Hierarchical Archimedean Copulae: The HAC Package

Ostap Okhrin
Dresden University of Technology

Alexander Ristig
Humboldt-Universität zu Berlin

Abstract

This paper presents the R package **HAC**, which provides user friendly methods for dealing with hierarchical Archimedean copulae (HAC). Computationally efficient estimation procedures allow to recover the structure and the parameters of HAC from data. In addition, arbitrary HAC can be constructed to sample random vectors and to compute the values of the corresponding cumulative distribution plus density functions. Accurate graphics of the HAC structure can be produced by the `plot` method implemented for these objects.

Keywords: copula, R, hierarchical Archimedean copula, HAC.

1. Introduction

The use of copulae in applied statistics began in the end of the 90ies, when Embrechts, McNeil, and Straumann (1999) introduced copula to empirical finance in the context of risk management. Nowadays, quantitative orientated sciences like biostatistics and hydrology use copulae to attempt measuring the dependence of random variables, e.g., Lakhal-Chaieb (2010); Acar, Craiu, and Yao (2011); Bárdossy (2006); Genest and Favre (2007); Bárdossy and Li (2008). In finance, copulae became a standard tool, explicitly on value at risk (VaR) measurement and in valuation of structured credit portfolios, see Mendes and Souza (2004); Junker and May (2005) and Li (2000). This paper aims at providing the necessary tools for academics and practitioners for simple and effective use of hierarchical Archimedean copulae (HAC) in their statistical analysis.

A copula is the function splitting a multivariate distribution into its margins and a pure dependency component. Formally, copulae are introduced in Sklar (1959) stating that if F is an arbitrary d-dimensional continuous distribution function of the random vector $X = (X_1, \dots, X_d)^\top$, then the associated copula is unique and defined as the continuous mapping $C : [0,1]^d \to [0,1]$ which satisfies the equality

$$C(u_1, \dots, u_d) = F\{F_1^{-1}(u_1), \dots, F_d^{-1}(u_d)\}, \quad u_1, \dots, u_d \in [0,1],$$

where $F_1^{-1}(\cdot), \dots, F_d^{-1}(\cdot)$ are the quantile functions of the corresponding continuous marginal distribution functions $F_1(x_1), \dots, F_d(x_d)$. Accordingly, a d-dimensional density $f(\cdot)$ can be split in the copula density $c(\cdot)$ and the product of the marginal densities. For an overview and recent developments of copulae we refer to Nelsen (2006), Cherubini, Luciano, and Vecchiato (2004), Joe (1997) and Jaworski, Durante, and Härdle (2013). If $F(\cdot)$ belongs to the class of elliptical distributions, then $C(\cdot)$ is an elliptical copula, which in most cases cannot be given explicitly because the distribution function $F(\cdot)$ and the inverse marginal distributions $F_j(\cdot)$ usually have integral representations. One of the classes that overcomes this drawback of elliptical copulae is the class of Archimedean copulae, which, however, is very restrictive yet for moderate dimensions. Among other R (R Core Team 2014) packages dealing with Archimedean copula (see for example Dutang 2014), we would like to mention the **copula** and the **fCopulae** package, c.f. Yan (2007); Kojadinovic and Yan (2010); Hofert and Maechler (2011); Hofert, Kojadinovic, Maechler, and Yan (2014) and Wuertz *et al.* (2013).

HAC generalize the concept of simple Archimedean copulae by substituting (a) marginal distribution(s) by a further HAC. This class is thoroughly analyzed in Embrechts, Lindskog, and McNeil (2003); Whelan (2004); Savu and Trede (2010); Hofert (2011); Okhrin, Okhrin, and Schmid (2013b). The first sampling algorithms for special HAC structures were provided by the **QRMlib** package of McNeil and Ulman (2011), which is not updated anymore, but several functions were ported to the **QRM** package (see Pfaff and McNeil 2013). Hofert and Maechler (2012) presented the comprehensive **nacopula** package which, among other features, allows sampling from arbitrary HAC and was integrated into the package **copula** from version 0.8-1 on. The central contribution of the **HAC** package (Okhrin and Ristig 2014) is the estimation of the parameter and the structure for this class of copulae, as discussed in Okhrin, Okhrin, and Schmid (2013a), including a simple and intuitive representation of HAC as R objects of the class '`hac`'. The main estimation procedure relies on a recursive multi-stage maximum likelihood (ML) procedure, which determines the parameter and the structure simultaneously. This elegant procedure endows the estimator with the usual asymptotic properties but avoids the computationally intensive one-step ML estimation, which is also implemented for a predetermined structure. Besides, the package offers functions for producing graphics of the copula's structure, for sampling random vectors from a given copula and for computing values of the corresponding distribution and density.

The paper is organized as follows. Section 2 describes shortly the theoretical aspects of HAC and its estimation. Section 3 presents the functions of the **HAC** package and Section 4 a simulation study. Section 5 concludes.

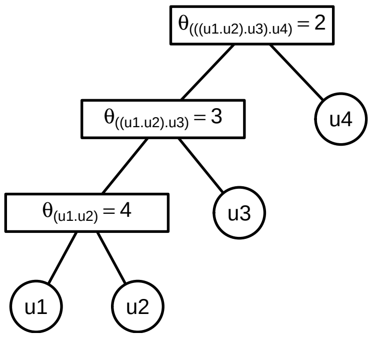
2. Hierarchical Archimedean copulae

As mentioned above, the large class of copulae, which can describe tail dependency, non-ellipticity, and, most importantly, has close form representation

$$C(u_1,\dots,u_d;\theta) = \phi_\theta\left\{\phi_\theta^{-1}(u_1)+\dots+\phi_\theta^{-1}(u_d)\right\}, \quad u_1,\dots,u_d \in [0,1], \tag{1}$$

where $\phi_\theta(\cdot) \in \mathfrak{L} = \{\phi_\theta : [0;\infty) \to [0,1] \,|\, \phi_\theta(0) = 1,\ \phi_\theta(\infty) = 0;\ (-1)^j\phi_\theta^{(j)} \geq 0;\ j \in \mathbb{N}\}$ and $(-1)^j\phi_\theta^{(j)}(x)$ being non-decreasing and convex on $[0,\infty)$, for $x > 0$, is the class of Archimedean copulae. The function $\phi(\cdot)$ is called the generator of the copula and commonly depends on a single parameter θ. For example, the Gumbel generator is given by $\phi_\theta(x) = \exp(-x^{1/\theta})$ for $0 \leq x < \infty$, $1 \leq \theta < \infty$. Detailed reviews of the properties of Archimedean copulae can be found in McNeil and Nešlehová (2009) and in Joe (1997).

A disadvantage of Archimedean copulae is the fact that the multivariate dependency structure is very restricted, since it typically depends on a single parameter of the generator function $\phi(\cdot)$. Moreover, the rendered dependency is symmetric with respect to the permutation of



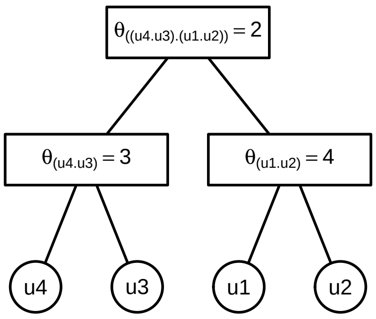


Figure 1: Fully and partially nested Archimedean copulae of dimension $d = 4$ with structures $s = (((12)3)4)$ on the left and $s = ((43)(12))$ on the right.

variables, i.e., the distribution is exchangeable. HAC (also called nested Archimedean copulae) overcome this problem by considering the compositions of simple Archimedean copulae. For example, the special case of four-dimensional fully nested HAC can be given by

$$\begin{aligned} C(u_1, u_2, u_3, u_4) &= C_3\{C_2(u_1, u_2, u_3), u_4\} \\ &= \phi_3\{\phi_3^{-1} \circ C_2(u_1, u_2, u_3) + \phi_3^{-1}(u_4)\}, \end{aligned} \tag{2}$$

where $C_j(u_1, \ldots, u_{j+1}) = \phi_j[\phi_j^{-1}\{C_{j-1}(u_1, \ldots, u_j)\} + \phi_j^{-1}(u_{j+1})]$, $j = 2, \ldots, d-1$, and $C_1 = \phi_1\{\phi_1^{-1}(u_1) + \phi_1^{-1}(u_2)\}$. The functional form of $C_j(\cdot)$ indicates that the composition can be applied recursively. A different segmentation of the variables leads naturally to more complex HAC. In the following, let d-dimensional HAC be denoted by $C(u_1, \ldots, u_d; s, \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ denotes the vector of feasible dependency parameters and $s = (\ldots(i_g i_k)i_\ell \ldots)$ the structure of the entire HAC, where $i_m \in \{1, \ldots, d\}$ is a reordering of the indices of the variables with $m = 1\ldots, d$, and $g, k, \ell \in \{1, \ldots, d : g \neq k \neq \ell\}$. Structures of subcopulae are denoted by s_j with $s = s_{d-1}$. For instance, the structure according to Equation 2 is $s = (s_2)4$ with $s_j = (s_{j-1}(j+1))$, $j = 2, 3$, for the sucopulae and $s_1 = (12)$. A clear definition of the structure is essential, as s is in fact a parameter to estimate. Thus, Equation 2 can be rewritten as

$$\begin{aligned} C(u_1, u_2, u_3, u_4; s = (((12)3)4), \boldsymbol{\theta}) &= C\{u_1, u_2, u_3, u_4; (s_2 4), (\theta_1, \theta_2, \theta_3)^\top\} \\ &= \quad \phi_{\theta_3}(\phi_{\theta_3}^{-1} \circ C_2\{u_1, u_2, u_3; (s_1(3)), (\theta_1, \theta_2)^\top\} + \phi_{\theta_3}^{-1}(u_4)). \end{aligned}$$

Figure 1 presents the four-dimensional fully and partially nested Archimedean copula.

HAC can adopt arbitrarily complex structures s. This makes it a very flexible and simultaneously parsimonious distribution model. The generators $\phi_{\theta_j}(\cdot)$ within a single nested Archimedean copula can come either from a single generator family or from different generator families. If the $\phi_{\theta_j}(\cdot)$'s belong to the same family, then the required complete monotonicity of $\phi_{\theta_{i+j}}^{-1}(\cdot) \circ \phi_{\theta_j}(\cdot)$ usually imposes some constraints on the parameters $\theta_1, \ldots, \theta_{d-1}$. Theorem 4.4 of McNeil (2008) provides sufficient conditions on the generator functions to guarantee that $C(\cdot)$ is a copula. It holds that if $\phi_{\theta_j}(\cdot) \in \mathfrak{L}$, for $j = 1, \ldots, d-1$, and $\phi_{\theta_{j+1}}^{-1}(\cdot) \circ \phi_{\theta_j}(\cdot)$ have completely monotone derivatives, then $C(\cdot)$ is a copula for $d \geq 2$. For the majority of generators feasible HAC require decreasing parameters from the highest to the lowest hierarchical level. However, in the case of different families within a single HAC, the condition of complete monotonicity is not always fulfilled, see Hofert (2011). In our study, we consider HAC with generators from the same family only. If we use the same single-parameter generator function on each level, but with a different value of θ, we may specify the whole distribution with at most $d-1$ parameters. From this point of view, the HAC approach can be seen as an alternative to covariance driven models. Nevertheless, for HAC not only the parameters are unknown, but also the structure has to be determined. One possible procedure for estimating both the parameters and the structure is to enumerate all possible structures and to estimate at first the parameters only. Next, the optimal structure can be determined by a suitable goodness-of-fit test. This approach is, however, unrealistic in practice because the variety of different structures is enormously large even in moderate dimensions. Okhrin *et al.* (2013a) suggest computationally efficient procedures, which allow to estimate HAC recursively. The **HAC** package provides these methods for estimating the parameters and structure in a user-friendly way.

2.1. Estimation of HAC

The entire procedure can be described in a recursive way where at the first iteration step we fit a bivariate copula to every couple of the variables. The couple of variables with the strongest dependency is selected. We denote the respective estimator of the parameter at the first level by $\hat{\theta}_1$ and the set of indices of the variables by I_1. The selected couple is joined together to define the pseudo-variable $Z_{I_1} \overset{\text{def}}{=} C\{(I_1);\hat{\theta}_1,\phi_1\}$. At the next step, we proceed in the same way by considering the remaining variables and the new pseudo-variable as the new set of variables. This procedure allows us to determine the estimated structure of the copula. As the restrictions on the parameters are always fulfilled due to shortening the parameter space, the procedure leads to a feasible copula funciton with $d-1$ parameters. Nevertheless, if the true copula is not binary, the procedure might return a slightly misspecified structure. Despite a difference in the structures, the difference in the distribution functions is in general minor. To allow more sophisticated structures, we aggregate the variables of the estimated copula afterwards. This is possible if the absolute value of the difference of two successive nodes is smaller than a fixed small threshold, i.e., $\theta_1 - \theta_2 < \epsilon$, with $\theta_1 > \theta_2$, as suggested by Okhrin *et al.* (2013a).

For better understanding, let us consider a three-dimensional example with u_j, $j = 1,2,3$, being uniformly distributed on $[0,1]$. All possible pairs $C_{(12)}(u_1,u_2,\hat{\theta}_{(12)})$, $C_{(13)}(u_1,u_3,\hat{\theta}_{(13)})$ and $C_{(23)}(u_2,u_3,\hat{\theta}_{(23)})$ are estimated by regular ML, see Franke, Härdle, and Hafner (2011). To compare the strengths of the fit one can use computationally complicated goodness-of-fit tests, which do not necessarily lead to a function which will be a copula on the final level of aggregation due to the restrictions on $\boldsymbol{\theta}$. For that reason we compare simply the estimates $\hat{\theta}_{(12)}$, $\hat{\theta}_{(13)}$ and $\hat{\theta}_{(23)}$. This is due to the fact that for most Archimedean copulae, the larger the parameter the stronger is the dependency (the larger the parameter the larger is Kendall's correlation coefficient). Let the strongest dependence be in the first pair $\hat{\theta}_1 \overset{\text{def}}{=} \hat{\theta}_{(12)} = \max\{\hat{\theta}_{(12)},\hat{\theta}_{(13)},\hat{\theta}_{(23)}\}$, then $I_1 = \{1,2\}$ and we introduce the pseudo-variable $Z_1 \overset{\text{def}}{=} C_1(I_1;\hat{\theta}_1) = C_1(u_1,u_2;\hat{\theta}_{(12)})$. At the next and final step for this example we join together u_3 and Z_1. The theoretical validation is also reported by Proposition 1 of Okhrin *et al.* (2013b) stating that HAC can be uniquely recovered from the marginal distribution functions and all bivariate copula functions. Crucially for the superior recursive ML estimation procedure, pseudo-variables are regarded as functions of the underlying random variables $X_1,\ldots,X_d$ and are not explicitly computed.

In practice, the marginal distributions F_j, $j = 1, \dots, d$, are either parametrically $\widehat{F}_j(\cdot) = F_j(\cdot, \hat{\boldsymbol{\alpha}}_j)$, where $\boldsymbol{\alpha}_j$ denotes the vector of parameters of the jth margin, or non-parametrically

$$\widehat{F}(x) = (n+1)^{-1} \sum_{i=1}^{n} \mathbf{I}(X_i \le x) \tag{3}$$

estimated in advance. Accordingly, the marginal densities $\hat{f}_j(\cdot)$, $j, \dots, d$, are estimated by an appropriate kernel density estimator or using a parametric density.

Following Okhrin *et al.* (2013a), the estimation of the copula parameters at each step of the iteration can be sketched as follows: at first stage, we estimate the parameter of the copula at the first hierarchical level assuming that the marginal distributions are known. At further stages, the next level copula parameter is estimated assuming that the margins as well as the copula parameters at lower levels are known. Let $\mathbf{X} = \{x_{ij}\}^\top$ be the respective sample, for $i = 1, \dots, n$, $j = 1, \dots, d$, and $\boldsymbol{\theta} = (\theta_1, \dots, \theta_{d-1})^\top$ be the parameters of the copula starting with the lowest up to the highest level. The recursive multi-stage ML estimator $\widehat{\boldsymbol{\theta}}$ solves the system

$$\left(\frac{\partial \mathcal{L}_1}{\partial \theta_1}, \dots, \frac{\partial \mathcal{L}_{d-1}}{\partial \theta_{d-1}} \right)^\top = \mathbf{0}, \tag{4}$$

where for $j = 1, \dots, d-1$

$$\mathcal{L}_j = \sum_{i=1}^{n} l_j(\mathbf{X}_i),$$

with for $i = 1 \dots, n$

$$l_j(\mathbf{X}_i) = \log \left\{ c_j \left[\{\widehat{F}_m(x_{im})\}_{m \in s_j};\, s_j, \theta_j \right] \prod_{m \in s_j} \hat{f}_m(x_{im}) \right\},$$

where s_j refers to the (pseudo)-variables considered at the jth estimation stage. Chen and Fan (2006) and Okhrin *et al.* (2013a) provide asymptotic behaviour of the estimates. At the moment, there are three different ways to estimate HAC:

(i) Ordinary (full) ML estimation, also denoted by FML, which is based on the complete log-likelihood and hence on a predetermined structure.

(ii) The ML setup is based on realized pseudo-variables, e.g., the pseudo-variable for the variables u_k and u_ℓ are computed according to Górecki, Hofert, and Holeňa (2014) as $Z_{k\ell} \overset{\text{def}}{=} \phi\left[2\phi^{-1}\{\max(u_k, u_\ell)\}\right]$, so that the bivariate density is maximized with respect to the copula parameter at each step of the procedure. This diagonal transformation of the copula avoids the bias around the initial node arising from the similar transformation $Z_{k\ell} \overset{\text{def}}{=} \phi\left\{\phi^{-1}(u_k) + \phi^{-1}(u_\ell)\right\}$. Note that this procedure is not supported by asymptotic theory.

(iii) More precise results can be obtained by the recursive ML (RML) procedure discussed in Okhrin *et al.* (2013a). The difference between the ML method and the recursive ML procedure results from the maximized log-likelihood. While the bivariate log-likelihood is considered at each estimation step of the ML method, the log-likelihood of the recursive ML procedure corresponds at each estimation step to the full log-likelihood for the marginal HAC regarded at that step. Compared to the full ML approach, the log-likelihood is only optimized with respect to the parameter at the root node taken the estimated parameter(s) at lower hierarchical levels as given, so that the final HAC being a copula is ensured by shortening the feasible parameter interval from above. From this point of view, the computational challenge is to build the log-likelihood for the full ML estimation, which is almost solved by constructing the d-dimensional density, see Section 3.4.

(iv) The penalized ML (PML) estimation procedure aims at determining a data driven sequence ϵ_n in order to fuse subsequent nodes, if $\hat{\theta}_1 - \hat{\theta}_2 < \epsilon_n$. This method merges advantages of the ML and RML procedure. The proposed method is, however, computationally intense, as the data driven thresholding parameter ϵ_n depends on tuning parameters arising from a non-concave penalization of the log-likelihood function, c.f., Fan and Li (2001).

3. Applications of HAC

Core of the **HAC** package is the function `estimate.copula` estimating the parameter and determining the structure for given data. Let us consider the dataset `finData` included in the **HAC** package. It contains the residuals of the filtered daily log-returns of four oil corporations: Chevron Corporation (`CVX`), Exxon Mobil Corporation (`XOM`), Royal Dutch Shell (`RDSA`) and Total (`FP`), covering $n = 283$ observations from 2011-02-02 to 2012-03-19. Intertemporal dependence is removed by usual ARMA-GARCH models, whose standardized residuals are plotted in Figure 2 and used in the subsequent analysis:

```
R> library("HAC")
R> data("finData")
R> system.time(result <- estimate.copula(finData, margins = "edf"))

   user  system elapsed
   0.05    0.00    0.04

R> result

Class: hac
Generator: Gumbel
((FP.RDSA)_{2.09}.(XOM.CVX)_{2.83})_{1.51}
```

The returned object `result` is of class '`hac`', whose properties are explored below. A practical illustration of the mechanism of `estimate.copula` related to the previous real data example is presented in Table 1.

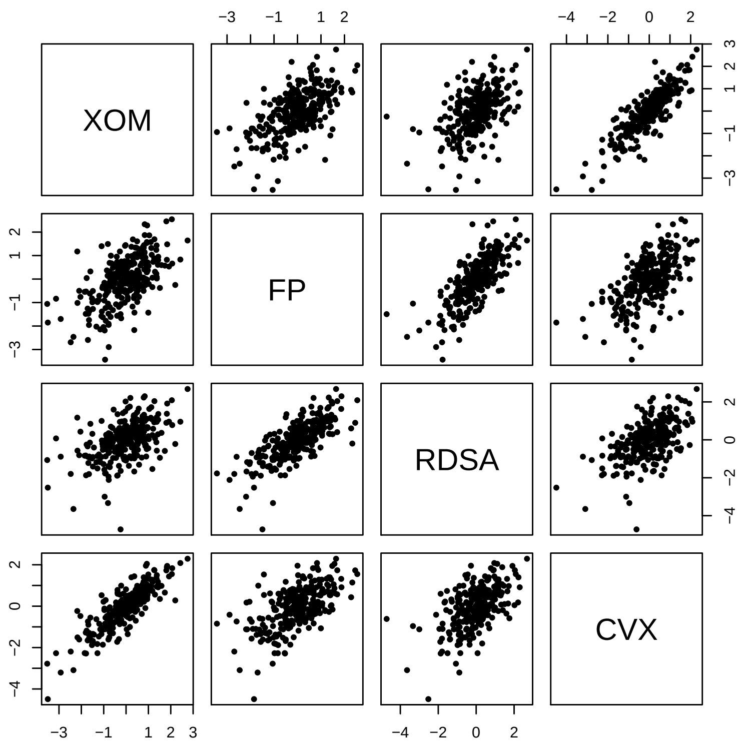


Figure 2: Scatterplot of the sample `finData`.

At the lowest hierarchical level, the parameters of all bivariate copulae are estimated. The couple $(X_{\texttt{CVX}}, X_{\texttt{XOM}})$ produces the strongest dependency, hence the best fit. Then, the pseudo-variable

$$Z_{(\texttt{CVX.XOM})} \overset{\text{def}}{=} \phi_{\hat{\theta}_{(\texttt{CVX.XOM})}} \left[\phi^{-1}_{\hat{\theta}_{(\texttt{CVX.XOM})}} \left\{ \widehat{F}_{\texttt{XOM}} \left(X_{\texttt{XOM}} \right) \right\} + \phi^{-1}_{(\hat{\theta}_{\texttt{CVX.XOM}})} \left\{ \widehat{F}_{\texttt{CVX}} \left(X_{\texttt{CVX}} \right) \right\} \right] \tag{5}$$

is defined, whose values are however not computed in practice, as the recursive ML procedure (`method = 3`) is used by default. At the next nesting level the parameters of all bivariate subsets are estimated and the variables $X_{\texttt{FP}}$ and $X_{\texttt{RDSA}}$ exhibit the best fit. Finally, the realizations of the remaining random variables $Z_{(\texttt{CVX.XOM})}$ and $Z_{(\texttt{FP.RDSA})}$ are grouped at the highest level of the hierarchy, where $Z_{(\texttt{FP.RDSA})}$ is defined analogously to $Z_{(\texttt{CVX.XOM})}$.

In general, `estimate.copula` includes the following arguments:

```
R> names(formals(estimate.copula))

 [1] "X"          "type"       "method"     "hac"        "epsilon"
 [6] "agg.method" "margins"    "na.rm"      "max.min"    "..."
```

The whole procedure is divided in three (optional) computational blocks. First, the margins are specified. Secondly, the copula parameter, $\boldsymbol{\theta}$, is estimated and finally the HAC is checked for aggregation possibilities. The `margins` of the $(n \times d)$ data matrix, `X`, are assumed to follow the standard uniform distribution by default, i.e., `margins = NULL`, but the function

	$z_{i,(\mathtt{CVX.XOM})} \overset{\text{def}}{=} \widehat{C}\{\widehat{F}_{\mathtt{CVX}}(x_{i,\mathtt{CVX}}), \widehat{F}_{\mathtt{XOM}}(x_{i,\mathtt{XOM}})\}$	$z_{i,(\mathtt{FP.RDSA})} \overset{\text{def}}{=} \widehat{C}\{\widehat{F}_{\mathtt{FP}}(x_{i,\mathtt{FP}}), \widehat{F}_{\mathtt{RDSA}}(x_{i,\mathtt{RDSA}})\}$
$(\mathtt{CVX.FP}) \rightsquigarrow \hat{\theta}_{(\mathtt{CVX.FP})}$ $(\mathtt{CVX.XOM}) \rightsquigarrow \hat{\theta}_{(\mathtt{CVX.XOM})}$ $(\mathtt{FP.RDSA}) \rightsquigarrow \hat{\theta}_{(\mathtt{FP.RDSA})}$ $(\mathtt{FP.XOM}) \rightsquigarrow \hat{\theta}_{(\mathtt{FP.XOM})}$ $(\mathtt{RDSA.XOM}) \rightsquigarrow \hat{\theta}_{(\mathtt{RDSA.XOM})}$	best fit $(\mathtt{CVX.XOM})$ $\Rightarrow$ $(\mathtt{CVX.XOM})\mathtt{FP} \rightsquigarrow \hat{\theta}_{(\mathtt{CVX.XOM})\mathtt{FP}}$ $(\mathtt{CVX.XOM})\mathtt{RDSA} \rightsquigarrow \hat{\theta}_{(\mathtt{CVX.XOM})\mathtt{RDSA}}$ $(\mathtt{FP.RDSA}) \rightsquigarrow \hat{\theta}_{(\mathtt{FP.RDSA})}$	best fit $(\mathtt{FP.RDSA})$ $\Rightarrow$ $((\mathtt{CVX.XOM})(\mathtt{FP.RDSA}))$ $\rightsquigarrow \hat{\theta}_{((\mathtt{CVX.XOM})(\mathtt{FP.RDSA}))}$

Table 1: The estimation procedure in practice.

Generator	Parameter	$\phi(u;\theta)$	$\tau(\theta)$
AMH	$\theta \in [0,1)$	$(1-\theta)/\{\exp(u)-\theta\}$	$1-2/(3\theta^2)\left\{\theta+(1-\theta)^2\log(1-\theta)\right\}$
Clayton	$\theta \in (0,\infty)$	$(u+1)^{-1/\theta}$	$\theta/(\theta+2)$
Frank	$\theta \in (0,\infty)$	$-\log\left[1-\{1-\exp(-\theta)\}\exp(-u)\right]/\theta$	$1+4/\theta\{D(\theta)-1\}$
Gumbel	$\theta \in [1,\infty)$	$\exp\left(-u^{1/\theta}\right)$	$1-1/\theta$
Joe	$\theta \in [1,\infty)$	$1-\{1-\exp(-u)\}^{1/\theta}$	$1-4\sum_{\ell=1}^{\infty}\left[\ell(\theta\ell+2)\{2+\theta(\ell-1)\}\right]^{-1}$

Table 2: Generator functions and the relations between the copula parameter and Kendall's $\tau(\cdot)$. The Debye function $D(\cdot)$ involved in $\tau(\cdot)$ for the family Frank is given by $D(\theta) = 1/\theta\int_0^\theta u/\{\exp(u)-1\}du$.

also permits non-uniformly distributed data as input if the argument `margins` is specified. The marginal distributions can be determined non-parametrically, `margins = "edf"`, or in a parametric way, e.g., `margins = "norm"`. Following the latter approach, the log-likelihood of the marginal distributions is optimized with respect to the first (and second) parameter(s) of the density `dxxx`. Based on these estimates, the values of the univariate margins are computed. If the argument is defined as scalar, all margins are computed according to this specification. Otherwise, different margins can be defined, e.g., `margins = c("norm", "t", "edf")` for a three-dimensional sample. Except the uniform distribution, all continuous distributions of the **stats** package (see `?Distributions`; R Core Team 2014) are available: `"beta"`, `"cauchy"`, `"chisq"`, `"exp"`, `"f"`, `"gamma"`, `"lnorm"`, `"norm"`, `"t"` and `"weibull"`. The values of non-parametrically estimated distributions are computed according to Equation 3.

Inappropriate usage of this argument might lead to misspecified margins, e.g., `margins = "exp"` although the sample contains negative values. Even though the margins might be assumed to follow parametric distributions if `margins != NULL`, no joint log-likelihood is maximized, but the margins are estimated in advance. As the asymptotic theory works well for parametric and nonparametric estimation of margins, for the univariate analysis we refer to other built-in packages. In practice, the column names of `X` should be specified, as the default names `X1, X2, ...` are given otherwise.

A further optional argument of `estimate.copula` determines the estimation `method`. As discussed above, we present three procedures: ML (`method = 1`), which is based on the bivariate density, full ML (`method = 2`) and recursive ML (`method = 3`) respectively. The routines of the **copula** package are imported if a simple Archimedean copula is fitted to the data, see Yan (2007); Kojadinovic and Yan (2010); Hofert and Maechler (2011).

At the final computational step of the procedure the binary HAC is checked for aggregation possibilities, if `epsilon > 0`. The new dependency parameter is computed according to the specification `agg.method`, i.e., the `"min"`, `"max"` or `"mean"` of the original parameters. To emphasize this point, recall the four-dimensional binary HAC

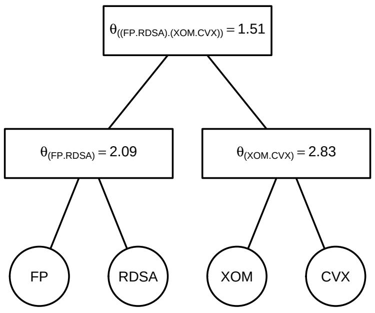
$$C(u_1,\dots,u_4;(((12)3)4),\boldsymbol{\theta}) \;= \phi_{\theta_3}\left\{\phi^{-1}_{\theta_3}\circ C\{u_1,\dots,u_3;((12)3),(\theta_1,\theta_2)^\top\} + \phi^{-1}_{\theta_3}(u_4)\right\}, \quad (6)$$

from Section 2. If we assume additionally $\theta_1 \approx \theta_2$, such that $\theta_1-\theta_2<\varepsilon$, the copula $C(\cdot)$ can be approximated by

$$C^*(u_1,\dots,u_4;((123)4),\boldsymbol{\theta}) \;= \phi_{\theta_3}\left\{\phi^{-1}_{\theta_3}\circ C\{u_1,\dots,u_3;(123),\theta^*\} + \phi^{-1}_{\theta_3}(u_4)\right\}, \quad (7)$$

where $\theta^* = (\theta_1+\theta_2)/2$ for instance. This is referred to as the associativity property of Archimedean copulae, see Theorem 4.1.5 of Nelsen (2006). If the variables of two nodes are aggregated, the new copula is checked for aggregation possibilities as well. Beside this threshold approach, the realized estimates $\hat{\theta}_1$ and $\hat{\theta}_2$ can obviously be used to test $H_0: \theta_1-\theta_2=0$, since the asymptotic distribution is known. On the other hand, this approach is extremely expensive computationally. The estimation results for the non-aggregated and the aggregated cases are presented in the following:

```
R> result.agg = estimate.copula(sample, margins = "edf", epsilon = 0.3)
R> plot(result, circles = 0.3, index = TRUE, l = 1.7)
R> plot(result.agg, circles = 0.3, index = TRUE, l = 1.7)
```



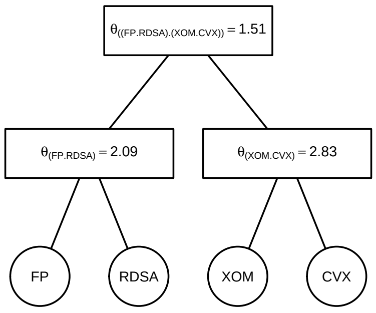


Figure 3: Plot of `result` on the left and `result.agg` on the right hand side.

3.1. The '`hac`' object

'`hac`' objects can be constructed by the general function `hac`, with the same name as the object it creates, and its simplified version `hac.full` for building fully nested HAC. For instance, consider the construction of a four-dimensional fully nested HAC with Gumbel generator, i.e.,

```
R> G.cop = hac.full(type  = 1,
+                   y     = c("X4", "X3", "X2", "X1"),
+                   theta = c(1.1, 1.8, 2.5))
R> G.cop

Class: hac
Generator: Gumbel
(((X1.X2)_{2.5}.X3)_{1.8}.X4)_{1.1}
```

where `y` denotes the vector of variables of class '`character`' and `theta` denotes the vector of dependency parameters. The parameters should be in ascending order, so that the first parameter, `1.1`, refers to the initial node of the HAC and the last parameter, `2.5`, corresponds to the first hierarchical level with variables `"X1"` and `"X2"`. The vector `y` has to contain one element more than the vector `theta`.

The S3 `print` method for '`hac`' objects gives an output structured in three lines: (i) the object's `Class`, (ii) the `Generator` family and (iii) the HAC structure s. The structure can also be produced by the supplementary function `tree2str`. Variables, grouped at the same node are separated by a dot "`.`" and the dependency parameters are printed within the curly parentheses.

Partially nested Archimedean copulae are constructed by `hac` with the main argument `tree`. For a better understanding let us first consider a four-dimensional simple Archimedean copula with dependency parameter $\theta = 2$:

```
R> hac(type = 1, tree = list("X1", "X2", "X3", "X4", 2))

Class: hac
Generator: Gumbel
(X1.X2.X3.X4)_{2}
```

The copula `tree` is constructed by a `list` consisting of four `character` objects, i.e., `"X1", "X2", "X3", "X4"`, and a number, which denotes the dependency parameter of the Archimedean copula. According to the theoretical construction of HAC in Section 2, we can induce structure by substituting margins through a subcopula. The four variables `"X1"`, `"X2"`, `"X3"`, `"X4"` can, for example, be structured by

```
R> hac(type = 1, tree = list(list("X1", "X2", 2.5), "X3", "X4", 1.5))

Class: hac
Generator: Gumbel
((X1.X2)_{2.5}.X3.X4)_{1.5}
```

where the nested component, `list("X1", "X2", 2.5)`, is the subcopula at the lowest hierarchical level. Note that the nested component is of the same general form `list(..., numeric(1))` as the simple Archimedean copula, where `numeric(1)` denotes the dependency parameter and "`...`" refers to arbitrary variables and subcopulae, which may contain subcopulae as well, like shown in the following:

```
R> HAC = hac(type = 1, tree = list(list("Y1", list("Z3", "Z4", 3), "Y2", 2.5),
+                              list("Z1", "Z2", 2), list("X1", "X2", 2.4),
+                              "X3", "X4", 1.5))
R> HAC

Class: hac
Generator: Gumbel
((Y1.(Z3.Z4)_{3}.Y2)_{2.5}.(Z1.Z2)_{2}.(X1.X2)_{2.4}.X3.X4)_{1.5}
```

We cannot avoid the notation becoming more cumbersome for higher dimensions, but the principle stays the same for arbitrary dimensions, i.e., variables are substituted by lists of the general form `list(..., numeric(1))`. The function `hac` provides a further argument for specifying the `type` of the HAC.

3.2. Graphics

As the string representation of the structure becomes more unclear as dimension increases, the package allows to produce graphics of '`hac`' objects using the S3 `plot` method for these objects. Figure 4 illustrates for example the dependence structure of the already defined object `HAC`.

```
R> plot(HAC, cex = 0.8, circles = 0.35)
```

The explanatory power of these plots can be enhanced by several of the usual `plot` parameters:

```
R> names(formals(plot.hac))

 [1] "x"         "xlim"      "ylim"      "xlab"      "ylab"
 [6] "col"       "fg"        "bg"        "col.t"     "lwd"
[11] "index"     "numbering" "theta"     "h"         "l"
[16] "circles"   "digits"    "..."
```

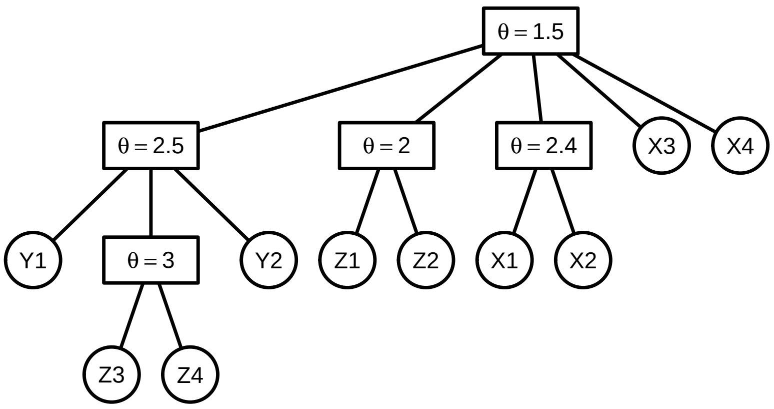


Figure 4: Plot of the object `HAC`.

The optional, boolean argument `theta` determines whether the dependency parameter of the copula θ or Kendall's τ is printed, whereby Kendall's τ cannot be easily interpreted in the usual way for more than two dimensions. The supplementary function `theta2tau` computes Kendall's rank correlation coefficient based on the value of the dependency parameter, whereas `tau2theta` corresponds to the inverse function, see Table 2. If `index = TRUE`, strings illustrating the subcopulae of the nodes are used as subscripts of the dependency parameters. If, additionally, `numbering = TRUE`, the parameters are numbered, such that the subscripts correspond to the estimation stages if the non-aggregated output of `estimate.copula` is plotted. The radius of the `circles`, the width `l` and the height `h` of the rectangles and the specific colors of the lines and the text can be adjusted. Further arguments "`...`" can, for example, be used to modify the font size `cex` or to include a subtitle `sub`.

3.3. Random sampling

To be in line with other R packages providing tools for different univariate and multivariate distributions we provide: (i) `dHAC` for computing the values of the copula density, (ii) `pHAC` for the cumulative distribution function and (iii) `rHAC` for simulations. Sampling methods are imported from the **copula** package and rely on the algorithm suggested in Hofert and Maechler (2011), who summarize the sampling procedure as follows:

Algorithm 1. *Let $C(\cdot)$ be a nested Archimedean copula with root copula $C_0(\cdot)$ generated by ϕ_0. Let U be a vector of the same dimension as $C_0(\cdot)$.*

```
1. Sample from inverse Laplace transform $\mathcal{LS}^{-1}$ of $\phi_0(\cdot)$, i.e., $V_0 \sim F_0(\cdot) \overset{\text{def}}{=} \mathcal{LS}^{-1}\{\phi_0(\cdot)\}$.
2. For all components $u$ of $C_0(\cdot)$ that are nested Archimedean copulae do:
    (a) Set $C_1(\cdot)$ with generator $\phi_1(\cdot)$ to the nested Archimedean copula $u$.
    (b) Sample $V_{01} \sim F_{01}(\cdot) \overset{\text{def}}{=} \mathcal{LS}^{-1}\{\phi_{01}(\cdot; V_0)\}$.
    (c) Set $C_0(\cdot) \overset{\text{def}}{=} C_1(\cdot), \phi_0(\cdot) \overset{\text{def}}{=} \phi_1(\cdot)$, and $V_0 \overset{\text{def}}{=} V_{01}$ and continue with 2.
3. For all other components $u$ of $C_0(\cdot)$ do:
```

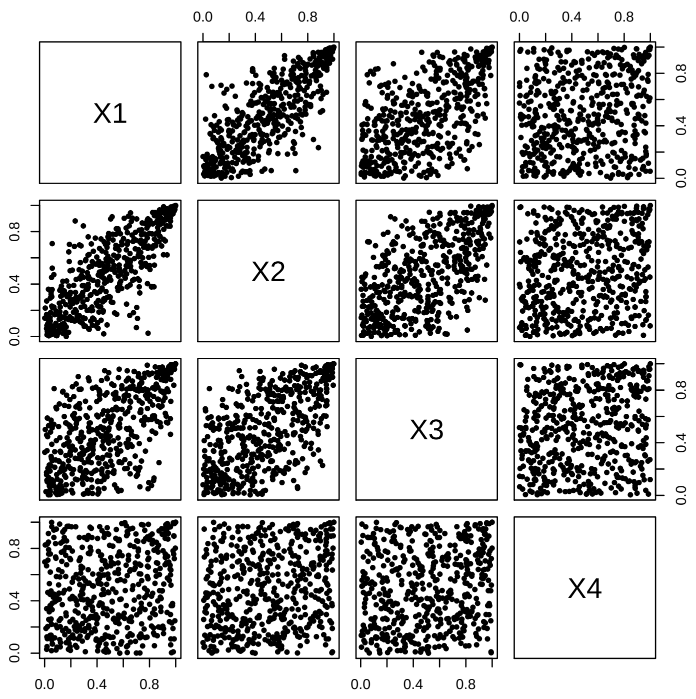


Figure 5: Scatterplot of the sample `sim.data`, which is simulated from `G.cop` associated with a four dimensional HAC-based Gumbel copula.

(a) Sample $R \sim Exp(1)$.

(b) Set the component of U corresponding to u to $\phi_0 (R/V_0)$.

4. Return U.

The function `rHAC` requires only two arguments: (i) the sample size `n` and (ii) an object of the class '`hac`' specifying the characteristics of the underlying HAC, e.g.,

```
[1] "X1 <-> 1"
[1] "X2 <-> 2"
[1] "X3 <-> 3"
[1] "X4 <-> 4"
```

```
R> sim.data = rHAC(500, G.cop)
R> pairs(sim.data, pch = 20)
```

In particular, the contributions of McNeil (2008), Hofert (2008) and Hofert (2011) provide the theoretical foundations to sample computationally efficient random vectors from HAC. Algorithm 1 exploits the recursively determined structure of HAC and samples from F_0 and F_{01}, which are comprehensively discussed in Hofert (2011) and Hofert and Maechler (2011).

3.4. The CDF and density

The arguments for `pHAC` are a '`hac`' object and a sample `X`, whose column names should be identical to the variables' names of the '`hac`' object, e.g.,

```
R> probs = pHAC(X = sim.data, hac = G.cop)
```

As the copula density is defined as dth derivative of the copula $C(\cdot)$ with respect to the arguments u_j, $j = 1, \dots, d$, c.f. Savu and Trede (2010), the explicit form of the density varies with the structure of the underlying HAC. Hence, including the explicit form of all possible d-dimensional copula densities is absolutely unrealistic. Our function `dHAC` derives an analytical expression of the density for a given '`hac`' object, which can be instantaneously evaluated if `eval = TRUE`. The analytical expression of the density is found by subsequently using the `D` function to differentiate the algebraic form of the copula "symbolically" with respect to the variables of the inserted '`hac`' object. Although the derivation and evaluation of the density is computationally and numerically demanding, `dHAC` provides a flexible way to work with HAC densities in practice, because they do not need to be manually derived or numerically approximated. Since the densities of the two-dimensional Archimedean copulae are frequently called during the pseudo multi-stage estimation procedure (1), their closed form expressions are given explicitly.

3.5. Empirical copula

As long as our package does not cover goodness-of-fit tests, which are difficult to implement in general and involve computational intensive techniques via bootstrapping, see Genest, Rémillard, and Beaudoin (2009), it might be difficult to justify the choice of a parametric assumption. However, the values of `probs` can be compared to those of the empirical copula, i.e.,

$$\widehat{C}\left(u_1, \dots, u_d\right) = n^{-1} \sum_{i=1}^{n} \prod_{j=1}^{d} \mathbf{I}\left\{\widehat{F}_j\left(X_{ij}\right) \le u_j\right\}, \qquad (8)$$

where $\widehat{F}_j(\cdot)$ denotes the estimated marginal distribution function of variable X_j. Figure 6 suggests a proper fit of the empirical copula computed by

```
R> probs.emp = emp.copula.self(sim.data, proc = "M")
```

There are two functions which can be used for computing the empirical copula:

```
R> emp.copula(u, x, proc = "M", sort = "none", margins = NULL,
+             na.rm = FALSE, ...)
R> emp.copula.self(x, proc = "M", sort = "none", margins = NULL,
+                  na.rm = FALSE, ...)
```

The difference between the arguments of these functions is that `emp.copula` requires a matrix `u`, at which the estimated function is evaluated. This can, in particular, be helpful, when the sample is decomposed to evaluate the out-of-sample performance as the empirical copula can be regarded as natural benchmark. In contrast, `emp.copula.self` evaluates $\widehat{C}(\cdot)$ at the sample `x` used for the estimation and thus, the returned values can be considered as in-sample fit. The argument `proc` enables the user to choose between two computational methods. We recommend to use the default method, `proc = "M"`, which is based on matrix manipulations,

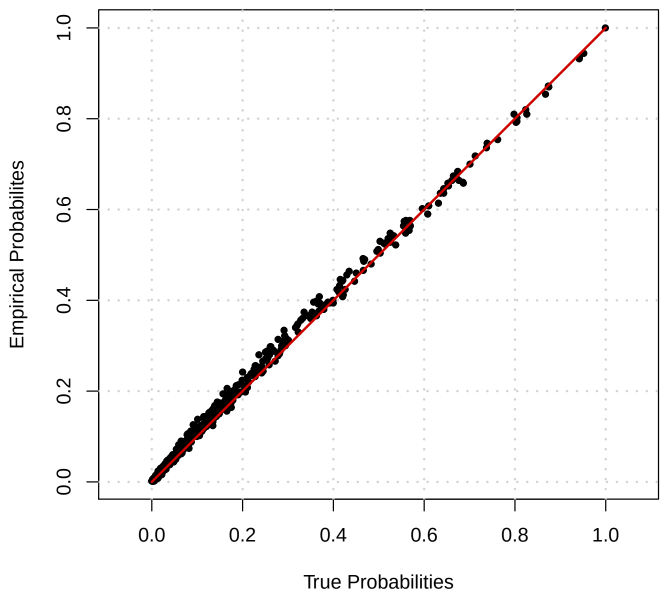


Figure 6: The values of `probs` on the x-axis against the values of `probs.emp`.

because its computational time is just a small fraction of the time taken by method `"A"`, which is based on `apply`, see Figure 7. However, method `"M"` is sensitive with respect to the size of the working memory and therefore inapplicable for very large datasets. Note that standard applications, e.g., measuring the VaR of a portfolio, are based on 250 or 500 observations. Figure 7 illustrates rapidly increasing computational times of the matrix-based method for more than 5000 observations until the method collapses. In contrast, the runtimes of the alternative method `proc = "A"` are more robust against an increasing sample size. The computational times are less sensitive with respect to the dimension and we recommend using the default method up to $d = 100$ for non-large sample sizes. Another possibility to deal with large datasets is specifying the matrix `u` manually in order to reduce the number of vectors which are to be evaluated.

4. Simulation study

To ensure the accuracy of the proposed methods, we generate random data from six copula models of different dimension $C_i^j \overset{\text{def}}{=} C^j(\cdot; s_i, \boldsymbol{\theta}_i)$, for $i = 1, 2, 3$ and $j = \text{C}, \text{G}$, and show that the estimates almost coincide with the true model specification. Here, j denotes the copula family (Clayton or Gumbel) and the structures are given by $s_1 = ((12)3)$, $s_2 = ((((12)3)4)5)$ and $s_3 = ((12)(34)5)$. The values of $\boldsymbol{\theta}_i$ are presented in Tables 3 and 4. They are chosen to obtain a similar strength of dependence by the Clayton and Gumbel based models.

The summary statistics of Tables 3 and 4 rely on $n = 1000$ estimates, whereby only estimates with the same structure can be compared. For this reason the procedure was m times replicated till $n = 1000$ estimates were available. As `estimate.copula` approximates the true structure, we set `epsilon = 0.15` for C_3^{G} and `epsilon = 0.20` for C_3^{C}, which are not based on a binary structure and employ the RML procedure. Note that the RML procedure

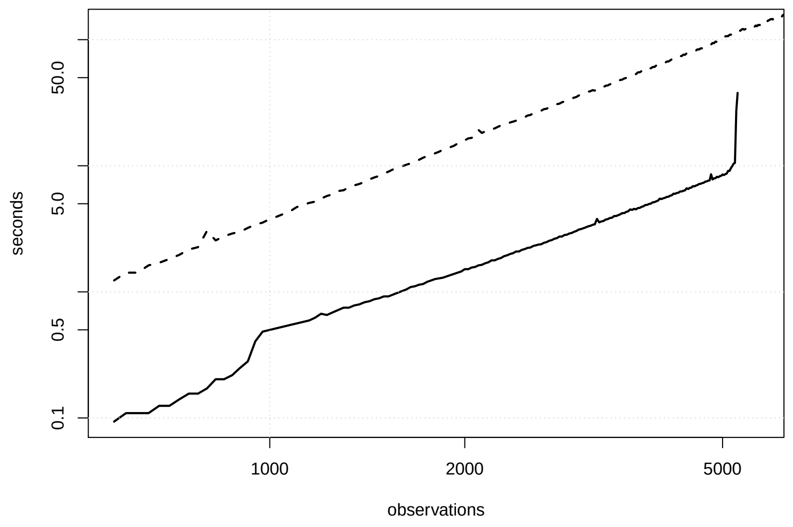


Figure 7: The plot shows the computational times for an increasing sample size but a fixed dimension $d = 5$ on a log-log scale. The solid line refers to `proc = "M"` and the dashed line to `proc = "A"`.

attempts at aggregating the copula tree after each estimation step. The simulated samples for the copula estimation consist of 250 observations for the copula types in order to illustrate the finite sample properties of the procedures. Tables 3 and 4 indicate, that the estimation procedure works properly for the suggested models, as the estimates are on average consistent with the true parameters. Nevertheless, a few points deserve being mentioned: (i) The multi-stage procedure detects the true structure for the binary HAC in $n/m = 100\%$ and the recursive ML procedure for the non-binary HAC in at least $n/m = 99\%$ of the cases as long as the parameters exhibit the imposed distance and the permutation symmetry of the variables at the same node is taken into consideration. (ii) The estimates at lower hierarchical levels show a higher volatility than the estimates close to the initial node and the estimates for the Clayton models are more volatile than the estimates of the Gumbel based HAC. (iii) All estimated models indicate more imprecise estimates for higher nesting levels, but the gains from full ML estimation regarding the precision are only observable for the estimates at the root node of C_3^{G}, see Table 4. However, this minor improvement is costly since the results are based on a preestimated structure. (iv) These observations justify choosing different values of `epsilon` for C_3^{G} and C_3^{C}, as the tuning parameter should reflect the variability of the parameters. Theoretically, `epsilon` can be different for each aggregation of the structure so that the parameter variability is correctly represented. This, however, becomes infeasible in practice, because the number of nodes contained in the true structure is generally unknown. If the parameters are closer and/or the value of `epsilon` is chosen smaller, the amount of correctly classified structures declines. On the other hand, larger sample sizes permit smaller values of `epsilon` as the parameters are more precisely estimated.

5. Conclusion

The **HAC** package focuses on the computationally efficient estimation of hierarchical Archime-

		Statistics				
Model	$\boldsymbol{\theta}$	min	median	mean	max	sd
C_1^{G}	$\theta_2 = 1.500$	1.29	1.50	1.50	1.79	0.07
	$\theta_1 = 3.000$	2.55	3.00	3.00	3.60	0.16
C_1^{C}	$\theta_2 = 1.000$	0.62	1.00	1.00	1.55	0.12
	$\theta_1 = 4.000$	3.26	4.01	4.01	5.00	0.27
C_2^{G}	$\theta_4 = 1.125$	1.00	1.13	1.13	1.29	0.05
	$\theta_3 = 1.500$	1.25	1.50	1.50	1.86	0.08
	$\theta_2 = 2.250$	1.95	2.25	2.25	2.62	0.12
	$\theta_1 = 4.500$	3.79	4.51	4.52	5.51	0.24
C_2^{C}	$\theta_4 = 0.250$	0.02	0.26	0.26	0.57	0.09
	$\theta_3 = 1.000$	0.62	1.00	1.00	1.50	0.12
	$\theta_2 = 2.500$	1.95	2.52	2.52	3.19	0.20
	$\theta_1 = 7.000$	5.95	7.00	7.02	8.40	0.43

Table 3: The models for the Gumbel family C_1^{G}, C_2^{G} and for the Clayton family C_1^{C}, C_2^{C}, where $\boldsymbol{\theta}$ denotes the true copula parameters.

		Statistics for recursive ML					
Model	$\boldsymbol{\theta}$	$\bar{s}$	min	median	mean	max	sd
C_3^{G}	$\theta_3 = 1.125$	$(((34)5)(12)) = 0.50\%$	1.01	1.10	1.11	1.22	0.03
	$\theta_2 = 1.500$	$((534)(12)) = 0.10\%$	1.28	1.50	1.50	1.87	0.07
	$\theta_1 = 3.000$		2.58	2.99	3.00	3.64	0.16
C_3^{C}	$\theta_3 = 0.250$	$(((34)5)(12)) = 0.40\%$	0.09	0.25	0.25	0.46	0.06
	$\theta_2 = 1.000$	$(((12)(34))5) = 0.30\%$	0.62	1.00	1.01	1.42	0.13
	$\theta_1 = 4.000$		3.08	4.00	4.01	5.05	0.29
		Statistics for full ML					
C_3^{G}	$\theta_3 = 1.125$	–	1.05	1.13	1.13	1.23	0.03
	$\theta_2 = 1.500$		1.29	1.50	1.50	1.86	0.07
	$\theta_1 = 3.000$		2.58	2.99	3.00	3.65	0.16
C_3^{C}	$\theta_3 = 0.250$	–	0.13	0.25	0.25	0.42	0.05
	$\theta_2 = 1.000$		0.60	1.00	1.01	1.42	0.13
	$\theta_1 = 4.000$		3.10	4.00	4.01	5.05	0.29

Table 4: The model for the Gumbel family C_3^{G} and for the Clayton family C_3^{C}, where $\boldsymbol{\theta}$ denotes the true copula parameters and the column $\bar{s}$ refers to the percentage of incorrectly classified structures based on $n = 1000$ replications.

dean copulae, which is based on grouping binary structures within a recursive multi-stage ML procedure. Its theoretical and practical advantages are (i) avoiding the demanding asymptotic theory, which arises due to constrained one-step ML estimation and (ii) the consecutive optimization of the log-likelihood instead of the singular optimization of the d-dimensional one with respect to several parameters. Since HAC permit modeling large-dimensional random vectors, the package provides a function for producing plots of the related '`hac`' objects. According to the usual naming of distributions in R, we provide `dHAC`, `pHAC` and `rHAC` to compute the values of density- and distribution functions or to sample from arbitrary HAC.

Finally, the accuracy of the methods has been shown in a small simulation study.

Acknowledgments

The authors are grateful to the editors of the Journal of Statistical Software and two anonymous referees for several helpful comments and suggestions. The research was supported by the Deutsche Forschungsgemeinschaft through the CRC 649 "Economic Risk", Humboldt-Universität zu Berlin and the International Research Training Group 1792.

References

Acar EF, Craiu RV, Yao F (2011). "Dependence Calibration in Conditional Copulas: A Nonparametric Approach." *Biometrics*, **67**(2), 445–453.

Bárdossy A (2006). "Copula-Based Geostatistical Models for Groundwater Quality Parameters." *Water Resources Research*, **42**(11), 1–12.

Bárdossy A, Li J (2008). "Geostatistical Interpolation Using Copulas." *Water Resources Research*, **44**(7), 1–15.

Chen X, Fan Y (2006). "Estimation and Model Selection of Semiparametric Copula-Based Multivariate Dynamic Models under Copula Misspecification." *Journal of Econometrics*, **135**(1–2), 125–154.

Cherubini U, Luciano E, Vecchiato W (2004). *Copula Methods in Finance*. John Wiley & Sons, New York.

Dutang C (2014). "CRAN Task View: Probability Distributions." Version 2014-03-31, URL `http://CRAN.R-project.org/view=Distributions`.

Embrechts P, Lindskog F, McNeil AJ (2003). "Modeling Dependence with Copulas and Applications to Risk Management." In ST Rachev (ed.), *Handbook of Heavy Tailed Distributions in Finance*, pp. 329–384. Elsevier, North-Holland.

Embrechts P, McNeil AJ, Straumann D (1999). "Correlation and Dependence in Risk Management: Properties and Pitfalls." In *Risk Management: Value at Risk and Beyond*, pp. 176–223. Cambridge University Press.

Fan J, Li R (2001). "Variable Selection via Nonconcave Penalized Likelihood and Its Oracle Properties." *Journal of the American Statistical Association*, **96**(456), 1348–1360.

Franke J, Härdle WK, Hafner C (2011). *Statistics of Financial Markets: An Introduction*. Universitext, 3rd edition. Springer-Verlag.

Genest C, Favre AC (2007). "Everything You Always Wanted to Know about Copula Modeling but Were Afraid to Ask." *Journal of Hydrologic Engineering*, **12**(4), 347–368.

Genest C, Rémillard B, Beaudoin D (2009). "Goodness-of-Fit Tests for Copulas: A Review and a Power Study." *Insurance: Mathematics and Economics*, **44**(2), 199–213.

Górecki J, Hofert M, Holeňa M (2014). "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.

Hofert M (2008). "Sampling Archimedean Copulas." *Computational Statistics & Data Analysis*, **52**(12), 5163–5174.

Hofert M (2011). "Efficiently Sampling Nested Archimedean Copulas." *Computational Statistics & Data Analysis*, **55**(1), 57–70.

Hofert M, Kojadinovic I, Maechler M, Yan J (2014). ***copula**: Multivariate Dependence with Copulas.* R package version 0.999-9, URL `http://CRAN.R-project.org/package=copula`.

Hofert M, Maechler M (2011). "Nested Archimedean Copulas Meet R: The **nacopula** Package." *Journal of Statistical Software*, **39**(9), 1–20. URL `http://www.jstatsoft.org/v39/i09/`.

Hofert M, Maechler M (2012). ***nacopula**: Nested Archimedean Copulas.* R package version 0.8-1, URL `http://CRAN.R-project.org/package=nacopula`.

Jaworski P, Durante F, Härdle WK (2013). *Copulae in Mathematical and Quantitative Finance*, volume 213 of *Lecture Notes in Statistics*. Springer-Verlag.

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

Junker M, May A (2005). "Measurement of Aggregate Risk with Copulas." *Econometrics Journal*, **8**(3), 428–454.

Kojadinovic I, Yan J (2010). "Modeling Multivariate Distributions with Continuous Margins Using the **copula** R Package." *Journal of Statistical Software*, **34**(9), 1–20. URL `http://www.jstatsoft.org/v34/i09/`.

Lakhal-Chaieb ML (2010). "Copula Inference under Censoring." *Biometrika*, **97**(2), 505–512.

Li DX (2000). "On Default Correlation: A Copula Function Approach." *Journal of Fixed Income*, **9**(4), 43–54.

McNeil AJ (2008). "Sampling Nested Archimedean Copulas." *Journal of Statistical Computation and Simulation*, **78**(6), 567–581.

McNeil AJ, Nešlehová J (2009). "Multivariate Archimedean Copulas, d-Monotone Functions and l_1 Norm Symmetric Distributions." *The Annals of Statistics*, **37**(5b), 3059–3097.

McNeil AJ, Ulman S (2011). ***QRMlib**: Provides R Language Code to Examine Quantitative Risk Management Concepts.* R package version 1.4.5.1, URL `http://CRAN.R-project.org/pacakge=QRMlib/`.

Mendes BVM, Souza RM (2004). "Measuring Financial Risks with Copulas." *International Review of Financial Analysis*, **13**(1), 53–77.

Nelsen RB (2006). *An Introduction to Copulas*. 2nd edition. Springer-Verlag, New York.

Okhrin O, Okhrin Y, Schmid W (2013a). "On the Structure and Estimation of Hierarchical Archimedean Copulas." *Journal of Econometrics*, **173**(2), 189–204.

Okhrin O, Okhrin Y, Schmid W (2013b). "Properties of Hierarchical Archimedean Copulas." *Statistics & Risk Modeling*, **30**(1), 21–54.

Okhrin O, Ristig A (2014). ***HAC**: Estimation, Simulation and Visualization of Hierarchical Archimedean Copulae (HAC).* R package version 0.3-2, URL http://CRAN.R-project.org/package=HAC.

Pfaff B, McNeil A (2013). ***QRM**: Provides R Language Code to Examine Quantitative Risk Management Concepts.* R package version 0.4-9, URL http://CRAN.R-project.org/package=QRM.

R Core Team (2014). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Savu C, Trede M (2010). "Hierarchies of Archimedean Copulas." *Quantitative Finance*, **10**(3), 295–304.

Sklar A (1959). "Fonctions de Répartition à n Dimension et Leurs Marges." *Publications de l'Institut de Statistique de l'Université de Paris*, **8**, 299–231.

Whelan N (2004). "Sampling from Archimedean Copulas." *Quantitative Finance*, **4**(3), 339–352.

Wuertz D, *et al.* (2013). ***fCopulae**: Rmetrics – Dependence Structures with Copulas.* R package version 3000.79, URL http://CRAN.R-project.org/package=fCopulae.

Yan J (2007). "Enjoy the Joy of Copulas: With a Package **copula**." *Journal of Statistical Software*, **21**(4), 1–21. URL http://www.jstatsoft.org/v21/i04/.

Affiliation:

Ostap Okhrin
Chair of Econometrics and Statistics esp. Transportation
Institute of Economics and Transport
Faculty of Transportation
Dresden University of Technology
01069 Dresden, Germany
E-mail: ostap.okhrin@tu-dresden.de
URL: https://www.wiwi.hu-berlin.de/professuren/quantitativ/statistik/members/personalpages/o2

Alexander Ristig
C.A.S.E - Center for Applied Statistics and Economics
Ladislaus von Bortkiewicz Chair of Statistics
Humboldt-Universität zu Berlin
10099 Berlin, Germany

E-mail: alexander.ristig@hu-berlin.de
URL: https://www.wiwi.hu-berlin.de/professuren/quantitativ/statistik/members/personalpages/ar