Package ‘VineCopula’

February 11, 2018

Type Package

Title Statistical Inference of Vine Copulas

Version 2.1.4

Description Provides tools for the statistical analysis of vine copula models. The package includes tools for parameter estimation, model selection, simulation, goodness-of-fit tests, and visualization. Tools for estimation, selection and exploratory data analysis of bivariate copula models are also provided.

Depends R (>= 3.1.0)

Imports graphics, grDevices, stats, utils, MASS, mvtnorm, network, methods, copula (>= 0.999-16), kdecopula (>= 0.8.0), ADGofTest, lattice, doParallel, parallel, foreach

Suggests CDVine, TSP, shiny

License GPL (>= 2)

LazyLoad yes

BugReports https://github.com/tnagler/VineCopula/issues

URL https://github.com/tnagler/VineCopula

RoxygenNote 6.0.1

NeedsCompilation yes

Author Ulf Schepsmeier [aut],
Jakob Stoeber [aut],
Eike Christian Brechmann [aut],
Benedikt Graeler [aut],
Thomas Nagler [aut, cre],
Tobias Erhardt [aut],
Carlos Almeida [ctb],
Aleksey Min [ctb, ths],
Claudia Czado [ctb, ths],
Mathias Hofmann [ctb],
Matthias Killiches [ctb],
Harry Joe [ctb],
Thibault Vatter [ctb]

Maintainer Thomas Nagler <thomas.nagler@tum.de>

Repository CRAN

Date/Publication 2018-02-11 17:33:36 UTC

R topics documented:

`VineCopula-package` *Statistical Inference of Vine Copulas*

Description

Vine copulas are a flexible class of dependence models consisting of bivariate building blocks (see e.g., Aas et al., 2009). This package is primarily made for the statistical analysis of vine copula models. The package includes tools for parameter estimation, model selection, simulation, goodness-of-fit tests, and visualization. Tools for estimation, selection and exploratory data analysis of bivariate copula models are also provided.

Details

Package:	VineCopula
Type:	Package
Version:	2.0.6
Date:	2016-09-29
License:	GPL (>=2)
Depends:	R ($\geq$ 2.11.0)
Imports:	graphics, grDevices, stats, utils, MASS, mvtnorm, network, methods, copula (>= 0.999-15), kdecopula (>= 0.6.0
Suggests:	CDVine, TSP, shiny
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).

Author(s)

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

References

Aas, K., C. Czado, A. Frigessi, and H. Bakken (2009). Pair-copula constructions of multiple dependence. Insurance: Mathematics and Economics 44 (2), 182-198.

Bedford, T. and R. M. Cooke (2001). Probability density decomposition for conditionally dependent random variables modeled by vines. Annals of Mathematics and Artificial intelligence 32, 245-268.

Bedford, T. and R. M. Cooke (2002). Vines - a new graphical model for dependent random variables. Annals of Statistics 30, 1031-1068.

Brechmann, E. C., C. Czado, and K. Aas (2012). Truncated regular vines in high dimensions with applications to financial data. Canadian Journal of Statistics 40 (1), 68-85.

Brechmann, E. C. and C. Czado (2011). Risk management with high-dimensional vine copulas: An analysis of the Euro Stoxx 50. Statistics & Risk Modeling, 30 (4), 307-342.

Brechmann, E. C. and U. Schepsmeier (2013). Modeling Dependence with C- and D-Vine Copulas: The R Package CDVine. Journal of Statistical Software, 52 (3), 1-27. http://www.jstatsoft.org/v52/i03/.

Czado, C., U. Schepsmeier, and A. Min (2012). Maximum likelihood estimation of mixed C-vines with application to exchange rates. Statistical Modelling, 12(3), 229-255.

Dissmann, J. F., E. C. Brechmann, C. Czado, and D. Kurowicka (2013). Selecting and estimating regular vine copulae and application to financial returns. Computational Statistics & Data Analysis, 59 (1), 52-69.

Eschenburg, P. (2013). Properties of extreme-value copulas Diploma thesis, Technische Universitaet Muenchen http://mediatum.ub.tum.de/node?id=1145695

Joe, H. (1996). Families of m-variate distributions with given margins and m(m-1)/2 bivariate dependence parameters. In L. Rueschendorf, B. Schweizer, and M. D. Taylor (Eds.), Distributions with fixed marginals and related topics, pp. 120-141. Hayward: Institute of Mathematical Statistics.

Joe, H. (1997). Multivariate Models and Dependence Concepts. London: Chapman and Hall.

Knight, W. R. (1966). A computer method for calculating Kendall's tau with ungrouped data. Journal of the American Statistical Association 61 (314), 436-439.

Kurowicka, D. and R. M. Cooke (2006). Uncertainty Analysis with High Dimensional Dependence Modelling. Chichester: John Wiley.

Kurowicka, D. and H. Joe (Eds.) (2011). Dependence Modeling: Vine Copula Handbook. Singapore: World Scientific Publishing Co.

Nelsen, R. (2006). An introduction to copulas. Springer

Schepsmeier, U. and J. Stoeber (2014). Derivatives and Fisher information of bivariate copulas. Statistical Papers, 55 (2), 525-542.
http://link.springer.com/article/10.1007/s00362-013-0498-x.

Schepsmeier, U. (2013) A goodness-of-fit test for regular vine copula models. Preprint http://arxiv.org/abs/1306.0818

Schepsmeier, U. (2015) Efficient information based goodness-of-fit tests for vine copula models with fixed margins. Journal of Multivariate Analysis 138, 34-52.

Stoeber, J. and U. Schepsmeier (2013). Estimating standard errors in regular vine copula models. Computational Statistics, 28 (6), 2679-2707
http://link.springer.com/article/10.1007/s00180-013-0423-8#.

White, H. (1982) Maximum likelihood estimation of misspecified models, Econometrica, 50, 1-26.

`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 = c(1, 1))
```

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

Joe, H., (1997). Multivariate Models and Dependence Concepts. Monogra. Stat. Appl. Probab. 73, London: Chapman and Hall.

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`.

Author(s)

Benedikt Graeler

References

Joe, H., (1997). Multivariate Models and Dependence Concepts. Monogra. Stat. Appl. Probab. 73, London: Chapman and Hall.

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 `r90` and 270 degree rotated `r270`) family for given parameters.

Usage

```
BB6Copula(param = c(1, 1))
```

Arguments

`param`	The parameter `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

Joe, H., (1997). Multivariate Models and Dependence Concepts. Monogra. Stat. Appl. Probab. 73, London: Chapman and Hall.

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` *Classes* `"BB6Copula"`, `"surBB6Copula"`, `"r90BB6Copula"` *and* `"r270BB6Copula"`

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`.

Author(s)

Benedikt Graeler

References

Joe, H., (1997). Multivariate Models and Dependence Concepts. Monogra. Stat. Appl. Probab. 73, London: Chapman and Hall.

See Also

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

Examples

```
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 = c(1, 1))
```

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

Joe, H., (1997). Multivariate Models and Dependence Concepts. Monogra. Stat. Appl. Probab. 73, London: Chapman and Hall.

See Also

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

Examples

```
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`.

Author(s)

Benedikt Graeler

References

Joe, H., (1997). Multivariate Models and Dependence Concepts. Monogra. Stat. Appl. Probab. 73, London: Chapman and Hall.

See Also

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

Examples

```
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

```
BB8Copula(param = c(1, 1))
```

Arguments

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

Value

One of the respective BB8 copula classes (BB8Copula, surBB8Copula, r90BB8Copula, r270BB8Copula).

Author(s)

Benedikt Graeler

References

Joe, H., (1997). Multivariate Models and Dependence Concepts. Monogra. Stat. Appl. Probab. 73, London: Chapman and Hall.

See Also

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

Examples

```
library(copula)

persp(BB8Copula(c(2,0.9)), dCopula, zlim = c(0,10))
persp(surBB8Copula(c(2,0.9)), dCopula, zlim = c(0,10))
persp(r90BB8Copula(c(-2,-0.9)), dCopula, zlim = c(0,10))
persp(r270BB8Copula(c(-2,-0.9)), dCopula, zlim = c(0,10))
```

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.

Author(s)

Benedikt Graeler

References

Joe, H., (1997). Multivariate Models and Dependence Concepts. Monogra. Stat. Appl. Probab. 73, London: Chapman and Hall.

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

Blomqvist, N. (1950). On a measure of dependence between two random variables. The Annals of Mathematical Statistics, 21(4), 593-600.

Nelsen, R. (2006). An introduction to copulas. Springer

See Also

TauMatrix, BiCopPar2Beta, RVinePar2Beta

Examples

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

# compute the empirical Blomqvist's betas
BetaMatrix(data)
```

BiCop *Constructing BiCop-objects*

Description

This function creates an object of class `BiCop` and checks for family/parameter consistency.

Usage

```
BiCop(family, par, par2 = 0, tau = NULL, check.pars = TRUE)
```

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 is `par2 = 0`). `par2` should be an positive integer for the Students's t copula `family = 2`.

`tau` numeric; value of Kendall's tau; has to lie in the interval (-1, 1). Can only be used with one-parameter families and the t copula. If `tau` is provided, `par` will be ignored.

`check.pars` logical; default is `TRUE`; if `FALSE`, checks for family/parameter-consistency are ommited (should only be used with care).

Value

An object of class `BiCop`. It is a list containing information about the bivariate copula. Its components are:

`family, par, par2`
copula family number and parameter(s),

`npars` number of parameters,

`familyname` name of the copula family,

`tau` Kendall's tau,

`beta` Blomqvist's beta,

`taildep` lower and upper tail dependence coefficients,

`call` the call that created the object.

Objects of this class are also returned by the `BiCopEst` and `BiCopSelect` functions. In this case, further information about the fit is added.

Note

For a comprehensive summary of the model, use `summary(object)`; to see all its contents, use `str(object)`.

Author(s)

Thomas Nagler

See Also

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

Examples

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

## see the object's content or a summary
str(obj)
summary(obj)

## a selection of functions 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)  # surface plot of copula density
contour(obj)  # contour plot with standard normal margins
print(obj)  # brief overview of BiCop object
summary(obj)  # comprehensive overview of BiCop object
```

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, check.pars = TRUE)
```

Arguments

`u1, u2`	numeric vectors of equal length with values in [0,1].
`family`	integer; single number or vector of size `length(u1)`; defines 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`	numeric; single number or vector of size `length(u1)`; copula parameter.
`par2`	numeric; single number or vector of size `length(u1)`; second parameter for

	bivariate copulas with two parameters (BB1, BB6, BB7, BB8, Tawn type 1 and type 2; default: `par2 = 0`).
`obj`	`BiCop` object containing the family and parameter specification.
`check.pars`	logical; default is `TRUE`; if `FALSE`, checks for family/parameter-consistency are ommited (should only be used with care).

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

- of the copula `family`
- with parameter(s) `par`, `par2`
- evaluated at `u1` and `u2`.

Note

The calculation of the cumulative distribution function (CDF) of the Student's t copula (`family = 2`) is only approximate. For numerical reasons, the degree of freedom parameter (`par2`) is rounded to an integer before calculation of the CDF.

Author(s)

Eike Brechmann

See Also

`BiCopPDF`, `BiCopHfunc`, `BiCopSim`, `BiCop`

Examples

```
## simulate from a bivariate Clayton copula
set.seed(123)
cop <- BiCop(family = 3, par = 3.4)
simdata <- BiCopSim(300, cop)

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

## select a bivariate copula for the simulated data
```

```
cop <- BiCopSelect(u1, u2)
summary(cop)
## and evaluate its CDF
BiCopCDF(u1, u2, cop)
```

BiCopCheck *Check for family/parameter consistency in bivariate copula models*

Description

The function checks if a certain combination of copula family and parameters can be used within other functions of this package.

Usage

```
BiCopCheck(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 is `par2 = 0`).

`...` used internally.

Value

A logical indicating wether the family can be used with the parameter specification.

Author(s)

Thomas Nagler

Examples

```
## check parameter of Clayton copula
BiCopCheck(3, 1)  # works

## Not run: BiCopCheck(3, -1)  # does not work (only positive parameter is allowed)
```

`BiCopChiPlot` *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", ...)
```

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 $u_{i,j},\ i=1,...,N,\ j=1,2,$ the chi-plot is based on the following two quantities: the chi-statistics

$$\chi_i = \frac{\hat{F}_{1,2}(u_{i,1},u_{i,2}) - \hat{F}_1(u_{i,1})\hat{F}_2(u_{i,2})}{\sqrt{\hat{F}_1(u_{i,1})(1-\hat{F}_1(u_{i,1}))\hat{F}_2(u_{i,2})(1-\hat{F}_2(u_{i,2}))}},$$

and the lambda-statistics

$$\lambda_i = 4sgn\left(\tilde{F}_1(u_{i,1}),\tilde{F}_2(u_{i,2})\right)\cdot\max\left(\tilde{F}_1(u_{i,1})^2,\tilde{F}_2(u_{i,2})^2\right),$$

where $\hat{F}_1$, $\hat{F}_2$ and $\hat{F}_{1,2}$ are the empirical distribution functions of the uniform random variables U_1 and U_2 and of (U_1,U_2), respectively. Further, $\tilde{F}_1 = \hat{F}_1 - 0.5$ and $\tilde{F}_2 = \hat{F}_2 - 0.5$.

These quantities only depend on the ranks of the data and are scaled to the interval $[0,1]$. λ_i measures a distance of a data point $(u_{i,1},u_{i,2})$ to the center of the bivariate data set, while χ_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 χ_i close to zero indicate independence—corresponding to $F_{1,2} = F_1F_2$.

When plotting these quantities, the pairs of (λ_i,χ_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 `u1`$=(u_{1,1},...,u_{N,1})$ and `u2`$=(u_{1,2},...,u_{N,2})$.

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 `u1` and where the chosen values correspond to an approximate significance level of 10%.

Author(s)

Natalia Belgorodski, Ulf Schepsmeier

References

Abberger, K. (2004). A simple graphical method to explore tail-dependence in stock-return pairs. Discussion Paper, University of Konstanz, Germany.

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

# simulate copula data
fam <- 1
tau <- 0.5
par <- BiCopTau2Par(fam, tau)
cop <- BiCop(fam, par)
set.seed(123)
dat <- BiCopSim(500, cop)

# create chi-plots
op <- 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")
par(op)
```

BiCopCompare — *Shiny app for bivariate copula selection*

Description

The function starts a shiny app which visualizes copula data and allows to compare it with overlays of density contours or simulated data from different copula families with fitted parameters. Several specifications for the margins are available.

Usage

```
BiCopCompare(u1, u2, familyset = NA, rotations = TRUE)
```

Arguments

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

`familyset` Vector of bivariate copula families to select from. The vector has to include at least one bivariate copula family that allows for positive and one that allows for negative dependence. If `familyset = NA` (default), selection among all possible families is performed. If a vector of negative numbers is provided, selection among all but `abs(familyset)` families is performed. Coding of bivariate copula families:
`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)

`rotations`	If `TRUE`, all rotations of the families in `familyset` are included (or substracted).

Value

A `BiCop` object containing the model selected by the user.

Author(s)

Matthias Killiches, Thomas Nagler

Examples

```
# load data
data(daxreturns)

# find a suitable copula family for the first two stocks
## Not run: fit <- BiCopCompare(daxreturns[, 1], daxreturns[, 2])
```

`BiCopCondSim` *Conditional simulation from a Bivariate Copula*

Description

This function simulates from a parametric bivariate copula, where on of the variables is fixed. I.e., we simulate either from $C_{2|1}(u_2|u_1;\theta)$ or $C_{1|2}(u_1|u_2;\theta)$, which are both conditional distribution functions of one variable given another.

Usage

```
BiCopCondSim(N, cond.val, cond.var, family, par, par2 = 0, obj = NULL,
  check.pars = TRUE)
```

Arguments

`N`	Number of observations simulated.
`cond.val`	numeric vector of length `N` containing the values to condition on.
`cond.var`	either `1` or `2`; the variable to condition on.
`family`	integer; single number or vector of size `N`; defines 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` numeric; single number or vector of size `N`; copula parameter.

`par2` numeric; single number or vector of size `N`; 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 a positive integer for the Students's t copula `family = 2`.

`obj` `BiCop` object containing the family and parameter specification.

`check.pars` logical; default is `TRUE`; if `FALSE`, checks for family/parameter-consistency are ommited (should only be used with care).

Details

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

```
BiCopCondSim(N, cond.val, cond.var, obj)
```

can be used.

Value

A length N vector of simulated from conditional distributions related to bivariate copula with family and parameter(s) par, par2.

Author(s)

Thomas Nagler

See Also

BiCopCDF, BiCopPDF, RVineSim

Examples

```
# create bivariate t-copula
obj <- BiCop(family = 2, par = -0.7, par2 = 4)

# simulate 500 observations of (U1, U2)
sim <- BiCopSim(500, obj)
hist(sim[, 1])  # data have uniform distribution
hist(sim[, 2])  # data have uniform distribution

# simulate 500 observations of (U2 | U1 = 0.7)
sim1 <- BiCopCondSim(500, cond.val = 0.7, cond.var = 1, obj)
hist(sim1)  # not uniform!

# simulate 500 observations of (U1 | U2 = 0.1)
sim2 <- BiCopCondSim(500, cond.val = 0.1, cond.var = 2, obj)
hist(sim2)  # not uniform!
```

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, check.pars = TRUE)
```

Arguments

- `u1, u2` — numeric vectors of equal length with values in [0,1].
- `family` — integer; single number or vector of size `length(u1)`; defines 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` — numeric; single number or vector of size `length(u1)`; copula parameter.
- `par2` — integer; single number or vector of size `length(u1)`; second parameter for the t-Copula; default is `par2 = 0`, should be an positive integer for the Students's t copula `family = 2`.
- `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.
- `check.pars` — logical; default is `TRUE`; if `FALSE`, checks for family/parameter-consistency are ommited (should only be used with care).

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

- of the copula `family`
- with parameter(s) `par`, `par2`
- with respect to `deriv`,
- evaluated at `u1` and `u2`.

Author(s)

Ulf Schepsmeier

References

Schepsmeier, U. and J. Stoeber (2014). Derivatives and Fisher information of bivariate copulas. Statistical Papers, 55 (2), 525-542.
http://link.springer.com/article/10.1007/s00362-013-0498-x.

See Also

RVineGrad, RVineHessian, BiCopDeriv2, BiCopHfuncDeriv, BiCop

Examples

```
## simulate from a bivariate Student-t copula
set.seed(123)
cop <- BiCop(family = 2, par = -0.7, par2 = 4)
simdata <- BiCopSim(100, cop)

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

## estimate a Student-t copula for the simulated data
cop <- BiCopEst(u1, u2, family = 2)
## and evaluate its derivative w.r.t. the second argument u2
BiCopDeriv(u1, u2, cop, deriv = "u2")
```

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

```
BiCopDeriv2(u1, u2, family, par, par2 = 0, deriv = "par", obj = NULL,
  check.pars = TRUE)
```

Arguments

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

family
: integer; single number or vector of size `length(u1)`; defines 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
: integer; single number or vector of size `length(u1)`; second parameter for the t-Copula; default is `par2 = 0`, should be an positive integer for the Students's t copula `family = 2`.

deriv
: Derivative 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.

check.pars logical; default is TRUE; if FALSE, checks for family/parameter-consistency are ommited (should only be used with care).

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-order bivariate copula derivative

- of the copula family
- with parameter(s) par, par2
- with respect to deriv
- evaluated at u1 and u2.

Author(s)

Ulf Schepsmeier, Jakob Stoeber

References

Schepsmeier, U. and J. Stoeber (2014). Derivatives and Fisher information of bivariate copulas. Statistical Papers, 55 (2), 525-542.
http://link.springer.com/article/10.1007/s00362-013-0498-x.

See Also

RVineGrad, RVineHessian, BiCopDeriv, BiCopHfuncDeriv, BiCop

Examples

```
## simulate from a bivariate Student-t copula
set.seed(123)
cop <- BiCop(family = 2, par = -0.7, par2 = 4)
simdata <- BiCopSim(100, cop)

## second derivative of the Student-t copula w.r.t. the first parameter
u1 <- simdata[,1]
u2 <- simdata[,2]
BiCopDeriv2(u1, u2, cop, deriv = "par")

## estimate a Student-t copula for the simulated data
cop <- BiCopEst(u1, u2, family = 2)
## and evaluate its second derivative w.r.t. the second argument u2
BiCopDeriv2(u1, u2, cop, deriv = "u2")
```

`BiCopEst` *Parameter Estimation for Bivariate Copula Data*

Description

This function estimates the parameter(s) of a bivariate copula using either inversion of empirical Kendall's tau (for one parameter copula families only) or maximum likelihood estimation for implemented copula families.

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` indicates the estimation method: either maximum likelihood estimation (`method = "mle"`; default) or inversion of Kendall’s tau (`method = "itau"`). For `method = "itau"` only one parameter families and the Student t copula can be used (`family = 1,2,3,4,5,6,13,14,16,23,24,26,33,34 or 36`). For the t-copula, `par2` is found by a crude profile likelihood optimization over the interval (2, 10].

`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 (opitional).

Details

If `method = "itau"`, 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 = "itau"`, 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 = "itau"`. 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, augmented with the following entries:

- `se, se2` standard errors for the parameter estimates (if `se = TRUE`,
- `nobs` number of observations,
- `logLik` log likelihood
- `AIC` Aikaike's Informaton Criterion,
- `BIC` Bayesian's Informaton Criterion,
- `emptau` empirical value of Kendall's tau,
- `p.value.indeptest` p-value of the independence test.

Note

For a comprehensive summary of the fitted model, use `summary(object)`; to see all its contents, use `str(object)`.

Author(s)

Ulf Schepsmeier, Eike Brechmann, Jakob Stoeber, Carlos Almeida

References

Joe, H. (1997). Multivariate Models and Dependence Concepts. Chapman and Hall, London.

See Also

BiCop, BiCopPar2Tau, BiCopTau2Par, RVineSeqEst, BiCopSelect,

Examples

```
## Example 1: bivariate Gaussian copula
dat <- BiCopSim(500, 1, 0.7)
u1 <- dat[, 1]
v1 <- dat[, 2]

# estimate parameters of Gaussian copula by inversion of Kendall's tau
est1.tau <- BiCopEst(u1, v1, family = 1, method = "itau")
est1.tau  # short overview
summary(est1.tau)  # comprehensive overview
str(est1.tau)  # see all contents of the object

# check if parameter actually coincides with inversion of Kendall's tau
tau1 <- cor(u1, v1, method = "kendall")
all.equal(BiCopTau2Par(1, tau1), est1.tau$par)

# maximum likelihood estimate for comparison
est1.mle <- BiCopEst(u1, v1, family = 1, method = "mle")
summary(est1.mle)


## Example 2: bivariate Clayton and survival Gumbel copulas
# simulate from a Clayton copula
dat <- BiCopSim(500, 3, 2.5)
u2 <- dat[, 1]
v2 <- dat[, 2]

# empirical Kendall's tau
tau2 <- cor(u2, v2, method = "kendall")

# inversion of empirical Kendall's tau for the Clayton copula
BiCopTau2Par(3, tau2)
BiCopEst(u2, v2, family = 3, method = "itau")

# inversion of empirical Kendall's tau for the survival Gumbel copula
BiCopTau2Par(14, tau2)
BiCopEst(u2, v2, family = 14, method = "itau")

# maximum likelihood estimates for comparison
BiCopEst(u2, v2, family = 3, method = "mle")
BiCopEst(u2, v2, family = 14, method = "mle")
```

BiCopEstList — *List of Maximum Likelihood Estimates for Several Bivariate Copula Families*

Description

This function allows to compare bivariate copula models accross a number of families w.r.t. the fit statistics log-likelihood, AIC, and BIC. For each family, the parameters are estimated by maximum likelihood.

Usage

```
BiCopEstList(u1, u2, familyset = NA, weights = NA, rotations = TRUE, ...)
```

Arguments

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

`familyset` Vector of bivariate copula families to select from. The vector has to include at least one bivariate copula family that allows for positive and one that allows for negative dependence. If `familyset = NA` (default), selection among all possible families is performed. Coding of bivariate copula families:
`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)

`weights` Numerical; weights for each observation (optional).

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

`...` further arguments passed to `BiCopEst`.

Details

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 $u_{i,j},\ i = 1, ..., N,\ j = 1, 2,$ the AIC of a bivariate copula family c with parameter(s) $\boldsymbol{\theta}$ is defined as

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

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

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

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

Value

A list containing

`models` a list of `BiCop` objects corresponding to the familyset (only families corresponding to the sign of the empirical Kendall's tau are used),

`summary` a data frame containing the log-likelihoods, AICs, and BICs of all the fitted models.

Author(s)

Thomas Nagler

References

Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In B. N. Petrov and F. Csaki (Eds.), Proceedings of the Second International Symposium on Information Theory Budapest, Akademiai Kiado, pp. 267-281.

Schwarz, G. E. (1978). Estimating the dimension of a model. Annals of Statistics 6 (2), 461-464.

See Also

`BiCop`, `BiCopEst`

Examples

```
## compare models
data(daxreturns)
comp <- BiCopEstList(daxreturns[, 1], daxreturns[, 4])
```

`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 (Wang and Wells, 2000; Genest et al., 2006). 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
`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"`)

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"; only for `method = "kendall"`)
18 = rotated BB6 copula (180 degrees; "survival BB6"; only for `method = "kendall"`)
19 = rotated BB7 copula (180 degrees; "survival BB7"; only for `method = "kendall"`)
20 = rotated BB8 copula (180 degrees; "survival BB8"; only for `method = "kendall"`)
23 = rotated Clayton copula (90 degrees)
24 = rotated Gumbel copula (90 degrees)
26 = rotated Joe copula (90 degrees)
27 = rotated BB1 copula (90 degrees; only for `method = "kendall"`)
28 = rotated BB6 copula (90 degrees; only for `method = "kendall"`)
29 = rotated BB7 copula (90 degrees; only for `method = "kendall"`)
30 = rotated BB8 copula (90 degrees; only for `method = "kendall"`)
33 = rotated Clayton copula (270 degrees)
34 = rotated Gumbel copula (270 degrees)
36 = rotated Joe copula (270 degrees)
37 = rotated BB1 copula (270 degrees; only for `method = "kendall"`)
38 = rotated BB6 copula (270 degrees; only for `method = "kendall"`)
39 = rotated BB7 copula (270 degrees; only for `method = "kendall"`)
40 = rotated BB8 copula (270 degrees; only for `method = "kendall"`)

`par` Copula parameter (optional).

`par2` Second parameter for bivariate t-copula (optional); default: `par2 = 0`.

`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

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

`B` Integer; number of bootstrap samples (default: `B = 100`). For `B = 0` only the 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

$$H_0 : \boldsymbol{H}(\theta) + \boldsymbol{C}(\theta) = 0$$

against the alternative

$$H_0 : \boldsymbol{H}(\theta) + \boldsymbol{C}(\theta) \neq 0,$$

where $\boldsymbol{H}(\theta)$ is the expected Hessian matrix and $\boldsymbol{C}(\theta)$ 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 $\boldsymbol{H}(\theta)$ and $\boldsymbol{C}(\theta)$. 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 proposed by Wang and Wells (2000). For computation of p-values, the parametric bootstrap described by Genest et al. (2006) is used. 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

Huang, W. and A. Prokhorov (2014). A goodness-of-fit test for copulas. Econometric Reviews, 33 (7), 751-771.

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

Genest, C., Quessy, J. F., and Remillard, B. (2006). Goodness-of-fit Procedures for Copula Models Based on the Probability Integral Transformation. Scandinavian Journal of Statistics, 33(2), 337-366. Luo J. (2011). Stepwise estimation of D-vines with arbitrary specified copula pairs and EDA tools. Diploma thesis, Technische Universitaet Muenchen.
http://mediatum.ub.tum.de/?id=1079291.

White, H. (1982) Maximum likelihood estimation of misspecified models, Econometrica, 50, 1-26.

See Also

BiCopDeriv2, BiCopDeriv, BiCopIndTest, BiCopVuongClarke

Examples

```
# simulate from a bivariate Clayton copula

simdata <- BiCopSim(100, 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 White'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
BiCopGofTest(u1, u2, family = 3, method = "kendall", B=50)

# perform Kendall's goodness-of-fit test for the Frank copula
BiCopGofTest(u1, u2, family = 5, method = "kendall", B=50)
```

BiCopHfunc *Conditional Distribution Function of a Bivariate Copula*

Description

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

Usage

```
BiCopHfunc(u1, u2, family, par, par2 = 0, obj = NULL, check.pars = TRUE)

BiCopHfunc1(u1, u2, family, par, par2 = 0, obj = NULL, check.pars = TRUE)

BiCopHfunc2(u1, u2, family, par, par2 = 0, obj = NULL, check.pars = TRUE)
```

Arguments

`u1, u2`	numeric vectors of equal length with values in [0,1].
`family`	integer; single number or vector of size `length(u1)`; defines 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` numeric; single number or vector of size `length(u1)`; copula parameter.

`par2` numeric; single number or vector of size `length(u1)`; 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.

`check.pars` logical; default is `TRUE`; if `FALSE`, checks for family/parameter-consistency are ommited (should only be used with care).

Details

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

$$h_1(u_2|u_1;\boldsymbol{\theta}) := P(U_2 \leq u_2|U_1 = u_1) = \frac{\partial C(u_1, u_2; \boldsymbol{\theta})}{\partial u_1},$$

$$h_2(u_1|u_2;\boldsymbol{\theta}) := P(U_1 \leq u_1|U_2 = u_2) = \frac{\partial C(u_1, u_2; \boldsymbol{\theta})}{\partial u_2},$$

where $(U_1, U_2) \sim C$, and C is a bivariate copula distribution function with parameter(s) $\boldsymbol{\theta}$. For more details see Aas et al. (2009).

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

```
BiCopHfunc(u1, u2, obj)
BiCopHfunc1(u1, u2, obj)
BiCopHfunc2(u1, u2, obj)
```

can be used.

Value

`BiCopHfunc` returns a list with

- `hfunc1` Numeric vector of the conditional distribution function (h-function) of the copula `family` with parameter(s) `par`, `par2` evaluated at `u2` given `u1`, i.e., $h_1(u_2|u_1;\boldsymbol{\theta})$.
- `hfunc2` Numeric vector of the conditional distribution function (h-function) of the copula `family` with parameter(s) `par`, `par2` evaluated at `u1` given `u2`, i.e., $h_2(u_1|u_2;\boldsymbol{\theta})$.

`BiCopHfunc1` is a faster version that only calculates `hfunc1`; `BiCopHfunc2` only calculates `hfunc2`.

Author(s)

Ulf Schepsmeier

References

Aas, K., C. Czado, A. Frigessi, and H. Bakken (2009). Pair-copula constructions of multiple dependence. Insurance: Mathematics and Economics 44 (2), 182-198.

See Also

`BiCopHinv`, `BiCopPDF`, `BiCopCDF`, `RVineLogLik`, `RVineSeqEst`, `BiCop`

Examples

```
data(daxreturns)

# h-functions of the Gaussian copula
cop <- BiCop(family = 1, par = 0.5)
h <- BiCopHfunc(daxreturns[, 2], daxreturns[, 1], cop)

# or using the fast versions
h1 <- BiCopHfunc1(daxreturns[, 2], daxreturns[, 1], cop)
h2 <- BiCopHfunc2(daxreturns[, 2], daxreturns[, 1], cop)
all.equal(h$hfunc1, h1)
all.equal(h$hfunc2, h2)
```

`BiCopHfuncDeriv` *Derivatives of the h-Function of a Bivariate Copula*

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

```
BiCopHfuncDeriv(u1, u2, family, par, par2 = 0, deriv = "par", obj = NULL,
  check.pars = TRUE)
```

Arguments

`u1, u2`	numeric vectors of equal length with values in [0,1].
`family`	integer; single number or vector of size `length(u1)`; defines the bivariate copula family: \cr `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	numeric; single number or vector of size length(u1); copula parameter.
par2	integer; single number or vector of size length(u1); second parameter for the t-Copula; default is par2 = 0, should be an positive integer for the Students's t copula family = 2.
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) "u2" = derivative with respect to the second argument u2
obj	BiCop object containing the family and parameter specification.
check.pars	logical; default is TRUE; if FALSE, checks for family/parameter-consistency are ommited (should only be used with care).

Details

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

```
BiCopHfuncDeriv(u1, u2, obj, deriv = "par")
```

can be used.

Value

A numeric vector of the conditional bivariate copula derivative

- of the copula family,
- with parameter(s) par, par2,
- with respect to deriv,
- evaluated at u1 and u2.

Author(s)

Ulf Schepsmeier

References

Schepsmeier, U. and J. Stoeber (2014). Derivatives and Fisher information of bivariate copulas. Statistical Papers, 55 (2), 525-542.
http://link.springer.com/article/10.1007/s00362-013-0498-x.

See Also

RVineGrad, RVineHessian, BiCopDeriv2, BiCopDeriv2, BiCopHfuncDeriv, BiCop

Examples

```
## simulate from a bivariate Student-t copula
set.seed(123)
cop <- BiCop(family = 2, par = -0.7, par2 = 4)
simdata <- BiCopSim(100, cop)

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

## estimate a Student-t copula for the simulated data
cop <- BiCopEst(u1, u2, family = 2)
## and evaluate the derivative of the conditional copula
## w.r.t. the second argument u2
BiCopHfuncDeriv(u1, u2, cop, deriv = "u2")
```

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,
  check.pars = TRUE)
```

Arguments

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

`family` integer; single number or vector of size `length(u1)`; defines 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` numeric; single number or vector of size `length(u1)`; copula parameter.

`par2` integer; single number or vector of size `length(u1)`; second parameter for the t-Copula; default is `par2 = 0`, should be an positive integer for the Students's t copula `family = 2`.

`deriv` Derivative 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)
`"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)
`"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.

`check.pars` logical; default is `TRUE`; if `FALSE`, checks for family/parameter-consistency are ommited (should only be used with care).

Details

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

```
BiCopHfuncDeriv2(u1, u2, obj, deriv = "par")
```

can be used.

Value

A numeric vector of the second-order conditional bivariate copula derivative

- of the copula `family`
- with parameter(s) `par`, `par2`
- with respect to `deriv`
- evaluated at `u1` and `u2`.

Author(s)

Ulf Schepsmeier, Jakob Stoeber

References

Schepsmeier, U. and J. Stoeber (2014). Derivatives and Fisher information of bivariate copulas. Statistical Papers, 55 (2), 525-542.
http://link.springer.com/article/10.1007/s00362-013-0498-x.

See Also

RVineGrad, RVineHessian, BiCopDeriv, BiCopDeriv2, BiCopHfuncDeriv, BiCop

Examples

```
## simulate from a bivariate Student-t copula
set.seed(123)
cop <- BiCop(family = 2, par = -0.7, par2 = 4)
simdata <- BiCopSim(100, cop)

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

## estimate a Student-t copula for the simulated data
cop <- BiCopEst(u1, u2, family = 2)
## and evaluate the derivative of the conditional copula
## w.r.t. the second argument u2
BiCopHfuncDeriv2(u1, u2, cop, deriv = "u2")
```

BiCopHinv *Inverse Conditional Distribution Function of a Bivariate Copula*

Description

Evaluate the inverse conditional distribution function (inverse h-function) of a given parametric bivariate copula.

Usage

```
BiCopHinv(u1, u2, family, par, par2 = 0, obj = NULL, check.pars = TRUE)

BiCopHinv1(u1, u2, family, par, par2 = 0, obj = NULL, check.pars = TRUE)

BiCopHinv2(u1, u2, family, par, par2 = 0, obj = NULL, check.pars = TRUE)
```

Arguments

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

`family` integer; single number or vector of size `length(u1)`; defines 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` numeric; single number or vector of size `length(u1)`; copula parameter.

`par2` numeric; single number or vector of size `length(u1)`; 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.

`check.pars` — logical; default is `TRUE`; if `FALSE`, checks for family/parameter-consistency are ommited (should only be used with care).

Details

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

$$h_1(u_2|u_1;\boldsymbol{\theta}) := P(U_2 \leq u_2|U_1 = u_1) = \frac{\partial C(u_1, u_2; \boldsymbol{\theta})}{\partial u_1},$$

$$h_2(u_1|u_2;\boldsymbol{\theta}) := P(U_1 \leq u_1|U_2 = u_2) = \frac{\partial C(u_1, u_2; \boldsymbol{\theta})}{\partial u_2},$$

where $(U_1, U_2) \sim C$, and C is a bivariate copula distribution function with parameter(s) $\boldsymbol{\theta}$. For more details see Aas et al. (2009).

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

```
BiCopHinv(u1, u2, obj),
BiCopHinv1(u1, u2, obj),
BiCopHinv2(u1, u2, obj)
```

can be used.

Value

`BiCopHinv` returns a list with

`hinv1` — Numeric vector of the inverse conditional distribution function (inverse h-function) of the copula `family` with parameter(s) `par`, `par2` evaluated at `u2` given `u1`, i.e., $h_1^{-1}(u_2|u_1;\boldsymbol{\theta})$.

`hinv2` — Numeric vector of the inverse conditional distribution function (inverse h-function) of the copula `family` with parameter(s) `par`, `par2` evaluated at `u1` given `u2`, i.e., $h_2^{-1}(u_1|u_2;\boldsymbol{\theta})$.

`BiCopHinv1` is a faster version that only calculates `hinv1`; `BiCopHinv2` only calculates `hinv2`.

Author(s)

Ulf Schepsmeier, Thomas Nagler

References

Aas, K., C. Czado, A. Frigessi, and H. Bakken (2009). Pair-copula constructions of multiple dependence. Insurance: Mathematics and Economics 44 (2), 182-198.

See Also

BiCopHfunc, BiCopPDF, BiCopCDF, RVineLogLik, RVineSeqEst, BiCop

Examples

```
# inverse h-functions of the Gaussian copula
cop <- BiCop(1, 0.5)
hi <- BiCopHinv(0.1, 0.2, cop)

# or using the fast versions
hi1 <- BiCopHinv1(0.1, 0.2, cop)
hi2 <- BiCopHinv2(0.1, 0.2, cop)
all.equal(hi$hinv1, hi1)
all.equal(hi$hinv2, hi2)

# check if it is actually the inverse
cop <- BiCop(3, 3)
all.equal(0.2, BiCopHfunc1(0.1, BiCopHinv1(0.1, 0.2, cop), cop))
all.equal(0.1, BiCopHfunc2(BiCopHinv2(0.1, 0.2, cop), 0.2, cop))
```

BiCopIndTest — *Independence Test for Bivariate Copula Data*

Description

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

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

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

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

$$\texttt{p.value} = 2 \times (1 - \Phi(T)),$$

where Φ is the standard normal distribution function.

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

```
## Example 1: Gaussian copula with large dependence parameter
cop <- BiCop(1, 0.7)
dat <- BiCopSim(500, cop)

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

## Example 2: Gaussian copula with small dependence parameter
cop <- BiCop(1, 0.01)
dat <- BiCopSim(500, cop)

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

`BiCopKDE` *Kernel estimate of a Bivariate Copula Density*

Description

A kernel density estimate of the copula density is visualized. The function provides the same options as `plot.BiCop`. Further arguments can be passed to `kdecop` to modify the estimate.

Usage

```
BiCopKDE(u1, u2, type = "contour", margins, size, kde.pars = list(), ...)
```

Arguments

u1, u2	numeric vectors of equal length with values in [0,1].
type	plot type; either "contour" or "surface" (partial matching is activated) for a contour or perspective/surface plot respectively.
margins	only relevant for types "contour" and "surface"; options are: "unif" for the original copula density, "norm" for the transformed density with standard normal margins, "exp" with standard exponential margins, and "flexp" with flipped exponential margins. Default is "norm" for type = "contour", and "unif" for type = "surface". "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; the plot is based on values on a size x size grid; default is 100 for type = "contour", and 25 for type = "surface".
kde.pars	list of arguments passed to kdecop.
...	optional arguments passed to contour or wireframe.

Details

For further details on estimation see kdecop.

Author(s)

Thomas Nagler

Examples

```
# simulate data from Joe copula
cop <- BiCop(3, tau = 0.3)
u <- BiCopSim(1000, cop)
contour(cop)  # true contours

# kernel contours with standard normal margins
BiCopKDE(u[, 1], u[, 2])
BiCopKDE(u[, 1], u[, 2], kde.pars = list(mult = 0.5))  # undersmooth
BiCopKDE(u[, 1], u[, 2], kde.pars = list(mult = 2))  # oversmooth

# kernel density with uniform margins
BiCopKDE(u[, 1], u[, 2], type = "surface", zlim = c(0, 4))
plot(cop, zlim = c(0, 4))  # true density

# kernel contours are also used in pairs.copuladata
data(daxreturns)
data <- as.copuladata(daxreturns)
pairs(data[c(4, 5, 14, 15)])
```

BiCopKPlot — *Kendall's Plot for Bivariate 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 `W.in` and `Hi.sort` are returned (see below; default: `PLOT = TRUE`).
`...`	Additional plot arguments.

Details

For observations $u_{i,j},\ i = 1, ..., N,\ j = 1, 2$, 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

$$W_{i:n} = N\binom{N-1}{i-1}\int\limits_0^1 \omega k_0(\omega)(K_0(\omega))^{i-1}(1-K_0(\omega))^{N-i}d\omega,$$

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).
`Hi.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

BiCopMetaContour, BiCopChiPlot, BiCopLambda, BiCopGofTest

Examples

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

# simulate from Gaussian copula
fam <- 1
par <- BiCopTau2Par(fam, tau)
cop1 <- BiCop(fam, par)
set.seed(123)
dat1 <- BiCopSim(n, cop1)

# simulate from Clayton copula
fam <- 3
par <- BiCopTau2Par(fam, tau)
cop2 <- BiCop(fam, par)
set.seed(123)
dat2 <- BiCopSim(n, cop2)

# create K-plots
op <- par(mfrow = c(1, 2))
BiCopKPlot(dat1[,1], dat1[,2], main = "Gaussian copula")
BiCopKPlot(dat2[,1], dat2[,2], main = "Clayton copula")
par(op)
```

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

Description

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

Usage

```
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; 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.
`...`	Additional plot arguments.

Details

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

```
BiCopLambda(obj, PLOT = TRUE, ...)
```

and

```
BiCopLambda((u1, u2, obj, PLOT = TRUE, ...)
```

can be used.

Value

`empLambda` If the empirical lambda-function is chosen and `PLOT = FALSE`, a vector of the empirical lambda's is returned.

`theoLambda` If the theoretical lambda-function is chosen and `PLOT = FALSE`, a vector of the theoretical lambda's is returned.

Note

The λ-function is characteristic for each bivariate copula family and defined by Kendall's distribution function K:

$$\lambda(v, \theta) := v - K(v, \theta)$$

with

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

For Archimedean copulas one has the following closed form expression in terms of the generator function φ of the copula C_θ:

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

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

For the bivariate Gaussian and Student-t copula no closed form expression for the theoretical λ-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 λ-function also shows the limits of the λ-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 `u1` and/or `u2`. In particular, for copulas rotated by 90 degrees `u1` has to be set to `1-u1`, for 270 degrees `u2` to `1-u2` and for survival copulas `u1` and `u2` to `1-u1` and `1-u2`, respectively. Then λ-functions for the corresponding non-rotated copula families can be considered.

Author(s)

Ulf Schepsmeier

References

Genest, C. and L.-P. Rivest (1993). Statistical inference procedures for bivariate Archimedean copulas. Journal of the American Statistical Association, 88 (423), 1034-1043.

Schepsmeier, U. (2010). Maximum likelihood estimation of C-vine pair-copula constructions based on bivariate copulas from different families. Diploma thesis, Technische Universitaet Muenchen. `http://mediatum.ub.tum.de/?id=1079296`.

See Also

`BiCopMetaContour`, `BiCopKPlot`, `BiCopChiPlot`, `BiCop`

Examples

```
# simulate from Clayton copula
cop <- BiCop(3, tau = 0.5)
dat <- BiCopSim(1000, cop)

# create lambda-function plots
op <- par(mfrow = c(1, 3))
BiCopLambda(dat[, 1], dat[, 2])  # empirical lambda-function
BiCopLambda(cop) # theoretical lambda-function
BiCopLambda(dat[, 1], dat[, 2], cop) # both
par(op)
```

BiCopMetaContour *Contour Plot of Bivariate Meta Distribution*

Description

Note: This function is deprecated and only available for backwards compatibility. See `contour.BiCop` for contour plots of parametric copulas, and `BiCopKDE` for kernel estimates.

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 `0` (default).

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

`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 = 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).
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 = 0 (independence copula) and margins = "unif" (uniform margins) is not possible because all z-values are equal.

Author(s)

Ulf Schepsmeier, Alexander Bauer

See Also

BiCopChiPlot, BiCopKPlot, BiCopLambda

Examples

```
## meta Clayton distribution  with Gaussian margins
cop <- BiCop(family = 1, tau = 0.5)
BiCopMetaContour(obj = cop, main = "Clayton - normal margins")
# better:
contour(cop, main = "Clayton - normal margins")
```

```
## empirical contour plot with standard normal margins
dat <- BiCopSim(1000, cop)
BiCopMetaContour(dat[, 1], dat[, 2], bw = 2, family = "emp",
                 main = "empirical - normal margins")
# better:
BiCopKDE(dat[, 1], dat[, 2],
        main = "empirical - normal margins")

## empirical contour plot with exponential margins
BiCopMetaContour(dat[, 1], dat[, 2], bw = 2,
                 main = "empirical - exponential margins",
                 margins = "exp", margins.par = 1)
# better:
BiCopKDE(dat[, 1], dat[, 2],
         main = "empirical - exponential margins",
         margins = "exp")
```

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).

No.	Short name	Long name
0	"I"	"Independence"
1	"N"	"Gaussian"
2	"t"	"t"
3	"C"	"Clayton"
4	"G"	"Gumbel"
5	"F"	"Frank"
6	"J"	"Joe"
7	"BB1"	"BB1"
8	"BB6"	"BB6"
9	"BB7"	"BB7"
10	"BB8"	"Frank-Joe"
13	"SC"	"Survival Clayton"
14	"SG"	"Survival Gumbel"

16	"SJ"	"Survival Joe"
17	"SBB1"	"Survival BB1"
18	"SBB6"	"Survival BB6"
19	"SBB7"	"Survival BB7"
20	"SBB8"	"Survival BB8"
23	"C90"	"Rotated Clayton 90 degrees"
24	"G90"	"Rotated Gumbel 90 degrees"
26	"J90"	"Rotated Joe 90 degrees"
27	"BB1_90"	"Rotated BB1 90 degrees"
28	"BB6_90"	"Rotated BB6 90 degrees"
29	"BB7_90"	"Rotated BB7 90 degrees"
30	"BB8_90"	"Rotated Frank-Joe 90 degrees"
33	"C270"	"Rotated Clayton 270 degrees"
34	"G270"	"Rotated Gumbel 270 degrees"
36	"J270"	"Rotated Joe 270 degrees"
37	"BB1_270"	"Rotated BB1 270 degrees"
38	"BB6_270"	"Rotated BB6 270 degrees"
39	"BB7_270"	"Rotated BB7 270 degrees"
40	"BB8_270"	"Rotated Frank-Joe 270 degrees"
104	"Tawn"	"Tawn type 1"
114	"Tawn180"	"Rotated Tawn type 1 180 degrees"
124	"Tawn90"	"Rotated Tawn type 1 90 degrees"
134	"Tawn270"	"Rotated Tawn type 1 270 degrees"
204	"Tawn2"	"Tawn type 2"
214	"Tawn2_180"	"Rotated Tawn type 2 180 degrees"
224	"Tawn2_90"	"Rotated Tawn type 2 90 degrees"
234	"Tawn2_270"	"Rotated Tawn type 2 270 degrees"

`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 number to character expression
```

```
family <- 1
BiCopName(family, short = TRUE)  # short version
BiCopName(family, short = FALSE)  # long version

## family character expression (short version) to number
family <- "C"
BiCopName(family) # as number

## family character expression (long version) to number
family <- "Clayton"
BiCopName(family) # as number

## vectors of families
BiCopName(1:10)    # as character expression
BiCopName(c("Clayton","t","J"))    # 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, par, par2 = 0, obj = NULL, check.pars = TRUE)
```

Arguments

`family` integer; single number or vector of size `n`; defines the bivariate copula family:
`0` = independence copula
`2` = Student t copula (t-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)
Note that the Student's t-copula is not allowed since the CDF of the t-copula is not implemented (see `BiCopCDF`).

`par` numeric; single number or vector of size `n`; copula parameter.

`par2` numeric; single number or vector of size `n`; 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.

`check.pars` logical; default is `TRUE`; if `FALSE`, checks for family/parameter-consistency are ommited (should only be used with care).

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) `par`, `par2`.

Note

The number n can be chosen arbitrarily, but must agree across arguments.

Author(s)

Ulf Schepsmeier

References

Blomqvist, N. (1950). On a measure of dependence between two random variables. The Annals of Mathematical Statistics, 21(4), 593-600.

Nelsen, R. (2006). An introduction to copulas. Springer

Examples

```
## Example 1: Gaussian copula
BiCopPar2Beta(family = 1, par = 0.7)
BiCop(1, 0.7)$beta  # alternative

## Example 2: Clayton copula
BiCopPar2Beta(family = 3, par = 2)

## Example 3: different copula families
BiCopPar2Beta(family = c(3,4,6), par = 2:4)
```

BiCopPar2TailDep *Tail Dependence Coefficients of a Bivariate Copula*

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, check.pars = TRUE)
```

Arguments

family — integer; single number or vector of size n; defines 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` numeric; single number or vector of size `n`; copula parameter.

`par2` numeric; single number or vector of size `n`; 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.

`check.pars` logical; default is `TRUE`; if `FALSE`, checks for family/parameter-consistency are ommited (should only be used with care).

Details

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

```
BiCopPar2TailDep(obj)
```

can be used.

Value

`lower` Lower tail dependence coefficient for the given bivariate copula `family` and parameter(s) `par`, `par2`:

$$\lambda_L = \lim_{u \searrow 0} \frac{C(u,u)}{u}$$

`upper` Upper tail dependence coefficient for the given bivariate copula family `family` and parameter(s) `par`, `par2`:

$$\lambda_U = \lim_{u \nearrow 1} \frac{1 - 2u + C(u,u)}{1-u}$$

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

No.	Lower tail dependence	Upper tail dependence
`1`	-	-
`2`	$2t_{\nu+1}\left(-\sqrt{\nu+1}\sqrt{\frac{1-\theta}{1+\theta}}\right)$	$2t_{\nu+1}\left(-\sqrt{\nu+1}\sqrt{\frac{1-\theta}{1+\theta}}\right)$
`3`	$2^{-1/\theta}$	-
`4`	-	$2-2^{1/\theta}$
`5`	-	-
`6`	-	$2-2^{1/\theta}$
`7`	$2^{-1/(\theta\delta)}$	$2-2^{1/\delta}$
`8`	-	$2-2^{1/(\theta\delta)}$
`9`	$2^{-1/\delta}$	$2-2^{1/\theta}$
`10`	-	$2-2^{1/\theta}$ if $\delta=1$ otherwise 0
`13`	-	$2^{-1/\theta}$
`14`	$2-2^{1/\theta}$	-
`16`	$2-2^{1/\theta}$	-
`17`	$2-2^{1/\delta}$	$2^{-1/(\theta\delta)}$
`18`	$2-2^{1/(\theta\delta)}$	-
`19`	$2-2^{1/\theta}$	$2^{-1/\delta}$
`20`	$2-2^{1/\theta}$ if $\delta=1$ otherwise 0	-
`23, 33`	-	-
`24, 34`	-	-
`26, 36`	-	-
`27, 37`	-	-
`28, 38`	-	-
`29, 39`	-	-
`30, 40`	-	-
`104,204`	-	$\delta+1-(\delta^\theta+1)^{1/\theta}$
`114, 214`	$1+\delta-(\delta^\theta+1)^{1/\theta}$	-
`124, 224`	-	-

134, 234 - -

Note

The number n can be chosen arbitrarily, but must agree across arguments.

Author(s)

Eike Brechmann

References

Joe, H. (1997). Multivariate Models and Dependence Concepts. Chapman and Hall, London.

See Also

BiCopPar2Tau

Examples

```
## Example 1: Gaussian copula
BiCopPar2TailDep(1, 0.7)
BiCop(1, 0.7)$taildep  # alternative

## Example 2: Student-t copula
BiCopPar2TailDep(2, c(0.6, 0.7, 0.8), 4)

## Example 3: different copula families
BiCopPar2TailDep(c(3, 4, 6), 2)
```

BiCopPar2Tau *Kendall's Tau Value of a Bivariate Copula*

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, check.pars = TRUE)
```

Arguments

family
: integer; single number or vector of size `m`; defines 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
: numeric; single number or vector of size `n`; copula parameter.

par2
: numeric; single number or vector of size `n`; second parameter for bivariate copulas with two parameters (t, BB1, BB6, BB7, BB8, Tawn type 1 and type 2; 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.

`check.pars` logical; default is TRUE; if FALSE, checks for family/parameter-consistency are ommited (should only be used with care).

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 vector $(\theta, \delta) =$ (par, par2).

No. (family)	Kendall's tau (tau)
1, 2	$\frac{2}{\pi}\arcsin(\theta)$
3, 13	$\frac{\theta}{\theta+2}$
4, 14	$1-\frac{1}{\theta}$
5	$1-\frac{4}{\theta}+4\frac{D_1(\theta)}{\theta}$
	with $D_1(\theta)=\int_0^\theta \frac{x/\theta}{\exp(x)-1}dx$ (Debye function)
6, 16	$1+\frac{4}{\theta^2}\int_0^1 x\log(x)(1-x)^{2(1-\theta)/\theta}dx$
7, 17	$1-\frac{2}{\delta(\theta+2)}$
8, 18	$1+4\int_0^1 -\log(-(1-t)^\theta+1)(1-t-(1-t)^{-\theta}+(1-t)^{-\theta}t)/(\delta\theta)dt$
9, 19	$1+4\int_0^1((1-(1-t)^\theta)^{-\delta}-1)/(-\theta\delta(1-t)^{\theta-1}(1-(1-t)^\theta)^{-\delta-1})dt$
10, 20	$1+4\int_0^1 -\log\left(((1-t\delta)^\theta-1)/((1-\delta)^\theta-1)\right)$
	$*(1-t\delta-(1-t\delta)^{-\theta}+(1-t\delta)^{-\theta}t\delta)/(\theta\delta)dt$
23, 33	$\frac{\theta}{2-\theta}$
24, 34	$-1-\frac{1}{\theta}$
26, 36	$-1-\frac{4}{\theta^2}\int_0^1 x\log(x)(1-x)^{-2(1+\theta)/\theta}dx$
27, 37	$-1-\frac{2}{\delta(2-\theta)}$
28, 38	$-1-4\int_0^1 -\log(-(1-t)^{-\theta}+1)(1-t-(1-t)^\theta+(1-t)^\theta t)/(\delta\theta)dt$
29, 39	$-1-4\int_0^1((1-(1-t)^{-\theta})^\delta-1)/(-\theta\delta(1-t)^{-\theta-1}(1-(1-t)^{-\theta})^{\delta-1})dt$
30, 40	$-1-4\int_0^1 -\log\left(((1+t\delta)^{-\theta}-1)/((1+\delta)^{-\theta}-1)\right)$
	$*(1+t\delta-(1+t\delta)^\theta-(1+t\delta)^\theta t\delta)/(\theta\delta)dt$
104,114	$\int_0^1 \frac{t(1-t)A''(t)}{A(t)}dt$
	with $A(t)=(1-\delta)t+[(\delta(1-t))^\theta+t^\theta]^{1/\theta}$
204,214	$\int_0^1 \frac{t(1-t)A''(t)}{A(t)}dt$
	with $A(t)=(1-\delta)(1-t)+[(1-t)^{-\theta}+(\delta t)^{-\theta}]^{-1/\theta}$
124,134	$-\int_0^1 \frac{t(1-t)A''(t)}{A(t)}dt$
	with $A(t)=(1-\delta)t+[(\delta(1-t))^{-\theta}+t^{-\theta}]^{-1/\theta}$

`224,234`	$-\int_0^1 \frac{t(1-t)A''(t)}{A(t)}dt$ with $A(t) = (1-\delta)(1-t) + [(1-t)^{-\theta} + (\delta t)^{-\theta}]^{-1/\theta}$

Note

The number n can be chosen arbitrarily, but must agree across arguments.

Author(s)

Ulf Schepsmeier, Tobias Erhardt

References

Joe, H. (1997). Multivariate Models and Dependence Concepts. Chapman and Hall, London.

Czado, C., U. Schepsmeier, and A. Min (2012). Maximum likelihood estimation of mixed C-vines with application to exchange rates. Statistical Modelling, 12(3), 229-255.

See Also

BiCopTau2Par, BiCop

Examples

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

## Example 3: different copula families
theta <- BiCopTau2Par(family = c(3,4,6), tau = c(0.4, 0.5, 0.6))
BiCopPar2Tau(family = c(3,4,6), par = theta)
```

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, check.pars = TRUE)
```

Arguments

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

`family` integer; single number or vector of size `length(u1)`; defines 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` numeric; single number or vector of size `length(u1)`; copula parameter.

`par2` numeric; single number or vector of size `length(u1)`; 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.

`check.pars` logical; default is `TRUE`; if `FALSE`, checks for family/parameter-consistency are ommited (should only be used with care).

Details

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

```
BiCopPDF(u1, u2, obj)
```

can be used.

Value

A numeric vector of the bivariate copula density

- of the copula `family`
- with parameter(s) `par`, `par2`
- evaluated at `u1` and `u2`.

Author(s)

Eike Brechmann

See Also

`BiCopCDF`, `BiCopHfunc`, `BiCopSim`, `BiCop`

Examples

```
## simulate from a bivariate Student-t copula
cop <- BiCop(family = 2, par = -0.7, par2 = 4)
simdata <- BiCopSim(100, cop)

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

## select a bivariate copula for the simulated data
fit <- BiCopSelect(u1, u2)
summary(fit)
## and evaluate its PDF
round(BiCopPDF(u1, u2, fit), 3)
```

`BiCopSelect` *Selection and Maximum Likelihood Estimation of Bivariate Copula Families*

Description

This function selects an appropriate bivariate copula family for given bivariate copula data using one of a range of methods. The corresponding parameter estimates are obtained by maximum likelihood estimation.

Usage

```
BiCopSelect(u1, u2, familyset = NA, selectioncrit = "AIC",
  indeptest = FALSE, level = 0.05, weights = NA, rotations = TRUE,
  se = FALSE, presel = TRUE, method = "mle")
```

Arguments

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

`familyset` Vector of bivariate copula families to select from. The vector has to include at least one bivariate copula family that allows for positive and one that allows for negative dependence. If `familyset = NA` (default), selection among all possible families is performed. If a vector of negative numbers is provided, selection among all but `abs(familyset)` families is performed. Coding of bivariate copula families:
`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)

`selectioncrit` Character indicating the criterion for bivariate copula selection. Possible choices: `selectioncrit = "AIC"` (default), `"BIC"`, or `"logLik"`.

`indeptest` Logical; whether a hypothesis test for the independence of `u1` and `u2` is performed 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 (or substracted).

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

`presel` Logical; whether to exclude families before fitting based on symmetry properties of the data. Makes the selection about 30 (on average), but may yield slightly worse results in few special cases.

`method` indicates the estimation method: either maximum likelihood estimation (`method = "mle"`; default) or inversion of Kendall's tau (`method = "itau"`). For `method = "itau"` only one parameter families and the Student t copula can be used (`family = 1,2,3,4,5,6,13,14,16,23,24,26,33,34` or `36`). For the t-copula, `par2` is found by a crude profile likelihood optimization over the interval (2, 10].

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 $u_{i,j},\ i = 1, ..., N,\ j = 1, 2$, the AIC of a bivariate copula family c with parameter(s) $\boldsymbol{\theta}$ is defined as

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

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

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

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, augmented with the following entries:

`se, se2` standard errors for the parameter estimates (if `se = TRUE`,

`nobs` number of observations,

`logLik` log likelihood

`AIC` Aikaike's Informaton Criterion,

`BIC` Bayesian's Informaton Criterion,

`emptau` empirical value of Kendall's tau,

`p.value.indeptest`
p-value of the independence test.

Note

For a comprehensive summary of the fitted model, use `summary(object)`; to see all its contents, use `str(object)`.

The parameters of the Student t and BB copulas are restricted (see defaults in `BiCopEst` to avoid being to close to their limiting cases.

Author(s)

Eike Brechmann, Jeffrey Dissmann, Thomas Nagler

References

Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In B. N. Petrov and F. Csaki (Eds.), Proceedings of the Second International Symposium on Information Theory Budapest, Akademiai Kiado, pp. 267-281.

Brechmann, E. C. (2010). Truncated and simplified regular vines and their applications. Diploma thesis, Technische Universitaet Muenchen.
http://mediatum.ub.tum.de/?id=1079285.

Manner, H. (2007). Estimation and model selection of copulas with an application to exchange rates. METEOR research memorandum 07/056, Maastricht University.

Schwarz, G. E. (1978). Estimating the dimension of a model. Annals of Statistics 6 (2), 461-464.

See Also

`BiCop`, `BiCopEst`, `RVineStructureSelect`, `RVineCopSelect`, `BiCopIndTest`,

Examples

```
## Example 1: Gaussian copula with large dependence parameter
par <- 0.7
fam <- 1
dat1 <- BiCopSim(500, fam, par)
# select the bivariate copula family and estimate the parameter(s)
cop1 <- BiCopSelect(dat1[, 1], dat1[, 2], familyset = 1:10,
                    indeptest = FALSE, level = 0.05)
cop1  # short overview
summary(cop1)  # comprehensive overview
str(cop1)  # see all contents of the object

## Example 2: Gaussian copula with small dependence parameter
par <- 0.01
fam <- 1
dat2 <- BiCopSim(500, fam, par)
# select the bivariate copula family and estimate the parameter(s)
cop2 <- BiCopSelect(dat2[, 1], dat2[, 2], familyset = 0:10,
                    indeptest = TRUE, level = 0.05)
summary(cop2)

## Example 3: empirical data
```

```
data(daxreturns)
cop3 <- BiCopSelect(daxreturns[, 1], daxreturns[, 4], familyset = 0:10)
summary(cop3)
```

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, check.pars = TRUE)
```

Arguments

N Number of bivariate observations simulated.

family integer; single number or vector of size N; defines 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`	numeric; single number or vector of size `N`; copula parameter.
`par2`	numeric; single number or vector of size `N`; 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 a positive integer for the Students's t copula `family = 2`.
`obj`	`BiCop` object containing the family and parameter specification.
`check.pars`	logical; default is `TRUE`; if `FALSE`, checks for family/parameter-consistency are ommited (should only be used with care).

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 with `family` and parameter(s) `par`, `par2`.

Author(s)

Ulf Schepsmeier

See Also

`BiCop`, `RVineSim`

Examples

```
# simulate from a bivariate t-copula
simdata <- BiCopSim(100, 2, -0.7, par2 = 4)

# or alternatively
obj <- BiCop(family = 2, par = -0.7, par2 = 4)
simdata2 <- BiCopSim(100, obj)
```

`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, check.taus = TRUE)
```

Arguments

`family` integer; single number or vector of size `n`; defines 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` numeric; single number or vector of size `n`; Kendall's tau value (vector with elements in [-1,1]).

`check.taus` logical; default is `TRUE`; if `FALSE`, checks for family/tau-consistency are ommited (should only be used with care).

Value

Parameter (vector) corresponding to the bivariate copula family and the value(s) of Kendall's tau (τ).

No. (`family`)	Parameter (`par`)
`1, 2`	$\sin(\tau \frac{\pi}{2})$
`3, 13`	$2\frac{\tau}{1-\tau}$
`4, 14`	$\frac{1}{1-\tau}$
`5`	no closed form expression (numerical inversion)
`6, 16`	no closed form expression (numerical inversion)
`23, 33`	$2\frac{\tau}{1+\tau}$
`24, 34`	$-\frac{1}{1+\tau}$
`26, 36`	no closed form expression (numerical inversion)

Note

The number `n` can be chosen arbitrarily, but must agree across arguments.

Author(s)

Jakob Stoeber, Eike Brechmann, Tobias Erhardt

References

Joe, H. (1997). Multivariate Models and Dependence Concepts. Chapman and Hall, London.

Czado, C., U. Schepsmeier, and A. Min (2012). Maximum likelihood estimation of mixed C-vines with application to exchange rates. Statistical Modelling, 12(3), 229-255.

See Also

BiCopPar2Tau

Examples

```
## Example 1: Gaussian copula
tau0 <- 0.5
rho <- BiCopTau2Par(family = 1, tau = tau0)
BiCop(1, tau = tau0)$par  # alternative

## Example 2:
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)
```

```
## Example 3: different copula families
theta <- BiCopTau2Par(family = c(3,4,6), tau = c(0.4, 0.5, 0.6))
BiCopPar2Tau(family = c(3,4,6), par = theta)
```

`BiCopVuongClarke` *Scoring Goodness-of-Fit Test based on Vuong And Clarke Tests for Bivariate Copula Data*

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

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

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
`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)

`correction` Correction for the number of parameters. Possible choices: `correction = FALSE` (no correction; default), `"Akaike"` and `"Schwarz"`.

`level` Numerical; significance level of the tests (default: `level = 0.05`).

`rotations` If `TRUE`, all rotations of the families in `familyset` are included (or substracted).

Details

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 documentations 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 $i = 1, ..., k, i \neq j$, 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

Belgorodski, N. (2010) Selecting pair-copula families for regular vines with application to the multivariate analysis of European stock market indices Diploma thesis, Technische Universitaet Muenchen. http://mediatum.ub.tum.de/?id=1079284.

Clarke, K. A. (2007). A Simple Distribution-Free Test for Nonnested Model Selection. Political Analysis, 15, 347-363.

Vuong, Q. H. (1989). Ratio tests for model selection and non-nested hypotheses. Econometrica 57 (2), 307-333.

See Also

BiCopGofTest, RVineVuongTest, RVineClarkeTest, BiCopSelect

Examples

```
# simulate from a t-copula
dat <- BiCopSim(500, 2, 0.7, 5)


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

C2RVine *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
`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, Eike Brechmann

See Also

`RVineMatrix`, `D2RVine`

Examples

```
# set up C-vine copula model with mixed pair-copulas
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)

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

# load package CDVine for comparison
library(CDVine)

# simulate a sample of size 500 from a 4-dimensional D-vine
type <- 1  # C-vine
simdata <- CDVineSim(500, family, par, par2, type)

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

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

contour.RVineMatrix *Plotting* RVineMatrix *objects.*

Description

There are two plotting generics for RVineMatrix objects. plot.RVineMatrix plots one or all trees of a given R-vine copula model. Edges can be labeld with information about the corresponding pair-copula. contour.RVineMatrix produces a matrix of contour plots (using plot.BiCop).

Usage

```
## S3 method for class 'RVineMatrix'
contour(x, tree = "ALL", xylim = NULL, cex.nums = 1,
  data = NULL, ...)

## S3 method for class 'RVineMatrix'
plot(x, tree = "ALL", type = 0, edge.labels = NULL,
  legend.pos = "bottomleft", interactive = FALSE, ...)
```

Arguments

x	RVineMatrix object.
tree	"ALL" or integer vector; specifies which trees are plotted.
xylim	numeric vector of length 2; sets xlim and ylim for the contours
cex.nums	numeric; expansion factor for font of the numbers.
data	a data matrix for creating kernel density contours of each pair.
...	Arguments passed to plot.network or plot.BiCop respectively.
type	integer; specifies how to make use of variable names: 0 = variable names are ignored, 1 = variable names are used to annotate vertices, 2 = uses numbers in plot and adds a legend for variable names.
edge.labels	character; either a vector of edge labels or one of the following: "family" = pair-copula family abbreviation (see BiCopName), "par" = pair-copula parameters, "tau" = pair-copula Kendall's tau (by conversion of parameters) "family-par" = pair-copula family and parameters "family-tau" = pair-copula family and Kendall's tau.
legend.pos	the x argument for legend.
interactive	logical; if TRUE, the user is asked to adjust the positioning of vertices with his mouse.

Details

If you want the contour boxes to be perfect squares, the plot height should be 1.25/length(tree)*(d - min(tree)) times the plot width.

Author(s)

Thomas Nagler, Nicole Barthel

See Also

RVineMatrix, plot.network, plot.BiCop, BiCopName, legend

Examples

```
## build vine model
strucmat <- matrix(c(3,   1, 2, 0, 2, 1, 0, 0, 1), 3, 3)
fammat   <- matrix(c(0,   1, 6, 0, 0, 3, 0, 0, 0), 3, 3)
parmat   <- matrix(c(0, 0.3, 3, 0, 0, 1, 0, 0, 0), 3, 3)
par2mat  <- matrix(c(0,   0, 0, 0, 0, 0, 0, 0, 0), 3, 3)
RVM  <- RVineMatrix(strucmat, fammat, parmat, par2mat)

# plot trees
## Not run: plot(RVM)

# show contour plots
contour(RVM)
```

---

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

```
# 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`, `C2RVine`

Examples

```
# set up D-vine copula model with mixed pair-copulas
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)

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

# load package CDVine for comparison
```

```
library(CDVine)

# simulate a sample of size 500 from a 4-dimensional D-vine
type <- 2  # D-vine
simdata <- CDVineSim(500, family, par, par2, type)

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

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

---

daxreturns *Major German Stocks*

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)
```

ddCopula *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.

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

```
joeBiCopula(param = 2)
```

Arguments

`param`	The parameter `param` defines the copula through `theta`.

Value

One of the respective Joe copula classes (`joeBiCopula`, `surJoeBiCopula`, `r90JoeBiCopula`, `r270JoeBiCopula`).

Author(s)

Benedikt Graeler

References

Joe, H., (1997). Multivariate Models and Dependence Concepts. Monogra. Stat. Appl. Probab. 73, London: Chapman and Hall.

See Also

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

Examples

```
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))
```

`joeBiCopula-class` *Classes* `"joeBiCopula"`, `"surJoeBiCopula"`, `"r90JoeBiCopula"` *and* `"r270JoeBiCopula"`

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`, `surJoeBiCopula`, `r90JoeBiCopula` and `r270JoeBiCopula`.

Author(s)

Benedikt Graeler

References

Joe, H., (1997). Multivariate Models and Dependence Concepts. Monogra. Stat. Appl. Probab. 73, London: Chapman and Hall.

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(11), margins = "norm")
```

Arguments

`x`	`copuladata` object.
`labels`	variable names/labels.
`...`	other graphical parameters (see `par`) or options passed to `BiCopKDE`.
`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`.
`method`	a character string indicating which correlation coefficients are computed. One of `"pearson"`, `"kendall"` (default), or `"spearman"`
`ccols`	colour to be used for the contour plots; default: `ccols = terrain.colors(30)`.
`margins`	character; margins for the contour plots. Options are: `"unif"` for the original copula density, `"norm"` for the transformed density with standard normal margins, `"exp"` with standard exponential margins, and `"flexp"` with flipped exponential margins.

Note

If the default panel functions are used

- `col` changes only the colour of the points in the scatter plot (`upper.panel`)
- `cex` changes only the magnification of the points in the scatter plot (`upper.panel`)

Author(s)

Tobias Erhardt

See Also

`pairs`, `as.copuladata`, `BiCopKDE`

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 = "flexp")

## 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)
  BiCopKDE(x, y,
           levels = c(0.01, 0.05, 0.1, 0.15, 0.2),
           margins = "exp",
           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 surface/perspective or contour plot. Optionally, the density can be coupled with standard normal margins (default for contour plots). Furthermore, a lambda-plot is available (cf., `BiCopLambda`).

Usage

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

## S3 method for class 'BiCop'
contour(x, margins = "norm", size = 100L, ...)
```

Arguments

`x`	`BiCop object.`
`type`	plot type; either `"surface"`, `"contour"`, 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"`; options are: `"unif"` for the original copula density, `"norm"` for the transformed density with standard normal margins, `"exp"` with standard exponential margins, and `"flexp"` with flipped exponential margins. 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 $sizexsize$ 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

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

## plots
plot(obj)  # surface plot of copula density
contour(obj)  # contour plot with standard normal margins
contour(obj, margins = "unif")  # contour plot of copula density
```

---

pobs *Pseudo-Observations*

Description

Compute the pseudo-observations for the given data matrix.

Arguments

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

Details

Given n realizations $\boldsymbol{x}_i = (x_{i1}, \dots, x_{id})^T$, $i \in \{1, \dots, n\}$ of a random vector $\boldsymbol{X}$, the pseudo-observations are defined via $u_{ij} = r_{ij}/(n+1)$ for $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, d\}$, where r_{ij} denotes the rank of x_{ij} among all x_{kj}, $k \in \{1, \dots, n\}$. 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.

Author(s)

Marius Hofert

Examples

```
## 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` *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

```
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 $\boldsymbol{\theta}$, then the Akaike Information Criterion (AIC) by Akaike (1973) is defined as

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

for observations $\boldsymbol{u} = (\boldsymbol{u}_1', ..., \boldsymbol{u}_N')'$.

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

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

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

Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In B. N. Petrov and F. Csaki (Eds.), Proceedings of the Second International Symposium on Information Theory Budapest, Akademiai Kiado, pp. 267-281.

Schwarz, G. E. (1978). Estimating the dimension of a model. Annals of Statistics 6 (2), 461-464.

See Also

RVineLogLik, RVineVuongTest, RVineClarkeTest

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 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 $\hat{\boldsymbol{\theta}}_1$ and $\hat{\boldsymbol{\theta}}_2$. The null hypothesis of statistical indistinguishability of the two models is

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

where $m_i := \log\left[\frac{c_1(\boldsymbol{u}_i|\hat{\boldsymbol{\theta}}_1)}{c_2(\boldsymbol{u}_i|\hat{\boldsymbol{\theta}}_2)}\right]$ for observations $\boldsymbol{u}_i,\ i = 1, ..., N$.

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 tets statistic

$$\texttt{statistic} := B = \sum_{i=1}^{N} \mathbf{1}_{(0,\infty)}(m_i),$$

where $\mathbf{1}$ 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 $Np = \frac{N}{2}$.

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

Clarke, K. A. (2007). A Simple Distribution-Free Test for Nonnested Model Selection. Political Analysis, 15, 347-363.

See Also

RVineVuongTest, RVineAIC, RVineBIC

Examples

```
# 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, se = FALSE,
  rotations = TRUE, method = "mle", cores = 1)
```

Arguments

`data`	N x d data matrix (with uniform margins).
`familyset`	integer vector of pair-copula families to select from. 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. If a vector of negative numbers is provided, selection among all but `abs(familyset)` is performed. Coding of pair copula families is the same as in `BiCop`.
`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), `"BIC"`, or `"logLik"` (see `BiCopSelect`).

indeptest	Logical; whether a hypothesis test for the independence of u1 and u2 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.
se	Logical; whether standard errors are estimated (default: se = FALSE).
rotations	logical; if TRUE, all rotations of the families in familyset are included.
method	indicates the estimation method: either maximum likelihood estimation (method = "mle"; default) or inversion of Kendall's tau (method = "itau"). For method = "itau" only one parameter families and the Student t copula can be used (family = 1,2,3,4,5,6,13,14,16,23,24,26,33,34 or 36). For the t-copula, par2 is found by a crude profile likelihood optimization over the interval (2, 10].
cores	integer; if cores > 1, estimation will be parallized within each tree (using foreach). Note that parallelization causes substantial overhead and may be slower than single-threaded computation when dimension, sample size, or familyset are small or method = "itau".

Details

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

Value

An RVineMatrix object with the selected families (RVM$family) as well as sequentially estimated parameters stored in RVM$par and RVM$par2. The object is augmented by the following information about the fit:

se, se2	standard errors for the parameter estimates (if se = TRUE; note that these are only approximate since they do not account for the sequential nature of the estimation,
nobs	number of observations,
logLik, pair.logLik	log likelihood (overall and pairwise)
AIC, pair.AIC	Aikaike's Informaton Criterion (overall and pairwise),
BIC, pair.BIC	Bayesian's Informaton Criterion (overall and pairwise),
emptau	matrix of empirical values of Kendall's tau,
p.value.indeptest	matrix of p-values of the independence test.

#'

Note

For a comprehensive summary of the vine copula model, use summary(object); to see all its contents, use str(object).

Author(s)

Eike Brechmann, Thomas Nagler

References

Brechmann, E. C., C. Czado, and K. Aas (2012). Truncated regular vines in high dimensions with applications to financial data. Canadian Journal of Statistics 40 (1), 68-85.

Dissmann, J. F., E. C. Brechmann, C. Czado, and D. Kurowicka (2013). Selecting and estimating regular vine copulae and application to financial returns. Computational Statistics & Data Analysis, 59 (1), 52-69.

See Also

RVineMatrix, BiCop, BiCopSelect, plot.RVineMatrix, contour.RVineMatrix, foreach

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 500 from the R-vine copula model
set.seed(123)
simdata <- RVineSim(500, RVM)
```

```
## determine the pair-copula families and parameters
RVM1 <- RVineCopSelect(simdata, familyset = c(1, 3, 4, 5 ,6), Matrix)

## see the object's content or a summary
str(RVM1)
summary(RVM1)

## inspect the fitted model using plots
## Not run: plot(RVM1)  # tree structure

contour(RVM1)  # contour plots of all pair-copulas
```

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 RVinePcor2ccor 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 inidicated 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

## 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

```
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 `RVM$family` 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 original 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

$$H_0 : \boldsymbol{H}(\theta) + \boldsymbol{C}(\theta) = 0$$

against the alternative

$$H_1 : \boldsymbol{H}(\theta) + \boldsymbol{C}(\theta) \neq 0,$$

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

$$H_0 : -\boldsymbol{H}(\theta)^{-1}\boldsymbol{C}(\theta) = I_p$$

against the alternative

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

The test statistic can then be calculated as

$$IR_n := tr(\Phi(\theta))/p$$

with $\Phi(\theta) = -\boldsymbol{H}(\theta)^{-1}\boldsymbol{C}(\theta)$, p is the number of parameters, i.e. the length of θ, 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 form the PIT are aggregated to univariate data by different aggregation functions $\Gamma(\cdot)$ in the sum

$$s_t = \sum_{i=1}^{d} \Gamma(y_{it}), t = 1, ..., n$$

. 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 $\Gamma(\cdot) = |\cdot -0.5|$ (`method="Berg"`) or $\Gamma(\cdot) = (\cdot - 0.5)^{\alpha}, \alpha = 2, 4, 6, ...$ (`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 $\Theta \subseteq R^p$ being the parameter space of dimension p. They are based on the empirical copula process (ECP)

$$\hat{C}_n(u) - C_{\hat{\theta}_n}(u),$$

with $u = (u_1, \ldots, u_d) \in [0,1]^d$ and $\hat{C}_n(u) = \frac{1}{n+1}\sum_{t=1}^n \mathbf{1}_{\{U_{t1} \leq u_1, \ldots, U_{td} \leq u_d\}}$. 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 = 0 or bootstrapped for B > 0

For method = "IR":

IR	test statistic (raw version as stated above)
p.value	So far no p-value is returned nigher a asymptotic nor a bootstrapped one. How to calculated a bootstrapped p-value is explained in Schepsmeier (2013). Be aware, that the test statistics than have to be adjusted with the empirical variance.

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 = 0 or bootstrapped for B > 0. 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

Warning: The code for all the p-values are not yet approved since some of them are moved from R-code to C-code. If you need p-values the best way is to write your own algorithm as suggested in Schepsmeier (2013) to get bootstrapped p-values.

Author(s)

Ulf Schepsmeier

References

Berg, D. and H. Bakken (2007) A copula goodness-of-fit apprach based on the conditional probability integral transformation. http://www.danielberg.no/publications/Btest.pdf

Breymann, W., A. Dias and P. Embrechts (2003) Dependence structures for multivariate high-frequence data in finance. Quantitative Finance 3, 1-14

Genest, C., B. Remillard, and D. Beaudoin (2009) Goodness-of-fit tests for copulas: a review and power study. Insur. Math. Econ. 44, 199-213.

Huang, w. and A. Prokhorov (2011). A goodness-of-fit test for copulas. to appear in Econometric Reviews

Schepsmeier, U. (2013) A goodness-of-fit test for regular vine copula models. Preprint http://arxiv.org/abs/1306.0818

Schepsmeier, U. (2015) Efficient information based goodness-of-fit tests for vine copula models with fixed margins. Journal of Multivariate Analysis 138, 34-52.

White, H. (1982) Maximum likelihood estimation of misspecified models, Econometrica, 50, 1-26.

See Also

BiCopGofTest, RVinePIT

Examples

```
## time-consuming example

# load data set
data(daxreturns)

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

# White test with asymptotic p-value
RVineGofTest(daxreturns[,1:5], RVM, B = 0)

# ECP2 test with Cramer-von-Mises test statistic and a bootstrap
# with 200 replications for the calculation of the p-value
RVineGofTest(daxreturns[,1:5], RVM, method = "ECP2",
             statistic = "CvM", B = 200)
```

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. (three matrices: `direct`, `indirect` and `value`).

Note

The gradient for R-vine copula models with two parameter Archimedean copulas, i.e. BB1, BB6, BB7, BB8 and their rotated versions can not yet be calculated. The derivatives of these bivariate copulas are more complicated.

Author(s)

Ulf Schepsmeier, Jakob Stoeber

References

Dissmann, J. F., E. C. Brechmann, C. Czado, and D. Kurowicka (2013). Selecting and estimating regular vine copulae and application to financial returns. Computational Statistics & Data Analysis, 59 (1), 52-69.

Schepsmeier, U. and J. Stoeber (2014) Derivatives and Fisher information of bivariate copulas. Statistical Papers, 55(2), 525-542. online first: http://link.springer.com/article/10.1007/s00362-013-0498-x.

Web supplement: Derivatives and Fisher Information of bivariate copulas. http://mediatum.ub.tum.de/node?id=1119201

Stoeber, J. and U. Schepsmeier (2013). Estimating standard errors in regular vine copula models. Computational Statistics, 28 (6), 2679-2707 http://link.springer.com/article/10.1007/s00180-013-0423-8#.

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
```

RVineHessian — *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) 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

Dissmann, J. F., E. C. Brechmann, C. Czado, and D. Kurowicka (2013). Selecting and estimating regular vine copulae and application to financial returns. Computational Statistics & Data Analysis, 59 (1), 52-69.

Schepsmeier, U. and J. Stoeber (2014) Derivatives and Fisher information of bivariate copulas. Statistical Papers, 55(2), 525-542. online first: http://link.springer.com/article/10.1007/s00362-013-0498-x.

Web supplement: Derivatives and Fisher Information of bivariate copulas. http://mediatum.ub.tum.de/node?id=1119201

Stoeber, J. and U. Schepsmeier (2013). Estimating standard errors in regular vine copula models. Computational Statistics, 28 (6), 2679-2707 http://link.springer.com/article/10.1007/s00180-013-0423-8#.

See Also

BiCopDeriv, BiCopDeriv2, BiCopHfuncDeriv, BiCopHfuncDeriv2, RVineMatrix, RVineMLE, 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)

# compute the Hessian matrix of the first row of the data
out2 <- RVineHessian(simdata[1,], RVM)
out2$hessian
```

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, check.pars = TRUE, calculate.V = 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.
`check.pars`	logical; default is `TRUE`; if `FALSE`, checks for family/parameter-consistency are ommited (should only be used with care).
`calculate.V`	logical; whether V matrices should be calculated. Default is `TRUE`, but requires a lot of memory when dimension is large. Use `FALSE` for a memory efficient version.

Details

For observations $\boldsymbol{u} = (\boldsymbol{u}'_1, ..., \boldsymbol{u}'_N)'$ 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

$$\texttt{loglik} := l_{RVine}(\boldsymbol{\theta}|\boldsymbol{u})$$

$$= \sum_{i=1}^{N} \sum_{\ell=1}^{d-1} \sum_{e \in E_\ell} \ln[c_{j(e),k(e)|D(e)}(F(u_{i,j(e)}|u_{i,D(e)}), F(u_{i,k(e)}|u_{i,D(e)})|\theta_{j(e),k(e)|D(e)})]$$

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

$$h(u|\boldsymbol{v}, \boldsymbol{\theta}) := F(u|\boldsymbol{v}) = dC_{uv_j|v_{-j}}(F(u|v_{-j}), F(v_j|v_{-j}))/dF(v_j|v_{-j}),$$

where $C_{uv_j|\boldsymbol{v}_{-j}}$ is a bivariate copula distribution function with parameter(s) $\boldsymbol{\theta}$ and $\boldsymbol{v}_{-j}$ 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

Dissmann, J. F., E. C. Brechmann, C. Czado, and D. Kurowicka (2013). Selecting and estimating regular vine copulae and application to financial returns. Computational Statistics & Data Analysis, 59 (1), 52-69.

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(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 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

```
RVineMatrix(Matrix, family = array(0, dim = dim(Matrix)), par = array(NA,
  dim = dim(Matrix)), par2 = array(NA, dim = dim(Matrix)), names = NULL,
  check.pars = TRUE)
```

Arguments

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

family Lower (or upper) triangular d x 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)
`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` 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: `par = array(NA, dim = dim(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: `par2 = array(NA, dim = dim(Matrix))`).

`names` A vector of names for the d variables; default: `names = NULL`.

`check.pars` logical; default is `TRUE`; if `FALSE`, checks for family/parameter-consistency are ommited (should only be used with care).

Value

An object of class `RVineMatrix`, i.e., a list with the following 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.

`names` variable names (defaults to `V1, V2, ...`).

`MaxMat, CondDistr`
additional matrices required internally for evaluating the density etc.,

`type` the type of the vine copula structure; possible types are:

- `"C-vine"`: all trees consist of a star,
- `"D-vine"`: all trees consist of a path,

- "R-vine": all strucutres that are neither a C- nor D-vine,

tau
: Kendall's tau matrix,

taildep
: matrices of lower and upper tail dependence coefficients,

beta
: Blomqvist's beta matrix.

Objects of this class are also returned by the RVineSeqEst, RVineCopSelect, and RVineStructureSelect functions. In this case, further information about the fit is added.

Note

For a comprehensive summary of the vine copula model, use summary(object); to see all its contents, use str(object).
The RVineMatrix function automatically checks if the given matrix is a valid R-vine matrix (see RVineMatrixCheck).
Although the function allows upper triangular matrices as its input, it will always store them as lower triangular matrices.

Author(s)

Jeffrey Dissmann, Thomas Nagler

References

Dissmann, J. F., E. C. Brechmann, C. Czado, and D. Kurowicka (2013). Selecting and estimating regular vine copulae and application to financial returns. Computational Statistics & Data Analysis, 59 (1), 52-69.

See Also

RVineMatrixCheck, RVineSeqEst, RVineCopSelect, RVineStructureSelect, RVineSim, C2RVine, D2RVine

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"))

## see the object's content or a summary
str(RVM)
summary(RVM)

## inspect the model using plots
## Not run: plot(RVM)  # tree structure
contour(RVM)  # contour plots of all pair-copulas

## simulate from the vine copula model
plot(RVineSim(500, RVM))
```

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.

Value

code	1 for OK; -4 matrix is neither lower nor upper triangular; -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

Joe H, Cooke RM and Kurowicka D (2011). Regular vines: generation algorithm and number of equivalence classes. In Dependence Modeling: Vine Copula Handbook, pp 219–231. World Scientific, Singapore.

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)
print(b1)
# 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)
print(b2)
# improper vine matrix, code=-2
A3 <- matrix(c(6, 0, 0, 0, 0, 0,
               3, 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)
b3 <- RVineMatrixCheck(A3)
print(b3)
```

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, 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)
```

RVineMatrixSample *Randomv sampling of R-Vine matices*

Description

Sample R-Vine matrices based on the algorithm of Joe et al. (2011).

Usage

```
RVineMatrixSample(d, size = 1, naturalOrder = FALSE)
```

Arguments

d	Dimension of the R-Vine matrices.
size	Number of matrices to sample.
naturalOrder	Should the matrices be in the natural order (default: `naturalOrder = FALSE`).

Value

A list of length `size` with each element containing one R-Vine matrix.

Note

For some reason, our implementaion of Joe et al.'s algorithm always returns a star in the first tree. To fix this, we sample a vine matrix of dimension d + 1 and remove the first tree afterwards

Author(s)

Thibault Vatter

References

Joe H, Cooke RM and Kurowicka D (2011). Regular vines: generation algorithm and number of equivalence classes. In Dependence Modeling: Vine Copula Handbook, pp 219–231. World Scientific, Singapore.

See Also

`RVineMatrix`, `RVineMatrixCheck`

Examples

```
# Matrix and sample sizes
d <- 10
size <- 5

# Sample R-vine matrices
RVM <- RVineMatrixSample(d, size)
```

```
sapply(RVM, RVineMatrixCheck)

# Sample R-vine matrices in the natural order
RVM <- RVineMatrixSample(d, size, naturalOrder = TRUE)
sapply(RVM, RVineMatrixCheck)
```

RVineMLE *Maximum Likelihood Estimation of an R-Vine Copula Model*

Description

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

Usage

```
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 RVM$family 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`. Additional information about the fit is added (e.g., log-likelihood, AIC, BIC).
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`).

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

Dissmann, J. F., E. C. Brechmann, C. Czado, and D. Kurowicka (2013). Selecting and estimating regular vine copulae and application to financial returns. Computational Statistics & Data Analysis, 59 (1), 52-69.

Stoeber, J. and U. Schepsmeier (2013). Estimating standard errors in regular vine copula models. Computational Statistics, 1-29 http://link.springer.com/article/10.1007/s00180-013-0423-8#.

See Also

`RVineSeqEst`, `RVineStructureSelect`, `RVineMatrix`, `RVineGrad`, `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 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, check.pars = TRUE)
```

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`).
- `check.pars` logical; default is `TRUE`; if `FALSE`, checks for family/parameter-consistency are ommited (should only be used with care).

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

```
# 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"))

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

RVinePar2Tau *Kendall's Tau Values of an R-Vine Copula Model*

Description

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

Usage

```
RVinePar2Tau(RVM, check.pars = TRUE)
```

Arguments

RVM	An RVineMatrix object.
check.pars	logical; default is TRUE; if FALSE, checks for family/parameter-consistency are ommited (should only be used with care).

Value

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

Author(s)

Jeffrey Dissmann

See Also

RVineMatrix, BiCopPar2Tau

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"))

# compute the Kendall's tau values
tau <- RVinePar2Tau(RVM)
```

`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, verbose = TRUE)
```

Arguments

newdata — An N x d data matrix that specifies where the density shall be evaluated.

RVM — An RVineMatrix object including the structure and the pair-copula families and parameters.

verbose — In case something goes wrong, additional output will be plotted.

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

$$\prod_{\ell=1}^{d-1} \prod_{e \in E_\ell} c_{j(e),k(e)|D(e)}(F(u_{j(e)}|u_{D(e)}), F(u_{k(e)}|u_{D(e)})|\theta_{j(e),k(e)|D(e)}),$$

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

$$h(u|\boldsymbol{v}, \boldsymbol{\theta}) := F(u|\boldsymbol{v}) = dC_{uv_j|v_{-j}}(F(u|v_{-j}), F(v_j|v_{-j}))/dF(v_j|v_{-j}),$$

where $C_{uv_j|\boldsymbol{v}_{-j}}$ is a bivariate copula distribution function with parameter(s) $\boldsymbol{\theta}$ and $\boldsymbol{v}_{-j}$ 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

Dissmann, J. F., E. C. Brechmann, C. Czado, and D. Kurowicka (2013). Selecting and estimating regular vine copulae and application to financial returns. Computational Statistics & Data Analysis, 59 (1), 52-69.

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(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"))

# 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

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

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 (2015).

Value

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

Author(s)

Ulf Schepsmeier

References

Rosenblatt, M. (1952). Remarks on a Multivariate Transformation. The Annals of Mathematical Statistics 23 (3), 470-472.

Schepsmeier, U. (2015) Efficient information based goodness-of-fit tests for vine copula models with fixed margins. Journal of Multivariate Analysis 138, 34-52.

See Also

`RVineGofTest`

Examples

```
# 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))
plot(daxreturns[,1], daxreturns[,2]) # correlated data
plot(pit[,1], pit[,2]) # i.i.d. data

cor(pit, method = "kendall")
```

RVineSeqEst *Sequential Estimation of an R-Vine Copula Model*

Description

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

Usage

```
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, cores = 1)
```

Arguments

`data`	An N x d data matrix (with uniform margins).
`RVM`	An RVineMatrix object including the structure, the pair-copula families and the pair-copula parameters (if they are known).
`method`	indicates the estimation method: either maximum likelihood estimation (`method = "mle"`; default) or inversion of Kendall's tau (`method = "itau"`). For `method = "itau"` only one parameter families and the Student t copula can be used (`family = 1,2,3,4,5,6,13,14,16,23,24,26,33,34 or 36`). For the t-copula, `par2` is found by a crude profile likelihood optimization over the interval (2, 10].
`se`	Logical; whether standard errors 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`; 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))`).
`progress`	Logical; whether the pairwise estimation progress is printed (default: `progress = FALSE`).
`weights`	Numerical; weights for each observation (opitional).
`cores`	integer; if `cores > 1`, estimation will be parallized within each tree (using foreach). However, the overhead caused by parallelization is likely to make the function run slower unless sample size is really large and `method = "itau"`.

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 BiCopEst.

Value

An RVineMatrix object with the sequentially estimated parameters stored in RVM$par and RVM$par2. The object is augmented by the following information about the fit:

- se, se2 — standard errors for the parameter estimates (if se = TRUE); note that these are only approximate since they do not account for the sequential nature of the estimation,
- nobs — number of observations,
- logLik, pair.logLik — log likelihood (overall and pairwise)
- AIC, pair.AIC — Aikaike's Informaton Criterion (overall and pairwise),
- BIC, pair.BIC — Bayesian's Informaton Criterion (overall and pairwise),
- emptau — matrix of empirical values of Kendall's tau,
- p.value.indeptest — matrix of p-values of the independence test.

Note

For a comprehensive summary of the fitted model, use summary(object); to see all its contents, use str(object).

Author(s)

Ulf Schepsmeier, Jeffrey Dissmann, Thomas Nagler

See Also

RVineMatrix, BiCop, BiCopEst, plot.RVineMatrix, contour.RVineMatrix, foreach

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)

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

Dissmann, J. F., E. C. Brechmann, C. Czado, and D. Kurowicka (2013). Selecting and estimating regular vine copulae and application to financial returns. Computational Statistics & Data Analysis, 59 (1), 52-69.

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, 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)
```

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 negaive 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

Dissmann, J. F., E. C. Brechmann, C. Czado, and D. Kurowicka (2013). Selecting and estimating regular vine copulae and application to financial returns. Computational Statistics & Data Analysis, 59 (1), 52-69.

Schepsmeier, U. and J. Stoeber (2014) Derivatives and Fisher information of bivariate copulas. Statistical Papers, 55(2), 525-542. online first: http://link.springer.com/article/10.1007/s00362-013-0498-x.

Web supplement: Derivatives and Fisher Information of bivariate copulas. http://mediatum.ub.tum.de/node?id=1119201

Stoeber, J. and U. Schepsmeier (2013). Estimating standard errors in regular vine copula models. Computational Statistics, 28 (6), 2679-2707 http://link.springer.com/article/10.1007/s00180-013-0423-8#.

See Also

BiCopDeriv, BiCopDeriv2, BiCopHfuncDeriv, BiCopHfuncDeriv2,
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)

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

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

RVineStructureSelect *Sequential Specification of R- and C-Vine Copula Models*

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

```
RVineStructureSelect(data, familyset = NA, type = 0,
  selectioncrit = "AIC", indeptest = FALSE, level = 0.05,
  trunclevel = NA, progress = FALSE, weights = NA, treecrit = "tau",
  se = FALSE, rotations = TRUE, method = "mle", cores = 1)
```

Arguments

data	An N x d data matrix (with uniform margins).
familyset	An integer vector of pair-copula families to select from. 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 is the same as in BiCop.
type	Type of the vine model to be specified: `0` or `"RVine"` = R-vine (default) `1` or `"CVine"` = C-vine C- and D-vine copula models with pre-specified order can be specified using `CDVineCopSelect` of the package CDVine. Similarly, R-vine copula models with pre-specified tree structure can be specified using RVineCopSelect.
selectioncrit	Character indicating the criterion for pair-copula selection. Possible choices:`selectioncrit = "AIC"` (default), `"BIC"`, or `"logLik"` (see BiCopSelect).
indeptest	logical; whether a hypothesis test for the independence of `u1` and `u2` 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	numeri; 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	numeric; weights for each observation (opitional).
treecrit	edge weight for Dissman's structure selection algorithm, see *Details*.
se	Logical; whether standard errors are estimated (default: `se = FALSE`).
rotations	If `TRUE`, all rotations of the families in `familyset` are included.

method — indicates the estimation method: either maximum likelihood estimation (`method = "mle"`; default) or inversion of Kendall's tau (`method = "itau"`). For `method = "itau"` only one parameter families and the Student t copula can be used (`family = 1,2,3,4,5,6,13,14,16,23,24,26,33,34 or 36`). For the t-copula, `par2` is found by a crude profile likelihood optimization over the interval (2, 10].

cores — integer; if `cores > 1`, estimation will be parallized within each tree (using `foreach`). Note that parallelization causes substantial overhead and may be slower than single-threaded computation when dimension, sample size, or familyset are small or `method = "itau"`.

Details

R-vine trees are selected using maximum spanning trees w.r.t. some edge weights. The most commonly used edge weigth is the absolute value of the empirical Kendall's tau, say $\hat{\tau}_{ij}$. Then, the following o ptimization problem is solved for each tree:

$$\max \sum_{\text{edges } e_{ij} \in \text{ in spanning tree}} |\hat{\tau}_{ij}|,$$

where 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.

Some commonly used edge weights are implemented:

`"tau"`	absolute value of empirical Kendall's tau.
`"rho"`	absolute value of empirical Spearman's rho.
`"AIC"`	Akaike information (multiplied by -1).
`"BIC"`	Bayesian information criterion (multiplied by -1).
`"cAIC"`	corrected Akaike information criterion (multiplied by -1).

If the data contain NAs, the edge weights in `"tau"` and `"rho"` are multiplied by the square root of the proportion of complete observations. This penalizes pairs where less observations are used.

The criteria `"AIC"`, `"BIC"`, and `"cAIC"` require estimation and model selection for all possible pairs. This is computationally expensive and much slower than `"tau"` or `"rho"`. The user can also specify a custom function to calculate the edge weights. The function has to be of type `function(u1, u2, weights) ...` and must return a numeric value. The weigths argument must exist, but does not has to be used. For example, `"tau"` (withouth using weights) can be implemented as follows:

```
function(u1, u2, weights)
abs(cor(u1, u2, method = "kendall", use = "complete.obs"))
```

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 $1 - |\hat{\tau_{ij}}|$. 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. The object is augmented by the following information about the fit:

- `se, se2` standard errors for the parameter estimates; note that these are only approximate since they do not account for the sequential nature of the estimation,
- `nobs` number of observations,
- `logLik, pair.logLik` log likelihood (overall and pairwise)
- `AIC, pair.AIC` Aikaike's Informaton Criterion (overall and pairwise),
- `BIC, pair.BIC` Bayesian's Informaton Criterion (overall and pairwise),
- `emptau` matrix of empirical values of Kendall's tau,
- `p.value.indeptest` matrix of p-values of the independence test.

Note

For a comprehensive summary of the vine copula model, use summary(object); to see all its contents, use str(object).

Author(s)

Jeffrey Dissmann, Eike Brechmann, Ulf Schepsmeier, Thomas Nagler

References

Brechmann, E. C., C. Czado, and K. Aas (2012). Truncated regular vines in high dimensions with applications to financial data. Canadian Journal of Statistics 40 (1), 68-85.

Dissmann, J. F., E. C. Brechmann, C. Czado, and D. Kurowicka (2013). Selecting and estimating regular vine copulae and application to financial returns. Computational Statistics & Data Analysis, 59 (1), 52-69.

See Also

RVineMatrix, BiCop, RVineCopSelect, plot.RVineMatrix, contour.RVineMatrix, foreach

Examples

```
# load data set
data(daxreturns)

# select the R-vine structure, families and parameters
# using only the first 4 variables and the first 250 observations
# we allow for the copula families: Gauss, t, Clayton, Gumbel, Frank and Joe
```

```
daxreturns <- daxreturns[1:250, 1:4]
RVM <- RVineStructureSelect(daxreturns, c(1:6), progress = TRUE)

## see the object's content or a summary
str(RVM)
summary(RVM)

## inspect the fitted model using plots
## Not run: plot(RVM)  # tree structure
contour(RVM)  # contour plots of all pair-copulas

## estimate a C-vine copula model with only Clayton, Gumbel and Frank copulas
CVM <- RVineStructureSelect(daxreturns, c(3,4,5), "CVine")

## determine the order of the nodes in a D-vine using the package TSP
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

Function is deprecated since `VineCopula 2.0`. Use `plot.RVineMatrix` instead.

Usage

```
RVineTreePlot(x, tree = "ALL", type = 0, edge.labels = NULL,
  legend.pos = "bottomleft", interactive = FALSE, ...)
```

Arguments

`x`	`RVineMatrix` object.
`tree`	`"ALL"` or integer vector; specifies which trees are plotted.
`type`	integer; specifies how to make use of variable names: `0` = variable names are ignored, 1 = variable names are used to annotate vertices, 2 = uses numbers in plot and adds a legend for variable names.
`edge.labels`	character; either a vector of edge labels or one of the following: `"family"` = pair-copula family abbreviation (see `BiCopName`), `"par"` = pair-copula parameters,

	"tau" = pair-copula Kendall's tau (by conversion of parameters) "family-par" = pair-copula family and parameters "family-tau" = pair-copula family and Kendall's tau.
legend.pos	the x argument for legend.
interactive	logical; if TRUE, the user is asked to adjust the positioning of vertices with his mouse.
...	Arguments passed to plot.network.

Author(s)

Thomas Nagler

See Also

plot.RVineMatrix

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 $\hat{\boldsymbol{\theta}}_1$ and $\hat{\boldsymbol{\theta}}_2$. We then compute the standardized sum, ν, of the log differences of their pointwise likelihoods $m_i := \log\left[\frac{c_1(\boldsymbol{u}_i|\hat{\boldsymbol{\theta}}_1)}{c_2(\boldsymbol{u}_i|\hat{\boldsymbol{\theta}}_2)}\right]$ for observations $\boldsymbol{u}_i \in [0,1],\ i=1,...,N$, i.e.,

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

Vuong (1989) shows that ν is asymptotically standard normal. According to the null-hypothesis

$$H_0 : E[m_i] = 0\ \forall i = 1, ..., N,$$

we hence prefer vine model 1 to vine model 2 at level α if

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

where Φ^{-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.

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

Vuong, Q. H. (1989). Ratio tests for model selection and non-nested hypotheses. Econometrica 57 (2), 307-333.

See Also

RVineClarkeTest, RVineAIC, RVineBIC

Examples

```
# 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)
vuong$statistic
vuong$statistic.Schwarz
vuong$p.value
```

```
vuong$p.value.Schwarz
```

---

surClaytonCopula   *Survival and Rotated Clayton Copulas*

Description

These are wrappers to functions from VineCopula-package

Usage

```
surClaytonCopula(param = 1)
```

Arguments

param	A single parameter defining the Copula.

Value

An object of class surClaytonCopula, r90ClaytonCopula or r270ClaytonCopula respectively.

Author(s)

Benedikt Graeler

Examples

```
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))
```

`surClaytonCopula-class` *Classes* `"surClaytonCopula"`, `"r90ClaytonCopula"` *and* `"r270ClaytonCopula"`

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.

Author(s)

Benedikt Graeler

See Also

`VineCopula-package`

Examples

```
library(copula)

persp(surClaytonCopula(.5),dCopula,zlim=c(0,10))
persp(r90ClaytonCopula(-.5),dCopula,zlim=c(0,10))
persp(r270ClaytonCopula(-.5),dCopula,zlim=c(0,10))
```

`surGumbelCopula` *Survival and Rotated Gumbel Copulas*

Description

These are wrappers to functions from `VineCopula-package`

Usage

```
surGumbelCopula(param = 1)
```

Arguments

`param`	A single parameter defining the Copula.

Value

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

Author(s)

Benedikt Graeler

Examples

```
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))
```

---

surGumbelCopula-class *Classes* "surGumbelCopula", "r90GumbelCopula" *and* "r270GumbelCopula"

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.

Author(s)

Benedikt Graeler

See Also

`VineCopula-package`

Examples

```
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))
```

TauMatrix *Matrix of Empirical Kendall's Tau Values*

Description

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

Usage

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

Arguments

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

Value

Matrix of the empirical Kendall's taus.

Author(s)

Ulf Schepsmeier

References

Knight, W. R. (1966). A computer method for calculating Kendall's tau with ungrouped data. Journal of the American Statistical Association 61 (314), 436-439.

See Also

`BiCopTau2Par`, `BiCopPar2Tau`, `BiCopEst`

Examples

```
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

```
tawnT1Copula(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.

Examples

```
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))
```

`tawnT1Copula-class` *Class* `"tawnT1Copula"`

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.

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))
```

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 *Class* "tawnT2Copula"

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.

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 Gaussian 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

Aas, K., C. Czado, A. Frigessi, and H. Bakken (2009). Pair-copula constructions of multiple dependence Insurance: Mathematics and Economics 44 (2), 182-198.

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)))
```

vineCopula-class *Class* "vineCopula"

Description

A class representing vine copulas in a object oriented implementations. 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`.

Author(s)

Benedikt Graeler

References

Aas, K., C. Czado, A. Frigessi, and H. Bakken (2009). Pair-copula constructions of multiple dependence Insurance: Mathematics and Economics 44 (2), 182-198.

See Also

`RVineMatrix` from package `VineCopula-package`

Examples

```
showClass("vineCopula")
```

Index