

# Package: neuroim (via r-universe)

November 6, 2024

**Imports** Matrix, rflann, Rcpp, iterators, abind, assertthat, mmap,  
igraph, plyr, RNifti

**LinkingTo** Rcpp

**License** GPL (>= 2)

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

**Type** Package

**Author** Bradley R. Buchsbaum

**Title** Data Structures and Handling for Neuroimaging Data

**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.

**Date** 2016-01-06

**Suggests** foreach, testthat, knitr, rgl, readr

**Version** 0.1.0

**Depends** R (>= 3.0.0), stringr, hash, methods, stats, grDevices, grid,  
utils

**VignetteBuilder** knitr

**Collate** 'AFNI\_IO.R' 'AllGeneric.R' 'AllClass.R' 'Axis.R'  
'BasisBrainVector.R' 'BinaryIO.R' 'BrainBucket.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' 'Searchlight.R' 'conncomp.R' 'datadoc.R'  
'neuroim.R' 'niml\_io.R'

**RoxygenNote** 6.0.1.9000

**Remotes** YeeJeremy/rflann

**Config/pak/sysreqs** libglpk-dev libicu-dev libxml2-dev

**Repository** <https://bbuchsbaum.r-universe.dev>

**RemoteUrl** <https://github.com/bbuchsbaum/neuroim>

**RemoteRef** HEAD

**RemoteSha** a0c2c5db6e717eeacf7ad3cb9be65a48a34d1f93

## Contents

addDim . . . . .	6
AFNIFileDescriptor-class . . . . .	6
AFNIMetaInfo . . . . .	7
as . . . . .	7
as.array,BrainData-method . . . . .	8
as.list,SparseBrainVector-method . . . . .	8
as.logical,BrainVolume-method . . . . .	9
as.mask . . . . .	9
as.matrix,BrainData-method . . . . .	10
as.matrix,ROIVector-method . . . . .	10
as.numeric,SparseBrainVolume-method . . . . .	11
as.sparse . . . . .	11
as.vector,BrainData-method . . . . .	12
axes . . . . .	12
AxisSet-class . . . . .	13
AxisSet1D-class . . . . .	13
AxisSet2D-class . . . . .	13
AxisSet3D-class . . . . .	14
AxisSet4D-class . . . . .	14
AxisSet5D-class . . . . .	14
axisToIndex . . . . .	15
Base-class . . . . .	15
BaseMetaInfo-class . . . . .	16
BaseSource-class . . . . .	16
BasisBrainVector-class . . . . .	16
BinaryReader . . . . .	17
BinaryReader-class . . . . .	17
BinaryWriter-class . . . . .	18
BootstrapSearchlight . . . . .	18
bounds . . . . .	19
BrainBucket-class . . . . .	20
BrainBucketSource-class . . . . .	20
BrainData-class . . . . .	21
BrainFileDescriptor-class . . . . .	21
BrainFileSource-class . . . . .	22
BrainMetaInfo-class . . . . .	22
BrainSlice . . . . .	23
BrainSlice-class . . . . .	23
BrainSource-class . . . . .	24
BrainSpace . . . . .	24
BrainSpace-class . . . . .	25

BrainVector-class	25
BrainVectorSource	26
BrainVectorSource-class	26
BrainVolume	27
BrainVolume-class	27
BrainVolumeSource-class	28
close,BinaryReader-method	28
ClusteredBrainVolume-class	29
ClusteredSearchlight	30
ColumnReader	30
ColumnReader-class	31
concat	31
connComp	32
connComp3D	33
coords	34
coordToGrid	34
coordToIndex	35
dataFile	36
dataFileMatches	36
dataReader	37
DenseBrainVector-class	37
DenseBrainVolume-class	38
dim,BrainData-method	38
dim,BrainSpace-method	39
dim,FileMetaInfo-method	39
dropDim	40
eachSeries	41
eachSlice	42
eachVolume	42
fileMatches	44
FileMetaInfo-class	44
fill	45
gridToCoord	46
gridToGrid	46
gridToIndex	47
headerFile	48
headerFileMatches	48
image,BrainVolume-method	49
IndexLookupVolume-class	50
indexToCoord	50
indexToGrid	51
indices	52
inverseTrans	52
Kernel	53
Kernel-class	54
length,ROIVolume-method	54
loadBucket	55
loadData	55

loadVector . . . . .	56
loadVolume . . . . .	57
loadVolumeList . . . . .	57
LogicalBrainVolume-class . . . . .	58
lookup . . . . .	58
makeVector . . . . .	59
makeVolume . . . . .	59
map . . . . .	60
mapToColors . . . . .	61
matchAnatomy2D . . . . .	61
matchAnatomy3D . . . . .	62
matrixToVolumeList . . . . .	62
MNI_SPACE_1MM . . . . .	63
NamedAxis-class . . . . .	63
names,BrainBucketSource-method . . . . .	63
ndim . . . . .	64
neuroim . . . . .	64
NIFTIFileDescriptor-class . . . . .	65
NIFTIMetaInfo . . . . .	65
NullMetaInfo-class . . . . .	65
numClusters . . . . .	66
origin . . . . .	66
overlay . . . . .	67
patchSet . . . . .	67
permMat . . . . .	68
print . . . . .	68
print,AxisSet2D-method . . . . .	69
print,AxisSet3D-method . . . . .	69
print,NamedAxis-method . . . . .	70
RandomSearchlight . . . . .	70
readAFNIHeader . . . . .	71
readColumns . . . . .	71
readElements . . . . .	72
readHeader . . . . .	72
readMetaInfo . . . . .	73
RegionCube . . . . .	73
RegionSphere . . . . .	74
RegionSquare . . . . .	75
render . . . . .	75
renderSlice . . . . .	76
reorient . . . . .	77
ROI-class . . . . .	77
ROIVector . . . . .	77
ROIVector-class . . . . .	78
ROIVolume . . . . .	78
ROIVolume-class . . . . .	79
scaleSeries . . . . .	79
Searchlight . . . . .	80

seriesIter . . . . .	80
series_roi . . . . .	81
show,AxisSet1D-method . . . . .	83
show,AxisSet2D-method . . . . .	83
show,AxisSet3D-method . . . . .	83
show,AxisSet4D-method . . . . .	84
show,BaseMetaInfo-method . . . . .	84
show,BrainSpace-method . . . . .	85
show,BrainVector-method . . . . .	85
show,BrainVectorSource-method . . . . .	86
show,BrainVolume-method . . . . .	86
show,FileMetaInfo-method . . . . .	87
show,NamedAxis-method . . . . .	87
show,NullMetaInfo-method . . . . .	88
show,ROIVolume-method . . . . .	88
show,SparseBrainVector-method . . . . .	89
slice . . . . .	89
sliceData . . . . .	90
space . . . . .	90
spacing . . . . .	91
SparseBrainVector-class . . . . .	92
SparseBrainVectorSource-class . . . . .	93
SparseBrainVolume-class . . . . .	93
splitFill . . . . .	94
splitReduce . . . . .	95
splitScale . . . . .	96
stripExtension . . . . .	98
subVector . . . . .	98
takeVolume . . . . .	99
trans . . . . .	100
values . . . . .	101
voxels . . . . .	101
writeElements . . . . .	102
writeVector . . . . .	102
writeVolume . . . . .	103
[,BrainBucket,index,missing,ANY-method . . . . .	104
[,ROIVolume,numeric,missing,ANY-method . . . . .	105
[,SparseBrainVector,missing,missing,ANY-method . . . . .	105
[,SparseBrainVector,missing,numeric,ANY-method . . . . .	106
[,SparseBrainVector,numeric,missing,ANY-method . . . . .	106
[,SparseBrainVector,numeric,numeric,ANY-method . . . . .	107
[,SparseBrainVolume,matrix,missing,ANY-method . . . . .	108
[,SparseBrainVolume,missing,missing,ANY-method . . . . .	108
[,SparseBrainVolume,missing,numeric,ANY-method . . . . .	109
[,SparseBrainVolume,numeric,missing,missing-method . . . . .	109
[,SparseBrainVolume,numeric,numeric,ANY-method . . . . .	110
[[,BrainBucket,index,missing-method . . . . .	110
[[,SparseBrainVector,numeric,ANY-method . . . . .	111

---

addDim	<i>Generic function to add a dimension to an object</i>
--------	---

---

### 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
<i>AFNIFileDescriptor</i>

---

### Description

This class supports the AFNI file format

---

AFNIMetaInfo

*AFNIMetaInfo*

---

### Description

Constructor for [AFNIMetaInfo](#) class

### Usage

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

### Arguments

descriptor      an instance of class [AFNIFileDescriptor](#)  
afni\_header     a list returned by `readAFNIHeader`

### Value

an instance of class [AFNIMetaInfo](#)

---

as

*conversion from 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, SparseBrainVector-method`  
*as.list*

---

**Description**

convert SparseBrainVector to list of [DenseBrainVolume](#)

**Usage**

```
## S4 method for signature 'SparseBrainVector'  
as.list(x)  
  
## S4 method for signature 'BrainVector'  
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                    *Convert to a LogicalBrainVolume*

---

### Description

Convert to a LogicalBrainVolume

### Usage

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

### Arguments

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

---

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

---

**Description**

convert BrainData instance to matrix

**Usage**

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

**Arguments**

x                    the object

---

as.matrix,ROIVector-method  
*convert a ROIVector to a matrix*

---

**Description**

convert SparseBrainVector to matrix

**Usage**

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

**Arguments**

x                    the object  
x                    the object  
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.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 1D 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-method

*convert BrainData instance to vector*

---

**Description**

convert BrainData instance to vector

**Usage**

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

**Arguments**

x                    the object

---

axes

*Generic getter function to extract image axes*

---

**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            *Base*

---

**Description**

Virtual base class representing an ordered set of named axes.

**Slots**

ndim the number of axes (or dimensions)

---

AxisSet1D-class            *AxisSet1D*

---

**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>Convert 1-dimensional real axis coordinates along a single axis dimension to a 1D index along the same axis.</i>
-------------	---

---

**Description**

Convert 1-dimensional real axis coordinates along a single axis dimension to a 1D index along the same axis.

Convert 1-dimensional index to a coordinate value along the same axis.

**Usage**

```
axisToIndex(x, real, dimNum)
```

```
indexToAxis(x, index, dimNum)
```

```
## S4 method for signature 'BrainSpace,numeric,numeric'
```

```
axisToIndex(x, real, dimNum)
```

```
## S4 method for signature 'BrainSpace,numeric,numeric'
```

```
indexToAxis(x, index, dimNum)
```

**Arguments**

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

**Value**

a vector of axis indices

a vector of axis indices

---

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

---

**Description**

Generic S4 Base class

---

BaseMetaInfo-class      *BaseMetaInfo*

---

**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

---

BasisBrainVector-class  
*BasisBrainVector*

---

**Description**

a class that stores a represents a 4-dimensional array as a set of basis functions (dictionary) and corresponding set of coefficients.

**Slots**

mask the mask defining the sparse domain

basis the matrix of bases, were each column is a basis vector.

coeffs the sparseMatrix of coefficients

map instance of class [IndexLookupVolume](#) is used to map between spatial and index/row coordinates



---

 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     *Create a spherical searchlight iterator that samples regions from within a mask.*

---

### Description

searchlight centers are sampled without replacement, but the same surround voxel can belong to multiple searchlight samples.

### Usage

```
BootstrapSearchlight(mask, radius = 8, iter = 100)
```

**Arguments**

mask	an image volume containing valid central voxels for roving searchlight
radius	in mm of spherical searchlight (can be a vector which is randomly sampled)
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* Base class for representing a data source for images. The purpose of this class is to provide a layer in between low level IO and image loading functionality.

---

### Description

BrainFileSource

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* This class contains meta information from an image

---

### Description

BrainMetaInfo

This class contains meta information from an image

This class contains meta information for 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	<i>BrainSlice constructor</i>
------------	-------------------------------

---

**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 used if data is a 1D vector.

**Examples**

```

ospace <- BrainSpace(c(64,64), spacing=c(1,1))
dat <- array(rnorm(64*64), c(64,64))
bslice <- BrainSlice(dat,ospace)
print(bslice)

```

---

BrainSlice-class	<i>BrainSlice</i>
------------------	-------------------

---

**Description**

Two-dimensional brain image

---

BrainSource-class	<i>BrainSource</i>
-------------------	--------------------

---

**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	<i>Constructor function for <a href="#">BrainSpace</a> class</i>
------------	--

---

**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 in the untransformed native grid space.  
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 <a href="#">BrainSpace</a> 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 <a href="#">BrainSource</a> object
label	a label of type character

**Value**

a concrete instance of [BrainVector](#) class. If mask is provided then [SparseBrainVector](#), otherwise [DenseBrainVector](#)

---

BrainVectorSource	<i>BrainVectorSource</i>
-------------------	--------------------------

---

**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	<i>BrainVectorSource</i>
-------------------------	--------------------------

---

**Description**

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

**Slots**

indices the index vector of the volumes to be loaded

---

BrainVolume	<i>BrainVolume</i>
-------------	--------------------

---

**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	<i>BrainVolume Base class for image representing 3D volumetric data.</i>
-------------------	--

---

**Description**

BrainVolume Base class for image representing 3D volumetric data.

BrainVolumeSource-class

*BrainVolumeSource* A class is used to produce a [BrainVolume](#) instance

---

### Description

BrainVolumeSource

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

---

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 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

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

ClusteredSearchlight    *Create a clustered Searchlight iterator*

---

**Description**

Create a clustered Searchlight iterator

**Usage**

```
ClusteredSearchlight(mask, csize)
```

**Arguments**

mask	an image volume containing valid central voxels for roving searchlight
csize	the number of clusters

**Value**

an iter class

---

ColumnReader    *ColumnReader*

---

**Description**

Constructor for [ColumnReader](#) class

**Usage**

```
ColumnReader(nrow, ncol, reader)
```

**Arguments**

nrow	the number of rows
ncol	the number of columns
reader	a function that takes a set of column indices and returns a matrix

---

ColumnReader-class	<i>ColumnReader</i>
--------------------	---------------------

---

**Description**

This class supports reading of data from a matrix-like storage format

**Slots**

nrow the number of rows

ncol the number of columns

reader a function that takes a set of column indices and returns a matrix

---

concat	<i>Concatenate two objects</i>
--------	--------------------------------

---

**Description**

Concatenate two objects

**Usage**

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

```
## S4 method for signature 'SparseBrainVector,missing'
```

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

```
## S4 method for signature 'SparseBrainVector,SparseBrainVector'
```

```
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,missing'
```

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

```
## S4 method for signature 'DenseBrainVolume,DenseBrainVolume'
```

```
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
...	additional 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, connect = c("6-connect", "18-connect", "26-connect"))
```

**Arguments**

mask	a 3D binary array
connect	the connectiivty constraint: "6-connect", "18-connect", or "26-connect"

**Value**

a two-element list of the connected components (cluster index and cluster size) The first element index is a 3D array containing the cluster index of the connected component for each voxel. The second element size is a 3D array consisting of the size of the connected component inhabited by each voxel.

**Examples**

```
dat <- array(as.logical(rnorm(10*10*10)>.5), c(10, 10, 10))
res1 <- connComp3D(dat, connect="6-connect")
res2 <- connComp3D(dat, connect="18-connect")
res3 <- connComp3D(dat, connect="26-connect")
```

---

coords	<i>Extract coordinates</i>
--------	----------------------------

---

**Description**

Extract coordinates  
 coords

**Usage**

```
coords(x, ...)
```

```
## S4 method for signature 'ROIVolume'
```

```
coords(x, real = FALSE)
```

```
## S4 method for signature 'SparseBrainVector'
```

```
coords(x, i)
```

```
## S4 method for signature 'IndexLookupVolume'
```

```
coords(x, i)
```

**Arguments**

x	the object to extract coordinates from
...	additional arguments
real	if TRUE, return coordinates in real world units
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 'NIFTIMetaInfo'
dataReader(x, offset = 0)

## S4 method for signature 'AFNIMetaInfo'
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	<i>DenseBrainVector</i>
------------------------	-------------------------

---

**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,numeric'
dropDim(x, dimnum)

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

**Arguments**

<code>x</code>	a dimensioned object
<code>dimnum</code>	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	<i>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.</i>
------------	--

---

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

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

## S4 method for signature 'BrainVector`,`function`,`missing'
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

**Usage**

```
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, ...)

## 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, ...)

```

### 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 volume 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	<i>FileMetaInfo</i>
--------------------	---------------------

---

**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  
 afni\_header a list of attributes specific to the AFNI file format

---

fill	<i>Generic function to map values from one set to another using a user-supplied lookup table</i>
------	--

---

## 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 coordinate coordinates to real world coordinates</i>
-------------	--

---

### Description

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

### Usage

```

gridToCoord(x, coords)

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

## S4 method for signature 'BrainSpace,numeric'
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

---

gridToGrid	<i>Generic function to convert voxel coordinates in the reference space (LPI) to native array space.</i>
------------	--

---

### Description

Generic function to convert voxel coordinates in the reference space (LPI) to native array space.

**Usage**

```

gridToGrid(x, vox)

## S4 method for signature 'BrainSpace,matrix'
gridToGrid(x, vox)

## S4 method for signature 'matrix,matrix'
gridToGrid(x, vox)

```

**Arguments**

x	the object
coords	a matrix of LPI voxel coordinates

**Value**

a matrix of native voxel 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

**Usage**

```
## S4 method for signature 'BrainVolume'
image(x, slice = dim(vol)[3]/2,
      col = gray((0:255)/255, alpha = 1), zero_col = "#000000", axis = 3, ...)
```

**Arguments**

slice                the voxel index of the slice to display  
 col                  a color map  
 zero\_col            the color to use when the value is 0 (e.g background color)  
 ...                  extra arguments to be passed to grid.raster

---

 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.

### 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,numeric'
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,numeric'
indexToGrid(x, idx)

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

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

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

## S4 method for signature 'BrainVolume,integer'
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 'ROIVector'
indices(x)

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

## S4 method for signature 'IndexLookupVolume'
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 from a function of distance from kernel center</i>
--------	--

---

**Description**

Create a Kernel object from a function of distance from kernel center

**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

---

length,ROIVolume-method

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

---

### Description

Get length of BrainVector. This is the number of 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)
```

---

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

Load data from a [BrainBucketSource](#)

loadData

load a BrainVolume

**Usage**

```
loadData(x, ...)
```

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

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

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

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

**Arguments**

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

**Value**

an instance of class [BrainVolume](#)

an instance of class [BrainVector](#)

---

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

---

**Description**

load an image volume from a file

**Usage**

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

**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
mmap	memory mapping if possible

**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>loadVolumeList</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 'SparseBrainVector,numeric'
```

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

```
## S4 method for signature 'IndexLookupVolume,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 masked area

---

mapToColors	<i>mapToColors</i>
-------------	--------------------

---

**Description**

map an matrix of intensity values to a matrix of color values.

**Usage**

```
mapToColors(imslice, col = heat.colors(128, alpha = 1),
  zero_col = "#00000000", alpha = 1, irange = range(imslice),
  threshold = c(0, 0))
```

**Arguments**

imslice	vector or matrix of intensity values
col	a color map
zero_col	the background color.
alpha	transparency multiplier

---

matchAnatomy2D	<i>given two named axes return AxisSet2D singleton</i>
----------------	--

---

**Description**

given two named axes return AxisSet2D singleton

**Usage**

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

**Arguments**

axis1	the first axis
axis2	the second axis

---

matchAnatomy3D	<i>given three named axes return AxisSet3D singleton</i>
----------------	--

---

**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

---

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	<i>names</i>
--------------------------------	--------------

---

**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.

**Details**

none



---

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	<i>numClusters</i>
-------------	--------------------

---

**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	<i>Generic getter to extract image origin</i>
--------	---

---

**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**

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

---

overlay	<i>overlay two objects</i>
---------	----------------------------

---

**Description**

overlay two objects

**Usage**

overlay(x, y, ...)

**Arguments**

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

---

patchSet	<i>patchSet</i>
----------	-----------------

---

**Description**

patchSet

**Usage**

patchSet(x, dims, mask, ...)

**Arguments**

x	the object to extract patches from
dims	a vector indicating the dimensions of the patches
mask	mask indicating the valid patch area

---

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

---

**Description**

Extract permutation matrix

permMat

permMat

permMat

**Usage**

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

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

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

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

**Arguments**

x	the object
...	additional 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 spherical random searchlight iterator
```

---

**Description**

Create an spherical random searchlight iterator

**Usage**

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

**Arguments**

mask	an volumetric image mask of type <a href="#">BrainVolume</a> containing valid searchlight voxel set.
radius	width 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)

---

readColumns	<i>Generic function to read a set of column vector from an input source (e.g. ColumnReader)</i>
-------------	---

---

**Description**

Generic function to read a set of column vector from an input source (e.g. ColumnReader)

**Usage**

readColumns(x, columnIndices)

```
## S4 method for signature 'ColumnReader,numeric'
readColumns(x, columnIndices)
```

**Arguments**

x                    the input channel  
columnIndices    the column indices

**Value**

a matrix consisting of the requested column vectors

---

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	<i>Create A Cuboid Region of Interest</i>
------------	---

---

**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 <i>voxel</i> coordinates
surround	the number of voxels on either side of the central voxel. A vector of length 3.
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**

Creates a Spherical ROI based on a Centroid.

**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 store as data
nonzero	if TRUE, keep only nonzero elements from bvol

**Value**

an instance of class ROIVolume

**Examples**

```
sp1 <- BrainSpace(c(10,10,10), c(1,2,3))
cube <- RegionSphere(sp1, c(5,5,5), 3.5)
vox <- coords(cube)
cds <- coords(cube, real=TRUE)
## fill in ROI with value of 6
cube1 <- RegionSphere(sp1, c(5,5,5), 3.5, fill=6)
all(cube1@data == 6)
```

---

RegionSquare	<i>Create a square region of interest where the z-dimension is fixed at one voxel coordinate.</i>
--------------	---

---

**Description**

Create a square region of interest where the z-dimension is fixed at one voxel 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 <i>voxel</i> coordinates.
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
```

---

render	<i>Render an image to create a drawable image.</i>
--------	--

---

**Description**

Render an image to create a drawable image.

**Usage**

```
render(x, width, height, colmap, ...)

## S4 method for signature 'BrainSlice,numeric,numeric,character'
render(x, width, height,
       colmap, zero_col = "#00000000", alpha = 1, units = "points",
       irange = range(x), threshold = c(0, 0))
```

**Arguments**

x	the object, e.g. an instance of type BrainSlice
width	width of the rendered image
height	height of the rendered image
colmap	the colors used to map from values to RGBA colors.
...	additional arguments
zero_col	color used when background intensity is 0.
alpha	transparency multiplier
units	grid unit type, e.g. "points", "mm", "inches", "npc"

---

renderSlice	<i>Render a slice at z coordinate</i>
-------------	---------------------------------------

---

**Description**

Render a slice at z coordinate

**Usage**

```
renderSlice(x, zpos, width, height, colmap, ...)
```

**Arguments**

x	the object, e.g. an instance of type Layer or Overlay
zpos	the z coordinate to slice through.
width	width of the rendered image
height	height of the rendered image
colmap	the colors used to map from values to RGBA colors.
...	additional arguments

---

reorient	<i>Generic function to remap the grid-to-world coordinates mapping of an image.</i>
----------	---

---

**Description**

Generic function to remap the grid-to-world coordinates mapping of an image.

**Usage**

```
reorient(x, orient)
```

```
## S4 method for signature 'BrainSpace,character'
reorient(x, orient)
```

**Arguments**

x	the object
orient	the orientation code indicating the "remapped" axes.

**Value**

a reoriented space

---

ROI-class	<i>ROI</i>
-----------	------------

---

**Description**

Base marker class for a region of interest (ROI)

---

ROIVector	<i>Create an instance of class <a href="#">ROIVector</a></i>
-----------	--

---

**Description**

Create an instance of class [ROIVector](#)

**Usage**

```
ROIVector(vspace, coords, data = rep(nrow(coords), 1))
```

**Arguments**

vspace	an instance of class BrainSpace
coords	matrix of voxel coordinates
data	the matrix of data values

**Value**

an instance of class ROIVector

---

ROIVector-class	<i>ROIVector</i>
-----------------	------------------

---

**Description**

A class that represents a vector-valued volumetric region of interest

**Slots**

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

---

ROIVolume	Create an instance of class <a href="#">ROIVolume</a>
-----------	---

---

**Description**

Create an instance of class [ROIVolume](#)

**Usage**

```
ROIVolume(vspace, coords, data = rep(nrow(coords), 1))
```

**Arguments**

coords	matrix of voxel coordinates
data	the data values, numeric vector
vspace	an instance of class BrainSpace

**Value**

an instance of class ROIVolume

---

ROIVolume-class	<i>ROIVolume</i>
-----------------	------------------

---

**Description**

A class that represents a volumetric region of interest

**Slots**

data the numeric data stored in 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
scale	a logical value indicating whether series should be divided by standard deviation

**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, eager = FALSE)
```

**Arguments**

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

**Value**

an iter class

---

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

---

**Description**

Construct a series iterator

**Usage**

```
seriesIter(x)

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

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

**Arguments**

x	the object to be iterated over. This is typically an instance of class <a href="#">BrainVector</a>
---	--



**Value**

an iter object from the iterators package.

**Methods (by class)**

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

---

series\_roi

---

*Extract one or more series from object and return as ROI object*


---

**Description**

Extract one or more series from object and return as ROI object

Extract one or more series from object

**Usage**

```
series_roi(x, i, ...)
```

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

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

```
## S4 method for signature 'BasisBrainVector,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)

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

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

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

## S4 method for signature 'BrainVector,ROIVolume'
series_roi(x, i)

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

## S4 method for signature 'BrainVector,LogicalBrainVolume'
series_roi(x, i)

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

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

```

### Arguments

x	the object
i	the series indices
...	additional arguments
j	index for 2nd dimension
k	index for 3rd dimension
x	the object
i	the series indices
...	additional arguments
j	index for 2nd dimension
k	index for 3rd dimension

---

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,missing'
slice(x, zlevel, along,
      orientation)

## S4 method for signature 'BrainVolume,numeric,BrainSpace,AxisSet3D'
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

---

sliceData	<i>sliceData</i>
-----------	------------------

---

**Description**

extract a 2D slice from a BrainVolume instance.

**Usage**

```
sliceData(vol, slice, axis = 3)
```

**Arguments**

vol	an BrainVolume instance
slice	the integer index of the slice to cut.
axis	the axis number (1, 2, 3) defining fixed axis of the 2D slice.

---

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 <a href="#">BrainVolume</a> or <a href="#">BrainVector</a>
...	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

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

---

**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

<code>data</code>	an array which can be a <code>matrix</code> or 4-D array
<code>space</code>	a <code>BrainSpace</code> instance
<code>mask</code>	a 3D array of type <code>logical</code>
<code>source</code>	the data source – an instance of class <code>BrainSource</code>
<code>label</code>	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

```
bspace <- 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 <a href="#">BrainMetaInfo</a>
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

**Details**

Image data is backed by `Matrix::sparseVector`.

**Value**

`SparseBrainVolume` instance

**Slots**

data a `sparseVector` instance

**Examples**

```
data <- 1:10
indices <- seq(1,1000, length.out=10)
bspace <- BrainSpace(c(64,64,64), spacing=c(1,1,1))
sparsevol <- SparseBrainVolume(data,bspace,indices=indices)
densevol <- BrainVolume(data,bspace,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" by the supplied function.

**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)

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

## S4 method for signature 'DenseBrainVector,factor,logical,missing'
splitScale(x, f, center)

## S4 method for signature 'DenseBrainVector,factor,logical,logical'
splitScale(x, f, center,
           scale)
```

**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**

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

**Value**

fileName without extension

---

subVector	<i>Generic function to extract a sub-vector from a <a href="#">BrainVector</a> object.</i>
-----------	--

---

**Description**

Generic function to extract a sub-vector from a [BrainVector](#) object.

**Usage**

```
subVector(x, i, ...)
```

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

```
## S4 method for signature 'DenseBrainVector,numeric'
subVector(x, i)
```

**Arguments**

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

**Value**

a BrainVector object that is a sub-sequence of the supplied object.

**Examples**

```
bvec <- BrainVector(array(rnorm(24*24*24*24), c(24,24,24,24)), BrainSpace(c(24,24,24,24), c(1,1,1)))
vec <- subVector(bvec,1:2)
all.equal(2, dim(vec)[4])

vec <- subVector(bvec, c(1,3,5,7))
all.equal(4, dim(vec)[4])

mask <- LogicalBrainVolume(rep(TRUE, 24*24*24), BrainSpace(c(24,24,24), c(1,1,1)))
svec <- SparseBrainVector(array(rnorm(24*24*24*24), c(24,24,24,24)),
BrainSpace(c(24,24,24,24), c(1,1,1)), mask)
vec <- subVector(svec, c(1,3,5))
all.equal(3, dim(vec)[4])
```

---

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 'SparseBrainVector,numeric'
takeVolume(x, i, merge = FALSE)

## S4 method for signature 'BrainVector,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"

```

---

trans

*Generic getter to extract image coordinate transformation*


---

**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.

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	<i>Generic function to extract data values of object</i>
--------	--

---

**Description**

Generic function to extract data values of object

**Usage**

```
values(x, ...)
```

```
## S4 method for signature 'ROIVolume'
```

```
values(x, ...)
```

```
## S4 method for signature 'ROIVector'
```

```
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	<i>Generic function to write a 4D image vector to disk</i>
-------------	--

---

**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(bvol, "out.nii")
writeVector(bvol, "out.nii.gz")
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.

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
      subset an ROIVolume
```

---

**Description**

subset an ROIVolume

**Usage**

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

```
## S4 method for signature 'ROIVolume,logical,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

---

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

---

**Description**

extractor  
extractor

**Usage**

```
## S4 method for signature 'SparseBrainVector,numeric,ANY'  
x[[i]]  
  
## S4 method for signature 'BrainVector,numeric,ANY'  
x[[i]]
```

**Arguments**

i            the volume index  
i            the volume index

# Index

\* **datasets**  
     MNI\_SPACE\_1MM, [63](#)  
 [,BrainBucket,index,missing,ANY-method,  
     [104](#)  
 [,ROIVolume,logical,missing,ANY-method  
     ([,ROIVolume,numeric,missing,ANY-method),  
     [105](#)  
 [,ROIVolume,numeric,missing,ANY-method,  
     [105](#)  
 [,SparseBrainVector,missing,missing,ANY-method,  
     [105](#)  
 [,SparseBrainVector,missing,numeric,ANY-method,  
     [106](#)  
 [,SparseBrainVector,numeric,missing,ANY-method,  
     [106](#)  
 [,SparseBrainVector,numeric,numeric,ANY-method,  
     [107](#)  
 [,SparseBrainVolume,matrix,missing,ANY-method,  
     [108](#)  
 [,SparseBrainVolume,missing,missing,ANY-method,  
     [108](#)  
 [,SparseBrainVolume,missing,numeric,ANY-method,  
     [109](#)  
 [,SparseBrainVolume,numeric,missing,missing-method,  
     [109](#)  
 [,SparseBrainVolume,numeric,numeric,ANY-method,  
     [110](#)  
 [[,BrainBucket,index,missing-method,  
     [110](#)  
 [[,BrainVector,numeric,ANY-method  
     ([[,SparseBrainVector,numeric,ANY-method),  
     [111](#)  
 [[,SparseBrainVector,numeric,ANY-method,  
     [111](#)  
  
 addDim, [6](#)  
 addDim,BrainSpace,numeric-method  
     (addDim), [6](#)  
 AFNIFileDescriptor, [7](#)  
 AFNIFileDescriptor-class, [6](#)  
  
 AFNIMetaInfo, [7, 7](#)  
 AFNIMetaInfo-class  
     (FileMetaInfo-class), [44](#)  
     as, [7](#)  
     as.array,BrainData-method, [8](#)  
     as.list,BrainVector-method  
         (as.list,SparseBrainVector-method),  
         [8](#)  
     as.list,SparseBrainVector-method, [8](#)  
     as.logical,BrainVolume-method, [9](#)  
     as.mask, [9](#)  
     as.mask,BrainVolume,missing-method  
         (as.mask), [9](#)  
     as.mask,BrainVolume,numeric-method  
         (as.mask), [9](#)  
     as.matrix,BrainData-method, [10](#)  
     as.matrix,DenseBrainVector-method  
         (as.matrix,ROIVector-method),  
         [10](#)  
     as.matrix,ROIVector-method, [10](#)  
     as.matrix,SparseBrainVector-method  
         (as.matrix,ROIVector-method),  
         [10](#)  
     as.numeric,SparseBrainVolume-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-method, [12](#)  
     axes, [12](#)  
     axes,BrainData-method (axes), [12](#)  
     axes,BrainSpace-method (axes), [12](#)  
     AxisSet-class, [13](#)



- AxisSet1D-class, 13
- AxisSet2D-class, 13
- AxisSet3D-class, 14
- AxisSet4D-class, 14
- AxisSet5D-class, 14
- axisToIndex, 15
- axisToIndex,BrainSpace,numeric,numeric-method  
(axisToIndex), 15
  
- Base-class, 15
- BaseMetaInfo-class, 16
- BaseSource, 21
- BaseSource-class, 16
- BasisBrainVector-class, 16
- BinaryReader, 17, 17
- BinaryReader-class, 17
- BinaryWriter, 18
- BinaryWriter (BinaryWriter-class), 18
- BinaryWriter-class, 18
- BootstrapSearchlight, 18
- bounds, 19
- bounds,BrainData-method (bounds), 19
- bounds,BrainSpace-method (bounds), 19
- BrainBucket, 20
- BrainBucket (BrainBucket-class), 20
- BrainBucket-class, 20
- BrainBucketSource, 20, 55
- BrainBucketSource  
(BrainBucketSource-class), 20
- BrainBucketSource-class, 20
- BrainData-class, 21
- BrainFileDescriptor, 36, 44, 48, 73, 98
- BrainFileDescriptor-class, 21
- BrainFileSource-class, 22
- BrainMetaInfo, 22, 93
- BrainMetaInfo (BrainMetaInfo-class), 22
- BrainMetaInfo-class, 22
- BrainSlice, 23
- BrainSlice-class, 23
- BrainSource, 25, 27, 29, 37, 38, 58–60, 92, 93
- BrainSource-class, 24
- BrainSpace, 6, 21, 24, 24, 25, 27, 37, 38, 58,  
90, 93, 100
- BrainSpace-class, 25
- BrainVector, 25, 26, 56, 57, 59, 80, 81, 90, 96
- BrainVector (BrainVector-class), 25
- BrainVector-class, 25
- BrainVectorSource, 26, 26
- BrainVectorSource-class, 26
  
- BrainVolume, 20, 26, 27, 27, 28, 56, 59, 60,  
70, 90
- BrainVolume-class, 27
- BrainVolumeSource  
(BrainVolumeSource-class), 28
- BrainVolumeSource-class, 28
  
- close,BinaryReader-method, 28
- close,BinaryWriter-method  
(close,BinaryReader-method), 28
- ClusteredBrainVolume, 29
- ClusteredBrainVolume  
(ClusteredBrainVolume-class),  
29
- ClusteredBrainVolume-class, 29
- ClusteredSearchlight, 30
- ColumnReader, 30, 30
- ColumnReader-class, 31
- concat, 31
- concat,BrainVector,BrainVector-method  
(concat), 31
- concat,BrainVector,BrainVolume-method  
(concat), 31
- concat,BrainVolume,BrainVector-method  
(concat), 31
- concat,DenseBrainVolume,DenseBrainVolume-method  
(concat), 31
- concat,DenseBrainVolume,missing-method  
(concat), 31
- concat,SparseBrainVector,missing-method  
(concat), 31
- concat,SparseBrainVector,SparseBrainVector-method  
(concat), 31
- connComp, 32
- connComp,BrainVolume-method (connComp),  
32
- connComp3D, 33
- coords, 34
- coords,IndexLookupVolume-method  
(coords), 34
- coords,ROIVolume-method (coords), 34
- coords,SparseBrainVector-method  
(coords), 34
- coordToGrid, 34
- coordToGrid,BrainSpace,matrix-method  
(coordToGrid), 34
- coordToGrid,BrainSpace,numeric-method  
(coordToGrid), 34

- coordToGrid,BrainVolume,matrix-method  
(coordToGrid), 34
- coordToIndex, 35
- coordToIndex,BrainSpace,matrix-method  
(coordToIndex), 35
- coordToIndex,BrainSpace,numeric-method  
(coordToIndex), 35
- coordToIndex,BrainVolume,matrix-method  
(coordToIndex), 35
- dataFile, 36
- dataFile,BrainFileDescriptor,character-method  
(dataFile), 36
- dataFileMatches, 36
- dataFileMatches,BrainFileDescriptor,character-method  
(dataFileMatches), 36
- dataReader, 37
- dataReader,AFNIMetaInfo-method  
(dataReader), 37
- dataReader,NIFTIMetaInfo-method  
(dataReader), 37
- DenseBrainVector, 26, 37, 38, 59
- DenseBrainVector  
(DenseBrainVector-class), 37
- DenseBrainVector-class, 37
- DenseBrainVolume, 8, 27, 38, 57, 60
- DenseBrainVolume  
(DenseBrainVolume-class), 38
- DenseBrainVolume-class, 38
- dim,BrainData-method, 38
- dim,BrainSpace-method, 39
- dim,FileMetaInfo-method, 39
- dropDim, 40
- dropDim,AxisSet2D,missing-method  
(dropDim), 40
- dropDim,AxisSet2D,numeric-method  
(dropDim), 40
- dropDim,AxisSet3D,missing-method  
(dropDim), 40
- dropDim,AxisSet3D,numeric-method  
(dropDim), 40
- dropDim,BrainSpace,missing-method  
(dropDim), 40
- dropDim,BrainSpace,numeric-method  
(dropDim), 40
- eachSeries, 41
- eachSeries,BrainVector,function,missing-method  
(eachSeries), 41
- eachSeries,DenseBrainVector,function,missing-method  
(eachSeries), 41
- eachSeries,SparseBrainVector,function,logical-method  
(eachSeries), 41
- eachSlice, 42
- eachSlice,BrainVolume,function,logical-method  
(eachSlice), 42
- eachSlice,BrainVolume,function,missing-method  
(eachSlice), 42
- eachVolume, 42
- eachVolume,BrainBucket,function,logical,ANY-method  
(eachVolume), 42
- eachVolume,BrainBucket,function,missing,missing-method  
(eachVolume), 42
- eachVolume,BrainVolume,BrainVector,function,logical,ANY-method  
(eachVolume), 42
- eachVolume,BrainVolume,BrainVector,function,missing,BrainVolume-method  
(eachVolume), 42
- eachVolume,BrainVolume,BrainVector,function,missing,missing-method  
(eachVolume), 42
- eachVolume,SparseBrainVector,function,logical,missing-method  
(eachVolume), 42
- eachVolume,SparseBrainVector,function,missing,LogicalBrain  
(eachVolume), 42
- eachVolume,SparseBrainVector,function,missing,missing-method  
(eachVolume), 42
- fileMatches, 44
- fileMatches,BrainFileDescriptor,character-method  
(fileMatches), 44
- FileMetaInfo, 72
- FileMetaInfo-class, 44
- fill, 45
- fill,BrainVolume,list-method (fill), 45
- fill,BrainVolume,matrix-method (fill),  
45
- gridToCoord, 46
- gridToCoord,BrainSpace,matrix-method  
(gridToCoord), 46
- gridToCoord,BrainSpace,numeric-method  
(gridToCoord), 46
- gridToCoord,BrainVolume,matrix-method  
(gridToCoord), 46
- gridToGrid, 46
- gridToGrid,BrainSpace,matrix-method  
(gridToGrid), 46
- gridToGrid,matrix,matrix-method  
(gridToGrid), 46

- gridToIndex, 47
- gridToIndex,BrainSlice,matrix-method (gridToIndex), 47
- gridToIndex,BrainSpace,matrix-method (gridToIndex), 47
- gridToIndex,BrainSpace,numeric-method (gridToIndex), 47
- gridToIndex,BrainVolume,matrix-method (gridToIndex), 47
- gridToIndex,BrainVolume,numeric-method (gridToIndex), 47
  
- headerFile, 48
- headerFile,BrainFileDescriptor,character-method (headerFile), 48
- headerFileMatches, 48
- headerFileMatches,BrainFileDescriptor,character-method (headerFileMatches), 48
  
- image,BrainVolume-method, 49
- IndexLookupVolume, 16, 92
- IndexLookupVolume (IndexLookupVolume-class), 50
- IndexLookupVolume-class, 50
- indexToAxis (axisToIndex), 15
- indexToAxis,BrainSpace,numeric,numeric-method (axisToIndex), 15
- indexToCoord, 50
- indexToCoord,BrainSpace,numeric-method (indexToCoord), 50
- indexToCoord,BrainVolume,index-method (indexToCoord), 50
- indexToGrid, 51
- indexToGrid,BrainSlice,index-method (indexToGrid), 51
- indexToGrid,BrainSpace,numeric-method (indexToGrid), 51
- indexToGrid,BrainVector,index-method (indexToGrid), 51
- indexToGrid,BrainVector,integer-method (indexToGrid), 51
- indexToGrid,BrainVolume,index-method (indexToGrid), 51
- indexToGrid,BrainVolume,integer-method (indexToGrid), 51
- indices, 52
- indices,IndexLookupVolume-method (indices), 52
- indices,ROIVector-method (indices), 52
- indices,ROIVolume-method (indices), 52
- indices,SparseBrainVector-method (indices), 52
- inverseTrans, 52
- inverseTrans,BrainData-method (inverseTrans), 52
- inverseTrans,BrainSpace-method (inverseTrans), 52
  
- Kernel, 53
- Kernel-class, 54
  
- length,BrainVector-method (length,ROIVolume-method), 54
- length,ROIVolume-method, 54
- loadBucket, 55
- loadData, 55
- loadData,BrainBucketSource-method (loadData), 55
- loadData,BrainVectorSource-method (loadData), 55
- loadData,BrainVolumeSource-method (loadData), 55
- loadData,SparseBrainVectorSource-method (loadData), 55
- loadVector, 56
- loadVolume, 57
- loadVolumeList, 57
- LogicalBrainVolume, 26, 29, 58
- LogicalBrainVolume (LogicalBrainVolume-class), 58
- LogicalBrainVolume-class, 58
- lookup, 58
- lookup,IndexLookupVolume,numeric-method (lookup), 58
- lookup,SparseBrainVector,numeric-method (lookup), 58
  
- makeVector, 59
- makeVolume, 59
- map, 60
- map,BrainVolume,Kernel-method (map), 60
- mapToColors, 61
- matchAnatomy2D, 61
- matchAnatomy3D, 62
- matrixToVolumeList, 62
- MNI\_SPACE\_1MM, 63
  
- NamedAxis-class, 63

- names,BrainBucket-method  
(names,BrainBucketSource-method),  
63
- names,BrainBucketSource-method, 63
- ndim, 64
- ndim,AxisSet-method (ndim), 64
- ndim,BrainData-method (ndim), 64
- ndim,BrainSpace-method (ndim), 64
- neuroim, 64
- neuroim-package (neuroim), 64
- NiftiFileDescriptor, 65
- NiftiFileDescriptor-class, 65
- NiftiMetaInfo, 65, 65
- NiftiMetaInfo-class  
(FileMetaInfo-class), 44
- NullMetaInfo-class, 65
- numClusters, 66
- numClusters,ClusteredBrainVolume-method  
(numClusters), 66
  
- origin, 66
- origin,BrainData-method (origin), 66
- origin,BrainSpace-method (origin), 66
- overlay, 67
  
- patchSet, 67
- permMat, 68
- permMat,AxisSet2D-method (permMat), 68
- permMat,AxisSet3D-method (permMat), 68
- permMat,BrainSpace-method (permMat), 68
- print, 68
- print,AxisSet2D-method, 69
- print,AxisSet3D-method, 69
- print,NamedAxis-method, 70
  
- RandomSearchlight, 70
- readAFNIHeader, 71
- readColumns, 71
- readColumns,ColumnReader,numeric-method  
(readColumns), 71
- readElements, 72
- readElements,BinaryReader,numeric-method  
(readElements), 72
- readHeader, 72
- readMetaInfo, 73
- readMetaInfo,AFNIFileDescriptor-method  
(readMetaInfo), 73
- readMetaInfo,NiftiFileDescriptor-method  
(readMetaInfo), 73
  
- RegionCube, 73
- RegionSphere, 74
- RegionSquare, 75
- render, 75
- render,BrainSlice,numeric,numeric,character-method  
(render), 75
- renderSlice, 76
- reorient, 77
- reorient,BrainSpace,character-method  
(reorient), 77
- ROI-class, 77
- ROIVector, 77, 77
- ROIVector-class, 78
- ROIVolume, 78, 78, 88
- ROIVolume-class, 79
  
- scaleSeries, 79
- scaleSeries,BrainVector,logical,logical-method  
(scaleSeries), 79
- scaleSeries,BrainVector,logical,missing-method  
(scaleSeries), 79
- scaleSeries,BrainVector,missing,logical-method  
(scaleSeries), 79
- scaleSeries,BrainVector,missing,missing-method  
(scaleSeries), 79
- Searchlight, 80
- series (series\_roi), 81
- series,BasisBrainVector,matrix-method  
(series\_roi), 81
- series,BasisBrainVector,numeric-method  
(series\_roi), 81
- series,BrainVector,LogicalBrainVolume-method  
(series\_roi), 81
- series,BrainVector,matrix-method  
(series\_roi), 81
- series,BrainVector,numeric-method  
(series\_roi), 81
- series,BrainVector,ROIVolume-method  
(series\_roi), 81
- series,SparseBrainVector,matrix-method  
(series\_roi), 81
- series,SparseBrainVector,numeric-method  
(series\_roi), 81
- series\_roi, 81
- series\_roi,BrainVector,LogicalBrainVolume-method  
(series\_roi), 81
- series\_roi,BrainVector,matrix-method  
(series\_roi), 81

- series\_roi, BrainVector, numeric-method (series\_roi), 81
- series\_roi, BrainVector, ROIVolume-method (series\_roi), 81
- seriesIter, 80
- seriesIter, BrainVector-method (seriesIter), 80
- seriesIter, SparseBrainVector-method (seriesIter), 80
- show, AxisSet1D-method, 83
- show, AxisSet2D-method, 83
- show, AxisSet3D-method, 83
- show, AxisSet4D-method, 84
- show, BaseMetaInfo-method, 84
- show, BrainSpace-method, 85
- show, BrainVector-method, 85
- show, BrainVectorSource-method, 86
- show, BrainVolume-method, 86
- show, FileMetaInfo-method, 87
- show, NamedAxis-method, 87
- show, NullMetaInfo-method, 88
- show, ROIVolume-method, 88
- show, SparseBrainVector-method, 89
- slice, 89
- slice, BrainVolume, numeric, BrainSpace, AxisSet3D-method (slice), 89
- slice, BrainVolume, numeric, numeric, missing-method (slice), 89
- sliceData, 90
- space, 90
- space, BrainData-method (space), 90
- space, BrainSpace-method (space), 90
- space, IndexLookupVolume-method (space), 90
- spacing, 91
- spacing, BrainData-method (spacing), 91
- spacing, BrainSpace-method (spacing), 91
- SparseBrainVector, 26, 50, 81, 93
- SparseBrainVector (SparseBrainVector-class), 92
- SparseBrainVector-class, 92
- SparseBrainVectorSource, 26
- SparseBrainVectorSource (SparseBrainVectorSource-class), 93
- SparseBrainVectorSource-class, 93
- SparseBrainVolume, 93, 94
- SparseBrainVolume (SparseBrainVolume-class), 93
- SparseBrainVolume-class, 93
- splitFill, 94
- splitFill, BrainVolume, factor, function-method (splitFill), 94
- splitReduce, 95
- splitReduce, BrainVector, factor, function-method (splitReduce), 95
- splitReduce, BrainVector, factor, missing-method (splitReduce), 95
- splitReduce, matrix, factor, function-method (splitReduce), 95
- splitReduce, matrix, factor, missing-method (splitReduce), 95
- splitReduce, matrix, integer, function-method (splitReduce), 95
- splitReduce, matrix, integer, missing-method (splitReduce), 95
- splitScale, 96
- splitScale, DenseBrainVector, factor, logical, logical-method (splitScale), 96
- splitScale, DenseBrainVector, factor, logical, missing-method (splitScale), 96
- splitScale, DenseBrainVector, factor, missing, missing-method (splitScale), 96
- splitScale, matrix, factor, logical, logical-method (splitScale), 96
- splitScale, matrix, factor, missing, missing-method (splitScale), 96
- stripExtension, 98
- stripExtension, BrainFileDescriptor, character-method (stripExtension), 98
- subVector, 98
- subVector, DenseBrainVector, numeric-method (subVector), 98
- subVector, SparseBrainVector, numeric-method (subVector), 98
- takeVolume, 99
- takeVolume, BrainVector, numeric-method (takeVolume), 99
- takeVolume, SparseBrainVector, numeric-method (takeVolume), 99
- trans, 100
- trans, BrainData-method (trans), 100
- trans, BrainMetaInfo-method (trans), 100
- trans, BrainSpace-method (trans), 100
- trans, NIFTIMetaInfo-method (trans), 100

values, [101](#)  
values,ROIVector-method (values), [101](#)  
values,ROIVolume-method (values), [101](#)  
voxels, [101](#)  
voxels,Kernel-method (voxels), [101](#)

writeElements, [102](#)  
writeElements,BinaryWriter,numeric-method  
    (writeElements), [102](#)  
writeVector, [102](#)  
writeVector,BrainVector,character,character,missing-method  
    (writeVector), [102](#)  
writeVector,BrainVector,character,missing,character,ANY-method  
    (writeVector), [102](#)  
writeVector,BrainVector,character,missing,character-method  
    (writeVector), [102](#)  
writeVector,BrainVector,character,missing,missing-method  
    (writeVector), [102](#)  
writeVolume, [103](#)  
writeVolume,BrainVolume,character,character,missing-method  
    (writeVolume), [103](#)  
writeVolume,BrainVolume,character,missing,character-method  
    (writeVolume), [103](#)  
writeVolume,BrainVolume,character,missing,missing-method  
    (writeVolume), [103](#)  
writeVolume,ClusteredBrainVolume,character,missing,missing-method  
    (writeVolume), [103](#)