



COMPUTO

ISSN 2824-7795

A Python Package for Sampling from Copulae: clayton

A Python Package for Sampling from Copulae: clayton

Alexis Boulin

Université Côte d’Azur, CNRS, LJAD, France
Inria, Lemon

Abstract

The package `clayton` is designed to be intuitive, user-friendly, and efficient. It offers a wide range of copula models, including Archimedean, Elliptical, and Extreme. The package is implemented in pure Python, making it easy to install and use. In addition, we provide detailed documentation and examples to help users get started quickly. We also conduct a performance comparison with existing R packages, demonstrating the efficiency of our implementation. The `clayton` package is a valuable tool for researchers and practitioners working with copulae in Python.

Keywords: Copulae, Random number generation

Contents

1	Introduction	2
2	Classes	3
2.1	The Archimedean class	3
2.2	The Extreme class	5
3	Random number generator	6
3.1	The bivariate case	7
3.2	The multivariate case	9
4	Case study : Modeling pairwise dependence between spatial maximas with missing data	12
4.1	Background	12
4.2	Numerical results	14
5	Discussion	17
5.1	Comparison of <code>clayton</code> with R packages	17
5.2	Conclusion	18
	References	19

6	Appendix	23
6.1	Bivariate Archimedean models	23
6.2	Implemented bivariate extreme models	25
6.3	Multivariate Archimedean copulae	26
6.4	Multivariate extreme models	26
6.5	Multivariate elliptical dependencies	28

1 Introduction

Modeling dependence relations between random variables is a topic of interest in probability theory and statistics. The most popular approach is based on the second moment of the underlying random variables, namely, the covariance. It is well known that only linear dependence can be captured by the covariance and it is only characteristic for a few models, e.g., the multivariate normal distribution or binary random variables. As a beneficial alternative to dependence, the concept of copulae, going back to Sklar (1959), has drawn a lot of attention. The copula $C : [0, 1]^d \rightarrow [0, 1]$ of a random vector $\mathbf{X} = (X_0, \dots, X_{d-1})$ with $d \geq 2$ allows us to separate the effect of dependence from the effect of the marginal distribution, such that:

$$\mathbb{P}\{X_0 \leq x_0, \dots, X_{d-1} \leq x_{d-1}\} = C(\mathbb{P}\{X_0 \leq x_0\}, \dots, \mathbb{P}\{X_{d-1} \leq x_{d-1}\}),$$

where $(x_0, \dots, x_{d-1}) \in \mathbb{R}^d$. The main consequence of this identity is that the copula completely characterizes the stochastic dependence between the margins of \mathbf{X} .

In other words, copulae allow us to model marginal distributions and dependence structure separately. Furthermore, motivated by Sklar's theorem, the problem of investigating stochastic dependence is reduced to the study of multivariate distribution functions under the unit hypercube $[0, 1]^d$ with uniform margins. The theory of copulae has been of prime interest for many applied fields of science, such as quantitative finance (Patton (2012)) or environmental sciences (Mishra and Singh (2011)). This increasing number of applications has led to a demand for statistical methods. For example, semiparametric estimation (Genest, Ghoudi, and Rivest (1995)), nonparametric estimation (Fermanian, Radulović, and Wegkamp (2004)) of copulae or nonparametric estimation of conditional copulae (Gijbels, Omelka, and Veraverbeke (2015), Portier and Segers (2018)) have been investigated. These results are established for a fixed arbitrary dimension $d \geq 2$, but several investigations (e.g. Einmahl and Lin (2006), Einmahl and Segers (2021)) are done for functional data for the tail copula, which captures dependence in the upper tail.

Software implementation of copulae has been extensively studied in R, for example in the packages A. G. Stephenson (2002), Jun Yan (2007), Schepsmeier et al. (2019). However, methods for working with copulae in Python are still limited. As far as we know, copula-dedicated packages in Python are mainly designed for modeling, such as Alvarez et al. (2021) and Bock and Chapman (2021). These packages use maximum likelihood methods to estimate the copula parameters from observed data and generate synthetic data using the estimated copula model. Other packages provide sampling methods for copulae, but they are typically restricted to the

bivariate case and the conditional simulation method (see, for example, Baudin et al. (2017)). Additionally, if the multivariate case is considered only Archimedean and elliptical copulae are under interest and those packages (see Nicolas (2022)) do not include the extreme value class in arbitrary dimensions $d \geq 2$. In this paper, we propose to implement a wide range of copulae, including the extreme value class, in arbitrary fixed dimension $d \geq 2$.

Through this paper we adopt the following notational conventions: all the indices will start at 0 as in Python. Consider $(\Omega, \mathcal{A}, \mathbb{P})$ a probability space and let $\mathbf{X} = (X_0, \dots, X_{d-1})$ be a d -dimensional random vector with values in $(\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d))$, with $d \geq 2$ and $\mathcal{B}(\mathbb{R}^d)$ the Borel σ -algebra of \mathbb{R}^d . This random vector has a joint distribution F with copula C and its margins are denoted by $F_j(x) = \mathbb{P}\{X_j \leq x\}$ for all $x \in \mathbb{R}$ and $j \in \{0, \dots, d-1\}$. Denote by $\mathbf{U} = (U_0, \dots, U_{d-1})$ a d random vector with copula C and uniform margins. All bold letters \mathbf{x} will denote a vector of \mathbb{R}^d .

The clayton package, whose Python code can be found in <https://github.com/Aleboul/clayton>, uses object-oriented features of the Python language. The package contains classes for Archimedean, elliptical, and extreme value copulae. In Section 2, we briefly describe the classes defined in the package. Section 3 presents methods for generating random vectors. In Section 4, we apply the clayton package to model pairwise dependence between maxima. Section 5 discusses potential improvements to the package and provides concluding remarks. Sections from Section 6.1 to Section 6.5 define and illustrate all the parametric copula models implemented in the package.

2 Classes

The architecture of the code is shown in Figure 1. At the third level of the architecture, we find important parametric models of Archimedean and extreme value copulae (depicted as blue in the figure). These parametric models contain methods such as the generator function φ (see Section 2.1) for Archimedean copulae and the Pickands dependence function A (see Section 2.2) for extreme value copulae (depicted as green in the figure). We provide a brief overview of Archimedean copulae and some of their properties in high-dimensional spaces in Section 2.1. A characterization of extreme value copulae is given in Section 2.2. The from Section 6.1 to Section 6.5 define and illustrate all the copula models implemented in the package.

2.1 The Archimedean class

Let φ be a generator that is a strictly decreasing, convex function from $[0, 1]$ to $[0, \infty]$ such that $\varphi(1) = 0$ and $\varphi(0) = \infty$. We denote the generalized inverse of φ by φ^\leftarrow . Consider the following equation:

$$C(\mathbf{u}) = \varphi^\leftarrow(\varphi(u_0) + \dots + \varphi(u_{d-1})). \quad (1)$$

If this relation holds and C is a copula function, then C is called an Archimedean copula. A necessary condition for Equation 1 to be a copula is that the generator φ is a d -monotonic function, i.e., it is differentiable up to the order d and its derivatives satisfy

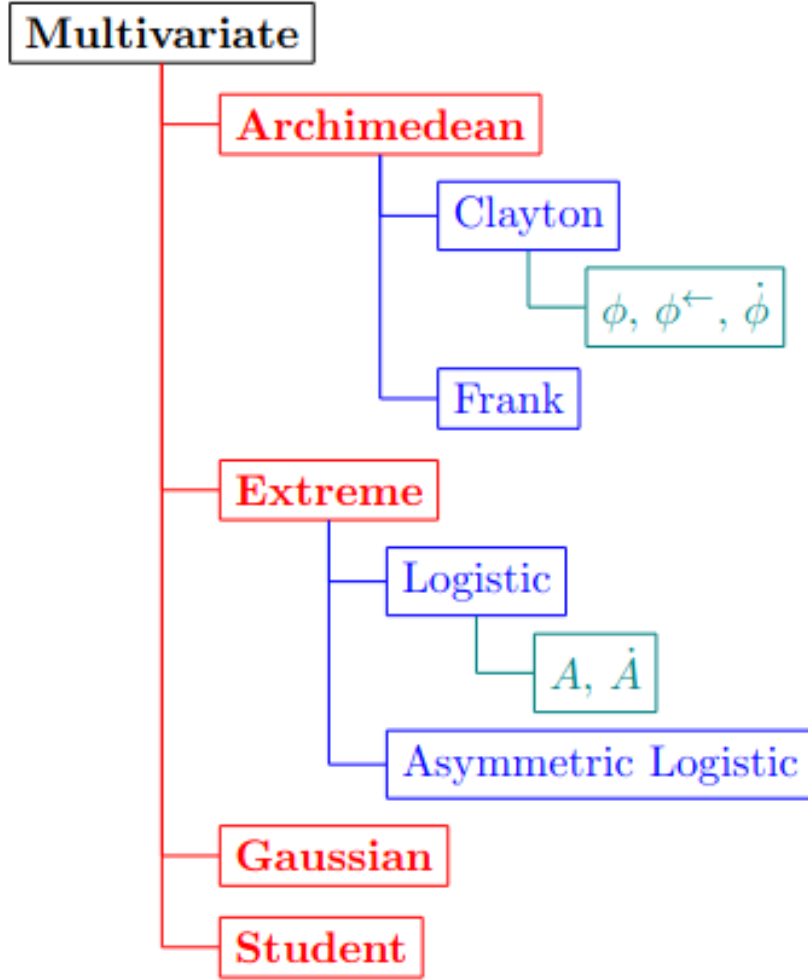


Figure 1: The figure shows a object diagram that structures the code. The **Multivariate** class serves as the root and is used to instantiate all its child classes **Archimedean**, **Extreme**, **Gaussian**, and **Student** in red. The blue-colored classes correspond to various parametric copula models, and the green-colored classes represent examples of methods. Symbols $\phi, \phi^{\leftarrow}, \dot{\phi}$ correspond to the generator function, its inverse, and its derivative, respectively, while A, \dot{A} refer to the Pickands dependence function and its derivative.

$$(-1)^k (\varphi)^{(k)}(x) \geq 0, \quad k \in \{1, \dots, d\} \quad (2)$$

for $x \in (0, \infty)$ (see Corollary 2.1 of McNeil and Nešlehová (2009)). Note that d -monotonic Archimedean inverse generators do not necessarily generate Archimedean copulae in dimensions higher than d (see McNeil and Nešlehová (2009)). As a result, some Archimedean subclasses are only implemented for the bivariate case as they do not generate an Archimedean copula in higher dimensions. In the bivariate case, Equation 2 can be interpreted as φ being a convex function.

The clayton package implements common one-parameter families of Archimedean copulae, such as the Clayton (Clayton (1978)), Gumbel (Gumbel (1960)), Joe (Joe (1997)), Frank (Frank (1979)), and AMH (Ali, Mikhail, and Haq (1978)) copulae for the multivariate case. It is worth noting that all Archimedean copulae are symmetric, and in dimensions 3 or higher, only positive associations are allowed. For the specific bivariate case, the package also implements other families, such as those numbered from 4.2.9 to 4.2.15 and 4.2.22 in Section 4.2 of Nelsen (2007). Definitions and illustrations of these parametric copula models can be found in Section 6.1 and Section 6.3.

2.2 The Extreme class

Investigating the notion of copulae within the framework of multivariate extreme value theory leads to the extreme value copulae (see Gudendorf and Segers (2010) for an overview) defined as

$$C(\mathbf{u}) = \exp(-\ell(-\ln(u_0), \dots, -\ln(u_{d-1}))), \quad \mathbf{u} \in (0, 1]^d, \quad (3)$$

where $\ell : [0, \infty)^d \rightarrow [0, \infty)$ the stable tail dependence function which is convex, homogeneous of order one, namely $\ell(c\mathbf{x}) = c\ell(\mathbf{x})$ for $c > 0$ and satisfies $\max(x_0, \dots, x_{d-1}) \leq \ell(x_0, \dots, x_{d-1}) \leq x_0 + \dots + x_{d-1}$, $\forall \mathbf{x} \in [0, \infty)^d$. Let $\Delta^{d-1} = \{\mathbf{w} \in [0, 1]^d : w_0 + \dots + w_{d-1} = 1\}$ be the unit simplex. The Pickands dependence function $A : \Delta^{d-1} \rightarrow [1/d, 1]$ characterizes ℓ by its homogeneity, which is the restriction of ℓ to the unit simplex Δ^{d-1} :

$$\ell(x_0, \dots, x_{d-1}) = (x_0 + \dots + x_{d-1})A(w_0, \dots, w_{d-1}), \quad w_j = \frac{x_j}{x_0 + \dots + x_{d-1}}, \quad (4)$$

for $j \in \{1, \dots, d-1\}$ and $w_0 = 1 - w_1 - \dots - w_{d-1}$ with $\mathbf{x} \in [0, \infty)^d \setminus \{\mathbf{0}\}$. The Pickands dependence function characterizes the extremal dependence structure of an extreme value random vector and verifies $\max\{w_0, \dots, w_{d-1}\} \leq A(w_0, \dots, w_{d-1}) \leq 1$ where the lower bound corresponds to comonotonicity and the upper bound corresponds to independence. Estimating this function is an active area of research, with many compelling studies having been conducted on the topic (see, for example, Bücher, Dette, and Volgushev (2011), Gudendorf and Segers (2012)).

From a practical point of view, the family of extreme value copulae is very rich and arises naturally as the limiting distribution of properly normalised componentwise maxima. Furthermore, it contains a rich variety of parametric models and allows asymmetric dependence, that is, for the bivariate case:

$$\exists(u_0, u_1) \in [0, 1]^2, \quad C(u_0, u_1) \neq C(u_1, u_0).$$

In the multivariate framework, the logistic copula (or Gumbel, see Gumbel (1960)), the asymmetric logistic copula (Tawn (1990)), the Hüsler and Reiss distribution (Hüsler and Reiss (1989)), the t-EV copula (Demarta and McNeil (2005)), Bilogistic model (Smith (1990)) are implemented. It's worth noting that the logistic copula is the sole model that is both Archimedean and extreme value. The library includes bivariate extreme value copulae such as asymmetric negative logistic (Joe (1990)), asymmetric mixed (Tawn (1988)). The reader is again invited to read from Section 6.2 to Section 6.4 for precise definitions of these models.

3 Random number generator

We propose a Python-based implementation for generating random numbers from a wide variety of copulae. The clayton package requires a few external libraries that are commonly used in scientific computing in Python.

- `numpy` version 1.6.1 or newer. This is the fundamental package for scientific computing, it contains linear algebra functions and matrix / vector objects (Harris et al. (2020)).
- `scipy` version 1.7.1 or newer. A library of open-source software for mathematics, science and engineering (Virtanen et al. (2020)).

The clayton package provides two methods for generating random vectors: `sample_unimargin` and `sample`. The first method generates a sample where the margins are uniformly distributed on the unit interval $[0, 1]$, while the second method generates a sample from the chosen margins.

In Section 3.1, we present an algorithm that uses the conditioning method to sample from a copula. This method is very general and can be used for any copula that is sufficiently smooth (see Equation 5 and Equation 8 below). However, the practical infeasibility of the algorithm in dimensions higher than 2 and the computational intensity of numerical inversion call for more efficient ways to sample in higher dimensions. The purpose of Section 3.2 is to present such methods and to provide details on the methods used in the clayton package. In each section, we provide examples of code to illustrate how to instantiate a copula and how to sample with clayton.

In the following sections, we will use Python code that assumes that the following packages have been loaded:

```
import clayton
from clayton.rng import base, evd, archimedean, monte_carlo
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import norm, expon
plt.style.use('qb-light.mplstyle') # for fancy figures
np.random.seed(10)
```

3.1 The bivariate case

In this subsection, we address the problem of generating a bivariate sample from a specified joint distribution with $d = 2$. Suppose that we want to sample a bivariate random vector \mathbf{X} with copula C . In the case where the components are independent, the sampling procedure is straightforward: we can independently sample X_0 and X_1 . However, in the general case where the copula is not the independent copula, this approach is not applicable.

One solution to this problem is to use the conditioning method to sample from the copula. This method relies on the fact that given (U_0, U_1) with copula C , the conditional law of U_1 given U_0 is written as:

$$c_{u_0}(u_1) \triangleq \mathbb{P}\{U_1 \leq u_1 | U_0 = u_0\} = \frac{\partial C(u_0, u_1)}{\partial u_0}. \quad (5)$$

This allows us to first sample U_0 from a uniform distribution on the unit interval, and then to use the copula to generate U_1 given U_0 . Finally, we can transform the resulting sample (U_0, U_1) into the original space by applying the inverse marginal distributions F_0^{-1} and F_1^{-1} to U_0 and U_1 respectively. Thus, an algorithm for sampling bivariate copulae is given in Figure 2. Algorithm in Figure 2 presents a procedure for generating a bivariate sample from a copula. The algorithm takes as input the length of the sample n , as well as the parameters of the copula (θ, ψ_1, ψ_2) . The output is a bivariate sample from the desired copula model, denoted $\{(u_0^{(1)}, u_1^{(1)}), \dots, (u_0^{(n)}, u_1^{(n)})\}$. This algorithm is applicable as long as the copula has a first partial derivative with respect to its first component.

Algorithm 1 Conditional sampling from a copula

- 1: **Data:** sample's length n .
 - 2: Parameter of the copula θ, ψ_1, ψ_2 .
 - 3: **Result:** Bivariate sample from the desired copula model $\{(u_0^{(1)}, u_1^{(1)}), \dots, (u_0^{(n)}, u_1^{(n)})\}$.
 - 4: **procedure** SAMPLING($n, \theta, \psi_1, \psi_2$)
 - 5: Generate two independent uniform random observations on the $[0, 1]$ segment u_0 and t_1 .
 - 6: Set $u_1 = c_{u_0}^{\leftarrow}(t_1)$ where $c_{u_0}^{\leftarrow}$ denotes the generalized inverse of c_{u_0} .
 - 7: The desired pair is (u_0, u_1) .
-

Figure 2: Conditional sampling from copula

For step 6 of the algorithm, we need to find $u_1 \in [0, 1]$ such that $c_{u_0}(u_1) - t_1 = 0$ holds. This u_1 always exists because for every $u \in]0, 1[$, we have $0 \leq c_{u_0}(u) \leq 1$, and the function $u \mapsto c_{u_0}(u)$ is increasing (see Theorem 2.2.7 of Nelsen (2007) for a proof). This step can be solved using the brentq function from the scipy package. A sufficient condition for a copula to have a first partial derivative with respect to its first component in the Archimedean and extreme value cases is that the generator φ and the Pickands dependence function A are continuously differentiable on $]0, 1[$, respectively. In this case, the first partial derivatives of the copula are given by:

$$\frac{\partial C}{\partial u_0}(u_0, u_1) = \frac{\varphi'(u_0)}{\varphi'(C(u_0, u_1))}, \quad (u_0, u_1) \in]0, 1[^2, \quad (6)$$

$$\frac{\partial C}{\partial u_1}(u_0, u_1) = \frac{\varphi'(u_1)}{\varphi'(C(u_0, u_1))}, \quad (u_0, u_1) \in]0, 1[^2, \quad (7)$$

where $t = \ln(u_1)/\ln(u_0 u_1) \in (0, 1)$ and $\mu(t) = A(t) - tA'(t)$.

We now have all the necessary theoretical tools to give details on how the clayton package is designed. The file `base.py` contains the **Multivariate** class and the `sample` method to generate random numbers from **X** with copula C . To do so, we use the inversion method that is to sample from **U** using algorithm in Figure 2 and we compose the corresponding uniform margins by F_j^{\leftarrow} . (??) indicates that the sole knowledge of A and φ and their respective derivatives are needed in order to perform the sixth step of algorithm in Figure 2. For that purpose, `cond_sim` method located inside **Archimedean** and **Extreme** classes performs algorithm in Figure 2. Then each child of the bivariate **Archimedean** (resp. **Extreme**) class is thus defined by its generator φ (resp. A), its derivative φ' (resp. A') and its inverse φ^{\leftarrow} as emphasized in green in Figure 1. Namely, we perform algorithm in Figure 2 for the **Archimedean** subclasses Frank, AMH, Clayton (when $\theta < 0$ for the previous three), Nelsen_9, Nelsen_10, Nelsen_11, Nelsen_12, Nelsen_13, Nelsen_14, Nelsen_15 and Nelsen_22. For the **Extreme** class, such algorithm is performed for the AsyNegLog and AsyMix. For other models, faster algorithms are known and thus implemented, we refer to Section 3.2 for details.

The following code illustrates the random vector generation for a bivariate Archimedean copula. By defining the parameter of the copula and the sample's length, the constructor for this copula is available and can be called using the `Clayton` method, such as:

```
n_samples, theta = 1024, -0.5
copula = archimedean.Clayton(theta=theta, n_samples=n_samples)
```

To obtain a sample with uniform margins and a Clayton copula, we can use the `sample_unimargin` method, as follows:

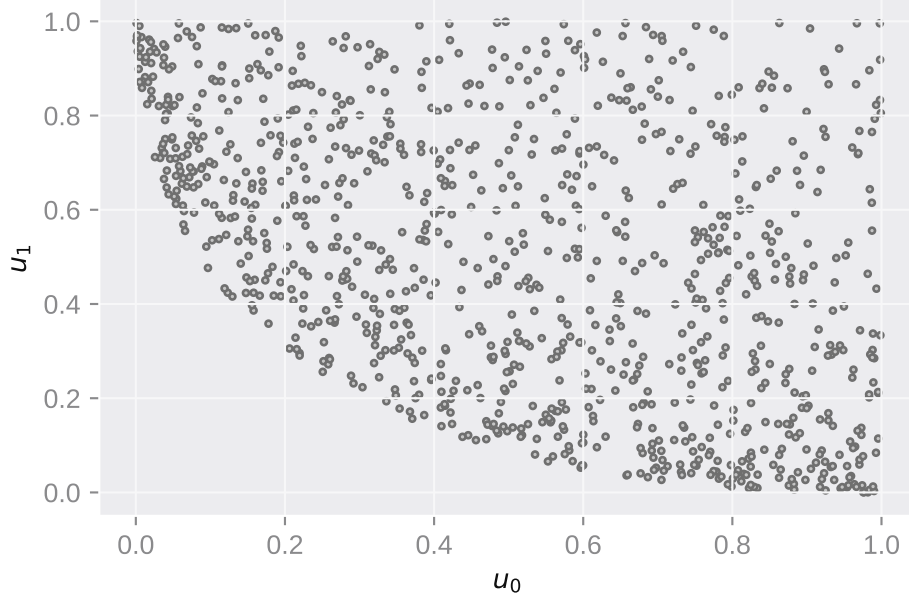
```
sample = copula.sample_unimargin()
```

Here, the `sample` object is a numpy array with 2 columns and 1024 rows, where each row contains a realization from a Clayton copula (see below)

```
fig, ax = plt.subplots()
ax.scatter(sample[:,0], sample[:,1],
           edgecolors='#6F6F6F', color='#C5C5C5', s=5)
ax.set_xlabel(r'$u_0$')
ax.set_ylabel(r'$u_1$')
```



```
plt.show()
```



3.2 The multivariate case

We will now address the generation of multivariate Archimedean and Extreme value copulae proposed in the Clayton package. In the multivariate case, the link between partial derivatives and the conditional law remains. Indeed, let (U_0, \dots, U_{d-1}) be a d -dimensional random vector with uniform margins and copula C . The conditional distribution of U_k given the values of U_0, \dots, U_{k-1} is

$$\mathbb{P}\{U_k \leq u_k | U_0 = u_0, \dots, U_{k-1} = u_{k-1}\} = \frac{\partial^{k-1} C(u_0, \dots, u_k, 1, \dots, 1) / \partial u_0 \dots \partial u_{k-1}}{\partial^{k-1} C(u_0, \dots, u_{k-1}, 1, \dots, 1) / \partial u_0 \dots \partial u_{k-1}}. \quad (8)$$

for $k \in 1, \dots, d-1$. The conditional simulation algorithm may be written as follows.

1. Generate d independent uniform random on $[0, 1]$ variates v_0, \dots, v_{d-1} .
2. Set $u_0 = v_0$.
3. For $k = 1, \dots, d-1$, evaluate the inverse of the conditional distribution given by (??) at v_k , to generate u_k .

Nevertheless, the evaluation of the inverse conditional distribution becomes increasingly complicated as the dimension d increases. Furthermore, it can be difficult for some models to derive a closed form of Equation 8 that makes it impossible to implement it in a general algorithm with only the dimension d as an input. For multivariate Archimedean copulae, McNeil and

Nešlehová (2009) give a method to generate a random vector from the d -dimensional copula C with generator φ (see Section 5.2 of McNeil and Nešlehová (2009)). A stochastic representation for Archimedean copulae generated by a d -monotone generator is given by

$$\mathbf{U} = (\varphi^{\leftarrow}(RS_1), \dots, \varphi^{\leftarrow}(RS_d)) \sim C, \quad (9)$$

where $R \sim F_R$, the radial distribution which is independent of S and S is distributed uniformly in the unit simplex Δ^{d-1} . One challenging aspect of this algorithm is to have an accurate evaluation of the radial distribution of the Archimedean copula and thus to numerically inverse this distribution. The associated radial distribution for the Clayton copula is given in Example 3.3 McNeil and Nešlehová (2009) while those of the Joe, AMH, Gumbel and Frank copulae are given in Hofert, Mächler, and McNeil (2012). In general, one can use numerical inversion algorithms for computing the inverse of the radial distribution, however it will lead to spurious numerical errors. Other algorithms exist when the generator is known to be the Laplace-Stieltjes transform, denoted as \mathcal{LS} , of some positive random variables (see Marshall and Olkin (1988), Frees and Valdez (1998)). This positive random variable is often referenced as the frailty distribution. In this framework, Archimedean copulae allow for the stochastic representation

$$\mathbf{U} = (\varphi^{\leftarrow}(E_1/V), \dots, \varphi^{\leftarrow}(E_d/V)) \sim C,$$

with $V \sim F = \mathcal{LS}^{-1}[\varphi^{\leftarrow}]$ the frailty and E_1, \dots, E_d are distributed i.i.d. according to a standard exponential and independent of V . Algorithm in Figure 3 presents a procedure for generating a multivariate sample from an Archimedean copula where the frailty distribution is known. The algorithm takes as an input the length of the sample n , as well as the parameter of the copula θ . The output is a d -variate sample from the desired copula model, denoted $\{(u_0^{(1)}, \dots, u_{d-1}^{(1)}), \dots, (u_0^{(n)}, \dots, u_{d-1}^{(n)})\}$.

Algorithm 2 Sampling from Archimedean copula using frailty distribution

- 1: **Data:** sample's length n .
 - 2: Parameter of the copula θ .
 - 3: **Result:** multivariate sample from the desired copula model.
 - 4: **procedure** `SAMPLING`(n, θ)
 - 5: Sample $V \sim F = \mathcal{LS}^{-1}[\varphi^{\leftarrow}]$.
 - 6: Sample $E_1, \dots, E_d \stackrel{i.i.d.}{\sim} \mathcal{E}(1)$, independent of V .
 - 7: Return $\mathbf{U} = (\varphi^{\leftarrow}(E_1/V), \dots, \varphi^{\leftarrow}(E_d/V))$.
-

Figure 3: Sampling from Archimedean copula using frailty distribution

In this framework, we define `_frailty_sim` method defined inside the **Archimedean** class which performs algorithm in Figure 3. Then, each Archimedean copula is defined by the generator φ , it's inverse φ^{\leftarrow} and the frailty distribution denoted as $\mathcal{LS}^{-1}[\varphi^{\leftarrow}]$ as long as we know the frailty. This is the case for Joe, Clayton, AMH or Frank.

For the extreme value case, algorithms have been proposed, as in A. Stephenson (2003) (see Algorithms 2.1 and 2.2), who proposes sampling methods for the Gumbel and the asymmetric logistic model. These algorithms are implemented in the `clayton` package. Note that these algorithms are model-specific, thus the `sample_unimargin` method is exceptionally located in the corresponding child of the multivariate **Extreme** class. Another procedure designed by Dombry, Engelke, and Oesting (2016) to sample from multivariate extreme value models using extremal functions (see Algorithm 2 in Dombry, Engelke, and Oesting (2016)) is also of prime interest. For the implemented models using this algorithm, namely **Hüsler-Reiss**, **tEV**, **Bilogistic** and **Dirichlet** models, a method called `_rextfunc` is located inside each classes which allows to generate an observation from the according law of the extremal function.

Samples from the Gaussian and Student copula are directly given by Algorithm 5.9 and 5.10 respectively of Alexander J. McNeil (2005). As each algorithm is model specific, the `sample_unimargin` method is located inside the **Gaussian** and **Student** classes.

We present how to construct a multivariate Archimedean copula and to generate random vectors from this model. Introducing the parameters of the copula, we appeal the following lines to construct our copula object:

```
d, theta, n_samples = 3, 2.0, 1024
copula = archimedean.Clayton(theta=theta, n_samples=n_samples, dim=d)
```

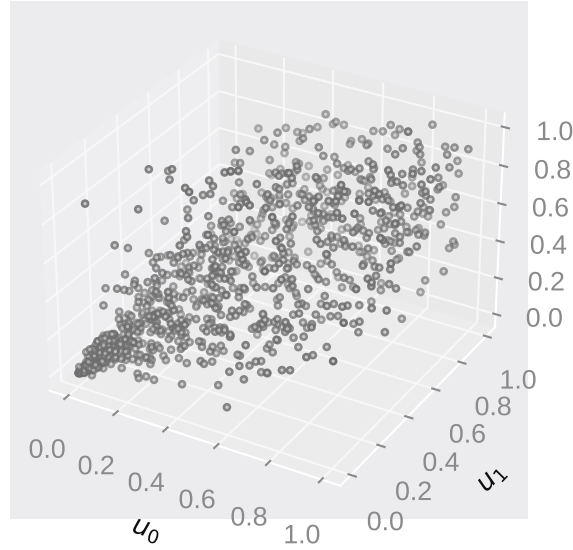
We now call the `sample_unimargin` method to obtain randomly generated vectors.

```
sample = copula.sample_unimargin()
```

We thus represent in three dimensions these realizations below.

```
fig = plt.figure()
ax = fig.add_subplot(111, projection = '3d')
ax.scatter3D(sample[:,0], sample[:,1], sample[:,2], s=5,
             edgecolors='#6F6F6F', color='#C5C5C5')
ax.set_xlabel(r'$u_0$')
ax.set_ylabel(r'$u_1$')
ax.set_zlabel(r'$u_2$')

plt.show()
```



4 Case study : Modeling pairwise dependence between spatial maximas with missing data

We now proceed to a case study where we use our python package to assess, under a finite sample framework, the asymptotic properties of an estimator of the λ -madogram when data are completely missing at random (MCAR). This case study comes from numerical results of Boulin et al. (2022). The λ -madogram belongs to a family of estimators, namely the madogram, which is of prime interest in environmental sciences, as it is designed to model pairwise dependence between maxima in space, see, e.g., Bernard et al. (2013), Bador et al. (2015), Saunders, Stephenson, and Karoly (2021) where the madogram was used as a dissimilarity measure to perform clustering. Where in several fields, for example econometrics (Wooldridge (2007)) or survey theory (Boistard, Chauvet, and Haziza (2016)), the MCAR hypothesis appears to be a strong hypothesis, this hypothesis is more realistic in environmental research as the missingness of one observation is usually due to instruments, communication and processing errors that may be reasonably supposed independent of the quantity of interest. In Section 4.1, we define objects and properties of interest while in Section 4.2 we describe a detailed tutorial in python and with clayton package to compare the asymptotic variance with an empirical counterpart of the λ -madogram with $\lambda = 0.5$.

4.1 Background

It was emphasized that the possible dependence between maxima can be described with the extreme value copula. This function is completely characterized by the Pickands dependence function (see Equation 4) where the latter is equivalent to the λ -madogram introduced by Naveau et al. (2009) and defined as

$$v(\lambda) = \mathbb{E} \left[\left| \{F_0(X_0)\}^{1/\lambda} - \{F_1(X_1)\}^{1/(1-\lambda)} \right| \right], \quad (10)$$

with $\lambda \in (0, 1)$, and if $\lambda = 0$ and $0 < u < 1$, then $u^{1/\lambda} = 0$ by convention. The λ -madogram took its inspiration from the extensively used geostatistics tool, the variogram (see Chapter 1.3 of Gaetan and Guyon (2008) for a definition and some classical properties). The λ -madogram can be interpreted as the L_1 -distance between the uniform margins elevated to the inverse of the corresponding weights λ and $1 - \lambda$. This quantity describes the dependence structure between extremes by its relation with the Pickands dependence function. If we suppose that C is an extreme value copula as in (??), we have

$$A(\lambda) = \frac{v(\lambda) + c(\lambda)}{1 - v(\lambda) - c(\lambda)}, \quad (11)$$

with $c(\lambda) = 2^{-1}(\lambda/(1 - \lambda) + (1 - \lambda)/\lambda)$ (see Proposition 3 of Marcon et al. (2017) for details).

We consider independent and identically distributed i.i.d. copies $\mathbf{X}_1, \dots, \mathbf{X}_n$ of \mathbf{X} . In presence of missing data, we do not observe a complete vector \mathbf{X}_i for $i \in \{1, \dots, n\}$. We introduce $\mathbf{I}_i \in \{0, 1\}^2$ which satisfies, $\forall j \in \{0, 1\}$, if $X_{i,j}$ is not observed then $I_{i,j} = 0$. To formalize incomplete observations, we introduce the incomplete vector $\tilde{\mathbf{X}}_i$ with values in the product space $\bigotimes_{j=1}^2 (\mathbb{R} \cup \{\text{NA}\})$ such as

$$\tilde{X}_{i,j} = X_{i,j}I_{i,j} + \text{NA}(1 - I_{i,j}), \quad i \in \{1, \dots, n\}, j \in \{0, \dots, d - 1\}.$$

We thus suppose that we observe a 4-tuple such as

$$(\mathbf{I}_i, \tilde{\mathbf{X}}_i), \quad i \in \{1, \dots, n\}, \quad (12)$$

i.e. at each $i \in \{1, \dots, n\}$, several entries may be missing. We also suppose that for all $i \in \{1, \dots, n\}$, \mathbf{I}_i are i.i.d copies from $\mathbf{I} = (I_0, I_1)$ where I_j is distributed according to a Bernoulli random variable $\mathcal{B}(p_j)$ with $p_j = \mathbb{P}(I_j = 1)$ for $j \in \{0, 1\}$. We denote by p the probability of observing completely a realization from \mathbf{X} , that is $p = \mathbb{P}(I_0 = 1, I_1 = 1)$. In Boulin et al. (2022), hybrid and corrected estimators, respectively denoted as $\hat{v}_n^{\mathcal{H}}$ and $\hat{v}_n^{\mathcal{H}*}$, are proposed to estimate nonparametrically the λ -madogram in presence of missing data completely at random. Furthermore, a closed expression of their asymptotic variances for $\lambda \in]0, 1[$ is also given. This result is summarized in the following proposition.

Theorem 4.1 (Boulin et al. (2022)). *Let $(\mathbf{I}_i, \tilde{\mathbf{X}}_i)_{i=1}^n$ be a sample given by Equation 12. For $\lambda \in]0, 1[$,*

if C is an extreme value copula with Pickands dependence function A , we have as $n \rightarrow \infty$

$$\begin{aligned}\sqrt{n}(\hat{v}_n^{\mathcal{H}}(\lambda) - v(\lambda)) &\xrightarrow{d} \mathcal{N}(0, \mathcal{S}^{\mathcal{H}}(p_1, p_2, p, \lambda)), \\ \sqrt{n}(\hat{v}_n^{\mathcal{H}^*}(\lambda) - v(\lambda)) &\xrightarrow{d} \mathcal{N}(0, \mathcal{S}^{\mathcal{H}^*}(p_1, p_2, p, \lambda)),\end{aligned}$$

where $\mathcal{S}^{\mathcal{H}}(p_1, p_2, p, \lambda)$ and $\mathcal{S}^{\mathcal{H}^*}(p_1, p_2, p, \lambda)$ are the asymptotic variances of the random variables.

4.2 Numerical results

Benefiting from generating data with clayton we are thus able, with Monte Carlo simulation, to assess theoretical results given by Theorem 4.1 in a finite sample setting. For that purpose, we implement a MonteCarlo class (in `monte_carlo.py` file) which contains some methods to perform some Monte Carlo iterations for a given extreme value copula. Now, we set up parameters to sample our bivariate dataset. For this subsection, we choose the asymmetric negative logistic model (see Section 6.2 for a definition) with parameters $\theta = 10, \psi_1 = 0.1, \psi_2 = 1.0$ and we define the following function:

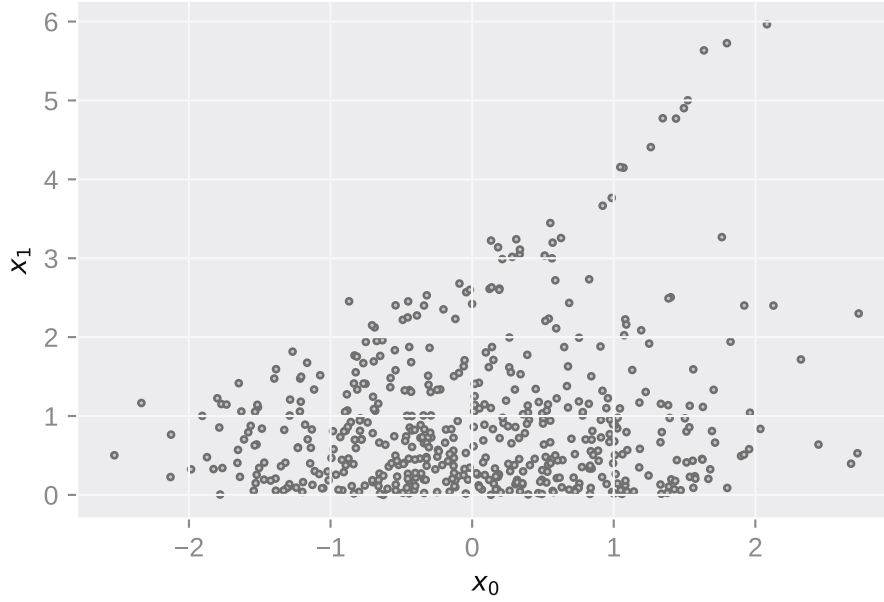
```
def gauss_function(x, x0, sigma):
    return (np.sqrt(1. / (2*np.pi * sigma**2)) *
            np.exp(-(x - x0) ** 2 / (2 * sigma**2)))
n_samples = 512
theta, psi1, psi2 = 10, 0.1, 1.0
```

We choose the standard normal and exponential as margins. To simulate this sample, the following lines should be typed:

```
copula = evd.AsyNegLog(theta=theta, psi1=psi1,
                      psi2=psi2, n_samples=n_samples)
sample = copula.sample(inv_cdf=[norm.ppf, expon.ppf])
```

The 1024×2 array `sample` contains 1024 realization of the **asymmetric negative logistic** model where the first column is distributed according to a standard normal random variable and the second column as a standard exponential. This distribution is depicted below. To obtain it, one needs the following lines of command:

```
fig, ax = plt.subplots()
ax.scatter(sample[:,0], sample[:,1],
          edgecolors="#6F6F6F", color="#C5C5C5", s=5)
ax.set_xlabel(r'$x_0$')
ax.set_ylabel(r'$x_1$')
plt.show()
```



Before going into further details, we will present the missing mechanism. Let V_0 and V_1 be random variables uniformly distributed under the $]0, 1[$ segment with copula $C_{(V_0, V_1)}$. We set $I_0 = 1\{V_0 \leq p_0\}$ and $I_1 = 1\{V_1 \leq p_1\}$. It is thus immediate that $I_0 \sim \mathcal{B}(p_0)$ and $I_1 \sim \mathcal{B}(p_1)$ and $p \triangleq \mathbb{P}\{I_0 = 1, I_1 = 1\} = C_{(V_0, V_1)}(p_0, p_1)$. For our illustration, we will take $C_{(V_0, V_1)}$ as a Joe copula with parameter $\theta = 2.0$ (we refer to Section 6.1 for a definition of this copula). For this copula, it is more likely to observe a realization $v_0 \geq 0.8$ from V_0 if $v_1 \geq 0.8$ from V_1 . If we observe $v_1 < 0.8$, the realization v_0 is close to being independent of v_1 . In climate studies, extreme events could damage the recording instrument in the surrounding regions where they occur, thus the missingness of one variable may depend on others. We initialize the copula $C_{(V_0, V_1)}$ with the following line:

```
copula_miss = archimedean.Joe(theta=2.0, n_samples=n_samples)
```

For a given $\lambda \in]0, 1[$, we now want to estimate a λ -madogram with a sample from the asymmetric negative logistic model, where some observations are missing due to the missing mechanism described above. We will repeat this step several times to compute an empirical counterpart of the asymptotic variance. The `MonteCarlo` object has been designed for this purpose: we specify the number of iterations n_{iter} (take $n_{iter} = 1024$), the chosen extreme value copula (asymmetric negative logistic model), the missing mechanism (described by $C_{(V_0, V_1)}$ and $p_0 = p_1 = 0.9$), and λ (noted w). We can write the following lines of code:

```
u = np.array([0.9, 0.9])
n_iter, P, w = 256, [[u[0], copula_miss._c(
    u)], [copula_miss._c(u), u[1]]], np.array([0.5, 0.5])
```

```

monte = monte_carlo.MonteCarlo(n_iter=n_iter, n_samples=n_samples,
                               copula=copula, copula_miss=copula_miss,
                               weight=w, matp=P)

```

The MonteCarlo object is thus initialized with all parameters needed. We may use the `simu` method to generate a DataFrame (a pandas object) composed out 1024 rows and 3 columns. Each row contains an estimate of the λ -madogram, $\hat{v}_n^{\mathcal{H}*}$ in Theorem 4.1 (`var_mado`), the sample length n (`n`) and the normalized estimation error (`scaled`). We thus call the `simu` method.

```

df_wmado = monte.finite_sample(inv_cdf=[norm.ppf, expon.ppf], corr=True)
print(df_wmado.head())

```

```

      wmado      n  scaled
0  0.148163  512.0 -0.128602
1  0.149337  512.0 -0.102024
2  0.153788  512.0 -0.001322
3  0.153169  512.0 -0.015324
4  0.155756  512.0  0.043209

```

The argument `corr=True` specifies that we compute the corrected estimator, $\hat{v}_n^{\mathcal{H}*}$ in Theorem 4.1. Now, using the `var_mado` method defined inside in the **Extreme** class, we obtain the asymptotic variance for the given model and parameters from the missing mechanism. We obtain this quantity as follows

```

var_mado = copula.var_mado(w, jointp=copula_miss._c(u), matp=P, corr=True)
print(var_mado)
print(df_wmado['scaled'].var())

```

```

0.015417245591834463
0.01370549107120327

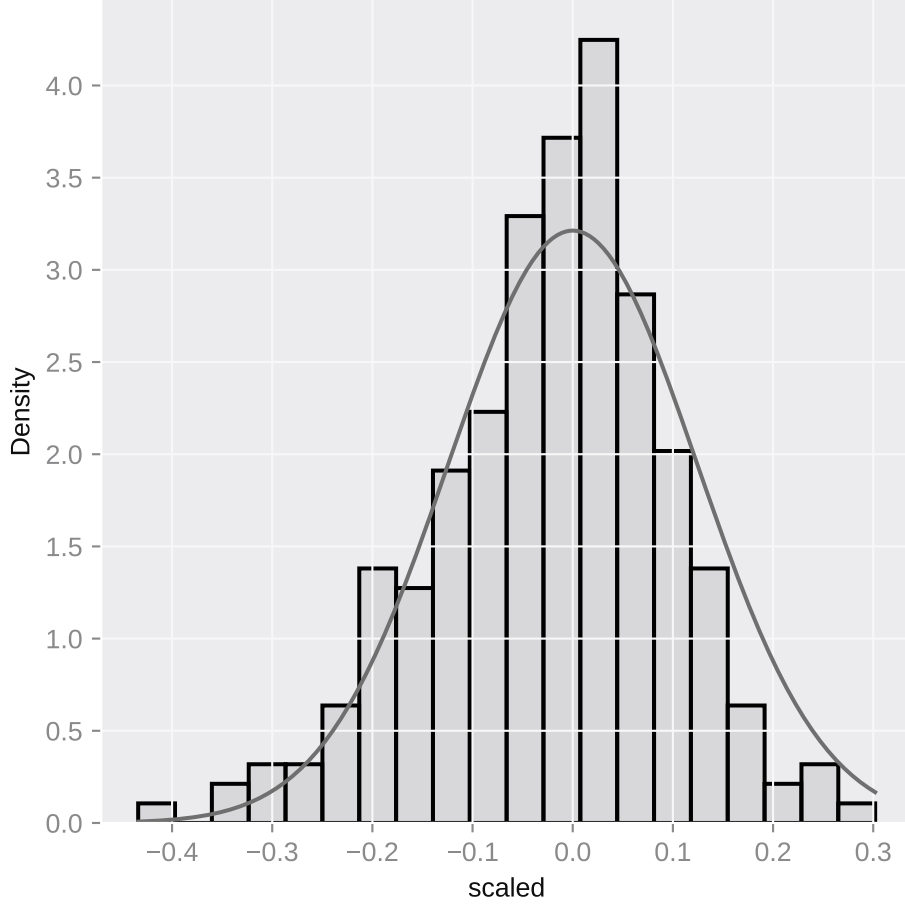
```

We propose here to check numerically the asymptotic normality with variance $\mathcal{S}^{\mathcal{H}*}$ of the normalized estimation error of the corrected estimator. We have all data in hand and the asymptotic variance was computed by lines above. We thus write:

```

sigma = np.sqrt(var_mado)
x = np.linspace(min(df_wmado['scaled']), max(df_wmado['scaled']), 1000)
gauss = gauss_function(x, 0, sigma)
sns.displot(data=df_wmado, x="scaled", color='#C5C5C5',
            kind='hist', stat='density', common_norm=False,
            alpha=0.5, fill=True, linewidth=1.5, bins=20)
plt.plot(x, gauss, color='#6F6F6F')

```

5 Discussion

5.1 Comparison of clayton with R packages

To compare clayton to existing packages in R, we consider the `copula` package (Kojadinovic and Yan (2010)) and `mev` (Belzile et al. (2022)) for sampling from Archimedean and multivariate extreme value distributions, respectively. To run the experiment, we use two computer clusters. The first cluster consists of five nodes, each with two 18-core Xeon Gold 3.1 GHz processors and 192 GB of memory, with 2933 MHz per socket. The second cluster has two CPU sockets, each containing a Xeon Platinum 8268 2.90 GHz processor with 24 cores. These configurations provide a significant amount of computational power and are well-suited for handling complex, data-intensive tasks. We use the first cluster to install the `copula` package and sample from the **Clayton**, **Frank**, and **Joe** models. We consider an increasing dimension $d \in \{50, 100, \dots, 1600\}$ for a fixed sample size of $n = 1000$. For the `copula` package, we compute the average time spent across 100 runs in order to cancel out variability. We use the second cluster to install the `mev` package and call some of its methods to sample from the **Husler Reiss**, **Logistic**, and

TEV distributions. Sampling from the latter is fast, but sampling from the two others is time consuming. Therefore, we only consider dimensions $d \in \{25, 50, \dots, 250\}$ for a fixed sample size of $n = 1000$.

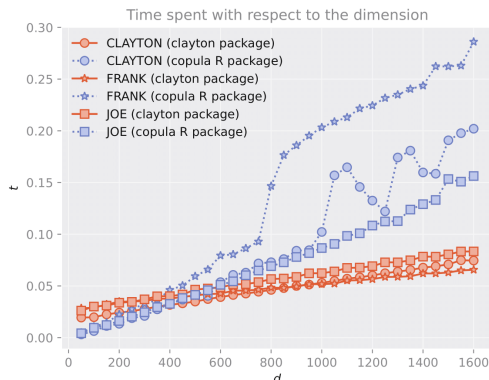


Figure 4: Archimedean

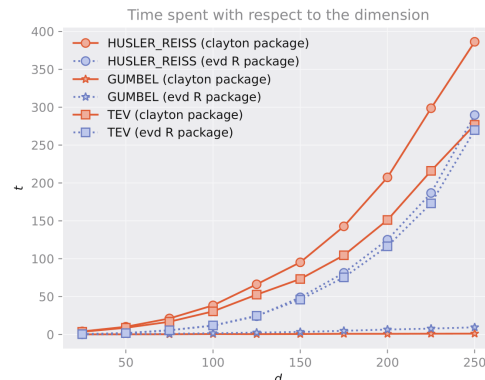


Figure 5: Multivariate extreme value

Figure 6: Comparison results. Time spent (in seconds) to sample from the corresponding models with respect to the dimension d . The left panel shows the results for sampling from **Clayton**, **Frank** and **Joe** using clayton in Python and copula in R. The right panel shows the results for sampling from **HuslerReiss**, **Logistic** and **TEV** by clayton in Python and mev in R. In both cases, 1000 vectors are generated for each model.

The figure shows the results of a comparison between the clayton and copula packages in R, and the mev package in Python. The comparison shows that the clayton package is more efficient at sampling from **Clayton**, **Frank** and **Joe** copulae than the copula package. The gap in efficiency may be due to the choice of algorithms used in the clayton package, which uses frailty distributions. The time required for sampling increases linearly with the dimension for the clayton package, but shows a more erratic behavior for the copula package.

5.2 Conclusion

This paper presents the construction and some implementations of the Python package clayton for random copula sampling. This is a seminal work in the field of software implementation of copula modeling in Python and there is much more potential for growth. It is hoped that the potential diffusion of the software through those who need it may bring further implementations for multivariate modeling with copulae under Python. For example, choosing a copula to fit the data is an important but difficult problem. A robust approach to estimating copulae has been investigated recently by Alquier et al. (2022) using Maximum Mean Discrepancy. In relation to our example, semiparametric estimation of copulae with missing data could be of great interest, as proposed by Hamori, Motegi, and Zhang (2019).

Additionally, implementation of the algorithm proposed by McNeil and Nešlehová (2009) for generating random vectors for Archimedean copulae has been tackled, but as expected, numerical

inversion gives spurious results, especially when the parameter θ and the dimension d are high. Furthermore, as the support of the radial distribution is contained in the real line, numerical inversion leads to increased computational time. Further investigation is needed in order to generate random vectors from classical Archimedean models using the radial distribution.

A direction of improvement for the clayton package is dependence modeling with Vine copulae, which have recently been a tool of high interest in the machine learning community (see, e.g., Lopez-Paz, Hernández-Lobato, and Zoubin (2013), Veeramachaneni, Cuesta-Infante, and O'Reilly (2015), Carrera, Santana, and Lozano (2016), Gonçalves, Von Zuben, and Banerjee (2016) or Sun, Cuesta-Infante, and Veeramachaneni (2019)). This highlights the need for dependence modeling with copulae in Python, as a significant part of the machine learning community uses this language. In relation to this paper, Vine copulae may be useful for modeling dependencies between extreme events, as suggested by Simpson, Wadsworth, and Tawn (2021), Nolde and Wadsworth (2021). Furthermore, other copula models could be implemented to model further dependencies. These implementations will expand the scope of dependence modeling with Python and provide high-quality, usable tools for anyone who needs them.

References

- Alexander J. McNeil, Paul Embrechts, Rudiger Frey. 2005. *Quantitative Risk Management - Concepts, Techniques and Tools*. Princeton Series in Finance. Princeton University Press. libgen.li/file.php?md5=478a0059673fec0c76229cd3d8884e7.
- Ali, Mir M, N. N Mikhail, and M.Safiul Haq. 1978. "A Class of Bivariate Distributions Including the Bivariate Logistic." *Journal of Multivariate Analysis* 8 (3): 405–12. [https://doi.org/https://doi.org/10.1016/0047-259X\(78\)90063-5](https://doi.org/https://doi.org/10.1016/0047-259X(78)90063-5).
- Alquier, Pierre, Badr-Eddine Chérif-Abdellatif, Alexis Derumigny, and Jean-David Fermanian. 2022. "Estimation of Copulas via Maximum Mean Discrepancy." *Journal of the American Statistical Association*, 1–16.
- Alvarez, M., C. Sala, Y. Sun, J. D. Pérez, K. A. Zhang, A. Montanez, G. Bonomi, K. Veeramachaneni, I. Ramírez, and F. A. Hofman. 2021. "Copulas." *GitHub Repository*. <https://github.com/sdv-dev/Copulas>; GitHub.
- Bador, Margot, Philippe Naveau, Eric Gilleland, Mercè Castellà, and Tatiana Arivelo. 2015. "Spatial Clustering of Summer Temperature Maxima from the CNRM-CM5 Climate Model Ensembles & E-OBS over Europe." *Weather and Climate Extremes* 9: 17–24. <https://doi.org/https://doi.org/10.1016/j.wace.2015.05.003>.
- Baudin, Michaël, Anne Dutfoy, Bertrand Iooss, and Anne-Laure Popelin. 2017. "Openturns: An Industrial Software for Uncertainty Quantification in Simulation." In *Handbook of Uncertainty Quantification*, 2001–38. Springer.
- Belzile, Leo et al. 2022. *mev: Modelling Extreme Values*. <https://CRAN.R-project.org/package=mev>.
- Bernard, Elsa, Philippe Naveau, Mathieu Vrac, and Olivier Mestre. 2013. "Clustering of Maxima: Spatial Dependencies among Heavy Rainfall in France." *Journal of Climate* 26 (20): 7929–37. <https://doi.org/10.1175/JCLI-D-12-00836.1>.
- Bock, Daniel, and Jacob Chapman. 2021. "Copulae." *GitHub Repository*. <https://github.com/>

[DanielBok/copulae](#); GitHub.

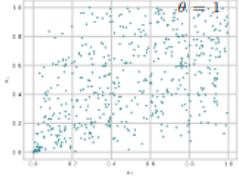
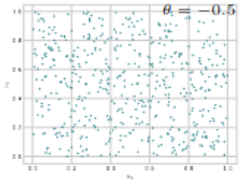
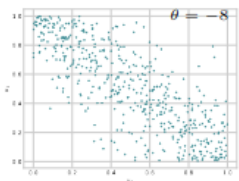
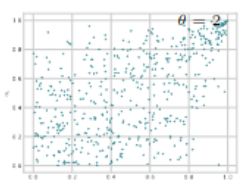


- Boistard, Helene, Guillaume Chauvet, and David Haziza. 2016. "Doubly Robust Inference for the Distribution Function in the Presence of Missing Survey Data." *Scandinavian Journal of Statistics* 43 (3): 683–99. <https://doi.org/https://doi.org/10.1111/sjos.12198>.
- Boulin, Alexis, Elena Di Bernardino, Thomas Laloë, and Gwladys Toulemonde. 2022. "Non-Parametric Estimator of a Multivariate Madogram for Missing-Data and Extreme Value Framework." *Journal of Multivariate Analysis* 192: 105059.
- Bücher, Axel, Holger Dette, and Stanislav Volgushev. 2011. "New Estimators of the Pickands Dependence Function and a Test for Extreme-Value Dependence." *The Annals of Statistics* 39 (4): 1963–2006.
- Carrera, Diana, Roberto Santana, and José Antonio Lozano. 2016. "Vine Copula Classifiers for the Mind Reading Problem." *Progress in Artificial Intelligence* 5: 289–305.
- Clayton, D. G. 1978. "A Model for Association in Bivariate Life Tables and Its Application in Epidemiological Studies of Familial Tendency in Chronic Disease Incidence." *Biometrika* 65 (1): 141–51. <http://www.jstor.org/stable/2335289>.
- Demarta, Stefano, and Alexander J. McNeil. 2005. "The t Copula and Related Copulas." *International Statistical Review* 73 (1): 111–29. <https://doi.org/https://doi.org/10.1111/j.1751-5823.2005.tb00254.x>.
- Dombry, Clément, Sebastian Engelke, and Marco Oesting. 2016. "Exact simulation of max-stable processes." *Biometrika* 103 (2): 303–17. <https://doi.org/10.1093/biomet/asw008>.
- Einmahl, John H. J., and Tao Lin. 2006. "Asymptotic Normality of Extreme Value Estimators on $\mathcal{C}([0, 1])$." *The Annals of Statistics* 34 (1): 469–92. <http://www.jstor.org/stable/25463423>.
- Einmahl, John H. J., and Johan Segers. 2021. "Empirical tail copulas for functional data." *The Annals of Statistics* 49 (5): 2672–96. <https://doi.org/10.1214/21-AOS2050>.
- Fermanian, Jean-David, Dragan Radulović, and Marten Wegkamp. 2004. "Weak Convergence of Empirical Copula Processes." *Bernoulli* 10 (5): 847–60. <https://doi.org/10.3150/bj/1099579158>.
- Frank, M. J. 1979. "On the Simultaneous Associativity of $f(x, y)$ and $x + y - f(x, y)$." *Aequationes Mathematicae* 19: 194–226. <http://eudml.org/doc/136825>.
- Frees, Edward W, and Emiliano A Valdez. 1998. "Understanding Relationships Using Copulas." *North American Actuarial Journal* 2 (1): 1–25.
- Gaetan, Carlo, and Xavier Guyon. 2008. *Modélisation Et Statistique Spatiales*. Mathématiques & Applications. Berlin Heidelberg New York: Springer.
- Genest, C., K. Ghoudi, and L.-P. Rivest. 1995. "A Semiparametric Estimation Procedure of Dependence Parameters in Multivariate Families of Distributions." *Biometrika* 82 (3): 543–52. <http://www.jstor.org/stable/2337532>.
- Gijbels, Irène, Marek Omelka, and Noël Veraverbeke. 2015. "Estimation of a Copula When a Covariate Affects Only Marginal Distributions." *Scandinavian Journal of Statistics* 42 (4): 1109–26. <http://www.jstor.org/stable/24586878>.
- Gonçalves, André R., Fernando J. Von Zuben, and Arindam Banerjee. 2016. "Multi-Task Sparse Structure Learning with Gaussian Copula Models." *J. Mach. Learn. Res.* 17 (1): 1205–34.
- Gudendorf, Gordon, and Johan Segers. 2010. "Extreme-Value Copulas." In *Copula Theory and Its Applications*, 198:127–45. Lect. Notes Stat. Proc. Springer, Heidelberg. https://doi.org/10.1007/978-3-642-12465-5/_6.

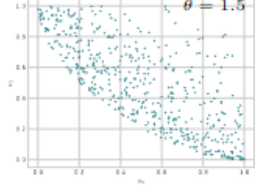
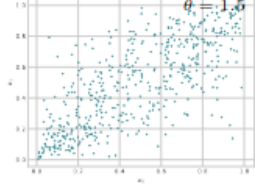
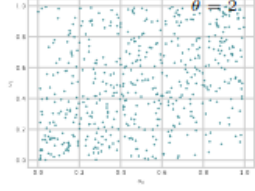
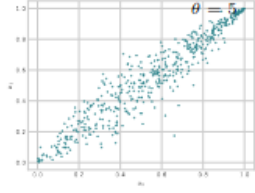
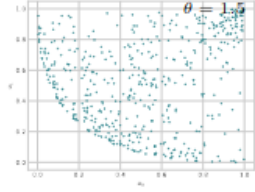
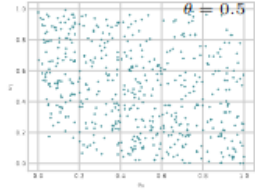
- . 2012. “Nonparametric Estimation of Multivariate Extreme-Value Copulas.” *Journal of Statistical Planning and Inference* 142 (12): 3073–85. <https://doi.org/https://doi.org/10.1016/j.jspi.2012.05.007>.
- Gumbel, E. J. 1960. “Distributions de Valeurs Extrêmes En Plusieurs Dimensions.” *Publications de l’institut de Statistique de l’Université de Paris* 9: 171–73.
- Hamori, Shigeyuki, Kaiji Motegi, and Zheng Zhang. 2019. “Calibration Estimation of Semiparametric Copula Models with Data Missing at Random.” *Journal of Multivariate Analysis* 173: 85–109. <https://doi.org/https://doi.org/10.1016/j.jmva.2019.02.003>.
- Harris, Charles R, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, et al. 2020. “Array Programming with NumPy.” *Nature* 585 (7825): 357–62.
- Hofert, Marius, Martin Mächler, and Alexander J McNeil. 2012. “Likelihood Inference for Archimedean Copulas in High Dimensions Under Known Margins.” *Journal of Multivariate Analysis* 110: 133–50.
- Hüsler, Jürg, and Rolf-Dieter Reiss. 1989. “Maxima of Normal Random Vectors: Between Independence and Complete Dependence.” *Statistics & Probability Letters* 7 (4): 283–86. [https://doi.org/https://doi.org/10.1016/0167-7152\(89\)90106-5](https://doi.org/https://doi.org/10.1016/0167-7152(89)90106-5).
- Joe, H. 1990. “Families of Min-Stable Multivariate Exponential and Multivariate Extreme Value Distributions.” *Statistics & Probability Letters* 9: 75–81.
- . 1997. *Multivariate Models and Multivariate Dependence Concepts*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis. <https://books.google.fr/books?id=iJbRZL2QzMAC>.
- Jun Yan. 2007. “Enjoy the Joy of Copulas: With a Package copula.” *Journal of Statistical Software* 21 (4): 1–21. <https://www.jstatsoft.org/v21/i04/>.
- Kojadinovic, Ivan, and Jun Yan. 2010. “Modeling Multivariate Distributions with Continuous Margins Using the Copula r Package.” *Journal of Statistical Software* 34: 1–20.
- Lopez-Paz, David, Jose Miguel Hernández-Lobato, and Ghahramani Zoubin. 2013. “Gaussian Process Vine Copulas for Multivariate Dependence.” In *International Conference on Machine Learning*, 10–18. PMLR.
- Marcon, G., S. A. Padoan, P. Naveau, P. Muliere, and J. Segers. 2017. “Multivariate Nonparametric Estimation of the Pickands Dependence Function Using Bernstein Polynomials.” *Journal of Statistical Planning and Inference* 183: 1–17. <https://doi.org/https://doi.org/10.1016/j.jspi.2016.10.004>.
- Marshall, Albert W., and Ingram Olkin. 1988. “Families of Multivariate Distributions.” *Journal of the American Statistical Association* 83 (403): 834–41. <http://www.jstor.org/stable/2289314>.
- McNeil, Alexander J., and Johanna Nešlehová. 2009. “Multivariate Archimedean copulas, d-monotone functions and l1-norm symmetric distributions.” *The Annals of Statistics* 37 (5B): 3059–97. <https://doi.org/10.1214/07-AOS556>.
- Mishra, Ashok K., and Vijay P. Singh. 2011. “Drought Modeling – a Review.” *Journal of Hydrology* 403 (1): 157–75. <https://doi.org/https://doi.org/10.1016/j.jhydrol.2011.03.049>.
- Naveau, Philippe, Armelle Guillou, Daniel Cooley, and Jean Diebolt. 2009. “Modelling Pairwise Dependence of Maxima in Space.” *Biometrika* 96 (1): 1–17. <https://doi.org/10.1093/biomet/asp001>.

- Nelsen, Roger B. 2007. *An Introduction to Copulas*. Springer Science & Business Media.
- Nicolas, Maxime LD. 2022. "Pycop: A Python Package for Dependence Modeling with Copulas." *Zenodo Software Package* 70: 7030034.
- Nolde, Natalia, and Jennifer L. Wadsworth. 2021. "Linking Representations for Multivariate Extremes via a Limit Set." <https://arxiv.org/abs/2012.00990>.
- Patton, Andrew J. 2012. "A Review of Copula Models for Economic Time Series." *Journal of Multivariate Analysis* 110: 4–18.
- Portier, François, and Johan Segers. 2018. "On the Weak Convergence of the Empirical Conditional Copula Under a Simplifying Assumption." *Journal of Multivariate Analysis* 166: 160–81. <https://doi.org/https://doi.org/10.1016/j.jmva.2018.03.002>.
- Saunders, K., A. Stephenson, and David Karoly. 2021. "A Regionalisation Approach for Rainfall Based on Extremal Dependence." *Extremes* 24 (June). <https://doi.org/10.1007/s10687-020-00395-y>.
- Schepsmeier, U., J. Stoeber, E. C. Brechmann, B. Graeler, T. Nagler, T. Erhardt, C. Almeida, et al. 2019. "VineCopula : Statistical Inference of Vine Copulas." *Package "VineCopula". R Package, Version 2.3.0*. <https://cran.r-project.org/web/packages/VineCopula/index.html>.
- Simpson, Emma S., Jennifer L. Wadsworth, and Jonathan A. Tawn. 2021. "A Geometric Investigation into the Tail Dependence of Vine Copulas." *Journal of Multivariate Analysis* 184: 104736. <https://doi.org/https://doi.org/10.1016/j.jmva.2021.104736>.
- Sklar, Abe. 1959. "Fonctions de Répartition à n Dimensions Et Leurs Marges." *Publications de l'Institut de Statistique de l'Université de Paris* 8: 229–31.
- Smith, R. L. 1990. *Extreme Value Theory*. Handbook of Applicable Mathematics (ed. W. Ledermann), vol. 7. Chichester: John Wiley, pp. 437–471.
- Stephenson, A. G. 2002. "evd: Extreme Value Distributions." *R News* 2 (2). <https://CRAN.R-project.org/doc/Rnews/>.
- Stephenson, Alec. 2003. "Simulating Multivariate Extreme Value Distributions of Logistic Type." *Extremes* 6 (1): 49–59.
- Sun, Yi, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019. "Learning Vine Copula Models for Synthetic Data Generation." *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (01): 5049–57. <https://doi.org/10.1609/aaai.v33i01.33015049>.
- Tawn, Jonathan A. 1988. "Bivariate extreme value theory: Models and estimation." *Biometrika* 75 (3): 397–415. <https://doi.org/10.1093/biomet/75.3.397>.
- . 1990. "Modelling Multivariate Extreme Value Distributions." *Biometrika* 77 (2): 245–53. <http://www.jstor.org/stable/2336802>.
- Veeramachaneni, Kalyan, Alfredo Cuesta-Infante, and Una-May O'Reilly. 2015. "Copula Graphical Models for Wind Resource Estimation." In *IJCAI*.
- Virtanen, Pauli, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, et al. 2020. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python." *Nature Methods* 17 (3): 261–72.
- Wooldridge, Jeffrey M. 2007. "Inverse Probability Weighted Estimation for General Missing Data Problems." *Journal of Econometrics* 141 (2): 1281–301. <https://doi.org/https://doi.org/10.1016/j.jeconom.2007.02.002>.

6 Appendix

6.1 Bivariate Archimedean models

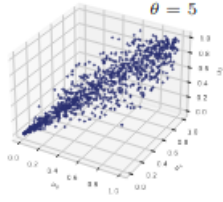
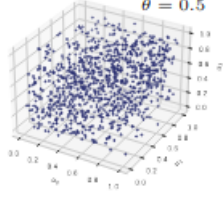
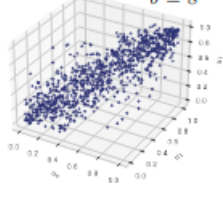
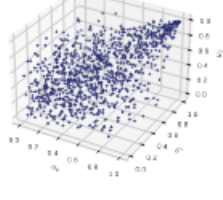
Name	$\varphi(t)$	Constraints	Figure
Clayton	$\frac{1}{\theta}(t^{-\theta} - 1)$	$\theta \in [-1, \infty) \setminus \{0\}$	
AMH	$\log(\frac{1-\theta(1-t)}{t})$	$\theta \in [-1, 1)$	
Frank	$-\log(\frac{e^{-\theta t}-1}{e^{-\theta}-1})$	$\theta \in \mathbb{R} \setminus \{0\}$	
Joe	$-\log(1 - (1-t)^\theta)$	$\theta \in [1, \infty)$	
Nelsen n°9	$\log(1 - \theta \log(t))$	$\theta \in]0, 1]$	
Nelsen n°10	$\log(2t^{-\theta} - 1)$	$\theta \in]0, 1]$	

Name	$\varphi(t)$	Constraints	Figure
Nelsen n°11	$\log(2 - t^\theta)$	$\theta \in]0, 0.5]$	
Nelsen n°12	$(\frac{1}{t} - 1)^\theta$	$\theta \in]0, \infty) \setminus \{0\}$	
Nelsen n°13	$(1 - \log(t))^\theta - 1$	$\theta \in]0, \infty[$	
Nelsen n°14	$(t^{-\frac{1}{\theta}} - 1)^\theta$	$\theta \in]1, \infty)$	
Nelsen n°15	$(1 - t^{\frac{1}{\theta}})^\theta$	$[1, \infty)$	
Nelsen n°22	$\arcsin(1 - t^\theta)$	$\theta \in [0, 1]$	

6.2 Implemented bivariate extreme models

Name	$A(w)$	Constraints	Figure
Gumbel	$[w^{1/\theta} + (1-w)^{1/\theta}]^\theta$	$\theta \in [0, 1]$	
Galambos	$1 - [w^{-\theta} + (1-w)^{-\theta}]^{-\frac{1}{\theta}}$	$\theta \in [0, \infty)$	
Asy. log.	$(1 - \psi_1)w + (1 - \psi_2)(1 - w) + [(\psi_1 w)^\theta + (\psi_2(1 - w))^\theta]^{\frac{1}{\theta}}$	$\theta \in [1, \infty)$ $\psi_1, \psi_2 \in (0, 1]$	
Asy. neg. log.	$1 - [(\psi_1 w)^{-\theta} + (\psi_2(1 - w))^{-\theta}]^{-\frac{1}{\theta}}$	$\theta \in [0, \infty)$ $\psi_1, \psi_2 \in (0, 1]$	
Asy. mixed	$1 - (\theta + \psi_1)w + \theta w^2 + \psi_1 w^3$	$\theta \geq 0,$ $\theta + 3\psi_1 \geq 0,$ $\theta + \psi_1 \leq 1,$ $\theta + 2\psi_1 \leq 1.$	
Husler Reiss	$(1 - w)\Phi(\theta + \frac{1}{2\theta}\log(\frac{1-w}{w})) + w\Phi(\theta + \frac{1}{2\theta}\log(\frac{w}{1-w}))$	$\theta \in (0, \infty)$	
t-EV	$wt_{\psi_1+1}(z_w) + (1-w)t_{\psi_1+1}(z_{1-w})$	$\theta \in (-1, 1)$	

6.3 Multivariate Archimedean copulae

Name	$\varphi(t)$	Constraints	Figure
Clayton	$\frac{1}{\theta}(t^{-\theta} - 1)$	$\theta \in [0, \infty) \setminus \{0\}$	
AMH	$\log(\frac{1-\theta(1-t)}{t})$	$\theta \in [-1, 1)$	
Frank	$-\log(\frac{e^{-\theta t} - 1}{e^{-\theta} - 1})$	$\theta \in \mathbb{R} \setminus \{0\}$	
Joe	$-\log(1 - (1 - t)^\theta)$	$\theta \in [1, \infty)$	

6.4 Multivariate extreme models

Before giving the main details, we introduce some notations. Let B be the set of all nonempty subsets of $\{1, \dots, d\}$ and $B_1 = \{b \in B, |b| = 1\}$, where $|b|$ denotes the number of elements in the set b . We note by $B_{(j)} = \{b \in B, j \in b\}$. For $d = 3$, the Pickands is expressed as

$$\begin{aligned}
 A(\mathbf{w}) = & \alpha_1 w_1 + \psi_1 w_2 + \phi_1 w_3 + \left((\alpha_2 w_1)^{\theta_1} + (\psi_2 w_2)^{\theta_1} \right)^{1/\theta_1} + \left((\alpha_3 w_2)^{\theta_2} + (\phi_2 w_3)^{\theta_2} \right)^{1/\theta_2} \\
 & + \left((\psi_3 w_2)^{\theta_3} + (\phi_3 w_3)^{\theta_3} \right)^{1/\theta_3} + \left((\alpha_4 w_1)^{\theta_4} + (\psi_4 w_2)^{\theta_4} + (\phi_4 w_3)^{\theta_4} \right)^{1/\theta_4},
 \end{aligned}$$

where $\alpha = (\alpha_1, \dots, \alpha_4)$, $\psi = (\psi_1, \dots, \psi_4)$, $\phi = (\phi_1, \dots, \phi_4)$ are all elements of Δ^3 . We take $\alpha = (0.4, 0.3, 0.1, 0.2)$, $\psi = (0.1, 0.2, 0.4, 0.3)$, $\phi = (0.6, 0.1, 0.1, 0.2)$ and $\theta = (\theta_1, \dots, \theta_4) = (0.6, 0.5, 0.8, 0.3)$ as the dependence parameter.

The Dirichlet model is a mixture of m Dirichlet densities, that is

$$h(\mathbf{w}) = \sum_{k=1}^m \theta_k \frac{\Gamma(\sum_{j=1}^d \sigma_{kj})}{\prod_{j=1}^d \Gamma(\sigma_{kj})} \prod_{j=1}^d w_j^{\sigma_{kj}-1},$$

with $\sum_{k=1}^m \theta_k = 1$, $\sigma_{kj} > 0$ for $k \in \{1, \dots, m\}$ and $j \in \{1, \dots, d\}$. Let $\mathcal{D} \in [0, \infty)^{(d-1) \times (d-1)}$ denotes the space of symmetric strictly conditionnaly negative definite matrices that is

$$\mathcal{D}_k = \left\{ \Gamma \in [0, \infty)^{k \times k} : a^\top \Gamma a < 0 \text{ for all } a \in \mathbb{R}^k \setminus \{\mathbf{0}\} \text{ with } \sum_{j=1}^{d-1} a_j = 0, \right. \\ \left. \Gamma_{ii} = 0, \Gamma_{ij} = \Gamma_{ji}, \quad 1 \leq i, j \leq k \right\}.$$

For any $2 \leq k \leq d$, consider $m' = (m_1, \dots, m_k)$ with $1 \leq m_1 < \dots < m_k \leq d$ define

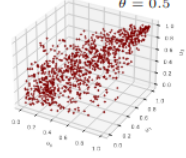
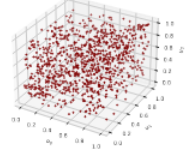
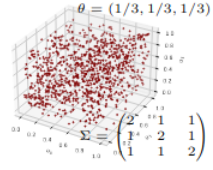
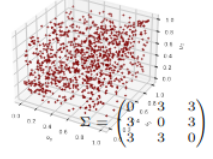
$$\Sigma_m^{(k)} = 2 \left(\Gamma_{m_i m_k} + \Gamma_{m_j m_k} - \Gamma_{m_i m_j} \right)_{m_i, m_j \neq m_k} \in [0, \infty)^{(d-1) \times (d-1)}.$$

Furthermore, note $S(\cdot | \Sigma_m^{(k)})$ denote the survival function of a normal random vector with mean vector $\mathbf{0}$ and covariance matrix $\Sigma_m^{(k)}$. We now define :

$$h_{km}(\mathbf{y}) = \int_{\mathcal{Y}_k}^\infty S\left((y_i - z + 2\Gamma_{m_i m_k})_{i=1}^{k-1} | \Gamma_{km}\right) e^{-z} dz$$

for $2 \leq k \leq d$. We denote by $\Sigma^{(k)}$ the summation over all k -vectors $m = (m_1, \dots, m_k)$ with $1 \leq m_1 < \dots < m_k \leq d$.

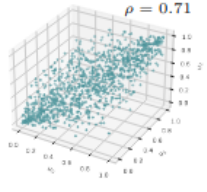
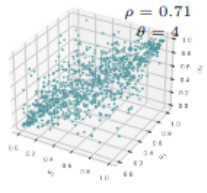
Table 5: Multivariate extreme models in **COPPY** module.

Name	$A(\mathbf{w})$	Constraints	Figure
Logistic	$\left(\sum_{j=1}^d w_j^{\frac{1}{\theta}}\right)^{\theta}$	$\theta \in]0, 1]$	
Asy. Log.	$\sum_{b \in B} \left(\sum_{j \in b} (\psi_{j,b} w_j)^{\frac{1}{\theta_b}}\right)^{\theta_b}$	$\theta_b \in]0, 1] \forall b \in B \setminus B_1,$ $\psi_{j,b} \in [0, 1] \forall b \in B \forall j \in b,$ $\sum_{b \in B(j)} \psi_{j,b} = 1, j \in \llbracket d-1 \rrbracket,$ $\theta_b = 1 \forall b \in B \setminus B_1 \implies$ $\psi_{j,b} = 0 \forall j \in b.$	
Dirichlet	Not specified	$\sum_{k=1}^m \theta_k = 1,$ $\sigma_{kj} > 0, k \in \{1, \dots, m\},$ $j \in \{1, \dots, d\}$	
Hüsler Reiss	$\sum_{k=1}^d (-1)^{k+1} \times \Sigma^{(k)} h_{km}(u_{m_1}, \dots, u_{m_k})$	$\Gamma \in \mathcal{D}_d$	

6.5 Multivariate elliptical dependencies

Let $\mathbf{X} \sim \mathbf{E}_d(\boldsymbol{\mu}, \Sigma, \psi)$ be an elliptical distributed random vector with cumulative distribution F and marginal F_0, \dots, F_{d-1} . Then, the copula C of F is called an elliptical copula. We denote by ϕ the standard normal distribution function and ϕ_{Σ} the joint distribution function of $\mathbf{X} \sim \mathcal{N}_d(\mathbf{0}, \Sigma)$, where $\mathbf{0}$ is the d -dimensional vector composed out of 0. In the same way, we note t_{θ} the distribution function of a standard univariate distribution t distribution and by $t_{\theta, \Sigma}$ the joint distribution function of the vector $\mathbf{X} \sim t_d(\theta, \mathbf{0}, \Sigma)$. A d squared matrix Σ is said to be positively semi definite if for all $u \in \mathbb{R}^d$ we have :

$$u^{\top} \Sigma u \geq 0$$

Name	C	Constraints	Figure
Gaussian	$\phi_{\Sigma}(\phi^{\leftarrow}(u_0), \dots, \phi^{\leftarrow}(u_{d-1}))$	Σ PSD	 <p>A 3D scatter plot showing a distribution of points in a 3D space. The axes are labeled u_0, u_1, and u_2. The points are colored in a gradient from blue to red. The plot is titled $\rho = 0.71$.</p>
Student	$t_{\theta, \Sigma}(t_{\theta}^{\leftarrow}(u_0), \dots, t_{\theta}^{\leftarrow}(u_{d-1}))$	$\theta > 0, \Sigma$ PSD	 <p>A 3D scatter plot showing a distribution of points in a 3D space. The axes are labeled u_0, u_1, and u_2. The points are colored in a gradient from blue to red. The plot is titled $\rho = 0.71$ and $\theta = 4$.</p>