Package 'bioimagetools'

September 18, 2015

Version 0.02.23

Date 2015-06-16

Title Some tools for bioimaging

Author Volker Schmid

Maintainer Volker Schmid <stats@volkerschmid.de>

Depends R (>= 2.14.0), utils

Imports EBImage, tiff

Suggests parallel, spatstat,
knitr

Description Some tools for bioimaging

License GPL

URL http://volkerschmid.de

VignetteBuilder knitr

R topics documented:

`bwlabel3d` *Binary segmentation in 3d*

Description

Binary segmentation in 3d

Usage

```
bwlabel3d(img)
```

Arguments

`x`	A 3d array. x is considered as a binary image, whose pixels of value 0 are considered as background ones and other pixels as foreground ones.

Value

A Grayscale 3d array, containing the labelled version of x.

`cmoments3d` *Computes moments from image objects*

Description

Computes intensity-weighted centers of objects and their mass (sum of intensities) and size.

Usage

```
cmoments3d(mask, ref)
```

Arguments

`mask`	a labeled stack as returned from bwlabel3d
`ref`	the original image stack

Value

a matrix with the moments of the objects in the stack

`cnnTest` *Permutation Test for cross-type nearest neighbor distances*

Description

Permutation Test for cross-type nearest neighbor distances

Usage

```
cnnTest(dist, n1, n2, w = rep(1, n1 + n2), B = 999, alternative = "less",
  returnSample = TRUE, papply = if (require("multicore")) mclapply else
  lapply, ...)
```

Arguments

`dist`	a distance matrix, the upper n1 x n1 part contains distances between objects of type 1 the lower n2 x n2 part contains distances between objects of type 2
`n1,`	n2 numbers of objects of type 1 and 2 respectively
`w`	(optional) weights of the objects (length n1+n2)
`B`	number of permutations to generate
`alternative`	alternative hypothesis ("less" to test H0:Colocalization)
`returnSample`	return sampled null distibution
`papply`	which apply function to use for generating the null distribution, defaults to mclapply if multicore is available, else lapply
`...`	additional arguments for papply

Value

a list with the p.value, the observed weighted mean of the cNN-distances, alternative and (if return-Sample) the simulated null dist

colors.in.classes *Title Compute colors in classes distribution*

Description

Title Compute colors in classes distribution

Usage

```
colors.in.classes(classes, color1, color2 = NULL, mask = array(TRUE,
  dim(classes)), N = max(classes, na.rm = TRUE), thresh1 = NULL,
  thresh2 = NULL, sd1 = 2, sd2 = 2, col1 = "green", col2 = "red",
  test = FALSE, plot = TRUE)
```

Arguments

`classes`	Image of classes
`color1`	Image of first color
`color2`	Image of second color
`mask`	Image mask
`N`	Maximum number of classes
`thresh1`	Threshold for first color image
`thresh2`	Threshold for second color image
`sd1`	For automatic threshold, that is: mean(color1)+sd1*sd(color1)
`sd2`	For automatic threshold of color2
`col1`	Name of color 1
`col2`	Name of color 2
`test`	Compute tests
`plot`	Plot barplots

Value

Table of classes with color 1 (and 2)

`crossNN` *Compute cross-type nearest neighbor distances*

Description

Compute cross-type nearest neighbor distances

Usage

```
crossNN(dist, n1, n2, w = rep(1, n1 + n2))
```

Arguments

`dist`	a distance matrix, the upper n1 x n1 part contains distances between objects of type 1 the lower n2 x n2 part contains distances between objects of type 2
`n1,`	n2 numbers of objects of type 1 and 2 respectively
`w`	optional weights of the objects (length n1+n2), defaults to equal weights

Value

a (n1+n2) x 2 matrix with the cross-type nearest neighbor distances and weights given as the sum of the weights of the involved objects

`distance2border` *A function to compute the distance from*

Description

Find distances to borders in classified image

Usage

```
distance2border(points, img.classes, x.microns, y.microns, z.microns, class1, class2 = NULL, mask = a
```

Arguments

`points`	Data frame containing the coordinates of points in microns as X-, Y-, and Z-variables.
`img.classes`	3D array (or image) of classes for each voxel.
`x.microns`	Size of image in x-direction in microns.
`y.microns`	Size of image in y-direction in microns.
`z.microns`	Size of image in z-direction in microns.
`class1`	Which class is the reference class. If is.null(class2), the function computes the distance of points to the border of class (in img.classes).

`class2`	Which class is the second reference class. If not is.null(class2), the function computes the distance of points from the border between classes class1 and class2. Default: class2=NULL.
`mask`	Array of mask. Needs to have same dimension as img.classes. Only voxels with mask[i,j,k]==TRUE are used. Default: array(TRUE,dim(img.classes))
`hist`	Automatically plot histogram using hist() function. Default: FALSE.
`main`	If (hist) title of histogramm. Default: "Minimal distance to border".
`xlab`	If (hist) description of x axis. Default: "Distance in Microns".
`xlim`	If (hist) vector of range of x axis (in microns). Default: c(-.3,.3)
`n`	If (hist) number of bins used in hist(). Default: 20.
`stats`	If (hist) write statistics into plot. Default: TRUE.
`file`	If (hist) the file name of the produced png. If NULL, the histogram is plotted to the standard device. Default: NULL.

Details

This function computes the distances from points to the border of a class or the border between two classes. For the latter, only points in these two classes are used.

Value

The function returns a vector with distances. Negative values correspond to points lying in class1.

Note

Warning: So far no consistency check for arguments is done. E.g., distance2border(randompoints,img.classes=array(1,c(100,
will fail with some cryptic error message (because class1 > max(img.classes)).

Author(s)

Volker Schmid

Examples

```
## Not run:
require(bioimagetools)
#simulate random data
randompoints<-data.frame("X"=runif(100,0,3),"Y"=runif(100,0,3),"Z"=runif(100,0,.5)) # coordinates in microns!
plot(randompoints$X,randompoints$Y,xlim=c(0,3),ylim=c(0,3),pch=19)

# points in a circle
circlepoints<-read.table(system.file("extdata","kreispunkte.table",package="bioimagetools"),header=TRUE)
plot(circlepoints$X,circlepoints$Y,xlim=c(0,3),ylim=c(0,3),pch=19)

# a circle like image
img<-readTIF(system.file("extdata","kringel.tif",package="bioimagetools"))
img<-array(img,dim(img)) # save as array for easier handling
image(img[,,1])
```

```
#and a mask
mask<-readTIF(system.file("extdata","amask.tif",package="bioimagetools"))
mask<-array(mask==65536,dim(mask)) # save as array for easier handling
image(mask[,,1])


xy.microns <- 3 # size in x and y direction (microns)
z.microns <- 0.5 # size in z direction (microns)

# distance from points to class
d1<-distance2border(randompoints, img, xy.microns, xy.microns, z.microns, class1=1,hist=TRUE)
d2<-distance2border(circlepoints, img, xy.microns, xy.microns, z.microns, class1=1,hist=FALSE)
plot(density(d2),type="l")
lines(c(0,0),c(0,10),lty=3)
lines(density(d1),col="blue")

# use mask, should give some small changes
d3<-distance2border(circlepoints, img, xy.microns, xy.microns, z.microns, class1=1,mask=mask,hist=FALSE)
plot(density(d2),type="l")
lines(c(0,0),c(0,10),lty=3)
lines(density(d3),col="blue")

# distance from border between classes
anotherimg<-img+mask
image(seq(0,3,length=300),seq(0,3,length=300),anotherimg[,,1])
points(circlepoints,pch=19)
d4<-distance2border(circlepoints, anotherimg, xy.microns, xy.microns, z.microns, class1=1,class2=2)
plot(density(d4),lwd=2)

# this should give the same answer
d5<-distance2border(circlepoints, anotherimg, xy.microns, xy.microns, z.microns, class1=2,class2=1)
lines(density(-d5),lty=3,col="blue",lwd=1.5)

## End(Not run)
```

`filter` *Apply filter to 3D images*

Description

A filter is applied to a 3D array representing an image. So far only variance filters are supported.

Usage

```
filter(img, filter="var", window, z.scale=1)
```

Arguments

`img`	is a 3d arrary representing an image.
`filter`	is the filter to be applied. Options: var: Variance filter.
`window`	half size of window; i.e. window=1 uses a window of 3 voxels in each direction.
`z.scale`	ratio of voxel dimension in x/y direction and z direction.

Value

Multi-dimensional array of filtered image data.

Author(s)

Volker J. Schmid `<volkerschmid@users.sourceforge.net>`

`img` *image for microscopy*

Description

image for microscopy

Usage

```
img(x, col = "grey", ...)
```

Arguments

`x`	Image, 2D Matrix
`col`	Color, "grey", "red" ("r"), "green" ("g") or "blue" ("b")
`...`	other parameters for graphics::image

Value

no return

`intensity3d` *Intensity of a 3d Dataset or a Model*

Description

Generic function for computing the intensity of a spatial dataset or spatial point process model.

Usage

```
intensity3d(X,...)
```

Arguments

`X`	A spatial dataset or a spatial point process model.
`...`	Further arguments depending on the class of `X`.

Details

This is a generic function for computing the intensity of a spatial dataset or spatial point process model. There are methods for point patterns (objects of class `"ppp"`) and fitted point process models (objects of class `"ppm"`). The empirical intensity of a dataset is the average density (the average amount of <e2><80><98>stuff<e2><80><99> per unit area or volume). The empirical intensity of a point pattern is computed by the method `intensity.ppp`. The theoretical intensity of a stochastic model is the expected density (expected amount of <e2><80><98>stuff<e2><80><99> per unit area or volume). The theoretical intensity of a fitted point process model is computed by the method `intensity.ppm`.

Value

Usually a numeric value or vector.

Author(s)

Adrian Baddeley `<Adrian.Baddeley@uwa.edu.au>` `http://www.maths.uwa.edu.au/~adrian/` and Rolf Turner `<r.turner@auckland.ac.nz>`

See Also

`intensity.ppp`, `intensity.ppm`.

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (x)
```

```
{
  }
```

Kcross3D *Multitype K Function (Cross-type)*

Description

For a multitype point pattern, estimate the multitype $K$ function which counts the expected number of points of type $j$ within a given distance of a point of type $i$.

Usage

```
Kcross3D(X, i, j, r=NULL, breaks=NULL, correction, ..., ratio=FALSE)
```

Arguments

`X`	The observed point pattern, from which an estimate of the cross type $K$ function $K_ij(r)$ will be computed. It must be a multitype point pattern (a marked point pattern whose marks are a factor). See under Details.
`i`	The type (mark value) of the points in `X` from which distances are measured. A character string (or something that will be converted to a character string). Defaults to the first level of `marks(X)`.
`j`	The type (mark value) of the points in `X` to which distances are measured. A character string (or something that will be converted to a character string). Defaults to the second level of `marks(X)`.
`r`	numeric vector. The values of the argument $r$ at which the distribution function $K_ij(r)$ should be evaluated. There is a sensible default. First-time users are strongly advised not to specify this argument. See below for important conditions on $r$.
`breaks`	This argument is for internal use only.
`correction`	A character vector containing any selection of the options `"border"`, `"bord.modif"`, `"isotropic"`, `"` or `"best"`. It specifies the edge correction(s) to be applied.
`...`	Ignored
`ratio`	Logical. If `TRUE`, the numerator and denominator of each edge-corrected estimate will also be saved, for use in analysing replicated point patterns.

Details

This function `Kcross` and its companions `Kdot` and `Kmulti` are generalisations of the function `Kest` to multitype point patterns.

A multitype point pattern is a spatial pattern of points classified into a finite number of possible <e2><80><9c>colours<e2><80><9d> or <e2><80><9c>types<e2><80><9d>. In the \textbfspatstat package, a multitype pattern is represented as a single point pattern object in which the points carry marks, and the mark value attached to each point determines the type of that point.

The argument `X` must be a point pattern (object of class `"ppp"`) or any data that are acceptable to `as.ppp`. It must be a marked point pattern, and the mark vector `X$marks` must be a factor.

The arguments `i` and `j` will be interpreted as levels of the factor `X$marks`. If `i` and `j` are missing, they default to the first and second level of the marks factor, respectively.

The <e2><80><9c>cross-type<e2><80><9d> (type $i$ to type $j$) $K$ function of a stationary multitype point process $X$ is defined so that $\lambda_j K_ij(r)$ equals the expected number of additional random points of type $j$ within a distance $r$ of a typical point of type $i$ in the process $X$. Here $\lambda_j$ is the intensity of the type $j$ points, i.e. the expected number of points of type $j$ per unit area. The function $K_ij$ is determined by the second order moment properties of $X$.

An estimate of $K_ij(r)$ is a useful summary statistic in exploratory data analysis of a multitype point pattern. If the process of type $i$ points were independent of the process of type $j$ points, then $K_ij(r)$ would equal $\pi r^2$. Deviations between the empirical $K_ij$ curve and the theoretical curve $\pi r^2$ may suggest dependence between the points of types $i$ and $j$.

This algorithm estimates the distribution function $K_ij(r)$ from the point pattern `X`. It assumes that `X` can be treated as a realisation of a stationary (spatially homogeneous) random spatial point process in the plane, observed through a bounded window. The window (which is specified in `X` as `X$window`) may have arbitrary shape. Biases due to edge effects are treated in the same manner as in `Kest`, using the border correction.

The argument $r$ is the vector of values for the distance $r$ at which $K_ij(r)$ should be evaluated. The values of $r$ must be increasing nonnegative numbers and the maximum $r$ value must not exceed the radius of the largest disc contained in the window.

The pair correlation function can also be applied to the result of `Kcross`; see `pcf`.

Value

An object of class `"fv"` (see `fv.object`).

Essentially a data frame containing numeric columns

`r`	the values of the argument $r$ at which the function $K_ij(r)$ has been estimated
`theo`	the theoretical value of $K_ij(r)$ for a marked Poisson process, namely $\pi r^2$

together with a column or columns named `"border"`, `"bord.modif"`, `"iso"` and/or `"trans"`, according to the selected edge corrections. These columns contain estimates of the function $K_ij(r)$ obtained by the edge corrections named.

If `ratio=TRUE` then the return value also has two attributes called `"numerator"` and `"denominator"` which are `"fv"` objects containing the numerators and denominators of each estimate of $K(r)$.

Warnings

The arguments `i` and `j` are always interpreted as levels of the factor `X$marks`. They are converted to character strings if they are not already character strings. The value `i=1` does \textbfnot refer to the first level of the factor.

Author(s)

Adrian Baddeley <Adrian.Baddeley@uwa.edu.au> http://www.maths.uwa.edu.au/~adrian/ and Rolf Turner <r.turner@auckland.ac.nz>

References

Cressie, N.A.C. \emphStatistics for spatial data. John Wiley and Sons, 1991. Diggle, P.J. \emphStatistical analysis of spatial point patterns. Academic Press, 1983. Harkness, R.D and Isham, V. (1983) A bivariate spatial point pattern of ants<e2><80><99> nests. \emphApplied Statistics \textbf32, 293<e2><80><93>303 Lotwick, H. W. and Silverman, B. W. (1982). Methods for analysing spatial processes of several types of points. \emphJ. Royal Statist. Soc. Ser. B \textbf44, 406<e2><80><93>413. Ripley, B.D. \emphStatistical inference for spatial processes. Cambridge University Press, 1988. Stoyan, D, Kendall, W.S. and Mecke, J. \emphStochastic geometry and its applications. 2nd edition. Springer Verlag, 1995.

See Also

`Kdot`, `Kest`, `Kmulti`, `pcf`

`L.cross.3D` *Multitype L-function (cross-type)*

Description

Calculates an estimate of the cross-type L-function for a multitype point pattern.

Usage

```
L.cross.3D(X, Y, Z, X2, Y2, Z2, psz = 25, width = 1, intensity = NULL,
  intensity2 = NULL, parallel = FALSE)
```

Arguments

`X`	X coordinate of first observed point pattern in microns.
`Y`	Y coordinate
`Z`	Z coordinate
`X2`	X coordinate of second observed point pattern
`Y2`	Y coordinate
`Z2`	Z coordinate
`psz`	pointsize used for discetization. Smaller values are more precise, but need more computation time.
`width`	maximum distance

`intensity`	intensity of first pattern. Only if $$\lambda(s)! = \lambda$$.
`intensity2`	intensity of second pattern
`parallel`	Logical. Can we use parallel computing?

Value

a list of breaks and counts.

`mexican.hat.brush` *Mexican hat brush to use with filter2*

Description

Mexican hat brush to use with filter2

Usage

```
mexican.hat.brush(n = 7, sigma2 = 1)
```

Arguments

`n`	size of brush
`sigma2`	standard deviation

Value

brush

`nextneighbourdistribution` *Ploting next neighbour distribution of 3D point patterns*

Description

Computes the minimal distances between next neighbours and plots their distribution. For marked points (e.g., red and green), same=FALSE computes the minimal distances to the next neighbour of the other mark (color). X, Y and Z coordinates of the point pattern have to be given in microns.

Usage

```
nndist(X,Y,Z,X2=X,Y2=Y,Z2=Z,same=TRUE,psz=25,main="Next neighbour distribution",file=NULL, return=FA
```

Arguments

`X`	X coordinate of (first) point pattern in microns.
`Y`	Y coordinate of (first) point pattern in microns.
`Z`	Z coordinate of (first) point pattern in microns.
`X2`	X2 coordinate of second point pattern in microns.
`Y2`	Y2 coordinate of second point pattern in microns.
`Z2`	Z2 coordinate of second point pattern in microns.
`same`	is a boolean parameter; TRUE if all points are of same color.
`psz`	is the approximate number of pixels per micron (for internal purposes only).
`main`	is the title of the plot.
`file`	is the file name of the png file. Default is NULL, which plots to the Null Device.
`return`	if TRUE, return next neighbour histogram (default: FALSE).

Value

Histogram of Next Neighbour Distribution (if return=TRUE).

Author(s)

Volker J. Schmid <`volkerschmid@users.sourceforge.net`>

`nextneighbourenvelope` *Envelope for distribution of next-neighbour distances of 3D point patterns*

Usage

```
nextneighbourenvelope(x)
```

Arguments

`x`

Author(s)

Volker J. Schmid <`volkerschmid@users.sourceforge.net`>

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (x)
{
  }
```

`outside` *Segmentation of the background of 3D images based on classes*

Description

Segmentation of the background of 3D images based on classes.

Usage

```
outside(img,what,blobsize=1)
```

Arguments

`img`	is a 3d arrary representing an image.
`what`	is an integer of the class of the background.
`blobsize`	is an integer, representing the minimal diameter for bridges from the outside. E.g., a blobsize=3 allows for holes of size 2*(blobsize-1)=4 in the edge of the object.

Value

A binary 3d array: 1 outside the object, 0 inside the object.

Author(s)

Volker J. Schmid <`volkerschmid@users.sourceforge.net`>

`points2class` *Count points in classified image*

Usage

```
points2class(x)
```

Arguments

`x`

Author(s)

Volker J. Schmid <`volkerschmid@users.sourceforge.net`>

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (x)
{
  }
```

`preprocess` *Get central moments of objects in a single-channel image stack*

Description

Uses the methodology used for segmentation in the EBImage vignette (threshhold->opening->fillHull) from all 3 spatial directions and overlays these results to get a binary image which is then segmented with bwlabel3d. Central moments are extracted with cmoments3d

Usage

```
preprocess(file, threshold = 0.95, threshW = 5, threshH = 5,
  brushsize = 3, quantile = TRUE)
```

Arguments

`file`	the path of the image stack
`threshold`	the quantile of intensities used for thresholding if quantile=TRUE or the intensity value if quantile=FALSE, defaults to the 80% quantile
`threshW,`	threshH width and height of the moving rectangular window for threshold, defaults to 5.
`brushsize`	the brushsize for makeBrush for opening, defaults to 3
`quantile`	defaults to TRUE

Value

a list with the original stack, the labeled stack, and the matrix of central moments of the found objects

`readBMP` *Read bitmap files*

Description

Read 2D grey-value BMP files

Usage

```
readBMP(file)
```

Arguments

`file`	A character vector of file names or URLs.

Value

Returns a matrix with BMP data as integer.

Author(s)

Volker J. Schmid

Examples

```
bi<-readBMP("http://www.statistik.lmu.de/institut/ag/bioimg/bit/ratbert.bmp")
image(bi,col=grey(seq(1,0,length=100)))
```

`readTIF` *Read tif stacks*

Description

Read tif stacks

Usage

```
readTIF(file = file.choose(), native = FALSE, as.is = FALSE,
  channels = NULL)
```

Arguments

`file`	Name of the file to read from.
`native`	determines the image representation - if FALSE (the default) then the result is an array, if TRUE then the result is a native raster representation (suitable for plotting).
`as.is`	attempt to return original values without re-scaling where possible
`channels`	number of channels

Value

3d or 4d array

---

segment — *Segmentation of 3D images using EM algorithms*

Description

egmentation of 3D images using EM algorithms

Usage

```
segment(img, nclust, beta, z.scale = 0, method = "cem", varfixed = TRUE,
  maxit = 30, mask = array(TRUE, dim(img)), priormu = rep(NA, nclust),
  priormusd = rep(NULL, nclust), min.eps = 10^{     -7 },
  inforce.nclust = FALSE, start = NULL)
```

Arguments

`img`	is a 3d arrary representing an image.
`nclust`	is the number of clusters/classes to be segmented.
`beta`	is a matrix of size nclust x nclust, representing the prior weight of classes neighbouring each other.
`z.scale`	ratio of voxel dimension in x/y direction and z direction. Will be multiplied on beta for neighbouring voxel in z direction.
`method`	only "cem" classification EM algorithm implemented.
`varfixed`	is a logical variable. If TRUE, variacne is equal in each class.
`maxit`	is the maximum number of iterations.
`mask`	is a logical array, representing the voxels to be used in the segmentation.
`priormu`	is a vector with mean of the normal prior of the expected values of all classes. Default is NA, which represents no prior assumption.
`priormusd`	is a vector with standard deviations of the normal prior of the expected values of all classes.
`min.eps`	stop criterion. Minimal change in sum of squared estimate of mean in order to stop.
`inforce.nclust`	if TRUE enforces number of clusters to be nclust. Otherwise classes might be removed during algorithm.
`start`	?

Value

A list with "class": 3d array of class per voxel; "mu" estimated means; "sigma": estimated standard deviations.

Examples

```
original<-array(1,c(300,300,50))
for (i in 1:5)original[(i*60)-(0:20),,]<-original[(i*60)-(0:20),,]+1
for (i in 1:10)original[,(i*30)-(0:15),]<-original[,(i*30)-(0:15),]+1
original[,,26:50]<-4-aperm(original[,,26:50],c(2,1,3))

img<-array(rnorm(300*300*50,original,.2),c(300,300,50))
img<-img-min(img)
img<-img/max(img)

beta<-matrix(rep(-.5,9),nrow=3)
beta<-beta+1.5*diag(3)

seg.img<-segment(img,3,beta,z.scale=.3)

print(sum(seg.img$class!=original)/prod(dim(original)))
## Not run:
  EBImage::display(seg.img$class/3)

## End(Not run)
```

`segment.outside` *Segmentation of the background of 3D images based on automatic threshold*

Description

Segmentation of the background of 3D images. Starting from the borders of the image, the algorithm trys to find the edges of an object in the middle of the image. From this, a threshold for the edge is defined automatically. The function then return the a logical array representing voxel inside the object.

Usage

```
segment.outside(img,blobsize=1)
```

Arguments

`img`	is a 3d arrary representing an image.
`blobsize`	is an integer, representing the minimal diameter for bridges from the outside. E.g., a blobsize=3 allows for holes of size 2*(blobsize-1)=4 in the edge of the object.

Value

A binary 3d array: 1 outside the object, 0 inside the object.

Author(s)

Volker J. Schmid <`volkerschmid@users.sourceforge.net`>

`standardize` *Standardize images*

Description

Standardizes images in order to compare different images. Mean of standardized image is 0.5, standard deviation is sd.

Usage

```
standardize(img,mask=array(TRUE,dim(img)),N=32,sd=1/6)
```

Arguments

`img`	is a 2d/3d arrary representing an image.
`mask`	a mask.
`N`	number of classes.
`sd`	standard deviation.

Value

Multi-dimensional array of standardized image.

Author(s)

Volker J. Schmid <volkerschmid@users.sourceforge.net>

Examples

```
#simuliere Daten zum Testen
test2<-runif(128*128,0,1)
test2<-sort(test2)
test2<-array(test2,c(128,128))
image(test2,col=grey(seq(0,1,by=1/1000)))

# Standardisiere test2 in 32 Klassen
std<-standardize(test2,N=32,sd=4)
```

tablen *Cross Tabulation and Table Creation (including empty classes)*

Usage

```
tablen(x)
```

Arguments

`...`	one or more objects which can be interpreted as factors (including character strings), or a list (or data frame) whose components can be so interpreted. (For `as.table` and `as.data.frame`, arguments passed to specific methods.)
`exclude`	levels to remove for all factors in .... If set to `NULL`, it implies `useNA = "always"`. See <e2><80><98>Details<e2><80><99> for its interpretation for non-factor arguments.
`useNA`	whether to include `NA` values in the table. See <e2><80><98>Details<e2><80><99>.
`dnn`	the names to be given to the dimensions in the result (the *dimnames names*).
`deparse.level`	controls how the default `dnn` is constructed. See <e2><80><98>Details<e2><80><99>.
`x`	an arbitrary `R` object, or an object inheriting from class `"table"` for the `as.data.frame` method.
`row.names`	a character vector giving the row names for the data frame.
`responseName`	The name to be used for the column of table entries, usually counts.
`stringsAsFactors`	logical: should the classifying factors be returned as factors (the default) or character vectors?

Details

If the argument `dnn` is not supplied, the internal function `list.names` is called to compute the <e2><80><98>dimname names<e2><80><99>. If the arguments in ... are named, those names are used. For the remaining arguments, `deparse.level = 0` gives an empty name, `deparse.level = 1` uses the supplied argument if it is a symbol, and `deparse.level = 2` will deparse the argument.

Only when `exclude` is specified and non-NULL (i.e., not by default), will `table` potentially drop levels of factor arguments.

`useNA` controls if the table includes counts of `NA` values: the allowed values correspond to never, only if the count is positive and even for zero counts. This is overridden by specifying `exclude = NULL`. Note that levels specified in `exclude` are mapped to `NA` and so included in `NA` counts.

Both `exclude` and `useNA` operate on an "all or none" basis. If you want to control the dimensions of a multiway table separately, modify each argument using `factor` or `addNA`.

It is best to supply factors rather than rely on coercion. In particular, `exclude` will be used in coercion to a factor, and so values (not levels) which appear in `exclude` before coercion will be mapped to `NA` rather than be discarded.

The `summary` method for class `"table"` (used for objects created by `table` or `xtabs`) which gives basic information and performs a chi-squared test for independence of factors (note that the function `chisq.test` currently only handles 2-d tables).

Value

`tablen()` returns a *contingency table*, an object of class `"table"`, an array of integer values. Note that unlike S the result is always an array, a 1D array if one factor is given.

`as.table` and `is.table` coerce to and test for contingency table, respectively.

The `as.data.frame` method for objects inheriting from class `"table"` can be used to convert the array-based representation of a contingency table to a data frame containing the classifying factors and the corresponding entries (the latter as component named by `responseName`). This is the inverse of `xtabs`.

Author(s)

Volker J. Schmid `<volkerschmid@users.sourceforge.net>`

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

See Also

`tabulate` is the underlying function and allows finer control.

Use `ftable` for printing (and more) of multidimensional tables. `margin.table, prop.table, addmargins`.

Examples

```
require(stats) # for rpois and xtabs
## Simple frequency distribution
table(rpois(100, 5))
## Check the design:
with(warpbreaks, table(wool, tension))
table(state.division, state.region)

# simple two-way contingency table
with(airquality, table(cut(Temp, quantile(Temp)), Month))

a <- letters[1:3]
table(a, sample(a))                    # dnn is c("a", "")
table(a, sample(a), deparse.level = 0) # dnn is c("", "")
table(a, sample(a), deparse.level = 2) # dnn is c("a", "sample(a)")

## xtabs() <-> as.data.frame.table() :
UCBAdmissions ## already a contingency table
DF <- as.data.frame(UCBAdmissions)
class(tab <- xtabs(Freq ~ ., DF)) # xtabs & table
## tab *is* "the same" as the original table:
all(tab == UCBAdmissions)
all.equal(dimnames(tab), dimnames(UCBAdmissions))

a <- rep(c(NA, 1/0:3), 10)
table(a)
table(a, exclude = NULL)
b <- factor(rep(c("A","B","C"), 10))
table(b)
table(b, exclude = "B")
d <- factor(rep(c("A","B","C"), 10), levels = c("A","B","C","D","E"))
table(d, exclude = "B")
print(table(b, d), zero.print = ".")

## NA counting:
is.na(d) <- 3:4
d. <- addNA(d)
d.[1:7]
table(d.) # ", exclude = NULL" is not needed
## i.e., if you want to count the NA's of 'd', use
table(d, useNA = "ifany")

## Two-way tables with NA counts. The 3rd variant is absurd, but shows
## something that cannot be done using exclude or useNA.
with(airquality,
   table(OzHi = Ozone > 80, Month, useNA = "ifany"))
with(airquality,
   table(OzHi = Ozone > 80, Month, useNA = "always"))
with(airquality,
   table(OzHi = Ozone > 80, addNA(Month)))
```

`testColoc` *Permutation Test for cross-type nearest neighbor distances*

Description

Permutation Test for cross-type nearest neighbor distances

Usage

```
testColoc(im1, im2, hres = 0.102381, vres = 0.25, B = 999,
  alternative = "less", returnSample = TRUE, ...)
```

Arguments

`im1,`	im2 image stacks as returned by preprocess
`hres,`	vres horizontal and vertical resolution of the stacks
`B`	number of permutations to generate
`alternative`	alternative hypothesis ("less" to test H0:Colocalization)
`returnSample`	return sampled null distibution
`...`	additional arguments for papply

Value

a list with the p.value, the observed weighted mean of the cNN-distances

writeTIF — *Writes image stack into a TIFF file. Wrapper for writeTIFF*

Description

Writes image stack into a TIFF file. Wrapper for writeTIFF

Usage

```
writeTIF(img, file, bps = NULL, twod = FALSE, reduce = TRUE,
  attr = attributes(img))
```

Arguments

`img`	An image, a 3d or 4d array.
`file`	File name.
`bps`	number of bits per sample (numeric scalar). Supported values in this version are 8, 16, and 32.
`twod`	Dimension of channels. TRUE for 2d images, FALSE for 3d images.
`attr`	Attributes of image stack. Will be propagated to each 2d image.

Index