Distributions and copulas in GOSM

Maël Forcier

April 2017

1 Introduction

We present a python library to use distribution with copulas.

We want to model the random properties of a data. In this section we will take the example of the production of wind power that we want to model by a gaussian.

Today we do not know what wind production there would be tomorrow., so we model this wind production by a random variable X that has a lot of properties.

2 Mathematical Definitions

Definition 1. *The cumulative density function (cdf) of a random variable $X \in \mathbb{R}$ is the function $F : \mathbb{R} \to [0, 1]$ where :*

$$F(x) = \mathbb{P}(X \leq x)$$

This definition also works when X is a multidimensional random variable i.e. $\in \mathbb{R}^d$ if we use the notation $(x_1, ..., x_d) \leq (y_1, ..., y_d) \Leftrightarrow \forall i, x_i \leq y_i$

Definition 2. *If the cdf of $X \in \mathbb{R}$ is differentiable, the probability density function (pdf) of X is the function $f : \mathbb{R} \to [0, 1]$ where :*

$$f(x) = \frac{dF(x)}{dx} = \frac{d\mathbb{P}(X \leq x)}{dx}$$

If $X \in \mathbb{R}^d$ and F is regular enough, the probability density function pdf of X is the function $f : \mathbb{R} \to [0, 1]$:

$$f(x) = \frac{d^d F(x)}{dx_1 ... dx_d}$$

Each one of these two functions gives all the informations about how the random variable X behaves. With them, we can compute different charasteristic values of the distribution.

Definition 3. *The mean of a random variable $X \in \mathbb{R}^d$ is :*

$$\mathbb{E}(X) = \int_{\mathbb{R}^d} x f(x) dx$$

Definition 4. *The variance of a random variable $X \in \mathbb{R}^d$ is :*

$$Var(X) = \mathbb{E}((X - \mathbb{E}(X))^2) = \int_{\mathbb{R}^d} (x - \mathbb{E}(X))^2 f(x) dx$$

Property 1. *If $X \in \mathbb{R}$ is a random variable with a continuous cdf F,*

then $U = F(X)$ is a uniform random variable in [0,1]

3 Distribution user manual

To model the random properties of data (for example the production of wind power or wind forecast errors). We define distribution as a python object that contains several methods. NB : This distribution python object does not correspond strictly to the mathematical definition of a distribution because it contains several different methods that describe many things linked to the probability law of the variable. To simplify, when we will talk about distribution, it will refer to this distribution python object.

$$\begin{tikzcd}
 & \boxed{input\_data} \arrow[d, "\_\_init\_\_"'] & \\
 & \boxed{parameters} \arrow[dl, "generates\_X"'] \arrow[dr, "calculus"] & \\
\boxed{sample} \arrow[rr, bend right, "counting"'] & & \boxed{functions} \arrow[ll, bend right, "CDF^{-1}(Uniform\ variable)"']
\end{tikzcd}$$

Figure 1: Simplified diagram of how a distribution object works

3.1 Methods and parameters of a distribution object

3.1.1 cdf, pdf and other functions

The methods cdf and pdf compute the two chararacteristic functions defined above. If the object `distrobject` represents a random variable X with cdf F and pdf f, `distrobject.cdf(x)` will return the value of $F(x)$ and `distrobject.pdf(x)` will return the value of $f(x)$. Most of the time, this value are calculated using analytical formulas.

For example, the pdf of a normal distribution is calculated using the formula

$$f(x) = \frac{1}{\sqrt{2\pi}} exp(\frac{(x-\mu)^2}{2\sigma^2})$$

To make others calculus and depending on the class of the object, one might need other functions linked to the cdf and the pdf. For example, the method `cdf_inverse` will compute the inverse of the cdf. `distrobject.cdf_inverse(x)` will return the value $F^{-1}(x)$.

3.1.2 mean, var and other parameters

One can see that these formulas depend on parameters. In the previous normal case, they are μ and σ.

Althought the mean and the variance can be interesting values that we calculate using the cdf or the pdf. They also can be considered as parameters. In the normal case, μ is also the mean and σ^2 the variance. If the object `distrobject` represents a random variable X, `distrobject.mean` will return the value of $\mathbb{E}(x)$ and `distrobject.var` will return the value of $Var(x)$.

Depending on the class of the object, it could have other parameters that would be explained in detail while presenting the different classes. For instance, we can quote the degree of freedom (`df`) for student distributions, the covariance matrix (`cov`) for multivariate distribution or the theta parameters (`theta`) for archimedian copulas.

3.1.3 Initializing Distributions

The method __init__ creates a new object of a distribution class. It will be called with the specific parameters fo the distribution. For instance, $distrobject = UnivariateNormalDistribution(mean = mu, var = sigma**2)$ will return an Normal distribution object with the parameters `mu` for the mean and `sigma**2` for the variance. But, in practice, it is very useful to fit a distribution to a data set. This can be done by calling the `fit` method of any distribution class with the fit method. This method will estimate the parameters for the distribution from the input data. For instance, `UnivariateNormalDistribution.fit(data = Y)` will return a normal distribution with the mean and variance fit to the data (in this case, these parameters are just taken to be the sample mean and sample variance of the data).

3.1.4 `generates_X` and sample

A distribution object modelizing a random variable X also has a method that can generates realisations of X following the probability law.

For instance, in the normal case we call the numpy function `numpy.random.randn` with the good parameters to generates such variables. In the case of univariate distribution it can also be obtained by generating a uniform variable in [0,1] and composing by the inverse of the cdf. $F^{-1}(U)$, where U is uniformly distributed on [0,1], follows the same probability law as X.

Because sometimes generating a lot of variables can be long, they are stored automatically in `distrobject.X`. Then, `distrobject.generates_X(n)` returns a vector of n independent realisations of X and store them in the attribute `mydistrobject.X` by appending this n new values by appending it to the old attribute.

3.2 Files and classes

In order to make comparisons and find the best distribution model for a variable, I implemented several different distributions. Even if they follow the same global pattern described above, their parameters, their functions and the way they are implemented can be very different because of a need to speed up the computation or simply because the mathetical object are really different. All the distributions are coded with classes that one can find in a certain file. I explain in this section how the files are organized and how the classes work. At the beginning of each file's subsection, one can see the whole list of the class it contains.

3.2.1 base_distribution.py

This file contains various abstract classes that define interfaces and common functions for distributions :

```
BaseDistribution
UnivariateDistribution
MultivariateDistribution
```

`BaseDistribution` is the base class from which all the distributions classes will inherit. This class describes at a minimum, what all distributions must have. For our purposes, we expect all distributions to define a `pdf` method and a `fit` method.

`UnivariateDistribution` inherits from `BaseDistribution`. It serves as the base class for all univariate distributions. This exports basic methods for computing the cdf, the inverse cdf, and the expectation over a specific region.

`MultivariateDistribution` inherits from `BaseDistribution`. It is the abstract class from which all the distributions with more than one variable will inherit. The particularity of multivariate distribution is that it works with dictionnaries. This permits not to confuse the different coordinates of the distribution. To this end, any distribution can be constructed or fit by passing by the keyword `dimkeys` a list of strings which will serve as names for the dimensions. Then one can call a function which expects a vector as input (say the pdf) with the a dictionary mapping dimension names to the values. This abstract class also contains the method `rect_prob` that computes the probability for the random variable to be in a n dimension rectangle defined by two opposed points `lowerdict` and `upperdict`. Even thaugh this calculus requires a non trivial recursion, it only needs the cdf to be computed and it is the same calculus for all multivariate distribution.

3.2.2 distributions.py

This file contains all the concrete classes of the classical distributions that are not built with copulas. If they are called Univariate, they directly inherit from `BaseDistribution`, if they are called Multi they inherit form `MultivariateDistribution` :

```
UnivariateBasicEmpiricalDistribution
UnivariateEmpiricalDistribution
UnivariateEpiSplineDistribution
UnivariateUniformDistribution
UnivariateNormalDistribution
UnivariateStudentDistribution
MultiNormalDistribution
MultiStudentDistribution
```

`UnivariateBasicEmpiricalDistribution` is the simplest distribution you can build without making an hypothesis on the model. It assumes that all the results in the `input_data` are the only possible ones and that they are equiprobable. This gives us a method to generate a sample, we simply pick randomly one element of `input_data`. If we note $Y = (Y_1, ..., Y_n) =$ `input_data`, the functions would be :

$$F(x) = \frac{1}{n}\sum_{k=1}^{n} 1_{Y_k \leq x}$$

$$\mathbb{P}(X = x) = \frac{1}{n}\sum_{k=1}^{n} 1_{Y_k = x}$$

One can notice that F is not continuous, that implies that f is not well defined.

`UnivariateEmpiricalDistribution` works almost like `UnivariateBasicEmpirical` but avoids the problem of continuity by cleverly interpolating lines between the points we have. When the set of data is big enough it gives the same result than `UnivariateBasicEmpricalDistribution`.

`UnivariateEpiSplineDistribution` is a more clever object obtained thanks to the resolution of an optimization problem where the cdf as a particular shape :

$$F(x) = e^{-g(x)}, \text{ where } g \text{ is a piecewise defined polynomial}$$

For example, one can choose the number N which indicates how much subsegments we want to divide our initial segment to create each piece of the polynomial function. Plus, this class needs the installation of the language pyomo and of the solver ipopt. To learn more about these installation, check `GOSM` user manual. This class is unusual because it was extracted from an earlier version of software called `Prescient`.

`UnivariateUniformDistribution` contains only two parameters called `a` and `b` who represent respectively the minimum and the maximum of the distribution. When an object of this class is called with `input_data`, the `__init__` method, assign the minimum of `input_data` to a and its maximum to b.

$$a = \texttt{a} = \texttt{min(input\_data)}$$

$$b = \texttt{b} = \texttt{max(input\_data)}$$

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{if } x \in [a,b] \\ 0 & \text{else} \end{cases}$$

$$F(x) = \begin{cases} 0 & \text{if } x \leq a \\ 1 & \text{if } x \geq b \\ \frac{x-a}{b-a} & \text{else} \end{cases}$$

`UnivariateNormalDistribution` has two parameters `mean` and `var` which are stored as attributes. We present below the main formulas that define this class. Each time we represent parameters with their names in the python code and with its usual mathematical notation so that the formulas are not too hard to read.

$$\mu = \texttt{mean} = \texttt{mean(input\_data)}$$

$$\sigma^2 = \texttt{var} = \texttt{var(input\_data)}$$

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{\sigma^2}}$$

$$F(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\mu)^2}{\sigma^2}} dy$$

`UnivariateStudentDistribution`

$$\mu = \texttt{mean} = \texttt{mean(input\_data)}$$

$$\nu = \texttt{df} = \texttt{2var(input\_data)/(var(input\_data)-1)}$$

$$f(x) = \frac{1}{\sqrt{\nu\pi}} \frac{\Gamma(\frac{\nu+1}{2})}{\Gamma(\frac{\nu}{2})} \left(1 + \frac{(x-\mu)^2}{\nu}\right)^{-\frac{\nu+1}{2}}$$

$$F(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{\nu\pi}} \frac{\Gamma(\frac{\nu+1}{2})}{\Gamma(\frac{\nu}{2})} \left(1 + \frac{(y-\mu)^2}{\nu}\right)^{-\frac{\nu+1}{2}} dy$$

`MultiNormalDistribution`

$$\mu = \texttt{mean} = \texttt{mean(input\_data)}$$

$$K = \texttt{cov} = \texttt{cov(input\_data)}$$

$$f(x) = \frac{1}{(2\pi)^{\frac{d}{2}}\sqrt{det(K)}} e^{-\frac{(x-\mu)^\top K^{-1}(x-\mu)x}{2}}$$

$$F(x) = \int_{-\infty}^{x} \frac{1}{(2\pi)^{\frac{d}{2}}\sqrt{det(K)}} e^{-\frac{(y-\mu)^\top K^{-1}(y-\mu)}{2}} dy$$

`MultiStudentDistribution`

$$\mu = \texttt{mean} = \texttt{mean(input\_data)}$$

$$\nu = \texttt{df}$$

$$K = \texttt{cov} = \texttt{cov(input\_data)}$$

$$f(x) = \frac{\Gamma(\frac{\nu+d}{2})}{\Gamma(\frac{\nu}{2})(\nu\pi)^{\frac{d}{2}}\sqrt{det(K)}} \left(1 + \frac{1}{\nu}(x-\mu)^\top K^{-1}(x-\mu)\right)^{-\frac{\nu+d}{2}}$$

$$F(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{\nu\pi}} \frac{\Gamma(\frac{\nu+1}{2})}{\Gamma(\frac{\nu}{2})} \left(1 + \frac{(y-\mu)^2}{\nu}\right)^{-\frac{\nu+1}{2}} dy$$

3.2.3 copula.py

This file contains the abstract class `CopulaBase` which inherits from `MultivariateDistribution` and from which all the concrete copula classes inherit. The mathematical details of copulas are explained in the next sections. As was the case with our distribution objects and classes in python, our copula objects and classes in python are different from the strict mathematical definition of a copula which is just what we call a C function. The inheritance from `MultivariateDistribution` is very convenient in our applications, but is non-standard for copula classes.

```
CopulaBase
GaussianCopula
StudentCopula
FrankCopula
ClaytonCopula
GumbelCopula
WeigthedCombinedCopula
IndependenceCopula
EmpiricalCopula
```

`CopulaBase` is the abstract class that contains the attributes and methods that all the copula classes have in common. The method is to use what we presented as property 1 on each diagonal. Instead of working on the dependence between X_1, X_2 ... and X_n, we prefer to focus on the dependence between $U_1 = F_1(X_1)$, $U_2 = F_2(X_2)$ ... and $U_n = F_n(X_n)$. It is now much easier to compare the dependence between coordonates that all have the same distribution which is uniform in [0,1]. We work in what we call the U-space, $[0,1]^d$, as opposed to the X-space, $\mathbb{R}^d$. One can easily pass from one to the other by composing coordonate by coordonate with the F_i or by using the formulas presented in the next sections. Like all the children of `MultivariateDistribution` a copula object has methods and attributes in the X-space, but it contains also its twins in the U-space, just like if a copula object contained the distribution of X and the distribution of U. Considering this analogy, C (as the cdf of U) plays in the U-space the role of cdf in the X-space. c (as the pdf of U) plays in the U-space the role of pdf in the X-space. Finally, generates_U permits to return samples of U with the good dependence, it plays in the U-space the same role as generates_X in the X-space.

3.2.4 vine.py

Vine copulas are copulas that are built using other bivariate copulas. I explain them with more details in the next section.

```
CVineCopula
DVineCopula
```

3.2.5 distribution_factory.py

In all the previous files, each definition of a new concrete class begins with `@register_distribution(name="name-of-the-class")`. This permits to give string names to the class. The file distribution_factory.py contains the code that allow us to write :

```
distr_class = distribution_factory(copula_string)
distr_object = distr_class(input)
```

instead of

```
if copula_string =="univariate-uniform":
   distr_object = UnivariateUniformDistribution(input)
if copula_string =="univariate-normal":
   distr_object = UnivariateNormalDistribution(input)
```

etc, for all the cases.

This pratical way of selecting the classes is also used to build distributions that are created thanks to other ones such as combined copulas or vine copulas.

3.2.6 tester.py

This file contains several unittests to check the distribution classes attempt. Some are just quick tests to check if the code runs. But most of them attempt to verify if the implemented formulas are correct. In order to do that,the method is trying to obtain the same result by different ways or by making a loop on diagram that should be commutative like the one in figure 1. For instance, `test_pdf_cdf` integrates the pdf to compare it to the cdf and differentiate the cdf to verify if it is equal to the pdf. `test_c_with_C_2_dim` integrates twice `c` and verify if the value is equal to `C`. `test_C_with_sample` checks the low loop of diagram in figure 1 in the U-space : with a big data set created by `generates_U`, it creates an empirical cdf of U (called `C_from_sample`) and compares it to the actual cdf of U which is `C`. Another example is `test_with_gaussian_copula_3_dim` which checks if making a distribution with normal (or gaussian) marginals and a gaussian copula give the same result than a creating directly a multinormal (or multigaussian) distribution.

```
MultiNormalDistributionTester
UnivariateNormalDistributionTester
MultiStudentDistributionTester
UnivariateStudentDistributionTester
CopulaTester
VineCopulaTester
```

4 Copulas properties

In order to study the dependance of several correlated random variables, we need to focus on a tool called copula. It is more practical to consider our different variables together as a multidimensional variable. The distribution univariate random variables are then called marginals. Copulas permit measurement of the dependance without caring about the marginals. In our object oriented program, it is really practical because one can work on the dependance with the copula while the marginals are just input variables. To introduce and define the copulas, let us quote Sklar's theorem :

Theorem 1. *Sklar, 1959. Let us consider a random vector $X = (X_1, ..., X_d)$ with a cumulative distibution function $F(x_1, ..., x_d) = \mathbb{P}(X_1 \leq x_1, .., X_d \leq x_d)$. Then, exists a function C called copula $C : [0,1]^d \to [0,1]$ so that :*

$$F(x_1, ...x_d) = C(F_1(x_1), ..., F_d(x_d))$$

where F_i is the cumulative density function of the ith marginal.
The copula C is unique if the marginals are continuous.

We will also use the density of the copula :

Definition 5. *The copula density of a copula C is the function $c : [0,1]^d \to \mathbb{R}^+$:*

$$c(u,v) = \frac{\partial^d C}{\partial u_1 ... \partial u_d}(u,v)$$

By differentiating, we obtain the equation :

$$f(x_1, ..., x_d) = c(F_1(x_1), ..., F_d(x_d))f_1(x_1)...f_d(x_d)$$

Definition 6. *The Kendall distribution function* $\mathcal{K}_F : [0,1] \to [0,1]$ *of a distribution of cumulative density function F is :*

$$\mathcal{K}_F(u) = \mathbb{P}(F(X) \leq u)$$

where X is a random variable following the distribution defined by F.

$$\begin{aligned}
\mathcal{K}_F(u) &= \mathbb{P}(F(X) \leq u) \\
&= \int_{\mathbb{R}^d} 1_{F(x)\leq u} f(x)dx \\
&= \int_{\mathbb{R}^d} 1_{C(F_1(x_1),...,F_d(x_d))\leq u} c(F_1(x_1), ..., F_d(x_d))f_1(x_1)...f_d(x_d)dx \\
&= \int_{[0,1]^d} 1_{C(y)\leq u} c(y)dy \\
&= \int_{K_u} c(y)dy \text{ , where } K_u = \{y \in [0,1]^d, C(y) \leq u\} \\
&= \mathbb{P}(C(U) \leq u)
\end{aligned}$$

Where U is a random variable on $[0,1]^d$ with cdf C (and uniform marginals).

Intuitively, when we do the changing of variable $y_i = F_i(x_i)$:
We have $dy_i = F_i'(x_i)dx_i = f_i(x_i)dx_i$. So, indeed $dy = f_1(x_1)...f_d(x_d)dx$.
The meticulous proof with the Jacobian determinant gives the same result.

$\mathcal{K}_F$ only depends on the copula C of the distribution function.
Thus, we will use the notation $\mathcal{K}_C$.
If we write $\Gamma_{u_2,...,u_d}(x) = C(x, u_2, ..., u_d)$, we also have :

$$\mathcal{K}_C(u) = \int_0^1 ... \int_0^1 \left(\int_0^{\Gamma^{-1}_{y_2,...,y_d}(u)} c(y)dy_1 \right) dy_2...dy_d$$

We want to use Vine copulas to study multidimensional dependance. To compute such vine copulas, we need to have the partial derivative of C function and its inverse. For each of the following copulas in 2 dimensions, I calculated the partial derivate $h(u, v, \theta) = \frac{\partial C_\theta}{\partial v}(u, v)$. I also calculated its inverse h^{-1} considering its first variable : $h(h^{-1}(u, v, \theta), v, \theta) = u$

$$\begin{tikzcd}
 & C \arrow[dl, "\frac{\partial}{\partial u_1}"'] \arrow[dr, "\frac{\partial^d}{\partial u_1 ... \partial u_d}"] & \\
\frac{\partial C}{\partial u_1} \arrow[rr, bend right, "\frac{\partial^{d-1}}{\partial u_2 ... \partial u_d}"'] & & c \arrow[ll, bend right, "\int ... \int"']
\end{tikzcd}$$

Figure 2: Relations between copula functions

5 Elliptic Copulas

Explain that they are copulas of multidimensional distributed vector with different law. Add the difference between correlation matrix and covariance matrix and say we only consider 1 on the diagonal.

5.1 Gaussian Copula

The gaussian copula of correlation matrix K is the copula of a gaussian vector $\mathcal{N}(0, K)$.

Because one can change the variance of each marginal, we will only consider covariance matrix where there are only 1 in the diagonal.

We will note $\mathcal{N}(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} dt$, the cumulative density function of the standard normal distribution and $\mathcal{N}^{-1}$ its inverse.

5.1.1 C function in dimension n

$$C_K(u_1, ...u_d) = \int_{-\infty}^{\mathcal{N}^{-1}(u_1)} ... \int_{-\infty}^{\mathcal{N}^{-1}(u_d)} \frac{1}{(2\pi)^{\frac{d}{2}} \sqrt{det(K)}} e^{-\frac{x^\top K^{-1} x}{2}} dx$$

With a change of variable, we have also

$$C_K(u_1, ...u_d) = \int_{0}^{u_1} ... \int_{0}^{u_d} \frac{1}{\sqrt{det(K)}} e^{-\frac{\mathcal{N}^{-1}(x)^\top (K^{-1} - I_d) \mathcal{N}^{-1}(x)}{2}} dx$$

where $\mathcal{N}^{-1}(x) = \begin{pmatrix} \mathcal{N}^{-1}(x_1) \\ \vdots \\ \mathcal{N}^{-1}(x_d) \end{pmatrix}$

5.1.2 C function in dimension 2

In dimension 2, all matrix of correlation can be written $K = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$. So we will only consider one parameter ρ.

$$C_\rho(u,v) = \int_{-\infty}^{\mathcal{N}^{-1}(u)} \int_{-\infty}^{\mathcal{N}^{-1}(v)} \frac{1}{2\pi\sqrt{1-\rho^2}} e^{-\frac{x^2-2\rho xy+y^2}{2(1-\rho^2)}} dxdy$$

With the same change of variable, we have also
$C_\rho(u,v) = \int_0^u \int_0^v \frac{1}{\sqrt{1-\rho^2}} e^{-\frac{\rho^2\mathcal{N}^{-1}(x)^2-2\rho\mathcal{N}^{-1}(x)\mathcal{N}^{-1}(y)+\rho^2\mathcal{N}^{-1}(y)^2}{2(1-\rho^2)}} dxdy$

5.1.3 Partial derivative of C

If we derivate the second formula, we have
$h(u,v,\rho) = \frac{\partial C_\rho}{\partial v}(u,v) = \int_0^u \frac{1}{\sqrt{1-\rho^2}} e^{-\frac{\rho^2\mathcal{N}^{-1}(x)^2-2\rho\mathcal{N}^{-1}(x)\mathcal{N}^{-1}(v)+\rho^2\mathcal{N}^{-1}(v)^2}{2(1-\rho^2)}} dx$

If we derivate the first formula above, we obtain
$h(u,v,\rho) = \frac{\partial C_\rho}{\partial v}(u,v) = \frac{1}{\mathcal{N}'(\mathcal{N}^{-1}(v))} \int_{-\infty}^{\mathcal{N}^{-1}(u)} \frac{1}{2\pi\sqrt{1-\rho^2}} e^{-\frac{x^2-2\rho x\mathcal{N}^{-1}(v)+\mathcal{N}^{-1}(v)^2}{2(1-\rho^2)}} dx$

$h(u,v,\rho) = \int_{-\infty}^{\mathcal{N}^{-1}(u)} \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-\rho\mathcal{N}^{-1}(v))^2}{2(1-\rho^2)}} dx$

$h(u,v,\rho) = \int_{-\infty}^{\frac{\mathcal{N}^{-1}(u)-\rho\mathcal{N}^{-1}(v)}{\sqrt{1-\rho^2}}} \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}dy}$

$h(u,v,\rho) = \mathcal{N}(\frac{\mathcal{N}^{-1}(u)-\rho\mathcal{N}^{-1}(v)}{\sqrt{1-\rho^2}})$

If we inverse this function, we obtain

$h^{-1}(u,v,\rho) = \mathcal{N}(\sqrt{1-\rho^2}\mathcal{N}^{-1}(u)+\rho\mathcal{N}^{-1}(v))$

5.1.4 c function in dimension n

$$c_K(u_1,...,u_d) = \frac{1}{\sqrt{det(K)}} e^{-\frac{\mathcal{N}^{-1}(u)^\top(K^{-1}-I_d)\mathcal{N}^{-1}(u)}{2}}$$

where $\mathcal{N}^{-1}(u) = \begin{pmatrix} \mathcal{N}^{-1}(u_1) \\ \vdots \\ \mathcal{N}^{-1}(u_d) \end{pmatrix}$

5.1.5 c function in dimension 2

$c_\theta(u,v) = \frac{1}{\sqrt{1-\rho^2}} e^{-\frac{\rho^2(\mathcal{N}^{-1}(u)^2+\mathcal{N}^{-1}(v)^2)-2\rho\mathcal{N}^{-1}(u)\mathcal{N}^{-1}(v)}{2(1-\rho^2)}}$

5.2 Student Copula

The student copula of correlation matrix K and of degree of freedom ν is the copula of a student random vector $t_\nu(0, K)$.

Because one can change the variance of each marginal, we will only consider covariance matrix where there are only 1 in the diagonal.

We will note $t_\nu(x) = \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi}\Gamma(\frac{\nu}{2})} \int_{-\infty}^{x} \left(1 + \frac{t^2}{\nu}\right)^{-\frac{\nu+1}{2}} dt$, the cumulative density function of the student distribution with degree of freedom ν and t_ν^{-1} its inverse.
We could write both with the regularized incomplete Beta function
$I_{a,b}(x) = \frac{1}{B(a,b)} \int_0^x t^{a-1}(1-t)^{b-1} dt$
$= \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \int_0^x t^{a-1}(1-t)^{b-1} dt$

and its inverse $I_{a,b}^{-1}$:

$$t_\nu(x) = \begin{cases} I_{\frac{\nu}{2},\frac{1}{2}}(\frac{\nu}{x^2+\nu}) & \text{if } x \leq 0 \\ 1 - I_{\frac{\nu}{2},\frac{1}{2}}(\frac{\nu}{x^2+\nu}) & \text{if } x \geq 0 \end{cases}$$

and

$$t_\nu^{-1}(x) = \begin{cases} -\sqrt{\nu(\frac{1}{I^{-1}_{\frac{\nu}{2},\frac{1}{2}}(2y)} - 1)} & \text{if } x \leq \frac{1}{2} \\ \sqrt{\nu(\frac{1}{I^{-1}_{\frac{\nu}{2},\frac{1}{2}}(2(1-y))} - 1)} & \text{if } x \geq \frac{1}{2} \end{cases}$$

5.2.1 Generates X

To generates our U vector in the $[0,1]^d$ space, we need to first generates a X vector in the $\mathbb{R}^d$ space following a Multivariate Student Distribution. Thus, we use the equality in law :

$$\sqrt{\frac{\nu}{S}} Z \sim t_\nu(0, K)$$

where $s \sim \chi^2_\nu$ and $Z \sim \mathcal{N}(0, K)$

5.2.2 C function in dimension n

$$C_K(u_1, ...u_d) = \int_{-\infty}^{t_\nu^{-1}(u_1)} ... \int_{-\infty}^{t_\nu^{-1}(u_d)} \frac{\Gamma(\frac{\nu+d}{2})}{\Gamma(\frac{\nu}{2})(\nu\pi)^{\frac{d}{2}}\sqrt{det(K)}} \left(1 + \frac{1}{\nu} x^\top K^{-1} x\right)^{-\frac{\nu+d}{2}} dx$$

With a change of variable, we have also

$$C_K(u_1, ...u_d) = \int_0^{u_1} ... \int_0^{u_d} \frac{\Gamma(\frac{\nu+d}{2})\Gamma(\frac{\nu}{2})^{d-1}}{\sqrt{det(K)}\Gamma(\frac{\nu+1}{2})^d \Pi_{j=1}^d 1 + \frac{t_\nu^{-1}(x_j)^2}{2})^{-\frac{\nu+1}{2}}} \left(1 + \frac{1}{\nu} t_\nu^{-1}(x)^\top K^{-1} t_\nu^{-1}(x)\right)^{-\frac{\nu+d}{2}} dx$$

where $t_\nu^{-1}(x) = \begin{pmatrix} t_\nu^{-1}(x_1) \\ \vdots \\ t_\nu^{-1}(x_d) \end{pmatrix}$

5.2.3 C function in dimension 2

In dimension 2, all matrix of correlation can be written $K = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$. So we will only consider one parameter ρ.

$$C_{\rho,\nu}(u,v) = \int_{-\infty}^{t_\nu^{-1}(u)} \int_{-\infty}^{t_\nu^{-1}(v)} \frac{1}{2\pi\sqrt{1-\rho^2}} \left(1 + \frac{x^2 - \rho xy + y^2}{\nu(1-\rho^2)}\right)^{-\frac{\nu+2}{2}} dxdy$$

5.2.4 Partial derivative of C

If we derivate the formula above, we obtain (cf Pair copula constructions of muliple dependance) :
$h(u,v,\rho,\nu) = \frac{\partial C_{\rho,\nu}}{\partial v}(u,v) = t_\nu\left(\frac{t_\nu^{-1}(u) - \rho t_\nu^{-1}(v)}{\sqrt{\frac{(\nu + t_\nu^{-1}(v)^2)(1-\rho^2)}{\nu+1}}}\right)$

If we inverse this function, we obtain

$h^{-1}(u,v,\rho,\nu) = t_\nu(\sqrt{\frac{(\nu + t_\nu^{-1}(v)^2)(1-\rho)}{\nu+1}} t_\nu^{-1}(u) + \rho t_\nu^{-1}(v))$

5.2.5 c function in dimension n

$c_{K,\nu}(u_1, ..., u_d) = \dfrac{\Gamma(\frac{\nu+d}{2})\Gamma(\frac{\nu}{2})^{d-1}}{\sqrt{det(K)}\Gamma(\frac{\nu+1}{2})^d \Pi_{j=1}^d (1 + \frac{t_\nu^{-1}(u_j)^2}{2})^{-\frac{\nu+1}{2}}} \left(1 + \frac{1}{\nu} t_\nu^{-1}(u)^\top K^{-1} t_\nu^{-1}(u)\right)^{-\frac{\nu+d}{2}}$

where $t_\nu^{-1}(u) = \begin{pmatrix} t_\nu^{-1}(u_1) \\ \vdots \\ t_\nu^{-1}(u_d) \end{pmatrix}$

5.2.6 c function in dimension 2

$c_\theta(u,v) = \dfrac{\Gamma(\frac{\nu}{2})\Gamma(\frac{\nu+2}{2})}{\Gamma(\frac{\nu+1}{2})^2 (1 + \frac{t_\nu^{-1}(u)^2}{\nu})^{-\frac{\nu+1}{2}} (1 + \frac{t_\nu^{-1}(v)^2}{\nu})^{-\frac{\nu+1}{2}} \sqrt{1-\rho^2}} \left(1 + \frac{t_\nu^{-1}(u)^2 + t_\nu^{-1}(v)^2 - 2\rho t_\nu^{-1}(u) t_\nu^{-1}(v)}{\nu(1-\rho^2)}\right)^{-\frac{\nu+}{2}}$

6 Archimedean Copulas

Archimedean copulas take the form :

$C(u_1, ..., u_d) = \Phi(\Phi^{-1}(u_1) + ... + \Phi^{-1}(u_d))$

where Φ is the Laplace transform of a positive nonzero random variable Y : $\Phi(u) = \mathbb{E}(e^{-uY})$

Φ is called the generator function.
It generally depends on a parameter called θ.
In some papers, the definition of the generator function and its inverse can be inverted. We will use the definition above which gives us simpler formulas.

All of the next formulas can be expressed with the generator function.
For example, if we refer to [1], we can write the Kendall distribution function thanks to the generator function :

$$\mathcal{K}_C(x) = \begin{cases} \frac{(-1)^{d-1}(\Phi^{-1}(0))^{d-1}}{(d-1)!}\Phi_-^{(d-1)}(\Phi^{-1}(0)) & \text{if } x = 0 \\ \sum_{k=0}^{d-2} \frac{(-1)^k(\Phi^{-1}(x))^k\Phi^{(k)}(\Phi^{-1}(x))}{k!} + \frac{(-1)^{d-1}(\Phi^{-1}(x))^{d-1}\Phi_-^{(d-1)}(\Phi^{-1}(x))}{(d-1)!} & \text{if } x \in]0,1] \end{cases}$$

6.1 Frank Copula

6.1.1 Generator, generator inverse and derivatives

$\theta \in \mathbb{R}\backslash\{0\}$

$\Phi_\theta(t) = -\frac{1}{\theta}log(1 + e^{-t}(e^{-\theta} - 1))$

$\Phi_\theta^{-1}(t) = -log(\frac{e^{-\theta t}-1}{e^{-\theta}-1})$

$\Phi'_\theta(t) = \frac{e^{-t}}{\theta(\frac{1}{e^{-\theta}-1}+e^{-t})}$

$\Phi''_\theta(t) = \frac{-e^{-t}}{\theta(e^{-\theta}-1)(\frac{1}{e^{-\theta}-1}+e^{-t})^2}$

6.1.2 C function in dimension 2

$C_\theta(u, v) = -\frac{1}{\theta}log(1 + \frac{(e^{\theta u}-1)(e^{\theta v}-1)}{e^{-\theta}-1})$

6.1.3 Partial derivative of C and inverse

$h(u, v, \theta) = \frac{\partial C_\theta}{\partial v}(u, v) = \frac{e^{\theta u}-1}{e^{\theta u}+e^{\theta v}-e^{\theta(u+v-1)}-1}$

We first solve $\frac{aX+b}{cX+d} = y$, where $a = 1, b = -1, c = 1 - e^{\theta(v-1)}, d = e^{\theta v} - 1$

We have then $X = \frac{dy-b}{a-cy}$

We solve $e^{\theta u} = X : u = \frac{1}{\theta}log(X)$

So, we now have
$h^{-1}(u, v, \theta) = \frac{1}{\theta}log(\frac{(e^{\theta v}-1)u+1}{1-(1-e^{\theta(v-1)})u})$

6.1.4 c function in dimension 2

$c_\theta(u,v) = \frac{\theta e^{\theta(u+v)}(1-e^{-\theta})}{(e^{\theta u}+e^{\theta v}-e^{\theta(u+v-1)}-1)^2}$

6.2 Gumbel Copula

6.2.1 Generator and generator inverse

$\theta \in [1, \infty[$

$\Phi_\theta(t) = e^{-t^{\frac{1}{\theta}}}$
$\Phi_\theta^{-1}(t) = (-log(t))^\theta$
$\Phi_\theta'(t) = -\frac{1}{\theta} t^{\frac{1-\theta}{\theta}} e^{-t^{\frac{1}{\theta}}}$
$\Phi_\theta''(t) = ((-\frac{1}{\theta} t^{\frac{1-\theta}{\theta}})^2 - \frac{1-\theta}{\theta^2} t^{\frac{1-2\theta}{\theta}}) e^{-t^{\frac{1}{\theta}}}$

6.2.2 C function in dimension 2

$C_\theta(u,v) = e^{-((-log(u))^\theta + (-log(v))^\theta)^{\frac{1}{\theta}}}$

6.2.3 Partial derivative of C and inverse

$h(u,v,\theta) = \frac{\partial C_\theta}{\partial v}(u,v) = \frac{1}{v}(-log(v))^{\theta-1}((-log(u))^\theta + (-log(v))^\theta)^{\frac{1}{\theta}-1} e^{-((-log(u)^\theta + (-log(v))^\theta)^{\frac{1}{\theta}}}$

We have to introduce the lambert W function which is the inverse of $x \rightarrow xe^x$:

$W(x)e^{W(x)} = x$

For $x \geq 0, ye^y = x$ has only one solution y.
So, there is no ambiguity to define $W(x)$ if $x \geq 0$.

We first solve $\lambda X^\alpha e^{-X} = y$ where $\lambda = \frac{(-log(v))^{\theta-1}}{v}, \alpha = 1 - \theta$

We obtain $X = \alpha W(\frac{1}{\alpha}(\frac{y}{\lambda})^{\frac{-1}{\theta-1}})$

Then, we solve $X = ((-log(u))^\theta + (-log(v))^\theta)^{\frac{1}{\theta}}$

which gives us $u = e^{-(X^\theta - (-log(v))^\theta)^{\frac{1}{\theta}}}$

Finally, we have $h^{-1}(u,v,\theta) = e^{-(((\theta-1)W(\frac{-log(v)}{\theta-1}(vu)^{\frac{-1}{\theta-1}}))^\theta - (-log(v))^\theta)^{\frac{1}{\theta}}}$

6.2.4 c function in dimension 2

$c_\theta(u,v) = \frac{C_\theta(u,v)}{uv}((-log(u)^\theta + (-log(v))^\theta)^{-2+\frac{2}{\theta}}[1+(\theta-1)((-log(u)^\theta + (-log(v))^\theta)^{-\frac{1}{\theta}}]$

6.3 Clayton Copula

6.3.1 Generator, generator inverse and derivatives

$\theta \in [-1, \infty[\backslash\{0\}$

$\Phi_\theta(t) = (1+\theta t)_+^{-\frac{1}{\theta}}$
$\Phi_\theta^{-1}(t) = \frac{1}{\theta}(t^{-\theta} - 1)$

$\Phi_\theta^{(n)}(t) = (-1)^n (1+\theta t)^{-\frac{1+n\theta}{\theta}} \prod_{k=0}^{n-1}(1+k\theta)$

6.3.2 C function in dimension 2

$C_\theta(u,v) = (max\{u^{-\theta} + v^{-\theta} - 1; 0\})^{-\frac{1}{\theta}}$

6.3.3 Partial derivative of C and inverse

$$h(u,v,\theta) = \frac{\partial C_\theta}{\partial v}(u,v) = \begin{cases} v^{-\theta-1}(u^{-\theta} + v^{-\theta} - 1)^{-\frac{1}{\theta}-1} & \text{if } u^{-\theta} + v^{-\theta} \geq 1 \\ 0 & \text{else} \end{cases}$$

$h^{-1}(u,v,\theta) = (u^{-\frac{\theta}{\theta+1}} v^{-\theta} - v^{\theta} + 1)^{-\frac{1}{\theta}}$ where u must be nonzero.

6.3.4 c function in dimension 2

$c_\theta(u,v) = (1+\theta)(uv)^{-1-\theta}(u^{-\theta} + v^{-\theta} - 1)^{-\frac{1}{\theta}-2}$

7 Empirical Copula

Having a sample of multivariate random variables, we can define the finite empirical distribution of this sample where each realisation of the sample is equiprobable. The empirical copula is the copula of this empirical distribution. Contrary to the other ones, the empirical copula depends on its marginals.

$$C(u_1, ..., u_d) = \sum_{k=1}^{n} 1_{\forall i, F_i(x_k) \leq u_i}$$

where, F_i is the cdf of the marginal i.

8 Vine Copulas

8.1 Canonical Vine Copula

8.1.1 Global probability density function

$$f(x_1, ..., x_d) = \prod_{k=1}^{d} f(x_k) \prod_{j=1}^{d-1} \prod_{i=1}^{d-j} c_{j,j+i|1,...,j-1}(F(x_j|x_1, ..., x_{j-1}), F(x_{j+i}|x_1, ..., x_{j-1}))$$

8.1.2 c function in dimension n

By identifying c in the previous equation, we have :

$$c_{1,...,d}(F_1(x_1),...,F_d(x_d)) = \prod_{j=1}^{d-1}\prod_{i=1}^{d-j} c_{j,j+i|1,...,j-1}(F(x_j|x_1,...,x_{j-1}), F(x_{j+i}|x_1,...,x_{j-1}))$$

We now do an approximation of conditional independance :
$c_{j,j+i|1,...,j-1} = 1$ when $(1,...,j-1) \neq \emptyset \Leftrightarrow j \neq 1$

We obtain :

$$c_{1,...,d}(F_1(x_1),...,F_d(x_d)) = \prod_{i=1}^{d-1} c_{1,i+1}(F_1(x_1), F_{i+1}(x_{i+1}))$$

Then,

$$c_{1,...,d}(u_1,...,u_d) = \prod_{i=1}^{d-1} c_{1,i+1}(u_1, u_d)$$

8.2 D Vine Copula

8.2.1 Global probability density function

$$f(x_1,...,x_d) = \prod_{k=1}^{d} f(x_k) \prod_{j=1}^{d-1}\prod_{i=1}^{d-j} c_{i,i+j|i+1,...,i+j-1}(F(x_i|x_{i+1},...,x_{i+j-1}), F(x_{i+j}|x_{i+1},...,x_{i+j-1}))$$

8.2.2 c function in dimension n

By identifying c in the previous equation, we have :

$$c_{1,...,d}(F_1(x_1),...,F_d(x_d)) = \prod_{j=1}^{d-1}\prod_{i=1}^{d-j} c_{i,i+j|i+1,...,i+j-1}(F(x_i|x_{i+1},...,x_{i+j-1}), F(x_{i+j}|x_{i+1},...,x_{i+j-1}))$$

We now do an approximation of conditional independance :
$c_{i,i+j|i+1,...,i+j-1} = 1$ when $(i+1,...,i+j-1) \neq \emptyset \Leftrightarrow j \neq 1$
We obtain :

$$c_{1,...,d}(F_1(x_1),...,F_d(x_d)) = \prod_{i=1}^{d-1} c_{i,i+1}(F_i(x_i), F_{i+1}(x_{i+1}))$$

Then,

$$c_{1,...,d}(u_1,...,u_d) = \prod_{i=1}^{d-1} c_{i,i+1}(u_i, u_{i+1})$$

References

[1] Alexander J. McNeil, Johanna Neslehova *Multivariate Archimedean Copulas, d-monotone Functions and l1-norm Symmetric Distributions*,Ann. Stat., 37:3059-3097, 2009.

[2] Kjersti Aas, Claudia Czado, Arnoldo Frigessi, Henrik Bakken, *Pair-copula construction of multiple dependence*, 2007.

[3] Kjersti Aas, *Modeling the dependance structure of financial assets : A survey of four copulas* 2004.

[4] Johanna F. Ziegel, Tilmann Gneiting, *Copula Calibration*, Electronic Journal of Statistics, 2014.