Package ‘regpro’

January 31, 2016

Version 0.1.1

Date 2016-01-29

Title Nonparametric Regression

Author Jussi Klemela <jussi.klemela@gmail.com>

Maintainer Jussi Klemela <jussi.klemela@gmail.com>

Depends denpro (>= 0.9.0)

Description Tools are provided for
(1) nonparametric regression (kernel, local linear),
(2) semiparametric regression (single index, additive models), and
(3) quantile regression (linear, kernel).

License GPL (>= 2)

URL http://jklm.fi/regpro

NeedsCompilation no

Repository CRAN

Date/Publication 2016-01-31 12:32:20

R topics documented:

`additive` *An additive model regression estimator for pointwise estimation*

Description

Computes the value of a regression function estimator at one point, when the estimator is based on the additive model.

Usage

```
additive(x, y, arg=NULL, eval=NULL, h=1, kernel="gauss", M=2, vect=FALSE)
```

Arguments

`x`	n*d data matrix; the matrix of the values of the explanatory variables
`y`	n vector; the values of the response variable
`arg`	d-vector; the point where the estimate is evaluated
`eval`	either NULL or a n*d matrix; the matrix that gives the evaluations of the coordinate functions at the data points
`h`	a positive real number; the smoothing parameter of the kernel estimate
`kernel`	a character; determines the kernel function; either "gauss" or "uniform"
`M`	integer >=2; the number of iterations
`vect`	TRUE or FALSE; internal parameter

Value

list of eval, value, and valvec; "eval" is a n*d matrix of the evaluations of the estimated component functions at the data points; "value" is a real number giving the estimated value of the regression function at one point; "valvec" is d vector giving the estimated values of the component functions at one point

Author(s)

Jussi Klemela

See Also

pcf.additive,

Examples

```
set.seed(1)
n<-100
d<-2
x<-8*matrix(runif(n*d),n,d)-3
C<-(2*pi)^(-d/2)
phi<-function(x){ return( C*exp(-sum(x^2)/2) ) }
D<-3; c1<-c(0,0); c2<-D*c(1,0); c3<-D*c(1/2,sqrt(3)/2)
func<-function(x){phi(x-c1)+phi(x-c2)+phi(x-c3)}
y<-matrix(0,n,1)
for (i in 1:n) y[i]<-func(x[i,])+0.01*rnorm(1)

additive(x,y)

arg<-c(0,0)
additive(x,y,arg=arg)
```

---

additive.stage — *Stagewise fitting of an additive model*

Description

Computes the value of an additive model regression estimator at one point using stagewise fitting

Usage

```
additive.stage(x, y=NULL, arg=NULL, residu=NULL, deet=NULL, h=1, kernel="gauss",
M=2, vect=FALSE)
```

Arguments

x	n*d data matrix; the matrix of the values of the explanatory variables
y	n vector or NULL; the values of the response variable; if "residu" is given, then "y" is not used
arg	d-vector; the point where the estimate is evaluated
residu	NULL or n*M matrix of residuals; for each step the n vector of residuals is given; the first residual is the vector of y-observations
deet	NULL or M vector of values 1,...,d; for each step the direction chosen by the optimizer
h	a positive real number; the smoothing parameter of the kernel estimate
kernel	a character; determines the kernel function; either "gauss" or "uniform"
M	integer >=2; the number of iterations
vect	TRUE or FALSE; internal parameter

Value

list of eval, residu, deet, value, and valvec; "eval" is a n*d matrix of the evaluations of the estimated component functions at the data points; "residu" is n*M matrix which contains the sequence of estimates evaluated at the observations; "deet" is M vector of values 1,...,d which indicates for each step the direction chose by the optimization procedure; "value" is a real number giving the estimated value of the regression function at one point; "valvec" is d vector giving the estimated values of the component functions at one point

Author(s)

Jussi Klemela

See Also

pcf.additive,

Examples

```
set.seed(1)
n<-100
d<-2
x<-8*matrix(runif(n*d),n,d)-3
C<-(2*pi)^(-d/2)
phi<-function(x){ return( C*exp(-sum(x^2)/2) ) }
D<-3; c1<-c(0,0); c2<-D*c(1,0); c3<-D*c(1/2,sqrt(3)/2)
func<-function(x){phi(x-c1)+phi(x-c2)+phi(x-c3)}
y<-matrix(0,n,1)
for (i in 1:n) y[i]<-func(x[i,])+0.01*rnorm(1)

as<-additive.stage(x,y)

arg<-c(0,0)
additive.stage(x,arg=arg,residu=as$residu,deet=as$deet)
```

---

copula.trans *Makes a copula transformation*

Description

Transforms a data matrix so that the marginals follow approximately the standard Gaussian distribution. Alternatively, the marginals can be transformed to follow approximately the uniform distribution on [0,1].

Usage

```
copula.trans(dendat, marginal=rep("gauss",dim(dendat)[2]), remna=TRUE)
```

Arguments

`dendat`	n*d data matrix; the data matrix of n observations and d variables
`marginal`	d-vector of character strings; the character strings can be "gauss" or "uniform"
`remna`	TRUE or FALSE; if remna=TRUE, then the rows containing a NA are removed

Value

n*d data matrix

Author(s)

Jussi Klemela

Examples

```
set.seed(2)
n<-100
d<-2
dendat<-matrix(runif(n*d),n,d)
x<-copula.trans(dendat)
```

`emp.distribu` *Empirical distribution function at one point*

Description

Computes the value of an empirical distribution function at one point.

Usage

```
emp.distribu(arg, dendat)
```

Arguments

`arg`	d-vector; the point where the estimate is evaluated
`dendat`	n*d data matrix; the data matrix of n observations and d variables

Value

a real number or a d vector; if d>1 the empirical distribution function is estimated for each column of the data matrix "dendat"

Author(s)

Jussi Klemela

See Also

emp.quantile,

Examples

```
set.seed(2)
n<-100
d<-2
x<-matrix(runif(n*d),n,d)

arg<-c(0,0)
emp.distribu(arg,x)
```

---

emp.quantile *Empirical quantile function at one point*

Description

Computes the value of an empirical quantile function at one point.

Usage

```
emp.quantile(arg, dendat)
```

Arguments

arg	d-vector; the point where the estimate is evaluated
dendat	n*d data matrix; the data matrix of n observations and d variables

Value

a real number or a d vector; if d>1 the empirical quantile function is estimated for each column of the data matrix "dendat"

Author(s)

Jussi Klemela

See Also

emp.distribu,

Examples

```
set.seed(2)
n<-100
d<-2
x<-matrix(runif(n*d),n,d)

arg<-c(0.5,0.5)
emp.quantile(arg,x)
```

---

kernesti.der — *An estimator of a partial derivative of a regression function at one point*

Description

Computes the value of a multivariate kernel estimator of a partial derivative of a regression function at a one point.

Usage

```
kernesti.der(arg, x, y, h=1, direc=1, kernel="gauss", vect=FALSE)
```

Arguments

arg	d-vector; the point where the estimate is evaluated
x	n*d data matrix; the matrix of the values of the explanatory variables
y	n vector; the values of the response variable
h	a positive real number; the smoothing parameter of the kernel estimate
direc	integer 1,...,d; indicates which partial derivative is estimated
kernel	a character; determines the kernel function; can only be "gauss"
vect	TRUE or FALSE; an internal parameter related to the method of calculation

Value

a real number

Author(s)

Jussi Klemela

See Also

pcf.kernesti.der,

Examples

```
set.seed(1)
n<-100
d<-2
x<-8*matrix(runif(n*d),n,d)-3
C<-(2*pi)^(-d/2)
phi<-function(x){ return( C*exp(-sum(x^2)/2) ) }
D<-3; c1<-c(0,0); c2<-D*c(1,0); c3<-D*c(1/2,sqrt(3)/2)
func<-function(x){phi(x-c1)+phi(x-c2)+phi(x-c3)}
y<-matrix(0,n,1)
for (i in 1:n) y[i]<-func(x[i,])+0.01*rnorm(1)

arg<-c(0,0)
kernesti.der(arg,x,y,h=0.5)
```

---

kernesti.quantile *Multivariate kernel conditional quantile estimator at one point*

Description

Computes the value of a multivariate kernel conditional quantile estimator at one point.

Usage

```
kernesti.quantile(arg, x, y, h=1, p=0.5, kernel="gauss")
```

Arguments

arg	d-vector; the point where the estimate is evaluated
x	n*d data matrix; the matrix of the values of the explanatory variables
y	n vector; the values of the response variable
h	a positive real number; the smoothing parameter of the kernel estimate
p	0<p<1; the p:th quantile function will be estimated
kernel	a character; determines the kernel function; either "gauss" or "uniform"; in the univariate case can also be "exp"

Value

a real number

Author(s)

Jussi Klemela

See Also

pcf.kernesti,

Examples

```
set.seed(1)
n<-100
d<-2
x<-8*matrix(runif(n*d),n,d)-3
C<-(2*pi)^(-d/2)
phi<-function(x){ return( C*exp(-sum(x^2)/2) ) }
D<-3; c1<-c(0,0); c2<-D*c(1,0); c3<-D*c(1/2,sqrt(3)/2)
func<-function(x){phi(x-c1)+phi(x-c2)+phi(x-c3)}
y<-matrix(0,n,1)
for (i in 1:n) y[i]<-func(x[i,])+0.01*rnorm(1)

arg<-c(0,0)
kernesti.quantile(arg,x,y,h=0.5)
```

---

kernesti.regr — *Multivariate kernel regression estimator at one point*

Description

Computes the value of a multivariate kernel regression estimator (Nadaraya-Watson estimator) at one point.

Usage

```
kernesti.regr(arg, x, y, h=1, kernel="gauss", g=NULL, gernel="gauss", vect=FALSE)
```

Arguments

`arg`	d-vector; the point where the estimate is evaluated
`x`	n*d data matrix; the matrix of the values of the explanatory variables
`y`	n vector; the values of the response variable
`h`	a positive real number; the smoothing parameter of the kernel estimate
`kernel`	a character; determines the kernel function; either "gauss" or "uniform"; in the univariate case can also be "exp"
`g`	a positive real number; the smoothing parameter of the kernel estimate for a simultaneous time space smoothing
`gernel`	a character; determines the kernel function for the time space smoothing; either "gauss", "uniform", "exp", or "bart"
`vect`	TRUE or FALSE; an internal parameter related to the method of calculation

Value

a real number

Author(s)

Jussi Klemela

See Also

pcf.kernesti,

Examples

```
set.seed(1)
n<-100
d<-2
x<-8*matrix(runif(n*d),n,d)-3
C<-(2*pi)^(-d/2)
phi<-function(x){ return( C*exp(-sum(x^2)/2) ) }
D<-3; c1<-c(0,0); c2<-D*c(1,0); c3<-D*c(1/2,sqrt(3)/2)
func<-function(x){phi(x-c1)+phi(x-c2)+phi(x-c3)}
y<-matrix(0,n,1)
for (i in 1:n) y[i]<-func(x[i,])+0.01*rnorm(1)

arg<-c(0,0)
kernesti.regr(arg,x,y,h=0.5)
```

---

linear *Multivariate linear ridge regression estimator*

Description

Computes the parameter estimates in a linear least squares ridge regression.

Usage

```
linear(x, y, eleg=TRUE, lambda=0)
```

Arguments

x	n*d data matrix; the matrix of the values of the explanatory variables
y	n vector; the values of the response variable
eleg	TRUE or FALSE; an internal parameter related to the method of calculation
lambda	nonnegative real number; the degree of penalization in ridge regression; if lambda=0, then the usual linear least squares estimates are calculated

Value

list of beta0 and beta1; beta0 is a real number and beta1 is a d vector; beta0 is the estimate of the intercept and beta1 is the vector containing the estimates of the coefficients

Author(s)

Jussi Klemela

See Also

linear.quan,

Examples

```
set.seed(1)
n<-100
d<-2
x<-8*matrix(runif(n*d),n,d)-3
C<-(2*pi)^(-d/2)
phi<-function(x){ return( C*exp(-sum(x^2)/2) ) }
D<-3; c1<-c(0,0); c2<-D*c(1,0); c3<-D*c(1/2,sqrt(3)/2)
func<-function(x){phi(x-c1)+phi(x-c2)+phi(x-c3)}
y<-matrix(0,n,1)
for (i in 1:n) y[i]<-func(x[i,])+0.01*rnorm(1)

linear(x,y)
```

---

linear.quan *Multivariate linear quantile regression estimator*

Description

Computes the estimates of parameters for a linear quantile regression estimator.

Usage

```
linear.quan(x, y, p=0.5)
```

Arguments

x	n*d data matrix; the matrix of the values of the explanatory variables
y	n vector; the values of the response variable
p	0<p<1; the p:th conditional quantile function will be estimated

Details

numerical optimization is used in the calculation

Value

list of beta0 and beta1; beta0 is a real number and beta1 is a d vector; beta0 is the estimate of the intercept and beta1 is the vector containing the estimates of the coefficients

Author(s)

Jussi Klemela

See Also

linear,

Examples

```
set.seed(1)
n<-100
d<-2
x<-8*matrix(runif(n*d),n,d)-3
C<-(2*pi)^(-d/2)
phi<-function(x){ return( C*exp(-sum(x^2)/2) ) }
D<-3; c1<-c(0,0); c2<-D*c(1,0); c3<-D*c(1/2,sqrt(3)/2)
func<-function(x){phi(x-c1)+phi(x-c2)+phi(x-c3)}
y<-matrix(0,n,1)
for (i in 1:n) y[i]<-func(x[i,])+0.01*rnorm(1)

linear.quan(x,y)
```

---

loclin *Multivariate local linear regression estimator at one point*

Description

Computes the value of a multivariate local linear regression estimator at one point.

Usage

```
loclin(arg, x, y, h=1, kernel="gauss", type=0)
```

Arguments

arg	d-vector; the point where the estimate is evaluated
x	n*d data matrix; the matrix of the values of the explanatory variables
y	n vector; the values of the response variable
h	a positive real number; the smoothing parameter of the kernel estimate
kernel	a character; determines the kernel function; either "gauss" or "uniform"; in the univariate case can also be "exp"
type	integer 0,...,d; if type=0, then the regression function is estimated, otherwise the first partial derivative of the variable indicated by type is estimated

Value

a real number

Author(s)

Jussi Klemela

See Also

pcf.loclin,

Examples

```
set.seed(1)
n<-100
d<-2
x<-8*matrix(runif(n*d),n,d)-3
C<-(2*pi)^(-d/2)
phi<-function(x){ return( C*exp(-sum(x^2)/2) ) }
D<-3; c1<-c(0,0); c2<-D*c(1,0); c3<-D*c(1/2,sqrt(3)/2)
func<-function(x){phi(x-c1)+phi(x-c2)+phi(x-c3)}
y<-matrix(0,n,1)
for (i in 1:n) y[i]<-func(x[i,])+0.01*rnorm(1)

arg<-c(0,0)
loclin(arg,x,y,h=0.5)
```

---

ma *Moving average of a time series*

Description

Computes a one-sided weighted moving average from a univariate time series. The one-sided moving average can be used to predict the next value of the sequence.

Usage

```
ma(x, h=1, kernel="exp", k=length(x))
```

Arguments

x	n vector; the observed values of the time series
h	a positive real number; the smoothing parameter of the moving average
kernel	a character; determines the kernel function; either "exp", "uniform", "gauss", or "bart"
k	a positive integer; the moving average includes at most k observations

Value

a real number

Author(s)

Jussi Klemela

See Also

kernesti.regr,

Examples

```
set.seed(1)
n<-100
x<-runif(n)
ma(x)
```

---

pcf.additive *Regression function estimator in the additive model*

Description

Computes the values of an additive model regression estimator on a regular grid.

Usage

```
pcf.additive(x, y, h, N, kernel="gauss", support=NULL, M=2, eval=NULL, direc=NULL)
```

Arguments

x	n*d data matrix; the matrix of the values of the explanatory variables
y	n vector; the values of the response variable
h	a positive real number; the smoothing parameter of the kernel estimate
N	vector of d positive integers; the number of grid points for each direction
kernel	a character; determines the kernel function; either "gauss" or "uniform"
support	either NULL or a 2*d vector; the vector gives the d intervals of a rectangular support in the form c(low_1,upp_1,...,low_d,upp_d)
M	integer >=2; the number of iterations
eval	either NULL or a n*d matrix; the matrix that gives the evaluations of the coordinate functions at the data points
direc	either NULL or an integer 1,...,d; if direc is NULL, then the complete regression function estimator is estimated, otherwise only the component indicated by "direc" is estimated

Value

a piecewise constant function

Author(s)

Jussi Klemela

See Also

additive,

Examples

```
set.seed(1)
n<-100
d<-2
x<-8*matrix(runif(n*d),n,d)-3
C<-(2*pi)^(-d/2)
phi<-function(x){ return( C*exp(-sum(x^2)/2) ) }
D<-3; c1<-c(0,0); c2<-D*c(1,0); c3<-D*c(1/2,sqrt(3)/2)
func<-function(x){phi(x-c1)+phi(x-c2)+phi(x-c3)}
y<-matrix(0,n,1)
for (i in 1:n) y[i]<-func(x[i,])+0.01*rnorm(1)

num<-30  # number of grid points in one direction
pcf<-pcf.additive(x,y,h=0.5,N=c(num,num))

dp<-draw.pcf(pcf,minval=min(y))
persp(dp$x,dp$y,dp$z,phi=30,theta=-30)
contour(dp$x,dp$y,dp$z,nlevels=30)
```

---

pcf.kern.quan *An estimator of a conditional quantile function*

Description

Computes the values of an estimator of a conditional quantile function on a regular grid.

Usage

```
pcf.kern.quan(x, y, h, N, p=0.5, kernel="gauss", support=NULL)
```

Arguments

x	n*d data matrix; the matrix of the values of the explanatory variables
y	n vector; the values of the response variable
h	a positive real number; the smoothing parameter of the kernel estimate
N	vector of d positive integers; the number of grid points for each direction
p	0<p<1; the p:th quantile function will be estimated
kernel	a character; determines the kernel function; "gauss" or "uniform"
support	either NULL or a 2*d vector; the vector gives the d intervals of a rectangular support in the form c(low_1,upp_1,...,low_d,upp_d)

Value

a piecewise constant function

Author(s)

Jussi Klemela

See Also

pcf.kernesti,

Examples

```
n<-100
d<-2
x<-8*matrix(runif(n*d),n,d)-3
C<-(2*pi)^(-d/2)
phi<-function(x){ return( C*exp(-sum(x^2)/2) ) }
D<-3; c1<-c(0,0); c2<-D*c(1,0); c3<-D*c(1/2,sqrt(3)/2)
func<-function(x){phi(x-c1)+phi(x-c2)+phi(x-c3)}
y<-matrix(0,n,1)
for (i in 1:n) y[i]<-func(x[i,])+0.01*rnorm(1)

num<-30  # number of grid points in one direction
pcf<-pcf.kern.quan(x,y,h=0.5,N=c(num,num))

dp<-draw.pcf(pcf,minval=min(y))
persp(dp$x,dp$y,dp$z,phi=30,theta=-30)
contour(dp$x,dp$y,dp$z,nlevels=30)
```

---

pcf.kernesti *Multivariate kernel regression estimator*

Description

Computes the values of a multivariate kernel regression estimator (Nadaraya-Watson estimator) on a regular grid.

Usage

```
pcf.kernesti(x, y, h, N, kernel="gauss", support=NULL)
```

Arguments

x	n*d data matrix; the matrix of the values of the explanatory variables
y	n vector; the values of the response variable
h	a positive real number; the smoothing parameter of the kernel estimate
N	vector of d positive integers; the number of grid points for each direction

kernel	a character; determines the kernel function; either "gauss" or "uniform"; in the multivariate case can also be "bart"
support	either NULL or a 2*d vector; the vector gives the d intervals of a rectangular support in the form c(low_1,upp_1,...,low_d,upp_d)

Value

a piecewise constant function

Author(s)

Jussi Klemela

See Also

kernesti.regr,

Examples

```
set.seed(1)
n<-100
d<-2
x<-8*matrix(runif(n*d),n,d)-3
C<-(2*pi)^(-d/2)
phi<-function(x){ return( C*exp(-sum(x^2)/2) ) }
D<-3; c1<-c(0,0); c2<-D*c(1,0); c3<-D*c(1/2,sqrt(3)/2)
func<-function(x){phi(x-c1)+phi(x-c2)+phi(x-c3)}
y<-matrix(0,n,1)
for (i in 1:n) y[i]<-func(x[i,])+0.01*rnorm(1)

num<-30  # number of grid points in one direction
pcf<-pcf.kernesti(x,y,h=0.5,N=c(num,num))

dp<-draw.pcf(pcf,minval=min(y))
persp(dp$x,dp$y,dp$z,phi=30,theta=-30)
contour(dp$x,dp$y,dp$z,nlevels=30)
```

---

pcf.kernesti.der — *A kernel estimator of a partial derivative of a regression function*

Description

Computes the values of an estimator of a partial derivative of a regression function on a regular grid. The estimator is a partial derivative of a kernel regression estimator of the regression function.

Usage

```
pcf.kernesti.der(x, y, h, N, kernel="gauss", support=NULL, direc=1, method="ratio")
```

Arguments

x	n*d data matrix; the matrix of the values of the explanatory variables
y	n vector; the values of the response variable
h	a positive real number; the smoothing parameter of the kernel estimate
N	vector of d positive integers; the number of grid points for each direction
kernel	a character; determines the kernel function; the only allowed value is "gauss"
support	either NULL or a 2*d vector; the vector gives the d intervals of a rectangular support in the form c(low_1,upp_1,...,low_d,upp_d)
direc	integer 1,...,d; indicates which partial derivative is estimated
method	a character; determines the applied formula in the 1D case

Value

a piecewise constant function

Author(s)

Jussi Klemela

See Also

kernesti.der,

Examples

```
n<-100
d<-2
x<-8*matrix(runif(n*d),n,d)-3
C<-(2*pi)^(-d/2)
phi<-function(x){ return( C*exp(-sum(x^2)/2) ) }
D<-3; c1<-c(0,0); c2<-D*c(1,0); c3<-D*c(1/2,sqrt(3)/2)
func<-function(x){phi(x-c1)+phi(x-c2)+phi(x-c3)}
y<-matrix(0,n,1)
for (i in 1:n) y[i]<-func(x[i,])+0.01*rnorm(1)

num<-30  # number of grid points in one direction
pcf<-pcf.kernesti.der(x,y,h=0.5,N=c(num,num))

dp<-draw.pcf(pcf,minval=min(y))
persp(dp$x,dp$y,dp$z,phi=30,theta=-30)
contour(dp$x,dp$y,dp$z,nlevels=30)
```

`pcf.kernesti.marg` *An estimator of a marginal average of a regression function*

Description

Computes the values of an estimator of a marginal average of a regression function on a regular grid. Marginal averages are called also partial dependency functions.

Usage

```
pcf.kernesti.marg(x, y, h, N, kernel="gauss", coordi=1)
```

Arguments

x	n*d data matrix; the matrix of the values of the explanatory variables
y	n vector; the values of the response variable
h	a positive real number; the smoothing parameter of the kernel estimate
N	vector of d positive integers; the number of grid points for each direction
kernel	a character; determines the kernel function; the only allowed value is "gauss"
coordi	integer 1,...,d; indicates which marginal average is calculated

Value

a piecewise constant function

Author(s)

Jussi Klemela

See Also

pcf.kernesti,

Examples

```
n<-10
d<-2
x<-8*matrix(runif(n*d),n,d)-3
C<-(2*pi)^(-d/2)
phi<-function(x){ return( C*exp(-sum(x^2)/2) ) }
D<-3; c1<-c(0,0); c2<-D*c(1,0); c3<-D*c(1/2,sqrt(3)/2)
func<-function(x){phi(x-c1)+phi(x-c2)+phi(x-c3)}
y<-matrix(0,n,1)
for (i in 1:n) y[i]<-func(x[i,])+0.01*rnorm(1)

num<-30  # number of grid points in one direction
pcf<-pcf.kernesti.marg(x,y,h=0.5,N=num)
```

```
dp<-draw.pcf(pcf,minval=min(y))
plot(dp$x,dp$y,type="l")
```

---

pcf.kernesti.slice *A slice of a multivariate kernel regression estimator*

Description

Computes the values of a univariate slice of multivariate kernel regression estimator (Nadaraya-Watson estimator) on a regular grid.

Usage

```
pcf.kernesti.slice(x, y, h, N, kernel="gauss", coordi=1, p=0.5,
center=NULL, direc=NULL, radius=NULL)
```

Arguments

x	n*d data matrix; the matrix of the values of the explanatory variables
y	n vector; the values of the response variable
N	vector of d positive integers; the number of grid points for each direction
h	a positive real number; the smoothing parameter of the kernel estimate
kernel	a character; determines the kernel function; either "gauss" or "uniform"
coordi	integer 1,...,d; the direction of the slice
p	0<p<1; the slice goes through the p:th quantile, estimated from data x; this parameter is used if center=NULL
center	either NULL or a d-vector; gives the point which is intersected by the slice
direc	either NULL or a d-vector; gives the direction of the slice
radius	either NULL or a positive real number; gives the radius of the slice

Value

a piecewise constant univariate function

Author(s)

Jussi Klemela

See Also

kernesti.regr,

Examples

```
n<-100
d<-2
x<-8*matrix(runif(n*d),n,d)-3
C<-(2*pi)^(-d/2)
phi<-function(x){ return( C*exp(-sum(x^2)/2) ) }
D<-3; c1<-c(0,0); c2<-D*c(1,0); c3<-D*c(1/2,sqrt(3)/2)
func<-function(x){phi(x-c1)+phi(x-c2)+phi(x-c3)}
y<-matrix(0,n,1)
for (i in 1:n) y[i]<-func(x[i,])+0.01*rnorm(1)

num<-30  # number of grid points in one direction
pcf<-pcf.kernesti.slice(x,y,h=0.5,N=num)

dp<-draw.pcf(pcf,minval=min(y))
plot(dp$x,dp$y,type="l")
```

`pcf.loclin` *Multivariate local linear estimator*

Description

Computes the values of a multivariate local linear regression estimator on a regular grid.

Usage

```
pcf.loclin(x, y, h, N, type=0, kernel="gauss", support=NULL, alt=FALSE, alt2=FALSE)
```

Arguments

`x`	n*d data matrix; the matrix of the values of the explanatory variables
`y`	n vector; the values of the response variable
`h`	a positive real number; the smoothing parameter of the kernel estimate
`N`	vector of d positive integers; the number of grid points for each direction
`type`	integer 0,...,d; if type=0, then the regression function is estimated, otherwise the first partial derivative of the variable indicated by type is estimated
`kernel`	a character; determines the kernel function; either "gauss" or "uniform"
`support`	either NULL or a 2*d vector; the vector gives the d intervals of a rectangular support in the form c(low_1,upp_1,...,low_d,upp_d)
`alt`	an internal parameter
`alt2`	an internal parameter

Value

a piecewise constant function

Author(s)

Jussi Klemela

See Also

pcf.kernesti,

Examples

```
set.seed(1)
n<-100
d<-2
x<-8*matrix(runif(n*d),n,d)-3
C<-(2*pi)^(-d/2)
phi<-function(x){ return( C*exp(-sum(x^2)/2) ) }
D<-3; c1<-c(0,0); c2<-D*c(1,0); c3<-D*c(1/2,sqrt(3)/2)
func<-function(x){phi(x-c1)+phi(x-c2)+phi(x-c3)}
y<-matrix(0,n,1)
for (i in 1:n) y[i]<-func(x[i,])+0.01*rnorm(1)

num<-30  # number of grid points in one direction
pcf<-pcf.loclin(x,y,h=0.5,N=c(num,num))

dp<-draw.pcf(pcf,minval=min(y))
persp(dp$x,dp$y,dp$z,phi=30,theta=-30)
contour(dp$x,dp$y,dp$z,nlevels=30)
```

---

pcf.single.index — *Regression function estimator in the single index model*

Description

Computes the values of a single index model regression estimator on a regular grid.

Usage

```
pcf.single.index(x, y, h, N, kernel="gauss", support=NULL, method="poid",
argd=colMeans(x), type="si")
```

Arguments

x	n*d data matrix; the matrix of the values of the explanatory variables
y	n vector; the values of the response variable
h	a positive real number; the smoothing parameter of the kernel estimate
N	vector of d positive integers; the number of grid points for each direction
kernel	a character; determines the kernel function; either "gauss" or "uniform"

support	either NULL or a 2*d vector; the vector gives the d intervals of a rectangular support in the form c(low_1,upp_1,...,low_d,upp_d)
method	character string; "poid", "aved", "iter", or "nume"; if method="poid", then the direction vector is estimated using the reference point optionally given in the argument "argd"; if method="aved", then the average derivative method is used; if method="iter", then an iterative algorithm is used; if method="nume", then numerical optimization is used
argd	d vector; the point optionally used in the estimation of the direction vector
type	character string; if type="si", then the direction vector is estimated once, and the same direction vector is used at every grid point, otherwise the estimate of the direction vector may depend on the point of estimation

Value

a piecewise constant function

Author(s)

Jussi Klemela

See Also

single.index,

Examples

```
set.seed(1)
n<-100
d<-2
x<-8*matrix(runif(n*d),n,d)-3
C<-(2*pi)^(-d/2)
phi<-function(x){ return( C*exp(-sum(x^2)/2) ) }
D<-3; c1<-c(0,0); c2<-D*c(1,0); c3<-D*c(1/2,sqrt(3)/2)
func<-function(x){phi(x-c1)+phi(x-c2)+phi(x-c3)}
y<-matrix(0,n,1)
for (i in 1:n) y[i]<-func(x[i,])+0.01*rnorm(1)

num<-30  # number of grid points in one direction
pcf<-pcf.single.index(x,y,h=0.5,N=c(num,num))

dp<-draw.pcf(pcf,minval=min(y))
persp(dp$x,dp$y,dp$z,phi=30,theta=-30)
contour(dp$x,dp$y,dp$z,nlevels=30)
```

pp.regression *Projection pursuit regression*

Description

Computes the value of a projection pursuit regression estimator at one point.

Usage

```
pp.regression(x, y=NULL, arg=NULL, residu=NULL, teet=NULL, h=1, kernel="gauss",
M=2, method="poid", argd=NULL, vect=FALSE, seed=1)
```

Arguments

x	n*d data matrix; the matrix of the values of the explanatory variables
y	n vector or NULL; the values of the response variable
arg	d vector or NULL; the point where the value of the regression function is estimated
residu	NULL or n*M matrix of residuals; for each step the n vector of residuals is given; the first residual is the vector of y-observations
teet	NULL or M*d matrix; for each step the direction vector chosen by the optimizer
h	a positive real number; the smoothing parameter of the kernel estimate
kernel	a character; determines the kernel function; either "gauss" or "uniform"
M	integer >=2; the number of iterations
method	character string; "poid", "aved", "iter", or "nume"; if method="poid", then the direction vector is estimated using the reference point optionally given in the argument "argd"; if method="aved", then the average derivative method is used; if method="iter", then an iterative algorithm is used; if method="nume", then numerical optimization is used
argd	d vector; the point optionally used in the estimation of the direction vector
vect	TRUE or FALSE; internal parameter
seed	real number; the seed for the random number generator

Value

list of eval, residu, teet, and value; "eval" is a n*d matrix of the evaluations of the estimated component functions at the data points; "residu" is n*M matrix which contains the sequence of estimates evaluated at the observations; "teet" is M*d matrix which gives for each iteration step the direction chosen by the optimization procedure; "value" is a real number giving the estimated value of the regression function at one point

Author(s)

Jussi Klemela

See Also

`single.index`,

Examples

```
set.seed(1)
n<-100
d<-2
x<-8*matrix(runif(n*d),n,d)-3
C<-(2*pi)^(-d/2)
phi<-function(x){ return( C*exp(-sum(x^2)/2) ) }
D<-3; c1<-c(0,0); c2<-D*c(1,0); c3<-D*c(1/2,sqrt(3)/2)
func<-function(x){phi(x-c1)+phi(x-c2)+phi(x-c3)}
y<-matrix(0,n,1)
for (i in 1:n) y[i]<-func(x[i,])+0.01*rnorm(1)

pp.regression(x,y)

arg<-c(0,0)
pp.regression(x,y,arg=arg)
```

`quantil.emp` *Empirical p:th quantile*

Description

Computes the value of the empirical p:th quantile.

Usage

```
quantil.emp(y, p)
```

Arguments

`y`	n vector; the observed data
`p`	0<p<1: the p:th quantile will be estimated

Value

a real number

Author(s)

Jussi Klemela

See Also

`emp.quantile`,

Examples

```
set.seed(2)
n<-100
y<-matrix(runif(n),n,1)
quantil.emp(y,p=0.05)
```

---

single.index *Single index model regression estimator at one point*

Description

Computes the value of a single index model regression estimator at one point.

Usage

```
single.index(x, y, arg=NULL, h=1, kernel="gauss", M=2, method="iter",
vect=FALSE, argd=arg, take=length(y), seed=1)
```

Arguments

x	n*d data matrix; the matrix of the values of the explanatory variables
y	n vector; the values of the response variable
arg	NULL or d vector; the point where the value of the regression function is estimated
h	a positive real number; the smoothing parameter of the kernel estimate
kernel	a character; determines the kernel function; either "gauss" or "uniform"
M	integer >=2; the number of iterations
method	character string; "poid", "aved", "iter", or "nume"; if method="poid", then the direction vector is estimated using the reference point optionally given in the argument "argd"; if method="aved", then the average derivative method is used; if method="iter", then an iterative algorithm is used; if method="nume", then numerical optimization is used
vect	TRUE or FALSE; internal parameter
argd	d vector; the point optionally used in the estimation of the direction vector
take	positive integer
seed	real number; the seed for the random number generator

Value

either d vector or a real value; if arg=NULL, then the estimated index is returned, otherwise the estimate at "arg" is returned

Author(s)

Jussi Klemela

See Also

pcf.single.index,

Examples

```
set.seed(1)
n<-100
d<-2
x<-8*matrix(runif(n*d),n,d)-3
C<-(2*pi)^(-d/2)
phi<-function(x){ return( C*exp(-sum(x^2)/2) ) }
D<-3; c1<-c(0,0); c2<-D*c(1,0); c3<-D*c(1/2,sqrt(3)/2)
func<-function(x){phi(x-c1)+phi(x-c2)+phi(x-c3)}
y<-matrix(0,n,1)
for (i in 1:n) y[i]<-func(x[i,])+0.01*rnorm(1)

single.index(x,y)

arg<-c(0,0)
single.index(x,y,arg=arg)
```

Index