# Package 'penDvine'

October 14, 2022

**Type** Package

**Title** Flexible Pair-Copula Estimation in D-Vines using Bivariate Penalized Splines

**Version** 0.2.4

**Date** 2015-07-02

**Depends** R (>= 2.15.1), lattice, TSP, fda, Matrix, foreach

**Imports** quadprog, latticeExtra, doParallel

**Author** Christian Schellhase <cschellhase@wiwi.uni-bielefeld.de>

**Maintainer** Christian Schellhase <cschellhase@wiwi.uni-bielefeld.de>

**Description** Flexible Pair-Copula Estimation in D-vines using Bivariate Penalized Splines.

**License** GPL (>= 2)

**LazyLoad** yes

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2015-07-09 18:11:27

## R topics documented:

---

`penDvine-package` *The package 'penDvine' offers routines for estimating densities and copula distribution of D-vines.*

---

## Description

The package 'penDvine' offers routines for estimating densities and distribution of D-vines. For details see the description of the function Dvine().

## Details

| | |
|---|---|
| Package: | penDvine |
| Type: | Package |
| Version: | 0.2.4 |
| Date: | 2015-07-02 |
| License: GPL (>= 2) LazyLoad: | yes |

The packages contributes the function 'Dvine()' for estimating densities and distributions of Dvines using penalized splines techniques.

## Author(s)

Christian Schellhase <cschellhase@wiwi.uni-bielefeld.de>

## References

Flexible Pair-Copula Estimation in D-vines using Bivariate Penalized Splines, Kauermann G. and Schellhase C. (2014), Statistics and Computing 24(6): 1081-1100).

## Examples

```
#This examples describes the estimation of a D-vine to winddata,
#available in this package. After the margins are prepared, we estimate a
#D-vine using B-splines with 9 (K+1) marginal knots and penalizing second (m=2)
#order differences (pen=1) of the basis coefficients.

data(Winddata)
ex.data<-Winddata[c(1:100),c(1:4)] #examplary subdata for fast calculation
```

```
wind.example<-Dvine(ex.data,K=7,pen=1,lambda=100,order.Dvine=FALSE,base="B-spline",m=2,cores=2)
```

---

`bernstein` *Flexible Pair-Copula Estimation in D-vines with Penalized Splines*

---

## Description

Calculation of Bernstein Polynomials, following the formula (choose(n, v) * x^v * (1 - x)^(n - v)) * (n + 1) .

## Usage

```
bernstein(v,x,n)
```

## Arguments

| | |
|---|---|
| v | n choose v. |
| x | Argument between 0 and 1. |
| n | n choose v. |

## Value

Returning a matrix, containing Bernstein Polynomials.

## Note

bernstein and bernstein2 are identical, only the first argument of the functions differs for different applications of 'apply'.

## Author(s)

Christian Schellhase <cschellhase@wiwi.uni-bielefeld.de>

## References

Flexible Pair-Copula Estimation in D-vines using Bivariate Penalized Splines, Kauermann G. and Schellhase C. (2014+), Statistics and Computing (to appear).

cal.Dvine *Flexible Pair-Copula Estimation in D-vines with Penalized Splines*

## Description

Calculating the density of the estimated Dvine at the point(s) val.

## Usage

```
cal.Dvine(obj,val)
```

## Arguments

| | |
|---|---|
| obj | object of class 'penDvine', result of 'Dvine'. |
| val | Values in which the current Dvine should be evaluated. |

## Details

The current Dvine is evaluated in val and the corresponding density values are returned.

## Value

The returing values are the density of the current Dvine at the point(s) 'val'.

## Author(s)

Christian Schellhase <cschellhase@wiwi.uni-bielefeld.de>

## References

Flexible Pair-Copula Estimation in D-vines using Bivariate Penalized Splines, Kauermann G. and Schellhase C. (2014+), Statistics and Computing (to appear).

cond.cop *Flexible Pair-Copula Estimation in D-vines with Penalized Splines*

## Description

Calculation of the conditional paircopulas.

## Usage

```
cond.cop(data,coef,K,diff="u2",Index.basis.D,base,q=2)
```

### Arguments

| | |
|---|---|
| data | Considered bivariate data, later used as "u1" and "u2". |
| coef | Considered coefficients of the splines. |
| K | Number of splines. |
| diff | Default="u2", alternatively diff="u1". Determining in which direction the expression is differentiated. |
| Index.basis.D | Vector of indices built in the program before. |
| base | "B-spline" or "Bernstein". |
| q | Order of the B-spline, default q=2 |

### Author(s)

Christian Schellhase <cschellhase@wiwi.uni-bielefeld.de>

### References

Flexible Pair-Copula Estimation in D-vines using Bivariate Penalized Splines, Kauermann G. and Schellhase C. (2014+), Statistics and Computing (to appear).

---

## Derv1 — *Calculating the first derivative of the paircopula likelihood function w.r.t. parameter b*

---

### Description

Calculating the first derivative of the paircopula likelihood function w.r.t. parameter v.

### Usage

```
Derv1(penden.env,temp=FALSE,lambda=NULL)
```

### Arguments

| | |
|---|---|
| penden.env | Containing all information, environment of paircopula(). |
| temp | Default=FALSE,if TRUE temporary calculations of optimal parameters are done. |
| lambda | Default=NULL, i.e. the saved smoothing parameter lambda in the environment is used. Alternatively, temporary values of lambda are used for optimization of lambda. |

## Details

The calculation of the first derivative of the paircopula likelihood function w.r.t. b equals

$$s(v,\lambda) = \partial l(v,\lambda)/\partial v = \sum_{i=1}^{n} \Phi(u_i)/c(u_i,v) - P(\lambda)v$$

with

$$P(\lambda)$$

is the penalty matrix, saved in the environment.

## Value

`Derv1.pen` first order derivation of the penalized likelihood function w.r.t. parameter v.

Derv1.pen is saved in the environment.

## Author(s)

Christian Schellhase <cschellhase@wiwi.uni-bielefeld.de>

## References

Flexible Pair-Copula Estimation in D-vines using Bivariate Penalized Splines, Kauermann G. and Schellhase C. (2014+), Statistics and Computing (to appear).

---

# Derv2 *Calculating the second order derivative of the paircopula likelihood function w.r.t. parameter b*

---

## Description

Calculating the second order derivative of the paircopula likelihood function w.r.t. parameter v.

## Usage

```
Derv2(penden.env,temp=FALSE,lambda=NULL)
```

## Arguments

| | |
|---|---|
| `penden.env` | Containing all information, environment of paircopula(). |
| `temp` | Default=FALSE,if TRUE temporary calculations of optimal parameters are done. |
| `lambda` | Default=NULL, i.e. the saved smoothing parameter lambda in the environment is used. Alternatively, temporary values of lambda are used for optimization of lambda. |

## Details

We approximate the second order derivative in this approach with the negative fisher information.

## Value

| | |
|---|---|
| Derv2.pen | second order derivative w.r.t. v with penalty |
| Derv2.cal | second order derivative w.r.t. v without penalty. Needed for calculating of e.g. AIC. |

Derv2.cal and Derv2.pen are saved in the environment.

## Author(s)

Christian Schellhase <cschellhase@wiwi.uni-bielefeld.de>

## References

Flexible Pair-Copula Estimation in D-vines using Bivariate Penalized Splines, Kauermann G. and Schellhase C. (2014+), Statistics and Computing (to appear).

---

# Dvine *Flexible Pair-Copula Estimation in D-vines with Penalized Splines*

---

## Description

Calculating D-vines with penalized B-splines or penalized Bernstein polynomials

## Usage

```
Dvine(data,K=8,lambda=100,order.Dvine=TRUE,pen=1,base="Bernstein",m=2,cores=NULL,q=2)
```

## Arguments

| | |
|---|---|
| data | 'data' contains the data. 'data' has to be a matrix or a data.frame with two columns. |
| K | K is the degree of the Bernstein polynomials. In the case of linear B-spline basis functions, K+1 nodes are used for the basis functions. |
| lambda | Starting value for lambda, default is lambda=100. |
| order.Dvine | Indicating if the first level of the Dvine is ordered, default (order.Dvine=TRUE). |
| pen | 'pen' indicates the used penalty. 'pen=1' for the difference penalty of m-th order. 'pen=2' is only implemented for Bernstein polynomials, it is the penalty based on the integrated squared second order derivatives of the Bernstein polynomials. |
| base | Type of basis function, default is "Bernstein". An alternative is base="B-spline". |
| m | Indicating the order of differences to be penalised. Default is "m=2". |
| cores | Default=NULL, the number of cpu cores used for parallel computing can be specified. |
| q | Order of B-splines. Default is q=2, NULL if Bernstein polynomials are used. |

## Details

The calculation of the Dvine is done stepwise. If the option 'order.Dvine' is selected, the order of the first level of the Dvine is specifed. From the second level, each paircopula is calculated (parallel or not) until the highest level. The specifications in 'Dvine' are done for every paircopula in the Dvine. There is no option to change parameters for some paircopulas.

## Value

Returning a list containing

| | |
|---|---|
| `Dvine` | an object of class 'Dvine' |
| `log.like` | the estimated log-likelihood |
| `AIC` | AIC value |
| `cAIC` | corrected AIC value |
| `K` | Number of K |
| `order` | the used order of the first level |
| `S` | Sequence seq(1:(dim(data)[2])) |
| `N` | Number of observations, that is dim(data)[1] |
| `base` | Used basis function |

## Author(s)

Christian Schellhase <cschellhase@wiwi.uni-bielefeld.de>

## References

Flexible Pair-Copula Estimation in D-vines using Bivariate Penalized Splines, Kauermann G. and Schellhase C. (2014+), Statistics and Computing (to appear).

---

`f.hat.val` *Calculating the actual fitted values 'f.hat.val' of the estimated density function*

---

## Description

Calculating the actual fitted values of the response, depending on the actual parameter set v

## Usage

```
f.hat.val(penden.env,cal=FALSE,temp=FALSE)
```

### Arguments

| | |
|---|---|
| penden.env | Containing all information, environment of paircopula() |
| cal | if TRUE, the final weights of one iteration are used for the calculation of the fitted values. |
| temp | if TRUE, the iteration for optimal weights is still in progress and the temporary weights are used for calculation of the fitted values. |

### Details

Calculating the actual fitted values of the response, depending on the actual parameter set v. Multiplying the actual set of parameters $v$ with the base 'base.den' delivers the fitted values.

### Value

| | |
|---|---|
| f.hat.val | Fitted values for the current coefficents |

. f.hat.val is saved in the environment.

### Author(s)

Christian Schellhase <cschellhase@wiwi.uni-bielefeld.de>

### References

Flexible Pair-Copula Estimation in D-vines using Bivariate Penalized Splines, Kauermann G. and Schellhase C. (2014+), Statistics and Computing (to appear).

---

## marg.likelihood *Calculating the marginal likelihood*

---

### Description

Calculating the marginal likelihood of paircopula().

### Usage

```
marg.likelihood(penden.env,pen.likelihood,temp=FALSE)
```

### Arguments

| | |
|---|---|
| penden.env | Containing all information, environment of paircopula(). |
| pen.likelihood | Actual penalized likelihood for calculation, temporary or not. |
| temp | Default=FALSE, indicating if temporary values throughout iteration are calculated. |

### Value

| | |
|---|---|
| marg.log.like | Marginal log-likelihood, saved in the environment |

### Author(s)

Christian Schellhase <cschellhase@wiwi.uni-bielefeld.de>

### References

Flexible Pair-Copula Estimation in D-vines using Bivariate Penalized Splines, Kauermann G. and Schellhase C. (2014+), Statistics and Computing (to appear).

---

## my.IC *Calculating the AIC-, cAIC- and BIC-value*

---

### Description

Calculating the AIC-, cAIC- and BIC- value of the paircopula density estimation. Therefore, we add the unpenalized log likelihood of the estimation and the degree of freedom.

### Usage

```
my.IC(penden.env,temp=FALSE)
```

### Arguments

| | |
|---|---|
| penden.env | Containing all information, environment of paircopula() |
| temp | Default=FALSE, if TRUE temporary values of AIC, cAIC and BIC are calculated. |

### Details

AIC is calculated as $AIC(\lambda) = -2 * l(\mathbf{u}, \hat{\mathbf{v}}) + 2 * df(\lambda)$

cAIC is calculated as $AIC(\lambda) = -2 * l(\mathbf{u}, \hat{\mathbf{v}}) + 2 * df(\lambda) + (2 * df * (df + 1))/(n - df - 1)$

BIC is calculated as $BIC(\lambda) = 2 * l(\mathbf{u}, \hat{\mathbf{v}}) + 2 * df(\lambda) * log(n)$

### Value

| | |
|---|---|
| AIC | sum of twice the negative non-penalized log likelihood and mytrace |
| cAIC | corrected AIC. |
| trace | calculated mytrace as the sum of the diagonal matrix df, which results as the product of the inverse of the penalized second order derivative of the log likelihood with the non-penalized second order derivative of the log likelihood |
| BIC | sum of twice the non-penalized log likelihood and log(n) |

All values are saved in the environment.

### Author(s)

Christian Schellhase <cschellhase@wiwi.uni-bielefeld.de>

## References

Flexible Pair-Copula Estimation in D-vines using Bivariate Penalized Splines, Kauermann G. and Schellhase C. (2014+), Statistics and Computing (to appear).

---

`my.loop` *Iterative loop for calculating the optimal coefficients 'v'.*

---

## Description

Calculating the optimal coefficients 'v' iteratively, using quadratic programing.

## Usage

```
my.loop(penden.env)
```

## Arguments

| | |
|---|---|
| `penden.env` | Containing all information, environment of pencopula() |

## Details

'my.loop' optimates the log-likelihhod iteratively. Therefore, the routine checks a) the relative chance in the optimal lambda and stops the iteration, if the relative change of lambda is less than one percent. During the calculations of new weights 'v' in the routine 'new.weights', most of the values are called '.temp'. This add on underlines the temporarily values. Alternatively b) for fixed lambda, 'my.loop' checks the relative change in the weights. If the change of a) the optimal lambda or b) of the basis coefficients 'v' are greater than one percent, the the real values are overwritten with the '.temp' values.

## Value

| | |
|---|---|
| `liste` | The results of each iteration are written in a matrix called 'liste', saved in the environment. 'liste' contains the penalized log-likelihood, the log-likelihood, 'lambda' and the weights 'v'. |

## Author(s)

Christian Schellhase <cschellhase@wiwi.uni-bielefeld.de>

## References

Flexible Pair-Copula Estimation in D-vines using Bivariate Penalized Splines, Kauermann G. and Schellhase C. (2014+), Statistics and Computing (to appear).

`new.weights` *Calculating new weights v.*

## Description

Calculating new weights v using quadratic programing.

## Usage

```
new.weights(penden.env, lambda.temp=NULL)
```

## Arguments

| | |
|---|---|
| `penden.env` | Containing all information, environment of paircopula() |
| `lambda.temp` | Default=NULL, if optimal lambda is calculated, the lambda are saved temporarily, also the resulted coefficients are saved temporarily until some convergences. |

## Details

The new weights are calculated solving a quadratic program. Therefore, the derivates of first and second order are needed, 'Derv1.pen' and 'Derv2.pen'. Moreover, we have to fulfill the side conditions v>=0, sum(v)=1 and that the marginal densities are uniform. All side conditions are saved as 'AA.help' in the environment.

If the quadratic program does not find a new feasible solution, the whole program terminates. For solving the quadratic program, we use the function 'solve.QP' from the R-package 'quadprog'.

## Value

| | |
|---|---|
| `ck.val.temp` | Calculated new values for the weights 'v'. The add on 'temp' means, that there is a check in the next step if the weights 'v' have been converted (in the case of fixed lambda). If converted, the new values 'ck.val.temp' are unnoted. If not converted, 'ck.val.temp' become the ordinary 'ck.val' for the next iteration. This check is done in my.loop. If the optimal value of lambda is calculated, the coefficients 'ck.val.temp' become the ordinary 'ck.val' for the next iteration if lambda is converted. t |

'ck.val.temp' is saved in the environment.

## Author(s)

Christian Schellhase <cschellhase@wiwi.uni-bielefeld.de>

## References

Flexible Pair-Copula Estimation in D-vines using Bivariate Penalized Splines, Kauermann G. and Schellhase C. (2014+), Statistics and Computing (to appear).

`order.Dvine` *Ordering the first level of the Dvine.*

## Description

Ordering the first level of the Dvine, depending on the pairwise cAIC-values.

## Usage

```
order.Dvine(help.env)
```

## Arguments

| | |
|---|---|
| `help.env` | Containing all information, environment of Dvine() |

## Details

Beginning in the top tree level of a p-dimensional D-vine, we calculate all (p,2) 'p over 2' marginal pairwise copulas fitted by penalized splines. For each pair (i,j) this gives the fitted maximized likelihood value. We order the variable pairs, subject to their increasing estimated pairwise cAIC and start with the pair of covariates with lowest estimated cAIC. We now select the pairs of variables such that the resulting selection gives a tree. The problem of finding this selection is equivalent to solve a traveler salesman problem by interpreting the cAIC as distance measure between two variables. Once this problem is solved, the specification of the first tree level completely defines the D-vine

## Value

| | |
|---|---|
| `order` | Calculated order of the first level of the Dvine, saved in the environment. |

## Author(s)

Christian Schellhase <cschellhase@wiwi.uni-bielefeld.de>

## References

Flexible Pair-Copula Estimation in D-vines using Bivariate Penalized Splines, Kauermann G. and Schellhase C. (2014+), Statistics and Computing (to appear).

paircopula *Flexible Pair-Copula Estimation in D-vines using Bivariate Penalized Splines*

## Description

Calculating paircopula with penalized B-splines or penalized Bernstein polynomials

## Usage

```
paircopula(data,K=8,base="Bernstein",max.iter=30,lambda=100,
       data.frame=parent.frame(),m=2,fix.lambda=FALSE,pen=1,q=2)
```

## Arguments

| | |
|---|---|
| data | 'data' contains the data. 'data' has to be a matrix or a data.frame with two columns. |
| K | K is the degree of the Bernstein polynomials. In the case of linear B-spline basis functions, K+1 nodes are used for the basis functions. |
| base | Type of basis function, default is "Bernstein". An alternative is base="B-spline". |
| max.iter | maximum number of iteration, the default is max.iter=30. |
| lambda | Starting value for lambda, default is lambda=100. |
| data.frame | reference to the data. Default reference is the parent.frame(). |
| m | Indicating the order of differences to be penalised. Default is "m=2". |
| fix.lambda | Determining if lambda is fixed or if the iteration for an optimal lambda is used, default 'fix.lambda=FALSE'. |
| pen | 'pen' indicates the used penalty. 'pen=1' for the difference penalty of m-th order. 'pen=2' is only implemented for Bernstein polynomials, it is the penalty based on the integrated squared second order derivatives of the Bernstein polynomials. Due to numerical difficulties handling the integral of Bernstein polynomials (that is the beta function), this approach works only for K<=15. |
| q | Order of B-spline basis, i.e. default q=2 means linear B-spline basis. |

## Details

Each paircopula is calculated using Bernstein polynomials or B-spline densities as basis functions. Optimal coefficients and optimal penalty parameter lambda are selected iteratively using quadratic programming.

## Value

Returning an object of class 'paircopula', consisting of the environment 'penden.env', which includes all values.

### Author(s)

Christian Schellhase <cschellhase@wiwi.uni-bielefeld.de>

### References

Flexible Pair-Copula Estimation in D-vines using Bivariate Penalized Splines, Kauermann G. and Schellhase C. (2014+), Statistics and Computing (to appear).

---

## pen.log.like *Calculating the log likelihood*

---

### Description

Calculating the considered log likelihood.

### Usage

```
pen.log.like(penden.env, cal=FALSE, temp=FALSE)
```

### Arguments

| | |
|---|---|
| penden.env | Containing all information, environment of paircopula() |
| cal | if TRUE, the final weights of one iteration are used for the calculation of the penalized log likelihood. |
| temp | if TRUE, the iteration for optimal weights is still in progress and the temporary weights are used for calculation. |

### Details

The calculation depends on the estimated weights v, the penalized splines Phi and the penalty paramters lambda.

$$l(v,\lambda) = \sum_{i=1}^{n} \left[ \log\{\sum_{i=1}^{n} \mathbf{\Phi}(u_i)\} v \right] - \frac{1}{2} v^T \boldsymbol{P}(\lambda) b$$

The needed values are saved in the environment.

### Value

| | |
|---|---|
| pen.log.like | Penalized log likelihood of the paircopula density. |
| log.like | Log-Likelihood of the paircopula density. |

The values are saved in the environment.

### Author(s)

Christian Schellhase <cschellhase@wiwi.uni-bielefeld.de>

## References

Flexible Pair-Copula Estimation in D-vines using Bivariate Penalized Splines, Kauermann G. and Schellhase C. (2014+), Statistics and Computing (to appear).

---

`pen.matrix` *Calculating the penalty matrix P*

---

## Description

Calculating the penalty matrix P depends on the used basis function and the selected kind of penalty.

## Usage

```
pen.matrix(penden.env)
```

## Arguments

| | |
|---|---|
| `penden.env` | Containing all information, environment of paircopula(). |

## Details

If paircopula is selected with 'pen=1', the differences of order 'm' are penalized. Only for Bernstein polynomials: If paircopula is selected with 'pen=2', the integrated squared second order derivatives are used as penalty.

## Value

| | |
|---|---|
| `DDD.sum` | Penalty matrix P |

Matrix is saved in the environment.

## Author(s)

Christian Schellhase <cschellhase@wiwi.uni-bielefeld.de>

## References

Flexible Pair-Copula Estimation in D-vines using Bivariate Penalized Splines, Kauermann G. and Schellhase C. (2014+), Statistics and Computing (to appear).

plot.paircopula *Flexible Pair-Copula Estimation in D-vines with Penalized Splines*

## Description

Plotting a paircopula of class 'paircopula'.

## Usage

```
## S3 method for class 'paircopula'
plot(x,val=NULL,marg=TRUE,plot.view=TRUE,int=FALSE,main.txt=NULL,
                    sub.txt=NULL,contour=FALSE,cuts=20,cex=1,cex.axes=1,
                   xlab=NULL,ylab=NULL,zlab=NULL,xlim=NULL,ylim=NULL,zlim=NULL,
                    show.observ=FALSE,margin.normal=FALSE,...)
```

## Arguments

| | |
|---|---|
| x | object of class 'paircopula', result of function 'paircopula'. |
| val | Default val = NULL, one can calculate the estimated density/distribution for bivariate vector, e.g. val=c(0.5,1). |
| marg | Default = TRUE, plotting the marginal densities. |
| plot.view | Default = TRUE, if 'FALSE' no plot is shown, e.g. for calculations with val != NULL. |
| int | Default = FALSE, if TRUE, the integral, i.e. the distribution of the copula density is plotted. |
| main.txt | Default = NULL shows 'K' and the value of lambda. |
| sub.txt | Default = NULL shows the log-likelihood, the penalized log-likelihood and the cAIC-value of the estimation. |
| contour | If TRUE, a contour plot is shown. Default = FALSE. |
| cuts | Number of cuts for the contour plots, if contour=TRUE. |
| cex | Default = 1, determing the size of the main of the plot. |
| cex.axes | Default = 1, determing the size of the labels at the axes. |
| xlab | Default = NULL and no text is printed at the xlab |
| ylab | Default = NULL and no text is printed at the ylab |
| zlab | Default = NULL and 'density' is printed at the zlab for int=FALSE and 'distribution' for int=TRUE. |
| xlim | Default = NULL, changes the range for the values of x in the case of a contour plot. |
| ylim | Default = NULL, changes the range for the values of y in the case of a contour plot. |
| zlim | Default = NULL and the range for the values of z are the range of calculated values. |

| | |
|---|---|
| show.observ | Default = FALSE. If TRUE, plotting the original observation into a contourplot. |
| margin.normal | Default = FALSE. If TRUE, the plot is presented with margins following standard normal distribution. |
| ... | further arguments |

## Value

If 'val' is not NULL, the function returns a matrix with the calculated density or distribution values for the set 'val'.

## Author(s)

Christian Schellhase <cschellhase@wiwi.uni-bielefeld.de>

## References

Flexible Pair-Copula Estimation in D-vines using Bivariate Penalized Splines, Kauermann G. and Schellhase C. (2014+), Statistics and Computing (to appear).

---

# sim.Dvine *Flexible Pair-Copula Estimation in D-vines with Penalized Splines*

---

## Description

Simulating a p-dimensional vector of the fitted p-dimensional D-vine.

## Usage

```
sim.Dvine(DV,N=NULL)
```

## Arguments

| | |
|---|---|
| DV | object of class 'penDvine', result of 'Dvine'. |
| N | Number of simulated p-dimensional pairs from the fitted D-vine 'DV'. |

## Value

The returing values are simulated values of the current Dvine.

## Author(s)

Christian Schellhase <cschellhase@wiwi.uni-bielefeld.de>

## References

Flexible Pair-Copula Estimation in D-vines using Bivariate Penalized Splines, Kauermann G. and Schellhase C. (2014+), Statistics and Computing (to appear).

`Winddata` *Maximal daily windspeed in khm.*

## Description

Maximal daily windspeed in khm at Frankfurt, Berlin, Bremen, Munich and Cuxhaven, observed in the time from 01/01/2000 until 31/12/2011.

## Usage

```
data(Winddata)
```

## Format

A data frame with 507 observations of the following 2 variables.

`Date` Date

`Frankfurt` Observations in Frankfurt

`Berlin` Observations in Berlin

`Bremen` Observations in Bremen

`Munich` Observations in Munich

`Cuxhaven` Observations in Cuxhaven

## Note

The data is available at the internet page of the 'Deutsche Wetterdienst' DWD, www.dwd.de. 'Winddata-original' contains the original data, 'Winddata' contains marginal fitted t-distributions of the data.

# Index