Package ‘locpol’

February 20, 2015

Version  0.6-0
Date     2012-10-25
Title    Kernel local polynomial regression
Author   Jorge Luis Ojeda Cabrera, <jojeda@unizar.es>
Description Computes local polynomial estimators.
Depends  R (>= 2.5.0)
Maintainer Jorge Luis Ojeda Cabrera <jojeda@unizar.es>
License  GPL-2
Repository CRAN
Date/Publication 2012-10-25 13:23:53
NeedsCompilation yes

R topics documented:

bivNPest  ................................................................. 2
compKernVals ............................................................... 3
denCVBwSelC ............................................................... 5
equivKernel ................................................................. 6
KernelChars ............................................................... 7
kernelCte ................................................................. 9
Kernels ................................................................. 10
locCteWeights ........................................................... 11
locpol ................................................................. 12
locpolSmoothers ........................................................ 15
pluginBw ............................................................... 17
PRDenEstC ............................................................... 18
regCVBwSelC .............................................................. 20
selKernel .............................................................. 21
simpleSmoothers ........................................................ 22
thumbBw ............................................................... 23

Index  26
bivN Pest

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 weights 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.
compKernVals

**Value**

A list containing:

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

**Author(s)**

Jorge Luis Ojeda Cabrera.

**Examples**

```r
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)
```

**Description**

Some R code provided to compute kernel related values.

**Usage**

```r
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)
```
compKernVals

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


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) computeMu2(x))
  mu2.ok <- sapply(kernels,mu2K)
  RK.ok <- sapply(kernels,RK)
**denCVBwSelC**

```r
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))
```

---

**Description**

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

**Usage**

```r
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**


**See Also**

`bw.nrd0`, `dpik`. 
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
deg
kernel
lower, upper
subdivisions

Orders of derivative to estimate.
Degree of Local polynomial estimator.
Kernel used to perform the estimation, see Kernels
Integration limits.
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


See Also
cteNuK, adjNuK.

Examples

```r
## 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 local 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)
}
```

<table>
<thead>
<tr>
<th>KernelChars</th>
<th>Kernel characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Description

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

\begin{itemize}
  \item \texttt{RK(K)}
  \item \texttt{Rdk(K)}
  \item \texttt{mu2K(K)}
  \item \texttt{mu0K(K)}
  \item \texttt{K4(K)}
  \item \texttt{dom(K)}
\end{itemize}

Arguments

\begin{itemize}
  \item \texttt{K} A kernel as given in \texttt{Kernels}
\end{itemize}

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:

\begin{itemize}
  \item \texttt{RK} The $L_2$ norm of $K$.
  \item \texttt{Rdk} The $L_2$ norm of the derivative of $K$.
  \item \texttt{mu2K} The second order moment of $K$.
  \item \texttt{mu0K} The zeroth order moment of $K$.
  \item \texttt{dom} The support of $K$.
  \item \texttt{K4} The fourth order autoconvolution of $K$ at $x = 0$.
\end{itemize}

Author(s)

Jorge Luis Ojeda Cabrera.

References


See Also

\texttt{Kernels}, \texttt{Compute kernel values}.
Examples

```r
# 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) computeMu2(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 band-
width selection, as proposed in Fan and Gijbels(1996).

**Usage**

```r
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 return numeric values.

Author(s)

Jorge Luis Ojeda Cabrera.

References


See Also

regCVBwSelC, pluginBw, integrate.

---

**kernels**

**Description**

Definition of common kernels used in local polynomial estimation.

**Usage**

\[
\text{CosK}(x) \\
\text{Epak}(x) \\
\text{EpaRk}(x) \\
\text{gaussK}(x) \\
\ldots
\]

**Arguments**

\[
x \quad \text{Numeric vector or value.}
\]

**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 \text{RK}, \text{mu0K}, etc.

**Author(s)**

Jorge Luis Ojeda Cabrera.
References


See Also

* RK, mu0K.

---

### locCteWeights

**Local Polynomial Weights**

**Description**

Local Constant and local Linear estimator with weight.

**Usage**

```r
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 ($X^T W X)^{-1} X^T W$ evaluated at xeval matrix.

**Details**

locCteWeightsC and locLinWeightsC computes local constant and local linear weights, say any of the entries of the vector ($X^T W X)^{-1} X^T W$ 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 \((n \cdot h \cdot f(x))^{p+1}\) being \(h\) the bandwidth \(bw\).
- `locWeig`: \((X^TWX)^{-1}X^TW\) evaluated at `xeval` Matrix.

Author(s)

Jorge Luis Ojeda Cabrera.

References


See Also

Kernels, locpol.

Examples

```r
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 <- locCteSmotherC(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 <- locLinSmotherC(d$x, d$y, xeval, bw, kernel)$beta0$mean((lle-lleB)^2)
```

locpol

*Local Polynomial estimation.*
locpol

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


See Also

locpoly from package KernSmooth, ksmooth and loess from package modreg.

Examples

N <- 250
oxval <- 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))
## Description

Computes the local polynomial estimation of the regression function.

## Usage

```r
locCteSmoothC(x, y, xeval, bw, kernel, weig = rep(1, length(y)))
locLinSmoothC(x, y, xeval, bw, kernel, weig = rep(1, length(y)))
locCuadSmoothC(x, y, xeval, bw, kernel, weig = rep(1, length(y)))
locPolSmoothC(x, y, xeval, bw, deg, kernel, DET = FALSE, weig = rep(1, length(y)))
looLocPolSmoothC(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 $X^T WX$. 

---

This code demonstrates the use of local polynomial smoothing with the `locpol` package in R. The code snippet below illustrates how to fit a local polynomial regression model and visualize the results.

```r
# 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$s <- d$x^2 - d$x + 1 + (rexp(10*N, rate=4)-.25)

posy <- d$s[ which(posy <- which(d$s>0) ) ];
d <- d[sample(which(posy, N, prob=posy, replace=FALSE),]

rbiased <- locpol(y~x,d)

plot(d)

points(r$lpFit[,r$X],r$lpFit[,r$Y],type="l",col="blue")

points(r$biased$lpFit[,r$biased$X],r$biased$lpFit[,r$biased$Y],type="l")

curve(x^2 - x + 1,add=TRUE,col="red")
```
Details

All these functions perform the estimation of the regression function for different degrees. While `locCteSmoothers`, `locLinSmoothers`, and `locCuadSmoothers` use direct computations for the degrees 0, 1, and 2 respectively, `locPolSmoothers` implements a general method for any degree. Particularly useful can be `looLocPolSmoothers` (Leave one out) which computes the local polynomial estimator for any degree as `locPolSmoothers` 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


See Also

`locpoly` from package *KernSmooth*, `ksmooth` and `loess` from package *modreg*.

Examples

```r
N <- 100
exeval <- 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$x + 1
cuest <- locCuadSmoothers(d$x, d$y, xeval, bw, Epak)
lpest2 <- locPolSmoothers(d$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$x, sd = 0.1)
cuest <- locCuadSmoothers(d$x, d$y, xeval, bw, Epak)
```
### 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 on 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


See Also

thumbBw, regCVbwSelC.

Examples

```r
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)
cvbBwSel <- regCVbwSelC(d$x, d$y, deg, kernel, interval = c(0, 0.25))
thetaBwSel <- thumbBw(d$x, d$y, deg, kernel)
pluginBwSel <- pluginBw(d$x, d$y, deg, kernel)
est <- function(bw, dat, x) return(locPolSmoootherC(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(thetaBwSel, d, xeval)
thIse <- ise(trueVal, xevalRes)
xevalRes <- est(pluginBwSel, d, xeval)
piIse <- ise(trueVal, xevalRes)
lines(xeval, xevalRes, col = "blue", lty = "dashed")
res <- rbind( bw = c(cvBwSel, thetaBwSel, pluginBwSel),
ise = c(cvIse, thIse, piIse) )
colnames(res) <- c("CV", "th", "PI")
res
```

PRDenEstC

---

Parzen–Rosenblatt density estimator.

Description

Parzen–Rosenblatt 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**


**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**

```r
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


See Also

thumbBw, pluginBw.
Examples

```r
size <- 200
sigma <- 0.25
deg <- 1
kernel <- EvaK
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))

est <- function(bw, dat, x) return(locPolSmoootherC(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)
lines(xeval, xevalRes, col = "blue", lty = "dashed")

xevalRes <- est(pibwSel, d, xeval)
piise <- ise(trueVal, xevalRes)
lines(xeval, xevalRes, col = "blue")

res <- rbind(c(cvBwSel, thBwSel, pibwSel),
ise = c(cvise, thise, piise))
colnames(res) <- c("CV", "th", "PI")
res
```

Description

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

Usage

```r
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.
**simpleSmoothers**

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**
simpleSmoothersC(x, y, xeval, bw, kernel, weig = rep(1, length(y)))
simpleSqSmoothersC(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.
simpleSqSmoothersC 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.
...


**thumbBw**

**Author(s)**

Jorge Luis Ojeda Cabrera.

**See Also**

PRDenEstC, Kernel characteristics

**Examples**

```r
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)$den
ssDen <- simpleSmotherC(x, y, xeval, bw, kernel)$reg
all(abs(prDen-ssDen)<1e-15)
## x kern. aver. should be f(x)*R2(K) aprox.
s2Den <- simpleSgsmotherC(x,y, xeval, bw, kernel)$reg
summary( abs(prDen*RK(kernel)-s2Den) )
summary( abs(1*RK(kernel)-s2Den) )
## x kern. aver. should be f(x)*R2(K) aprox.
for(n in c(1000,1e4,1e5))
{
  s2D <- simpleSgsmotherC(runif(n), rep(1,n), xeval, bw, kernel)$reg
  cat("\n",n,"\n")
  print( summary( abs(1*RK(kernel)-s2D)) )
}
```

**Description**

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

**Usage**

```r
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

See Also
regCVBwSelC, pluginBw.

Examples
```r
type <- 200
sigma <- 0.25
deg <- 1
kernel <- Epak
x <- runif(type, min = 0, max = 100)
regFun <- function(x) x^3
y <- regFun(x) + rnorm(x, sd = sigma)
d <- data.frame(x, y)
cVwSel <- regCVBwSelC(d$x, d$y, deg, kernel, interval = c(0, 0.25))

thBwSel <- thumbBw(d$x, d$y, deg, kernel)
pluginBwSel <- pluginBw(d$x, d$y, deg, kernel)
```
est <- function(bw, dat, x) return(locPolSmotherC(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

*Topic nonparametric
  bivNPest, 2
  compKernVals, 3
  denCVBwSelC, 5
  equivKernel, 6
  KernelChars, 7
  kernelCte, 9
  Kernels, 10
  locCteWeights, 11
  locpol, 12
  locpolSmoothers, 15
  pluginBw, 17
  PRDenEstC, 18
  regCVBwSelC, 20
  selKernel, 21
  simpleSmoothers, 22
  thumbBw, 23

+Topic smooth
  bivNPest, 2
  compKernVals, 3
  denCVBwSelC, 5
  equivKernel, 6
  KernelChars, 7
  kernelCte, 9
  Kernels, 10
  locCteWeights, 11
  locpol, 12
  locpolSmoothers, 15
  pluginBw, 17
  PRDenEstC, 18
  regCVBwSelC, 20
  selKernel, 21
  simpleSmoothers, 22
  thumbBw, 23
  bivReg (bivNPest), 2
  biweigK (Kernels), 10
  bkde, 19
  bw.nrd0, 5, 19
  compDerEst (thumbBw), 23
  compKernVals, 3
  Compute kernel values, 8, 9
  Compute kernel values (compKernVals), 3
  computeK4 (compKernVals), 3
  computeMu (compKernVals), 3
  computeMu0 (compKernVals), 3
  computeRK (compKernVals), 3
  confInterval (locpol), 12
  CosK (Kernels), 10
  cteNuK, 7
  cteNuK (kernelCte), 9
  denCVBwSelC, 5, 19
  density, 19
  dom (KernelChars), 7
  dpik, 5, 19
  Epa2K (Kernels), 10
  EpaK (Kernels), 10
  epaK2d (bivNPest), 2
  equivKernel, 6
  fitted.locpol (locpol), 12
  gauK2d (bivNPest), 2
  gaussK (Kernels), 10
  gaussK1f (Kernels), 10
  integrate, 4, 9, 10
  K4 (KernelChars), 7
  Kconvol (compKernVals), 3
  Kernel characteristics, 4
  Kernel characteristics (KernelChars), 7
  KernelChars, 7

adjNuK, 7
adjNuK (kernelCte), 9

bivDens (bivNPest), 2
bivNPest, 2
<table>
<thead>
<tr>
<th>Function</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>kernelCte, 9</td>
<td></td>
</tr>
<tr>
<td>Kernels, 4–6, 8, 9, 10, 11–15, 17, 19, 20, 22</td>
<td></td>
</tr>
<tr>
<td>ksmooth, 14, 16</td>
<td></td>
</tr>
<tr>
<td>locCteSmotherC (locpolSmoothers), 15</td>
<td></td>
</tr>
<tr>
<td>locCteWeights, 11</td>
<td></td>
</tr>
<tr>
<td>locCteWeightsC (locCteWeights), 11</td>
<td></td>
</tr>
<tr>
<td>locCuadSmotherC (locpolSmoothers), 15</td>
<td></td>
</tr>
<tr>
<td>locLinSmotherC (locpolSmoothers), 15</td>
<td></td>
</tr>
<tr>
<td>locLinWeightsC (locCteWeights), 11</td>
<td></td>
</tr>
<tr>
<td>locpol, 12, 12</td>
<td></td>
</tr>
<tr>
<td>locPolSmotherC (locpolSmoothers), 15</td>
<td></td>
</tr>
<tr>
<td>locpolSmoothers, 15</td>
<td></td>
</tr>
<tr>
<td>locPolWeights (locCteWeights), 11</td>
<td></td>
</tr>
<tr>
<td>locpoly, 14, 16</td>
<td></td>
</tr>
<tr>
<td>locWeightsEval (locCteWeights), 11</td>
<td></td>
</tr>
<tr>
<td>locWeightsEvalC (locCteWeights), 11</td>
<td></td>
</tr>
<tr>
<td>loess, 14, 16</td>
<td></td>
</tr>
<tr>
<td>looLocPolSmootherC (locpolSmoothers), 15</td>
<td></td>
</tr>
<tr>
<td>maybeBwSel (bivNPest), 2</td>
<td></td>
</tr>
<tr>
<td>mu0K, 10, 11</td>
<td></td>
</tr>
<tr>
<td>mu0K (KernelChars), 7</td>
<td></td>
</tr>
<tr>
<td>mu2K (KernelChars), 7</td>
<td></td>
</tr>
<tr>
<td>persp, 2</td>
<td></td>
</tr>
<tr>
<td>plot.bivNPest (bivNPest), 2</td>
<td></td>
</tr>
<tr>
<td>plot.locpol (locpol), 12</td>
<td></td>
</tr>
<tr>
<td>plotBivNPestOpts (bivNPest), 2</td>
<td></td>
</tr>
<tr>
<td>pluginBw, 10, 17, 20, 24</td>
<td></td>
</tr>
<tr>
<td>PRDenEstC, 18, 23</td>
<td></td>
</tr>
<tr>
<td>predict.bivNPest (bivNPest), 2</td>
<td></td>
</tr>
<tr>
<td>print.locpol (locpol), 12</td>
<td></td>
</tr>
<tr>
<td>QuartK (Kernels), 10</td>
<td></td>
</tr>
<tr>
<td>RdK (KernelChars), 7</td>
<td></td>
</tr>
<tr>
<td>regCVBwSelC, 10, 18, 20, 24</td>
<td></td>
</tr>
<tr>
<td>residuals.locpol (locpol), 12</td>
<td></td>
</tr>
<tr>
<td>RK, 4, 10, 11</td>
<td></td>
</tr>
<tr>
<td>RK (KernelChars), 7</td>
<td></td>
</tr>
<tr>
<td>selKernel, 21</td>
<td></td>
</tr>
<tr>
<td>simpleSmotherC (simpleSmoothers), 22</td>
<td></td>
</tr>
<tr>
<td>simpleSmoothers, 22</td>
<td></td>
</tr>
<tr>
<td>simpleSqsMotherC (simpleSmoothers), 22</td>
<td></td>
</tr>
<tr>
<td>sqk (Kernels), 10</td>
<td></td>
</tr>
<tr>
<td>summary.locpol (locpol), 12</td>
<td></td>
</tr>
<tr>
<td>thumbBw, 18, 20, 23</td>
<td></td>
</tr>
</tbody>
</table>