Package 'tractor.base'

May 8, 2026

Version 3.5.0

Date 2025-10-13

Title Read, Manipulate and Visualise Magnetic Resonance Images

Imports methods, ore (>= 1.3.0), RNifti (>= 1.8.0), reportr, shades

Suggests mmand, loder, divest, yaml, tinytest

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

License GPL-2

URL https://www.tractor-mri.org.uk, https://github.com/tractor/tractor

BugReports https://github.com/tractor/tractor/issues

RoxygenNote 7.3.2

NeedsCompilation no

Author Jon Clayden [cre, aut] (ORCID: <https://orcid.org/0000-0002-6608-0619>)

Maintainer Jon Clayden <code@clayden.org>

Repository CRAN

Date/Publication 2025-10-29 14:30:02 UTC

Contents

allEqual *Test whether all elements of a vector are equal*

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

```
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:

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

See Also

equivalent for elementwise equivalence of two vectors.

Examples

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

`asDiffusionScheme` *Create a DiffusionScheme object from data*

Description

This is a generic function that converts another object to a `DiffusionScheme` by extracting the relevant information. There are methods for matrices (of gradient directions) and `MriImage` objects. An image passed as an argument to the latter must have the `bValues` and `bVectors` tags set appropriately.

Usage

```
asDiffusionScheme(x, ...)

## S3 method for class 'matrix'
asDiffusionScheme(x, bValues, ...)

## S3 method for class 'MriImage'
asDiffusionScheme(x, ...)

## S3 method for class 'DiffusionScheme'
asDiffusionScheme(x, ...)
```

Arguments

`x`	An object to coerce to a `DiffusionScheme`.
`...`	Additional arguments to methods.
`bValues`	A vector of b-values, required when `x` is a matrix of gradient directions.

Value

A `DiffusionScheme` object.

Author(s)

Jon Clayden

References

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

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

asMriImage *Creating MriImage objects from data*

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 fucntion for creating an object from its constituents: an array of voxel values and some metadata (and/or a template image).

Usage

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

extractMriImage(image, dim, loc)

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

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:

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

See Also

MriImage

---

augmentedInfoPanel *A simple interactive viewer for MriImage objects*

Description

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

Usage

```
augmentedInfoPanel(indexNames = NULL)

polarPlotPanel(directions, bValues = NULL)

viewImages(images, colourScales = NULL, point = NULL, interactive = TRUE,
  crosshairs = TRUE, orientationLabels = TRUE,
  infoPanel = RNifti::defaultInfoPanel, ...)
```

Arguments

indexNames	A list whose elements are either NULL or a named character vector giving the names associated with each index in the image.
directions	A matrix of 3D acquisition direction vectors, one per row.
bValues	A vector of b-values, if the image is diffusion-weighted.
images	An MriImage object, or list of MriImage objects.
colourScales	A list of colour scales to use for each image, which will be recycled to the length of images. See getColourScale for details. The default is to use greyscale.

point	A length-3 integer vector giving the initial location of the crosshairs, in voxels.
interactive	A single logical value. If TRUE, the plot is interactive.
crosshairs	A single logical value. If TRUE, the crosshairs are displayed.
orientationLabels	A single logical value. If TRUE, orientation labels are displayed.
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 from package RNifti 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:

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

See Also

getColourScale

---

createNeighbourhoodInfo

Image neighbourhoods

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:

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

---

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

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

createContactSheetGraphic(image, axis, 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:

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

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, possibly with subsequent elements' content appended to it. Unnamed elements are retained.

Usage

```
deduplicate(..., merge = FALSE)
```

Arguments

`...`	One or more vectors of any mode, usually named.
`merge`	If `FALSE`, the default, duplicate elements will simply be discarded. If `TRUE`, additional elements with the same name will be appended to the retained one. This does not apply to unnamed elements. If this kind of deduplication actually happens, the return value will be a list, regardless of the source type.

Value

The concatenated and deduplicated vector.

Author(s)

Jon Clayden

References

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

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

DicomMetadata-class *The DicomMetadata class*

Description

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

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

DiffusionScheme-class *The DiffusionScheme class*

Description

This class represents a diffusion MRI acquisition scheme. It encapsulates a series of diffusion-weighted volume acquisitions, each with an associated diffusion-sensitising gradient direction, in one or more "shells" with a given weighting ("b-value").

Fields

`bValues` A vector of b-values in seconds per square millimetre, one per volume acquired.

`gradientDirections` A matrix of gradient directions, one row per volume acquired. Columns give the x, y and z components of the directions, relative to the image axes.

`shellIndices` An integer vector giving the shell index associated with each volume. This is calculated internally and coded such that every volume with a b-value below `unweightedThreshold` gets an index of 0, and then others are grouped into shells with b-values differing by less than the relative `tolerance` specified (2 grouping is indicative only, and not stored in direction files.

`shellValues` A vector of b-values associated with each shell. These are calculated as the mode of the assigned volume b-values in each case.

Methods

`summarise()` Summarise key aspects of the object

`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:

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

Examples

```
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

```
emptyMatrix()

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:

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

`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

```
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:

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

See Also

`all.equal`

Examples

```
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
```

FileMap-class *The FileMap class*

Description

A reference class to represent a file map and handle persisting it to file. A file map is a directory-specific dictionary keyed by file names (without suffixes), whose values point to the actual locations of the associated files. This saves duplicating large files such as images, in a way similar to symbolic linking, but without requiring special file system support, and with only one link per file set rather than one per file. The map is serialised in YAML format to a file called `map.yaml` in the relevant directory. Reading from this file happens on object creation and when the `read` method is called; writing is only by an explicit call to the `write` method.

Fields

`directory` A character string representing the directory being mapped.

`map` A list representation of the map in memory.

Methods

`dropElements(keys)` Remove elements with the specified keys from the in-memory map

`getElements(keys)` Return the values associated with the specified keys

`getFile()` Return the path to the map file, which may not yet exist

`getMap(sorted = FALSE)` Return the in-memory map, optionally sorted by key

`initialize(path = "", ...)` Create a FileMap object for the specified paths

`read()` Read the map file into memory

`setElements(keys, values)` Replace the specified keys with new values, which should be strings

`write()` Write the in-memory map to file, or delete the file if it is empty

FileSet-class *The FileSet class*

Description

This reference class manages sets of related files based on a common file stem and required and/or optional suffixes. It is designed for handling alternative file formats or composite file types. The class represents the group of file formats in the abstract; its methods handle specific files and support operations such as copying, moving, symlinking, validating and deleting those files, ensuring that all constituent files are handled consistently.

Fields

`formats` A named list mapping format names to required file suffixes.

`validators` An optional named list of validation functions for some or all of the supported formats.

`auxiliaries` A character vector of optional auxiliary file suffixes.

Methods

`atPaths(paths)` Return a handle object to manipulate files at specific paths

`fileStem(paths)` Obtain a file stem for each specified path, dropping format-specific suffixes

`findFormat(paths, intent = c("read", "write"), preference = NULL, all = FALSE)` Identify and potentially validate files to find the format used at specific paths

`resolvePaths(paths)` Perform any special path handling and resolve final paths

`subset(match, ...)` Create a variant object that only encapsulates the specified subset of formats

See Also

`ImageFileSet` is a subclass specialised for image files.

`fillShells` *Assign image volumes to shells*

Description

This function guesses the assignment of image volumes to shells, expanding a list of unique b-values to one value per volume. This is achieved by performing k-means clustering on the mean intensities of each volume. The set of unique b-values must be known beforehand, and no sanity checking is performed on the intensities for each shell except ordering.

Usage

```
fillShells(image, bValues)
```

Arguments

`image`	A 4D diffusion-weighted image, whose mean volume intensities will be used to cluster the acquisition into shells. It does not need to already have gradient direction or b-value information associated with it.
`bValues`	A numeric vector of unique b-values.

Value

A numeric vector expanding the original `bValues` to one value per volume of the `image`.

Note

The procedure performed by this function is a heuristic, and may produce inaccurate results if some shell b-values are close together, if there are many unique b-values, if the assumption of decreasing mean intensity with increasing b-value doesn't hold, etc. If the full acquisition scheme is known then it is alway better to specify it explicitly.

Author(s)

Jon Clayden

References

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

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

`fx` *Shorthand anonymous functions*

Description

These functions provide a shorthand route to simple anonymous functions.

Usage

```
fx(expr)

fxy(expr)

fxyz(expr)

fi(expr)
```

Arguments

`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:

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

`getColourScale` *Obtaining colour scales*

Description

The `getColourScale` function can be used to obtain a standard or customised colour scale for use in the package's image visualisation functions.

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 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:

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

See Also

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

Examples

```
getColourScale(1)
```

`identifyImageFileNames` *Resolve image paths*

Description

This function identifies characteristics of an image on disk, including associated auxiliary files. Its functionality has essentially been superseded by methods of the `ImageFileSet` class, but it remains as an alternative interface to that functionality for backwards compatibility.

Usage

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

Arguments

`fileName`	A character vector of image paths.
`errorIfMissing`	Logical value: raise an error if no suitable files were found?
`auxiliaries`	A character vector of auxiliary file suffixes to search for.
`...`	Additional arguments to `resolvePath`.

Value

A list with the following elements, describing the identified files:

fileStem The file name without extension.

headerFile The full header file name.

imageFile The full image file name.

auxiliaryFiles The full path to any auxiliary files.

format The basic format of the files (`"Nifti"`, `"Analyze"`, etc.).

headerSuffix The file suffix associated with the header file.

imageSuffix The file suffix associated with the image file.

auxiliarySuffixes The file suffixes associated with any auxiliary files.

Author(s)

Jon Clayden

References

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

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

See Also

`imageFiles`, `resolvePath`

`imageFiles` *Image file sets*

Description

TractoR supports several medical imaging file formats, some of which are made up of multiple files on disk or have compressed and uncompressed variants. There is also support for auxiliary "sidecar" files, which have the same path stem as the associated image files but different suffixes, and contain additional metadata. Typically the image files are in a binary format for space-efficiency, while any auxiliary files are in text-based formats for easier human readability. This interface, and the unpinning reference classes, handle and abstract away this complexity.

Usage

```
imageFiles(paths = NULL)

imageFileExists(fileName)

removeImageFiles(fileName, ...)

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

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

Arguments

paths	Optionally, a character vector of image paths with or without suitable suffixes.
fileName, from	Character vectors of image paths.
...	Currently unused.
to	Character vector of target file names, or a single existing directory to copy or symlink into.
overwrite	Logical value: if TRUE, existing files will be overwritten; otherwise creating files with existing names will just fail.
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")
deleteOriginals	Logical value: if TRUE, copyImageFiles performs a move rather than a copy.

Details

The imageFiles function returns either a preinitialised instance of the ImageFileSet class, which handles file set logic in the abstract, or (if given a vector of paths as an argument) calls that instance's atPaths method, which facilitates handling specific files. If non-default parameters to the class constructor are required then a custom ImageFileSet object can be created directly instead.

imageFileExists, removeImageFiles, symlinkImageFiles and copyImageFiles are simple wrapper functions that exist for backwards compatibility. copyImageFiles(from, to), for example, is equivalent to imageFiles(from)$copy(to).

Value

If paths is NULL, imageFiles returns a singleton reference object of class ImageFileSet which can be used to identify and manipulate image files anywhere on the file system. If paths is specified, an S3 object of class fileSetHandle, a list of functions which can be used to manipulate the actual files at those paths. The functions are

stems() Return the file stems associated with the paths.

info() Return information about existing files, as a list with one element per path. An element will be NULL if no corresponding files currently exist.

read(...) Read the images into memory and return them as MriImage objects.

copy(target, overwrite=TRUE) Copy the files to target paths (new file names or a directory).

delete() Delete the files or any map reference to them.

map(target, overwrite=TRUE, relative=TRUE) Map the files to a new location (see FileMap for details).

move(target, overwrite=TRUE) Move the files to target paths (new file names or a directory).

present() Return Boolean values indicating whether or not files exist at each path.

symlink(target, overwrite=TRUE, relative=TRUE) Symlink the files to target paths (if supported by the OS and file system).

imageFileExists returns a logical vector indicating whether or not valid image files exist at each specified path. Other functions are called for their side-effects.

Author(s)

Jon Clayden

References

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

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

See Also

Using `ImageFileSet` provides a lower-level and more flexible interface; `readImageFile` can be used if you just want to read an image into memory.

Examples

```
path <- system.file("extdata", "analyze", "maskedb0.img.gz",
  package="tractor.base")
im <- imageFiles(path)
print(im)
im$present()
```

ImageFileSet-class *The ImageFileSet class*

Description

`ImageFileSet` is a subclass of `FileSet` that is specialised for images, by pre-populating the format and auxiliary file specification (although this can be overridden) and by supporting file mapping using the `FileMap` class. If a file set exists on disk and in a map then the files take priority.

Fields

`defaultMap` A list representing a default file map which will be consulted in the absence of on-disk files or an existing file map to try to resolve paths.

Methods

`atPaths(paths)` Return a handle object to manipulate files at specific paths

`findFormat(paths, intent = c("read", "write"), preference = NULL, all = FALSE)` Identify and potentially validate files to find the format used at specific paths

`resolvePaths(paths)` Perform any special path handling and resolve final paths

`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.

Author(s)

Jon Clayden

References

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

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

See Also

`paste`

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"
```

`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.

Author(s)

Jon Clayden

References

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

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

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:

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

See Also

where, which resolves a value if an expression is TRUE. Several calls to that function can be conveniently chained together with this one.

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

```
isDeserialisable(object, expectedClass = NULL)

serialiseReferenceObject(object, file = NULL)

deserialiseReferenceObject(file = NULL, object = NULL, raw = FALSE)

registerDeserialiser(className, deserialiser)
```

Arguments

object	For serialiseReferenceObject, any object, but only those inheriting from SerialisableObject and loso objects are treated specially. 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 (if specified) inherits from the required class, otherwise FALSE. If its argument is serialisable then serialiseReferenceObject returns the serialised version, invisibly; otherwise it returns the unmodified object, again invisibly. 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:

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

See Also

SerialisableObject, save, load, loso, writeImageFile.

---

locateExecutable *Find or run an external executable file*

Description

The execute function is a wrapper around the system2 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)

execute(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:

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

See Also

`system2`

---

loso *Lists of serialisable objects*

Description

This function creates objects of S3 class `"loso"`, which are standard lists with this class attribute set. These objects can be used within `SerialisableObject` objects for fields that contain multiple objects which may themselves be serialisable. Otherwise their functionality is the same as a normal list.

Usage

```
loso(..., count = NA)
```

Arguments

`...`	List elements.
`count`	If only one element is provided, an optional integer specifying the number of times to repeat the element. This allows lists of a fixed size to be created using a default value.

Value

A list of class `"loso"`.

Note

TractoR's serialisable objects are reference classes, and so naive use of `rep` to create a list of object copies will have reference semantics whereby each object refers to the same set of fields, which is rarely the intention. This function explicitly creates a deep copy for each replicate to avoid this.

Author(s)

Jon Clayden

References

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

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

See Also

`SerialisableObject`, `save`, `load`, `writeImageFile`.

`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`, `NA`s 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:

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

See Also

`MriImage`

MriImage-class *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

`getDataAtPoint(...)` Obtain the value of the image at a particular point

`getMetadata()` Obtain a version of the image with any data removed

`getNonzeroIndices(array = TRUE, positiveOnly = FALSE)` Find voxels whose values are not zero

`getSlice(dim, loc)` Extract data from a slice of the image along one dimension

`getSparseness()` Obtain the proportion of zeroes in the image

`getTags(keys = NULL)` Retrieve some or all of the tags stored with the image

`getXform(implicit = TRUE)` Retrieve the stored or implicit xform matrix

`map(fun, ..., sparse = NULL)` Replace the current data with the result of a function

`mask(maskImage)` Mask the image, setting zero voxels in the mask to zero

`setData(newData)` Replace the data in the image

`setOrigin(newOrigin)` Update the origin of the image

`setSource(newSource)` Update the source of the image

`setTags(..., merge = FALSE)` Add, replace or merge metadata tags

`setXform(newXform)` Update the xform matrix associated with the image

`summarise()` Summarise key aspects of the object

`threshold(level, defaultValue = 0)` Threshold the image by setting values below the threshold level to zero

---

newSparseArrayWithData

Create a SparseArray object

Description

This function creates a `SparseArray` object from its constituent parts.

Usage

```
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:

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

---

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:

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

See Also

SerialisableObject

`Optional` *Optional types*

Description

This function declares an optional type, a union of the named class (which must already be defined) and `NULL`. It is meant for use in packages.

Usage

```
Optional(className)
```

Arguments

`className`	A string naming an existing (S4) class.

Value

The name of the union class (invisibly), which will be the original class name with "Optional" prepended to it.

Author(s)

Jon Clayden

References

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

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

See Also

setClassUnion

`pluralise` *Number agreement with a vector*

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)
```

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:

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

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

```
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.

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:

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

See Also

setOutputLevel for the reportr output level system.

promote *Promote a vector to a single-column or single-row matrix*

Description

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

Usage

```
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:

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

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.

Author(s)

Jon Clayden

References

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

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

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.

Author(s)

Jon Clayden

References

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

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

See Also

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

---

readDiffusionScheme *Read and write diffusion schemes*

Description

These functions read diffusion acquisition scheme objects from, and write them to, text-based matrix files. They can handle both FSL-style (two-file) and TractoR-style (single file) formats; in the two-file case either file name can be specified.

Usage

```
readDiffusionScheme(fileName, bValues = NULL, imagePath = NULL, ...)

writeDiffusionScheme(scheme, fileName)
```

Arguments

fileName	A string specifying a file name or stem.
bValues	A numeric vector of b-values, or a single string naming a file containing them (in a format readable by read.table). The default is NULL, meaning that they should be read from the fileName or a sidecar "bval" file.
imagePath	An optional string giving the path to an image. If fewer b-values are given than the number of gradient directions in the file, the image will be read and passed to fillShells to fill in the gaps. This is a convenience feature, but is heuristic-based and so may not always be reliable.
...	Further arguments to the DiffusionScheme constructor, to adjust how shells are interpreted.
scheme	A DiffusionScheme object to write to file.

Details

Three main naming conventions for these files are recognised. TractoR's preferred format is a single file with a ".dirs" extension, with the direction information stored one volume per row and b-values in the final column. FSL uses files called "bvecs" and "bvals", with values stored one volume per column; if the specified basic file name is exactly one of these two then this format is assumed. BIDS uses the same format, but in files named for the associated image with ".bvec" and ".bval" extensions. Any other naming convention can be forced when writing by wrapping the file name in a call to I, the "as-is" function.

Value

readDiffusionScheme returns a DiffusionScheme object, or NULL if one cannot be read. writeDiffusionScheme is called for its side effect.

Author(s)

Jon Clayden

References

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

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

---

readImageFile *Reading and writing image files*

Description

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

Usage

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

writeImageFile(image, fileName = NULL,
  fileType = getOption("tractorFileType"), overwrite = TRUE,
  datatype = "fit", writeTags = FALSE)
```

Arguments

fileName	A file path, with or without appropriate extension.
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 optional MriImage object representing a mask, outside of which the image to be read should be considered to be zero. This can be used to save memory when only a small part of a large image is of interest. Ignored if sparse is not TRUE.
reorder	Logical value: should the image data be reordered to LAS? This is recommended in most circumstances.

...	Additional arguments to identifyImageFileNames.
image	An MriImage object.
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.
overwrite	Logical value: overwrite an existing image file? An error will be raised if there is an existing file and this is set to FALSE.
datatype	A datatype string, such as "uint8" or "float", specifying the pixel datatype to use when storing the data. If specified, this must be a type supported by the requested (or default) file format. The default, "fit", results in a datatype being chosen that is wide enough to fit the range of the data elements. An error will arise if there's no such type.
writeTags	Logical value: should tags be written in YAML format to an auxiliary file?

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, and MRtrix format by the software of the same name. 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/MRtrix 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 for writing) are "NIFTI", "NIFTI_PAIR" (the two-file NIfTI format), "MGH", and corresponding gzipped versions of these with "_GZ" appended. File types "ANALYZE" and "MRTRIX", and "_GZ" variants, are additionally available for reading only. "NIFTI_GZ" 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.

Value

readImageFile returns an MriImage object. writeImageFile returns a list giving details of the file paths that were written to.

Author(s)

Jon Clayden

References

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

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

See Also

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

---

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:

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

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.

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(sum(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:

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

See Also

crossprod for the matrix cross product.

SerialisableObject-class

The SerialisableObject class

Description

This reference class extends TractorObject by adding a function for simple serialisation of the data fields of an object, either to a list or a file. This is intended to be used for classes whose state can meaningfully be restored from a list of standard R objects (not including transient C/C++ pointers, for example). A serialised object may be deserialised using the deserialiseReferenceObject function.

Methods

serialise(file = NULL) Serialise the object to a list or file

See Also

save

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.
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 be created within the specified original directory. Otherwise they will be created in the 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:

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

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 proportion of entries are zero. The coordinates of nonzero entries are stored along with their values, with all remaining entries assumed to be zero. Methods are provided to index into the array in the standard way, 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

`summarise()` Summarise key aspects of the object

`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:

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

See Also

`tempfile`

TractorObject-class *The TractorObject class*

Description

This reference class extends the standard envRefClass class, adding methods for finding all of the field or methods available for an object. There is also a method for summarising key elements of the object as a named character vector, which can be suitable overridden by inheriting classes. The show method prints this summary as a labelled list.

Methods

`fields()` Retrieve a list of all field names

`methods()` Retrieve a list of all method names

`summarise()` Summarise key aspects of the object

where *Compact conditional values*

Description

This simple function checks whether its first argument is a logical value that evaluates to TRUE. If so, it returns its second argument. If not, it returns its third argument.

Usage

```
where(condition, value, fallback = NULL)
```

Arguments

condition	An expression that resolves to a single logical value.
value, fallback	Any expression.

Details

This function differs from the standard ifelse function in that it does not act elementwise, and that the third argument is optional, defaulting to NULL.

Value

value, if condition evaluates to TRUE; otherwise fallback.

Author(s)

Jon Clayden

References

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

J.D. Clayden, S. Muñoz Maniega, A.J. Storkey, M.D. King, M.E. Bastin & C.A. Clark (2011). TractoR: Magnetic resonance imaging and tractography with R. Journal of Statistical Software 44(8):1-18. doi:10.18637/jss.v044.i08.

See Also

ifelse

`[,SparseArray,ANY,ANY-method`

Indexing methods

Description

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

Usage

```
## 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
```

Arguments

x	An object of the appropriate type.
i, j, ...	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