Package 'lpme'

October 31, 2017

Type Package

Title Nonparametric Estimation of Measurement Error Models

Version 1.1.1

Date 2017-10-31

Author Haiming Zhou <zhouh@niu.edu> and Xianzheng Huang <huang@stat.sc.edu>

Maintainer Haiming Zhou <zhouh@niu.edu>

Description Provide nonparametric methods for mean regression model, modal regression and conditional density estimation in the presence/absence of measurement error. Bandwidth selection is also provided for each method.

License GPL (>= 2)

Depends R (>= 3.0.2)

Imports Rcpp (>= 0.11.1), decon, flexmix, splines, locpol

LinkingTo Rcpp, RcppArmadillo (>= 0.4.300.0)

NeedsCompilation yes

Repository CRAN

Date/Publication 2017-10-31 21:31:42 UTC

R topics documented:

densityreg *Conditional Density Estimation with Covariate Measurement Error*

Description

This function provides nonparametric estimators (Huang and Zhou, 2018) for the density of a response conditioning on an error-prone covariate. The corresponding estimators in the absence of measurement error are also provided.

Usage

```
densityreg(Y, W, bw, xgrid=NULL, ygrid = NULL, sig=NULL, K1 = "Gauss",
           K2 = "Gauss", mean.estimate = NULL, spline.df = 5)
```

Arguments

Y	an n by 1 response vector.
W	an n by 1 predictor vector.
bw	bandwidth.
xgrid	the grid values in x-axis to estimate the conditional density.
ygrid	the grid values in y-axis to estimate the conditional density.
sig	standard deviation of the measurement error; sig=NULL returns the naive estimators ignoring measurement error.
K1	kernel function for X; choices include "Gauss" and SecOrder.
K2	kernel function for Y; choices include "Gauss" and SecOrder.
mean.estimate	method to estimate the naive mean function of Y given X; choices include "spline" and "kernel". If NULL, the methods 1 and 3 in the reference below are considered.
spline.df	the degrees of freedom for B-splines when mean.estimate="spline".

Value

The results include the grid points xgrid for X and ygrid for Y, the fitted density values fitxy.

Author(s)

Haiming Zhou <zhouh@niu.edu> and Xianzheng Huang <huang@stat.sc.edu>

References

Huang, X. and Zhou, H. (2018). Conditional density estimation with covariate measurement error. Submitted.

See Also

densityregbw

Examples

```
##########################################
## X - True covariates
## W - Observed covariates
## Y - individual response
rm(list=ls())
library(lpme)
## generate laplace
rlap=function (use.n, location = 0, scale = 1)
{
  location <- rep(location, length.out = use.n)
  scale <- rep(scale, length.out = use.n)
  rrrr <- runif(use.n)
  location - sign(rrrr - 0.5) * scale * (log(2) + ifelse(rrrr < 0.5, log(rrrr), log1p(-rrrr)))
}
## sample size:
n =100;
## Function f(y|x) to estimate#
mofx = function(x){ x }
sofx = function(x){ 0.5 }
fy_x=function(y,x) dnorm(y, mofx(x), sofx(x));
## Generate data
sigma_x = 1; X = rnorm(n, 0, sigma_x);
## Sample Y
Y = rep(0, n);
for(i in 1:n){
  Y[i] = mofx(X[i]) + rnorm(1, 0, sofx(X[i]));
}

## reliability ratio
lambda=0.7;
sigma_u  = sqrt(1/lambda-1)*sigma_x;
#W=X+rnorm(n,0,sigma_u);
W=X+rlap(n,0,sigma_u/sqrt(2));

##----- naive kernel density estimate  ----
fitbw = densityregbw(Y, W, K1 = "Gauss", K2 = "Gauss")
fhat1 = densityreg(Y, W, bw=fitbw$bw, K1 = "Gauss", K2 = "Gauss");

###----- naive kernel density estimate with mean adjustment  ----
#fitbw = densityregbw(Y, W, K1 = "Gauss", K2 = "Gauss",
#                     mean.estimate = "kernel")
#fhat2 = densityreg(Y, W, bw=fitbw$bw, K1 = "Gauss", K2 = "Gauss",
#                   mean.estimate = "kernel");

##----- proposed method without mean adjustment  ----
fitbw = densityregbw(Y, W, sig=sigma_u, K1="SecOrder", K2="Gauss")
fhat3 = densityreg(Y, W, bw=fitbw$bw, sig=sigma_u, K1="SecOrder", K2="Gauss");

##----- proposed method wit mean adjustment  ----
#fitbw = densityregbw(Y, W, sig=sigma_u, K1="SecOrder", K2="SecOrder",
#                     mean.estimate = "kernel")
```

```
#fhat4 = densityreg(Y, W, bw=fitbw$bw, sig=sigma_u, K1="SecOrder", K2="SecOrder",
#                   mean.estimate = "kernel");

par(mfrow=c(2,2))
plot(W,Y, col=2)
points(X,Y)
x0 = seq(0, 1, length.out = 3)
for(i in 1:length(x0)){
  plot(fhat1$ygrid, fy_x(fhat1$ygrid, x0[i]), "l", lwd="2", xlab = "y", ylab = "density",
       main = paste("Conditional density at x=", x0[i], sep=""), ylim=c(0,1.5));
  indx = which.min(abs(fhat1$xgrid-x0[i])) ## the index of xgrid that is closest to x0[i].
  lines(fhat1$ygrid, fhat1$fitxy[indx,], lwd=3, lty=2, col=1)
  #lines(fhat2$ygrid, fhat2$fitxy[indx,], lwd=3, lty=2, col=2)
  lines(fhat3$ygrid, fhat3$fitxy[indx,], lwd=3, lty=2, col=3)
  #lines(fhat4$ygrid, fhat4$fitxy[indx,], lwd=3, lty=2, col=4)
}
```

`densityregbw` *Bandwidth Selector for Conditional Density Estimation with Covariate Measurement Error*

Description

This function selects the bandwidth for the nonparametric estimators (Huang and Zhou, 2018) for the density of a response conditioning on an error-prone covariate. The corresponding methods in the absence of measurement error are also provided.

Usage

```
densityregbw(Y, W, h1=NULL, h2=NULL, sig = NULL,
             xinterval = quantile(W, probs=c(0.025, 0.975), names = FALSE),
             K1 = "Gauss", K2 = "Gauss", mean.estimate = NULL, spline.df = 5)
```

Arguments

`Y`	an n by 1 response vector.
`W`	an n by 1 predictor vector.
`h1`	bandwidth vector for h1; default is `NULL`, and h1 is chosen automatically. See the reference below for details. It is recommended to carefully specify a fine grid for h1.
`h2`	bandwidth vector for h2; default is `NULL`, and h2 is chosen automatically. See the reference below for details. It is recommended to carefully specify a fine grid for h2.
`sig`	standard deviation of the measurement error; `sig=NULL` returns the naive estimators ignoring measurement error.
`xinterval`	the interval within which the modes will be estimated.

K1	kernel function for X; choices include "Gauss" and SecOrder.
K2	kernel function for Y; choices include "Gauss" and SecOrder.
mean.estimate	method to estimate the naive mean function of Y given X; choices include "spline" and "kernel". If NULL, the methods 1 and 3 in the reference below are considered.
spline.df	the degrees of freedom for B-splines when mean.estimate="spline".

Value

The results include the bandwidth bw, grid values for h1 and grid values for h2.

Author(s)

Haiming Zhou <zhouh@niu.edu> and Xianzheng Huang <huang@stat.sc.edu>

References

Huang, X. and Zhou, H. (2018). Conditional density estimation with covariate measurement error. Submitted.

See Also

densityreg

Examples

```
##############################################
## X - True covariates
## W - Observed covariates
## Y - individual response
rm(list=ls())
library(lpme)
## generate laplace
rlap=function (use.n, location = 0, scale = 1)
{
  location <- rep(location, length.out = use.n)
  scale <- rep(scale, length.out = use.n)
  rrrr <- runif(use.n)
  location - sign(rrrr - 0.5) * scale * (log(2) + ifelse(rrrr < 0.5, log(rrrr), log1p(-rrrr)))
}
## sample size:
n =100;
## Function f(y|x) to estimate#
mofx = function(x){ x }
sofx = function(x){ 0.5 }
fy_x=function(y,x) dnorm(y, mofx(x), sofx(x));
## Generate data
sigma_x = 1; X = rnorm(n, 0, sigma_x);
## Sample Y
Y = rep(0, n);
for(i in 1:n){
```

```
  Y[i] = mofx(X[i]) + rnorm(1, 0, sofx(X[i]));
}

## reliability ratio
lambda=0.7;
sigma_u  = sqrt(1/lambda-1)*sigma_x;
#W=X+rnorm(n,0,sigma_u);
W=X+rlap(n,0,sigma_u/sqrt(2));

##----- naive kernel density estimate  ----
fitbw = densityregbw(Y, W, K1 = "Gauss", K2 = "Gauss")
fhat1 = densityreg(Y, W, bw=fitbw$bw, K1 = "Gauss", K2 = "Gauss");

###----- naive kernel density estimate with mean adjustment  ----
#fitbw = densityregbw(Y, W, K1 = "Gauss", K2 = "Gauss",
#                     mean.estimate = "kernel")
#fhat2 = densityreg(Y, W, bw=fitbw$bw, K1 = "Gauss", K2 = "Gauss",
#                   mean.estimate = "kernel");

##----- proposed method without mean adjustment  ----
fitbw = densityregbw(Y, W, sig=sigma_u, K1="SecOrder", K2="Gauss")
fhat3 = densityreg(Y, W, bw=fitbw$bw, sig=sigma_u, K1="SecOrder", K2="Gauss");

##----- proposed method wit mean adjustment  ----
#fitbw = densityregbw(Y, W, sig=sigma_u, K1="SecOrder", K2="SecOrder",
#                     mean.estimate = "kernel")
#fhat4 = densityreg(Y, W, bw=fitbw$bw, sig=sigma_u, K1="SecOrder", K2="SecOrder",
#                   mean.estimate = "kernel");

par(mfrow=c(2,2))
plot(W,Y, col=2)
points(X,Y)
x0 = seq(0, 1, length.out = 3)
for(i in 1:length(x0)){
  plot(fhat1$ygrid, fy_x(fhat1$ygrid, x0[i]), "l", lwd="2", xlab = "y", ylab = "density",
       main = paste("Conditional density at x=", x0[i], sep=""), ylim=c(0,1.5));
  indx = which.min(abs(fhat1$xgrid-x0[i])) ## the index of xgrid that is closest to x0[i].
  lines(fhat1$ygrid, fhat1$fitxy[indx,], lwd=3, lty=2, col=1)
  #lines(fhat2$ygrid, fhat2$fitxy[indx,], lwd=3, lty=2, col=2)
  lines(fhat3$ygrid, fhat3$fitxy[indx,], lwd=3, lty=2, col=3)
  #lines(fhat4$ygrid, fhat4$fitxy[indx,], lwd=3, lty=2, col=4)
}
```

---

meanreg *Local Polynomial Estimators for Nonparametric Mean Regression*

Description

This function provides both the DFC (Delaigle, Fan, and Carroll, 2009) and HZ (Huang and Zhou, 2017) local linear estimators for solving the errors-in-variables problem. The local linear estimator

in the absence of measurement error is also provided.

Usage

```
meanreg(Y, W, bw, xgrid=NULL, method="HZ", sig=NULL, error="laplace", FT_fu)
```

Arguments

Y	an n by 1 response vector.
W	an n by 1 predictor vector.
bw	bandwidth.
xgrid	the grid values to estimate the responses.
method	the method to be used; method="HZ" returns the estimator proposed by Huang and Zhou (2017); method="DFC" returns the estimator proposed by Delaigle, Fan, and Carroll (2009); method="naive" returns the local linear estimator ignoring measurement error.
sig	standard deviation of the measurement error; sig=NULL returns the naive estimators ignoring measurement error.
error	the distribution assumed for the measurement error; error="laplace" is for Laplace distribution; error="normal" is for Gaussian distribution; error="user" is for user-assumed distribution.
FT_fu	a function for the Fourier transform of density of error, which is required only when error="user".

Value

The results include the grid points xgrid for predictor and corresponding fitted responses yhat.

Author(s)

Haiming Zhou <zhouh@niu.edu> and Xianzheng Huang <huang@stat.sc.edu>

References

Huang, X. and Zhou, H. (2017). An alternative local polynomial estimator for the error-in-variables problem. *Journal of Nonparametric Statistics*, 29: 301-325.

Delaigle, A., Fan, J., and Carroll, R. (2009). A design-adaptive local polynomial estimator for the errors-in-variables problem. *Journal of the American Statistical Association*, 104: 348-359.

See Also

meanregbwSIMEX

Examples

```
###############################################
## X - True covariates
## W - Observed covariates
## Y - individual response
rm(list=ls())
library(lpme)

## sample size:
n =100;
## Function gofx(x) to estimate
gofx  = function(x){ 1/4*x + x^2/4 }
xgrid = seq(-2, 2, 0.02)

## Generate data
sigma_e  = 0.5;
sigma_x = 1; X = rnorm(n, 0, sigma_x);
## Sample Y
Y  = gofx(X) + rnorm(n, 0, sigma_e);
##------------------  method Based on X ---------------------------
ghat_X= meanreg(Y, X, 0.1, method="naive", xgrid=xgrid);

## reliability ratio
lambda=0.85;
sigma_u  = sqrt(1/lambda-1)*sigma_x;
print( sigma_x^2/(sigma_x^2 + sigma_u^2) );
W=X+rnorm(n,0,sigma_u);
#W=X+rlaplace(n,0,sigma_u/sqrt(2));

##------------------  method Based on W ---------------------------
ghat_W=meanreg(Y, W, 0.1, method="naive", xgrid=xgrid);

##------------------ JASA method -------------------------------------
h = 0.13;
ghat_JASA=meanreg(Y, W, h, method="DFC", sig=sigma_u,
                  error="laplace", xgrid=xgrid);

##------------------ Our method -------------------------
ghat_NEW=meanreg(Y, W, h, method="HZ", sig=sigma_u,
                 error="laplace", xgrid=xgrid);

## plots
plot(xgrid, gofx(xgrid), "l", main="Individual", lwd="2")
lines(xgrid, ghat_NEW$yhat, lty="dashed", col="2",lwd="3")
lines(xgrid, ghat_JASA$yhat, lty="dotted", col="3",lwd="3")
lines(xgrid, ghat_X$yhat, lty="dashed", col="4",lwd="2")
lines(xgrid, ghat_W$yhat, lty="dashed", col="5",lwd="3")
```

`meanregbwSIMEX` *Cross-Validation Bandwidth Selector Using SIMEX for Nonparametric Mean Regression*

Description

This function selects the bandwidth for both the DFC (Delaigle, Fan, and Carroll, 2009) and HZ (Huang and Zhou, 2017) estimators.

Usage

```
meanregbwSIMEX(Y, W, method="HZ", sig, error="laplace", k_fold=5, B=10,
               h1=NULL, h2=NULL, length.h=10, lconst=0.5, rconst=2, Wdiff=NULL)
```

Arguments

`Y`	an n by 1 response vector.
`W`	an n by 1 predictor vector.
`method`	the method to be used; `method="HZ"` uses the estimator proposed by Huang and Zhou (2017); `method="DFC"` uses the estimator proposed by Delaigle, Fan, and Carroll (2009). It currently does not support bandwidth selection for naive estimators.
`sig`	standard deviation of the measurement error.
`error`	the distribution assumed for the measurement error; `error="laplace"` is for Laplace distribution; `error="normal"` is for Gaussian distribution. It currently does not support user-assumed distribution.
`k_fold`	gives fold of cross-validation to be used; default is 2.
`B`	total number of cross-validation criteria to average over; defualt is 10.
`h1`	bandwidth vector for the first level error contamination; default is `NULL`, and h1 is chosen automatically. See Huang and Zhou (2017) for details.
`h2`	bandwidth vector for the second level error contamination; defualt is `NULL`, and h2 is chosen automatically. See Huang and Zhou (2017) for details.
`length.h`	number of grid points for each of h1 and h2; default is 10.
`lconst, rconst`	used to control the searching windows for bandwidths h1 and h2. For example, `seq(bw1*lconst,bw1*rconst,length.out=length.h)` is used to obtain bandwidth grid points for h1, where bw1 is an initial bandwidth; see Huang and Zhou (2017) for details of finding the initial bandwith.
`Wdiff`	an n by 1 vector of `(W1-W2)/2`, where W1, W2 are two replicated measurements; default is `NULL`, which indicates that the errors are generated from the assumed error distribution, otherwise, the errors are generated from Wdiff with replacement.

Value

The results include the bandwidth `bw`.

Author(s)

Haiming Zhou <zhouh@niu.edu> and Xianzheng Huang <huang@stat.sc.edu>

References

Huang, X. and Zhou, H. (2017). An alternative local polynomial estimator for the error-in-variables problem. *Journal of Nonparametric Statistics*, 29: 301-325.

Delaigle, A. and Hall, P. (2008). Using SIMEX for smoothing-parameter choice in errors-in-variables problems. *Journal of the American Statistical Association*, 103: 280-287.

See Also

meanreg

Examples

```
###############################################
## X - True covariates
## W - Observed covariates
## Y - individual response
rm(list=ls())
library(lpme)
## generate laplace
rlap=function (use.n, location = 0, scale = 1)
{
location <- rep(location, length.out = use.n)
scale <- rep(scale, length.out = use.n)
rrrr <- runif(use.n)
location-sign(rrrr-0.5)*scale*(log(2)+ifelse(rrrr<0.5, log(rrrr), log1p(-rrrr)))
}

## sample size:
n =100;
## Function gofx(x) to estimate
gofx  = function(x){ 2*x*exp(-10*x^4/81) }

## Generate data
sigma_e  = 0.2;
sigma_x = 1; X = rnorm(n, 0, sigma_x);
## Sample Y
Y  = gofx(X) + rnorm(n, 0, sigma_e);
## reliability ratio
lambda=0.85;
sigma_u  = sqrt(1/lambda-1)*sigma_x;
print( sigma_x^2/(sigma_x^2 + sigma_u^2) );
#W=X+rnorm(n,0,sigma_u);
W=X+rlap(n,0,sigma_u/sqrt(2));

#### SIMEX
#**Note: larger values for B and length.h are needed for accurate estimates.
#**e.g., k_fold=5, B=10, length.h=10 will be generally good.
hwNEW = meanregbwSIMEX(Y, W, method="HZ", sig=sigma_u, error="laplace", k_fold=2,
                       B=1, length.h=1)$bw
ghat_NEW = meanreg(Y, W, hwNEW , method="HZ", sig=sigma_u, error="laplace");
```

```
## plots
x = ghat_NEW$xgrid;
plot(x, gofx(x), "l", main="Individual", lwd="2")
lines(ghat_NEW$xgrid, ghat_NEW$yhat, lty="dashed", col="2",lwd="3")
```

`modereg` *Nonparametric Estimators for Nonparametric Mode Regression*

Description

This function provides the nonparametric estimators (Zhou and Huang, 2016; Zhou and Huang, 2018) for nonparametric modal regression. The corresponding estimators in the absence of measurement error are also provided.

Usage

```
modereg(Y, W, bw, xgrid=NULL, sig=NULL, nstart=4, p.order=0, maxiter = 500,
        tol=.Machine$double.eps^0.25, mesh=NULL, PLOT=FALSE, ...)
```

Arguments

`Y`	an n by 1 response vector.
`W`	an n by 1 predictor vector.
`bw`	bandwidth.
`xgrid`	the grid values to estimate the responses.
`sig`	standard deviation of the measurement error; `sig=NULL` returns the naive estimators ignoring measurement error.
`nstart`	the starting number of modes for each grid value.
`p.order`	the order of polynomial, up to 1; `p.order=0` returns local constant estimators and `p.order=1` returns local linear estimators.
`maxiter`	the maximum number of iterations performed for the mean shift algorithm if not converage.
`tol`	the deisered accurary (convergence tolerrance).
`mesh`	a matrix of initial mode points, where each row corresponds a mode in `(x,y)` coordinate; if `mesh=NULL`, it will be chosen automatically according to `xgrid` and `nstart`.
`PLOT`	a logical value indicating whether the estimated modes will be plotted.
`...`	further arguments to be passed to or from other methods.

Value

The results include the grid points `xgrid` for predictor, the number of modes for each grid `x.num`, the initial mesh points `mesh`, and corresponding fitted modes `mode`.

Author(s)

Haiming Zhou <zhouh@niu.edu> and Xianzheng Huang <huang@stat.sc.edu>

References

Zhou. H. and Huang, X. (2016). Nonparametric modal regression in the presence of measurement error. *Electronic Journal of Statistics*, 10: 3579-3620.

Zhou, H. and Huang, X. (2018). Bandwidth selection for nonparametric modal regression. *Communications in Statistics - Simulation and Computation*, in press.

See Also

moderegbwSIMEX,moderegbw

Examples

```
rm(list=ls())
library(lpme)

rlaplace=function (use.n, location = 0, scale = 1)
{
  location <- rep(location, length.out = use.n)
  scale <- rep(scale, length.out = use.n)
  rrrr <- runif(use.n)
  location - sign(rrrr - 0.5) * scale *
    (log(2) + ifelse(rrrr < 0.5, log(rrrr), log1p(-rrrr)))
}

## sample size:
n =100;
## Function m(x) to estimate#
gofx1  = function(x){ (x+x^2) }
gofx2  = function(x){ (x+x^2)-6 }
xgrid = seq(-2, 2, length.out=100);
ngrid = length(xgrid)

## Sample X
X = rnorm(n, 0, 1); sigma_x=1;
## Sample Y
Y = rep(0, n);
U = runif(n);
for(i in 1:n){
  if(U[i]<0.5){
    Y[i] = rnorm(1, gofx1(X[i]), 1);
  }else{
    Y[i] = rnorm(1, gofx2(X[i]), 1);
  }
}
## reliability ratio
lambda=0.9;
sigma_u  = sqrt(1/lambda-1)*sigma_x;
```

```
W=X+rlaplace(n,0,sigma_u/sqrt(2));

## mode estimates
hhxy = c(0.15, 1)
## Note you needs to use the following code to calculate bandwidth
## It is not run here due to the time constrain of runing examples.
#hhxy = moderegbwSIMEX(Y, W, method="CV-density", p.order=0,
#                      sig=sigma_u, B=5, length.h=10)$bw;
fit = modereg(Y, W, xgrid=xgrid, bw=hhxy, sig=sigma_u, p.order=0,
              PLOT=TRUE);

## Plot
plot(xgrid, gofx1(xgrid), "l", lwd="2", ylim=c(-9,7), xlim=c(-2,2));
lines(xgrid, gofx2(xgrid), "l", lwd="2");
points(rep(fit$xgrid,fit$x.num), fit$mode, col="3",lwd="2")
```

`moderegbw` *Cross-Validation Bandwidth Selector for Nonparametric Mode Regression*

Description

This function selects the bandwidth (Zhou and Huang, 2018) for the local polynomial estimators for nonparametric modal regression in the absence of measurement error.

Usage

```
moderegbw(Y, X, method="CV-density", p.order=0, h1=NULL, h2=NULL, nstart = 4,
          xinterval = quantile(X, probs=c(0.025, 0.975), names = FALSE),
          df=5, ncomp=5, nboot=5)
```

Arguments

`Y`	an n by 1 response vector.
`X`	an n by 1 predictor vector.
`method`	`method="CV-density"` is for density-based CV; `method="CV-mode"` is for mode-based CV; `metohd="bootstrap"` is for a bootstap method; see Zhou and Huang (2017) for details. For non-measurement error models, `method="CV-mode"` is recommended.
`p.order`	the order of polynomial, up to 1; `p.order=0` returns local constant estimators and `p.order=1` returns local linear estimators.
`h1`	bandwidth vector for h1; default is `NULL`, and h1 is chosen automatically. See Zhou and Huang (2017) for details. It is recommended to carefully specify a fine grid for h1.
`h2`	bandwidth vector for h2; default is `NULL`, and h2 is chosen automatically. See Zhou and Huang (2017) for details. It is recommended to carefully specify a fine grid for h2.

nstart	the starting number of modes for each grid value.
xinterval	the interval within which the modes will be estimated.
df	the degrees of freedom of splines used in the mixture normal regression for bootstrap method.
ncomp	the number of components used in the mixture normal regression for bootstrap method.
nboot	the number of bootstrap samples.

Value

The results include the bandwidth bw.

Author(s)

Haiming Zhou <zhouh@niu.edu> and Xianzheng Huang <huang@stat.sc.edu>

References

Zhou, H. and Huang, X. (2018). Bandwidth selection for nonparametric modal regression. *Communications in Statistics - Simulation and Computation*, in press.

See Also

moderegbwSIMEX,modereg

Examples

```
rm(list=ls())
library(lpme)
## sample size:
n =100;
## Function m(x) to estimate#
gofx1  = function(x){ (x+x^2) }
gofx2  = function(x){ (x+x^2)-6 }
xgrid = seq(-2, 2, length.out=100);
ngrid = length(xgrid)

## Sample X
X = rnorm(n, 0, 1); sigma_x=1;
## Sample Y
Y = rep(0, n);
U = runif(n);
for(i in 1:n){
  if(U[i]<0.5){
    Y[i] = rnorm(1, gofx1(X[i]), 1);
  }else{
    Y[i] = rnorm(1, gofx2(X[i]), 1);
  }
}
```

```
## mode estimates
h1ref = c(1.06*sd(X)*n^(-0.2));
h2ref = c(1.06*sd(Y)*n^(-0.2));
## In practice moer fine grids are desired.
hx = seq(h1ref*0.2, h1ref*1.5, length.out = 10);
hy = seq(h2ref*0.8, h2ref, length.out = 2);
hhxy = moderegbw(Y, X, method="CV-mode", p.order=0,
                 h1=hx, h2=hy)$bw;
fit = modereg(Y, X, xgrid=xgrid, bw=hhxy, p.order=0, PLOT=TRUE);

## Plot
plot(xgrid, gofx1(xgrid), "l", lwd="2", ylim=c(-9,7), xlim=c(-2,2));
lines(xgrid, gofx2(xgrid), "l", lwd="2");
points(rep(fit$xgrid,fit$x.num), fit$mode, col="3",lwd="2")
```

`moderegbwSIMEX` *Cross-Validation Bandwidth Selector Using SIMEX for Nonparametric Mode Regression*

Description

This function selects the bandwidth (Zhou and Huang, 2016) for the local polynomial estimators for nonparametric modal regression in the presence of measurement error.

Usage

```
moderegbwSIMEX(Y, W, method="CV-density", p.order=0, sig, B=5,
               h1=NULL, h2=NULL, length.h=10, CIregion=0.95, nstart=4)
```

Arguments

`Y`	an n by 1 response vector.
`W`	an n by 1 predictor vector.
`method`	`method="CV-density"` is for density-based CV; `method="CV-mode"` is for mode-based CV; see Zhou and Huang (2016) for details. For measurement error models, `method="CV-density"` is recommended.
`p.order`	the order of polynomial, up to 1; `p.order=0` returns local constant estimators and `p.order=1` returns local linear estimators.
`sig`	standard deviation of the measurement error.
`B`	total number of cross-validation criteria to average over; defualt is 5.
`h1`	bandwidth vector for h1; default is `NULL`, and h1 is chosen automatically. See Zhou and Huang (2017) for details. It is recommended to carefully specify a fine grid for h1.
`h2`	bandwidth vector for h2; default is `NULL`, and h2 is chosen automatically. See Zhou and Huang (2017) for details. It is recommended to carefully specify a fine grid for h2.

length.h	number of grid points for each of h1 and h2; default is 10.
CIregion	used to determine the interval on which the mode is estimated.
nstart	the starting number of modes for each grid value.

Value

The results include the bandwidth bw.

Author(s)

Haiming Zhou <zhouh@niu.edu> and Xianzheng Huang <huang@stat.sc.edu>

References

Zhou. H. and Huang, X. (2016). Nonparametric modal regression in the presence of measurement error. *Electronic Journal of Statistics*, 10: 3579-3620.

See Also

modereg, moderegbw

Examples

```
rm(list=ls())
library(lpme)

rlaplace=function (use.n, location = 0, scale = 1)
{
  location <- rep(location, length.out = use.n)
  scale <- rep(scale, length.out = use.n)
  rrrr <- runif(use.n)
  location - sign(rrrr - 0.5) * scale *
    (log(2) + ifelse(rrrr < 0.5, log(rrrr), log1p(-rrrr)))
}

## sample size:
n =100;
## Function m(x) to estimate#
gofx1  = function(x){ (x+x^2) }
gofx2  = function(x){ (x+x^2)-6 }
xgrid = seq(-2, 2, length.out=100);
ngrid = length(xgrid)

## Sample X
X = rnorm(n, 0, 1); sigma_x=1;
## Sample Y
Y = rep(0, n);
U = runif(n);
for(i in 1:n){
  if(U[i]<0.5){
    Y[i] = rnorm(1, gofx1(X[i]), 1);
```

```
  }else{
    Y[i] = rnorm(1, gofx2(X[i]), 1);
  }
}
## reliability ratio
lambda=0.9;
sigma_u  = sqrt(1/lambda-1)*sigma_x;
W=X+rlaplace(n,0,sigma_u/sqrt(2));

## mode estimates
hhxy = c(0.15, 1)
## Note you needs to use the following code to calculate bandwidth
## It is not run here due to the time constrain of runing examples.
#hhxy = moderegbwSIMEX(Y, W, method="CV-density", p.order=0,
#                      sig=sigma_u, B=5, length.h=10)$bw;
fit = modereg(Y, W, xgrid=xgrid, bw=hhxy, sig=sigma_u, p.order=0,
              PLOT=TRUE);

## Plot
plot(xgrid, gofx1(xgrid), "l", lwd="2", ylim=c(-9,7), xlim=c(-2,2));
lines(xgrid, gofx2(xgrid), "l", lwd="2");
points(rep(fit$xgrid,fit$x.num), fit$mode, col="3",lwd="2")
```

Index