

# Package ‘CDVineCopulaConditional’

January 20, 2025

**Type** Package

**Version** 0.1.1

**Date** 2017-03-01 today

**Title** Sampling from Conditional C- and D-Vine Copulas

**Description** Provides tools for sampling from a conditional copula density decomposed via Pair-Copula Constructions as C- or D- vine. Here, the vines which can be used for such a sampling are those which sample as first the conditioning variables (when following the sampling algorithms shown in Aas et al. (2009) <[DOI:10.1016/j.insmatheco.2007.02.001](https://doi.org/10.1016/j.insmatheco.2007.02.001)>). The used sampling algorithm is presented and discussed in Bevacqua et al. (2017) <[DOI:10.5194/hess-2016-652](https://doi.org/10.5194/hess-2016-652)>, and it is a modified version of that from Aas et al. (2009) <[DOI:10.1016/j.insmatheco.2007.02.001](https://doi.org/10.1016/j.insmatheco.2007.02.001)>. A function is available to select the best vine (based on information criteria) among those which allow for such a conditional sampling. The package includes a function to compare scatterplot matrices and pair-dependencies of two multivariate datasets.

**Author** Emanuele Bevacqua [aut, cre]

**Maintainer** Emanuele Bevacqua <[emanuele.bevacqua@uni-graz.at](mailto:emanuele.bevacqua@uni-graz.at)>

**License** GPL (>= 2)

**Encoding** UTF-8

**LazyData** true

**Imports** combinat, graphics, stats, VineCopula

**RoxygenNote** 5.0.1

**Suggests** testthat

**Depends** R (>= 2.10)

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2017-07-28 17:20:10 UTC

## Contents

CDVineCondFit . . . . .	2
-------------------------	---

CDVineCondListMatrices . . . . .	6
CDVineCondRank . . . . .	8
CDVineCondSim . . . . .	11
dataset . . . . .	14
overplot . . . . .	15

<b>Index</b>	<b>18</b>
--------------	-----------

---

CDVineCondFit	<i>Selection of a C- or D- vine copula model for conditional sampling</i>
---------------	---

---

## Description

This function fits either a C- or a D- vine model to a d-dimensional dataset of uniform variables. The fit of the pair-copula families is performed sequentially through the function [RVineCopSelect](#) of the package `VineCopula`. The vine structure is selected among a group of C- and a D- vines which satisfy the requirement discussed in *Bevacqua et al. (2017)*. This group is composed by all C- and D- vines from which the conditioning variables would be sampled as first when following the algorithms from *Aas et al. (2009)*. Alternatively, if the vine matrix describing the vine structure is given to the function, the fit of the pair-copulas is directly performed skipping the vine structure selection procedure.

## Usage

```
CDVineCondFit(data, Nx, treecrit = "AIC", type = "CVine-DVine",
  selectioncrit = "AIC", familyset = NA, indeptest = FALSE,
  level = 0.05, se = FALSE, rotations = TRUE, method = "mle",
  Matrix = FALSE)
```

## Arguments

<code>data</code>	An $N \times d$ data matrix (with uniform margins). The data of the conditioning variable(s) have to occupy the last column(s) of this matrix.
<code>Nx</code>	Number of conditioning variables.
<code>treecrit</code>	Character indicating the criteria used to select the vine. All possible vines are fitted trough the function <a href="#">RVineCopSelect</a> of the package <code>VineCopula</code> . Then the vines are ranked with respect the Akaike information criterion ( <code>treecrit = "AIC"</code> , default) or Bayesian information criterion ( <code>treecrit = "BIC"</code> ). This need the estimation and model selection for all the pairs of all the possible vines, therefore could require long time in case of large datasets, i.e. large $N \times d$ .
<code>type</code>	Type of vine to be fitted: C-Vine: "CVine" or 1; D-Vine: "DVine" or 2; Both C and D-Vine: "CVine-DVine" or "1-2" (default).
<code>selectioncrit</code>	Character indicating the criterion for pair-copula selection. Possible choices are "AIC" (default) and "BIC".

familyset	<p>"Integer vector of pair-copula families to select from. The vector has to include at least one pair-copula family that allows for positive and one that allows for negative dependence. Not listed copula families might be included to better handle limit cases. If familyset = NA (default), selection among all possible families is performed. If a vector of negative numbers is provided, selection among all but abs(familyset) is performed. Coding of bivariate copula families:</p> <ul style="list-style-type: none"> <li>0 = independence copula</li> <li>1 = Gaussian copula</li> <li>2 = Student t copula (t-copula)</li> <li>3 = Clayton copula</li> <li>4 = Gumbel copula</li> <li>5 = Frank copula</li> <li>6 = Joe copula</li> <li>7 = BB1 copula</li> <li>8 = BB6 copula</li> <li>9 = BB7 copula</li> <li>10 = BB8 copula</li> <li>13 = rotated Clayton copula (180 degrees; "survival Clayton")</li> <li>14 = rotated Gumbel copula (180 degrees; "survival Gumbel")</li> <li>16 = rotated Joe copula (180 degrees; "survival Joe")</li> <li>17 = rotated BB1 copula (180 degrees; "survival BB1")</li> <li>18 = rotated BB6 copula (180 degrees; "survival BB6")</li> <li>19 = rotated BB7 copula (180 degrees; "survival BB7")</li> <li>20 = rotated BB8 copula (180 degrees; "survival BB8")</li> <li>23 = rotated Clayton copula (90 degrees)</li> <li>24 = rotated Gumbel copula (90 degrees)</li> <li>26 = rotated Joe copula (90 degrees)</li> <li>27 = rotated BB1 copula (90 degrees)</li> <li>28 = rotated BB6 copula (90 degrees)</li> <li>29 = rotated BB7 copula (90 degrees)</li> <li>30 = rotated BB8 copula (90 degrees)</li> <li>33 = rotated Clayton copula (270 degrees)</li> <li>34 = rotated Gumbel copula (270 degrees)</li> <li>36 = rotated Joe copula (270 degrees)</li> <li>37 = rotated BB1 copula (270 degrees)</li> <li>38 = rotated BB6 copula (270 degrees)</li> <li>39 = rotated BB7 copula (270 degrees)</li> <li>40 = rotated BB8 copula (270 degrees)</li> <li>104 = Tawn type 1 copula</li> <li>114 = rotated Tawn type 1 copula (180 degrees)</li> <li>124 = rotated Tawn type 1 copula (90 degrees)</li> <li>134 = rotated Tawn type 1 copula (270 degrees)</li> <li>204 = Tawn type 2 copula</li> <li>214 = rotated Tawn type 2 copula (180 degrees)</li> <li>224 = rotated Tawn type 2 copula (90 degrees)</li> <li>234 = rotated Tawn type 2 copula (270 degrees)</li> </ul>
indeptest	Logical; whether a hypothesis test for the independence of u1 and u2 is per-

	formed before bivariate copula selection (default: <code>indeptest = FALSE</code> ; see <code>BiCopIndTest</code> ). The independence copula is chosen for a (conditional) pair if the null hypothesis of independence cannot be rejected.
<code>level</code>	numeric; significance level of the independence test (default: <code>level = 0.05</code> ).
<code>se</code>	Logical; whether standard errors are estimated (default: <code>se = FALSE</code> ).
<code>rotations</code>	logical; if TRUE, all rotations of the families in <code>familyset</code> are included.
<code>method</code>	indicates the estimation method: either maximum likelihood estimation ( <code>method = "mle"</code> ; default) or inversion of Kendall's tau ( <code>method = "itau"</code> ). For <code>method = "itau"</code> only one parameter families and the Student t copula can be used ( <code>family = 1, 2, 3, 4, 5, 6, 13, 14, 16, 23, 24, 26, 33, 34</code> or <code>36</code> ). For the t-copula, <code>par2</code> is found by a crude profile likelihood optimization over the interval $(2, 10]$ . (VineCopula Documentation, version 2.1.1, pp. 73-75)
<code>Matrix</code>	$d \times d$ matrix that defines the vine structure. If <code>Matrix</code> is not given, the routine finds the best vine structure according to <code>selectioncrit</code> . If <code>Matrix</code> is given, the fit is performed only if the structure respects the necessary conditions for the conditional sampling (if the conditions are not respected, an error message is returned).

### Value

An `RVineMatrix` object describing the selected copula model (for further details about `RVineMatrix` objects see the documentation file of the `VineCopula` package). The selected families are stored in `$family`, and the sequentially estimated parameters in `$par` and `$par2`. The fit of the model is performed via the function `RVineCopSelect` of the package `VineCopula`.

"The object `RVineMatrix` includes the following information about the fit:

`se`, `se2` standard errors for the parameter estimates (if `se = TRUE`; note that these are only approximate since they do not account for the sequential nature of the estimation,

`nobs` number of observations,

`logLik`, `pair.logLik` log likelihood (overall and pairwise)

`AIC`, `pair.AIC` Aikake's Informaton Criterion (overall and pairwise),

`BIC`, `pair.BIC` Bayesian's Informaton Criterion (overall and pairwise),

`emptau` matrix of empirical values of Kendall's tau,

`p.value.indeptest` matrix of p-values of the independence test.

### Note

For a comprehensive summary of the vine copula model, use `summary(object)`; to see all its contents, use `str(object)`". (VineCopula Documentation, version 2.1.1, pp. 103)

### Author(s)

Emanuele Bevacqua

## References

Bevacqua, E., Maraun, D., Hobaek Haff, I., Widmann, M., and Vrac, M.: Multivariate statistical modelling of compound events via pair-copula constructions: analysis of floods in Ravenna (Italy), *Hydrol. Earth Syst. Sci.*, 21, 2701-2723, <https://doi.org/10.5194/hess-21-2701-2017>, 2017. [\[link\]](#)

Aas, K., Czado, C., Frigessi, A. and Bakken, H.: Pair-copula constructions of multiple dependence, *Insurance: Mathematics and Economics*, 44(2), 182-198, <doi:10.1016/j.insmatheco.2007.02.001>, 2009. [\[link\]](#)

Ulf Schepsmeier, Jakob Stoeber, Eike Christian Brechmann, Benedikt Graeler, Thomas Nagler and Tobias Erhardt (2017). *VineCopula: Statistical Inference of Vine Copulas*. R package version 2.1.1. [\[link\]](#)

## See Also

[CDVineCondSim](#), [CDVineCondRank](#)

## Examples

```
# Example 1

# Read data
data(dataset)
data <- dataset$data[1:100,1:5]

# Define the variables Y and X. X are the conditioning variables,
# which have to be positioned in the last columns of the data.frame
colnames(data) <- c("Y1", "Y2", "X3", "X4", "X5")

## Not run:
# Select and fit a C- vine copula model, requiring that the
RVM <- CDVineCondFit(data, Nx=3, treecrit="BIC", type="CVine", selectioncrit="AIC")
summary(RVM)
RVM$Matrix

## End(Not run)

# Example 2

# Read data
data(dataset)
data <- dataset$data[1:80,1:5]

# Define the variables Y and X. X are the conditioning variables,
# which have to be positioned in the last columns of the data.frame
colnames(data) <- c("Y1", "Y2", "X3", "X4", "X5")

# Define a VineMatrix which can be used for conditional sampling
ListVines <- CDVineCondListMatrices(data, Nx=3)
```

```

Matrix=ListVines$DVine[[1]]
Matrix

## Not run:
# Fit copula families for the defined vine:
RVM <- CDVineCondFit(data,Nx=3,Matrix=Matrix)
summary(RVM)
RVM$Matrix
RVM$family

# check
identical(RVM$Matrix,Matrix)

# Fit copula families for the defined vine, given a group of families to select from:
RVM <- CDVineCondFit(data,Nx=3,Matrix=Matrix,familyset=c(1,2,3,14))
summary(RVM)
RVM$Matrix
RVM$family

# Try to fit copula families for a vine which is not among those
# that allow for conditional sampling:
Matrix
Matrix[which(Matrix==4)]=40
Matrix[which(Matrix==2)]=20
Matrix[which(Matrix==40)]=2
Matrix[which(Matrix==20)]=4
Matrix
RVM <- CDVineCondFit(data,Nx=3,Matrix=Matrix)
RVM

## End(Not run)

```

---

CDVineCondListMatrices

*List of the possible C- and D- vines allowing for conditional simulation*

---

## Description

Provides a list of the C- and D- vines which allow for conditional sampling, under the condition discussed in the description of [CDVineCondFit](#).

## Usage

```
CDVineCondListMatrices(data, Nx, type = "CVine-DVine")
```

**Arguments**

data	An $N \times d$ data matrix. Data of the conditioning variable(s) have to occupy the last column(s) of this matrix.
Nx	Number of conditioning variables.
type	Type of vine to be considered: C-Vine: "CVine" or 1; D-Vine: "DVine" or 2; Both C and D-Vine: "CVine-DVine" or "1-2" (default).

**Value**

Listes of matrices describing C- (`$CVine`) and D- (`$DVine`) Vines. Each matrix corresponds to a vine, according to the same notation used for `RVineMatrix` objects (for further details about `RVineMatrix` objects see the documentation file of the `VineCopula` package). The index  $i$  in the matrix corresponds to the variable in the  $i$ -th column of data.

**Author(s)**

Emanuele Bevacqua

**References**

- Bevacqua, E., Maraun, D., Hobaek Haff, I., Widmann, M., and Vrac, M.: Multivariate statistical modelling of compound events via pair-copula constructions: analysis of floods in Ravenna (Italy), *Hydrol. Earth Syst. Sci.*, 21, 2701-2723, <https://doi.org/10.5194/hess-21-2701-2017>, 2017. [\[link\]](#)
- Aas, K., Czado, C., Frigessi, A. and Bakken, H.: Pair-copula constructions of multiple dependence, *Insurance: Mathematics and Economics*, 44(2), 182-198, <doi:10.1016/j.insmatheco.2007.02.001>, 2009. [\[link\]](#)

**See Also**

[CDVineCondFit](#)

**Examples**

```
# Read data
data(dataset)
data <- dataset$data[1:100,1:5]

# Define the variables Y and X. X are the conditioning variables,
# which have to be positioned in the last columns of the data.frame
colnames(data) <- c("Y1", "Y2", "X3", "X4", "X5")

# List possible D-Vines:
ListVines <- CDVineCondListMatrices(data, Nx=3, "DVine")
ListVines$DVine

# List possible C-Vines:
```

```
ListVines <- CDVineCondListMatrices(data,Nx=3,"CVine")
ListVines$CVine

# List possible C- and D-Vines:
ListVines <- CDVineCondListMatrices(data,Nx=3,"CVine-DVine")
ListVines
```

---

CDVineCondRank      *Ranking of C- and D- vines allowing for conditional simulation*

---

### Description

Provides a ranking of the C- and D- vines which allow for conditional sampling, under the condition discussed in the description of [CDVineCondFit](#).

### Usage

```
CDVineCondRank(data, Nx, treecrit = "AIC", selectioncrit = "AIC",
  familyset = NA, type = "CVine-DVine", indeptest = FALSE, level = 0.05,
  se = FALSE, rotations = TRUE, method = "mle")
```

### Arguments

data	An $N \times d$ data matrix (with uniform margins). Data of the conditioning variable(s) have to occupy the last column(s) of this matrix.
Nx	Number of conditioning variables.
treecrit	Character indicating the criteria used to select the vine. All possible vines are fitted through the function <a href="#">RVineCopSelect</a> of the package <a href="#">VineCopula</a> . Then the vines are ranked with respect to the Akaike information criterion ( <code>treecrit = "AIC"</code> , default) or Bayesian information criterion ( <code>treecrit = "BIC"</code> ). This needs the estimation and model selection for all the pairs of all the possible vines, therefore could require long time in case of large datasets, i.e. large $N \times d$ .
selectioncrit	Character indicating the criterion for pair-copula selection. Possible choices are <code>selectioncrit = "AIC"</code> (default) and <code>"BIC"</code> .
familyset	"Integer vector of pair-copula families to select from. The vector has to include at least one pair-copula family that allows for positive and one that allows for negative dependence. Not listed copula families might be included to better handle limit cases. If <code>familyset = NA</code> (default), selection among all possible families is performed. If a vector of negative numbers is provided, selection among all but <code>abs(familyset)</code> is performed. Coding of bivariate copula families: $\emptyset$ = independence copula 1 = Gaussian copula 2 = Student t copula (t-copula) 3 = Clayton copula



	<p>4 = Gumbel copula  5 = Frank copula  6 = Joe copula  7 = BB1 copula  8 = BB6 copula  9 = BB7 copula  10 = BB8 copula  13 = rotated Clayton copula (180 degrees; "survival Clayton")  14 = rotated Gumbel copula (180 degrees; "survival Gumbel")  16 = rotated Joe copula (180 degrees; "survival Joe")  17 = rotated BB1 copula (180 degrees; "survival BB1")  18 = rotated BB6 copula (180 degrees; "survival BB6")  19 = rotated BB7 copula (180 degrees; "survival BB7")  20 = rotated BB8 copula (180 degrees; "survival BB8")  23 = rotated Clayton copula (90 degrees)  24 = rotated Gumbel copula (90 degrees)  26 = rotated Joe copula (90 degrees)  27 = rotated BB1 copula (90 degrees)  28 = rotated BB6 copula (90 degrees)  29 = rotated BB7 copula (90 degrees)  30 = rotated BB8 copula (90 degrees)  33 = rotated Clayton copula (270 degrees)  34 = rotated Gumbel copula (270 degrees)  36 = rotated Joe copula (270 degrees)  37 = rotated BB1 copula (270 degrees)  38 = rotated BB6 copula (270 degrees)  39 = rotated BB7 copula (270 degrees)  40 = rotated BB8 copula (270 degrees)  104 = Tawn type 1 copula  114 = rotated Tawn type 1 copula (180 degrees)  124 = rotated Tawn type 1 copula (90 degrees)  134 = rotated Tawn type 1 copula (270 degrees)  204 = Tawn type 2 copula  214 = rotated Tawn type 2 copula (180 degrees)  224 = rotated Tawn type 2 copula (90 degrees)  234 = rotated Tawn type 2 copula (270 degrees)" (VineCopula Documentation, version 2.1.1, pp. 73-74)</p>
type	Type of vine to be fitted: C-Vine: "CVine" or 1; D-Vine: "DVine" or 2; Both C and D-Vine: "CVine-DVine" or "1-2" (default).
indeptest	"Logical; whether a hypothesis test for the independence of u1 and u2 is performed before bivariate copula selection (default: indeptest = FALSE; see BiCopIndTest). The independence copula is chosen for a (conditional) pair if the null hypothesis of independence cannot be rejected.
level	numeric; significance level of the independence test (default: level = 0.05).
se	Logical; whether standard errors are estimated (default: se = FALSE).

rotations logical; if TRUE, all rotations of the families in familyset are included.

method indicates the estimation method: either maximum likelihood estimation (method = "mle"; default) or inversion of Kendall's tau (method = "itau"). For method = "itau" only one parameter families and the Student t copula can be used (family = 1, 2, 3, 4, 5, 6, 13, 14, 16, 23, 24, 26, 33, 34 or 36). For the t-copula, par2 is found by a crude profile likelihood optimization over the interval (2, 10]." (VineCopula Documentation, version 2.1.1, pp. 74-75)

### Value

table A table with the ranking of the vines, with vine index *i*, values of the selected treecrit and vine type (1 for "CVine" and 2 for D-Vine).

vines A list where the element `[[i]]` is an `RVineMatrix` object corresponding to the *i*-th vine in the ranking shown in table. Each `RVineMatrix` object contains the selected families (`$family`) as well as sequentially estimated parameters stored in `$par` and `$par2`. Details about `RVineMatrix` objects are given in the documentation file of the VineCopula package). The fit of each model is performed via the function `RVineCopSelect` of the package VineCopula. "The object is augmented by the following information about the fit:

se, se2 standard errors for the parameter estimates (if `se = TRUE`; note that these are only approximate since they do not account for the sequential nature of the estimation

nobs number of observations

logLik, pair.logLik log likelihood (overall and pairwise)

AIC, pair.AIC Aikake's Informaton Criterion (overall and pairwise)

BIC, pair.BIC Bayesian's Informaton Criterion (overall and pairwise)

emptau matrix of empirical values of Kendall's tau

p.value.indeptest matrix of p-values of the independence test.

### Note

For a comprehensive summary of the vine copula model, use `summary(object)`; to see all its contents, use `str(object)`." (VineCopula Documentation, version 2.1.1, pp. 103)

### Author(s)

Emanuele Bevacqua

### References

- Bevacqua, E., Maraun, D., Hobaek Haff, I., Widmann, M., and Vrac, M.: Multivariate statistical modelling of compound events via pair-copula constructions: analysis of floods in Ravenna (Italy), Hydrol. Earth Syst. Sci., 21, 2701-2723, <https://doi.org/10.5194/hess-21-2701-2017>, 2017. [\[link\]](#)
- Aas, K., Czado, C., Frigessi, A. and Bakken, H.: Pair-copula constructions of multiple dependence, Insurance: Mathematics and Economics, 44(2), 182-198, <doi:10.1016/j.insmatheco.2007.02.001>, 2009. [\[link\]](#)
- Ulf Schepsmeier, Jakob Stoeber, Eike Christian Brechmann, Benedikt Graeler, Thomas Nagler and Tobias Erhardt (2017). VineCopula: Statistical Inference of Vine Copulas. R package version 2.1.1. [\[link\]](#)

**See Also**[CDVineCondFit](#)**Examples**

```

# Read data
data(dataset)
data <- dataset$data[1:100,1:5]

# Define the variables Y and X. X are the conditioning variables,
# which have to be positioned in the last columns of the data.frame
colnames(data) <- c("Y1","Y2","X3","X4","X5")

# Rank the possible D-Vines according to the AIC
## Not run:
Ranking <- CDVineCondRank(data,Nx=3,"AIC",type="DVine")
Ranking$table
#   tree      AIC type
# 1     1 -292.8720   2
# 2     2 -290.2941   2
# 3     3 -288.5719   2
# 4     4 -288.2496   2
# 5     5 -287.8006   2
# 6     6 -285.8503   2
# 7     7 -282.2867   2
# 8     8 -278.9371   2
# 9     9 -275.8339   2
# 10    10 -272.9459   2
# 11    11 -271.1526   2
# 12    12 -270.5269   2

Ranking$vines[[1]]$AIC
# [1] -292.8720
summary(Ranking$vines[[1]])

## End(Not run)

```

---

CDVineCondSim

*Simulation from a conditional C- or D-vine*


---

**Description**

Simulates from a d-dimensional conditional C- or D-vine of the variables  $(\mathbf{Y}, \mathbf{X})$ , given the fixed conditioning variables  $\mathbf{X}$ . The algorithm works for vines satisfying the requirements discussed in *Bevacqua et al. (2017)*. The algorithm implemented here is a modified version of those from *Aas et al. (2009)* and is shown in *Bevacqua et al. (2017)*.

**Usage**

```
CDVineCondSim(RVM, Condition, N)
```

**Arguments**

RVM	An <a href="#">RVineMatrix</a> object containing the information of the d-dimensional C- or D-Vine model (for further details about <a href="#">RVineMatrix</a> objects see the documentation file of the <a href="#">VineCopula</a> package). If the full copula is 2-dimensional, RVM can be an <a href="#">RVineMatrix</a> object or a <code>data.frame</code> (or <code>list</code> ) object where <code>\$family</code> , <code>\$par</code> and <code>\$par2</code> are specified.
Condition	A $N \times N$ matrix of the $N$ conditioning variables. For D-vine: data corresponding to the conditioning variable whose index is in <code>RVM\$Matrix[i, i]</code> , are in $i$ -th column of <code>Condition</code> . For C-vine: data corresponding to the conditioning variable whose index is in <code>RVM\$Matrix[i, i]</code> , are in $[(d+1)-i]$ -th column of <code>Condition</code> . See examples.
N	Number of data to be simulated. By default $N$ is taken from <code>Condition</code> , which is a $N \times N$ matrix. It is necessary to specify $N$ only when <code>Condition</code> is not given.

**Value**

A  $N \times d$  matrix of the simulated variables from the given C- or D-vine copula model. In the first columns there are the simulated conditioned variables, and in the last columns the conditioning variables `Condition`. For more details about the exact order of the variables in the columns see the examples. The function is built to work easily in combination with [CDVineCondFit](#).

**Author(s)**

Emanuele Bevacqua

**References**

- Bevacqua, E., Maraun, D., Hobaek Haff, I., Widmann, M., and Vrac, M.: Multivariate statistical modelling of compound events via pair-copula constructions: analysis of floods in Ravenna (Italy), *Hydrol. Earth Syst. Sci.*, 21, 2701-2723, <https://doi.org/10.5194/hess-21-2701-2017>, 2017. [\[link\]](#)
- Aas, K., Czado, C., Frigessi, A. and Bakken, H.: Pair-copula constructions of multiple dependence, *Insurance: Mathematics and Economics*, 44(2), 182-198, <doi:10.1016/j.insmatheco.2007.02.001>, 2009. [\[link\]](#)
- Ulf Schepsmeier, Jakob Stoeber, Eike Christian Brechmann, Benedikt Graeler, Thomas Nagler and Tobias Erhardt (2017). *VineCopula: Statistical Inference of Vine Copulas*. R package version 2.1.1. [\[link\]](#)

**See Also**

[CDVineCondFit](#)

**Examples**

```

# Example 1: conditional sampling from a C-Vine

# Read data
data(dataset)
data <- dataset$data[1:400,1:4]

# Define the variables Y and X. X are the conditioning variables,
# which have to be positioned in the last columns of the data.frame
colnames(data) <- c("Y1", "Y2", "X3", "X4")

## Not run:
# Select a vine and fit the copula families, specifying that there are 2 conditioning variables
RVM <- CDVineCondFit(data, Nx=2, type="CVine")

# Set the values of the conditioning variables as those used for the calibration.
# Order them with respect to RVM$Matrix, considering that is a C-Vine
d=dim(RVM$Matrix)[1]
cond1 <- data[,RVM$Matrix[(d+1)-1,(d+1)-1]]
cond2 <- data[,RVM$Matrix[(d+1)-2,(d+1)-2]]
condition <- cbind(cond1,cond2)

# Simulate the variables
Sim <- CDVineCondSim(RVM,condition)

# Plot the simulated variables over the observed
Sim <- data.frame(Sim)
overplot(Sim,data)

# Example 2: conditional sampling from a D-Vine

# Read data
data(dataset)
data <- dataset$data[1:100,1:4]

# Define the variables Y and X. X are the conditioning variables,
# which have to be positioned in the last columns of the data.frame
colnames(data) <- c("Y1", "Y2", "X3", "X4")

# Select a vine and fit the copula families, specifying that there are 2 conditioning variables
RVM <- CDVineCondFit(data, Nx=2, type="DVine")
summary(RVM) #It is a D-Vine.

# Set the values of the conditioning variables as those used for the calibration.
# Order them with respect to RVM$Matrix, considering that is a D-Vine.
cond1 <- data[,RVM$Matrix[1,1]]
cond2 <- data[,RVM$Matrix[2,2]]
condition <- cbind(cond1,cond2)

# Simulate the variables

```

```

Sim <- CDVineCondSim(RVM,condition)

# Plot the simulated variables over the observed
Sim <- data.frame(Sim)
overplot(Sim,data)

# Example 3

# Read data
data(dataset)
data <- dataset$data[1:100,1:2]
colnames(data) <- c("Y1","X2")

# Fit copula
require(VineCopula)
BiCop <- BiCopSelect(data$Y1,data$X2)
BiCop

# Fix conditioning variable to low values and simulate
condition <- data$X2/10
Sim <- CDVineCondSim(BiCop,condition)

# Plot the simulated variables over the observed
Sim <- data.frame(Sim)
overplot(Sim,data)

## End(Not run)

```

---

dataset

*Random dataset from a given vine copula model*


---

### Description

A random dataset simulated from a given 5-dimensional vine copula model.

### Usage

```
dataset
```

### Format

\$data An 1000 x 5 data set (format `data.frame`) with the uniform variables (U1, U2, U3, U4, U5).  
 \$vine [RVineMatrix](#) object defining the vine copula model from where \$data was sampled.

### Author(s)

Emanuele Bevacqua

**Examples**

```
# Load data
data(dataset)

# Extract data
data <- dataset$data
plot(data)

# Extract the RVineMatrix object from where the dataset was randomly sampled
vine <- dataset$vine
vine$Matrix
vine$family
vine$par
vine$par2
summary(vine)
```

---

overplot

*overplot*


---

**Description**

This function overlays the scatterplot matrices of two multivariate datasets. Moreover, it shows the dependencies among all the pairs for both datasets.

**Usage**

```
overplot(data1, data2, col1 = "black", col2 = "grey", xlim = NA,
  ylim = NA, labels = NA, method = "pearson", cex.cor = 1,
  cex.labels = 1, cor.signif = 2, cex.axis = 1, pch1 = 1, pch2 = 1)
```

**Arguments**

data1, data2	Two N x d matrices of data to be plotted.
col1, col2	Colors used for data1 and data2 during the plot. Default is col1="black" and col2="grey".
xlim, ylim	Two bidimensional vectors indicating the limits of x and y axes for all the scatterplots. If not given, they are automatically computed for each of the scatterplots.
labels	A character vector with the variable names to be printed (if not given, the names of data1 variables are printed).
method	Character indicating the dependence types to be computed between the pairs. Possibilities: "kendall", "spearman" and "pearson" (default)
cex.cor	Number: character dimension of the printed dependencies. Default cex.cor=1.

<code>cex.labels</code>	Number: character dimension of the printed variable names. Default <code>cex.labels=1</code> .
<code>cor.signif</code>	Number: number of significant numbers of the printed dependencies. Default <code>cor.signif=2</code> .
<code>cex.axis</code>	Number: dimension of the axis numeric values. Default <code>cex.axis=1</code> .
<code>pch1, pch2</code>	Paramter to specify the symbols to use when plotting points of <code>data1</code> and <code>data2</code> . Default is <code>pch1=1</code> and <code>pch2=1</code> .

### Value

A matrix of overlaying scatterplots of the multivariate datasets `data1` and `data2`, with the dependencies of the pairs.

### Author(s)

Emanuele Bevacqua

### Examples

```
# Example 1

# Read and prepare the data for the plot
data(dataset)
data1 <- dataset$data[1:300,]
data2 <- dataset$data[301:600,]
overplot(data1, data2, xlim=c(0,1), ylim=c(0,1), method="kendall")

## Not run:
# Example 2

# Read and prepare the data for the plot
data(dataset)
data <- dataset$data[1:200, 1:5]
colnames(data) <- c("Y1", "Y2", "X3", "X4", "X5")

# Fit copula families for the defined vine:
ListVines <- CDVineCondListMatrices(data, Nx=3)
Matrix=ListVines$CVine[[1]]
RVM <- CDVineCondFit(data, Nx=3, Matrix=Matrix)

# Simulate data:
d=dim(RVM$Matrix)[1]
cond1 <- data[, RVM$Matrix[(d+1)-1, (d+1)-1]]
cond2 <- data[, RVM$Matrix[(d+1)-2, (d+1)-2]]
cond3 <- data[, RVM$Matrix[(d+1)-3, (d+1)-3]]
condition <- cbind(cond1, cond2, cond3)
Sim <- CDVineCondSim(RVM, condition)

# Plot the simulated variables Sim over the observed
Sim <- data.frame(Sim)
```



```
overplot(data[,1:2],Sim[,1:2],xlim=c(0,1),ylim=c(0,1),method="spearman")  
overplot(data,Sim,xlim=c(0,1),ylim=c(0,1),method="spearman")  
  
## End(Not run)
```

# Index

## \* **datasets**

dataset, [14](#)

CDVineCondFit, [2](#), [6–8](#), [11](#), [12](#)

CDVineCondListMatrices, [6](#)

CDVineCondRank, [5](#), [8](#)

CDVineCondSim, [5](#), [11](#)

dataset, [14](#)

overplot, [15](#)

RVineCopSelect, [2](#), [4](#), [8](#), [10](#)

RVineMatrix, [4](#), [7](#), [10](#), [12](#), [14](#)