Package 'QPdecon'

September 17, 2019

Type Package

Title An R Package for Density Deconvolution with Additive Measurement Errors using Quadratic Programming

Version 1.0

Date 2019-09-17

Author Ran Yang, D. W. Apley

Maintainer <ranyang2011@u.northwestern.edu><apley@northwestern.edu>

Description The purpose of the QPdecon package is to solve a deconvolution problem with regularization to estimate the density of X in the model of Y=X+Z with Z noise and given a sample of observations of Y. The optimization problem finds the discretized pdf of X to minimize a quadratic programming problem subject to various constraints. The constraints of nonnegativity and integrate-to-one are always recommnented while other constraints like monotonicity, convexity (over specified tail regions) and support can also be included if available. The package can be used to estimate the cdf when needed. The associated regularization parameter can be automatically determined by SURE method implemented in the QPdecon package, and the package also provides a simple graphical method using SURE plot to assist determining the value of the regularization parameter.

Imports quadprog

License GPL(>= 3)

NeedsCompilation no

R topics documented:

`QPdecon-package` *An R Package for Density Deconvolution*

Description

The purpose of the QPdecon package is to solve a deconvolution problem with regularization to estimate the density of X in the model Y=X+Z with Z noise and given a sample of observation of Y. The optimization problem finds the estimator $\hat{f}_X$ to minimize a quadratic programming objective function $\|\hat{f}_Y - Cf_X\|^2 + \lambda Q(f_X)$ subject to various constraints. The term of $Q(f_X)$ is a regularization penalty. The constraints of nonnegativity and integrate-to-one are always recommnented and are the default settings, while other constraints like monotonicity, convexity (over specified tail regions) and support can also be included when available. The package can be used to estimate the cdf when needed. The associated regularization parameter λ can be automatically determined by SURE method, and the package also provides a simple graphical method using SURE plot which plots the regularization penalty $Q(f_X)$ against a sequence of λ to help determine the value of the regularization parameter.

Details

There are two main functions in the QPdecon package, and they are `QPdecon()` and `SUREPlot()`. The first function implements the quadratic programming approach to solve a density deconvolution problem, while the second function offers a SURE plot for checking the associated regularization parameter. Another function `plot.QPdecon()` can be used to plot the density estimation results

Author(s)

Ran Yang, D. W. Apley

Maintainer: ranyang2011@u.northwester.edu

References

Ran Yang, Daniel Apley, Jeremy Staum and David Ruppert. Density Deconvolution with Additive Measurement Errors using Quadratic Programming.

See Also

`QPdecon` for QP estimation, `SUREPlot` for regularization parameter check using SURE-plot method, `plot.QPdecon` for plotting QP estimators

---

QPcdf *Create the CDF estimator*

Description

This is a function used to transform an object (PDF estimator) from the QPdecon function and returns a CDF estimator

Usage

```
QPcdf(obj,prob=NULL)
```

Arguments

`obj`	a QPdecon object
`prob`	if `prob!= NULL`, quantiles corresponding to the specified `prob` will be returned

See Also

QPdecon

Examples

```
n <- 500
X <- rgamma(n,5,1)
Z <- rnorm(n,0,sd=1.8)
Y <- X+Z # the observed sample of data
K <- 100
t1 <- proc.time()
L <- QPdecon(Y=Y,K=K,f_Z=1.8,lambda=0.02,reg="2Deriv",integr=TRUE,
nonneg=TRUE)
cdfEst = QPcdf(L,prob = c(0.1,0.5,0.75,0.9))
plot(cdfEst$x,cdfEst$cdf,type="l")
abline(h=0.1)
abline(v=cdfEst$quantiles[1])
abline(h=0.5)
abline(v=cdfEst$quantiles[2])
abline(h=0.75)
abline(v=cdfEst$quantiles[3])
```

QPdecon *Density deconvolution using quadratic programming*

Description

QPdecon is a function that solves a deconvolution problem to estimate the density of X in the model Y = X + Z with noise Z, given a sample of observations of Y. The deconvolution problem is formulated as a quadratic programming problem with regularization, and the function finds a discretized version of f_X to minimize the quadratic programming problem subject to various constraints

Usage

```
QPdecon(Y, Pdf = TRUE, K = NULL, f_Z, lambda = "SURE", reg = "2Deriv",
integr = TRUE, nonneg = TRUE, monotone = NULL,unimod = FALSE,
convex = NULL, support = NULL)
```

Arguments

Y	N-length vector of N observations of Y
Pdf	set Pdf=TRUE to obtain the pdf estimator, or set Pdf=FALSE to obtain the cdf estimator
K	number of discrete points at which the pdfs will be evaluated over the range of X and Y
f_Z	the pdf of the noise Z. For zero-mean Gaussian Z, f_Z is a constant that represents the standard deviation of Z. For any other noise distribution, f_Z should be a user-defined function that takes a scalar (say z) as an input argument and returns a scalar value for the noise density f_Z(z) at $Z = z$

lambda	To use a specific value for the regularization parameter, set lambda equal to the value. Otherwise, to have lambda chosen automatically (by the SURE method), set lambda ="SURE"
reg	regularization method can be set as reg = "Gauss", "2Deriv" or "Unif". reg="Gauss" sets the regularization to be $\|\hat{f}_Y - Cf_X\|^2$, where f_reg is a Gaussian distribution having the same mean and variance as X (which is determined internally, based on the sample variance of Y and the noise variance from f_Z); reg = "2Deriv", which is the default, penalizes the magnitude of the second-derivative of f_X; reg="Unif" sets f_reg as a uniform distribution over the range of Y
integr	if Pdf=TRUE, setting integr=TRUE enforces the integrate-to-one constraint on f_X; if Pdf=FALSE, i.e. to obtain cdf estimator, then setting integr=TRUE enforces the last element in the vector f_X to be 1
nonneg	set nonneg=TRUE to include the nonnegativity constraint that each element of f_X is nonnegative
monotone	set as a two-length vector monotone=c(l_mon,u_mon) to specify the monotonic regions. f_X is constrained to be monotonically increasing for x below l_mon and decreasing above u_mon. If monotone=NULL, no monotonicity constraints will be used. For only upper-tail monotonicity, specify l_mon as NA, and let u_mon equal the value after which the pdf is monotonically decreasing. Likewise, for lower-tail monotonicity, let l_mon equal the value below which the pdf is monotonically increassing, and set u_mon as NA
unimod	set to TRUE to enforce the pdf estimator is unimodal; not active for cdf estimator
convex	set as a two-length vector convex=c(l_con,u_con) to specify the convex regions. f_X is constrained to be convex for x below l_con and above u_con. If convex=NULL, no convexity constraints will be used. For only upper-tail convexity, specify l_con as NA, and let u_con equal the value after which the pdf is convex. Likewise, for lower-tail convexitity, let l_con equal the value below which the pdf is convex, and set u_con as NA
support	set as a two-length vector support=c(l_sup,u_sup) to specify the lower and upper end of the support; the default is support=c(min(Y),max(Y)); the lower and upper end can be specified to be -Inf or Inf, respectively

Value

The function returns an object of class QPdecon, which is composed of the following attributes:

x	input domain: K-lenth vector of evenly-spaced values in the observed range of Y
f_X	K-length vector of the estimated pdf or cdf of X, at values in x
SURE.value	value of SURE criterion
f_Y	K-length vector of the estimated pdf or cdf of Y from its histogram, evaluated at the values in x
lambda	automatically selected or user-specified value of regularization parameter
mymode	the automatically estimated mode location when unimod constraint is TRUE; otherwise, mymode="N/A"

Author(s)

Ran Yang, D. W. Apley

References

Ran Yang, Daniel Apley, Jeremy Staum and David Ruppert. Density Deconvolution with Additive Measurement Errors using Quadratic Programming.

See Also

`plot.QPdecon` for plotting the QP estimator

Examples

```
rm(list=ls())

X=rgamma(3000,5,1)
Z=rnorm(3000,0,sd=1.8)
Y=X+Z
f_Z<-function(z){dnorm(z,mean=0,sd=1.8)}
K=min(200,floor(3*sqrt(3000)))

#### QP estimation ####
QPobj=QPdecon(Y=Y,Pdf=TRUE,K=K,f_Z=f_Z,lambda=0.02,reg="2Deriv",integr=TRUE,
nonneg=TRUE)

#### Plot the estimation ####
plot(QPobj,histo=TRUE)
#### compare with the true pdf ####
lines(QPobj$x,dgamma(QPobj$x,5,1),lwd=1)
```

`SUREPlot` *SURE plot generation*

Description

This function returns a SURE plot to help determining the value of the regularization parameter λ in the QP objective function $\|\hat{f}_Y - Cf_X\|^2 + \lambda Q(f_X)$ used in the QPdecon function. The SURE plot is a plot of the SURE values (in log base 10 scale) for the optimization solution, as a function of λ, and the best λ value should be chosen just to the right of the elbow point of the SURE plot.

Usage

```
SUREPlot(Y, Pdf = TRUE, K = NULL, f_Z, lambda = NULL, reg = "2Deriv",
integr = TRUE,nonneg = TRUE,monotone = NULL,convex = NULL,support = NULL)
```

Arguments

`Y`	N-length vector of N observations of Y
`Pdf`	set `Pdf=TRUE` to obtain the pdf estimator, or set `Pdf=FALSE` to obtain the cdf estimator
`K`	number of discrete points at which the pdfs will be evaluated over the range of X and Y

`f_Z` the pdf of the noise Z. For zero-mean Gaussian Z, `f_Z` is a constant that represents the standard deviation of Z. For any other noise distribution, `f_Z` should be a user-defined function that takes a scalar (say z) as an input argument and returns a scalar value for the noise density f_Z(z) at $Z = z$

`lambda` To use a specific value for the regularization parameter, set `lambda` equal to the value. Otherwise, to have `lambda` chosen automatically (by the SURE method), set `lambda = NULL`

`reg` regularization method can be set as `reg = "Gauss", "2Deriv"` or `"Unif"`. `reg="Gauss"` sets the regularization to be $\|\hat{f}_Y - Cf_X\|^2$, where f_reg is a Gaussian distribution having the same mean and variance as X (which is determined internally, based on the sample variance of Y and the noise variance from `f_Z`); `reg = "2Deriv"`, which is the default, penalizes the magnitude of the second-derivative of `f_X`; `reg="Unif"` sets f_reg as a uniform distribution over the range of Y

`integr` if `Pdf=TRUE`, setting `integr=TRUE` enforces the integrate-to-one constraint on `f_X`; if `Pdf=FALSE`, i.e. to obtain cdf estimator, then setting `integr=TRUE` enforces the last element in the vector `f_X` to be 1

`nonneg` set `nonneg=TRUE` to include the nonnegativity constraint that each element of `f_X` is nonnegative

`monotone` set as a two-length vector `monotone=c(l_mon,u_mon)` to specify the monotonic regions. `f_X` is constrained to be monotonically increasing for `x` below `l_mon` and decreasing above `u_mon`. If `monotone=NULL`, no monotonicity constraints will be used. For only upper-tail monotonicity, specify `l_mon` as `NA`, and let `u_mon` equal the value after which the pdf is monotonically decreasing. Likewise, for lower-tail monotonicity, let `l_mon` equal the value below which the pdf is monotonically increassing, and set `u_mon` as `NA`

`convex` set as a two-length vector `convex=c(l_con,u_con)` to specify the convex regions. `f_X` is constrained to be convex for `x` below `l_con` and above `u_con`. If `convex=NULL`, no convexity constraints will be used. For only upper-tail convexity, specify `l_con` as `NA`, and let `u_con` equal the value after which the pdf is convex. Likewise, for lower-tail convexitity, let `l_con` equal the value below which the pdf is convex, and set `u_con` as `NA`

`support` set as a two-length vector `support=c(l_sup,u_sup)` to specify the lower and upper end of the support; the default is `support=c(min(Y),max(Y))`; the lower and upper end can be specified to be `-Inf` or `Inf`, respectively

Value

Return a SURE plot for regularization parameter correction and:

`lambda.seq` a vector of λ values on the horizontal axis of the SURE plot, automatically determined by the `SUREPlot` function or specified by users

`SURE.value` a vector of the SURE values corresponding to `lambda.seq` (on the vertical axis of the SURE plot)

Author(s)

Ran Yang, D. W. Apley

References

Ran Yang, Daniel Apley, Jeremy Staum and David Ruppert. Density Deconvolution with Additive Measurement Errors using Quadratic Programming.

Examples

```
rm(list=ls())

X=rgamma(3000,5,1)
Z=rnorm(3000,0,sd=1.8)
Y=X+Z
f_Z<-function(z){dnorm(z,mean=0,sd=1.8)}
K=min(200,floor(3*sqrt(3000)))

###SURE plots####
S=SUREPlot(Y=Y,Pdf=TRUE,K=K,f_Z=f_Z,lambda=NULL,
reg="2Deriv",integr=TRUE, nonneg=TRUE)
```