Package ‘VineCopula’
June 5, 2015

Type Package
Title Statistical Inference of Vine Copulas
Version 1.5
Date 2015-06-05
Author Ulf Schepsmeier, Jakob Stoebner, Eike Christian Brechmann, Benedikt Graeler, Thomas Nagler, Tobias Erhardt
Maintainer Tobias Erhardt <tobias.erhardt@tum.de>
Depends R (>= 2.11.0)
Imports MASS, mvtnorm, igraph, methods, copula, ADGofTest, lattice
Suggests CDVine, TSP
Description Tools for bivariate exploratory data analysis, bivariate copula selection and (vine) tree construction are provided. Vine copula models can be estimated either sequentially or by joint maximum likelihood estimation. Sampling algorithms and plotting methods are included. Data is assumed to lie in the unit hypercube (so-called copula data). For C- and D-vines links to the package ‘CDVine’ are provided.
License GPL (>= 2)
LazyLoad yes
NeedsCompilation yes
Archs i386, x64

R topics documented:

VineCopula-package .................................................. 3
as.copuladata ...................................................... 6
BB1Copula ............................................................ 7
BB1Copula-class .................................................... 8
BB6Copula ............................................................ 9
BB6Copula-class ................................................... 10
BB7Copula ........................................................... 11
BB7Copula-class ................................................... 12
BB8Copula ........................................................... 13
BB8Copula-class ................................................... 14
BetaMatrix .......................................................... 15
BiCop ................................................................. 16
BiCopCDF ............................................................ 17
**Description**

Tools for bivariate exploratory data analysis, bivariate copula selection and (vine) tree construction are provided. Vine copula models can be estimated either sequentially or by joint maximum likelihood estimation. Sampling algorithms and plotting methods are included. Data is assumed to lie in the unit hypercube (so-called copula data). For C- and D-vines links to the package ‘CDVine’ are provided.

**Details**

- **Package:** VineCopula
- **Type:** Package
- **Version:** 1.5
- **Date:** 2015-06-05
- **License:** GPL (>=2)
- **Depends:** R (>= 2.11.0)
- **Imports:** MASS, mvtnorm, igraph, methods, copula, ADGofTest, lattice
- **Suggests:** CDVine, TSP
- **LazyLoad:** yes

**Remark**

The package VineCopula is a continuation of the package CDVine by U. Schepsmeier and E. C. Brechmann (see Brechmann and Schepsmeier (2013)). It includes all functions implemented in CDVine for the bivariate case (BiCop-functions).

**Bivariate copula families**

In this package several bivariate copula families are included for bivariate analysis as well as for multivariate analysis using vine copulas. It provides functionality of elliptical (Gaussian and Student-t) as well as Archimedean (Clayton, Gumbel, Frank, Joe, BB1, BB6, BB7 and BB8) copulas to cover a large bandwidth of possible dependence structures. For the Archimedean copula
families rotated versions are included to cover negative dependence too. The two parameter BB1, BB6, BB7 and BB8 copulas are however numerically instable for large parameters, in particular, if BB6, BB7 and BB8 copulas are close to the Joe copula which is a boundary case of these three copula families. In general, the user should be careful with extreme parameter choices.

As an asymmetric extension of the Gumbel copula, the Tawn copula with three parameters is also included in the package. Both the Gumbel and the Tawn copula are extreme-value copulas, which can be defined in terms of their corresponding Pickands dependence functions. For simplicity, we implemented two versions of the Tawn copula with two parameters each. Each type has one of the asymmetry parameters fixed to 1, so that the corresponding Pickands dependence is either left- or right-skewed. In the manual we will call these two new copulas “Tawn type 1” and “Tawn type 2”. The following table shows the parameter ranges of bivariate copula families with parameters par and par2:

<table>
<thead>
<tr>
<th>Copula family</th>
<th>par</th>
<th>par2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian</td>
<td>(-1,1)</td>
<td>-</td>
</tr>
<tr>
<td>Student t Cltayton</td>
<td>(-1,1)</td>
<td>(2,∞)</td>
</tr>
<tr>
<td>(Survival) Gumbel</td>
<td>(0,∞)</td>
<td>-</td>
</tr>
<tr>
<td>Frank</td>
<td>R1{0}</td>
<td>-</td>
</tr>
<tr>
<td>(Survival) Joe</td>
<td>(1,∞)</td>
<td>-</td>
</tr>
<tr>
<td>Rotated Clayton (90 and 270 degrees)</td>
<td>(-∞,0)</td>
<td>-</td>
</tr>
<tr>
<td>Rotated Gumbel (90 and 270 degrees)</td>
<td>(-∞,-1]</td>
<td>-</td>
</tr>
<tr>
<td>Rotated Joe (90 and 270 degrees)</td>
<td>(-∞,-1]</td>
<td>-</td>
</tr>
<tr>
<td>(Survival) Clayton-Gumbel (BB1)</td>
<td>(0,∞)</td>
<td>[1,∞)</td>
</tr>
<tr>
<td>(Survival) Joe-Gumbel (BB6)</td>
<td>[1,∞)</td>
<td>[1,∞)</td>
</tr>
<tr>
<td>(Survival) Joe-Clayton (BB7)</td>
<td>[1,∞)</td>
<td>(0,∞)</td>
</tr>
<tr>
<td>(Survival) Joe-Frank (BB8)</td>
<td>[1,∞)</td>
<td>(0,1]</td>
</tr>
<tr>
<td>Rotated Clayton-Gumbel (90 and 270 degrees)</td>
<td>(-∞,0)</td>
<td>(-∞,-1]</td>
</tr>
<tr>
<td>Rotated Joe-Gumbel (90 and 270 degrees)</td>
<td>(-∞,-1]</td>
<td>(-∞,-1]</td>
</tr>
<tr>
<td>Rotated Joe-Clayton (90 and 270 degrees)</td>
<td>(-∞,-1]</td>
<td>(-∞,0)</td>
</tr>
<tr>
<td>Rotated Joe-Frank (90 and 270 degrees)</td>
<td>(-∞,-1]</td>
<td>[-1,0]</td>
</tr>
<tr>
<td>(Survival) Tawn type 1 and type 2</td>
<td>[1,∞)</td>
<td>[0,1]</td>
</tr>
<tr>
<td>Rotated Tawn type 1 and type 2 (90 and 270 degrees)</td>
<td>(-∞,-1]</td>
<td>[0,1]</td>
</tr>
</tbody>
</table>

R-vine copula models

The specification of an R-vine is done in matrix notation, introduced by Dissmann et al. (2013). One matrix contains the R-vine tree structure, one the copula families utilized and two matrices corresponding parameter values. These four matrices are stored in an \texttt{rvinematrix} object created by the function \texttt{rvinematrix}. Each matrix is a d x d lower triangular matrix. Since C- and D-vines are special cases, boundary cases, of R-vines one can write each C- or D-vine in R-vine notation. The transformation of notation to an R-vine can be done via \texttt{cRrvine} and \texttt{dRrvine}, which provide an interface to the package CDVine. For more details see the documentation of the functions.

Acknowledgment

We acknowledge substantial contributions by our working group at Technische Universitaet Muenchen, in particular by Carlos Almeida and Aleksey Min. In addition, we like to thank Shing (Eric) Fu, Feng Zhu, Guang (Jack) Yang, and Harry Joe for providing their implementation of the method by Knight (1966) for efficiently computing the empirical Kendall’s tau. We are especially grateful to Harry Joe for his contributions to the implementation of the bivariate Archimedian copulas.
Author(s)

Ulf Schepsmeier, Jakob Stoeber, Eike Christian Brechmann, Benedikt Graeler, Thomas Nagler, Tobias Erhardt

References


as.copuladata

Copula Data Objects

Description

The function as.copuladata coerces an object (data.frame, matrix, list) to a copuladata object.

Usage

as.copuladata(data)

Arguments

data

Either a data.frame, a matrix or a list containing copula data (i.e. data with uniform margins on [0,1]). The list elements have to be vectors of identical length.

Author(s)

Tobias Erhardt

See Also

pobs, pairs.copuladata

Examples

data(daxreturns)

data <- as(daxreturns, "matrix")
class(as.copuladata(data))

data <- as(daxreturns, "data.frame")
class(as.copuladata(data))

data <- as(daxreturns, "list")
names(data) <- names(daxreturns)
class(as.copuladata(data))
BB1Copula

Constructor of the BB1 Family and Rotated Versions thereof

Description

Constructs an object of the BB1Copula (survival sur, 90 degree rotated r90 and 270 degree rotated r270) family for given parameters.

Usage

BB1Copula(param)
surBB1Copula(param)
r90BB1Copula(param)
r270BB1Copula(param)

Arguments

param The parameter param defines the copula through theta and delta.

Value

One of the respective BB1 copula classes (BB1Copula, surBB1Copula, r90BB1Copula, r270BB1Copula).

Author(s)

Benedikt Graeler

References


See Also

See also BB6Copula, BB7Copula, BB8Copula and joecopula for further wrapper functions to the VineCopula-package.

Examples

library(copula)
persp(BB1Copula(c(1,1.5)), dCopula, zlim = c(0,10))
persp(surBB1Copula(c(1,1.5)), dCopula, zlim = c(0,10))
persp(r90BB1Copula(c(-1,-1.5)), dCopula, zlim = c(0,10))
persp(r270BB1Copula(c(-1,-1.5)), dCopula, zlim = c(0,10))
BB1Copula-class

Classes "BB1Copula", "surBB1Copula", "r90BB1Copula" and "r270BB1Copula"

Description

Wrapper classes representing the BB1, survival BB1, 90 degree and 270 degree rotated BB1 copula families (Joe 1997) from VineCopula-package.

Objects from the Classes

Objects can be created by calls of the form new("BB1Copula", ...), new("surBB1Copula", ...), new("r90BB1Copula", ...) and new("r270BB1Copula", ...) or by the functions BB1Copula, surBB1Copula, r90BB1Copula and r270BB1Copula.

Slots

family: Object of class "numeric" defining the family number in VineCopula-package
dimension: Object of class "integer" defining the dimension of the copula
parameters: Object of class "numeric" the two-place parameter vector
param.names: Object of class "character", parameter names.
param.lowbnd: Object of class "numeric", lower bounds of the copula parameters
param.upbnd: Object of class "numeric", upper bounds of the copula parameters
fullname: Object of class "character", family name of the copula.

Extends

Class "copula", directly. Class "Copula", by class "copula", distance 2.

Methods

dduCopula signature(u = "matrix", copula = "BB1Copula"): ...
dduCopula signature(u = "numeric", copula = "BB1Copula"): ...
ddvCopula signature(u = "matrix", copula = "BB1Copula"): ...
ddvCopula signature(u = "numeric", copula = "BB1Copula"): ...
getKendallDistr signature(copula = "BB1Copula"): ...
kendallDistribution signature(copula = "BB1Copula"): ...

Author(s)

Benedikt Graeler

References

BB6Copula

See Also
See also BB6Copula, BB7Copula, BB8Copula and joeCopula for further wrapper classes to the VineCopula-package.

Examples
showClass("BB1Copula")

BB6Copula Constructor of the BB6 Family and Rotated Versions thereof

Description
Constructs an object of the BB6Copula (survival sur, 90 degree rotated \( r^90 \) and 270 degree rotated \( r^\text{270} \) family for given parameters.

Usage
BB6Copula(param)
surBB6Copula(param)
r90BB6Copula(param)
r270BB6Copula(param)

Arguments
param The parameter \( \text{param} \) defines the copula through \( \theta \) and \( \delta \).

Value
One of the respective BB6 copula classes (BB6Copula, surBB6Copula, r90BB6Copula, r270BB6Copula).

Author(s)
Benedikt Graeler

References

See Also
See also BB6Copula, BB7Copula, BB8Copula and joeCopula for further wrapper functions to the VineCopula-package.

Examples
library(copula)
persp(BB6Copula(c(1,1.5)), dCopula, zlim = c(0,10))
persp(surBB6Copula(c(1,1.5)), dCopula, zlim = c(0,10))
persp(r90BB6Copula(c(-1,-1.5)), dCopula, zlim = c(0,10))
persp(r270BB6Copula(c(-1,-1.5)), dCopula, zlim = c(0,10))
BB6Copula-class

Description

Wrapper classes representing the BB6, survival BB6, 90 degree and 270 degree rotated BB6 copula families (Joe 1997) from the VineCopula-package.

Objects from the Classes

Objects can be created by calls of the form `new("BB6Copula", ...), new("surBB6Copula", ...), new("r90BB6Copula", ...) and new("r270BB6Copula", ...)` or by the functions `BB6Copula, surBB6Copula, r90BB6Copula` and `r270BB6Copula`.

Slots

- family: Object of class "numeric" defining the family number in VineCopula-package
- dimension: Object of class "integer" defining the dimension of the copula
- parameters: Object of class "numeric" the two-place parameter vector
- param.names: Object of class "character", parameter names.
- param.lowbd: Object of class "numeric", lower bounds of the copula parameters
- param.upbd: Object of class "numeric", upper bounds of the copula parameters
- fullname: Object of class "character", family name of the copula.

Extends

Class "copula", directly. Class "Copula", by class "copula", distance 2.

Methods

- `dduCopula` signature(u = "matrix", copula = "BB6Copula"): ...
- `dduCopula` signature(u = "numeric", copula = "BB6Copula"): ...
- `ddvCopula` signature(u = "matrix", copula = "BB6Copula"): ...
- `ddvCopula` signature(u = "numeric", copula = "BB6Copula"): ...
- `getKendallDistr` signature(copula = "BB6Copula"): ...
- `kendallDistribution` signature(copula = "BB6Copula"): ...

Author(s)

Benedikt Graeler

References


**BB7Copula**

**See Also**

See also **BB1Copula**, **BB7Copula**, **BB8Copula** and **joeCopula** for further wrapper classes to the VineCopula-package.

**Examples**

```r
showClass("BB6Copula")
```

---

**BB7Copula**  
*Constructor of the BB7 Family and Rotated Versions thereof*

**Description**

Constructs an object of the **BB7Copula** (survival sur, 90 degree rotated r90 and 270 degree rotated r270) family for given parameters.

**Usage**

```
BB7Copula(param)  
surBB7Copula(param)  
r90BB7Copula(param)  
r270BB7Copula(param)
```

**Arguments**

- `param`  
The parameter `param` defines the copula through `theta` and `delta`.

**Value**

One of the respective BB7 copula classes (**BB7Copula**, **surBB7Copula**, **r90BB7Copula**, **r270BB7Copula**).

**Author(s)**

Benedikt Graeler

**References**


**See Also**

See also **BB6Copula**, **BB7Copula**, **BB8Copula** and **joeCopula** for further wrapper functions to the VineCopula-package.

**Examples**

```r
library(copula)

persp(BB7Copula(c(1,1.5)), dCopula, zlim = c(0,10))  
persp(surBB7Copula(c(1,1.5)), dCopula, zlim = c(0,10))  
persp(r90BB7Copula(c(-1,-1.5)), dCopula, zlim = c(0,10))  
persp(r270BB7Copula(c(-1,-1.5)), dCopula, zlim = c(0,10))
```
BB7Copula-class

Classes "BB7Copula", "surBB7Copula", "r90BB7Copula" and "r270BB7Copula"

Description

Wrapper classes representing the BB7, survival BB7, 90 degree and 270 degree rotated BB7 copula families (Joe 1997) from the VineCopula-package package.

Objects from the Classes

Objects can be created by calls of the form `new("BB7Copula", ...)`, `new("surBB7Copula", ...)`, `new("r90BB7Copula", ...)` and `new("r270BB7Copula", ...)` or by the functions `BB7Copula`, `surBB7Copula`, `r90BB7Copula` and `r270BB7Copula`.

Slots

family: Object of class "numeric" defining the family number in VineCopula-package
dimension: Object of class "integer" defining the dimension of the copula
parameters: Object of class "numeric" the two-place parameter vector
param.names: Object of class "character", parameter names.
param.lwbnd: Object of class "numeric", lower bounds of the copula parameters
param.upbnd: Object of class "numeric", upper bounds of the copula parameters
fullname: Object of class "character", family name of the copula.

Extends

Class "copula", directly. Class "Copula", by class "copula", distance 2.

Methods

`dduCopula` signature(u = "matrix", copula = "BB7Copula"): ...
`dduCopula` signature(u = "numeric", copula = "BB7Copula"): ...
`ddvCopula` signature(u = "matrix", copula = "BB7Copula"): ...
`ddvCopula` signature(u = "numeric", copula = "BB7Copula"): ...
`getKendallDistr` signature(copula = "BB7Copula"): ...
`kendallDistribution` signature(copula = "BB7Copula"): ...

Author(s)

Benedikt Graeler

References

See Also

See also `BB1Copula`, `BB6Copula`, `BB8Copula` and `joeCopula` for further wrapper classes to the `VineCopula-package`.

Examples

```r
showClass("BB7Copula")
```

---

**BB8Copula**  
*Constructor of the BB8 Family and Rotated Versions thereof*

**Description**

Constructs an object of the **BB8Copula** (survival sur, 90 degree rotated $r90$ and 270 degree rotated $r270$) family for given parameters.

**Usage**

```r
BB8Copula(param)  
survBB8Copula(param)  
r90BB8Copula(param)  
r270BB8Copula(param)
```

**Arguments**

- **param**  
The parameter `param` defines the copula through theta and delta.

**Value**

One of the respective BB8 copula classes (`BB8Copula`, `survBB8Copula`, `r90BB8Copula`, `r270BB8Copula`).

**Author(s)**

Benedikt Graeler

**References**


**See Also**

See also `BB6Copula`, `BB7Copula`, `BB8Copula` and `joeCopula` for further wrapper functions to the `VineCopula-package`.

**Examples**

```r
library(copula)

persp(BB8Copula(c(2,0.9)), dCopula, zlim = c(0,1))
persp(survBB8Copula(c(2,0.9)), dCopula, zlim = c(0,1))
persp(r90BB8Copula(c(-2,-0.9)), dCopula, zlim = c(0,1))
persp(r270BB8Copula(c(-2,-0.9)), dCopula, zlim = c(0,1))
```
BB8Copula-class

Classes "BB8Copula", "surBB8Copula", "r90BB8Copula" and "r270BB8Copula"

Description

Wrapper classes representing the BB8, survival BB8, 90 degree and 270 degree rotated BB8 copula families (Joe 1997) from the VineCopula-package package.

Objects from the Classes

Objects can be created by calls of the form new("BB8Copula", ...), new("surBB8Copula", ...), new("r90BB8Copula", ...) and new("r270BB8Copula", ...) or by the functions BB8Copula, surBB8Copula, r90BB8Copula and r270BB8Copula.

Slots

family: Object of class "numeric" defining the family number in VineCopula-package

dimension: Object of class "integer" defining the dimension of the copula

d parameters: Object of class "numeric" the two-place parameter vector

d param.names: Object of class "character", parameter names.

d param.lowbnd: Object of class "numeric", lower bounds of the copula parameters

d param.upbnd: Object of class "numeric", upper bounds of the copula parameters

fullname: Object of class "character", family name of the copula.

Extends

Class "copula", directly. Class "Copula", by class "copula", distance 2.

Methods

dduCopula signature(u = "matrix", copula = "BB8Copula"): ...

dduCopula signature(u = "numeric", copula = "BB8Copula"): ...

ddvCopula signature(u = "matrix", copula = "BB8Copula"): ...

ddvCopula signature(u = "numeric", copula = "BB8Copula"): ...

getKendallDistr signature(copula = "BB8Copula"): ...

kendallDistribution signature(copula = "BB8Copula"): ...

Author(s)

Benedikt Graeler

References

See Also

See also BB1Copula, BB6Copula, BB7Copula and joecopula for further wrapper classes to the VineCopula-package.

Examples

showClass("BB8Copula")

---

**BetaMatrix**

*Matrix of Empirical Blomqvist’s Beta Values*

**Description**

This function computes the empirical Blomqvist’s beta.

**Usage**

BetaMatrix(data)

**Arguments**

data An N x d data matrix.

**Value**

Matrix of the empirical Blomqvist’s betas.

**Author(s)**

Ulf Schepsmeier

**References**


**See Also**

TauMatrix, BiCopPar2Beta, RVinePar2Beta

**Examples**

data(daxreturns)
Data <- as.matrix(daxreturns)

# compute the empirical Blomqvist's betas
BetaMatrix(Data)
Description
This function creates an object of class BiCop and checks for family/parameter consistency.

Usage
BiCop(family, par, par2 = 0)

Arguments
family An integer defining the bivariate copula family:
0 = independence copula
1 = Gaussian copula
2 = Student t copula (t-copula)
3 = Clayton copula
4 = Gumbel copula
5 = Frank copula
6 = Joe copula
7 = BB1 copula
8 = BB6 copula
9 = BB7 copula
10 = BB8 copula
13 = rotated Clayton copula (180 degrees; “survival Clayton”)  
14 = rotated Gumbel copula (180 degrees; “survival Gumbel”)  
16 = rotated Joe copula (180 degrees; “survival Joe”)  
17 = rotated BB1 copula (180 degrees; “survival BB1”)  
18 = rotated BB6 copula (180 degrees; “survival BB6”)  
19 = rotated BB7 copula (180 degrees; “survival BB7”)  
20 = rotated BB8 copula (180 degrees; “survival BB8”)  
23 = rotated Clayton copula (90 degrees)  
24 = rotated Gumbel copula (90 degrees)  
26 = rotated Joe copula (90 degrees)  
27 = rotated BB1 copula (90 degrees)  
28 = rotated BB6 copula (90 degrees)  
29 = rotated BB7 copula (90 degrees)  
30 = rotated BB8 copula (90 degrees)  
33 = rotated Clayton copula (270 degrees)  
34 = rotated Gumbel copula (270 degrees)  
36 = rotated Joe copula (270 degrees)  
37 = rotated BB1 copula (270 degrees)  
38 = rotated BB6 copula (270 degrees)  
39 = rotated BB7 copula (270 degrees)  
40 = rotated BB8 copula (270 degrees)  
104 = Tawn type 1 copula  
114 = rotated Tawn type 1 copula (180 degrees)  
124 = rotated Tawn type 1 copula (90 degrees)  
134 = rotated Tawn type 1 copula (270 degrees)  
204 = Tawn type 2 copula
214 = rotated Tawn type 2 copula (180 degrees)
224 = rotated Tawn type 2 copula (90 degrees)
234 = rotated Tawn type 2 copula (270 degrees)

par  
Copula parameter.

par2  
Second parameter for bivariate copulas with two parameters (t, BB1, BB6, BB7, BB8, Tawn type 1 and type 2; default: par2 = 0). par2 should be an positive integer for the Students’s t copula family = 2.

Value
An object of class BiCop, i.e., a list containing

family  
Copula family
par, par2  
Copula parameter(s).

Objects of this class are returned by the BiCopEst and BiCopSelect functions.

Author(s)
Thomas Nagler

See Also

BiCopPDF, BiCopHfunc, BiCopSim, BiCopEst, BiCopSelect, plot.BiCop

Examples

## create BiCop object for bivariate t-copula
obj <- BiCop(family = 2, par = 0.4, par2 = 6)

## a selection of function that can be used with BiCop objects
simdata <- BiCopSim(300, obj) # simulate data
BiCopPDF(0.5, 0.5, obj) # evaluate density in (0.5,0.5)
plot(obj) # normal contour plot

---

**BiCopCDF**

*Distribution Function of a Bivariate Copula*

Description

This function evaluates the cumulative distribution function (CDF) of a given parametric bivariate copula.

Usage

BiCopCDF(u1, u2, family, par, par2 = 0, obj = NULL)
Arguments

u1, u2 Numeric vectors of equal length with values in [0,1].

family An integer defining the bivariate copula family:
0 = independence copula
1 = Gaussian copula
3 = Clayton copula
4 = Gumbel copula
5 = Frank copula
6 = Joe copula
7 = BB1 copula
8 = BB6 copula
9 = BB7 copula
10 = BB8 copula
13 = rotated Clayton copula (180 degrees; “survival Clayton”)  
14 = rotated Gumbel copula (180 degrees; “survival Gumbel”)  
16 = rotated Joe copula (180 degrees; “survival Joe”)  
17 = rotated BB1 copula (180 degrees; "survival BB1")  
18 = rotated BB6 copula (180 degrees; "survival BB6")  
19 = rotated BB7 copula (180 degrees; "survival BB7")  
20 = rotated BB8 copula (180 degrees; "survival BB8")  
23 = rotated Clayton copula (90 degrees)  
24 = rotated Gumbel copula (90 degrees) 
26 = rotated Joe copula (90 degrees) 
27 = rotated BB1 copula (90 degrees) 
28 = rotated BB6 copula (90 degrees) 
29 = rotated BB7 copula (90 degrees) 
30 = rotated BB8 copula (90 degrees) 
33 = rotated Clayton copula (270 degrees) 
34 = rotated Gumbel copula (270 degrees) 
36 = rotated Joe copula (270 degrees) 
37 = rotated BB1 copula (270 degrees) 
38 = rotated BB6 copula (270 degrees) 
39 = rotated BB7 copula (270 degrees) 
40 = rotated BB8 copula (270 degrees) 
104 = Tawn type 1 copula 
114 = rotated Tawn type 1 copula (180 degrees)  
124 = rotated Tawn type 1 copula (90 degrees) 
134 = rotated Tawn type 1 copula (270 degrees) 
204 = Tawn type 2 copula 
214 = rotated Tawn type 2 copula (180 degrees) 
224 = rotated Tawn type 2 copula (90 degrees) 
234 = rotated Tawn type 2 copula (270 degrees)

par Copula parameter.

par2 Second parameter for bivariate copulas with two parameters (t, BB1, BB6, BB7, BB8, Tawn type 1 and type 2; default: par2 = 0). par2 should be an positive integer for the Students’s t copula family = 2.

obj BiCop object containing the family and parameter specification.
Details
If the family and parameter specification is stored in a BiCop object obj, the alternative version

BiCopCDF(u1, u2, obj)

can be used.

Value
A numeric vector of the bivariate copula distribution function evaluated at u1 and u2.

Note
The calculation of the cumulative distribution function (CDF) of the Student’s t copula (family = 2)
is not implemented any more since the calculation was wrong for non-integer degrees-of-freedom.

Author(s)
Eike Brechmann

See Also
BiCopPDF, BiCopHfunc, BiCopSim, BiCop

Examples
```r
## simulate from a bivariate Clayton
simdata <- BiCopSim(300, 3, 3.4)

## evaluate the distribution function of the bivariate Clayton copula
u1 <- simdata[,1]
u2 <- simdata[,2]
BiCopCDF(u1, u2, 3, 3.4)

## estimate a bivariate copula from the data and evaluate its CDF
cop <- BiCopSelect(u1, u2)
BiCopCDF(u1, u2, cop)
```

---

**Chi-plot for Bivariate Copula Data**

Description
This function creates a chi-plot of given bivariate copula data.

Usage
BiCopChiPlot(u1, u2, PLOT = TRUE, mode = "NULL", ...)
BiCopChiPlot

**Arguments**

- `u1,u2` Data vectors of equal length with values in [0,1].
- `PLOT` Logical; whether the results are plotted. If `PLOT = FALSE`, the values `lambda`, `chi` and `control.bounds` are returned (see below; default: `PLOT = TRUE`).
- `mode` Character; whether a general, lower or upper chi-plot is calculated. Possible values are `mode = "NULL"`, `"upper"` and `"lower"`. "NULL" = general chi-plot (default) "upper" = upper chi-plot "lower" = lower chi-plot
- `...` Additional plot arguments.

**Details**

For observations \( u_{i,j} \), \( i = 1, \ldots, N, \ j = 1, 2 \), the chi-plot is based on the following two quantities: the chi-statistics

\[
\chi_i = \frac{\hat{F}_{U_1,U_2}(u_{i,1},u_{i,2}) - \hat{F}_{U_1}(u_{i,1})\hat{F}_{U_2}(u_{i,2})}{\sqrt{\hat{F}_{U_1}(u_{i,1})(1 - \hat{F}_{U_1}(u_{i,1}))\hat{F}_{U_2}(u_{i,2})(1 - \hat{F}_{U_2}(u_{i,2}))}},
\]

and the lambda-statistics

\[
\lambda_i = 4 \text{sgn} (\hat{F}_{U_1}(u_{i,1}), \hat{F}_{U_2}(u_{i,2})) \cdot \max (\hat{F}_{U_1}(u_{i,1})^2, \hat{F}_{U_2}(u_{i,2})^2),
\]

where \( \hat{F}_{U_1} \) and \( \hat{F}_{U_2} \) are the empirical distribution functions of the uniform random variables \( U_1 \) and \( U_2 \) and of \( (U_1,U_2) \), respectively. Further, \( \hat{F}_{U_1} = \hat{F}_{U_1} - 0.5 \) and \( \hat{F}_{U_2} = \hat{F}_{U_2} - 0.5 \).

These quantities only depend on the ranks of the data and are scaled to the interval \([0, 1]\). \( \lambda_i \) measures a distance of a data point \( (u_{i,1}, u_{i,2}) \) to the center of the bivariate data set, while \( \chi_i \) corresponds to a correlation coefficient between dichotomized values of \( U_1 \) and \( U_2 \). Under independence it holds that \( \chi_i \sim \mathcal{N}(0, \frac{1}{n}) \) and \( \lambda_i \sim \mathcal{U}[-1, 1] \) asymptotically, i.e., values of \( \chi_i \) close to zero indicate independence—corresponding to \( F_{U_1,U_2} = F_{U_1} F_{U_2} \).

When plotting these quantities, the pairs of \( (\lambda_i, \chi_i) \) will tend to be located above zero for positively dependent margins and vice versa for negatively dependent margins. Control bounds around zero indicate whether there is significant dependence present.

If `mode = "lower"` or `"upper"`, the above quantities are calculated only for those \( u_{i,1}'s \) and \( u_{i,2}'s \) which are smaller/larger than the respective means of \( u_{1,1} \) and \( u_{2,2} \).

**Value**

- `lambda` Lambda-statistics (x-axis).
- `chi` Chi-statistics (y-axis).
- `control.bounds` A 2-dimensional vector of bounds \((1.54/\sqrt{n}, -1.54/\sqrt{n})\), where \( n \) is the length of \( u_1 \) and where the chosen values correspond to an approximate significance level of 10%.

**Author(s)**

Natalia Belgorodski, Ulf Schepsmeier
BiCopDeriv

References


Genest, C. and A. C. Favre (2007). Everything you always wanted to know about copula modeling but were afraid to ask. Journal of Hydrologic Engineering, 12 (4), 347-368.

See Also

BiCopMetaContour, BiCopKPlot, BiCopLambda

Examples

# chi-plots for bivariate Gaussian copula data
n <- 500
tau <- 0.5

# simulate copula data
fam <- 1
theta <- BiCopTau2Par(fam, tau)
set.seed(123)
dat <- BiCopSim(n, fam, theta)

# create chi-plots
par(mfrow = c(1,3))
BiCopChiPlot(dat[,1], dat[,2], xlim = c(-1,1), ylim = c(-1,1),
           main="General chi-plot")
BiCopChiPlot(dat[,1], dat[,2], mode = "lower", xlim = c(-1,1),
           ylim = c(-1,1), main = "Lower chi-plot")
BiCopChiPlot(dat[,1], dat[,2], mode = "upper", xlim = c(-1,1),
           ylim = c(-1,1), main = "Upper chi-plot")

BiCopDeriv

Derivatives of a Bivariate Copula Density

Description

This function evaluates the derivative of a given parametric bivariate copula density with respect to its parameter(s) or one of its arguments.

Usage

BiCopDeriv(u1, u2, family, par, par2 = 0, deriv = "par", log = FALSE, obj = NULL)

Arguments

u1, u2 Numeric vectors of equal length with values in [0,1].
family An integer defining the bivariate copula family:
0 = independence copula
1 = Gaussian copula
2 = Student t copula (t-copula)
3 = Clayton copula
4 = Gumbel copula
5 = Frank copula
6 = Joe copula
13 = rotated Clayton copula (180 degrees; “survival Clayton”)
14 = rotated Gumbel copula (180 degrees; “survival Gumbel”)
16 = rotated Joe copula (180 degrees; “survival Joe”)
23 = rotated Clayton copula (90 degrees)
24 = rotated Gumbel copula (90 degrees)
26 = rotated Joe copula (90 degrees)
33 = rotated Clayton copula (270 degrees)
34 = rotated Gumbel copula (270 degrees)
36 = rotated Joe copula (270 degrees)

par
Copula parameter.

par2
Second parameter for bivariate t-copula; default: par2 = 0.

deriv
Derivative argument
"par" = derivative with respect to the first parameter (default)
"par2" = derivative with respect to the second parameter (only available for the t-copula)
"u1" = derivative with respect to the first argument u1
"u2" = derivative with respect to the second argument u2

log
Logical; if TRUE than the derivative of the log-likelihood is returned (default: log = FALSE; only available for the derivatives with respect to the parameter(s) (deriv = "par" or deriv = "par2").

obj
BiCop object containing the family and parameter specification.

Details
If the family and parameter specification is stored in a BiCop object obj, the alternative version

BiCopDeriv(u1, u2, obj, deriv = "par", log = FALSE)

can be used.

Value
A numeric vector of the bivariate copula derivative with respect to deriv evaluated at u1 and u2 with parameter(s) par and par2.

Author(s)
Ulf Schepsmeier

References

See Also
RVineGrad, RVineHessian, BiCopDeriv2, BiCopHfuncDeriv, BiCop
Examples

```r
## simulate from a bivariate t-copula
simdata <- BiCopSim(300, 2, -0.7, par2 = 4)

## derivative of the bivariate t-copula with respect to the first parameter
u1 <- simdata[,1]
u2 <- simdata[,2]
BiCopDeriv(u1, u2, 2, -0.7, par2 = 4, deriv = "par")

## estimate a bivariate copula from the data
## and evaluate its derivative w.r.t. the parameter
cop <- BiCopEst(u1, u2, family = 2)
BiCopDeriv(u1, u2, cop, deriv = "par")
```

---

**BiCopDeriv2**  
*Second Derivatives of a Bivariate Copula Density*

**Description**

This function evaluates the second derivative of a given parametric bivariate copula density with respect to its parameter(s) and/or its arguments.

**Usage**

```r
BiCopDeriv2(u1, u2, family, par, par2 = 0, deriv = "par", obj = NULL)
```

**Arguments**

- `u1,u2` Numeric vectors of equal length with values in [0,1].
- `family` An integer defining the bivariate copula family:
  - 0 = independence copula
  - 1 = Gaussian copula
  - 2 = Student t copula (t-copula)
  - 3 = Clayton copula
  - 4 = Gumbel copula
  - 5 = Frank copula
  - 6 = Joe copula
  - 13 = rotated Clayton copula (180 degrees; “survival Clayton”)
  - 14 = rotated Gumbel copula (180 degrees; “survival Gumbel”)
  - 16 = rotated Joe copula (180 degrees; “survival Joe”)
  - 23 = rotated Clayton copula (90 degrees)
  - 24 = rotated Gumbel copula (90 degrees)
  - 26 = rotated Joe copula (90 degrees)
  - 33 = rotated Clayton copula (270 degrees)
  - 34 = rotated Gumbel copula (270 degrees)
  - 36 = rotated Joe copula (270 degrees)
- `par` Copula parameter.
- `par2` Second parameter for bivariate t-copula; default: `par2 = 0`. 
Deriv argument
"par"  = second derivative with respect to the first parameter (default)
"par2" = second derivative with respect to the second parameter (only available for the t-copula)
"u1"  = second derivative with respect to the first argument u1
"u2"  = second derivative with respect to the second argument u2
"par1par2" = second derivative with respect to the first and second parameter (only available for the t-copula)
"par1u1" = second derivative with respect to the first parameter and the first argument
"par2u1" = second derivative with respect to the second parameter and the first argument (only available for the t-copula)
"par1u2" = second derivative with respect to the first parameter and the second argument
"par2u2" = second derivative with respect to the second parameter and the second argument (only available for the t-copula)

obj BiCop object containing the family and parameter specification.

Details
If the family and parameter specification is stored in a BiCop object obj, the alternative version

BiCopDeriv2(u1, u2, obj, deriv = "par")

can be used.

Value
A numeric vector of the second bivariate copula derivative with respect to deriv evaluated at u1 and u2 with parameter(s) par and par2.

Author(s)
Ulf Schepsmeier, Jakob Stoeber

References

See Also
RVineGrad, RVineHessian, BiCopDeriv, BiCopHfuncDeriv, BiCop

Examples

## simulate from a bivariate t-copula
simdata <- BiCopSim(300, 2, -0.7, par2 = 4)

## second derivative of the bivariate t-copula with respect to the first parameter
u1 <- simdata[,1]
u2 <- simdata[,2]
BiCopEst

Parameter Estimation for Bivariate Copula Data

Description

This function estimates the parameter(s) for a bivariate copula using either inversion of empirical Kendall’s tau for single parameter copula families or maximum likelihood estimation for one and two parameter copula families supported in this package.

Usage

BiCopEst(u1, u2, family, method = "mle", se = FALSE, max.df = 30, max.BB = list(BB1=c(5,6), BB6=c(6,6), BB7=c(5,6), BB8=c(6,1)), weights = NA)

Arguments

u1,u2
Data vectors of equal length with values in [0,1].

family
An integer defining the bivariate copula family:
0 = independence copula
1 = Gaussian copula
2 = Student t copula (t-copula)
3 = Clayton copula
4 = Gumbel copula
5 = Frank copula
6 = Joe copula
7 = BB1 copula
8 = BB6 copula
9 = BB7 copula
10 = BB8 copula
13 = rotated Clayton copula (180 degrees; “survival Clayton”)  
14 = rotated Gumbel copula (180 degrees; “survival Gumbel”)  
16 = rotated Joe copula (180 degrees; “survival Joe”)  
17 = rotated BB1 copula (180 degrees; “survival BB1”)  
18 = rotated BB6 copula (180 degrees; “survival BB6”)  
19 = rotated BB7 copula (180 degrees; “survival BB7”)  
20 = rotated BB8 copula (180 degrees; “survival BB8”)  
23 = rotated Clayton copula (90 degrees)  
24 = rotated Gumbel copula (90 degrees)  
26 = rotated Joe copula (90 degrees)  
27 = rotated BB1 copula (90 degrees)  
28 = rotated BB6 copula (90 degrees)  
29 = rotated BB7 copula (90 degrees)  
30 = rotated BB8 copula (90 degrees)
33 = rotated Clayton copula (270 degrees)
34 = rotated Gumbel copula (270 degrees)
36 = rotated Joe copula (270 degrees)
37 = rotated BB1 copula (270 degrees)
38 = rotated BB6 copula (270 degrees)
39 = rotated BB7 copula (270 degrees)
40 = rotated BB8 copula (270 degrees)
104 = Tawn type 1 copula
114 = rotated Tawn type 1 copula (180 degrees)
124 = rotated Tawn type 1 copula (90 degrees)
134 = rotated Tawn type 1 copula (270 degrees)
204 = Tawn type 2 copula
214 = rotated Tawn type 2 copula (180 degrees)
224 = rotated Tawn type 2 copula (90 degrees)
234 = rotated Tawn type 2 copula (270 degrees)

method Character indicating the estimation method: either maximum likelihood estimation (method = "mle"; default) or inversion of Kendall’s tau (method = "i tau").

For method = "i tau" only one parameter bivariate copula families can be used (family = 1, 3, 4, 5, 6, 13, 14, 16, 23, 24, 26, 33, 34 or 36).

se Logical; whether standard error(s) of parameter estimates is/are estimated (default: se = FALSE).

max.df Numeric; upper bound for the estimation of the degrees of freedom parameter of the t-copula (default: max.df = 30).

max.BB List; upper bounds for the estimation of the two parameters (in absolute values) of the BB1, BB6, BB7 and BB8 copulas (default: max.BB = list(BB1 = c(5, 6), BB6 = c(6, 6), BB7 = c(5, 6), BB8 = c(6, 1))).

weights Numerical; weights for each observation (optional).

Details

If method = "i tau", the function computes the empirical Kendall’s tau of the given copula data and exploits the one-to-one relationship of copula parameter and Kendall’s tau which is available for many one parameter bivariate copula families (see BiCopPar2Tau and BiCopTau2Par). The inversion of Kendall’s tau is however not available for all bivariate copula families (see above). If a two parameter copula family is chosen and method = "i tau", a warning message is returned and the MLE is calculated.

For method = "mle" copula parameters are estimated by maximum likelihood using starting values obtained by method = "i tau". If no starting values are available by inversion of Kendall’s tau, starting values have to be provided given expert knowledge and the boundaries max.df and max.BB respectively. Note: The MLE is performed via numerical maximazation using the L_BFGS-B method. For the Gaussian, the t- and the one-parametric Archimedean copulas we can use the gradients, but for the BB copulas we have to use finite differences for the L_BFGS-B method.

A warning message is returned if the estimate of the degrees of freedom parameter of the t-copula is larger than max.df. For high degrees of freedom the t-copula is almost indistinguishable from the Gaussian and it is advised to use the Gaussian copula in this case. As a rule of thumb max. df = 30 typically is a good choice. Moreover, standard errors of the degrees of freedom parameter estimate cannot be estimated in this case.

Value

An object of class BiCop, i.e., a list containing
family Copula family
par, par2 Estimated copula parameter(s).
se, se2 Standard error(s) of the parameter estimate(s) (if se = TRUE).

Author(s)
Ulf Schepsmeier, Eike Brechmann, Jakob Stoeber, Carlos Almeida

References

See Also
BiCopPar2Tau, BiCopTau2Par, RVineSeqEst, BiCopSelect, BiCop

Examples
```r
## Example 1: bivariate Gaussian copula
data <- BiCopSim(500, 1, 0.7)
ud <- data[,1]
v1 <- data[,2]

# empirical Kendalls tau
tau1 <- cor(ud, v1, method = "kendall")

# inversion of empirical Kendalls tau
BiCopPar2Par(1, tau1)
BiCopEst(ud, v1, family = 1, method = "itau")$par

# maximum likelihood estimate for comparison
BiCopEst(ud, v1, family = 1, method = "mle")$par

## Example 2: bivariate Clayton and survival Gumbel copulas
# simulate from a Clayton copula
data <- BiCopSim(500, 3, 2.5)
ud <- data[,1]
v2 <- data[,2]

# empirical Kendalls tau
tau2 <- cor(ud, v2, method = "kendall")

# inversion of empirical Kendalls tau for the Clayton copula
BiCopTau2Par(3, tau2)
BiCopEst(ud, v2, family = 3, method = "itau", se = TRUE)

# inversion of empirical Kendalls tau for the survival Gumbel copula
BiCopTau2Par(14, tau2)
BiCopEst(ud, v2, family = 14, method = "itau", se = TRUE)

# maximum likelihood estimates for comparison
BiCopEst(ud, v2, family = 3, method = "mle", se = TRUE)
BiCopEst(ud, v2, family = 14, method = "mle", se = TRUE)
```
BiCopGofTest

Goodness-of-Fit Test for Bivariate Copulas

Description

This function performs a goodness-of-fit test for bivariate copulas, either based on White’s information matrix equality (White 1982) as introduced by Huang and Prokhorov (2011) or based on Kendall’s process. It computes the test statistics and p-values.

Usage

BiCopGofTest(u1, u2, family, par = 0, par2 = 0, method = "white",
max.df = 30, B = 100, obj = NULL)

Arguments

- u1, u2: Numeric vectors of equal length with values in [0,1].
- family: An integer defining the bivariate copula family:
  0 = independence copula
  1 = Gaussian copula
  2 = Student t copula (t-copula) (only for method = "white"; see details)
  3 = Clayton copula
  4 = Gumbel copula
  5 = Frank copula
  6 = Joe copula (only for method = "kendall")
  7 = BB1 copula (only for method = "kendall")
  8 = BB6 copula (only for method = "kendall")
  9 = BB7 copula (only for method = "kendall")
  10 = BB8 copula (only for method = "kendall")
  11 = rotated Clayton copula (180 degrees; "survival Clayton")
  12 = rotated Gumbel copula (180 degrees; "survival Gumbel")
  13 = rotated Joe copula (180 degrees; "survival Joe")
  14 = rotated BB1 copula (180 degrees; "survival BB1")
  15 = rotated BB6 copula (180 degrees; "survival BB6")
  16 = rotated BB7 copula (180 degrees; "survival BB7")
  17 = rotated BB8 copula (180 degrees; "survival BB8")
  18 = rotated Clayton copula (90 degrees)
  19 = rotated Gumbel copula (90 degrees)
  20 = rotated Joe copula (90 degrees)
  21 = rotated BB1 copula (90 degrees; only for method = "kendall")
  22 = rotated BB6 copula (90 degrees; only for method = "kendall")
  23 = rotated BB7 copula (90 degrees; only for method = "kendall")
  24 = rotated BB8 copula (90 degrees; only for method = "kendall")
  25 = rotated Clayton copula (270 degrees)
  26 = rotated Gumbel copula (270 degrees)
  27 = rotated Joe copula (270 degrees)
  28 = rotated BB1 copula (270 degrees; only for method = "kendall")
  29 = rotated BB6 copula (270 degrees; only for method = "kendall")
  30 = rotated BB7 copula (270 degrees; only for method = "kendall")
  31 = rotated BB8 copula (270 degrees; only for method = "kendall")
  32 = rotated Clayton copula (270 degrees)
  33 = rotated Gumbel copula (270 degrees)
  34 = rotated Joe copula (270 degrees)
  35 = rotated BB1 copula (270 degrees; only for method = "kendall")
  36 = rotated BB6 copula (270 degrees; only for method = "kendall")
  37 = rotated BB7 copula (270 degrees; only for method = "kendall")
  38 = rotated BB8 copula (270 degrees; only for method = "kendall")
par  Copula parameter (optional).
par2  Second parameter for bivariate t-copula (optional); default: par2 = 0.
max.df Numeric; upper bound for the estimation of the degrees of freedom parameter of the t-copula (default: max.df = 30).
method A string indicating the goodness-of-fit method:
"white" = goodness-of-fit test based on White’s information matrix equality (default)
"kendall" = goodness-of-fit test based on Kendall’s process
b Integer; number of bootstrap samples (default: b = 100). For b = 0 only the test statistics are returned.
WARNING: If b is chosen too large, computations will take very long.
obj BiCop object containing the family and parameter specification.

Details

method = "white":
This goodness-of-fit test uses the information matrix equality of White (1982) and was investigated by Huang and Prokhorov (2011). The main contribution is that under correct model specification the Fisher Information can be equivalently calculated as minus the expected Hessian matrix or as the expected outer product of the score function. The null hypothesis is

\[ H_0: H(\theta) + C(\theta) = 0 \]

against the alternative

\[ H_0: H(\theta) + C(\theta) \neq 0, \]

where \( H(\theta) \) is the expected Hessian matrix and \( C(\theta) \) is the expected outer product of the score function. For the calculation of the test statistic we use the consistent maximum likelihood estimator \( \hat{\theta} \) and the sample counter parts of \( H(\theta) \) and \( C(\theta) \). The correction of the covariance-matrix in the test statistic for the uncertainty in the margins is skipped. The implemented tests assumes that where is no uncertainty in the margins. The correction can be found in Huang and Prokhorov (2011). It involves two-dimensional integrals.
WARNING: For the t-copula the test may be instable. The results for the t-copula therefore have to be treated carefully.

method = "kendall":
This copula goodness-of-fit test is based on Kendall’s process as investigated by Genest and Rivest (1993) and Wang and Wells (2000). For rotated copulas the input arguments are transformed and the goodness-of-fit procedure for the corresponding non-rotated copula is used.

Value

For method = "white":

- p.value  Asymptotic p-value.
- statistic  The observed test statistic.

For method = "kendall"

- p.value.CvM  Bootstrapped p-value of the goodness-of-fit test using the Cramer-von Mises statistic (if b > 0).
p.value.KS  Bootstrapped p-value of the goodness-of-fit test using the Kolmogorov-Smirnov statistic (if B > 0).

statistic.CvM  The observed Cramer-von Mises test statistic.

statistic.KS  The observed Kolmogorov-Smirnov test statistic.

Author(s)
Ulf Schepsmeier, Wanling Huang, Jiying Luo, Eike Brechmann

References


http://mediatum.ub.tum.de/id/1079291.


See Also
BiCopDeriv2, BiCopDeriv, BiCopIndTest, BiCopVuongClarke

Examples
# simulate from a bivariate Clayton copula
set.seed(123)
simdata <- BiCopSim(300, 3, 2)
u1 <- simdata[,1]
u2 <- simdata[,2]

# perform White's goodness-of-fit test for the true copula
BiCopGofTest(u1, u2, family = 3)

# perform Kendall's goodness-of-fit test for the Frank copula
BiCopGofTest(u1, u2, family = 5)

# perform Kendall's goodness-of-fit test for the true copula
gof <- BiCopGofTest(u1, u2, family = 3, method = "kendall", B=50)
gof$p.value.CvM
gof$p.value.KS

# perform Kendall's goodness-of-fit test for the Frank copula
gof <- BiCopGofTest(u1, u2, family = 5, method = "kendall", B=50)
gof$p.value.CvM
gof$p.value.KS
**Description**

This function evaluates the conditional distribution function (h-function) of a given parametric bivariate copula.

**Usage**

\[ \text{BiCophfunc}(u1, u2, \text{family}, \text{par}, \text{par2} = \emptyset, \text{obj} = \text{NULL}) \]

**Arguments**

- \( u1, u2 \): Numeric vectors of equal length with values in \([0,1]\).
- \( \text{family} \): An integer defining the bivariate copula family:
  - 0 = independence copula
  - 1 = Gaussian copula
  - 2 = Student t copula (t-copula)
  - 3 = Clayton copula
  - 4 = Gumbel copula
  - 5 = Frank copula
  - 6 = Joe copula
  - 7 = BB1 copula
  - 8 = BB6 copula
  - 9 = BB7 copula
  - 10 = BB8 copula
  - 13 = rotated Clayton copula (180 degrees; “survival Clayton”)
  - 14 = rotated Gumbel copula (180 degrees; “survival Gumbel”)
  - 16 = rotated Joe copula (180 degrees; “survival Joe”)
  - 17 = rotated BB1 copula (180 degrees; “survival BB1”)
  - 18 = rotated BB6 copula (180 degrees; “survival BB6”)
  - 19 = rotated BB7 copula (180 degrees; “survival BB7”)
  - 20 = rotated BB8 copula (180 degrees; “survival BB8”)
  - 23 = rotated Clayton copula (90 degrees)
  - 24 = rotated Gumbel copula (90 degrees)
  - 26 = rotated Joe copula (90 degrees)
  - 27 = rotated BB1 copula (90 degrees)
  - 28 = rotated BB6 copula (90 degrees)
  - 29 = rotated BB7 copula (90 degrees)
  - 30 = rotated BB8 copula (90 degrees)
  - 33 = rotated Clayton copula (270 degrees)
  - 34 = rotated Gumbel copula (270 degrees)
  - 36 = rotated Joe copula (270 degrees)
  - 37 = rotated BB1 copula (270 degrees)
  - 38 = rotated BB6 copula (270 degrees)
  - 39 = rotated BB7 copula (270 degrees)
  - 40 = rotated BB8 copula (270 degrees)
  - 104 = Tawn type 1 copula
  - 114 = rotated Tawn type 1 copula (180 degrees)
  - 124 = rotated Tawn type 1 copula (90 degrees)
32

\textbf{BiCopHfunc}

\begin{itemize}
\item 134 = rotated Tawn type 1 copula (270 degrees)
\item 204 = Tawn type 2 copula
\item 214 = rotated Tawn type 2 copula (180 degrees)
\item 224 = rotated Tawn type 2 copula (90 degrees)
\item 234 = rotated Tawn type 2 copula (270 degrees)
\end{itemize}

\textbf{par}

Copula parameter.

\textbf{par2}

Second parameter for bivariate copulas with two parameters (t, BB1, BB6, BB7, BB8, Tawn type 1 and type 2; default: par2 = 0).

\textbf{obj}

BiCop object containing the family and parameter specification.

**Details**

The $h$-function is defined as the conditional distribution function of a bivariate copula, i.e.,

$$h(u|v, \theta) := F(u|v) = \frac{\partial C(u, v)}{\partial v},$$

where $C$ is a bivariate copula distribution function with parameter(s) $\theta$. For more details see Aas et al. (2009).

If the family and parameter specification is stored in a BiCop object \textit{obj}, the alternative version

\begin{verbatim}
BiCopHfunc(u1, u2, obj)
\end{verbatim}

can be used.

**Value**

\begin{itemize}
\item \texttt{hfunc1} Numeric vector of the conditional distribution function (h-function) evaluated at $u_2$ given $u_1$, i.e., $h(u_2|u_1, \theta)$.
\item \texttt{hfunc2} Numeric vector of the conditional distribution function (h-function) evaluated at $u_1$ given $u_2$, i.e., $h(u_1|u_2, \theta)$.
\end{itemize}

**Author(s)**

Ulf Schepsmeier

**References**


**See Also**

\begin{verbatim}
BiCopPDF, BiCopCDF, RVineLogLik, RVineSeqEst, BiCop
\end{verbatim}

**Examples**

\begin{verbatim}
# load data set
data(daxreturns)

# h-functions of the Gaussian copula
h1 <- BiCopHfunc(daxreturns[,2], daxreturns[,1], 1, 0.5)
\end{verbatim}
Description

This function evaluates the derivative of a given conditional parametric bivariate copula (h-function) with respect to its parameter(s) or one of its arguments.

Usage

\[
\text{BiCopHfuncDeriv}(u_1, u_2, \text{family}, \text{par}, \text{par}_2 = 0, \text{deriv} = "\text{par}"', \text{obj} = \text{NULL})
\]

Arguments

- \(u_1, u_2\): Numeric vectors of equal length with values in \([0,1]\).
- \text{family\}: An integer defining the bivariate copula family:
  - 0 = independence copula
  - 1 = Gaussian copula
  - 2 = Student t copula (t-copula)
  - 3 = Clayton copula
  - 4 = Gumbel copula
  - 5 = Frank copula
  - 6 = Joe copula
  - 13 = rotated Clayton copula (180 degrees; “survival Clayton”)
  - 14 = rotated Gumbel copula (180 degrees; “survival Gumbel”)
  - 16 = rotated Joe copula (180 degrees; “survival Joe”)
  - 23 = rotated Clayton copula (90 degrees)
  - 24 = rotated Gumbel copula (90 degrees)
  - 26 = rotated Joe copula (90 degrees)
  - 33 = rotated Clayton copula (270 degrees)
  - 34 = rotated Gumbel copula (270 degrees)
  - 36 = rotated Joe copula (270 degrees)
- \text{par\}: Copula parameter.
- \text{par}_2\): Second parameter for bivariate t-copula; default: \(\text{par}_2 = 0\).
- \text{deriv\}: Derivative argument
  - "par" = derivative with respect to the first parameter (default)
  - "par\_2" = derivative with respect to the second parameter (only available for the t-copula)
  - "u2" = derivative with respect to the second argument \(u_2\)
- \text{obj\}: BiCop object containing the family and parameter specification.

Details

If the family and parameter specification is stored in a BiCop object \text{obj}, the alternative version

\[
\text{BiCopHfuncDeriv}(u_1, u_2, \text{obj}, \text{deriv} = "\text{par}")
\]

can be used.
Value

A numeric vector of the conditional bivariate copula derivative with respect to deriv evaluated at u1 and u2 with parameter(s) par and par2.

Author(s)

Ulf Schepsmeier

References


See Also

RVineGrad, RVineHessian, BiCopDeriv2, BiCopDeriv2, BiCopHfuncDeriv, BiCop

Examples

```r
## simulate from a bivariate t-copula
simdata <- BiCopSim(300, 2, -0.7, par2 = 4)

## derivative of the conditional bivariate t-copula
## with respect to the first parameter
u1 <- simdata[,1]
u2 <- simdata[,2]
BiCopHfuncDeriv(u1, u2, -0.7, par2 = 4, deriv = "par")

## estimate a bivariate copula from the data and
## evaluate its derivative w.r.t. the parameter
cop <- BiCopEst(u1, u2, family = 2)
BiCopHfuncDeriv(u1, u2, cop, deriv = "par")
```

BiCopHfuncDeriv2

Second Derivatives of the h-Function of a Bivariate Copula

Description

This function evaluates the second derivative of a given conditional parametric bivariate copula (h-function) with respect to its parameter(s) and/or its arguments.

Usage

`BiCopHfuncDeriv2(u1, u2, family, par, par2 = 0, deriv = "par", obj = NULL)`

Arguments

- `u1, u2`: Numeric vectors of equal length with values in [0,1].
family  An integer defining the bivariate copula family:
   \( 0 \) = independence copula
   \( 1 \) = Gaussian copula
   \( 2 \) = Student t copula (t-copula)
   \( 3 \) = Clayton copula
   \( 4 \) = Gumbel copula
   \( 5 \) = Frank copula
   \( 6 \) = Joe copula
   \( 13 \) = rotated Clayton copula (180 degrees; “survival Clayton”)
   \( 14 \) = rotated Gumbel copula (180 degrees; “survival Gumbel”)
   \( 16 \) = rotated Joe copula (180 degrees; “survival Joe”)
   \( 23 \) = rotated Clayton copula (90 degrees)
   \( 24 \) = rotated Gumbel copula (90 degrees)
   \( 26 \) = rotated Joe copula (90 degrees)
   \( 33 \) = rotated Clayton copula (270 degrees)
   \( 34 \) = rotated Gumbel copula (270 degrees)
   \( 36 \) = rotated Joe copula (270 degrees)

par     Copula parameter.
par2    Second parameter for bivariate t-copula; default: \( \text{par2} = 0 \).
deriv   Derivative argument
   "\text{par}" = second derivative with respect to the first parameter (default)
   "\text{par2}" = second derivative with respect to the second parameter (only available
                  for the t-copula)
   "\text{u2}" = second derivative with respect to the second argument \( u2 \)
   "\text{par1par2}" = second derivative with respect to the first and second parameter
                     (only available for the t-copula)
   "\text{par1u2}" = second derivative with respect to the first parameter and the second
                   argument
   "\text{par2u2}" = second derivative with respect to the second parameter and the sec-
                   ond argument (only available for the t-copula)

obj     BiCop object containing the family and parameter specification.

Details

If the family and parameter specification is stored in a BiCop object \( \text{obj} \), the alternative version

\[
\text{BiCopHfuncDeriv2(u1, u2, obj, deriv = "par")}
\]

can be used.

Value

A numeric vector of the second conditional bivariate copula derivative with respect to \( \text{deriv} \) evalu-
ated at \( u1 \) and \( u2 \) with parameter(s) \( \text{par} \) and \( \text{par2} \).

Author(s)

Ulf Schepsmeier, Jakob Stoeber
BiCopIndTest

Description

This function returns the p-value of a bivariate asymptotic independence test based on Kendall’s tau.

Usage

BiCopIndTest(u1, u2)

Arguments

u1, u2 Data vectors of equal length with values in [0, 1].

Details

The test exploits the asymptotic normality of the test statistic

\[ \text{statistic} := T = \sqrt{\frac{9N(N-1)}{2(2N+5)}} \times |\hat{\tau}|, \]

where \( N \) is the number of observations (length of \( u1 \)) and \( \hat{\tau} \) the empirical Kendall’s tau of the data vectors \( u1 \) and \( u2 \). The p-value of the null hypothesis of bivariate independence hence is asymptotically

\[ p.\text{value} = 2 \times (1 - \Phi(T)), \]

where \( \Phi \) is the standard normal distribution function.

References


See Also

RVineGrad, RVineHessian, BiCopDeriv, BiCopDeriv2, BiCopHfuncDeriv, BiCop

Examples

## simulate from a bivariate t-copula
simdata <- BiCopSim(300, 2, -0.7, par2 = 4)

## second derivative of the conditional bivariate t-copula
## with respect to the first parameter
u1 <- simdata[1]
u2 <- simdata[2]
BiCopHfuncDeriv2(u1, u2, -0.7, par2 = 4, deriv = “par”)

## estimate a bivariate copula from the data and
## evaluate its 2nd order derivative w.r.t. the parameter
cop <- BiCopEst(u1, u2, family = 2)
BiCopHfuncDeriv2(u1, u2, cop, deriv = “par”)
**BiCopKPlot**

**Value**
- **statistic**: Test statistic of the independence test.
- **p.value**: P-value of the independence test.

**Author(s)**
Jeffrey Dissmann

**References**
Genest, C. and A. C. Favre (2007). Everything you always wanted to know about copula modeling but were afraid to ask. Journal of Hydrologic Engineering, 12 (4), 347-368.

**See Also**
- BiCopGofTest, BiCopPar2Tau, BiCopTau2Par, BiCopSelect,
- RVineCopSelect, RVineStructureSelect

**Examples**

```r
## Example 1: Gaussian copula with large dependence parameter
par1 <- 0.7
fam1 <- 1
dat1 <- BiCopSim(500, fam1, par1)

# perform the asymptotic independence test
BiCopIndTest(dat1[,1], dat1[,2])

## Example 2: Gaussian copula with small dependence parameter
par2 <- 0.01
fam2 <- 1
dat2 <- BiCopSim(500, fam2, par2)

# perform the asymptotic independence test
BiCopIndTest(dat2[,1], dat2[,2])
```

---

**Kendall’s Plot for Bivariante Copula Data**

**Description**
This function creates a Kendall’s plot (K-plot) of given bivariate copula data.

**Usage**

`BiCopKPlot(u1, u2, PLOT = TRUE, ...)`

**Arguments**
- **u1, u2**: Data vectors of equal length with values in [0,1].
- **PLOT**: Logical; whether the results are plotted. If PLOT = FALSE, the values in and Hi. sort are returned (see below; default: PLOT = TRUE).
- **...**: Additional plot arguments.
Details

For observations \( u_{i,j}, i = 1, \ldots, N, j = 1, 2 \), the K-plot considers two quantities: First, the ordered values of the empirical bivariate distribution function \( H_i := \hat{F}_{U_1, U_2}(u_{i,1}, u_{i,2}) \) and, second, \( W_{i:N} \), which are the expected values of the order statistics from a random sample of size \( N \) of the random variable \( W = C(U_1, U_2) \) under the null hypothesis of independence between \( U_1 \) and \( U_2 \). \( W_{i:N} \) can be calculated as follows

\[
W_{i:n} = N \binom{N - 1}{i - 1} \int_0^1 \omega k_0(\omega)(K_0(\omega))^{i-1}(1 - K_0(\omega))^{N-i} d\omega,
\]

where

\[
K_0(\omega) = \omega - \omega \log(\omega),
\]

and \( k_0(\cdot) \) is the corresponding density.

K-plots can be seen as the bivariate copula equivalent to QQ-plots. If the points of a K-plot lie approximately on the diagonal \( y = x \), then \( U_1 \) and \( U_2 \) are approximately independent. Any deviation from the diagonal line points towards dependence. In case of positive dependence, the points of the K-plot should be located above the diagonal line, and vice versa for negative dependence. The larger the deviation from the diagonal, the stronger is the degree of dependency. There is a perfect positive dependence if points \( (W_{i:N}, H_i) \) lie on the curve \( K_0(\omega) \) located above the main diagonal. If points \( (W_{i:N}, H_i) \) however lie on the x-axis, this indicates a perfect negative dependence between \( U_1 \) and \( U_2 \).

Value

\( W \).in W-statistics (x-axis).
\( H \).sort H-statistics (y-axis).

Author(s)

Natalia Belgorodski, Ulf Schepsmeier

References

Genest, C. and A. C. Favre (2007). Everything you always wanted to know about copula modeling but were afraid to ask. Journal of Hydrologic Engineering, 12 (4), 347-368.

See Also

\texttt{BiCopMetaContour}, \texttt{BiCopChiPlot}, \texttt{BiCopLambda}, \texttt{BiCopGofTest}

Examples

# Gaussian and Clayton copulas
n <- 500
tau <- 0.5

# simulate from Gaussian copula
fam1 <- 1
theta1 <- BiCopTau2Par(fam1, tau)
set.seed(123)
dat1 <- BiCopSim(n, fam1, theta1)
# simulate from Clayton copula
fam2 <- 3
theta2 <- BiCopTau2Par(fam2, tau)
set.seed(123)
dat2 <- BiCopSim(n, fam2, theta2)

# create K-plots
par(mfrow=c(1,2))
BiCopKPlot(dat1[,1], dat1[,2], main = "Gaussian copula")
BiCopKPlot(datR[,1], datR[,2], main = "Clayton copula")

---

**BiCopLambda**

**Lambda-Function (Plot) for Bivariate Copula Data**

**Description**

This function plots/returns the lambda-function of given bivariate copula data.

**Usage**

```r
BiCopLambda(u1 = NULL, u2 = NULL, family = "emp",
            par = 0, par2 = 0, PLOT = TRUE, obj = NULL, ...)
```

**Arguments**

- `u1`, `u2` Data vectors of equal length with values in $[0,1]$ (default: `u1` and `u2` = `NULL`).
- `family` An integer defining the bivariate copula family or indicating the empirical lambda-function:
  - "emp" = empirical lambda-function (default)
  - 1 = Gaussian copula; the theoretical lambda-function is simulated (no closed formula available)
  - 2 = Student t copula (t-copula); the theoretical lambda-function is simulated (no closed formula available)
  - 3 = Clayton copula
  - 4 = Gumbel copula
  - 5 = Frank copula
  - 6 = Joe copula
  - 7 = BB1 copula
  - 8 = BB6 copula
  - 9 = BB7 copula
  - 10 = BB8 copula
- `par` Copula parameter; if the empirical lambda-function is chosen, `par` = `NULL` or 0 (default).
- `par2` Second copula parameter for t-, BB1, BB6, BB7 and BB8 copulas (default: `par2` = 0).
- `PLOT` Logical; whether the results are plotted. If `PLOT` = `FALSE`, the values `emplambda` and/or `theolambda` are returned (see below; default: `PLOT` = `TRUE`).
- `obj` BiCop object containing the family and parameter specification.
Details

If the family and parameter specification is stored in a BiCop object \( \text{obj} \), the alternative version

\[
\text{BiCopLambda}(\text{obj}, \text{PLOT} = \text{TRUE}, \ldots)
\]

can be used.

Value

- emplambda: If the empirical lambda-function is chosen and \text{PLOT=FALSE}, a vector of the empirical lambda’s is returned.
- theoLambda: If the theoretical lambda-function is chosen and \text{PLOT=FALSE}, a vector of the theoretical lambda’s is returned.

Note

The \( \lambda \)-function is characteristic for each bivariate copula family and defined by Kendall’s distribution function \( K \):

\[
\lambda(v, \theta) := v - K(v, \theta)
\]

with

\[
K(v, \theta) := P(C_\theta(U_1, U_2) \leq v), \ v \in [0, 1].
\]

For Archimedean copulas one has the following closed form expression in terms of the generator function \( \varphi \) of the copula \( C_\theta \):

\[
\lambda(v, \theta) = \frac{\varphi(v)}{\varphi'(v)},
\]

where \( \varphi' \) is the derivative of \( \varphi \). For more details see Genest and Rivest (1993) or Schepsmeier (2010).

For the bivariate Gaussian and t-copula no closed form expression for the theoretical \( \lambda \)-function exists. Therefore it is simulated based on samples of size 1000. For all other implemented copula families there are closed form expressions available.

The plot of the theoretical \( \lambda \)-function also shows the limits of the \( \lambda \)-function corresponding to Kendall’s tau = 0 and Kendall’s tau = 1 (\( \lambda = 0 \)).

For rotated bivariate copulas one has to transform the input arguments \( u_1 \) and/or \( u_R \). In particular, for copulas rotated by 90 degrees \( u_1 \) has to be set to \( 1-u_1 \), for 270 degrees \( u_2 \) to \( 1-u_2 \) and for survival copulas \( u_1 \) and \( u_2 \) to \( 1-u_1 \) and \( 1-u_2 \), respectively. Then \( \lambda \)-functions for the corresponding non-rotated copula families can be considered.

Author(s)

Ulf Schepsmeier

References


See Also

BiCopMetaContour, BiCopKPlot, BiCopChiPlot, BiCop

Examples

# Clayton and rotated Clayton copulas
n <- 1000
tau <- 0.5

# simulate from Clayton copula
fam <- 3
theta <- BiCopTau2Par(fam, tau)
set.seed(123)
dat <- BiCopSim(n, fam, theta)

# create lambda-function plots
par(mfrow = c(1, 3))
BiCopLambda(dat[, 1], dat[, 2])  # empirical lambda-function
BiCopLambda(family = fam, par = theta)  # theoretical lambda-function
BiCopLambda(dat[, 1], dat[, 2], family = fam, par = theta)  # both

# lambda-function of estimated copula
fit <- BiCopSelect(dat[, 1], dat[, 2])
par(mfrow = c(1, 1))
BiCopLambda(fit)

# simulate from rotated Clayton copula (90 degrees)
theta <- BiCopTau2Par(fam, -tau)
set.seed(123)
dat <- BiCopSim(n, fam, theta)

# rotate the data to standard Clayton copula data
rot_dat <- 1 - dat[, 1]

par(mfrow = c(1, 3))
BiCopLambda(rot_dat, dat[, 2])  # empirical lambda-function
BiCopLambda(family = 3, par = -theta)  # theoretical lambda-function
BiCopLambda(rot_dat, dat[, 2], family = 3, par = -theta)  # both

Description

This function plots a bivariate contour plot corresponding to a bivariate meta distribution with different margins and specified bivariate copula and parameter values or creates corresponding empirical contour plots based on bivariate copula data.

Usage

BiCopMetaContour(u1 = NULL, u2 = NULL, bw = 1, size = 100,
levels = c(0.01, 0.05, 0.1, 0.15, 0.2),
family = "emp", par = 0, par2 = 0, PLOT = TRUE,
margins = "norm", margins.par = 0, xylim = NA, obj = NULL, ...)
**Arguments**

- **u1, u2**
  Data vectors of equal length with values in [0,1] (default: u1 and u2 = NULL).

- **bw**
  Bandwidth (smoothing factor; default: bw = 1).

- **size**
  Number of grid points; default: size = 100.

- **levels**
  Vector of contour levels. For Gaussian, Student t or exponential margins the default value (levels = c(0.01, 0.05, 0.1, 0.15, 0.2)) typically is a good choice. For uniform margins we recommend levels = c(0.1, 0.3, 0.5, 0.7, 0.9, 1.1, 1.3, 1.5) and for Gamma margins levels = c(0.005, 0.01, 0.03, 0.05, 0.07, 0.09).

- **family**
  An integer defining the bivariate copula family or indicating an empirical contour plot:
  "emp" = empirical contour plot (default; margins can be specified by margins)
  0 = independence copula
  1 = Gaussian copula
  2 = Student t copula (t-copula)
  3 = Clayton copula
  4 = Gumbel copula
  5 = Frank copula
  6 = Joe copula
  7 = BB1 copula
  8 = BB6 copula
  9 = BB7 copula
  10 = BB8 copula
  13 = rotated Clayton copula (180 degrees; “survival Clayton”)  
  14 = rotated Gumbel copula (180 degrees; “survival Gumbel”)  
  16 = rotated Joe copula (180 degrees; “survival Joe”)  
  17 = rotated BB1 copula (180 degrees; “survival BB1”)  
  18 = rotated BB6 copula (180 degrees; “survival BB6”)  
  19 = rotated BB7 copula (180 degrees; “survival BB7”)  
  20 = rotated BB8 copula (180 degrees; “survival BB8”)  
  23 = rotated Clayton copula (90 degrees)  
  24 = rotated Gumbel copula (90 degrees)  
  26 = rotated Joe copula (90 degrees)  
  27 = rotated BB1 copula (90 degrees)  
  28 = rotated BB6 copula (90 degrees)  
  29 = rotated BB7 copula (90 degrees)  
  30 = rotated BB8 copula (90 degrees)  
  33 = rotated Clayton copula (270 degrees)  
  34 = rotated Gumbel copula (270 degrees)  
  36 = rotated Joe copula (270 degrees)  
  37 = rotated BB1 copula (270 degrees)  
  38 = rotated BB6 copula (270 degrees)  
  39 = rotated BB7 copula (270 degrees)  
  40 = rotated BB8 copula (270 degrees)  
  104 = Tawn type 1 copula  
  114 = rotated Tawn type 1 copula (180 degrees)  
  124 = rotated Tawn type 1 copula (90 degrees)  
  134 = rotated Tawn type 1 copula (270 degrees)  
  204 = Tawn type 2 copula  
  214 = rotated Tawn type 2 copula (180 degrees)
224 = rotated Tawn type 2 copula (90 degrees)
234 = rotated Tawn type 2 copula (270 degrees)

par        Copula parameter; if empirical contour plot, par = NULL or \emptyset (default).
par2       Second copula parameter for t-, BB1, BB6, BB7, BB8, Tawn type 1 and type 2
copulas (default: par2 = \emptyset).

PLOT       Logical; whether the results are plotted. If PLOT = FALSE, the values x, y and z
are returned (see below; default: PLOT = TRUE).

margins    Character; margins for the bivariate copula contour plot. Possible margins are:
"norm" = standard normal margins (default)
"t" = Student t margins with degrees of freedom as specified by margins.par
"gamma" = Gamma margins with shape and scale as specified by margins.par
"exp" = Exponential margins with rate as specified by margins.par
"unif" = uniform margins

margins.par Parameter(s) of the distribution of the margins if necessary (default: margins.par = \emptyset),
i.e.,
\begin{itemize}
  \item a positive real number for the degrees of freedom of Student t margins (see
dt),
  \item a 2-dimensional vector of positive real numbers for the shape and scale
parameters of Gamma margins (see dgamma),
  \item a positive real number for the rate parameter of exponential margins (see
dexp).
\end{itemize}

xylim      A 2-dimensional vector of the x- and y-limits. By default (xylim = NA) standard
limits for the selected margins are used.

obj        BiCop object containing the family and parameter specification.

...        Additional plot arguments.

Value

x          A vector of length size with the x-values of the kernel density estimator with
Gaussian kernel if the empirical contour plot is chosen and a sequence of values in
xylim if the theoretical contour plot is chosen.

y          A vector of length size with the y-values of the kernel density estimator with
Gaussian kernel if the empirical contour plot is chosen and a sequence of values in
xylim if the theoretical contour plot is chosen.

z          A matrix of dimension size with the values of the density of the meta distribution
with chosen margins (see margins and margins.par) evaluated at the grid
points given by x and y.

Note

The combination family = \emptyset (independence copula) and margins = "unif" (uniform margins)
is not possible because all z-values are equal.

Author(s)

Ulf Schepsmeier, Alexander Bauer
BiCopName

See Also

BiCopChiPlot, BiCopKPlot, BiCopLambda

Examples

## Example 1: contour plot of meta Gaussian copula distribution
## with Gaussian margins
tau <- 0.5
fam <- 1
theta <- BiCopTau2Par(fam, tau)
BiCopMetaContour(u1 = NULL, u2 = NULL, bw = 1, size = 100,
    levels = c(0.01, 0.05, 0.1, 0.15, 0.2),
    family = fam, par = theta, main = "tau = 0.5")

## Example 2: empirical contour plot with standard normal margins
dat <- BiCopSim(N = 1000, fam, theta)
BiCopMetaContour(dat[,1], dat[,2], bw = 2, size = 100,
    levels = c(0.01, 0.05, 0.1, 0.15, 0.2),
    par = 0, family = "emp", main = "N = 1000")

# empirical contour plot with exponential margins
BiCopMetaContour(dat[,1], dat[,2], bw = 2, size = 100,
    levels = c(0.01, 0.05, 0.1, 0.15, 0.2),
    par = 0, family = "exp", main = "n = 500",
    margins = "exp", margins.par = 1)

---

BiCopName

**Bivariate Copula Family Names**

**Description**

This function transforms the bivariate copula family number into its character expression and vice versa.

**Usage**

```
BiCopName(family, short = TRUE)
```

**Arguments**

- `family` Bivariate copula family, either its number or its character expression (see table below).

**Table**

<table>
<thead>
<tr>
<th>No.</th>
<th>Short name</th>
<th>Long name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>&quot;I&quot;</td>
<td>&quot;Independence&quot;</td>
</tr>
<tr>
<td>1</td>
<td>&quot;N&quot;</td>
<td>&quot;Gaussian&quot;</td>
</tr>
<tr>
<td>2</td>
<td>&quot;t&quot;</td>
<td>&quot;t&quot;</td>
</tr>
<tr>
<td>3</td>
<td>&quot;C&quot;</td>
<td>&quot;Clayton&quot;</td>
</tr>
<tr>
<td>4</td>
<td>&quot;G&quot;</td>
<td>&quot;Gumbel&quot;</td>
</tr>
<tr>
<td>5</td>
<td>&quot;F&quot;</td>
<td>&quot;Frank&quot;</td>
</tr>
<tr>
<td>6</td>
<td>&quot;J&quot;</td>
<td>&quot;Joe&quot;</td>
</tr>
<tr>
<td>7</td>
<td>&quot;BB1&quot;</td>
<td>&quot;Clayton-Gumbel&quot;</td>
</tr>
</tbody>
</table>
short Logical; if the number of a bivariate copula family is used and short = TRUE (default), a short version of the corresponding character expression is returned, otherwise the long version.

Value

The transformed bivariate copula family (see table above).

Author(s)

Ulf Schepsmeier

See Also

RVineTreePlot

Examples

# family as number
family = 1
BiCopPar2Beta(family, short = TRUE) # short version
BiCopPar2Beta(family, short = FALSE) # long version

# family as character expression (short version)
family = "C"
BiCopName(family) # as number

# long version
family = "Clayton"
BiCopName(family) # as number

BiCopPar2Beta  Blomqvist’s Beta Value of a Bivariate Copula

Description

This function computes the theoretical Blomqvist’s beta value of a bivariate copula for given parameter values.

Usage

BiCopPar2Beta(family, par1, par2 = 0, obj = NULL)

Arguments

family   An integer defining the bivariate copula family:
0 = independence copula
1 = Gaussian copula
3 = Clayton copula
4 = Gumbel copula
5 = Frank copula
6 = Joe copula
7 = BB1 copula
8 = BB6 copula
9 = BB7 copula
10 = BB8 copula
13 = rotated Clayton copula (180 degrees; “survival Clayton”)  
14 = rotated Gumbel copula (180 degrees; “survival Gumbel”)  
16 = rotated Joe copula (180 degrees; “survival Joe”)  
17 = rotated BB1 copula (180 degrees; “survival BB1”)  
18 = rotated BB6 copula (180 degrees; “survival BB6”)  
19 = rotated BB7 copula (180 degrees; “survival BB7”)  
20 = rotated BB8 copula (180 degrees; “survival BB8”)  
23 = rotated Clayton copula (90 degrees)  
24 = rotated Gumbel copula (90 degrees)  
26 = rotated Joe copula (90 degrees)  
27 = rotated BB1 copula (90 degrees)  
28 = rotated BB6 copula (90 degrees)  
29 = rotated BB7 copula (90 degrees)  
30 = rotated BB8 copula (90 degrees)  
33 = rotated Clayton copula (270 degrees)
BiCopPar2Beta

34 = rotated Gumbel copula (270 degrees)
36 = rotated Joe copula (270 degrees)
37 = rotated BB1 copula (270 degrees)
38 = rotated BB6 copula (270 degrees)
39 = rotated BB7 copula (270 degrees)
40 = rotated BB8 copula (270 degrees)
104 = Tawn type 1 copula
114 = rotated Tawn type 1 copula (180 degrees)
124 = rotated Tawn type 1 copula (90 degrees)
134 = rotated Tawn type 1 copula (270 degrees)
204 = Tawn type 2 copula
214 = rotated Tawn type 2 copula (180 degrees)
224 = rotated Tawn type 2 copula (90 degrees)
234 = rotated Tawn type 2 copula (270 degrees)
Note that the Student’s t-copula is not allowed since the CDF of the t-copula is not implemented (see BiCopCDF).

par
Copula parameter.

par2
Second parameter for the two parameter BB1, BB6, BB7, BB8, Tawn type 1 and type 2 copulas (default: par2 = 0).

obj
BiCop object containing the family and parameter specification.

Details
If the family and parameter specification is stored in a BiCop object obj, the alternative version

BiCopPar2Beta(obj)

can be used.

Value
Theoretical value of Blomqvist’s beta corresponding to the bivariate copula family and parameter(s)

Author(s)
Ulf Schepsmeier

References

Examples
#Blomqvist's beta for the Clayton copula
BiCopPar2Beta(family = 3, par = 2)
Description

This function computes the theoretical tail dependence coefficients of a bivariate copula for given parameter values.

Usage

BiCopPar2TailDep(family, par, par2 = 0, obj = NULL)

Arguments

family  An integer defining the bivariate copula family:
0 = independence copula
1 = Gaussian copula
2 = Student t copula (t-copula)
3 = Clayton copula
4 = Gumbel copula
5 = Frank copula
6 = Joe copula
7 = BB1 copula
8 = BB6 copula
9 = BB7 copula
10 = BB8 copula
13 = rotated Clayton copula (180 degrees; “survival Clayton”)
14 = rotated Gumbel copula (180 degrees; “survival Gumbel”)
16 = rotated Joe copula (180 degrees; “survival Joe”)
17 = rotated BB1 copula (180 degrees; “survival BB1”)
18 = rotated BB6 copula (180 degrees; “survival BB6”)
19 = rotated BB7 copula (180 degrees; “survival BB7”)
20 = rotated BB8 copula (180 degrees; “survival BB8”)
23 = rotated Clayton copula (90 degrees)
24 = rotated Gumbel copula (90 degrees)
26 = rotated Joe copula (90 degrees)
27 = rotated BB1 copula (90 degrees)
28 = rotated BB6 copula (90 degrees)
29 = rotated BB7 copula (90 degrees)
30 = rotated BB8 copula (90 degrees)
33 = rotated Clayton copula (270 degrees)
34 = rotated Gumbel copula (270 degrees)
36 = rotated Joe copula (270 degrees)
37 = rotated BB1 copula (270 degrees)
38 = rotated BB6 copula (270 degrees)
39 = rotated BB7 copula (270 degrees)
40 = rotated BB8 copula (270 degrees)
104 = Tawn type 1 copula
114 = rotated Tawn type 1 copula (180 degrees)
124 = rotated Tawn type 1 copula (90 degrees)
134 = rotated Tawn type 1 copula (270 degrees)
BiCopPar2TailDep

204 = Tawn type 2 copula
214 = rotated Tawn type 2 copula (180 degrees)
224 = rotated Tawn type 2 copula (90 degrees)
234 = rotated Tawn type 2 copula (270 degrees)

\[ \text{par} \] Copula parameter.
\[ \text{par2} \] Second parameter for the two parameter t-, BB1, BB6, BB7, BB8, Tawn type 1 and type 2 copulas (default: \text{par2} = 0).

\[ \text{obj} \] BiCop object containing the family and parameter specification.

Details

If the family and parameter specification is stored in a BiCop object \text{obj}, the alternative version

\[
\text{BiCopPar2TailDep}(\text{obj})
\]
can be used.

Value

\[ \text{lower} \] Lower tail dependence coefficient of the given bivariate copula family \( C \):
\[
\lambda_L = \lim_{u \searrow 0} \frac{C(u, u)}{u}
\]
\[ \text{upper} \] Upper tail dependence coefficient of the given bivariate copula family \( C \):
\[
\lambda_U = \lim_{u \nearrow 1} \frac{1 - 2u + C(u, u)}{1 - u}
\]

Lower and upper tail dependence coefficients for bivariate copula families and parameters (\( \theta \) for one parameter families and the first parameter of the t-copula with \( \nu \) degrees of freedom, \( \theta \) and \( \delta \) for the two parameter BB1, BB6, BB7 and BB8 copulas) are given in the following table.

<table>
<thead>
<tr>
<th>No.</th>
<th>Lower tail dependence</th>
<th>Upper tail dependence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>( 2^\frac{-1}{\theta} ) ( -\sqrt{\nu + 1} \sqrt{\frac{\theta - 1}{\nu+1}} )</td>
<td>( 2^\frac{-1}{\theta} ) ( -\sqrt{\nu + 1} \sqrt{\frac{\theta - 1}{\nu+1}} )</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>( 2 - 2^\frac{1}{\theta} )</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>( 2 - 2^\frac{1}{\delta} )</td>
</tr>
<tr>
<td>7</td>
<td>( 2^\frac{-1}{(\theta \delta)} )</td>
<td>( 2 - 2^\frac{1}{\delta} )</td>
</tr>
<tr>
<td>8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>( 2^\frac{-1}{\delta} )</td>
<td>( 2 - 2^\frac{1}{\delta} )</td>
</tr>
<tr>
<td>10</td>
<td>-</td>
<td>( 2 - 2^\frac{1}{\delta} ) if ( \delta = 1 ) otherwise 0</td>
</tr>
<tr>
<td>13</td>
<td>-</td>
<td>( 2^\frac{-1}{\theta} )</td>
</tr>
<tr>
<td>14</td>
<td>( 2 - 2^\frac{1}{\theta} )</td>
<td>-</td>
</tr>
<tr>
<td>16</td>
<td>( 2 - 2^\frac{1}{\theta} )</td>
<td>-</td>
</tr>
<tr>
<td>17</td>
<td>( 2 - 2^\frac{1}{\delta} )</td>
<td>( 2 - 1/(\theta \delta) )</td>
</tr>
<tr>
<td>18</td>
<td>( 2 - 2^\frac{1}{(\theta \delta)} )</td>
<td>-</td>
</tr>
<tr>
<td>19</td>
<td>( 2 - 2^\frac{1}{\theta} )</td>
<td>( 2^\frac{-1}{\delta} )</td>
</tr>
</tbody>
</table>
BiCopPar2Tau

Description

This function computes the theoretical Kendall’s tau value of a bivariate copula for given parameter values.

Usage

BiCopPar2Tau(family, par, par2 = 0, obj = NULL)

Examples

## Example 1: Gaussian copula
BiCopParTailDep(1, 0.7)

## Example 2: t copula
BiCopParTailDep(2, 0.7, 4)

Author(s)

Eike Brechmann

References


See Also

BiCopPar2Tau
Arguments

family
An integer defining the bivariate copula family:
0 = independence copula
1 = Gaussian copula
2 = Student t copula (t-copula)
3 = Clayton copula
4 = Gumbel copula
5 = Frank copula
6 = Joe copula
7 = BB1 copula
8 = BB6 copula
9 = BB7 copula
10 = BB8 copula
13 = rotated Clayton copula (180 degrees; “survival Clayton”)
14 = rotated Gumbel copula (180 degrees; “survival Gumbel”)
16 = rotated Joe copula (180 degrees; “survival Joe”)
17 = rotated BB1 copula (180 degrees; “survival BB1”)
18 = rotated BB6 copula (180 degrees; “survival BB6”)
19 = rotated BB7 copula (180 degrees; “survival BB7”)
20 = rotated BB8 copula (180 degrees; “survival BB8”)
23 = rotated Clayton copula (90 degrees)
24 = rotated Gumbel copula (90 degrees)
26 = rotated Joe copula (90 degrees)
27 = rotated BB1 copula (90 degrees)
28 = rotated BB6 copula (90 degrees)
29 = rotated BB7 copula (90 degrees)
30 = rotated BB8 copula (90 degrees)
33 = rotated Clayton copula (270 degrees)
34 = rotated Gumbel copula (270 degrees)
36 = rotated Joe copula (270 degrees)
37 = rotated BB1 copula (270 degrees)
38 = rotated BB6 copula (270 degrees)
39 = rotated BB7 copula (270 degrees)
40 = rotated BB8 copula (270 degrees)
104 = Tawn type 1 copula
114 = rotated Tawn type 1 copula (180 degrees)
124 = rotated Tawn type 1 copula (90 degrees)
134 = rotated Tawn type 1 copula (270 degrees)
204 = Tawn type 2 copula
214 = rotated Tawn type 2 copula (180 degrees)
224 = rotated Tawn type 2 copula (90 degrees)
234 = rotated Tawn type 2 copula (270 degrees)

cop
Copula parameter (vector).
par2
Second parameter (vector of same length as par) for the two parameter t-, BB1, BB6, BB7, BB8, Tawn type 1 and type 2 copulas (default: par2 = 0). Note that the degrees of freedom parameter of the t-copula does not need to be set, because the theoretical Kendall’s tau value of the t-copula is independent of this choice.
obj
Bicop object containing the family and parameter specification.
Details

If the family and parameter specification is stored in a BiCop object obj, the alternative version

BiCopPar2Tau(obj)

can be used.

Value

Theoretical value of Kendall’s tau (vector) corresponding to the bivariate copula family and parameter(vectors) \((\theta, \delta)\) Kendall’s tau (vector) for one parameter families and the first parameter of the t-copula, \(\theta\) and \(\delta\) for the two parameter BB1, BB6, BB7, BB8, Tawn type 1 and type 2 copulas).

<table>
<thead>
<tr>
<th>No. (family)</th>
<th>Kendall’s tau (tau)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 2</td>
<td>(\frac{2}{\pi} \arcsin(\theta))</td>
</tr>
<tr>
<td>3, 13</td>
<td>(\frac{\theta}{\pi} x^2)</td>
</tr>
<tr>
<td>4, 14</td>
<td>(1 - \frac{1}{\theta})</td>
</tr>
<tr>
<td>5</td>
<td>(1 - \frac{1}{\theta} + 4 \frac{\beta(\theta)}{\theta}) with (\beta(\theta) = \int_0^\theta \frac{x/\theta}{\exp(x) - 1} dx) (Debye function)</td>
</tr>
<tr>
<td>6, 16</td>
<td>(1 + \frac{1}{\pi} \int_0^1 x \log(x)(1 - x)^{2(1+\theta)/\theta} dx)</td>
</tr>
<tr>
<td>7, 17</td>
<td>(1 - \frac{1}{\pi(\theta+2)})</td>
</tr>
<tr>
<td>8, 18</td>
<td>(1 + 4 \int_0^1 - \log(- (1-t)^\theta + 1)(1 - t - (1-t)^{-\theta} + (1-t)^{-\theta})/\delta dt)</td>
</tr>
<tr>
<td>9, 19</td>
<td>(1 + 4 \int_0^1 \frac{((1 - (1-t)^\theta)^{-\delta} - 1)/(1 - \delta)^{\theta - 1}(1 - (1-t)^{\theta - 1}) \cdot (1-t\delta - (1-t)^{-\theta} + (1-t)^{-\theta})/(\delta dt)}{\delta}(\theta) dt)</td>
</tr>
<tr>
<td>10, 20</td>
<td>(1 + 4 \int_0^1 - \log((1 - t\delta)\theta - 1)/(1 - \delta)^{\theta - 1}) \cdot (1-t\delta - (1-t)^{-\theta} + (1-t)^{-\theta})/(\delta dt))</td>
</tr>
<tr>
<td>23, 33</td>
<td>(\frac{\theta}{\pi} x^2)</td>
</tr>
<tr>
<td>24, 34</td>
<td>(-1 - \frac{1}{\theta})</td>
</tr>
<tr>
<td>26, 36</td>
<td>(-1 - \frac{1}{\pi(\theta+2)})</td>
</tr>
<tr>
<td>27, 37</td>
<td>(-1 - \frac{1}{\pi(\theta+2)})</td>
</tr>
<tr>
<td>28, 38</td>
<td>(-1 - 4 \int_0^1 - \log(- (1-t)^{-\theta} + 1)(1 - t - (1-t)^{\theta} + (1-t)^{\theta})/\delta dt)</td>
</tr>
<tr>
<td>29, 39</td>
<td>(-1 - 4 \int_0^1 \frac{1 - (1 - t)^{-\theta})\delta - 1)/(\delta \delta (1 - (1-t)^{-\theta - 1}) - (1 - (1-t)^{-\theta}) ) \cdot (1-t\delta - (1-t)^{-\theta} + (1-t)^{-\theta})/(\delta dt))</td>
</tr>
<tr>
<td>30, 40</td>
<td>(-1 - 4 \int_0^1 - \log((1 + t\delta)^{-\theta} - 1)/(1 + \delta)^{-\theta - 1}) \cdot (1-t\delta - (1-t)^{-\theta} + (1-t)^{-\theta})/(\delta dt))</td>
</tr>
<tr>
<td>104, 114</td>
<td>(\int_0^1 t(1-t)^{A(t)} dt) with (A(t) = (1 - \delta)(1 - t) + [(\delta(1 - t))^{\theta} + t^{\theta + \delta}]^{-1/\theta})</td>
</tr>
<tr>
<td>204, 214</td>
<td>(\int_0^1 ((1-t)^{A(t)}) dt) with (A(t) = (1 - \delta) (1 - t) + [(1-t)^{-\theta} + (\delta t)^{-\theta}]^{-1/\theta})</td>
</tr>
<tr>
<td>124, 134</td>
<td>(-\int_0^1 ((1-t)^{A(t)}) dt) with (A(t) = (1 - \delta) (1 - t) + [(\delta(1 - t))^{-\theta} + t^{-\theta}]^{-1/\theta})</td>
</tr>
<tr>
<td>224, 234</td>
<td>(-\int_0^1 (1-t)^{A(t)} dt) with (A(t) = (1 - \delta) (1 - t) + [(1-t)^{-\theta} + (\delta t)^{-\theta}]^{-1/\theta})</td>
</tr>
</tbody>
</table>

Author(s)

Ulf Schepsmeier, Tobias Erhardt
BiCopPDF

References


See Also

BiCopTau2Par, BiCop

Examples

```r
## Example 1: Gaussian copula
tau0 <- 0.5
rho <- BiCopTau2Par(family = 1, tau = tau0)

# transform back
tau <- BiCopPar2Tau(family = 1, par = rho)
tau <- 2/pi*asin(rho)

## Example 2: Clayton copula
theta <- BiCopTau2Par(family = 3, tau = c(0.4, 0.5, 0.6))
BiCopPar2Tau(family = 3, par = theta)

## Example 3:
vpars <- seq(from = 1.1, to = 10, length.out = 100)
tauC <- BiCopPar2Tau(family = 3, par = vpars)
tauG <- BiCopPar2Tau(family = 4, par = vpars)
tauF <- BiCopPar2Tau(family = 5, par = vpars)
tauJ <- BiCopPar2Tau(family = 6, par = vpars)
plot(tauC ~ vpars, type = "l", ylim = c(0,1))
lines(tauG ~ vpars, col = 2)
lines(tauF ~ vpars, col = 3)
lines(tauJ ~ vpars, col = 4)
```

BiCopPDF

Density of a Bivariate Copula

Description

This function evaluates the probability density function (PDF) of a given parametric bivariate copula.

Usage

BiCopPDF(u1, u2, family, par, par2 = 0, obj = NULL)
Arguments

$u_1, u_2$ Numeric vectors of equal length with values in $[0,1]$.

family An integer defining the bivariate copula family:
0 = independence copula
1 = Gaussian copula
2 = Student t copula (t-copula)
3 = Clayton copula
4 = Gumbel copula
5 = Frank copula
6 = Joe copula
7 = BB1 copula
8 = BB6 copula
9 = BB7 copula
10 = BB8 copula
13 = rotated Clayton copula (180 degrees; “survival Clayton”)
14 = rotated Gumbel copula (180 degrees; “survival Gumbel”)
16 = rotated Joe copula (180 degrees; “survival Joe”)
17 = rotated BB1 copula (180 degrees; “survival BB1”)
18 = rotated BB6 copula (180 degrees; “survival BB6”)
19 = rotated BB7 copula (180 degrees; “survival BB7”)
20 = rotated BB8 copula (180 degrees; “survival BB8”)
23 = rotated Clayton copula (90 degrees)
24 = rotated Gumbel copula (90 degrees)
26 = rotated Joe copula (90 degrees)
27 = rotated BB1 copula (90 degrees)
28 = rotated BB6 copula (90 degrees)
29 = rotated BB7 copula (90 degrees)
30 = rotated BB8 copula (90 degrees)
33 = rotated Clayton copula (270 degrees)
34 = rotated Gumbel copula (270 degrees)
36 = rotated Joe copula (270 degrees)
37 = rotated BB1 copula (270 degrees)
38 = rotated BB6 copula (270 degrees)
39 = rotated BB7 copula (270 degrees)
40 = rotated BB8 copula (270 degrees)
104 = Tawn type 1 copula
114 = rotated Tawn type 1 copula (180 degrees)
124 = rotated Tawn type 1 copula (90 degrees)
134 = rotated Tawn type 1 copula (270 degrees)
204 = Tawn type 2 copula
214 = rotated Tawn type 2 copula (180 degrees)
224 = rotated Tawn type 2 copula (90 degrees)
234 = rotated Tawn type 2 copula (270 degrees)

par Copula parameter.

par2 Second parameter for the two parameter t-, BB1, BB6, BB7, BB8, Tawn type 1 and type 2 copulas (default: par2 = 0).

obj BiCop object containing the family and parameter specification.
Details

If the family and parameter specification is stored in a BiCop object obj, the alternative version

\[
\text{BiCopPDF}(u_1, u_2, \text{obj})
\]

can be used.

Value

A numeric vector of the bivariate copula density evaluated at \(u_1\) and \(u_2\).

Author(s)

Eike Brechmann

See Also

BiCopCDF, BiCopHfunc, BiCopSim, BiCop

Examples

```r
## simulate from a bivariate t-copula
simdata <- BiCopSim(300, 2, -0.7, par2 = 4)

## evaluate the density of the bivariate t-copula
u1 <- simdata[,1]
u2 <- simdata[,2]
BiCopPDF(u1, u2, 2, -0.7, par2 = 4)

## estimate a bivariate copula from the data and evaluate its PDF
cop <- BiCopSelect(u1, u2)
round(BiCopPDF(u1, u2, cop), 8)
```
Arguments

\texttt{BiCopSelect} Arguments

\begin{itemize}
\item \texttt{u1, u2} Data vectors of equal length with values in [0,1].
\item \texttt{familyset} Vector of bivariate copula families to select from (the independence copula MUST NOT be specified in this vector, otherwise it will be selected). The vector has to include at least one bivariate copula family that allows for positive and one that allows for negative dependence. Not listed copula families might be included to better handle limit cases. If \texttt{familyset = NA} (default), selection among all possible families is performed. Coding of bivariate copula families:
\begin{itemize}
\item 1 = Gaussian copula
\item 2 = Student t copula (t-copula)
\item 3 = Clayton copula
\item 4 = Gumbel copula
\item 5 = Frank copula
\item 6 = Joe copula
\item 7 = BB1 copula
\item 8 = BB6 copula
\item 9 = BB7 copula
\item 10 = BB8 copula
\item 13 = rotated Clayton copula (180 degrees; “survival Clayton”)
\item 14 = rotated Gumbel copula (180 degrees; “survival Gumbel”)
\item 16 = rotated Joe copula (180 degrees; “survival Joe”)
\item 17 = rotated BB1 copula (180 degrees; “survival BB1”)
\item 18 = rotated BB6 copula (180 degrees; “survival BB6”)
\item 19 = rotated BB7 copula (180 degrees; “survival BB7”)
\item 20 = rotated BB8 copula (180 degrees; “survival BB8”)
\item 23 = rotated Clayton copula (90 degrees)
\item 24 = rotated Gumbel copula (90 degrees)
\item 26 = rotated Joe copula (90 degrees)
\item 27 = rotated BB1 copula (90 degrees)
\item 28 = rotated BB6 copula (90 degrees)
\item 29 = rotated BB7 copula (90 degrees)
\item 30 = rotated BB8 copula (90 degrees)
\item 33 = rotated Clayton copula (270 degrees)
\item 34 = rotated Gumbel copula (270 degrees)
\item 36 = rotated Joe copula (270 degrees)
\item 37 = rotated BB1 copula (270 degrees)
\item 38 = rotated BB6 copula (270 degrees)
\item 39 = rotated BB7 copula (270 degrees)
\item 40 = rotated BB8 copula (270 degrees)
\item 104 = Tawn type 1 copula
\item 114 = rotated Tawn type 1 copula (180 degrees)
\item 124 = rotated Tawn type 1 copula (90 degrees)
\item 134 = rotated Tawn type 1 copula (270 degrees)
\item 204 = Tawn type 2 copula
\item 214 = rotated Tawn type 2 copula (180 degrees)
\item 224 = rotated Tawn type 2 copula (90 degrees)
\item 234 = rotated Tawn type 2 copula (270 degrees)
\end{itemize}
\item \texttt{selectioncrit} Character indicating the criterion for bivariate copula selection. Possible choices: \texttt{selectioncrit = “AIC”} (default) or \texttt{"BIC"}.
\item \texttt{indeptest} Logical; whether a hypothesis test for the independence of \texttt{u1} and \texttt{u2} is per-
formed before bivariate copula selection (default: indeptest = FALSE; see BiCopIndTest). The independence copula is chosen if the null hypothesis of independence cannot be rejected.

level Numeric; significance level of the independence test (default: level = 0.05).

weights Numerical; weights for each observation (optional).

rotations If TRUE, all rotations of the families in familyset are included.

Details

Copulas can be selected according to the Akaike and Bayesian Information Criteria (AIC and BIC, respectively). First all available copulas are fitted using maximum likelihood estimation. Then the criteria are computed for all available copula families (e.g., if u1 and u2 are negatively dependent, Clayton, Gumbel, Joe, BB1, BB6, BB7 and BB8 and their survival copulas are not considered) and the family with the minimum value is chosen. For observations \( u_{i,j}, \ i = 1, \ldots, N, \ j = 1, 2 \), the AIC of a bivariate copula family \( c \) with parameter(s) \( \theta \) is defined as

\[
AIC := -2 \sum_{i=1}^{N} \ln[c(u_{i,1}, u_{i,2} | \theta)] + 2k,
\]

where \( k = 1 \) for one parameter copulas and \( k = 2 \) for the two parameter t-, BB1, BB6, BB7 and BB8 copulas. Similarly, the BIC is given by

\[
BIC := -2 \sum_{i=1}^{N} \ln[c(u_{i,1}, u_{i,2} | \theta)] + \ln(N)k.
\]

Evidently, if the BIC is chosen, the penalty for two parameter families is stronger than when using the AIC.

Additionally a test for independence can be performed beforehand.

Value

An object of class BiCop, i.e., a list containing

- family The selected bivariate copula family.
- par, par2 The estimated bivariate copula parameter(s).
- p.value.indet Ineptest P-value of the independence test if performed.

Note

When the bivariate t-copula is considered and the degrees of freedom are estimated to be larger than 30, then the bivariate Gaussian copula is taken into account instead. Similarly, when BB1 (Clayton-Gumbel), BB6 (Joe-Gumbel), BB7 (Joe-Clayton) or BB8 (Joe-Frank) copulas are considered and the parameters are estimated to be very close to one of their boundary cases, the respective one parameter copula is taken into account instead.

Author(s)

Eike Brechmann, Jeffrey Dissmann
References


http://mediatum.ub.tum.de/?id=1079285.


See Also

RVineStructureSelect, RVineCopSelect, BiCopIndTest, BiCop

Examples

```r
## Example 1: Gaussian copula with large dependence parameter
par1 <- 0.7
fam1 <- 1
dat1 <- BiCopSim(500, fam1, par1)

# select the bivariate copula family and estimate the parameter(s)
cop1 <- BiCopSelect(dat1[,1], dat1[,2], familyset = c(1:10),
                    indeptest = FALSE, level = 0.05)
cop1family
cop1par
cop1parR

## Example 2: Gaussian copula with small dependence parameter
par2 <- 0.01
fam2 <- 1
dat2 <- BiCopSim(500, fam2, par2)

# select the bivariate copula family and estimate the parameter(s)
cop2 <- BiCopSelect(dat2[,1], dat2[,2], familyset = c(1:10),
                    indeptest = TRUE, level = 0.05)
cop2family
cop2par
cop2parR

## Example 3: empirical data
data(daxreturns)
cop3 <- BiCopSelect(daxreturns[,1], daxreturns[,4],
                   familyset = c(1:10, 13, 14, 16,
                                 23, 24, 26, 33, 34, 36))
cop3family
cop3par
cop3parR
```
BiCopSim

Simulation from a Bivariate Copula

Description

This function simulates from a given parametric bivariate copula.

Usage

BiCopSim(N, family, par, par2 = 0, obj = NULL)

Arguments

N Number of bivariate observations simulated.
family An integer defining the bivariate copula family:
0 = independence copula
1 = Gaussian copula
2 = Student t copula (t-copula)
3 = Clayton copula
4 = Gumbel copula
5 = Frank copula
6 = Joe copula
7 = BB1 copula
8 = BB6 copula
9 = BB7 copula
10 = BB8 copula
13 = rotated Clayton copula (180 degrees; “survival Clayton”)
14 = rotated Gumbel copula (180 degrees; “survival Gumbel”)
16 = rotated Joe copula (180 degrees; “survival Joe”)
17 = rotated BB1 copula (180 degrees; “survival BB1”)
18 = rotated BB6 copula (180 degrees; “survival BB6”)
19 = rotated BB7 copula (180 degrees; “survival BB7”)
20 = rotated BB8 copula (180 degrees; “survival BB8”)
23 = rotated Clayton copula (90 degrees)
24 = rotated Gumbel copula (90 degrees)
26 = rotated Joe copula (90 degrees)
27 = rotated BB1 copula (90 degrees)
28 = rotated BB6 copula (90 degrees)
29 = rotated BB7 copula (90 degrees)
30 = rotated BB8 copula (90 degrees)
33 = rotated Clayton copula (270 degrees)
34 = rotated Gumbel copula (270 degrees)
36 = rotated Joe copula (270 degrees)
37 = rotated BB1 copula (270 degrees)
38 = rotated BB6 copula (270 degrees)
39 = rotated BB7 copula (270 degrees)
40 = rotated BB8 copula (270 degrees)
104 = Tawn type 1 copula
114 = rotated Tawn type 1 copula (180 degrees)
124 = rotated Tawn type 1 copula (90 degrees)
134 = rotated Tawn type 1 copula (270 degrees)
BiCopTau2Par

204 = Tawn type 2 copula  
214 = rotated Tawn type 2 copula (180 degrees)  
224 = rotated Tawn type 2 copula (90 degrees)  
234 = rotated Tawn type 2 copula (270 degrees)

par = Copula parameter.

par2 = Second parameter for the two parameter BB1, BB6, BB7, BB8, Tawn type 1 and type 2 copulas (default: par2 = 0).

obj = BiCop object containing the family and parameter specification.

Details

If the family and parameter specification is stored in a BiCop object obj, the alternative version

BiCopSim(N, obj)

can be used.

Value

An N x 2 matrix of data simulated from the bivariate copula.

Author(s)

Ulf Schepsmeier

See Also

BiCopCDF, BiCopPDF, RVineSim

Examples

# simulate from a bivariate t-copula
simdata <- BiCopSim(300, 2, -0.7, par2 = 4)

BiCopTau2Par

Parameter of a Bivariate Copula for a given Kendall’s Tau Value

Description

This function computes the parameter of a (one parameter) bivariate copula for a given value of Kendall’s tau.

Usage

BiCopTau2Par(family, tau)
Arguments

family

An integer defining the bivariate copula family:

0 = independence copula
1 = Gaussian copula
2 = Student t copula (Here only the first parameter can be computed)
3 = Clayton copula
4 = Gumbel copula
5 = Frank copula
6 = Joe copula
13 = rotated Clayton copula (180 degrees; “survival Clayton”)
14 = rotated Gumbel copula (180 degrees; “survival Gumbel”)
16 = rotated Joe copula (180 degrees; “survival Joe”)
23 = rotated Clayton copula (90 degrees)
24 = rotated Gumbel copula (90 degrees)
26 = rotated Joe copula (90 degrees)
33 = rotated Clayton copula (270 degrees)
34 = rotated Gumbel copula (270 degrees)
36 = rotated Joe copula (270 degrees)

Note that (with exception of the t-copula) two parameter bivariate copula families cannot be used.

tau

Kendall’s tau value (vector with elements in [-1,1]).

Value

Parameter (vector) corresponding to the bivariate copula family and the value(s) of Kendall’s tau ($\tau$).

<table>
<thead>
<tr>
<th>No. (family)</th>
<th>Parameter (par)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 2</td>
<td>$\sin(\frac{\tau \pi}{2})$</td>
</tr>
<tr>
<td>3, 13</td>
<td>$2 \frac{\tau}{1-\tau}$</td>
</tr>
<tr>
<td>4, 14</td>
<td>$\frac{1}{1-\tau}$</td>
</tr>
<tr>
<td>5</td>
<td>no closed form expression (numerical inversion)</td>
</tr>
<tr>
<td>6, 16</td>
<td>no closed form expression (numerical inversion)</td>
</tr>
<tr>
<td>23, 33</td>
<td>$2 \frac{\tau}{\tau+1}$</td>
</tr>
<tr>
<td>24, 34</td>
<td>$-\frac{1}{\tau+1}$</td>
</tr>
<tr>
<td>26, 36</td>
<td>no closed form expression (numerical inversion)</td>
</tr>
</tbody>
</table>

Author(s)

Jakob Stoeber, Eike Brechmann, Tobias Erhardt

References


See Also

BiCopPar2Tau
Examples

```r
## Example 1: Gaussian copula
tau0 <- 0.5
rho <- BiCopTau2Par(family = 1, tau = tau0)

# transform back
tau <- BiCopPar2Tau(family = 1, par = rho)
tau - 2/pi*asin(rho)

## Example 2: Clayton copula
theta <- BiCopTau2Par(family = 3, tau = c(0.4, 0.5, 0.6))
BiCopPar2Tau(family = 3, par = theta)

## Example 3:
vtau <- seq(from = 0.1, to = 0.8, length.out = 100)
thetaC <- BiCopTau2Par(family = 3, tau = vtau)
thetaG <- BiCopTau2Par(family = 4, tau = vtau)
thetaF <- BiCopTau2Par(family = 5, tau = vtau)
thetaJ <- BiCopTau2Par(family = 6, tau = vtau)
plot(thetaC ~ vtau, type = "l", ylim = range(thetaF))
lines(thetaG ~ vtau, col = 2)
lines(thetaF ~ vtau, col = 3)
lines(thetaJ ~ vtau, col = 4)
```

### Description

Based on the Vuong and Clarke tests this function computes a goodness-of-fit score for each bivariate copula family under consideration. For each possible pair of copula families the Vuong and the Clarke tests decides which of the two families fits the given data best and assigns a score—pro or contra a copula family—according to this decision.

### Usage

```r
BiCopVuongClarke(u1, u2, familyset = NA,
  correction = FALSE, level = 0.05)
```

### Arguments

- `u1, u2` Data vectors of equal length with values in \([0,1]\).
- `familyset` An integer vector of bivariate copula families under consideration, i.e., which are compared in the goodness-of-fit test. If `familyset = NA` (default), all possible families are compared. Possible families are:
  - `0` = independence copula
  - `1` = Gaussian copula
The Vuong as well as the Clarke test compare two models against each other and based on their
null hypothesis, allow for a statistically significant decision among the two models (see the doc-
umentations of `RvineVuongTest` and `RvineClarkeTest` for descriptions of the two tests). In the
goodness-of-fit test proposed by Belgorodski (2010) this is used for bivariate copula selection. It
compares a model 0 to all other possible models under consideration. If model 0 is favored over
another model, a score of "+1" is assigned and similarly a score of "-1" if the other model is
determined to be superior. No score is assigned, if the respective test cannot discriminate between
two models. Both tests can be corrected for the numbers of parameters used in the copulas. Either no correction (correction = FALSE), the Akaike correction (correction = “Akaike”) or the parsimonious Schwarz correction (correction = “Schwarz”) can be used.

The models compared here are bivariate parametric copulas and we would like to determine which family fits the data better than the other families. E.g., if we would like to test the hypothesis that the bivariate Gaussian copula fits the data best, then we compare the Gaussian copula against all other copulas under consideration. In doing so, we investigate the null hypothesis "The Gaussian copula fits the data better than all other copulas under consideration", which corresponds to \( k - 1 \) times the hypothesis "The Gaussian copula \( C_j \) fits the data better than copula \( C_i \)" for all \( i = 1, \ldots, k, i \neq j \), where \( k \) is the number of bivariate copula families under consideration (length of familyset). This procedure is done not only for one family but for all families under consideration, i.e., two scores, one based on the Vuong and one based on the Clarke test, are returned for each bivariate copula family. If used as a goodness-of-fit procedure, the family with the highest score should be selected.

For more and detailed information about the goodness-of-fit test see Belgorodski (2010).

**Value**

A matrix with Vuong test scores in the first and Clarke test scores in the second row. Column names correspond to bivariate copula families (see above).

**Author(s)**

Ulf Schepsmeier, Eike Brechmann, Natalia Belgorodski

**References**


**See Also**

`BiCopGofTest`, `RVineVuongTest`, `RVineClarkeTest`, `BiCopSelect`

**Examples**

```r
# simulate from a t-copula
set.seed(123)
dat <- BiCopSim(500, 2, 0.7, 5)

# apply the test for families 1-10
vcgof <- BiCopVuongClarke(dat[,1], dat[,2], familyset = 1:6)

# display the Vuong test scores
vcgof[1,]

# display the Clarke test scores
vcgof[2,]
```
Transform C-Vine to R-Vine Structure

Description

This function transforms a C-vine structure from the package CDVine to the corresponding R-vine structure.

Usage

C2RVine(order, family, par, par2 = rep(0, length(family)))

Arguments

order A d-dimensional vector specifying the order of the root nodes in the C-vine.
family A d*(d-1)/2 vector of pair-copula families with values
(theta) = independence copula
1 = Gaussian copula
2 = Student t copula (t-copula)
3 = Clayton copula
4 = Gumbel copula
5 = Frank copula
6 = Joe copula
7 = BB1 copula
8 = BB6 copula
9 = BB7 copula
10 = BB8 copula
13 = rotated Clayton copula (180 degrees; “survival Clayton”)
14 = rotated Gumbel copula (180 degrees; “survival Gumbel”)
16 = rotated Joe copula (180 degrees; “survival Joe”)
17 = rotated BB1 copula (180 degrees; “survival BB1”)
18 = rotated BB6 copula (180 degrees; “survival BB6”)
19 = rotated BB7 copula (180 degrees; “survival BB7”)
20 = rotated BB8 copula (180 degrees; “survival BB8”)
23 = rotated Clayton copula (90 degrees)
24 = rotated Gumbel copula (90 degrees)
26 = rotated Joe copula (90 degrees)
27 = rotated BB1 copula (90 degrees)
28 = rotated BB6 copula (90 degrees)
29 = rotated BB7 copula (90 degrees)
30 = rotated BB8 copula (90 degrees)
33 = rotated Clayton copula (270 degrees)
34 = rotated Gumbel copula (270 degrees)
36 = rotated Joe copula (270 degrees)
37 = rotated BB1 copula (270 degrees)
38 = rotated BB6 copula (270 degrees)
39 = rotated BB7 copula (270 degrees)
40 = rotated BB8 copula (270 degrees)
104 = Tawn type 1 copula
114 = rotated Tawn type 1 copula (180 degrees)
124 = rotated Tawn type 1 copula (90 degrees)
134 = rotated Tawn type 1 copula (270 degrees)
204 = Tawn type 2 copula
214 = rotated Tawn type 2 copula (180 degrees)
224 = rotated Tawn type 2 copula (90 degrees)
234 = rotated Tawn type 2 copula (270 degrees)

par = A d*(d-1)/2 vector of pair-copula parameters.
par2 = A d*(d-1)/2 vector of second pair-copula parameters (optional; default: par2 = rep(0, length(family))), necessary for the t-, BB1, BB6, BB7, BB8, Tawn type 1 and type 2 copulas.

Value

An RVineMatrix object.

Author(s)

Ulf Schepsmeier, Eike Brechmann

See Also

RVineMatrix,D2RVine

Examples

# simulate a sample of size 500 from a 4-dimensional C-vine
# copula model with mixed pair-copulas
# load package CDVine
library(CDVine)
d <- 4
dd <- d*(d-1)/2
order <- 1:d
family <- c(1, 2, 3, 4, 7, 3)
par <- c(0.5, 0.4, 2, 1.5, 1.2, 1.5)
par2 <- c(0, 5, 0, 0, 2, 0)
type <- 1
simdata <- CDVineSim(500, family, par, par2, type)

# determine log-likelihood
out <- CDVineLogLik(simdata, family, par, par2, type)
out$loglik

# transform to R-vine matrix notation
RVM <- C2RVine(order, family, par, par2)

# check that log-likelihood stays the same
out2 <- RVineLogLik(simdata, RVM)
out2$loglik
**copulaFromFamilyIndex**  
*Construction of a Copula Object from a VineCopula Family Index*

**Description**
A VineCopula family index along with its parameters is used to construct a corresponding copula object.

**Usage**
copulaFromFamilyIndex(family, par, par2 = 0)

**Arguments**
- family: a family index as defined in `VineCopula-package`
- par: its first parameter value
- par2: if present, its second parameter

**Value**
An object inheriting `copula` corresponding to the specific family.

**Author(s)**
Benedikt Graeler

**Examples**

```r
# normalCopula with parameter 0.5
copulaFromFamilyIndex(1, 0.5)

# rotated Tawn T2 copula with parameters
copulaFromFamilyIndex(224, -2, 0.5)
```

---

**D2RVine**  
*Transform D-Vine to R-Vine Structure*

**Description**
This function transforms a D-vine structure from the package CDVine to the corresponding R-vine structure.

**Usage**
D2RVine(order, family, par, par2 = rep(0, length(family)))
**copulaFromFamilyIndex**  
*Construction of a Copula Object from a VineCopula Family Index*

**Description**
A VineCopula family index along with its parameters is used to construct a corresponding copula object.

**Usage**
copulaFromFamilyIndex(family, par, par2 = 0)

**Arguments**
- family: a family index as defined in `VineCopula-package`
- par: its first parameter value
- par2: if present, its second parameter

**Value**
An object inheriting `copula` corresponding to the specific family.

**Author(s)**
Benedikt Graeler

**Examples**

```r
# normalCopula with parameter 0.5
copulaFromFamilyIndex(1, 0.5)

# rotated Tawn T2 copula with parameters
copulaFromFamilyIndex(224, -2, 0.5)
```

---

**D2RVine**  
*Transform D-Vine to R-Vine Structure*

**Description**
This function transforms a D-vine structure from the package CDVine to the corresponding R-vine structure.

**Usage**
D2RVine(order, family, par, par2 = rep(0, length(family)))
Arguments

order          A d-dimensional vector specifying the order of the nodes in the D-vine.
family        A d*(d-1)/2 vector of pair-copula families with values
               0 = independence copula
               1 = Gaussian copula
               2 = Student t copula (t-copula)
               3 = Clayton copula
               4 = Gumbel copula
               5 = Frank copula
               6 = Joe copula
               7 = BB1 copula
               8 = BB6 copula
               9 = BB7 copula
              10 = BB8 copula
              13 = rotated Clayton copula (180 degrees; “survival Clayton”)
              14 = rotated Gumbel copula (180 degrees; “survival Gumbel”)
              16 = rotated Joe copula (180 degrees; “survival Joe”)
              17 = rotated BB1 copula (180 degrees; “survival BB1”)
              18 = rotated BB6 copula (180 degrees; “survival BB6”)
              19 = rotated BB7 copula (180 degrees; “survival BB7”)
              20 = rotated BB8 copula (180 degrees; “survival BB8”)
              23 = rotated Clayton copula (90 degrees)
              24 = rotated Gumbel copula (90 degrees)
              26 = rotated Joe copula (90 degrees)
              27 = rotated BB1 copula (90 degrees)
              28 = rotated BB6 copula (90 degrees)
              29 = rotated BB7 copula (90 degrees)
              30 = rotated BB8 copula (90 degrees)
              33 = rotated Clayton copula (270 degrees)
              34 = rotated Gumbel copula (270 degrees)
              36 = rotated Joe copula (270 degrees)
              37 = rotated BB1 copula (270 degrees)
              38 = rotated BB6 copula (270 degrees)
              39 = rotated BB7 copula (270 degrees)
              40 = rotated BB8 copula (270 degrees)
              104 = Tawn type 1 copula
             114 = rotated Tawn type 1 copula (180 degrees)
             124 = rotated Tawn type 1 copula (90 degrees)
             134 = rotated Tawn type 1 copula (270 degrees)
             204 = Tawn type 2 copula
             214 = rotated Tawn type 2 copula (180 degrees)
             224 = rotated Tawn type 2 copula (90 degrees)
             234 = rotated Tawn type 2 copula (270 degrees)

par            A d*(d-1)/2 vector of pair-copula parameters.
par2           A d*(d-1)/2 vector of second pair-copula parameters (optional; default:
               par2 = rep(0, length(family))), necessary for the t-, BB1, BB6, BB7, BB8,
               Tawn type 1 and type 2 copulas.

Value

An RVineMatrix object.
Author(s)
Ulf Schepsmeier

See Also
RVineMatrix, C2R Vine

Examples

```r
# simulate a sample of size 500 from a 4-dimensional D-vine
# copula model with mixed pair-copulas
library(CDVine)
d <- 4
dd <- d*(d-1)/2
order <- 1:d
family <- c(1, 2, 3, 4, 7, 3)
par <- c(0.5, 0.4, 2, 1.5, 1.2, 1.5)
par2 <- c(0, 5, 0, 2, 0)
type <- 2
simdata <- CDVineSim(500, family, par, par2, type)

# determine log-likelihood
out <- CDVineLogLik(simdata, family, par, par2, type)
out$loglik

# transform to R-vine matrix notation
RVM <- D2RVine(order, family, par, par2)

# check that log-likelihood stays the same
out2 <- RVineLogLik(simdata, RVM)
out2$loglik
```

Description

This data set contains transformed standardized residuals of daily log returns of 15 major German stocks represented in the index DAX observed from January 2005 to August 2009. Each time series is filtered using a GARCH(1,1) model with Student t innovations.

Format

A data frame with 1158 observations on 15 variables. Column names correspond to ticker symbols of the stocks.

Source

Yahoo! Finance

See Also

RVineStructureSelect
Examples

# load the data set
data(daxreturns)

# compute the empirical Kendall's tau matrix
TauMatrix(daxreturns)

---

dduCopula  Partial Derivatives of Copulas

Description

Similar to dCopula and pCopula the function dduCopula evaluates the partial derivative \( \frac{\partial}{\partial u} C(u, v) \) and the function ddvCopula evaluates the partial derivative \( \frac{\partial}{\partial v} C(u, v) \) of the provided copula.

Usage

dduCopula(u, copula, ...)
ddvCopula(u, copula, ...)

Arguments

u  Pairs of values for which the partial derivative should be evaluated.
copula  The copula object representing the family member of interest.
...  additional arguments can be passed on to the underlying functions.

Value

A vector of the evaluated partial derivatives of the same length as rows in u.

Author(s)

Benedikt Graeler

Examples

library(copula)

BB1Cop <- BB1Copula()
BB1CopSmpl <- rCopula(100, BB1Cop)

# conditional probabilities of a Gaussian copula given u
BB1GivenU <- dduCopula(BB1CopSmpl, BB1Cop)

# vs. conditional probabilities of a Gaussian copula given v
BB1GivenV <- ddvCopula(BB1CopSmpl[,c(2,1)], BB1Cop)

plot(BB1GivenU, BB1GivenV)
abline(0,1)
**joeBiCopula**

Constructor of the Joe Family and Rotated Versions thereof

**Description**

Constructs an object of the (survival `surjoeBiCopula`, 90 degree rotated `r90joeBiCopula` and 270 degree rotated `r270joeBiCopula`) family for a given parameter. Note that package `copula-package` provides a class `joeCopula` as well.

**Usage**

```r
surjoeBiCopula(param)
```

```r
r90joeBiCopula(param)
```

```r
r270joeBiCopula(param)
```

**Arguments**

- `param`: The parameter `param` defines the copula through `theta` and `delta`.

**Value**

One of the respective Joe copula classes (`joeBiCopula`, `surjoeBiCopula`, `r90joeBiCopula`, `r270joeBiCopula`).

**Author(s)**

Benedikt Graeler

**References**


**See Also**

See also `BB1Copula`, `BB6Copula`, `BB7Copula` and `BB8Copula` for further wrapper functions to the `VineCopula-package`.

**Examples**

```r
library(copula)
persp(surjoeBiCopula(1.5), dCopula, zlim = c(0,10))
persp(r90joeBiCopula(-1.5), dCopula, zlim = c(0,10))
persp(r270joeBiCopula(-1.5), dCopula, zlim = c(0,10))
```
Description

Wrapper classes representing the bivariate Joe, survival Joe, 90 degree and 270 degree rotated Joe copula families (Joe 1997) from VineCopula-package. Note that package copula-package provides a class JoeCopula as well.

Objects from the Classes

Objects can be created by calls of the form new("joeBiCopula", ...), new("surjoeBiCopula", ...), new("r90joeBiCopula", ...) and new("r270joeBiCopula", ...) or by the functions joeBiCopula, survjoeBiCopula, r90joeBiCopula and r270joeBiCopula.

Slots

family: Object of class "numeric" defining the family number in VineCopula-package
dimension: Object of class "integer" defining the dimension of the copula
parameters: Object of class "numeric" the single parameter
param.names: Object of class "character", parameter name.
param.lowbnd: Object of class "numeric", lower bound of the copula parameter
param.upbnd: Object of class "numeric", upper bound of the copula parameter
fullname: Object of class "character", family name of the copula.

Extends

Class "copula", directly. Class "Copula", by class "copula", distance 2.

Methods

dduCopula signature(u = "matrix", copula = "joeBiCopula"): ...

ddvCopula signature(u = "numeric", copula = "joeBiCopula"): ...

kendallDistribution signature(copula = "joeBiCopula"): ...

Author(s)

Benedikt Graeler

References

pairs.copuladata

See Also

See also BB1Copula, BB6Copula, BB7Copula and BB8Copula for further wrapper classes to the VineCopula-package.

Examples

showClass("surJoeBiCopula")

pairs.copuladata  Pairs Plot of Copula Data

Description

This function provides pair plots for copula data. Using default setting it plots bivariate contour plots on the lower panel, scatter plots and correlations on the upper panel and histograms on the diagonal panel.

Usage

## S3 method for class 'copuladata'
pairs(x, labels = names(x), ..., lower.panel = lp.copuladata,
      upper.panel = up.copuladata,
      diag.panel = dp.copuladata,
      label.pos = 0.85, cex.labels = 1, gap = 0,
      method = "kendall", ccols = terrain.colors(30),
      margins = "norm", margins.par = 0)

Arguments

x  copuladata object.
labels  variable names/labels.
...  other graphical parameters (see par).
lower.panel  panel function to be used on the lower diagonal panels (if not supplied, a default function is used)
upper.panel  panel function to be used on the upper diagonal panels (if not supplied, a default function is used)
diag.panel  panel function to be used on the diagonal panels (if not supplied, a default function is used)
label.pos  y position of labels in the diagonal panel; default: label.pos = 0.85.
cex.labels  magnification to be used for the labels of the diagonal panel; default: cex.labels = 1.
gap  distance between subplots, in margin lines; default: gap = 0. If the default panel function upper.panel is used, the following parameters can be set additionally:
method  a character string indicating which correlation coefficients are computed. One of "pearson", "kendall" (default), or "spearman". If the default panel function lower.panel (see BiCopMetaContour) is used, the following parameters can be set additionally:
ccols  colour to be used for the contour plots; default: ccols = terrain.colors(30).
margins character; margins for the contour plots. Possible margins are:
"norm" = standard normal margins (default)
"t" = Student t margins with degrees of freedom as specified by margins.par
"gamma" = Gamma margins with shape and scale as specified by margins.par
"exp" = Exponential margins with rate as specified by margins.par
"unif" = uniform margins

margins.par parameter(s) of the distribution of the margins (of the contour plots) if necessary
(default: margins.par = 0), i.e.,
• a positive real number for the degrees of freedom of Student t margins (see dt),
• a 2-dimensional vector of positive real numbers for the shape and scale
  parameters of Gamma margins (see dgamma),
• a positive real number for the rate parameter of exponential margins (see
dexp).

Note

If the default panel functions are used

• col changes only the colour of the points in the scatter plot (upperNpanel)
• cex changes only the magnification of the points in the scatter plot (upperNpanel)

Author(s)

Tobias Erhardt

See Also

pairs, as.copuladata, BiCopMetaContour

Examples

data(daxreturns)

data <- as.copuladata(daxreturns)
 sel <- c(4,5,14,15)

## pairs plot with default settings
 pairs(data[sel])

## pairs plot with custom settings
 nlevels <- 20
 pairs(data[sel], cex = 2, pch = 1, col = "black",
       diag.panel = NULL, label.pos = 0.5,
       cex.labels = 2.5, gap = 1,
       method = "pearson", ccols = heat.colors(nlevels),
       margins = "gamma", margins.par = c(1,1))

## pairs plot with own panel functions
 up <- function(x, y) {
# upper panel: empirical contour plot
 op <- par(usr = c(-3, 3, -3, 3), new = TRUE)

...
plot.BiCop

BiCopMetaContour(x, y, bw = 2, levels = c(0.01, 0.05, 0.1, 0.15, 0.2),
  # exponential margins
  margins = "exp", margins.par = 1,
  axes = FALSE)
  on.exit(par(op))
)

lp <- function(x, y) {
  # lower panel: scatter plot (copula data) and correlation
  op <- par(usr = c(0, 1, 0, 1), new = TRUE)
  points(x, y, pch = 1, col = "black")
  r <- cor(x, y, method = "spearman")  # Spearman's rho
  txt <- format(x = r, digits = 3, nsmall = 3)[1]
  text(x = 0.5, y = 0.5, labels = txt, cex = 1 + abs(r) * 2, col = "blue")
  on.exit(par(op))
}

dp <- function(x) {
  # diagonal panel: histograms (copula data)
  op <- par(usr = c(0, 1, 0, 1.5), new = TRUE)
  hist(x, freq = FALSE, add = TRUE, col = "brown", border = "black", main = "")
  abline(h = 1, col = "black", lty = 2)
  on.exit(par(op))
}

nlevels <- 20
pairs(data[sel],
  lower.panel = lp, upper.panel = up, diag.panel = dp, gap = 0.5)

plot.BiCop

Plotting tools for BiCop objects

Description

There are several options for plotting BiCop objects. The density of a bivariate copula density can
be visualized as contour or surface/perspective plot. Optionally, the density can be coupled with
standard normal margins (default for contour plots). Furthermore, a lambda-plot is available (c.f.
BiCopLambda).

Usage

## S3 method for class 'BiCop'
plot(x, type = "contour", margins, size, ...)

Arguments

x BiCop object.

type plot type; either "contour", "surface" or "lambda" (partial matching is activated); the latter is only implemented for a few families (c.f. BiCopLambda).

margins only relevant for types "contour" and "surface"; either "unif" for the original copula density or "norm" for the transformed density with standard normal margins (partial matching is activated). Default is "norm" for type = "contour", and "unif" for type = "surface".
size integer; only relevant for types "contour" and "surface"; the plot is based on values on a size x size grid; default is 100 for type = "contour", and 25 for type = "surface".

... optional arguments passed to `contour` or `wireframe`.

Author(s)

Thomas Nagler

See Also

`BiCop`, `contour`, `wireframe`

Examples

```r
# construct BiCop object for a tawn copula
obj <- BiCop(family = 104, par = 2.5, par2 = 0.4)

# plots
plot(obj)  # (marginal normal) contour plot
plot(obj, margins = "unif")  # contour plot of actual copula density
plot(obj, type = "surf")  # surface plot of actual copula density
```

pobs **Pseudo-Observations**

Description

Compute the pseudo-observations for the given data matrix.

Usage

```r
pobs(x, na.last = "keep",
     ties.method = c("average", "first", "random", "max", "min"),
     lower.tail = TRUE)
```

Arguments

- **x** `n x d`-matrix of random variates to be converted to pseudo-observations.
- **na.last** passed to `rank`; see there.
- **ties.method** logical which, if FALSE, returns the pseudo-observations when applying the empirical marginal survival functions.
- **lower.tail** logical which, if FALSE, returns the pseudo-observations when applying the empirical marginal survival functions.

Details

Given `n` realizations \( x_i = (x_{i1}, \ldots, x_{id})^T \), \( i \in \{1, \ldots, n\} \) of a random vector `X`, the pseudo-observations are defined via \( u_{ij} = r_{ij}/(n + 1) \) for \( i \in \{1, \ldots, n\} \) and \( j \in \{1, \ldots, d\} \), where \( r_{ij} \) denotes the rank of \( x_{ij} \) among all \( x_{kj}, k \in \{1, \ldots, n\} \). The pseudo-observations can thus also be computed by component-wise applying the empirical distribution functions to the data and scaling the result by \( n/(n + 1) \). This asymptotically negligible scaling factor is used to force the variates to fall inside the open unit hypercube, for example, to avoid problems with density evaluation at the boundaries. Note that `pobs(, lower.tail=FALSE)` simply returns `1-pobs()`. 
Value

matrix of the same dimensions as x containing the pseudo-observations.

Note

This function is borrowed from the copula-package, see pobs.

Author(s)

Marius Hofert

Examples

```r
## Simple definition of the function:
pobs

## simulate data from a multivariate normal distribution
library(mvtnorm)
set.seed(123)
Sigma <- matrix(c(2, 1, -0.2, 1, 1, 0.3, -0.2, 0.3, 0.5), 3, 3)
mu <- c(-3, 2, 1)
dat <- rmvnorm(500, sigma = Sigma)
pairs(dat) # plot observations

## compute pseudo-observations for copula inference
udat <- pobs(dat)
pairs(udat)
# estimate vine copula model
fit <- RVineStructureSelect(udat, familyset = c(1, 2))
```

RVineAIC/BIC  

**AIC and BIC of an R-Vine Copula Model**

Description

These functions calculate the Akaike and Bayesian Information criteria of a d-dimensional R-vine copula model for a given copula data set.

Usage

```r
RVineAIC(data, RVM, par = RVM$par, par2 = RVM$par2)
RVineBIC(data, RVM, par = RVM$par, par2 = RVM$par2)
```

Arguments

- `data` An N x d data matrix (with uniform margins).
- `RVM` An RVineMatrix object including the structure and the pair-copula families and parameters.
- `par` A d x d matrix with the pair-copula parameters (optional; default: par = RVM$par).
- `par2` A d x d matrix with the second parameters of pair-copula families with two parameters (optional; default: par2 = RVM$par2).
Details

If $k$ denotes the number of parameters of an R-vine copula model with log-likelihood $l_{RVine}$ and parameter set $\theta$, then the Akaike Information Criterion (AIC) by Akaike (1973) is defined as

$$AIC := -2l_{RVine}(\theta|u) + 2k,$$

for observations $u = (u_1', ..., u_N')$.

Similarly, the Bayesian Information Criterion (BIC) by Schwarz (1978) is given by

$$BIC := -2l_{RVine}(\theta|u) + \log(N)k.$$

Value

AIC, BIC The computed AIC or BIC value, respectively.
pair.AIC, pair.BIC A d x d matrix of individual contributions to the AIC or BIC value for each pair-copula, respectively. Note: AIC = sum(pair.AIC) and similarly BIC = sum(pair.BIC).

Author(s)

Eike Brechmann

References


See Also

RVineLogLik, RVineVuongTest, RVineClarkeTest

Examples

```r
# define 5-dimensional R-vine tree structure matrix
Matrix <- c(5, 2, 3, 1, 4,
0, 2, 3, 4, 1,
0, 0, 3, 4, 1,
0, 0, 0, 4, 1,
0, 0, 0, 0, 1)
Matrix <- matrix(Matrix, 5, 5)

# define R-vine pair-copula family matrix
family <- c(0, 1, 3, 4, 4,
0, 0, 3, 4, 1,
0, 0, 0, 4, 1,
0, 0, 0, 0, 3,
0, 0, 0, 0, 0)
family <- matrix(family, 5, 5)

# define R-vine pair-copula parameter matrix
par <- c(0, 0.2, 0.9, 1.5, 3.9,
0, 0, 1.1, 1.6, 0.9,
0, 0, 0.9, 0.8, 0.7)
```
RVineClarkeTest

```r
par <- matrix(par, 5, 5)

# define second R-vine pair-copula parameter matrix
par2 <- matrix(0, 5, 5)

# define RVineMatrix object
RVM <- RVineMatrix(Matrix = Matrix, family = family, par = par, par2 = par2,
                    names = c("V1", "V2", "V3", "V4", "V5"))

# simulate a sample of size 300 from the R-vine copula model
set.seed(123)
simdata <- RVineSim(300, RVM)

# compute AIC and BIC
RVineAIC(simdata, RVM)
RVineBIC(simdata, RVM)
```

---

RVineClarkeTest

Clarke Test Comparing Two R-Vine Copula Models

**Description**

This function performs a Clarke test between two d-dimensional R-vine copula models as specified by their `RVineMatrix` objects.

**Usage**

RVineClarkeTest(data, RVM1, RVM2)

**Arguments**

data An N x d data matrix (with uniform margins).
RVM1, RVM2 `RVineMatrix` objects of models 1 and 2.

**Details**

The test proposed by Clarke (2007) allows to compare non-nested models. For this let $c_1$ and $c_2$ be two competing vine copulas in terms of their densities and with estimated parameter sets $\theta_1$ and $\theta_2$. The null hypothesis of statistical indistinguishability of the two models is

$$H_0 : P(m_i > 0) = 0.5 \quad \forall i = 1, ..., N,$$

where $m_i := \log \left[ \frac{c_1(u_i|\theta_1)}{c_2(u_i|\theta_2)} \right]$ for observations $u_i$, $i = 1, ..., N$.

Since under statistical equivalence of the two models the log likelihood ratios of the single observations are uniformly distributed around zero and in expectation 50% of the log likelihood ratios greater than zero, the test statistic

$$\text{statistic} := B = \sum_{i=1}^{N} 1_{(0,\infty)}(m_i),$$
where 1 is the indicator function, is distributed Binomial with parameters $N$ and $p = 0.5$, and critical values can easily be obtained. Model 1 is interpreted as statistically equivalent to model 2 if $B$ is not significantly different from the expected value $N p = \frac{N^2}{2}$.

Like AIC and BIC, the Clarke test statistic may be corrected for the number of parameters used in the models. There are two possible corrections; the Akaike and the Schwarz corrections, which correspond to the penalty terms in the AIC and the BIC, respectively.

**Value**

- statistic, statistic.Akaike, statistic.Schwarz
  - Test statistics without correction, with Akaike correction and with Schwarz correction.
- p.value, p.value.Akaike, p.value.Schwarz
  - P-values of tests without correction, with Akaike correction and with Schwarz correction.

**Author(s)**

Jeffrey Dissmann, Eike Brechmann

**References**


**See Also**

RVineVuongTest, RVineAIC, RVineBIC

**Examples**

```r
## Not run:  
# vine structure selection time-consuming (~ 20 sec)  
# load data set  
data(daxreturns)  

# select the R-vine structure, families and parameters  
RVM <- RVineStructureSelect(daxreturns[,1:5], c(1:6))  
RVM$Matrix  
RVM$par  
RVM$par2  

# select the C-vine structure, families and parameters  
CVM <- RVineStructureSelect(daxreturns[,1:5], c(1:6), type = "CVine")  
CVM$Matrix  
CVM$par  
CVM$par2  

# compare the two models based on the data  
clarke <- RVineClarkeTest(daxreturns[,1:5], RVM, CVM)  
clarke$statistic  
clarke$statistic.Schwarz  
clarke$p.value  
clarke$p.value.Schwarz
```
RVineCopSelect  

Sequential Pair-Copula Selection and Estimation for R-Vine Copula Models

Description

This function fits a R-vine copula model to a d-dimensional copula data set. Pair-copula families are selected using BiCopSelect and estimated sequentially.

Usage

RVineCopSelect(data, familyset = NA, Matrix, selectioncrit = "AIC", indeptest = FALSE, level = 0.05, trunclevel = NA, rotations = TRUE)

Arguments

data  An N x d data matrix (with uniform margins).

familyset  An integer vector of pair-copula families to select from (the independence copula MUST NOT be specified in this vector unless one wants to fit an independence vine!). The vector has to include at least one pair-copula family that allows for positive and one that allows for negative dependence. Not listed copula families might be included to better handle limit cases. If familyset = NA (default), selection among all possible families is performed. The coding of pair-copula families is shown below.

Matrix  Lower or upper triangular d x d matrix that defines the R-vine tree structure.

selectioncrit  Character indicating the criterion for pair-copula selection. Possible choices: selectioncrit = "AIC" (default) or "BIC" (see BiCopSelect).

indeptest  Logical; whether a hypothesis test for the independence of u1 and uR is performed before bivariate copula selection (default: indeptest = FALSE; see BiCopIndTest). The independence copula is chosen for a (conditional) pair if the null hypothesis of independence cannot be rejected.

level  Numeric; significance level of the independence test (default: level = 0.05).

trunclevel  Integer; level of truncation.

rotations  If TRUE, all rotations of the families in familyset are included.

Details

R-vine copula models with unknown structure can be specified using RVineStructureSelect.

Value

An RVineMatrix object with the following matrix components

Matrix  R-vine tree structure matrix as given by the argument Matrix.
<table>
<thead>
<tr>
<th>family</th>
<th>Selected pair-copula family matrix with values corresponding to</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>independence copula</td>
</tr>
<tr>
<td>1</td>
<td>Gaussian copula</td>
</tr>
<tr>
<td>2</td>
<td>Student t copula (t-copula)</td>
</tr>
<tr>
<td>3</td>
<td>Clayton copula</td>
</tr>
<tr>
<td>4</td>
<td>Gumbel copula</td>
</tr>
<tr>
<td>5</td>
<td>Frank copula</td>
</tr>
<tr>
<td>6</td>
<td>Joe copula</td>
</tr>
<tr>
<td>7</td>
<td>BB1 copula</td>
</tr>
<tr>
<td>8</td>
<td>BB6 copula</td>
</tr>
<tr>
<td>9</td>
<td>BB7 copula</td>
</tr>
<tr>
<td>10</td>
<td>BB8 copula</td>
</tr>
<tr>
<td>13</td>
<td>rotated Clayton copula (180 degrees; “survival Clayton”)</td>
</tr>
<tr>
<td>14</td>
<td>rotated Gumbel copula (180 degrees; “survival Gumbel”)</td>
</tr>
<tr>
<td>16</td>
<td>rotated Joe copula (180 degrees; “survival Joe”)</td>
</tr>
<tr>
<td>17</td>
<td>rotated BB1 copula (180 degrees; “survival BB1”)</td>
</tr>
<tr>
<td>18</td>
<td>rotated BB6 copula (180 degrees; “survival BB6”)</td>
</tr>
<tr>
<td>19</td>
<td>rotated BB7 copula (180 degrees; “survival BB7”)</td>
</tr>
<tr>
<td>20</td>
<td>rotated BB8 copula (180 degrees; “survival BB8”)</td>
</tr>
<tr>
<td>23</td>
<td>rotated Clayton copula (90 degrees)</td>
</tr>
<tr>
<td>24</td>
<td>rotated Gumbel copula (90 degrees)</td>
</tr>
<tr>
<td>26</td>
<td>rotated Joe copula (90 degrees)</td>
</tr>
<tr>
<td>27</td>
<td>rotated BB1 copula (90 degrees)</td>
</tr>
<tr>
<td>28</td>
<td>rotated BB6 copula (90 degrees)</td>
</tr>
<tr>
<td>29</td>
<td>rotated BB7 copula (90 degrees)</td>
</tr>
<tr>
<td>30</td>
<td>rotated BB8 copula (90 degrees)</td>
</tr>
<tr>
<td>33</td>
<td>rotated Clayton copula (270 degrees)</td>
</tr>
<tr>
<td>34</td>
<td>rotated Gumbel copula (270 degrees)</td>
</tr>
<tr>
<td>36</td>
<td>rotated Joe copula (270 degrees)</td>
</tr>
<tr>
<td>37</td>
<td>rotated BB1 copula (270 degrees)</td>
</tr>
<tr>
<td>38</td>
<td>rotated BB6 copula (270 degrees)</td>
</tr>
<tr>
<td>39</td>
<td>rotated BB7 copula (270 degrees)</td>
</tr>
<tr>
<td>40</td>
<td>rotated BB8 copula (270 degrees)</td>
</tr>
<tr>
<td>104</td>
<td>Tawn type 1 copula</td>
</tr>
<tr>
<td>114</td>
<td>rotated Tawn type 1 copula (180 degrees)</td>
</tr>
<tr>
<td>124</td>
<td>rotated Tawn type 1 copula (90 degrees)</td>
</tr>
<tr>
<td>134</td>
<td>rotated Tawn type 1 copula (270 degrees)</td>
</tr>
<tr>
<td>204</td>
<td>Tawn type 2 copula</td>
</tr>
<tr>
<td>214</td>
<td>rotated Tawn type 2 copula (180 degrees)</td>
</tr>
<tr>
<td>224</td>
<td>rotated Tawn type 2 copula (90 degrees)</td>
</tr>
<tr>
<td>234</td>
<td>rotated Tawn type 2 copula (270 degrees)</td>
</tr>
</tbody>
</table>

| par    | Estimated pair-copula parameter matrix.                       |
| par2   | Estimated second pair-copula parameter matrix with parameters of pair-copula families with two parameters. |

**Author(s)**

Eike Brechmann
References


See Also

RVineStructureSelect, BiCopSelect, RVineSeqEst

Examples

# define 5-dimensional R-vine tree structure matrix
Matrix <- c(5, 2, 3, 1, 4,
            0, 2, 3, 4, 1,
            0, 0, 3, 4, 1,
            0, 0, 0, 4, 1,
            0, 0, 0, 0, 1)
Matrix <- matrix(Matrix, 5, 5)

# define R-vine pair-copula family matrix
family <- c(0, 1, 3, 4, 4,
           0, 0, 3, 4, 1,
           0, 0, 0, 4, 1,
           0, 0, 0, 0, 3,
           0, 0, 0, 0, 0)
family <- matrix(family, 5, 5)

# define R-vine pair-copula parameter matrix
par <- c(0, 0.2, 0.9, 1.5, 3.9,
        0, 0, 1.1, 1.6, 0.9,
        0, 0, 1.9, 0.5,
        0, 0, 0, 4.8,
        0, 0, 0, 0)
par <- matrix(par, 5, 5)

# define second R-vine pair-copula parameter matrix
par2 <- matrix(0, 5, 5)

# define RVineMatrix object
RVM <- RVineMatrix(Matrix = Matrix, family = family,
                    par = par, par2 = par2,
                    names = c("V1", "V2", "V3", "V4", "V5"))

# simulate a sample of size 1000 from the R-vine copula model
set.seed(123)
simdata <- RVineSim(1000, RVM)

# determine the pair-copula families and parameters
RVM1 <- RVineCopSelect(simdata, familyset = c(1, 3, 4, 5, 6), Matrix)
RVineCor2pcor  (Partial) Correlations for R-Vine Copula Models

Description

Correlations to partial correlations and vice versa for R-vines with independence, Gaussian and t-copulas.

Usage

RVineCor2pcor(RVM, corMat)
RVinePcor2cor(RVM)

Arguments

RVM     RVineMatrix defining only the R-vine structure for Cor2pcor and providing as well the partial correlations for Pcor2cor.
corMat  correlation matrix

Value

RVM     RVineMatrix with transformed partial correlations (for Cor2pcor)
cor     correlation matrix (for Pcor2cor)

Note

The behavior of RVinePcor2cor differs from older versions (<= 1.4). The RVM object is now normalized such that the order of the returned correlation matrix conforms with the correlation matrix of the data. If RVM$names are non-default, the initial ordering of the variables cannot be traced back and the matrix has to be interpreted as indicated by the row- and column names.

Examples

## create RVineMatrix-object for Gaussian vine
Matrix <- matrix(c(1, 3, 4, 2,
                   0, 3, 4, 2,
                   0, 0, 4, 2,
                   0, 0, 0, 2), 4, 4)
family <- matrix(c(0, 1, 1, 1,
                    0, 0, 1, 1,
                    0, 0, 0, 1,
                    0, 0, 0, 0), 4, 4)
par <- matrix(c(0, 0.2, 0, 0.6,
                0, 0, 0.2, 0.6,
                0, 0, 0, 0.6,
                0, 0, 0, 0), 4, 4)
RVM <- RVineMatrix(Matrix, family, par)

## calculate correlation matrix corresponding to the R-Vine model
newcor <- RVinePcor2cor(RVM)

## transform back to partial correlations
RVineCor2pcor(RVM, newcor)$par
RVineGofTest 85

### check if they are equal
all.equal(RVM$par, RVineCor2pcor(RVM, newcor)$par)

---

**RVineGofTest**  
*Goodness-of-Fit Tests for R-Vine Copula Models*

**Description**

This function performs a goodness-of-fit test for R-vine copula models. There are 15 different goodness-of-fit tests implemented, described in Schepsmeier (2013).

**Usage**

```r
RVineGofTest(data, RVM, method = "White", statistic = "CvM", B = 200, alpha = 2)
```

**Arguments**

- **data**  
  An N x d data matrix (with uniform margins).

- **RVM**  
  RVineMatrix objects of the R-vine model under the null hypothesis. Only the following copula families are allowed in RVineMatrix due to restrictions in RVineGrad and RVineHessian:
  - 0 = independence copula
  - 1 = Gaussian copula
  - 2 = Student t copula (t-copula)
  - 3 = Clayton copula
  - 4 = Gumbel copula
  - 5 = Frank copula
  - 6 = Joe copula
  - 13 = rotated Clayton copula (180 degrees; “survival Clayton”)
  - 14 = rotated Gumbel copula (180 degrees; “survival Gumbel”)
  - 16 = rotated Joe copula (180 degrees; “survival Joe”)
  - 23 = rotated Clayton copula (90 degrees)
  - 24 = rotated Gumbel copula (90 degrees)
  - 26 = rotated Joe copula (90 degrees)
  - 33 = rotated Clayton copula (270 degrees)
  - 34 = rotated Gumbel copula (270 degrees)
  - 36 = rotated Joe copula (270 degrees)

- **method**  
  A string indicating the goodness-of-fit method:
  - "White" = goodness-of-fit test based on White’s information matrix equality (default)
  - "IR" = goodness-of-fit test based on the information ratio
  - "Breymann" = goodness-of-fit test based on the probability integral transform (PIT) and the aggregation to univariate data by Breymann et al. (2003).
  - "Berg" = goodness-of-fit test based on the probability integral transform (PIT) and the aggregation to univariate data by Berg and Bakken (2007).
  - "Berg2" = second goodness-of-fit test based on the probability integral transform (PIT) and the aggregation to univariate data by Berg and Bakken (2007).
  - "ECP" = goodness-of-fit test based on the empirical copula process (ECP)
"ECP2" = goodness-of-fit test based on the combination of probability integral transform (PIT) and empirical copula process (ECP) (Genest et al. 2009)

**statistic**

A string indicating the goodness-of-fit test statistic type:

- "CvM" = Cramer-von Mises test statistic (univariate for "Breymann", "Berg" and "Berg2", multivariate for "ECP" and "ECP2")
- "KS" = Kolmogorov-Smirnov test statistic (univariate for "Breymann", "Berg" and "Berg2", multivariate for "ECP" and "ECP2")
- "AD" = Anderson-Darling test statistic (only univariate for "Breymann", "Berg" and "Berg2")

**B**

an integer for the number of bootstrap steps (default B = 200)

For B = 0 the asymptotic p-value is returned if available, otherwise only the test statistic is returned.

WARNING: If B is chosen too large, computations will take very long.

**alpha**

an integer of the set 2, 4, 6, ... for the "Berg2" goodness-of-fit test (default alpha = 2)

**Details**

**method = "White":**

This goodness-of fit test uses the information matrix equality of White (1982) and was originally investigated by Huang and Prokhorov (2011) for copulas. Schepsmeier (2012) enhanced their approach to the vine copula case. The main contribution is that under correct model specification the Fisher Information can be equivalently calculated as minus the expected Hessian matrix or as the expected outer product of the score function. The null hypothesis is

\[ H_0 : H(\theta) + C(\theta) = 0 \]

against the alternative

\[ H_1 : H(\theta) + C(\theta) \neq 0, \]

where \( H(\theta) \) is the expected Hessian matrix and \( C(\theta) \) is the expected outer product of the score function.

For the calculation of the test statistic we use the consistent maximum likelihood estimator \( \hat{\theta} \) and the sample counterparts of \( H(\theta) \) and \( C(\theta) \).

The correction of the Covariance-Matrix in the test statistic for the uncertainty in the margins is skipped. The implemented test assumes that there is no uncertainty in the margins. The correction can be found in Huang and Prokhorov (2011) for bivariate copulas and in Schepsmeier (2013) for vine copulas. It involves multi-dimensional integrals.

**method = "IR":**

As the White test the information matrix ratio test is based on the expected Hessian matrix \( H(\theta) \) and the expected outer product of the score function \( C(\theta) \).

\[ H_0 : -H(\theta)^{-1}C(\theta) = I_p \]

against the alternative

\[ H_1 : -H(\theta)^{-1}C(\theta) \neq I_p. \]

The test statistic can then be calculated as

\[ IR_n := tr(\Phi(\theta))/p \]

with \( \Phi(\theta) = -H(\theta)^{-1}C(\theta) \), \( p \) is the number of parameters, i.e. the length of \( \theta \), and \( tr(A) \) is the trace of the matrix \( A \)
For details see Schepsmeier (2013).

method = "Breymann", method = "Berg" and method = "Berg2":

These tests are based on the multivariate probability integral transform (PIT) applied in RVinePIT. The multivariate data \( y_i \) returned from the PIT are aggregated to univariate data by different aggregation functions \( \Gamma(\cdot) \) in the sum

\[
s_t = \sum_{i=1}^{d} \Gamma(y_{id}), t = 1, \ldots, n
\]

In Breymann et al. (2003) the weight function is suggested as \( \Gamma(\cdot) = \Phi^{-1}(\cdot)^2 \), while in Berg and Bakken (2007) the weight function is either \( \Gamma(\cdot) = |\cdot - 0.5|^{\alpha} \), \( \alpha = 2, 4, 6, \ldots \) (method = "Berg") or \( \Gamma(\cdot) = (\cdot - 0.5)^{\alpha} \) (method = "Berg2"). Furthermore, the "Berg" and "Berg2" test are based on the order statistics of the PIT returns. See Berg and Bakken (2007) or Schepsmeier (2013) for details.

method = "ECP" and method = "ECP2":

Both tests are test for \( H_0 : C \in C_0 \) against \( H_1 : C \notin C_0 \) where \( C \) denotes the (vine) copula distribution function and \( C_0 \) is a class of parametric (vine) copulas with \( \Theta \subseteq \mathbb{R}^p \) being the parameter space of dimension \( p \). They are based on the empirical copula process (ECP)

\[
\hat{C}_n(u) - C_{\Theta_n}(u),
\]

with \( u = (u_1, \ldots, u_d) \in [0, 1]^d \) and \( \hat{C}_n(u) = \frac{1}{n+1} \sum_{t=1}^{n} \mathbb{1}\{U_{1t} \leq u_1, \ldots, U_{dt} \leq u_d\} \). The ECP is utilized in a multivariate Cramer-von Mises (CvM) or multivariate Kolmogorov-Smirnov (KS) based test statistic. An extension of the ECP-test is the combination of the multivariate PIT approach with the ECP. The general idea is that the transformed data of a multivariate PIT should be "close" to the independence copula Genest et al. (2009). Thus a distance of CvM or KS type between them is considered. This approach is called ECP2. Again we refer to Schepsmeier (2013) for details.

Value

For method = "White":

White test statistic
p.value p-value, either asymptotic for B = \emptyset or bootstrapped for B > \emptyset

For method = "IR":

IR test statistic
p.value So far no p-value is returned neither a asymptotic nor a bootstrapped one. How to calculate a bootstrapped p-value is explained in Schepsmeier (2013)

For method = "Breymann", method = "Berg" and method = "Berg2":

CvM, KS, AD test statistic according to the choice of statistic
p.value p-value, either asymptotic for B = \emptyset or bootstrapped for B > \emptyset. A asymptotic p-value is only available for the Anderson-Darling test statistic if the R-package ADGofTest is loaded. Furthermore, a asymptotic p-value can be calculated for the Kolmogorov-Smirnov test statistic. For the Cramer-von Mises no asymptotic p-value is available so far.

For method = "ECP" and method = "ECP2":

CvM, KS test statistic according to the choice of statistic
p.value bootstrapped p-value
RVineGrad

Gradient of the Log-Likelihood of an R-Vine Copula Model

Description

This function calculates the gradient of the log-likelihood of a d-dimensional R-vine copula model with respect to the copula parameter and evaluates it on a given copula data set.
Usage

RVineGrad(data, RVM, par = RVM$par, par2 = RVM$par2,
        start.V = NA, posParams = (RVM$family > 0))

Arguments

data An N x d data matrix (with uniform margins).
RVM  An RVineMatrix object including the structure and the pair-copula families and parameters.
      Only the following copula families are allowed in RVM$family
      0 = independence copula
      1 = Gaussian copula
      2 = Student t copula (t-copula)
      3 = Clayton copula
      4 = Gumbel copula
      5 = Frank copula
      6 = Joe copula
      13 = rotated Clayton copula (180 degrees; “survival Clayton”)
      14 = rotated Gumbel copula (180 degrees; “survival Gumbel”)
      16 = rotated Joe copula (180 degrees; “survival Joe”)
      23 = rotated Clayton copula (90 degrees)
      24 = rotated Gumbel copula (90 degrees)
      26 = rotated Joe copula (90 degrees)
      33 = rotated Clayton copula (270 degrees)
      34 = rotated Gumbel copula (270 degrees)
      36 = rotated Joe copula (270 degrees)
par  A d x d matrix with the pair-copula parameters (optional; default: par = RVM$par).
par2 A d x d matrix with the second parameters of pair-copula families with two parameters (optional; default: par2 = RVM$par2).
start.V Transformations (h-functions and log-likelihoods of each pair-copula) of previous calculations (see output; default: start.V = NA).
posParams A d x d matrix indicating which copula has to be considered in the gradient (default: posParams = (RVM$family > 0)).

Details

The ordering of the gradient is due to the ordering of the R-vine matrix. The gradient starts at the lower right corner of the R-vine matrix and goes column by column to the left and up, i.e. the first entry of the gradient is the last entry of the second last column of the par-matrix followed by the last entry of the third last column and the second last entry of this column. If there is a copula family with two parameters, i.e. the t-copula, the derivative with respect to the second parameter is at the end of the gradient vector in order of their occurrence.

Value

gradient The calculated gradient of the log-likelihood value of the R-vine copula model.

Note

The gradient for R-vine copula models with two parameter Archimedean copulas, i.e. BB1, BB6, BB7, BB8 and their rotated versions.
Author(s)

Ulf Schepsmeier, Jakob Stoeber

References


See Also

BiCopDeriv, BiCopDeriv2, BiCopHfuncDeriv, BiCopHfuncDeriv2, RVineMatrix, RVineMLE, RVineHessian

Examples

# define 5-dimensional R-vine tree structure matrix
Matrix <- c(5, 2, 3, 1, 4,
           0, 2, 3, 4, 1,
           0, 0, 3, 4, 1,
           0, 0, 0, 4, 1,
           0, 0, 0, 0, 1)
Matrix <- matrix(Matrix, 5, 5)

# define R-vine pair-copula family matrix
family <- c(0, 1, 3, 4, 4,
            0, 0, 3, 4, 1,
            0, 0, 0, 4, 1,
            0, 0, 0, 0, 3,
            0, 0, 0, 0, 0)
family <- matrix(family, 5, 5)

# define R-vine pair-copula parameter matrix
par <- c(0, 0.2, 0.9, 1.5, 3.9,
        0, 0, 1.1, 1.6, 0.9,
        0, 0, 0, 1.9, 0.5,
        0, 0, 0, 0, 4.8,
        0, 0, 0, 0, 0)
par <- matrix(par, 5, 5)

# define second R-vine pair-copula parameter matrix
par2 <- matrix(0, 5, 5)

# define RVineMatrix object
RVM <- RVineMatrix(Matrix = Matrix, family = family,
                    par = par, par2 = par2,
                    names = c("V1", "V2", "V3", "V4", "V5"))

# simulate a sample of size 300 from the R-vine copula model
set.seed(123)
simdata <- RVineSim(300, RVM)
# compute the gradient of the first row of the data
out2 <- RVineGrad(simdata[,1], RVM)
out2$gradient

---

R Vine Hessian

**Hessian Matrix of the Log-Likelihood of an R-Vine Copula Model**

**Description**

This function calculates the Hessian matrix of the log-likelihood of a d-dimensional R-vine copula model with respect to the copula parameter and evaluates it on a given copula data set.

**Usage**

`RVineHessian(data, RVM)`

**Arguments**

- `data` An N x d data matrix (with uniform margins).
- `RVM` An `RVineMatrix` object including the structure, the pair-copula families, and the parameters.
  
  Only the following copula families are allowed in `RVM$family`:
  
  - `0` = independence copula
  - `1` = Gaussian copula
  - `2` = Student t copula (t-copula) (WARNING: see details)
  - `3` = Clayton copula
  - `4` = Gumbel copula
  - `5` = Frank copula
  - `6` = Joe copula
  - `13` = rotated Clayton copula (180 degrees; “survival Clayton”)
  - `14` = rotated Gumbel copula (180 degrees; “survival Gumbel”)
  - `16` = rotated Joe copula (180 degrees; “survival Joe”)
  - `23` = rotated Clayton copula (90 degrees)
  - `24` = rotated Gumbel copula (90 degrees)
  - `26` = rotated Joe copula (90 degrees)
  - `33` = rotated Clayton copula (270 degrees)
  - `34` = rotated Gumbel copula (270 degrees)
  - `36` = rotated Joe copula (270 degrees)

**Value**

- `hessian` The calculated Hessian matrix of the log-likelihood value of the R-vine copula model.
- `der` The product of the gradient vector with its transposed version.

**Note**

The Hessian matrix is not available for R-vine copula models with two parameter Archimedean copulas, i.e. BB1, BB6, BB7, BB8 and their rotated versions.
Author(s)
Ulf Schepsmeier, Jakob Stoeber

References


See Also
bicopderiv, bicopderivR, bicophfuncderiv, bicophfuncderivR, RVineMatrix, RVineMLE, RVineGrad

Examples

```r
# define 5-dimensional R-vine tree structure matrix
Matrix <- c(5, 2, 3, 1, 4,
           0, 2, 3, 1, 4,
           0, 0, 3, 4, 1,
           0, 0, 0, 4, 1,
           0, 0, 0, 0, 1)
Matrix <- matrix(Matrix, 5, 5)

# define R-vine pair-copula family matrix
family <- c(0, 1, 3, 4, 4,
            0, 0, 3, 4, 1,
            0, 0, 0, 4, 1,
            0, 0, 0, 0, 3,
            0, 0, 0, 0, 0)
family <- matrix(family, 5, 5)

# define R-vine pair-copula parameter matrix
par <- c(0, 0.2, 0.9, 1.5, 3.9,
        0, 0, 1.1, 1.6, 0.9,
        0, 0, 0, 1.9, 0.5,
        0, 0, 0, 0, 4.8,
        0, 0, 0, 0, 0)
par <- matrix(par, 5, 5)

# define second R-vine pair-copula parameter matrix
par2 <- matrix(0, 5, 5)

# define RVineMatrix object
RVM <- RVineMatrix(Matrix = Matrix, family = family,
                    par = par, par2 = par2,
                    names = c("V1", "V2", "V3", "V4", "V5"))

# simulate a sample of size 300 from the R-vine copula model
set.seed(123)
simdata <- RVineSim(300, RVM)
```
RVineLogLik

Log-Likelihood of an R-Vine Copula Model

Description

This function calculates the log-likelihood of a d-dimensional R-vine copula model for a given copula data set.

Usage

RVineLogLik(data, RVM, par = RVM$par, par2 = RVM$par2, separate = FALSE, verbose=TRUE)

Arguments

data  An N x d data matrix (with uniform margins).
RVM  An RVineMatrix object including the structure and the pair-copula families and parameters.
par  A d x d matrix with the pair-copula parameters (optional; default: par = RVM$par).
par2  A d x d matrix with the second parameters of pair-copula families with two parameters (optional; default: par2 = RVM$par2).
separate Logical; whether log-likelihoods are returned point wisely (default: separate = FALSE).
verbose  In case something goes wrong, additional output will be plotted.

Details

For observations \( \mathbf{u} = (u'_1, ..., u'_N)' \) the log-likelihood of a d-dimensional R-vine copula with \( d - 1 \) trees and corresponding edge sets \( E_1, ..., E_{d-1} \) is given by

\[
\log \text{lik} := l_{RVine}(\mathbf{u}) = \sum_{i=1}^{N} \sum_{\ell=1}^{d-1} \sum_{e \in E \ell} \ln \left[ c_{j(e),k(e)|D(e)} \left( F(u_{i,j(e)} | u_{i,D(e)}), F(u_{i,k(e)} | u_{i,D(e)}) | \theta_{j(e),k(e)|D(e)} \right) \right],
\]

where \( u_i = (u_i,1, ..., u_i,d)' \in [0,1]^d, \ i = 1, ..., N \). Further \( c_{j(e),k(e)|D(e)} \) denotes a bivariate copula density associated to an edge \( e \) and with parameter(s) \( \theta_{j(e),k(e)|D(e)} \). Conditional distribution functions such as \( F(u_{i,j(e)} | u_{i,D(e)}) \) are obtained recursively using the relationship

\[
h(u|v, \theta) := F(u|v) = \frac{\partial C_{uvj|v_{-j}}(F(u|v_{-j}), F(v_j|v_{-j}))}{\partial F(v_j|v_{-j})},
\]

where \( C_{uvj|v_{-j}} \) is a bivariate copula distribution function with parameter(s) \( \theta \) and \( v_{-j} \) denotes a vector with the \( j \)-th component \( v_j \) removed. The notation of h-functions is introduced for convenience. For more details see Dissmann et al. (2013).
Value

loglik  The calculated log-likelihood value of the R-vine copula model.

V  The stored transformations (h-functions and log-likelihoods of each pair-copula) which may be used for posterior updates (three matrices: direct, indirect and value).

Author(s)

Ulf Schepsmeier, Jeffrey Dissmann, Jakob Stoeber

References


See Also

BiCopHfunc, RVineMatrix, RVineMLE, RVineAIC, RVineBIC

Examples

# define 5-dimensional R-vine tree structure matrix
Matrix <- c(5, 2, 3, 1, 4,
0, 2, 3, 4, 1,
0, 0, 3, 4, 1,
0, 0, 0, 4, 1,
0, 0, 0, 0, 1)
Matrix <- matrix(Matris, 5, 5)

# define R-vine pair-copula family matrix
family <- c(0, 1, 3, 4, 4,
0, 0, 3, 4, 1,
0, 0, 0, 4, 1,
0, 0, 0, 0, 3,
0, 0, 0, 0, 0)
family <- matrix(family, 5, 5)

# define R-vine pair-copula parameter matrix
par <- c(0, 0.2, 0.9, 1.5, 3.9,
0, 0, 1.1, 1.6, 0.9,
0, 0, 1.9, 0.5,
0, 0, 0, 4.8,
0, 0, 0, 0, 0)
par <- matrix(par, 5, 5)

# define second R-vine pair-copula parameter matrix
par2 <- matrix(0, 5, 5)

# define RVineMatrix object
RVM <- RVineMatrix(Matrix = Matrix, family = family,
par = par, par2 = par2,
names = c("V1", "V2", "V3", "V4", "V5"))

# simulate a sample of size 300 from the R-vine copula model
set.seed(123)
simdata <- RVineSim(300, RVM)

# compute the log-likelihood
ll <- RVineLogLik(simdata, RVM, separate = FALSE)
ll$loglik

# compute the pointwise log-likelihoods
ll <- RVineLogLik(simdata, RVM, separate = TRUE)
ll$loglik

---

**RVineMatrix**

**R-Vine Copula Model in Matrix Notation**

**Description**

This function creates an `RVineMatrix` object which encodes an R-vine copula model. It contains the matrix identifying the R-vine tree structure, the matrix identifying the copula families utilized and two matrices for corresponding parameter values.

**Usage**

```r
RVineMatrix(Matrix, family = array(0, dim = dim(Matrix)),
par = array(NA, dim = dim(Matrix)),
par2 = array(NA, dim = dim(Matrix)), names=NULL)
```

**Arguments**

- **Matrix**
  Lower (or upper) triangular $d \times d$ matrix that defines the R-vine tree structure.
- **family**
  Lower (or upper) triangular $d \times d$ matrix with zero diagonal entries that assigns the pair-copula families to each (conditional) pair defined by `Matrix` (default: `family = array(0, dim=dim(Matrix)))`. The bivariate copula families are defined as follows:
  - $0 =$ independence copula
  - $1 =$ Gaussian copula
  - $2 =$ Student t copula (t-copula)
  - $3 =$ Clayton copula
  - $4 =$ Gumbel copula
  - $5 =$ Frank copula
  - $6 =$ Joe copula
  - $7 =$ BB1 copula
  - $8 =$ BB6 copula
  - $9 =$ BB7 copula
  - $10 =$ BB8 copula
  - $13 =$ rotated Clayton copula (180 degrees; “survival Clayton”)
  - $14 =$ rotated Gumbel copula (180 degrees; “survival Gumbel”)
  - $16 =$ rotated Joe copula (180 degrees; “survival Joe”)
  - $17 =$ rotated BB1 copula (180 degrees; “survival BB1”)
  - $18 =$ rotated BB6 copula (180 degrees; “survival BB6”)
  - $19 =$ rotated BB7 copula (180 degrees; “survival BB7”)
  - $20 =$ rotated BB8 copula (180 degrees; “survival BB8”)
  - $23 =$ rotated Clayton copula (90 degrees)
4 = rotated Gumbel copula (90 degrees)
6 = rotated Joe copula (90 degrees)
7 = rotated BB1 copula (90 degrees)
8 = rotated BB6 copula (90 degrees)
9 = rotated BB7 copula (90 degrees)
10 = rotated BB8 copula (90 degrees)
11 = rotated Clayton copula (270 degrees)
12 = rotated Gumbel copula (270 degrees)
13 = rotated Joe copula (270 degrees)
14 = rotated BB1 copula (270 degrees)
15 = rotated BB6 copula (270 degrees)
16 = rotated BB7 copula (270 degrees)
17 = rotated BB8 copula (270 degrees)
18 = rotated Tawn type 1 copula
19 = rotated Tawn type 1 copula (180 degrees)
20 = rotated Tawn type 1 copula (90 degrees)
21 = rotated Tawn type 1 copula (270 degrees)
22 = rotated Tawn type 2 copula (90 degrees)
23 = rotated Tawn type 2 copula (270 degrees)
24 = rotated Tawn type 2 copula (180 degrees)
25 = rotated Tawn type 2 copula

par Lower (or upper) triangular d x d matrix with zero diagonal entries that assigns the (first) pair-copula parameter to each (conditional) pair defined by Matrix (default: \( \text{par} = \text{array}(\text{NA}, \text{dim} = \text{dim}(\text{Matrix})) \)).

par2 Lower (or upper) triangular d x d matrix with zero diagonal entries that assigns the second parameter for pair-copula families with two parameters to each (conditional) pair defined by Matrix (default: \( \text{par2} = \text{array}(\text{NA}, \text{dim} = \text{dim}(\text{Matrix})) \)).

names A vector of names for the d variables; default: names = \text{NULL}.

Value

An \text{RVineMatrix} object with the following matrix components:

Matrix R-vine tree structure matrix.

family Pair-copula family matrix with values as above.

par Pair-copula parameter matrix.

par2 Second pair-copula parameter matrix with parameters necessary for pair-copula families with two parameters.

Note

The print function writes the R-vine matrix defined by Matrix. A detailed output is given by print(RVM, detail=TRUE), where RVM is the \text{RVineMatrix} object.

The \text{RVineMatrix} function automatically checks if the given matrix is a valid R-vine matrix (see \text{RVineMatrixCheck}).

Although the function allows upper triangular matrices as its input, it will always store them as lower triangular matrices.

Author(s)

Jeffrey Dissmann
RVineMatrixCheck

References


See Also

RVineMatrixCheck, RVineMLE, RVineSim, C2RVine, D2RVine

Examples

# define 5-dimensional R-vine tree structure matrix
Matrix <- c(5, 2, 3, 4, 1,
0, 2, 3, 4, 1,
0, 0, 3, 4, 1,
0, 0, 0, 4, 1,
0, 0, 0, 0, 1)
Matrix <- matrix(Matrix, 5, 5)

# define R-vine pair-copula family matrix
family <- c(0, 1, 3, 4, 4,
0, 0, 3, 4, 1,
0, 0, 0, 4, 1,
0, 0, 0, 0, 3,
0, 0, 0, 0, 0)
family <- matrix(family, 5, 5)

# define R-vine pair-copula parameter matrix
par <- c(0, 0.2, 0.9, 1.5, 3.9,
0, 0, 1.1, 1.6, 0.9,
0, 0, 1.9, 0.5,
0, 0, 0, 4.8,
0, 0, 0, 0)
par <- matrix(par, 5, 5)

# define second R-vine pair-copula parameter matrix
par2 <- matrix(0, 5, 5)

# define RVineMatrix object
RVM <- RVineMatrix(Matrix = Matrix, family = family,
par = par, par2 = par2,
names = c("V1", "V2", "V3", "V4", "V5"))

# Print detailed information
print(RVM, detail = TRUE)

RVineMatrixCheck

R-Vine Matrix Check

Description

The given matrix is tested to be a valid R-vine matrix.
Usage
RVineMatrixCheck(M)

Arguments
M  A dxd vine matrix: only lower triangle is used; For the check, M is assumed to be in natural order, i.e. d:1 on diagonal. Further M[j+1,j]=d-j and M[j,j]=d-j

Value
code  1 for OK;
-3 diagonal can not be put in order d:1;
-2 for not permutation of j:d in column d-j;
-1 if cannot find proper binary array from array in natural order.

Note
The matrix M do not have to be given in natural order or the diagonal in order d:1. The test checks if it can be done in order to be a valid R-vine matrix.
If a function in this package needs the natural order the RVineMatrix object is automatically "normalized".
The function RVineMatrix automatically checks if the given R-vine matrix is valid.

Author(s)
Harry Joe

References

See Also
RVineMatrix

Examples
A1 <- matrix(c(6, 0, 0, 0, 0, 0,
5, 5, 0, 0, 0, 0,
3, 4, 4, 0, 0, 0,
4, 3, 3, 3, 0, 0,
1, 1, 2, 2, 2, 0,
2, 2, 1, 1, 1, 1), 6, 6, byrow = TRUE)
b1 <- RVineMatrixCheck(A1)
# improper vine matrix, code=-1
A2 <- matrix(c(6, 0, 0, 0, 0, 0,
5, 5, 0, 0, 0, 0,
4, 4, 4, 0, 0, 0,
1, 3, 3, 3, 0, 0,
3, 1, 2, 2, 2, 0,
2, 2, 1, 1, 1, 1), 6, 6, byrow = TRUE)
b2 <- RVineMatrixCheck(A2)
RVineMatrixNormalize  Normalization of R-Vine Matrix

Description

An RVineMatrix is permuted to achieve a natural ordering (i.e. diag(RVM$Matrix) == d:1)

Usage

RVineMatrixNormalize(RVM)

Arguments

RVM  RVineMatrix defining the R-vine structure

Value

RVM  An RVineMatrix in natural ordering with entries in RVM$names keeping track of the reordering.

Examples

Matrix <- matrix(c(5, 2, 3, 1, 4, 0, 2, 3, 4, 1, 0, 0, 3, 4, 1, 0, 0, 0, 4, 1, 0, 0, 0, 0, 1), 5, 5)
family <- matrix(1,5,5)
par <- matrix(c(0, 0.2, 0.9, 0.5, 0.8, 0, 0, 0.1, 0.6, 0.9, 0, 0, 0.7, 0.5, 0, 0, 0, 0, 0.8, 0, 0, 0, 0, 0), 5, 5)

# define RVineMatrix object
RVM <- RVineMatrix(Matrix, family, par)

# normalise the RVine
RVineMatrixNormalize(RVM)
Maximum Likelihood Estimation of an R-Vine Copula Model

**Description**

This function calculates the maximum likelihood estimate (MLE) of the R-vine copula model parameters using sequential estimates as initial values (if not provided).

**Usage**

```r
RVineMLE(data, RVM, start = RVM$par, start2 = RVM$par2,  
  maxit = 200, max.df = 30,  
  max.BB = list(BB1=c(5,6),BB6=c(6,6),BB7=c(5,6),BB8=c(6,1)),  
  grad = FALSE, hessian = FALSE, se = FALSE, ...)
```

**Arguments**

- `data` An N x d data matrix (with uniform margins).
- `RVM` An RVineMatrix object including the structure and the pair-copula families and parameters (if known).
- `start` Lower triangular d x d matrix with zero diagonal entries with starting values for the pair-copula parameters (optional; otherwise they are calculated via RVineSeqEst; default: `start = RVM$par`).
- `start2` Lower triangular d x d matrix with zero diagonal entries with starting values for the second parameters of pair-copula families with two parameters (optional; otherwise they are calculated via RVineSeqEst; default: `start2 = RVM$par2`).
- `maxit` The maximum number of iteration steps (optional; default: `maxit = 200`).
- `max.df` Numeric; upper bound for the estimation of the degrees of freedom parameter of the t-copula (default: `max.df = 30`; for more details see BiCopest).
- `max.BB` List; upper bounds for the estimation of the two parameters (in absolute values) of the BB1, BB6, BB7 and BB8 copulas (default: `max.BB = list(BB1=c(5,6),BB6=c(6,6),BB7=c(5,6),BB8=c(6,1)`).
- `grad` If RVMSfamily only contains one parameter copula families or the t-copula the analytical gradient can be used for maximization of the log-likelihood (see RVineGrad; default: `grad = FALSE`).
- `hessian` Logical; whether the Hessian matrix of parameter estimates is estimated (default: `hessian = FALSE`). Note that this is not the Hessian Matrix calculated via RVineHessian but via finite differences.
- `se` Logical; whether standard errors of parameter estimates are estimated on the basis of the Hessian matrix (see above; default: `se = FALSE`).
- `...` Further arguments for optim (e.g. factr controls the convergence of the "L-BFGS-B" method, or trace, a non-negative integer, determines if tracing information on the progress of the optimization is produced.) For more details see the documentation of optim.
Value

- **RVM**: `RVineMatrix` object with the calculated parameters stored in `RVM$par` and `RVM$par2`.
- **value**: Optimized log-likelihood value corresponding to the estimated pair-copula parameters.
- **convergence**: An integer code indicating either successful convergence (`convergence = 0`) or an error:
  - 1 = the iteration limit `maxit` has been reached
  - 51 = a warning from the "L-BFGS-B" method; see component `message` for further details
  - 52 = an error from the "L-BFGS-B" method; see component `message` for further details
- **message**: A character string giving any additional information returned by `optim`, or `NULL`.
- **counts**: A two-element integer vector giving the number of calls to `fn` and `gr` respectively. This excludes those calls needed to compute the Hessian, if requested, and any calls to `fn` to compute a finite-difference approximation to the gradient.
- **hessian**: If `hessian = TRUE`, the Hessian matrix is returned. Its calculation is on the basis of finite differences (output of `optim`).
- **se**: If `se = TRUE`, the standard errors of parameter estimates are returned. Their calculation is based on the Hesse matrix (see above).

Note

`RVineMLE` uses the L-BFGS-B method for optimization.
If the analytical gradient is used for maximization, computations may be up to 10 times faster than using finite differences.

Author(s)

Ulf Schepsmeier, Jeffrey Dissmann

References


See Also

`RVineSeqEst,RVineStructureSelect,RVineMatrix,RVineGrad,RVineHessian`

Examples

```R
# define 5-dimensional R-vine tree structure matrix
Matrix <- c(5, 2, 3, 1, 4,
           0, 2, 3, 4, 1,
           0, 0, 3, 4, 1,
           0, 0, 0, 4, 1,
           0, 0, 0, 0, 1)
```
Matrix <- matrix(5, 5)

# define R-vine pair-copula family matrix
family <- c(0, 1, 3, 4, 4,
            0, 0, 3, 4, 1,
            0, 0, 0, 4, 1,
            0, 0, 0, 0, 3,
            0, 0, 0, 0, 0)
family <- matrix(family, 5, 5)

# define R-vine pair-copula parameter matrix
par <- c(0, 0.2, 0.9, 1.5, 3.9,
        0, 0, 1.1, 1.6, 0.9,
        0, 0, 1.9, 0.5,
        0, 0, 0, 4.8,
        0, 0, 0, 0, 0)
par <- matrix(par, 5, 5)

# define second R-vine pair-copula parameter matrix
par2 <- matrix(0, 5, 5)

# define RVineMatrix object
RVM <- RVineMatrix(Matrix = Matrix, family = family,
                    par = par, par2 = par2,
                    names = c("V1", "V2", "V3", "V4", "V5"))

# simulate a sample of size 300 from the R-vine copula model
set.seed(123)
simdata <- RVineSim(300, RVM)

# compute the MLE
mle <- RVineMLE(simdata, RVM, grad = TRUE, trace = 0)

# compare parameters
round(mle$RVM$par - RVM$par, 2)

RVinePar2Beta

Blomqvist's Beta Values of an R-Vine Copula Model

Description
This function computes the values of Blomqvist’s beta corresponding to the parameters of an R-vine copula model.

Usage
RVinePar2Beta(RVM)

Arguments
RVM An RVineMatrix object.
Note that the Student’s t-copula is not allowed since the CDF of the t-copula is not implemented (see BiCopCDF and BiCopPar2Beta).
Value

Matrix with the same structure as the family and parameter matrices of the `RVineMatrix` object `RVM` where the entries are values of Blomqvist’s beta corresponding to the families and parameters of the R-vine copula model given by `RVM`.

Author(s)

Ulf Schepsmeier

See Also

`RVineMatrix`, `BiCopPar2Beta`

Examples

```R
# define 5-dimensional R-vine tree structure matrix
Matrix <- c(5, 2, 3, 1, 4,
           0, 2, 3, 4, 1,
           0, 0, 3, 4, 1,
           0, 0, 0, 4, 1,
           0, 0, 0, 0, 1)
Matrix <- matrix(Matrix, 5, 5)

# define R-vine pair-copula family matrix
family <- c(0, 1, 3, 4, 4,
           0, 0, 3, 4, 1,
           0, 0, 0, 4, 1,
           0, 0, 0, 0, 3,
           0, 0, 0, 0, 0)
family <- matrix(family, 5, 5)

# define R-vine pair-copula parameter matrix
par <- c(0, 0.2, 0.9, 1.5, 3.9,
         0, 0, 1.1, 1.6, 0.9,
         0, 0, 1.9, 0.5,
         0, 0, 0, 4.8,
         0, 0, 0, 0)
par <- matrix(par, 5, 5)

# define second R-vine pair-copula parameter matrix
par2 <- matrix(0, 5, 5)

# define RVineMatrix object
RVM <- RVineMatrix(Matrix = Matrix, family = family,
                    par = par, par2 = par2,
                    names = c("V1", "V2", "V3", "V4", "V5"))

# compute the Blomqvist's beta values
BlomBeta <- RVinePar2Beta(RVM)
```
RVinePar2Tau  

\textit{Kendall’s Tau Values of an R-Vine Copula Model}

\textbf{Description}

This function computes the values of Kendall’s tau corresponding to the parameters of an R-vine copula model.

\textbf{Usage}

RVinePar2Tau(RVM)

\textbf{Arguments}

\begin{itemize}
  \item \textbf{RVM} \hspace{1em} An \texttt{RVineMatrix} object.
\end{itemize}

\textbf{Value}

Matrix with the same structure as the family and parameter matrices of the \texttt{RVineMatrix} object \texttt{RVM} where the entries are values of Kendall’s tau corresponding to the families and parameters of the R-vine copula model given by \texttt{RVM}.

\textbf{Author(s)}

Jeffrey Dissmann

\textbf{See Also}

\texttt{RVineMatrix,BiCopPar2Tau}

\textbf{Examples}

\begin{verbatim}
# define 5-dimensional R-vine tree structure matrix
Matrix <- c(5, 2, 3, 1, 4,
           0, 2, 3, 4, 1,
           0, 0, 3, 4, 1,
           0, 0, 0, 4, 1,
           0, 0, 0, 0, 1)
Matrix <- matrix(Matrix, 5, 5)

# define R-vine pair-copula family matrix
family <- c(0, 1, 3, 4, 4,
           0, 0, 3, 4, 1,
           0, 0, 0, 4, 1,
           0, 0, 0, 0, 3,
           0, 0, 0, 0, 0)
family <- matrix(family, 5, 5)

# define R-vine pair-copula parameter matrix
par <- c(0, 0.2, 0.9, 1.5, 3.9,
         0, 0, 1.1, 1.6, 0.9,
         0, 0, 0.9, 0.5,
         0, 0, 0, 4.8,
         0, 0, 0, 0)
\end{verbatim}
RVinePDF

PDF of an R-Vine Copula Model

Description

This function calculates the probability density function of a d-dimensional R-vine copula.

Usage

RVinePDF(newdata, RVM)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>newdata</td>
<td>An N x d data matrix that specifies where the density shall be evaluated.</td>
</tr>
<tr>
<td>RVM</td>
<td>An RVineMatrix object including the structure and the pair-copula families and parameters.</td>
</tr>
</tbody>
</table>

Details

The density of a d-dimensional R-vine copula with d−1 trees and corresponding edge sets E_1, ..., E_{d−1} is given by

\[
\prod_{\ell=1}^{d-1} \prod_{e \in E_\ell} c_{j(e), k(e)|D(e)} \left( F(u_{j(e)}|u_{D(e)}), F(u_{k(e)}|u_{D(e)}) \right) \],
\]

where u = (u_1, ..., u_d)' ∈ [0, 1]^d. Further c_{j(e), k(e)|D(e)} denotes a bivariate copula density associated to an edge e and with parameter(s) \( \theta_{j(e), k(e)|D(e)} \). Conditional distribution functions such as \( F(u_{j(e)}|u_{D(e)}) \) are obtained recursively using the relationship

\[
h(u|v, \theta) := F(u|v) = \frac{\partial C_{uvj|v_{-j}}(F(u|v_{-j}), F(v_j|v_{-j}))}{\partial F(v_j|v_{-j})},
\]

where \( C_{uvj|v_{-j}} \) is a bivariate copula distribution function with parameter(s) \( \theta \) and \( v_{-j} \) denotes a vector with the j-th component \( v_j \) removed. The notation of h-functions is introduced for convenience. For more details see Dissmann et al. (2013).

The function is actually just a wrapper to RVineLogLik.

Author(s)

Thomas Nagler
References


See Also

BiCopHfunc, RVineMatrix, RVineMLE, RVineAIC, RVineBIC

Examples

```r
# define 5-dimensional R-vine tree structure matrix
Matrix <- c(5, 2, 3, 1, 4,
           0, 2, 3, 4, 1,
           0, 0, 3, 4, 1,
           0, 0, 0, 4, 1,
           0, 0, 0, 0, 1)
Matrix <- matrix(Matrix, 5, 5)

# define R-vine pair-copula family matrix
family <- c(0, 1, 3, 4, 4,
           0, 0, 3, 4, 1,
           0, 0, 0, 4, 1,
           0, 0, 0, 0, 3,
           0, 0, 0, 0, 0)
family <- matrix(family, 5, 5)

# define R-vine pair-copula parameter matrix
par <- c(0, 0.2, 0.9, 1.5, 3.9,
        0, 0, 1.1, 1.6, 0.9,
        0, 0, 0, 1.9, 0.5,
        0, 0, 0, 0.4, 0.8,
        0, 0, 0, 0, 0)
par <- matrix(par, 5, 5)

# define second R-vine pair-copula parameter matrix
par2 <- matrix(0, 5, 5)

# define RVineMatrix object
RVM <- RVineMatrix(Matrix = Matrix, family = family,
                   par = par, par2 = par2,
                   names = c("V1", "V2", "V3", "V4", "V5"))

# compute the density at (0.1, 0.2, 0.3, 0.4, 0.5)
RVinePDF(c(0.1, 0.2, 0.3, 0.4, 0.5), RVM)
```

**RVinePIT**  
Probability Integral Transformation for R-Vine Copula Models

**Description**

This function applies the probability integral transformation (PIT) for R-vine copula models to given copula data.
Usage

RVinePIT(data, RVM)

Arguments

data An N x d data matrix (with uniform margins).

RVM RVineMatrix objects of the R-vine model.

Details

The multivariate probability integral transformation (PIT) of Rosenblatt (1952) transforms the copula data $u = (u_1, \ldots, u_d)$ with a given multivariate copula $C$ into independent data in $[0, 1]^d$, where $d$ is the dimension of the data set.

Let $u = (u_1, \ldots, u_d)$ denote copula data of dimension $d$. Further let $C$ be the joint cdf of $u = (u_1, \ldots, u_d)$. Then Rosenblatt’s transformation of $u$, denoted as $y = (y_1, \ldots, y_d)$, is defined as

$$y_1 := u_1, \quad y_2 := C(u_2 | u_1), \ldots, \quad y_d := C(u_d | u_1, \ldots, u_{d-1}),$$

where $C(u_k | u_1, \ldots, u_{k-1})$ is the conditional copula of $U_k$ given $U_1 = u_1, \ldots, U_{k-1} = u_{k-1}, k = 2, \ldots, d$. The data vector $y = (y_1, \ldots, y_d)$ is now i.i.d. with $y_i \sim U[0, 1]$. The algorithm for the R-vine PIT is given in the appendix of Schepsmeier (2013).

Value

An N x d matrix of PIT data from the given R-vine copula model.

Author(s)

Ulf Schepsmeier

References


See Also

RVineGofTest

Examples

```r
# load data set
data(daxreturns)

# select the R-vine structure, families and parameters
RVM <- RVineStructureSelect(daxreturns[,1:3], c(1:6))

# PIT data
pit <- RVinePIT(daxreturns[,1:3], RVM)
par(mfrow = c(1,2))
```
Description

This function sequentially estimates the pair-copula parameters of a d-dimensional R-vine copula model as specified by the corresponding \texttt{RVineMatrix} object.

Usage

\begin{verbatim}
RVineSeqEst(data, RVM, method = "mle", se = FALSE, max.df = 30,
            max.BB = list(BB1=c(5,6),BB6=c(6,6),BB7=c(5,6),BB8=c(6,1)),
            progress = FALSE, weights = NA)
\end{verbatim}

Arguments

- **data**: An \( N \times d \) data matrix (with uniform margins).
- **RVM**: An \texttt{RVineMatrix} object including the structure, the pair-copula families and the pair-copula parameters (if they are known).
- **method**: Character indicating the estimation method: either pairwise maximum likelihood estimation (method = "mle"; default) or inversion of Kendall’s tau (method = "itau"; see \texttt{BiCopEst}). For method = "itau" only one parameter pair-copula families can be used (family = 1, 3, 4, 5, 6, 13, 14, 16, 23, 24, 26, 33, 34 or 36).
- **se**: Logical; whether standard errors are estimated (default: \texttt{se = FALSE}).
- **max.df**: Numeric; upper bound for the estimation of the degrees of freedom parameter of the t-copula (default: \texttt{max.df = 30}; for more details see \texttt{BiCopEst}).
- **max.BB**: List; upper bounds for the estimation of the two parameters (in absolute values) of the BB1, BB6, BB7 and BB8 copulas (default: \texttt{max.BB = list(BB1=c(5,6),BB6=c(6,6),BB7=c(5,6),BB8=c(6,1)))}.
- **progress**: Logical; whether the pairwise estimation progress is printed (default: \texttt{progress = FALSE}).
- **weights**: Numerical; weights for each observation (optional).

Details

The pair-copula parameter estimation is performed tree-wise, i.e., for each R-vine tree the results from the previous tree(s) are used to calculate the new copula parameters using \texttt{BiCopEst}.

Value

- **RVM**: \texttt{RVineMatrix} object with the sequentially estimated parameters stored in \texttt{RVM$par} and \texttt{RVM$par2}.
- **se**: Lower triangular \( d \times d \) matrix with estimated standard errors of the (first) pair-copula parameters for each (conditional) pair defined in the \texttt{RVineMatrix} object (if \texttt{se = TRUE}).
se2  
Lower triangular \(d \times d\) matrix with estimated standard errors of the second parameters for pair-copula families with two parameters for each (conditional) pair defined in the \texttt{RVineMatrix} object (if se = TRUE).

Author(s)

Ulf Schepsmeier, Jeffrey Dissmann

See Also

\texttt{BICopEst}, \texttt{BICopHfunc}, \texttt{RVineLogLik}, \texttt{RVineMLE}, \texttt{RVineMatrix}

Examples

# define 5-dimensional R-vine tree structure matrix
Matrix <- c(5, 2, 3, 1, 4,
0, 2, 3, 4, 1,
0, 0, 3, 4, 1,
0, 0, 0, 4, 1,
0, 0, 0, 0, 1)
Matrix <- matrix(Matrix, 5, 5)

# define R-vine pair-copula family matrix
family <- c(0, 1, 3, 4, 4,
0, 0, 3, 4, 1,
0, 0, 0, 4, 1,
0, 0, 0, 0, 3,
0, 0, 0, 0, 0)
family <- matrix(family, 5, 5)

# define R-vine pair-copula parameter matrix
par <- c(0, 0.2, 0.9, 1.5, 3.9,
0, 0.1, 1.6, 0.9,
0, 0, 1.9, 0.5,
0, 0, 0, 4.8,
0, 0, 0, 0, 0)
par <- matrix(par, 5, 5)

# define second R-vine pair-copula parameter matrix
par2 <- matrix(0, 5, 5)

# define RVineMatrix object
RVM <- RVineMatrix(Matrix = Matrix, family = family,
par = par, par2 = par2,
names = c("V1", "V2", "V3", "V4", "V5"))

# simulate a sample of size 300 from the R-vine copula model
set.seed(123)
simdata <- RVineSim(300, RVM)

# sequential estimation
RVineSeqEst(simdata, RVM, method = "itau", se = TRUE)
RVineSeqEst(simdata, RVM, method = "mle", se = TRUE)
RVineSim

Simulation from an R-Vine Copula Model

Description

This function simulates from a given R-vine copula model.

Usage

RVineSim(N, RVM, U = NULL)

Arguments

N   Number of d-dimensional observations to simulate.
RVM An RVineMatrix object containing the information of the R-vine copula model.
U   If not NULL, an (N,d)-matrix of U[0,1] random variates to be transformed to the copula sample.

Value

An N x d matrix of data simulated from the given R-vine copula model.

Author(s)

Jeffrey Dissmann

References


See Also

RVineMatrix, BiCopSim

Examples

# define 5-dimensional R-vine tree structure matrix
Matrix <- c(5, 2, 3, 1, 4,
           0, 2, 3, 4, 1,
           0, 0, 3, 4, 1,
           0, 0, 0, 4, 1,
           0, 0, 0, 0, 1)
Matrix <- matrix(Matrix, 5, 5)

# define R-vine pair-copula family matrix
family <- c(0, 1, 3, 4, 4,
           0, 0, 3, 4, 1,
           0, 0, 0, 4, 1,
           0, 0, 0, 0, 3,
           0, 0, 0, 0, 0)
family <- matrix(family, 5, 5)
# define R-vine pair-copula parameter matrix
par <- c(0, 0.2, 0.9, 1.5, 3.9,
        0, 0, 1.1, 1.6, 0.9,
        0, 0, 0.19, 0.5,
        0, 0, 0, 4.8,
        0, 0, 0, 0, 0)
par <- matrix(par, 5, 5)

# define second R-vine pair-copula parameter matrix
par2 <- matrix(0, 5, 5)

# define RVineMatrix object
RVM <- RVineMatrix(Matrix = Matrix, family = family,
par = par, par2 = par2,
names = c("V1", "V2", "V3", "V4", "V5"))

# simulate a sample of size 300 from the R-vine copula model
set.seed(123)
simdata <- RVineSim(300, RVM)

----------

RVineStdError  Standard Errors of an R-Vine Copula Model

Description

This function calculates the standard errors of a d-dimensional R-vine copula model given the Hessian matrix.

Usage

RVineStdError(hessian, RVM)

Arguments

hessian  The Hessian matrix of the given R-vine.
RVM  An RVineMatrix object including the structure, the pair-copula families, and the parameters.

Value

se  The calculated standard errors for the first parameter matrix. The entries are ordered with respect to the ordering of the RVM$par matrix.
se2  The calculated standard errors for the second parameter matrix.

Note

The negative Hessian matrix should be positive semidefinite. Otherwise NAs will be returned in some entries and the non-NA entries may be wrong. If the negative Hessian matrix is negative definite, then one could try a near positive matrix. The package Matrix provides a function called nearPD to estimate a matrix which is positive definite and close to the given matrix.
Author(s)
Ulf Schepsmeier, Jakob Stoeber

References


See Also
bicopderiv, bicopderivR, bicophfuncderiv, bicophfuncderivR, RVineMatrix, RVineHessian, RVineGrad

Examples
# define 5-dimensional R-vine tree structure matrix
Matrix <- c(5, 2, 3, 1, 4,
           0, 2, 3, 4, 1,
           0, 0, 3, 4, 1,
           0, 0, 0, 4, 1,
           0, 0, 0, 0, 1)
Matrix <- matrix(Matrix, 5, 5)

# define R-vine pair-copula family matrix
family <- c(0, 1, 3, 4, 4,
           0, 0, 3, 4, 1,
           0, 0, 0, 4, 1,
           0, 0, 0, 0, 3,
           0, 0, 0, 0, 0)
family <- matrix(family, 5, 5)

# define R-vine pair-copula parameter matrix
par <- c(0, 0.2, 0.9, 1.5, 3.9,
         0, 0, 1.1, 1.6, 0.9,
         0, 0, 0, 1.9, 0.5,
         0, 0, 0, 0, 4.8,
         0, 0, 0, 0, 0)
par <- matrix(par, 5, 5)

# define second R-vine pair-copula parameter matrix
par2 <- matrix(0, 5, 5)

# define RVineMatrix object
RVM <- RVineMatrix(Matrix = Matrix, family = family,
                    par = par, par2 = par2,
                    names = c("V1", "V2", "V3", "V4", "V5"))

# simulate a sample of size 300 from the R-vine copula model
set.seed(123)
simdata <- RVineSim(300, RVM)
RVineStructureSelect

```r
# compute the Hessian matrix of the first row of the data
out2 <- RVineHessian(simdata,RVM)

# get the standard errors
RVineStdError(out2$hessian, RVM)
```

**Description**

This function fits either an R- or a C-vine copula model to a d-dimensional copula data set. Tree structures are determined and appropriate pair-copula families are selected using `BiCopSelect` and estimated sequentially (forward selection of trees).

**Usage**

```r
RVineStructureSelect(data, familyset = NA, type = 0, selectioncrit = "AIC",
                    indeptest = FALSE, level = 0.05, trunclevel = NA,
                    progress = FALSE, weights = NA, rotations = TRUE)
```

**Arguments**

- `data` An N x d data matrix (with uniform margins).
- `familyset` An integer vector of pair-copula families to select from (the independence copula MUST NOT be specified in this vector unless one wants to fit an independence vine!). The vector has to include at least one pair-copula family that allows for positive and one that allows for negative dependence. Not listed copula families might be included to better handle limit cases. If `familyset = NA` (default), selection among all possible families is performed. Coding of pair-copula families:
  1 = Gaussian copula
  2 = Student t copula (t-copula)
  3 = Clayton copula
  4 = Gumbel copula
  5 = Frank copula
  6 = Joe copula
  7 = BB1 copula
  8 = BB6 copula
  9 = BB7 copula
  10 = BB8 copula
  13 = rotated Clayton copula (180 degrees; “survival Clayton”)
  14 = rotated Gumbel copula (180 degrees; “survival Gumbel”)
  16 = rotated Joe copula (180 degrees; “survival Joe”)
  17 = rotated BB1 copula (180 degrees; “survival BB1”)
  18 = rotated BB6 copula (180 degrees; “survival BB6”)
  19 = rotated BB7 copula (180 degrees; “survival BB7”)
  20 = rotated BB8 copula (180 degrees; “survival BB8”)
  23 = rotated Clayton copula (90 degrees)
  24 = rotated Gumbel copula (90 degrees)
26 = rotated Joe copula (90 degrees)
27 = rotated BB1 copula (90 degrees)
28 = rotated BB6 copula (90 degrees)
29 = rotated BB7 copula (90 degrees)
30 = rotated BB8 copula (90 degrees)
33 = rotated Clayton copula (270 degrees)
34 = rotated Gumbel copula (270 degrees)
36 = rotated Joe copula (270 degrees)
37 = rotated BB1 copula (270 degrees)
38 = rotated BB6 copula (270 degrees)
39 = rotated BB7 copula (270 degrees)
40 = rotated BB8 copula (270 degrees)
104 = Tawn type 1 copula
114 = rotated Tawn type 1 copula (180 degrees)
124 = rotated Tawn type 1 copula (90 degrees)
134 = rotated Tawn type 1 copula (270 degrees)
204 = Tawn type 2 copula
214 = rotated Tawn type 2 copula (180 degrees)
224 = rotated Tawn type 2 copula (90 degrees)
234 = rotated Tawn type 2 copula (270 degrees)

- **Type**: Type of the vine model to be specified:
  - 0 or "RVine" = R-vine (default)
  - 1 or "CVine" = C-vine
- **Selection crit**: Character indicating the criterion for pair-copula selection. Possible choices: selectioncrit = "AIC" (default) or "BIC" (see bicopselect).
- **indeptest**: Logical; whether a hypothesis test for the independence of $u_1$ and $u_2$ is performed before bivariate copula selection (default: indeptest = FALSE; see BicopIndTest). The independence copula is chosen for a (conditional) pair if the null hypothesis of independence cannot be rejected.
- **level**: Numerical; significance level of the independence test (default: level = 0.05).
- **trunclevel**: Integer; level of truncation.
- **progress**: Logical; whether the tree-wise specification progress is printed (default: progress = FALSE).
- **weights**: Numerical; weights for each observation (optional).
- **rotations**: If TRUE, all rotations of the families in familyset are included.

**Details**

R-vine trees are selected using maximum spanning trees with absolute values of pairwise Kendall’s taus as weights, i.e., the following optimization problem is solved for each tree:

$$\max_{edges e_{ij} in spanning tree} \sum |\hat{\tau}_{ij}|,$$

where $\hat{\tau}_{ij}$ denote the pairwise empirical Kendall’s taus and a spanning tree is a tree on all nodes.

The setting of the first tree selection step is always a complete graph. For subsequent trees, the setting depends on the R-vine construction principles, in particular on the proximity condition.
The root nodes of C-vine trees are determined similarly by identifying the node with strongest dependencies to all other nodes. That is we take the node with maximum column sum in the empirical Kendall’s tau matrix.

Note that a possible way to determine the order of the nodes in the D-vine is to identify a shortest Hamiltonian path in terms of weights $1 - |\tau_{ij}|$. This can be established for example using the package TSP. Example code is shown below.

**Value**

An RVineMatrix object with the selected structure (RVM$Matrix) and families (RVM$family) as well as sequentially estimated parameters stored in RVM$par and RVM$par2.

**Author(s)**

Jeffrey Dissmann, Eike Brechmann, Ulf Schepsmeier

**References**


**See Also**

RVineTreePlot, RVineCopSelect

**Examples**

```r
# load data set
data(daxreturns)

# select the R-vine structure, families and parameters
# using only the first 4 variables and the first 750 observations
# we allow for the copula families: Gauss, t, Clayton, Gumbel, Frank and Joe
RVM <- RVineStructureSelect(daxreturns[,1:750,1:4], c(1:6), progress = TRUE)

## Not run:
# specify a C-vine copula model with only Clayton, Gumbel and Frank copulas (time-consuming)
CVM <- RVineStructureSelect(daxreturns, c(3,4,5), "CVine")

## Not run(Not run)

## Not run:
# determine the order of the nodes in a D-vine using the package TSP (time-consuming)
library(TSP)
d <- dim(daxreturns)[2]
M <- 1 - abs(TauMatrix(daxreturns))
hamilton <- insert_dummy(TSP(M), label = "cut")
sol <- solve_TSP(hamilton, method = "repetitive_nn")
order <- cut_tour(sol, "cut")
DVM <- D2RVine(order, family = rep(0,d*(d-1)/2), par = rep(0, d*(d-1)/2))
RVineCopSelect(daxreturns, c(1:6), DVM$Matrix)
```
RVineTreePlot

Visualisation of R-Vine Tree Structure

Description

This function plots one or all trees of a given R-vine copula model.

Usage

RVineTreePlot(data = NULL, RVM, method = "mle", max.df = 30,
max.BB = list(BB1 = c(5, 6), BB6 = c(6, 6), BB7 = c(5, 6), BB8 = c(6, 1)),
tree = "ALL", edge.labels = c("family"), P = NULL)

Arguments

data An N x d data matrix (with uniform margins), default: data = NULL.
RVM An RVineMatrix object including the structure and the pair-copula families and parameters.
method Character indicating the estimation method: either maximum likelihood estimation (method = "mle"; default) or inversion of Kendall’s tau (method = "itau").
max.df Numeric; upper bound for the estimation of the degrees of freedom parameter of the t-copula (default: max.df = 30; for more details see BiCopEst).
max.BB List; upper bounds for the estimation of the two parameters (in absolute values) of the BB1, BB6, BB7 and BB8 copulas (default: max.BB = list(BB1 = c(5, 6), BB6 = c(6, 6), BB7 = c(5, 6), BB8 = c(6, 1))).
tree Number of the tree to be plotted or tree = "ALL" (default) to plot all trees.
edge.labels Vector of edge labels. Possible choices:
FALSE: no edge labels
"family": pair-copula families (default)
"par": pair-copula parameters
"par2": second pair-copula parameters
"theotau": theoretical Kendall’s tau values corresponding to pair-copula families and parameters (see BiCopPar2Tau)
"emptau": empirical Kendall’s tau values (only if data is provided!)
"pair": indices of (conditioned) pair of variables identified by the edges
P A list of matrices with two columns for the x-y-coordinates of the nodes in the plot(s) (optional; default: P = NULL).

Value

A list of matrices P with two columns for the x-y-coordinates of the nodes in the plot(s).
Note

The function computes the positions of the nodes automatically with the Fruchterman-Reingold algorithm (see `plot.igraph` for a detailed description). If one would like to set the positions manually, one has to specify a list of matrices `P` in the argument list. A good starting point may be to run the function `R VineTreePlot` and manipulate the returning matrix `P`.

If data is provided, the parameters of the R-vine copula model are estimated sequentially using `R VineSeqEst/BiCopEst`. Then the edge width is chosen according to the empirical Kendall’s tau values. Otherwise theoretical values are used.

Author(s)

Eike Brechmann

See Also

`BiCopName`

Examples

```r
# define 5-dimensional R-vine tree structure matrix
Matrix <- c(5, 2, 3, 1, 4,
           0, 2, 3, 4, 1,
           0, 0, 3, 4, 1,
           0, 0, 0, 4, 1,
           0, 0, 0, 0, 1)
Matrix <- matrix(Matrix, 5, 5)

# define R-vine pair-copula family matrix
family <- c(0, 1, 3, 4, 4,
           0, 0, 3, 4, 1,
           0, 0, 0, 4, 1,
           0, 0, 0, 0, 3,
           0, 0, 0, 0, 0)
family <- matrix(family, 5, 5)

# define R-vine pair-copula parameter matrix
par <- c(0, 0.2, 0.9, 1.5, 3.9,
        0, 0.1, 1.6, 0.9,
        0, 0.9, 1.9, 0.5,
        0, 0, 0, 4.8,
        0, 0, 0, 0, 0)
par <- matrix(par, 5, 5)

# define second R-vine pair-copula parameter matrix
par2 <- matrix(0, 5, 5)

# define RVineMatrix object
RVM <- RVineMatrix(Matrix = Matrix, family = family,
                   par = par, par2 = par2,
                   names = c("V1", "V2", "V3", "V4", "V5"))

# set random seed for testing
set.seed(123)

# plot all trees with pair-copula families and
# theoretical Kendall's tau values as edge labels
P <- RVineTreePlot(data = NULL, RVM = RVM, tree = "ALL",
edge.labels = c("family","thetau"),
P = NULL)

# manipulate the first matrix of x-y-coordinates
P[[1]][[1],] = P[[1]][1] * 2

# re-set random seed for testing
set.seed(123)
# plot only the first tree with new coordinates
P <- RVineTreePlot(data = NULL, RVM = RVM, tree = 1,
edge.labels = FALSE, P = P)

## RVineVuongTest

### Vuong Test Comparing Two R-Vine Copula Models

#### Description
This function performs a Vuong test between two d-dimensional R-vine copula models as specified by their RVineMatrix objects.

#### Usage
RVineVuongTest(data, RVM1, RVM2)

#### Arguments
- data: An N x d data matrix (with uniform margins).
- RVM1, RVM2: RVineMatrix objects of models 1 and 2.

#### Details
The likelihood-ratio based test proposed by Vuong (1989) can be used for comparing non-nested models. For this let $c_1$ and $c_2$ be two competing vine copulas in terms of their densities and with estimated parameter sets $\hat{\theta}_1$ and $\hat{\theta}_2$. We then compute the standardized sum, $\nu$, of the log differences of their pointwise likelihoods $m_i := \log \left[ \frac{c_1(u_i; \hat{\theta}_1)}{c_2(u_i; \hat{\theta}_2)} \right]$ for observations $u_i \in [0, 1]$, $i = 1, \ldots, N$, i.e.,

$$\text{statistic} := \nu = \frac{\frac{1}{N} \sum_{i=1}^{N} m_i}{\sqrt{\sum_{i=1}^{N} (m_i - \bar{m})^2}}.$$

Vuong (1989) shows that $\nu$ is asymptotically standard normal. According to the null-hypothesis $H_0 : E[m_i] = 0 \forall i = 1, \ldots, N$,
we hence prefer vine model 1 to vine model 2 at level $\alpha$ if

$$\nu > \Phi^{-1} \left(1 - \frac{\alpha}{2}\right),$$

where $\Phi^{-1}$ denotes the inverse of the standard normal distribution function. If $\nu < -\Phi^{-1} \left(1 - \frac{\alpha}{2}\right)$ we choose model 2. If, however, $|\nu| \leq \Phi^{-1} \left(1 - \frac{\alpha}{2}\right)$, no decision among the models is possible.

Like AIC and BIC, the Vuong test statistic may be corrected for the number of parameters used in the models. There are two possible corrections; the Akaike and the Schwarz corrections, which correspond to the penalty terms in the AIC and the BIC, respectively.
surClaytonCopula

Description

These are wrappers to functions from VineCopula-package

Value

statistic, statistic.Akaike, statistic.Schwarz
Test statistics without correction, with Akaike correction and with Schwarz correction.

p.value, p.value.Akaike, p.value.Schwarz
P-values of tests without correction, with Akaike correction and with Schwarz correction.

Author(s)

Jeffrey Dissmann, Eike Brechmann

References


See Also

RVineClarkeTest, RVineAIC, RVineBIC

Examples

## Not run:
# vine structure selection time-consuming (~ 20 sec)

# load data set
data(daxreturns)

# select the R-vine structure, families and parameters
RVM <- RVineStructureSelect(daxreturns[,1:5], c(1:6))

# select the C-vine structure, families and parameters
CVM <- RVineStructureSelect(daxreturns[,1:5], c(1:6), type = "CVine")

# compare the two models based on the data
vuong <- RVineVuongTest(daxreturns[,1:5], RVM, CVM)
uuong$statistic
vuong$statistic.Schwarz
vuong$p.value
vuong$p.value.Schwarz

## End(Not run)
Usage

surClaytonCopula(param)
r90ClaytonCopula(param)
r270ClaytonCopula(param)

Arguments

param A single parameter defining the Copula.

Value

An object of class `surClaytonCopula`, `r90ClaytonCopula` or `r270ClaytonCopula` respectively.

Author(s)

Benedikt Graeler

Examples

```r
library(copula)

persp(surClaytonCopula(1.5), dCopula, zlim = c(0,10))
persp(r90ClaytonCopula(-1.5), dCopula, zlim = c(0,10))
persp(r270ClaytonCopula(-1.5), dCopula, zlim = c(0,10))
```

Description

A class representing rotated versions of the Clayton copula family (survival, 90 and 270 degree rotated).

Objects from the Class

Objects can be created by calls of the form `new("surClaytonCopula", ...)`, `new("r90ClaytonCopula", ...)` and `new("r270ClaytonCopula", ...)` or by the function `surClaytonCopula`, `r90ClaytonCopula` and `r270ClaytonCopula` respectively.

Slots

- family: Object of class "numeric" The family number in `VineCopula-package`
- dimension: Object of class "integer" The dimension of the copula (2).
- parameters: Object of class "numeric" The parameter
- param.names: Object of class "character" name of the parameter
- param.lowbnd: Object of class "numeric" lower bound of the parameter
- param.upbnd: Object of class "numeric" upper bound of the parameter
- fullname: Object of class "character" descriptive name of the family
surGumbelCopula

Extends

Class "copula", directly. Class "Copula", by class "copula", distance 2.

Methods

dduCopula signature(u = "matrix", copula = "surClaytonCopula"): ...
dduCopula signature(u = "numeric", copula = "surClaytonCopula"): ...
ddvCopula signature(u = "matrix", copula = "surClaytonCopula"): ...
ddvCopula signature(u = "numeric", copula = "surClaytonCopula"): ...
dduCopula signature(u = "matrix", copula = "r90ClaytonCopula"): ...
dduCopula signature(u = "numeric", copula = "r90ClaytonCopula"): ...
ddvCopula signature(u = "matrix", copula = "r90ClaytonCopula"): ...
ddvCopula signature(u = "numeric", copula = "r90ClaytonCopula"): ...
dduCopula signature(u = "matrix", copula = "r270ClaytonCopula"): ...
dduCopula signature(u = "numeric", copula = "r270ClaytonCopula"): ...
ddvCopula signature(u = "matrix", copula = "r270ClaytonCopula"): ...
ddvCopula signature(u = "numeric", copula = "r270ClaytonCopula"): ...

Author(s)

Benedikt Graeler

See Also

VineCopula-package

Examples

library(copula)

persp(surClaytonCopula(.5), dCopula, zlim=c(0,1))
persp(r90ClaytonCopula(-.5), dCopula, zlim=c(0,1))
persp(r270ClaytonCopula(-.5), dCopula, zlim=c(0,1))

surGumbelCopula  Survival and Rotated Gumbel Copulas

Description

These are wrappers to functions from VineCopula-package

Usage

surGumbelCopula(param)
r90GumbelCopula(param)
r270GumbelCopula(param)
Arguments

param A single parameter defining the Copula.

Value

An object of class `surGumbelCopula`, `r90GumbelCopula` or `r270GumbelCopula` respectively.

Author(s)

Benedikt Graeler

Examples

```r
library(copula)
persp(surGumbelCopula(1.5), dCopula, zlim = c(0,10))
persp(r90GumbelCopula(1.5), dCopula, zlim = c(0,10))
persp(r270GumbelCopula(1.5), dCopula, zlim = c(0,10))
```

Description

A class representing rotated versions of the Gumbel copula family (survival, 90 and 270 degree rotated).

Objects from the Class

Objects can be created by calls of the form `new("surGumbelCopula", ...)`, `new("r90GumbelCopula", ...)` and `new("r270GumbelCopula", ...)` or by the function `surGumbelCopula`, `r90GumbelCopula` and `r270GumbelCopula` respectively.

Slots

- `family`: Object of class "numeric" The family number in `VineCopula-package`
- `dimension`: Object of class "integer" The dimension of the copula (2).
- `parameters`: Object of class "numeric" The parameter
- `param.names`: Object of class "character" name of the parameter
- `param.lowbnd`: Object of class "numeric" lower bound of the parameter
- `param.upbnd`: Object of class "numeric" upper bound of the parameter
- `fullname`: Object of class "character" descriptive name of the family

Extends

Class "copula", directly. Class "Copula", by class "copula", distance 2.
**Methods**

```r
dduCopula signature(u = "matrix", copula = "surGumbelCopula"): ...
ddvCopula signature(u = "numeric", copula = "surGumbelCopula"): ...
ddvCopula signature(u = "matrix", copula = "surGumbelCopula"): ...
ddvCopula signature(u = "numeric", copula = "surGumbelCopula"): ...
ddvCopula signature(u = "matrix", copula = "r90GumbelCopula"): ...
ddvCopula signature(u = "numeric", copula = "r90GumbelCopula"): ...
ddvCopula signature(u = "matrix", copula = "r270GumbelCopula"): ...
ddvCopula signature(u = "numeric", copula = "r270GumbelCopula"): ...
ddvCopula signature(u = "matrix", copula = "rRWPgumbelCopula"): ...
ddvCopula signature(u = "numeric", copula = "rRWPgumbelCopula"): ...
```

**Author(s)**

Benedikt Graeler

**See Also**

`vinecopula-package`

**Examples**

```r
library(copula)
persp(surGumbelCopula(1.5), dCopula, zlim=c(0,10))
persp(r90GumbelCopula(-1.5), dCopula, zlim=c(0,10))
persp(r270GumbelCopula(-1.5), dCopula, zlim=c(0,10))
```

**Description**

This function computes the empirical Kendall’s tau using the algorithm by Knight (1966).

**Usage**

```r
TauMatrix(data, weights = NA)
```

**Arguments**

- `data`: An N x d data matrix.
- `weights`: Numerical; weights for each observation (optional).

**Value**

Matrix of the empirical Kendall’s taus.
Author(s)
Ulf Schepsmeier

References

See Also
BiCopTau2Par, BiCopPar2Tau, BiCopEst

Examples
```r
data(daxreturns)
Data <- as.matrix(daxreturns)

# compute the empirical Kendall's taus
TauMatrix(Data)
```

**tawnT1Copula**
*Constructor of the Tawn Type 1 Family and Rotated Versions thereof*

Description
Constructs an object of the tawnT1Copula (survival sur, 90 degree rotated r90 and 270 degree rotated r270) family for given parameters.

Usage
```r
tawnT1Copula(param = c(2, 0.5))
surtawnt1copula(param = c(2, 0.5))
r9Ptawnt1copula(param = c(-2, 0.5))
r270Tawnt1Copula(param = c(-2, 0.5))
```

Arguments
- `param` The parameter `param` defines the copula through `param1` and `param2`.

Value
One of the Tawn type 1 copula classes (tawnT1Copula, surTawnT1Copula, r90TawnT1Copula, r270TawnT1Copula).

Author(s)
Benedikt Graeler

See Also
tawnT2Copula and the package VineCopula-package for implementation details.
**tawnT1Copula-class**

**Examples**

```r
library(copula)

persp(tawnT1Copula(), dCopula, zlim = c(0,10))
persp(surTawnT1Copula(), dCopula, zlim = c(0,10))
persp(r90TawnT1Copula(), dCopula, zlim = c(0,10))
persp(r270TawnT1Copula(), dCopula, zlim = c(0,10))
```

---

**Description**

S4-class representation of the Tawn Copula family of type 1 and rotated versions there of.

**Objects from the Class**

Objects can be created by calls of the form `new("tawnT1Copula", ...)`, or through the explicit constructors `tawnT1Copula`, `surTawnT1Copula`, `r90TawnT1Copula` and `r270TawnT1Copula` respectively.

**Slots**

- `family`: Object of class "numeric" providing the unique number in VineCopula.
- `dimension`: Object of class "integer" and fixed to 2L.
- `parameters`: Object of class "numeric" representing the two parameters.
- `param.names`: Object of class "character" providing the names of the parameters.
- `param.lwbnd`: Object of class "numeric" providing the lower bounds of the parameters.
- `param.upbnd`: Object of class "numeric" providing the upper bounds of the parameters.
- `fullname`: Object of class "character" providing a textual summary of the copula class.

**Extends**

Class "copula", directly. Class "Copula", by class "copula", distance 2.

**Methods**

- `dCopula` signature(u = "matrix", copula = "tawnT1Copula"): ...
- `dCopula` signature(u = "numeric", copula = "tawnT1Copula"): ...
- `dduCopula` signature(u = "matrix", copula = "tawnT1Copula"): ...
- `dduCopula` signature(u = "numeric", copula = "tawnT1Copula"): ...
- `ddvCopula` signature(u = "matrix", copula = "tawnT1Copula"): ...
- `ddvCopula` signature(u = "numeric", copula = "tawnT1Copula"): ...
- `pCopula` signature(u = "matrix", copula = "tawnT1Copula"): ...
- `pCopula` signature(u = "numeric", copula = "tawnT1Copula"): ...
- `rCopula` signature(n = "numeric", copula = "tawnT1Copula"): ...
- `taillIndex` signature(copula = "tawnT1Copula"): ...
- `tau` signature(copula = "tawnT1Copula"): ...
Author(s)
Benedikt Graeler

See Also
tawnT2Copula and the package VineCopula-package for implementation details.

Examples
showClass("tawnT1Copula")

---

**tawnT2Copula**

*Constructor of the Tawn Type 2 Family and Rotated Versions thereof*

Description
Constructs an object of the tawnT2Copula (survival sur, 90 degree rotated r90 and 270 degree rotated r270) family for given parameters.

Usage
```
tawnT2Copula(param = c(2, 0.5))
surTawnT2Copula(param = c(2, 0.5))
r90TawnT2Copula(param = c(-2, 0.5))
r270TawnT2Copula(param = c(-2, 0.5))
```

Arguments
- **param**
  
The parameter param defines the copula through param1 and param2.

Value
One of the Tawn type 2 copula classes (tawnT2Copula, surTawnT2Copula, r90TawnT2Copula, r270TawnT2Copula).

Author(s)
Benedikt Graeler

See Also
tawnT2Copula and the package VineCopula-package for implementation details.

Examples
```
library(copula)
persp(tawnT2Copula(), dCopula, zlim = c(0,10))
persp(surTawnT2Copula(), dCopula, zlim = c(0,10))
persp(r90TawnT2Copula(), dCopula, zlim = c(0,10))
persp(r270TawnT2Copula(), dCopula, zlim = c(0,10))
```
**tawnT2Copula-class**

**Description**

S4-class representation of the Tawn Copula family of type 2 and rotated versions there of.

**Objects from the Class**

Objects can be created by calls of the form `new("tawnT2Copula", ...), or through the explicit constructors `tawnT2Copula, surTawnT2Copula, r90TawnT2Copula` and `r270TawnT2Copula` respectively.

**Slots**

- **family**: Object of class "numeric" providing the unique number in VineCopula.
- **dimension**: Object of class "integer" and fixed to 2L.
- **parameters**: Object of class "numeric" representing the two parameters.
- **param.names**: Object of class "character" providing the names of the parameters.
- **param.lowbnd**: Object of class "numeric" providing the lower bounds of the parameters.
- **param.upbnd**: Object of class "numeric" providing the upper bounds of the parameters.
- **fullname**: Object of class "character" providing a textual summary of the copula class.

**Extends**

Class "copula", directly. Class "Copula", by class "copula", distance 2.

**Methods**

- `dCopula` signature(`u = "matrix", copula = "tawnT2Copula"):
- `dCopula` signature(`u = "numeric", copula = "tawnT2Copula"):
- `dduCopula` signature(`u = "matrix", copula = "tawnT2Copula"):
- `dduCopula` signature(`u = "numeric", copula = "tawnT2Copula"):
- `ddvCopula` signature(`u = "matrix", copula = "tawnT2Copula"):
- `ddvCopula` signature(`u = "numeric", copula = "tawnT2Copula"):
- `pCopula` signature(`u = "matrix", copula = "tawnT2Copula"):
- `pCopula` signature(`u = "numeric", copula = "tawnT2Copula"):
- `rCopula` signature(`n = "numeric", copula = "tawnT2Copula"):
- `tailIndex` signature(copula = "tawnT2Copula"):
- `tau` signature(copula = "tawnT2Copula"):

**Author(s)**

Benedikt Graeler
See Also

tawnT1Copula and the package VineCopula-package for implementation details.

Examples

showClass("tawnT2Copula")

---

vineCopula

Constructor of the Class vineCopula.

Description

Constructs an instance of the vineCopula class.

Usage

vineCopula(RVM, type = "CVine")

Arguments

- **RVM**: An object of class RVineMatrix generated from RVineMatrix in the package VineCopula-package or an integer (e.g. 4L) defining the dimension (an independent C-vine of this dimension will be constructed).
- **type**: A predefined type if only the dimension is provided and ignored otherwise, the default is a canonical vine

Value

An instance of the vineCopula class.

Author(s)

Benedikt Graeler

References


Examples

# a C-vine of independent copulas
vine <- vineCopula(4L, "CVine")

library(copula)
library(lattice)

cloud(V1 ~ V2 + V3, as.data.frame(rCopula(500, vine)))
Description

A class representing vine copulas in an object oriented implementation. Many functions go back to the package VineCopula-package.

Objects from the Class

Objects can be created by calls of the form new("vineCopula", ...) or through the function vineCopula.

Slots

- rvm: An RVineMatrix object from RVineMatrix describing the R-Vine structure.
- copulas: Object of class "list" holding all copulas.
- dimension: Object of class "integer"; the vines dimension.
- parameters: Object of class "numeric": empty
- param.names: Object of class "character": empty
- param.lowbnd: Object of class "numeric": empty
- param.upbnd: Object of class "numeric": empty
- fullname: Object of class "character" providing a descriptive name of the vine copula.

Extends

Class "copula", directly. Class "Copula", by class "copula", distance 2.

Methods

No additional methods yet, but uses e.g. dCopula, pCopula, rCopula and rCopula as any other copula. Via the method argument in fitCopula, control over the fit of the RVine can be taken via entries StructureSelect, indeptest and familyset. See RVineCopSelect and RVineStructureSelect for further details on the underlying functions. Missing entries are treated as default values, i.e. StructureSelect=FALSE, indeptest=FALSE and familyset=NA.

Author(s)

Benedikt Graeler

References


See Also

RVineMatrix from package VineCopula-package

Examples

showClass("vineCopula")
Index

- Topic **textasciitildekwd1**
  - copulaFromFamilyIndex, 67
- Topic **textasciitildekwd2**
  - copulaFromFamilyIndex, 67
- Topic **classes**
  - BB1Copula-class, 8
  - BB6Copula-class, 10
  - BB7Copula-class, 12
  - BB8Copula-class, 14
  - JoeBiCopula-class, 72
  - surClaytonCopula-class, 120
  - surGumbelCopula-class, 122
  - tawnTICopula-class, 125
  - tawnTZCopula-class, 127
  - vineCopula-class, 129
- Topic **conditional probabilities**
  - dduCopula, 70
- Topic **copula**
  - BB1Copula, 7
  - surClaytonCopula, 119
  - surGumbelCopula, 121
  - tawnTICopula, 124
  - tawnTZCopula, 126
- Topic **distribution**
  - BB1Copula, 7
  - tawnTICopula, 124
  - tawnTZCopula, 126
  - vineCopula, 128
- Topic **multivariate**
  - vineCopula, 128
- Topic **partial correlation**
  - RVineCor2pcor, 84
- Topic **partial derivative**
  - dduCopula, 70
- Topic **plot**
  - plot.BiCop, 75
- Topic **vine**
  - RVineCor2pcor, 84
  - RVineMatrixNormalize, 99

as.copuladata, 6, 74
BB1Copula, 7, 7, 8, 11, 13, 15, 71, 73
BB1Copula-class, 8
BB6Copula, 7, 9, 9, 10, 11, 13, 15, 71, 73
BB6Copula-class, 10
BB7Copula, 7, 9, 11, 12, 13, 15, 71, 73
BB7Copula-class, 12
BB8Copula, 7, 9, 11, 13, 14, 71, 73
BB8Copula-class, 14
BetaMatrix, 15
BiCop, 16, 17, 19, 22, 24, 26, 27, 32–36, 40, 41, 47, 52, 53, 55, 57, 58, 60, 76
BiCopCDF, 17, 32, 47, 55, 60, 102
BiCopChPlot, 19, 38, 41, 44
BiCopDeriv, 21, 24, 30, 36, 90, 92, 112
BiCopDeriv2, 22, 23, 30, 34, 36, 90, 92, 112
BiCopEst, 17, 25, 100, 108, 109, 116, 117, 124
BiCopGofTest, 28, 37, 38, 64, 88
BiCopHfunc, 17, 19, 31, 55, 94, 106, 109
BiCopHfuncDeriv, 22, 24, 33, 34, 36, 90, 92, 112
BiCopHfuncDeriv2, 34, 90, 92, 112
BiCopIndTest, 30, 36, 57, 58, 81, 114
BiCopKPlot, 21, 37, 41, 44
BiCopLambda, 21, 38, 39, 44, 75
BiCopMetaContour, 21, 38, 41, 41, 73, 74
BiCopName, 44, 117
BiCopPar2Beta, 15, 46, 102, 103
BiCopPar2TailDep, 48
BiCopPar2Tau, 26, 27, 37, 50, 50, 61, 104, 116, 124
BiCopPDF, 17, 19, 32, 53, 60
BiCopSelect, 17, 27, 37, 55, 64, 81, 83, 113, 114
BiCopSim, 17, 19, 55, 110
BiCopTau2Par, 26, 27, 37, 53, 60, 124
BiCopVuongClarke, 30, 62
C2RVine, 65, 69, 97
contour, 76
Copula, 8, 10, 12, 14, 72, 121, 122, 125, 127, 129
copula, 8, 10, 12, 14, 46, 72, 121, 122, 125, 127, 129
copulaFromFamilyIndex, 67
D2RVine, 66, 67, 97
INDEX

131

daxreturns, 69
dCopula, 70, 129
dduCopula, 70

dduCopula, matrix, BB1Copula-method
(BB1Copula-class), 8

dduCopula, matrix, BB6Copula-method
(BB6Copula-class), 10

dduCopula, matrix, BB7Copula-method
(BB7Copula-class), 12

dduCopula, matrix, BB8Copula-method
(BB8Copula-class), 14

dduCopula, matrix, JoeBiCopula-method
(JoeBiCopula-class), 72

dduCopula, matrix, r270BB1Copula-method
(BB1Copula-class), 8

dduCopula, matrix, r270BB6Copula-method
(BB6Copula-class), 10

dduCopula, matrix, r270BB7Copula-method
(BB7Copula-class), 12

dduCopula, matrix, r270BB8Copula-method
(BB8Copula-class), 14

dduCopula, matrix, r270ClaytonCopula-method
(surClaytonCopula-class), 120

dduCopula, matrix, r270GumbelCopula-method
(surGumbelCopula-class), 122

dduCopula, matrix, r270JoeBiCopula-method
(JoeBiCopula-class), 72

dduCopula, matrix, r270Tawn1Copula-method
(tawn1Copula-class), 125

dduCopula, matrix, r270Tawn2Copula-method
(tawn2Copula-class), 127

dduCopula, matrix, r90BB1Copula-method
(BB1Copula-class), 8

dduCopula, matrix, r90BB6Copula-method
(BB6Copula-class), 10

dduCopula, matrix, r90BB7Copula-method
(BB7Copula-class), 12

dduCopula, matrix, r90BB8Copula-method
(BB8Copula-class), 14

dduCopula, matrix, r90ClaytonCopula-method
(surClaytonCopula-class), 120

dduCopula, matrix, r90GumbelCopula-method
(surGumbelCopula-class), 122

dduCopula, matrix, r90JoeBiCopula-method
(JoeBiCopula-class), 72

dduCopula, matrix, r90Tawn1Copula-method
(tawn1Copula-class), 125

dduCopula, matrix, r90Tawn2Copula-method
(tawn2Copula-class), 127

dduCopula, numeric, BB1Copula-method
(BB1Copula-class), 8

dduCopula, numeric, BB6Copula-method
(BB6Copula-class), 10
ddvCopula, numeric, BB1Copula-method (BB1Copula-class), 8
ddvCopula, numeric, BB6Copula-method (BB6Copula-class), 10
ddvCopula, numeric, BB7Copula-method (BB7Copula-class), 12
ddvCopula, numeric, BB8Copula-method (BB8Copula-class), 14
ddvCopula, numeric, JoeBiCopula-method (JoeBiCopula-class), 72
ddvCopula, numeric, r270BB1Copula-method (BB1Copula-class), 8
ddvCopula, numeric, r270BB6Copula-method (BB6Copula-class), 10
ddvCopula, numeric, r270BB7Copula-method (BB7Copula-class), 12
ddvCopula, numeric, r270BB8Copula-method (BB8Copula-class), 14
ddvCopula, numeric, r270ClaytonCopula-method (surClaytonCopula-class), 120
ddvCopula, numeric, r270GumbelCopula-method (surGumbelCopula-class), 122
ddvCopula, numeric, r270JoeBiCopula-method (JoeBiCopula-class), 72
ddvCopula, numeric, r270TawnT1Copula-method (tawnT1Copula-class), 125
ddvCopula, numeric, r270TawnT2Copula-method (tawnT2Copula-class), 127
ddvCopula, numeric, r90BB1Copula-method (BB1Copula-class), 8
ddvCopula, numeric, r90BB6Copula-method (BB6Copula-class), 10
ddvCopula, numeric, r90BB7Copula-method (BB7Copula-class), 12
ddvCopula, numeric, r90BB8Copula-method (BB8Copula-class), 14
ddvCopula, numeric, r90ClaytonCopula-method (surClaytonCopula-class), 120
ddvCopula, numeric, r90GumbelCopula-method (surGumbelCopula-class), 122
ddvCopula, numeric, r90JoeBiCopula-method (JoeBiCopula-class), 72
ddvCopula, numeric, r90TawnT1Copula-method (tawnT1Copula-class), 125
ddvCopula, numeric, r90TawnT2Copula-method (tawnT2Copula-class), 127
ddvCopula, numeric, surBB1Copula-method (BB1Copula-class), 8
ddvCopula, numeric, surBB6Copula-method (BB6Copula-class), 10
ddvCopula, numeric, surBB7Copula-method (BB7Copula-class), 12
ddvCopula, numeric, surBB8Copula-method (BB8Copula-class), 14
ddvCopula, numeric, surClaytonCopula-method (surClaytonCopula-class), 120
ddvCopula, numeric, surGumbelCopula-method (surGumbelCopula-class), 122
ddvCopula, numeric, surJoeBiCopula-method (JoeBiCopula-class), 72
ddvCopula, numeric, surTawnT1Copula-method (tawnT1Copula-class), 125
ddvCopula, numeric, surTawnT2Copula-method (tawnT2Copula-class), 127
ddvCopula, numeric, tawnT1Copula-method (tawnT1Copula-class), 125
ddvCopula, numeric, tawnT2Copula-method (tawnT2Copula-class), 127
dexp, 43, 74
dgamma, 43, 74
dt, 43, 74
fitCopula, twoParamBiCop-method (BB8Copula-class), 14
fitCopula, vineCopula-method (vineCopula-class), 129
getKendallDistr, BB1Copula-method (BB1Copula-class), 8
getKendallDistr, BB6Copula-method (BB6Copula-class), 10
getKendallDistr, BB7Copula-method (BB7Copula-class), 12
getKendallDistr, BB8Copula-method (BB8Copula-class), 14
getKendallDistr, JoeBiCopula-method (JoeBiCopula-class), 72
joeBiCopula, 71, 71, 72
joeBiCopula-class, 72
joeCopula, 7, 9, 11, 13, 15, 71, 72
kendallDistribution, BB1Copula-method (BB1Copula-class), 8
kendallDistribution, BB6Copula-method (BB6Copula-class), 10
kendallDistribution, BB7Copula-method (BB7Copula-class), 12
kendallDistribution, BB8Copula-method (BB8Copula-class), 14
kendallDistribution, JoeBiCopula-method (JoeBiCopula-class), 72
logical, 76
NULL, 110
optim, 100, 101
pairs, 74
pairs.copuladata, 6, 73
par, 73
pCopula, 70, 129
plot.Bicop, 17, 75
plot.igraph, 117
pobs, 6, 76, 77
r270B81copula, 7, 8
r270B81copula(B81Copula), 7
r270B81copula-class(B81Copula-class), 8
r270B86Copula, 9, 10
r270B86Copula(B86Copula), 9
r270B86copula-class(B86Copula-class), 10
r270B87Copula, 11, 12
r270B87copula(B87Copula), 11
r270B87copula-class(B87Copula-class), 12
r270BB8Copula, 13, 14
r270BB8Copula(B88Copula), 13
r270BB8Copula-class(B88Copula-class), 14
r270ClaytonCopula, 120
r270ClaytonCopula(surClaytonCopula), 119
r270ClaytonCopula-class
(surClaytonCopula-class), 120
r270GumbelCopula, 122
r270GumbelCopula(surGumbelCopula), 121
r270GumbelCopula-class
(surGumbelCopula-class), 122
r270JoeBiCopula, 71, 72
r270JoeBiCopula(joeBiCopula), 71
r270JoeBiCopula-class
(joeBiCopula-class), 72
r270TawnT1Copula, 124, 125
r270TawnT1Copula(tawnT1Copula), 124
r270TawnT1copula-class
(tawnT1copula-class), 125
r270TawnT2Copula, 126, 127
r270TawnT2Copula(tawnT2Copula), 126
r270TawnT2copula-class
(tawnT2copula-class), 127
r90BB7Copula, 7, 8
r90BB7copula(BB7Copula), 7
r90BB7copula-class(BB7Copula-class), 8
r90BB8Copula, 9, 10
r90BB8Copula(BB8Copula), 9
r90BB8Copula-class(BB8Copula-class), 10
r90BB7Copula, 11, 12
r90BB7Copula-class(BB7Copula-class), 11
r90BB8Copula, 13, 14
r90BB8Copula(BB8Copula), 13
r90BB8Copula-class(BB8Copula-class), 14
r90ClaytonCopula, 120
r90ClaytonCopula(surClaytonCopula), 119
r90ClaytonCopula-class
(surClaytonCopula-class), 120
r90GumbelCopula, 122
r90GumbelCopula(surGumbelCopula), 121
r90GumbelCopula-class
(surGumbelCopula-class), 122
r90JoeBiCopula, 71, 72
r90JoeBiCopula(joeBiCopula), 71
r90JoeBiCopula-class
(joeBiCopula-class), 72
r90TawnT1Copula, 124, 125
r90TawnT1Copula(tawnT1Copula), 124
r90TawnT1copula-class
(tawnT1copula-class), 125
r90TawnT2Copula, 126, 127
r90TawnT2Copula(tawnT2Copula), 126
r90TawnT2copula-class
(tawnT2copula-class), 127
rank, 76
rcopula, 129
RVineAIC, 80, 94, 106, 119
RVineAIC(RVineAIC/BIC), 77
RVineAIC/BIC, 77
RVineBIC, 80, 94, 106, 119
RVineBIC(RVineAIC/BIC), 77
RVineClarkeTest, 63, 64, 78, 79, 119
RVineCopSelect, 37, 58, 81, 114, 115, 129
RVineCor2pcor, 84
RVineGofTest, 85, 107
RVineGrad, 22, 24, 34, 36, 85, 88, 92, 100, 101, 112
RVineHessian, 22, 24, 34, 36, 85, 90, 91, 100, 101, 112
RVineLogLik, 32, 78, 93, 105, 109
RVineMatrixCheck, 96, 97, 97
RVineMatrixNormalize, 99
RVineMLE, 90, 92, 94, 97, 100, 106, 109
RVinePar2Beta, 15, 102
RVinePar2Tau, 104
RVinePcorr2cor(RVineCor2pcor), 84
RVinePDF, 105
RVinePIT, 87, 88, 106
INDEX

RVineSeqEst, 27, 32, 83, 100, 101, 108, 117
RVineSim, 60, 97, 110
RVineStdError, 111
RVineStructureSelect, 37, 58, 69, 81, 83, 101, 113, 129
RVineTreePlot, 45, 115, 116, 117
RVineVuongTest, 63, 64, 78, 80, 118

surBB1Copula, 7, 8
surBB1Copula (BB1Copula), 7
surBB1Copula-class (BB1Copula-class), 8
surBB6Copula, 9, 10
surBB6Copula (BB6Copula), 9
surBB6Copula-class (BB6Copula-class), 10
surBB7Copula, 11, 12
surBB7Copula (BB7Copula), 11
surBB7Copula-class (BB7Copula-class), 12
surBB8Copula, 13, 14
surBB8Copula (BB8Copula), 13
surBB8Copula-class (BB8Copula-class), 14
surClaytonCopula, 119, 120
surClaytonCopula-class, 120
surGumbelCopula, 121, 122
surGumbelCopula-class, 122
surJoeBiCopula, 71, 72
surJoeBiCopula (JoeBiCopula), 71
surJoeBiCopula-class (JoeBiCopula-class), 72
surTawnT1Copula, 124, 125
surTawnT1Copula (tawnT1Copula), 124
surTawnT1Copula-class (tawnT1Copula-class), 125
surTawnT2Copula, 126, 127
surTawnT2Copula (tawnT2Copula), 126
surTawnT2Copula-class (tawnT2Copula-class), 127

TauMatrix, 15, 123
tawnT1Copula, 124, 124, 125, 128
tawnT1Copula-class, 125
tawnT2Copula, 124, 126, 126, 127
tawnT2Copula-class, 127

VineCopula (VineCopula-package), 3
vineCopula, 128, 128, 129
vineCopula-class, 129
VineCopula-package, 3

wireframe, 76