Package 'RNifti'

August 21, 2026

Version 1.10.0

Date 2026-08-07

Title Fast R and C++ Access to NIfTI Images

Imports Rcpp (>= 0.11.0)

Suggests tinytest, covr, reportr, shades, jsonlite

Enhances oro.nifti, tractor.base

LinkingTo Rcpp

Description Provides very fast read and write access to images stored in the NIfTI-1, NIfTI-2 and ANALYZE-7.5 formats, with seamless synchronisation of in-memory image objects 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 and applies spatial transformations.

License GPL-2

URL https://github.com/jonclayden/RNifti

BugReports https://github.com/jonclayden/RNifti/issues

Encoding UTF-8

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Jon Clayden [cre, aut] (ORCID: <https://orcid.org/0000-0002-6608-0619>),
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 2026-08-20 23:12:14 UTC

Contents

asNifti . . . 2
channels . . . 4
defaultInfoPanel . . . 5
dim.internalImage . . . 5
ExtensionCodes . . . 6
extensions . . . 7
fromBidsJson . . . 8
imageAttributes . . . 9
ndim . . . 10
niftiHeader . . . 11
niftiVersion . . . 12
pixdim . . . 13
readNifti . . . 14
rgbArray . . . 16
view . . . 17
voxelToWorld . . . 19
writeNifti . . . 20
xform . . . 22
$.niftiImage . . . 24

Index **25**

`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 `x` will replace those from the reference object, if they are available.

Usage

```
asNifti(x, ...)

## Default S3 method:
asNifti(x, reference = NULL, datatype = "auto", internal = NA, ...)

## S3 method for class 'nifti'
asNifti(x, datatype = "auto", internal = NA, ...)

## S3 method for class 'MriImage'
asNifti(x, datatype = "auto", internal = NA, ...)
```

Arguments

x	Any suitable object (see Details).
...	Additional parameters to methods.
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 match x. However, if a new datatype is set, or the final object contains no pixel data, then this value is implicitly TRUE.

Details

If x has an internal NIfTI pointer, that will be retrieved directly. This will be the case for existing "niftiImage" objects, including internal images. Otherwise, if it is a string, it will be taken to be a filename. A list of the form produced by niftiHeader will convert to an image with no data, while a numeric array or matrix, or RGB array, will be converted using default image parameters.

asNifti is (S3) generic, so methods can be defined for other object types. Methods for "nifti" objects from the oro.nifti package, and "MriImage" objects from tractor.base, are provided.

For the default method, if reference is a niftiHeader object containing a complete list of NIfTI-1 header fields, or an image, it will be used to create the internal object, and then the data and metadata associated with x will overwrite the appropriate parts. If reference is an incomplete list, x 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`

dim.internalImage *Internal images*

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

```
## 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>

ExtensionCodes *NIfTI extension codes*

Description

NIfTI extension codes

Usage

```
ExtensionCodes
```

`extensions` *NIfTI extensions*

Description

The NIfTI-1 and NIfTI-2 formats have a simple extension mechanism that allows additional metadata to be stored with their headers. The format of this extension data is unspecified by the NIfTI standard, but extension codes indicate what type of information is present. These functions provide access to this extension metadata.

Usage

```
extensions(image)

extension(
  image,
  code,
 mode = c("raw", "character", "numeric", "double", "integer", "logical", "complex"),
  ...,
  simplify = TRUE
)

extensions(image) <- value

extension(image, code) <- value
```

Arguments

`image`	An image, in any acceptable form (see `asNifti`).
`code`	Integer value, expression or string specifying which extension code is required.
`mode`	The required mode of the extracted data.
`...`	Additional arguments to `readBin`.
`simplify`	Logical value. If `TRUE`, the default, a single extension will be returned as a vector; otherwise a list is always returned.
`value`	New value for the extension(s).

Details

The plural version, `extensions`, extracts or replaces all extensions at once. The retrieval form returns a list of raw vectors, each with the corresponding code in an attribute, and the replacement form accepts a list of atomic vectors with code attributes, or `NULL`, which removes all extensions. The singular version, `extension`, gets all extensions with the specified code, or appends an extension with that code. Valid extension codes are stored in the `ExtensionCodes` vector.

NIfTI extensions are stored as a simple, unstructured byte stream, which is naturally represented in R as a vector of mode `"raw"`. However, these functions will perform some conversion to and

from other atomic types for convenience. The NIfTI standard makes no guarantees about byte order within the data stream, but the `endian` argument to `readBin` can be passed through when converting to a non-raw type.

Value

For `extensions`, a list of raw vectors containing the bytes stored in each available header. For `extension`, a list of vector of values, converted to the required mode, for the extension code specified. If the extension code is not used in the image, the return value is `NULL`. The replacement forms return the modified image.

Author(s)

Jon Clayden <code@clayden.org>

See Also

`ExtensionCodes` for the valid extension codes.

`fromBidsJson` *Conversion to and from BIDS JSON*

Description

Functions to convert to and from BIDS JSON format for image metadata. They are wrappers around functions from the `jsonlite` package, with the additional ability to convert between the tag naming convention used by the `divest` and `tractor.base` packages and the BIDS equivalent. The differences are mostly in capitalisation, and the units used for magnetic resonance echo, repetition and inversion times.

Usage

```
fromBidsJson(source, rename = FALSE)

toBidsJson(source, path = NULL, rename = FALSE)
```

Arguments

`source`	A list containing metadata (see `imageAttributes`) or, for `fromBidsJson`, a string containing literal JSON or the path to a file containing it.
`rename`	Logical value. If `TRUE`, element names are also converted to or from the BIDS convention; otherwise this is just a conversion between an R list and a JSON string.
`path`	For `toBidsJson`, the path to write the JSON output to. If `NULL`, the default, the JSON text is returned in an object.

Value

`fromBidsJson` returns a list of image attributes. `toBidsJson` returns a character vector if `path` is `NULL`, otherwise nothing.

Note

These functions do not check BIDS metadata for validity in either direction, either in terms of the names of the fields or their contents.

Author(s)

Jon Clayden <code@clayden.org>

References

More information about metadata captured by the BIDS format can be found at https://bids.neuroimaging.io or in the paper cited below.

K.J. Gorgolewski, T. Auer, V.D. Calhoun, et al. The brain imaging data structure, a format for organizing and describing outputs of neuroimaging experiments (2016). Scientific Data 3:160044. doi:10.1038/sdata.2016.44.

---

imageAttributes *Extended image attributes*

Description

These functions extract and replace medical image attributes that go beyond the core metadata associated with the NIfTI file formats.

Usage

```
imageAttributes(x)

imageAttributes(x) <- value
```

Arguments

`x`	A `niftiImage` object.
`value`	A list of new image attributes to replace any existing ones.

Details

The DICOM (Digital Imaging and Communications in Medicine) format, and BIDS (Brain Imaging Data Structure), which extends NIfTI, can both encapsulate copious amounts of metadata about a scan and the patient. This metadata can be useful for more advanced or research-focussed post-processing methods, and standard R attributes are a natural place to store it. The `imageAttributes` function returns a list of just these extended attributes, if they exist, ignoring other attributes used by the package. The replacement form allows this metadata to be modified or removed. These functions currently only act on objects inheriting from the `niftiImage` class.

Value

A list of image attributes, or a modified object with these changed. These are essentially all attributes except those used for basic `niftiImage` objects.

Author(s)

Jon Clayden <code@clayden.org>

References

More information about metadata captured by the BIDS format can be found at https://bids.neuroimaging.io or in the paper cited below.

K.J. Gorgolewski, T. Auer, V.D. Calhoun, et al. The brain imaging data structure, a format for organizing and describing outputs of neuroimaging experiments (2016). Scientific Data 3:160044. doi:10.1038/sdata.2016.44.

See Also

The `divest` package can convert DICOM files to NIfTI formats, and extract embedded metadata. More information about DICOM is available at https://www.dicomstandard.org.

Examples

```
path <- system.file("extdata", "example.nii.gz", package="RNifti")
image <- readNifti(path)
imageAttributes(image)
```

---

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

```
niftiHeader(image = list(), unused = FALSE)

analyzeHeader(image = list())

## S3 method for class 'niftiHeader'
print(x, ...)

## S3 method for class 'analyzeHeader'
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.
`unused`	Logical value. If `TRUE`, legacy ANALYZE and padding fields that are unused by the relevant NIfTI standard are included in the return value. These are occasionally used by software packages.
`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.

Both of these functions call `asNifti` on their arguments to coerce it to NIfTI, except in one specific circumstance: when `analyzeHeader` is called with a single-element character-mode argument that is not an `"internalImage"` object. In this case the string is taken to be a path and the header is reported as stored on disk. This is because otherwise the header may be changed by the process of converting it to NIfTI and back.

Author(s)

Jon Clayden <code@clayden.org>

References

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

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)))
```

`niftiVersion` *Check the format version of a file*

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>

See Also

`readNifti`, `niftiHeader`

Examples

```
path <- system.file("extdata", "example.nii.gz", package="RNifti")
niftiVersion(path)       # 1
```

---

pixdim *Pixel dimensions and units*

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.

Usage

```
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
```

Arguments

object	An R object, generally an image.
value	Numeric vector of pixel dimensions along each axis, or character vector of abbreviated units. For dimensions, a scalar value will be recycled if necessary.

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

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

---

readNifti *Read NIfTI or ANALYZE format files*

Description

This function reads one or more NIfTI-1, NIfTI-2 or ANALYZE-7.5 files into R, using the standard NIfTI C library.

Usage

```
readNifti(
  file,
  internal = FALSE,
  volumes = NULL,
  json = c("ignore", "read", "convert")
)
```

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.
json	A string determining whether any BIDS-style JSON sidecar file is read in alongside the image(s). If "ignore", JSON files are ignored; if "read", they are read and attributes attached to the image using their names in the file; if "convert", element names are additionally converted to the down-cased naming convention of the tractor.base package. The jsonlite package is required for any reading of JSON.

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 (https://www.nitrc.org/docman/view.php/26/64/nifti1.h).

See Also

writeNifti

Examples

```
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. The indexing method returns another object of the same type.

Usage

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

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

## S3 method for class 'rgbArray'
as.raster(x, ...)

## 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 indexing method, additional indices.
x	An rgbArray object.
i, j	Index vectors, which are passed to the array method.
drop	Whether or not to drop unitary dimensions. rgbArray objects currently always have a dim attribute, so if the result is a vector it will have a remaining single-element dimension equal to its length.

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 and the indexing ([) method return an integer-mode array of class "rgbArray". The as.raster method returns a raster object, valid for 2D arrays only. The as.character method returns a character-mode vector of colour strings with or without dimensions.

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,
  alpha = 1,
  label = NULL,
  dict = NULL
)
```

Arguments

...	One or more images, or "viewLayer" objects, to display. These will be plotted in the order specified, so that the first image forms the lowest layer.
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. If a *named* character vector is given instead, it is taken to be a lookup table assigning specific colours to specific numerical values (named by the value, as for dict), for categorical images such as atlases or parcellations; any value without a matching name will not be shown. 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.
alpha	A single opacity value, between 0 (fully transparent) and 1 (fully opaque, the default), to apply to the layer's colour scale. Setting this requires the shades package.
label	An optional character string overriding the default label for the layer, which is otherwise obtained by deparsing the image argument.

`dict` An optional named vector or list giving a dictionary of labels for particular numerical values in the image, for layers whose values are categorical rather than continuous (e.g., an atlas image with regions of interest indicated by integer codes). Names should be the numeric codes, coerced to strings, and values the corresponding labels. If given, these labels will be shown instead of raw numbers in the default information panel.

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.

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 or ANALYZE format file*

Description

These functions write an image to NIfTI-1, NIfTI-2 or ANALYZE-7.5 format, using the standard NIfTI C library.

Usage

```
writeNifti(
  image,
  file,
  template = NULL,
  datatype = "auto",
  version = 1,
  compression = 6,
  json = FALSE
)

writeAnalyze(image, file, template = NULL, datatype = "auto", compression = 6)
```

Arguments

image	An image, in any acceptable form (see asNifti).
file	A character string containing a file name. If this has no file extension suffix, or ends with "nii", the single-file NIfTI format is used; if the suffix is "nii.gz", the compressed single-file format is used; if the suffix is "hdr" or "img", the NIfTI pair two-file format is used; if it's "hdr.gz" or "img.gz", the compressed pair format is used. If any other extension is present it will be ignored, and ".nii" appended.
template	An optional template object to derive NIfTI 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.
version	An integer (1 or 2) giving the NIfTI file format version to use. Version 2 is usually only needed for very large images or where metadata needs to be stored with high precision. The types available for storing the pixel data are the same in both cases.
compression	The gzip compression level to use, an integer between 0 (none) and 9 (maximum). Ignored if an uncompressed format is implied by the requested file name.
json	Logical value. If TRUE, a BIDS-style JSON sidecar file is created alongside the image file(s), containing all image attributes. The jsonlite package is required in this case. No renaming of attributes is performed.

Value

An invisible, named character vector giving the image and header file names written to.

Note

The ANALYZE-7.5 file format is a legacy format and use of it is not recommended, except for compatibility. In particular, the format does not reliably specify the spatial orientation of the image.

Author(s)

Jon Clayden <code@clayden.org>

References

The NIfTI-1 standard (https://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.

Value

For `xform`, an affine matrix corresponding to the "qform" or "sform" information in the image header, with an `"imagedim"` attribute giving the original image dimensions and 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 (https://www.nitrc.org/docman/view.php/26/64/nifti1.h) is the definitive reference on "xform" conventions.

Examples

```
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))
```

`$.niftiImage` *Access to metadata elements*

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

```
## 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

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

Index

[.internalImage *(dim.internalImage)*, 5
[.rgbArray *(rgbArray)*, 16
[<-.internalImage *(dim.internalImage)*, 5
$.niftiImage, *4*, 24
$<-.niftiImage *($.niftiImage)*, 24

analyzeHeader *(niftiHeader)*, 11
as.array.internalImage
(dim.internalImage), 5
as.character.rgbArray *(rgbArray)*, 16
as.raster.rgbArray *(rgbArray)*, 16
asNifti, 2, *7*, *11*, *12*, *21*, *22*, *24*

channels, 4, *17*

defaultInfoPanel, 5, *18*, *19*
dim.internalImage, *4*, 5
dim<-.internalImage
(dim.internalImage), 5
dumpNifti *(niftiHeader)*, 11

extension *(extensions)*, 7
extension<- *(extensions)*, 7
ExtensionCodes, 6, *7*, *8*
extensions, 7
extensions<- *(extensions)*, 7

fromBidsJson, 8

imageAttributes, *8*, 9
imageAttributes<- *(imageAttributes)*, 9
internalImage *(dim.internalImage)*, 5

locator, *19*
lyr *(view)*, 17

ndim, 10
niftiHeader, *3*, 11, *13*, *15*, *24*
niftiVersion, *12*, 12

orientation, *19*
orientation *(xform)*, 22
orientation<- *(xform)*, 22
origin *(voxelToWorld)*, 19

pixdim, *4*, 13, *20*
pixdim<- *(pixdim)*, 13
pixunits, *19*, *20*
pixunits *(pixdim)*, 13
pixunits<- *(pixdim)*, 13
print.analyzeHeader *(niftiHeader)*, 11
print.niftiHeader *(niftiHeader)*, 11

qform<- *(xform)*, 22

readAnalyze *(readNifti)*, 14
readBin, *7*, *8*
readNifti, *4*, *13*, 14, *22*
retrieveNifti *(asNifti)*, 2
rgbArray, *4*, 16
rotation *(xform)*, 22

sform<- *(xform)*, 22
structure, *16*

timeSeriesPanel *(defaultInfoPanel)*, 5
toBidsJson *(fromBidsJson)*, 8

updateNifti *(asNifti)*, 2

view, *5*, 17
voxelToWorld, 19

worldToVoxel *(voxelToWorld)*, 19
writeAnalyze *(writeNifti)*, 20
writeNifti, *15*, 20

xform, *4*, *20*, 22