Package 'LS2W'

May 7, 2026

Version 1.3.7

Date 2025-09-07

Title Locally Stationary Two-Dimensional Wavelet Process Estimation Scheme

Maintainer Idris Eckley <i.eckley@lancaster.ac.uk>

Depends R (>= 3.0), wavethresh (>= 4.5)

Imports MASS,methods

Description Estimates two-dimensional local wavelet spectra.

License GPL-2

NeedsCompilation yes

Author Idris Eckley [aut, cre],
Guy Nason [aut],
Sarah Taylor [ctb],
Matthew Nunes [ctb]

Repository CRAN

Date/Publication 2025-09-08 07:20:19 UTC

Contents

`LS2W-package` *Estimates the locally stationary wavelet spectrum for locally stationary 2-D wavelets processes*

Description

The LS2W package which provides an implementation of the modelling approach proposed by Eckley, Nason, and Treloar (2010) for locally stationary spatial covariance structure of (regular) lattice processes, such as images.

The approach, which is an extension of NvSK's time series modelling paradigm, is in part motivated by a frequently erent scales. Moreover, several researchers have highlighted that the human and mammalian visual systems process images in a multiscale manner, preserving both local and global information (see for example, Daugman (1990) or Field (1999)). The LS2W package which we introduce implements the estimation scheme described by Eckley et al. (2010) for such processes. Note that LS2W should be used in conjunction with the WaveThresh package developed by Nason (1998).

Details

Note that this package should be used in conjunction with the WaveThresh package. It is advised that WaveThresh should be loaded first, followed by LS2W. This is because LS2W currently overwrites a couple of WaveThresh functions to, e.g., provide consistent naming structures for some of the objects created during estimation of 2-D LSW processes.

The central function within the LS2W package is cddews. This executes the LS2W estimation algorithm as summarised in Algorithm 1, drawing on other functions to calculate the autocorrelation (ac) wavelets and their inner product matrix. The output of this function is stored in an object of class "cddews". Print and summary methods are provided for this class of objects.

Below we provide brief descriptions of the main functions contained within the package. Please see individual help files for additional information.

`PhiJ`: Computes discrete autocorrelation father wavelets. These are automatically stored within the R session, using a naming convention governed by Phi1Dname.

`PsiiJ`: Computes discrete autocorrelation mother wavelets. These are automatically stored within the R session, using a naming convention governed by Psi1Dname.

`D2ACW`: Computes two-dimensional autocorrelation wavelets. These are automatically stored within the R session, using a naming convention governed by Psi2Dname.

`D2autoplot`: Can be used to generate images of two-dimensional discrete autocorrelation wavelets.

`D2Amat`: Computes the inner product matrix of two-dimensional discrete autocorrelation wavelets. These objects are automatically stored within a session following a naming convention governed by A2name.

`cddews`: Computes the local wavelet spectrum estimate as described in Eckley et al. (2010), returning an object of class cddews.

`specplot`: Plots the local wavelet periodogram associated with a cddews object.

`Haar2MA.diag`: Generates a two-dimentional Haar MA object (diagonal direction) of specied size and order. The functions Haar2MA.vert, Haar2MA.horiz can be used to construct equivalent realisations for process in the vertical and horizontal direction respectively whilst HaarMontage generates a realisation of a concatenated two-dimensional Haar MA process (for an example, see Fig 3 of Eckley and Nason 2009).

Author(s)

Idris Eckley and Guy Nason, with contributions from Sarah Taylor and Matt Nunes Maintainer: Who to complain to <i.eckley@lancs.ac.uk>

References

Eckley, I.A., Nason, G.P. and Treloar, R.L. (2010) Locally stationary wavelet fields with application to the modelling and analysis of image texture. Journal of the Royal Statistical Society (Series C), 59, 595 - 616.

Eckley, I.A. and Nason, G.P. (2011). LS2W: Implementing the Locally Stationary 2D Wavelet Process Approach in R, Journal of Statistical Software, 43(3), 1-23. URL http://www.jstatsoft.org/v43/i03/.

Examples

```
#
# See comprehensive examples in the help pages for the major functions
# outlined above.
#
```

A — *Examples of textured images*

Description

A, B, and C are examples of different textured images (grayscale). Each matrix object contains an image. Each entry of the matrix contains an image intentsity value.

Usage

```
data(textures)
```

Format

Matricies of varying sizes which each contain a textured image

References

Eckley, I.A. and Nason, G.P. (2011). LS2W: Implementing the Locally Stationary 2D Wavelet Process Approach in R, Journal of Statistical Software, 43(3), 1-23. URL http://www.jstatsoft.org/v43/i03/.

A2name — *Return a D2Amat list object style name.*

Description

This function returns a character string according to a particular format for naming D2Amat objects.

Usage

```
A2name(J, filter.number, family, switch = "direction")
```

Arguments

`J`	Discrete autocorrelation wavelets will be computed for scales -1 up to scale J. This number should be a negative integer.
`filter.number`	The index number of the wavelet used to build the PsiJ/PhiJ object.
`family`	The wavelet family used to build the PhiJIE object.
`switch`	Allows the user to define how they wish their inner product matrix to be formed. There are two available options: `switch = "direction"` - structures the matrix by scale within each decomposition direction. Thus, the ordering goes as follows (-1, V), (-2, V), ... `switch = "scale"` - structures the matrix by direction within each scale. Thus the ordering is as follows (-1,V), (-1, H), (-1, D), (-2, V),(-2, H), ...

Details

Some of the matrices computed by D2Amat take a long time to compute. Hence it is a good idea to store and re-use them. This function generates a name according to a particular naming scheme that permits a search algorithm to easily find the matrices. Each matrix has three defining characteristics: its order, filter.number, family and the method of construction of the matrix (switch="direction" or "level"). Each of these characteristics are concatenated together to form a name.

Value

A character string containing the name of a matrix according to a particular naming scheme.

Author(s)

Idris Eckley

References

Eckley, I.A., Nason, G.P. and Treloar, R.L. (2010) Locally stationary wavelet fields with application to the modelling and analysis of image texture. Journal of the Royal Statistical Society (Series C), 59, 595 - 616.

Eckley, I.A. and Nason, G.P. (2011). LS2W: Implementing the Locally Stationary 2D Wavelet Process Approach in R, Journal of Statistical Software, 43(3), 1-23. URL http://www.jstatsoft.org/v43/i03/.

See Also

D2Amat

Examples

```
#
# What's the name of the order 4 Haar matrix when structured by scale within decompositon direction?
#
A2name(J=-4, filter.number=1, family="DaubExPhase", switch="direction")
#[1] "D2Amat.d.4.1.DaubExPhase"
#
# What's the name of the order 15 Daubechies least-asymmetric wavelet
# with 7 vanishing moments, when ordered by direction within each scale?
#
A2name(J=-15, filter.number=7, family="DaubLeAsymm")
#[1] "D2Amat.l.15.7.DaubLeAsymm"
```

`AvBasis.wst2D` *Perform basis averaging for (packet-ordered) 2D non-decimated wavelet transform.*

Description

Perform basis averaging for (packet-ordered) 2D non-decimated wavelet transform. Note: This is a copy of the function from wavethresh, but this version uses the LS2W C code implementation.

Usage

```
## S3 method for class 'wst2D'
AvBasis(wst2D, ...)
```

Arguments

`wst2D`	An object of class `wst2D` that contains coefficients of a packet ordered 2D non-decimated wavelet transform (e.g. produced by the `wst2D` function.
`...`	any other arguments

Details

The packet-ordered 2D non-decimated wavelet transform computed by `wst2D` computes the coefficients of an input matrix with respect to a library of all shifts of wavelet basis functions at all scales. Here "all shifts" means all integral shifts with respect to the finest scale coefficients with shifts in both the horizontal and vertical directions, and "all scales" means all dyadic scales from 0 (the coarsest) to J-1 (the finest) where `2^J = n` where `n` is the dimension of the input matrix. As such the packet-ordered 2D non-decimated wavelet transform contains a library of all possible shifted wavelet bases.

Basis averaging. Rather than select *a* basis it is often useful to preserve information from all of the bases. For examples, in curve estimation, after thresholding, the coefficients are coefficients of an estimate of the truth with respect to all of the shifted basis functions. Rather than select one of them we can average over all estimates. This sometimes gives a better curve estimate and can, for examples, get rid of Gibbs effects. See Coifman and Donoho (1995) for more information on how to do curve estimation using the packet ordered non-decimated wavelet transform, thresholding and basis averaging. See Lang et al. (1995) for further details of surface/image estimation using the 2D non-decimated DWT.

Value

A square matrix of dimension $2^nlevels$ containing the average-basis “reconstruction” of the `wst2D` object.

RELEASE

Version 3.9 Copyright Guy Nason 1998

Author(s)

G P Nason

See Also

wst2D, wst2D.object

Examples

```
#
# Generate some test data
#
#test.data <- matrix(rnorm(16), 4,4)
#
# Now take the 2D packet ordered DWT
#
#tdwst2D <- wst2D(test.data)
#
# Now "invert" it using basis averaging
#
#tdwstAB <- AvBasis(tdwst2D)
#
# Let's compare it to the original
#
#sum( (tdwstAB - test.data)^2)
#
# [1] 1.61215e-17
#
# Very small. They're essentially same.
#
```

---

cddews *Compute the local wavelet spectrum estimate*

Description

This function computes the local wavelet spectrum (LWS) estimate of an image (or non-decimated wavelet transform of a time series). The estimate is computed by taking the non-decimated wavelet transform of the image, squaring the detail coefficients, smoothing using wavelet shrinkage and then correcting the redundancy caused by use of the non-decimated wavelet transform.

Usage

```
cddews(data, filter.number = 1, family = "DaubExPhase", switch = "direction",
correct = TRUE, verbose = FALSE, smooth = TRUE,
sm.filter.number = 4, sm.family = "DaubExPhase", levels = 3:6, type = "hard",
policy = "LSuniversal", by.level = FALSE, value = 0, dev = var)
```

Arguments

`data`	The image you want to analyse.
`filter.number`	This selects the index of the wavelet used in the analysis of the time series (i.e.\ the wavelet basis functions used to model the time series). For Daubechies compactly supported wavelets the filter number is the number of vanishing moments.
`family`	This selects the wavelet family to use in the analysis of the time series (i.e.\ which wavelet family to use to model the time series). Only use the Daubechies compactly supported wavelets `DaubExPhase` and `DaubLeAsymm`.
`switch`	This allows one to order the corrected spectrum by scale or decomposition direction. Two options are available `switch = "direction"`: structures the matrix by scale within each decomposition direction. Thus, the ordering goes as follows $(-1, V), (-2, V), (-3, V)\ ...$. `switch = "level"` structures the matrix by direction within each scale. Thus the ordering is as follows $(-1,V), (-1, H), (-1, D), (-2, V), (-2, H), (-2, D), ...$. For further details, see Eckley Nason and Treloar (2010).
`correct`	Eckley, Nason and Treloar (2009) have demonstrated that, as a consequence of the inherent redundancy of the non-decimated wavelet transform, the raw wavelet spectrum is biased. However, an asymptotically unbiased estimator may be obtained by applying the inverse of the inner product matrix of discrete autocorrelation wavelets. This argument permits the user to decide whether or not to correct for this inherent bias.
`verbose`	Allows certain informative messages to be printed on screen.
`smooth`	This binary argument allows the user to specify whether or not the resulting local wavelet periodogram should be smoothed to obtain. It is advised that this option be set to `TRUE` in order that consistent estimates be obtained.
`sm.filter.number`	Selects the index number of the wavelet that smooths each scale of the wavelet periodogram.
`sm.family`	Selects the wavelet family that smooths each scale of the wavelet periodogram.
`levels`	This specifies the levels which are smoothed when performing the wavelet shrinkage.
`type`	The type of shrinkage: either `"hard"` or `"soft"`.
`policy`	This dictates the threshold selection method used for smoothing. For LWS estimation `LSuniversal` is recommended for thi Chi-squared nature of the periodogram coefficients.
`by.level`	If `TRUE` then the wavelet shrinkage is performed by computing and applying a separate threshold to each level in the transform of each scale. Note that each scale in the LWS is smoothed separately and independently. Each smooth consists of taking the (second-stage) non-decimated wavelet transform and applying a threshold to each level of a wavelet transformed scale. If `FALSE` then the same threshold is applied to the discrete wavelet transform of a scale. Different thresholds may be computed for different scales but the threshold will be the same for each level arising from the non-decimated transform of a scale.

value	This argument supplies the threshold value used when a manual policy is adopted.
dev	The method for estimating the variance of the empirical wavelet coefficients for smoothing purposes.

Details

This function computes an estimate of the directionally dependent wavelet spectrum of an image according to the work of Eckley, Nason and Treloar (2010). The function works as follows:

1. The non-decimated wavelet transform of the series is computed.

2. The squared modulus of the non-decimated wavelet transform is computed (this is the raw wavelet periodogram, which is returned).

3. The squared modulus is smoothed using wavelet shrinkage.

4. The smoothed coefficients are corrected using the inverse of the inner product matrix of the autocorrelation wavelets. To display the LWS use the specplot function on the S component (see the examples below).

Value

A list with the following components:

S:	The directionally dependent wavelet spectral estimate of the input data. This is a *large* array, the first dimension refers to a specific scale-direction pair (see Eckley et al. (2009) for further details). The next dimension refers to the rows of the spectral image, whilst the third element refers to the columns of the image.
datadim	The dimension of the original image.
filter.number:	This gives the index of the wavelet used in the analysis of the image (i.e. the wavelet basis functions used in the modelling). For Daubechies compactly supported wavelets the filter number is the number of vanishing moments.
family:	This contains the wavelet family used in the analysis of the image (i.e. the wavelet family used in the modelling).
structure:	Explains the structure of the inner product matrix and S. It can only take two values, direction and scale.
nlevels:	The number of levels in the decomposition.
correct:	TRUE or FALSE, depending on whether the user corrected for the bias.
smooth:	TRUE or FALSE, depending on whether the LWP has been smoothed.
date:	The date when the analysis was perfromed.

Author(s)

Idris Eckley

References

Eckley, I.A., Nason, G.P. and Treloar, R.L. (2010) Locally stationary wavelet fields with application to the modelling and analysis of image texture. Journal of the Royal Statistical Society (Series C), 59, 595 - 616.

Eckley, I.A. and Nason, G.P. (2011). LS2W: Implementing the Locally Stationary 2D Wavelet Process Approach in R, Journal of Statistical Software, 43(3), 1-23. URL http://www.jstatsoft.org/v43/i03/.

See Also

D2Amat, specplot

Examples

```
# Apply the cddews estimate function to a HaarMontage realisation
#
monty <- HaarMontage(direction="diagonal")
monty.cddews <- cddews(monty, filter.number=1, family="DaubExPhase")
monty.cddews
```

---

cdtoimwd *Convert a cddews type object to an imwd type object*

Description

Converting from a cddews object to an imwd object required for simulating an LS2W process.

Usage

```
cdtoimwd(cddews)
```

Arguments

cddews	A cddews object, which may be the output of cddews.

Details

Simulation of an LS2W process for Daubechies wavelets requires that our spectral structure is ordered as an imwd object rather than a cddews object. This function works by extracting the spectral information from the given cddews object, determining where in the (null) imwd object it should be and adding the power to this appropriate location. This function is used within the function LS2Wsim.cddews and as such does not need to be made use of directly by the user.

Value

An object of class imwd.

Author(s)

Sarah L Taylor

See Also

imwd, cddews, LS2Wsim.cddews

Examples

```
#
#Obtain a cddews type object from an image X
#
X<-matrix(rnorm(32*32),nrow=32,ncol=32)
#
CDDEWS <- cddews(X,correct=FALSE,smooth=FALSE)
#
#Verify the class of Matxcddews
#
class(CDDEWS)
#
#Convert to imwd
#
CDDEWSimwd<-cdtoimwd(CDDEWS)
#
#Verify new class
#
class(CDDEWSimwd)
#
```

`convertimwd` *Convert a non-decimated imwd object to a wst2D object.*

Description

Converting from a spatially ordered `imwd` object to a packet-ordered `wst2D` object.

Usage

```
convertimwd(imwd, ...)
```

Arguments

`imwd`	A imwd type object obtained by a stationary wavelet transform, i.e. `imwd$type` must be `"station"`.
`...`	Any additional arguments.

Details

Inverting a 2-D non-decimated wavelet transform requires that the output is structured as a packet ordered transform. This function allows us to convert from the non-decimated (time ordered) `imwd` object to the packet ordered `wst2D` object. The function extracts information at a given scale/direction from the `imwd` object, converts it into the appropriate format and inserts it in the appropriate location of a (null) `wst2D` object. This function is used when simulating an LS2W process within the function `LS2Wsim.cddews` and does not need to be used separately.

Value

An object of class `wst2D`

Author(s)

Matt Nunes

See Also

`imwd`, `wst2D`, `packetj`

Examples

```
#
#Obtain an imwd class object
#
testimage <- HaarMontage(256, "diagonal")
#
IMWDobject<- imwd(testimage, type="station")

# Verify the class of this object
#
class(IMWDobject)
#
#Convert to packet ordered
#
IMWDconverted<-convertimwd(IMWDobject)
#
#Verify new class
#
class(IMWDconverted)
#
```

`D1Amat` *Inner product matrix of ac wavelets(1-D)*

Description

This function calculates the inner product matrix of discrete autocorrelation wavelets (1D).

Usage

```
D1Amat(J, filter.number = 10, family = "DaubLeAsymm", tol = 1e-100, verbose = FALSE)
```

Arguments

J	The level to which the decomposition must extend. This number should be a positive integer.
filter.number	The index of the wavelet used to compute the correction m atrix A.
family	The wavelet family used to compute A.
tol	In the brute force computation for Daubechies compactly supported wavelets many inner product computations are performed. This tolerance discounts any results which are smaller than tol which effectively defines how long the inner product/autocorrelation products are.
verbose	Logical variable, if set to TRUE informative statements are printed to screen during execution of the function.

Value

A matrix of order (-J)x(-J) containing the inner product matrix of the discrete non-decimated autocorrelation matrices.

Note

An equivalent function ipndacw already exists in WaveThresh. This function is added to help create a consistent naming convention across both the one- and two-dimensional inner product matrix of ac wavelets.

Author(s)

Idris Eckley

References

Nason, G.P., von Sachs, R. and Kroisandt, G. (2000) Wavelet processes and adaptive estimation of the evolutionary wavelet spectrum. J. R. Statist. Soc. Series B, 62, 271-292.

Eckley, I.A. and Nason, G.P. (2011). LS2W: Implementing the Locally Stationary 2D Wavelet Process Approach in R, Journal of Statistical Software, 43(3), 1-23. URL http://www.jstatsoft.org/v43/i03/.

See Also

D2Amat

D2ACW *Compute 2-D discrete autocorrelation wavelets.*

Description

This function computes two-dimensional discrete autocorrelation wavelets. The inner products of these wavelets are required for correction of the (biased) raw wavelet periodograms.

Usage

```
D2ACW(J, filter.number = 1, family = "DaubExPhase", switch = "direction",
tol = 1e-100, OPLENGTH = 2000, verbose = FALSE)
```

Arguments

J	Discrete autocorrelation wavelets will be computed for scales 1 to J within each decomposition direction (horizontal, vertical and diagonal). This number should be a positive integer.
filter.number	The index of the wavelet used to compute the discrete autocorrelation wavelets.
family	The wavelet family used to compute the discrete autocorrelation wavelets.
switch	Allows the user to define how they wish their inner product matrix to be formed. There are two available options:\ switch = "direction" - structures the matrix by scale within each decomposition direction. Thus, the ordering goes as follows $(1, V), (2, V), ...$.\ switch = "level" - structures the matrix by direction within each scale. Thus the ordering is as follows $(-1,V), (-1, H), (-1, D), (-2, V), (-2, H),...$.
tol	In the brute force computation for Daubechies compactly supported wavelets many inner product computations are performed. This tolerance discounts any results which are smaller than tol which effectively defines how long the inner product/autocorrelation products are.
OPLENGTH	This integer variable defines some workspace of length OPLENGTH. The code uses this workspace. If the workspace is not long enough then the routine will stop and tell you what OPLENGTH should be set to.
verbose	If TRUE various informative statements are printed to screen.

Details

This function computes the 2-D discrete autocorrelation wavelets. It does not have any direct use for space-scale analysis. The construction method is a brute force approach – a more elegant solution would be based on the recursive schemes as described in Eckley and Nason (2005). The routine returns only the values of the discrete autocorrelation wavelets, not their spatial positions. Each discrete autocorrelation wavelet is compactly supported. This support is determined from the discrete wavelets upon which these autocorrelations are based.

Value

A list containing $3J$ components, numbered from 1 to $3J$. If switch="direction", the first J components contain the vertical autocorrelation wavelet coefficients, the second set of J components contains the horizontal autocorrelation wavelet coefficients (scales $1,\ldots, J$) and the last J components constitute the diagonal autocorrelation wavelet coefficients. However, if switch="level", then the first 3 components contain the finest scale autocorrelation wavelet coefficients in the vertical, horizontal and diagonal decomposition directions respectively. The second set of 3 contains the vertical, horizontal and diagonal coefficients at scale 2 etc. \\ Note that these 2-D autocorrelation wavelets are stored as matrices. The central element of the matrix refers to lag 0.

Author(s)

Idris Eckley

References

Eckley, I.A., Nason, G.P. and Treloar, R.L. (2010) Locally stationary wavelet fields with application to the modelling and analysis of image texture. Journal of the Royal Statistical Society (Series C), 59, 595 - 616.

Eckley, I.A. and Nason, G.P. (2011). LS2W: Implementing the Locally Stationary 2D Wavelet Process Approach in R, Journal of Statistical Software, 43(3), 1-23. URL http://www.jstatsoft.org/v43/i03/.

See Also

D2autoplot

Examples

```
#
# Let us create the discrete autocorrelation wavelets for the Haar wavelet.
# We shall create up to scale 2.
#
D2ACW(J=-2, filter.number=1, family="DaubExPhase", switch="direction")
#[[1]]:
#      [,1] [,2]  [,3]
#[1,] -0.25 -0.5 -0.25
#[2,]  0.50  1.0  0.50
#[3,] -0.25 -0.5 -0.25
#
#[[2]]:
#        [,1]   [,2]    [,3]  [,4]    [,5]   [,6]    [,7]
#[1,] -0.0625 -0.125 -0.1875 -0.25 -0.1875 -0.125 -0.0625
#[2,] -0.1250 -0.250 -0.3750 -0.50 -0.3750 -0.250 -0.1250
#[3,]  0.0625  0.125  0.1875  0.25  0.1875  0.125  0.0625
#[4,]  0.2500  0.500  0.7500  1.00  0.7500  0.500  0.2500
#[5,]  0.0625  0.125  0.1875  0.25  0.1875  0.125  0.0625
#[6,] -0.1250 -0.250 -0.3750 -0.50 -0.3750 -0.250 -0.1250
#[7,] -0.0625 -0.125 -0.1875 -0.25 -0.1875 -0.125 -0.0625
#
#... and the remaining terms follow suit.
```

D2ACWmat *Compute 2-D discrete autocorrelation wavelets but return in array form.*

Description

This function computes two-dimensional discrete autocorrelation wavelets, but results in a matrix form, rather than a list.

Usage

```
D2ACWmat(J, filter.number = 10, family = "DaubLeAsymm", switch = "direction",
OPLENGTH = 1e+05)
```

Arguments

J	Discrete autocorrelation wavelets will be computed for scales -1 up to scale J within each decomposition direction (horiznotal, vertical and diagonal). This number should be a negative integer.
filter.number	The index of the wavelet used to compute the discrete autocorrelation wavelets.
family	The family of wavelet used to compute the discrete autocorrelation wavelets.
switch	Allows the user to define how they wish their inner product matrix to be formed. There are two available options: switch = "direction" - structures the matrix by scale within each decomposition direction. Thus, the ordering goes as follows (-1, V), (-2, V), ... switch = "direction" - structures the matrix by direction within each scale. Thus the ordering is as follows (-1,V), (-1, H), (-1, D), (-2, V), (-2, H), ...
OPLENGTH	This integer variable defines some workspace of length OPLENGTH. The code uses this workspace. If the workspace is not long enough then the routine will stop and probably tell you what OPLENGTH should be set to.

Details

This function computes the 2-D discrete autocorrelation wavelets. It does not have any direct use for space-scale analysis (e.g. cddews). However, it is useful to be able to numerically compute the discrete autocorrelation wavelets for arbitrary wavelets and scales as there are still unanswered theoretical questions concerning the wavelets. The method is a brute force – a more elegant solution would probably be based on interpolatory schemes (see Eckley and Nason (2005) for example).

This routine returns only the values of the discrete autocorrelation wavelets and not their spatial positions. Each discrete autocorrelation wavelet is compactly supported with the support determined from the compactly supported wavelet that generates it. See the paper by Eckley, Nason, and Treloar which defines the spatial scale (but basically the finer scale discrete autocorrelation wavelets are interpolated versions of the coarser ones. When one goes from scale j to j-1 (negative j remember) an extra point is inserted between all of the old points and the discrete autocorrelation wavelet value is

computed there. Thus as J tends to negative infinity the numerical approximation tends towards the continuous autocorrelation wavelet.

This function stores any 2-D discrete autocorrelation wavelet sets that it computes. The Psiname2D function defines the naming convention for objects returned by this function. The storage mechanism is not as advanced as that for ipndacw and its subsidiary routines rmname and firstdot but helps a little bit.

Sometimes it is useful to have the discrete autocorrelation wavelets stored in matrix form. The D2ACWmat does this.

Value

A matrix containing -3J sub-matrices, each of dimension 2L_J-1 x 2L_J-1, where L_J denotes the support of the coarsest discret autocorrelation wavelet. Each sub-matrix, contains the values of the discrete autocorrelation wavelet for a different scale-direction pair.

The middle position of each sub-matrix is the value of the discrete autocorrelation wavelet at zero — by definition, this is always 1. The discrete autocorrelation wavelet is symmetric about this point.

If `switch="direction"`, the first -J sub-matrices contain the vertical autocorrelation wavelet coefficients, the second set of -J components contains the horizontal autocorrelation wavelet coefficients (scales -1, ...-J) and the last -J components constitute the diagonal autocorrelation wavelet coefficients.

However, if `switch="level"`, then the first 3 rows contain the finest scale autocorrelation wavelet coefficients in the vertical, horizontal and diagonal decomposition directions respectively. The second set of 3 contains the vertical, horizontal and diagonal coefficients at scale -2 etc, etc.

Author(s)

Idris Eckley

References

Eckley IA, Nason GP (2005). Efficient computation of the inner-product matrix of discrete autocorrelation wavelets. Statistics and Computing, 15, 83-92.

Eckley, I.A., Nason, G.P. and Treloar, R.L. (2010) Locally stationary wavelet fields with application to the modelling and analysis of image texture. Journal of the Royal Statistical Society (Series C), 59, 595 - 616.

Eckley, I.A. and Nason, G.P. (2011). LS2W: Implementing the Locally Stationary 2D Wavelet Process Approach in R, Journal of Statistical Software, 43(3), 1-23. URL http://www.jstatsoft.org/v43/i03/.

See Also

D2ACW, D2Amat, D2autoplot

Examples

```
#
# Let us create the discrete autocorrelation wavelets for the Haar wavelet.
# We shall create up to scale 4.
#
```

```
D2ACWmat(J=-2, filter.number=1, family="DaubExPhase")
#Computing The two-dimensional (discrete) autocorrelation coefficients:
#
#The output will be structured as follows ....
#
#
#
#Levels 1 to  2  contain the vertical autocorrelation wavelet coefficients.
#
#Levels  3  to  4  contain the horizontal autocorrelation wavelet coefficients.
#
#Levels  5  to  6  contain the horizontal autocorrelation wavelet coefficients.
#
#
#
#Returning precomputed version
#Returning precomputed version
#Returning precomputed version
#Took  NA  seconds
#         [,1]   [,2]    [,3]  [,4]    [,5]   [,6]    [,7]
# [1,]  0.0000  0.000  0.0000  0.00  0.0000  0.000  0.0000
# [2,]  0.0000  0.000  0.0000  0.00  0.0000  0.000  0.0000
# [3,]  0.0000  0.000 -0.2500 -0.50 -0.2500  0.000  0.0000
# [4,]  0.0000  0.000  0.5000  1.00  0.5000  0.000  0.0000
# [5,]  0.0000  0.000 -0.2500 -0.50 -0.2500  0.000  0.0000
# [6,]  0.0000  0.000  0.0000  0.00  0.0000  0.000  0.0000
# [7,]  0.0000  0.000  0.0000  0.00  0.0000  0.000  0.0000
# [8,] -0.0625 -0.125 -0.1875 -0.25 -0.1875 -0.125 -0.0625
# [9,] -0.1250 -0.250 -0.3750 -0.50 -0.3750 -0.250 -0.1250
#[10,]  0.0625  0.125  0.1875  0.25  0.1875  0.125  0.0625
#[11,]  0.2500  0.500  0.7500  1.00  0.7500  0.500  0.2500
#[12,]  0.0625  0.125  0.1875  0.25  0.1875  0.125  0.0625
#[13,] -0.1250 -0.250 -0.3750 -0.50 -0.3750 -0.250 -0.1250
#[14,] -0.0625 -0.125 -0.1875 -0.25 -0.1875 -0.125 -0.0625
#[15,]  0.0000  0.000  0.0000  0.00  0.0000  0.000  0.0000
#[16,]  0.0000  0.000  0.0000  0.00  0.0000  0.000  0.0000
#[17,]  0.0000  0.000 -0.2500  0.50 -0.2500  0.000  0.0000
#[18,]  0.0000  0.000 -0.5000  1.00 -0.5000  0.000  0.0000
#[19,]  0.0000  0.000 -0.2500  0.50 -0.2500  0.000  0.0000
#[20,]  0.0000  0.000  0.0000  0.00  0.0000  0.000  0.0000
#[21,]  0.0000  0.000  0.0000  0.00  0.0000  0.000  0.0000
#[22,] -0.0625 -0.125  0.0625  0.25  0.0625 -0.125 -0.0625
#[23,] -0.1250 -0.250  0.1250  0.50  0.1250 -0.250 -0.1250
#[24,] -0.1875 -0.375  0.1875  0.75  0.1875 -0.375 -0.1875
#[25,] -0.2500 -0.500  0.2500  1.00  0.2500 -0.500 -0.2500
#[26,] -0.1875 -0.375  0.1875  0.75  0.1875 -0.375 -0.1875
#[27,] -0.1250 -0.250  0.1250  0.50  0.1250 -0.250 -0.1250
#[28,] -0.0625 -0.125  0.0625  0.25  0.0625 -0.125 -0.0625
#[29,]  0.0000  0.000  0.0000  0.00  0.0000  0.000  0.0000
#[30,]  0.0000  0.000  0.0000  0.00  0.0000  0.000  0.0000
#[31,]  0.0000  0.000  0.2500 -0.50  0.2500  0.000  0.0000
#[32,]  0.0000  0.000 -0.5000  1.00 -0.5000  0.000  0.0000
#[33,]  0.0000  0.000  0.2500 -0.50  0.2500  0.000  0.0000
```

```
#[34,]  0.0000  0.000  0.0000  0.00  0.0000  0.000  0.0000
#[35,]  0.0000  0.000  0.0000  0.00  0.0000  0.000  0.0000
#[36,]  0.0625  0.125 -0.0625 -0.25 -0.0625  0.125  0.0625
#[37,]  0.1250  0.250 -0.1250 -0.50 -0.1250  0.250  0.1250
#[38,] -0.0625 -0.125  0.0625  0.25  0.0625 -0.125 -0.0625
#[39,] -0.2500 -0.500  0.2500  1.00  0.2500 -0.500 -0.2500
#[40,] -0.0625 -0.125  0.0625  0.25  0.0625 -0.125 -0.0625
#[41,]  0.1250  0.250 -0.1250 -0.50 -0.1250  0.250  0.1250
#[42,]  0.0625  0.125 -0.0625 -0.25 -0.0625  0.125  0.0625
```

`D2Amat` *Creates the A matrix required for analysing LS2W processes.*

Description

This function creates the matrix used to correct the raw periodogram of a LS2W process.

Usage

```
D2Amat(J, filter.number = 10, family = "DaubLeAsymm", OPLENGTH = 10000,
switch = "direction", verbose = FALSE)
```

Arguments

`J`	The level to which the decomposition must extend. This number should be a positive integer.
`filter.number`	The index of the wavelet used to compute the correction matrix A.
`family`	The wavelet family used to compute A.
`OPLENGTH`	This integer variable defines some workspace of length OPLENGTH which is used by the code. If the workspace is not long enough, then the routine will stop and tell you what OPLENGTH should be set to.
`switch`	Dictates the structure of the matrix [by direction or by scale].
`verbose`	Allows certain informative messages to be printed on screen.

Value

A matrix of order (3J)*(3J) containing the elements A_(j,l) defined in Eckley, Nason and Treloar (2010). Each element is the sum over all lags of the product of the matrix coefficients of a 2-D DACW matrix at level j_1 in direction $ l_1$ with that of another (not necessarily different) matrix of DACW coefficients at level j_2 in direction l_2. The structure of this matrix is as follows: the rows and columns of the matrix are labeled $1,..., 3J$ in accordance with the notation of Eckley, Nason and Treloar (2010). When `switch="direction"` the matrix has the following structure:

`Levels 1 to J`	the different levels of the decomposition in the vertical direction. $1=$fine and $J =$coarse scale.

`Levels J+1 to 2J`
: the different levels in the horizontal direction.

`Levels 2J+1 to 3J`
: the different directions in the diagonal direction.

When `switch="level"`, the row and column elements cycle as follows:\ level 1 vertical, level 1 horizontal, level 1 diagonal, level 2 vertical, etc.

Author(s)

Idris Eckley

References

Eckley, I.A., Nason, G.P. and Treloar, R.L. (2010) Locally stationary wavelet fields with application to the modelling and analysis of image texture. Journal of the Royal Statistical Society (Series C), 59, 595 - 616.

Eckley, I.A. and Nason, G.P. (2011). LS2W: Implementing the Locally Stationary 2D Wavelet Process Approach in R, Journal of Statistical Software, 43(3), 1-23. URL http://www.jstatsoft.org/v43/i03/.

Examples

```
#
# Let's compute the A matrix for the Haar wavelet in 2-D.
#
D2Amat(J=-2, filter.number=1, family="DaubExPhase", switch="direction")
#       1      2      3      4      5      6
#1 2.2500 1.3125 0.2500 0.3125 0.7500 0.9375
#2 1.3125 4.8125 0.3125 0.5625 0.1875 1.3125
#3 0.2500 0.3125 2.2500 1.3125 0.7500 0.9375
#4 0.3125 0.5625 1.3125 4.8125 0.1875 1.3125
#5 0.7500 0.1875 0.7500 0.1875 2.2500 0.5625
#6 0.9375 1.3125 0.9375 1.3125 0.5625 3.0625
#
# And now for the same matrix structured by level
#
D2Amat(J=-2, filter.number=1, family="DaubExPhase", switch="level")
#       1      2      3      4      5      6
#1 2.2500 0.2500 0.7500 1.3125 0.3125 0.9375
#2 0.2500 2.2500 0.7500 0.3125 1.3125 0.9375
#3 0.7500 0.7500 2.2500 0.1875 0.1875 0.5625
#4 1.3125 0.3125 0.1875 4.8125 0.5625 1.3125
#5 0.3125 1.3125 0.1875 0.5625 4.8125 1.3125
#6 0.9375 0.9375 0.5625 1.3125 1.3125 3.0625
```

D2autoplot *Plots a two-dimensional autocorrelation wavelet.*

Description

Plots a 2-D autocorrelation wavelet for a given wavelet ata specific scale and decomposition direction.

Usage

```
D2autoplot(J, filter.number = 1, family = "DaubExPhase", direction = 3,
main = "2-D Autocorrelation Wavelet", OPLENGTH = 10000, scaling = "other", box="TRUE")
```

Arguments

J	A negative integer representing the order of the autocorrelation wavelet to be displayed.
filter.number	The index of the wavelet to be used.
family	The wavelet family employed.
direction	This variable dictates which of the three possible wavelets at a given level is being displayed. Recall from the work of Daubechies, that one way of constructing 2-D wavelets results in three wavelets. One corresponds to the vertical components of an image, another refers to the horizontal components whilst the final wavelet deals with the diagonal. Thus, keeping this construction in mind, it is easy to see that there are three autocorrelation wavelets when we are in two dimensions. This argument simply allows the user to specify which decomposition direction is to be displayed. Direction 1 corresponds to the Vertical direction, Direction 2 to the Horizontal direction whilst specifying direction=3 means that we wish the Diagonal ACW to be displayed.
main	Allows the user to specify a common title for each of the various spectral plots generated
OPLENGTH	This integer variable defines some workspace of length OPLENGTH which is used by the code. If the workspace is not long enough, then the routine will stop and tell you what OPLENGTH should be set to.
scaling	Allows the user to scale the x and y-axes (of minimal use!). There are currently two options. The first is Haar, which scales the x and y axes so that their entries are those of the actual Haar 2-D ACW. The second is other - which basically allows S-plus to do its ow
box	Allows the user to specify whether the resulting plotted is boxed.

Value

None.

Side Effects

Produces an image corresponding to the 2-D discrete autocorrelation wavelet of a given wavelet at a specific scale and decomposition direction.

Author(s)

Idris Eckley

References

Eckley, I.A., Nason, G.P. and Treloar, R.L. (2010) Locally stationary wavelet fields with application to the modelling and analysis of image texture. Journal of the Royal Statistical Society (Series C), 59, 595 - 616.

Eckley, I.A. and Nason, G.P. (2011). LS2W: Implementing the Locally Stationary 2D Wavelet Process Approach in R, Journal of Statistical Software, 43(3), 1-23. URL http://www.jstatsoft.org/v43/i03/.

Examples

```
#
# To produce the Haar 2-D discrete autocorrelation wavelet at level -6
# in the diagonal horizontal direction type
#
 D2ACW(J=-6, filter.number=1, family="DaubExPhase")
#
#Now let's look at the actual autocorrelation wavelet at that level
#

 D2autoplot(J=-6, filter.number=1, family="DaubExPhase", scaling="Haar")
```

DWEnv *Environment containing precomputed objects.*

Description

Package environment for commonly used objects in the LS2W methodology.

Details

The DWEnv environment is used to store precomputed objects for the LS2W methodology. Certain computations can be fairly long or intensive, and thus it is useful to store precomputed objects for reuse. These objects are namely autocorrelation wavelets (output from PsiJ and PhiJ), and bias correction matrices produced by D2Amat.

`example.ls2w` *Example of how the LS2W package can be used in texture analysis*

Description

This function provides an example of how LS2W can be used to discriminate between three different textures. The approach provided simply consists of (i) sampling a number of subimages from the specified data sets; (ii) estimating the local wavelet spectrum properties for each sub-image; (iii) summarising this information in a feature vector; (iv) using all feature vectors to identify whether it is possible to discriminate between the different image types. Linear Discriminant Analysis is the approach which we adopt in this example.

Usage

```
example.ls2w(n=25, size=64)
```

Arguments

`n`	The number of sub-images to be sampled from each texture type.
`size`	The number of rows-columns required for each sub-image.

Value

An object of class `lda`.

Author(s)

Idris Eckley

References

Eckley, I.A., Nason, G.P. and Treloar, R.L. (2010) Locally stationary wavelet fields with application to the modelling and analysis of image texture. Journal of the Royal Statistical Society (Series C), 59, 595 - 616.

Eckley, I.A. and Nason, G.P. (2011). LS2W: Implementing the Locally Stationary 2D Wavelet Process Approach in R, Journal of Statistical Software, 43(3), 1-23. URL http://www.jstatsoft.org/v43/i03/.

`getdata` *Extracts detail coefficients from imwd object*

Description

This function is called by cddews to extract the detail coefficients from an object of class imwd.

Usage

```
getdata(imwd, switch, co.type = "sqr", verbose = FALSE)
```

Arguments

`imwd`	An object of class imwd.
`switch`	Dictates whether the extracted information is structured by `"direction"` or `"scale"`.
`co.type`	Dictates the coefficient type which is used. For cddews this must be `"sqr"`.
`verbose`	Allows certain informative statements to be printed to the screen

Value

An array which can be used by cddews.

Author(s)

Idris Eckley

References

Eckley, I.A., Nason, G.P. and Treloar, R.L. (2010) Locally stationary wavelet fields with application to the modelling and analysis of image texture. Journal of the Royal Statistical Society (Series C), 59, 595 - 616.

Eckley, I.A. and Nason, G.P. (2011). LS2W: Implementing the Locally Stationary 2D Wavelet Process Approach in R, Journal of Statistical Software, 43(3), 1-23. URL http://www.jstatsoft.org/v43/i03/.

See Also

cddews

`Haar2MA.diag` *Generate 2-D Haar MA process (diagonal/vertical/horizontal direction).*

Description

These functions generate an arbitrary number of observations from a Haar MA process of any order with a particular variance. We will focus here on `Haar2MA.diag` — the routine which generates processes having spectral structure solely in the diagonal decomposition direction.

Usage

```
Haar2MA.diag(n, sd = 1, order = 5)
```

Arguments

`n`	The dimension of the realisation that you want to create. Note that `n` does NOT have to be a power of two, though it is square (nxn).
`sd`	The standard deviation of the innovations.
`order`	The order of the MA process.

Details

A two-dimensional Haar MA process is a special kind of moving-average (MA) field. A *diagonal* Haar MA process of order k is a MA field of order 2^k-1, the coefficients of the process being given by the filter coefficients of the two-dimensional, discrete Haar wavelet at various scales within the diagonal direction. For example: the diagonal Haar MA field of order 1 is an MA process of order 1. It is possible to define such processes for other wavelets as well.

Value

A matrix containing a realisation of the specified dimension, order and standard deviation.

Author(s)

Idris Eckley

References

Eckley, I.A., Nason, G.P. and Treloar, R.L. (2010) Locally stationary wavelet fields with application to the modelling and analysis of image texture. Journal of the Royal Statistical Society (Series C), 59, 595 - 616.

Eckley, I.A. and Nason, G.P. (2011). LS2W: Implementing the Locally Stationary 2D Wavelet Process Approach in R, Journal of Statistical Software, 43(3), 1-23. URL http://www.jstatsoft.org/v43/i03/.

See Also

`HaarMontage`

Examples

```
#
# Generate a realisation of a diagonal component 2-D MA field
# of order 4.
#
image1 <- Haar2MA.diag(n=128, sd=3, order=4)
#
#
#
```

`Haar2MA.horiz` *Generate 2-D Haar MA process (horizontal direction).*

Description

These functions generate an arbitrary number of observations from a Haar MA process of any order with a particular variance. We will focus here on `Haar2MA.horiz` — the routine which generates processes having spectral structure solely in the horizonal decomposition direction.

Usage

```
Haar2MA.horiz(n, sd = 1, order = 5)
```

Arguments

`n`	The dimension of the realisation that you want to create. Note that `n` does NOT have to be a power of two, though it is square (nxn).
`sd`	The standard deviation of the innovations.
`order`	The order of the MA process.

Details

A two-dimensional Haar MA process is a special kind of moving-average (MA) field. A *horizontal* Haar MA process of order k is a MA field of order 2^k-1, the coefficients of the process being given by the filter coefficients of the two-dimensional, discrete Haar wavelet at various scales within the horizontal direction. For example the horizontal Haar MA field of order 1 is an MA process of order 1. It is possible to define such processes for other wavelets as well.

Value

A matrix containing a realisation of the specified dimension, order and standard deviation.

Author(s)

Idris Eckley

References

Eckley, I.A., Nason, G.P. and Treloar, R.L. (2010) Locally stationary wavelet fields with application to the modelling and analysis of image texture. Journal of the Royal Statistical Society (Series C), 59, 595 - 616.

Eckley, I.A. and Nason, G.P. (2011). LS2W: Implementing the Locally Stationary 2D Wavelet Process Approach in R, Journal of Statistical Software, 43(3), 1-23. URL http://www.jstatsoft.org/v43/i03/.

See Also

Haar2MA.diag,HaarMontage

Examples

```
#
# Generate a realisation of a diagonal component 2-D MA field
# of order 4.
#
image1 <- Haar2MA.horiz(n=128, sd=3, order=4)
#
#
#
```

`Haar2MA.vert` *Generate 2-D Haar MA process (vertical direction).*

Description

These functions generate an arbitrary number of observations from a Haar MA process of any order with a particular variance. We will focus here on `Haar2MA.vert` — the routine which generates processes having spectral structure solely in the vertical decomposition direction.

Usage

```
Haar2MA.vert(n, sd = 1, order = 5)
```

Arguments

`n`	The dimension of the realisation that you want to create. Note that `n` does NOT have to be a power of two, though it is square (nxn).
`sd`	The standard deviation of the innovations.
`order`	The order of the MA process.

Details

A two-dimensional Haar MA process is a special kind of moving-average (MA) field. A *vertical* Haar MA process of order k is a MA field of order 2^k-1, the coefficients of the process being given by the filter coefficients of the two-dimensional, discrete Haar wavelet at various scales within the vertical direction. For example: the vertical Haar MA field of order 1 is an MA process of order 1. It is possible to define such processes for other wavelets as well.

Value

A matrix containing a realisation of the specified dimension, order and standard deviation.

Author(s)

Idris Eckley

References

Eckley, I.A., Nason, G.P. and Treloar, R.L. (2010) Locally stationary wavelet fields with application to the modelling and analysis of image texture. Journal of the Royal Statistical Society (Series C), 59, 595 - 616.

Eckley, I.A. and Nason, G.P. (2011). LS2W: Implementing the Locally Stationary 2D Wavelet Process Approach in R, Journal of Statistical Software, 43(3), 1-23. URL http://www.jstatsoft.org/v43/i03/.

See Also

Haar2MA.diag, HaarMontage

Examples

```
#
# Generate a realisation of a diagonal component 2-D MA field
# of order 4.
#
image1 <- Haar2MA.vert(n=128, sd=3, order=4)
#
#
#
```

HaarMontage *Generate a 2-D Haar MA process.*

Description

This function generates a particular set of four 2-D Haar MA processes. These are subsequently collated to form a montage.

Usage

```
HaarMontage(n=128, direction = "diagonal", sd = 1)
```

Arguments

n	The dimension of the realisations that you want to create. Note that n does NOT have to be a power of two, though the output will always be square (n x n).
direction	Three directions can be specified: horizontal, vertical and diagonal. The direction chosen dictates the decomposition direction in which the wavelet spectral structure exists.
sd	The standard deviation of the innovations.

Details

This function generates a realisation of a particular kind of non-stationary lattice model, an example of which is displayed in figure 3 of Eckley, Nason and Treloar (2009). The returned lattice is the result of combining four HaarMA processes. One process is of order 1, whilst another is of order 2. The two remaining processes are of order 3 and 4 respectively. Each individual lattice has dimension n/2*n/2. The standard deviation of the innovations is sd.

Value

A vector containing n*n observations from four collated 2-D Haar MA processes.

Author(s)

Idris Eckley and Sarah Taylor

References

Eckley, I.A., Nason, G.P. and Treloar, R.L. (2010) Locally stationary wavelet fields with application to the modelling and analysis of image texture. Journal of the Royal Statistical Society (Series C), 59, 595 - 616.

Eckley, I.A. and Nason, G.P. (2011). LS2W: Implementing the Locally Stationary 2D Wavelet Process Approach in R, Journal of Statistical Software, 43(3), 1-23. URL http://www.jstatsoft.org/v43/i03/.

See Also

Haar2MA.diag,Haar2MA.horiz,Haar2MA.vert

Examples

```
# Generate a realisation of the non-stationary combined Haar MA
# process with structure in the vertical direction.
MyHaar2 <- HaarMontage(128,direction="diagonal",sd=1)
# Plot it.
#
 image(MyHaar2)
```

imwd *Two-dimensional wavelet transform (decomposition).*

Description

This function replaces WaveThresh's imwd which currently contains a minor bug (for the case `type="station"`_.The function can perform two types of two-dimensional discrete wavelet transform (DWT). The standard transform (`type="wavelet"`) computes the 2D DWT according to Mallat's pyramidal algorithm (Mallat, 1989). The spatially ordered non-decimated 2D DWT (NDWT) (`type="station"`) contains all possible spatially shifted versions of the DWT. The order of computation of the DWT is O(n), and it is O(n log n) for the NDWT if n is the number of pixels.

Usage

```
imwd(image, filter.number = 10, family = "DaubLeAsymm", type = "wavelet",
bc = "periodic", RetFather = TRUE, verbose = FALSE)
```

Arguments

`image`	A square matrix containing the image data you wish to decompose. The side-length of this matrix must be a power of 2.
`filter.number`	This selects the smoothness of wavelet that you want to use in the decomposition. By default this is 10, the Daubechies least-asymmetric orthonormal compactly supported wavelet with 10 vanishing moments.
`family`	Specifies the family of wavelets that you want to use. The options are "DaubExPhase" and "DaubLeAsymm".
`type`	Specifies the type of wavelet transform. This can be "wavelet" (default) in which case the standard 2D DWT is performed (as in previous releases of WaveThresh). If type is "station" then the 2D spatially-ordered non-decimated DWT is performed. At present, only periodic boundary conditions can be used with the 2D spatially ordered non-decimated wavelet transform.
`bc`	specifies the boundary handling. If bc=="periodic" the default, then the function you decompose is assumed to be periodic on it's interval of definition, if bc=="symmetric" then the function beyond its boundaries is assumed to be a symmetric reflection of the function in the boundary. The symmetric option was the implicit default in releases prior to 2.2. Note that only periodic boundary conditions are valid for the 2D spatially-ordered non-decimated wavelet transform.
`RetFather`	If TRUE then this argument causes the scaling function coefficients at each resolution level to be returned as well as the wavelet coefficients. If FALSE then no scaling function coefficients are returned. The opportunity of returning father wavelet coefficients has been added since previous versions of WaveThresh.
`verbose`	Controls the printing of "informative" messages whilst the computations progress. Such messages are generally annoying so it is turned off by default.

Value

An object of class imwd object containing the two-dimensional wavelet transform (possibly spatially-ordered non-decimated).

Author(s)

Idris Eckley and Guy Nason

References

Mallat, S.G. (1989b). A theory for multiresolution signal decomposition: the wavelet representation. IEEE Trans. Pattn Anal. Mach. Intell., 11, 674-693.

See Also

cddews

Examples

```
#
#First let's create an image
#
tmp <- HaarMontage(direction="diagonal")
#
# Now let's do the 2D discrete wavelet transform on the image.
#
lwd <- imwd(tmp)
#
# Let's look at the coefficients
#
plot(lwd)
```

---

imwr.imwd *Inverse two-dimensional discrete wavelet transform.*

Description

This functions performs the reconstruction stage of Mallat's pyramid algorithm (i.e. the inverse discrete wavelet transform) for images. NOTE: This function replaces the wavethresh version to use LS2W C code due to memory reasons.

Usage

```
## S3 method for class 'imwd'
imwr(imwd, bc=imwd$bc, verbose=FALSE, ...)
```

Arguments

imwd	An object of class 'imwd'. This type of object is returned by 'imwd'.
bc	This argument specifies the boundary handling, it is best left to be the boundary handling specified by that in the supplied imwd (as is the default).
verbose	If this argument is true then informative messages are printed detailing the computations to be performed
...	any other arguments

Details

Details of the algorithm are to be found in Mallat (1989). Similarly to the decomposition function, imwd the inverse algorithm works by applying many 1D reconstruction algorithms to the coefficients. The filters in these 1D reconstructions are incorporated in the supplied imwd.object and originally created by the filter.select function in WaveThresh3.

This function is a method for the generic function imwr for class imwd.object. It can be invoked by calling imwr for an object of the appropriate class, or directly by calling imwr.imwd regardless of the class of the object.

Value

A matrix, of dimension determined by the original data set supplied to the initial decomposition (more precisely, determined by the nlevels component of the imwd.object). This matrix is the highest resolution level of the reconstruction. If a imwd two-dimensional wavelet transform is followed immediately by a imwr inverse two-dimensional wavelet transform then the returned matrix will be exactly the same as the original image.

RELEASE

Version 3.5.3 Copyright Guy Nason 1994

Author(s)

G P Nason

See Also

imwd, imwd.object, imwr.

Examples

```
#
# Do a decomposition, then exact reconstruction
# Look at the error
#
test.image <- matrix(rnorm(32*32), nrow=32)
#
# Test image is just some sort of  square matrix whose side length
# is a power of two.
#
```

```
max( abs(imwr(imwd(test.image)) - test.image))
# [1] 1.014611e-11
```

LS2Wsim *Generic method for simulation of LS2W processes*

Description

Simulates a realisation of an LS2W process of a given spectral structure

Usage

```
LS2Wsim(spectrum,...)
```

Arguments

spectrum	True spectrum from which to simulate data.
...	Any other arguments passed into the simulation function methods, such as the distribution of the LS2W innovations.

Details

Generic method for simulating LS2W processes as defined by Eckley et al. (2010).

Value

Returns a single image matrix

Author(s)

Aimee Gott

References

Eckley, I.A., Nason, G.P., and Treloar, R.L. (2010) Locally stationary wavelet fields with application to the modelling and analysis of image texture *Journal of the Royal Statistical Society Series C*, **59**, 595-616.

`LS2Wsim.cddews` *Simulate an LS2W process with underlying Daubechies wavelet.*

Description

Simulates a realisation of an LS2W process of a given spectral structure for Daubechies wavelets.

Usage

```
## S3 method for class 'cddews'
LS2Wsim(spectrum,innov=rnorm,...)
```

Arguments

`spectrum`	True spectrum of class `cddews` which may be the output of `cddews`.
`innov`	The distribution of the innovations of the LS2W process. Defaults to the (unit) normal distribution.
`...`	Any other arguments passed into the `innov` function.

Details

This function uses the provided spectral structure to simulate an LS2W process. The provided spectral structure should take positive values, any negative values will be set to zero by the function.

The process of simulation for Daubechies wavelets follows by firstly extracting the coefficients from the provided spectrum and squaring. They are then multiplied by random increments (from `rnorm()`) and an appropriate multiple of 4 (depending on the scale) which allows for the effect of basis averaging. After converting to the appropriate form (a `wst2D` object) the basis average is taken to give a realisation of an LS2W process.

Value

A simulated image matrix that will exhibit the spectral characteristics defined by `spectrum`.

Author(s)

Sarah L Taylor

See Also

`HaarMontage`

Examples

```
#Generate an empty spectrum
#
Spectrum<-cddews(matrix(0,64,64),smooth=FALSE)
#
#Add power at the first scale, in the vertical direction
#
Spectrum$S[1,,]<-matrix(1,64,64)
#
# Simulate an LS2W process with this structure
#
testimage<- LS2Wsim(Spectrum)
#
```

`packetj` *Converts an imwd object to a wst2D object at a given level.*

Description

Reorders the informtion at a given level of an `imwd` object to be in the form required for a `wst2D` object.

Usage

```
packetj(imwd, level, o)
```

Arguments

`imwd`	An `imwd` object from a non-decimated wavelet transform, i.e. `type` is `station`.
`level`	The level we wish to convert.
`o`	Computes weaving permutation for conversion from imwd to wst2D.

Details

This function combines the information from an `imwd` object at a given level and reorders it to be in the approriate form for a `wst2D` object. This function is required by `convertimwd`.

Value

Returns a matrix to be put into a `wst2D` object.

Note

Not intended to be used

Author(s)

Matt Nunes

Examples

```
#Obtain an imwd object
#
testimage <- HaarMontage(256, "diagonal")
testimageIMWD<- imwd(testimage, type="station")
#
#Specify our weaving permutation
#
arrvec <- getarrvec(9, sort=FALSE)
#
#Convert level 6 coefficients into packet ordered object
#
o <- arrvec[,2]
packmat <- packetj(testimageIMWD, 6, o)
#
```

Phi1Dname *Return a PhiJ list object style name.*

Description

This function returns a character string according to a particular format for naming PhiJ objects.

Usage

```
Phi1Dname(J, filter.number, family)
```

Arguments

J	A negative integer representing the order of the PhiJ object.
filter.number	The index number of the wavelet used to build the PhiJ object.
family	The wavelet family used to build the PhiJ object.

Details

Some of the objects computed by PhiJ take a long time to compute. Hence it is a good idea to store them and reuse them. This function generates a name according to a particular naming scheme that permits a search algorithm to easily find the matrices.

Each object has three defining characteristics: its order, filter.number and family. Each of these three characteristics are concatenated together to form a name.

This function performs exactly the same role as rmname except for objects produced by PhiJ.

Value

A character string containing the name of an object according to a particular naming scheme.

Author(s)

Idris Eckley

References

Eckley, I.A., Nason, G.P. and Treloar, R.L. (2010) Locally stationary wavelet fields with application to the modelling and analysis of image texture. Journal of the Royal Statistical Society (Series C), 59, 595 - 616.

Eckley, I.A. and Nason, G.P. (2011). LS2W: Implementing the Locally Stationary 2D Wavelet Process Approach in R, Journal of Statistical Software, 43(3), 1-23. URL http://www.jstatsoft.org/v43/i03/.

Examples

```
#
# What's the name of the order 4 Haar PhiJ object?
#
 Phi1Dname(-4, filter.number=1, family="DaubExPhase")
#[1] "D1Phi.4.1.DaubExPhase"
#
# What's the name of the order 12 Daubechies least-asymmetric wavelet PhiJ
# with 7 vanishing moments?
#
 Phi1Dname(-12, filter.number=7, family="DaubLeAsymm")
#[1] "D1Phi.12.7.DaubLeAsymm"
```

PhiJ *Compute discrete autocorrelation scaling function.*

Description

This function computes discrete autocorrelation scaling function.

Usage

```
PhiJ(J, filter.number = 10, family = "DaubLeAsymm", tol = 1e-100,
OPLENGTH = 2000, verbose = FALSE)
```

Arguments

J	Discrete autocorrelation wavelets will be computed for scales -1 up to scale J. This number should be a negative integer.
filter.number	The index of the wavelet used to compute the discrete autocorrelation wavelets.
family	The family of wavelet used to compute the discrete autocorrelation wavelets.
tol	In the brute force computation for Daubechies compactly supported wavelets many inner product computations are performed. This tolerance discounts any results which are smaller than tol which effectively defines how long the inner product/autocorrelation products are.

OPLENGTH	This integer variable defines some workspace of length OPLENGTH. The code uses this workspace. If the workspace is not long enough then the routine will stop and probably tell you what OPLENGTH should be set to.
verbose	A logical variable. If set to TRUE certain helpful statements are printed to screen during execution of this fundcion.

Details

This function computes the discrete autocorrelation scaling function. It does not have any direct use for location-scale analysis (e.g. ewspec). However, it is useful to be able to numerically compute the discrete autocorrelation wavelets for arbitrary wavelets and scales as there are still unanswered theoretical questions concerning the wavelets. The method is a brute force – a more elegant solution would probably be based on interpolatory schemes.

Horizontal scale. This routine returns only the values of the discrete autocorrelation scaling function and not their horizontal positions. Each discrete autocorrelation scaling function is compactly supported with the support determined from the compactly supported wavelet that generates it. See the paper by Nason, von Sachs and Kroisandt which defines the horizontal scale (but basically the finer scale discrete autocorrelation scaling function are interpolated versions of the coarser ones. When one goes from scale j to j-1 (negative j remember) an extra point is inserted between all of the old points and the discrete autocorrelation scaling function value is computed there. Thus as J tends to negative infinity the numerical approximation tends towards the continuous autocorrelation scaling function.

This function stores any discrete autocorrelation wavelet sets that it computes. The storage mechanism is not as advanced as that for ipndacw and its subsidiary routines rmget and firstdot but helps a little bit. The PhinameIE function defines the naming convention for objects returned by this function.

Sometimes it is useful to have the discrete autocorrelation scaling functions stored in matrix form. The PhiJmat does this.

Value

A list containing -J components, numbered from 1 to -J. The [[j]]th component contains the discrete autocorrelation scaling function at scale j.

Author(s)

Idris Eckley

References

Eckley, I.A., Nason, G.P. and Treloar, R.L. (2010) Locally stationary wavelet fields with application to the modelling and analysis of image texture. Journal of the Royal Statistical Society (Series C), 59, 595 - 616.

Eckley, I.A. and Nason, G.P. (2011). LS2W: Implementing the Locally Stationary 2D Wavelet Process Approach in R, Journal of Statistical Software, 43(3), 1-23. URL http://www.jstatsoft.org/v43/i03/.

See Also

PsiJ

Examples

```
#
# Let us create the discrete autocorrelation wavelets for the Haar wavelet.
# We shall create up to scale 4.
#
haardw4<- PhiJ(-4, filter.number=1, family="DaubExPhase")
haardw4
#[[1]]:
#[1] 0.5  1.0 0.5
#
#[[2]]:
#[1] 0.25 0.50  0.75  1.00  0.75 0.50 0.25
#
#[[3]]:
# [1] 0.125 0.250 0.375 0.500 0.625  0.750  0.875  1.000  0.875  0.750
#[11] 0.625 0.500 0.375 0.250 0.125
#
#[[4]]:
# [1] 0.0625 0.1250 0.1875 0.2500 0.3125 0.3750 0.4375 0.5000 0.5625
#[10] 0.6250  0.6875  0.7500  0.8125  0.8750  0.9375  1.0000  0.9375  0.8750
#[19]  0.8125  0.7500  0.6875 0.6250 0.5625 0.5000 0.4375 0.3750 0.3125
#[28] 0.2500 0.1875 0.1250 0.0625
#
# You can plot the fourth component to get an idea of what the
# autocorrelation wavelet looks like.
#
# Note that the previous call stores the autocorrelation wavelet
# in D1.Phi.4.1.DaubExPhase in the environment DWEnv. This is mainly so that
# it doesn't have to be recomputed.
#
# Note that the x-coordinates in the following are approximate.
#
 plot(seq(from=-1, to=1, length=length(haardw4[[4]])),haardw4[[4]], type="l",
xlab = "t", ylab = "Haar Autocorrelation Scaling function")
```

`print.cddews` *Print out information about a cddews object in readable form.*

Description

This function prints out information about an cddews object in a nice human-readable form.

Note that this function is automatically called by R whenever the name of an ccdews object is typed or whenever such an object is returned to the top level of the R interpreter.

Usage

```
## S3 method for class 'cddews'
print(x, ...)
```

Arguments

x	An object of class ccdews that you wish to print out.
...	This argument actually does nothing in this function!

Details

See description.

Side Effects

Prints out information about ccdews objects in nice readable format.

Author(s)

Idris Eckley

References

Eckley, I.A., Nason, G.P. and Treloar, R.L. (2010) Locally stationary wavelet fields with application to the modelling and analysis of image texture. Journal of the Royal Statistical Society (Series C), 59, 595 - 616.

Eckley, I.A. and Nason, G.P. (2011). LS2W: Implementing the Locally Stationary 2D Wavelet Process Approach in R, Journal of Statistical Software, 43(3), 1-23. URL http://www.jstatsoft.org/v43/i03/.

See Also

cddews, summary.cddews

Examples

```
#
# Generate a cddews object for a HaarMontage realisation
#
monty <- HaarMontage(direction="diagonal")
tmp <- cddews(monty, filter.number=1, family="DaubExPhase", switch="direction",correct=FALSE)
#
# Now get R to use print.cddews
#
tmp
```

`Psi1Dname` *Return a PsiJ list object style name.*

Description

This function returns a character string according to a particular format for naming PsiJ objects. Note that this construct is different from that used in WaveThresh as we wish to be able to differentiate between one and two-dimensional autocorrelation wavelets!

Usage

```
Psi1Dname(J, filter.number, family)
```

Arguments

`J`	A negative integer representing the order of the PsiJ object.
`filter.number`	The index number of the wavelet used to build the PsiJ object.
`family`	The wavelet family used to build the PsiJ object.

Details

Some of the objects computed by PsiJ take a long time to compute. Hence it is a good idea to store them and reuse them. This function generates a name according to a particular naming scheme that permits a search algorithm to easily find the matrices.

Each object has three defining characteristics: its order, filter.number and family. Each of these three characteristics are concatenated together to form a name.

This function performs exactly the same role as rmname except for objects produced by PsiJ.

Value

A character string containing the name of an object according to a particular naming scheme.

Author(s)

Idris Eckley

References

Nason, G.P., von Sachs, R. and Kroisandt, G. (1998). Wavelet processes and adaptive estimation of the evolutionary wavelet spectrum. Technical Report, Department of Mathematics University of Bristol/ Fachbereich Mathematik, Kaiserslautern.

Eckley, I.A. and Nason, G.P. (2011). LS2W: Implementing the Locally Stationary 2D Wavelet Process Approach in R, Journal of Statistical Software, 43(3), 1-23. URL http://www.jstatsoft.org/v43/i03/.

See Also

`PsiJ`,`Psi1Dname`

Examples

```
#
# What's the name of the order 4 Haar PsiJ object?
#
 Psiname(-4, filter.number=1, family="DaubExPhase")
#[1] "D1Psi.4.1.DaubExPhase"
#
# What's the name of the order 12 Daubechies least-asymmetric wavelet PsiJ
# with 7 vanishing moments?
#
 Psiname(-12, filter.number=7, family="DaubLeAsymm")
#[1] "D1Psi.12.7.DaubLeAsymm"
```

`Psi2Dname` *Return a D2ACW list object style name.*

Description

This function returns a character string according to a particular format for naming D2ACW objects.

Usage

```
Psi2Dname(J, filter.number, family, switch)
```

Arguments

`J`	A negative integer representing the order of the D2ACW object.
`filter.number`	The index number of the wavelet used to build the D2ACW object.
`family`	The wavelet family used to build the D2ACW object.
`switch`	Can take the values `direction` or `level` - enabling control of whether the returned value is structured by scales within a given direction (-1, D), (-2, D), (-3, D), ... or by direction within scales (-1, H), (-1, V), (-1, D), (-2, H), ...

Details

Some of the objects computed by D2ACW take a long time to compute. Hence it is a good idea to store them and reuse them. This function generates a name according to a particular naming scheme that permits a search algorithm to easily find the matrices.

Each object has three defining characteristics: its order, filter.number and family. Each of these three characteristics are concatenated together to form a name.

This function performs exactly the same role as rmname except for objects produced by D2ACW.

Value

A character string containing the name of an object according to a particular naming scheme.

Author(s)

Idris Eckley

References

Eckley, I.A. and Nason, G.P. (2005). Efficient computation of the inner-product matrix of discrete autocorrelation wavelets. Statistics and Computing, 15, 83-92.

Eckley, I.A. and Nason, G.P. (2011). LS2W: Implementing the Locally Stationary 2D Wavelet Process Approach in R, Journal of Statistical Software, 43(3), 1-23. URL http://www.jstatsoft.org/v43/i03/.

See Also

D2ACW,Psi1Dname

Examples

```
#
# What's the name of the order 4 Haar PsiJ object?
#
 Psi2Dname(-4, filter.number=1, family="DaubExPhase", switch="direction")
#[1] "D1Psi.d.4.1.DaubExPhase"
```

PsiJ *Compute discrete autocorrelation wavelets.*

Description

This function computes discrete autocorrelation wavelets.

The inner products of the discrete autocorrelation wavelets are computed by the routine ipndacw.

Usage

```
PsiJ(J, filter.number = 10, family = "DaubLeAsymm", tol = 1e-100,
OPLENGTH = 2000, verbose = FALSE)
```

Arguments

J	Discrete autocorrelation wavelets will be computed for scales -1 up to scale J. This number should be a negative integer.
filter.number	The index of the wavelet used to compute the discrete autocorrelation wavelets.
family	The family of wavelet used to compute the discrete autocorrelation wavelets.
tol	In the brute force computation for Daubechies compactly supported wavelets many inner product computations are performed. This tolerance discounts any results which are smaller than tol which effectively defines how long the inner product/autocorrelation products are.

OPLENGTH	This integer variable defines some workspace of length OPLENGTH. The code uses this workspace. If the workspace is not long enough then the routine will stop and probably tell you what OPLENGTH should be set to.
verbose	If this TRUE certain informative statements are printed to the screen when this function is being executed.

Details

This function computes the discrete autocorrelation wavelets. It does not have any direct use for time-scale analysis (e.g. ewspec). However, it is useful to be able to numerically compute the discrete autocorrelation wavelets for arbitrary wavelets and scales as there are still unanswered theoretical questions concerning the wavelets. The method is a brute force – a more elegant solution would probably be based on interpolatory schemes.

Horizontal scale. This routine returns only the values of the discrete autocorrelation wavelets and not their horizontal positions. Each discrete autocorrelation wavelet is compactly supported with the support determined from the compactly supported wavelet that generates it. See the paper by Nason, von Sachs and Kroisandt which defines the horizontal scale (but basically the finer scale discrete autocorrelation wavelets are interpolated versions of the coarser ones. When one goes from scale j to j-1 (negative j remember) an extra point is inserted between all of the old points and the discrete autocorrelation wavelet value is computed there. Thus as J tends to negative infinity the numerical approximation tends towards the continuous autocorrelation wavelet.

This function stores any discrete autocorrelation wavelet sets that it computes. The storage mechanism is not as advanced as that for ipndacw and its subsidiary routines rmget and firstdot but helps a little bit. The Psiname function defines the naming convention for objects returned by this function.

Sometimes it is useful to have the discrete autocorrelation wavelets stored in matrix form. The PsiJmat does this.

Value

A list containing -J components, numbered from 1 to -J. The [[j]]th component contains the discrete autocorrelation wavelet at scale j.

Author(s)

Idris Eckley

References

Eckley, I.A., Nason, G.P. and Treloar, R.L. (2010) Locally stationary wavelet fields with application to the modelling and analysis of image texture. Journal of the Royal Statistical Society (Series C), 59, 595 - 616.

Eckley, I.A. and Nason, G.P. (2011). LS2W: Implementing the Locally Stationary 2D Wavelet Process Approach in R, Journal of Statistical Software, 43(3), 1-23. URL http://www.jstatsoft.org/v43/i03/.

See Also

Psi1Dname,PhiJ

Examples

```
#
# Let us create the discrete autocorrelation wavelets for the Haar wavelet.
# We shall create up to scale 4.
#
haardw4<-PsiJ(-4, filter.number=1, family="DaubExPhase", verbose=TRUE)
haardw4
#Computing PsiJ
#Returning precomputed version
#Took  0.00999999  seconds
#
#[[1]]:
#[1] -0.5  1.0 -0.5
#
#[[2]]:
#[1] -0.25 -0.50  0.25  1.00  0.25 -0.50 -0.25
#
#[[3]]:
# [1] -0.125 -0.250 -0.375 -0.500 -0.125  0.250  0.625  1.000  0.625  0.250
#[11] -0.125 -0.500 -0.375 -0.250 -0.125
#
#[[4]]:
# [1] -0.0625 -0.1250 -0.1875 -0.2500 -0.3125 -0.3750 -0.4375 -0.5000 -0.3125
#[10] -0.1250  0.0625  0.2500  0.4375  0.6250  0.8125  1.0000  0.8125  0.6250
#[19]  0.4375  0.2500  0.0625 -0.1250 -0.3125 -0.5000 -0.4375 -0.3750 -0.3125
#[28] -0.2500 -0.1875 -0.1250 -0.0625
#
#
# You can plot the fourth component to get an idea of what the
# autocorrelation wavelet looks like.
#
# Note that the previous call stores the autocorrelation wavelet
# in D1Psi.4.1.DaubExPhase in the environment DWEnv. This is mainly so that it doesn't have to
# be recomputed.
#
# Note that the x-coordinates in the following are approximate.
#
 plot(seq(from=-1, to=1, length=length(haardw4[[4]])),haardw4[[4]], type="l",
xlab = "t", ylab = "Haar Autocorrelation Wavelet")
#
#
# Now let us repeat the above for the Daubechies Least-Asymmetric wavelet
# with 10 vanishing moments.
# We shall create up to scale 6, a higher resolution version than last
# time.
#
PsiJ(-6, filter.number=10, family="DaubLeAsymm", OPLENGTH=5000)
#[[1]]:
# [1]  3.537571e-07  5.699601e-16 -7.512135e-06 -7.705013e-15  7.662378e-05
# [6]  5.637163e-14 -5.010016e-04 -2.419432e-13  2.368371e-03  9.976593e-13
#[11] -8.684028e-03 -1.945435e-12  2.605208e-02  6.245832e-12 -6.773542e-02
#[16]  4.704777e-12  1.693386e-01  2.011086e-10 -6.209080e-01  1.000000e+00
```

```
#[21] -6.209080e-01  2.011086e-10  1.693386e-01  4.704777e-12 -6.773542e-02
#[26]  6.245832e-12  2.605208e-02 -1.945435e-12 -8.684028e-03  9.976593e-13
#[31]  2.368371e-03 -2.419432e-13 -5.010016e-04  5.637163e-14  7.662378e-05
#[36] -7.705013e-15 -7.512135e-06  5.699601e-16  3.537571e-07
#
#[[2]]
#        scale 2 etc. etc.
#
#[[3]]   scale 3 etc. etc.
#
#scales [[4]] and [[5]]...
#
#[[6]]
#...
#   remaining scale 6 elements...
#...
#[2371] -1.472225e-31 -1.176478e-31 -4.069848e-32 -2.932736e-41  6.855259e-33
#[2376]  5.540202e-33  2.286296e-33  1.164962e-42 -3.134088e-35  3.427783e-44
#[2381] -1.442993e-34 -2.480298e-44  5.325726e-35  9.346398e-45 -2.699644e-36
#[2386] -4.878634e-46 -4.489527e-36 -4.339365e-46  1.891864e-36  2.452556e-46
#[2391] -3.828924e-37 -4.268733e-47  4.161874e-38  3.157694e-48 -1.959885e-39
#
#
# Let's now plot the 6th component (6th scale, this is the finest
# resolution, all the other scales will be coarser representations)
#
# Note that the previous call stores the autocorrelation wavelet
# in D1Psi.6.10.DaubLeAsymm in the DWEnv environment.
#
# Note that the x-coordinates in the following are non-existant!
#
 #
LA10l6<-get("D1Psi.6.10.DaubLeAsymm",envir=DWEnv)

 plot(seq(from=-1, to=1, length=length(LA10l6[[6]])),LA10l6[[6]], type="l",
xlab="t", ylab ="Daubechies N=10 least-asymmetric Autocorrelation Wavelet")
```

`sample.stats` *A function to calculate sample statistics for textured images using LS2W*

Description

This is one, of many, possible ways of calculating a feature vector for a given textured image using the LS2W modelling framework. Please refer to Eckley et al. (2010) for details about the texture statistic being used. This function is only intended to be used with sample.stats.

Usage

```
sample.stats(x, n=25, size=64)
```

Arguments

x	The textured image which is going to be analysed.
n	The number of sub-images to be sampled from the main texture.
size	The number of rows-columns required for each sub-image.

Value

A matrix containing the feature vectors for each sub-image. Each row contains the feature vector for one specific subimage

Author(s)

Idris Eckley

References

Eckley, I.A., Nason, G.P. and Treloar, R.L. (2010) Locally stationary wavelet fields with application to the modelling and analysis of image texture. Journal of the Royal Statistical Society (Series C), 59, 595 - 616.

Eckley, I.A. and Nason, G.P. (2011). LS2W: Implementing the Locally Stationary 2D Wavelet Process Approach in R, Journal of Statistical Software, 43(3), 1-23. URL http://www.jstatsoft.org/v43/i03/.

See Also

example.ls2w

---

specplot *Plot the LWP associated with a cddews object*

Description

This function displays the LWP associated with a cddews object, allowing the user to dictate display type.

Usage

```
specplot(cddews, scaling = "by.level", arrangement = c(3, 3), page = TRUE,
dataname = "Image", verbose = FALSE, display = "persp", reset = TRUE, wtitle="TRUE")
```

Arguments

cddews	An object of class cddews must be supplied to the function.
scaling	Two scaling options are available. The default setting is to scale "by.level" – an option which is useful if you wish to compare coefficients within a resolution level. The alternative setting is global, whereby one scale factor is chosen for all plots. This factor depends on the largest coefficient which is to be included in the suite of plots.

arrangement	Allows the user to specify the number of spectral plots which are to appear on any given page.
page	An argument which allows the user to request that they be prompted when a new page of plots appears. Two options are available: TRUE or FALSE.
dataname	A name for the image whose LWP is being displayed. This will appear as part of the title associated with each plot.
verbose	If set to TRUE certain informative statements are printed to screen during execution.
display	Two display methods are available. Using the option display="persp" displays a 3-dimensional plot of the LWP, using the routine persp. The option display="image" displays the LWP as a collection of images.
reset	If set to TRUE, this restores the plot settings to their default configuration (i.e. par(mfrow=c(1,1))). If FALSE, then the current settings will remain in operation.
wtitle	A logical variable which dictates whether a common title is displayed on all spectral plots.

Value

No value is returned.

Author(s)

Idris Eckley

References

Eckley, I.A., Nason, G.P. and Treloar, R.L. (2010) Locally stationary wavelet fields with application to the modelling and analysis of image texture. Journal of the Royal Statistical Society (Series C), 59, 595 - 616.

Eckley, I.A. and Nason, G.P. (2011). LS2W: Implementing the Locally Stationary 2D Wavelet Process Approach in R, Journal of Statistical Software, 43(3), 1-23. URL http://www.jstatsoft.org/v43/i03/.

See Also

cddews

Examples

```
#
# First let us create a textured image and create a cddews object.
#
## Not run:
monty <- HaarMontage(direction="diagonal")
monty.cddews <- cddews(monty, filter.number=1, family="DaubExPhase")
#
# Finally let's view this using a perspective plot.
#
```

```
specplot(monty.cddews, display = "persp")

## End(Not run)
```

`summary.cddews` *Use summary on a cddews object*

Description

This function prints out more information about a cddews object in a nice readable format.

Usage

```
## S3 method for class 'cddews'
summary(object, ...)
```

Arguments

`object`	An object of class cddews about which you wish to print out more information.
`...`	This argument actually does nothing in this function!

Value

No value is returned.

Author(s)

Idris Eckley

References

Eckley, I.A., Nason, G.P. and Treloar, R.L. (2010) Locally stationary wavelet fields with application to the modelling and analysis of image texture. Journal of the Royal Statistical Society (Series C), 59, 595 - 616.

Eckley, I.A. and Nason G.P. (2011). LS2W: Implementing the Locally Stationary 2D Wavelet Process Approach in R, Journal of Statistical Software, 43(3), 1-23. URL http://www.jstatsoft.org/v43/i03/.

See Also

`cddews`, `print.cddews`

`threshold.imwd` *Threshold two-dimensional wavelet decomposition object*

Description

This funcion provides various ways to threshold a `imwd` class object. It contains a minor variation of the equivalent WaveThresh function, which is necessary for the LS2W package.

Usage

```
## S3 method for class 'imwd'
threshold(x, levels = 3:(x$nlevels - 1), type = "hard", policy = "universal",
by.level = FALSE, value = 0, dev = var, verbose = FALSE,
return.threshold = FALSE, compression = TRUE, Q = 0.05, ...)
```

Arguments

`x`	The two-dimensional wavelet decomposition object that you wish to threshold.
`levels`	a vector of integers which determines which scale levels are thresholded in the decomposition. Each integer in the vector must refer to a valid level in the `imwd` object supplied. This is usually any integer from 0 to `nlevels(wd)-1` inclusive. Only the levels in this vector contribute to the computation of the threshold and it application.
`type`	Determines whether the type of thresholding is `"hard"` or `"soft"`.
`policy`	selects the technique by which the threshold value is selected. Each policy corresponds to a method in the literature. At present the different policies are: `"universal"`, `"manual"`, `"fdr"`, `"probability"`.
`by.level`	If `FALSE` then a global threshold is computed on and applied to all scale levels defined in levels. If `TRUE` a threshold is computed and applied separately to each scale level.
`value`	This argument conveys the user supplied threshold. If the `policy="manual"` then value is the actual threshold value; if `policy="probability"` then value conveys the the user supplied quantile level.
`dev`	This argument supplies the function to be used to compute the spread of the absolute values coefficients. The function supplied must return a value of spread on the variance scale (i.e. not standard deviation) such as the `var()` function. A popular, useful and robust alternative is the `madmad` function.
`verbose`	If `TRUE` then the function prints out informative messages as it progresses.
`return.threshold`	If this option is `TRUE` then the actual value of the threshold is returned. If this option is `FALSE` then a thresholded version of the input is returned.
`compression`	If this option is `TRUE` then this function returns a comressed two-dimensional wavelet transform object of class imwdc. This can be useful as the resulting object will be smaller than if it was not compressed. The compression makes use of the fact that many coefficients in a thresholded object will be exactly zero. If this option is `FALSE` then a larger imwd object will be returned.

Q	Parameter for the false discovery rate "fdr" policy.
...	There are no other arguments for this function!

Value

An object of class imwdc if the compression option above is TRUE, otherwise a imwd object is returned. In either case the returned object contains the thresholded coefficients. Note that if the return.threshold option is set to TRUE then the threshold values will be returned rather than the thresholded object.

Author(s)

Idris Eckley and Guy Nason

References

Please refer to the equivalent wavethresh help page.

Guy Nason (2010). wavethresh: Wavelets statistics and transforms. R package version 4.5. URL http://CRAN.R-project.org/package=wavethresh

See Also

imwd

Index