Package 'rvinecopulib'

May 9, 2018

Type Package

Title High Performance Algorithms for Vine Copula Modeling

Version 0.2.8.1.0

Description Provides an interface to 'vinecopulib', a C++ library for vine copula modeling based on 'Boost' and 'Eigen'. The 'rvinecopulib' package implements the core features of the popular 'VineCopula' package, in particular inference algorithms for both vine copula and bivariate copula models. Advantages over 'VineCopula' are a sleeker and more modern API, improved performances, especially in high dimensions, nonparametric and multi-parameter families. The 'rvinecopulib' package includes 'vinecopulib' as header-only C++ library (currently version 0.2.8). Thus users do not need to install 'vinecopulib' itself in order to use 'rvinecopulib'. Since their initial releases, 'vinecopulib' is licensed under the MIT License, and 'rvinecopulib' is licensed under the GNU GPL version 3.

License GPL-3 | file LICENSE

Encoding UTF-8

LazyData true

NeedsCompilation yes

Imports assertthat, cctools, graphics, grDevices, kde1d, lattice, Rcpp (>= 0.12.12), stats, utils

Suggests igraph, ggplot2, ggraph, testthat

LinkingTo BH, Rcpp, RcppEigen, RcppThread

SystemRequirements C++11

RoxygenNote 6.0.1

Author Thomas Nagler [aut, cre], Thibault Vatter [aut]

Maintainer Thomas Nagler <info@vinecopulib.org>

Repository CRAN

Date/Publication 2018-05-09 17:50:13 UTC

R topics documented:

`bicop` *Bivariate copula models*

Description

Bivariate copula models

Usage

```
bicop(data, family_set = "all", par_method = "mle",
  nonpar_method = "quadratic", mult = 1, selcrit = "bic", psi0 = 0.9,
  presel = TRUE, keep_data = TRUE, cores = 1)

bicop_dist(family = "indep", rotation = 0, parameters = numeric(0))
```

Arguments

`data`	a matrix or data.frame (copula data should have approximately uniform margins).
`family_set`	a character vector of families; see *Details* for additional options.
`par_method`	the estimation method for parametric models, either `"mle"` for maximum likelihood or `"itau"` for inversion of Kendall's tau (only available for one-parameter families and `"t"`.

nonpar_method	the estimation method for nonparametric models, either "constant" for the standard transformation estimator, or "linear"/"quadratic" for the local-likelihood approximations of order one/two.
mult	multiplier for the smoothing parameters of nonparametric families. Values larger than 1 make the estimate more smooth, values less than 1 less smooth.
selcrit	criterion for family selection, either "loglik", "aic", "bic", "mbic". For vinecop() there is the additional option "mbicv".
psi0	see mBICV().
presel	whether the family set should be thinned out according to symmetry characteristics of the data.
keep_data	whether the data should be stored (necessary for computing fit statistics and using fitted()).
cores	number of cores to use; if more than 1, estimation for multiple families is done in parallel.
family	the copula family, a string containing the family name (see *Details* for all possible families).
rotation	the rotation of the copula, one of 0, 90, 180, 270.
parameters	a vector or matrix of copula parameters.

Details

The implemented families are:

"indep" = Independence copula.
"gaussian" = Gaussian copula.
"t" = Student t copula.
"clayton" = Clayton copula.
"gumbel" = Gumbel copula.
"frank" = Frank copula.
"joe" = Joe copula.
"bb1" = BB1 copula.
"bb6" = BB6 copula.
"bb7" = BB7 copula.
"bb8" = BB8 copula.
"tll" = transformation kernel local likelihood, only for bicop().

In addition, the following convenience definitions can be used (and combined) with bicop:

"all" = all families.
"parametric" = parametric families.
"nonparametric" = nonparametric families.
"archimedean" = archimedean families.
"elliptical" = elliptical families.
"bbs" = BB families.
"oneparametric" = one parameter families.

"twoparametric" = two parameter families.
"itau" = one parameter families and Student t copula.
Partial matching is activated. For example, "gauss" is equivalent to "gaussian", or you can write "nonpar" instead of "nonparametric".

Value

Objects inheriting from bicop_dist for bicop_dist(), and bicop and bicop_dist for bicop().

Object from the bicop_dist class are lists containing:

- family, a character indicating the copula family.
- rotation, an integer indicating the rotation (i.e., either 0, 90, 180, or 270).
- parameters, a numeric vector or matrix of parameters.
- npars, a numeric with the (effective) number of parameters.

Additionally, objects from the bicop class contain:

- data (optionally, if keep_data = TRUE was used), the dataset that was passed to bicop().
- controls, a list with the set of fit controls that was passed to bicop().
- nobs, an integer with the number of observations that was used to fit the model.

Examples

```
## bicop_dist objects
bicop_dist("gaussian", 0, 0.5)
str(bicop_dist("gauss", 0, 0.5))
bicop <- bicop_dist("clayton", 90, 3)

## bicop objects
u <- rbicop(500, "gauss", 0, 0.5)
fit1 <- bicop(u, "par")
fit1
```

---

bicop_distributions *Bivariate copula distributions*

Description

Density, distribution function, random generation and h-functions (with their inverses) for the bivariate copula distribution.

Usage

```
dbicop(u, family, rotation, parameters)

pbicop(u, family, rotation, parameters)

rbicop(n, family, rotation, parameters, U = NULL)

hbicop(u, cond_var, family, rotation, parameters, inverse = FALSE)
```

Arguments

u	evaluation points, either a length 2 vector or a two-column matrix.
family	the copula family, a string containing the family name (see bicop for all possible families).
rotation	the rotation of the copula, one of 0, 90, 180, 270.
parameters	a vector or matrix of copula parameters.
n	number of observations. If 'length(n) > 1", the length is taken to be the number required.
U	optionally, an $n \times 2$ matrix of values in $(0, 1)$. The result is then the inverse Rosenblatt transform of U; if U is a matrix of independent $U(0, 1)$ variables, this simulates data from vinecop.
cond_var	either 1 or 2; cond_var = 1 conditions on the first variable, cond_var = 2 on the second.
inverse	whether to compute the h-function or its inverse.

Details

See bicop for the various implemented copula families.

H-functions (hbicop()) are conditional distributions derived from a copula. If $C(u, v) = P(U \leq u, V \leq v)$ is a copula, then

$$h_1(u, v) = P(V \leq v | U = u) = \partial C(u, v)/\partial u,$$

$$h_2(u, v) = P(U \leq u | V = v) = \partial C(u, v)/\partial v.$$

In other words, the H-function number refers to the conditioning variable. When inverting H-functions, the inverse is then taken with respect to the other variable, that is v when cond_var = 1 and u when cond_var = 2.

Value

dbicop() gives the density, pbicop() gives the distribution function, rbicop() generates random deviates, and hbicop() gives the h-functions (and their inverses).

The length of the result is determined by n for rbicop(), and the number of rows in u for the other functions.

The numerical arguments other than n are recycled to the length of the result.

Note

The functions can optionally be used with a bicop_dist object, e.g., `dbicop(c(0.1, 0.5), bicop_dist("indep"))`.

Examples

```
## evaluate the copula density
dbicop(c(0.1, 0.2), "clay", 90, 3)
dbicop(c(0.1, 0.2), bicop_dist("clay", 90, 3))

## evaluate the copula cdf
pbicop(c(0.1, 0.2), "clay", 90, 3)

## simulate data
plot(rbicop(500, "clay", 90, 3))

## h-functions
joe_cop <- bicop_dist("joe", 0, 3)
# h_1(0.1, 0.2)
hbicop(c(0.1, 0.2), 1, "bb8", 0, c(2, 0.5))
# h_2(0.1, 0.2)
hbicop(c(0.1, 0.2), 2, joe_cop)
# h_1^{-1}(0.1, 0.2)
hbicop(c(0.1, 0.2), 1, "bb8", 0, c(2, 0.5), inverse = TRUE)
# h_2^{-1}(0.1, 0.2)
hbicop(c(0.1, 0.2), 2, joe_cop, inverse = TRUE)
```

---

bicop_predict_and_fitted

Predictions and fitted values for a bivariate copula model

Description

Predictions of the density, distribution function, h-functions (with their inverses) for a bivariate copula model.

Usage

```
## S3 method for class 'bicop'
predict(object, newdata, what = "pdf", ...)

## S3 method for class 'bicop'
fitted(object, what = "pdf", ...)
```

Arguments

`object`	a `bicop` object.
`newdata`	points where the fit shall be evaluated.
`what`	what to predict, one of `"pdf"`, `"cdf"`, `"hfunc1"`, `"hfunc2"`, `"hinv1"`, `"hinv2"`.
`...`	unused.

Value

`fitted()` and `logLik()` have return values similar to `dbicop()`, `pbicop()`, and `hbicop()`.

Examples

```
# Simulate and fit a bivariate copula model
u <- rbicop(500, "gauss", 0, 0.5)
fit <- bicop(u, "par")

# Predictions
all.equal(predict(fit, u, "hfunc1"), fitted(fit, "hfunc1"))
```

`check_rvine_matrix` *R-vine matrices*

Description

R-vine matrices are compressed representations of the vine structure. It needs to satisfy several properties that can be checked by `check_rvine_matrix()`, see *Details*.

Usage

```
check_rvine_matrix(matrix)
```

Arguments

`matrix`	a quadratic matrix, see *Details*.

Details

The R-vine matrix notation in vinecopulib is different from the one in the VineCopula package. An example matrix is

```
1 1 1 1
2 2 2 0
3 3 0 0
4 0 0 0
```

which encodes the following pair-copulas:

tree	edge	pair-copulas
0	0	`(4, 1)`
	1	`(3, 1)`
	2	`(2, 1)`
1	0	`(4, 2; 1)`
	1	`(3, 2; 1)`
2	0	`(4, 3; 2, 1)`

Denoting by `M[i, j]` the matrix entry in row `i` and column `j` (the pair-copula index for edge `e` in tree `t` of a `d` dimensional vine is `(M[d - 1 - t, e], M[t, e]; M[t - 1, e], ..., M[0, e])`. Less formally,

1. Start with the counter-diagonal element of column `e` (first conditioned variable).
2. Jump up to the element in row `t` (second conditioned variable).
3. Gather all entries further up in column `e` (conditioning set).

A valid R-vine matrix must satisfy several conditions which are checked when `RVineMatrix()` is called:

1. The lower right triangle must only contain zeros.
2. The upper left triangle can only contain numbers between 1 and d.
3. The anti-diagonal must contain the numbers 1, ..., d.
4. The anti-diagonal entry of a column must not be contained in any column further to the right.
5. The entries of a column must be contained in all columns to the left.
6. The proximity condition must hold: For all t = 1, ..., d - 2 and e = 0, ..., d - t - 1 there must exist an index j > d, such that `(M[t, e], {M[0, e], ..., M[t-1, e]})` equals either `(M[d-j-1, j], {M[0, j], ..., M[t-1, j]})` or `(M[t-1, j], {M[d-j-1, j], M[0, j], ..., M[t-2, j]})`.

Condition 6 already implies conditions 2-5, but is more difficult to check by hand.

Value

Throws an error if `matrix` is not a valid R-vine matrix, otherwise `TRUE` is returned invisibly.

Examples

```
mat <- matrix(c(1, 2, 3, 4, 1, 2, 3, 0, 1, 2, 0, 0, 1, 0, 0, 0), 4, 4)
check_rvine_matrix(mat)

# throws an error
mat[4, 4] <- 5
try(check_rvine_matrix(mat))
```

---

getters *Extracts components of* `bicop_dist` *and* `vinecop_dist` *objects*

Description

Extracts either the structure matrix (for `vinecop_dist` only), or pair-copulas, their parameters, Kendall's taus, or families (for `bicop_dist` and `vinecop_dist`).

Usage

```
get_matrix(object)

get_pair_copula(object, tree = NA, edge = NA)

get_parameters(object, tree = NA, edge = NA)

get_ktau(object, tree = NA, edge = NA)

get_family(object, tree = NA, edge = NA)

get_all_pair_copulas(object, trees = NA)

get_all_parameters(object, trees = NA)

get_all_ktaus(object, trees = NA)

get_all_families(object, trees = NA)
```

Arguments

object	a bicop_dist, vinecop_dist or vine_dist object.
tree	tree index (not required if object is of class bicop_dist).
edge	edge index (not required if object is of class bicop_dist).
trees	the trees to extract from object (trees = NA extracts all trees).

Details

The get_matrix method (for vinecop_dist or vine_dist objects only) extracts the structure matrix (see check_rvine_matrix for more details).

The other get_xyz methods for vinecop_dist or vine_dist objects return the entries corresponding to the pair-copula indexed by its tree and edge. When object is of class bicop_dist, tree and edge are not required.

get_pair_copula = the pair-copula itself (see bicop).
get_parameters = the parameters of the pair-copula (i.e., a numeric scalar, vector, or matrix).
get_family = a character for the family (see bicop for implemented families).
get_ktau = a numeric scalar with the pair-copula Kendall's tau.

The get_all_xyz methods (for vinecop_dist or vine_dist objects only) return lists of lists, with each element corresponding to a tree in trees, and then elements of the sublists correspond to edges. The returned lists have two additional attributes:

"d" = the dimension of the model.
"trees" = the extracted trees.

Value

The structure matrix, or pair-copulas, their parameters, Kendall's taus, or families.

Examples

```
# specify pair-copulas
bicop <- bicop_dist("bb1", 90, c(3, 2))
pcs <- list(
    list(bicop, bicop),  # pair-copulas in first tree
    list(bicop)          # pair-copulas in second tree
)

# specify R-vine matrix
mat <- matrix(c(1, 2, 3, 1, 2, 0, 1, 0, 0), 3, 3)

# set up vine copula model
vc <- vinecop_dist(pcs, mat)

# get the structure matrix
all.equal(get_matrix(vc), mat)

# get pair-copulas
get_pair_copula(vc, 1, 1)
get_all_pair_copulas(vc)
all.equal(get_all_pair_copulas(vc), pcs, check.attributes = FALSE)
```

---

mBICV

calculates the vine copula Bayesian information criterion (vBIC), which is defined as

$\mathrm{BIC} = -2\,\mathrm{loglik}+\nu\ln(n), -2*\sum\_t = 1\hat{}(d-1)\{q\_tlog(\psi\_0\hat{}t)-(d-t-q\_t)log(1-\psi\_0\hat{}t)\}$

where loglik *is the log-likelihood and ν is the (effective) number of parameters of the model, t is the tree level ψ_0 is the prior probability of having a non-independence copula and q_t is the number of non-independence copulas in tree t. The vBIC is a consistent model selection criterion for parametric sparse vine copula models.*

Description

calculates the vine copula Bayesian information criterion (vBIC), which is defined as

$$\mathrm{BIC} = -2\,\mathrm{loglik} + \nu\ln(n), -2 * \sum_{t=1}^{(} d-1)\{q_t log(\psi_0^t) - (d-t-q_t)log(1-\psi_0^t)\}$$

where loglik is the log-likelihood and ν is the (effective) number of parameters of the model, t is the tree level ψ_0 is the prior probability of having a non-independence copula and q_t is the number of non-independence copulas in tree t. The vBIC is a consistent model selection criterion for parametric sparse vine copula models.

Usage

```
mBICV(object, psi0 = 0.9)
```

Arguments

object	a fitted vinecop object.
psi0	baseline prior probability of a non-independence copula.

---

par_to_ktau *Conversion between Kendall's tau and parameters*

Description

Conversion between Kendall's tau and parameters

Usage

```
par_to_ktau(family, rotation, parameters)

ktau_to_par(family, tau)
```

Arguments

family	a copula family (see bicop_dist()) or a bicop_dist object.
rotation	the rotation of the copula, one of 0, 90, 180, 270.
parameters	vector or matrix of copula parameters, not used when family is a bicop_dist object.
tau	Kendall's τ.

Examples

```
# the following are equivalent
par_to_ktau(bicop_dist("clayton", 0, 3))
par_to_ktau("clayton", 0, 3)

ktau_to_par("clayton", 0.5)
ktau_to_par(bicop_dist("clayton", 0, 3), 0.5)
```

plot.bicop_dist *Plotting tools for* bicop_dist *and* bicop *objects*

Description

There are several options for plotting bicop_dist objects. The density of a bivariate copula density can be visualized as surface/perspective or contour plot. Optionally, the density can be coupled with standard normal margins (default for contour plots).

Usage

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

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

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

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

Arguments

x	bicop_dist object.
type	plot type; either "surface" or "contour".
margins	options are: "unif" for the original copula density, "norm" for the transformed density with standard normal margins, "exp" with standard exponential margins, and "flexp" with flipped exponential margins. Default is "norm" for type = "contour", and "unif" for type = "surface".
size	integer; the plot is based on values on a $sizexsize$ grid, default is 100.
...	optional arguments passed to contour or wireframe.

See Also

bicop_dist, contour, wireframe

Examples

```
## construct bicop_dist object for a student t copula
obj <- bicop_dist(family = "t", rotation = 0, parameters = c(0.7,4))

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

```
contour(obj, margins = "unif")  # contour plot of copula density
```

---

plot.vinecop_dist *Plotting* vinecop_dist *and* vinecop *objects.*

Description

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

Usage

```
## S3 method for class 'vinecop_dist'
plot(x, tree = 1, var_names = "ignore",
  edge_labels = NULL, ...)

## S3 method for class 'vinecop'
plot(x, tree = 1, var_names = "ignore",
  edge_labels = NULL, ...)

## S3 method for class 'vinecop_dist'
contour(x, tree = "ALL", cex.nums = 1, ...)

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

Arguments

x	vinecop_dist object.
tree	"ALL" or integer vector; specifies which trees are plotted.
var_names	integer; specifies how to make use of variable names: "ignore" = variable names are ignored, "use" = variable names are used to annotate vertices, "legend" = uses numbers in plot and adds a legend for variable names.
edge_labels	character; either a vector of edge labels or one of the following: "family" = pair-copula family (see bicop_dist), "tau" = pair-copula Kendall's tau "family_tau" = pair-copula family and Kendall's tau, "pair" = for the name of the involved variables.
...	Unused for plot and passed to contour.bicop for contour.
cex.nums	numeric; expansion factor for font of the numbers.

Details

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

Author(s)

Thomas Nagler, Thibault Vatter

See Also

`vinecop_dist`, `plot.bicop`

Examples

```
# set up vine copula model
d <- 20
n <- 2e2
u <- matrix(runif(n*d), n, d)
vc <- vinecop(u, "indep")

# plot
plot(vc, tree = c(1,2))
plot(vc, edge_labels = "pair")

# set up another vine copula model
pcs <- lapply(1:3, function(j) # pair-copulas in tree j
    lapply(runif(4-j), function(cor) bicop_dist("gaussian", 0, cor)))
mat <- matrix(c(1, 2, 3, 4, 1, 2, 3, 0, 1, 2, 0, 0, 1, 0, 0, 0), 4, 4)
vc <- vinecop_dist(pcs, mat)

# contour plot
contour(vc)
```

---

pseudo_obs *Pseudo-Observations*

Description

Compute the pseudo-observations for the given data matrix.

Usage

```
pseudo_obs(x, ties_method = "average", lower_tail = TRUE)
```

Arguments

x	vector or matrix random variates to be converted (column wise) to pseudo-observations.
ties_method	similar to ties.method of rank() (only "average", "first" and "random" currently available).
lower_tail	logical which, if 'FALSE", returns the pseudo-observations when applying the empirical marginal survival functions.

Details

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

The pseudo-observations can thus also be computed by component-wise applying the empirical distribution functions to the data and scaling the result by $n/(n+1)$. This asymptotically negligible scaling factor is used to force the variates to fall inside the open unit hypercube, for example, to avoid problems with density evaluation at the boundaries.

When lower_tail = FALSE, then pseudo_obs() simply returns 1 - pseudo_obs().

Value

a vector of matrix of the same dimension as the input containing the pseudo-observations.

Examples

```
# pseudo-observations for a vector
pseudo_obs(rnorm(10))

# pseudo-observations for a matrix
pseudo_obs(cbind(rnorm(10), rnorm(10)))
```

---

rvinecopulib *High Performance Algorithms for Vine Copula Modeling*

Description

Provides an interface to 'vinecopulib', a C++ library for vine copula modeling based on 'Boost' and 'Eigen'. The 'rvinecopulib' package implements the core features of the popular 'VineCopula' package, in particular inference algorithms for both vine copula and bivariate copula models. Advantages over 'VineCopula' are a sleeker and more modern API, improved performances, especially in high dimensions, nonparametric and multi-parameter families. The 'rvinecopulib' package includes 'vinecopulib' as header-only C++ library (currently version 0.2.8). Thus users do not need to install 'vinecopulib' itself in order to use 'rvinecopulib'. Since their initial releases, 'vinecopulib' is licensed under the MIT License, and 'rvinecopulib' is licensed under the GNU GPL version 3.

Author(s)

Thomas Nagler, Thibault Vatter

Examples

```
## bicop_dist objects
bicop_dist("gaussian", 0, 0.5)
str(bicop_dist("gauss", 0, 0.5))
bicop <- bicop_dist("clayton", 90, 3)

## bicop objects
u <- rbicop(500, "gauss", 0, 0.5)
fit1 <- bicop(u, "par")
fit1

## vinecop_dist objects
## specify pair-copulas
bicop <- bicop_dist("bb1", 90, c(3, 2))
pcs <- list(
    list(bicop, bicop),  # pair-copulas in first tree
    list(bicop)          # pair-copulas in second tree
)
## specify R-vine matrix
mat <- matrix(c(1, 2, 3, 1, 2, 0, 1, 0, 0), 3, 3)
## build the vinecop_dist object
vc <- vinecop_dist(pcs, mat)
summary(vc)

## vinecop objects
u <- sapply(1:3, function(i) runif(50))
vc <- vinecop(u, "par")
summary(vc)

## vine_dist objects
vc <- vine_dist(list(name = "norm"), pcs, mat)
summary(vc)

#' ## vine objects
x <- sapply(1:3, function(i) rnorm(50))
vc <- vine(x, copula_controls = list(family_set = "par"))
summary(vc)
```

`truncate_model` *extract a truncated sub-vine based on truncation level supplied by user.*

Description

extract a truncated sub-vine based on truncation level supplied by user.

Usage

```
truncate_model(object, trunc_lvl = NA)
```

Arguments

object	a vinecop or a vine object.
trunc_lvl	truncation level for the vine copula.

---

vine *Vine copula models*

Description

Automated fitting or creation of custom vine copula models

Usage

```
vine(data, margins_controls = list(mult = NULL, xmin = NaN, xmax = NaN, bw =
  NA), copula_controls = list(family_set = "all", matrix = NA, par_method =
  "mle", nonpar_method = "constant", mult = 1, selcrit = "bic", psi0 = 0.9,
  presel = TRUE, trunc_lvl = Inf, tree_crit = "tau", threshold = 0, keep_data =
  TRUE, show_trace = FALSE, cores = 1))

vine_dist(margins, pair_copulas, matrix)
```

Arguments

data	a matrix or data.frame.
margins_controls	a list with arguments to be passed to kde1d::kde1d(). Currently, there can be • mult numeric; all bandwidths for marginal kernel density estimation are multiplied with mult_1d. Defaults to log(1 + d) where d is the number of variables after applying cctools::expand_as_numeric(). • xmin numeric vector of length d; see kde1d::kde1d(). • xmax numeric vector of length d; see kde1d::kde1d(). • bw numeric vector of length d; see kde1d::kde1d().
copula_controls	a list with arguments to be passed to vinecop().
margins	A list with with each element containing the specification of a marginal stats::Distributions. Each marginal specification should be a list with containing at least the name and optionally the parameters, e.g. list(list(name = "norm"), list(name = "norm", mu = 1), list(n Note that parameters that have no default values have to be provided. Furthermore, if margins has length one, it will be recycled for every component.
pair_copulas	A nested list of 'bicop_dist' objects, where pair_copulas[[t]][[e]] corresponds to the pair-copula at edge e in tree t.

matrix — a quadratic matrix specifying the structure matrix (see check_rvine_matrix()); for vinecop_dist(), the dimension must be length(pair_copulas)-1; for vinecop(), matrix = NA performs automatic structure selection.

Details

vine_dist() creates a vine copula by specifying the margins, a nested list of bicop_dist objects and a quadratic structure matrix.

vine() provides automated fitting for vine copula models. margins_controls is a list with the same parameters as kde1d::kde1d() (except for x). copula_controls is a list with the same parameters as vinecop() (except for data).

Value

Objects inheriting from vine_dist for vine_dist(), and vine and vine_dist for vine().

Objects from the vine_dist class are lists containing:

- margins, a list of marginals (see below).
- copula, an object of the class vinecop_dist, see vinecop_dist().

For objects from the vine class, copula is also an object of the class vine, see vinecop(). Additionally, objects from the vine class contain:

- margins_controls, a list with the set of fit controls that was passed to kde1d::kde1d() when estimating the margins.
- copula_controls, a list with the set of fit controls that was passed to vinecop() when estimating the copula.
- data (optionally, if keep_data = TRUE was used), the dataset that was passed to vinecop().
- nobs, an integer containing the number of observations that was used to fit the model.

Concerning margins:

- For objects created with vine_dist(), it simply corresponds to the margins argument.
- For objects created with vine(), it is a list of objects of class kde1d, see kde1d::kde1d().

Examples

```
# specify pair-copulas
bicop <- bicop_dist("bb1", 90, c(3, 2))
pcs <- list(
    list(bicop, bicop),  # pair-copulas in first tree
    list(bicop)          # pair-copulas in second tree
 )

# specify R-vine matrix
mat <- matrix(c(1, 2, 3, 1, 2, 0, 1, 0, 0), 3, 3)

# set up vine copula model with Gaussian margins
vc <- vine_dist(list(name = "norm"), pcs, mat)
```

```
# show model
summary(vc)

# simulate some data
x <- rvine(50, vc)

# estimate a vine copula model
fit <- vine(x, copula_controls = list(family_set = "par"))
summary(fit)
```

---

vinecop *Vine copula models*

Description

Automated fitting or creation of custom vine copula models

Usage

```
vinecop(data, family_set = "all", matrix = NA, par_method = "mle",
  nonpar_method = "constant", mult = 1, selcrit = "bic", psi0 = 0.9,
  presel = TRUE, trunc_lvl = Inf, tree_crit = "tau", threshold = 0,
  keep_data = TRUE, show_trace = FALSE, cores = 1)

vinecop_dist(pair_copulas, matrix)
```

Arguments

data	a matrix or data.frame (copula data should have approximately uniform margins).
family_set	a character vector of families; see bicop() for additional options.
matrix	a quadratic matrix specifying the structure matrix (see check_rvine_matrix()); for vinecop_dist(), the dimension must be length(pair_copulas)-1; for vinecop(), matrix = NA performs automatic structure selection.
par_method	the estimation method for parametric models, either "mle" for maximum likelihood or "itau" for inversion of Kendall's tau (only available for one-parameter families and "t".
nonpar_method	the estimation method for nonparametric models, either "constant" for the standard transformation estimator, or "linear"/"quadratic" for the local-likelihood approximations of order one/two.
mult	multiplier for the smoothing parameters of nonparametric families. Values larger than 1 make the estimate more smooth, values less than 1 less smooth.
selcrit	criterion for family selection, either "loglik", "aic", "bic", "mbic". For vinecop() there is the additional option "mbicv".

psi0	prior probability of a non-independence copula (only used for selcrit = "mbic" and selcrit = "mbicv").
presel	whether the family set should be thinned out according to symmetry characteristics of the data.
trunc_lvl	the truncation level of the vine copula; Inf means no truncation, NA indicates that the truncation level should be selected automatically by mBICV().
tree_crit	the criterion for tree selection, one of "tau", "rho", "hoeffd", or "mcor" for Kendall's τ, Spearman's ρ, Hoeffding's D, and maximum correlation, respectively.
threshold	for thresholded vine copulas; NA indicates that the threshold should be selected automatically by mBICV().
keep_data	whether the data should be stored (necessary for computing fit statistics and using fitted()).
show_trace	logical; whether a trace of the fitting progress should be printed.
cores	number of cores to use; if more than 1, estimation of pair copulas within a tree is done in parallel.
pair_copulas	A nested list of 'bicop_dist' objects, where pair_copulas[[t]][[e]] corresponds to the pair-copula at edge e in tree t.

Details

vinecop_dist() creates a vine copula by specifying a nested list of bicop_dist objects and a quadratic structure matrix.

vinecop() provides automated fitting for vine copula models. The function inherits the parameters of bicop(). Optionally, a quadratic matrix can be used as input to pre-specify the vine structure. tree_crit describes the criterion for tree selection, one of "tau", "rho", "hoeffd" for Kendall's tau, Spearman's rho, and Hoeffding's D, respectively. Additionally, threshold allows to threshold the tree_crit and trunc_lvl to truncate the vine copula, with threshold_sel and trunc_lvl_sel to automatically select both parameters.

Value

Objects inheriting from vinecop_dist for vinecop_dist(), and vinecop and vinecop_dist for vinecop().

Object from the vinecop_dist class are lists containing:

- pair_copulas, a list of lists. Each element of pair_copulas corresponds to a tree, which is itself a list of bicop_dist objects, see bicop_dist().
- matrix, an R-vine matrix, namely a compressed representation of the vine structure, see check_rvine_matrix().
- npars, a numeric with the number of (effective) parameters.

For objects from the vinecop class, elements of the sublists in pair_copulas are also bicop objects, see bicop(). Additionally, objects from the vinecop class contain:

- threshold, the (set or estimated) threshold used for thresholding the vine.

- `data` (optionally, if `keep_data = TRUE` was used), the dataset that was passed to `vinecop()`.
- `controls`, a `list` with the set of fit controls that was passed to `vinecop()`.
- `nobs`, an `integer` with the number of observations that was used to fit the model.

Examples

```
# specify pair-copulas
bicop <- bicop_dist("bb1", 90, c(3, 2))
pcs <- list(
    list(bicop, bicop),  # pair-copulas in first tree
    list(bicop)          # pair-copulas in second tree
 )

# specify R-vine matrix
mat <- matrix(c(1, 2, 3, 1, 2, 0, 1, 0, 0), 3, 3)

# set up vine copula model
vc <- vinecop_dist(pcs, mat)

# show model
summary(vc)

# simulate some data
u <- rvinecop(50, vc)

# estimate a vine copula model
fit <- vinecop(u, "par")
fit
summary(fit)
str(fit, 3)
```

`vinecop_distributions` *Vine copula distributions*

Description

Density, distribution function and random generation for the vine copula distribution.

Usage

```
dvinecop(u, vinecop)

pvinecop(u, vinecop, n_mc = 10^4)

rvinecop(n, vinecop, U = NULL)
```

Arguments

u	evaluation points, either a length d vector or a d-column matrix, where d is the number of variables in the vine.
vinecop	an object of class "vinecop_dist".
n_mc	number of samples used for quasi Monte Carlo integration.
n	number of observations.
U	optionally, an $n \times d$ matrix of values in $(0, 1)$. The result is then the inverse Rosenblatt transform of U; if U is a matrix of independent $U(0, 1)$ variables, this simulates data from vinecop.

Details

See vinecop for the estimation and construction of vine copula models. Here, the density, distribution function and random generation for the vine copulas are standard.

The Rosenblatt transform (Rosenblatt, 1952) $U = T(V)$ of a random vector $V = (V_1, \ldots, V_d)$ C is defined as

$$U_1 = V_1, U_2 = C(V_2|V_1), \ldots, U_d = C(V_d|V_1, \ldots, V_{d-1}),$$

where $C(v_k|v_1, \ldots, v_{k-1})$ is the conditional distribution of V_k given $V_1 \ldots, V_{k-1}, k = 2, \ldots, d$. The vector V are then independent standard uniform variables. The inverse operation

$$V_1 = U_1, V_2 = C^{-1}(U_2|U_1), \ldots, V_d = C^{-1}(U_d|U_1, \ldots, U_{d-1}),$$

can can be used to simulate from a copula. For any copula C, if U is a vector of independent random variables, $V = T^{-1}(U)$ has distribution C.

Value

dvinecop() gives the density, pvinecop() gives the distribution function, and rvinecop() generates random deviates.

The length of the result is determined by n for rvinecop(), and the number of rows in u for the other functions.

The vinecop object is recycled to the length of the result.

Examples

```
# specify pair-copulas
bicop <- bicop_dist("bb1", 90, c(3, 2))
pcs <- list(
    list(bicop, bicop),  # pair-copulas in first tree
    list(bicop)          # pair-copulas in second tree
 )

# specify R-vine matrix
mat <- matrix(c(1, 2, 3, 1, 2, 0, 1, 0, 0), 3, 3)

# set up vine copula model
vc <- vinecop_dist(pcs, mat)
```

```
# simulate from the model
u <- rvinecop(200, vc)
pairs(u)

# evaluate the density and cdf
dvinecop(u[1, ], vc)
pvinecop(u[1, ], vc)

# get single pair copula
get_pair_copula(vc, 1, 1)

# get all pair copulas
get_all_pair_copulas(vc)

# get vine matrix
get_matrix(vc)

# extract a truncated sub-vine based on truncation level supplied by user
truncate_model(vc, 1)
```

`vinecop_predict_and_fitted` *Predictions and fitted values for a vine copula model*

Description

Predictions of the density and distribution function for a vine copula model.

Usage

```
## S3 method for class 'vinecop'
predict(object, newdata, what = "pdf", n_mc = 10^4, ...)

## S3 method for class 'vinecop'
fitted(object, what = "pdf", n_mc = 10^4, ...)
```

Arguments

`object`	a `vinecop` object.
`newdata`	points where the fit shall be evaluated.
`what`	what to predict, either `"pdf"` or `"cdf"`.
`n_mc`	number of samples used for quasi Monte Carlo integration when `what = "cdf"`.
`...`	unused.

Value

`fitted()` and `predict()` have return values similar to `dvinecop()` and `pvinecop()`.

Examples

```
u <- sapply(1:5, function(i) runif(50))
fit <- vinecop(u, "par")
all.equal(predict(fit, u), fitted(fit))
```

`vine_distributions` *Vine based distributions*

Description

Density, distribution function and random generation for the vine based distribution.

Usage

```
dvine(x, vine)

pvine(x, vine, n_mc = 10^4)

rvine(n, vine, U = NULL)
```

Arguments

`x`	evaluation points, either a length d vector or a d-column matrix, where d is the number of variables in the vine.
`vine`	an object of class `"vine_dist"`.
`n_mc`	number of samples used for quasi Monte Carlo integration.
`n`	number of observations.
`U`	optionally, an $n \times d$ matrix of values in $(0, 1)$. The result is then the inverse Rosenblatt transform of `U`; if `U` is a matrix of independent $U(0, 1)$ variables, this simulates data from `vine`.

Details

See vine for the estimation and construction of vine models. Here, the density, distribution function and random generation for the vine distributions are standard.

The functions are based on `dvinecop()`, `pvinecop()` and `rvinecop()` for vinecop objects, and either `kde1d::dkde1d()`, `kde1d::pkde1d()` and `kde1d::qkde1d()` for estimated vines (i.e., output of `vine()`), or the standard *d/p/q-xxx* from stats::Distributions for customly created vines (i.e., output of `vine_dist()`)

Value

`dvine()` gives the density, `pvine()` gives the distribution function, and `rvine()` generates random deviates.

The length of the result is determined by `n` for `rvine()`, and the number of rows in `u` for the other functions.

The `vine` object is recycled to the length of the result.

Examples

```
# specify pair-copulas
bicop <- bicop_dist("bb1", 90, c(3, 2))
pcs <- list(
    list(bicop, bicop),  # pair-copulas in first tree
    list(bicop)          # pair-copulas in second tree
 )

# specify R-vine matrix
mat <- matrix(c(1, 2, 3, 1, 2, 0, 1, 0, 0), 3, 3)

# set up vine copula model
vc <- vine_dist(list(name = "norm"), pcs, mat)

# simulate from the model
x <- rvine(200, vc)
pairs(x)

# evaluate the density and cdf
dvine(x[1, ], vc)
pvine(x[1, ], vc)
```

`vine_predict_and_fitted` *Predictions and fitted values for a vine copula model*

Description

Predictions of the density and distribution function for a vine copula model.

Usage

```
## S3 method for class 'vine'
predict(object, newdata, what = "pdf", n_mc = 10^4, ...)

## S3 method for class 'vine'
fitted(object, what = "pdf", n_mc = 10^4, ...)
```

Arguments

`object`	a `vine` object.
`newdata`	points where the fit shall be evaluated.
`what`	what to predict, either `"pdf"` or `"cdf"`.
`n_mc`	number of samples used for quasi Monte Carlo integration when `what = "cdf"`.
`...`	unused.

Value

`fitted()` and `predict()` have return values similar to `dvine()` and `pvine()`.

Examples

```
x <- sapply(1:5, function(i) rnorm(50))
fit <- vine(x, copula_controls = list(family_set = "par"))
all.equal(predict(fit, x), fitted(fit))
```

Index