Package ‘pencopula’

February 26, 2014

Type Package

Title Flexible Copula Density Estimation with Penalized Hierarchical B-Splines

Version 0.3.5

Date 2014-02-17

Depends R (>= 2.15.1), lattice, fda, quadprog, latticeExtra

Suggests MASS

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

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

Description Flexible copula density estimation with penalized hierarchical B-Splines.

License GPL (>= 2)

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2014-02-26 15:11:20

R topics documented:

  pencopula-package ........................................ 2
  bernstein ............................................... 3
  Derv1 ..................................................... 4
  Derv2 ..................................................... 5
  DeutscheBank ............................................. 6
  distr.func.help ......................................... 6
  f.hat.val ............................................... 7
  hierarch.bs ............................................ 8
  knots.start ............................................. 9
  Lufthansa ............................................... 10
The package 'pencopula' offers routines for estimating multivariate penalized copula densities and copula distribution.

Description

The package 'pencopula' offers routines for estimating multivariate penalized copula densities and copula distribution. For details see the description of the function pencopula().

Details

Package: pencopula
Type: Package
Version: 0.3.5
Date: 2014-02-17
License: GPL (>= 2) LazyLoad: yes

The packages contributes the function 'pencopula()' for estimating copula densities and copula distributions using penalized splines techniques.

Author(s)

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

References


Examples

Y <- cbind(runif(200), runif(200))
Calculating a bernstein polynomial.

Description
Calculating a bernstein polynomial, transformed to be a density.  `int.bernstein` calculates the integral of the bernstein density polynomial.

Usage

bernstein(v, x, n)
int.bernstein(x, n)

Arguments

- `v`  number of the current basis element, e.g. v is in the interval [0,2^K]
- `x`  data points for constructing of the bernstein polynomial basis.
- `n`  number of polynomials in the bernstein polynomial basis, e.g n = 2^K

Value

The bernstein polynomial density basis is created using the function 'apply' for some data 'x'.

\[ K \leftarrow 3 \]
\[ \text{index.b} \leftarrow \text{matrix}(0:2^K) \]  ## Bernstein polynomial B \leftarrow \text{apply}($\text{index.b},1,\text{bernstein,x,n=2^K}$)

The integral of the Basis B of degree n is: B \leftarrow \text{int.bernstein}(x,n=2^K)

Author(s)

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

References

Calculating the first derivative of the pencopula likelihood function w.r.t. parameter b

Description

Calculating the first derivative of the pencopula likelihood function w.r.t. parameter b.

Usage

Derv1(penden.env)

Arguments

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

Details

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

\[ s(b, \lambda) = \frac{\partial l(b, \lambda)}{\partial b} = \sum_{i=1}^{n} \frac{\Phi(u_i)}{c(u_i, b)} - P(\lambda)b \]

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 b. Derv1.pen is saved in the environment.

Author(s)

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

References

Calculating the second order derivative with and without penalty.

Description
Calculating the second order derivative of the likelihood function of the pencopula approach w.r.t. the parameter b. Thereby, for later use, the program calculates the second order derivative with and without the penalty. Moreover, Derv2 separates the calculation for temporary weights b in iteration and final weights b.

Usage
Derv2(penden.env, temp = FALSE)

Arguments
penden.env  Containing all information, environment of pendensity()
temp  smoothing parameter 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. beta with penalty
Derv2.cal  second order derivative w.r.t. beta 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
DeutscheBank

Daily final prices (DAX) of the German stock Deutsche Bank in the years 2006 and 2007

Description

Containing the daily final prices of the German stock Deutsche Bank in the years 2006 and 2007.

Usage

data(DeutscheBank)

Format

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

Date  Date
ClosingPrice  ClosingPrice

distr.func.help

These functions are used for calculating the integral of the B-spline density basis.

Description

These functions cooperate with each other for calculating the integral of the B-spline density basis. The functions 'distr.func.help' is the main program, calling 'poly.part', calculating the integral of the B-spline density basis in sections between neighbouring knots. 'distr.func.help' calculates analytical functions of the integral. Therefore the function 'poly.part' is needed to construct these functions.

Usage

distr.func.help(base, knots, penden.env, q, y, index)
poly.part(i, j, knots, help.env, q, yi=NULL, poly=FALSE)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>base</td>
<td>values of the considered B-spline basis</td>
</tr>
<tr>
<td>knots</td>
<td>knots of the considered B-spline basis</td>
</tr>
<tr>
<td>penden.env</td>
<td>Containing all information, environment of pencopula()</td>
</tr>
<tr>
<td>q</td>
<td>degree of the B-Spline basis</td>
</tr>
<tr>
<td>y</td>
<td>data of the marginal B-spline basis</td>
</tr>
<tr>
<td>index</td>
<td>columns of the whole B-spline basis, each hierarchy level is integrated separately</td>
</tr>
</tbody>
</table>
Calculating the actual fitted values ‘f.hat.val’ of the estimated density function

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

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

Arguments

penden.env Containing all information, environment of pencopula()
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 b. Multiplying the actual set of parameters b with the base ‘base.den’ delivers the fitted values.
Value

- \texttt{f.hat.val} Fitted values for the current coefficients
- \texttt{.f.hat.val} is saved in the environment.

Author(s)

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

References


hierarch.bs \hspace{1em} Construction of the hierarchical B-spline density basis.

Description

Construction of the hierarchical B-spline density basis.

Usage

\texttt{hierarch.bs(x, d, plot.bsp, typ, penden.env, int=FALSE)}

Arguments

- \texttt{x} Marginal data for construction.
- \texttt{d} Hierarchy level of the marginal hierarchical B-spline density.
- \texttt{plot.bsp} Default = FALSE. If TRUE, each B-spline basis is plotted.
- \texttt{typ} typ==1 without open B-splines at the boundary typ==2 with open B-splines at the boundary.
- \texttt{penden.env} Containing all information, environment of pencopula().
- \texttt{int} Default = FALSE. If TRUE, the integral of the hierarchical B-spline density basis is calculated (used for the distribution function of the estimation).

Details

First, the marginal hierarchical B-spline density basis is constructed for each covariate ‘B.tilde’. After the construction of each marginal basis, the hierarchical B-spline density basis is built in the main program pencopula(), using an object 'Index.basis.D' (saved in the environment). 'Index.basis.D' notes which component of the marginal basis has to be selected. In the main program the object 'tilde.Psi.d.D' is constructed. D refers to the maximum hierarchy level and 'd' is the hierarchy level of the marginal hierarchical B-spline.
**knots.start**

**Value**

- **B.tilde**  
  'B.tilde' is the hierarchical B-spline density basis, returned by 'hierarch.bs'.

- **int.B.tilde**  
  If 'int=TRUE', the integral of the hierarchical B-spline density basis is calculated and returned by 'hierarch.bs'.

**Author(s)**

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

**References**


---

| knots.start | Calculating the knots. |

**Description**

Calculating the equidistant knots for the estimation. Moreover, transformation of the knots are possible.

**Usage**

```r
knots.start(penden.env)  
knots.transform(d, alpha = 0, symmetric = TRUE)  
knots.order(penden.env)
```

**Arguments**

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

- **d**  
  Hierarchy level of the marginal hierarchical B-spline basis.

- **alpha**  
  Default = 0. Alpha is a tuning parameter, shifting the knots.

- **symmetric**  
  Default = TRUE. If FALSE, the knots are selected without symmetry.

**Details**

'Knots.order' sorts the knots in the order, in which they disappear in the hierarchical B-spline basis.
Value

knots  Selected and sorted marginal knots for the estimation.
knots.help  Extended set of knots. It is needed for calculating the distribution function, help points for the integration of the B-spline density basis.
k.order  Order of the knots, corresponding to their order in the hierarchical B-spline density basis.
knots.t  The knots ordered with 'k.order' for further functions.
tilde.Psi.knots.d  Hierarchical B-Spline density basis for 'knots'.
tilde.Psi.knots.d.help  Hierarchical B-Spline density basis for 'knots.help'.

All values are saved in the environment.

Author(s)

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

References


---

Lufthansa  Daily final prices (DAX) of the German stock Lufthansa in the years 2006 and 2007

Description

Containing the daily final prices of the German stock Lufthansa in the years 2006 and 2007.

Usage

data(Lufthansa)

Format

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

Date  Date
ClosingPrice  ClosingPrice
**Description**

'my.bspline' Integrates the normal B-Spline basis to a B-spline density basis. The dimension of the basis depends on the input of number of knots 'k' and of the order of the B-spline basis 'q'. 'int.my.bspline' is a function for transformation of open B-spline basis at the boundary to become a B-spline basis density.

**Usage**

```r
my.bspline(h, q, knots, y, K, plot.bsp, typ)
int.my.bspline(help.env)
```

**Arguments**

- **h**: if equidistant knots are used (default in pencopula()), h is the distance between two neighbouring knots
- **q**: selected order of the B-spline basis
- **knots**: selected values for the knots
- **y**: values of the response variable
- **K**: the number of knots for the construction of the base
- **plot.bsp**: Indicator variable TRUE/FALSE if the integrated B-spline basis should be plotted
- **typ**: typ==1 without open B-splines at the boundary typ==2 with open B-splines at the boundary
- **help.env**: Internal environment of my.bspline().

**Details**

Firstly, the function constructs the B-spline basis to the given number of knots 'K' and the given locations of the knots. Due to the recursive construction of the B-Spline, for all orders greater than 2, the dimension of the B-spline basis of given K grows up with help.degree=q-2. There are two typs of B-spline basis possible. First, a B-spline basis without open B-splines at the boundary (typ==1) and a regulat B-spline basis with open B-splines at the boundary (typ==2). Both typs are integrated to become B-spline density basis. To integrate a basis of typ 1 we use the well-known factor 'q/(knots.val[i+q]-knots.val[i])'. For typ==2 we determine functions analytically for the integration. Moreover, one can draw the integrated basis and, if one calls this function with the argument 'plot.bsp=TRUE'.

Value

- **base.den**: The integrated B-Spline base of order q
- **stand.num**: The coefficients for standardization of the ordinary B-Spline basis
- **knots.val**: This return is a list. It considers the used knots 'knots.val$val', the help knots 'knots.val$help' and the additional knots 'knots.val$all', used for the construction of the base and the calculation of the distribution function of each B-Spline.
- **K**: The transformed value of K, due to used order 'q' and the input of 'K'

Note

This function uses the fda-package to build the B-Spline density basis.

Author(s)

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

References


---

**my.IC**

---

Calculating the AIC- and BIC-value

Description

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

Usage

```r
my.IC(penden.env)
```

Arguments

- **penden.env**: Containing all information, environment of pencopula()

Details

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

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

AIC  sum of twice the negative non-penalized log likelihood and mytrace
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


Description

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

Usage

my.loop(penden.env)

Arguments

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

Details

'my.loop' optimates the log-likelihood iteratively. Therefore, the routine checks the relative chance in the weights and stops the iteration, if the relative change of all weights 'b' is less than one percent. During the calculations of new weights 'b' in the routine 'new.weights', most of the values are called '.temp'. This add on unterlines the temporarily values. 'my.loop' checks the relative change in the weights. If the change is 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 'b'.

Iterative loop for calculating the optimal coefficients 'b'.

my.loop
my.positive.definite.solve

Description
Reverses a quadratic positive definite matrix.

Usage
my.positive.definite.solve(A, eps = 1e-15)

Arguments
A
quadratic positive definite matrix
eps
level of the lowest eigenvalue to consider

Details
The program makes an eigenvalue decomposition of the positive definite matrix A and searches all eigenvalues greater than eps. The value of return is the inverse matrix of A, constructed with the matrix product of the corresponding eigenvalues and eigenvectors.

Value
The return is the inverse matrix of A.

Author(s)
Christian Schellhase <cschellhase@wiwi.uni-bielefeld.de>

References
Calculating new weights $b$.

Description

Calculating new weights $b$ using quadratic programing.

Usage

new.weights(penden.env)

Arguments

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

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 $c(u,b)\geq 0$, $\text{sum}(b)=1$ and that the marginal densities are uniform. All side conditions are saved as 'AA.help' in the environment. There exists two different algorithms, due to the fact if one works with or without an adaptive grid for the side condition $c(u,b)\geq 0$.

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'.

Using a Bernstein polynomial basis, the side conditions are $\text{sum}(b)=1$, $b\geq 0$ and that the marginal densities are uniform.

Value

ck.val.temp Calculated new values for the weights 'b'. The add on 'temp' means, that there is a check in the next step if the weights 'b' have been converted or not. 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.

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

Note

For high dimensional data (p\geq 4) and high hierachy level 'd' of the marginal hierarchical B-spline, the computational time for solving a quadratic program increases.

Author(s)

Christian Schellhase <cschellhase@wiwi.uni-bielefeld.de>
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 pencopula()
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 \( b \), the penalized hierarchical B-splines \( \Phi \) and the penalty parameters \( \lambda \).

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

with

\[
P(\lambda) = \sum_{j=1}^{p} p\lambda_j P_j
\]

The needed values are saved in the environment.

Value
pen.log.like Penalized log likelihood of the copula density.
log.lik Log-Likelihood of the copula density.

The values are saved in the environment.

Author(s)
Christian Schellhase <cschellhase@wiwi.uni-bielefeld.de>

References
penalty.matrix

References


penalty.matrix  Calculating the penalty matrix \( P(\lambda) \)

Description

Calculating the penalty matrix \( P \) depends on the number of covariates \( p \), the order of differences to be penalized \( \text{pen.order} \), the number of observations \( n \) and the penalty parameters \( \lambda \).

Usage

penalty.matrix(penden.env, temp = FALSE)

Arguments

penden.env  Containing all information, environment of pencopula().
temp  If TRUE, the iteration for a new \( b \) is not finished and a temporary penalty matrix is calculated, default = FALSE.

Details

The penalty matrix is calculated as
\[
P(\lambda) = \sum_{j=1}^{p} \lambda_j P_j
\]
with
\[
P_j = \left( \bigotimes_{l=1}^{j-1} I \right) \otimes \{(A^{-1})^T P A^{-1}\} \otimes \left( \bigotimes_{l=j+1}^{p} I \right)
\]
The needed values are calculated or saved in the environment \( \text{penden.env} \).
There is no penalty used for the Bernstein polynomial basis, \( P=0 \).

Value

\text{DDD.sum}  Penalty matrix \( P \)
Matrix is saved in the environment.

Author(s)

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

References

Calculating penalized copula density with penalized hierarchical B-splines

Description

Main program for estimation of copula densities with penalized hierarchical B-splines. The estimation can be done for multivariate datasets. The response is called 'y', the covariates 'x'. We estimate densities using penalized hierarchical B-splines. This is done by choosing the polynomial degree of the univariate B-spline density basis, that is built upon 2^d+1 equidistant knots. 'd' refers to the hierarchy level of the marginal hierarchical B-spline and 'D' to the maximum hierarchy level of the hierarchical B-spline basis. We penalize the m-order differences of the coefficients 'b' to estimate new weights 'b'.

Usage

pencopula(data, d=3, D=d, q=1, base = "B-spline", max.iter = 20, plot.bsp = FALSE, lambda = NULL, pen.order=2, adapt.grid = FALSE, add = TRUE, alpha = 0, symmetric = TRUE, data.frame=parent.frame())

Arguments

data 'data' contains the data. 'data' has to be a matrix or a data.frame. The number of columns of 'data' is p.
d refers to the hierarchy level of the marginal hierarchical B-spline, default is d=3.
D referes to the maximum hierarchy level, default is D=3. If D<d, it follows D<-d.
q degree of the marginal hierarchical B-spline.
base By default, the used marginal basis is a 'B-spline'. Second possible option is 'Bernstein', using a Bernstein polynomial basis.
max.iter maximum number of iteration, the default is max.iter=20.
plot.bsp TRUE or FALSE, if TRUE the used B-Spline base should be plotted.
lambda p-dimensional vector of penalty parameters, the values can be different. Default is rep(10000,p).
pen.order The order of differences for the penalization, default is pen.order=2.
adapt.grid Default = FALSE, if TRUE the used grid for the side condition c(u,b)>=0 is reduced.
add Default = TRUE. Due to numerical rounding errors, some results in the quadratic programming are misleading. So we add a small epsilon in the side conditions c(u,b)>=0 to the actual weights 'b' in each iteration.
alpha Default = 0, that results in equidistant knots. If alpha !=0 the knots are moved.
symmetric Default = TRUE. Tuning parameter for the location of the knots if alpha != 0.
data.frame reference to the data. Default reference is the parent.frame().
Details

The estimation of the copula density is done in several steps. 1. Preparations: Calculating the number of marginal knots 'ddb' and the number of spline coefficients 'DD', depending on 'd' and 'D'. 'Index.Basis.D' is created, that is an index set, which marginal basis will belong to the hierarchical B-spline density basis 'tilde.Psi.d.D'. 2. Creating marginal spline coefficients 'T.marg', depending on 'Index.basis.D'. 3. Building the hierarchical B-spline density basis 'tilde.PSI.d.D'. 4. Calculating the knots for the estimation, by default the knots are equidistant. Transformations of the knots are possible, see 'knots.start'. Moreover a B-spline density basis 'tilde.Psi.d.knots' is created, with the knots in the hierarchical order 'knots.t'. 'tilde.Psi.d.knots' is needed for the restriction, that each marginal density has to be uniform. 5. Depending on the knots, start values for the weights 'b' are calculated. This is done in 'startval.grid'. Moreover 'startval.grid' determines an grid for the side condition c(u,b)>=0 of the quadratic program. 6. Each marginal basis is estimated to be uniform. Therefore a matrix 'A.Restrict' is calculated, such that

\[ A.Restrict \% \% b = 1 \]

for all p-covariates. 7. The penalty matrix P is created, see 'penalty.matrix'. 8. The first calculation of coefficients 'b' is done. 9. 'my.loop' iterates the calculation of the optimal weights 'b' until some convergence, see 'my.loop'. The maximal number of iterations are limited, default is max.iter=20. 10. If 'my.loop' terminates, the information criteria are calculated, see 'my.IC'.

Value

Returning an object of class pencopula. The class pencopula consists of the environment 'pen-den.env', which includes all calculated values of the estimation approach. For a fast overview of the main results, one can use the function 'print.pencopula()'.

Note

If the estimation with the current setup does not work, e.g. the quadratic program can not be solved, the routine returns log.like=0, pen.log.like=0, AIC=0 and BIC=0. Please restart the estimation, e.g. with a different penalty parameter.

Author(s)

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

References


Examples

```r
require(MASS)
data(Lufthansa)
data(DeutscheBank)
data.dbank <- data.lufth <- c()
```
**pendenForm**

Formula interpretation and data transfer

**Description**

Function 'pendenForm' interprets the input 'form' of the function pencopula(), transfers the data back to the main program.

**Usage**

```r
pendenForm(penden.env)
```

**Arguments**

penden.env  
environment used in pendensity()

**Value**

Returning the values 'Y', the number of values 'n' and covariates 'p'.

**Author(s)**

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

**References**

plot.pencopula  

Plot the estimated copula density or copula distribution.

Description

The function plots the estimated copula density or the copula distribution, using the R-package 'lattice'.

Usage

```r
## S3 method for class 'pencopula'
plot(x, val = NULL, marg = TRUE, plot = TRUE, int = FALSE,
     mainNtxt = NULL, subNtxt = NULL, contour = FALSE, cond = NULL, cuts = 20,
     cexNaxes = 1, cexNcontour = 1, xlab = NULL, ylab = NULL,
     zlab = NULL, bivNmargin = NULL, showNobserv = FALSE,...)
```

Arguments

- `x`: object of class 'pencopula'.
- `val`: Default val = NULL, one can calculate the estimated density in for p-dimensional vector, e.g. val=c(0.5,1) for the two dimensional case.
- `marg`: Default = TRUE, plotting the marginal densities.
- `plot`: 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.
- `mainNtxt`: Default = NULL shows 'd', 'D', the values of lambda, the penalty order and the degree of the B-splines.
- `subNtxt`: Default = NULL shows the log-likelihood, the penalized log-likelihood and the AIC-value of the estimation.
- `contour`: If TRUE, a contour plot is shown. Default = FALSE.
- `cond`: Default = NULL, if the dimension of data 'p' is higher than 2, one can plot a two-dimensional conditional plot. The user specifies p-2 values for the plot, indicating with '-1'. So for a three-dimensional plot, cond=c(0,-1,-1) shows the density/distribution ith fixed first covariate and the second and third covariates vary.
- `cuts`: Number of cuts for the contour plots, if contour=TRUE.
- `cex`: Default = 1, determing the size of the main of the plot.
- `cexNaxes`: Default = 1, determing the size of the labels at the axes.
- `cexNcontour`: Default = 1, determing the size of the labels at the cuts of the contourplot.
- `xlab`: Default = NULL and no text is printed at the xlab
- `ylab`: Default = NULL and no text is printed at the ylab
print.pencopula

print.pencopula
Printing the main results of the penalized copula density estimation

Description

Printing the call of the estimation, the used basis, lambda and the corresponding values of AIC and BIC. Need an object of class pencopula.

Usage

## S3 method for class 'pencopula'
print(x, ...)

Arguments

x         x has to be object of class pencopula

...      further arguments
Calculating the start values 'b' for the first iteration of the quadratic program.

Description
Calculating the start values 'b' for the first iteration of the quadratic program. Moreover, the grid of values for the side condition c>=0 of the quadratic program are calculated. If "adapt.grid", the number of grid points is reduced to speed up the quadratic program.

Usage
start.valgrid(penden.env)

Arguments
penden.env Containing all information, environment of pencopula()

Details
The grid of values for the side conditions of the quadratic program c>=0 is constructed as the tensor product of all knots. If $p$ and $d$ increase, the number of conditions and computational time of the quadratic program increase enormously, e.g. a full tensor product $u_j$ for $p=4$ and $d=4$ contains 83521 entries. If the data $u_j$ is not high correlated, i.e. the data is not from a extreme value copula like a Clayton copula, one can reduce the full tensor product. In 'pencopula' one can choose the option 'adapt.grid' which effects the following and may reduce the calculating time without any loss of accuracy. One can omit points in $u_j$ in sections of $[0,1]^p$ which are in the neighbourhood of many observations in $u_j$, because the data itself induces a positive density in these areas by construction. Therefore, we calculate the minimal $p$-dimensional euclidean distance $e_i$ of each $u_i, i=1,\ldots,(2^{d+1})^p$ to the data $u_j$ and omit the points corresponding to the first quartile of minimal euclidean distance $e_i$ in $u_j$, we call this new set of points $u_{min}$. This amount of points is used in the first iteration step to estimate weights $b_j$ corresponding to a copula density.
Value

X.knots.g  
If adapt.grid=TRUE, set of reduced grid values, in which the side condition of the quadratic program \( c(u,b) \geq 0 \) will be postulated.

X.knots.g.all  
Set of all grid values, in which the side condition of the quadratic program \( c(u,b) \geq 0 \) will be postulated.

The values are saved in the environment.

Author(s)

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

References

Index

+ Topic **IO**
  - pendenform, 20
+ Topic **algebra**
  - my.positive.definite.solve, 14
+ Topic **datasets**
  - DeutscheBank, 6
  - Lufthansa, 10
+ Topic **math**
  - bernstein, 3
  - Derv1, 4
  - Derv2, 5
  - distr.func.help, 6
  - my.bspline, 11
  - my.IC, 12
  - my.loop, 13
+ Topic **nonparametric**
  - f.hat.val, 7
  - hierarch.bs, 8
  - new.weights, 15
  - pen.log.like, 16
  - penalty.matrix, 17
  - pencopula, 18
  - pencopula-package, 2
  - start.valgrid, 23
+ Topic **plot**
  - plot.pencopula, 21
+ Topic **print**
  - print.pencopula, 22

bernstein, 3
Derv1, 4
Derv2, 5
DeutscheBank, 6
distr.func.help, 6
f.hat.val, 7
grid.points(start.valgrid), 23
hierarch.bs, 8

int.bernstein (bernstein), 3
int.my.bspline (my.bspline), 11
knots.order (knots.start), 9
knots.start, 9
knots.transform(knots.start), 9
Lufthansa, 10
my.bspline, 11
my.IC, 12
my.loop, 13
my.positive.definite.solve, 14
new.weights, 15
pen.log.like, 16
penalty.matrix, 17
pencopula, 18
pencopula-package, 2
pen denform, 20
plot.pencopula, 21
poly.part(distr.func.help), 6
print (print.pencopula), 22
print.pencopula, 22
start.valgrid, 23