Package ‘cort’

May 14, 2020

Title Some Empiric and Nonparametric Copula Models

Version 0.3.1

Description Provides S4 classes and methods to fit several copula models: The classic empiri-
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 proposed by Laverny, Maume-Deschamps, Masiello and Rul-
lière (2020) <arXiv:2005.02912>, including the localised dimension reduction, which fits a copu-
ula by recursive splitting of the copula domain. We also provide an efficient way of mixing copu-
las, 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 7.0.2

Depends R (>= 2.10)

Imports Rdpack, methods, purrr, nloptr, osqp, Rcpp, 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.R' 'generics.R' 'ConvexCombCopula.R'

'empiricalCopula.R' 'Cort.R' 'CortForest.R' 'RcppExports.R'

'chCopula.R' 'cbkmCopula.R' 'cort-package.R' 'data.R'

VignetteBuilder knitr

RdMacros Rdpack

LinkingTo Rcpp

1
**biv_rho**

Spearman’s rho matrix of a copula

Computes the bivariate Spearman’s rho matrix for a copula.

**Usage**

```r
biv_rho(copula)
```

## S4 method for signature 'Cort'

biv_rho(copula)
**biv_tau**

**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**

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

---

**Description**

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

**Usage**

```r
biv_tau(copula)
```

```r
## 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**

```r
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). Asymptotics 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 \( I = \{ i = (i_1, \ldots, i_d) \subset \{1, \ldots, m\}^d \} \) 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 \( \mathbb{R} \), that is:

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

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 \( \forall i \in \{1, \ldots, d\} \), \( R_i^1, \ldots, R_i^d \) be the marginal ranks for the variable \( i \).

• \( \forall x \in I^d \) let \( \hat{\mu}((0, x)) = \frac{1}{n} \sum_{k=1}^{n} I_{R_i^1 \leq x_1, \ldots, R_i^d \leq x_d} \)
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**


---

**cbkmCopula-Class**

**Checkerboard with known margins**

**Description**

cbkmCopula constructor
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. See the corresponding vignette for more details.

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))
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

**Details**

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

**References**


constraint_infl

**Constraint influence of the model**

**Description**

Compute the constraint influence of the model

**Usage**

```r
constraint_infl(object)
```

**Arguments**

```r
object          the copula object
```
ConvexCombCopula-Class

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

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

ConvexCombCopula-Class

*Convex Combination of copulas.*

Description

ConvexCombCopula class

Usage

```r
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.

Value

- a ConvexCombCopula object
**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,
    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 localised dimension reduction test
- **min_node_size**: The minimum number of observation 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 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 optimisation. You can pass a call to `osqp::osqpSettings`, or `NULL` for defaults.
- **N**: The number of bootstrap resamples for `p_values` computations.
- **force_grid**: boolean. set to `TRUE` to force breakpoint to be on the n-checkerboard grid.
Details
This class implements the CORT algorithm to fit a multivariate copula using piece constant density. See O. Laverny, V. Maume-Deschamps, E. Masiello and D. Rullière (2020) for the details of this density estimation procedure.

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

References

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

CortForest-Class

Description
CortForest class

Usage
CortForest(
  x,
  p_value_for_dim_red = 0.75,
  n_trees = 10,
  compute_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 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.

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 oob 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.

Value

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

References


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(LifeCycleSavings,m = 5)
dCopula(rep(0,5),cop)
dCopula(rep(0.5,5),cop)
dCopula(rep(1,5),cop)

funcdep_data

Dataset funcdep_data

Description

This dependence structure is constructed by applying the function :

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

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

Usage

funcdep_data

Format

A matrix with 500 rows and 3 columns

Details

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

References


Dataset impossible_data

Description

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

\[ h(u) = \left( u_1, \frac{u_2}{2} + \frac{1}{2} I_{u_1 \notin \left( \frac{1}{3}, \frac{2}{3} \right)} \right) \]

to a 200x2 uniform sample, and taking ranks.

Usage

impossible_data

Format

A matrix with 200 rows and 2 columns

Details

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

References

**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)
```
Description

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(LifeCycleSavings[,1:3])
loss(cop)

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 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
**project_on_dims**

*Projection on smaller dimensions*

**Description**

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

**Usage**

```r
project_on_dims(object, dims)
```

**Value**

other cort object

**Examples**

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

---

**quad_norm**

*Quadratic norm of the model*

**Description**

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(LifeCycleSavings[,1:3])
quadr_norm(cop)

Description

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
quad_prod_with_data

Value
the quadratic product between the trees

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

Examples

```r
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

```r
quad_prod_with_data(object)
```

## S4 method for signature 'Cort'
```r
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

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

```r
cop <- cbCopula(LifeCycleSavings, 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

**Details**

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

**References**


---

**vCopula**  
*Copula volume on hyper-boxes*

**Description**

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

**Usage**

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

## S4 method for signature 'matrix, matrix'

```r
vCopula(u, v, copula)
```
**vCopula**

**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**


**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

*Topic **datasets**
  - clayton_data, 7
  - funcdep_data, 12
  - impossible_data, 13
  - recoveryyourself_data, 21

biv_rho, 2
biv_rho,Cort-method(biv_rho), 2
biv_tau, 3
biv_tau,Cort-method(biv_tau), 3
cbCopula(cbcCopula-Class), 4
cbCopula-Class, 4
cbkmCopula(cbckmCopula-Class), 5
cbkmCopula-Class, 5
clayton_data, 7
constraint_infl, 7
constraint_infl,Cort-method
  (constraint_infl), 7
constraint_infl,CortForest-method
  (constraint_infl), 7
ConvexCombCopula
  (ConvexCombCopula-Class), 8
ConvexCombCopula-Class, 8
Cort(Cort-Class), 9
Cort-Class, 9
CortForest(CortForest-Class), 10
CortForest-Class, 10
dCopula, 11
dCopula,matrix,cbCopula-method
  (dCopula), 11
dCopula,matrix,Cort-method(dCopula), 11
dCopula,matrix,CortForest-method
  (dCopula), 11
funcdep_data, 12
impossible_data, 13
kendall_func, 14
kendall_func,Cort-method
  (kendall_func), 14
loss, 15
loss,Cort-method(loss), 15
pCopula, 15
pCopula,matrix,cbCopula-method
  (pCopula), 15
pCopula,matrix,cbkmCopula-method
  (pCopula), 15
pCopula,matrix,ConvexCombCopula-method
  (pCopula), 15
pCopula,matrix,Cort-method(pCopula), 15
pCopula,matrix,CortForest-method
  (pCopula), 15
project_on_dims, 17
project_on_dims,Cort-method
  (project_on_dims), 17
quad_norm, 17
quad_norm,Cort-method(quad_norm), 17
quad_norm,CortForest-method
  (quad_norm), 17
quad_prod, 18
quad_prod,Cort,Cort-method(quad_prod), 18
quad_prod_with_data, 19
quad_prod_with_data,Cort-method
  (quad_prod_with_data), 19
rCopula, 20
rCopula,numeric,cbCopula-method
  (rCopula), 20
rCopula,numeric,cbkmCopula-method
  (rCopula), 20
rCopula,numeric,ConvexCombCopula-method
  (rCopula), 20
rCopula,numeric,Cort-method(rCopula), 20
rCopula, numeric, CortForest-method (rCopula), 20
recoveryourself_data, 21

vCopula, 21
vCopula, matrix, matrix, Copula (vCopula), 21
vCopula, matrix, matrix-method (vCopula), 21