Enjoy the Joy of Copulas

Jun Yan

Department of Statistics and Actuarial Science, University of Iowa, 241 Schaeffer Hall, Iowa City, IA 52242, USA

Abstract

Copulas have become a popular tool in multivariate modeling successfully applied in many fields. A good open-source implementation of copulas is much needed for more practitioners to enjoy the joy of copulas. This article presents the design, features, and some implementation details of the R package `copula`. The package provides a carefully designed and easily extensible platform for multivariate modeling with copulas in R. S4 classes for most frequently used elliptical copulas and Archimedean copulas are implemented, with methods for density/distribution evaluation, random number generation, and graphical display. Fitting copula-based models with maximum likelihood method is provided as template examples. With the classes and methods in the package, users can easily extend the package can be easily extended by user-defined copulas and margins to solve problems.

Key words: copula, multivariate analysis, R

1 Introduction

Copulas have become a popular multivariate modeling tool in many fields where the multivariate dependence is of great interest and the usual multivariate normality is in question. In actuarial science, copulas are used in modeling dependent mortality and losses (Frees et al., 1996; Frees and Valdez, 1998; Frees and Wang, 2005). In finance, copulas are used in asset allocation, credit scoring, default risk modeling, derivative pricing, and risk management (Bouyè et al., 2000; Embrechts et al., 2003; Cherubini et al., 2004). In biomedical studies, copulas are used in modeling correlated event times and competing risks (Wang and Wells, 2000; Escarela and Carrière, 2003). In engineering, copulas are used in multivariate process control and hydrological modeling (Yan, 2006b; Genest and Favre, 2006).

Email address: `jyan@stat.uiowa.edu` (Jun Yan).

A copula is a multivariate distribution whose marginals are all uniform over $(0, 1)$. For a p-dimensional vector U on the unit cube, a copula C is

$$C(u_1, \ldots, u_p) = \Pr(U_1 \leq u_1, \ldots, U_p \leq u_p). \tag{1}$$

Combined with the fact that any continuous random variable can be transformed to be uniform over $(0, 1)$ by its probability integral transformation, copulas can be used to provide multivariate dependence structure separately from the marginal distributions. Copulas first appeared in the probability metrics literature. Let F be a p-dimensional distribution function with margins $F_1, \ldots, F_p$. Sklar (1959) first showed that there exists a p-dimensional copula C such that for all x in the domain of F,

$$F(x_1, \ldots, x_p) = C\{F_1(x_1), \ldots, F_p(x_p)\}. \tag{2}$$

The last two decades, particularly the last 10 years, witnessed the spread of copulas in statistical modeling. Joe (1997) and Nelsen (1999) are the two comprehensive treatments on this topic. A frequently cited and widely accessible reference is Genest and MacKay (1986), titled "The Joy of Copulas", which gives properties of an important family of copulas, Archimedean copulas; see Section 2.

For the joy of copulas to be enjoyable by everyone in need, software implementation is important. Unfortunately, there are very few software packages available for copula-based modeling. One of the exceptions is the `finmetrics` module (Insightful Corporation, 2002) of `Splus` (Insightful Corporation, 2005). For an array of commonly used copulas, the `finmetrics` module provides functions to evaluate their density and distribution, generate random numbers from them, and fit them for given data. These functionalities, however, are limited because only bivariate copulas are implemented. Furthermore, the software is commercial. It is desirable to have an open source platform for the development of copula methods and applications.

"`R` is a free software environment for statistical computing and graphics" (R Development Core Team, 2006a). It runs on all platforms including Unix/Linux, Windows, and MacOS. Cutting-edge statistical developments are easily incorporated into R by the mechanism of contributed packages with quality assurance (R Development Core Team, 2006b). It provides excellent graphics and interfaces easily with lower level compiled code such as `C/C++` or `FORTRAN`. An active developer-user interaction is available through the R-help mailing list. Hundreds of contributed packages are available and many existing functionalities, for example, the density and distribution functions of multivariate normal and t distributions, can be used for copulas. Therefore, it is a natural choice to write an `R` package for copulas.

The package `copula` (Yan, 2006a) is designed using the object-oriented features of the S language (Chambers, 1998). It is publicly available at the Comprehensive R Archive Network (CRAN, http://www.r-project.org). The new S4-style classes are created for elliptical copulas, and Archimedean copulas with arbitrary dimension. For each copula family, methods of density, distribution, and random number generator are implemented. For visualization purpose, methods of contour and perspective plots are provided for bivariate copulas. Maximum likelihood method for fitting copula-based models is also available and can be easily extended.

The rest of the article is organized as follows. Section 2 briefly presents the S4-style classes defined in the package. Section 3 describe methods for copula classes, including density, distribution, random number generator, and graphics. Section 4 discusses how to fit copula-based models to data. Section5 concludes.

In the sequel, all R code demonstration assumes that the package has been loaded:

```
> library(copula)
```

The required packages `mvtnorm`(Genz et al., 2005), `sn` (Azzalini, 2005), and `scatterplot3d` (Ligges and Mächler, 2003), if not already loaded, will too be loaded by this call. To make all the illustrations reproducible, we set the random seed:

```
> set.seed(1)
```

2 Classes

Two main classes are defined in the `copula` package: `copula` and `mvdc`. The `copula` class is for defining copulas, while the `mvdc` class is for defining multivariate distributions via copula.

2.1 The `copula` class

Two most frequently used copula families are elliptical copulas and Archimedean copulas. The `copula` package has implemented virtual classes `ellipCopula` and `archmCopula`, both extending the virtual class `copula`. These virtual classes are designed to provide a flexible way to associate actual classes, that share some properties but have different representations (Chambers, 1998, p.292).

An elliptical copula is the copula corresponding to an elliptical distribution by the Sklar's theorem. General discussion about elliptical distributions can be found in Fang et al. (1990). Let F be the multivariate CDF of an elliptical distribution. Let F_i be CDF of the ith margin and F_i^{-1} be its inverse function (quantile function), $i = 1, \ldots, p$. The elliptical copula determined by F is

$$C(u_1, \ldots, u_p) = F[F_1^{-1}(u_1), \ldots, F_p^{-1}(u_p)]. \tag{3}$$

Elliptical copulas are have become very popular in finance and risk management their easy implementation. Convenience in obtaining conditional distributions is another advantage in using them for predicting (Frees and Wang, 2005).

Actual elliptical copula classes implemented in the package are `normalCopula` for normal copula `tCopula` for t-copula, specified by multivariate normal and multivariate t distribution. Both copulas has a dispersion matrix, inherited from the elliptical distributions, and t-copula has one more parameter, the degrees of freedom (`df`). Since copulas are invariant to monotonic transformation of the margins, the standardized dispersion matrix, or correlation matrix, determines the dependence structure. Commonly used dispersion structures are implemented: autoregressive of order 1 (`ar1`), exchangeable (`ex`), Toeplitz (`toep`), and unstructured (`un`). The corresponding correlation matrices are, for example, in the case of dimension $p = 3$,

$$\begin{pmatrix} 1 & \rho_1 & \rho_1^2 \\ \rho_1 & 1 & \rho_1 \\ \rho_1^2 & \rho_1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & \rho_1 & \rho_1 \\ \rho_1 & 1 & \rho_1 \\ \rho_1 & \rho_1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & \rho_1 & \rho_2 \\ \rho_1 & 1 & \rho_1 \\ \rho_2 & \rho_1 & 1 \end{pmatrix}, \text{ and } \begin{pmatrix} 1 & \rho_1 & \rho_2 \\ \rho_1 & 1 & \rho_3 \\ \rho_2 & \rho_3 & 1 \end{pmatrix}, \tag{4}$$

where ρ_j's are dispersion parameters. The following code creates a `normal-Copula` object and a `tCopula` object, with self-explanatory arguments:

```
> myCop.norm <- ellipCopula(family = "normal", dim = 3, dispstr = "ex",
     param = 0.4)
> myCop.t <- ellipCopula(family = "t", dim = 3, dispstr = "toep",
     param = c(0.8, 0.5), df = 8)
```

These objects can be used to apply methods defined in Section 3.

An Archimedean copula is constructed through a generator φ as

$$C(u_1, \ldots, u_p) = \varphi^{-1}\left\{\varphi(u_1) + \cdots + \varphi(u_p)\right\}, \tag{5}$$

where φ^{-1} is the inverse of the generator φ. In order for (5) to be a copula, the generator needs to be a complete monotonic function (see, for example, Nelsen, 1999, Theorem 4.6.2). A generator uniquely (up to a scalar multiple) deter-

Table 1
Summary of three One-Parameter (α) Archimedean Copulas for $p > 2$

Family	Parameter Space	Generator $\varphi(t)$	Generator Inverse $\varphi^{-1}(s)$	Frailty Distribution
Clayton (1978)	$\alpha \geq 0$	$t^{-\alpha} - 1$	$(1+s)^{-1/\alpha}$	Gamma
Frank (1979)	$\alpha \geq 0$	$\ln \frac{e^{\alpha t} - 1}{e^{\alpha} - 1}$	$\alpha^{-1} \ln (1 + e^{s}(e^{\alpha} - 1))$	Log series
Gumbel (1960)	$\alpha \geq 1$	$(-\ln t)^{\alpha}$	$\exp(-s^{1/\alpha})$	Positive stable

mines an Archimedean copula. Details of generators for various Archimedean copulas can be found in Nelsen (1999).

Implemented Archimedean copula classes in the package are commonly used one-parameter families, such as `calytonCopula` for Clayton copula (Clayton, 1978), `frankCopula` for Frank copula (Frank, 1979), and `gumbelCopula` for Gumbel copula (Gumbel, 1960). Constructors of these copulas are available. For example:

```
> myCop.clayton <- archmCopula(family = "clayton", dim = 3, param = 2)
```

It is worth noting that Archimedean copulas with dimension 3 or higher only allows positive association. Negative association is allowed for bivariate Archimedean copulas. The three one-parameter multivariate Archimedean copulas ($p > 2$) implemented in the package are summarized in Table 1. The parameter value at the boundary of parameter space gives the independent copula after taking the limit. The generator inverse and frailty distribution are used in random number generation.

2.2 The `mvdc` class

The `mvdc` class is designed to construct multivariate distributions with given margins using copulas as in (2). This class is an actual class. It has three major components: `copula` specifies the copula C in (2); `margins` specifies the names of the marginal distributions $F_1, \ldots, F_p$; and `paramMargins` is a list of list, each specifying the parameter values of the corresponding margin.

The following code creates a `mvdc` object which represents a trivariate distribution with standard normal margins and Clayton copula:

```
> myMvd <- mvdc(copula = myCop.clayton, margins = c("norm", "norm",
     "norm"), paramMargins = list(list(mean = 0, sd = 2), list(mean = 0,
     sd = 1), list(mean = 0, sd = 2)))
```

`R` provides a comprehensive set of statistical distributions R Development Core Team (2006a), such as `norm, t, gamma, lnorm, weibull`, etc. They can be used to specify the margins. User-defined distributions can be used as long as the PDF, CDF, and quantile function of the distribution, with prefix `d`, `p`, and `q`, are available. For instance, if functions `dfancy`, `pfancy`, and `qfancy` have been supplied, then distribution `fancy` can be used in the margins. Of note is that these functions should be vectorized.

3 Methods

3.1 Distribution and Density

The method functions for distribution and density for a `copula` object are `pcopula` and `dcopula`

For an elliptical copula, the distribution is (3). Evaluation of (3) needs implementation of the joint CDF of the elliptical distribution and univariate quantile functions for each margin. Differentiating (3) gives the density of an elliptical copula

$$c(u_1, \ldots, u_p) = \frac{f[F_1^{-1}(u_1), \ldots, F_p^{-1}(u_p)]}{\prod_{i=1}^p f_i[F_i^{-1}(u_i)]}, \tag{6}$$

where f is the joint PDF of the elliptical distribution, and $f_1, \ldots, f_p$ are marginal density functions. For example, after some algebra, the density of a Gaussian copula with dispersion matrix Σ is (Song, 2000)

$$c(u_1, \ldots, u_p|\Sigma) = |\Sigma|^{-1/2} \exp\left\{\frac{1}{2}\boldsymbol{c}^\top (I_p - \Sigma^{-1})\boldsymbol{c}\right\}, \tag{7}$$

where $\boldsymbol{c} = (q_1, \ldots, q_p)^\top$ with $q_i = \Phi^{-1}(u_i)$ for $i = 1, \ldots, p$, and Φ is the CDF of $N(0,1)$. Evaluation of (6) needs implementation of f and $f_1, \ldots, f_p$ in addition to the univariate quantiles $F_1^{-1}, \ldots, F_p^{-1}$. Fortunately, the R package `mvtnorm` and `sn` provide F and f for multivariate normal and multivariate t. Other functions are available in the R base.

For an Archimedean copula, the distribution and density both depend on the generator function and its inverse function. These functions are defined for each Archimedean copula. The density of (5) is to be obtained by differentiation, which, in many situations, can be very tedious. Fortunately, R has some simple symbolic differentiation facility. With this advantage, symbolic expressions of CDF and PDF can be obtained and evaluated in fast vector operation. For example:

```
> myCop.clayton@exprdist$cdf

(1 + (u1^(-alpha) - 1 + u2^(-alpha) - 1 + u3^(-alpha) - 1))^(-1/alpha)

> myCop.clayton@exprdist$pdf

(1 + (u1^(-alpha) - 1 + u2^(-alpha) - 1 + u3^(-alpha) - 1))^((((-1/alpha) -
    1) - 1) - 1) * ((((-1/alpha) - 1) - 1) * (u3^((-alpha) -
    1) * (-alpha))) * (((-1/alpha) - 1) * (u2^((-alpha) - 1) *
    (-alpha))) * ((-1/alpha) * (u1^((-alpha) - 1) * (-alpha)))
```

These expressions can be ported into other programming languages. As the dimension p increases, the memory needed to do the symbolic differentiation to obtain PDF expression increases dramatically. Since the PDF expression is used in likelihood evaluation, a more elegant solution is needed. One possibility is to compute once for a range of p values, for example, $p < 20$, and then store it statically, as opposed to computing them on the fly.

The method functions for distribution and density for a `mvdc` are `pmvdc` and `dmvdc` The distribution function is defined in (2). The density function is, by differentiating (2),

$$f(x_1, \ldots, x_p) = c[F_1(x_1), \ldots, F_p(x_p)] \prod_{i=1}^{p} f_i(x_i) \tag{8}$$

where c is the density of C, and $f_1, \ldots, f_p$ are marginal densities.

Example code to evaluate distribution and density will be given in the next subsection.

3.2 Random Number Generator

The random number generator method is `rcopula` for an `copula` object and `rmvdc` for an `mvdc` object.

The random number generator of an elliptical copula is straightforward given a random number generator of the corresponding elliptical distribution. The Sklar's theorem implies that random numbers from a copula can be generated by transforming each margin of random numbers from a multivariate distribution with its probability integral transformation. The `copula` package provides generators for normal copula and t-copula using the random number generators for multivariate normal and multivariate t in package `mvtnorm`.

Generating variables form a general copula can be done by iterative conditioning (Bouyè et al., 2000). For some commonly used Archimedean copulas with $p > 2$, however, fast algorithm exits when the inverse generator function φ^{-1} is known to be the Laplace transform of some positive random variable (Marshall and Olkin, 1988; Frees and Valdez, 1998). This positive random variable is often referred to as frailty. Let γ be a realization of the frailty. Let $v_1, \ldots, v_p$ be independent realizations of uniform variables over $(0, 1)$. Then $u_i = \varphi^{-1}(-\gamma^{-1} \log v_i)$, $i = 1, \ldots, p$, is a realization from the Archimedean copula with generator φ. This algorithm is very easy to implement and fast, given that a random number generator of the frailty is available. It is known that the frailty distribution for Clayton, Frank, and Gumbel copulas are gamma, log-series, and positive stable, respectively; see Table 1. Gamma variable generator is available in the R base. Algorithms for generating positive stable and log series variables can be found in Chambers et al. (1976) and Kemp (1981), respectively. In particular, the `copula` uses a Fortran implementation of Nolan (2006), which is a revised version of Chambers et al. (1976), to generate positive stable variables. From the compound construction, this algorithm only allows positive association.

For bivariate Archimedean copulas ($p = 2$), negative association is allowed. Random number generator for bivariate Archimedean copulas are therefore separately implemented.

The following code illustrates the random number generation and evaluation of distribution and density for the `copula` object `myCop.t` created in Section 2:

```
> u <- rcopula(myCop.t, 4)
> u

          [,1]      [,2]      [,3]
[1,] 0.3508325 0.6165205 0.7459244
[2,] 0.3912433 0.2189641 0.2556491
[3,] 0.3925507 0.7579099 0.9157623
[4,] 0.9822296 0.9611676 0.8896553

> cbind(dcopula(myCop.t, u), pcopula(myCop.t, u))

          [,1]      [,2]
[1,]  2.265954 0.3081520
[2,]  3.493735 0.1359238
[3,]  1.878803 0.3777087
[4,] 31.481423 0.8771844
```

To generate random numbers from a `mvdc` object, one only needs to apply the quantile function to random numbers of the specified copula on each margin. The following code illustrates the random number generation and evaluation of distribution and density for the `mvdc` object `mymvd` created in Section 2:

```
> x <- rmvdc(myMvd, 4)
```

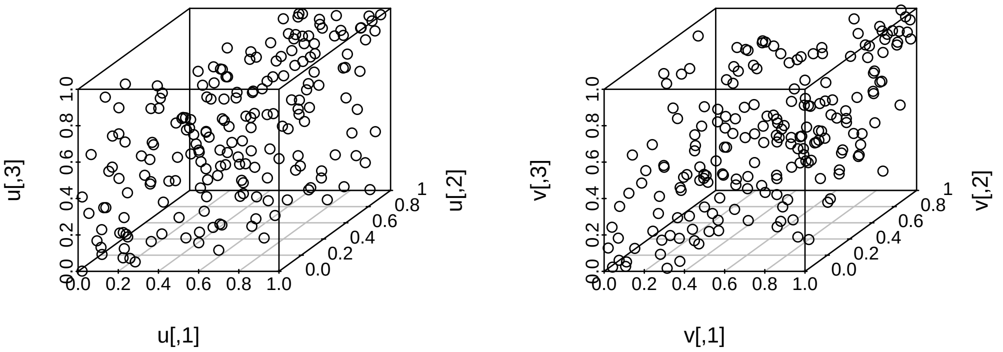

Fig. 1. Scatter plots of random numbers from a normal copula and a *t*-copula.

```
> x

            [,1]      [,2]       [,3]
[1,]  1.2384606 0.2724823  2.8546293
[2,] -0.3988947 1.3611864  0.6201294
[3,]  0.9119850 0.6937386  0.1522380
[4,] -0.8224510 0.4412545 -0.5282348

> cbind(dmvdc(myMvd, x), pmvdc(myMvd, x))

             [,1]      [,2]
[1,] 0.009682354 0.5164269
[2,] 0.005771651 0.3668884
[3,] 0.021743229 0.4266208
[4,] 0.018830608 0.2562096
```

3.3 Graphics

Graphics are important tools for in illustrating and presenting the results copula-based modeling. The 3D scatter plot from the package `scatterplot3d` can be used to show scatters. For example, the following code plots 200 random points from a trivariate normal copula and a trivariate *t*-copula in Figure 1.

```
> par(mfrow = c(1, 2), mar = c(2, 2, 1, 1), oma = c(1, 1, 0, 0),
     mgp = c(2, 1, 0))
> u <- rcopula(myCop.norm, 200)
> scatterplot3d(u)
> v <- rcopula(myCop.norm, 200)
> scatterplot3d(v)
```

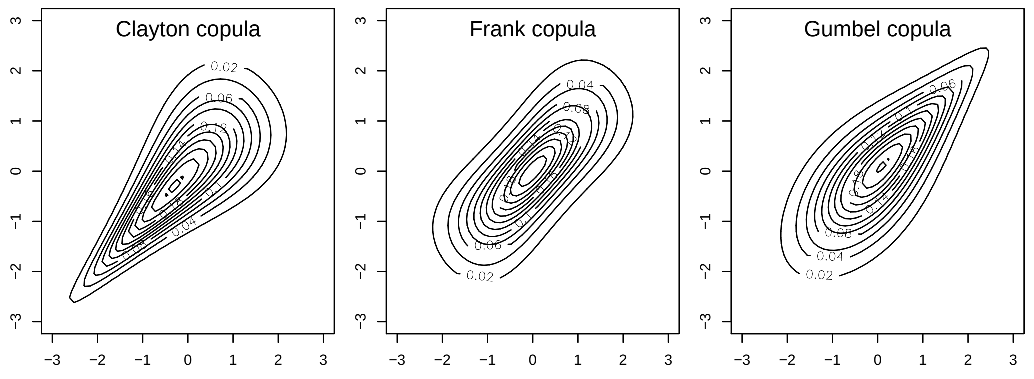

Fig. 2. Contour plots

For both `copula` and `mvdc` objects, the `copula` package has implemented methods to draw perspective and contour plot for density and distribution. These method functions are `persp` and `contour`. We illustrate the usage of `contour` for `mvdc` objects. The following code plots the density contours of bivariate distributions defined with Clayton, Frank, and Gumbel copulas, all with both margins being standard normal:

```
> myMvd1 <- mvdc(copula = archmCopula(family = "clayton", param = 2),
     margins = c("norm", "norm"), paramMargins = list(list(mean = 0,
         sd = 1), list(mean = 0, sd = 1)))
> myMvd2 <- mvdc(copula = archmCopula(family = "frank", param = 5.736),
     margins = c("norm", "norm"), paramMargins = list(list(mean = 0,
         sd = 1), list(mean = 0, sd = 1)))
> myMvd3 <- mvdc(copula = archmCopula(family = "gumbel", param = 2),
     margins = c("norm", "norm"), paramMargins = list(list(mean = 0,
         sd = 1), list(mean = 0, sd = 1)))
> par(mfrow = c(1, 3), mar = c(2, 2, 1, 1), oma = c(1, 1, 0, 0),
     mgp = c(2, 1, 0))
> contour(myMvd1, dmvdc, xlim = c(-3, 3), ylim = c(-3, 3))
> contour(myMvd2, dmvdc, xlim = c(-3, 3), ylim = c(-3, 3))
> contour(myMvd3, dmvdc, xlim = c(-3, 3), ylim = c(-3, 3))
```

The contour plots are shown in Figure 2. Note that the parameters of the copulas are chosen such that the Kendall's τ for all three distributions are 0.5.

The `persp` method can be called similarly. The first argument in these two calls are the signature that determines which method to call. It should be either a `copula` object or a `mvdc` object. The second argument specifies the function for which the plots are to drawn, that is, PDF or CDF.

4 Fit a Copula Model

With density functions for `copula` and `mvdc` objects available, one can easily fit copula-based models with the maximum likelihood method. The package provides functions `loglikCopula` and `loglikMvdc` to evaluate the loglikelihood of the data under the specified copula model. These functions can be passed to an optimizer to obtain the maximum likelihood estimate. The package provides functions `fitCopula` and `fitMvdc` to carry out the estimation and report the results.

Suppose that we observe n indepent realizations from a multivariate distribution, $\{(X_{i1}, \ldots, X_{ip})^\top : i = 1, \ldots, n\}$. Suppose that the multivariate distribution is specified by p margins with CDF F_i and PDF f_i, $i = 1, \ldots, p$, and a copula with density c. Let β be the vector of marginal parameters and α be the vector of copula parameters. The parameter vector to be estimated is $\theta = (\beta^\top, \alpha^\top)^\top$. The loglikelihood function is

$$l(\theta) = \sum_{i=1}^{n} \log c\left\{F_1(X_{i1}; \beta), \ldots, F_p(X_{ip}; \beta); \alpha\right\} + \sum_{i=1}^{n}\sum_{j=1}^{p} \log f_i(X_{ij}; \beta). \tag{9}$$

The ML estimator of θ is

$$\hat{\theta}_{\mathrm{ML}} = \arg\max_{\theta \in \Theta} l(\theta),$$

where Θ is the parameter space.

To illustrate, we generate a sample from a bivariate distribution with gamma margins and a normal copula:

```
> myMvd <- mvdc(copula = ellipCopula(family = "normal", param = 0.5),
      margins = c("gamma", "gamma"), paramMargins = list(list(shape = 2,
          scale = 1), list(shape = 3, scale = 2)))
> n <- 200
> dat <- rmvdc(myMvd, n)
```

The parameters to be estimated consist of marginal parameters $\beta = (2, 1, 3, 2)^\top$ and copula parameter $\alpha = 0.5$. The loglikelihood at the true parameter value is:

```
> loglikMvdc(c(2, 1, 3, 2, 0.5), dat, myMvd)
```

```
[1] -781.1641
```

To obtain $\hat{\theta}_{\mathrm{ML}}$, one implements the loglikelihood function $l(\theta)$ and feed it to an optimizer. The function `fitMvdc` is a wrapper to the optimization routine optim in R. An initial search point is needed for optim. Simpel method of moments estimate will serve as initial point.

```
> mm <- apply(dat, 2, mean)
> vv <- apply(dat, 2, var)
> b1.0 <- c(mm[1]^2/vv[1], vv[1]/mm[1])
> b2.0 <- c(mm[2]^2/vv[2], vv[2]/mm[2])
> a.0 <- sin(cor(dat[, 1], dat[, 2], method = "kendall") * pi/2)
> start <- c(b1.0, b2.0, a.0)
> fit <- fitMvdc(dat, myMvd, start = start, optim.control = list(trace = TRUE,
     maxit = 2000))
```

The first three arguments of function fitMvdc are the data, the mvdc object, and a starting value. Control parameters to the optimizing routine can passed in through argument optim.control. The starting values here are arbitarily chosen. In real problems, the starting values of marginal parameters can be chosen by fitting each margin separately. The result of the estimation is summarized as:

```
> fit
```

```
The ML estimation is based on  200  observations.
Margin  1 :
         Estimate Std. Error
m1.shape 1.830479  0.1689802
m1.scale 1.037388  0.1100475
Margin  2 :
         Estimate Std. Error
m2.shape 3.515646  0.3362403
m2.scale 1.628037  0.1672812
Copula:
      Estimate Std. Error
rho.1 0.419909 0.05820196
The maximized loglikelihood is  -777.327
The convergence code is  0
```

As the dimension p gets large, the number of parameters increases, and the optimization problem gets harder. Joe and Xu (1996) proposes a two-stage estimation method called inference functions for margins (IFM). The IFM method estimates the marginal parameters β in a first step by

$$\hat{\beta}_{\mathrm{IFM}} = \arg\max_{\beta} \sum_{i=1}^{n} \sum_{j=1}^{p} \log f_i(X_{ij}; \beta), \tag{10}$$

and then estimates the association parameters α given $\hat{\beta}_{\text{IFM}}$ by

$$\hat{\alpha}_{\text{IFM}} = \arg\max_{\alpha} \sum_{i=1}^{n} \log c\left(F_1(X_{i1}; \hat{\beta}_{\text{IFM}}), \ldots, F_p(X_{ip}; \hat{\beta}_{\text{IFM}}); \alpha\right). \tag{11}$$

When each marginal distribution F_i has its own parameters β_i so that $\beta = (\beta_1^\top, \ldots, \beta_p^\top)^\top$, the first step consists of an ML estimation for each margin $j = 1, \ldots, p$:

$$\hat{\beta}_{j\text{IFM}} = \arg\max_{\beta_j} \sum_{i=1}^{n} \log f(X_{ij}; \beta_j). \tag{12}$$

In this case, each maximization task has a very small number of parameters, greatly reducing the computational difficulty. This approach is called the two-stage parametric ML method by Shih and Louis (1995) in a censored data setting. In our illustration, the method can be carried out with the function `fitCopula`.

```
> loglik.marg <- function(b, x) sum(dgamma(x, shape = b[1], scale = b[2],
      log = TRUE))
> ctrl <- list(fnscale = -1)
> b1hat <- optim(b1.0, fn = loglik.marg, x = dat[, 1], control = ctrl)$par
> b2hat <- optim(b2.0, fn = loglik.marg, x = dat[, 2], control = ctrl)$par
> udat <- cbind(pgamma(dat[, 1], shape = b1hat[1], scale = b1hat[2]),
      pgamma(dat[, 2], shape = b2hat[1], scale = b2hat[2]))
> fit.ifl <- fitCopula(udat, myMvd@copula, start = a.0)
```

The estimate from the two stage is summarized as

```
> c(b1hat, b2hat, fit.ifl@est)

[1] 1.8301906 1.0375602 3.5167486 1.6284508 0.4199346

> fit.ifl

The ML estimation is based on  200  observations.
       Estimate Std. Error  z value     Pr(>|z|)
rho.1 0.4199346 0.05368112 7.822762 5.107026e-15
The maximized loglikelihood is  19.41620
The convergence code is  0
```

Note that the IFM estimate is close to the ML estimate. The standard error of $\hat{\alpha}_{\text{IFM}}$ is underestimated because the variation of $\hat{\beta}_{\text{IFM}}$ is not appropriately taken care of.

When consistent estimation of the dependence parameter α is important, it can be estimated with the canonical ML (CML) method without specifying the marginal distributions. This approach uses the empirical CDF of each marginal distribution to transform the observations $(X_{i1}, \ldots, X_{ip})^\top$ into pseudo-observations with uniform margins $(U_{i1}, \ldots, U_{ip})^\top$ and then estimates α as

$$\hat{\alpha}_{\mathrm{CML}} = \arg\max_{\alpha} \sum_{i=1}^{n} \log c(U_{i1}, \ldots, U_{ip}; \alpha). \tag{13}$$

The method can be carried out with `fitCopula` as well:

```
> eu <- cbind((rank(dat[, 1]) - 0.5)/n, (rank(dat[, 2]) - 0.5)/n)
> fit.cml <- fitCopula(eu, myMvd@copula, start = a.0)
> fit.cml

The ML estimation is based on  200  observations.
       Estimate Std. Error z value     Pr(>|z|)
rho.1 0.4304444 0.05311244  8.1044 4.440892e-16
The maximized loglikelihood is  20.29229
The convergence code is  0
```

The CML estimate $\hat{\alpha}_{\mathrm{CML}}$ is noticeably different from $\hat{\alpha}_{\mathrm{ML}}$ and $\hat{\alpha}_{\mathrm{IFM}}$. It has the advantage of not relying on marginal specifications.

Maximum likelihood estimation of copula-based models can be easily extended to solve more complicated problems. For example, when covariates are to be incorporated into the margins, all one needs to do is to write the loglikelihood function (9), which is the summation of the loglikelihood of the copula and the the loglikelihood of all the margins. Then, maximum likelihood estimates can be obtained by feeding the loglikelihood function to an optimizer. Example code for incorporating covariates into the margins is presented in the Appendix where two margins are modeled by gamma regression and log-normal regression, respectively. Covariates can also be incorporated into copula parameters in a similar fashion.

5 Discussion

This article presents the design, features, and some implementation details of the R package `copula` for multivariate modeling with copulas. The package provides functions to evaluate density/distribution, generate random numbers, plot, and fit copula-based models. It is hoped that, through the dissemination of the software, everyone who may need them may access easily copula-based models in daily computing and therefore enjoy, as put by Genest and MacKay (1986), the joy of copulas.

The random number generator of high dimensional Archimedean copulas in the package currently uses the compound construction algorithm (Marshall and Olkin, 1988; Frees and Valdez, 1998). When the frailty distribution is known, this algorithm is very efficient and elegant. Whelan (2004) proposed an algorithm based on integral representations, and the algorithm can be used with composite nested Archimedean copulas. Wu et al. (2006) recently proposed an algorithm for sampling from exchangeable Archimedean copulas based on an extension of the variable transformation technique from the bivariate case to multivariate case (Genest and Rivest, 2001). These method can be useful when the frailly distribution is unknown. An alternative in that case is to obtain the frailty distribution with numerical inversion of Laplace transformation (Melchiori, 2006).

Given a dataset, choosing a copula to fit the data is an important but difficult problem (Durrleman et al., 2000). The true data generation mechanism is unknown, for a given amount of data, it is possible that several candidate copulas fit the data reasonably well or that none of the candidate fits the data well. When maximum likelihood method is used, the general practice is to fit the data with all the candidate copulas and choose the ones with the highest likelihood (Frees and Wang 2005). A graphical tool to choose among Archimedean copulas is based on the Kendall's process (Genest and Rivest 1993). Recent works on goodness-of-fit tests of copulas are mostly chi-squared type tests; see, for example, Fermanian (2005) and references therein. Copula selection and goodness-of-fit are active research areas.

Although the package is still under active development, it can be easily extended under the carefully designed structure. A user can write one's own code to implement a copula that is not already implemented in the package. That is, one only needs to implement `dcopula`, `pcopula`, and `rcopula` methods for this copula. As long as the class and method structure are followed, the graphics and fitting facility in the package can be used without further coding.

More facilities, such as extreme value copulas, association measures and tail dependence measures, will be included in future releases of the package.

A Example Code with Covariates in Margins

This section illustrates how to construct the loglikelihood function using the facilities in the `copula` package when covariates are to be incorporated into the margins. Suppose that we observe n bivariate observations $\{(Y_{i1}, Y_{i2}) : i = 1, \ldots, n\}$, and for each margin, there is a corresponding covariate matrix. The first margin Y_{1i} follows a gamma distribution with shape $\exp(X_{1i}^\top \beta_1)$ and scale ν. The second margin, after log transformation, $\log(Y_{2i})$ follows a normal distribution with mean $X_{2i}^\top \beta_2$ and standard deviation σ.

The following code defines the three components of the loglikelihood: gamma margin, log-normal margin, and copula.

```
> loglik.m1 <- function(b, y, x) {
     l <- length(b)
     sum(dgamma(y, shape = exp(x %*% b[-l]), scale = b[l], log = TRUE))
 }
> loglik.m2 <- function(b, y, x) {
     l <- length(b)
     sum(dlnorm(y, meanlog = x %*% b[-l], sdlog = b[l], log = TRUE))
 }
> loglik.cop <- function(a, u, copula) {
     copula@parameters <- a
     sum(log(dcopula(copula, u)))
 }
```

Note that the contribution from the copula needs to be fed with probability integral transformed margins. The following code provides the transformation.

```
> probtrans.m1 <- function(b, y, x) {
     l <- length(b)
     pgamma(y, shape = exp(x %*% b[-l]), scale = b[l])
 }
> probtrans.m2 <- function(b, y, x) {
     l <- length(b)
     plnorm(y, meanlog = x %*% b[-l], sdlog = b[l])
 }
```

The loglikelihood function can be easily composed as:

```
> myloglik <- function(theta, y, xmat, copula) {
     l1 <- ncol(xmat[[1]]) + 1
     l2 <- ncol(xmat[[2]]) + 1
     b1 <- theta[1:l1]
     b2 <- theta[(l1 + 1):(l1 + l2)]
     a <- theta[-(1:(l1 + l2))]
     u <- cbind(probtrans.m1(b1, y[, 1], xmat[[1]]), probtrans.m2(b2,
         y[, 2], xmat[[2]]))
     copula@parameters <- a
     loglik <- loglik.m1(b1, y[, 1], xmat[[1]]) + loglik.m2(b2,
         y[, 2], xmat[[2]]) + loglik.cop(a, u, copula)
     loglik
 }
```

This function can then be fed to an optimization routine.

To illustrate, we define a function to generate response variables from given parameter vector, design matrices, and copula structure:

```
> genY <- function(theta, xmat, copula) {
     l1 <- ncol(xmat[[1]]) + 1
     l2 <- ncol(xmat[[2]]) + 1
     b1 <- theta[1:l1]
     b2 <- theta[(l1 + 1):(l1 + l2)]
     a <- theta[-(1:(l1 + l2))]
     n <- nrow(xmat[[1]])
     u <- rcopula(copula, n)
     y1 <- qgamma(u[, 1], shape = exp(xmat[[1]] %*% b1[-l1]),
         scale = b1[l1])
     y2 <- qlnorm(u[, 2], meanlog = xmat[[2]] %*% b2[-l1], sdlog = b2[l2])
     cbind(y1, y2)
 }
```

Now, we generate a sample of size $n = 200$. The design matrices are generated from a continuous normal variable and a binary variable, respectively.

```
> n <- 200
> xmat <- list(model.matrix(~rnorm(n)), model.matrix(~rbinom(n,
     prob = 0.5, size = 1)))
> b1 <- c(1, 0.5, 2)
> b2 <- c(2, -1, 3)
> a <- 0.5
> theta <- c(b1, b2, a)
> myCop <- normalCopula(a, dim = 2, dispstr = "ex")
> y <- genY(theta, xmat, myCop)
> myloglik(theta, y, xmat, myCop)

[1] -1281.640
```

The optimization routine needs an initial search point. The initial point may be chosen based on regressions on each margin and MoM on the copula. For illustration purpose, we use the true value as starting value and see how fast it converges:

```
> fit <- optim(theta, myloglik, control = list(fnscale = -1, trace = 1),
     y = y, xmat = xmat, copula = myCop, method = "BFGS")

initial  value 1281.639657
iter  10 value 1279.312818
final  value 1279.311152
converged

> fit

$par
[1]  1.0711075  0.5129254  1.7993063  2.2886217 -1.1995043  2.8907541  0.4166082
```

```
$value
[1] -1279.311

$counts
function gradient
      48       11

$convergence
[1] 0

$message
NULL
```

The variance matrix of the estimator can be estimated by inverting the Fisher information matrix, which is approximated by the negative Hessian matrix.

References

Azzalini, A., 2005. R package 'sn':The skew-normal and skew-t distribution (version 0.40). Università di Padova, Italia.
URL `URL http://azzalini.stat.unipd.it/SN`

Bouyè, E., Durrleman, V., Bikeghbali, A., Riboulet, G., Rconcalli, T., 2000. Copulas for finance – A reading guide and some applications. Working paper, Goupe de Recherche Opérationnelle, Crédit Lyonnais.

Chambers, J. M., 1998. Programming with Data: A Guide to the S Language. Springer-Verlag Inc.

Chambers, J. M., Mallows, C. L., Stuck, B. W., 1976. A method for simulating stable random variables (Corr: V82 p704; V83 p581). Journal of the American Statistical Association 71, 340–344.

Cherubini, U., Luciano, E., Vecchiato, W., 2004. Copula Methods in Finance. John Wieley & Son Ltd.

Clayton, D. G., 1978. A model for association in bivariate life tables and its application in epidemiological studies of familial tendency in chronic disease incidence. Biometrika 65, 141–152.

Durrleman, V., Nikeghbali, A., Rconcalli, T., 2000. Which copula is the right one? Working paper, Goupe de Recherche Opérationelle, Crédit Lyonnais.

Embrechts, P., Lindskog, F., McNeil, A., 2003. Modelling dependence with copulas and applications to risk management. In: Rachev, S. (Ed.), Handbook of Heavy Tailed Distribution in Finance. Elsevier, pp. 329–384.

Escarela, G., Carrière, J. F., 2003. Fitting competing risks with an assumed copula. Statistical Methods in Medical Research 12 (4), 333–349.

Fang, K.-T., Kotz, S., Ng, K. W., 1990. Symmetric Multivariate and Related Distributions. Chapman & Hall Ltd.

Fermanian, J.-D., 2005. Goodness-of-fit tests for copulas. Journal of Multivariate Analysis 95 (1), 119–152.

Frank, M. J., 1979. On the simultaneous associativity of $f(x, y)$ and $x + y - f(x, y)$. Aequationes Mathematicae 19, 194–226.

Frees, E. W., Carriere, J., Valdez, E. A., 1996. Annuity valuation with dependent mortality. Journal of Risk and Insurance 63, 229–261.

Frees, E. W., Valdez, E. A., 1998. Understanding relationships using copulas. North American Actuarial Journal 2 (1), 1–25.

Frees, E. W., Wang, P., 2005. Credibility using copulas. North American Actuarial Journal 9 (2), 31–48.

Genest, C., Favre, A.-C., 2006. Everything you always wanted to know about copula modeling but were afraid to ask. Journal of Hydrologic Engineering 11, in press.

Genest, C., MacKay, J., 1986. The joy of copulas: Bivariate distributions with uniform marginals (Com: 87V41 p248). The American Statistician 40, 280–283.

Genest, C., Rivest, L.-P., 2001. On the multivariate probability integral transformation. Statistics & Probability Letters 53 (4), 391–399.

Genz, A., Bretz, F., Hothorn, T., 2005. mvtnorm: Multivariate Normal and T Distribution. R package version 0.7-2.

Gumbel, E. J., 1960. Bivariate exponential distributions. Journal of the American Statistical Association 55, 698–707.

Insightful Corporation, 2002. S+Finmetrics Reference Manual. Seattle, WA. URL `http://www.insightful.com`

Insightful Corporation, 2005. S-PLUS (version 7.0). Seattle, WA. URL `http://www.insightful.com`

Joe, H., 1997. Multivariate Models and Dependence Concepts. Chapman & Hall Ltd.

Joe, H., Xu, J., 1996. The estimation method of inference functions for margins for multivariate models. Technical Report 166, Department of Statistics, University of British Columbia.

Kemp, A. W., 1981. Efficient generation of logarithmically distributed pseudo-random variables. Applied Statistics 30, 249–253.

Ligges, U., Mächler, M., 2003. Scatterplot3d - an r package for visualizing multivariate data. Journal of Statistical Software 8 (11), 1–20. URL `http://www.jstatsoft.org`

Marshall, A. W., Olkin, I., 1988. Families of multivariate distributions. Journal of the American Statistical Association 83, 834–841.

Melchiori, M. R., 2006. Tools for sampling multivariate archimedean copulas. Online paper, Universidad Nacional del Litoral.

Nelsen, R. B., 1999. An Introduction to Copulas. Springer-Verlag Inc.

Nolan, J. P., 2006. Stable distributions – Models for heavy tailed data. URL `http://academic2.american.edu/∼jpnolan`

R Development Core Team, 2006a. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria,

ISBN 3-900051-07-0.
URL `http://www.R-project.org`

R Development Core Team, 2006b. Writing R Extensions. R Foundation for Statistical Computing, Vienna, Austria, ISBN 3-900051-07-0.
URL `http://www.R-project.org`

Shih, J. H., Louis, T. A., 1995. Inferences on the association parameter in copula models for bivariate survival data. Biometrics 51, 1384–1399.

Sklar, A. W., 1959. Fonctions de répartition à n dimension et leurs marges. Publications de l'Institut de Statistique de l'Université de Paris 8, 229–231.

Song, P. X.-K., 2000. Multivariate dispersion models generated from Gaussian copula. Scandinavian Journal of Statistics 27 (2), 305–320.

Wang, W., Wells, M. T., 2000. Model selection and semiparametric inference for bivariate failure-time data (C/R: p73-76). Journal of the American Statistical Association 95 (449), 62–72.

Whelan, N., 2004. Sampling from archimedean copulas. Quantitative Finance 4 (3), 339–352.

Wu, F., Valdez, E. A., Sherris, M., 2006. Simulating exchangeable mutivariate archimedean copulas and its applications. Working paper, School of Actuarial Studies, University of New South Wales.

Yan, J., 2006a. copula: Multivariate Dependence with Copula. R package version 0.3-7.

Yan, J., 2006b. Multivariate modeling with copulas and engineering applications. In: Pham, H. (Ed.), Handbook in Engineering Statistics. Springer, pp. 973–990.