Package ‘fDKDE’

August 26, 2015

Type Package

Title Compute deconvolution kernel density estimator and its bandwidths

Version 1.0

Date 2015-06-09

Author Aurore Delaigle

Maintainer Tianying Wang <tianying@stat.tamu.edu>

Description This package illustrates how to use the functions for computing the deconvolution kernel density estimator and its bandwidths.

License GPL-2

R topics documented:

`fDKDE-package` *What the package does (short line) Compute deconvolution kernel density estimator and its bandwidths*

Description

This package illustrates how to use the functions for computing the deconvolution kernel density estimator and its bandwidths.

Details

Package:	fDKDE
Type:	Package
Version:	1.0
Date:	2015-06-09
License:	GPL>=2

Author(s)

Aurore Delaigle Maintainer: Tianying Wang <tianying@stat.tamu.edu>

References

Delaigle, A. and I. Gijbels (2002). Estimation of integrated squared density derivatives from a contaminated sample, Journal of the Royal Statistical Society, B, 64, 869-886. Delaigle, A. and I. Gijbels (2004). Practical bandwidth selection in deconvolution kernel density estimation, Computational Statistics and Data Analysis, 45, 249 - 267

Examples

```
#Noise to signal ratio=varU/varX
NSR=0.2

#Sample size
n=500

#Generate data from a normal mixture
X=rnorm(n,5,.4);
X2=matrix(rnorm(n*n,2,1),nrow=n,ncol=n,byrow=TRUE);

pmix=0.75;
tmp=matrix(runif(n,0,1),nrow=1,ncol=n,byrow=TRUE);
X[which(tmp<pmix)]=X2[which(tmp<pmix)];

#Specify error distribution (normal or Laplace in this case) and generate data from this error distribution
errortype="Lap";
#normal case
if (errortype=="norm")
{sigU=sqrt(NSR*var(X));
 U=rnorm(n,0,sigU);
 varU=sigU^2;}

#Laplace case
if (errortype=="Lap")
{sigU=sqrt(NSR*var(X)/2);
 varU=2*sigU^2;
 U=rlap(sigU,1,n);}

#Contaminated data
W=as.vector(X+U);
outcome<-fDKDE(W,errortype,NSR,n,varU,sigU)
```

CVdeconv *Compute CV bandwidth*

Description

compute CV bandwidth for kernel deconvolution estimator as in Stefanski and Carroll (1990) Stefanski, L., Carroll, R.J. (1990). Deconvolutingkernel density estimators. Statistics 2, 169<81>0<83>2184.Delaigle, A. and I. Gijbels (2004). Practical bandwidth selection in deconvolution kernel density estimation, Computational Statistics and Data Analysis, 45, 249-267

Usage

```
CVdeconv (W,errortype,sigU)
```

Arguments

W	observed variables (vector of contaminated data)
errortype	"Lap" or "norm", which means error will follow normal or Laplace distribution.
sigU	standard deviation of errors U_i, parameter of Laplace or normal errors used only to define phiU.

Details

The kernel you use must be the same as the kernel defined in the function fdecUknown.m

If you change the kernel you have to chage muK2, RK and the range of t-values (these must correspond to the support of phiK)

In case of multiple bandwidth solutions, by default this code takes the largest solution: you can change this to your preferred way of breaking ties. Often if you plot CV you will see that the first few solutions seem unreasonable (CV fluctuates widely). You can take the first minimum that looks reasonable. The outcome is the bandwidth generated by CV.

Author(s)

Aurore Delaigle

References

Stefanski, L., Carroll, R.J. (1990). Deconvolutingkernel density estimators. Statistics 2, 169<81>0<83>2184.Delaigle, A. and I. Gijbels (2004). Practical bandwidth selection in deconvolution kernel density estimation, Computational Statistics and Data Analysis, 45, 249-267

Examples

```
#Noise to signal ratio=varU/varX
NSR=0.2

#Sample size
n=500

#Generate data from a normal mixture
X=rnorm(n,5,.4);
```

```
X2=matrix(rnorm(n*n,2,1),nrow=n,ncol=n,byrow=TRUE);

pmix=0.75;
tmp=matrix(runif(n,0,1),nrow=1,ncol=n,byrow=TRUE);
X[which(tmp<pmix)]=X2[which(tmp<pmix)];
sigU=sqrt(NSR*var(X));
U=rnorm(n,0,sigU);
varU=sigU^2
W=as.vector(X+U);
hCV=CVdeconv(W,"norm",sigU)
```

`fdecUknown` *Compute the deconvolution kernel density estimator*

Description

Compute the deconvolution kernel density estimator

Usage

```
fdecUknown(xx,W,h,errortype,sigU)
```

Arguments

`xx`	vector of x-values where to compute the deconvolution kernel density estimator
`W`	contaminated sample
`h`	bandwidth
`errortype`	"Lap" for Laplace errors and "norm" for normal errors.
`sigU`	parameter of Laplace or normal errors used only to define phiU.

Details

rescale: to rescale the estimator so that it integrates to 1 after the negative parts have been truncated to zero Rescaling requires xx to be a fine grid of equispaced x-values that covers the whole range of x-values where the estimated density is significantly non zero. In this case dx=distance between two neighbour points in the xx grid.

The outcome is the DKDE estimator.

Author(s)

Aurore Delaigle

Examples

```
NSR=0.2

#Sample size
n=500

#Generate data from a normal mixture
X=rnorm(n,5,.4);
X2=matrix(rnorm(n*n,2,1),nrow=n,ncol=n,byrow=TRUE);
```

```
pmix=0.75;
tmp=matrix(runif(n,0,1),nrow=1,ncol=n,byrow=TRUE);
X[which(tmp<pmix)]=X2[which(tmp<pmix)];
errortype="Lap";
if (errortype=="Lap")
{sigU=sqrt(NSR*var(X)/2);
varU=2*sigU^2;
U=rlap(sigU,1,n);}

#Contaminated data
W=as.vector(X+U);
xx=seq(-2,8,0.1);
dx=xx[2]-xx[1];

#PI bandwidth of Delaigle and Gijbels
hPI=PI_deconvUknownth4(W,errortype,varU,sigU);

#DKDE estimator without rescaling (density does not integrate exactly to 1)
y=fdecUknown(xx,W,hPI,errortype,sigU);
#With rescaling: here xx must be equispaced and must cover the range where the estimated density is significan
y2=fdecUknown(xx,W,hPI,errortype,sigU,dx);
```

fDKDE *Compute deconvolution kernel density estimator and its bandwidths*

Description

This function illustrates how to use the functions for computing the deconvolution kernel density estimator and its bandwidths.

Usage

```
fDKDE (W,errortype,NSR,n,varU,sigU)
```

Arguments

W	observed variables (vector of contaminated data)
errortype	"Lap" or "norm", which means error will follow normal or Laplace distribution.
NSR	Noise to signal ratio=varU/varX.
n	sample size
varU	variance of errors U_i
sigU	standard deviation of errors U_i, parameter of Laplace or normal errors used only to define phiU.

Details

In outcomes, PI_bandwidth is PI bandwidth of Delaigle and Gijbels; CV_bandwidth is CV bandwidth of Stefanski and Carroll; DKDE_nonrescaled is DKDE estimator without rescaling (density does not integrate exactly to 1); DKDE_rescaled is rescaled DKDE estimator; normal_bandwidth is using normal reference bandwidth and standard normal kernel; naive_KDE is naive KDE estimator that ignores the error.

Author(s)

Aurore Delaigle

References

Matlab code from the website http://www.ms.unimelb.edu.au/~aurored/links.html#Code

Examples

```
#Noise to signal ratio=varU/varX
NSR=0.2

#Sample size
n=500

#Generate data from a normal mixture
X=rnorm(n,5,.4);
X2=matrix(rnorm(n*n,2,1),nrow=n,ncol=n,byrow=TRUE);

pmix=0.75;
tmp=matrix(runif(n,0,1),nrow=1,ncol=n,byrow=TRUE);
X[which(tmp<pmix)]=X2[which(tmp<pmix)];

#Specify error distribution (normal or Laplace in this case) and generate data from this error distribution
errortype="Lap";
#normal case
if (errortype=="norm")
{sigU=sqrt(NSR*var(X));
 U=rnorm(n,0,sigU);
 varU=sigU^2;}

#Laplace case
if (errortype=="Lap")
{sigU=sqrt(NSR*var(X)/2);
 varU=2*sigU^2;
 U=rlap(sigU,1,n);}

#Contaminated data
W=as.vector(X+U);
outcome<-fDKDE(W,errortype,NSR,n,varU,sigU)
```

`outerop` *Calculate an outer operation on two vectors*

Description

Calculates resultant matrix when the OPERATOR is applied to all combinations of the elements of vector A and the elements of vector B e.g. the outer product of A and B is outerop(A,B,"*"), the outer sum of A and B is outerop(A,B,"+"). If OPERATOR is omitted "+" is assumed this function is equivalent to the APL language's circle.dot operation. e.g. in APL Ao.*B is the outer product of A and B Copyright Murphy O'Brien 2005 all rights unreserved

Usage

```
outerop (a,b,operator)
```

Arguments

a	real number
b	real number
operator	string

Author(s)

Aurore Delaigle

Examples

```
xx1<-seq(-2,8,0.1);
xx2=seq(-5,5,0.1);
outerop(xx1,xx2,"+")
```

PI_deconvUknownth4 *Compute 2-stage plug-in bandwidth for kerndel deconvolution estimator*

Description

Compute 2-stage plug-in bandwidth for kerndel deconvolution estimator as in: Delaigle, A. and I. Gijbels (2002). Estimation of integrated squared density derivatives from a contaminated sample, Journal of the Royal Statistical Society, B, 64, 869-886. Delaigle, A. and I. Gijbels (2004). Practical bandwidth selection in deconvolution kernel density estimation, Computational Statistics and Data Analysis, 45, 249 - 267

Usage

```
PI_deconvUknownth4 (W,errortype,varU,sigU)
```

Arguments

W	vector of contaminated data
errortype	"Lap" for Laplace errors and "norm" for normal errors.
varU	variance of errors U_i
sigU	standard deviation of errors U_i, parameter of Laplace or normal errors used only to define phiU.

Details

The outcome is the 2-stage plug-in bandwidth for kerndel deconvolution estimator.

Note

The range of t-values -1 and 1 correspond to the support of phiK. If you change phiK and take a kernel for which phiK is not supported on [-1,1] you have to change -1 and 1 accordingly.

Author(s)

Aurore Delaigle

References

Delaigle, A. and I. Gijbels (2002). Estimation of integrated squared density derivatives from a contaminated sample, Journal of the Royal Statistical Society, B, 64, 869-886. Delaigle, A. and I. Gijbels (2004). Practical bandwidth selection in deconvolution kernel density estimation, Computational Statistics and Data Analysis, 45, 249 - 267

Examples

```
NSR=0.2
n=500
X=rnorm(n,5,.4);
X2=matrix(rnorm(n*n,2,1),nrow=n,ncol=n,byrow=TRUE);
pmix=0.75;
tmp=matrix(runif(n,0,1),nrow=1,ncol=n,byrow=TRUE);
X[which(tmp<pmix)]=X2[which(tmp<pmix)];
sigU=sqrt(NSR*var(X)/2);
varU=2*sigU^2;
U=rlap(sigU,1,n);
errortype="Lap";
W=as.vector(X+U);
#PI bandwidth of Delaigle and Gijbels
hPI=PI_deconvUknownth4(W,errortype,varU,sigU);
```

`rlap` *Generate a matrix of size n1 x n2 from a Laplace(szC)*

Description

Generate a matrix of size n1 x n2 from a Laplace(szC)

Usage

```
rlap(szC,n1,n2)
```

Arguments

`szC`	real number
`n1`	real number
`n2`	real number

Author(s)

Aurore Delaigle

Examples

```
#Noise to signal ratio=varU/varX
NSR=0.2

#Sample size
n=500

#Generate data from a normal mixture
X=rnorm(n,5,.4);
X2=matrix(rnorm(n*n,2,1),nrow=n,ncol=n,byrow=TRUE);
pmix=0.75;
tmp=matrix(runif(n,0,1),nrow=1,ncol=n,byrow=TRUE);
X[which(tmp<pmix)]=X2[which(tmp<pmix)];

sigU=sqrt(NSR*var(X)/2);
 varU=2*sigU^2;
 U=rlap(sigU,1,n)
```

---

truedens   *Compute true density of X*

Description

Compute true density of X

Usage

```
truedens (xx)
```

Arguments

xx	real number

Author(s)

Aurore Delaigle

Examples

```
xx=seq(-2,8,0.1);
pmix=0.75;
truedens(xx)
```

Index