Package 'neuroim'

January 27, 2015

Imports Matrix, yaImpute,

License GPL (>= 2)

Maintainer Bradley Buchsbaum <brad.buchsbaum@gmail.com>

Title neuroim: R software for reading, writing and representing brain imaging data.

Type Package

Author Bradley R. Buchsbaum

Description neuroim is a collection of data structures that represent volumetric brain imaging data. The focus is on basic data handling for 3D and 4D neuroimaging data. In addition, there are function to read and write ANALYZE7.5 and NIFTI files and limited support for reading AFNI files.

Version 0.0.3

Date 2014-04-01

Depends R (>= 3.0.0), stringr, hash, iterators, abind, methods

Collate 'AFNI_IO.R' 'AllGeneric.R' 'AllClass.R' 'Axis.R' 'BinaryIO.R' 'BrainData.R' 'common.R' 'NIFTI_IO.R' 'BrainFileDescriptor.R' 'BrainMetaInfo.R' 'BrainRegion3D.R' 'BrainSlice.R' 'BrainSpace.R' 'SparseBrainVector.R' 'BrainVector.R' 'BrainVolume.R' 'Display.R' 'IndexLookupVolume.R' 'Ops.R' 'conncomp.R'

NeedsCompilation no

Repository CRAN

Date/Publication 2014-04-04 08:13:51

R topics documented:

addDim *Generic function to add a dimension to an object*

Description

Generic function to add a dimension to an object

add dimension to BrainSpace

Usage

```
addDim(x, n)

## S4 method for signature 'BrainSpace,numeric'
addDim(x, n)
```

Arguments

x	a dimensioned object
n	the size of the dimension to add

AFNIFileDescriptor-class *This class supports the AFNI file format*

Description

This class supports the AFNI file format

AFNIMetaInfo *AFNIMetaInfo*

Description

Constructor for AFNIMetaInfo class

Usage

```
AFNIMetaInfo(descriptor, afni_header)
```

Arguments

descriptor	an instance of class AFNIFileDescriptor
afni_header	a list returned by readAFNIHeader

Value

an instance of class AFNIMetaInfo

---

as *conversion from ROIVolume to DenseBrainVolume*

Description

conversion from ROIVolume to DenseBrainVolume

conversion from DenseBrainVolume to array

conversion from SparseBrainVolume to array

conversion from SparseBrainVolume to numeric

conversion from BrainVolume to LogicalBrainVolume

conversion from DenseBrainVolume to LogicalBrainVolume

conversion from ClusteredBrainVolume to LogicalBrainVolume

conversion from BrainVolume to array

---

as.array,BrainData-method
convert BrainData *instance to array*

Description

convert BrainData instance to array

Usage

```
## S4 method for signature 'BrainData'
as.array(x)
```

Arguments

x	an R object.

as.list,SparseBrainVector-method *as.list*

Description

convert SparseBrainVector to list of DenseBrainVolumes

Usage

```
## S4 method for signature 'SparseBrainVector'
as.list(x)
```

Arguments

x	object to be coerced or tested.

as.logical,BrainVolume-method *as.logical*

Description

Convert BrainVolume to logical vector

Usage

```
## S4 method for signature 'BrainVolume'
as.logical(x)
```

Arguments

x	object to be coerced or tested.

as.mask *Convert to a LogicalBrainVolume*

Description

Convert to a LogicalBrainVolume

Usage

```
as.mask(x, indices)

## S4 method for signature 'BrainVolume,missing'
as.mask(x)

## S4 method for signature 'BrainVolume,numeric'
as.mask(x, indices)
```

Arguments

x	the object to binarize
indices	the indices to set to TRUE

as.matrix,BrainData-method *convert* BrainData *instance to matrix*

Description

convert BrainData instance to matrix

Usage

```
## S4 method for signature 'BrainData'
as.matrix(x)
```

Arguments

x	an R object.

as.matrix,DenseBrainVector-method

convert a DenseBrainVector to a matrix

Description

convert a DenseBrainVector to a matrix

convert SparseBrainVector to matrix

Usage

```
## S4 method for signature 'DenseBrainVector'
as.matrix(x)

## S4 method for signature 'SparseBrainVector'
as.matrix(x)
```

Arguments

x	an R object.

as.numeric,SparseBrainVolume-method

Convert SparseBrainVolume to numeric

Description

Convert SparseBrainVolume to numeric

Usage

```
## S4 method for signature 'SparseBrainVolume'
as.numeric(x)
```

Arguments

x	object to be coerced or tested.

as.raster,Layer-method *as.raster*

Description

as.raster

Usage

```
## S4 method for signature 'Layer'
as.raster(x, zpos, thresh = c(0, 0), axis = 3)
```

Arguments

x	the layer to convert
zpos	the z coordinate
thresh	the threshold range
axis	the axis index (1,2,3)

as.sparse *Convert to sparse representation*

Description

Convert to sparse representation

convert a DenseBrainVector to a SparseBrainVector

Usage

```
as.sparse(x, mask, ...)

## S4 method for signature 'DenseBrainVector,numeric'
as.sparse(x, mask)
```

Arguments

x	the object to sparsify
mask	the elements to retain
...	additional arguments

as.vector,BrainData,ANY-method *convert* BrainData *instance to vector*

Description

convert BrainData instance to vector

Usage

```
## S4 method for signature 'BrainData,ANY'
as.vector(x)
```

Arguments

x	an R object.

axes *Generic getter function to extract image axes*

Description

Generic getter function to extract image axes

axes

axes

Usage

```
axes(x)

## S4 method for signature 'BrainSpace'
axes(x)

## S4 method for signature 'BrainData'
axes(x)
```

Arguments

x	an object with a set of axes

`AxisSet-class` *Virtual base class representing an ordered set of named axes.*

Description

Virtual base class representing an ordered set of named axes.

Slots

`ndim` the number of axes (or dimensions)

`AxisSet1D-class` *A one-dimensional axis set*

Description

A one-dimensional axis set

Slots

`i` the first axis

`AxisSet2D-class` *A two-dimensional axis set*

Description

A two-dimensional axis set

Slots

`j` the first axis

`AxisSet3D-class` *A three-dimensional axis set*

Description

A three-dimensional axis set

Slots

`k` the third axis

AxisSet4D-class *A four-dimensional axis set*

Description

A four-dimensional axis set

Slots

`l` the fourth axis

AxisSet5D-class *A five-dimensional axis set*

Description

A five-dimensional axis set

Slots

`m` the fifth axis

axisToIndex *Generic function to convert 1-dimensional real axis coordinates along a single axis dimension to an 1D index along the same axis*

Description

Generic function to convert 1-dimensional real axis coordinates along a single axis dimension to an 1D index along the same axis

axisToIndex

Usage

```
axisToIndex(x, real, dimNum)

## S4 method for signature 'BrainSpace,numeric,numeric'
axisToIndex(x, real, dimNum)
```

Arguments

`x`	the object
`real`	the axis coordinates
`dimNum`	the dimension number of the axis (e.g. 1, 2, 3)

Value

a vector of axis indices

`BaseMetaInfo-class` *This is a base class to represent meta information*

Description

This is a base class to represent meta information

`BaseSource-class` *This is a base class to represent a data source*

Description

This is a base class to represent a data source

Slots

`metaInfo` meta information for the data source

`BinaryReader` *BinaryReader*

Description

Constructor for `BinaryReader` class

Usage

```
BinaryReader(input, byteOffset, dataType, bytesPerElement,
  endian = .Platform$endian)
```

Arguments

`input`	file name to read from or else a `connection` object
`byteOffset`	the number of bytes to skip at the start of input
`dataType`	R data type of binary elements
`bytesPerElement`	number of bytes in each data element (e.g. 4 or 8 for floating point numbers)
`endian`	endianness of binary input connection

BinaryReader-class *BinaryReader*

Description

This class supports reading of bulk binary data from a connection

Slots

`input` the binary input connection

`byteOffset` the number of bytes to skip at the start of input

`dataType` the dataType of the binary Elements

`bytesPerElement` number of bytes in each data element (e.g. 4 or 8 for floating point numbers)

`endian` endianness of binary input connection

BinaryWriter-class *BinaryWriter*

Description

This class supports writing of bulk binary data to a connection

Constructor for `BinaryWriter` class

Usage

```
BinaryWriter(output, byteOffset, dataType, bytesPerElement,
  endian = .Platform$endian)
```

Arguments

`output`	file name to write to or else a `connection` object
`byteOffset`	the number of bytes to skip at the start of output
`dataType`	R data type of binary elements
`bytesPerElement`	number of bytes in each data element (e.g. 4 or 8 for floating point numbers)
`endian`	endianness of binary output connection

Slots

`output` the binary output connection

`byteOffset` the number of bytes to skip at the start of input

`dataType` the dataType of the binary Elements

`bytesPerElement` number of bytes in each data element (e.g. 4 or 8 for floating point numbers)

`endian` endianness of binary output connection

---

bounds *Generic function to extract the spatial bounds (origin + dim * spacing) of an image param x the object*

Description

Generic function to extract the spatial bounds (origin + dim * spacing) of an image param x the object

bounds

bounds

Usage

```
bounds(x)

## S4 method for signature 'BrainSpace'
bounds(x)

## S4 method for signature 'BrainData'
bounds(x)
```

Arguments

x	the object with bounds property

BrainBucket-class *BrainBucket*

Description

a four-dimensional image that conists of a sequence of labeled image volumes backed by a list

Slots

`source` the data source for the bucket volumes

`labels` the names of the sub-volumes contained in the bucket

`data` a list of `BrainVolume` instances with names corresponding to volume labels

BrainBucketSource-class *BrainBucketSource*

Description

A class that is used to produce a BrainBucket instance

Constructor function for BrainBucketSource class

Usage

```
BrainBucketSource(fileName, pattern = NULL, indices = NULL)
```

Arguments

fileName	the name of the bucket file
pattern	optional regular expression used to filter the sub-volumes using associated labels
indices	optional set of sub-volume indices to load

Slots

sourceList a list of sources for the bucket sub-volumes

cache a cache used to store data in memory

BrainData-class *BrainData*

Description

Base class for brain image data

Slots

source an instance of class BaseSource to store the source of the data

space an instance of class BrainSpace to represent the geometry of the data space

BrainFileDescriptor-class

This class represents a neuroimaging file format

Description

This class represents a neuroimaging file format

Slots

`fileFormat` the name of the file format (e.g. NIfTI)

`headerEncoding` the file encoding of the header file (e.g. 'raw' for binary, 'gzip' for gz compressed')

`headerExtension` the file extension for the header file (e.g. 'nii' for NIfTI single files)

`dataEncoding` the file encoding for the data file

`dataExtension` the file extension for the data file (e.g. 'nii' for NIfTI single files)

BrainFileSource-class

Base class for representing a data source for images. The purpose of this class is to provide a layer in between low level IO and image loading functionality.

Description

Base class for representing a data source for images. The purpose of this class is to provide a layer in between low level IO and image loading functionality.

Slots

`metaInfo` meta information for the data source

BrainMetaInfo-class

This class contains meta information from an image

Description

This class contains meta information from an image

This class contains meta information from an image

Usage

```
BrainMetaInfo(Dim, spacing, origin = rep(0, length(spacing)),
  dataType = "FLOAT", label = "",
  spatialAxes = OrientationList3D$AXIAL_LPI, additionalAxes = NullAxis)
```

Arguments

Dim	image dimensions
spacing	voxel dimensions
origin	coordinate origin
dataType	the type of the data (e.g. " FLOAT")
label	name(s) of images
spatialAxes	image axes for spatial dimensions (x,y,z)
additionalAxes	axes for dimensions > 3 (e.g. time, color band, direction)

Value

an instance of class BrainMetaInfo

Slots

dataType the data type code, e.g. FLOAT

Dim image dimensions

spatialAxes image axes for spatial dimensions (x,y,z)

additionalAxes axes for dimensions > 3 (e.g. time, color band, direction)

spacing voxel dimensions

origin coordinate origin

label name(s) of images

BrainSlice *BrainSlice constructor*

Description

BrainSlice constructor

Usage

```
BrainSlice(data, space, indices = NULL)
```

Arguments

data	data vector or matrix
space	an instance of class BrainSpace
indices	linear indices corresponding to data elements

`BrainSlice-class` *BrainSlice*

Description

Two-dimensional brain image

`BrainSource-class` *Base class for representing a data source for images. The purpose of this class is to provide a layer in between low level IO and image loading functionality.*

Description

Base class for representing a data source for images. The purpose of this class is to provide a layer in between low level IO and image loading functionality.

Slots

`metaInfo` meta information for the data source

`BrainSpace` *Constructor function for* `BrainSpace` *class*

Description

Constructor function for `BrainSpace` class

Usage

```
BrainSpace(Dim, origin = NULL, spacing = NULL, axes = NULL,
  trans = NULL)
```

Arguments

`Dim`	a vector describing the dimensions of the spatial grid
`origin`	the coordinate origin of the image space
`spacing`	the real-valued voxel dimensions (usually in millimeters)
`axes`	the image axes ordering (default is based on the NIFTI standard, Left-Posterior-Inferior)
`trans`	a matrix representing the coordinate transformation associated with the image space (default is based on the NIFTI standard, Left-Posterior-Inferior)

Value

an instance of class BrainSpace

Note

one should rarely need to create a new BrainSpace instance, as it will almost always be created automatically using information stored in an image header. Also, If one already has an existing image object, its BrainSpace instance can be easily extracted with the space method.

Examples

```
bspace <- BrainSpace(c(64,64,64), origin=c(0,0,0), spacing=c(2,2,2))
print(bspace)
origin(bspace)
axes(bspace)
trans(bspace)
```

BrainSpace-class *BrainSpace*

Description

This class represents the geometry of a brain image

Slots

`Dim` the grid dimensions of the image

`origin` the coordinates of the spatial origin

`spacing` the dimensions (in mm) of the grid units (voxels)

`axes` the set of named spatial axes

`trans` an affine transformation matrix that moves from grid -> real world coordinates

`inverseTrans` an inverse matrix that moves from real world -> grid coordinates

---

BrainVector-class *Four-dimensional brain image*

Description

Four-dimensional brain image

constructor function for virtual class BrainVector

Usage

```
BrainVector(data, space, mask = NULL, source = NULL, label = "")
```

Arguments

data	the image data
space	a BrainSpace object
mask	an optional array of type logical
source	an optional BrainSource object
label	a label of type character

Value

a concrete instance of BrainVector class

BrainVectorSource-class

BrainVectorSource

Description

A class that is used to produce a BrainVector instance

Construct a BrainVectorSource object

Usage

```
BrainVectorSource(fileName, indices = NULL, mask = NULL)
```

Arguments

fileName	name of the 4-dimensional image file
indices	the subset of volume indices to load – if NULL then all volumes will be loaded
mask	the subset of voxels that will be loaded

Slots

indices the index vector of the volumes to be loaded

BrainVolume-class *Three-dimensional brain image*

Description

Three-dimensional brain image

Construct a BrainVolume instance, using default (dense) implementation

Usage

```
BrainVolume(data, space, source = NULL, label = "", indices = NULL)
```

Arguments

data	a three-dimensional array
space	an instance of class BrainSpace
source	an instance of class BrainSource
label	a character string to identify volume
indices	an 1D vector that gives the linear indices of the associated data vector

Value

a DenseBrainVolume instance

Examples

```
bspace <- BrainSpace(c(64,64,64), spacing=c(1,1,1))
dat <- array(rnorm(64*64*64), c(64,64,64))
bvol <- BrainVolume(dat,bspace, label="test")
print(bvol)
```

BrainVolumeSource-class

A class is used to produce a BrainVolume instance

Description

A class is used to produce a BrainVolume instance

Constructor for BrainVolumeSource

Usage

```
BrainVolumeSource(input, index = 1)
```

Arguments

input	the input file name
index	the image subvolume index

Slots

index the index of the volume to be read – must be of length 1.

---

close,BinaryReader-method *close*

Description

close

close

Usage

```
## S4 method for signature 'BinaryReader'
close(con)

## S4 method for signature 'BinaryWriter'
close(con)
```

Arguments

con	a connection.

---

clusterCenters *clusterCenters*

Description

clusterCenters

extract cluster centers in a ClusteredBrainVolume

Usage

```
clusterCenters(x, features, FUN)

## S4 method for signature 'ClusteredBrainVolume,matrix,missing'
clusterCenters(x, features)
```

Arguments

x	the object to extract cluster centers from
features	additional features
FUN	a user-supplied function

ClusteredBrainVolume-class *ClusteredBrainVolume*

Description

Three-dimensional brain image that is divided into N disjoint partitions

Construct a ClusteredBrainVolume instance

Usage

```
ClusteredBrainVolume(mask, clusters, labelMap = NULL, source = NULL,
  label = "")
```

Arguments

mask	an instance of class LogicalBrainVolume
clusters	a vector of clusters ids
labelMap	as list that maps from cluster id to a cluster label
source	an instance of class BrainSource
label	a character string

Value

ClusteredBrainVolume instance

---

concat *Concatenate two objects*

Description

Concatenate two objects

concatenate two BrainVolumes

concat

Usage

```
concat(x, y, ...)

## S4 method for signature 'BrainVector,BrainVolume'
concat(x, y, ...)

## S4 method for signature 'BrainVolume,BrainVector'
concat(x, y, ...)

## S4 method for signature 'BrainVector,BrainVector'
concat(x, y, ...)

## S4 method for signature 'DenseBrainVolume,DenseBrainVolume'
concat(x, y, ...)

## S4 method for signature 'SparseBrainVector,SparseBrainVector'
concat(x, y, ...)
```

Arguments

x	the first object
y	the second object
...	additional objects

Note

dimensions of x and y must be equal

---

connComp *Find connected components*

Description

Find connected components

find connected components in BrainVolume

Usage

```
connComp(x, ...)

## S4 method for signature 'BrainVolume'
connComp(x, threshold = 0, clusterTable = TRUE,
  localMaxima = TRUE, localMaximaDistance = 15)
```

Arguments

x	the image object
...	additonal arguments
threshold	threshold defining lower intensity bound for image mask
clusterTable	return clusterTable
localMaxima	return table of local maxima
localMaximaDistance	the distance used to define minum distance between local maxima

---

connComp3D *Extract connected components from a 3D mask*

Description

Extract connected components from a 3D mask

Usage

```
connComp3D(mask)
```

Arguments

mask	a 3D binary array

Value

a two-element list of the connected components (cluster index and cluster sizes)

---

coords *Extract coordinates*

Description

Extract coordinates

coords

coords

Usage

```
coords(x, ...)

## S4 method for signature 'ROIVolume'
coords(x)

## S4 method for signature 'IndexLookupVolume'
coords(x, i)

## S4 method for signature 'SparseBrainVector'
coords(x, i)
```

Arguments

x	the object to extract coordinates from
...	additional arguments
i	the index in to the lookup volume

---

coordToGrid *Generic function to convert N-dimensional real world coordinates to grid coordinates*

Description

Generic function to convert N-dimensional real world coordinates to grid coordinates

coordToGrid

Usage

```
coordToGrid(x, coords)

## S4 method for signature 'BrainSpace,matrix'
coordToGrid(x, coords)

## S4 method for signature 'BrainVolume,matrix'
coordToGrid(x, coords)
```

Arguments

x	the object
coords	a matrix of real world coordinates

Value

a matrix of grid coordinates

`coordToIndex` *Generic function to convert N-dimensional real world coordinates to 1D indices*

Description

Generic function to convert N-dimensional real world coordinates to 1D indices

coordToIndex

Usage

```
coordToIndex(x, coords)

## S4 method for signature 'BrainSpace,matrix'
coordToIndex(x, coords)

## S4 method for signature 'BrainVolume,matrix'
coordToIndex(x, coords)
```

Arguments

`x`	the object
`coords`	a matrix of real world coordinates

Value

a vector of indices

`dataFile` *Generic function to get the name of the data file, given a file name and a* `BrainFileDescriptor` *instance.*

Description

Generic function to get the name of the data file, given a file name and a `BrainFileDescriptor` instance.

Usage

```
dataFile(x, fileName)

## S4 method for signature 'BrainFileDescriptor,character'
dataFile(x, fileName)
```

Arguments

x	descriptor instance
fileName	file name to be stripped of its extension

Value

the correct header name

---

dataFileMatches *Generic function to test whether a file name conforms to the given a BrainFileDescriptor instance. Will test for match to data file only*

Description

Generic function to test whether a file name conforms to the given a BrainFileDescriptor instance. Will test for match to data file only

Usage

```
dataFileMatches(x, fileName)

## S4 method for signature 'BrainFileDescriptor,character'
dataFileMatches(x, fileName)
```

Arguments

x	object for which the file name is to matched to
fileName	file name to be matched

Value

TRUE for match, FALSE otherwise

---

dataReader *Generic function to create data reader*

Description

Generic function to create data reader

Usage

```
dataReader(x, offset)

## S4 method for signature 'NIfTIMetaInfo'
dataReader(x, offset = 0)

## S4 method for signature 'AFNIMetaInfo'
dataReader(x, offset = 0)
```

Arguments

x	an object specifying the infromation required to produce the reader
offset	the byte offset (number of bytes to skip before reading)

DenseBrainVector-class *DenseBrainVector*

Description

Four-dimensional brain image, backed by an array

constructor function for class DenseBrainVector

Usage

```
DenseBrainVector(data, space, source = NULL, label = "")
```

Arguments

data	a 4-dimensonal array
space	a BrainSpace object
source	an optional BrainSource object
label	a label of type character

Value

DenseBrainVector instance

DenseBrainVolume-class *DenseBrainVolume*

Description

Three-dimensional brain image, backed by an `array`

Construct a `DenseBrainVolume` instance

Usage

```
DenseBrainVolume(data, space, source = NULL, label = "", indices = NULL)
```

Arguments

`data`	a three-dimensional `array`
`space`	an instance of class `BrainSpace`
`source`	an instance of class `BrainSource`
`label`	a `character` string
`indices`	an optional 1-d index vector

Value

`DenseBrainVolume` instance

dim,BrainData-method *dim*

Description

dim

dim

Usage

```
## S4 method for signature 'BrainData'
dim(x)

## S4 method for signature 'FileMetaInfo'
dim(x)

## S4 method for signature 'BrainSpace'
dim(x)
```

Arguments

x	an R object, for example a matrix, array or data frame.

---

dropDim *Generic function to drop a dimension from an object*

Description

Generic function to drop a dimension from an object

dropDim

dropDim

dropDim

dropDim

dropDim

Usage

```
dropDim(x, dimnum)

## S4 method for signature 'AxisSet2D,numeric'
dropDim(x, dimnum)

## S4 method for signature 'AxisSet2D,missing'
dropDim(x)

## S4 method for signature 'AxisSet3D,numeric'
dropDim(x, dimnum)

## S4 method for signature 'AxisSet3D,missing'
dropDim(x)

## S4 method for signature 'BrainSpace,missing'
dropDim(x)
```

Arguments

x	a dimensioned object
dimnum	the index of the dimension to drop

eachSeries *Generic functions to apply a function to each series of a 4D image That is, if the 4th dimension is 'time' each series is a 1D time series.*

Description

Generic functions to apply a function to each series of a 4D image That is, if the 4th dimension is 'time' each series is a 1D time series.

Usage

```
eachSeries(x, FUN, withIndex, ...)

## S4 method for signature 'BrainVector,`function`,missing'
eachSeries(x, FUN,
  withIndex = FALSE, ...)

## S4 method for signature 'SparseBrainVector,`function`,logical'
eachSeries(x, FUN,
  withIndex = FALSE, ...)
```

Arguments

x	a four dimensional image
FUN	a function taking one or two arguments (depending on the value of withIndex
withIndex	whether the index of the series is supplied as the second argument to the function
...	additional arguments

eachSlice *Generic functions to apply a function to each (2D) slice of an image*

Description

Generic functions to apply a function to each (2D) slice of an image

Usage

```
eachSlice(x, FUN, withIndex, ...)

## S4 method for signature 'BrainVolume,`function`,missing'
eachSlice(x, FUN)

## S4 method for signature 'BrainVolume,`function`,logical'
eachSlice(x, FUN, withIndex)
```

Arguments

x	the object
FUN	a function taking one or two arguments (depending on the value of withIndex
withIndex	whether the index of the slice is supplied as the second argument to the function
...	additional arguments

---

eachVolume *Generic function to apply a function to each volume of a four-dimensional image*

Description

Generic function to apply a function to each volume of a four-dimensional image

eachVolume

eachVolume

eachVolume

Usage

```
eachVolume(x, FUN, withIndex, ...)

## S4 method for signature 'BrainVector,`function`,missing'
eachVolume(x, FUN, withIndex, ...)

## S4 method for signature 'BrainBucket,`function`,missing'
eachVolume(x, FUN, withIndex, ...)

## S4 method for signature 'BrainBucket,`function`,logical'
eachVolume(x, FUN, withIndex, ...)

## S4 method for signature 'BrainVector,`function`,logical'
eachVolume(x, FUN, withIndex, ...)

## S4 method for signature 'SparseBrainVector,`function`,logical'
eachVolume(x, FUN,
  withIndex = FALSE, ...)

## S4 method for signature 'SparseBrainVector,`function`,missing'
eachVolume(x, FUN, withIndex,
  ...)
```

Arguments

x	four-dimensional image
FUN	a function taking one or two arguments (depending on the value of withIndex
withIndex	whether the index of the volume supplied as the second argument to the function
...	additional arguments

---

fileMatches *Generic function to test whether a file name conforms to the given BrainFileDescriptor instance. Will test for match to either header file or data file*

Description

Generic function to test whether a file name conforms to the given BrainFileDescriptor instance. Will test for match to either header file or data file

Usage

```
fileMatches(x, fileName)

## S4 method for signature 'BrainFileDescriptor,character'
fileMatches(x, fileName)
```

Arguments

x	object for which the file name is to matched to
fileName	file name to be matched

Value

TRUE for match, FALSE otherwise

FileMetaInfo-class *This class contains meta information from an image data file*

Description

This class contains meta information from an image data file

This class contains meta information for a NIfTI image file

This class contains meta information for a AFNI image file

Slots

`headerFile` name of the file containing meta information

`dataFile` name of the file containing data

`fileDescriptor` descriptor of image file format

`endian` byte order of data ('little' or 'big')

`dataOffset` the number of bytes preceding the start of image data in data file

`bytesPerElement` number of bytes per element

`intercept` constant value added to image – multiple values allowed (must equal numer of sub-images)

`slope` image multiplier – multiple values allowed (must equal numer of sub-images)

`header` a list of format specific attributes

`nifti_header` a list of attributes specific to the NIfTI file format

`afni_header` a list of attributes specific to the AFNI file format

---

fill *Generic function to map values from one set to another using a user-supplied lookup table*

Description

Generic function to map values from one set to another using a user-supplied lookup table

Usage

```
fill(x, lookup)

## S4 method for signature 'BrainVolume,matrix'
fill(x, lookup)
```

Arguments

`x`	the object to map values from
`lookup`	the lookup table

Value

a new object where the original values have been filled in with the values in the lookup table

`gridToIndex` *Generic function to convert N-dimensional grid coordinate to 1D indices*

Description

Generic function to convert N-dimensional grid coordinate to 1D indices

gridToIndex

gridToIndex

gridToIndex

Usage

```
gridToIndex(x, coords)

## S4 method for signature 'BrainSlice,matrix'
gridToIndex(x, coords)

## S4 method for signature 'BrainSpace,matrix'
gridToIndex(x, coords)

## S4 method for signature 'BrainSpace,numeric'
gridToIndex(x, coords)

## S4 method for signature 'BrainVolume,matrix'
gridToIndex(x, coords)

## S4 method for signature 'BrainVolume,numeric'
gridToIndex(x, coords)
```

Arguments

`x`	the object
`coords`	a matrix where each row is a corodinate or a vector of length N

Value

a vector of indices

`headerFile` *Generic function to get the name of the header file, given a file name and a* `BrainFileDescriptor` *instance.*

Description

Generic function to get the name of the header file, given a file name and a `BrainFileDescriptor` instance.

Usage

```
headerFile(x, fileName)

## S4 method for signature 'BrainFileDescriptor,character'
headerFile(x, fileName)
```

Arguments

`x`	descriptor instance
`fileName`	file name to be stripped of its extension

Value

the correct header name

`headerFileMatches` *Generic function to test whether a file name conforms to the given* `BrainFileDescriptor` *instance. Will test for match to header file only*

Description

Generic function to test whether a file name conforms to the given `BrainFileDescriptor` instance. Will test for match to header file only

Usage

```
headerFileMatches(x, fileName)

## S4 method for signature 'BrainFileDescriptor,character'
headerFileMatches(x, fileName)
```

Arguments

`x`	object for which the file name is to matched to
`fileName`	file name to be matched

Value

TRUE for match, FALSE otherwise

`image,BrainVolume-method`
image

Description

image

image

image

Usage

```
## S4 method for signature 'BrainVolume'
image(x, slice, col = heat.colors(128, alpha = 1),
  zero.col = "#00000000")

## S4 method for signature 'Overlay'
image(x, zpos, axis = 3)

## S4 method for signature 'Layer'
image(x, zpos, axis = 3)
```

Arguments

`slice`	the voxel index of the slice to display
`col`	a color map
`zero.col`	the color to use when the value is 0 (e.g background color)
`zpos`	the z coordinate
`axis`	the axis index
`x`	locations of grid lines at which the values in `z` are measured. These must be finite, non-missing and in (strictly) ascending order. By default, equally spaced values from 0 to 1 are used. If `x` is a `list`, its components `x$x` and `x$y` are used for `x` and `y`, respectively. If the list has component `z` this is used for `z`.

IndexLookupVolume-class *IndexLookupVolume*

Description

Three-dimensional brain image that can be used as a map between 1D grid indices and a table of values Currently used in the SparseBrainVector class.

IndexLookupVolume

Usage

```
IndexLookupVolume(space, indices)
```

Arguments

space	a BrainSpace object
indices	the set of 1-d indices defining the lookup map

indexToCoord *Generic function to convert 1D indices to N-dimensional real world coordinates*

Description

Generic function to convert 1D indices to N-dimensional real world coordinates

indexToCoord

Usage

```
indexToCoord(x, idx)

## S4 method for signature 'BrainSpace,index'
indexToCoord(x, idx)

## S4 method for signature 'BrainVolume,index'
indexToCoord(x, idx)
```

Arguments

x	the object
idx	the 1D indices

Value

a matrix of real coordinates

indexToGrid *Generic function to convert 1D indices to N-dimensional grid coordinates*

Description

Generic function to convert 1D indices to N-dimensional grid coordinates

indexToGrid

indexToGrid

Usage

```
indexToGrid(x, idx)

## S4 method for signature 'BrainSlice,index'
indexToGrid(x, idx)

## S4 method for signature 'BrainSpace,index'
indexToGrid(x, idx)

## S4 method for signature 'BrainVector,index'
indexToGrid(x, idx)

## S4 method for signature 'BrainVolume,index'
indexToGrid(x, idx)
```

Arguments

x	the object
idx	the 1D indices

Value

a matrix of grid coordinates

indices *Extract indices*

Description

Extract indices

indices

indices

indices

Usage

```
indices(x)

## S4 method for signature 'ROIVolume'
indices(x)

## S4 method for signature 'IndexLookupVolume'
indices(x)

## S4 method for signature 'SparseBrainVector'
indices(x)
```

Arguments

x	the object to extract indices

`inverseTrans` *Generic getter to extract inverse image coordinate transformation*

Description

Generic getter to extract inverse image coordinate transformation

inverseTrans

inverseTrans

Usage

```
inverseTrans(x)

## S4 method for signature 'BrainSpace'
inverseTrans(x)

## S4 method for signature 'BrainData'
inverseTrans(x)
```

Arguments

x	an object

Kernel *Create a Kernel object*

Description

Create a Kernel object

Usage

```
Kernel(kerndim, vdim, FUN = dnorm, ...)
```

Arguments

kerndim	the dimensions in voxels of the kernel
vdim	the dimensions of the voxels in real units
FUN	the kernel function taking as its first argument representing the distance from the center of the kernel
...	additional parameters to the kernel FUN

Kernel-class *Kernel*

Description

A class representing an image kernel

Slots

`width` the width in voxels of the kernel

`weights` the kernel weights

`voxels` the relative voxel coordinates of the kernel

`coords` the relative real coordinates of the kernel

Layer *create a Layer object*

Description

create a Layer object

Usage

```
Layer(vol, colorMap = gray((0:255)/255, alpha = 1), thresh = c(0, 0))
```

Arguments

vol	an image volume
colorMap	a lookup table defining mapping from image intensity values to colors
thresh	a range (min,max) defining the threshold window for determining image opacity

Value

an object of class Layer

Layer-class *Layer*

Description

A class used for displaying 2D images with color maps

Slots

vol the BrainVolume that provides the data for the layer.

colorMap a character vector of colors in hexadecimal rgb format. Can be generated by calls to rainbow, heat.colors, topo.colors, terrain.colors or similar functions.

thresh cut-off value above which vlaues will be made transparent.

`length,BrainVector-method`

Get length of `BrainVector`. *This is the numbe rof volumes in the volume vector (e.g. the 4th image dimension)*

Description

Get length of `BrainVector`. This is the numbe rof volumes in the volume vector (e.g. the 4th image dimension)

Usage

```
## S4 method for signature 'BrainVector'
length(x)
```

Arguments

x	an R object. For replacement, a vector or factor.

`length,ROIVolume-method`

length

Description

length

Usage

```
## S4 method for signature 'ROIVolume'
length(x)
```

Arguments

x	an R object. For replacement, a vector or factor.

loadBucket *loadBucket*

Description

load a BrainBucket object from file

Usage

```
loadBucket(fileName, pattern = NULL, indices = NULL)
```

Arguments

fileName	the name of the file to load
pattern	optional regular expression used to filter the sub-volumes using associated labels
indices	optional set of sub-volume indices to load

loadData *Generic function to load data from a data source*

Description

Generic function to load data from a data source

Load data from a BrainVectorSource

Load data from a BrainBucketSource

load a BrainVolume

loadData

Usage

```
loadData(x, ...)

## S4 method for signature 'BrainVectorSource'
loadData(x, mmap = FALSE)

## S4 method for signature 'BrainBucketSource'
loadData(x, key)

## S4 method for signature 'BrainVolumeSource'
loadData(x)

## S4 method for signature 'SparseBrainVectorSource'
loadData(x)
```

Arguments

x	a data source
...	additional arguments
mmap	use memory-mapped file
key	the name or index of the bucket to load

Value

an instance of class BrainVector

an instance of class BrainVolume

---

loadVector *loadVector*

Description

load an image volume from a file

Usage

```
loadVector(fileName, indices = NULL, mask = NULL)
```

Arguments

fileName	the name of the file to load
indices	the indices of the sub-volumes to load (e.g. if the file is 4-dimensional)
mask	a mask defining the spatial elements to load

Value

an BrainVector object

loadVolume *load an image volume from a file*

Description

load an image volume from a file

Usage

```
loadVolume(fileName, index = 1)
```

Arguments

fileName	the name of the file to load
index	the index of the volume (e.g. if the file is 4-dimensional)

Value

an instance of the class BrainVolume

Examples

```
fname <- system.file("extdata", "global_mask.nii", package="neuroim")
x <- loadVolume(fname)
print(dim(x))
space(x)
```

loadVolumeList *loadVolList*

Description

load a list of image volumes and return a BrainVector instance

Usage

```
loadVolumeList(fileNames, mask = NULL)
```

Arguments

fileNames	a list of files to load
mask	an optional mask indicating subset of voxels to load

Value

an instance of class BrainVector

LogicalBrainVolume-class *LogicalBrainVolume*

Description

Three-dimensional brain image where all values are either TRUE or FALSE

Construct a LogicalBrainVolume instance

Usage

```
LogicalBrainVolume(data, space, source = NULL, label = "", indices = NULL)
```

Arguments

data	a three-dimensional array
space	an instance of class BrainSpace
source	an instance of class BrainSource
label	a character string
indices	an optional 1-d index vector

Value

LogicalBrainVolume instance

lookup *Index Lookup operation*

Description

Index Lookup operation

lookup

lookup

Usage

```
lookup(x, i, ...)

## S4 method for signature 'IndexLookupVolume,numeric'
lookup(x, i)

## S4 method for signature 'SparseBrainVector,numeric'
lookup(x, i)
```

Arguments

x	the object to query
i	the index to lookup
...	additional arguments

---

makeVector *makeVector*

Description

Construct a BrainVector instance, using default (dense) implementation

Usage

```
makeVector(data, refdata, source = NULL, label = "")
```

Arguments

data	a four-dimensional array
refdata	an instance of class BrainVector or BrainVolume containing the reference space for the new vector.
label	a character string
source	an instance of class BrainSource

Value

DenseBrainVector instance

---

makeVolume *makeVolume*

Description

Construct a BrainVolume instance, using default (dense) implementation

Usage

```
makeVolume(data = NULL, refvol, source = NULL, label = "",
  indices = NULL)
```

Arguments

data	a three-dimensional array
refvol	an instance of class BrainVolume containing the reference space for the new volume.
label	a character string
source	an instance of class BrainSource
indices	an optional 1-d index vector

Value

DenseBrainVolume instance

---

map *Generic function to apply a function to an object*

Description

Generic function to apply a function to an object

apply a kernel function to a BrainVolume

Usage

```
map(x, m, ...)

## S4 method for signature 'BrainVolume,Kernel'
map(x, m, mask = NULL)
```

Arguments

x	the object that is mapped
m	the mapping object
...	additional arguments
mask	restrict application of kernel to maksed area

matchAnatomy2D *given two named axes return AxisSet2D singleton*

Description

given two named axes return AxisSet2D singleton

Usage

```
matchAnatomy2D(axis1, axis2)
```

Arguments

axis1	the first axis
axis2	the second axis

matchAnatomy3D *given three named axes return AxisSet3D singleton*

Description

given three named axes return AxisSet3D singleton

Usage

```
matchAnatomy3D(axis1, axis2, axis3)
```

Arguments

axis1	the first axis
axis2	the second axis
axis3	the thrid axis

`mergePartitions` *mergePartitions*

Description

mergePartitions

merge partititons in a ClusteredBrainVolume

Usage

```
mergePartitions(x, K, features, ...)

## S4 method for signature 'ClusteredBrainVolume,numeric,matrix'
mergePartitions(x, K, features)
```

Arguments

`x`	the object to merge
`K`	the number of merged partitions
`features`	the features used to define the partition
`...`	additional arguments

`NamedAxis-class` *This class represents an axis with a name attribute*

Description

This class represents an axis with a name attribute

Slots

`axis` the name of the axis

`direction` of axis (-1,+1)

names,BrainBucketSource-method *extract names of* BrainBucketSource *instance*

Description

extract names of BrainBucketSource instance

extract names of BrainBucket instance

Usage

```
## S4 method for signature 'BrainBucketSource'
names(x)

## S4 method for signature 'BrainBucket'
names(x)
```

Arguments

x	an R object.

ndim *Generic function to extract the number of dimensions of an object*

Description

Generic function to extract the number of dimensions of an object

ndim

ndim

Usage

```
ndim(x, ...)

## S4 method for signature 'AxisSet'
ndim(x)

## S4 method for signature 'BrainData'
ndim(x)

## S4 method for signature 'BrainSpace'
ndim(x)
```

Arguments

x	n-dimensional object
...	additional arguments

NIfTIFileDescriptor-class *This class supports the NIfTI file format*

Description

This class supports the NIfTI file format

NIfTIMetaInfo *Constructor for* NIfTIMetaInfo *class*

Description

Constructor for NIfTIMetaInfo class

Usage

```
NIfTIMetaInfo(descriptor, nifti_header)
```

Arguments

descriptor	an instance of class NIfTIFileDescriptor
nifti_header	a list returned by readNIftiHeader

Value

an instance of class NIfTIMetaInfo

NullMetaInfo-class *This is class is used to denote the absense of meta information*

Description

This is class is used to denote the absense of meta information

numClusters *numClusters*

Description

numClusters

get number of clusters in a ClusteredBrainVolume

Usage

```
numClusters(x)

## S4 method for signature 'ClusteredBrainVolume'
numClusters(x)
```

Arguments

x	the object to extract number of clusters

origin *Generic getter to extract image origin*

Description

Generic getter to extract image origin

origin

origin

Usage

```
origin(x)

## S4 method for signature 'BrainSpace'
origin(x)

## S4 method for signature 'BrainData'
origin(x)
```

Arguments

x	an object with an origin

overlay *overlay two objects*

Description

overlay two objects

overlay

Usage

```
overlay(x, y, ...)

## S4 method for signature 'Layer,Layer'
overlay(x, y)
```

Arguments

x	the underlay object
y	the overlay object
...	additional arguments for class-specific implementations

partition *partition*

Description

partition

partition a ClusteredBrainVolume into K spatial disjoint components for every existing partition in the volume

Usage

```
partition(x, K, features, ...)

## S4 method for signature 'ClusteredBrainVolume,numeric,matrix'
partition(x, K, features,
  method = "kmeans")
```

Arguments

x	the object to partition
K	the number of partitions
features	the features used to define the partition
...	additional arguments
method	clustering method

permMat *extract permutation matrix*

Description

extract permutation matrix

permMat

permMat

Usage

```
permMat(x, ...)

## S4 method for signature 'AxisSet2D'
permMat(x)

## S4 method for signature 'AxisSet3D'
permMat(x)
```

Arguments

x	the object
...	additional arguments

pick *pick*

Description

pick

Usage

```
pick(x, mask, ...)
```

Arguments

x	the object to pick from
mask	a mask object
...	addiitonal arguments

print,NamedAxis-method *print a* NamedAxis *instance*

Description

print a NamedAxis instance

print a AxisSet2D instance

print a AxisSet3D instance

Usage

```
## S4 method for signature 'NamedAxis'
print(x, ...)

## S4 method for signature 'AxisSet2D'
print(x, ...)

## S4 method for signature 'AxisSet3D'
print(x, ...)
```

Arguments

x	an object used to select a method.
...	further arguments passed to or from other methods.

RandomSearchlight *Create an Random Searchlight iterator*

Description

Create an Random Searchlight iterator

Usage

```
RandomSearchlight(mask, radius)
```

Arguments

mask	an image volume containing valid central voxels for roving searchlight
radius	in mm of spherical searchlight

readElements *Generic function to read a sequence of elements from an input source*

Description

Generic function to read a sequence of elements from an input source

readElements

Usage

```
readElements(x, numElements)

## S4 method for signature 'BinaryReader,numeric'
readElements(x, numElements)
```

Arguments

x	the input channel
numElements	the number of elements to read

Value

the elements as a vector

readHeader *read header information of an image file*

Description

read header information of an image file

Usage

```
readHeader(fileName)
```

Arguments

fileName	the name of the file to read

Value

an instance of class FileMetaInfo

readMetaInfo *Generic function to read image meta info given a file and a BrainFileDescriptor instance.*

Description

Generic function to read image meta info given a file and a BrainFileDescriptor instance.

Usage

```
readMetaInfo(x, fileName)

## S4 method for signature 'NIfTIFileDescriptor'
readMetaInfo(x, fileName)

## S4 method for signature 'AFNIFileDescriptor'
readMetaInfo(x, fileName)
```

Arguments

x	descriptor instance
fileName	file name contianing meta information

RegionCube *Create A Cuboid Region of Interest*

Description

Create A Cuboid Region of Interest

Usage

```
RegionCube(bvol, centroid, surround, fill = NULL, nonzero = TRUE)
```

Arguments

bvol	an image volume
centroid	the center of the cube in voxel space
surround	the number of voxels on either side of the central voxel
fill	optional value(s) to assign to data slot
nonzero	keep only nonzero elements from bvol

Value

an instance of class ROIVolume

RegionSphere *Create A Spherical Region of Interest*

Description

Create A Spherical Region of Interest

Usage

```
RegionSphere(bvol, centroid, radius, fill = NULL, nonzero = TRUE)
```

Arguments

bvol	an image volume
centroid	the center of the sphere in voxel space
radius	the radius in real units (e.g. millimeters) of the spherical ROI
fill	optional value(s) to assign to data slot
nonzero	keep only nonzero elements from bvol

Value

an instance of class ROIVolume

ROIVolume *Create an instance of class ROIVolume*

Description

Create an instance of class ROIVolume

Usage

```
ROIVolume(vspace, coords, data = rep(length(indices), 1))
```

Arguments

vspace	the volume BrainSpace
coords	matrix of voxel coordinates
data	the data values

Value

an instance of class ROIVolume

ROIVolume-class *ROIVolume*

Description

A class that is used to produce a SparseBrainVector instance

Slots

`data` the data stored in the ROI

`coords` the coordinates of the ROI

Searchlight *Create an exhaustive searchlight iterator*

Description

Create an exhaustive searchlight iterator

Usage

```
Searchlight(mask, radius)
```

Arguments

`mask`	an image volume containing valid central voxels for roving searchlight
`radius`	in mm of spherical searchlight

series *Extract vector series from object*

Description

Extract vector series from object

series

series

Usage

```
series(x, i, ...)

## S4 method for signature 'BrainVector,matrix'
series(x, i)

## S4 method for signature 'BrainVector,numeric'
series(x, i, j, k)

## S4 method for signature 'SparseBrainVector,matrix'
series(x, i)

## S4 method for signature 'SparseBrainVector,numeric'
series(x, i, j, k)
```

Arguments

x	the object
i	the series index
...	additional arguments
j	index of second dimension
k	index of third dimension

---

seriesIter *Construct a series iterator*

Description

Construct a series iterator

Usage

```
seriesIter(x)

## S4 method for signature 'BrainVector'
seriesIter(x)

## S4 method for signature 'SparseBrainVector'
seriesIter(x)
```

Arguments

x	the object to be iterated

show,BrainVolume-method *show*

Description

show

Usage

```
## S4 method for signature 'BrainVolume'
show(object)
```

Arguments

object	Any R object

slice *extract a 2D slice from an image volume*

Description

extract a 2D slice from an image volume

Usage

```
slice(x, zlevel, along, orientation, ...)

## S4 method for signature 'BrainVolume,numeric,numeric,character'
slice(x, zlevel, along,
  orientation)
```

Arguments

x	the object
zlevel	coordinate (in voxel units) along the sliced axis
along	the axis along which to slice
orientation	the target orientation of the 2D slice
...	additional arguments

space *Generic function to extract the* space *member variable*

Description

Generic function to extract the space member variable

space

space

Usage

```
space(x, ...)

## S4 method for signature 'BrainData'
space(x)

## S4 method for signature 'IndexLookupVolume'
space(x)
```

Arguments

x	the object to query
...	additional arguments

Value

an object representing the geometric space of the image

spacing *Generic function to extract the voxel dimensions of an image*

Description

Generic function to extract the voxel dimensions of an image

spacing

spacing

Usage

```
spacing(x)

## S4 method for signature 'BrainData'
spacing(x)

## S4 method for signature 'BrainSpace'
spacing(x)
```

Arguments

x	the object

Value

a numeric vector

`SparseBrainVector-class` *SparseBrainVector*

Description

a sparse four-dimensional brain image, backed by a `matrix`, where each column represents a vector spanning the fourth dimension (e.g. time)

constructs a SparseBrainVector object

Usage

```
SparseBrainVector(data, space, mask, source = NULL, label = "")
```

Arguments

`data`	an array which can be a `matrix` or 4-D `array`
`space`	a BrainSpace instance
`mask`	a 3D `array` of type `logical`
`source`	the data source – an instance of class `BrainSource`
`label`	associated sub-image labels

Slots

`mask` the mask defining the sparse domain

`data` the matrix of series, where rows span across voxel space and columns span the fourth dimensions

`map` instance of class `IndexLookupVolume` is used to map between spatial and index/row coordinates

SparseBrainVectorSource-class

SparseBrainVectorSource

Description

A class that is used to produce a SparseBrainVector instance

constructs a SparseBrainVectorSource object

Usage

```
SparseBrainVectorSource(metaInfo, indices, mask)
```

Arguments

metaInfo	an object of class BrainMetaInfo
indices	a vector of 1D indices
mask	a 3D array of type logical

Slots

mask the subset of voxels that will be stored in memory

SparseBrainVolume-class

SparseBrainVolume

Description

Three-dimensional brain image, backed by a sparseVector for Matrix package

Construct a SparseBrainVolume instance

Usage

```
SparseBrainVolume(data, space, source = NULL, label = "", indices = NULL)
```

Arguments

data	a numeric vector
space	an instance of class BrainSpace
source	an instance of class BrainSource
label	a character string
indices	a 1-d index vector

Value

DenseBrainVolume instance

---

splitFill *Generic function to fill disjoint sets of values with the output of a function*

Description

Generic function to fill disjoint sets of values with the output of a function

split values by factor apply function and then fill in new volume

Usage

```
splitFill(x, fac, FUN)

## S4 method for signature 'BrainVolume,factor,`function`'
splitFill(x, fac, FUN)
```

Arguments

x	the object to split
fac	the factor to split by
FUN	the function to summarize the the clusters

Value

a new object where the original values have been replaced by the function output

Note

FUN can return one value per category or one value per voxel

---

splitReduce *Generic function to summarize subsets of an object*

Description

Generic function to summarize subsets of an object

Usage

```
splitReduce(x, fac, FUN)

## S4 method for signature 'matrix,factor,`function`'
splitReduce(x, fac, FUN)
```

Arguments

x	a numeric matrix(like) object
fac	the factor to define subsets of the object
FUN	the function to apply to each subset

Value

a new matrix(like) object where the original values have been scaled

---

splitScale *Generic function to center/scale subsets of an object*

Description

Generic function to center/scale subsets of an object

Usage

```
splitScale(x, f, center, scale)

## S4 method for signature 'matrix,factor,logical,logical'
splitScale(x, f, center = TRUE,
  scale = TRUE)

## S4 method for signature 'matrix,factor,missing,missing'
splitScale(x, f)
```

Arguments

x	a numeric matrix(like) object
f	the conditioning expression (usually a factor)
center	should values be centered?
scale	should values be scaled?

Value

a new matrix(like) object where the original values have been grouped by a factor and then centered and/or scaled for each grouping

stripExtension *Generic function to strip extension from file name, given a* BrainFileDescriptor *instance.*

Description

Generic function to strip extension from file name, given a BrainFileDescriptor instance.

Usage

```
stripExtension(x, fileName)

## S4 method for signature 'BrainFileDescriptor,character'
stripExtension(x, fileName)
```

Arguments

x	descriptor instance
fileName	file name to be stripped of its extension

Value

fileName without extension

takeSeries *Generic function to extract a set of series from a 4D image*

Description

Generic function to extract a set of series from a 4D image

Usage

```
takeSeries(x, indices, ...)
```

Arguments

x	a four dimensional image
indices	the indices of the series' to extract
...	additional arguments

takeVolume *Generic function to extract a volume from a four-dimensional image*

Description

Generic function to extract a volume from a four-dimensional image

takeVolume extract volume from SparseBrainVector

Usage

```
takeVolume(x, i, ...)

## S4 method for signature 'BrainVector,numeric'
takeVolume(x, i, merge = FALSE)

## S4 method for signature 'SparseBrainVector,numeric'
takeVolume(x, i, merge = FALSE)
```

Arguments

x	four-dimensional image
i	the indices of the volume(s) to extract
...	additional arguments
merge	concatenate extracted volumes

tesselate *tesselate*

Description

tesselate

tesselate a LogicalBrainVolume into K spatial disjoint components

Usage

```
tesselate(x, K, ...)

## S4 method for signature 'LogicalBrainVolume,numeric'
tesselate(x, K, features = NULL,
  spatialWeight = 4)
```

Arguments

x	the object to tesselate
K	the number of partitions
...	extra arguments
features	use addiitonal feature set to tesselate volume
spatialWeight	weight voxels according to distance

---

trans *Generic getter to extract image coordinate transformation*

Description

Generic getter to extract image coordinate transformation

trans

trans

Usage

```
trans(x)

## S4 method for signature 'BrainMetaInfo'
trans(x)

## S4 method for signature 'NIfTIMetaInfo'
trans(x)

## S4 method for signature 'BrainSpace'
trans(x)

## S4 method for signature 'BrainData'
trans(x)
```

Arguments

x	an object with a transformation

voxels *extract voxel coordinates*

Description

extract voxel coordinates

extract voxels from a `Kernel` object

Usage

```
voxels(x, ...)

## S4 method for signature 'Kernel'
voxels(x, centerVoxel = NULL)
```

Arguments

`x`	the object to extract voxels from
`...`	addiitonal arguments to function
`centerVoxel`	the absolute location of the center of the voxel, default is (0,0,0)

writeElements *Generic function to write a sequence of elements from an input source*

Description

Generic function to write a sequence of elements from an input source

writeElements

Usage

```
writeElements(x, els)

## S4 method for signature 'BinaryWriter,numeric'
writeElements(x, els)
```

Arguments

`x`	the output channel
`els`	the elements to write

writeVector *Generic function to write an image vector to disk*

Description

Generic function to write an image vector to disk

writeVector

writeVector

writeVector

Usage

```
writeVector(x, fileName, format, dataType)

## S4 method for signature 'BrainVector,character,missing,missing'
writeVector(x, fileName)

## S4 method for signature 'BrainVector,character,character,missing'
writeVector(x, fileName,
  format)

## S4 method for signature 'BrainVector,character,missing,character'
writeVector(x, fileName,
  dataType)
```

Arguments

x	the image to write
fileName	the bane of the file to write
format	the file format
dataType	the numeric data type

writeVolume *Generic function to write an image volume to disk*

Description

Generic function to write an image volume to disk

Usage

```
writeVolume(x, fileName, format, dataType)

## S4 method for signature 'BrainVolume,character,missing,missing'
writeVolume(x, fileName)

## S4 method for signature 'ClusteredBrainVolume,character,missing,missing'
writeVolume(x,
  fileName)

## S4 method for signature 'BrainVolume,character,character,missing'
writeVolume(x, fileName,
  format)

## S4 method for signature 'BrainVolume,character,missing,character'
writeVolume(x, fileName,
  dataType)
```

Arguments

x	an image object
fileName	a file name
format	file format string
dataType	output data type

[,ROIVolume,numeric,missing,ANY-method
extract data from ROIVolume

Description

extract data from ROIVolume

Usage

```
## S4 method for signature 'ROIVolume,numeric,missing,ANY'
x[i, j, drop]
```

Arguments

j	index for second dimension (missing)
x	object from which to extract element(s) or in which to replace element(s).

i	indices specifying elements to extract or replace. Indices are numeric or character vectors or empty (missing) or NULL. Numeric values are coerced to integer as by as.integer (and hence truncated towards zero). Character vectors will be matched to the names of the object (or for matrices/arrays, the dimnames): see 'Character indices' below for further details. For [-indexing only: i, j, ... can be logical vectors, indicating elements/slices to select. Such vectors are recycled if necessary to match the corresponding extent. i, j, ... can also be negative integers, indicating elements/slices to leave out of the selection. When indexing arrays by [a single argument i can be a matrix with as many columns as there are dimensions of x; the result is then a vector with elements corresponding to the sets of indices in each row of i. An index value of NULL is treated as if it were integer(0).
drop	For matrices and arrays. If TRUE the result is coerced to the lowest possible dimension (see the examples). This only works for extracting elements, not for the replacement. See drop for further details.

`[,SparseBrainVector,missing,missing,ANY-method`

extract data from SparseBrainVector

Description

extract data from SparseBrainVector

Usage

```
## S4 method for signature 'SparseBrainVector,missing,missing,ANY'
x[i, j, k, m, ...,
  drop = TRUE]
```

Arguments

j	index for dimension 2
k	index for dimension 3
m	index for dimension 4
...	additional arguments
x	object from which to extract element(s) or in which to replace element(s).
i	indices specifying elements to extract or replace. Indices are numeric or character vectors or empty (missing) or NULL. Numeric values are coerced to integer as by as.integer (and hence truncated towards zero). Character vectors will be matched to the names of the object (or for matrices/arrays, the dimnames): see 'Character indices' below for further details. For [-indexing only: i, j, ... can be logical vectors, indicating elements/slices to select. Such vectors are recycled if necessary to match the corresponding

extent. `i`, `j`, `...` can also be negative integers, indicating elements/slices to leave out of the selection.

When indexing arrays by `[` a single argument `i` can be a matrix with as many columns as there are dimensions of `x`; the result is then a vector with elements corresponding to the sets of indices in each row of `i`.

An index value of `NULL` is treated as if it were `integer(0)`.

`drop` For matrices and arrays. If `TRUE` the result is coerced to the lowest possible dimension (see the examples). This only works for extracting elements, not for the replacement. See `drop` for further details.

`[,SparseBrainVector,missing,numeric,ANY-method`

extract data from SparseBrainVector

Description

extract data from SparseBrainVector

Usage

```
## S4 method for signature 'SparseBrainVector,missing,numeric,ANY'
x[i, j, k, m, ...,
  drop = TRUE]
```

Arguments

`j`	index for dimension 2
`k`	index for dimension 3
`m`	index for dimension 4
`...`	additional arguments
`x`	object from which to extract element(s) or in which to replace element(s).
`i`	indices specifying elements to extract or replace. Indices are `numeric` or `character` vectors or empty (missing) or `NULL`. Numeric values are coerced to integer as by `as.integer` (and hence truncated towards zero). Character vectors will be matched to the `names` of the object (or for matrices/arrays, the `dimnames`): see 'Character indices' below for further details. For `[`-indexing only: `i`, `j`, `...` can be logical vectors, indicating elements/slices to select. Such vectors are recycled if necessary to match the corresponding extent. `i`, `j`, `...` can also be negative integers, indicating elements/slices to leave out of the selection. When indexing arrays by `[` a single argument `i` can be a matrix with as many columns as there are dimensions of `x`; the result is then a vector with elements corresponding to the sets of indices in each row of `i`. An index value of `NULL` is treated as if it were `integer(0)`.

drop — For matrices and arrays. If TRUE the result is coerced to the lowest possible dimension (see the examples). This only works for extracting elements, not for the replacement. See drop for further details.

`[,SparseBrainVector,numeric,missing,ANY-method`

extract data from SparseBrainVector

Description

extract data from SparseBrainVector

Usage

```
## S4 method for signature 'SparseBrainVector,numeric,missing,ANY'
x[i, j, k, m, ...,
  drop = TRUE]
```

Arguments

j	index for dimension 2
k	index for dimension 3
m	index for dimension 4
...	additional arguments
x	object from which to extract element(s) or in which to replace element(s).
i	indices specifying elements to extract or replace. Indices are numeric or character vectors or empty (missing) or NULL. Numeric values are coerced to integer as by as.integer (and hence truncated towards zero). Character vectors will be matched to the names of the object (or for matrices/arrays, the dimnames): see 'Character indices' below for further details. For [-indexing only: i, j, ... can be logical vectors, indicating elements/slices to select. Such vectors are recycled if necessary to match the corresponding extent. i, j, ... can also be negative integers, indicating elements/slices to leave out of the selection. When indexing arrays by [a single argument i can be a matrix with as many columns as there are dimensions of x; the result is then a vector with elements corresponding to the sets of indices in each row of i. An index value of NULL is treated as if it were integer(0).
drop	For matrices and arrays. If TRUE the result is coerced to the lowest possible dimension (see the examples). This only works for extracting elements, not for the replacement. See drop for further details.

`[,SparseBrainVector,numeric,numeric,ANY-method`
extract data from SparseBrainVector

Description

extract data from SparseBrainVector

Usage

```
## S4 method for signature 'SparseBrainVector,numeric,numeric,ANY'
x[i, j, k, m, ...,
  drop = TRUE]
```

Arguments

`j`	index for dimension 2
`k`	index for dimension 3
`m`	index for dimension 4
`...`	additional arguments
`x`	object from which to extract element(s) or in which to replace element(s).
`i`	indices specifying elements to extract or replace. Indices are `numeric` or `character` vectors or empty (missing) or `NULL`. Numeric values are coerced to integer as by `as.integer` (and hence truncated towards zero). Character vectors will be matched to the `names` of the object (or for matrices/arrays, the `dimnames`): see 'Character indices' below for further details. For `[`-indexing only: `i`, `j`, `...` can be logical vectors, indicating elements/slices to select. Such vectors are recycled if necessary to match the corresponding extent. `i`, `j`, `...` can also be negative integers, indicating elements/slices to leave out of the selection. When indexing arrays by `[` a single argument `i` can be a matrix with as many columns as there are dimensions of `x`; the result is then a vector with elements corresponding to the sets of indices in each row of `i`. An index value of `NULL` is treated as if it were `integer(0)`.
`drop`	For matrices and arrays. If `TRUE` the result is coerced to the lowest possible dimension (see the examples). This only works for extracting elements, not for the replacement. See `drop` for further details.

`[[,BrainBucket,character,missing-method`
extract labeled volume from `BrainBucket`

Description

extract labeled volume from `BrainBucket`

Usage

```
## S4 method for signature 'BrainBucket,character,missing'
x[[i]]
```

Arguments

`x`	object from which to extract element(s) or in which to replace element(s).
`i`	indices specifying elements to extract or replace. Indices are `numeric` or `character` vectors or empty (missing) or `NULL`. Numeric values are coerced to integer as by `as.integer` (and hence truncated towards zero). Character vectors will be matched to the `names` of the object (or for matrices/arrays, the `dimnames`): see 'Character indices' below for further details. For `[`-indexing only: `i`, `j`, `...` can be logical vectors, indicating elements/slices to select. Such vectors are recycled if necessary to match the corresponding extent. `i`, `j`, `...` can also be negative integers, indicating elements/slices to leave out of the selection. When indexing arrays by `[` a single argument `i` can be a matrix with as many columns as there are dimensions of `x`; the result is then a vector with elements corresponding to the sets of indices in each row of `i`. An index value of `NULL` is treated as if it were `integer(0)`.

`[[,BrainBucket,numeric,missing-method`
extract indexed volume from `BrainBucket`

Description

extract indexed volume from `BrainBucket`

Usage

```
## S4 method for signature 'BrainBucket,numeric,missing'
x[[i]]
```

Arguments

x	object from which to extract element(s) or in which to replace element(s).
i	indices specifying elements to extract or replace. Indices are `numeric` or `character` vectors or empty (missing) or `NULL`. Numeric values are coerced to integer as by `as.integer` (and hence truncated towards zero). Character vectors will be matched to the `names` of the object (or for matrices/arrays, the `dimnames`): see 'Character indices' below for further details. For `[`-indexing only: `i`, `j`, `...` can be logical vectors, indicating elements/slices to select. Such vectors are recycled if necessary to match the corresponding extent. `i`, `j`, `...` can also be negative integers, indicating elements/slices to leave out of the selection. When indexing arrays by `[` a single argument `i` can be a matrix with as many columns as there are dimensions of `x`; the result is then a vector with elements corresponding to the sets of indices in each row of `i`. An index value of `NULL` is treated as if it were `integer(0)`.

Index