Package ‘tractor.base’

June 10, 2019

Version 3.3.2
Date 2019-06-06
Title Read, Manipulate and Visualise Magnetic Resonance Images
Author Jon Clayden
Maintainer Jon Clayden <code@clayden.org>
Imports methods, ore (>= 1.3.0), reportr, shades, RNifti
Suggests mmmand, loder, divest, jsonlite, yaml, testthat
Enhances oro.nifti
Description Functions for working with magnetic resonance images. Reading and writing of popular file formats (DICOM, Analyze, NIfTI-1, NIfTI-2, MGH); interactive and non-interactive visualisation; flexible image manipulation; metadata and sparse image handling.
Encoding UTF-8
LazyLoad yes
LazyData yes
License GPL-2
BugReports https://github.com/tractor/tractor/issues
RoxygenNote 6.1.0
NeedsCompilation no
Repository CRAN
Date/Publication 2019-06-10 12:20:03 UTC

R topics documented:

  allEqual ................................................. 2
  asMriImage ........................................... 3
  createNeighbourhoodInfo .............................. 5
  createSliceGraphic .................................. 6
**Description**

This function tests whether all elements of the specified vector are equal to each other, i.e., whether the vector contains only a single unique value. For lists, equality is determined using `equivalent`.

**Usage**

```r
allEqual(x, ignoreMissing = FALSE, ...)
```
Arguments

- `x` A vector of any mode, including a list.
- `ignoreMissing` If TRUE, missing elements will be ignored. Otherwise the presence of missing values will result in a return value of FALSE.
- `...` Additional arguments to `all.equal`, via `equivalent`.

Value

TRUE if all elements test equivalent; FALSE otherwise.

Author(s)

Jon Clayden

References

Please cite the following reference when using TractoR in your work:


See Also

equivalent for elementwise equivalence of two vectors.

Examples

```r
allEqual(c(1,1,1))  # TRUE
allEqual(c(1,1,NA)) # FALSE
allEqual(c(1,1,NA), ignoreMissing=TRUE)  # TRUE
```

Description

Functions for creating MriImage objects from data, including other images. All of these functions use data from arrays or MriImage objects to create a new MriImage object. `asMriImage` is the basic function for creating an object from its constituents: an array of voxel values and some metadata (and/or a template image).
**Usage**

```r
asMriImage(data, templateImage = nilObject(), imageDims = NA,
voxelDims = NA, voxelDimUnits = NA, origin = NA, tags = NA,
reordered = NA)
```

```r
extractMriImage(image, dim, loc)
```

```r
trimMriImage(image, clearance = 4, indices = NULL)
```

```r
reorderMriImage(image)
```

**Arguments**

- **data**
  An array of pixel/voxel data.

- **templateImage**
  An optional `MriImage` object, to be used as a metadata template.

- **imageDims, voxelDims, voxelDimUnits, origin, tags, reordered**
  Metadata for the new image object. These values override any from the metadata object or data array. See `MriImage` class documentation for details.

- **image**
  An `MriImage` object.

- **dim, loc**
  The dimension and location along that dimension for which data should be extracted.

- **clearance**
  The number of voxels’ clearance left around a trimmed image.

- **indices**
  A list of indices to keep along each dimension. Determined from the specified clearance if NULL.

**Details**

- `extractMriImage` reduces the dimensionality of the source image by one, by extracting a single “slice” of data along one dimension. `trimMriImage` trims empty space from the edges of an image, reducing the dimensions of the image and thus avoiding the storage of lots of zeroes. `reorderMriImage` reorders the image data (and corresponding metadata) to the LAS convention, an operation which is usually performed when an image is read from file.

**Value**

An `MriImage` object.

**Author(s)**

Jon Clayden

**References**

Please cite the following reference when using TractoR in your work:

createNeighbourhoodInfo

Description
This function calculates information about a cuboidal region of an image, with a centre and a fixed voxel width.

Usage
createNeighbourhoodInfo(width, dim = 3, centre = rep(0, dim))

Arguments
- width: An integer voxel width. Must be odd.
- dim: An integer giving the dimensionality of the neighbourhood. Currently must be 3.
- centre: A numeric vector giving the centre voxel of the neighbourhood. Must have exactly dim elements.

Value
createNeighbourhoodInfo returns a list with class "neighbourhoodInfo" and elements
- width: Copied from the width argument.
- dim: Copied from the dim argument.
- centre: Copied from the centre argument.
- vectors: dim x width^dim matrix whose columns give the locations of each point in the neighbourhood.
- innerProducts: A square, symmetric matrix of inner products between every location in the neighbourhood and every other.

Author(s)
Jon Clayden

References
Please cite the following reference when using TractoR in your work:
**createSliceGraphic**  
*Visualise MriImage objects*

**Description**

Visualise MriImage objects noninteractively using an R graphics device. See `viewImages` for an interactive alternative. These functions create 2D visualisations of 3D images by slicing or maximum intensity projection.

**Usage**

```r
createSliceGraphic(image, x = NA, y = NA, z = NA,  
  device = c("internal", "png"), colourScale = 1, add = FALSE,  
  file = NULL, zoomFactor = 1, windowLimits = NULL)

createProjectionGraphic(image, axis = 1, device = c("internal", "png"),  
  colourScale = 1, add = FALSE, file = NULL, zoomFactor = 1,  
  windowLimits = NULL)

createContactSheetGraphic(image, axis = 1, device = c("internal", "png"),  
  colourScale = 1, add = FALSE, file = NULL, zoomFactor = 1,  
  windowLimits = NULL, clearance = NULL, nColumns = NULL)
```

**Arguments**

- **image**: An MriImage object.
- **x, y, z**: Integer vectors, each of length 1. Exactly one of these must be specified to indicate the plane of interest.
- **device**: Either "internal" for display on the default graphics device, or "png" for creating PNG format image file(s). Abbreviations are fine.
- **colourscale**: A colour scale definition, of the sort generated by `getColourscale`.
- **add**: Overlay the graphic on a previous one. Used only when device is "internal".
- **file**: A file name, to be used when device is "png".
- **zoomFactor**: Factor by which to enlarge the image. Applies only when device is "png".
- **windowLimits**: Numeric vector of length 2 giving the limits of the colour scale, or NULL for limits matching the range of the image data. Passed as the `zlim` argument to `image`.
- **axis**: A vector of axes along which slice/projection images should be created. 1 is left-right, 2 is anterior-posterior, 3 is superior-inferior.
- **clearance**: Number of voxels' clearance to leave around each slice image in the contact sheet. Passed to `trimMriImage`.
- **nColumns**: Number of slices per row in the contact sheet grid. If NULL, the function will aim for a square grid.
Value

These functions are called for their side effects.

Note

When the device option is set to "png", the "png" and "mmand" packages are required by these functions.

Author(s)

Jon Clayden

References

Please cite the following reference when using TractoR in your work:

See Also

See `viewImages` for an interactive alternative, and `getColourScale` for details of how colour scales are specified. Also `image`, which is used as the underlying plot function.

---

**deduplicate**

*Concatenate and deduplicate vectors*

**Description**

This function returns its arguments, after concatenating them using `c` and then removing elements with duplicate names. The first element with each name will remain. Unnamed elements are retained.

**Usage**

deduplicate(...)

**Arguments**

... One or more vectors of any mode, usually named.

**Value**

The concatenated and deduplicated vector.

**Author(s)**

Jon Clayden
References

Please cite the following reference when using TractoR in your work:

defaultInfoPanel \( \text{A simple interactive viewer for MriImage objects} \)

Description

The \text{viewImages} function provides a simple interactive viewer for \text{MriImage} objects. 3D and 4D images may be used.

Usage

defaultInfoPanel(point, data, imagenames)

timeSeriesPanel(point, data, imagenames)

polarPlotPanel(point, data, imagenames, directions, bValues = NULL)

viewImages(images, colourScales = NULL, point = NULL, interactive = TRUE, crosshairs = TRUE, orientationLabels = TRUE, fixedWindow = TRUE, indexNames = NULL, infoPanel = defaultInfoPanel, ...)

Arguments

point \hspace{1cm} \text{For \text{viewImages}, a length 3 integer vector giving the initial location of the crosshairs, in voxels. For info panel functions, the current location of the crosshairs.}

data \hspace{1cm} \text{A list giving the data value(s) at the current crosshair location in each image displayed. Typically numeric, but in principle may be of any mode, and will be character mode when \text{indexNames} is not NULL.}

imageName \hspace{1cm} \text{A character vector giving a name for each image displayed.}

directions \hspace{1cm} \text{A matrix of 3D acquisition direction vectors, one per row.}

bValues \hspace{1cm} \text{A vector of b-values, if the image is diffusion-weighted.}

images \hspace{1cm} \text{An \text{MriImage} object, or list of \text{MriImage} objects.}

colourScales \hspace{1cm} \text{A list of colour scales to use for each image, which will be recycled to the length of images. See \text{getColourScale} for details. The default is to use greyscale.}

interactive \hspace{1cm} \text{A single logical value. If TRUE, the plot is interactive.}

crosshairs \hspace{1cm} \text{A single logical value. If TRUE, the crosshairs are displayed.}

orientationLabels \hspace{1cm} \text{A single logical value. If TRUE, orientation labels are displayed.}
**DicomMetadata-class**

**Description**

This class represents DICOM metadata, which typically contains detailed information about the scan parameters and subject.

---

**fixedWindow**

A single logical value. If TRUE, each image is windowed globally, rather than for each slice.

**indexNames**

A list whose elements are either NULL or a named character vector giving the names associated with each index in the image.

**infoPanel**

A function with at least three arguments, which must plot something to fill the bottom-right panel of the viewer after each change of crosshair location. The three mandatory arguments correspond to the current location in the image, the image values at that location, and the names of each image. The defaultInfoPanel and timeSeriesPanel functions are valid examples.

... Additional arguments to infoPanel.

**Value**

These functions are called for their side effects.

**Note**

The defaultInfoPanel and timeSeriesPanel functions are not intended to be called directly. They are simple examples of valid values for the infoPanel argument to viewImages.

**Author(s)**

Jon Clayden

**References**

Please cite the following reference when using TractoR in your work:


**See Also**

getColourScale
Fields

- **source**: String naming the source file
- **tags**: Data frame of tag information
- **tagOffset**: Starting offset for tags in the file
- **dataOffset**: Starting offset for pixel data in the file
- **dataLength**: Pixel data length
- **explicitTypes**: Logical value indicating whether explicit types are used in the file
- **endian**: String naming the endianness of the file
- **asciiFields**: Character vector containing the contents of the ASCII header, if requested and present in the file.
- **transferSyntax**: Transfer syntax string, if specified in the file; otherwise the empty string.

Methods

- **getAsciiFields(regex = NULL)**: Retrieve the value of one or more fields in the ASCII header. Returns NA if no fields match
- **getTagValue(group, element)**: Retrieve the value of a given tag, using an appropriate R type. Returns NA if the tag is missing

---

**embrace**

*Combine similar strings into one*

**Description**

Merge a vector of strings with a common prefix and/or suffix into one string with the unique parts in braces, comma-separated.

**Usage**

`embrace(strings)`

**Arguments**

- **strings**: A vector, which will be coerced to mode character.

**Value**

A single merged string, with the common prefix and suffix as attributes.

**Author(s)**

Jon Clayden
References

Please cite the following reference when using TractoR in your work:

Examples

```r
embrace(c("image.hdr", "image.img"))
```

---

**emptyMatrix**

The empty matrix

**Description**

The empty matrix is a standard matrix of dimensions 0 x 0. It is intended to be used as a placeholder where a matrix is required but no information is stored.

**Usage**

```r
emptyMatrix()
```

```r
is.emptyMatrix(object)
```

**Arguments**

- `object` Any object.

**Value**

`emptyMatrix` returns the empty matrix, equivalent to `matrix(NA, 0, 0)`. `is.emptyMatrix` returns `TRUE` if its argument is identical to the empty matrix.

**Author(s)**

Jon Clayden

**References**

Please cite the following reference when using TractoR in your work:
equivalent

Test two numeric vectors for equivalence

Description

This function is a wrapper for `isTRUE(all.equal(x, y, ...))`, but with the additional capability of doing sign-insensitive comparison.

Usage

```r
equivalent(x, y, signMatters = TRUE, ...)
```

Arguments

- `x` The first numeric vector.
- `y` The second numeric vector.
- `signMatters` Logical value: if FALSE then equivalence in absolute value is sufficient.
- `...` Additional arguments to `all.equal`, notably `tolerance`.

Value

TRUE if all elements of `x` match all elements of `y` to within tolerance, ignoring signs if required. FALSE otherwise.

Author(s)

Jon Clayden

References

Please cite the following reference when using TractoR in your work:


See Also

- `all.equal`

Examples

```r
equivalent(c(-1,1), c(1,1))  # FALSE
equivalent(c(-1,1), c(1,1), signMatters=FALSE)  # TRUE
equivalent(1:2, 2:3, tolerance=2)  # TRUE
```
Shorthand anonymous functions

Description
These functions provide a shorthand route to simple anonymous functions.

Usage
- \texttt{fx}(\texttt{expr})
- \texttt{fxy}(\texttt{expr})
- \texttt{fxyz}(\texttt{expr})
- \texttt{fi}(\texttt{expr})

Arguments
- \texttt{expr} A (single or compound) expression forming the body of the function.

Value
The function constructed.

Author(s)
Jon Clayden

References
Please cite the following reference when using TractoR in your work:

Obtaining colour scales

Description
The \texttt{getColourScale} function can be used to obtain a standard or customised colour scale for use in the package’s image visualisation functions.
getColourScale

Usage
getColourScale(n)

Arguments
n A number, colour name or list (see Details).

Details
Colour scales can be specified in any of three ways. Firstly, by a single number, representing a predefined colour scale. Currently valid values are 1 (greyscale, black background), 2 (red to yellow heat scale, red background), 3 (blue to red rainbow scale, blue background), 4 (blue to white to red diverging scale, white background), 5 (white to red, white background), 6 (white to blue, white background), 7 (yellow to orange to red) and 8 (purple to green to yellow, perceptually uniform). Secondly, a single colour name can be given (see colours); in this case the background will be black. This is useful for binary images. Thirdly, and most flexibly, a list with two named elements can be given: colours, a vector of colours representing the colour scale, perhaps created using the shades package; and background, a single colour representing the background.

Value
A list with elements

  colours A character-mode vector representing the colours in the scale, usually of length 100. This can be passed as a colour scale to R’s plotting functions.

  background A single character string representing the background colour.

Author(s)
Jon Clayden

References
Please cite the following reference when using TractoR in your work:

See Also
colours, rgb, colorRamp, and the shades package for colour manipulation.

Examples
getColourScale(1)
identifyImageFileNames

Working with MRI images stored in NIfTI, Analyze and MGH formats

Description

Functions for reading, writing, locating, copying and removing MRI images stored in NIfTI, Analyze and MGH formats.

Usage

identifyImageFileNames(fileName, fileType = NULL,
    errorIfMissing = TRUE, auxiliaries = c("dirs", "lut", "tags"), ...)

imageFileExists(fileName, fileType = NULL)

removeImageFiles(fileName, ...)

symlinkImageFiles(from, to, overwrite = FALSE, relative = TRUE, ...)

copyImageFiles(from, to, overwrite = FALSE, deleteOriginals = FALSE, ...)

readImageFile(fileName, fileType = NULL, metadataOnly = FALSE,
    volumes = NULL, sparse = FALSE, mask = NULL, reorder = TRUE, ...)

writeImageFile(image, fileName = NULL, fileType = NA,
    overwrite = TRUE, maxSize = NULL, writeTags = FALSE)

Arguments

fileName, from, to
    File names, with or without appropriate extension.

fileType
    A character vector of length one, giving the file type required or expected. If this option is missing, the file type used for writing images will be taken from the tractorFileType option. See Details.

errorIfMissing
    Logical value: raise an error if no suitable files were found?

auxiliaries
    A character vector of auxiliary file suffixes to search for.

...  
    For identifyImageFileNames, additional arguments to resolvePath. Elsewhere, additional arguments to identifyImageFileNames.

overwrite
    Logical value: overwrite an existing image file? For writeImageFile, an error will be raised if there is an existing file and this is set to FALSE.

relative
    Logical value: if TRUE, the path stored in the symlink will be relative (e.g. "./some_dir/some_image.nii") rather than absolute (e.g. "/path/to/some_dir/some_image.nii")
**Details**

NIfTI and Analyze are related formats for storing magnetic resonance images. NIfTI is a more recent extension of Analyze, and contains more specific information about, for example, the orientation of the image. Its use is therefore recommended where possible. MGH format is used by the popular image processing package FreeSurfer. These formats use a number of different file extensions, but the details are abstracted away from the user by these functions.

TractoR does not allow for files with the same basic name using multiple Analyze/NIfTI/ MGH formats in a single directory (e.g. "foo.nii" AND "foo.img"), and these functions will produce an error if multiple compatible files exist.

Suitable values for `fileType` (and the `tractorFileType` option, which is used as a default) are `ANALYZE`, `NIFTI`, `NIFTI_PAIR` (the two-file NIfTI format), `MGH`, `ANALYZE_GZ`, `NIFTI_GZ`, `NIFTI_PAIR_GZ` and `MGH_GZ`. The latter four are gzipped versions of the former four. `NIFTI_GZ` is recommended unless there is a need for one of the others. This is the default value for the `tractorFileType` option, but that can be changed using a call to `options`, or by setting the `TRACTOR_FILETYPE` environment variable before loading the `tractor.base` package.

Since multiple files may be involved, copying, moving or symlinking images is not trivial. `copyImageFiles` and ` symlinkImageFiles` are wrappers around the standard functions `file.copy` and `file.symlink` which handle this complexity.

**Value**

`readImageFile` returns an `MriImage` object. `imageFileExists` returns `TRUE` if an existing file with the specified name exists (all file extensions are checked), and `FALSE` otherwise. `removeImageFiles` returns the result of `unlink` applied to all relevant files. `writeImageFile` and `identifyImageFileNames` return a list with the following elements, describing the identified or written files:

- **fileStem**: The file name without extension.
headerFile  The full header file name.
imageFile  The full image file name.
format  The format of the files ("Nifti", "Analyze" or "Mgh"). Not returned by writeImageFile. copyImageFiles and symlinkImageFiles are called for their side effects.

Author(s)
Jon Clayden

References
Please cite the following reference when using TractoR in your work:

See Also
The NIfTI-1 standard (http://nifti.nimh.nih.gov/nifti-1) and MriImage.

implode  

Create a character string by concatenating the elements of a vector

Description
Create a character string by concatenating the elements of a vector, using a separator and optional final separator.

Usage
implode(strings, sep = "", finalSep = NULL, ranges = FALSE)

Arguments
  strings  A vector, which will be coerced to mode character.
  sep  A unit length character vector giving the separator to insert between elements.
  finalSep  An optional unit length character vector giving the separator to insert between the final two elements.
  ranges  Logical value. If TRUE and strings can be interpreted as integers, collapse runs of consecutive numbers into range notation.

Value
A character vector of length one.
indexList

Extract one or more elements from a list

Description

Given a list-like first argument, this function extracts one or more of its elements. Numeric and character indexing are allowed.

Usage

indexList(list, index = NULL)

Arguments

- list: A list-like object, with a [[ indexing method.
- index: A vector of integers or strings, or NULL.

Value

If index is NULL, the whole list is returned. Otherwise, if index has length one, the corresponding element is extracted and returned. Otherwise a list containing the requested subset is returned.

Note

This function is not type-safe, in the sense that its return type depends on its arguments. It should therefore be used with care.

Examples

implode(1:3, "", ) # "1, 2, 3"
implode(1:3, "", and "") # "1, 2 and 3"
implode(1:2, "", and "") # "1 and 2"
implode(1:3, "", ranges=TRUE) # "1–3"

See Also

paste
infix

Author(s)

Jon Clayden

References

Please cite the following reference when using TractoR in your work:

| infix | Resolve a variable to a default when NULL |

Description

This is a very simple infix function for the common TractoR idiom whereby NULL is used as a default argument value, but later needs to be resolved to a meaningful value if not overridden in the call. It returns its first argument unless it is NULL, in which case it falls back on the second argument.

Usage

X || Y

Arguments

X, Y R objects, possibly NULL.

Value

X, if it is not NULL; otherwise Y.

Author(s)

Jon Clayden

References

Please cite the following reference when using TractoR in your work:
isDeserialisable
Reference object serialisation and deserialisation

Description

Rather than using R’s `save` and `load` functions directly for reference objects, TractoR uses the `SerialisableObject` class and these functions to save and load objects. The main difference is that this approach stores only the data in the object, and not the functions which operate on them. This helps backward compatibility when new member functions are added.

Usage

```r
isDeserialisable(object, expectedClass = NULL)
serialiseReferenceObject(object, file = NULL)
deserialiseReferenceObject(file = NULL, object = NULL, raw = FALSE)
registerDeserialiser(className, deserialiser)
```

Arguments

- `object` For `serialiseReferenceObject`, a list or object inheriting from `SerialisableObject`. For other functions, an object in (raw) serialised form. See Details.
- `expectedClass` A class name which the object is expected to inherit. Any class is acceptable if this parameter is `NULL`.
- `file` A file name to deserialise from.
- `raw` If `TRUE`, the raw serialised object is returned; otherwise the object is converted back to its original class.
- `className` A string naming a class to be handled by the specified deserialiser.
- `deserialiser` A function taking as its argument a list of serialised fields, and returning a suitable deserialised object.

Details

The `serialiseReferenceObject` function, or the `serialise` member function of the `SerialisableObject` class can be used to create and/or `save` a version of an object which contains a hierarchical representation of the data embedded in it. These serialised objects are standard R lists, with an “originalClass” attribute describing the class of the original object. The `deserialiseReferenceObject` function can be used to deserialise them. Custom deserialisers can be specified using `registerDeserialiser`, typically for legacy classes.

Note that this should generally NOT be used as the primary mechanism for saving and loading `MriImage` objects. Saving to standard NIfTI/Analyze format is usually preferable, and can be done using `writeImageFile`. 
Value

isDeserialisable returns TRUE if the object is deserialisable and inherits from the specified class. deserialiseReferenceObject returns a raw or reconstituted object after deserialisation.

Author(s)

Jon Clayden

References

Please cite the following reference when using TractoR in your work:

See Also

SerialisableObject, save, load, writeImageFile.

data.locateExecutable

Find or run an external executable file

Description

The execute function is a wrapper around the system function in base, which additionally echoes the command being run (including the full path to the executable) if the reportr output level is Debug. locateExecutable simply returns the path to an executable file on the system PATH.

Usage

locateExecutable(fileName, errorIfMissing = TRUE)

e(xecute(executable, params = NULL, errorOnFail = TRUE,
   silent = FALSE, ...)

Arguments

executable, fileName
   Name of the executable to run.
params
   A character vector giving the parameters to pass to the executable, if any. Elements will be separated by a space.
errorOnFail, errorIfMissing
   Logical value: should an error be produced if the executable can’t be found?
silent
   Logical value: should the executable be run without any output?
... Additional arguments to system.
Value

For execute, the return value of the underlying call to system2. For locateExecutable, the location of the requested executable, or NULL if it could not be found.

Note

These functions are designed for Unix systems and may not work on Windows.

Author(s)

Jon Clayden

References

Please cite the following reference when using TractoR in your work:

See Also

system2

mergeMriImages Merging MriImage objects

Description

This function concatenates the data from a series of MriImage objects, and then attempts to work out the final dimensions of the merged image and returns it.

Usage

mergeMriImages(..., bindDim = NULL, padTags = FALSE)

Arguments

... MriImage objects. They do not need to have the same dimensionality.
bindDim An integer specifying the dimension along which to bind the data, or NULL (the default). The latter case resolves to one number higher than the last dimension common to all images.
padTags Logical value. If TRUE, NAs will be used to pad tags which appear to be partially missing in the merged dataset. If FALSE, incomplete tags will be dropped.

Value

A merged image.
Note

Tags are retained as-is if they are identical in each image. Otherwise they are concatenated if their lengths match the number of blocks in each image, or concatenated with NAs for missing values if padTags is TRUE.

Author(s)

Jon Clayden

References

Please cite the following reference when using TractoR in your work:

See Also

MriImage

The MriImage class

Description

This class represents an MRI image. An object of this class is made up of some voxel data, stored as a sparse or dense numeric array, and some metadata, such as the file it was read from, the voxel dimensions, and so on. The group generic functions Math, Ops and Summary are defined for this class, as are methods for coercing to and from a standard array.

Fields

imageDims  Integer vector of dimensions
voxelDims  Numeric vector of pixel/voxel spacings
voxelDimUnits  Character vector of spatial and/or temporal spacing units. Millimetres and seconds (i.e., c("mm","s")) are typical
source  String naming the file(s) that the image was read from. This is reset to the empty string if the image is modified
origin  Numeric vector giving the spatial coordinate origin
xform  Numeric matrix giving the NIfTI-style xform matrix associated with the image, which indicates its orientation
reordered  Logical value indicating whether the image has been reordered. See reorderMriImage
tags  Named list of arbitrary DICOM-style tags
data  Sparse or dense array of data, or NULL
Methods

apply(...) Apply a function to the margins of the image
binarise() Binarise the image by setting nonzero values to one
fill(value) Fill the image with a particular value
find(fun = NULL, ..., array = TRUE) Find voxels whose values are not zero, or satisfy a function
dataAtPoint(...) Obtain the value of the image at a particular point
dataAtMetadata() Obtain a version of the image with any data removed
getNonzeroIndices(array = TRUE, positiveOnly = FALSE) Find voxels whose values are not zero
dataSlice(dim, loc) Extract data from a slice of the image along one dimension
dataSparseness() Obtain the proportion of zeroes in the image
dataTags(keys = NULL) Retrieve some or all of the tags stored with the image
dataXform(implicit = TRUE) Retrieve the stored or implicit xform matrix
dataMap(fun, ..., sparse = NULL) Replace the current data with the result of a function
dataMask(maskImage) Mask the image, setting zero voxels in the mask to zero
dataSet(data) Replace the data in the image
dataSetOrigin(newOrigin) Update the origin of the image
dataSetSource(newSource) Update the source of the image
dataSetTexts(...) Add or replace metadata tags
dataSetXform(newXform) Update the xform matrix associated with the image
threshold(level, defaultValue = 0) Threshold the image by setting values below the threshold level to zero

newMriImageFromDicomDirectory

Deprecated functions

Description

These functions are deprecated, generally in favour of more succinct alternatives.

Usage

newMriImageFromDicomDirectory(dicomDir, readDiffusionParams = FALSE, untileMosaics = TRUE)

newMriImageFromFile(fileName, fileType = NULL, metadataOnly = FALSE, volumes = NULL, sparse = FALSE, mask = NULL, reorder = TRUE)

writeMriImageToFile(image, fileName = NULL, fileType = NA,
newMriImageFromDicomDirectory

overwrite = TRUE)
newMriImageByExtraction(image, dim, loc)
extractDataFromMriImage(image, dim, loc)
newMriImageByReordering(image)
newMriImageByTrimming(image, clearance = 4)
newMriImageWithData(data, templateImage = nilObject(), imageDims = NA,
voxelDims = NA, voxelDimUnits = NA, origin = NA, tags = NA)
newMriImageWithSimpleFunction(image, fun, ...)
newMriImageWithBinaryFunction(image1, image2, fun, ...)
newMriImageByMasking(image, mask)
newMriImageByThresholding(image, level, defaultValue = 0)
newDicomMetadataFromFile(fileName, checkFormat = TRUE,
dictionary = NULL, stopTag = NULL, ignoreTransferSyntax = FALSE)
removeImageFilesWithName(fileName)

Arguments

dicomDir Character vector of length one giving the name of a directory containing DICOM files.
readDiffusionParams Logical value. Should diffusion MRI parameters (b-values and gradient directions) be retrieved from the files if possible?
untileMosaics Logical value. Should Siemens mosaic images be converted into 3D volumes? This may occasionally be performed in error, which can be prevented by setting this value to FALSE.
fileName File names, with or without appropriate extension.
fileType A character vector of length one, giving the file type required or expected. If this option is missing, the file type used for writing images will be taken from the tractorFileType option. See Details.
metadataOnly Logical value: if TRUE, only metadata are read into the object.
volumes An optional integer vector specifying a subset of volumes to read (generally to save memory). If given, only the requested volumes in the 4D file will be read.
sparse Logical value: should the image data be stored in a SparseArray object?
mask An array whose nonzero voxel locations will be masked in.
reorder Logical value: should the image data be reordered to LAS? This is recommended in most circumstances.
image, image1, image2
MriImage objects.

overwrite Logical value: overwrite an existing image file? For writeImageFile, an error will be raised if there is an existing file and this is set to FALSE.

dim The dimension and location along that dimension for which data should be extracted.

loc The dimension and location along that dimension for which data should be extracted.

clearance The number of voxels’ clearance left around a trimmed image.

data An array of pixel/voxel data.

templateImage An optional MriImage object, to be used as a metadata template.

imageDims Metadata for the new image object. These values override any from the metadata object or data array. See MriImage class documentation for details.

voxelDims Metadata for the new image object. These values override any from the metadata object or data array. See MriImage class documentation for details.

voxelDimUnits Metadata for the new image object. These values override any from the metadata object or data array. See MriImage class documentation for details.

origin Metadata for the new image object. These values override any from the metadata object or data array. See MriImage class documentation for details.

tags Metadata for the new image object. These values override any from the metadata object or data array. See MriImage class documentation for details.

fun A function, of the appropriate arity.

... Additional argument to fun.

level The threshold level, below which all voxels will be reset.

defaultValue The value to reset to.

checkFormat If TRUE, the function will check for the magic string “DICOM” at byte offset 128. This string should be present, but in reality not all files contain it.

dictionary Ignored.

stopTag An integer vector giving the group and element numbers (in that order) of a DICOM tag, or NULL. If not NULL, the function will stop parsing the DICOM file if the specified tag is encountered. This can be used to speed up the process if a specific tag is required.

ignoreTransferSyntax If TRUE, any transfer syntax stored in the file will be ignored, and the code will try to deduce the transfer syntax using heuristics. This may occasionally be necessary for awkward DICOM files, but is not generally recommended.
**newSparseArrayWithData**

Create a SparseArray object

**Description**

This function creates a SparseArray object from its constituent parts.

**Usage**

```r
newSparseArrayWithData(data, coordinates, dims)
```

**Arguments**

- `data`: A vector of (nonzero) array elements.
- `coordinates`: A matrix with as many rows as `data` has elements, containing the coordinates of each nonzero element in the array.
- `dims`: The dimensions of the array.

**Value**

A SparseArray object.

**Author(s)**

Jon Clayden

**References**

Please cite the following reference when using TractoR in your work:


**nilObject**

The nil object

**Description**

The nil object is an empty object of class SerialisableObject. It can be used as a placeholder where such an object of this class, or one of its subclasses, is required. It serialises to the empty list.
Usage
nilObject()

is.nilObject(object)

Arguments
object Any object.

Value
nilObject returns the nil object. is.nilObject returns TRUE if its argument is identical to the nil object, or if it is equivalent in the sense of serialising to an identical result.

Author(s)
Jon Clayden

References
Please cite the following reference when using TractoR in your work:

See Also
SerialisableObject

Description
This function chooses the singular or plural form of a word based on the length of an associated vector, or an integer.

Usage
pluralise(singular, x = NULL, n = NULL, plural = NULL)
printLabelledValues

Arguments

- **singular**: The singular form of the word.
- **x**: A vector of any mode, whose length is used to choose the correct word form, unless `n` is specified.
- **n**: An integer which is used to choose the correct word form (singular if `n` = 1, plural otherwise). Take priority over `x` if not NULL.
- **plural**: The plural form of the word. If NULL, an 's' is simply appended to the singular form.

Value

Either `singular` or `plural`, as appropriate.

Author(s)

Jon Clayden

References

Please cite the following reference when using TractoR in your work:


**printLabelledValues**  
*Pretty print labelled information*

**Description**

This is a simple function to print a series of labels and associated data values, or key-value pairs.

**Usage**

```r
printLabelledValues(labels, values, outputLevel = OL$Info,
                     leftJustify = FALSE)
```

**Arguments**

- **labels**: A character vector of labels.
- **values**: A character vector of values. Must have the same length as `labels`.
- **outputLevel**: The output level to print the output to. See `setOutputLevel`, in the reportr package.
- **leftJustify**: Logical value: if TRUE the labels will be left justified; otherwise they will be right justified.
Description

The `promote` function promotes a vector argument to a single-column or single-row matrix. Matrix arguments are returned unmodified.

Usage

```r
promote(x, byrow = FALSE)
```

Arguments

- `x` A vector or matrix.
- `byrow` Logical value: if TRUE, a vector will be promoted to a single-row matrix; otherwise a single-column matrix will result.

Value

A matrix version of the `x` argument.

Author(s)

Jon Clayden
References

Please cite the following reference when using TractoR in your work:


See Also

matrix

readDicomDirectory (Read a directory of DICOM files)

Description

This function scans a directory for files in DICOM format, and converts them to a single Analyze/NIfTI-format image of the appropriate dimensionality.

Usage

readDicomDirectory(dicomDir, method = c("internal", "divest"),
                   readDiffusionParams = FALSE, untileMosaics = TRUE, ...)

Arguments

dicomDir Character vector of length one giving the name of a directory containing DICOM files.
method Character string specifying whether to use the internal DICOM reading code or use the divest package.
readDiffusionParams Logical value. Should diffusion MRI parameters (b-values and gradient directions) be retrieved from the files if possible?
untileMosaics Logical value. Should Siemens mosaic images be converted into 3D volumes? This may occasionally be performed in error, which can be prevented by setting this value to FALSE.
... Additional arguments to readDicom, if the divest method is used.

Value

A list containing elements

image An MriImage object.
bValues Diffusion b-values, if requested. Will be NA if the information could not be found in files.
bVectors Diffusion gradient vectors, if requested. Will be NA if the information could not be found in the files.
readDicomFile

Author(s)
Jon Clayden

References
Please cite the following reference when using TractoR in your work:

See Also
DicomMetadata, MriImage, sortDicomDirectories.

readDicomFile  Read a DICOM file into a DicomMetadata object

Description
This function reads a DICOM file into a DicomMetadata object. Only DICOM files from magnetic resonance scanners are supported.

Usage
readDicomFile(fileName, checkFormat = TRUE, stopTag = NULL,
ignoreTransferSyntax = FALSE, ascii = TRUE)

Arguments
fileName The name of a DICOM file.
checkFormat If TRUE, the function will check for the magic string "DICM" at byte offset 128. This string should be present, but in reality not all files contain it.
stopTag An integer vector giving the group and element numbers (in that order) of a DICOM tag, or NULL. If not NULL, the function will stop parsing the DICOM file if the specified tag is encountered. This can be used to speed up the process if a specific tag is required.
ignoreTransferSyntax If TRUE, any transfer syntax stored in the file will be ignored, and the code will try to deduce the transfer syntax using heuristics. This may occasionally be necessary for awkward DICOM files, but is not generally recommended.
ascii If TRUE, the function will attempt to read an embedded Siemens ASCII header, if one exists.

Value
readDicomFile returns a DicomMetadata object, or NULL on failure.
resolvePath

Author(s)

Jon Clayden

References

Please cite the following reference when using TractoR in your work:

See Also

The DICOM standard, found online at http://dicom.nema.org/. (Warning: may produce headaches!) Also readDicomDirectory for information on how to create MriImage objects from DICOM files.

resolvePath Functions for file name and path manipulation

Description

Functions for expanding file paths, finding relative paths and ensuring that a file name has the required suffix.

Usage

resolvePath(path, ...)
relativePath(path, referencePath)
matchPaths(path, referencePath)
registerPathHandler(regex, handler)
expandFileName(fileName, base = getwd())
ensureFileSuffix(fileName, suffix, strip = NULL)

Arguments

path, referencePath
    Character vectors whose elements represent file paths (which may or may not currently exist).
...
    Additional arguments to custom path handlers.
regex
    A Ruby-style regular expression.
handler
    A function taking and returning a string.
fileName
    A character vector of file names.
base
If fileName is a relative path, this option gives the base directory which the path is relative to. If fileName is an absolute path, this argument is ignored.

suffix
A character vector of file suffixes, which will be recycled if shorter than fileName.

strip
A character vector of suffixes to remove before appending suffix. The intended suffix does not need to be given here, as the function will not append it if the specified file name already has the correct suffix.

Details
The resolvePath function passes its arguments elementwise through any matching path handler, and returns the resolved paths. Nonmatching elements are returned as-is. registerPathHandler registers a new path handler for special syntaxes, and is for advanced use only. relativePath returns the specified path, expressed relative to referencePath. matchPaths resolves a vector of paths against a vector of reference paths. expandFileName returns the full path to the specified file name, collapsing ".." elements if appropriate. ensureFileSuffix returns the specified file names with the requested suffixes appended (if they are not already).

Value
A character vector.

Author(s)
Jon Clayden

References
Please cite the following reference when using TractoR in your work:

See Also
normalizePath does most of the work for expandFileName.

---

resolveVector

Miscellaneous vector functions

Description
These functions provide the (Euclidean) length of a vector, the vector cross product or angle between two vectors.
resolveVector

Usage

resolveVector(len, ...)

vectorLength(vector)

vectorCrossProduct(a, b)

angleBetweenVectors(v1, v2)

Arguments

len The expected length of the vector.
...
Elements of the vector, to be concatenated together.
vector, v1, v2 Numeric vectors of any length.
a, b Numeric 3-vectors.

Value

For vectorLength, the Euclidean norm or length of the specified vector, given by $\sqrt{\text{sum}(\text{vector}^2)}$.
For vectorCrossProduct, the vector cross product of the two specified vectors; and for angleBetweenVectors, the angle (in radians) between the two specified vectors. The resolveVector function concatenates the values given in ..., and if the result is a vector of length len then it is returned. If not, NULL is returned.

Author(s)

Jon Clayden

References

Please cite the following reference when using TractoR in your work:


See Also

crossprod for the matrix cross product.
The SerialisableObject class

Description

This reference class extends the standard envRefClass class, adding a function for simple serialization of the data fields of an object, and one for finding all of the methods available for an object. A serialised object may be deserialised using the deserialiseReferenceObject function.

Methods

fields() Retrieve a list of all field names
serialise(file = NULL) Serialise the object to a list or file

sortDicomDirectories Sort a directory of DICOM files into series

Description

This function sorts a directory containing DICOM files into subdirectories by series UID (DICOM tag 0x0020,0x000e), subject name (0x0010,0x0010) and/or scan date (0x0008,0x0020). Each unique identifier, together with its description for series, will be used as the name for a new subdirectory, and all relevant files will be copied into that subdirectory. Duplicate file names are disambiguated if necessary.

Usage

sortDicomDirectories(directories, method = c("internal", "divest"),
deleteOriginals = FALSE, sortOn = "series", seriesId = c("UID",
"number", "time"), nested = TRUE, ...)

Arguments

directories A character vector giving the directories to search for DICOM files. Subdirectories will also be searched.
method Character string specifying whether to use the internal DICOM reading code or use the divest package.
deleteOriginals A single logical value. If TRUE, then the source files will be deleted after being copied to their new locations, making the operation a move rather than a copy. Nothing will be deleted if the copy fails.
sortOn The string "series", "subject" or "date", or any combination in the order desired. This will be the basis of the sort, which will be nested if more than one type is specified.
SparseArray-class

**seriesId**
A string describing the kind of series identifier to use for sorting by series:
"UID" (DICOM tag 0x0020,0x000e; the default), "number" (0x0020,0x0011)
or "time" (0x0008,0x0031).

**nested**
Logical value. If TRUE and directories is of length 1, subdirectories will becreated within the specified original directory. Otherwise they will be created inthe working directory.

... Additional arguments to pass to readDicomFile.

**Value**
This function is called for its side effect.

**Author(s)**
Jon Clayden

**References**
Please cite the following reference when using TractoR in your work:

**See Also**
readDicomDirectory for reading DICOM files into an MriImage object.

---

**SparseArray-class**

*The SparseArray class*

**Description**
This class represents an array with any number of dimensions, in which a significant proportionof entries are zero. The coordinates of nonzero entries are stored along with their values, with allremaining entries assumed to be zero. Methods are provided to index into the array in the standardway, using matrix or vector indices; and for coercing between SparseArray objects and standard(dense) arrays.

**Fields**
- **data**: Vector of nonzero data values
- **coords**: Integer matrix of nonzero data locations, one per row
- **dims**: Integer vector of dimensions
Methods

aperm(perm) Permute the dimensions of the array
apply(margin, fun, ...) Apply a function to margins of the array
flip(dimsToFlip) Flip the array along one or more directions
setCoordinatesAndData(newCoords, newData) Update the nonzero locations and data values in the array
setDimensions(newDims) Change the dimensions of the image

threadSafeTempFile  Obtain thread-safe temporary file names

Description

This function is a wrapper around `tempfile`, which creates temporary file names whose path contains the process ID of the calling process. This avoids clashes between threads created by functions such as `mclapply` (in the “parallel” package), which can easily occur with the standard `tempfile` function.

Usage

`threadSafeTempFile(pattern = "file")`

Arguments

`pattern` Character vector giving the initial part of each file name.

Value

A character vector of temporary file names. No files are actually created.

Author(s)

Jon Clayden

References

Please cite the following reference when using TractoR in your work:

See Also

tempfile
Indexing methods

Description

Indexing methods for \texttt{SparseArray} and \texttt{MriImage} objects. For the latter class, arguments are passed to the equivalents for \texttt{array} or \texttt{SparseArray}, except where \(i\) is another \texttt{MriImage} object, where its nonzero region will be used to provide the indices. For \texttt{SparseArray}, indexing may be blank, or by numeric vector or matrix.

Usage

```r
## S4 method for signature 'SparseArray,ANY,ANY'
x[i, j, ..., drop = TRUE]

## S4 replacement method for signature 'SparseArray,ANY,ANY'
x[i, j, ...] <- value

## S4 method for signature 'MriImage,missing,missing'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'MriImage,ANY,missing'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'MriImage,missing,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'MriImage,ANY,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'MriImage,MriImage,missing'
x[i, j, ..., drop = TRUE]

## S4 replacement method for signature 'MriImage,missing,missing'
x[i, j, ...] <- value

## S4 replacement method for signature 'MriImage,ANY,missing'
x[i, j, ...] <- value

## S4 replacement method for signature 'MriImage,missing,ANY'
x[i, j, ...] <- value

## S4 replacement method for signature 'MriImage,ANY,ANY'
x[i, j, ...] <- value

## S4 replacement method for signature 'MriImage,MriImage,missing'
x[i, j, ...] <- value
```
\[.\text{SparseArray}, \text{ANY}, \text{ANY}-\text{method}\]

\[
x[i, j, \ldots] <- \text{value}
\]

**Arguments**

- **x**: An object of the appropriate type.
- **i, j, \ldots**: Indexing objects.
- **drop**: Scalar value: should unitary dimensions be dropped?
- **value**: New value(s) for replacement forms.

**Value**

A vector, array or **SparseArray**.

**Author(s)**

Jon Clayden
Index

[r, MriImage, ANY, ANY-method
   ([, SparseArray, ANY, ANY-method), 39
[r, MriImage, ANY, missing-method
   ([, SparseArray, ANY, ANY-method), 39
[r, MriImage, MriImage, missing-method
   ([, SparseArray, ANY, ANY-method), 39
[r, MriImage, missing, ANY-method
   ([, SparseArray, ANY, ANY-method), 39
[r, MriImage, missing, missing-method
   ([, SparseArray, ANY, ANY-method), 39
[r, SparseArray, ANY, ANY-method, 39
<r, MriImage, ANY, ANY-method
   ([, SparseArray, ANY, ANY-method), 39
<r, MriImage, ANY, missing-method
   ([, SparseArray, ANY, ANY-method), 39
<r, MriImage, MriImage, missing-method
   ([, SparseArray, ANY, ANY-method), 39
<r, MriImage, missing, ANY-method
   ([, SparseArray, ANY, ANY-method), 39
<r, MriImage, missing, missing-method
   ([, SparseArray, ANY, ANY-method), 39
<r, SparseArray, ANY, ANY-method
   ([, SparseArray, ANY, ANY-method), 39

all.equal, 12
allEqual, 2
angleBetweenVectors (resolveVector), 34
array, 23
asMriImage, 3
colorRamp, 14
colours, 14
copyImageFiles
   (/identifyImageFileNames), 15
createContactSheetGraphic
   (createSliceGraphic), 6
createNeighbourhoodInfo, 5
createProjectionGraphic
   (createSliceGraphic), 6
createSliceGraphic, 6
crossprod, 35
deduplicate, 7
defaultInfoPanel, 8
deserialiseReferenceObject, 36
deserialiseReferenceObject
   (/isDeserialisable), 20
DicomMetadata, 32
DicomMetadata (DicomMetadata-class), 9
DicomMetadata-class, 9
embrace, 10
emptyMatrix, 11
ensureFileSuffix (resolvePath), 33
envRefClass, 36
equivalent, 2, 3, 12
execute (locateExecutable), 21
expandFileName (resolvePath), 33
extractDataFromMriImage
   (/newMriImageFromDicomDirectory), 24
extractMriImage (asMriImage), 3
fi (fx), 13
file.copy, 16
file.symlink, 16
fx, 13
fxy (fx), 13
fxyz (fx), 13
getcher, 6–9, 13
identifyImageFileNames, 15
image, 6, 7
imageFileExists
    (identifyImageFileNames), 15
implode, 17
indexList, 18
infix, 19
is.emptyMatrix (emptyMatrix), 11
is.nilObject (nilObject), 27
isDeserialisable, 20
load, 20, 21
locateExecutable, 21
matchPaths (resolvePath), 33
Math, 23
matrix, 31
mergeMriImages, 22
MriImage, 4–6, 16, 17, 20, 23, 26, 31–33, 39
MriImage (MriImage-class), 23
MriImage-class, 23
neighbourhoodInfo
    (createNeighbourhoodInfo), 5
newDicomMetadataFromFile
    (newMriImageFromDicomDirectory), 24
newMriImageByExtraction
    (newMriImageFromDicomDirectory), 24
newMriImageByMasking
    (newMriImageFromDicomDirectory), 24
newMriImageByReordering
    (newMriImageFromDicomDirectory), 24
newMriImageByThresholding
    (newMriImageFromDicomDirectory), 24
newMriImageByTrimming
    (newMriImageFromDicomDirectory), 24
newMriImageFromDicomDirectory, 24
newMriImageFromFile
    (newMriImageFromDicomDirectory), 24
newMriImageWithBinaryFunction
    (newMriImageFromDicomDirectory), 24
newMriImageWithData (newMriImageFromDicomDirectory), 24
newMriImageWithSimpleFunction (newMriImageFromDicomDirectory), 24
newSparseArrayWithData, 27
nilObject, 27
normalizePath, 34
Ops, 23
options, 16
paste, 18
paths (resolvePath), 33
pluralise, 28
polarPlotPanel (defaultInfoPanel), 8
printLabelledValues, 29
promote, 30
readDicomDirectory, 31, 33, 37
readDicomFile, 32, 37
readImageFile (identifyImageFileNames), 15
registerDeserialiser (isDeserialisable), 20
registerPathHandler (resolvePath), 33
relativePath (resolvePath), 33
removeImageFiles
    (identifyImageFileNames), 15
removeImageFilesWithName
    (newMriImageFromDicomDirectory), 24
reorderMriImage, 23
reorderMriImage (asmriImage), 3
resolvePath, 15, 33
resolveVector, 34
rgb, 14
save, 20, 21
SerialisableObject, 20, 21, 27, 28
SerialisableObject (SerialisableObject-class), 36
SerialisableObject-class, 36
serialisation (isDeserialisable), 20
serialiseReferenceObject (isDeserialisable), 20
setOutputLevel, 30
sortDicomDirectories, 32, 36
SparseArray, 16, 25, 27, 39, 40
SparseArray (SparseArray-class), 37
SparseArray-class, 37
Summary, 23
symlinkImageFiles
  (identifyImageFileNames), 15
system, 21
system2, 21, 22
tempfile, 38
threadSafeTempFile, 38
timeSeriesPanel (defaultInfoPanel), 8
trimMriImage, 6
trimMriImage (asMriImage), 3
unlink, 16
vectorCrossProduct (resolveVector), 34
vectorLength (resolveVector), 34
viewImages, 6, 7
viewImages (defaultInfoPanel), 8
visualisation (createSliceGraphic), 6
writeImageFile, 20, 21
writeImageFile
  (identifyImageFileNames), 15
writeMriImageToFile
  (newMriImageFromDicomDirectory), 24