Package 'vines'

October 12, 2022

Type Package

Title Multivariate Dependence Modeling with Vines

Version 1.1.5

Description Implementation of the vine graphical model for building high-dimensional probability distributions as a factorization of bivariate copulas and marginal density functions. This package provides S4 classes for vines (C-vines and D-vines) and methods for inference, goodness-of-fit tests, density/distribution function evaluation, and simulation.

License GPL (>= 2)

URL https://github.com/yasserglez/vines

Depends methods, copula

Imports ADGofTest, cubature, TSP

Suggests testthat

Collate h.R hinverse.R Vine.R show.R dimnames.R vineParameters.R vineIter.R vineFit.R vineFitML.R vinePIT.R vineGoF.R vineOrder.R dvine.R pvine.R rvine.R

NeedsCompilation yes

Author Yasser Gonzalez-Fernandez [aut, cre],
Marta Soto [aut],
Joris Meys [ctb]

Maintainer Yasser Gonzalez-Fernandez <ygonzalezfernandez@gmail.com>

Repository CRAN

Date/Publication 2016-07-28 11:49:43

R topics documented:

h-methods 2
hinverse-methods 4
RVine-classes 5
Vine 5

Vine-class 6
Vine-distribution 7
vineFit 8
vineFit-class 10
vineFitML-class 11
vineGoF 11
vineGoF-class 13
vineLogLik 13
vineOrder 14
vineParameters 15
vinePIT-methods 16

Index **18**

`h-methods` *Methods for the h-functions*

Description

The h function represents the conditional distribution function of a bivariate copula and it should be defined for every copula used in a pair-copula construction. It is defined as the partial derivative of the distribution function of the copula w.r.t. the second argument $h(x, v) = F(x|v) = \partial C(x, v)/\partial v$.

Usage

```
h(copula, x, v, eps)
```

Arguments

`copula`	A bivariate `copula` object.
`x`	Numeric vector with values in $[0, 1]$.
`v`	Numeric vector with values in $[0, 1]$.
`eps`	To avoid numerical problems for extreme values, the values of `x`, `v` and return values close to `0` and `1` are substituted by `eps` and `1 - eps`, respectively. The default `eps` value for most of the copulas is `.Machine$double.eps^0.5`.

Methods

`signature(copula = "copula")` Default definition of the h function for a bivariate copula. This method is used if no particular definition is given for a copula. The partial derivative is calculated numerically using the `numericDeriv` function.

`signature(copula = "indepCopula")` The h function of the independence copula.

$$h(x, v) = x$$

`signature(copula = "normalCopula")` The h function of the normal copula.

$$h(x, v; \rho) = \Phi\left(\frac{\Phi^{-1}(x) - \rho\,\Phi^{-1}(v)}{\sqrt{1-\rho^2}}\right)$$

`signature(copula = "tCopula")` The h function of the t copula.

$$h(x, v; \rho, \nu) = t_{\nu+1}\left(\frac{t_\nu^{-1}(x) - \rho\, t_\nu^{-1}(v)}{\sqrt{\frac{(\nu+(t_\nu^{-1}(v))^2)(1-\rho^2)}{\nu+1}}}\right)$$

`signature(copula = "claytonCopula")` The h function of the Clayton copula.

$$h(x, v; \theta) = v^{-\theta-1}(x^{-\theta} + v^{-\theta} - 1)^{-1-1/\theta}$$

`signature(copula = "gumbelCopula")` The h function of the Gumbel copula.

$$h(x, v; \theta) = C(x, v; \theta)\,\frac{1}{v}\,(-\log v)^{\theta-1}\left((-\log x)^\theta + (-\log v)^\theta\right)^{1/\theta-1}$$

`signature(copula = "fgmCopula")` The h function of the Farlie-Gumbel-Morgenstern copula.

$$h(x, v; \theta) = (1 + \theta\,(-1+2v)\,(-1+x))\,x$$

`signature(copula = "frankCopula")` The h function of the Frank copula.

$$h(x, v; \theta) = \frac{e^{-\theta v}}{\frac{1-e^{-\theta}}{1-e^{-\theta x}} + e^{-\theta v} - 1}$$

`signature(copula = "galambosCopula")` The h function of the Galambos copula.

$$h(x, v; \theta) = \frac{C(x, v; \theta)}{v}\left(1 - \left[1 + \left(\frac{-\log v}{-\log x}\right)^\theta\right]^{-1-1/\theta}\right)$$

References

Aas, K. and Czado, C. and Frigessi, A. and Bakken, H. (2009) Pair-copula constructions of multiple dependence. *Insurance: Mathematics and Economics* **44**, 182–198.

Schirmacher, D. and Schirmacher, E. (2008) Multivariate dependence modeling using pair-copulas. *Enterprise Risk Management Symposium, Chicago*.

`hinverse-methods` *Methods for the Inverse of the h-functions*

Description

The h^{-1} function represents the inverse of the h function with respect to its first argument. It should be defined for every copula used in a pair-copula construction (or it will be evaluated numerically).

Usage

```
hinverse(copula, u, v, eps)
```

Arguments

`copula`	A bivariate `copula` object.
`u`	Numeric vector with values in $[0, 1]$.
`v`	Numeric vector with values in $[0, 1]$.
`eps`	To avoid numerical problems for extreme values, the values of `u`, `v` and return values close to `0` and `1` are substituted by `eps` and `1 - eps`, respectively. The default `eps` value for most of the copulas is `.Machine$double.eps^0.5`.

Methods

`signature(copula = "copula")` Default definition of the h^{-1} function for a bivariate copula. This method is used if no particular definition is given for a copula. The inverse is calculated numerically using the `uniroot` function.

`signature(copula = "indepCopula")` The h^{-1} function of the Independence copula.

$$h^{-1}(u, v) = u$$

`signature(copula = "normalCopula")` The h^{-1} function of the normal copula.

$$h^{-1}(u, v; \rho) = \Phi\left(\Phi^{-1}(u)\sqrt{1-\rho^2} + \rho\,\Phi^{-1}(v)\right)$$

`signature(copula = "tCopula")` The h^{-1} function of the t copula.

$$h^{-1}(u, v; \rho, \nu) = t_\nu\left(t_{\nu+1}^{-1}(u)\sqrt{\frac{(\nu + (t_\nu^{-1}(v))^2)(1-\rho^2)}{\nu+1}} + \rho\,t_\nu^{-1}(v)\right)$$

`signature(copula = "claytonCopula")` The h^{-1} function of the Clayton copula.

$$h^{-1}(u, v; \theta) = \left(\left(u\,v^{\theta+1}\right)^{-\frac{\theta}{\theta+1}} + 1 - v^{-\theta}\right)^{-1/\theta}$$

`signature(copula = "frankCopula")` The h^{-1} function of the Frank copula.

$$h^{-1}(u, v; \theta) = -\log\left(1 - \frac{1-e^{-\theta}}{(u^{-1}-1)e^{-\theta v}+1}\right)/\theta$$

References

Aas, K. and Czado, C. and Frigessi, A. and Bakken, H. (2009) Pair-copula constructions of multiple dependence. *Insurance: Mathematics and Economics* **44**, 182–198.

Schirmacher, D. and Schirmacher, E. (2008) Multivariate dependence modeling using pair-copulas. *Enterprise Risk Management Symposium, Chicago*.

`RVine-classes` *Classes for Regular Vines*

Description

Extend the `Vine` class to represent regular vines.

Slots

See `Vine` for information about the inherited slots.

Methods

show `signature(object = "CVine")`: Print a textual representation of the C-vine.

show `signature(object = "DVine")`: Print a textual representation of the D-vine.

See Also

`Vine`.

`Vine` *Create Vine Objects*

Description

Functions to create `Vine` objects.

Usage

```
Vine(type, dimension = 2, trees = dimension - 1,
     copulas = matrix(list(indepCopula()),
                      dimension - 1, dimension - 1))
CVine(dimension = 2, trees = dimension - 1,
      copulas = matrix(list(indepCopula()),
                       dimension - 1, dimension - 1))
DVine(dimension = 2, trees = dimension - 1,
      copulas = matrix(list(indepCopula()),
                       dimension - 1, dimension - 1))
```

Arguments

type	Type of vine. Supported values: "CVine" and "DVine".
dimension	See the documentation of the Vine slot.
trees	See the documentation of the Vine slot.
copulas	See the documentation of the Vine slot.

See Also

Vine, CVine, DVine.

Examples

```
dimension <- 3
copulas <- matrix(list(normalCopula(0.5),
                       claytonCopula(2.75),
                       tCopula(0.75, df = 2),
                       NULL),
                  ncol = dimension - 1,
                  nrow = dimension - 1,
                  byrow = TRUE)

Vine("DVine", dimension = dimension, trees = dimension - 1,
     copulas = copulas)
DVine(dimension = dimension, trees = dimension - 1,
      copulas = copulas)
```

Vine-class *Base Vine Class*

Description

Base class of all classes that implement vine models in the package. It is a virtual class, no objects may be created from it.

Slots

type: Object of class "character". Descriptive name of the type of vine: Canonical vine or D-vine.

dimension: Object of class "numeric". Dimension of the vine.

dimensionNames: Object of class "character". Names of the variables of the vine. It is either an empty character vector or a vector with one element for each variable of the vine.

copulas: Object of class "matrix". It contains the copula objects for each copula in the pair-copula decomposition. The indexes of the matrix follow the notation used for the subscripts of Θ in (Aas et al., 2009).

trees: Object of class "numeric". Number of dependence trees of the vine. It should be an integer between 0 and dimension - 1, including both endpoints. Functions acting on vines assume that copulas in arcs of all trees greater than trees are independence copulas.

Methods

show `signature(object = "Vine")`: Print a textual representation of the vine.

dimnames `signature(x = "Vine")`: Retrieve the names of the variables of the vine.

dimnames<- `signature(x = "Vine")`: Set the names of the variables of the vine.

References

Aas, K. and Czado, C. and Frigessi, A. and Bakken, H. (2009) Pair-copula constructions of multiple dependence. *Insurance: Mathematics and Economics* **44**, 182–198.

See Also

`RVine`, `CVine`, `DVine`.

`Vine-distribution` *Vine Distribution Functions*

Description

Density evaluation, distribution function evaluation, and random number generation.

Usage

```
dvine(vine, u)
pvine(vine, u)
rvine(vine, n)
```

Arguments

`vine`	A `Vine` object.
`u`	Vector of the same dimension of the vine or a matrix with one column for each variable of the vine.
`n`	Number of observations.

Details

The implementation of the `dvine` function for `CVine` and `DVine` objects is based on the Algorithms 3 and 4 of (Aas et al. 2009), respectively.

The `pvine` function is evaluated through the numerical integration of the density function (using the `cubature` package). This is a computationally demanding procedure, even for small dimensions.

The implementation of the `rvine` function for `CVine` and `DVine` objects is based on the Algorithms 1 and 2 of (Aas et al. 2009), respectively.

Value

dnorm returns a vector with the evaluation of the density. pnorm returns a vector with the evaluation of the distribution function. rvine returns a matrix with one column for each variable of the vine and one row for each observation.

References

Aas, K. and Czado, C. and Frigessi, A. and Bakken, H. (2009) Pair-copula constructions of multiple dependence. *Insurance: Mathematics and Economics* **44**, 182–198.

Bedford, T. and Cooke, R. M. (2001) Monte Carlo simulation of vine dependent random variables for applications in uncertainty analysis. In *2001 Proceedings of ESREL2001, Turin, Italy*.

Bedford, T. and Cooke, R. M. (2001) Probability density decomposition for conditionally dependent random variables modeled by vines. *Annals of Mathematics and Artificial Intelligence* **32**, 245–268.

Kurowicka, D. and Cooke, R. M. (2005) Sampling algorithms for generating joint uniform distributions using the vine-copula method. In *3rd IASC World Conference on Computational Statistics & Data Analysis, Limassol, Cyprus*.

Examples

```
dimension <- 3
copulas <- matrix(list(normalCopula(0.5),
                       claytonCopula(2.75),
                       tCopula(0.75, df = 2),
                       NULL),
                  ncol = dimension - 1,
                  nrow = dimension - 1)
vine <- DVine(dimension = dimension, trees = dimension - 1,
              copulas = copulas)
dimnames(vine) <- c("A", "B", "C")

data <- rvine(vine, 1)
dvine(vine, data)
pvine(vine, data)
```

---

vineFit *Vine Inference*

Description

Estimate a vine model from multivariate data in the unit hypercube. Data can be pseudo-observations constructed from empirical or parametric marginal cumulative distribution functions.

Usage

```
vineFit(type, data, method = "ml", ...)
```

Arguments

type	Type of vine. Supported values: "CVine" and "DVine".
data	Data matrix of pseudo-observations.
method	Inference method. Supported values: "ml" (Maximum Likelihood).
...	Additional arguments for the inference method.

Details

The "ml" (Maximum Likelihood) method starts with the sequential estimation procedure described in (Aas et al., 2009) and then executes a numerical maximization of the full log-likelihood of the vine. The sequential procedure is used to determine the family and the initial values of the parameters of each bivariate copula in the decomposition. Additional arguments for this method are:

selectCopula Function provided by the user to select the copulas in the pair-copula construction. This function should return a copula object and expect the following arguments.

vine Vine object being constructed.

j, i Indexes of the copula under selection in the matrix of the copulas slot of the vine.

x, y Bivariate sample.

The default value is function(vine, j, i, x, y) indepCopula() that assigns the independence copula to all the arcs of the vine.

trees Maximum number of dependence trees of the vine. Independence copulas will be used in all the arcs of the following trees. The final number of dependence trees could be smaller because of the use of a truncation method. The default value is ncol(data) - 1.

truncMethod Method used to automatically truncate the vine if enough dependence is captured in a given number of trees. Supported methods are "AIC" and "BIC". See (Brechmann, 2010; Brechmann et al., 2010) for information about these methods. The default value is "" that means no truncation.

optimMethod optim method used in the optimization of the log-likelihood function. If "" is specified the optimization is disabled and the vine calculated using the sequential estimation procedure is returned. The default value is "Nelder-Mead".

optimControl List of control parameters for optim. The default value is list().

Value

A vineFit object or a subclass with specific information about inference method used. The vine slot of this object contains the fitted Vine object.

References

Aas, K. and Czado, C. and Frigessi, A. and Bakken, H. (2009) Pair-copula constructions of multiple dependence. *Insurance: Mathematics and Economics* **44**, 182–198.

Brechmann, E. C. (2010) Truncated and simplified regular vines and their applications. Diploma thesis. *Technische Universitaet Muenchen*.

Brechmann, E. C. and Czado, C. and Aas, K. (2010) Truncated regular vines in high dimensions with application to financial data. *Norwegian Computing Center, NR*. Note SAMBA/60/10.

See Also

CVine, DVine, vineFit, vineFitML.

Examples

```
data <- matrix(runif(5 * 100), ncol = 5, nrow = 100)
colnames(data) <- c("A", "B", "C", "D", "E")

selectCopula <- function (vine, j, i, x, y) {
    data <- cbind(x, y)
    fit <- fitCopula(normalCopula(), data, method = "itau")
    fit@copula
}
fit <- vineFit("DVine", data, method = "ml",
               selectCopula = selectCopula,
               optimMethod = "")

show(fit)
show(fit@vine)
```

---

vineFit-class *Class for the Results of Vine Inference*

Description

Base class of all classes providing information about vine inference. Objects of this class (or subclasses) are created by calling the vineFit function.

Slots

vine: Object of class Vine. Fitted vine.

observations: Object of class numeric. Number of observations in the sample.

method: Object of class character. Inference method.

Methods

show signature(object = "vineFit"): Print the result of the vine inference.

See Also

vineFit, vineFitML.

vineFitML-class *Class for the Results of Vine Inference by Maximum Likelihood*

Description

Extends the vineFit class to include information about the Maximum Likelihood inference.

Slots

See vineFit for information about inherited slots.

optimMethod: Object of class character. optim method.

optimConv: Object of class numeric. optim convergence code (0 indicates successful completion).

startParams: Object of class numeric. Vector with the parameters of the copulas in the pair-copula construction calculated using the sequential estimation procedure.

finalParams: Object of class numeric. Vector with the parameters of the copulas in the pair-copula construction after the maximization of the log-likelihood.

Methods

show signature(object = "vineFitML"): Print information about the fitted vine.

See Also

vineFit, vineFit, vineParameters.

vineGoF *Vine Goodness-of-fit Tests*

Description

Goodness-of-fit tests to verify whether the dependence structure of a sample is appropriately modeled by vine model.

Usage

```
vineGoF(vine, data, method = "PIT", ...)
```

Arguments

vine	A Vine object.
data	Data matrix of pseudo-observations.
method	Goodness-of-fit method. Supported values: "PIT" (Probability Integral Transform).
...	Additional arguments for the goodness-of-fit method.

Details

The "PIT" (Probability Integral Transform) method uses the vinePIT function to transform the data into variables which are independent and Uniform(0,1) and then use a hypothesis test to verify whether the resulting variables are independent and Uniform(0,1). The additional parameter statistic specifies the test to be applied for this purpose.

statistic Statistic used to verify if the transformed variables are independent and Uniform(0,1). The default value is "Breymann" and supported methods are:

"Breymann" Test proposed in the Section 7.1 of (Aas et al., 2009). See (Breymann et al., 2003) for more information.

Value

A vineGoF or a subclass with specific information about the goodness-of-fit method used. The statistic slot of this object contains the value of the statistic and pvalue the p-value.

References

Aas, K. and Czado, C. and Frigessi, A. and Bakken, H. (2009) Pair-copula constructions of multiple dependence. *Insurance: Mathematics and Economics* **44**, 182–198.

Breymann, W. and Dias, A. and Embrechts, P. (2003) Dependence structures for multivariate high-frequency data in finance. *Quantitative Finance* **1**, 1–14.

See Also

vineGoF, vinePIT.

Examples

```
copula <- normalCopula(c(-0.25, -0.21, 0.34, 0.51, -0.07, -0.18),
                       dispstr = "un", dim = 4)
data <- rCopula(100, copula)

selectCopula <- function (vine, j, i, x, y) {
    data <- cbind(x, y)
    fit <- fitCopula(normalCopula(), data, method = "itau")
    fit@copula
}
normalCVine <- vineFit("CVine", data, method = "ml",
                       selectCopula = selectCopula,
                       optimMethod = "")@vine
normalDVine <- vineFit("DVine", data, method = "ml",
                       selectCopula = selectCopula,
                       optimMethod = "")@vine
show(normalCVine)
show(normalDVine)

normalCVineGof <- vineGoF(normalCVine, data, method = "PIT",
                          statistic = "Breymann")
normalDVineGof <- vineGoF(normalDVine, data, method = "PIT",
```

```
                          statistic = "Breymann")
show(normalCVineGof)
show(normalDVineGof)
```

---

vineGoF-class *Class for the Results of Vine Goodness-of-fit Tests*

Description

Base class of all classes containing information about vine goodness-of-fit tests. Objects of this class (or subclasses) are created by calling the vineGoF function.

Slots

`method`**:** Object of class `character`. Goodness-of-fit method.

`statistic`**:** Object of class `numeric`. Value of the test statistic.

`pvalue`**:** Object of class `numeric`. P-value.

Methods

show `signature(object = "vineGoF")`: Print the result of the goodness-of-fit test.

See Also

vineGoF.

---

vineLogLik *Vine Log-likelihood Evaluation*

Description

Evaluate the log-likelihood of a Vine.

Usage

```
vineLogLik(vine, data)
```

Arguments

vine	A Vine object.
data	Data matrix of pseudo-observations.

Examples

```
dimension <- 3
copulas <- matrix(list(normalCopula(0.5),
                       claytonCopula(2.75),
                       NULL, NULL),
                  ncol = dimension - 1,
                  nrow = dimension - 1,
                  byrow = TRUE)
vine <- DVine(dimension = dimension, trees = 1,
              copulas = copulas)

data <- matrix(runif(dimension * 100),
               ncol = dimension, nrow = 100)

vineLogLik(vine, data)
```

`vineOrder` *Select an Order of the Variables*

Description

Select an order of the variables. The order of the variables determines the bivariate dependencies that will be explicit modeled in the first tree of the vine.

Usage

```
vineOrder(type, data, method = "greedy", ...)
```

Arguments

`type`	Type of vine. Supported values: `"CVine"` and `"DVine"`.
`data`	Data matrix of pseudo-observations.
`method`	Ordering method. Supported values: `"random"`, `"greedy"`.
`...`	Additional arguments for the order method.

Details

In D-vines, the order of the variables determines the structure of all the trees of the vine. This is not the case for C-vines where the root node of each tree can be selected.

The `"random"` method returns a random permutation of the variables.

The `"greedy"` method returns an order of the variables that intends to capture as much dependence as possible in the first tree of the vine. The method finds the order of the variables that defines a tree that maximizes a given dependence measure used as edge weights. For C-vines, it is determined iteratively checking each variable as root node. For D-vines, it is equivalent to solve the traveling salesman problem (TSP), see (Brechmann, 2010) for details. The TSP is solved using the cheapest insertion algorithm implemented by the `solve_TSP` function of the **TSP** package. The following are additional parameters for this method.

`according` Dependence measure. The default value is `"kendall"` and supported values are:

`"kendall"` Absolute value of Kendall's τ.

`"spearman"` Absolute value of Spearman's ρ.

`"pearson"` Absolute value of Pearson's product moment correlation coefficient.

`"df"` Smaller degrees of freedom of a bivariate t copula.

Value

A vector with the ordered indexes of the variables. This vector should be used to reorder the variables of the `data` matrix.

References

Brechmann, E. C. (2010) Truncated and simplified regular vines and their applications. Diploma thesis. *Technische Universitaet Muenchen*.

Examples

```
data <- matrix(runif(5 * 100), ncol = 5, nrow = 500)

vineOrder("CVine", data, method = "random")
vineOrder("DVine", data, method = "greedy",
          according = "spearman")
```

`vineParameters` *Parameters of a Vine*

Description

Retrieve or set the parameters of a `Vine`.

Usage

```
vineParameters(vine)
vineParameters(vine) <- value
```

Arguments

`vine`	A `Vine` object.
`value`	Vector with the parameters of the copulas in the pair-copula construction. This vector is the result of the concatenation of the `parameters` slots of the `copula` objects in the `copulas` slot of the `Vine` object (by rows).

Examples

```
dimension <- 5
copulas <- matrix(c(list(tCopula(-0.25, df = 2),
                         tCopula(-0.5, df = 4),
                         tCopula(0.25, df = 6),
                         tCopula(0.5, df = 8)),
                    rep(list(NULL), 12)),
                  ncol = dimension - 1,
                  nrow = dimension - 1,
                  byrow = TRUE)
vine <- DVine(dimension = dimension, trees = 1,
              copulas = copulas)
dimnames(vine) <- c("A", "B", "C", "D", "E")

vineParameters(vine)
show(vine)

vineParameters(vine) <- c(-0.25, 3, -0.5, 5, 0.25, 7, 0.5, 9)

vineParameters(vine)
show(vine)
```

`vinePIT-methods` *Vine Probability Integral Transform Methods*

Description

Probability integral transform (PIT) of (Rosenblatt, 1952) for vine models. The PIT converts a set of dependent variables into a new set of variables which are independent and uniformly distributed in $(0, 1)$ under the hypothesis that the data follows a given multivariate distribution.

Usage

```
vinePIT(vine, u)
```

Arguments

`vine`	A `Vine` object.
`u`	Vector with one component for each variable of the vine or a matrix with one column for each variable of the vine.

Value

A matrix with one column for each variable of the vine and one row for each observation.

Methods

`signature(vine = "CVine")` PIT algorithm for `CVine` objects based on the Algorithm 5 of (Aas et al., 2009).

`signature(vine = "DVine")` PIT algorithm for `DVine` objects based on the Algorithm 6 of (Aas et al., 2009).

References

Aas, K. and Czado, C. and Frigessi, A. and Bakken, H. (2009) Pair-copula constructions of multiple dependence. *Insurance: Mathematics and Economics* **44**, 182–198.

Rosenblatt, M. (1952) Remarks on multivariate transformation. *Annals of Mathematical Statistics* **23**, 1052–1057.

See Also

`vinePIT`.

Examples

```
dimension <- 3
copulas <- matrix(list(normalCopula(0.5),
                       claytonCopula(2.75),
                       NULL, NULL),
                  ncol = dimension - 1,
                  nrow = dimension - 1,
                  byrow = TRUE)
vine <- CVine(dimension = dimension, trees = 1,
              copulas = copulas)

data <- matrix(runif(dimension * 100),
               ncol = dimension, nrow = 100)

vinePIT(vine, data)
```

Index

∗ **classes**
RVine-classes, 5
Vine-class, 6
vineFit-class, 10
vineFitML-class, 11
vineGoF-class, 13
∗ **distribution**
Vine-distribution, 7
∗ **htest**
vineGoF-class, 13
∗ **methods**
h-methods, 2
hinverse-methods, 4
Vine-distribution, 7
vinePIT-methods, 16
∗ **models**
RVine-classes, 5
Vine-class, 6
vineFit-class, 10
vineFitML-class, 11
vineGoF-class, 13
∗ **multivariate**
RVine-classes, 5
Vine-class, 6
Vine-distribution, 7
vineFit-class, 10
vineFitML-class, 11
vineGoF-class, 13

copula, *2*, *4*, *6*, *9*, *15*
cubature, *7*
CVine, *6, 7*, *10*, *17*
CVine *(*Vine*)*, 5
CVine-class *(*RVine-classes*)*, 5

dimnames,Vine-method *(*Vine-class*)*, 6
dimnames<-,Vine,ANY-method *(*Vine-class*)*, 6
dimnames<-,Vine-method *(*Vine-class*)*, 6
DVine, *6, 7*, *10*, *17*
DVine *(*Vine*)*, 5
dvine *(*Vine-distribution*)*, 7
dvine,CVine-method *(*Vine-distribution*)*, 7
dvine,DVine-method *(*Vine-distribution*)*, 7
DVine-class *(*RVine-classes*)*, 5
dvine-methods *(*Vine-distribution*)*, 7

h *(*h-methods*)*, 2
h,claytonCopula-method *(*h-methods*)*, 2
h,copula-method *(*h-methods*)*, 2
h,fgmCopula-method *(*h-methods*)*, 2
h,frankCopula-method *(*h-methods*)*, 2
h,galambosCopula-method *(*h-methods*)*, 2
h,gumbelCopula-method *(*h-methods*)*, 2
h,indepCopula-method *(*h-methods*)*, 2
h,normalCopula-method *(*h-methods*)*, 2
h,tCopula-method *(*h-methods*)*, 2
h-methods, 2
hinverse *(*hinverse-methods*)*, 4
hinverse,claytonCopula-method *(*hinverse-methods*)*, 4
hinverse,copula-method *(*hinverse-methods*)*, 4
hinverse,frankCopula-method *(*hinverse-methods*)*, 4
hinverse,indepCopula-method *(*hinverse-methods*)*, 4
hinverse,normalCopula-method *(*hinverse-methods*)*, 4
hinverse,tCopula-method *(*hinverse-methods*)*, 4
hinverse-methods, 4

numericDeriv, *2*

optim, *9*, *11*

pvine *(*Vine-distribution*)*, 7

pvine,CVine-method *(*Vine-distribution*)*, 7
pvine,DVine-method *(*Vine-distribution*)*, 7
pvine-methods *(*Vine-distribution*)*, 7

RVine, *7*
rvine *(*Vine-distribution*)*, 7
rvine,CVine-method *(*Vine-distribution*)*, 7
rvine,DVine-method *(*Vine-distribution*)*, 7
RVine-class *(*RVine-classes*)*, 5
RVine-classes, 5
rvine-methods *(*Vine-distribution*)*, 7

show,CVine-method *(*RVine-classes*)*, 5
show,DVine-method *(*RVine-classes*)*, 5
show,Vine-method *(*Vine-class*)*, 6
show,vineFit-method *(*vineFit-class*)*, 10
show,vineFitML-method *(*vineFitML-class*)*, 11
show,vineGoF-method *(*vineGoF-class*)*, 13
solve_TSP, *14*

uniroot, *4*

Vine, *5*, 5, *6, 7*, *9–11*, *13*, *15, 16*
Vine-class, 6
Vine-distribution, 7
vineFit, 8, *9–11*
vineFit-class, 10
vineFitML, *10*
vineFitML-class, 11
vineGoF, 11, *12, 13*
vineGoF-class, 13
vineLogLik, 13
vineOrder, 14
vineParameters, *11*, 15
vineParameters<- *(*vineParameters*)*, 15
vinePIT, *12*, *17*
vinePIT *(*vinePIT-methods*)*, 16
vinePIT,CVine-method *(*vinePIT-methods*)*, 16
vinePIT,DVine-method *(*vinePIT-methods*)*, 16
vinePIT-methods, 16