

# Package ‘neuroim’

July 2, 2015

**Imports** Matrix, yaImpute, Rcpp, stringr, hash, iterators, abind,  
assertthat, stats, grDevices

**VignetteBuilder** knitr

**LinkingTo** Rcpp

**License** GPL (>= 2)

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

**Title** Read, Write, and Process Brain Imaging Data

**Type** Package

**Author** Bradley R. Buchsbaum

**Description** 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 NIFTI files and limited support for reading AFNI files.

**Version** 0.0.4

**Date** 2015-06-23

**Depends** R (>= 3.0.0), grid, methods

**Suggests** foreach, testthat, knitr

**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'  
'RcppExports.R' 'conncomp.R' 'datadoc.R' 'neuroim.R'

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2015-07-02 16:07:14

## R topics documented:

addDim . . . . . 5

AFNIFileDescriptor-class . . . . .	6
AFNIMetaInfo . . . . .	6
as . . . . .	7
as.array,BrainData-method . . . . .	7
as.list,BrainVector-method . . . . .	8
as.logical,BrainVolume-method . . . . .	8
as.mask . . . . .	9
as.matrix,BrainData-method . . . . .	9
as.matrix,DenseBrainVector-method . . . . .	10
as.numeric,SparseBrainVolume-method . . . . .	10
as.raster,Layer-method . . . . .	11
as.sparse . . . . .	11
as.vector,BrainData,ANY-method . . . . .	12
axes . . . . .	13
AxisSet-class . . . . .	13
AxisSet1D-class . . . . .	13
AxisSet2D-class . . . . .	14
AxisSet3D-class . . . . .	14
AxisSet4D-class . . . . .	14
AxisSet5D-class . . . . .	14
axisToIndex . . . . .	15
Base-class . . . . .	15
BaseMetaInfo-class . . . . .	15
BaseSource-class . . . . .	16
BinaryReader . . . . .	16
BinaryReader-class . . . . .	16
BinaryWriter-class . . . . .	17
BootstrapSearchlight . . . . .	17
bounds . . . . .	18
BrainBucket-class . . . . .	19
BrainBucketSource-class . . . . .	19
BrainData-class . . . . .	20
BrainFileDescriptor-class . . . . .	20
BrainFileSource-class . . . . .	21
BrainMetaInfo-class . . . . .	21
BrainSlice . . . . .	22
BrainSlice-class . . . . .	22
BrainSource-class . . . . .	22
BrainSpace . . . . .	23
BrainSpace-class . . . . .	24
BrainVector-class . . . . .	24
BrainVectorSource . . . . .	25
BrainVectorSource-class . . . . .	25
BrainVolume . . . . .	26
BrainVolume-class . . . . .	26
BrainVolumeSource-class . . . . .	27
close,BinaryReader-method . . . . .	27
clusterCenters . . . . .	28

ClusteredBrainVolume-class . . . . .	28
concat . . . . .	29
connComp . . . . .	30
connComp3D . . . . .	31
coords . . . . .	31
coordToGrid . . . . .	32
coordToIndex . . . . .	32
dataFile . . . . .	33
dataFileMatches . . . . .	34
dataReader . . . . .	34
DenseBrainVector-class . . . . .	35
DenseBrainVolume-class . . . . .	35
dim,BrainData-method . . . . .	36
dim,BrainSpace-method . . . . .	36
dim,FileMetaInfo-method . . . . .	37
dropDim . . . . .	37
eachSeries . . . . .	38
eachSlice . . . . .	39
eachVolume . . . . .	39
fileMatches . . . . .	41
FileMetaInfo-class . . . . .	42
fill . . . . .	42
gridToCoord . . . . .	43
gridToIndex . . . . .	44
headerFile . . . . .	44
headerFileMatches . . . . .	45
image,BrainVolume-method . . . . .	46
imageGrid . . . . .	46
IndexLookupVolume-class . . . . .	47
indexToCoord . . . . .	47
indexToGrid . . . . .	48
indices . . . . .	49
inverseTrans . . . . .	50
Kernel . . . . .	50
Kernel-class . . . . .	51
Layer . . . . .	51
Layer-class . . . . .	52
length,ROIVolume-method . . . . .	52
loadBucket . . . . .	53
loadData . . . . .	53
loadVector . . . . .	54
loadVolume . . . . .	55
loadVolumeList . . . . .	55
LogicalBrainVolume-class . . . . .	56
lookup . . . . .	56
makeVector . . . . .	57
makeVolume . . . . .	57
map . . . . .	58

matchAnatomy2D . . . . .	59
matchAnatomy3D . . . . .	59
matrixToVolumeList . . . . .	60
mergePartitions . . . . .	60
MNI_SPACE_1MM . . . . .	61
NamedAxis-class . . . . .	61
names,BrainBucketSource-method . . . . .	62
ndim . . . . .	62
neuroim . . . . .	63
NIFTIFileDescriptor-class . . . . .	63
NIFTIMetaInfo . . . . .	63
NullMetaInfo-class . . . . .	64
numClusters . . . . .	64
origin . . . . .	64
overlay . . . . .	65
partition . . . . .	66
permMat . . . . .	66
pick . . . . .	67
print . . . . .	67
print,AxisSet2D-method . . . . .	68
print,AxisSet3D-method . . . . .	68
print,NamedAxis-method . . . . .	69
RandomSearchlight . . . . .	69
readAFNIHeader . . . . .	70
readElements . . . . .	70
readHeader . . . . .	71
readMetaInfo . . . . .	71
RegionCube . . . . .	72
RegionSphere . . . . .	72
RegionSquare . . . . .	73
ROIVolume . . . . .	74
ROIVolume-class . . . . .	74
scaleSeries . . . . .	75
Searchlight . . . . .	76
series . . . . .	76
seriesIter . . . . .	77
show,AxisSet1D-method . . . . .	78
show,AxisSet2D-method . . . . .	78
show,AxisSet3D-method . . . . .	78
show,AxisSet4D-method . . . . .	79
show,BaseMetaInfo-method . . . . .	79
show,BrainSpace-method . . . . .	80
show,BrainVector-method . . . . .	80
show,BrainVectorSource-method . . . . .	81
show,BrainVolume-method . . . . .	81
show,FileMetaInfo-method . . . . .	82
show,NamedAxis-method . . . . .	82
show,NullMetaInfo-method . . . . .	83

show,ROIVolume-method . . . . .	83
show,SparseBrainVector-method . . . . .	84
slice . . . . .	84
space . . . . .	85
spacing . . . . .	85
SparseBrainVector-class . . . . .	86
SparseBrainVectorSource-class . . . . .	87
SparseBrainVolume-class . . . . .	88
splitFill . . . . .	89
splitReduce . . . . .	90
splitScale . . . . .	91
stripExtension . . . . .	92
takeSeries . . . . .	93
takeVolume . . . . .	93
tessellate . . . . .	94
trans . . . . .	94
values . . . . .	95
voxels . . . . .	96
writeElements . . . . .	96
writeVector . . . . .	97
writeVolume . . . . .	98
[,BrainBucket,index,missing,ANY-method . . . . .	99
[,ROIVolume,numeric,missing,ANY-method . . . . .	99
[,SparseBrainVector,missing,missing,ANY-method . . . . .	100
[,SparseBrainVector,missing,numeric,ANY-method . . . . .	100
[,SparseBrainVector,numeric,missing,ANY-method . . . . .	101
[,SparseBrainVector,numeric,numeric,ANY-method . . . . .	102
[,SparseBrainVolume,matrix,missing,ANY-method . . . . .	102
[,SparseBrainVolume,missing,missing,ANY-method . . . . .	103
[,SparseBrainVolume,missing,numeric,ANY-method . . . . .	104
[,SparseBrainVolume,numeric,missing,missing-method . . . . .	104
[,SparseBrainVolume,numeric,numeric,ANY-method . . . . .	105
[[,BrainBucket,index,missing-method . . . . .	105

**Index****106**

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

**Examples**

```
x = BrainSpace(c(10,10,10), c(1,1,1))
x1 <- addDim(x, 10)
ndim(x1) == 4
dim(x1)[4] == 10
```

---

AFNIFileDescriptor-class

*AFNIFileDescriptor*

---

**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 <a href="#">AFNIFileDescriptor</a>
afni_header	a list returned by readAFNIHeader

**Value**

an instance of class [AFNIMetaInfo](#)

---

as *conversion from DenseBrainVolume to array*

---

### Description

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            the object

---

as.list,BrainVector-method

*convert a BrainVector to list of volumes.*

---

### Description

convert a BrainVector to list of volumes.

convert SparseBrainVector to list of [DenseBrainVolume](#)

### Usage

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

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

### Arguments

x                    the object

---

as.logical,BrainVolume-method

*as.logical*

---

### Description

Convert BrainVolume to linkS4class{LogicalBrainVolume}

### Usage

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

### Arguments

x                    the object

### Details

the image values will be converted to using R base function `as.logical` and wrapped in `LogicalBrainVolume`

### Value

an instance of linkS4class{LogicalBrainVolume}



---

as.mask	<i>Convert to a LogicalBrainVolume</i>
---------	--

---

**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	<i>convert BrainData instance to matrix</i>
----------------------------	---

---

**Description**

convert BrainData instance to matrix

**Usage**

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

**Arguments**

x	the 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                    the 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                    the object to convert

---

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 in coordinate space
thresh	the threshold range
axis	the axis index (1,2,3)

---

as.sparse                      *Convert to from dense to sparse representation*

---

**Description**

Convert to from dense to sparse representation

**Usage**

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

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

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

```
## S4 method for signature 'DenseBrainVolume,LogicalBrainVolume'
as.sparse(x, mask)
```

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

**Arguments**

x	the object to make sparse, e.g. DenseBrainVolume or DenseBrainVector
mask	the elements to retain
...	additional arguments

**Details**

mask can be an integer vector of ID indices or a mask volume of class LogicalBrainVolume

**Examples**

```
bvol <- BrainVolume(array(runif(24*24*24), c(24,24,24)), BrainSpace(c(24,24,24), c(1,1,1)))
indmask <- sort(sample(1:(24*24*24), 100))
svol <- as.sparse(bvol, indmask)
```

```
mask <- LogicalBrainVolume(runif(length(indmask)), space=space(bvol), indices=indmask)
sum(mask) == 100
```

---

```
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	the object
---	------------

---

axes	<i>Generic getter function to extract image axes</i>
------	--

---

**Description**

Generic getter function to extract image 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	<i>Base</i>
---------------	-------------

---

**Description**

Virtual base class representing an ordered set of named axes.

**Slots**

ndim the number of axes (or dimensions)

---

AxisSet1D-class	<i>AxisSet1D</i>
-----------------	------------------

---

**Description**

A one-dimensional axis set

**Slots**

i the first axis

---

AxisSet2D-class      *AxisSet2D*

---

**Description**

A two-dimensional axis set

**Slots**

j the second axis

---

AxisSet3D-class      *AxisSet3D*

---

**Description**

A three-dimensional axis set

**Slots**

k the third axis

---

AxisSet4D-class      *AxisSet4D*

---

**Description**

A four-dimensional axis set

**Slots**

l the fourth axis

---

AxisSet5D-class      *AxisSet5D*

---

**Description**

A five-dimensional axis set

**Slots**

m the fifth axis

---

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

---

**Description**

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

**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

---

Base-class	<i>Base</i>
------------	-------------

---

**Description**

Generic S4 Base class

---

BaseMetaInfo-class	<i>BaseMetaInfo</i>
--------------------	---------------------

---

**Description**

This is a base class to represent meta information

---

BaseSource-class      *BaseSource*

---

**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

---

BootstrapSearchlight	<i>Create a searchlight iterator that samples regions from within a mask. Searchlight centers are sampled *without* replacement, but the same voxel can belong to multiple searchlight samples. It is in the latter sense that this is a bootstrap resampling scheme.</i>
----------------------	---

---

**Description**

Create a searchlight iterator that samples regions from within a mask. Searchlight centers are sampled \*without\* replacement, but the same voxel can belong to multiple searchlight samples. It is in the latter sense that this is a bootstrap resampling scheme.

**Usage**

```
BootstrapSearchlight(mask, radius, iter = 100)
```

**Arguments**

mask	an image volume containing valid central voxels for roving searchlight
radius	in mm of spherical searchlight
iter	the total number of searchlights to sample (default is 100)

---

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

---

**Description**

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

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
---	---------------------------------

**Value**

a matrix where each row contains the min (column 1) and max (column 2) bounds of the image dimension from 1 to `ndim(image)`.

**Examples**

```
bspace <- BrainSpace(c(10,10,10), c(2,2,2))
b <- bounds(bspace)
nrow(b) == ndim(bspace)
ncol(b) == 2
```

---

BrainBucket-class      *BrainBucket*

---

**Description**

a four-dimensional image that consists of a sequence of labeled image volumes backed by a list  
 Constructor function for [BrainBucket](#) class

**Usage**

```
BrainBucket(volumelist)
```

**Arguments**

volumelist      a named list of [BrainVolume](#) instances

**Value**

an instance of class [BrainBucket](#)

**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

**Examples**

```
vol1 <- BrainVolume(rnorm(24*24*24), BrainSpace(c(24,24,24), c(1,1,1)))
vol2 <- BrainVolume(rnorm(24*24*24), BrainSpace(c(24,24,24), c(1,1,1)))
vol3 <- BrainVolume(rnorm(24*24*24), BrainSpace(c(24,24,24), c(1,1,1)))
vlist <- list(vol1,vol2,vol3)
names(vlist) <- paste0("V", 1:3)
bucket <- BrainBucket(vlist)
all.equal(dim(bucket[[1]]), dim(vol1))
```

---

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  
                                  *BrainFileDescriptor*

---

**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 *BrainFileSource*


---

**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 *BrainMetaInfo*


---

**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             *BrainSource*

---

**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, spacing = NULL, origin = NULL, axes = NULL,  
           trans = NULL)
```

**Arguments**

Dim	a vector describing the dimensions of the spatial grid
spacing	the real-valued voxel dimensions (usually in millimeters)
origin	the coordinate origin of the image space
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      *BrainVector*

---

### Description

Four-dimensional brain image

constructor function for virtual class [BrainVector](#)

### Usage

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

### Arguments

data                    the image data which can be a matrix, a 4d array, or a list of BrainVolumes. If the latter, the geometric space of the data BrainSpace will be inferred from the constituent volumes, which must all be identical.

space                  a [BrainSpace](#) object. Does not need to be included if data argument is a list of BrainVolumes

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. If mask is provided then [SparseBrainVector](#), otherwise [DenseBrainVector](#)



---

BrainVectorSource      *BrainVectorSource*


---

**Description**

Construct a [BrainVectorSource](#) object

**Usage**

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

**Arguments**

fileName	name of the 4-dimensional image file
indices	the subset of integer volume indices to load – if NULL then all volumes will be loaded
mask	image volume indicating the subset of voxels that will be loaded. If provided, function returns <a href="#">SparseBrainVectorSource</a>

**Details**

If a mask is supplied then it should be a [LogicalBrainVolume](#) or [BrainVolume](#) instance. If the latter, then the mask will be defined by nonzero elements of the volume.

**Value**

a instance deriving from [BrainVectorSource](#)

---

BrainVectorSource-class  
*BrainVectorSource*


---

**Description**

A class that is used to produce a [BrainVector](#) instance

**Slots**

indices the index vector of the volumes to be loaded

---

 BrainVolume

*BrainVolume*


---

### Description

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 <a href="#">BrainSpace</a>
source	an instance of class <a href="#">BrainSource</a>
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)
```

---

 BrainVolume-class

*Base class for image representing 3D volumetric data.*


---

### Description

Base class for image representing 3D volumetric data.

---

BrainVolumeSource-class  
*BrainVolumeSource*

---

**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

**Usage**

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

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

**Arguments**

con	the object to close
-----	---------------------

---

clusterCenters	<i>clusterCenters</i>
----------------	-----------------------

---

**Description**

clusterCenters

**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 <a href="#">LogicalBrainVolume</a>
clusters	a vector of clusters ids with length equal to number of nonzero voxels in mask
labelMap	an optional list that maps from cluster id to a cluster label, e.g. (1 -> "FFA", 2 -> "PPA")
source	an optional instance of class <a href="#">BrainSource</a>
label	an optional character string used to label of the volume

**Value**

[ClusteredBrainVolume](#) instance

**Examples**

```

bspace <- BrainSpace(c(16,16,16), spacing=c(1,1,1))
grid <- indexToGrid(bspace, 1:(16*16*16))
kres <- kmeans(grid, centers=10)
mask <- BrainVolume(rep(1, 16^3), bspace)
clusvol <- ClusteredBrainVolume(mask, kres$cluster)

```

---

concat

*Concatenate two data objects*


---

**Description**

Concatenate two data objects

**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, typically BrainVolume or BrainVector
y	the second object, typically BrainVolume or BrainVector
...	additional objects

**Details**

The x and y images must have compatible dimensions. a BrainVolume can be concatenated to BrainVector, and vice versa. See examples.

**Note**

dimensions of x and y must be equal

**Examples**

```

bv1 <- BrainVolume(rep(1,1000), BrainSpace(c(10,10,10), c(1,1,1)))
bv2 <- BrainVolume(rep(2,1000), BrainSpace(c(10,10,10), c(1,1,1)))
bv3 <- concat(bv1,bv2)
inherits(bv3, "BrainVector")

bv4 <- concat(bv3, bv1)
dim(bv4)[4] == 3
bv5 <- concat(bv1, bv3)
dim(bv4)[4] == 3

bv6 <- concat(bv4,bv5)
dim(bv6)[4] == 6

```

---

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	<i>Extract connected components from a 3D mask</i>
------------	--

---

**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	<i>Extract coordinates</i>
--------	----------------------------

---

**Description**

Extract coordinates  
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	<i>Generic function to convert N-dimensional real world coordinates to grid coordinates</i>
-------------	---

---

**Description**

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

**Usage**

```
coordToGrid(x, coords)

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

## S4 method for signature 'BrainSpace,numeric'
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	<i>Generic function to convert N-dimensional real world coordinates to 1D indices</i>
--------------	---

---

**Description**

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

**Usage**

```
coordToIndex(x, coords)

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



```
## S4 method for signature 'BrainSpace,numeric'
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	<i>Generic function to get the name of the data file, given a file name and a <a href="#">BrainFileDescriptor</a> instance.</i>
----------	---

---

### 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	<i>Generic function to test whether a file name conforms to the given a <a href="#">BrainFileDescriptor</a> instance. Will test for match to data file only</i>
-----------------	---

---

**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 be matched to
fileName	file name to be matched

**Value**

TRUE for match, FALSE otherwise

---

dataReader	<i>Generic function to create data reader</i>
------------	---

---

**Description**

Generic function to create data reader

**Usage**

```
dataReader(x, offset)
```

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

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

**Arguments**

x	an object specifying the information 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-dimensional array or a 2-dimension matrix that is either nvoxels by ntime-points or ntime-points by nvoxels
space	a <a href="#">BrainSpace</a> object
source	an optional <a href="#">BrainSource</a> 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 <a href="#">BrainSpace</a>
source	an instance of class <a href="#">BrainSource</a>
label	a character string
indices	an optional 1-d index vector

**Value**

[DenseBrainVolume](#) instance

---

*dim,BrainData-method*    *dim of BrainData object*

---

**Description**

dim of BrainData object

**Usage**

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

**Arguments**

x                    the object

---

*dim,BrainSpace-method*    *dim*

---

**Description**

dim

**Usage**

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

**Arguments**

x                    the object

---

dim,FileMetaInfo-method  
*dim of FileMetaInfo*

---

### Description

dim of FileMetaInfo

### Usage

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

### Arguments

x                    the object

---

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

---

### Description

Generic function to drop a dimension from an object

### Usage

```
dropDim(x, dimnum)

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

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

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

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

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

**Arguments**

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

**Examples**

```
x = BrainSpace(c(10,10,10), c(1,1,1))
x1 <- dropDim(x)
ndim(x1) == 2
dim(x1)[2] == 10
```

---

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

**Details**

when x is a SparseBrainVector eachSeries only iterates over nonzero series.

**Examples**

```
bvec <- BrainVector(array(rnorm(24*24*24*24), c(24,24,24,24)), BrainSpace(c(24,24,24,24), c(1,1,1)))
res <- eachSeries(bvec, mean)
length(res) == 24*24*24
```

---

eachSlice	<i>Generic functions to apply a function to each (2D) slice of an image</i>
-----------	---

---

**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	<i>Generic function to apply a function to each volume of a four-dimensional image</i>
------------	--

---

**Description**

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

eachVolume

eachVolume

eachVolume

eachVolume

**Usage**

```

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

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

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

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

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

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

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

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

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

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

```

**Arguments**

x	four-dimensional image, e.g. of class BrainVector
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
mask	an image mask indicating subset of elements to apply function over.



... additional arguments

### Value

a list of results of apply FUN to each volume.

### Examples

```
bvec <- BrainVector(array(rnorm(24*24*24*24), c(24,24,24,24)), BrainSpace(c(24,24,24,24), c(1,1,1)))
res <- eachVolume(bvec, mean)

res <- eachVolume(bvec, function(x,i) median(x), withIndex=TRUE)
```

---

fileMatches	<i>Generic function to test whether a file name conforms to the given <a href="#">BrainFileDescriptor</a> instance. Will test for match to either header file or data file</i>
-------------	--

---

### 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 be matched to
fileName	file name to be matched

### Value

TRUE for match, FALSE otherwise

---

FileMetaInfo-class      *FileMetaInfo*

---

### 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 number of sub-images)

slope image multiplier – multiple values allowed (must equal number 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,list'
fill(x, lookup)
```

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

**Arguments**

x                    the object to map values from  
 lookup              the lookup table. The first column is the "key" the second column is the "value".

**Value**

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

**Examples**

```
x <- BrainSpace(c(10,10,10), c(1,1,1))
vol <- BrainVolume(sample(1:10, 10*10*10, replace=TRUE), x)

## lookup table is list
lookup <- lapply(1:10, function(i) i*10)
ovol <- fill(vol, lookup)

## lookup table is matrix. First column is key, second column is value
names(lookup) <- 1:length(lookup)
lookup.mat <- cbind(as.numeric(names(lookup)), unlist(lookup))
ovol2 <- fill(vol, lookup.mat)
all.equal(as.vector(ovol2), as.vector(ovol))
```

---

gridToCoord	<i>Generic function to convert N-dimensional grid coordinates to real world coordinates</i>
-------------	---

---

**Description**

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

**Usage**

```
gridToCoord(x, coords)

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

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

**Arguments**

x                    the object  
 coords              a matrix of grid coordinates

**Value**

a matrix of real coordinates

---

gridToIndex	<i>Generic function to convert N-dimensional grid coordinate to 1D indices</i>
-------------	--

---

### Description

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

### 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, typically a BrainVolume or BrainSpace instance.
coords	a matrix where each row is a coordinate or a vector of length equal to ndim(x)

### Value

a vector of indices

---

headerFile	<i>Generic function to get the name of the header file, given a file name and a <a href="#">BrainFileDescriptor</a> instance.</i>
------------	---

---

### 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	<i>Generic function to test whether a file name conforms to the given <a href="#">BrainFileDescriptor</a> instance. Will test for match to header file only</i>
-------------------	---

---

**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 be 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 = gray((0:255)/255, alpha = 1),
      zero.col = "#000000", axis = 3, ...)

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

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

### Arguments

x	the object to display
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)
axis	the axis index
...	extra arguments to passed to grid.raster
zpos	the z coordinate

---

imageGrid                      *imageGrid*

---

### Description

Display a set of mages slices in a 2D montage

### Usage

```
imageGrid(layer, gridDim = c(3, 3), zstart, zend, panelSize = 3,
          panelUnit = "inches", interpolate = FALSE, fontCol = "red")
```

**Arguments**

layer	the layer to display
gridDim	the dimensions of the 2D grid montage
zstart	the z coordinate of the first slice
zend	the z coordinate of the last slice
panelSize	the size of each panel in the montage (default unit is inches)
panelUnit	the unit for the panel size (default is inches)
interpolate	whether to interpolate pixel values
fontCol	color of labels indicating slice level

---

 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

**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

**Examples**

```

bvol <- BrainVolume(array(0, c(10,10,10)), BrainSpace(c(10,10,10), c(1,1,1)))
idx <- 1:10
g <- indexToCoord(bvol, idx)
idx2 <- coordToIndex(bvol, g)
all.equal(idx, idx2)

```

---

indexToGrid	<i>Generic function to convert 1D indices to N-dimensional grid coordinates</i>
-------------	---

---

**Description**

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

**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 vector of indices

**Value**

a matrix of grid coordinates

**Examples**

```
bvol <- BrainVolume(array(0, c(10,10,10)), BrainSpace(c(10,10,10), c(1,1,1)))
idx <- 1:10
g <- indexToGrid(bvol, idx)
bvol[g]
```

---

indices	<i>Extract indices</i>
---------	------------------------

---

**Description**

Extract 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	<i>Generic getter to extract inverse image coordinate transformation</i>
--------------	--

---

**Description**

Generic getter to extract inverse image coordinate transformation

**Usage**

```
inverseTrans(x)

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

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

**Arguments**

x                    an object

**Examples**

```
bspace <- BrainSpace(c(10,10,10), c(2,2,2))
itrans <- inverseTrans(bspace)
identical(trans(bspace) %*% inverseTrans(bspace), diag(4))
```

---

Kernel	<i>Create a Kernel object</i>
--------	-------------------------------

---

**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	<i>Kernel</i>
--------------	---------------

---

**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	<i>Layer</i>
-------	--------------

---

**Description**

create a [Layer](#) object

**Usage**

```
Layer(vol, colorMap = gray((0:255)/255, alpha = 1), thresh = c(0, 0),
      axis = 3, zero.col = "#000000")
```

**Arguments**

vol	volume instance of <a href="#">BrainVolume</a>
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.
axis	the axis index of the axis perpendicular to the xy plane (options: 1,2,3; default is 3)
zero.col	the color used when the value is zero.

**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.

axis the axis index of perpendicular to the xy plane (option: 1,2,3; default is 3)

zero.col the color pixels with intensity of zero. This value overrides the color from the slot colorMap

---

 length,ROIVolume-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 'ROIVolume'
length(x)
```

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

### Arguments

x the object to get length

---

loadBucket	<i>loadBucket</i>
------------	-------------------

---

**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	<i>Generic function to load data from a data source</i>
----------	---

---

**Description**

Generic function to load data from a data source

loadData

Load data from a [BrainBucketSource](#)

load a BrainVolume

**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	<i>loadVector</i>
------------	-------------------

---

**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	<i>load an image volume from a file</i>
------------	---

---

**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 [DenseBrainVolume](#)

**Examples**

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

---

loadVolumeList	<i>loadVolList</i>
----------------	--------------------

---

**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, a 1D vector with length equal to <code>prod(dim(space))</code> , or a set of indices where elements are TRUE
space	an instance of class <a href="#">BrainSpace</a>
source	an instance of class <a href="#">BrainSource</a>
label	a character string
indices	an optional 1-d index vector

### Value

[LogicalBrainVolume](#) instance

---

lookup

*Index Lookup operation*

---

### Description

Index Lookup operation

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	<i>makeVector</i>
------------	-------------------

---

**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 <a href="#">BrainVector</a> or <a href="#">BrainVolume</a> containing the reference space for the new vector.
source	an instance of class <a href="#">BrainSource</a>
label	a character string

**Value**

[DenseBrainVector](#) instance

---

makeVolume	<i>makeVolume</i>
------------	-------------------

---

**Description**

Construct a [BrainVolume](#) instance, using default (dense) implementation

**Usage**

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

**Arguments**

data	an optional one- or three-dimensional vector or array
refvol	an instance of class <a href="#">BrainVolume</a> containing the reference space for the new volume.
source	an optional instance of class <a href="#">BrainSource</a>
label	an optional character string
indices	an optional 1d vector of indices in to the 3d space

**Value**

[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")
bvol2 <- makeVolume(dat, bvol)
all.equal(as.array(bvol),as.array(bvol2))
data <- 1:10
indices = seq(1,1000, length.out=10)
bvol3 <- makeVolume(data,bvol,indices=indices)
sum(bvol3) == sum(data)

```

---

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 third axis

---

matrixToVolumeList	<i>matrixToVolumeList converts a matrix to a list of BrainVolumes with values filled at grid coordinates determined by the vox argument.</i>
--------------------	--

---

### Description

matrixToVolumeList converts a matrix to a list of BrainVolumes with values filled at grid coordinates determined by the vox argument.

### Usage

```
matrixToVolumeList(voxmat, mat, mask, default = NA)
```

### Arguments

voxmat	an N by 3 matrix of voxel coordinates
mat	an N by M matrix of values where M is the number of volumes to create (e.g. one volume per column in mat)
mask	a reference volume defining the geometry of the output volumes. This can either be of type BrainSpace or BrainVolume
default	the value that will be used for voxels not contained within voxmat (default is NA)

### Value

a list of BrainVolume instances, one for each column of mat

---

mergePartitions	<i>mergePartitions</i>
-----------------	------------------------

---

### 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

---

MNI_SPACE_1MM	<i>MNI SPACE 1MM</i>
---------------	----------------------

---

**Description**

This is a BrainSpace object encoding the geometry of the MNI\_1MM template.

**Usage**

```
data(MNI_SPACE_1MM)
```

**Format**

a BrainSpace instance

---

NamedAxis-class	<i>NamedAxis</i>
-----------------	------------------

---

**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  
*names*

---

### Description

names

### Usage

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

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

### Arguments

x                    the object to get names of

---

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

---

### Description

Generic function to extract the number of dimensions of an object

### 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

**Examples**

```
x = BrainSpace(c(10,10,10), c(1,1,1))
ndim(x) == 3
x = BrainSpace(c(10,10,10,3), c(1,1,1,1))
ndim(x) == 4
```

---

neuroim

*neuroim*


---

**Description**

Data structures for analysis of neuroimaging data.

---

NIFTIFileDescriptor-class

*NIFTIFileDescriptor*


---

**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      *NullMetaInfo*

---

### Description

This class is used to denote the absence 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

### Usage

```
origin(x)

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

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



**Arguments**

x                    an object with an origin

**Examples**

```
bspaced <- BrainSpace(c(10,10,10), c(2,2,2))
origin(bspaced)
```

---

overlay                    *overlay two objects*

---

**Description**

overlay two objects

overlay

**Usage**

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

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

```
## S4 method for signature 'Overlay,Layer'
e1 + e2
```

```
## S4 method for signature 'Layer,Layer'
e1 + e2
```

**Arguments**

x                    the underlay object

y                    the overlay object

...                  additional arguments for class-specific implementations

e1                   the left operand

e2                   the right operand

---

partition	<i>partition</i>
-----------	------------------

---

**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	<i>Extract permutation matrix</i>
---------	-----------------------------------

---

**Description**

Extract permutation matrix

permMat

**Usage**

```
permMat(x, ...)
```

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

**Arguments**

x	the object
...	additional arguments

---

pick	<i>pick</i>
------	-------------

---

**Description**

pick

**Usage**

pick(x, mask, ...)

**Arguments**

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

---

print	<i>Generic function to print an object</i>
-------	--

---

**Description**

Generic function to print an object

**Usage**

print(x, ...)

**Arguments**

x	the object to print
...	additional arguments

---

`print,AxisSet2D-method`

*print a AxisSet2D instance*

---

### **Description**

print a AxisSet2D instance

### **Usage**

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

### **Arguments**

<code>x</code>	the object
<code>...</code>	extra args

---

`print,AxisSet3D-method`

*print a AxisSet3D instance*

---

### **Description**

print a AxisSet3D instance

### **Usage**

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

### **Arguments**

<code>x</code>	the object
<code>...</code>	extra args

---

print,NamedAxis-method  
*print a NamedAxis*

---

**Description**

print a NamedAxis

**Usage**

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

**Arguments**

x	the object
...	extra arguments

---

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

---

readAFNIHeader	<i>readAFNIHeader</i>
----------------	-----------------------

---

**Description**

readAFNIHeader

**Usage**

```
readAFNIHeader(fileName)
```

**Arguments**

fileName	the name of the AFNI header file (ending in .HEAD)
----------	--

---

readElements	<i>Generic function to read a sequence of elements from an input source</i>
--------------	---

---

**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	<i>read header information of an image file</i>
------------	---

---

**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	<i>Generic function to read image meta info given a file and a <a href="#">BrainFileDescriptor</a> instance.</i>
--------------	--

---

**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 containing meta information

---

RegionCube                      *Create A Cuboid Region of Interest*

---

### Description

Create A Cuboid Region of Interest

### Usage

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

### Arguments

bvol	an BrainVolume or BrainSpace instance
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. If bvol is A BrainSpace then this argument is ignored.

### Value

an instance of class ROIVolume

### Examples

```
sp1 <- BrainSpace(c(10,10,10), c(1,1,1))
cube <- RegionCube(sp1, c(5,5,5), 3)
vox = coords(cube)
cube2 <- RegionCube(sp1, c(5,5,5), 3, fill=5)
```

---

RegionSphere                      *Create A Spherical Region of Interest*

---

### Description

Create A Spherical Region of Interest

### Usage

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



**Arguments**

bvol	an BrainVolume or BrainSpace instance
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

**Examples**

```
sp1 <- BrainSpace(c(10,10,10), c(1,1,1))
cube <- RegionSphere(sp1, c(5,5,5), 3.5)
vox = coords(cube)
```

---

RegionSquare	<i>Create A Square Region of Interest where the z-dimension is fixed at one coordinate.</i>
--------------	---

---

**Description**

Create A Square Region of Interest where the z-dimension is fixed at one coordinate.

**Usage**

```
RegionSquare(bvol, centroid, surround, fill = NULL, nonzero = FALSE,
  fixdim = 3)
```

**Arguments**

bvol	an BrainVolume or BrainSpace instance
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. If bvol is A BrainSpace then this argument is ignored.
fixdim	the fixed dimension is the third, or z, dimension.

**Value**

an instance of class ROIVolume

**Examples**

```

sp1 <- BrainSpace(c(10,10,10), c(1,1,1))
square <- RegionSquare(sp1, c(5,5,5), 1)
vox <- coords(square)
## a 3 X 3 X 1 grid
nrow(vox) == 9

```

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 representing a volumetric region of interest (ROI).

**Slots**

data	the numeric data stored in the ROI
coords	the voxel coordinates of the ROI

---

scaleSeries	<i>Generic functions to scale (center and/or normalize by standard deviation) each series of a 4D image That is, if the 4th dimension is 'time' each series is a 1D time series.</i>
-------------	--

---

### Description

Generic functions to scale (center and/or normalize by standard deviation) each series of a 4D image  
That is, if the 4th dimension is 'time' each series is a 1D time series.

### Usage

```
scaleSeries(x, center, scale)

## S4 method for signature 'BrainVector,logical,logical'
scaleSeries(x, center, scale)

## S4 method for signature 'BrainVector,missing,logical'
scaleSeries(x, center, scale)

## S4 method for signature 'BrainVector,missing,missing'
scaleSeries(x, center, scale)

## S4 method for signature 'BrainVector,logical,missing'
scaleSeries(x, center, scale)
```

### Arguments

x	a four dimensional image
center	a logical value indicating whether series should be centered. TRUE if not specified.
scale	a logical value indicating whether series should be divided by standard deviation. TRUE if not specified.

### Examples

```
bvec <- BrainVector(array(rnorm(24*24*24*24), c(24,24,24,24)), BrainSpace(c(24,24,24,24), c(1,1,1)))
res <- scaleSeries(bvec, TRUE, TRUE)
```

---

Searchlight	<i>Create an exhaustive searchlight iterator</i>
-------------	--

---

**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	<i>Extract vector series from object</i>
--------	--

---

**Description**

Extract vector series from object

**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 for 2nd dimension
k	index for 3rd dimension

---

seriesIter	<i>seriesIter</i>
------------	-------------------

---

**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 over. This is typically an instance of class [BrainVector](#)

**Value**

an iter object from the iterators package.

**Methods (by class)**

- BrainVector: get a series iterator for a [BrainVector](#) instance
- SparseBrainVector: get a seriesIter for a [SparseBrainVector](#) instance

**Examples**

```
## create a BrainVector with 10X10X10X10, where the last dimension is
## by convention the fourth dimension.
bvec <- BrainVector(array(rnorm(10*10*10*10), rep(10,4)), BrainSpace(rep(10,4), c(1,1,1)))
iter <- seriesIter(bvec)

## compute mean of each series
library(foreach)
library(iterators)
foreach(i=iter, .combine=c) %do% { mean(i) }
iter <- seriesIter(bvec)

## combine all series into a matrix
foreach(i=iter, .combine=rbind) %do% { i }

## scale all series, add as columns in matrix.
foreach(i=seriesIter(bvec), .combine=cbind) %do% { scale(i) }
```

---

*show,AxisSet1D-method* *show an AxisSet1D*

---

**Description**

show an AxisSet1D

**Usage**

```
## S4 method for signature 'AxisSet1D'  
show(object)
```

**Arguments**

object            the object

---

*show,AxisSet2D-method* *show an AxisSet2D*

---

**Description**

show an AxisSet2D

**Usage**

```
## S4 method for signature 'AxisSet2D'  
show(object)
```

**Arguments**

object            the object

---

*show,AxisSet3D-method* *show an AxisSet3D*

---

**Description**

show an AxisSet3D

**Usage**

```
## S4 method for signature 'AxisSet3D'  
show(object)
```

**Arguments**

object            the object

---

show,AxisSet4D-method *show an AxisSet4D*

---

### **Description**

show an AxisSet4D

### **Usage**

```
## S4 method for signature 'AxisSet4D'  
show(object)
```

### **Arguments**

object            the object

---

show,BaseMetaInfo-method  
*show a BaseMetaInfo*

---

### **Description**

show a BaseMetaInfo

### **Usage**

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

### **Arguments**

object            the object

---

show,BrainSpace-method

*show a BrainSpace*

---

### **Description**

show a BrainSpace

### **Usage**

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

### **Arguments**

object            the object

---

show,BrainVector-method

*show a BrainVector*

---

### **Description**

show a BrainVector

### **Usage**

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

### **Arguments**

object            the object



---

show,BrainVectorSource-method  
*show a BrainVectorSource*

---

### **Description**

show a BrainVectorSource

### **Usage**

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

### **Arguments**

object            the object

---

show,BrainVolume-method  
*show a BrainVolume*

---

### **Description**

show a BrainVolume

### **Usage**

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

### **Arguments**

object            the object

show,FileMetaInfo-method  
*show a FileMetaInfo*

---

**Description**

show a FileMetaInfo

**Usage**

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

**Arguments**

object            the object

---

show,NamedAxis-method    *show an NamedAxis*

---

**Description**

show an NamedAxis

**Usage**

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

**Arguments**

object            the object

---

show,NullMetaInfo-method  
*show a NullMetaInfo*

---

**Description**

show a NullMetaInfo

**Usage**

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

**Arguments**

object            the object

---

show,ROIVolume-method *show an ROIVolume*

---

**Description**

show an ROIVolume

**Usage**

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

**Arguments**

object            the object

---

show, SparseBrainVector-method  
*show a SparseBrainVector*

---

### Description

show a SparseBrainVector

### Usage

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

### Arguments

object            the 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	<i>Generic function to extract geometric properties of an image.</i>
-------	--

---

**Description**

Generic function to extract geometric properties of an image.

**Usage**

```
space(x, ...)  
  
## S4 method for signature 'BrainData'  
space(x)  
  
## S4 method for signature 'BrainSpace'  
space(x)  
  
## S4 method for signature 'IndexLookupVolume'  
space(x)
```

**Arguments**

x	the object to query, e.g. an instance of <code>BrainVolume</code> or <code>BrainVector</code>
...	additional arguments

**Value**

an object representing the geometric space of the image of type `BrainSpace`

**Examples**

```
x = BrainSpace(c(10,10,10), c(1,1,1))  
vol <- BrainVolume(rnorm(10*10*10), x)  
identical(x, space(vol))
```

---

spacing	<i>Generic function to extract the voxel dimensions of an image</i>
---------	---

---

**Description**

Generic function to extract the voxel dimensions of an image

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

**Examples**

```
bspace <- BrainSpace(c(10,10,10), c(2,2,2))
all.equal(spacing(bspace), c(2,2,2))
```

---

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 <code>BrainSpace</code> instance
mask	a 3D array of type <code>logical</code>
source	the data source – an instance of class <code>BrainSource</code>
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

**Examples**

```

bpace <- BrainSpace(c(10,10,10,100), c(1,1,1))
mask <- array(rnorm(10*10*10) > .5, c(10,10,10))
mat <- matrix(rnorm(sum(mask)), 100, sum(mask))
svec <- SparseBrainVector(mat, bspace,mask)
length(indices(svec)) == sum(mask)

```

---

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, indices = NULL, source = NULL, label = "")
```

### Arguments

data	a numeric vector
space	an instance of class <a href="#">BrainSpace</a>
indices	a index vector indicating the 1-d coordinates of the data values
source	an instance of class <a href="#">BrainSource</a>
label	a character string

### Value

[SparseBrainVolume](#) instance

### Slots

data a sparseVector instance

### Examples

```
data <- 1:10
indices = seq(1,1000, length.out=10)
ospace <- BrainSpace(c(64,64,64), spacing=c(1,1,1))
sparsevol <- SparseBrainVolume(data,ospace,indices=indices)
densevol <- BrainVolume(data,ospace,indices=indices)
sum(sparsevol) == sum(densevol)
```



---

splitFill	<i>Generic function to fill disjoint sets of values with the output of a function</i>
-----------	---

---

### Description

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

### 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 sets

### Details

FUN can either return a scalar for each input vector or a vector equal to the length of the input vector. If it returns a scalar then every voxel in the set will be filled with that value in the output vector.

### Value

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

### Examples

```
## summarize with mean -- FUN returns a scalar  
x = BrainSpace(c(10,10,10), c(1,1,1))  
vol <- BrainVolume(rnorm(10*10*10), x)  
fac <- factor(rep(1:10, length.out=1000))  
ovol.mean <- splitFill(vol, fac, mean)  
identical(dim(ovol.mean), dim(vol))  
length(unique(as.vector(ovol.mean))) == 10  
## transform by reversing vector -- FUN returns a vector.  
ovol2 <- splitFill(vol, fac, rev)
```

---

splitReduce	<i>Generic function to summarize subsets of an object by first splitting by row and then "reducing" by a summary function</i>
-------------	---

---

### Description

Generic function to summarize subsets of an object by first splitting by row and then "reducing" by a summary function

### Usage

```
splitReduce(x, fac, FUN)

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

## S4 method for signature 'matrix,integer,missing'
splitReduce(x, fac)

## S4 method for signature 'matrix,factor,missing'
splitReduce(x, fac)

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

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

## S4 method for signature 'BrainVector,factor,missing'
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. if FUN is missing, than the mean of each sub-matrix column is computed.

### Details

if FUN is supplied it must take a vector and return a single scalar value. If it returns more than one value, an error will occur.

if x is a BrainVector instance then voxels (dims 1:3) are treated as columns and time-series (dim 4) as rows. The summary function then is applied to groups of voxels. However, if the goal is to apply a function to groups of time-points, then this can be achieved as follows:

```
splitReduce(t(as.matrix(bvec)), fac)
```

**Value**

a new matrix where the original values have been reduced

**Examples**

```
mat = matrix(rnorm(100*100), 100, 100)
fac = sample(1:3, nrow(mat), replace=TRUE)
## compute column means of each sub-matrix
ms <- splitReduce(mat, fac)
all.equal(row.names(ms), levels(fac))

## compute column medians of each sub-matrix
ms <- splitReduce(mat, fac, median)

## compute time-series means grouped over voxels.
## Here, \code{length(fac)} must equal the number of voxels: \code{prod(dim(bvec)[1:3])}
bvec <- BrainVector(array(rnorm(24*24*24*24), c(24,24,24,24)), BrainSpace(c(24,24,24,24), c(1,1,1)))
fac <- factor(sample(1:3, prod(dim(bvec)[1:3]), replace=TRUE))
ms <- splitReduce(bvec, fac)
ms2 <- splitReduce(bvec, fac, mean)
all.equal(row.names(ms), levels(fac))
all.equal(ms,ms2)
```

---

splitScale

*Generic function to center/scale row-subsets of a matrix or matrix-like object*

---

**Description**

Generic function to center/scale row-subsets of a matrix or matrix-like 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 or matrix-like object

f the splitting object, typically a factor or set of integer indices. must be equal to number of rows of matrix.

center	should values within each submatrix be centered? (mean removed from each column of submatrix)
scale	should values be scaled? (divide vector by standard deviation from each column of submatrix)

**Value**

a new matrix or matrix-like object where the original rows have been grouped by `f` and then centered and/or scaled for each grouping

**Examples**

```
M <- matrix(rnorm(1000), 10, 100)
fac <- factor(rep(1:2, each=5))
Ms <- splitScale(M, fac)

## correctly centered
all(abs(apply(Ms[fac == 1,], 2, mean)) < .000001)
all(abs(apply(Ms[fac == 2,], 2, mean)) < .000001)

# correctly scaled
all.equal(apply(Ms[fac == 1,], 2, sd), rep(1, ncol(Ms)))
all.equal(apply(Ms[fac == 2,], 2, sd), rep(1, ncol(Ms)))
```

---

stripExtension	<i>Generic function to strip extension from file name, given a <a href="#">BrainFileDescriptor</a> instance.</i>
----------------	--

---

**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**

<code>x</code>	descriptor instance
<code>fileName</code>	file name to be stripped of its extension

**Value**

`fileName` without extension

---

takeSeries	<i>Generic function to extract a set of series from a 4D image</i>
------------	--

---

**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	<i>Generic function to extract a one or more individual volumes from a four-dimensional image</i>
------------	---

---

**Description**

Generic function to extract a one or more individual volumes from a four-dimensional image

**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

**Value**

a list of BrainVolume elements

**Examples**

```

bvec <- BrainVector(array(rnorm(24*24*24*24), c(24,24,24,24)), BrainSpace(c(24,24,24,24), c(1,1,1)))
vol <- takeVolume(bvec,1)
all.equal(dim(vol), c(24,24,24))

vol <- takeVolume(bvec,1:3)
length(vol) == 3
class(vol) == "list"

```

---

tessellate	<i>tessellate</i>
------------	-------------------

---

**Description**

tessellate  
tessellate a LogicalBrainVolume into K spatial disjoint components

**Usage**

```

tessellate(x, K, ...)

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

```

**Arguments**

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

---

trans	<i>Generic getter to extract image coordinate transformation</i>
-------	--

---

**Description**

Generic getter to extract image coordinate transformation

**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

**Details**

This function returns a transformation that can be used to go from "grid coordinates" to "real world coordinates" in millimeters. see [BrainSpace](#)

**Examples**

```

bspace <- BrainSpace(c(10,10,10), c(2,2,2))
trans(bspace)
all.equal(dim(trans(bspace)), c(4,4))

```

---

values                    *Generic function to extract data values of object*

---

**Description**

Generic function to extract data values of object

**Usage**

```

values(x, ...)

## S4 method for signature 'ROIVolume'
values(x, ...)

```

**Arguments**

x                    the object to get values from  
...                    additional arguments

---

voxels	<i>extract voxel coordinates</i>
--------	----------------------------------

---

**Description**

extract voxel coordinates

**Usage**

```
voxels(x, ...)
```

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

**Arguments**

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

---

writeElements	<i>Generic function to write a sequence of elements from an input source</i>
---------------	--

---

**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 a 4D image vector to disk*

---

## Description

Generic function to write a 4D image vector to disk

## 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	an image object, typically a BrainVector instance.
fileName	output file name.
format	file format string. Since "NIFTI" is the only currently supported format, this parameter can be safely ignored and omitted.
dataType	the numeric data type. If specified should be a character vector of: "BINARY", "UBYTE", "SHORT", "INT", "FLOAT", "DOUBLE". Otherwise output format will be inferred from R the datatype of the image.

## Examples

```
bvec <- BrainVector(array(0, c(10,10,10,10)), BrainSpace(c(10,10,10,10), c(1,1,1)))
## Not run:
writeVector(bvec, "out.nii")
writeVector(bvec, "out.nii.gz")

## End(Not run)
```

---

writeVolume                      *Generic function to write a 3D image volume to disk*

---

### Description

Generic function to write a 3D 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, typically a BrainVolume instance.
fileName	output file name
format	file format string. Since "NIFTI" is the only currently supported format, this parameter can be safely ignored and omitted.
dataType	output data type. If specified should be a character vector of: "BINARY", "UBYTE", "SHORT", "INT", "FLOAT", "DOUBLE". Otherwise output format will be inferred from R the datatype of the image.

### Details

The output format will be inferred from file extension. `writeVolume(x, "out.nii")` outputs a NIFTI file. `writeVolume(x, "out.nii.gz")` outputs a gzipped NIFTI file.

No other file output formats are currently supported.

### Examples

```
bvol <- BrainVolume(array(0, c(10,10,10)), BrainSpace(c(10,10,10), c(1,1,1)))  
## Not run:  
writeVolume(bvol, "out.nii")  
writeVolume(bvol, "out.nii.gz")  
  
## End(Not run)
```

---

[,BrainBucket,index,missing,ANY-method  
*extract labeled volume from BrainBucket*

---

### Description

extract labeled volume from BrainBucket

### Usage

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

### Arguments

x	the object
i	first index

---

[,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

x	the object
i	first index
j	second index
drop	drop dimension

---

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

---

**Description**

extractor

**Usage**

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

**Arguments**

x	the object
i	first index
j	second index
k	third index
m	the fourth index
...	additional args
drop	dimension

---

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

---

**Description**

extractor

**Usage**

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

**Arguments**

x	the object
i	first index
j	second index
k	third index
m	the fourth index
...	additional args
drop	dimension

---

[,SparseBrainVector,numeric,missing,ANY-method  
*extractor*

---

**Description**

extractor

**Usage**

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

**Arguments**

x	the object
i	first index
j	second index
k	third index
m	the fourth index
...	additional args
drop	dimension

---

[,SparseBrainVector,numeric,numeric,ANY-method  
*extractor*

---

**Description**

extractor

**Usage**

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

**Arguments**

x	the object
i	first index
j	second index
k	third index
m	the fourth index
...	additional args
drop	dimension

---

[,SparseBrainVolume,matrix,missing,ANY-method  
*extractor*

---

**Description**

extractor

**Usage**

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

**Arguments**

x	the object
i	first index
j	second index
k	third index
...	additional args
drop	dimension

---

[,SparseBrainVolume,missing,missing,ANY-method  
*extractor*

---

**Description**

extractor

**Usage**

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

**Arguments**

x	the object
i	first index
j	second index
k	third index
...	additional args
drop	dimension

---

[,SparseBrainVolume,missing,numeric,ANY-method  
*extractor*

---

**Description**

extractor

**Usage**

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

**Arguments**

x	the object
i	first index
j	second index
k	third index
...	additional args
drop	dimension

---

[,SparseBrainVolume,numeric,missing,missing-method  
*extractor*

---

**Description**

extractor

**Usage**

```
## S4 method for signature 'SparseBrainVolume,numeric,missing,missing'
x[i, j, k, ..., drop]
```

**Arguments**

x	the object
i	first index
j	second index
k	third index
...	additional args
drop	drop dimension



---

[,SparseBrainVolume,numeric,numeric,ANY-method  
*extractor*

---

### Description

extractor

### Usage

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

### Arguments

x	the object
i	first index
j	second index
k	third index
...	additional args
drop	dimension

---

[[,BrainBucket,index,missing-method  
*extract labeled volume from BrainBucket*

---

### Description

extract labeled volume from BrainBucket

### Usage

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

### Arguments

x	the object
i	the first index

# Index

- \*Topic **datasets**
- MNI\_SPACE\_1MM, [61](#)
- +, Layer, Layer-method (overlay), [65](#)
- +, Overlay, Layer-method (overlay), [65](#)
- [, BrainBucket, index, missing, ANY-method, [99](#)
- [, ROIVolume, numeric, missing, ANY-method, [99](#)
- [, SparseBrainVector, missing, missing, ANY-method, [100](#)
- [, SparseBrainVector, missing, numeric, ANY-method, [100](#)
- [, SparseBrainVector, numeric, missing, ANY-method, [101](#)
- [, SparseBrainVector, numeric, numeric, ANY-method, [102](#)
- [, SparseBrainVolume, matrix, missing, ANY-method, [102](#)
- [, SparseBrainVolume, missing, missing, ANY-method, [103](#)
- [, SparseBrainVolume, missing, numeric, ANY-method, [104](#)
- [, SparseBrainVolume, numeric, missing, missing-method, [104](#)
- [, SparseBrainVolume, numeric, numeric, ANY-method, [105](#)
- [[, BrainBucket, index, missing-method, [105](#)
  
- addDim, [5](#)
- addDim, BrainSpace, numeric-method (addDim), [5](#)
- AFNIFileDescriptor, [6](#)
- AFNIFileDescriptor-class, [6](#)
- AFNIMetaInfo, [6, 6](#)
- AFNIMetaInfo-class (FileMetaInfo-class), [42](#)
- as, [7](#)
- as.array, BrainData-method, [7](#)
- as.list, BrainVector-method, [8](#)
- as.list, SparseBrainVector-method (as.list, BrainVector-method), [8](#)
- as.logical, BrainVolume-method, [8](#)
- as.mask, [9](#)
- as.mask, BrainVolume, missing-method (as.mask), [9](#)
- as.mask, BrainVolume, numeric-method (as.mask), [9](#)
- as.matrix, BrainData-method, [9](#)
- as.matrix, DenseBrainVector-method, [10](#)
- as.matrix, SparseBrainVector-method (as.matrix, DenseBrainVector-method), [10](#)
- as.numeric, SparseBrainVolume-method, [10](#)
- as.raster, Layer-method, [11](#)
- as.sparse, [11](#)
- as.sparse, DenseBrainVector, LogicalBrainVolume-method (as.sparse), [11](#)
- as.sparse, DenseBrainVector, numeric-method (as.sparse), [11](#)
- as.sparse, DenseBrainVolume, LogicalBrainVolume-method (as.sparse), [11](#)
- as.sparse, DenseBrainVolume, numeric-method (as.sparse), [11](#)
- as.vector, BrainData, ANY-method, [12](#)
- axes, [13](#)
- axes, BrainData-method (axes), [13](#)
- axes, BrainSpace-method (axes), [13](#)
- AxisSet-class, [13](#)
- AxisSet1D-class, [13](#)
- AxisSet2D-class, [14](#)
- AxisSet3D-class, [14](#)
- AxisSet4D-class, [14](#)
- AxisSet5D-class, [14](#)
- axisToIndex, [15](#)
- axisToIndex, BrainSpace, numeric, numeric-method (axisToIndex), [15](#)
  
- Base-class, [15](#)

- BaseMetaInfo-class, 15
- BaseSource, 20
- BaseSource-class, 16
- BinaryReader, 16, 16
- BinaryReader-class, 16
- BinaryWriter, 17
- BinaryWriter (BinaryWriter-class), 17
- BinaryWriter-class, 17
- BootstrapSearchlight, 17
- bounds, 18
- bounds,BrainData-method (bounds), 18
- bounds,BrainSpace-method (bounds), 18
- BrainBucket, 19
- BrainBucket (BrainBucket-class), 19
- BrainBucket-class, 19
- BrainBucketSource, 19, 53
- BrainBucketSource (BrainBucketSource-class), 19
- BrainBucketSource-class, 19
- BrainData-class, 20
- BrainFileDescriptor, 33, 34, 41, 44, 45, 71, 92
- BrainFileDescriptor-class, 20
- BrainFileSource-class, 21
- BrainMetaInfo, 21, 87
- BrainMetaInfo (BrainMetaInfo-class), 21
- BrainMetaInfo-class, 21
- BrainSlice, 22
- BrainSlice-class, 22
- BrainSource, 24, 26, 28, 35, 56–58, 86, 88
- BrainSource-class, 22
- BrainSpace, 5, 20, 23, 23, 24, 26, 35, 56, 85, 88, 95
- BrainSpace-class, 24
- BrainVector, 24, 25, 54, 55, 57, 77
- BrainVector (BrainVector-class), 24
- BrainVector-class, 24
- BrainVectorSource, 25, 25
- BrainVectorSource-class, 25
- BrainVolume, 19, 25, 26, 26, 27, 51, 54, 57, 58
- BrainVolume-class, 26
- BrainVolumeSource (BrainVolumeSource-class), 27
- BrainVolumeSource-class, 27
- close,BinaryReader-method, 27
- close,BinaryWriter-method (close,BinaryReader-method), 27
- clusterCenters, 28
- clusterCenters,ClusteredBrainVolume,matrix,missing-method (clusterCenters), 28
- ClusteredBrainVolume, 28, 29
- ClusteredBrainVolume (ClusteredBrainVolume-class), 28
- ClusteredBrainVolume-class, 28
- concat, 29
- concat,BrainVector,BrainVector-method (concat), 29
- concat,BrainVector,BrainVolume-method (concat), 29
- concat,BrainVolume,BrainVector-method (concat), 29
- concat,DenseBrainVolume,DenseBrainVolume-method (concat), 29
- concat,SparseBrainVector,SparseBrainVector-method (concat), 29
- connComp, 30
- connComp,BrainVolume-method (connComp), 30
- connComp3D, 31
- coords, 31
- coords,IndexLookupVolume-method (coords), 31
- coords,ROIVolume-method (coords), 31
- coords,SparseBrainVector-method (coords), 31
- coordToGrid, 32
- coordToGrid,BrainSpace,matrix-method (coordToGrid), 32
- coordToGrid,BrainSpace,numeric-method (coordToGrid), 32
- coordToGrid,BrainVolume,matrix-method (coordToGrid), 32
- coordToIndex, 32
- coordToIndex,BrainSpace,matrix-method (coordToIndex), 32
- coordToIndex,BrainSpace,numeric-method (coordToIndex), 32
- coordToIndex,BrainVolume,matrix-method (coordToIndex), 32
- dataFile, 33
- dataFile,BrainFileDescriptor,character-method (dataFile), 33
- dataFileMatches, 34
- dataFileMatches,BrainFileDescriptor,character-method (dataFileMatches), 34

- dataReader, 34
- dataReader, AFNIMetaInfo-method  
(dataReader), 34
- dataReader, NIFTIMetaInfo-method  
(dataReader), 34
- DenseBrainVector, 24, 35, 57
- DenseBrainVector  
(DenseBrainVector-class), 35
- DenseBrainVector-class, 35
- DenseBrainVolume, 8, 26, 35, 36, 55, 58
- DenseBrainVolume  
(DenseBrainVolume-class), 35
- DenseBrainVolume-class, 35
- dim, BrainData-method, 36
- dim, BrainSpace-method, 36
- dim, FileMetaInfo-method, 37
- dropDim, 37
- dropDim, AxisSet2D, missing-method  
(dropDim), 37
- dropDim, AxisSet2D, numeric-method  
(dropDim), 37
- dropDim, AxisSet3D, missing-method  
(dropDim), 37
- dropDim, AxisSet3D, numeric-method  
(dropDim), 37
- dropDim, BrainSpace, missing-method  
(dropDim), 37
- eachSeries, 38
- eachSeries, BrainVector, function, missing-method  
(eachSeries), 38
- eachSeries, SparseBrainVector, function, logical-method  
(eachSeries), 38
- eachSlice, 39
- eachSlice, BrainVolume, function, logical-method  
(eachSlice), 39
- eachSlice, BrainVolume, function, missing-method  
(eachSlice), 39
- eachVolume, 39
- eachVolume, BrainBucket, function, logical, ANY-method  
(eachVolume), 39
- eachVolume, BrainBucket, function, missing, missing-method  
(eachVolume), 39
- eachVolume, BrainVector, function, logical, ANY-method  
(eachVolume), 39
- eachVolume, BrainVector, function, missing, BrainVolume-method  
(eachVolume), 39
- eachVolume, BrainVector, function, missing, missing-method  
(eachVolume), 39
- eachVolume, SparseBrainVector, function, logical, missing-method  
(eachVolume), 39
- eachVolume, SparseBrainVector, function, missing, LogicalBrainVector-method  
(eachVolume), 39
- eachVolume, SparseBrainVector, function, missing, missing-method  
(eachVolume), 39
- fileMatches, 41
- fileMatches, BrainFileDescriptor, character-method  
(fileMatches), 41
- FileMetaInfo, 71
- FileMetaInfo-class, 42
- fill, 42
- fill, BrainVolume, list-method (fill), 42
- fill, BrainVolume, matrix-method (fill),  
42
- gridToCoord, 43
- gridToCoord, BrainSpace, matrix-method  
(gridToCoord), 43
- gridToCoord, BrainVolume, matrix-method  
(gridToCoord), 43
- gridToIndex, 44
- gridToIndex, BrainSlice, matrix-method  
(gridToIndex), 44
- gridToIndex, BrainSpace, matrix-method  
(gridToIndex), 44
- gridToIndex, BrainSpace, numeric-method  
(gridToIndex), 44
- gridToIndex, BrainVolume, matrix-method  
(gridToIndex), 44
- gridToIndex, BrainVolume, numeric-method  
(gridToIndex), 44
- headerFile, 44
- headerFile, BrainFileDescriptor, character-method  
(headerFile), 44
- headerFileMatches, 45
- headerFileMatches, BrainFileDescriptor, character-method  
(headerFileMatches), 45
- image, BrainVolume-method, 46
- image, BrainVolume-method  
(image, BrainVolume-method), 46
- image, Overlay-method  
(image, BrainVolume-method), 46
- IndexGrid, 46
- IndexLookupVolume, 87
- IndexLookupVolume  
(IndexLookupVolume-class), 47

IndexLookupVolume-class, [47](#)  
 indexToCoord, [47](#)  
 indexToCoord,BrainSpace,index-method  
     (indexToCoord), [47](#)  
 indexToCoord,BrainVolume,index-method  
     (indexToCoord), [47](#)  
 indexToGrid, [48](#)  
 indexToGrid,BrainSlice,index-method  
     (indexToGrid), [48](#)  
 indexToGrid,BrainSpace,index-method  
     (indexToGrid), [48](#)  
 indexToGrid,BrainVector,index-method  
     (indexToGrid), [48](#)  
 indexToGrid,BrainVolume,index-method  
     (indexToGrid), [48](#)  
 indices, [49](#)  
 indices,IndexLookupVolume-method  
     (indices), [49](#)  
 indices,ROIVolume-method (indices), [49](#)  
 indices,SparseBrainVector-method  
     (indices), [49](#)  
 inverseTrans, [50](#)  
 inverseTrans,BrainData-method  
     (inverseTrans), [50](#)  
 inverseTrans,BrainSpace-method  
     (inverseTrans), [50](#)  
  
 Kernel, [50](#)  
 Kernel-class, [51](#)  
  
 Layer, [51](#), [51](#)  
 Layer-class, [52](#)  
 length,BrainVector-method  
     (length,ROIVolume-method), [52](#)  
 length,ROIVolume-method, [52](#)  
 loadBucket, [53](#)  
 loadData, [53](#)  
 loadData,BrainBucketSource-method  
     (loadData), [53](#)  
 loadData,BrainVectorSource-method  
     (loadData), [53](#)  
 loadData,BrainVolumeSource-method  
     (loadData), [53](#)  
 loadData,SparseBrainVectorSource-method  
     (loadData), [53](#)  
 loadVector, [54](#)  
 loadVolume, [55](#)  
 loadVolumeList, [55](#)  
 LogicalBrainVolume, [25](#), [28](#), [56](#)

LogicalBrainVolume  
     (LogicalBrainVolume-class), [56](#)  
 LogicalBrainVolume-class, [56](#)  
 lookup, [56](#)  
 lookup,IndexLookupVolume,numeric-method  
     (lookup), [56](#)  
 lookup,SparseBrainVector,numeric-method  
     (lookup), [56](#)  
  
 makeVector, [57](#)  
 makeVolume, [57](#)  
 map, [58](#)  
 map,BrainVolume,Kernel-method (map), [58](#)  
 matchAnatomy2D, [59](#)  
 matchAnatomy3D, [59](#)  
 matrixToVolumeList, [60](#)  
 mergePartitions, [60](#)  
 mergePartitions,ClusteredBrainVolume,numeric,matrix-method  
     (mergePartitions), [60](#)  
 MNI\_SPACE\_1MM, [61](#)  
  
 NamedAxis-class, [61](#)  
 names,BrainBucket-method  
     (names,BrainBucketSource-method),  
     [62](#)  
 names,BrainBucketSource-method, [62](#)  
 ndim, [62](#)  
 ndim,AxisSet-method (ndim), [62](#)  
 ndim,BrainData-method (ndim), [62](#)  
 ndim,BrainSpace-method (ndim), [62](#)  
 neuroim, [63](#)  
 neuroim-package (neuroim), [63](#)  
 NIFTIFileDescriptor, [63](#)  
 NIFTIFileDescriptor-class, [63](#)  
 NIFTIMetaInfo, [63](#), [63](#)  
 NIFTIMetaInfo-class  
     (FileMetaInfo-class), [42](#)  
 NullMetaInfo-class, [64](#)  
 numClusters, [64](#)  
 numClusters,ClusteredBrainVolume-method  
     (numClusters), [64](#)  
  
 origin, [64](#)  
 origin,BrainData-method (origin), [64](#)  
 origin,BrainSpace-method (origin), [64](#)  
 overlay, [65](#)  
 overlay,Layer,Layer-method (overlay), [65](#)  
  
 partition, [66](#)

- partition, ClusteredBrainVolume, numeric, matrix-method (partition), 66
- permMat, 66
- permMat, AxisSet2D-method (permMat), 66
- pick, 67
- print, 67
- print, AxisSet2D-method, 68
- print, AxisSet3D-method, 68
- print, NamedAxis-method, 69
- RandomSearchlight, 69
- readAFNIHeader, 70
- readElements, 70
- readElements, BinaryReader, numeric-method (readElements), 70
- readHeader, 71
- readMetaInfo, 71
- readMetaInfo, AFNIFileDescriptor-method (readMetaInfo), 71
- readMetaInfo, NIFTIFileDescriptor-method (readMetaInfo), 71
- RegionCube, 72
- RegionSphere, 72
- RegionSquare, 73
- ROIVolume, 74
- ROIVolume-class, 74
- scaleSeries, 75
- scaleSeries, BrainVector, logical, logical-method (scaleSeries), 75
- scaleSeries, BrainVector, logical, missing-method (scaleSeries), 75
- scaleSeries, BrainVector, missing, logical-method (scaleSeries), 75
- scaleSeries, BrainVector, missing, missing-method (scaleSeries), 75
- Searchlight, 76
- series, 76
- series, BrainVector, matrix-method (series), 76
- series, BrainVector, numeric-method (series), 76
- series, SparseBrainVector, matrix-method (series), 76
- series, SparseBrainVector, numeric-method (series), 76
- seriesIter, 77
- seriesIter, BrainVector-method (seriesIter), 77
- seriesIter, SparseBrainVector-method (seriesIter), 77
- show, AxisSet1D-method, 78
- show, AxisSet2D-method, 78
- show, AxisSet3D-method, 78
- show, AxisSet4D-method, 79
- show, BaseMetaInfo-method, 79
- show, BrainSpace-method, 80
- show, BrainVector-method, 80
- show, BrainVectorSource-method, 81
- show, BrainVolume-method, 81
- show, FileMetaInfo-method, 82
- show, NamedAxis-method, 82
- show, NullMetaInfo-method, 83
- show, ROIVolume-method, 83
- show, SparseBrainVector-method, 84
- slice, 84
- slice, BrainVolume, numeric, numeric, character-method (slice), 84
- space, 85
- space, BrainData-method (space), 85
- space, BrainSpace-method (space), 85
- space, IndexLookupVolume-method (space), 85
- spacing, 85
- spacing, BrainData-method (spacing), 85
- spacing, BrainSpace-method (spacing), 85
- SparseBrainVector, 24, 47, 77, 87
- SparseBrainVector (SparseBrainVector-class), 86
- SparseBrainVector-class, 86
- SparseBrainVectorSource, 25
- SparseBrainVectorSource (SparseBrainVectorSource-class), 87
- SparseBrainVectorSource-class, 87
- SparseBrainVolume, 88
- SparseBrainVolume (SparseBrainVolume-class), 88
- SparseBrainVolume-class, 88
- splitFill, 89
- splitFill, BrainVolume, factor, function-method (splitFill), 89
- splitReduce, 90
- splitReduce, BrainVector, factor, function-method (splitReduce), 90
- splitReduce, BrainVector, factor, missing-method (splitReduce), 90

- splitReduce,matrix,factor,function-method  
(splitReduce), 90
- splitReduce,matrix,factor,missing-method  
(splitReduce), 90
- splitReduce,matrix,integer,function-method  
(splitReduce), 90
- splitReduce,matrix,integer,missing-method  
(splitReduce), 90
- splitScale, 91
- splitScale,matrix,factor,logical,logical-method  
(splitScale), 91
- splitScale,matrix,factor,missing,missing-method  
(splitScale), 91
- stripExtension, 92
- stripExtension,BrainFileDescriptor,character-method  
(stripExtension), 92
  
- takeSeries, 93
- takeVolume, 93
- takeVolume,BrainVector,numeric-method  
(takeVolume), 93
- takeVolume,SparseBrainVector,numeric-method  
(takeVolume), 93
- tessellate, 94
- tessellate,LogicalBrainVolume,numeric-method  
(tessellate), 94
- trans, 94
- trans,BrainData-method (trans), 94
- trans,BrainMetaInfo-method (trans), 94
- trans,BrainSpace-method (trans), 94
- trans,NIfTMetaInfo-method (trans), 94
  
- values, 95
- values,ROIVolume-method (values), 95
- voxels, 96
- voxels,Kernel-method (voxels), 96
  
- writeElements, 96
- writeElements,BinaryWriter,numeric-method  
(writeElements), 96
- writeVector, 97
- writeVector,BrainVector,character,character,missing-method  
(writeVector), 97
- writeVector,BrainVector,character,missing,character,ANY-method  
(writeVector), 97
- writeVector,BrainVector,character,missing,character-method  
(writeVector), 97
- writeVector,BrainVector,character,missing,missing-method  
(writeVector), 97