analysis, Dißmann's Algorithm 1 is used with edge weights "corrected for ties". This means in practice, that the estimator (2.17) of Definition **??** is used in lines 1 and 5 of Dißmann's Algorithm 1.

2.4 Sample Quantiles

Definition 2.4.1 (sample quantiles)**.** The quantile function $Q : (0,1) \to \mathbb{R}$ of a distribution F is defined to be:

$$Q(p) = F^{-1}(p) := \inf\{x : F(x) \geq p\} \,.$$

Remark 2.4.1. By Hyndman and Fan (1996) the sample quantiles $\hat{Q}(p)$ for independent observations $\{X_1, \ldots X_n\}$ of F with order statistics $\{X_{(1)}, \ldots, X_{(n)}\}$, that are commonly used in statistical packages, are of the form:

$$\hat{Q}(p) := (1-\gamma)\, X_{(j)} + \gamma\, X_{(j+1)} \tag{2.65}$$

[8]In case of the in-sample modified version of the AIC as edge weight, a minimum instead of a maximum spanning tree search is conducted.

with $0 \leq \gamma < 1$ a function of j and g and:

$$\begin{aligned} \frac{j-m}{n} &\leq p < \frac{j-m+1}{n} \\ j &= \lfloor pn + m \rfloor \\ g &= pn + m - j \ , \end{aligned}$$

where $\lfloor x \rfloor := max\{k \in \mathbb{Z} \mid k \leq x\}$. The sample quantiles of type 1 of `stats::quantile` correspond to the inverse of the empirical distribution function with $m = 0$ and:

$$\gamma = \begin{cases} 1 \text{ for } g = pn - j > 0 \\ 0 \text{ for } g = 0 \end{cases} \quad .$$

The sample quantiles of type 8 of `stats::quantile` are derived on the basis of the assumption:

$$p_k = median\big(F(X_{(k)})\big)$$

with $p_k \in [0,1]$ an observed value. Hyndman and Fan (1996) use the approximation:

$$median\big(F(X_{(k)})\big) \approx \frac{k - \frac{1}{3}}{n + \frac{1}{3}} \ .$$

This gives sample quantiles of the form (2.65) with:

$$\gamma = g \ , \quad m = \frac{1}{3}(p+1)$$

and the remaining constants derived from m as stated above. Sample quantiles of type 8 are considered as the best choice among the commonly used sample quantile definitions, as they give (approximately) median-unbiased estimates of $Q(p)$ regardless of the distribution F. For ordinal random variables the sample quantiles of type 1 are a sensible choice.

3 An Introduction to Classification

A *classification task* - as the name already suggests - is the task of deciding which class, out of a set of known classes, a previously unknown observation (rather) belongs to. This decision is executed by a function, which is called *classifier*.
So let $\boldsymbol{X} \in \mathbb{R}^d$ be a random vector and $Y \in G$ be the random variable indicating the class an observation of $\boldsymbol{X}$ belongs to with G the known set of possible classes. Let then $\boldsymbol{x}_i \in \mathbb{R}^d,\ i \in [n]$ be n realizations of $\boldsymbol{X}$ with $n \in \mathbb{N}$ and d different feature values $x_{ij} \in \mathbb{R},\ j \in [d]$.
The discrimination of the observations $\boldsymbol{x}_i \in \mathbb{R}^d,\ i \in [n]$ by the classes in G (in the best case) makes sense[9] and each respective true class membership y_i can (manually and with some effort) be checked. However, we do not want to assign the class y_i of $\boldsymbol{x}_i$ by hand but, instead we perform a classification to let the class y_i of $\boldsymbol{x}_i$ be estimated by the classifier f:

$$(\boldsymbol{X}, Y) = (\boldsymbol{x}_i, y_i)\ ,$$

$$f : \mathbb{R}^d \to G\,, \quad f(\boldsymbol{x}_i) = \hat{y}_i\ .$$

We do this based on the assumption, that a set of rules, which underlies the data $(\boldsymbol{x}_i^T\ y_i)_{i\in[n]}$, determines the true class y_i of each observation $\boldsymbol{x}_i$. We assume, that the classifier f can learn these rules from data, such that ideally the estimated label $\hat{y}_i := f(\boldsymbol{x}_i)$ is equal to the true label y_i or at least is as "close" as possible to the truth:

$$\begin{aligned} \text{estimated label :} &\quad \hat{y}_i := f(\boldsymbol{x}_i) \\ \text{true label :} &\quad y_i \end{aligned}$$

This "closeness" is measured by a so-called *loss function* $L(Y, f(\boldsymbol{X})) \geq 0$, which penalizes prediction errors of f. Two common choices of a loss function for classification are:

$$L(Y, f(\boldsymbol{X})) = \begin{cases} \mathbb{1}\{Y \neq f(\boldsymbol{X})\} & \text{0-1 loss,} \\ -2\,\log P(Y = y \,|\, \boldsymbol{X}) & -2\times \text{ log-likelihood, deviance.} \end{cases} \tag{3.1}$$

Here $\mathbb{1}(\cdot)$ denotes the indicator function. For a set A it is defined as:

$$\mathbb{1}_A(x) := \begin{cases} 1\,, & \text{for } x \in A \\ 0\,, & \text{else} \end{cases} \quad ,$$

and, if clear from the context, sometimes abbreviated by $\mathbb{1}\{x \in A\}$. For our analysis we use the deviance loss as default.
For building this set of rules, the classifier f is *trained* on a data set consisting of $m \in \mathbb{N}$, $m < n$ observations $\boldsymbol{x}_i \in \mathbb{R}^d$ and their respective true class label $y_i \in G$ with $i \in T \subset [n]$, i.e. through *learning by example.* For this, the data $(\boldsymbol{x}_i^T\ y_i)_{i\in[n]}$ are split into a training and a test set, where observation $(\boldsymbol{x}_i^T\ y_i)$ belong to the training set, if $i \in T$.[10]

[9]The discrimination of observations by class labels, which were randomly assigned to them, is regarded as not sensible: there is no discriminating feature.
[10]We denote $\boldsymbol{x}_j^{train} := \boldsymbol{x}_i$ and $y_j^{train} := y_i$ where i is the jth element of T arranged in ascending order.

To choose f we minimize the expected prediction error (EPE), quantified by the loss $L(Y, f(\boldsymbol{X}))$:

$$EPE(f) = E[L(Y, f(\boldsymbol{X}))] \ ,$$

which, by conditioning, can be rewritten as:

$$EPE(f) = E_{\boldsymbol{X}}\big[E_{Y|\boldsymbol{X}}[L(Y, f(\boldsymbol{X}))|\boldsymbol{X}]\big] = E_{\boldsymbol{X}}\Big[\sum_{y \in G} L(y, f(\boldsymbol{X})P(y \,|\, \boldsymbol{X})\Big] \ .$$

By pointwise minimization we obtain:

$$\hat{f}(\boldsymbol{x}) = \ \mathrm{argmin}_{k \in G} \sum_{y \in G} L(y, k) P(y|\boldsymbol{X} = \boldsymbol{x}) \ , \tag{3.2}$$

and the solution depends on the chosen loss function $L(\cdot, \cdot)$. If we take the 0-1 loss function of Equation (3.1), Equation (3.2) is equal to:

$$\hat{f}(\boldsymbol{x}) = \ \mathrm{argmin}_{k \in G}[1 - P(k|\boldsymbol{X} = \boldsymbol{x})] \ ,$$

which gives:

$$\hat{f}(\boldsymbol{x}) = y \quad \text{if} \quad P(y|\boldsymbol{X} = \boldsymbol{x}) = \max_{k \in G} P(y|\boldsymbol{X} = \boldsymbol{x}) \ .$$

The classifier $\hat{f}(\cdot)$ of Equation (3.2) is estimated from the training data:

$$\big(x \ \boldsymbol{y}\big)_{train} := \begin{pmatrix} \boldsymbol{x}_1^{T,\, train} & y_1^{train} \\ \vdots & \vdots \\ \boldsymbol{x}_m^{T,\, train} & y_m^{train} \end{pmatrix} := \big(\boldsymbol{x}_i^T \quad y_i\big)_{i \in T} \in \mathbb{R}^{m \times (d+1)} \ . \tag{3.3}$$

This form of learning is called *supervised learning.*

So how do we do the pointwise minimization and estimation in Equation (3.2)? We first need to decide on a loss function. Apparently, the pointwise minimum in Equation (3.2) depends on the model $F : \mathbb{R}^d \to [0, 1]$ we use to estimate the conditional probability $P(y|\boldsymbol{X} = \boldsymbol{x})$ from the training data. It would be ideal, but in practice it is not possible to minimize over all models F without setting up further constraints. Possible model classes F are, see Czado and Brechmann (2021):

- $P(y|\boldsymbol{X} = \boldsymbol{x}) = F(\boldsymbol{x}^T\boldsymbol{\beta})$ with F a known cumulative distribution function, such as the standard normal distribution, see page 141:

  $$F(\boldsymbol{x}^T\boldsymbol{\beta}) = \Phi(\boldsymbol{x}^T\boldsymbol{\beta}; R)$$

  with density

  $$\phi(\boldsymbol{x}^T\boldsymbol{\beta}; R) := \frac{1}{(2\pi)^{d/2}}|R|^{-1/2} \exp\Big\{ -\frac{1}{2}(\boldsymbol{x}^T\boldsymbol{\beta})^T R^{-1} \boldsymbol{x}^T\boldsymbol{\beta}\Big\} \ ,$$

- $P(y|\boldsymbol{X} = \boldsymbol{x}) = F(\boldsymbol{x}^T\boldsymbol{\beta})$ with F a known function class, such as:

$$F(\boldsymbol{x}^T\boldsymbol{\beta}) = \frac{\exp(\boldsymbol{x}^T\boldsymbol{\beta})}{1 + \exp(\boldsymbol{x}^T\boldsymbol{\beta})} \,,$$

- $P(y|\boldsymbol{X} = \boldsymbol{x}) = F(\boldsymbol{x};\boldsymbol{\theta})$ with $F : \mathbb{R}^d \times \Theta^q \to [0,1]$ with $\Theta^q \subset \mathbb{R}^q, q \in \mathbb{N}$, with $\boldsymbol{\theta}$ parameters of an iterative and/or graphical method

Then Equation (3.2) can be specified for the chosen model class and optimized with regard to the respective parameter(s) of the model class. This can be either solved explicitly or with a numerical procedure on the training data.
Summarized, depending on the class of f or the constraints put on f and the loss function $L(Y, f(\boldsymbol{X}))$ used we arrive at different types of classifiers.

Imagine for example you are blind-folded and handed a fruit bowl. Your task is to name the type of fruit by touching each fruit with your hands. Before being blind-folded you were able to inspect another fruit bowl containing the same type of fruit (apples, oranges, pears, bananas and kiwis) but not the identical number or pieces of fruit. While preparing for your classification task, you tried to recognize specific features of a type of fruit, e.g. the lengthy shape of a banana, the furry surface of a kiwi or the weight of an orange, by touching the fruit and connecting the haptic information to the type of fruit you see in your hands: You set up a set of rules of what each fruit feels like. Blind-folded and with the new fruit bowl in front of you, you classify the fruit by the rules you learned beforehand. How successfully you executed your task can be assessed by removing the blindfold and comparing your estimation to the true fruit type of each fruit in the bowl.

The fruit bowl example already gives a good impression on how data should be handled in a classification task. We notice, that there are two phases as part of the classification: the *training* and the *testing*.
The performance of the classifier is tested, as seen in the fruit bowl example, on a separate *test* data set. This should remain untouched during training, because otherwise the performance of the classifier can be biased: obviously the classification rules learnt on the training data apply well to the training data. The difficulty of a classification task is to learn general and at the same time precise rules, which can be transferred to a new data set and still deliver a good discrimination between classes.

For a more detailed introduction the reader is referred to Friedman et al. (2001), which this section is based on.

3.1 Types of Classifiers

Logistic Regression

Imagine we wish to have a model, that answers one of the following modeling questions based on observed features:

- Does this picture show a dog: Yes or no?
- Given medical or vital data, does this patient have cancer: Yes or no?

The response Y is an indicator taking on values $Y \in \{\text{"no"}, \text{"yes"}\}$ or more mathematically taking on the classes $Y \in \{0, 1\}$. We want to model the posterior probability of the two classes $\{0, 1\}$ via linear functions in the input data $\boldsymbol{x}$, while at the same time ensuring that they sum to one and remain in $[0, 1]$, as Friedman et al. (2001) precisely expressed it. The model fulfilling this is called *logistic regression model* and is of the form:

$$ln\Big(\frac{p(\boldsymbol{x})}{1-p(\boldsymbol{x})}\Big) = ln\Big(\frac{P(Y=1\,|\,\boldsymbol{X}=\boldsymbol{x})}{P(Y=0\,|\,\boldsymbol{X}=\boldsymbol{x})}\Big) = \beta_0 + \sum_{j=1}^{d} \beta_j x_j \ , \tag{3.4}$$

with $p(\boldsymbol{x}) := P(Y = 1\,|\,\boldsymbol{X} = \boldsymbol{x})$. Rearranging (3.4) we get:

$$p(x) = P(Y=1\,|\,\boldsymbol{X}=\boldsymbol{x}) = \frac{exp(\eta)}{1+exp(\eta)} \ , \qquad \text{with:} \quad \eta := \beta_0 + \sum_{j=1}^{d} \beta_j x_j \ ,$$

and a sanity check confirms: $\frac{exp(\eta)}{1+exp(\eta)}$ is a:

$$\begin{aligned} \text{continuous:} \quad & \frac{exp(\eta)}{1+exp(\eta)} \in \mathcal{C}^{\infty}\big(\mathbb{R}\big) \ , \\ \text{monotonously increasing:} \quad & \frac{\partial}{\partial\eta}\frac{exp(\eta)}{1+exp(\eta)} = \frac{exp(\eta)}{\big(1+exp(\eta)\big)^2} > 0 \quad \forall\, \eta \in \mathbb{R} \ , \\ \text{and bounded:} \quad & \lim_{\eta\to-\infty}\frac{exp(\eta)}{1+exp(\eta)} = 0 \ , \qquad \lim_{\eta\to\infty}\frac{exp(\eta)}{1+exp(\eta)} = 1 \ , \end{aligned}$$

function and thus:

$$p(x) = \frac{exp(\eta)}{1+exp(\eta)} \in [0, 1] \ .$$

The expression in (3.4) can be seen as the logarithm of the *odds* of event $Y = 1$ occurring compared to $Y = 0$, which in this case is the reference class.[11]

How can we estimate the model parameters $\beta_0, \ldots, \beta_d$? Nelder and Wedderburn (1972) apply maximum likelihood estimation for generalized linear models, such as the binomial logistic regression model. They show, that "the solution of the maximum likelihood equations is equivalent to an iterative weighted least-squares procedure":
The log-likelihood equations are non-linear $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_d)$, which is why the *Fisher scoring method*, is used to solve them. This can be re-expressed as a weighted least squares procedure.
For data analysis we use implementation of the `R`-function `glm` of the package `stats`, R Core Team (2022).

[11]In the dichotomous case this is also called *odds of success*.

Multinomial Logistic Regression Let us now assume, that we do not just want to answer:

- Given medical data, does the patient have cancer: Yes or no?

but:

- Given medical data, how severe is the cancer the patient suffers from? Or more precise: what is the patient's cancer severity status, ranging from 1 (mild) to K (life-threatening)?

For answering this question we extend the logistic regression model with dichotomous response to the *multinomial logistic regression model* with multinomial response: Let $Y \in \{1, \dots, K\}$ for some $K \in \mathbb{N}$. We regard let $Y = K$ as the reference class. As a multinomial generalization of (3.4) we obtain:

$$ln\Big(\frac{P(Y = k \mid \boldsymbol{X} = \boldsymbol{x})}{P(Y = K \mid \boldsymbol{X} = \boldsymbol{x})}\Big) = \beta_{0k} + \sum_{j=1}^{d} \beta_{jk} x_j \ , \quad k \in [K-1] \ ,$$

where we have a different set of βs for each class $k \in [K-1]$, i.e. $\{\beta_{0k}, \dots, \beta_{dk}\}$. As before this can be rearranged and we obtain the *multinomial logistic regression model*:

$$P(Y = k \mid \boldsymbol{X} = \boldsymbol{x}) = \frac{exp\big\{\beta_{0k} + \sum_{j=1}^{d} \beta_{jk} x_j\big\}}{1 + \sum_{l=1}^{K-1} exp\big\{\beta_{0l} + \sum_{j=1}^{d} \beta_{jl} x_j\big\}} \ , \quad k \in [K-1] \ , \tag{3.5}$$

$$P(Y = K \mid \boldsymbol{X} = \boldsymbol{x}) = \frac{1}{1 + \sum_{l=1}^{K-1} exp\big\{\beta_{0l} + \sum_{j=1}^{d} \beta_{jl} x_j\big\}} \ . \tag{3.6}$$

Here the coefficient β_{jk} weights the influence of the jth covariate x_j for the case that $Y = k, \ k \in [K]$.
Let $\big(\boldsymbol{x} \ \boldsymbol{y}\big)$ be a data set:

$$\big(\boldsymbol{x} \ \boldsymbol{y}\big) = \big(x_{ij} \ y_i\big)_{i \in [n], \, j \in [d]} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1d} & y_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nd} & y_n \end{pmatrix} = \big(\boldsymbol{x}_1 \quad \cdots \quad \boldsymbol{x}_d \quad \boldsymbol{y}\big) \in \mathbb{R}^{n \times (d+1)} \ ,$$

$$\boldsymbol{x}_j, \boldsymbol{y} \in \mathbb{R}^n, \ j \in [d] \ ,$$

and let $\boldsymbol{x}'_j, \ j \in [d]$, be the vectors of standardized covariate values, i.e.:

$$x'_{ij} := \frac{x_{ij} - \bar{x}_j}{s_j}, \quad \boldsymbol{x}'_j = \begin{pmatrix} x'_{1j} \\ \vdots \\ x'_{nj} \end{pmatrix} \ ,$$

with the sample mean and sample standard deviation[12] given by:

[12]Both estimators are unbiased.

$$\bar{x}_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij} \ , \quad s_j := s(\boldsymbol{x}_j) = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_{ij} - \bar{\boldsymbol{x}}_j)^2} \ .$$

Let the multinomial logistic regression model estimated on a data set $\big(\boldsymbol{x}\ \boldsymbol{y}\big)$ be:

$$\hat{P}(Y = k \mid \boldsymbol{X}_i = \boldsymbol{x}_i) = \frac{exp\big\{\hat{\beta}'_{0k} + \sum_{j=1}^{d}\hat{\beta}'_{jk}\ x'_{ij}\big\}}{1 + \sum_{l=1}^{K-1} exp\big\{\hat{\beta}'_{0l} + \sum_{j=1}^{d}\hat{\beta}'_{jl}\ x'_{ij}\big\}} \ , \quad k \in [K-1] \ , \tag{3.7}$$

$$\hat{P}(Y = K \mid \boldsymbol{X}_i = \boldsymbol{x}_i) = \frac{1}{1 + \sum_{l=1}^{K-1} exp\big\{\hat{\beta}'_{0l} + \sum_{j=1}^{d}\hat{\beta}'_{jl}\ x'_{ij}\big\}} \ , \tag{3.8}$$

where the superscript $'$ indicates, that $x_j,\ j \in [d]$ have been standardized before fitting. Then β'_{jk} is the coefficient of the standardized covariate X'_j for $Y = k$. To illustrate how to obtain β'_{jk} from the non-standardized β_{jk}, we compute the so called *log-odds* of class $k \in [K-1]$:

$$\begin{aligned}
log\left(\frac{\hat{P}(Y = k \mid \boldsymbol{X}_i = \boldsymbol{x}_i)}{\hat{P}(Y = K \mid \boldsymbol{X}_i = \boldsymbol{x}_i)}\right) &= log\Bigg(\frac{\Big(exp\big\{\hat{\beta}_{0k} + \sum_{j=1}^{d}\hat{\beta}_{jk}x_{ij}\big\}\Big)}{1 + \sum_{l=1}^{K-1} exp\big\{\hat{\beta}_{0l} + \sum_{j=1}^{d}\hat{\beta}_{jl}x_{ij}\big\}} \\
&\qquad \cdot \Big(1 + \sum_{l=1}^{K-1} exp\big\{\hat{\beta}_{0l} + \sum_{j=1}^{d}\hat{\beta}_{jl}x_{ij}\big\}\Big)\Bigg) \\
&= log\Big(exp\big\{\hat{\beta}_{0k} + \sum_{j=1}^{d}\hat{\beta}_{jk}x_{ij}\big\}\Big) \\
&= \hat{\beta}_{0k} + \sum_{j=1}^{d}\hat{\beta}_{jk}x_{ij} \\
&= \hat{\beta}'_{0k} + \sum_{j=1}^{d}\hat{\beta}'_{jk}x'_{ij} \\
&= \hat{\beta}'_{0k} + \sum_{j=1}^{d}\hat{\beta}'_{jk}\frac{x_{ij} - \bar{\boldsymbol{x}}_j}{s_j} \\
&= \Big(\hat{\beta}'_{0k} - \sum_{j=1}^{d}\frac{\hat{\beta}'_{jk}\bar{\boldsymbol{x}}_j}{s_j}\Big) + \sum_{j=1}^{d}\frac{\hat{\beta}'_{jk}}{s_j}x_{ij} \ .
\end{aligned}$$

By this we conclude, that:

$$\hat{\beta}_{jk} = \frac{\hat{\beta}'_{jk}}{s_j} \quad \Longleftrightarrow \quad \hat{\beta}'_{jk} = s_j\hat{\beta}_{jk} \ , \tag{3.9}$$

$$\hat{\beta}_{0k} = \hat{\beta}'_{0k} - \sum_{j=1}^{d}\frac{\hat{\beta}'_j\bar{\boldsymbol{x}}_j}{s_j} \quad \Longleftrightarrow \quad \hat{\beta}'_{0k} = \hat{\beta}_{0k} + \sum_{j=1}^{d}\frac{\hat{\beta}'_j\bar{\boldsymbol{x}}_j}{s_j} \ . \tag{3.10}$$

Model Selection with Lasso We want to include only those covariates into the logistic regression model that matter for prediction on a new data set. That is why we perform *model selection.* The resulting model becomes better "interpretable and has possibly a lower prediction error than the full model", Friedman et al. (2001). For logistic regression model selection can be done with *shrinkage* or *regularization* methods. We introduce one of them, namely the *Lasso.*

The general idea behind the *Lasso* is to maximize the log-likelihood by minimizing the negative log-likelihood while at the same time penalizing the L_1-norm of the coefficient vector. The L_1-norm of a vector is defined as:

Definition 3.1.1 (L_1-norm)**.** Let $\boldsymbol{x} \in \mathbb{R}^d$. Then:

$$\|\boldsymbol{x}\|_1 := \sum_{i=1}^{d} |x_i| \ ,$$

Brokate et al. (2016).

(a) **dichotomous classification problem:** For a dichotomous classification problem the *logistic regression model with Lasso* can be estimated by maximizing the penalized log-likelihood function of the logistic regression model of Equation (3.4):

$$\max_{\beta_0,\boldsymbol{\beta}} \left\{ \sum_{i=1}^{n} \left[y_i(\beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}_i) - \log(1 + e^{\beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}_i}) \right] - \lambda \sum_{j=1}^{d} |\beta_j| \right\} , \tag{3.11}$$

with $\boldsymbol{\beta} := (\beta_1, \ldots, \beta_d)^T$, see Friedman et al. (2001). They also give an estimation algorithm. In our analyses we use the implementation of the `R`-package `glmnet`, Simon et al. (2011).

Note, that the λ-parameter governs the amount of penalization on $\|\boldsymbol{\beta}\|_1$ in Equation (3.11). The higher it is, the stricter the penalty on $\|\boldsymbol{\beta}\|_1$ and thus the smaller the number of predictors included into the model. The penalty parameter λ needs to be set prior to estimation. It is chosen with k-fold cross-validation with deviance as loss, see Equation (3.1). A typical value is $k = 10$.

The value for λ is picked that results in the "[...] *least complex model within one standard error of the best* [...]" model, Friedman et al. (2001).

This means: the value for λ is picked, which corresponds to the model with deviance one standard error above (i.e. towards the positive λ-direction of) the minimal deviance achieved.

It makes sense, to pick the λ value, that results in the model with minimal loss. The reason for selecting a higher and thus more restrictive value for λ is, that over-parameterized models tend to be selected when choosing the λ value that gives a model with minimal deviance.

We use the implementation of the `cv.glmnet`-function of the `R`-package `glmnet` by Simon et al. (2011), in which 10-fold cross-validation and deviance as loss are the default values. The sequence of λs evaluated in cross-validation, which can be supplied by the user or chosen by `cv.glmnet`, can be displayed with the `plot`-function of `glmnet` as shown in Figure 6.

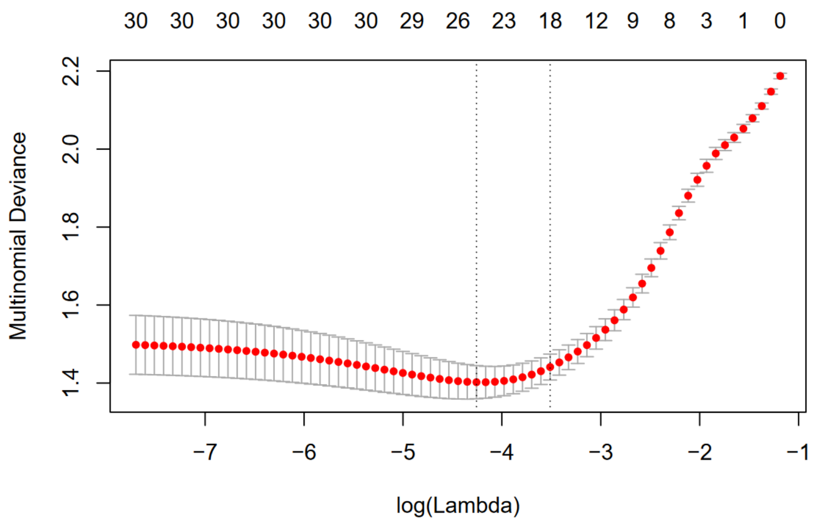

Figure 6: Displaying the sequence of λ values evaluated in cross-validation for the Lasso penalty parameter, Hastie and Qian (2014). The first vertical line marks the minimal deviance value, the vertical line to its right marks the λ corresponding to the deviance one standard error away from the minimal deviance towards the positive λ direction.

(b) **multinomial classification problem:** For Lasso-regularization in the multinomial regression model we need some pre-considerations:

Using the model formulation of Equations (3.5) and (3.6), where we set K as the reference class, we can only estimate whether a predictor has influence on an observation rather belonging to class $k \in [K-1]$ than to the reference class K.

However, we want to only then penalize a predictor, if it has weak influence on whether an observation belongs to *any* class in $[K]$. That is why for multinomial logistic regression with Lasso Simon et al. (2013) use a *symmetric* formulation of the multinomial logistic regression model. We use α for the model parameters to not confuse the symmetric formulation with the model formulation of Equations (3.5) and (3.6):

$$P(Y = k \mid \boldsymbol{X} = \boldsymbol{x}) = \frac{exp\left\{\alpha_{0k} + \sum_{j=1}^{d} \alpha_{jk} x_j\right\}}{\sum_{l=1}^{K} exp\left\{\alpha_{0l} + \sum_{j=1}^{d} \alpha_{jl} x_j\right\}} \,, \qquad k \in [K] \,. \tag{3.12}$$

Simon et al. (2013) show, that this formulation is unique if a penalty like in the

Lasso is added.

We estimate the *multinomial logistic regression model with Lasso* by minimizing the penalized negative log-likelihood function:

$$\min_{\boldsymbol{\alpha}} \Bigg\{ -\left[\frac{1}{n} \sum_{i=1}^{n} \Big(\sum_{k=1}^{K} z_{ik}(\alpha_{0k} + \boldsymbol{x}_i^T \boldsymbol{\alpha}_k) - \log\big(\sum_{k=1}^{K} e^{\alpha_{0k} + \boldsymbol{x}_i^T \boldsymbol{\alpha}_k}\big) \right] \\ + \lambda \left[\sum_{j=1}^{p} \|(\alpha_{1j}, \dots, \alpha_{Kj})^T\|_1 \right] \Bigg\}, \tag{3.13}$$

with $\boldsymbol{\alpha}_k := (\alpha_{k1}, \dots, \alpha_{kd})^T$ and $z_{ik} := I(y_i = k)$. Note that in the multinomial case, i.e. for $K > 1$ we obtain $K \times d$ regularized coefficients, with K the number of classes and d the number of covariates.

The algorithm solving this minimization problem is based on Park and Hastie (2007) and we use its implementation of the `R`-function `glmnet` of the `glmnet`-package.

As for the binomial logistic regression model the λ-parameter governs the amount of penalization on $\|(\alpha_{11}, \dots, \alpha_{d1}, \dots, \alpha_{1K}, \dots, \alpha_{dK})^T\|_1$ in Equation (3.13). For estimating the multinomial logistic regression model with Lasso we need to choose λ. This is done as explained above for the binomial logistic regression with Lasso.

Random Forest

The following section is based on Breiman (2001) and Friedman et al. (2001).

A *random forest* is a classification model, which consists itself of an ensemble of decorrelated classifiers, namely *decision trees.* A random forest prediction is equal to the label predicted most often by the trees of the random forest. By this the random forest exploits the idea of *bagging*: reducing its variance by averaging many noisy decision trees, which, if grown sufficiently deep, show relatively low bias.

What is a decision tree and how are they grown to obtain a random forest classifier?
A *decision tree* or *classification tree* divides a data set and allocates its observations to the tree's nodes.[13] It is grown on a training data set by splitting an existing node of the tree into two new nodes, which is equal to allocating the observations of the existing node to the two new nodes. This is done according to whether an observation's value of a randomly selected feature is above or below the selected split value. The split feature as well as the split point are chosen, which minimize a so-called *node impurity measure*, the *Gini index*:

[13]An introduction to graph theory and trees can be found on page 15 and following. In the context of random forests, the nodes are often also called *regions*.

Definition 3.1.2 (Gini index)**.** Let the nodes of a decision tree T be numbered from 1 to M. For a node $m \in [M]$ representing a region R_m containing $N_m \in \mathbb{N}$ observations $(\boldsymbol{x}_i, y_i)_{i \in [N_m]}$ let:

$$\hat{p}_{mk} := \frac{1}{N_m} \sum_{\boldsymbol{x}_i \in R_m} \mathbb{1}(y_i = k) \ .$$

The *Gini index* is then defined as:

$$\sum_{k \neq j} \hat{p}_{mk} \hat{p}_{mj} = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \ , \tag{3.14}$$

see Friedman et al. (2001).

There exist other node impurity measures for classification, such as the misclassification error or the cross-entropy or deviance, see Friedman et al. (2001). For growing a random forest on data we use the `R`-package `randomForest` by Liaw and Wiener (2002) with the Gini index, which is also equal to the default impurity measure in `randomForest`.

Algorithm 2 illustrates, how a random forest classifier is built from data and how it predicts the label $\hat{y}_i$ of an observation $\boldsymbol{x}_i$ for $i \in [n]$.

Algorithm 2: Random Forest, see Friedman et al. (2001)

```
Input: training data $(\boldsymbol{x}\ \boldsymbol{y})_{train} \in \mathbb{R}^{n \times (d+1)}$, random forest parameters $n_{tree}$, $m_{try}$,
    $s_{nodesize}$
Output: random forest classifier $\hat{C}_{rf}^{n_{tree}}(\cdot)$ and predictions $\hat{\boldsymbol{y}}$
1 for $b \in \{i, \dots, n_{tree}\}$ do
2     draw a bootstrap sample $Z^*$ by drawing $n$ observations from the training data
        with replacement
3     grow a random forest tree $T_b$ to the bootstrapped data $Z^*$ by recursively
        repeating the following steps for each terminal node of the tree, until the
        minimum node size $s_{nodesize}$ is reached:
          (i) select $m_{try}$ covariates at random from all $d$ predictors
         (ii) pick best split variable and split point pair with regard to the Gini index
              among the $m_{try}$ covariates
        (iii) split the current node into two new nodes
      output the ensemble of trees $\{T_b\}_{b \in [n_{tree}]}$
      $\hat{y}_i \leftarrow \hat{C}_{rf}^{n_{tree}}(\boldsymbol{x}_i) :=$ majority vote of $\{\hat{C}_b(\boldsymbol{x}_i)\}_{b \in [n_{tree}]}$ with $\hat{C}_b(\boldsymbol{x}_i)$ the class
      prediction of the $b$th random forest tree
4 end
5 return random forest classifier $\hat{C}_{rf}^{n_{tree}}(\cdot)$ and predictions $\hat{\boldsymbol{y}}$
```

For fitting a random forest classifier to data as illustrated in Algorithm 2 we need to set or tune the model parameters:

- n_{tree}: The number of trees in the random forest. A good estimate can be obtained by using the out-of-bag (OOB) error estimate, introduced below in Definition 3.1.3. The number of trees, for which this value stabilizes, will be picked.

  Definition 3.1.3 (out-of-bag (OOB) error estimate)**.** For each observation $(\boldsymbol{x}_i, y_i)_{i\in[n]}$ of the training data $\big(\boldsymbol{x}\ \boldsymbol{y}\big)_{train} \in \mathbb{R}^{n\times(d+1)}$ construct its random forest classifier by averaging *only* over those trees corresponding to the bootstrap samples Z^* in which $(\boldsymbol{x}_i, y_i)$ did *not* appear. We denote it as $\hat{C}_{rf}^{-i}(\cdot)$. Then the *out-of-bag error estimate* is the prediction error estimate averaged over all random forest classifiers obtained as described above:

  $$\text{OOB error estimate:}\quad \frac{1}{n}\sum_{i=1}^{n} L(y_i, \hat{C}_{rf}^{-i}(\boldsymbol{x_i}))\ ,$$

  see Friedman et al. (2001).

  The OOB error estimate is similar to N-fold cross-validation. It can be estimated "along the way" while fitting the random forest classifier as described in Algorithm 2.

- m_{try}: The cardinality of the subset of covariates $\{X_{i_1}, \ldots X_{i_p}\}$, that are randomly drawn from all the d covariates $\{X_1, \ldots, X_d\}$ in each branching process in each decision tree:

  $$\{i_1, \ldots, i_p\} \subset [d] \quad \text{with} \quad m_{try} = |\{i_1, \ldots, i_p\}|\ .$$

  The current split variable is then chosen from $\{X_{i_1}, \ldots X_{i_p}\}$ by minimization of the Gini index defined in Definition 3.1.2. The value $m_{try} = \lfloor\sqrt{d}\rfloor$ with d the total number of covariates is recommended by Friedman et al. (2001) as a typical value. It is also the default value in the `R`-package `randomForest`, see Liaw and Wiener (2002).

- $s_{nodesize}$: The minimal size of all terminal nodes. Here "size" means the number of observations "in the terminal node". These are the observations classified according to the rule corresponding to the path in the decision tree from its root node to the respective terminal node. Intuitively it makes sense, that the smaller $s_{nodesize}$ the more interactions a classification tree can capture and the smaller its prediction error. Therefore $s_{nodesize}$ for our analyses we set $s_{nodesize}$ to 1, which corresponds to the default value in the `R`-package `randomForest`, Liaw and Wiener (2002).

3.2 Classification Performance Measures

An overview of measures for comparing classifications and clusterings can be found in Wagner and Wagner (2007). Let $\big(\mathcal{X}\ \boldsymbol{y}\big) \in \mathbb{R}^{n\times(d+1)}$ with n, $d \in \mathbb{N}$ be a data set:

$$\big(\mathcal{X}\ \boldsymbol{y}\big) := \begin{pmatrix} \boldsymbol{x}_1^T & y_1 \\ \vdots & \vdots \\ \boldsymbol{x}_n^T & y_n \end{pmatrix} \in \mathbb{R}^{n\times(d+1)} ,$$

where y_i is the true class label of observation $\boldsymbol{x}_i^T$ with $i \in [n]$. Let there exist a subset of $\big(\mathcal{X}\ \boldsymbol{y}\big)$, the training set $\big(\mathcal{X}\ \boldsymbol{y}\big)_{train}$, containing $m < n$ observations:

$$\big(\mathcal{X}\ \boldsymbol{y}\big)_{train} := \big(\boldsymbol{x}_i^{T,\,train} \quad y_i^{train}\big)_{i\in[m]} := \big(\boldsymbol{x}_i^T \quad y_i\big)_{i\in T} \in \mathbb{R}^{m\times(d+1)}$$

We use the notation:

$$\mathcal{X}_{train} := \begin{pmatrix} \boldsymbol{x}_1^{T,\,train} \\ \vdots \\ \boldsymbol{x}_m^{T,\,train} \end{pmatrix} \in \mathbb{R}^{m\times d} , \qquad \boldsymbol{y}_{train} := \begin{pmatrix} y_1^{train} \\ \vdots \\ y_m^{train} \end{pmatrix} \in \mathbb{R}^{m} ,$$

$$\big(\mathcal{X}\ \boldsymbol{y}\big)_{train} = \big(\mathcal{X}_{train}\ \boldsymbol{y}_{train}\big) .$$

The remaining observations of $\big(\mathcal{X}\ \boldsymbol{y}\big)$ belong to the test set:

$$\big(\mathcal{X}\ \boldsymbol{y}\big)_{test} := \big(\boldsymbol{x}_i^{T,\,test} \quad y_i^{test}\big)_{i\in[n-m]} := \big(\boldsymbol{x}_i^T \quad y_i\big)_{i\in[n]\setminus T} ,$$

The following classification performance measures are based on the comparison of the estimated class label $\hat{y}_i^{test} = \hat{f}(\boldsymbol{x}_i^{test})$ to a class label y_i^{test}. Here $\hat{f}$, the estimate of the classifier f, is obtained on $\big(\mathcal{X}\ \boldsymbol{y}\big)_{train}$.

Confusion Matrix

A very intuitive way of comparing estimated with true class labels on the test set $\big(\mathcal{X}\ \boldsymbol{y}\big)_{test}$ is counting the observations, where the class membership was correctly predicted by the classifier $f(\cdot)$, i.e. $(y_i, \hat{y}_i) = (k, k)$ with $k \in [K]$, and the ones, where the true class was confused with another class, i.e. $(y_i, \hat{y}_i) = (k, l)$ for $k \neq l \in [K]$ and $i \in [n] \setminus T$. So for $K \in \mathbb{N}$ distinct classes this can then be displayed or stored in a $K \times K$-matrix. This matrix is appropriately called *confusion matrix* and defined as:

$$\begin{aligned} C &= (c_{kl})_{k,l\in[K]} \in \mathbb{R}^{K\times K} , \\ c_{kl} &:= \big|\{i \in [n] \setminus T \mid (y_i, \hat{y}_i) = (k,l),\ k,l \in [K]\}\big| , \\ \hat{y}_i &:= \hat{f}(\boldsymbol{x}_i) \quad \text{with} \quad i \in [n] \setminus T . \end{aligned} \tag{3.15}$$

Thus $\sum_{i\in[K]} c_{ii}$, the sum of diagonal elements of C gives the number of pairs $(y_i, \hat{y}_i)$, where the classifier correctly estimated the true label.
We illustrate the confusion matrix in an example:

Example 3.2.1. Let $Y \in \{0,1\}$, let $\hat{f}(\cdot)$ be a classifier trained on training data $\big(\boldsymbol{x}\ \boldsymbol{y}\big)_{train}$ and let the test data be:

$$\big(\boldsymbol{x}\ \boldsymbol{y}\big)_{test} = \begin{pmatrix} \boldsymbol{x}_1^T & 1 \\ \boldsymbol{x}_2^T & 1 \\ \boldsymbol{x}_3^T & 0 \\ \boldsymbol{x}_4^T & 1 \\ \boldsymbol{x}_5^T & 0 \\ \boldsymbol{x}_6^T & 1 \\ \boldsymbol{x}_7^T & 1 \\ \boldsymbol{x}_8^T & 0 \\ \boldsymbol{x}_9^T & 1 \\ \boldsymbol{x}_{10}^T & 1 \end{pmatrix} ,$$

with predictions:

$$\begin{aligned} &\hat{y}_1 = \hat{f}(\boldsymbol{x}_1) = 1 , \quad \hat{y}_2 = \hat{f}(\boldsymbol{x}_2) = 1 , \quad \hat{y}_3 = \hat{f}(\boldsymbol{x}_3) = 1 , \\ &\hat{y}_4 = \hat{f}(\boldsymbol{x}_4) = 1 , \quad \hat{y}_5 = \hat{f}(\boldsymbol{x}_5) = 1 , \quad \hat{y}_6 = \hat{f}(\boldsymbol{x}_6) = 1 , \\ &\hat{y}_7 = \hat{f}(\boldsymbol{x}_7) = 0 , \quad \hat{y}_8 = \hat{f}(\boldsymbol{x}_8) = 0 , \quad \hat{y}_9 = \hat{f}(\boldsymbol{x}_9) = 0 , \quad \hat{y}_{10} = \hat{f}(\boldsymbol{x}_{10}) = 0 . \end{aligned}$$

This yields the following confusion matrix:

$$C = \begin{pmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} .$$

$\mathcal{F}$-Measure for Dichotomous Classification Problems

We introduce the *$\mathcal{F}$-measure* used for dichotomous classification problems: Let $Y \in \{0,1\}$ and let therefore the confusion matrix C be as defined in (3.15):

$$C = \begin{pmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{pmatrix} , \tag{3.16}$$

which in form of a table looks like:

prediction $\hat{y}$

actual y	$\hat{y} = 0$	$\hat{y} = 1$	$\sum$
$y = 0$	c_{00} (true "0")	c_{01} (false "1")	$c_{00} + c_{01}$ (actual "0")
$y = 1$	c_{10} (false "0")	c_{11} (true "1")	$c_{10} + c_{11}$ (actual "1")
$\sum$	$c_{00} + c_{10}$ (predicted "0")	$c_{01} + c_{11}$ (predicted "1")	n

Table 3: Contingency table of a 0-1-classification problem corresponding to the confusion matrix C in (3.16).

The *precision* of the classification with confusion matrix C (3.16) and contingency table in Table 3 can be described as the fraction of the predicted "1"s, for which the prediction is correct, i.e. $y = 1$ and thus:

$$\text{precision} := \frac{c_{11}}{c_{11} + c_{01}} = \frac{\text{true "1"}}{\text{predicted "1"}} \in [0, 1] \; . \tag{3.17}$$

The *recall* on the other hand gives the fraction of observations with label $y = 1$, that a classifier could correctly recall in its prediction:

$$\text{recall} := \frac{c_{11}}{c_{11} + c_{10}} = \frac{\text{true "1"}}{\text{actual "1"}} \in [0, 1] \; . \tag{3.18}$$

The higher the value for precision and recall respectively the better we regard the classifier $\hat{f}(\cdot)$ tested on $(\boldsymbol{x}\ \boldsymbol{y})_{test}$. Ideally we want a classifier to have high values for precision and recall simultaneously. We illustrate why in two small examples:

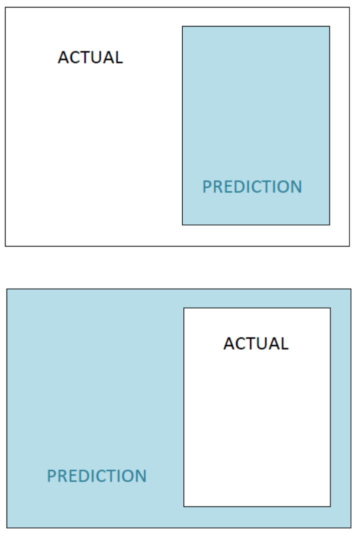

Figure 7: Two 0-1-classification examples for illustrating precision and recall. Observations within a rectangle have (estimated) label 1. The respective larger rectangle is equal to the domain.

- The top image of Figure 7 shows a 0-1-classification example, where all observations have class labels 1. This yields precision $= 1$, but recall < 0.5.
- The bottom image of Figure 7 shows a 0-1-classification example, where all labels are estimated to be 1. This yields recall $= 1$, but precision < 0.5.

The classifiers of both examples would be of no use in an application, as either "1" (top) or "0" (bottom) would be estimated incorrectly more than half of the time. However, for precision and recall we get the following values:

- **top:**

$$\text{precision} = \frac{\text{true "1"}}{\text{predicted "1"}} = \frac{\text{area of "prediction"}}{\text{area of "prediction"}} = 1 \ ,$$

$$\text{recall} = \frac{\text{true "1"}}{\text{actual "1"}} = \frac{\text{area of "prediction"}}{\text{are of "actual"}} < 0.5 \ .$$

- **bottom:**

$$\text{precision} = \frac{\text{true "1"}}{\text{predicted "1"}} = \frac{\text{area of "actual"}}{\text{area of "prediction"}} < 0.5 \ ,$$

$$\text{recall} = \frac{\text{true "1"}}{\text{actual "1"}} = \frac{\text{area of "actual"}}{\text{are of "actual"}} = 1 \ .$$

Consulting just the precision in the top case or the recall in the bottom case could naively make us assume, that we have a good classifier at hand in both cases.

Due to this it is reasonable to consult the *$\mathcal{F}$-measure*. It combines precision and recall in the *harmonic mean*:

Definition 3.2.1 (harmonic mean)**.** The *harmonic mean* of the n observations $x_1, \dots, x_n$ with $n \in \mathbb{N}$ is defined as:

$$H := \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}} = \frac{n \cdot \prod_{i=1}^{n} x_i}{\sum_{i=1}^{n} x_i} \ .$$

Definition 3.2.2 ($\mathcal{F}$-measure)**.** For precision $\in (0,1]$ and recall $\in (0,1]$ we defined the *$\mathcal{F}$-measure* as:

$$\mathcal{F} := \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{2 \cdot \text{true "1"}}{\text{actual "1"} \ + \ \text{predicted "1"}} \ .$$

Here $(a, b]$ is the notation for the half open interval excluding a for $a, b \in \mathbb{R}$ with $a < b$.

Remark 3.2.1. If we have precision = recall = 0, Definition 3.2.2 does not apply. Following Ballabio et al. (2018) we set in this case:

$$\mathcal{F} := 0 \ .$$

Remark 3.2.2. We can reformulate the $\mathcal{F}$-measure in terms of the expression used in Table 3 as:

$$\mathcal{F} = \frac{2 \cdot \text{true "1"}}{\text{actual "1"} \ + \ \text{predicted "1"}} \ .$$

We quickly show the following property:

Lemma 3.2.1. For precision $\in (0,1]$ and recall $\in (0,1]$ we have, that:

(i) $\mathcal{F} \in (0,1]$,

(ii) $\mathcal{F}$ is strictly monotonously increasing in precision and recall.

Proof. Let $x :=$ precision and $y :=$ recall and we denote $\mathcal{F}(x,y)$ as the $\mathcal{F}$-measure is a function of precision and recall:

$$\mathcal{F} : (0,1]^2 \to \mathbb{R} \; .$$

(i) Then:

$$\begin{aligned}\mathcal{F}(x,y) &= \frac{2 \cdot x \cdot y}{x+y} = \frac{2}{\frac{1}{x}+\frac{1}{y}} \\ &\leq \max_{(x,y)\in[0,1]^2} \frac{2}{\frac{1}{x}+\frac{1}{y}}\end{aligned}$$

Finding the maximum:

$$\max_{(x,y)\in[0,1]^2} \frac{1}{\frac{1}{x}+\frac{1}{y}} \; ,$$

is equivalent to:

$$\Big(\min_{(x,y)\in(0,1]^2} \frac{1}{x} + \frac{1}{y}\Big)^{-1} = \frac{1}{2} \; .$$

Thus we obtain:

$$\mathcal{F}(x,y) \leq \max_{(x,y)\in(0,1]^2} \frac{2}{\frac{1}{x}+\frac{1}{y}} = 1 \; .$$

It obviously holds, that:

$$\mathcal{F}(x,y) \geq 0 \qquad \forall\ (x,y) \in (0,1]^2 \; ,$$

as both numerator and denominator are positive functions on $(0,1]^2$, where 0 cannot be reached due to a positive numerator for any $(x,y) \in (0,1]^2$.

(ii) Obviously, $\mathcal{F}(x,y) \in \mathcal{C}^1\big((0,1]^2\big)$. We can derive:

$$\nabla\mathcal{F}(x,y) = \Big(\frac{2y^2}{(x+y)^2}, \frac{2x^2}{(x+y)^2}\Big)^T \; .$$

Then for any $y \in (0,1]$ it holds, that:

$$\frac{\partial}{\partial x}\mathcal{F}(x,y) = \frac{2y^2}{(x+y)^2} > 0 \; ,$$

and equivalently for any $x \in (0,1]$:

$$\frac{\partial}{\partial y}\mathcal{F}(x,y) = \frac{2x^2}{(x+y)^2} > 0 \; .$$

Thus for any $(x,y) \in (0,1]^2$ and any $(\tilde{x},\tilde{y}) \in (0,1]^2$ with $\tilde{x} \geq x$ and $\tilde{y} \geq y$, where at least one of the inequalities is strict, we have that:

$$\mathcal{F}(x,y) < \mathcal{F}(\tilde{x},\tilde{y}) \; .$$

□

Due to (ii) we can say, that: the higher the value of the $\mathcal{F}$-measure, the better we regard the respective classifier.

Why is it the harmonic mean instead of the arithmetic mean[14] that is used to compare the rates in the $\mathcal{F}$-measure? Ferger (1931) illustrates the reason for this very well: "If it is desired to keep constant that factor which is constant in the rates as stated, then the arithmetic mean should be used: if, on the contrary, the recorded rates make variable the factor desired to be constant, then the harmonic mean is the correct average to be employed."
For the precision we set the true "1" into relation to the predicted "1". For the recall we set the number of true "1" into relation to the actual "1". Through the harmonic mean we obtain a measure, which compares the true "1" to the constant rate (actual "1! + predicted "1").

Hand and Christen (2018) show, that $\mathcal{F}$-measure can also be formulated as a weighted arithmetic mean of precision and recall. These weights *depend on the classification itself*, i.e. on the cell counts of the corresponding confusion matrix. This is a "fundamental flaw" of the $\mathcal{F}$-measure. We decide to still use it as a performance measure for classification due to its widespread application.

Receiver Operating Characteristics (ROC), Area Under the ROC Curve and Its Multi-Class Generalization

The following introduction on the *receiver operating characteristic (ROC)* and the *area under the receiver operating characteristics curve* is based on Fawcett (2006) and Hand and Till (2001).

We can measure the performance of a classification model $f : \mathbb{R}^d \to \{0,1\}$ for a dichotomous classification problem with *true positive rate (tp rate)* or *sensitivity* and *false positive rate (fp rate)* or *specificity*:

Definition 3.2.3 (true positive rate (tp rate), false positive rate (fp rate))**.** Let $f : \mathbb{R}^d \to \{0,1\}$ be a classifier for a dichotomous classification task and let with $(c_{ii})_{i\in[0,1]}$ be its confusion matrix as defined in Table 3. Then the *sensitivity* and *specificity* of f are defined as:

$$\text{true positive rate} = \text{sensitivity} = \text{recall} = \frac{c_{11}}{c_{11}+c_{10}} = \frac{\text{true "1"}}{\text{actual "1"}} \in [0,1]\ , \tag{3.19}$$

$$\text{false positive rate} = \text{specificity} = \frac{c_{01}}{c_{01}+c_{00}} = \frac{\text{false "1"}}{\text{actual "0"}} \in [0,1]\ . \tag{3.20}$$

As the names already suggest, the *true positive rate* measures, which percentage of actual "positives" or "1" are recognized as "1" by the classifier. The *false positive rate* measures how many "negative" or "0" are incorrectly classified as "positive" or "1" when comparing

[14]The *arithmetic mean* of n observations $x_1,\dots,x_n$ with $n \in \mathbb{N}$ is given by: $A := \frac{1}{n}\sum_{i=1}^{n} x_i$.

to the total number of "0". Thus, the tp rate can be viewed as benefit and the fp rate can be viewed as cost.

The *receiver operating characteristics* graph plots the true positive rate of a classifier f against its false positive rate, see Figure 8. It therefore "depicts relative trade-offs between benefits (true positives) and costs (false positives)", Fawcett (2006). Some remarks to the ROC plot:

Remark 3.2.3.

- One point in the plot corresponds to one classifier, such as the round dot in Figure 8. The corresponding classifier has a tp rate of 80% and a fp rate of 20%.
- A *perfect classifier*, which predicts all labels correctly, would score a tp rate of 1 and a fp rate of 0. This corresponds to the star-shaped point on Figure 8.
- Obviously, a classifier f_1 is better than a classifier f_2, if:

  $$\text{tp rate}_1 \geq \text{tp rate}_2 \quad \wedge \quad \text{fp rate}_1 \leq \text{fp rate}_2 \ ,$$

  with at least one inequality being strict. Informally speaking, f_1 is better than f_2, if its point in the ROC plot "is to the northwest of" f_2's ROC-point, Fawcett (2006).
- A classifier, which was able to gain information on the response Y given the covariate values $\boldsymbol{X} = \boldsymbol{x}$ in training, should score:

  $$\text{tp rate} > \text{fp rate} \ .$$

  It should thus yield a point in the ROC plot lying above the dashed diagonal line tp rate = fp rate in Figure 8. The latter corresponds to the performance of classifying by random guessing and serves as a benchmark for comparison.

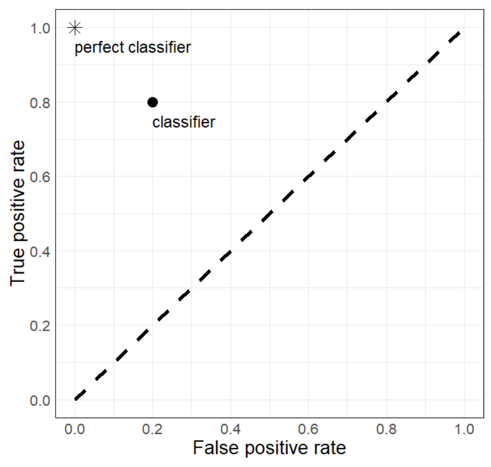

Figure 8: ROC plot: the classifier displayed achieves a true positive rate of 0.8 and a false positive rate of 0.2.

By now we obtained only a single point in the ROC plot depicting the performance of a classifier f. How do we obtain ROC *curves* illustrating the performance of f?

In most cases the classifier at hand is not a *discrete* classifier outputting a class labels, but it is a *probabilistic* classifier: its predictions are posterior probabilities of the label given the predictor values:

$$\hat{p}(\boldsymbol{x}) := \hat{P}(Y = 1 \,|\, \boldsymbol{X} = \boldsymbol{x}) = \hat{f}(\boldsymbol{x}) \in [0, 1] \; .$$

To obtain labels from the predictions $\hat{p}(\boldsymbol{x})$ of f we need to *threshold* the posterior probability estimates with a *threshold value* $t \in [0, 1]$:

$$\hat{y} := \begin{cases} 1 & \text{if} \quad \hat{p}(\boldsymbol{x}) \geq t \; , \\ 0 & \text{else} \; . \end{cases}$$

For each threshold value $t \in [0, 1]$ we obtain so to say a different version of f, which we call $f_{(t)}$ with a different set of predictions:

$$f_{(t)}(\boldsymbol{x}) := \big(\mathbb{1}_{\{x \geq t\}} \circ \hat{f}\big)(\boldsymbol{x}) = \mathbb{1}_{\{x \geq t\}}\big(\hat{f}(\boldsymbol{x})\big)$$

We obtain a *ROC curve* for a classifier f by moving the threshold t from 0 to 1 and computing the resulting tp and fp rate of $f_{(t)}$ for each $t \in [0, 1]$. An algorithm for efficiently computing ROC curves can be found in Fawcett (2006).

The *area under the ROC curve (AUC)* is a one-number-summary of the ROC curve and therefore enables to directly compare two classifiers f_1 and f_2. An algorithm for computing the AUC can again be found in Fawcett (2006). Some remarks on the AUC:

Remark 3.2.4.

- As the ROC curve has the unit cube $[0, 1]^2$ as support, the AUC takes on values between 0 and 1.
- A perfect classifier, which classifies all observation correctly for all $t \in [0, 1]$ achieves an AUC = 1.
- A classifier is considered better than random guessing, if it scores an AUC ≥ 0.5, which is equal to the AUC of classification by random guessing.
- For randomly chosen observation $(\boldsymbol{x}_i, y_i)$ and $(\boldsymbol{x}_j, y_j)$ with $y_i =$ "1" and $y_j =$ "0" the AUC of a classifier f is equivalent to:

  $$AUC = P(p(\boldsymbol{x}_i) > p(\boldsymbol{x}_j)) \; ,$$

  with $p(\boldsymbol{x}_i) = f(\boldsymbol{x}_i)$ and $p(\boldsymbol{x}_j) = f(\boldsymbol{x}_j)$, Fawcett (2006).

How can we generalise the AUC to a performance measure for a multi-class classification problem?
Let C be the set of all classes of a classification problem, e.g. $C = [K]$ with $K \in \mathbb{R}$. The basic idea behind the multi-class generalization of the AUC by Hand and Till (2001) is to compare all pairs of classes in C with the of the AUC and average the resulting values over the number of distinct class pairs existing. The resulting measure, which Hand and Till (2001) call M, is insensitive to class distribution. Fawcett (2006) reformulates it as:

Definition 3.2.4 (multi-class AUC (mAUC))**.** Let C be the set of classes of a multi-class classification problem. Then the *multi-class AUC (mAUC)* is defined as:

$$mAUC := \frac{2}{|C|(|C|-1)} \sum_{\{c_i,c_j\}\in C} AUC(c_i, c_j) \tag{3.21}$$

with $AUC(c_i, c_j)$ the AUC of a dichotomous classification problem only involving classes $c_i \in C$ and $c_j \in C$.

In our analyses we use the `R`-package `pROC` by Robin et al. (2011) for the computation of the AUC and the mAUC.

4 Methodology: Synthetic Data for Classification Generated with Vines

Usually, the task is to produce a simulated data set, which resemble the true one as much as possible, but preserves privacy. We turn the problem to a more specific context. We assume, that the purpose of the analysis to be done on the true and on the synthetic data set is classification. This means, that the data have a label per unit or patient, which is used for classification. Because of this specific context, we are interested in producing a synthetic data set, which allows the user to estimate a classification rule from the synthetic data, which is similar to the classification rule, which would be estimated on the true data. We also want, that the classifier trained on this synthetic data can reproduce well the predictive performance of the classifier trained on the true data set. Because we want the synthetic data to exhibit the same dependence structure of the true data, as this is needed in our case, we believe that vine copulas can be appropriate to generate synthetic data from a true data set.

We restrict ourselves to a setup, where the true data consist of a data matrix $\mathcal{X}_{true} \in \mathbb{R}^{n\times d}$ of quantitative random variables $X_{true,\,i,j},\ i \in [n],\ j \in [d]$ and an ordinal response vector $\boldsymbol{Y}_{true} \in \{1,\dots,K\}^n$:

$$\mathcal{X}_{true} := \begin{pmatrix} X_{11,\,true} & X_{12,\,true} & \cdots & X_{1d,\,true} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n1,\,true} & X_{n2,\,true} & \cdots & X_{nd,\,true} \end{pmatrix} = \begin{pmatrix} \boldsymbol{X}^T_{1,\,true} \\ \vdots \\ \boldsymbol{X}^T_{n,\,true} \end{pmatrix} \in \mathbb{R}^{n\times d}, \quad \boldsymbol{X}^T_{i,\,true} \in \mathbb{R}^d,\ i \in [n]\ ,$$

$$\boldsymbol{Y}_{true} := \begin{pmatrix} Y_{1,\,true} \\ \vdots \\ Y_{n,\,true} \end{pmatrix} \in \{1,\dots,K\}^n\,, \qquad n, d, K \in \mathbb{N}.$$

Each row $\boldsymbol{X}^T_{true,\,i}$ of $\mathcal{X}_{true}$ can for example be interpreted as patient i in a clinical study: it captures the patient's features $X_{true,\,i,1},\dots,X_{true,\,i,d}$. The label $Y_{true,\,i}$, which is assigned to patient i, gives his or her health status, for example it indicates the severity of patient i's disease from mild, i.e. 1, to life-threatening, i.e. K.

Let F_{true} be a $(d+1)$-dimensional probability distribution function. We assume, that:

$$\left(\boldsymbol{X}^T_{1,\,true} \quad Y_{1,\,true}\right),\dots,\left(\boldsymbol{X}^T_{n,\,true} \quad Y_{n,\,true}\right) \overset{i.i.d.}{\sim} F_{true}\ . \tag{4.1}$$

From $\mathcal{X}_{true}$ and $\boldsymbol{Y}_{true}$ we generate the synthetic data $\boldsymbol{x}_{synth} \in \mathbb{R}^{m\times d}$ with synthetic response $\boldsymbol{y}_{synth} \in \mathbb{R}^n$, where $m \in \mathbb{N}$ can be possibly different from n:

$$\boldsymbol{x}_{synth} := \begin{pmatrix} x_{11,\,synth} & x_{12,\,synth} & \cdots & x_{1d,\,synth} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1,\,synth} & x_{m2,\,synth} & \cdots & x_{md,\,synth} \end{pmatrix} \in \mathbb{R}^{m\times d}\ ,$$

$$\boldsymbol{y}_{synth} := \begin{pmatrix} y_{1,\,synth} \\ \vdots \\ y_{m,\,synth} \end{pmatrix} \in \{1,\dots,K\}^m\ .$$

4.1 Generating Synthetic Data with Vines

We aim to produce synthetic data $\boldsymbol{x}_{synth}$ and $\boldsymbol{y}_{synth}$ such that a classifier trained on $\left(\boldsymbol{x}_{synth}\ \boldsymbol{y}_{synth}\right)$ learns a similar rule to the one it would have learned on $\left(\boldsymbol{\mathcal{X}}_{true}\ \boldsymbol{Y}_{true}\right)$.

Let $\left(\boldsymbol{\mathcal{X}}_{true}\ \boldsymbol{Y}_{true}\right)$ take on observed values $\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right)$:

$$\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right) := \begin{pmatrix} x_{11,\,true} & x_{12,\,true} & \cdots & x_{1d,\,true} & y_{1,\,true} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{n1,\,true} & x_{n2,\,true} & \cdots & x_{nd,\,true} & y_{n,\,true} \end{pmatrix},$$

where $\boldsymbol{Y}_{true} \in \{1, \ldots, K\}^n$ is a discrete random vector. We first fit a vine copula model $\hat{F}_{true}$ to the true data $\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right)$ to approximate F_{true}. The synthetic data $\left(\boldsymbol{x}_{synth}\ \boldsymbol{y}_{synth}\right)$ are then generated by sampling from $\hat{F}_{true}$. We do not seek the model $\hat{F}_{true}$, that resembles F_{true} best. Our goal is to capture the "important" characteristics of $\left(\boldsymbol{\mathcal{X}}_{true}\ \boldsymbol{Y}_{true}\right)$ in $\hat{F}_{true}$, that let a classifier learn a "good" classification rule on the synthetic data. Subsection 4.2 explains, how the quality of the classification rule estimated on the synthetic data is assessed.
As illustrated in Figure 9 the synthetic data are generated in two steps:

(1) a vine copula model $\hat{F}_{true}$ is fit to $\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right) \in \mathbb{R}^{n\times(d+1)}$,

(2) $\left(\boldsymbol{x}_{synth}\ \boldsymbol{y}_{synth}\right) \in \mathbb{R}^{m\times(d+1)}$ is sampled from $\hat{F}_{true}$.

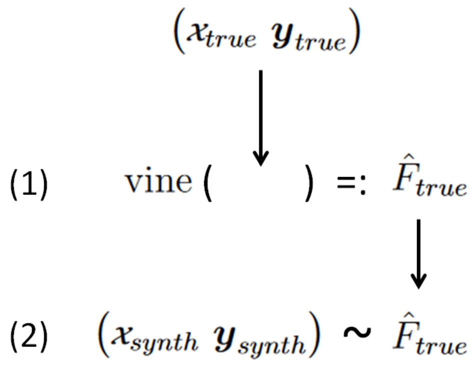
$\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right)$
(1) vine ( ) =: $\hat{F}_{true}$
(2) $\left(\boldsymbol{x}_{synth}\ \boldsymbol{y}_{synth}\right) \sim \hat{F}_{true}$

Figure 9: Generating synthetic data with vines illustrated in two steps.

A vine copula is a very flexible model: It allows to model the joint dependence separately from the marginals. The marginal distribution families again are not tied to the family of the joint distribution chosen. This is a major advantage over other graphical models such as Bayesian networks. For those, both joint and marginal distributions are typically Gaussian. Moreover a vine copula does not require conditional independence between pairs of random variables, like Markov trees do. Instead it also allows to model bivariate conditional dependence. Hence it can capture various dependence structures between pairs of random variables. In sum, a vine copula can fit complex data well in many situations.

	distribution	**samples from distribution**	**description**
methodology	F_{true}	$(\mathcal{X}_{true}\ \boldsymbol{\mathcal{Y}}_{true})$	- see Equation (4.1) - F_{true} is the distribution we assume to underlie the **true data**
	$\hat{F}_{true}$	$(\mathcal{X}_{synth}\ \boldsymbol{\mathcal{Y}}_{synth})$	- see Figure 9 - $\hat{F}_{true}$ is the distribution we sample the **synthetic data** from - $\hat{F}_{true}$ is an **estimate** of the distribution of the true data produced by a **vine copula model**

Table 4: An overview of the distributions used in the methodology of this section.

Depending on the properties of F_{true} and the structure of $(\mathcal{X}_{true}\ \boldsymbol{Y}_{true})$, it can well be, that some vine copula models are more appropriate to capture the relation between $Y_{i,true}$ and $\boldsymbol{X}_{i,true}^T$ than others.
In this project we will experiment in **Step (1)** with several vine types, in order to study, which of them is best at generating synthetic data for training a classifier, that is similar to the one trained on the true data. The type of classifier chosen may further influence which vine estimation method performs best. We compare setups 1 to 3 of Table 5.

setup	**R-vine tree sequence selection**	**pair copula estimation**
1	estimated with Dißmann	non-parametric: `tll` (transformation kernel)
2		parametric: `mle` (maximum-likelihood estimation)
3		mixed

Table 5: Vine copula estimation methods for synthetic data generation.

When fitting the vine copula model, the R-vine tree sequence $\mathcal{V}$ is selected by Dißmann's algorithm by Dissmann et al. (2013), which is shortly covered in Subsection 2.3.

The dependence between pairs of random variables, which is captured in the pair copulas, can be estimated non-parametrically, parametrically and using both approaches together. For non-parametric estimation transformation kernels, which were introduced on page 30 in Subsection 2.3, are used. For the parametric alternative we use the maximum likelihood estimator, `mle`, which can be found in Subsection 2.3 as well. It is applicable for the pair copula families independence, Gaussian, Frank, Joe, Clayton, Gumbel, Student's `t`, BB1, BB6, BB7 and BB8, which are introduced in Subsection 2.2. Then the copula family is chosen by the Akaike information criterion (AIC) of Akaike (1998), see Equation 2.54.

The inversion of Kendall's τ, `itau`, could also be used as parameter estimation method for pair copulas with one or less parameters (i.e. independence, Gaussian, Frank, Joe, Clayton, Gumbel and central Student's t). However, it is less precise, which is why we choose to use maximum likelihood estimation instead.

Furthermore it is possible to repeatedly sample from $\hat{F}_{true}$. That is why we can execute **Step (2)** several times and by this obtain $p_{synth} \in \mathbb{N}$ synthetic data sets $\left(\boldsymbol{x}_{synth}^{(l)}\ \boldsymbol{y}_{synth}^{(l)}\right),\ l \in [p_{synth}]$ from one true data set $\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right)$. Therefore we can train a chosen type of classifier on all p_{synth} synthetic data sets and assess its predictive performance on true data several times. Hence we can construct confidence intervals.

The procedure of generating $\left(\boldsymbol{x}_{synth}^{(l)}\ \boldsymbol{y}_{synth}^{(l)}\right),\ l \in [p_{synth}]$ from the data set $\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right)$ is summarized in Algorithm 3.

The sample quantile function $\hat{Q}(\cdot)$ is defined in Equation 2.4.1 of Subsection 2.4. For copula data[15] of the covariates sample quantiles of type 8 are used, whereas for copula data of the response variable we use quantiles of type 1, see Remark 2.4.1.

[15]The term *copula data* is used to speak of data on the copula scale.

Algorithm 3: Algorithm for producing $\left(\boldsymbol{x}^{(l)}_{synth}\ \boldsymbol{y}^{(l)}_{synth}\right),\ l \in [p_{synth}]$ from $\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right)$

```
    Input: $\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right) \in \mathbb{R}^{n\times(d+1)}$, $p_{synth}$, $m$, vine model specification
    Output: $\left(\boldsymbol{x}^{(l)}_{synth}\ \boldsymbol{y}^{(l)}_{synth}\right) \in \mathbb{R}^{m\times(d+1)}$, $l \in [p_{synth}]$
 1  for $j \in \{1,\dots,d\}$ do
 2      for $i \in \{1,\dots,n\}$ do
 3          $u_{ij,\,true} \leftarrow \frac{1}{n+1}\sum_{k=1}^{n} \mathbb{1}\{x_{kj,\,true} \leq x_{ij,\,true}\}$: compute pseudo-observations on copula scale
 4      end
 5  end
 6  for $i \in \{1,\dots,n\}$ do
 7      $u_{Y_i,\,true} \leftarrow \frac{1}{n}\sum_{k=1}^{n} \mathbb{1}\{y_{k,\,true} \leq y_{i,\,true}\}$: compute pseudo-observations on copula scale
 8  end
 9  $\mathcal{U}_{true} \leftarrow \begin{pmatrix} u_{11,\,true} & u_{12,\,true} & \cdots & u_{1d,\,true} & u_{Y_1,\,true} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ u_{n1,\,true} & u_{n2,\,true} & \cdots & u_{nd,\,true} & u_{Y_n,\,true} \end{pmatrix}$
10  $\widehat{vine} \leftarrow vine\left(\mathcal{U}_{true},\ \text{setup}\right)$ : fit vine copula model of chosen vine estimation method to pseudo observations
11  for $l \in \{1,\dots,p_{synth}\}$ do
12      Sample $m$ observations $(u^{(l)}_{ij,\,synth})_{i\in[m],\,j\in[d+1]} \sim \widehat{vine}$
13      for $j \in \{1,\dots,d\}$ do
14          for $i \in \{1,\dots,m\}$ do
15              $x^{(l)}_{ij,\,synth} \leftarrow \hat{Q}(u^{(l)}_{ij,\,synth})$
16          end
17      end
18      for $i \in \{1,\dots,m\}$ do
19          $y^{(l)}_{i,\,synth} \leftarrow \hat{Q}(u^{(l)}_{i(d+1),\,synth})$
20      end
21      $\left(\boldsymbol{x}^{(l)}_{synth}\ \boldsymbol{y}^{(l)}_{synth}\right) \leftarrow \begin{pmatrix} x^{(l)}_{11,\,synth} & x^{(l)}_{12,\,synth} & \cdots & x^{(l)}_{1d,\,synth} & y^{(l)}_{1,\,synth} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x^{(l)}_{m1,\,synth} & x^{(l)}_{m2,\,synth} & \cdots & x^{(l)}_{md,\,synth} & y^{(l)}_{m,\,synth} \end{pmatrix}$
22  end
23  return $\left(\boldsymbol{x}^{(l)}_{synth}\ \boldsymbol{y}^{(l)}_{synth}\right) \in \mathbb{R}^{m\times(d+1)}$, $l \in [p_{synth}]$
```

4.2 Assessing the Quality of Synthetic Data

Did we manage to produce synthetic data $\big(\boldsymbol{x}_{synth}^{(l)}\ \boldsymbol{y}_{synth}^{(l)}\big)$, $l \in [p_{synth}]$ with vines, such that a classifier trained on $\big(\boldsymbol{x}_{synth}^{(l)}\ \boldsymbol{y}_{synth}^{(l)}\big)$ learns a similar rule to the one it would have learned on data $\big(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\big)$?

To answer this question we first split true and synthetic data into training and test data (typically 70% / 30%), and denote them by:

$$\big(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\big)_{train},\ \big(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\big)_{test},\ \big(\boldsymbol{x}_{synth}^{(l)}\ \boldsymbol{y}_{synth}^{(l)}\big)_{train},\ \big(\boldsymbol{x}_{synth}^{(l)}\ \boldsymbol{y}_{synth}^{(l)}\big)_{test},\quad l \in [p_{synth}]\ ,$$

where we will in the following use the notation:

$$\big(\boldsymbol{x}_{true}^{train}\ \boldsymbol{y}_{true}^{train}\big) := \big(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\big)_{train}\ ,\quad \big(\boldsymbol{x}_{true}^{test}\ \boldsymbol{y}_{true}^{test}\big) := \big(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\big)_{test}\ ,$$

$$\big(\boldsymbol{x}_{synth}^{(l),\,train}\ \boldsymbol{y}_{synth}^{(l),\,train}\big) := \big(\boldsymbol{x}_{synth}^{(l)}\ \boldsymbol{y}_{synth}^{(l)}\big)_{train}\ ,\quad \big(\boldsymbol{x}_{synth}^{(l),\,test}\ \boldsymbol{y}_{synth}^{(l),\,test}\big) := \big(\boldsymbol{x}_{synth}^{(l)}\ \boldsymbol{y}_{synth}^{(l)}\big)_{test}$$

equivalently.

Then we train a classifier $g(\cdot)$ on both $\big(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\big)_{train}$ and each $\big(\boldsymbol{x}_{synth}^{(l)}\ \boldsymbol{y}_{synth}^{(l)}\big)_{train}$, $l \in [p_{synth}]$:

$$\hat{g}_{true}(\cdot) := g\Big(\big(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\big)_{train},\ \hat{\boldsymbol{\theta}}_{true}\Big),\quad \hat{g}_{synth}^{(l)}(\cdot) := g\Big(\big(\boldsymbol{x}_{synth}^{(l)}\ \boldsymbol{y}_{synth}^{(l)}\big)_{train},\ \hat{\boldsymbol{\theta}}_{synth}^{(l)}\Big)\ .$$

Afterwards we compute a prediction of $\boldsymbol{y}_{true}^{test}$ with $\hat{g}_{true}(\cdot)$ and $\hat{g}_{synth}^{(l)}(\cdot)$ on $\boldsymbol{x}_{true}^{test}$:

$$\hat{\boldsymbol{y}}_{true}^{test} := \hat{g}_{true}(\boldsymbol{x}_{true}^{test}),\quad \hat{\boldsymbol{y}}_{synth}^{(l),\,test} := \hat{g}_{synth}^{(l)}(\boldsymbol{x}_{true}^{test})\ ,$$

Notice that we use the same test data for $\boldsymbol{y}_{true}^{test}$ and $\hat{\boldsymbol{y}}_{synth}^{(l),\,test}$, $l \in [p_{synth}]$, which here can be thought as the new patients who need to be classified.
We compare the predictions $\hat{\boldsymbol{y}}_{true}^{test}$ and $\hat{\boldsymbol{y}}_{synth}^{(l),\,test}$ with $m(\cdot)$, a classification performance measure, which is specified in the following, (i) directly to each other as well as (ii) through the comparison to the true labels $\boldsymbol{y}_{true}^{test}$:

$$\text{(i)}\ m(\hat{\boldsymbol{y}}_{true}^{test},\ \hat{\boldsymbol{y}}_{synth}^{(l),\,test})\ ,\qquad \text{(ii)}\ |\,m(\boldsymbol{y}_{true}^{test},\ \hat{\boldsymbol{y}}_{true}^{test}) - m(\boldsymbol{y}_{true}^{test},\ \hat{\boldsymbol{y}}_{synth}^{(l),\,test})\,|\ .$$

If the vine copula model is able to capture those characteristics of $\big(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\big)$ appropriately, which induce a good classification, we assume, that (i) will score well according to the scale of the chosen $m(\cdot)$ and that (ii) will take on small values.

As an intuition, the comparison through $m(\cdot)$ enables us to set the performance of the classifier on the synthetic data into perspective. Namely, if the classification task was already hard on the true data, it will be similarly hard or even harder on the synthetic data. This intuitively makes sense because of the error we make on the way from true to synthetic data. Here we first have to select a model and then estimate its parameters in order to sample synthetic data from the model. In these two steps we introduce an error consisting of model error (selecting the wrong model) and estimation error (for the selected model: the deviation of the estimated parameter from the parameter values, which would be estimated from an infinite amount of data). Thus the comparison of classification performance measures, as proposed above, lets us see a more complete picture. Summarized, we focus not on properties of the synthetic data themselves, but on the performance of the procedures we want to use the data for.

Performance Measures

Possible choices of $m(\cdot)$ are:

(a) **confusion matrix:** We compute the confusion matrix introduced on page 53 for *dichotomous* classification problems, i.e. $K = 2$. The component value m_{kl} of the confusion matrix $M = (m_{kl})_{k,l\in[K]}$ is given by:

$$m_{kl} := \left|\{i \in [n] \mid (\hat{y}^{test}_{i,\,true}, \hat{y}^{(l),\,test}_{i,\,synth}) = (k, l),\ k, l \in [K]\}\right| . \tag{4.2}$$

Thus $\sum_{i\in[K]} m_{ii}$, the sum of diagonal elements of M, gives the number of pairs $(\hat{y}^{test}_{i,\,true}, \hat{y}^{(l),\,test}_{i,\,synth})$, where the synthetic prediction is equal to the prediction from the classifier trained on the true data.

(b) **$\mathcal{F}$-measure:** For *dichotomous* classification problems we also compute the $\mathcal{F}$-measure, introduced on page 54 and following of Subsection 3.2.

(c) **positive relative mean error (ME^{+}_{rel}) and negative relative mean error (ME^{-}_{rel}):** The sign of $(\hat{y}^{(l),\,test}_{i,\,synth} - \hat{y}^{test}_{i,\,true})$ tells us, whether the classifier trained on the synthetic data over- or underestimates the severity of patient i's disease compared to the classifier trained on the true data. This information can be important for researchers, who want base the patient's further treatment on the severity score obtained from the synthetically trained classifier. Hence instead of looking at absolute differences, we also take the positive and negative relative error (ME^{+}_{rel}, ME^{-}_{rel}) into consideration for the quality assessment of the synthetic data. It only makes sense to calculate them for *multinomial* classification problems. Let:

$$B^{+}_{l} := \{i \in [n] \mid \hat{y}^{test}_{i,\,true} < \hat{y}^{(l),\,test}_{i,\,synth}\} , \tag{4.3}$$

$$B^{-}_{l} := \{i \in [n] \mid \hat{y}^{test}_{i,\,true} > \hat{y}^{(l),\,test}_{i,\,synth}\} , \tag{4.4}$$

be the sets of indices of synthetic predictions $\hat{y}^{(l),\,test}_{i,\,synth}$, which over- or underestimate $\hat{y}^{test}_{i,\,true}$ respectively. Then the positive relative mean error:

$$ME^{+}_{rel}(\hat{\boldsymbol{y}}^{test}_{true}, \hat{\boldsymbol{y}}^{(l),\,test}_{synth}) := \frac{1}{|B^{+}_{l}|} \sum_{i\in B^{+}_{l}} \frac{|\hat{y}^{test}_{i,\,true} - \hat{y}^{(l),\,test}_{i,\,synth}|}{\hat{y}^{test}_{i,\,true}} , \tag{4.5}$$

gives the average share by which $\hat{y}^{test}_{i,\,true}$ was overestimated and the negative relative mean error:

$$ME^{-}_{rel}(\hat{\boldsymbol{y}}^{test}_{true}, \hat{\boldsymbol{y}}^{(l),\,test}_{synth}) := \frac{1}{|B^{-}_{l}|} \sum_{i\in B^{-}_{l}} \frac{|\hat{y}^{test}_{i,\,true} - \hat{y}^{(l),\,test}_{i,\,synth}|}{\hat{y}^{test}_{i,\,true}} , \tag{4.6}$$

gives the average percentage by which $\hat{y}_{i,\,true}^{test}$ was underestimated in the synthetic data set l.

(d) **multi-class AUC ($mAUC$):** We compare the multi-class generalization of the area under the receiver operating curve (mAUC) of Definition 3.21, see page 58 and following, in two ways:

(i) We compute the mAUC of $\hat{\boldsymbol{y}}_{synth}^{(l)}$ with respect to $\hat{\boldsymbol{y}}_{true}$ to directly compare the predictions:

$$mAUC_{\boldsymbol{dir}}^{(l)} := m(\hat{\boldsymbol{y}}_{true}^{test},\, \hat{\boldsymbol{y}}_{synth}^{(l),\,test}) = mAUC(\hat{\boldsymbol{y}}_{true}^{test},\, \hat{\boldsymbol{y}}_{synth}^{(l),\,test})\ , \tag{4.7}$$

(ii) Additionally, we compute the mAUC of $\hat{\boldsymbol{y}}_{true}^{test}$ with respect to $\boldsymbol{y}_{true}^{test}$ and the mAUC of $\hat{\boldsymbol{y}}_{synth}^{(l),\,test}$ with respect to $\boldsymbol{y}_{true}^{test}$ and compare the two:

$$mAUC_{\boldsymbol{indir}}^{(l)} := |mAUC(\boldsymbol{y}_{true}^{test},\, \hat{\boldsymbol{y}}_{true}^{test}) - mAUC(\boldsymbol{y}_{true}^{test},\, \hat{\boldsymbol{y}}_{synth}^{(l),\,test})|\ . \tag{4.8}$$

Figure 10 illustrates, how the quality of the synthetic data is assessed.

Rather than focusing on the best predictive performance possible, it could be more important for researches, who use the synthetic data to learn a classification rule, that this synthetic classification rule allows a similar interpretation as the classification rule estimated on the true data. In terms of for example a logistic regression with Lasso this would mean, that the β-coefficients of the synthetic classification model are similar in their values to the ones of the model trained on the true data. This implies then, that on both true and synthetic data similar features impact the response with similar strength. Hence the synthetic model also enables a meaningful interpretation. For this reason we introduce a second measure to compare synthetic and true data through classification, namely by performance of variable selection.

Measures for Variable Selection Performance

Recall from Subsection 3.1, that the multinomial logistic regression model is given by Equations (3.5) and (3.6). Let the estimated logistic Lasso models on $\big(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\big)_{train}$ and $\big(\boldsymbol{x}_{synth}^{(l)}\ \boldsymbol{y}_{synth}^{(l)}\big)_{train}$, $l \in [p_{synth}]$ be as in Equations (3.7) and (3.8) with covariates $x_j,\ j \in [d]$, that have been standardized before fitting as shown in Subsection 3.1. For both true and synthetic training data β'_{jk} is the coefficient of the standardized covariate X'_j for $Y = k$. On this basis we can compare $\hat{\beta}'_{jk,\,true}$ and $\hat{\beta}_{jk,\,synth}^{(l)}{}'$ and see, whether they allow a similar interpretation.

(a) Therefore the average distance of the logistic regression β-coefficients is computed:

$$Dist(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth}) := \frac{1}{p_{synth} \cdot K \cdot d} \sum_{l=1}^{p_{synth}} \sum_{k=1}^{K} \sum_{j=1}^{d} \left|\hat{\beta}'_{jk,\,true} - \hat{\beta}_{jk,\,synth}^{(l)}{}'\right|\ . \tag{4.9}$$

The closer (4.9) is to 0, the better we regard the performance in variable selection of the classification rule estimated data set $\big(\boldsymbol{x}_{synth}^{(l)}\ \boldsymbol{y}_{synth}^{(l)}\big)_{train}$, $l \in [p_{synth}]$.

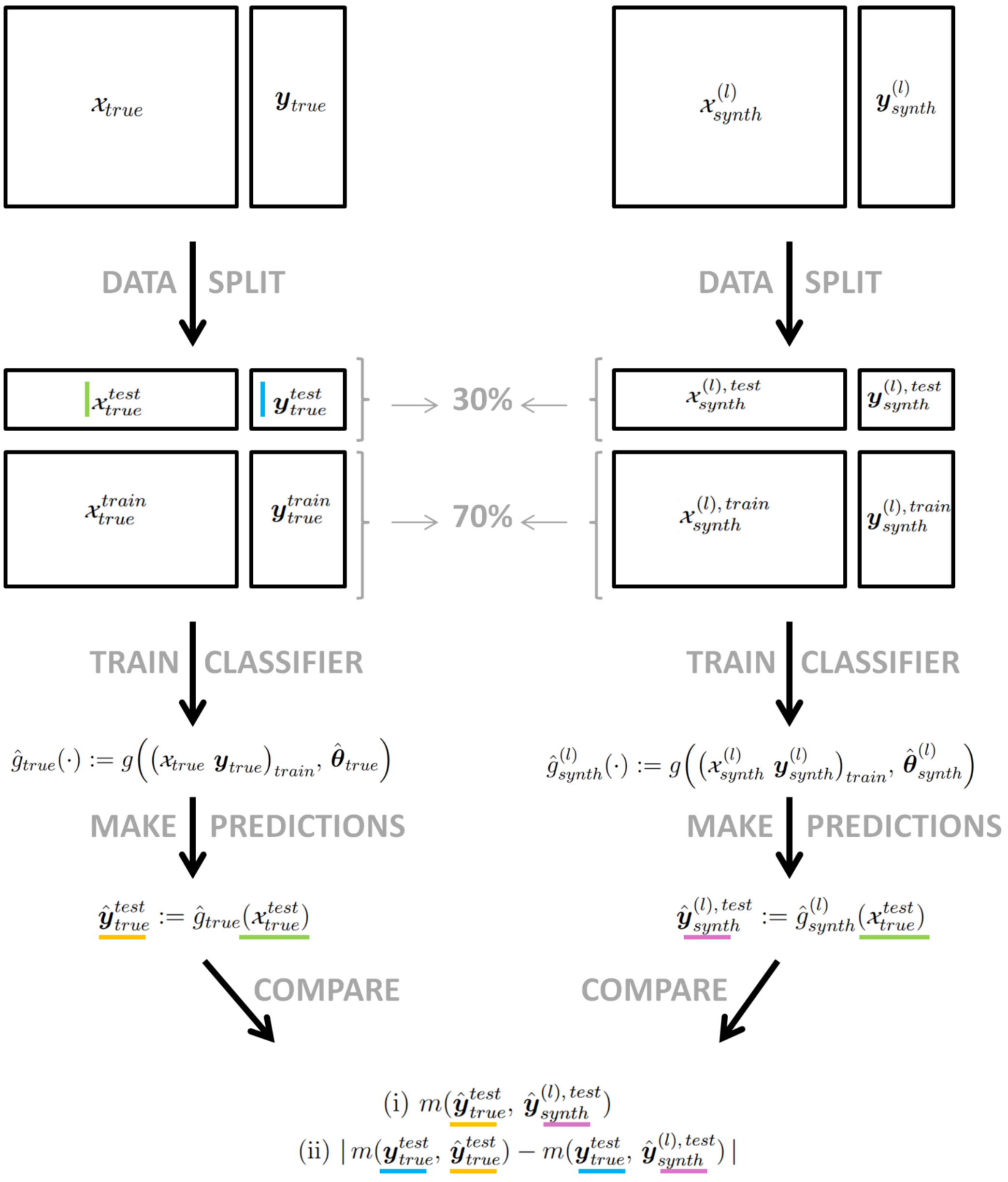
x_{true}
y_{true}
$x_{synth}^{(l)}$
$y_{synth}^{(l)}$
DATA SPLIT
DATA SPLIT
x_{true}^{test}
y_{true}^{test}
30%
$x_{synth}^{(l),test}$
$y_{synth}^{(l),test}$
x_{true}^{train}
y_{true}^{train}
70%
$x_{synth}^{(l),train}$
$y_{synth}^{(l),train}$
TRAIN CLASSIFIER
TRAIN CLASSIFIER
$\hat{g}_{true}(\cdot) := g\Big((x_{true}\ y_{true})_{train}, \hat{\theta}_{true}\Big)$
$\hat{g}_{synth}^{(l)}(\cdot) := g\Big((x_{synth}^{(l)}\ y_{synth}^{(l)})_{train}, \hat{\theta}_{synth}^{(l)}\Big)$
MAKE PREDICTIONS
MAKE PREDICTIONS
$\hat{y}_{true}^{test} := \hat{g}_{true}(x_{true}^{test})$
$\hat{y}_{synth}^{(l),test} := \hat{g}_{synth}^{(l)}(x_{true}^{test})$
COMPARE
COMPARE
(i) $m(\hat{y}_{true}^{test}, \hat{y}_{synth}^{(l),test})$
(ii) $|m(y_{true}^{test}, \hat{y}_{true}^{test}) - m(y_{true}^{test}, \hat{y}_{synth}^{(l),test})|$

Figure 10: Assessing the quality of the synthetic data produced.

(b) It is also interesting to see, whether the logistic Lasso considered a chosen covariate less (or more) influential in the synthetic data as in the true data. In other words we are interested in the cases, where for a feature X_j we have $\hat{\beta}_{j,\,synth}^{(l)}{}' = 0$ whereas it is $\hat{\beta}'_{j,\,true} \neq 0$ or the other way around. That is why we count the cases of discordant β-pairs and average them over p_{synth}. The average number of discordant β-pairs obtained is then compared to the total number of regularized β-coefficients, which is $K \cdot d$ for multinomial logistic regression, see **??**. In the binomial case the total number of regularized β-coefficients is equal to d.

$$Discor(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth}) := \frac{1}{p_{synth}} \sum_{l=1}^{p_{synth}} \sum_{k=1}^{K} \sum_{j=1}^{d} \mathbb{1}_{\{1,-1\}} \Big(\mathbb{1}_{\{0\}} \big(\hat{\beta}'_{jk,\, true} \big) - \mathbb{1}_{\{0\}} \big(\hat{\beta}^{(l)}_{jk,\, synth}{}' \big) \Big) \,, \tag{4.10}$$

with $\mathbb{1}(\cdot)$ the indicator function.

Important for the comparison of the classification performance as well as the variable selection performance is, that on $\big(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\big)$ and each $\big(\boldsymbol{x}^{(l)}_{synth}\ \boldsymbol{y}^{(l)}_{synth}\big)$, $l \in [p_{synth}]$ we use the same classifier and the same parameter tuning method. As classifiers we choose logistic regression with Lasso with penalty parameter s tuned by 10-fold cross-validation. Note, that for the calculation of $Dist(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth})$ and $Discor(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth})$ the intercept $\hat{\beta}_{0k}$ is excluded.

5 Simulation Study: Parametrically Simulated True Data

Overview

We apply the methodology introduced in Section 4 to data. We do this to answer the question:

> Given a ground truth data set with specific properties and structure, which vine copula model is most appropriate to produce synthetic data, on which the classification rule estimated is similar to the classification rule estimated on the ground truth data? Can we even formulate recommendations for the vine copula model depending on the specific classifier used?

In Section 5 we use apply our methodology on a simulated true data set $\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right)$. We obtain $\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right)$ by sampling the data $\left(\boldsymbol{x}_{true}\ \widetilde{\boldsymbol{y}}_{base}\right)$ from a distribution denoted by $\widetilde{F}$, which we specify beforehand. The response $\boldsymbol{y}_{true}$ is obtained by perturbing $\widetilde{\boldsymbol{y}}_{base}$, the response vector: $\boldsymbol{y}_{true}$ is a random forest prediction of $\widetilde{\boldsymbol{y}}_{base}$, the original sample from $\widetilde{F}$. So $\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right)$ is obtained by sampling from the distribution $\widetilde{F}$ and perturbing the response. In the following we will use the term *random forest perturbation* to refer to the random forest classifier, that is applied to produce a prediction of the original sample of the response, which was obtained from $\widetilde{F}$. This prediction is denoted by $\boldsymbol{y}_{true}$. It is the vector of response values, which we work with in the following as true response values. Taken together, we can control the relationship between quantitative $\boldsymbol{X}_{i,true}^{T}$ and ordinal $Y_{i,true}$, $i \in [n]$ through $\widetilde{F}$ and the parameters of the perturbing random forest. We aim to find out, which characteristics of $\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right)$ and which type of classifier[16] used promote which vine copula model for synthetic data generation. Therefore we tweak the parameters of $\widetilde{F}$ and the response generating random forest perturbation. Then we analyse the effect this has on the performance of the classifier $\hat{g}_{synth}^{(l)}(\cdot)$ on $\boldsymbol{x}_{true}^{test}$ compared to the performance of $\hat{g}_{true}(\cdot)$ on $\boldsymbol{x}_{true}^{test}$ for each $l \in [p_{synth}]$.

5.1 Simulation of $\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right)$

We simulate the true data $\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right) \in \mathbb{R}^{n\times(d+1)}$. They are obtained from a data set denoted by $\left(\widetilde{\boldsymbol{x}}_{base}\ \widetilde{\boldsymbol{y}}_{base}\right)$ by perturbing the response values and setting $\boldsymbol{x}_{true} := \widetilde{\boldsymbol{x}}_{base}$.

The data $\left(\widetilde{\boldsymbol{x}}_{base}\ \widetilde{\boldsymbol{y}}_{base}\right)$ themselves are sampled from the $(d+1)$-dimensional distribution $\widetilde{F}$. It is the unique distribution, that realizes the R-vine specification on $d+1$ elements $(\widetilde{\boldsymbol{F}}, \widetilde{\mathcal{V}}, \widetilde{B})$[17], which will be explicitly given in the subsequent. Then a random forest perturbation is added on $\widetilde{\boldsymbol{y}}_{base}$ to produce $\boldsymbol{y}_{true}$. In other words: $\boldsymbol{y}_{true}$ are predictions of

[16]It is not to confuse with the random forest classifier used to perturb the response $\boldsymbol{y}_{true}$ of the true data. By this we mean the classifier, that will be trained with synthetic data to deliver a classification rule, which is similar to the rule estimated on the true data.

[17]For $\widetilde{F}$ to be the unique distribution, that satisfies $(\widetilde{\boldsymbol{F}}, \widetilde{\mathcal{V}}, \widetilde{B})$ as stated in Theorem 2.3.2, we ensure, that the R-vine specification has the following properties: the pair copulas satisfy the simplifying assumption

$\widetilde{\boldsymbol{y}}_{base}$ produced by a random forest classifier. The latter is trained beforehand on the data $(\widetilde{\mathcal{X}}_{rf}\ \widetilde{\boldsymbol{y}}_{rf})$. As $(\widetilde{\mathcal{X}}_{base}\ \widetilde{\boldsymbol{y}}_{base})$, the data $(\widetilde{\mathcal{X}}_{rf}\ \widetilde{\boldsymbol{y}}_{rf})$ are sampled from $\widetilde{F}$.
Thus $(\widetilde{\mathcal{X}}_{rf}\ \widetilde{\boldsymbol{y}}_{rf})$ can be seen as a mere training data set for the random forest classifier, that perturbs $\widetilde{\boldsymbol{y}}_{base}$ to obtain $\boldsymbol{y}_{true}$. As $\boldsymbol{y}_{true}$ should be different from $\widetilde{\boldsymbol{y}}_{base}$, but its dependence on $\mathcal{X}_{true}$ should remain similar, we sample $(\widetilde{\mathcal{X}}_{rf}\ \widetilde{\boldsymbol{y}}_{rf})$ from $\widetilde{F}$ as we did for $(\widetilde{\mathcal{X}}_{base}\ \widetilde{\boldsymbol{y}}_{base})$. After the random forest has been trained on $(\widetilde{\mathcal{X}}_{rf}\ \widetilde{\boldsymbol{y}}_{rf})$, the data set will not be used again.

So as a first layer we have the data sets $(\widetilde{\mathcal{X}}_{base}\ \widetilde{\boldsymbol{y}}_{base})$ and $(\widetilde{\mathcal{X}}_{rf}\ \widetilde{\boldsymbol{y}}_{rf})$. They are both observations of the random matrix $(\widetilde{\mathcal{X}}\ \widetilde{\boldsymbol{Y}})$:

$$(\widetilde{\mathcal{X}}\ \widetilde{\boldsymbol{Y}}) := \begin{pmatrix} \widetilde{X}_{11} & \widetilde{X}_{12} & \cdots & \widetilde{X}_{1d} & \widetilde{Y}_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \widetilde{X}_{n1} & \widetilde{X}_{n2} & \cdots & \widetilde{X}_{nd} & \widetilde{Y}_n \end{pmatrix} = \begin{pmatrix} \widetilde{\boldsymbol{X}}_1^T & \widetilde{Y}_1 \\ \vdots & \vdots \\ \widetilde{\boldsymbol{X}}_n^T & \widetilde{Y}_n \end{pmatrix}, \tag{5.1}$$

$$\widetilde{\boldsymbol{X}}_i^T \in \mathbb{R}^d, \quad \widetilde{Y}_i \in \{1, 2, 3, 4, 5\}, \quad i \in [n], \tag{5.2}$$

$$\left(\widetilde{\boldsymbol{X}}_1^T\ \ \widetilde{Y}_1\right), \dots, \left(\widetilde{\boldsymbol{X}}_n^T\ \ \widetilde{Y}_n\right) \overset{i.i.d.}{\sim} \widetilde{F}, \tag{5.3}$$

$$(\widetilde{\mathcal{X}}_{base}\ \widetilde{\boldsymbol{y}}_{base}) = \begin{pmatrix} \widetilde{\boldsymbol{x}}_{1,\,base}^T & \widetilde{y}_{1,\,base} \\ \vdots & \vdots \\ \widetilde{\boldsymbol{x}}_{n,\,base}^T & \widetilde{y}_{n,\,base} \end{pmatrix} \sim \widetilde{F}, \qquad (\widetilde{\mathcal{X}}_{rf}\ \widetilde{\boldsymbol{y}}_{rf}) = \begin{pmatrix} \widetilde{\boldsymbol{x}}_{1,\,rf}^T & \widetilde{y}_{1,\,rf} \\ \vdots & \vdots \\ \widetilde{\boldsymbol{x}}_{n,\,rf}^T & \widetilde{y}_{n,\,rf} \end{pmatrix} \sim \widetilde{F}. \tag{5.4}$$

How $(\widetilde{\mathcal{X}}_{base}\ \widetilde{\boldsymbol{y}}_{base})$ and $(\widetilde{\mathcal{X}}_{rf}\ \widetilde{\boldsymbol{y}}_{rf})$ are sampled from $\widetilde{F}$ is illustrated in 5.1.

We then obtain $(\mathcal{X}_{true}\ \boldsymbol{y}_{true})$ from $(\widetilde{\mathcal{X}}_{base}\ \widetilde{\boldsymbol{y}}_{base})$ and $(\widetilde{\mathcal{X}}_{rf}\ \widetilde{\boldsymbol{y}}_{rf})$ by setting:

$$\hat{f}_{RF}(\,\cdot\,) := f_{RF}\Big((\widetilde{\mathcal{X}}_{rf}\ \widetilde{\boldsymbol{y}}_{rf}),\ \hat{\boldsymbol{\theta}}_{RF}\Big),$$

$$\boldsymbol{x}_{i,\,true}^T := \widetilde{\boldsymbol{x}}_{i,\,base}^T, \quad y_{i,\,true} := \widehat{\widetilde{y}_{i,\,base}} = \hat{f}_{RF}\big(\widetilde{\boldsymbol{x}}_{i,\,base}^T\big), \quad i \in [n],$$

$$(\mathcal{X}_{true}\ \boldsymbol{y}_{true}) := \begin{pmatrix} \boldsymbol{x}_{1,\,true}^T & y_{1,\,true} \\ \vdots & \vdots \\ \boldsymbol{x}_{n,\,true}^T & y_{n,\,true} \end{pmatrix} \in \mathbb{R}^{n\times(d+1)},$$

where $f_{RF}(\cdot)$ denotes a random forest classifier. The steps of producing $(\mathcal{X}_{true}\ \boldsymbol{y}_{true})$ are illustrated in Figure 11. Table 6 gives an overview of the distributions used in the methodology in Section 4 and in the simulation study of this section.

Here we apply the non-linear transformation $\hat{f}_{RF}(\cdot)$ to obtain $\boldsymbol{y}_{true}$ for the following reason:

and have densities.

	distribution	**samples from distribution**	**description**
methodology	F_{true}	$\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right)$	- see Equation (4.1) - F_{true} is the distribution we assume to underlie the **true data**
	$\hat{F}_{true}$	$\left(\boldsymbol{x}_{synth}\ \boldsymbol{y}_{synth}\right)$	- see Figure 9 - $\hat{F}_{true}$ is the distribution we sample the **synthetic data** from - $\hat{F}_{true}$ is an **estimate** of the distribution of the true data produced by a **vine copula model**
simulation study	$\widetilde{F}$	$\left(\widetilde{\boldsymbol{x}}_{base}\ \widetilde{\boldsymbol{y}}_{base}\right)^{(h)}$, $\left(\widetilde{\boldsymbol{x}}_{rf}\ \widetilde{\boldsymbol{y}}_{rf}\right)^{(h)}$, $h \in [p_{base}]$	- see Equations (5.1) to (5.4) and page 72 - $\widetilde{F}$ is the distribution that realizes the R-vine specification on $d+1$ elements $(\widetilde{\boldsymbol{F}}, \widetilde{\mathcal{V}}, \widetilde{B})$ - **perturbed** samples of $\widetilde{F}$ are **simulated true data** $\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right)$ used in the simulation study

Table 6: An overview of the distributions used in the methodology, Section 4 and in the simulation study of this section.

- Assume we directly set $\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right) := \left(\widetilde{\boldsymbol{x}}_{base}\ \widetilde{\boldsymbol{y}}_{base}\right)$ and do not perform a transformation. We expect a vine copula to capture the relation between $\widetilde{\boldsymbol{X}}_{i,\,base}^{T}$ and $\widetilde{Y}_{i,\,base}$ quite well, as their joint distribution $\widetilde{F}$ is itself merely governed by the R-vine specification $(\widetilde{\boldsymbol{F}}, \widetilde{\mathcal{V}}, \widetilde{B})$. Thus we expect the synthetic data generated from $\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right)$ to be very similar to $\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right)$. If this is the case, a classifier, e.g. the logistic Lasso classifier, trained on the synthetic data will learn a very similar rule to a classifier trained on $\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right)$. However, this tells us very little about how well vine copulas in general are suited to produce synthetic data for classification. The task is posed too easy.

- Also a data setup, where the data are merely generated from a vine copula model, is rather artificial with regard to real world data. In order to create more realistic data and gain more knowledge from the simulation study, we introduce an error through the non-linear transformation obtained by a random forest.

There are two things we have to note here.

Firstly, drawing only one sample $\left(\widetilde{\boldsymbol{x}}_{base}\ \widetilde{\boldsymbol{y}}_{base}\right)$ and one sample $\left(\widetilde{\boldsymbol{x}}_{rf}\ \widetilde{\boldsymbol{y}}_{rf}\right)$ from $\widetilde{F}$ to produce true data could bias our analysis: We could be lucky and end up with samples $\left(\widetilde{\boldsymbol{x}}_{base}\ \widetilde{\boldsymbol{y}}_{base}\right)$

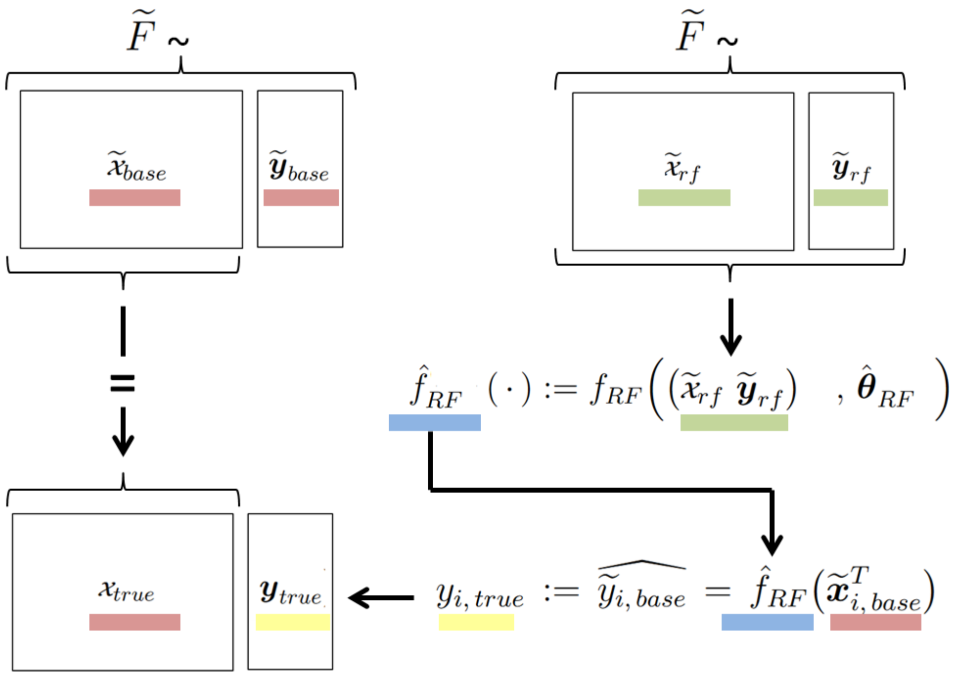
$\widetilde{F} \sim$
$\widetilde{x}_{base}$
$\widetilde{\boldsymbol{y}}_{base}$
$\widetilde{F} \sim$
$\widetilde{x}_{rf}$
$\widetilde{\boldsymbol{y}}_{rf}$
$=$
$\hat{f}_{RF}(\cdot) := f_{RF}\left((\widetilde{x}_{rf}\ \widetilde{\boldsymbol{y}}_{rf}), \hat{\boldsymbol{\theta}}_{RF}\right)$
x_{true}
$\boldsymbol{y}_{true}$
$y_{i,true} := \widehat{\widetilde{y}_{i,base}} = \hat{f}_{RF}(\widetilde{\boldsymbol{x}}_{i,base}^T)$

Figure 11: An illustration how the true data is produced.

and $(\widetilde{x}_{rf}\ \widetilde{\boldsymbol{y}}_{rf})$, such that the relation between the features in $(\widetilde{x}_{base}\ \widetilde{\boldsymbol{y}}_{base})$ and the predicted response $\boldsymbol{y}_{true}$, which is given by the random forest classifier trained on data $(\widetilde{x}_{rf}\ \widetilde{\boldsymbol{y}}_{rf})$, is captured very well by the vine we use to generate synthetic data. Or it could be the other way around and our vine copula model performs very poorly on the sampled and perturbed true data due to "poor" samples $(\widetilde{x}_{base}\ \widetilde{\boldsymbol{y}}_{base})$ and $(\widetilde{x}_{rf}\ \widetilde{\boldsymbol{y}}_{rf})$.[18] To level out this randomness we do not draw one but $p_{base} \in \mathbb{N}$ samples of $(\widetilde{x}_{base}\ \widetilde{\boldsymbol{y}}_{base})$ and $(\widetilde{x}_{rf}\ \widetilde{\boldsymbol{y}}_{rf})$ from the distribution $\widetilde{F}$:

$$(\widetilde{x}_{base}\ \widetilde{\boldsymbol{y}}_{base})^{(h)} \sim \widetilde{F}, \quad (\widetilde{x}_{rf}\ \widetilde{\boldsymbol{y}}_{rf})^{(h)} \sim \widetilde{F}, \qquad h \in [p_{base}] \ ,$$

where we use the notation $(\widetilde{x}\ \widetilde{\boldsymbol{y}})^{(*)} := (\widetilde{x}^{(*)}\ \widetilde{\boldsymbol{y}}^{(*)})$ for simplicity, compare to the notation in Section 4.

Secondly, the predictability of $Y_{i,true}$ through $\boldsymbol{X}_{i,true}^T$, $i \in [n]$ is based on the structure of $\widetilde{F}$. The amount of perturbation added on $Y_{i,true}$, $i \in [n]$ through the random forest influences this predictability on top of $\widetilde{F}$ as a second layer. For a fixed $h \in [p_{base}]$ we can therefore generate not one but $p_{true} \in \mathbb{N}$ true data sets from the samples $(\widetilde{x}_{base}\ \widetilde{\boldsymbol{y}}_{base})^{(h)}$ and $(\widetilde{x}_{rf}\ \widetilde{\boldsymbol{y}}_{rf})^{(h)}$ generated from $\widetilde{F}$. By using different parameter configurations of the random forest perturbation we can produce several predictions $\boldsymbol{y}_{true}^{(h,r)}$, $r \in [p_{true}]$. Hence we obtain several true data sets from one sample of $(\widetilde{x}_{base}\ \widetilde{\boldsymbol{y}}_{base})^{(h)}$ and $(\widetilde{x}_{rf}\ \widetilde{\boldsymbol{y}}_{rf})^{(h)}$ respectively:

$$\hat{f}_{RF}^{(h,r)}(\cdot) := f_{RF}\left((\widetilde{x}_{rf}\ \widetilde{\boldsymbol{y}}_{rf})^{(h)}, \hat{\boldsymbol{\theta}}_{RF}^{(h,r)}\right) ,$$

[18] We do not make any statement about what "good" or "poor" samples $(\widetilde{x}_{base}\ \widetilde{\boldsymbol{y}}_{base})$ and $(\widetilde{x}_{rf}\ \widetilde{\boldsymbol{y}}_{rf})$ mean in this case.

$$\boldsymbol{x}_{i,\,true}^{T\,(h,\,r)} := \widetilde{\boldsymbol{x}}_{i,\,base}^{T\,(h)}\,, \quad y_{i,\,true}^{(h,\,r)} := \widehat{\widetilde{y}_{i,\,base}}^{(h,\,r)} = \hat{f}_{RF}^{(h,\,r)}\big(\widetilde{\boldsymbol{x}}_{i,\,base}^{T\,(h)}\big), \quad i \in [n]\,,$$
$$\big(\boldsymbol{\mathcal{X}}_{true}\ \boldsymbol{y}_{true}\big)^{(h,\,r)}, \qquad h \in [p_{base}],\ r \in [p_{true}]\,.$$

By using our methodology on the different simulated $\big(\boldsymbol{\mathcal{X}}_{true}\ \boldsymbol{y}_{true}\big)^{(h,\,r)}$, we can hence analyse, which vine copula model suits a certain configuration of $\big(\boldsymbol{\mathcal{X}}_{true}\ \boldsymbol{y}_{true}\big)^{(h,\,r)}$ best to generate (in our sense) good synthetic data.

For each true data set $\big(\boldsymbol{\mathcal{X}}_{true}\ \boldsymbol{y}_{true}\big)^{(h,\,r)}$, $h \in [p_{base}]$, $r \in [p_{true}]$ we then generate p_{synth} synthetic data sets $\big(\boldsymbol{\mathcal{X}}_{synth}\ \boldsymbol{y}_{synth}\big)^{(h,\,r,\,l)}$, $l \in [p_{synth}]$ as described in Subsection 4.1.

The indices and symbols used are summarized below in Table 7. Table 8 additionally gives an overview of the size of the data sets used. Figure 12 sketches the links between the different data sets.

$$\begin{tikzcd}[column sep=tiny]
 & & & (\widetilde{\boldsymbol{F}}, \widetilde{\mathcal{V}}, \widetilde{B}) \arrow[d, Leftrightarrow] & & & \\
 & & & \widetilde{F} \arrow[dll] \arrow[drr] & & & \\
 & (\widetilde{\boldsymbol{\mathcal{X}}}_{base}\ \widetilde{\boldsymbol{y}}_{base})^{(1)},\ (\widetilde{\boldsymbol{\mathcal{X}}}_{rf}\ \widetilde{\boldsymbol{y}}_{rf})^{(1)} \arrow[dl, "\hat{f}_{RF}^{(1,1)}"'] \arrow[d, shift right] \arrow[d, shift left] \arrow[dr, "\hat{f}_{RF}^{(1,p_{true})}"] & & & & (\widetilde{\boldsymbol{\mathcal{X}}}_{base}\ \widetilde{\boldsymbol{y}}_{base})^{(p_{base})},\ (\widetilde{\boldsymbol{\mathcal{X}}}_{rf}\ \widetilde{\boldsymbol{y}}_{rf})^{(p_{base})} \arrow[dl, "\hat{f}_{RF}^{(p_{base},1)}"'] \arrow[d, shift right] \arrow[d, shift left] \arrow[dr, "\hat{f}_{RF}^{(p_{base},p_{true})}"] & \\
(\boldsymbol{\mathcal{X}}_{true}\ \boldsymbol{y}_{true})^{(1,1)} \arrow[d] & \cdots\cdots & (\boldsymbol{\mathcal{X}}_{true}\ \boldsymbol{y}_{true})^{(1,p_{true})} \arrow[d] & & (\boldsymbol{\mathcal{X}}_{true}\ \boldsymbol{y}_{true})^{(p_{base},1)} \arrow[d] & \cdots\cdots & (\boldsymbol{\mathcal{X}}_{true}\ \boldsymbol{y}_{true})^{(p_{base},p_{true})} \arrow[d] \\
\widehat{vine}^{(1,1)} \arrow[d] & \cdots\cdots & \widehat{vine}^{(1,p_{true})} \arrow[d] & & \widehat{vine}^{(p_{base},1)} \arrow[d] & \cdots\cdots & \widehat{vine}^{(p_{base},p_{true})} \arrow[d] \\
(\boldsymbol{\mathcal{X}}_{synth}\ \boldsymbol{y}_{synth})^{(1,1,1)} & & (\boldsymbol{\mathcal{X}}_{synth}\ \boldsymbol{y}_{synth})^{(1,p_{true},1)} & & (\boldsymbol{\mathcal{X}}_{synth}\ \boldsymbol{y}_{synth})^{(p_{base},1,1)} & & (\boldsymbol{\mathcal{X}}_{synth}\ \boldsymbol{y}_{synth})^{(p_{base},p_{true},1)} \\
\vdots & & \vdots & & \vdots & & \vdots \\
(\boldsymbol{\mathcal{X}}_{synth}\ \boldsymbol{y}_{synth})^{(1,1,p_{synth})} & & (\boldsymbol{\mathcal{X}}_{synth}\ \boldsymbol{y}_{synth})^{(1,p_{true},p_{synth})} & & (\boldsymbol{\mathcal{X}}_{synth}\ \boldsymbol{y}_{synth})^{(p_{base},1,p_{synth})} & & (\boldsymbol{\mathcal{X}}_{synth}\ \boldsymbol{y}_{synth})^{(p_{base},p_{true},p_{synth})}
\end{tikzcd}$$

Figure 12: An overview of the different data sets used in Subsection 5

Summarized, we specify $\widetilde{F}$ to sample $\big(\widetilde{\boldsymbol{\mathcal{X}}}_{base}\ \widetilde{\boldsymbol{y}}_{base}\big)^{(h)}$ and $\big(\widetilde{\boldsymbol{\mathcal{X}}}_{rf}\ \widetilde{\boldsymbol{y}}_{rf}\big)^{(h)}$, $h \in [p_{base}]$. After that for each $r \in [p_{true}]$ we fit $\hat{f}_{RF}^{(h,\,r)}(\,\cdot\,,\ \hat{\boldsymbol{\theta}}_{RF}^{(h,\,r)})$ to $\big(\widetilde{\boldsymbol{\mathcal{X}}}_{rf}\ \widetilde{\boldsymbol{y}}_{rf}\big)^{(h)}$ and apply it to $\big(\widetilde{\boldsymbol{\mathcal{X}}}_{base}\ \widetilde{\boldsymbol{y}}_{base}\big)^{(h)}$ to finally obtain $\big(\boldsymbol{\mathcal{X}}_{true}\ \boldsymbol{y}_{true}\big)^{(h,\,r)}$. These two steps are further elaborated below.

Index/Symbol	**Range/Value**	**Description**
n	$n = 500$	number of observations stored in the data matrices
d	$d = 100$	number of covariates $X_j,\ j \in [n]$
p_{base}	$p_{base} = 10$	number of data sets $\left(\widetilde{\boldsymbol{x}}_{base}\ \widetilde{\boldsymbol{y}}_{base}\right)^{(h)}$ and $\left(\widetilde{\boldsymbol{x}}_{rf}\ \widetilde{\boldsymbol{y}}_{rf}\right)^{(h)},\ h \in [p_{base}]$ sampled from $\widetilde{F}$
p_{true}	$p_{true} = 7$	number of true simulated data sets $\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right)^{(h,\,r)}$ $r \in [p_{true}]$ produced from $\left(\widetilde{\boldsymbol{x}}_{base}\ \widetilde{\boldsymbol{y}}_{base}\right)^{(h)}$
p_{synth}	$p_{synth} = 15$	number of synthetic data sets $\left(\boldsymbol{x}_{synth}\ \boldsymbol{y}_{synth}\right)^{(h,\,r,\,l)}$ $l \in [p_{synth}]$ produced for each true data set $\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right)^{(h,\,r)}$
i	$i \in [n]$	row index of data matrices indexing observations
j	$j \in [d]$	column index of data matrices indexing covariates X_j
h	$h \in [p_{base}]$	index for base data sets $\left(\widetilde{\boldsymbol{x}}_{base}\ \widetilde{\boldsymbol{y}}_{base}\right)^{(h)}$
r	$r \in [p_{true}]$	index for simulated true data sets $\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right)^{(h,\,r)}$ under a random forest specification r
l	$l \in [p_{synth}]$	index for synthetic data sets produced from one true data set $\left(\boldsymbol{x}_{synth}\ \boldsymbol{y}_{synth}\right)^{(h,\,r,\,l)}$

Table 7: Table of symbols and indices.

Sampling $\left(\widetilde{\boldsymbol{x}}_{base}\ \widetilde{\boldsymbol{y}}_{base}\right)^{(h)}$ and $\left(\widetilde{\boldsymbol{x}}_{rf}\ \widetilde{\boldsymbol{y}}_{rf}\right)^{(h)}$ from $\widetilde{F}$:

Recall the following:

- We set:
$$\left(\widetilde{\boldsymbol{X}}_i^T \quad \widetilde{Y}_i\right) \sim \widetilde{F} \quad \forall i \in [n]\ .$$
- Here $\widetilde{F}$ is the unique distribution, which realizes the R-vine specification $(\widetilde{\boldsymbol{F}}, \widetilde{\mathcal{V}}, \widetilde{B})$.
- The data sets $\left(\widetilde{\boldsymbol{x}}_{base}\ \widetilde{\boldsymbol{y}}_{base}\right)^{(h)}$ and $\left(\widetilde{\boldsymbol{x}}_{rf}\ \widetilde{\boldsymbol{y}}_{rf}\right)^{(h)},\ h \in [p_{base}]$ are both realizations of $\left(\widetilde{\boldsymbol{X}}_i^T \quad \widetilde{Y}_i\right) \sim \widetilde{F} \quad \forall i \in [n]$ with:
  - $\left(\widetilde{\boldsymbol{x}}_{base}\ \widetilde{\boldsymbol{y}}_{base}\right)^{(h)}$ used as a basis for $\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right)^{(h,\,r)},\ h \in [p_{base}],\ r \in [p_{true}]$,
  - $\left(\widetilde{\boldsymbol{x}}_{rf}\ \widetilde{\boldsymbol{y}}_{rf}\right)^{(h)}$ used to train the perturbing random forest $\hat{f}_{RF}^{(h,\,r)}(\,\cdot\,) := f_{RF}\Big(\left(\widetilde{\boldsymbol{x}}_{rf}\ \widetilde{\boldsymbol{y}}_{rf}\right)^{(h)},\ \hat{\boldsymbol{\theta}}_{RF}^{(h,\,r)}\Big)$ for obtaining $\boldsymbol{y}_{true}^{(h,\,r)}$.

Data set and dimension	**Description**
$\left(\widetilde{\boldsymbol{x}}_{base}\ \widetilde{\boldsymbol{y}}_{base}\right)^{(h)} \in \mathbb{R}^{500\times 101}$, $h \in [10]$	data set, of which the true data is a version
$\left(\widetilde{\boldsymbol{x}}_{rf}\ \widetilde{\boldsymbol{y}}_{rf}\right)^{(h)} \in \mathbb{R}^{500\times 101}$, $h \in [10]$	data set used to train the random forest classifier, which is used to perturb $\widetilde{\boldsymbol{y}}_{base}^{(h)}$
$\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right)^{(h,r)} \in \mathbb{R}^{500\times 101}$, $h \in [10],\ r \in [7]$	simulated true data set, on which we use the methodology introduced in Section 4
$\left(\boldsymbol{x}_{synth}\ \boldsymbol{y}_{synth}\right)^{(h,r,l)} \in \mathbb{R}^{500\times 101}$, $h \in [10],\ r \in [7],\ l \in [15]$	synthetic data set produced from $\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right)^{(h,r)}$, on which a classifier will be trained

Table 8: Table indicating dimensions of data sets used.

So through the R-vine specification $(\widetilde{\boldsymbol{F}}, \widetilde{\mathcal{V}}, \widetilde{B})$ we specify $\widetilde{F}$, from which we sample $\left(\widetilde{\boldsymbol{x}}_{base}\ \widetilde{\boldsymbol{y}}_{base}\right)^{(h)}$ and $\left(\widetilde{\boldsymbol{x}}_{rf}\ \widetilde{\boldsymbol{y}}_{rf}\right)^{(h)}$, $h \in [p_{base}]$ through the following steps:

(a) **Sampling copula data:**
First we sample copula data from $(\boldsymbol{I}, \widetilde{\mathcal{V}}, \widetilde{B})$, where the symbol $\boldsymbol{I} := (U[0,1], \ldots, U[0,1])$ denotes the vector of uniform distributions on $[0,1]$. This is done as shown in Subsection **Simulating from Regular Vine** on page 32. Therefore we specify (1) the R-vine tree sequence $\widetilde{\mathcal{V}}$ and (2) set of pair copulas $\widetilde{B}$.

(b) **Transform copula data of covariates to original scale:**
For this we then apply the inverse of the corresponding marginal distribution, i.e. the respective quantile function, to each covariate. Therefore (3) the vector of marginal distributions $\widetilde{\boldsymbol{F}}$ is given.

(c) **Transform copula data of response to original scale through thresholding:**
As a last step we obtain the observations of the response variable through thresholding the corresponding copula data:

$$\widetilde{Y}_i := k\ , \qquad \text{if} \quad \widetilde{u}_{i,(d+1)} \in [\frac{k-1}{K}, \frac{k}{K}]\ , \quad k \in [K]\ .$$

We set $K = 5$.

We give (1) the R-vine tree sequence $\widetilde{\mathcal{V}}$, (2) the set of pair copulas $\widetilde{B}$ and (3) the vector of marginal distributions $\widetilde{\boldsymbol{F}}$:

(1) **R-vine tree sequence $\widetilde{\mathcal{V}}$:**
We choose $\widetilde{\mathcal{V}}$ to be a star shaped C-vine with root nodes:

$$R := \left\{\widetilde{Y}_i,\ (\widetilde{Y}_i,\ \widetilde{X}_{id}),\ (\widetilde{Y}_i,\ \widetilde{X}_{id};\ \widetilde{Y}_i),\ \right\} \cup \left\{(\widetilde{Y}_i,\ \widetilde{X}_{ij};\ \widetilde{X}_{i(j+1)}, \ldots, \widetilde{X}_{id},\ \widetilde{Y}_i) \mid j \in [d-1]\right\}\ . \tag{5.5}$$

- *Why?*
  The reason for this is, that we want to enforce the response $\widetilde{Y}_i$ to depend on (some of) the covariates $\widetilde{X}_{ij},\ j \in [d]$. This then allows to predict $\widetilde{Y}_i$ from $\widetilde{\boldsymbol{X}}_i^T$, which makes a classification task meaningful.
- *How does it work?*
  Through the C-vine structure of $\widetilde{\mathcal{V}}$ together with the root nodes given in R we can directly govern the pairwise dependence between $\widetilde{Y}_i$ and each covariate $\widetilde{X}_{ij},\ j \in [d]$: In T_1 each $\widetilde{X}_{ij},\ j \in [d]$ is connected to $\widetilde{Y}_i$ by an edge $e \in E_1$, which corresponds to the pair copula $C_e \in \widetilde{B}$. We can set the pairwise dependence between $\widetilde{X}_{ij},\ j \in [d]$ and $\widetilde{Y}_i$ by specifying the pair copula C_e with $e \in E_1$.
- *How does it look like?*
  Recall from (2.49) of Remark 2.3.5 the special band structure of the R-vine matrix M of C-vine. By setting d, R in (5.5) and the order of the random vector to $\left(\widetilde{\boldsymbol{X}}_i^T\ \ \widetilde{Y}_i\right) \in \mathbb{R}^d \times \{1,2,3,4,5\}$ with $\widetilde{Y}_i$ in the last component, the R-vine matrix M_C in (2.49) represents $\widetilde{\mathcal{V}}$.
- *Example:*
  To illustrate this we give an example with $d = 8$. Here $\widetilde{\mathcal{V}}$ is represented by the R-vine matrix M in (5.6). The first tree T_1 of the R-vine tree sequence is displayed in Figure 13, where the index 9 corresponds to Y and indices $1,\dots,8$ correspond to covariates $X_1,\dots,X_8$.

$$M = \begin{pmatrix} 1 &&&&&&&& \\ 2 & 2 &&&&&&& \\ 3 & 3 & 3 &&&&&& \\ 4 & 4 & 4 & 4 &&&&& \\ 5 & 5 & 5 & 5 & 5 &&&& \\ 6 & 6 & 6 & 6 & 6 & 6 &&& \\ 7 & 7 & 7 & 7 & 7 & 7 & 7 && \\ 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & \\ 9 & 9 & 9 & 9 & 9 & 9 & 9 & 9 & 9 \end{pmatrix}. \tag{5.6}$$

(2) **Set of pair copulas $\widetilde{B}$:**
We specify $C_e \in \widetilde{B}$ by setting their copula families, corresponding parameter(s) and rotations.

We require:

- a *sparse setting*: 20% of the covariates have a significant influence on the response variable, the remaining 80% of the covariates have low to no association with the response variable and
- some *dependence* within these 20% influential covariates.

→ *Hence:* We decide on the first covariates $\widetilde{X}_{i1},\dots,\widetilde{X}_{i20}$ of the $d = 100$ covariates to have a high association with $\widetilde{Y}_i$.[19]

[19]This is w.l.o.g. as we can reorder the covariates as we wish.

(T1)

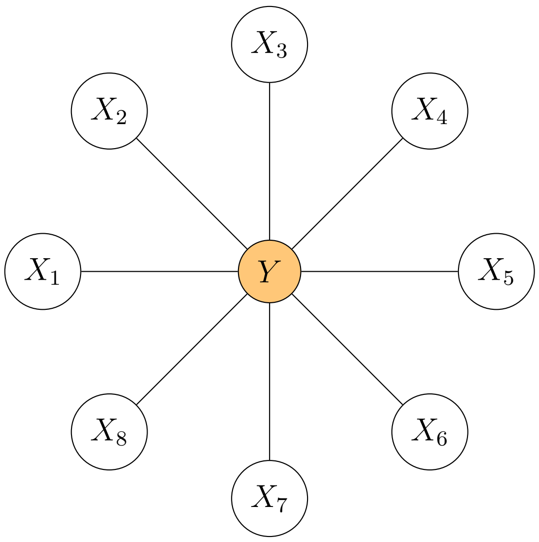

Figure 13: An example of the first tree T_1 of a C-vine $\mathcal{V}$ of a random vector $(X_1, X_2, ...X_8, Y)$. Here Y is the central node of the star.

The pair copulas are specified through their *families*, their *rotations* and their *parameter(s)* as follows:

(i) For each pair copula $(\widetilde{X}_{ij}, \widetilde{Y}_i)$ with $j \in [20]$:

- *family:* Its family is drawn uniformly from the set:

$$A_{copula} := \{\text{Gaussian, Clayton, Joe, Gumbel, Frank}\} .$$

- *rotation:* Similarly, its rotation is sampled uniformly taking on values in:

$$A_{rotation} := \{0°, 90°, 180°, 270°\} .$$

- *parameter:* The inversion of Kendall's τ is applicable. Hence, for a fixed pair copula the parameter is then obtained by:
  - The Kendall's τ value is uniformly sampled from the interval $[0.6, 0.8]$.
  - According to the rotation of the pair copula $sign(\tau)$ is adjusted.
  - Then:
  $$\theta := k^{-1}_{a_{copula}}(\tau) ,$$
  with $a_{copula} \in A_{copula}$.

(ii) For each pair copula $(\widetilde{X}_{ij}, \widetilde{X}_{ik}\,;\, \widetilde{X}_{ik+1}, \ldots, \widetilde{X}_{i100}, \widetilde{Y}_i)$ with $j, k \in [20],\ j < k$:

- *family:* As before, its family is drawn uniformly from the set A_{copula}.
- *rotation:* Similarly, its rotation is sampled uniformly taking on values in $A_{rotation}$.
- *parameter:* The inversion of Kendall's τ is again applicable. Hence, for a fixed pair copula the parameter is then obtained by:
  - The Kendall's τ value is uniformly sampled from the interval $[0.4, 0.7]$.
  - According to the rotation of the pair copula $sign(\tau)$ is adjusted.

Distribution family $(F_\theta)_{\theta\in\Theta}$	**Parameter space** Θ_F
normal	$(\mu,\sigma) \in [-10,10] \times [1,4]$
beta	$(\alpha,\beta) \in [1,20] \times [0.3,10]$
gamma	$(\alpha,\frac{1}{\sigma}) \in [2,10] \times [1,2]$
Student's t	$df \in \{1,2,...,15\}$
exponential	$\lambda \in [0.2,3]$
uniform	$(min,max) \in [-100,20] \times [20,100]$

Table 9: Prespecified parameter spaces for the different distribution families. For the gamma distribution family α is the shape and $\frac{1}{\sigma}$ the rate parameter with σ the scale parameter.

* Then:

$$\theta := k^{-1}_{a_{copula}}(\tau) \ ,$$

with $a_{copula} \in A_{copula}$.

(iii) For all the remaining pair copulas:

- *family:* The family is sampled to be either independence with 80% probability or Gauss with 20%.
- *rotation:* For these two pair copula families rotation does not make sense.
- *parameter:* For the Gauss pair copulas the inversion of Kendall's τ is again applicable, which is why:
  * The Kendall's τ value is uniformly sampled from the interval $[0,0.3]$.
  * Again:

$$\theta := k^{-1}_{a_{copula}}(\tau) \ ,$$

with $a_{copula} \in A_{copula}$.

(3) **Vector of marginal distributions $\widetilde{\boldsymbol{F}}$:**
The marginal distribution of each $\widetilde{X}_{ij},\ j \in [d]$ is randomly selected:

- First the distribution family $(F_\theta)_{\theta\in\Theta}$ is sampled discrete uniformly from:

$$A_{dist} := \{\text{normal, beta, gamma, Student's t, exponential, uniform}\} \ .$$

- Then the respective marginal parameter vector $\boldsymbol{\theta} \in \mathbb{R}^k$ of the chosen distribution family is sampled uniformly from the prespecified parameter space $\Theta_F \subset \mathbb{R}^k$, see Table 9.

Algorithm 4 schematizes, how $\left(\widetilde{\mathbf{x}}_{base}\ \widetilde{\mathbf{y}}_{base}\right)$ and $\left(\widetilde{\mathbf{x}}_{rf}\ \widetilde{\mathbf{y}}_{rf}\right)$ are generated.

$\widetilde{F}$ is specified through $(\widetilde{\boldsymbol{F}},\widetilde{\mathcal{V}},\widetilde{B})$. Its specific value can be found on page **??** of the Appendix 9.

An alternative way of generating $\left(\widetilde{\boldsymbol{x}}_{base}\ \widetilde{\boldsymbol{y}}_{base}\right)$ and $\left(\widetilde{\boldsymbol{x}}_{rf}\ \widetilde{\boldsymbol{y}}_{rf}\right)$ is for example to sample both from a $(d+1)$-dimensional Gauss distribution, where the mean $\boldsymbol{\mu} \in \mathbb{R}^{d+1}$ and the covariance matrix $\Sigma \in \mathbb{R}^{(d+1)\times(d+1)}$ are randomly selected.[20] However, with the joint distribution being a multivariate Gauss the marginal distributions are set to be also Gauss. To create more complex data and pose an interesting task, we generate $\left(\widetilde{\boldsymbol{x}}_{base}\ \widetilde{\boldsymbol{y}}_{base}\right)$ and $\left(\widetilde{\boldsymbol{x}}_{rf}\ \widetilde{\boldsymbol{y}}_{rf}\right)$ as described above.

Fitting the Random Forest Classifier $\hat{f}_{RF}^{(h,r)}(\,\cdot\,, \hat{\boldsymbol{\theta}}_{RF}^{(h,r)}), \quad h \in [p_{base}],\ r \in [p_{true}]$:

Recall from Section 3.1, that to fit a random forest classifier f_{RF} to $\left(\widetilde{x}_{rf}\ \widetilde{y}_{rf}\right)^{(h)}$ we need to tune or set its parameters $\boldsymbol{\theta}_{RF}^{(h,r)} := (n_{tree}^{(h,r)}, m_{try}^{(h,r)}, s_{nodesize}^{(h,r)})$, see page **??** and following:

- n_{tree}: The number of trees in the random forest is estimated with the out-of-bag (OOB) error estimate, see 3.1.3. The number of trees, for which this value stabilizes, will be picked. We set the maximum number of trees to 2200.
- m_{try}: The number of covariates randomly selected from all d covariates to choose the current split variable and spilt point from. It is set to $m_{try} = \lfloor\sqrt{d}\rfloor$, which is recommended by Friedman et al. (2001) and the default value in the `R`-package `randomForest`, see Liaw and Wiener (2002).
- $s_{nodesize} \in \mathbb{N}$: The minimal nodesize of all terminal nodes in each classification tree. We set $s_{nodesize} = 1$, which corresponds to the default value in the `R`-package `randomForest`, see Liaw and Wiener (2002).

So as a first step we tune $\boldsymbol{\theta}_{RF}$ "optimally" and obtain the values $\hat{\boldsymbol{\theta}}_{RF}^{*(h)} = (\hat{n}_{tree}^{*(h)}, \hat{m}_{try}^{*}, \hat{s}_{nodesize}^{*})$. By "optimal" we mean, that $(n_{tree}, m_{try}, s_{nodesize})$ are set to their recommended default value or tuned according to best practice, see Section 3. With this we set:

$$\hat{\boldsymbol{\theta}}_{RF}^{(h,1)} := \hat{\boldsymbol{\theta}}_{RF}^{*(h)}, \quad h \in [p_{base}]\ ,$$

$$\hat{f}_{RF}^{(h,1)}(\cdot) := f_{RF}\Big(\left(\widetilde{\boldsymbol{x}}_{rf}\ \widetilde{\boldsymbol{y}}_{rf}\right)^{(h)}, \hat{\boldsymbol{\theta}}_{RF}^{(h,1)}\Big)\ ,$$

and:

$$\boldsymbol{y}_{true}^{(h,1)} := \hat{f}_{RF}^{(h,1)}\Big(\widetilde{\boldsymbol{x}}_{base}^{(h)}\Big)\ .$$

So for each $h \in [p_{base}]$ the first version of the true data $\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right)^{(h,1)}$ is the least perturbed one.

Starting from $\hat{\boldsymbol{\theta}}_{RF}^{(h,1)}$ we change the parameter values to get the remaining $\hat{\boldsymbol{\theta}}_{RF}^{(h,r)}$, $r \in [p_{true}] \setminus \{1\}$.

By gradually perturbing $\boldsymbol{y}_{true}^{(h,r)}$ we gradually "worsen" one of the parameters n_{tree}, m_{try} or $s_{nodesize}$ at a time, while keeping the remaining to their optimal value $\hat{n}_{tree}^{*(h)}$, $\hat{m}_{try}^{*}$ and $\hat{s}_{nodesize}^{*}$. This means:

[20]This corresponds to a vine copula, where all pair copulas and marginal distributions are Gauss.

- In each iteration step we either reduce n_{tree} to take on the non-optimal values:

$$\text{for } r \in \{2,3\} \ : \qquad n_{tree}^{(h,r)} \in \{650, 1300\}\ .$$

  Recall, that $n_{tree}^{(h,1)} = \hat{n}_{tree}^{*\,(h)}$ is the OOB estimate of the optimal number of trees in the random forest. Thus $n_{tree}^{(h,r)} \in \{650, 1300, \hat{n}_{tree}^{*\,(h)}\}$, $r \in \{1,2,3\}$. The remaining parameters are fixed to their optimal/tuned values:

$$\text{for } r \in \{1,2,3\} \ : \qquad \left(m_{try}^{(h,r)},\, s_{nodesize}^{(h,r)}\right) := \left(m_{try}^{*},\, s_{nodesize}^{*}\right) = (10, 1)\ .$$

- Or, starting from the value $m_{try}^{(h,1)} = \lfloor\sqrt{d}\rfloor = 10$, we reduce m_{try} to take on values:

$$\text{for } r \in \{4,5\} \ : \qquad m_{try}^{(h,r)} \in \{3, 6\}\ .$$

  The remaining parameters are fixed to their optimal/tuned values:

$$\text{for } r \in \{4,5\} \ : \qquad \left(n_{tree}^{(h,r)},\, s_{nodesize}^{(h,r)}\right) := \left(n_{tree}^{*\,(h)},\, s_{nodesize}^{*}\right) = \left(n_{tree}^{*\,(h)}, 1\right)\ .$$

- As a last possibility we increase $s_{nodesize}$ from the starting value $s_{nodesize}^{(h,1)} = 1$ to take on values:

$$\text{for } r \in \{6,7\} \ : \qquad s_{nodesize}^{(h,r)} \in \{50, 100\}\ .$$

  Again the remaining parameters are fixed to their optimal/tuned values:

$$\text{for } r \in \{6,7\} \ : \qquad \left(n_{tree}^{(h,r)},\, m_{try}^{(h,r)}\right) := \left(n_{tree}^{*\,(h)},\, m_{try}^{*}\right) = \left(n_{tree}^{*\,(h)}, 10\right)\ .$$

Table 10 summarizes the different parameters $\hat{\boldsymbol{\theta}}_{RF}^{(h,r)}$, $r \in [p_{true}]$ we use for a fixed $h \in p_{base}$ for the random forest perturbation:

$r \in [p_{true}]$	$(n_{tree}^{(h,r)},\, m_{try}^{(h,r)},\, s_{nodesize}^{(h,r)})$
1	$(n_{tree}^{*\,(h)}, 10, 1)$
2	$(1300, 10, 1)$
3	$(650, 10, 1)$
4	$(n_{tree}^{*\,(h)}, 6, 1)$
5	$(n_{tree}^{*\,(h)}, 3, 1)$
6	$(n_{tree}^{*\,(h)}, 10, 50)$
7	$(n_{tree}^{*\,(h)}, 10, 100)$

Table 10: Values of the parameter of the perturbing random forest $\hat{\boldsymbol{\theta}}_{RF}^{(h,r)}$, $r \in [p_{true}]$ for a fixed $h \in [p_{base}]$. Recall, that $\hat{n}_{tree}^{*\,(h)}$ is the OOB estimate of the optimal number of trees in the random forest, see 3.1.3.

Table 11 gives the optimal number of trees in the random forest $n_{tree}^{*\,(h)}$ obtain as the OOB error estimate for each $h \in p_{base}$.

$h \in [p_{base}]$	$n_{tree}^{*\,(h)}$	$h \in [p_{base}]$	$n_{tree}^{*\,(h)}$
1	$n_{tree}^{*\,(1)} = 2100$	6	$n_{tree}^{*\,(6)} = 2000$
2	$n_{tree}^{*\,(2)} = 2000$	7	$n_{tree}^{*\,(7)} = 1300$
3	$n_{tree}^{*\,(3)} = 2000$	8	$n_{tree}^{*\,(8)} = 2200$
4	$n_{tree}^{*\,(4)} = 2000$	9	$n_{tree}^{*\,(9)} = 2200$
5	$n_{tree}^{*\,(5)} = 1100$	10	$n_{tree}^{*\,(10)} = 2000$

Table 11: Values of $n_{tree}^{*\,(h)}$ for $h \in [p_{base}]$, the OOB error estimate, see Subsection **Random Forests** on page 52 .

With $\hat{\boldsymbol{\theta}}_{RF}^{(h,r)}$ set, the classifier f_{RF} is refit:

$$\hat{f}_{RF}^{(h,r)}(\cdot) := f_{RF}\left(\left(\widetilde{\boldsymbol{x}}_{rf}\ \widetilde{\boldsymbol{y}}_{rf}\right)^{(h)},\ \hat{\boldsymbol{\theta}}_{RF}^{(h,r)}\right),$$

and:

$$\boldsymbol{y}_{true}^{(h,r)} := \hat{f}_{RF}^{(h,r)}\left(\widetilde{\boldsymbol{x}}_{base}^{(h)}\right),$$

for $h \in [p_{base}]$ and $r \in [p_{true}] \setminus \{1\}$.

To simplify the interpretation we introduce the some nomenclature. We speak of a true data set with *low*, *medium* or *high* perturbation introduced on the response variable, if the random forest parameter currently varied has the following values given in Table 12:

	perturbation:		
	low	**medium**	**high**
n_{tree}	$n_{tree}^{*\,(h)}$	1300	650
m_{try}	10	6	3
$s_{nodesize}$	1	50	100

Table 12: Nomenclature regarding perturbation introduced on the simulated true data set depending on the value of the parameter currently varied. Recall, that $\hat{n}_{tree}^{*\,(h)}$ is the OOB estimate of the optimal number of trees in the random forest, see 3.1.3.

Table 13 gives again the parameter values $\hat{\boldsymbol{\theta}}_{RF}^{(h,r)}$, $r \in [p_{true}]$ of the perturbing random forest for a fixed $h \in [p_{base}]$ and links them to the nomenclature introduced in Table 12.

$r \in [p_{true}]$	$(n_{tree}^{(h,r)}, m_{try}^{(h,r)}, s_{nodesize}^{(h,r)})$	**perturbation**
1	$(n_{tree}^{*(h)}, 10, 1)$	low
2	$(1300, 10, 1)$	medium
3	$(650, 10, 1)$	high
4	$(n_{tree}^{*(h)}, 6, 1)$	medium
5	$(n_{tree}^{*(h)}, 3, 1)$	high
6	$(n_{tree}^{*(h)}, 10, 50)$	medium
7	$(n_{tree}^{*(h)}, 10, 100)$	high

Table 13: Values of the parameter of the perturbing random forest $\hat{\boldsymbol{\theta}}_{RF}^{(h,r)}$, $r \in [p_{true}]$ for a fixed $h \in [p_{base}]$. The perturbation introduced on the response $\boldsymbol{y}_{true}^{(h,r)}$ of the simulated true data set $\left(x_{true}\ \boldsymbol{y}_{true}\right)^{(h,r)}$ is **low**, **medium** or **high** depending on the parameter $\hat{\boldsymbol{\theta}}_{RF}^{(h,r)}$ given above, $r \in [p_{true}]$, $h \in [p_{base}]$. Recall, that $\hat{n}_{tree}^{*(h)}$ is the OOB estimate of the optimal number of trees in the random forest, see 3.1.3.

Summarized we obtain the following number of data sets given in Table 14 for low, medium and high perturbation respectively.

perturbation	**no. of true data sets**	**no. of synthetic data sets**
low	$10\ (= p_{base} \cdot 1)$	$150\ (= p_{base} \cdot 1 \cdot p_{synth})$
medium	$30\ (= p_{base} \cdot 3)$	$450\ (= p_{base} \cdot 3 \cdot p_{synth})$
high	$30\ (= p_{base} \cdot 3)$	$450\ (= p_{base} \cdot 3 \cdot p_{synth})$

Table 14: Total number of true and corresponding synthetic data sets per **low**, **medium** and **high** perturbation respectively. Recall from Table 7, that $\left(x_{true}\ \boldsymbol{y}_{true}\right)^{(h,r)}$ and $\left(x_{synth}\ \boldsymbol{y}_{synth}\right)^{(h,r,l)}$ with $r \in [p_{true}]$, $h \in [p_{base}]$, $l \in [p_{synth}]$ and $p_{base} = 10$, $p_{true} = 7$, $p_{synth} = 15$.

The entire process of generating the true data sets can be traced in Algorithm 5.

Algorithm 4: Algorithm for producing $\left(\widetilde{\boldsymbol{x}}_{base}\ \widetilde{\boldsymbol{y}}_{base}\right)$ and $\left(\widetilde{\boldsymbol{x}}_{rf}\ \widetilde{\boldsymbol{y}}_{rf}\right)$.

```
Input: $n$, $d$, $A_{dist}$, $\Theta_F$ for all $F \in A_{dist}$, $A_{copula}$, $A_{rotation}$, root nodes $R$ and
       ordering $\left(\widetilde{\boldsymbol{X}}_i^T \quad \widetilde{Y}_i\right)$
Output: $\left(\widetilde{\boldsymbol{x}}_{base}\ \widetilde{\boldsymbol{y}}_{base}\right)$, $\left(\widetilde{\boldsymbol{x}}_{rf}\ \widetilde{\boldsymbol{y}}_{rf}\right)$
1  for $j \in [d]$ do
2      $F_j \leftarrow$ distribution family sampled from $A_{dist}$
3      $\boldsymbol{\theta}_j \leftarrow$ parameter vector of $F_j$ sampled from $\Theta_{F_j}$
4  end
5  $\widetilde{\boldsymbol{F}} \leftarrow (F_{1,\boldsymbol{\theta}_1}, \dots, F_{d,\boldsymbol{\theta}_d})$
6  $\widetilde{\mathcal{V}} \leftarrow$ C-vine on $d+1$ elements according to $R$
7  for each tree $T_i = (V_i, E_i) \in \mathcal{V}$ do
8      for each pair copula $C_e$ with edge $e \in E_i$ do
9          if e corresponds to $(\widetilde{X}_{ij}, \widetilde{Y}_i)$ with $j \in [20]$ then
10             $family_e \leftarrow$ copula family sampled from $A_{copula}$
11             $rotation_e \leftarrow$ rotation sampled from $A_{rotation}$
12             $|\tau_e| \leftarrow$ Kendall's $\tau$ sampled from $[0.6, 0.8]$
13         else if e corresponds to $(\widetilde{X}_{ij}, \widetilde{X}_{ik}\,;\, \widetilde{X}_{k+1}, \dots, \widetilde{X}_{i100}, \widetilde{Y}_i)$ with
            $j, k \in [20],\ j < k$ then
14             $family_e \leftarrow$ copula family sampled from $A_{copula}$
15             $rotation_e \leftarrow$ rotation sampled from $A_{rotation}$
16             $|\tau_e| \leftarrow$ Kendall's $\tau$ sampled from $[0.4, 0.7]$
17         else
18             $family_e \leftarrow$ Gauss with $p^* = 0.2$, independence with $1 - p^* = 0.8$
19             $rotation_e \leftarrow 0$
20             $|\tau_e| \leftarrow$ Kendall's $\tau$ sampled from $[0, 0.3]$
21         end
22         $sign(\tau_e) \leftarrow$ sign according to $rotation_e$ for $family_e \neq$ independence
23         $\theta_e \leftarrow k^{-1}(\tau_e)$
24         $C_e \leftarrow$ pair copula of $family_e$ with $rotation_e$ and parameter $\theta_e$
25     end
26 end
27 $\widetilde{B} \leftarrow \{C_e \mid e \in E_i$ for $i \in [d]\}$
28 $\widetilde{\mathcal{U}}_{base} \leftarrow n$ from $(d+1)$-dimensional copula $C$ specified by $(\boldsymbol{I}, \widetilde{\mathcal{V}}, \widetilde{B})$
29 $\widetilde{\mathcal{U}}_{rf} \leftarrow n$ from $(d+1)$-dimensional copula $C$ specified by $(\boldsymbol{I}, \widetilde{\mathcal{V}}, \widetilde{B})$
30 for $j \in [d]$ do
31     $\widetilde{\boldsymbol{x}}_{j,\,base} \leftarrow F^{-1}_{j,\boldsymbol{\theta}_j}(\widetilde{\boldsymbol{u}}_{j,\,base})$
32     $\widetilde{\boldsymbol{x}}_{j,\,rf} \leftarrow F^{-1}_{j,\boldsymbol{\theta}_j}(\widetilde{\boldsymbol{u}}_{j,\,rf})$
33 end
34 $\widetilde{\boldsymbol{y}}_{base} \leftarrow$ back-transform $\widetilde{\boldsymbol{u}}_{(d+1),\,base}$ to x-scale by thresholding
35 $\widetilde{\boldsymbol{y}}_{rf} \leftarrow$ back-transform $\widetilde{\boldsymbol{u}}_{(d+1),\,rf}$ to x-scale by thresholding
36 $\left(\widetilde{x}_{base}\ \widetilde{y}_{base}\right) \leftarrow \begin{pmatrix} | & | & & | & | \\ \widetilde{\boldsymbol{x}}_{1,\,base} & \widetilde{\boldsymbol{x}}_{2,\,base} & \cdots & \widetilde{\boldsymbol{x}}_{d,\,base} & \widetilde{\boldsymbol{y}}_{base} \\ | & | & & | & | \end{pmatrix}$
37 $\left(\widetilde{x}_{rf}\ \widetilde{y}_{rf}\right) \leftarrow \begin{pmatrix} | & | & & | & | \\ \widetilde{\boldsymbol{x}}_{1,\,rf} & \widetilde{\boldsymbol{x}}_{2,\,rf} & \cdots & \widetilde{\boldsymbol{x}}_{d,\,rf} & \widetilde{\boldsymbol{y}}_{rf} \\ | & | & & | & | \end{pmatrix}$
38 return $\left(\widetilde{x}_{base}\ \widetilde{y}_{base}\right)$, $\left(\widetilde{x}_{rf}\ \widetilde{y}_{rf}\right)$
```

Algorithm 5: Algorithm for generating $\left(x_{true}\ y_{true}\right)^{(h,r)}$, $r \in [p_{true}]$ from $\left(\widetilde{x}_{base}\ \widetilde{\boldsymbol{y}}_{base}\right)^{(h)}$ and $\left(\widetilde{x}_{rf}\ \widetilde{\boldsymbol{y}}_{rf}\right)^{(h)}$ with $h \in [p_{base}]$ fixed.

```
   Input: for $h \in [p_{base}]$ fixed: $\left(\widetilde{x}_{base}\ \widetilde{\boldsymbol{y}}_{base}\right)^{(h)} \in \mathbb{R}^{n\times(d+1)}$, $\left(\widetilde{x}_{rf}\ \widetilde{\boldsymbol{y}}_{rf}\right)^{(h)} \in \mathbb{R}^{n\times(d+1)}$,
          $d = 100$
   Output: $\left(x_{true}\ y_{true}\right)^{(h,r)} \in \mathbb{R}^{n\times(d+1)}$, $r \in [p_{true}]$
 1 $n_{tree}^{*(h)} \leftarrow$ OOB estimate of $n_{tree}$ on $\left(\widetilde{x}_{rf}\ \widetilde{\boldsymbol{y}}_{rf}\right)^{(h)}$
 2 $\hat{\boldsymbol{\theta}}_{RF}^{(h,1)} \leftarrow \hat{\boldsymbol{\theta}}_{RF}^{*(h)} := (n_{tree}^{*(h)}, \lfloor\sqrt{d}\rfloor, 1) = (n_{tree}^{*(h)}, 10, 1)$
 3 $\hat{f}_{RF}^{(h,1)}(\cdot) \leftarrow f_{RF}\left(\left(\widetilde{x}_{rf}\ \widetilde{\boldsymbol{y}}_{rf}\right)^{(h)}, \hat{\boldsymbol{\theta}}_{RF}^{(h,1)}\right)$
 4 $\boldsymbol{y}_{true}^{(h,1)} \leftarrow \hat{f}_{RF}^{(h,1)}\left(\widetilde{x}_{base}^{(h)}\right)$
 5 $r \leftarrow 2$
 6 for $n_{tree} \in \{650, 1300\}$ do
 7     $n_{tree}^{(h,r)} \leftarrow n_{tree}$
 8     $\hat{\boldsymbol{\theta}}_{RF}^{(h,r)} \leftarrow (n_{tree}^{(h,r)}, \lfloor\sqrt{d}\rfloor, 1)$
 9     $\hat{f}_{RF}^{(h,r)}(\cdot) \leftarrow f_{RF}\left(\left(\widetilde{x}_{rf}\ \widetilde{\boldsymbol{y}}_{rf}\right)^{(h)}, \hat{\boldsymbol{\theta}}_{RF}^{(h,r)}\right)$
10     $\boldsymbol{y}_{true}^{(h,r)} \leftarrow \hat{f}_{RF}^{(h,r)}\left(\widetilde{x}_{base}^{(h)}\right)$
11     $r \leftarrow r + 1$
12 end
13 for $m_{try} \in \{3, 6\}$ do
14     $m_{try}^{(h,r)} \leftarrow m_{try}$
15     $\hat{\boldsymbol{\theta}}_{RF}^{(h,r)} \leftarrow (n_{tree}^{*(h)}, m_{try}^{(h,r)}, 1)$
16     $\hat{f}_{RF}^{(h,r)}(\cdot) \leftarrow f_{RF}\left(\left(\widetilde{x}_{rf}\ \widetilde{\boldsymbol{y}}_{rf}\right)^{(h)}, \hat{\boldsymbol{\theta}}_{RF}^{(h,r)}\right)$
17     $\boldsymbol{y}_{true}^{(h,r)} \leftarrow \hat{f}_{RF}^{(h,r)}\left(\widetilde{x}_{base}^{(h)}\right)$
18     $r \leftarrow r + 1$
19 end
20 for $s_{nodesize} \in \{50, 100\}$ do
21     $s_{nodesize}^{(h,r)} \leftarrow s_{nodesize}$
22     $\hat{\boldsymbol{\theta}}_{RF}^{(h,r)} \leftarrow (n_{tree}^{*(h)}, \lfloor\sqrt{d}\rfloor, s_{nodesize}^{(h,r)})$
23     $\hat{f}_{RF}^{(h,r)}(\cdot) \leftarrow f_{RF}\left(\left(\widetilde{x}_{rf}\ \widetilde{\boldsymbol{y}}_{rf}\right)^{(h)}, \hat{\boldsymbol{\theta}}_{RF}^{(h,r)}\right)$
24     $\boldsymbol{y}_{true}^{(h,r)} \leftarrow \hat{f}_{RF}^{(h,r)}\left(\widetilde{x}_{base}^{(h)}\right)$
25 end
26 for $r \in [p_{true}]$ do
27     $\left(x_{true}\ y_{true}\right)^{(h,r)} \leftarrow \left(\widetilde{x}_{base}^{(h)}\ y_{true}^{(h,r)}\right)$
28 end
29 return $\left(x_{true}\ y_{true}\right)^{(h,r)} \in \mathbb{R}^{n\times(d+1)}$, $r \in [p_{true}]$
```

5.2 Results – Simulation Study: Parametrically Simulated True Data

We generate synthetic $\big(\boldsymbol{x}_{synth}\ \boldsymbol{y}_{synth}\big)^{(h,\,r,\,l)}$, $l \in [p_{synth}]$ from true $\big(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\big)^{(h,\,r)}$, $r \in [p_{true}]$ as explained in Subsection 4.1. For executing the steps schematized in Algorithm 3 the R-package `rvinecopulib`, see Nagler and Vatter (2021) is used:
To obtain pseudo-observations $\hat{\mathcal{U}}_{true}$ of the true data on the $(d+1)$-dimensional unit cube the function `pseudo_obs` for observations $\boldsymbol{x}_{true}$ or the manually computed empirical marginal distribution $\hat{F}_{Y_{true}}$ for $\boldsymbol{y}_{true}$ is applied:

$$y \in \mathbb{R} \ : \quad \hat{F}_{Y_{true}}(y) := \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{\{x\,\leq\,y\}}(y_i)\ . \tag{5.7}$$

Then the vine copula estimation methods of Table 5 are fit to $\hat{\mathcal{U}}_{true}$ with the function `vinecop`. As $Y_{i,\,true} \in \{1,2,3,4,5\}$ is an ordinal random variable, additional observations on the copula scale $u_{Y_i,\,true} := \hat{F}_{Y_{true}}(y_i - 1)$, $i \in [n]$ are needed to fit a vine model with `vinecop` as explained in Subsection **Regular Vines with Discrete Components** on page 32. With the function `rvinecop` synthetic data $\mathcal{U}^{(l)}_{synth}$ on the copula-scale are sampled. They are transformed back to $\big(\boldsymbol{x}^{(l)}_{synth}\ \boldsymbol{y}^{(l)}_{synth}\big)$ to the original scale by applying the sample quantiles with the function `quantile` from the R-package `stats`, see R Core Team (2021). We use the quantile type 8 (2.4.1) to obtain observations $\boldsymbol{x}^{(l)}_{synth}$ and quantile type 1 (2.4.1) to obtain observations $\boldsymbol{y}^{(l)}_{synth}$.

After that we assess the quality of the synthetic data generated with the **non-parametric**, **parametric** and **mixed** vine estimation methods as shown in Subsection 4.2. We compute the performance measures and plot them: the positive relative mean error (ME^{+}_{rel}) defined in (4.5) on page 90 and the negative relative mean error (ME^{-}_{rel}) defined in (4.6) on page 93, the direct mAUC given in (4.7) on page 96 and the indirect mAUC given in (4.8) on page 99 versus the one parameter of n_{tree}, m_{try} and $s_{nodesize}$, that is currently varied.
Additionally to ME^{+}_{rel} and ME^{-}_{rel} we compute how often $\hat{y}^{(h,\,r,\,l),\,test}_{i,\,synth}$ overestimates and underestimates respectively $\hat{y}^{test}_{i,\,true}{}^{(h,\,r)}$ per low, medium and high perturbation while averaging over p_{base} and p_{synth}:

$$\begin{aligned} &\text{average no. of overestimations}: \quad \frac{1}{p_{base}\cdot p_{synth}} \sum_{\substack{h\in[p_{base}],\\ l\in[p_{synth}]}} B^{+\,(h,\,r)}_{l}\ , \\ &\text{average no. of underestimations}: \quad \frac{1}{p_{base}\cdot p_{synth}} \sum_{\substack{h\in[p_{base}],\\ l\in[p_{synth}]}} B^{-\,(h,\,r)}_{l}\ . \end{aligned} \tag{5.8}$$

The definition of $B^{+\,(h,\,r)}_{l}$ and $B^{-\,(h,\,r)}_{l}$ can be found in Equation (4.3). The average number of over- and underestimations should be set in relation to the size of the test data set $\big(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\big)^{(h,\,r)}_{test}$, $h \in [p_{base}]$, $r \in [p_{true}]$, which consists of 150 observations.

We also compute and plot the variable selection performance measures, i.e. the average distance of the logistic β-coefficients, see (4.9) on page 102, and the average number of discordant β-pairs, see (4.10) on page 105 derived from $\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right)^{(r)}$ and $\left(\boldsymbol{x}_{synth}\ \boldsymbol{y}_{synth}\right)^{(r,l)}$, $r \in [p_{true}]$, $l \in [p_{synth}]$ versus the one parameter of n_{tree}, m_{try} and $s_{nodesize}$, that is currently varied. One point in each graph corresponds to one measure computed.

Note, that in the R-package `glmnet` of Simon et al. (2011) the covariates can be standardized prior to fitting the multinomial logistic regression (with Lasso) with the function `glmnet` by setting the argument `standardize = TRUE`, which is the default. The β-coefficients returned are however on the original scale. In order to compute the average distance of the *standardized* logistic β-coefficients of (4.9), we need to standardize $\hat{\beta}_{jk,\,true}$ and $\hat{\beta}^{(l)}_{jk,\,synth}$, $j \in [d]$, $k \in [K]$ by hand as illustrated in (3.9).

For the following subsections recall the nomenclature of Table 15.

	perturbation:		
	low	**medium**	**high**
n_{tree}	$n^{*(h)}_{tree}$	1300	650
m_{try}	10	6	3
$s_{nodesize}$	1	50	100

Table 15: Nomenclature regarding perturbation introduced on the simulated true data set depending on the value of the parameter currently varied. Recall, that $\hat{n}^{*(h)}_{tree}$ is the OOB estimate of the optimal number of trees in the random forest, see 3.1.3.

Also recall the number of true and synthetic date sets respectively used in the simulation study given in Table 16.

perturbation	**no. of true data sets**	**no. of synthetic data sets**
low	10 $(= p_{base} \cdot 1)$	150 $(= p_{base} \cdot 1 \cdot p_{synth})$
medium	30 $(= p_{base} \cdot 3)$	450 $(= p_{base} \cdot 3 \cdot p_{synth})$
high	30 $(= p_{base} \cdot 3)$	450 $(= p_{base} \cdot 3 \cdot p_{synth})$

Table 16: Total number of true and corresponding synthetic data sets per **low**, **medium** and **high** perturbation respectively. Recall from Table 7, that $\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right)^{(h,r)}$ and $\left(\boldsymbol{x}_{synth}\ \boldsymbol{y}_{synth}\right)^{(h,r,l)}$ with $r \in [p_{true}]$, $h \in [p_{base}]$, $l \in [p_{synth}]$ and $p_{base} = 10$, $p_{true} = 7$, $p_{synth} = 15$.

Positive Relative Mean Error ME^{+}_{rel}

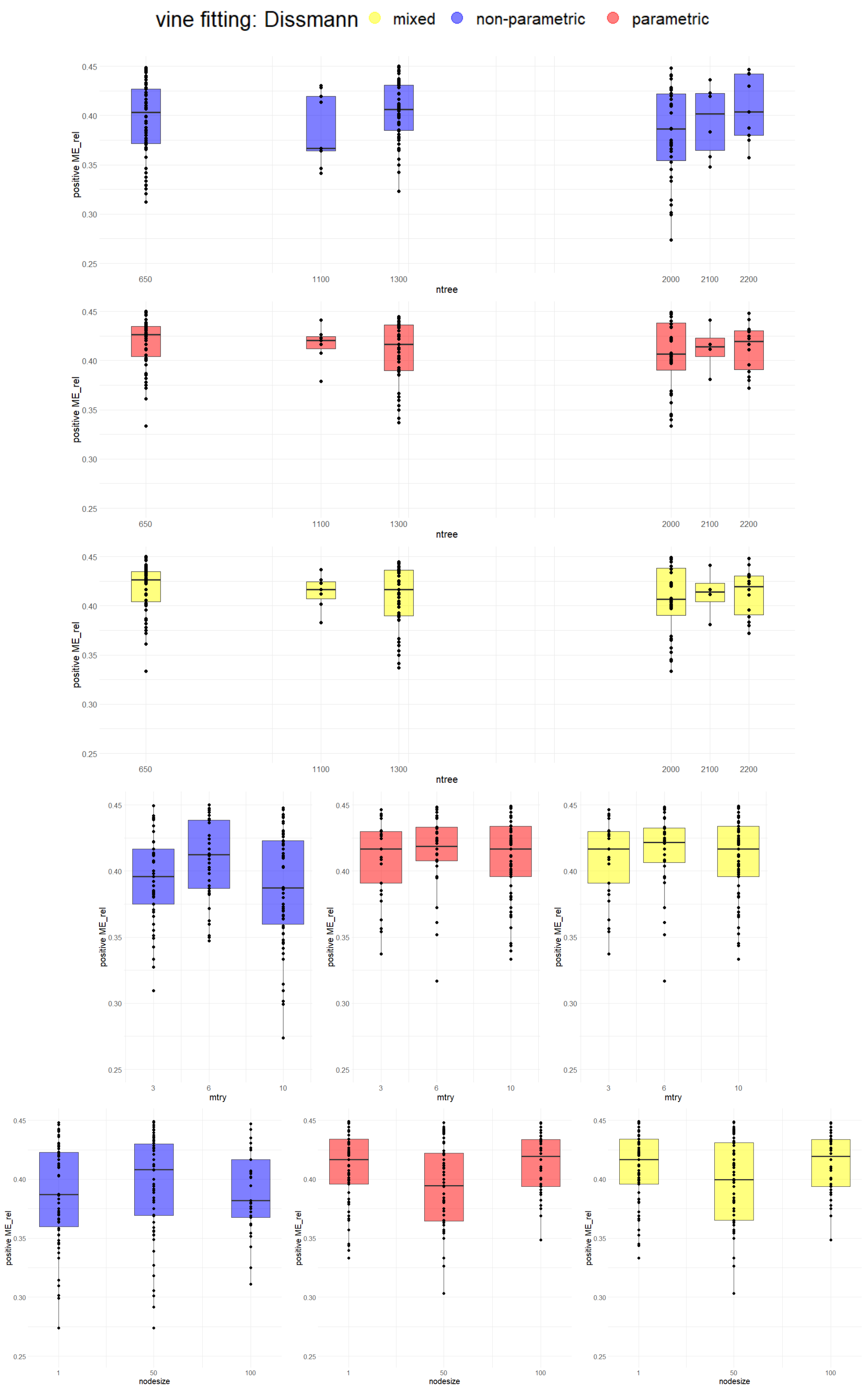

Figure 14: ME^{+}_{rel} obtained from synthetic data of the vine estimation methods **non-parametric**, **parametric** and **mixd** versus the perturbing random forest parameters n_{tree}, m_{try} and $s_{nodesize}$. Per vine fitting and colour the classification performance on **600 synthetic data sets** is displayed with $n = 500$ observations per data set, $d = 100$ features, $p_{base} = 10$, $p_{true} = 7$, $p_{synth} = 15$. The positive relative mean error is given in (4.5), the three vine estimation methods in Table (5).

LOW PERTURBATION:						
measure	**vine estimation method**	**perturbing parameter**	**1^{st} quartile**	**median**	**3^{rd} quartile**	**average no. of overestimations**
$\boldsymbol{ME^{+}_{rel}}$	non-parametric	n_{tree}	0.4066	0.4583	0.5000	18.6741
		m_{try}				
		$s_{nodesize}$				
	parametric	n_{tree}	0.4222	0.4635	0.5108	16.7852
		m_{try}				
		$s_{nodesize}$				
	mixed	n_{tree}	0.4222	0.4635	0.5115	16.8074
		m_{try}				
		$s_{nodesize}$				

MEDIUM PERTURBATION:						
measure	**vine estimation method**	**perturbing parameter**	**1^{st} quartile**	**median**	**3^{rd} quartile**	**average no. of overestimations**
$\boldsymbol{ME^{+}_{rel}}$	non-parametric	n_{tree}	0.4167	0.4722	0.5321	19.5533
		m_{try}	0.4395	0.4889	0.5386	17.48
		$s_{nodesize}$	0.4111	0.4621	0.5251	16.8
	parametric	n_{tree}	0.4363	0.4815	0.5321	17.8467
		m_{try}	0.4497	0.4944	0.5564	16.14
		$s_{nodesize}$	0.4094	0.4583	0.5116	15.1533
	mixed	n_{tree}	0.4363	0.4815	0.5324	17.8267
		m_{try}	0.4476	0.4944	0.5564	16.14
		$s_{nodesize}$	0.4106	0.4833	0.5116	15.1333

HIGH PERTURBATION:						
measure	**vine estimation method**	**perturbing parameter**	**1^{st} quartile**	**median**	**3^{rd} quartile**	**average no. of overestimations**
$\boldsymbol{ME^{+}_{rel}}$	non-parametric	n_{tree}	0.4125	0.4573	0.5070	18.3467
		m_{try}	0.4152	0.4833	0.5532	18.3867
		$s_{nodesize}$	0.4521	0.5000	0.5613	17.3867
	parametric	n_{tree}	0.4314	0.4766	0.5117	17.14
		m_{try}	0.4625	0.5088	0.5762	17.4267
		$s_{nodesize}$	0.4495	0.5201	0.5882	16.2933
	mixed	n_{tree}	0.4314	0.4766	0.5128	17.1533
		m_{try}	0.4625	0.5088	0.5762	17.4067
		$s_{nodesize}$	0.4495	0.5201	0.5879	16.3067

Table 17: Comparing $\boldsymbol{ME^{+}_{rel}}$ scores of the **non-parametric**, **parametric** and **mixed** vine estimation method for **low**, **medium** and **high perturbation**. Recall the number of true and synthetic data sets per degree of perturbation from Tables 13 and 16. The average number of overestimations, defined in Equation (5.8), should be set in relation to the size of the test data set $\left(x_{true}\ \boldsymbol{y}_{true}\right)^{(h,\,r)}_{test}$, $h \in [p_{base}]$, $r \in [p_{true}]$, which consists of 150 observations.

From Figure 14 and Table 17 we observe:

- Overall $\hat{y}_{i,\,synth}^{(l),\,test}$ **overestimate** $\hat{y}_{i,\,true}^{test}$ **by 46% up to 52.0%** at median, more detailed:
  - $\approx$ 46% to 50% at median for the **non-parametric** vine fitting and
  - $\approx$ 46% to 52% at median for the **parametric and mixed** vine fitting.
- This occurs overall on average in 16 to 20 out of 150 observations of the test set. This equals **10% to 13% of the cases**:
  - $\approx$ 17 to 20 (11% to 13%) at mean for the **non-parametric** vine fitting and
  - $\approx$ 15 to 18 (10% to 12%) at mean for the **parametric and mixed** vine fitting.
- Overall we observe an increase in ME_{rel}^{+} as well as the number of overestimations for increasing perturbation in the true data.
- For the non-parametric vine fitting this trend is not so clearly visible.
- The **parametric** and **mixed** vine fitting yield **very similar, almost identical** results with regard to ME_{rel}^{+}.
- The **non-parametric** vine fitting performs **slightly better** in terms of ME_{rel}^{+} than the parametric and mixed ones:

  it yields a ME_{rel}^{+}, which is at median 1% lower than the median ME_{rel}^{+} of the parametric and mixed vine fitting for low and medium perturbations. For high perturbations the ME_{rel}^{+} is at median 2% lower compared to the median ME_{rel}^{+} of the parametric and mixed vine fitting.
- However, the **number** of predictions $\hat{y}_{i,\,synth}^{(l),\,test}$, that overestimate $\hat{y}_{i,\,true}^{test}$, are **slightly higher for the non-parametric** vine fitting: it exceeds the parametric and mixed vine fitting by approximately 1 on average.

Negative Relative Mean Error ME_{rel}^{-}

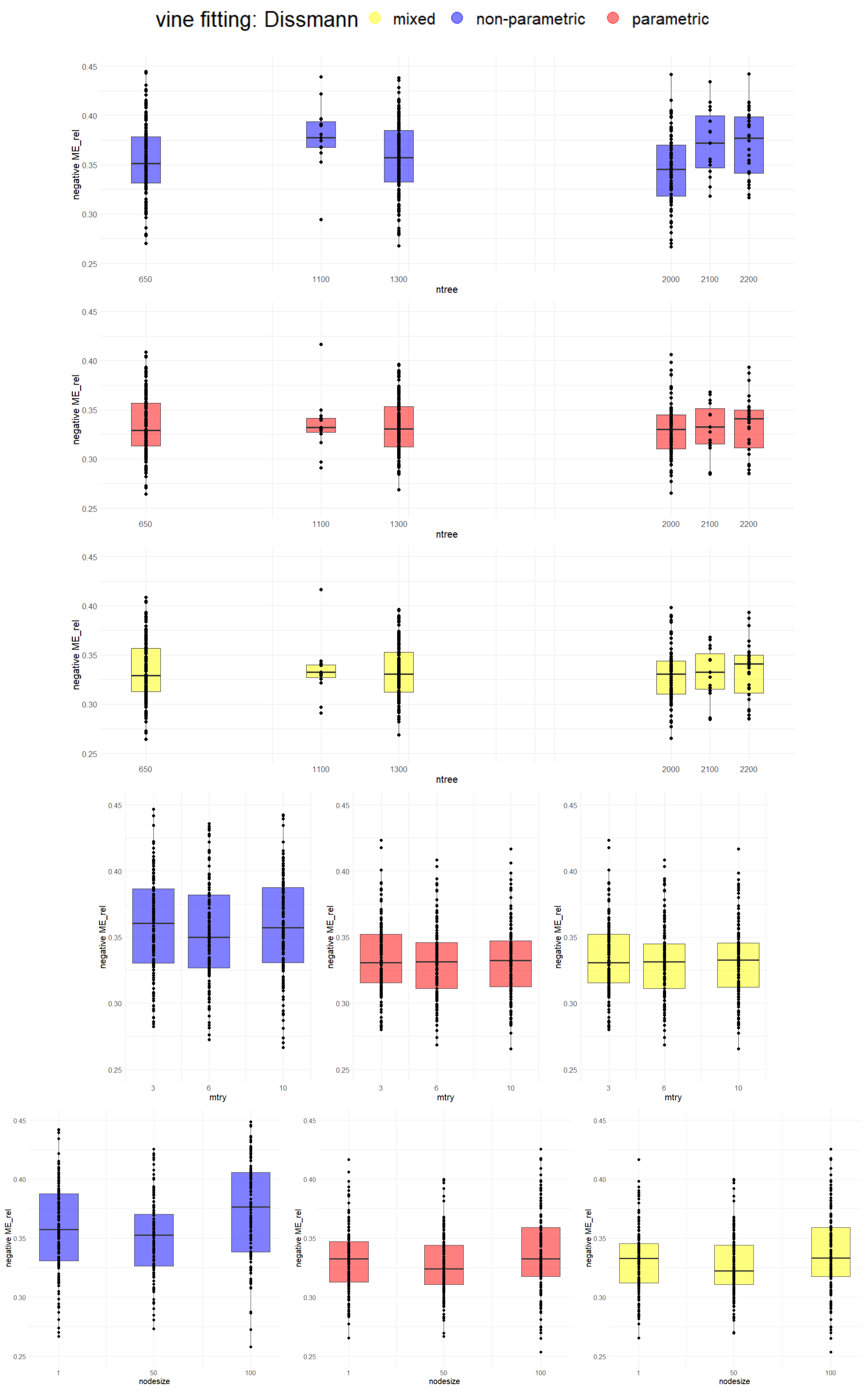

Figure 15: ME_{rel}^{-} obtained from synthetic data of the vine estimation methods **non-parametric**, **parametric** and **mixd** versus the perturbing random forest parameters n_{tree}, m_{try} and $s_{nodesize}$. Per vine fitting and colour the classification performance on **600 synthetic data sets** is displayed with $n = 500$ observations per data set, $d = 100$ features, $p_{base} = 10$, $p_{true} = 7$, $p_{synth} = 15$. The negative relative mean error is given in (4.6), the three vine estimation methods in Table (5).

LOW PERTURBATION:

measure	vine estimation method	perturbing parameter	1st quartile	median	3rd quartile	average no. of underestimations
$\boldsymbol{ME_{rel}^{-}}$	non-parametric	n_{tree}	0.3307	0.3573	0.3876	21.9556
		m_{try}				
		$s_{nodesize}$				
	parametric	n_{tree}	0.3127	0.3324	0.3473	18.4296
		m_{try}				
		$s_{nodesize}$				
	mixed	n_{tree}	0.3122	0.3327	0.3455	18.4296
		m_{try}				
		$s_{nodesize}$				

MEDIUM PERTURBATION:

measure	vine estimation method	perturbing parameter	1st quartile	median	3rd quartile	average no. of underestimations
$\boldsymbol{ME_{rel}^{-}}$	non-parametric	n_{tree}	0.3325	0.3572	0.3850	23.56
		m_{try}	0.3266	0.3500	0.3821	22.9467
		$s_{nodesize}$	0.3263	0.3525	0.3705	22.08
	parametric	n_{tree}	0.3123	0.3306	0.3532	19.66
		m_{try}	0.3111	0.3314	0.3461	18.6333
		$s_{nodesize}$	0.3107	0.3239	0.3441	18.7867
	mixed	n_{tree}	0.3122	0.3306	0.3528	19.6467
		m_{try}	0.3111	0.3314	0.3448	18.64
		$s_{nodesize}$	0.3107	0.3222	0.3441	18.7733

HIGH PERTURBATION:

measure	vine estimation method	perturbing parameter	1st quartile	median	3rd quartile	average no. of underestimations
$\boldsymbol{ME_{rel}^{-}}$	non-parametric	n_{tree}	0.3314	0.3515	0.3784	22.8133
		m_{try}	0.3314	0.3606	0.3891	21.3467
		$s_{nodesize}$	0.3420	0.3806	0.4170	20.4933
	parametric	n_{tree}	0.3131	0.3290	0.3566	18.5533
		m_{try}	0.3156	0.3306	0.3522	18.5
		$s_{nodesize}$	0.3170	0.3325	0.3597	17.66
	mixed	n_{tree}	0.3129	0.3290	0.3566	18.56
		m_{try}	0.3156	0.3306	0.3523	18.5
		$s_{nodesize}$	0.3170	0.3333	0.3597	17.6467

Table 18: Comparing $\boldsymbol{ME_{rel}^{-}}$ scores of the **non-parametric**, **parametric** and **mixed** vine estimation method for **low**, **medium** and **high perturbation**. Recall the number of true and synthetic data sets per degree of perturbation from Tables 13 and 16. The average number of underestimations, defined in Equation (5.8), should be set in relation to the size of the test data set $\left(x_{true}\ \boldsymbol{y}_{true}\right)_{test}^{(h,\,r)}$, $h \in [p_{base}]$, $r \in [p_{true}]$, which consists of 150 observations.

From Figure 15 and Table 18 we observe:

- Overall $\hat{y}_{i,\,synth}^{(l),\,test}$ **underestimate** $\hat{y}_{i,\,true}^{test}$ at median **by 33% up to 38%**, more detailed:
  - $\approx$ 35% to 38% at median for the **non-parametric** vine fitting and
  - $\approx$ 33% at median for the **parametric and mixed** vine fitting.
- Overall, this occurs on average in 18 to 24 out of 150 observations of the test set. This equals **12% to 16% of the cases**:
  - $\approx$ 20 to 24 (13% to 16%) at mean for the **non-parametric** vine fitting and
  - $\approx$ 18 to 19 (12%) at mean for the **parametric and mixed** vine fitting.
- Overall, we observe a quite constant ME_{rel}^{-} as well as the number of underestimations for increasing perturbation in the true data.
- For the non-parametric vine fitting ME_{rel}^{-} is increased for high perturbation, while the number of underestimations is increased for medium perturbation.
- As for ME_{rel}^{+}, the **parametric** and **mixed** vine fitting yield **very similar, almost identical** results with regard to ME_{rel}^{-}.
- The **parametric and mixed** vine fitting perform **slightly better** in terms of ME_{rel}^{-} than the non-parametric one:

  they yield a ME_{rel}^{-}, which is at median 2% to 3% lower compared to the median ME_{rel}^{-} of the non-parametric vine fitting.
- Also the **number** of predictions $\hat{y}_{i,\,synth}^{(l),\,test}$, that underestimate $\hat{y}_{i,\,true}^{test}$, are **slightly higher for the non-parametric** vine fitting: it exceeds the parametric and mixed vine fitting by approximately 3 on average.

Direct $mAUC_{dir}$

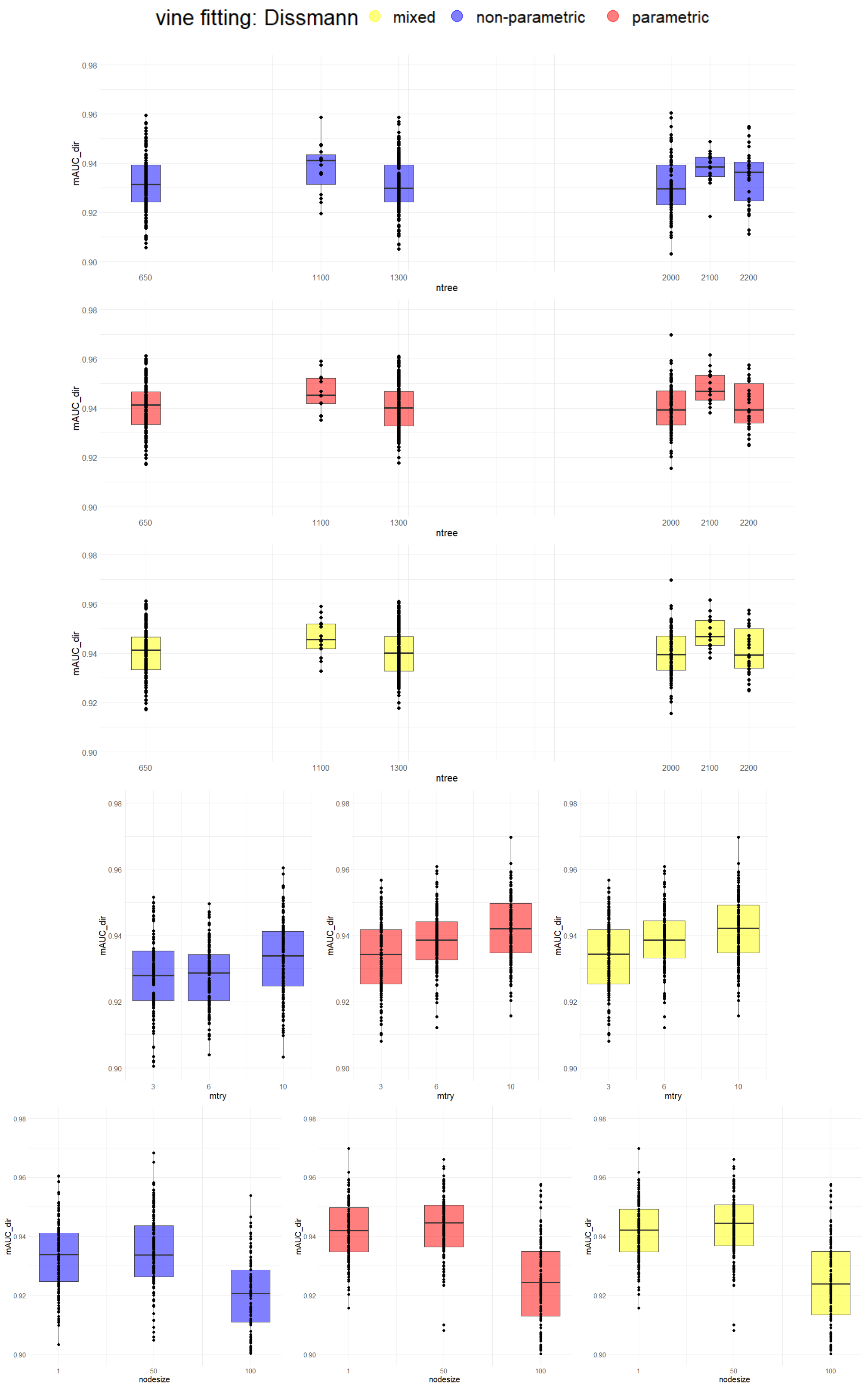

Figure 16: $\boldsymbol{mAUC_{dir}}$ obtained from synthetic data of the vine estimation methods **non-parametric**, **parametric** and **mixd** versus the perturbing random forest parameters $\boldsymbol{n_{tree}}$, $\boldsymbol{m_{try}}$ and $\boldsymbol{s_{nodesize}}$. Per vine fitting and colour the classification performance on **600 synthetic data sets** is displayed with $\boldsymbol{n = 500}$ observations per data set, $\boldsymbol{d = 100}$ features, $\boldsymbol{p_{base} = 10}$, $\boldsymbol{p_{true} = 7}$, $\boldsymbol{p_{synth} = 15}$. The direct mAUC is given in (4.7), the three vine estimation methods in Table (5).

LOW PERTURBATION:					
measure	**vine estimation method**	**perturbing parameter**	**1st quartile**	**median**	**3rd quartile**
$\boldsymbol{mAUC_{dir}}$	non-parametric	n_{tree}	0.9247	0.9338	0.9412
		m_{try}			
		$s_{nodesize}$			
	parametric	n_{tree}	0.9347	0.9420	0.9498
		m_{try}			
		$s_{nodesize}$			
	mixed	n_{tree}	0.9347	0.9421	0.9492
		m_{try}			
		$s_{nodesize}$			

MEDIUM PERTURBATION:					
measure	**vine estimation method**	**perturbing parameter**	**1st quartile**	**median**	**3rd quartile**
$\boldsymbol{mAUC_{dir}}$	non-parametric	n_{tree}	0.9243	0.9298	0.9393
		m_{try}	0.9203	0.9287	0.9342
		$s_{nodesize}$	0.9262	0.9336	0.9435
	parametric	n_{tree}	0.9328	0.9401	0.9468
		m_{try}	0.9327	0.9387	0.9441
		$s_{nodesize}$	0.9364	0.9446	0.9506
	mixed	n_{tree}	0.9328	0.9401	0.9470
		m_{try}	0.9332	0.9387	0.9444
		$s_{nodesize}$	0.9368	0.9444	0.9507

HIGH PERTURBATION:					
measure	**vine estimation method**	**perturbing parameter**	**1st quartile**	**median**	**3rd quartile**
$\boldsymbol{mAUC_{dir}}$	non-parametric	n_{tree}	0.9242	0.9315	0.9393
		m_{try}	0.9194	0.9266	0.9345
		$s_{nodesize}$	0.8992	0.9142	0.9250
	parametric	n_{tree}	0.9335	0.9413	0.9467
		m_{try}	0.9254	0.9343	0.9418
		$s_{nodesize}$	0.9120	0.9232	0.9348
	mixed	n_{tree}	0.9335	0.9413	0.9467
		m_{try}	0.9254	0.9344	0.9418
		$s_{nodesize}$	0.9120	0.9232	0.9348

Table 19: Comparing $\boldsymbol{mAUC_{dir}}$ scores of the **non-parametric**, **parametric** and **mixed** vine estimation method for **low**, **medium** and **high perturbation**. Recall the number of true and synthetic data sets per degree of perturbation from Tables 13 and 16.

From Figure 16 and Table 19 we observe:

- Overall, the $mAUC_{\boldsymbol{dir}}$, which is directly comparing $\hat{y}_{i,\,synth}^{(l),\,test}$ with $\hat{y}_{i,\,true}^{test}$, takes on a value of **91% to 94%** at median. In more detail:
  - $\approx$ 91% to 93% at median for the **non-parametric** vine fitting and
  - $\approx$ 92% to 94% at median for the **parametric and mixed** vine fitting.
- Only for perturbation through the random forest parameter m_{try} we observe a slight decrease in $mAUC_{\boldsymbol{dir}}$ for increasing perturbation in the true data. Otherwise there is no clear trend visible for the $mAUC_{\boldsymbol{dir}}$ for increasing perturbation.
- The **parametric** and **mixed** vine fitting yield **very similar, almost identical** results with regard to $mAUC_{\boldsymbol{dir}}$.
- The **parametric and mixed** vine fitting perform **slightly better** in terms of $mAUC_{\boldsymbol{dir}}$ than the non-parametric one:

  they yield a $mAUC_{\boldsymbol{dir}}$, which is at median 1% higher than the median $mAUC_{\boldsymbol{dir}}$ of the non-parametric vine fitting.

Indirect $mAUC_{indir}$

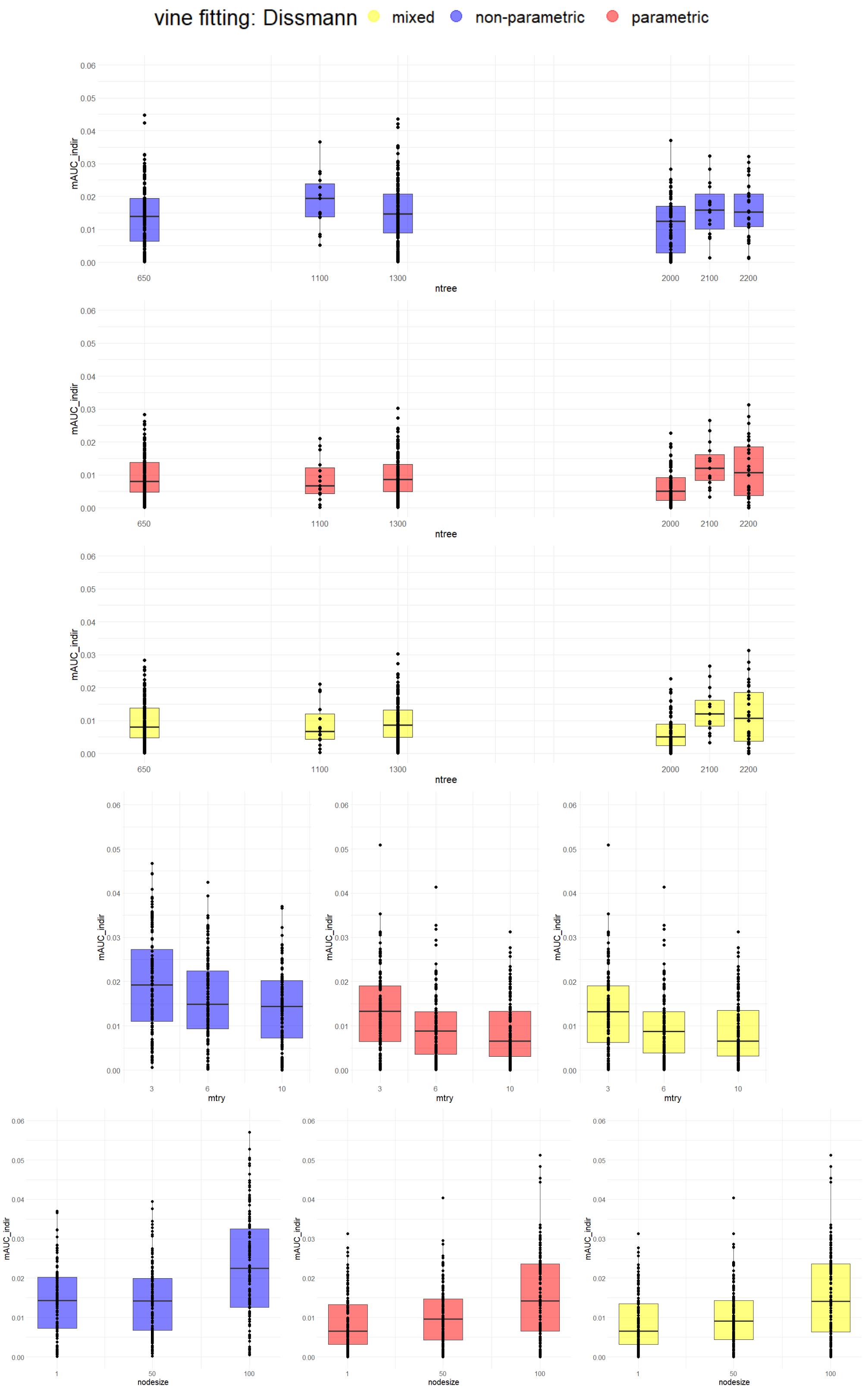

Figure 17: $\boldsymbol{mAUC_{indir}}$ obtained from synthetic data of the vine estimation methods **non-parametric**, **parametric** and **mixd** versus the perturbing random forest parameters $\boldsymbol{n_{tree}}$, $\boldsymbol{m_{try}}$ and $\boldsymbol{s_{nodesize}}$. Per vine fitting and colour the classification performance on **600 synthetic data sets** is displayed with $\boldsymbol{n = 500}$ observations per data set, $\boldsymbol{d = 100}$ features, $\boldsymbol{p_{base} = 10}$, $\boldsymbol{p_{true} = 7}$, $\boldsymbol{p_{synth} = 15}$. The direct mAUC is given in (4.8), the three vine estimation methods in Table (5).

LOW PERTURBATION:					
measure	**vine estimation method**	**perturbing parameter**	**1st quartile**	**median**	**3rd quartile**
$\boldsymbol{mAUC_{indir}}$	non-parametric	n_{tree}	0.0072	0.0144	0.0203
		m_{try}			
		$s_{nodesize}$			
	parametric	n_{tree}	0.0031	0.0065	0.0133
		m_{try}			
		$s_{nodesize}$			
	mixed	n_{tree}	0.0032	0.0065	0.0135
		m_{try}			
		$s_{nodesize}$			

MEDIUM PERTURBATION:					
measure	**vine estimation method**	**perturbing parameter**	**1st quartile**	**median**	**3rd quartile**
$\boldsymbol{mAUC_{indir}}$	non-parametric	n_{tree}	0.0089	0.0148	0.0208
		m_{try}	0.0093	0.0148	0.0224
		$s_{nodesize}$	0.0067	0.0142	0.0120
	parametric	n_{tree}	0.0049	0.0086	0.0132
		m_{try}	0.0036	0.0088	0.0132
		$s_{nodesize}$	0.0043	0.0096	0.0147
	mixed	n_{tree}	0.0049	0.0087	0.0132
		m_{try}	0.0039	0.0088	0.0132
		$s_{nodesize}$	0.0044	0.0091	0.0143

HIGH PERTURBATION:					
measure	**vine estimation method**	**perturbing parameter**	**1st quartile**	**median**	**3rd quartile**
$\boldsymbol{mAUC_{indir}}$	non-parametric	n_{tree}	0.0064	0.0139	0.0194
		m_{try}	0.0110	0.0192	0.0270
		$s_{nodesize}$	0.0126	0.0229	0.0330
	parametric	n_{tree}	0.0047	0.0080	0.0138
		m_{try}	0.0065	0.0133	0.0191
		$s_{nodesize}$	0.0066	0.0142	0.0236
	mixed	n_{tree}	0.0047	0.0080	0.0138
		m_{try}	0.0062	0.0132	0.0191
		$s_{nodesize}$	0.0063	0.0142	0.0236

Table 20: Comparing $\boldsymbol{mAUC_{indir}}$ scores of the **non-parametric**, **parametric** and **mixed** vine estimation method for **low**, **medium** and **high perturbation**. Recall the number of true and synthetic data sets per degree of perturbation from Tables 13 and 16.

From Figure 17 and Table 20 we observe:

- Overall, the $mAUC_{\boldsymbol{indir}}$, which is indirectly comparing $\hat{y}_{i,synth}^{(l),test}$ with $\hat{y}_{i,true}^{test}$, takes on a value of **0.7% to 2.3%** at median, more detailed:
  - $\approx$ 1.4% to 2.3% at median for the **non-parametric** vine fitting and
  - $\approx$ 0.7% to 1.4% at median for the **parametric and mixed** vine fitting.
- Overall, for increasing perturbation the values of $mAUC_{\boldsymbol{indir}}$ increase for all three vine fitting methods. This trend in $mAUC_{\boldsymbol{indir}}$ is not so clear for perturbation through the random forest parameter n_{tree}.
- The **parametric** and **mixed** vine fitting yield **very similar, almost identical** results with regard to $mAUC_{\boldsymbol{indir}}$.
- The **parametric and mixed** vine fitting perform **slightly better** in terms of $mAUC_{\boldsymbol{indir}}$ than the non-parametric one.

Average Distance of the Logistic Regression β-Coefficients $Dist(\hat{\beta}_{true}, \hat{\beta}_{synth})$

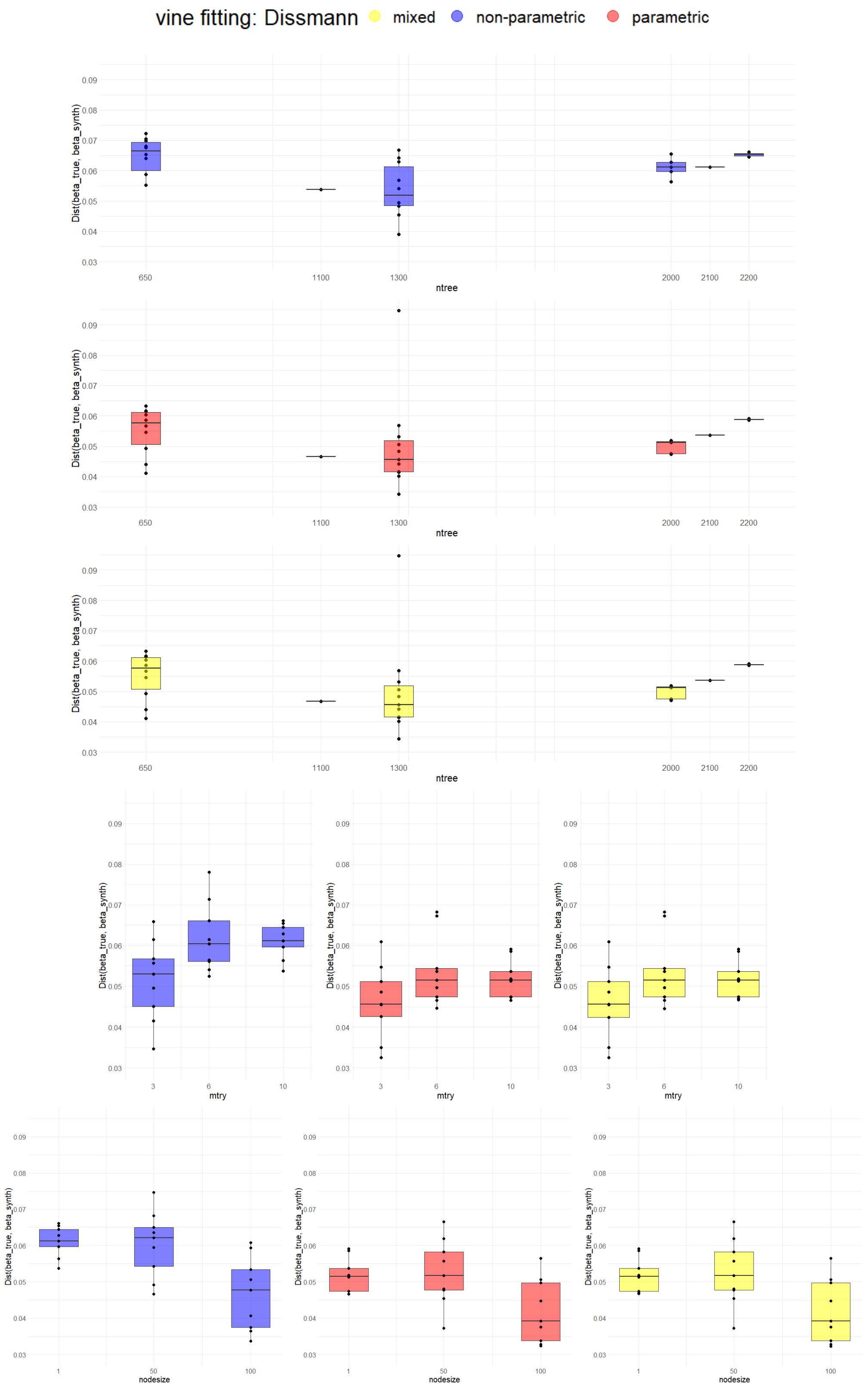

Figure 18: $Dist(\hat{\beta}_{true}, \hat{\beta}_{synth})$ obtained from synthetic data of the vine estimation methods **non-parametric**, **parametric** and **mixd** versus the perturbing random forest parameters n_{tree}, m_{try} and $s_{nodesize}$. Per vine fitting and colour the classification performance on **600 synthetic data sets** is displayed with $n = 500$ observations per data set, $d = 100$ features, $p_{base} = 10$, $p_{true} = 7$, $p_{synth} = 15$. The average distance of the logistic β-coefficients is given in (4.9), the three vine estimation methods in Table (5).

LOW PERTURBATION:					
measure	**vine estimation method**	**perturbing parameter**	**1st quartile**	**median**	**3rd quartile**
$\boldsymbol{Dist(\hat{\beta}_{true}, \hat{\beta}_{synth})}$	non-parametric	n_{tree}	0.0596	0.0612	0.0645
		m_{try}			
		$s_{nodesize}$			
	parametric	n_{tree}	0.0474	0.0515	0.0537
		m_{try}			
		$s_{nodesize}$			
	mixed	n_{tree}	0.0474	0.0515	0.0537
		m_{try}			
		$s_{nodesize}$			

MEDIUM PERTURBATION:					
measure	**vine estimation method**	**perturbing parameter**	**1st quartile**	**median**	**3rd quartile**
$\boldsymbol{Dist(\hat{\beta}_{true}, \hat{\beta}_{synth})}$	non-parametric	n_{tree}	0.0488	0.0541	0.0635
		m_{try}	0.0561	0.0605	0.0662
		$s_{nodesize}$	0.0543	0.0622	0.0650
	parametric	n_{tree}	0.0415	0.0456	0.0518
		m_{try}	0.0475	0.0516	0.0544
		$s_{nodesize}$	0.0477	0.0517	0.0582
	mixed	n_{tree}	0.0415	0.0456	0.0518
		m_{try}	0.0475	0.0516	0.0544
		$s_{nodesize}$	0.0477	0.0517	0.0582

HIGH PERTURBATION:					
measure	**vine estimation method**	**perturbing parameter**	**1st quartile**	**median**	**3rd quartile**
$\boldsymbol{Dist(\hat{\beta}_{true}, \hat{\beta}_{synth})}$	non-parametric	n_{tree}	0.0601	0.0665	0.0693
		m_{try}	0.0451	0.0530	0.0568
		$s_{nodesize}$	0.0375	0.0478	0.0534
	parametric	n_{tree}	0.0505	0.0577	0.0612
		m_{try}	0.0426	0.0456	0.0512
		$s_{nodesize}$	0.0337	0.0393	0.0497
	mixed	n_{tree}	0.0507	0.0577	0.0612
		m_{try}	0.0424	0.0456	0.0512
		$s_{nodesize}$	0.0337	0.0393	0.0497

Table 21: Comparing $\boldsymbol{Dist(\hat{\beta}_{true}, \hat{\beta}_{synth})}$ scores of the **non-parametric**, **parametric** and **mixed** vine estimation method for **low**, **medium** and **high perturbation**. Recall the number of true and synthetic data sets per degree of perturbation from Tables 13 and 16.

From Figure 18 and Table 21 we observe:

- Overall, the $Dist(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth})$ takes on values of **3.9% to 6.6%** at median, more detailed:
  - $\approx$ 4.7% to 6.6% at median for the **non-parametric** vine fitting and
  - $\approx$ 3.9% to 5.7% at median for the **parametric and mixed** vine fitting.
- Overall, the **parametric and mixed** vine fitting perform **slightly better** in terms of $Dist(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth})$ than the non-parametric one.
- However, there is **no clear trend** visible for the median of $Dist(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth})$ when perturbation through the random forest parameters is increasing.
- The **parametric** and **mixed** vine fitting yield overall **very similar, almost identical** results with regard to $mAUC_{\boldsymbol{indir}}$.

Average Number of Discordant β-Pairs $Discor(\hat{\beta}_{true}, \hat{\beta}_{synth})$

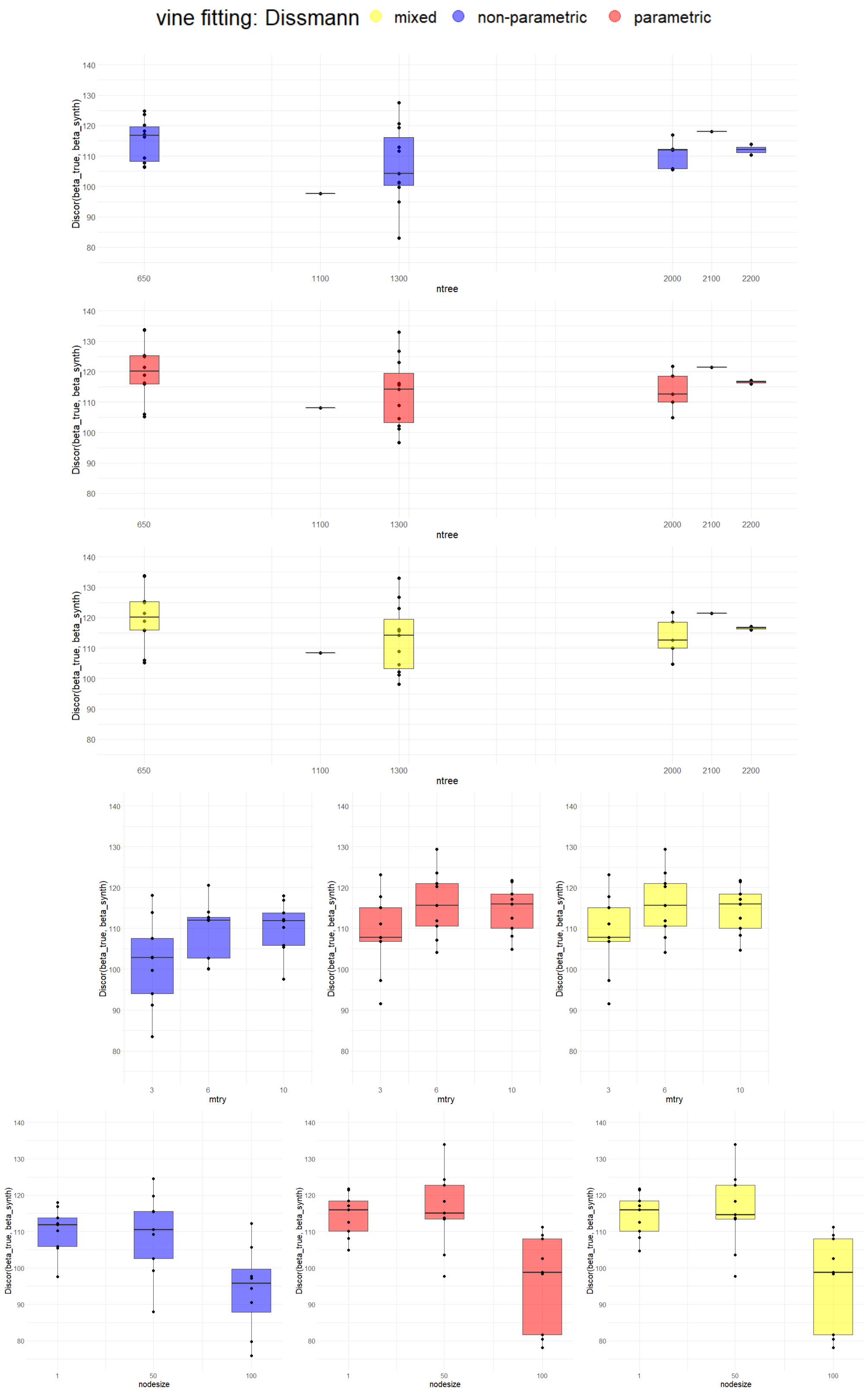

Figure 19: $Discor(\hat{\beta}_{true}, \hat{\beta}_{synth})$ obtained from synthetic data of the vine estimation methods **non-parametric**, **parametric** and **mixd** versus the perturbing random forest parameters n_{tree}, m_{try} and $s_{nodesize}$. Per vine fitting and colour the classification performance on **600 synthetic data sets** is displayed with $n = 500$ observations per data set, $d = 100$ features, $p_{base} = 10$, $p_{true} = 7$, $p_{synth} = 15$. The average number of discordant β-pairs is given in (4.10), the three vine estimation methods in Table (5).

LOW PERTURBATION:					
measure	**vine estimation method**	**perturbing parameter**	**1st quartile**	**median**	**3rd quartile**
$Discor(\hat{\beta}_{true}, \hat{\beta}_{synth})$	non-parametric	n_{tree}	105.867	111.867	113.800
		m_{try}			
		$s_{nodesize}$			
	parametric	n_{tree}	110.067	116.000	118.467
		m_{try}			
		$s_{nodesize}$			
	mixed	n_{tree}	110.067	116.000	118.467
		m_{try}			
		$s_{nodesize}$			

MEDIUM PERTURBATION:					
measure	**vine estimation method**	**perturbing parameter**	**1st quartile**	**median**	**3rd quartile**
$Discor(\hat{\beta}_{true}, \hat{\beta}_{synth})$	non-parametric	n_{tree}	100.433	104.267	116.167
		m_{try}	102.733	111.933	112.733
		$s_{nodesize}$	102.533	110.600	115.533
	parametric	n_{tree}	103.333	114.267	119.567
		m_{try}	110.600	115.667	121.000
		$s_{nodesize}$	113.467	115.133	122.800
	mixed	n_{tree}	103.333	114.267	119.567
		m_{try}	110.600	115.667	121.000
		$s_{nodesize}$	113.467	114.667	122.733

HIGH PERTURBATION:					
measure	**vine estimation method**	**perturbing parameter**	**1st quartile**	**median**	**3rd quartile**
$Discor(\hat{\beta}_{true}, \hat{\beta}_{synth})$	non-parametric	n_{tree}	108.183	116.700	119.717
		m_{try}	94.067	102.867	107.533
		$s_{nodesize}$	79.800	94.400	97.733
	parametric	n_{tree}	116.033	120.167	125.217
		m_{try}	106.800	107.800	115.133
		$s_{nodesize}$	81.667	98.867	108.067
	mixed	n_{tree}	115.883	120.167	125.217
		m_{try}	106.800	107.800	115.133
		$s_{nodesize}$	81.667	98.867	108.067

Table 22: Comparing $Discor(\hat{\beta}_{true}, \hat{\beta}_{synth})$ scores of the **non-parametric**, **parametric** and **mixed** vine estimation method for **low**, **medium** and **high perturbation**. Recall the number of true and synthetic data sets per degree of perturbation from Tables 13 and 16.

From Figure 19 and Table 22 we observe:

- Overall, the $Discor(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth})$ takes on values of **94 to 120** discordant β-pairs at median:
  - $\approx$ 94 to 116 at median for the **non-parametric** vine fitting and
  - $\approx$ 98 to 120 at median for the **parametric and mixed** vine fitting.
- This should be compared to the total number of regularized β-coefficients in the logistic regression model with Lasso, which is equal to $K \cdot d = 500$ due to the symmetric model formulation, see Equation **??**. For the:
  - **non-parametric** vine fitting we have $\approx$ 19% to 23% discordant β-pairs at median and
  - for the **parametric and mixed** vine fitting we have $\approx$ 20% to 24% discordant β-pairs at median.
- The **non-parametric** vine fitting therefore performs **slightly better** in terms of $Discor(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth})$ than the parametric and mixed ones.
- However, there is **no clear trend** in median of $Dist(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth})$ visible for increasing perturbation in the random forest parameters.
- The **parametric** and **mixed** vine fitting yield **very similar, almost identical** results with regard to $Dist(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth})$.
- For all three vine fitting methods it holds, that:
  - high perturbation in m_{try} and $s_{nodesize}$ decreases $Discor(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth})$
  - while on the contrary high perturbation in n_{tree} increases $Discor(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth})$.

Vine Estimation Method Comparison – Simulation Study

Summarized:

- For each measure and degree of perturbation the **parametric and the mixed vine estimation method** are very similar or almost identical. The reason for this might be, that in fitting the mixed vine most of the times a parametric pair copula is estimated instead of non-parametric one.
- All three vine estimation methods cause a classifier trained on the synthetic data to **overestimate** $\hat{y}_{i,\,true}^{test}$, $i \in [n]$ **by a higher percentage** than underestimating them. In detail:
  - overestimation by around 46% to 52%,
  - underestimation by around 33% to 38%.
- However, **underestimation** occurs slightly **more often** than overestimation, namely in detail:
  - overestimation in 12% to 16% of the 150 predictions,
  - underestimation in 10% to 13% of the cases.
- The estimates $\hat{y}_{i,\,synth}^{(l),\,test}$, $i \in [n]$ of the **non-parametric** vine estimation method **overestimate** $\hat{y}_{i,\,true}^{test}$, $i \in [n]$ to a **slightly lower** percentage than the ones obtained from the parametric or mixed vine estimation method.
- On the other hand $\hat{y}_{i,\,synth}^{(l),\,test}$, $i \in [n]$ of the **parametric and mixed** vine estimation method **underestimate** $\hat{y}_{i,\,true}^{test}$, $i \in [n]$ **by less** than the ones obtained from the non-parametric vine estimation.
- Thus, if for example in a clinical study underestimating a patient's cancer severity is a more severe error than overestimating it, the user should choose the **parametric or mixed vine estimation method**.

  If for other applications the contrary is the case, namely overestimation of the label is a more severe error than underestimation, the user should choose the **non-parametric** vine fitting.
- Measured with $mAUC_{\boldsymbol{dir}}$ and $mAUC_{\boldsymbol{indir}}$ the **parametric and mixed** vine estimation method **outperform** the non-parametric vine estimation method **slightly**. This might be due to the fact, that the true data is simulated parametrically!
- If the user wishes to gain a similar interpretation for a fixed feature's influence (i.e. by the magnitude of its logistic β-coefficient), $Dist(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth})$ indicates, that the **parametric and mixed vine estimation method** are to be **slightly preferred** to the non-parametric vine fitting.
- Similarly, the $Discor(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth})$ indicates, that the **parametric and mixed vine estimation method** are to be **slightly preferred** to the non-parametric vine fitting, if the logistic Lasso classifier trained on the synthetic data should recognize

similar features as important for the predictions as the logistic Lasso classifier trained on the true data.

- The parametric and mixed vine fitting yield almost identical results with respect to all measures. This lets us assume, that the mixed vine fitting chooses to fit most of the pair copulas parametrically.
- All in all the differences between the non-parametric, parametric and mixed vine estimation methods in terms of (variable selection) performance are small and all three methods are able to generate synthetic data, that lets a classifier learn a similar rule as from the true data.

Table 23 illustrates at a glance, which vine estimation method should be chosen to achieve good performance and interpretability respectively for which degree of perturbation.

	measure	**perturbation level**		
		low	**medium**	**high**
performance	ME^{+}_{rel}	non-parametric	non-parametric	non-parametric
		parametric	parametric	parametric
		mixed	mixed	mixed
	ME^{-}_{rel}	non-parametric	non-parametric	non-parametric
		parametric	parametric	parametric
		mixed	mixed	mixed
	$mAUC_{\boldsymbol{dir}}$	non-parametric	non-parametric	non-parametric
		parametric	parametric	parametric
		mixed	mixed	mixed
	$mAUC_{\boldsymbol{indir}}$	non-parametric	non-parametric	non-parametric
		parametric	parametric	parametric
		mixed	mixed	mixed
interpretability	$Dist(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth})$	non-parametric	non-parametric	non-parametric
		parametric	parametric	parametric
		mixed	mixed	mixed
	$Discor(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth})$	non-parametric	non-parametric	non-parametric
		parametric	parametric	parametric
		mixed	mixed	mixed

Table 23: Comparison of the **non-parametric**, **parametric** and **mixed** vine estimation method at a glance for the simulation study. Coloured cells indicate, which vine fitting should be preferred for a chosen measure and degree of perturbation.

6 Application I: MAGIC Gamma Telescope Data Set

Additionally to the parametrically simulated true data of Section 5 we try our methodology on the *MAGIC gamma telescope data set* of the UCI Machine Learning Repository Bock (2007). It was generated by a Monte Carlo program, Corsika, which is described in Heck et al. (1998), to simulate registration of high energy gamma particles in a ground-based atmospheric Cherenkov gamma telescope. We do this in order to obtain a complete picture on how well the three different vine estimation methods of Table 5 perform in generating synthetic data for training a classifier in our sense.
The MAGIC gamma telescope data set consists of:

- 19020 instances of
- 10 continuous covariates numbered as 1 to 10 and
- one dichotomous response variable `class` $\in \{g, h\}$:

  $$\begin{aligned} g = &\text{ gamma (signal) : 12332 times} \\ h = &\text{ hadron (background) : 6688 times .} \end{aligned}$$

  We should note, that "[...] classifying a background event as signal is worse than classifying a signal event as background" Bock (2007).
- The 10 covariates show a non-sparse correlation structure given in Figure 20. That is why we decide to use logistic regression **without** regularization for classification.

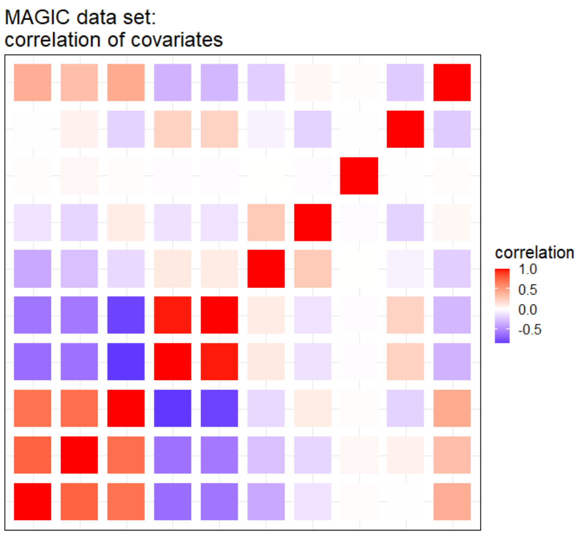

Figure 20: Coloured correlation matrix of the 10 covariates of the MAGIC data set displaying the magnitude of the Pearson correlation. As it is not sparse, we use a logistic regression **without** regularization for classification.

We use the methodology of Section 4 with the three different vine estimation methods of Table 5 on the MAGIC gamma telescope data set. For the analysis we code the labels the following way:

$$"2" := g \text{ (signal)} \ , \quad "1" := h \text{ (background)} \ . \tag{6.1}$$

6.1 Results – Application I: MAGIC Gamma Telescope Data Set

The methodology of Section 4 was used on the MAGIC gamma telescope data set. We generated:

- $p_{synth} = 15$ synthetic data sets $\left(\boldsymbol{x}_{synth}^{(l)} \ \boldsymbol{y}_{synth}^{(l)}\right)$, $l \in [15]$ from the MAGIC gamma telescope data,
- each containing the same number of instances as the original MAGIC gamma telescope data, namely 19020 instances.
- For a fixed $l \in [15]$ we compare:
  - $\lfloor$ 30% test share $\times 19020 \rfloor = 5706$ predictions of a (logistic regression) classifier trained on 70% of $\left(\boldsymbol{x}_{synth}^{(l)} \ \boldsymbol{y}_{synth}^{(l)}\right)$ to
  - $\lfloor$ 30% test share $\times 19020 \rfloor = 5706$ predictions of the same type of classifier trained on 70% of the MAGIC gamma telescope data.
- This yields $15 \cdot 5706 = 85590$ comparisons in total.

A Closer Look at the Mixed Vine Estimation Method

Before looking at the performance measures we are interested to see, how often a pair copula of the first tree level is estimated non-parametrically instead of parametrically in the mixed vine fitting due to a better AIC value.[21] In the first tree level this is the case for 8 out of 10 pair copulas. We have a look at their normalized contour plots[22] displayed in Figure 21. We do this in order to see, whether the non-parametric estimation is appropriate for the respective pair copula. We say, that this is the case, since the normalized contour plot in Figure 21 does not show a shape similar to one of the parametric pair copula families, which could have been chosen instead for estimation. Fitting the pair copula of a pair of variables non-parametrically when this is not appropriate could lead to overfitting. We want to avoid this as a vine consisting of overfitted pair copulas might not be well suited to generate synthetic data that let a classifier learn a similar rule as it would have learned on the true data.
The contour plots of Figure 21 suggest, that a non-parametric fit of the pair copula is appropriate for pairs $(8, 2)$, $(6, 7)$, $(7, 1)$, $(10, 1)$, $(1, 4)$, $(2, 4)$ and $(3, 4)$. We refit the pair copula of pair $(3, 9)$ parametrically and compare the log-likelihood of the parametric pair

[21]Recall the procedure of estimating R-vines illustrated in Section **Estimating Regular Vines** on page 26.
[22]Normalized contour plots were shortly introduced on page 11.

copula fit to its non-parametric counterpart. Using the function `BiCopEstList` of the `R`-package `VineCopula` by Nagler et al. (2021). The BB8 copula rotated by 270° yields the best fit in terms of log-likelihood with a log-likelihood equal to 911.13. The non-parametric fit clearly outperforms parametric fit with a log-likelihood equal to 1410.92.

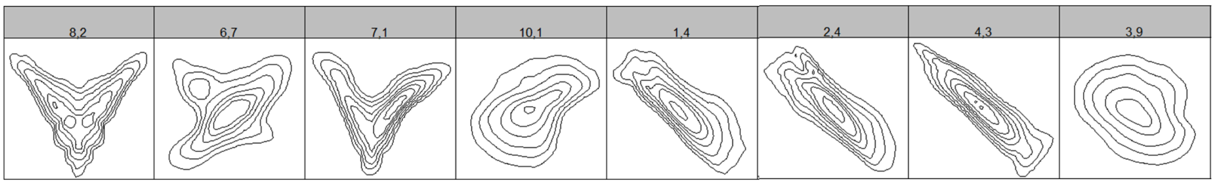

Figure 21: MAGIC data: Normalized contour plots of the pair copulas of the first tree level, that were estimated non-parametrically.

Performance Measures

- **confusion matrix and contingency table:** We compare the confusion matrices of the **non-parametric** vine fitting, the **parametric** vine fitting and the **mixed** vine fitting and the ***benchmark*** confusion matrix in the tables of Figure 22.

  The ***benchmark*** confusion matrix refers to the confusion matrix comparing the predictions of the logistic regression model *trained on the true data* to the true labels of the test set. It gives an impression on how difficult the classification problem is already on the true data.

- **$\mathcal{F}$-measure:** Recall Definition 3.2.2 of the $\mathcal{F}$-measure:

$$\boldsymbol{\mathcal{F}} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \ .$$

  The $\mathcal{F}$-measure values for **non-parametric**, **parametric** and **mixed** vine fitting can be found in Table 24. The ***benchmark***-value refers to the $\mathcal{F}$-measure regarding the predictions of the classifier *trained on the true data.* It gives an impression on how difficult the classification problem is already on the true data.

vine fitting	**$\mathcal{F}$-measure**	**precision, recall**
non-parametric	0.8400	precision = 0.7802, recall = 0.9096
parametric	0.8459	precision = 0.7937, recall = 0.9054
mixed	0.8412	precision = 0.7870, recall = 0.9052
benchmark	0.8545	precision = 0.8088, recall = 0.9057

Table 24: **$\mathcal{F}$-measure** of MAGIC gamma telescope data set.

- **AUC_{dir} and AUC_{dir}:** The performance in terms of the direct and indirect multi-class AUC (4.7) and (4.8), which in the dichotomous case is the regular AUC, can be read off of Figure 23 and Table 25.

vine fitting: NON-PARAMETRIC:

	$\hat{\boldsymbol{y}}^{(l),\,test}_{i,\,synth}=1$	$\hat{\boldsymbol{y}}^{(l),\,test}_{i,\,synth}=2$	$\sum$
$\boldsymbol{y}^{test}_{i,\,true}=1$	15639	14271	29910
$\boldsymbol{y}^{test}_{i,\,true}=2$	5031	50649	55680
$\sum$	20670	64920	85590

vine fitting: NON-PARAMETRIC:

	$\hat{\boldsymbol{y}}^{(l),\,test}_{i,\,synth}=1$	$\hat{\boldsymbol{y}}^{(l),\,test}_{i,\,synth}=2$	$\sum$
$\boldsymbol{y}^{test}_{i,\,true}=1$	0.1827	0.1667	0.3495
$\boldsymbol{y}^{test}_{i,\,true}=2$	0.0588	0.5918	0.6505
$\sum$	0.2415	0.7585	1

vine fitting: PARAMETRIC:

	$\hat{\boldsymbol{y}}^{(l),\,test}_{i,\,synth}=1$	$\hat{\boldsymbol{y}}^{(l),\,test}_{i,\,synth}=2$	$\sum$
$\boldsymbol{y}^{test}_{i,\,true}=1$	16806	13104	29910
$\boldsymbol{y}^{test}_{i,\,true}=2$	5266	50414	55680
$\sum$	22072	63518	85590

vine fitting: PARAMETRIC:

	$\hat{\boldsymbol{y}}^{(l),\,test}_{i,\,synth}=1$	$\hat{\boldsymbol{y}}^{(l),\,test}_{i,\,synth}=2$	$\sum$
$\boldsymbol{y}^{test}_{i,\,true}=1$	0.1964	0.1531	0.3495
$\boldsymbol{y}^{test}_{i,\,true}=2$	0.0615	0.5890	0.6505
$\sum$	0.2579	0.7421	1

vine fitting: MIXED:

	$\hat{\boldsymbol{y}}^{(l),\,test}_{i,\,synth}=1$	$\hat{\boldsymbol{y}}^{(l),\,test}_{i,\,synth}=2$	$\sum$
$\boldsymbol{y}^{test}_{i,\,true}=1$	16271	13639	29910
$\boldsymbol{y}^{test}_{i,\,true}=2$	5278	50402	55680
$\sum$	21549	64041	85590

vine fitting: MIXED:

	$\hat{\boldsymbol{y}}^{(l),\,test}_{i,\,synth}=1$	$\hat{\boldsymbol{y}}^{(l),\,test}_{i,\,synth}=2$	$\sum$
$\boldsymbol{y}^{test}_{i,\,true}=1$	0.1901	0.1594	0.3495
$\boldsymbol{y}^{test}_{i,\,true}=2$	0.0617	0.5889	0.6505
$\sum$	0.2518	0.7482	1

BENCHMARK:

	$\hat{\boldsymbol{y}}^{test}_{i,\,true}=1$	$\hat{\boldsymbol{y}}^{test}_{i,\,true}=2$	$\sum$
$\boldsymbol{y}^{test}_{i,\,true}=1$	1199	795	1994
$\boldsymbol{y}^{test}_{i,\,true}=2$	350	3362	3712
$\sum$	1549	4157	5706

BENCHMARK:

	$\hat{\boldsymbol{y}}^{test}_{i,\,true}=1$	$\hat{\boldsymbol{y}}^{test}_{i,\,true}=2$	$\sum$
$\boldsymbol{y}^{test}_{i,\,true}=1$	0.2101	0.1393	0.3495
$\boldsymbol{y}^{test}_{i,\,true}=2$	0.0613	0.5892	0.6505
$\sum$	0.2715	0.7285	1

Figure 22: Comparing the confusion matrices of the **non-parametric**, **parametric** and **mixed** vine estimation method with the **benchmark** confusion matrix of the **MAGIC gamma telescope data**. In the left column the confusion matrices are given, in the right column the values of the corresponding confusion matrix are divided by the total number of observations giving percentages.

- **comparing $\boldsymbol{AUC(y^{test}_{true},\,\hat{y}^{test}_{true})}$ and $\boldsymbol{AUC(y^{test}_{true},\,\hat{y}^{(l),\,test}_{synth})}$:**

  To determine how well vines are suited for generating synthetic data for classification, we calculate $AUC(\boldsymbol{y}^{test}_{true},\,\hat{\boldsymbol{y}}^{test}_{true})$, from which we learn, how difficult the classification problem is on the true data. Then we compute the synthetic counterparts $AUC(\boldsymbol{y}^{test}_{true},\,\hat{\boldsymbol{y}}^{(l),\,test}_{synth})$, $l \in [15]$ and compare to $AUC(\boldsymbol{y}^{test}_{true},\,\hat{\boldsymbol{y}}^{test}_{true})$, see Figure 24.

  We find, that the $AUC(\boldsymbol{y}^{test}_{true},\,\hat{\boldsymbol{y}}^{(l),\,test}_{synth})$, $l \in [15]$ of all three vine estimation methods are overall less than 4% off compared to $AUC(\boldsymbol{y}^{test}_{true},\,\hat{\boldsymbol{y}}^{test}_{true}) \approx 0.754$, see Figure 24.

measure	**vine estimation method**	**1st quartile**	**median**	**3rd quartile**
$\boldsymbol{mAUC_{dir}}$	non-parametric	0.9186	0.9206	0.9226
	parametric	0.9172	0.9210	0.9231
	mixed	0.9130	0.9150	0.9186
$\boldsymbol{mAUC_{indir}}$	non-parametric	0.0356	0.0377	0.0385
	parametric	0.0185	0.0205	0.0212
	mixed	0.0264	0.0295	0.0313

Table 25: Comparing the $\boldsymbol{mAUC_{dir}}$ and the $\boldsymbol{mAUC_{indir}}$ scores of the **non-parametric, parametric and mixed vine estimation methods** for the MAGIC gamma telescope data set.

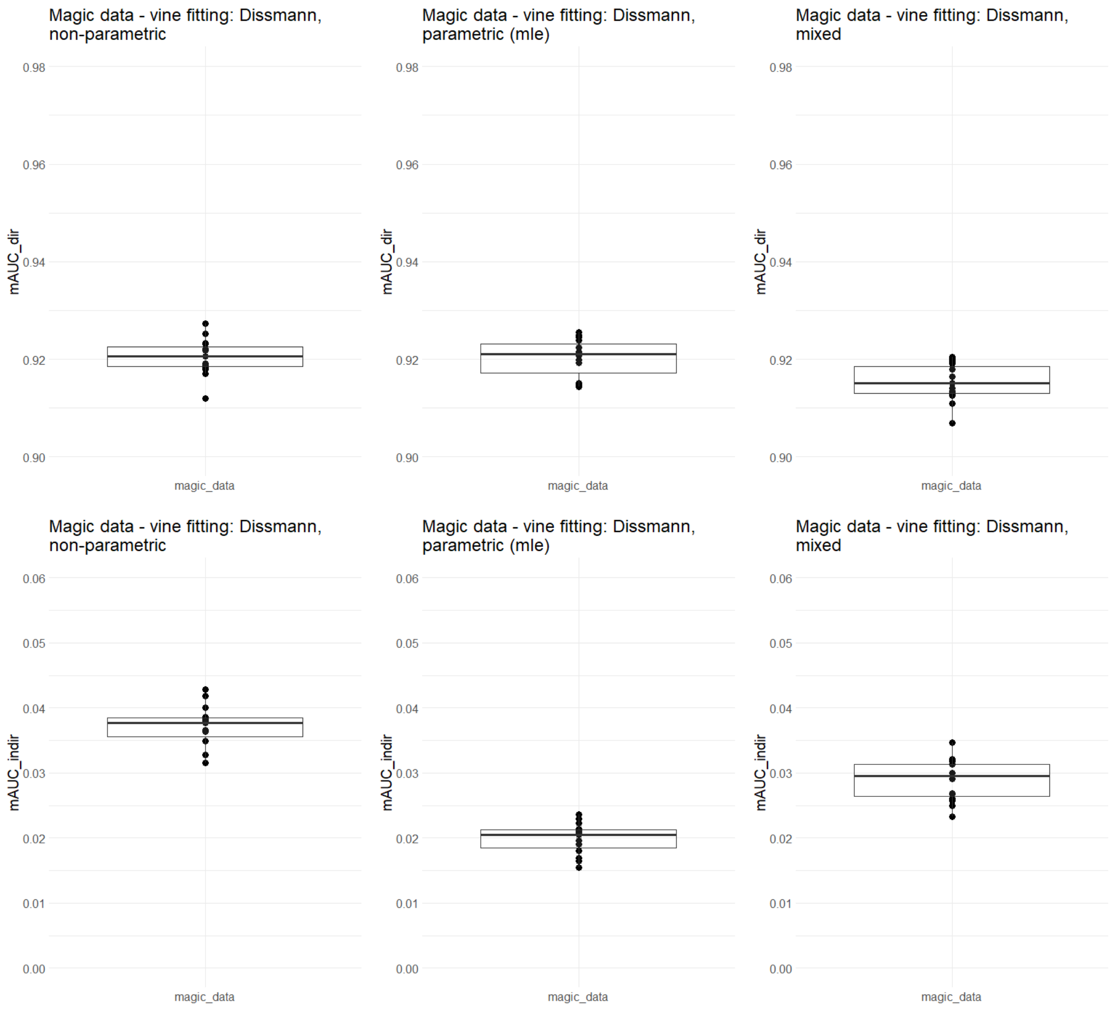

Figure 23: $\boldsymbol{mAUC_{dir}}$ and $\boldsymbol{mAUC_{dir}}$, direct and indirect mAUC of the MAGIC gamma telescope data.

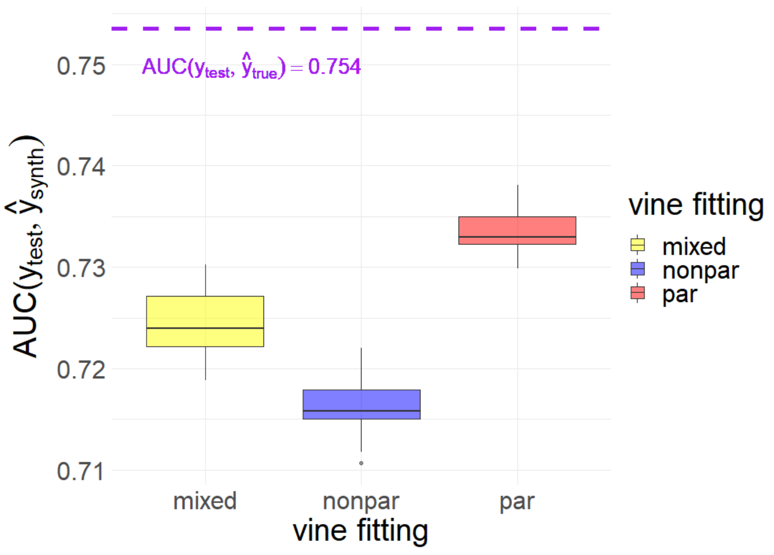

Figure 24: Plotting the $AUC(\boldsymbol{y}^{test}_{true}, \hat{\boldsymbol{y}}^{(l),\,test}_{synth})$, $l \in [15]$ per vine estimation method (non-parametric, parametric and mixed) in a boxplot. The dashed purple horizontal line indicates the value of $AUC(\boldsymbol{y}^{test}_{true}, \hat{\boldsymbol{y}}^{test}_{true})$, which the boxplots should be compared to.

This is still a satisfactory performance.

Variable Selection Performance Measures

As we are using binomial logistic regression **without** regularization for classification on the MAGIC data set, it does not make sense to consider $Discor(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth})$, the avergae number of discordant β-pairs, but only $Dist(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth})$, the average distance of the logistic regression β-coefficients.

measure	vine fitting	value	reference
$Dist(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth})$ (4.9)	non-parametric	0.0998	standardized β-coefficients, see (3.9)
	parametric (mle)	0.1806	
	mixed	0.1231	
	benchmark	–	

Table 26: $\boldsymbol{Dist(\hat{\beta}_{true}, \hat{\beta}_{synth})}$: variable selection performance measure for the MAGIC gamma telescope data set.

Vine Estimation Method Comparison – MAGIC Gamma Telescope Data

- **All three vine estimation methods perform well** in terms of prediction and variable selection in generating synthetic data suitable to train a logistic classifier similar as on the true data.

- The **non-parametric vine estimation method** produces synthetic estimates $\hat{y}^{(l),\,test}_{i,\,synth}$, that **underestimate** $\hat{y}^{test}_{i,\,true}$ the **least often**, namely in only 5.88% of the cases. This is even better than the benchmark with 6.13% and might be due to randomness, see tables of Figure 22.

- The **parametric** and **mixed** vine estimation methods follow with 6.15% and 6.17% respectively, see tables of Figure 22.
- At the same time, the **non-parametric vine fitting** produces $\hat{y}_{i,\,synth}^{(l),\,test}$, that **overestimate** $\hat{y}_{i,\,true}^{test}$ in 16.67% of the cases. This is the **most often**.
- In terms of **overestimation** the **parametric vine fitting** is to be preferred with 15.31% labels overestimated, closely followed by the **mixed vine fitting** with 15.94%, see again tables of Figure 22. All of the vine fitting methods score close to the benchmark with 13.93%, see Table **??**.
- In terms of the $\mathcal{F}$-measure **all three vine estimation methods** perform **similarly well**: the **parametric vine fitting** only slightly scores better with 0.8459 than the **mixed vine estimation method** with 0.8412 and the **non-parametric vine fitting** with 0.8400, see Table 24. These values are very close to the benchmark with 0.8545, see Table 24.
- Regarding $mAUC_{\boldsymbol{dir}}$ as well as $mAUC_{\boldsymbol{indir}}$ the **parametric vine estimation method** is to be slightly preferred to the **mixed** and the **non-parametric** vine fitting. All three methods achieve a $mAUC_{\boldsymbol{dir}} \geq 91.5\%$ at median and a $mAUC_{\boldsymbol{indir}} \leq 3.8\%$ at median, which is satisfactory.
- Is the focus on **variable selection performance**, the **non-parametric vine estimation method** outperforms the remaining two in terms of $Dist(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth})$.

Overall Performance of the Vine Copula Synthetic Data Generator on the MAGIC Data

How well are vine copulas in general suited for generating synthetic data for classifying the MAGIC gamma telescope data? According to the confusion matrices of Table 22 **very well**: A model trained on the true data classifies around 20 out of 100 patients incorrectly. A model trained on synthetic data produced by vine copulas classifies 21 to 22 out of 100 patients incorrectly. This is a very good performance.

For the $\mathcal{F}$-measure Table 24 shows a similarly good picture. The $\mathcal{F}$-measure of the classifier trained on the true data is at most $\approx 1.5\%$ higher than compared to the one of the classifier trained on synthetic data by vines, which is very good.

Finally, the $AUC(\boldsymbol{y}_{true}^{test}, \hat{\boldsymbol{y}}_{synth}^{(l),\,test})$, $l \in [15]$ of all three vine estimation methods are at mean overall less than 4% off compared to $AUC(\boldsymbol{y}_{true}^{test}, \hat{\boldsymbol{y}}_{true}^{test}) \approx 0.754$, see Figure 24. This is satisfactory.

	measure	**vine fitting**
performance	underestimation	🟦 non-parametric
		parametric
		mixed
	overestimation	non-parametric
		🟦 parametric
		mixed
	$\mathcal{F}$-measure	non-parametric
		🟦 parametric
		mixed
	$mAUC_{\boldsymbol{dir}}$	non-parametric
		🟦 parametric
		mixed
	$mAUC_{\boldsymbol{indir}}$	non-parametric
		🟦 parametric
		mixed
interpretability	$Dist(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth})$	🟦 non-parametric
		parametric
		mixed
	$Discor(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth})$	–
		–
		–

Table 27: Comparison of non-parametric, parametric and mixed vine estimation method at a glance for the MAGIC gamma telescope data set. Coloured cells indicate, which vine fitting should be preferred for a chosen measure.

7 Application II: Blueprint Study Data Set

We apply our methodology to the *Blueprint study data* of Qian et al. (2020). They analyse the stromal compartment of the tumor microenvironment, which consists of a heterogeneous set of tissue-resident and tumor-infiltrating cells. The latter are profoundly moulded by cancer cells. "An outstanding question is to what extent this heterogeneity is similar between cancers affecting different organs." In their paper Qian et al. (2020) profile 233,591 single cells from patients with lung, colorectal, ovary and breast cancer (n = 36) and construct a pan-cancer blueprint of stromal cell heterogeneity using different single-cell RNA and protein-based technologies.

For our application:

- we sample 1000 observations from the original Blueprint data set.
- As response variable we choose the expression of gene *ESR1*, which is a continuous variable with observations in $[-2.7664, 10.0]$:

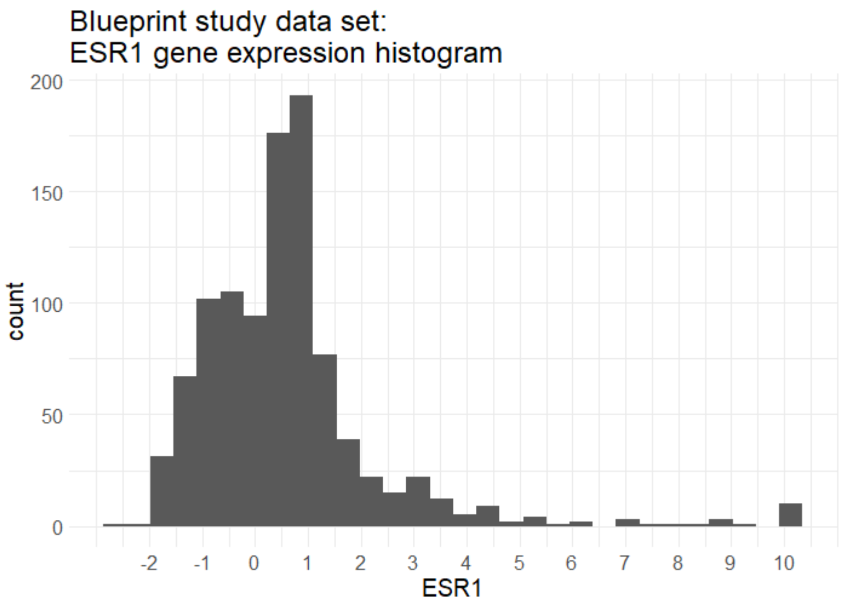

Figure 25

- We limit ourselves to 100 covariates. We choose the genes, whose expression is the most correlated with the expression of ESR1. The data is still sparse in its correlation structure as can be seen in Figure 26. That is why we use a logistic regression with Lasso for classification.
- The gene expression data, i.e. the covariates are continuous.
- To obtain labels $Y \in \{1, 2\}$, we threshold ESR1 and set:

$$y_i := \text{"1" for } |ESR1_i| < 1 \ , \qquad y_i := \text{"2" for } |ESR1_i| \geq 1 \ , \quad i \in [1000] \ , \tag{7.1}$$

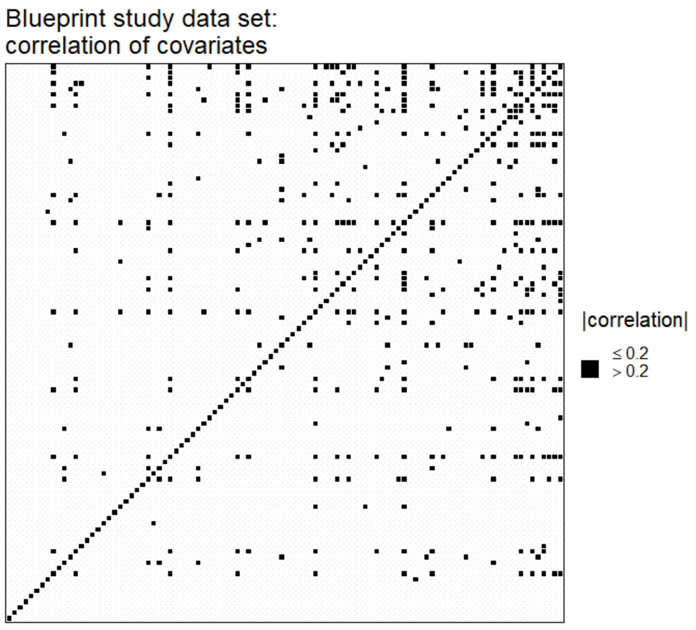

Figure 26: Correlation matrix of the 100 covariates of the Blueprint study data set. Components of the matrix are coloured in black, if the corresponding covariates have an **absolute correlation > 0.2**. As it is sparse, we use a logistic regression with Lasso for classification.

where the "2" is the biologically interesting event. By this we obtain a more balanced data set in terms of response labels with 630 observations with label "1" and 370 observations with label "2".

7.1 Results – Application II: Blueprint Study Data Set

The methodology of Section 4 was used on the Blueprint study data set. We generated:

- $p_{synth} = 15$ synthetic data sets $\left(\boldsymbol{x}_{synth}^{(l)}\ \boldsymbol{y}_{synth}^{(l)}\right)$, $l \in [15]$ from the Blueprint study data,
- each containing the same number of instances as the original Blueprint study data, namely 1000 instances.
- For a fixed $l \in [15]$ we compare:
  - $\lfloor$ 30% Lasso test share $\times 1000\rfloor = 300$ predictions of a (logistic Lasso) classifier trained on 70% of $\left(\boldsymbol{x}_{synth}^{(l)}\ \boldsymbol{y}_{synth}^{(l)}\right)$ to
  - $\lfloor$ 30% Lasso test share $\times 1000\rfloor = 300$ predictions of the same type of classifier trained on 70% of the Blueprint study data.
- This yields $15 \cdot 300 = 4500$ comparisons in total.

A Note on the Selection of the Lasso Regularization Parameter λ

As explained on pages 48 and following, we need to choose the penalty parameter λ, which governs the amount of regularization in the logistic regression model with Lasso. This is done with 10-fold cross-validation and deviance as loss. The value for λ is picked, which corresponds to the model with deviance one standard error above (i.e. towards the positive λ-direction of) the minimal deviance achieved. The reason for selecting a more restrictive value for λ is, that over-parameterized models tend to be selected when choosing the λ value that results in the model with minimal deviance.

When we follow this standard procedure for selecting the regularization parameter λ for the logistic Lasso model on the Blueprint study data, all β-coefficients are set to 0. However, we *know* that the features have an influence on the response as the Blueprint breast cancer data has been studied before. Thus, a logistic regression model containing only the intercept and no predictors does not make sense! The chosen λ is too strict.

For this reason we deviate from the standard procedure of selecting the Lasso regularization parameter and pick λ to be equal to the λ value, that results in the model with minimal deviance.

Pair Copula Estimation in the Parametric and Mixed Vine Fitting

It is interesting to observe, which pair copula families were chosen in the estimation of the R-vine distribution in $\boldsymbol{T_1}$ displayed in Figure 27 as well as for **all trees $\boldsymbol{T_1, \ldots, T_{d-1}}$** displayed in Figure 28 for the **parametric** and the **mixed** vine fitting:

- We find, that in the **mixed** vine estimation method **97 out of the 100** pair copulas of T_1 are **estimated non-parametrically** with transformation kernels, which corresponds to the symbol "tll" in Figure 27.
- In T_1 the **parametric** vine fitting selects the Student's t pair copula most often, i.e. in almost 50% of the cases, followed by the BB8 pair copula, see Figure 27.
- From Figure 27 we observe overall, that in T_1 the parametric and mixed vine fitting preferably select pair copula families with two parameters (e.g. BB8 and Student's t) or very flexible pair copulas, i.e. the transformation kernel estimator.
- In Figure 28 we find, that for **non-parametric, parametric and mixed** vine fitting the **pair copula selected the most often** in trees $T_1, \ldots, T_{d-1}$ is the **independence copula**. This lets us assume, that at a certain tree level we have truncation, meaning that the remaining conditional pair copulas are set to independence copulas.
- For the **parametric** vine fitting the BB8, Frank and Student's t copula are overall predominating besides the independence copula.
- For the **mixed** vine fitting the Student's t, Frank and Clayton pair copula and the non-parametric transformation kernel estimation are overall predominating.

- Overall we observe more than **double the number of independence pair copulas** selected in trees $T_1, \ldots, T_{d-1}$ by the **non-parametric** vine fitting as by the parametric or by the mixed vine fitting. This indicates a **lower truncation level** of the R-vine distribution estimated with the non-parametric vine fitting. This might suggest that the non-parametric vine fitting was able to capture important dependence structures already on lower tree levels than the parametric and mixed vine fitting.
- Overall the independence pair copulas are clearly the dominating pair copula family.
- Besides those, pair copulas, which are able to model tail dependence, such as the Student's t, Gumbel, Clayton and Joe, as well as pair copulas not modeling tail dependence, such as the Gauss or Frank pair copula are fit to data. There seems to be no clear pattern recognizable.

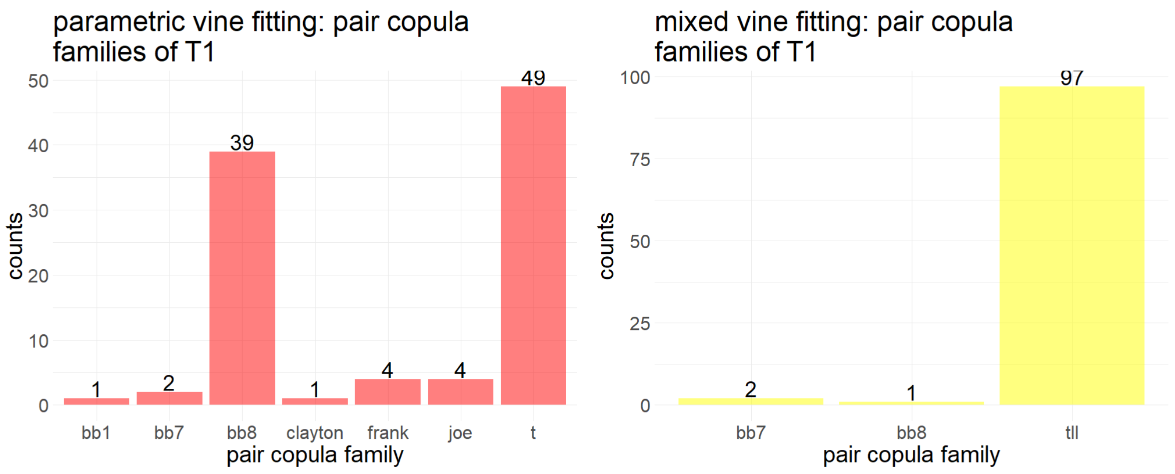

Figure 27: Counting the pair copula families chosen for estimation of the R-vine distribution in $\boldsymbol{T_1}$ for the **parametric** and **mixed** vine fitting. The total number of pair copulas estimated in T_1 is equal to 100.

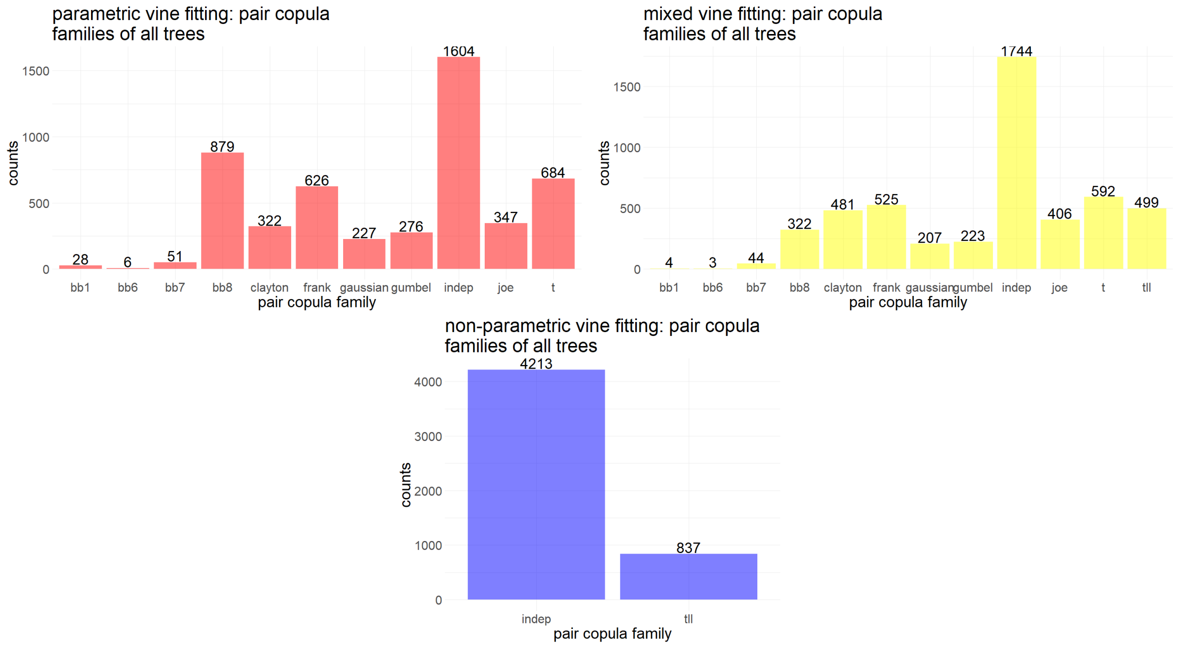

Figure 28: Counting the pair copula families chosen for estimation of the R-vine distribution in T_1 for the **non-parametric**, **parametric** and **mixed** vine fitting. The total number of pair copulas estimated in the R-vine distribution is equal to 5050.

Performance Measures

- **confusion matrix and contingency table:** We compare the confusion matrices of the **non-parametric** vine fitting, the **parametric** vine and the **mixed** vine fitting with the ***benchmark*** confusion matrix in the tables of Figure 29.

  The ***benchmark*** confusion matrix refers to the confusion matrix comparing the predictions of the classifier *trained on the true data* to the true labels. It gives an impression on how difficult the classification problem is already on the true data.

vine fitting: NON-PARAMETRIC:

	$\hat{y}^{(l),\,test}_{i,\,synth} = 1$	$\hat{y}^{(l),\,test}_{i,\,synth} = 2$	$\sum$
$y^{test}_{i,\,true} = 1$	2670	75	2745
$y^{test}_{i,\,true} = 2$	1605	150	1755
$\sum$	4275	225	4500

vine fitting: NON-PARAMETRIC:

	$\hat{y}^{(l),\,test}_{i,\,synth} = 1$	$\hat{y}^{(l),\,test}_{i,\,synth} = 2$	$\sum$
$y^{test}_{i,\,true} = 1$	0.5933	0.0167	0.6100
$y^{test}_{i,\,true} = 2$	0.3567	0.0333	0.3900
$\sum$	0.9500	0.0500	1

vine fitting: PARAMETRIC:

	$\hat{y}^{(l),\,test}_{i,\,synth} = 1$	$\hat{y}^{(l),\,test}_{i,\,synth} = 2$	$\sum$
$y^{test}_{i,\,true} = 1$	2484	261	2745
$y^{test}_{i,\,true} = 2$	1258	497	1755
$\sum$	3742	758	4500

vine fitting: PARAMETRIC:

	$\hat{y}^{(l),\,test}_{i,\,synth} = 1$	$\hat{y}^{(l),\,test}_{i,\,synth} = 2$	$\sum$
$\hat{y}^{test}_{i,\,true} = 1$	0.5520	0.0580	0.6100
$\hat{y}^{test}_{i,\,true} = 2$	0.2796	0.1104	0.3900
$\sum$	0.8316	0.1684	1

vine fitting: MIXED:

	$\hat{y}^{(l),\,test}_{i,\,synth} = 1$	$\hat{y}^{(l),\,test}_{i,\,synth} = 2$	$\sum$
$y^{test}_{i,\,true} = 1$	2571	174	2745
$y^{test}_{i,\,true} = 2$	1366	389	1755
$\sum$	3937	563	4500

vine fitting: *MIXED*

	$\hat{y}^{(l),\,test}_{i,\,synth} = 1$	$\hat{y}^{(l),\,test}_{i,\,synth} = 2$	$\sum$
$y^{test}_{i,\,true} = 1$	0.5713	0.0387	0.6100
$y^{test}_{i,\,true} = 2$	0.3036	0.0864	0.3900
$\sum$	0.8749	0.1251	1

BENCHMARK:

	$\hat{y}^{test}_{i,\,true} = 1$	$\hat{y}^{test}_{i,\,true} = 2$	$\sum$
$y^{test}_{i,\,true} = 1$	173	10	183
$y^{test}_{i,\,true} = 2$	89	28	117
$\sum$	262	38	300

BENCHMARK:

	$\hat{y}^{test}_{i,\,true} = 1$	$\hat{y}^{test}_{i,\,true} = 2$	$\sum$
$y^{test}_{i,\,true} = 1$	0.5767	0.0333	0.6100
$y^{test}_{i,\,true} = 2$	0.2967	0.0933	0.3900
$\sum$	0.0873	0.1267	1

Figure 29: Comparing the confusion matrices of the **non-parametric**, **parametric** and **mixed** vine estimation method with the **benchmark** confusion matrix of the **Blueprint study data**. In the left column the confusion matrices are given, in the right column the values of the corresponding confusion matrix are divided by the total number of observations giving percentages.

- **$\mathcal{F}$-measure:** Recall Definition 3.2.2 of the $\mathcal{F}$-measure. The $\mathcal{F}$-measure values for **non-parametric**, **parametric** and **mixed** vine fitting can be found in Table 28. The ***benchmark***-value refers to the $\mathcal{F}$-measure regarding the predictions of the classifier trained on the true data. It gives an impression on how difficult the classification problem is already on the true data.

vine fitting	**$\mathcal{F}$-measure**	**precision, recall**
non-parametric	0.1515	precision = 0.6667, recall = 0.0855
parametric	0.3955	precision = 0.6557, recall = 0.2832
mixed	0.3356	precision = 0.6909, recall = 0.2217
benchmark	0.3613	precision = 0.7368, recall = 0.2393

Table 28: **$\mathcal{F}$-measure** of Blueprint study data set.

- **AUC_{dir} and AUC_{dir}:** The performance in terms of the direct and indirect multi-class AUC (4.7) and (4.8), which in the dichotomous case is the regular AUC, can be read off of Figure 30 and Table 29.

measure	**vine estimation method**	**1st quartile**	**median**	**3rd quartile**
AUC_{dir}	non-parametric	0.6513	0.7086	0.7387
	parametric	0.7861	0.8285	0.8605
	mixed	0.8142	0.8457	0.8653
AUC_{indir}	non-parametric	0.0516	0.0617	0.0747
	parametric	0.0068	0.0214	0.0281
	mixed	0.0081	0.0204	0.0309

Table 29: Comparing the AUC_{dir} and the AUC_{indir} scores of the **non-parametric, parametric and mixed vine estimation methods** for the MAGIC gamma telescope data set.

- **comparing $AUC(y^{test}_{true}, \hat{y}^{test}_{true})$ and $AUC(y^{test}_{true}, \hat{y}^{(l),test}_{synth})$:**

  Again we want to also determine how well vines are suited for generating synthetic data for classification. Therefore we calculate $AUC(\boldsymbol{y}^{test}_{true}, \hat{\boldsymbol{y}}^{test}_{true})$, from which we learn, how difficult the classification problem is on the true data. Then we compute the synthetic counterparts $AUC(\boldsymbol{y}^{test}_{true}, \hat{\boldsymbol{y}}^{(l),test}_{synth})$, $l \in [15]$ and compare to $AUC(\boldsymbol{y}^{test}_{true}, \hat{\boldsymbol{y}}^{test}_{true})$, see Figure 24.

  - We find, that the $AUC(\boldsymbol{y}^{test}_{true}, \hat{\boldsymbol{y}}^{(l),test}_{synth})$, $l \in [15]$ of the parametric vine estimation method even exceeds $AUC(\boldsymbol{y}^{test}_{true}, \hat{\boldsymbol{y}}^{test}_{true}) \approx 0.69$, see Figure 24. This is might be due to randomness in the synthetic data.

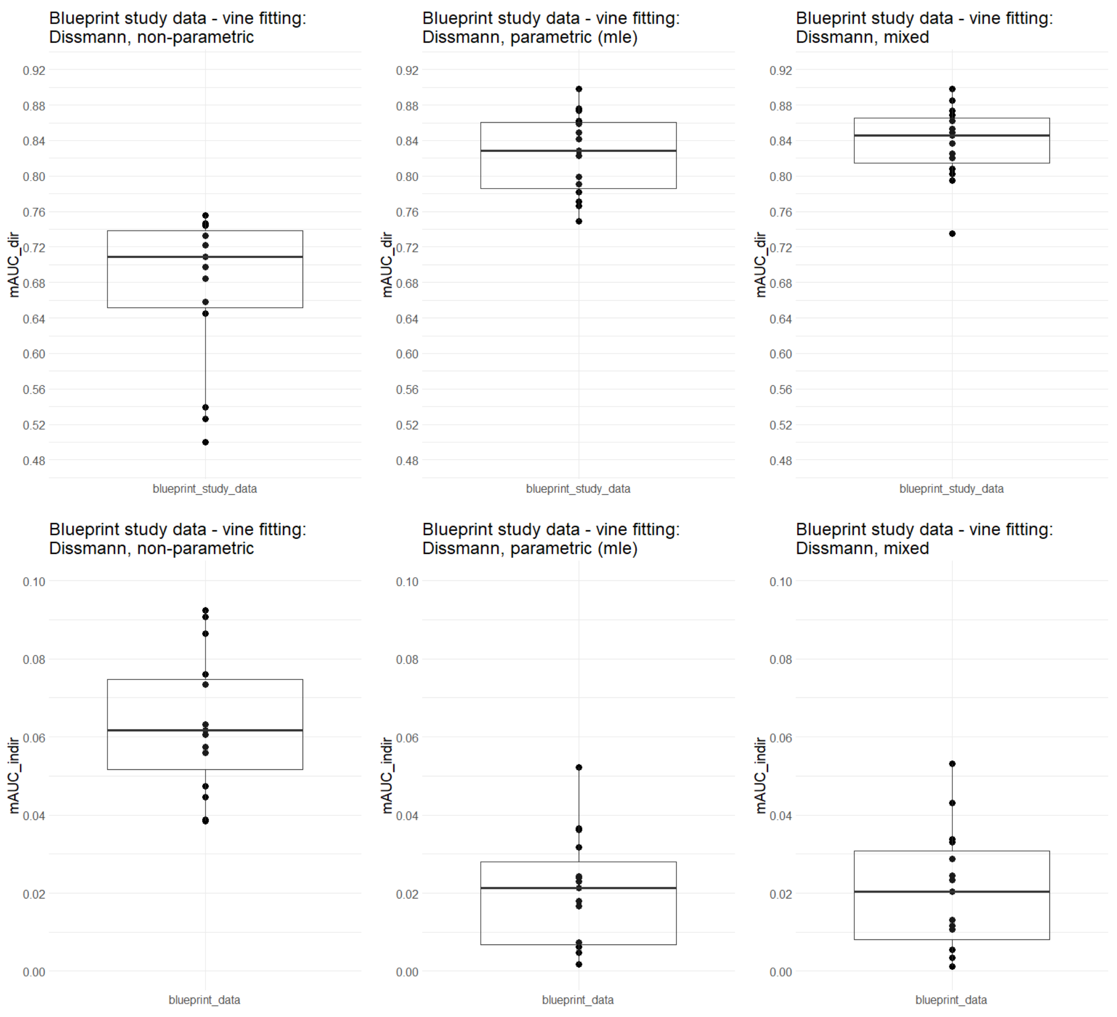

Figure 30: $\boldsymbol{mAUC_{dir}}$ and $\boldsymbol{mAUC_{dir}}$, direct and indirect mAUC of the Blueprint study data.

- The $AUC(\boldsymbol{y}_{true}^{test}, \hat{\boldsymbol{y}}_{synth}^{(l),test})$, $l \in [15]$ of the mixed vine fitting is at mean around 4% lower than $AUC(\boldsymbol{y}_{true}^{test}, \hat{\boldsymbol{y}}_{true}^{test})$.
- Summarized the vine estimation methods using (among others) parametric pair copula estimation are quite produce synthetic data, such that a classifier trained on them scores close in terms of the AUC to a classifier trained on the true data.
- Data from the non-parametric vine fitting yield classifier that scores an AUC that is around 6% lower at mean than the true AUC, which is still satisfactory.

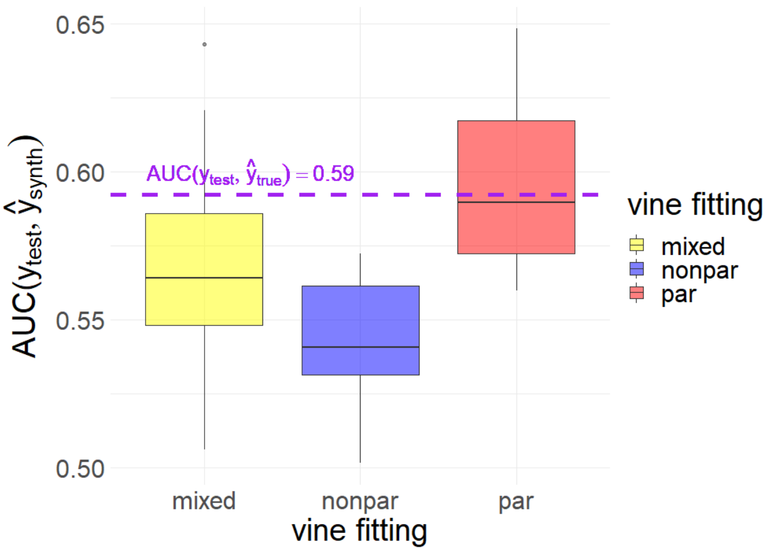

Figure 31: Plotting the $AUC(\boldsymbol{y}_{true}^{test}, \hat{\boldsymbol{y}}_{synth}^{(l),\,test})$, $l \in [15]$ per vine estimation method (non-parametric, parametric and mixed). The dashed purple horizontal line indicates the value of $AUC(\boldsymbol{y}_{true}^{test}, \hat{\boldsymbol{y}}_{true}^{test})$, which the boxplots should be compared to.

Variable Selection Performance Measures

measure	vine fitting	value	reference
$Dist(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth})$ (4.9)	non-parametric	0.0089	standardized β-coefficients, see (3.9)
	parametric (mle)	0.0181	
	mixed	0.0114	
	benchmark	-	
$Discor(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth})$ (4.10)	non-parametric	26.13	total number of regularized β-coefficients (without intercept): 100
	parametric (mle)	38.27	
	mixed	31.13	
	benchmark	-	

Table 30: $\boldsymbol{Dist(\hat{\beta}_{true}, \hat{\beta}_{synth})}$ and $\boldsymbol{Discor(\hat{\beta}_{true}, \hat{\beta}_{synth})}$: variable selection performance measures for the Blueprint study data set.

Variable Selection of the Non-Parametric Vine Estimation Method

We find, that in terms of variable selection the **non-parametric** vine estimation method performs better than the parametric or mixed vine fitting. That is why we have a closer look at it:
We plot the coefficient profiles of the logistic Lasso classifier trained on $\left(\boldsymbol{x}_{synth}\ \boldsymbol{y}_{synth}\right)^{(l)}$, $l \in [15]$ obtained from the **non-parametric** vine estimation method in Figure 32. It displays a curve for each regularized coefficient β_j, $j \in [d]$ of the logistic regression model, see Equation (3.11): the coefficient value is plotted versus the L_1-norm of the coefficient vector $\|(\beta_1, \ldots, \beta_d)^T\|_1$. We compare the coefficient profile plots of the 15 classifiers trained on $\left(\boldsymbol{x}_{synth}\ \boldsymbol{y}_{synth}\right)^{(l)}$, $l \in [15]$ to the logistic Lasso classifier trained on $\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right)$ of Figure 33.
Note, that the plot displays the *non-standardized* $\beta_j, j \in [d]$ values, which is due to the implementation in `glmnet`, Simon et al. (2011).

We also compare the following quantities in Table 31:

- the number of coefficients, that the Lasso includes into the logistic regression model when trained on $\left(\boldsymbol{x}_{synth}\ \boldsymbol{y}_{synth}\right)^{(l)}$, $l \in [15]$ and on $\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right)$ respectively (column 2):

$$|\{j \in [d] : \beta'_{j,true} \neq 0\}| \,.$$

- We compute the percentage of covariates $X_{j,true}, j \in [d]$ included in the true model, that are re-detected in the synthetic model (column 3):

$$\frac{|\{j \in [d] : \beta'_{j,synth} \neq 0 \wedge \beta'_{j,true} \neq 0\}|}{|\{j \in [d] : \beta'_{j,true} \neq 0\}|}$$

- and the number of covariates selected by the Lasso for the classifier trained on the synthetic data, but not for the classifier trained on the true data, which we call "false findings" (column 4):

$$|\{j \in [d] : \beta'_{j,\,synth} \neq 0 \wedge \beta'_{j,\,true} = 0\}| \,.$$

- Then we average the absolute value of the standardized coefficients, see Equation (3.9), denoted by $\beta'_j, j \in [d]$ of the "false findings" (column 5):

$$\frac{1}{|\{j \in [d] : \beta'_{j,\,synth} \neq 0 \wedge \beta'_{j,\,true} = 0\}|} \sum_{\substack{j \in [d] : \beta'_{j,\,synth} \neq 0 \,\wedge \\ \beta'_{j,\,true} = 0}} \beta'_{j,\,synth} \,,$$

- and give the average absolute standardized β-value of the selected covariates of the models built from true and synthetic data respectively (column 6):

$$\frac{1}{|\{\beta'_j \neq 0\}|} \sum_{\beta'_j \neq 0} |\beta'_j| \,.$$

For the variable selection measures as well as the values given in Table 31 we use the *standardized* coefficients $\beta'_j, j \in [d]$ with the prime $'$ indicating standardization as explained in Equation (3.9)!

Summarized we find:

- The classifier trained on the true data includes (by far) the most covariates, namely 24 of 100 possible.
- The majority, namely 8 out of 15 ($\approx 53\%$) of the logistic Lasso models trained on the synthetic data include 1 to 8 covariates. This is equal to a third or less of the covariates selected by the true classifier.
- In 6 out of 15 of (40%) of the cases the synthetic logistic Lasso model includes 9 to 16 covariates, which equals $\frac{1}{3}$ to $\frac{2}{3}$ of the covariates selected by the true classifier.
- Only 1 out of 15 synthetic models has approximately the size (21 covariates) of the model trained on the true data.
- We find, that each of the 15 synthetic logistic Lasso models is able to recognize at least 1 of the covariates, which are selected by the classifier trained on the true data.

  From the coefficient profile plots of Figure 33 we observe, that this is the covariate X_{97}. It is not just recognized each time: the coefficient profile of ${\beta^{(l)}_{97,\,synth}}'$ is for all $l \in [15]$ very similar to the coefficient profile of $\beta'_{97,\,true}$.
- It is very plausible, that the number of covariates included in the true model, which are correctly recognized by the synthetic Lasso model, increases as the synthetic model increases in size.

$l \in [p_{synth}]$	no. of $\beta'_j \neq 0$, $j \in [100]$	% (no.) of $X_{j,true}$ re-detected	no. of false findings	average $\lvert\beta'\rvert$ of false findings	average $\lvert\beta'\rvert$ of $\beta' \neq 0$
1	4	8.33% (2)	2	0.054	0.109
2	8	12.5% (3)	5	0.027	0.064
3	15	20.83% (5)	10	0.049	0.063
4	1	4.17% (1)	0	–	0.323
5	11	25% (6)	5	0.040	0.063
6	5	12.5% (3)	2	0.014	0.081
7	1	4.14 % (1)	0	–	0.248
8	2	4.17% (1)	1	0.003	0.055
9	4	4.17% (1)	3	0.024	0.075
10	8	12.5% (3)	5	0.042	0.078
11	12	20.83% (5)	7	0.032	0.064
12	13	16.67% (4)	9	0.027	0.073
13	14	20.83% (5)	9	0.039	0.075
14	21	20.83% (5)	16	0.052	0.066
15	15	25% (6)	9	0.049	0.069
true	24	–	–	–	0.077

Table 31: Comparing the logistic regression model with Lasso trained on $\left(\boldsymbol{x}_{synth}\ \boldsymbol{y}_{synth}\right)^{(l)}$, $l \in [15]$ and $\left(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true}\right)$ in detail. Description of the column values can be found on page 127 and following.

- In 9 of the 15 cases, the number of false findings exceeds the number of correctly identified features.
- We find that for each $l \in [15]$ the average $|\beta'|$ of the false findings does not exceed the average $|\beta'|$ of predictors included into the model. The average $|\beta'|$ of the logistic Lasso classifiers trained on $\left(\boldsymbol{x}_{synth}\ \boldsymbol{y}_{synth}\right)^{(l)}$, $l \in [15] \setminus \{4, 7\}$ are of the same magnitude as the average $|\beta'|$ of the logistic Lasso classifiers trained on the true data. The models trained on $\left(\boldsymbol{x}_{synth}\ \boldsymbol{y}_{synth}\right)^{(l)}$, $l \in \{4, 7\}$ only include one predictor, namely X_{97}, and estimate its coefficient higher than the average $|\beta'|$ obtained from the true data.
- No synthetic classifier is able to recognize more than 25% of the features included into the true logistic regression model with Lasso, see Tabl 31. It is at the same time important to keep in mind that the classification task on the blueprint data is difficult, see for example the benchmark $\mathcal{F}$-measure of Table 28 which is around 0.36. The degree to which the response Y depends on the covariates $X_1, \ldots, X_d$ is limited. For this reason, it may be more difficult for the logistic Lasso classifier trained on the synthetic data to find among the limited influential predictors the ones chosen by the model trained on the true data.

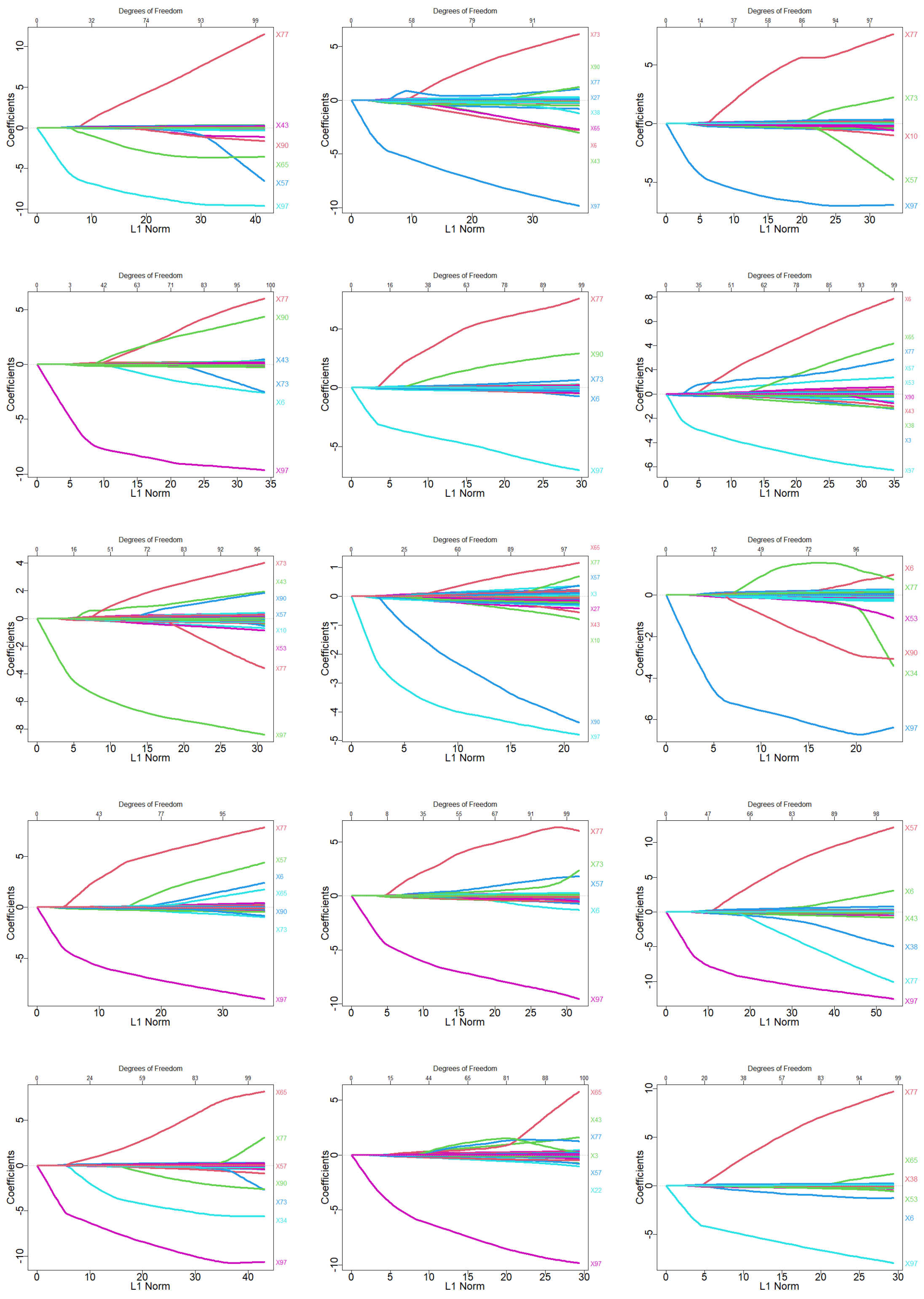

Figure 32: Plotting the **coefficient profiles of the logistic Lasso classifier** trained on the **synthetic data** $(x_{synth}\ \boldsymbol{y}_{synth})^{(l)}$, $\quad l \in [15]$ obtained from the **non-parametric** vine fitting. Each curve corresponds to a regularized coefficient β_j, $j \in [d]$ of the logistic regression model of Equation (3.11) plotting the coefficient value versus $\|(\beta_1, \ldots, \beta_d)^T\|_1$.

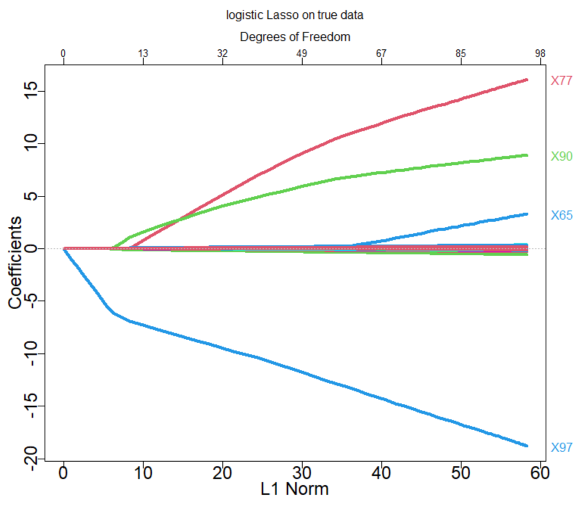

Figure 33: Plotting the **coefficient profiles of the logistic Lasso classifier** trained on the **true data** $(\boldsymbol{x}_{true}\ \boldsymbol{y}_{true})$. Each curve corresponds to a regularized coefficient β_j, $j \in [d]$ of the logistic regression model of Equation (3.11) plotting the coefficient value versus $\|(\beta_1, \ldots, \beta_d)^T\|_1$.

Vine Estimation Method Comparison – Blueprint Study Data

- The **non-parametric vine estimation method** produces synthetic estimates $\hat{y}_{i,synth}^{(l),test}$, that **overestimate** $\hat{y}_{i,true}^{test}$ the **least often**, namely in only 1.67% of the cases. This is even better than the benchmark with 3.33% and might be due to randomness.

- The **mixed** and **parametric** vine estimation methods follow with 3.87% and 5.8% respectively, see tables of Figure 29.

- In terms of **underestimation** the **parametric vine fitting** is to be preferred with 27.96% labels overestimated, followed by the **mixed vine fitting** with 30.36% and the **non-parametric vine fitting** with 35.67%, which performs worst here.

- The **parametric vine fitting** even outperforms the benchmark, in which 29.67% of the labels are underestimated by the classifier trained on the synthetic data, see again tables of Figure 29. This might be due to randomness.

- In terms of the $\mathcal{F}$-measure the **non-parametric vine fitting** clearly scores **worse** with 0.1515 than the remaining two vine fittings and the benchmark. This is due to a poor recall of 0.0855, see Table 28.

- The $\mathcal{F}$-measure of the **mixed vine fitting** is with 0.3356 quite close to the benchmark value of 0.3613. It is even exceeded by the **parametric vine fitting** with 0.3955. Although it has a lower precision, the recall of the parametric vine fitting is better than in the benchmark. This might again be due to randomness.

- Regarding $mAUC_{\boldsymbol{dir}}$ as well as $mAUC_{\boldsymbol{indir}}$ the **mixed vine estimation method** is to be **slightly preferred** to the **parametric** vine fitting: with a $mAUC_{\boldsymbol{dir}}$-value of 0.8457 at median for the mixed and 0.8285 at median for the parametric and a $mAUC_{\boldsymbol{indir}}$-value of 0.0204 at median for the mixed and 0.0214 at median for the parametric. The **non-parametric** vine fitting performs worst by some distance with 0.7086 for $mAUC_{\boldsymbol{dir}}$ and 0.617 for $mAUC_{\boldsymbol{indir}}$.

- Is the focus on **variable selection performance**, the **non-parametric vine estimation method** outperforms the remaining two in terms of $Dist(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth})$ and $Discor(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth})$. Especially in the latter **non-parametric vine fitting** performs quite well.

- It is interesting to observe, that the **parametric** vine fitting scores worst in terms of the variable selection performance measures $Dist(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth})$ and $Discor(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth})$ compared to the non-parametric and mixed vine estimation method. It is also interesting to observe, that the distance of the $Dist(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth})$ values and $Discor(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth})$ values between the parametric and the mixed vine fitting is larger than the distance between the two values of mixed and non-parametric vine fitting. This might be due to the fact, that in the mixed vine estimation 97 out of 100 pair copulas of the first tree level are estimated non-parametrically instead of parametrically.

- Why is parametric vine fitting performing quite well compared to the performance measure scores of the mixed and the non-parametric vine fitting, when we find, that in the mixed fitting 97 out of 100 pairs of T_1 are estimated non-parametrically and thus have a better AIC than the parametric pair copula fits?

  The reason for this might be low dependence in the data. When plotting the absolute value of Kendall's τ estimates of T_1 of the mixed vine fitting in Figure 34 we find this assumption supported to some extent: 78 out of 100 of the estimated pair copulas of T_1 have a absolute Kendall's τ value below 0.4, which can be regarded as low dependence.

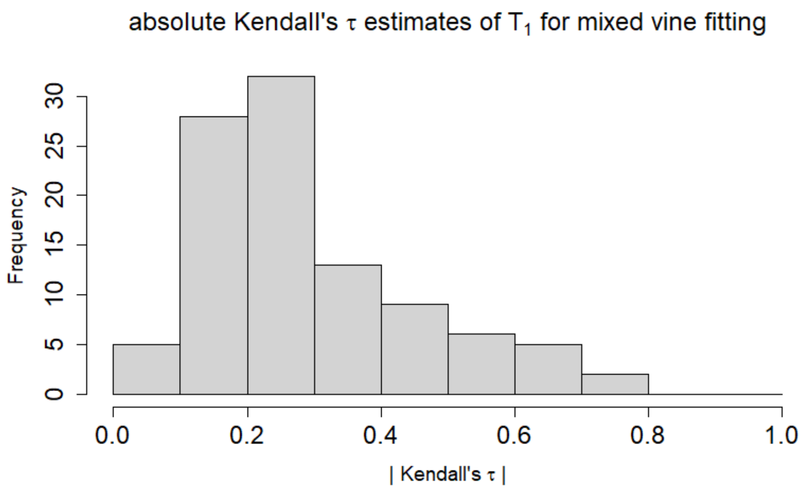

Figure 34: Histogram of the absolute Kendall's τ values of T_1 of the mixed vine fitting. We find that 78 out of 100 of the estimated pair copulas of T_1 have a absolute Kendall's τ value below 0.4.

Overall Performance of the Vine Copula Synthetic Data Generator on the Blueprint Data

How well are vine copulas in general suited for generating synthetic data for classifying the Bluepttrint breast cancer study data? According to the confusion matrices of Table 29 **quite well**: A model trained on the true data classifies around 31 out of 100 patients incorrectly. The classification task at hand is rather difficult. A model trained on synthetic data produced by vine copulas classifies 33 to 37 out of 100 patients incorrectly. This is a very close to the performance of the classifier trained on the true data.

For the $\mathcal{F}$-measure Table 28 shows a similar picture. The $\mathcal{F}$-measure of the classifier trained on the true data is only around $\approx 2.5\%$ off compared to the one of the classifier trained on synthetic data by vines using (also) parametric pair copula estimation, i.e. the parametric and mixed vine fitting. Only the non-parametric vine fitting shows some difficulties in terms of the $\mathcal{F}$-measure, more precisely in the recall, see Table 28.

	measure	**vine fitting**
performance	underestimation	non-parametric
		parametric 🟦
		mixed
	overestimation	non-parametric 🟦
		parametric
		mixed
	$\mathcal{F}$-measure	non-parametric
		parametric 🟦
		mixed
	$mAUC_{\boldsymbol{dir}}$	non-parametric
		parametric
		mixed 🟦
	$mAUC_{\boldsymbol{indir}}$	non-parametric
		parametric 🟦
		mixed 🟦
interpretability	$Dist(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth})$	non-parametric 🟦
		parametric
		mixed
	$Discor(\hat{\boldsymbol{\beta}}_{true}, \hat{\boldsymbol{\beta}}_{synth})$	non-parametric 🟦
		parametric
		mixed

Table 32: Comparison of non-parametric, parametric and mixed vine estimation method at a glance for the Blueprint study data set. Coloured cells indicate, which vine fitting should be preferred for a chosen measure.

Finally, the $AUC(\boldsymbol{y}_{true}^{test}, \hat{\boldsymbol{y}}_{synth}^{(l),test})$, $l \in [15]$ of all three vine estimation methods are at mean less than 6% off compared to $AUC(\boldsymbol{y}_{true}^{test}, \hat{\boldsymbol{y}}_{true}^{test})$, see Figure 31. This is satisfactory.

8 Conclusion and Further Work

In this master thesis we used vine copulas to generate synthetic data for classification: We fit a vine copula model to the true data using Dißmann's algorithm for R-vine selection and non-parametric, parametric and mixed pair copula estimation. From this fitted model we sampled synthetic data, which we used to train a logistic regression classifier (with Lasso).

We applied this method to data sets from astronomy (MAGIC gamma telescope data) and cancer genomics (Blueprint breast cancer study data). The classifier trained on the synthetic data generated with our methodology achieved overall very comparable results to those of the same classifier trained on the true data (benchmark) considering classification error, the $\mathcal{F}$-measure and AUC.

We also compared the three vine estimation methods, which utilize non-parametric, parametric and mixed pair copula estimation, to each other in simulations and on the previously mentioned MAGIC Gamma telescope data and the Blueprint breast cancer study data. On the Magic data we found that the parametric vine estimation method performs best in terms of classification performance. When the focus is on variable selection, the non-parametric vine estimation method achieved the best results. When applying the methodology to the Blueprint data, the picture regarding classification performance was slightly less clear: parametric and mixed vine fitting achieved the best results with respect to classification performance. Only regarding overestimation did the non-parametric vine fitting outperform the remaining two. If we look at the variable selection performance, the non-parametric vine fitting again clearly dominates. Overall, the differences between the vine fitting methods in classification and variable selection performance are not very large but increase with increasing difficulty of the classification task. In the closer examination of the non-parametric vine estimation method on the Blueprint data, it became clear that the classifier trained on the synthetic data is able to recognize the covariate that the true classifier weights highest on each of the 15 synthetic data sets produced. However, the percentage of recognized features did not exceed 25%. One reason for this may be that the classification task on the Blueprint data is difficult in the first place, which may be why it is challenging to identify the set of predictors that influence the response the most.

A continuation of this work should verify, whether synthetic data generated by vine copulas protect the privacy of the real data within the framework of differential privacy, see Dwork et al. (2006), Dwork (2011) and Dwork et al. (2014). Vine copulas as synthetic data generators should furthermore be compared to competitor methods, such as GANs and VAEs on a classification task. It would also be very interesting to apply D-vine regression to generate synthetic data. One could also investigate whether vine fitting with non-parametric pair copula estimation using local likelihood transformation estimator, see Nagler (2014), instead of transformation kernels could generate synthetic data improving classification results.

References

K. Aas, C. Czado, A. Frigessi, and H. Bakken. Pair-copula constructions of multiple dependence. *Insurance: Mathematics and economics*, 44(2):182–198, 2009.

H. Akaike. Information theory and an extension of the maximum likelihood principle. In *Selected papers of hirotugu akaike*, pages 199–213. Springer, 1998.

M. Arjovsky and L. Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

D. Ballabio, F. Grisoni, and R. Todeschini. Multivariate comparison of classification performance measures. *Chemometrics and Intelligent Laboratory Systems*, 174:33–44, 2018.

T. Bedford and R. M. Cooke. Probability density decomposition for conditionally dependent random variables modeled by vines. *Annals of Mathematics and Artificial intelligence*, 32(1):245–268, 2001.

T. Bedford and R. M. Cooke. Vines–a new graphical model for dependent random variables. *The Annals of Statistics*, 30(4):1031–1068, 2002.

R. Bock. UCI machine learning repository: Magic data set, 2007. URL `https://archive.ics.uci.edu/ml/datasets/MAGIC+Gamma+Telescope`.

L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

M. Brokate, N. Henze, F. Hettlich, A. Meister, G. Schranz-Kirlinger, and T. Sonar. *Grundwissen mathematikstudium*. Springer, 2016.

A. Charpentier, J.-D. Fermanian, and O. Scaillet. The estimation of copulas: Theory and practice. *Copulas: From theory to application in finance*, pages 35–64, 2007.

E. Choi, S. Biswal, B. Malin, J. Duke, W. F. Stewart, and J. Sun. Generating multi-label discrete patient records using generative adversarial networks. In *Machine learning for healthcare conference*, pages 286–305. PMLR, 2017.

C. Czado. Analyzing dependent data with vine copulas. *Lecture Notes in Statistics, Springer*, 2019.

C. Czado and E. Brechmann. Generalized linear models with applications. *Unpublished, as of January*, 15, 2021.

R. Diestel. Graph theory. fifth. vol. 173. *Graduate Texts in Mathematics. Paperback edition of [MR3644391]. Springer, Berlin*, 2018.

J. Dissmann, E. C. Brechmann, C. Czado, and D. Kurowicka. Selecting and estimating regular vine copulae and application to financial returns. *Computational Statistics & Data Analysis*, 59:52–69, 2013.

J. F. Dißmann. Statistical inference for regular vines and application. 2010.

C. Dwork. A firm foundation for private data analysis. *Communications of the ACM*, 54 (1):86–95, 2011.

C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

C. Dwork, A. Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

T. Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

W. F. Ferger. The nature and use of the harmonic mean. *Journal of the American Statistical Association*, 26(173):36–40, 1931.

M. Frid-Adar, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan. Synthetic data augmentation using gan for improved liver lesion classification. In *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*, pages 289–293. IEEE, 2018.

J. Friedman, T. Hastie, R. Tibshirani, et al. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

C. Genest and J. Nešlehová. A primer on copulas for count data. *ASTIN Bulletin: The Journal of the IAA*, 37(2):475–515, 2007.

H.-O. Georgii. *Stochastics.* de Gruyter, 2012.

T. Gneiting and A. E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

G. Gudendorf and J. Segers. Extreme-value copulas. In *Copula theory and its applications*, pages 127–145. Springer, 2010.

I. H. Haff, K. Aas, and A. Frigessi. On the simplified pair-copula construction—simply useful or too simplistic? *Journal of Multivariate Analysis*, 101(5):1296–1310, 2010.

C. Han, H. Hayashi, L. Rundo, R. Araki, W. Shimoda, S. Muramatsu, Y. Furukawa, G. Mauri, and H. Nakayama. Gan-based synthetic brain mr image generation. In *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*, pages 734–738. IEEE, 2018.

D. Hand and P. Christen. A note on using the f-measure for evaluating record linkage algorithms. *Statistics and Computing*, 28(3):539–547, 2018.

D. J. Hand and R. J. Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine learning*, 45(2):171–186, 2001.

T. Hastie and J. Qian. Glmnet vignette. *Retrieved June*, 9(2016):1–30, 2014.

D. Heck, J. Knapp, J. Capdevielle, G. Schatz, T. Thouw, et al. Corsika: A monte carlo code to simulate extensive air showers. *Report fzka*, 6019(11), 1998.

R. J. Hyndman and Y. Fan. Sample quantiles in statistical packages. *The American Statistician*, 50(4):361–365, 1996.

H. Joe. Families of m-variate distributions with given margins and m (m-1)/2 bivariate dependence parameters. *Lecture Notes-Monograph Series*, pages 120–141, 1996.

S. Kamthe, S. Assefa, and M. Deisenroth. Copula flows for synthetic data generation. *arXiv preprint arXiv:2101.00598*, 2021.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

S. Kotz and S. Nadarajah. *Multivariate t-distributions and their applications.* Cambridge University Press, 2004.

D. Kurowicka. Vine truncations. In *3rd Vine Copula Workshop, Oslo*, 2009.

D. Kurowicka and R. Cooke. A parameterization of positive definite matrices in terms of partial correlation vines. *Linear Algebra and its Applications*, 372:225–251, 2003.

D. Kurowicka and R. M. Cooke. *Uncertainty analysis with high dimensional dependence modelling.* John Wiley & Sons, 2006.

A. Liaw and M. Wiener. Classification and regression by randomforest. *R News*, 2(3): 18–22, 2002. URL `https://CRAN.R-project.org/doc/Rnews/`.

C. Loader. *Local regression and likelihood.* Springer Science & Business Media, 2006.

T. Nagler. Kernel methods for vine copula estimation. 2014.

T. Nagler. kdecopula: An r package for the kernel estimation of bivariate copula densities. *arXiv preprint arXiv:1603.04229*, 2016.

T. Nagler and T. Vatter. *rvinecopulib: High Performance Algorithms for Vine Copula Modeling*, 2021. URL `https://CRAN.R-project.org/package=rvinecopulib`. R package version 0.5.5.1.1.

T. Nagler, U. Schepsmeier, J. Stoeber, E. C. Brechmann, B. Graeler, and T. Erhardt. *VineCopula: Statistical Inference of Vine Copulas*, 2021. URL `https://CRAN.R-project.org/package=VineCopula`. R package version 2.4.3.

J. A. Nelder and R. W. Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society: Series A (General)*, 135(3):370–384, 1972.

R. B. Nelsen. *An introduction to copulas.* Springer Science & Business Media, 2007.

A. Panagiotelis, C. Czado, and H. Joe. Pair copula constructions for multivariate discrete data. *Journal of the American Statistical Association*, 107(499):1063–1072, 2012.

A. Panagiotelis, C. Czado, H. Joe, and J. Stöber. Model selection for discrete regular vine copulas. *Computational Statistics & Data Analysis*, 106:138–152, 2017.

M. Y. Park and T. Hastie. L1-regularization path algorithm for generalized linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(4):659–677, 2007.

J. Qian, S. Olbrecht, B. Boeckx, H. Vos, D. Laoui, E. Etlioglu, E. Wauters, V. Pomella, S. Verbandt, P. Busschaert, et al. A pan-cancer blueprint of the heterogeneous tumor microenvironment revealed by single-cell profiling. *Cell research*, 30(9):745–762, 2020.

R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2021. URL `https://www.R-project.org/`.

R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2022. URL `https://www.R-project.org/`.

X. Robin, N. Turck, A. Hainard, N. Tiberti, F. Lisacek, J.-C. Sanchez, and M. Müller. proc: an open-source package for r and s+ to analyze and compare roc curves. *BMC bioinformatics*, 12(1):1–8, 2011.

M. Rosenblatt. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, 27(3):832–837, 1956. ISSN 00034851. URL `http://www.jstor.org/stable/2237390`.

N. Schallhorn. D-vine quantile regression for mixed discrete and continuous data with applications to bank stress testing. 2017.

N. Simon, J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for cox's proportional hazards model via coordinate descent. *Journal of Statistical Software*, 39 (5):1–13, 2011. URL `https://www.jstatsoft.org/v39/i05/`.

N. Simon, J. Friedman, and T. Hastie. A blockwise descent algorithm for group-penalized multiresponse and multinomial regression. *arXiv preprint arXiv:1311.6529*, 2013.

M. Sklar. Fonctions de repartition an dimensions et leurs marges. *Publ. inst. statist. univ. Paris*, 8:229–231, 1959.

J. Stöber and C. Czado. Pair copula constructions. In *Simulating Copulas: Stochastic Models, Sampling Algorithms and Applications*, pages 185 – 230. 2nd edition, 2017.

N. Tagasovska, D. Ackerer, and T. Vatter. Copulas as high-dimensional generative models: Vine copula autoencoders. *Advances in neural information processing systems*, 32, 2019.

S. Wagner and D. Wagner. *Comparing clusterings: an overview.* Universität Karlsruhe, Fakultät für Informatik Karlsruhe, 2007.

Z. Wan, Y. Zhang, and H. He. Variational autoencoder based synthetic data generation for imbalanced learning. In *2017 IEEE symposium series on computational intelligence (SSCI)*, pages 1–7. IEEE, 2017.

L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni. Modeling tabular data using conditional gan. *Advances in Neural Information Processing Systems*, 32, 2019.

9 Appendix

Commonly Used Distributions

- **Univariate Normal distribution, Gauss distribution:** The random vector $X \in \mathbb{R}$ is *univariate Gauss distributed* or *univariate normally distributed* with mean $\mu \in \mathbb{R}$ and positive variance $\sigma^2 > 0$, if it has the probability density function:
$$f(x;\mu,\sigma) := \frac{1}{\sqrt{2\pi\sigma^2}} \exp\big\{ - \frac{1}{\sigma^2}(x-\mu)^2 \big\} \,.$$
This is denoted by: $X \sim \mathcal{N}(\mu,\sigma^2)$. For $\mu = 0$ and $\sigma^2 = 1$, i.e. $X \sim \mathcal{N}(0,1)$, we say, that X is *standard normally distributed* and denote its density with $\phi(x)$ and its distribution with $\Phi(x)$. See Georgii (2012).

- **Multivariate Normal distribution, Gauss distribution:** In the same manner the random vector $\boldsymbol{X} \in \mathbb{R}^d$ is *multivariate Gauss distributed* or *multivariate normally distributed* with mean vector $\boldsymbol{\mu} \in \mathbb{R}^d$ and positive definite covariance matrix $\Sigma \in \mathbb{R}^{d\times d}$, if it has the probability density function:
$$f_d(\boldsymbol{x};\boldsymbol{\mu},\Sigma) := \frac{1}{(2\pi)^{d/2}}|\Sigma|^{-1/2} \exp\big\{ - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) \big\} \,.$$
In particular, if it holds for $\boldsymbol{X}$, that for all $i \in [d]$ it is $X_i \sim \mathcal{N}(0,1)$, then $\boldsymbol{X}$ is said to be *multivariate standard normally distributed* and its density is denoted by $\phi(\boldsymbol{x};R)$ and its distribution by $\Phi(\boldsymbol{x};R)$ with $R \in \mathbb{R}^{d\times d}$ the positive definite correlation matrix. See Georgii (2012).

- **Univartiate Student's t distribution:** The random vector $X \in \mathbb{R}$ is *univariate Student's t distributed* with degree of freedom parameter $\nu > 0$, mean $\mu \in \mathbb{R}$ and scale parameter $\sigma^2 > 0$, if it has the probability density function:
$$f_t(x;\nu,\mu,\sigma^2) := \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\pi\nu\sigma^2}\Gamma(\frac{\nu}{2})}\Big\{1 + \Big(\frac{x-\mu}{\sigma}\Big)^2\frac{1}{\nu}\Big\}^{-\frac{\nu+1}{2}} \,.$$
It is denoted by: $X \sim t_\nu(\mu,\sigma^2)$. For $\mu = 0$ and $\sigma^2 = 1$, i.e. $X \sim t_\nu(0,1)$, we say, that X is *standard Student's t distributed* and denote its density function with $t_\nu(x)$ and its distribution with $T_\nu(x)$.

- **Multivartiate Student's t distribution:** The direct generalization of the univartiate Student's t distribution is according to Kotz and Nadarajah (2004) given as the following: The random vector $\boldsymbol{X} \in \mathbb{R}^d$ is *multivartiate Student's t distributed* with degrees of freedom ν, mean vector $\boldsymbol{\mu} \in \mathbb{R}^d$ and correlation matrix $R \in \mathbb{R}^{d\times d}$, if it has the probability density function:
$$f_t(\boldsymbol{x};\nu,\boldsymbol{\mu},R) := \frac{\Gamma(\frac{\nu+d}{2})}{(\pi\nu)^{p/2}\,\Gamma(\frac{\nu}{2})\,|R|^{1/2}}\Big\{1 + \frac{1}{\nu}(\boldsymbol{x}-\boldsymbol{\mu})^T R^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\Big\}^{-\frac{\nu+d}{2}} \,.$$
It is denoted by: $\boldsymbol{X} \sim t_\nu(\boldsymbol{\mu},R)$. For if $\boldsymbol{\mu} = 0$ the distribution is said to be *central* and we denote its density function with $t(\boldsymbol{x};\nu,R)$ and its distribution function with $T(\boldsymbol{x};\nu,R)$. Otherwise it is called *non-central*. The degrees of freedom parameter ν is also referred to as the *shape parameter*.