Package ‘aws’

January 27, 2015

Version 1.9-4
Date 2014-03-04
Title Adaptive Weights Smoothing
Depends R (>= 2.14.0), methods, awsMethods (>= 1.0-1), gsl
Description The package contains R-functions implementing the
Propagation-Separation Approach to adaptive smoothing as
described in J. Polzehl and V. Spokoiny (2006),
Propagation-Separation Approach for Local Likelihood Estimation,
regression estimation: Propagation-separation approach,
WIAS-Preprint 998.
License GPL (>= 2)
Copyright This package is Copyright (C) 2005-2014 Weierstrass
Institute for Applied Analysis and Stochastics.
URL http://www.wias-berlin.de/projects/matheon_a3/
Author Joerg Polzehl [aut, cre],
      Felix Anker [ctb]
Maintainer Joerg Polzehl <joerg.polzehl@wias-berlin.de>
NeedsCompilation yes
Repository CRAN
Date/Publication 2014-03-05 18:21:44

R topics documented:

aws-package .......................................................... 2
aws .............................................................. 3
aws-class .......................................................... 7
aws.gaussian ...................................................... 9
aws.irreg .......................................................... 12
aws.segment ....................................................... 15

Details

| Package: | aws |
| Version: | 1.6 |
| Date: | 2009-04-07 |
| License: | GPL (>=2) |
| Copyright: | 2008 Weierstrass Institute for Applied Analysis and Stochastics, |
| URL: | http://www.wias-berlin.de/project-areas/stat/ |

Index:

aws AWS for local constant models on a grid
aws

aws.gaussian Adaptive weights smoothing for Gaussian data with variance depending on the mean.
aws.irreg local constant AWS for irregular (1D/2D) design
aws.segment Segmentation by adaptive weights for Gaussian models.
awsdata Extract information from an object of class aws
binning Binning in 1D, 2D or 3D
lpaws Local polynomial smoothing by AWS
kernsm 1D, 2D, 3D nonparametric kernel smoothing via fft
ICsmooth pointwise adaptive kernel smoothing
ICcombined pointwise adaptive kernel smoothing with fusing

Author(s)

Joerg Polzehl <polzehl@wias-berlin.de>
Maintainer: Joerg Polzehl <polzehl@wias-berlin.de>

References


aws AWS for local constant models on a grid

Description

The function implements the propagation separation approach to nonparametric smoothing (formerly introduced as Adaptive weights smoothing) for varying coefficient likelihood models on a 1D, 2D or 3D grid. For "Gaussian" models, i.e. regression with additive "Gaussian" errors, a homoskedastic or heteroskedastic model is used depending on the content of sigma2

Usage

```r
aws(y,hmax=NULL,aws=TRUE, memory=FALSE, family="Gaussian", lkern="Triangle", homogen=TRUE, aggkern="Uniform", sigma2=NULL, shape=NULL, scorr=0, spmin=0.25, ladjust=1, wghts=NULL, u=NULL, graph=FALSE, demo=FALSE, testprop=FALSE, maxni=FALSE)
```
Arguments

**y**
- y contains the observed response data. \( \text{dim}(y) \) determines the dimensionality and extend of the grid design.

**hmax**
- hmax specifies the maximal bandwidth. Defaults to \( hmax=250, 12, 5 \) for \( dd=1, 2, 3 \), respectively.

**aws**
- logical: if TRUE structural adaptation (AWS) is used.

**memory**
- logical: if TRUE stagewise aggregation is used as an additional adaptation scheme.

**family**
- family specifies the probability distribution. Default is family="Gaussian", also implemented are "Bernoulli", "Poisson", "Exponential", "Volatility", "Variance" and "NCchi". family="Volatility" specifies a Gaussian distribution with expectation \( 0 \) and unknown variance. family="Volatility" specifies that \( p \cdot y/\theta \) is distributed as \( \chi^2 \) with \( p \cdot \text{shape} \) degrees of freedom. family="NCchi" uses a noncentral Chi distribution with \( p \cdot \text{shape} \) degrees of freedom and noncentrality parameter \( \theta \)

**lkern**
- character: location kernel, either "Triangle", "Plateau", "Quadratic", "Cubic" or "Gaussian"

**homogen**
- logical: if TRUE the function tries to determine regions where weights can be fixed to 1. This may increase speed.

**aggkern**
- character: kernel used in stagewise aggregation, either "Triangle" or "Uniform"

**sigma2**
- sigma2 allows to specify the variance in case of family="Gaussian". Not used if family!="Gaussian". Defaults to NULL. In this case a homoskedastic variance estimate is generated. If \( \text{length}(\text{sigma2})=\text{length}(y) \) then sigma2 is assumed to contain the pointwise variance of \( y \) and a heteroscedastic variance model is used.

**shape**
- Allows to specify an additional shape parameter for certain family models. Currently only used for family="Variance", that is \( \chi^2 \)-Square distributed observations with shape degrees of freedom.

**scorr**
- The vector scorr allows to specify a first order correlations of the noise for each coordinate direction, defaults to 0 (no correlation).

**spmin**
- Determines the form (size of the plateau) in the adaptation kernel. Not to be changed by the user.

**ladjust**
- factor to increase the default value of lambda

**wghts**
- wghts specifies the diagonal elements of a weight matrix to adjust for different distances between grid-points in different coordinate directions, i.e. allows to define a more appropriate metric in the design space.

**u**
- a "true" value of the regression function, may be provided to report risks at each iteration. This can be used to test the propagation condition with \( u=0 \)

**graph**
- If graph=TRUE intermediate results are illustrated after each iteration step. Defaults to graph=FALSE.

**demo**
- If demo=TRUE the function pauses after each iteration. Defaults to demo=FALSE.

**testprop**
- If set this provides diagnostics for testing the propagation condition. The values of y should correspond to the specified family and a global model.
maxni

If TRUE use $\max_{l\leq k}(\mathcal{N}_i^{(l)})$ instead of $(\mathcal{N}_i^{(k)})$ in the definition of the statistical penalty.

Details

The function implements the propagation separation approach to nonparametric smoothing (formerly introduced as Adaptive weights smoothing) for varying coefficient likelihood models on a 1D, 2D or 3D grid. For "Gaussian" models, i.e. regression with additive "Gaussian" errors, a homoskedastic or heteroskedastic model is used depending on the content of $\sigmaR$. $\text{aws}==\text{FALSE}$ provides the stagewise aggregation procedure from Belomestny and Spokoiny (2004). $\text{memory}==\text{FALSE}$ provides Adaptive weights smoothing without control by stagewise aggregation.

The essential parameter in the procedure is a critical value $\lambda$. This parameter has an interpretation as a significance level of a test for equivalence of two local parameter estimates. Optimal values mainly depend on the chosen family. Values set internally are chosen to fulfill a propagation condition, i.e. in case of a constant (global) parameter value and large $h_{\max}$ the procedure provides, with a high probability, the global (parametric) estimate. More formally we require the parameter $\lambda$ to be specified such that $E|\hat{\theta}_k - \theta| \leq (1 + \alpha)E|\tilde{\theta}_k - \theta|$ where $\hat{\theta}_k$ is the aws-estimate in step $k$ and $\tilde{\theta}_k$ is corresponding nonadaptive estimate using the same bandwidth ($\lambda=\text{inf}$).

The value of $\lambda$ can be adjusted by specifying the factor $\text{ladjust}$. Values $\text{ladjust}^Q$ lead to an less effective adaptation while $\text{ladjust}<\frac{1}{2}$ may lead to random segmentation of, with respect to a constant model, homogeneous regions.

The numerical complexity of the procedure is mainly determined by $h_{\max}$. The number of iterations is approximately $\text{Const} \times d \times \log(\max_{\max})/\log(1.25)$ with $d$ being the dimension of $y$ and the constant depending on the kernel $\text{lkern}$. Complexity in each iteration step is $\text{Const} \times h_{\text{akt}} \times n$ with $h_{\text{akt}}$ being the actual bandwidth in the iteration step and $n$ the number of design points. $\max_{\max}$ determines the maximal possible variance reduction.

Value

returns an object of class $\text{aws}$ with slots

- $y$ = "numeric" y
- $dy$ = "numeric" dim(y)
- $x$ = "numeric" numeric(0)
- $ni$ = "integer" integer(0)
- $mask$ = "logical" logical(0)
- $\theta$ = "numeric"
  Estimates of regression function, length: length(y)
- $\text{mae}$ = "numeric"
  Mean absolute error for each iteration step if $u$ was specified, numeric(0) else
- $\text{var}$ = "numeric"
  approx. variance of the estimates of the regression function. Please note that this does not reflect variability due to randomness of weights.
- $\text{xmin}$ = "numeric"
  numeric(0)
xmax = "numeric"
    numeric(0)
wghts = "numeric"
    numeric(0)
degree = "integer"
    0
hmax = "numeric"
    effective hmax
sigma2 = "numeric"
    provided or estimated error variance
scorr = "numeric"
    scorr
family = "character"
    family
shape = "numeric"
    shape
lkern = "integer"
    integer code for lkern, 1="Plateau", 2="Triangle", 3="Quadratic", 4="Cubic", 5="Gaussian"
lambda = "numeric"
    effective value of lambda
ladjust = "numeric"
    effective value of ladjust
aws = "logical"
    aws
memory = "logical"
    memory
homogen = "logical"
    homogen
earlystop = "logical"
    FALSE
vcoef = "numeric"
    "Constant"
numeric(0)
call = "function"
    the arguments of the call to aws

Author(s)
Joerg Polzehl, <polzehl@wias-berlin.de>, http://www.wias-berlin.de/project-areas/stat/projects/adaptive-image-processing.html
References


See Also

See also lpaws, link{awsdata}, aws.reg, aws.gaussian

Examples

```r
require(aws)
# 1D local constant smoothing
## Not run: demo(aws_ex1)
## Not run: demo(aws_ex2)
# 2D local constant smoothing
## Not run: demo(aws_ex3)
```

aws-class

Class "aws"

Description

The "aws" class is used for objects obtained by functions aws, lpaws, aws.reg and aws.gaussian.

Objects from the Class

Objects are created by calls to functions aws, lpaws, aws.reg and aws.gaussian.

Slots

- `.Data`: Object of class "list", usually empty.
- `y`: Object of class "array" containing the original (response) data
- `dy`: Object of class "numeric" dimension attribute of `y`
- `x`: Object of class "numeric" if provided the design points
- `ni`: Object of class "numeric" sum of weights used in final estimate
- `mask`: Object of class "logical" mask of design points where computations are performed
- `theta`: Object of class "array" contains the smoothed object and in case of function lpaws its derivatives up to the specified degree. Dimension is dim(theta)=c(dy,p)
- `mae`: Object of class "numeric" Mean absolute error with respect to array in argument `u` if provided.
var: Object of class "numeric" pointwise variance of theta[...1]
xmin: Object of class "numeric" min of x in case of irregular design
xmax: Object of class "numeric" max of x in case of irregular design
wghts: Object of class "numeric" weights used in location penalty for different coordinate directions
degree: Object of class "integer" degree of local polynomials used in function lpaws
hmax: Object of class "numeric" maximal bandwidth
sigmaR: Object of class "numeric" estimated error variance
scorr: Object of class "numeric" estimated spatial correlation
family: Object of class "character" distribution of y, can be any of c("Gaussian", "Bernoulli", "Poisson", "Exponential")
shape: Object of class "numeric" possible shape parameter of distribution of y
lkern: Object of class "integer" location kernel, can be any of c("Triangle", "Quadratic", "Cubic", "Plateau", "Gaussian")
defaults to "Triangle"
lambda: Object of class "numeric" scale parameter used in adaptation
ladjust: Object of class "numeric" factor to adjust scale parameter with respect to its predetermined default.
aws: Object of class "logical" Adaptation by Propagation-Separation
memory: Object of class "logical" Adaptation by Stagewise Aggregation
homogen: Object of class "logical" detect regions of homogeneity (used to speed up the calculations)
earlystop: Object of class "logical" further speedup in function lpaws estimates are fixed if sum of weights does not increase with iterations.
varmodel: Object of class "character" variance model used in function aws.gaussian
vcoef: Object of class "numeric" estimates variance parameters in function aws.gaussian
call: Object of class "call" that created the object.

Methods

extract signature(x = "aws"): ...

risk signature(y = "aws"): ...

plot Method for Function ‘plot’ in Package ‘aws’.

show Method for Function ‘show’ in Package ‘aws’.

print Method for Function ‘print’ in Package ‘aws’.

summary Method for Function ‘summary’ in Package ‘aws’.

Author(s)

Joerg Polzehl, <polzehl@wias-berlin.de>
aws.gaussian

References


See Also

aws, lpaws, aws.irreg, aws.gaussian

Examples

showClass("aws")

aws.gaussian Adaptive weights smoothing for Gaussian data with variance depending on the mean.

Description

The function implements a semiparametric adaptive weights smoothing algorithm designed for regression with additive heteroskedastic Gaussian noise. The noise variance is assumed to depend on the value of the regression function. This dependence is modeled by a global parametric (polynomial) model.

Usage

aws.gaussian(y, hmax = NULL, hpre = NULL, aws = TRUE, memory = FALSE, varmodel = "Constant", lkern = "Triangle", homogen = TRUE, aggkern = "Uniform", scorr = 0, mask=NULL, ladjust = 1, wghts = NULL, u = NULL, varprop = 0.1, graph = FALSE, demo = FALSE)

Arguments

y y contains the observed response data. dim(y) determines the dimensionality and extent of the grid design.

hmax hmax specifies the maximal bandwidth. Defaults to hmax=250, 12, 5 for dd=1, 2, 3, respectively.

hpre Describe hpre Bandwidth used for an initial nonadaptive estimate. The first estimate of variance parameters is obtained from residuals with respect to this estimate.

aws logical: if TRUE structural adaptation (AWS) is used.

memory logical: if TRUE stagewise aggregation is used as an additional adaptation scheme.

varmodel Implemented are "Constant", "Linear" and "Quadratic" referring to a polynomial model of degree 0 to 2.
1kern character: location kernel, either "Triangle", "Plateau", "Quadratic", "Cubic" or "Gaussian"

homogen logical: if TRUE the function tries to determine regions where weights can be fixed to 1. This may increase speed.

agglern character: kernel used in stagewise aggregation, either "Triangle" or "Uniform"

scorr The vector scorr allows to specify a first order correlations of the noise for each coordinate direction, defaults to 0 (no correlation).

mask Restrict smoothing to points where mask==TRUE. Defaults to TRUE in all voxel.

ladjust factor to increase the default value of lambda

wghts wghts specifies the diagonal elements of a weight matrix to adjust for different distances between grid-points in different coordinate directions, i.e. allows to define a more appropriate metric in the design space.

u a "true" value of the regression function, may be provided to report risks at each iteration. This can be used to test the propagation condition with u=0

varprop Small variance estimates are replaced by varprop times the mean variance.

graph If graph=TRUE intermediate results are illustrated after each iteration step. Defaults to graph=FALSE.

demo If demo=TRUE the function pauses after each iteration. Defaults to demo=FALSE.

Details

The function implements the propagation separation approach to nonparametric smoothing (formerly introduced as Adaptive weights smoothing) for varying coefficient likelihood models on a 1D, 2D or 3D grid. In contrast to function aws observations are assumed to follow a Gaussian distribution with variance depending on the mean according to a specified global variance model. aws==FALSE provides the stagewise aggregation procedure from Belomestny and Spokoiny (2004). memory==FALSE provides Adaptive weights smoothing without control by stagewise aggregation.

The essential parameter in the procedure is a critical value lambda. This parameter has an interpretation as a significance level of a test for equivalence of two local parameter estimates. Values set internally are choosen to fulfil a propagation condition, i.e. in case of a constant (global) parameter value and large hmax the procedure provides, with a high probability, the global (parametric) estimate. More formally we require the parameter lambda to be specified such that E|\hat{\theta}_k - \theta| \leq (1 + \alpha)E|\hat{\theta} - \theta| where \hat{\theta}_k is the aws-estimate in step k and \hat{\theta} is corresponding nonadaptive estimate using the same bandwidth (lambda=Inf). The value of lambda can be adjusted by specifying the factor ladjust. Values ladjust>1 lead to an less effective adaptation while ladjust<<1 may lead to random segmentation of, with respect to a constant model, homogeneous regions.

The numerical complexity of the procedure is mainly determined by hmax. The number of iterations is approximately Const*d*log(hmax)/log(1.25) with d being the dimension of y and the constant depending on the kernel 1kern. Complexity in each iteration step is Const*hakt*n with hakt being the actual bandwidth in the iteration step and n the number of design points. hmax determines the maximal possible variance reduction.

Value

returns an object of class aws with slots
aws.gaussian

y = "numeric" y
dy = "numeric" dim(y)
x = "numeric" numeric(0)
ni = "integer" integer(0)
mask = "logical"
    logical(0)
theta = "numeric"
    Estimates of regression function, length: length(y)
mae = "numeric"
    Mean absolute error for each iteration step if u was specified, numeric(0) else
var = "numeric"
    approx. variance of the estimates of the regression function. Please note that
    this does not reflect variability due to randomness of weights.
xmin = "numeric"
    numeric(0)
xmax = "numeric"
    numeric(0)
wghts = "numeric"
    numeric(0)
degree = "integer"
    0
hmax = "numeric"
    effective hmax
sigma2 = "numeric"
    provided or estimated error variance
scorr = "numeric"
    scorr
family = "character"
    "Gaussian"
shape = "numeric"
    NULL
lkern = "integer"
    integer code for lkern, 1="Plateau", 2="Triangle", 3="Quadratic", 4="Cubic", 5="Gaussian"
lambda = "numeric"
    effective value of lambda
ladjust = "numeric"
    effective value of ladjust
aws = "logical"
    aws
memory = "logical"
    memory
homogen = "logical"
    homogen
earlystop = "logical"
    FALSE
varmodel = "character"
    varmodel
vcoef = "numeric"
    estimated parameters of the variance model
call = "function"
    the arguments of the call to aws.gaussian

Author(s)

Joerg Polzehl, <polzehl@wias-berlin.de>, http://www.wias-berlin.de/project-areas/stat/projects/adaptive-image-processing.html

References


See Also

See also aws, link{awsdata}, aws.irreg

Examples

require(aws)

aws.irreg(y, x, hmax = NULL, aws=TRUE, memory=FALSE, varmodel = "Constant",
    lkern = "Triangle", aggkern = "Uniform", sigma2 = NULL, nbins = 100,
    hpre = NULL, henv = NULL, ladjust =1, varprop = 0.1, graph = FALSE)
Arguments

- **y**: The observed response vector (length n)
- **x**: Design matrix, dimension n x d (\%in\% 1:2)
- **hmax**: hmax specifies the maximal bandwidth. Unit is binwidth in the first dimension.
- **aws**: logical: if TRUE structural adaptation (AWS) is used.
- **memory**: logical: if TRUE stagewise aggregation is used as an additional adaptation scheme.
- **varmodel**: determines the model that relates variance to mean. Either "Constant", "Linear" or "Quadratic".
- **lkern**: character: location kernel, either "Triangle", "Plateau", "Quadratic", "Cubic" or "Gaussian".
- **aggkern**: character: kernel used in stagewise aggregation, either "Triangle" or "Uniform".
- **sigma2**: sigma2 allows to specify the variance in case of varmodel="Constant", estimated if not given.
- **nbins**: number of bins, can be NULL, a positive integer or a vector of positive integers (length d)
- **hpre**: smoothing bandwidth for initial variance estimate
- **henv**: radius of balls around each observed design point where estimates will be calculated
- **ladjust**: factor to increase the default value of lambda
- **varprop**: exclude the largest 100*varprop\% squared residuals when estimating the error variance
- **graph**: If graph=TRUE intermediate results are illustrated after each iteration step. Defaults to graph=FALSE.

Details

Data are first binned (1D/2D), then aws is performed on all datapoints within distance <= henv of nonempty bins.

Value

returns an object of class aws with slots

- **y** = "numeric" y
- **dy** = "numeric" dim(y)
- **x** = "numeric" x
- **ni** = "integer" number of observations per bin
- **mask** = "logical"
  bins where parameters have been estimated
- **theta** = "numeric"
  Estimates of regression function, length: length(y)
mae = "numeric"
    numeric(0)
var = "numeric"
    approx. variance of the estimates of the regression function. Please note that
    this does not reflect variability due to randomness of weights.
xmin = "numeric"
    vector of minimal x-values (bins)
xmax = "numeric"
    vector of maximal x-values (bins)
wghts = "numeric"
    relative binwidths
degree = "integer"
    0
hmax = "numeric"
    effective hmax
sigma2 = "numeric"
    provided or estimated error variance
scorr = "numeric"
    0
family = "character"
    "Gaussian"
shape = "numeric"
    numeric(0)
lkern = "integer"
    integer code for lkern, 1="Plateau", 2="Triangle", 3="Quadratic", 4="Cubic",
    5="Gaussian"
lambda = "numeric"
    effective value of lambda
ladjust = "numeric"
    effective value of ladjust
aws = "logical"
    aws
memory = "logical"
    memory
homogen = "logical"
    FALSE
everystop = "logical"
    FALSE
vmodel = "character"
    vmodel
vcoef = "numeric"
    estimated coefficients in variance model
call = "function"
    the arguments of the call to aws

Author(s)

Joerg Polzehl,<polzehl@wias-berlin.de>
aws.segment  

References


See Also

See also lpaws, link(awsdata), lpaws

Examples

```r
require(aws)
# 1D local constant smoothing
## Not run: demo(irreg_ex1)
# 2D local constant smoothing
## Not run: demo(irreg_ex2)
```

aws.segment  

Segmentation by adaptive weights for Gaussian models.

Description

The function implements a modification of the adaptive weights smoothing algorithm for segmentation into three classes. The

Usage

```r
aws.segment(y, level, delta = 0, hmax = NULL, hpre = NULL, varmodel = "Constant", lkern = "Triangle", scorr = 0, ladjust = 1, wgts = NULL, u = NULL, varprop = 0.1, ext = 0, graph = FALSE, demo = FALSE, fov=NULL)
```

Arguments

- `y`: y contains the observed response data. `dim(y)` determines the dimensionality and extend of the grid design.
- `level`: center of second class
- `delta`: half width of second class
- `hmax`: hmax specifies the maximal bandwidth. Defaults to `hmax=250`, 12, 5 for `dd=1, 2, 3`, respectively.
- `hpre`: Describe `hpre` Bandwidth used for an initial nonadaptive estimate. The first estimate of variance parameters is obtained from residuals with respect to this estimate.
- `varmodel`: Implemented are "Constant", "Linear" and "Quadratic" refering to a polynomial model of degree 0 to 2.
- `lkern`: character: location kernel, either "Triangle", "Plateau", "Quadratic", "Cubic" or "Gaussian"
The vector `scorr` allows to specify a first order correlations of the noise for each coordinate direction, defaults to 0 (no correlation).

`lambda`  
Factor to increase the default value of lambda.

`wgths`  
Specifies the diagonal elements of a weight matrix to adjust for different distances between grid-points in different coordinate directions, i.e. allows to define a more appropriate metric in the design space.

`u`  
A "true" value of the regression function, may be provided to report risks at each iteration. This can be used to test the propagation condition with $u=0$.

`varprop`  
Small variance estimates are replaced by `varprop` times the mean variance.

`ext`  
Intermediate results are fixed if the test statistics exceeds the critical value by `ext`.

`graph`  
If `graph=TRUE` intermediate results are illustrated after each iteration step. Defaults to `graph=FALSE`.

`demo`  
If `demo=TRUE` the function pauses after each iteration. Defaults to `demo=FALSE`.

`fov`  
Field of view. Size of region (sample size) to adjust for in multiscale testing.

Details

The image is segmented into three parts by performing multiscale tests of the hypotheses $H_1$ value $\geq$ level $- \delta$ and $H_2$ value $\leq$ level $+ \delta$. Pixel where the first hypothesis is rejected are classified as $MQ$ (segment 1) while rejection of $H_2$ results in classification $Q$ (segment 3). Pixel where neither $H_1$ or $H_2$ are rejected are assigned to a value $P$ (segment 2). Critical values for the tests are adjusted for smoothness at the different scales inspected in the iteration process using results from multiscale testing, see e.g. Dümbgen and Spokoiny (2001). Critical values also depend on the size of the region of interest specified in parameter `fov`.

Within segment 2 structural adaptive smoothing is performed while if a pair of pixel belongs to segment 1 or segment 3 the corresponding weight will be nonadaptive.

Value

returns an object of class `aws` with slots

- `y = "numeric"`  
  y

- `dy = "numeric"`  
  dim(y)

- `x = "numeric"`  
  numeric(0)

- `ni = "integer"`  
  integer(0)

- `mask = "logical"`  
  logical(0)

- `segment = "integer"`  
  Segmentation results, class numbers 1-3

- `theta = "numeric"`  
  Estimates of regression function, length: length(y)

- `mae = "numeric"`  
  Mean absolute error for each iteration step if u was specified, numeric(0) else
var = "numeric"
    approx. variance of the estimates of the regression function. Please note that
    this does not reflect variability due to randomness of weights.

xmin = "numeric"
    numeric(0)

xmax = "numeric"
    numeric(0)

wghts = "numeric"
    numeric(0)

degree = "integer"
    0

hmax = "numeric"
    effective hmax

sigma2 = "numeric"
    provided or estimated error variance

scorr = "numeric"
    scorr

family = "character"
    "Gaussian"

shape = "numeric"
    NULL

lkern = "integer"
    integer code for lkern, 1="Plateau", 2="Triangle", 3="Quadratic", 4="Cubic",
    5="Gaussian"

lambda = "numeric"
    effective value of lambda

ladjust = "numeric"
    effective value of ladjust

aws = "logical"
    aws

memory = "logical"
    memory

homogen = "logical"
    FALSE

earlystop = "logical"
    FALSE

vmodel = "character"
    vmodel

vcoef = "numeric"
    estimated parameters of the variance model

call = "function"
    the arguments of the call to aws.gaussian

Note

This function is still experimental and may be changes considerably in future.
Author(s)
Joerg Polzehl, <polzehl@wias-berlin.de>, http://www.wias-berlin.de/project-areas/stat/projects/adaptive-image-processing.html

References

See Also
aws, aws.gaussian

Examples
require(aws)

awsdata(awsobj, what)

Arguments
awsobj an object of class aws
what can be "data" (extracts observed response), "theta" (estimated parameters), "est" (estimated regression function), "var" (approx. variance of estimated regression function), "sd" (approx. standard deviation of estimated regression function), "sigma2" (error variance), "mae" (mean absolute error for each iteration step, if available), "ni" (number of observations per bin), "mask" (logical indicator for bins where the regression function is estimated). "bi" (array of sum of weights or NULL) "bi2" (array of sum of squared weights or NULL)

Details
The returned object is formatted as an array if appropriate. The returned object may be NULL if the information is not available.
**Value**

an vector or array containing the specified information.

**Author(s)**

Joerg Polzehl <polzehl@wias-berlin.de>

**References**


**See Also**

link{awsdata}, aws, aws.irreg

**Examples**

```r
require(aws)
# 1D local constant smoothing
## Not run: demo(aws_ex1)
## Not run: demo(aws_ex2)
# 2D local constant smoothing
## Not run: demo(aws_ex3)
# 1D local polynomial smoothing
## Not run: demo(lpaws_ex1)
# 2D local polynomial smoothing
## Not run: demo(lpaws_ex2)
# 1D irregular design
## Not run: demo(irreg_ex1)
# 2D irregular design
## Not run: demo(irreg_ex2)
```

---

**awssegment-class**

Class "awssegment"

**Description**

The "aws" class is used for objects obtained by functions aws.segment
Objects from the Class

Objects are created by calls to functions aws.segment

Slots

.data: Object of class "list", usually empty.
y: Object of class "array" containing the original (response) data
dy: Object of class "numeric" dimension attribute of y
x: Object of class "numeric" if provided the design points
ni: Object of class "numeric" sum of weights used in final estimate
mask: Object of class "logical" mask of design points where computations are performed
segment: Object of class "array" segmentation results (3 segments coded by c(-1, 0, 1))
level: Object of class "numeric" center of segment 0
delta: Object of class "numeric" half width of segment 0
theta: Object of class "array" ~
theta: Object of class "array" contains the smoothed object and in case of function lpaws its
derivatives up to the specified degree. Dimension is dim(theta)=c(dy,p)
mae: Object of class "numeric" Mean absolute error with respect to array in argument u if pro-
vided.
var: Object of class "numeric" pointwise variance of theta[,...,1]
xmin: Object of class "numeric" not used
xmax: Object of class "numeric" not used
weights: Object of class "numeric" weights used in location penalty for different coordinate direc-
tions
degree: not used
hmax: Object of class "numeric" maximal bandwidth
sigma2: Object of class "numeric" estimated error variance
scorr: Object of class "numeric" estimated spatial correlation
family: Object of class "character" distribution of y, can be any of c("Gaussian","Bernoulli","Poisson","Exponent-
shape: Object of class "numeric" possible shape parameter of distribution of y
lkern: Object of class "integer" location kernel, can be any of c("Triangle","Quadratic","Cubic","Plateau","Gaus-
defaults to "Triangle"
lambda: Object of class "numeric" scale parameter used in adaptation
ladjust: Object of class "numeric" factor to adjust scale parameter with respect to its predeter-
mined default.
aws: Object of class "logical" Adaptation by Propagation-Separation
memory: Object of class "logical" Adaptation by Stagewise Aggregation
homogen: Object of class "logical" detect regions of homogeneity (used to speed up the calcula-
tions) currently FALSE
earlystop: Object of class "logical" currently FALSE
vmodel: Object of class "character" variance model used currently "Gaussian"
vcoef: Object of class "numeric" contains NULL
call: Object of class "call" that created the object.
Description

The function enables testing of the propagation condition in order to select appropriate values for the parameter lambda in function aws.

Usage

awstestprop(dy, hmax, theta = 1, family = "Gaussian", lkern = "Triangle",
            aws = TRUE, memory = FALSE, shape = 2, homogeneous=TRUE, varadapt=FALSE,
            ladjust = 1, spmin=0.25, seed = 1, minlevel=1e-6, maxz=25, diffz=.5,
            maxni=FALSE, verbose=FALSE)

Arguments

dy Dimension of grid used in 1D, 2D or 3D. May also be specified as an array of values. In this case data are generated with parameters dy=mean(dy)+theta and the propagation condition is testet as if theta is the true parameter. This can be used to study properties for a slightly misspecified structural assumption.

hmax Maximum bandwidth.

theta Parameter determining the distribution in case of family %in% c("Poisson","Bernoulli")
family
family specifies the probability distribution. Default is family="Gaussian".
also implemented are "Bernoulli", "Poisson", "Exponential", "Volatility", "Variance" and "NCchi". family="Volatility" specifies a Gaussian distribution with expectation 0 and unknown variance. family="Volatility" specifies that p*y/theta is distributed as χ² with p=shape degrees of freedom. family="NCchi" uses a noncentral Chi distribution with p=shape degrees of freedom and noncentrality parameter theta.
lkern
character: location kernel, either "Triangle", "Plateau", "Quadratic", "Cubic" or "Gaussian"
aws
logical: if TRUE structural adaptation (AWS) is used.
memory
logical: if TRUE stagewise aggregation is used as an additional adaptation scheme.
shape
Allows to specify an additional shape parameter for certain family models. Currently only used for family="Variance", that is χ-Square distributed observations with shape degrees of freedom.
homogeneous
if homogeneous==FALSE and family==Gaussian then create heterogeneous variances according to a chi-squared distribution with number of degrees of freedom given by sphere
varadapt
if varadapt==TRUE use inverse of variance reduction instead of sum of weights in definition of statistical penalty.
ladjust
Factor to increase the default value of lambda
spmin
Determines the form (size of the plateau) in the adaptation kernel. Not to be changed by the user.
seed
Seed value for random generator.
minlevel
Minimum exceedence probability to use in contour plots.
maxz
Maximum of z-scale in plots.
diffz
Gridlength in z
maxni
If TRUE use max_{l=k}(N_i^{(l)}) instead of (N_i^{(k)}) in the definition of the statistical penalty.
verbose
If TRUE provide additional information.

Details
Estimates exceedence probabilities
Results for intermediate steps are provided as contour plots. For a good choice of lambda (ladjust) the contours up to probabilities of 1e-5 should be vertical.

Value
A list with components
h
Sequence of bandwidths used
z
seq(θ, 3θ, .5), the quantiles exceedence probabilities refer to
prob
the matrix of exceedence probabilities, columns corresponding to h
probna
the matrix of exceedence probabilities for corresponding nonadaptive estimates, columns corresponding to h
The function performs a binning in 1D, 2D or 3D.

Usage

```
binning(x, y, nbins, xrange = NULL)
```

Arguments

- `x`: design matrix, dimension n x d.
- `y`: either a response vector of length n or NULL.
- `nbins`: vector of length d containing number of bins for each dimension, may be set to NULL.
- `xrange`: range for endpoints of bins for each dimension, either matrix of dimension 2 x d or NULL. `xrange` is increased if the cube defined does not contain all design points.

Value

A list with components

- `x`: matrix of coordinates of non-empty bin centers.
- `x.freq`: number of observations in nonempty bins.
- `midpoints.x1`: Bin centers in dimension 1.
- `midpoints.x2`: if d=1 Bin centers in dimension 2.
- `midpoints.x3`: if d=2 Bin centers in dimension 3.
- `breaks.x1`: Break points dimension 1.
- `breaks.x2`: if d>1 Break points dimension 2.
- `breaks.x3`: if d>2 Break points dimension 3.
- `table.freq`: number of observations per bin.
- `means`: if !is.null(y) mean of y in non-empty bins.
- `devs`: if !is.null(y) standard deviations of y in non-empty bins.
Note
This function has been adapted from the code of function binning in package sm.

Author(s)
Joerg Polzehl, <polzehl@wias-berlin.de>

See Also
See Also as aws.irreg

---

### Description
The method extract and/or compute specified statistics from object of class "aws","awssegment", \*ICI\*smooth and "kernsm".

### Usage
```r
# S4 method for signature 'aws'
extract(x, what="y")
# S4 method for signature 'awssegment'
extract(x, what="y")
# S4 method for signature 'ICIsmooth'
extract(x, what="y")
# S4 method for signature 'kernsm'
extract(x, what="y")
```

### Arguments
- `x` object
- `what` Statistics to extract, defaults to `what="y"` corresponding to the original data (response variable). Alternatives are `what="yhat"` for the smoothed response, `what="vhat"` for the estimated variance of the smoothed response, `what="sigma2"` for the estimated error variance of the original data, `what="vred"` for the variance reduction achieved and in case of signature(x = "ICIsmooth") `what="hbest"` for the selected bandwidth. A vector of any of these choices may be provided.

### Methods
- `signature(x = "ANY")` Returns a message that method extract is not defined.
- `signature(x = "aws")` Returns a list with components containing the requested statistics. Component names correspond to `tolower(what)`
- `signature(x = "awssegment")` Returns a list with components containing the requested statistics. Component names correspond to `tolower(what)`
signature(x = "ICIsmooth") Returns a list with components containing the requested statistics. Component names correspond to `tolower(what)`.

signature(x = "kernsm") Returns a list with components containing the requested statistics. Component names correspond to `tolower(what)`.

---

**ICIcombined**

*Adaptive smoothing by Intersection of Confidence Intervals (ICI) using multiple windows*

**Description**

The function performs adaptive smoothing by Intersection of Confidence Intervals (ICI) using multiple windows as described in Katkovnik et al (2006).

**Usage**

```r
ICIcombined(y, hmax, hinc = 1.45, thresh = NULL, kern = "Gaussian", m = 0, sigma = NULL, nsector = 1, symmetric = FALSE, presmooth = FALSE)
```

**Arguments**

- **y** Object of class "array" containing the original (response) data on a grid
- **hmax** maximum bandwidth
- **hinc** factor used to increase the bandwidth from scale to scale
- **thresh** threshold used in tests to determine the best scale
- **kern** Determines the kernel function. Object of class "character" kernel, can be any of c("Gaussian","Uniform","Triangle","Epanechnikov","Biweight","Triweight"). Defaults to kern="Gaussian".
- **m** Object of class "integer" vector of length length(dy) determining the order of derivatives specified for the coordinate directions.
- **sigma** error standard deviation
- **nsector** number of sectors to use.
- **symmetric** Object of class "logical" determines if sectors are symmetric with respect to the origin.
- **presmooth** Object of class "logical" determines if bandwidths are smoothed for more stable results.

**Details**

This mainly follows Chapter 6.2 in Katkovnik et al (2006).

**Value**

An object of class ICIsmooth
Author(s)
Joerg Polzehl <polzehl@wias-berlin.de>

References

See Also
ICIsmooth, ICIsmooth-class, kernsm

ICIsmooth

Adaptive smoothing by Intersection of Confidence Intervals (ICI)

Description
The function performs adaptive smoothing by Intersection of Confidence Intervals (ICI) as described in Katkovnik et al (2006)

Usage
`ICIsmooth(y, hmax, hinc = 1.45, thresh = NULL, kern = "Gaussian", m = 0, sigma = NULL, 
nsector = 1, sector = 1, symmetric = FALSE, presmooth = FALSE)`

Arguments
- `y` Object of class "array" containing the original (response) data on a grid
- `hmax` maximum bandwidth
- `hinc` factor used to increase the bandwidth from scale to scale
- `thresh` threshold used in tests to determine the best scale
- `kern` Determines the kernel function. Object of class "character" kernel, can be any of c("Gaussian", "Uniform", "Triangle", "Epanechnicov", "Biweight", "Triweight"). Defaults to kern="Gaussian".
- `m` Object of class "integer" vector of length length(dy) determining the order of derivatives specified for the coordinate directions.
- `sigma` error standard deviation
- `nsector` number of sectors to use. Positive weights are restricted to the sector selected by sector
- `sector` Object of class "integer" between 1 and nsector. sector used.
- `symmetric` Object of class "logical" determines if sectors are symmetric with respect to the origin.
- `presmooth` Object of class "logical" determines if bandwidths are smoothed for more stable results.
ICIsmooth-class

Details
This mainly follows Chapter 6.1 in Katkovnik et al (2006).

Value
An object of class ICIsmooth

Author(s)
Joerg Polzehl <polzehl@wias-berlin.de>

References

See Also
ICIcombined, ICIsmooth-class, kernsm

ICISMOOTH-class

Class "ICISMOOTH"

Description
The "ICISMOOTH" class is used for objects obtained by functions ICISMOOTH and ICIcombined.

Objects from the Class
Objects can be created by calls of the form new("ICISMOOTH", ...) or by functions ICISMOOTH and ICIcombined.

Slots
.Data: Object of class "list", usually empty.
y: Object of class "array" containing the original (response) data
dy: Object of class "numeric" dimension attribute of y
x: Object of class "numeric" if provided the design points
hmax: Object of class "numeric" maximum bandwidth
hinc: Object of class "numeric" initial bandwidth
thresh: Object of class "numeric" threshold used for bandwidth selection
kern: Object of class "character" kernel, can be any of c("Gaussian", "Uniform", "Triangle", "Epanechnicov", "Biweight". Defaults to kern="Gaussian".)
m: Object of class "integer" vector of length length(dy) determining the order of derivatives specified for the coordinate directions.
nsector: Object of class "integer" number of sectors to use.
sector: Object of class "integer" sector used.
symmetric: Object of class "logical" sectors are symmetric with respect to the origin.
yhat: Object of class "array" smoothed response variable
vhat: Object of class "array" estimated variance of smoothed response variable
hbest: Object of class "array" selected bandwidth(s))
sigma: Object of class "numeric" estimated standard deviation of errors in y
call: Object of class "call" that created the object.

Methods

extract signature(x = "ICISmooth"): ...

risk signature(y = "ICISmooth"): ...

plot Method for Function 'plot' in Package 'aws'.

show Method for Function 'show' in Package 'aws'.

print Method for Function 'print' in Package 'aws'.

summary Method for Function 'summary' in Package 'aws'.

Author(s)

Joerg Polzehl <polzehl@wias-berlin.de>

References


See Also

ICISmooth, ICICombined, kernsm, aws

Examples

showClass("ICISmooth")
kernsm

Kernel smoothing on a 1D, 2D or 3D grid

Description

Performs Kernel smoothing on a 1D, 2D or 3D grid by fft

Usage

kernsm(y, h = 1, kern = "Gaussian", m = 0, nsector = 1, sector = 1, symmetric = FALSE)

Arguments

- **y**: Object of class "array" containing the original (response) data on a grid
- **h**: bandwidth
- **kern**: Determines the kernel function. Object of class "character" kernel, can be any of c("Gaussian", "Uniform", "Triangle", "Epanechnicov", "Biweight", "Triweight"). Defaults to kern="Gaussian"
- **m**: Object of class "integer" vector of length length(dy) determining the order of derivatives specified for the coordinate directions.
- **nsector**: number of sectors to use. Positive weights are restricted to the sector selected by sector.
- **sector**: Object of class "integer" between 1 and nsector. sector used.
- **symmetric**: Object of class "logical" determines if sectors are symmetric with respect to the origin. ~~Describe symmetric here~~

Details

In case of any(m>0) derivative kernels are generated and applied for the corresponding coordinate directions. If nsector>1 the support of the kernel is restricted to a circular sector determined by sector.

Value

An object of class kernsm

Author(s)

Joerg Polzehl <polzehl@wias-berlin.de>

References

See Also

kernsm-class, ICISmooth, ICICombined

---

**kernsm-class**  
Class "kernsm"

---

**Description**

This class refers to objects created by function kernsm. These objects contain

**Objects from the Class**

Objects can be created by calls of the form `new("kernsm", ...)`. They are usually created by a call to function `kernsm`.

**Slots**

- `.Data`: Object of class "list", usually empty.
- `y`: Object of class "array" containing the response in nonparametric regression. The design is assumed to be a 1D, 2D or 3D grid, with dimensionality determined by `dim(y)`.
- `dy`: Object of class "numeric" containing `dim(y)`.
- `x`: Object of class "numeric" currently not used.
- `h`: Object of class "numeric" containing the bandwidth employed.
- `kern`: Object of class "character" determining the kernel that was used, can be one of c("Gaussian","Uniform","Triangle","Epanechnikov","Biweight","Triweight").
- `m`: Object of class "integer" with length `length(dy)` determining the order of derivatives in the corresponding coordinate directions. If `m[i]>0` a derivative kernel derived from `kern` has been used for the corresponding coordinate direction.
- `nsector`: Object of class "integer". If `nsector>1` positive weights are restricted to a segment of a circle (1D or 2D only). The segment is given by `sector`.
- `sector`: Object of class "integer" containing the number of the segment used in case of `nsector>1`.
- `symmetric`: Object of class "logical" determines if the sector is mirrored at the origin.
- `yhat`: Object of class "array" with same size and dimension as `y` providing the convolution of `y` with the chosen kernel.
- `vred`: Object of class "array" Variance reduction achieved by convolution assuming independence.
- `call`: Object of class "function", call that created the object.

**Methods**

- `extract` signature(`x = "aws"`): ...
- `risk` signature(`y = "aws"`): ...
- `plot` Method for Function ‘plot’ in Package ‘aws’.
- `show` Method for Function ‘show’ in Package ‘aws’.
- `print` Method for Function ‘print’ in Package ‘aws’.
lpaws

Author(s)
J"org Polzehl <polzehl@wias-berlin.de>

See Also
kernsm, ICIsmooth, ICIconbined, ICIsmooth

Examples
showClass("kernsm")

Description
The function allows for structural adaptive smoothing using a local polynomial (degree <=2) structural assumption. Response variables are assumed to be observed on a 1 or 2 dimensional regular grid.

Usage
lpaws(y, degree = 1, hmax = NULL, aws = TRUE, memory = FALSE, lkern = "Triangle", homogen = TRUE, earlystop = TRUE, aggkern = "Uniform", sigma2 = NULL, hw = NULL, ladjust = 1, u = NULL, graph = FALSE, demo = FALSE)

Arguments
y Response, either a vector (1D) or matrix (2D). The corresponding design is assumed to be a regular grid in 1D or 2D, respectively.
degree Polynomial degree of the local model
hmax maximal bandwidth
aws logical: if TRUE structural adaptation (AWS) is used.
memory logical: if TRUE stagewise aggregation is used as an additional adaptation scheme.
lkern character: location kernel, either "Triangle", "Plateau", "Quadratic", "Cubic" or "Gaussian"
homogen logical: if TRUE the function tries to determine regions where weights can be fixed to 1. This may increase speed.
earlystop logical: if TRUE the function tries to determine points where the homogeneous region is unlikely to change in further steps. This may increase speed.
aggkern character: kernel used in stagewise aggregation, either "Triangle" or "Uniform"
sigma2 Error variance, the value is estimated if not provided.
Regularisation bandwidth, used to prevent from unidentifiability of local estimates for small bandwidths.

Factor to increase the default value of lambda

A "true" value of the regression function, may be provided to report risks at each iteration. This can be used to test the propagation condition with u=0.

Logical: If TRUE intermediate results are illustrated graphically. May significantly slow down the computations in 2D. Please avoid using the default if the system build with cairo, use instead (faster by a factor of 30).

Logical: if TRUE wait after each iteration

Returns an object of class `aws` with slots

- `y = "numeric"` y
- `dy = "numeric"` dim(y)
- `x = "numeric"` numeric(0)
- `ni = "integer"` integer(0)
- `mask = "logical"` logical(0)
- `theta = "numeric"` Estimates of regression function and derivatives, length: length(y)*(degree+1)
- `mae = "numeric"` Mean absolute error for each iteration step if u was specified, numeric(0) else
- `var = "numeric"` approx. variance of the estimates of the regression function. Please note that this does not reflect variability due to randomness of weights.
- `xmin = "numeric"` numeric(0)
- `xmax = "numeric"` numeric(0)
- `wghts = "numeric"` numeric(0)
- `degree = "integer"` degree
- `hmax = "numeric"` effective hmax
- `sigma2 = "numeric"` provided or estimated error variance
- `scorr = "numeric"` 0
- `family = "character"` "Gaussian"
shape = "numeric"
    numeric(0)
lkern = "integer"
    integer code for lkern, 1="Plateau", 2="Triangle", 3="Quadratic", 4="Cubic", 5="Gaussian"
lambda = "numeric"
    effective value of lambda
ladjust = "numeric"
    effective value of ladjust
aws = "logical"
    aws
memory = "logical"
    memory
homogen = "logical"
    homogen
earlystop = "logical"
    earlustop
vmodel = "character"
    "Constant"
vcoef = "numeric"
    numeric(0)
call = "function"
    the arguments of the call to lpaws

Note

If you specify graph=TRUE for 2D problems avoid using the default X11() on systems build with cairo, use X11(type="Xlib") instead (faster by a factor of 30).

Author(s)

Joerg Polzehl <polzehl@wias-berlin.de>

References


See Also

link{awsdata}, aws, aws.irreg

Examples

library(aws)
# 1D local polynomial smoothing
## Not run: demo(lpaws_ex1)
# 2D local polynomial smoothing
plot-methods

Methods for Function ‘plot’ from package ‘graphics’ in Package ‘aws’

Description
Visualization of objects of class "aws", "awssgment", "kernsm" and "ICISmooth"

Methods
signature(x = "ANY") Generic function: see plot.
signature(x = "aws") Visualization of objects of class "aws"
signature(x = "awssgment") Visualization of objects of class "awssgment"
signature(x = "ICISmooth") Visualization of objects of class "ICISmooth"
signature(x = "kernsm") Visualization of objects of class "kernsm"

Author(s)
J"org Polzehl <polzehl@wias-berlin.de>

See Also
aws, awssgment, ICISmooth kernsm

print-methods

Methods for Function ‘print’ from package ‘base’ in Package ‘aws’

Description
The function provides information on data dimensions, creation of the object and existing slot-names for objects of class "aws", "awssgment", "ICISmooth" and "kernsm"

Methods
signature(x = "ANY") Generic function: see print.
signature(x = "aws") Provide information on data dimensions, creation of the object and existing slot-names for objects of class "aws"
signature(x = "awssgment") Provide information on data dimensions, creation of the object and existing slot-names for objects of class "awssgment"
signature(x = "ICISmooth") Provide information on data dimensions, creation of the object and existing slot-names for objects of class "ICISmooth"
signature(x = "kernsm") Provide information on data dimensions, creation of the object and existing slot-names for objects of class "kernsm"
**Description**

Methods function `risk` in package `aws`. For a given array `u` the following statistics are computed:
- Root Mean Squared Error (RMSE) \( \text{rmse} \approx \sqrt{\text{mean}(y-u)^2} \),
- Signal to Noise Ratio (SNR) \( \text{snr} \approx 10 \log(\text{mean}(u^2)/\text{MSE}, 10) \),
- Peak Signal to Noise Ratio (PSNR) \( \text{psnr} \approx 10 \log(\max(u^2)/\text{MSE}, 10) \),
- Mean Absolute Error (MAE) \( \text{mae} \approx \text{mean}(\text{abs}(y-u)) \),
- Maximal Absolute Error (MaxAE) \( \text{maxae} \approx \max(\text{abs}(y-u)) \),
- Universal Image Quality Index (UIQI) (Wang and Bovik (2002)).

**Usage**

```r
## S4 method for signature 'array'
risk(y, u=0)
## S4 method for signature 'aws'
risk(y, u=0)
## S4 method for signature 'awssegment'
risk(y, u=0)
## S4 method for signature 'ICIsmooth'
risk(y, u=0)
## S4 method for signature 'kernsm'
risk(y, u=0)
## S4 method for signature 'numeric'
risk(y, u=0)
```

**Arguments**

- `y` object
- `u` array of dimension `dim(y)` or `dim(extract(y, what="yhat")$y)` or scalar value used in comparisons.

**Methods**

- `signature(y = "ANY")` The method `extract` and/or `compute` specified statistics from object of class
- `signature(y = "array")` Returns a list with components `RMSE`, `SNR`, `PSNR`, `MAE`, `MaxAE`, `UIQI`
- `signature(y = "aws")` Returns a list with components `RMSE`, `SNR`, `PSNR`, `MAE`, `MaxAE`, `UIQI`
- `signature(y = "awssegment")` Returns a list with components `RMSE`, `SNR`, `PSNR`, `MAE`, `MaxAE`, `UIQI`
show-methods

signature(y = "ICIsmooth") Returns a list with components RMSE, SNR, PSNR, MAE, MaxAE, UIQI
signature(y = "kernsm") Returns a list with components RMSE, SNR, PSNR, MAE, MaxAE, UIQI
signature(y = "numeric") Returns a list with components RMSE, SNR, PSNR, MAE, MaxAE, UIQI

Author(s)
Joerg Polzehl <polzehl@wias-berlin.de>

References

Description
The function provides information on data dimensions, data source and existing slot-names for objects of class "aws", "awssegment", "ICIsmooth" and "kernsm" in package aws.

Methods
signature(object = "ANY") Generic function.
signature(object = "aws") Provide information on data dimensions, data source and existing slot-names for objects of class "dti" and classes that extent "aws".
signature(object = "awssegment") Provide information on data dimensions, data source and existing slot-names for objects of class "dti" and classes that extent "awssegment".
signature(object = "ICIsmooth") Provide information on data dimensions, data source and existing slot-names for objects of class "dti" and classes that extent "ICIsmooth".
signature(object = "kernsm") Provide information on data dimensions, data source and existing slot-names for objects of class "dti" and classes that extent "kernsm".

Author(s)
Karsten Tabelow <tabelow@wias-berlin.de>
Joerg Polzehl <polzehl@wias-berlin.de>

See Also
aws, awssegment, ICIsmooth kernsm
**summary-methods**

Methods for Function ‘summary’ from package ‘base’ in Package ‘aws’

**Description**

The method provides summary information for objects of class "aws".

**Arguments**

- **object**
  Object of class "dti","dtiData","dtiTensor","dwiMixtensor","dtiIndices", "dwiQball" or "dwiFiber".

- **...**
  Additional arguments in ... are passed to function quantile, e.g. argument `probs` may be specified here.

**Methods**

- signature(object = "ANY") Generic function: see *summary*.
- signature(object = "aws") The function provides summary information for objects of class "aws"
- signature(object = "awssegment") The function provides summary information for objects of class "awssegment"
- signature(object = "ICISmooth") The function provides summary information for objects of class "ICISmooth"
- signature(object = "kernsm") The function provides summary information for objects of class "kernsm"

**Author(s)**

Jörg Polzehl <polzehl@wias-berlin.de>

**See Also**

aws, awssegment, ICISmooth kernsm
Index

*Topic **classes**
  aws-class, 7
  awssegment-class, 19
  ICisMOOTH-class, 27
  kernsm-class, 30
*Topic **hplot**
  plot-methods, 34
*Topic **manip**
  awndata, 18
  binning, 23
  extract-methods, 24
*Topic **methods**
  extract-methods, 24
  plot-methods, 34
  print-methods, 34
  risk-methods, 35
  show-methods, 36
  summary-methods, 37
*Topic **nonparametric**
  awndata, 18
  awsssegment, 34–37
  awsssegment-class, 19
  awstestprop, 21
  awstestprop, 21
  binning, 23
*Topic **package**
  awspackage, 2
*Topic **regression**
  awndata, 18
  awsssegment, 34–37
  awsssegment-class, 19
  awstestprop, 21
*Topic **smooth**
INDEX

ICISmooth, 26, 26, 28, 30, 31, 34–37
ICISmooth-class, 27
kernsm, 26–28, 29, 31, 34–37
kernsm-class, 30
lpaws, 7, 9, 15, 31
plot, 34
plot, ANY-method (plot-methods), 34
plot, aws-method (plot-methods), 34
plot, awssegment-method (plot-methods), 34
plot, ICISmooth-method (plot-methods), 34
plot, kernsm-method (plot-methods), 34
plot-methods, 34
print, 34
print, ANY-method (print-methods), 34
print, aws-method (print-methods), 34
print, awssegment-method (print-methods), 34
print, ICISmooth-method (print-methods), 34
print, kernsm-method (print-methods), 34
print-methods, 34
risk, ANY-method (risk-methods), 35
risk, array-method (risk-methods), 35
risk, aws-method (risk-methods), 35
risk, awssegment-method (risk-methods), 35
risk, ICISmooth-method (risk-methods), 35
risk, kernsm-method (risk-methods), 35
risk, numeric-method (risk-methods), 35
risk-methods, 35
show, ANY-method (show-methods), 36
show, aws-method (show-methods), 36
show, awssegment-method (show-methods), 36
show, ICISmooth-method (show-methods), 36
show, kernsm-method (show-methods), 36
show-methods, 36
summary, 37
summary, ANY-method (summary-methods), 37
summary, aws-method (summary-methods), 37
summary, awssegment-method (summary-methods), 37
summary, ICISmooth-method (summary-methods), 37
summary, kernsm-method (summary-methods), 37
summary-methods, 37