Journal of Intelligent Information Systems manuscript No.
(will be inserted by the editor)

An Approach to Structure Determination and Estimation of Hierarchical Archimedean Copulas and Its Application to Bayesian Classification

Jan Górecki · Marius Hofert · Martin Holeňa

the date of receipt and acceptance should be inserted later

Abstract Copulas are distribution functions with standard uniform univariate marginals. Copulas are widely used for studying dependence among continuously distributed random variables, with applications in finance and quantitative risk management; see, e.g., the pricing of collateralized debt obligations [27]. The ability to model complex dependence structures among variables has recently become increasingly popular in the realm of statistics, one example being data mining (e.g., cluster analysis, evolutionary algorithms or classification).

The present work considers an estimator for both the structure and the parameters of hierarchical Archimedean copulas. Such copulas have recently become popular alternatives to the widely used Gaussian copulas. The proposed estimator is based on a pairwise inversion of Kendall's tau estimator recently considered in the literature but can be based on other estimators as well, such as likelihood-based. A simple algorithm implementing the proposed estimator is provided. Its performance is investigated in several experiments including a comparison to other available estimators. The results show that the proposed estimator can be a suitable alternative in the terms of goodness-of-fit and computational efficiency. Additionally, an application of the estimator to copula-based Bayesian classification is presented. A set of new Archimedean and hierarchical Archimedean copula-based Bayesian classifiers is compared with other commonly known classifiers in terms

The work of Jan Górecki was funded by the project SGS/21/2014 - Advanced methods for knowledge discovery from data and their application in expert systems, Czech Republic. The work of Martin Holeňa was funded by the Czech Science Foundation (GA ČR) grant 13-17187S. This article has been originally published in Journal of Intelligent Information Systems (2015), DOI: 10.1007/s10844-014-0350-3.

Jan Górecki
Department of Informatics, SBA in Karviná, Silesian University in Opava, Karviná, Czech Republic, E-mail: gorecki@opf.slu.cz

Marius Hofert
Department of Statistics and Actuarial Science, University of Waterloo, 200 University Avenue West, Waterloo, ON, Canada, E-mail: marius.hofert@uwaterloo.ca

Martin Holeňa
Institute of Computer Science, Academy of Sciences of the Czech Republic, Praha, Czech Republic, E-mail: martin@cs.cas.cz

of accuracy on several well-known datasets. The results show that the hierarchical Archimedean copula-based Bayesian classifiers are, despite their limited applicability for high-dimensional data due to expensive time consumption, similar to highly-accurate classifiers like support vector machines or ensemble methods on low-dimensional data in terms of accuracy while keeping the produced models rather comprehensible.

Keywords copula · hierarchical Archimedean copula · copula estimation · structure determination · Kendall's tau · Bayesian classification

1 Introduction

Studying relationships among random variables is a crucial task in the field of knowledge discovery and data mining (KDDM). Having a dataset collected, the relationships among the observed variables can be studied by means of an appropriate measure of stochastic dependence. Under the assumption that the marginal distributions of the variables are continuous, Sklar's Theorem [51] can be used to decompose the joint multivariate distribution in two parts, the univariate marginal distributions and the unique dependence structure, i.e., the copula of the joint distribution. Thus, studying dependence among continuously distributed random variables can be restricted without loss of generality to studying the underlying copula.

Despite the fact that a large part of the success of copulas is attributed to finance, copulas are increasingly adopted also in KDDM, where their ability to capture complex dependence structures among variables is used. Applications of copulas can be found in water-resources and hydro-climatic analysis [13,30,31,35, 38], gene analysis [37,56], cluster analysis [11,32,46] or in evolutionary algorithms, in particular estimation of distribution algorithms [17,54]. For an illustrative example, we refer to [30], which describes an application of copulas to detecting weather anomalies in a climate change dataset.

For certain types of applications, hierarchical Archimedean copulas (HACs) are a frequently used alternative to Gaussian copulas due to several desirable properties, e.g., HACs are not restricted to radial symmetry; HACs are expressible in a closed form; they are able to model asymmetric distributions with tail dependence; and HACs are able to model complex relationships while keeping the number of parameters comparably small; see [23,27]. The last point is important from a data mining point of view because models with a small number of parameters are more easily understandable. Denoting the data dimension by d, on the one hand, if using Gaussian copulas, the number of parameters grows quadratically in d and the obtained models can quickly become challenging from a computational point of view. On the other hand, if using Archimedean copulas (ACs), the obtained models contain only one parameter (provided an AC is based on a one-parametric generator), which is rarely feasible in real-world applications. In this context, HACs are often a good trade-off between these two extremes and provide relatively simple and flexible dependency models.

Despite the popularity of HACs, feasible techniques for their parameter and structure estimation are addressed only in few papers. Most of them assume a given hierarchical structure, which is motivated by applications in economics, e.g., [48, 49]. On the contrary, in [50], only structure determination of a HAC is addressed. We are aware of only one paper [43] that addresses both structure determination and parameter estimation via a multi-stage procedure. That paper mainly focuses on the estimation of the parameters using the maximum-likelihood (ML) technique and briefly mentions the inversion of Kendall's tau as an alternative. For structure determination, six approaches are presented. Two of them are based on the inversion of Kendall's tau, one on the Chen test statistic [8] and the remaining three approaches on the ML technique. All but one approach lead to biased estimators, which can be seen from the results of the reported study. The unbiased estimator, denoted by θ_{RML}, which shows the best goodness-of-fit (measured by Kullback-Leibler divergence) in the study, is simply the maximum likelihood estimator based on initial values computed from one of the biased estimators. Due to this construction, θ_{RML} often does not approximate the true parameters well when the structure determined by the biased estimator is not the true structure. The number of such cases rapidly increases in large dimensions, as we show later in Section 4.

In the present work, we propose a new estimator for both the structure and the parameters of HACs. On the one hand, this estimator is also a multi-stage procedure where the structure and the parameters are estimated in a bottom-up manner. On the other hand, it is based on the fact that a HAC can be uniquely recovered from all its bivariate margins and thus allows to estimate the copula parameters just from the parameters of the bivariate marginal copulas. Assuming the true copula is a HAC, our estimator approximates the true copula closer (measured by a selected goodness-of-fit statistic) than the previously mentioned methods. Moreover, the ratio of structures properly determined using our estimator is higher compared with the estimators mentioned above. Finally, avoiding a time-consuming computation of initial values, we also gain computational efficiency. The experiments based on simulated data in Section 4 show that our approach outperforms the above-mentioned methods with respect to goodness-of-fit, the properly determined structures ratio and also the consumed run-time.

In addition, we consider Bayesian classifiers that are based on Gaussian copulas, ACs and HACs. When fitting those classifiers, efficient estimation methods for a given copula class are needed. In the Gaussian and Archimedean case, such estimation methods are known, whereas for HACs, we can now apply our proposed estimator. We compare it with other copula-based Bayesian classifiers, as well as with other types of commonly used classifiers.

The paper is structured as follows. The following section summarizes some needed theoretical concepts concerning ACs and HACs. Section 3 presents the new estimation approach for HACs, and Section 4 describes the experiments based on simulated data. Section 5 presents a copula-based approach to Bayesian classification and includes an experimental comparison of several classifiers based on real-world datasets. Section 6 concludes this paper.

2 Preliminaries

2.1 Copulas

Definition 1 For every $d \geq 2$, a *d-dimensional copula* (shortly, *d-copula*) is a d-variate distribution function on $\mathbb{I}^d$ ($\mathbb{I} = [0, 1]$), whose univariate margins are uniformly distributed on $\mathbb{I}$.

Copulas establish a connection between joint distribution functions (d.f.s) and their univariate margins, which is well-know due to Sklar's Theorem.

Theorem 1 ***(Sklar's Theorem (1959))*** *[51] Let H be a d-variate d.f. with univariate margins $F_1, ..., F_d$. Let A_j denote the range of F_j, $A_j := F_j(\overline{\mathbb{R}}), j = 1, ..., d, \overline{\mathbb{R}} := \mathbb{R} \cup \{-\infty, +\infty\}$. Then there exists a copula C such for all $(x_1, ..., x_d) \in \overline{\mathbb{R}}^d$,*

$$H(x_1, ..., x_d) = C(F_1(x_1), ..., F_d(x_d)). \tag{1}$$

Such a C is uniquely determined on $A_1 \times ... \times A_d$. Conversely, if $F_1, ..., F_d$ are univariate d.f.s, and if C is any d-copula, then the function $H : \overline{\mathbb{R}}^d \to \mathbb{I}$ defined by (1) is a d-dimensional distribution function with margins $F_1, ..., F_d$.

Through Sklar's Theorem, one can derive for any d-variate d.f. with continuous margins its unique copula C using (1). C is given by $C(u_1, ..., u_d) = H(F_1^-(u_1), ..., F_d^-(u_d))$, where $F_i^-, i \in \{1, ..., d\}$, denotes the pseudo-inverse of F_i given by $F_i^-(s) = \inf\{t \mid F_i(t) \geq s\}, s \in \mathbb{I}$. Implicit copulas are derived in this way from popular joint d.f.s, e.g., the popular class of Gaussian copulas is derived from multivariate normal distributions. However, using this process often results in copulas which do not have a closed form, which can be a drawback for certain applications, e.g., if explicit probabilities and thus copula values have to be computed.

2.2 Archimedean Copulas

Due to their explicit construction, Archimedean copulas (ACs) are typically expressible in closed form. To construct ACs in arbitrary dimensions, we need the notion of an *Archimedean generator* and of *complete monotonicity.*

Definition 2 An *Archimedean generator* (shortly, *generator*) is a continuous, nonincreasing function $\psi : [0, \infty] \to [0, 1]$, which satisfies $\psi(0) = 1, \psi(\infty) = \lim_{t\to\infty} \psi(t) = 0$ and which is strictly decreasing on $[0, \inf\{t \mid \psi(t) = 0\}]$. We denote the set of all generators by Ψ. If ψ satisfies $(-1)^k f^{(k)}(t) \geq 0$, for all $k \in \mathbb{N}, t \in [0, \infty)$, ψ is called completely monotone. We denote the set of all completely monotone generators by Ψ_∞.

Definition 3 Any d-copula C is called an *Archimedean copula* (we denote it d-AC) based on a generator $\psi \in \Psi$, if it admits the form

$$C(\mathbf{u}) := C(\mathbf{u}; \psi) := \psi(\psi^{-1}(u_1) + ... + \psi^{-1}(u_d)), \mathbf{u} \in \mathbb{I}^d, \tag{2}$$

where $\psi^{-1} : [0, 1] \to [0, \infty]$ is defined by $\psi^{-1}(s) = \inf\{t \mid \psi(t) = s\}, s \in \mathbb{I}$.

A condition sufficient for C to be indeed a proper copula is $\psi \in \Psi_\infty$; see [40].

Table 1 Completely monotone (c.m.) one-parametric Archimedean families from [42, p. 116] considered in this paper. The table contains the corresponding families, the parameter ranges and the sufficient nesting condition for two generators from the same family (see Section 2.3 in [23]). The sufficient nesting condition involves generators ψ_1 and ψ_2 from the same family with parameters equal to θ_1 and θ_2, respectively.

Family	θ	$\psi(t)$	$(\psi_1^{-1} \circ \psi_2)'(t)$ c.m.
Clayton (C)	$(0, \infty)$	$(1+t)^{-1/\theta}$	$\theta_1 \leq \theta_2$
Frank (F)	$(0, \infty)$	$-\log(1-(1-e^{-\theta})\exp(-t))/\theta$	$\theta_1 \leq \theta_2$
Gumbel (G)	$[1, \infty)$	$\exp(-t^{1/\theta})$	$\theta_1 \leq \theta_2$

As we can see from Definition 3, if a random vector $\mathbf{U}$ is distributed according to some AC, all its k-dimensional marginal copulas are equal. Thus, e.g., the dependence among all pairs of components is identical. This symmetry of ACs is often considered to be a rather strong restriction, especially in high dimensions; see [26] for a discussion and possible applications.

To obtain an explicit form of an AC, we need ψ and ψ^{-1} to be explicit; many such generators can be found, e.g., in [42]. In this paper, we use the three well-known parametric generators of the Clayton, Frank and Gumbel families; see Table 1. We selected these three families of generators because of two reasons. The first reason relates to flexibility of these families to model tail dependence in pairs of random variables, as this is a copula property. The Clayton family allows lower tail dependence in a pair (being upper tail independent), the Gumbel family allows oppositely upper tail dependence in a pair (being lower tail independent), and models from the Frank family are both lower and upper independent, similarly to Gaussian copulas; see [22, p. 43]. The second reason is that this choice allows for a comparison of our results with the results in [43] and [28]. More precisely, in [43], HAC estimation experiments involving HACs based on Clayton and Gumbel generators are reported; these experiments relate to our experiments described in Section 4. In [28], a visual representation of a HAC structure involving Frank generators obtained from the Iris dataset is presented; this tree-like representation relates to our dendrogram-like representation described in Section 5.

2.3 Hierarchical Archimedean Copulas

To allow for asymmetries, one may consider the class of HACs (also called *nested Archimedean copulas*), recursively defined as follows.

Definition 4 [23] A d-dimensional copula C is called a *hierarchical Archimedean copula* if it is either an AC, or if it is obtained from an AC through replacing some of its arguments with other hierarchical Archimedean copulas. In particular, if C is given recursively by (2) for $d = 2$ and

$$C(\mathbf{u}; \psi_1, ..., \psi_{d-1}) = \psi_1(\psi_1^{-1}(u_1) + \psi_1^{-1}(C(u_2, ..., u_d; \psi_2, ..., \psi_{d-1}))), \mathbf{u} \in \mathbb{I}^d, \quad (3)$$

for $d \geq 3$, C is called *fully-nested hierarchical Archimedean copula (FHAC)*[1] with $d-1$ *nesting levels*. Otherwise, C is a *partially-nested hierarchical Archimedean copula (PHAC)*[2].

[1] also called *fully-nested Archimedean copula*

[2] also called *partially-nested Archimedean copula*

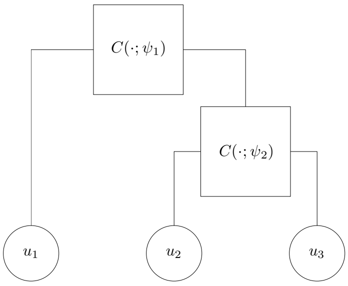

Fig. 1 Tree-like structure of a 3-FNAC.

Remark 1 We denote a d-dimensional HAC as d-HAC, and analogously d-FHAC and d-PHAC.

From the definition, we can see that ACs are special cases of HACs. The most simple proper 3-HAC is a two-level FHAC given by

$$\begin{aligned} C(\mathbf{u};\psi_1,\psi_2) &= C(u_1, C(u_2,u_3;\psi_2);\psi_1) \\ &= \psi_1(\psi_1^{-1}(u_1) + \psi_1^{-1}(\psi_2(\psi_2^{-1}(u_2) + \psi_2^{-1}(u_3)))),\ \mathbf{u} \in \mathbb{I}^3. \end{aligned} \quad (4)$$

and its structure can be represented via a tree-like graph; see Figure 1.

Assume that a random vector (U_1,U_2,U_3) is distributed according to the 3-FHAC given by (4), i.e., $(U_1,U_2,U_3) \sim C(\mathbf{u};\psi_1,\psi_2)$. Then $C(u_1,u_2,1;\psi_1,\psi_2) = C(u_1,u_2;\psi_1)$, $C(u_1,1,u_3;\psi_1,\psi_2) = C(u_1,u_3;\psi_1)$ and $C(1,u_2,u_3;\psi_1,\psi_2) = C(u_2,u_3;\psi_2)$ for all $u_1,u_2,u_3 \in \mathbb{I}$. This means that this 3-FHAC involves two different bivariate marginal copulas, the 2-AC based on ψ_1, which is the distribution of the pairs (U_1,U_2) and (U_1,U_3), and the 2-AC based on ψ_2, which is the distribution of the pair (U_2,U_3). The asymmetry of this 3-HAC is a motivating example for nesting of ACs. The theoretical soundness of nesting is addressed in Theorem 2.

As in the case of ACs, we can ask for sufficient conditions for the function C given by (3) to be a proper copula. An answer to this question is provided by the following theorem. Note that another important result concerning stochastic representation of HACs is provided by Theorem 3.2 in [24].

Theorem 2 ***(McNeil (2008))*** *[39] If $\psi_j \in \Psi_\infty, j \in \{1,...,d-1\}$ such that $\psi_k^{-1} \circ \psi_{k+1}$ have completely monotone derivatives for all $k \in \{1,...,d-2\}$, then $C(\boldsymbol{u};\psi_1,...,\psi_{d-1})$, $\boldsymbol{u} \in \mathbb{I}^d$, given by (3) is a copula.*

Theorem 2 is stated only for fully-nested HACs, but it can be easily translated to partially-nested HACs. The condition for $(\psi_k^{-1} \circ \psi_{k+1})'$ to be completely monotone is often called a *sufficient nesting condition.*

Any d-HAC structure can be expressed as a tree with $k \leq d-1$ non-leaf nodes (shortly, nodes), which correspond to the generators $\psi_1,...,\psi_k$, and d leaves, which correspond to the variables $u_1,...,u_d$. If the structure corresponds to a binary tree, then $k = d-1$, otherwise $k < d-1$. For the sake of simplicity, we assume only *binary-structured* HACs in the following. A binary-structured HAC is a HAC with the structure which corresponds to a binary tree and for each parent-child pair of generators (ψ_i,ψ_j) in the structure holds that $\psi_i \neq \psi_j$.

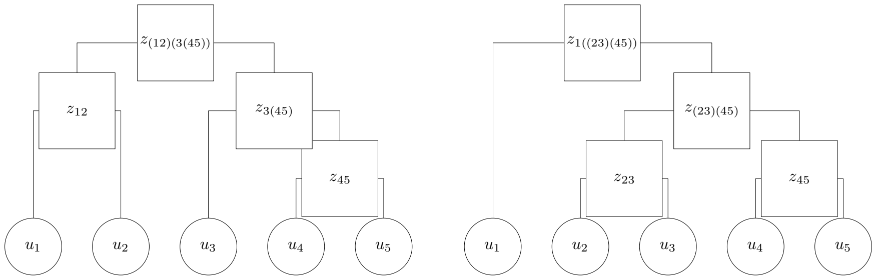

Fig. 2 Two 5-PHAC structures denoted by ((12)(3(45))) and (1((23)(45))) are depicted on the left and on the right side, respectively.

Similar to any 2-AC being determined by its corresponding generator, we identify each node in a HAC structure with one generator. Thus we always have the nodes $\psi_1, ..., \psi_{d-1}$. For a node ψ, denote by $\mathcal{D}(\psi)$ the set of all descendant non-leaf nodes of ψ, $\mathcal{D}_l$ the set of all descendant leaves of ψ, $\mathcal{A}(\psi)$ the set of all ancestor nodes of ψ, $\mathcal{H}_l(\psi)$ the left child of ψ and $\mathcal{H}_r(\psi)$ the right child of ψ. Next, let z be a non-leaf node or a leaf, and, assuming z is not the root of the structure, denote by $\mathcal{P}(z)$ the parent node of z.

For simplicity, a d-HAC structure s is denoted by a sequence of reordered indices $\{1, ..., d\}$ using parentheses to mark the variables with the same parent node. For example, the structure of the copula given by (4) is denoted as (1(23)). The inner pair of parentheses corresponds to the variables u_2, u_3, for which $\mathcal{P}(u_2) = \mathcal{P}(u_3) = \psi_2$. As u_2, u_3 are connected through their parent, we can introduce a new variable denoted by z_{23}, which represents the variables u_2, u_3 and is defined by $z_{23} = C(u_2, u_3; \psi_2)$. Then (4) translates to $\psi_1(\psi_1^{-1}(u_1) + \psi_1^{-1}(z_{23})) = C(u_1, z_{23}; \psi_1)$, and thus the outer pair of parenthesis in the notation of the structure corresponds to the variables u_1, z_{23}, for which $\mathcal{P}(u_1) = \mathcal{P}(z_{23}) = \psi_1$. The structure of the 4-FHAC according to Definition 4 is therefore $s =$ (1(2(34))), for the 5-FHAC, $s =$ (1(2(3(45)))), etc. Analogously, for PHACs, $s =$ ((12)(3(45))) and $s =$ (1((23)(45))) denote the structures depicted on the left-hand and on the right-hand side in Figure 2, respectively.

When using HACs in applications, there exist many possible structures, for example for $d = 10$, more than 280 millions structures exist (including also non-binary ones) and each 10-HAC can incorporate up to 9 parameters (using only one-parametric generators, possibly from different families). On the one hand, choosing the model (structure and parameters) that fits the data best is a much more complex relative to the case when using ACs which have just one structure. On the other hand, this complexity is compensated by a substantially higher flexibility of obtained models. Due to the asymmetry in HAC-based models (different dependencies in pairs of variables are allowed), these models fit most data better than AC models, which is illustrated by the experimental results presented below in Section 5. There, different copula-based Bayesian classifiers are evaluated in terms of accuracy and, due to the flexibility of HAC models, the Bayesian classifiers based on HACs mostly score higher than the Bayesian classifiers based on ACs .

To derive an explicit parametric form a d-HAC C, we need explicit parametric forms for the generators $\psi_1, ..., \psi_{d-1}$, which involve the parameters $\theta_1, ..., \theta_{d-1}$, respectively, and its structure s. Due to this, the copula C is also denoted by $C_{\psi,\theta;s}(u_1, ..., u_d)$ in what follows. For example, the 3-HAC given by (4) can be denoted by $C_{\psi_1,\psi_2,\theta_1,\theta_2;(1(23))}$ and its parametric form, assuming, e.g., both of its generators ψ_1, ψ_2 to be Clayton generators, is given by

$$C_{\psi_1,\psi_2,\theta_1,\theta_2;(1(23))}(u_1, u_2, u_3) = \left(\left(\left({u_2}^{-\theta_2} + {u_3}^{-\theta_2} - 1 \right)^{-\frac{1}{\theta_2}} \right)^{-\theta_1} + {u_1}^{-\theta_1} - 1 \right)^{-\frac{1}{\theta_1}}. \tag{5}$$

2.4 Kendall's tau and an extension to more than two dimensions

Let (X_1, Y_1) and (X_2, Y_2) be independent copies of a random vector (X, Y). Then the *population version of Kendall's tau* is defined as the probability of concordance minus the probability of discordance, i.e.,

$$\tau = \tau_{XY} = \mathbb{P}((X_1 - X_2)(Y_1 - Y_2) > 0) - \mathbb{P}((X_1 - X_2)(Y_1 - Y_2) < 0). \tag{6}$$

It can be shown, see, e.g., [13], that

$$\tau(C) = 4 \int_{\mathbb{I}^2} C(u_1, u_2) dC(u_1, u_2) - 1, \tag{7}$$

so τ depends only on the copula of (X, Y). If C is a 2-AC based on a twice continuously differentiable generator ψ with $\psi(t) > 0$ for all $t \in [0, \infty)$, Kendall's tau can be represented as [29, p. 91], [42, p. 163]

$$\tau(\psi) = \tau(C(\cdot; \psi)) = 1 - 4 \int_0^\infty t(\psi'(t))^2 dt = 1 - 4 \int_0^1 \frac{\psi^{-1}(t)}{(\psi^{-1})'(t)} dt. \tag{8}$$

Hence, (8) states a relationship between θ and τ, which can often be expressed in closed form. For example, if C is a Clayton copula, see Table 1, we get $\tau = \theta/(\theta+2)$ (the relationship between θ and τ for other generators can be found, e.g., in [22]). The inversion of this relationship establishes a method-of-moments-like estimator of the parameter θ given by $\hat{\theta}_n = \tau^{-1}(\tau_n)$, based on the empirical version τ_n of τ, given by

$$\tau_n = \frac{4}{n(n-1)} \left(\sum_{i=1,j=1}^{n} \mathbf{1}_{\{(u_{i1} - u_{j1})(u_{i2} - u_{j2}) > 0\}} \right) - 1, \tag{9}$$

where $(u_{\bullet 1}, u_{\bullet 2})$ denotes a realization of n independent and identically distributed (i.i.d) copies of $(U_1, U_2) \sim C$; see [16]. Since we do not observe realizations from C directly, note that τ can be computed based on the realizations of (X, Y). If $\tau(\hat{\theta}_n) = \tau_n$ has no solution, this estimation method does not lead to an estimator. Unless there is an explicit form for τ^{-1}, $\hat{\theta}_n$ is computed by numerical root finding [26].

This estimation method can also be generalized to ACs when $d > 2$, see [4, 26, 34, 49]. One of the methods proposed in [4, 49] uses a sample version of the

Kendall correlation matrix. Denote by $(\tau_{ij}) = (\tau_{X_i,X_j})_{ij}$ the population version of the Kendall correlation matrix for continuous random variables $X_1, ..., X_d$. Note that $(\tau_{X_i,X_j})_{ij} = (\tau_{U_i,U_j})_{ij}$, where $F_1(X_1) = U_1, ..., F_d(X_d) = U_d$. Similarly, denote the sample version of Kendall correlation matrix by (τ^n_{ij}), where τ^n_{ij} denotes the sample version of Kendall's tau between the i-th and j-th data column. Then θ is estimated by

$$\hat{\theta}_n = \tau^{-1}\left(\binom{d}{2}^{-1} \sum_{1\leq i\leq j\leq d} \tau^n_{ij}\right). \tag{10}$$

As can be seen, the parameter is chosen such that the value of Kendall's tau equals the average over all pairwise sample versions of Kendall's tau. Properties of this estimator are not known and also not easy to derive since the average is taken over dependent data columns [33]. However, simulations conducted in [26] suggest consistency of this estimator. Moreover, $\binom{d}{2}^{-1} \sum_{1\leq i\leq j\leq d} \tau^n_{ij}$ is an unbiased estimator of $\tau(\theta)$. This is an important property and we transfer it later to an estimator that we use for the structure determination which we base on appropriately selected pairwise sample versions of Kendall's tau.

For applying this generalized estimation approach to HACs, we define a generalization of τ for m (possibly > 2) random variables (r.v.s) based on the following notation. Let $I, J \subset \{1, ..., d\},\ I \neq \emptyset,\ J \neq \emptyset,\ (U_1, ..., U_d) \sim C$ and C be a d-HAC. Denote a set of pairs of r.v.s by $\mathbf{U}_{IJ} = \{(U_i, U_j)|(i, j) \in I \times J\}$ and a set of pairs of data columns by $\mathbf{u}_{IJ} = \{(u_{\bullet i}, u_{\bullet j})|(i, j) \in I \times J\}$, where $u_{\bullet i}, u_{\bullet j}$denotes realizations of (U_i, U_j).

Definition 5 Any function $g : \mathbb{I}^k \to \mathbb{I}, k \in \mathbb{N}$, satisfying 1) $g(u, ..., u) = u$ for all $u \in \mathbb{I}$ and 2) $g(u_{p_1}, ..., u_{p_k}) = g(u_1, ..., u_k)$ for all $u_1, ..., u_k \in \mathbb{I}$ and all permutations $(p_1, ..., p_k)$ of $(1, ..., k)$ is called an $\mathbb{I}$-*aggregation* function.
Examples of $\mathbb{I}$-aggregation functions are the functions max, min or mean restricted to $\mathbb{I}^k$.

Definition 6 Let g be an $\mathbb{I}$-aggregation function. Then define a *g-aggregated Kendall's tau* (or simply an *aggregated Kendall's tau*) τ^g as

$$\tau^g(\mathbf{U}_{IJ}) = \begin{cases} \tau(U_i, U_j), & \text{if } I = \{i\}, J = \{j\}, \\ g(\tau(U_{i_1}, U_{j_1}), \tau(U_{i_1}, U_{j_2}), ..., \tau(U_{i_l}, U_{j_q})), & \text{otherwise,} \end{cases} \tag{11}$$

where $I = \{i_1, ..., i_l\}, J = \{j_1, ..., j_q\}$ are non-empty disjoint subsets of $\{1, ..., d\}$.

Note that the sets I and J are assumed to be disjoint because we are interested only in the values of Kendall's tau for bivariate margins of a HAC. For example, if $I = \{1, 2\}$ and $J = \{2, 3\}$, then $\tau^g(\mathbf{U}_{IJ})$ would involve $\tau(U_2, U_2)$, which is not related to any bivariate margin of a HAC.

As the aggregated τ^g depends only on the pairwise τ and the aggregation function g, we can easily derive its empirical version τ^g_n by substituting τ in τ^g by its empirical version τ_n given by (9). Analogously to the case of ACs, the parameter can then be estimated as $\hat{\theta}_n = \tau^{-1}(\tau^g_n)$. This is further explained in Remark 3 of Section 3.1.

2.5 Goodness-of-fit tests

Assume i.i.d. random vectors $\mathbf{X}_i = (X_{i1}, ..., X_{id}),\ i \in \{1, ..., n\}$, distributed according to a joint distribution function H with continuous margins $F_j,\ j \in \{1, ..., d\}$, and the binary-structured HAC C generated by one-parametric generators $\psi_1, ..., \psi_{d-1}$. All generators $\psi_1, ..., \psi_{d-1}$ are assumed to belong to a one-parametric family of generators (e.g., to one of the families listed in Table 1) and their parameters are denoted by $\theta_1, ..., \theta_{d-1}$.

Once we have the parameters estimated, we can ask how well our fitted model fits the data. This can be done using methods known as *goodness-of-fit tests* (GoF tests). In the following, we recall three GoF tests based on statistics that are analogues to Cramér-von Mises statistics [10]. A large value of such statistics leads to the rejection of $H_0 : C \in \mathcal{C}_0$, where $\mathcal{C}_0 = \{C_\theta : \theta \in \mathcal{O}\}$ and $\mathcal{O}$ is an open subset of $\mathbb{R}^p,\ p \geq 1$. Thus for measuring the fitting quality of copula models, we can, informally, assess copula models with lower value of such statistics as "better".

Now consider that, if the margins $F_j,\ j \in \{1, ..., d\}$, are known, $U_{ij} = F_j(X_{ij}),\ i \in \{1, ..., n\}, j \in \{1, ..., d\}$, is a random sample from C. In practice, the margins are typically unknown and must be estimated parametrically or non-parametrically. In the following, we will work under unknown margins and thus we consider the *pseudo-observations*

$$U_{ij} = \frac{n}{n+1}\hat{F}_{n,j}(X_{ij}) = \frac{R_{ij}}{n+1} \tag{12}$$

where $\hat{F}_{n,j}$ denotes the *empirical distribution function* corresponding to the jth margin and R_{ij} denotes the *rank* of X_{ij} among $X_{1j}, ..., X_{nj}$. The information contained in pseudo-observations is conveniently summarized by the associated empirical distribution given by

$$C_n(\mathbf{u}) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{\{U_{i1} \leq u_1, ..., U_{id} \leq u_d\}}, \tag{13}$$

where $\mathbf{u} = (u_1, ..., u_d) \in \mathbb{I}^d$. This distribution is usually called "empirical copula", though it is not a copula except asymptotically [15].

The first GoF test is based on the empirical process

$$\mathbb{C}_n = \sqrt{n}(C_n - C_{\theta_n}), \tag{14}$$

and uses a rank-based version of the Cramér-von Mises statistics

$$S_n = \int_{\mathbb{I}^d} \mathbb{C}_n(\mathbf{u})^2 \mathrm{d}\mathbb{C}_n(\mathbf{u}) = \sum_{i=1}^{n} (C_n(\mathbf{u}_i) - C_{\theta_n}(\mathbf{u}_i))^2. \tag{15}$$

Large values of this statistic lead to the rejection of $H_0 : C \in \mathcal{C}_0$. It is shown in [14] that the test is consistent, i.e., if $C \notin \mathcal{C}_0$, then H_0 is rejected with probability approaching 1 as $n \to \infty$. Appropriate p-values can be obtained via specially adapted Monte Carlo methods described in [15].

The second GoF test, proposed in [13], uses a probability integral transformation of the data, the so-called Kendall's transform

$$\mathbf{X} \mapsto V = H(\mathbf{X}) = C(U_1, ..., U_d), \tag{16}$$

where $(U_1, ..., U_d) \sim C$; see [15]. Let K denote the univariate d.f. of V and $\mathbf{U}_1, ..., \mathbf{U}_n$ the pseudo-observations $\mathbf{U}_i = (\frac{R_{i1}}{n+1}, ..., \frac{R_{id}}{n+1}),\ i \in \{1, .., n\}$. Then K can be estimated nonparametrically by the empirical distribution function of a rescaled version of the pseudo-observations $V_1 = C_n(\mathbf{U}_1), ..., V_n = C_n(\mathbf{U}_n)$ given by

$$K_n(v) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{\{V_i \leq v\}},\ v \in \mathbb{I}, \tag{17}$$

which is a consistent estimator of the underlying distribution K. Under H_0, $\mathbf{U} = (U_1, ..., U_d)$ is distributed as C_θ for some $\theta \in \mathcal{O}$ and hence $C_\theta(\mathbf{U}) \sim K_\theta$. One can then test

$$H_0' : K \in \mathcal{K}_0 = \{K_\theta : \theta \in \mathcal{O}\} \tag{18}$$

based on $\mathbb{K}_n = \sqrt{n}(K_n - K_{\theta_n})$, where K_{θ_n} denotes the distribution function of $C_{\theta_n}(\mathbf{U})$. Generally, because $H_0 \subset H_0'$ the nonrejection of H_0' does not entail the nonrejection of H_0 and consequently, the consistency of the above tests using (14) does not imply the consistency of the tests using $\mathbb{K}_n = \sqrt{n}(K_n - K_{\theta_n})$. But, in the case of bivariate ACs, H_0' and H_0 are equivalent; see [15]. As we are mainly interested in 2-ACs as building blocks of HACs, this test is thus convenient for our purposes. The specific statistic considered in [13] is a rank-based analogue of the Cramér-von Mises statistic

$$S_n^{(K)} = \int_{\mathbb{I}} \mathbb{K}_n(v)^2 \mathrm{d}K_{\theta_n}. \tag{19}$$

This statistic can be easily computed as follows [13]:

$$\begin{aligned} S_n^{(K)} = \ & \frac{n}{3} + n\sum_{j=1}^{n-1} K_n^2\left(\frac{j}{n}\right)\left\{K_{\theta_n}\left(\frac{j+1}{n}\right) - K_{\theta_n}\left(\frac{j}{n}\right)\right\} \\ & - n\sum_{j=1}^{n-1} K_n\left(\frac{j}{n}\right)\left\{K_{\theta_n}^2\left(\frac{j+1}{n}\right) - K_{\theta_n}^2\left(\frac{j}{n}\right)\right\}. \end{aligned} \tag{20}$$

The third GoF test (proposed in [15]) is based on another probability integral transform - namely on the *Rosenblatt's transform*, which is a mapping $\mathcal{R}_\theta : (0,1)^d \to (0,1)^d$ such that $e_1 = u_1$ and for each $j = 2, ..., d$,

$$e_j = \frac{\partial^{j-1} C_\theta(u_1, ..., u_j, 1, ..., 1)}{\partial u_1 ... u_{j-1}} \Bigg/ \frac{\partial^{j-1} C_\theta(u_1, ..., u_{j-1}, 1, ..., 1)}{\partial u_1 ... u_{j-1}}. \tag{21}$$

A crucial property of Rosenblatt's transform is that $\mathbf{U} \sim C_\theta$ if and only if the distribution of $\mathcal{R}_\theta(C_\theta)$ is the d-variate *independence copula* $C_\Pi(\mathbf{u}) = u_1 u_2 ... u_d$; see, e.g., [15]. Thus for all $\theta \in \mathcal{O}$, $H_0 : C_\theta \in \mathcal{C}_0$ is equivalent to $H_0'' : \mathcal{R}_\theta(\mathbf{U}) \sim C_\Pi$.

To test H_0'', we can therefore use the fact that under H_0, the transformed pseudo-observations $\mathbf{E}_1 = \mathcal{R}_\theta(\mathbf{U}_1), ..., \mathbf{E}_n = \mathcal{R}_\theta(\mathbf{U}_n)$, can be interpreted as a sample from the independence copula C_Π. Defining the empirical distribution function on $\mathbf{E}_1, ..., \mathbf{E}_n$ as

$$D_n(\mathbf{u}) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{\{\mathbf{E}_i \leq \mathbf{u}\}},\ \mathbf{u} \in \mathbb{I}^d, \tag{22}$$

it should be close to C_{Π} under H_0. Cramér-von Mises statistics based on Rosenblatt's transformation are given by

$$S_n^{(C)} = n \int_{\mathbb{I}^d} (D_n(\mathbf{u}) - C_{\Pi}(\mathbf{u}))^2 \mathrm{d}D_n(\mathbf{u}) = \sum_{i=1}^{n} \{D_n(\mathbf{E}_i) - C_{\Pi}(\mathbf{E}_i)\}^2; \tag{23}$$

see [15]. All three test statistics performed well in a large scale simulation study conducted at [15] in the bivariate case. We choose them as good candidates for our purpose of goodness-of-fit testing.

We now introduce a *g-aggregated statistic* that will be used for the GoF assessment of d-HAC estimates in Section 4.

Definition 7 Let C be a d-HAC, g be an $\mathbb{I}$-aggregation function and ${}_2S_n((u_{\bullet i}, u_{\bullet j}), C_2(\cdot;\psi))$ be the statistic corresponding to a GoF test, e.g., $S_n, S_n^{(K)}$ or $S_n^{(C)}$, for a bivariate copula $C_2(u_1, u_2; \psi)$ and a pair of data columns $(u_{\bullet i}, u_{\bullet j})$. A *g-aggregated statistics* ${}_2S_n^g$ is

$$\begin{aligned} {}_2S_n^g(u_{\bullet 1}, ..., u_{\bullet d}, C) = \; & g\big({}_2S_n((u_{\bullet 1}, u_{\bullet 2}), C_{12}),\ {}_2S_n((u_{\bullet 1}, u_{\bullet 3}), C_{13}), ..., \\ & {}_2S_n((u_{\bullet 1}, u_{\bullet d}), C_{1d}),\ {}_2S_n((u_{\bullet 2}, u_{\bullet 3}), C_{23}), ..., \\ & {}_2S_n((u_{\bullet 2}, u_{\bullet d}), C_{2d}), ...,\ {}_2S_n((u_{\bullet d-1}, u_{\bullet d}), C_{(d-1)d})\big), \end{aligned} \tag{24}$$

where $C_{ij}, 1 \leq i < j \leq d$, are the bivariate marginal copulas of C.

We employ g-aggregated statistics in order to simplify the computation of $S_n^{(K)}$ and $S_n^{(C)}$ for $d > 2$. Considering the $S_n^{(K)}$ statistic, the main difficulty in its computation consists in expressing K_{θ_n}. For $d = 2$, given a 2-AC $C(\cdot;\psi_{\theta_n})$, where ψ_{θ_n} denotes a generator with a parameter θ_n, K_{θ_n} is the bivariate probability integral transform, which can be easily computed as $K_{\theta_n}(t) = t - \frac{\psi_{\theta_n}^{-1}(t)}{(\psi_{\theta_n}^{-1})'(t)}$; see [16]. However, for $d > 2$ and particularly for HACs, the complexity of K_{θ_n} dramatically increases. In [44], its computation is addressed for HACs, however, the authors restrict only to FNACs, which rarely occurs in our experiments, and, even for FHACs the obtained formulas involve multivariate integration that substantially increases the complexity of their application.

Considering the statistic $S_n^{(C)}$, the main difficulty in its computation consists in expressing e_j, for $j = 2, ..., d$, given by (21). Observe that e_d includes $d-1$ partial derivatives of C_θ, thus its complexity quickly grows in d and the time consumption of its computation exceeds reasonable limits already for $d = 6$, particularly for families with a more complex generator, e.g., for the Frank family. Using g-aggregated statistics, computations for $d > 2$ are substantially simplified.

2.6 Okhrin's algorithm for the structure determination of HAC

We recall the algorithm presented in [44] for the structure determination of HACs, which returns the structure for some unknown HAC C using only the known forms of its bivariate margins. The algorithm uses the following definition.

Definition 8 Let C be a d-HAC with generators $\psi_1, ..., \psi_{d-1}$ and $(U_1, ..., U_d) \sim C$. Define $\mathcal{U}_C(\psi_k) = \{i \in \{1, ..., d\} \mid \text{there exists } j \in \{i+1, ..., d\} \text{ such that } (U_i, U_j) \sim C(\cdot; \psi_k)\}$, $k = 1, ..., d-1$.

Note that $(U_j, U_i) \sim C(\cdot; \psi_k)$ if and only if $(U_i, U_j) \sim C(\cdot; \psi_k)$.

Proposition 1 *[19] Defining $\mathcal{U}_C(u_i) = \{i\}$ for the leaf u_i, $1 \leq i \leq d$, there is a unique disjoint decomposition of $\mathcal{U}_C(\psi_k)$ given by*

$$\mathcal{U}_C(\psi_k) = \mathcal{U}_C(\mathcal{H}_l(\psi_k)) \cup \mathcal{U}_C(\mathcal{H}_r(\psi_k)). \tag{25}$$

For an unknown d-HAC C with all bivariate margins known, its structure can be easily determined using Algorithm 1. We start from the sets $\mathcal{U}_C(u_1), ..., \mathcal{U}_C(u_d)$ joining them together through (25) until we reach the node ψ for which $\mathcal{U}_C(\psi) = \{1, ..., d\}$.

Algorithm 1 HAC structure determination [19]

```
  Input:
  1) $\mathcal{U}_C(\psi_1), ..., \mathcal{U}_C(\psi_{d-1})$,
  2) $\mathcal{I} = \{1, ..., d-1\}$

  while $\mathcal{I} \neq \emptyset$ do
    1. $k = \text{argmin}_{i \in \mathcal{I}}(\#\mathcal{U}_C(\psi_i))$, if there are more minima, then choose one of them.
    2. Find the nodes $\psi_l, \psi_r$, for which $\mathcal{U}_C(\psi_k) = \mathcal{U}_C(\psi_l) \cup \mathcal{U}_C(\psi_r)$.
    3. $\mathcal{H}_l(\psi_k) := \psi_l, \mathcal{H}_r(\psi_k) := \psi_r$.
    4. Set $\mathcal{I} := \mathcal{I} \backslash \{k\}$.
  end while

  Output:
The structure stored in $\mathcal{H}_l(\psi_k), \mathcal{H}_r(\psi_k), k = 1, ..., d-1$
```

2.7 Example

We illustrate Algorithm 1 for a 5-HAC given by $C(C(u_1, u_2; \psi_2), C(u_3, C(u_4, u_5; \psi_4); \psi_3); \psi_1) = C_{\psi_1, ..., \psi_4; ((12)(3(45)))}(u_1, ..., u_5)$. The structure of this copula is depicted on the left side in Figure 2 and its bivariate margins are:

$$\begin{array}{lll}
(U_1, U_2) \sim C(\cdot; \psi_2), & (U_1, U_3) \sim C(\cdot; \psi_1), & (U_1, U_4) \sim C(\cdot; \psi_1), \\
(U_1, U_5) \sim C(\cdot; \psi_1), & (U_2, U_3) \sim C(\cdot; \psi_1), & (U_2, U_4) \sim C(\cdot; \psi_1), \\
(U_2, U_5) \sim C(\cdot; \psi_1), & (U_3, U_4) \sim C(\cdot; \psi_3), & (U_3, U_5) \sim C(\cdot; \psi_3), \\
(U_4, U_5) \sim C(\cdot; \psi_4). & &
\end{array}$$

Now assume that the structure is unknown and only the bivariate margins are known. We see that $\mathcal{U}_C(\psi_1) = \{1, 2, 3, 4, 5\}, \mathcal{U}_C(\psi_2) = \{1, 2\}, \mathcal{U}_C(\psi_3) = \{3, 4, 5\}, \mathcal{U}_C (\psi_4) = \{4, 5\}$. For the leaves $u_1, ..., u_5$, we have $\mathcal{U}_C(u_i) = \{i\}, i = 1, ..., 5$. In Step 1 of Algorithm 1, there are two minima: $k = 2$ and $k = 4$. We arbitrarily choose $k = 4$. As $\mathcal{U}_C(\psi_4) = \mathcal{U}_C(u_4) \cup \mathcal{U}_C(u_5)$, we set $\mathcal{H}_l(\psi_4) := u_4$ and $\mathcal{H}_r(\psi_4) := u_5$ in Step 3. In Step 4, we set $\mathcal{I} = \{1, 2, 3, 5\}$. In the second loop, $k = 2$. As $\mathcal{U}_C(\psi_2) = \mathcal{U}_C(u_1) \cup$

$\mathcal{U}_C(u_2)$, we set $\mathcal{H}_l(\psi_2) := u_1$ and $\mathcal{H}_r(\psi_2) := u_2$ in Step 3. In the third loop, we have $k = 3$. As $\mathcal{U}_C(\psi_3) = \mathcal{U}_C(u_3) \cup \mathcal{U}_C(\psi_4)$, we set $\mathcal{H}_l(\psi_3) := u_3$ and $\mathcal{H}_r(\psi_3) := \psi_4$ in Step 3. In the last loop, we have $k = 1$. As $\mathcal{U}_C(\psi_1) = \mathcal{U}_C(\psi_2) \cup \mathcal{U}_C(\psi_3)$, we set $\mathcal{H}_l(\psi_1) := \psi_2$ and $\mathcal{H}_r(\psi_1) := \psi_3$ in Step 3. Observing the original copula form and Figure 2, we see that we have determined the correct structure, which is stored in $\mathcal{H}_l(\psi_k), \mathcal{H}_r(\psi_k), k = 1, ..., 4$.

3 Our Approach

3.1 HAC structure determination based on Kendall's tau

According to Theorem 2, our goal is to build the HAC such that the sufficient nesting condition is satisfied for each generator and its parent in a HAC structure. The sufficient nesting condition typically results in constraints on the parameters θ_1, θ_2 of the involved generators ψ_1, ψ_2; see, e.g., Table 1 or [23]. As $\theta_i, i = 1, 2$ is related to τ through (8), there is also an important relationship between the values of τ and the HAC tree structure following from the sufficient nesting condition. This relationship is described for the fully-nested 3-HAC (4) in Remark 2.3.2 in [22]. There, it is shown that if the sufficient nesting condition holds for the parent-child pair (ψ_1, ψ_2), then $0 \le \tau(\psi_1) \le \tau(\psi_2)$. We generalize this statement as follows.

Proposition 2 *Let C be a d-HAC with the structure s and the generators $\psi_1, ..., \psi_{d-1}$, where each parent-child pair satisfies the sufficient nesting condition. Then $\tau(\psi_i) \le \tau(\psi_j)$*, where $\psi_j \in \mathcal{D}(\psi_i)$, *holds for each $\psi_i, i = 1, ..., d-1$.*

Proof As $\psi_j \in \mathcal{D}(\psi_i)$, there exists a unique sequence $\psi_{k_1}, ..., \psi_{k_l}$, where $1 \le k_m \le d-1$, $m = 1, ..., l$, $l \le d-1$, $\psi_{k_1} = \psi_i$, $\psi_{k_l} = \psi_j$ and $\psi_{k-1} = \mathcal{P}(\psi_k)$ for $k = 2, ..., l$. Applying the above mentioned remark for each pair $(\psi_{k-1}, \ \psi_k)$, $k = 2, ..., l$, we get $\tau(\psi_{k_1}) \le ... \le \tau(\psi_{k_l})$. □

Thus, having a branch from s, all its nodes are uniquely ordered according to their value of τ assuming unequal values of τ for all parent-child pairs. This provides an alternative algorithm for determining the structure of a HAC. We assign generators with the highest values of τ to the lowest levels of the branches in the structure. Ascending higher up in the tree we assign generators with lower values of τ. Now consider the following definition and proposition.

Definition 9 Let C be a d-HAC and u_i, u_j are two different leaves from the structure of the d-HAC. Then we call *youngest common ancestor* of u_i, u_j (denoted $\mathcal{A}_y(u_i, u_j)$) the node ψ, for which $(\psi \in \mathcal{A}(u_i) \cap \mathcal{A}(u_j)) \wedge (\mathcal{A}(u_i) \cap \mathcal{A}(u_j) \cap \mathcal{D}(\psi) = \emptyset)$.

Remark 2 Let ψ be a generator from a d-HAC structure, $u_i \in \mathcal{D}_l(\mathcal{H}_l(\psi))$ and $u_j \in \mathcal{D}_l(\mathcal{H}_r(\psi))$. Then $\mathcal{A}_y(u_i, u_j) = \psi$.

Note that due to clear correspondence of the variables in a d-HAC and the leaves in the structure of the same d-HAC, both the variables and the leaves are denoted by the same $u_1, \ldots, u_d$. This can be made without a worry to confuse the reader.

Proposition 3 *Let C be a d-HAC with the structure s with generators $\psi_1, ..., \psi_{d-1}$. Then*

$$C(1, ..., 1, u_i, 1, ..., 1, u_j, 1, ..., 1) = C(u_i, u_j; \mathcal{A}_y(u_i, u_j)), 1 \leq i < j \leq d. \tag{26}$$

Proof The proof is leaded by induction. Let $d = 2$. Then $C(u_1, u_2) = C(u_1, u_2; \psi_1)$, i.e., the leaves u_1 and u_2 are the children of ψ_1. It implies that $(\psi_1 \in \mathcal{A}(u_1) \cap \mathcal{A}(u_2)) \wedge (\mathcal{A}(u_1) \cap \mathcal{A}(u_2) \cap \mathcal{D}(\psi_1) = \emptyset)$ and thus $\psi_1 = \mathcal{A}_y(u_1, u_2)$ according to Definition 9.

Assume $d \geq 3$ and that (26) holds for $d-1,\ d-2, ...,\ 3$. Start denoting the root node of s as ψ_m. The bivariate marginal copula of C corresponding to variables u_i, u_j is $C(1, ..., 1,\ u_i, 1, ..., 1,\ u_j, 1, ..., 1; \psi_1, ..., \psi_{d-1})$. To simplify notation, we show in each involved inner HAC only the generator corresponding to the highest node in its structure. Thus, for the bivariate marginal copula, we simplify its notation to $C(1, ..., 1,\ u_i, 1, ..., 1,\ u_j, 1, ..., 1; ..., \psi_m, ...)$. Note that $C(1, ..., 1) = 1$ and $C(1, ..., 1, u, 1, ..., 1) = u, u \in \mathbb{I}$ for an arbitrary copula C.

If $\mathcal{H}_l(\psi_m) = u_k, k = 1, ..., d$, we just formally define $\psi_l = u_k$ and $C(\cdot; \psi_l) = u_k$. If $\mathcal{H}_r(\psi_m) = u_k, k = 1, ..., d$, we also just formally define $\psi_r = u_k$ and $C(\cdot; \psi_r) = u_k$. Although neither $C(\cdot; \psi_l)$ nor $C(\cdot; \psi_r)$ are copulas, this will simplify the notation used in the proof. In other case, we set $\psi_l = \mathcal{H}_l(\psi_m), \psi_r = \mathcal{H}_r(\psi_m)$. Now, we distinguish the three following situations:

1. If $u_i \in \mathcal{D}_l(\psi_l)$ and $u_j \in \mathcal{D}_l(\psi_r)$, then $C(C(1, ..., 1, u_i, 1, ..., 1; ..., \psi_l, ...), C(1, ..., 1, u_j, 1, ..., 1; ..., \psi_r, ...); \psi_m) = C(u_i, u_j; \psi_m)$. As $\psi_m = \mathcal{A}_y(u_i, u_j)$ (Remark 2), the statement holds.
2. If $\{u_i, u_j\} \subset \mathcal{D}_l(\psi_l)$, then $C(C(1, ..., 1, u_i, 1, ..., 1, u_j, 1, ..., 1; ..., \psi_l, ...),\ C(1, ..., 1; ..., \psi_r, ...); \psi_m) = C(1, ..., 1, u_i, 1, ..., 1, u_j, 1, ..., 1; ..., \psi_l, ...)$. Since the tree rooted in ψ_l has less leaves than the tree rooted in ψ_m, for $C(1, ..., 1, u_i, 1, ..., 1, u_j, 1, ..., 1; ..., \psi_l, ...)$ we already know that (26) holds, thus it holds also for $C(1, ..., 1, u_i, 1, ..., 1, u_j, 1, ..., 1;\ ..., \psi_m, ...)$.
3. If $\{u_i, u_j\} \subset \mathcal{D}_l(\psi_r)$, then $C(C(1, ..., 1; ..., \psi_l, ...), C(1, ..., 1, u_i, 1, ..., 1, u_j, 1, ..., 1; ..., \psi_r, ...); \psi_m) = C(1, ..., 1, u_i, 1, ..., 1, u_j, 1, ..., 1\ ; ..., \psi_r, ...)$. Since the tree rooted in ψ_r has less leaves than the tree rooted in ψ_m, for $C(1, ..., 1, u_i, 1, ..., 1, u_j, 1, ..., 1; ..., \psi_r, ...)$ we already know that (26) holds, thus it holds also for $C(1, ..., 1, u_i, 1, ..., 1, u_j, 1, ..., 1; ..., \psi_m, ...)$. □

Thus (U_i, U_j) is distributed according to the 2-AC $C(\cdot; \mathcal{A}_y(u_i, u_j))$ for all $i, j \in \{1, ..., d\},\ i \neq j$. This fact allows to prove the following proposition.

Proposition 4 *Let C be a d-HAC with the generators $\psi_1, ..., \psi_{d-1}$, $(U_1, ..., U_d) \sim C$ and (τ_{ij}) be the population version of the Kendall correlation matrix of $(U_1, ..., U_d)$. Then, given $k \in \{1, ..., d-1\}$,*

$$\tau(\psi_k) = \tau_{ij} \tag{27}$$

for all $(u_i, u_j) \in \mathcal{D}_l(\mathcal{H}_l(\psi_k)) \times \mathcal{D}_l(\mathcal{H}_r(\psi_k))$.

Proof Recall that $\tau_{ij} = \tau_{U_i, U_j}$ and $\tau(\psi_k) = \tau(C(\cdot; \psi_k))$ by definition and let $k \in \{1, ..., d-1\}$ and $(u_i, u_j) \in \mathcal{D}_l(\mathcal{H}_l(\psi_k)) \times \mathcal{D}_l(\mathcal{H}_r(\psi_k))$. Using Proposition 3, it implies $(U_i, U_j) \sim C(\cdot; \mathcal{A}_y(u_i, u_j))$. As $\psi_k = \mathcal{A}_y(u_i, u_j)$ according to Remark 2, it follows that $(U_i, U_j) \sim C(\cdot; \psi_k)$. Hence, $\tau_{U_i, U_j} = \tau(C(\cdot; \psi_k))$. □

Algorithm 2 HAC structure determination based on τ

```
Input:
  1) $\mathcal{I} = \{1, ..., d\}$,
  2) $(U_1, ..., U_d) \sim C$,
  3) $\tau^g$ ... an aggregated Kendall's tau with an $\mathbb{I}$-aggregation function $g$,
  4) $z_k = u_k,\ \mathcal{U}_C(z_k) = \{k\},\ k = 1, ..., d$

  The structure determination:
  for $k = 1, ..., d-1$ do
    1. $(i, j) := \underset{i^* < j^*, i^* \in \mathcal{I}, j^* \in \mathcal{I}}{\operatorname{argmax}} \tau^g(\mathbf{U}_{\mathcal{U}_C(z_{i^*})\mathcal{U}_C(z_{j^*})})$
    2. $\mathcal{U}_C(z_{d+k}) := \mathcal{U}_C(z_i) \cup \mathcal{U}_C(z_j)$
    3. $\mathcal{I} := \mathcal{I} \cup \{d+k\} \backslash \{i, j\}$
  end for

Output:
  $\mathcal{U}_C(\psi_k) = \mathcal{U}_C(z_{d+k}), k = 1, ..., d-1$
```

Remark 3 It holds that $\tau(\psi_k) = \tau^g(\mathbf{U}_{\mathcal{D}_l(\mathcal{H}_l(\psi_k)) \times \mathcal{D}_l(\mathcal{H}_r(\psi_k))})$ for a d-HAC C and for each $k = 1, ..., d-1$. This is because, given $k \in \{1, ..., d-1\}$, the values of τ_{ij} for $(u_i, u_j) \in \mathcal{D}_l(\mathcal{H}_l(\psi_k)) \times \mathcal{D}_l(\mathcal{H}_r(\psi_k))$ are all equal to $\tau(\psi_k)$, see Proposition 4, and $g(u, ..., u) = u$ for all $u \in \mathbb{I}$.

Computing $\tau(\psi_k), k = 1, ..., d-1$, according to Remark 3 and using Proposition 2 leads to an alternative algorithm for HAC structure determination; see Algorithm 2. This algorithm can be used for arbitrary $d > 2$ (see [19] for more details including an example for $d = 4$). It returns the sets $\mathcal{U}_C(\psi_k), k = 1, ..., d-1$. Passing them to Algorithm 1, we avoid the computation of $\mathcal{U}_C(\psi_k), k = 1, ..., d-1$ in Definition 8 and we get the requested d-HAC structure without having to know the forms of the bivariate margins. Assuming a parametric family for each ψ_k, the $\theta - \tau$ relationship for the given family can be used to obtain the parameters, i.e., $\theta_k = \tau_\theta^{-1}(\tau(\psi_k)), k = 1, ..., d-1$, where τ_θ^{-1} denotes this $\theta - \tau$ relationship, e.g., for the Clayton family $\tau_\theta^{-1}(\tau) = 2\tau/(1-\tau)$. In other words, assuming $(U_1, ..., U_d) \sim C$, where C is a d-HAC with one-parametric generators $\psi_1, ..., \psi_{d-1}$ from the same family, if C is unknown but the population version of the Kendall correlation matrix (τ_{ij}) is known, both structure and parameters of C can be obtained from (τ_{ij}) using Algorithms 1 and 2. Based on the empirical version of the Kendall correlation matrix, we thus obtain the following approach for both determining the structure and estimating parameters of C.

3.2 Structure determination and parameter estimation of a HAC

Using τ_n^g instead of τ^g, we can easily derive a new approach for structure determination and parameter estimation of a HAC from Algorithms 1 and 2. The approach is summarized in Algorithm 3. The algorithm returns the parameters $\hat{\theta}_1, ..., \hat{\theta}_{d-1}$ of the estimate $\hat{C}$ and the sets $\mathcal{U}_{\hat{C}}(\psi_k), k = 1, ..., d-1$. Passing the sets to Algorithm 1, we get the requested $\hat{C}$ structure.

From Algorithm 3, the reader can see our motivation for basing the estimation process on Kendall's tau. Firstly, the matrix (τ_{ij}^n) is computed in order to determine the structure of a HAC. Then, the computed values of (τ_{ij}^n) are reused for the

Algorithm 3 HAC structure and parameter estimation

```
Input:
  1) $(\tau^n_{ij})$ ... the sample version of the Kendall correlation matrix,
  2) $g$ ... an $\mathbb{I}$-aggregation function,
  3) $\mathcal{I} = \{1, ..., d\}$,
  4) $z_i = u_i,\ \mathcal{U}_{\hat{C}}(z_i) = \{i\},\ \tilde{\theta}_i = \infty,\ i = 1, ..., d$,
  5) Archimedean family based on a generator $\psi$, and the corresponding $\tau^{-1}$

  Estimation:
  for $k = 1, ..., d-1$ do
    1. $(i, j) := \underset{\tilde{i}<\tilde{j}, \tilde{i}\in\mathcal{I}, \tilde{j}\in\mathcal{I}}{\operatorname{argmax}} g\big((\tau^n_{\tilde{\tilde{i}}\tilde{\tilde{j}}})_{(\tilde{\tilde{i}},\tilde{\tilde{j}})\in\mathcal{U}_{\hat{C}}(z_{\tilde{i}})\times\mathcal{U}_{\hat{C}}(z_{\tilde{j}})}\big)$
    2. $\tilde{\theta}_{d+k} := \tau^{-1}\big(g((\tau^n_{\tilde{i}\tilde{j}})_{(\tilde{i},\tilde{j})\in\mathcal{U}_{\hat{C}}(z_i)\times\mathcal{U}_{\hat{C}}(z_j)})\big)$
    3. $\tilde{\theta}_{d+k} := \min\{\tilde{\theta}_{d+k}, \tilde{\theta}_i, \tilde{\theta}_j\}$
    4. $z_{d+k} := (u_i, u_j; \psi)$ ... formal introduction of the variable $z_{d+k}$
    5. $\mathcal{U}_{\hat{C}}(z_{d+k}) := \mathcal{U}_{\hat{C}}(z_i) \cup \mathcal{U}_{\hat{C}}(z_j)$
    6. $\mathcal{I} := \mathcal{I} \cup \{d+k\} \backslash \{i, j\}$
  end for

Output:
  $\hat{\theta}_k = \tilde{\theta}_{d+k},\ \mathcal{U}_{\hat{C}}(\psi_k) = \mathcal{U}_{\hat{C}}(z_{d+k}),\ k = 1, ..., d-1$
```

estimation of the parameters. The latter can be done effectively as the function τ^{-1} is known in closed form for many Archimedean families, e.g., for the Clayton and Gumbel families listed in Table 1, cf. [23]. As we will see in Section 4, the estimator is comparably fast to compute, at least if d is not too large. Theoretically, Spearman's rho or Blomqvist's beta could be considered for this task as well despite the fact that these rank correlation measures are much less popular in this domain. It is also known that Kendall's tau works well in comparison to Blomqvist's beta; see [26].

If g is set to be the average function then $\tau^{\text{avg}}_n(\theta_k) = g\big((\tau^n_{\tilde{i}\tilde{j}})_{(\tilde{i},\tilde{j})\in\mathcal{U}_{\hat{C}}(z_i)\times\mathcal{U}_{\hat{C}}(z_j)}\big)$ (i, j are the indices found in Step 1 of Algorithm 3) is an unbiased estimator of $\tau(\theta_k)$, and thus the structure determination is based only on unbiased estimators, which is another favorable property of the proposed method. Note that recently an approach allowing for consistent estimation of all parameters of a HAC been published [18]. Its comparison with the approach presented here is a topic of future research.

In order to fulfill the sufficient nesting condition, the parameter $\tilde{\theta}_{d+k}$ is trimmed in Step 3 in order to obtain a proper d-HAC. Note that one can allow the generators to be from different Archimedean families. However, this case is more complex and we do not address it in this paper; see [21,22].

Note that Algorithm 3 is a variation of the algorithm for agglomerative hierarchical clustering (AHC) [9, p. 414]. Defining $\delta_{ij} = 1 - \tau^n_{ij}$, δ_{ij} is a commonly used distance between the random variables U_i, U_j. Setting g to be the aggregation function minimum, average or maximum, the algorithm results in complete-linkage, average-linkage or single-linkage AHC, respectively [9, p. 414]. As many types of statistical software include an implementation of AHC, the implementation of the proposed algorithm is straightforward. Moreover, adding the dendrogram obtained during AHC simplifies the interpretation of the estimator; see Figure 8 in Section 5.

4 Experiments on simulated data

4.1 Design of the performed experiments

In this section, we compare our methods for HAC estimation based on Algorithm 3 with several methods presented in [43], which are implemented in R, see [45]. As we are interested in binary structured HACs, we choose for the comparison the methods θ_{bin}, θ_{RML}, τ_{bin}, which return binary structured HAC estimates as their results (note that the θ_{RML} method also allows for non-binary structured HACs estimation). The first two methods are based on the ML estimation technique, whereas the third method is based on the $\theta - \tau$ relationship. Our methods are denoted by $\tau_{\text{bin}}^{\text{min}}, \tau_{\text{bin}}^{\text{max}}$ and $\tau_{\text{bin}}^{\text{avg}}$, i.e., the involved function g, see Algorithm 3, is selected to be the minimum, maximum and average, respectively. The first two functions are selected as they represent "extremes" of $\mathbb{I}$-aggregation functions. The last function is selected due to the reasons mentioned in Section 3.2, i.e., if g is the average function, the structure determination is based on unbiased estimates of $\tau(\theta_k), k = 1, ..., d-1$.

The comparison is performed on simulated data for $d \in \{5, 6, 7, 9\}$. We selected the maximal dimension $d = 9$ for two reasons. The first reason is that the results for $d > 9$ do not bring any surprising information about the differences among the considered methods. The second reason is that, for $d \leq 9$, the obtained structure estimate representations (described in the following paragraph) involve single-digit numbers only, which allows for more concise notation. We simulated $N = 1000$ samples of size $n = 500$ according to [23] for 4 copula models based on Clayton generators. Our choice of the Clayton family of generators was due to the intended comparison of our method with the above-mentioned methods that are implemented for the Gumbel and Clayton family of generators only. The Clayton family of generators was chosen arbitrarily from these two after we have experimented with both families and have found out that results for both of them are similar.

The first considered model is $((12)_{\frac{3}{4}}(3(45)_{\frac{4}{4}})_{\frac{3}{4}})_{\frac{2}{4}}$. The natural numbers in the model notation (as in [43]) are the indexes of the copula variables, i.e., 1,...5, the parentheses correspond to each $\mathcal{U}_C(\cdot)$ of individual copulas, i.e., $\mathcal{U}_C(\psi_1) = \{1, 2, 3, 4, 5\}, \mathcal{U}_C(\psi_2) = \{3, 4, 5\}, \mathcal{U}_C(\psi_3) = \{1, 2\}, \mathcal{U}_C(\psi_4) = \{4, 5\}$, and the subscripts are the model parameters, i.e, $(\theta_1, \theta_2, \theta_3, \theta_4) = (\frac{2}{4}, \frac{3}{4}, \frac{3}{4}, \frac{4}{4})$. Note that the indices of the 4 generators could be permuted arbitrarily, and our particular selection of their ordering just serves for better illustration. The other 3 models are given with analogously by $(1((23)_{\frac{5}{4}}(4(56)_{\frac{6}{4}})_{\frac{5}{4}})_{\frac{4}{4}})_{\frac{2}{4}}$, $(1((23)_{\frac{5}{4}}(4(5(67)_{\frac{7}{4}})_{\frac{6}{4}})_{\frac{5}{4}})_{\frac{4}{4}})_{\frac{2}{4}}$ and $((1(2(34)_{\frac{5}{4}})_{\frac{4}{4}})_{\frac{3}{4}}\ ((56)_{\frac{4}{4}}(7(89)_{\frac{5}{4}})_{\frac{4}{4}})_{\frac{3}{4}})_{\frac{2}{4}}$. The smallest difference between the parameters is set to $\frac{1}{4}$ and the values of the parameters are set in the way that the sufficient nesting condition is satisfied for each parent-child pair of the generators. As we discovered while experimenting with different parametrizations, a larger difference in the parameters could hide the impact of the bias in most of the methods of [43] on the structure determination, and the results obtained by different methods can be similar for those parametrizations. Smaller differences than $\frac{1}{4}$ were not necessary as setting them to $\frac{1}{4}$ fully reveals the impact of the bias and clearly shows the difference among the methods. This fact is illustrated in the following subsection in the part where the methods are assessed in terms of ability to determine the true copula structure.

4.2 Results of the experiments

The results for $d \in \{5, 6\}$ are shown in Tables 2 and 4, where the first table concerns the structures determined by the methods, whereas the second table concerns goodness-of-fit of the HACs estimated by the methods and time consumption of the methods. Similarly, the results for $d \in \{7, 9\}$ are shown in Tables 3 and 5. Result for different models are separated by double lines. Note that all experiments were performed on a PC with Intel Core 2.3 GHz CPU and 4GB RAM. As θ_{RML} failed in most cases for $d = 9$ on the described hardware configuration, the result of the method for this dimension is not presented.

The third column in Tables 2 and 3 shows the number of different estimated copula structures (denoted #d.s.) in $N = 1000$ runs of the considered method. The value gives us information on how much the resulting estimated structure varies for a given method and model. The lower the value is, the more stable the structure determination can be considered. For $d = 5, 6$, θ_{bin} and θ_{RML} show the strongest stability, whereas τ_{bin} shows the weakest stability. For $d = 7$, the situation slightly changes and θ_{bin} and τ_{bin} clearly represent two extremes – the first showing substantially stronger stability than the remaining methods and the latter represents the opposite. As the dimension increases, we observe comparably increasing stability for $\tau_{\mathrm{bin}}^{\mathrm{avg}}$ until it reaches the best stability for $d = 9$. In all considered dimensions, we observe that $\tau_{\mathrm{bin}}^{\mathrm{max}}$ shows slightly worse stability than $\tau_{\mathrm{bin}}^{\mathrm{min}}$ and $\tau_{\mathrm{bin}}^{\mathrm{avg}}$.

The next two columns in Tables 2 and 3 address the ability of the methods to determine the true copula structure. The fourth column shows the three most frequent structures obtained by the method (if the true structure is not one of three the most frequent structures, then we add it in the fourth row corresponding to the method) with average parameter values. The true structure is emphasized by bold text. The fifth column shows the frequency of the true structure in all estimated structures. The methods $\tau_{\mathrm{bin}}^{\mathrm{min}}$ and $\tau_{\mathrm{bin}}^{\mathrm{avg}}$ dominate in the ability to determine the true copula structure in all four cases ($d \in \{5, 6, 7, 9\}$). The $\tau_{\mathrm{bin}}^{\mathrm{max}}$ method ranks as the third best, also in all four cases. The remaining methods show very poor ability to detect the true structure, especially for $d \geq 7$. For example, for $d = 7$, θ_{RML} returned the true structure only 2 times out of 1000. For $d = 9$, the difference between our and the remaining methods is most obvious. The worst performance shows the θ_{bin} method, which did not return *any* estimate with the true structure. The τ_{bin} method, which returned 6.2%, is also substantially outperformed by all of our methods.

The ability of the methods to determine the true copula structure is additionally illustrated in Figure 3, which shows the frequency of the true structure in 1000 estimated structures for the considered methods, for sample sizes $10, 20, ..., 500$ and for the differences in the parameters set consecutively to $1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}$, namely for four 5-HAC models $((12)_{3*q}(3(45)_{4*q})_{3*q})_{2*q}$ with $q = 1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}$, respectively. For $q = 1$, we observe that the frequency of the true structure for the considered sample sizes is similar for all the considered methods except the θ_{RML} method and approaches to 100% as the sample size increases. For θ_{RML}, the frequency never exceeds 55%

Table 2 The first part of the results for the copula models for $d \in \{5, 6\}$. The columns contain: method denotation; total number of different estimated structures (#d.s); the 3 most frequent estimated structures with average parameter values; frequency of the true structure in all estimated structures (in %). The values corresponding to the true structure are in bold.

d	Method	#d.s.	Structure(s)	%
5	θ_{bin}	9	$(3((12)_{0.77}(45)_{1.00})_{0.75})_{0.24}$	78.7
			$\mathbf{((12)_{0.68}(3(45)_{1.03})_{0.73})_{0.68}}$	**19**
			$(5((12)_{0.78}(34)_{0.91})_{0.78})_{0.24}$	0.8
	θ_{RML}	9	$\mathbf{((12)_{0.71}(3(45)_{1.01})_{0.78})_{0.53}}$	**49.7**
			$((45)_{1.00}(3(12)_{0.80})_{0.72})_{0.62}$	47.1
			$(3((12)_{0.89}(45)_{0.83})_{0.54})_{0.53}$	1.2
	τ_{bin}	20	$\mathbf{((12)_{0.81}(3(45)_{1.01})_{0.93})_{0.89}}$	**45.3**
			$(1(2(3(45)_{1.02})_{0.93})_{0.78})_{0.86}$	22.2
			$(2(1(3(45)_{1.03})_{0.93})_{0.78})_{0.85}$	20.9
	$\tau_{\text{bin}}^{\text{min}}$	11	$\mathbf{((12)_{0.76}(3(45)_{1.01})_{0.70})_{0.41}}$	**92**
			$((12)_{0.75}(5(34)_{0.92})_{0.74})_{0.40}$	3.4
			$((12)_{0.75}(4(35)_{0.90})_{0.75})_{0.40}$	2.8
	$\tau_{\text{bin}}^{\text{max}}$	15	$\mathbf{((12)_{0.77}(3(45)_{1.01})_{0.80})_{0.59}}$	**83.6**
			$(1(2(3(45)_{1.06})_{0.82})_{0.66})_{0.61}$	3.9
			$((12)_{0.75}(5(34)_{0.92})_{0.87})_{0.60}$	3.3
	$\tau_{\text{bin}}^{\text{avg}}$	11	$\mathbf{((12)_{0.76}(3(45)_{1.01})_{0.75})_{0.50}}$	**91.3**
			$((12)_{0.75}(5(34)_{0.92})_{0.80})_{0.50}$	3.4
			$((12)_{0.75}(4(35)_{0.90})_{0.80})_{0.50}$	2.8
6	θ_{bin}	14	$(1(4((23)_{1.29}(56)_{1.50})_{1.29})_{0.56})_{0.18}$	51.7
			$((14)_{0.57}((23)_{1.25}(56)_{1.49})_{1.25})_{0.57}$	24.2
			$\mathbf{(1((23)_{1.16}(4(56)_{1.55})_{1.23})_{1.16})_{0.22}}$	**17.5**
	θ_{RML}	14	$(1((56)_{1.50}(4(23)_{1.30})_{1.21})_{1.08})_{0.51}$	47.3
			$\mathbf{(1((23)_{1.21}(4(56)_{1.52})_{1.27})_{1.00})_{0.50}}$	**45**
			$(1((23)_{1.22}(5(46)_{1.39})_{1.31})_{1.01})_{0.50}$	2.2
	τ_{bin}	26	$(1(2(3(4(56)_{1.53})_{1.48})_{1.39})_{1.38})_{0.70}$	37.6
			$(1(3(2(4(56)_{1.54})_{1.50})_{1.41})_{1.40})_{0.70}$	36.7
			$\mathbf{(1((23)_{1.43}(4(56)_{1.54})_{1.50})_{1.40})_{0.72}}$	**5.5**
	$\tau_{\text{bin}}^{\text{min}}$	21	$\mathbf{(1((23)_{1.26}(4(56)_{1.52})_{1.20})_{0.88})_{0.43}}$	**83.6**
			$(1((23)_{1.24}(5(46)_{1.38})_{1.19})_{0.85})_{0.41}$	5.8
			$(1((23)_{1.27}(6(45)_{1.47})_{1.24})_{0.88})_{0.44}$	3.6
	$\tau_{\text{bin}}^{\text{max}}$	22	$\mathbf{(1((23)_{1.28}(4(56)_{1.52})_{1.30})_{1.11})_{0.58}}$	**68.2**
			$(1(2(3(4(56)_{1.52})_{1.31})_{1.16})_{1.12})_{0.57}$	7.4
			$(1(3(2(4(56)_{1.56})_{1.34})_{1.17})_{1.11})_{0.59}$	6.5
	$\tau_{\text{bin}}^{\text{avg}}$	21	$\mathbf{(1((23)_{1.26}(4(56)_{1.52})_{1.25})_{1.00})_{0.50}}$	**83.1**
			$(1((23)_{1.24}(5(46)_{1.38})_{1.25})_{0.98})_{0.49}$	5.7
			$(1((23)_{1.27}(6(45)_{1.46})_{1.30})_{1.00})_{0.52}$	3.6

and the same holds for the remaining $q = \frac{1}{2}, \frac{1}{3}, \frac{1}{4}$. This fact indicates that, from a certain level that is lower than 100%, the θ_{RML} method is not able to improve in estimation of the true structure even with increasing sample size. Decreasing in q, the difference between our methods and the remaining methods in the frequency of the true structure for the considered sample sizes increases. We also observe that the $\tau_{\text{bin}}^{\text{min}}$ and $\tau_{\text{bin}}^{\text{avg}}$ methods are methods that most quickly approach to 100% frequency of the true structure for for all $q = 1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}$ while increasing the sample size. The third most successful method is clearly $\tau_{\text{bin}}^{\text{min}}$ for $q = \frac{1}{2}, \frac{1}{3}, \frac{1}{4}$. For the remaining methods and $q = \frac{1}{3}, \frac{1}{4}$, the frequency of the true structure remains below 70%, 60%, respectively. Surprisingly, for $q = \frac{1}{3}, \frac{1}{4}$, the θ_{bin} method shows (approximately) decreasing frequency of the true structure with increasing sample size for the sample sizes larger than (approximately) 200.

Table 3 The first part of the results for the copula models for $d \in \{7, 9\}$. The columns contain: method denotation; total number of different estimated structures (#d.s); the 3 most frequent estimated structures with average parameter values; frequency of the true structure in all estimated structures (in %). The values corresponding to the true structure are in bold.

d	Method	#d.s.	Structure(s)	%
7	θ_{bin}	10	$(1((23)_{1.00}((45)_{1.01}(67)_{1.01})_{0.96})_{0.78})_{0.16}$	82.4
			$\mathbf{((1(23)_{1.06})_{0.75}((45)_{0.99}(67)_{0.99})_{0.94})_{0.75}}$	**9.7**
			$(1((67)_{0.88}((23)_{1.01}(45)_{1.05})_{0.91})_{0.87})_{0.16}$	3.1
	θ_{RML}	33	$((23)_{1.01}(1((45)_{1.01}(67)_{1.01})_{0.58})_{0.57})_{0.56}$	29.2
			$((67)_{1.00}((23)_{1.08}(1(45)_{0.93})_{0.77})_{0.63})_{0.63}$	16.7
			$((45)_{1.00}((23)_{1.07}(1(67)_{0.93})_{0.76})_{0.63})_{0.62}$	15.7
			$\mathbf{((1(23)_{0.77})_{0.53}((45)_{1.01}(67)_{1.02})_{0.77})_{0.53}}$	**0.2**
	τ_{bin}	97	$\mathbf{((1(23)_{1.01})_{0.96}((45)_{1.06}(67)_{1.05})_{1.00})_{1.03}}$	**13**
			$(1((23)_{1.00}((45)_{1.06}(67)_{1.05})_{1.00})_{0.92})_{0.96}$	8.5
			$((1(23)_{1.00})_{0.95}(4(5(67)_{1.05})_{0.99})_{0.99})_{1.03}$	8
	$\tau_{\text{bin}}^{\text{min}}$	22	$\mathbf{((1(23)_{1.02})_{0.70}((45)_{1.01}(67)_{1.01})_{0.67})_{0.38}}$	**87.5**
			$((3(12)_{0.91})_{0.73}((45)_{0.97}(67)_{0.98})_{0.64})_{0.36}$	3.5
			$((2(13)_{0.91})_{0.75}((45)_{1.02}(67)_{1.02})_{0.68})_{0.37}$	2.6
	$\tau_{\text{bin}}^{\text{max}}$	38	$\mathbf{((1(23)_{1.02})_{0.80}((45)_{1.01}(67)_{1.01})_{0.83})_{0.63}}$	**72.5**
			$((1(23)_{1.04})_{0.80}(4(5(67)_{1.03})_{0.90})_{0.85})_{0.62}$	3.6
			$(1((23)_{1.00}((45)_{1.05}(67)_{1.04})_{0.86})_{0.72})_{0.68}$	3.4
	$\tau_{\text{bin}}^{\text{avg}}$	20	$\mathbf{((1(23)_{1.02})_{0.75}((45)_{1.01}(67)_{1.01})_{0.75})_{0.50}}$	**85.5**
			$((3(12)_{0.91})_{0.80}((45)_{0.99}(67)_{0.98})_{0.74})_{0.49}$	3.3
			$((2(13)_{0.91})_{0.80}((45)_{1.01}(67)_{1.02})_{0.76})_{0.50}$	2.7
9	θ_{bin}	34	$((17)_{0.50}((2(34)_{1.26})_{0.91}((56)_{1.02}(89)_{1.26})_{1.02})_{0.90})_{0.50}$	67.5
			$(1((2(34)_{1.25})_{0.87}((56)_{0.96}(7(89)_{1.28})_{1.00})_{0.95})_{0.87})_{0.13}$	9.4
			$(1((56)_{0.88}((2(34)_{1.26})_{0.96}(7(89)_{1.29})_{0.96})_{0.93})_{0.71})_{0.12}$	5.3
	τ_{bin}	116	$((1(2(34)_{1.27})_{1.21})_{1.07}(5(6(7(89)_{1.28})_{1.20})_{1.09})_{1.09})_{1.11}$	13.2
			$((1(2(34)_{1.29})_{1.22})_{1.07}(6(5(7(89)_{1.28})_{1.22})_{1.10})_{1.10})_{1.11}$	12.1
			$(1((2(34)_{1.26})_{1.19}(5(6(7(89)_{1.30})_{1.23})_{1.12})_{1.11})_{1.05})_{1.03}$	11.4
			$\mathbf{((1(2(34)_{1.26})_{1.22})_{1.09}((56)_{1.09}(7(89)_{1.30})_{1.24})_{1.08})_{1.12}}$	**6.2**
	$\tau_{\text{bin}}^{\text{min}}$	32	$\mathbf{((1(2(34)_{1.27})_{0.96})_{0.68}((56)_{1.01}(7(89)_{1.28})_{0.95})_{0.65})_{0.36}}$	**76.7**
			$((1(2(34)_{1.24})_{0.91})_{0.66}((56)_{1.00}(9(78)_{1.16})_{0.98})_{0.65})_{0.35}$	4
			$((1(4(23)_{1.14})_{0.98})_{0.66}((56)_{0.97}(7(89)_{1.28})_{0.96})_{0.64})_{0.37}$	3.7
	$\tau_{\text{bin}}^{\text{max}}$	55	$\mathbf{((1(2(34)_{1.27})_{1.06})_{0.82}((56)_{1.02}(7(89)_{1.28})_{1.05})_{0.85})_{0.65}}$	**62.5**
			$((1(2(34)_{1.30})_{1.06})_{0.81}(5(6(7(89)_{1.29})_{1.09})_{0.91})_{0.85})_{0.65}$	4.4
			$((1(2(34)_{1.25})_{1.05})_{0.84}(6(5(7(89)_{1.33})_{1.09})_{0.93})_{0.89})_{0.65}$	3.8
	$\tau_{\text{bin}}^{\text{avg}}$	26	$\mathbf{((1(2(34)_{1.27})_{1.01})_{0.75}((56)_{1.01}(7(89)_{1.28})_{1.00})_{0.75})_{0.50}}$	**78.7**
			$((1(2(34)_{1.24})_{0.96})_{0.73}((56)_{1.01}(9(78)_{1.16})_{1.04})_{0.74})_{0.49}$	4.2
			$((1(4(23)_{1.14})_{1.03})_{0.73}((56)_{0.98}(7(89)_{1.28})_{1.00})_{0.75})_{0.50}$	3.8

Next, we assess the methods by means of goodness-of-fit. The results can be seen in columns 3-6 in Tables 4 and 5, where the averages and standard deviations of four GoF statistics are shown. The values in each row correspond to the averages of the GoF statistics over all estimates with the structure corresponding to the one shown in the same row in Tables 2 and 3. The ${}_dS_n$ corresponds directly to the statistics given by (15). By the lower index d in the notation, we accentuate the fact that this is non-aggregated, i.e, “truly” d-dimensional statistics, as the rest of the statistics, ${}_2S_n^{\max}$, ${}_2S_n^{(K)\max}$, ${}_2S_n^{(C)\max}$, are the aggregated (using max function) statistics given by Definition 7 that are based on the bivariate statistics $S_n, S_n^{(K)}, S_n^{(C)}$, respectively. The reason for choosing the maximum function as the $\mathbb{I}$-aggregation function g is that then this g-aggregated statistics can be interpreted in the way that it evaluate how the estimate fits the data according to its worst fitting bivariate margin. Observing the results, we see that the $\tau_{\text{bin}}^{\text{avg}}$

Table 4 The second part of the results for the copula models for $d \in \{5,6\}$. The columns contain: method denotation; GoF test statistics ${}_dS_n$, ${}_2S_n^{\max}$, ${}_2S_n^{(K)\max}$, ${}_2S_n^{(C)\max}$; the average estimation time of one estimation process in s. The values corresponding to the true structure are in bold. The values in parenthesis are the corresponding standard deviations. The last row for each dimension and each method, denoted by *false structures* in the second column, shows averages of the considered statistics over all estimates with structures different to the true structure.

d	Method	${}_dS_n$	${}_2S_n^{\max}$	${}_2S_n^{(K)\max}$	${}_2S_n^{(C)\max}$	time (in s)
5	θ_{bin}	0.18 (0.09)	0.63 (0.29)	2.11 (0.4)	0.69 (0.29)	0.079 (0.023)
		0.11 (0.09)	**0.38 (0.22)**	**0.51 (0.25)**	**0.35 (0.18)**	
		0.20 (0.08)	0.76 (0.4)	2.84 (0.5)	0.82 (0.4)	
	false structures	0.18 (0.09)	0.64 (0.29)	2.11 (0.5)	0.69 (0.29)	
	θ_{RML}	**0.08 (0.06)**	**0.31 (0.19)**	**0.21 (0.09)**	**0.27 (0.13)**	0.172 (0.024)
		0.10 (0.08)	0.36 (0.2)	0.50 (0.25)	0.33 (0.16)	
		0.08 (0.03)	0.34 (0.13)	0.45 (0.2)	0.33 (0.12)	
	false structures	0.10 (0.08)	0.37 (0.2)	0.50 (0.25)	0.33 (0.16)	
	τ_{bin}	**0.25 (0.14)**	**0.43 (0.23)**	**1.22 (0.28)**	**0.51 (0.21)**	0.190 (0.008)
		0.21 (0.13)	0.40 (0.22)	0.92 (0.26)	0.44 (0.21)	
		0.20 (0.12)	0.37 (0.2)	0.95 (0.27)	0.42 (0.18)	
	false structures	0.21 (0.13)	0.40 (0.22)	0.96 (0.27)	0.45 (0.2)	
	$\tau_{\text{bin}}^{\min}$	**0.10 (0.07)**	**0.32 (0.18)**	**0.37 (0.2)**	**0.29 (0.15)**	0.065 (0.02)
		0.10 (0.08)	0.33 (0.22)	0.43 (0.18)	0.31 (0.19)	
		0.09 (0.04)	0.31 (0.15)	0.41 (0.14)	0.28 (0.17)	
	false structures	0.10 (0.06)	0.33 (0.18)	0.47 (0.22)	0.32 (0.18)	
	$\tau_{\text{bin}}^{\max}$	**0.08 (0.05)**	**0.30 (0.17)**	**0.28 (0.15)**	**0.26 (0.13)**	0.062 (0.02)
		0.09 (0.07)	0.32 (0.18)	0.33 (0.14)	0.31 (0.18)	
		0.09 (0.06)	0.35 (0.22)	0.31 (0.15)	0.32 (0.16)	
	false structures	0.09 (0.06)	0.33 (0.18)	0.36 (0.18)	0.32 (0.16)	
	$\tau_{\text{bin}}^{\text{avg}}$	**0.07 (0.04)**	**0.29 (0.16)**	**0.18 (0.07)**	**0.26 (0.13)**	0.06 (0.001)
		0.07 (0.04)	0.31 (0.2)	0.20 (0.07)	0.29 (0.14)	
		0.07 (0.04)	0.30 (0.16)	0.19 (0.05)	0.26 (0.15)	
	false structures	0.07 (0.04)	0.29 (0.17)	0.20 (0.08)	0.26 (0.14)	
6	θ_{bin}	0.40 (0.22)	0.72 (0.4)	1.99 (0.4)	0.87 (0.4)	0.127 (0.026)
		0.13 (0.09)	0.57 (0.28)	1.74 (0.5)	0.72 (0.3)	
		0.19 (0.16)	**0.51 (0.26)**	**1.20 (0.3)**	**0.49 (0.23)**	
	false structures	0.32 (0.23)	0.67 (0.4)	1.92 (0.4)	0.82 (0.4)	
	θ_{RML}	0.09 (0.08)	0.36 (0.23)	0.31 (0.14)	0.31 (0.16)	1.5 (0.7)
		0.09 (0.08)	**0.34 (0.21)**	**0.22 (0.09)**	**0.28 (0.14)**	
		0.10 (0.06)	0.33 (0.19)	0.21 (0.07)	0.26 (0.14)	
	false structures	0.10 (0.08)	0.37 (0.24)	0.31 (0.14)	0.32 (0.16)	
	τ_{bin}	0.21 (0.13)	0.39 (0.23)	0.65 (0.17)	0.45 (0.19)	0.312 (0.007)
		0.19 (0.12)	0.36 (0.2)	0.65 (0.18)	0.41 (0.17)	
		0.17 (0.1)	**0.36 (0.18)**	**0.69 (0.18)**	**0.44 (0.15)**	
	false structures	0.20 (0.13)	0.38 (0.22)	0.65 (0.18)	0.43 (0.18)	
	$\tau_{\text{bin}}^{\min}$	**0.10 (0.07)**	**0.34 (0.21)**	**0.34 (0.14)**	**0.31 (0.16)**	0.09 (0.002)
		0.11 (0.07)	0.35 (0.18)	0.35 (0.12)	0.33 (0.15)	
		0.12 (0.1)	0.38 (0.21)	0.37 (0.14)	0.36 (0.14)	
	false structures	0.10 (0.07)	0.34 (0.19)	0.39 (0.16)	0.33 (0.15)	
	$\tau_{\text{bin}}^{\max}$	**0.08 (0.05)**	**0.32 (0.2)**	**0.27 (0.12)**	**0.29 (0.13)**	0.096 (0.0025)
		0.08 (0.04)	0.30 (0.17)	0.29 (0.11)	0.28 (0.11)	
		0.09 (0.05)	0.33 (0.18)	0.30 (0.12)	0.29 (0.12)	
	false structures	0.09 (0.06)	0.32 (0.18)	0.30 (0.11)	0.29 (0.13)	
	$\tau_{\text{bin}}^{\text{avg}}$	**0.07 (0.04)**	**0.31 (0.19)**	**0.17 (0.05)**	**0.27 (0.13)**	0.093 (0.0021)
		0.07 (0.04)	0.33 (0.18)	0.18 (0.05)	0.28 (0.12)	
		0.08 (0.07)	0.35 (0.19)	0.18 (0.05)	0.30 (0.12)	
	false structures	0.07 (0.05)	0.31 (0.17)	0.18 (0.05)	0.28 (0.12)	

Table 5 The second part of the results for the copula models for $d \in \{7, 9\}$. The columns contain: method denotation; GoF test statistics ${}_dS_n$, ${}_2S_n^{\max}$, ${}_2S_n^{(K)\max}$, ${}_2S_n^{(C)\max}$; the average estimation time of one estimation process in s. The values corresponding to the true structure are in bold. The values in parenthesis are the corresponding standard deviations. The last row for each dimension and each method, denoted by *false structures* in the second column, shows averages of the considered statistics over all estimates with structures different to the true structure.

d	Method	${}_dS_n$	${}_2S_n^{\max}$	${}_2S_n^{(K)\max}$	${}_2S_n^{(C)\max}$	time (in s)
7	θ_{bin}	0.14 (0.06)	0.80 (0.3)	3.01 (0.5)	0.86 (0.3)	0.190 (0.028)
		0.16 (0.15)	**0.51 (0.27)**	**0.89 (0.4)**	**0.49 (0.23)**	
		0.13 (0.04)	0.74 (0.28)	3.04 (0.5)	0.81 (0.28)	
	false structures	0.15 (0.06)	0.81 (0.3)	3.02 (0.6)	0.87 (0.4)	
	θ_{RML}	0.07 (0.05)	0.42 (0.2)	0.53 (0.2)	0.39 (0.17)	7.4 (8)
		0.07 (0.05)	0.43 (0.21)	0.66 (0.29)	0.42 (0.18)	
		0.07 (0.05)	0.45 (0.22)	0.65 (0.27)	0.42 (0.18)	
		0.07 (0.04)	**0.34 (0.01)**	**0.34 (0.14)**	**0.26 (0.06)**	
	false structures	0.07 (0.05)	0.44 (0.21)	0.59 (0.24)	0.41 (0.18)	
	τ_{bin}	**0.40 (0.16)**	**0.62 (0.27)**	**2.07 (0.4)**	**0.80 (0.23)**	0.470 (0.009)
		0.33 (0.16)	0.56 (0.3)	1.43 (0.26)	0.65 (0.26)	
		0.41 (0.16)	0.64 (0.3)	2.03 (0.4)	0.84 (0.28)	
	false structures	0.36 (0.16)	0.59 (0.29)	1.75 (0.4)	0.74 (0.28)	
	$\tau_{\text{bin}}^{\min}$	**0.10 (0.06)**	**0.38 (0.2)**	**0.53 (0.22)**	**0.35 (0.17)**	0.128 (0.003)
		0.09 (0.05)	0.36 (0.13)	0.59 (0.2)	0.33 (0.12)	
		0.11 (0.07)	0.43 (0.18)	0.67 (0.3)	0.38 (0.16)	
	false structures	0.10 (0.06)	0.39 (0.16)	0.63 (0.3)	0.37 (0.16)	
	$\tau_{\text{bin}}^{\max}$	**0.07 (0.05)**	**0.36 (0.2)**	**0.43 (0.19)**	**0.34 (0.16)**	0.129 (0.003)
		0.09 (0.05)	0.46 (0.23)	0.50 (0.22)	0.43 (0.15)	
		0.07 (0.04)	0.33 (0.11)	0.54 (0.2)	0.34 (0.11)	
	false structures	0.08 (0.05)	0.37 (0.2)	0.50 (0.21)	0.35 (0.16)	
	$\tau_{\text{bin}}^{\text{avg}}$	**0.04 (0.025)**	**0.33 (0.18)**	**0.22 (0.08)**	**0.29 (0.13)**	0.135 (0.004)
		0.04 (0.018)	0.32 (0.15)	0.23 (0.06)	0.28 (0.11)	
		0.05 (0.028)	0.36 (0.15)	0.25 (0.1)	0.29 (0.12)	
	false structures	0.05 (0.02)	0.33 (0.14)	0.24 (0.08)	0.29 (0.12)	
9	θ_{bin}	0.08 (0.05)	0.71 (0.3)	1.71 (0.5)	0.79 (0.28)	0.467 (0.028)
		0.12 (0.05)	0.98 (0.4)	3.61 (0.6)	1.04 (0.4)	
		0.13 (0.04)	0.99 (0.4)	3.79 (0.5)	1.05 (0.4)	
	false structures	0.10 (0.06)	0.79 (0.4)	2.32 (1.2)	0.87 (0.3)	
	τ_{bin}	0.53 (0.19)	0.75 (0.3)	2.52 (0.4)	0.99 (0.3)	0.726 (0.011)
		0.51 (0.16)	0.71 (0.29)	2.65 (0.5)	1.01 (0.26)	
		0.46 (0.14)	0.65 (0.28)	1.96 (0.3)	0.82 (0.23)	
		0.51 (0.18)	**0.72 (0.3)**	**2.60 (0.5)**	**0.98 (0.3)**	
	false structures	0.49 (0.17)	0.71 (0.3)	2.22 (0.5)	0.92 (0.29)	
	$\tau_{\text{bin}}^{\min}$	**0.10 (0.05)**	**0.44 (0.14)**	**0.66 (0.21)**	**0.42 (0.12)**	0.195 (0.004)
		0.09 (0.06)	0.42 (0.23)	0.65 (0.26)	0.39 (0.2)	
		0.10 (0.05)	0.44 (0.14)	0.66 (0.21)	0.42 (0.12)	
	false structures	0.10 (0.06)	0.44 (0.2)	0.71 (0.26)	0.42 (0.19)	
	$\tau_{\text{bin}}^{\max}$	**0.07 (0.05)**	**0.41 (0.2)**	**0.54 (0.21)**	**0.38 (0.16)**	0.198 (0.004)
		0.07 (0.03)	0.40 (0.18)	0.51 (0.22)	0.40 (0.14)	
		0.07 (0.05)	0.41 (0.2)	0.54 (0.21)	0.38 (0.16)	
	false structures	0.08 (0.05)	0.43 (0.2)	0.57 (0.23)	0.40 (0.16)	
	$\tau_{\text{bin}}^{\text{avg}}$	**0.03 (0.02)**	**0.38 (0.18)**	**0.25 (0.08)**	**0.33 (0.13)**	0.205 (0.013)
		0.03 (0.02)	0.37 (0.19)	0.27 (0.08)	0.32 (0.16)	
		0.04 (0.02)	0.40 (0.17)	0.26 (0.07)	0.35 (0.13)	
	false structures	0.04 (0.017)	0.39 (0.18)	0.27 (0.08)	0.34 (0.13)	

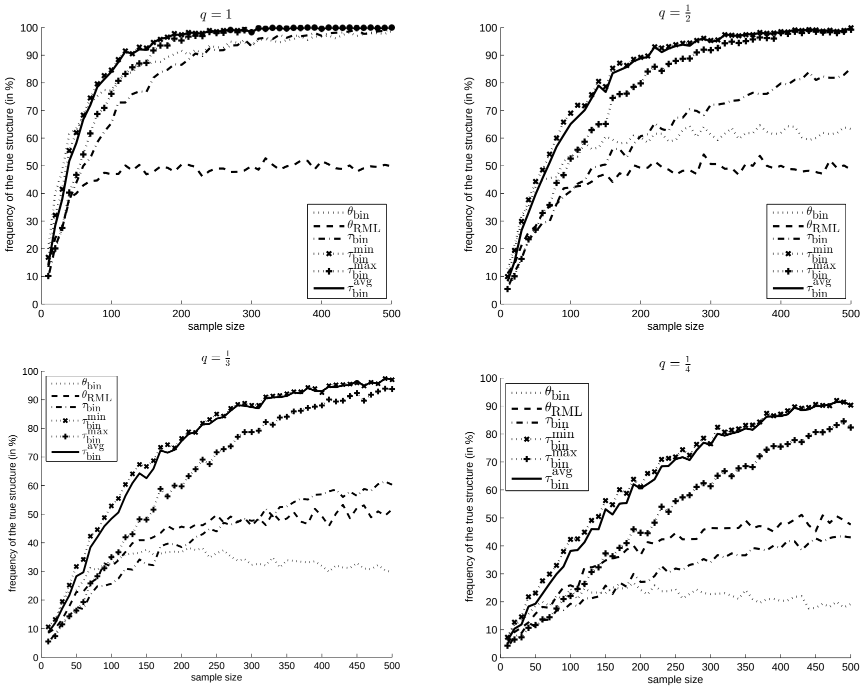

Fig. 3 The frequency of the true structure in 1000 estimated structures for the considered methods, for sample sizes $10, 20, ..., 500$ and for the differences in the parameters set consecutively to $1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}$, i.e., for four 5-HAC models $((12)_{3*q}(3(45)_{4*q})_{3*q})_{2*q}$ with $q = 1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}$, respectively.

method dominates in GoF in all four dimensions. The methods θ_{RML} and $\tau_{\mathrm{bin}}^{\mathrm{max}}$ show good results as well, but the time consumption of θ_{RML} for comparable results is considerably higher (especially for $d = 7$). A surprising result shows the $\tau_{\mathrm{bin}}^{\mathrm{min}}$ method. Despite it shows very good ability in estimating the true structure, it is ranked as the third best in GoF, i.e., it shows the results opposite to $\tau_{\mathrm{bin}}^{\mathrm{max}}$, which is very good in GoF but is ranked as the third in the ability to estimate the true structure. Here, it is worth to note that $\tau_{\mathrm{bin}}^{\mathrm{avg}}$ performs very good both in the ability to determine the true structure and in GoF. The remaining methods show poor results, what is additionally illustrated by the discrepancy between the estimated average parameter values shown in the fourth column in Tables 2 and 3 and the true parameter values.

The last row for each dimension and each method in Tables 4 and 5, denoted by *false structures* in the second column, shows averages of the considered statistics over all estimates with structures different to the true structure, say *false structured* estimates. These results allow for studying the performance of the methods when the true structure is misspecified. Comparing the results for the false structured estimates among the considered methods, we observe that the $\tau_{\mathrm{bin}}^{\mathrm{avg}}$ method shows the lowest values for each dimension and statistic considered. The second and the third lowest values show alternately the θ_{RML} and $\tau_{\mathrm{bin}}^{\mathrm{max}}$ methods for each $d \in \{5, 6, 7\}$ and statistics considered. The fourth lowest values mostly shows the

$\tau_{\mathrm{bin}}^{\mathrm{min}}$ method. For the remaining two methods, the results are varying. Summarizing these results, a false structured HAC estimate fits the best to the data if it is obtained by the $\tau_{\mathrm{bin}}^{\mathrm{avg}}$ method.

The last column in Tables 4 and 5 shows the average computing time needed for a single estimation process. In this case, $\tau_{\mathrm{bin}}^{\mathrm{min}}$, $\tau_{\mathrm{bin}}^{\mathrm{max}}$, $\tau_{\mathrm{bin}}^{\mathrm{avg}}$ show similar results that are slightly better than the binary methods θ_{bin}, τ_{bin}, whereas θ_{RML} shows substantially (several times) higher time consumption, particularly for $d \geq 6$.

Based on all experimental results presented in this section, we can rank the presented methods as follows:

1. the $\tau_{\mathrm{bin}}^{\mathrm{avg}}$ method only. We can claim that this method is the clear winner out of all here presented methods. It shows the best results in goodness-of-fit, even for the cases when the true structure is not determined; it is also one of the two best methods (together with $\tau_{\mathrm{bin}}^{\mathrm{min}}$) in the evaluation of the ability to determine the true structure, including the analysis of this ability for different sample sizes; it offers comparably low run-time (together with $\tau_{\mathrm{bin}}^{\mathrm{min}}$ and $\tau_{\mathrm{bin}}^{\mathrm{max}}$); its stability in structure determination increases in d if compared to the remaining methods.
2. the methods $\tau_{\mathrm{bin}}^{\mathrm{min}}$, $\tau_{\mathrm{bin}}^{\mathrm{max}}$. These methods show in some comparisons results similar to $\tau_{\mathrm{bin}}^{\mathrm{avg}}$, e.g., $\tau_{\mathrm{bin}}^{\mathrm{min}}$ in the ability to determine the true structure, however, in other comparisons, e.g., in goodness-of-fit, these methods show worse results than $\tau_{\mathrm{bin}}^{\mathrm{avg}}$.
3. the θ_{RML} method only. This method shows, on the one hand, comparably good results in goodness-of-fit (mostly similar to $\tau_{\mathrm{bin}}^{\mathrm{max}}$), on the other hand, it show poor results in the ability to determine the true structure, particularly when analyzed for the different sample sizes, and its run-time is substantially higher than the run-time of all other considered methods.
4. the methods θ_{bin} and τ_{bin}. These methods score poorly in most of the presented comparisons.

Note that a similar experiment was reported in [20], where $N = 100$ was used instead. Comparing the results of both experiments, we see that they are almost the same for $d = 5, 6$. For the two higher dimensions $d = 7, 9$, the results show several rather smaller differences, mostly for rarely occurring estimated structures. Considering the $\tau_{\mathrm{bin}}^{\mathrm{avg}}$ method, the results in both experiments for the same statistics considered, i.e., ${}_2S_n^{(K)\max}$, ${}_2S_n^{(C)\max}$ (denoted by $S_n^{(K)}, S_n^{(C)}$, respectively, in [20]) and frequencies of the 3 most frequent estimated structures, are almost the same for the considered dimensions.

5 Copula-based Bayesian classification

5.1 Construction of copula-based Bayesian classifiers

Bayesian classifiers belong to the most popular classifiers and are used for pattern recognition in several image processing, statistical learning and data mining applications. Here we briefly recall some basics for Bayesian classifiers and a way how copulas could be integrated in them as proposed in [47]. Later we describe experiments that involve Bayesian classifiers based on Gaussian copulas, ACs or

HACs. Note that we introduce Bayesian classifiers based on ACs and HACs here for the first time.

Let $\Omega = \{\omega_1, ..., \omega_m\}$ be a finite set of m classes. The problem of *classification* is to assign each $\mathbf{x}$ from the variable space $\mathbb{R}^d$ to a class from Ω. A Bayesian classifier is said to assign $\mathbf{x}$ to the class ω_i if,

$$g_i(\mathbf{x}) > g_j(\mathbf{x}) \qquad \text{for all } j \neq i, \tag{28}$$

where $g_i : \mathbb{R}^d \mapsto \mathbb{R}, i = 1, ..., m$ are known as *discriminant functions*, [47], defined by

$$g_i(\mathbf{x}) = \mathbb{P}(\omega_i|\mathbf{x}) = \frac{f(\mathbf{x}|\omega_i)\mathbb{P}(\omega_i)}{\sum_{j=1}^{m} f(\mathbf{x}|\omega_j)\mathbb{P}(\omega_j)}. \tag{29}$$

Here, $f : \mathbb{R}^d \mapsto [0, \infty)$ is a probability density function (pdf) and $\mathbb{P}(\omega_i), i = 1, ..., m$ are the prior probabilities of the classes from Ω . Since any monotonically increasing function $Q : \mathbb{R} \to \mathbb{R}$ keeps the classification unaltered, the discriminant functions can be simplified by $g_i := Q \circ g_i$ with $Q(t) = \ln(t \sum_{j=1}^{m} f(\mathbf{x}|\omega_j)\ \mathbb{P}(\omega_j))$ from (29) to

$$g_i(\mathbf{x}) = \ln f(\mathbf{x}|\omega_i) + \ln \mathbb{P}(\omega_i). \tag{30}$$

If $f(\mathbf{x}|\omega_i)$ is assumed to be, e.g., a Gaussian pdf (leading to the *normal Bayesian* classifier [9, p. 242]), all the margins are distributed according to the same type of distribution. It follows that the corresponding classifier does not accurately classify samples with marginal distributions of different types. This drawback can be addressed by assuming the variables to be independent. This assumption, which leads to the *Naive Bayesian* classifier [9, p. 241], does not impose any restrictions on the margins. However, if there exists dependence among the variables, the Naive Bayesian classifier is also inappropriate for the task. An elegant solution that overcomes the drawbacks of both mentioned approaches can be achieved by bringing copulas into play.

Provided H in (1) is an absolutely continuous multivariate distribution function with marginals $F_1, ..., F_d$, the pdf f of H can be expressed as

$$f(x_1, ..., x_d) = c(F_1(x_1), ..., F_d(x_d)) \prod_{k=1}^{d} f_k(x_k), \tag{31}$$

where $c(u_1, ..., u_d) = \frac{\partial^d C(u_1, ..., u_d)}{\partial u_1 ... \partial u_d}$ denotes the density of the copula $C(u_1, ..., u_d)$ and f_k denotes the density of $F_k,\ k = 1, ..., d$. Returning to (30), $f(\mathbf{x}|\omega_i)$ can then be rewritten as

$$f(\mathbf{x}|\omega_i) = c(F_1(x_1|\omega_i), ..., F_d(x_d|\omega_i)|\omega_i) \prod_{k=1}^{d} f_k(x_k|\omega_i), \tag{32}$$

which turns (30) into

$$g_i(\mathbf{x}) = \ln\big(c(F_1(x_1|\omega_i), ..., F_d(x_d|\omega_i)|\omega_i))\big) + \sum_{k=1}^{d} \ln(f_k(x_k|\omega_i)) + \ln(\mathbb{P}(\omega_i)). \tag{33}$$

In this way, the discriminant function g_i is represented using three ingredients: the conditional copula density $c(\cdot|\omega_i)$, the conditional marginal densities $f_1(\cdot|\omega_i), ..., f_d(\cdot|\omega_i)$, and the prior probability $\mathbb{P}(\omega_i)$. These ingredients do not impose any restrictions on each other, hence, any assumption made on the dependence structure represented by the copula density $c(\cdot|\omega_i)$ is unrelated to assumptions made on the marginal distributions $f_1(\cdot|\omega_i), ..., f_d(\cdot|\omega_i)$. This flexibility overcomes the mentioned drawbacks of the normal and the Naive Bayesian classifier, which is also confirmed by the experimental results presented in Section 5.2.

The training of such a copula-based Bayesian classifier can be performed for each class $\omega_i, i = 1, ...m$, separately as follows. Let $\mathbb{X}^i$ be training data corresponding to the class ω_i. Compute parametric or non-parametric estimates $\hat{F}_1(\cdot|\omega_i), ..., \hat{F}_d(\cdot|\omega_i)$ based on $\mathbb{X}^i$. Compute a parametric or non-parametric estimate $\hat{C}(\cdot|\omega_i)$ based on $\mathbb{X}^i$. Compute an estimate $\hat{\mathbb{P}}(\omega_i)$ of $\mathbb{P}(\omega_i)$ as the proportion of the class ω_i in the training data $\{\mathbb{X}^1, ..., \mathbb{X}^m\}$. The triplet $(\hat{C}(\cdot|\omega_i);\ \hat{F}_1(\cdot|\omega_i), ..., \hat{F}_d(\cdot|\omega_i);\ \hat{\mathbb{P}}(\omega_i))$ uniquely determines the discriminant function g_i.

5.2 Evaluation of the accuracy of copula-based Bayesian classifiers

In what follows, we evaluate the accuracy of such copula-based Bayesian classifiers (CBCs). Note that a similar evaluation study have been conducted only for Gaussian copula-based classifiers (against SVM) and only for simulated data; see [47]. On real-world data, all here presented CBCs are evaluated for the first time.

We construct three types of CBCs, each type involving different classes of copulas:

- a **Gaussian copula-based Bayesian classifier** (GCBC). For any GCBC, it is assumed that $\hat{C}(\cdot|\omega_i)$ is a Gaussian copula. The computation of the estimator of $\hat{C}(\cdot|\omega_i)$ is described in [5] and is implemented by the Matlab's Statistics toolbox function `copulafit`. We used all the arguments of `copulafit` with their default values
- an **AC-based Bayesian classifier** (ACBC). For any ACBC, it is assumed that $\hat{C}(\cdot|\omega_i)$ is an AC. Given a family of generators, the copula parameter is estimated by the inversion of pairwise Kendall's tau, see (10). In our experiments, we used the families listed in Table 1, however, an ACBC is not restricted to them. A family is considered as an input parameter of a ACBC and we selected the family of $\hat{C}(\cdot|\omega_i)$ based on a 10-fold cross-validation. Note that for $d \geq 3$, ACs based on the Laplace-Stieltjes transform generators are generally unable to model negative dependencies [22], i.e., the cases where $\tau_{X,Y} < 0$ for some random variables X and Y. If X and Y are continuous then $\tau_{-X,Y} = \tau_{X,-Y} = -\tau_{X,Y}$. We employ this fact and invert, i.e., $X := -X$, some of the variables to reduce the negative dependence among the variables using Algorithm 4, i.e., in each sample $\mathbb{X}^i, i = 1, ..., m$, we inverted columns corresponding to the indices in $\mathcal{I}$ obtained by Algorithm 4 with Input 1) given $\mathbb{X} := \mathbb{X}^i$. Note that even if we do not have a proof that it is possible to reduce, using this inverting process, the negative dependence to an extent that $\hat{\theta}_n \geq 0$ is satisfied, we were able to get $\hat{\theta}_n \geq 0$ in all performed experiments.
- a **HAC-based Bayesian classifier** (HACBC). For any HACBC, it is assumed that $\hat{C}(\cdot|\omega_i)$ is an HAC. Given a family, the copula estimation is based on the

Algorithm 4 Inverting procedure

```
Input:
1) $\mathbb{X}$ ... a sample from the r.v. $(X_1, ..., X_d)$, $X_i$ is continuous for all $i = 1, ..., d$,
2) denote by $\tau(\mathbb{X})$ the value of $\sum_{1 \leq i < j \leq d} \tau_{ij}^n$, where $(\tau_{ij}^n)$ is a sample version of Kendall correlation matrix computed for $\mathbb{X}$
3) denote by $\mathbb{X}_i^-$ the sample data $\mathbb{X}$ with the $i$-th column inverted
4) $\mathcal{I} = \emptyset$

The inverting procedure:
1. $\tau := -\infty$
while $\tau(\mathbb{X}) > \tau$ do
    2. $\tau := \tau(\mathbb{X})$
    3. $i := \underset{j \in \{1,...,d\}}{\text{argmax}}\ \tau(\mathbb{X}_j^-)$
    4. $\mathcal{I} := \mathcal{I} \cup \{i\}$
    5. $\mathbb{X} := \mathbb{X}_i^-$
end while
6. $\mathcal{I} := \mathcal{I} \backslash \{i\}$ ... remove the last added index

Output:
The set of indices $\mathcal{I}$
```

procedure described in Section 3.2, which is summarized by Algorithm 3. The $\mathbb{I}$-aggregation function g is set to be the average function. The choice of this function is based on the results presented in Section 4. As for ACBCs, we use in our experiments the families listed in Table 1. Which particular among those 3 families to use is considered an input parameter of a HACBC and we selected the family of $\hat{C}(\cdot|\omega_i)$ based on a 10-fold cross-validation. As HACs based on the Laplace-Stieltjes transform generators are also generally unable to model negative dependencies, which is a property they inherit from ACs, we use the same inverting process for the variables as described above for the ACBC type. However, contrarily to the ACBC case, we were sometimes not able to reduce the negative dependence to an extent that $\hat{\theta}_k \geq 0$ for all $k \in \{1, ..., d-1\}$, where $\hat{\theta}_k$ is the parameter estimate computed in Step 2 of Algorithm 3. Consider a Kendall correlation matrix $(\tau_{ij}^n) \in [-1, 1]^{4\times4}$ with $\tau_{12}^n = \tau_{34}^n = 0.5, \tau_{13}^n = \tau_{23}^n = \tau_{24}^n = 0$ and $\tau_{14}^n = -0.1$. The reader can easily see that, whichever variable is inverted or if all variables are left unchanged, the argument of $\tau^{-1}(\cdot)$ in Step 2 of Algorithm 3 is negative at least for one $k \in \{1, 2, 3\}$ providing g is the average function. For the latter case, we would obtain, using Algorithm 3, a 4-PNAC estimate $((12)_{\hat{\theta}_1}(34)_{\hat{\theta}_2})_{\hat{\theta}_3}$, where $\tau(\hat{\theta}_1) = \tau(\hat{\theta}_2) = 0.5$ and $\tau(\hat{\theta}_3) = -0.025$. Due to this fact, we use $\max(0, \hat{\theta}_k)$ instead of $\hat{\theta}_k$ computed in Step 2 of Algorithm 3.

The estimates $\hat{F}_1(\cdot|\omega_i), ..., \hat{F}_d(\cdot|\omega_i)$ of the margins are computed in the same way for all above-mentioned classifiers using the Kernel smoothing function `ksdensity` in Matlab with the parameter `function` set to `cdf`. Note that, if fitting a GCBC, these estimates are also used for transforming the data to [0, 1]. If fitting an ACBC or a HACBC, the transformation of the the data to [0, 1] is not necessary, because the corresponding copula estimation process is based just on the sample version of the Kendall correlation matrix.

These CBCs are compared in terms of accuracy with four non-copula-based classifiers and one copula-based classifier, which are all available in Matlab's Statistical toolbox. These are:

1. the Classification and regression trees method [7], which is implemented by the class `ClassificationTree` and is referred as CART in the following. Each classification tree was first trained to the deepest possible level and then it was pruned to the optimal level, obtained by the function `test`, using the `crossvalidate` method;
2. an ensemble method based on *bagging* of classification trees [6]. The classifier, referred as TREEBAG in the following, is implemented by the function `fitensemble` with its parameters `Method` set to `Bag` and `Learners` set to `ClassificationTree.template('MinLeaf', MinLeaf)`, respectively. In each training phase, we tuned the parameters `NLearn` and `MinLeaf` as they shown to be most influential on the accuracy. From all pairs (`NLearn`, `MinLeaf`) $\in \{1, ..., 200\} \times \{1, ..., 5\}$, we always chose the pair corresponding to the highest accuracy based on a 10-fold cross-validation.
3. an ensemble method based on *boosting* of classification trees [12]. The classifier, referred as ADABOOST in the following, is implemented by the function `fitensemble` with its parameters `Method` set to `AdaBoostM1` (for the datasets with two classes and `AdaBoostM2` for the datasets with three or more classes) and `Learners` set to `ClassificationTree.template('MinLeaf', MinLeaf)`, respectively. In each training phase, we tuned the parameters `NLearn` and `MinLeaf` in the same way as for TREEBAG.
4. a support vector machine [53]. The classifier, referred as SVM in the following, is implemented by the function `smvtrain`. The parameter `KernelFunction` is set to `rbf` as this setting provided the highest accuracy on the considered datasets. In each training phase, we tuned the parameters `boxconstraint` and `rbf_sigma` as they shown to be most influential on the accuracy. The parameters were tuned using unconstrained nonlinear optimization (implemented by the function `fminsearch`) in order to get the maximal accuracy computed based on a 10-fold cross-validation. To search for a global maximum, we always repeated the optimization task 5 times, each time with different initial values of the parameters.
5. the Naive Bayes classifier, which is actually a CBC that assumes independence copulas $\hat{C}(\cdot|\omega_i), i = 1, ..., m$ and is referred as NAIVE in the following. We used the implementation by the function `fitNaiveBayes` and in each training phase, we tuned the parameter `Distribution`. Its value (`normal` or `kernel`) was chosen based on a 10-fold cross-validation. Default parameters are used otherwise.

All in all, we evaluate 8 classifiers on 6 commonly known datasets obtained from the UCI Repository [3], namely on Iris (4 variables, i.e., $d = 4$), BankNote (4 variables), Vertebral (6 variables), Seeds (7 variables), BreastTissue (9 variables), and Wine (13 variables), as well as on the dataset Appendicitis (7 variables) from the KEEL-dataset repository [2], and on one dataset from a recent real-world application in catalysis [41] (we refer to the last dataset as Catalysis), which contains 4 variables. The variables in the Catalysis dataset are proportions of oxides of the metals La, Pt, Ag, Au used during the conversion of methane and ammonia to hydrocyanic (HCN) acid [41]. As most of the UCI and the KEEL datasets contain 3 classes, we have created arbitrarily 3 equi-frequent classes (low, medium, high) also for the Catalysis dataset using the continuous output variable HCN yield. These datasets are selected in order to every considered classifier could be applicable to every dataset. Particularly, as CBCs require continuous input variables, all datasets include only such input variables. Moreover, as using HACBC classifiers is challenging in higher dimensions as described below in detail, we preferred low-dimensional datasets.

The accuracy computation for a given classifier and a given dataset is based on a 10-fold cross-validation and repeated 10 times, more precisely, each classifier except GCBC was tuned and trained 100 times and each tuning of its parameter(s) involved another "inner" 10-fold cross-validation, by which we refer to the cross-validation that is mentioned in the description of the classifier. All computations were performed in Matlab on a PC with Intel Core 2.3 GHz CPU and 4GB RAM.

Here we must mention the most serious restriction we faced when using a HACBC. Such classifier relies on discriminant functions $g_i, 1, ..., m$ given by (33), each involving the density of a HAC estimate $\hat{C}(\cdot; \omega_i)$. To assign new data to one of the m classes, d partial derivatives for each $\hat{C}(\cdot; \omega_i)$ have to be evaluated. Consider that complexity of such a density function quickly grows in d, which cause that the time consumption of its evaluation exceeds reasonable limits already for $d = 5$, particularly for families with a more complex generator, e.g., for the Frank family. Note that this problem is similar to the problem of computation of the statistic $S_n^{(C)}$ mentioned in Section 2.5. To be able to evaluate our experiments in reasonable time, we thus projected all datasets to $d = 4$, i.e., before any evaluation of all classifiers on a dataset had started, we performed the feature selection and selected only 4 variables from the dataset. With such a comparison of the classifiers on such low-dimensional data presented below in Section 5.3, we are able to demonstrate capabilities of CBCs, particularly capabilities of HACBCs, when compared to other well-known classifiers.

However, we are aware of the fact that such a comparison is too limited from the practical point of view and it discriminates against the classifiers that easily scale up to high dimensions. For this reason, we provide another comparison presented below in Section 5.4, where all the datasets are considered in their original dimension. However, due to the above-mentioned reasons, such an evaluation would not be viable for HACBCs for the datasets with $d > 4$, hence, we again involve the feature selection, which is, in contrast to the first comparison, performed on training data as a part of the training phase of a HACBC just before tuning of its parameter. With this comparison, we aim to demonstrate applicability of a HACBCs for data with $d > 4$ provided we deal with the above-mentioned restriction using the feature selection.

Note that the feature selection was performed using the function `sequentialfs` and we based the selection process on the discriminant analysis [36] implemented by the function `classify`. The reason for choosing the discriminant analysis, i.e., a classifier that is different from all the evaluated classifiers, is that we tried not to favour any of the evaluated classifiers. The feature selection process is indeed performed for Appendicitis, BreastTissue, Seeds, Vertebral and Wine datasets. As the Iris, BankNote and Catalysis datasets have all the dimension $d = 4$, evaluation for these datasets does not involve the feature selection process and we include it in both above-mentioned comparisons.

It is also important to note that the evaluation presented here is not meant to be an exhaustive study of possibilities of CBCs. Rather, this study should be

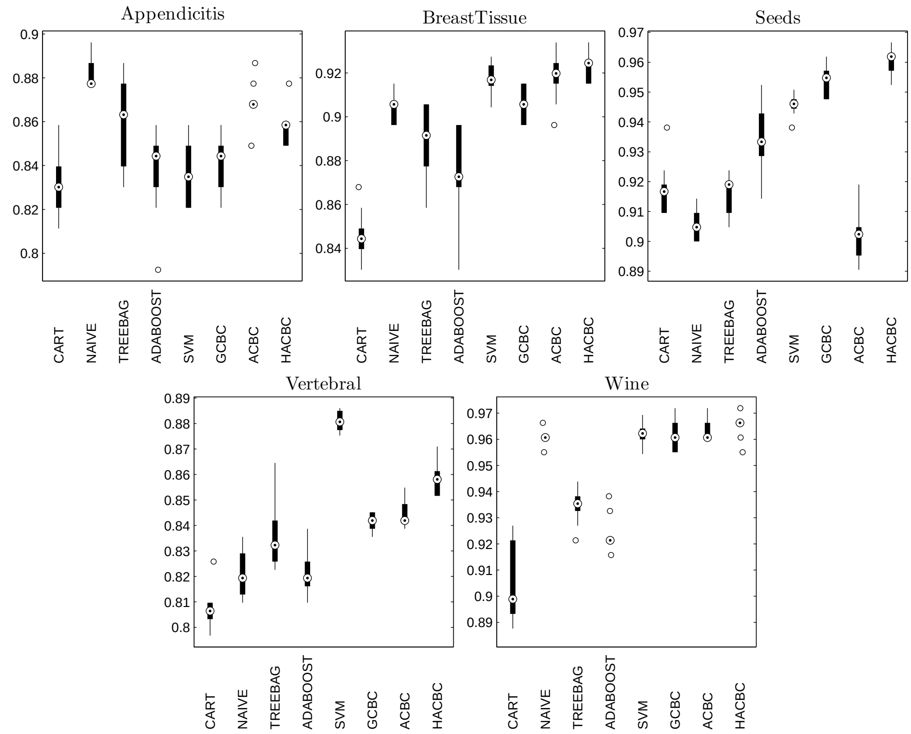

Fig. 4 The accuracy of the classifiers measured on 4-dimensional projections of the Appendicitis, BreastTissue, Seeds, Vertebral and Wine datasets.

viewed as a first example considering applicability of ACs and HACs in Bayesian classification, which shows that such classifiers, despite the above-mentioned restriction, provide simplicity and accuracy, as discussed below.

5.3 The first comparison (all datasets projected to $d = 4$ dimensions)

The accuracy of the classifiers computed on the datasets projected to $d = 4$ dimensions using the feature selection is shown in Figures 4 and 5. It can be observed that there is not a clear winning classifier on all the datasets, what is not surprising in the context of the “No Free Lunch Theorem” [55]. However, some of the classifiers score higher substantially more often then the others. This observation is supported by the rankings of the classifiers in Table 6.

Each of classifiers is ranked according to its averaged accuracy: 1 stands for the highest and 8 stand for the lowest averaged accuracy on the given dataset. Observing the averages of these ranks – the average rank row in Table 6 – four groups of the classifiers can be distinguished:

- the highest-ranked group - SVM (average rank = 2.875) and HACBC (2.875);
- the middle-high-ranked group - GCBC (3.625) and ACBC (3.875);

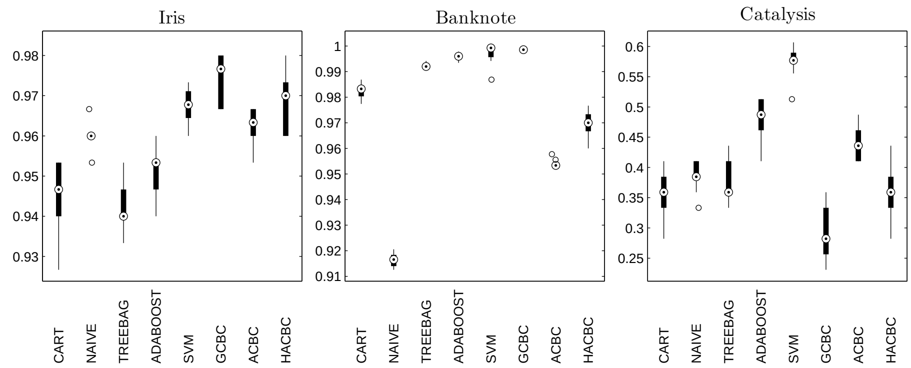

Fig. 5 The accuracy (boxplot) of the classifiers measured on the Iris, BankNote and Catalysis datasets.

Table 6 Rankings of the classifiers in the *first* comparison according to the averaged accuracy on a given (1st column) dataset. The top-three ranks are in bold. The penultimate row shows the average rank of a classifier. The last row shows how many times a classifier is ranked in the top-three.

classifier	CART	NAIVE	TREEBAG	ADABOOST	SVM	GCBC	ACBC	HACBC
Iris	7	5	8	6	**3**	**1**	4	**2**
BankNote	5	8	4	**3**	**2**	**1**	7	6
Catalysis	7	4	5	**2**	**1**	8	**3**	6
Appendicitis	8	**1**	**3**	6	7	5	**2**	4
BreastTissue	8	5	6	7	**3**	4	**2**	**1**
Seeds	5	7	6	4	**3**	**2**	8	**1**
Vertebral	8	7	5	6	**1**	4	**3**	**2**
Wine	8	5	6	7	**3**	4	**2**	**1**
average rank	7	5.25	5.375	5.125	2.875	3.625	3.875	2.875
# top-three	0	1	1	2	7	3	5	5

– the middle-low-ranked group - NAIVE (5.25), TREEBAG (5.375) and ADABOOST (5.125);
– the lowest-ranked group - CART (7).

This high-low ranking is also supported by another ranking – the top-three ranking, which counts how many times a classifier is ranked among the three best. We see that the classifiers from the highest-ranked and the middle-high-ranked group reside more frequently in the top-three than the classifiers from the lowest-ranked and the middle-low-ranked group.

If we divide the classifiers into four groups according to their type – 1) simple classifiers (CART and NAIVE), 2) ensemble classifiers (TREEBAG and ADABOOST) 3) SVM 4) CBCs (GCBC, ACBC and HACBC) – we can also observe the superiority of SVM and CBCs to the remaining types of classifiers. This is illustrated by the first two rows of graphs in Figure 7, which show the boxplot of the best 4 (according to the averaged accuracy) classifiers out of these four groups.

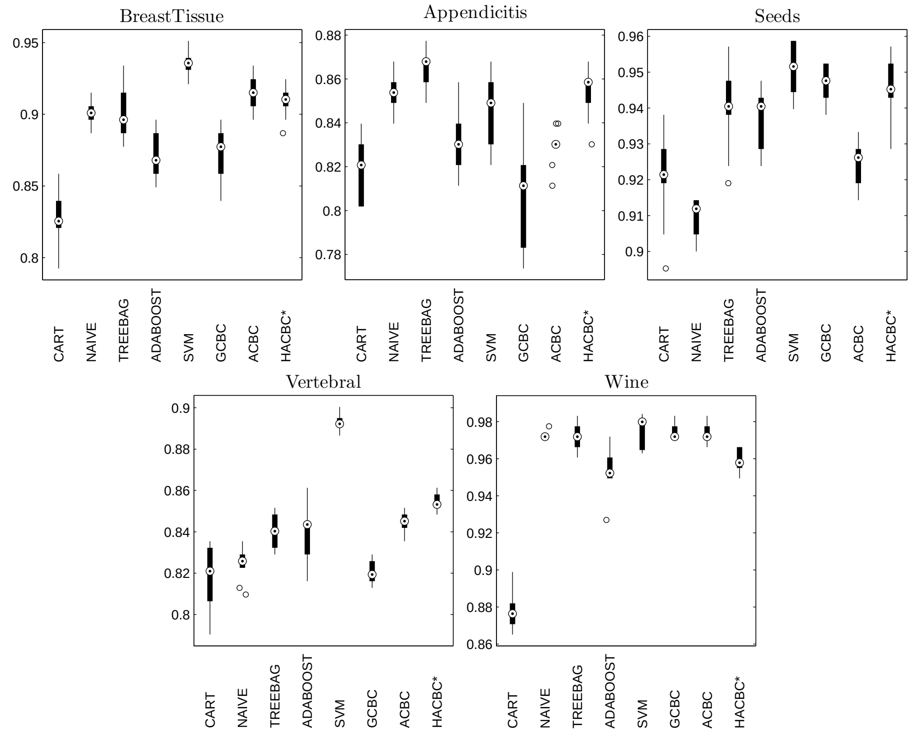

Fig. 6 The accuracy of the classifiers measured on Appendicitis, BreastTissue, Seeds, Vertebral and Wine datasets.. The asterisk for HACBC emphasize that the feature selection was performed for this classifier.

We can see that SVM and the best representative of the CBCs score better than the best representatives of simple and ensemble classifiers on most of the datasets.

5.4 The second comparison (all datasets in their original dimension)

The accuracy of the classifiers computed on the datasets in their original dimension except for the HACBCs is shown in Figures 5 and 6. We again observe that there is no classifier that wins on all the datasets. In contrast to the first comparison, we observe that the difference between the two best ranked classifiers is substantially higher, which is supported by the rankings of the classifiers in Table 7.

Now, observing the averages of the ranks in Table 7, again, four groups of the classifiers similar to the first comparison can be distinguished, however, with one important switch between the first two groups :

– the highest-ranked group - SVM (average rank = 1.875);
– the middle-high-ranked group - GCBC (4.5), ACBC (4) and HACBC (3.75);
– the middle-low-ranked group - NAIVE (5.375), TREEBAG (4.375) and ADABOOST (5);
– the lowest-ranked group - CART (7.125).

Table 7 Rankings of the classifiers in the *second* comparison according to the averaged accuracy on a given (1st column) dataset. The top-three ranks are in bold. The penultimate row shows the average rank of a classifier. The last row shows how many times a classifier is ranked in the top-three. The asterisk for HACBC emphasize that the feature selection was performed for this classifier.

classifier	CART	NAIVE	TREEBAG	ADABOOST	SVM	GCBC	ACBC	HACBC*
Iris	7	5	8	6	**3**	**1**	4	**2**
BankNote	5	8	4	**3**	**2**	**1**	7	6
Catalysis	7	4	5	**2**	**1**	8	**3**	6
Appendicitis	7	**3**	**1**	5	4	8	6	**2**
BreastTissue	8	4	5	7	**1**	6	**2**	**3**
Seeds	7	8	4	5	**1**	**2**	6	**3**
Vertebral	8	6	4	5	**1**	7	**3**	**2**
Wine	8	5	4	7	**2**	**3**	**1**	6
average rank	7.125	5.375	4.375	5	1.875	4.5	4	3.75
# top-three	0	1	1	2	7	4	4	5

We see that HACBC substantially decreased in the ranking and it is now more convenient to put it in the middle-high-ranked group. As addressed before, due to the extreme time consumption of HACBCs in high dimensions, here presented results for these classifiers involve the feature selection, which, on the one hand, considerably influence their accuracy, on the other hand, allows for at least some applicability of HACBCs in higher dimensions. The remaining classifiers show results similar to the first comparison, again supported by the top-three ranking.

The supremacy of the SVM and the CBCs to other types of classifiers is again observable, now illustrated by the second and the third row of the graphs in Figure 7. We again observe that SVM and the best representative of CBCs score better than the best representatives of simple and ensemble classifiers on most of the datasets.

We can conclude that, in these experiments, CBCs and particularly HACBC classifiers have shown to be competitive for low-dimensional data with highly-accurate classifiers like SVM or ensemble methods in terms of accuracy while keeping the produced models rather comprehensible, as also discussed in Section 5.5. If there appears a way how to compute efficiently the density function of a HAC, e.g., as the simplification of the density functions for the five popular AC families presented in [25], it is possible that results similar to the results for the HACBCs in low-dimensions could also be obtained for HACBCs in high-dimensions.

Here, it is important to note that none of the results presented here must be over-generalized and we recall that, when selecting the datasets, we selected the ones with all continuous input variables and we also preferred low-dimensional ones. Hence, the results, e.g., for ensemble methods, which are applicable to much wider classes of data, must be considered with this in mind.

In further research, we will aim to confirm here presented results for the CBCs on substantially larger amount of datasets produced by diverse applications. Moreover, as there exist many other copula classes, e.g., pair copulas [1], skew t-copulas

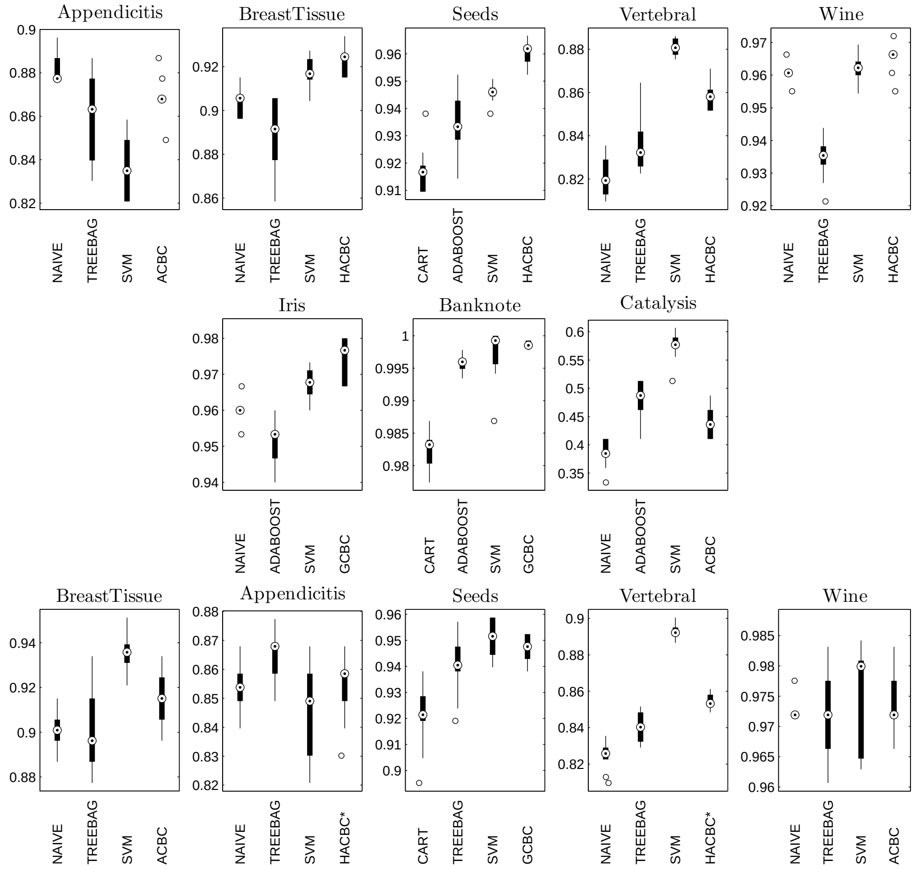

Fig. 7 The accuracy of the representative classifiers on all considered datasets. The first row and the second row of the graphs correspond to the first comparison described in Section 5.3, whereas the second row and the third row of the graphs correspond to the second comparison described in Section 5.4. The asterisk for HACBC emphasize that the feature selection was performed for this classifier.

[52], etc., which could be used for a CBC construction in the same way as for the above-mentioned copula classes, we will also consider these CBC types. To put CBCs more firmly into the framework of commonly used classifiers, CBCs should be compared with other types of classifiers, e.g., neural-networks-based classifiers, K-Nearest Neighbors, etc. Apart from accuracy and simplicity, the classifiers should also be compared in terms of classification run-times and memory usage.

5.5 Note on Accuracy vs Comprehensibility

At this point, we want to consider the typical trade-off between the accuracy and the comprehensibility of a classification model. In most cases, the accuracy of a classification model grows at the expense of its comprehensibility. In our comparison, two easily comprehensible classifiers participate – CART and NAIVE – which, on the one hand, produce easy to understand models and, on the other hand, score lower in the accuracy computed on the selected datasets. In contrast, the highly-accurate classifier SVM produces highly complex models which, however, are not so easy to understand.

From this point of view, CBCs could be, in our opinion, considered as a good trade-off between those two extremes. On the one hand, we observe that the accuracy of CBSs is close to the accuracy of SVM on low-dimensional data, on the other hand, the models produced by the classifiers are also easily interpretable with some knowledge of copulas.

Here we want to emphasize the HACBC classifiers, which produce models of the joint dependency among the variables in the form of a HAC. As we know, a HAC can be expressed as a tree-like graph. As an specific example, see Figure 8. The figure shows the HAC parameters and structure estimates for the classes Setosa, Versicolor and Virginica in the Iris dataset that were obtained using Algorithm 3 with the assumption that all the generators are from the Frank family. The $\hat{\theta}_1, ..., \hat{\theta}_3$ are the parameter values of each HAC estimate. The dendrogram-like representation of the trees has the advantage that, instead of showing only the structure of the HAC, it also visualize the strength of dependency among the variables. This is because each generator node is vertically positioned according the value of the Kendall's tau that corresponds to its parameter. Such a representation enables one (with some knowledge of HACs) to get fuller picture of the dependencies among the variables than the standard HAC tree-like representation. It is worth mentioning that the dependencies also can be obtained from such graphs in a more formal way as sentences of an observational calculus, as recently proposed in [28].

6 Conclusion

We proposed a new approach to structure determination and parameter estimation of hierarchical Archimedean copulas, which combines the advantages of existing methods in terms of the correctly determined structures ratio, the goodness-of-fit of the estimates, and run-time. This has been confirmed in several experiments based on simulated data in different dimensions and copula models. The proposed method is particularly attractive in lower-dimensional ($d \leq 100$) applications where a good approximation and computational efficiency are crucial. However, as the computation of Kendall's tau for all pairs of data columns has complexity $O(d^2 n \log n)$, the approach becomes demanding in high dimensions. Also note that the proposed method restricts to binary structured HACs, i.e., any d-HAC estimate has $d-1$ parameters. In high dimensions, substantially less parameters are often required, hence, a generalization to non-binary structured HACs should also be considered, e.g., in a way proposed in [43].

The presented work does not explicitly consider the following:

1. The proposed method assumes all generators of the estimated HAC to be from the same family, i.e., it assumes that a *homogeneous* HAC results from the estimation process. Despite the possibility of mixing different families in a HAC, see [23], the estimation of such non-homogenous HACs has not been addressed in the literature in detail except in [43], which, however, addresses this issue

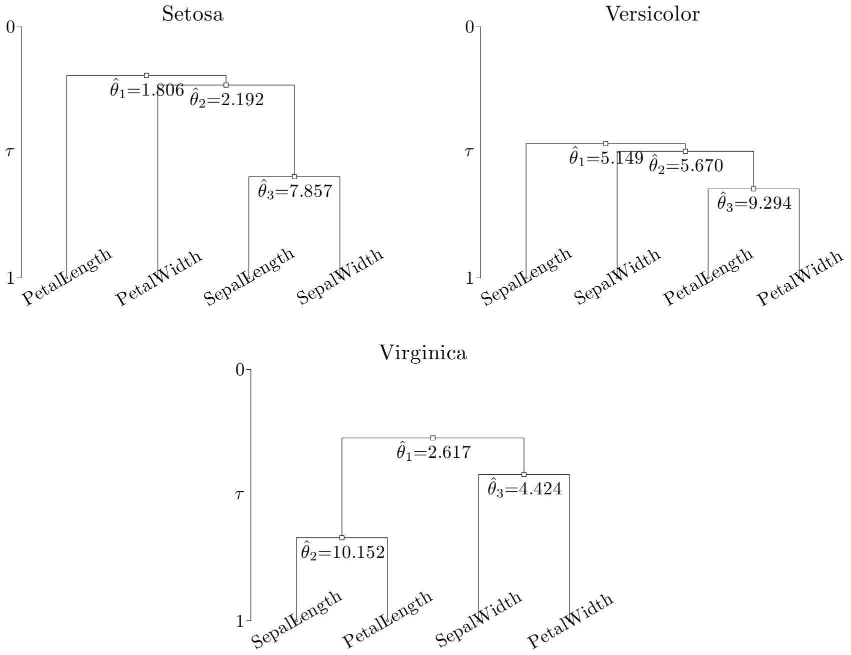

Fig. 8 The HAC estimates based on the Frank generators for the classes Setosa, Versicolor and Virginica in Iris dataset. The θ estimates are the parameters of the generators.

only briefly without any experimental results. From the construction of our estimation method, it becomes clear that it easily extends to non-homogeneous HACs as long as the sufficient nesting condition is fulfilled;

2. Until now, all HAC estimation methods that estimate both the structure and the parameters of a HAC, incorporate either ML estimator or estimator based on the inversion of Kendall's tau. However, there exist also different types of estimation methods, e.g., estimation based on Blomqvists beta, Simulated maximum-likelihood estimation, Minimum distance estimation or Diagonal maximum-likelihood estimation, see [26], which have been originally designed for the estimation of ACs, but could also be considered in HACs estimation. Our estimation method is not restricted to the estimator based on the inversion of Kendall's tau and can easily be extended for using with other estimators like the above-mentioned ones.

Additionally, we applied the proposed method to the construction of copula-based Bayesian classifiers, which are experimentally compared with other types of commonly used classifiers on several real-world datasets. Two types of such classifiers, namely the AC-based and the HAC-based Bayesian classifiers, were evaluated for the first time. Due to the restrictions addressed in Section 5.2, applicability of HAC-based Bayesian classifiers for high-dimensional data is limited, however, the experimental results for low-dimensional data show that these classifiers are competitive with highly-accurate classifiers like SVM or ensemble methods in terms of accuracy while keeping the produced models rather comprehensible.

Acknowledgment

For a BibTex citation of this article, see `http://suzelly.opf.slu.cz/~gorecki/en/research.php`.

References

1. Aas, K., Czado, C., Frigessi, A., Bakken, H.: Pair-copula constructions of multiple dependence. Insurance: Mathematics and Economics **44**(2), 182–198 (2009)
2. Alcalá, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., Herrera, F.: Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. Journal of Multiple-Valued Logic and Soft Computing **17**, 255–287 (2010)
3. Bache, K., Lichman, M.: UCI machine learning repository (2013). URL `http://archive.ics.uci.edu/ml`
4. Berg, D.: Copula goodness-of-fit testing: an overview and power comparison. The European Journal of Finance **15**(7-8), 675–701 (2009)
5. Bouyé, E., Durrleman, V., Nikeghbali, A., Riboulet, G., Roncalli, T.: Copulas for finance - a reading guide and some applications. Available at SSRN 1032533 (2000)
6. Breiman, L.: Bagging predictors. Machine learning **24**(2), 123–140 (1996)
7. Breiman, L., Freidman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth (1984)
8. Chen, X., Fan, Y., Patton, A.J.: Simple tests for models of dependence between multiple financial time series, with applications to us equity returns and exchange rates (2004). Discussion paper 483, Financial Markets Group, London School of Economics
9. Clarke, B., Fokoue, E., Zhang, H.H.: Principles and Theory for Data Mining and Machine Learning. Springer (2009)
10. Cramér, H.: On the composition of elementary errors: First paper: Mathematical deductions. Scandinavian Actuarial Journal **1928**(1), 13–74 (1928)
11. Cuvelier, E., Noirhomme-Fraitur, M.: Clayton copula and mixture decomposition. In: Applied Stochastic Models and Data Analysis, ASMDA'05. Brest (2005)
12. Freund, Y., Schapire, R.E.: A desicion-theoretic generalization of on-line learning and an application to boosting. In: Computational learning theory, pp. 23–37. Springer (1995)
13. Genest, C., Favre, A.: Everything you always wanted to know about copula modeling but were afraid to ask. Hydrol. Eng. **12**, 347–368 (2007)
14. Genest, C., Rémillard, B.: Validity of the parametric bootstrap for goodness-of-fit testing in semiparametric models. In: Annales de l'Institut Henri Poincaré: Probabilités et Statistiques, vol. 44, pp. 1096–1127 (2008)
15. Genest, C., Rémillard, B., Beaudoin, D.: Goodness-of-fit tests for copulas: A review and a power study. Insurance: Mathematics and Economics **44**(2), 199–213 (2009)
16. Genest, C., Rivest, L.P.: Statistical inference procedures for bivariate archimedean copulas. Journal of the American statistical Association **88**(423), 1034–1043 (1993)
17. González-Fernández, Y., Soto, M.: copulaedas: An R package for estimation of distribution algorithms based on copulas. CoRR **abs/1209.5429** (2012)
18. Górecki, J., Hofert, M., Holeňa, M.: On the consistency of an estimator for hierarchical archimedean copulas. In: J. Talašová, J. Stoklasa, T. Talášek (eds.) 32nd International Conference on Mathematical Methods in Economics, pp. 239–244. Palacký University, Olomouc (2014)
19. Górecki, J., Holeňa, M.: An alternative approach to the structure determination of hierarchical Archimedean copulas. Proceedings of the 31st International Conference on Mathematical Methods in Economics (MME 2013), pp. 201 – 206. Jihlava (2013)
20. Górecki, J., Holeňa, M.: Structure determination and estimation of hierarchical Archimedean copulas based on Kendall correlation matrix. In: A. Appice, M. Ceci, C. Loglisci, G. Manco, E. Masciari, Z.W. Ras (eds.) New Frontiers in Mining Complex Patterns, Lecture Notes in Computer Science, pp. 132–147. Springer International Publishing (2014)
21. Hofert, M.: Construction and sampling of nested Archimedean copulas. In: P. Jaworski, F. Durante, W.K. Hardle, T. Rychlik (eds.) Copula Theory and Its Applications, *Lecture Notes in Statistics*, vol. 198, pp. 147–160. Springer Berlin Heidelberg (2010)

22. Hofert, M.: Sampling Nested Archimedean Copulas with Applications to CDO Pricing. Suedwestdeutscher Verlag fuer Hochschulschriften (2010)
23. Hofert, M.: Efficiently sampling nested Archimedean copulas. Computational Statistics and Data Analysis **55**(1), 57–70 (2011)
24. Hofert, M.: A stochastic representation and sampling algorithm for nested Archimedean copulas. Journal of Statistical Computation and Simulation **82**(9), 1239–1255 (2012). DOI http://dx.doi.org/10.1080/00949655.2011.574632
25. Hofert, M., Mächler, M., Mcneil, A.J.: Likelihood inference for archimedean copulas in high dimensions under known margins. Journal of Multivariate Analysis **110**, 133–150 (2012)
26. Hofert, M., Mächler, M., McNeil, A.J.: Archimedean copulas in high dimensions: Estimators and numerical challenges motivated by financial applications. Journal de la Société Française de Statistique **154**(1), 25–63 (2013)
27. Hofert, M., Scherer, M.: CDO pricing with nested Archimedean copulas. Quantitative Finance **11**(5), 775–787 (2011)
28. Holeňa, M., Ščavnický, M.: Application of copulas to data mining based on observational logic. In: ITAT 2013: Information Technologies - Applications and Theory Workshops, Posters, and Tutorials., pp. 77–85. North Charleston : CreateSpace Independent Publishing Platform, Donovaly, Slovakia (2013)
29. Joe, H.: Multivariate Models and Dependence Concepts. Chapman & Hall, London (1997)
30. Kao, S.C., Ganguly, A.R., Steinhaeuser, K.: Motivating complex dependence structures in data mining: A case study with anomaly detection in climate. Data Mining Workshops, International Conference on **0**, 223–230 (2009). DOI http://doi.ieeecomputersociety.org/10.1109/ICDMW.2009.37
31. Kao, S.C., Govindaraju, R.S.: Trivariate statistical analysis of extreme rainfall events via plackett family of copulas. Water Resour. Res. **44** (2008)
32. Kojadinovic, I.: Hierarchical clustering of continuous variables based on the empirical copula process and permutation linkages. Computational Statistics & Data Analysis **54**(1), 90 – 108 (2010)
33. Kojadinovic, I., Yan, J.: Comparison of three semiparametric methods for estimating dependence parameters in copula models. Insurance: Mathematics and Economics **47**, 52–63 (2010)
34. Kojadinovic, I., Yan, J.: Modeling multivariate distributions with continuous margins using the copula r package. Journal of Statistical Software **34**(9), 1–20 (2010)
35. Kuhn, G., Khan, S., Ganguly, A.R., Branstetter, M.L.: Geospatial-temporal dependence among weekly precipitation extremes with applications to observations and climate model simulations in south america. Adv. Water Resour. **30(12)**, 2401–2423 (2007)
36. Lachenbruch, P.A.: Discriminant analysis. Wiley Online Library (1975)
37. Lascio, F., Giannerini, S.: A copula-based algorithm for discovering patterns of dependent observations. Journal of Classification **29**, 50–75 (2012). DOI 10.1007/s00357-012-9099-y. URL `http://dx.doi.org/10.1007/s00357-012-9099-y`
38. Maity, R., Kumar, D.N.: Probabilistic prediction of hydroclimatic variables with nonparametric quantification of uncertainty. J. Geophys. Res. **113** (2008)
39. McNeil, A.J.: Sampling nested Archimedean copulas. Journal of Statistical Computation and Simulation **78**(6), 567–581 (2008)
40. McNeil, A.J., Nešlehová, J.: Multivariate Archimedean copulas, d-monotone functions and l_1-norm symmetric distributions. The Annals of Statistics **37**, 3059–3097 (2009)
41. Moehmel, S., Steinfeldt, N., Engelschalt, S., Holena, M., Kolf, S., Baerns, M., Dingerdissen, U., Wolf, D., Weber, R., Bewersdorf, M.: New catalytic materials for the high-temperature synthesis of hydrocyanic acid from methane and ammonia by high-throughput approach. Applied Catalysis A: General **334**(1), 73–83 (2008)
42. Nelsen, R.: An Introduction to Copulas, 2nd edn. Springer (2006)
43. Okhrin, O., Okhrin, Y., Schmid, W.: On the structure and estimation of hierarchical Archimedean copulas. Journal of Econometrics **173**(2), 189–204 (2013). URL `http://www.sciencedirect.com/science/article/pii/S0304407612002667`
44. Okhrin, O., Okhrin, Y., Schmid, W.: Properties of hierarchical Archimedean copulas. Statistics & Risk Modeling **30**(1), 21–54 (2013)
45. Okhrin, O., Ristig, A.: Hierarchical Archimedean copulae: The HAC package. Journal of Statistical Software **58**(4) (2014). URL `http://www.jstatsoft.org/v58/i04`
46. Rey, M., V., R.: Copula mixture model for dependency-seeking clustering. In: Proceedings of the 29th International Conference on Machine Learning (ICML 2012). Edinburgh, Scotland, UK (2012)

47. Sathe, S.: A novel Bayesian classifier using copula functions. arXiv preprint cs/0611150 (2006)
48. Savu, C., Trede, M.: Goodness-of-fit tests for parametric families of Archimedean copulas. Quantitative Finance **8**(2), 109–116 (2008)
49. Savu, C., Trede, M.: Hierarchies of Archimedean copulas. Quantitative Finance **10**, 295–304 (2010)
50. Segers, J., Uyttendaele, N.: Nonparametric estimation of the tree structure of a nested Archimedean copula. Computational Statistics & Data Analysis **72**, 190–204 (2014)
51. Sklar, A.: Fonctions de répartition a n dimensions et leurs marges. Publ. Inst. Stat. Univ. Paris **8**, 229–231 (1959)
52. Smith, M.S., Gan, Q., Kohn, R.J.: Modelling dependence using skew t copulas: Bayesian inference and applications. Journal of Applied Econometrics **27**(3), 500–522 (2012)
53. Vapnik, V.: The nature of statistical learning theory. springer (2000)
54. Wang, L., Guo, X., J., Z., Hong, Y.: Copula estimation of distribution algorithms based on exchangeable Archimedean copula. International Journal of Computer Applications in Technology **43**, 13 – 20 (2012)
55. Wolpert, D.H.: The supervised learning no-free-lunch theorems. In: Soft Computing and Industry, pp. 25–42. Springer (2002)
56. Yuan, A., Chen, G., Zhou, Z.C., Bonney, G., Rotimi, C.: Gene copy number analysis for family data using semiparametric copula model. Bioinform Biol Insights. **2.**, 343–355 (2008)