Package ‘aws’
January 11, 2021

Version 2.5-1
Date 2021-01-11
Title Adaptive Weights Smoothing
Author Joerg Polzehl [aut, cre],
       Felix Anker [ctb]
Maintainer Joerg Polzehl <joerg.polzehl@wias-berlin.de>
Depends R (>= 3.4.0), awsMethods (>= 1.1-1)
Imports methods, 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,
    the Intersecting Confidence Intervals (ICI), variational approaches and a non-local means filter.
    The package is described in detail in Polzehl J, Papafitsoros K, Tabelow K (2020).
    <doi:10.18637/jss.v095.i06>.
    Usage of the package in neuroimaging is illustrated in Polzehl and Tabelow (2019),
License GPL (>= 2)
Copyright This package is Copyright (C) 2005-2020 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 2021-01-11 22:40:12 UTC

R topics documented:
aws-package ................................................................. 2
auxiliary ................................................................. 5
Details

The DESCRIPTION file:

Package: aws
Version: 2.5-1
Date: 2021-01-11
Title: Adaptive Weights Smoothing
Author: Joerg Polzehl [aut, cre], Felix Anker [ctb]
Maintainer: Joerg Polzehl <joerg.polzehl@wias-berlin.de>
Depends: R (>= 3.4.0), awsMethods (>= 1.1-1)
Imports: methods, gsl
Description: We provide a collection of R-functions implementing adaptive smoothing procedures in 1D, 2D and 3D. This package is a companion to the book by Tabelow (2019), Magnetic Resonance Brain Imaging, Appendix A, Springer, Use R! Series. [doi:10.1007/978-3-030-29184-6_6].
License: GPL (>=2)
Copyright: This package is Copyright (C) 2005-2020 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.
awsLocalSigma 3D variance estimation
awsdata Extract information from an object of class aws
awssegment-class Class "awssegment"
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'
gethani Auxiliary functions (for internal use)
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'
smooth3D  Auxiliary 3D smoothing routines
smse3ms  Adaptive smoothing in orientation space SE(3)
summary-methods  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
Description

Function `gethani` determines a bandwidth that leads to, for the specified kernel, a variance reduction for a non-adaptive kernel estimate by a factor of `value`. `getvofh` calculates the sum of location weights for a given bandwidth vector and kernel. `sofmchi` precomputes the variance of a non-central chi distribution with $2L$ degrees of freedom as a function of the noncentrality parameter for an interval $c(0, to)$. Functions `residualVariance` and `residualSpatialCorr` are used in package `fmri` to calculate variances and spatial correlations from residual objects.

Usage

```
gethani(x, y, lkern, value, wght, eps = 0.01)
getvofh(bw, lkern, wght)
sofmchi(L, to = 50, delta = 0.01)
residualVariance(residuals, mask, resscale = 1, compact = FALSE)
residualSpatialCorr(residuals, mask, lags = c(5, 5, 3), compact = FALSE)
```

Arguments

- `x`: lower bound of search interval
- `y`: upper bound of search interval
- `lkern`: code for location kernel
- `value`: target sum of location weights
- `wght`: relative size of voxel dimensions $c(0,0)$ for 1D and $c(w1,0)$ for 2D problems.
- `eps`: attempted precision for bandwidth search
- `bw`: vector of bandwidths, length equal to 1,2 or 3 depending on the dimensionality of the problem.
- `L`: number of effective coils, $2L$ is the degree of freedom of the non-central chi distribution.
- `to`: upper interval bound.
- `delta`: discretization width.
- `residuals`: array of residuals, if `compact` only containing voxel with mask, otherwise for complete data cubes.
- `mask`: mask of active voxel (e.g. brain masks)
- `resscale`: scale for residuals (residuals may be scaled for optimal integer*2 storage)
- `compact`: logical, determines if only information for voxel within mask or full for full data cubes is given.
- `lags`: positive integer vector of length 3, maximum lags for spatial correlations for each coordinate direction to be computed
Details

These are auxiliary functions not to be used by the user. They are only exported to be available for internal use in packages fmri, dti, qMRI and adimpro.

Value

gethani returns a vector of bandwidths, getvofh returns the variance reduction that would be obtained with a kernel estimate employing the specified kernel and bandwidth, sofmcpi returns a list with, e.g., components ncp and s2 containing vectors of noncentrality parameter values and corresponding variances, respectively, for the specified noncentral Chi distribution, residualVariance returns a vector (compact==TRUE) or array(compact==FALSE) of voxelwise residual variances, residualSpatialCorr returns an array of dimension lags containing spatial correlations.

Note

These functions are for internal use only. They are only exported to be available in other packages.

Author(s)

Joerg Polzehl <polzehl@wias-berlin.de>

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

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

Arguments

y array y containing the observe response (image intensity) 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 1D, 2D, 3D images, respectively. In case of lkern="Gaussian" the bandwidth is assumed to be given in full width half maximum (FWHM) units, i.e., 0.42466 times gridsize.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aws</td>
<td>logical: if TRUE structural adaptation (AWS) is used.</td>
</tr>
<tr>
<td>mask</td>
<td>optional logical mask, same dimensionality as y</td>
</tr>
<tr>
<td>memory</td>
<td>logical: if TRUE stagewise aggregation is used as an additional adaptation scheme.</td>
</tr>
<tr>
<td>family</td>
<td>family specifies the probability distribution. Default is family=&quot;Gaussian&quot;, also implemented are &quot;Bernoulli&quot;, &quot;Poisson&quot;, &quot;Exponential&quot;, &quot;Volatility&quot;, &quot;Variance&quot; and &quot;NCchi&quot;. family=&quot;Volatility&quot; specifies a Gaussian distribution with expectation 0 and unknown variance. family=&quot;Volatility&quot; specifies that p*y/theta is distributed as χ² with p=shape degrees of freedom. family=&quot;NCchi&quot; uses a noncentral Chi distribution with p=shape degrees of freedom and noncentrality parameter theta</td>
</tr>
<tr>
<td>lkern</td>
<td>character: location kernel, either &quot;Triangle&quot;, &quot;Plateau&quot;, &quot;Quadratic&quot;, &quot;Cubic&quot; or &quot;Gaussian&quot;. The default &quot;Triangle&quot; is equivalent to using an Epanechnikov kernel, &quot;Quadratic&quot; and &quot;Cubic&quot; refer to a Bi-weight and Tri-weight kernel, see Fan and Gijbels (1996). &quot;Gaussian&quot; is a truncated (compact support) Gaussian kernel. This is included for comparisons only and should be avoided due to its large computational costs.</td>
</tr>
<tr>
<td>aggkern</td>
<td>character: kernel used in stagewise aggregation, either &quot;Triangle&quot; or &quot;Uniform&quot;</td>
</tr>
<tr>
<td>sigma2</td>
<td>sigma2 allows to specify the variance in case of family=&quot;Gaussian&quot;. Not used if family!=&quot;Gaussian&quot;. 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.</td>
</tr>
<tr>
<td>shape</td>
<td>Allows to specify an additional shape parameter for certain family models. Currently only used for family=&quot;Variance&quot;, that is χ-Square distributed observations with shape degrees of freedom.</td>
</tr>
<tr>
<td>scorr</td>
<td>The vector scorr allows to specify a first order correlations of the noise for each coordinate direction, defaults to 0 (no correlation).</td>
</tr>
<tr>
<td>spmin</td>
<td>Determines the form (size of the plateau) in the adaptation kernel. Not to be changed by the user.</td>
</tr>
<tr>
<td>ladjust</td>
<td>factor to increase the default value of lambda</td>
</tr>
<tr>
<td>wghts</td>
<td>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.</td>
</tr>
<tr>
<td>u</td>
<td>a &quot;true&quot; 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</td>
</tr>
<tr>
<td>graph</td>
<td>If graph=TRUE intermediate results are illustrated after each iteration step. Defaults to graph=FALSE.</td>
</tr>
<tr>
<td>demo</td>
<td>If demo=TRUE the function pauses after each iteration. Defaults to demo=FALSE.</td>
</tr>
<tr>
<td>testprop</td>
<td>If set this provides diagnostics for testing the propagation condition. The values of y should correspond to the specified family and a global model.</td>
</tr>
<tr>
<td>maxni</td>
<td>If TRUE use max_{i&lt;=k} (N_i^{(l)}) instead of (N_i^{(k)}) in the definition of the statistical penalty.</td>
</tr>
</tbody>
</table>
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 \( \sigma^2 \). \( \text{aws==FALSE} \) provides the stagewise aggregation procedure from Belomestny and Spokoiny (2004). \( \text{memory==FALSE} \) 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 \( \text{family} \). Values set internally are chosen to fulfil 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 \( E|\hat{\theta}^k - \theta| \leq (1 + \alpha)E|\bar{\theta}^k - \theta| \) where \( \hat{\theta}^k \) is the aws-estimate in step \( k \) and \( \bar{\theta}^k \) 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 \( \text{ladjust}>1 \) lead to an less effective adaptation while \( \text{ladjust}<1 \) may lead to random segmentation of, with respect to a constant model, homogeneous regions.

The numerical complexity of the procedure is mainly determined by \( h_{\max} \). 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 \( \text{l kern} \). 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 \( \text{aws} \) with slots

- \( y = \text{"numeric"} \)
- \( \text{dy = "numeric"} \) \( \text{dim(y)} \)
- \( x = \text{"numeric"} \) \( \text{numeric(0)} \)
- \( \text{ni = "integer"} \) \( \text{integer(0)} \)
- \( \text{mask = "logical"} \) \( \text{logical(0)} \)
- \( \text{theta = "numeric"} \) Estimates of regression function, \( \text{length: length(y)} \)
- \( \text{mae = "numeric"} \) Mean absolute error for each iteration step if \( u \) was specified, \( \text{numeric(0)} \) 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.
- \( \text{xmin = "numeric"} \) \( \text{numeric(0)} \)
- \( \text{xmax = "numeric"} \) \( \text{numeric(0)} \)
- \( \text{wghts = "numeric"} \) \( \text{numeric(0)}, \text{ratio of distances wghts[-1]/wghts[1]} \)
- \( \text{degree = "integer"} \) \( 0 \)
hmax = "numeric"
  effective hmax
sigma2 = "numeric"
  provided or estimated error variance
scorr = "numeric"
family = "character"
shape = "numeric"
likern = "integer"
  integer code for likern, 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"
  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.irreg`, `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)
```

Description

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

Objects from the Class

Objects are created by calls to functions `aws`, `lpaws`, `aws.irreg` 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`
- `nvec`: Object of class "integer" leading dimension of `y` in vector valued data.
- `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[,...,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, corresponds to ratios of distances in coordinate directions 2 and 3 to and distance in coordinate direction 1.
degree: Object of class "integer" degree of local polynomials used in function lpaws
hmax: Object of class "numeric" maximal bandwidth
sigma2: Object of class "numeric" estimated error variance
corr: 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>

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 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**

```r
aws.gaussian(y, hmax = NULL, hpre = NULL, aws = TRUE, memory = FALSE,
  varmodel = "Constant", lkern = "Triangle",
  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" refering to a polynomial model of degree 0 to 2.

- **lkern**
  - character: location kernel, either "Triangle", "Plateau", "Quadratic", "Cubic" or "Gaussian". The default "Triangle" is equivalent to using an Epanechnikov kernel, "Quadratic" and "Cubic" refer to a Bi-weight and Tri-weight kernel, see Fan and Gijbels (1996). "Gaussian" is a truncated (compact support) Gaussian kernel. This is included for comparisons only and should be avoided due to its large computational costs.
implemented 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 chosen to fulfil a propagation condition, i.e. in case of a constant (global) parameter value and large $h_{\text{max}}$ 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 $\text{ladjust}>1$ lead to less effective adaptation while $\text{ladjust}<<1$ may lead to random segmentation of, with respect to a constant model, homogeneous regions.

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

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), ratio of distances wghts[-1]/wghts[1]
deregree = "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
vcoef = "numeric"  estimated parameters of the variance model
call = "function"  the arguments of the call to aws.gaussian
**aws.irreg**

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`, `link(awsdata)`, `aws.irreg`

Examples

```r
require(aws)
aws.irreg
```

Description

The function implements the propagation separation approach to nonparametric smoothing (formerly introduced as Adaptive weights smoothing) for varying coefficient Gaussian models on a 1D or 2D irregular design. The function allows for a parametric (polynomial) mean-variance dependence.

Usage

```r
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, 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.
aws.irreg

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
Author(s)

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

References


See Also

See also lpaws, link{awsdata}, lpaws
aws.segment

`aws.segment(y, level, delta = 0, hmax = NULL, hpre = NULL, mask = NULL,
varmodel = "Constant", lkern = "Triangle", scorr = 0, ladjust = 1,
wghts = 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 extent 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.
- **mask** : optional logical mask, same dimensionality as y
- **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 default "Triangle" is equivalent to using an Epanechnikov kernel, "Quadratic" and "Cubic" refer to a Bi-weight and Tri-weight kernel, see Fan and Gijbels (1996). "Gaussian" is a truncated (compact support) Gaussian kernel. This is included for comparisons only and should be avoided due to its large computational costs.
- **scorr** : 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: 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 H1 value >= level -delta and H2 value <= level + delta. Pixel where the first hypothesis is rejected are classified as -1 (segment 1) while rejection of H2 results in classification 1 (segment 3). Pixel where neither H1 or H2 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

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
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)
aws.segment

\begin{verbatim}
xmax = "numeric"
   numeric(0)
wghts = "numeric"
   numeric(0), ratio of distances wghts[-1]/wghts[1]
dereg = "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"
   vmodel
vcoef = "numeric"
   estimated parameters of the variance model
call = "function"
   the arguments of the call to aws.gaussian
\end{verbatim}

\textbf{Note}

This function is still experimental and may be changes considerably in future.

\textbf{Author(s)}

Joerg Polzehl, \texttt{<polzehl@wias-berlin.de>}, \url{http://www.wias-berlin.de/people/polzehl/}
References


See Also

aws, aws.gaussian

Examples

require(aws)

```r
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)
```

---

awsLocalSigma

3D variance estimation

Description

Usage

awsLocalSigma(y, steps, mask, ncoils, vext = c(1, 1), lambda = 5, 
    minni = 2, hsig = 5, sigma = NULL, family = c("NCchi", "Gauss"),
    verbose = FALSE, trace = FALSE, u = NULL)

awslinsd(y, hmax = NULL, hpre = NULL, h0 = NULL, mask = NULL,
    ladjust = 1, wghts = NULL, varprop = 0.1, A0, A1)

AFLocalSigma(y, ncoils, level = NULL, mask = NULL, h = 2, hadj = 1,
    vext = c(1, 1))

estGlobalSigma(y, mask = NULL, ncoils = 1, steps = 16, vext = c(1, 1),
    lambda = 20, hinit = 2, hadj = 1, q = 0.25, level = NULL,
    sequence = FALSE, method = c("awsVar", "awsMAD", "AFmodevn",
        "AFmode1chi", "AFbkm2chi", "AFbkm1chi"))

estimateSigmaCompl(magnitude, phase, mask, kstar = 20, kmin = 8, hsig = 5,
    lambda = 12, verbose = TRUE)

Arguments

- **y**: 3D array of image intensities.
- **steps**: number of steps in adaptive weights smoothing, used to reveal the underlying mean structure.
- **mask**: restrict computations to voxel in mask, if is.null(mask) all voxel are used. In function estGlobalSigma mask should refer to background for method %in% c(“modem1chi”, “bkm2chi”, “bkm1chi”) and to voxel within the head for method==”modevn”.
- **ncoils**: effective number of coils, or equivalently number of effective degrees of freedom of non-central chi distribution divided by 2.
- **vext**: voxel extensions or relative voxel extensions
- **lambda**: scale parameter in adaptive weights smoothing
- **minni**: minimal bandwidth for calculating local variance estimates
- **hsig**: bandwidth for median filter
- **sigma**: optional initial global variance estimate
- **family**: type of distribution, either noncentral Chi ("NCchi") or Gaussian ("Gauss")
- **verbose**: if verbose==TRUE density plots and quantiles of local estimates of sigma are provided.
- **trace**: if trace==TRUE intermediate results for each step are returned in component terms for all voxel in mask.
- **u**: if verbose==TRUE an array of noncentrality paramters for comparisons. Internal use for tests only
- **hmax**: maximal bandwidth
- **hpre**: minimal bandwidth
- **h0**: bandwidth vector characterizing to spatial correlation as correlation induced by convolution with a Gaussian kernel
- **ladjust**: correction factor for lambda
- **wghts**: relative voxel extensions
varprop defines a lower bound for the estimated variance as varprop*mean(sigma2hat)
A0 select voxel with A0 < theta < A1 to estimate parameters of the variance model
A1 select voxel with A0 < theta < A1 to estimate parameters of the variance model
level threshold for mask definition
h bandwidth for local variance estimates.
hinit minimal bandwidth for local variance estimates with method="awsxxx".
hadj bandwidth for mode estimation
q Quantile for interquantile estimate of standard deviation
sequence logical, return sequence of estimated variances for iterative methods.
method determines variance estimation method
magnitude magnitude of complex 3D image
phase phase of complex 3D image
kstar number of steps in adaptive weights smoothing, used to reveal the underlying mean structure.
kmin iteration to start adaptation

Value
all functions return lists with variance estimates in component sigma

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

References
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" containes 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", "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 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
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

showClass("awssegment")
awstestprop

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)
pawstestprop(dy, hmax, theta = 1, family = "Gaussian", lkern = "Triangle",
aws = TRUE, patchsize=1, shape = 2,
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 tested 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.
patchsize patchsize in case of paws.
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.
awsweights

maxz  Maximum of z-scale in plots.
diffz Gridlength in z
maxni If TRUE use $max_{l \leq k} (N_i^{(l)})$ 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

Author(s)

Joerg Polzehl <polzehl@wias-berlin.de>

References

Becker (2013)

See Also

- *aws*

---

**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`

---

**binning**

_Binning in 1D, 2D or 3D_

**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, d \%in\% 1\%3.
- **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", "ICIsmooth" 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")
```
ICIcombined

Arguments

x          object
what       Statistics to extract, defaults to what="y" corresponding to the original data (response variable). Alternatives are what="what" for the smoothed response, what="what" 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).

Description

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

Usage

ICIcombined(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 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.

combine Either "weighted" or "minvar". Determines how to combine sectorial results a weighted (with inverse variance) mean or to choose 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", "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 directios.
- **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
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","Triweight"). Defaults to kern="Gaussian".
m: Object of class "integer" vector of length length(dy) determining the order of derivatives specified for the coordinate directios.
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
what: 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’.

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, 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","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.
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,ICIcombined
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’.
Author(s)

Jörg Polzehl <polzehl@wias-berlin.de>

See Also

kernsm, ICIsMOOTH, ICIcombined, 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". The default "Triangle" is equivalent to using an Epanechnikov kernel, "Quadratic" and "Cubic" refer to a Bi-weight and Tri-weight kernel, see Fan and Gijbels (1996). "Gaussian" is a truncated (compact support) Gaussian kernel. This is included for comparisons only and should be avoided due to its large computational costs.
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="Xlib") 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"
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), ratio of distances wghts[-1]/wghts[1]
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"
    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


See Also

link{awsdata}, aws, aws.irreg
nlmeans

Examples

library(aws)
# 1D local polynomial smoothing
## Not run: demo(lpaws_ex1)
# 2D local polynomial smoothing
## Not run: demo(lpaws_ex2)

nlmeans  

NLMeans filter in 1D/2D/3D

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 of observed response (image intensity) data.

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 \((2*\text{patchhw}+1)^d\) for d-dimensional array).

searchhw  
Half width of search area (size of search area is \((2*\text{searchhw}+1)^d\) for d-dimensional array).

pd  
If \(pd < (2*\text{patchhw}+1)^d\) 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/denoisingadv_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


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, mask=NULL, onestep = FALSE, aws = TRUE, family = "Gaussian", lkern = "Triangle", aggkern = "Uniform", sigma2 = NULL, shape = NULL, scorr = 0, spmin = 0.25, ladjust = 1, wgths = NULL, u = NULL, graph = FALSE, demo = FALSE, patchsize = 1)

Arguments

y array y containing the observe response (image intensity) data. 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 hmax=250,12,5 for 1D, 2D, 3D images, respectively. In case of lkern="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 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 \( p \) degrees of freedom. family="NCchi" uses a noncentral Chi distribution with \( p \) degrees of freedom and noncentrality parameter \( \theta \).

lkern

character: location kernel, either "Triangle", "Plateau", "Quadratic", "Cubic" or "Gaussian". The default "Triangle" is equivalent to using an Epanechnikov kernel, "Quadratic" and "Cubic" refer to a Bi-weight and Tri-weight kernel, see Fan and Gijbels (1996). "Gaussian" is a truncated (compact support) Gaussian kernel. This is included for comparisons only and should be avoided due to its large computational costs.

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 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 \( p \) 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.

patchsize

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

Details

see 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 aws with slots

\( y \) = "numeric" \ y
\( dy \) = "numeric" \ dim(y)
\( x \) = "numeric" \ numeric(0)
\( ni \) = "integer" \ integer(0)
\( mask \) = "logical"
\( \text{logical}(0) \)
\( \theta \) = "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
\( \text{var} \) = "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"
\( \text{numeric}(0) \)
\( \text{xmax} \) = "numeric"
\( \text{numeric}(0) \)
\( \text{wghts} \) = "numeric"
\( \text{numeric}(0), \text{ratio of distances} \ wghts[-1]/wghts[1] \)
\( \text{degree} \) = "integer"
\( 0 \)
\( \text{hmax} \) = "numeric" \ effective hmax
\( \text{sigma2} \) = "numeric" \ provided or estimated error variance
\( \text{scorr} \) = "numeric" \ scorr
\( \text{family} \) = "character" \ family
\( \text{shape} \) = "numeric" \ shape
\( \text{lkern} \) = "integer" \ integer code for \( \text{lkern}, 1=\text{"Plateau"}, 2=\text{"Triangle"}, 3=\text{"Quadratic"}, 4=\text{"Cubic"}, 5=\text{"Gaussian"} \)
\( \text{lambda} \) = "numeric" \ effective value of \( \text{lambda} \)
\( \text{ladjust} \) = "numeric" \ effective value of \( \text{ladjust} \)
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, lpaws, vpaws, link{awsdata}

Examples

```r
## Not run:
setCores(2)
y <- array(rnorm(64^3),c(64,64,64))
yhat <- paws(y,hmax=6)

## 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)

Jörg 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"

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

**Author(s)**

Joerg Polzehl <polzehl@wias-berlin.de>

**See Also**

aws, awsegment, ICISmooth kernsm

---

**qmeasures** *Quality assessment for image reconstructions.*

**Description**

Computes selected criteria for quality assessments of

**Usage**

```r
def 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.

**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/](http://www.wias-berlin.de/people/polzehl/)
Risks Methods

Description

Methods function `risk` in package `aws`. For an array `u` the following statistics are computed:

- Root Mean Squared Error `RMSE <- sqrt(mean((y-u)^2))`
- Signal to Noise Ratio `SNR <- 10*log(mean(u^2)/MSE,10)`
- Peak Signal to Noise Ratio `PSNR <- 10*log(max(u^2)/MSE,10)`
- Mean Absolute Error `MAE <- mean(abs(y-u))`
- Maximal Absolute Error `MaxAE <- max(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
- `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>
show-methods

References


show-methods Methods for Function ‘show’ in Package ‘aws’

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 extend "aws".

signature(object = "awssegment") Provide information on data dimensions, data source and existing slot-names for objects of class "dti" and classes that extend "awssegment".

signature(object = "ICIsmooth") Provide information on data dimensions, data source and existing slot-names for objects of class "dti" and classes that extend "ICIsmooth".

signature(object = "kernsm") Provide information on data dimensions, data source and existing slot-names for objects of class "dti" and classes that extend "kernsm".

Author(s)

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

See Also

aws, awsssegment, ICIsmooth kernsm
**smooth3D**

**Auxiliary 3D smoothing routines**

**Description**

smooth3D and medianFilter3D are auxiliary functions for non-adaptive smoothing of 3D image data using kernel or median smoothing. Both function restrict to sub-areas determined by a mask. The functions are used in packages dti and qMRI.

Functions aws3Dmask and aws3Dmaskfull perform adaptive weights smoothing on statistical parametric maps in fMRI. Variability of results is determined from smoothed (using the same weighting schemes) residuals in order to correctly account for spatial correlation. These functions are intended to be used internally in package fmri. They have been moved here because they share significant parts of the openMP parallelized Fortran code underlying function aws.

**Usage**

```r
smooth3D(y, h, mask, lkern = "Gaussian", weighted = FALSE, sigma2 = NULL, wghts = NULL)
medianFilter3D(y, h = 10, mask = NULL)
aws3Dmask(y, mask, lambda, hmax, res = NULL, sigma2 = NULL, lkern = "Gaussian", skern = "Plateau", weighted = TRUE, u = NULL, wghts = NULL, h0 = c(0, 0, 0), testprop = FALSE)
aws3Dmaskfull(y, mask, lambda, hmax, res = NULL, sigma2 = NULL, lkern = "Gaussian", skern = "Plateau", weighted = TRUE, u = NULL, wghts = NULL, testprop = FALSE)
```

**Arguments**

- **y**: 3D array of data in case of functions smooth3D and medianFilter3D. For aws3Dmask* with !is.null(mask) a vector of length sum(mask) containing only data values within the specified mask.
- **lkern**: character: location kernel, either "Triangle", "Plateau", "Quadratic", "Cubic" or "Gaussian". The default "Triangle" is equivalent to using an Epanechnikov kernel, "Quadratic" and "Cubic" refer to a Bi-weight and Tri-weight kernel, see Fan and Gijbels (1996). "Gaussian" is a truncated (compact support) Gaussian kernel. This is included for comparisons only and should be avoided due to its large computational costs.
- **weighted**: logical: use inverse variances as weights.
- **sigma2**: sigma2 allows to specify the variance of data entries.
- **mask**: optional logical mask, same dimensionality as y
- **h**: bandwidth to use. In case of lkern="Gaussian" this is in FWHM (full width half maximum) units. Value refers to first voxel dimension.
- **wghts**: voxel dimensions. Defaults to c(1,1,1)
- **lambda**: kritical scale parameter in hypothesis testing (adaptive weights smoothing)
smse3ms

hmax  maximum bandwidth for adaptive weights smoothing
res   array of residuals with dimension c(nres, sum(mask)).
skern specifies the kernel for the statistical penalty. Defaults to "Plateau", the
alternatives are "Triangle" and "Exp". "Plateau" specifies a kernel that is equal to
1 in the interval (0,.3), decays linearly in (.3,1) and is 0 for arguments larger than
1. lkern="Plateau" and lkern="Triangle" allow for much faster computa-
tion (saves up to 50% CPU-time). lkern="Plateau" produces a less random
weighting scheme.
u    For test purposes in simulations: noisless 3D data.
h0   Vector of 3 bandwidths corresponding to a Gaussian kernel that would produce
a comparable spatial correlation by convoluting iid data.
testprop logical: test the validity of a propagation condition for the specified value of
lambda.

Value

Functions smooth3D and medianFilter3D return a 3D array. Functions awsmask* return a list with
smoothed values of y in component theta and smoothed residuals in component res.

Note

Functions awsmask* are used internally in package fmri. They refer to the situation, typical for
fMRI, where the data are spatially correlated and this correlation can be accessed using residuals
with respect to a model.

Author(s)

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

---

smse3ms  Adaptive smoothing in orientation space SE(3)

Description

The functions perform adaptive weights smoothing for data in orientation space SE(3), e.g. diffusion
weighted MR data, with spatial coordinates given by voxel location within a mask and spherical
information given by gradient direction. Observations can belong to different shells characterized
by b-value bv. The data provided should only refer to voxel within mask.

Usage

smse3ms(sb, s0, bv, grad, kstar, lambda, kappa0, mask, sigma,
    ns0 = 1, ws0 = 1, vext = NULL, ncoils = 1, verbose = FALSE, usemaxni = TRUE)
smse3(sb, s0, bv, grad, mask, sigma, kstar, lambda, kappa0,
    ns0 = 1, vext = NULL, vred = 4, ncoils = 1, model = 0, dist = 1,
    verbose = FALSE)
Arguments

**sb**  
2D array of diffusion weighted data, first dimension refers to index of voxel within the mask, second dimension to the number diffusion weighted images.

**s0**  
vector of length \(\sum(mask)\) containing values within mask of an average non-diffusion weighted image.

**bv**  
vector of b-values.

**grad**  
matrix of gradient directions with \(\text{dim}(\text{grad})[1]=3\).

**kstar**  
number of steps in adaptive weights smoothing.

**lambda**  
Scale parameter in adaptation

**kappa0**  
determines amount of smoothing on the sphere. Larger values correspond to stronger smoothing on the sphere. If \(\text{kappa0}=\text{NULL}\) a value is that corresponds to a variance reduction with factor \(vred\) on the sphere.

**mask**  
3D image defining a mask (logical)

**sigma**  
Error standard deviation. Assumed to be known and homogeneous in the current implementation. A reasonable estimate may be defined as the modal value of standard deviations obtained using method \(\text{getsdofsb}\).

**ns0**  
Actual number of non-diffusion weighted images used to obtain \(s0\) by averaging.

**ws0**  
Weight for non-diffusion weighted images in statistical penalty.

**vext**  
Voxel extensions.

**ncoils**  
Effective number of receiver coils (in case of e.g. GRAPPA reconstructions), should be 1 in case of SENSE reconstructions. \(2\times ncoils\) is the number of degrees of freedom of the intensity distribution used.

**verbose**  
If \(\text{verbose}=\text{TRUE}\) additional reports are given.

**usemaxni**  
If \"usemaxni==\text{TRUE}\" a strikter penalization is used.

**vred**  
Used if \(\text{kappa0}=\text{NULL}\) to specify the variance reduction on the sphere when suggesting a value of \(\text{kappa0}\).

**model**  
Determines which quantities are smoothed. Possible values are \"Chi\" for observed values (assumed to be distributed as noncentral Chi with \(2\times ncoils\) degrees of freedom), \"Chi2\" for squares of observed values (assumed to be distributed as noncentral Chi-squared with \(2\times ncoils\) degrees of freedom). \"Gapprox\" and \"Gapprox2\" use a Gaussian approximation for the noncentral Chi distribution to smooth observed and squared values, respectively.

**dist**  
Distance in SE3. Reasonable values are 1 (default, see Becker et.al. 2012), 2 (a slight modification of 1: with \(k6^2\) instead of \(\text{abs}(k6)\)) and 3 (using a 'naive' distance on the sphere)

Value

The functions return lists with main results in components \(\text{th}\) and \(\text{th0}\) containing the smoothed data.
Note

These functions are intended to be used internally in package dti only.

Author(s)

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

References


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

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 matrix of noisy 2D image data. In case of TV_denoising_colour and TGV_denoising_colour and array with third dimension refering 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/

References

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, whgts = NULL, u = NULL, maxni = FALSE)
vawscov(y, kstar = 16, invcov = NULL, mask = NULL, scorr = 0, spmin = 0.25, ladjust = 1, whgts = 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  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<k} (N_l(l)) instead of (N_l(k)) in the definition of the statistical
penalty.

Details

see aws. Expets 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: 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.
  xmin = "numeric"  numeric(0)
  xmax = "numeric"  numeric(0)
  wghts = "numeric"  numeric(0), ratio of distances wghts[-1]/wghts[1]
  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`, `link{awsdata}`

Examples

```r
## Not run:
setCores(2)
y <- array(rnorm(4*64^3),c(4,64,64,64))
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, invcov = NULL, mask = NULL, scorr = 0, spmin = 0.25,
      ladjust = 1, wghts = NULL, u = NULL, 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)

vpawscov2(y, kstar = 16, invcov = NULL, mask = NULL, scorr = 0, spmin = 0.25,
          lambda = NULL, ladjust = 1, wghts = NULL, patchsize = 1,
          data = NULL, verbose = TRUE)
```

**Arguments**

- `y` y can be a full array of vector valued data, or, if `mask` is provided, be a matrix with columns corresponding to points/pixel/voxel within the mask. In the first case `dim(y)` determines the dimensionality and extend of the grid design, in the second case this information is obtained from the dimensions of `mask`. The 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 (or matrix) of voxelwise inverse covariance matrices, first index corresponds to upper diagonal inverse covariance matrix.
- `mask` logical mask. All computations are restricted to design points within the mask.
vpaws

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

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

maxni  require growing sum of weights

lambda  explicit value of lambda

data  optional vector-valued images to be smoothed using the weighting scheme of the last step

verbose  logical: provide information on progress.

Details

see vaws. Parameter y 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.

Function vpawscov2 is intended for internal use in package qMRI only.

Value

function vpaws returns returns an object of class aws with slots

\[
\begin{align*}
y & = \text{"numeric"} & \text{y} \\
dy & = \text{"numeric"} & \dim(y) \\
x & = \text{"numeric"} & \text{numeric(0)} \\
ni & = \text{"integer"} & \text{integer(0)} \\
mask & = \text{"logical"} & \text{logical(0)} \\
theta & = \text{"numeric"} & \text{Estimates of regression function, length: \text{length(y)} } \\
hseq & = \text{"numeric"} & \text{sequence of bandwidths employed} \\
mae & = \text{"numeric"} & \text{Mean absolute error for each iteration step if u was specified, numeric(0) else} \\
psnr & = \text{"numeric"} & \text{Peak signal-to-noise ratio for each iteration step if u was specified, numeric(0) else} \\
var & = \text{"numeric"} & \text{approx. variance of the estimates of the regression function. Please note that this does not reflect variability due to randomness of weights. Currently also uses factor 1/ni instead of the correct sum(wij^2)/ni^2}
\end{align*}
\]
xmin = "numeric"
    numeric(0)
xmax = "numeric"
    numeric(0)
wghts = "numeric"
    numeric(0), ratio of distances wghts[-1]/wghts[1]
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

If y contained only information (condensed data) for positions within a mask, then the returned object only contains results for these positions.

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

* classes
  aws-class, 10
  awssegment-class, 25
  ICIsmooth-class, 34
  kernsm-class, 37
* hplot
  plot-methods, 46
* manip
  awndata, 21
  binning, 29
  extract-methods, 30
* methods
  extract-methods, 30
  plot-methods, 46
  print-methods, 46
  risk-methods, 48
  show-methods, 49
  summary-methods, 53
* misc
  auxiliary, 5
  awsweight, 28
  smooth3D, 50
* nonparametric
  aws, 6
  aws.gaussian, 12
  aws.irreg, 15
  aws.segment, 18
  awndata, 21
  awstestprop, 26
  ICICombined, 31
  ICIsmooth, 33
  kernsm, 35
  lpaws, 38
  paws, 42
  vaws, 55
  vpaws, 58
* package
  aws-package, 2
* regression
  aws, 6
  aws.gaussian, 12
  aws.irreg, 15
  aws.segment, 18
  awndata, 21
  lpaws, 38
  paws, 42
  vaws, 55
  vpaws, 58
* smooth
  awndata, 21
  lpaws, 38
  paws, 42
  risk-methods, 48
  smooth3D, 50
  smse3ms, 51
  TV_denoising, 54
  vaws, 55
  vpaws, 58
* utilities
  show-methods, 49
* utilities
  awslasLocalSigma, 22
  print-methods, 46
  qmeasures, 47
  summary-methods, 53

AFLocalSigma(awsLocalSigma), 22
auxiliary, 5
aws, 6, 12, 15, 21, 22, 28, 29, 35, 40, 42, 43, 45–47, 49, 54–56, 58
aws-class, 10
aws-package, 2
aws.gaussian, 10, 12, 12, 21
aws.irreg, 10, 12, 12, 21, 30, 40
aws.segment, 18, 26
aws3Dmask (smooth3D), 50
aws3Dmaskfull (smooth3D), 50
aws.data, 21
awslinsd (awsLocalSigma), 22
awsLocalSigma, 22
awssegment, 46, 47, 49, 54
awssegment-class, 25
awstestprop, 26
aws.weights, 28
binning, 29
estGlobalSigma (awsLocalSigma), 22
estimateSigmaCompl (awsLocalSigma), 22
extract, ANY-method (extract-methods), 30
extract, aws-method (extract-methods), 30
extract, awssegment-method (extract-methods), 30
extract, ICIsmooth-method (extract-methods), 30
extract, kernsm-method (extract-methods), 30
extract-methods, 30
gethani (auxiliary), 5
getvofh (auxiliary), 5
ICIcombined, 31, 34–36, 38
ICIsmooth, 32, 33, 35, 36, 46, 47, 49, 54
ICIsmooth-class, 34
kernsm, 32, 34, 35, 38, 46, 47, 49, 54
kernsm-class, 37
lpaws, 10, 12, 17, 38, 45, 61
medianFilter3D (smooth3D), 50
nlmeans, 41
paws, 10, 42, 58
pawsm (paws), 42
pawstestprop (awstestprop), 26
plot, 46
plot, ANY-method (plot-methods), 46
plot, aws-method (plot-methods), 46
plot, awssegment-method (plot-methods), 46
plot, ICIsmooth-method (plot-methods), 46
plot, kernsm-method (plot-methods), 46
plot-methods, 46
print, 46
print, ANY-method (print-methods), 46
print, aws-method (print-methods), 46
print, awssegment-method (print-methods), 46
print, ICIsmooth-method (print-methods), 46
print, kernsm-method (print-methods), 46
print-methods, 46
qmeasures, 47
residualSpatialCov (auxiliary), 5
residualVariance (auxiliary), 5
risk, ANY-method (risk-methods), 48
risk, array-method (risk-methods), 48
risk, aws-method (risk-methods), 48
risk, awssegment-method (risk-methods), 48
risk, ICIsmooth-method (risk-methods), 48
risk, kernsm-method (risk-methods), 48
risk, numeric-method (risk-methods), 48
risk-methods, 48
show, ANY-method (show-methods), 49
show, aws-method (show-methods), 49
show, awssegment-method (show-methods), 49
show, ICIsmooth-method (show-methods), 49
show, kernsm-method (show-methods), 49
show-methods, 49
smooth3D, 50
smse3 (smse3ms), 51
smse3ms, 51
sofmchi (auxiliary), 5
summary, 53
summary, ANY-method (summary-methods), 53
summary, aws-method (summary-methods), 53
summary, awssegment-method (summary-methods), 53
summary,ICIsmooth-method
    (summary-methods), 53
summary,kernsm-method
    (summary-methods), 53
summary-methods, 53

TGV_denoising (TV_denoising), 54
TGV_denoising_colour (TV_denoising), 54
TV_denoising, 54
TV_denoising_colour (TV_denoising), 54

vaws, 10, 55, 58, 59, 61
vawscov, 61
vawscov (vaws), 55
vpaws, 45, 58, 58
vpawscov (vpaws), 58
vpawscov2 (vpaws), 58