"On the estimation of nested Archimedean copulas: A theoretical and an experimental comparison"

Uyttendaele, Nathan

Abstract

A lot of progress regarding estimation of nested Archimedean cop- ulas has been booked since their introduction by Joe (1997). The currently published procedures can be seen as particular cases of two different, more general, approaches. In the first approach, the tree structure of the target nested Archimedean copulas is estimated us- ing hierarchical clustering to get a binary tree and then parts of this binary tree are collapsed according to some strategy. This two-step estimation of the tree structure paves the way for an easy estimation of the generators afterwards. In contrast to the first approach, the sec- ond approach estimates the tree structure free of any concern for the generators. While this is the main strength of this second approach, it is also its main weakness: estimation of the generators afterwards still lacks a solution. In this paper, both approaches are formally explored, detailed explanations and examples are given, as well as results from a performance study...

Document type : *Document de travail (Working Paper)*

I N S T I T U T D E S T A T I S T I Q U E
B I O S T A T I S T I Q U E E T
S C I E N C E S A C T U A R I E L L E S
(I S B A)

UNIVERSITÉ CATHOLIQUE DE LOUVAIN

D I S C U S S I O N
P A P E R

2016/05

On the estimation of nested Archimedean copulas: A theoretical and an experimental comparison

N. UYTTENDAELE

On the estimation of nested Archimedean copulas: A theoretical and an experimental comparison

Nathan Uyttendaele*
na.uytten@gmail.com

January 27, 2016

Abstract

A lot of progress regarding estimation of nested Archimedean copulas has been booked since their introduction by Joe (1997). The currently published procedures can be seen as particular cases of two different, more general, approaches. In the first approach, the tree structure of the target nested Archimedean copulas is estimated using hierarchical clustering to get a binary tree and then parts of this binary tree are collapsed according to some strategy. This two-step estimation of the tree structure paves the way for an easy estimation of the generators afterwards. In contrast to the first approach, the second approach estimates the tree structure free of any concern for the generators. While this is the main strength of this second approach, it is also its main weakness: estimation of the generators afterwards still lacks a solution. In this paper, both approaches are formally explored, detailed explanations and examples are given, as well as results from a performance study where a new way of comparing tree structure estimators is offered. A nested Archimedean copula is also estimated based on exams results from 482 students, and a naive attempt to check the fit is made using principal component analysis.

Keywords: nested Archimedean copulas, hierarchical Archimedean copulas, estimation, hierarchical clustering, rooted tree, structure determination, Kendall's tau, phylogenetics.

*Université catholique de Louvain, Institut de Statistique, Biostatistique et Sciences Actuarielles, Voie du Roman Pays 20, B-1348 Louvain-la-Neuve, Belgium.

1 Introduction

Nested Archimedean copulas (NACs), also called hierarchical Archimedean copulas (HACs), introduced by Joe (1997, pp. 87–89), are a natural generalization of Archimedean copulas. The key feature of an Archimedean copula is its generator, which can be loosely defined as a one-parameter or two-parameter function of a single argument (in the rest of this paper, we will always assume this is a one-parameter function). Nested Archimedean copulas are made up of two parts: a rooted phylogenetic tree and a set of generators. They offer more flexibility for modelling dependencies in a high-dimensional setting while still reducing to Archimedean copulas in simpler cases.

Estimation of nested Archimedean copulas has been the main topic of several papers, see Okhrin et al. (2013a), Segers and Uyttendaele (2014) or Górecki et al. (2015). Segers and Uyttendaele (2014) developed a procedure to estimate the target phylogenetic tree of a NAC. In contrast to what is done by Okhrin et al. (2013a), the procedure does not assume anything about the generators of the target NAC prior to the estimation of its tree structure. However, while the end result of the procedure developed by Okhrin et al. (2013a) is a full NAC, Segers and Uyttendaele (2014) only offer partial estimation of the target NAC, as the end result of their procedure is only a tree structure: what to do next is left to the reader.

Close to what is done by Okhrin et al. (2013a), Górecki et al. (2015) also offer full estimation of the target NAC. Unlike Segers and Uyttendaele (2014) and Okhrin et al. (2013a), however, their procedure can only output binary tree structures, making their tree structure estimator biased for any target NAC such that the related tree structure is not a binary one. Górecki et al. (2015) nonetheless experimentally showed that Okhrin et al. (2013a), which use the Kendall transformation of random variables in order to estimate the parameter of each generator, often get poor estimates. The reason for this is addressed in detail by Górecki et al. (2014), where a correction is proposed[1]. Note that, on their side, Górecki et al. (2015) estimate the parameter of a given generator by averaging the Kendall correlation coefficients related to the pairs of random variables interacting through this generator.

Okhrin et al. (2013a), Segers and Uyttendaele (2014) and Górecki et al. (2015) can all be seen as particular cases of two different, more general, approaches, introduced with detailed explanations and examples in Section 3 and in Section 4. Prior to Section 3 and Section 4, Section 2 gives a short introduction to Archimedean copulas and nested Archimedean copulas as an attempt to help the reader not used to Archimedean and nested Archimedean copulas go through this paper; that section also defines most of the

[1]This correction has already been implemented in Ohkrin's R package HAC.

notation used throughout this paper.

Results of a performance study are then shown in Section 5, where special focus is given on the ability of estimating the tree structure of various target NACs, including a target structure spanned on fifteen random variables. A new estimator given in Section 3 as an example turns out to be the most performant one of those tested.

Finally, in Section 6, before a discussion section, full estimation of a nested Archimedean copula based on exams results from 482 students using a particular case from the first approach is made, as well as a naive attempt to check the fit using principal component analysis.

2 Archimedean and Nested Archimedean copulas in a nutshell

Let $(X_1, \ldots, X_d)$ be a vector of continuous random variables. The copula of this vector is defined as

$$C(u_1, \ldots, u_d) = P(U_1 \leq u_1, \ldots, U_d \leq u_d),$$

where $(U_1, \ldots, U_d) = (F_{X_1}(X_1), \ldots, F_{X_d}(X_d))$, and where $F_{X_1}, \ldots, F_{X_d}$ are the marginal cumulative distribution functions (CDFs) of $X_1, \ldots, X_d$.

Archimedean copulas (ACs) can always be written in closed form as

$$C(u_1, \ldots, u_d) = \psi(\psi^{-1}(u_1) + \cdots + \psi^{-1}(u_d)),$$

where ψ is called the generator and ψ^{-1} is its generalized inverse, with $\psi : [0, \infty) \to [0, 1]$, a convex, non-increasing function such that $\psi(0) = 1$ and $\psi(\infty) = 0$. In order for C to be a d-dimensional copula, the generator is required to be d-monotone on $[0, \infty)$, see McNeil and Nešlehová (2009) for more details.

In the case of Archimedean copulas, $C(u_1, \ldots, u_d)$ is a symmetric function in its arguments and this is why Archimedean copulas are sometimes called *exchangeable*. The result of this exchangeability property is easily seen by plotting a cloud of points generated from a bivariate Archimedean copula: the $y = x$ axis is a clear axis of reflection symmetry for the underlying distribution. For a cloud of points generated from a trivariate Archimedean copula, even more complex symmetries for the underlying trivariate distribution can be observed.

Moreover, given a d-variate Archimedean copula and $m \in \{2, \ldots, d-1\}$, any two m-variate margins from that Archimedean copula describe the same m-variate distribution. For instance, with $m = 3$ and assuming the joint distribution of $(U_1, \ldots, U_{10})$ is an Archimedean copula, the joint distribution of (U_6, U_5, U_3) is equal to the joint distribution of (U_3, U_{10}, U_2) or (U_1, U_4, U_8).

It is clear that, for modelling purposes, this exchangeability property becomes an increasingly strong assumption as the dimension d grows.

This exchangeability property can be relaxed using nested Archimedean copulas. NACs are obtained by plugging in Archimedean copulas into each other (Joe, 1997, pp. 87–89). The following example shows how a bivariate Archimedean copula C_{23} can be plugged into a bivariate Archimedean copula C_{123}:

$$C_{123}(u_1, \boldsymbol{C_{23}(u_2, u_3)}) = \psi_{123}\big(\psi_{123}^{-1}(u_1) + \psi_{123}^{-1}(\boldsymbol{\psi_{23}(\psi_{23}^{-1}(u_2) + \psi_{23}^{-1}(u_3))})\big) \quad (2.1)$$

The above trivariate copula, a nested Archimedean copula on (U_1, U_2, U_3), is still such that the $y = x$ axis remains an axis of reflection symmetry for all bivariate margins. However, while even more complex symmetries are observed on the trivariate level of an AC, part of these symmetries are lost on the trivariate level of the NAC described by (2.1). Moreover, while all bivariate margins are the same in an AC, the distribution of (U_2, U_3) is not the same as the distribution of (U_1, U_2) or (U_1, U_3) in the NAC described by (2.1). All these remarks hold provided the generators ψ_{123} and ψ_{23} are different, for otherwise (2.1) can be simplified back to a trivariate AC. In general, the more the generators of successive nodes from a NAC are different, the more that NAC will be said to be *resolved*, a word mainly used in Section 5. Poorly resolved NACs are almost ACs, and their estimation is, in general, harder.

In a NAC, the way Archimedean copulas are nested corresponds to a rooted phylogenetic tree λ. This rooted phylogenetic tree is such that any internal node, an internal node being any node different from a leaf, must have at least two children and every node but the root must have one and only one parent. Rooted trees where each internal node has two and only two children are a subset of the set of rooted phylogenetic trees and are called binary trees.

Nested Archimedean copulas, such as the one in (2.1) and for which the tree can be seen on the left panel of Figure 1, are defined through a rooted phylogenetic tree and through a collection of generators Ψ, one generator for each internal node in the tree. Each generator fully describes the dependence between the random variables interacting through the related internal node. Archimedean copulas can be seen as a special case of NACs: they exhibit a trivial structure such as the one on the right-hand panel of Figure 1 and have only one generator, the one related to the only internal node, the root. A trivial tree structure of dimension d is sometimes called a d-fan (Ng and Wormald, 1996).

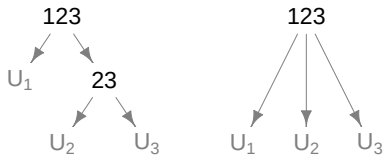


Figure 1: Left: the tree structure implied by (2.1). The dependence between U_2 and U_3 is described by ψ_{23} while the dependence between U_1 and U_2 or U_1 and U_3 is described by ψ_{123}. Right: the structure of a trivariate AC. The dependence between any two random variables is here described by ψ_{123}. The arrows in both structures are called edges. Note that the labels of the internal nodes in both structures (123 and 23 for the left structure, 23 for the right structure) are irrelevant.

In general, a set Ψ of arbitrary generators does not ensure that the resulting NAC defined through (λ, Ψ) will be a proper copula. In case we are working with fully nested Archimedean copula structures, that is, structures where each branching node has either two leaves as children, or one leaf and another branching node, a sufficient nesting condition ensuring this NAC is a proper copula was however developed by Joe (1997) and McNeil (2008). If all elements of Ψ belong to the same parametric family (if not, see Hofert, 2011), this sufficient nesting condition simply states that the parameters θ_I and θ_J related to ψ_I and ψ_J must be such that $\theta_I < \theta_J$ for every pair of internal nodes I and J in the NAC tree structure such that J is a child of I. Roughly speaking, it means the strength of the dependence between two random variables increases as these variables are able to interact through a node farther away from the root node. It was later proved that this sufficient nesting condition holds for all NACs, not only fully nested ones, see Holena et al. Also note that a very recent paper by Rezapour (2015) gave weak sufficient and *necessary* conditions concerning the generators, guaranteeing that the constructed function is a copula.

In the first approach to estimate a NAC, in Section 3, the target tree structure is estimated in such a way that, should the generators all belong to the same parametric family, their parameters can be easily estimated so that they meet the sufficient nesting condition. In the second approach to estimate a NAC, in Section 4, the target tree structure is estimated free of any concern for this sufficient nesting condition, but the problem of estimation of the generators afterwards in order to get a proper copula, even when all these generators belong to the same parametric family, remains an open problem.

Regarding rooted phylogenetic trees, one of the key findings of Segers and Uyttendaele (2014), which will be useful later, is the following: any rooted phylogenetic tree λ can be broken down into a unique set, denoted by ${}^3(\lambda)$, consisting of $\binom{d}{3}$ rooted trivariate tree structures, one trivariate structure for each combination of the elements of the vector $(U_1, \ldots, U_d)$

on which λ spans, taken three at the time without repetition. Moreover, a given set ${}^3(\lambda)$ can in turn be used to retrieve the tree structure λ from which ${}^3(\lambda)$ was calculated.

In order to estimate the tree structure λ of a target nested Archimedean copula, Segers and Uyttendaele (2014) therefore suggest to estimate, one at the time, each element of ${}^3(\lambda)$, thus effectively getting $\widehat{{}^3(\lambda)}$ which can then be used to build $\hat{\lambda}$.

The ability to estimate the tree structure of a nested Archimedean copula spanned on three random variables (U_i, U_j, U_k) based on n observations from (X_i, X_j, X_k) is a critical requirement for the estimation of ${}^3(\lambda)$. As outlined by Segers and Uyttendaele (2014), there are only four possible structures in the trivariate case: a trivial structure, such as the one on the right-hand side of Figure 1, or a structure where one variable is left apart and the two others are put together, as seen on the left-hand side of Figure 1, where U_2 and U_3 are put together and U_1 is located closer to the root.

If the trivariate target structure is assumed not to be the trivial structure, then picking one of the three remaining structures as estimate of the trivariate target structure is not a complicated problem: just estimate the Kendall distribution for each of the tree pairs (X_i, X_j), (X_i, X_k) and (X_j, X_k), and find out which are the two estimated Kendall distributions that are the closest according to some distance. If, for instance, the estimated Kendall distributions of (X_i, X_j) and (X_i, X_k) are the closest, then the trivariate target tree structure must be a structure where U_i is left apart while U_j and U_k are together.

Please note that if for some reason the target structure λ spanned on $(U_1, \ldots, U_n)$ is known to be a binary structure, then so is each element of ${}^3(\lambda)$. Therefore, in this particular case, each element of ${}^3(\lambda)$ can be estimated using only what is described in the previous paragraph.

Finding out if a trivariate target structure is actually the trivial structure or not is a much more difficult problem, for which Segers and Uyttendaele (2014) developed a hypothesis test where they try to see if the average of the two closest estimated Kendall distributions is significantly different from the third estimated Kendall distribution. If it is not the case, then one cannot rule out that the three underlying Kendall distributions all coincide, and the trivariate target structure is estimated by a 3-fan. The distribution of their test statistic being unknown under the null, they rely on the bootstrap to get a p-value for the test. As the estimation of all elements from ${}^3(\lambda)$ will require this test to be performed $\binom{d}{3}$ times, getting $\hat{\lambda}$ using their approach can be computationally expensive, especially as the value of d increases.

Some suggested papers for the readers eager to learn more about NACs are: McNeil (2008), Hofert and Pham (2013) or Okhrin et al. (2013b).

3 A first approach to estimate a NAC

In this section, a first approach to estimate a nested Archimedean copula (λ, Ψ) is presented. With this approach, λ is first estimated in two steps: a binary tree is built using hierarchical clustering and then is collapsed if necessary, according to some strategy. Second, if the target generators are assumed to belong to a same known parametric family F, estimation of the set of parameters Θ so that the sufficient nesting condition (see Section 2) is met can easily be performed. Okhrin et al. (2013a) and Górecki et al. (2015) are both particular cases of this first approach.

Estimation of the target tree structure λ. It is first assumed that the target structure spanned on $(U_1, \ldots, U_d)$ is a binary tree, that is, a structure where each internal node has two and only two children (this assumption will be later relaxed). Example of a binary tree: the tree on the left-hand side of Figure 1 or on the left-hand side of Figure 2.

Based on an iid sample of size n from $(X_1, ..., X_d)$ and knowing that the tree structure spanned on $(U_1, \ldots, U_d)$ is a binary tree, a distance for each couple (X_i, X_j) with distinct $i, j \in \{1, \ldots, d\}$ is first estimated. The random variables are then clustered, using hierarchical clustering, one at the time, according to the estimated distances. Getting an estimated binary tree structure this way however makes sense only if the estimated distance d for each couple (X_i, X_j) is a function of a measure of association $\bowtie$ such that highly related random variables will be considered close and therefore the distance d between the two will be small, if the tree is built from leaves to the root using the smallest distances first and if the target NAC is such that the dependence between any two random variables in the structure increases as the variables are able to interact through nodes that are farther away from the root, that is, if the target NAC (λ, Ψ) is assumed to meet the sufficient nesting condition.

A measure of association $\bowtie$ between two random variables can be obtained, for instance, through

- Kendall's τ,
- a distance between the (theoretical) Kendall distribution of two independent variables and the empirical Kendall distribution of the two variables under study,
- Hoeffding's $\mathbb{D}$ statistic (see Hoeffding, 1948),
- or Spearman's rho.

Note that these measures of association are all such that $\bowtie (X_i, X_j) = \bowtie (U_i, U_j)$, so that the binary tree estimated on $(X_1, ..., X_d)$ is actually the binary tree estimated on $(U_1, \ldots, U_d)$.

For the linkage criteria, single, complete or average linkage can be used (and where experimentally tested by Górecki et al., 2015). In the rest of

this paper, only average linkage will be considered.

If somehow the target tree structure λ is known to be a rooted binary tree structure, it is possible to directly move on to the problem of estimation of the generators from here. However, a target NAC structure is not always a rooted binary structure. To allow for a more general estimation, suggestion is to collapse, if necessary, one or several parts of the estimated binary structure $\hat{\lambda}_b$ produced by the hierarchical clustering.

Let $N = \{n_1, ..., n_g\}$ be the set of internal nodes in $\hat{\lambda}_b$. For any two nodes $\{n_l, n_m\}$ with $l, m \in \{1, ..., g\}$ such that n_l is a parent of n_m, collapse the two nodes if you suspect the generator ψ_l to be too similar to the generator ψ_m. The right-hand side of Figure 2 shows the collapsing of nodes 234 and 34 into a new node. Notice the resulting tree is not a binary tree anymore.

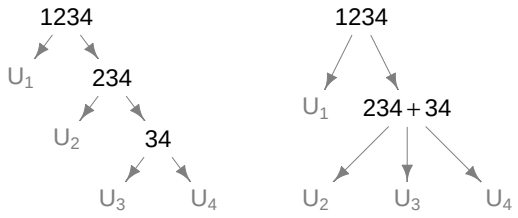


Figure 2: Left: a binary structure spanned on four random variables. Right: nodes 234 and 34 have been collapsed into one node.

To compare two successive generators ψ_l and ψ_m, one needs first to estimate these generators, as they are unknown. If they are known to belong to a same given parametric family F, the problem is then to estimate θ_l and θ_m and to collapse if $\left|\hat{\theta}_l - \hat{\theta}_m\right| < \theta_c$. Note that at this point, we are not concerned by the sufficient nesting condition: the goal here is only to decide if two successive nodes in the binary tree should be collapsed, based on a rough estimation of the related generators ψ_l and ψ_m. Formal estimation of the generators is done later. To make the problem of whether or not two successive nodes should be collapsed easier, each of the two related generators is summarized as a scalar measure s reflecting the dependence between the random variables interacting through the related node, and the two nodes are collapsed if the absolute difference between their respective estimated scalar measure is lower than a chosen threshold, that is, if

$$|\hat{s}_l - \hat{s}_m| < s_c.$$

For instance, looking back at Figure 2, one can estimate a summary of the dependence between the random variables interacting through node 234 by averaging the estimated Kendall correlation coefficients, calculated

within the random pairs (U_2, U_3) and (U_2, U_4). The estimated summary of the generator related to node 34 is equal to the estimated Kendall's τ between the random variables (U_3, U_4). The two nodes are collapsed if the inequality

$$\left| \left(\frac{\hat{\tau}_{23} + \hat{\tau}_{24}}{2} \right) - \hat{\tau}_{34} \right| < \tau_c \tag{3.1}$$

holds, where τ_c is the critical threshold for collapsing.

Instead of using Kendall's τ in (3.1), other measures of association can be used, for instance Spearman's ρ, Hoeffding's $\mathbb{D}$ statistic, etc.

Another strategy to decide if two successive nodes should be collapsed or not is to break down both structures before and after collapsing of two given nodes into their respective set of trivariate pieces (refer to Section 2 or to Segers and Uyttendaele, 2014). Since the structures before and after collapsing are different, some of the trivariate pieces will be different as well. Let P be the set of trivariate pieces before collapsing and P_+ the set of pieces after collapsing. Let P_Δ be the set of vectors of the form (U_i, U_j, U_k) such that the tree structure spanned on (U_i, U_j, U_k) is not the same in P and P_+. In P_+, the tree structure spanned on (U_i, U_j, U_k) is a 3-fan, while in P, it is not the case. The decision to collapse or not is made by performing the hypothesis test developed by Segers and Uyttendaele (2014), since this test precisely aims at deciding whether or not the tree structure spanned on three random variables (U_i, U_j, U_k) is a trivial tree structure.

Looking back at Figure 2, one can observe that, in the left structure, the tree spanned on (U_2, U_3, U_4) is not a 3-fan, while in the right structure it is. If the p-value of the test is lower than or equal to a threshold α, the nodes 234 and 34 should not be collapsed into one.

When there is more than one vector of the form (U_i, U_j, U_k) in P_Δ, the hypothesis test developed by Segers and Uyttendaele (2014) must be applied several times, and the end result is a set of p-values. For such cases, a rule of thumb such as the one hereafter can be used (do not collapse if the inequality holds):

$$\text{average p-value} \leq \alpha \tag{3.2}$$

Estimation of the set of generators Ψ. Once the target tree structure λ has been estimated, one needs to estimate the generators. Usually, as done in Okhrin et al. (2013a) and Górecki et al. (2015), it is assumed that all these generators belong to a same known parametric family F. In such case, the only thing left to estimate is a set of parameters Θ.

An interesting by-product of the hierarchical clustering used to build $\hat{\lambda}_b$ is a set of non-increasing measures of association from leaves to the root of $\hat{\lambda}_b$. Since the sufficient nesting condition requires the θs in Θ to decrease

from leaves to the root of the phylogenetic tree, any monotonically increasing map δ:

$$\begin{array}{l} \text{space of the chosen measure of association used to build } \hat{\lambda}_b \\ \rightarrow \text{ parameter's space for the parametric family } F \text{ chosen for the generators,} \end{array}$$

applied to these non-increasing measures of association, will lead to a set $\hat{\Theta}$ allowing to get an estimated NAC $(\hat{\lambda}, \hat{\Psi})$ meeting the sufficient nesting condition.

In Okhrin et al. (2013a), the parametric family for the generators is specified prior to the hierarchical clustering, allowing them to directly use, as if it was a measure of association, the θ parameter related to that parametric family (the fact the θ parameter is not an actual measure of association is irrelevant in this case). The by-product of the hierarchical clustering allows therefore to directly get $(\hat{\lambda}, \hat{\Psi})$ meeting the sufficient nesting condition without any extra effort. In other words, $\delta(\bullet) = \bullet$.

In Górecki et al. (2015), the measure of association used for the hierarchical clustering is Kendall's τ. The main advantage of using Kendall's τ is that there is a direct, well known, relationship between Kendall's τ and θ for most parametric family of generators. That is, a possible map $\delta(\bullet)$ is easy to find. For instance, if the parametric family for the generators is assumed to be the Clayton family, a possible map between Kendall's τ and θ is

$$\delta(\tau) = \frac{2\tau}{1-\tau} = \theta \tag{3.3}$$

See Hofert and Maechler (2011) for the map using other families.

Using other measures of association, such as Hoeffding's $\mathbb{D}$ statistic for instance, a suitable map δ is not known. However, as long as δ is a monotonically increasing function, $(\hat{\lambda}, \hat{\Psi})$ will be a proper copula, but the resulting estimator could be biased or with a very high mean square error.

To conclude this section, the first approach to estimate a target NAC using hierarchical clustering is summarized in Algorithm 1 hereafter. Note that the major difference between Algorithm 1 and Algorithm 3 in Górecki et al. (2015) is the collapsing step, starting at line 6 and ending at line 22 of Algorithm 1, while missing in Górecki et al. (2015).

Input. An iid sample from $(X_1, ..., X_d)$ such that the related copula is assumed to be a NAC, a measure of association $\bowtie$ and a distance d based on $\bowtie$ (a large value for $\bowtie$ leading to a small distance), a linkage criteria L, a strategy S for how to collapse successive nodes in the binary tree resulting from the hierarchical clustering, a generator's family F and a map δ.

Output. An estimated NAC $(\hat{\lambda}, \hat{\Psi})$ meeting the sufficient nesting condition.

Algorithm 1 Estimating a NAC using hierarchical clustering.

1: **for all** pairs (X_i, X_j) such that $i, j \in \{1, ..., d\}$ **do**
2: get $d(X_i, X_j)$ and store it in a distance matrix.
3: **end for**
4: using hierarchical clustering with L and the distance matrix just built, get a binary tree $\hat{\lambda}_b$.
5: Also store the set of ordered measures of association for later use as A, this set being a by-product of the hierarchical clustering. Note that the cardinality of A is $d - 1$.
6: set $\hat{\lambda}_t = \hat{\lambda}_b$.
7: **repeat**
8: set $\hat{\lambda} = \hat{\lambda}_t$.
9: let $N = \{n_1, ..., n_g\}$ be the set of internal nodes in $\hat{\lambda}_t$.
10: **for all** pairs (n_l, n_m) in N such that n_l is the parent of n_m **do**
11: try to collapse n_l and n_m using strategy S.
12: store the decision, do not apply it yet.
13: **end for**
14: update $\hat{\lambda}_t$ by collapsing what needs to be collapsed.
15: do this based on the set of decisions one gets from the loop just above.
16: **until** $\hat{\lambda} == \hat{\lambda}_t$.
17: the repeat until loop automatically stops as soon as no further collapsing of $\hat{\lambda}_t$ is possible.
18: next, update the set of ordered measures of association A by comparing $\hat{\lambda}_b$ to $\hat{\lambda}$ as described hereafter.
19: **for all** sets of nodes in $\hat{\lambda}_b$ that have been collapsed into one node in $\hat{\lambda}$ **do**
20: get the minimum of the related ordered measures of association in A.
21: replace the related ordered measures of association in A by this minimum.
22: **end for**
23: apply the map δ on each element of A to get a set of estimated parameters $\hat{\Theta}$, which can then be used with F to get $\hat{\Psi}$.
24: **return** $(\hat{\lambda}, \hat{\Psi})$

Although this algorithm might seem long, its implementation is rather straightforward.

4 A second approach to estimate a NAC

Segers and Uyttendaele (2014) paved the way for an alternative approach to

that of Section 3. In Section 3, hierarchical clustering is mandatory, and $\hat{\lambda}$ always results from it. In Segers and Uyttendaele (2014), the tree structure λ is estimated free of any concern for the generators: the user can get a tree $\hat{\lambda}$ without knowing anything about the target NAC. Moreover, the simulation study in Segers and Uyttendaele (2014) showed that they were, in general, able to retrieve the target tree λ more often than Okhrin et al. (2013a), where hierarchical clustering is used.

What to do next, once λ has been estimated, remains an open problem left for future research by Segers and Uyttendaele (2014). If the generators across λ are assumed to belong to a same parametric family F, estimation of the related set of parameters Θ can be performed by simple inversion of the estimated Kendall correlation coefficients, the same way it was done using equation (3.3) in Section 3. In most cases, this is expected to lead to a proper NAC $(\hat{\lambda}, \hat{\Psi})$. But in some cases, it will unfortunately not be true. This is because estimated Kendall correlation coefficients along trees built by Segers and Uyttendaele (2014) are not always ordered, making the sufficient nesting condition hard to satisfy. A stage-wise estimation of the parameters based on a quasi-maximum likelihood approach (from the lowest to the highest nesting level), as done by Okhrin and Ristig (2014), with subsequent shortening of the feasible parameter space, could however help ensure that the final object is a well defined NAC.

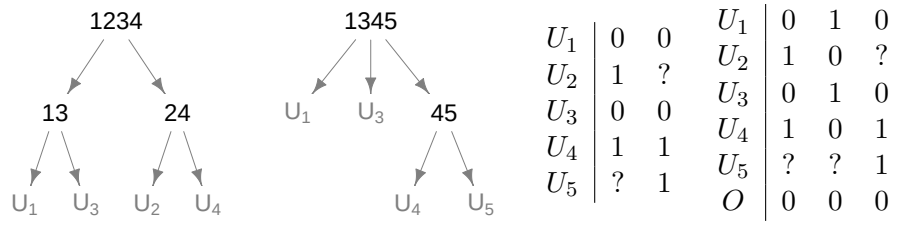
While this section does not solve the problem of estimation of the generators for this second approach, a new way to estimate λ free of any concern for the generators, using a *supertree method*, is presented hereafter, as an alternative to the tree structure estimator from Segers and Uyttendaele (2014).

Supertree methods have been extensively studied in the field of phylogenetics. They are designed to output a phylogenetic tree such that this phylogenetic tree will be the best match of an input set of smaller trees, this input set including trees of various sizes, conflicting trees and also missing trees (that is, some information to build the supertree is actually lacking). Some interesting references to get started are Bininda-Emonds (2004), Wilkinson et al. (2005) or Swenson et al. (2012).

Two supertree methods implemented in the R package phytools (Revell, 2012) have been tested. They both take as input a set of unrooted trees and output an unrooted tree. In a rooted tree, there is always a special node called the root and all edges are directed away from that root. Any node in a rooted tree but the root has always one and only one parent and the root is the common ancestor to all other nodes. In an unrooted tree, it is not possible to identify the parent of an arbitrary node, as a neighbouring node could be about anything (child or parent), since the edges are undirected, thanks to the absence of root in the graph. The process of rooting an unrooted tree is straightforward: pick an internal node and define it as the

root. The process of unrooting a rooted tree is a little more complex: if the root has two and only two children, the root is removed and the two children become directly connected. Otherwise, the root remains in the graph. In both cases, the edges become undirected.

To start, and assuming all input trees are unrooted (if they are not, they are first unrooted), all the topological information available across the input trees is gathered as a matrix, called the character matrix. Let Λ be the set of all unrooted input trees. Let $E = \{e_1, ..., e_g\}$ be the set of internal edges (an internal edge is an edge between two internal nodes) across all trees in Λ. Let $U = \{U_1, ..., U_d\}$ be the set of leaves such that each of these leaves exists in at least one of the input trees. Elements of U that are on one side of edge e_i, $i = \{1, ..., g\}$, will receive the state 0, elements that are on the other side will receive the state 1. Elements of U that are on neither of the two sides of e_i receive the unknown state for that edge. The $d \times g$ character matrix contains all of the states. A supertree is by definition a tree that spans on all of the elements of U and the goal of a supertree method is to find a supertree such that some loss function, this loss function comparing the topological information of the supertree to the $d \times g$ character matrix, is optimized.



U_1	0	0
U_2	1	?
U_3	0	0
U_4	1	1
U_5	?	1

U_1	0	1	0
U_2	1	0	?
U_3	0	1	0
U_4	1	0	1
U_5	?	?	1
O	0	0	0

Figure 3: Two rooted input trees and the related character matrix built on the unrooted version of these two trees first without and second with an outgroup added.

As an example, consider the two tree structures in Figure 3. Suppose one wants to build a tree spanned on $U = \{U_1, \ldots, U_5\}$ based on these two trees. Both input trees are first unrooted. The first rooted tree in Figure 3 loses one internal edge in the process and becomes a tree with structure of the form $>-<$, as shown on the top left side of Figure 4, where the only remaining internal edge (the $-$ in $>-<$) has nodes 13 and 24 at its tips.

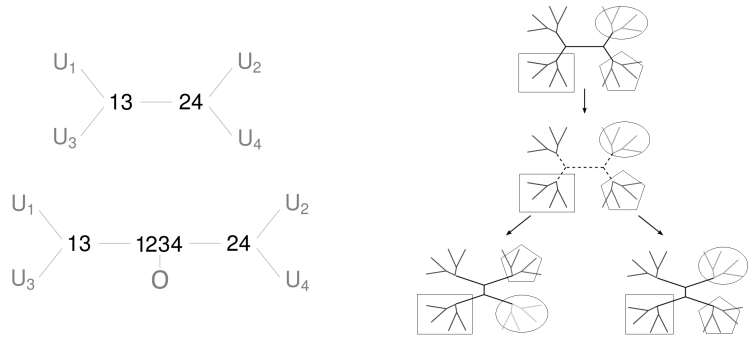


Figure 4: Top left: the unrooted version of the first rooted tree displayed in Figure 3. Bottom left: an outgroup O has been attached to the root of the first rooted tree displayed in Figure 3 before the tree is unrooted. Thanks to this outgroup, we are able to keep track of the root, even though the tree has been unrooted. Right: an example of nearest neighbour interchange (NNI) from Wikipedia.

The second rooted tree in Figure 3, when unrooted, also becomes a tree with structure of the form $>-<$, where the internal edge has nodes 1345 and 45 at its tips. The character matrix that gathers all the topological information available based on these two unrooted trees is the 5×2 matrix displayed in Figure 3. Looking at the internal edge in the first unrooted tree (top left of Figure 4), we see that U_1 and U_3 are on one side of that edge and U_2 and U_4 are on the other side. We assign the label (or state) 0 to U_1 and U_3, and the label 1 to U_4 and U_2. The leaf U_5 does not appear in this input unrooted tree and it is therefore not known to which side this leaf belongs. We therefore assign an unknown state to U_5, thus ending the first column of the 5×2 character matrix. For the second unrooted tree, U_1 and U_3 are on one side of the internal edge and U_4 and U_5 are on the other side. We do not know to which side U_2 belongs. This makes up the second column of the 5×2 character matrix.

Once the character matrix has been built, the next step is to find an unrooted supertree that will agree as much as possible with the topological information available in the matrix. As loss function, phylogeneticians use what is called the parsimony score, which can be easily calculated for a given supertree using the algorithm developed by Fitch (1971).

To find the supertree with the minimum parsimony score, the strategy is to pick a starting supertree and then to apply topological rearrangements to that supertree in a recursive fashion. Rearrangements leading to a supertree with a lower parsimony score are kept, changes such that the resulting supertree has a higher parsimony score are not kept. The final supertree is one such that no further rearrangements of the supertree allows to lower the parsimony score.

The two supertree methods tested in this paper differ only in the way the starting supertree is defined and in the way the starting supertree is recur-

sively modified. The first of these methods, later denoted by NJNNI, uses as starting supertree a tree built based on the neighbor joining (NJ) clustering method from Saitou and Nei (1987). The changes applied recursively on the tree are NNI rearrangements, see Felsenstein (2004, pp. 39) for a detailed description and the right side of Figure 4 for an example.

For the second method, later denoted by RNix, the starting supertree is chosen at random and is then modified according to Nixon (1999).

Both these methods output an unrooted supertree and, by unrooting input rooted trees, destroy the topological information that could be used to root the outputted supertree.

There is one way to avoid this, though. The idea is to make use of an extra leaf, called the outgroup. The outgroup is attached to the root of each input rooted tree prior to the process of unrooting them. The set U defined earlier becomes $\{U_1, ..., U_d, O\}$. As an example, the character matrix with such outgroup for the two input trees in Figure 3 is displayed on the most right part of the same figure. The unrooted version of the first tree in Figure 3 with this outgroup attached is displayed at the bottom left of Figure 4. Note that the character matrix, with outgroup, has now as many columns as there are edges in the original, rooted, input trees. A supertree based on this 6×3 character matrix will span on $\{U_1, \ldots, U_5\}$ and the outgroup O. Once a satisfying supertree is found, the final step is to use the outgroup to root the supertree before removing that outgroup.

In order to estimate a target binary NAC tree structure using one of the two supertree methods described above, the suggestion is to use, as input set of trees, the set of binary trivariate trees one gets by estimating the binary tree spanned on each vector of three random variables (U_i, U_j, U_k) with distinct $i, j, k \in \{1, \ldots, d\}$, refer to Section 2 for more details.

As it was done in Section 3, the estimated binary tree $\hat{\lambda}_b$ can then be collapsed according to some strategy S in order to get $\hat{\lambda}$. Comparison between this new approach to get $\hat{\lambda}$ and the one from Segers and Uyttendaele (2014) is performed in Section 5.

5 Performance study

In this section, we are concerned with the ability of estimating λ based on a sample from (λ, Ψ). When the sample size is n, the methodology used in this section to estimate the performance of a NAC tree structure estimator is described through the following steps:

- generate $N = 100$ samples of size n from (λ, Ψ);
- apply the NAC tree structure estimator on each of these N samples while considering the univariate margins unknown (that is, work on ranks) and get N estimates of λ;

- calculate a distance between each estimate of λ and λ itself;
- get the average of the N resulting distances or some other descriptive measure of these distances, such as:

$$(\text{average of the distances})^2 + \text{variance of the distances}; \tag{5.1}$$

- the lower the average of the distances or (5.1) is, the better the performance of the estimator for the NAC defined through (λ, Ψ) is, given a fixed sample size n.

Two distances between a given estimate of λ and λ itself are considered. The first one is a 01-distance: if the estimate of λ is actually equal to λ, then the distance is 0. Otherwise, the distance is 1. The second distance, called the tri-distance, is based on the comparison of the trivariate pieces of the estimate of λ and the trivariate pieces of λ itself. If both trees are equal, all the trivariate pieces will be equal as well, and the distance is 0. If among the $\binom{d}{3}$ trivariate pieces from the estimate of λ and the $\binom{d}{3}$ trivariate pieces from λ a total of k pieces differ, then the distance is k. The maximum possible tri-distance is therefore $\binom{d}{3}$. Unlike the 01-distance, this last distance allows to assess how far from the target structure a misspecified structure produced by a given estimator is, something never studied before.

Since the estimator from Segers and Uyttendaele (2014) is based on a hypothesis test, it is required to choose a threshold α prior to the estimation of a target tree structure.

Regarding estimators based on hierarchical clustering (Section 3), the choice of a threshold to decide if any two successive nodes in the estimated binary tree structure should be collapsed is also required prior to the estimation of a target tree structure, as seen in (3.2) or (3.1).

Comparison of the performance of different tree structure estimators is therefore a challenge, as the performance of a given estimator depends on the chosen threshold for that estimator. Given a sample size n, a target NAC (λ, Ψ) and a tree structure estimator, the suggestion is to use a threshold ensuring that $P(\hat{\lambda}_n = \lambda)$ is maximized. This particular threshold will be called the optimal threshold. By making use of the related optimal threshold for each estimator, one only takes into account the best performance of each estimator, which should allow for "fair" comparisons between estimators.

In the particular case where λ is a binary structure,

- the estimator from Segers and Uyttendaele (2014), which is based on $\binom{d}{3}$ hypothesis tests, should always reject the nulls. If one null is not rejected, it means the final estimated structure contains at least a 3-fan, and therefore the estimate of λ cannot be equal to λ itself. Thus the threshold α should be set to 100% or more so that all nulls are always rejected.
- Regarding estimators based on hierarchical clustering, in case λ is a binary structure, $P(\hat{\lambda}_n = \lambda)$ is maximized if the collapsing step is

skipped. This can be achieved by setting α to 100% or more in (3.2), and τ_c to 0 or less in (3.1).

Although unknown when λ is not a binary structure, the optimal threshold for an estimator can be estimated. Indeed, given $N = 100$ samples of size n, a target NAC (λ, Ψ) and a tree structure estimator, the estimated optimal threshold is the value such that the average of the $N = 100$ 01-distances between each estimate of λ and λ itself, is minimal. Note that on real data, since the target structure is by definition unknown, the optimal threshold cannot be estimated as it is done here.

In this section, all tree structure estimators but the one from Segers and Uyttendaele (2014) rely on two steps: a binary tree is first built, and then parts of it are collapsed. For the first step, RNix and NJNNI refer to the two supertree methods described in Section 4 while kind, hD and kt refer to distance matrices related to measures of deviation from the independent bivariate Kendall distribution, Hoeffding's $\mathbb{D}$ statistics or Kendall correlation coefficients, respectively (a large measure of deviation, a large $\mathbb{D}$ or a large Kendall's τ leading to a small distance). For the second step, kb and kagg refer to (3.2) and (3.1), respectively. The tree structure estimator developed by Segers and Uyttendaele (2014) will be referred to as S&U.

Target structures are given in Figures 5 and 6.

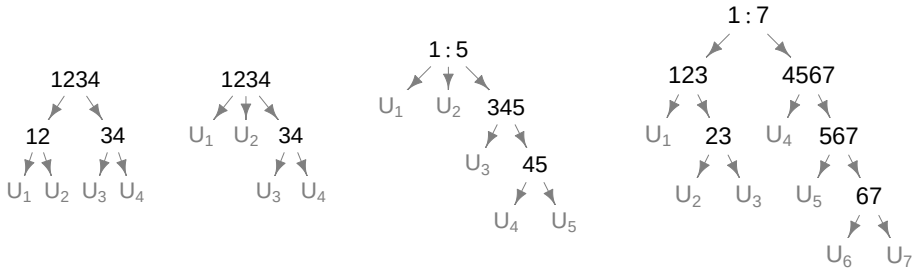


Figure 5: two four-variate structures, a five-variate structure and a seven-variate structure.

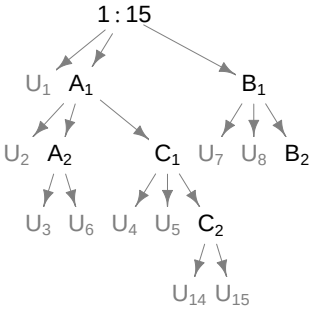


Figure 6: A fifteen-variate structure, where B_2 refer to an Archimedean copula spanned on U_9 through U_{13}.

Figure 7 through 11 give the simulation results for these target structures. The generators used across each structure and the related parameters, expressed as Kendall correlation coefficients for convenience, are specified below the figures.

Notice the average of the 01-distances is actually equal to the percentage of estimates that are unequal to the target structure, so that a value of 1 means not a single estimate of λ among the $N = 100$ available was equal to the target λ, while a value of 0 means all estimates of λ were equal to λ.

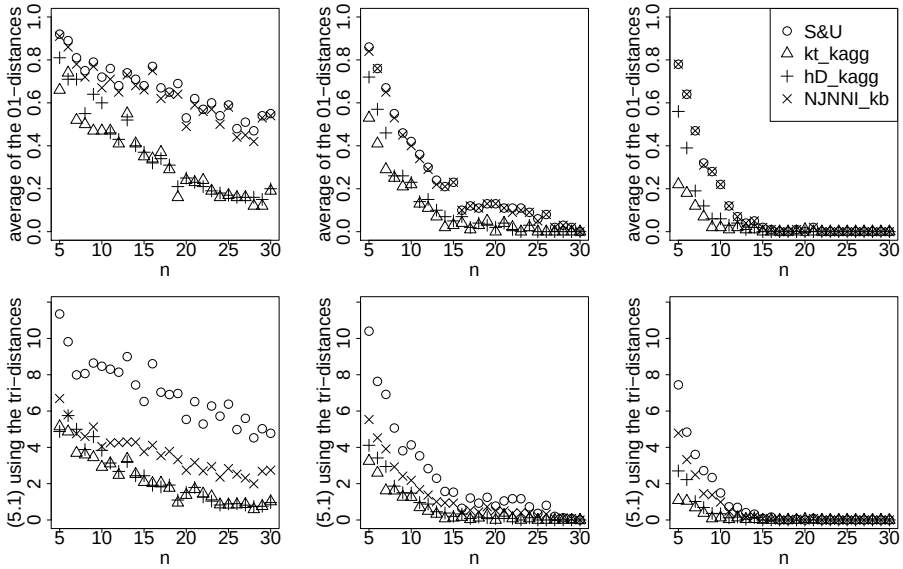


Figure 7: results for the four-variate binary structure. The generators used across the structure are all Clayton generators. The related sets of parameters are ($\tau_{1234} = 0.4, \tau_{12} = 0.6, \tau_{34} = 0.6$) for the left-hand side of the figure, ($\tau_{1234} = 0.3, \tau_{12} = 0.7, \tau_{34} = 0.7$) in the middle, and ($\tau_{1234} = 0.2, \tau_{12} = 0.8, \tau_{34} = 0.8$) for the right-hand side of the figure.

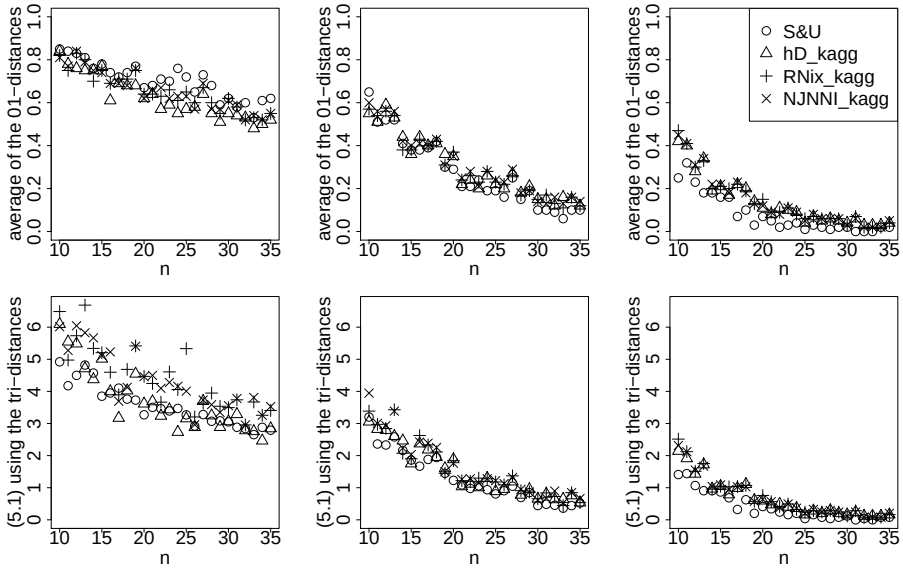


Figure 8: results for the second four-variate structure. The generators used across the structure are, again, all Clayton generators. The related sets of parameters are ($\tau_{1234} = 0.4, \tau_{34} = 0.6$) for the hand-left side of the figure, ($\tau_{1234} = 0.3, \tau_{34} = 0.7$) in the middle, and ($\tau_{1234} = 0.2, \tau_{34} = 0.8$) for the hand-right side of the figure.

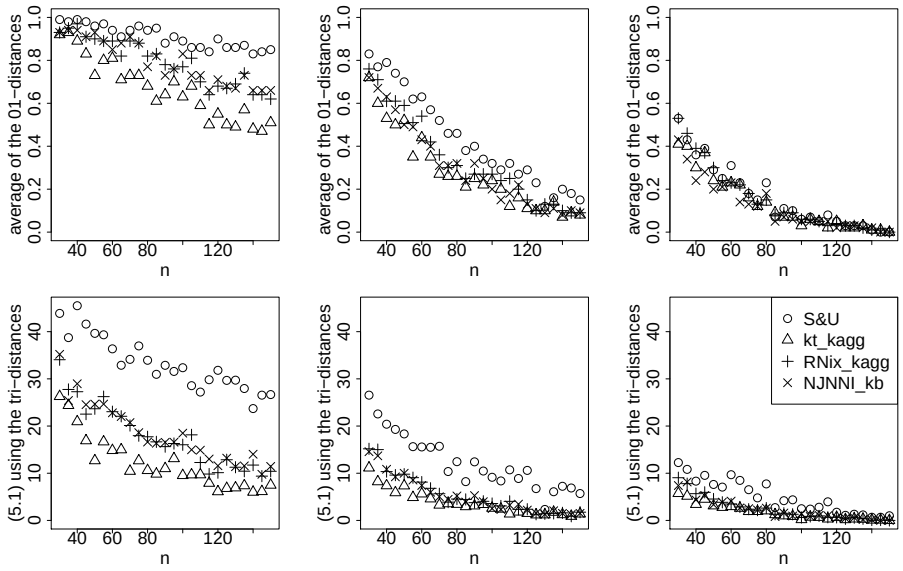


Figure 9: results for the five-variate structure. The generators used across the structure are all Gumbel generators. The related sets of parameters are ($\tau_{1:5} = 0.4, \tau_{345} = 0.5, \tau_{34} = 0.6$) for the left-hand side of the figure, ($\tau_{1:5} = 0.3, \tau_{345} = 0.5, \tau_{34} = 0.7$) in the middle, and ($\tau_{1:5} = 0.2, \tau_{345} = 0.5, \tau_{34} = 0.8$) for the right-hand side of the figure.

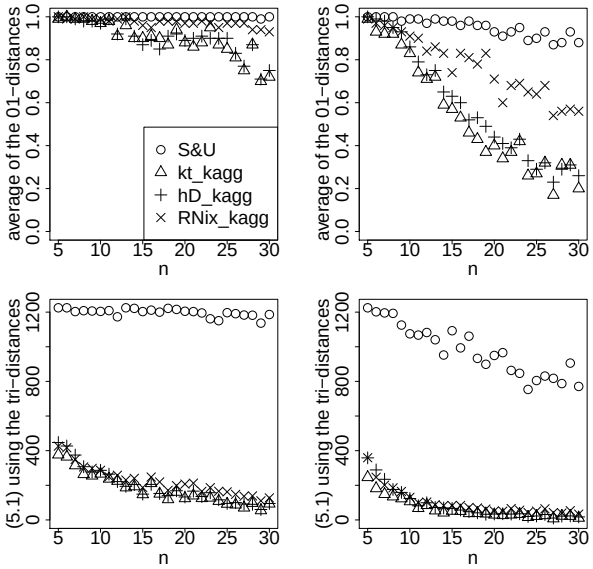


Figure 10: results for the seven-variate structure. The generators used across the structure are all Frank generators. The related sets of parameters are $(\tau_{1:7} = 0.35, \tau_{123} = 0.5, \tau_{23} = 0.65, \tau_{4:7} = 0.45, \tau_{567} = 0.55, \tau_{67} = 0.65)$ for the left-hand side of the figure and $(\tau_{1:7} = 0.2, \tau_{123} = 0.5, \tau_{23} = 0.8, \tau_{4:7} = 0.4, \tau_{567} = 0.6, \tau_{67} = 0.8)$ for the right-hand side of the figure.

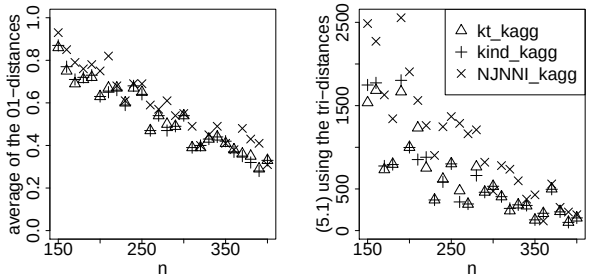


Figure 11: results for the fifteen-variate structure. The generators used across the structure are all Joe generators. The set of parameters is $(\tau_{1:15} = 0.1, \tau_{A_1} = 0.25, \tau_{A_2} = 0.5, \tau_{B_1} = 0.5, \tau_{B_2} = 0.75, \tau_{C_1} = 0.35, \tau_{C_2} = 0.45)$.

Some comments are:

- As the sample size increases, all estimators perform better.
- The more the target NAC is resolved, the better the performance, as seen on Figure 7 through 10. The tree structure of a poorly resolved NAC is, in general, harder to estimate.
- The group of estimators used in this section are clearly not a homogeneous group regarding performances (or execution times). For instance, at the top left of Figure 7, one can see that the NJNNI_kb estimator performs slightly better than the S&U estimator in terms

of mean 01-distance, while the kt_kagg and hD_kagg estimators both significantly perform better.

- Several estimators beat the S&U estimator by a large amount in case of binary target structures (Figure 7 and 10). For instance, top right of Figure 10, the kt_kagg estimator can be seen to get the target structure wrong only 20% of the time when the sample size is 30, while the S&U estimator gets it wrong more than 80% of the time on the same samples. Moreover, the bottom right part of the same figure shows that, when the kt_kagg gets the target structure wrong, it's only by a very small, almost negligible, amount. When the S&U estimator gets it wrong, the resulting estimate of λ is usually far away from λ itself.
- As seen in Figure 11, the kind_kagg and kt_kagg estimators seem to offer the same ability to retrieve the target structure. Note however that the kt_kagg estimator is easier to compute.
- No matter what the generators of the target NAC are, notice that kt_kagg remains the top runner in almost all figures where it is used. This suggest that what the generators of the target tree are is of little importance regarding performance.
- In the simulations for the four-variate binary structure, the kt_kagg estimator turned out to be the one with the smallest execution times, producing estimates up to a 100 times faster than the S&U estimator. Whatever the target NAC, estimators using hierarchical clustering seemed to always exhibit smaller execution times than the S&U estimator on the same data.

6 Exams results from 482 students

In this section, a phylogenetic tree is estimated based on the results of 482 students to their exams, using the kt_kagg tree structure estimator with τ_c set to 0.075. The main reason we use the kt_kagg tree structure estimator rather than another is because it was the best performing one in our performance study, as well as the fastest.

The 482 students are in their first year, studying economics and management in a Belgian university. They took 14 exams, that is, there are 14 random variables: Private Law, Psychology and Management, Sociology, Chemistry, Geography, English, Dutch, History, Mathematics, Physics, Statistics, a course designed to make sure students have the required prerequisites in sciences (BasicSci), and, finally, micro and macro economics (Econ103) and a related course (Econ104). Figure 12 shows the estimated tree.

To help interpret the estimated structure, an estimated summary of the generator at each internal node of the structure is obtained by averaging the

estimated Kendall correlation coefficients of all the pairs of random variables interacting through that node. For instance, 0.51 was obtained by averaging the Kendall correlation coefficients of the pairs (Hist, Math), (Hist, Phys), (Hist, Stat), (Hist, BasicSci), (Hist, Econ103), (Hist, Econ104), (Math, Phys), (Math, Stat), (Math, BasicSci), (Math, Econ103), (Math, Econ104), (Phys, Stat), (Phys, BasicSci), (Phys, Econ103), (Phys, Econ104), (Stat, BasicSci), (Stat, Econ103), (Stat, Econ104), (BasicSci, Econ103) and (BasicSci, Econ104).

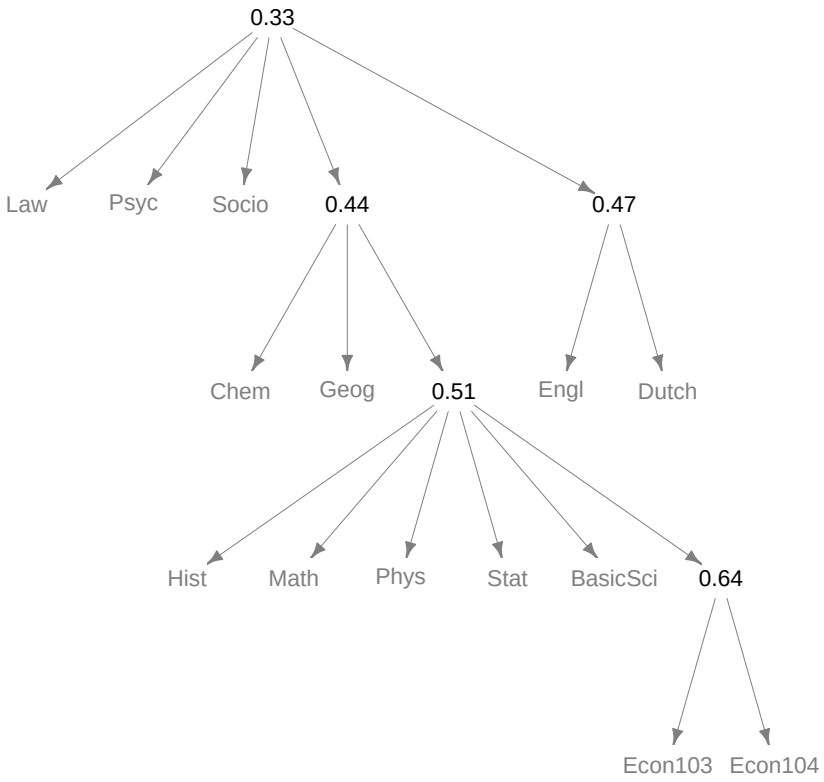


Figure 12: Phylogenetic tree built on the grades of 482 students.

The estimated average Kendall correlation coefficient at the root is 0.33, suggesting that students tend to have good grades everywhere or bad grades everywhere, and less often a mix of good and bad grades. The strongest estimated Kendall's τ can be observed between the courses Econ103 and Econ104. Merging both courses in one exam could be a time-saving idea for the teachers in charge. English and Dutch courses are apart from the rest

of the tree, with an estimated Kendall's τ of 0.47. Natural sciences such as Mathematics, Physics or Statistics are related through a rather strong estimated mean Kendall's τ (0.51), while courses such as Psychology or Sociology are both directly connected to the root where the dependence is the weakest.

If one is to assume that the five generators of the tree shown in Figure 12 all belong to a same parametric family F, say the Clayton family, estimation of the related set of parameter Θ can be easily performed by inverting the mean Kendall correlation coefficients displayed in Figure 12 according to equation (3.3).

To check the fit, one could compare the empirical correlation matrix, built on the raw data, versus the correlation matrix one gets from $(\hat{\lambda}, \hat{\Psi})$. A good match between the two matrices does not mean that the true copula underlying the dataset is a NAC, but is nonetheless a key requirement.

Since comparison of two 14×14 correlation matrices is quite obnoxious to perform, suggestion is to use principal component analysis to help us perform this check. The 482 students, projected on the first two components, can be seen in the top left panel of Figure 13. 5000 points from $(\hat{\lambda}, \hat{\Psi})$ can be seen in the top right panel of Figure 13.

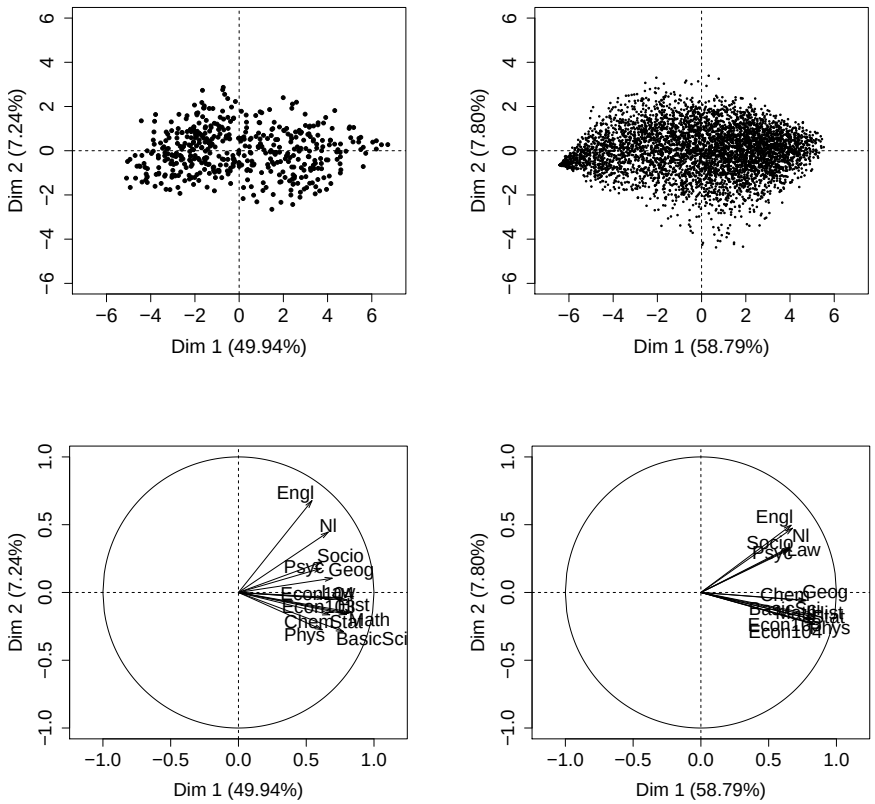


Figure 13

The circle of correlations based on the 482 students can be seen in the bottom left panel of Figure 13. The circle of correlations based on 5000 observations from $(\hat{\lambda}, \hat{\Psi})$ can be seen in the bottom right panel of Figure 13. Both figures show a relative good match between the original data and the one simulated from $(\hat{\lambda}, \hat{\Psi})$.

In most papers related to NACs, a generator with the same functional form is used throughout the tree structure. Different generators can however be used, as discussed by Hofert (2011), offering extra flexibility. In his paper, Hofert (2011) gives several examples of different Archimedean families one can use while meeting the sufficient nesting condition. For instance, Hofert (2011) tells us that if a parent node is defined through a Clayton generator, then the child node can be defined through generator

$$\psi_h(t) = (1 + t^{1/\theta_h})^{-1},$$

as long as the parameter's value of the Clayton generator $\in (0, 1]$. Translated as a Kendall correlation coefficient, it means we cannot exceed $1/3$ for the

correlation coefficient on the parent node.

Looking back at Figure 12, at least one opportunity for the generator ψ_h can be seen: node 0.47. Indeed, the parent node has a Kendall correlation coefficient just below 1/3. The only remaining task is to revert the value 0.47 in order to get $\hat{\theta}_h$. Straighforward calculations can show that $\theta_h = \frac{4}{6(1-\tau)}$, leading to $\hat{\theta}_h = 1.258$. The resulting estimated NAC now uses different generators across its tree structure, and, thanks to Hofert (2011), we know this estimated NAC is a proper copula.

Of course, one has no reason to use different generators unless the resulting NAC can be shown to fit the data better. To check if using $\psi_h(t)$ at node 0.47 allows to improve the fit, let us compare the log-likelihood of the Archimedean copula on (Engl, Dutch) using a Clayton generator first and then using $\psi_h(t)$. Straighforward calculations can show that the copula density using $\psi_h(t)$ as generator is

$$c(u, w; \theta) = \frac{2(-1 + 1/u)^\theta (-1 + 1/w)^\theta}{D},$$

where

$$D = u^2 w^2 (-1 + 1/u)(-1 + 1/w)\big(1 + 1/\theta((-1 + 1/u)^\theta + (-1 + 1/w)^\theta)\big)^3,$$

allowing to calculate the related log-likelihood. The log-likehood using a Clayton generator is 74.00. Using $\psi_h(t)$, the log-likelihood is 50.10. So in our case, it seems we are better off with a Clayton generator at node 0.47.

7 Discussion

Based on the performance study, it seems that the tree structure estimator kt_kagg, introduced in this paper as an extension of the work from Górecki et al. (2015), is most likely the best NAC tree structure estimator we have at the moment, not only because it is able to retrieve the target structure more often than other estimators, but also because it is one of the fastest tree structure estimator there is. The new tree structure estimator based on supertree methods has failed to beat the kt_kagg estimator in our study, but nonetheless outperforms the one from Segers and Uyttendaele (2014) in both speed and ability to retrieve the target tree structure. Can we find a better tree structure estimator than kt_kagg? Research on the matter must continue. Hopefully, this will be facilitated by the framework developed in the current paper, as well as by the novel way of comparing tree structure estimators, using the tri-distance introduced in Section 5.

References

Olaf RP Bininda-Emonds. The evolution of supertrees. *Trends in Ecology & Evolution*, 19(6):315–322, 2004.

J. Felsenstein. *Inferring Phylogenies*. Sinauer Associates, Incorporated, 2004. ISBN 9780878931774.

Walter M Fitch. Toward defining the course of evolution: minimum change for a specific tree topology. *Systematic Biology*, 20(4):406–416, 1971.

Jan Górecki, Marius Hofert, and Martin Holeňa. On the consistency of an estimator for hierarchical archimedean copulas. 2014.

Jan Górecki, Marius Hofert, and Martin Holeňa. An approach to structure determination and estimation of hierarchical archimedean copulas and its application to bayesian classification. *Journal of Intelligent Information Systems*, pages 1–39, 2015.

Wassily Hoeffding. A non-parametric test of independence. *The Annals of Mathematical Statistics*, pages 546–557, 1948.

M. Hofert and M. Maechler. Nested Archimedean Copulas Meet R: The nacopula Package. *Journal of Statistical Software*, 39(9):1–20, 3 2011. ISSN 1548-7660. Please note the package nacopula has been merged with the package copula.

Marius Hofert. Efficiently sampling nested Archimedean copulas. *Computational Statistics & Data Analysis*, 55(1):57–70, 2011.

Marius Hofert and David Pham. Densities of nested archimedean copulas. *Journal of Multivariate Analysis*, 118:37–52, 2013.

Martin Holena, Lukas Bajer, and Martin Scavnicky. Using copulas in data mining based on the observational calculus.

H. Joe. *Multivariate Models and Dependence Concepts*. Chapman and Hall, London, 1997.

A. J. McNeil and J. Nešlehová. Multivariate Archimedean copulas, d-monotone functions and l_1-norm symmetric distributions. *The Annals of Statistics*, 37:3059–3097, 2009.

Alexander J. McNeil. Sampling nested Archimedean copulas. *Journal of Statistical Computation and Simulation*, 78(6):567–581, 2008. doi: 10.1080/00949650701255834.

Meei Pyng Ng and Nicholas C Wormald. Reconstruction of rooted trees from subtrees. *Discrete Applied Mathematics*, 69(1):19–31, 1996.

Kevin C Nixon. The parsimony ratchet, a new method for rapid parsimony analysis. *Cladistics*, 15(4):407–414, 1999.

O. Okhrin and A. Ristig. Hierarchical Archimedean Copulae: The HAC Package. *Journal of Statistical Software*, 58:1–20, 2014.

Ostap Okhrin, Yarema Okhrin, and Wolfgang Schmid. On the structure and estimation of hierarchical Archimedean copulas. *Journal of Econometrics*, 173(2):189–204, 2013a. ISSN 0304-4076.

Ostap Okhrin, Yarema Okhrin, and Wolfgang Schmid. Properties of hierarchical Archimedean copulas. *Statistics & Risk Modeling*, 30(1):21–54, 2013b.

Liam J. Revell. phytools: An R package for phylogenetic comparative biology (and other things). *Methods in Ecology and Evolution*, 3:217–223, 2012.

Mohsen Rezapour. On the construction of nested archimedean copulas for d-monotone generators. *Statistics & Probability Letters*, 101:21–32, 2015.

Naruya Saitou and Masatoshi Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution*, 4(4):406–425, 1987.

Johan Segers and Nathan Uyttendaele. Nonparametric estimation of the tree structure of a nested archimedean copula. *Computational Statistics & Data Analysis*, 72:190–204, 2014.

M Shel Swenson, Rahul Suri, C Randal Linder, and Tandy Warnow. Superfine: fast and accurate supertree estimation. *Systematic biology*, 61(2): 214–227, 2012.

Mark Wilkinson, James A Cotton, Chris Creevey, Oliver Eulenstein, Simon R Harris, Francois-Joseph Lapointe, Claudine Levasseur, James O Mcinerney, Davide Pisani, and Joseph L Thorley. The shape of supertrees to come: tree shape related properties of fourteen supertree methods. *Systematic biology*, 54(3):419–431, 2005.