

# Package ‘SimCop’

February 22, 2026

**Type** Package

**Title** Simulate from Arbitrary Copulae

**Version** 0.7.4

**Date** 2026-02-22

**Description** Provides a framework to generating random variates from arbitrary multivariate copulae, while concentrating on (bivariate) extreme value copulae. Particularly useful if the multivariate copulae are not available in closed form. Detailed discussion of the methodologies used can be found in Tajvidi and Turlach (2018)  
<[doi:10.1111/anzs.12209](https://doi.org/10.1111/anzs.12209)>.

**Depends** R (>= 4.5.0)

**Imports** grDevices,  
graphics,  
quadprog,  
rgl,  
stats

**License** GPL (>=2)

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.3.2

## Contents

GenerateRV . . . . .	2
GenerateRV.CopApprox . . . . .	2
GetApprox . . . . .	4
MaxTemp . . . . .	5
NewBEVAsyLogisticCopula . . . . .	6
NewBEVAsyMixedModelCopula . . . . .	7
NewBEVLogisticCopula . . . . .	8
NewBEVMixedModelCopula . . . . .	9
NewBEVSplineCopula . . . . .	10
NewMEVAsyLogisticCopula . . . . .	10
NewMEVGumbelCopula . . . . .	12
NewMVClaytonCopula . . . . .	13

NewMVFrankCopula . . . . .	13
NonparEstDepFct . . . . .	14
plot.CopApprox . . . . .	16
plot.SimCop . . . . .	17
print.SimCop . . . . .	19
SimCop . . . . .	19
SplineFitDepFct . . . . .	20

<b>Index</b>	<b>22</b>
--------------	-----------

---

GenerateRV	<i>GenerateRV generic</i>
------------	---------------------------

---

## Description

A generic to sample random variates from an object.

## Usage

```
GenerateRV(obj, n, ...)
```

## Arguments

obj	object from which to sample.
n	number of items to sample.
...	further arguments for methods.

## Author(s)

Berwin A. Turlach <berwin.turlach@gmail.com>

---

GenerateRV.CopApprox	<i>Generates random variates from a copula approximation</i>
----------------------	--

---

## Description

Method to sample random variates from an object of [class](#) ‘CopApprox’.

## Usage

```
## S3 method for class 'CopApprox'
GenerateRV(
  obj,
  n,
  MH = FALSE,
  trace = FALSE,
  PDF = NULL,
  burnin = 500,
  thinning = 5,
  ...
)
```

## Arguments

obj	object from which to sample.
n	number of items to sample.
MH	logical, should a Metropolis-Hastings algorithm be used to refine the sample? Default is FALSE.
trace	logical, indicating whether the function should be verbose.
PDF	probability density function corresponding to copula used to create obj, only used if MH is TRUE; see ‘Details’.
burnin	the number of burn-in iterations of the MH sampler, only used if MH is TRUE, defaults to 500.
thinning	the thinning parameter, only used if MH is TRUE, defaults to 5.
...	not used.

## Details

If argument MH is FALSE, the default, random variates are directly sampled from the approximation, as described in Tajvidi and Turlach (2018).

If MH is TRUE, GenerateRV uses additionally a Metropolis-Hastings refinement. It first samples from the approximation, but uses those samples then as proposals in a Metropolis-Hastings algorithm. The latter needs the probability density function of the copula. This density function has either to be passed to the argument PDF, or the copula (stored in argument obj) belonging to the approximation must have the density function (with name ‘pdfCopula’) stored in its environment. In the latter case, the argument PDF can be NULL (its default).

## Value

A matrix of dimension n times dim, where dim is the dimension for which the copula approximation was determined.

If MH was TRUE the return value has an attribute called ‘AcceptanceRate’, indicating the fraction of samples that were accepted in the Metropolis-Hastings step. This fraction is based on all burnin + (n-1)\*thinning + 1 samples that are initially generated from the approximation.

## References

Tajvidi, N. and Turlach, B.A. (2018). A general approach to generate random variates for multivariate copulae, *Australian & New Zealand Journal of Statistics* **60**(1): 140–155. doi:10.1111/anzs.12209.

## See Also

[GetApprox](#)

## Examples

```
cop <- NewBEVAsyMixedModelCopula(theta=1, phi=-0.25)
approx1 <- GetApprox(cop)
approx2 <- GetApprox(cop, type = 1)
sample1 <- GenerateRV(approx1, 100)
plot(sample1)
sample2 <- GenerateRV(approx2, 100)
plot(sample2)
```

```
sample1 <- GenerateRV(approx1, 50, MH = TRUE, trace = TRUE)
plot(sample1)
sample2 <- GenerateRV(approx2, 50, MH = TRUE)
plot(sample2)
```

GetApprox

*Approximate a copula by a histogram density*

### Description

Approximates the “density” of a copula by a piece-wise constant function.

### Usage

```
GetApprox(
  Cop,
  dim = 2,
  depth = ifelse(type == 1, 10, 32),
  type = 1,
  TOL = 1000 * .Machine$double.eps
)
```

### Arguments

Cop	A function defining the copula.
dim	The approximation should be calculated on the dim-dimensional unit cube, defaults to 2.
depth	The number of hyperrectangles to be used to divide the unit cube, defaults to 10 for Approximation I and to 32 for Approximation II.
type	Whether Approximation I or Approximation II should be used, defaults to one.
TOL	A numerical tolerance used when calculating Approximation I.

### Details

This function provides two methods for subdividing the  $d$ -dimensional unit cube into hyper-rectangles, with  $d$  being passed to the parameter `dim`. As most of the functions in this package which create a new copula return a function that can be evaluated at points in arbitrary dimensions, it is necessary to specify for which dimension  $d$  one wishes to calculate the approximation to the copula’s “density”.

The first method (Approximation I) determines  $2^m$  hyper-rectangles (where  $m$  is the parameter `depth`), each containing the same probability mass with respect to the copula. The second method (Approximation II) divides the unit cube into  $m^d$  hyper-squares.

These approximations can be interpreted as piecewise constant approximations of the copula’s probability density function if the copula is absolutely continuous. For further details see ‘References’.

**Value**

GetApprox returns an object of `class` ‘CopApprox’ according to its inputs. The returned object is a list containing a matrix that holds the information of the approximation, the argument Cop, which approximation was determined, and other auxiliary information.

The only method for objects of class ‘CopApprox’ implemented so far are for the generic function `plot`, and then only for the case if dim was 2.

**Author(s)**

Berwin A. Turlach <berwin.turlach@gmail.com>

**References**

Tajvidi, N. and Turlach, B.A. (2018). A general approach to generate random variates for multivariate copulae, *Australian & New Zealand Journal of Statistics* **60**(1): 140–155. doi:[10.1111/anzs.12209](https://doi.org/10.1111/anzs.12209).

**See Also**

`plot.CopApprox`

**Examples**

```
Cop <- NewMEVGumbelCopula(3)
CopApprox <- GetApprox(Cop, dim=2)
plot(CopApprox)
```

---

MaxTemp

*Extreme temperatures at two West Australian meteorological stations*


---

**Description**

A dataset on maximum annual values of average daily temperature measurements at two meteorological stations—Leonora (latitude 28.53S, longitude 121.19E) and Menzies (latitude 29.42S, longitude 121.02E)—in Western Australia, for the period 1898–1993.

**Usage**

MaxTemp

**Format**

A data frame with 96 rows and 2 variables:

**Leonora** annual maximal temperature at Leonora, in degrees Celsius

**Menzies** annual maximal temperature at Menzies, in degrees Celsius

**References**

Hall, P. and Tajvidi, N. (2004). Prediction regions for bivariate extreme events. *Australian & New Zealand Journal of Statistics* **46**(1): 99–112. doi:[10.1111/j.1467842X.2004.00316.x](https://doi.org/10.1111/j.1467842X.2004.00316.x).

**Examples**

```
plot(Menzies ~ Leonora, MaxTemp,
     xlab = expression("Temperature at Leonora (*degree*C)"),
     ylab = expression("Temperature at Menzies (*degree*C)"))
```

---

NewBEVAsyLogisticCopula

*Creates a bivariate asymmetric logistic model extreme value copula*


---

**Description**

Creates an instance of the bivariate asymmetric logistic model extreme value copula with parameters  $r$ ,  $\theta$  and  $\phi$ .

**Usage**

```
NewBEVAsyLogisticCopula(r, theta, phi)
```

**Arguments**

<code>r</code>	real.
<code>theta</code>	real.
<code>phi</code>	real.

**Details**

The dependence function for this bivariate EV copula is

$$A(w) = (\theta(1-w)^r + (\phi w)^r)^{1/r} + (\theta - \phi)w + 1 - \theta$$

Necessary and sufficient conditions for the dependence function to be valid are

- $r \geq 1$
- $0 \leq \theta \leq 1$
- $0 \leq \phi \leq 1$

For  $\theta = \phi = 1$  this model reduces to the symmetric logistic model.

**Value**

A function that evaluates the bivariate asymmetric logistic model EV copula (with parameters  $r$ ,  $\theta$  and  $\phi$ ) at a given 2-dimensional vector in the unit square. The environment of the function also contains a function called `pdfCopula` that evaluates the probability density function of the bivariate asymmetric mixed model EV copula via automatic differentiation.

**Author(s)**

Berwin A. Turlach <berwin.turlach@gmail.com>

**See Also**

[NewBEVLogisticCopula](#), [NewMEVAsyLogisticCopula](#)

---

`NewBEVAsyMixedModelCopula`*Creates a bivariate asymmetric mixed model extreme value copula*

---

**Description**

Creates an instance of the bivariate asymmetric mixed model extreme value copula with parameters  $\phi$  and  $\theta$ .

**Usage**

```
NewBEVAsyMixedModelCopula(theta, phi)
```

**Arguments**

<code>theta</code>	real.
<code>phi</code>	real.

**Details**

The dependence function for this bivariate EV copula is

$$A(w) = \phi w^3 + \theta w^2 - (\phi + \theta)w + 1$$

Necessary and sufficient conditions for the dependence function to be valid are

- $\theta \geq 0$
- $\theta + 3\phi \geq 0$
- $\theta + \phi \leq 1$
- $\theta + 2\phi \leq 1$

If  $\phi = 0$  we obtain the symmetric mixed model.

**Value**

A function that evaluates the bivariate asymmetric mixed model EV copula (with parameters  $\phi$  and  $\theta$ ) at a given 2-dimensional vector in the unit square. The environment of the function also contains a function called `pdfCopula` that evaluates the probability density function of the bivariate asymmetric mixed model EV copula via automatic differentiation.

**Author(s)**

Berwin A. Turlach <berwin.turlach@gmail.com>

**See Also**

[NewBEVMixedModelCopula](#)

---

NewBEVLogisticCopula	<i>Creates a bivariate logistic model extreme value copula</i>
----------------------	--

---

### Description

Creates an instance of the bivariate logistic model extreme value copula with parameter  $r$ .

### Usage

```
NewBEVLogisticCopula(r)
```

### Arguments

<code>r</code>	real.
----------------	-------

### Details

The dependence function for this bivariate EV copula is

$$A(w) = ((1 - w)^r + w^r)^{1/r}$$

Necessary and sufficient conditions for the dependence function to be valid are

- $r \geq 1$

### Value

A function that evaluates the bivariate logistic model EV copula (with parameter  $r$ ) at a given 2-dimensional vector in the unit square. The environment of the function also contains a function called `pdfCopula` that evaluates the probability density function of the bivariate asymmetric mixed model EV copula via automatic differentiation.

### Author(s)

Berwin A. Turlach <berwin.turlach@gmail.com>

### See Also

[NewBEVAsyLogisticCopula](#), [NewMEVGumbelCopula](#)



---

`NewBEVMixedModelCopula`*Creates a bivariate mixed model extreme value copula*

---

**Description**

Creates an instance of the bivariate asymmetric mixed model extreme value copula with parameter  $\theta$ .

**Usage**

```
NewBEVMixedModelCopula(theta)
```

**Arguments**

`theta`                `real`.

**Details**

The dependence function for this bivariate EV copula is

$$A(w) = \theta w^2 - \theta w + 1$$

Necessary and sufficient conditions for the dependence function to be valid are

- $0 \leq \theta \leq 1$

**Value**

A function that evaluates the bivariate asymmetric mixed model EV copula (with parameter  $\theta$ ) at a given 2-dimensional vector in the unit square. The environment of the function also contains a function called `pdfCopula` that evaluates the probability density function of the bivariate asymmetric mixed model EV copula via automatic differentiation.

**Author(s)**

Berwin A. Turlach <berwin.turlach@gmail.com>

**See Also**

[NewBEVAsyMixedModelCopula](#)

---

NewBEVSplineCopula	<i>Creates a flexible extreme value copula</i>
--------------------	--

---

**Description**

Creates a bivariate extreme value copula from a spline estimate of its dependence function.

**Usage**

```
NewBEVSplineCopula(spl)
```

**Arguments**

spl	a spline function.
-----	--------------------

**Value**

A function that evaluates the bivariate EV copula (whose dependence function is given by the spline) at a given 2-dimensional vector in the unit square. The environment of the function also contains a function called pdfCopula that evaluates the probability density function of the bivariate asymmetric mixed model EV copula via automatic differentiation.

**Author(s)**

Berwin A. Turlach <berwin.turlach@gmail.com>

**See Also**

[SplineFitDepFct](#)

---

NewMEVasyLogisticCopula	<i>Creates a multivariate asymmetric logistic copula</i>
-------------------------	--

---

**Description**

Creates an instance of the multivariate asymmetric copula with parameters  $\theta$  and  $r$ .

**Usage**

```
NewMEVasyLogisticCopula(theta, r)
```

**Arguments**

theta	a $k \times d$ matrix of reals.
r	a vector of $k$ reals

## Details

If theta has entries  $\theta_{ij}$  and r has entries  $r_j$  ( $i = 1, \dots, k$  and  $j = 1, \dots, d$ ), then the following parameterisation of the copula is used:

$$C(u_1, \dots, u_d) = \exp \left( - \sum_{i=1}^k \left\{ \sum_{j=1}^d (\theta_{ij} \bar{u}_j)^{r_i} \right\}^{1/r_i} \right)$$

where  $\bar{u}_j = -\log(u_j)$ ,  $j = 1, \dots, d$ .

Necessary and sufficient conditions for the parameters are

- all entries in theta have to be non-negative.
- each column of theta has to add to one.
- each row of theta must have a unique pattern of non-zero values, including the pattern that has no zeros in a row.
- if a row of theta has only one non-zero value, then the corresponding entry in r has to be one.
- if a row of theta has more than one non-zero value, then the corresponding entry of r must be greater than one.

## Value

A function that evaluates the multivariate asymmetric logistic copula (with parameters  $\theta$  and  $r$ ) at a given  $d$ -dimensional vector in the unit square. Note that for this function the dimension of vectors at which the copula can be evaluated is determined by the input parameters. The environment of the function also contains a function called pdfCopula that evaluates the probability density function of the multivariate asymmetric logistic copula via automatic differentiation.

## Author(s)

Berwin A. Turlach <berwin.turlach@gmail.com>

## See Also

[NewBEVAsyLogisticCopula](#), [NewMEVGumbelCopula](#)

## Examples

```
theta <- rbind(c(0, 0.2, 0.8), c(1,0.8,0.2))
r <- c(2,3)
cop <- NewMEVAsyLogisticCopula(theta, r)

## Creates the same copula
theta <- 0.2
phi <- 0.4
r <- 2
cop1 <- NewBEVAsyLogisticCopula(r, theta, phi)
theta <- cbind(c(phi, 1-phi, 0), c(theta, 0, 1-theta))
r <- c(r, 1, 1)
cop2 <- NewMEVAsyLogisticCopula(theta, r)
```

---

NewMEVGumbelCopula	<i>Creates a Gumpel copula</i>
--------------------	--------------------------------

---

### Description

Creates an instance of the Gumbel copula with parameter  $r$ . This family is also known as the Gumbel–Hougaard copula or the logistic model.

### Usage

```
NewMEVGumbelCopula(r = 2)
```

### Arguments

$r$  real, the parameter of the Gumbel copula, defaults to 2, must be larger or equal to one.

### Details

The following parameterisation of the copula is used:

$$C(u_1, \dots, u_d) = \exp \left( - \left\{ \sum_{j=1}^d \bar{u}_j^r \right\}^{1/r} \right)$$

where  $\bar{u}_j = -\log(u_j)$ ,  $j = 1, \dots, d$ .

### Value

A function that evaluates the Gumbel copula (with parameter  $r$ ) at a given  $d$ -dimensional vector in the unit cube. The environment of the function also contains a function called `pdfCopula` that evaluates the probability density function of the Gumbel copula via automatic differentiation.

### Author(s)

Berwin A. Turlach <berwin.turlach@gmail.com>

### See Also

[NewMEVAsyLogisticCopula](#)

---

NewMVClaytonCopula	<i>Creates a Clayton copula</i>
--------------------	---------------------------------

---

**Description**

Creates an instance of the Clayton copula with parameter  $\theta$ .

**Usage**

```
NewMVClaytonCopula(theta = 1)
```

**Arguments**

theta                      real, the parameter of the Clayton copula, defaults to 1; must be positive.

**Details**

The following parameterisation of the copula is used:

$$C(u_1, \dots, u_d) = \left( \left\{ \sum_{j=1}^d u_j^{-\theta} \right\} - (d-1) \right)^{-1/\theta}$$

**Value**

A function that evaluates the Clayton copula (with parameter  $\alpha$ ) at a given  $d$ -dimensional vector in the unit cube. The environment of the function also contains a function called pdfCopula that evaluates the probability density function of the Clayton copula via automatic differentiation.

**Author(s)**

Berwin A. Turlach <berwin.turlach@gmail.com>

---

NewMVFrankCopula	<i>Creates a Frank copula</i>
------------------	-------------------------------

---

**Description**

Creates an instance of the Frank copula with parameter  $\alpha$ .

**Usage**

```
NewMVFrankCopula(alpha = 1)
```

**Arguments**

alpha                      real, the parameter of the Frank copula, defaults to 1; must be positive.

**Details**

The following parameterisation of the copula is used:

$$C(u_1, \dots, u_d) = -\log(1 + \exp(s) * t) / \alpha$$

where  $s = \sum_{j=1}^d \log\left(\frac{\exp(-\alpha u_j) - 1}{t}\right)$  and  $t = \exp(-\alpha) - 1$ .

**Value**

A function that evaluates the Frank copula (with parameter  $\alpha$ ) at a given  $d$ -dimensional vector in the unit cube. The environment of the function also contains a function called `pdfCopula` that evaluates the probability density function of the Frank copula via automatic differentiation.

**Author(s)**

Berwin A. Turlach <berwin.turlach@gmail.com>

---

NonparEstDepFct

---

*Nonparametric estimator of bivariate dependence function*


---

**Description**

Function to calculate nonparametric estimates of the dependence functions of bivariate extreme value copula.

**Usage**

```
NonparEstDepFct(
  x,
  y = NULL,
  w.length = 101,
  transf.to.frechet = TRUE,
  convex.hull = TRUE,
  verbose = FALSE
)
```

**Arguments**

<code>x, y</code>	vectors giving the observations of the extreme values. Alternatively a single plotting structure can be specified: see <a href="#">xy.coords</a> .
<code>w.length</code>	number of grid points (using an equidistant grid from 0 to 1) on which the dependence function is estimated.
<code>transf.to.frechet</code>	logical, controls whether <code>x</code> and <code>y</code> are first transformed to have standard Fréchet margins: see ‘Details’; defaults to TRUE.
<code>convex.hull</code>	logical, controls whether the convex hull of the modified Pickands estimator is returned; defaults to TRUE.
<code>verbose</code>	logical, controls whether progress messages are given; defaults to FALSE.

## Details

If `transf.to.frechet` is TRUE, the default, then a generalised extreme value (GEV) distribution is fitted to each margin and the fitted parameters are used to transform the data to have standard Fréchet margins. The parameterisation of the cumulative distribution of the GEV that is used is, if  $\gamma \neq 0$ :

$$G(z) = \exp \left( - \left[ 1 + \gamma \left( \frac{z - \mu}{\sigma} \right) \right]^{-1/\gamma} \right)$$

and for  $\gamma = 0$ :

$$G(z) = \exp(-\exp(-z))$$

If  $\gamma < 0$ , then the support of the GEV is the interval  $(-\infty, \mu - \sigma/\gamma]$ , while it is  $[\mu - \sigma/\gamma, \infty)$  if  $\gamma > 0$ . For  $\gamma = 0$ , the support is the real line.

If `verbose` is TRUE, not the default, and `transf.to.frechet` is TRUE, the estimates for the fitted GEV distribution are printed out using [cat](#).

## Value

A list with two named components. The component `x` contains a vector with the grid points at which the dependence function was estimated. The component `y` contains the estimated dependence functions.

## Author(s)

Nader Tajvidi <Nader.Tajvidi@matstat.lu.se>

## References

Hall, P. and Tajvidi, N. (2000). Distribution and dependence-function estimation for bivariate extreme-value distributions. *Bernoulli* **6**(5): 835–844. doi:10.2307/3318758.

Hall, P. and Tajvidi, N. (2004). Prediction regions for bivariate extreme events. *Australian & New Zealand Journal of Statistics* **46**(1): 99–112. doi:10.1111/j.1467842X.2004.00316.x.

## See Also

[SplineFitDepFct](#)

## Examples

```
## Data from Hall and Tajvidi (2004, ANZJS)
EstDF1 <- NonparEstDepFct(MaxTemp)

## Plot modified Pickands Function and area in which
## dependence function must lie
plot(EstDF1, ylim = c(0.5,1), xlab = "w", ylab = "A(w)", type="l", lty="longdash")
polygon(c(0, 0.5, 1, 0), c(1, 0.5, 1, 1))
```

plot.CopApprox

*Plot the histogram density approximation to a copula*

## Description

Plots the histogram density approximation to a copula as determined by [GetApprox](#). Currently works only for bivariate copulae.

## Usage

```
## S3 method for class 'CopApprox'
plot(
  x,
  type = c("rgl", "original"),
  col = if (type == "rgl") "#cccccc" else grey.colors(100, start = 0, end = 0.8),
  qcut = 0.95,
  cut,
  alpha = 1,
  topcol = "#ff0000",
  sidecol = "#aaaaaa",
  linecol = "#000000",
  ...
)
```

## Arguments

<code>x</code>	an object of <a href="#">class</a> 'CopApprox'.
<code>type</code>	specifies the type of plot to produce. Possible values are "rgl" and "original"; see 'Details'. Can be abbreviated.
<code>col</code>	colour(s) to be used; see 'Details'.
<code>qcut, cut</code>	used if type is "rgl"; see 'Details'.
<code>alpha, topcol, sidecol, linecol</code>	used if type is "rgl"; see 'Details'.
<code>...</code>	used if type is "original"; see 'Details'.

## Details

If type is "original" then plots are produced of the kind shown in Tajvidi and Turlach (2018). In this case the dot arguments are passed to [plot](#) when the initial plot is created. Thus, they can be used to set the main title, axes labels and so forth. If the approximation is of type Approximation II, then argument `col` is used to colour the squares while they are plotted and filled via [polygon](#). For this, the ranks of the probability masses of the squares are mapped (linearly) onto the provided colours.

If type is "rgl" then plots are used using [rgl](#). The code used is based on the 'hist3d' demo of the [rgl](#) package. The argument `col` sets the background colour of the plot. Arguments `topcol` and `sidecol` are used to set the colour of the top and sides of the cuboids, and the edges of the cuboids are drawn using `linecol`. Argument `alpha` sets the transparency. Finally, as the heights of the cuboids can be large, in particular for extreme value copulae, their heights are truncated using either `cut` (absolute value) or `qcut` (corresponding quantile of all heights as determined by the [quantile](#) function). After truncation, the heights are rescaled to be between 0 and 1, thus the unit on the "z"-axis is meaningless.



**Value**

NULL is returned [invisibly](#).

**Author(s)**

Berwin A. Turlach <berwin.turlach@gmail.com>

**References**

Tajvidi, N. and Turlach, B.A. (2018). A general approach to generate random variates for multivariate copulae, *Australian & New Zealand Journal of Statistics* **60**(1): 140–155. [doi:10.1111/anzs.12209](#).

**Examples**

```
Cop <- NewMEVGumbelCopula(4)
CopApprox1 <- GetApprox(Cop, dim=2)
plot(CopApprox1)
plot(CopApprox1, type = "o")
CopApprox2 <- GetApprox(Cop, dim=2, type=2)
plot(CopApprox2)
plot(CopApprox2, type = "o", xlab = expression(u[1]), ylab = expression(u[2]))
plot(CopApprox2, type = "o", col = heat.colors(100))
```

---

plot.SimCop

---

*Plot a bivariate copula or its density*


---

**Description**

Plot a bivariate copula or its density function. Essentially hands over immediately to [persp3d](#).

**Usage**

```
## S3 method for class 'SimCop'
plot(x, ...)

## S3 method for class 'SimCop'
persp3d(
  x,
  ...,
  type = c("cdf", "pdf"),
  nx = 129,
  ny = 129,
  col = heat.colors(100),
  qcut = 0.95,
  cut,
  xlab = expression(u[1]),
  ylab = expression(u[2])
)
```

## Arguments

x	an object of <code>class</code> ‘SimCop’.
...	additional arguments passed to <code>persp3d</code> .
type	specifies whether the copula (“cdf”) or its probability density function (“pdf”) is plotted; see ‘Details’. Can be abbreviated.
nx, ny	length of grids that define the gridlines on which the plotted function is evaluated.
col	colour scheme to use for the surface; see ‘Details’.
qcut, cut	used if type is “pdf”; see ‘Details’.
xlab, ylab	default labels for the two axes.

## Details

If type is “pdf”, then x must have a function with name “pdfCopula” in its environment which evaluates the probability density function of the copula stored in x.

For plotting the copula, the “x”-grid and “y”-grid are produced by using, respectively, nx and ny equispaced points between 0 and 1 (inclusive). To avoid evaluating the density function at the boundary, if type is “pdf”, then grids are initially generated in the same manner, after which the first grid point is removed and all remaining grid points are shifted towards the origin by half the distance between two neighbouring grid points.

The function to be plotted is evaluated at all possible combination of the grid points. As the function values of the density function can be large, in particular for extreme value copulae, the values are truncated using either cut (absolute value) or qcut (corresponding quantile of all values as determined by the `quantile` function).

Finally, the colour scheme passed by argument col is used to assign a colour to every point at which the function was evaluated by linearly mapping the function values onto the provided colours.

## Value

Returns a list with four components `invisibly`. Components x and y contain the location of grid lines at which the function was evaluate. Component z contains the function evaluations (possibly truncated) and component col contains the col used.

## Author(s)

Berwin A Turlach <berwin.turlach@gmail.com>

## Examples

```
Cop1 <- NewMVFrankCopula(2)
plot(Cop1)
plot(Cop1, type = "p", qcut = 1)
```

---

print.SimCop	<i>Print basic information on a copula</i>
--------------	--

---

### Description

Prints basic information on a copula created with the methods in this package.

### Usage

```
## S3 method for class 'SimCop'
print(x, ...)
```

### Arguments

x	an object of <code>class</code> 'SimCop'.
...	not used.

### Author(s)

Berwin A. Turlach <berwin.turlach@gmail.com>

---

SimCop	<i>SimCop: A package to simulate random variates from an arbitrary multivariate copula</i>
--------	--

---

### Description

R code to support Tajvidi and Turlach (2018). The main functions implemented for the SimCop package are:

- New\*Copula, various functions that create objects of `class` 'SimCop'. These functions return a copula function with various helpful information stored in the environment of the function. Details of the implementation are subject to change and should not be relied on. Only a `print` method is implemented for this class so far.
- GetApprox, a function that calculates approximations to a copula and returns an object of `class` 'CopApprox'.  
For bivariate copulae a method for `plot` is implemented for this class.
- GenerateRV, a generic function that generates random variates from an object, together with a method for objects of class 'CopApprox'.

### Author(s)

**Maintainer:** Berwin A. Turlach <Berwin.Turlach@gmail.com> ([ORCID](#))

Other contributors:

- Nader Tajvidi <Nader.Tajvidi@matstat.lu.se> ([ORCID](#)) [contributor]

### References

Tajvidi, N. and Turlach, B.A. (2018). A general approach to generate random variates for multivariate copulae, *Australian & New Zealand Journal of Statistics* **60**(1): 140–155. doi:[10.1111/anzs.12209](#).

---

SplineFitDepFct

*Fit a dependence function by spline smoothing*


---

## Description

Given estimates for the dependence function of a bivariate extreme value copula at specified points, this function fits a natural cubic smoothing spline, that is constrained to fulfill all the conditions of a dependence function, to the given estimates via quadratic programming.

## Usage

```
SplineFitDepFct(x, y = NULL, alpha = 0.01, integ.value)
```

## Arguments

x, y	vectors giving the coordinates of the points to be approximated. Alternatively a single plotting structure can be specified: see <a href="#">xy.coords</a> .
alpha	real, the smoothing parameter for the smoothing splines.
integ.value	real, non-negative value that should be less than two; see ‘Details’

## Details

integ.value should be between 0 and 2. If a value is specified, then an additional constraint is added to the quadratic program to ensure that the integral (over 0 to 1) of the second derivative of the spline is larger or equal to integ.value. Choosing values close to 2 may lead to quadratic programmes on which [solve.QP](#) reports inconsistent constraints.

## Value

A function, created by [splinefun](#), that evaluates the natural cubic spline that was fitted to the data.

## Author(s)

Nader Tajvidi <Nader.Tajvidi@matstat.lu.se>  
 Berwin A Turlach <Berwin.Turlach@gmail.com>

## References

Hall, P. and Tajvidi, N. (2000). Distribution and dependence-function estimation for bivariate extreme-value distributions. *Bernoulli* **6**(5): 835–844. doi:10.2307/3318758.  
 Hall, P. and Tajvidi, N. (2004). Prediction regions for bivariate extreme events. *Australian & New Zealand Journal of Statistics* **46**(1): 99–112. doi:10.1111/j.1467842X.2004.00316.x.

## See Also

[NonparEstDepFct](#), [NewBEVSplineCopula](#)

**Examples**

```
## Data from Hall and Tajvidi (2004, ANZJS)
EstDF2 <- NonparEstDepFct(MaxTemp, convex = FALSE)

## Plot modified Pickands Function and area in which
## dependence function must lie
plot(EstDF2, ylim = c(0.5,1), xlab = "w", ylab = "A(w)", type="l", lty="longdash")
polygon(c(0, 0.5, 1, 0), c(1, 0.5, 1, 1))

## Fit spline to Pickands function and add to plot
splfit <- SplineFitDepFct(EstDF2)
curve(splfit, n = 301, add = TRUE, lty = "dashed")
```

# Index

## \* datasets

MaxTemp, [5](#)

## \* distribution

GenerateRV, [2](#)

GenerateRV.CopApprox, [2](#)

NewBEVAsyLogisticCopula, [6](#)

NewBEVAsyMixedModelCopula, [7](#)

NewBEVLogisticCopula, [8](#)

NewBEVMixedModelCopula, [9](#)

NewBEVSplineCopula, [10](#)

NewMEVAsyLogisticCopula, [10](#)

NewMEVGumbelCopula, [12](#)

NewMVClaytonCopula, [13](#)

NewMVFrankCopula, [13](#)

## \* hplot

plot.CopApprox, [16](#)

plot.SimCop, [17](#)

## \* nonparametric

NonparEstDepFct, [14](#)

## \* print

print.SimCop, [19](#)

## \* smooth

SplineFitDepFct, [20](#)

cat, [15](#)

class, [2](#), [5](#), [16](#), [18](#), [19](#)

GenerateRV, [2](#)

GenerateRV.CopApprox, [2](#)

GetApprox, [3](#), [4](#), [16](#)

invisibly, [17](#), [18](#)

MaxTemp, [5](#)

NewBEVAsyLogisticCopula, [6](#), [8](#), [11](#)

NewBEVAsyMixedModelCopula, [7](#), [9](#)

NewBEVLogisticCopula, [6](#), [8](#)

NewBEVMixedModelCopula, [7](#), [9](#)

NewBEVSplineCopula, [10](#), [20](#)

NewMEVAsyLogisticCopula, [6](#), [10](#), [12](#)

NewMEVGumbelCopula, [8](#), [11](#), [12](#)

NewMVClaytonCopula, [13](#)

NewMVFrankCopula, [13](#)

NonparEstDepFct, [14](#), [20](#)

persp3d, [17](#), [18](#)

persp3d.SimCop (plot.SimCop), [17](#)

plot, [5](#), [16](#), [19](#)

plot.CopApprox, [5](#), [16](#)

plot.SimCop, [17](#)

polygon, [16](#)

print, [19](#)

print.SimCop, [19](#)

quantile, [16](#), [18](#)

rgl, [16](#)

SimCop, [19](#)

SimCop-package (SimCop), [19](#)

solve.QP, [20](#)

SplineFitDepFct, [10](#), [15](#), [20](#)

splinefun, [20](#)

xy.coords, [14](#), [20](#)