Package 'HAC'

March 14, 2022

Type Package

Title Estimation, Simulation and Visualization of Hierarchical Archimedean Copulae (HAC)

Version 1.1-0

Date 2022-03-08

Author Ostap Okhrin <ostap.okhrin@tu-dresden.de> and Alexander Ristig

Maintainer Martin Waltz <martin.waltz@tu-dresden.de>

LazyLoad yes

Depends R (>= 3.1.0), copula

Imports graphics, stats

Description Package provides the estimation of the structure and the parameters, sampling methods and structural plots of Hierarchical Archimedean Copulae (HAC).

License GPL (>= 3)

URL https://tu-dresden.de/bu/verkehr/ivw/osv/die-professur/inhaber-in

NeedsCompilation no

Repository CRAN

Date/Publication 2022-03-14 18:10:14 UTC

R topics documented:

`aggregate.hac` *Aggregation of variables*

Description

`aggregate` tests, whether the absolute difference of the parameters of two subsequent nodes is smaller than a constant, i.e. $|\theta_2 - \theta_1| < \epsilon$, where θ_i denotes the dependency parameter with $\theta_2 < \theta_1$, $\epsilon \geq 0$. If the absolute difference is smaller than the constant, the variables of the nodes are aggregated in a single node with new dependency parameter, e.g. $\theta_{new} = (\theta_1 + \theta_2)/2$. This procedure is applied to all consecutive nodes of the HAC `x`.

Usage

```
## S3 method for class 'hac'
aggregate(x, epsilon = 0, method = "mean", ...)
```

Arguments

`x`	an object of the class `hac`.
`epsilon`	scalar ≥ 0.
`method`	determines, whether the new parameter is the `"mean"`, `"min"` or `"max"` of the fused parameters.
`...`	further arguments passed to or from other methods.

Value

an object of the class `hac`.

See Also

`hac`

Examples

```
# Example 1:
# an object of the class hac is constructed, whose parameters are close
copula = hac(type = 1, tree = list("X1", list("X2", "X3", 2.05), 2))

# the function aggregate returns a simple Archimedean copula

copula_ag = aggregate(copula, epsilon = 0.1)
tree2str(copula_ag) # [1] "(X1.X2.X3)_{2.02}"
```

```
# the structure does not change for a smaller epsilon

copula_ag = aggregate(copula, epsilon = 0.01)
tree2str(copula_ag) # [1] "((X2.X3)_{2.05}.X1)_{2}"

# Example 2:
# consider the binary tree

Object = hac.full(type = 1, y = c("X1", "X2", "X3", "X4", "X5"),
theta = c(1.01, 1.02, 2, 2.01))

tree2str(Object) # [1] "((((X5.X4)_{2.01}.X3)_{2}.X2)_{1.02}.X1)_{1.01}"

# applying aggregate.hac with epsilon = 0.02 leads to

Object_ag = aggregate(Object, 0.02)
tree2str(Object_ag) # [1] "((X3.X5.X4)_{2}.X1.X2)_{1.02}"
```

`copMult` *d-dim copula*

Description

This function returns the values for d-dimensional Archimedean copulae.

Usage

```
copMult(X, theta, type)
```

Arguments

`X`	a $n \times d$ matrix, where d refers to the dimension of the copula.
`theta`	the parameter of the copula.
`type`	all copula-types produced by Archimedean generators, see `phi` for an overview of implemented families.

Details

If warnings are returned, see `phi`.

Value

A vector containing the values of the copula.

See Also

`pHAC`

Examples

```
# the arguments are defined
X = matrix(runif(300), ncol = 3)

# the values are computed
cop = copMult(X, theta = 1.5, type = 1)
```

---

dHAC, pHAC, rHAC *pdf, cdf and random sampling*

Description

dHAC and pHAC compute the values of the copula's density and cumulative distribution function respectively. rHAC samples from HAC.

Usage

```
dHAC(X, hac, eval = TRUE, margins = NULL, na.rm = FALSE, ...)
pHAC(X, hac, margins = NULL, na.rm = FALSE, ...)
rHAC(n, hac)
```

Arguments

X	a data matrix. The number of columns and the corresponding names have to coincide with the specifications of the copula model hac.
hac	an object of the class hac.
n	number of observations.
margins	specifies the margins. The data matrix X is assumed to contain the values of the marginal distributions by default, i.e. margins = NULL. If raw data are used, the margins can be determined nonparametrically, "edf", or in parametric way, e.g. "norm". See estimate.copula for a detailed explanation.
na.rm	boolean. If na.rm = TRUE, missing values, NA, contained in X are removed.
eval	boolean. If eval = FALSE, a non-evaluated function is returned. Note, that attr "gradient" of the returned function corresponds to the values density.
...	arguments to be passed to na.omit.

Details

Sampling schemes of hierarchical and densities of simple Archimedean copula are based on functions of the copula package.

Value

rHAC retruns a $n \times d$ matrix, where d refers to the dimension of the HAC. dHAC and pHAC return vectors. The computation of the density might be time consuming for high-dimensions, since the density is defined as d-th derivative of the HAC with respect to its arguments $u_1, \ldots, u_d$.

References

Hofert, M. 2011, Efficiently Sampling Nested Archimedean Copulas, *Computational Statistics & Data Analysis* 55, 57-70.

Joe, H. 1997, Multivariate Models and Dependence Concepts, *Chapman & Hall*.

McNeil, A. J. 2008, Sampling Nested Archimedean Copulas, *Journal of Statistical Computation and Simulation* 78, 567-581.

Nelsen, R. B. 2006, An Introduction to Copulas, *Spinger*, 2nd Edition.

Okhrin, O. and Ristig, A. 2014, Hierarchical Archimedean Copulae: The HAC Package", *Journal of Statistical Software*, 58(4), 1-20, doi: 10.18637/jss.v058.i04.

Savu, C. and Trede, M. 2010, Hierarchies of Archimedean copulas, *Quantitative Finance* 10, 295-304.

See Also

`estimate.copula`, `to.logLik`

Examples

```
# AC example
# define the underlying model
model = hac(type = 4, tree = list("X1", "X2", 2))

# sample from model
sample = rHAC(100, model)

# returns the pdf/cdf at each vector of the sample
d.values = dHAC(sample, model)
p.values = pHAC(sample, model)

# HAC example
# the underlying model
y = c("X1", "X2", "X3")
theta = c(1.5, 3)
model = hac.full(type = 1, y, theta)

# define sample from copula model
sample = rHAC(100, model)

# returns the pdf/cdf at each point of the sample
d.values = dHAC(sample, model)
p.values = pHAC(sample, model)

# construct a hac-model
tree = list(list("X1", "X5", 3), list("X2", "X3", "X4", 4), 2)
model = hac(type = 1, tree = tree)

# sample from copula model
sample = rHAC(1000, model)
```

```
# check the accurancy of the estimation procedure
result1 = estimate.copula(sample)
result2 = estimate.copula(sample, epsilon = 0.2)
```

`emp.copula` *Empirical copula*

Description

`emp.copula` and `emp.copula.self` compute the empirical copula for a given sample. The difference between these functions is, that `emp.copula.self` does not require a matrix `u`, at which the function is evaluated.

Usage

```
emp.copula(u, x, proc = "M", sort = "none", margins = NULL,
na.rm = FALSE, ...)
emp.copula.self(x, proc = "M", sort = "none", margins = NULL,
na.rm = FALSE, ...)
```

Arguments

`u`	a matrix, at which the function is evaluated. According to the dimension of the data matrix `x`, it can be a scalar, a vector or a matrix. The entries of `u` should be within the interval $[0, 1]$.
`x`	denotes the matrix of marginal distributions, if `margins = NULL`. The number of columns should be equal the dimension d, whereas the number of rows should be equal to the number of observations n, with $n > d$.
`proc`	enables the user to choose between two different methods. It is recommended to use the default method, `"M"`, because it takes only a small fraction of the computational time of method `"A"`. However, method `"M"` is sensitive with respect to the size of the working memory and therefore, non-applicable for very large datasets.
`sort`	defines, whether the output is ordered. `sort = "asc"` refers to ascending values, which might be interesting for plotting and `sort = "desc"` refers to descending values.
`margins`	specifies the margins. The data matrix is assumed to contain the values of the marginal distributions by default, i.e. `margins = NULL`. If raw data are used, the margins can be determined nonparametrically, `"edf"`, or in parametric way, e.g. `"norm"`. See `estimate.copula` for a detailed explanation.
`na.rm`	boolean. If `na.rm = TRUE`, missing values, `NA`, contained in `x` and `u` are removed.
`...`	arguments to be passed to `na.omit`.

Details

The estimated copula follows the formula

$$\widehat{C}\left(u_1, \ldots, u_d\right) = n^{-1} \sum_{i=1}^{n} \prod_{j=1}^{d} \mathbf{I}\left\{\widehat{F}_j\left(X_{ij}\right) \leq u_j\right\},$$

where $\widehat{F}_j$ denotes the empirical marginal distribution function of variable X_j.

Value

A vector containing the values of the empirical copula.

References

Okhrin, O. and Ristig, A. 2014, Hierarchical Archimedean Copulae: The HAC Package", *Journal of Statistical Software*, 58(4), 1-20, doi: 10.18637/jss.v058.i04.

See Also

pHAC

Examples

```
v = seq(-4, 4, 0.05)
X = cbind(matrix(pt(v, 1), 161, 1), matrix(pnorm(v), 161, 1))

# both methods lead to the same result
z = emp.copula.self(X, proc = "M")
which(((emp.copula.self(X[1:100, ], proc = "M") - emp.copula.self(X[1:100, ],
proc = "A")) == 0) == "FALSE")
# integer(0)

# the contour plot
out = outer(z, z)
contour(x = X[,1], y = X[,2], out, main = "Contour Plot",
xlab = "Cauchy Margin", ylab = "Standard Normal Margin",
labcex = 1, lwd = 1.5, nlevels = 15)
```

`estimate.copula` *Estimation of Hierarchical Archimedean Copulae*

Description

The function estimates the parameters and determines the structure of Hierarchical Archimedean Copulae.

Usage

```
estimate.copula(X, type = 1, method = 1, hac = NULL, epsilon = 0,
agg.method = "mean", margins = NULL, na.rm = FALSE, max.min = TRUE, ...)
```

Arguments

X	a $n \times d$ matrix. If there are no colnames provided, the names X1,X2,... will be given.
type	defines the copula family, see phi for an overview of implemented families.
method	the estimation method. Select between quasi Maximum Likelihood 1, full Maximum Likelihood 2, recursive Maximum Likelihood 3 and penalized Maximum Likelihood 4.
hac	a hac object, which determines the structure and provides initial values. An object must be provided, if method = 2 referring to the full Maximum Likelihood procedure.
epsilon	scalar ≥ 0. The variables of consecutive nodes are aggregated, if the difference of the dependency parameters is smaller than epsilon. For a detailed explanation see also aggregate.hac.
agg.method	if $\epsilon > 0$, the new dependency parameter can be determined by "mean", "min" or "max" of the two parameters, see aggregate.hac.
margins	specifies the margins. The data matrix is assumed to contain the values of the marginal distributions by default, i.e. margins = NULL. If raw data are used, the margins can be determined nonparametrically, "edf", or in a parametric way, e.g. "norm". Following the latter approach, the parameters of the distributions are estimated by Maximum Likelihood. Building on these estimates the values of the univariate margins are computed. If the argument is defined as scalar, all margins are computed according to this specification. Otherwise, different margins can be defined, e.g. c("norm","t","edf") for a 3-dimensional sample. Almost all continuous functions of Distributions are available. Inappropriate usage of this argument might lead to misspecified margins, e.g. application of "exp" even though the sample contains negative values.
na.rm	boolean. If na.rm = TRUE, missing values, NA, contained in X are removed.
max.min	boolean. If max.min = TRUE and an element of X is ≥ 1 or ≤ 0, it is set to $1 - 10^{-8}$ and 10^{-8} respectively.
...	further arguments passed to or from other methods, e.g. na.omit.

Value

A hac object is returned.

References

Genest, C., Ghoudi, K., and Rivest, L. P. 1995, A Semiparametric Estimation Procedure of Dependence Parameters in Multivariate Families of Distributions, *Biometrika* 82, 543-552.

Gorecki, J., Hofert, M. and Holena, M. 2014, On the Consistency of an Estimator for Hierarchical Archimedean Copulas, In Talaysova, J., Stoklasa, J., Talaysek, T. (Eds.) *32nd International Conference on Mathematical Methods in Economics, Olomouc: Palacky University*, 239-244.

Joe, H. 2005, Asymptotic Efficiency of the Two-Stage Estimation Method for Copula-Based Models, *Journal of Multivariate Analysis* 94(2), 401-419.

Okhrin, O., Okhrin, Y. and Schmid, W. 2013, On the Structure and Estimation of Hierarchical Archimedean Copulas, *Journal of Econometrics* 173, 189-204.

Okhrin, O. and Ristig, A. 2014, Hierarchical Archimedean Copulae: The HAC Package", *Journal of Statistical Software*, 58(4), 1-20, doi: 10.18637/jss.v058.i04.

Okhrin, O., Ristig, A., Sheen J. and Trueck, S. 2015, Conditional Systemic Risk with Penalized Copula, *SFB 649 Discussion Paper 2015-038, Sonderforschungsbereich 649, Humboldt University, Germany*.

Examples

```
# define the copula model
tree = list(list("X1", "X5", 3), list("X2", "X3", "X4", 4), 2)
model = hac(type = 1, tree = tree)

# sample from copula model
x = rHAC(100, model)

# in the following case the true model is binary approximated
est.obj = estimate.copula(x, type = 1, method = 1, epsilon = 0)
plot(est.obj)

# consider also the aggregation of the variables
est.obj = estimate.copula(x, type = 1, method = 1, epsilon = 0.2)
plot(est.obj)

# full ML estimation to yield more precise parameter
est.obj.full = estimate.copula(x, type = 1, method = 2, hac = est.obj)

# recursive ML estimation leads to almost identical results
est.obj.r = estimate.copula(x, type = 1, method = 3)
```

`finData` *Financial data*

Description

This data set contains the standardized residuals of the filtered daily log-returns of four oil corporations: Chevron Corporation (`CVX`), Exxon Mobil Corporation (`XOM`), Royal Dutch Shell (`RDSA`) and Total (`FP`), covering $n = 283$ observations from 2011-02-02 to 2012-03-19. Intertemporal dependence is removed by usual ARMA-GARCH models, whose standardized residuals are used as `finData`.

Format

A matrix containing 283 observations of 4 stocks. The tickers of the stocks are presented as `colnames`.

Source

Yahoo! Finance

Examples

```
# load the data
data(finData)
```

`get.params` *Dependency parameters of a HAC*

Description

This function returns the copula parameter(s). They are ordered from top to down and left to right.

Usage

```
get.params(hac, sort.v = FALSE, ...)
```

Arguments

`hac`	an object of the class `hac`.
`sort.v`	boolean. If `sort.v = TRUE`, the output is sorted.
`...`	further arguments passed to `sort`.

See Also

`tree2str`

Examples

```
# construct a copula model
tree = list(list("X1", "X5", "X2", 4), list("X3", "X4", "X6", 3), 2)
model = hac(type = 1, tree)

# return the parameter
get.params(model) # [1] 2 4 3
get.params(model, sort.v = TRUE, decreasing = TRUE) # [1] 4 3 2
```

hac *Construction of* hac *objects*

Description

hac objects are required as input argument for several functions, e.g. `plot.hac` and `rHAC`. They can be constructed by hac and `hac.full`. The latter function produces only fully nested Archimedean copulae, whereas hac can construct arbitrary dependence structures for a given family. Moreover, the functions `hac2nacopula` and `nacopula2hac` ensure the compatability with the `copula` package.

Usage

```
hac(type, tree)
hac.full(type, y, theta)
## S3 method for class 'hac'
print(x, digits = 2, ...)
hac2nacopula(x)
nacopula2hac(outer_nacopula)
```

Arguments

`y`	a vector containing the variables, which are denoted by a `character`, e.g. `"X1"`.
`theta`	a vector containing the HAC parameters, which should be ordered from top to down. The length of `theta` must be equal to `length(y) -1`.
`tree`	a `list` object of the general structure `list(...,numeric(1))`. The last argument of the list, `numeric(1)`, denotes the dependency parameter. The arguments `...` are either of the same structure or of the class `character`. The character objects denote variables and embedded lists refer to nested subcopulae.
`type`	all copula-types are admissible, see `phi` for an overview of implemented families.
`x`	a hac object.
`outer_nacopula`	an `nacopula` object. The variables of the `outer_nacopula` object `1,2,...` are translated into the `character`s `"X1","X2",...`.
`digits`	specifies the digits, see `tree2str`.
`...`	arguments to be passed to `cat`.

Value

A hac object is returned.

`type`	the specified copula type.
`tree`	the structure of the HAC.

References

Hofert, M. and Maechler, M. 2011, Nested Archimedean Copulas Meet R: The nacopula Package, *Journal of Statistical Software*, 39(9), 1-20, doi: 10.18637/jss.v039.i09.

Hofert, M., Kojadinovic, I., Maechler, M. and Yan, J. 2015, copula: Multivariate Dependence with Copulas, *R package version 0.999-14*, https://CRAN.R-project.org/package=copula.

Kojadinovic, I., Yan, J. 2010, Modeling Multivariate Distributions with Continuous Margins Using the copula R Package, *Journal of Statistical Software*, 34(9), 1-20. doi: 10.18637/jss.v034.i09.

Okhrin, O. and Ristig, A. 2014, Hierarchical Archimedean Copulae: The HAC Package", *Journal of Statistical Software*, 58(4), 1-20, doi: 10.18637/jss.v058.i04.

Yan, J. 2007, Enjoy the Joy of Copulas: With a Package copula, *Journal of Statistical Software*, 21(4), 1-21, doi: 10.18637/jss.v021.i04.

Examples

```
# it might be helpful to plot the hac objects
# Example 1: 4-dim AC
tree = list("X1", "X2", "X3", "X4", 2)
AC = hac(type = 1, tree = tree)

# Example 2: 4-dim HAC
y = c("X1", "X4", "X3", "X2")
theta = c(2, 3, 4)

HAC1 = hac.full(type = 1, y = y, theta = theta)
HAC2 = hac(type = 1, tree = list(list(list("X2", "X3", 4),
"X4", 3), "X1", 2))
tree2str(HAC1) == tree2str(HAC2) # [1] TRUE

# Example 3: 9-dim HAC

HAC = hac(type = 1, tree = list("X6", "X5", list("X2", "X4", "X3", 4.4),
list("X1", "X7", 3.3), list("X8", "X9", 4), 2.3))
plot(HAC)
```

---

par.pairs *Parameter of the HAC*

Description

This function returns a matrix of HAC parameters. They are pairwise ordered, so that the parameters correspond to the lowest node, at which the variables are joined.

Usage

```
par.pairs(hac, FUN = NULL, ...)
```

Arguments

hac	an object of the class hac.
FUN	the parameters of the HAC are returned by default. If FUN = "TAU", theta2tau is applied to the parameters. FUN can also be a self-defined function.
...	further arguments passed to FUN.

See Also

get.params

Examples

```
# construct a copula model
tree = list(list("X1", "X5", "X2", 4), list("X3", "X4", "X6", 3), 2)
model = hac(type = 1, tree)

# returns the pairwise parameter
par.pairs(model)

# Kendall's TAU
par.pairs(model, FUN = "TAU")

# sqrt of the parameter
par.pairs(model, function(r)sqrt(r))
```

---

phi, phi.inv *Generator function*

Description

The Archimedean generator function and its inverse.

Usage

```
phi(x, theta, type)
phi.inv(x, theta, type)
```

Arguments

x	a scalar, vector or matrix at which the function is evaluated. The support of the functions has to be taken into account, i.e. $x \in [0, \infty]$ for the generator function and $x \in [0, 1]$ for its inverse.
theta	the feasible copula parameter, i.e. $\theta \in [1, \infty)$ for the Gumbel and Joe family, $\theta \in (0, \infty)$ for the Clayton and Frank family and $\theta \in [0, 1)$ for the Ali-Mikhail-Haq family.
type	select between the following integer numbers for specifying the type of the hierarchical Archimedean copula (HAC) or Archimedean copula (AC):

- 1 = HAC Gumbel
- 2 = AC Gumbel
- 3 = HAC Clayton
- 4 = AC Clayton
- 5 = HAC Frank
- 6 = AC Frank
- 7 = HAC Joe
- 8 = AC Joe
- 9 = HAC Ali-Mikhail-Haq
- 10 = AC Ali-Mikhail-Haq

Examples

```
x = runif(100, min = 0, max = 100)
phi(x, theta = 1.2, type = 1)

# do not run
# phi(x, theta = 0.8, type = 1)
# In phi(x, theta = 0.8, type = 1) : theta >= 1 is required.
```

---

plot.hac *Plot of a HAC*

Description

The function plots the structure of Hierarchical Archimedean Copulae.

Usage

```
## S3 method for class 'hac'
plot(x, xlim = NULL, ylim = NULL, xlab = "", ylab = "",
col = "black", fg = "black", bg = "white", col.t = "black", lwd = 2,
index = FALSE, numbering = FALSE, theta = TRUE, h = 0.4, l = 1.2,
circles = 0.25, digits = 2, ...)
```

Arguments

x	a hac object. It can be constructed by hac or be the result of estimate.copula.
xlim, ylim	numeric vectors of length 2, giving the limits of the x and y axes. The default values adjust the size of the coordinate plane automatically with respect to the dimension of the HAC.
xlab, ylab	titles for the x and y axes.
col	defines the color of the lines, which connect the circles and rectangles.
fg	defines the color of the lines of the rectangles and circles equivalent to the color settings in R.

bg	defines the background color of the rectangles and circles equivalent to the color settings in R.
col.t	defines the text color equivalent to the color settings in R.
lwd	the width of the lines.
index	boolean. If index = TRUE, strings, which illustrate the subcopulae of the nodes, are used as subsrcipts of the dependency parameters.
numbering	boolean. If index = TRUE and numbering = TRUE, the dependency parameters are numbered. If x is returned by estimate.copula, the numbers correpsond to the estimation stages.
theta	boolean. Determines, whether the dependency parameter θ or Kendall's rank correlation coefficient τ is printed.
h	the height of the rectangles.
l	the width of the rectangles.
circles	a positive number giving the radius of the circles.
digits	an integer specifying the number of digits of the dependence parameter.
...	arguments to be passed to methods, e.g. graphical parameters (see par).

References

Okhrin, O. and Ristig, A. 2014, Hierarchical Archimedean Copulae: The HAC Package", *Journal of Statistical Software*, 58(4), 1-20, doi: 10.18637/jss.v058.i04.

See Also

estimate.copula

Examples

```
# a hac object is created

tree = list(list("X1", "X5", 3), list("X2", "X3", "X4", 4), 2)
model = hac(type = 1, tree = tree)
plot(model)

# the same procedure works for an estimated object

sample = rHAC(2000, model)
est.obj = estimate.copula(sample, epsilon = 0.2)
plot(est.obj)
```

theta2tau, tau2theta *Kendall's rank correlation coefficient*

Description

Kendall's rank correlation coefficient and its inverse.

Usage

```
theta2tau(theta, type)
tau2theta(tau, type)
```

Arguments

theta	the dependency parameter. It can be either a scalar, a vector or a matrix and has to lie within a certain interval, i.e. $\theta \in [1, \infty)$ for the Gumbel and Joe family, $\theta \in (0, \infty)$ for the Clayton and Frank family and $\theta \in [0, 1)$ for the Ali-Mikhail-Haq family.
tau	Kendall's rank correlation coefficient. It can be either a scalar, a vector or a matrix and it is to ensure, that $\tau \in [0, 1)$.
type	all types are available, see phi for an overview of implemented families.

Examples

```
# computation of the dependency parameter
x = runif(10)
theta = tau2theta(x, type = 1)

# computation of kendall's tau
y = runif(10, 1, 100)
tau = theta2tau(y, type = 1)
```

to.logLik *log-likelihood*

Description

to.logLik returns either the log-likehood function depending on a vector theta for a given sample X or the value of the log-likelihood, if eval = TRUE.

Usage

```
to.logLik(X, hac, eval = FALSE, margins = NULL, sum.log = TRUE,
na.rm = FALSE, ...)
```

Arguments

X	a data matrix. The number of columns and the corresponding names have to coincide with the specifications of the copula model hac. The sample X has to contain at least 2 rows (observations), as the values of the underlying density cannot be computed otherwise.
hac	an object of the class hac.
eval	boolean. If eval = FALSE, the non-evaluated log-likelihood function depending on a parameter vector theta is returned and one default argument, the density, is returned. The values of theta are increasingly ordered.
margins	specifies the margins. The data matrix X is assumed to contain the values of the marginal distributions by default, i.e. margins = NULL. If raw data are used, the margins can be determined nonparametrically, "edf", or in parametric way, e.g. "norm". See estimate.copula for a detailed explanation.
sum.log	boolean. If sum.log = FALSE, the values of the individual log-likelihood contributions are returned.
na.rm	boolean. If na.rm = TRUE, missing values, NA, contained in X are removed.
...	arguments to be passed to na.omit.

See Also

dHAC

Examples

```
# construct a hac-model
tree = list(list("X1", "X5", 3), list("X2", "X3", "X4", 4), 2)
model = hac(type = 1, tree = tree)

# sample from copula model
sample = rHAC(1000, model)

# check the accurancy of the estimation procedure
ll = to.logLik(sample, model)
ll.value = to.logLik(sample, model, eval = TRUE)

ll(c(2, 3, 4)) == ll.value # [1] TRUE
```

---

tree2str *String structure of HAC*

Description

The function prints the structure of HAC as string, so that the important characteristics of the copula can be identified.

Usage

```
tree2str(hac, theta = TRUE, digits = 2)
```

Arguments

hac	an object of the class hac.
theta	boolean. Determines, whether the values of the dependency parameter(s) are printed (TRUE) or not (FALSE).
digits	a non-negative integer value specifying the number of digits of the dependency parameter(s).

Value

a string of the class character.

See Also

plot.hac

Examples

```
# construct a hac object
tree = list(list("X1", "X5", "X2", 3), list("X3", "X4", "X6", 4), 2)
model = hac(type = 1, tree = tree)

# the parameters are returned within the curly brackets
# variables nested at the same node are separated by a dot

tree2str(model) # [1] "((X1.X5.X2)_{3}.(X3.X4.X6)_{4})_{2}"

# (X1.X5.X2)_{3} and (X3.X4.X6)_{4} are the two variables nested at the
# initial node with dependency parameter 2

tree2str(model, theta = FALSE) # [1] "((X1.X5.X2).(X3.X4.X6))"

# if theta = FALSE, only the structure of the variables is returned

# alternatively consider the following nested AC

tree = list("X1", list("X5", "X2", 3), list("X3", "X4", "X6", 4), 1.01)
model = hac(type = 1, tree = tree)

tree2str(model) # [1] "(X1.(X5.X2)_{3}.(X3.X4.X6)_{4})_{1.01}"

# _{1.01} represents the initial node
# the first three variables are given by the subtrees (X3.X4.X6)_{4},
# (X5.X2)_{3} and X1
```

Index