kdecopula: An R Package for the Kernel Estimation of Copula Densities

Thomas Nagler
Technische Universität München

Abstract

We describe the R package **kdecopula** which provides fast implementations of various kernel estimators for the copula density. Due to its several plotting options it is particularly useful for the exploratory analysis of dependence structures. It can be further used for accurate nonparametric estimation of copula densities and resampling. The implementation features spline interpolation of the estimates to allow for fast evaluation of density estimates and integrals thereof. We utilize this for a fast renormalization scheme that ensures that estimates are *bona fide* copula densities and additionally improves the estimators' accuracy. The performance of the methods is illustrated by simulations.

Keywords: dependence, copula, nonparametric, kernel density, exploratory data analysis, R.

1. Introduction

Dependence modeling with copulas has attracted a lot of attention in recent decades. By now, copulas are established tools in many fields of applied statistics, such as finance (Cherubini, Luciano, and Vecchiato 2004), hydrology (Salvadori and De Michele 2007), or machine learning (Elidan 2013).

At the very heart of copula theory is the famous theorem of Sklar (Sklar 1959). It states that any multivariate distribution function can be decomposed into the marginal distributions and a copula, which captures the dependence between variables. Let X and Y be two continuous random variables with joint distribution F and marginal distributions F_X and F_Y. Then, for all (x, y) in the support of the random vector (X, Y),

$$F(x, y) = C\big(F_X(x), F_Y(y)\big).$$

The copula $C : [0, 1]^2 \to [0, 1]$ is the bivariate distribution function of the random vector $(U, V) = \big(F_X(X), F_Y(Y)\big)$ which has uniform marginal distribution. If F admits a density, we can also decompose the density f into

$$f(x, y) = c\big(F_X(x), F_Y(y)\big) f_X(x) f_Y(y), \tag{1}$$

where c, f_X and f_Y are the densities corresponding to C, F_X and F_Y, respectively.

One of the major benefits of copula-based modeling is that inference for marginal distributions can be separated from the modeling of the dependence structure, i.e., the copula. For the estimation of the copula density c, it is most common to take a two-step approach: First,

obtain estimates $\widehat{F}_X, \widehat{F}_Y$ of the marginal distributions. A convenient and flexible way to do this is to use the empirical distribution function as an estimator. Second, define *pseudo-observations* $\left(\widehat{U}, \widehat{V}\right) = \left(\widehat{F}_X(X), \widehat{F}_Y(Y)\right)$. The copula density is then estimated as the joint density of $\left(\widehat{U}, \widehat{V}\right)$.

Often, one assumes a parametric model for the copula density c and estimates its parameters by maximum-likelihood. Although there is a large variety of parametric copula models, they notoriously lack flexibility and bear the risk of misspecification. Nonparametric density estimators remedy these issues. But since copulas live on a bounded support — the unit hypercube – estimators have to be carefully tailored to this problem.

A specific class of nonparametric density estimators are kernel estimators. They are a popular tool for exploratory data analysis and widely used in many disciplines (e.g. Aitken and Lucy 2004; Kie, Matthiopoulos, Fieberg, Powell, Cagnacci, Mitchell, Gaillard, and Moorcroft 2010). The package **kdecopula** implements several bivariate kernel copula density estimators that have been proposed in recent years. In a nutshell, the package provides methods for:

- estimation
- bandwidth selection
- simulation
- visualization.

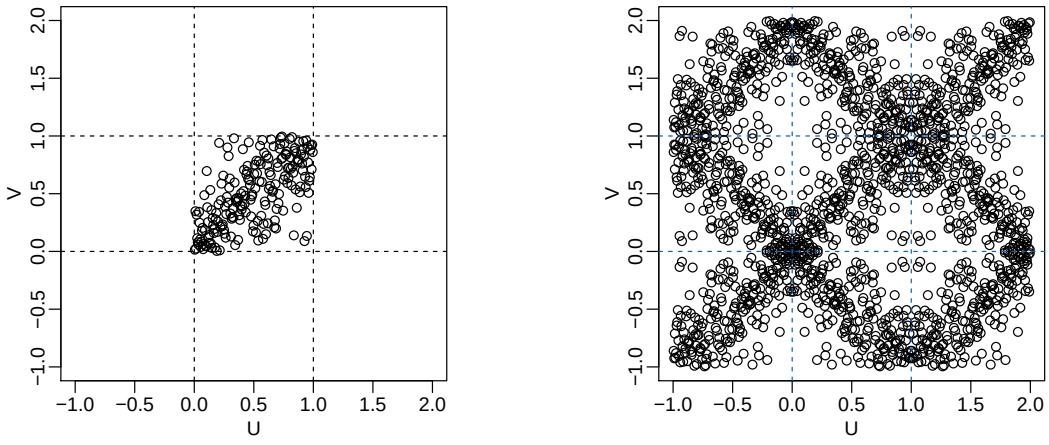
Existing methods for the kernel estimation of copula densities in R are provided by the function `kcopula` from the **ks** package (Duong 2014), the function `npcopula` from the **np** package (Hayfield and Racine 2008), and the function `BiCopMetaContour` from **VineCopula** (Schepsmeier, Stoeber, Brechmann, Graeler, Nagler, and Erhardt 2015). However, all these implementations merely constitute ad hoc solutions to the problem and do not reflect the numerous specialized contributions on the topic. Apart from kernel estimators, Schellhase (2014) implemented nonparametric copula density estimators based on penalized likelihood estimation in the **pencopula** package (using B-splines or Bernstein polynomials). The extension **penDvine** (Schellhase 2015) provides a convenient version with automatic bandwidth selection.

In Section 2, we give a review of kernel copula density estimators and point to the relevant literature. Section 3 describes the functionality of the package and gives examples for its use. In Section 4, we give background on the implementation of the estimators using spline interpolation for fast evaluation and renormalization of the estimates. The statistical accuracy of the estimators in this package and other nonparametric copula density estimators (see previous paragraph) is compared in Section 5. A summary is given in Section 6.

2. Kernel estimators of the copula density: a review

This section will review different approaches to kernel estimation of the copula density. As is common in the literature, we focus on the bivariate case.

Assume we have *iid* observations (U_i, V_i), $i = 1, \ldots, n$, from a bivariate copula C and are interested in the estimation of the corresponding density $c(u, v)$. One could apply the usual



(a) Original data (b) Augmented data

Figure 1: The data augmentation process. The set in (b) is obtained by reflecting all original data points w.r.t. to all corners and edges.

kernel density estimator to this problem:

$$\widehat{c}_n(u,v) = \frac{1}{n}\sum_{i=1}^{n} K_{b_n}\big(u - U_i\big)K_{b_n}\big(v - V_i\big), \quad \text{for all } (u,v) \in [0,1]^2,$$

where we used the notation $K_b(\cdot) = K(\cdot/b)/b$. K is typically assumed to be a symmetric, bounded probability density function on $\mathbb{R}^2$ and $b_n > 0$ is the smoothing or bandwidth parameter. There is a problem, however. The estimator will put a considerable amount of probability mass outside of the unit square. This implies that $\widehat{c}_n$ is not a density function on $[0,1]^2$, because it does not integrate to one. The estimator will additionally suffer from severe bias at the boundaries (see, e.g., Charpentier, Fermanian, and Scaillet 2006). Three different approaches to tackle this problem have emerged. All three techniques arose initially in the context of univariate kernel density estimation on the unit line. The following sections explain the ideas behind them (in the context of copulas) and give references for more detailed accounts.

2.1. The mirror-reflection method

An intuitive way of adapting $\widehat{c}_n$ to make sure that it is a density on $[0,1]^2$ is the following: gather all probability mass that was put outside of the unit square, and redistribute it back to $[0,1]^2$. This is the idea behind the *mirror-reflection technique* which was proposed for copula density estimation by Gijbels and Mielniczuk (1990). As indicated by the name, all data are reflected at the corners and edges of the boundary region. The augmented data set containing all reflections is given by

$$\begin{aligned}\big(\tilde{U}_{ik}, \tilde{V}_{ik}\big)_{k=1,\dots,9} = \Big\{&(U_i, V_i), (-U_i, V_i), (U_i, -V_i), (-U_i, -V_i), (U_i, 2 - V_i),\\ &(-U_i, 2 - V_i), (2 - U_i, V_i), (2 - U_i, -V_i), (2 - U_i, 2 - V_i)\Big\}.\end{aligned}$$

A visualization of the augmented data set is given in Figure 1. The *mirror-reflection estimator* is then defined as the usual kernel density estimator on the augmented data:

$$\widehat{c}_n^{(MR)}(u,v) = \frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{9} K_{b_n}\big(u-\tilde{U}_{ik}\big)K_{b_n}\big(v-\tilde{V}_{ik}\big), \qquad \text{for all } (u,v)\in[0,1]^2.$$

By reflecting all data points at the corners and edges also the probability mass outside of the unit square gets reflected back to the interior. As a result, the estimator now integrates to one. A detailed analysis of the asymptotic properties and a method for automatic bandwidth selection are given in Nagler (2014).

2.2. The Beta kernel method

A second approach is to use kernels whose support matches the support of the density we want to estimate, and vary the shape of those kernels depending on the point where density shall be estimated. This is achieved by so-called *boundary kernels*, and *Beta kernels* are one instance. An estimator of the copula density based on this idea was proposed by Charpentier *et al.* (2006):

$$c^{(\beta)}(u,v) = \frac{1}{n}\sum_{i=1}^{n}\beta\left(U_i, \frac{u}{b_n}+1, \frac{1-u}{b_n}+1\right)\beta\left(V_i, \frac{v}{b_n}+1, \frac{1-v}{b_n}+1\right), \quad (u,v)\in[0,1]^2,$$

where $\beta(\cdot,p,q)$ is the density of a Beta(p,q)-distributed random variable. We refer to Nagler (2014) for details on asymptotics and bandwidth selection.

2.3. The transformation method

A third approach is inspired by the early work of Devroye and Györfi (1985) and was introduced to kernel copula density estimation by Charpentier *et al.* (2006). The simple idea is to transform the data so that it is supported on the full $\mathbb{R}^2$ (instead of the unit cube). On this transformed domain, standard kernel techniques can be used to estimate the density. An adequate back-transformation then yields an estimate of the copula density. For the transformation, the inverse of standard normal *cdf* is most common, since it is known that kernel estimators tend to do well for Gaussian random variables.

Denote Φ as the standard Gaussian *cdf* and ϕ its first order derivative. Then $(X_i,Y_i) = \big(\Phi^{-1}(U_i),\Phi^{-1}(V_i)\big)$ is a random vector with Gaussian margins and copula C. By Sklar's Theorem, the corresponding density f can be written as

$$f(x,y) = c\big(\Phi(x),\Phi(x)\big)\phi(x)\phi(y). \tag{2}$$

This density can be easily estimated by a standard kernel estimator. From such an estimator $\widehat{f}_n$, we can derive an estimator for the copula density c by isolating c in (2):

$$\widehat{c}_n^{(T)}(u,v) = \frac{\widehat{f}_n\big(\Phi^{-1}(u),\Phi^{-1}(v)\big)}{\phi\big(\Phi^{-1}(u)\big)\phi\big(\Phi^{-1}(v)\big)}, \quad \text{for all } (u,v)\in[0,1]^2.: \tag{3}$$

This procedure is illustrated in Figure 2. The left panel shows the original data from the copula density c; next to it we see the transformed data after the inverse Gaussian *cdf* has

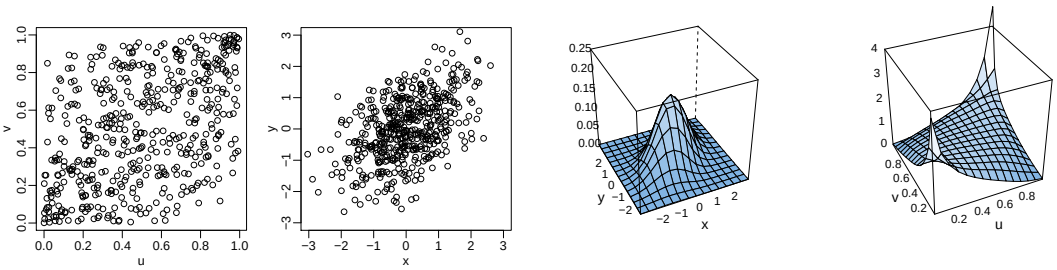


Figure 2: The transformation estimator. From left to right: copula sample, transformed sample, kernel density estimate for transformed sample and kernel estimate of the copula density.

been applied. The third plot shows a kernel estimate of the density of the transformed data; and finally, the fourth plot shows the corresponding kernel estimate of the copula density.

The most natural choice for f_n is the conventional kernel density estimator. More recently, Geenens, Charpentier, and Paindaveine (2014) proposed to use a local likelihood estimator with nearest-neighbor bandwidths instead. Another recent extension was introduced by Wen and Wu (2015) who suggested to taper the back-transformation in the tails by increasing the variance of the Gaussian densities in the denominator of (3). For more details, we refer to the original papers.

3. The package's functionality

In the following, we describe the most important functions provided by the package. All function either produce or take objects of the S3-class `kdecopula` for which several methods are available.

3.1. Estimation and bandwidth selection: `kdecop`

At the core of the **kdecopula** package is the function `kdecop` which estimates the copula density from data. The only mandatory input is an $n \times 2$ matrix of copula data, i.e., data with standard uniform margins. Such data is usually obtained in a first step by applying the empirical marginal *cdf*s to the data. This is equivalent to a rank transformation as shown below.

```
library("kdecopula")  # load package
data("wdbc")  # load data

# apply empirical cdf to obtain copula data
UV <- apply(wdbc[, c(2, 8)], 2, rank) / (nrow(wdbc) + 1)

# kernel estimation of the copula density
kde.fit <- kdecop(UV)
```

```
summary(kde.fit)
> Kernel copula density estimate
> ------------------------------
> Variables:    mean radius -- mean concavity
> Observations: 569
> Method:       Transformation local likelihood, log-quadratic ('TLL2')
> Bandwidth:    alpha = 0.353621
>               B = matrix(c(0.71, 0.7, -1.09, 1.09), 2, 2)
> ---
> logLik: 197.28    AIC: -360.19    cAIC: -359.06    BIC: -285.56
> Effective number of parameters: 17.18
```

The output of the function `kdecop` is an object of class `kdecopula` that contains all information collected during the estimation process and summary statistics such as *AIC* or the *effective number of parameters/degrees of freedom*. These can also be accessed via the usual generic functions, e.g.,

```
logLik(kde.fit)
> 'log Lik.' 197.2755 (df=17.18016)
AIC(kde.fit)
> [1] -360.1907
```

The function `kdecop` provides all estimation methods mentioned in Section 2. The estimation method can be specified via the `method` argument, e.g., `kdecop(..., method = "MR")`. For each method, we have implemented an automatic bandwidth selection procedure. Below we list all implemented methods including a reference to the bandwidth selection procedure used:

`MR`
: The mirror-reflection estimator of Gijbels and Mielniczuk (1990). Smoothing parameters are selected by minimizing the AMISE using the Frank copula as the reference copula (cf., Nagler 2014, Section 3.2.4).

`beta`
: The beta kernel estimator of Charpentier *et al.* (2006). Smoothing parameters are selected by minimizing the AMISE using the Frank copula as the reference copula (cf., Nagler 2014, Section 3.3.3).

`T`
: The transformation estimator of Charpentier *et al.* (2006), but allowing for a bandwidth matrix and not just one parameter. Smoothing parameters are selected by the normal reference rule on the transformed domain (cf., Nagler 2014, Section 3.4.4).

`TLL1`, `TLL2 (default)`
: The transformation local likelihood estimator of Geenens *et al.* (2014) with nearest-neighbor bandwidths. `TLL1` approximates the log-density linearly; `TLL2` by quadratic polynomials. Smoothing parameters are selected based on univariate least-squares cross-validation on the first principal component in the transformed domain (cf., Geenens *et al.* 2014, Section 4)

`TTCV`, `TTPI`
: Tapered transformation estimator of Wen and Wu (2015)[1]. Smoothing parameters are selected in the transformed domain by profile cross-validation (`TTCV`, cf., Wen and Wu 2015, Section 4.2) or plug-in minimization of the AMISE (`TTPI`, cf., Wen and Wu 2015, Section 4.1).

It is possible to specify the bandwidths manually using the `bw` argument of `kdecop`, although we recommend against it. If it is necessary to manually make an estimate more or less smooth, we advise to use the bandwidth multiplier argument `kdecop(..., mult = 1)`. Values larger then one will make the estimate more smooth; values less then one make the estimate less smooth.

3.2. Working with the estimated density: `(d/p/r)kdecop`

In analogy to the usual `(d/p/r)`-prefixes for distribution families in R, we provide `(d/p/r)`-versions for the `kdecop`-family. The functions `dkdecop` and `pkdecop` can be used to evaluate the density and *cdf* of a `kdecopula` object, respectively.

```
dkdecop(c(0.1, 0.2), kde.fit)  # estimated copula density
> [1] 1.690376
pkdecop(cbind(c(0.1, 0.9), c(0.1, 0.9)), kde.fit)  # corresponding copula cdf
> [1] 0.03254441 0.85154412
```

The `rkdecop` function simulates data from the estimated density. This can be done in two ways: a) using pseudo-random numbers based on `runif`, b) using quasi-random numbers based on `ghalton` from the **qrng** package (Hofert and Lemieux 2015).

```
pseudo <- rkdecop(500, kde.fit)
quasi  <- rkdecop(500, kde.fit, quasi = TRUE)
```

3.3. Visualization: the `plot` and `contour` generics

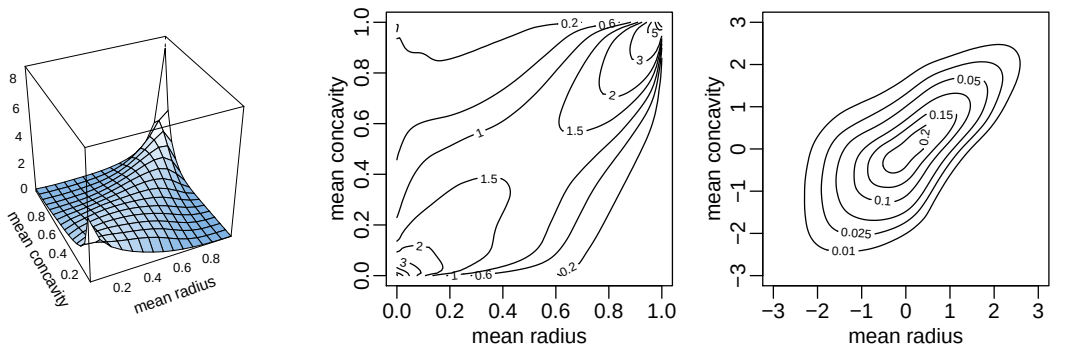
For many people, the most interesting feature is probably to make exploratory plots. There are three common ways to visualize a copula density: (a) a surface (or perspective) plot of the copula density, (b) a contour plot of the copula density, (c) a contour plot of the copula density when combined with standard normal margins.

```
plot(kde.fit)  # (a)
contour(kde.fit, margins = "unif")  # (b)
contour(kde.fit)  # (c)
```

The three plots are displayed in Figure 3. Optionally, further arguments can be passed to improve the aesthetics.

In the author's experience, the most useful plot is (c), the marginal normal contour plot. Let us elaborate briefly on its interpretation. If the true copula is the independence copula, the

[1]The implementation of the tapered transformation estimators was kindly provided by Kuangyu Wen.



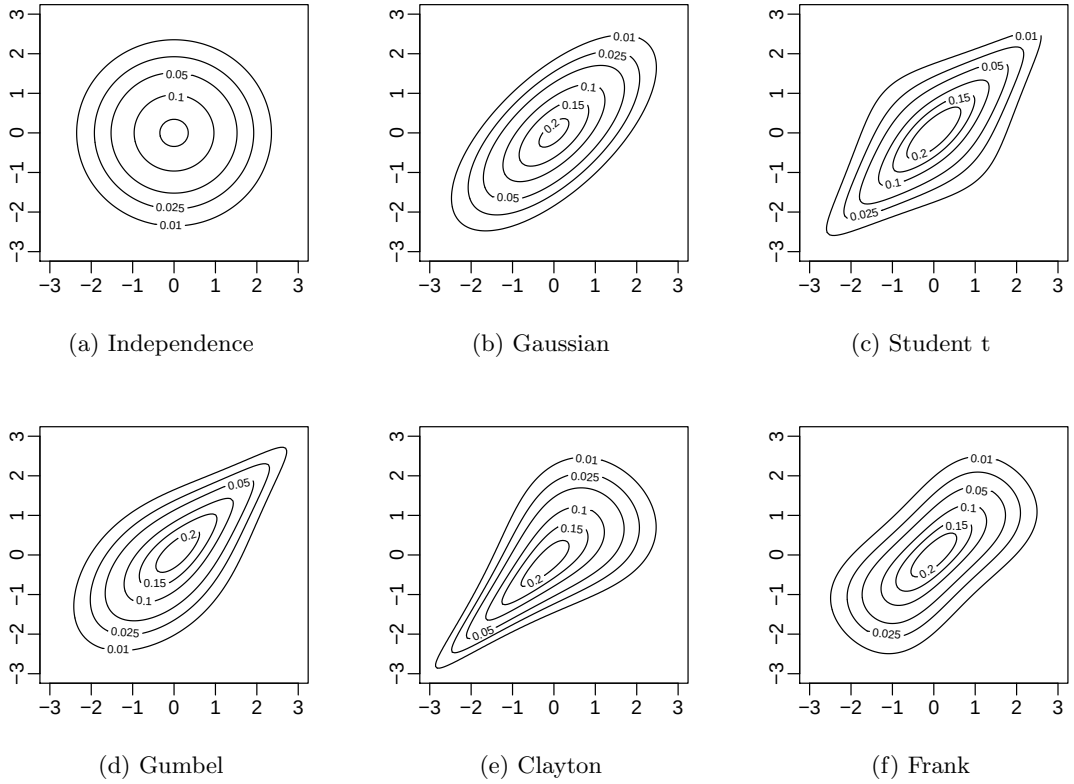
(a) surface plot (b) contour plot (c) marginal normal contour plot

Figure 3: Three ways to visualize a copula density.

contours are perfect circles (cf., Figure 4a). This is obviously not the case for the estimated density in Figure 3. Figure 4b shows a Gaussian copula with Kendall's τ set to the estimated τ from the data. A Gaussian copula combined with Gaussian margins results in a bivariate Gaussian density and its contours are ellipses. Since most statisticians are familiar with this kind of distribution it seems natural to use this as a benchmark when interpreting the marginal normal contour plot for other copulas. The next plot, Figure 4c, shows the Student t copula ($df = 3$). Here the contours look like a diamond due to the higher density values in the tails (i.e., the corners of the square). This reflects that — in contrast to the Gaussian copula — the Student t copula exhibits *tail dependence* (e.g., Joe 2014), a concept that is very important in the modeling of risks. In general, a spiky shape in the corners of the contours is an indication of tail dependence in the respective corner. This can be observed again in Figure 4d and Figure 4e where the Gumbel and Clayton copula are shown, respectively. The Gumbel copula is asymmetric and features upper tail dependence only. This is reflected by a spiky shape in the upper right corner and a flatter shape in the lower left corner. For the Clayton copula it iss the other way around. Finally the Frank copula has no tail dependence and has lighter tails than the Gaussian which corresponds to a more flat shape of the contours. Going back to the estimated density in Figure 3, we see a rather flat shape in the lower left corner and a more spiky shape in the upper right corner. This would indicate that there is no lower, but upper tail dependence. Hence, the Gumbel copula is the most appropriate fit choosing from the parametric families in Figure 4. However, we also observe some asymmetry w.r.t. to the main diagonal which is not featured by any of the parametric models under consideration. This is a sign that a nonparametric modeling approach could be useful.

4. Implementation based on spline interpolation

Typically, the evaluation of a kernel density estimate requires going back to the original data. As a result, the computational effort increases with the sample size. We avoid that issue by evaluating the actual density estimate only once on a fixed number of grid points. For further evaluations we use cubic spline interpolation between the values on this grid. This way, the density can be evaluated efficiently — independently of the sample size. It has the additional advantage that analytical expressions for integrals of the (interpolated) density



(a) Independence (b) Gaussian (c) Student t

(d) Gumbel (e) Clayton (f) Frank

Figure 4: Marginal normal contour plots for several parametric copula families.

estimate become available. We make use of that fact to implement a fast renormalization scheme that ensures that the the density estimate is close to a *bona fide* copula density.

4.1. Evaluating the estimate by cubic spline interpolation

Recall that the support of the copula density is the unit cube $[0,1]^2$. Let $m \in \mathbb{N}$ and define a finite set of points $p_j \in [0,1]$ such that $0 \leq p_1 < \cdots < p_m \leq 1$. Then, the set

$$\mathcal{P}_m = \{(u_j, v_k) : (j,k) \in \{1,\dots,m\}^2\} = \{(p_j, p_k) : (j,k) \in \{1,\dots,m\}^2\}$$

defines a symmetric grid on the unit cube. Cubic splines are piecewise cubic polynomials that can be used to approximate or interpolate some function between points on a grid. We will show how cubic spline interpolation can be used to approximate a copula density estimate. We explain in detail how a one-dimensional cubic spline interpolation is constructed when one of the coordinates is fixed. The two-dimensional case is a straightforward extension and only sketched.

The one-dimensional case

Let us first fix v_k and assume that the values of an estimate $\widehat{c}(\cdot, v_k) : [0,1] \to \mathbb{R}_+$ are available on the grid points $u_j, 1 \leq j \leq m$. We want to interpolate the function $\widehat{c}(\cdot, v_k)$ at another point

u^*, where $u_j < u^* < u_{j+1}$ for some $j \in \{2, \dots, m-2\}$. We define the interpolated curve segment $\tilde{c}^{j,j+1}(\cdot, v_k) : [u_j, u_{j+1}] \to \mathbb{R}$ as some cubic polynomial

$$\tilde{c}^{j,j+1}(u, v_k) = a_0^{j,j+1} + a_1^{j,j+1}(u - u_j) + a_2^{j,j+1}(u - u_j)^2 + a_3^{j,j+1}(u - u_j)^3.$$

A cubic polynomial defined on a closed interval is fully determined by its function values and first derivatives at the boundary points. Define $\tilde{c}_1^{j,j+1}$ as the partial derivative of $\tilde{c}^{j,j+1}$ w.r.t. its first argument. After some simple algebraic manipulations, we find that the coefficients of a cubic spline approximation can be written as

$$\begin{aligned}
a_0^{j,j+1} &= \tilde{c}^{j,j+1}(u_j, v_k), \\
a_1^{j,j+1} &= \tilde{c}_1^{j,j+1}(u_j, v_k), \\
a_2^{j,j+1} &= -3\tilde{c}^{j,j+1}(u_j, v_k) + 3\tilde{c}^{j,j+1}(u_{j+1}, v_k) - 2\tilde{c}_1^{j,j+1}(u_j, v_k) - \tilde{c}_1^{j,j+1}(u_{j+1}, v_k), \\
a_3^{j,j+1} &= 2\tilde{c}^{j,j+1}(u_j, v_k) - 2\tilde{c}^{j,j+1}(u_{j+1}, v_k) + \tilde{c}_1^{j,j+1}(u_j, v_k) + \tilde{c}_1^{j,j+1}(u_{j+1}, v_k),
\end{aligned}$$

Now we replace $\tilde{c}^{j,j+1}(u_j, v_k)$ and $\tilde{c}^{j,j+1}(u_{j+1}, v_k)$ by the known values $\widehat{c}(u_j, v_k)$ and $\widehat{c}(u_{j+1}, v_k)$. Similarly, we want to replace the derivatives $\tilde{c}_1^{j,j+1}(u_j, v_k)$ and $\tilde{c}_1^{j,j+1}(u_{j+1}, v_k)$ by $\widehat{c}_1(u_j, v_k)$ and $\widehat{c}_1(u_{j+1}, v_k)$. These are unknown, but can approximated by a finite difference scheme. We set

$$\begin{aligned}
\tilde{c}_1^{j,j+1}(u_j, v_k) &= \frac{\widehat{c}(u_j, v_k) - \widehat{c}(u_{j-1}, v_k)}{u_j - u_{j-1}} - \frac{\widehat{c}(u_{j+1}, v_k) - \widehat{c}(u_{j-1}, v_k)}{u_{j+1} - u_{j-1}} + \frac{\widehat{c}(u_{j+1}, v_k) - \widehat{c}(u_j, v_k)}{u_{j+1} - u_j}, \\
\tilde{c}_1^{j,j+1}(u_{j+1}, v_k) &= \frac{\widehat{c}(u_{j+1}, v_k) - \widehat{c}(u_j, v_k)}{u_{j+1} - u_j} - \frac{\widehat{c}(u_{j+2}, v_k) - \widehat{c}(u_j, v_k)}{u_{j+2} - u_j} + \frac{\widehat{c}(u_{j+2}, v_k) - \widehat{c}(u_{j+1}, v_k)}{u_{j+2} - u_{j+1}}.
\end{aligned}$$

Note that these can only be computed for $2 \leq j \leq m-2$, since four distinct values $\widehat{c}(u_{j+j^*}, v_k)$, $j^* = -1, 0, 1, 2$, show up in the above formulas. For fixed v_k and some $u \in [u_2, u_{k-1})$, the spline approximation $\tilde{c}(u, v_k)$ of the function $\widehat{c}(u, v_k)$ can then be written as

$$\tilde{c}(u, v_k) = \sum_{j=2}^{m-2} \tilde{c}^{j,j+1}(u, v_k) \mathbb{1}_{[u_j, u_{j+1})}(u).$$

We extended this to allow for the full range ($u \in [0, 1]$) by extrapolating the 'outer' two polynomials at the borders, i.e.

$$\tilde{c}(u, v_k) = \sum_{j=2}^{m-2} \tilde{c}^{j,j+1}(u, v_k) \mathbb{1}_{A_j}(u), \qquad \text{where} \quad A_j = \begin{cases} [0, u_3) & \text{for } j = 2, \\ [u_j, u_{j+1}) & \text{for } 3 \leq j \leq k-3, \\ [u_{m-2}, 1] & \text{for } j = m-2. \end{cases}$$

The advantage of cubic spline interpolation is that it is easy compute. In particular, the computational effort only depends on m, the number of knots. Additionally, the above approximation allows to write integrals as a sum of quartic polynomials which can be computed equally fast. This will prove advantageous in Section 4.2, where we use such integrals to renormalize the copula density estimates.

The two-dimensional case

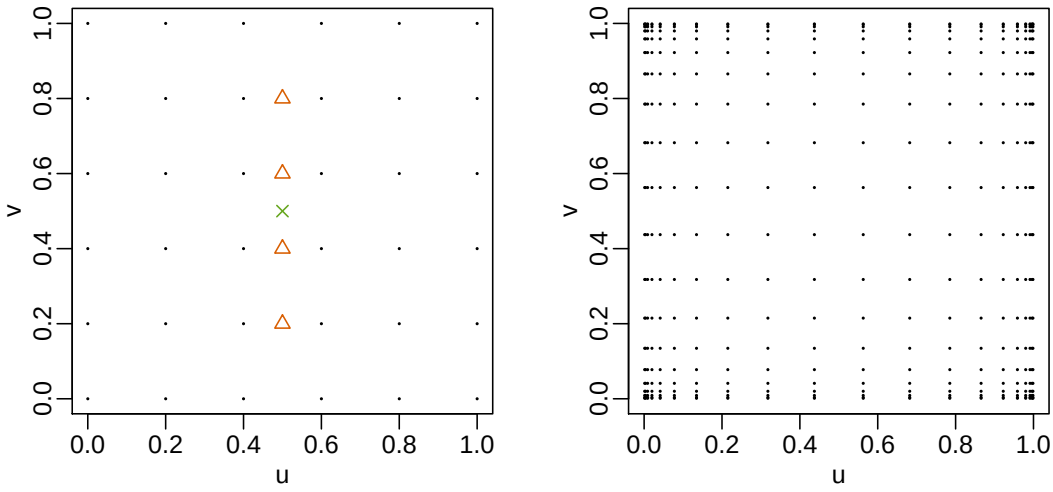


Figure 5: Left: Visualization of a two-dimensional interpolation as the sequence of two one-dimensional interpolations. Right: A grid with 20 knots that is equally spaced after transformation with the inverse Gaussian *cdf*.

Bivariate functions can be approximated similarly by a sequence of two one-dimensional interpolations. We will illustrate this by a small example and omit any further details (for more, cf., Habermann and Kindermann 2007). Figure 5a shows the unit cube with grid points $(u_j, v_k) = (j/5, k/5), j, k = 0, \dots, 5$, indicated as dots. Assume we know all functions values $\widehat{c}(u_j, v_k)$ on this grid and want to approximate the function at the point $(0.5, 0.5)$ indicated by a cross. We first do four one-dimensional (horizontal) interpolations $\tilde{c}(0.5, v_k), j \in \{1, 2, 3, 4\}$ (triangles). Note that all values that are required to calculate the spline coefficients are known. Another one-dimensional (vertical) interpolation based on the four new values $\tilde{c}(0.5, v_k)$ gives us the final interpolated value $\tilde{c}(0, 5, 0.5)$.

Choice of grid

In Figure 5a we showed a grid that has equal spacings between grid points. This seems natural, but in our context we found it more appropriate to use a grid that is equally spaced after a transformation by the inverse Gaussian *cdf*. Figure 5b depicts such a grid with 20 knots. It was constructed by placing 20 equidistant knots on the line segment $[-3, 3]$ and then applying the Gaussian *cdf* to them. The two-dimensional set-product of these 20 points yields the final two-dimensional grid.

We see that the grid points are more sparse in the center of the unit square and concentrate towards the boundaries and corners. This choice takes into account that for copula densities the areas near the corners are most important. In those areas, copula densities often explode while being rather flat in the center. This allows us to keep the approximation errors in the important areas small. A nice side-effect is that the marginal normal contour plots described in the last section can be visualized more nicely. The number of knots can be controlled by the `knots` argument of `kdecop` and defaults to 30. A smaller number reduces computation time, but comes at the cost of a larger approximation error.

4.2. Renormalization of the density estimate

We now introduce the idea of iterative renormalization of kernel copula density estimators. Recall that by the definition of a copula, the marginal densities have to be uniform, i.e.,

$$\int_0^1 c(u,s)ds = \int_0^1 c(s,u)ds = 1, \qquad \text{for all } u \in [0,1]. \tag{4}$$

This property is of particular importance, when other functionals of the density are of interest. For example, assume that we integrate the density estimate to obtain an estimate for the corresponding conditional *cdf*, $C(v|u) = \int_0^v c(u,s)ds$. This is a common task in vine copula models which gained a lot of popularity following the seminal paper of Aas, Czado, Frigessi, and Bakken (2009). If the estimated density does not satisfy the uniform margins property, the estimate of the conditional *cdf* may exceed unity which makes it problematic. The lack of uniform margins was mostly ignored in the literature, although kernel estimates usually do not satisfy the *uniform margins property* (4).

Now let $\widehat{c}(u,v)$ be a consistent kernel estimator of $c(u,v)$ for all $(u,v) \in [0,1]^2$. From Sklar's theorem for density functions (1), we know that dividing a bivariate density by its marginal densities yields a copula density. Hence, a simple way to ensure that an estimator satisfies (4) is to divide the initial estimate by $\int_0^1 \widehat{c}(u,s)ds \int_0^1 \widehat{c}(s,v)ds$. The renormalized estimator writes

$$\widehat{c}^*(u,v) = \frac{\widehat{c}(u,v)}{\int_0^1 \widehat{c}(u,s) \int_0^1 \widehat{c}(s,v)ds}. \tag{5}$$

For sophisticated kernel estimators — such as the beta kernel or local likelihood estimators — the two integrals have to be computed numerically. Conveniently, the spline approximations introduced in the previous section allow for fast computation of the integrals in (4).

The proposed renormalization procedure can be split into two steps.

1. Find a spline approximation of the initial estimate that is defined by its values on a finite grid.
2. Renormalize the approximated density values on this grid by dividing by the (approximated) marginal densities (see eq. (5)).

The resulting estimate will typically be closer to a *bona fide* copula density. However, the renormalization is only carried out on a finite number of grid points. Apart from that grid, the renormalized estimate typically still does not satisfy the uniform margins property. But we can simply repeat the two steps above until a satisfactory result is achieved. Our experience suggests that a very small number of iterations is sufficient. The number of iterations can be set by the `renorm.iter` argument of `kdecop` and defaults to three.

The renormalization will turn out to have two benefits: functionals of kernel estimates will show the desired behavior and, additionally, the estimates are more accurate (see Section 5). For the latter there is an intuitive interpretation. By ensuring that margins are uniform, we incorporate additional information about the true density. This reduces the set of plausible estimates and increases the probability of being 'close' to the true one.

5. Comparison of methods

Family	Indep.		Gaussian				Gumbel			
τ	0.0	0.0	0.3	0.3	0.7	0.7	0.3	0.3	0.7	0.7
n	400	2 000	400	2 000	400	2 000	400	2 000	400	2 000
MR	**0.02**	0.02	0.11	0.08	0.20	0.13	0.11	0.08	0.20	0.13
beta	**0.02**	**0.01**	0.11	0.07	0.18	0.12	0.11	0.07	0.18	0.12
T	0.09	0.06	0.10	0.06	0.11	0.07	0.10	0.06	0.11	0.07
TLL1	0.09	0.06	0.11	0.07	0.25	0.14	0.11	0.07	0.25	0.14
TLL2	0.07	0.03	**0.07**	**0.03**	**0.06**	**0.03**	**0.07**	**0.03**	**0.06**	**0.03**
TTPI	0.08	0.05	0.09	0.06	0.15	0.08	0.09	0.06	0.15	0.08
TTCV	0.05	0.02	0.07	0.04	0.11	0.06	**0.07**	0.04	0.11	0.06
np	0.21	0.17	0.20	0.16	0.23	0.16	0.20	0.16	0.23	0.16
ks	0.18	0.14	0.61	0.59	2.43	2.46	0.61	0.59	2.43	2.46
Bspl	0.03	**0.01**	0.09	0.06	0.19	0.14	0.09	0.06	0.19	0.14
Bern	0.03	**0.01**	0.08	0.05	0.36	0.31	0.08	0.05	0.36	0.31

Table 1: Simulation results for various specifications of copula family, Kendall's τ, and sample size (n). All methods above the dotted line are included in **kdecopula** package.

We compare all estimators implemented in this package in a simulation study. Additionally, we include results for other nonparametric copula density estimators available in R.

We consider the following estimators:

- MR, beta, T, TLL1, TLL2, TTPI, TTCV: These estimators are provided the `kdecop` function presented in this paper (cf., Section 2).
- np: The kernel estimator provided by the `npcopula` function in the **np** package (Hayfield and Racine 2008).
- ks: The kernel estimator provided by the `kcopula.de` function in the **ks** package (Duong 2014).
- Bspl, Bern: The penalized B-spline and Bernstein polynomial estimators provided by the `paircopula` function in the **penDvine** package (Schellhase 2015).

Default settings are used for all implementations. In particular, smoothing parameters are selected automatically.

As a performance measure, we use the *integrated absolute error (IAE)*

$$\text{IAE}\big[\widehat{c}(u,v)\big] = \int_{[0,1]}\int_{[0,1]} \big|\widehat{c}(u,v) - c(u,v)\big| du dv,$$

where we estimate the integrals as the mean over the grid $(j/101, k/101)$, $j,k = 1,\dots,100$. Table 1 shows the mean of the IAE over 1 000 replication for various scenarios. We consider three copula types (Independence, Gaussian, Gumbel) and two sample sizes ($n = 400, n = 2\,000$). For the Gaussian and Gumbel copulas we have scenarios with weak and strong dependence (Kendall's τ of 0.3 and 0.7). All methods above the dotted line are included in **kdecopula** package; the methods below are from other packages available on CRAN. We observe that TLL2 is the top performer in all but the independence scenario. In this case,

the beta kernel estimator is preferable. Overall, the estimators implemented in **kdecopula** perform very well compared with existing methods from other packages. Only the B-spline and Bernstein polynomial estimators can catch up in scenarios with low dependence.

In Section 4.2 we have claimed that the renormalization algorithm implemented in this package improves the performance of the estimators. The results presented in Table 1 used default settings, i.e., three iterations of the algorithm. Table 2 show the relative reduction of IAE compared to non-normalized estimators. We observe that the performance could be improved for all estimators in all scenarios. The gain ranges between 9% and 62% with a mean of over 30%. These numbers are quite remarkable and contributed significantly to the good performance observed in Table 1.

Family	Indep.		Gaussian				Gumbel			
τ	0.0	0.0	0.3	0.3	0.7	0.7	0.3	0.3	0.7	0.7
n	400	2 000	400	2 000	400	2 000	400	2 000	400	2 000
MR	54%	48%	15%	9%	15%	10%	15%	9%	15%	10%
beta	61%	62%	22%	16%	18%	13%	22%	16%	18%	13%
T	41%	36%	38%	32%	29%	25%	38%	32%	29%	25%
TLL1	45%	42%	40%	40%	22%	29%	40%	40%	22%	29%
TLL2	39%	37%	41%	39%	49%	47%	41%	39%	49%	47%
TTPI	32%	27%	30%	25%	26%	24%	30%	25%	26%	24%
TTCV	43%	42%	33%	29%	30%	28%	33%	29%	30%	28%

Table 2: The effect of renormalization: percentage numbers indicate by how much the IAE could be improved after three iterations of the renormalization algorithm.

6. Summary

We have described the R package **kdecopula** which implements several cutting-edge kernel estimation techinques for copula densities. The package allows for automatic selection of the smoothing parameter and resampling. Several plotting options make it particularly useful for the exploratory analysis of copula data. Its abilities have been illustrated by small code examples.

The implementation utilizes spline interpolation for fast evaluation and renormalization of the density estimates. Simulations show that the implementations in this package perform best among available methods for nonparametric copula density estimation. A contributing factor for the good performance is the rernormalization of the estimators which was shown to notably improve the accuracy.

References

Aas K, Czado C, Frigessi A, Bakken H (2009). "Pair-copula Constructions of Multiple Dependence." *Insurance: Mathematics and Economics*, **44**(2), 182–198.

Aitken CGG, Lucy D (2004). "Evaluation of Trace Evidence in the Form of Multivariate Data."

Journal of the Royal Statistical Society: Series C (Applied Statistics), **53**(1), 109–122. ISSN 1467-9876.

Charpentier A, Fermanian JD, Scaillet O (2006). "The Estimation of Copulas: Theory and Practice." In J Rank (ed.), *Copulas: From theory to application in finance*. Risk Books.

Cherubini U, Luciano E, Vecchiato W (2004). *Copula Methods in Finance*. Wiley finance series. Wiley.

Devroye L, Györfi L (1985). *Nonparametric Density Estimation: The L_1 View*. John Wiley and Sons.

Duong T (2014). ***ks**: Kernel Smoothing*. R package version 1.9.3, URL http://CRAN.R-project.org/package=ks.

Elidan G (2013). "Copulas in Machine Learning." In P Jaworski, F Durante, WK Härdle (eds.), *Copulae in Mathematical and Quantitative Finance*, volume 213 of *Lecture Notes in Statistics*, pp. 39–60. Springer Berlin Heidelberg.

Geenens G, Charpentier A, Paindaveine D (2014). "Probit Transformation for Nonparametric Kernel Estimation of the Copula Density." *arXiv:1404.4414 [stat.ME]*.

Gijbels I, Mielniczuk J (1990). "Estimating the Density of a Copula Function." *Communications in Statistics - Theory and Methods*, **19**(2), 445–464.

Habermann C, Kindermann F (2007). "Multidimensional Spline Interpolation: Theory and Applications." *Computational Economics*, **30**(2), 153–169.

Hayfield T, Racine JS (2008). "Nonparametric Econometrics: The **np** Package." *Journal of Statistical Software*, **27**(5). URL http://www.jstatsoft.org/v27/i05/.

Hofert M, Lemieux C (2015). ***qrng**: (Randomized) Quasi-Random Number Generators*. R package version 0.0-2, URL https://CRAN.R-project.org/package=qrng.

Joe H (2014). *Dependence Modeling with Copulas*. Chapman & Hall/CRC.

Kie JG, Matthiopoulos J, Fieberg J, Powell RA, Cagnacci F, Mitchell MS, Gaillard JM, Moorcroft PR (2010). "The home-range concept: are traditional estimators still relevant with modern telemetry technology?" *Philosophical Transactions of the Royal Society B: Biological Sciences*, **365**(1550), 2221–2231. doi:10.1098/rstb.2010.0093.

Nagler T (2014). *Kernel Methods for Vine Copula Estimation*. Master's thesis, TU München.

Salvadori G, De Michele C (2007). "On the Use of Copulas in Hydrology: Theory and Practice." *Journal of Hydrologic Engineering*, **12**(4), 369–380.

Schellhase C (2014). ***pencopula**: Flexible Copula Density Estimation with Penalized Hierarchical B-Splines*. R package version 0.3.5, URL https://CRAN.R-project.org/package=pencopula.

Schellhase C (2015). ***penDvine**: Flexible Pair-Copula Estimation in D-Vines using Bivariate Penalized Splines*. R package version 0.2.4, URL https://CRAN.R-project.org/package=penDvine.

Schepsmeier U, Stoeber J, Brechmann EC, Graeler B, Nagler T, Erhardt T (2015). ***VineCopula**: Statistical Inference of Vine Copulas*. R package version 1.6, URL https://CRAN.R-project.org/package=VineCopula.

Sklar A (1959). *Fonctions de Répartition à n Dimensions et Leurs Marges*. Université Paris 8.

Wen K, Wu X (2015). "Transformation-Kernel Estimation of the Copula Density." URL http://agecon2.tamu.edu/people/faculty/wu-ximing/agecon2/public/copula.pdf.

Affiliation:

Thomas Nagler
Technische Universität München
Zentrum Mathematik
Lehrstuhl für Mathematische Statistik
Boltzmannstraße 3 85748 Garching