# Modelling dependence with copulas and R package *acopula* ver. 0.9.2

Tomáš Bacigál [*]

**Abstract:** *We introduce* acopula *package (run under R) that aims to support researchers as well as practitioners in the field of modelling stochastic dependence. Description of tools with examples are given, namely several probability related functions, estimation and testing procedures, and two utility functions.*

Keywords: *Archimax copula, R, estimation, GOF test, copula quantile.*

## 1 Introduction

Copula is a $d$-dimensional function $C\colon [0,1]^d \to [0,1]$, $d \geq 2$, that can combine any univariate cumulative distribution functions to form a joint distribution function $F$ of a random vector $\mathbf{X} = (X_1, \ldots, X_d)$, such that

$$F(x_1, \ldots, x_d) = C\left(F_{X_1}(x_1), \ldots, F_{X_d}(x_d)\right) \tag{1}$$

with $F_{X_i}$ being distribution function associated with $i$-th random variable. Copula itself is a joint distribution function with uniform marginals, thus it is d-increasing, has 1 as neutral element and 0 as annihilator (see [12] for an exhaustive introduction).

Since the turn of century when copulas began to attract attention of masses, several software tools arose. The first public yet commercial to mention was EVANESCE library [7] included in FinMetrics extension to S programming environment (predecessor of R), that provided a rich battery of copula classes, though only bivariate. With emergence of R (free software environment for statistical computing and graphics, [13]) there came open-source packages like *copula* [8] (recently incorporating *nacopula*) and *CDVine* [3] with successor *VineCopula*, that are still under vivid development. For further reading about recent copula software see, e.g., [1].

Here we introduce an R package that extends current offerings on the one hand by class of *Archimax* copulas and on the other by several handy tools to test, modify, manipulate and inference from them and *arbitrary* user-defined absolutely continuous copulas, thus making copulas ready for *application*. That explains the initial letter of the package name.

In short, Archimax copula is a copula $C_{\phi,A}$, that can be represented in the form

$$C_{\phi,A}(u_1, \ldots, u_d) = \phi^{-1}\left[\left(\sum_{i=1}^{d} \phi(u_i)\right) A\left(\frac{\phi(u_1)}{\sum_{i=1}^{d}\phi(u_i)}, \ldots, \frac{\phi(u_d)}{\sum_{i=1}^{d}\phi(u_i)}\right)\right] \tag{2}$$

where $\phi\colon [0,1] \searrow [0,\infty]$, $\phi(1) = 0$, is a so-called generator of strict Archimedean copula and $A\colon \Delta_{d-1} \to [0,1]$ is a Pickands dependence function defined on unit simplex $\Delta_{d-1} = \{(w_1, \ldots, w_d) \in [0,1]^d | \sum_i^d w_i = 1\}$ and fulfilling boundary constraint $A(\mathbf{e}_i) = 1$ where $\mathbf{e}_i = (0, \ldots, 1, \ldots, 0)$ is the unit vector with 1 at position $i$. In bold we will denote a $d$-dimensional vector, e.g., $\mathbf{w} = (w_1, \ldots, w_d)$. Whenever $\phi(t) = -\log(t)$, $\forall t \in [0,1]$, copula $C_{\phi,A}$ belongs also to the class of Extreme-Value (EV) copulas, and equally, with $A \equiv 1$ the Archimax class degenerates to Archimedean class. The only additional constraints put on both functions $\phi, A$ to generate a bivariate Archimax copula are that they need to be convex and $1 \geq A \geq \max$, as proved by Capéraà et al. [4].

[*] Slovak University of Technology in Bratislava, *tomas.bacigal@stuba.sk*

However, in more dimensions things get rather complicated. McNeil and Nešlehova [10] showed that $\phi$ is a generator of an strict Archimedean copula[1] iff its inverse is $d$-monotone, i.e., (I) $\psi = \phi^{-1}$ has continuous derivatives $\psi^{(k)}$ on $[0, \infty]$ and (II) $(-1)^k \psi^{(k)}(t) \geq 0$ for any $k = 1, \ldots, d-2$, and also (III) $(-1)^{d-2}\psi^{(d-2)}$ is non-negative, non-increasing and convex on $[0, \infty]$. On the other hand, as summarized in [6], a $d$-variate copula $C_l$ is an EV copula iff there exists a finite Borel measure $H$ on $\Delta_{d-1}$, called spectral measure, such that $C_l(\mathbf{u}) = \exp(-l(u_1, \ldots, u_d))$ with tail dependence function $l\colon [0,\infty)^d \to [0,\infty)$ given by $l(\mathbf{x}) = \int_{\Delta_{d-1}} \bigvee_{j=1}^{d}(w_j x_j) dH(\mathbf{w})$ [2] and related to Pickands dependence function (due to its homogeneity) via $l(\mathbf{x}) = (\sum_{i=1}^{d} x_i)A(\mathbf{w})$, $w_j = x_j / \sum_{i=1}^{d} x_i$, $j = 1, \ldots, d$. The spectral measure $H$ is arbitrary except for the $d$ moment constraints $\int_{\Delta_{d-1}} w_j dH(\mathbf{w}) = 1$, $j \in \{1, \ldots, d\}$, that stem from the requirement for the margins of copula to be standard uniform, the constraints imply that $H(\Delta_{d-1}) = d$. The function $A$ is still convex and bounded by $\max(\mathbf{w})$ and 1, however these properties do not characterize the class of Pickands dependence functions any more.

The question whether any $\phi$ can be combined with any $l$ or $A$ so that $C_{\phi,A}$ is a copula is still an open problem whenever $d > 2$. In [2] several positive examples based on partitions and general convex sum are given.

In the package, Pickands dependence function is implemented to accept $d-1$ dimensional argument since the last element is complementary, formally $A'(w_1, \ldots, w_{d-1}) = A(w_1, \ldots, w_{d-1}, 1 - \sum_{i=1}^{d-1} w_i)$.

Structure of the R package *acopula* is relatively simple, does NOT use object-oriented S4 classes and is comprehensible from the source code accompanied by explanation notes, so that even inexperienced user can, e.g., track erroneous behaviour if any occurs. Also it does not depend on any additional packages, though it suggest to use some. In the next sections particular functions are detailed and demonstrated on examples.

## 2 Definition lists

Every parametric family of copulas is defined within a list, either by its generator (in case of Archimedean copulas), Pickand's dependence function (Extreme-Value copulas) or directly by cumulative distribution function (CDF) with/or its density. Example of one such definition list follows[3] for generator of Gumbel-Hougaard family of Archimedean copulas

```
> genGumbel()
$parameters
[1] 4

$pcopula
function (t, pars) exp(-sum((-log(t))^pars[1])^(1/pars[1]))

$gen
function (t, pars) (-log(t))^pars[1]

$gen.der
function (t, pars) -pars[1]*(-log(t))^(pars[1]-1)/t

$gen.der2
function (t, pars) pars[1]*(-log(t))^(pars[1]-2)*(pars[1]-1-log(t))/t^2

$gen.inv
function (t, pars) exp(-t^(1/pars[1]))

$gen.inv.der
function (t, pars) -exp(-t^(1/pars[1]))*t^(1/pars[1]-1)/pars[1]

$kendall$coef
function (t) 1 - 1/t
```

[1]In fact, they showed it also for non-strict Archimedean copula when a pseudo-inverse of the generator needs to be used.

[2]$\bigvee$ denotes maximum (join).

[3]Output printing is simplified here whenever contains irrelevant parts.

```
$kendall$icoef
function (t) 1/(1 - t)

$kendall$bounds
[1] 0 1

$spearman$coef
function (t) pPareto(t, c(1.41917, 2.14723, 1, 1))

$spearman$icoef
function (t) qPareto(t, c(1.41917, 2.14723, 1, 1))

$spearman$bounds
[1] 0 1

$lower
[1] 1

$upper
[1] Inf

$id
[1] "Gumbel"
```

where, although some items may be fully optional (here `$pcopula`, `$kendall` and `$spearman`), they can contribute to better performance. The user is encouraged to define new parametric families of Archimedean copula generator (similarly of Pickands dependence function or copula in general) according to his/her needs, bounded only by this convention and allowed to add `pcopula` (stands for probability distribution function or CDF), `dcopula` (density) and `rcopula` (random sample generator) items, however compatibility with desired dimension has to be kept in mind. Currently implemented generators can be listed,

```
> ls("package:acopula",pattern="gen")
[1] "genAMH" "genClayton" "generator" "genFrank" "genGumbel" "genJoe" "genLog"
```

notice the generic function `generator` which points to specified definition lists.

Similarly, Pickands dependence functions are defined, namely Gumbel-Hougaard, Tawn, Galambos, Hüsler-Reiss (last three form only bivariate EV), extremal dep. functions and generalized convex combination of arbitrary valid dep. functions (see [11]). So are definition lists available for generic (i.e., not necessarily Archimax) copula, e.g. normal, Farlie-Gumbel-Morgenstern, Plackett and Gumbel-Hougaard parametric family. Their corresponding function names starts with `dep` and `cop`, respectively.

Since the class of Archimax copulas contains Archimedean and EV class as its special cases, the setting `depfu = dep1()` and `generator = genLog()` can distinguish them, respectively.

There are not many known dependence function parametric families capable of producing more-than-2-dimensional EV copula, much less Archimax copula, for that reason the (generalized) convex combination may come useful, used for instance in partition-based approach introduced by [2]. More specifically, having $m$ dependence functions $A_j$, $j = 1, \dots, m$, the function $A_{gcc}\colon \Delta_{d-1} \to [\frac{1}{d}, 1]$ given as

$$A_{gcc}(\mathbf{w}) = \sum_{j=1}^{m} s_j A_j \left( \frac{\alpha_{j1} w_1}{s_j}, \dots, \frac{\alpha_{jd} w_d}{s_j} \right), \quad \text{with } s_j = \left( \sum_{i=1}^{d} \alpha_{ji} w_i \right) \text{ and } \alpha_{ji} > 0. \tag{3}$$

Note that $\sum_{j=1}^{m} \alpha_{ji} = 1$, and if $\alpha_{ji} = \lambda_j$ for all $i, j$, then $A_{gcc} = \sum_{j=1}^{m} \lambda_j A_j$ is standard convex combination (thus symmetric). Furthermore consider a partition $\mathcal{P} = \{B_1, \dots, B_k\}$ of $\{1, \dots, d\}$. Then the function $l_{\mathcal{P}}(x_1, \dots, x_d) = \sum_{j=1}^{k} \left( \bigvee_{i \in B_j} x_i \right)$ is a tail dependence function based on spectral measure $H(\mathbf{w}) = \sum_{j=1}^{k} (\operatorname{card} B_j)\, \delta_{B_j} \left( (\operatorname{card} B_j) w_1, \dots, (\operatorname{card} B_j) w_d \right)$ where $\delta_{B_j}\colon [0, \infty)^d \to [0, \infty)$ is the generalized Dirac measure given by $\delta_{B_j}(1_{B_j}) = 1$ and $\delta_{B_j}(x) = 0$ whenever $x \neq 1_{B_j}$. For fixed

$d = 3$ there are 5 partitions of $\{1, 2, 3\}$ with corresponding Pickands dependence functions [4]

$$\begin{aligned}
A^*(w_1, w_2, w_3) &= w_1 + w_2 + w_3 \quad \text{when } \mathcal{P}^* = \{\{1\}, \{2\}, \{3\}\} \\
A_*(w_1, w_2, w_3) &= w_1 \vee w_2 \vee w_3 \qquad \mathcal{P}_* = \{\{1, 2, 3\}\} \\
A_1(w_1, w_2, w_3) &= w_1 + w_2 \vee w_3 \qquad \mathcal{P}_1 = \{\{1\}, \{2, 3\}\} \\
A_2(w_1, w_2, w_3) &= w_2 + w_1 \vee w_3 \qquad \mathcal{P}_2 = \{\{2\}, \{1, 3\}\} \\
A_3(w_1, w_2, w_3) &= w_3 + w_1 \vee w_2 \qquad \mathcal{P}_3 = \{\{3\}, \{1, 2\}\}.
\end{aligned}$$

Thus we get special parametric class `depGCC(ldepPartition3D(),dim=3)` with $3\times 5$ parameters leading to 3-dimensional copula.

Any definition list item can be replaced already during the function call as shown in the next subsections. Thus one can set starting value of parameter(s) and their range in estimation routine, for instance.

## 3 Probability functions

First thing one would expect from a copula package is to obtain a value of desired copula in some specific point. To show variability in typing commands, consider again Gumbel-Hougaard copula with parameter equal to 3.5 in point (0.2,0.3). Then the following commands give the same result.

```
> pCopula(data=c(0.2,0.3),generator=genGumbel(),gpars=3.5)
> pCopula(data=c(0.2,0.3),generator=genGumbel(parameters=3.5))
> pCopula(data=c(0.2,0.3),generator=generator("Gumbel"),gpars=3.5)
> pCopula(data=c(0.2,0.3),generator=generator("Gumbel",parameters=3.5))
> pCopula(data=c(0.2,0.3),copula=copGumbel(),pars=3.5)
> pCopula(data=c(0.2,0.3),copula=copGumbel(parameters=3.5))
> pCopula(data=c(0.2,0.3),generator=genLog(),depfun=depGumbel(),dpars=3.5)
> pCopula(data=c(0.2,0.3),generator=genLog(),depfun=depGumbel(parameters=3.5))
[1] 0.1723903
```

If we need probabilities that a random vector would not exceed several points, those can be supplied to `data` in rows of matrix or data frame.

Conversely, given an incomplete point and a probability, the corresponding quantile emerge.

```
> pCopula(c(0.1723903,0.3),gen=genGumbel(),gpar=3.5,quantile=1)
> pCopula(c(NA,0.3),gen=genGumbel(),gpar=3.5,quan=1,prob=0.1723903)
> qCopula(c(0.3),quan=1,prob=0.1723903,gen=genGumbel(),gpar=3.5)
[1] 0.1999985
```

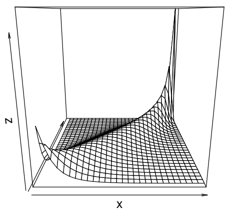
Conditional probability $P(X < x|Y = y)$ of a random vector $(X, Y)$ has similar syntax.

```
> cCopula(c(0.2,0.3),conditional.on=2,gen=genGumbel(),gpar=3.5)
[1] 0.2230437
> qCopula(c(0.3),quan=1,prob=0.2230437,cond=c(2),gen=genGumbel(),gpar=3.5)
[1] 0.200005
```

Sometimes the density of a copula is of interest, perhaps for visualisation purposes, such as in the following example

```
x <- seq(0,1,length.out=30)
y <- seq(0,1,length.out=30)
z <- dCopula(expand.grid(x,y),generator=genGumbel(),gpars=3.5)
dim(z) <- c(30,30)
persp(x,y,z)
```

[4] For computational convenience, the maximum operator $\vee$ is approximated to have smooth edges, so that $w_1 \vee \ldots w_d = \bigvee_{i=1}^d w_i \approx \left(\sum_{i=1}^d w_i^\lambda\right)^{1/\lambda}$, where power $\lambda$ defaults to 8 and can be changed by passing the argument `power`.

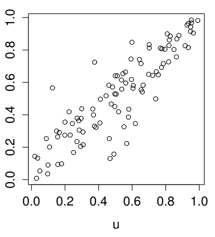


where instead of `persp` from package *graphics* a more impressive output is given by package *rgl* with function `persp3D`.

If definition lists do not contain explicit formulas for (constructing) density, the partial derivatives are approximated linearly. This is mostly the case with 3- and more-dimensional copulas.

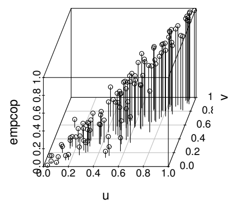
Sampling from the copula is, unsurprisingly, also provided.

```
sample <- rCopula(n=100,dim=2, generator=genGumbel(), gpars=3.5)
plot(sample)
```



Sometimes no assumption about parametric family of copula is made, instead an empirical distribution is of more interest. Then for a given data, say, the previous random sample, one may ask for value of empirical copula in specific point(s) and more easily in the points of its discontinuity.

```
> pCopulaEmpirical(c(0.2,0.3),base=sample)
[1] 0.14
> empcop <- pCopulaEmpirical(sample)
> scatterplot3d::scatterplot3d(cbind(sample,empcop),type="h",angle=70)
```



## 4 Estimation

Currently, there are two universal methods for parameters estimation implemented in the package (named `technique`): "ML", maximum (pseudo)likelihood method employing copula density, and "LS", least squares method minimizing distance to empirical copula. In short, given a random sample $\mathbf{U}_j$, $j = 1, \ldots, n$, from a continuous distribution with uniform (1-dimensional) margins, joint distribution function $C_\theta$ and density $c_\theta$, define the maximum likelihood estimator as

$$\hat{\theta} = \arg\max_{\theta \in \mathcal{Q}} \sum_{j=1}^{n} \log[c_\theta(U_{j1}, \ldots, U_{jd})] \tag{4}$$

and least-squares estimator as

$$\hat{\theta} = \arg\min_{\theta \in \mathcal{Q}} \sum_{j=1}^{n} \left(C_n(U_{j1}, \ldots, U_{jd}) - C_\theta(U_{j1}, \ldots, U_{jd})\right)^2, \tag{5}$$

where $\mathcal{Q}$ is parameter space of copula $C_\theta$ parameters set, $C_n(\mathbf{u}) = \frac{1}{n}\sum_{j=1}^{n} 1(U_{j1} \le u_1, \ldots, U_{jd} \le u_d)$, $\mathbf{u} \in [0,1]^d$, is the so-called empirical copula and $1(\cdot)$ is the indicator function which yields 1 whenever $\cdot$ is true and 0 otherwise. Both 'techniques' supplies function to perform optimization `procedure` over, thus finding those parameters that correspond to an optimum. The 'procedures' are three: "optim", "nlminb" and "grid". First two are system native, based on well-documented smart optimization methods, the third one uses brute force to get approximate global maximum/minimum and can be useful with multi-parameter copulas, at least to provide starting values for the other two 'procedures'.

For one-parameter bivariate copula families we also provide estimation method based on relation between copula parameter and rank-based measures of dependence (`technique="icorr"`), currently Kendall's tau (`corrtype="kendall"`),

$$\tau = 4 \iint_{[0,1]^2} C_\theta(u,v)\, dC_\theta(u,v) - 1, \tag{6}$$

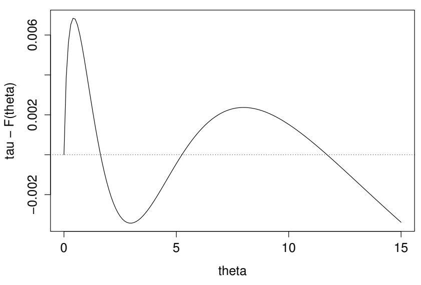
and Spearman's rho (`corrtype="spearman"`),

$$\rho = 12 \iint_{[0,1]^2} C_\theta(u,v)\, duv - 3. \tag{7}$$

Some of them have no closed form and need to be approximated, mostly by cumulative distribution function of the Pareto type IV distribution defined by

$$F_{\text{Pareto}}(x) = \begin{cases} 1 - \left(1 + \left(\frac{x-p_4}{p_1}\right)^{\frac{1}{p_2}}\right)^{-p_3} & x \ge \mu \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

with parameters $p_i > 0$, $i = 1,2,3$, and $p_4 \in \mathcal{R}$. Since $F_{\text{Pareto}}$ is easily invertible, the estimation is fast and still acceptably precise keeping the approximation error in between -0.01 and +0.01 for dependence strength up to $\rho = 0.96$. For instance, relation between $\tau$ and Frank copula parameter (for positive dependence) is approximated by $F_{\text{Pareto}}$ with parameters $p \approx (7.5, 1.3, 0.9, 0)$ and error $\tau(\theta) - \hat{F}_{\text{Pareto}}(\theta)$ plotted bellow.



The next few examples sketch various options one has got for copula fitting.

```
> eCopula(sample,gen=genClayton(),dep=depGumbel(),
+ technique="ML",procedure="optim",method="L-BFGS-B")
generator parameters:  0.09357958
   depfun parameters:  3.52958
   ML function value:  82.63223
    convergence code:  0
> eCopula(sample,gen=genClayton(),dep=depGumbel(),tech="ML",proc="nlminb")
```

```
generator parameters:  0.09183014
   depfun parameters:  3.533706
   ML function value:  82.63228
    convergence code:  0
> eCopula(sample,gen=genClayton(),dep=depGumbel(), tech="ML",proc="grid",
+ glimits=list(c(0),c(5)),dlimits=list(c(1),c(10)),pgrid=10)
generator parameters:  0.5555556
   depfun parameters:  3
   ML function value:  80.63322
    convergence code:
> eCopula(sample,gen=genGumbel(),technique="icorr",corrtype="kendall")
generator parameters:  3.442281
   depfun parameters:
   icorr function value:
    convergence code:
```

In addition, ”optim” procedure has several `method`s to choose from: ”L-BFGS-B”, ”Nelder-Mead”, ”BFGS”, ”CG”, ”SANN”, ”Brent”.

So far, no precision for copula parameters is provided.

## 5 Testing

Having set of observations, it is often of great interest to test whether the estimated copula suffices to describe dependence structure in the data. For this purpose many goodness-of-fit tests were proposed, yet the principle remains to use different criterion than that employed with estimation of the copula parameters. Here we implement one of the 'blanket' tests described in [5] that is based on Kendall's transform of joint distribution function, $\mathcal{K}_\theta(t) = P(C_\theta(\mathbf{u}) \le t)$, which reduces multivariate problem to one dimension. Its empirical version can be computed by $\mathcal{K}_n(t) = \frac{1}{n}\sum_{j=1}^n 1(C_n(\mathbf{U}_j) \le t)$, $t \in [0,1]$. To test whether theoretical $\mathcal{K}$ matches the empirical one, the Cramér von-Mises test statistic $S_n = \int_0^1 \sqrt{n}(\mathcal{K}_\theta(t) - \mathcal{K}_n(t))^2 dt$ is available. As the asymptotic distributions of $S_n$ depend on unknown copula $C_\theta$ and on $\theta$ in particular, approximate $p$-values must be found via simulation. The specific parametric bootstrap procedure is minutely described in [5].

In the example below normal copula is tested on the Gumbel copula sample data.

```
> gCopula(sample,cop=copNormal(),
+ etechnique="ML",eprocedure="optim",ncores=1,N=100)
Loading required package: mvtnorm
  |==============================================================| 100%

 Blanket GOF test based on Kendall's transform

statistic       q95   p.value
0.1195500 0.1658125 0.1800000
-----------------------------
data:  sample
copula:  normal
estimates:
      pars     fvalue
 0.9155766 80.3420886
```

Although the p-value does not lead to rejection of the copula adequacy, its low value and small data length arouse suspicion. As for the other arguments, `N` sets number of bootstrap cycles and their parallel execution can be enabled by setting number of processor cores in `ncores` (not available under Windows OS). Package *mvtnorm* has been loaded to assist with simulation from normal copula, and when missing, internal but slower routine would be run instead.

The traditional parametric bootstrap-based procedure to approximate p-value, when theoretical probability distribution of the test statistic is unknown, is reliable yet computationally very exhaustive, therefore recently a method based on multiplier central limit theorem and proposed by [9] becomes popular with large-sample testing. Its implementation to testing goodness of parametric copula fit is scheduled for future package updates. Nevertheless, the multiplier method takes part here in another test comparing two empirical copulas, i.e. dependence structure of two data sets, see [14] and package *TwoCop*. In the following example, random sample of the above Gumbel-Hougaard copula is tested for sharing common dependence structure with sample simulated from Clayton copula, parameter of which corresponds to the same Kendall's rank correlation ($\tau = 0.714$).

```
> sampleCl <- rCopula(n=100,dim=2,generator=genClayton(),gpars=5)
> gCopula(list(sample,sampleCl),ncores=1,N=100)
  |=========================================================| 100%

 Test of equality between 2 empirical copulas

 statistic        q95    p.value
0.09791672 0.52893392 0.66000000
-----------------------------
data:  sample sampleCl
copula:
estimates:
NULL
```

Obviously, the test fails to distinguish copulas with differing tail dependence, at least having small and moderate number of observations, however it is sensitive enough to a difference in rank correlation.

The last procedure to mention checks the properties of a $d$-dimensional copula ($d \geq 2$), that is, being $d$-increasing as well as having 1 as neutral element and 0 as annihilator. The purpose is to assist approval of new copula constructs when theoretical proof is too complicated. The procedure examines every combination of discrete sets of copula parameters, in the very same fashion as within ”grid” procedure of `eCopula`, by computing a) first differences recursively over all dimensions of an even grid of data points,i.e., C-volumes of subcopulas, b) values on the margin where one argument equals zero and c) where all arguments but one equals unity. Then whenever the result is a) negative, b) non-zero or c) other than the one particular argument, respectively, a record is made and first 5 are printed as shown below. In the example we examine validity of an assumed Archimedean copula generated by Gumbel-Hougaard generator family, only with a parameter being out of bounds.

```
> isCopula(generator=genGumbel(lower=0),dim=3,glimits=list(0.5,2),
+ dagrid=10,pgrid=4,tolerance=1e-15)

 Does the object appears to be a copula(?):  FALSE

 Showing 2 of 2 issues:

  dim property      value gpar
1   2    monot -0.1534827  0.5
2   3    monot -0.1402209  0.5
```

Three parameter values $(0.5, 1, 1.5, 2)$ were used, each supposed copula were evaluated in $10^3$ grid nodes, and every violation of copula properties (the most extremal value per dimension and exceeding `tolerance`) were reported. Thus it is seen, that parameter value $0.5$ does not result in copula because 3-monotonicity is not fulfilled (negative difference already in the second-dimension run). Note that without redefinition of lower bound the parameter value $0.5$ would be excluded from the set of Gumbel-Hougaard copula parameters.

## 6 Utilities

For the *acopula* package to work many utility functions were created during development that were neither available in the basic R libraries nor they were found in contributed package under CRAN. Most of them are hidden within the procedures described above, however the two following are accessible on demand. The first to mention is a linear approximation of partial derivative of any-dimensional function and of any order with specification of increment (theoretically fading to zero) and area (to allow semi-differentiability)

```
> fun <- function(x,y,z) x^2*y*exp(z)
> nderive(fun,point=c(0.2,1.3,0),order=c(2,0,1),difference=1e-04,area=0)
[1] 2.600004
```

whereas the second utility function numerically approximates integration (by trapezoidal rule) such as demonstrated on example of joint standard normal density with zero correlation parameter

```
> nintegrate(function(x,y) mvtnorm::dmvnorm(c(x,y)),
+           lower=c(-5.,-5.),upper=c(0.5,1),subdivisions=30)
[1] 0.5807843
> pnorm(0.5)*pnorm(1)
[1] 0.5817583
```

fine-tuned by number of subdivisions. However, it must be admitted that numerical integration performs better with package *cubature*.

## 7 Conclusion

All the introduced and exemplified procedures are (a) extendible to arbitrary dimension, which is one of the significant contributions of the package. If explicit formulas are unavailable (through definition lists) then numerical approximation does the job. Another significant benefit is brought by (b) conditional probability and quantile function of the copula, as well as estimation methods based on least squares and grid complementing the usual maximum-likelihood method. Together with implementing (c) generalization of Archimedean and Extreme-Value by Archimax class with a (d) construction method of Pickands dependence function, (e) fast non-parametric estimation of one-parameter copulas , (f) numerical check of copula properties useful in new parametric families development, and (g) parallelized goodness-of-fit test based on Kendall's transform, these all (and under one roof) make the package competitive among both proprietary and open-source software tools for copula based analysis, to the date.

Yet because the routines are written solely in R language and rely on no non-standard packages (optionally), some tasks may take longer to perform. Nevertheless the source code is easy to access, understand and modify if necessary.

Future improvement is seen mainly in providing additional methods for parameters estimation (for multi-parameter copulas based on various dependence measures) and GoF tests, as well as connecting with other copula packages to simplify practical analysis.

Author appreciates any comments, bug reports or suggestions.

## Acknowledgement

The work on this contribution was supported by grants APVV-0074-10 and APVV-0496-10.

## References

[1] Bacigál, T.: Recent tools for modelling dependence with copulas and R. Forum Statisticum Slovacum **8**(1), 62–67 (2012)

[2] Bacigál, T., Mesiar, R.: 3-dimensional Archimax copulas and their fitting to real data. In: COMPSTAT 2012, 20th International conference on computational statistics. Limassol,Cyprus,27.-31.8.2012. The International Statistical Institute, 81–88 (2012).

[3] Brechmann, E.C., Schepsmeier, U.: Modeling Dependence with C- and D-Vine Copulas: The R Package CDVine. Journal of Statistical Software **52**(3), 1–27 (2013).

[4] Capéraà, P., Fougères, A.-L., Genest, C.: Bivariate distributions with given extreme value attractor. J.Multivariate Anal. **72**(1) 30–49 (2000).

[5] Genest, C., Rémillard, B. and Beaudoin, D.: Goodness-of-fit tests for copulas: A review and a power study. Insurance: Mathematics and Economics **44**, 199–213 (2009).

[6] Gudendorf, G. and Segers, J.: Extreme-value copulas. In Copula Theory and Its Applications. Springer Berlin Heidelberg, 127–145 (2010).

[7] Insightful Corp.: EVANESCE Implementation in S-PLUS FinMetrics Module (2002). `faculty.washington.edu/ezivot/book/QuanCopula.pdf` Cited 15 Feb 2013.

[8] Kojadinovic, I. and Yan, J.: Modeling Multivariate Distributions with Continuous Margins Using the copula R Package. Journal of Statistical Software **34**(9), 1–20 (2010).

[9] Kojadinovic, I., Yan, J., Holmes, M.: Fast large-sample goodness-of-fit for copulas. Statistica Sinica **21**(2), 841–871 (2011).

[10] McNeil, A.J., Nešlehová, J.: Multivariate Archimedean copulas, d-monotone functions and $l_1$-norm symmetric distributions, Annals of Statistics **37**(5B), 3059–3097 (2009).

[11] Mesiar, R., Jágr, V.: $d$-Dimensional dependence functions and Archimax copulas. Fuzzy Sets and Systems **228**, 78–87 (2012).

[12] Nelsen, R. B.: An introduction to copulas. Springer (2006).

[13] R Development Core Team. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria (2012). `http://www.R-project.org/` Cited 15 Feb 2013.

[14] Rémillard, B., Scaillet, O.: Testing for equality between two copulas. Journal of Multivariate Analysis **100**(3), 377386 (2009).