Package 'fDKDEheterosc'

June 24, 2015

Type Package

Title Computing the deconvolution kernel density estimator and its bandwidths in the heteroscedastic case

Version 1.0

Date 2015-06-12

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 in the heteroscedastic case.

License GPL-2

R topics documented:

`fDKDEheterosc-package` *Computing the deconvolution kernel density estimator and its bandwidths in the heteroscedastic case*

Description

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

Details

Package:	fDKDEheterosc
Type:	Package
Version:	1.0
Date:	2015-06-12
License:	GPL>=2

Author(s)

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

References

Delaigle, A. and Meister, A. (2008). Density estimation with heteroscedastic error. Bernoulli, 14, 562-579

Examples

```
#Sample size
n=200

mu1=-3;
mu2=2;
sig1=1;
sig2=1;

#Generate data from a normal mixture
X=rnorm(n,mu1,sig1);
X2=rnorm(n,mu2,sig2);

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 distributions (normal or Laplace in this case) and generate data from these error distributions
errortype="norm";

sigU=0.6;
#Parameters of the Laplace distribution (one parameter oer observation, so sigmaj is of length n)
sigmaj=sigU*sqrt(1.0+(1:n)/n)*sqrt(0.5);
if (errortype=="norm")
{#normal case
  U=rep(1,n);
  for (i in 1:n)
    U[i]=rnorm(1,0,sigmaj[i]);}


if (errortype=="Lap")
{#Laplace case
  U=rep(1,n);
  for (i in 1:n)
```

```
    U[i]=rlap(sigmaj[i],1,1);}

#Contaminated data
W=as.vector(X+U);
outcome<-fDKDEheterosc(W,errortype,sigmaj,n)
```

---

fdecUknownhet *Compute the deconvolution kernel density estimator*

Description

Compute the deconvolution kernel density estimator when the errors are heteroscedastic, as in Delaigle, A. and Meister, A. (2008). Density estimation with heteroscedastic error. Bernoulli, 14, 562-579

Usage

```
fdecUknownhet (xx,W,h,errortype,sigUj)
```

Arguments

xx	vector of x-values where to compute the deconvolution kernel density estimator
W	contaminated sample
h	bandwidth
errortype	"Lap" for the case where the error densities are all Laplace densities and "norm" for the case where the error densities are all normal.
sigUj	vector of length n which contains the parameters of each of the n 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 deconvolution kernel density estimator when the errors are heteroscedastic.

Author(s)

Aurore Delaigle

References

Delaigle, A. and Meister, A. (2008). Density estimation with heteroscedastic error. Bernoulli, 14, 562-579

Examples

```
#Sample size
n=200

mu1=-3;
mu2=2;
sig1=1;
sig2=1;

#Generate data from a normal mixture
X=rnorm(n,mu1,sig1);
X2=rnorm(n,mu2,sig2);

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 distributions (normal or Laplace in this case) and generate data from these error distributions
errortype="norm";

sigU=0.6;
#Parameters of the Laplace distribution (one parameter oer observation, so sigmaj is of length n)
sigmaj=sigU*sqrt(1.0+(1:n)/n)*sqrt(0.5);
if (errortype=="norm")
{#normal case
U=rep(1,n);
for (i in 1:n)
U[i]=rnorm(1,0,sigmaj[i]);}
#Contaminated data
W=as.vector(X+U);
varW=var(W);

#Grid where to estimate the true density
xx=seq(-8,7,0.1);
dx=xx[2]-xx[1];
#PI bandwidth of Delaigle and Gijbels
hPI=PI_deconvUknownth4het(W,errortype,sigmaj);

#DKDE estimator without rescaling (density does not integrate exactly to 1)
y=fdecUknownhet(xx,W,hPI,errortype,sigmaj);

#With rescaling: here xx must be equispaced and must cover the range where the estimated density is significan
y2=fdecUknownhet(xx,W,hPI,errortype,sigmaj,dx);
```

`fDKDEheterosc` *Computing the deconvolution kernel density estimator and its bandwidths in the heteroscedastic case*

Description

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

Usage

```
fDKDEheterosc (W,errortype,sigmaj,n)
```

Arguments

W	vector of contaminated sample
errortype	need to be either "norm" or "Lap", which means error will follow normal or Laplace distribution.
sigmaj	the i-th element in sigmaj is used to generate i-th element in error (U).
n	sample size.

Details

In outcomes, PI_bandwidth is PI bandwidth of Delaigle and Gijbels; 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

```
#Sample size
n=200

mu1=-3;
mu2=2;
sig1=1;
sig2=1;

#Generate data from a normal mixture
X=rnorm(n,mu1,sig1);
X2=rnorm(n,mu2,sig2);

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 distributions (normal or Laplace in this case) and generate data from these error distributions
errortype="norm";

sigU=0.6;
#Parameters of the Laplace distribution (one parameter oer observation, so sigmaj is of length n)
sigmaj=sigU*sqrt(1.0+(1:n)/n)*sqrt(0.5);
if (errortype=="norm")
{#normal case
  U=rep(1,n);
  for (i in 1:n)
```

```
    U[i]=rnorm(1,0,sigmaj[i]);}


if (errortype=="Lap")
{#Laplace case
  U=rep(1,n);
  for (i in 1:n)
    U[i]=rlap(sigmaj[i],1,1);}

#Contaminated data
W=as.vector(X+U);
outcome<-fDKDEheterosc(W,errortype,sigmaj,n)
```

`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_deconvUknownth4het *Compute 2-stage plug-in bandwidth for heteroscedastic kerndel deconvolution estimator*

Description

Compute 2-stage plug-in bandwidth for heteroscedastic kerndel deconvolution estimator as in: Delaigle, A. and Meister, A. (2008). Density estimation with heteroscedastic error. Bernoulli, 14, 562-579

Usage

```
PI_deconvUknownth4het (W,errortype,sigUj)
```

Arguments

W	vector of contaminated sample
errortype	need to be either "norm" or "Lap", which means error will follow normal or Laplace distribution.
sigmaj	the i-th element in sigmaj is used to generate i-th element in error (U).

Details

The outcome is the 2-stage plug-in bandwidth for heteroscedastic 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 Meister, A. (2008). Density estimation with heteroscedastic error. Bernoulli, 14, 562-579

Examples

```
#Sample size
n=200

mu1=-3;
mu2=2;
sig1=1;
sig2=1;

#Generate data from a normal mixture
X=rnorm(n,mu1,sig1);
X2=rnorm(n,mu2,sig2);
```

```
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 distributions (normal or Laplace in this case) and generate data from these error distributions
errortype="norm";

sigU=0.6;
#Parameters of the Laplace distribution (one parameter oer observation, so sigmaj is of length n)
sigmaj=sigU*sqrt(1.0+(1:n)/n)*sqrt(0.5);
if (errortype=="norm")
{#normal case
U=rep(1,n);
for (i in 1:n)
U[i]=rnorm(1,0,sigmaj[i]);}
#Contaminated data
W=as.vector(X+U);
varW=var(W);

#PI bandwidth of Delaigle and Gijbels
hPI=PI_deconvUknownth4het(W,errortype,sigmaj);
```

`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,nq,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;
n=200
mu1=-3;
mu2=2;
sig1=1;
sig2=1;
truedens(xx)
```

Index