Package ‘RNifti’

January 31, 2020

Version 1.1.0
Date 2020-01-31
Title Fast R and C++ Access to NIfTI Images
Imports Rcpp (>= 0.11.0)
Suggests testthat (>= 0.11.0), covr, reportr, shades
Enhances oro.nifti, tractor.base
LinkingTo Rcpp
Description Provides very fast read and write access to images stored in the NIfTI-1 and ANALYZE-7.5 formats, with seamless synchronisation between compiled C and interpreted R code. Also provides a simple image viewer, and a C/C++ API that can be used by other packages. Not to be confused with ‘RNiftyReg’, which performs image registration.
License GPL-2
URL https://github.com/jonclayden/RNifti
BugReports https://github.com/jonclayden/RNifti/issues
Encoding UTF-8
RoxygenNote 6.1.1
NeedsCompilation yes
Author Jon Clayden [cre, aut],
   Bob Cox [aut],
   Mark Jenkinson [aut],
   Matt Hall [ctb],
   Rick Reynolds [ctb],
   Kate Fissell [ctb],
   Jean-loup Gailly [cph],
   Mark Adler [cph]
Maintainer Jon Clayden <code@clayden.org>
Repository CRAN
Date/Publication 2020-01-31 18:20:02 UTC
R topics documented:

asNifti .................................................. 2
channels .................................................. 4
defaultInfoPanel ........................................ 4
dim.internalImage ........................................ 5
ndim ..................................................... 6
niftiHeader ............................................... 7
niftiVersion ............................................. 8
pixdim .................................................... 9
readNifti .................................................. 10
rgbArray .................................................. 11
view ...................................................... 12
voxelToWorld ............................................ 14
writeNifti ............................................... 15
xform .................................................... 16
$.niftiImage ............................................. 18

Index 19

---

asNifti

Create or modify an NIfTI image object

Description

This function converts a filename, array or other image class into an object of class "niftiImage", and optionally updates its metadata from a reference image and/or changes its internal datatype. The dimensions and pixel dimensions from the image will replace those from the reference object, if they are available.

Usage

asNifti(image, reference = NULL, datatype = "auto", internal = NA)

Arguments

- **image**: Any suitable object (see Details).
- **reference**: An image, or a named list of NIfTI-1 properties like that produced by `niftiHeader`. The default of NULL will have no effect.
- **datatype**: The NIfTI datatype to use within the internal image. The default, "auto" uses the R type. Other possibilities are "float", "int16", etc., which may be preferred to reduce object size. However, no checks are done to ensure that the coercion maintains precision, and this option is for advanced usage only.
- **internal**: Logical value. If FALSE, the result will be an array of class "niftiImage" containing the image pixel or voxel values, with some metadata in attributes. If TRUE, the result will be an object of class "internalImage", which exposes some basic metadata to R but stores the pixel data internally. If NA, the default, the result will be an internal image only if the input image is. If a new datatype is set then this value is implicitly TRUE.
Details

If the image has an internal NIfTI pointer, that will be retrieved directly. Otherwise, if it is a string, it will be taken to be a filename. If it looks like a "nifti" object (from package oro.nifti), or an "MriImage" object (from package tractor.base), a conversion will be performed. A list will be assumed to be of the form produced by niftiHeader. Finally, a numeric array or matrix, or RGB array, will be converted using default image parameters.

If reference is a complete list of NIfTI-1 header fields, like that produced by niftiHeader, or an image, then it will be used to create the internal object, and then the data and metadata associated with the image will overwrite the appropriate parts. If reference is an incomplete list, the image will be used to create the internal object, and then the specified fields will be overwritten from the list. This allows users to selectively update certain fields while leaving others alone (but see the note below).

If multiple values are passed for a field that expects a scalar (which is most of them), the first element of the vector will be used, with a warning. An empty vector will be ignored, also with a warning. If a value of the wrong length is passed to a vector-valued field, an error will be generated.

Datatype information in a list reference is ignored. The datatype can only be changed using the datatype argument, but in this case the internal object gets out of sync with the R array, so an internal image is returned to avoid the mismatch. Changing the internal datatype in this way is for advanced usage only.

retrieveNifti and updateNifti are soft-deprecated alternative interfaces to this function, which behave like the pre-existing functions of the same names. They may be removed in future.

Value

An array or internal image, with class "niftiImage" (and possibly also "internalImage").

Note

The scl_slope and scl_inter fields affect the numerical interpretation of the pixel data, so it is impossible in general to change them without also changing the array values on both the C and the R side. Therefore, to avoid unexpected side-effects, these fields are not affected by this function. The dim and pixdim fields can be changed, but for most users the accessor functions of the same name are much safer, and should be used in preference.

Author(s)

Jon Clayden <code@clayden.org>

See Also

readNifti, $.niftiImage, dim.internalImage, pixdim, xform
channels

Extract channels from RGB data

Description

Extract one or more channels from an RGB data array that was obtained from an RGB NIfTI image or created by the rgbArray function. The result is more amenable to numeric manipulation.

Usage

channels(array, channels = c("red", "green", "blue", "alpha"), raw = FALSE)

Arguments

array
An image, an rgbArray, or another array that can be converted to the latter.

channels
A character vector of channels to extract.

raw
Boolean value: if TRUE, return a raw array, which is the most compact representation; otherwise return an integer array.

Value

A raw-mode or integer-mode array with one more dimension than the first argument, corresponding to channels.

Author(s)

Jon Clayden <code@clayden.org>

defaultInfoPanel

Info panels for the built-in viewer

Description

A default info panel for view, which shows the labels and values of each image at the current point, and a panel suitable for plotting four-dimensional time series images.

Usage

defaultInfoPanel(point, data, labels)

timeSeriesPanel(point, data, labels)
Arguments

- **point**: A numeric vector giving the current point location.
- **data**: A list of data values for each image at the current point. Note that, for images of more than three dimensions, there will be more than one value per image.
- **labels**: A character vector of image labels.

Author(s)

Jon Clayden <code@clayden.org>

See Also

- view

Description

An internal image is a simple R object with a few attributes including a pointer to an internal C structure, which contains the full image data. They are used in the package for efficiency, but can be converted to a normal R array using the `as.array` method. Attributes of these objects should not be changed.

Usage

```r
## S3 method for class 'internalImage'
dim(x)

## S3 replacement method for class 'internalImage'
dim(x) <- value

## S3 method for class 'internalImage'
as.array(x, ...)

## S3 method for class 'internalImage'
x[i, j, ..., drop = TRUE]

## S3 replacement method for class 'internalImage'
x[i, j, ...] <- value
```

Arguments

- **x**: An "internalImage" object.
- **value**: Not used. Changing the dimensions of (or data in) an internal image is invalid, and will produce an error. Convert to an array first.
Additional parameters to methods. Only used for additional indices.

i, j  Index vectors. May be missing, which indicates that the whole of the relevant dimension should be obtained.

drop  If TRUE (the default), unitary indices in the result will be dropped. This mirrors the behaviour of standard array indexing.

Author(s)

Jon Clayden <code@clayden.org>

ndim  Number of dimensions

Description

This function is shorthand for `length(dim(object))`.

Usage

`ndim(object)`

Arguments

object  An R object.

Value

The dimensionality of the object. Objects without a `dim` attribute will produce zero.

Author(s)

Jon Clayden <code@clayden.org>

Examples

`ndim(array(0L, dim=c(10,10)))`
**niftiHeader**

*Dump or construct a raw NIfTI or ANALYZE header*

**Description**

These functions extract the contents of a NIfTI-1 or ANALYZE-7.5 header, closely approximating how it is (or would be) stored on disk. Defaults will be used where information is missing, but no processing is performed on the metadata.

**Usage**

```r
niftiHeader(image = list())
analyzeHeader(image = list())
```

### S3 method for class 'niftiHeader'

```r
print(x, ...)
```

### S3 method for class 'analyzeHeader'

```r
print(x, ...)
```

**Arguments**

- `image`    
  An image, in any acceptable form (see `asNifti`). A list containing partial header information is acceptable, including an empty list, which returns defaults for every field.

- `x`        
  A "niftiHeader" object.

- `...`      
  Ignored.

**Details**

The NIfTI-1 standard was originally formulated as a roughly backwards-compatible improvement on the ANALYZE format. Both formats use a binary header structure of 348 bytes, but the field names and their interpretation is often non-equivalent. These functions dump these fields, without regard to whether or not the result makes proper sense.

`dumpNifti` is an alias of `niftiHeader`, but the former is now soft-deprecated.

**Value**

For `niftiHeader`, a list of class "niftiHeader", with named components corresponding to the elements in a raw NIfTI-1 header. For `analyzeHeader`, the equivalent for ANALYZE-7.5.

**Note**

Several medical image analysis packages, such as SPM and FSL, use the ANALYZE `originator` field to store a coordinate origin. This interpretation is also returned, in the `origin` field.
niftiVersion

Author(s)
Jon Clayden <code@clayden.org>

References
The NIfTI-1 standard (http://www.nitrc.org/docman/view.php/26/64/nifti1).

See Also
niftiVersion

Examples
niftiHeader(system.file("extdata", "example.nii.gz", package="RNifti"))

# Default header for a standard R array
niftiHeader(array(0L, dim=c(10,10)))

Description
This function identifies the likely NIfTI format variant used by one or more files on disk.

Usage
niftiVersion(file)

Arguments
file A character vector of file names.

Value
A vector of integers, of the same length as file. Each element will be 0 for ANALYZE format (the precursor to NIfTI-1), 1 for NIfTI-1 (which is now most common), 2 for NIFTI-2, or -1 if the file doesn’t exist or doesn’t look plausible in any of these formats.

Note
NIFTI-2 format, mostly a variant of NIfTI-1 with wider datatypes used for many fields, is not currently supported for reading, but it is detected by this function.

Author(s)
Jon Clayden <code@clayden.org>
\textbf{pixdim}

\textbf{See Also}

\texttt{readNifti, niftiHeader}

\textbf{Examples}

\begin{verbatim}
path <- system.file("extdata", "example.nii.gz", package="RNifti")
niftiVersion(path)  # 1
\end{verbatim}

---

\textbf{pixdim}  \hspace{1cm} \textit{Pixel dimensions and units}

\textbf{Description}

By default, these generic functions return or replace the "pixdim" and "pixunits" attributes of their arguments. These represent the physical step size between pixel or voxel centre points, and the spatial and temporal units that they are given in. The former defaults to 1 in each dimension, if there is no attribute.

\textbf{Usage}

\begin{verbatim}
pixdim(object)

## Default S3 method:
pixdim(object)

pixdim(object) <- value

## Default S3 replacement method:
pixdim(object) <- value

pixunits(object)

## Default S3 method:
pixunits(object)

pixunits(object) <- value

## Default S3 replacement method:
pixunits(object) <- value
\end{verbatim}

\textbf{Arguments}

\begin{verbatim}
object  \hspace{1cm} An R object, generally an image.
value  \hspace{1cm} Numeric vector of pixel dimensions along each axis, or character vector of abbreviated units. For dimensions, a scalar value will be recycled if necessary.
\end{verbatim}
readNifti

Value

pixdim returns a numeric vector of pixel dimensions. pixunits returns a character vector of length up to two, giving the spatial and temporal unit names.

Author(s)

Jon Clayden <code@clayden.org>

Examples

```r
im <- readNifti(system.file("extdata", "example.nii.gz", package="RNifti"))
pixdim(im)
pixunits(im)
```

---

**readNifti**  
*Read a NIfTI-1 format file*

**Description**

This function reads one or more NIfTI-1 or ANALYZE-7.5 files into R, using the standard NIfTI-1 C library.

**Usage**

```r
readNifti(file, internal = FALSE, volumes = NULL)
```

**Arguments**

- `file`  
  A character vector of file names.

- `internal`  
  Logical value. If FALSE (the default), an array of class "niftiImage", containing the image pixel or voxel values, will be returned. If TRUE, the return value will be an object of class "internalImage", which contains only minimal metadata about the image. Either way, the return value has an attribute which points to a C data structure containing the full image.

- `volumes`  
  An integer vector giving the volumes to read (counting along all dimensions beyond the third jointly), or NULL, the default, in which case every volume is read. This cannot currently be set differently for each file read.

**Value**

An array or internal image, with class "niftiImage" (and possibly also "internalImage"), or a list of such objects if file has length greater than one.
Note

If the `internal` argument is `FALSE` (the default), the data type of the image pointer will be set to match one of R’s native numeric data types, i.e., 32-bit signed integer or 64-bit double-precision floating-point. In these circumstances the data type reported by the `niftiHeader` function will therefore not, in general, match the storage type used in the file. See also the `datatype` argument to `writeNifti`.

Author(s)

Jon Clayden <code@clayden.org>

References

The NIfTI-1 standard (http://www.nitrc.org/docman/view.php/26/64/nifti1.h).

See Also

`writeNifti`

Examples

```r
path <- system.file("extdata", "example.nii.gz", package="RNifti")
readNifti(path)
readNifti(path, internal=TRUE)
```

---

rgbArray

RGB arrays

Description

The `rgbArray` function constructs an integer array whose values are byte-packed representations of 8-bit RGBA colour values. The `channels` attribute (with value 3 or 4) indicates how many channels are being used. The resulting array can be used to construct an RGB(A) NIfTI image, or converted to standard R colour strings using the `as.character` method.

Usage

```r
rgbArray(red, green, blue, alpha, max = NULL, dim = NULL, ...)
```

## S3 method for class 'rgbArray'

```
as.character(x, flatten = TRUE, ...)
```
Arguments

red  
A numeric vector (or array) of red channel values. If this is the only channel argument, it can also be a character vector of colour values (including alpha, if required), or a numeric array whose last dimension is 2 (for grey + alpha), 3 (for RGB) or 4 (for RGBA).

green, blue, alpha  
Numeric vectors (or arrays) containing values for the appropriate channel. These will be combined with the red values using cbind, and hence recycled as necessary. Alpha, or green and blue, can be missing.

max  
The maximum possible value for any channel. The default is 255 when the data is of integer mode, and 1 otherwise. Values above this, or below zero, will be clipped to the appropriate extreme.

dim  
An integer vector of dimensions for the final array. The dimensions of red are used if this is NULL.

...  
For rgbArray, additional attributes to set on the result, such as pixdim. These are passed directly to structure. For the as.character method, this argument is ignored.

x  
An rgbArray object.

flatten  
Logical value. If FALSE, the dimensions of x will be retained in the result. The default is TRUE, for consistency with the usual behaviour of as.character, which strips all attributes.

Value

rgbArray returns an integer-mode array of class "rgbArray". The as.character method returns a character-mode vector of colour strings.

Note

The values of an "rgbArray" are not easily interpreted, and may depend on the endianness of the platform. For manipulation or use as colours they should generally be converted to character mode, or the channels extracted using the channels function.

Author(s)

Jon Clayden <code@clayden.org>

---

view  
A basic 3D image viewer

Description

This function displays one or more 2D or 3D images, with optional click-to-navigate interactivity.
Usage

view(..., point = NULL, radiological = getOption("radiologicalView", FALSE), interactive = base::interactive(), crosshairs = TRUE, labels = TRUE, infoPanel = defaultInfoPanel)

lyr(image, scale = "grey", min = NULL, max = NULL, mask = NULL)

Arguments

... One or more images, or "viewLayer" objects, to display.
point A numeric vector giving the location to initially centre the view on. If crosshairs are in use, they will be placed at this point. For 3D images, this parameter also determines the planes shown in each subview.
radiological Logical value. If TRUE, images will be displayed in the radiological convention whereby the left of the image is shown on the right; otherwise left is on the left.
interactive Logical value. If TRUE, the user can navigate around the image by repeatedly clicking on a new centre point; otherwise the view is fixed.
crosshairs Logical value, indicating whether crosshairs should be shown or not.
labels Logical value, indicating whether orientation labels should be shown or not. Ignored (defaulting to FALSE) if the image is 2D or orientation information is not available.
infoPanel A function of three arguments, which must produce a plot for the information panel of the view. defaultInfoPanel is the default, which shows the labels and values of each image at the current point. A NULL value will suppress the panel.
image The image being shown in this layer.
scale A character vector of colour values for the scale, or a single string naming a predefined scale: "grey" or "gray" for greyscale, "heat" for a heatmap, "rainbow" for a rainbow scale, or any of the scales defined in the shades package (see ?shades::gradient, if that package is installed). A fixed colour can be used by wrapping a string in a call to I. Ignored for RGB images.
min, max The window minimum and maximum for the layer, i.e., the black and white points. These are ignored for RGB images. Otherwise, if NULL, the default, they are taken from the cal_min or cal_max NIfTI header fields. If either is NA, the image has no window stored in its header, or the two values are equal, then the 1st and 99th percentiles of the data are used, with values close to zero rounded to that extreme. If these values are still equal, the untrimmed range of the image data is used.
mask A optional mask array, which may be of lower dimensionality than the main image. If specified, this is converted to logical mode and pixels that evaluate FALSE will be set to NA for that layer, meaning they will not be plotted. This operation is performed last, and so will not affect auto-windowing.

Value

lyr returns a list of class "viewLayer", to be used in a view. view is called for its side-effect of showing a view.
Note
Because of the way R’s main run-loop interacts with graphics, it will not be possible to issue further commands to the terminal while interactive mode is enabled. Instructions for leaving this mode are shown by the default info panel; see also locator, which is the underlying core function.

Author(s)
Jon Clayden <code@clayden.org>

See Also
defaultInfoPanel, orientation, locator

Examples
im <- readNifti(system.file("extdata", "example.nii.gz", package="RNifti"))
view(im, interactive=FALSE)
view(lyr(im, max=800), interactive=FALSE)
view(lyr(im, mask=im<800), interactive=FALSE)

voxelToWorld
Transform points between voxel and "world" coordinates

Description
These functions are used to transform points from dimensionless pixel or voxel coordinates to "real-world" coordinates, typically in millimetres, and back. Actual pixel units can be obtained using the pixunits function. The origin function gives the voxel coordinates of the real-world origin.

Usage
voxelToWorld(points, image, simple = FALSE, ...)
worldToVoxel(points, image, simple = FALSE, ...)
origin(image, ...)

Arguments
points A vector giving the coordinates of a point, or a matrix with one point per row.
image The image in whose space the points are given, or a 4x4 numeric xform matrix.
simple A logical value: if TRUE then the transformation is performed simply by rescaling the points according to the voxel dimensions recorded in the image. Otherwise the full xform matrix is used.
... Additional arguments to xform.
writeNifti

Value

A vector or matrix of transformed points.

Note

Voxel coordinates are assumed by these functions to use R’s indexing convention, beginning from 1.

Author(s)

Jon Clayden <code@clayden.org>

See Also

xform, pixdim, pixunits

Examples

im <- readNifti(system.file("extdata", "example.nii.gz", package="RNifti"))

# Find the origin
origin(im)

writeNifti

Write a NIfTI-1 format file

Description

This function writes an image to NIfTI-1 format, using the standard NIfTI-1 C library.

Usage

writeNifti(image, file, template = NULL, datatype = "auto")

Arguments

image An image, in any acceptable form (see asNifti).
file A character string containing a file name.
template An optional template object to derive NIfTI-1 properties from. Passed to asNifti if image is an array.
datatype The NIfTI datatype to use when writing the data out. The default, "auto" uses the R type or, for internal images, the original datatype. Other possibilities are "float", "int16", etc., which may be preferred to reduce file size. However, no checks are done to ensure that the coercion maintains precision.
Author(s)
Jon Clayden <code@clayden.org>

References
The NIfTI-1 standard (http://www.nitrc.org/docman/view.php/26/64/nifti1.h).

See Also
readNifti, asNifti

Examples
## Not run: writeNifti(im, "image.nii.gz", datatype="float")

---

xform

Obtain or replace the “xform” transforms for an image

Description
These functions convert the “qform” or “sform” information in a NIfTI header to or from a corresponding affine matrix. These two “xform” mechanisms are defined by the NIfTI standard, and may both be in use in a particular image header. They define the relationship between the storage order of the image and real space.

Usage
xform(image, useQuaternionFirst = TRUE)
qform(x) <- value
sform(x) <- value
orientation(x, useQuaternionFirst = TRUE)
orientation(x) <- value
rotation(x, useQuaternionFirst = TRUE)

Arguments
image, x
An image, in any acceptable form (see asNifti), or a 4x4 numeric xform matrix.
useQuaternionFirst
A single logical value. If TRUE, the “qform” matrix will be used first, if it is defined; otherwise the “sform” matrix will take priority.
value
A new 4x4 qform or sform matrix, or orientation string. If a matrix has a "code" attribute, the appropriate qform or sform code is also set.
Details

Image orientation is indicated using a three-character string, with each character indicating the approximate world-space direction of the positive axes in the first, second and third dimensions, in order. Each character may be ‘R’ for left-to-right, ‘L’ for right-to-left, ‘A’ for posterior-to-anterior, ‘P’ for anterior-to-posterior, ‘S’ for inferior-to-superior, or ‘I’ for superior-to-inferior. The default for NIfTI is RAS, meaning that the first dimension points towards the right, the second towards the front and the third towards the top. An xform matrix is an affine transform relative to that default. The upper-left 3x3 matrix in a 3D affine transform governs scale, rotation and skew, while the last column is a translation. (The rotation function extracts the rotation part alone.) The final row is always (0,0,0,1). Reorienting an image involves permuting and possibly reversing some of the axes, both in the data and the metadata. The sense of the translation may also need to be reversed, but this is only possible if the image dimensions are known, which isn’t the case when reorienting an xform alone.

Value

For xform, an affine matrix corresponding to the “qform” or “sform” information in the image header, with a "code" attribute giving the corresponding xform code. For orientation, a string with three characters indicating the (approximate) orientation of the image. The replacement forms return the modified object.

Note

The qform and sform replacement functions are for advanced users only. Modifying the transforms without knowing what you’re doing is usually unwise, as you can make the image object inconsistent.

Author(s)

Jon Clayden <code@clayden.org>

References

The NIfTI-1 standard (http://www.nitrc.org/docman/view.php/26/64/nifti1.h) is the definitive reference on “xform” conventions.

Examples

```r
im <- readNifti(system.file("extdata", "example.nii.gz", package="RNifti"))
xform(im)

# Remove the qform information
qform(im) <- structure(diag(4), code=0L)

# The same as above, since the sform is unmodified
xform(im)

# The identity matrix corresponds to RAS orientation
orientation(diag(4))
```
Description

These methods provide shorthand access to metadata elements from the NIfTI header corresponding to an image. The extraction version returns the corresponding element from the result of `niftiHeader`, while the replacement version calls `asNifti` to replace it.

Usage

```r
## S3 method for class 'niftiImage'
x$name

## S3 replacement method for class 'niftiImage'
x$name <- value
```

Arguments

- **x**: A "niftiImage" object, internal or otherwise.
- **name**: A string naming the field required.
- **value**: A new value for the field.

Author(s)

Jon Clayden <code@clayden.org>

See Also

- `niftiHeader`, `asNifti`

Examples

```r
im <- readNifti(system.file("extdata", "example.nii.gz", package="RNifti"))
print(im$descrip)
```
Index

[.internalImage (dim.internalImage), 5
[<-.internalImage (dim.internalImage), 5
$.niftiImage, 3, 18
$<-.niftiImage ($.niftiImage), 18
analyzeHeader (niftiHeader), 7
as.array.internalImage
  (dim.internalImage), 5
as.character.rgbArray (rgbArray), 11
asNifti, 2, 7, 15, 16, 18
channels, 4, 12
defaultInfoPanel, 4, 13, 14
dim.internalImage, 3, 5
dim<-.internalImage
  (dim.internalImage), 5
dumpNifti (niftiHeader), 7
internalImage (dim.internalImage), 5
locator, 14
lyr (view), 12
ndim, 6
niftiHeader, 2, 3, 7, 9, 11, 18
niftiVersion, 8, 8
orientation, 14
orientation (xform), 16
orientation<-. (xform), 16
origin (voxelToWorld), 14
pixdim, 3, 9, 15
pixdim<-. (pixdim), 9
pixunits, 14, 15
pixunits (pixdim), 9
pixunits<-. (pixdim), 9
print.analyzeHeader (niftiHeader), 7
print.niftiHeader (niftiHeader), 7
qform<-. (xform), 16
readNifti, 3, 9, 10, 16
retrieveNifti (asNifti), 2
rgbArray, 4, 11
rotation (xform), 16
sform<-. (xform), 16
structure, 12
timeSeriesPanel (defaultInfoPanel), 4
updateNifti (asNifti), 2
view, 4, 5, 12
voxelToWorld, 14
worldToVoxel (voxelToWorld), 14
writeNifti, 11, 15
xform, 3, 14, 15, 16