Package 'LS2Wstat'

February 3, 2020

Type Package

Title A Multiscale Test of Spatial Stationarity for LS2W Processes

Version 2.1-2

Date 2020-02-02

Depends LS2W(>= 1.3-1), matrixStats, RandomFields, spdep

Description Wavelet-based methods for testing stationarity and quadtree segmenting of images, see Taylor et al (2014) <doi:10.1080/00401706.2013.823890>.

License GPL-2

Author Sarah Taylor [aut],
Matt Nunes [aut, cre],
Idris Eckley [ctb, ths]

Maintainer Matt Nunes <nunesrpackages@gmail.com>

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2020-02-03 20:40:11 UTC

R topics documented:

LS2Wstat-package 2
avespecvar 3
compareImages 4
countTextures 5
cropimage 6
getpval 7
imageQT 8
lincurve 10
mix2images 11
plot.imageQT 12
plotmtx 13
print.imageQT 14
print.TOS2D 15

scurve 16
simTexture 17
TOS2D 19

Index **21**

`LS2Wstat-package` *Stationarity testing for locally stationary wavelet fields*

Description

This package contains functions for testing for stationarity within images, specifically locally stationary wavelet (LS2W) fields. In addition the package contains functions for implementing quadtree image decompositions, as well as code for simulating LS2W processes for a given spectral structure.

Details

Package:	LS2Wstat
Type:	Package
Version:	2.1
Date:	2018-05-21
License:	GPL-2
LazyLoad:	yes

Author(s)

Sarah L. Taylor and Matt Nunes

Maintainer: Matthew Nunes <nunesrpackages@gmail.com>

References

For details on testing LS2W fields for stationarity, see

Taylor, S.L., Eckley, I.A., and Nunes, M.A. (2014) A Test of Stationarity for Textured Images. *Technometrics*, **56** (3), 291-301.

For further information on LS2W processes, see

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.

avespecvar *A test statistic for spatial stationarity.*

Description

Calculates a test statistic for a test of stationarity based on the local wavelet spectrum.

Usage

```
avespecvar(spectrum)
```

Arguments

spectrum	A local wavelet spectrum estimate, i.e. a cddews object.

Details

The test statistic given by Taylor et al. (2014) for a test for stationarity is computed for use in the boostrap testing procedure (TOS2D).

Value

statistic	The value of the test statistic for the given spectrum.

Author(s)

Sarah L. Taylor

References

Taylor, S.L., Eckley, I.A., and Nunes, M.A. (2014) A Test of Stationarity for Textured Images. *Technometrics*, **56** (3), 291-301.

See Also

TOS2D

Examples

```
#Generate a cddews object
#
X <- Haar2MA.diag(64)

testspec<-cddews(X,smooth=FALSE)

#Find the value of the test statistic
#
avespecvar(testspec)
#
```

compareImages *Assesses whether two textured images are the same texture.*

Description

The function combines two images together, and then tests the montage for stationarity.

Usage

```
compareImages(Im1, Im2, testsize = min(nrow(Im1), nrow(Im2)), alpha=0.05,...)
```

Arguments

Im1	The first image to be compared.
Im2	The second image to be compared.
testsize	The size of the combined image montage to be tested for stationarity.
alpha	The significance of the stationarity test.
...	Any other optional arguments to TOS2D.

Details

An image *montage* of two images is created, and the homogeneity measure TOS2D is used in combination with getpval to assess stationarity of the montage. If the image is assessed as stationary, the two images are considered as the same texture.

Value

montageres	A boolean value indicating whether the montage of Im1 and Im2 is stationary.

Author(s)

Sarah L. Taylor and Matt Nunes

References

Taylor, S.L., Eckley, I.A., and Nunes, M.A. (2014) A Test of Stationarity for Textured Images. *Technometrics*, **56** (3), 291-301.

See Also

TOS2D, countTextures

Examples

```
# create two images to be compared:
X1<-simTexture(32,K=1,imtype="S1")[[1]]
X2<-simTexture(32,K=1,imtype="S1", sd=1.6)[[1]]

# use the test to compare them:

test<-compareImages(X1,X2,nsamples=100, smooth=FALSE)
```

---

countTextures *countTextures*

Description

Groups a list of (stationary) images into texture classes.

Usage

```
countTextures(Imgs, medpol = TRUE, ...)
```

Arguments

Imgs	A list of images to classify into textures.
medpol	A boolean value indicating whether to zero mean the images (with Tukey's median polish) prior to classification.
...	Any other optional arguments to the classification function compareImages.

Details

The procedure recursively uses the function compareImages to decide whether two images are of the same texture or not. More specifically, the first image is sequentially tested with all others in the list, assigning the images the label "1" if assessed as the same texture as the first image. All other (unclassified) images are then similarly compared with candidates from different texture classes, until all images have been assigned a group label. Testing recursively in this way, there are at most choose(length(Imgs),2) comparisons performed, but in reality the number could be a lot fewer.

Value

Iclass	A vector (of length length(Imgs)) of texture labels corresponding to each image in Imgs.

Author(s)

Sarah L. Taylor and Matt Nunes

References

Taylor, S.L., Eckley, I.A., and Nunes, M.A. (2014) A Test of Stationarity for Textured Images. *Technometrics*, **56** (3), 291-301.

See Also

compareImages

Examples

```
## Not run:
 X1<-simTexture(128,K=1,imtype="S1")[[1]]
 X2<-simTexture(128,K=1,imtype="S1")[[1]]
 X3<-simTexture(128,K=1,imtype="S1",sd=1.6)[[1]]

 Xlist<-list(X1,X2,X3)

 Xlist.class<-countTextures(Xlist, bs=100)

## End(Not run)
```

---

cropimage *Crops a rectangular image to a specified (square) dimension*

Description

If the input image is not of dimension $2^n \times 2^n$, for some n, then the image is cropped to an optionally specified size.

Usage

```
cropimage(image, newsize = NULL, pos = "e")
```

Arguments

image	The image you wish to crop.
newsize	An optional dimension (smaller than the original image dimension), to which the image should be cropped.
pos	The position of the subimage to take when cropping an image. See the documentation for mix2images for more details.

Details

As we often wish to work with images whose dimensions are some power of 2, this function will determine whether the image is of an appropriate size and if not it will crop the image so that it is. The optional pos argument specifies the position of the cropped subimage to be returned; for example pos="e" specifies the central region. See mix2images for more details on the positioning argument.

Value

subim	A square image with dimension $2^n \times 2^n$, where n is either newsize or the largest feasible dyadic power smaller than the original image dimensions.

Author(s)

Matt Nunes

See Also

mix2images

Examples

```
#
#Create an image with dimensions not a power of two
#
testimage <- matrix(rnorm(300^2),nrow=300,ncol=300)
#
#Crop the image
#
Newimage <- cropimage(testimage)
#
# Check new dimension size.
#
dim(Newimage)
#
```

---

getpval *Computes a p-value for the output of the test for stationarity.*

Description

Computes and returns a p-value from the output of the bootstrap test for stationarity.

Usage

```
getpval(statvec, verbose = TRUE)
```

Arguments

statvec	A vector of test statistics, such as that given by TOS2D. The first value must be the value of the test statistic for the original image.
verbose	If TRUE then the p-value is printed and a sentance declaring "stationary" or "not stationary" is printed.

Value

p	The p-value of the test.

Author(s)

Sarah L. Taylor

References

Taylor, S.L., Eckley, I.A., and Nunes, M.A. (2014) A Test of Stationarity for Textured Images. *Technometrics*, **56** (3), 291-301.

See Also

TOS2D

Examples

```
#Generate a stationary image

testimage <- matrix(rnorm(64*64), nrow=64, ncol=64)

# Run test of stationarity

## Not run: TestofStat<-TOS2D(testimage)

# Obtain p-value

getpval(TestofStat$samples)

## End(Not run)
```

---

imageQT *Performs an image quadtree decomposition.*

Description

The quadtree decomposition is achieved by recursively splitting subimages into regions of stationarity.

Usage

```
imageQT(image, test = TOS2D, minsize = 64,alpha=0.05, ...)
```

Arguments

image	An image to be decomposed.
test	A function for assessing regions of spatial homogeneity, for example TOS2D.
minsize	The smallest region to test for homogeneity.
alpha	The significance level for the homogeneity test test.
...	Any other (optional) arguments to TOS2D.

Details

This function works by assessing an image for homogeneity. If it is not homogeneous, the image is split into its four subquadrants. Each of these is then tested for homogeneity. The heterogeneous subimages are then again subdivided and tested again. This procedure is repeated until either all subimages are deemed stationary or the minimum testing size minsize is reached.

Value

An object of class imageQT with the following components:

data.name	The image analysed.
indl	The index representation of the nonstationary images in the quadtree decomposition.
resl	The results of the stationarity testing (from binfun) during the quadtree decomposition. The results giving 0 match those contained in the indl component and the results giving 1 match those contained in the indS component.
imsize	The original image dimension.
imS	The stationary subimages in the quadtree decomposition.
indS	The index representation of the stationary images in the quadtree decomposition.
minsize	The minimum testing region used during the quadtree decomposition.

Author(s)

Sarah L. Taylor and Matt Nunes

References

Sonka, M., Boyle, R., and Hlavic, V. (1999) Image processing, analysis and machine vision. 2nd Edition, PWS Publishing.
Taylor, S.L., Eckley, I.A., and Nunes, M.A. (2014) A Test of Stationarity for Textured Images. *Technometrics*, **56** (3), 291-301.

See Also

plot.imageQT, TOS2D

Examples

```
# generate an image:
X<-simTexture(128,K=1,imtype="NS2")[[1]]

## Not run: XQT<-imageQT(X,binfun=TOS2D.bin)
```

---

lincurve *A linear function between two constant values.*

Description

A function with which to generate nonstationary covariance structure.

Usage

```
lincurve(x, start = 1, end = 2, a = 0.25)
```

Arguments

x	a sequence of x-values.
start	a starting value for the linear function.
end	an ending value for the linear function.
a	a proportion of the x-values for the linear part of the function.

Value

y	the y-values associated to the linear function.

Author(s)

Matt Nunes

See Also

scurve, simTexture

Examples

```
x<-seq(0,1,length=128)

y<-lincurve(x,start=1,end=2,a=.25)

plot(x,y,type="l")
```

`mix2images` *Insert one image into another.*

Description

Image A is re-sized to a specified proportion of Image B, then inserted into Image B at a given position.

Usage

```
mix2images(imageA, imageB, prop = 0.25, pos = "e")
```

Arguments

`imageA`	The first image which is resized and placed inside the second image.
`imageB`	The second image, into which the first is placed.
`prop`	The proportion of Image B to be taken up by Image A.
`pos`	The exact position of image A in image B. Possible options are "a", "b", "c", "d", "e" which corresponds to (a) top-right, (b) bottom-right, (c) top-left, (d) bottom-left and (e) centred. A more exact location may be specified by inputting `pos=c(x,y)`, which represents the position in pixels from the top-left of the image (i.e. `c(x,y)` puts Image A `x` pixels down and `y` pixels across from the top-left corner of Image B.)

Details

This function first of all crops Image A to be a given proportion of Image B and then inserts it into image B at the specified location. If image B is too small for the size of image A required then the whole of image A is placed in image B. Both must be dyadic in length and square images.

Value

`ImageB`	A matrix with the specified values selected exchanged to those of Image A.

Author(s)

Sarah L. Taylor

References

Taylor, S.L., Eckley, I.A., and Nunes, M.A. (2014) A Test of Stationarity for Textured Images. *Technometrics*, **56** (3), 291-301.

Examples

```
# Generate 2 images.
#
ImageA <- matrix(rnorm(256^2), nrow=256, ncol=256)
ImageB <- matrix(rnorm(256^2, sd=2.8), nrow=256, ncol=256)
#
# Insert Image A into Image B at a proportion of 0.25
#
MixImaImb <- mix2images(ImageA, ImageB, prop=0.25, pos="e")
#
```

---

plot.imageQT *A plot function for quadtree decompositions.*

Description

A plot function for quadtree decompositions.

Usage

```
## S3 method for class 'imageQT'
plot(x, cires, unclassval = 0, class = FALSE, QT = TRUE,
return = FALSE, qtl = 1, ...)
```

Arguments

x	A quadtree decomposition object, such as output from imageQT.
cires	Results of countTextures for the classification of subimages produced by the quadtree decomposition.
unclassval	A value for unclassified values in a quadtree decomposition.
class	A boolean value indicating whether to plot the results from countTextures.
QT	A boolean value indicating whether to plot the quadtree decomposition.
return	A boolean value indicating whether to return the matrix associated to the plotted image.
qtl	Colour specification for the lines drawn in the image segmentation (for QT=TRUE).
...	Any other optional arguments to image.

Details

The function plots the chosen quadtree decomposition, and optionally the textured region classification output from `countTextures`. If the classification output is plotted (`class=TRUE`), each textured region is uniquely coloured according to its texture group.

Value

`immat`	the matrix associated to the plotted image.

Author(s)

Sarah L. Taylor and Matt Nunes

See Also

`imageQT`, `countTextures`

Examples

```
## Not run:

X<-simTexture(256,K=1,imtype="NS2")[[1]]

XQT<-imageQT(X, bs=100, smooth=FALSE)

XCI <- Tex(XQT$imS, bs=100, smooth=FALSE)

plot(XQT, XCI, QT=T, class=T)

## End(Not run)
```

---

plotmtx *Image manipulation*

Description

A function which rearranges image content for nice plotting.

Usage

```
plotmtx(m)
```

Arguments

`m`	An image (matrix) for converting so that it can be plotted.

Details

Due to the input and plotting output of the R base function image, this function reorders the pixels within an image such that, when used, the image function produces a plot of a image (matrix) "as is".

Value

m.out	The manipulated image corresponding to the input image.

Author(s)

Matt Nunes

See Also

image

Examples

```
Im<-simTexture(n=256,type="NS4",K=1)[[1]]

image(plotmtx(Im))
```

---

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

Description

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

Usage

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

Arguments

x	An object of class 'imageQT' about which you wish to print information.
...	This argument actually does nothing in this function!

Author(s)

Matt Nunes

See Also

imageQT

Examples

```
## Not run:
#
# Generate a imageQT object for a HaarMontage realisation
#
X<-simTexture(n=256,K=1,imtype="S1")[[1]]

Xres <- imageQT(X)

print(Xres)

## End(Not run)
```

`print.TOS2D` *Print out information about a TOS2D object in readable form.*

Description

This function prints out information about a TOS2D object in a nice human-readable form.

Usage

```
## S3 method for class 'TOS2D'
print(x, ...)
## S3 method for class 'TOS2D'
summary(object, ...)
```

Arguments

`x, object`	An object of class 'TOS2D' about which you wish to print information.
`...`	This argument actually does nothing in this function!

Author(s)

Matt Nunes

See Also

TOS2D

Examples

```
## Not run:
#
# Generate a TOS2D object for a HaarMontage realisation
#
X<-simTexture(n=256,K=1,imtype="S1")[[1]]
```

```
Xres <- TOS2D(X)

summary(Xres)

## End(Not run)
```

---

scurve *An S curve function between two constant values.*

Description

A function with which to generate nonstationary covariance structure.

Usage

```
scurve(x, a = 1, start = 1, end = 2)
```

Arguments

x	a sequence of x-values.
a	The coefficient of slope of the curve.
start	a starting value for the curve
end	an ending value for the curve

Value

y	the function values associated to x depicting an S-curve.

Author(s)

Matt Nunes

See Also

lincurve, simTexture

Examples

```
x<-seq(0,1,length=100)

y<-scurve(x,.4,1,2)

plot(x,y,type="l")
```

simTexture *simTextureulation function for LS2W processes.*

Description

This function will generate images of a specified type

Usage

```
simTexture(n = 256, sd = 1, K = 150, imtype = "S1", ...)
```

Arguments

n	The dimension of the image to be generated.
sd	The standard deviation of the increments of the LS2W process to be generated.
K	The number of images to generate.
imtype	The type of image(s) to create. Must be one of "S1","S2","S3","S4", "NS1","NS2", "NS3", "NS4","NS5","NS6", "NS7". See details for descriptions of the processes.
...	Any other optional arguments needed for the image generation (see details).

Details

Several different processes can be generated with the simTexture function. The stationary processes are: a random normal process of specified standard deviation, sd (S1); a spatial moving average process with parameter rho (S2); an isotropic random field with a Matern covariance with shape parameter nu (S3) and a *diagonal Haar moving average* process of a specified order order and standard deviation sd (S4) (see the Haar2MA.diag function in the LS2W package for more details).
The nonstationary processes are: a random field with unit standard deviation on the first half-plane, concatenated with a random normal half-plane of standard deviation sd (NS1); a white noise half-plane concatenated with a Matern stationary process (NS2); a *Haar Montage* of specified standard deviation sd (NS3) (see the LS2W HaarMontage function for more details); a process with a slowly-varying covariance structure (NS4); a white noise process with a central subregion of random Normal deviates with non-unit standard deviation sd (NS5); a white noise process with a subregion of random Normal deviates with non-unit standard deviation in the middle section of the top left quadrant sd (NS6); the final process is similar to NS5, except that there is an additional texture in a subregion of the image. In other words, the image is a montage of three two-dimensional Normal processes with differing standard deviations. The base texture is again of unit variance, whereas the other two textures have standard deviations sd and sd2 (NS7).

The other optional arguments for simTexture are as follows:
type - the type of neighbourhood dependence for the random field, either "queen" or "rook" (see the cell2nb function documentation in the spdep package for more details).
rho - moving average parameter for the process S2.
nu - shape parameter for the Matern covariance for process S3.
order - Haar moving average order for S4.

fn - scurve or lincurve for NS4.
start - start value for NS4 (passed into scurve or lincurve).
end - end value for NS4 (passed into scurve or lincurve).
a - "gradient" for NS4 (passed into scurve or lincurve).
prop - proportion of inserted subimage for NS5, NS6 and the first subimage (NS7).
sd2 - standard deviation of second inserted subimage for NS7.
prop2 - proportion of second inserted subimage for NS7.
pos1 - position of first inserted subimage for NS7.
pos2 - position of second inserted subimage for NS7.

Value

images	A list of length K, with each list entry being an image of dimension n x n with the chosen spectral structure.

Warning

Generating lots of images of high dimension may take a long time!

Author(s)

Sarah L. Taylor and Matt Nunes

References

Matern, B. (1960) Spatial variation. Stochastic models and their application to some problems in forest surveys and other sampling investigations *Meddleanden fran statens Skogsforskningsinstitut* **49** (5).
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.
Taylor, S.L., Eckley, I.A., and Nunes, M.A. (2014) A Test of Stationarity for Textured Images. *Technometrics*, **56** (3), 291-301.

See Also

HaarMontage

Examples

```
X1 <- simTexture(128,K=1,imtype="S4",order=3)
X2 <- simTexture(128,K=1,imtype="NS4",fn=lincurve,a=.25,start=1,end=2)
X3 <- simTexture(128,K=1,imtype="NS5",sd=1.6,prop=.25)
X4 <- simTexture(128,K=1,imtype="NS6",sd=1.6,prop=.25)
X5 <- simTexture(128,K=1,imtype="NS7",sd=1.6,prop=.25,sd2=2.8, prop2=0.25,
pos1=c(10,10),pos2="e")

# try plotting the images:

## Not run: image(plotmtx(X1[[1]]))
```

TOS2D *Perform bootstrap stationarity test for images.*

Description

For a given image this function performs bootstrapping to test the hypothesis that the image is stationary.

Usage

```
TOS2D(image, detrend = FALSE, nsamples = 100, theTS = avespecvar, verbose = TRUE,...)
```

Arguments

image	The image you want to analyse.
detrend	This specifies whether to use Tukey's median polish to remove the image trend.
nsamples	Number of bootstrap simulations to carry out.
theTS	Specifies the particular test statistic to be used. This function should measure the departure from constancy of the wavelet spectrum.
verbose	If TRUE informative messages are printed.
...	Any other arguments supplied to the LS2W function cddews.

Details

This function first of all crops the image (if necessary) to have dyadic dimensions. The test statistic (theTS), which should be based upon the local wavelet spectrum, is calculated for this original image and the local wavelet spectrum under the null hypothesis is calculated, so as to be able to simulate realisations under the null hypothesis. nsamples images are simulated and test statistic is found for each. The function returns all the test statistic values which may be passed to getpval in order to find a p-value for the test. For full details on this testing procedure see Taylor et al. (2014).

Value

A list with the following components:

data.name	The name of the image analysed.
samples	A vector of length nsamples+1. The first entry is the value of the test statistic computed on the original image while the remaining entries are test statistic values for the simulated images.
statistic	The name of the test statistic used.
p.value	The bootstrap p-value for the test.

Author(s)

Sarah L. Taylor

References

Taylor, S.L., Eckley, I.A., and Nunes, M.A. (2014) A Test of Stationarity for Textured Images. *Technometrics*, **56** (3), 291-301.

See Also

avespecvar, getpval

Examples

```
# Generate a stationary image
#
testimage <- matrix(rnorm(64*64), nrow=64, ncol=64)
#
#Run test of stationarity

## Not run: TestofStat<-TOS2D(testimage)
```

Index

∗Topic **datagen**
simTexture, 17
∗Topic **manip**
compareImages, 4
cropimage, 6
imageQT, 8
lincurve, 10
mix2images, 11
plotmtx, 13
scurve, 16
∗Topic **methods**
countTextures, 5
∗Topic **nonparametric**
avespecvar, 3
∗Topic **package**
LS2Wstat-package, 2
∗Topic **plot**
plot.imageQT, 12
∗Topic **print**
print.imageQT, 14
print.TOS2D, 15
∗Topic **statistic**
getpval, 7
TOS2D, 19

avespecvar, 3, *20*

cell2nb, *17*
compareImages, 4, *6*
countTextures, *4*, 5, *13*
cropimage, 6

getpval, 7, *19, 20*

Haar2MA.diag, *17*
HaarMontage, *17, 18*

image, *12*, *14*
imageQT, 8, *13, 14*

lincurve, 10, *16*
LS2Wstat *(*LS2Wstat-package*)*, 2
LS2Wstat-package, 2

mix2images, *6, 7*, 11

plot.imageQT, *10*, 12
plotmtx, 13
print.imageQT, 14
print.TOS2D, 15

scurve, *10*, 16
simTexture, *10*, *16*, 17
summary.TOS2D *(*print.TOS2D*)*, 15

TOS2D, *3, 4*, *8*, *10*, *15*, 19