Package ‘SimCop’

January 20, 2025

Type Package

Title Simulate from Arbitrary Copulae

Version 0.7.0

Date 2017-12-16

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.

Depends R (>= 3.4.0)

Imports grDevices, graphics, quadprog, stats

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

NeedsCompilation no

Author Berwin A. Turlach [aut, cre],
Nader Tajvidi [ctb]

Maintainer Berwin A. Turlach <Berwin.Turlach@gmail.com>

Repository CRAN

Date/Publication 2017-12-20 12:07:17 UTC

Contents

`GenerateRV` *GenerateRV generic*

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` *Generates random variates from a copula approximation*

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 thining 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 (2017).

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. (2017). A general approach to generate random variates for multivariate copulae, *Australian & New Zealand Journal of Statistics*. 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 devide 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) dividies 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 futher 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. (2017). A general approach to generate random variates for multivariate copulae, *Australian & New Zealand Journal of Statistics*. Doi: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.1467-842X.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, θ and ϕ.

Usage

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

Arguments

`r`	real.
`theta`	real.
`phi`	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

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

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, θ 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 differentation.

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 ϕ and θ.

Usage

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

Arguments

`theta`	real.
`phi`	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 ϕ and θ) 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 differentation.

Author(s)

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

See Also

`NewBEVMixedModelCopula`

NewBEVLogisticCopula *Creates a bivariate logistic model extreme value copula*

Description

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

Usage

```
NewBEVLogisticCopula(r)
```

Arguments

`r`	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 differentation.

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 θ.

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 θ) 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 differentation.

Author(s)

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

See Also

NewBEVAsyMixedModelCopula

NewBEVSplineCopula *Creates a flexible extreme value copula*

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 differentation.

Author(s)

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

See Also

SplineFitDepFct

NewMEVAsyLogisticCopula *Creates a multivariate asymmetric logistic copula*

Description

Creates an instance of the multivariate asymmetric copula with parameters θ 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 θ_{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 θ 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 differentation.

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` *Creates a Gumpel copula*

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:

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

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

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 differentation.

Author(s)

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

See Also

`NewMEVAsyLogisticCopula`

`NewMVClaytonCopula` *Creates a Clayton copula*

Description

Creates an instance of the Clayton copula with parameter θ.

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:

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

Value

A function that evaluates the Clayton copula (with parameter α) 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 differentation.

Author(s)

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

`NewMVFrankCopula` *Creates a Frank copula*

Description

Creates an instance of the Frank copula with parameter α.

Usage

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

Arguments

`alpha`	real, the parameter of the Frank copula, defaults to 2; 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 α) 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 differentation.

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

- `x, y` vectors giving the observations of the extreme values. Alternatively a single plotting structure can be specified: see `xy.coords`.
- `w.length` number of grid points (using an equidistant grid from 0 to 1) on which the dependence function is estimated.
- `transf.to.frechet` logical, controls whether `x` and `y` are first transformed to have standard Fréchet margins: see 'Details'; defaults to `TRUE`.
- `convex.hull` logical, controls whether the convex hull of the modified Pickands estimator is returned; defaults to `TRUE`.
- `verbose` 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.1467-842X.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, ...)
```

Arguments

x	an object of class 'CopApprox'.
...	not used.

Author(s)

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

References

Tajvidi, N. and Turlach, B.A. (2017). A general approach to generate random variates for multivariate copulae, *Australian & New Zealand Journal of Statistics*. Doi:10.1111/anzs.12209.

Examples

```
Cop <- NewMEVGumbelCopula(4)
CopApprox1 <- GetApprox(Cop, dim=2)
plot(CopApprox1)
CopApprox2 <- GetApprox(Cop, dim=2, type=2)
plot(CopApprox2)
```

print.SimCop — *Print basic information on a copula*

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 `class` 'SimCop'.
`...`	not used.

Author(s)

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

`SimCop` *SimCop: A package to simulate random variates from an arbitrary multivariate copula*

Description

R code to support Tajvidi and Turlach (2017). 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'.

References

Tajvidi, N. and Turlach, B.A. (2017). A general approach to generate random variates for multivariate copulae, *Australian & New Zealand Journal of Statistics*. 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 xy.coords.
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 integeral (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 programms 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.1467-842X.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