Package 'cort'

December 1, 2020

Title Some Empiric and Nonparametric Copula Models

Version 0.3.2

Description Provides S4 classes and methods to fit several copula models: The classic empirical checkerboard copula and the empirical checkerboard copula with known margins, see Cuberos, Masiello and Maume-Deschamps (2019) <doi:10.1080/03610926.2019.1586936> are proposed. These two models allow to fit copulas in high dimension with a small number of observations, and they are always proper copulas. Some flexibility is added via a possibility to differentiate the checkerboard parameter by dimension. The last model consist of the implementation of the Copula Recursive Tree algorithm proposed by Laverny, Maume-Deschamps, Masiello and Rullière (2020) <arXiv:2005.02912>, including the localised dimension reduction, which fits a copula by recursive splitting of the copula domain. We also provide an efficient way of mixing copulas, allowing to bag the algorithm into a forest, and a generic way of measuring d-dimensional boxes with a copula.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Depends R (>= 2.10)

Imports Rdpack, methods, purrr, nloptr, osqp, Rcpp, furrr (>= 0.2.0)

URL https://github.com/lrnv/cort

BugReports https://github.com/lrnv/cort/issues

Suggests covr, testthat (>= 2.1.0), spelling, knitr, rmarkdown

Language en-US

Collate 'utils.R' 'generics.R' 'ConvexCombCopula.R' 'empiricalCopula.R' 'Cort.R' 'CortForest.R' 'RcppExports.R' 'cbCopula.R' 'cbkmCopula.R' 'cort-package.R' 'data.R'

VignetteBuilder knitr

RdMacros Rdpack

LinkingTo Rcpp

NeedsCompilation yes

Author Oskar Laverny [aut, cre] (<https://orcid.org/0000-0002-7508-999X>)

Maintainer Oskar Laverny <oskar.laverny@gmail.com>

Repository CRAN

Date/Publication 2020-12-01 00:30:20 UTC

R topics documented:

biv_rho . . . 2
biv_tau . . . 3
cbCopula-Class . . . 4
cbkmCopula-Class . . . 5
clayton_data . . . 7
constraint_infl . . . 8
ConvexCombCopula-Class . . . 8
Cort-Class . . . 9
CortForest-Class . . . 11
dCopula . . . 13
funcdep_data . . . 14
impossible_data . . . 15
kendall_func . . . 16
loss . . . 17
pCopula . . . 17
project_on_dims . . . 19
quad_norm . . . 19
quad_prod . . . 20
quad_prod_with_data . . . 21
rCopula . . . 22
recoveryourself_data . . . 23
vCopula . . . 24

Index **26**

`biv_rho` *Spearman's rho matrix of a copula*

Description

Computes the bivariate Spearmann's rho matrix for a copula.

Usage

```
biv_rho(copula)

## S4 method for signature 'Cort'
biv_rho(copula)
```

Arguments

copula	the copula object

Value

the density of the copula on each observation

Functions

- biv_rho,Cort-method: Method for the class Cort

Examples

```
cop <- Cort(LifeCycleSavings[,1:3])
biv_rho(cop)
```

---

biv_tau *Kendall's tau matrix of a copula*

Description

Computes the bivariate Kendall's tau matrix for a copula.

Usage

```
biv_tau(copula)

## S4 method for signature 'Cort'
biv_tau(copula)
```

Arguments

copula	the copula object

Value

the density of the copula on each observation

Functions

- biv_tau,Cort-method: Method for the class Cort

Examples

```
cop <- Cort(cort::funcdep_data[1:10,1:3])
biv_tau(cop)
```

`cbCopula-Class` *Checkerboard copulas*

Description

cbCopula contructor

Usage

```
cbCopula(x, m = rep(nrow(x), ncol(x)), pseudo = FALSE)
```

Arguments

`x`	the data to be used
`m`	checkerboard parameters
`pseudo`	Boolean, defaults to `FALSE`. Set to `TRUE` if you are already providing pseudo data into the `x` argument.

Details

The cbCopula class computes a checkerboard copula with a given checkerboard parameter m, as described by A. Cuberos, E. Masiello and V. Maume-Deschamps (2019). Assymptotics for this model are given by C. Genest, J. Neslehova and R. bruno (2017). The construction of this copula model is as follows :

Start from a dataset with n i.i.d observation of a d-dimensional copula (or pseudo-observations), and a checkerboard parameter m,dividing n.

Consider the ensemble of multi-indexes $I = \{i = (i_1, .., i_d) \subset \{1, ..., m\}^d\}$ which indexes the boxes :

$$B_i = \left]\frac{i-1}{m}, \frac{i}{m}\right]$$

Let now λ be the dimension-unspecific lebesgue measure on any power of R, that is :

$$\forall d \in N, \forall x, y \in R^p, \lambda((x, y)) = \prod_{p=1}^{d}(y_i - x_i)$$

Let furthermore μ and $\hat{\mu}$ be respectively the true copula measure of the sample at hand and the classical Deheuvels empirical copula, that is :

- For n i.i.d observation of the copula of dimension d, let $\forall i \in \{1, ..., d\}$, $R_i^1, ..., R_i^d$ be the marginal ranks for the variable i.
- $\forall x \in I^d$ let $\hat{\mu}((0, x)) = \frac{1}{n}\sum_{k=1}^{n} I_{R_1^k \leq x_1, ..., R_d^k \leq x_d}$

The checkerboard copula, C, and the empirical checkerboard copula, $\hat{C}$, are then defined by the following :

$$\forall x \in (0,1)^d, C(x) = \sum_{i \in I} m^d \mu(B_i) \lambda((0,x) \cap B_i)$$

Where $m^d = \lambda(B_i)$.

This copula is a special form of patchwork copulas, see F. Durante, J. Fernández Sánchez and C. Sempi (2013) and F. Durante, J. Fernández Sánchez, J. Quesada-Molina and M. Ubeda-Flores (2015). The estimator has the good property of always being a copula.

The checkerboard copula is a kind of patchwork copula that only uses independent copula as fill-in, only where there are values on the empirical data provided. To create such a copula, you should provide data and checkerboard parameters (depending on the dimension of the data).

Value

An instance of the `cbCopula` S4 class. The object represent the fitted copula and can be used through several methods to query classical (r/d/p/v)Copula methods, etc.

References

Cuberos A, Masiello E, Maume-Deschamps V (2019-mar). "Copulas Checker-Type Approximations: Application to Quantiles Estimation of Sums of Dependent Random Variables." *Communications in Statistics - Theory and Methods*, 1–19.

Genest C, NeÅ¡lehovÃ¡ JG, RÃ©millard B (2017-jul). "Asymptotic Behavior of the Empirical Multilinear Copula Process under Broad Conditions." *Journal of Multivariate Analysis*, **159**, 82–110.

Durante F, FernÃ¡ndez SÃ¡nchez J, Sempi C (2013-nov). "Multivariate Patchwork Copulas: A Unified Approach with Applications to Partial Comonotonicity." *InsuranceMathematics and Economics*, **53**, 897–905.

Durante F, FernÃ¡ndez-SÃ¡nchez J, Quesada-Molina JJ, Ãšbeda-Flores M (2015-dec). "Convergence Results for Patchwork Copulas." *European Journal of Operational Research*, **247**, 525–531.

`cbkmCopula-Class` *Checkerboards with known margins*

Description

cbkmCopula contructor

Usage

```
cbkmCopula(
  x,
  m = rep(nrow(x), ncol(x)),
  pseudo = FALSE,
  margins_numbers = NULL,
  known_cop = NULL
)
```

Arguments

x	the data to be used
m	checkerboard parameter
pseudo	Boolean, defaults to FALSE. Set to TRUE if you are already providing pseudo-data into the x argument.
margins_numbers	numeric integers which determines the margins for the known copula.
known_cop	Copula a copula object representing the known copula for the selected margins.

Details

Given some empirical data, and given some known copula estimation on a sub-vector of this data, the checkerboard with known margins construction consist in a conditional pattern where a checkerboard copula is fitted (similar the the cbCopula algorithm), but conditionally on some known margins.

See the corresponding vignette for more details.

Value

An instance of the cbkmCopula S4 class. The object represent the fitted copula and can be used through several methods to query classical (r/d/p/v)Copula methods, etc.

Examples

```
dataset <- apply(LifeCycleSavings,2,rank)/(nrow(LifeCycleSavings)+1)
known_copula <- cbCopula(dataset[,2:3],m=10)
(cop <- cbkmCopula(x = dataset,
                   m = 5,
                   pseudo = TRUE,
                   margins_numbers = c(2,3),
                   known_cop = known_copula))
```

clayton_data *Dataset clayton_data*

Description

This dataset is a simulation of 200 points from a 3-dimensional clayton copula with $\theta = 7$, hence highly dependent, for the first, third and fourth marginals. The second marginal is added as independent uniform draws. Lastly, the third marginal is flipped, inducing a negative dependence structure.

Usage

```
clayton_data
```

Format

A matrix with 200 rows and 4 columns

The example section below gives the code to re-generate this data if needed.

Details

This dataset is studied in O. Laverny, V. Maume-Deschamps, E. Masiello and D. Rullière (2020).

References

Laverny O, Maume-Deschamps V, Masiello E, RulliÃ¨re D (2020). "Dependence Structure Estimation Using Copula Recursive Trees." *arXiv preprint arXiv:2005.02912*.

Examples

```
psi <- function(t,alpha) (1 + sign(alpha)*t) ^ (-1/alpha) # generator
rClayton <- function(n,dim,alpha){
  val <- matrix(runif(n * dim), nrow = n)
  gam <- rgamma(n, shape = 1/alpha, rate = 1)
  gam <- matrix(gam, nrow = n, ncol = dim)
  psi(- log(val) / gam,alpha)
}
set.seed(12,kind = "Mersenne-Twister",normal.kind = "Inversion")
clayton_data <- matrix(nrow=200,ncol=4)
clayton_data[,c(1,4,3)] = rClayton(n=200,dim=3,alpha=7)
clayton_data[,2] = runif(200)
clayton_data[,3] <- 1 - clayton_data[,3]
```

constraint_infl *Constraint influence of the model (if it has one)*

Description

Currently only implemented for Cort models. Compute the constraint influence of the model

Usage

```
constraint_infl(object)

## S4 method for signature 'Cort'
constraint_infl(object)

## S4 method for signature 'CortForest'
constraint_infl(object)
```

Arguments

object	the copula object

Value

The constraint influence statistic of the model

Functions

- constraint_infl,Cort-method: Method for the class Cort
- constraint_infl,CortForest-method: Method for the class CortForest

Examples

```
cop <- Cort(cort::recoveryourself_data[1:10,])
constraint_infl(cop)
```

ConvexCombCopula-Class *Convex Combination of copulas.*

Description

ConvexCombCopula class

Usage

```
ConvexCombCopula(copulas, alpha = rep(1, length(copulas)))
```

Arguments

copulas	a list of copulas of same dimension
alpha	a vector of (positive) weights

Details

The ConvexCombcopula class is used to build convex combinations of copulas, with given positives weights. The rCopula and pCopula functions works for those copulas, assuming they work for the given copulas that we combined in a convex way.

See the corresponding vignette for more details about the implementation.

Value

An instance of the `ConvexCombCopula` S4 class. The object represent the copula that results from a convex combinaison of other copulas, and can be used through several methods to query classical (r/d/p/v)Copula methods, etc.

Examples

```
dataset <- apply(LifeCycleSavings,2,rank)/(nrow(LifeCycleSavings)+1)
copulas <- list(
  cbCopula(dataset[,2:3],m=10),
  cbCopula(dataset[,2:3],m=5)
)
alpha <- c(1,4)
(cop <- ConvexCombCopula(copulas,alpha))
```

Cort-Class *Cort copulas*

Description

Cort class

Usage

```
Cort(
  x,
  p_value_for_dim_red = 0.75,
  min_node_size = 1,
  pseudo_data = FALSE,
  number_max_dim = NULL,
  verbose_lvl = 1,
  slsqp_options = NULL,
  osqp_options = NULL,
  N = 999,
  force_grid = FALSE
)
```

Arguments

x	The data, must be provided as a matrix with each row as an observation.
p_value_for_dim_red	a p_value for the localized dimension reduction test
min_node_size	The minimum number of observation available in a leaf to initialize a split.
pseudo_data	set to True if you are already providing data on the copula space.
number_max_dim	The maximum number of dimension a split occurs in. Defaults to be all of the dimensions.
verbose_lvl	numeric. set the verbosity. 0 for no output and bigger you set it the most output you get.
slsqp_options	options for nloptr::slsqp to find breakpoints : you can change defaults.
osqp_options	options for the weights optimization. You can pass a call to osqp::osqpSettings, or NULL for defaults.
N	The number of bootstrap samples for p_values computations.
force_grid	Set to TRUE to force breakpoints to be on the n-checkerboard grid.

Details

This class implements the CORT algorithm to a fit a multivariate copula using piece constant density. Given a dataset `x`, the function will produce an estimator for the copula of this dataset that is tree-shaped, by recursive partitioning of the unit hypercube. the `min_node_size` parameter controls the stopping conditions for the splitting procedure. Once the space is splitted, we ran a quadratic solver, which options can be tweaked via the `osqp_options` parameter, to ensure that the weights respect the copula conditions.

Once the model is fitted, it can be used through the classical (r/d/p/v)Copula functions to compute, respectively, random number generations, the density, the cdf and the volume function of the copula.

See O. Laverny, E. Masiello, V. Maume-Deschamps and D. Rullière (2020) for the details of this density estimation procedure, and `vignettes(package='cort')` for examples of usecases.

Value

An instance of the `Cort` S4 class. The object represent the fitted copula and can be used through several methods to query classical (r/d/p/v)Copula methods, constraint influence, etc. Beside returning some inputted parameters, notable slots are :

- `data` Your original data
- `dim` The dimension of problem, number of columns of your dataset
- `f` The empirical frequency in the leaves
- `p` The fitted probabilities of each leaf
- `a` Minimum points of leaves
- `b` Maximum points of leaves
- `vols` Volume of the leaves

More details about these slots can be found in the reference.

References

Laverny O, Maume-Deschamps V, Masiello E, RulliÃ¨re D (2020). "Dependence Structure Estimation Using Copula Recursive Trees." *arXiv preprint arXiv:2005.02912*.

Examples

```
(Cort(LifeCycleSavings[,1:3]))
```

`CortForest-Class` *Bagged Cort copulas*

Description

CortForest class

Usage

```
CortForest(
  x,
  p_value_for_dim_red = 0.75,
  n_trees = 10,
  compte_loo_weights = FALSE,
  min_node_size = 1,
  pseudo_data = FALSE,
  number_max_dim = NULL,
  verbose_lvl = 2,
  force_grid = FALSE,
  oob_weighting = TRUE
)
```

Arguments

`x`	The data, must be provided as a matrix with each row as an observation.
`p_value_for_dim_red`	a p_value for the localised dimension reduction test
`n_trees`	Number of trees
`compte_loo_weights`	Defaults to FALSE. Allows to use an automatic re-weighting of the trees in the forest, based on leave-one-out considerations.
`min_node_size`	The minimum number of observation avaliable in a leaf to initialise a split.
`pseudo_data`	set to True if you are already providing data on the copula space.
`number_max_dim`	The maximum number of dimension a split occurs in. Defaults to be all of the dimensions.
`verbose_lvl`	verbosity level : can be 0 (default) or an integer. bigger the integer bigger the output level.

force_grid	boolean (default: FALSE). set to TRUE to force breakpoint to be on the n-checkerboard grid in every tree.
oob_weighting	boolean (default : TRUE) option to weight the trees with an oob criterion (otherwise they are equally weighted)

Details

This class implements the bagging of CORT models, with an out-of-bag error minimisation in the weights.

See O. Laverny, V. Maume-Deschamps, E. Masiello and D. Rullière (2020) for the details of this density estimation procedure, and `vignettes(package='cort')` for examples of usecases.

Value

An instance of the `CortForest` S4 class. The object represent the fitted copula and can be used through several methods to query classical (r/d/p/v)Copula methods, constraint influence, etc. Beside returning some inputted parameters, notable slots are :

- `trees` A list of Cort objects representing each fitted tree in the forest.
- `weights` The weigths of each tree.
- `indexes` The indexes of data points that were selected for fitting the trees
- `pmf` The density of each tree on data points
- `norm_matrix` The matrix of scalar product between trees
- `oob_pmf` The density of each tree on data points it did not see during fitting
- `oob_kl` The out-of-bag Kullback-Leibler divergence of each tree
- `oob_ise` The out-of-bag Integrated Square Error of each tree

More details about these slots can be found in the reference.

References

Laverny O, Maume-Deschamps V, Masiello E, RulliÃ¨re D (2020). "Dependence Structure Estimation Using Copula Recursive Trees." *arXiv preprint arXiv:2005.02912*.

Examples

```
(CortForest(LifeCycleSavings[,1:3],number_max_dim=2,n_trees=2))
```

dCopula *Copula density*

Description

This function returns the density of a given copula on given observations.

Usage

```
dCopula(u, copula, ...)

## S4 method for signature 'matrix,Cort'
dCopula(u, copula)

## S4 method for signature 'matrix,CortForest'
dCopula(u, copula)

## S4 method for signature 'matrix,cbCopula'
dCopula(u, copula)
```

Arguments

u	numeric matrix : one row per observation
copula	the copula object
...	other parameter to be passed to methods for this generic.

Value

The density of the copula on each observation

Functions

- dCopula,matrix,Cort-method: Method for the class Cort
- dCopula,matrix,CortForest-method: Method for the class CortForest
- dCopula,matrix,cbCopula-method: Method for the cbCopula

Examples

```
cop <- cbCopula(cort::funcdep_data[1:10,1:2], m = 5)
dCopula(rep(0,2),cop)
dCopula(rep(0.5,2),cop)
dCopula(rep(1,2),cop)
```

funcdep_data *Dataset funcdep_data*

Description

This dependence structure is constructed by applying the function :

$$h(u_1, u_2, u_3) = (u_1, \sin(2\pi u_1) - \frac{u_2}{\pi}, (1 + \frac{u_3}{\pi^2})(\frac{u_3}{2} I_{\frac{1}{4} \geq u_1} - \sin(\pi^{x_1}) I_{\frac{1}{4} < u_1}))$$

to uniformly drawn 3-dimensional random vectors. The dataset is the ranks of $h(u)$.

Usage

```
funcdep_data
```

Format

A matrix with 500 rows and 3 columns

The example section below gives the code to re-generate this data if needed.

Details

This dataset is studied in O. Laverny, V. Maume-Deschamps, E. Masiello and D. Rullière (2020).

References

Laverny O, Maume-Deschamps V, Masiello E, RulliÃ¨re D (2020). "Dependence Structure Estimation Using Copula Recursive Trees." *arXiv preprint arXiv:2005.02912*.

Examples

```
set.seed(seed = 12,kind = "Mersenne-Twister",normal.kind = "Inversion")
x = matrix(runif(1500),500,3)
x[,2] = sin(2*pi*x[,1])-x[,2]/pi
x[,3] = (x[,3]*(x[,1]<1/4)/2 - sin(pi**(x[,1]))*(x[,1]>1/4))*(1+x[,3]/(pi^2))
funcdep_data = apply(x,2,function(x){return(rank(x,ties.method = "max"))})/(501)
```

`impossible_data` *Dataset impossible_data*

Description

We simulate from a density inside the piecewise linear copula class, by applying the function:

$$h(u) = (u_1, \frac{u_2}{2} + \frac{1}{2} I_{u_1 \notin (\frac{1}{3}, \frac{2}{3})})$$

to a 200x2 uniform sample, and taking ranks.

Usage

```
impossible_data
```

Format

A matrix with 200 rows and 2 columns

The example section below gives the code to re-generate this data if needed.

Details

This dataset is studied in O. Laverny, V. Maume-Deschamps, E. Masiello and D. Rullière (2020).

References

Laverny O, Maume-Deschamps V, Masiello E, RulliÃ¨re D (2020). "Dependence Structure Estimation Using Copula Recursive Trees." *arXiv preprint arXiv:2005.02912*.

Examples

```
set.seed(seed = 12, kind = "Mersenne-Twister", normal.kind = "Inversion")
x = matrix(runif(400),200,2)
x = t(apply(x, 1,function(u){
  if(u[1]< 1/3){
    u[2] = 1/2 + u[2]/2
  } else{ if(u[1]<2/3){
    u[2] = u[2]/2
  } else {
    u[2] = 1/2 + u[2]/2
  }}
  return(u)
}))
impossible_data = apply(x,2,function(x){return(rank(x,ties.method = "max"))})/(201)
```

kendall_func *Kendall function of a copula (if it has one)*

Description

Currently only implemented for Cort models. Compute the Kendall cdf from the model in a point t

Usage

```
kendall_func(object, t, ...)

## S4 method for signature 'Cort'
kendall_func(object, t, M = 1000)
```

Arguments

object	: the tree
t	: the value where to compute the kendall function, may be a vector of evaluation values;
...	other parameters passed to methods
M	the number of simulations

Value

the quadratic product between the trees

Functions

- kendall_func,Cort-method: Method for the class Cort

Examples

```
cop <- Cort(LifeCycleSavings[,1:3])
kendall_func(cop,0.5)
```

loss *Loss of a copula estimation (if the model has one)*

Description

Currently only implemented for Cort models. Compute the loss of the model.

Usage

```
loss(object)

## S4 method for signature 'Cort'
loss(object)
```

Arguments

object	the copula object

Value

the Integrated square error loss of the model

Functions

- loss,Cort-method: Method for the class Cort

Examples

```
cop <- Cort(cort::recoveryourself_data[1:10,])
loss(cop)
```

pCopula *Copula cdf*

Description

This function returns the value of the copula itself on given points.

Usage

```
pCopula(u, copula, ...)

## S4 method for signature 'matrix,ConvexCombCopula'
pCopula(u, copula)

## S4 method for signature 'matrix,Cort'
pCopula(u, copula)

## S4 method for signature 'matrix,CortForest'
pCopula(u, copula)

## S4 method for signature 'matrix,cbCopula'
pCopula(u, copula)

## S4 method for signature 'matrix,cbkmCopula'
pCopula(u, copula)
```

Arguments

u	numeric matrix : one row per observation
copula	the copula object
...	other parameter to be passed to methods for this generic.

Value

The value of the copula on each observation

Functions

- pCopula,matrix,ConvexCombCopula-method: Method for the cbCopula
- pCopula,matrix,Cort-method: Method for the class Cort
- pCopula,matrix,CortForest-method: Method for the class CortForest
- pCopula,matrix,cbCopula-method: Method for the cbCopula
- pCopula,matrix,cbkmCopula-method: Method for the cbCopula

Examples

```
cop <- cbCopula(cort::recoveryourself_data,m = 5)
pCopula(rep(0,2),cop) == 0
pCopula(rep(0.5,2),cop)
pCopula(rep(1,2),cop) == 1
```

project_on_dims *Projection on smaller dimensions of a copula (if implemented)*

Description

Currently only implemented for Cort models. Compute, as a Cort object, the projection on a smaller set of dimensions of a Cort object.

Usage

```
project_on_dims(object, dims)

## S4 method for signature 'Cort'
project_on_dims(object, dims)
```

Arguments

object	: the tree
dims	the set of dimensions

Value

other cort object

Functions

- project_on_dims,Cort-method: Method for the class Cort

Examples

```
cop <- Cort(LifeCycleSavings[,1:3])
projection = project_on_dims(cop,c(1,2))
```

quad_norm *Quadratic norm of the model (if it has one)*

Description

Currently only implemented for Cort models. Compute the L2 norm of the model

Usage

```
quad_norm(object)

## S4 method for signature 'Cort'
quad_norm(object)

## S4 method for signature 'CortForest'
quad_norm(object)
```

Arguments

object	the copula object

Value

the Integrated square error quad_norm of the model

Functions

- quad_norm,Cort-method: Method for the class Cort
- quad_norm,CortForest-method: Method for the class CortForest

Examples

```
cop <- Cort(cort::impossible_data)
quad_norm(cop)
```

---

quad_prod *Quadratic product of two copulas (if they have one)*

Description

Currently only implemented for Cort models. Compute the L2 quadratic product of 2 trees

Usage

```
quad_prod(object, other_tree)

## S4 method for signature 'Cort,Cort'
quad_prod(object, other_tree)
```

Arguments

object	: the tree
other_tree	: the other tree

Value

the quadratic product between the trees

Functions

- quad_prod,Cort,Cort-method: Method for the class Cort

Examples

```
cop <- Cort(LifeCycleSavings[,1:3])
all.equal(quad_prod(cop,cop),quad_norm(cop))
```

---

quad_prod_with_data *Quadratic product with data of the model (if it has one)*

Description

Currently only implemented for Cort models. Compute the quadratic product with the empirical density from the data

Usage

```
quad_prod_with_data(object)

## S4 method for signature 'Cort'
quad_prod_with_data(object)
```

Arguments

object	the copula object

Value

the quad_prod_with_data of the model

Functions

- quad_prod_with_data,Cort-method: Method for the class Cort

Examples

```
cop <- Cort(LifeCycleSavings[,1:3])
quad_prod_with_data(cop)
```

`rCopula` *Copula random generation*

Description

Random number generation following the given copula. This function performs the simulation of random vectors following the copula.

Usage

```
rCopula(n, copula, ...)

## S4 method for signature 'numeric,ConvexCombCopula'
rCopula(n, copula)

## S4 method for signature 'numeric,Cort'
rCopula(n, copula)

## S4 method for signature 'numeric,CortForest'
rCopula(n, copula)

## S4 method for signature 'numeric,cbCopula'
rCopula(n, copula)

## S4 method for signature 'numeric,cbkmCopula'
rCopula(n, copula)
```

Arguments

`n`	the number of simulations
`copula`	the copula object
`...`	other parameter to be passed to methods for this generic.

Value

A matrix with `n` rows, each representing a random vector generated from the provided copula.

Functions

- `rCopula,numeric,ConvexCombCopula-method`: Method for the cbCopula
- `rCopula,numeric,Cort-method`: Method for the class Cort
- `rCopula,numeric,CortForest-method`: Method for the class CortForest
- `rCopula,numeric,cbCopula-method`: Method for the cbCopula
- `rCopula,numeric,cbkmCopula-method`: Method for the cbCopula

Examples

```
cop <- cbCopula(cort::clayton_data,m = 5)
xx <- rCopula(1000,cop)
```

`recoveryourself_data` *Dataset recoveryourself_data*

Description

This dataset is a simple test: we simulate random samples from a density inside the piecewise copula class, and test whether or not the estimator can recover it. For that, we will use a 2-dimensional sample with 500 observations, uniform on the unit hypercube, and apply the following function:

$$h(u) = (u_1, \frac{u_2 + I_{u_1 \leq \frac{1}{4}} + 2I_{u_1 \leq \frac{1}{2}} + I_{\frac{3}{4} \leq u_1}}{4})$$

Usage

```
recoveryourself_data
```

Format

A matrix with 500 rows and 2 columns

The example section below gives the code to re-generate this data if needed.

Details

This dataset is studied in O. Laverny, V. Maume-Deschamps, E. Masiello and D. Rullière (2020).

References

Laverny O, Maume-Deschamps V, Masiello E, RulliÃ¨re D (2020). "Dependence Structure Estimation Using Copula Recursive Trees." *arXiv preprint arXiv:2005.02912*.

Examples

```
set.seed(seed = 12, kind = "Mersenne-Twister", normal.kind = "Inversion")
x = matrix(runif(1000),500,2)
recoveryourself_data = t(apply(x, 1,function(u){
  if(u[1]< 1/4){
    u[2] = 3/4 + u[2]/4
  } else{ if(u[1]<1/2){
    u[2] = 1/2 + u[2]/4
  } else { if(u[1]<3/4){
    u[2] = u[2]/4
  } else {
    u[2] = 1/4 + u[2]/4
  }}}
```

```
  return(u)
}))
```

---

vCopula *Copula volume on hyper-boxes*

Description

u must be piecewise smaller than v, otherwise the function will return an error.

Usage

```
vCopula(u, v, copula, ...)

## S4 method for signature 'matrix,matrix'
vCopula(u, v, copula)
```

Arguments

u	numeric matrix : minimum point of the hyper-rectangles, one row per observation.
v	numeric matrix : maximum point of the hyper-rectangle, one row per observation.
copula	the copula that we compute the measure on the box (u,v)
...	other parameter to be passed to methods for this generic.

Details

A method is currently implemented for the main virtual class 'Copula', but it assumes that a pCopula method is avaliable for the given copula. This method could be used with Copulas that are not from this package, assuming that pCopula(u,cop) works.

This function computes the measure of the copula according to the algorithm proposed by Cherubini U, Romagnoli S (2009-oct).

Value

the measure of the copula.

References

Cherubini U, Romagnoli S (2009-oct). "Computing the Volume of *n* -Dimensional Copulas." *Applied Mathematical Finance*, **16**, 307–314.

Examples

```
cop <- cbCopula(LifeCycleSavings,m = 5)
vCopula(rep(0,5),rep(1,5),cop) == 1
vCopula(rep(0,5),rep(0.5,5),cop)
```

Index

∗ **datasets**
clayton_data, 7
funcdep_data, 14
impossible_data, 15
recoveryourself_data, 23

biv_rho, 2
biv_rho,Cort-method *(*biv_rho*)*, 2
biv_tau, 3
biv_tau,Cort-method *(*biv_tau*)*, 3

cbCopula *(*cbCopula-Class*)*, 4
cbCopula-Class, 4
cbkmCopula *(*cbkmCopula-Class*)*, 5
cbkmCopula-Class, 5
clayton_data, 7
constraint_infl, 8
constraint_infl,Cort-method *(*constraint_infl*)*, 8
constraint_infl,CortForest-method *(*constraint_infl*)*, 8
ConvexCombCopula *(*ConvexCombCopula-Class*)*, 8
ConvexCombCopula-Class, 8
Cort *(*Cort-Class*)*, 9
Cort-Class, 9
CortForest *(*CortForest-Class*)*, 11
CortForest-Class, 11

dCopula, 13
dCopula,matrix,cbCopula-method *(*dCopula*)*, 13
dCopula,matrix,Cort-method *(*dCopula*)*, 13
dCopula,matrix,CortForest-method *(*dCopula*)*, 13

funcdep_data, 14

impossible_data, 15

kendall_func, 16
kendall_func,Cort-method *(*kendall_func*)*, 16

loss, 17
loss,Cort-method *(*loss*)*, 17

pCopula, 17
pCopula,matrix,cbCopula-method *(*pCopula*)*, 17
pCopula,matrix,cbkmCopula-method *(*pCopula*)*, 17
pCopula,matrix,ConvexCombCopula-method *(*pCopula*)*, 17
pCopula,matrix,Cort-method *(*pCopula*)*, 17
pCopula,matrix,CortForest-method *(*pCopula*)*, 17
project_on_dims, 19
project_on_dims,Cort-method *(*project_on_dims*)*, 19

quad_norm, 19
quad_norm,Cort-method *(*quad_norm*)*, 19
quad_norm,CortForest-method *(*quad_norm*)*, 19
quad_prod, 20
quad_prod,Cort,Cort-method *(*quad_prod*)*, 20
quad_prod_with_data, 21
quad_prod_with_data,Cort-method *(*quad_prod_with_data*)*, 21

rCopula, 22
rCopula,numeric,cbCopula-method *(*rCopula*)*, 22
rCopula,numeric,cbkmCopula-method *(*rCopula*)*, 22
rCopula,numeric,ConvexCombCopula-method *(*rCopula*)*, 22
rCopula,numeric,Cort-method *(*rCopula*)*, 22

rCopula,numeric,CortForest-method
(rCopula), 22
recoveryourself_data, 23

vCopula, 24
vCopula,matrix,matrix,Copula (vCopula),
24
vCopula,matrix,matrix-method (vCopula),
24