Nested Archimedean Copulas Meet R— The nacopula Package

Marius Hofert*
ETH Zurich

Martin Mächler
ETH Zurich

Abstract

The package **copula** (formerly **nacopula**) provides procedures for constructing nested Archimedean copulas in any dimensions and with any kind of nesting structure, generating vectors of random variates from the constructed objects, computing function values and probabilities of falling into hypercubes, as well as evaluation of characteristics such as Kendall's tau and the tail-dependence coefficients. As by-products, algorithms for various distributions, including exponentially tilted stable and Sibuya distributions, are implemented. Detailed examples are given.

Keywords: Archimedean copulas, nested Archimedean copulas, sampling algorithms, Kendall's tau, tail-dependence coefficients, exponentially tilted stable distribution, R.

1. Introduction [1]

A *copula* is a multivariate distribution function with standard uniform univariate margins. Standard references for an introduction are Joe (1997) or Nelsen (2007).

Sklar (1959) shows that for any multivariate distribution function H with margins F_j, $j \in \{1, \dots, d\}$, there exists a copula C such that

$$H(x_1, \dots, x_d) = C(F_1(x_1), \dots, F_d(x_d)), \ \boldsymbol{x} \in \mathbb{R}^d. \tag{1}$$

Conversely, given a copula C and arbitrary univariate distribution functions F_j, $j \in \{1, \dots, d\}$, H defined by (1) is a distribution function with marginals F_j, $j \in \{1, \dots, d\}$. On one hand, Sklar's Theorem tells us that we can decompose any given multivariate distribution function into its margins and a copula. By this decomposition, copulas allow us to study multivariate distributions functions independently of the margins. This is of particular interest in statistics. On the other hand, Sklar's Theorem provides a tool for constructing large classes of multivariate distributions and is therefore often used for sampling multivariate distributions via copulas. This is indispensable for many applications in the areas of statistics and finance. For sampling the multivariate distribution H it suffices to sample the common dependence structure, given by the copula C, and to transform the obtained variates to the correct margins F_j, $j \in \{1, \dots, d\}$. Since this transformation is usually easy to achieve (simply apply the *generalized inverse* $F_j^-(y) = \inf\{x \in \mathbb{R} : F_j(x) \geq y\}$ corresponding to F_j, $j \in \{1, \dots, d\}$, with the convention that $\inf \emptyset = \infty$), sampling from H usually boils down to sampling the copula C under consideration.

*The author (Willis Research Fellow) thanks Willis Re for financial support while this work was completed.

[1] a version of this paper, for **nacopula** 0.4_4, has been published in JSS, http://www.jstatsoft.org/v39/i09.

Alongside *elliptical copulas*, i.e., the copulas arising from elliptical distributions via Sklar's Theorem (see, e.g., Embrechts, Lindskog, and McNeil (2003)), Archimedean copulas play an important role in practical applications. In contrast to elliptical ones, Archimedean copulas are given explicitly in terms of a generator. They are able to capture different kinds of tail dependencies, e.g., only upper tail dependence and no lower tail dependence or both lower and upper tail dependence but of different magnitude. With the algorithm of Marshall and Olkin (1988), Archimedean copulas are usually easy to sample. Their functional symmetry (in u_j, $j = 1, \dots, d$), also referred to as *exchangeability*, however, is often considered to be a drawback, e.g., in risk-management applications where the considered portfolios are typically high-dimensional. To circumvent exchangeability, Archimedean copulas can be nested within each other under certain conditions. The resulting copulas are referred to as "nested Archimedean copulas" and allow to model hierarchical dependence structures.

The R package **copula** (formerly **nacopula**) implements several functions for working with Archimedean and nested Archimedean copulas. In contrast to other R packages dealing with Archimedean copulas, e.g., **copula** (Yan (2007); Kojadinovic and Yan (2010)) or **fCopulae** (Wuertz *et al.* (2009)), particular focus is put on nested Archimedean copulas. The R package **nacopula** had been the first R package dealing with these functions, and is now part of the R package **copula**. It is available from the Comprehensive R Archive Network at `https://CRAN.R-project.org/package=copula`.

2. Archimedean copulas

2.1. Archimedean copulas and their properties

An *Archimedean generator*, or simply *generator*, is a continuous, decreasing function $\psi : [0, \infty] \to [0, 1]$ which satisfies $\psi(0) = 1$, $\psi(\infty) := \lim_{t\to\infty} \psi(t) = 0$, and which is strictly decreasing on $[0, \inf\{t : \psi(t) = 0\}]$. A d-dimensional copula is called *Archimedean* if it is of the form

$$C(\boldsymbol{u}; \psi) = \psi(\psi^{-1}(u_1) + \cdots + \psi^{-1}(u_d)), \ \boldsymbol{u} \in [0, 1]^d, \tag{2}$$

for some generator ψ with inverse $\psi^{-1} : [0, 1] \to [0, \infty]$, where $\psi^{-1}(0) = \inf\{t : \psi(t) = 0\}$. McNeil and Nešlehová (2009) show that a generator defines a d-dimensional Archimedean copula if and only if ψ is *d-monotone*, i.e., ψ is continuous on $[0, \infty]$, admits derivatives up to the order $d - 2$ satisfying $(-1)^k \frac{d^k}{dt^k}\psi(t) \geq 0$ for all $k \in \{0, \dots, d - 2\}$, $t \in (0, \infty)$, and $(-1)^{d-2} \frac{d^{d-2}}{dt^{d-2}}\psi(t)$ is decreasing and convex on $(0, \infty)$. A necessary and sufficient condition for an Archimedean generator ψ to generate a proper copula in all dimensions d is that ψ is *completely monotone*, i.e., $(-1)^k \psi^{(k)}(t) \geq 0$ for all $t \in (0, \infty)$ and $k \in \mathbb{N}_0$, see Kimberling (1974) in the context of t-norms or Hofert (2010, p. 54) for a reworking in terms of copulas. Unless otherwise stated, we assume ψ to be completely monotone in what follows. Finally, let us note that the most simple dependence model, namely, independence, is provided by $\psi(t) = \exp(-t)$, with $\psi^{-1}(t) = -\log(t)$, and corresponding *independence* copula $C(\boldsymbol{u}) = \prod_{j=1}^d u_j$.

The class of all completely monotone Archimedean generators is denoted by Ψ_∞ in what follows. Bernstein's Theorem, see, e.g., Feller (1971, p. 439), shows that this class coincides with the class of Laplace-Stieltjes transforms of distribution functions F on the positive real line, where the Laplace-Stieltjes transform of F, also known as the *Laplace transform of the*

distribution, is defined as

$$\mathcal{LS}[F](t) := \int_0^\infty \exp(-tx)\, dF(x),\ t \in [0, \infty).$$

For a $\psi \in \Psi_\infty$, we hence have the relation

$$\psi = \mathcal{LS}[F],\ \text{or, equivalently, } F = \mathcal{LS}^{-1}[\psi].$$

for a distribution function F on the positive real line.

Note that the distribution function F is known for virtually all commonly used Archimedean generators, see, e.g., Hofert (2010, p. 62). The package **copula** currently provides the most widely used families of Ali-Mikhail-Haq, Clayton, Frank, Gumbel, and Joe; see Table 1 for the generators and their corresponding distribution functions. Except for Clayton's family, where we use a slightly simpler generator, these generators are the ones given in Nelsen (2007, pp. 116). Also note that for the families of Ali-Mikhail-Haq, Frank, and Joe, F is discrete with support $\mathbb{N}$. In this case, ψ is of the form $\psi(t) = \sum_{k=1}^\infty p_k \exp(-kt)$, $t \in [0, \infty]$, where $(p_k)_{k=1}^\infty$ denotes the probability mass function corresponding to F.

These Archimedean copula families are provided as `"acopula"` R objects (for more information about the statistical software R, see R Development Core Team (2010, 2.11.1)), containing as slots the corresponding generator ψ, `psi`, its inverse ψ^{-1}, `iPsi`, and the "sampler", i.e., random number generator for $V \sim F$, as `V0`:

```
R> require(copula)
R> ls("package:copula", pattern = "^cop[A-Z]")

[1] "copAMH"     "copClayton" "copFrank"   "copGumbel"  "copJoe"

R> copClayton

Archimedean copula ("acopula"), family "Clayton"
 It contains further slots, named
  "psi", "iPsi", "paraInterval", "absdPsi", "paraConstr",
  "absdiPsi", "dDiag", "dacopula", "score", "uscore", "V0",
  "dV0", "tau", "iTau", "lambdaL", "lambdaLInv", "lambdaU",
  "lambdaUInv", "nestConstr", "V01", "dV01"

R> copClayton@psi

function (t, theta)
pmax(1 + sign(theta) * t, 0)^(-1/theta)
<bytecode: 0x5635e8094390>
<environment: namespace:copula>

R> copClayton@iPsi # the inverse of psi(), psi^{-1}

function (u, theta, log = FALSE)
{
    res <- .iPsiClayton(u, theta)
    if (log)
        log(res)
    else res
}
<bytecode: 0x5635e80aadb0>
<environment: 0x5635e80938d8>
```

```
R> copClayton@V0     # "sampler" for  V ~ F()
function (n, theta, log = FALSE)
{
    if (log)
        stop("'log=TRUE' not yet implemented")
    else rgamma(n, shape = 1/theta)
}
<bytecode: 0x5635e80bede0>
<environment: 0x5635e80938d8>
```

The majority of slots of such copula objects are functions, encoding properties of that copula family. In what follows, some of these functions are presented.

Sampling Archimedean copulas

From a mixture representation with respect to F, the following algorithm may be derived for sampling Archimedean copulas, see Marshall and Olkin (1988).

Algorithm 2.1 (Marshall and Olkin (1988))
(1) **sample** $V \sim F = \mathcal{LS}^{-1}[\psi]$
(2) **sample** $R_j \sim \mathrm{Exp}(1)$, $j \in \{1, \dots, d\}$
(3) **set** $U_j = \psi(R_j/V)$, $j \in \{1, \dots, d\}$
(4) **return** $\boldsymbol{U} = (U_1, \dots, U_d)^\mathsf{T}$

In order for this algorithm to be easily applied, we need to know how to sample the distribution functions $F = \mathcal{LS}^{-1}[\psi]$. For the families of Ali-Mikhail-Haq, Clayton, Frank, Gumbel, and Joe, see Table 1.

Family	Parameter	$\psi(t)$	$V \sim F = \mathcal{LS}^{-1}[\psi]$
Ali-Mikhail-Haq	$\theta \in [0, 1)$	$(1-\theta)/(\exp(t)-\theta)$	$\mathrm{Geo}(1-\theta)$
Clayton	$\theta \in (0, \infty)$	$(1+t)^{-1/\theta}$	$\Gamma(1/\theta, 1)$
Frank	$\theta \in (0, \infty)$	$-\log(1-(1-e^{-\theta})\exp(-t))/\theta$	$\mathrm{Log}(1-e^{-\theta})$
Gumbel	$\theta \in [1, \infty)$	$\exp(-t^{1/\theta})$	$\mathrm{S}(1/\theta, 1, \cos^\theta(\pi/(2\theta)), \mathbf{1}_{\{\theta=1\}}; 1)$
Joe	$\theta \in [1, \infty)$	$1-(1-\exp(-t))^{1/\theta}$	$\mathrm{Sibuya}(1/\theta)$

Table 1: Commonly used one-parameter Archimedean generators.

For the family of Ali-Mikhail-Haq, $\mathrm{Geo}(p)$ denotes a *geometric distribution* with success probability $p \in (0, 1]$ and probability mass function $p_k = p(1-p)^{k-1}$ at $k \in \mathbb{N}$ (which in R is `dgeom(k, p)`). For Clayton's family, $\Gamma(\alpha, \beta)$ denotes the *gamma distribution* with shape $\alpha \in (0, \infty)$, rate $\beta \in (0, \infty)$, and density $f(x) = \beta^\alpha x^{\alpha-1}\exp(-\beta x)/\Gamma(\alpha)$, $x \in (0, \infty)$ (provided in R by `[dpqr]gamma(., shape=`α`, rate=`β`)`). For the family of Frank, $\mathrm{Log}(p)$ denotes a *logarithmic distribution* with parameter $p \in (0, 1)$ and mass function $p_k = p^k/(-k\log(1-p))$ at $k \in \mathbb{N}$. For sampling this distribution, we provide `rlog(., p)`, using the algorithm "LK" of Kemp (1981). For Gumbel's family, F corresponds to a $S(\alpha, \beta, \gamma, \delta; 1)$, i.e., a *stable*, distribution with characteristic function

$$\phi(t) = \exp\big(i\delta t - \gamma^\alpha |t|^\alpha (1 - i\beta\,\mathrm{sgn}(t) w(t, \alpha))\big),\ t \in \mathbb{R}, \tag{3}$$

where

$$w(t,\alpha) = \begin{cases} \tan(\alpha\pi/2), & \alpha \neq 1, \\ -2\log(|t|)/\pi, & \alpha = 1, \end{cases}$$

see, e.g., Nolan (2011, p. 8) for this "1-parameterization". For sampling from $S(.)$, we provide `rstable1(., α, β, γ, δ, 1)`, having implemented an algorithm for sampling stable distributions according to the ideas presented in Chambers, Mallows, and Stuck (1976), and improving on previous implementations in R. For the family of Joe, Sibuya(α) denotes a *Sibuya distribution* with probability mass function $p_k = \binom{\alpha}{k}(-1)^{k-1}$ at $k \in \mathbb{N}$, where $\alpha \in (0,1]$; $\binom{\alpha}{k} = \alpha(\alpha-1)\ldots(\alpha-k+1)/k!$ denotes the (generalized) binomial coefficient, which is implemented in R via `choose(α, k)`. This distribution can be sampled via the R function `rSibuya(., α)` which we implemented based on an algorithm presented in Hofert (2011).

The rank-correlation coefficient Kendall's tau

In practical applications, it is often desirable to measure the degree of dependence between random variables by a real number, a generalized correlation. Such measures are referred to as *measures of association* and are usually studied for in the bivariate case, i.e., for pairs of random variables. One such measure is Kendall's tau. For a bivariate vector of continuously distributed random variables $(X_1, X_2)^\mathsf{T}$, *Kendall's tau* is defined by

$$\begin{aligned} \tau &= \mathsf{E}[\operatorname{sign}((X_1 - X_1')(X_2 - X_2'))] \\ &= \mathsf{P}((X_1 - X_1')(X_2 - X_2') > 0) - \mathsf{P}((X_1 - X_1')(X_2 - X_2') < 0), \end{aligned}$$

where $(X_1', X_2')^\mathsf{T}$ is an independent and identically distributed copy of $(X_1, X_2)^\mathsf{T}$ and $\operatorname{sign}(x) = \mathbf{1}_{(0,\infty)}(x) - \mathbf{1}_{(-\infty,0)}(x)$ denotes the signum function (as R's `sign(x)`). Since Kendall's tau equals the probability of concordance minus the probability of discordance, it is a measure of concordance, see, e.g., Scarsini (1984). Informally, it measures, as a number in $[-1,1]$, the probability with which large values of one variable are associated with large values of the other. If C is a bivariate Archimedean copula generated by a twice continuously differentiable generator ψ with $\psi(t) > 0$, $t \in [0,\infty)$, Kendall's tau can be represented in semi-closed form as

$$\tau = 1 + 4\int_0^1 \frac{\psi^{-1}(t)}{(\psi^{-1}(t))'}\,dt = 1 - 4\int_0^\infty t(\psi'(t))^2\,dt,$$

see Joe (1997, p. 91). For the Archimedean families of Ali-Mikhail-Haq, Clayton, and Gumbel, this integral can be evaluated explicitly, for Frank's family it involves the *Debye function of order one*, i.e., $D_1(\theta) = \frac{1}{\theta}\int_0^\theta t/(e^t-1)\,dt$, and for Joe's family, it is given as a series, see Table 2.

Tail-dependence coefficients

Another notion of association is tail dependence. Tail dependence measures the probability that one random variable takes on values in its tail, given the other one takes on values in its tail. To be more precise, if $X_j \sim F_j$, $j \in \{1,2\}$, are continuously distributed random variables, the *lower tail-dependence coefficient*, respectively the *upper tail-dependence coefficient*, of X_1 and X_2 are defined as

$$\lambda_l = \lim_{t\downarrow 0} \mathsf{P}(X_2 \le F_2^-(t) \,|\, X_1 \le F_1^-(t)), \quad \lambda_u = \lim_{t\uparrow 1} \mathsf{P}(X_2 > F_2^-(t) \,|\, X_1 > F_1^-(t)),$$

provided that the limits exist. These measures of association can be expressed in terms of the copula C corresponding to $(X_1, X_2)^\mathsf{T}$. If C is a bivariate Archimedean copula generated by ψ with $\psi(t) > 0$, $t \in [0, \infty)$, then

$$\lambda_l = \lim_{t\to\infty} \frac{\psi(2t)}{\psi(t)} = 2 \lim_{t\to\infty} \frac{\psi'(2t)}{\psi'(t)}, \quad \lambda_u = 2 - \lim_{t\downarrow 0} \frac{1-\psi(2t)}{1-\psi(t)} = 2 - 2\lim_{t\downarrow 0} \frac{\psi'(2t)}{\psi'(t)},$$

where the equalities involving derivatives are obtained by l'Hôpital's rule and therefore the corresponding assumptions are required to hold. For the implemented Archimedean families, these limits can easily be found and are given in Table 2.

Family	Parameter	τ	λ_l	λ_u
Ali-Mikhail-Haq	$\theta \in [0, 1)$	$1 - 2(\theta + (1-\theta)^2 \log(1-\theta))/(3\theta^2)$	0	0
Clayton	$\theta \in (0, \infty)$	$\theta/(\theta + 2)$	$2^{-1/\theta}$	0
Frank	$\theta \in (0, \infty)$	$1 + 4(D_1(\theta) - 1)/\theta$	0	0
Gumbel	$\theta \in [1, \infty)$	$(\theta - 1)/\theta$	0	$2 - 2^{1/\theta}$
Joe	$\theta \in [1, \infty)$	$1 - 4\sum_{k=1}^{\infty} 1/(k(\theta k + 2)(\theta(k-1) + 2))$	0	$2 - 2^{1/\theta}$

Table 2: Kendall's tau and tail-dependence coefficients for commonly used one-parameter Archimedean generators.

2.2. A three-dimensional Joe copula

As an example, we define a three-dimensional Joe copula with parameter chosen such that Kendall's tau (for the corresponding bivariate marginal copula of the Archimedean type) equals 0.5; this is achieved with the function `iTau`, which computes the parameter θ such that Kendall's tau equals 0.5.

```
R> (theta <- copJoe@iTau(0.5))

[1] 2.856238

R> C3joe.5 <- onacopula("Joe", C(theta, 1:3))
```

The internal structure of this object[2] is

```
R> str(C3joe.5)  # str[ucture] of object

Formal class 'outer_nacopula' [package "copula"] with 3 slots
  ..@ copula   :Formal class 'acopula' [package "copula"] with 23 slots
  .. .. ..@ name        : chr "Joe"
  .. .. ..@ psi         :function (t, theta)
  .. .. ..@ iPsi        :function (u, theta, log = FALSE)
  .. .. ..@ paraInterval:Formal class 'interval' [package "copula"] with 2 slots
  .. .. .. .. ..@ .Data: num [1:2] 1 Inf
  .. .. .. .. ..@ open : logi [1:2] FALSE TRUE
  .. .. ..@ absdPsi     :function (t, theta, degree = 1, n.MC = 0, method = eval(formals(polyJ)$met
    log = FALSE)
  .. .. ..@ theta       : num 2.86
  .. .. ..@ paraConstr  :function (theta, dim)
```

[2]Note that we use a parametric nested Archimedean copula (see below), without any nesting, and corresponding `*nacopula()` functions, since they generalize the present non-nested case.

```
.. .. ..@ absdiPsi    :function (u, theta, degree = 1, log = FALSE)
.. .. ..@ dDiag       :function (u, theta, d, log = FALSE)
.. .. ..@ dacopula    :function (u, theta, n.MC = 0, checkPar = TRUE, method = eval(formals(polyJ
  log = FALSE)
.. .. ..@ score       :function (u, theta, method = eval(formals(polyJ)$method))
.. .. ..@ uscore      :function (u, theta, d, method = eval(formals(polyJ)$method))
.. .. ..@ V0          :function (n, theta, log = FALSE)
.. .. ..@ dV0         :function (x, theta, log = FALSE)
.. .. ..@ tau         :function (theta, method = c("hybrid", "digamma", "sum"), noTerms = 446)
.. .. ..@ iTau        :function (tau, tol = .Machine$double.eps^0.25, ...)
.. .. ..@ lambdaL     :function (theta)
.. .. ..@ lambdaLInv  :function (lambda)
.. .. ..@ lambdaU     :function (theta)
.. .. ..@ lambdaUInv  :function (lambda)
.. .. ..@ nestConstr  :function (theta0, theta1)
.. .. ..@ V01         :function (V0, theta0, theta1, approx = 10000)
.. .. ..@ dV01        :function (x, V0, theta0, theta1, method = eval(formals(dsumSibuya)$method)
  log = FALSE)
..@ comp     : int [1:3] 1 2 3
..@ childCops: list()
```

Figure 1 visualizes 500 vectors of random variates (each in $[0,1]^3$) from this copula with a scatter-plot matrix. The `splom2()` function is a version of standard's **lattice** (Sarkar (2008)) `splom()`, with the addition of using nice labels $U_1, U_2, \dots, U_d$ by default.

```
R> require(lattice)
R> set.seed(1)
R> dim(U3 <- rnacopula(500, C3joe.5))
```

```
[1] 500   3
```

The matrix of pairwise sample versions of Kendall's tau corresponding to the generated data is given by

```
R> round(cor(U3, method="kendall"), 3)
```

```
      [,1]  [,2]  [,3]
[1,] 1.000 0.467 0.476
[2,] 0.467 1.000 0.483
[3,] 0.476 0.483 1.000
```

Note that the entries are close to 0.5 which is the chosen population version of Kendall's tau for the simulated data.

Next, let us evaluate this Joe copula at $(0.5, 0.5, 0.5)^\mathsf{T}$ and $(0.99, 0.99, 0.99)^\mathsf{T}$.

```
R> c(pnacopula(C3joe.5, c(.5, .5, .5)),
+    pnacopula(C3joe.5, c(.99,.99,.99)))
```

```
[1] 0.3009056 0.9853092
```

Now let us answer the question what the probability is for $\boldsymbol{U}$ to fall in the cube $(0.8, 1]^3$.

```
R> prob(C3joe.5, c(.8, .8, .8), c(1, 1, 1))
```

```
[1] 0.1293358
```

Finally, the lower and upper tail-dependence coefficients for this copula can be obtained as follows.

```
R> splom2(U3, cex = 0.4)
```

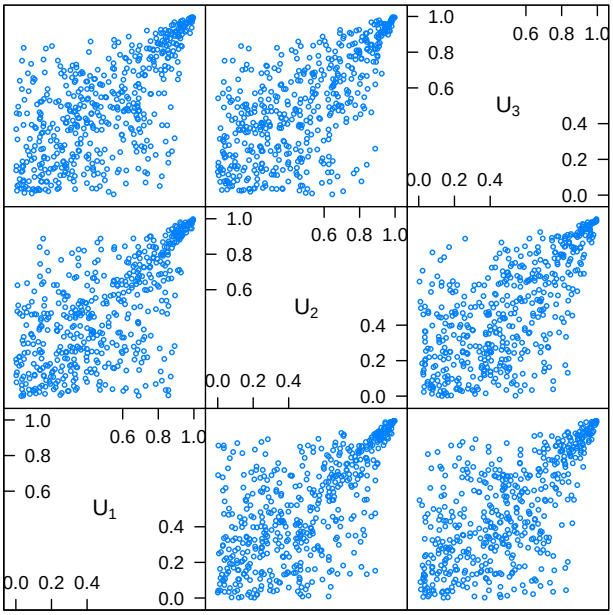

Figure 1: 500 vectors of random variates following a trivariate Joe copula with (pairwise) Kendall's tau equal to 0.5.

```
R> c(copJoe@lambdaL(theta),
+    copJoe@lambdaU(theta))
[1] 0.0000000 0.7253414
```

3. Nested Archimedean copulas

3.1. Construction

In contrast to elliptical copulas, Archimedean copulas can capture different tail dependencies, i.e., $\lambda_l \neq \lambda_u$. Further, they are given explicitly, which facilitates computing probabilities for such dependence models. However, the exchangeability inherent in Archimedean copulas implies that all margins of the same dimension are equal. For modeling purposes, this becomes an increasingly strong assumption in the dimension. Asymmetries, i.e., more realistic dependencies, can be modeled by a hierarchical structure of Archimedean copulas, obtained by plugging in Archimedean copulas into each other. In practical applications, these hierarchical structures are often naturally motivated, e.g., by different macroeconomic effects, political decisions, or consumer trends affecting the components of a random vector.

A d-dimensional copula C is called *nested Archimedean* if it is an Archimedean copula with arguments possibly replaced by other nested Archimedean copulas. If C is given recursively by (2) for $d = 2$ and, up to permutation of the arguments, for $d \geq 3$, by

$$C(u_1, \dots, u_d; \psi_0, \dots, \psi_{d-2}) = \psi_0\big(\psi_0^{-1}(u_1) + \psi_0^{-1}(C(u_2, \dots, u_d; \psi_1, \dots, \psi_{d-2}))\big), \qquad (4)$$

then C is called *fully nested Archimedean copula* with $d-1$ *nesting levels* or *hierarchies*. Otherwise, C is called *partially nested Archimedean copula*. Fully and partially nested Archimedean copulas are summarized as *nested* (or *hierarchical*) *Archimedean copulas*.

Note that the structure of a nested Archimedean copula can be depicted by a tree. For example, the three-dimensional nested Archimedean copula of Type (4) involving the generators ψ_0 and ψ_1, which is given by

$$C(u_1, u_2, u_3) = C(u_1, C(u_2, u_3; \psi_1); \psi_0),$$

can be depicted by the tree structure as given in Figure 2. Due to this tree representation, we

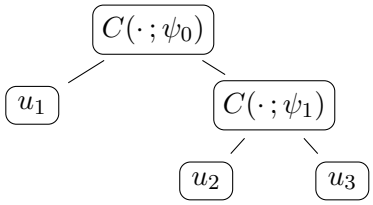

Figure 2: Tree structure of a three-dimensional fully nested Archimedean copula.

refer to the outermost Archimedean copula (generated by ψ_0) as *root copula*. Further, we call an Archimedean copula *parent copula* if it has at least one nested Archimedean copula as one of its components in the tree structure of a nested Archimedean copula. We refer to these components as *child copulas*. For example, the three-dimensional nested Archimedean copula as given in Figure 2 has parent copula $C(\cdot;\psi_0)$ (which is also the root copula here) with child copula $C(\cdot;\psi_1)$; as another example, $C(\cdot;\psi_1)$ in Figure 3 is the parent copula of the child copula $C(\cdot;\psi_2)$ and $C(\cdot;\psi_0)$ is the parent of $C(\cdot;\psi_1)$.

In R, because of the recursive tree structure, a powerful approach is to use a recursive class definition: In the **copula** package, we define the `nacopula` (**n**ested **a**rchimedean **copula**) class, with three components (`slots`),

```
setClass("nacopula",
         representation(copula    = "acopula",
                        comp      = "integer",  # from 1:d -- of length in [0,d]
                        childCops = "list"      # of nacopulas, possibly empty
                       ),
         validity = function(object) {
            ......
             if(!all("nacopula" == sapply(object@childCops, class)))
                 return("All 'childCops' elements must be 'nacopula' objects")
            ......
         })
```

i.e., by its root copula (slot `@copula`, an `"acopula"` object), a vector of indices of its "direct components" (slot `@comp = 1` for u_1 in the example), and a list of child copulas (slot `@childCops`). The `childCops` list is the tool to contain sub branches of the "tree" of nested copulas, and will be empty, i.e., `list()`, for the final nodes (referred to as *leaves*) of the tree. In the example, it will be of length one, containing a (non-nested) bivariate copula for the components u_2 and u_3.

The class `outer_nacopula` is a version of the same class (i.e., it "contains" `nacopula` without any further slots), just with stricter validity checking, namely requiring that all components

from all child copulas are exactly the set $\{1, 2, \ldots, d\}$.

```
setClass("outer_nacopula", contains = "nacopula",
         validity = function(object) {
             ## *Extra* checks in addition to those of "nacopula"
             .......
         })
```

The `onacopula()` function[3] allows to specify (outer) `nacopula`s with convenient specification of the nesting structure and parameters for each copula, using a "formula"-like notation $C(\theta, c(i_1, \ldots, i_c), \texttt{list}(C(..), \ldots, C(..)))$ similar to the mathematical notation above; for more, see its help page. For example, (one parametrization of) the three-dimensional example from Figure 2 is

```
R> ( C3 <- onacopula("A", C(0.2, 1, C(0.8, 2:3))) )

Nested Archimedean copula ("outer_nacopula" of dim. 3), with slot
'comp'   = (1)  and root
'copula' = Archimedean copula ("acopula"), family "AMH", theta= (0.2)
and 1 child copula
   Nested Archimedean copula ("nacopula"), with slot
   'comp'   = (2, 3)  and root
   'copula' = Archimedean copula ("acopula"), family "AMH", theta= (0.8)
   and *no* child copulas
```

This is a shortened form of the following.

```
R> stopifnot(identical(C3,
+        onacopula("A", C(0.2, 1, list(C(0.8, 2:3, list()))))
+  ))
```

The recursive definition (4) of nested Archimedean copulas not only leads to recursive class definitions in R, but also to recursive functions for computations involving such "nacopulas". All the following functions and methods (from our package `copula`) are defined recursively, typically using `lapply(x@childCops, <fun>)`: The utilities `dim()`, `allComp()`, `printNacopula()` (which is the hidden `show()` method), and the principal functions `pnacopula()` and `rncopula()` (via recursive utility `rnchild()`). As a simple example of these, `pnacopula(x, u)` simply evaluates the (recursive) Formula (4), recursively applying itself to its child copulas:

```
pnacopula <- function(x,u) {
    stopifnot(is.numeric(u), 0 <= u, u <= 1, length(u) >= dim(x)) # can be larger
    C <- x@copula
    th <- C@theta
    ## use u[j] for the direct components 'comp':
    C@psi(sum(unlist(lapply(u[x@comp], C@iPsi, theta=th)),
              C@iPsi(unlist(lapply(x@childCops, pnacopula, u = u)),
                     theta=th)),
          theta=th)
}
```

In order for (4) to be indeed a proper copula, Joe (1997, p. 88) and McNeil (2008) present the *sufficient nesting condition* that $\psi_i^{-1} \circ \psi_j$ is completely monotone for all nodes (with parent i and child j) appearing in a nested Archimedean copula. This condition can be derived from a mixture representation of C based on the distribution functions $F_0 = \mathcal{LS}^{-1}[\psi_0]$ and

[3] In addition to `onacopula()`, there's `onacopulaL()` ("L" for **L**ist) which requires a more formal specification of the nesting structure by a `list`.

$F_{ij} = \mathcal{LS}^{-1}[\psi_{ij}(\cdot\,; V_0)]$ where

$$\psi_{ij}(t; x) = \exp\big(-x\psi_i^{-1}(\psi_j(t))\big),\ t \in [0, \infty],\ x \in (0, \infty).$$

The sufficient nesting condition is often easily verified if all generators appearing in the nested structure come from the same parametric family. For each of the implemented Archimedean families of Ali-Mikhail-Haq, Clayton, Frank, Gumbel, and Joe, two generators ψ_i, ψ_j of the same family with corresponding parameters θ_i, θ_j, respectively, fulfill the sufficient nesting condition if $\theta_i \le \theta_j$; equivalently, if $\tau_i \le \tau_j$ for the corresponding Kendall's tau. Verifying the sufficient nesting condition if ψ_i and ψ_j belong to different Archimedean families is usually more complicated, see, Hofert (2010). Such combinations of Archimedean families are therefore currently not implemented in the R package **copula**.

3.2. Sampling

If the sufficient nesting condition is fulfilled for all nodes in the nested Archimedean structure, the following algorithm, based on proposals by McNeil (2008) and Hofert (2011), may be derived for sampling nested Archimedean copulas.

Algorithm 3.1
Let C be a nested Archimedean copula with root copula C_0 generated by ψ_0. Let $\boldsymbol{U}$ be a vector of the same dimension as C.

(1) **sample** $V_0 \sim F_0 = \mathcal{LS}^{-1}[\psi_0]$
(2) **for** all components u of C_0 that are nested Archimedean copulas **do** {
(3) **set** C_1 with generator ψ_1 to the nested Archimedean copula u
(4) **sample** $V_{01} \sim F_{01} = \mathcal{LS}^{-1}[\psi_{01}(\cdot\,; V_0)]$
(5) **set** $C_0 := C_1$, $\psi_0 := \psi_1$, and $V_0 := V_{01}$ and **continue** with (2)
(6) }
(7) **for** all other components u of C_0 **do** {
(8) **sample** $R \sim \mathrm{Exp}(1)$
(9) **set** the component of $\boldsymbol{U}$ corresponding to u to $\psi_0(R/V_0)$
(10) }
(11) **return** $\boldsymbol{U}$

Note that for sampling nested Archimedean copulas when all generators involved belong to the same parametric family, it suffices to know how to sample

$$V_0 \sim F_0 = \mathcal{LS}^{-1}[\psi_0], \quad F_{01} = \mathcal{LS}^{-1}[\psi_{01}(\cdot\,; V_0)]$$

as all distribution functions F_{ij} take the same form as F_{01}, only the parameters may differ. In our R package **copula** (formerly **nacopula**), the supported Archimedean family objects therefore provide the three slots `V0`, `nestConstr`, and `V01`, all functions. `nestConstr`, is a `function(`θ_0`, `θ_1`)` which returns `TRUE` if the sufficient nesting condition is fulfilled and `V0` and `V01` are random number generating functions, generating V from Table 1 and V_{01} from Table 3, respectively. For example, for Ali-Mikhail-Haq,

```
R> copAMH@nestConstr
```

```
function (theta0, theta1)
{
    C.@paraConstr(theta0) && C.@paraConstr(theta1) && theta1 >=
        theta0
}
<bytecode: 0x5635e7d48d28>
<environment: 0x5635e7eef0e8>
```

```
R> copAMH@V01
```

```
function (V0, theta0, theta1)
{
    rnbinom(length(V0), V0, (1 - theta1)/(1 - theta0)) + V0
}
<bytecode: 0x5635e7d45a58>
<environment: 0x5635e7eef0e8>
```

Sampling strategies for F_0 and F_{01} for many known Archimedean generators are presented in Hofert (2008), Hofert (2010), and Hofert (2011). For the families of Ali-Mikhail-Haq, Clayton, Frank, Gumbel, and Joe, Table 3 summarizes stochastic representations for V_0 and V_{01}.

Family	Nesting	$V_{01} \sim F_{01} = \mathcal{LS}^{-1}[\psi_{01}(\cdot\,; V_0)],\ \alpha = \theta_0/\theta_1$
Ali-Mikhail-Haq	$\theta_0 \le \theta_1$	$V_0 + \mathrm{NB}(V_0, (1-\theta_1)/(1-\theta_0))$
Clayton	$\theta_0 \le \theta_1$	$\tilde{\mathrm{S}}(\alpha, 1, (\cos(\alpha\pi/2)V_0)^{1/\alpha}, V_0\mathbf{1}_{\{\alpha=1\}}, \mathbf{1}_{\{\alpha\neq 1\}}; 1)$
Frank	$\theta_0 \le \theta_1$	$\sum_{l=1}^{V_0} V_l,\ V_l \sim p_k = (\theta_1/(1-e^{-\theta_0}))k p_k^{\mathrm{Sibuya}(\alpha)} p_k^{\mathrm{Log}(1-e^{-\theta_1})}$
Gumbel	$\theta_0 \le \theta_1$	$\mathrm{S}(\alpha, 1, (\cos(\alpha\pi/2)V_0)^{1/\alpha}, V_0\mathbf{1}_{\{\alpha=1\}}; 1)$
Joe	$\theta_0 \le \theta_1$	$\sum_{l=1}^{V_0} V_l,\ V_l \sim \mathrm{Sibuya}(\alpha)$

Table 3: Sufficient nesting conditions and stochastic representations for V_{01}.

First note that for nested Archimedean copulas based on generators belonging to the same Archimedean family, all implemented families indeed lead to proper copulas if the generators on a more nested (inner) level have larger parameter values than the ones on a lower (outer) level. This is equivalent to saying that Kendall's tau for a pair of random variables having a bivariate Archimedean copula which resides on a deeper nesting level as margin has to be larger than or equal to the one for a pair of random variables having an Archimedean marginal copula residing on a lower nesting level. Slightly more informally, the inner (or lower) nested components u_j are more correlated than the outer ("higher up") ones.

Some comments on the distributions F_{01} of V_{01} for the different implemented families: For the family of Ali-Mikhail-Haq, V_{01} admits the stochastic representation $V_0 + X$, where X follows a negative binomial distribution with parameters as given in Table 3. The parameterization is $\mathrm{NB}(r, p)$, $r \in (0, \infty)$, $p \in (0, 1)$, with mass function $p_k = \binom{k+r-1}{r-1} p^r (1-p)^k$, $k \in \mathbb{N}_0$, which in R is `dnbinom(`k`, size=`r`, prob=`p`)`.

For Clayton's family, F_{01} can be interpreted as a special case (take $h = 1$, $\alpha = \theta_0/\theta_1$) of the *exponentially tilted stable distribution*

$$\tilde{\mathrm{S}}(\alpha, 1, (\cos(\alpha\pi/2)V_0)^{1/\alpha}, V_0\mathbf{1}_{\{\alpha=1\}}, h\mathbf{1}_{\{\alpha\neq 1\}}; 1) \tag{5}$$

with $\alpha \in (0, 1]$, $h \in [0, \infty)$, $V_0 \in (0, \infty)$, and corresponding Laplace-Stieltjes transform

$$\psi(t) = \exp\big(-V_0((h+t)^\alpha - h^\alpha)\big),\ t \in [0, \infty].$$

Hofert (2011) suggested a fast rejection algorithm for sampling this distribution. Devroye (2009) suggested an algorithm for sampling the exponentially tilted stable distribution

$$\tilde{\mathrm{S}}(\alpha, 1, \cos(\alpha\pi/2)^{1/\alpha}, \mathbf{1}_{\{\alpha=1\}}, \lambda\mathbf{1}_{\{\alpha\neq 1\}}; 1)$$

with $\alpha \in (0,1]$, $\lambda \in [0,\infty)$, and corresponding Laplace-Stieltjes transform

$$\psi(t) = \exp\big(-((\lambda+t)^\alpha - \lambda^\alpha)\big),\ t \in [0,\infty].$$

One can easily check that by setting $\lambda = hV_0^{1/\alpha}$ and generating V_λ with the algorithm of Devroye (2009), the random variable V_{01} from F_{01} as given by (5) can be obtained via $V_{01} = V_0^{1/\alpha} V_\lambda$. Therefore, the distribution as given in (5) may be sampled by either the algorithm of Devroye (2009) or the one of Hofert (2011). The former author reports that the complexity of his algorithm is bounded, the latter author shows that the complexity of his algorithm is $\mathcal{O}(V_0 h^\alpha)$. We implemented both algorithms for sampling (5) in the package **nacopula** (now **copula**) and decide for each drawn V_0 which method is to be applied. As a simple rule, investigated by several parameter combinations, the default chooses the method of Hofert (2011) if $V_0 h^\alpha < 4$ and the one of Devroye (2009) otherwise. As mentioned before, note that for sampling nested Clayton copulas, we have $h = 1$. Further, $\mathsf{E}[V_0] = 1/\theta_0$. Hence, in the mean, the algorithm of Hofert (2011) is more often applied if $\theta_0 > 1/4$, equivalently, if Kendall's tau for the bivariate Archimedean copula generated by ψ_0 is greater than $1/9$.

For the family of Gumbel, $\psi_{01}(t; V_0) = \exp(-V_0 t^\alpha)$, $\alpha = \theta_0/\theta_1$, hence, see Table 1, F_{01} corresponds to the stable distribution as given in Table 3. For Joe's family, V_{01} can be represented as a V_0-fold sum of independent and identically Sibuya distributed random variables V_k, $k \in \{1, \dots, V_0\}$, which can be sampled with the R function `rSibuya(., α)` we provide. For large V_0 an approximation based on a stable distribution may be used, see Hofert (2011). Finally, Frank's family is more complicated. We refer to the source code of the package **copula** and references therein for more details.

3.3. A nine-dimensional nested Clayton copula

In this example, we consider a nine-dimensional partially nested Clayton copula C of the form

$$C(\boldsymbol{u}) = C(u_3, u_6, u_1, C(u_9, u_2, u_7, u_5, C(u_8, u_4; \psi_2); \psi_1); \psi_0)$$

with tree structure depicted in Figure 3. Such a copula can be defined as follows, where we

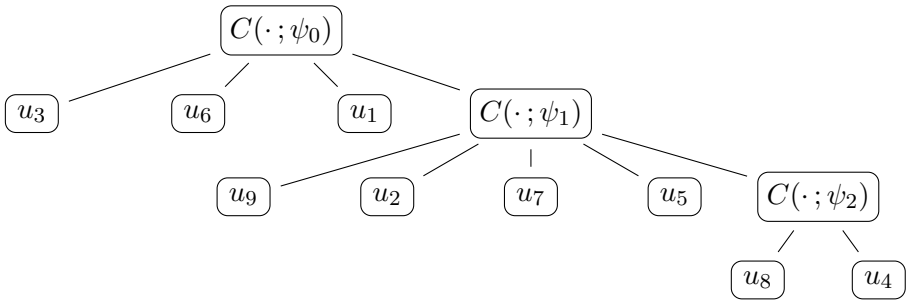

Figure 3: Tree structure of the nine-dimensional partially nested Archimedean copula C.

choose the parameters of ψ_0, ψ_1, and ψ_2 such that the corresponding Kendall's tau are 0.2, 0.5, and 0.8, respectively.

```
R> theta0 <- copClayton@iTau(0.2)
R> theta1 <- copClayton@iTau(0.5)
R> theta2 <- copClayton@iTau(0.8)
R> c(theta0, theta1, theta2)
```

```
[1] 0.5 2.0 8.0
```

```
R> C_9_clayton <- onacopula("Clayton",
+                           C(theta0, c(3,6,1),
+                             C(theta1, c(9,2,7,5),
+                               C(theta2, c(8,4)))))
R> C_9_clayton
```

```
Nested Archimedean copula ("outer_nacopula" of dim. 9), with slot
'comp'   = (3, 6, 1)  and root
'copula' = Archimedean copula ("acopula"), family "Clayton", theta= (0.5)
and 1 child copula
   Nested Archimedean copula ("nacopula"), with slot
   'comp'   = (9, 2, 7, 5)  and root
   'copula' = Archimedean copula ("acopula"), family "Clayton", theta= (2)
   and 1 child copula
      Nested Archimedean copula ("nacopula"), with slot
      'comp'   = (8, 4)  and root
      'copula' = Archimedean copula ("acopula"), family "Clayton", theta= (8)
      and *no* child copulas
```

Figure 4 visualizes 500 sampled random vectors (each in $[0,1]^9$) from this copula (this involves our efficient procedure for exponentially tilted stable distributions) with a scatter-plot matrix. For plotting the data, we re-order the columns according to the same strength of dependence.

```
R> set.seed(1)
R> U9 <- rnacopula(500, C_9_clayton)
R> j <- allComp(C_9_clayton)# copula component "numbers": 1:9 but in "correct order"
R> (vnames <- do.call(expression,
+                     lapply(j, function(i) substitute( U[I], list(I=0+i)))))
```

```
expression(U[3], U[6], U[1], U[9], U[2], U[7], U[5], U[8], U[4])
```

The population version of Kendall's tau for $(U_4, U_5)^\mathsf{T}$ is 0.5. Let us check if the sample version of Kendall's tau is close to this value.

```
R> round(cor(U9[,9],U9[,7], method="kendall"), 3)
```

```
[1] 0.5
```

Evaluating this nine-dimensional nested Clayton copula at $(0.5, \dots, 0.5)^\mathsf{T}$ and near the upper corner $(0.99, \dots, 0.99)^\mathsf{T}$ leads to the following results.

```
R> c(pnacopula(C_9_clayton, rep(.5,9)),
+    pnacopula(C_9_clayton, rep(.99,9)))
```

```
[1] 0.09375995 0.91747302
```

The probability mass in the cube $(0.8, 1]^9$ can be determined as follows.

```
R> prob(C_9_clayton, rep(.8,9), rep(1,9))
```

```
[1] 0.001061674
```

Finally, let us find the different lower and upper tail-dependence coefficients appearing on the different levels of this nested Archimedean copula.

```
R> splom2(U9[, j], varnames= vnames, cex = 0.4, pscales = 0)
```

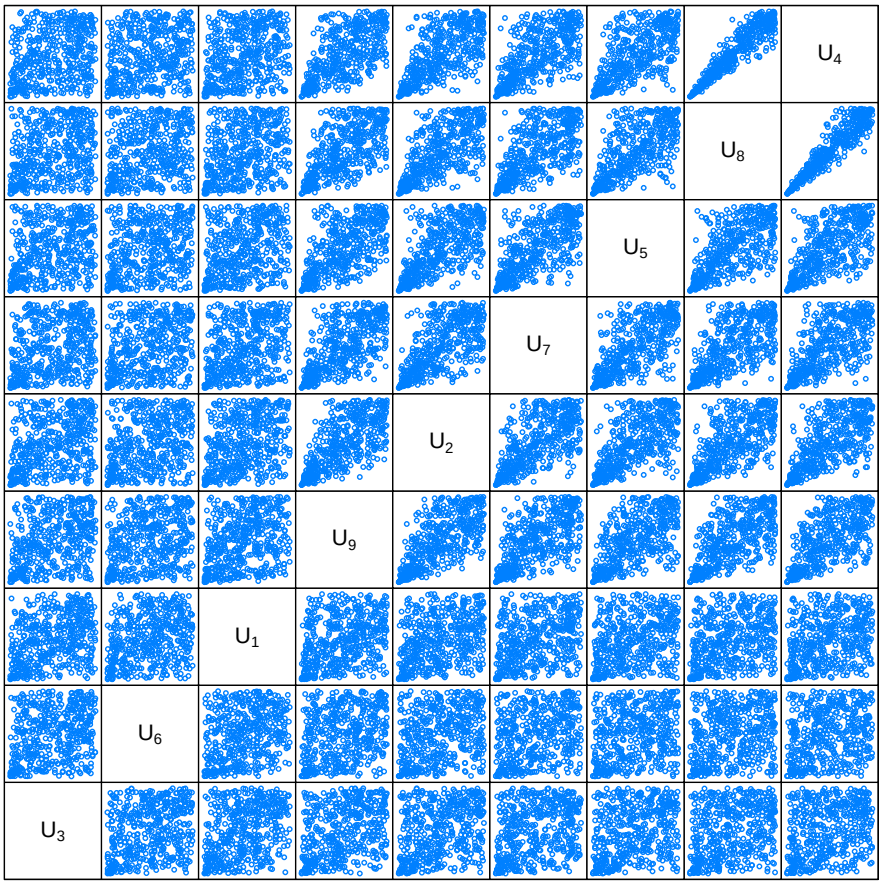

Figure 4: 500 vectors of random variates following a nine-dimensional nested Clayton copula with three different levels of dependence.

```
R> c(copClayton@lambdaL(theta0),
+    copClayton@lambdaU(theta0))

[1] 0.25 0.00

R> c(copClayton@lambdaL(theta1),
+    copClayton@lambdaU(theta1))

[1] 0.7071068 0.0000000

R> c(copClayton@lambdaL(theta2),
+    copClayton@lambdaU(theta2))

[1] 0.917004 0.000000
```

4. Outer power Archimedean copulas

For an Archimedean generator $\psi \in \Psi_\infty$, $\psi(t^{1/\theta})$ is also a valid generator in Ψ_∞ for all $\theta \in [1, \infty)$, see, e.g., Feller (1971, p. 441). The resulting copulas are referred to as *outer power Archimedean copulas*. Note that two parametric Archimedean generators $\psi(t^{1/\theta_0})$, $\theta_0 \in [1, \infty)$, and $\psi(t^{1/\theta_1})$, $\theta_1 \in [1, \infty)$, of this type, constructed with the same "base" generator $\psi(t)$, $t \in [0, \infty)$, fulfill the sufficient nesting condition if $\theta_0 \leq \theta_1$. Therefore, one can build so-called *nested outer power Archimedean copulas*. Hofert (2011) derives some results for these copulas, including instructions for sampling the corresponding random variables V_0 and V_{01}, as well as an explicit formula for Kendall's tau in terms of the Kendall's tau of the copula generated by the base generator ψ. Further, note that if the tail-dependence coefficients exist, they are greater than or equal to the ones corresponding to the copula generated by the base generator. For the Archimedean families implemented in the package **copula**, these can all be computed explicitly.

The goal of this section is to show how one might work with outer power Archimedean copulas with the package **copula**. For now, we use the outer power transformation `opower()` which generates an outer power family based on the provided copula family `copbase` with corresponding parameter `thetabase`.

```
R> str(opower)

function (copbase, thetabase)
```

We believe that it is both more natural and flexible to work with copula families that are generalizations of our current families, each including the power as an extra parameter (such that `theta`, i.e., θ, will become 2-dimensional), rather than using the `opower()` construction below. This section therefore should be considered mainly as an outlook to further features of the package **copula**, where this transformation is not required anymore for working with outer power Archimedean copulas.

Using this transformation, we define a valid outer power Clayton copula with base generator of Clayton's type and with base parameter such that Kendall's tau equals 0.5.

```
R> thetabase <- copClayton@iTau(.5)
R> (opow.Clayton <- opower(copClayton, thetabase))

Archimedean copula ("acopula"), family "opower:Clayton"
 It contains further slots, named
  "psi", "iPsi", "paraInterval", "absdPsi", "paraConstr",
  "absdiPsi", "dDiag", "dacopula", "score", "uscore", "V0",
```

```
  "dV0", "tau", "iTau", "lambdaL", "lambdaLInv", "lambdaU",
  "lambdaUInv", "nestConstr", "V01", "dV01"
```

Based on this copula generator, we would like to define and sample a three-dimensional fully nested outer power Clayton copula with parameters such that Kendall's tau are 2/3 and 0.75.

```
R> theta0 <- opow.Clayton@iTau(2/3) # will be 1.5
R> theta1 <- opow.Clayton@iTau(.75) # will be 2
R> opC3 <- onacopula(opow.Clayton, C(theta0, 1, C(theta1, c(2,3))))
```

Now we sample 500 random variates from this copula and visualize the generated data with a scatter-plot matrix, see Figure 5. In contrast to Clayton copulas, note that outer power Clayton copulas allow for both lower and upper tail dependence.

```
R> U3 <- rnacopula(500, opC3) ; stopifnot(dim(U3) == c(500,3))

R> splom2(U3, cex = 0.4)
```

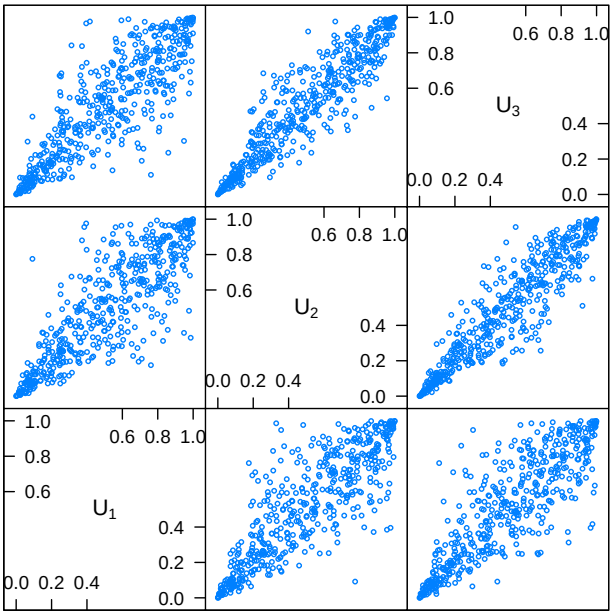

Figure 5: 500 vectors of random variates following a trivariate nested outer power Clayton copula.

Further, we can compare the population and sample versions of Kendall's tau for the generated data. The $(1,2)$- and $(1,3)$-entry of the matrix of pairwise sample versions of Kendall's tau should be close to $2/3$, the $(2,3)$-entry should be close to 0.75.

```
R> round(cor(U3, method="kendall"), 3)
```

```
      [,1]  [,2]  [,3]
[1,] 1.000 0.671 0.678
[2,] 0.671 1.000 0.777
[3,] 0.678 0.777 1.000
```

The different lower and upper tail-dependence coefficients for this copula can be obtained as follows.

```
R> rbind(th0 =
+        c(L = opow.Clayton@lambdaL(theta0),
+          U = opow.Clayton@lambdaU(theta0)),
+        th1 =
+        c(L = opow.Clayton@lambdaL(theta1),
+          U = opow.Clayton@lambdaU(theta1)))
```

```
            L         U
th0 0.7937005 0.4125989
th1 0.8408964 0.5857864
```

5. Session Information

```
R> toLatex(sessionInfo())
```

- R version 4.0.3 Patched (2020-12-08 r79596), x86_64-pc-linux-gnu
- Locale: LC_CTYPE=en_US.UTF-8, LC_NUMERIC=C, LC_TIME=en_US.UTF-8, LC_COLLATE=C, LC_MONETARY=en_US.UTF-8, LC_MESSAGES=C, LC_PAPER=en_US.UTF-8, LC_NAME=C, LC_ADDRESS=C, LC_TELEPHONE=C, LC_MEASUREMENT=en_US.UTF-8, LC_IDENTIFICATION=C
- Running under: Debian GNU/Linux bullseye/sid
- Matrix products: default
- BLAS: /srv/R/R-patched/build.20-12-10/lib/libRblas.so
- LAPACK: /srv/R/R-patched/build.20-12-10/lib/libRlapack.so
- Base packages: base, datasets, grDevices, graphics, grid, methods, parallel, splines, stats, stats4, tools, utils
- Other packages: Rmpfr 0.8-2, VGAM 1.1-4, abind 1.4-5, bbmle 1.0.23.1, copula 1.0-1, gmp 0.6-1, gridExtra 2.3, gsl 2.1-6, lattice 0.20-41, mev 1.13.1, qrng 0.0-8, randtoolbox 1.30.1, rngWELL 0.10-6, rugarch 1.4-4, sfsmisc 1.1-7
- Loaded via a namespace (and not attached): ADGofTest 0.3, DistributionUtils 0.6-0, GeneralizedHyperbolic 0.8-4, KernSmooth 2.23-18, MASS 7.3-53, Matrix 1.2-18, Rcpp 1.0.5, Rsolnp 1.16, Runuran 0.33, SkewHyperbolic 0.4-0, TruncatedNormal 2.2, alabama 2015.3-1, bdsmatrix 1.3-4, boot 1.3-25, compiler 4.0.3, digest 0.6.27, evaluate 0.14, evd 2.3-3, gtable 0.3.0, htmltools 0.5.0, knitr 1.30, ks 1.11.7, magrittr 2.0.1, mclust 5.4.7, mvtnorm 1.1-1, nleqslv 3.3.2, nloptr 1.2.2.2, numDeriv 2016.8-1.1, partitions 1.9-22, pcaPP 1.9-73, polynom 1.4-0, pspline 1.0-18, rlang 0.4.9, rmarkdown 2.5, spd 2.0-1, stabledist 0.7-1, stringi 1.5.3, stringr 1.4.0, truncnorm 1.0-8, xfun 0.19, xts 0.12.1, yaml 2.2.1, zoo 1.8-8

```
R> my.strsplit(  packageDescription("copula")[c("Date", "Revision")]  )
```

```
-- 2020-12-07
```

6. Conclusion

The package **copula** (formerly **nacopula**) allows us to easily construct and work with nested Archimedean copulas. First and foremost, fast sampling algorithms for these copulas are implemented. As a by-product, the package also provides related mathematical and random number generating functions, e.g., efficient sampling algorithms for exponentially tilted stable and Sibuya distributions. Further features include the evaluation of nested Archimedean copulas, as well as computing probabilities of a random vector falling into a given hypercube. Concerning measures of association, Kendall's tau and the tail-dependence coefficients are implemented. Currently supported Archimedean families include the well-known families of Ali-Mikhail-Haq, Clayton, Frank, Gumbel, and Joe. Each can be used in generalized form via "outer power" transformations.

References

Chambers JM, Mallows CL, Stuck BW (1976). "A Method for Simulating Stable Random Variables." *Journal of the American Statistical Association*, **71**(354), 340–344.

Devroye L (2009). "Random Variate Generation for Exponentially and Polynomially Tilted Stable Distributions." *ACM Transactions on Modeling and Computer Simulation*, **19**(4), 18.

Embrechts P, Lindskog F, McNeil AJ (2003). "Modelling Dependence with Copulas and Applications to Risk Management." In S Rachev (ed.), *Handbook of Heavy Tailed Distributions in Finance*, pp. 329–384. North-Holland. ISBN 0-444-50896-1.

Feller W (1971). *An Introduction to Probability Theory and Its Applications*, volume 2. 2nd edition. John Wiley & Sons.

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

Hofert M (2010). *Sampling Nested Archimedean Copulas with Applications to CDO Pricing*. Südwestdeutscher Verlag für Hochschulschriften AG & Co. KG. ISBN 978-3-8381-1656-3. PhD thesis.

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

Joe H (1997). *Multivariate Models and Dependence Concepts*. Chapman & Hall/CRC.

Kemp AW (1981). "Efficient Generation of Logarithmically Distributed Pseudo-Random Variables." *Journal of the Royal Statistical Society C*, **30**(3), 249–253.

Kimberling CH (1974). "A Probabilistic Interpretation of Complete Monotonicity." *Aequationes Mathematicae*, **10**, 152–164.

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/.

Marshall AW, Olkin I (1988). "Families of Multivariate Distributions." *Journal of the American Statistical Association*, **83**, 834–841.

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

Nelsen RB (2007). *An Introduction to Copulas*. 2nd edition. Springer-Verlag, New York.

Nolan JP (2011). *Stable Distributions – Models for Heavy Tailed Data*. Birkhäuser. Chapter 1 online at http://academic2.american.edu/~jpnolan/stable/chap1.pdf.

R Development Core Team (2010). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

Sarkar D (2008). ***lattice**: Multivariate Data Visualization with R*. Springer-Verlag, New York.

Scarsini M (1984). "On Measures of Concordance." *Stochastica*, **8**(3), 201–218.

Sklar A (1959). "Fonctions de Répartition à n Dimensions et Leurs Marges." *Publications de L'Institut de Statistique de L'Université de Paris*, **8**, 229–231.

Wuertz D, *et al.* (2009). ***fCopulae**: Dependence Structures with Copulas*. R package version 2110.78, URL https://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:

Marius Hofert
Department of Mathematics
ETH Zurich
8092 Zurich, Switzerland
E-mail: marius.hofert@math.ethz.ch
URL: http://www.math.ethz.ch/~hofertj/

Martin Mächler
Seminar für Statistik, HG G 16
ETH Zurich
8092 Zurich, Switzerland
E-mail: maechler@stat.math.ethz.ch
URL: http://stat.ethz.ch/people/maechler