Package 'imagine'

January 13, 2025

Type Package

Title IMAGing engINEs, Tools for Application of Image Filters to Data Matrices

Version 2.1.2

Date 2025-01-13

URL https://github.com/LuisLauM/imagine

BugReports https://github.com/LuisLauM/imagine/issues

Maintainer Wencheng Lau-Medrano <luis.laum@gmail.com>

Description Provides fast application of image filters to data matrices, using R and C++ algorithms.

License GPL (>= 2)

LazyData TRUE

Depends R (>= 3.1.0)

Imports Rcpp, RcppArmadillo

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.3.2

Suggests knitr, rmarkdown

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation yes

Author Wencheng Lau-Medrano [aut, cre]

Repository CRAN

Date/Publication 2025-01-13 17:50:14 UTC

Contents

`agenbagFilters` *Performs algorithms from Agenbag et al. (2003)*

Description

This function performs two (gradient) calculation approaches for SST, as outlined in the paper by Agenbag et al. (2003).

Usage

```
agenbagFilters(X, algorithm = c(1, 2), ...)
```

Arguments

`X`	A numeric `matrix` used as main input.
`algorithm`	`integer` specifying the type of method that will be used. See Details.
`...`	Not used.

Details

Section 2.2.4 of the paper by Agenbag et al. (2003) introduces the following two methods:

Method 1: Based on the equation

$$Y_{i,j} = \sqrt{(X_{i+1,j} - X_{i-1,j})^2 + (X_{i,j+1} - X_{i,j-1})^2}$$

where $Y_{i,j}$ represents the output value for each $X_{i,j}$ pixel value of a given X matrix.

Method 2: the standard deviation in a 3x3 pixel area centered on position (i, j).

As outlined in the original study, this method conducts searches within a 1-pixel vicinity of each point. For method 1, it only returns a value for points where none of the four involved values are NA. Conversely, for method 2, the standard deviation calculation is performed only for points where at least 3 non-NA values are found in the 3x3 neighborhood.

Value

`agenbagFilters` returns a `matrix` object with the same dimensions of `X`.

References

Agenbag, J.J., A.J. Richardson, H. Demarcq, P. Freon, S. Weeks, and F.A. Shillington. "Estimating Environmental Preferences of South African Pelagic Fish Species Using Catch Size- and Remote Sensing Data". Progress in Oceanography 59, No 2-3 (October 2003): 275-300. (doi:10.1016/j.pocean.2003.07.004).

Examples

```
data(wbImage)

# Agenbag, method 1
agenbag1 <- agenbagFilters(X = wbImage, algorithm = 1)

# Agenbag, method 2
agenbag2 <- agenbagFilters(X = wbImage, algorithm = 2)

# Plotting results
par(mfrow = c(3, 1), mar = rep(0, 4))

# Original
image(wbImage, axes = FALSE, col = gray.colors(n = 1e3))

# Calculated
cols <- hcl.colors(n = 1e3, palette = "YlOrRd", rev = TRUE)
image(agenbag1, axes = FALSE, col = cols)
image(agenbag2, axes = FALSE, col = cols)
```

---

contextualMF *Performs Contextual Median Filter*

Description

This function implements the Contextual Median Filter (CMF) algorithm, which was first described by Belkin & O'Reilly (2009), following the pseudocode provided in their paper.

Usage

```
contextualMF(X)
```

Arguments

X	A numeric `matrix` object used for apply filters.

Details

Following the definition of CMF, since **imagine** v.2.0.0, `times` argument will not be available anymore.

imagine offers the CMF algorithm but for the using to find out oceanographic fronts, it is recommended to see and use the functions of the **grec** package.

Value

`contextualMF` returns a `matrix` object with the same dimensions of `X`.

References

Belkin, I. M., & O'Reilly, J. E. (2009). An algorithm for oceanic front detection in chlorophyll and SST satellite imagery. Journal of Marine Systems, 78(3), 319-326 (doi:10.1016/j.jmarsys.2008.11.018).

Examples

```
data(wbImage)

cmfOut <- contextualMF(X = wbImage)

# image(cmfOut)
```

`convolution2D` *Make convolution calculations from numeric matrix*

Description

This function takes a `matrix` object, and for each cell multiplies its neighborhood by the `kernel`. Finally, it returns for each cell the mean of the kernel-weighted sum.

Usage

```
convolution2D(X, kernel, times = 1, normalize = FALSE, na_only = FALSE)

convolutionQuantile(
  X,
  kernel,
  probs,
  times = 1,
  normalize = FALSE,
  na_only = FALSE
)

convolutionMedian(X, kernel, times = 1, na_only = FALSE)
```

Arguments

`X`	A numeric `matrix` object used for apply filters.
`kernel`	A little matrix used as mask for each cell of `X`.
`times`	How many times do you want to apply the filter?
`normalize`	`logical` indicating if results will (or not) be normalized. See details.
`na_only`	`logical`, Do you want to apply the filter ONLY in cells with NA?
`probs`	`numeric` vector of probabilities with values in [0,1].

Details

Convolution is a mathematical operation that combines two arrays of numbers to produce an array of the same structure. The output will consist of only valid values, meaning those where both arrays have non-missing data. Consequently, any missing values (NAs) in the input matrix will propagate outwards to the extent of the convolution kernel.

Through normalization, the output of each convolution window is scaled by dividing it by the sum of the absolute values of the kernel (`sum(abs(as.numeric(kernel)))`, disabled by default).

`na_only` performs two actions at once: (1) it applies the filter only in the positions where the original value is `NA` and (2) for the rest of the cells, it is replaced with the value of the original matrix.

Value

`convolution2D` returns a `matrix` object with the same dimensions of `X`.

`convolutionQuantile` uses the kernel but, for each cell, it returns the position of quantile 'probs' (value between 0 and 1).

`convolutionMedian` is a wrapper of `convolutionQuantile` with probs = 0.5.

Examples

```
# Generate example matrix
nRows <- 50
nCols <- 100

myMatrix <- matrix(runif(nRows*nCols, 0, 100), nrow = nRows, ncol = nCols)
kernel <- diag(3)

# Make convolution
myOutput1 <- convolution2D(myMatrix, kernel)
myOutput2 <- convolutionQuantile(myMatrix, kernel, probs = 0.7)

# Plot results
par(mfrow = c(2, 2))
image(myMatrix, zlim = c(0, 100))
image(myOutput1, zlim = c(0, 100))
image(myOutput2, zlim = c(0, 100))
```

`meanFilter` *Make a 2D filter calculations from numeric matrix*

Description

This functions take a `matrix` object, and for each cell calculate mean, median or certain quantile around a squared/rectangular neighborhood.

Usage

```
meanFilter(X, radius, times = 1, na_only = FALSE)

quantileFilter(X, radius, probs, times = 1, na_only = FALSE)

medianFilter(X, radius, times = 1, na_only = FALSE)
```

Arguments

X	A numeric matrix object used for apply filters.
radius	Size of squared or rectangular kernel to apply median. See Details.
times	How many times do you want to apply the filter?
na_only	logical, Do you want to apply the filter ONLY in cells with NA?
probs	numeric vector of probabilities with values in [0,1].

Details

radius must be defined as a 2-length numeric vector specifying the number of rows and columns of the window which will be used to make calculations. If the length of radius is 1, the window will be a square.

Functions use C++ algorithms for running some statistical calculations. The mean is far obvious, however, there are several ways to perform quantiles. quantileFilter function uses arma::quantile: a RcppArmadillo function, which is equivalent to use R quantile funtion with type = 5.

medianFilter is a wraper of quantileFilter, so the same observations are applied to it.

na_only performs two actions at once: (1) it applies the filter only in the positions where the original value is NA and (2) for the rest of the cells, it is replaced with the value of the original matrix.

Value

A matrix object with the same dimensions of X.

quantileFilter don't use a kernel but, for each cell, it returns the position of quantile 'probs' (value between 0 and 1).

medianFilter is a wrapper of quantileFilter with probs = 0.5.

Examples

```
# Generate example matrix
nRows <- 50
nCols <- 100

myMatrix <- matrix(runif(nRows*nCols, 0, 100), nrow = nRows, ncol = nCols)
radius <- 3

# Make convolution
myOutput1 <- meanFilter(X = myMatrix, radius = radius)
myOutput2 <- quantileFilter(X = myMatrix, radius = radius, probs = 0.1)
myOutput3 <- medianFilter(X = myMatrix, radius = radius)
```

```
# Plot results
par(mfrow = c(2, 2))
image(myMatrix, zlim = c(0, 100), title = "Original")
image(myOutput1, zlim = c(0, 100), title = "meanFilter")
image(myOutput2, zlim = c(0, 100), title = "quantileFilter")
image(myOutput3, zlim = c(0, 100), title = "medianFilter")
```

---

wbImage *Data matrix to be used as example image.*

Description

`matrix` object containing numeric data to plot a image. The photo was taken by the author on 2016.

Usage

```
wbImage
```

Format

A `matrix` with dimensions 1280x720.

Index