Shine of multiple dispatch: the `Copulas.jl` case.

When dependence structures modeling meets multiple dispatch

Oskar Laverny

2023-07-28

Table of contents i

Introduction on copulas

Definition

Definition (Copulas)

A copula, usually denoted C, is the distribution function of a random vector, supported on $[0,1]^d$, with $\mathcal{U}([0,1])$-distributed marginals.

Let $\boldsymbol{X} = (X_i, i \in 1, ..., d)$ be an (absolutely continuous) random vector in $\mathbb{R}^d$. Denote by $F_{\boldsymbol{X}} = \mathbb{P}(\boldsymbol{X} \leq \boldsymbol{x})$ and $F_{X_i}(x) = \mathbb{P}(X_i \leq x)$ the distributions functions of the random vector and of the marginals respectively. Then there is a known link between the two:

Theorem (Existance and uniqueness (see Sklar 1959))

For any absolutely continuous random vector with distribution function F, there exists a unique copula C such that

$$F_{\boldsymbol{X}}(\boldsymbol{x}) = C\left(F_i(x_i), i \in 1, ..., n\right).$$

A few comments

Remark (Division of labor)

The function C actually describes and contains the *dependence structure* of the whole random vector, apart from its marginals distributions.

Example (First examples)

Independance copula: $\Pi(\boldsymbol{u}) = \prod_{i=1}^{d} u_i$

Fréchet-Hoeffding minimum: $W(\boldsymbol{u}) = 1 + \langle \mathbf{1}, \boldsymbol{u} - \mathbf{1} \rangle$

Fréchet-Hoeffding maximum: $M(\boldsymbol{u}) = \min_i u_i$

Remark (Families)

As for univariate distributions, there exists a lot of better-or-lesser known parametric families of copulas.

Elliptical copulas

Definition (Elliptical random vector)

A random vector $\boldsymbol{X}$ is said to be Spherical if for every orthogonal matrix $\boldsymbol{A} \in \mathcal{O}_d(\mathbb{R})$, $\boldsymbol{AX} \sim \boldsymbol{X}$. Any linear transformation of $\boldsymbol{X}$ is then elliptical.

An elliptical copula is simply derived from an elliptical random vector by the Sklar theorem. There is no easier expression.

Example (Elliptical examples)

The Gaussian and Sttudent families of elliptical random vectors are two classical used parametric models. There is also the possibility to provide your own elliptical generator.

Examples

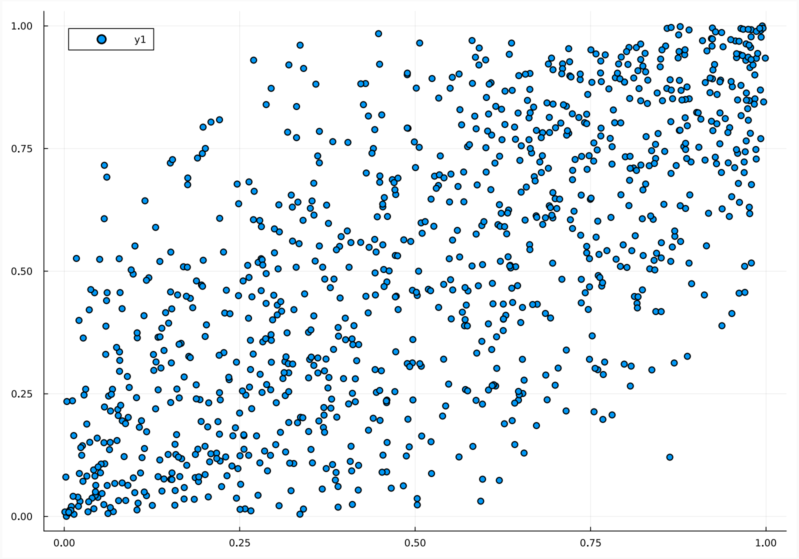

Figure 1: Sample from bivariate Gaussian Copula with sigma=0.7

Archimedean copulas

Definition (d-monotone functions)

A function $\varphi(t)$ is said d-monotone if it has $d-2$ derivatives which satisfy $(-1)^k\varphi^{(k)}(t) \geq 0$ and $(-1)^{d-2}\varphi^{(d-2)}$ is a non-increasing convex function.

Definition (Archimedean generator)

A d-archimedean generator is a d-monotone function from $\mathbb{R}_+$ to $[0,1]$ such that $\varphi(0) = 1$ and $\varphi(x) \to 0$ when $x \to \infty$.

Definition (Archimedean copula)

The function $C(\boldsymbol{u}) = \varphi\left(\sum_{i=1}^{d} \varphi(u_i)\right)$ is a copula if and only if φ is a d-archimedean generator.

Classical archimedean examples

Example (Classical parametric families)

$\varphi(t) = e^{-t}$ generates Π, the independence copula !
$\varphi(t) = (1 + t\theta)^{-\theta^{-1}}$ generates the `Clayton`(θ) copula.
$\varphi(t) = \exp\{-t^{\theta^{-1}}\}$ generates the `Gumbel`(θ) copula.
There are others : Franck, AMH, etc...

See (Nelsen 2006) for a comprehensive list of other notable generators.

Stochastic representations

Proposition (Radial-Simplex decomposition)

A d-variate random vector $\boldsymbol{U}$ *following an archimedean copula with generator* φ *can be decomposed into*

$$\boldsymbol{U}. = \varphi.(\boldsymbol{S}R),$$

where $\boldsymbol{S}$ *is uniformely distributed on the d-variate simplex, and* R *is a non-negative random variable, independant from* S*, defined as the (inverse) Williamson-d-transform of* φ*.*

Remark (Frailty reprensentation)

When φ is completely monotone, $W = 1/R$ is a non-negative random varaible that has φ as its Laplace transform.

See (Hofert, Mächler, and McNeil 2013), (McNeil 2008), and (McNeil and Nešlehová 2010) for details on these repesentations.

Exemples

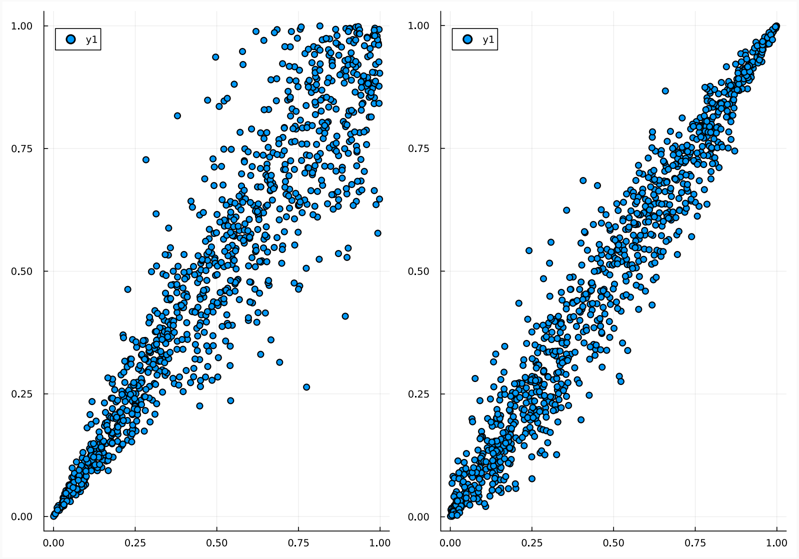

Figure 2: Sample from bivariate Clayton and Gumbel

Handling empirical data

Definition (Empirical copula : renormalized ranks.)

From a (n, d)-sized array $\boldsymbol{x}$, we can extract a (n, d)-sized array $\boldsymbol{u}$ corresponding to renormalized marginals ranks by:

$$u_{i,j} = \frac{\texttt{Rank}(x_{i,j} \texttt{ in } x_{\cdot,j})}{N+1}$$

Then the empirical copula of $\boldsymbol{x}$ is the `ecdf` of $\boldsymbol{u}$.

Smoothing possibilities: Bernstein Copula, Beta copula, checkerboard copula, etc. . . with a simple interface:

```
C = EmpiricalCopula(x,pseudos=false)
C = EmpiricalCopula(u,pseudos=true)
```

About the implementation

The SklarDist type and the fit function

As any distribution following Distributions.jl's standard, our code allows to fit Copula object, but also full models through SklarDist :

```julia
using Copulas, Distributions, Random
X1,X2,X3 = Gamma(2,3), Pareto(), LogNormal(0,1)
C = ClaytonCopula(3,0.7)
D = SklarDist(C,(X1,X2,X3))
simu = rand(D,1000)
est_D = fit(SklarDist{FrankCopula,Tuple{Gamma,Normal,LogNormal}}, simu)
# probably a bad fit..
```

From Distributions.jl's documentation: The fit function will choose a reasonable way to fit the distribution, which, in most cases, is maximum likelihood estimation.

About the implementation

A quick example with the Clayton implementation from the package:

```
struct ClaytonCopula{d,T} <: ArchimedeanCopula{d}
    theta::T
end
phi(C::ClaytonCopula, t) = (1+sign(C.theta)*t)^(-1/C.theta)
phi_inv(C::ClaytonCopula,t) = sign(C.theta)*(t^(-C.theta)-1)
tau(C::ClaytonCopula) = C.theta/(C.theta+2)
tau_inv(::Type{ClaytonCopula},tau) = 2tau/(1-tau)
radial_dist(C::ClaytonCopula) = Distributions.Gamma(1/C.theta,1)
```

Note the quite small amount of code needed... compared to `R::copula`.

Modularity of the API

The Archimedean API is modular:

- To sample an archimedean, only `radial_dist` and `phi` are needed.
- To evaluate the cdf and (log-)density in any dimension, only `phi` and `inv_phi` are needed.
- Currently, to fit the copula `itau` is needed as we use the inverse tau moment method. But we plan on also implementing inverse rho and MLE (density needed).
- We plan on implementing the **Williamson transforms** so that `radial-dist` can be automatically deduced from `phi` and vice versa, if you dont know much about your archimedean family

Shine of automatic differentiation

To compute archimedean copula densities, the d^{th} derivativer of the generator is needed:

```julia
function phi_d(C::ArchimedeanCopula{d},t) where d
    X = Taylor1(eltype(t),d)
    taylor_expansion = phi(C,t+X)
    coef = getcoeff(taylor_expansion,d) # gets the dth coef.
    return coef * factorial(d) # gets the dth derivative of $\phi$ taken i
end
```

This piece of code is type-stable since d is part of the type, and *much faster* than the equivalent in R::copula that is relying on a C++ implementation of partial derivatives.

A few usage examples

Ex 1: TuringLang/Turing.jl

```julia
using Turing
@model function model(dataset)
    # Priors
    t  ~ TruncatedNormal(1.0, 1.0, 0, Inf)
    t1 ~ TruncatedNormal(1.0, 1.0, 0, Inf)
    t2 ~ TruncatedNormal(1.0, 1.0, 0, Inf)
    X1 = Exponential(t1)
    X2 = Pareto(t2)
    C = SurvivalCopula(ClaytonCopula(2,t),(1,))
    D = SklarDist(C, (X1, X2))
    Turing.Turing.@addlogprob! loglikelihood(D, dataset)
end
```

Other possibility: **dependence between residuals** in bayesian regression context, for exemple.

Ex 2: `SciML/GlobalSensitivity.jl`

Shapley effects models the influence of inputs of a black-box-model on the outputs. This requires dependence structures modeling and is implemented on top of `Copulas.jl` by `SciML/GlobalSensitivity.jl`

See their docs: https://docs.sciml.ai/GlobalSensitivity/stable/tutorials/shapley/

Ex 3: JuliaActuary/EconomicScenarioGenerators.jl

From their readme:

```julia
using EconomicScenarioGenerators, Copulas, Plots
m = BlackScholesMerton(0.01,0.02,.15,100.)
s = ScenarioGenerator(
                      1,  # timestep
                      30, # projection horizon
                      m,  # model
                  )
ss = [s,s] # these don't have to be the exact same, but do need same shape
c = Correlated(ss,ClaytonCopula(2,6))
```

The future

Potential for other integrations

Already identified:

- `AnderGray/ProbabilityBoundsAnalysis.jl` and `AnderGray/PossibilisticArithmetic.jl` : They use copulas but implemented their own versions before this package existed, there is a plan to remove duplicated code and leverage `Copulas.jl`.
- `lucaferranti/FuzzyLogic.jl`: A copula is a T-norm and therefore a fuzzy `AND`: we could leverage `Copulas.jl` to construct new parametric T-norms, or even higher-dimensional ones.

Maybe others you think about?

Potential for future developpement

Future implementations directions depends on your feedback and needs:

- More comprehensive documentation
- Hierarchical archimedeans copulas ? Louivilles ?
- Pair-copulas / Vines constructions ?
- Dependence metrics : strong/weak tail dependance functions for a given copula?

Together with, of course, straightforward to use estimators from real data.

Refs

Hofert, Marius, Martin Mächler, and Alexander J McNeil. 2013. "Archimedean Copulas in High Dimensions: Estimators and Numerical Challenges Motivated by Financial Applications." *Journal de La Société Française de Statistique* 154 (1): 25–63.

McNeil, Alexander J. 2008. "Sampling Nested Archimedean Copulas." *Journal of Statistical Computation and Simulation* 78 (6): 567–81. https://doi.org/10.1080/00949650701255834.

McNeil, Alexander J., and Johanna Nešlehová. 2010. "From Archimedean to Liouville Copulas." *Journal of Multivariate Analysis* 101 (8): 1772–90. https://doi.org/10.1016/j.jmva.2010.03.015.

Nelsen, Roger B. 2006. *An Introduction to Copulas*. 2nd ed. Springer Series in Statistics. New York: Springer.

Sklar, A. 1959. "Fonctions de Repartition à n Dimension Et Leurs Marges." *Université Paris* 8 (3.2): 1–3.