Package ‘bioimagetools’

May 29, 2020

Version  1.1.5
Date  2020-05-28
Title  Tools for Microscopy Imaging
Author  Volker Schmid [aut, cre],
        Priyanka Kukreja [ctb],
        Fabian Scheipl [ctb]
Maintainer  Volker Schmid <stats@volkerschmid.de>
Depends  R (>= 3.5.0)
biocViews
Imports  parallel, tiff, stats, grDevices, utils, EBImage, httr
SystemRequirements  tiff fftw libcurl openssl
Description  Tools for 3D imaging, mostly for biology/microscopy.
             Read and write TIFF stacks. Functions for segmentation, filtering and analyzing 3D point patterns.
License  GPL-3
URL  https://bioimaginggroup.github.io/bioimagetools
BugReports  https://github.com/bioimaginggroup/bioimagetools/issues
RoxygenNote  7.1.0
Suggests  knitr, rmarkdown
VignetteBuilder  knitr
NeedsCompilation  yes
Repository  CRAN
Date/Publication  2020-05-29 13:20:02 UTC

R topics documented:

  bwlabel3d .............................................. 2
  cmoments3d ........................................... 3
  cnnTest .............................................. 3
  crossNN ............................................... 4
Binary segmentation in 3d

**Usage**

```r
bwlabel3d(img)
```

**Arguments**

- `img` 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 labeled version of x.

**Author(s)**

Fabian Scheipl, Volker Schmid
cmoments3d

Computes moments from image objects

Description

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

Usage

```r
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.

Author(s)

Volker Schmid

---

cnnTest

Permutation Test for cross-type nearest neighbor distances

Description

Permutation Test for cross-type nearest neighbor distances

Usage

```r
cnnTest(
  dist,
  n1,
  n2,
  w = rep(1, n1 + n2),
  B = 999,
  alternative = "less",
  returnSample = TRUE,
  parallel = FALSE,
  ...
)
```
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  numbers of objects of type 1

n2  numbers of objects of type 2

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 distribution

parallel  Logical. Should we use parallel computing?

...  additional arguments for mclapply

Value

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

Author(s)

Fabian Scheipl

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  numbers of objects of type 1

n2  numbers of objects of type 2

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

Author(s)

Fabian Scheipl

distance2border  A function to compute the distance from spots to borders of classes

Description

A function to compute the distance from spots to borders of classes

Usage

distance2border(
  points,
  img.classes,
  x.microns,
  y.microns,
  z.microns,
  class1,
  class2 = NULL,
  mask = array(TRUE, dim(img.classes)),
  voxel = FALSE,
  hist = FALSE,
  main = "Minimal distance to border",
  xlab = "Distance in Microns",
  xlim = c(-0.3, 0.3),
  n = 20,
  stats = TRUE,
  file = NULL,
  silent = FALSE,
  parallel = FALSE
)

Arguments

points  Data frame containing the coordinates of points in microns as X-, Y-, and Z-

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).
distance2border

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))

voxel Logical. If TRUE, points coordinates are given as voxels rather than in microns.

hist Automatically plot histogram using hist() function. Default: FALSE.

main If (hist) title of histogram. 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.

silent if TRUE, function remains silent during running time

parallel Logical. Can we use parallel computing?

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(100,100,2)),3,3,1,class1=2) will fail with some cryptic error message (because class1 > max(img.classes)).

Examples
## Not run:
#simulate random data
randompoints<-data.frame("X"=runif(1000,0,3),"Y"=runif(1000,0,3),"Z"=runif(1000,0,.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
filterImage3d

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

filterImage3d(img, filter = "var", window, z.scale = 1, silent = FALSE)
Arguments

- **img** is a 3d array 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.
- **silent** Logical. If FALSE, information on progress will be printed.

Value

Multi-dimensional array of filtered image data.

---

**folder.choose**  
*Choose a folder interactively*

**Description**

Choose a folder interactively by choosing a file in that folder.

**Usage**

```r
folder.choose()
```

**Value**

A character vector of length one giving the folder path.

---

**img**  
*Display an image stack*

**Description**

Display an image stack.

**Usage**

```r
img(
    x,
    z = NULL,
    ch = NULL,
    mask = NULL,
    col = "grey",
    low = NULL,
    up = NULL,
    ...
)
```
intensity3D

Arguments

x Image, 2D or 3D Matrix
z slice to show, default: NULL, expects x to be 2d or 2d+channels
ch channel. Default: NULL, either only one channel, rgb or channel will be assumed from col
mask mask for image, voxel outside the mask will be transparent (default: NULL, no mask)
col Color, either a character ("grey" or "gray", "greyinvert" or "grayinvert", "red" ("r"), "green" ("g") or "blue" ("b"), rgb" for 3D matrices), a vector of character with hex rgb values or a function.
low minimal value of shown intensity. Default: NULL: use min(x, na.rm=TRUE).
up maximal value of shown intensity. Default: NULL: use max(x, na.rm=TRUE).
... other parameters for graphics::image

Value

no return

intensity3D Intensity of a 3d Dataset or a Model

Description

Computing the intensity of a 3d point pattern using kernel smoothing.

Usage

tensity3D(X, Y, Z, bw = NULL, psz = 25, kernel = "Square")

Arguments

X X coordinate
Y Y coordinate
Z Z coordinate
bw bandwidth
psz pointsize used for discretization (large: fast, but not precise)
kernel "Square" or "Uniform"

Value

3d Array
K.cross.3D

K-function cross-type in 3D

Description

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

Usage

K.cross.3D(
  X,
  Y,
  Z,
  X2,
  Y2,
  Z2,
  psz = 25,
  width = 1,
  intensity = NULL,
  intensity2 = NULL,
  parallel = FALSE,
  verbose = 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 discretization. Smaller values are more precise, but need more
  computation time.
width
  maximum distance
intensity
  intensity of first pattern. Only if
  λ(s)! = λ

intensity2
  intensity of second pattern
parallel
  Logical. Can we use parallel computing?
verbose
  Plot verbose information

Value

a list of breaks and counts.
L.cross.3D \textit{L-function cross-type in 3d}

\textbf{Description}
Calculates an estimate of the cross-type L-function for a multitype point pattern.

\textbf{Usage}

\begin{verbatim}
L.cross.3D(
  X,
  Y,
  Z,
  X2,
  Y2,
  Z2,
  psz = 25,
  width = 1,
  intensity = NULL,
  intensity2 = NULL,
  parallel = FALSE,
  verbose = FALSE
)
\end{verbatim}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{X} \hspace{1cm} X coordinate of first observed point pattern in microns.
  \item \texttt{Y} \hspace{1cm} Y coordinate
  \item \texttt{Z} \hspace{1cm} Z coordinate
  \item \texttt{X2} \hspace{1cm} X coordinate of second observed point pattern
  \item \texttt{Y2} \hspace{1cm} Y coordinate
  \item \texttt{Z2} \hspace{1cm} Z coordinate
  \item \texttt{psz} \hspace{1cm} pointsize used for discretization. Smaller values are more precise, but need more computation time.
  \item \texttt{width} \hspace{1cm} maximum distance
  \item \texttt{intensity} \hspace{1cm} intensity of first pattern. Only if \(\lambda(s)! = \lambda\)
  \item \texttt{intensity2} \hspace{1cm} intensity of second pattern
  \item \texttt{parallel} \hspace{1cm} Logical. Can we use parallel computing?
  \item \texttt{verbose} \hspace{1cm} Plot verbose information
\end{itemize}

\textbf{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**

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

**Arguments**

- `n` size of brush
- `sigma2` standard deviation

**Value**

brush

---

nearest.neighbour.distribution

*Nearest neighbor distribution (D curve)*

**Description**

Nearest neighbor distribution (D curve)

**Usage**

```r
nearest.neighbour.distribution(  
  X,  
  Y,  
  Z,  
  X2 = X,  
  Y2 = Y,  
  Z2 = Z,  
  same = TRUE,  
  psz = 25,  
  main = "Nearest neighbour distribution",  
  file = NULL,  
  return = FALSE
)
```
nearestClassDistance

Arguments

X X coordinates of point pattern 1
Y Y coordinates of point pattern 1
Z Z coordinates of point pattern 1
X2 X coordinates of point pattern 2
Y2 Y coordinates of point pattern 2
Z2 Z coordinates of point pattern 2
same binary, FALSE for cross D curve
psz pointsize for discretization
main Title for graphic
file File name for PNG file. If NULL, plots to standard device.
return Logical. Return histogram?

Value

histogram of nearest neighbors

Examples

p<-read.csv(system.file("extdata","cell.csv",package="bioimagetools"))
nearest.neighbour.distribution(p$X,p$Y,p$Z)

---

nearestClassDistance     Title Find distance to next neighbour of a specific class

Description

Title Find distance to next neighbour of a specific class

Usage

nearestClassDistance(coord, img, class, voxelsize, step = 0)

Arguments

coord coordinate of relevant voxel
img image array of classes
class class to find
voxelsize vector of length three. size of voxel in X-/Y-/Z-direction
step size of window to start with

Value

distance to nearest voxel of class "class"
nearestClassDistances  *Find all distances to next neighbor of all classes*

**Description**

Find all distances to next neighbor of all classes

**Usage**

```r
earestClassDistances(
  img,
  voxelsize = NULL,
  size = NULL,
  classes = 7,
  maxdist = NULL,
  silent = FALSE,
  cores = 1
)
```

**Arguments**

- `img`: Image array of classes
- `voxelsize`: Real size of voxels in microns.
- `size`: Real size of image in microns. Either size or voxelsize must be given.
- `classes`: Number of classes
- `maxdist`: Maximum distance to consider
- `silent`: Remain silent?
- `cores`: Number of cores available for parallel computing

**Value**

array with distances

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

**Description**

Segmentation of the background of 3D images based on classes

**Usage**

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

**img**
is a 3d array 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 $2\times(blobsize-1)=4$ in the edge of the object.

Value

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

---

plotNearestClassDistances

*Title Plot nearest class distances*

Description

Title Plot nearest class distances

Usage

```r
plotNearestClassDistances(
  distances,
  method,
  classes = length(distances),
  ylim = c(0, 1),
  qu = 0.01,
  mfrow = NULL
)
```

Arguments

**distances**
list of list with distances as produced by nearestClassDistances()

**method**
"boxplot", "min" or "quantile"

**classes**
number of classes, default=7

**ylim**
limits for distances, default=c(0,1)

**qu**
quantile for method="quantile"; default 0.01

**mfrow**
mfrow option forwarded to par; default NULL, computes some optimal values

Value

plots
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**
```r
bi <- readBMP(system.file("extdata/V.bmp", package="bioimagetools"))
image(bi, col = grey(seq(1, 0, length = 100)))
```

readClassTIF

**Read TIF file with classes**

**Description**
Read TIF file with classes

**Usage**
readClassTIF(file, n = 7)

**Arguments**
- file: file
- n: number of classes

**Value**
array
readTIF

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. Can also be an URL.
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

Examples

kringel <- readTIF(system.file("extdata","kringel.tif",package="bioimagetools"))
img(kringel)

segment

Segmentation of 3D images using EM algorithms

Description

Segmentation 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,
  silent = FALSE
)

Arguments

img is a 3d array 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 neighboring each other.
z.scale ratio of voxel dimension in x/y direction and z direction. Will be multiplied on beta for neighboring voxel in z direction.
method only "cem" classification EM algorithm implemented.
varfixed is a logical variable. If TRUE, the variance 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 ?
silent if TRUE, function remains silent during running time

Value

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

```r
## Not run:
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] <- 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)

try1 <- segment(img, 3, beta = 0.5, z.scale = .3)
print(sum(try1$class != original) / prod(dim(original)))

beta <- matrix(rep(-.5, 9), nrow = 3)
beta <- beta + 1.5 * diag(3)
try2 <- segment(img, 3, beta, z.scale = .3)
print(sum(try2$class != original) / prod(dim(original)))

par(mfrow = c(2, 2))
img(original)
img(img)
img(try1$class)
img(try2$class)
## 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 tries 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 returns a logical array representing voxel inside the object.

Usage

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

Arguments

- **img** is a 3d array 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.

Examples

```r
kringel <- readTIF(system.file("extdata","kringel.tif",package="bioimagetools"))
out <- segment.outside(kringel)
img(out, z=1)
```

---

**spots**

Find spots based on threshold and minimum total intensity

Description

Find spots based on threshold and minimum total intensity

Usage

```r
spots(
  img,
  mask,
  thresh.offset = 0.1,
  window = c(5, 5),
  min.sum.intensity = 0,
  zero = NA,
  max.spots = NULL,
  return = "intensity"
)
```

Arguments

- `img`: image array.
- `mask`: mask array.
- `thresh.offset`: threshold for minimum voxel intensity.
- `window`: Half width and height of the moving rectangular window.
- `min.sum.intensity`: threshold for minimum total spot intensity
- `zero`: if NA, background is set to NA, if 0, background is set to 0.
- `max.spots`: find max.spots spots with highest total intensity.
- `return`: "mask" returns binarized mask, "intensity" returns intensity for spots, zero or NA otherwise "label" return labeled (numbered) spots.

Value

array
**standardize**

*Standardize images*

**Description**

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

**Usage**

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

**Arguments**

- `img` is a 2d/3d array representing an image.
- `mask` a mask.
- `log` Logical. Transform to log scale before standardization?
- `N` number of classes.
- `sd` standard deviation.

**Value**

Multi-dimensional array of standardized image.

**Examples**

```r
# simuliere Daten zum Testen
test2<-runif(128*128,0,1)
test2<-sort(test2)
test2<-array(test2,c(128,128))
img(test2)
# Standardisiere test2 in 32 Klassen
std<-standardize(test2,N=32,sd=4)
```

---

**table.n**

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

**Description**

Cross Tabulation and Table Creation (including empty classes)
Usage

```r
table.n(
  x,
  m = max(x, na.rm = TRUE),
  percentage = FALSE,
  weight = NULL,
  parallel = FALSE
)
```

Arguments

- `x`: R object with classes
- `m`: maximum number of classes
- `percentage`: boolean. If TRUE result is in percentages.
- `weight`: weight for each voxel
- `parallel`: Logical. Can we use parallel computing?

Value

vector with (weighted) counts (including empty classes)

Author(s)

Volker Schmid 2013-2016

Examples

```r
x <- c(1,1,2,2,4,4,4)
table.n(x)
# [1] 2 2 0 3
table.n(x, m=5)
# [1] 2 2 0 3 0
table.n(x, weight=c(1,1,2,.5,.5,.5))
# [1] 2.0 3.0 0.0 1.5
```

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

Value

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

Author(s)

Fabian Scheipl

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 = attributes(img)$bits.per.sample,
  twod = FALSE,
  reduce = TRUE,
  attr = attributes(img),
  compression = "none"
)

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.
reduce if TRUE then writeTIFF will attempt to reduce the number of planes in native rasters by analyzing the image to choose one of RGBA, RGB, GA or G formats, whichever uses the least planes without any loss. Otherwise the image is always saved with four planes (RGBA).
attr  Attributes of image stack. Will be propagated to each 2d image.
compression (see ?writeTIFF)