Copulas.jl: Implementation of standard copula routines in Julia

When dependence structures modeling meets multiple dispatch

Oskar Laverny

2023-12-17

Aix-Marseille Université, SESSTIM

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))

There exists a unique copula C such that

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

C is uniquely determined where the corresponding random vector is absolutely continuous.

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

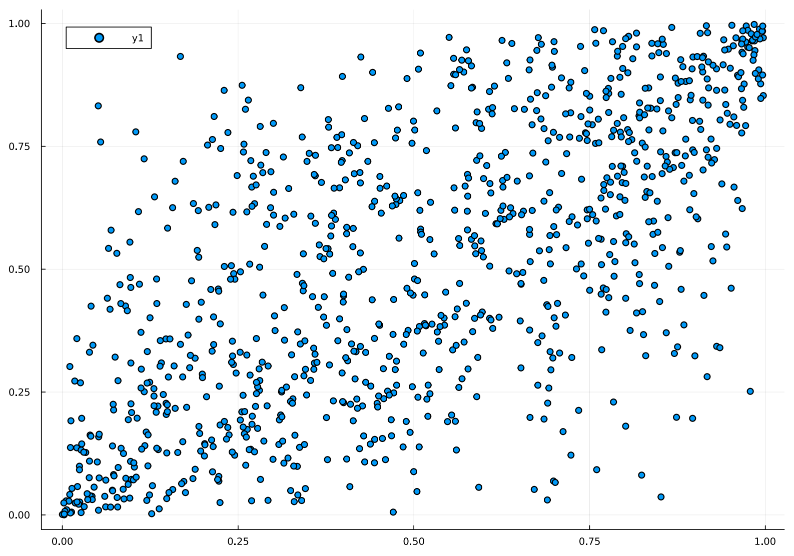

Figure 1: Sample from bivariate Gaussian Copula with sigma=0.7

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 $\texttt{Clayton}(\theta)$ copula.
$\varphi(t) = \exp\{-t^{\theta^{-1}}\}$ generates the $\texttt{Gumbel}(\theta)$ 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, it is the Laplace transfrom of the non-negative r.v. $W = \Gamma_d/R$, where $\Gamma_d \sim \mathrm{Erlang}(d)$ is independent from R.

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

Parametric Exemples

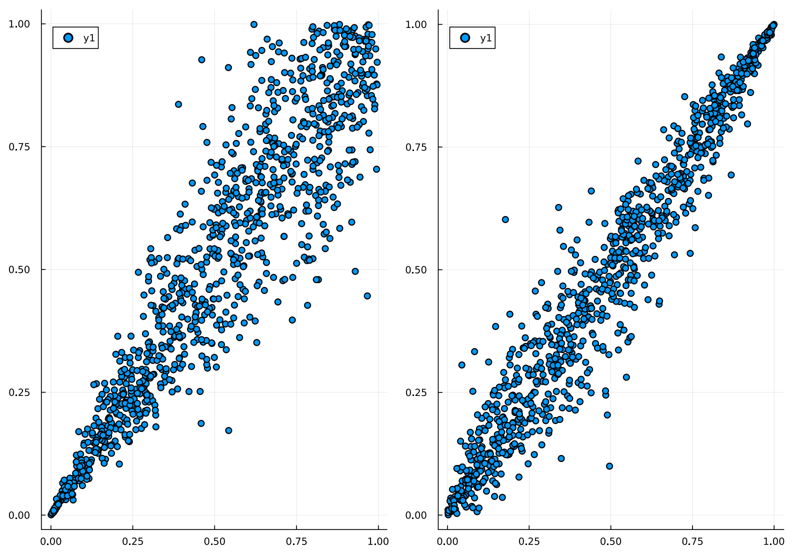

Figure 2: Sample from bivariate Clayton and Gumbel

Generic archimedeans through `WilliamsonTransforms.jl`

Due to generic code, our interface allows for input of any generator, directly or through Williamson d-transfroms:

```julia
using Copulas
G1 = WilliamsonGenerator(Lognormal(), 10)
G2 = GenericGenerator(t -> exp(-2t), monotonicity=Inf)
C1 = ArchimedeanCopula(10,G1)
C2 = ArchimedeanCopula(3,G2)
# You can then use : rand(), pdf(), cdf(), etc...
```

Produced copulas fullfill the full API. Of course fast paths exists for classical models: Claytons, Gumbels, etc.. But you may patch only the methods you need !

Generic Liouville copulas... for free !

Due to multiple dispatch, PR #83 implements Liouville with only ~ 50 more lines of code, with the same genericity:

```
C1 = LiouvilleCopula((1,4,1,10), WilliamsonGenerator(Pareto(), 16))
C2 = LiouvilleCopula((1,7,3), G2)
C3 = LiouvilleCopula((1,2,1), ClaytonGenerator(-0.2))
# You can then use : rand(), pdf(), cdf(), etc...
```

Note that here `ClaytonGenerator` with negative dependence works in any dimension. The *practical limits* of the package's implementation *are* the theoretical limits of d-monotonicity, *contrary to `R::copula` that only implements special cases...*

Non-parametric models

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:

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

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

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

Comming soon: Bernstein Copula, Beta copula, checkerboards and pachwork copula, etc.

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` :

```
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 new archimedean generator can be constructed as follows:

```
struct MyGenerator <: Generator end
phi(G::MyGenerator,t) = exp(-t) # may be any d-monotonous function.
max_monotony(G::MyGenerator) = Inf # may depend on G's params
C = LiouvilleCopula((1,4,3),MyGenerator())
# rand(C,100), cdf(C,spl), pdf(C,spl) etc.. will work.
```

Note the quite small amount of code needed... compared to `R::copula`. The inverse and derivatives of φ can also the provided for performance reasons, but are not needed.

Shine of automatic differentiation

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

```
function phi_d(C::ArchimedeanCopula{d,TG},t) where {d,TG}
    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(t).
end
```

The `getcoeff(phi(C,t+Taylor1(...)),d)` piece **folds out completely at compile time** into a fully precompiled Faa-di-bruno extraction of the dth derivative.

Thus we obtain much better performances than `R::copula` that is relying on a C_{++} implementation of numerical d^{th} derivatives and/or specific implementation for some generators. *Also amount of code...*

A few usage examples

Ex 1: TuringLang/Turing.jl

```julia
using Turing
@model function model(dataset)
    # Priors
    t1, t2  .~ TruncatedNormal(1.0, 1.0, 0, Inf)
    X1,X2,X3 = (Exponential(t1), Pareto(t2), Normal(t2,1))
    C = SurvivalCopula(PlackettCopula(3,t1),(1,))
    D = SklarDist(C, (X1, X2, X3))
    Turing.Turing.@addlogprob! loglikelihood(D, dataset)
end
chain = sample(model(yourdata), NUTS(), MCMCThreads(), 100, 4)
```

Other example : dependence between residuals in bayesian regression context. See our docs: https://lrnv.github.io/Copulas.jl/stable/exemples/turing/

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

Copulas are used to model dependent stocks. From their readme:

```
using EconomicScenarioGenerators, Copulas, Plots
m = BlackScholesMerton(0.01,0.02,.15,100.)
s = ScenarioGenerator(1,  # timestep
                      30, # projection horizon
                      m,)
ss = [s,s] # these don't have to be the exact same
c1 = Correlated(ss,RafteryCopula(2,6))
# you may vary the dependence structure:
c2 = Correlated(ss,FGMCopula(2,0.8))
```

Get out scenario values, options pricing, yields curves, etc.. See https://github.com/JuliaActuary/EconomicScenarioGenerators.jl

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.

- Hierarchical Archimedeans: no necessary and sufficient nesting condition *yet*..
- Vines, but *really* generic.
- Non-parametric estimators: Bernstein, Beta, Patchwork, etc.
- More dependence metrics: tails ? others ?

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

\alert{**Thank you very much!**

Do not hesitate to star the repo if you find it usefull :)}

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.