The copula Package

February 16, 2008

Version 0.6-6

Date 2008/01/22

Title Multivariate Dependence with Copula

Author Jun Yan <jyan@stat.uconn.edu> and Ivan Kojadinovic <ivan@stat.auckland.ac.nz>.

Maintainer Jun Yan <jyan@stat.uconn.edu>

Depends methods, mvtnorm, scatterplot3d, sn, adapt

Enhances nor1mix

Description Classes (S4) of commonly used copulas including elliptical, Archimidean, extreme value and Farlie-Gumbel-Morgenstern families. Methods for density, distribution, random number generators, bivariate association measures, persp, and contour. Functions for fitting copula models. Independence tests among random variables and random vectors based on the empirical copula process. Serial independence tests for univariate and multivariate continuous time series based on the empirical copula process.

License GPL (>= 3)

R topics documented:

Copula 2
Mvdc 3
archmCopula-class 5
archmCopula 6
AssocMeasures 7
contour-methods 7
copula-class 8
ellipCopula-class 9
ellipCopula 10
empcopm.test 11
empcops.test 13
empcopsm.test 15
empcopu.test 17

evCopula-class 20
evCopula 21
fgmCopula-class 22
fgmCopula 23
fitCopula-class 24
fitCopula 24
generator 26
mvdc-class 26
persp-methods 27
plackettCopula 27
show-methods 28
summary-methods 28

Index **29**

`Copula` *The Copula Distribution*

Description

Density, distribution function, and random generation for a `"copula"` object.

Usage

```
dcopula(copula, u)
pcopula(copula, u)
rcopula(copula, n)
```

Arguments

`copula`	a `"copula"` object.
`u`	a vector of the copula dimension or a matrix with number of rows being the copula dimension, giving the coordinates of the points where the density of distribution function need to be evaluated.
`n`	number of observations to be generated.

Details

The density function of an Archimedean copula is obtained by differentiating the distribution function symbolically using D.

The distribution function of a t copula uses pmvt from package mvtnorm. The density function of a t copula uses the dmst from package sn.

The random number generator for an Archiimedean copula uses the conditional approach for bivariate case and the Marshal-Olkin (1988) approach for dimension greater than 2.

Value

'dcopula' gives the density, 'pcopula' gives the distribution function, and 'rcopula' generates random variates.

Author(s)

Jun Yan <jyan@stat.uconn.edu>

References

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

Nelsen (1999), *An introduction to Copulas*, Springer, New York.

See Also

`copula-class`, `ellipCopula`, `archmCopula`, `fgmCopula`.

Examples

```
norm.cop <- normalCopula(0.5)
norm.cop
x <- rcopula(norm.cop, 100)
plot(x)
dcopula(norm.cop, x)
pcopula(norm.cop, x)
persp(norm.cop, dcopula)
contour(norm.cop, pcopula)
## a 3-dimensional normal copula
u <- rcopula(normalCopula(0.5, dim = 3), 1000)
## scatterplot3d(u)
## a 3-dimensional clayton copula
v <- rcopula(claytonCopula(2, dim = 3), 1000)
## scatterplot3d(v)
```

Mvdc *Multivariate Distribution via Copula*

Description

Density, distribution function, and random generator for a multivariate distribution via copula.

Usage

```
mvdc(copula, margins, paramMargins)
dmvdc(mvdc, x)
pmvdc(mvdc, x)
rmvdc(mvdc, n)
```

Arguments

copula	an object of copula.
margins	a character vector specifying all the marginal distributions. See details below.
paramMargins	a list of list with names components, giving the parameter values of the marginal distributions. See details below.
mvdc	a mvdc object.
x	a vector of the copula dimension or a matrix with number of rows being the copula dimension, giving the coordinates of the points where the density of distribution function need to be evaluated.
n	number of observations to be generated.

Details

The characters in argument margins are used to construct function names of density, distribution, and quantile functions. For example, "norm" can be used to specify marginal distribution, because "dnorm", "pnorm", and "qnorm" are all available.

Each component list in argument paramMargins is a list with named component which are used to specify the parameters of the marginal distributions. For example, paramMargins = list(list(mean = 0, sd = 2), list(rate = 2)) can be used to specify that the first margin is normal with mean 0 and sd 2, and the second margin is exponential with rate 2.

Value

'mvdc' constructs an object of class "mvdc". 'dmvdc' gives the density, 'pmvdc' gives the distribution function, and 'rmvdc' generates random variates.

Author(s)

Jun Yan <jyan@stat.uconn.edu>

See Also

ellipCopula, archmCopula, mvdc-class, copula-class

Examples

```
## construct a bivariate distribution whose marginals
## are normal and exponential respectively, coupled
## together via a normal copula
x <- mvdc(normalCopula(0.75), c("norm", "exp"),
          list(list(mean = 0, sd =2), list(rate = 2)))
x.samp <- rmvdc(x, 100)
dmvdc(x, x.samp)
pmvdc(x, x.samp)
persp(x, dmvdc, xlim = c(-4, 4), ylim=c(0, 1))
```

`archmCopula-class` *Class "archmCopula"*

Description

Archimedean copula

Objects from the Class

Objects can be created by calls of the form `new("archmCopula", ...)` or by function 'archmCopula'.

Slots

`exprdist`: Object of class `"expression"`, expressions for the cdf and pdf of the copula. These expressions are used in function 'pcopula' and 'dcopula'.

`dimension`: Object of class `"numeric"`

`parameters`: Object of class `"numeric"`

`param.names`: Object of class `"character"`

`param.lowbnd`: Object of class `"numeric"`

`param.upbnd`: Object of class `"numeric"`

`message`: Object of class `"character"`

Methods

dcopula `signature(copula = "claytonCopula")`: ...

pcopula `signature(copula = "claytonCopula")`: ...

rcopula `signature(copula = "claytonCopula")`: ...

dcopula `signature(copula = "frankCopula")`: ...

pcopula `signature(copula = "frankCopula")`: ...

rcopula `signature(copula = "frankCopula")`: ...

dcopula `signature(copula = "gumbelCopula")`: ...

pcopula `signature(copula = "gumbelCopula")`: ...

rcopula `signature(copula = "gumbelCopula")`: ...

dcopula `signature(copula = "amhCopula")`: ...

pcopula `signature(copula = "amhCopula")`: ...

rcopula `signature(copula = "amhCopula")`: ...

Extends

Class `"archmCopula"` extends class `"copula"` directly. Class `"claytonCopula"`, `"frankCopula"`, `"gumbelCopula"` and `"amhCopula"` extends class `"archmCopula"` directly.

Note

The expressions of pdf are obtained by differentiating the cdf expression using function 'D'.

"gumbelCopula" is also of class "evCopula".

Author(s)

Jun Yan <jyan@stat.uconn.edu>

See Also

archmCopula, copula-class.

---

archmCopula *Construction of Archimedean Copula Class Object*

Description

Constructs an Archimedean copula class object with its corresponding parameter and dimension.

Usage

```
archmCopula(family, param, dim = 2, ...)
claytonCopula(param, dim = 2)
frankCopula(param, dim = 2)
gumbelCopula(param, dim = 2)
amhCopula(param, dim = 2)
```

Arguments

family	a character string specifying the family of an Archimedean copula. Implemented families are "clayton", "frank", and "gumbel".
param	a numeric vector specifying the parameter values.
dim	the dimension of the copula.
...	currently nothing.

Value

An Archimedean copula object of class "claytonCopula", "frankCopula", "gumbelCopula", or "amhCopula".

Note

"archmCopula" is a wrapper for "claytonCopula", "frankCopula", "gumbelCopula" and "amhCopula".

For dim > 6, the expression of pdf is not available due to intensive compting involved in symbolically differentiating the cdf. Therefore, for dim > 6, likelihood estimation cannot be done yet.

Author(s)

Jun Yan <jyan@stat.uconn.edu>

See Also

ellipCopula

Examples

```
clayton.cop <- claytonCopula(2, dim = 3)
##scatterplot3d(rcopula(clayton.cop, 1000))
frank.cop <- frankCopula(3)
persp(frank.cop, dcopula)
gumbel.cop <- archmCopula("gumbel", 5)
contour(gumbel.cop, dcopula)
```

AssocMeasures *Association Measures in Package 'copula'*

Description

Methods to compute association measures such as Kendall's Tau, Spearman's Rho, and tail dependence index for bivaraite copulas. Calibration is supplied for Kendall's Tau and Spearman's Rho.

Methods

copula = "copula" Association measure for a "copula" object.

---

contour-methods *Methods for Function contour in Package 'copula'*

Description

Methods for function contour in package **copula**

Details

When x is of class "copula", these arguments can be supplied: fun: the function to be plotted, "dcopula" or "pcopula". n = 51: the number of points to do the plotting. theta = -30, phi = 30, expand = 0.618: arguments for "contour"

when x is of class "mvdc", these arguments are expected to replace the effect of n = 51: xlim: the range of the x variable. ylim: the range of the y variable. nx: the number of points for x to expand. ny: the number of points for y to expand.

Methods

x = "copula" Contour plot for a "copula" object.

x = "mvdc" Contour plot for a "mvdc" object.

Examples

```
contour(frankCopula(-0.8), dcopula)
contour(claytonCopula(2), pcopula)
x <- mvdc(gumbelCopula(3), c("norm", "norm"),
          list(list(mean = 0, sd =1), list(mean = 1)))
contour(x, dmvdc, xlim=c(-2, 2), ylim=c(-1, 3))
contour(x, pmvdc, xlim=c(-2, 2), ylim=c(-1, 3))
```

---

copula-class *Class "copula"*

Description

A class of multivariate distribution with uniform margins.

Objects from the Class

Objects can be created by calls of the form `new("copula", ...)`.

Slots

dimension: Object of class "numeric", dimension of the copula.

parameters: Object of class "numeric", parameter values.

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

param.lowbnd: Object of class "numeric", parameter lower bounds.

param.upbnd: Object of class "numeric", parameter upper bounds.

message: Object of class "character", families of the copula.

Warning

This implementation is still at the experimental stage and is subject to change during the development.

Note

The 'copula' class is extended by the 'evCopula', 'archmCopula' and 'ellipCopula' classes. Objects of implemented copulas can be created from functions 'evCopula', 'archmCopula' and 'ellipcopula'.

"plackettCopula" is a special type of copula which does not belong to either one of the three classes above.

Author(s)

Jun Yan <jyan@stat.uconn.edu>

See Also

archmCopula-class, ellipCopula-class, evCopula-class, fgmCopula-class

---

ellipCopula-class *Class "ellipCopula"*

Description

Copulas generated from elliptical multivariate distributions.

Objects from the Class

Objects can be created by calls of the form `new("ellipCopula", ...)`, or by function 'ellipCopula'.

Slots

dispstr: Object of class `"character"`, indicating the type of the dispersion matrix such as 'ex', 'ar1', 'toep', or 'un'.

dimension: Object of class `"numeric"`

parameters: Object of class `"numeric"`

param.names: Object of class `"character"`

param.lowbnd: Object of class `"numeric"`

param.upbnd: Object of class `"numeric"`

message: Object of class `"character"`

Extends

Class `"ellipCopula"` extends class `"copula"` directly. Class `"normalCopula"` and `"tCopula"` extends class `"ellipCopula"` directly.

Methods

dcopula `signature(copula = "normalCopula")`: ...

pcopula `signature(copula = "normalCopula")`: ...

rcopula `signature(copula = "normalCopula")`: ...

dcopula `signature(copula = "tCopula")`: ...

pcopula `signature(copula = "tCopula")`: ...

rcopula `signature(copula = "tCopula")`: ...

Author(s)

Jun Yan <jyan@stat.uconn.edu>

See Also

ellipCopula, copula-class.

`ellipCopula` *Construction of Elliptical Copula Class Object*

Description

Constructs an elliptical copula class object with its corresponding parameter and dimension.

Usage

```
ellipCopula(family, param, dim = 2, dispstr = "ex", df = 5, ...)
normalCopula(param, dim = 2, dispstr = "ex")
tCopula(param, dim = 2, dispstr = "ex", df = 5)
```

Arguments

`family`	a character string specifying the family of an elliptical copula. Implemented families are "normal" and "t".
`param`	a numeric vector specifying the parameter values.
`dim`	the dimension of the copula.
`dispstr`	a character string specifying the type of the symmetric positive definite matrix characterizing the elliptical copula. Implemented structures are "ex" for exchangeable, "ar1" for AR(1), "toep" for toeplitz, , and "un" for unstructured. For normal copula, this defines the structure of the correlation matrix.
`df`	a numerical value specifying the degree of freedom for the multivariate t distribution used to construct the t copulas.
`...`	currently nothing.

Value

An elliptical copula object of class `"normalCopula"` or `"tCopula"`.

Note

`"ellipCopula"` is a wrapper for `"normalCopula"` and `"tCopula"`.

Author(s)

Jun Yan <jyan@stat.uconn.edu>

See Also

archmCopula

Examples

```
norm.cop <- normalCopula(c(0.5, 0.6, 0.7), dim = 3, dispstr = "un")
t.cop <- tCopula(c(0.5, 0.3), dim = 3, dispstr = "toep", df = 2)
## from the wrapper
norm.cop <- ellipCopula("normal", param = c(0.5, 0.6, 0.7),
                        dim = 3, dispstr = "un")
## 3d scatter plot of 1000 random observations
##scatterplot3d(rcopula(norm.cop, 1000))
##scatterplot3d(rcopula(t.cop, 1000))
```

---

empcopm.test *Independence test among continuous random vectors based on the empirical copula process*

Description

Analog of the independence test based on the empirical copula process proposed by Christian Genest and Bruno Rémillard (see empcopu.test) for *random vectors*. The main difference comes from the fact that critical values and p-values are obtainted through the bootstrap/permutation methodology, since, here, test statistics are not distribution-free.

Usage

```
empcopm.test(x, d, m=length(d), N=1000, alpha=0.05)
```

Arguments

x	Data frame or data matrix containing realizations (one per line) of the random vectors whose independence is to be tested.
d	Dimensions of the random vectors whose realizations are given in x. It is required that sum(d)=ncol(x).
m	Maximum cardinality of the subsets of random vectors for which a test statistic is to be computed. It makes sense to consider m << p especially when p is large.
N	Number of bootstrap/permutation samples.
alpha	Significance level used in the computation of the critical values for the test statistics.

Details

See the references below for more details, especially the last one.

Value

The function `empcopm.test` returns an object of class `empcop.test` whose attributes are: `subsets`, `statistics`, `critical.values`, `pvalues`, `fisher.pvalue` (a p-value resulting from a combination *à la* Fisher of the subset statistic p-values), `tippett.pvalue` (a p-value resulting from a combination *à la* Tippett of the subset statistic p-values), `alpha` (global significance level of the test), `beta` (1 - `beta` is the significance level per statistic), `global.statistic` (value of the global Cramér-von Mises statistic derived directly from the independence empirical copula process - see `In` in the last reference) and `global.statistic.pvalue` (corresponding p-value).

Author(s)

Ivan Kojadinovic, ⟨ivan@stat.auckland.ac.nz⟩

References

P. Deheuvels (1979), *La fonction de dépendance empirique et ses propriétés: un test non paramétrique d'indépendance*, Acad. Roy. Belg. Bull. Cl. Sci. 5th Ser. 65, 274-292.

P. Deheuvels (1981), *A non parametric test for independence*, Publ. Inst. Statist. Univ. Paris 26, 29-50.

C. Genest and B. Rémillard (2004), *Tests of independence and randomness based on the empirical copula process*, Test 13, 335-369.

C. Genest, J.-F. Quessy and B. Rémillard (2006), *Local efficiency of a Cramer-von Mises test of independence*, Journal of Multivariate Analysis 97, 274-294.

C. Genest, J.-F. Quessy and B. Rémillard (2007), *Asymptotic local efficiency of Cramér-von Mises tests for multivariate independence*, The Annals of Statistics 35, 166-191.

I. Kojadinovic and M. Holmes (2008), *Tests of independence among continuous random vectors based on Cramér-von Mises functionals of the empirical copula process*, submitted.

See Also

`empcopu.test`,`empcops.test`,`empcopsm.test`,`dependogram`

Examples

```
## Consider the following example taken from
## Kojadinovic and Holmes (2007):

n <- 100

## Generate data
y <- matrix(rnorm(6*n),n,6)
y[,1] <- y[,2]/2 + sqrt(3)/2*y[,1]
y[,3] <- y[,4]/2 + sqrt(3)/2*y[,3]
y[,5] <- y[,6]/2 + sqrt(3)/2*y[,5]

nc <- normalCopula(0.3,dim=3)
x <- cbind(y,rcopula(nc,n),rcopula(nc,n))
```

```
x[,1] <- abs(x[,1]) * sign(x[,3] * x[,5])
x[,2] <- abs(x[,2]) * sign(x[,3] * x[,5])
x[,7] <- x[,7] + x[,10]
x[,8] <- x[,8] + x[,11]
x[,9] <- x[,9] + x[,12]

## Dimensions of the random vectors
d <- c(2,2,2,3,3)

## Run the test
test <- empcopm.test(x,d)
test

## Display the dependogram
dependogram(test)
```

`empcops.test` *Serial independence test for continuous time series based on the empirical copula process*

Description

Serial independence test based on the empirical copula process as proposed in Ghoudi et al. (2001) and Genest and Rémillard (2004). The test, which is the serial analog of `empcopu.test`, can be seen as composed of three steps: (i) a simulation step, which consists in simulating the distribution of the test statistics under serial independence for the sample size under consideration; (ii) the test itself, which consists in computing the approximate p-values of the test statistics with respect to the empirical distributions obtained in step (i); and (iii) the display of a graphic, called a *dependogram*, enabling to understand the type of departure from serial independence, if any. More details can be found in the articles cited in the reference section.

Usage

```
empcops.simulate(n, lag.max, m=lag.max+1, N=1000)
empcops.test(x, d, alpha=0.05)
```

Arguments

`n`	Length of the time series when simulating the distribution of the test statistics under serial independence.
`lag.max`	Maximum lag.
`m`	Maximum cardinality of the subsets of 'lags' for which a test statistic is to be computed. It makes sense to consider `m << lag.max+1` especially when `lag.max` is large.
`N`	Number of repetitions when simulating under serial independence.
`x`	Numeric vector containing the time series whose serial independence is to be tested.

d	Object of class empcops.distribution as returned by the function empcops.simulate. It can be regarded as the empirical distribution of the test statistics under serial independence.
alpha	Significance level used in the computation of the critical values for the test statistics.

Details

See the references below for more details, especially the third and fourth ones.

Value

The function empcops.simulate returns an object of class empcops.distribution whose attributes are: sample.size, lag.max, max.card.subsets, number.repetitons, subsets (list of the subsets for which test statistics have been computed), subsets.binary (subsets in binary 'integer' notation), dist.statistics.independence (a N line matrix containing the values of the test statistics for each subset and each repetition) and dist.global.statistic.independence (a vector a length N containing the values of the serial version of the global Cramér-von Mises test statistic for each repetition - see last reference p 175).

The function empcops.test returns an object of class empcop.test whose attributes are: subsets, statistics, critical.values, pvalues, fisher.pvalue (a p-value resulting from a combination *à la* Fisher of the subset statistic p-values), tippett.pvalue (a p-value resulting from a combination *à la* Tippett of the subset statistic p-values), alpha (global significance level of the test), beta (1 - beta is the significance level per statistic), global.statistic (value of the global Cramér-von Mises statistic derived directly from the serial independence empirical copula process - see last reference p 175) and global.statistic.pvalue (corresponding p-value).

Author(s)

Ivan Kojadinovic, ⟨ivan@stat.auckland.ac.nz⟩

References

P. Deheuvels (1979), *La fonction de dépendance empirique et ses propriétés: un test non paramétrique d'indépendance*, Acad. Roy. Belg. Bull. Cl. Sci. 5th Ser. 65, 274-292.

P. Deheuvels (1981), *A non parametric test for independence*, Publ. Inst. Statist. Univ. Paris 26, 29-50.

K. Ghoudi, R. Kulperger, and B. Rémillard (2001), *A nonparametric test of serial independence for times series and residuals*, Journal of Multivariate Analysis, 79:191-218.

C. Genest and B. Rémillard (2004), *Tests of independence and randomness based on the empirical copula process*, Test 13, 335-369.

C. Genest, J.-F. Quessy and B. Rémillard (2007), *Asymptotic local efficiency of Cramér-von Mises tests for multivariate independence*, The Annals of Statistics 35, 166-191.

See Also

empcopu.test,empcopm.test,empcopsm.test,dependogram

Examples

```
## AR 1 process

ar <-  numeric(200)
ar[1] <- rnorm(1)
for (i in 2:200)
  ar[i] <- 0.5 * ar[i-1] + rnorm(1)
x <- ar[101:200]

## In order to test for serial independence, the first step consists
## in simulating the distribution of the test statistics under
## serial independence for the same sample size, i.e. n=100.
## As we are going to consider lags up to 3, i.e., subsets of
## {1,...,4} whose cardinality is between 2 and 4 containing {1},
## we set lag.max=3. This may take a while...

d <- empcops.simulate(100,3)

## The next step consists in performing the test itself:

test <- empcops.test(x,d)

## Let us see the results:

test

## Display the dependogram:

dependogram(test)

## NB: In order to save d for future use, the save function can be used.
```

`empcopsm.test` *Serial independence test for multivariate continuous time series based on the empirical copula process*

Description

Analog of the serial independence test based on the empirical copula process proposed by Christian Genest and Bruno Rémillard (see `empcops.test`) for *multivariate* time series. The main difference comes from the fact that critical values and p-values are obtainted through the bootstrap/permutation methodology, since, here, test statistics are not distribution-free.

Usage

```
empcopsm.test(x, lag.max, m=lag.max+1, N=1000, alpha=0.05)
```

Arguments

x	Data frame or data matrix containing realizations the multivaraite continuous time series whose serial independence is to be tested.
lag.max	Maximum lag.
m	Maximum cardinality of the subsets of 'lags' for which a test statistic is to be computed. It makes sense to consider m << lag.max+1 especially when lag.max is large.
N	Number of bootstrap/permutation samples.
alpha	Significance level used in the computation of the critical values for the test statistics.

Details

See the references below for more details, especially the last one.

Value

The function empcopsm.test returns an object of class empcop.test whose attributes are: subsets, statistics, critical.values, pvalues, fisher.pvalue (a p-value resulting from a combination *à la* Fisher of the subset statistic p-values), tippett.pvalue (a p-value resulting from a combination *à la* Tippett of the subset statistic p-values), alpha (global significance level of the test), beta (1 - beta is the significance level per statistic), global.statistic (value of the global Cramér-von Mises statistic derived directly from the independence empirical copula process - see In in the last reference) and global.statistic.pvalue (corresponding p-value).

Author(s)

Ivan Kojadinovic, ⟨ivan@stat.auckland.ac.nz⟩

References

P. Deheuvels (1979), *La fonction de dépendance empirique et ses propriétés: un test non paramétrique d'indépendance*, Acad. Roy. Belg. Bull. Cl. Sci. 5th Ser. 65, 274-292.

P. Deheuvels (1981), *A non parametric test for independence*, Publ. Inst. Statist. Univ. Paris 26, 29-50.

K. Ghoudi, R. Kulperger, and B. Rémillard (2001), *A nonparametric test of serial independence for times series and residuals*, Journal of Multivariate Analysis, 79:191-218.

C. Genest and B. Rémillard (2004), *Tests of independence and randomness based on the empirical copula process*, Test 13, 335-369.

I. Kojadinovic and J. Yan (2008), *Tests of multivariate serial independence based on a Möbius decomposition of the independence empirical copula process*, submitted.

See Also

empcops.test,empcopu.test,empcopm.test,dependogram

Examples

```
## A multivariate time series
d <- 2
n <- 100
param <- 0.25
ar <- matrix(0,2*n,d)
ar[1,] <- rnorm(d)
for (i in 2:(2*n))
  ar[i,] <- matrix(param,d,d) %*% ar[i-1,] + rnorm(d)
x <- ar[(n+1):(2*n),]

## Run the test
test <- empcopsm.test(x,3)
test

## Display the dependogram
dependogram(test)
```

`empcopu.test` *Independence test among continuous random variables based on the empirical copula process*

Description

Multivariate independence test based on the empirical copula process as proposed by Christian Genest and Bruno Rémillard. The test can be seen as composed of three steps: (i) a simulation step, which consists in simulating the distribution of the test statistics under independence for the sample size under consideration; (ii) the test itself, which consists in computing the approximate p-values of the test statistics with respect to the empirical distributions obtained in step (i); and (iii) the display of a graphic, called a *dependogram*, enabling to understand the type of departure from independence, if any. More details can be found in the articles cited in the reference section.

Usage

```
empcopu.simulate(n, p, m = p, N = 1000)
empcopu.test(x, d, alpha=0.05)
dependogram(test, pvalues = FALSE)
```

Arguments

`n`	Sample size when simulating the distribution of the test statistics under independence.
`p`	Dimension of the data when simulating the distribution of the test statistics under independence.
`m`	Maximum cardinality of the subsets of variables for which a test statistic is to be computed. It makes sense to consider `m << p` especially when `p` is large.
`N`	Number of repetitions when simulating under independence.

x	Data frame or data matrix containing realizations (one per line) of the random vector whose independence is to be tested.
d	Object of class empcopu.distribution as returned by the function empcopu.simulate. It can be regarded as the empirical distribution of the test statistics under independence.
alpha	Significance level used in the computation of the critical values for the test statistics.
test	Object of class empcop.test as return by the function empcopu.test.
pvalues	Logical indicating whether the dependogram should be drew from test statistics or the corresponding p-values.

Details

See the references below for more details, especially the third one.

Value

The function empcopu.simulate returns an object of class empcopu.distribution whose attributes are: sample.size, data.dimension, max.card.subsets, number.repetitons, subsets (list of the subsets for which test statistics have been computed), subsets.binary (subsets in binary 'integer' notation), dist.statistics.independence (a N line matrix containing the values of the test statistics for each subset and each repetition) and dist.global.statistic.indepen (a vector a length N containing the values of the global Cramér-von Mises test statistic for each repetition - see last reference p 175).

The function empcopu.test returns an object of class empcop.test whose attributes are: subsets, statistics, critical.values, pvalues, fisher.pvalue (a p-value resulting from a combination *à la* Fisher of the subset statistic p-values), tippett.pvalue (a p-value resulting from a combination *à la* Tippett of the subset statistic p-values), alpha (global significance level of the test), beta (1 - beta is the significance level per statistic), global.statistic (value of the global Cramér-von Mises statistic derived directly from the independence empirical copula process - see last reference p 175) and global.statistic.pvalue (corresponding p-value).

Author(s)

Ivan Kojadinovic, ⟨ivan@stat.auckland.ac.nz⟩

References

P. Deheuvels (1979), *La fonction de dépendance empirique et ses propriétés: un test non paramétrique d'indépendance*, Acad. Roy. Belg. Bull. Cl. Sci. 5th Ser. 65, 274-292.

P. Deheuvels (1981), *A non parametric test for independence*, Publ. Inst. Statist. Univ. Paris 26, 29-50.

C. Genest and B. Rémillard (2004), *Tests of independence and randomness based on the empirical copula process*, Test 13, 335-369.

C. Genest, J.-F. Quessy and B. Rémillard (2006), *Local efficiency of a Cramer-von Mises test of independence*, Journal of Multivariate Analysis 97, 274-294.

C. Genest, J.-F. Quessy and B. Rémillard (2007), *Asymptotic local efficiency of Cramér-von Mises tests for multivariate independence*, The Annals of Statistics 35, 166-191.

See Also

empcops.test,empcopm.test,empcopsm.test

Examples

```
## Consider the following example taken from
## Genest and Remillard (2004), p 352:

x <- matrix(rnorm(500),100,5)
x[,1] <- abs(x[,1]) * sign(x[,2] * x[,3])
x[,5] <- x[,4]/2 + sqrt(3) * x[,5]/2

## In order to test for independence "within" x, the first step consists
## in simulating the distribution of the test statistics under
## independence for the same sample size and dimension,
## i.e. n=100 and p=5. As we are going to consider all the subsets of
## {1,...,5} whose cardinality is between 2 and 5, we set p=m=5.
## This may take a while...

d <- empcopu.simulate(100,5)

## The next step consists in performing the test itself:

test <- empcopu.test(x,d)

## Let us see the results:

test

## Display the dependogram:

dependogram(test)

## We could have tested for a weaker form of independence, for instance,
## by only computing statistics for subsets whose cardinality is between 2
## and 3. Consider for instance the following data:
y <- matrix(runif(500),100,5)
## and perform the test:
d <- empcopu.simulate(100,5,3)
test <- empcopu.test(y,d)
test
dependogram(test)

## NB: In order to save d for future use, the save function can be used.
```

`evCopula-class` *Class "evCopula"*

Description

Extreme value copula

Objects from the Class

Objects can be created by calls of the form `new("evCopula", ...)` or by function 'evCopula'.

Slots

`dimension`: Object of class `"numeric"`
`parameters`: Object of class `"numeric"`
`param.names`: Object of class `"character"`
`param.lowbnd`: Object of class `"numeric"`
`param.upbnd`: Object of class `"numeric"`
`message`: Object of class `"character"`

Methods

dcopula `signature(copula = "galambosCopula")`: ...
pcopula `signature(copula = "galambosCopula")`: ...
rcopula `signature(copula = "galambosCopula")`: ...
dcopula `signature(copula = "huslerReissCopula")`: ...
pcopula `signature(copula = "huslerReissCopula")`: ...
rcopula `signature(copula = "huslerReissCopula")`: ...

Extends

Class `"evCopula"` extends class `"copula"` directly. Class `"galambosCopula"` and `"huslerReissCopula"` extends class `"evCopula"` directly.

Note

The expressions of pdf are obtained by differentiating the cdf expression using function 'deriv'.

`"gumbelCopula"` is also of class `"archmCopula"`.

Author(s)

Jun Yan <jyan@stat.uconn.edu>

See Also

`evCopula`, `copula-class`.

`evCopula` *Construction of Extreme Value Copula Class Object*

Description

Constructs an extreme value copula class object with its corresponding parameter.

Usage

```
evCopula(family, param, dim = 2, ...)
galambosCopula(param)
huslerReissCopula(param)
```

Arguments

`family`	a character string specifying the family of an extreme value copula. Implemented families are "galambos" and "gumbel".
`param`	a numeric vector specifying the parameter values.
`dim`	the dimension of the copula.
`...`	currently nothing.

Value

An extreme value copula object of class `"galambosCopula"` `"gumbelCopula"`, or `"huslerReissCopula"`.

Note

Gumbel copula is both Archimedean copula and extreme value copula.

Author(s)

Jun Yan <jyan@stat.uconn.edu>

See Also

`ellipCopula`, `archmCopula`

Examples

```
gumbel.cop <- evCopula("gumbel", param=2)
contour(gumbel.cop, dcopula)
galambos.cop <- galambosCopula(2)
contour(galambos.cop, dcopula)
```

`fgmCopula-class` *Class "fgmCopula"*

Description

Multivariate multi-parameter Farlie-Gumbel-Morgenstern copula as defined in Nelsen (1999, p 87).

Objects from the Class

Objects can be created by calls of the form `new("fgmCopula", ...)`.

Slots

exprdist: Object of class `"expression"`, expressions for the cdf and pdf of the copula. These expressions are used in function 'pcopula' and 'dcopula'.

dimension: Object of class `"numeric"`

parameters: Object of class `"numeric"`

param.names: Object of class `"character"`

param.lowbnd: Object of class `"numeric"`

param.upbnd: Object of class `"numeric"`

message: Object of class `"character"`

Methods

dcopula `signature(copula = "fgmCopula")`: ...

pcopula `signature(copula = "fgmCopula")`: ...

rcopula `signature(copula = "fgmCopula")`: ...

Extends

Class `"fgmCopula"` extends class `"copula"` directly.

Note

The verification of the validity of the parameter values is of high complexity and may not work for high dimensional copulas.

The random number generation needs to be properly tested, especially for dimensions higher than 2.

Author(s)

Ivan Kojadinovic, ⟨ivan@stat.auckland.ac.nz⟩

References

Nelsen (1999), *An introduction to Copulas*, Springer, New York.

See Also

copula-class, fgmCopula-class.

`fgmCopula` *Construction of a fgmCopula Class Object*

Description

Constructs a multivariate multi-parameter Farlie-Gumbel-Morgenstern copula (see Nelsen 1999, p 87) class object with its corresponding parameters and dimension.

Usage

```
fgmCopula(param, dim = 2)
```

Arguments

`param`	a numeric vector specifying the parameter values.
`dim`	the dimension of the copula.
`...`	currently nothing.

Value

A Farlie-Gumbel-Morgenstern copula object of class `"fgmCopula"`.

Note

The verification of the validity of the parameter values is of high complexity and may not work for high dimensional copulas.

The random number generation needs to be properly tested, especially for dimensions higher than 2.

Author(s)

Ivan Kojadinovic, ⟨ivan@stat.auckland.ac.nz⟩

References

Nelsen (1999), *An introduction to Copulas*, Springer, New York.

See Also

Copula, copula-class.

Examples

```
## a bivariate example
fgm.cop <- fgmCopula(1)
x <-rcopula(fgm.cop, 1000)
cor(x,method="kendall")
kendallsTau(fgm.cop)
cor(x,method="spearman")
spearmansRho(fgm.cop)
persp(fgm.cop, dcopula)
contour(fgm.cop, dcopula)

## a trivariate example with wrong parameter values
## fgm2.cop <- fgmCopula(c(1,1,1,1), dim=3)

## a trivariate example with satisfactory parameter values
fgm2.cop <- fgmCopula(c(.2,-.2,-.4,.6), dim=3)
fgm2.cop
```

---

fitCopula-class *Class "fitCopula"*

Description

Classes and summaries for fitting copula models.

Objects from the Class

Objects can be created by calls of the form `fitCopula, fitMvdc` or their `summary` methods.

Author(s)

Jun Yan <jyan@stat.uconn.edu>

---

fitCopula *Maximum Likelihood Estimation of Copula Models*

Description

Fit a copula model to multivariate data.

Usage

```
loglikCopula(param, x, copula)
loglikMvdc(param, x, mvdc)
fitCopula(data, copula, start, lower=NULL, upper=NULL,
          optim.control = list(NULL), method = "BFGS")
fitMvdc(data, mvdc, start, optim.control = list(NULL), method = "BFGS")
```

Arguments

param	a vector of parameter values
x	a data matrix
copula	a 'copula' object
mvdc	a 'mvdc' object
data	a data matrix
start	a vector of starting value for param
lower, upper	bounds on the variables for the '"L-BFGS-B"' method.
optim.control	a list of control to be passed to optim
method	the method for optim

Value

The return values of 'loglikCopula' and 'loglikMvdc' are the loglikelihood evaluated at the given value of 'param'.

The return values of 'fitCopula' and 'fitMvdc' are an object of class 'fitCopula' and 'fitMvdc', respectively, containing slots:

est	the estimate of the parameters
var.est	variance matrix of the estimate
loglik	loglikelihood at est
fit	the result of optim

Note

When covariates are available for marginal distributions or copula, one can construct loglikelihood function and feed it to optim to estimate all the parameters.

Author(s)

Jun Yan <jyan@stat.uconn.edu>

References

Yan (2006), *Multivariate Modeling with Copulas and Engineering Applications. In Handbook of Engineering Statistics*, Ed. Pham, Springer.

See Also

Copula, mvdc

Examples

```
gmb <- gumbelCopula(3, dim=2)
myMvd <- mvdc(gmb, c("exp","exp"), list(list(rate=2),list(rate=4)))
x <- rmvdc(myMvd, 1000)
fit <- fitMvdc(x, myMvd, c(1,1,2))
fit
```

---

generator *Generators for Archimedean and EV Copulas*

Description

Methods function to evaluate generator function, inverse generator function, and derivatives of generator function.

Methods

copula = "copula" Association measure for a `"copula"` object.

---

mvdc-class *Class "mvdc"*

Description

A class of multivariate distribution via copula.

Objects from the Class

Objects can be created by calls of the form `new("mvdc", ...)` or by function `mvdc`.

Slots

`copula:` Object of class `"copula"`, specifying the copula.

`margins:` Object of class `"character"`, specifying the marginal distributions.

`paramMargins:` Object of class `"list"`, a list of list, with eaching list giving the names parameter values of the margins.

Methods

contour `signature(x = "mvdc")`: ...

persp `signature(x = "mvdc")`: ...

Author(s)

Jun Yan <jyan@stat.uconn.edu>

See Also

mvdc

---

persp-methods *Methods for Function persp in Package 'copula'*

Description

Methods for function persp in package **copula**

Details

When x is of class "copula", these arguments can be supplied: fun: the function to be plotted, "dcopula" or "pcopula". n = 51: the number of points to do the plotting. theta = -30, phi = 30, expand = 0.618: arguments for "persp"

when x is of class "mvdc", two more arguments are expected to replace the effect of n = 51: xlim: the range of the x variable. ylim: the range of the y variable.

Methods

x = "copula" Perspective plot for a "copula" object.

x = "mvdc" Perspective plot for a "mvdc" object.

Examples

```
persp(frankCopula(-0.8), dcopula)
persp(claytonCopula(2), pcopula)
x <- mvdc(gumbelCopula(3), c("norm", "norm"),
          list(list(mean = 0, sd =1), list(mean = 1)))
persp(x, dmvdc, xlim=c(-2, 2), ylim=c(-1, 3))
persp(x, pmvdc, xlim=c(-2, 2), ylim=c(-1, 3))
```

---

plackettCopula *Construction of Plackett Copula Class Object*

Description

Constructs an Plackett copula class object with its corresponding parameter.

Usage

```
plackettCopula(param)
```

Arguments

param	a numeric vector specifying the parameter values.

Value

A Plackett copula object of class `"plackettCopula"`.

Author(s)

Jun Yan <jyan@stat.uconn.edu>

See Also

`ellipCopula`, `archmCopula`

Examples

```
plackett.cop <- plackettCopula(param=2)
tailIndex(plackett.cop)
```

---

show-methods *Methods for Function show in Package 'copula'*

Description

Methods for function `show` in package **copula**.

Methods

object = "copula" describe this method here
object = "fitCopula" describe this method here
object = "fitMvdc" describe this method here

Examples

---

summary-methods *Methods for Function summary in Package 'copula'*

Description

Methods for function `summary` in package **copula**.

Methods

object = "fitCopula" describe this method here
object = "fitMvdc" describe this method here

Examples

Index

∗Topic **classes**
archmCopula-class, 4
copula-class, 7
ellipCopula-class, 8
evCopula-class, 19
fgmCopula-class, 22
fitCopula-class, 24
mvdc-class, 26
∗Topic **distribution**
archmCopula, 5
Copula, 1
ellipCopula, 9
evCopula, 21
fgmCopula, 23
Mvdc, 3
plackettCopula, 27
∗Topic **hplot**
contour-methods, 7
persp-methods, 27
∗Topic **htest**
empcopm.test, 11
empcops.test, 13
empcopsm.test, 15
empcopu.test, 17
∗Topic **methods**
AssocMeasures, 6
contour-methods, 7
generator, 26
persp-methods, 27
show-methods, 28
summary-methods, 28
∗Topic **models**
fitCopula, 24
∗Topic **multivariate**
archmCopula, 5
AssocMeasures, 6
Copula, 1
ellipCopula, 9
evCopula, 21
fgmCopula, 23
fitCopula, 24
generator, 26
Mvdc, 3
plackettCopula, 27
∗Topic **print**
show-methods, 28

Afun *(generator)*, 26
Afun,galambosCopula-method *(generator)*, 26
Afun,gumbelCopula-method *(generator)*, 26
Afun,huslerReissCopula-method *(generator)*, 26
Afun-methods *(generator)*, 26
amhCopula *(archmCopula)*, 5
amhCopula-class *(archmCopula-class)*, 4
archmCopula, *2*, *4*, 5, *5*, *10*, *21*, *28*
archmCopula-class, *8*
archmCopula-class, 4
AssocMeasures, 6

calibKendallsTau *(AssocMeasures)*, 6
calibKendallsTau,ANY-method *(AssocMeasures)*, 6
calibKendallsTau,claytonCopula-method *(AssocMeasures)*, 6
calibKendallsTau,copula-method *(AssocMeasures)*, 6
calibKendallsTau,ellipCopula-method *(AssocMeasures)*, 6
calibKendallsTau,gumbelCopula-method *(AssocMeasures)*, 6
calibKendallsTau,normalCopula-method *(AssocMeasures)*, 6
calibKendallsTau,tCopula-method *(AssocMeasures)*, 6

calibKendallsTau-methods
(AssocMeasures), 6
calibSpearmansRho
(AssocMeasures), 6
calibSpearmansRho,ANY-method
(AssocMeasures), 6
calibSpearmansRho,copula-method
(AssocMeasures), 6
calibSpearmansRho,ellipCopula-method
(AssocMeasures), 6
calibSpearmansRho,normalCopula-method
(AssocMeasures), 6
calibSpearmansRho,tCopula-method
(AssocMeasures), 6
calibSpearmansRho-methods
(AssocMeasures), 6
claytonCopula (archmCopula), 5
claytonCopula-class
(archmCopula-class), 4
contour,copula-method
(contour-methods), 7
contour,mvdc-method
(contour-methods), 7
contour-methods, 7
Copula, 1, 23, 25
copula-class, 2, 4, 5, 9, 20, 23
copula-class, 7

dcopula (Copula), 1
dcopula,amhCopula-method
(Copula), 1
dcopula,claytonCopula-method
(Copula), 1
dcopula,fgmCopula-method
(Copula), 1
dcopula,frankCopula-method
(Copula), 1
dcopula,galambosCopula-method
(Copula), 1
dcopula,gumbelCopula-method
(Copula), 1
dcopula,huslerReissCopula-method
(Copula), 1
dcopula,normalCopula-method
(Copula), 1
dcopula,plackettCopula-method
(Copula), 1
dcopula,tCopula-method (Copula), 1
dependogram, 12, 14, 16
dependogram (empcopu.test), 17
dmvdc (Mvdc), 3

ellipCopula, 2, 4, 6, 9, 9, 21, 28
ellipCopula-class, 8
ellipCopula-class, 8
empcopm.test, 11, 14, 16, 18
empcops.simulate (empcops.test),
13
empcops.test, 12, 13, 15, 16, 18
empcopsm.test, 12, 14, 15, 18
empcopu.simulate (empcopu.test),
17
empcopu.test, 11–14, 16, 17
evCopula, 20, 21
evCopula-class, 8
evCopula-class, 19

fgmCopula, 2, 23
fgmCopula-class, 8, 23
fgmCopula-class, 22
fitCopula, 24
fitCopula-class, 24
fitMvdc (fitCopula), 24
fitMvdc-class (fitCopula-class),
24
frankCopula (archmCopula), 5
frankCopula-class
(archmCopula-class), 4

galambosCopula (evCopula), 21
galambosCopula-class
(evCopula-class), 19
generator, 26
genFun (generator), 26
genFun,amhCopula-method
(generator), 26
genFun,claytonCopula-method
(generator), 26
genFun,frankCopula-method
(generator), 26
genFun,gumbelCopula-method
(generator), 26
genFun-methods (generator), 26
genFunDer1 (generator), 26
genFunDer1,amhCopula-method
(generator), 26
genFunDer1,claytonCopula-method
(generator), 26

genFunDer1,frankCopula-method
(generator), 26
genFunDer1,gumbelCopula-method
(generator), 26
genFunDer1-methods (generator), 26
genFunDer2 (generator), 26
genFunDer2,amhCopula-method
(generator), 26
genFunDer2,claytonCopula-method
(generator), 26
genFunDer2,frankCopula-method
(generator), 26
genFunDer2,gumbelCopula-method
(generator), 26
genFunDer2-methods (generator), 26
genInv (generator), 26
genInv,amhCopula-method
(generator), 26
genInv,claytonCopula-method
(generator), 26
genInv,frankCopula-method
(generator), 26
genInv,gumbelCopula-method
(generator), 26
genInv-methods (generator), 26
gumbelCopula (archmCopula), 5
gumbelCopula-class
(archmCopula-class), 4

huslerReissCopula (evCopula), 21
huslerReissCopula-class
(evCopula-class), 19

kendallsTau (AssocMeasures), 6
kendallsTau,amhCopula-method
(AssocMeasures), 6
kendallsTau,ANY-method
(AssocMeasures), 6
kendallsTau,archmCopula-method
(AssocMeasures), 6
kendallsTau,claytonCopula-method
(AssocMeasures), 6
kendallsTau,copula-method
(AssocMeasures), 6
kendallsTau,fgmCopula-method
(AssocMeasures), 6
kendallsTau,frankCopula-method
(AssocMeasures), 6
kendallsTau,gumbelCopula-method
(AssocMeasures), 6
kendallsTau,normalCopula-method
(AssocMeasures), 6
kendallsTau,tCopula-method
(AssocMeasures), 6
kendallsTau-methods
(AssocMeasures), 6

loglikCopula (fitCopula), 24
loglikMvdc (fitCopula), 24

Mvdc, 3
mvdc, *25*, *27*
mvdc (Mvdc), 3
mvdc-class, *4*
mvdc-class, 26

normalCopula (ellipCopula), 9
normalCopula-class
(ellipCopula-class), 8

pcopula (Copula), 1
pcopula,amhCopula-method
(Copula), 1
pcopula,claytonCopula-method
(Copula), 1
pcopula,fgmCopula-method
(Copula), 1
pcopula,frankCopula-method
(Copula), 1
pcopula,galambosCopula-method
(Copula), 1
pcopula,gumbelCopula-method
(Copula), 1
pcopula,huslerReissCopula-method
(Copula), 1
pcopula,normalCopula-method
(Copula), 1
pcopula,plackettCopula-method
(Copula), 1
pcopula,tCopula-method (Copula), 1
persp,copula-method
(persp-methods), 27
persp,mvdc-method
(persp-methods), 27
persp-methods, 27
plackettCopula, 27
plackettCopula-class
(copula-class), 7

pmvdc *(Mvdc)*, 3

rcopula *(Copula)*, 1
rcopula,amhCopula-method *(Copula)*, 1
rcopula,claytonCopula-method *(Copula)*, 1
rcopula,fgmCopula-method *(Copula)*, 1
rcopula,frankCopula-method *(Copula)*, 1
rcopula,galambosCopula-method *(Copula)*, 1
rcopula,gumbelCopula-method *(Copula)*, 1
rcopula,huslerReissCopula-method *(Copula)*, 1
rcopula,normalCopula-method *(Copula)*, 1
rcopula,plackettCopula-method *(Copula)*, 1
rcopula,tCopula-method *(Copula)*, 1
rmvdc *(Mvdc)*, 3

show,copula-method *(show-methods)*, 28
show,fitCopula-method *(show-methods)*, 28
show,fitMvdc-method *(show-methods)*, 28
show,normalCopula-method *(show-methods)*, 28
show,tCopula-method *(show-methods)*, 28
show-methods, 28
spearmansRho *(AssocMeasures)*, 6
spearmansRho,ANY-method *(AssocMeasures)*, 6
spearmansRho,copula-method *(AssocMeasures)*, 6
spearmansRho,fgmCopula-method *(AssocMeasures)*, 6
spearmansRho,frankCopula-method *(AssocMeasures)*, 6
spearmansRho,normalCopula-method *(AssocMeasures)*, 6
spearmansRho,tCopula-method *(AssocMeasures)*, 6
spearmansRho-methods *(AssocMeasures)*, 6
summary,fitCopula-method *(summary-methods)*, 28
summary,fitMvdc-method *(summary-methods)*, 28
summary-methods, 28
summaryFitCopula-class *(fitCopula-class)*, 24
summaryFitMvdc-class *(fitCopula-class)*, 24

tailIndex *(AssocMeasures)*, 6
tailIndex,ANY-method *(AssocMeasures)*, 6
tailIndex,claytonCopula-method *(AssocMeasures)*, 6
tailIndex,copula-method *(AssocMeasures)*, 6
tailIndex,evCopula-method *(AssocMeasures)*, 6
tailIndex,frankCopula-method *(AssocMeasures)*, 6
tailIndex,gumbelCopula-method *(AssocMeasures)*, 6
tailIndex,normalCopula-method *(AssocMeasures)*, 6
tailIndex,tCopula-method *(AssocMeasures)*, 6
tailIndex-methods *(AssocMeasures)*, 6
tCopula *(ellipCopula)*, 9
tCopula-class *(ellipCopula-class)*, 8