Package 'oro.nifti'

July 2, 2014

Version 0.4.0

Date 2013-12-23

Title Rigorous - NIfTI+ANALYZE+AFNI Input / Output

Author Brandon Whitcher, Volker Schmid and Andrew Thornton, with contributions by Karsten Tabelow and Jon Clayden

Maintainer Brandon Whitcher <bwhitcher@gmail.com>

Description Functions for the input/output and visualization of medical imaging data that follow either the ANALYZE, NIfTI or AFNI formats. This package is part of the Rigorous Analytics bundle.

Depends R (>= 2.14.0)

Suggests XML

Imports bitops, splines, graphics, grDevices, methods, utils

Enhances fmri

License BSD_3_clause + file LICENSE

URL http://rigorousanalytics.blogspot.com, http://rig.oro.us.com

LazyData yes

ZipData yes

NeedsCompilation no

Repository CRAN

Date/Publication 2013-12-27 16:10:39

R topics documented:

afni-class *Class "afni"*

Description

The AFNI class for medical imaging data.

Objects from the Class

Objects can be created by calls of the form `new("afni", data, dim, dimnames, ...)`.

Slots

`.Data`: Object of class `"array"` contains the imaging data

`DATASET_RANK`: Object of class `"integer"`

`DATASET_DIMENSIONS`: Object of class `"integer"`

`TYPESTRING`: Object of class `"character"`

`SCENE_DATA`: Object of class `"integer"`

`ORIENT_SPECIFIC`: Object of class `"integer"`

`ORIGIN`: Object of class `"numeric"`

`DELTA`: Object of class `"numeric"`

`TAXIS_NUMS`: Object of class `"integer"`

`TAXIS_FLOATS`: Object of class `"numeric"`

`TAXIS_OFFSETS`: Object of class `"numeric"`

`IDCODE_STRING`: Object of class `"character"`

`IDCODE_DATE`: Object of class `"character"`

`BYTEORDER_STRING`: Object of class `"character"`

`BRICK_STATS`: Object of class `"numeric"`

`BRICK_TYPES`: Object of class `"integer"`

`BRICK_FLOAT_FACS`: Object of class `"numeric"`

`BRICK_LABS`: Object of class `"character"`

`BRICK_STATAUX`: Object of class `"numeric"`

`STAT_AUX`: Object of class `"numeric"`

`HISTORY_NOTE`: Object of class `"character"`

`NOTES_COUNT`: Object of class `"integer"`

`NOTE_NUMBER`: Object of class `"character"`

`TAGALIGN_MATVEC`: Object of class `"numeric"`

`VOLREG_MATVEC`: Object of class `"array"`

`VOLREG_ROTCOM`: Object of class `"character"`

VOLREG_CENTER_OLD: Object of class "numeric"

VOLREG_CENTER_BASE: Object of class "numeric"

VOLREG_ROTPARENT_IDCODE: Object of class "character"

VOLREG_ROTPARENT_NAME: Object of class "character"

VOLREG_GRIDPARENT_IDCODE: Object of class "character"

VOLREG_GRIDPARENT_NAME: Object of class "character"

VOLREG_INPUT_IDCODE: Object of class "character"

VOLREG_INPUT_NAME: Object of class "character"

VOLREG_BASE_IDCODE: Object of class "character"

VOLREG_BASE_NAME: Object of class "character"

VOLREG_ROTCOM_NUM: Object of class "integer"

IDCODE_ANAT_PARENT: Object of class "character"

TO3D_ZPAD: Object of class "integer"

IDCODE_WARP_PARENT: Object of class "character"

WARP_TYPE: Object of class "integer"

WARP_DATA: Object of class "numeric"

MARKS_XYZ: Object of class "numeric"

MARKS_LAB: Object of class "character"

MARKS_HELP: Object of class "character"

MARKS_FLAGS: Object of class "integer"

TAGSET_NUM: Object of class "integer"

TAGSET_FLOATS: Object of class "numeric"

TAGSET_LABELS: Object of class "character"

LABEL_1: Object of class "character"

LABEL_2: Object of class "character"

DATASET_NAME: Object of class "character"

DATASET_KEYWORDS: Object of class "character"

BRICK_KEYWORDS: Object of class "character"

Extends

Class "array", from data part.
Class "matrix", by class "array", distance 2, with explicit test and coerce.
Class "structure", by class "array", distance 2.
Class "vector", by class "array", distance 3, with explicit coerce.
Class "vector", by class "array", distance 5, with explicit test and coerce.

Methods

show `signature(object = "afni")`: ...

writeAFNI `signature(nim = "afni")`: ...

Author(s)

Karsten Tabelow <karsten.tabelow@wias-berlin.de>

References

AFNI
http://afni.nimh.nih.gov/pub/dist/src/README.attributes

See Also

nifti, anlz

Examples

```
showClass("afni")
```

---

anlz *Constructor for Analyze*

Description

Constructor for Analyze class objects.

Usage

```
anlz(img = array(0, dim = rep(1, 4)), dim, datatype = 2, ...)
is.anlz(x)
```

Arguments

img	is a multidimensional array of data.
dim	is the dimension of the data (default = missing).
datatype	is an integer that denotes the type of data contained in each voxel. See convert.datatype.anlz or the ANALYZE documentation for more details.
...	allows for additional 'slots' to be specified.
x	is an object to be checked.

Value

An object of class anlz.

Author(s)

Brandon Whitcher <bwhitcher@gmail.com>

References

ANALYZE 7.5
http://www.mayo.edu/bir/PDF/ANALYZE75.pdf

See Also

anlz, nifti, nifti, convert.datatype.anlz

Examples

```
(aim <- anlz()) # default
```

---

anlz-class *Class "anlz"*

Description

The ANALYZE class for medical imaging data.

Objects from the Class

Objects can be created by calls of the form `new("anlz", data, dim, dimnames, ...)` or by calling the `anlz` function.

Slots

`.Data`: Object of class `"array"` contains the imaging data

`sizeof_hdr`: Object of class `"numeric"` contains the size of the header (= 348)

`data_type`: Object of class `"character"`

`db_name`: Object of class `"character"`

`extents`: Object of class `"numeric"`

`session_error`: Object of class `"numeric"`

`regular`: Object of class `"character"`

`hkey_un0`: Object of class `"character"`

`dim_`: Object of class `"vector"` contains the dimensions of the imaging data

`vox_units`: Object of class `"character"`

`cal_units`: Object of class `"character"`

`unused1`: Object of class `"numeric"`

`datatype`: Object of class `"numeric"`

`bitpix`: Object of class `"numeric"` contains the number of bits per voxel (pixel)

`dim_un0`: Object of class `"numeric"`

`pixdim`: Object of class `"vector"` contains the real-world dimensions of the imaging data

`vox_offset`: Object of class `"numeric"`
`funused1`: Object of class `"numeric"`
`funused2`: Object of class `"numeric"`
`funused3`: Object of class `"numeric"`
`cal_max`: Object of class `"numeric"` contains the maximum display intensity
`cal_min`: Object of class `"numeric"` contains the minimum display intensity
`compressed`: Object of class `"numeric"`
`verified`: Object of class `"numeric"`
`glmax`: Object of class `"numeric"`
`glmin`: Object of class `"numeric"`
`descrip`: Object of class `"character"`
`aux_file`: Object of class `"character"`
`orient`: Object of class `"character"`
`origin`: Object of class `"numeric"`
`generated`: Object of class `"character"`
`scannum`: Object of class `"character"`
`patient_id`: Object of class `"character"`
`exp_date`: Object of class `"character"`
`exp_time`: Object of class `"character"`
`hist_un0`: Object of class `"character"`
`views`: Object of class `"numeric"`
`vols_added`: Object of class `"numeric"`
`start_field`: Object of class `"numeric"`
`field_skip`: Object of class `"numeric"`
`omax`: Object of class `"numeric"`
`omin`: Object of class `"numeric"`
`smax`: Object of class `"numeric"`
`smin`: Object of class `"numeric"`

Extends

Class `"array"`, from data part.
Class `"matrix"`, by class "array", distance 2, with explicit test and coerce.
Class `"structure"`, by class "array", distance 2.
Class `"vector"`, by class "array", distance 3, with explicit coerce.
Class `"vector"`, by class "array", distance 5, with explicit test and coerce.

Methods

descrip<- `signature(x = "anlz")`: replaces the "description" field
descrip `signature(object = "anlz")`: returns the "description" field
image `signature(x = "anlz")`: diplays the image(s)
show `signature(object = "anlz")`: prints out a summary of the imaging data

Author(s)

Brandon Whitcher <bwhitcher@gmail.com>

References

ANALYZE 7.5
http://www.mayo.edu/bir/PDF/ANALYZE75.pdf

See Also

nifti, niftiExtension

Examples

```
showClass("anlz")
```

---

as.anlz/nifti *as.nifti*

Description

Internal function that converts multidimensional arrays to NIfTI class objects.

Usage

```
as.anlz(from, value = NULL, verbose = FALSE)
as.nifti(from, value = NULL, verbose = FALSE)
```

Arguments

from	is the object to be converted.
value	is the nifti class object to use as a template for various ANALYZE/NIfTI header information.
verbose	is a logical variable (default = FALSE) that allows text-based feedback during execution of the function.

Value

An object of class anlz or nifti.

Author(s)

Andrew Thornton <zeripath@users.sourceforge.net> and Brandon Whitcher <bwhitcher@gmail.com>

`Audit Trails` *Facilitate the Creation and Modification of Audit Trails*

Description

Facilitate the creation and modification of audit trails for NIfTI class objects.

Usage

```
oro.nifti.info(type)
enableAuditTrail()
newAuditTrail()
niftiExtensionToAuditTrail(nim, workingDirectory = NULL,
                           filename = NULL, call = NULL)
niftiAuditTrailToExtension(nim, workingDirectory = NULL,
                           filename = NULL, call = NULL)
niftiAuditTrailSystemNode(type="system-info",
                          workingDirectory = NULL,
                          filename = NULL, call = NULL)
niftiAuditTrailSystemNodeEvent(trail, type = NULL, call = NULL,
                               workingDirectory = NULL,
                               filename = NULL, comment=NULL)
niftiAuditTrailCreated(history = NULL, call = NULL,
                       workingDirectory = NULL, filename = NULL)
niftiAuditTrailEvent(trail, type = NULL, call = NULL,
                     comment = NULL)
getLastCallWithName(functionName)
```

Arguments

`nim`	is an object of class `niftiAuditTrail` or can be converted to such.
`workingDirectory`	The working directory associated with the 'filename'.
`filename`	The filename associated with the nifti object.
`call`	A `call`, function name in the call-stack or a string.
`type`	An identifier to add some meaning to the event.
`trail`	The `XMLAbstractNode` representing the audit trail or the `niftiAuditTrail` object with a trail that will be amended.
`comment`	Some textual comment
`history`	An `XMLAbstractNode` to store historical events for inclusion in the 'trail'.
`functionName`	The name of a function on the call stack.

Details

The function oro.nifti.info is used to find the ecode or the XML namespace relevant to the audit trail.

The function enableAuditTrail is turned "off" by default to minimize package dependencies. Should one wish to turn "on" the audit trail functionality, then one should set the option NIfTI.audit.trail to TRUE and call the function enableAuditTrail. Setting the option NIfTI.audit.trail to FALSE will disable the audit trail.

The function newAuditTrail returns an XMLAbstractNode representing the root node of an audit trail. This is mostly intended as an internal function.

The function niftiExtensionToAuditTrail takes an object representing a NIfTI object, casts it as a niftiAuditTrail and checks if there is an extension (a niftiExtensionSection) with ecode equal to oro.nifti.info("ecode"); i.e. has a extension with data representing a serialized audit trail. The function will then strip the object of this extension parsing the serialized edata into an audit trail and adding a 'read' event to the trail.

The function niftiAuditTrailToExtension takes a niftiAuditTrail and returns a niftiExtensionSection with edata containing the serialized form of the audit trail after adding a 'saved' event to the trail.

The function niftiAuditTrailSystemNodeEvent adds an element with name equal to type to the trail. It uses the niftiAuditTrailSystemNode function to create the node.

The function niftiAuditTrailSystemNode is an internal function creating an XMLAbstractNode element with name type and attributes giving information about the R system and library. The filename and call will also be added as attributes if available.

The function niftiAuditTrailEvent adds an element with name event to the trail. The arguments type, filename, call are added as attributes and the comment is the text value of the element.

The function niftiAuditTrailCreated will create a new audit trail containing a system node element created with the child history with the contents history. If the last element of the history given is an event with type="processing", then this node will be removed from the history and its call attribute will be used as the value of the call attribute on the created node.

The function getLastCallWithName will search the call stack for a call of the function functionName, returning last call to that function if possible. It will default to the call of the function which called the function which called getLastCallWithName if there was no such call (and if there was no such call it will return the call of itself).

Note

These functions are mostly intended to be used internally in order to document the changes that occur to NIfTI objects due to functions that are audit-trail aware. However, as the precise manner in which these functions are used is not documented anywhere else, we shall proceed to describe which functions are audit-trail aware and how they interact with the audit trail.

as.nifti and its S4 alias as(nim, "nifti") will always produce niftiAuditTrail objects if the functionality is turned on. The function niftiAuditTrailCreated will be used and if an exemplar object is provided (e.g., as.nifti(array, niftiExemplar)) then the trail of the exemplar will be used as the history.

readNIfTI and writeNIfTI also always produce niftiAuditTrail objects if the functionality is turned on. The functions niftiExtensionToAuditTrail and niftiAuditTrailToExtension are used internally by these functions to facilitate this behaviour.

Author(s)

Andrew Thornton <zeripath@users.sourceforge.net> and Brandon Whitcher <bwhitcher@gmail.com>

Examples

```
## A good example of the use of these functions is shown by this
## wrapper function which takes a function fun(nim, ...) returning
## lists of arrays which are nifti-ized using as(...)
options("niftiAuditTrail"=TRUE)
enableAuditTrail()

wrapper <- function(functionToWrap, nameOfCallingFunction, nim, ...) {
  if (!is(nim, "nifti"))
    nim <- as(nim, "nifti")

  if (is(nim, "niftiAuditTrail")) {
    ## This will force as(...) to set the call which created the
    ## results to the calling function's call rather than
    ## as(result, nifti) as it would otherwise do
    nim@trail <- niftiAuditTrailEvent(nim@trail, "processing",
                                      nameOfCallingFunction)
  }

  result <- functionToWrap(nim, ...)
  as(result, "nifti") <- nim
  return(result)
}

## An example of how wrapper is used follows:
functionToWrap <- function(ignored, x, y) {
  return (array(1, dim=c(x,y)))
}

## The niftiized form
niftiizedForm <- function(nim,...) {
  return(wrapper(functionToWrap, "niftiizedForm", nim, ...))
}

## Not run:
## compare the trails
if (isTRUE(getOption("niftiAuditTrail"))) {
  print((as.nifti(functionToWrap(nifti(), 4, 4), nifti()))@trail)
  print(niftiizedForm(nifti(), 4, 4)@trail)
}

## End(Not run)
```

audit.trail-methods *Extract or Replace NIfTI Audit Trail*

Description

Operators that act on the audit trail (XML) in the NIfTI header.

Usage

```
## S4 method for signature 'nifti'
audit.trail(object)
```

Arguments

object	is of class nifti.

Methods

object = "nifti" Extract or replace NIfTI audit trail.

Author(s)

Andrew Thornton <zeripath@users.sourceforge.net>

Examples

```
## Not run:
## Sternberg Item Recognition Paradigm (SIRP) fMRI Study + XML
## Extension Data
URL <- "http://nifti.nimh.nih.gov/nifti-1/data/sirp_fmri_study_ver4.tar.gz"
download.file(URL, dest="sirp.tar.gz", quiet=TRUE)
fnames <- system("tar zxvf sirp.tar.gz", intern=TRUE)
sirp <- readNIfTI(fnames[1]) # newsirp_final_XML.nii
(sirp.xml <- xmlTreeParse(sirp@"extensions"[[1]]@"edata", asText=TRUE))

## End(Not run)
```

aux.file-methods *Extract or Replace NIfTI/Analyze Auxiliary File*

Description

Methods that act on the "auxiliary file" character string in the NIfTI or Analyze header.

Usage

```
## S4 method for signature 'nifti'
aux.file(object)
## S4 method for signature 'anlz'
aux.file(object)
```

Arguments

`object`	is an object of class `nifti` or `anlz`.

Methods

object = "anlz" Extract or replace Analyze auxiliary file.

object = "nifti" Extract or replace NIfTI auxiliary file.

Author(s)

Brandon Whitcher <bwhitcher@gmail.com>

Examples

```
## Not run:
url <- "http://nifti.nimh.nih.gov/nifti-1/data/avg152T1_RL_nifti.nii.gz"
urlfile <- file.path(system.file("nifti", package="oro.nifti"),
                     "mniRL.nii.gz")
download.file(url, urlfile, quiet=TRUE)

## End(Not run)
options("niftiAuditTrail"=FALSE)

urlfile <- file.path(system.file("nifti", package="oro.nifti"),
                     "mniRL.nii.gz")
mniRL <- readNIfTI(urlfile)
aux.file(mniRL)
aux.file(mniRL) <- "avg152T1_RL_nifti"
aux.file(mniRL)
```

`Bitwise Conversion Subroutines`

Bitwise Conversion Subroutines

Description

Units of spatial and temporal dimensions, and MRI-specific spatial and temporal information.

Usage

```
xyzt2space(xyzt)
xyzt2time(xyzt)
space.time2xyzt(ss, tt)
dim2freq(di)
dim2phase(di)
dim2slice(di)
```

Arguments

xyzt	represents the units of pixdim[1..4] in the NIfTI header.
ss	is the character string of spatial units. Valid strings are: "Unknown", "meter", "mm" and "micron".
tt	is the character string of temporal units. Valid strings are: "sec", "msec", "usec", "Hz", "ppm" and "rads".
di	represents MRI slice ordering in the NIfTI header.

Details

The functions xyzt2space and xyzt2time can be used to mask off the undesired bits from the xyzt_units fields, leaving "pure" space and time codes.
http://nifti.nimh.nih.gov/nifti-1/documentation/nifti1fields/nifti1fields_pages/xyzt_units.html

The functions dim2freq, dim2phase, and dim2slice can be used to extract values from the dim_info byte.
http://nifti.nimh.nih.gov/nifti-1/documentation/nifti1fields/nifti1fields_pages/dim_info.html

Value

For diminfo: the frequency, phase and slice dimensions encode which spatial dimension (1,2, or 3) corresponds to which acquisition dimension for MRI data. For xyzt_units: the codes are used to indicate the units of pixdim. Dimensions 1,2,3 are for x,y,z; dimension 4 is for time (t).

Author(s)

B. Whitcher <bwhitcher@gmail.com>

References

Neuroimaging Informatics Technology Initiative (NIfTI)
http://nifti.nimh.nih.gov/

See Also

convert.units, convert.slice

cal.min-methods *Extract or Replace NIfTI/Analyze Min or Max Values*

Description

Methods that act on the 'cal.min' and 'cal.max' numeric value in the NIfTI or ANALYZE header.

Usage

```
## S4 method for signature 'nifti'
cal.min(object)
## S4 method for signature 'nifti'
cal.max(object)
```

Arguments

object	is an object of class nifti or anlz.

Methods

object = "anlz" Extract or replace the ANALYZE "cal_min" or "cal_max" value.

object = "nifti" Extract or replace the NIfTI "cal_min" or "cal_max" value.

Author(s)

Brandon Whitcher <bwhitcher@gmail.com>

Examples

```
## Not run:
url <- "http://nifti.nimh.nih.gov/nifti-1/data/avg152T1_LR_nifti.nii.gz"
urlfile <- file.path(system.file("nifti", package="oro.nifti"),
                     "mniLR.nii.gz")
download.file(url, urlfile, quiet=TRUE)

## End(Not run)
urlfile <- file.path(system.file("nifti", package="oro.nifti"),
                     "mniLR.nii.gz")
mniLR <- readNIfTI(urlfile)
cal.min(mniLR)
cal.max(mniLR)
```

coerce-methods *Force an Object to Belong to the ANALYZE or NIfTI Class*

Description

Methods for function coerce in Package 'methods'.

Usage

```
## S4 method for signature 'array,anlz'
as(object, Class)
## S4 replacement method for signature 'array,anlz'
as(object, Class) <- value
## S4 method for signature 'array,nifti'
as(object, Class)
## S4 replacement method for signature 'array,nifti'
as(object, Class) <- value
```

Arguments

object	is an object of class array or inherits from array.
Class	is the name of the class to which 'object' should be coerced; i.e., nifti.
value	is the values used to modify 'object' (see the discussion below). You should supply an object with class nifti in order to pass NIfTI header information.

Details

If

Methods

from = "anlz", to = "nifti" An object of class anlz is coerced into a NIfTI object.

from = "array", to = "anlz" An object of class array is coerced into an ANALYZE object.

from = "array", to = "nifti" An object of class array is coerced into a NIfTI object.

from = "list", to = "anlz" All objects of class array in the list are coerced into ANALYZE objects. All other objects are left alone. The original list structure is retained.

from = "list", to = "nifti" All objects of class array in the list are coerced into NIfTI objects. All other objects are left alone. The original list structure is retained.

Author(s)

Andrew Thornton <zeripath@users.sourceforge.net> and Brandon Whitcher <bwhitcher@gmail.com>

See Also

as

`Convert ANALYZE Codes` *Convert NIfTI Codes*

Description

Codes that appear in the ANALYZE header are mapped to meaningful chartacter strings.

Usage

```
convert.bitpix.anlz(bitpix)
convert.datatype.anlz(datatype.code)
convert.orient.anlz(orientation)
```

Arguments

`bitpix`	is the bit-per-pixel code.
`datatype.code`	defines data type.
`orientation`	defines the orientation.

Details

`switch` statements are used to map a numeric code to the appropriate string.

Value

A character string.

Author(s)

Brandon Whitcher <bwhitcher@gmail.com>

References

ANALYZE 7.5
http://www.mayo.edu/bir/PDF/ANALYZE75.pdf

See Also

`convert.datatype`, `convert.bitpix`, `convert.intent`, `convert.form`, `convert.units`, `convert.slice`

Examples

```
##  4 = SIGNED_SHORT
convert.datatype.anlz(4)
## 16 = FLOAT
convert.datatype.anlz(16)
##  2 = "saggital unflipped"
convert.orient.anlz(2)
```

```
##  4 = "coronal flipped"
convert.orient.anlz(4)
```

Convert Between fmridata and oro.nifti

Convert Between fmridata and oro.nifti Objects

Description

NIfTI data can be converted between `fmridata` S3 objects (from the **fmri** package) and `nifti` S4 objects.

Usage

```
oro2fmri(from, value = NULL, level = 0.75, setmask = TRUE)
fmri2oro(from, value = NULL, verbose = FALSE, reorient = FALSE,
         call = NULL)
```

Arguments

`from`	is the object to be converted.
`value`	`NULL`
`level`	is the quantile level defining the mask.
`setmask`	is a logical variable (default = `TRUE`), whether to define a suitable mask based on `level`.
`verbose`	is a logical variable (default = `FALSE`) that allows text-based feedback during execution of the function.
`reorient`	is a logical variable (default = `TRUE`) that enforces Qform/Sform transformations.
`call`	keeps track of the current function call for use in the NIfTI extension.

Details

These functions enhance the capabilities of **fmri** by allowing the exchange of data objects between `nifti` and `fmridata` classes.

Value

The function `oro2fmri` produces an S3 object of class `fmridata`. The function `fmri2oro` produces an S4 object of class `nifti`.

Author(s)

Brandon Whitcher <bwhitcher@gmail.com>

See Also

`readNIfTI`, `read.NIFTI`

`Convert NIfTI Codes` *Convert NIfTI Codes*

Description

Codes that appear in the NIfTI-1 header are mapped to meaningful chartacter strings.

Usage

```
convert.datatype(datatype.code = NULL)
convert.bitpix(bitpix = NULL)
convert.intent(intent.code = NULL)
convert.form(form.code)
convert.units(units, inverse = FALSE)
convert.slice(slice.code)
```

Arguments

`datatype.code`	defines data type.
`bitpix`	is the bit-per-pixel code.
`intent.code`	is the NIfTI intent code.
`form.code`	is the (x, y, z) coordinate system.
`units`	is the units of pixdim[1..4].
`inverse`	is a logical value that denotes the direction of unit conversion.
`slice.code`	is the slice timing order.

Details

`switch` statements are used to map a numeric code to the appropriate string.

Value

A character string.

Author(s)

Brandon Whitcher `<bwhitcher@gmail.com>`

References

Neuroimaging Informatics Technology Initiative (NIfTI)
http://nifti.nimh.nih.gov/

See Also

`convert.datatype.anlz`, `convert.orient.anlz`

Examples

```
## No arguments produces a list structure
## Use the abbreviation to obtain the code
convert.datatype()$INT16
## Use the code directly to obtain the abbreviation
convert.datatype(4)
## No arguments produces a list structure
## Use the abbreviation to obtain the code
convert.bitpix()$INT16
## No arguments produces a list structure
## Use the abbreviation to obtain the code
convert.intent()$Estimate
## Use the code directly to obtain the abbreviation
convert.intent(1001)
## 4 = MNI_152
convert.form(4)
## 2 = mm
convert.units(2)
## 8 = sec
convert.units(8)
## 0 = Unknown
convert.slice(0)
```

---

convert.scene *Convert AFNI data codes*

Description

Codes that appear in the AFNI header are mapped to meaningful character strings.

Usage

```
convert.scene(scene.data, typestring)
```

Arguments

scene.data	defines data type.
typestring	defines whether func or anat data.

Details

switch statements are used to map a numeric code to the appropriate string.

Value

A character string.

Author(s)

Karsten Tabelow <karsten.tabelow@wias-berlin.de>

References

AFNI
http://afni.nimh.nih.gov/pub/dist/src/README.attributes

See Also

convert.datatype.anlz, convert.orient.anlz

Examples

```
## 4 = CT for anatomic data
convert.scene(4, "3DIM_HEAD_ANAT")
```

---

descrip-methods *Extract or Replace NIfTI/Analyze Description*

Description

Methods that act on the "description" character string in the NIfTI or ANALYZE header.

Usage

```
## S4 method for signature 'nifti'
descrip(object)
```

Arguments

object	is an object of class nifti or anlz.

Methods

object = "anlz" Extract or replace Analyze description.

object = "nifti" Extract or replace NIfTI description.

Author(s)

Brandon Whitcher <bwhitcher@gmail.com>

Examples

```
## Not run:
url <- "http://nifti.nimh.nih.gov/nifti-1/data/avg152T1_LR_nifti.nii.gz"
urlfile <- file.path(system.file("nifti", package="oro.nifti"),
                     "mniLR.nii.gz")
download.file(url, urlfile, quiet=TRUE)

## End(Not run)
```

```
urlfile <- file.path(system.file("nifti", package="oro.nifti"),
                     "mniLR.nii.gz")
mniLR <- readNIfTI(urlfile)
descrip(mniLR)
descrip(mniLR) <- paste(descrip(mniLR), version$version.string, sep="; ")
descrip(mniLR)
```

Dimension Accessor Functions

Dimension Accessor Functions

Description

Functions to extract the higher dimensions from ANALYZE/NIfTI data.

Usage

```
nsli(x)
ntim(x)
NSLI(x)
NTIM(x)
```

Arguments

x	is a three- or four-dimensional array (e.g., read in from an ANALYZE/NIfTI file).

Details

Simple calls to `dim` to replicate the functionality of `nrow` and `ncol` for higher dimensions of an array that are commonly required when manipulating medical imaging data.

Value

Third (slice) or fourth (time) dimension of the array.

Author(s)

Brandon Whitcher <bwhitcher@gmail.com>

See Also

`readNIfTI`, `readANALYZE`

hotmetal *Hot Metal Color Table*

Description

The hotmetal color table patterned after the one used in Matlab.

Usage

```
hotmetal(n=64)
```

Arguments

n	is the number of color levels (default = 64).

Details

Based on the `tim.colors` function in the **fields** package. The `hotmetal` function has been modified to break any dependence on code in the **fields** package. Spline interpolation (`interpSpline`) is used when the number of requested colors is not the default.

Value

A vector of character strings giving the colors in hexadecimal format.

See Also

`terrain.colors`, `tim.colors`, `topo.colors`

Examples

```
hotmetal(10)
image(outer(1:20,1:20,"+"), col=hotmetal(75), main="hotmetal")
```

image-methods *Methods for Function 'image'*

Description

Produce "lightbox" layout of images for `nifti`, `anlz` and `afni` objects.

Usage

```
## S4 method for signature 'nifti'
image(x, z=1, w=1, col=gray(0:64/64),
      plane=c("axial", "coronal", "sagittal"),
      plot.type=c("multiple","single"), zlim=NULL, xlab="",
      ylab="", axes=FALSE, oma=rep(0,4), mar=rep(0,4),
      bg="black", ...)
```

Arguments

x	is an object of class nifti or similar.
z	is the slice to be displayed (ignored when plot.type = "multiple").
w	is the time point to be displayed (4D arrays only).
col	is grayscale (by default).
plane	is the plane of acquisition to be displayed (choices are 'axial', 'coronal', 'sagittal').
plot.type	allows the choice between all slices being displayed, in a matrix (left-to-right, top-to-bottom), or a single slice.
zlim	is set to NULL by default and utilizes the internal image range.
xlab	is set to "" since all margins are set to zero.
ylab	is set to "" since all margins are set to zero.
axes	is set to FALSE since all margins are set to zero.
oma	is the size of the outer margins in the par function.
mar	is the number of lines of margin in the par function.
bg	is the background color in the par function.
...	other arguments to the image function may be provided here.

Details

Uses the S3 generic function image, with medical-image friendly settings, to display nifti, anlz and afni class objects in a "lightbox" layout.

Methods

x = "ANY" Generic function: see image.

x = "nifti" Produce images for x.

x = "anlz" Produce images for x.

x = "afni" Produce images for x.

Author(s)

Brandon Whitcher <bwhitcher@gmail.com>

See Also

orthographic-methods, overlay-methods

---

integerTranslation *integerTranslation*

Description

...

Usage

```
integerTranslation(nim, data, verbose = FALSE)
invertIntegerTranslation(nim, verbose = FALSE)
```

Arguments

nim	is an object of class nifti.
data	is ...
verbose	is a logical variable (default = FALSE) that allows text-based feedback during execution of the function.

Details

...

Value

...

Author(s)

Andrew Thornton <zeripath@users.sourceforge.net>

`is.afni` *is.afni*

Description

check object

Usage

```
is.afni(x)
```

Arguments

x	is an object to be checked.

Details

Check whether object is of class `afni`

Value

Logical indicating whether object is of class `afni`

Author(s)

Karsten Tabelow <karsten.tabelow@wias-berlin.de>

References

AFNI
http://afni.nimh.nih.gov/pub/dist/src/README.attributes

See Also

`afni`

`nifti` *Constructor for NIfTI*

Description

Constructor for NIfTI class objects.

Usage

```
nifti(img = array(0, dim = rep(1, 4)), dim, datatype = 2,
      cal.min = NULL, cal.max = NULL, pixdim = NULL, ...)
is.nifti(x)
```

Arguments

`img`	is a multidimensional array of data.
`dim`	is the dimension of the data (default = `missing`).
`datatype`	is an integer that denotes the type of data contained in each voxel. See `convert.datatype` or the NIfTI documentation for more details.
`cal.min`	allows user-specified minimum value in the array (visualization purposes only).
`cal.max`	allows user-specified minimum value in the array (visualization purposes only).
`pixdim`	allows user-specified pixel dimension vector (length = 8).
`...`	allows for additional 'slots' to be specified.
`x`	is an object to be checked.

Value

An object of class `nifti`.

Author(s)

Brandon Whitcher <bwhitcher@gmail.com>

References

NIfTI-1
http://nifti.nimh.nih.gov/

See Also

`nifti`, `anlz`, `convert.datatype`

Examples

```
options("niftiAuditTrail"=FALSE)

nim <- nifti() # default
nim
nim <- nifti(datatype=4) # 2-byte integers
nim
```

`nifti-class` *Class "nifti"*

Description

The NIfTI class for medical imaging data.

Objects from the Class

Objects can be created by calls of the form `new("nifti", data, dim, dimnames, ...)` or by calling the `nifti` function.

Slots

`.Data`: Object of class `"array"` contains the imaging data

`sizeof_hdr`: Object of class `"numeric"` contains the size of the header (= 348)

`data_type`: Object of class `"character"`

`db_name`: Object of class `"character"`

`extents`: Object of class `"numeric"`

`session_error`: Object of class `"numeric"`

`regular`: Object of class `"character"`

`dim_info`: Object of class `"numeric"` contains MRI slice ordering

`dim_`: Object of class `"vector"` contains the dimensions of the imaging data

`intent_p1`: Object of class `"numeric"`

`intent_p2`: Object of class `"numeric"`

`intent_p3`: Object of class `"numeric"`

`intent_code`: Object of class `"numeric"`

`datatype`: Object of class `"numeric"`

`bitpix`: Object of class `"numeric"` contains the number of bits per voxel (pixel)

`slice_start`: Object of class `"numeric"`

`pixdim`: Object of class `"vector"` contains the real-world dimensions of the imaging data

`vox_offset`: Object of class `"numeric"` contains the voxel offset (= 352 when no extensions exist)

`scl_slope`: Object of class `"numeric"`

`scl_inter`: Object of class "numeric"

`slice_end`: Object of class "numeric"

`slice_code`: Object of class "numeric"

`xyzt_units`: Object of class "numeric"

`cal_max`: Object of class "numeric" contains the maximum display intensity

`cal_min`: Object of class "numeric" contains the minimum display intensity

`slice_duration`: Object of class "numeric"

`toffset`: Object of class "numeric"

`glmax`: Object of class "numeric"

`glmin`: Object of class "numeric"

`descrip`: Object of class "character"

`aux_file`: Object of class "character"

`qform_code`: Object of class "numeric"

`sform_code`: Object of class "numeric"

`quatern_b`: Object of class "numeric"

`quatern_c`: Object of class "numeric"

`quatern_d`: Object of class "numeric"

`qoffset_x`: Object of class "numeric"

`qoffset_y`: Object of class "numeric"

`qoffset_z`: Object of class "numeric"

`srow_x`: Object of class "vector"

`srow_y`: Object of class "vector"

`srow_z`: Object of class "vector"

`intent_name`: Object of class "character"

`magic`: Object of class "character"

`extender`: Object of class "vector"

`reoriented`: Object of class "logical"

Extends

Class "array", from data part.
Class "matrix", by class "array", distance 2, with explicit test and coerce.
Class "structure", by class "array", distance 2.
Class "vector", by class "array", distance 3, with explicit coerce.
Class "vector", by class "array", distance 5, with explicit test and coerce.

Methods

aux.file<- `signature(x = "nifti")`: replaces the "auxiliary file" field

aux.file `signature(object = "nifti")`: returns the "auxiliary file" field

descrip<- `signature(x = "nifti")`: replaces the "description" field

descrip `signature(object = "nifti")`: returns the "description" field

image `signature(x = "nifti")`: diplays the image(s)

orthographic `signature(x = "nifti")`: displays the image(s)

overlay `signature(x = "nifti", y = "nifti")`: displays the image(s)

show `signature(object = "nifti")`: prints out a summary of the imaging data

Author(s)

Brandon Whitcher <bwhitcher@gmail.com>

References

NIfTI-1
http://nifti.nimh.nih.gov/

See Also

anlz, niftiExtension, niftiAuditTrail

Examples

```
showClass("nifti")
```

`niftiAuditTrail-class` *Class "niftiAuditTrail"*

Description

An extension of the NIfTI class that adds an audit trail in XML format.

Objects from the Class

Objects can be created by calls of the form `new("niftiAuditTrail", data, dim, dimnames, ...)`.

Slots

`.Data`: Object of class "array"

`trail`: Object of class "XMLAbstractNode" contains the XML version of the audit trail.

`extensions`: Object of class "list" contains the list of all extensions.

`sizeof_hdr`: Object of class "numeric"

`data_type`: Object of class "character"

`db_name`: Object of class "character"

`extents`: Object of class "numeric"

`session_error`: Object of class "numeric"

`regular`: Object of class "character"

`dim_info`: Object of class "numeric"

`dim_`: Object of class "vector"

`intent_p1`: Object of class "numeric"

`intent_p2`: Object of class "numeric"

`intent_p3`: Object of class "numeric"

`intent_code`: Object of class "numeric"

`datatype`: Object of class "numeric"

`bitpix`: Object of class "numeric"

`slice_start`: Object of class "numeric"

`pixdim`: Object of class "vector"

`vox_offset`: Object of class "numeric"

`scl_slope`: Object of class "numeric"

`scl_inter`: Object of class "numeric"

`slice_end`: Object of class "numeric"

`slice_code`: Object of class "numeric"

`xyzt_units`: Object of class "numeric"

`cal_max`: Object of class "numeric"

`cal_min`: Object of class "numeric"

`slice_duration`: Object of class "numeric"

`toffset`: Object of class "numeric"

`glmax`: Object of class "numeric"

`glmin`: Object of class "numeric"

`descrip`: Object of class "character"

`aux_file`: Object of class "character"

`qform_code`: Object of class "numeric"

`sform_code`: Object of class "numeric"

`quatern_b`: Object of class "numeric"

`quatern_c`: Object of class "numeric"

`quatern_d`: Object of class "numeric"

`qoffset_x`: Object of class "numeric"

`qoffset_y`: Object of class "numeric"

`qoffset_z`: Object of class "numeric"

`srow_x`: Object of class "vector"

`srow_y`: Object of class "vector"

`srow_z`: Object of class "vector"

`intent_name`: Object of class "character"

`magic`: Object of class "character"

`extender`: Object of class "vector"

`reoriented`: Object of class "logical"

Extends

Class "niftiExtension", directly.
Class "nifti", by class "niftiExtension", distance 2.
Class "array", by class "niftiExtension", distance 3.
Class "matrix", by class "niftiExtension", distance 4, with explicit test and coerce.
Class "structure", by class "niftiExtension", distance 4.
Class "vector", by class "niftiExtension", distance 5, with explicit coerce.
Class "vector", by class "niftiExtension", distance 7, with explicit test and coerce.

Methods

No methods defined with class "niftiAuditTrail" in the signature.

Author(s)

Andrew Thornton <zeripath@users.sourceforge.net>

References

NIfTI-1
http://nifti.nimh.nih.gov/

See Also

nifti, niftiExtension

Examples

```
showClass("niftiAuditTrail")
```

niftiExtension-class *Class "niftiExtension"*

Description

An extension of the NIfTI class that allows "extensions" that conform to the NIfTI data standard.

Objects from the Class

Objects can be created by calls of the form `new("niftiExtension", data, dim, dimnames, ...)`.

Slots

`.Data`: Object of class `"array"`

`extensions`: Object of class `"list"` contains a list of all extensions.

`sizeof_hdr`: Object of class `"numeric"`

`data_type`: Object of class `"character"`

`db_name`: Object of class `"character"`

`extents`: Object of class `"numeric"`

`session_error`: Object of class `"numeric"`

`regular`: Object of class `"character"`

`dim_info`: Object of class `"numeric"`

`dim_`: Object of class `"vector"`

`intent_p1`: Object of class `"numeric"`

`intent_p2`: Object of class `"numeric"`

`intent_p3`: Object of class `"numeric"`

`intent_code`: Object of class `"numeric"`

`datatype`: Object of class `"numeric"`

`bitpix`: Object of class `"numeric"`

`slice_start`: Object of class `"numeric"`

`pixdim`: Object of class `"vector"`

`vox_offset`: Object of class `"numeric"`

`scl_slope`: Object of class `"numeric"`

`scl_inter`: Object of class `"numeric"`

`slice_end`: Object of class `"numeric"`

`slice_code`: Object of class `"numeric"`

`xyzt_units`: Object of class `"numeric"`

`cal_max`: Object of class `"numeric"`

`cal_min`: Object of class `"numeric"`

`slice_duration`: Object of class `"numeric"`

`toffset`: Object of class `"numeric"`

`glmax`: Object of class `"numeric"`

`glmin`: Object of class `"numeric"`

`descrip`: Object of class `"character"`

`aux_file`: Object of class `"character"`

`qform_code`: Object of class `"numeric"`

`sform_code`: Object of class `"numeric"`

`quatern_b`: Object of class `"numeric"`

`quatern_c`: Object of class `"numeric"`

`quatern_d`: Object of class `"numeric"`

`qoffset_x`: Object of class `"numeric"`

`qoffset_y`: Object of class `"numeric"`

`qoffset_z`: Object of class `"numeric"`

`srow_x`: Object of class `"vector"`

`srow_y`: Object of class `"vector"`

`srow_z`: Object of class `"vector"`

`intent_name`: Object of class `"character"`

`magic`: Object of class `"character"`

`extender`: Object of class `"vector"`

`reoriented`: Object of class `"logical"`

Extends

Class `"nifti"`, directly.
Class `"array"`, by class "nifti", distance 2.
Class `"matrix"`, by class "nifti", distance 3, with explicit test and coerce.
Class `"structure"`, by class "nifti", distance 3.
Class `"vector"`, by class "nifti", distance 4, with explicit coerce.
Class `"vector"`, by class "nifti", distance 6, with explicit test and coerce.

Methods

No methods defined with class `"niftiExtension"` in the signature.

Author(s)

Andrew Thornton <zeripath@users.sourceforge.net>

References

NIfTI-1
http://nifti.nimh.nih.gov/

See Also

nifti, niftiAuditTrail

Examples

```
showClass("niftiExtension")
```

---

niftiExtensionSection-class
Class "niftiExtensionSection"

Description

A niftiExtensionSection contains the fields that conform to the NIfTI standard regarding header extensions. A niftiExtension is composed of one or more of these objects.

Objects from the Class

Objects can be created by calls of the form new("niftiExtensionSection", data, dim, dimnames, ...).

Slots

esize: Object of class "numeric"; the number of bytes that form the extended header data.

ecode: Object of class "numeric"; a non-negative integer that indicates the format of the extended header data that follows (default = 1002).

edata: Object of class "character"; Note that the other contents of the extended header data section are totally unspecified by the NIfTI-1 standard.

Methods

No methods defined with class "niftiExtensionSection" in the signature.

Author(s)

Brandon Whitcher <bwhitcher@gmail.com>,
Andrew Thornton <zeripath@users.sourcefore.net>

References

NIfTI-1
http://nifti.nimh.nih.gov/

See Also

niftiExtension, nifti

Examples

```
showClass("niftiExtensionSection")
```

orientation-methods *Extract NIfTI 3D Image Orientation*

Description

Methods that act on the "qform" and "sform" information in the NIfTI header.

Usage

```
## S4 method for signature 'nifti'
qform(object)
## S4 method for signature 'nifti'
sform(object)
```

Arguments

object	is an object of class nifti.

Methods

object = "nifti" Extract or replace NIfTI description.

Author(s)

Brandon Whitcher <bwhitcher@gmail.com>

Examples

```
## Not run:
url <- "http://nifti.nimh.nih.gov/nifti-1/data/avg152T1_LR_nifti.nii.gz"
urlfile <- file.path(system.file("nifti", package="oro.nifti"),
                     "mniLR.nii.gz")
download.file(url, urlfile, quiet=TRUE)

## End(Not run)
urlfile <- file.path(system.file("nifti", package="oro.nifti"),
                     "mniLR.nii.gz")
mniLR <- readNIfTI(urlfile)
sform(mniLR)
```

orthographic-methods *Methods for Function 'orthographic' in Package 'dcemriS4'*

Description

Produce orthographic display for nifti, anlz and afni objects.

Usage

```
## S4 method for signature 'nifti'
orthographic(x, y = NULL, xyz = NULL, w = 1,
             col = gray(0:64/64), col.y = hotmetal(), zlim = NULL,
             zlim.y = NULL, crosshairs = TRUE, col.crosshairs = "red",
             xlab = "", ylab = "", axes = FALSE,
             oma = rep(0,4), mar = rep(0,4), bg = "black",
             text = NULL, text.color="white", text.cex = 2, ...)
```

Arguments

x	is an object of class nifti or similar.
y	is an object of class nifti or similar for the overlay.
xyz	is the coordinate for the center of the crosshairs.
w	is the time point to be displayed (4D arrays only).
col	is grayscale (by default).
col.y	is hotmetal (by default).
zlim	is the minimum and maximum 'z' values passed into image.
zlim.y	is the minimum and maximum 'z' values passed into image for the overlay.
crosshairs	is a logical value for the presence of crosshairs in all three orthogonal planes (default = TRUE).
col.crosshairs	is the color of the crosshairs (default = red).
xlab	is set to "" since all margins are set to zero.
ylab	is set to "" since all margins are set to zero.
axes	is set to FALSE since all margins are set to zero.
oma	is the size of the outer margins in the par function.
mar	is the number of lines of margin in the par function.
bg	is the background color in the par function.
text	allows the user to specify text to appear in the fourth (unused) pane.
text.color	is the color of the user-specified text (default = "white".
text.cex	is the size of the user-specified text (default = 2).
...	other arguments to the image function may be provided here.

Methods

x = "afni" Produce orthographic display for x.

x = "anlz" Produce orthographic display for x.

x = "array" Produce orthographic display for x.

x = "nifti" Produce orthographic display for x.

Author(s)

Brandon Whitcher <bwhitcher@gmail.com>

See Also

image-methods, overlay-methods

---

overlay-methods *Methods for Function overlay*

Description

Methods for function overlay

Usage

```
## S4 method for signature 'nifti,nifti'
overlay(x, y, z=1, w=1, col.x=gray(0:64/64),
        col.y=hotmetal(), zlim.x=NULL, zlim.y=NULL,
        plane=c("axial", "coronal", "sagittal"),
        plot.type=c("multiple","single"), xlab="", ylab="",
        axes=FALSE, oma=rep(0,4), mar=rep(0,4), bg="black", ...)
```

Arguments

x,y	is an object of class nifti or similar.
z	is the slice to be displayed (ignored when plot.type = "multiple").
w	is the time point to be displayed (4D arrays only).
col.x	is grayscale (by default).
col.y	is hotmetal (by default).
zlim.x,zlim.y	are set to NULL (by default) and taken from the header information.
plane	is the plane of acquisition to be displayed (choices are 'axial', 'coronal', 'sagittal').

plot.type	allows the choice between all slices being displayed, in a matrix (left-to-right, top-to-bottom), or a single slice.
xlab	is set to "" since all margins are set to zero.
ylab	is set to "" since all margins are set to zero.
axes	is set to FALSE since all margins are set to zero.
oma	is the size of the outer margins in the par function.
mar	is the number of lines of margin in the par function.
bg	is the background color in the par function.
...	other arguments to the image function may be provided here.

Details

The image command is used multiple times to simultaneously visualize one of the three orthogonal planes in two multidimensional arrays, one on top of the other, for medical imaging data.

Methods

x = "nifti", y = "nifti" Produce overlay of y on x.

x = "anlz", y = "anlz" Produce overlay of y on x.

x = "afni", y = "afni" Produce overlay of y on x.

Author(s)

Brandon Whitcher <bwhitcher@gmail.com>

See Also

image-methods, overlay-methods

---

performPermutation *performPermutation*

Description

Given an orthogonal permutation matrix T, an array of dimensions and a one-dimensional representation of data. It will return a transformed array with the transformed dimensions.

Usage

```
performPermutation(T, real.dimensions, data, verbose=FALSE)
```

Arguments

T	is an orthogonal matrix.
real.dimensions	is a one-dimensional array, representing the length of dimensions in data.
data	is a one-dimensional representation of the data to be transformed.
verbose	is a logical variable (default = FALSE) that allows text-based feedback during execution of the function.

Details

This function is mainly used by the reorient function to transform nifti data into neuroradiological convention.

Author(s)

Andrew Thornton <zeripath@users.sourceforge.net>

See Also

reorient,inverseReorient

---

pixdim-methods *Extract or Replace NIfTI/Analyze Pixel Dimensions*

Description

Methods that act on the "pixdim" numeric vector in the NIfTI or ANALYZE header.

Usage

```
## S4 method for signature 'nifti'
pixdim(object)
```

Arguments

object	is an object of class nifti or anlz.

Methods

object = "anlz" Extract or replace ANALYZE "pixdim" value.

object = "nifti" Extract or replace NIfTI "pixdim" value.

Author(s)

Brandon Whitcher <bwhitcher@gmail.com>

Examples

```
## Not run:
url <- "http://nifti.nimh.nih.gov/nifti-1/data/avg152T1_LR_nifti.nii.gz"
urlfile <- file.path(system.file("nifti", package="oro.nifti"),
                     "mniLR.nii.gz")
download.file(url, urlfile, quiet=TRUE)

## End(Not run)
urlfile <- file.path(system.file("nifti", package="oro.nifti"),
                     "mniLR.nii.gz")
mniLR <- readNIfTI(urlfile)
pixdim(mniLR)
```

---

quaternion2rotation *Convert Quaternion into a Rotation Matrix*

Description

The affine/rotation matrix R is calculated from the quaternion parameters.

Usage

```
quaternion2mat44(nim, tol = 1e-7)
quaternion2rotation(b, c, d)
```

Arguments

nim	is an object of class nifti.
tol	is a very small value used to judge if a number is essentially zero.
b	is the quaternion b parameter.
c	is the quaternion c parameter.
d	is the quaternion d parameter.

Details

The quaternion representation is chosen for its compactness in representing rotations. The orientation of the (x, y, z) axes relative to the (i, j, k) axes in 3D space is specified using a unit quaternion $[a, b, c, d]$, where $a^2 + b^2 + c^2 + d^2 = 1$. The (b, c, d) values are all that is needed, since we require that $a = [1 - (b^2 + c^2 + d^2)]^{1/2}$ be non-negative. The (b, c, d) values are stored in the (quatern_b, quatern_c, quatern_d) fields.

Value

The (proper) 3×3 rotation matrix or 4×4 affine matrix.

Author(s)

Brandon Whitcher <bwhitcher@gmail.com>

References

NIfTI-1
http://nifti.nimh.nih.gov/

Examples

```
## This R matrix is represented by quaternion [a,b,c,d] = [0,1,0,0]
## (which encodes a 180 degree rotation about the x-axis).
(R <- quaternion2rotation(1, 0, 0))
```

---

readAFNI *readAFNI*

Description

These functions read in the header information and multidimensional array from a binary file in AFNI format into a afni-class object.

Usage

```
readAFNI(fname, vol = NULL, verbose = FALSE, warn = -1, call = NULL)
```

Arguments

fname	is the file name of the AFNI file.
vol	vector of brick numbers to be read from file.
verbose	is a logical variable (default = FALSE) that allows text-based feedback during execution of the function.
warn	is a number to regularegulatete the display of warnings (default = -1). See options for more details.
call	keeps track of the current function call for use in the AFNI extension.

Details

The readAFNI function utilizes internal methods readBin and readLines to efficiently extract information from the header and binary file(s).

Current acceptable data types include

"INT16" DT SIGNED SHORT (16 bits per voxel)

"FLOAT32" DT FLOAT (32 bits per voxel)

"COMPLEX128" DT COMPLEX (128 bits per voxel)

Value

object of class afni

Author(s)

Karsten Tabelow <karsten.tabelow@wias-berlin.de>

References

AFNI
http://afni.nimh.nih.gov/pub/dist/src/README.attributes

See Also

readANALYZE, readNIfTI

Examples

```
## Taken from the AFNI Matlab Library
## http://afni.nimh.nih.gov/pub/dist/data/afni_matlab_data.tgz
afni.path <- system.file("afni", package="oro.nifti")
orig <- readAFNI(file.path(afni.path, "ARzs_CW_avvr.DEL+orig"))
image(orig, zlim=c(0.5,256), oma=rep(2,4))
orthographic(orig, zlim=c(0.5,256), oma=rep(2,4))
```

---

readANALYZE *readANALYZE*

Description

These functions read in the header information and multi-dimensional array from a binary file in Analyze 7.5 format.

Usage

```
readANALYZE(fname, SPM = FALSE, verbose = FALSE, warn = -1)
```

Arguments

fname	Pathname of the Analyze pair of files .img and .hdr without the suffix.
SPM	is a logical variable (default = FALSE) that forces the voxel data values to be rescaled using the funused1 ANALYZE header field. This is an undocumented convention of ANALYZE files processed using the Statistical Parametric Mapping (SPM) software.
verbose	is a logical variable (default = FALSE) that allows text-based feedback during execution of the function.
warn	is a number to regulate the display of warnings (default = -1). See options for more details.

Details

The internal functions readBin and rawToChar are utilized in order to efficiently extract information from a binary file. The types of data are limited to 1- and 2-byte integers, 4-byte floats and 8-byte doubles.

Value

An object of class anlz is produced.

Author(s)

Brandon Whitcher <bwhitcher@gmail.com>,
Volker Schmid <volkerschmid@users.sourceforge.net>

References

ANALYZE 7.5
http://www.mayo.edu/bir/PDF/ANALYZE75.pdf

See Also

readNIfTI

Examples

```
## avg152T1
anlz.path <- system.file("anlz", package="oro.nifti")
mni152 <- readANALYZE(file.path(anlz.path, "avg152T1"))
image(mni152, oma=rep(2,4))
orthographic(mni152, oma=rep(2,4))
```

---

readNIfTI *readNIfTI*

Description

These functions read in the header information and multidimensional array from a binary file in NIfTI-1 format into a nifti-class object.

Usage

```
readNIfTI(fname, verbose=FALSE, warn=-1, reorient=TRUE, call=NULL)
```

Arguments

fname	is the file name of the NIfTI file(s).
verbose	is a logical variable (default = FALSE) that allows text-based feedback during execution of the function.
warn	is a number to regulate the display of warnings (default = -1). See options for more details.
reorient	is a logical variable (default = TRUE) that enforces Qform/Sform transformations.
call	keeps track of the current function call for use in the NIfTI extension.

Details

The readNIfTI function utilizes internal methods readBin and readChar to efficiently extract information from the binary file(s).

Current acceptable data types include

"UINT8" BINARY (1 bit per voxel)

"INT16" SIGNED SHORT (16 bits per voxel)

"INT32" SINGED INT (32 bits per voxel)

"FLOAT32" FLOAT (32 bits per voxel)

"DOUBLE64" DOUBLE (64 bits per voxel)

"UINT16" UNSIGNED SHORT (16 bits per voxel)

"UINT32" UNSIGNED INT (32 bits per voxel)

Value

An object of class nifti.

Author(s)

Brandon Whitcher <bwhitcher@gmail.com>,
Volker Schmid <volkerschmid@users.sourceforge.net>,
Andrew Thornton <zeripath@users.sourceforge.net>

References

NIfTI-1
http://nifti.nimh.nih.gov/

See Also

readAFNI, readANALYZE

Examples

```
## Not run:
url <- "http://nifti.nimh.nih.gov/nifti-1/data/filtered_func_data.nii.gz"
urlfile <- file.path(system.file("nifti", package="oro.nifti"),
                     "filtered_func_data")
download.file(url, urlfile, quiet=TRUE)

## End(Not run)
## The NIfTI file provided here contains the first 18 volumes (10%)
## of the original data set
urlfile <- file.path(system.file("nifti", package="oro.nifti"),
                     "filtered_func_data")
(ffd <- readNIfTI(urlfile))
image(ffd, oma=rep(2,4))
orthographic(ffd, oma=rep(2,4))
## Not run:
## 27 scans of Colin Holmes (MNI) brain co-registered and averaged
## NIfTI two-file format
URL <- "http://imaging.mrc-cbu.cam.ac.uk/downloads/Colin/colin_1mm.tgz"
urlfile <- file.path(tempdir(), "colin_1mm.tgz")
download.file(URL, dest=urlfile, quiet=TRUE)
untar(urlfile, exdir=tempdir())
colin <- readNIfTI(file.path(tempdir(), "colin_1mm"))
image(colin, oma=rep(2,4))
orthographic(colin, oma=rep(2,4))

## End(Not run)
```

`reorient` *reorient*

Description

Transforms in the NIfTI header are parsed and normalized versions of these transforms are applied.

Usage

```
reorient(nim, data, verbose = FALSE, invert = FALSE, tol = 1e-07)
inverseReorient(nim, verbose = FALSE)
```

Arguments

`nim`	is an object of class `nifti`.
`data`	is an array associated with `nim`.
`verbose`	is a logical variable (default = `FALSE`) that allows text-based feedback during execution of the function.
`invert`	stores the inverse transform.
`tol`	is a very small value used to judge if a number is essentially zero.

Details

This function utilizes the `performPermutation` function internally.

Author(s)

Andrew Thornton <zeripath@users.sourceforge.net> and Brandon Whitcher <bwhitcher@gmail.com>

See Also

`performPermutation`

`rmniigz` *Remove File Extensions Around the NIfTI/ANALYZE Formats*

Description

Simple function(s) that remove file extensions commonly found when using NIfTI-1 or ANALYZE format files.

Usage

```
rmniigz(x)
rmnii(x)
rmgz(x)
rmhdrgz(x)
rmhdr(x)
rmimggz(x)
rmimg(x)
```

Arguments

`x`	is the file name.

Value

The file name without offending suffix.

Author(s)

Brandon Whitcher <bwhitcher@gmail.com>

`tim.colors` *Tim's Useful Color Table*

Description

A pleasing rainbow style color table patterned after that used in Matlab.

Usage

```
tim.colors(n=64)
```

Arguments

`n`	is the number of color levels (default = 64).

Details

Based on the `tim.colors` function in the **fields** package. The `tim.colors` function here has been modified to break any dependence on code in the **fields** package. Spline interpolation (`interpSpline`) is used when the number of requested colors is not the default.

Value

A vector of character strings giving the colors in hexadecimal format.

Author(s)

Tim Hoar (GSP-NCAR); modified by B. Whitcher

See Also

`hotmetal`, `topo.colors`, `terrain.colors`

Examples

```
tim.colors(10)
image(outer(1:20, 1:20, "+"), col=tim.colors(75), main="tim.colors")
```

translateCoordinate *translateCoordinate*

Description

Translates a voxel index into the continuous coordinate space defined by the NIfTI qform and sform information.

Usage

```
translateCoordinate(i, nim, verbose=FALSE)
```

Arguments

i	An index vector in nim.
nim	An object of class nifti.
verbose	Provide detailed output to the user.

Details

This function takes as input a nifti object and an index vector in the voxel space of the object and translates that voxel index into the continuous coordinate space defined by the object's qform and sform.

Please note:

1. By default the index i varies most rapidly, etc.
2. The ANALYZE 7.5 coordinate system is

   +x = Left
   +y = Anterior
   +z = Superior

   (A left-handed co-ordinate system)
3. The three methods below give the locations of the voxel centres in the x,y,z system. In many cases programs will want to display the data on other grids. In which case the program will be required to convert the desired (x,y,z) values in to voxel values using the inverse transformation.
4. Method 2 uses a factor qfac which is either -1 or 1. qfac is stored in pixdim[0]. If pixdim[0] != 1 or -1, which should not occur, we assume 1.
5. The units of the xyzt are set in xyzt_units field.

Value

A nifti-class object with translated coordinates.

Author(s)

Andrew Thornton <zeripath@users.sourceforge.net>

Examples

```
ffd <- readNIfTI(file.path(system.file("nifti", package="oro.nifti"),
                           "filtered_func_data"))
xyz <- c(1,1,1)
translateCoordinate(xyz, ffd, verbose=TRUE)
xyz <- trunc(dim(ffd)[1:3]/2)
translateCoordinate(xyz, ffd, verbose=TRUE)
```

---

writeAFNI-methods *writeAFNI*

Description

This function saves a afni-class object to HEAD/BRIK pair in AFNI format.

Usage

```
## S4 method for signature 'afni'
writeAFNI(nim, fname, verbose = FALSE, warn = -1)
```

Arguments

nim	is an object of class afni.
fname	is the path and file name to save the AFNI file (.HEAD/BRIK) **without** the suffix.
verbose	is a logical variable (default = FALSE) that allows text-based feedback during execution of the function.
warn	is a number to regulate the display of warnings (default = -1). See options for more details.

Details

The writeAFNI function utilizes the internal writeBin and writeLines command to write information to header/binary file pair.

Current acceptable data types include

"INT16" DT SIGNED SHORT (16 bits per voxel)

"FLOAT32" DT FLOAT (32 bits per voxel)

"COMPLEX128" DT COMPLEX (128 bits per voxel)

Value

Nothing.

Methods

nim = "afni" Write AFNI volume to disk.

nim = "ANY" not implemented.

Author(s)

Karsten Tabelow <karsten.tabelow@wias-berlin.de>

References

AFNI
http://afni.nimh.nih.gov/pub/dist/src/README.attributes

See Also

writeANALYZE, writeNIfTI

Examples

```
## Taken from the AFNI Matlab Library
## http://afni.nimh.nih.gov/pub/dist/data/afni_matlab_data.tgz
afni.path <- system.file("afni", package="oro.nifti")
orig <- readAFNI(file.path(afni.path, "ARzs_CW_avvr.DEL+orig"))
writeAFNI(orig, "test-afni-image", verbose=TRUE)

data <- readAFNI("test-afni-image", verbose=TRUE)
image(orig, zlim=c(0.5,256), oma=rep(2,4), bg="white")
image(data, zlim=c(0.5,256), oma=rep(2,4), bg="white")
abs.err <- abs(data - orig)
image(as(abs.err, "nifti"), zlim=range(0,1), oma=rep(2,4),
      bg="white")
```

---

writeANALYZE-methods *writeANALYZE*

Description

This function saves an Analyze-class object to a single binary file in Analyze format.

Usage

```
## S4 method for signature 'anlz'
writeANALYZE(aim, filename, gzipped = TRUE, verbose = FALSE,
             warn = -1)
```

Arguments

aim	is an object of class anlz.
filename	is the path and file name to save the Analyze file pair (.hdr,img) **without** the suffixes.
gzipped	is a character string that enables exportation of compressed (.gz) files (default = TRUE).
verbose	is a logical variable (default = FALSE) that allows text-based feedback during execution of the function.
warn	is a number to regulate the display of warnings (default = -1). See options for more details.

Details

The writeANALYZE function utilizes the internal writeBin and writeChar command to write information to a binary file.

Value

Nothing.

Methods

object = "anlz" Write ANALYZE volume to disk.

Author(s)

Brandon Whitcher <bwhitcher@gmail.com>

References

Analyze 7.5
http://www.mayo.edu/bir/PDF/ANALYZE75.pdf

See Also

writeAFNI, writeNIfTI

Examples

```
norm <- dnorm(seq(-5, 5, length=32), sd=2)
norm <- (norm-min(norm)) / max(norm-min(norm))
img <- outer(outer(norm, norm), norm)
img <- round(255*img)
img[17:32,,] <- 255 - img[17:32,,]
img.anlz <- anlz(img) # create Analyze object

writeANALYZE(img.anlz, "test-anlz-image-uint8", verbose=TRUE)
## These files should be viewable in, for example, FSLview
## Make sure you adjust the min/max values for proper visualization
```

```
data <- readANALYZE("test-anlz-image-uint8", verbose=TRUE)
image(img.anlz, oma=rep(2,4), bg="white")
image(data, oma=rep(2,4), bg="white")
abs.err <- abs(data - img.anlz)
image(as(abs.err, "anlz"), zlim=range(img.anlz), oma=rep(2,4), bg="white")

## Not run:
## Loop through all possible data types
datatypes <- list(code=c(2, 4, 8, 16, 64),
                  name=c("uint8", "int16", "int32", "float", "double"))
equal <- vector("list")
for (i in 1:length(datatypes$code)) {
  fname <- paste("test-anlz-image-", datatypes$name[i], sep="")
  rm(img.anlz)
  img.anlz <- anlz(img, datatype=datatypes$code[i])
  writeANALYZE(img.anlz, fname)
  equal[[i]] <- all(readANALYZE(fname) == img)
}
names(equal) <- datatypes$name
unlist(equal)

## End(Not run)
```

`writeNIfTI-methods` *writeNIfTI*

Description

This function saves a NIfTI-class object to a single binary file in NIfTI format.

Usage

```
## S4 method for signature 'nifti'
writeNIfTI(nim, filename, onefile = TRUE, gzipped = TRUE,
            verbose = FALSE, warn = -1)
## S4 method for signature 'anlz'
writeNIfTI(nim, filename, onefile = TRUE, gzipped = TRUE,
            verbose = FALSE, warn = -1)
## S4 method for signature 'array'
writeNIfTI(nim, filename, onefile = TRUE, gzipped = TRUE,
            verbose = FALSE, warn = -1)
```

Arguments

`nim`	is an object of class `nifti` or `anlz`.
`filename`	is the path and file name to save the NIfTI file (.nii) **without** the suffix.
`onefile`	is a logical value that allows the scanning of single-file (.nii) or dual-file format (.hdr and .img) NIfTI files (default = `TRUE`).

gzipped	is a character string that enables exportation of compressed (.gz) files (default = TRUE).
verbose	is a logical variable (default = FALSE) that allows text-based feedback during execution of the function.
warn	is a number to regulate the display of warnings (default = -1). See options for more details.

Details

The writeNIfTI function utilizes the internal writeBin and writeChar command to write information to a binary file.

Current acceptable data types include

"UINT8" DT BINARY (1 bit per voxel)

"INT16" DT SIGNED SHORT (16 bits per voxel)

"INT32" DT SINGED INT (32 bits per voxel)

"FLOAT32" DT FLOAT (32 bits per voxel)

"DOUBLE64" DT DOUBLE (64 bits per voxel)

"UINT16" DT UNSIGNED SHORT (16 bits per voxel)

Value

Nothing.

Methods

object = "anlz" Convert ANALYZE object to class nifti and write the NIfTI volume to disk.

object = "array" Convert array to class nifti and write the NIfTI volume to disk.

object = "nifti" Write NIfTI volume to disk.

Author(s)

Brandon Whitcher <bwhitcher@gmail.com>,
Volker Schmid <volkerschmid@users.sourceforge.net>

References

NIfTI-1
http://nifti.nimh.nih.gov/

See Also

writeAFNI, writeANALYZE

Examples

```
norm <- dnorm(seq(-5, 5, length=32), sd=2)
norm <- (norm-min(norm)) / max(norm-min(norm))
img <- outer(outer(norm, norm), norm)
img <- round(255 * img)
img[17:32,,] <- 255 - img[17:32,,]
img.nifti <- nifti(img) # create NIfTI object

writeNIfTI(img.nifti, "test-nifti-image-uint8", verbose=TRUE)
## These files should be viewable in, for example, FSLview
## Make sure you adjust the min/max values for proper visualization
data <- readNIfTI("test-nifti-image-uint8", verbose=TRUE)
image(img.nifti, oma=rep(2,4), bg="white")
image(data, oma=rep(2,4), bg="white")
abs.err <- abs(data - img.nifti)
image(as(abs.err, "nifti"), zlim=range(img.nifti), oma=rep(2,4),
      bg="white")

## Not run:
## Loop through all possible data types
datatypes <- list(code=c(2, 4, 8, 16, 64),
                  name=c("uint8", "int16", "int32", "float", "double"))
equal <- vector("list")
for (i in 1:length(datatypes$code)) {
  fname <- paste("test-nifti-image-", datatypes$name[i], sep="")
  rm(img.nifti)
  img.nifti <- nifti(img, datatype=datatypes$code[i])
  writeNIfTI(img.nifti, fname, verbose=TRUE)
  equal[[i]] <- all(readNIfTI(fname) == img)
}
names(equal) <- datatypes$name
unlist(equal)

## End(Not run)
```

[<-methods *~~ Methods for Function [<- in Package 'base' ~~*

Description

~~ Methods for function [<- in Package 'base' ~~

Methods

x = "nifti", i = "ANY", j = "ANY", value = "ANY" Replaces the data at the provided co-ordinates with the value provided and updates the header

x = "nifti", i = "numeric", j = "numeric", value = "ANY" Replaces the data at the provided co-ordinates with the value provided and updates the header

x = "nifti", i = "ANY", j = "missing", value = "ANY" Replaces the data row i of the provided nifti object with the value provided and updates the header

x = "nifti", i = "numeric", j = "missing", value = "ANY" Replaces the data row i of the provided nifti object with the value provided and updates the header

x = "nifti", i = "missing", j = "missing", value = "array" Replaces the data of the provided nifti object with the array provided and updates the header

Index