

# Package ‘aws’

January 2, 2012

**Version** 1.7-1

**Date** 2011-12-09

**Title** Adaptive Weights Smoothing

**Author** Joerg Polzehl <polzehl@wias-berlin.de>

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

**Depends** R (>= 2.5.0), methods

**Description** The package contains R-functions implementing the Propagation-Separation Approach to adaptive smoothing as described in J. Polzehl and V. Spokoiny (2006), Propagation-Separation Approach for Local Likelihood Estimation, Prob. Theory and Rel. Fields, 135(3):335-362. and J. Polzehl and V. Spokoiny (2004) Spatially adaptive regression estimation: Propagation-separation approach, WIAS-Preprint 998.

**License** GPL (>= 2)

**Copyright** This package is Copyright (C) 2005-2010 Weierstrass Institute for Applied Analysis and Stochastics.

**URL** [http://www.wias-berlin.de/projects/matheon\\_a3/](http://www.wias-berlin.de/projects/matheon_a3/)

**Repository** CRAN

**Date/Publication** 2011-12-11 17:42:19

## R topics documented:

aws-package . . . . .	2
aws . . . . .	3
aws.gaussian . . . . .	6
aws.irreg . . . . .	10
aws.segment . . . . .	12
awsdata . . . . .	15
binning . . . . .	17
lpaws . . . . .	18

**Index****21**


---

aws-package                      *Adaptive Weights Smoothing*


---

**Description**

The package contains R-functions implementing the Propagation-Separation Approach to adaptive smoothing as described in J. Polzehl and V. Spokoiny (2006) Propagation-Separation Approach for Local Likelihood Estimation, *Prob. Theory and Rel. Fields* 135(3):335-362. and J. Polzehl and V. Spokoiny (2004) Spatially adaptive regression estimation: Propagation-separation approach, WIAS-Preprint 998.

**Details**

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

**Index:**

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

**Author(s)**

Joerg Polzehl <polzehl@wias-berlin.de>

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

**References**

J. Polzehl and V. Spokoiny (2006) Propagation-Separation Approach for Local Likelihood Estimation, *Prob. Theory and Rel. Fields* **135(3)**, 335-362.

J. Polzehl and V. Spokoiny (2004) Spatially adaptive regression estimation: Propagation-separation approach, WIAS-Preprint 998.

---

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, aws=TRUE, memory=FALSE, family="Gaussian",
    lkern="Triangle", homogen=TRUE, aggkern="Uniform",
    sigma2=NULL, shape=NULL, scorr=0,
    ladjust=1, wghts=NULL, u=NULL, graph=FALSE, demo=FALSE,
    testprop=FALSE)
```

### Arguments

y	y contains the observed response data. dim(y) determines the dimensionality and extend of the grid design.
hmax	hmax specifies the maximal bandwidth. Defaults to hmax=250, 12, 5 for dd=1, 2, 3, respectively.
aws	logical: if TRUE structural adaptation (AWS) is used.
memory	logical: if TRUE stagewise aggregation is used as an additional adaptation scheme.
family	family specifies the probability distribution. Default is family="Gaussian", also implemented are "Bernoulli", "Poisson", "Exponential", "Volatility" and "Variance". family="Volatility" specifies a Gaussian distribution with expectation 0 and unknown variance. family="Volatility" specifies that $p \cdot y / \theta$ is distributed as $\chi^2$ with p=shape degrees of freedom.
lkern	character: location kernel, either "Triangle", "Plateau", "Quadratic", "Cubic" or "Gaussian"
homogen	logical: if TRUE the function tries to determine regions where weights can be fixed to 1. This may increase speed.
aggkern	character: kernel used in stagewise aggregation, either "Triangle" or "Uniform"
sigma2	sigma2 allows to specify the variance in case of family="Gaussian". Not used if family!="Gaussian". Defaults to NULL. In this case a homoskedastic variance estimate is generated. If 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 $\chi$ -Square distributed observations with shape degrees of freedom.

scorr	The vector <code>scorr</code> 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 <code>lambda</code>
wghts	<code>wghts</code> 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 <code>u=0</code>
graph	If <code>graph=TRUE</code> intermediate results are illustrated after each iteration step. Defaults to <code>graph=FALSE</code> .
demo	If <code>demo=TRUE</code> the function pauses after each iteration. Defaults to <code>demo=FALSE</code> .
testprop	If set this provides diagnostics for testing the propagation condition. The values of <code>y</code> should correspond to the specified family and a global model.

### Details

The function implements the propagation separation approach to nonparametric smoothing (formerly introduced as Adaptive weights smoothing) for varying coefficient likelihood models on a 1D, 2D or 3D grid. For "Gaussian" models, i.e. regression with additive "Gaussian" errors, a homoskedastic or heteroskedastic model is used depending on the content of `sigma2`. `aws==FALSE` provides the stagewise aggregation procedure from Belomestny and Spokoiny (2004). `memory==FALSE` provides Adaptive weights smoothing without control by stagewise aggregation.

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

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

### Value

returns an object of class `aws` with slots

```

y = "numeric"  y
dy = "numeric" dim(y)
x = "numeric"  numeric(0)
ni = "integer" integer(0)

```

```

mask = "logical"
      logical(0)
theta = "numeric"
      Estimates of regression function, length: length(y)
mae = "numeric"
      Mean absolute error for each iteration step if u was specified, numeric(0) else
var = "numeric"
      approx. variance of the estimates of the regression function. Please note that
      this does not reflect variability due to randomness of weights.
xmin = "numeric"
      numeric(0)
xmax = "numeric"
      numeric(0)
wghts = "numeric"
      numeric(0)
degree = "integer"
      0
hmax = "numeric"
      effective hmax
sigma2 = "numeric"
      provided or estimated error variance
scorr = "numeric"
      scorr
family = "character"
      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

```

**Author(s)**

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

**References**

Joerg Polzehl, Vladimir Spokoiny, Adaptive Weights Smoothing with applications to image restoration, J. R. Stat. Soc. Ser. B Stat. Methodol. 62 , (2000) , pp. 335–354

Joerg Polzehl, Vladimir Spokoiny, Propagation-separation approach for local likelihood estimation, Probab. Theory Related Fields 135 (3), (2006) , pp. 335–362.

Joerg Polzehl, Vladimir Spokoiny, in V. Chen, C.; Haerdle, W. and Unwin, A. (ed.) Handbook of Data Visualization Structural adaptive smoothing by propagation-separation methods Springer-Verlag, 2008, 471-492

**See Also**

See also [lpaws](#), [link{awsdata}](#), [aws.irreg](#), [aws.gaussian](#)

**Examples**

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

---

aws.gaussian

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

---

**Description**

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

**Usage**

```
aws.gaussian(y, hmax = NULL, hpre = NULL, aws = TRUE, memory = FALSE, varmodel = "Constant", lkern = "Tri
```

**Arguments**

y	y contains the observed response data. $\dim(y)$ determines the dimensionality and extend of the grid design.
hmax	hmax specifies the maximal bandwidth. Defaults to hmax=250, 12, 5 for dd=1, 2, 3, respectively.
hpre	Describe hpre Bandwidth used for an initial nonadaptive estimate. The first estimate of variance parameters is obtained from residuals with respect to this estimate.
aws	logical: if TRUE structural adaptation (AWS) is used.
memory	logical: if TRUE stagewise aggregation is used as an additional adaptation scheme.
varmodel	Implemented are "Constant", "Linear" and "Quadratic" referring to a polynomial model of degree 0 to 2.
lkern	character: location kernel, either "Triangle", "Plateau", "Quadratic", "Cubic" or "Gaussian"
homogen	logical: if TRUE the function tries to determine regions where weights can be fixed to 1. This may increase speed.
aggkern	character: kernel used in stagewise aggregation, either "Triangle" or "Uniform"
scorr	The vector scorr allows to specify a first order correlations of the noise for each coordinate direction, defaults to 0 (no correlation).
mask	Restrict smoothing to points where mask==TRUE. Defaults to TRUE in all voxel.
ladjust	factor to increase the default value of lambda
wghts	wghts specifies the diagonal elements of a weight matrix to adjust for different distances between grid-points in different coordinate directions, i.e. allows to define a more appropriate metric in the design space.
u	a "true" value of the regression function, may be provided to report risks at each iteration. This can be used to test the propagation condition with u=0
varprop	Small variance estimates are replaced by varprop times the mean variance.
graph	If graph=TRUE intermediate results are illustrated after each iteration step. Defaults to graph=FALSE.
demo	If demo=TRUE the function pauses after each iteration. Defaults to demo=FALSE.

**Details**

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

The essential parameter in the procedure is a critical value lambda. This parameter has an interpretation as a significance level of a test for equivalence of two local parameter estimates. Values set

internally are chosen to fulfil a propagation condition, i.e. in case of a constant (global) parameter value and large hmax the procedure provides, with a high probability, the global (parametric) estimate. More formally we require the parameter lambda to be specified such that  $\mathbf{E}|\hat{\theta}^k - \theta| \leq (1 + \alpha)\mathbf{E}|\tilde{\theta}^k - \theta|$  where  $\hat{\theta}^k$  is the aws-estimate in step k and  $\tilde{\theta}^k$  is corresponding nonadaptive estimate using the same bandwidth (lambda=Inf). The value of lambda can be adjusted by specifying the factor ladjust. Values ladjust>1 lead to an less effective adaptation while ladjust<<1 may lead to random segmentation of, with respect to a constant model, homogeneous regions.

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

### Value

returns an object of class aws with slots

```

y = "numeric"  y
dy = "numeric" dim(y)
x = "numeric"  numeric(0)
ni = "integer" integer(0)
mask = "logical"
                logical(0)
theta = "numeric"
                Estimates of regression function, length: length(y)
mae = "numeric"
                Mean absolute error for each iteration step if u was specified, numeric(0) else
var = "numeric"
                approx. variance of the estimates of the regression function. Please note that
                this does not reflect variability due to randomness of weights.
xmin = "numeric"
                numeric(0)
xmax = "numeric"
                numeric(0)
wghts = "numeric"
                numeric(0)
degree = "integer"
                0
hmax = "numeric"
                effective hmax
sigma2 = "numeric"
                provided or estimated error variance
scorr = "numeric"
                scor
family = "character"
                "Gaussian"

```

```

shape = "numeric"
          NULL
lkern = "integer"
          integer code for lkern, 1="Plateau", 2="Triangle", 3="Quadratic", 4="Cubic",
          5="Gaussian"
lambda = "numeric"
          effective value of lambda
ladjust = "numeric"
          effective value of ladjust
aws = "logical"
          aws
memory = "logical"
          memory
homogen = "logical"
          homogen
earlystop = "logical"
          FALSE
varmodel = "character"
          varmodel
vcoef = "numeric"
          estimated parameters of the variance model
call = "function"
          the arguments of the call to aws.gaussian

```

**Author(s)**

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

**References**

Joerg Polzehl, Vladimir Spokoiny, Adaptive Weights Smoothing with applications to image restoration, J. R. Stat. Soc. Ser. B Stat. Methodol. 62 , (2000) , pp. 335–354

Joerg Polzehl, Vladimir Spokoiny, Propagation-separation approach for local likelihood estimation, Probab. Theory Related Fields 135 (3), (2006) , pp. 335–362.

Joerg Polzehl, Vladimir Spokoiny, in V. Chen, C.; Haerdle, W. and Unwin, A. (ed.) Handbook of Data Visualization Structural adaptive smoothing by propagation-separation methods Springer-Verlag, 2008, 471-492

**See Also**

See also [aws](#), [link{awsdata}](#), [aws.irreg](#)

**Examples**

```
require(aws)
```

---

aws.irreg                      *local constant AWS for irregular (1D/2D) design*

---

### 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

```
aws.irreg(y, x, hmax = NULL, aws=TRUE, memory=FALSE, varmodel = "Constant", lkern = "Triangle", aggkern
```

### 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.
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  $\leq$  henv of nonempty bins.

**Value**

returns an object of class `aws` with slots

```

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

```

```

homogen = "logical"
          FALSE
earlystop = "logical"
           FALSE
varmodel = "character"
           varmodel
vcoef = "numeric"
        estimated coefficients in variance model
call = "function"
      the arguments of the call to aws

```

**Author(s)**

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

**References**

Joerg Polzehl, Vladimir Spokoiny, in V. Chen, C.; Haerdle, W. and Unwin, A. (ed.) Handbook of Data Visualization Structural adaptive smoothing by propagation-separation methods Springer-Verlag, 2008, 471-492

**See Also**

See also [lpaws](#), [link{awsdata}](#), [lpaws](#)

**Examples**

```

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

```

---

aws.segment

*Segmentation by adaptive weights for Gaussian models.*

---

**Description**

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

**Usage**

```
aws.segment(y, level, delta = 0, hmax = NULL, hpre = NULL, varmodel = "Constant", lkern = "Triangle", sco
```

**Arguments**

y	y contains the observed response data. $\dim(y)$ determines the dimensionality and extend of the grid design.
level	center of second class
delta	half width of second class
hmax	hmax specifies the maximal bandwidth. Defaults to hmax=250, 12, 5 for dd=1, 2, 3, respectively.
hpre	Describe hpre Bandwidth used for an initial nonadaptive estimate. The first estimate of variance parameters is obtained from residuals with respect to this estimate.
varmodel	Implemented are "Constant", "Linear" and "Quadratic" referring to a polynomial model of degree 0 to 2.
lkern	character: location kernel, either "Triangle", "Plateau", "Quadratic", "Cubic" or "Gaussian"
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  $\geq \text{level} - \text{delta}$  and  $H2$  value  $\leq \text{level} + \text{delta}$ . 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)
ni = "integer" integer(0)
mask = "logical"
                logical(0)
segment = "integer"
                Segmentation results, class numbers 1-3
theta = "numeric"
                Estimates of regression function, length: length(y)
mae = "numeric"
                Mean absolute error for each iteration step if u was specified, numeric(0) else
var = "numeric"
                approx. variance of the estimates of the regression function. Please note that
                this does not reflect variability due to randomness of weights.
xmin = "numeric"
                numeric(0)
xmax = "numeric"
                numeric(0)
wghts = "numeric"
                numeric(0)
degree = "integer"
                0
hmax = "numeric"
                effective hmax
sigma2 = "numeric"
                provided or estimated error variance
scorr = "numeric"
                scorr
family = "character"
                "Gaussian"
shape = "numeric"
                NULL
lkern = "integer"
                integer code for lkern, 1="Plateau", 2="Triangle", 3="Quadratic", 4="Cubic",
                5="Gaussian"
lambda = "numeric"
                effective value of lambda
ladjust = "numeric"
                effective value of ladjust
aws = "logical"
                aws

```

```
memory = "logical"
      memory
homogen = "logical"
      FALSE
earlystop = "logical"
      FALSE
varmodel = "character"
      varmodel
vcoef = "numeric"
      estimated parameters of the variance model
call = "function"
      the arguments of the call to aws.gaussian
```

### Note

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

### Author(s)

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

### References

- Duembgen, L. and Spokoiny, V. (2001). Multiscale testing of qualitative hypotheses. *Ann. Stat.* 29, 124–152.
- Polzehl, J. and Spokoiny, V. (2006). Propagation-Separation Approach for Local Likelihood Estimation. *Probability Theory and Related Fields.* 3 (135) 335 - 362.

### See Also

[aws](#), [aws.gaussian](#)

### Examples

```
require(aws)
```

---

awsdata

*Extract information from an object of class aws*

---

### Description

Extract data and estimates from an object of class aws

### Usage

```
awsdata(awsobj, what)
```

**Arguments**

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

**Details**

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

**Value**

an vector or array containing the specified information.

**Author(s)**

Joerg Polzehl <polzehl@wias-berlin.de>

**References**

Joerg Polzehl, Vladimir Spokoiny, Adaptive Weights Smoothing with applications to image restoration, *J. R. Stat. Soc. Ser. B Stat. Methodol.* 62 , (2000) , pp. 335–354

Joerg Polzehl, Vladimir Spokoiny, Propagation-separation approach for local likelihood estimation, *Probab. Theory Related Fields* 135 (3), (2006) , pp. 335–362.

Joerg Polzehl, Vladimir Spokoiny, in V. Chen, C.; Haerdle, W. and Unwin, A. (ed.) *Handbook of Data Visualization Structural adaptive smoothing by propagation-separation methods* Springer-Verlag, 2008, 471-492

**See Also**

[link{awsdata}](#), [aws](#), [aws.irreg](#)

**Examples**

```
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)
```

---

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](#)

---

 lpaws

---

*Local polynomial smoothing by AWS*


---

**Description**

The function allows for structural adaptive smoothing using a local polynomial (degree  $\leq 2$ ) 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 = FALSE)
```

**Arguments**

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

u	a "true" value of the regression function, may be provided to report risks at each iteration. This can be used to test the propagation condition with u=0
graph	logical: If TRUE intermediate results are illustrated graphically. May significantly slow down the computations in 2D. Please avoid using the default X11() on systems build with cairo, use X11(type="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"
                logical(0)
theta = "numeric"
                Estimates of regression function and derivatives, length: length(y)*(degree+1)
mae = "numeric"
                Mean absolute error for each iteration step if u was specified, numeric(0) else
var = "numeric"
                approx. variance of the estimates of the regression function. Please note that
                this does not reflect variability due to randomness of weights.
xmin = "numeric"
                numeric(0)
xmax = "numeric"
                numeric(0)
wghts = "numeric"
                numeric(0)
degree = "integer"
                degree
hmax = "numeric"
                effective hmax
sigma2 = "numeric"
                provided or estimated error variance
scorr = "numeric"
                0
family = "character"
                "Gaussian"
shape = "numeric"
                numeric(0)
lkern = "integer"
                integer code for lkern, 1="Plateau", 2="Triangle", 3="Quadratic", 4="Cubic",
                5="Gaussian"

```

```

lambda = "numeric"
           effective value of lambda
ladjust = "numeric"
           effective value of ladjust
aws = "logical"
           aws
memory = "logical"
           memory
homogen = "logical"
           homogen
earlystop = "logical"
           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**

Joerg Polzehl, Vladimir Spokoiny, in V. Chen, C.; Haerdle, W. and Unwin, A. (ed.) Handbook of Data Visualization Structural adaptive smoothing by propagation-separation methods Springer-Verlag, 2008, 471-492

**See Also**

`link{awsdata}`, `aws`, [aws.irreg](#)

**Examples**

```

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

```

# Index

\*Topic **manip**

awsdata, 15

binning, 17

\*Topic **nonparametric**

aws, 3

aws.gaussian, 6

aws.irreg, 10

aws.segment, 12

awsdata, 15

lpaws, 18

\*Topic **package**

aws-package, 2

\*Topic **regression**

aws, 3

aws.gaussian, 6

aws.irreg, 10

aws.segment, 12

awsdata, 15

lpaws, 18

\*Topic **smooth**

aws, 3

aws.gaussian, 6

aws.irreg, 10

aws.segment, 12

awsdata, 15

lpaws, 18

aws, 3, 9, 15, 16, 20

aws-package, 2

aws.gaussian, 6, 6, 15

aws.irreg, 6, 9, 10, 16, 18, 20

aws.segment, 12

awsdata, 15

binning, 17

lpaws, 6, 12, 18