Package: ImageArray (via r-universe)

June 1, 2026

Type Package

Title ImageArray: a framework for on-disk and in-memory image arrays

Version 0.99.6

Date 2026-01-20

Description ImageArray provides a framework for on-disk and in-memory image arrays, specifically for pyramidal images stored in either HDF5 files or Zarr stores.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

biocViews Software, Visualization

Depends EBImage

Imports methods, grDevices, S4Arrays, S4Vectors, DelayedArray, rhdf5, HDF5Array, Rarr, magick

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/pak/sysreqs libblosc-dev cmake libfftw3-dev make libmagick++-dev gsfonts libjpeg-dev libpng-dev libtiff-dev libuv1-dev libxml2-dev libzstd-dev libssl-dev zlib1g-dev

Repository https://artur-man.r-universe.dev

Date/Publication 2026-01-20 21:13:20 UTC

RemoteUrl https://github.com/Artur-man/ImageArray

RemoteRef HEAD

RemoteSha 2f90694fa33ed8320a6a2f7e0ad5c8a1796cbfc6

Contents

as.raster,ImageArray-method . . . 2
BFArray-methods . . . 3
createImageArray . . . 4
getImageInfo . . . 5
ImageArray-methods . . . 6
path,ImageArray-method . . . 8
path<-,ImageArray-method . . . 9
realize,ImageArray-method . . . 9
writeImageArray . . . 10
zarr-utils . . . 11

Index **13**

`as.raster,ImageArray-method`
as.raster method for ImageArray object

Description

as.raster method for ImageArray object

Usage

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

Arguments

`x`	an ImageArray object
`level`	level
`max.pixel.size`	maximum pixel size
`min.pixel.size`	minimum pixel size

Value

A raster array

Examples

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

# create ImageArray
dir.create(td <- tempfile())
output_h5ad <- file.path(td, "h5test")
```

```
imgarray <- writeImageArray(img.file,
                            output = output_h5ad,
                            name = "image",
                            format = "HDF5ImageArray",
                            replace = TRUE, verbose = FALSE)
imgarray_raster <- as.raster(imgarray)
```

BFArray-methods *BFArray constructor method*

Description

A function for creating objects of BFArray class

Usage

```
BFArray(image.file, series, resolution)

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

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

Arguments

image.file	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
x	A BFArray object

Value

A BFArray object

Functions

- dim(BFArraySeed): dim 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")
bfa <- BFArray(img.file, series = 1, resolution = 2)
dim(bfa)
type(bfa)
```

`createImageArray` *createImageArray*

Description

creates an object of ImageArray class

Usage

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

Arguments

`image`	the image
`n.levels`	the number of levels of the pyramidal image, typical an integer starting from 1
`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.
`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. If `n.levels` is provided, this parameter will be ignored.
`engine`	the package to use for each image layer: either `EBImage` or `magick-image`
`verbose`	verbose

Value

An ImageArray object

Examples

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

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

`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_h5ad <- file.path(td, "h5test")
imgarray <- writeImageArray(img.file,
                            output = output_h5ad,
                            name = "image",
                            format = "HDF5ImageArray",
                            replace = TRUE, verbose = FALSE)
getImageInfo(imgarray)

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

```
getImageInfo(imgarray)
```

ImageArray-methods *Methods for ImageArray*

Description

Methods for ImageArray objects

Usage

```
## S4 method for signature 'ImageArray,numeric,numeric,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)

ImageArray(meta, levels)

## S4 method for signature 'ImageArray'
rotate(object, degrees)

## 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'
flip(object)
```

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

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

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

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
meta	the metadata of the ImageArray object.
levels	levels of the pyramid image, typically a vector of integers starting with 1
degrees	value between 0 and 360 for how many degrees to rotate
perm	perm
brightness	the brightness of the new image in percentage, e.g. 120
ind	index list

Value

dim of the first level of the ImageArray object

type of ImageArray object

length of ImageArray object

An ImageArray object

Functions

- x[i: subset and crop for ImageArray objects
- x[[i: 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
- ImageArray(): ImageArray constructor method
  A function for creating objects of ImageArray class

- `rotate(ImageArray)`: rotate image array to 90, 180, 270 degrees
- `aperm(ImageArray)`: permute image
- `negate(ImageArray)`: negate image
- `modulate(ImageArray)`: modulate image
- `flip(ImageArray)`: vertical flipping image
- `flop(ImageArray)`: horizontal flipping image
- `crop(ImageArray)`: cropping image
- `axes(ImageArray)`: get axes metadata of the ImageArray object

Examples

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

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

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

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

# manipulate images
imgarray <- crop(imgarray, ind = list(100:200, 100:200))
imgarray <- rotate(imgarray, degrees = 90)
imgarray <- flip(imgarray)
imgarray <- flop(imgarray)
```

---

path,ImageArray-method

path of ImageArray image

Description

path of ImageArray image

Usage

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

Arguments

`object`	an ImageArray object

Value

the path to ImageArray object store

`path<-,ImageArray-method`
path of ImageArray image

Description

path of ImageArray image

Usage

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

Arguments

`object`	an ImageArray object
`value`	the new path

Value

does not return a value, updates the path of the ImageArray object

---

`realize,ImageArray-method`
as.array

Description

as.array method for ImageArray object

Usage

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

Arguments

`x`	an ImageArray object
`level`	level
`max.pixel.size`	maximum pixel size
`min.pixel.size`	minimum pixel size

Value

An array object

Examples

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

# create ImageArray
dir.create(td <- tempfile())
output_h5ad <- file.path(td, "h5test")
imgarray <- writeImageArray(img.file,
                            output = output_h5ad,
                            name = "image",
                            format = "HDF5ImageArray",
                            replace = TRUE, verbose = FALSE)
imgarray <- realize(imgarray)
```

---

writeImageArray *writeImageArray*

Description

Writing image arrays on disk

Usage

```
writeImageArray(
  image,
  output = "my_image",
  name = "",
  format = c("InMemoryImageArray", "HDF5ImageArray", "ZarrImageArray"),
  replace = FALSE,
  n.levels = NULL,
  chunkdim = NULL,
  level = NULL,
  engine = "EBImage",
  verbose = FALSE,
  ...
)
```

Arguments

image	image
output	output file name
name	name of the group

format	on disk format
replace	Should the existing file be removed or not
n.levels	the number of levels if the image supposed to be pyramidal.
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.
engine	the package to use for each image layer: either EBImage or magick-image
verbose	verbose
...	additional parameters passed to createImageArray.

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_h5ad <- file.path(td, "h5test")
imgarray <- writeImageArray(img.file,
                            output = output_h5ad,
                            name = "image",
                            format = "HDF5ImageArray",
                            replace = TRUE, verbose = FALSE)
imgarray_raster <- as.raster(imgarray)
plot(imgarray_raster)
```

---

zarr-utils *zarrcreateGroup*

Description

get information of an ImgArray object

Usage

```
zarrcreateGroup(store, name)

open_zarr(dir, name)
```

Arguments

store	the location of (zarr) store
name	name of the zarr store or group
dir	directory/location of zarr store

Value

does not return anything, creates a zarr group in the store instead

does not return anything, opens a zarr store instead

Functions

- zarrcreateGroup(): create zarr group
- open_zarr(): open zarr stores

Examples

```
# zarr store path
dir.create(td <- tempfile())
zarr_name <- "test.zarr"
output_zarr <- file.path(td, zarr_name)

# open zarr store
open_zarr(dir = td, name = zarr_name)

# create group
zarrcreateGroup(store = output_zarr, name = "sample")
```

Index

∗ **zarr-utils**
    zarr-utils, 11
[,ImageArray,numeric,numeric,ANY-method
        *(*ImageArray-methods*)*, 6
[[,ImageArray,numeric-method
        *(*ImageArray-methods*)*, 6
[[<-,ImageArray,numeric-method
        *(*ImageArray-methods*)*, 6

aperm,ImageArray-method
        *(*ImageArray-methods*)*, 6
as.raster,ImageArray-method, 2
axes *(*ImageArray-methods*)*, 6
axes,ImageArray-method
        *(*ImageArray-methods*)*, 6

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

createImageArray, 4, *11*
crop *(*ImageArray-methods*)*, 6
crop,ImageArray-method
        *(*ImageArray-methods*)*, 6

dim,BFArraySeed-method
        *(*BFArray-methods*)*, 3
dim,ImageArray-method
        *(*ImageArray-methods*)*, 6

flip *(*ImageArray-methods*)*, 6
flip,ImageArray-method
        *(*ImageArray-methods*)*, 6
flop *(*ImageArray-methods*)*, 6
flop,ImageArray-method
        *(*ImageArray-methods*)*, 6

getImageInfo, 5

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

length,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

open_zarr *(*zarr-utils*)*, 11

path,ImageArray-method, 8
path<-,ImageArray-method, 9

realize,ImageArray-method, 9
rotate *(*ImageArray-methods*)*, 6
rotate,ImageArray-method
        *(*ImageArray-methods*)*, 6

type,BFArraySeed-method
        *(*BFArray-methods*)*, 3
type,ImageArray-method
        *(*ImageArray-methods*)*, 6

writeImageArray, 10

zarr-utils, 11
zarrcreateGroup *(*zarr-utils*)*, 11