Package ‘cort’

April 1, 2020

**Title**  Some Empiric and Nonparametric Copula Models

**Version**  0.3.0

**Description**  Provides S4 classes and methods to fit several copula models: The classic empiric-  
cal checkerboard copula and the empirical checkerboard copula with known mar-  
gins, see Cuberos, Masiello and Maume-Deschamps (2019) <doi:10.1080/03610926.2019.1586936> are proposed. These two models al-  
low to fit copulas in high dimension with a small number of observations, and they are al-  
ways proper copulas. Some flexibility is added via a possibility to differentiate the checker-  
board parameter by dimension. The last model consist of the implementation of the Copula Re-  
cursive Tree algorithm, including the localised dimension reduction, which fits a copula by recur-  
sive splitting of the copula domain. We also provide an efficient way of mixing copulas, allow-  
ing 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**  7.0.2

**Imports**  Rdpack, magrittr, methods, purrr, nloptr, osqp, Repp, furrr

**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.pipe.R' 'utils.R' 'generics.R' 'Box.R'
  'ConvexCombCopula.R' 'empiricalCopula.R' 'WeightedBox.R'
  'Cort.R' 'CortForest.R' 'ReppExports.R' 'cbCopula.R'
  'cbkmCopula.R' 'cort-package.R'

**VignetteBuilder**  knitr

**RdMacros**  Rdpack

**LinkingTo**  Repp

**NeedsCompilation**  yes
Description

Computes the bivariate Spearman’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)

cop <- Cort(LifeCycleSavings[,1:3])
biv_tau(cop)
cbCopula-Class  
Checkerboard copulas

Description

cbCopula constructor

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 $\lambda$ be the dimension-unspecific lebesgue measure on any power of $R$, that is:

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

Let furthermore $\mu$ 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 $\hat{\mu}((0, x)) = \frac{1}{n} \sum_{k=1}^{n} I_{R_i^1 \leq x_1, ..., R_i^d \leq x_d}$

...
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

a cbCopula object

References


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 the checkerboard part is conditional on the known part of the copula.

Value

a cbkmCopula object

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))

constraint_infl  Constraint influence of the model

Description

Compute the constraint influence of the model
ConvexCombCopula-Class

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 Cort

Examples

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

```r
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**

The Cort estimator

**Description**

Cort class

**Usage**

```r
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
)
```

**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.
- **min_node_size**: The minimum number of observations available 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 dimensions a split occurs in. Defaults to be all of the dimensions.
**CortForest-Class**

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.

**Value**

a Cort object that can be fitted easily to produce a copula estimate.

**Examples**

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

---

**CortForest-Class**  
**Bagged Cort estimates**

---

**Description**

CortForest class

**Usage**

```
CortForest(
  x,
  p_value_for_dim_red = 0.75,
  n_trees = 10,
  conect_loo_weights = FALSE,
  min_node_size = 1,
  pseudo_data = FALSE,
  number_max_dim = NULL,
  verbose_lvl = 2
)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>The data, must be provided as a matrix with each row as an observation.</td>
</tr>
<tr>
<td>p_value_for_dim_red</td>
<td>a p_value for the localised dimension reduction test</td>
</tr>
<tr>
<td>n_trees</td>
<td>Number of trees</td>
</tr>
<tr>
<td>conect_loo_weights</td>
<td>Defaults to FALSE. Allows to use an automatic re-weighting of the trees in the forest, based on leave-one-out considerations.</td>
</tr>
<tr>
<td>min_node_size</td>
<td>The minimum number of observation available in a leaf to initialise a split.</td>
</tr>
</tbody>
</table>
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 (none) or an integer. bigger the integer bigger the output level.

Details
This class implements the bagging of CORT models, with an oob error minimisation in the weights.

Value
a CortForest object that can be fitted easily to produce a copula estimate.

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
**kendall_func**

**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**

```r
cop <- cbCopula(LifeCycleSavings[, m = 5])
dCopula(rep(0,5), cop)
dCopula(rep(0.5,5), cop)
dCopula(rep(1,5), cop)
```

<table>
<thead>
<tr>
<th>kendall_func</th>
<th><strong>Kendall function</strong></th>
</tr>
</thead>
</table>

**Description**

Compute the kendall cdf from the model in a point t

**Usage**

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

### S4 method for signature 'Cort'

```r
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**

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

*Loss of the model*

**Description**

Compute the loss of the model

**Usage**

```r
loss(object)
```

```r
## 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**

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

---

pCopula  

*Copula density*

**Description**

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

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 density 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(LifeCycleSavings, m = 5)
pCopula(rep(0,5),cop) == 0
pCopula(rep(0.5,5),cop)
pCopula(rep(1,5),cop) == 1
Description

Compute, as a cort tree, the projection on a smaller set of dimensions of a cort tree.

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))

Description

Compute the L2 norm of the model
### quad_prod

#### Usage

quad_norm(object)

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

```r
## 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 Cort

#### Examples

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

---

### quad_prod

**Quadratic product of 2 trees**

#### Description

Compute the L2 quadratic product of 2 trees

#### Usage

quad_prod(object, other_tree)

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

#### Arguments

- **object**: the tree
- **other_tree**: the other tree
quad_prod_with_data

Value
the quadratic product between the trees

Functions

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

Examples

cop <- Cort(LifeCycleSavings[,1:3])
quad_prod(cop,cop) == quad_norm(cop)

quad_prod_with_data

Quadratic product with data of the model

Description
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 variables simulation

Description

This function simulate random variables from a 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

the density of the copula on each observation

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(LifeCycleSavings, m = 5)
xx <- rCopula(1000, cop)

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 available for the given copula.

This function computes the measure of the copula according to the algorithm proposed by the referenced paper.

Value

the measure of the copula.

References

vCopula

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>biv_rho</td>
<td>2</td>
</tr>
<tr>
<td>biv_rho,Cort-method (biv_rho)</td>
<td>2</td>
</tr>
<tr>
<td>biv_tau</td>
<td>3</td>
</tr>
<tr>
<td>biv_tau,Cort-method (biv_tau)</td>
<td>3</td>
</tr>
<tr>
<td>cbCopula (cbCopula-Class)</td>
<td>4</td>
</tr>
<tr>
<td>cbkmCopula (cbkmCopula-Class)</td>
<td>5</td>
</tr>
<tr>
<td>constraint_infl</td>
<td>6</td>
</tr>
<tr>
<td>constraint_infl,Cort-method</td>
<td>6</td>
</tr>
<tr>
<td>constraint_infl,CortForest-method</td>
<td>6</td>
</tr>
<tr>
<td>ConvexCombCopula</td>
<td>7</td>
</tr>
<tr>
<td>Cort (Cort-Class)</td>
<td>8</td>
</tr>
<tr>
<td>Cort-Class</td>
<td>8</td>
</tr>
<tr>
<td>CortForest (CortForest-Class)</td>
<td>9</td>
</tr>
<tr>
<td>CortForest-Class</td>
<td>9</td>
</tr>
<tr>
<td>dCopula</td>
<td>10</td>
</tr>
<tr>
<td>dCopula,matrix,cbCopula-method</td>
<td>10</td>
</tr>
<tr>
<td>dCopula,matrix,Cort-method</td>
<td>10</td>
</tr>
<tr>
<td>dCopula,matrix,CortForest-method</td>
<td>10</td>
</tr>
<tr>
<td>kendall_func</td>
<td>11</td>
</tr>
<tr>
<td>kendall_func,Cort-method</td>
<td>11</td>
</tr>
<tr>
<td>loss</td>
<td>12</td>
</tr>
<tr>
<td>loss,Cort-method (loss)</td>
<td>12</td>
</tr>
<tr>
<td>pCopula</td>
<td>12</td>
</tr>
<tr>
<td>pCopula,matrix,cbCopula-method</td>
<td>12</td>
</tr>
<tr>
<td>pCopula,matrix,CortForest-method</td>
<td>12</td>
</tr>
<tr>
<td>project_on_dims</td>
<td>14</td>
</tr>
<tr>
<td>project_on_dims,Cort-method</td>
<td>14</td>
</tr>
<tr>
<td>quad_norm</td>
<td>14</td>
</tr>
<tr>
<td>quad_norm,Cort-method (quad_norm)</td>
<td>14</td>
</tr>
<tr>
<td>quad_norm,CortForest-method</td>
<td>14</td>
</tr>
<tr>
<td>quad_prod</td>
<td>15</td>
</tr>
<tr>
<td>quad_prod,Cort,Cort-method</td>
<td>15</td>
</tr>
<tr>
<td>quad_prod_with_data</td>
<td>16</td>
</tr>
<tr>
<td>quad_prod_with_data,Cort-method</td>
<td>16</td>
</tr>
<tr>
<td>rCopula</td>
<td>17</td>
</tr>
<tr>
<td>rCopula,numeric,cbCopula-method</td>
<td>17</td>
</tr>
<tr>
<td>rCopula,numeric,cbkmCopula-method</td>
<td>17</td>
</tr>
<tr>
<td>rCopula,numeric,ConvexCombCopula-method</td>
<td>17</td>
</tr>
<tr>
<td>rCopula,numeric,Cort-method</td>
<td>17</td>
</tr>
<tr>
<td>rCopula,numeric,CortForest-method</td>
<td>17</td>
</tr>
<tr>
<td>rCopula,numeric,CortForest-method</td>
<td>17</td>
</tr>
<tr>
<td>vCopula</td>
<td>18</td>
</tr>
<tr>
<td>vCopula,matrix,matrix,Copula</td>
<td>18</td>
</tr>
<tr>
<td>vCopula,matrix,matrix-method</td>
<td>18</td>
</tr>
</tbody>
</table>