Package 'nandb'

July 11, 2017

Type Package

Title Number and Brightness Image Analysis

Version 0.2.1

Maintainer Rory Nolan <rorynoolan@gmail.com>

URL https://github.com/rorynolan/nandb

BugReports https://github.com/rorynolan/nandb/issues

Description Functions for calculation of molecular number and brightness from images, as detailed in Digman et al. 2008 <doi:10.1529/biophysj.107.114645>. Includes the implementation of the novel ``automatic detrending'' technique.

License BSD_3_clause + file LICENSE

LazyData TRUE

LinkingTo Rcpp, BH

Imports Rcpp, filesstrings (>= 1.1.0), magrittr, readr, reshape2, R.utils, autothresholdr, abind, viridis, dplyr, matrixStats, BiocParallel, stringr, EBImage, RSAGA, ggplot2, grDevices, RcppRoll, tibble, purrr, hexbin, ore, DescTools

RoxygenNote 6.0.1

Suggests knitr, rmarkdown, covr, testthat

VignetteBuilder knitr

NeedsCompilation yes

Author Rory Nolan [aut, cre, cph], Luis Alvarez [ctb], Sergi Padilla-Parra [ctb, ths, cph]

Repository CRAN

Date/Publication 2017-07-11 10:35:22 UTC

R topics documented:

`ApplyOnPillars` *Apply a function to each pillar of a 3-dimensional array.*

Description

Define a 'pillar' of a 3-dimensional array as pillar `i,j` off array `arr` being `arr[i, j, ]`. This function applies a specified function to each pillar.

Usage

```
ApplyOnPillars(mat3d, FUN)
```

Arguments

mat3d	A 3-dimensional array.
FUN	A function which takes a vector as imput and, for a given input length, outputs a vector of constant length (can be 1).

Value

If FUN is returning length 1 vectors, a matrix whereby mat[i, j] = FUN(mat3d[i, j,]). If FUN is returning vectors of length l > 1, a 3-dimensional array whereby arr[i, j,] = FUN(mat3d[i, j,]).

ArrArrHexPlot *Plot the values in two arrays against each other.*

Description

Plot the values of two arrays of identical dimension against each other using a hexagonal heatmap.

Usage

```
ArrArrHexPlot(arr.x, arr.y, bins = 60, log.trans = FALSE, colours = NULL,
  limits = NULL, breaks = NULL, include.breaks = NULL)
```

Arguments

arr.x, arr.y	The two arrays. The arrx values will be along the x axis and the arry values along the y axis.
bins	Passed to ggplot2::geom_hex().
log.trans	Do you want to log-transform the colour scaling?
colours	Here you may specify the colours (to be passed to ggplot2::scale_fill_gradientn()) to create the continuous colour band. It is specified as a character vector, with the colors specified either as the values in colors() or as in the value of the rgb() function. Note that this allows the use of grDevices::rainbow() and friends. The default uses viridis::viridis().
limits	A numeric vector of length two providing limits of the scale.
breaks	Where do you want tick marks to appear on the legend colour scale?
include.breaks	If you don't want to specify all the breaks, but you want some specific ones to be included on the legend colour scale, specify those specific ones here.

Value

This is a ggplot2 object and can be manipulated thus.

Examples

```
library(EBImage)
img <- ReadImageData(system.file('extdata', '50.tif', package = 'nandb'))
display(normalize(img[, , 1]), method = 'raster')
brightness <- Brightness(img, tau = NA, mst = "Huang")
mean.intensity <- MeanPillars(img)
ArrArrHexPlot(mean.intensity, brightness) +
ggplot2::labs(x = 'intensity', y = 'brightness')
```

`BestTau` *Find the best tau for exponential filtering detrend.*

Description

Say you have an image series that you wish to detrend before performing a brightness calculation. This function finds the best `tau` for an exponential filtering detrend. See `vignette('Adaptive Detrending',package = 'n` for more details.

Usage

```
BestTau(arr3d, mst = NULL, tol = 1)
```

Arguments

`arr3d`	A 3-dimensional array (the image stack) where the nth slice is the nth image in the time series. To perform this on a file that has not yet been read in, set this argument to the path to that file (a string).
`mst`	Do you want to apply an intensity threshold prior to calculating brightness (via `autothresholdr::mean_stack_thresh()`)? If so, set your thresholding *method* here.
`tol`	What size of error in the estimate of the *ideal* `tau` (aside from the error introduced by the random image simulation, see `vignette('AdaptiveDetrend', package = 'nandb')`) are you willing to tolerate? The default is 1.

Value

A number. The estimate of the ideal `tau` to use, with an attribute '`brightness.immobile`' giving the brightness of the simulated (from all immobile particles) image series after detrending with this `tau` (this should be very close to 1).

Examples

```
img <- ReadImageData(system.file('extdata', '50.tif', package = 'nandb'))
BestTau(img, tol = 3)
```

Bin2Tiff *Convert a binary image to tiff.*

Description

Read a binary image (pixels must be integers), specifying its dimension and write it back to disk as a tifff file. The `Bin2TiffFolder` function does this for an entire folder.

Usage

```
Bin2Tiff(bin.path, bits, height, width, frames = 1,
  endian = .Platform$endian, signed = TRUE)

Bin2TiffFolder(folder.path = ".", bits, height, width, frames = 1,
  endian = .Platform$endian, signed = TRUE)
```

Arguments

`bin.path`	The path to the binary file.
`bits`	How many bits are there per pixel? This should be a multiple of 8.
`height`	The height of the image (in pixels).
`width`	The width of the image (in pixels).
`frames`	How many frames are there in the stack (default 1).
`endian`	Eigher `"big"` or `"little"` (see readBin).
`signed`	Logical. Only used for integers of sizes 1 and 2, when it determines if the quantity on file should be regarded as a signed or unsigned integer.
`folder.path`	The path to the folder (defaults to current location).

Details

When using `Bin2TiffFolder`, the images must all have the same dimension.

Examples

```
setwd(tempdir())
dir.create("temp_dir")
file.copy(system.file("extdata", "b.bin", package = "nandb"), "temp_dir")
Bin2Tiff("temp_dir/b.bin", height = 2, width = 2, bits = 8)
Bin2TiffFolder("temp_dir", height = 2, width = 2, bits = 8)
list.files("temp_dir")
```

Brightness *Calculate brightness from image series.*

Description

Given a time stack of images, Brightness performs a calculation of the brightness for each pixel. BrightnessTxtFolder does this calculation for an entire folder, writing the results as text files via WriteImageTxt(). Brightnesses calculates the brightnesses for several image series in parallel.

Usage

```
Brightness(arr, tau = NA, mst = NULL, filt = NULL, n.ch = 1,
  verbose = FALSE)

BrightnessTxtFolder(folder.path = ".", tau = NA, mst = NULL,
  filt = NULL, ext = "tif", n.ch = 1, mcc = parallel::detectCores(),
  verbose = FALSE, seed = NULL)

Brightnesses(arr3d.list, tau = NA, mst = NULL, filt = NULL, n.ch = 1,
  verbose = FALSE, mcc = parallel::detectCores(), seed = NULL)
```

Arguments

arr	An array, can be 3- or 4-dimensional. The first two slots give the x- and y-cordinates of pixels respectively. If the array is 3-dimensional, the third slot gives the index of the frame. If it is 4-dimensional, the third slot indexes the channel and the fourth indexes the frame in the stack. To perform this on a file that has not yet been read in, set this argument to the path to that file (a string).
tau	If this is specified, bleaching correction is performed with CorrectForBleaching() which uses exponential filtering with time constant tau (where the unit of time is the time between frames). If this is set to 'auto', then the value of tau is calculated automatically via BestTau().
mst	Do you want to apply an intensity threshold prior to calculating brightness (via autothresholdr::mean_stack_thresh())? If so, set your thresholding *method* here.
filt	Do you want to smooth (filt = 'smooth') or median (filt = 'median') filter the brightness image using SmoothFilterB() or MedianFilterB() respectively? If selected, these are invoked here with a filter radius of 1 and with the option na_count = TRUE. If you want to smooth/median filter the brightness image in a different way, first calculate the brightnesses without filtering (filt = NULL) using this function and then perform your desired filtering routine on the result.
n.ch	The number of channels in the image (default 1).
verbose	If arr3d is specified as a file name, print a message to tell the user that that file is now being processed (useful for BrightnessTxtFolder, does not work with multiple cores).

folder.path	The path (relative or absolute) to the folder you wish to process.
ext	The file extension of the images in the folder that you wish to process. You must wish to process all files with this extension; if there are files that you don't want to process, take them out of the folder. The default is for tiff files. Do not use regular expression in this argument.
mcc	The number of cores to use for the parallel processing.
seed	If using parallel processing (mcc > 1), a seed for the random number generation for BestTau. Don't use set.seed, it won't work.
arr3d.list	A list of 3-dimensional arrays. To perform this on files that have not yet been read in, set this argument to the path to these files (a character vector).

Details

Do not try to parallelize the use of Brightness and friends yourself (e.g. with mclapply()) because it will throw an error (this is because the autothresholdr package does not work in parallel (indeed anything run using the rJava package won't)). Always use Brightnesses for this purpose (this has a workaround whereby it does the thresholding in series and does the rest in parallel).

Value

Brightness returns a matrix, the brightness image; Brightnesses returns a list of these. The result of BrightnessTxtFolder is the text csv files written to disk (in the same folder as the input images).

Examples

```
library(magrittr)
img <- ReadImageData(system.file('extdata', '50.tif', package = 'nandb'))
EBImage::display(EBImage::normalize(img[, , 1]), method = 'raster')
brightness <- Brightness(img, tau = "auto", mst = 'Huang', filt = 'median')
MatrixRasterPlot(brightness, log.trans = TRUE)
two.channel.img <- abind::abind(img, img, along = 4) %>% aperm(c(1, 2, 4, 3))
brightness.2ch <- Brightness(two.channel.img, n.ch = 2)
setwd(tempdir())
WriteIntImage(img, '50.tif')
WriteIntImage(img, '50again.tif')
BrightnessTxtFolder(tau = NA, mcc = 2)
list.files()
file.remove(list.files())  # cleanup
img.paths <- rep(system.file('extdata', '50.tif', package = 'nandb'), 2)
brightnesses <- Brightnesses(img.paths, tau = NA, mcc = 2)
```

`BrightnessPlotFolder` *Make brightness plots (images) for an entire folder.*

Description

This requires that the folder already have the brightnesses saved as csv files. Output images are saved as pdf.

Usage

```
BrightnessPlotFolder(folder.path = ".", patt = ".*[Bb]rightness.*\\.csv$",
  verbose = TRUE, ...)
```

Arguments

`folder.path`	The path (relative or absolute) to the folder you wish to process.
`patt`	The pattern (in regular expression) of the csv files in the folder that are brightness images. The default matches any .csv files which contains 'Brightness' or 'brightness' in its name.
`verbose`	Get a real time report on progress?
`...`	Parameters to pass to `MatrixRasterPlot()` (these *must* be named arguments).

Value

This is a `ggplot2` object and can be manipulated thus.

Examples

```
setwd(tempdir())
img <- ReadImageData(system.file('extdata', '50.tif', package = 'nandb'))
WriteIntImage(img, '50.tif')
WriteIntImage(img, '50again.tif')
BrightnessTxtFolder(tau = NA, mst = "Huang", mcc = 2)
BrightnessPlotFolder()
list.files()
file.remove(list.files())  # cleanup
```

`BrightnessTimeSeries` *Create a Brightness time-series.*

Description

Given a stack of images `arr`, use the first `frames.per.set` of them to create one brightness image, the next `frames.per.set` of them to create the next brightness image and so on to get a time-series of brightness images. If `tau` is specified, bleaching correction is performed via `CorrectForBleaching()`.

Usage

```
BrightnessTimeSeries(arr, frames.per.set, tau = NA, mst = NULL,
  filt = NULL, n.ch = 1, verbose = FALSE, mcc = parallel::detectCores(),
  seed = NULL)

BrightnessTimeSeriesTxtFolder(folder.path = ".", frames.per.set,
  ext = "tif", tau = NA, mst = NULL, filt = NULL, n.ch = 1,
  verbose = FALSE, mcc = parallel::detectCores(), seed = NULL)
```

Arguments

`arr`	An array, can be 3- or 4-dimensional. The first two slots give the x- and y-cordinates of pixels respectively. If the array is 3-dimensional, the third slot gives the index of the frame. If it is 4-dimensional, the third slot indexes the channel and the fourth indexes the frame in the stack. To perform this on a file that has not yet been read in, set this argument to the path to that file (a string).
`frames.per.set`	The number of frames with which to calculate the successive brightnesses.
`tau`	If this is specified, bleaching correction is performed with `CorrectForBleaching()` which uses exponential filtering with time constant `tau` (where the unit of time is the time between frames). If this is set to `'auto'`, then the value of `tau` is calculated automatically via `BestTau()`.
`mst`	Do you want to apply an intensity threshold prior to calculating brightness (via `autothresholdr::mean_stack_thresh()`)? If so, set your thresholding *method* here.
`filt`	Do you want to smooth (`filt = 'smooth'`) or median (`filt = 'median'`) filter the brightness image using `SmoothFilterB()` or `MedianFilterB()` respectively? If selected, these are invoked here with a filter radius of 1 and with the option `na_count = TRUE`. If you want to smooth/median filter the brightness image in a different way, first calculate the brightnesses without filtering (`filt = NULL`) using this function and then perform your desired filtering routine on the result.
`n.ch`	The number of channels in the image (default 1).
`verbose`	If arr3d is specified as a file name, print a message to tell the user that that file is now being processed (useful for `BrightnessTxtFolder`, does not work with multiple cores).

mcc	The number of cores to use for the parallel processing.
seed	If using parallel processing (mcc > 1), a seed for the random number generation for BestTau. Don't use set.seed, it won't work.
folder.path	The path to the folder to process (defaults to current location).
ext	The file extension of the images in the folder that you wish to process. You must wish to process all files with this extension; if there are files that you don't want to process, take them out of the folder. The default is for tiff files. Do not use regular expression in this argument.

Details

This may discard some images, for example if 175 frames are in the input and `frames.per.set = 50`, then the last 25 are discarded. If bleaching or/and thresholding are selected, they are performed on the whole image stack before the sectioning is done for calculation of brightnesses.

Value

An array where the ith slice is the ith brightness image.

See Also

`Brightness()`.

Examples

```
library(magrittr)
img <- ReadImageData(system.file('extdata', '50.tif', package = 'nandb'))
EBImage::display(EBImage::normalize(img[, , 1]), method = 'raster')
bts <- BrightnessTimeSeries(img, 20, tau = NA, mst = "Huang", mcc = 2)
setwd(tempdir())
WriteIntImage(img, '50.tif')
WriteIntImage(img, '50again.tif')
BrightnessTimeSeriesTxtFolder(tau = NA, mcc = 2, frames.per.set = 20)
list.files()
file.remove(list.files())  # cleanup
```

Closest *What's the closest value in a vector?*

Description

Given a number x and a numeric vector vec, what's the closest value to x in vec?

Usage

```
Closest(x, vec, index = FALSE)
```

Arguments

x	A number.
vec	A numeric vector.
index	If set to TRUE, return the index (rather than the value) of the closest element.

Value

A number.

Examples

```
Closest(pi, 0:10)
Closest(pi, 0:10, index = TRUE)
```

CollapseRanges *Collapse a big set of ranges into a smaller set.*

Description

Say you have many ranges (or bins) in which you're assigning continuous values and you'd like to collapse these ranges such that there are fewer of them (but they still cover the same part of that continuous scale). This function is here to help.

Usage

```
CollapseRanges(ranges, n.out, preserve = NULL, prefer.low = FALSE,
  prefer.high = FALSE)
```

Arguments

ranges	A strictly increasing (numeric) vector. Each set of adjacent elements are interpreted as the bounds of a range (or bin).
n.out	A natural number. How many ranges should the output have?
preserve	A vector. Are there any original ranges that you'd like to preserve? If so set them here. The first range is the interval [ranges[1], ranges[2]) and so on.
prefer.low	Are you more interested in the lower ranges? If so, set this to true and all the high ranges will be collapsed into one.
prefer.high	Are you more interested in the higher ranges? If so, set this to true and all the high ranges will be collapsed into one.

Details

One property of this procedure is that each new range is the union of old ranges.

Value

A vector of the new ranges.

Examples

```
set.seed(0)
ranges <- sort(sample(1:100, 11))
print(ranges)
CollapseRanges(ranges, 6, c(3, 7))
CollapseRanges(ranges, 6, c(3, 7), prefer.low = TRUE)
CollapseRanges(ranges, 6, c(3, 7), prefer.high = TRUE)
```

CorrectForBleaching *Detrend an image series*

Description

`CorrectForBleaching` applies detrending to an image time series using the method described in Nolan et al. 2017. `CorrectForBleachingFolder` performs this correction on all images in a folder, writing the corrected images to disk.

Usage

```
CorrectForBleaching(arr, tau, n.ch = 1)

CorrectForBleachingFolder(folder.path = ".", tau = NA, mst = NULL,
  ext = "tif", na = "error", mcc = parallel::detectCores(), seed = NULL)
```

Arguments

`arr`	An array, can be 3- or 4-dimensional. The first two slots give the x- and y-cordinates of pixels respectively. If the array is 3-dimensional, the third slot gives the index of the frame. If it is 4-dimensional, the third slot indexes the channel and the fourth indexes the frame in the stack.. To perform this on a file that has not yet been read in, set this argument to the path to that file (a string).
`tau`	The time constant for the exponential filtering. If this is set to `'auto'`, then the value of `tau` is calculated automatically via `BestTau()`.
`n.ch`	The number of channels in the image (default 1).
`folder.path`	The path (relative or absolute) to the folder you wish to process.
`mst`	Do you want to apply an intensity threshold prior to correcting for bleaching (via `autothresholdr::mean_stack_thresh()`)? If so, set your thresholding *method* here.
`ext`	The file extension of the images in the folder that you wish to process. You must wish to process all files with this extension; if there are files that you don't want to process, take them out of the folder. The default is for tiff files. Do not use regular expression in this argument.

na	How do you want to treat NA values? R can only write integer values (and hence not NAs) to tiff pixels. na = 'saturate' sets them to saturated value. na = 'zero' sets them to zero, while na = 'error' will give an error if the image contains NAs. Note that if you threshold, you are almost certain to get NAs.
mcc	The number of cores to use for the parallel processing.
seed	A seed for the random number generation for BestTau. Don't use set.seed, it won't work.

Details

If you wish to apply thresholding and bleaching correction, aplpluy the thresholding first. CorrectforBleachingFolder takes care of this for you.

Value

CorrectForBleaching returns the detrended image series.

References

Stroud, P. D.: A recursive exponential filter for time-sensitive data, Los Alamos national Laboratory, LAUR-99-5573, public.lanl.gov/stroud/ExpFilter/ExpFilter995573.pdf, 1999.

Examples

```
library(magrittr)
img <- ReadImageData(system.file("extdata", "50.tif", package = "nandb"))
autotau <- CorrectForBleaching(img, "auto")
setwd(tempdir())
img <- ReadImageData(system.file('extdata', '50.tif', package = 'nandb'))
WriteIntImage(img, '50.tif')
WriteIntImage(img, '50again.tif')
set.seed(33)
CorrectForBleachingFolder(tau = 1000, mst = "Huang", mcc = 2, na = "s")
list.files()
file.remove(list.files())
```

ExpSmooth *Exponentially smooth a series of observations.*

Description

This function assumes that the observations are evenly spaced and separated by 1 time unit (so choose your tau based on that).

Usage

```
ExpSmooth(obs, tau, extended = FALSE)
```

Arguments

obs	A numeric vector of observations (in order).
tau	The time scale for the exponential smoothing (see Stroud 1999).
extended	Logical. Has the series (obs) already been extended via via nandb:::MedReflectExtend()? If not, ExpSmooth will do this prior to smoothing as an edge-correction technique. You will probably never set this to TRUE, but BestTau() needs this feature.

Value

The smoothed series, a numeric vector of the same length.

Examples

```
ExpSmooth(1:10, 1)
```

ExpSmoothPillars *Exponentially smooth pillars of a 3-dimensional array*

Description

For a 3-dimensional array mat3d, pillar i,j is defined as mat3d[i, j,]. ExpSmoothPillars function performs ExpSmooth on each pillar. ExpSmoothRows performs ExpSmooth on each row of a matrix.

Usage

```
ExpSmoothPillars(mat3d, tau)

ExpSmoothRows(mat, tau, extended = FALSE)
```

Arguments

mat3d	A 3-dimensional array.
tau	The time scale for the exponential smoothing (see Stroud 1999).
mat	A matrix.
extended	Logical. Has the series (obs) already been extended via via nandb:::MedReflectExtend()? If not, ExpSmooth will do this prior to smoothing as an edge-correction technique. You will probably never set this to TRUE, but BestTau() needs this feature.

Value

For ExpSmoothPillars, a 3-dimensional array where each pillar has been smoothed. For ExpSmoothRows, a matrix where each row has been smoothed.

Examples

```
m3 <- array(1:12, dim = c(2, 2, 3))
ExpSmoothPillars(m3, 7)
```

`FixLUTError` *Fix lookup table error when reading images.*

Description

Even if an image stored on disk has 1 channel only, if it has an associated LUT (lookup table, to tell programs like ImageJ to display them in (for example) green rather than grey, then `EBImage::readImage()` and hence `ReadImageData()` will read in the image as a 3-channel rgb colour image, where two of the channels have all-zero intensity values and one of them has the values you wanted. This function deletes the zero channels from the array. If there were no such LUT errors and the image read in as you desired, then this function does nothing.

Usage

```
FixLUTError(arr, ndim.out)
```

Arguments

`arr`	An array, representing the read image.
`ndim.out`	How many dimensions do you want the output array to have?

Value

An array. If the function implemented a 'fix', then the output array will have one less dimension than the input array.

Examples

```
has.lut.error <- abind::abind(matrix(0, nrow = 2, ncol = 2),
                              matrix(1:4, nrow = 2),
                              matrix(0, nrow = 2, ncol = 2),
                              along = 3)
has.lut.error
FixLUTError(has.lut.error, 2)
has.lut.error3d <- abind::abind(has.lut.error, has.lut.error, along = 4)
FixLUTError(has.lut.error3d, 3)
```

ForceChannels *Fix an image that didn't recognise channels while reading*

Description

Sometimes, when you open an image in ImageJ, it displays the channels as you would like, but when you read it into R, it has just mashed all the channels (which you would like to be separated somehow) into a stack. In my expreience, it always does so in a way that, say you have a stack of 3 channels and 5 z positions, then the red images would occupy `[, , 1]`, `[, , 6]` and `[, , 11]`. This function fixes this kind of confusion. So, in that example it would have a 3d array as input and a 4d as output with dimensions (assuming our images are 256x256 pixels) `256, 256, 3, 5`.

Usage

```
ForceChannels(img.arr, n.ch)
```

Arguments

`img.arr`	An array, the read image.
`n.ch`	The number of channels that you want the read image to have.

Value

An array, channel indices in the third slot, slice indices in the fourth.

Examples

```
library(magrittr)
x <- lapply(1:300, function(x) matrix(runif(4), nrow = 2)) %>%
  Reduce(function(x, y) abind::abind(x, y, along = 3), .)
str(x)
ForceChannels(x, 6) %>% str
```

GroupClose *Group together close adjacent elements of a vector.*

Description

Given a strictly increasing vector (each element is bigger than the last), group together stretches of the vector where *adjacent* elements are separeted by at most some specified distance. Hence, each element in each group has at least one other element in that group that is *close* to it. See the examples.

Usage

```
GroupClose(vec.ascending, max.gap = 1)
```

Arguments

vec.ascending	A strictly increasing numeric vector.
max.gap	The biggest allowable gap between adjacent elements for them to be considered part of the same *group*.

Value

A where each element is one group, as a numeric vector.

Examples

```
GroupClose(1:10, 1)
GroupClose(1:10, 0.5)
GroupClose(c(1, 2, 4, 10, 11, 14, 20, 25, 27), 3)
```

KmerArray *Get an image where each pixel represents a kmer.*

Description

Based on a brightness image (array, can be more than two-dimensional), create a kmer image (array) where each pixel represents a kmer (0 for immobile, 1 for monomer, 2 for dimer and so on).

Usage

```
KmerArray(brightness.arr, monomer)
```

Arguments

brightness.arr	The brightness array.
monomer	The (median) brightness of a monomer.

Value

A matrix.

Examples

```
img <- ReadImageData(system.file('extdata', '50.tif', package = 'nandb'))
brightness <- Brightness(img, tau = NA, mst = 'Huang', filt = 'median')
ka <- KmerArray(brightness, 1.1)
```

KmerPlot *A brightness image with a different colour for each kmer.*

Description

Make a colour image based on a brightness image where each kmer has its own colour. This requires you te specify the brightness of a monomer (which should be greater than 1).

Usage

```
KmerPlot(brightness.mat, monomer.brightness, log.trans = FALSE)
```

Arguments

brightness.mat	The brightness matrix.
monomer.brightness	The (median) brightness of a monomer.
log.trans	Do you want to log-transform the colour scaling?
	This is a ggplot2 object and can be manipulated thus.

Examples

```
library(EBImage)
img <- ReadImageData(system.file('extdata', '50.tif', package = 'nandb'))
display(normalize(img[, , 1]), method = 'raster')
brightness <- Brightness(img, tau = NA, mst = "Huang")
KmerPlot(brightness, 1.02)
KmerPlot(brightness, 1.12)
KmerPlot(brightness, 100)
KmerPlot(brightness, 1.02, log.trans = TRUE)
KmerPlot(MedianFilterB(brightness), 1.02)
```

KmersFromBrightnesses *Calculate numbers of kmers based on brightnesses*

Description

Calculate the number pixels occupied of molecules in each oligomeric state (monomer, dimer, trimer, ..., kmer, ...) based on the brightnesses of those pixels.

Usage

```
KmersFromBrightnesses(brightnesses, monomer)
```

Arguments

brightnesses	Vector or array of pixel brightnesses
monomer	The median brightness of a monomer. You must know what this is to use this function correctly. This must be greater than 1 (this is the 'apparent brightness' of a monomer). If you're reading the Digman et al. (2008) paper, this is $1 + \epsilon$. This function takes the brightnesses in the range (1, 1 + 2 * monomer to be monomers, those in the range (1 + 2 * monomer, 1 + 4 * monomer to be dimers, (1 + 4 * monomer, 1 + 6 * monomer to be trimers and so on.

Value

A named vector (named '1mers' '2mers' '3mers' and so on) with each element detailing the number of that kmer found.

Examples

```
brightnesses <- runif(100, 1, 3)
monomer <- 1.2
KmersFromBrightnesses(brightnesses, monomer)
```

KmersFromImage *Calculate numbers of kmers based on an image time-series*

Description

Given an image (as a file path or an array), KmersFromImage does the brightness calculation via Brightness() and then counts the numbers of each kmer via KmersFromBrightnesses(). KmersFromImagesFolder does this for an entire folder (directory) of images and outputs a csv file of the results.

Usage

```
KmersFromImage(arr3d, monomer, tau = NA, mst = NULL, filt = NULL,
  verbose = TRUE)

KmersFromImages(arrs3d, monomer, tau = NA, mst = NULL, filt = NULL,
  verbose = TRUE, mcc = parallel::detectCores(), seed = NULL)

KmersFromImagesFolder(folder.path = ".", monomer, tau = NA, mst = NULL,
  filt = NULL, out.name = "results", ext = "\\.tif$",
  mcc = parallel::detectCores(), seed = NULL, verbose = TRUE)
```

Arguments

arr3d	A 3-dimensional array that one would might input to Brightness() *or* the path to an image file on disk.
monomer	The median brightness of a monomer. You must know what this is to use this function correctly. This must be greater than 1 (this is the 'apparent brightness' of a monomer). If you're reading the Digman et al. (2008) paper, this is $1 + \epsilon$.
tau	The time constant for the exponential filtering (see Brightness()).
mst	Do you want to apply an intensity threshold prior to calculating brightness (via autothresholdr::mean_stack_thresh())? If so, set your thresholding *method* here.
filt	Do you want to smooth (filt = 'smooth') or median (filt = 'median') filter the brightness image using SmoothFilterB() or MedianFilterB() respectively? If selected, these are invoked here with a filter radius of 1 and with the option na_count = TRUE. If you want to smooth/median filter the brightness image in a different way, first calculate the brightnesses without filtering (filt = NULL) using this function and then perform your desired filtering routine on the result.
verbose	If arr3d is specified as a file name, print a message to tell the user that that file is now being processed (useful for BrightnessFolder, does not work with multiple cores) and to tell when KmersFromImagesFolder is done.
arrs3d	A list of 3d arrays. To perform this on files that have not yet been read in, set this argument to the path to these files (a character vector).
mcc	The number of cores to use for the parallel processing.
seed	A seed for the random number generation for BestTau. Don't use set.seed, it won't work.
folder.path	The path (relative or absolute) to the folder you wish to process.
out.name	The name of the results csv file.
ext	the file extension of the images in the folder that you wish to process (can be rooted in regular expression for extra-safety, as in the default). You must wish to process all files with this extension; if there are files that you don't want to process, take them out of the folder.

Value

A named vector (named '1mers' '2mers' '3mers' and so on) with each element detailing the number of that kmer found, or for KmersFromImagesFolder, a csv file is written to disk detailing one of these vectors for each image. This vector also has an attribute 'mean.intensity' giving the mean intensity of the input image.

Examples

```
img <- system.file('extdata', '50.tif', package = 'nandb')
KmersFromImage(img, 1.1, tau = NA, mst = "huang")

KmersFromImages(c(img, img), 1.1, tau = NA, mst = "huang")
```

```
setwd(tempdir())
file.copy(img, ".")
KmersFromImagesFolder(monomer = 1.11)
file.remove(list.files())  # cleanup
```

KmerTIFFsFromBrightnessCSVs

Create kmer tiff files from brightness csvs

Description

For each brightness csv image in a folder, given a monomeric brightness, create a tiff file of the kmer positions using KmerArray().

Usage

```
KmerTIFFsFromBrightnessCSVs(monomer, csv.paths = NULL, out.names = NULL,
  na = "saturate", mcc = parallel::detectCores(), verbose = TRUE)
```

Arguments

monomer	The (median) brightness of a monomer.
csv.paths	The paths to the brightness csv files, defaults to list.files(pattern = '[Bb]rightness.*\.csv').
out.names	The names you want the output files to have (will be forced to .tif).
na	See WriteIntImage().
mcc	The number of parallel cores to use for the processing.
verbose	Do you want to print a message when the function is done?

Examples

```
img <- system.file('extdata', '50.tif', package = 'nandb')
setwd(tempdir())
file.copy(img, ".")
BrightnessTxtFolder()
KmerTIFFsFromBrightnessCSVs(1.111)
file.remove(list.files())  # cleanup
```

ListPillars *Turn a 3d array into a list of pillars*

Description

Suppose the array is of dimension `n1 * n2 * n3`, then pillar `[i, j, ]` is list element `i + n1 * (j - 1)`, so the first element is pillar `[1, 1, ]`, the second is pillar `[1, 2, ]` and so on.

Usage

```
ListPillars(arr3d)
```

Arguments

`arr3d`	A 3-dimensional array.

Value

A list.

See Also

PillarsListToArr

Examples

```
arr <- array(1:27, dim = rep(3, 3))
print(arr)
ListPillars(arr)
```

MatrixRasterPlot *Make a raster plot of a matrix.*

Description

Given a matrix `mat`, make a raster plot of the matrix whereby in the plot, the pixel at $x =$`i`, $y =$`j` has colour based on the value of `mat[i, j]` and the x axis points right and the y axis points down (see 'Details').

Usage

```
MatrixRasterPlot(mat, scale.name = "scale", limits = NULL, ranges = NULL,
  range.names = NULL, colours = NULL, na.colour = "black", clip = FALSE,
  clip.low = FALSE, clip.high = FALSE, log.trans = FALSE, breaks = NULL,
  include.breaks = NULL)
```

Arguments

mat	The matrix you wish to plot.
scale.name	A string. The title of the color scale on the right of the plot.
limits	This gives the user the option to set all values outside a certain range to their nearest value within that range (if clip = TRUE) or to NA (if clip = FALSE. For example, to set all values outside the range [1.5, 2.6) to NA, use limits = c(1.5, 2.6), clip = FAL The colour range will cover all values within these specified limits.
ranges	A numeric vector. If you want specific ranges of values to have the same color, specify these ranges via an increasing numeric vector. For example, if you want the ranges 0.5-1.2 and 1.2-3, use ranges = c(0.5, 1.2, 3). If ranges is specified as a number (a numeric vector of length 1) n, this is equivalent to setting ranges to be n equal-length intervals within the range of the matrix, i.e. it is equivalent to setting ranges = seq(min(mat), max(mat), length.out = n + 1). At most one of ranges and limits should be set. If ranges is set, the behaviour for values which are not in any of the ranges are set by the clip arguments as in the limits argument.
range.names	A character vector. If your colour scale is discrete, here you can set the names which will label each range in the legend.
colours	If you have set ranges, here you may specify which colors you wish to colour each range. It must have the same length as the number of intervals you have specified in ranges. If you have not specified ranges, here you may specify the colours (to be passed to ggplot2::scale_fill_gradientn()) to create the continuous colour band. It is specified as a character vector, with the colors specified either as the values in colors() or as in the value of the rgb() function. Note that this allows the use of grDevices::rainbow() and friends. The default uses viridis::viridis().
na.colour	Which colour should the NA pixels be? Default is black.
clip	If either limits or ranges are set (one should never set both), there may be values that fall outside the specified limits/ranges. If clip = TRUE, values outside these limits/ranges are set to their nearest values within them, but if clip = FALSE, these values are set to NA. Note that setting clip = TRUE is equivalent to setting both clip.low and clip.high to TRUE.
clip.low	Setting this to TRUE (and leaving clip = FALSE, clip.high = FALSE) will set all values falling below the specified limits/ranges to their nearest value within them, but all values falling above those limits/ranges will be set to NA.
clip.high	Setting this to TRUE (and leaving clip = FALSE, clip.low = FALSE) will set all values falling above the specified limits/ranges to their nearest value within them, but all values falling below those limits/ranges will be set to NA.
log.trans	Do you want to log-transform the colour scaling?
breaks	Where do you want tick marks to appear on the legend colour scale?
include.breaks	If you don't want to specify all the breaks, but you want some specific ones to be included on the legend colour scale, specify those specific ones here.

Details

The pixel at x =i, y =j has colour based on the value of mat[i, j] where the x axis points right and the y axis points down. This is in accordance with how EBImage::EBImage() and ReadImageData() (which wraps EBImage's EBImage::readImage()). However, when one prints a matrix in a console (or views it in a program such as excel), the value in position x =i, y =j is from mat[j, i], so if you're confused about a transposed plot, this is why.

Value

In the graphics console, a raster plot (via ggplot2::geom_raster()) will appear with the matrix values represented as pixel colours, with a named scale bar.

Examples

```
library(EBImage)
img <- ReadImageData(system.file('extdata', '50.tif', package = 'nandb'))
display(normalize(img[, , 1]), method = 'raster')
brightness <- Brightness(img, tau = NA, mst = "Huang")
MatrixRasterPlot(brightness, scale.name = 'brightness')
MatrixRasterPlot(brightness, scale.name = 'brightness', log.trans = TRUE)
MatrixRasterPlot(brightness, scale.name = 'brightness', log.trans = TRUE,
                 include.breaks = 1.35)
MatrixRasterPlot(brightness, scale.name = 'brightness', log.trans = TRUE,
                 breaks = 1:3)
MatrixRasterPlot(brightness, scale.name = 'brightness',
  ranges = seq(0.5, 3, length.out = 6), range.names = paste0(1:5, 'mer'))
MatrixRasterPlot(brightness, scale.name = "brightness",
                 ranges = seq(0.5, 3, length.out = 6),
                 range.names = paste0(1:5, "mer"), log.trans = TRUE)
MatrixRasterPlot(brightness, scale.name = "brightness",
  include.breaks = 1.25, range.names = NULL, log.trans = FALSE)
MatrixRasterPlot(brightness, scale.name = "brightness",
                 include.breaks = 1.25, log.trans = TRUE)
MatrixRasterPlot(brightness, scale.name = "brightness",
                 limits = c(1, 1.25), clip = TRUE)
MatrixRasterPlot(brightness, scale.name = "brightness",
                 include.breaks = 1.25)
```

MeanIntensity *Calculate mean intensity from image series.*

Description

Given a time stack of images, MeanIntensity calculates the mean intensity for each pixel, returning a matrix. MeanIntensityTxtFolder does this for every image in a folder, writing the results as text files via WriteImageTxt().

Usage

```
MeanIntensity(arr, n.ch = 1, mst = NULL, filt = NULL,
  skip.consts = FALSE, verbose = FALSE)

MeanIntensityTxtFolder(folder.path = ".", mst = NULL, ext = "tif",
  mcc = parallel::detectCores(), verbose = TRUE)

MeanIntensities(arr3d.list, mst = NULL, skip.consts = FALSE, fail = NA,
  filt = NULL, verbose = FALSE, mcc = parallel::detectCores())
```

Arguments

arr	An array, can be 3- or 4-dimensional. The first two slots give the x- and y-cordinates of pixels respectively. If the array is 3-dimensional, the third slot gives the index of the frame. If it is 4-dimensional, the third slot indexes the channel and the fourth indexes the frame in the stack.. To perform this on a file that has not yet been read in, set this argument to the path to that file (a string).
n.ch	The number of channels in the image (default 1).
mst	Do you want to apply an intensity threshold prior to calculating mean intensities (via autothresholdr::mean_stack_thresh())? If so, set your thresholding *method* here. Pixels failing to exceed the threshold are set to NA.
filt	Do you want to smooth (filt = 'smooth') or median (filt = 'median') filter the mean intensity image using SmoothFilterB() or MedianFilterB() respectively? If selected, these are invoked here with a filter radius of 1 and with the option na_count = TRUE. If you want to smooth/median filter the mean intensity image in a different way, first calculate the mean intensities without filtering (filt = NULL) using this function and then perform your desired filtering routine on the result.
skip.consts	An image array with only one value (a 'constant array') won't threshold properly. By default the function would give an error, but by setting this parameter to TRUE, the array would instead be skipped (the function will return the original array) and give a warning.
verbose	If arr is specified as a file name, print a message to tell the user that that file is now being processed (useful for MeanIntensityTxtFolder, does not work with multiple cores) and to tell when MeanIntensityTxtFolder is done.
folder.path	The path (relative or absolute) to the folder you wish to process.
ext	The file extension of the images in the folder that you wish to process. You must wish to process all files with this extension; if there are files that you don't want to process, take them out of the folder. The default is for tiff files. Do not use regular expression in this argument.
mcc	The number of parallel cores to use for the processing.
arr3d.list	A list of 3-dimensional arrays. To perform this on files that have not yet been read in, set this argument to the path to these files (a character vector).
fail	If thresholding is done, to which value should pixels not exceeeding the threshold be set?

Value

`MeanIntensity` returns a matrix, the mean-intensity image. `MeanIntensities` returns a list of these. The result of `MeanIntensityTxtFolder` is csv files written to disk (in the same folder as the input images).

Examples

```
library(EBImage)
library(magrittr)
img <- ReadImageData(system.file('extdata', '50.tif', package = 'nandb'))
display(normalize(img[, , 1]), method = 'raster')
mean.intensity <- MeanIntensity(img, mst = 'Huang', filt = 'median')
display(normalize(mean.intensity), method = 'r')
two.channel.img <- abind::abind(img, img, along = 4) %>% aperm(c(1, 2, 4, 3))
mint.2ch <- MeanIntensity(two.channel.img, mst = "h", filt = "med", n.ch = 2)
setwd(tempdir())
img <- ReadImageData(system.file('extdata', '50.tif', package = 'nandb'))
WriteIntImage(img, '50.tif')
WriteIntImage(img, '50again.tif')
MeanIntensityTxtFolder(mcc = 2)

img.paths <- rep(system.file('extdata', '50.tif', package = 'nandb'), 2)
mean.intensities <- MeanIntensities(img.paths, mst = 'Huang', mcc = 2)
```

MeanPillars *Get the means/medians/variances of pillars of a 3d array*

Description

For a 3-dimensional array `mat3d`, pillar `ij` is defined as `mat3d[i, j, ]`. These functions compute the mean, median and variance of each pillar.

Usage

```
MeanPillars(mat3d)

VarPillars(mat3d)

MedianPillars(mat3d)
```

Arguments

`mat3d`	A 3-dimensional array.

Value

A matrix where element `i,j` is equal to `mean(mat3d[i, j, ])`, `median(mat3d[i, j, ])`, or `var(mat3d[i, j, ])`.

Examples

```
m3 <- array(1:16, dim = c(2, 2, 4))
MeanPillars(m3)
MedianPillars(m3)
VarPillars(m3)
```

`MedianFilterB` *Smooth and median filters with options for handling NAs.*

Description

These is an alternative to EBImage's filter2 and medianFilter for smooth and median filtering respectively. These functions have many options for dealing with `NA` values which `EBImage`'s functions lack.

Usage

```
MedianFilterB(mat, size = 1L, na_rm = FALSE, na_count = FALSE)

SmoothFilterB(mat, size = 1L, na_rm = FALSE, na_count = FALSE)
```

Arguments

`mat`	A matrix (representing an image).
`size`	An integer; the median filter radius.
`na_rm`	Should `NA`s be ignored?
`na_count`	If this is TRUE, in each median calculation, if the majority of arguments are `NA`s, `NA` is returned but if the `NA`s are in the minority, they are ignored as in `median(x, na.rm = TRUE)`.

Details

The behavior at image boundaries is such as the source image has been padded with pixels whose values equal the nearest border pixel value.

Value

A matrix (the median filtered image).

Examples

```
m <- matrix(1:9, nrow = 3)
m[2:3, 2:3] <- NA
print(m)
MedianFilterB(m)
MedianFilterB(m, na_rm = TRUE)
MedianFilterB(m, na_count = TRUE)

MedianFilterB(m)
MedianFilterB(m, na_rm = TRUE)
MedianFilterB(m, na_count = TRUE)
```

---

nandb *nandb: Number and brightness in R.*

Description

The nandb package gives functions for calculation of molecular number and brightness from images, as detailed in Digman et al. 2008. It comes with an implementation of the novel 'automatic detrending' technique.

References

Digman MA, Dalal R, Horwitz AF, Gratton E. Mapping the Number of Molecules and Brightness in the Laser Scanning Microscope. Biophysical Journal. 2008;94(6):2320-2332. doi: 10.1529/biophysj.107.114645.

Number *Calculate number from image series.*

Description

Given a time stack of images, Number performs a calculation of the number for each pixel. NumberTxtFolder does this calculation for an entire folder, writing the results as text files via WriteImageTxt(). Numbers calculates the numbers for several image series in parallel.

Usage

```
Number(arr, tau = NA, mst = NULL, filt = NULL, n.ch = 1,
  verbose = FALSE)

NumberTxtFolder(folder.path = ".", tau = NA, mst = NULL, filt = NULL,
  ext = "tif", mcc = parallel::detectCores(), seed = NULL,
  verbose = FALSE)

Numbers(arr3d.list, tau = NA, mst = NULL, fail = NA, filt = NULL,
  n.ch = 1, verbose = FALSE, mcc = parallel::detectCores(), seed = NULL)
```

Arguments

arr	A 3-dimensional array (the image stack) where the nth slice is the nth image in the time series. To perform this on a file that has not yet been read in, set this argument to the path to that file (a string).
tau	If this is specified, bleaching correction is performed with CorrectForBleaching() which uses exponential filtering with time constant tau (where the unit of time is the time between frames). If this is set to 'auto', then the value of tau is calculated automatically via BestTau().
mst	Do you want to apply an intensity threshold prior to calculating number (via autothresholdr::mean_stack_thresh())? If so, set your thresholding *method* here.
filt	Do you want to smooth (filt = 'smooth') or median (filt = 'median') filter the number image using SmoothFilterB() or MedianFilterB() respectively? If selected, these are invoked here with a filter radius of 1 and with the option na_count = TRUE. If you want to smooth/median filter the number image in a different way, first calculate the numbers without filtering (filt = NULL) using this function and then perform your desired filtering routine on the result.
n.ch	The number of channels in the image (default 1).
verbose	If arr3d is specified as a file name, print a message to tell the user that that file is now being processed (useful for NumberTxtFolder, does not work with multiple cores).
folder.path	The path (relative or absolute) to the folder you wish to process.
ext	The file extension of the images in the folder that you wish to process. You must wish to process all files with this extension; if there are files that you don't want to process, take them out of the folder. The default is for tiff files. Do not use regular expression in this argument.
mcc	The number of cores to use for the parallel processing.
seed	If using parallel processing (mcc > 1), a seed for the random number generation for BestTau. Don't use set.seed, it won't work.
arr3d.list	A list of 3-dimensional arrays. To perform this on files that have not yet been read in, set this argument to the path to these files (a character vector).
fail	If thresholding is done, to which value should pixels not exceeeding the threshold be set?

Details

Do not try to parallelize the use of Number and friends yourself (e.g. with mclapply()) because it will throw an error (this is because the autothresholdr package does not work in parallel (indeed anything run using the rJava package won't)). Always use Numbers for this purpose (this has a workaround whereby it does the thresholding in series and does the rest in parallel).

Value

Number returns a matrix, the number image; Numbers returns a list of these. The result of NumberTxtFolder is the text csv files written to disk (in the same folder as the input images).

Examples

```
library(EBImage)
img <- ReadImageData(system.file('extdata', '50.tif', package = 'nandb'))
display(normalize(img[, , 1]), method = 'raster')
number <- Number(img, tau = NA, mst = "Huang")
setwd(tempdir())
WriteIntImage(img, '50.tif')
WriteIntImage(img, '50again.tif')
NumberTxtFolder(tau = NA, mst = "Huang", mcc = 2)
file.remove(list.files())  # cleanup
img.paths <- rep(system.file('extdata', '50.tif', package = 'nandb'), 2)
numbers <- Numbers(img.paths, mst = 'otsu', mcc = 2)
```

`NumberTimeSeries` *Create a number time-series.*

Description

Given a stack of images `arr3d`, use the first `frames.per.set` of them to create one number image, the next `frames.per.set` of them to create the next number image and so on to get a time-series of number images. If `tau` is specified, bleaching correction is performed via `CorrectForBleaching()`.

Usage

```
NumberTimeSeries(arr, frames.per.set, tau = NA, mst = NULL, filt = NULL,
  n.ch = 1, verbose = FALSE, mcc = parallel::detectCores(), seed = NULL)
```

Arguments

`arr`	A 3-dimensional array (the image stack) where the nth slice is the nth image in the time series. To perform this on a file that has not yet been read in, set this argument to the path to that file (a string).
`frames.per.set`	The number of frames with which to calculate the successive numbers.
`tau`	If this is specified, bleaching correction is performed with `CorrectForBleaching()` which uses exponential filtering with time constant `tau` (where the unit of time is the time between frames). If this is set to `'auto'`, then the value of `tau` is calculated automatically via `BestTau()`.
`mst`	Do you want to apply an intensity threshold prior to calculating number (via `autothresholdr::mean_stack_thresh()`)? If so, set your thresholding *method* here.
`filt`	Do you want to smooth (`filt = 'smooth'`) or median (`filt = 'median'`) filter the number image using `SmoothFilterB()` or `MedianFilterB()` respectively? If selected, these are invoked here with a filter radius of 1 and with the option `na_count = TRUE`. If you want to smooth/median filter the number image in a different way, first calculate the numbers without filtering (`filt = NULL`) using this function and then perform your desired filtering routine on the result.

n.ch	The number of channels in the image (default 1).
verbose	If arr3d is specified as a file name, print a message to tell the user that that file is now being processed (useful for NumberTxtFolder, does not work with multiple cores).
mcc	The number of cores to use for the parallel processing.
seed	If using parallel processing (mcc > 1), a seed for the random number generation for BestTau. Don't use set.seed, it won't work.

Details

This may discard some images, for example if 175 frames are in the input and frames.per.set = 50, then the last 25 are discarded. If bleaching correction is selected, it is performed on the whole image stack before the sectioning is done for calculation of numbers.

Value

An array where the ith slice is the ith number image.

See Also

Number().

Examples

```
library(EBImage)
img <- ReadImageData(system.file('extdata', '50.tif', package = 'nandb'))
bts <- NumberTimeSeries(img, 20, tau = NA, mst = "Huang", mcc = 2)
```

PillarsDF *Make each pillar of a 3D array into a column of a tibble*

Description

Create a data frame (tibble) where pillar [i, j,] is column i + n1 * (j - 1) with column name "i_j" (we use an underscore here rather than a comma because a comma is the delimiter in csv files so writing this data frame to a csv could cause confusion).

Usage

```
PillarsDF(arr3d)
```

Arguments

arr3d	A 3-dimensional array.

Value

A tibble.

Examples

```
arr <- array(1:27, dim = rep(3, 3))
print(arr)
ListPillars(arr)
PillarsDF(arr)
```

`PillarsListToArr` *Make a list of pillars back into a 3D array.*

Description

#' Suppose the array is of dimension `n1 * n2 * n3`, then pillar `[i, j, ]` is list element `i + n1 * (j - 1)`, so the first element is pillar `[1, 1, ]`, the second is pillar `[1, 2, ]` and so on.

Usage

```
PillarsListToArr(pillars.list, dim)
```

Arguments

`pillars.list`	The list of pillars
`dim`	The desired dimension of the output array.

Value

A 3-dimensional array.

See Also

ListPillars

Examples

```
arr <- array(1:27, dim = rep(3, 3))
print(arr)
ListPillars(arr)
PillarsListToArr(ListPillars(arr), dim(arr))
```

`ReadImageData` *Read image as array object.*

Description

Read in an image file from the disk as an array of pixel intensities.

Usage

```
ReadImageData(image.name, fix.lut = NULL)
```

Arguments

`image.name`	The path to the image file on disk. The file extension must be one of '.jpeg', '.png', '.tiff' or '.tif'.
`fix.lut`	When reading in images (via `EBImage::readImage()`), R can give an array of different dimensionality than you expect. If you suspect this happening, set the value of this parameter to the *number of dimensions* that you expect your read image to have and this function will try to automatically give you the image array in the form you want. Read `FixLUTError()` to find out more.

Details

This function wraps `EBImage::readImage()` and `EBImage::imageData()`. By default, `readImage` reads in pixel intensities in the range `[0, 1]`. `ReadImageData` reads in pixel intensities as integers as they would be represented in a tiff file and displayed therefrom in ImageJ. This is necessary when calculating number and brightness, where we need pixel values to be in units of 'counts'.

Thinking of the read image as a matrix `mat`, the pixel at $x =$`i`, $y =$`j` has colour based on the value of `mat[i, j]` where the x axis points right and the y axis points down. This is in accordance with how EBImage::EBImage's `EBImage::readImage()` (which this function wraps). However, when one prints a matrix in a console (or views it in a program such as excel), the value in position $x =$`i`, $y =$`j` is from `mat[j, i]`, so if you're confused about a phantom transposition, this is why.

Sometimes (for whatever reason) the reading of image values as integers with `readTIFF(..., as.is = TRUE)` fails such that the values which should be 1, 2, 3 etc. end up as 256, 512, 778 etc. This function corrects for this as follows: if the greatest common divisor of the elements in the array are one of 2^8, 2^12, 2^16 or 2^32, then divide each element of the array by this greatest common divisor. This will return 256, 512, 778 etc. to 1, 2, 3 etc.

Value

An array of integers representing the image.

Examples

```
img <- ReadImageData(system.file('extdata', '50.tif', package = 'nandb'))
```

SpreadSpecific *Get the best* spread *of numbers in an interval.*

Description

Say we have an interval $[a, b]$ and we want to evenly spread n numbers in this interval, however m of these n numbers have been chosen beforehand, so we have to fithe the other $n - m$ in around them the best we can. Our goal is to maximize the minimum distance between adjacent elements.

Usage

```
SpreadSpecific(interval, specific, n, log = FALSE)
```

Arguments

interval	A length 2 numeric vector. The real interval over which one wants to spread the numbers.
specific	A numeric vector. The specific numbers that one wants to be part of our spread.
n	A number. The total number of numbers that you want to be in the spread (including the number(s) in specific).
log	Measure the difference between adjacent elements as the difference in their log instead?

Details

The idea is to, for an interval of size s, put in $floor(s/(n - m))$ numbers into each of these intervals. Then, if there any numbers left over, put exactly one into each of the intervals with the biggest $(s/(n-m)) - `floor`(s/(n-m))$ (starting with the biggest and working one's way down) until there are no numbers left to assign. The end intervals need special treatment since (assuming the ends are not in the specific numbers), the end intervals are open on one side (one boundary has no point on it), whereas all the middle intervals are not.

Value

A numeric vector. The chosen n numbers.

Examples

```
SpreadSpecific(c(0, 10), 1, 3)
```

Stack2DTifs *Put individual tif files into one tif stack.*

Description

Say you have saved what you would like to be one 3D tif stack as a series of 2D tif files. This helps you stack them into one file.

Usage

```
Stack2DTifs(file.names, out.name, mcc = parallel::detectCores())
```

Arguments

file.names	The names of the files to stack (in order).
out.name	The name of the output .tif file.
mcc	The number of parallel cores to use for the processing.

Examples

```
setwd(tempdir())
img <- ReadImageData(system.file('extdata', '50.tif', package = 'nandb'))
WriteIntImage(img[, , 1], '50_1.tif')
WriteIntImage(img[, , 2], '50_2.tif')
Stack2DTifs(c('50_1.tif', '50_2.tif'), '50_1_2')
file.remove(list.files())
```

WriteImageTxt *Read/write an image array to/from disk as text file(s).*

Description

If (as with brightness) you wish for the pixel values in an image to be represented by real numbers that aren't necessarily integers, the tiff format won't work. As a workaround we represent images (arrays) as comma-separated value (csv) files on disk, where for image stacks (3-dimensional arrays), we write one file for each slice, numbering it with the slice number.

Usage

```
WriteImageTxt(img.arr, file.name)

ReadImageTxt(file.name)
```

Arguments

img.arr	An image, represented by a 2- or 3-dimensional array.
file.name	The name of the input/output output file(s), *without* a file extension.

Details

The image slices are transposed prior to being written to disk to ensure that displaying an image with EBImage::display() in R will yield the same result (as opposed to a transposed image) as displaying the written text file in ImageJ (i.e. I've made a modification to ensure the files display correctly in ImageJ). These functions do not work for 4-dimensional arrays (e.g. a z stack with several channels).

Examples

```
setwd(tempdir())
img <- ReadImageData(system.file('extdata', '50.tif', package = 'nandb'))
WriteImageTxt(img, 'temp')

img <- ReadImageTxt('temp_01.csv')
file.remove(list.files())  # cleanup
```

WriteIntImage *Write an integer array to disk as a tiff image.*

Description

EBImage::writeImage truncates all values above 1 to 1 and all below 0 to 0 when writing images. This function allows you to write integer-vlued arrays to disk as tiff files as you would want.

Usage

```
WriteIntImage(img.arr, file.name, na = "error")
```

Arguments

img.arr	An integer array.
file.name	The name of the tif file (with or without the .tif) that you wish to write.
na	How do you want to treat NA values? R can only write integer values (and hence not NAs) to tiff pixels. na = 'saturate' sets them to saturated value. na = 'zero' sets them to zero, while na = 'error' will give an error if the image contains NAs. You can also specify directly here a natural number (must be between 0 and 2 ^ 16 - 1) to use in place of NAs.

Examples

```
img <- ReadImageData(system.file('extdata', '50.tif', package = 'nandb'))
setwd(tempdir())
WriteIntImage(img, '50.tif')
file.remove(list.files())  # cleanup
```

Index