Package 'ANTsR'

April 30, 2015

Type Package

Title ANTs in R

Version 0.3

Author Brian B. Avants, Benjamin M. Kandel, Jeff T. Duda, Philip A. Cook, Nicholas J. Tustison, Shrinidhi KL

Maintainer Brian B. Avants <stnava@gmail.com>

Description ANTsR interfaces state of the art image processing with R statistical methods. The project grew out of the need, at University of Pennsylvania, to develop large-scale analytics pipelines that track provenance from scanner to scientific study. ANTsR wraps an ANTs and ITK C++ core via Rcpp to access these frameworks from within R and support reproducible analyses.

License GPL (>=2)

LazyLoad yes

Depends methods

Imports Rcpp, tools, magrittr

Suggests abind, BGLR, caret, cluster, d3Network, DMwR, e1071, extremevalues, fastICA, fpc, glasso, glmnet, grid, igraph, knitr, magic, MASS, mFilter, misc3d, moments, pixmap, png, psych, randomForest, rgl, robust, robustbase, signal, sna, testthat, visreg, wmtsa

LinkingTo Rcpp, ITKR

SystemRequirements cmake, git, clang (recommended)

NeedsCompilation yes

OS_type unix

Additional_repositories https://github.com/InsightSoftwareConsortium/ITK.git, https://github.com/stnava/ANTs.git

VignetteBuilder knitr

R topics documented:

aal *aal*

Description

A data frame label numbers and names for the neuroanatomical AAL labels.

Usage

```
data(aal)
```

Format

A data frame listing the following variables.

`label_num` Numerical label value.

`label_name` Shorthand anatomical value.

`cerebellum` binary value

`cortex` binary value

`deepgrey` binary value

`frontal` binary value

`isdmn` binary value

`issalience` binary value

`left` binary value

`limbic` binary value

`occipital` binary value

`parietal` binary value

`right` binary value

`temporal` binary value

`unknown` binary value

References

http://en.wikipedia.org/wiki/Automated_Anatomical_Labeling

Examples

```
data(aal)
```

abpBrainExtraction *An ants-based brain extraction script.*

Description

Brain extraction based on mapping a template image and its mask to the input image. Should be preceded by abpN4.

Usage

```
abpBrainExtraction(img = NA, tem = NA, temmask = NA, tdir = NA)
```

Arguments

img	image to which we map a brain mask
tem	Template image which has an associated label mask.
temmask	Template's antsImage brain mask.
tdir	temporary directory (optional)

Value

outputs a brain image and brain mask.

Author(s)

Tustison N, Avants BB

Examples

```
fn<-getANTsRData("r16")
img<-antsImageRead(fn)
img<-resampleImage(img,c(128,128),1,0)
tf<-getANTsRData("r27")
tem<-antsImageRead(tf)
tem<-resampleImage(tem,c(128,128),1,0)
temmask<-antsImageClone( tem )
temmask[ tem  > 20 ]<-1
temmask[ tem  <= 20 ]<-0
bm<-abpBrainExtraction(img=img,tem=tem,temmask=temmask)
```

abpN4 *MR image bias correction based on the N4 algorithm.*

Description

Truncate outlier intensities and bias correct with the N4 algorithm.

Usage

```
abpN4(img, intensityTruncation = c(0.025, 0.975, 256), mask = NA,
  usen3 = FALSE)
```

Arguments

img	image to be bias corrected
intensityTruncation	quantiles for image truncation.
mask	optional antsImage mask
usen3	Use N3 algorithm instead of N4

Value

outputs a bias corrected image. 1 indicates failure.

Author(s)

Tustison N, Avants BB

Examples

```
img <- antsImageRead(getANTsRData("r16"))
img2 <- abpN4( img )
```

---

affineInitializer *a multi-start optimizer for affine registration*

Description

Searches over the sphere to find a good initialization for further registration refinement, if needed. This is a arapper for the ANTs function antsAffineInitializer.

Usage

```
affineInitializer(fixedImage, movingImage, searchFactor = 20,
  radianFraction = 0.1, usePrincipalAxis = FALSE,
  localSearchIterations = 10, mask, txfn)
```

Arguments

fixedImage	the fixed reference image
movingImage	the moving image to be mapped to the fixed space
searchFactor	degree of increments on the sphere to search
radianFraction	between zero and one, defines the arc to search over
usePrincipalAxis	boolean to initialize by principal axis
localSearchIterations	gradient descent iterations
mask	optional mask to restrict registration
txfn	filename for the transformation

Value

transformationMatrix

Author(s)

Avants BB

Examples

```
fi <- antsImageRead(getANTsRData("r16"))
mi <- antsImageRead(getANTsRData("r27"))
tx <- affineInitializer( fi, mi )
```

---

antsApplyTransforms *Apply transforms to images.*

Description

Apply a transform list to map an image from one domain to another. In image registration, one computes mappings between (usually) pairs of images. These transforms are often a sequence of increasingly complex maps, e.g. from translation, to rigid, to affine to deformation. The list of such transforms is passed to this function to interpolate one image domain into the next image domain, as below. The order matters strongly and the user is advised to familiarize with the standards established in examples.

Usage

```
antsApplyTransforms(fixed, moving, transformlist = "",
  interpolator = "Linear", imagetype = 0, whichtoinvert = NA, ...)
```

Arguments

`fixed`	fixed image defining domain into which the moving image is transformed.
`moving`	moving image to be mapped to fixed space.
`transformlist`	list of transforms generated by antsRegistration where each transform is a filename.
`interpolator`	Choice of interpolator. One of: Linear, NearestNeighbor, MultiLabel, CosineWindowedSinc, WelchWindowedSinc, HammingWindowedSinc, LanczosWindowedSinc, Gaussian, or BSpline. other interpolator available in antsApplyTransforms.
`imagetype`	choose 0/1/2/3 mapping to scalar/vector/tensor/time-series .
`whichtoinvert`	list of booleans, same length as transforms
`...`	other parameters to pass to antsApplyTransforms.

Value

an antsImage is output. 1 – Failure

Author(s)

Shrinidhi KL, Avants BB

See Also

antsRegistration

Examples

```
# will give the full form of help
antsApplyTransforms("-h")
# see antsRegistration
# example 1 - simplified
fixed <- antsImageRead( getANTsRData("r16") ,2)
moving <- antsImageRead( getANTsRData("r64") ,2)
fixed <- resampleImage(fixed,c(64,64),1,0)
moving <- resampleImage(moving,c(64,64),1,0)
mytx <- antsRegistration(fixed=fixed , moving=moving ,
  typeofTransform = c("SyN") )
mywarpedimage <- antsApplyTransforms( fixed=fixed,moving=moving,
  transformlist=mytx$fwdtransforms )
mywarpedimage <- antsApplyTransforms( fixed=moving,moving=fixed,
  transformlist=mytx$invtransforms )
# full access via listing the inputs in standard ANTs format
```

`antsApplyTransformsToPoints` *Apply transforms to points.*

Description

Apply a transform list to map a pointset from one domain to another. In registration, one computes mappings between pairs of domains. These transforms are often a sequence of increasingly complex maps, e.g. from translation, to rigid, to affine to deformation. The list of such transforms is passed to this function to interpolate one image domain into the next image domain, as below. The order matters strongly and the user is advised to familiarize with the standards established in examples. Importantly, point mapping goes the opposite direction of image mapping, for both reasons of convention and engineering.

Usage

```
antsApplyTransformsToPoints(dim, points, transformlist = "",
  whichtoinvert = NA)
```

Arguments

`dim`	dimensionality of the transformation.
`points`	moving point set with n-points in rows of at least dim columns - we maintain extra information in additional columns. this may be either a dataframe or a 2D antsImage - the latter may be better for large pointsets.
`transformlist`	character vector of transforms generated by antsRegistration where each transform is a filename.
`whichtoinvert`	vector of booleans, same length as transforms

Value

same type as input. 1 – Failure

Author(s)

Avants BB

See Also

antsRegistration

Examples

```
fixed <- antsImageRead( getANTsRData("r16") ,2)
moving <- antsImageRead( getANTsRData("r64") ,2)
fixed <- resampleImage(fixed,c(64,64),1,0)
moving <- resampleImage(moving,c(64,64),1,0)
mytx <- antsRegistration(fixed=fixed , moving=moving ,
  typeofTransform = c("SyN") )
pts=data.frame(  x=c(110.5,120,130) , y=c(108.1,121.0,130),
  label=c(1,2,3) )
wpts <- antsApplyTransformsToPoints( dim=2, points=pts,
  transformlist=mytx$fwdtransforms )
wptsi <- antsApplyTransformsToPoints( dim=2, points=wpts,
  transformlist=mytx$invtransforms ) # close to pts

## Not run:
fixed <- antsImageRead( getANTsRData("r16") ,2)
moving <- antsImageRead( getANTsRData("r64") ,2)
fpts = kmeansSegmentation( fixed , 3 )$segmentation %>%
  thresholdImage(1,1) %>%
  labelClusters( 5 ) %>% getCentroids(5)
wpts <- antsApplyTransformsToPoints( dim=2, points=fpts,
 transformlist=mytx$fwdtransforms )
labimgf=fixed*0
labimgm=moving*0
for ( p in 1:nrow(wpts))
  {
  pt=as.numeric( wpts[p,1:2] )
  idx=round( antsTransformPhysicalPointToIndex(moving, pt ) )
  labimgm[ idx[1], idx[2] ]=p
  pt=as.numeric( fpts[p,1:2] )
  idx=round( antsTransformPhysicalPointToIndex(fixed, pt ) )
  labimgf[ idx[1], idx[2] ]=p
  }
plot(fixed,labimgf %>% iMath("GD",2) )
plot(moving,labimgm  %>% iMath("GD",2)  )

## End(Not run)
```

---

antsAverageImages *Computes average of image list*

Description

Calculate the mean of a list of antsImages

Usage

```
antsAverageImages(imageList, normalize = FALSE)
```

Arguments

imageList	list of antsImages
normalize	boolean determines if image is divided by mean before averaging

Author(s)

Avants BB, Pustina D

Examples

```
r16 <- antsImageRead(getANTsRData(r16))
r64 <- antsImageRead(getANTsRData(r64))
mylist <- list(r16, r64)
antsAverageImages(mylist)
```

---

antsBOLDNetworkAnalysis

a basic framework for network analysis that produces graph metrics

---

Description

An implementation of a network analysis framework for BOLD data. We expect that you mapped a label image (e.g. aal) to the 3D BOLD space. We build a network and graph metrics from this image and these labels based on the user-defined graph density level.

Usage

```
antsBOLDNetworkAnalysis(bold = NA, mask = NA, labels = NA, motion,
  gdens = 0.2, threshLo = 1, threshHi = 90, freqLo = 0.01,
  freqHi = 0.1, winsortrim = 0.02, throwaway)
```

Arguments

bold	input 4D image
mask	antsImage defines areas of interest
labels	antsImage defines regions of interest ie a parcellation
motion	motion parameters - if missing, will estimate from data
gdens	graph density applied to network covariance matrix
threshLo	lower threshold for the label image
threshHi	upper threshold for the label image
freqLo	lower frequency cutoff
freqHi	upper frequency cutoff
winsortrim	winsorize the bold signal by these values eg 0.02
throwaway	this number of initial bold volumes

Value

list of outputs

Author(s)

BB Avants

Examples

```
# none yet - this is not very well tested with recent ANTsR
## Not run:
myimg <- antsImageRead(getANTsRData( "ch2" ), 3)
mylab <- antsImageRead(getANTsRData( "ch2a" ), 3)
boldfn <- getANTsRData( "pcasl" )
bold <- antsImageRead( boldfn , 4 )
avgbold <- getAverageOfTimeSeries( bold )
breg <- antsRegistration( avgbold, myimg, typeofTransform = c("AffineFast") )
warpedParcellation <- antsApplyTransforms( avgbold, mylab,
    transformlist=breg$fwdtransforms, interpolator="NearestNeighbor" )
mask <- getMask( avgbold )
result <- antsBOLDNetworkAnalysis( bold=bold, mask=mask, warpedParcellation )

## End(Not run)
```

---

antsCopyImageInfo *Copy header info*

Description

Copy origin, direction, and spacing from one `antsImage` to another

Usage

```
antsCopyImageInfo(reference, target)
```

Arguments

`reference`	image object of S4 class `antsImage` to get values from.
`target`	image object of S4 class `antsImage` to copy values to.

Value

Target image with reference header information.

Examples

```
img <- makeImage(c(10,10),rnorm(100))
img2 <- makeImage(c(10,10), rnorm(100))
img2 <- antsCopyImageInfo(img, img2)
```

antsImage-class *An S4 class for an antsImage*

Description

C++ type used to represent an ITK image pointed to by 'pointer'. the actual image is of C++ type 'itk::image< pixeltype , dimension >::Pointer'

Usage

```
## S4 method for signature antsImage
show(object)

## S4 method for signature antsImage
initialize(.Object, pixeltype = "float",
  dimension = 3, components = 1)

## S4 method for signature antsImage
dim(x)

## S4 method for signature antsImage
min(x)

## S4 method for signature antsImage
max(x)

## S4 method for signature antsImage
var(x)

## S4 method for signature antsImage
sd(x)

## S4 method for signature antsImage
is.na(x)

## S4 method for signature antsImage
as.numeric(x, mask = logical(),
  region = new("antsRegion", index = integer(), size = integer()))

## S4 method for signature antsImage
as.matrix(x, mask = logical(),
  region = new("antsRegion", index = integer(), size = integer()))

## S4 method for signature antsImage
as.array(x, mask = logical(),
  region = new("antsRegion", index = integer(), size = integer()))

## S4 method for signature antsImage,ANY
e1 == e2

## S4 method for signature antsImage,ANY
```

```
e1 != e2

## S4 method for signature antsImage,ANY
e1 <= e2

## S4 method for signature antsImage,ANY
e1 >= e2

## S4 method for signature antsImage,ANY
e1 < e2

## S4 method for signature antsImage,ANY
e1 > e2
```

Arguments

object	input object to convert
.Object	input object to convert
pixeltype	string e.g. "float" "unsigned char" "int"
dimension	dimensionality of the image
components	number of components per pixel
x	input object to convert
mask	mask for the region
region	antsRegion for the image
e1	internal control for types
e2	internal control for types

Methods (by generic)

- show:
- initialize:
- dim:
- min:
- max:
- var:
- sd:
- is.na:
- as.numeric:
- as.matrix:
- as.array:
- ==:
- !=:
- <=:
- >=:
- <:
- >:

Slots

`pixeltype` usually float, can be other types unsigned char, int, double etc noting that short is not supported

`dimension` usually 2 or 3 but can be 4

`components` number of pixel components

`pointer` the memory location

`antsImageClone` *Clone an* `antsImage`.

Description

Clone an image object of S4 class `antsImage`. N.B.: You cannot use `a <- img` because the R assignment operator does not deal with the underlying C++ pointers.

Usage

```
antsImageClone(in_image, out_pixeltype = in_image@pixeltype)
```

Arguments

`in_image`	image object of S4 class `antsImage` to be cloned.
`out_pixeltype`	C++ datatype to be used to represent the pixels in the output image. Allowed values: `double`, `float`, `unsigned int`, `unsigned char`.

Value

object of class `antsImage`

Author(s)

Shrinidhi KL

Examples

```
img <- antsImageRead(getANTsRData("r16"), 2)
img2 <- antsImageClone(img)
img.int <- antsImageClone(img , "unsigned int")
```

antsImageGetSet *antsImageGetSet*

Description

Get and set methods for image header information

Usage

```
antsGetSpacing(x)

antsSetSpacing(x, spacing)

antsGetOrigin(x)

antsSetOrigin(x, origin)

antsGetDirection(x)

antsSetDirection(x, direction)
```

Arguments

x	antsImage to access, of dimensionality d.
spacing	numeric vector of length d.
origin	numeric vector of length d.
direction	matrix of size d * d.

Value

For get methods, vector of length d (origin, spacing) or matrix of size d * d (direction). For set methods, 0 to indicate success.

Examples

```
img <- makeImage(c(5,5), rnorm(25))
antsSetSpacing(img, c(2.0, 2.0))
antsSetOrigin(img, c(0.5, 0.5))
```

antsImageHeaderInfo *Read file info from image header*

Description

Read file info from image header

Usage

```
antsImageHeaderInfo(filename)
```

Arguments

filename	name of image file to scan for info

Value

outputs a list containing:

- pixelclass: Type of pixel (scalar, vector, etc).
- pixeltype: Type of pixel values (int, float, etc).
- nDimensions: Number of image dimensions.
- nComponents: Number of pixel dimensions.
- dimensions: Size of image dimensions.
- spacing: Pixel resolution.
- origin: Spatial origin of image
- pixelclass: Spatial directions of image axes.

Author(s)

Duda JT

Examples

```
antsImageHeaderInfo( getANTsRData("r16") )
```

---

antsImageMutualInformation

mutual information between two images

Description

compute mutual information between two images

Usage

```
antsImageMutualInformation(in_image1, in_image2)
```

Arguments

in_image1	antsImage
in_image2	antsImage

Value

mutual information value

Author(s)

Brian B. Avants

Examples

```
fi<-antsImageRead( getANTsRData("r16") ,2)
mi<-antsImageRead( getANTsRData("r64") ,2)
mival<-antsImageMutualInformation(fi,mi)
```

---

antsImagePhysicalSpaceConsistency *Check for physical space consistency*

Description

Check if two antsImage objects occupy the same physical space

Usage

```
antsImagePhysicalSpaceConsistency(img1, img2, coordinate.tolerance = 1e-06,
  direction.tolerance = 1e-06)
```

Arguments

img1	Image object of S4 class antsImage.
img2	Image object of S4 class antsImage.
coordinate.tolerance	floating point error tolerance in origin
direction.tolerance	floating point error tolerance in direction matrix

Value

Boolean indicating consistency of physical space

Examples

```
img1 <- makeImage(c(10,10), rnorm(100))
img2 <- makeImage(c(10,10), rnorm(100))
check <- antsImagePhysicalSpaceConsistency(img1, img2)
```

---

antsImageRead *Image Read*

Description

Read an image file into an S4 object of class 'antsImage'.

Usage

```
antsImageRead(filename, dimension = NULL, pixeltype = "float")
```

Arguments

filename	Name of the file to read the image from.
dimension	Number of dimensions of the image read. This need not be the same as the dimensions of the image in the file. Allowed values: 2, 3, 4. If not provided, the dimension is obtained from the image file
pixeltype	C++ datatype to be used to represent the pixels read. This datatype need not be the same as the datatype used in the file. Allowed values: 'double', 'float' , 'unsigned int' , 'unsigned char'.

Value

S4 object of Class 'antsImage' – Success
1 – Failure

Author(s)

Shrinidhi KL

See Also

antsImageWrite

Examples

```
fn <- getANTsRData( "r16" )
fi <- antsImageRead( fn )
img <- antsImageRead( fn , dimension = 2  )
img <- antsImageRead( fn , dimension = 2 , double )
```

---

antsImageWrite *Image Write*

Description

Write an image object of S4 class antsImage to a file.

Usage

```
antsImageWrite(image, filename)
```

Arguments

image	Image object of S4 class antsImage to be written.
filename	Name of the file to write the image to.

Value

0 – Success
1 – Failure

Author(s)

Shrinidhi KL

See Also

antsImageRead

Examples

```
fn <- getANTsRData( "r16" )
fi <- antsImageRead( fn )
antsImageWrite( fi , tempfile( fileext = ".nii.gz" ) )
antsImageWrite( fi , tempfile( fileext = ".mha" ) )
antsImageWrite( fi , tempfile( fileext = ".nrrd" ) )
antsImageWrite( antsImageClone( fi, "unsigned int" ) ,
  tempfile( fileext = ".jpg" )  )
antsImageWrite( antsImageClone( fi, "float" ) ,
  tempfile( fileext = ".tif" )  )
antsImageWrite( fi, tempfile( fileext = ".mrc" )  )
antsImageWrite( fi, tempfile( fileext = ".hd5" )  )
```

---

antsMatrix-class *An S4 class to hold an antsMatrix imported from ITK types C++ type used to represent an element of the matrix pointer to the actual image C++ type 'itk::image< pixeltype , dimension >::Pointer'*

Description

An S4 class to hold an antsMatrix imported from ITK types C++ type used to represent an element of the matrix pointer to the actual image C++ type 'itk::image< pixeltype , dimension >::Pointer'

Usage

```
## S4 method for signature antsMatrix
initialize(.Object, elementtype)

## S4 method for signature antsMatrix
as.data.frame(x)

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

Arguments

.Object	input object to convert
elementtype	string e.g. "float"
x	input object to convert

Methods (by generic)

- initialize:
- as.data.frame:
- as.list:

Slots

`elementtype`

`pointer` the memory location

`antsMotionCalculation` *Correct 4D time-series data for motion.*

Description

Correct 4D time-series data for motion.

Usage

```
antsMotionCalculation(img, mask=NA, fixed=NA, moreaccurate=1, framewise=1)
```

Arguments

`img`	antsImage, usually 4D.
`mask`	mask for image (3D). If not provided, estimated from data.
`fixed`	Fixed image to register all timepoints to. If not provided, mean image is used.
`moreaccurate`	Level of accuracy desired for motion correction. Higher is more accurate.
`framewise`	Calculate framewise displacement?

Value

List containing:

- moco_img Motion corrected time-series image.
- moco_params Data frame of translation parameters.
- moco_avg_img Average motion-corrected image.
- moco_mask Mask used to calculate framewise displacement.
- tsDisplacement Time-series displacement image.
- dvars DVARS, derivative of frame-wise intensity changes.

Author(s)

Benjamin M. Kandel

Examples

```
set.seed(120)
simimg<-makeImage(rep(5,4), rnorm(5^4))
# for real data, use simimg <- antsImageRead(getANTsRData(pcasl), 4)
antsMotionCalculation(simimg,moreaccurate=0)
```

antsMotionCorr *Motion Correction*

Description

This program is a user-level registration application meant to utilize ITKv4-only classes. The user can specify any number of *stages* where a *stage* consists of – a transform, an image metric, number of iterations, shrink factors, and smoothing sigmas for each level. Specialized for 4D time series data: fixed image is 3D, moving image should be the 4D time series. Fixed image is a reference space or time slice.

Usage

```
antsMotionCorr(...)
```

Arguments

...	antsMotionCorr parameters, as in ANTs. this is a direct call to the low level c++ and as such has non-standard parameters.

Value

0 – Success
1 – Failure

Author(s)

Shrinidhi KL

Examples

```
# boldfn <- getANTsRData( "pcasl" )
# bold <- antsImageRead( boldfn , 4 )
set.seed( 123 )
bold <- makeImage( c(10,10,10,20) , rnorm( 10*10*10*20)+1 )
bold <- iMath( bold, "PadImage", 5 )
aimg <- new("antsImage", "float", 3)
aimg <- new("antsImage", "float", 3)
mocoImg <- new("antsImage", "float", 4)
mocoParams <- new("antsMatrix", "double")
antsMotionCorr( list( d = 3 , a = bold , o = aimg ) )
antsMotionCorr( list( d = 3 ,
  o = list( mocoParams , mocoImg , aimg ) ,
  m = list( name = "MI" , aimg , bold , 1 , 32 , "Regular", 0.1 ) ,
  t = "Rigid[0.01]" , i = 25 ,
  u = 1 , e = 1 , s = 0 , f = 1 , n = 25 ) )
motiondf <- as.data.frame( mocoParams )
```

antsRegion-class *An S4 class to hold a region of an antsImage*

Description

An S4 class to hold a region of an antsImage

Slots

index

size

antsRegistration *Perform registration between two images.*

Description

Register a pair of images either through the full or simplified interface to the ANTs registration method.

Usage

```
antsRegistration(fixed = NA, moving = NA, typeofTransform = "SyN",
  initialTransform = NA, outprefix = "", mask = NA, gradStep = NA, ...)
```

Arguments

fixed	fixed image to which we register the moving image.
moving	moving image to be mapped to fixed space.
typeofTransform	Either a one stage rigid/affine mapping or a 2-stage affine+syn mapping. Mutual information metric by default. See Details. One of Rigid, Affine, AffineFast, SyN, SyNCC,
initialTransform	transforms to prepend
outprefix	output will be named with this prefix.
mask	mask the registration.
gradStep	gradient step size (not for all tx)
...	additional options see antsRegistration in ANTs

Details

typeofTransform can be one of:

- "Rigid": Rigid transformation: Only rotation and translation.
- "Affine": Affine transformation: Rigid + scaling.
- "AffineFast": Fast version of `Affine`.
- "SyN": Symmetric normalization: Affine + deformable transformation, with mutual information as optimization metric.
- "SyNCC": SyN, but with cross-correlation as the metric.
- "SynBold": SyN, but optimized for registrations between BOLD and T1 images.
- "SyNAggro": SyN, but with more aggressive registration (fine-scale matching and more deformation). Takes more time than `SyN`.
- "TVMSQ": time-varying diffeomorphism with mean square metric

Value

outputs a list containing:

- warpedmovout: Moving image warped to space of fixed image.
- warpedfixout: Fixed image warped to space of moving image.
- fwdtransforms: Transforms to move from moving to fixed image.
- invtransforms: Transforms to move from fixed to moving image.

Ouptut of 1 indicates failure

Author(s)

Shrinidhi KL, Tustison NJ, Avants BB

Examples

```
fi <- antsImageRead(getANTsRData("r16") ,2)
mi <- antsImageRead(getANTsRData("r64") ,2)
fi<-resampleImage(fi,c(60,60),1,0)
mi<-resampleImage(mi,c(60,60),1,0) # speed up
mytx <- antsRegistration(fixed=fi, moving=mi, typeofTransform = c(SyN) )
mywarpedimage <- antsApplyTransforms( fixed=fi, moving=mi,
  transformlist=mytx$fwdtransforms )
```

`antsrimpute` *antsrimpute*

Description

Impute NA's on data frame.

Usage

```
antsrimpute(mydat, FUN = mean, ...)
```

Arguments

mydat	Data frame to be imputed.
FUN	Method to be used for imputation (defaults to mean).
...	Additional parameters to pass to FUN.

Details

Imputes NA's on data frame (column-wise), using a user-specified function (mean is default). also works on a vector.

Value

Returns mydat with NA's replaced by imputed numbers.

Author(s)

Kandel BM, Avants BB

Examples

```
mydat <- data.frame(A=c(1,2,4,5), B=c(1,NA,4,5))
mean.impute <- antsrimpute(mydat)
median.impute <- antsrimpute(mydat, median)
```

---

antsSetPixels *Set a pixel value at an index*

Description

Set a pixel value at an index in an 'antsImage'.

Usage

```
antsSetPixels(x, i = NA, j = NA, k = NA, l = NA, value)
```

Arguments

x	Image object of S4 class 'antsImage'.
i	the slowest moving index to the image
j	the next slowest moving index to the image, similar for k (2d)
k	the next slowest moving index to the image (3d)
l	the next slowest moving index to the image (4d)
value	the value to place at this location

Value

array of pixel values

Examples

```
img<-makeImage(c(10,10),rnorm(100))
antsSetPixels(img,2,3,value=Inf)
```

antsSpatialICAfMRI *Perform spatial ICA on fMRI bold data.*

Description

Perform spatial ICA on group or individual fMRI data. Preprocessing should be performed prior to calling this function (cf preprocessfMRI.R).

Usage

```
antsSpatialICAfMRI(boldImages, maskImage = NA, numberOfICAComponents = 20,
  normalizeComponentImages = TRUE)
```

Arguments

boldImages	a list of 4-D ANTs image fMRI data.
maskImage	A 3-D ANTs image defining the region of interest. This must be specified.
numberOfICAComponents	Number of estimated observers (components).
normalizeComponentImages	Boolean to specify whether each component vector element is normalized to its z-score.

Value

Output list includes standard ICA matrices from the fastICA algorithm:

X = pre-processed data matrix

K = pre-whitening matrix that projects data onto the first n.comp principal components

W = estimated un-mixing matrix (see definition in details)

A = estimated mixing matrix

S = estimated source matrix

and the component images.

Author(s)

Tustison NJ, Avants BB

Examples

```
set.seed( 123 )
boldImages <- list()
n=16
nvox <- n*n*n*12
dims <- c(n,n,n,12)
boldImages[[1]] <- makeImage( dims , rnorm( nvox )+500 )
boldImages[[2]] <- makeImage( dims , rnorm( nvox )+500 )
boldImages[[3]] <- makeImage( dims , rnorm( nvox )+500 )

cleanBoldImages <- list()
for( i in 1:length( boldImages ) )
```

```
  {
  fmri <- preprocessfMRI( boldImages[[i]], residualizeMatrix=FALSE )
  if( i == 1 ) maskImage <- fmri$maskImage
  cleanBoldImages[[i]] <- fmri$cleanBoldImage
  }

icaResults <- antsSpatialICAfMRI( cleanBoldImages, maskImage,
  numberOfICAComponents = 2 )
```

`antsTransformIndexToPhysicalPoint`
Get Spatial Point from Index

Description

Get spatial point from index of an `antsImage`.

Usage

```
antsTransformIndexToPhysicalPoint(x, index)
```

Arguments

`x`	image object of S4 class `antsImage` to get values from.
`index`	image index

Value

array of pixel values

Examples

```
img <- makeImage(c(10,10),rnorm(100))
pt <- antsTransformIndexToPhysicalPoint(img, c(2,2))
```

`antsTransformPhysicalPointToIndex`
Get Index from Spatial Point

Description

Get index from spatial point of an 'antsImage'.

Usage

```
antsTransformPhysicalPointToIndex(x, point)
```

Arguments

`x`	Image object of S4 class 'antsImage' to get values from.
`point`	image physical point

Value

array of pixel values

Examples

```
img<-makeImage(c(10,10),rnorm(100))
pt<-antsTransformPhysicalPointToIndex(img,c(2,2))
```

`as.antsMatrix` *as.antsMatrix*

Description

convert types to an antsMatrix

Usage

```
as.antsMatrix(object, ...)

## S4 method for signature list
as.antsMatrix(object, elementtype = "float")

## S4 method for signature data.frame
as.antsMatrix(object, elementtype = "float")

## S4 method for signature matrix
as.antsMatrix(object, elementtype = "float")
```

Arguments

`object`	An object
`...`	other parameters
`elementtype`	e.g. "float" or "double"

Methods (by class)

- `list`:
- `data.frame`:
- `matrix`:

Examples

```
as.antsMatrix(matrix(rnorm(10), nrow=2))
```

aslAveraging *Average ASL tag-control pairs to estimate perfusion*

Description

This function averages arterial spin labeling (ASL) functional MRI tag-control image pairs to estimate perfusion.

Usage

```
aslAveraging(asl, mask = NA, nuisance = NA, method = "regression")
```

Arguments

asl	input asl image
mask	in which to calculate perfusion
nuisance	nuisance covariates to include in regression
method	method to use for computing average. One of sincSubtract, simpleSubtract, regression, or bayesian. See Details.

Author(s)

Kandel BM, Avants BB

Examples

```
nvox <- 5 * 5 * 5 * 10
dims <- c(5, 5, 5, 10)
voxvals <- array(rnorm(nvox) + 500, dim=dims)
voxvals[, , , 5] <- voxvals[, , , 5] + 600
asl <- makeImage(dims, voxvals) %>% iMath("PadImage", 2)
censored <- aslCensoring(asl)
avg <- aslAveraging(censored$asl.inlier)
```

aslCensoring *Censor bad volumes from ASL data.*

Description

Censor bad volumes from ASL data.

Usage

```
aslCensoring(asl, mask = NA, nuis = NA, method = "outlier", ...)
```

Arguments

asl	input asl image
mask	mask for calculating perfusion
nuis	fixed nuisance parameters
method	one of 'outlier', 'robust', or 'scor'. See Details.
...	Additional arguments to pass to censoring method. See Details.

Details

aslCensoring is an interface to ASL timepoint censoring algorithms. Three options are currently provided, with different additional arguments:

1. outlier Outlier rejection from Tan et al. This method rejects volumes that are either far from the mean of the time-series or whose standard deviation is far from the standard deviations of the individual volumes. Accepts two additional arguments:
   - sigma.mean: how many standard deviations the mean of the volume can be from the mean of all the volumes before being thrown out.
   - sigma.sd: how many standard deviations from the mean of standard deviations can the standard deviation of the volume be before being thrown out.
2. robust Uses a robust regression approach to estimate volumes with high leverage. Accepts three arguments:
   - nuis: Nuisance regressors to use as covariates.
   - robthresh: Threshold for weights on leverage estimates. Points with weights under this value will be thrown out; defaults to 0.95.
   - skip: Proportion of points to skip when estimating leverage. Defaults to 20 (1/20 of the image is used).
3. scor SCOR method of Dolui et al. No parameters.

Value

vector of the same length as number of timepoints in asl, with 1 indicating the corresponding timepoint is included and 0 indicating exclusion.

Author(s)

Kandel BM

References

Tan H. et al., "A Fast, Effective Filtering Method for Improving Clinical Pulsed Arterial Spin Labeling MRI," JMRI 2009.

Examples

```
nvox <- 5 * 5 * 5 * 10
dims <- c(5, 5, 5, 10)
voxvals <- array(rnorm(nvox) + 500, dim=dims)
voxvals[, , , 5] <- voxvals[, , , 5] + 600
asl <- makeImage(dims, voxvals) %>% iMath("PadImage", 2)
censored <- aslCensoring(asl)
```

aslDenoiseR *WIP: data-driven denoising for ASL MRI*

Description

Denoises regression based reconstruction of CBF from arterial spin labeling

Usage

```
aslDenoiseR(boldmatrix, targety, motionparams = NA, selectionthresh = 0.1,
  maxnoisepreds = 1:12, debug = FALSE, polydegree = 4,
  crossvalidationgroups = 4, scalemat = F, noisepoolfun = max,
  usecompcor = F)
```

Arguments

boldmatrix	input bold matrix
targety	target to predict
motionparams	motion parameters / nuisance variables
selectionthresh	e.g. 0.1 take 10 percent worst variables for noise estimation
maxnoisepreds	integer search range e.g 1:10
debug	boolean
polydegree	eg 4 for polynomial nuisance variables
crossvalidationgroups	prior defined or integer valued
scalemat	boolean
noisepoolfun	function to help select noise pool e.g. max
usecompcor	boolean

Value

matrix is output

Author(s)

Avants BB

Examples

```
# asl<-antsImageRead( getANTsRData("pcasl") )
set.seed(1)
nvox <- 10*10*10*20
dims <- c(10,10,10,20)
asl <- makeImage( dims , rnorm( nvox )+500 )
aslmean <- getAverageOfTimeSeries( asl )
aslmask <- getMask( aslmean  )
aslmat<-timeseries2matrix( asl, aslmask )
for ( i in 1:10 ) aslmat[,i*2]<-aslmat[,i*2]*2
```

```
asl<-matrix2timeseries( asl, aslmask, aslmat )
tc<-as.factor(rep(c("C","T"),nrow(aslmat)/2))
dv<-computeDVARS(aslmat)
dnz<-aslDenoiseR( aslmat, tc, motionparams=dv, selectionthresh=0.1,
  maxnoisepreds=c(1:2), debug=TRUE, polydegree=2, crossvalidationgroups=2 )
## Not run:
# a classic regression approach to estimating perfusion
# not recommended, but shows the basic idea.
# see ?quantifyCBF for a better approach
perfmodel<-lm( aslmat ~ tc + dnz$noiseu  )
perfimg<-antsImageClone(aslmask)
perfimg[ aslmask == 1 ]<-bigLMStats( perfmodel )$beta[1,]
m0<-getAverageOfTimeSeries(asl)
ctl<-c(1:(nrow(aslmat)/2))*2
m0[ aslmask==1 ]<-colMeans(aslmat[ctl,])
pcasl.parameters<-list( sequence="pcasl", m0=m0 )
cbf <- quantifyCBF( perfimg, aslmask, pcasl.parameters )

## End(Not run)
```

`aslPerfusion` *ASL-based Perfusion from PASL, CASL or pCASL.*

Description

This function will estimate perfusion from an ASL time series using a (robust) regression approach. It will do motion correction, compcorr, factor out nuisance variables and use regression to estimate the perfusion itself. It will mask the image too based on a simple procedure (should fix this in the future by allowing the user to optionally pass a mask in from the outside). WARNING: the function will estimate the m0 image from the mean of the control tags assuming that the data is acquired T C T C as is most of JJ's data. Quantitative CBF can be obtained by mutiplying the output of this function by a scalar.

Usage

```
aslPerfusion(asl, maskThresh = 0.75, moreaccurate = 1, dorobust = 0.92,
  m0 = NA, skip = 20, mask = NA, checkmeansignal = 100,
  moco_results = NULL, regweights = NULL, useDenoiser = NA,
  useBayesian = 0, verbose = FALSE, ncompcor = 0, N3 = FALSE)
```

Arguments

`asl`	input asl image
`maskThresh`	for estimating a brain mask
`moreaccurate`	zero, one or two with the last being the most accurate
`dorobust`	robustness parameter, lower value keeps more data
`m0`	known M0 if any
`skip`	stride to speed up robust regression weight estimates
`mask`	known brain mask
`checkmeansignal`	throw out volumes with mean lower than this thresh

moco_results	passes prior motion results so moco does not get repeated
regweights	known temporal weights on regression solution, if any
useDenoiser	use the aslDenoiser if this value is a range gt than zero
useBayesian	strength of bayesian prior
verbose	bool
ncompcor	number of compcor parameters
N3	bool correct target image before motion corr

Value

output a list of relevant objects

Author(s)

Avants BB

Examples

```
# image available at http://files.figshare.com/1701182/PEDS012_20131101.zip
# fn<-PEDS012_20131101_pcasl_1.nii.gz
# asl<-antsImageRead(fn,4)
set.seed(1)
nvox <- 5*5*5*10
dims <- c(5,5,5,10)
asl <- makeImage( dims , rnorm( nvox )+500 ) %>% iMath("PadImage" , 2 )
aslmean <- getAverageOfTimeSeries( asl )
aslmask <- getMask( aslmean , 0.001 , Inf )
aslmat<-timeseries2matrix( asl, aslmask )
for ( i in 1:10 ) aslmat[,i*2]<-aslmat[,i*2]*2
asl<-matrix2timeseries( asl, aslmask, aslmat )
pcasl.processing <- aslPerfusion( asl, moreaccurate=1, dorobust=0 )
pcasl.processing <- aslPerfusion( asl, moreaccurate=1, ncompcor=2 )
# allow some rejection
pcasl.processing <- aslPerfusion( asl, moreaccurate=1, dorobust=0.925 )
```

---

atropos *FMM Segmentation*

Description

A finite mixture modeling (FMM) segmentation approach with possibilities for specifying prior constraints. These prior constraints include the specification of a prior label image, prior probability images (one for each class), and/or an MRF prior to enforce spatial smoothing of the labels. Similar algorithms include FAST and SPM. atropos can also perform multivariate segmentation if you pass a list of images in: e.g. `a=c(img1,img2)`.

Usage

```
atropos(a, x, i = "KMeans[3]", m = "[0.2,1x1]", c = "[5,0]",
  priorweight = 0.25, ...)
```

Arguments

a	One or more scalar images to segment. If priors are not used, the intensities of the first image are used to order the classes in the segmentation output, from lowest to highest intensity. Otherwise the order of the classes is dictated by the order of the prior images.
x	mask image.
i	initialization usually KMeans[N] for N classes or a list of N prior probability images. See Atropos in ANTs for full set of options.
m	mrf parameters as a string, usually "[smoothingFactor,radius]" where smoothingFactor determines the amount of smoothing and radius determines the MRF neighborhood, as an ANTs style neighborhood vector eg "1x1x1" for a 3D image. The radius must match the dimensionality of the image, eg 1x1 for 2D and The default in ANTs is smoothingFactor=0.3 and radius=1. See Atropos for more options.
c	convergence parameters, "[numberOfIterations,convergenceThreshold]". A threshold of 0 runs the full numberOfIterations, otherwise Atropos tests convergence by comparing the mean maximum posterior probability over the whole region of interest defined by the mask x.
priorweight	usually 0 (priors used for initialization only), 0.25 or 0.5.
...	more parameters, see Atropos help in ANTs

Value

0 – Success
1 – Failure

Author(s)

Shrinidhi KL, B Avants

Examples

```
img <- antsImageRead( getANTsRData("r16") , 2 )
img <- resampleImage( img, c(64,64), 1, 0 )
mask <- getMask(img)
segs1 <- atropos( d = 2, a = img, m = [0.2,1x1],
   c = [2,0],  i = kmeans[3], x = mask )

# Use probabilities from k-means seg as priors

segs2 <- atropos( d = 2, a = img, m = [0.2,1x1],
   c = [2,0],  i = segs1$probabilityimages, x = mask )

feats <- list(img, iMath(img,"Laplacian"), iMath(img,"Grad") )
segs3 <- atropos( d = 2, a = feats, m = [0.2,1x1],
   c = [2,0],  i = segs1$probabilityimages, x = mask )
```

basicInPaint *Inpaints missing imaging data from boundary data*

Description

Smooths data along the boundary into the missing region.

Usage

```
basicInPaint(img, paintMask, speedimage = NA, its = 0, gparam = 0.05)
```

Arguments

img	antsImage to be approximated / painted
paintMask	painting mask with values 1 or values 1 and 2 - if there is a 2 then it will learn from label 1 to paint label 2. should cover the brain.
speedimage	- larger means faster/better
its	- iterations of graddescent
gparam	- graddescent param e.g. 0.05

Value

inpainted image

Author(s)

Brian B. Avants

Examples

```
set.seed(123)
fi<-abs(replicate(100, rnorm(100)))
fi[1:10,]<-fi[,1:10]<-fi[91:100,]<-fi[,91:100]<-0
mask<-fi
mask[ mask > 0 ]<-1
mask2<-mask
mask2[11:20,11:20]<-2
mask<-as.antsImage( mask2  )
fi<-as.antsImage( fi )
fi<-smoothImage( fi, 3 )
painted<-basicInPaint( fi, mask )
## Not run:
# lmask<-antsImageRead( "brainmask.nii.gz", 2 )
# limg<-antsImageRead( "r16slice_lesion.nii.gz", 2 )
# mm<-basicInPaint(limg,lmask)
# plot(mm)
# mm2<-basicInPaint(limg,lmask,its=10,gparam=0.05)
# plot(mm2)

## End(Not run)
```

bayesianCBF *Uses probabilistic segmentation to constrain pcasl-based cbf computation.*

Description

Employs a robust regression approach to learn the relationship between a sample image and a list of images that are mapped to the same space as the sample image.

Usage

```
bayesianCBF(pcasl, segmentation, tissuelist, myPriorStrength = 30,
  useDataDrivenMask = 3, denoisingComponents = 1:8,
  robustnessvalue = 0.95, localweights = F, priorBetas = NA)
```

Arguments

pcasl	img antsImage for cbf
segmentation	image, should cover the brain.
tissuelist	a list containing antsImages eg list(prob1,...,probN)
myPriorStrength	- e.g 30
useDataDrivenMask	- morphology parameters e.g. 3
denoisingComponents	- data-driven denoising parameters
robustnessvalue	- value (e.g. 0.95) that throws away time points
localweights	Use estimate of voxel-wise reliability to inform prior weight?
priorBetas	prior betas for each tissue and predictor

Value

estimated cbf image

Author(s)

Brian Beaumont Avants and Benjamin M. Kandel

Examples

```
## Not run:
set.seed(123)
# see fMRIANTs github repository for data and I/O suggestions

## End(Not run)
```

`bayesianlm` *Simple bayesian regression function.*

Description

Take a prior mean and precision matrix for the regression solution and uses them to solve for the regression parameters. The Bayesian model, here, is on the multivariate distribution of the parameters.

Usage

```
bayesianlm(X, y, priorMean, priorPrecision, priorIntercept = 0, regweights,
  includeIntercept = F)
```

Arguments

`X`	data matrix
`y`	outcome
`priorMean`	expected parameters
`priorPrecision`	inverse covariance matrix of the parameters -
`priorIntercept`	inverse covariance matrix of the parameters -
`regweights`	weights on each y, a vector as in lm
`includeIntercept`	include the intercept in the model

Value

bayesian regression solution is output

Author(s)

Avants BB

Examples

```
# make some simple data
  set.seed(1)
  n<-20
  rawvars<-sample(1:n)
  nois<-rnorm(n)
  # for some reason we dont know age is correlated w/noise
  age<-as.numeric(rawvars)+(abs(nois))*sign(nois)
  wt<-( sqrt(rawvars) + rnorm(n) )
  mdl<-lm( wt ~ age + nois )
  summary(mdl)
  X<-model.matrix( mdl )
  priorcoeffs<-coefficients(mdl)
  covMat<-diag(length(priorcoeffs))+0.1
  # make some new data
  rawvars2<-sample(1:n)
  nois2<-rnorm(n)
```

```
  # now age is correlated doubly w/noise
  age2<-as.numeric(rawvars2)+(abs(nois2))*sign(nois2)*2.0
  wt2<-( sqrt(rawvars2) + rnorm(n) )
  mdl2<-lm( wt2 ~ age2 + nois2 )
  X2<-model.matrix( mdl2 )
  precisionMat<-solve( covMat )
  precisionMat[2,2]<-precisionMat[2,2]*1.e3 # heavy prior on age
  precisionMat[3,3]<-precisionMat[3,3]*1.e2 # some prior on noise
  bmdl<-bayesianlm( X2, wt2, priorMean=priorcoeffs, precisionMat )
  bmdlNoPrior<-bayesianlm( X2, wt2 )
  print(priorcoeffs)
  print(bmdl$beta)
  print(bmdlNoPrior$beta)
  ## Not run:
  fn<-PEDS012_20131101_pcasl_1.nii.gz
  fn<-getANTsRData("pcasl")
# image available at http://files.figshare.com/1701182/PEDS012_20131101.zip
  asl<-antsImageRead(fn,4)
  tr<-antsGetSpacing(asl)[4]
  aslmean<-getAverageOfTimeSeries( asl )
  aslmask<-getMask(aslmean,lowThresh=mean(aslmean),cleanup=TRUE)
  aslmat<-timeseries2matrix(asl,aslmask)
  tc<-as.factor(rep(c(C,T),nrow(aslmat)/2))
  dv<-computeDVARS(aslmat)
  # do some comparison with a single y, no priors
  y<-rowMeans(aslmat)
  perfmodel<-lm( y ~ tc + dv ) # standard model
  tlm<-bigLMStats( perfmodel )
  X<-model.matrix( perfmodel )
  blm<-bayesianlm(  X, y )
  print( tlm$beta.p )
  print( blm$beta.p )
  # do some bayesian learning based on the data
  perfmodel<-lm( aslmat ~ tc + dv ) # standard model
  X<-model.matrix( perfmodel )
  perfmodel<-lm( aslmat ~ tc + dv )
  bayesianperfusionloc<-rep(0,ncol(aslmat))
  smoothcoeffmat<-perfmodel$coefficients
  nmatimgs<-list()
  for ( i in 1:nrow(smoothcoeffmat) )
    {
    temp<-antsImageClone( aslmask )
    temp[ aslmask == 1 ] <- smoothcoeffmat[i,]
#   temp<-iMath(temp,PeronaMalik,150,10)
    temp<-smoothImage(temp,1.5)
    nmatimgs[[i]]<-getNeighborhoodInMask(temp,aslmask,
      rep(2,3), boundary.condition = mean)
    smoothcoeffmat[i,]<-temp[ aslmask==1 ]
    }
  prior  <- rowMeans( smoothcoeffmat  )
  invcov <- solve( cov( t( smoothcoeffmat ) ) )
  blm2<-bayesianlm(  X, y, prior, invcov*1.e4 )
  print( blm2$beta.p )
  for ( i in 1:ncol(aslmat) )
    {
    parammat<-nmatimgs[[1]][,i]
    for ( k in 2:length(nmatimgs))
```

```
    parammat<-cbind( parammat, nmatimgs[[k]][,i] )
  locinvcov<-solve( cov( parammat ) )
  localprior<-(smoothcoeffmat[,i])
  blm<-bayesianlm(  X, aslmat[,i], localprior, locinvcov*1.e4 )
  bayesianperfusionloc[i]<-blm$beta[1]
  }
perfimg<-antsImageClone(aslmask)
basicperf<-bigLMStats( perfmodel )$beta[1,]
perfimg[ aslmask == 1 ]<-basicperf
antsImageWrite(perfimg,perf.nii.gz)
perfimg[ aslmask == 1 ]<-bayesianperfusionloc
antsImageWrite(perfimg,perf_bayes.nii.gz)
print( cor.test(basicperf, perfimg[ aslmask == 1 ] ) )

## End(Not run)
```

`bigLMStats` *Efficiently compute basic statistical inference from regressions with multiple outcomes*

Description

This function simplifies calculating p-values from linear models in which there are many outcome variables, such as in voxel-wise regressions. To perform such an analysis in R, you can concatenate the outcome variables column-wise into an n by p matrix `y`, where there are n subjects and p outcomes (see `Examples`). Calling `lm(y~x)` calculates the coefficients, but statistical inference is not provided. This function provides basic statistical inference efficiently.

Usage

```
bigLMStats(mylm, lambda = 0, includeIntercept = F)
```

Arguments

`mylm`	Object of class `lm`.
`lambda`	Value of ridge penalty for inverting ill-conditioned matrices.
`includeIntercept`	Whether or not to include p-values for intercept term in result.

Value

A list containing objects:

`fstat`	F-statistic of whole model (one value per outcome).
`pval.model`	p-value of model (one value per outcome).
`beta`	Values of coefficients (one value per predictor per outcome).
`beta.std`	Standard error of coefficients.
`beta.t`	T-statistic of coefficients.
`beta.pval`	p-value of coefficients.

Author(s)

Kandel BM.

Examples

```
nsub <- 100
set.seed(1500)
x <- 1:nsub
y <- matrix(c(x + rnorm(nsub), sin(x)), nrow=nsub)
x <- cbind(x, x^2)
y1 <- y[, 1]
y2 <- y[, 2]
lm1 <- lm(y1~x)
lm2 <- lm(y2~x)
mylm <- lm(y ~ x)

myest <- bigLMStats(mylm)
print(paste("R beta estimates for first outcome is", summary(lm1)$coefficients[-1,1],
            "and for second outcome is", summary(lm2)$coefficients[-1,1]))
print(paste("and our estimate is", as.numeric(myest$beta[,1]), as.numeric(myest$beta[,2])))
print(paste("R std error estimate for first outcome is", summary(lm1)$coefficients[-1,2],
            "and for second outcome is", summary(lm2)$coefficients[-1,2],
            "and our estimate is", myest$beta.std[,1], myest$beta.std[,2]))
print(paste("R t value estimate for first outcome is", summary(lm1)$coefficients[-1,3],
            "and for second outcome is", summary(lm2)$coefficients[-1,3],
            "and our estimate is", myest$beta.t[,1], myest$beta.t[,2]))
print(paste("R pval for first outcome is", summary(lm1)$coefficients[-1,4],
            "and for second outcome is", summary(lm2)$coefficients[-1,4],
            "and our estimate is", myest$beta.pval[,1], myest$beta.pval[,2]))
```

---

blockStimulus *Block stimulus model for FMRI Data*

Description

Create the block BOLD stimulus for a given task indicator function. Current units in terms of volumes. Sort of a bug, should be in TR.

Usage

```
blockStimulus(scans = 1, onsets = c(1), durations = c(1), rt = 3)
```

Arguments

scans	number of scans
onsets	vector of onset times (in scans)
durations	vector of duration of ON stimulus in scans or seconds (if `!is.null(times)`)
rt	time between scans in seconds (TR)

Details

None

Value

Vector with dimension `c(scans, 1)`.

Author(s)

b avants <stnava@gmail.com>

References

Worsley, K.J., Liao, C., Aston, J., Petre, V., Duncan, G.H., Morales, F., Evans, A.C. (2002). A general statistical analysis for fMRI data. NeuroImage, 15:1-15.

Examples

```
# Example 1
  hrf <- blockStimulus(107, c(18, 48, 78), 15, 2)
```

`bold_correlation_matrix`

bold_correlation_matrix

Description

A data frame with an example square BOLD correlation matrix.

Usage

```
data(bold_correlation_matrix)
```

Format

A data frame square correlation matrix listed by AAL anatomical column names.

References

http://en.wikipedia.org/wiki/Automated_Anatomical_Labeling

Examples

```
data(bold_correlation_matrix)
```

`clusterTimeSeries` *Split time series image into k distinct images*

Description

Uses clustering methods to split a time series into similar subsets.

Usage

```
clusterTimeSeries(mat, krange = 2:10, nsvddims = NA, criterion = "asw")
```

Arguments

`mat`	input time series matrix
`krange`	k cluster range to explore
`nsvddims`	eg 2
`criterion`	for clustering see pamk

Value

matrix is output

Author(s)

Avants BB

Examples

```
## Not run:
  if (!exists("fn") ) fn<-getANTsRData("pcasl")
   # high motion subject
  asl<-antsImageRead(fn,4)
  tr<-antsGetSpacing(asl)[4]
  aslmean<-getAverageOfTimeSeries( asl )
  aslmask<-getMask(aslmean,lowThresh=mean(aslmean),cleanup=TRUE)
  omat<-timeseries2matrix(asl, aslmask )
  clustasl<-clusterTimeSeries( omat, krange=4:10 )
  for ( ct in 1:max(clustasl$clusters) )
    {
    sel<-clustasl$clusters != ct
    img<-matrix2timeseries( asl, aslmask, omat[sel,] )
    perf <- aslPerfusion( img,
      dorobust=0.9, useDenoiser=4, skip=10, useBayesian=0,
      moreaccurate=0, verbose=F, mask=aslmask )
    perfp <- list( sequence="pcasl", m0=perf$m0 )
    cbf <- quantifyCBF( perf$perfusion, perf$mask, perfp )
    }

## End(Not run)
```

combineNuisancePredictors *Combine and reduce dimensionality of nuisance predictors.*

Description

Combine and select nuisance predictors to maximize correlation between `inmat` and `target`.

Usage

```
combineNuisancePredictors(inmat, target, globalpredictors=NA,
  maxpreds=4, localpredictors=NA, method="cv", k=5, covariates=NA, ordered=F)
```

Arguments

`inmat`	Input predictor matrix.
`target`	Target outcome matrix.
`globalpredictors`	Global predictors of size `nrow(inmat)` by `n`, where `n` is the number of global predictors.
`maxpreds`	Maximum number of predictors to output.
`localpredictors`	Local predictor array of size `nrow(inmat)` by `ncol(inmat)` by `m`, where `m` is the number of local predictors.
`method`	Method of selecting noisy voxels. One of 'svd' or 'cv'. See `Details`.
`k`	Number of cross-validation folds.
`covariates`	Covariates to be considered when assessing prediction of `target`.
`ordered`	Can the predictors be assumed to be ordered from most important to least important, as in output from PCA? Computation is much faster if so.

Value

Array of size `nrow(aslmat)` by `npreds`, containing a timeseries of all the nuisance predictors. If `localpredictors` is not NA, array is of size `nrow(aslmat)` by `ncol(aslmat)` by `npreds`.

Author(s)

Benjamin M. Kandel, Brian B. Avants

Examples

```
set.seed(120)
simimg<-makeImage( c(10,10,10,20), rnorm(1000*20) )
moco <- antsMotionCalculation( simimg , moreaccurate=0)
# for real data use below
# moco <- antsMotionCalculation(getANTsRData("pcasl"))
aslmat <- timeseries2matrix(moco$moco_img, moco$moco_mask)
tc <- rep(c(0.5, -0.5), length.out=nrow(aslmat))
noise <- getASLNoisePredictors(aslmat, tc, 0.5 )
noise.sub <- combineNuisancePredictors(aslmat, tc, noise, 2)
```

compcor *Compcors the input matrix using SVD and returns the result.*

Description

Compcors the input matrix using SVD and returns the result.

Usage

```
compcor(fmri, ncompcor = 4, variance_extreme = 0.975, mask = NA,
  useimagemath = FALSE, randomSamples = 1, returnv = FALSE,
  returnhighvarmat = FALSE, returnhighvarmatinds = FALSE,
  highvarmatinds = NA)
```

Arguments

fmri	input fmri image or matrix
ncompcor	n compcor vectors
variance_extreme	high variance threshold e.g 0.95 for 95 percent
mask	optional mask for image
useimagemath	use the imagemath implementation instead
randomSamples	take this many random samples to speed things up
returnv	return the spatial vectors
returnhighvarmat	bool to return the high variance matrix
returnhighvarmatinds	bool to return the high variance matrix indices
highvarmatinds	index list

Value

dataframe of nuisance predictors is output

Author(s)

Avants BB

Examples

```
mat <- matrix( rnorm(50000) ,ncol=500)
compcorrdf<-compcor( mat )
```

computeDVARS *computeDVARS*

Description

compute the DVARS quality control metric

Usage

```
computeDVARS(boldMatrix)
```

Arguments

boldMatrix	matrix of bold signal

Value

DVARS vector.

Author(s)

Tustison NJ, Avants BB

Examples

```
mat <- matrix(c(0,1,2,0,0,1,2,2,2),ncol=3)
 dv<-computeDVARS(mat)
```

convolveImage *convolve one image with another*

Description

convolves images together

Usage

```
convolveImage(image, kernelImage, crop = TRUE)
```

Arguments

image	antsImage to convolve
kernelImage	antsImage acting as kernel
crop	boolean automatically crops kernelImage

Value

convimage

Author(s)

Brian B. Avants

Examples

```
fi<-antsImageRead( getANTsRData("r16") ,2)
convimg<-makeImage( c(3,3) , c(1,0,1,0,-4,0,1,0,1) )
convout<-convolveImage( fi, convimg )
convimg2<-makeImage( c(3,3) , c(0,1,0,1,0,-1,0,-1,0) )
convout2<-convolveImage( fi, convimg2 )
```

---

corw *produces a correlation matrix via weighted correlation.*

Description

Uses weighted regression to compute pairwise correlation matrix on input matrix - by columns.

Usage

```
corw(mat, weights)
```

Arguments

mat	input matrix
weights	input weights, size of nrow of matrix

Value

matrix is output

Author(s)

Avants BB

Examples

```
mat <- matrix( rnorm(100), nrow=10 )
wcmat<-corw( mat , weights = abs( nrorm( nrow(mat) ) ) )
```

createJacobianDeterminantImage *createJacobianDeterminantImage*

Description

Compute the jacobian determinant from a transformation file

Usage

```
createJacobianDeterminantImage(domainImg, tx, doLog = 0)
```

Arguments

domainImg	image that defines transformation domain
tx	deformation transformation file name
doLog	return the log jacobian

Value

jacobianImage

Author(s)

BB Avants

Examples

```
fi<-antsImageRead( getANTsRData("r16") ,2)
mi<-antsImageRead( getANTsRData("r64") ,2)
fi<-resampleImage(fi,c(128,128),1,0)
mi<-resampleImage(mi,c(128,128),1,0)
mytx<-antsRegistration(fixed=fi , moving=mi, typeofTransform = c("SyN") )
jac<-createJacobianDeterminantImage(fi,mytx$fwdtransforms[[1]],1)
# plot(jac)
```

cropImage *crop a sub-image via a mask*

Description

uses a label image to crop a smaller image from within a larger image

Usage

```
cropImage(image, labelImage, label = 1)
```

Arguments

image	antsImage to crop
labelImage	antsImage with label values. If not supplied, estimated from data.
label	the label value to use

Value

subimage

Author(s)

Brian B. Avants, Nicholas J. Tustison

Examples

```
fi <- antsImageRead(getANTsRData("r16"))
cropped <- cropImage(fi)
cropped <- cropImage(fi, fi, 250)
```

---

cropIndices *crop a sub-image by image indices*

Description

create a proper antsImage sub-image by indexing the image with indices. this is similar to but different from array sub-setting in that the resulting sub-image can be decropped back into its place without having to store its original index locations explicitly.

Usage

```
cropIndices(image, lowerind, upperind)
```

Arguments

image	antsImage to crop
lowerind	vector of lower index, should be length image dimensionality
upperind	vector of upper index, should be length image dimensionality

Value

subimage

Author(s)

Brian B. Avants, Nicholas J. Tustison

Examples

```
fi <- antsImageRead( getANTsRData("r16"))
cropped <- cropIndices( fi, c(2,10), c(5,15) )
cropped<-smoothImage( cropped, 5 )
decropped<-decropImage( cropped, fi )
```

crossvalidatedR2 *Cross-Validated R^2 value*

Description

Computes an R^2 value for predicting an outcome measure using a k-fold cross-validation scheme.

Usage

```
crossvalidatedR2(x, y, ngroups=5, covariates=NA, fast=F)
```

Arguments

x	Input predictor matrix.
y	Target dependent variable.
ngroups	Number of cross-validation folds to use or the fold labels themselves, equal to the length of y. e.g. c(1,1,1,2,2,2...)
covariates	Covariate predictors.
fast	Use low-level `lm.fit` instead of `lm`. Much faster, but less error checking.

Value

Matrix of size `ngroups` by `ncol(x)`, which each row corresponding to one fold and the columns corresponding to the R2 values for each predictor.

Author(s)

Brian B Avants, Benjamin M. Kandel

Examples

```
set.seed(300)
ncol <- 30
nrow <- 20
covariate <- sin((1:nrow)*2*pi/nrow)
x <- matrix(rep(NA, nrow*ncol), nrow=nrow)
xsig <- seq(0,1,length.out=nrow)
y <- xsig + covariate + rnorm(nrow, sd=0.5)
for(i in 1:ncol){
  x[, i] <- xsig + rnorm(nrow, sd=i/ncol)
}
r2 <- crossvalidatedR2(x, y, covariates=covariate)
```

`cvEigenanatomy` *Cross-validation method for eigenanatomy decompositions.*

Description

Perform cross-validation on an image set using eigencomponents to predict an outcome variable.

Usage

```
cvEigenanatomy(demog, images, outcome, ratio = 10, mask = NA,
  sparseness = 0.01, nvecs = 50, its = 5, cthresh = 250, ...)
```

Arguments

`demog`	Demographics information that includes outcome and (optional) covariates.
`images`	n by p input image matrix, where n is the number of subjects and p is the number of voxels.
`outcome`	Name of outcome variable. Must be present in `demog`.
`ratio`	If greater than 1, number of folds for cross-validation. If less than 1, one testing-training step will be performed, using `ratio` of the data for training and the rest for testing.
`mask`	Mask image of type `antsImage`.
`sparseness`	Desired level of sparsity in decomposition.
`nvecs`	Number of eigenvectors to use in decomposition.
`its`	Number of iterations for decomposition.
`cthresh`	Cluster threshold for decomposition.
`...`	Additional options passed to `regressProjections`.

Value

A result, or (if ratio > 1) list of results, from `regressProjection`.

Author(s)

Kandel BM and Avants B

Examples

```
## Not run:
# generate simulated outcome
nsubjects <- 100
x1 <- seq(1, 10, length.out=nsubjects) + rnorm(nsubjects, sd=2)
x2 <- seq(25, 15, length.out=nsubjects) + rnorm(nsubjects, sd=2)
outcome <- 3 * x1 + 4 * x2 + rnorm(nsubjects, sd=1)
# generate simulated images with outcome predicted
# by sparse subset of voxels
voxel.1 <- 3 * x1 + rnorm(nsubjects, sd=2)
voxel.2 <- rnorm(nsubjects, sd=2)
voxel.3 <- 2 * x2 + rnorm(nsubjects, sd=2)
voxel.4 <- rnorm(nsubjects, sd=3)
```

```
input   <- cbind(voxel.1, voxel.2, voxel.3, voxel.4)
mask    <- as.antsImage(matrix(c(1,1,1,1), nrow=2))
# generate sample demographics that do not explain outcome
age <- runif(nsubjects, 50, 75)
demog <- data.frame(outcome=outcome, age=age)
result <- cvEigenanatomy(demog, input, outcome, ratio=5, mask,
            sparseness=0.25, nvecs=4)

## End(Not run)
```

`decropImage` *decrop a sub-image back into the full image*

Description

the inverse function for `cropImage`

Usage

```
decropImage(croppedImage, fullImage)
```

Arguments

`croppedImage`	cropped antsImage
`fullImage`	antsImage to put back into

Value

decroppedImage

Author(s)

Brian B. Avants, Nicholas J. Tustison

Examples

```
fi <- antsImageRead( getANTsRData("r16"))
mask <- getMask( fi )
cropped <- cropImage( fi, mask, 1 )
cropped <- smoothImage( cropped, 1)
decropped <- decropImage( cropped , fi )
```

DesikanKillianyTourville *DesikanKillianyTourville*

Description

A data frame label numbers and names for the neuroanatomical DesikanKillianyTourville labels.

Usage

```
data(DesikanKillianyTourville)
```

Format

A data frame listing the following variables.

`label_num` Numerical label value.

`label_name` Shorthand anatomical value.

References

http://www.ncbi.nlm.nih.gov/pubmed/23227001

Examples

```
data(DesikanKillianyTourville)
```

---

eigSeg *Segmentation for eigenanatomy.*

Description

Segment a mask into regions based on the max value in an image list. At a given voxel the segmentation label will contain the index to the image that has the largest value. If the 3rd image has the greatest value, the segmentation label will be 3 at that voxel.

Usage

```
eigSeg(mask = NA, imgList = NA, applySegmentationToImages = TRUE)
```

Arguments

`mask`	D-dimensional mask > 0 defining segmentation region.
`imgList`	list containing antsImages or filenames pointing to antsImages.
`applySegmentationToImages`	boolean determines if original image list is modified by the segmentation.

Value

segmentation image.

Author(s)

Avants BB

Examples

```
mylist<-list( antsImageRead( getANTsRData("r16") ),
  antsImageRead( getANTsRData("r27") ),
  antsImageRead( getANTsRData("r85") ) )
myseg<-eigSeg( getMask( mylist[[1]] ) , mylist )
```

---

exemplarInpainting *Uses example images to inpaint or approximate an existing image.*

Description

Employs a robust regression approach to learn the relationship between a sample image and a list of images that are mapped to the same space as the sample image. The regression uses data from an image neighborhood.

Usage

```
exemplarInpainting(img, paintMask, imageList, featureRadius = 2,
  scaleInpaintIntensity = 0, sharpen = FALSE, feather = 1,
  predalgorithm = "lm", debug = FALSE)
```

Arguments

img	antsImage to be approximated / painted
paintMask	painting mask with values 1 or values 1 and 2 - if there is a 2 then it will learn from label 1 to paint label 2. should cover the brain.
imageList	a list containing antsImages
featureRadius	- radius of image neighborhood e.g. 2
scaleInpaintIntensity	- brighter or darker painted voxels, default of 0 sets this parameter automatically
sharpen	- sharpen the approximated image
feather	- value (e.g. 1) that helps feather the mask for smooth blending
predalgorithm	- string svm or lm
debug	- TRUE or FALSE

Value

inpainted image

Author(s)

Brian B. Avants

Examples

```
set.seed(123)
fi<-abs(replicate(100, rnorm(100)))
fi[1:10,]<-fi[,1:10]<-fi[91:100,]<-fi[,91:100]<-0
mask<-fi
mask[ mask > 0 ]<-1
mask2<-mask
mask2[11:20,11:20]<-2
mask<-as.antsImage( mask , "float" )
fi<-as.antsImage( fi , "float" )
fi<-smoothImage(fi,3)
mo<-as.antsImage( replicate(100, rnorm(100)) , "float" )
mo2<-as.antsImage( replicate(100, rnorm(100)) , "float" )
ilist<-list(mo,mo2)
painted<-exemplarInpainting(fi,mask,ilist)
mask2<-as.antsImage( mask2 , "float" )
painted2<-exemplarInpainting(fi,mask2,ilist)
# just use 1 image, so no regression is performed
painted3<-exemplarInpainting(fi,mask2, list(ilist[[1]]))
```

`extractSlice` *extract a slice from an image*

Description

extract a slice from an image and return an image of dimensionality, d-1

Usage

```
extractSlice(image, slice, direction)
```

Arguments

`image`	antsImage to crop
`slice`	which slice, integer
`direction`	which axis, integer

Value

antsImage of dimension - 1

Author(s)

Brian B. Avants, Nicholas J. Tustison

Examples

```
fi <- makeImage( c(10,10,10), rnorm(1000) )
slice <- extractSlice( fi, 1, 1 )
```

`filterfMRIforNetworkAnalysis` *Basic pre-processing for BOLD or ASL-based network analysis.*

Description

This function works for either raw BOLD time-series data, ASL-based BOLD time-series data or ASL-based CBF time series data. In all 3 cases, this function performs motion-correction, factoring out motion and compcor nuisance paramters, frequency filtering and masking. The output contains the filtered time series (matrix form), the mask and a vector of temporal signal variance. Some ASL MR sequences allow network analysis of either BOLD or ASL signal. See "Implementation of Quantitative Perfusion Imaging Techniques for Functional Brain Mapping using Pulsed Arterial Spin Labeling" by Wong et al, 1997 for an overview. This function employs "surround" techniques for deriving either CBF or BOLD signal from the input ASL. This is a WIP.

Usage

```
filterfMRIforNetworkAnalysis(aslmat, tr, freqLo = 0.01, freqHi = 0.1,
  cbfnetwork = "ASLCBF", mask = NA, labels = NA, graphdensity = 0.5,
  seg = NA, useglasso = NA, nuisancein = NA, usesvd = FALSE,
  robustcorr = FALSE)
```

Arguments

`aslmat`	The filename to an antsr image or pointer to an antsr image
`tr`	The sequence's TR value , typically 3 or 4.
`freqLo`	The lower frequency limit, e.g. 0.01 in band-pass filter
`freqHi`	The higher frequency limit, e.g. 0.1 in band-pass filter
`cbfnetwork`	"ASLCBF" A string dictating whether to do nothing special (standard BOLD) or get CBF (ASLCBF) or BOLD (ASLBOLD) signal from ASL
`mask`	the mask image
`labels`	the label image
`graphdensity`	desired density
`seg`	a segmentation image
`useglasso`	use sparse inverse covariance for network estimation
`nuisancein`	nuisance variable data frame
`usesvd`	bool, to reduce nuisance variables
`robustcorr`	bool

Value

output is a list containing "filteredTimeSeries" "mask" "temporalvar"

or

1 – Failure

Author(s)

Avants BB

Examples

```
## Not run:
if (!exists("fn") ) fn<-getANTsRData("pcasl")
img<-antsImageRead( fn )
mask<-getMask( getAverageOfTimeSeries( img )  )
fmat<-timeseries2matrix( img, mask )
myres<-filterfMRIforNetworkAnalysis(fmat,tr=4,0.01,0.1,cbfnetwork="BOLD", mask = mask )

## End(Not run)
```

---

frequencyFilterfMRI *Band pass filtering for BOLD image.*

Description

This function works for a BOLD time-series data

Usage

```
frequencyFilterfMRI(boldmat, tr, freqLo = 0.01, freqHi = 0.1,
  opt = "butt")
```

Arguments

boldmat	Time series matrix for bold image
tr	The sequence's TR value , typically 3 or 4.
freqLo	The lower frequency limit, e.g. 0.01 in band-pass filter
freqHi	The higher frequency limit, e.g. 0.1 in band-pass filter
opt	one of 'trig','butt','stl' Type of filter to use: butterworth, trigonometric, stl.

Value

output is the filtered time series.

Author(s)

Avants BB

Examples

```
fmat<-replicate(1000, rnorm(200))
k<-1
for ( ftype in c("butt","stl","trig") ) {
  myres<-frequencyFilterfMRI( fmat, tr = 4, freqLo = 0.01, freqHi = 0.05, opt = ftype )
  comparemat<-cbind(fmat[,k],myres[,k])
  plot(ts(comparemat),main=ftype)
  Sys.sleep(0.3)
}
```

geoSeg *brain segmentation based on geometric priors*

Description

uses topological constraints to enhance accuracy of brain segmentation

Usage

```
geoSeg(img, brainmask, priors, seginit, vesselopt = "none", vesselk = 2,
  gradStep = 1.25, mrfval = 0.1, atroposits = 10, jacw = NA,
  beta = 0.9)
```

Arguments

img	input image or list of images (multiple features) where 1st image would typically be the primary constrast
brainmask	binary image
priors	spatial priors, assume first is csf, second is gm, third is wm
seginit	a previously computed segmentation which should have the structure of atropos or kmeansSegmentation output
vesselopt	one of bright, dark or none
vesselk	integer for kmeans vessel-based processing
gradStep	scalar for registration
mrfval	e.g. 0.05 or 0.1
atroposits	e.g. 5 iterations
jacw	precomputed diffeo jacobian
beta	for sigma transformation (thksig output variable)

Value

list of segmentation result images

Author(s)

Brian B. Avants

Examples

```
## Not run:
img = antsImageRead( getANTsRData("simple") ,2)
img = n3BiasFieldCorrection( img , 4 )
img = n3BiasFieldCorrection( img , 2 )
bmk = getMask( img )
segs <- kmeansSegmentation( img, 3, bmk )
priors = segs$probabilityimages
seg = geoSeg( img, bmk, priors )

## End(Not run)
```

getANTsRData *getANTsRData*

Description

Downloads antsr test data

Usage

```
getANTsRData(fileid, usefixedlocation = FALSE)
```

Arguments

fileid	one of the permitted file ids or pass "show" to list all valid possibilities. Note that most require internet access to download.
usefixedlocation	directory to which you download files

Value

filename string

Author(s)

Avants BB

Examples

```
fi <- getANTsRData( "r16" )
```

getASLNoisePredictors *Get nuisance predictors from ASL images*

Description

Get nuisance predictors from ASL images

Usage

```
getASLNoisePredictors(aslmat, tc, noisefrac = 0.1, polydegree = 3, k = 5,
  npreds = 12, method = "noisepool", covariates = NA,
  noisepoolfun = max)
```

Arguments

aslmat	ASL input matrix.
tc	Tag-control sawtooth pattern vector.
noisefrac	Fraction of data to include in noise pool.
polydegree	Degree of polynomial for detrending. A value of 0 indicates no detrending.
k	Number of cross-validation folds.
npreds	Number of predictors to output.
method	Method of selecting noisy voxels. One of 'compcor' or 'noisepool'. See Details.
covariates	Covariates to be considered when assessing prediction of tc pattern.
noisepoolfun	Function used for aggregating R^2 values.

Value

Matrix of size nrow(aslmat) by npreds, containing a timeseries of all the nuisance predictors.

Author(s)

Brian B. Avants, Benjamin M. Kandel

Examples

```
# for real data do img<-antsImageRead(getANTsRData("pcasl"),4)
set.seed(120)
img<-makeImage( c(10,10,10,20), rnorm(1000*20)+1 )
moco <- antsMotionCalculation(img,moreaccurate=0)
aslmat <- timeseries2matrix(moco$moco_img, moco$moco_mask)
tc <- rep(c(0.5, -0.5), length.out=nrow(aslmat))
noise <- getASLNoisePredictors(aslmat, tc)
```

---

getAverageOfTimeSeries *getAverageOfTimeSeries*

Description

Returns a 3D average of a 4D time series.

Usage

```
getAverageOfTimeSeries(img)
```

Arguments

img	input 4D image

Value

3D ants image is output

Author(s)

Avants BB

Examples

```
img<-as.antsImage( array(data = rep(0,10^4), dim = c(10,10,10,10) ) )
avg<-getAverageOfTimeSeries( img )
```

---

getCentroids *Convert an image to the geometric centroids of its signal*

Description

Reduces a variate/statistical/network image to a set of centroids describing the center of each stand-alone non-zero component in the image.

Usage

```
getCentroids(img, clustparam = 250, outprefix = NA)
```

Arguments

img	the image to reduce to centroids - presumably some kind of statistical or network map
clustparam	look at regions greater than or equal to this size
outprefix	prefix if you want to output to a file

Value

the centroids are output in matrix of size npoints by 3

Author(s)

Avants BB

Examples

```
img<-antsImageRead( getANTsRData( "r16" ) )
img<-thresholdImage( img, 90, 120 )
img<-labelClusters( img, 10 )
cents<-getCentroids( img  )
```

getfMRInuisanceVariables *Extract generic fMRI nuisance variables for ASL or BOLD*

Description

Will motion correct, run compcorr and estimate global signal. Outputs a list with the time series data matrix (time by voxels), motion and other nuisance variables, global signal (for BOLD or ASL), the mask and the average time series image. Meant to be used before filterfMRIforNetworkAnalysis or any other results-oriented processing.

Usage

```
getfMRInuisanceVariables(fmri, maskThresh = 500, moreaccurate = 1,
  mask = NA)
```

Arguments

fmri	input antsImage or filename
maskThresh	will use this intensity threshold to estimate a mask otherwise will use the mask passed in
moreaccurate	zero, one or two with increasing accuracy/computation
mask	binary image masking the intput image, precedent over mask thresh

Value

outputs list described above.

Author(s)

Avants BB

Examples

```
## Not run:
if (!exists("fn") ) fn<-getANTsRData("pcasl")
pcasl<-antsImageRead( fn )
aslmean<-getAverageOfTimeSeries( pcasl )
aslmask<-getMask(aslmean)
ee<-getfMRInuisanceVariables( pcasl, mask = aslmask ,
        moreaccurate=F )

## End(Not run)
```

getMask *Get Mask*

Description

Get a binary mask image from the given image after thresholding.

Usage

```
getMask(img, lowThresh, highThresh, cleanup = 2)
```

Arguments

img	Input image. Can be an antsImage of 2, 3 or 4 dimensions. If img is NULL, a file chooser dialog will appear.
lowThresh	An inclusive lower threshold for voxels to be included in the mask.
highThresh	An inclusive upper threshold for voxels to be included in the mask.
cleanup	If >0, morphological operations will be applied to clean up the mask by eroding away small or weakly-connected areas, and closing holes.

Details

If cleanup is >0, the following steps are applied

1. Erosion with radius 2 voxels
2. Retain largest component
3. Dilation with radius 1 voxel
4. Morphological closing

These functions are available in iMath, see the operations "ME", "GetLargestComponent", "MD", "FillHoles".

Value

Object of type antsImage containing the mask image. The voxel intensities will be binarized, 1 for voxels in the mask and 0 outside.

Author(s)

Shrinidhi KL, Cook PA, Avants BB

Examples

```
img<-antsImageRead( getANTsRData("r16"))
mask<-getMask( img )
```

`getMultivariateTemplateCoordinates` *Label multivariate components by an anatomical coordinate system.*

Description

This function will provide a mapping that labels the list of input images and each of their blobs.

Usage

```
getMultivariateTemplateCoordinates(imageSetToBeLabeledIn, templateWithLabels,
  labelnames = NA, outprefix = NA, convertToTal = FALSE, pvals = NA,
  threshparam = 1, clustparam = 250, identifier)
```

Arguments

`imageSetToBeLabeledIn`	a template paired with (most likely) the output of a multivariate sparse decomposition or (alternatively) could be just a statistical map with zeroes in non-interesting areas
`templateWithLabels`	e..g the mni image and brodmann label set
`labelnames`	a list of names for the labels
`outprefix`	if output to a file, provide file prefix
`convertToTal`	bool, return talairach coordinates
`pvals`	the already computed pvalue for each component
`threshparam`	for pvals
`clustparam`	for clusters
`identifier`	unique ID for this study

Details

Uses getTemplateCoordinates as a sub-routine.

TBN

Value

The output point coordinates are in approximate Talairach / MNI (or whatever) template space.

Author(s)

Avants, BB

Examples

```
## Not run:
 tem<-antsImageRead( getANTsRData("ch2") )
 temlab<-antsImageRead( getANTsRData("ch2b")  )
 temlab2<-antsImageRead( getANTsRData("ch2a")  )
 # try getANTsRData if you have www access
 mymni<-list( antsImageRead(getANTsRData("mni"),3),
       antsImageRead(getANTsRData("mnib"),3),
       antsImageRead(getANTsRData("mnia"),3) )
 mytem<-list( smoothImage(tem,3) ,temlab,temlab2)
 # mynetworkdescriptor<-getMultivariateTemplateCoordinates(
 #  mytem, mymni , convertToTal = TRUE , pvals=c(0.01,0.05) )

## End(Not run)
```

---

getNeighborhoodAtVoxel

Get a hypercube neighborhood at a voxel

Description

Get the values in a local neighborhood of an antsImage.

Usage

```
getNeighborhoodAtVoxel(image, center, kernel, physical.coordinates = FALSE)
```

Arguments

image	Image object of S4 class antsImage to get values from.
center	array of indices for neighborhood center
kernel	either an array of values for neighborhood radius (in voxels) or a binary array of the same dimension as the image, specifying the shape of the neighborhood to extract
physical.coordinates	a logical indicating if voxel indices and offsets should be in voxel or physical coordinates

Value

a list

- valuesnumeric vector of values
- indicesmatrix providing the coordinates for each value

Author(s)

Duda JT

Examples

```
img<-makeImage(c(10,10),rnorm(100))
center <- dim(img)/2
radius <- rep(3,2)
nhlist <- getNeighborhoodAtVoxel(img,center,radius)
kernel <- 1*(rnorm(49)>0)
dim(kernel) <- c(7,7)
randlist <- getNeighborhoodAtVoxel(img,center,kernel)
```

`getNeighborhoodInMask` *Get neighborhoods for voxels within mask*

Description

Get neighborhoods for voxels within mask

Usage

```
getNeighborhoodInMask(image, mask, radius, physical.coordinates = FALSE,
  boundary.condition = "NA", spatial.info = FALSE, get.gradient = FALSE)
```

Arguments

`image`	image object of S4 class `antsImage` to get values from.
`mask`	image object of S4 class `antsImage` indicating which voxels to examine. Each voxel > 0 will be used as the center of a neighborhood
`radius`	array of values for neighborhood radius (in voxels)
`physical.coordinates`	logical indicating if voxel indices and offsets should be in voxel or physical coordinates
`boundary.condition`	string indicating how to handle voxels in a neighborhood, but not in the mask. See `Details`.
`spatial.info`	a boolean indicating of voxel locations and neighborhood offsets should be returned along with pixel values.
`get.gradient`	a boolean indicating if a matrix of gradients (at the center voxel) should be returned in addition to the value matrix (WIP)

Details

`boundary.condition` should be one of:

- `NA`: Fill values with `NA`.
- `image`: Use image value, even if not in mask.
- `mean`: Use man of all non-`NA` values for that neighborhood.

Value

if spatial.info is false: a matrix of pixel values where the number of rows is the size of the neighborhood and there is a column for each voxel

if spatial.info is true, a list containing three matrices:

- values: matrix of pixel values where the number of rows is the size of the neighborhood and there is a column for each voxel.
- indices: matrix providing the center coordinates for each neighborhood
- offsets: matrix providing the offsets from center for each voxel in a neighborhood

Author(s)

Duda JT

Examples

```
r16 <- getANTsRData("r16")
r16 <- antsImageRead(r16,2)
mask <- getMask(r16,lowThresh=mean(r16),cleanup=1)
radius <- rep(2,2)
mat <- getNeighborhoodInMask(r16,mask,radius)
```

---

getPixels *Get Pixels*

Description

Get pixel values from an 'antsImage'.

Usage

```
getPixels(x, i = NA, j = NA, k = NA, l = NA)
```

Arguments

x	Image object of S4 class 'antsImage' to get values from.
i	indices in first dimension
j	indices in second dimension
k	indices in third dimension
l	indices in forth dimension

Value

array of pixel values

Examples

```
img<-makeImage(c(10,10),rnorm(100))
pixel<-getPixels(img,i=c(1,2),j=1)
```

`getTemplateCoordinates` *Define an anatomical coordinate system in a new image based on a template*

Description

This function will provide a mapping that labels an input image and its blobs.

Usage

```
getTemplateCoordinates(imagePairToBeLabeled, templatePairWithLabels,
  labelnames = NA, outprefix = NA, convertToTal = FALSE)
```

Arguments

`imagePairToBeLabeled`	e.g. the template image and the activation map
`templatePairWithLabels`	e..g the mni image and brodmann label set
`labelnames`	a list of names for the labels
`outprefix`	if output to a file, provide file prefix
`convertToTal`	bool, return talairach coordinates

Details

Uses Matthew Brett's mni2tal to get the final Talairach coordinates from MNI space.

This is a standard approach but it's not very accurate.

Value

The output point is in approximate template space.

Author(s)

Avants, BB

Examples

```
## Not run:
#
# ch2bet is available in chris rordens mricron
#  but you can do something with any other image
#  e.g. a statistical image
#
  tem<-antsImageRead(getANTsRData("ch2"),3)
  clust <- antsImageClone( tem )
  clust[ tem < 80 ]<- 0
  clust[ tem > 90 ]<- 0
  clust[  tem > 80 & tem < 90 ]<- 1
  clust<-iMath(clust,"ME",1)  # erosion
```

```
  clust <- labelClusters( clust , minClusterSize=30, minThresh=1, maxThresh=1)
  if ( ! exists("mymni") ) {
  # try getANTsRData if you have www access
    mymni<-list( antsImageRead(getANTsRData("mni"),3),
                 antsImageRead(getANTsRData("mnib"),3),
                 antsImageRead(getANTsRData("mnia"),3) )
  }
  template_cluster_pair<-list(tem,clust)
  gcoords<-getTemplateCoordinates( template_cluster_pair ,
      mymni , convertToTal = TRUE )
# output will be like
# > gcoords$templatepoints
#     x   y   z t label Brodmann                AAL
# 1  -12  13  -3 0     1        0           Caudate_R
# 2   13  16   5 0     2        0           Caudate_L
#
# can also use a white matter label set ...
#

## End(Not run)
```

---

hemodynamicRF *Linear Model for FMRI Data*

Description

Create the expected BOLD response for a given task indicator function. Borrowed from the fmri package.

Usage

```
hemodynamicRF(scans = 1, onsets = c(1), durations = c(1), rt = 3,
  times = NULL, mean = TRUE, a1 = 6, a2 = 12, b1 = 0.9, b2 = 0.9,
  cc = 0.35)
```

Arguments

scans	number of scans
onsets	vector of onset times (in scans)
durations	vector of duration of ON stimulus in scans or seconds (if `!is.null(times)`)
rt	time between scans in seconds (TR)
times	onset times in seconds. If present `onsets` arguments is ignored.
mean	logical. if TRUE the mean is substracted from the resulting vector
a1	parameter of the hemodynamic response function (see details)
a2	parameter of the hemodynamic response function (see details)
b1	parameter of the hemodynamic response function (see details)
b2	parameter of the hemodynamic response function (see details)
cc	parameter of the hemodynamic response function (see details)

Details

The functions calculates the expected BOLD response for the task indicator function given by the argument as a convolution with the hemodynamic response function. The latter is modelled by the difference between two gamma functions as given in the reference (with the defaults for a1, a2, b1, b2, cc given therein):

$$\left(\frac{t}{d_1}\right)^{a_1} \exp\left(-\frac{t-d_1}{b_1}\right) - c\left(\frac{t}{d_2}\right)^{a_2} \exp\left(-\frac{t-d_2}{b_2}\right)$$

The parameters of this function can be changed through the arguments `a1`, `a2`, `b1`, `b2`, `cc`.

The dimension of the function value is set to `c(scans,1)`.

If `!is.null(times)` durations are specified in seconds.

If `mean` is TRUE (default) the resulting vector is corrected to have zero mean.

Value

Vector with dimension `c(scans, 1)`.

Author(s)

Karsten Tabelow <tabelow@wias-berlin.de>

References

Worsley, K.J., Liao, C., Aston, J., Petre, V., Duncan, G.H., Morales, F., Evans, A.C. (2002). A general statistical analysis for fMRI data. NeuroImage, 15:1-15.

Polzehl, J. and Tabelow, K. (2007) *fmri: A Package for Analyzing fmri Data*, R News, 7:13-17 .

Examples

```
# Example 1
  hrf <- hemodynamicRF(107, c(18, 48, 78), 15, 2)
  # Example 2: effect of varying parameter cc
  cc<-round(seq(0,1,length.out=10),2)
  nlev<-length(cc)
  cscale<-rgb(seq(0,1,length.out=nlev),seq(1,0,length.out=nlev),0,1)
  mat<-matrix(NA,nrow=nlev,ncol=20)
  for (i in 1:nlev) {
    hrf<-ts( hemodynamicRF(scans=20, onsets=1, durations=2, rt=1,cc=cc[i],a1=4,a2=3))
    mat[i,]<-hrf
  }
  matplot(seq(1,20),t(mat),l,lwd=1,col=cscale,xlab=Time,ylab=Response,main=Parameter cc)
  legend(x=topleft,legend=cc,text.col=cscale)
  # Example 3: effect of varying parameter a1
  a1<-seq(1,10)
  nlev<-length(a1)
  cscale<-rgb(seq(0,1,length.out=nlev),seq(1,0,length.out=nlev),0,1)
  mat<-matrix(NA,nrow=nlev,ncol=20)
```

```
for (i in 1:nlev) {
  hrf<-ts( hemodynamicRF(scans=20, onsets=1, durations=2, rt=1,a1=a1[i],a2=3))
  mat[i,]<-hrf
}
matplot(seq(1,20),t(mat),l,lwd=1,col=cscale,xlab=Time,ylab=Response,main=Parameter a1)
legend(x=topleft,legend=a1,text.col=cscale)
# Example 4: effect of varying parameter a2
a2<-seq(1,10)
nlev<-length(a2)
cscale<-rgb(seq(0,1,length.out=nlev),seq(1,0,length.out=nlev),0,1)
mat<-matrix(NA,nrow=nlev,ncol=20)
for (i in 1:nlev) {
  hrf<-ts( hemodynamicRF(scans=20, onsets=1, durations=2, rt=1,a1=4,a2=a2[i]))
  mat[i,]<-hrf
}
matplot(seq(1,20),t(mat),l,lwd=1,col=cscale,xlab=Time,ylab=Response,main=Parameter a2)
legend(x=topleft,legend=a2,text.col=cscale)
# Example 5: effect of varying parameter b1
b1<-seq(0.4,1.3,by=0.1)
nlev<-length(b1)
cscale<-rgb(seq(0,1,length.out=nlev),seq(1,0,length.out=nlev),0,1)
mat<-matrix(NA,nrow=nlev,ncol=20)
for (i in 1:nlev) {
  hrf<-ts( hemodynamicRF(scans=20, onsets=1, durations=2, rt=1,a1=4,a2=3, b1=b1[i]))
  mat[i,]<-hrf
}
matplot(seq(1,20),t(mat),l,lwd=1,col=cscale,xlab=Time,ylab=Response,main=Parameter b1)
legend(x=topleft,legend=b1,text.col=cscale)
# Example 6: effect of varying parameter b2
b2<-seq(0.4,1.3,by=0.1)
nlev<-length(b2)
cscale<-rgb(seq(0,1,length.out=nlev),seq(1,0,length.out=nlev),0,1)
mat<-matrix(NA,nrow=nlev,ncol=20)
for (i in 1:nlev) {
  hrf<-ts( hemodynamicRF(scans=20, onsets=1, durations=2, rt=1,a1=4,a2=3, b2=b2[i]))
  mat[i,]<-hrf
}
matplot(seq(1,20),t(mat),l,lwd=1,col=cscale,xlab=Time,ylab=Response,main=Parameter b2)
legend(x=topleft,legend=b2,text.col=cscale)
```

`iBind` *iBind*

Description

bind two images along their edge

Usage

```
iBind(img1, img2, along = NA)
```

Arguments

`img1`	input object, an antsImage

img2	second antsImage, same size as first
along	dimension to bind along

Author(s)

BB Avants

Examples

```
fi<-antsImageRead( getANTsRData("r16") , 2 )
mi<-antsImageRead( getANTsRData("r62") , 2 )
bi<-iBind( fi, mi , 1 )
multismoo<- fi %>% iBind( smoothImage(fi,2) ) %>% iBind( smoothImage(fi,4) )
```

---

icawhiten *Simple icawhitening function.*

Description

Icawhitens the input matrix using SVD and returns the result.

Usage

```
icawhiten(Xin, n.comp, verbose = FALSE)
```

Arguments

Xin	input matrix
n.comp	number of components on which to project
verbose	bool

Value

matrix is output

Author(s)

Avants BB, fastICA package (see CRAN)

Examples

```
mat <- matrix(c(rep(1,100),rep(0,200)),ncol=50)
wmat<-icawhiten( mat, 2 )
```

image2ClusterImages *Converts an image to several independent images.*

Description

Produces a unique image for each connected component 1 through N of size > minClusterSize

Usage

```
image2ClusterImages(x, minClusterSize = 50, minThresh = 1e-06,
  maxThresh = 1)
```

Arguments

x	input antsImage e.g. a statistical map
minClusterSize	throw away clusters smaller than this value
minThresh	threshold to a statistical map
maxThresh	threshold to a statistical map

Value

the original image broken into a list of cluster images is the output

Author(s)

Avants BB

Examples

```
## Not run:
img <- antsImageRead(  getANTsRData(r16) )
img <- thresholdImage( img, 1 , Inf )
imageclusterlist<-image2ClusterImages( img )

## End(Not run)
```

imageFileNames2ImageList *Simple imageFileNames2ImageListing function.*

Description

ImageFileNames2ImageLists converts the input list of file names to a list containing antsImages.

Usage

```
imageFileNames2ImageList(x, dim)
```

Arguments

x	input file name list
dim	img dimensionality (optional)

Value

a list like this : mylist<-list(img1, img2 , etcetera) is the output ... img* are antsImages

Author(s)

Avants BB

Examples

```
## Not run:
  # gglb<-paste(gmView1vec*.nii.gz,sep=)
  # gfnl<-imageFileNames2ImageList( list.files(path=statdir,
  #   pattern = glob2rx(gglb),full.names = T,recursive = T) )

## End(Not run)
```

---

imageListToMatrix *Read Images into a Matrix*

Description

Read images into rows of a matrix.

Usage

```
imageListToMatrix(imageList, mask)
```

Arguments

imageList	A character vector containing a list of image files to read, in order - these are image objects, not file names.
mask	An antsImage containing a binary mask, voxels in the mask are placed in the matrix. If not provided, estimated from first image in list.

Value

A matrix containing the masked data, the result of calling as.numeric(image, mask) on each input image.

Author(s)

Cook PA, Avants B, Kandel BM

See Also

matrixToImages, getMask

Examples

```
img <- antsImageRead(getANTsRData(r16), 2)
 imglist <- list()
 nvox <- dim(img)[1] * dim(img)[2]
 nsubj <- 50
 for(ii in 1:nsubj){
   imglist[ii] <- img + rnorm(nvox, sd=mean(img[img!=0]))
 }
 mask <- getMask(img)
 imgmat <- imageListToMatrix(imglist, mask)
```

`imageMath` *R access to the ANTs program ImageMath*

Description

Perform math-operations on the given image. The first argument should be dimensionality, the 2nd the output image, the third a string describing the operation. The fourth should be in the input image. Additional parameters should be specific for each function. See the the imageMath help in ANTs.

Usage

```
imageMath(...)
```

Arguments

...	all parameters

Author(s)

Shrinidhi KL

Examples

```
fi<-antsImageRead( getANTsRData("r16") ,2)
mask<-getMask(fi)
imageMath( 2 , fi , "GD", fi , 1 )  # gray matter dilation
imageMath( 2 , mask , "Neg", mask )  # negate
imageMath( 2 , mask , "D", mask )  # distance transform
plot(mask)
```

imagesToMatrix *Read Images into a Matrix*

Description

Read images into rows of a matrix, given a mask - much faster for large datasets as it is based on C++ implementations.

Usage

```
imagesToMatrix(imageList, mask)
```

Arguments

imageList	A character vector containing a list of image files to read, in order.
mask	An antsImage containing a binary mask, voxels in the mask are placed in the matrix.

Value

A matrix containing the masked data, the result of calling as.numeric(image, mask) on each input image.

Author(s)

Cook PA, Avants BB (C++ version)

See Also

matrixToImages, getMask

Examples

```
# make some simulated images and convert them to a matrix

  n <- 2
  tdir<-tempdir()
  for ( i in 1:n ) {
    simimg<-as.antsImage( replicate(64, rnorm(64) ) )
    antsImageWrite( simimg, tempfile(fileext=.mha))
  }
  imageList = list.files(tdir, pattern = ".mha", full.names = TRUE)
  mask = getMask( antsImageRead( imageList[1] ) )
  mat = imagesToMatrix(imageList, mask)
  print(dim(mat))
```

iMath *iMath*

Description

Perform various (often mathematical) operations on the input image. Additional parameters should be specific for each operation. See the the full ImageMath in ANTs, on which this function is based.

Usage

```
iMath(img, operation, param = NA, ...)
```

Arguments

img	input object, usually antsImage
operation	a character string e.g. "GetLargestComponent" ... the special case of "GetOperations" or "GetOperationsFull" will return a list of operations and brief description. Some operations may not be valid (WIP), but most are.
param	... additional parameters
...	further parameter options

Author(s)

BB Avants

Examples

```
fi<-antsImageRead( getANTsRData("r16") , 2 )
mask<-getMask( fi )
op1<-iMath( fi , "GD" , 1 )  # gray matter dilation by 1 voxel
op2<-iMath( mask , "Neg" )  # negate
op3<-iMath( mask , "D" )  # distance transform
ops<-iMath( mask , "GetOperations" )  # list all ops

if ( usePkg("magrittr") ) { # string ops together
  lapgd <- fi %>% iMath("Laplacian",1)  %>% iMath("GD",3)
}
```

iMathOps *iMathOps*

Description

Examples, description and categorization of iMath operations.

Usage

```
data(iMathOps)
```

Format

A data frame listing the following variables.

`Operation` Name of operation

`OperationType` Enumerated type of the operation (filter, etc)

`Parameters` descriptive parameters

`Example` working example code

`Description` free text description

`OutputDimensionalityChange` NA if not image, otherwise increment to output image dimensionality relative to input image

Examples

```
i<-antsImageRead( getANTsRData(r16) , 2 )
roiImg<-getMask(i)
roiImg2<-iMath(roiImg,ME,25)
if ( sum(roiImg==1) == sum(roiImg2==1)  ) stop("erosion failure")
roiImg2<-iMath(roiImg,MD,25)
if ( sum(roiImg==1) == sum(roiImg2==1)  ) stop("dilation failure")
data( "iMathOps", package = "ANTsR", envir = environment() ) #
for ( j in c(1:nrow(iMathOps)) )
  {
  op <- as.character( iMathOps$Operation[j] )
  #  k  <- eval( parse( text = toString( iMathOps$Example[j] ) ) )
  }
```

---

initializeEigenanatomy

Convert a matrix to a form that can be used to initialize sparse cca and pca.

Description

InitializeEigenanatomy is a helper function to initialize sparseDecom and sparseDecom2.

Usage

```
initializeEigenanatomy(initmat, mask = NA, nreps = 1)
```

Arguments

`initmat`	input matrix where rows provide initial vector values
`mask`	mask if available
`nreps`	nrepetitions to use

Value

list is output

Author(s)

Avants BB

Examples

```
mat<-t(replicate(3, rnorm(100)) )
for ( i in 1:nrow(mat) ) mat[i, abs(mat[i,]) < 1 ]<-0
initdf<-initializeEigenanatomy( mat )
dmat<-replicate(100, rnorm(20))
eanat<-sparseDecom( dmat, inmask=initdf$mask,
  sparseness=0, smooth=0,
  initializationList=initdf$initlist, cthresh=0,
  nvecs=length(initdf$initlist) )
initdf2<-initializeEigenanatomy( mat, nreps=2 )
eanat<-sparseDecom( dmat, inmask=initdf$mask,
  sparseness=0, smooth=0, z=-0.5,
  initializationList=initdf2$initlist, cthresh=0,
  nvecs=length(initdf2$initlist) )
# now a regression
eanatMatrix<-imageListToMatrix(  eanat$eigenanatomyimages, initdf$mask )
# averages loosely speaking anyway
myEigenanatomyRegionAverages<-dmat %*% t( eanatMatrix )
dependentvariable<-rnorm( nrow(dmat) )
summary(lm( dependentvariable ~ myEigenanatomyRegionAverages ))

nvox<-1000
dmat<-replicate(nvox, rnorm(20))
dmat2<-replicate(30, rnorm(20))
mat<-t(replicate(3, rnorm(nvox)) )
initdf<-initializeEigenanatomy( mat )
eanat<-sparseDecom2( list(dmat,dmat2), inmask=c(initdf$mask,NA),
  sparseness=c( -0.1, -0.2 ), smooth=0,
  initializationList=initdf$initlist, cthresh=c(0,0),
  nvecs=length(initdf$initlist), priorWeight = 0.1 )
```

---

interleaveMatrixWithItself

Simple interleaveMatrixWithItself function.

Description

Replicate columns of input matrix n times as neighbors of columns in original matrix.

Usage

```
interleaveMatrixWithItself(x, n = 1)
```

Arguments

x	input matrix
n	number of interleaves

Value

matrix is output

Author(s)

Avants BB

Examples

```
mat<-replicate(100, rnorm(20))
wmat<-interleaveMatrixWithItself( mat , 5 )
```

`invariantImageSimilarity` *similarity metrics between two images as a function of geometry*

Description

compute similarity metric between two images as image is rotated about its center w/or w/o optimization

Usage

```
invariantImageSimilarity(in_image1, in_image2, thetas,
  localSearchIterations = 0, metric = "MI", scaleImage = 1,
  doReflection = 0, txfn = "")
```

Arguments

`in_image1`	reference image
`in_image2`	moving image
`thetas`	numeric vector of search angles
`localSearchIterations`	integer controlling local search in multistart
`metric`	which metric MI or GC (string)
`scaleImage`	global scale
`doReflection`	reflect image about principal axis
`txfn`	if present, write optimal tx to .mat file

Value

vector of similarity values

Author(s)

Brian B. Avants

Examples

```
fi<-antsImageRead( getANTsRData("r16"))
mi<-antsImageRead( getANTsRData("r64"))
mival<-invariantImageSimilarity( fi, mi, c(0,10,20) )
```

is.antsImage *is.antsImage*

Description

Tests if object is antsImage

Usage

```
is.antsImage(x)
```

Arguments

x	An object

Value

TRUE if object is antsImage; FALSE otherwise.

Examples

```
is.antsImage(antsImageRead(getANTsRData(r16), 2))
```

joinEigenanatomy *Simple joinEigenanatomy function.*

Description

joinEigenanatomy joins the input matrix using a community membership approach.

Usage

```
joinEigenanatomy(datamatrix, mask = NA, listEanatImages,
  graphdensity = 0.65, joinMethod = NA, verbose = F)
```

Arguments

datamatrix	input matrix before decomposition
mask	mask used to create datamatrix
listEanatImages	list containing pointers to eanat images
graphdensity	target graph density or densities to search over
joinMethod	see igraph's community detection
verbose	bool

Value

return(list(fusedlist = newelist, fusedproj = myproj, memberships = communitymembership , graph=gg, bestdensity=graphdensity))

Author(s)

Avants BB

Examples

```
# if you dont have images
mat<-replicate(100, rnorm(20))
mydecom<-sparseDecom( mat )
kk<-joinEigenanatomy( mat, mask=NA, mydecom$eigenanatomyimages , 0.1 )
# or select optimal parameter from a list
kk<-joinEigenanatomy( mat, mask=NA, mydecom$eigenanatomyimages , c(1:10)/50 )
# something similar may be done with images
mask<-as.antsImage( t(as.matrix(array(rep(1,ncol(mat)),ncol(mat)))) )
mydecom<-sparseDecom( mat, inmask=mask )
kki<-joinEigenanatomy( mat, mask=mask, mydecom$eigenanatomyimages , 0.1 )
if ( usePkg("igraph") ) {
  mydecomf<-sparseDecom( mat, inmask=mask, initializationList=kki$fusedlist ,
    sparseness=0, nvecs=length(kki$fusedlist) )
 }
```

---

jointIntensityFusion *joint intensity fusion*

Description

Estimates an image from another set of images - intensity generalization of joint label fusion. Search radius used only when employing labels - WIP to speed it up.

Usage

```
jointIntensityFusion(targetI, targetIMask, atlasList, beta = 4, rad = NA,
  labelList = NA, doscale = TRUE, doNormalize = TRUE,
  maxAtlasAtVoxel = c(1, Inf), rho = 0.01, useSaferComputation = FALSE,
  usecor = FALSE, boundary.condition = "mean", rSearch = 2,
  segvals = NA, includezero = FALSE, computeProbs = FALSE)
```

Arguments

targetI	antsImage to be approximated
targetIMask	mask with value 1
atlasList	list containing antsImages
beta	weight sharpness, default to 2
rad	neighborhood radius, default to 4
labelList	list containing antsImages
doscale	scale neighborhood intensities
doNormalize	normalize each image range to 0, 1
maxAtlasAtVoxel	min/max n atlases to use at each voxel
rho	ridge penalty increases robustness to outliers but also makes image converge to average

useSaferComputation	slower but more error checking
usecor	employ correlation as local similarity
boundary.condition	one of 'image' 'mean' 'NA'
rSearch	radius of search, default is 2
segvals	list of labels to expect
includezero	boolean - try to predict the zero label
computeProbs	boolean - requires more memory

Value

approximated image, segmentation and probabilities

Author(s)

Brian B. Avants, Hongzhi Wang, Paul Yushkevich

Examples

```
set.seed(123)
ref<-antsImageRead( getANTsRData("r16"))
ref<-resampleImage(ref,c(50,50),1,0)
ref<-iMath(ref,"Normalize")
mi<-antsImageRead( getANTsRData("r27"))
mi2<-antsImageRead( getANTsRData("r30"))
mi3<-antsImageRead( getANTsRData("r62"))
mi4<-antsImageRead( getANTsRData("r64"))
mi5<-antsImageRead( getANTsRData("r85"))
refmask<-getMask(ref)
refmask<-iMath(refmask,"ME",2) # just to speed things up
ilist<-list(mi,mi2,mi3,mi4,mi5)
seglist<-list()
for ( i in 1:length(ilist) )
 {
 ilist[[i]]<-iMath(ilist[[i]],"Normalize")
 mytx<-antsRegistration(fixed=ref , moving=ilist[[i]] ,
   typeofTransform = c("Affine") )
 mywarpedimage<-antsApplyTransforms(fixed=ref,moving=ilist[[i]],
   transformlist=mytx$fwdtransforms)
 ilist[[i]]=mywarpedimage
 seg<-kmeansSegmentation( ilist[[i]], k=3, kmask = refmask)
 seglist[[i]]<-seg$segmentation
 }
r<-2
d<-2
pp<-jointIntensityFusion(ref,refmask,ilist, rSearch=0,
  labelList=seglist, rad=rep(r,d) )
```

jointIntensityFusion3D *jointIntensityFusion3D*

Description

Estimates an image/labelset from another set of 3D images

Usage

```
jointIntensityFusion3D(targetI, targetIMask, atlasList, beta = 4, rad = NA,
  labelList = NA, doscale = TRUE, doNormalize = TRUE,
  maxAtlasAtVoxel = c(1, Inf), rho = 0.01, useSaferComputation = FALSE,
  usecor = FALSE, rSearch = 0, slices = NA, includezero = FALSE,
  computeProbs = FALSE)
```

Arguments

targetI	antsImage to be approximated
targetIMask	mask with value 1
atlasList	list containing antsImages
beta	weight sharpness, default to 2
rad	neighborhood radius, default to 4
labelList	list containing antsImages
doscale	scale neighborhood intensities
doNormalize	normalize each image range to 0, 1
maxAtlasAtVoxel	min/max n atlases to use at each voxel
rho	ridge penalty increases robustness to outliers but also makes image converge to average
useSaferComputation	slower but more error checking
usecor	employ correlation as local similarity
rSearch	radius of search, default is 2
slices	vector defining slices to use (speeds parameter selection)
includezero	boolean - try to predict the zero label
computeProbs	boolean - requires more memory

Details

intensity generalization of joint label fusion, still supports segmentation. this version is more efficient, memory-wise, for 3D images. it is a thin wrapper that goes slice-by-slice but produces the same results.

Value

approximated image, segmentation and probabilities (latter are WIP, might be done by the time your read this) ...

Author(s)

Brian B. Avants, Hongzhi Wang, Paul Yushkevich

Examples

```
# see jointIntensityFusion for a detailed example
# defaults for this function are current recommended parameters
```

`kellyKapowski` *Compute cortical thickness using the DiReCT algorithm.*

Description

Diffeomorphic registration-based cortical thickness based on probabilistic segmentation of an image. This is an optimization algorithm.

Usage

```
kellyKapowski(s = NA, g = NA, w = NA, its = 50, r = 0.025, m = 1.5,
  ...)
```

Arguments

`s`	segmentation image
`g`	gray matter probability image
`w`	white matter probability image
`its`	convergence params - controls iterations
`r`	gradient descent update parameter
`m`	gradient field smoothing parameter
`...`	anything else, see KK help in ANTs

Value

thickness antsImage

Author(s)

Shrinidhi KL, Avants BB

Examples

```
img<-antsImageRead( getANTsRData("r16") ,2)
img<-resampleImage(img,c(64,64),1,0)
mask<-getMask( img )
segs<-kmeansSegmentation( img, k=3, kmask = mask)
thk<-kellyKapowski( s=segs$segmentation, g=segs$probabilityimages[[2]],
  w=segs$probabilityimages[[3]],its=45,r=0.5,m=1 )
```

kmeansSegmentation *k means image segmentation.*

Description

k means image segmentation that is a wrapper around atropos

Usage

```
kmeansSegmentation(img, k, kmask = NA, mrf = 0.1)
```

Arguments

img	input image
k	integer number of classes
kmask	segment inside this mask
mrf	smoothness, higher is smoother

Value

segmentation and probability images

Author(s)

Brian B. Avants

Examples

```
fi<-antsImageRead( getANTsRData("r16") ,2)
fi<-n3BiasFieldCorrection(fi,4)
seg<-kmeansSegmentation( fi, 3 )
```

labelClusters *Simple labelClustering function.*

Description

This will give a unique ID to each connected component 1 through N of size > minClusterSize

Usage

```
labelClusters(imagein, minClusterSize = 50, minThresh = 1e-06,
  maxThresh = 1)
```

Arguments

imagein	input antsImage e.g. a statistical map
minClusterSize	throw away clusters smaller than this value
minThresh	threshold to a statistical map
maxThresh	threshold to a statistical map

Value

labeled cluster image is output

Author(s)

Avants BB

Examples

```
## Not run:
img<-antsImageRead( getANTsRData("mnib"), 3 )
outimage<-labelClusters( img )

## End(Not run)
```

`labelGeometryMeasures` *labelGeometryMeasures*

Description

Wrapper for the ANTs funtion labelGeometryMeasures

Usage

```
labelGeometryMeasures(labelImage, intensityImage = NA)
```

Arguments

`labelImage`	on which to compute geometry
`intensityImage`	optional

Value

none

Author(s)

Avants BB, Tustison NJ

Examples

```
fi<-antsImageRead( getANTsRData("r16") , 2 )
seg<-kmeansSegmentation( fi, 3 )$segmentation
geom<-labelGeometryMeasures(seg,fi)
```

`labelImageCentroids` *labelImageCentroids*

Description

Converts a label image to coordinates summarizing their positions

Usage

```
labelImageCentroids(img, physical = FALSE, convex = TRUE)
```

Arguments

`img`	ants image of labels (ints)
`physical`	boolean if you want physical space coordinates or not
`convex`	- if TRUE, return centroid, if FALSE return point with min average distance to other points with same label

Value

list with labels , array of label values, and centroids <- coordinates of label centroids

Author(s)

Avants BB, Duda JT

Examples

```
labelImage<-makeImage( c(2,2,2) , 0:7 )
labelImageCentroids(labelImage)
```

`labelStats` *labelStats*

Description

Get label statistics from image.

Usage

```
labelStats(image, labelImage)
```

Arguments

`image`	Image to calculate statistics from.
`labelImage`	Label image.

Value

Data frame with columns : LabelValues, Mean, Min, Max, Variance, Count, Volume

Examples

```
img <- antsImageRead( getANTsRData("r16") , 2 )
img <- resampleImage( img, c(64,64), 1, 0 )
mask <- getMask(img)
segs1 <- atropos( d = 2, a = img, m = [0.2,1x1],
 c = [2,0],  i = kmeans[3], x = mask )
labelStats(img, segs1$segmentation)
```

---

lappend *Simple list append tool*

Description

Append an item to a list , creating a new list or combine two lists

Usage

```
lappend(lst, obj)
```

Arguments

lst	list to which you will append the 2nd item
obj	to be added to the list

Value

outputs the appended list

Author(s)

Avants BB, the internet

Examples

```
## Not run:
mat<-replicate(100, rnorm(20))
mat2<-replicate(100, rnorm(20))
mylist<-list(mat)
mylist<-lappend( mylist, mat2 )

## End(Not run)
```

lowrankRowMatrix *Produces a low rank version of the input matrix*

Description

Produces a low rank version of the input matrix

Usage

```
lowrankRowMatrix(A, k = 2)
```

Arguments

A	input matrix
k	rank to use

Value

matrix is output

Author(s)

Avants BB

Examples

```
mat <- matrix(rnorm(300),ncol=50)
lrmat <- lowrankRowMatrix( mat , 2 )
```

make3ViewPNG *Rotate an existing 3d window into different views.*

Description

The make3ViewPNG function rotates the existing viewport according to 3 different rotation matrices passed in by the user. The output of these 3 views is munged together along the left/right edge and written to a png file.

Usage

```
make3ViewPNG(rotationView1, rotationView2, rotationView3, fnprefix)
```

Arguments

rotationView1	Leftmost view
rotationView2	Center view
rotationView3	Rightmost view
fnprefix	Output file name prefix.

Value

NA

Author(s)

Avants BB

Examples

```
## Not run:
bm<-getMask( antsImageRead( getANTsRData("ch2") ) )
brain<-renderSurfaceFunction( surfimg =list( bm ) ,
    alphasurf=0.1 ,smoothsval = 1.5 )
make3ViewPNG(  diag(4), diag(4), diag(4),  tempfile( fileext=.png) )

## End(Not run)
```

---

makeGraph *Simple function to create and measure a graph from a square input matrix.*

Description

Creates an igraph object from a square input correlation matrix - only positive correlations are used. Based on the graph.adjacency function of igraph. gplot is helpful for visualization.

Usage

```
makeGraph(mat, graphdensity = 1, communityMethod = NA,
  getEfficiency = FALSE)
```

Arguments

mat	input matrix
graphdensity	fraction of edges to keep
communityMethod	see igraph's community detection
getEfficiency	boolean, this is slow to compute

Value

a named list is output including the graph object, adjacency matrix and several graph metrics

Author(s)

Avants BB

Examples

```
## Not run:
mat <- matrix(c(rep(1,100)),ncol=10)
gobj<-makeGraph( mat )
mat <- matrix(  c( 1, 0.5, 0.2, -0.1, 1, 0.3, -0.2, 0.6, 1 ) , ncol= 3 )
gobj<-makeGraph( mat , 0.5 )
# gplot( gobj$adjacencyMatrix ) # need sna library for this

## End(Not run)
```

---

makeImage *Simple makeImage function.*

Description

Make an image with given size and voxel value or given a mask and vector.

Usage

```
makeImage(imagesize, voxval = 1)
```

Arguments

imagesize	input image size or mask
voxval	input image value or vector, size of mask

Value

antsImage is output

Author(s)

Avants BB

Examples

```
outimg<-makeImage( c(2,10) , 1)
outimg<-makeImage( outimg ,  c(2,10) )
```

maskImage *Mask input image by mask image.*

Description

Mask an input image by a mask image. If the mask image has multiple labels, it is possible to specify which label(s) to mask at.

Usage

```
maskImage(img.in, img.mask, level = 1, binarize = FALSE)
```

Arguments

img.in	Input image.
img.mask	Mask or label image.
level	Level(s) at which to mask image. If vector or list of values, output image is non-zero at all locations where label image matches any of the levels specified.
binarize	binarize the output image?

Value

An object of type antsImage.

Author(s)

Kandel BM and Avants B.

Examples

```
myimg <- antsImageRead(getANTsRData("r16"))
mask <- getMask(myimg)
myimg.mask <- maskImage(myimg, mask, 3)
seg <- kmeansSegmentation(myimg, 3)
myimg.mask <- maskImage(myimg, seg$segmentation, c(1,3))
```

matrix2timeseries *Simple matrix2timeseries function.*

Description

matrix2timeseriess converts a matrix to a 4D image.

Usage

```
matrix2timeseries(referenceImage, maskImage, timeSeriesMatrix)
```

Arguments

`referenceImage`	reference 4D image
`maskImage`	mask image defining voxels of interest
`timeSeriesMatrix`	matrix to convert to image

Value

antsImage in 4D is output

Author(s)

Avants BB

`matrixToImages` *Convert rows of a matrix to images*

Description

Unmasks rows of a matrix and writes as images.

Usage

```
matrixToImages(dataMatrix, mask)
```

Arguments

`dataMatrix`	A matrix such as that created by `imagesToMatrix`.
`mask`	An `antsImage` containing a binary mask. Rows of the matrix are unmasked and written as images. The mask defines the output image space.

Author(s)

Cook PA

See Also

`imagesToMatrix`, `getMask`

Examples

```
## Not run:
  # mat = matrixToImages( aMat, mask )

## End(Not run)
```

`mean,antsImage-method` *arith.antsImage*

Description

Atomic arithmetic operators for antsImages

Usage

```
## S4 method for signature antsImage
mean(x, mask = logical())

## S3 method for class antsImage
x + y

## S3 method for class antsImage
x - y

## S3 method for class antsImage
x / y

## S3 method for class antsImage
x * y

## S3 method for class antsImage
x ^ y

## S3 method for class antsImage
x %% y

## S3 method for class antsImage
log(x, ...)

## S3 method for class antsImage
exp(x)
```

Arguments

x	antsImage
mask	antsImage logical mask (optional)
y	antsImage or numeric
...	Additional arguments passed to underlying R operator

Examples

```
r16 <- antsImageRead(getANTsRData(r16), 2)
r64 <- antsImageRead(getANTsRData(r64), 2)
r16 + r64
r16 + 5
r16 / 10
log(r16, base=10)
```

mergeChannels *merge images into a multiChannel antsImage*

Description

merge images into a multiChannel antsImage

Usage

```
mergeChannels(imageList)
```

Arguments

imageList	a list of antsImage objects to merge

Value

A multiChannel antsImage object

Author(s)

Duda, JT

Examples

```
r <- floor( seq(1:(64*64)) / (64*64) * 255 )
 dim(r) <- c(64,64)
 r <- as.antsImage(r)
 g <- r*0
 b <- r*0
 rgbImage = mergeChannels( list(r,g,b) )
```

mni2tal *Brett's mni2tal*

Description

mni2tal for converting from ch2/mni space to tal - very approximate.

Usage

```
mni2tal(xin = 0)
```

Arguments

xin	point in mni152 space.

Details

This is a standard approach but it's not very accurate.

Value

output point in approximate Talairach space.

Author(s)

Matthew Brett, adapted for ANTsR by B Avants

References

http://bioimagesuite.yale.edu/mni2tal/501_95733_More%20Accurate%20Talairach%20Coordinates%20SLIDES.pdf, http://imaging.mrc-cbu.cam.ac.uk/imaging/MniTalairach

Examples

```
mni2tal( c(10,12,14) )
```

`mrvnrfs` *multi-res voxelwise neighborhood random forest segmentation learning*

Description

Represents multiscale feature images as a neighborhood and uses the features to build a random forest segmentation model from an image population

Usage

```
mrvnrfs(y, x, labelmask, rad = NA, nsamples = 1, ntrees = 500,
  multiResSchedule = c(4, 2, 1), asFactors = TRUE)
```

Arguments

`y`	list of training labels. either an image or numeric value
`x`	a list of lists where each list contains feature images
`labelmask`	a mask for the features (all in the same image space) the labelmask defines the number of parallel samples that will be used per subject sample. two labels will double the number of predictors contributed from each feature image.
`rad`	vector of dimensionality d define nhood radius
`nsamples`	(per subject to enter training)
`ntrees`	(for the random forest model)
`multiResSchedule`	an integer vector defining multi-res levels
`asFactors`	boolean - treat the y entries as factors

Value

list a 4-list with the rf model, training vector, feature matrix and the random mask

Author(s)

Avants BB, Tustison NJ

Examples

```
mask<-makeImage( c(10,10), 0 )
mask[ 3:6, 3:6 ]<-1
mask[ 5, 5:6]<-2
ilist<-list()
lablist<-list()
inds<-1:50
scl<-0.33 # a noise parameter
for ( predtype in c("label","scalar") )
{
for ( i in inds ) {
  img<-antsImageClone(mask)
  imgb<-antsImageClone(mask)
  limg<-antsImageClone(mask)
  if ( predtype == "label") {  # 4 class prediction
    img[ 3:6, 3:6 ]<-rnorm(16)*scl+(i %% 4)+scl*mean(rnorm(1))
    imgb[ 3:6, 3:6 ]<-rnorm(16)*scl+(i %% 4)+scl*mean(rnorm(1))
    limg[ 3:6, 3:6 ]<-(i %% 4)+1  # the label image is constant
    }
    if ( predtype == "scalar") {
      img[ 3:6, 3:6 ]<-rnorm(16,1)*scl*(i)+scl*mean(rnorm(1))
      imgb[ 3:6, 3:6 ]<-rnorm(16,1)*scl*(i)+scl*mean(rnorm(1))
      limg<-i^2.0  # a real outcome
      }
    ilist[[i]]<-list(img,imgb)  # two features
    lablist[[i]]<-limg
  }
rad<-rep( 1, 2 )
mr <- c(1.5,1)
rfm<-mrvnrfs( lablist , ilist, mask, rad=rad, multiResSchedule=mr,
     asFactors = (  predtype == "label" ) )
rfmresult<-mrvnrfs.predict( rfm$rflist,
     ilist, mask, rad=rad, asFactors=(  predtype == "label" ),
     multiResSchedule=mr )
if ( predtype == "scalar" )
  print( cor( unlist(lablist) , rfmresult$seg ) )
} # end predtype loop
```

`mrvnrfs.predict` *multi-res voxelwise neighborhood random forest segmentation*

Description

Represents multiscale feature images as a neighborhood and uses the features to apply a random forest segmentation model to a new image

Usage

```
mrvnrfs.predict(rflist, x, labelmask, rad = NA, multiResSchedule = c(4, 2,
  1), asFactors = TRUE)
```

Arguments

rflist	a list of random forest models from mrvnrfs
x	a list of lists where each list contains feature images
labelmask	a mask for the features (all in the same image space)
rad	vector of dimensionality d define nhood radius
multiResSchedule	an integer vector defining multi-res levels
asFactors	boolean - treat the y entries as factors

Value

list a 4-list with the rf model, training vector, feature matrix and the random mask

Author(s)

Avants BB, Tustison NJ

---

n3BiasFieldCorrection *Bias Field Correction*

Description

Perform Bias Field Correction on the given image

Usage

```
n3BiasFieldCorrection(img, downsampleFactor)
```

Arguments

img	antsImage to correct
downsampleFactor	integer e.g. 4 downsample by a factor of 4

Value

antsImage

Author(s)

Shrinidhi KL

Examples

```
img<-makeImage(c(10,10),rnorm(100))
n3img<-n3BiasFieldCorrection(  img, 1 )
```

n4BiasFieldCorrection *Bias Field Correction*

Description

Perform Bias Field Correction on the given image

Usage

```
n4BiasFieldCorrection(img, mask = NA, shrinkFactor = 4,
  convergence = list(iters = c(50, 50, 50, 50), tol = 1e-07),
  splineParam = 200)
```

Arguments

img	input antsImage
mask	input mask, if one is not passed one will be made
shrinkFactor	Shrink factor for multi-resolution correction, typically integer less than 4
convergence	List of: iters, vector of maximum number of iterations for each shrinkage factor, and tol, the convergence tolerance.
splineParam	Parameter controlling number of control points in spline. Either single value, indicating how many control points, or vector with one entry per dimension of image, indicating the spacing in each direction.

Value

outimg Bias-corrected image

Author(s)

BB Avants

Examples

```
img<-makeImage( c(50,50), rnorm(2500) )
 n4img<-n4BiasFieldCorrection(img)
```

networkEiganat *Convenience wrapper for eigenanatomy decomposition.*

Description

Decomposes a matrix into sparse eigenevectors to maximize explained variance.

Usage

```
networkEiganat(Xin, sparseness = c(0.1, 0.1), nvecs = 5, its = 5,
  gradparam = 1, mask = NA, v, prior, pgradparam = 0.1, clustval = 0,
  downsample = 0, doscale = T, domin = T, verbose = F, dowhite = 0,
  timeme = T, addb = T, useregression = T)
```

Arguments

Xin	n by p input images , subjects or time points by row , spatial variable lies along columns
sparseness	sparseness pair c(0.1 , 0.1)
nvecs	number of vectors
its	number of iterations
gradparam	gradient descent parameter for data
mask	optional antsImage mask
v	the spatial solultion
prior	the prior
pgradparam	gradient descent parameter for prior term
clustval	integer greater than or equal to zero
downsample	bool
doscale	bool
domin	bool
verbose	bool
dowhite	bool
timeme	bool
addb	bool
useregression	bool

Value

outputs a decomposition of a population or time series matrix

Author(s)

Avants BB

Examples

```
## Not run:
mat<-replicate(100, rnorm(20))
mydecom<-networkEiganat( mat, nvecs=5 )
ch1<-usePkg(randomForest)
ch2<-usePkg(BGLR)
if ( ch1 & ch2 ) {
data(mice)
snps<-quantifySNPs( mice.X )
numericalpheno<-as.matrix( mice.pheno[,c(4,5,13,15) ] )
numericalpheno<-residuals( lm( numericalpheno ~
   as.factor(mice.pheno$Litter) ) )
phind<-3
nfolds<-6
train<-sample( rep( c(1:nfolds), 1800/nfolds ) )
train<-( train < 4 )
lowr<-lowrankRowMatrix(as.matrix( snps[train,] ),900)
snpdS<-sparseDecom( lowr , nvecs=2 , sparseness=( -0.001), its=3  )
snpdF<-sparseDecom( lowrankRowMatrix(as.matrix( snps[train,] ),100),
```

```
  nvecs=2 , sparseness=( -0.001), its=3 )
projmat<-as.matrix( snpdS$eig )
projmat<-as.matrix( snpdF$eig )
snpdFast<-networkEiganat( as.matrix( snps[train,] ), nvecs=2 ,
  sparseness=c( 1, -0.001 ) , downsample=45, verbose=T, its=3,
  gradparam=10 )
snpdSlow<-networkEiganat( as.matrix( snps[train,] ), nvecs=2 ,
  sparseness=c( 1, -0.001 ) , downsample=0, verbose=T,
  its=3, gradparam=10 )
snpd<-snpdSlow
snpd<-snpdFast
projmat<-as.matrix( snpd$v )
snpdF<-sparseDecom( lowrankRowMatrix(as.matrix( snps[train,] ),10) ,
  nvecs=2 , sparseness=( -0.001), its=3  )
projmat<-as.matrix( snpdS$eig )
snpse<-as.matrix( snps[train, ]  ) %*% projmat
traindf<-data.frame( bmi=numericalpheno[train,phind] , snpse=snpse)
snpse<-as.matrix( snps[!train, ]  ) %*% projmat
testdf <-data.frame( bmi=numericalpheno[!train,phind] , snpse=snpse )
myrf<-glm( bmi ~ . , data=traindf )
preddf<-predict(myrf, newdata=testdf )
cor.test(preddf, testdf$bmi )
if ( usePkg(visreg) ) {
mydf<-data.frame( PredictedBMIfromSNPs=preddf, RealBMI=testdf$bmi )
mymdl<-lm( PredictedBMIfromSNPs ~ RealBMI, data=mydf)
visreg::visreg(mymdl) }
############
# vs glmnet #
############
haveglm<-usePkg(glmnet)
if ( haveglm ) {
kk<-glmnet(y=numericalpheno[train,phind],x=snps[train,] )
ff<-predict(kk,newx=snps[!train,])
cor.test(ff[,25],numericalpheno[!train,phind])
mydf<-data.frame( PredictedBMIfromSNPs=ff[,25], RealBMI=testdf$bmi )
mymdl<-lm( PredictedBMIfromSNPs ~ RealBMI, data=mydf)
} # glmnet check
} # ch1 and ch2
############

## End(Not run)
```

`pairwiseImageDistanceMatrix` *Simple pairwiseImageDistanceMatrix function for images*

Description

Output contains the NImages x NImages matrix of c('PearsonCorrelation','Mattes') or any Image Metric values available in iMath. Similarity is computed after an affine registration is performed. You can also cluster the images via the dissimilarity measurement, i.e. the negated similarity metric. So, the estimated dissimilarity is returned in the matrix.

Usage

```
pairwiseImageDistanceMatrix(dim, myFileList,
  metrictype = "PearsonCorrelation", nclusters = NA)
```

Arguments

dim	imageDimension
myFileList	dd<-'MICCAI-2013-SATA-Challenge-Data/CAP/training-images/' myFileList<-list.files(path=dd, pattern = glob2rx('*nii.gz'),full.names = T,recursive = T)
metrictype	similarity function
nclusters	integer controlling max number of clusters to search over

Value

raw dissimilarity matrix is output, symmetrized matrix and clustering (optional) in a list

Author(s)

Avants BB

Examples

```
## Not run:
  # dsimdata<-pairwiseImageDistanceMatrix( 3, imagefilelist, nclusters = 5 )

## End(Not run)
```

---

partialVolumeCorrection

Perform partial volume correction for ASL images.

Description

This function performs a partial volume correction for ASL images by dividing the observed CBF by the gray matter and white matter probabilities, as described in Johnson et al., Radiology 2005: CBF_corrected = CBF_observed / (GM_prob + 0.4 * WM_prob)

Usage

```
partialVolumeCorrection(img, img.gm, img.wm, mask = NULL, proportion = 0.4)
```

Arguments

img	Low-resolution image to be corrected. All input images can be either of type antsImage or numeric vectors (if numeric vectors, the mask is ignored).
img.gm	Gray matter probability image for partial volume correction.
img.wm	White matter probability image for partial volume correction.
mask	Brain mask for image.
proportion	Ratio of activity for white matter to gray matter. Assumed to be 0.4.

Value

Returns partial volume corrected antsImage.

Author(s)

Kandel BM

References

Method: Pattern of cerebral hypoperfusion in Alzheimer disease and mild cognitive impairment measured with arterial spin-labeling MR imaging: initial experience. Johnson NA, Jahng GH, Weiner MW, Miller BL, Chui HC, Jagust WJ, Gorno-Tempini ML, Schuff N. Radiology 2005.

Ratio of GM to WM activity: Quantitative magnetic resonance imaging of human brain perfusion at 1.5 T using steady-state inversion of arterial water. Roberts DA, Detre JA, Bolinger L, Insko EK, Leigh JS Jr. PNAS 1994.

Examples

```
activity.gm <- 10
activity.wm <- activity.gm * 0.4
percent.gm <- matrix(seq(0.1, 1, by=0.1), nrow=2)
percent.wm <- -percent.gm + 1
activity.obs <- percent.gm * rnorm(n=length(percent.gm), mean=activity.gm, sd=5) +
                  rnorm(n=length(percent.wm), mean=activity.wm, sd=5)
activity.corrected <- partialVolumeCorrection(activity.obs, percent.gm, percent.wm)
```

---

perfusionregression *Perfusion Regression*

Description

Estimate CBF using standard regression and optionally robust regression.

Usage

```
perfusionregression(mask_img, mat, xideal, nuis = NA, dorobust = 0,
  skip = 20, selectionValsForRegweights = NULL, useBayesian = 0)
```

Arguments

mask_img	Mask image selects the voxels where CBF will be estimated. Voxels corresponding to logical FALSE are not computed.
mat	Matrix with a column for every time-series voxel. Number of rows equals the number of time units in the series.
xideal	1D time-series signal to be used a ideal or model for regression.
nuis	Nuisance parameters obtained from '.get_perfusion_predictors'.
dorobust	Real value in interval from 0 to 1. If greater than 0, then robust regression will be performed. A typical value would be 0.95 i.e. use voxels with 95 percent confidence.
skip	skip / stride over this number of voxels to increase speed

selectionValsForRegweights
: scalar function to guide parameter est.

useBayesian
: if greater than zero, use a bayesian prior w/this weight

Value

Success – An object of type 'antsImage' containing the CBF estimate for voxels corresponding to the mask input

Author(s)

Shrinidhi KL Avants BB

Examples

```
## Not run:
#
# cbf <- perfusionregression( mask_img, mat, xideal , nuis )

## End(Not run)
```

---

plot.antsImage *Plotting an image slice or multi-slice with optional color overlay.*

Description

This is a plotting utility for antsImage types with a background and color overlay option. Useful for displaying statistical results overlaid on a background image.

Usage

```
## S3 method for class antsImage
plot(x, y, color.img = "white", color.overlay = c("jet",
  "red", "blue", "green", "yellow"), axis = 2, slices,
  colorbar = missing(y), title.colorbar, title.img, color.colorbar,
  window.img, window.overlay, quality = 4, outname = NA, alpha = 0.5,
  newwindow = FALSE, nslices = 10, ...)
```

Arguments

x
: the reference image on which to overlay.

y
: image or list of images to use as overlays.

color.img
: color for main image.

color.overlay
: the color for the overlay , e.g c('blue','red') length of this list should match the image list.

axis
: the axis to slice (1 , 2 or 3)

slices
: vector of slices to plot (e.g., c(10, 15, 20))

colorbar
: make colorbar?

title.colorbar
: title for colorbar

title.img	title for main image
color.colorbar	color scale to use for colorbar
window.img	lower and upper thresholds for display of main image
window.overlay	lower and upper thresholds for display of overlay
quality	integer quality magnification factor 1 => large (e.g. 10)
outname	name of output file if you want to write result to file, e.g. plot.jpg.
alpha	opacity
newwindow	boolean controlling if we open a new device for this plot
nslices	number of slices to view
...	other parameters

Value

output is plot to standard R window

Author(s)

Avants BB

Examples

```
img <- makeImage(c(4,4), rnorm(4*4))
mask <- makeImage(c(4,4),
   as.matrix(c(0,0,0,0,
            0,1,1,0,
            0,1,1,0,
            0,0,0,0), nrow=4))
plot(img, list(mask))
## Not run:
  mnit<-getANTsRData(mni)
  mnit<-antsImageRead(mnit)
  mniafn<-getANTsRData(mnia)
  mnia<-antsImageRead(mniafn)
  mnia<-thresholdImage(mnia,22,25)
  mnia<-smoothImage(mnia,1.5)
  mnia2<-antsImageRead(mniafn)
  mnia2<-thresholdImage(mnia2,1,4)
  mnia2<-smoothImage(mnia2,1.5)
  ofn<-paste(tempfile(),.png,sep=)
  # write directly to a file
  plot( mnit, list(mnia,mnia2), slices=seq(50, 140, by=5),
    window.overlay = c(0.25,1), axis=2,
    overlay.color=c(red,blue), outname = ofn )

## End(Not run)
```

plotBasicNetwork *Simple plotBasicNetwork function.*

Description

takes an object output from renderSurfaceFunction and a list of centroids and plots the centroid network over the rendering object

Usage

```
plotBasicNetwork(centroids, brain, weights = NA, edgecolors = 0,
  nodecolors = "blue", nodetype = "s", scaling = c(0, 0), lwd = 2,
  radius = 3, showOnlyConnectedNodes = TRUE)
```

Arguments

centroids	input matrix of size number of 3D points (in rows) by 3 (in columns)
brain	input rendering object which is output of renderSurfaceFunction or a function derived from renderSurfaceFunction
weights	edge weights
edgecolors	a color(map) for edges
nodecolors	a color(map) for nodes
nodetype	sphere or other node type
scaling	controls functional range
lwd	line width
radius	for nodes
showOnlyConnectedNodes	boolean

Value

None

Author(s)

Avants BB and Duda JT

Examples

```
## Not run:
# more complete example
  mnit<-getANTsRData("mni")
  mnit<-antsImageRead(mnit)
  mnia<-getANTsRData("mnia")
  mnia<-antsImageRead(mnia)
  mnit<-thresholdImage( mnit, 1, max(mnit) )
  mnit<-iMath(mnit,"FillHoles")
  cnt<-getCentroids( mnia, clustparam = 50 )
  aalcnt<-cnt[1:90,]
  brain<-renderSurfaceFunction( surfimg =list( mnit ) , alphasurf=0.1 ,smoothsval = 1.5 )
```

```
  testweights<-matrix( rep( 0, 90*90 ) ,nrow=90)
  testweights[31,37]<-1  # ant cingulate to hipp
  testweights[31,36]<-2  # ant cingulate to post cingulate
  testweights[11,65]<-3  # broca to angular
  plotBasicNetwork( centroids = aalcnt , brain , weights=testweights )
  id<-rgl::par3d(userMatrix)
  rid<-rotate3d( id , -pi/2, 1, 0, 0 )
  rid2<-rotate3d( id , pi/2, 0, 0, 1 )
  rid3<-rotate3d( id , -pi/2, 0, 0, 1 )
  rgl::par3d(userMatrix = id )
  dd<-make3ViewPNG(  rid, id, rid2,  paste(network1,sep=) )
  rgl::par3d(userMatrix = id )

## End(Not run)
```

`plotPrettyGraph` *Simple plotPrettyGraph function saves to png.*

Description

PlotPrettyGraph given inputs from the makeGraph function. adapted from http://is-r.tumblr.com/.

Usage

```
plotPrettyGraph(adjacencyMatrix, functionToPlot, pngfn = "graph.png",
  scaleText = 0.5, vertexSize = NA, figScale = 11, layoutmode = "eigen",
  hueval = 0)
```

Arguments

`adjacencyMatrix`	igraph adjacencyMatrix
`functionToPlot`	igraph node-level graph value e.g. degree, page.rank, etc
`pngfn`	filename for output png or to screen if NA
`scaleText`	relative size of text to vertices
`vertexSize`	cex size of vertices
`figScale`	the figure will be of square size 2^figScale in pixels
`layoutmode`	see gplot.layout in sna package
`hueval`	controls the hue in hsv

Value

no output

Author(s)

Avants BB, Christopher DeSante and David Sparks

Examples

```
data("bold_correlation_matrix",package="ANTsR")
dmat<-data.matrix(bold_correlation_matrix)
if ( usePkg("igraph") ) {
 gg<-makeGraph( dmat, 0.1 )
 rownames(gg$adjacencyMatrix)<-colnames(bold_correlation_matrix)
 plotPrettyGraph( gg$adjacencyMatrix, gg$degree , figScale=12 , scaleText=5 )
 }
```

---

preprocessfMRI *Preprocess BOLD fMRI image data.*

Description

Preprocess fMRI data by performing compcor/motion correction, nuisance regression, band-pass filtering, and spatial smoothing.

Usage

```
preprocessfMRI(boldImage, maskImage = NA, maskingMeanRatioThreshold = 0.75,
  initialNuisanceVariables = NA, numberOfCompCorComponents = 6,
  doMotionCorrection = TRUE, useMotionCorrectedImage = FALSE,
  motionCorrectionAccuracyLevel = 1,
  meanBoldFixedImageForMotionCorrection = NA, frequencyLowThreshold = NA,
  frequencyHighThreshold = NA, spatialSmoothingType = "none",
  spatialSmoothingParameters = 0, residualizeMatrix = TRUE)
```

Arguments

boldImage	4-D ANTs image fMRI data.
maskImage	3-D ANTs image defining the region of interest.
maskingMeanRatioThreshold	If mask image is not specified, a mask image is created using the specified threshold which is in terms of the mean of the average image ie 0.8 means threshold at 0.8 of the mean.
initialNuisanceVariables	Optional initial nuisance variables.
numberOfCompCorComponents	Numer of CompCor nuisance components.
doMotionCorrection	Boolean indicating whether motion correction should be performed and used in nuisance regression.
useMotionCorrectedImage	Boolean indicating whether or not the motion corrected image should be used in the rest of the pipeline. This is off by default to avoid additional interpolation.
motionCorrectionAccuracyLevel	Accuracy for the motion correcting registration: 0 = fast/debug parameters, 1 = intrasession parameters, or 2 = intersession/intersubject parameters.
meanBoldFixedImageForMotionCorrection	Optional target fixed image for motion correction.

`frequencyLowThreshold`
: Lower threshold for bandpass filtering.

`frequencyHighThreshold`
: Upper threshold for bandpass filtering.

`spatialSmoothingType`
: Either `none`, `gaussian` (isotropic) or `perona-malik` (anisotropic) smoothing.

`spatialSmoothingParameters`
: For gaussian smoothing, this is a single scalar designating the smoothing sigma (in mm). For perona-malik, a vector needs to be specified with the conductance parameter and the number of iterations, e.g. `c(0.25, 5)`.

`residualizeMatrix`
: boolean

Value

List of:

- cleanBOLDImage: Cleaned BOLD image.
- maskImage: mask image.
- DVARS: Framewise change in BOLD signal, as in Powers et al.
- DVARSPostCleaning: DVARS after cleaning image.
- FD: Framewise displacement.
- globalSignal: Global signal.
- nuisanceVariables: Nuisance variables used in denoising.

Author(s)

Tustison NJ, Avants BB

References

Power et al. 2012, "Spurious but systematic correlations in functional connectivity MRI networks arise from subject motion." NeuroImage 59, 2142-2154.

Examples

```
set.seed(123)
n=16
nvox <- n*n*n*12
dims <- c(n,n,n,12)
boldImage <- makeImage(dims, rnorm(nvox) + 500) %>% iMath("PadImage", 2)
# for real data: boldImage <- antsImageRead(getANTsRData(pcasl))
cleanfMRI <- preprocessfMRI(boldImage)
```

projectImageAlongAxis *Simple projectImageAlongAxis function.*

Description

Will turn an N-D image into an N-1-D image by summation (0), max-intensity (1) or min-intensity (2) projection along an orthogonal axis.

Usage

```
projectImageAlongAxis(imageND, referenceImageNDminus1, projtype = 0,
  axis = NA)
```

Arguments

imageND	input image
referenceImageNDminus1	down-dimensional image to help define physical #' space
projtype	projection type 0, 1 or 2
axis	should be less than image dimension and greater than or equal to 0

Value

image of n-dimensions-1 is output

Author(s)

Avants BB

Examples

```
img<-makeImage(c(5,5,5,5),rnorm(5^4))
img3d<-makeImage(c(5,5,5),rnorm(5^3))
mask4dproj<-projectImageAlongAxis( img, img3d, axis=3, projtype=0 )
```

quantifyCBF *quantifyCBF*

Description

Computes CBF from ASL - pasl or pcasl

Usage

```
quantifyCBF(perfusion, mask, parameters, M0val = NA, outlierValue = 0.02)
```

Arguments

perfusion	input asl matrix
mask	3D image mask (antsImage)
parameters	list with entries for sequence and m0 (at minimimum)
M0val	baseline M0 value (optional)
outlierValue	trim outliers by this fractional value (optional)

Value

a list is output with 3 types of cbf images

Author(s)

Avants BB, Kandel B, Duda JT

Examples

```
## Not run:
  if (!exists("fn") ) fn<-getANTsRData("pcasl")
  # PEDS029_20101110_pcasl_1.nii.gz # high motion subject
  asl<-antsImageRead(fn)
# image available at http://files.figshare.com/1701182/PEDS012_20131101.zip
  pcasl.bayesian <- aslPerfusion( asl,
        dorobust=0., useDenoiser=4, skip=11, useBayesian=1000,
        moreaccurate=0, verbose=T, maskThresh=0.5 ) # throw away lots of data
# user might compare to useDenoiser=FALSE
  pcasl.parameters <- list( sequence="pcasl", m0=pcasl.bayesian$m0 )
  cbf <- quantifyCBF( pcasl.bayesian$perfusion, pcasl.bayesian$mask,
     pcasl.parameters )
  meancbf <- cbf$kmeancbf
  print(mean(meancbf[ pcasl.bayesian$mask==1 ]))
  antsImageWrite( meancbf , "temp.nii.gz")
  pcasl.processing <- aslPerfusion( asl, moreaccurate=0,
    dorobust=0.95, useDenoiser=NA, skip=5,  useBayesian=0 )
  # user might compare to useDenoiser=FALSE
  pcasl.parameters <- list( sequence="pcasl", m0=pcasl.processing$m0 )
  cbf <- quantifyCBF( pcasl.processing$perfusion, pcasl.processing$mask, pcasl.parameters )
  meancbf <- cbf$kmeancbf
  print(mean(meancbf[ pcasl.processing$mask==1 ]))
  antsImageWrite( meancbf , "temp2.nii.gz" )
  plot(  meancbf, slices="1x50x1" )

## End(Not run)
```

---

quantifySNPs *Simple quantifySNPs function.*

Description

quantifySNPs converts trinary snps to frequency data

Usage

```
quantifySNPs(snps, freqthresh = 0.1, shiftit = FALSE, replaceWithF = T,
  traitvecin = NA, trainvec = NA)
```

Arguments

snps	input matrix
freqthresh	remove snps below this frequency
shiftit	shift the snps to smooth the estimate
replaceWithF	replaces snps with frequency values
traitvecin	map snps to trait vector
trainvec	defines training data

Value

matrix is output

Author(s)

Avants BB

Examples

```
mat <- matrix(c(0,1,2,0,0,1,2,2,2),ncol=3)
wmat<-quantifySNPs( mat , freqthresh=0)
```

---

rapidlyInspectImageData

Simple rapidlyInspectImageData function.

Description

rapidlyInspectImageData collects basic statistics over a dataset and returns them in a dataframe.

Usage

```
rapidlyInspectImageData(myfiles, dimension = 3)
```

Arguments

myfiles	input list of filenames
dimension	image dimensionality

Value

matrix is output

Author(s)

Avants BB

Examples

```
## Not run:
 fnl <- c( getANTsRData("r16"),
 getANTsRData("r27"),
 getANTsRData("r62"),
 getANTsRData("r64"),
 getANTsRData("r85") )
mm<-rapidlyInspectImageData( fnl )
if ( !usePkg("DMwR") | ! usePkg("fpc") )
   { print("Need DMwR and fpc packages") } else {
  pamres <- fpc::pamk(mm,1:4)
  outlier.scores <- DMwR::lofactor( mm, k=3 )
  outliers <- order(outlier.scores)
  }

## End(Not run)
```

`reflectImage` *reflectImage*

Description

reflects an image along its axis

Usage

```
reflectImage(img1, axis = NA, tx = NA)
```

Arguments

`img1`	input object, an antsImage
`axis`	which dimension to reflect across
`tx`	transformation type to estimate after reflection

Author(s)

BB Avants

Examples

```
fi<-antsImageRead( getANTsRData("r16") , 2 )
asym<-reflectImage( fi, 1, "Affine" )$warpedmovout
asym<-asym-fi
```

`regressionNetworkViz` *Visualize a regression result by a d3 network visualization.*

Description

Use either a force directed graph or a Sankey graph to show relationships between predictors and outcome variables. correlateMyOutcomes should correspond to the outcome variables ...

Usage

```
regressionNetworkViz(mylm, sigthresh = 0.05, whichviz = "Sankey",
  outfile = "temp.html", mygroup = 0, logvals = TRUE, verbose = FALSE,
  correlateMyOutcomes = NA, corthresh = 0.9, zoom = F, doFDR = TRUE)
```

Arguments

`mylm`	lm model output from bigLMStats
`sigthresh`	significance threshold
`whichviz`	which visualization method
`outfile`	significance threshold
`mygroup`	color each entry by group membership
`logvals`	bool
`verbose`	bool
`correlateMyOutcomes`	not sure, see code
`corthresh`	correlation threshold
`zoom`	zooming factor
`doFDR`	bool

Value

html file is output

Author(s)

Avants BB

Examples

```
## Not run:
# colnames(brainpreds)<-paste(Vox,c(1:ncol(brainpreds)),sep=)
# colnames( mylm$beta.pval )<-colnames(brainpreds)
# demognames<-rownames(mylm$beta.pval)
# regressionNetworkViz( mylm , sigthresh=0.05, outfile=temp2.html)

## End(Not run)
```

`regressProjections` *Regression on image set projection*

Description

Perform regression on a training set of images, projected onto (provided) eigenvectors, and test on testing images.

Usage

```
regressProjections(input.train, input.test, demog.train, demog.test,
  eigenvectors, mask, outcome, covariates = "1", model.function = glm,
  which.eigenvectors = "all", ...)
```

Arguments

`input.train`	Masked imaging data from training set of type `antsImage` or `matrix`. This will typically be read from a `train.mha` file generated by, e.g., `sccan --imageset-to-matrix`.
`input.test`	Masked imaging data from testing set of type `antsImage` or `matrix`. This will typically be read from a `test.mha` file generated by, e.g., `sccan --imageset-to-matrix`.
`demog.train`	Data frame of demographics information for training images.
`demog.test`	Data frame of demographics information for testing images.
`eigenvectors`	List of eigenvector images for dimensionality reduction.
`mask`	Mask image of type `antsImage`.
`outcome`	Name of outcome variable to be predicted. Must be present in `demog.train` and `demog.test`.
`covariates`	List of names of covariates to be used for the prediction. All names must be present in `demog.train` and `demog.test`.
`model.function`	Modeling function for predicting outcome from input data. Can be any function that has `formula` and `data` arguments and a `predict` method. Defaults to `glm`, but svm, randomForest, etc. will also work (assuming necessary libraries are loaded).
`which.eigenvectors`	Method for selecting eigenvectors. Can be either `all` (uses all eigenvectors in `vector.names`) or `optimal` (uses BIC stepwise model selection to select eigenvectors). `optimal` only works for `model.function` arguments that include an `extractAIC` method and a `coefficients` attribute.
`...`	Additional arguments for input to `model.function`. Example input would be `family=binomial` for classification instead of regression when using `glm`.

Details

`regressProjections` is a convenient way to perform training and testing of predictions of demographic information from imaging data. It takes as input demographics information, imaging data, and eigenvectors, and performs prediction of the outcome variable from the projection of the imaging data on the eigenvectors.

Value

A list of diagnostic and statistical information generated from the prediction, including: `stats`: Statistics on computed fits. For numeric outcomes, mean squared error, correlation coefficients, and p-value of prediction for training and testing data. For factor outcomes, misclassification rate and p-value of classification model. `outcome.comparison`: Data frame comparing real vs. predicted values for testing data. `eigenvectors`: List of eigenvectors retained in model building.

Author(s)

Kandel BM and Avants B

See Also

`sparseDecom2`

Examples

```
# generate simulated outcome
nsubjects <- 100
x1 <- seq(1, 10, length.out=nsubjects) + rnorm(nsubjects, sd=2)
x2 <- seq(25, 15, length.out=nsubjects) + rnorm(nsubjects, sd=2)
outcome <- 3 * x1 + 4 * x2 + rnorm(nsubjects, sd=1)
# generate simulated images with outcome predicted
# by sparse subset of voxels
voxel.1 <- 3 * x1 + rnorm(nsubjects, sd=2)
voxel.2 <- rnorm(nsubjects, sd=2)
voxel.3 <- 2 * x2 + rnorm(nsubjects, sd=2)
voxel.4 <- rnorm(nsubjects, sd=3)
input   <- cbind(voxel.1, voxel.2, voxel.3, voxel.4)
# simulate eigenvectors and mask
mydecom <- sparseDecom(input, sparseness=0.25, nvecs=4)
mask    <- as.antsImage(matrix(c(1,1,1,1), nrow=2))
# generate sample demographics that do not explain outcome
age <- runif(nsubjects, 50, 75)
demog <- data.frame(outcome=outcome, age=age)
# randomly divide data into training and testing
data.split <- splitData(demog, 2/3, return.rows=TRUE)
result <- regressProjections(input[data.split$rows.in, ],
  input[data.split$rows.out, ],
  data.split$data.in, data.split$data.out,
  mydecom$eigenanatomyimages,
  mask, outcome)
```

---

renderImageLabels *3D surface-based rendering of image segmentation labels*

Description

Will use rgl to render surfaces

Usage

```
renderImageLabels(labelsimg, surfval = 0.5, smoothsval = 0, alphasurf = 1,
  physical = TRUE, color = c())
```

Arguments

labelsimg	3D images of integer labels
surfval	intensity level that defines isosurface
smoothsval	sigma for smoothing of each extracted label image
alphasurf	opacity of each rendered surface
physical	flag to use true spatial coordinates
color	colors to use for each label

Value

0 – Success
1 – Failure

Author(s)

Duda, J

Examples

```
## Not run:
renderImageLabels(labels)
renderImageLabels(labels, smoothsval=0.5, alphasurf=0.3  )

## End(Not run)
```

---

renderSurfaceFunction *3D surface-based rendering of volume images.*

Description

Will use rgl to render a substrate (e.g. anatomical) and overlay image (e.g. functional).

Usage

```
renderSurfaceFunction(surfimg, funcimg, surfval = 0.5, basefval, offsetfval,
  smoothsval = 0, smoothfval = 0, blobrender = TRUE, alphasurf = 1,
  alphafunc = 1, outdir = "./", outfn = NA, mycol, physical = TRUE)
```

Arguments

surfimg	Input image to use as rendering substrate.
funcimg	Input list of images to use as functional overlays.
surfval	intensity level that defines isosurface
basefval	intensity level that defines lower threshold for functional image
offsetfval	intensity level that defines upper threshold for functional image
smoothsval	smoothing for the surface image
smoothfval	smoothing for the functional image
blobrender	render a blob as opposed to a surface patch

alphasurf	alpha for the surface contour
alphafunc	alpha value for functional blobs
outdir	output directory
outfn	output file name
mycol	name of color or colors
physical	boolean

Value

0 – Success
1 – Failure

Author(s)

Avants B, Kandel B

See Also

plotBasicNetwork

Examples

```
## Not run:
       mnit<-getANTsRData("mni")
       mnit<-antsImageRead(mnit)
       mnia<-getANTsRData("mnia")
       mnia<-antsImageRead(mnia)
       mnit<-thresholdImage( mnit, 1, max(mnit) )
       mnia<-thresholdImage( mnia, 1, 2 )
       brain<-renderSurfaceFunction( surfimg =list( mnit ) ,
          list(mnia), alphasurf=0.1 ,smoothsval = 1.5 )

## End(Not run)
```

---

reorientImage *reorient image by its principal axis*

Description

align along a specific axis

Usage

```
reorientImage(img, axis1, axis2 = NA, doreflection = 0, doscale = 0,
  txfn = NA)
```

Arguments

img	antsImage
axis1	vector of size dim, might need to play w/axis sign
axis2	vector of size dim for 3D
doreflection	boolean
doscale	scalar value, 1 allows automated estimate of scaling
txfn	file name for transformation

Value

reoriented image

Author(s)

Brian B. Avants

Examples

```
fi<-antsImageRead( getANTsRData("r16"))
reofi<-reorientImage(fi,c(1,0))
```

---

resampleImage *resampleImage*

Description

Resample image by spacing or number of voxels with various interpolators

Usage

```
resampleImage(image, resampleParams, useVoxels = 0, interpType = 1)
```

Arguments

image	input antsImage matrix
resampleParams	vector of size dimension with numeric values
useVoxels	true means interpret resample params as voxel counts
interpType	one of 0 (linear), 1 (nearest neighbor), 2 (gaussian), 3 (windowed sinc), 4 (bspline)

Value

output antsImage

Author(s)

Avants BB

Examples

```
fi<-antsImageRead( getANTsRData("r16"))
finn<-resampleImage(fi,c(50,60),1,0)
filin<-resampleImage(fi,c(1.5,1.5),0,1)
```

`rfSegmentation` *A rfSegmentation function.*

Description

Image segmentation via random forests.

Usage

```
rfSegmentation(labelimg, featureimages, ntrees = 100, verbose = FALSE)
```

Arguments

`labelimg`	input antsImage labelimage — assume non-zero entries create a mask
`featureimages`	input list of antsImage feature images - length n means n predictors in rf
`ntrees`	number of rf trees
`verbose`	boolean

Value

list of n-probability images is output where n is number of classes

Author(s)

Tustison NJ, Avants BB

Examples

```
if ( usePkg("randomForest") ) {
img<-antsImageRead( getANTsRData("r16"))
mask<-getMask( img )
segs<-kmeansSegmentation( img, k=3, kmask = mask)
fimgs<-lappend( img, segs$probabilityimages )
rfsegs<-rfSegmentation( segs$segmentation, fimgs , verbose=TRUE )
plot( rfsegs$segmentation )
# now use in atropos w/priors
segs2<-atropos( a = img, m = [0.2,1x1],
  c = [5,0],  i = rfsegs$probabilityimages, x = mask)
 }
```

`rfSegmentationPredict` *A rfSegmentationPredict function.*

Description

Predict image segmentation via random forests.

Usage

```
rfSegmentationPredict(rfSegmentationModel, featureimages, mask,
  verbose = FALSE)
```

Arguments

rfSegmentationModel	input rf model
featureimages	input list of antsImage feature images
mask	antsImage mask
verbose	bool

Value

segmentation is output

Author(s)

Tustison NJ, Avants BB

Examples

```
if ( usePkg(randomForest) ) {
img<-antsImageRead( getANTsRData("r16"))
mask<-getMask( img )
mask2<-getMask( img )
mask [ 129:255, 1:255 ]<-0
mask2[ 2:128, 1:255 ]<-0
segs<-kmeansSegmentation( img, k=3, kmask = mask)
fimgs<-list( img )
rfsegs<-rfSegmentation( segs$segmentation, fimgs , ntrees=100 )
rfseg2<-rfSegmentationPredict(  rfsegs$rfModel , fimgs , mask2 )
plot( rfseg2 )
## Not run:
   img<-antsImageRead( getANTsRData("r16"))
   img2<-antsImageRead( getANTsRData("r64"))
   mask<-getMask( img )
   mask2<-getMask( img2 )
   segs<-kmeansSegmentation( img, k=3, kmask = mask)
   nimg<-iMath(img,"Normalize")
   fimgs<-list( nimg )
   rfsegs<-rfSegmentation( segs$segmentation, fimgs , ntrees=100 )
   mytx<-antsRegistration(fixed=img , moving=img2 ,
            typeofTransform = c(SyN) )
   fimgs2<-list( iMath(mytx$warpedmovout,"Normalize") )
   rfseg2<-rfSegmentationPredict(  rfsegs$rfModel , fimgs2 , mask )
   plot( rfseg2 )

## End(Not run)
}
```

---

rsfDenoise *WIP: data-driven denoising for resting state fMRI*

Description

Uses a target function to denoise resting bold data

Usage

```
rsfDenoise(boldmatrix, targety, motionparams = NA, selectionthresh = 0.1,
  maxnoisepreds = 1:12, debug = FALSE, polydegree = 4,
  crossvalidationgroups = 4, tr = 1, scalemat = F, noisepoolfun = max)
```

Arguments

boldmatrix	input bold matrix
targety	target to predict
motionparams	motion parameters / nuisance variables
selectionthresh	e.g. 0.1 take 10 percent worst variables for noise estimation
maxnoisepreds	integer search range e.g 1:10
debug	boolean
polydegree	eg 4 for polynomial nuisance variables
crossvalidationgroups	prior defined or integer valued
tr	bold tr
scalemat	boolean
noisepoolfun	function to help select noise pool e.g. max

Value

matrix is output

Author(s)

Avants BB

Examples

```
## Not run:
# if (!exists("fn") ) fn<-getANTsRData("pcasl")
# bold <- antsImageRead( fn )
# avgbold<-getAverageOfTimeSeries(bold)
# boldmask<-getMask( avgbold )
# roimask<-antsImageRead("roi.nii.gz")
# timeselect<-10:(dim(bold)[4]-10)
# # can do this if you like its approach
# cleanfMRI <- preprocessfMRI( bold, maskImage=boldmask,
#   frequencyLowThreshold = 0.01, frequencyHighThreshold = 0.1,
#   spatialSmoothingType = "gaussian", spatialSmoothingParameters = 2,
#   residualizeMatrix=TRUE, numberOfCompCorComponents=2 )
# boldmat<-timeseries2matrix( cleanfMRI$cleanBoldImage, cleanfMRI$maskImage )
# roimat<-timeseries2matrix( cleanfMRI$cleanBoldImage, roimask )
# roimean<-rowMeans( roimat ) # svd instead?
# roimat<-matrix( roimean, ncol=1)
# dnz<-rsfDenoise( boldmat[timeselect,] ,
#   roimat[timeselect,1], motionparams=NA,
#   polydegree=1, crossvalidationgroups = 8, maxnoisepreds=c(2:4), debug=F )
# # might iterate over above to further refine noise variables
# mdl<-bigLMStats( lm( boldmat[timeselect,] ~ roimean[timeselect] +
```

```
#    dnz$polys + dnz$noiseu  ), 0.001 )
# betas<-mdl$beta.t[1,]
# sum(betas[betas > 3])
# betaimg<-antsImageClone( boldmask )
# betaimg[ boldmask == 1 ]<-betas
# antsImageWrite( betaimg, "betas2.nii.gz" )
#
# # more complex
# bold<-antsImageRead("bold.nii.gz")
# boldmask<-antsImageRead("meanboldmask.nii.gz")
# aalimg<-antsImageRead("meanboldAALmask.nii.gz")
# data("aal",package="ANTsR")
# dmnlabels<-aal$label_num[aal$isdmn>0]
# aalvec<-aalimg > 0
# whichregion<-3
# for ( i in 1:max(aalimg) )
#   {
#   if ( ! ( i %in% dmnlabels[whichregion] ) )
#     {
#     aalimg[ aalimg == as.numeric(i) ]<-0
#     }
#   }
# maskvec<-boldmask > 0 & aalimg == whichregion
# boldmask[ maskvec ]<-0
# timeselect<-10:dim(bold)[4]
# if ( ! exists("moco") ) {
#   moco<-.motion_correction(bold,moreaccurate=1)
#   moco<-as.matrix( moco$moco_params )[timeselect,3:ncol(moco$moco_params)]
# }
# boldmat<-timeseries2matrix(bold,boldmask)
# boldmat<-boldmat[timeselect,]
# aalimg[aalimg > 0 ]<-1
# dmnvec<-rowMeans(timeseries2matrix(bold,aalimg))[timeselect]
# dmnvec<-(stl(ts(dmnvec, frequency = 4),"per")$time.series)[,2]
# dmnvec2<-(stl(ts(dmnvec, frequency = 100),"per")$time.series)[,2]
# dmnvec<-ts(as.numeric(dmnvec)-as.numeric(dmnvec2))
# dmnmat<-matrix( dmnvec, ncol=1)
# dnz<-rsfDenoise( boldmat , dmnmat[,1], motionparams=moco, polydegree=4,
#                 crossvalidationgroups = 8, maxnoisepreds=1:4, debug=F )
# print(paste("Best number of noise regressors",dnz$n))
# # now recompute the matrix using the full mask
# boldmask<-antsImageRead("meanboldmask.nii.gz")
# boldmat<-timeseries2matrix(bold,boldmask)
# boldmat<-boldmat[timeselect,]
# mdl<-bigLMStats( lm( boldmat ~ dmnmat[,1] + dnz$polys + dnz$noiseu ), 0.001 )
# betas<-mdl$beta.t[1,]

## End(Not run)
```

---

save.ANTsR *save.ANTsR*

Description

Save and load ANTsR sessions.

Usage

```
save.ANTsR(filename="./.ANTsRsession", objects=NA,
  env=as.environment(1), ...)

load.ANTsR(filename="./.ANTsRsession", env=as.environment(1))
```

Arguments

filename	Prefix for folder to store data.
objects	Vector of character names of objects to store. Can be antsImages.
env	Environment to save from or load to.
...	Additional arguments to pass to save.

Examples

```
## Not run:  # causes problems with devtools::run_examples()
# a <- 1
# b <- c( 2, 3, 4)
# save.ANTsR(objects=c(b, a))
# load.ANTsR("./.ANTsRsession")

## End(Not run)
```

---

segmentShapeFromImage *convolution-based shape identification*

Description

takes an image and a shape and relates the 2nd with the first to yield a feature image - the function is user modifiable but defaults to cor

Usage

```
segmentShapeFromImage(img, shape, mask = NA, rad = NA, scfun,
  maskZeroes = TRUE)
```

Arguments

img	antsImage
shape	to define features
mask	with values 1 or 0
rad	max radius (optional, not recommended)
scfun	function to apply to create feature image (defaults to cor)
maskZeroes	if TRUE (default), zeroes will not influence the result

Value

feature image

Author(s)

Brian B. Avants

Examples

```
set.seed(123)
fi<-makeImage(c(20,20),rnorm(400,mean=1,sd=0.1))
mask<-getMask(fi,0.8,Inf,0)
segs<-kmeansSegmentation( fi ,3 , mask)$segmentation
segs[ segs != 1 ]<-0
shp<-labelClusters( segs,1)
shp[ shp != 1 ]<-0
fimg<-segmentShapeFromImage(fi,shp,mask)
```

---

sliceTimingCorrection *slice timing correction for fMRI.*

Description

Alters BOLD signal according to acquisition parameters (TR).

Usage

```
sliceTimingCorrection(fmri, sliceTime = NA, interpolation = "sinc",
  sincRadius = 4, bsplineOrder = 3)
```

Arguments

fmri	input bold image
sliceTime	the TR
interpolation	sinc recommended but linear is much faster
sincRadius	recommend 4 for interpolation
bsplineOrder	recommend 3 for interpolation

Value

matrix is output

Author(s)

Avants BB

smoothImage *Smooth image*

Description

Perform smoothing on the given image with a given sigma, defined in physical space units

Usage

```
smoothImage( inimg, sigma )
```

Arguments

inimg	Input image to operate on
sigma	Smoothing factor. Either scalar, or vector of length dim for dim-dimensional image.

Value

antsImage smoothed

Author(s)

Shrinidhi KL, Avants BB

Examples

```
img<-makeImage(c(5,5),rnorm(25))
simg<-smoothImage( img ,c(1.2,1.5) )
```

sparseDecom *Convenience wrapper for eigenanatomy decomposition.*

Description

Decomposes a matrix into sparse eigenevectors to maximize explained variance.

Usage

```
sparseDecom(inmatrix = NA, inmask = 0, sparseness = 0.01, nvecs = 50,
  its = 5, cthresh = 250, statdir = NA, z = 0, smooth = 0,
  initializationList = list(), mycoption = 0, robust = 0, ell1 = 1,
  getSmall = 0)
```

Arguments

inmatrix	n by p input images , subjects or time points by row , spatial variable lies along columns
inmask	optional antsImage mask
sparseness	lower values equal more sparse
nvecs	number of vectors
its	number of iterations
cthresh	cluster threshold
statdir	place on disk to save results
z	u penalty, experimental
smooth	smoothness eg 0.5
initializationList	see initializeEigenanatomy
mycoption	0, 1 or 2 all produce different output 0 is combination of 1 (spatial orthogonality) and 2 (subject space orthogonality)
robust	rank transform input data - good for data checking
ell1	the ell1 grad descent param
getSmall	try to get smallest evecs (bool)

Value

outputs a decomposition of a population or time series matrix

Author(s)

Avants BB

Examples

```
mat<-replicate(100, rnorm(20))
mydecom<-sparseDecom( mat )
## Not run:
# for prediction
if ( usePkg("randomForest") & usePkg("spls")  & usePkg(BGLR) ) {
data(lymphoma) # from spls
training<-sample( rep(c(TRUE,FALSE),31)  )
sp<-0.02 ; myz<-0
ldd<-sparseDecom( lymphoma$x[training,], nvecs=5 , sparseness=( sp ),
  mycoption=1, z=myz ) # NMF style
traindf<-data.frame( lclass=as.factor(lymphoma$y[ training  ]),
  eig = lymphoma$x[training,]  %*% as.matrix(ldd$eigenanatomyimages ))
testdf<-data.frame(  lclass=as.factor(lymphoma$y[ !training ]),
 eig = lymphoma$x[!training,] %*% as.matrix(ldd$eigenanatomyimages ))
myrf<-randomForest( lclass ~ . ,   data=traindf )
predlymp<-predict(myrf, newdata=testdf)
print(paste(N-errors:,sum(abs( testdf$lclass != predlymp ) ),
   non-zero ,sum(abs( ldd$eigenanatomyimages ) > 0 ) ) )
# compare to http://arxiv.org/pdf/0707.0701v2.pdf
# now SNPs
data(mice)
```

```
snps<-quantifySNPs( mice.X, shiftit = TRUE )
numericalpheno<-as.matrix( mice.pheno[,c(4,5,13,15) ] )
nfolds<-6
train<-sample( rep( c(1:nfolds), 1800/nfolds ) )
train<-( train < 4 )
lrmat<-lowrankRowMatrix( as.matrix( snps[train,] ) ,  50 )
snpd<-sparseDecom( lrmat, nvecs=20 , sparseness=( 0.001), z=-1 )
projmat<-as.matrix( snpd$eig )
snpse<-as.matrix( snps[train, ]  ) %*% projmat
traindf<-data.frame( bmi=numericalpheno[train,3] , snpse=snpse)
snpse<-as.matrix( snps[!train, ]  ) %*% projmat
testdf <-data.frame( bmi=numericalpheno[!train,3] , snpse=snpse )
myrf<-randomForest( bmi ~ . , data=traindf )
preddf<-predict(myrf, newdata=testdf )
cor.test(preddf, testdf$bmi )
plot(preddf, testdf$bmi )
}

## End(Not run)
```

`sparseDecom2` *Convenience wrapper for 2-view eigenanatomy decomposition.*

Description

Decomposes two matrices into paired sparse eigenevectors to maximize canonical correlation.

Usage

```
sparseDecom2(inmatrix, inmask = c(NA, NA), sparseness = c(0.01, 0.01),
  nvecs = 3, its = 2, cthresh = c(0, 0), statdir = NA, perms = 0,
  uselong = 0, z = 0, smooth = 0, robust = 0, mycoption = 0,
  initializationList = list(), initializationList2 = list(), ell1 = 0.05,
  priorWeight = 0)
```

Arguments

`inmatrix`	input as inmatrix=list(mat1,mat2). n by p input matrix and n by q input matrix , spatial variable lies along columns.
`inmask`	optional pair of antsImage masks
`sparseness`	a c(.,.) pair of values e.g c(0.01,0.1) enforces an unsigned 99 percent and 90 percent sparse solution for each respective view
`nvecs`	number of eigenvector pairs
`its`	number of iterations, 10 or 20 usually sufficient
`cthresh`	cluster threshold pair
`statdir`	temporary directory if you want to look at full output
`perms`	number of permutations
`uselong`	enforce solutions of both views to be the same - requires matrices to be the same size
`z`	subject space (low-dimensional space) sparseness value

smooth	smooth the data (only available when mask is used)
robust	rank transform input matrices
mycoption	enforce 1 - spatial orthogonality, 2 - low-dimensional orthogonality or 0 - both
initializationList	initialization for first view
initializationList2	initialization for 2nd view
ell1	gradient descent parameter, if negative then l0 otherwise use l1
priorWeight	Scalar value weight on prior between 0 (prior is weak) and 1 (prior is strong). Only engaged if initialization is used

Value

outputs a decomposition of a pair of matrices

Author(s)

Avants BB

Examples

```
mat<-replicate(100, rnorm(20))
mat2<-replicate(100, rnorm(20))
mydecom<-sparseDecom2( inmatrix=list(mat,mat2),
  sparseness=c(0.1,0.3) , nvecs=3, its=3, perms=0)
wt<-0.666
mat3<-mat*wt+mat2*(1-wt)
mydecom<-sparseDecom2( inmatrix=list(mat,mat3),
  sparseness=c(0.2,0.2), nvecs=5, its=10, perms=5 )

## Not run:
# a masked example
im<-antsImageRead( getANTsRData("r64"))
dd<- im > 250
mask<-antsImageClone( im )
mask[ !dd ]<-0
mask[ dd ]<-1
mat1<-matrix( rnorm(sum(dd)*10) , nrow=10 )
mat2<-matrix( rnorm(sum(dd)*10) , nrow=10 )
initlist<-list()
for ( nvecs in 1:2 ) {
  init1<-antsImageClone( mask )
  init1[dd]<-rnorm(sum(dd))
  initlist<-lappend( initlist, init1 )
}
ff<-sparseDecom2( inmatrix=list(mat1,mat2), inmask=list(mask,mask),
  sparseness=c(0.1,0.1) ,nvecs=length(initlist) , smooth=1,
  cthresh=c(0,0), initializationList = initlist ,ell1 = 11 )
### now SNPs ###
rf<-usePkg(randomForest)
bg<-usePkg(BGLR)
if ( bg & rf ) {
data(mice)
snps<-mice.X
numericalpheno<-as.matrix( mice.pheno[,c(4,5,13,15) ] )
```

```
numericalpheno<-residuals( lm( numericalpheno ~
  as.factor(mice.pheno$Litter) ) )
nfolds<-6
train<-sample( rep( c(1:nfolds), 1800/nfolds ) )
train<-( train < 4 )
snpd<-sparseDecom2( inmatrix=list( ( as.matrix(snps[train,]) ),
  numericalpheno[train,] ), nvecs=20, sparseness=c( 0.001, -0.5 ),
  its=3, ell1=0.1 , z=-1 )
for ( j in 3:3) {
traindf<-data.frame( bmi=numericalpheno[ train,j] ,
   snpse=as.matrix( snps[train, ] ) %*% as.matrix( snpd$eig1 ) )
testdf <-data.frame( bmi=numericalpheno[!train,j] ,
   snpse=as.matrix( snps[!train,] ) %*% as.matrix( snpd$eig1 ) )
myrf<-randomForest( bmi ~ . , data=traindf )
preddf<-predict(myrf, newdata=testdf )
print( cor.test(preddf, testdf$bmi ) )
plot( preddf, testdf$bmi )
}
} # check bg and rf

## End(Not run)
```

`sparseDecom2boot` *Convenience wrapper for 2-view eigenanatomy decomposition w/bootstrap initialization.*

Description

Decomposes two matrices into paired sparse eigenevectors to maximize canonical correlation.

Usage

```
sparseDecom2boot(inmatrix, inmask = c(NA, NA), sparseness = c(0.01, 0.01),
  nvecs = 50, its = 5, cthresh = c(0, 0), statdir = NA, perms = 0,
  uselong = 0, z = 0, smooth = 0, robust = 0, mycoption = 1,
  initializationList = list(), initializationList2 = list(), ell1 = 0.05,
  nboot = 10, nsamp = 1, doseg = FALSE)
```

Arguments

`inmatrix`	input as inmatrix=list(mat1,mat2). n by p input matrix and n by q input matrix , spatial variable lies along columns.
`inmask`	optional pair of antsImage masks
`sparseness`	a c(.,.) pair of values e.g c(0.01,0.1) enforces an unsigned 99 percent and 90 percent sparse solution for each respective view
`nvecs`	number of eigenvector pairs
`its`	number of iterations, 10 or 20 usually sufficient
`cthresh`	cluster threshold pair
`statdir`	temporary directory if you want to look at full output
`perms`	number of permutations

uselong	enforce solutions of both views to be the same - requires matrices to be the same size
z	subject space (low-dimensional space) sparseness value
smooth	smooth the data (only available when mask is used)
robust	rank transform input matrices
mycoption	enforce 1 - spatial orthogonality, 2 - low-dimensional orthogonality or 0 - both
initializationList	initialization for first view
initializationList2	initialization for 2nd view
ell1	gradient descent parameter, if negative then l0 otherwise use l1
nboot	n bootstrap runs
nsamp	number of samples e.g. 0.9 indicates 90 percent of data
doseg	boolean to control matrix orthogonality during bootstrap

Value

outputs a decomposition of a pair of matrices

Author(s)

Avants BB

Examples

```
## Not run:
mat<-replicate(100, rnorm(20))
mat2<-replicate(100, rnorm(20))
mydecom<-sparseDecom2boot( inmatrix=list(mat,mat2),
  sparseness=c(0.1,0.3) , nvecs=3, its=3, perms=0)
wt<-0.666
mat3<-mat*wt+mat2*(1-wt)
mydecom<-sparseDecom2boot( inmatrix=list(mat,mat3),
  sparseness=c(0.2,0.2), nvecs=5, its=10, perms=200 )


## End(Not run)
```

---

sparseDecomboot *Convenience wrapper for eigenanatomy decomposition.*

Description

Decomposes a matrix into sparse eigenevectors to maximize explained variance.

Usage

```
sparseDecomboot(inmatrix = NA, inmask = 0, sparseness = 0.01,
  nvecs = 50, its = 5, cthresh = 250, statdir = NA, z = 0,
  smooth = 0, initializationList = list(), mycoption = 0, nboot = 10,
  nsamp = 0.9, robust = 0, doseg = TRUE)
```

Arguments

inmatrix	n by p input images , subjects or time points by row , spatial variable lies along columns
inmask	optional antsImage mask
sparseness	lower values equal more sparse
nvecs	number of vectors
its	number of iterations
cthresh	cluster threshold
statdir	place on disk to save results
z	u penalty, experimental
smooth	smoothness eg 0.5
initializationList	see initializeEigenanatomy
mycoption	0, 1 or 2 all produce different output 0 is combination of 1 (spatial orthogonality) and 2 (subject space orthogonality)
nboot	boostrap integer e.g. 10 equals 10 boostraps
nsamp	value less than or equal to 1, e.g. 0.9 means 90 percent of data will be used in each boostrap resampling
robust	boolean
doseg	orthogonalize bootstrap results

Author(s)

Avants BB

Examples

```
mat<-replicate(100, rnorm(20))
mydecom<-sparseDecomboot( mat, nboot=5, nsamp=0.9, nvecs=2 )

## Not run:
# for prediction
if ( usePkg("randomForest") & usePkg("spls") ) {
data(lymphoma)
training<-sample( rep(c(TRUE,FALSE),31)  )
sp<-0.001 ; myz<-0 ; nv<-5
ldd<-sparseDecomboot( lymphoma$x[training,], nvecs=nv ,
  sparseness=( sp ), mycoption=1, z=myz , nsamp=0.9, nboot=50 ) # NMF style
outmat<-as.matrix(ldd$eigenanatomyimages )
# outmat<-t(ldd$cca1outAuto)
traindf<-data.frame( lclass=as.factor(lymphoma$y[ training  ]),
  eig = lymphoma$x[training,]  %*% outmat )
testdf<-data.frame(  lclass=as.factor(lymphoma$y[ !training ]),
  eig = lymphoma$x[!training,] %*% outmat )
myrf<-randomForest( lclass ~ . ,   data=traindf )
predlymp<-predict(myrf, newdata=testdf)
print(paste(N-errors:,sum(abs( testdf$lclass != predlymp ) ),
  non-zero ,sum(abs( outmat ) > 0 ) ) )
for ( i in 1:nv )
  print(paste( non-zero ,i, is: ,sum(abs( outmat[,i] ) > 0 ) ) )
```

```
}

## End(Not run) # end dontrun
```

`sparseRegression` *Sparse regression on input images.*

Description

Compute a sparse, spatially coherent regression from a set of input images (with mask) to an outcome variable.

Usage

```
sparseRegression(inmatrix, demog, outcome, mask = NA, sparseness = 0.05,
  nvecs = 10, its = 5, cthresh = 250, statdir = NA, z = 0,
  smooth = 0)
```

Arguments

`inmatrix`	Input data matrix, with dimension number of subjects by number of voxels.
`demog`	Input demographics data frame. Contains outcome variable to regress against.
`outcome`	Name of column in `demog` to regress against.
`mask`	Mask for reconstructing `inmatrix` in physical space.
`sparseness`	Level of sparsity desired. `0.05`, for example, makes 5% of the matrix be non-zero.
`nvecs`	Number of eigenvectors to return.
`its`	Number of cross-validation folds to run.
`cthresh`	Cluster threshold.
`statdir`	Where to put results. If not provided, a temp directory is created.
`z`	Row (subject-wise) sparseness.
`smooth`	Amount of smoothing.

Value

A list of values:

`eigenanatomyimages`	Coefficient vector images.
`umatrix`	Projections of input images on the sparse regression vectors. Can be used for, e.g., subsequent classification/predictions.
`projections`	Predicted values of outcome variable.

Author(s)

Kandel BM, Avants BB.

References

Kandel B.M., D. Wolk, J. Gee, and B. Avants. Predicting Cognitive Data from Medical Images Using Sparse Linear Regression. Information Processing in Medical Imaging, 2013. literature/web site here ~

Examples

```
## Not run:
nsubj <- 1000
prop.train <- 1/2
subj.train <- sample(1:nsubj, prop.train*nsubj, replace=F)
input <- t(replicate(nsubj, rnorm(125)))
outcome <- seq(1, 5, length.out=nsubj)
demog <- data.frame(outcome=outcome)
input[, 40:60] <- 30 + outcome + rnorm(length(input[, 40:60]), sd=2)
input.train <- input[subj.train, ]
input.test <- input[-subj.train, ]
demog.train <- data.frame(outcome=demog[subj.train, ])
demog.test <- data.frame(outcome=demog[-subj.train, ])
mymask <- as.antsImage(array(rep(1, 125), dim=c(5,5,5)))
myregression <- sparseRegression(input.train, demog.train, outcome, mymask,
    sparseness=0.05, nvecs=5, its=3, cthresh=250)
# visualization of results
sample <- rep(0, 125)
sample[40:60] <-1
signal.img <- as.antsImage(array(rep(0,125), dim=c(5, 5, 5)))
signal.img[signal.img >= 0 ] <- sample
plot( signal.img, axis=2, slices=1x5x1) # actual source of signal
# compare against first learned regression vector
myimgs <- list()
for( i in 1:5){
  myarray <- as.array(myregression$eigenanatomyimages[[ i ]])
  myarray <- myarray / max(abs(myarray)) # normalize for visualization
  myimgs[[ i ]] <- antsImageClone(myregression$eigenanatomyimages[[ i ]])
  myimgs[[ i ]][mymask > 0] <- myarray
}
plot(myimgs[[1]], axis=2, slices=1x5x1)
# use learned eigenvectors for prediction
result <- regressProjections(input.train, input.test, demog.train,
    demog.test, myregression$eigenanatomyimages, mymask, outcome)
plot(result$outcome.comparison$real, result$outcome.comparison$predicted)

## End(Not run)
```

`spatialbayesianlm` *spatially constrained bayesian regression function.*

Description

Take a standard lm result and use bayesian regression to impose spatial regularity.

Usage

```
spatialbayesianlm(mylm, ymat, mask, smth = 1, priorWeight = 1, nhood = NA,
  regweights = NA, smoothcoeffmat = NA)
```

Arguments

mylm	standard lm result of the form mylm<-lm(ymat~.)
ymat	outcome matrix - usually from imaging data
mask	mask with non-zero entries n-columns of ymat
smth	smoothness parameter
priorWeight	weight on the prior
nhood	size of neighborhood
regweights	weights on rows - size of ymat
smoothcoeffmat	prior coefficient matrix

Value

bayesian regression solution is output as a list of images

Author(s)

Avants BB

Examples

```
# make some simple data
  ## Not run:
  if (!exists("fn") ) fn<-getANTsRData("pcasl")
  asl<-antsImageRead(fn)
  tr<-antsGetSpacing(asl)[4]
  aslmean<-getAverageOfTimeSeries( asl )
  aslmask<-getMask(aslmean,lowThresh=mean(aslmean),cleanup=TRUE)
  pcaslpre <- aslPerfusion( asl, dorobust=0, useDenoiser=NA, skip=1,
     useBayesian=0, moreaccurate=0, verbose=T, mask=aslmask )
  # user might compare to useDenoiser=FALSE
  pcasl.parameters <- list( sequence="pcasl", m0=pcaslpre$m0 )
  aslmat<-timeseries2matrix(asl,aslmask)
  tc<-as.factor(rep(c("C","T"),nrow(aslmat)/2))
  dv<-computeDVARS(aslmat)
  perfmodel<-lm( aslmat ~ tc + stats::poly(dv,4) ) # standard model
  ssp<-spatialbayesianlm( perfmodel, aslmat, aslmask,
    priorWeight=1.e2 ,smth=1.6, nhood=rep(2,3) )
  plot( ssp[[1]], slices="2x16x2", axis=3 )

## End(Not run)
```

---

splitChannels *split a multichannel antsImage*

Description

split a multichannel antsImage into a list of scalar antsImages

Usage

```
splitChannels(image)
```

Arguments

image	a multichannel antsImage to split

Value

list of scalar antsImages

Author(s)

Duda, JT

Examples

```
r <- floor( seq(1:(64*64)) / (64*64) * 255 )
 dim(r) <- c(64,64)
 r <- as.antsImage(r)
 g <- r*0
 b <- r*0
 rgbImage = mergeChannels( list(r,g,b) )
 imgList = splitChannels( rgbImage )
```

---

splitData *Split data for testing and training*

Description

Split data into testing and training sets for cross-validation.

Usage

```
splitData(data.source, ratio, return.rows = FALSE)
```

Arguments

data.source	Data frame or matrix of input data to be split.
ratio	If greater than one, number of folds to split data into. Otherwise, proportion of rows in training data. See Value.
return.rows	If TRUE, row numbers of testing and training data are also returned.

Value

A list containing the input data, split into testing and training sets. If ratio is greater than one, returns a list with ratio entries, each with 1/ratio of the input data in the testing set (data.out) and the rest in the training set (data.in). Otherwise, returns a list with one split of the data, with ratio of the input data in the testing set and the rest in the training set.

Author(s)

Kandel BM and Avants B

Examples

```
## Not run:
  n <- 30
  ratio <- 2/3
  data.source <- data.frame(value=1:n)
  out <- splitData(data.source, ratio)

## End(Not run)
```

`subgradientL1Regression` *Simple subgradientL1Regression function.*

Description

SubgradientL1Regression solves y approx x beta

Usage

```
subgradientL1Regression(y, x, s = 0.01, percentvals = 0.1, nits = 100,
  betas = NA, sparval = NA)
```

Arguments

`y`	outcome variable
`x`	predictor matrix
`s`	gradient descent parameter
`percentvals`	percent of values to use each iteration
`nits`	number of iterations
`betas`	initial guess at solution
`sparval`	sparseness

Value

output has a list of summary items

Author(s)

Avants BB

Examples

```
mat<-replicate(1000, rnorm(200))
y<-rnorm(200)
wmat<-subgradientL1Regression( y, mat, percentvals=0.05 )
print( wmat$resultcorr )
```

subjectDataToGroupDataFrame
convert subject matrix data to a row in a dataframe.

Description

SubjectDataToGroupDataFrames take a list of subject-level csv files of the same type and converts them to a data frame with consistent column naming for the values and where rows correspond to subjects.

Usage

```
subjectDataToGroupDataFrame(csvlist, usecol, mycolname = NA,
  datarownames = NA)
```

Arguments

csvlist	input list of csv files eg by Sys.glob ...
usecol	a column name or number e.g. 1 or 'Volume'
mycolname	rename the column by this string (optional)
datarownames	the desired now names (optional)

Value

data frame is output

Author(s)

Avants BB

Examples

```
## Not run:
# data(aal,package="ANTsR")
# csvlist<-Sys.glob("*md*csv")
# mypopulationdataframe<-subjectDataToGroupDataFrame( csvlist , "Mean" ,
#  datarownames=aal$label_name )
# should have each ROI value for each subject listed in a large data frame

## End(Not run)
```

taskFMRI *Simple taskFMRI function.*

Description

Input 4D time series matrix. (Perform slice timing correction externally). Estimate hemodynamicRF from block design. Compute brain mask on average bold image. Get nuisance variables : motion , compcor , globalsignal. High-frequency filter the time series (externally). Correct for autocorrelation using bullmore 1996 MRM and AR(2) model with parameters derived from global residual signal. Estimate final glm.

Usage

```
taskFMRI(mat, hrf, myvars, correctautocorr = FALSE,
  residualizedesignmatrix = FALSE, myformula = NA)
```

Arguments

mat	input matrix
hrf	input hrf
myvars	output of getfMRInuisanceVariables
correctautocorr	correction auto correlation boolean
residualizedesignmatrix	boolean
myformula	statistical equation to be assessed at each voxel

Value

list of betas and other names entries is output

Author(s)

Avants BB

Examples

```
## Not run:
# read the fmri image in and maybe do slice timing correction
  fmri<-getANTsRData("pcasl")
  fmri<-antsImageRead( fmri )
#  fmri<-iMath(fmri,"SliceTimingCorrection","bspline") # optional
  myvars<-getfMRInuisanceVariables( fmri, moreaccurate = 0, maskThresh=100 )
  mat <- myvars$matrixTimeSeries
  mat<-frequencyFilterfMRI(mat, 2.5, freqLo=0.01, freqHi=0.1, opt="butt")
  blockfing = c(0, 36, 72 )
  hrf <- hemodynamicRF( scans=dim(fmri)[4] , onsets=blockfing ,
    durations=rep(  12,  length( blockfing ) ) ,  rt=2.5 )
  activationBeta<-taskFMRI(  mat , hrf , myvars )

## End(Not run)
```

temporalwhiten *Simple autocorrelation-based temporal whitening function.*

Description

Temporally whitens the input matrix using autoregression and returns the result.

Usage

```
temporalwhiten(mat, myord = 2)
```

Arguments

mat	input matrix
myord	integer order value

Value

matrix is output

Author(s)

Avants BB

Examples

```
mat<-replicate(100, rnorm(20))
wmat<-temporalwhiten( mat )
```

thresholdImage *Threshold Image*

Description

Threshold Image

Usage

```
thresholdImage(inimg, lothresh, hithresh, inval=1, outval=0)
```

Arguments

inimg	Input image to operate on
lothresh	Lower edge of threshold window
hithresh	Higher edge of threshold window
inval	Output value for image voxels in between lothresh and hithresh
outval	Output value for image voxels lower than lothresh or higher than hithresh

Value

antsImage

Author(s)

Shrinidhi KL

Examples

```
img <- makeImage(c(5,5), rnorm(25)+0.5)
imgt<-thresholdImage( img, 0.5, Inf )
```

---

timeseries2matrix *Time-series image to matrix*

Description

Extract a matrix from a time-series image after applying a mask.

Usage

```
timeseries2matrix(img, mask)
```

Arguments

img	Input image of type 'antsImage' or R array.
mask	Input mask of type 'antsImage' or R array. In either case, the number of voxels in the mask may be either equal to that of input image 'img' or equal to number of voxels in one time unit of input image 'img'. In the second case, the mask is reused for every time unit of the 'img'. A mask of n-labels will cause the function to return a matrix containing the mean time series within each label.

Value

Success – an R matrix of dimensions (dim(img)[length(dim(img))] , sum(mask==1))

Author(s)

Shrinidhi KL

Examples

```
img <- makeImage( c(10,10,10,5) , 0 )
# or use antsImageRead ...
mask <- array( 1 , dim(img)[1:3] )
mat <- timeseries2matrix( img , mask )
```

timeseriesN3 *Run N3 on slices of timeseries.*

Description

Repeatedly run N3 on slices of time series and return image

Usage

```
timeseriesN3(boldimg, mask, ncorrections = c(4, 2, 2))
```

Arguments

boldimg	input matrix
mask	mask to use
ncorrections	levels at which to apply n3

Value

antsImage is output

Author(s)

Avants BB

Examples

```
bold<-makeImage( c(10,10,10,4), rnorm(4000) )
  mask<-getMask( getAverageOfTimeSeries( bold ) )
  boldn3<-timeseriesN3( bold, mask, c(4,2) )
```

timeserieswindow2matrix *Time-series image to matrix of time windows around user-specified events*

Description

Extract a matrix from a time-series image after applying a mask where each row is a space-time vector

Usage

```
timeserieswindow2matrix(timeseriesmatrix, mask, eventlist, timewindow,
  zeropadvalue = 0, spacing = NA)
```

Arguments

timeseriesmatrix	Input timeseriesmatrix from timeseries2matrix
mask	Input mask of type 'antsImage' ... the mask will be replicated into a 4D image of length timewindow + zeropadvalue.
eventlist	time indices for the events to extract
timewindow	n-timepoints around each event, forward in time.
zeropadvalue	pads the mask by this amount fwd and bwd in time.
spacing	optional 4d spacing vector that will impact smoothing parameters

Value

Success – an R matrix of dimensions ntimewindow*sizeofnonzerovaluesinmask*nevents

Author(s)

Avants BB

Examples

```
img <- makeImage( c(10,10,10,50) , 0 )
mask <- as.antsImage( array( 1 , dim(img)[1:3] ) )
mat<-timeseries2matrix( img, mask )
mat <- timeserieswindow2matrix( mat , mask, c(1, 4, 5 ), 2, 0 )
print( dim(mat$eventmatrix) )
print( dim(mat$mask4d) )

##### another approach
dim4<-c( 20, 30, 10, 100 )
i1<-10:15
i2<-11:18
i3<-4:8
i4<-10:90
arr3d<-array(data = 0, dim = dim4[1:3] )
arr3d[i1,i2,i3]<-1
arr<-array(data = 0, dim =dim4 )
for ( t in i4 ) arr[,,,t]<-t
nois<-which( arr > 0 )
noisv<-rnorm( length(nois) )
arr[ nois ]<-arr[ nois ]+noisv*0.0
msk <- as.antsImage( arr3d )
img <- as.antsImage( arr )
mat<-timeseries2matrix( img, msk )
eanat<-sparseDecom(  mat,  msk, sparseness=0.1, z=0.5,nvecs=2, its=5,cthresh=0, mycoption=1)
eanat2<-sparseDecom( mat,    0, sparseness=0.1,z=0.5,nvecs=2, its=5,cthresh=0, mycoption=1)
enomask<-eanat2$eigenanatomyimages[,1]
emask<-eanat$eigenanatomyimages[[1]][ msk == 1 ]
print( enomask[31:40] )
print(   emask[31:40] )

# same thing with event matrices ....
ttt<-timeserieswindow2matrix( mat, msk, c(20,40,60,70) , 6, 0 )
tte<-ttt$eventmatrix
```

```
eanat<-sparseDecom(  tte, ttt$mask4d, sparseness=-0.9, z=0.5,nvecs=2, its=5,cthresh=0, mycoption=1)
eanat2<-sparseDecom( tte,          0, sparseness=-0.9, z=0.5,nvecs=2, its=5,cthresh=0, mycoption=1)
enomask<-eanat2$eigenanatomyimages[,1]
# back to timematrix
tmat<-matrix( enomask,  nrow=6 )
# back to image
eimg<-antsImageClone( msk )
eimg[ msk == 1 ]<-tmat[1,]
# convert image space to evec space
emat<-timeseries2matrix( eanat$eigenanatomyimages[[1]], msk )
# convert emat to events
eavent<-timeserieswindow2matrix( data.matrix(emat) , msk, 1 , 6, 0 )
emask<-eavent$eventmatrix[1,]
##############################
```

`tracts` *tracts*

Description

A data frame label numbers and names for the white matter tracts labels.

Usage

```
data(tracts)
```

Format

A data frame listing the following variables.

`label_num` Numerical label value.

`label_name` Shorthand anatomical value.

References

http://www.ncbi.nlm.nih.gov/pubmed/18407524

Examples

```
data(tracts)
```

usePkg *Use any package. If package is not installed, this will install from CRAN.*

Description

Use any package. If package is not installed, this will install from CRAN.

Usage

```
usePkg(packageName, allowInstall = FALSE)
```

Arguments

packageName	Name of package as *string*.
allowInstall	let the package be installed from CRAN

Value

T if package successfully loaded, F otherwise.

Author(s)

Benjamin M. Kandel, BB Avants

Examples

```
usePkg("randomForest")
```

vwnrfs *voxelwise neighborhood random forest segmentation and prediction*

Description

Represents feature images as a neighborhood and uses the features to build a random forest prediction from an image population

Usage

```
vwnrfs(y, x, labelmask, rad = NA, nsamples = 1, ntrees = 500,
  asFactors = TRUE)
```

Arguments

y	list of training label images, can be a factor or numeric vector this can also be a regular old vector
x	a list of lists where each list contains feature images
labelmask	a mask for the features (all in the same image space) the labelmask defines the number of parallel samples that will be used per subject sample. two labels will double the number of predictors contributed from each feature image.
rad	vector of dimensionality d define nhood radius
nsamples	(per subject to enter training)
ntrees	(for the random forest model)
asFactors	boolean - treat the y entries as factors

Value

list a 4-list with the rf model, training vector, feature matrix and the random mask

Author(s)

Avants BB, Tustison NJ

Examples

```
mask<-makeImage( c(10,10), 0 )
mask[ 3:6, 3:6 ]<-1
mask[ 5, 5:6]<-2
ilist<-list()
lablist<-list()
inds<-1:50
scl<-0.33 # a noise parameter
for ( predtype in c("label","scalar") )
{
for ( i in inds ) {
  img<-antsImageClone(mask)
  imgb<-antsImageClone(mask)
  limg<-antsImageClone(mask)
  if ( predtype == "label") {  # 4 class prediction
    img[ 3:6, 3:6 ]<-rnorm(16)*scl+(i %% 4)+scl*mean(rnorm(1))
    imgb[ 3:6, 3:6 ]<-rnorm(16)*scl+(i %% 4)+scl*mean(rnorm(1))
    limg[ 3:6, 3:6 ]<-(i %% 4)+1  # the label image is constant
    }
    if ( predtype == "scalar") {
      img[ 3:6, 3:6 ]<-rnorm(16,1)*scl*(i)+scl*mean(rnorm(1))
      imgb[ 3:6, 3:6 ]<-rnorm(16,1)*scl*(i)+scl*mean(rnorm(1))
      limg<-i^2.0  # a real outcome
      }
    ilist[[i]]<-list(img,imgb)  # two features
    lablist[[i]]<-limg
  }
  rfm<-vwnrfs( lablist , ilist, mask, rad=c(2,2) )
  if ( predtype == "label" )
    print(  sum( rfm$tv != predict(rfm$rfm) ) )
  if ( predtype == "scalar" )
    print( cor(as.numeric(rfm$tv) , as.numeric(predict(rfm$rfm) ) ) )
} # end predtype loop
```

weingartenImageCurvature *image curvature for 3D*

Description

uses the weingarten map to estimate image mean or gaussian curvature

Usage

```
weingartenImageCurvature(image, sigma = 1, opt = "mean")
```

Arguments

image	antsImage
sigma	smoothing parameter
opt	mean by default, otherwise gaussian or characterize

Value

image

Author(s)

Brian B. Avants

References

Avants, B, J. Gee, and B. Avants. Shape operator for differential image analysis Information Processing in Medical Imaging, 2003.

Examples

```
img = makeImage( c(10,10,10) , rnorm( 1000 ) )
fik <- weingartenImageCurvature( img )
if ( abs( mean(fik) - 128 ) > 1 ) stop("weingartenImageCurvature failure")
## Not run:
fi <- antsImageRead( getANTsRData( "mni" ) )
fik <- weingartenImageCurvature( fi )

## End(Not run)
```

whiten *Simple whitening function.*

Description

Whitens the input matrix using SVD and returns the result.

Usage

```
whiten(x, k = NA, reducex = FALSE)
```

Arguments

x	input matrix
k	rank to use
reducex	reduce the input matrix to k-size subspace

Value

matrix is output

Author(s)

Avants BB

Examples

```
mat <- matrix(rnorm(300),ncol=50)
wmat<-whiten( mat )
wmat2<-whiten( mat, 2, TRUE )
```

[,antsImage,NULL,ANY-method *as.antsImage*

Description

convert types to antsImage

Usage

```
## S4 method for signature antsImage,NULL,ANY
x[i, j, ..., drop = TRUE]

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

## S4 method for signature antsImage,logical,ANY
x[i, j, ..., drop = TRUE]
```

```
## S4 method for signature antsImage,logical,ANY
x[i, j, ..., drop = TRUE]

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

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

## S4 method for signature antsImage,NULL,NULL
x[i, j, k = NA, l = NA, ..., drop]

## S4 method for signature antsImage,NULL,NULL
x[i, j, k = NA, l = NA, ..., drop]

## S4 method for signature antsImage,numeric,numeric
x[i, j, k = NA, l = NA, ..., drop]

## S4 method for signature antsImage,numeric,numeric
x[i, j, k = NA, l = NA, ..., drop]

## S4 method for signature antsImage,numeric,NULL
x[i, j, k = NA, l = NA, ..., drop]

## S4 method for signature antsImage,numeric,NULL
x[i, j, k = NA, l = NA, ..., drop]

## S4 method for signature antsImage,NULL,numeric
x[i, j, k = NA, l = NA, ..., drop]

## S4 method for signature antsImage,NULL,numeric
x[i, j, k = NA, l = NA, ..., drop]

## S4 method for signature antsImage,missing,numeric
x[i, j, k = NA, l = NA, ..., drop]

## S4 method for signature antsImage,numeric,missing
x[i, j, k = NA, l = NA, ..., drop]

## S4 method for signature antsImage,missing,missing
x[i, j, k = NA, l = NA, ..., drop]

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

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

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

## S4 replacement method for signature antsImage,matrix,ANY,ANY
```

```
x[i, j, ...] <- value

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

## S4 replacement method for signature antsImage,NULL,antsRegion,ANY
x[i, j, ...] <- value

## S4 replacement method for signature antsImage,logical,antsRegion,ANY
x[i, j, ...] <- value

## S4 replacement method for signature antsImage,array,antsRegion,ANY
x[i, j, ...] <- value

## S4 replacement method for signature antsImage,matrix,antsRegion,ANY
x[i, j, ...] <- value

## S4 replacement method for signature antsImage,NULL,NULL,numeric
x[i, j, ...] <- value

## S4 replacement method for signature antsImage,numeric,numeric,numeric
x[i, j, ...] <- value

## S4 replacement method for signature antsImage,numeric,NULL,numeric
x[i, j, ...] <- value

## S4 replacement method for signature antsImage,NULL,numeric,numeric
x[i, j, ...] <- value

as.antsImage(object, ...)

## S4 method for signature matrix
as.antsImage(object, pixeltype = "float",
  spacing = as.numeric(seq.int(from = 1, by = 0, length.out =
  length(dim(object)))), origin = as.numeric(seq.int(from = 0, by = 0,
  length.out = length(dim(object)))))

## S4 method for signature array
as.antsImage(object, pixeltype = "float",
  spacing = as.numeric(seq.int(from = 1, by = 0, length.out =
  length(dim(object)))), origin = as.numeric(seq.int(from = 0, by = 0,
  length.out = length(dim(object)))))
```

Arguments

x	antsImage
i	logical or i-th dimension
j	not used or j-th dimension
...	Extra named arguments passed to FUN
drop	method for missing data
k	not used or k-th dimension

l	not used or l-th dimension
value	ok
object	An object
pixeltype	a character string e.g. "float"
spacing	numeric vector matching image dimensionality e.g. c(1.2,1.2)
origin	numeric vector matching image dimensionality e.g. c(0,0)

Methods (by class)

- x = antsImage,i = NULL,j = ANY:
- x = antsImage,i = NULL,j = ANY:
- x = antsImage,i = logical,j = ANY:
- x = antsImage,i = logical,j = ANY:
- x = antsImage,i = ANY,j = ANY:
- x = antsImage,i = ANY,j = ANY:
- x = antsImage,i = NULL,j = NULL:
- x = antsImage,i = NULL,j = NULL:
- x = antsImage,i = numeric,j = numeric:
- x = antsImage,i = numeric,j = numeric:
- x = antsImage,i = numeric,j = NULL:
- x = antsImage,i = numeric,j = NULL:
- x = antsImage,i = NULL,j = numeric:
- x = antsImage,i = NULL,j = numeric:
- x = antsImage,i = missing,j = numeric:
- x = antsImage,i = numeric,j = missing:
- x = antsImage,i = missing,j = missing:
- x = antsImage,i = NULL,j = ANY,value = ANY:
- x = antsImage,i = logical,j = ANY,value = ANY:
- x = antsImage,i = array,j = ANY,value = ANY:
- x = antsImage,i = matrix,j = ANY,value = ANY:
- x = antsImage,i = list,j = ANY,value = ANY:
- x = antsImage,i = NULL,j = antsRegion,value = ANY:
- x = antsImage,i = logical,j = antsRegion,value = ANY:
- x = antsImage,i = array,j = antsRegion,value = ANY:
- x = antsImage,i = matrix,j = antsRegion,value = ANY:
- x = antsImage,i = NULL,j = NULL,value = numeric:
- x = antsImage,i = numeric,j = numeric,value = numeric:
- x = antsImage,i = numeric,j = NULL,value = numeric:
- x = antsImage,i = NULL,j = numeric,value = numeric:
- matrix:
- array:

`%>%` *Pipe an object forward*

Description

Chain operators together

Usage

```
lhs %>% rhs
```

Arguments

`lhs`	input from left side
`rhs`	additional params

Details

The `%>>%` operator pipes the object on the left-hand side to the right-hand side according to the syntax.

Index