Report on a Sampling Algorithm for Nested Archimedian Copulas

Robert Zimmerman

May 9, 2021

Contents

1 Introduction

Copulas are highly useful tools in statistics; they allow us to separate the marginal distributions in a multivariate distribution from the dependence structures that tie them together. The broad range of known copulas lets us capture a plethora of dependence structures, well beyond simple dependence assumptions such as multivariate normality. Copulas have been used prominently in financial risk theory, in applications such as credit scoring (Frey et al. [2001]), share price modelling (Ivanov et al. [2017]), and volatility modelling (Ning et al. [2015]), among countless others. Beyond finance, copulas have found additional applications in fields as diverse as health policy (Candio et al. [2021]), ecology (French et al. [2019]), hydrology (Bezak et al. [2018]), and even sports psychology (Ötting et al. [2021]).

In statistics, the ability to sample from hypothetical or unknown distributions is of fundamental importance, and copulas are certainly no exception. Sampling from univariate distributions is usually straightforward via inverse transform sampling or methods based on rejection sampling. In contrast, sampling from multivariate distributions is not a trivial task in general. Any efficient sampling algorithm must depend upon specific properties of the multivariate distribution being sampled from. While there exists an algorithm – the *conditional sampling* method – to sample from any copula in principle, it is infeasible to implement in practice, as it requires the computation of all of the mixed partial derivatives of the copula (Cherubini et al. [2004]). The paper which is the focus of our discussion, *A stochastic representation and sampling algorithm for nested Archimedean*

copulas (Hofert [2012]), presents an algorithm that addresses this problem for a certain class of copulas used to model hierarchical dependence structures in multidimensional settings.

This report is organized as follows: Section 2 gives a brief overview of copulas and briefly develops nested Archimedean copulas in the context of sampling, in order to motivate Hofert's paper. In Section 3, we describe the theory that underlies the innovations in Hofert's paper. In Section 4, we implement Hofert's simulation algorithm on a range of examples. In Section 5, we summarize our work, discuss limitations, and suggest future research directions. We provide `R` code for our simulations in Appendix A.

2 Archimedean Copulas

A d-dimensional *copula* $C : [0,1]^d \to [0,1]$ is simply a multivariate distribution function with standard uniform margins (that is, $(U_1, \dots, U_d) \sim C$ implies $U_i \sim \text{Uniform}[0,1]$ for each $1 \leq i \leq d$). The power of copulas is a consequence of *Sklar's theorem*, upon which much of the theory and applications of copulas rests. The first half of Sklar's theorem states that every d-dimensional distribution function with margins $F_1, \dots, F_d$ can be written as $C(F_1(x_1), \dots, F_d(x_d))$ for some d-dimensional copula $C(\cdot, \dots, \cdot)$ (Nelsen [2007]). Thus, copulas allow one to "couple" together marginal distributions into a multivariate distribution, thereby separating completely the marginal distributions from the dependence structure that connects them.

Although a vast array of copulas are known, many of them lack convenient closed-form expressions, which can make statistical inference challenging. One famous example of such an *implicit copula* (Marcantoni [2014]) is the Gauss copula, which is defined in terms of a d-dimensional multivariate Normal distribution function with standard Normal margins, as in the statement of Sklar's theorem. In contrast, *explicit copulas* are usually easier to work with. Perhaps the most well-known example of these is the class of *Archimedean copulas* – so named because they satisfy a certain property akin to the Archimedean property of $\mathbb{R}$ (Nelsen [2007]). They are particularly useful because of their functional form: each d-dimensional Archimedean copula can be written as

$$C(u_1, \dots, u_d) = \psi_\theta(\psi_\theta^{-1}(u_1) + \cdots + \psi_\theta^{-1}(u_d))$$

for $(u_1, \dots, u_d) \in [0,1]^d$, where the function $\psi_\theta : [0,1] \times \Theta \to [0, \infty]$ must satisfy a number of properties on which we elaborate in Section 3. Such a function is called an *(Archimedean) generator*. The ability to represent an Archimedean copula simply in terms of its generator allows us to employ a broad range of theory that is not otherwise applicable to more general copulas.

The subscript θ on the generator ψ_θ refers to a univariate parameter of the generator that measures the "strength" of the dependence between the components of the copula, so that $\{\psi_\theta : \theta \in \Theta\}$ defines a parametric family of copulas. Among the most popular and well-studied of these are the *Clayton*, *Gumbel*, *Frank*, *Joe*, and *Ali-Mikhail-Haq (AMH)* families, which are sometimes referred to as the *classical families*. Concordance measures such as Spearman's rho and Kendall's tau for any particular Archimedean copula are often expressible as functions of θ (in the latter case, $\kappa = 1 - 4\int_0^\infty t \cdot (\psi'(t))^2 \, \mathrm{d}t$ holds for any sufficiently regular generator ψ), and typically limiting values of θ result in the independence copula, the comonotonic copula or the countermonotonic copula (Genest and MacKay [1986]). Because we will be using other subscripts to distinguish different generators, we will keep the dependence on θ implicit and simply write ψ to refer to a generic generator in the sequel.

For all of their benefits, Archimedean copulas have some drawbacks as well. For example, it is evident from the generator form of an Archimedean copula that such copulas are *exchangeable* (i.e., symmetric in their arguments). This symmetry is not necessarily desirable, because it implies that all pairs of marginals have the same joint dependence structure (Di Bernardino and Rullière [2016]); this can make estimation of the generator ψ particularly difficult, as the symmetry constraint becomes harder and harder to satisfy as the as the dimension d grows. Harry Joe observed in Joe [1997] that one can carefully construct a multivariate copula by nesting generators so that each bivariate margin is distributed according to a (possibly unique) bivariate Archimedean copula, resulting in different degrees of dependence between pairs of marginals. For example, to construct a 3-dimensional copula such that the second and third components depend on each other through a generator ψ_1 and whose first component depends on both the second and third through another generator ψ_0, we might take

$$C(u_1, u_2, u_3; \psi_0, \psi_1) = \psi_0\Big(\psi_0^{-1}\Big(u_1\Big) + \psi_0^{-1}\Big(\psi_1(\psi_1^{-1}(u_2) + \psi_1^{-1}(u_3))\Big)\Big).$$

The idea of nesting Archimedean copulas was soon formalized into a new class of copulas appropriately labelled *nested Archimedean copulas*, which quickly found use in a range of applications, from tourism management (Zhang et al. [2012]) to hydrology (Serinaldi and Grimaldi [2007]). In a groundbreaking work, Hofert and Scherer [2011] used these copulas to price collateralized debt obligations in the wake of the Great Recession.

3 Sampling from Archimedean and Nested Archimedean Copulas

We noted above that sampling from general multivariate distributions is a difficult problem. For the case of Archimedean copulas, however, Marshall and Olkin [1988] discovered a sampling algorithm which is substantially simpler. The *Marshall-Olkin algorithm* exploits the fact that any generator ψ that *defines* an Archimedean copula is the Laplace-Stieltjes transform $\mathcal{LS}[F]$ of some univariate distribution function F, thereby reducing the computational challenges to the (relatively) straightforward task of sampling from F.

As mentioned in Section 2, the function $\psi : [0,1] \to [0,\infty]$ must satisfy several properties in order for it to be a generator (i.e., in order for the function $(u_1, \ldots, u_d) \mapsto \psi(\psi^{-1}(u_1) + \cdots + \psi^{-1}(u_d))$ to actually be a copula). Specifically, it is necessary that ψ be continuous and decreasing with $\psi(0) = 1$ and $\lim_{t\to\infty} \psi(t) = 0$, and moreover that ψ be strictly decreasing on $[0, \inf\{t : \psi(t) = 0\}]$ (Nelsen [2007]). This is not sufficient, however. A result of Kimberling [1974] shows that a generator defines an Archimedean copula in all dimensions if and only if $(-1)^k \frac{\mathrm{d}^k}{\mathrm{d}t^k}\psi(t) \geq 0$ for all $t > 0$ and $k \in \mathbb{N}_0$ – a property called *complete monotonicity*. Remarkably, the Bernstein-Widder theorem (Bernstein et al. [1929], Widder [1941]) states that *any* completely monotone function $\psi(t)$ can be written as $\mathbb{E}\left[e^{-tX}\right] = \int_0^\infty e^{-tx}\,\mathrm{d}F(x)$, where X is a random variable distributed according to some distribution F supported on $[0,\infty)$. In other words, ψ is the *Laplace-Stieltjes transform* of F, and F is therefore the *inverse Laplace transform* of ψ. We write these statements as $\psi = \mathcal{LS}[F]$ and $F = \mathcal{LS}^{-1}[\psi]$, respectively.

It is this deep connection between Archimedean generators and Laplace-Stieltjes transforms that leads to Marshall and Olkin's practical algorithm for sampling from a d-dimensional Archimedean copula generated by ψ. For if $\psi(t) = \int_0^\infty e^{-tx}\,\mathrm{d}F(x)$ where $F = \mathcal{LS}^{-1}[\psi]$, then it immediately

follows that

$$\begin{aligned} C(\mathbf{u}) &= \psi\left(\psi^{-1}(u_1) + \cdots + \psi^{-1}(u_d)\right) \\ &= \int_0^\infty e^{-x\left(\psi^{-1}(u_1)+\cdots+\psi^{-1}(u_d)\right)}\,\mathrm{d}F(x) \\ &= \int_0^\infty \prod_{j=1}^d e^{-x\psi^{-1}(u_j)}\,\mathrm{d}F(x) \end{aligned} \tag{1}$$

Equation 1 shows that $C(\mathbf{u})$ is a *scale mixture* distribution – namely, the distribution $G(\mathbf{u} \mid x) = \prod_{j=1}^d e^{-x\psi^{-1}(u_j)}$ on $[0,1]^d$ compounded by F. It is well known how to sample from such a scale mixture, assuming sampling from F is feasible: one first samples V from the mixing distribution $F(\cdot)$, and then samples from $G(\cdot \mid V)$. Clearly, conditional on V, the latter step can be accomplished by independently sampling $U_j \mid V$ from the univariate distribution $e^{-V\cdot\psi^{-1}(\cdot)}$ each for $1 \leq j \leq d$. This itself is trivial: since $\left(u \mapsto e^{-V\cdot\psi^{-1}(u)}\right)^{-1} = \psi(-\log(u)/V)$, it follows from the probability integral transform that $\psi(-\log(U)/V) \sim e^{-V\cdot\psi^{-1}(\cdot)}$, where $U \sim \mathrm{Unif}[0,1]$. Moreover, from the elementary fact that $-\log(U) \sim \mathrm{Exp}(1)$, we thus obtain the following stochastic representation of $\mathbf{U} \sim C$:

$$\mathbf{U} \overset{d}{=} \left(\psi\left(\frac{R_1}{V}\right), \ldots, \psi\left(\frac{R_d}{V}\right)\right), \tag{2}$$

where $R_1, \ldots, R_d \overset{iid}{\sim} \mathrm{Exp}(1)$. The Marshall-Olkin algorithm is exactly the preceding sampling scheme, shown in Algorithm 1.

Algorithm 1: Marshall-Olkin Algorithm

Input: An Archimedean generator ψ as in Equation 1
Output: A random variate $\mathbf{U} \sim C$
Sample $V \sim F = \mathcal{LS}^{-1}[\psi]$
for $j = 1$ **to** d **do**
  Sample $R_j \sim \mathrm{Exp}(1)$
  Set $U_j = \psi(R_j/V)$
end
return $\mathbf{U} = (U_1, \ldots, U_d)^T$

The only potential difficulty in implementing the Marshall-Olkin algorithm is that sampling from $F = \mathcal{LS}^{-1}[\psi]$ may be challenging (even working out the Laplace-Stieltjes transform of an arbitrary completely monotone function may be a challenge). No fully general method exists for doing this, and so one must work out a way to sample from F on a case-by-case basis. Fortunately, the Laplace-Stieltjes transforms of generators for the five classical families of Archimedean generators are known, as are the transforms for a slew of other Archimedean generators (Joe [2014]). Some of the transforms – such as variants of the Geometric and the Gamma distributions – are familiar distributions, while others – such as the Sibuya and the tilted positive stable distributions – are more esoteric. Importantly though, efficient sampling methods are known for all of them (in fact, Hofert himself provides a novel sampling scheme for the Sibuya distribution later in Hofert [2012]).

In McNeil [2008], McNeil sought to extend the Marshall-Olkin algorithm to d-dimensional nested Archimedean copulas with nested completely monotone generators $\psi_0, \psi_1, \ldots, \psi_{d-2}$, focusing (for

convenience) on the *fully nested* Archimedean copulas. With a slight abuse of notation, these take the recursive form

$$C(u_1, u_2, \ldots, u_d; \psi_0, \ldots, \psi_{d-2}) = \psi_0\Big(\psi_0^{-1}\big(u_1\big) + \psi_0^{-1}\Big(C(u_2, \ldots, u_d; \psi_1, \ldots, \psi_{d-2})\Big)\Big), \quad u_i \in [0,1], \tag{3}$$

where the outermost generator ψ_0 generates the *root* copula $C(\cdot; \psi_0)$ and the generators without further nesting generate *leaf* copulas. McNeil first showed that Equation 3 defines a valid copula if $(\psi_k^{-1} \circ \psi_{k+1})'$ is completely monotone for each $0 \leq k \leq d-3$ (this *sufficient nesting condition* is not necessary, however). Modifying Equation 1 to accommodate Equation 3, we observe that

$$C(u_1, u_2, \ldots, u_d; \psi_0, \ldots, \psi_{d-2}) = \int_0^\infty e^{-x\psi_1^{-1}(u_1)} e^{-x\psi_0^{-1}(C(u_2,\ldots,u_d;\psi_1,\ldots,\psi_{d-2}))}\, \mathrm{d}F_0(x). \tag{4}$$

McNeil's key insight was that the second term in the integrand is itself a $(d-1)$-dimensional nested Archimedean copula:

$$e^{-x\psi_0^{-1}(C(u_2,\ldots,u_d;\psi_1,\ldots,\psi_{d-2}))} = C(e^{-x\psi_0^{-1}(u_2)}, \ldots, e^{-x\psi_0^{-1}(u_d)}; \psi_{0,1}(\cdot; x), \ldots, \psi_{0,d-2}(\cdot; x)),$$

where $\psi_{0,j}(t, x) := e^{-x\psi_0^{-1}(\psi_j(t))}$ is another generator. Thus, we have another scale mixture – this time, it is the product of $e^{-x\psi_1^{-1}(u_1)}$ and $C(e^{-x\psi_0^{-1}(u_2)}, \ldots, e^{-x\psi_0^{-1}(u_d)}; \psi_{0,1}(\cdot; x), \ldots, \psi_{0,d-2}(\cdot; x))$ that is now compounded by $F_0 = \mathcal{LS}^{-1}[\psi_0]$. Applying the same principle as in the Marshall-Olkin algorithm, to sample from $C(u_1, u_2, \ldots, u_d; \psi_0, \ldots, \psi_{d-2})$, we may first sample $V_0 \sim F_0$, and then independently sample from $e^{-V_0 \cdot \psi_1^{-1}(u_1)}$ and $C(e^{-V_0\psi_0^{-1}(u_2)}, \ldots, e^{-V_0\psi_0^{-1}(u_d)}; \psi_{0,1}(\cdot; V_0), \ldots, \psi_{0,d-2}(\cdot; V_0))$. Evidently, the last step requires sampling from *another* nested Archimedean copula. Applying the same principle recursively to the child copulas that follow, we will eventually arrive at a leaf copula which can be sampled directly using the Marshall-Olkin algorithm. This observation immediately leads to McNeil's recursive algorithm, shown in 2.

Algorithm 2: McNeil's Algorithm

Input: Archimedean generators $\psi_0, \psi_1, \ldots, \psi_{d-2}$ as in Equation 3
Output: A random variate $\mathbf{U} \sim C$
Sample $V_0 \sim F_0 = \mathcal{LS}^{-1}[\psi_0]$
Sample $(X_2, \ldots, X_d)^T \sim C\left(u_2, \ldots, u_d; \psi_{0,1}(\cdot; V_0), \ldots, \psi_{0,d-2}(\cdot; V_0)\right)$ using this algorithm
Sample $X_1 \sim \mathrm{Unif}[0,1]$
for $j = 1$ **to** d **do**
  Set $U_j = \psi_0(-\log(X_j)/V_0)$
end
return $\mathbf{U} = (U_1, \ldots, U_d)^T$

While the *McNeil algorithm* was theoretically groundbreaking, it involved tedious computational demands that made it impractical to use in large dimensions. The paper of our focus, *A stochastic representation and sampling algorithm for nested Archimedean copulas* (Hofert [2012]), presents an algorithm that improves significantly upon the McNeil algorithm to allow for efficient sampling from nested Archimedean copulas. As with the Marshall-Olkin algorithm, Hofert identifies the primary difficulty in the McNeil algorithm as sampling from the Laplace-Stieltjes transform of the parent generator of each nested copula; the specific functional form of each generator depends on that of

its parent, so that when the number of dimensions d is large, the number of function compositions required to evaluate each new generator becomes unfeasible.

To recursively apply McNeil's algorithm requires sampling a mixing variable V_{l_j} for every leaf copula at level l_j. For $l_j = 1$, this is simply a matter of sampling from $V_0 \sim F_0$. If $l_j = 2$ and k refers to the leaf copula in question, then this requires sampling $V_{0,k} \sim \mathcal{LS}^{-1}[\psi_{0,k}(\cdot; V_0)]$. For $l_j \geq 3$, the procedure continues recursively, where each scale random variable V_{l_j} must be sampled from an inverse Laplace-Stieltjes transformation of its predecessor – which therefore depends on all of the generators from ψ_0 down to ψ_{l_j} in the hierarchical structure. The key observation in Hofert [2012] (which was also noted earlier in McNeil [2008]) is that

$$e^{-x\psi_{k,k+1}^{-1}(\psi_{k,j}(t;x');x')} = \psi_{k+1,j}(t;x), \quad x, x' > 0, \tag{5}$$

in which the subscripts (k, j) are such that k refers to the generator of the parent copula, while j refers to to the generator of the child copula. Thus, sampling V_{l_j} in fact only requires knowledge of the copula at level l_j and that of the corresponding parent copula; evaluating each of the generators down the tree is not actually required.

This insight leads to a stochastic representation in the same spirit as 2, which requires a description of the notation used by Hofert. For a general d-dimensional nested Archimedean copula C with root copula ψ_0 which includes m leaf copulas, the set of component indices $\{1, \ldots, d\}$ can be partitioned into disjoint subsets $J_1, \ldots, J_m$ such that J_i contains the indices of the components of $\mathbf{U}$ which share the i'th leaf copula. If l_j denotes the level of the leaf copula in which the j'th component resides, then $l_j = l_{j'}$ whenever j and j' are in the same subset J_i, and the same scaling random variable V is used to sample the jth and j'th components from $\mathcal{LS}^{-1}[e^{-V\psi_i^{-1}(\psi_l(t))}]$, where i refers to the generator of the parent copula of components j and j', and l refers to the generator of the child copula in which components j and j' reside. Thus, for $\mathbf{U} \sim C$, we have that

$$\mathbf{U} = \left(\psi_{l_1}\left(\frac{R_1}{V_{l_1}}\right), \ldots, \psi_{l_d}\left(\frac{R_d}{V_{l_d}}\right)\right),$$

where $R_1, \ldots, R_d \overset{iid}{\sim} \text{Exp}(1)$. Hofert's algorithm is thus summarized in Algorithm 3.

The remainder of Hofert [2012] discusses how to sample specific Laplace-Stieltjes transforms for members of the five Archimedean families mentioned in 2 (including their implementations in the `copula` library in `R`), and presents benchmark tests for sampling from several examples of nested Archimedean copulas with increasingly complex hierarchies. Hofert demonstrates that his algorithm runs quickly and efficiently, even in relatively high dimensions.

Algorithm 3: Hofert's Algorithm

Input: Archimedean generators $\psi_0, \psi_1, \ldots, \psi_{d-2}$ and a hierarchy of parent and child copulas
Output: A random variate $\mathbf{U} \sim C$
Sample $V_0 \sim F_0 = \mathcal{LS}^{-1}[\psi_0]$
Set C_0 to be the root copula generated by ψ_0
for *all components u of C_0 which are nested Archimedean copulas* **do**
    Set C_1 with generator ψ_1 to be the nested Archimedean copula u
    Sample $V_{01} \sim F_{01} = \mathcal{LS}^{-1}[\psi_{01}(\cdot; V_0)]$
    Set $C_0 \leftarrow C_1$, $\psi_0 \leftarrow \psi_1$, and $V_0 \leftarrow V_{01}$ and continue
end
for *all other components u of C_0* **do**
    Sample $R \sim \text{Exp}(1)$
    Set the component of $\mathbf{U}$ corresponding to u to $\psi_0(R/V_0)$
end
return $\mathbf{U} = (U_1, \ldots, U_d)^T$

4 Simulations

In this section, we reproduce one of the simulation studies in Hofert [2012], and provide several additional ones of our own. While applying Hofert's algorithm is easy if we know how to sample from the Laplace-Stieltjes transforms of the relevant generators, it can be performed only if the generators and their hierarchy do define a legitimate copula; this is usually checked by verifying the sufficient nesting condition for the pair of generators ψ_k and ψ_{k+1} that respectively characterize each parent and child copula at the k'th level of the hierarchy. This is trivial when both ψ_k and ψ_{k+1} are in exactly one of the five Archimedean families mentioned in 2, for Joe [1997] shows that the sufficient nesting condition is satisfied provided that the parameter θ_k of ψ_k is bounded above by the parameter θ_{k+1} of the ψ_{k+1}. When the two generators live in different families, however, the sufficient nesting condition must be manually verified, either by checking the complete monotonicity of each pair $(\psi_k^{-1} \circ \psi_{k+1})'$ or by exploiting known analytical results for the relevant families on a case-by-case basis, as was done by Hofert [2010] for several examples of mixed pairs.

We have implemented Hofert's algorithm to sample from five different nested Archimedean copulas. Our first three are 4-dimensional fully nested copulas of the form in Equation 3, where all nested generators are from the same family. Figure 1, Figure 2, and Figure 3 show illustrate the AMH family, the Clayton family, and the Gumbel family (respectively). We have chosen the parameter values in order to create interesting changes in the dependence structures at deeper hierarchical levels (ensuring, of course, that the sufficient nesting condition is satisfied).

In Figure 4, we have reproduced Hofert's example of a nested Archimedean copula with generators in different families. This example, which Hofert also presented in Hofert [2010], is a 4-dimensional fully tested copula with generators belonging to the AMH family, the Clayton family, and the family $\{\psi_\theta(t) = (\log(t+e))^{1/\theta} : \theta \in (0, \infty)\}$ (a family referred to as "20", due to its placement in Nelsen [2007]'s large table of one-parameter Archimedean copulas).

Finally, in Figure 5, we plot an 8-dimensional nested Archimedean copula with all generators from the Frank family, where each parameter θ_j is chosen so that $\tau(\theta_j) = 0.1(j+1)$, where $\tau(\cdot)$ is Kendall's tau written as a function of the parameter θ (for this family, a linearly spaced set of

values for Kendall's tau corresponds to a set of exponentially spaced values for θ). For bivariate pairs involving lower-order components, we observe joint distributions resembling the independence copula; on the other hand, as we move up and to the right of the scatterplot matrix, we, we observe stronger and stronger dependence pairs involving higher-order components, as the nested copulas approach the comonotonicity copula (Nelsen [2007]).

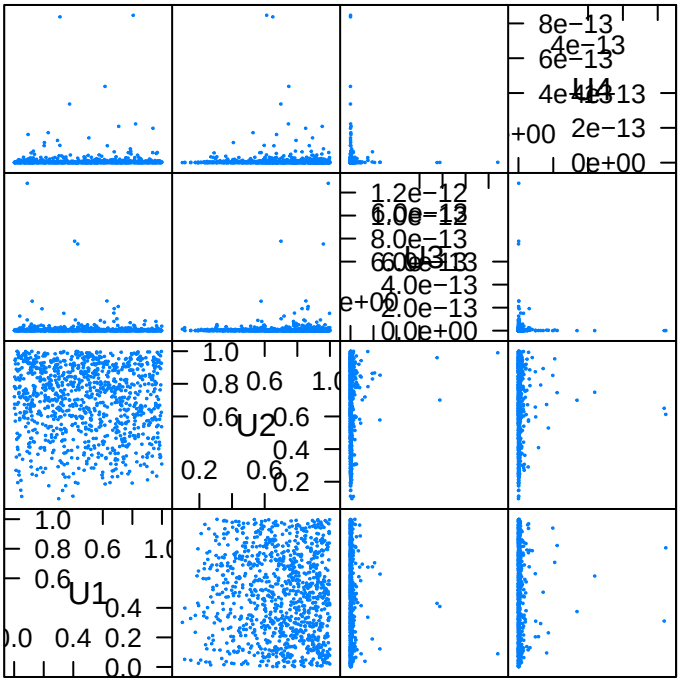

Figure 1: 1000 samples from a four-dimensional fully nested Archimedean copula with generators belonging to the Ali-Mikhail-Haq families, with parameters $\theta_0 \approx 0$, $\theta_1 = 0.5$, and $\theta_2 \approx 1$

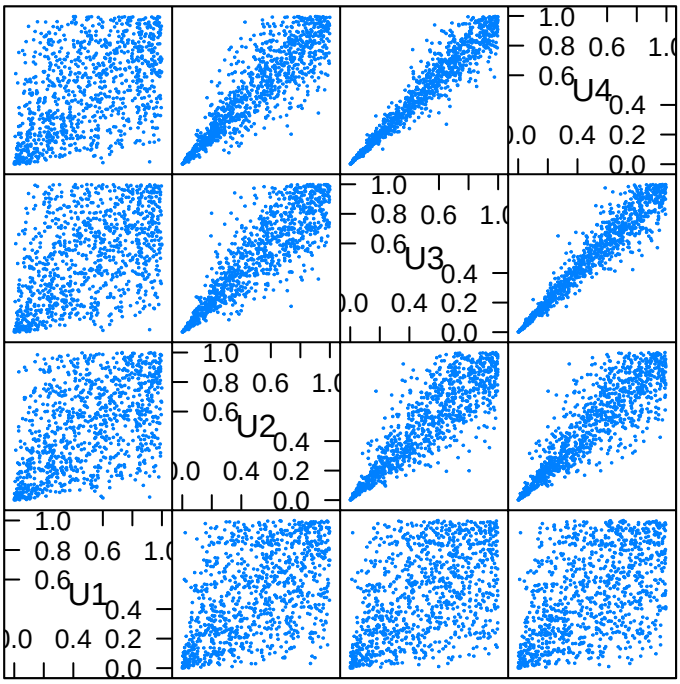

Figure 2: 1000 samples from a four-dimensional fully nested Archimedean copula with generators belonging to the Clayton family, with parameters $\theta_0 = 1$, $\theta_1 = 5$, and $\theta_2 = 10$

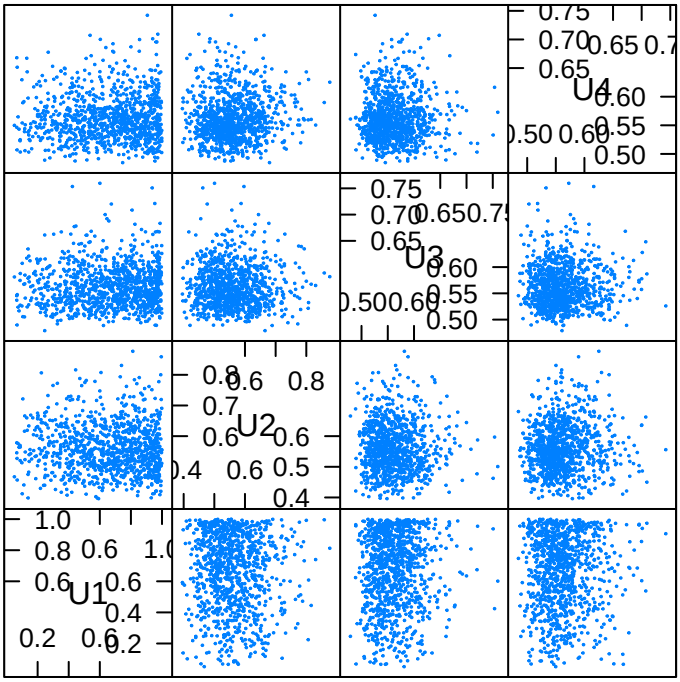

Figure 3: 1000 samples from a four-dimensional fully nested Archimedean copula with generators belonging to the Gumbel family, with parameters $\theta_0 = 1$, $\theta_1 = 5$, and $\theta_2 = 10$

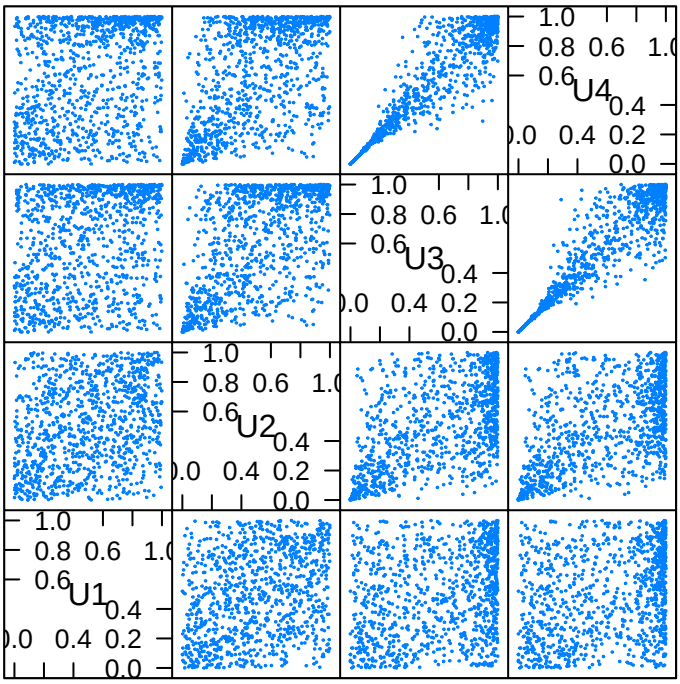

Figure 4: Reproduction of Figure 5 (left) in Hofert [2012]: 1000 samples from a four-dimensional fully nested Archimedean copula with generators belonging to the Ali-Mikhail-Haq, Clayton, and $\{\psi_\theta(t) = (\log(t+e))^{1/\theta} : \theta \in (0,\infty)\}$ families

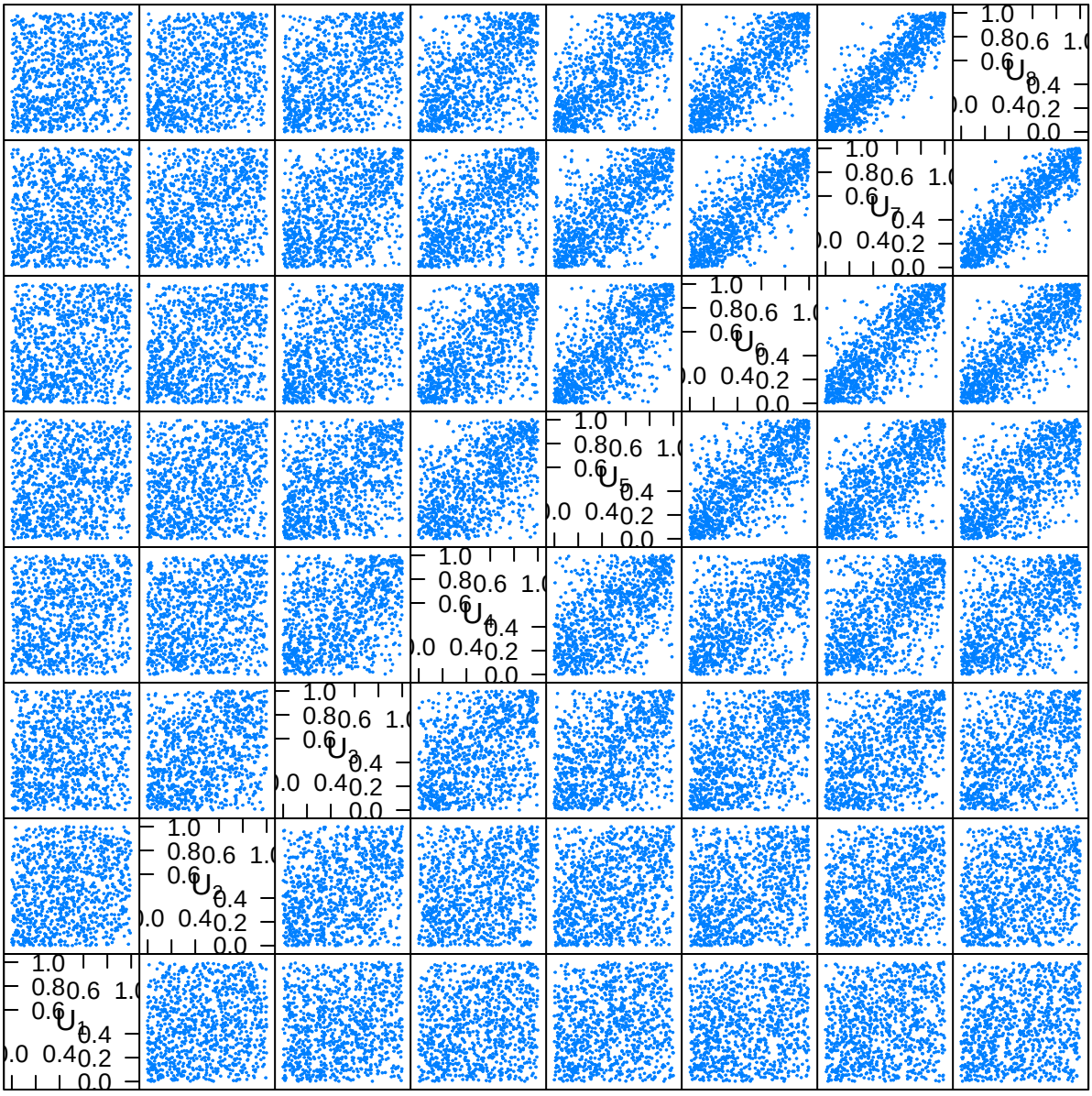

Figure 5: 1000 samples from an eight-dimensional fully nested Archimedean copula with generators belonging to the Frank family, with parameters $\theta_0, \ldots, \theta_6$ chosen such that $\tau_j = 0.1(j+1)$ for $j = 0, \ldots, 6$

5 Discussion

In this report, we have summarized the ideas underlying Marius Hofert's paper *A stochastic representation and sampling algorithm for nested Archimedean copulas*. We began with a brief exposition of copulas themselves – focusing on Archimedean copulas and their nested variants – and then discussed the problem of sampling from such copulas. To summarize, Marshall and Olkin [1988] showed how to sample from Archimedean copulas by providing a stochastic representation of the random variable to be sampled in terms of other quantities that are (relatively) easy to calculate and sample from. While McNeil [2008] provided the first formal algorithm for sampling from nested Archimedean copulas, Hofert substantially increased the practicality of this algorithm by deriving a stochastic representation of the random variable to be sampled in a spirit similar to Marshall and Olkin, thereby recasting McNeil's algorithm in a way that avoided most of the challenging computations.

Hofert's paper was published in 2012. In the near-decade since then, a number of papers have been published which relate to sampling from Archimedean copulas, by Hofert as well as others. Grothe and Hofert [2015] extended Archimedean Lévy copulas to nested versions, and Hofert et al. [2018] did the same for Archimax copulas; both papers allude to sampling algorithms for their respective nested copulas which follow the same approach as Hofert [2012]. Härdle et al. [2013] adapted the algorithm to handle time series of nested Archimedean copulas. On the inferential side, Hofert and Pham [2013] used similar strategies to develop explicit forms for the densities of nested Archimedean copulas in arbitrary dimensions, and Górecki et al. [2014] used the stochastic representation as an ingredient in deriving the consistency of an estimator for nested Archimedean copulas (the parameters and the structure). More recently, Ng et al. [2021] used neural networks to represent the latent variables in the stochastic representation (denoted by the V_{l_j}'s in Section 3), and provided sampling algorithms modified to this new structure.

While Hofert's algorithm alleviates much of the computational burdens of McNeil's algorithm for sampling from an arbitrary nested Archimedean copula, it is still limited by the need to derive and then sample from inverse Laplace-Stieltjes transforms of arbitrary generators – both of the generators $\psi_0, \dots, \psi_d$ in the specification of the copula, and (much more challengingly) of the "connecting" generators $\psi_{il}(t; V) = e^{-V\psi_i^{-1}(\psi_l(t))}$. Those who opt to use generators outside of the classical families (for which most of the known results apply to) must resort to their own facilities. While Hofert [2010] proposed a number of numerical schemes for approximating inverse Laplace-Stieltjes transforms, these methods are both inexact and computationally expensive.

Nested Archimedean copulas themselves are also limited in that not every choice of nesting structure and set of generators actually defines a valid copula. Unlike the case for vanilla Archimedean copulas, there is no universal condition on the generators that characterizes the validity of the resulting copula; in some scenarios, one can imagine the sufficient nesting condition being too restrictive to be relied upon. The sufficient nesting condition can be difficult to check for pairs of generators in different parametric families (Hofert's `copula` library only allows for the construction of nested Archimedean copulas with generators in the same classical family, for which the sufficient nesting condition is trivial to verify). Thus, further avenues of work could involve designing exact algorithms for calculating inverse Laplace-Stieltjes transforms and for verifying the sufficient nesting condition for mixed families of generators.

A R Code

```
knitr::opts_chunk$set(echo = FALSE, cache = TRUE, warning = FALSE, message = FALSE,
    fig.pos = "!h")
library(copula)
#### Sampling from a 4-dimensional nested Archimedean copula with AMH
#### generators

N <- 1000  # number of samples


# Set our generator parameters
theta0 <- .Machine$double.eps
theta1 <- 0.5
theta2 <- 1 - .Machine$double.eps


# Define our Archimedean generators
psi.AMH <- function(t, theta) {
    (1 - theta)/(exp(t) - theta)
}


# Sample V0 and V01
V0 <- 1 + rgeom(n = N, p = 1 - theta0)
V01 <- V0 + rnbinom(n = 1, size = V0, prob = (1 - theta1)/(1 - theta0))
V12 <- V01 + rnbinom(n = 1, size = V01, prob = (1 - theta1)/(1 - theta0))

# Sample (U1, U2, U3, U4)
R1 <- rexp(n = N)
R2 <- rexp(n = N)
R3 <- rexp(n = N)
R4 <- rexp(n = N)
U1 <- psi.AMH(R1/V0, theta0)
U2 <- psi.AMH(R2/V01, theta1)
U3 <- psi.AMH(R3/V12, theta2)
U4 <- psi.AMH(R4/V12, theta2)
U <- cbind(U1, U2, U3, U4)
AMHplot4D <- splom2(U, cex = 0.1)

AMHplot4D

#### Sampling from a 4-dimensional nested Archimedean copula with Clayton
#### generators

N <- 1000  # number of samples
```

```r
# Set our generator parameters
theta0 <- 1
theta1 <- 5
theta2 <- 10
alpha01 <- theta0/theta1
alpha12 <- theta1/theta2

# Define our Archimedean generator
psi.C <- function(t, theta) {
    (1 + t)^(-1/theta)
}


# Sample V0 and V01
V0 <- rgamma(n = N, shape = 1/theta0, rate = 1)
V01 <- retstable(alpha = alpha01, V0 = V0, h = 1)
V12 <- retstable(alpha = alpha12, V0 = V01, h = 1)

# Sample (U1, U2, U3, U4)
R1 <- rexp(n = N)
R2 <- rexp(n = N)
R3 <- rexp(n = N)
R4 <- rexp(n = N)
U1 <- psi.C(R1/V0, theta0)
U2 <- psi.C(R2/V01, theta1)
U3 <- psi.C(R3/V12, theta2)
U4 <- psi.C(R4/V12, theta2)
U <- cbind(U1, U2, U3, U4)
Claytonplot4D <- splom2(U, cex = 0.1)

Claytonplot4D

#### Sampling from a 4-dimensional nested Archimedean copula with Gumbel
#### generators

N <- 1000  # number of samples

# Set our generator parameters
theta0 <- 1
theta1 <- 5
theta2 <- 10
alpha01 <- theta0/theta1
alpha12 <- theta1/theta2

# Define our Archimedean generators
psi.G <- function(t, theta) {
    exp(-t^(1/theta))
```

```
}


# Sample V0 and V01
V0 <- rstable1(n = N, alpha = 1/theta0, beta = 1, gamma = cos(pi/(2 * theta0))^theta0,
    delta = 1 * (theta0 == 1))
V01 <- rstable1(n = 1, alpha = alpha01, beta = 1, gamma = (cos(alpha01 *
    pi/2) * V0)^(1/alpha01), delta = V0 * (alpha01 == 1))
V12 <- rstable1(n = 1, alpha = alpha12, beta = 1, gamma = (cos(alpha12 *
    pi/2) * V01)^(1/alpha12), delta = V01 * (alpha12 == 1))

# Sample (U1, U2, U3, U4)
R1 <- rexp(n = N)
R2 <- rexp(n = N)
R3 <- rexp(n = N)
R4 <- rexp(n = N)
U1 <- psi.G(R1/V0, theta0)
U2 <- psi.G(R2/V01, theta1)
U3 <- psi.G(R3/V12, theta2)
U4 <- psi.G(R4/V12, theta2)
U <- cbind(U1, U2, U3, U4)
Gumbelplot4D <- splom2(U, cex = 0.1)

Gumbelplot4D

#### Sampling from a 4-dimensional nested Archimedean copula with a
#### AMH(Clayton(20)) generator structure (20 is 4.2.20 in p.118 of Nelson
#### (2007))

# Set our generator parameters
theta0 <- copAMH@iTau(0.2)
theta1 <- copClayton@iTau(0.4)
theta2 <- 1.3773

# Define our Archimedean generators
psi0 <- function(t) {
    (1 - theta0)/(exp(t) - theta0)
}
psi1 <- function(t) {
    (1 + t)^(-1/theta1)
}
psi2 <- function(t) {
    (log(t + exp(1)))^(-1/theta2)
}

alpha <- theta1/theta2
```

```r
N <- 1000  # number of samples

# Sample V0
V0 <- 1 + rgeom(n = N, p = 1 - theta0)


# Sample V01
V <- sapply(X = 1:N, FUN = function(n) rgamma(n = 1, shape = V0[n], rate = 1/(1 -
    theta0)))
Stilde <- sapply(X = 1:N, FUN = function(n) retstable(alpha = 1/theta1,
    V0 = 1, h = V[n]^theta1))
V01 <- Stilde * V^theta1


# Sample V12
h <- exp(1) - 1
m <- function(V) {
    # for the fast rejection algorithm
    psi <- function(t) {
        exp(-V * log(1 + t)^alpha)
    }
    c <- 1/psi(h)
    logc <- log(c)
    Flogc <- floor(logc)
    Clogc <- ceiling(logc)
    if (log(c) <= 1) {
        return(1)
    } else if (logc > 1 && Flogc * c^(1/Flogc) <= Clogc * c^(1/Clogc)) {
        return(Flogc)
    } else if (logc > 1 && Flogc * c^(1/Flogc) > Clogc * c^(1/Clogc)) {
        return(Clogc)
    }
}
m <- Vectorize(m)

mV01 <- m(V01)
V12 <- rep(0, times = N)

for (i in 1:N) {
    # fast rejection algorithm
    for (k in 1:mV01[i]) {
        rej <- T
        while (rej) {
            Vtilde <- rgamma(n = 1, shape = rstable1(n = 1, alpha = alpha,
                beta = 1, gamma = (cos(alpha * pi/2) * V01[i]/mV01[i])^(1/alpha),
                delta = 0), rate = 1)
            U <- runif(n = 1)
```

```
            if (U <= exp(-h * Vtilde)) {
                # standard rejection algorithm
                rej <- F
            }
            V12[i] <- V12[i] + Vtilde
        }
    }
}

# Sample (U1, U2, U3, U4)
R1 <- rexp(n = N)
R2 <- rexp(n = N)
R3 <- rexp(n = N)
R4 <- rexp(n = N)
U1 <- psi0(R1/V0)
U2 <- psi1(R2/V01)
U3 <- psi2(R3/V12)
U4 <- psi2(R4/V12)
U <- cbind(U1, U2, U3, U4)

NACplot4D <- splom2(U, cex = 0.1)  # and produce a pairs plot

NACplot4D

#### Sampling from a 4-dimensional nested Archimedean copula with Gumbel
#### generators
N <- 1000  # number of samples

# Set our generator parameters
tau <- seq(0.1, 0.8, length.out = 8)
theta <- sapply(X = tau, FUN = function(t) copFrank@iTau(t))
alpha <- theta[1:7]/theta[2:8]

# Define our Archimedean generators
psi.F <- function(t, theta) {
    -log(1 - (1 - exp(-theta)) * exp(-t))/theta
}


# Sample V0 and V01
V <- matrix(0L, nrow = N, ncol = 8)
V[, 1] <- rlog(n = N, lp = exp(-theta0))
for (j in 2:7) {
    V[, j] <- rF01Frank(V[, j - 1], theta[j - 1], theta[j], r = 1, approx = 10000)
}

# Sample (U1,...,U8)
```

```r
U <- matrix(0L, nrow = N, ncol = 8)
for (j in 1:7) {
    U[, j] <- psi.F(rexp(n = N)/V[, j], theta[j])
}
U[, 8] <- psi.F(rexp(n = N)/V[, 7], theta[7])

Frankplot8D <- splom2(U, cex = 0.1)

Frankplot8D
```

References

Serge Bernstein et al. Sur les fonctions absolument monotones. *Acta Mathematica*, 52:1–66, 1929.

Nejc Bezak, Katarina Zabret, and Mojca Šraj. Application of copula functions for rainfall interception modelling. *Water*, 10(8):995, 2018.

Paolo Candio, Andrew J Hill, Stavros Poupakis, Anni-Maria Pulkki-Brännström, Chris Bojke, and Manuel Gomes. Copula models for addressing sample selection in the evaluation of public health programmes: An application to the leeds let's get active study. *Applied Health Economics and Health Policy*, pages 1–8, 2021.

Umberto Cherubini, Elisa Luciano, and Walter Vecchiato. *Copula methods in finance*. John Wiley & Sons, 2004.

Elena Di Bernardino and Didier Rullière. On an asymmetric extension of multivariate archimedean copulas based on quadratic form. *Dependence Modeling*, 4(1), 2016.

Justin T French, Hsiao-Hsuan Wang, William E Grant, and John M Tomeček. Dynamics of animal joint space use: a novel application of a time series approach. *Movement ecology*, 7(1):1–12, 2019.

Rüdiger Frey, Alexander J McNeil, and Mark Nyfeler. Copulas and credit models. *Risk*, 10 (111114.10), 2001.

Christian Genest and R Jock MacKay. Copules archimédiennes et families de lois bidimensionnelles dont les marges sont données. *Canadian Journal of Statistics*, 14(2):145–159, 1986.

Jan Górecki, Marius Hofert, and Martin Holena. On the consistency of an estimator for hierarchical archimedean copulas. In *32nd International Conference on Mathematical Methods in Economics*, pages 239–244, 2014.

Oliver Grothe and Marius Hofert. Construction and sampling of archimedean and nested archimedean lévy copulas. *Journal of Multivariate Analysis*, 138:182–198, 2015.

Wolfgang Karl Härdle, Ostap Okhrin, and Yarema Okhrin. Dynamic structured copula models. *Statistics & Risk Modeling*, 30(4):361–388, 2013.

Marius Hofert. *Sampling nested Archimedean copulas with applications to CDO pricing*. PhD thesis, Universität Ulm, 2010.

Marius Hofert. A stochastic representation and sampling algorithm for nested archimedean copulas, 2012.

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

Marius Hofert and Matthias Scherer. Cdo pricing with nested archimedean copulas. *Quantitative Finance*, 11(5):775–787, 2011.

Marius Hofert, Raphaël Huser, and Avinash Prasad. Hierarchical archimax copulas. *Journal of Multivariate Analysis*, 167:195–211, 2018.

Eugen Ivanov, Aleksey Min, and Franz Ramsauer. Copula-based factor models for multivariate asset returns. *Econometrics*, 5(2):20, 2017.

Harry Joe. *Multivariate models and multivariate dependence concepts*. CRC Press, 1997.

Harry Joe. *Dependence modeling with copulas*. CRC press, 2014.

Clark H Kimberling. A probabilistic interpretation of complete monotonicity. *Aequationes mathematicae*, 10(2):152–164, 1974.

Enrico Marcantoni. *Collateralized Debt Obligations: A Moment Matching Pricing Technique Based on Copula Functions*. Springer Science & Business Media, 2014.

Albert W Marshall and Ingram Olkin. Families of multivariate distributions. *Journal of the American statistical association*, 83(403):834–841, 1988.

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

Roger B Nelsen. *An introduction to copulas*. Springer Science & Business Media, 2007.

Yuting Ng, Ali Hasan, Khalil Elkhalil, and Vahid Tarokh. Generative archimedean copulas. *arXiv preprint arXiv:2102.11351*, 2021.

Cathy Ning, Dinghai Xu, and Tony S Wirjanto. Is volatility clustering of asset returns asymmetric? *Journal of Banking & Finance*, 52:62–76, 2015.

Marius Ötting, Roland Langrock, and Antonello Maruotti. A copula-based multivariate hidden markov model for modelling momentum in football. *AStA Advances in Statistical Analysis*, pages 1–19, 2021.

Francesco Serinaldi and Salvatore Grimaldi. Fully nested 3-copula: procedure and application on hydrological data. *Journal of Hydrologic Engineering*, 12(4):420–430, 2007.

DV Widder. The laplace transformf. *Princeton Mathematical Series*, 1941.

Hui Zhang, Junyi Zhang, and Masashi Kuwano. An integrated model of tourists' time use and expenditure behaviour with self-selection based on a fully nested archimedean copula function. *Tourism Management*, 33(6):1562–1573, 2012.