Package ‘aws’

July 30, 2018

Version 2.2-0
Date 2018-07-30
Title Adaptive Weights Smoothing
Author Joerg Polzehl [aut, cre],
      Felix Anker [ctb]
Maintainer Joerg Polzehl <joerg.polzehl@wias-berlin.de>
Depends R (>= 3.2.0), methods, awsMethods (>= 1.0-1), gsl
Description We provide a collection of R-functions implementing
    adaptive smoothing procedures in 1D, 2D and 3D. This includes the
    Propagation-Separation Approach to adaptive smoothing as
    described in ``J. Polzehl and V. Spokoiny (2006)
    <DOI:10.1007/s00440-005-0464-1>'',
    the Intersecting Confidence Intervals (ICI), variational approaches and a non-local means filter.
License GPL (>= 2)
Copyright This package is Copyright (C) 2005-2018 Weierstrass
Institute for Applied Analysis and Stochastics.
URL http://www.wias-berlin.de/people/polzehl/
RoxygenNote 5.0.1
NeedsCompilation yes
Repository CRAN
Date/Publication 2018-07-30 16:00:05 UTC

R topics documented:

aws-package ............................................................... 2
aws ................................................................. 4
aws-class .......................................................... 8
aws.gaussian ....................................................... 10
aws.irreg ............................................................ 13
aws-package

Description


Details

The DESCRIPTION file:

Package:      aws
Version:      2.2-0
Date:         2018-07-30
Title:        Adaptive Weights Smoothing
Authors@R:    c(person("Joerg","Polzehl",role=c("aut","cre"),email="joerg.polzehl@wias-berlin.de"),person("Felix","Anker"))
Author: Joerg Polzehl [aut, cre], Felix Anker [ctb]
Maintainer: Joerg Polzehl <joerg.polzehl@wias-berlin.de>
Depends: R (>= 3.2.0), methods, awsMethods (>= 1.0-1), gsl
Description: We provide a collection of R-functions implementing adaptive smoothing procedures in 1D, 2D and 3D. This includes local constant models on a grid, adaptive weights smoothing for Gaussian data with variance depending on the mean, local constant AWS for irregular (1D/2D) design, segmentation by adaptive weights for Gaussian models, and non-local means filter.
License: GPL (>=2)
Copyright: This package is Copyright (C) 2005-2018 Weierstrass Institute for Applied Analysis and Stochastics.
URL: http://www.wias-berlin.de/people/polzehl/
RoxygenNote: 5.0.1

Index of help topics:

ICIcombined Adaptive smoothing by Intersection of Confidence Intervals (ICI) using multiple windows
ICIsMOOTH Adaptive smoothing by Intersection of Confidence Intervals (ICI)
ICIsMOOTH-class Class "ICIsMOOTH"
TV_denoising TV/TGV denoising of image data
aws AWS for local constant models on a grid
aws-class Class "aws"
aws-package Adaptive Weights Smoothing
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
awsssegment-class Class "awsssegment"
awstestprop Propagation condition for adaptive weights smoothing
awsweights Generate weight scheme that would be used in an additional aws step
binning Binning in 1D, 2D or 3D
extract-methods Methods for Function 'extract' in Package 'aws'
kernsm Kernel smoothing on a 1D, 2D or 3D grid
kernsm-class Class "kernsm"
lpaws Local polynomial smoothing by AWS
nlmeans NLMeans filter in 1D/2D/3D
paws Adaptive weights smoothing using patches
plot-methods Methods for Function 'plot' from package 'graphics' in Package 'aws'
print-methods Methods for Function 'print' from package 'base' in Package 'aws'
qmeasures Quality assessment for image reconstructions.
risk-methods Compute risks characterizing the quality of smoothing results
show-methods Methods for Function 'show' in Package 'aws'
Methods for Function 'summary' from package 'base' in Package 'aws'

vaws

vector valued version of function 'aws' The function implements the propagation separation approach to nonparametric smoothing (formerly introduced as Adaptive weights smoothing) for varying coefficient likelihood models with vector valued response on a 1D, 2D or 3D grid.

vpaws

vector valued version of function 'paws' with homogeneous covariance structure

Author(s)

Joerg Polzehl [aut, cre], Felix Anker [ctb]

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

References


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

aws(y,hmax=NULL,aws=TRUE,mem=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 contains the observed response data. dim(y) determines the dimensionality and extend of the grid design.

hmax specifies the maximal bandwidth. Defaults to hmax=250, 12, 5 for dd=1, 2, 3, respectively. In case of l Kern="Gaussian" the bandwidth is assumed to be given in full width half maximum (FWHM) units, i.e., 0.42466 times gridsize.

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

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

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 \( \chi^2 \) 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"

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 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 length(sigma2)==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 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.
If set this provides diagnostics for testing the propagation condition. The values of y should correspond to the specified family and a global model.

If TRUE use \( \max_{l\leq k} (N_i^{(l)}) \) instead of \( (N_i^{(k)}) \) 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 sigma2. 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. 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

\[
E|\hat{\theta}_k - \theta| \leq (1 + \alpha)E|\hat{\theta}_k - \theta|
\]

where \( \hat{\theta}_k \) is the aws-estimate in step \( k \) and \( \hat{\theta}_k \) is corresponding nonadaptive estimate using the same bandwidth (\( \lambda = \inf \)). The value of \( \lambda \) can be adjusted by specifying the factor \( \lambda_{\text{adjust}} \). Values \( \lambda_{\text{adjust}} > 1 \) lead to an less effective adaptation while \( \lambda_{\text{adjust}} < 1 \) may lead to random segmentation of, with respect to a constant model, homogeneous regions.

The numerical complexity of the procedure is mainly determined by \( h_{\max} \). The number of iterations is approximately \( \text{Const} \times d \times \log(h_{\max}) / \log(1.25) \) with \( d \) being the dimension of \( y \) and the constant depending on the kernel \( l_{\text{kernel}} \). 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. \( h_{\max} \) determines the maximal possible variance reduction.

**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)
- `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.
aws

xmin = "numeric"
   numeric(0)
xmax = "numeric"
   numeric(0)
wgths = "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
varmodel = "character"
   "Constant"
vcoef = "numeric"
   numeric(0)
call = "function"
   the arguments of the call to aws

Note

use setCores='number of threads' to enable parallel execution.

Author(s)

Joerg Polzehl,<polzehl@wias-berlin.de>, http://www.wias-berlin.de/people/polzehl/
References


See Also

See also paws, lpaws, vaws, 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)
- `hseq`: Sequence of bandwidths employed.
mae: Object of class "numeric" Mean absolute error with respect to array in argument u if provided.

psnr: Object of class "numeric" Peak Signal to Noise Ratio (PSNR) with respect to array in argument u if provided.

var: Object of class "numeric" pointwise variance of \( \theta[\ldots,1] \)

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","Volatility","Variance")

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 weigths 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>
References


See Also

aws, lpaaws, 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 an 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 extend 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" refering to a polynomial model of degree 0 to 2.
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.

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 chosen to fulfill 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 $E|\hat{\theta}^k - \theta| \leq (1 + \alpha)E|\tilde{\theta}^k - \theta|$ where $\hat{\theta}^k$ is the `aws`-estimate in step $k$ and $\tilde{\theta}^k$ is corresponding nonadaptive estimate using the same bandwidth (`lambda`=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 `lkern`. 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
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
aws.irreg

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/people/polzehl/

References


See Also

See also aws, link(awsdata), aws.irreg

Examples

require(aws)

aws.irreg(x, y, 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
- **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
- **dy**: numeric
- **x**: numeric
- **ni**: integer
- **mask**: logical
- **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
earlystop = "logical"
    FALSE
varmodel = "character"
    varmodel
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

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

aws.segment(y, level, delta = 0, hmax = NULL, hpre = NULL, varmodel = "Constant", lkern = "Triangle", scorr = 0, ladjust = 1, wgths = 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" referring 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).

`ladjust` factor to increase the default value of lambda

`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.

`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 $\geq \text{level} + \delta$ and $H_2$ value $\leq \text{level} + \delta$. Pixel where the first hypothesis is rejected are classified as $1$ (segment 1) while rejection of $H_2$ results in classification $3$ (segment 3). Pixel where neither $H_1$ or $H_2$ are rejected are assigned to a value $0$ (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. Duembgen 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

```r
y = "numeric"  y
dy = "numeric"  dim(y)
x = "numeric"  numeric(0)
i = "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
```
approx. variance of the estimates of the regression function. Please note that this does not reflect variability due to randomness of weights.

var = "numeric"

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

varmodel = "character"
varmodel

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/people/polzehl/

References

See Also
aws, aws.gaussian

Examples
require(aws)

awsdata Extract information from an object of class aws

Description
Extract data and estimates from an object of class aws

Usage
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 provided.
var: Object of class "numeric" pointwise variance of theta[,...,1]
xmin: Object of class "numeric" not used
xmax: Object of class "numeric" not used
wghts: Object of class "numeric" weights used in location penalty for different coordinate directions
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","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) currently FALSE
earlystop: Object of class "logical" currently FALSE
varmodel: 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.
Methods

- **extract** signature(x = "awssegment"): ...
- **plot** signature(x = "awssegment"): ...
- **print** signature(x = "awssegment"): ...
- **risk** signature(y = "awssegment"): ...
- **show** signature(object = "awssegment"): ...
- **summary** signature(object = "awssegment"): ...

Author(s)

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

See Also

- **aws.segment**

Examples

```
sFromClass("awssegment")
```

---

**awstestprop**  
Propagation condition for adaptive weights smoothing

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 \( \chi^2 \) 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 \( \chi^2 \)-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 maxl<<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(0, 0.5), 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
**awsweights**

Generate weight scheme that would be used in an additional aws step

**Description**

Utility function to create a weighting scheme for an additional aws step. Intended to be used for illustrations only.

**Usage**

```r
awsweights(awsobj, spmin = 0.25, inx = NULL)
```

**Arguments**

- `awsobj` object obtained by a call to function `aws`
- `spmin` Size of the plateau in the adaptation kernel.
- `inx` either a matrix of dimension `length(awsobj@dy) x number of points containing the integer coordinates of points of interest or NULL. In the latter case the weight scheme for all points is generated.

**Value**

an array of either dimension `awsobj@dy x number of points` or `awsobj@dy x awsobj@dy`

**Author(s)**

Joerg Polzehl, <polzehl@wias-berlin.de>, [http://www.wias-berlin.de/people/polzehl/](http://www.wias-berlin.de/people/polzehl/)

**References**


**See Also**

See also `aws`
Description

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>
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)`.

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.
ICIconbined

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

ICIconbined(y, hmax, hinc = 1.45, thresh = NULL, kern = "Gaussian", m = 0,
       sigma = NULL, nsector = 1, symmetric = FALSE, presmooth = FALSE,
       combine = "weighted", unit = c("SD","FWHM"))

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 direction.
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.
combine Either "weighted" or "minvar". Determines how whether to combine sectorial results a weighted (with inverse variance) mean or to chose the sectorial estimate with minimal variance.
unit How should the bandwidth be interpreted in case of a Gaussian kernel. For "SD" the bandwidth refers to the standard deviation of the kernel while "FWHM" interprets the bandwidth in terms of Full Width Half Maximum of the kernel.

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, unit = c("SD","FWHM"))

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. 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.
unit How should the bandwidth be interpreted in case of a Gaussian kernel. For "SD" the bandwidth refers to the standard deviation of the kernel while "FWHM" interprets the bandwidth in terms of Full Width Half Maximum of the kernel.

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
ICIconbined, ICISmooth-class, kernsm

Description
The "ICISmooth" class is used for objects obtained by functions ICISmooth and ICIconbined.

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

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’.

Author(s)

Joerg Polzehl <polzehl@wias-berlin.de>

References


See Also

ICIsmooth, ICIcombined, kernsm, aws
Examples

```r
showClass("ICIsmooth")
```

---

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

**Description**

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

**Usage**

```r
kernsm(y, h = 1, kern = "Gaussian", m = 0, nsector = 1, sector = 1,
       symmetric = FALSE, unit = c("SD","FWHM"))
```

**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","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.
- `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.
- `unit` How should the bandwidth be interpreted in case of a Gaussian kernel. For "SD" the bandwidth refers to the standard deviation of the kernel while "FWHM" interprets the bandwidth in terms of Full Width Half Maximum of the kernel.

**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, ICIComined

---

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

**Author(s)**

Jörg Polzehl <polzehl@wias-berlin.de>

**See Also**

- `kernsm`, `ICIsmooth`, `ICICombined`, `ICIsmooth`

**Examples**

```r
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**

```r
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.
hw Regularisation bandwidth, used to prevent from unidentifiability of local estimates for small bandwidths.
ladjust factor to increase the default value of lambda
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 logical: If TRUE intermediate results are illustrated graphically. May significantly slow down the computations in 2D. Please avoid using the default x11() on systems build with cairo, use x11(type="x11") instead (faster by a factor of 30).
demo logical: if TRUE wait after each iteration

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)

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
lpaws

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"
    eralustop
varmodel = "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

nlmeans

Description

Implements the Non-Local-Means Filter of Buades et al 2005

Usage

nlmeans(x, lambda, sigma, patchhw = 1, searchhw = 7, pd = NULL)

Arguments

x  
1, 2 or 3-dimensional array

lambda  
scale factor for kernel in image space

sigma  
error standard deviation (for additive Gaussian errors)

patchhw  
Half width of patches in each dimension (patchsize is \(2\times patchhw+1\)^d for d-dimensional array)

searchhw  
Half width of search area (size of search area is \(2\times searchhw+1\)^d for d-dimensional array)

pd  
If \(pd < (2\times patchhw+1)^d\) use pd principal components instead of complete patches.

Details

The implementation follows the description of the Non-Local-Means Filter of Buades et al 2005 on http://www.numerical-tours.com/matlab/steering_adv_6_nl_means/#biblio that incorporates dimension reduction for patch comparisons by PCA.
Value

A list of class "nlmeans" with components

theta       Denoised array
lambda      Scale parameter used
sigma       The error standard deviation
patchhw     Half width of patches
pd          Effective patchsize used
searchhw    Half width of search area

Note

use setCores='number of threads' to enable parallel execution.

Author(s)

Joerg Polzehl, <polzehl@wias-berlin.de>, http://www.wias-berlin.de/people/polzehl/

References

one. Simulation, 4, 490-530.

http://www.numerical-tours.com/matlab/denoisingadv_6_nl_means/#biblio

Description

The function implements a version the propagation separation approach that uses patches instead
of individual voxels for comparisons in parameter space. Functionality is analog to function aws.
Using patches allows for an improved handling of locally smooth functions and in 2D and 3D for
improved smoothness of discontinuities at the expense of increased computing time.

Usage

paws(y, hmax = NULL, onestep = FALSE, aws = TRUE, family = "Gaussian",
    lkern = "Triangle", aggkern = "Uniform", sigma2 = NULL, shape = NULL,
    scorr = 0, spmin = 0.25, ladjust = 1, wghts = NULL, u = NULL,
    graph = FALSE, demo = FALSE, maxni = FALSE, patchsize = 1,
    patchkrit = "max", pquant = NULL)
pawsm(y, mask, hmax = NULL, onestep = FALSE, aws = TRUE, family = "Gaussian",
    lkern = "Triangle", aggkern = "Uniform", sigma2 = NULL, shape = NULL,
    scorr = 0, spmin = 0.25, ladjust = 1, wghts = NULL, u = NULL,
    graph = FALSE, demo = FALSE, patchsize = 1)
Arguments

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

**mask**
logical array defining a mask. All computations are restricted to the mask.

**hmax**
hmax specifies the maximal bandwidth. Defaults to \(h_{\text{max}}=250, 12, 5\) for \(d=1, 2, 3\), respectively. In case of \(\text{lkern}="\text{Gaussian}"\) the bandwidth is assumed to be given in full width half maximum (FWHM) units, i.e., \(0.42466\) times gridsize.

**onestep**
apply the last step only (use for test purposes only)

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

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

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

**aggg Kern**
character: kernel used in stagewise aggregation, either "Triangle" or "Uniform"

**sigma2**
sigma2 allows to specify the variance in case of \(\text{family}="\text{Gaussian}"\). Not used if \(\text{family}!="\text{Gaussian}"\). Defaults to NULL. In this case a homoskedastic variance estimate is generated. If length(sigma2)==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 \(\text{family}="\text{Variance}"\), that is \(\chi^2\)-Square distributed observations with shape degrees of freedom.

**scorr**
The vector \(\text{scorr}\) 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

**wghths**
wghths 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.

**maxni**
If TRUE use \(\max_{l<k}(N^{(l)}_i)\) instead of \(N^{(k)}_i\) in the definition of the statistical penalty.
patchsize positive integer defining the size of patches. Number of grid points within the patch is \((2 \times \text{patchsize} + 1)^d\) with \(d\) denoting the dimensionality of the design.

patchkrit defaults to our recommended statistical penalty. Alternatives are "quantile" and numerical values (specifying a norm for patch comparisons). This argument is for internal use (tests) only.

pquant quantile to use with patchkrit="quantile". For internal use only.

Details

see \texttt{aws}. The procedure is supposed to produce superior results if the assumption of a local constant image is violated or if smoothness of discontinuities is desired.

Value

returns an object of class \texttt{aws} with slots:

\begin{itemize}
  \item \texttt{y = "numeric"} \ y
  \item \texttt{dy = "numeric"} \ dim(y)
  \item \texttt{x = "numeric"} \ numeric\(0\)
  \item \texttt{ni = "integer"} \ integer\(0\)
  \item \texttt{mask = "logical"} \ logical\(0\)
  \item \texttt{theta = "numeric"}
    \begin{itemize}
      \item Estimates of regression function, length: length\(\text{y}\)
    \end{itemize}
  \item \texttt{hseq = "numeric"}
    \begin{itemize}
      \item sequence of bandwidths employed
    \end{itemize}
  \item \texttt{mae = "numeric"}
    \begin{itemize}
      \item Mean absolute error for each iteration step if \(u\) was specified, numeric\(0\) else
    \end{itemize}
  \item \texttt{psnr = "numeric"}
    \begin{itemize}
      \item Peak signal-to-noise ratio for each iteration step if \(u\) was specified, numeric\(0\) else
    \end{itemize}
  \item \texttt{var = "numeric"}
    \begin{itemize}
      \item approx. variance of the estimates of the regression function. Please note that this does not reflect variability due to randomness of weights.
    \end{itemize}
  \item \texttt{xmin = "numeric"} \ numeric\(0\)
  \item \texttt{xmax = "numeric"} \ numeric\(0\)
  \item \texttt{wghts = "numeric"} \ numeric\(0\)
  \item \texttt{degree = "integer"} \ 0
  \item \texttt{hmax = "numeric"}
    \begin{itemize}
      \item effective hmax
    \end{itemize}
  \item \texttt{sigma2 = "numeric"}
    \begin{itemize}
      \item provided or estimated error variance
    \end{itemize}
\end{itemize}
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
vmodel = "character"
  "Constant"
vcoef = "numeric"
  numeric(0)
call = "function"
  the arguments of the call to aws

Note

use setCores='number of threads' to enable parallel execution.

Author(s)

Joerg Polzehl, <polzehl@wias-berlin.de>, http://www.wias-berlin.de/people/polzehl/

References


See Also

See also aws, lpaws, vpaws, link{awsdata}
Examples

```r
## Not run:
setCores(2)
y <- array(rnorm(64*3),c(64,64,64))
yhat <- pawsHyLhmax[V]

## End(Not run)
```

plot-methods

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

Description

Visualization of objects of class "aws", "awsssegment", "kernsm" and "ICIsmooth"

Methods

signature(x = "ANY") Generic function: see plot.
signature(x = "aws") Visualization of objects of class "aws"
signature(x = "awsssegment") Visualization of objects of class "awsssegment"
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, awsssegment, 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", "awsssegment", "ICIsmooth" and "kernsm"
qmeasures

Quality assessment for image reconstructions.

Description
Computes selected criteria for quality assessments of

Usage
qmeasures(img, ref, which = c("PSNR", "MAE", "MSE", "RMSE", "SSIM", "MAGE", "RMSGE"), mask = FALSE)

Arguments
- **img**: 2D/3D image, object of class "aws", "ICIsMOOTH", "kernsm", "nlmeans" or array.
- **ref**: Reference image (array, matrix or vector) for comparison.
- **which**: Criterion to use for Quality assessment. Please specify a subset of "PSNR" (Peak Signal to Noise Ratio), "MAE" (Mean Absolute Error), "MSE" (Mean Squared Error), "RMSE" (Root Mean Squared Error), "SSIM" (Structural SIMilarity), "MAGE" (Mean Absolute Gradient Error), "RMSGE" (Root Mean Squared Gradient Error).
- **mask**: Logical of same dimension as img/ref. Calculation can be restricted to mask.

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 = "awssegment") Provide information on data dimensions, creation of the object and existing slot-names for objects of class "awssegment"
- 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"

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

See Also
aws, awssegment, ICIsMOOTH kernsm
Details

Calculates specified quality indices.

Value

A vector with names as specified in which.

Author(s)

Joerg Polzehl, <polzehl@wias-berlin.de>, http://www.wias-berlin.de/people/polzehl/

Description

Methods function risk in package aws. For an given array u the following statistics are computed:
Root Mean Squared Error RMSE $\approx \sqrt{\text{mean}(y-u)^2)}$. Signal to Noise Ratio SNR $\approx 10\log\left(\text{mean}(u^2)/\text{MSE}, 10\right)$. Peak Signal to Noise Ratio PSNR $\approx 10\log\left(\max(u^2)/\text{MSE}, 10\right)$. Mean Absolute Error MAE $\approx \text{mean}(|y-u|)$. Maximal Absolute Error MaxAE $\approx \max(|y-u|)$. 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
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

V. Katkovnik, K. Egiazarian and J. Astola, Local Approximation Techniques in Signal And Image
Processing, SPIE Society of Photo-Optical Instrumentation Engin., 2006, PM157
N3, pp. 81-84, 2002.
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)

J\"org Polzehl <polzehl@wias-berlin.de>

See Also

- `aws`, `awssegment`, `ICISmooth kernsm`
TV_denoising

TV/TGV denoising of image data

Description

Total variation and total generalized variation are classical energy minimizing methods for image denoising.

Usage

TV_denoising(datanoisy, alpha, iter = 1000, tolmean = 1e-06, 
tolsup = 1e-04, scale = 1, verbose=FALSE)
TGV_denoising(datanoisy, alpha, beta, iter = 1000, tolmean = 1e-06, 
tolsup = 1e-04, scale = 1, verbose=FALSE)
TV_denoising_colour(datanoisy, alpha, iter = 1000, tolmean = 1e-06, 
tolsup = 1e-04, scale = 1, verbose=FALSE)
TGV_denoising_colour(datanoisy, alpha, beta, iter = 1000, tolmean = 1e-06, 
tolsup = 1e-04, scale = 1, verbose=FALSE)

Arguments

datanoisy  noisy 2D image data matrix. In case of TV_denoising_colour and TGV_denoising_colour and array with third dimension referring to RGB channels.
alpha  TV regularization parameter.
beta  additional TGV regularization parameter.
iter  max. number of iterations
tolmean  requested accuracy for mean image correction
tolsup  requested accuracy for max (over pixel) image correction
scale  image scale
verbose  report convergence diagnostics.

Details

Reimplementation of original matlab code by Kostas Papafitsoros (WIAS).

Value

TV/TGV reconstructed image data (2D array)

Author(s)

Joerg Polzehl, <polzehl@wias-berlin.de>, http://www.wias-berlin.de/people/polzehl/
vaws

vector valued version of function aws The function implements the propagation separation approach to nonparametric smoothing (formerly introduced as Adaptive weights smoothing) for varying coefficient likelihood models with vector valued response on a 1D, 2D or 3D grid.

Description

The function implements a version the propagation separation approach that uses vector valued instead of scalar responses.

Usage

vaws(y, kstar = 16, sigma2 = 1, mask = NULL, scorr = 0, spmin = 0.25, ladjust = 1, wghts = NULL, u = NULL, maxni = FALSE)
vawscov(y, kstar = 16, invcov = NULL, mask = NULL, scorr = 0, spmin = 0.25, ladjust = 1, wghts = NULL, u = NULL, maxni = FALSE)

Arguments

y y contains the observed response data. dim(y) determines the dimensionality and extend of the grid design. First component varies over components of the response vector.
kstar maximal number of steps to employ. Determines maximal bandwidth.
sigma2 specifies a homogeneous error variance.
invcov array of voxelwise inverse covariance matrixes, first index corresponds to upper diagonal inverse covariance matrix.
mask logical mask. All computations are restricted to design points within the mask.
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.

References

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 \( \hat{u} = 0 \).

If TRUE use \( \max_{l \leq k} \hat{N}_l(y) \) instead of \( \hat{N}_l^{(k)} \) in the definition of the statistical
penalty.

**Details**

see `aws`. Expcts vector valued responses. Currently only implements the case of additive Gaussian
errors.

**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)
- `theta = "numeric"` Estimates of regression function, length: \( \text{length}(y) \)
- `hseq = "numeric"` sequence of bandwidths employed
- `mae = "numeric"` Mean absolute error for each iteration step if \( \hat{u} \) was specified, numeric(0) else
- `psnr = "numeric"` Peak signal-to-noise ratio for each iteration step if \( \hat{u} \) 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 (inverse) 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
varmodel = "character"
    "Constant"
vcoef = "numeric"
    numeric(0)
call = "function"
    the arguments of the call to aws

Note
    use setCores='number of threads' to enable parallel execution.

Author(s)
    Joerg Polzehl, <polzehl@wias-berlin.de>, http://www.wias-berlin.de/people/polzehl/

References

See Also
    See also aws, vpaws, vpawscov, link{awsdata}
Examples

```r
## Not run:
setCores(2)
y <- array(rnorm(4*64*3),c(4,64,64))
yhat <- vaws(y,kstar=20)
## End(Not run)
```

vpaws  

---

vector valued version of function paws with homogeneous covariance structure

Description

The function implements a vector-valued version the propagation separation approach that uses patches instead of individual voxels for comparisons in parameter space. Functionality is analog to function vaws. Using patches allows for an improved handling of locally smooth functions and in 2D and 3D for improved smoothness of discontinuities at the expense of increased computing time.

Usage

```r
vpaws(y, kstar = 16, sigma2 = 1, mask = NULL, scorr = 0, spmin = 0.25,
ladjust = 1, wghts = NULL, u = NULL, maxni = FALSE, patchsize = 1)
vpawscov(y, kstar = 16, invcov = NULL, mask = NULL, scorr = 0, spmin = 0.25,
ladjust = 1, wghts = NULL, u = NULL, maxni = FALSE, patchsize = 1)
```

Arguments

- **y**: contains the observed response data. `dim(y)` determines the dimensionality and extend of the grid design. First component varies over components of the response vector.
- **kstar**: maximal number of steps to employ. Determines maximal bandwidth.
- **sigma2**: specifies a homogeneous error variance.
- **invcov**: array of voxelwise inverse covariance matrixes, first index corresponds to upper diagonal inverse covariance matrix.
- **mask**: logical mask. All computations are restricted to design points within the mask.
- **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**: 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.

maxni If TRUE use $\max_{l<z_k}(N_l^{(i)})$ instead of $(N_{z_k}^{(i)})$ in the definition of the statistical penalty.

patchsize a positive integer defining the size of patches. Number of grid points within the patch is $(2 \times \text{patchsize} + 1)^d$ with $d$ denoting the dimensionality of the design.

Details

see `vaws`. The procedure is supposed to produce superior results if the assumption of a local constant image is violated or if smoothness of discontinuities is desired.

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)
- `theta = "numeric"` Estimates of regression function, length: length(y)
- `hseq = "numeric"` sequence of bandwidths employed
- `mae = "numeric"` Mean absolute error for each iteration step if u was specified, numeric(0) else
- `psnr = "numeric"` Peak signal-to-noise ratio 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. Currently also uses factor 1/ni instead of the correct $\sum(w_{ij}^2)/ni^2$
- `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"
family = "character"
shape = "numeric"
lkern = "integer"
    integer code for lkern, 1="Plateau", 2="Triangle", 3="Quadratic", 4="Cubic", 5="Gaussian"
lambda = "numeric"
ladjust = "numeric"
aws = "logical"
memory = "logical"
homogen = "logical"
earlystop = "logical"
    FALSE
varmodel = "character"
    "Constant"
vcoef = "numeric"
    numeric(0)
call = "function"
    the arguments of the call to aws

Note
    use setCores='number of threads' to enable parallel execution.

Author(s)
    Joerg Polzehl, <polzehl@wias-berlin.de>, http://www.wias-berlin.de/people/polzehl/

References

See Also
    See also vaws, lpaws, vawscov, link{awsdata}
Examples

```r
## Not run:
setCores(2)
y <- array(rnorm(4*64*3),c(4,64,64))
yhat <- vpaws(y,kstar=20)

## End(Not run)
```
Index

*Topic classes
  aws-class, 8
  awssegment-class, 20
  ICIsMOOTH-class, 29
  kernsm-class, 32
*Topic hplot
  plot-methods, 41
*Topic manip
  awndata, 19
  binning, 25
  extract-methods, 26
*Topic methods
  extract-methods, 26
  plot-methods, 41
  print-methods, 41
  risk-methods, 43
  show-methods, 44
  summary-methods, 45
*Topic misc
  awsweight, 24
*Topic nonparametric
  aw, 4
  aw回暖gaussian, 10
  aw回暖irreg, 13
  aw回暖segment, 16
  awndata, 19
  awswtestprop, 22
  ICICOMBined, 27
  ICISMOOTH, 28
  kernsm, 31
  lpaws, 33
  paws, 37
  vaws, 47
  vpaws, 50
*Topic package
  awspackage, 2
*Topic regression
  aw, 4
  aw回暖gaussian, 10
  aw回暖irreg, 13
  aw回暖segment, 16
  awndata, 19
  awswtestprop, 22
  ICICOMBined, 27
  ICISMOOTH, 28
  kernsm, 31
  lpaws, 33
  paws, 37
  risk-methods, 43
  TV_denoising, 46
  vaws, 47
  vpaws, 50
*Topic utilities
  show-methods, 44
*Topic utilities
  print-methods, 41
  qmeasures, 42
  summary-methods, 45

aws, 4, 10, 13, 19, 20, 24, 30, 36, 37, 39–42, 45, 47–49
aws-class, 8
aws-package, 2
aws回暖gaussian, 8, 10, 10, 19
aws回暖irreg, 8, 10, 13, 13, 20, 26, 36
aws回暖segment, 16, 22
awndata, 19
awssegment, 41, 42, 45
awssegment-class, 20
awstestprop, 22
awsweight, 24
binner, 25
extract, ANY-method (extract-methods), 26
extract, aws-method (extract-methods), 26
extract, awsssegment-method
(extract-methods), 26
extract, ICISmooth-method
(extract-methods), 26
extract, kerIms-class
(extract-methods), 26
ICIcombined, 27, 29, 32, 33
ICISmooth, 28, 29, 32, 33, 41, 42, 45
ICISmooth-class, 29
kerIms, 28, 30, 31, 33, 41, 42, 45
kerIms-class, 32
lpaw, 8, 10, 16, 33, 40, 52
nlmeans, 36
paw, 8, 37, 50
pawsm(paw), 37
plot, 41
plot, ANY-method (plot-methods), 41
plot, aws-method (plot-methods), 41
plot, awsssegment-method (plot-methods), 41
plot, ICISmooth-method (plot-methods), 41
plot, kerIms-method (plot-methods), 41
plot-methods, 41
print, 42
print, ANY-method (print-methods), 41
print, aws-method (print-methods), 41
print, awsssegment-method
(print-methods), 41
print, ICISmooth-method (print-methods), 41
print, kerIms-method (print-methods), 41
print-methods, 41
qmeasures, 42
risk, ANY-method (risk-methods), 43
risk, array-method (risk-methods), 43
risk, aws-method (risk-methods), 43
risk, awsssegment-method (risk-methods), 43
risk, ICISmooth-method (risk-methods), 43
risk, kerIms-method (risk-methods), 43
risk, numeric-method (risk-methods), 43
risk-methods, 43
show, ANY-method (show-methods), 44
show, aws-method (show-methods), 44
show, awsssegment-method (show-methods), 44
show, ICISmooth-method (show-methods), 44
show, kerIms-method (show-methods), 44
show-methods, 44
summary, 45
summary, ANY-method (summary-methods), 45
summary, aws-method (summary-methods), 45
summary, awsssegment-method
(summary-methods), 45
summary, ICISmooth-method
(summary-methods), 45
summary, kerIms-method
(summary-methods), 45
summary-methods, 45
TGV_denoising (TV_denoising), 46
TGV_denoising_colour (TV_denoising), 46
TV_denoising, 46
TV_denoising_colour (TV_denoising), 46
vaw, 8, 47, 50–52
vawscov, 52
vawscov (vaw), 47
vpaw, 40, 49, 50
vpawscov, 49
vpawscov (vpaw), 50