Package ‘locpol’

November 29, 2022

Version 0.8.0

Title Kernel Local Polynomial Regression

Description Computes local polynomial estimators for the regression and also density. It comprises several different utilities to handle kernel estimators.

Depends R (>= 2.5.0), graphics, stats

License GPL (>= 2)

NeedsCompilation yes

Author Jorge Luis Ojeda Cabrera [aut, cre], Bastiaan Quast [ctb] (<https://orcid.org/0000-0002-2951-3577>)

Maintainer Jorge Luis Ojeda Cabrera <jojeda@unizar.es>

Repository CRAN

Date/Publication 2022-11-29 07:35:21 UTC

R topics documented:

bivNPest *Bivariate Local estimation.*

Description

Simple bivariate Local density and regression estimation with weights.

Usage

```
    bivDens(X,weig,K,H)
    bivReg(X,Y,weig,K,H)
    ## S3 method for class 'bivNpEst'
predict(object,newdata,...)
    ## S3 method for class 'bivNpEst'
plot(x,...)
```

Arguments

`X`	Covariate or independent data, should be a `data.frame` or `matrix`, whose two first two columns are used.
`Y`	Response data, a `vector`.
`weig`	Vector of weigths for each observations.
`K`	Bivariate kernel function as `bivDens` and `bivReg`.
`H`	Bandwidth matrix. Its default value is determined by `mayBeBwSel`.
`object, x`	`bivNpEst` class objects, those returned by `bivDens` and `bivReg` functions.
`newdata`	Data, should be a `data.frame` where the density or regressions is going to be predicted.
`...`	Further graphical parameters. These parameters should agree with those in `persp`.

Details

The functions `bivDens` and `bivReg` provide a very basic interface that allows bivariate local estimation with weights. It implements basic kernel density estimator and Nadaraya–Watson estimator for bivariate data. Very simple interface methods allow the prediction and plotting of these estimators.

The only bivariate kernels provided are `epaK2d` and `gauK2d`. New ones can be added in the same way as functions with a vector of length 2.

The default bandwidth selector (see `mayBeBwSel`) that has been provided *is not optimal or good in any sense*. It has been added as a simple way to provide an easy, fast and simple way to be able to use the estimators.

The graphical parameters allowed for `...` in `plot(x,...)` are those that appears in the function persp. The list `plotBivNpEstOpts` provide a default for some of these graphical parameters.

Value

A list containing:

X	Covariate data.
Y	Response data
H	Bandwidth matrix
estFun	Estimator function.

Author(s)

Jorge Luis Ojeda Cabrera.

Examples

```
n <- 100
d <- data.frame(x=rexp(n,rate=1/2),y=rnorm(n))
## x is a length-biased version of an exp. dist. with rate 1.
dDen <- bivDens(d,weig=1/d$x)
plot(dDen,r=5)
d <- data.frame(X1=runif(n),X2=runif(n))
d$Y <-  exp(10*d$X1+d$X2^2)
dDen <- bivDens(d[,c("X1","X2")])
plot(dDen,r=5)
dReg <- bivReg(d[,c("X1","X2")],d$Y)
plot(dReg,r=5)
plot(dReg,r=5,phi=20,theta=40)
```

---

compKernVals *Compute kernel values.*

Description

Some R code provided to compute kernel related values.

Usage

```
computeRK(kernel, lower=dom(kernel)[[1]], upper=dom(kernel)[[2]],
subdivisions = 25)
computeK4(kernel, lower=dom(kernel)[[1]], upper=dom(kernel)[[2]],
subdivisions = 25)
computeMu(i, kernel, lower=dom(kernel)[[1]], upper=dom(kernel)[[2]],
subdivisions = 25)
computeMu0(kernel, lower=dom(kernel)[[1]], upper=dom(kernel)[[2]],
subdivisions = 25)
Kconvol(kernel,lower=dom(kernel)[[1]],upper=dom(kernel)[[2]],
subdivisions = 25)
```

Arguments

`kernel`	Kernel used to perform the estimation, see `Kernels`
`i`	Order of kernel moment to compute
`lower, upper`	Integration limits.
`subdivisions`	the maximum number of subintervals.

Details

These functions uses function `integrate`.

Value

A numeric value returning:

`computeK4`	The fourth order autoconvolution of `K`.
`computeRK`	The second order autoconvolution of `K`.
`computeMu0`	The integral of `K`.
`computeMu2`	The second order moment of `K`.
`computeMu`	The i-th order moment of `K`.
`Kconvol`	The autoconvolution of `K`.

These functions are implemented by means of `integrate`.

Author(s)

Jorge Luis Ojeda Cabrera.

References

Fan, J. and Gijbels, I. *Local polynomial modelling and its applications*\/. Chapman & Hall, London (1996).

Wand, M.~P. and Jones, M.~C. *Kernel smoothing*\/. Chapman and Hall Ltd., London (1995).

See Also

`RK`, Kernel characteristics, `integrate`.

Examples

```
## Note that lower and upper params are set in the definition to
## use 'dom()' function.
g <- function(kernels)
{
mu0 <- sapply(kernels,function(x) computeMu0(x,))
mu0.ok <- sapply(kernels,mu0K)
mu2 <- sapply(kernels,function(x) computeMu(2,x))
mu2.ok <- sapply(kernels,mu2K)
Rk.ok <- sapply(kernels,RK)
```

```
RK <- sapply(kernels,function(x) computeRK(x))
K4 <- sapply(kernels,function(x) computeK4(x))
res <- data.frame(mu0,mu0.ok,mu2,mu2.ok,RK,Rk.ok,K4)
res
}
g(kernels=c(EpaK,gaussK,TriweigK,TrianK))
```

---

denCVBwSelC *CV bandwidth selector for density*

Description

Computes Cross Validation bandwidth selector for the Parzen–Rosenblatt density estimator...

Usage

```
denCVBwSelC(x, kernel = gaussK, weig = rep(1, length(x)),
            interval = .lokestOptInt)
```

Arguments

`x`	vector with data points.
`kernel`	Kernel used to perform the estimation, see `Kernels`.
`weig`	Vector of weights for observations.
`interval`	A range of values where to look for the bandwidth parameter.

Details

The selector is implemented using its definition.

Value

A numeric value with the bandwidth.

Author(s)

Jorge Luis Ojeda Cabrera.

References

Fan, J. and Gijbels, I. *Local polynomial modelling and its applications*\/. Chapman & Hall, London (1996).

Wand, M.~P. and Jones, M.~C. *Kernel smoothing*\/. Chapman and Hall Ltd., London (1995).

See Also

`bw.nrd0`, `dpik`.

Examples

```
stdy <- function(size=100,rVar=rnorm,dVar=dnorm,kernel=gaussK,x=NULL)
{
if( is.null(x) ) x <- rVar(size)
Tc <- system.time( dbwc <- denCVBwSelC(x,kernel) )[3]
ucvT <- system.time( ucvBw <- bw.ucv(x,lower=0.00001,upper=2.0) )[3]
nrdT <- system.time( nrdBw <- bw.nrd(x) )[3]
{
xeval <- seq( min(x)+dbwc , max(x)-dbwc ,length=50)
hist(x,probability=TRUE)
lines(xeval,trueDen <- dVar(xeval),col="red")
lines(density(x),col="cyan")
xevalDenc <- PRDenEstC(x,xeval,dbwc,kernel)
dencMSE <- mean( (trueDen-xevalDenc)^2 )
xevalucvDen <- PRDenEstC(x,xeval,ucvBw,kernel)
ucvMSE <- mean( (trueDen-xevalucvDen)^2 )
xevalDenNrd <- PRDenEstC(x,xeval,nrdBw,kernel)
nrdMSE <- mean( (trueDen-xevalDenNrd)^2 )
lines(xevalDenc,col="green")
lines(xevalucvDen,col="blue")
lines(xevalDenNrd,col="grey")
}
return( cbind(  bwVal=c(evalC=dbwc,ucvBw=ucvBw,nrdBw=nrdBw),
mse=c(dencMSE,ucvMSE,nrdMSE),
time=c(Tc,ucvT,nrdT) ) )
}
stdy(100,kernel=gaussK)
stdy(100,rVar=rexp,dVar=dexp,kernel=gaussK)
## check stdy with other kernel, distributions
```

`equivKernel` *Equivalent Kernel.*

Description

Computes the Equivalent kernel for the local polynomial estimation.

Usage

```
equivKernel(kernel,nu,deg,lower=dom(kernel)[[1]],upper=dom(kernel)[[2]],
subdivisions=25)
```

Arguments

`nu`	Orders of derivative to estimate.
`deg`	Degree of Local polynomial estimator.
`kernel`	Kernel used to perform the estimation, see `Kernels`
`lower, upper`	Integration limits.
`subdivisions`	the maximum number of subintervals.

Details

The definition of the Equivalent kernel for the local polynomial estimation can be found in page 64 in Fan and Gijbels(1996). The implementation uses `computeMu` to compute matrix S and then returns a function object

Value

Returns a vector whose components are the equivalent kernel used to compute the local polynomial estimator for the derivatives in `nu`.

Author(s)

Jorge Luis Ojeda Cabrera.

References

Fan, J. and Gijbels, I. *Local polynomial modelling and its applications*\/. Chapman & Hall, London (1996).

See Also

`cteNuK`, `adjNuK`.

Examples

```
## Some kernels and equiv. for higher order
## compare with p=1
curve(EpaK(x),-3,3,ylim=c(-.5,1))
f <- equivKernel(EpaK,0,3)
curve(f(x),-3,3,add=TRUE,col="blue")
curve(gaussK(x),-3,3,add=TRUE)
f <- equivKernel(gaussK,0,3)
curve(f(x),-3,3,add=TRUE,col="blue")
## Draw several Equivalent locl polynomial kernels
curve(EpaK(x),-3,3,ylim=c(-.5,1))
for(p in 1:5){
curve(equivKernel(gaussK,0,p)(x),-3,3,add=TRUE)
    }
```

`KernelChars` *Kernel characteristics*

Description

For a given kernel these functions return some of the most commonly used numeric values related to them.

Usage

```
RK(K)
RdK(K)
mu2K(K)
mu0K(K)
K4(K)
dom(K)
```

Arguments

K	A kernel as given in Kernels

Details

Most of these functions are implemented as an attribute of every kernel. For the computations of the numeric value for these quantities, see references.

Value

A numeric value returning:

RK	The L_2 norm of K.
RdK	The L_2 norm of the derivative of K.
mu2K	The second order moment of K.
mu2K	The zeroth order moment of K.
dom	The support of K.
K4	The fourth order autoconvolution of K at $x = 0$.

Author(s)

Jorge Luis Ojeda Cabrera.

References

Fan, J. and Gijbels, I. *Local polynomial modelling and its applications*\/. Chapman & Hall, London (1996).

Wand, M.~P. and Jones, M.~C. *Kernel smoothing*\/. Chapman and Hall Ltd., London (1995).

See Also

Kernels, Compute kernel values.

Examples

```
## Note that lower and upper params are set in the definition to
## use 'dom()' function.
g <- function(kernels)
{
mu0 <- sapply(kernels,function(x) computeMu0(x,))
mu0.ok <- sapply(kernels,mu0K)
mu2 <- sapply(kernels,function(x) computeMu(2,x))
mu2.ok <- sapply(kernels,mu2K)
Rk.ok <- sapply(kernels,RK)
RK <- sapply(kernels,function(x) computeRK(x))
K4 <- sapply(kernels,function(x) computeK4(x))
res <- data.frame(mu0,mu0.ok,mu2,mu2.ok,RK,Rk.ok,K4)
res
}
g(kernels=c(EpaK,gaussK,TriweigK,TrianK))
```

`kernelCte` *Kernel Constants used in Bandwidth Selection.*

Description

These are values depending on the kernel and the local polynomial degrees that are used in bandwidth selection, as proposed in Fan and Gijbels(1996).

Usage

```
cteNuK(nu,p,kernel,lower=dom(kernel)[[1]],upper=dom(kernel)[[2]],
subdivisions= 25)
adjNuK(nu,p,kernel,lower=dom(kernel)[[1]],upper=dom(kernel)[[2]],
subdivisions= 25)
```

Arguments

`nu`	Order of derivative to estimate.
`p`	Degree of Local polynomial estimator.
`kernel`	Kernel used to perform the estimation, see `Kernels`
`lower, upper`	Integration limits.
`subdivisions`	the maximum number of subintervals.

Details

`cteNuK` is computed using Compute kernel values and `link{equivKernel}` jointly with the numerical integration utility `integrate`. `adjNuK` is implemented using quotients of previous functions. See Fan and Gijbels(1996) pages 67 and 119.

Value

Both functions returns numeric values.

Author(s)

Jorge Luis Ojeda Cabrera.

References

Fan, J. and Gijbels, I. *Local polynomial modelling and its applications*\/. Chapman & Hall, London (1996).

Wand, M.~P. and Jones, M.~C. *Kernel smoothing*\/. Chapman and Hall Ltd., London (1995).

See Also

`regCVBwSelC`, `pluginBw`, `integrate`.

`Kernels` *Kernels.*

Description

Definition of common kernels used in local polynomial estimation.

Usage

```
CosK(x)
EpaK(x)
Epa2K(x)
gaussK(x)
```

Arguments

`x`	Numeric vector o value.

Details

The implementation of these kernels is done by means functions that can operate on vectors.

Most common referred numeric values for these kernels are provided as attributes, see `RK`, `mu0K`, etc....

Author(s)

Jorge Luis Ojeda Cabrera.

References

Fan, J. and Gijbels, I. *Local polynomial modelling and its applications\/*. Chapman & Hall, London (1996).

Wand, M.~P. and Jones, M.~C. *Kernel smoothing\/*. Chapman and Hall Ltd., London (1995).

See Also

RK, mu0K.

---

locCteWeights — *Local Polynomial Weights*

Description

Local Constant and local Linear estimator with weight.

Usage

```
locCteWeightsC(x, xeval, bw, kernel, weig = rep(1, length(x)))
locLinWeightsC(x, xeval, bw, kernel, weig = rep(1, length(x)))
locPolWeights(x, xeval, deg, bw, kernel, weig = rep(1, length(x)))
locWeightsEval(lpweig, y)
locWeightsEvalC(lpweig, y)
```

Arguments

`x`	x covariate data values.
`y`	y response data values.
`xeval`	Vector with evaluation points.
`bw`	Smoothing parameter, bandwidth.
`deg`	Local polynomial estimation degree (p).
`kernel`	Kernel used to perform the estimation, see Kernels
`weig`	Vector of weights for observations.
`lpweig`	Local polynomial weights $(X^TWX)^{-1}X^TW$ evaluated at `xeval` matrix.

Details

`locCteWeightsC` and `locLinWeightsC` computes local constant and local linear weights, say any of the entries of the vector $(X^TWX)^{-1}X^TW$ for $p = 0$ and $p = 1$ resp. `locWeightsEvalC` and `locWeightsEval` computes local the estimator for a given vector of responses `y`

Value

`locCteWeightsC` and `locLinWeightsC` returns a list with two components

`den`	Estimation of $(n * h * f(x))^{p+1}$ being h the bandwidth `bw`.
`locWeig`	$(X^TWX)^{-1}X^TW$ evaluated at `xeval` Matrix.

Author(s)

Jorge Luis Ojeda Cabrera.

References

Fan, J. and Gijbels, I. *Local polynomial modelling and its applications*\/. Chapman & Hall, London (1996).

Wand, M.~P. and Jones, M.~C. *Kernel smoothing*\/. Chapman and Hall Ltd., London (1995).

See Also

Kernels, locpol.

Examples

```
size <- 200
sigma <- 0.25
deg <- 1
kernel <- EpaK
bw <- .25
xeval <- 0:100/100
regFun <- function(x) x^3
x <- runif(size)
y <- regFun(x) + rnorm(x, sd = sigma)
d <- data.frame(x, y)
lcw <- locCteWeightsC(d$x, xeval, bw, kernel)$locWeig
lce <- locWeightsEval(lcw, y)
lceB <- locCteSmootherC(d$x, d$y, xeval, bw, kernel)$beta0
mean((lce-lceB)^2)
    llw <- locLinWeightsC(d$x, xeval, bw, kernel)$locWeig
lle <- locWeightsEval(llw, y)
lleB <- locLinSmootherC(d$x, d$y, xeval, bw, kernel)$beta0
mean((lle-lleB)^2)
```

---

locpol *Local Polynomial estimation.*

Description

Formula interface for the local polynomial estimation.

Usage

```
    locpol(formula,data,weig=rep(1,nrow(data)),bw=NULL,kernel=EpaK,deg=1,
             xeval=NULL,xevalLen=100)
    confInterval(x)
    ## S3 method for class 'locpol'
residuals(object,...)
```

```
    ## S3 method for class 'locpol'
fitted(object,deg=0,...)
    ## S3 method for class 'locpol'
summary(object,...)
    ## S3 method for class 'locpol'
print(x,...)
    ## S3 method for class 'locpol'
plot(x,...)
```

Arguments

formula	formula as in lm, only first covariate is used.
data	data frame with data.
weig	Vector of weights for each observations.
bw	Smoothing parameter, bandwidth.
kernel	Kernel used to perform the estimation, see Kernels
deg	Local polynomial estimation degree (p).
xeval	Vector of evaluation points. By default xevalLen points between the min. and the max. of the regressors.
xevalLen	Length of xeval if it is NULL
x	A locpol object.
object	A locpol object.
...	Any other required argument.

Details

This is an interface to the local polynomial estimation function that provides basic lm functionality. summary and print methods shows very basic information about the fit, fitted return the estimation of the derivatives if deg is larger than 0, and plot provides a plot of data, local polynomial estimation and the variance estimation.

Variance estimation is carried out by means of the local constant regression estimation of the squared residuals.

confInterval provides confidence intervals for all points in x$lpFit$[,x$X], say those in xeval.

Value

A list containing among other components:

mf	Model frame for data and formula.
data	data frame with data.
weig	Vector of weight for each observations.
xeval	Vector of evaluation points.
bw	Smoothing parameter, bandwidth.
kernel	Kernel used, see Kernels

`KName`	Kernel name, a string with the name of kernel.
`deg`	Local polynomial estimation degree (p).
`X,Y`	Names in `data` of the response and covariate. They are also used in `lpFit` to name the fitted data.
`residuals`	Residuals of the local polynomial fit.
`lpFit`	Data frame with the local polynomial fit. It contains covariate, response, derivatives estimation, X density estimation, and variance estimation.

Author(s)

Jorge Luis Ojeda Cabrera.

References

Fan, J. and Gijbels, I. *Local polynomial modelling and its applications\/*. Chapman & Hall, London (1996).

Wand, M.~P. and Jones, M.~C. *Kernel smoothing\/*. Chapman and Hall Ltd., London (1995).

Crist'obal, J. A. and Alcal\'a, J. T. (2000). *Nonparametric regression estimators for length biased data\/*. J. Statist. Plann. Inference, 89, pp. 145-168.

Ahmad, Ibrahim A. (1995) *On multivariate kernel estimation for samples from weighted distributions\/*. Statistics & Probability Letters, 22, num. 2, pp. 121-129

See Also

`locpoly` from package **KernSmooth**, `ksmooth` and `loess` in **stats** (but from earlier package `modreg`).

Examples

```
N <- 250
xeval <- 0:100/100
##  ex1
d <- data.frame(x = runif(N))
d$y <- d$x^2 - d$x + 1 + rnorm(N, sd = 0.1)
r <- locpol(y~x,d)
plot(r)
##  ex2
d <- data.frame(x = runif(N))
d$y <- d$x^2 - d$x + 1 + (1+d$x)*rnorm(N, sd = 0.1)
r <- locpol(y~x,d)
plot(r)
## notice:
rr <- locpol(y~x,d,xeval=runif(50,-1,1))
## notice x has null dens. outside (0,1)
## plot(rr) raises an error, no conf. bands outside (0,1).
## length biased data !!
d <- data.frame(x = runif(10*N))
d$y <- d$x^2 - d$x + 1 + (rexp(10*N,rate=4)-.25)
posy <- d$y[ whichYPos <- which(d$y>0) ];
d <- d[sample(whichYPos, N,prob=posy,replace=FALSE),]
```

```
rBiased <- locpol(y~x,d)
r <- locpol(y~x,d,weig=1/d$y)
plot(d)
points(r$lpFit[,r$X],r$lpFit[,r$Y],type="l",col="blue")
points(rBiased$lpFit[,rBiased$X],rBiased$lpFit[,rBiased$Y],type="l")
curve(x^2 - x + 1,add=TRUE,col="red")
```

`locpolSmoothers` *Local Polynomial estimation.*

Description

Computes the local polynomial estimation of the regression function.

Usage

```
locCteSmootherC(x, y, xeval, bw, kernel, weig = rep(1, length(y)))
locLinSmootherC(x, y, xeval, bw, kernel, weig = rep(1, length(y)))
locCuadSmootherC(x, y, xeval, bw, kernel, weig = rep(1, length(y)))
locPolSmootherC(x, y, xeval, bw, deg, kernel, DET = FALSE,
weig = rep(1, length(y)))
looLocPolSmootherC(x, y, bw, deg, kernel, weig = rep(1, length(y)),
        DET = FALSE)
```

Arguments

`x`	x covariate data values.
`y`	y response data values.
`xeval`	Vector of evaluation points.
`bw`	Smoothing parameter, bandwidth.
`kernel`	Kernel used to perform the estimation, see `Kernels`
`weig`	Vector of weights for observations.
`deg`	Local polynomial estimation degree (p).
`DET`	Boolean to ask for the computation of the determinant if the matrix X^TWX.

Details

All these function perform the estimation of the regression function for different degrees. While `locCteSmootherC`, `locLinSmootherC`, and `locCuadSmootherC` uses direct computations for the degrees 0,1 and 2 respectively, `locPolSmootherC` implements a general method for any degree. Particularly useful can be `looLocPolSmootherC`(Leave one out) which computes the local polynomial estimator for any degree as `locPolSmootherC` does, but estimating $m(x_i)$ without using i–th observation on the computation.

Value

A data frame whose components gives the evaluation points, the estimator for the regression function $m(x)$ and its derivatives at each point, and the estimation of the marginal density for x to the $p+1$ power. These components are given by:

x
: Evaluation points.

beta0, beta1, beta2,...
: Estimation of the i-th derivative of the regression function $(m^{(i)}(x))$ for $i = 0, 1, \ldots$.

den
: Estimation of $(n * h * f(x))^{p+1}$, being h the bandwidth bw.

Author(s)

Jorge Luis Ojeda Cabrera.

References

Fan, J. and Gijbels, I. *Local polynomial modelling and its applications*\/. Chapman & Hall, London (1996).

Wand, M.~P. and Jones, M.~C. *Kernel smoothing*\/. Chapman and Hall Ltd., London (1995).

See Also

locpoly from package **KernSmooth**, ksmooth and loess in **stats** (but from earlier package modreg).

Examples

```
N <- 100
xeval <- 0:10/10
d <- data.frame(x = runif(N))
bw <- 0.125
fx <- xeval^2 - xeval + 1
## Non random
d$y <- d$x^2 - d$x + 1
cuest <- locCuadSmootherC(d$x, d$y ,xeval, bw, EpaK)
lpest2 <- locPolSmootherC(d$x, d$y , xeval, bw, 2, EpaK)
print(cbind(x = xeval, fx, cuad0 = cuest$beta0,
lp0 = lpest2$beta0, cuad1 = cuest$beta1, lp1 = lpest2$beta1))
## Random
d$y <- d$x^2 - d$x + 1 + rnorm(d$x, sd = 0.1)
cuest <- locCuadSmootherC(d$x,d$y , xeval, bw, EpaK)
lpest2 <- locPolSmootherC(d$x,d$y , xeval, bw, 2, EpaK)
lpest3 <- locPolSmootherC(d$x,d$y , xeval, bw, 3, EpaK)
cbind(x = xeval, fx, cuad0 = cuest$beta0, lp20 = lpest2$beta0,
lp30 = lpest3$beta0, cuad1 = cuest$beta1, lp21 = lpest2$beta1,
lp31 = lpest3$beta1)
```

pluginBw *Plugin Bandwidth selector.*

Description

Implements a plugin bandwidth selector for the regression function.

Usage

```
pluginBw(x, y, deg, kernel, weig = rep(1,length(y)))
```

Arguments

`x`	x covariate values.
`y`	y response values.
`deg`	degree of the local polynomial.
`kernel`	Kernel used to perform the estimation, see `Kernels`.
`weig`	Vector of weights for observations.

Details

Computes the plug-in bandwidth selector as shown in Fan and Gijbels(1996) book using pilots estimates as given in page 110-112 (Rule of thumb for bandwidth selection). Currently, only even values of p are can be used.

Value

A numeric value.

Note

Currently, only even values of p are can be used.

Author(s)

Jorge Luis Ojeda Cabrera.

References

Fan, J. and Gijbels, I. *Local polynomial modelling and its applications*\/. Chapman & Hall, London (1996).

Wand, M.~P. and Jones, M.~C. *Kernel smoothing*\/. Chapman and Hall Ltd., London (1995).

See Also

`thumbBw`, `regCVBwSelC`.

Examples

```
size <- 200
sigma <- 0.25
deg <- 1
kernel <- EpaK
xeval <- 0:100/100
regFun <- function(x) x^3
x <- runif(size)
y <- regFun(x) + rnorm(x, sd = sigma)
d <- data.frame(x, y)
cvBwSel <- regCVBwSelC(d$x,d$y, deg, kernel, interval = c(0, 0.25))
thBwSel <- thumbBw(d$x, d$y, deg, kernel)
piBwSel <- pluginBw(d$x, d$y, deg, kernel)
est <- function(bw, dat, x) return(locPolSmootherC(dat$x,dat$y, x, bw, deg,
kernel)$beta0)
ise <- function(val, est) return(sum((val - est)^2 * xeval[[2]]))
plot(d$x, d$y)
trueVal <- regFun(xeval)
lines(xeval, trueVal, col = "red")
xevalRes <- est(cvBwSel, d, xeval)
cvIse <- ise(trueVal, xevalRes)
lines(xeval, xevalRes, col = "blue")
xevalRes <- est(thBwSel, d, xeval)
thIse <- ise(trueVal, xevalRes)
xevalRes <- est(piBwSel, d, xeval)
piIse <- ise(trueVal, xevalRes)
lines(xeval, xevalRes, col = "blue", lty = "dashed")
res <- rbind( bw = c(cvBwSel, thBwSel, piBwSel),
ise = c(cvIse, thIse, piIse) )
colnames(res) <- c("CV", "th", "PI")
res
```

PRDenEstC *Parzen–Rosenblatt denstiy estimator.*

Description

Parzen–Rosenblat univariate density estimator.

Usage

```
PRDenEstC(x, xeval, bw, kernel, weig = rep(1, length(x)))
```

Arguments

x	vector with data points.
xeval	Vector of evaluation points.
bw	Smoothing parameter, bandwidth.
kernel	Kernel used to perform the estimation, see Kernels
weig	Vector of weights for observations.

Details

Simple Parzen–Rosenblat univariate density estimation, computed using definition.

Value

Returns an (x,den) data frame.

x	Evaluation points.
den	Density at each x point.

Author(s)

Jorge Luis Ojeda Cabrera.

References

Fan, J. and Gijbels, I. *Local polynomial modelling and its applications*\/. Chapman & Hall, London (1996).

Wand, M.~P. and Jones, M.~C. *Kernel smoothing*\/. Chapman and Hall Ltd., London (1995).

See Also

density, that uses FT to compute a kernel density estimator, bkde from package **KernSmooth** for a binned version, and bw.nrd0, dpik, denCVBwSelC for bandwidth selection.

Examples

```
N <- 100
x <-  runif(N)
xeval <- 0:10/10
b0.125 <- PRDenEstC(x, xeval, 0.125, EpaK)
b0.05 <- PRDenEstC(x, xeval, 0.05, EpaK)
cbind(x = xeval, fx = 1, b0.125 = b0.125$den, b0.05 = b0.05$den)
```

---

regCVBwSelC *Cross Validation Bandwidth selector.*

Description

Implements Cross validation bandwidth selector for the regression function.

Usage

```
regCVBwSelC(x, y, deg, kernel=gaussK,weig=rep(1,length(y)),
interval=.lokestOptInt)
```

Arguments

x	x covariate values.
y	y response values.
deg	degree of the local polynomial.
kernel	Kernel used to perform the estimation, see Kernels.
weig	Vector of weights for observations.
interval	An interval where to look for the bandwidth.

Details

Computes the weighted ASE for every bandwidth returning the minimum. The function is implemented by means of a C function that computes for a single bandwidth the ASE, and a call to `optimise` on a given interval.

Value

A numeric value.

Author(s)

Jorge Luis Ojeda Cabrera.

References

Fan, J. and Gijbels, I. *Local polynomial modelling and its applications\/*. Chapman & Hall, London (1996).

H\"ardle W.(1990) *Smoothing techniques*. Springer Series in Statistics, New York (1991).

Wand, M.~P. and Jones, M.~C. *Kernel smoothing\/*. Chapman and Hall Ltd., London (1995).

See Also

thumbBw, pluginBw.

Examples

```
size <- 200
sigma <- 0.25
deg <- 1
kernel <- EpaK
xeval <- 0:100/100
regFun <- function(x) x^3
x <- runif(size)
y <- regFun(x) + rnorm(x, sd = sigma)
d <- data.frame(x, y)
cvBwSel <- regCVBwSelC(d$x,d$y, deg, kernel, interval = c(0, 0.25))
thBwSel <- thumbBw(d$x, d$y, deg, kernel)
piBwSel <- pluginBw(d$x, d$y, deg, kernel)
est <- function(bw, dat, x) return(locPolSmootherC(dat$x,dat$y, x, bw, deg,
```

```
kernel)$beta0)
ise <- function(val, est) return(sum((val - est)^2 * xeval[[2]]))
plot(d$x, d$y)
trueVal <- regFun(xeval)
lines(xeval, trueVal, col = "red")
xevalRes <- est(cvBwSel, d, xeval)
cvIse <- ise(trueVal, xevalRes)
lines(xeval, xevalRes, col = "blue")
xevalRes <- est(thBwSel, d, xeval)
thIse <- ise(trueVal, xevalRes)
xevalRes <- est(piBwSel, d, xeval)
piIse <- ise(trueVal, xevalRes)
lines(xeval, xevalRes, col = "blue", lty = "dashed")
res <- rbind( bw = c(cvBwSel, thBwSel, piBwSel),
ise = c(cvIse, thIse, piIse) )
colnames(res) <- c("CV", "th", "PI")
res
```

`selKernel` *Kernel selection.*

Description

Uses kernel attributes to selects kernels. This function is mainly used for internal purposes.

Usage

```
selKernel(kernel)
```

Arguments

`kernel` kernel to use.

Details

Uses `RK(K)` to identify a kernel. The integer is used in the C code part to perform computations with given kernel. It allows for a kernel selection in C routines. It is used only for internal purposes.

Value

An integer that is unique for each kernel.

Warning

Used only for internal purposes.

Author(s)

Jorge Luis Ojeda Cabrera.

`simpleSmoothers` *Simple smoother*

Description

Computes simple kernel smoothing

Usage

```
simpleSmootherC(x, y, xeval, bw, kernel, weig = rep(1, length(y)))
simpleSqSmootherC(x, y, xeval, bw, kernel)
```

Arguments

`x`	x covariate data values.
`y`	y response data values.
`xeval`	Vector with evaluation points.
`bw`	Smoothing parameter, bandwidth.
`kernel`	Kernel used to perform the estimation, see `Kernels`
`weig`	weights if they are required.

Details

Computes simple smoothing, that is to say: it averages `y` values times kernel evaluated on `x` values. `simpleSqSmootherC` does the average with the square of such values.

Value

Both functions returns a `data.frame` with

`x`	x evaluation points.
`reg`	the smoothed values at `x` points.

...

Author(s)

Jorge Luis Ojeda Cabrera.

See Also

`PRDenEstC`, `Kernel characteristics`

Examples

```
size <- 1000
x <- runif(100)
bw <- 0.125
kernel <- EpaK
xeval <- 1:9/10
y <- rep(1,100)
## x kern. aver. should give density f(x)
prDen <- PRDenEstC(x,xeval,bw,kernel)$den
ssDen <- simpleSmootherC(x,y,xeval,bw,kernel)$reg
all(abs(prDen-ssDen)<1e-15)
## x kern. aver. should be f(x)*R2(K) aprox.
s2Den <- simpleSqSmootherC(x,y,xeval,bw,kernel)$reg
summary( abs(prDen*RK(kernel)-s2Den) )
summary( abs(1*RK(kernel)-s2Den) )
## x kern. aver. should be f(x)*R2(K) aprox.
for(n in c(1000,1e4,1e5))
{
s2D <- simpleSqSmootherC(runif(n),rep(1,n),xeval,bw,kernel)$reg
cat("\n",n,"\n")
print( summary( abs(1*RK(kernel)-s2D) ) )
}
```

`thumbBw` *Rule of thumb for bandwidth selection.*

Description

Implements Fan and Gijbels(1996)'s Rule of thumb for bandwidth selection

Usage

```
thumbBw(x, y, deg, kernel, weig = rep(1, length(y)))
compDerEst(x, y, p, weig = rep(1, length(y)))
```

Arguments

`x`	x covariate data values.
`y`	y response data values.
`p`	order of local polynomial estimator.
`deg`	Local polynomial estimation degree(p).
`kernel`	Kernel used to perform the estimation.
`weig`	weights if they are required.

Details

See Fan and Gijbels(1996) book, Section 4.2. This implementation is also considering weights. `compDerEst` computes the $p + 1$ derivative of the regression function in a simple manner, assuming it is a polynomial in x. `thumbBw` gives a bandwidth selector by means of pilot estimator given by `compDerEst` and the mean of residuals.

Value

`thumbBw` returns a single numeric value, while `compDerEst` returns a data frame whose components are:

`x`	x values.
`y`	y values.
`res`	residuals for the parametric estimation.
`der`	derivative estimation at x values.

Author(s)

Jorge Luis Ojeda Cabrera.

References

Fan, J. and Gijbels, I. *Local polynomial modelling and its applications*\/. Chapman & Hall, London (1996).

Wand, M.~P. and Jones, M.~C. *Kernel smoothing*\/. Chapman and Hall Ltd., London (1995).

See Also

`regCVBwSelC`, `pluginBw`.

Examples

```
size <- 200
sigma <- 0.25
deg <- 1
kernel <- EpaK
xeval <- 0:100/100
regFun <- function(x) x^3
x <- runif(size)
y <- regFun(x) + rnorm(x, sd = sigma)
d <- data.frame(x, y)
cvBwSel <- regCVBwSelC(d$x,d$y, deg, kernel, interval = c(0, 0.25))
thBwSel <- thumbBw(d$x, d$y, deg, kernel)
piBwSel <- pluginBw(d$x, d$y, deg, kernel)
est <- function(bw, dat, x) return(locPolSmootherC(dat$x,dat$y, x, bw, deg,
kernel)$beta0)
ise <- function(val, est) return(sum((val - est)^2 * xeval[[2]]))
plot(d$x, d$y)
trueVal <- regFun(xeval)
lines(xeval, trueVal, col = "red")
```

```
xevalRes <- est(cvBwSel, d, xeval)
cvIse <- ise(trueVal, xevalRes)
lines(xeval, xevalRes, col = "blue")
xevalRes <- est(thBwSel, d, xeval)
thIse <- ise(trueVal, xevalRes)
xevalRes <- est(piBwSel, d, xeval)
piIse <- ise(trueVal, xevalRes)
lines(xeval, xevalRes, col = "blue", lty = "dashed")
res <- rbind( bw = c(cvBwSel, thBwSel, piBwSel),
ise = c(cvIse, thIse, piIse) )
colnames(res) <- c("CV", "th", "PI")
res
```

Index