Package ‘spc4sts’

December 9, 2019

Type Package

Title Statistical Process Control for Stochastic Textured Surfaces

Version 0.5.2

Author Anh Tuan Bui [aut, cre] and Daniel W. Apley [ths]

Maintainer Anh Bui <atbui@u.northwestern.edu>

Depends LS2Wstat, rpart, gridExtra, parallel

Description Provides statistical process control tools for stochastic
textured surfaces. The current version supports the following tools:
(1) generic modeling of stochastic textured surfaces.
(2) local defect monitoring and diagnostics in stochastic
textured surfaces, which was proposed by Bui and Apley (2018a)
<doi:10.1080/00401706.2017.1302362>.
(3) global change monitoring in the nature of stochastic
textured surfaces, which was proposed by Bui and Apley (2018b)
<doi:10.1080/00224065.2018.1507559>.
(4) computation of dissimilarity matrix of stochastic textured
surface images, which was proposed by Bui and Apley (2019b)
<doi:10.1016/j.csda.2019.01.019>.

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2019-12-09 20:50:02 UTC

R topics documented:

`spc4sts-package` *Statistical Process Control for Stochastic Textured Surfaces*

Description

Provides statistical process control tools for stochastic textured surfaces. Some tools in the package can also be used in non-SPC contexts that deal with stochastic textured surface images (see Section Details below). The current version supports the following tools:

(1) generic modeling of stochastic textured surfaces (Bui and Apley 2018a, 2018b)

(2) local defect monitoring and diagnostics in stochastic textured surfaces (Bui and Apley 2018a)

(3) global change monitoring in the nature of stochastic textured surfaces (Bui and Apley 2018b)

(4) computation of dissimilarity matrix of stochastic textured surface images (Bui and Apley 2019b).

Details

Stochastic textured surface (STS) is the term used in Bui and Apley (2018a) to refer to a class of measurement data of material surfaces that have no distinct features other than stochastic characteristics that vary randomly. A few examples of STS data include microscopy images of material microstructure samples and images of lumber surfaces, engineered stone countertops, ceramic capacitor surfaces, and textile materials that show weave patterns (Bui and Apley 2017a, 2017b, 2019a).

For STS data, even of the same nature, each image is completely different from others on a pixel-by-pixel basis. In addition, it is not straightforward to align, transform, or warp them into a common "gold standard" image, as a basis for comparison. The existence of a gold standard is a fundamental requirement for most of the statistical process control (SPC) literature for profile and multivariate data that are not STSs. An example of a gold standard in non-STS data is an image of a circuit assembly with perfectly positioned chips, to which images of actual assemblies with chips positioned inaccurately are compared for SPC quality control purposes. Existing SPC methods that may be applicable to STS data rely on some form of feature extraction from the STS images (e.g., a specific frequency component from a spectral analysis of the image), but they are problem specific because prior knowledge of abnormal behavior is needed to define suitable features.

The **spc4sts** (Statistical Process Control for Stochastic Textured Surfaces) package is the first implementation of the methods in Bui and Apley (2018a), Bui and Apley (2018b), and Bui and Apley (2019), and serves as the first off-the-shelf toolkit for performing SPC for general STS data without prior knowledge of abnormal behavior. The package is applicable to a wide range of materials as mentioned above, including random heterogeneous materials.

Some tools in the package can also be used in non-SPC contexts that deal with STS images. First, the STS modeling tool can be used in STS image characterization and reconstruction (e.g., powder materials micrograph characterization and materials microstructure image reconstruction). Second, the surface dissimilarity calculation tool can be used for STS image classification, clustering, and outlier detection. Some examples are medical microscopy image classification, cancer tissue image clustering, and outlying mammalian cell image detection.

Brief descriptions of the main functions of the package are provided below:

`surfacemodel()` builds a supervised learning model (a regression tree in this version) to characterize the statistical behavior of the given stochastic textured surface data sample.

`monitoringStat()` computes the monitoring statistic(s) (for local defects and/or global changes) for the given image, based on the model built from `surfacemodel()`.

`climit()` establishes the control limits (for local defects and/or global changes) at the given false alarm rates based on the monitoring statistics (for local defects and/or global changes) computed for a set of in-control images (i.e., without local defects or global changes) using `monitoringStat()`. It also constructs the diagnostic thresholds (for diagnosing local defects) to be used for `diagnoseLD()`.

`diagnoseLD()` produces a binary diagnostic image that highlights local defects (if any) in the given stochastic textured surface image.

`disMat()`: computes KL and/or AKL dissimilarity matrices for the given stochastic textured surface images.

Author(s)

Anh Tuan Bui and Daniel W. Apley

Maintainer: Anh Tuan Bui <atbui@u.northwestern.edu>

References

Bui, A.T., and Apley, D.W. (2017a), textile: Textile Images, R package version 0.1.2. https://cran.r-project.org/package=textile.

Bui, A.T., and Apley, D.W. (2017b), Textile Images 2, Mendeley Data, v1. http://dx.doi.org/10.17632/wy3pndgpcx.1.

Bui, A.T. and Apley., D.W. (2018a) "A Monitoring and Diagnostic Approach for Stochastic Textured Surfaces", Technometrics, 60, 1-13.

Bui, A.T. and Apley, D.W. (2018b) "Monitoring for changes in the nature of stochastic textured surfaces", Journal of Quality Technology, 50, 363-378.

Bui, A.T. and Apley D.W. (2019a) "Textile Image 3", figshare, http://dx.doi.org/10.6084/m9.figshare.7619351.v1.

Bui, A.T. and Apley, D.W. (2019b) "An exploratory analysis approach for understanding variation in stochastic textured surfaces", Computational Statistics & Data Analysis, 137, 33-50.

Examples

```
#
# See the examples in the help pages for the main functions mentioned above.
#
```

---

ad *One-Sample Anderson-Darling Statistic*

Description

Computes the one-sample Anderson-Darling (AD) statistic.

Usage

```
ad(r, P)
```

Arguments

`r`	the given vector/matrix of observations
`P`	the vector/matrix containing the values of a (reference) cumulative distribution function evaluated at the values in `r`.

Value

The AD statistic.

Author(s)

Anh Bui

References

Bui, A.T. and Apley., D.W. (2018a) "A Monitoring and Diagnostic Approach for Stochastic Textured Surfaces", Technometrics, 60, 1-13.

See Also

`exptailecdf`,`sms`,`bp`

Examples

```
img <- matrix(rnorm(100), 10, 10)
ad(img, pnorm(img))
```

---

bp	*Box-Pierce-Type Statistic*

Description

Compute a Box-Pierce-type (BP) statistic for pixels in a given image. `bp2()` cannot be used for pixels with the boundary problem, but is more efficient than `bp()` for other pixels.

Usage

```
bp(img, i1, i2, w, K)
bp2(img, i1, i2, w , K)
```

Arguments

`img`	the given image
`i1`	the row index of the pixel to compute the BP statistic for.
`i2`	the column index of the pixel to compute the BP statistic for.
`w`	the dimension of the spatial (square) moving window of the BP statistic. Must be an odd number >= 3.
`K`	the weighted (kernel) matrix.

Value

The BP statistic.

Warning

For pixels with the boundary problem, `bp()` must be used.

Note

`bp()` is only used in `sms()` for pixels with the boundary problem. It is less efficient than `bp2()` for other pixels.

Author(s)

Anh Bui

References

Bui, A.T. and Apley., D.W. (2018a) "A Monitoring and Diagnostic Approach for Stochastic Textured Surfaces", Technometrics, 60, 1-13.

See Also

kerMat, spaCov, sms, ad

Examples

```
img <- matrix(rnorm(100),10,10)
w <- 3
K <- kerMat((w + 1)/2)
## for pixels with the boundary problem, e.g., Pixel (5,1),
# running bp2(img,5,1,w,K) will produce an error; instead, use bp() in this case:
bp(img,5,1,w,K)

## for pixels without the boundary problem, e.g., Pixel (5,5),
# both can be used, but bp2() is more efficient than bp()
bp2(img,5,5,w,K)
bp(img,5,5,w,K)
```

---

climit   *Control Limit and Diagnostic Threshold Construction*

Description

Establish control limits (for local defects and/or global changes) and diagnostic thresholds (for local defects) from the given Phase I images. `climit` is used for the first time. `climit2` can update the control limits and diagnostic thresholds given the output of `climit`. See Warning. To plot histograms of the Phase I monitoring statistics, use `plot.climit`.

Usage

```
climit(imgs, fa.rate, model, type, stat = c("ad", "bp"), w,
       nD = 10, no_cores = 1, verbose = FALSE)
climit2(cl, fa.rate, nD)
```

Arguments

`imgs`	a 3-dimensional array containing all Phase I in-control images.
`fa.rate`	the false alarm rate, which asserts the rate of in-control images that are falsely alarmed as out-of-control. This can be a vector, in which case several levels of the control limit are returned.
`model`	the object returned by `surfacemodel`.
`type`	for local defects, `type = 1`; for global changes, `type = 2`; for both, `type = 1:2`.
`stat`	for local defects only. The statistic used in the spatial moving statistics. Must be either `"ad"` (default) or `"bp"`.
`w`	for local defects only. The dimension of the spatial (square) moving window. Must be an odd number >= 3.

`nD`	for local defects only. The parameter to construct the diagnostic threshold. It is the average number of highlighted pixels in the diagnostic image for an in-control image.
`no_cores`	if > 1, parallely compute Phase I monitoring statistics using `no_cores` processors.
`verbose`	if `TRUE`, show the computing progress.
`cl`	the object returned by `climit` or `climit2`.

Value

An object of class `climit`. See `climit.object`.

Author(s)

Anh Bui

References

Bui, A.T. and Apley., D.W. (2018a) "A Monitoring and Diagnostic Approach for Stochastic Textured Surfaces", Technometrics, 60, 1-13.

Bui, A.T. and Apley, D.W. (2018b) "Monitoring for changes in the nature of stochastic textured surfaces", Journal of Quality Technology, 50, 363-378.

See Also

`monitoringStat`,`diagnoseLD`

Examples

```
## build the in-control model
img <- sarGen(m = 50, n = 50, border = 50) # training image
model <- surfacemodel(img, nb = 1, keep.residuals = TRUE)

## after that, generate Phase I images
imgs <- array(0, c(50,50,3))
for (j in 1:dim(imgs)[3])
  imgs[,,j] <- sarGen(phi1 = .6, phi2 = .35, m = 50, n = 50, border = 50)

## establish control limits and diagnostic thresholds
# construct control limits (for both local defects and global changes)
# and diagnostic thresholds (for local defects) for the first time
cl <- climit(imgs, fa.rate = .05, model, type = 1:2, stat = "ad", w = 5, nD = 50)
cl
# update new control limit and diagnostic threshold
cl2 <- climit2(cl, fa.rate = .01, nD = 5)
# plots histograms of Phase I monitoring statistics
plot(cl2)

## after that, monitor a Phase II image as follows:
# create a new image with a local defect
```

```
img2 <- sarGen(phi1 = .6, phi2 = .35, m = 50, n = 50, border = 50) # simulate a new image
img3 <- imposeDefect(img2)$img # add a local defect to this image
ms3 <- monitoringStat(img = img3, model = model, cl = cl2) # computing monitoring statistics
# now create a new image with parameters reduced by 5% (representing a global change)
img4 <- sarGen(phi1 = .6*.95, phi2 = .35*.95, m = 50, n = 50, border = 50)
ms4 <- monitoringStat(img = img4, model = model, cl = cl2) # computing monitoring statistics

## diagnose for local defect regions in img3
bimg <- diagnoseLD(ms3, dth = 9, plot.it = FALSE) # use climit() to find dth
par(mfcol = c(1, 2))
par(mar = c(2, 0.5, 1, 0.5))
image(xaxt = 'n', yaxt = 'n', as.matrix(t(apply(img3 , 2, rev))),
      col = gray((0:32)/32), xlab = '', ylab = '', asp = 1, bty = 'n')
image(xaxt = 'n', yaxt = 'n', as.matrix(t(apply(bimg , 2, rev))),
      col = gray(c(1, .5)), xlab = '', ylab = '', asp = 1, bty = 'n')

#
# NOTE: The above example is just for quick illustration. To obtain a good
# control limit, the training image should be representative (e.g., set
# m = 250, n = 250, and border = 200). The number of Phase I images also
# needs to be large (e.g., 100 images or more).
#
# For real images in a textile application, use the R data package "textile".
#
```

---

climit.object *Control Limit and Diagnostic Threshold Construction Object*

Description

Tthe object returned by `climit` or `climit2`.

Value

`type`	the `type` argument of `climit`.
`fa.rate`	the `fa.rate` argument of `climit`.
`localStat`	contains values for local defect monitoring: `nDmaxSms` is a vector that stores the `(nD*N + 1)` largest SMS values computed for all `N` Phase I images. `PIstats` is a vector that stores the monitoring statistics computed for all the Phase I images. `diagnostic.threshold` is a scalar/vector that stores the established diagnostic threshold(s). `stat` and `w` are the `stat` and `w` arguments of the `climit` function. `control.limit` is a scalar/vector that stores the established control limit(s).
`globalStat`	contains values for global change monitoring: `PIstats` is a vector that stores the monitoring statistics computed for all the Phase I images. `xval` is the `xval` argument of the `climit` function. `control.limit.trans_chi2` and `control.limit.ecdf` are a scalar/vector that stores the established control limit(s) using the parametric approximation of the empirical distributions and the empirical distributions directly, respectively. The former is recommended when the number of Phase I images is not enough for using directly the empirical distribution.

Author(s)

Anh Bui

References

Bui, A.T. and Apley., D.W. (2018a) "A Monitoring and Diagnostic Approach for Stochastic Textured Surfaces", Technometrics, 60, 1-13.

Bui, A.T. and Apley, D.W. (2018b) "Monitoring for changes in the nature of stochastic textured surfaces", Journal of Quality Technology, 50, 363-378.

See Also

`climit`

`dataPrep` *Neighborhood Data Preparation*

Description

Prepares a neighborhood data from a given image, using the left-to-right then top-to-bottom raster scan order (see Bui and Apley 2018a).

Usage

```
dataPrep(img, nb, vars = NULL, subsample = 1)
```

Arguments

`img`	the given image in the matrix format.
`nb`	the size of the neighborhood. It must be a 1-length or 3-length vector of positive integer(s). If the former, it is the same with a 3-length vector with the same elements.
`vars`	names of variables to be selected in the neighborhood data.
`subsample`	the portion of data rows be returned. It takes values in (0, 1]. If `subsample = 1`, all data rows will be returned, and if `subsample = .5`, only roughly a half will be returned.

Value

A dataframe with column names "V1", "V2", "V3", ... The first column "V1" contains the response pixel, whereas the other columns contain pixels in the neighborhood (with size `nb`) of the response pixel.

Note

Only rows without missing values (corresponding to pixels with full neighborhood) are returned.

Author(s)

Anh Bui

References

Bui, A.T. and Apley., D.W. (2018a) "A Monitoring and Diagnostic Approach for Stochastic Textured Surfaces", Technometrics, 60, 1-13.

See Also

surfacemodel,monitoringStat

Examples

```
## construct a neighborhood data for an unrealistically small mock image (7x9 pixels).
mock.img <- matrix(sample(0:255, 63, replace = TRUE), 7, 9)
mock.img
dataPrep(img = mock.img, nb = 2) # the same with nb = c(2, 2, 2)

## select only columns "V2", "V5", and "V13" in the output
dataPrep(img = mock.img, nb = 2, vars = c("V2", "V5", "V13"))

## return only a half number of rows
dataPrep(img = mock.img, nb = 2, subsample = .5)
```

`diagnoseLD` *Diagnose Local Defects on Stochastic Textured Surfaces*

Description

Produces a binary diagnostic image of a given stochastic textured surface image based on its spatial moving statistics.

Usage

```
diagnoseLD(ms, dth, plot.it = TRUE)
```

Arguments

`ms`	the object return by `monitoringStat()`
`dth`	the diagnostic threshold
`plot.it`	plots the binary diagnositc image if set to `TRUE`

Value

The binary diagnostic image in the matrix format.

Author(s)

Anh Bui

References

Bui, A.T. and Apley., D.W. (2018a) "A Monitoring and Diagnostic Approach for Stochastic Textured Surfaces", Technometrics, 60, 1-13.

See Also

`monitoringStat`,`climit`

Examples

```
## ## see the examples in the help file of climit()
?climit
```

`disMat` *Pairwise Dissimilarity Matrix of Stochastic Textured Surfaces*

Description

Compute KL and ALK dissimiarlity matrices for the given stochastic textured surface images.

Usage

```
disMat(imgs, nb, cp=1e-3, subsample = c(1, .5),
                     standardize = TRUE, keep.fits = FALSE, verbose=FALSE)
```

Arguments

`imgs`	a 3-dimensional array containing all images.
`nb`	the size of the neighborhood. It must be a 1-length or 3-length vector of positive integer(s). If the former, it is the same with a 3-length vector with the same elements.
`cp`	the minimal value for the `rpart` complexity models. The smaller `cp` is, the more complex the `rpart` models are fit.
`subsample`	the portion of pixels in the given image `img` to be used when fitting models (the first component) and computing dissimilarities (the second component). It takes values in (0, 1] (e.g., `subsample = c(1,.5)` means that the whole image is used when fitting models, and roughly a half of that is used when compute dissimilarities).
`standardize`	if `TRUE`, standardize the given image `img <-(img -mean(img))/sd(img)`. This reduces the effect of different lighting conditions when images are taken.
`keep.fits`	if `TRUE`, save all the fitted models in the "fits.Rdata" under the wokring directory.
`verbose`	if set to `TRUE`, output some computational time information.

Value

the KL and AKL dissimilarity matrices.

Author(s)

Anh Bui

References

Bui, A.T. and Apley, D.W. (2019b) "An exploratory analysis approach for understanding variation in stochastic textured surfaces", Computational Statistics & Data Analysis, 137, 33-50.

Examples

```
## generate images: the first two are similar, the third is different with the other two
phi1 <- c(.6, .6, .5)
phi2 <- c(.35, .35, .3)
imgs <- array(0, c(100,100,3))
for (j in 1:dim(imgs)[3])
  imgs[,,j] <- sarGen(phi1 = phi1[j], phi2 = phi2[j], m = 100, n = 100, border = 50)
## compute KL and AKL dissimilarity matrices
disMat(imgs = imgs, nb = 1)
```

`exptailecdf` *Empirical Cumulative Distribution Function with Exponential Tail Approximation*

Description

Computes the empirical cumulative distribution funciton (ecdf) of a given vector of observations, and approximates the tails of the ecdf with exponential curves.

Usage

```
exptailecdf(x, N = max(2, 0.002 * length(x)), m = min(N, 5))
```

Arguments

x	the given vector of observations
N	the number of observations at each tail of the ecdf used for estimating the exponential curves.
m	the mth observation from each extreme of the ecdf is the starting point to use the estimated exponential curves.

Details

An ecdf has a probability of 0 or 1 for any new observation that lies beyond the range of the data of the cedf. This is a problem when using the ecdf as the reference cdf for the one-sample Anderson-Darling (AD) statistic because the computational formula of the AD statistic is infinite with such probabilities. The ecdf with exponential tail approximation replaces the tails of the ecdf with exponential curves, which extend to infinity, to solve this problem. The exponential curves are estimated using the observations at the tails of the ecdf. See Bui and Apley (2018a) for more details.

Value

An object of class `exptailecdf`. See `exptailecdf.object`

Author(s)

Anh Bui

References

Bui, A.T. and Apley., D.W. (2018a) "A Monitoring and Diagnostic Approach for Stochastic Textured Surfaces", Technometrics, 60, 1-13.

See Also

`exptailecdf.object`,`pexptailecdf`,`ecdf`,`ad`

Examples

```
r <- rnorm(1000)
Fr <- exptailecdf(r)
```

`exptailecdf.object` *Empirical Cumulative Distribution Function with Exponential Tail Approximation Object*

Description

The object returned by `exptailecdf`.

Value

`ecdf`	the ecdf returned by the `stats::ecdf()`
`lambda`	the parameters estimated for the exponential curves. `lambda[1]` corresponds to the left tail.
`joint`	where the ecdf started to be replaced by the exponential curves. `joint[1]` corresponds to the left tail.

Author(s)

Anh Bui

See Also

`exptailecdf`

`imposeDefect` *Superimpose A Local Defect*

Description

Superimposes a local defect (a 2D stochastic AR(1) image from `sarGen`) on a given image.

Usage

```
imposeDefect(img, loc = NULL, a = 4, b = 10, eps = 0.05, phi1 = 0, phi2 = 0, sigma = 0.01)
```

Arguments

`img`	the image to be superimposed a defect.
`loc`	the location of the defect in the generated image.
`a`	`2*a + 1` is the vertical axis length of the ellipsoidal defect.
`b`	`2*b + 1` is the vertical axis length of the ellipsoidal defect.
`eps`	controls the curvature of the ellipsoidal defect.
`phi1`	the parameter `phi1` of the defect.
`phi2`	the parameter `phi2` of the defect.
`sigma`	the parameter `sigma` of the defect.

Details

The defect is generated using `sarGen`.

Value

A list of the following:

`img`	the generated image in the matrix format.
`defect.info`	the information of the defects.

Author(s)

Anh Bui

References

Bui, A.T. and Apley., D.W. (2018a) "A Monitoring and Diagnostic Approach for Stochastic Textured Surfaces", Technometrics, 60, 1-13.

Examples

```
## generate an image without defects
img <- sarGen(m = 100, n = 100, border = 50)
image(img,col=gray(c(0:32)/32))

## superimpose a defect
img2 <- imposeDefect(img)
image(img2$img,col=gray(c(0:32)/32))
```

`kerMat` *Epanechnikov quadratic kernel matrix*

Description

Computes the Epanechnikov quadratic kernel in 2-D, and returns the positive kernel values.

Usage

```
kerMat(p)
```

Arguments

`p`	the bandwidth parameter

Value

A matrix containing all the positive kernel values

Author(s)

Anh Bui

References

Bui, A.T. and Apley., D.W. (2018a) "A Monitoring and Diagnostic Approach for Stochastic Textured Surfaces", Technometrics, 60, 1-13.

See Also

`bp`

Examples

```
kerMat(5)
```

`mbChange` *Matchbox Change*

Description

Modifies a given image to have a matchbox change.

Usage

```
mbChange(img, alpha = 1)
```

Arguments

`img`	the image to be matchboxed
`alpha`	the amount of matchboxing

Details

Each column `i` of `img` is modified as follows: `img[2:nrow(img),i] <-(1 -alpha*(i-1)/ncol(img))*img[2:nrow(img),`
`+ alpha*(i-1)/ncol(img)*img[1:(nrow(img)-1),i]`

Value

The matchboxed image in the matrix format.

Author(s)

Anh Bui

References

Bui, A.T. and Apley, D.W. (2018b) "Monitoring for changes in the nature of stochastic textured surfaces", Journal of Quality Technology, 50, 363-378.

`monitoringStat` *Monitoring Statistic for Stochastic Textured Surfaces*

Description

Computes monitoring statistic(s) for local defects (see Bui and Apley 2018a) and/or global changes (see Bui and Apley 2018b) for a given stochastic textured surface image.

Usage

```
monitoringStat(img, model, type, stat = c("ad", "bp"), w, cl = NULL, verbose = FALSE)
```

Arguments

`img`	the given image in the matrix format.
`model`	the object returned by `surfacemodel`
`type`	for local defects, `type = 1`; for global changes, `type = 2`; for both, `type = 1:2`
`stat`	for local defects only. The statistic used in the spatial moving statistics. Must be either `"ad"` (default) or `"bp"`.
`w`	for local defects only. The dimension of the spatial (square) moving window. Must be an odd number >= 3.
`cl`	the object returned by `climit` or `climit2`.
`verbose`	if set to `TRUE`, output monitoring outcome.

Value

A `monitoringStat` object containing the following components:

`sms`	a matrix of the SMS values computed for pixels in `img`
`stat`	the `stat` argument
`w`	the `w` argument
`localStat`	the monitoring statistic for local defects of `img`
`globalStat`	the monitoring statistic for global changes of `img`

Author(s)

Anh Bui

References

Bui, A.T. and Apley., D.W. (2018a) "A Monitoring and Diagnostic Approach for Stochastic Textured Surfaces", Technometrics, 60, 1-13.

Bui, A.T. and Apley, D.W. (2018b) "Monitoring for changes in the nature of stochastic textured surfaces", Journal of Quality Technology, 50, 363-378.

See Also

`surfacemodel`, `sms`, `dataPrep`

Examples

```
# run the example in the help file of climit()
?climit
```

`pexptailecdf` *Predictions from an Exptailecdf Object*

Description

Returns the values of the `exptailecdf` object at given observations.

Usage

```
pexptailecdf(Fx, y)
```

Arguments

`Fx`	the object of class `exptailecdf`, containing an ecdf with exponential tail approximation.
`y`	the given observations in the scalar/vector/matrix format.

Value

An object of the same type with `y` that stores the evaluations of the `exptailecdf` object at the given `y`.

Author(s)

Anh Bui

References

Bui, A.T. and Apley., D.W. (2018a) "A Monitoring and Diagnostic Approach for Stochastic Textured Surfaces", Technometrics, 60, 1-13.

See Also

`exptailecdf.object`,`exptailecdf`

Examples

```
r <- rnorm(1000)
Fr <- exptailecdf(r)

pexptailecdf(Fr, max(r) + .1)
pexptailecdf(Fr, c(min(r) - .1, max(r) + .1))
pexptailecdf(Fr, matrix(c(.8, .9, 1, 1.1), 2, 2))
```

`plotcc` *Control Chart Plotting*

Description

Plotting a control chart.

Usage

```
plotcc(statsII, CL, statsI = NULL)
```

Arguments

`statsII`	the Phase II monitoring statistics.
`CL`	the control limit of the control chart.
`statsI`	(some of) the Phase I monitoring statistics.

Author(s)

Anh Bui

`sarGen` *Stochastic Autoregressive Image Generator*

Description

Generates a 2D stochastic AR(1) image.

Usage

```
sarGen(phi1 = .6, phi2 = .35, sigma = .01, m = 250, n = 250, border = 200)
```

Arguments

`phi1`	the parameter `phi1` of the process.
`phi2`	the parameter `phi2` of the process.
`sigma`	the parameter `sigma` of the process.
`m`	the number of rows of the generated image.
`n`	the number of columns of the generated image.
`border`	the number of top rows/left columns to be cut off from the generated image. This helps reduce the effect of the starting condition.

Details

The pixel `y(i,j)` of the 2D AR(1) process satisfies: `y(i,j) = phi1*y(i-1,j) + phi2*y(i,j-1) + e(i,j)`, where `e(i,j)` follows a zero-mean Gaussian distribution with standard deviation of `sigma`. The process is then rescaled to [0, 255] to produce a greyscale image.

Value

The generated image in the matrix format.

Author(s)

Anh Bui

References

Bui, A.T. and Apley., D.W. (2018a) "A Monitoring and Diagnostic Approach for Stochastic Textured Surfaces", Technometrics, 60, 1-13.

See Also

`imposeDefect`

Examples

```
## generate an image without defects
img <- sarGen(m = 100, n = 100, border = 50)
image(img,col=gray(c(0:32)/32))
```

---

showNb *Show Neighborhood*

Description

Shows the neighborhood corresponding to the left-to-right then top-to-bottom raster scan order with additional information: variable names of the data frame returned by `dataPrep`, predictors used in the model returned by `surfacemodel`, or their percentage importance in the model (currently extracted from the `rpart` object). This function is useful for choosing a good neighborhood size and understanding relationship between pixels (e.g., periodicity).

Usage

```
showNb(model, what = c("neighborhood", "predictors", "importance"), plot.it = TRUE)
```

Arguments

`model`	either the object returned by `surfacemodel` or a positive vector of length 1 or 3 specifying the neighboorhood. If it is a vector, `what <-"neighborhood"`.
`what`	what to show in the neighorhood. `"neighborhood"` shows variable names of the data frame returned by `dataPrep`, `"predictors"` shows predictors used in the model returned by `surfacemodel`, and `"importance"` shows their percentage importance in the model.
`plot.it`	if `TRUE`, plot the neighborhood.

Value

A matrix that contains the information for the plot (using the `grid.table` function).

Author(s)

Anh Bui

References

Bui, A.T. and Apley., D.W. (2018a) "A Monitoring and Diagnostic Approach for Stochastic Textured Surfaces", Technometrics, 60, 1-13.

See Also

`dataPrep`,`surfacemodel`

Examples

```
## show the neighorhood with variables names of the data frame constructed by dataPrep()
img <- matrix(1:25, 5, 5) # an image of size 5x5 pixels
img
dataPrep(img, 2)
showNb(c(2, 2, 2)) # showNb(2) has the same effect

## show the neighorhood with predictors and their importance used in the model returned
## by surfacemodel()
img <- sarGen(m = 100, n = 100, border = 50) # training image
model <- surfacemodel(img, nb = 3)
showNb(model, "predictors") # show predictors
showNb(model, "importance") # show predictor percentage importance
```

sms *Spatial Moving Statistic*

Description

Computes the spatial moving statistics (SMS) for pixels in a given image.

Usage

```
sms(img, stat = c("ad", "bp"), w, Fr, gamma = (w + 1)/2)
```

Arguments

`img`	the image to compute the SMS for.
`stat`	the statistic used in the SMS. Must be either `"ad"` (default) or `"bp"`.
`w`	the dimension of the square moving window of the SMS. It must be an odd number >= 3.
`Fr`	the reference ecdf with exponential tail approximation (see `exptailecdf`). Only used when `stat = "ad"`.
`gamma`	the bandwidth parameter for `kerMat`. It must be a positive integer and is only used when `stat = "bp"`. The default value is recommended.

Value

A matrix containing the SMS values computed for the pixels in `img`.

Author(s)

Anh Bui

References

Bui, A.T. and Apley., D.W. (2018a) "A Monitoring and Diagnostic Approach for Stochastic Textured Surfaces", Technometrics, 60, 1-13.

See Also

`ad`,`bp`,`monitoringStat`

Examples

```
img <- matrix(rnorm(100),10,10)
ms.ad <- sms(img, "ad", 3, exptailecdf(rnorm(1000)))
ms.bp <- sms(img, "bp", 3)
```

spaCov *Spatial Weighted Covariance*

Description

Computes the spatial weighted covariance of a pair of pixels in a given image.

Usage

```
spaCov(img, i1, i2, j1, j2, K)
```

Arguments

`img`	the given image
`i1`	the row index of the first pixel in the pair.
`i2`	the column index of the first pixel in the pair.
`j1`	the row index of the second pixel in the pair.
`j2`	the column index of the second pixel in the pair.
`K`	the weighted matrix.

Value

The spatial weighted covariance.

Author(s)

Anh Bui

References

Bui, A.T. and Apley., D.W. (2018a) "A Monitoring and Diagnostic Approach for Stochastic Textured Surfaces", Technometrics, 60, 1-13.

See Also

`kerMat`,`bp`

stationaryTest *Stationary Test for Images*

Description

This function is a wrapper for the `TOS2D{LS2Wstat}` function that tests if a given image is stationary or not.

Usage

```
stationaryTest(img, nsamples = 100, ...)
```

Arguments

`img`	the given image to be tested.
`nsamples`	the number of bootstrap samples in the stationary test.
`...`	other arguments of the `TOS2D` function.

Details

See `TOS2D`.

Value

See `TOS2D`.

surfacemodel *Statistical reprentations of stochastic textured surfaces using supervised learning*

Description

Provides a statistical represenation for a given stochastic textured surface image via a supervised learning model (a regression tree in this version).

Usage

```
surfacemodel(img, nb, trim.vars = TRUE, cp = 1e-5,
             xval = 5, standardize = TRUE, subsample = 1,
             verbose = FALSE, keep.residuals = FALSE,
             stationary.test = FALSE, conf.level = .95, nsamples = 20)
```

Arguments

`img`	the given stochastic textured surface image in the matrix format.
`nb`	the size of the neighborhood. It must be a 1-length or 3-length vector of positive integer(s). If the former, it is the same with a 3-length vector with the same elements.
`trim.vars`	if `TRUE`, refit the model using only the variables that were used in the first fit.
`cp`	the complexity parameter for rpart fits (see `rpart.control`).
`xval`	the number of folds in cross-validation (see `rpart.control`). If `xval <= 1`, cross-validation will not be used.
`standardize`	if `TRUE`, standardize the given image `img <-(img -mean(img))/sd(img)`. This reduces the effect of different lighting conditions when images are taken.
`subsample`	the portion of pixels in the given image `img` to be used. It takes values in (0,1]. `subsample = .5` means that roughly a half number of pixels is used.
`verbose`	if `TRUE`, output some model fitting information.
`keep.residuals`	if `TRUE`, keep residuals of the fitted model in the output.
`stationary.test`	if `TRUE`, perform a stationary test for the given image `img`.
`conf.level`	the confidence level of the stationary test. If the *p*-value of the test is not greater than `1 -conf.level`, the function will stop and return the *p*-value.
`nsamples`	the number of bootstrap samples in the stationary test.

Value

A `surfacemodel` object containing the following components:

`fit`	the pruned `rpart` tree using cross-validation.
`trim.vars`	the `trim.vars` argument.
`nb`	the `nb` argument.
`Fr`	the empirical cdf with exponential tail approximation of the model residuals.
`MSE`	the mean squared residuals.
`standardize`	the `standardize` argument.
`R2cv`	the cross-validated R-squared of `fit`.
`complexity`	the complexity value of the returned `fit`.
`vars`	the variables used in the formula when fitting the model.
`p.value`	the p-value of the stationary test.
`nsamples`	the number of bootstrap samples in the stationary test.
`residuals`	the residuals of the fitted model.

Note

The best value for the neighborhood size nb argument can be chosen by comparing the cross-validated R-squared values R2cv of models built with different values of nb. Users may use 'surfacemodel' with some initial large nb, and then use the showNb() function to visualize the importance of the predictors used in the fitted model to have some idea about the range of important predictors to reduce (or increase if necessary) nb.

After finalizing the choice of nb, it is better to set trim.vars = TRUE to further remove some unused variables within that neighborhood.

The raster scan order for constructing the neiborhood data in dataPrep() is left-to-right then top-to-bottom (see Bui and Apley 208a). Rotating the image by every 90 degrees could be used to quicly change to some other raster scan orders. Again, the cross-validated R-squared R2cv output can be used to select the best raster scan order. See the below examples.

The stationary test is performed by using the TOS2D function in the LS2Wstat package.

plot.surfacemodel() is a generic function for surfacemodel() that produces two plots: a plot of the cross-validation R-squared against the complexity parameter and a histogram of the residuals (along with a normal density curve) of the fitted model.

Author(s)

Anh Bui

References

Bui, A.T. and Apley., D.W. (2018a) "A Monitoring and Diagnostic Approach for Stochastic Textured Surfaces", Technometrics, 60, 1-13.

See Also

dataPrep, showNb, monitoringStat, rpart

Examples

```
## fit a model to characterize the surface of a simulated image:
img <- sarGen(m = 50, n = 50, border = 50) # training image
model <- surfacemodel(img, nb = 1, keep.residuals = TRUE) # see Note above for how to select nb
model
# plot cross-validation R-squared against complexity parameter and residual histogram
plot(model, type=1:2)

## change the raster scan order from left-to-right then top-to-bottom to
## left-to-right then bottom-to-top, and re-fit the model
## (see the Note section above)
img2  <- as.matrix(t(apply(img , 2, rev)))
model2 <- surfacemodel(img2, nb = 1)
model2$R2cv # cross-validation R-squared
```

twms *Time-Weighted Moving Statistic*

Description

Computes time-weighted moving statistics EWMA or tabular CUSUM

Usage

```
twms(x, type = c("ewma", "cusum"), lambda, mu0, K, x0 = 0)
```

Arguments

`x`	the vector of observations to compute the time-weighted moving statistic for.
`type`	the type of statistic used in the computation.
`lambda`	the parameter of EWMA
`mu0`	the mean of the observations
`K`	the parameter of tabular CUSUM
`x0`	the starting value for the time-weighted moving statistics.

Value

the EWMA or tabular CUSUM statistics

Author(s)

Anh Bui

Examples

```
z <- twms(1:10, "ewma", lambda=0.2)
C <- twms(1:10, "cusum", mu0=5, K=1)
```

Index