Package 'ImageArray'

November 6, 2026

Type Package

Title A framework for on-disk and in-memory image arrays

Version 1.1.7

Description ImageArray provides a framework for on-disk and in-memory image arrays, specifically for pyramidal images stored in HDF5, Zarr and life sciences image file formats (OME Bio-Formats).

License MIT + file LICENSE

Encoding UTF-8

biocViews Software, Visualization

Depends R (>= 4.5.0), EBImage

Imports methods, grDevices, S4Arrays, S4Vectors, DelayedArray, rhdf5, HDF5Array, Rarr (>= 2.1.27), ZarrArray, magick, tools, BiocGenerics

Suggests testthat (>= 3.0.0), knitr, ggplot2, shiny, RBioFormats, BiocFileCache, BiocStyle

Config/testthat/edition 3

VignetteBuilder knitr

URL https://github.com/BIMSBbioinfo/ImageArray

BugReports https://github.com/BIMSBbioinfo/ImageArray/issues

Config/roxygen2/version 8.1.0

git_url https://git.bioconductor.org/packages/ImageArray

git_branch devel

git_last_commit 81809f2

git_last_commit_date 2026-08-10

Repository Bioconductor 3.24

Date/Publication 2026-11-05

Author Artür Manukyan [aut, cre] (ORCID: <https://orcid.org/0000-0002-0441-9517>), Helmholtz Association [fnd] (https://www.helmholtz.de, ROR: <https://ror.org/0281dp749>)

Maintainer Artür Manukyan <artur-man@hotmail.com>

Contents

BFArray-methods 2
BFPath 3
getImageInfo 4
ImageArray-class 4
ImageArray-constructor 5
ImageArray-methods 6
trans 9
trans_seed 12
writeImageArray 13

Index **15**

`BFArray-methods` *BFArray constructor method*

Description

A function for creating objects of BFArray class

Usage

```
BFArray(image)

## S4 method for signature 'BFArraySeed'
dim(x)

## S4 method for signature 'BFArray'
axes(object)

## S4 method for signature 'BFArraySeed'
axes(object)

## S4 method for signature 'BFArraySeed'
type(x)
```

Arguments

`image`	A BFPath object
`x, object`	A BFArray or BFArraySeed object

Value

A BFArray object

Functions

- `dim(BFArraySeed)`: dim function for BFArray objects
- `axes(BFArray)`: axes function for BFArray objects
- `axes(BFArraySeed)`: axes function for BFArray objects
- `type(BFArraySeed)`: type function for BFArray objects

Examples

```
# get image
library(RBioFormats)
img.file <- system.file("extdata",
                        "xy_12bit__plant.ome.tiff",
                        package = "ImageArray")
bfp <- BFPath(img.file, series = 1, resolution = 2)
bfa <- BFArray(bfp)
dim(bfa)
type(bfa)
```

`BFPath` *BFPath constructor method*

Description

A function for creating objects of BFPath class

Usage

```
BFPath(filepath, series = NULL, resolution = NULL)
```

Arguments

`filepath`	the path to the image read by RBioFormats
`series`	the series IDs of the pyramidal image, typical an integer starting from 1
`resolution`	the resolution IDs of the pyramidal image, typical an integer starting from 1

Value

A BFArray object

Examples

```
# get image
library(RBioFormats)
img.file <- system.file("extdata",
                        "xy_12bit__plant.ome.tiff",
                        package = "ImageArray")
bfp <- BFPath(img.file, series = 1, resolution = 2)
series(bfp)
resolution(bfp)
path(bfp)
```

`getImageInfo` *getImageInfo*

Description

get information of an ImageArray object

Usage

```
getImageInfo(object)
```

Arguments

`object`	an ImageArray object

Value

a data frame of width and height info

Examples

```
# get image
library(EBImage)
img.file <- system.file("images", "sample.png", package="EBImage")

# create ImageArray
dir.create(td <- tempfile())
output_h5 <- tempfile(fileext = ".h5")
imgarray <- writeImageArray(img.file,
                            output = output_h5,
                            name = "image",
                            verbose = FALSE)
getImageInfo(imgarray)

# create ImageArray
imgarray <- ImageArray(img.file, n.levels = 3)
imgarray_raster <- as.raster(imgarray, max.pixel.size = 300)
getImageInfo(imgarray)
```

`ImageArray-class` *ImageArray class*

Description

An S4 container for a multi-resolution (pyramidal) image, holding the pyramid levels together with their scales. Objects are created with `ImageArray`.

Slots

`levels` an `ImageList` of pyramid levels, ordered from the highest to the lowest resolution

`scales` a list of named numeric vectors, one per level, where values are the scales of these axes. The names of these vectors define the axes of the object, hence they are a subset of `c("c", "y", "x", "z", "t")` and are shared, in the same order, by all levels. See https://ngff.openmicroscopy.org/ for more information.

`ImageArray-constructor`

ImageArray

Description

creates an object of ImageArray class

Usage

```
ImageArray(
  image,
  axes = NULL,
  n.levels = NULL,
  max.pixel.threshold = 700,
  scales = NULL,
  engine = "EBImage",
  series = NULL,
  resolution = NULL,
  verbose = FALSE
)

createImageArray(
  image,
  n.levels = NULL,
  series = NULL,
  resolution = NULL,
  max.pixel.threshold = max.pixel.threshold,
  engine = "EBImage",
  verbose = FALSE
)
```

Arguments

`image`	a path to a file, a single array or a list of arrays containing the pixel intensities of an image.
`axes`	a character vector of axes names for images. Should be a subset of c("c", "y", "x", "z", "t")
`n.levels`	the number of levels of the pyramidal image, typical an integer starting from 1. Will be ignored if `scales` is provided

`max.pixel.threshold`
: the maximum width and height pixel dimension that the lowest level of the image pyramid should have, thus the image will be downscaled two folds until both width and height is below the threshold. Default is 700 pixels. Will be ignored if `n.levels` is provided

`scales`
: a list of named numeric vectors where names are a subset of `axes` and values are associated with scales of these axes. See https://ngff.openmicroscopy.org/ for more information. When provided, `axes`, `n.levels` and `max.pixel.threshold` will be overridden.

`engine`
: the package to use for each image layer: either `EBImage` or `magick-image`

`series`
: the series IDs of the pyramidal image, typical an integer starting from 1.

`resolution`
: the resolution IDs of the pyramidal image, typical an integer starting from 1.

`verbose`
: verbose

Value

An ImageArray object

Functions

- `createImageArray()`: deprecated function

Examples

```
# get image
library(EBImage)
img.file <- system.file("images", "sample.png", package="EBImage")

# create ImageArray
imgarray <- ImageArray(img.file, n.levels = 3)
imgarray_raster <- as.raster(imgarray, max.pixel.size = 300)
plot(imgarray_raster)
```

`ImageArray-methods` *Methods for ImageArray*

Description

Methods for `ImageArray` objects

Usage

```
## S4 method for signature 'ImageArray,ANY,ANY,ANY'
x[i, j, ..., drop = FALSE]

## S4 method for signature 'ImageArray,numeric'
x[[i]]

## S4 replacement method for signature 'ImageArray,numeric'
x[[i, j, ...]] <- value
```

```
## S4 method for signature 'ImageArray'
dim(x)

## S4 method for signature 'ImageArray'
type(x)

## S4 method for signature 'ImageArray'
length(x)

## S4 method for signature 'ImageArray'
axes(object)

## S4 replacement method for signature 'ImageArray'
axes(object, ...) <- value

## S4 method for signature 'ImageArray'
scales(object)

## S4 replacement method for signature 'ImageArray'
scales(object, ...) <- value

## S4 method for signature 'ImageArray'
realize(x, level = NULL, max.pixel.size = NULL, min.pixel.size = NULL)

## S3 method for class 'ImageArray'
as.raster(x, level = NULL, max.pixel.size = NULL, min.pixel.size = NULL, ...)

## S4 method for signature 'ImageArray'
aperm(a, perm)

## S4 method for signature 'ImageArray'
negate(object)

## S4 method for signature 'ImageArray'
modulate(object, brightness)

## S4 method for signature 'ImageArray'
crop(object, index)

## S4 method for signature 'ImageArray'
extent(x)

## S4 method for signature 'ImageArray'
path(object)

## S4 replacement method for signature 'ImageArray'
path(object) <- value
```

Arguments

x, a, object	An ImageArray object
i, j, value	Depends on the usage

	`[[`, `[[<-` Here `i` is the level of the image pyramid. You can use the `length` function to get the number of the layers in the pyramid. When used with `crop`, arguments `i` and `j` are associated with indices of image dimensions (e.g. width, height)
`...`	Arguments passed to other methods
`drop`	ignored
`level`	level
`max.pixel.size`	maximum pixel size
`min.pixel.size`	minimum pixel size
`perm`	perm
`brightness`	the brightness of the new image in percentage, e.g. 120
`index`	a named or unnamed list of indices for cropping/subsetting the

Value

dim of the first level of the ImageArray object

type of ImageArray object

length of ImageArray object

Functions

- `x = ImageArray[i = ANY, j = ANY, drop = ANY]`: subset and crop for `ImageArray` objects
- `x = ImageArray[[i = numeric]]`: Layer access for `ImageArray` objects
- `x = ImageArray[[i = numeric]] <- value`: Layer access for `ImageArray` objects
- `dim(ImageArray)`: dimensions of an ImageArray
- `type(ImageArray)`: dimensions of an ImageArray
- `length(ImageArray)`: length of an ImageArray
- `axes(ImageArray)`: get axes metadata of the ImageArray object
- `axes(ImageArray) <- value`: replace axes metadata of the ImageArray object, the replacement can only be a permutation of the existing axes
- `scales(ImageArray)`: get scales metadata of the ImageArray object
- `scales(ImageArray) <- value`: replace scales metadata of the ImageArray object, each vector should be named by a permutation of the existing axes
- `realize(ImageArray)`: realize the array
- `as.raster(ImageArray)`: create a raster object
- `aperm(ImageArray)`: permute image
- `negate(ImageArray)`: negate image
- `modulate(ImageArray)`: modulate image
- `crop(ImageArray)`: cropping image
- `extent(ImageArray)`: extent
- `path(ImageArray)`: path of an ImageArray object
- `path(ImageArray) <- value`: replace method for path(ImageArray)

Examples

```
# get image
library(EBImage)
img.file <- system.file("images", "sample.png", package="EBImage")

# create ImageArray
imgarray <- ImageArray(img.file, n.levels = 3)

# access layers
imgarray[[1]]
imgarray[[2]]

# dimensions and length
dim(imgarray)
length(imgarray)

# extent
extent(imgarray)

# manipulate images
imgarray <- crop(imgarray, ind = list(100:200, 100:200))
imgarray <- crop(imgarray, ind = list(x = 10:20, y = 10:20))
imgarray <- rotate(imgarray, angle = 90)
imgarray <- flip(imgarray)
imgarray <- flop(imgarray)

# create ImageArray on disk as HDF5 format
dir.create(td <- tempfile())
output_h5 <- tempfile(fileext = ".h5")
imgarray <- writeImageArray(img.file,
                            output = output_h5,
                            name = "image",
                            verbose = FALSE)

# path
path(imgarray)

# as.raster
imgarray_raster <- as.raster(imgarray)

# realize
imgarray <- realize(imgarray)
```

`trans` *Transformations*

Description

Transformation methods for objects of ImageArray class. By default, these operations are delayed thus transformations are only performed when realized into the memory

Usage

```
## S4 method for signature 'ImageArray'
```

```
affine(
  x,
  m,
  axes = NULL,
  output.dim = NULL,
  filter = c("bilinear", "none"),
  bg.col = "black",
  antialias = TRUE
)

## S4 method for signature 'ImageArray'
rotate(
  x,
  angle,
  filter = c("bilinear", "none"),
  bg.col = "black",
  antialias = TRUE
)

## S4 method for signature 'ImageArray'
scale(
  x,
  output.dim = NULL,
  output.origin = c(0, 0),
  axes = NULL,
  filter = c("bilinear", "none"),
  bg.col = "black",
  antialias = TRUE,
  ...
)

## S4 method for signature 'ImageArray'
translation(x, shift = c(0, 0), axes = NULL)

## S4 method for signature 'ImageArray'
flip(x)

## S4 method for signature 'ImageArray'
flop(x)
```

Arguments

x	an ImageArray object
m	A 3x2 matrix describing the affine transformation. See Details.
axes	axes
output.dim	A vector of 2 numbers indicating the dimension of the output image. For `affine` and `translate` the default is `dim(x)`, for `resize` it equals `c(w, h)`, and for `rotate` it defaults to the bounding box size of the rotated image.
filter	A character string indicating the interpolating sampling filter. Valid values are 'none' or 'bilinear'. See Details.
bg.col	Color used to fill the background pixels, defaults to "black". In the case of multi-frame images the value is recycled, and individual background for each frame

	can be specified by providing a vector.
antialias	If TRUE, perform bilinear sampling at image edges using bg.col.
angle	A numeric specifying the image rotation angle in degrees.
output.origin	A vector of 2 numbers indicating the output coordinates of the origin in pixels.
...	arguments passed to other methods
shift	translation shift parameter, a vector of length 2

Value

an ImageArray object.

Functions

- affine(ImageArray): affine transformation
- rotate(ImageArray): rotation transformation
- scale(ImageArray): scale transformation
- translation(ImageArray): translation transformation
- flip(ImageArray): vertical flipping
- flop(ImageArray): horizontal flipping

Examples

```
ome.tiff.file <- system.file("extdata", "xy_12bit__plant.ome.tiff",
                             package = "ImageArray")

library(RBioFormats)
read.metadata(ome.tiff.file)

# define ImageArray object
imgarray <- ImageArray(ome.tiff.file, series = 1, resolution = 1:2)

# translate image
imgarray_trans <- translation(imgarray, shift = c(100,20))

# scale image
imgarray_scale <- scale(imgarray, output.dim = c(200,300),
                        filter = "bilinear")
imgarray_scale

# rotate image
imgarray_rotate <- rotate(imgarray, angle = 45, filter = "bilinear")
imgarray_rotate
plot(as.raster(imgarray_rotate))

# image with affine transformation
m <- matrix(c(1, -.5, 128, 0, 1, 0), nrow=3, ncol=2)
imgarray_affine <- affine(imgarray,
                          m = m,
                          filter = "bilinear")
imgarray_affine
plot(as.raster(imgarray_affine))

# get extent of the transformed image
```

```
extent(imgarray_affine)

# a sequence of transformations
m <- matrix(c(1, -.5, 128, 0, 1, 0), nrow=3, ncol=2)
imgarray_trans <- affine(imgarray,
                         m = m,
                         filter = "bilinear")
imgarray_trans

imgarray_trans <- scale(imgarray_trans,
                        output.dim = c(200,300),
                        filter = "bilinear")
imgarray_trans <- translation(imgarray_trans,
                              shift = c(100,20))
```

`trans_seed` *DelayedTransformSeed Methods*

Description

Methods for Delayed Transformations seeds

Usage

```
## S4 method for signature 'DelayedTransformSeed'
dim(x)

## S4 method for signature 'DelayedTransformSeed'
dimnames(x)

## S4 method for signature 'DelayedTransformSeed'
type(x)
```

Arguments

`x`	DelayedTransformSeed object

Value

dimension lengths of DelayedTransformSeed object

dimension names of DelayedTransformSeed object

type of the data of DelayedTransformSeed object

writeImageArray *writeImageArray*

Description

Writing image arrays on disk

Usage

```
writeImageArray(
  image,
  output = "my_image",
  name = "",
  format = NULL,
  replace = FALSE,
  chunkdim = NULL,
  level = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

image	an Image object (EBImage), a magick object or the path to an image file,
output	output file name
name	name of the group
format	on disk format, either "h5" for HDF5 format, "zarr" for zarr format, or "in-memory" for in-memory ImageArray object. If not provided, the format will be inferred from the file extension of the output path.
replace	Should the existing file be removed or not
chunkdim	The dimensions of the chunks to use for writing the data to disk.
level	The compression level to use for writing the data to disk.
verbose	verbose
...	additional parameters passed to ImageArray.

Value

An ImageArray object

Examples

```
# get image
library(EBImage)
img.file <- system.file("images", "sample.png", package="EBImage")

# create ImageArray
dir.create(td <- tempfile())
output_h5 <- tempfile(fileext = ".h5")
imgarray <- writeImageArray(img.file,
                            output = output_h5,
```

```
                          name = "image",
                          verbose = FALSE)
imgarray_raster <- as.raster(imgarray)
plot(imgarray_raster)
```

Index

.ImageArray *(ImageArray-class)*, 4
[,ImageArray,ANY,ANY,ANY-method *(ImageArray-methods)*, 6
[[,ImageArray,numeric-method *(ImageArray-methods)*, 6
[[<-,ImageArray,numeric-method *(ImageArray-methods)*, 6

affine *(trans)*, 9
affine,ImageArray-method *(trans)*, 9
aperm,ImageArray-method *(ImageArray-methods)*, 6
as.raster *(ImageArray-methods)*, 6
as.raster,ImageArray-method *(ImageArray-methods)*, 6
as.raster.ImageArray *(ImageArray-methods)*, 6
axes *(ImageArray-methods)*, 6
axes,BFArray-method *(BFArray-methods)*, 2
axes,BFArraySeed-method *(BFArray-methods)*, 2
axes,ImageArray-method *(ImageArray-methods)*, 6
axes<- *(ImageArray-methods)*, 6
axes<-,ImageArray-method *(ImageArray-methods)*, 6

BFArray *(BFArray-methods)*, 2
BFArray-methods, 2
BFPath, 3

createImageArray *(ImageArray-constructor)*, 5
createImageArray,ImageArray-method *(ImageArray-constructor)*, 5
crop *(ImageArray-methods)*, 6
crop,ImageArray-method *(ImageArray-methods)*, 6

dim,BFArraySeed-method *(BFArray-methods)*, 2
dim,DelayedTransformSeed-method *(trans_seed)*, 12
dim,ImageArray-method *(ImageArray-methods)*, 6
dimnames,DelayedTransformSeed-method *(trans_seed)*, 12

extent *(ImageArray-methods)*, 6
extent,ImageArray-method *(ImageArray-methods)*, 6

flip,ImageArray-method *(trans)*, 9
flop,ImageArray-method *(trans)*, 9

getImageInfo, 4

ImageArray, *4*, *13*
ImageArray *(ImageArray-constructor)*, 5
ImageArray-class, 4
ImageArray-constructor, 5
ImageArray-methods, 6

length,ImageArray-method *(ImageArray-methods)*, 6

meta *(ImageArray-methods)*, 6
meta,ImageArray-method *(ImageArray-methods)*, 6
modulate *(ImageArray-methods)*, 6
modulate,ImageArray-method *(ImageArray-methods)*, 6

negate *(ImageArray-methods)*, 6
negate,ImageArray-method *(ImageArray-methods)*, 6

path *(ImageArray-methods)*, 6
path,BFPath-method *(BFPath)*, 3
path,ImageArray-method *(ImageArray-methods)*, 6
path<-,ImageArray-method *(ImageArray-methods)*, 6

realize *(ImageArray-methods)*, 6
realize,ImageArray-method *(ImageArray-methods)*, 6
resolution *(BFPath)*, 3
resolution,BFPath-method *(BFPath)*, 3
rotate,ImageArray-method *(trans)*, 9

scale (trans), 9
scale,ImageArray-method (trans), 9
scales (ImageArray-methods), 6
scales,ImageArray-method
        (ImageArray-methods), 6
scales<- (ImageArray-methods), 6
scales<-,ImageArray-method
        (ImageArray-methods), 6
series (BFPath), 3
series,BFPath-method (BFPath), 3

trans, 9
trans_seed, 12
translation (trans), 9
translation,ImageArray-method (trans), 9
type,BFArraySeed-method
        (BFArray-methods), 2
type,DelayedTransformSeed-method
        (trans_seed), 12
type,ImageArray-method
        (ImageArray-methods), 6

writeImageArray, 13