Package ‘DeconWK’

July 20, 2010

Type Package

Title Deconvolution by Weighted Kernels

Version 0.5-2

Date 2010-07-20

Author Martin L Hazelton and Berwin A Turlach

Maintainer Berwin A Turlach <berwin@maths.uwa.edu.au>

Description This package contains code for density deconvolution using weighted kernel estimators.

Depends R (>= 2.7.0), kernlab, stats

License GPL (>= 2)

LazyLoad yes

URL https://r-forge.r-project.org/projects/deconwk/

R topics documented:

DeconWK-package — *Deconvolution by Weighted Kernels*

Description

This package contains code for density deconvolution using weighted kernel estimators. Type 'citation("DeconWK")' for details of the implemented methods.

Details

The main functions are:

`w.hat`:	Calculates the weights for density deconvolution using weighted kernel estimators
`wkde`:	Calculates a weighted kernel density estimates
`decon.f`:	Calculates a classical deconvolution estimate

Author(s)

Authors: Martin L Hazelton and Berwin A Turlach

Maintainer: Berwin A Turlach <berwin@maths.uwa.edu.au>

References

Hazelton, M.L. and Turlach, B.A. (2009). Nonparametric density deconvolution by weighted kernel estimators, Statistics and Computing 19(3): 217–228. http://dx.doi.org/10.1007/s11222-008-9086-7.

cv.score — *Calculate the CV score for determining regularisation parameter*

Description

Evaluates the cross-validation criterion (11) of Hazelton and Turlach (2009).

Usage

```
cv.score(y, sigma, h, gamma,
         METHOD=c("exact", "svm"), K=5, verb=FALSE)
```

Arguments

`y`	the observed values.
`sigma`	the standard deviation of the contaminating (normal) distribution.
`h`	the smoothing parameter to be used.
`gamma`	vector of values from which a suitable value is to be selected
`METHOD`	method to be used to solve the quadratic programming problem involved in calculating the weights; if `"exact"` then `solveqp` is used, otherwise the routine `ipop` from the `kernlab` package is used.

K	number of folds to be used if `gamma` is chosen by cross-validation; defaults to 5.
verb	logical; if `TRUE` some progress report will be printed during cross-validation.

Value

A vector containing the cross-validation criterion evaluated at the values given in `gamma`.

Author(s)

Berwin A Turlach <berwin@maths.uwa.edu.au>

References

Hazelton, M.L. and Turlach, B.A. (2009). Nonparametric density deconvolution by weighted kernel estimators, Statistics and Computing 19(3): 217–228. http://dx.doi.org/10.1007/s11222-008-9086-7.

See Also

`w.hat`

Examples

```
set.seed(100719)
sig <- sqrt(29/40)  # Var(Z)/Var(X) = 0.1
y <- rden(100, DEN=3, sigma=sig)
h <- bw.SJ(y, method="dpi")
gamma <- exp(seq(from=0, to=6, length=17))

save.seed <- .Random.seed
cv1 <- cv.score(y, sigma=sig, h=h, gamma=gamma, METHOD="exact", verb=TRUE)
plot(log(gamma), cv1, type="b")
tmp <- getmin(log(gamma), cv1, which="r")
abline(v=tmp$xmin)

.Random.seed <- save.seed
cv2 <- cv.score(y, sigma=sig, h=h, gamma=gamma, METHOD="svm", verb=TRUE)
plot(log(gamma), cv2, type="b")
tmp <- getmin(log(gamma), cv2, which="r")
abline(v=tmp$xmin)

.Random.seed <- save.seed
cv1 <- cv.score(y, sigma=sig, h=h, gamma=gamma, METHOD="exact", K=10, verb=TRUE)
plot(log(gamma), cv1, type="b")
tmp <- getmin(log(gamma), cv1, which="r")
abline(v=tmp$xmin)

.Random.seed <- save.seed
cv2 <- cv.score(y, sigma=sig, h=h, gamma=gamma, METHOD="svm", K=10, verb=TRUE)
plot(log(gamma), cv2, type="b")
tmp <- getmin(log(gamma), cv2, which="r")
abline(v=tmp$xmin)
```

`decon.f` *Classical deconvolution density estimate*

Description

Calculates the classical deconvolution density estimate given in equation (4) of Hazelton and Turlach (2009).

Usage

```
decon.f(y, eval = NA, h = NA, sigma)
```

Arguments

`y`	the observed values.
`eval`	grid on which the deconvolution density estimate be calculated.
`h`	the smoothing parameter to be used.
`sigma`	the standard deviation of the contaminating (normal) distribution.

Details

If `"eval"` is not specified, it defaults to `seq(min(y)-sd(y), max(y)+sd(y), length=100)`.

If `"h"` is not specified, the plug-in bandwidth selector developed by Delaigle and Gijbels (2004) is used.

Value

A matrix with two columns named `"x"` and `"y"`; the first column contains the evaluation grid, `"eval"`, and the second column the deconvolution density estimate.

Author(s)

Martin L Hazelton `<m.hazelton@massey.ac.nz>`

References

Delaigle, A. and Gijbels, I. (2004). Practical bandwidth selection in deconvolution kernel density estimation. Computational Statistics & Data Analysis 45(2): 249–267.

Hazelton, M.L. and Turlach, B.A. (2009). Nonparametric density deconvolution by weighted kernel estimators, Statistics and Computing 19(3): 217–228. http://dx.doi.org/10.1007/s11222-008-9086-7.

Examples

```
set.seed(100712)
y <- rden(100, DEN=3, sigma=sqrt(29/40)) # Var(Z)/Var(X) = 0.1
f.hat <- decon.f(y, sigma=sqrt(29/40))
plot(f.hat, type="l")
```

getmin *Approximates the minimum of a function given on a grid*

Description

Approximates the minimum of a function given on a grid. Quadratic approximation around the point where the mimimal function value is observed is used (if that point is in the interior).

Usage

```
getmin(x, y, which="global", count.minima=FALSE, verbose=TRUE)
```

Arguments

`x`	Vector with the x-values at which the function is observed. Should be sorted.
`y`	Vector with the function values.
`which`	Defines which miminum we want to find. Possible values are "global" for the global minimum, "left" for the left-most local minimum and "right" for the right-most local minimum. Abbreviations ("g", "r", "gl", etc.) may be used.
`count.minima`	If TRUE, the number of local minima in the observed function values is returned.
`verbose`	If TRUE, the routine will give a warning if any exceptions occur.

Value

A list with the following elements is returned:

`xmin`	The x-coordinate of the minimum.
`ymin`	The approximate value of the function at the minimum.
`nmin`	The number of local minima in the y-vector (if requested, otherwise 0).
`excep`	Indicates whether an exeption has occurred: -1 if the minimum was found at the left end, 1 if the minimum was found at the right end, 5 if the minimum was in the middle but the quadratic fit yielded a location of the minimum which was outside of the interval defined by the three points used for the quadratic fit and 0 in all other cases.

Note

The vector x must be sorted.

Author(s)

Berwin A Turlach `<berwin@maths.uwa.edu.au>`

Examples

```
x <- -100:100/50
y <- x*x
getmin(x,y)
```

`rden` *Specific (contaminated) distributions*

Description

Density functions and random generation from the distributions considered in Hazelton and Turlach (2009); details of the distributions (all Gaussian mixtures) are given on pages 221–222.

Usage

```
dden(eval, DEN=1, sigma=0)
rden(N, DEN = 1, sigma=0)
```

Arguments

`eval`	vector of quantiles.
`N`	number of observations to be simulated; Should be a single number.
`DEN`	density to simulate from; possible values are 1, 2, 3 and 4 corresponding to the densities described in the paper.
`sigma`	the standardard deviation of the contaminating measurement error.

Details

The generated random variates are from $X + Z$ where the distribution of X is determined by the argument `DEN` and Z has a normal distribution with mean zero and standard deviation `sigma`; X and Z are independent.

Value

A vector with the generated random variates.

Author(s)

Martin L Hazelton `<m.hazelton@massey.ac.nz>`

References

Hazelton, M.L. and Turlach, B.A. (2009). Nonparametric density deconvolution by weighted kernel estimators, Statistics and Computing 19(3): 217–228. http://dx.doi.org/10.1007/s11222-008-9086-7.

Examples

```
##
## Figure 1 from paper
##
opar <- par(mfrow=c(2,2))
eval1 <- seq(-4,4,length=200)
eval2 <- eval1
eval3 <- seq(-8,7,length=300)
eval4 <- seq(-2,30,length=320)
```

```
plot(eval1,dden(eval1,DEN=1),type="l",xlab="",ylab="density")
lines(eval1,dden(eval1,DEN=1,sigma=0.5),lty=3,lwd=1.25)
lines(eval1,dden(eval1,DEN=1,sigma=0.5/sqrt(2.5)),lty=2,lwd=1.25)
lines(eval1,dden(eval1,DEN=1,sigma=0.5*sqrt(2)),lty=4,lwd=1.25)
title("Density 1")
plot(eval2,dden(eval2,DEN=2),type="l",xlab="",ylab="density")
lines(eval2,dden(eval2,DEN=2,sigma=sqrt(51/300)),lty=3,lwd=1.25)
lines(eval2,dden(eval2,DEN=2,sigma=sqrt(51/300)/sqrt(2.5)),lty=2,lwd=1.25)
lines(eval2,dden(eval2,DEN=2,sigma=sqrt(51/300)*sqrt(2)),lty=4,lwd=1.25)
title("Density 2")
plot(eval3,dden(eval3,DEN=3),type="l",xlab="",ylab="density")
lines(eval3,dden(eval3,DEN=3,sigma=sqrt(1.8125)),lty=3,lwd=1.25)
lines(eval3,dden(eval3,DEN=3,sigma=sqrt(1.8125)/sqrt(2.5)),lty=2,lwd=1.25)
lines(eval3,dden(eval3,DEN=3,sigma=sqrt(1.8125)*sqrt(2)),lty=4,lwd=1.25)
title("Density 3")
plot(eval4,dden(eval4,DEN=4),type="l",xlab="",ylab="density")
lines(eval4,dden(eval4,DEN=4,sigma=sqrt(2.516)),lty=3,lwd=1.25)
lines(eval4,dden(eval4,DEN=4,sigma=sqrt(2.516)/sqrt(2.5)),lty=2,lwd=1.25)
lines(eval4,dden(eval4,DEN=4,sigma=sqrt(2.516)*sqrt(2)),lty=4,lwd=1.25)
title("Density 4")
par(opar)
```

`solveqp` *Solves a specific quadratic programming problem*

Description

Solves the quadratic programming problem (9) of Hazelton and Turlach via a homotopy algorithm approach as described in Appendix B.

Usage

```
solveqp(Qmat, bvec)
```

Arguments

`Qmat`	The matrix $\mathbf{Q}$ in equation (9a) of Hazelton and Turlach (2009)
`bvec`	The vector $\boldsymbol{b}$ in equation (9a) of Hazelton and Turlach (2009)

Value

The vector $\boldsymbol{w}$ that solves the quadratic problem (9).

Note, the entries in this vector add to one as the code works with a different parameterisation of the weight vector.

Author(s)

Berwin A Turlach `<berwin@maths.uwa.edu.au>`

References

Hazelton, M.L. and Turlach, B.A. (2009). Nonparametric density deconvolution by weighted kernel estimators, Statistics and Computing 19(3): 217–228. http://dx.doi.org/10.1007/s11222-008-9086-7.

See Also

ipop

---

w.hat *Calculate weights for deconvolution*

Description

Routine to calculate the weights for deconvolution via weighted kernel density estimates.

Usage

```
w.hat(y, sigma, h, gamma,
      METHOD=c("exact", "exact.cv", "svm", "svm.cv"), K=5, verb=FALSE)
```

Arguments

y	the observed, contaminated data.
sigma	the standard deviation of the contaminating (normal) distribution.
h	the bandwidth to be used for the weighted kernel density estimate; if missing the bandwidth returned by `bw.SJ(y, method="dpi")` will be used.
gamma	the regularisation parameter to be used; either a scalar for methods `"exact"` and `"svm"`, or a vector of values from which a suitable value is selected via K-fold cross-validation for methods `"exact.cv"` and `"svm.cv"`.
METHOD	method to be used to solve the quadratic programming problem involved in calculating the weights; if `"exact"` or `"exact.cv"` then `solveqp` is used, otherwise `ipop` from the `kernlab` package is used.
K	number of folds to be used if `gamma` is chosen by cross-validation; defaults to 5.
verb	logical; if `TRUE` some progress report will be printed during cross-validation.

Value

A vector containing the weights; if `gamma` is chosen by cross-validation, the selected value is returned as an attribute.

Author(s)

Martin L Hazelton <m.hazelton@massey.ac.nz>

Berwin A Turlach <berwin@maths.uwa.edu.au>

References

Hazelton, M.L. and Turlach, B.A. (2009). Nonparametric density deconvolution by weighted kernel estimators, Statistics and Computing 19(3): 217–228. http://dx.doi.org/10.1007/s11222-008-9086-7.

See Also

wkde

Examples

```
set.seed(100719)
sig <- sqrt(29/40)  # Var(Z)/Var(X) = 0.1
y <- rden(100, DEN=3, sigma=sig)
gamma <- exp(seq(from=0, to=6, length=17))

save.seed <- .Random.seed
w1 <- w.hat(y, sigma=sig, gamma=gamma, METHOD="exact.cv", verb=TRUE)
plot(y, w1, type="h")
attributes(w1)
.Random.seed <- save.seed
w2 <- w.hat(y, sigma=sig, gamma=gamma, METHOD="svm.cv", verb=TRUE)
plot(y, w2, type="h")
attributes(w2)

.Random.seed <- save.seed
w1 <- w.hat(y, sigma=sig, gamma=gamma, METHOD="exact.cv", K=10, verb=TRUE)
plot(y, w1, type="h")
attributes(w1)
.Random.seed <- save.seed
w2 <- w.hat(y, sigma=sig, gamma=gamma, METHOD="svm.cv", K=10, verb=TRUE)
plot(y, w2, type="h")
attributes(w2)
```

---

wkde *Weighted kernel density estimate*

Description

Calculates a weighted kernel density estimate as defined by equation (5) of Hazelton and Turlach (2009).

Usage

```
wkde(y, eval = NA, w = NA, h = NA)
```

Arguments

`y`	the observed values.
`eval`	grid on which the deconvolution density estimate be calculated.
`w`	the weights to be used.
`h`	the smoothing parameter to be used.

Details

If `"eval"` is not specified, it defaults to `seq(min(y)-0.1*sd(y), max(y)+0.1*sd(y), length=100)`.

If `"w"` is not specified, it defaults to a vector of ones.

If `"h"` is not specified, it defaults to `bw.SJ(y, method="dpi")`.

Value

A matrix with two columns named "x" and "y"; the first column contains the evaluation grid, "eval", and the second column the deconvolution density estimate.

Author(s)

Martin L Hazelton <m.hazelton@massey.ac.nz>

References

Hazelton, M.L. and Turlach, B.A. (2009). Nonparametric density deconvolution by weighted kernel estimators, Statistics and Computing 19(3): 217–228. http://dx.doi.org/10.1007/s11222-008-9086-7.

See Also

w.hat

Examples

```
set.seed(100712)
sig <- sqrt(29/40)  # Var(Z)/Var(X) = 0.1
y <- rden(100, DEN=3, sigma=sig)
f.hat <- wkde(y)
plot(f.hat, type="l", ylim=c(0, 0.2))
w <- w.hat(y, sigma=sig, gamma=2.05)
fd.hat <- wkde(y, w=w)
lines(fd.hat, col="red")
w <- w.hat(y, sigma=sig, gamma=4.4)
fd.hat <- wkde(y, w=w)
lines(fd.hat, col="blue")
```

Index