

Package: ANTsR (via r-universe)

October 29, 2026

Type Package

Title ANTs in R: Quantification Tools for Biomedical Images

Version 0.6.7

Date 2026-05-08

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

Description Quantification tools for biomedical images that interface 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. Specialized functionality in ANTsR includes image segmentation and registration along imaging specific variations of principal component and canonical correlation analysis.

License Apache License (== 2.0) | file LICENSE

LazyLoad yes

Depends R (>= 3.5.0), methods

Imports ANTsRCore (>= 0.8.1), dplyr, ggplot2, graphics, grDevices, magrittr, purrr, reshape2, reticulate, stats, stringr, tibble, utils, rsvd, GPArotation, glmnet

Suggests colormap, magic, abind, BGLR, caret, cluster, corpcor, e1071, fastICA, fpc, GGally, glasso, ggpubr, ggdendro, grid, igraph, irlba, knitr, MASS, Matrix, mFilter, moments, patchwork, pixmap, png, pbapply, pROC, pracma, randomForest, RcppEigen, RcppHNSW, rgl, rmarkdown, robust, robustbase, signal, sna, testthat, tools, torch, tsne, viridis, xgboost, FNN, gridExtra, hdf5r, salad, DiagrammeR, PMA, RGCCA, elasticnet, ggrepel, glue, psych, scales, sparsepca, tidyr, DT, clue, progress, forcats

LazyData TRUE

Remotes ANTsX/ANTsRCore, ANTsX/ITKR

VignetteBuilder knitr

Encoding UTF-8

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.3

Config/pak/sysreqs cmake git make libicu-dev libpng-dev python3 libnode-dev

Repository <https://antsx.r-universe.dev>

Date/Publication 2026-10-29 18:17:29 UTC

RemoteUrl <https://github.com/ANTsX/ANTsR>

RemoteRef HEAD

RemoteSha 059f86fcc1687a2118faeb7a56116c45465ba0d9

Contents

.int_antsProcessArguments	13
.motion_correction	13
[<-,antsImage,NULL,ANY,ANY-method	14
%>%	18
aal	18
abpBrainExtraction	19
abpN4	21
addNoiseToImage	22
adjusted_rvcoef	23
affineInitializer	23
alffmap	24
antsApplyTransforms	26
antsApplyTransformsToPoints	28
antsAverageImages	30
antsBOLDNetworkAnalysis	31
antsCompileFlags	33
antsCopyImageInfo	33
antsCopyImageInfo2	34
antsDir	34
antsImage-class	35
antsImage-math	36
antsImage-operators	39
antsImage-summary	47
antsImageClone	49
antsImageGetSet	49
antsImageHeaderInfo	51
antsImageIterator	52
antsImageIterator-class	52
antsImageIteratorGet	53
antsImageIteratorGetIndex	54
antsImageIteratorGoToBegin	54
antsImageIteratorGoToReverseBegin	55

antsImageIteratorIsAtEnd	55
antsImageIteratorIsAtReverseEnd	56
antsImageIteratorNext	57
antsImageIteratorPrevious	57
antsImageIteratorRemaining	58
antsImageIteratorSet	58
antsImageIteratorSetIndex	59
antsImageMutualInformation	60
antsImagePhysicalSpaceConsistency	61
antsImageRead	62
antsImageTypeCast	63
antsImageWrite	64
antsIncludes	65
antsLibs	66
antsMatrix-class	66
antsMotionCalculation	67
antsMotionCorr	68
antspymm_nuisance_names	69
antspymm_predictors	69
antspymm_qc_names	70
antspymm_vartype	70
antsr_mcca_features	71
antsr_pca_features	72
antsr_random_features	73
antsr_rgcca_baseline	73
antsr_rgcca_sparse	74
antsr_scca_features	75
antsRegion-class	76
antsRegistration	76
antsrGetPointerName	81
antsrimpute	82
antsrMetric-class	83
antsrMetricCreate	83
antsrMetricGetDerivative	85
antsrMetricGetValue	85
antsrMetricInitialize	86
antsrMetricSetFixedImage	87
antsrMetricSetFixedImageMask	87
antsrMetricSetFixedTransform	88
antsrMetricSetMovingImage	89
antsrMetricSetMovingImageMask	89
antsrMetricSetMovingTransform	90
antsrMetricSetNumberOfHistogramBins	91
antsrMetricSetRadius	91
antsrMetricSetSampling	92
antsrMotionCalculation	92
antsrSurf	94
antsrTransform-class	96

antsrTransformFromDisplacementField	97
antsrVersions	98
antsrVol	98
antsSetPixels	100
antsSpatialICAfMRI	100
antsTransformIndexToPhysicalPoint	102
antsTransformPhysicalPointToIndex	102
antsVersions	103
apply	104
apply_simlr_matrices	105
apply_simlr_matrices_dtfix	106
apply_transform_matrix	107
applyAntsrTransform	107
applyAntsrTransformToImage	108
applyAntsrTransformToPoint	109
applyAntsrTransformToVector	109
as.antsImage	110
as.antsMatrix	111
as.array	112
aslAveraging	114
aslCensoring	116
aslDenoiseR	118
aslPerfusion	120
atropos	122
ba_svd	124
backup_simlr	125
basal_forebrain	127
basicInPaint	127
bayesianCBF	128
bayesianlm	130
bigLMStats	132
blockStimulus	134
bloodPerfusionSVD	135
bold_correlation_matrix	136
boxplot.antsImage	137
buildTemplate	138
calculate_simlr_energy	139
calculate_simlr_gradient	140
cerebellum	141
cerebellum_labels	141
check_ants-methods	141
check_zero_variance	142
cit168_brainstem	143
cit168_reinf_learn	143
CIT168_Reinf_Learn_v1_label_descriptions_pad	143
CIT168_T1w_700um_pad_adni_brainstem	144
clip_gradient_by_quantile	144
clip_gradient_norm	145

clusterTimeSeries	146
coerce_mask	147
combineNuisancePredictors	147
compcor	149
composeAntsrTransforms	150
composeDisplacementFields	151
composeTransformsToField	151
compute_communalities	152
compute_variance_explained_per_factor	153
computeDVARs	154
computeDVARsSpatialMap	154
construct_sim_names	155
convolveImage	156
corw	156
create_optimizer	157
createAntsrTransform	158
createJacobianDeterminantImage	159
createWarpedGrid	160
cropImage	161
cropImageCenter	162
cropImageFromCenterPoint	162
cropIndices	163
crossvalidatedR2	164
cut.antsImage	165
cvEigenanatomy	165
dataAugmentation	167
decode_pymm_names	169
deconvolutionSVD	170
decropImage	171
denoiseImage	171
density.antsImage	172
DesikanKillianyTourville	173
dice_overlap_soft_matrix	174
dice_overlap_soft_vector	174
displacementFieldFromAntsrTransform	175
divide_by_column_sum	176
dkt	176
dkt_cit_labels	176
dkt_cortex_cit_deep_brain	177
dkt_labels	177
dlbs	177
eanatDef	178
eanatSelect	180
eigSeg	181
ensembled_sparsity	182
error_not_antsImage	183
estimate_joint_rank	183
estimate_learning_rate_nsa	184

estimate_rank_by_permutation_rv	185
estSmooth	186
euler	188
exemplarInpainting	189
exploratory_visualization	190
extend_simlr_embedding_with_new_modalities	191
extractImagePatchCoordinates	193
extractImagePatches	194
extractSlice	195
FA_JHU_labels_edited	196
fast_whiten	196
fastMarchingExtension	197
filterfMRIforNetworkAnalysis	198
fitBsplineDisplacementField	200
fitBsplineObjectToScatteredData	202
fitThinPlateSplineDisplacementField	204
fitTimeVaryingTransformToPointSets	205
fitTransformToPairedPoints	207
fMRINormalization	209
frequencyFilterfMRI	212
frob	213
fsl2antsrTransform	214
functionalLungSegmentation	214
fusedRidge	216
fusedRidgeDirect	218
fusedRidgeTorch	220
fuzzySpatialCMeansSegmentation	222
generate_structured_multiview_data	223
generateAifMaskImage	224
geoSeg	225
getANTsRData	226
getAntsrTransformFixedParameters	227
getAntsrTransformParameters	228
getASLNoisePredictors	228
getAverageOfTimeSeries	229
getCenterOfMass	230
getCentroids	231
getfMRIInuisanceVariables	232
getMask	233
getMultiResFeatureMatrix	234
getMultivariateTemplateCoordinates	235
getNamesFromDataframe	236
getNeighborhoodAtVoxel	237
getNeighborhoodInMask	238
getPixels	240
getTemplateCoordinates	241
glm_impute	242
gradient_invariant_orthogonality_defect	243

gradient_invariant_orthogonality_salad	244
hausdorffDistance	245
hemisphere	245
hemodynamicRF	246
hessianObjectness	249
hist.antsImage	250
histogramEqualizeImage	251
histogramMatchImage	251
histogramMatchImage2	252
histogramWarpImageIntensities	253
humanize_anatomical_name	254
iBind	255
icawhiten	255
ilr	256
ilr.predict	257
image2ClusterImages	258
imageDomainToSpatialMatrix	259
imageFileNames2ImageList	260
imageListToMatrix	260
imageSimilarity	262
imagesToMatrix	263
iMath	264
iMathOps	265
indicator_opt_both_ways	266
initializeEigenanatomy	266
initializeSimlr	268
integrateVectorField	269
integrateVelocityField	270
interleaveMatrixWithItself	271
interpret_simlr_vector	272
interpret_simlr_vector2	273
inv_sqrt_sym	274
inv_sqrt_sym_newton	275
invariant_orthogonality_defect	275
invertAntsrTransform	276
invertDisplacementField	276
is.antsImage	277
jhu_fa_labels	278
jhu_md_labels	278
jlfProp	278
joinEigenanatomy	280
jointLabelFusion	281
jointSmoothMatrixReconstruction	284
kellyKapowski	287
kmeansSegmentation	289
knnSmoothImage	290
knnSmoothingMatrix	291
l1_normalize_features	292

labelClusters	293
labelGeometryMeasures	294
labelImageCentroids	295
labelImageRegistration	296
labelOverlapMeasures	298
labels2matrix	299
labelStats	300
lappend	300
list_simlr_optimizers	301
load.ANTsR	302
lobes	302
lobes_brainstem	303
localGyrificationIndex	303
localJointLabelFusion	304
lowrankRowMatrix	306
make3ViewPNG	306
makeGraph	307
makeImage	308
makePointsImage	309
makePowersPointsImage	310
mapAsymVar	311
mapLRAverageVar	312
mask_values	312
maskImage	313
matrix2timeseries	314
matrixToImages	315
MD_JHU_labels_edited	315
mean.antsImage	316
median.antsImage	317
mergeChannels	317
mergeListToNDImage	318
mild	319
milr	321
milr.predict	323
mni2tal	325
morphology	326
mrnrfs	327
mrnrfs.predict	329
mtl_description	330
mtl_labels	331
multichannelPCA	331
multichannelToVector	332
multigrep	333
multiLabelMorphology	334
multiResRandomForestRegression	335
multiscaleSVD	338
multiview_pca	339
n3BiasFieldCorrection	341

n3BiasFieldCorrection2	341
n4BiasFieldCorrection	343
nbm3_labels	345
nbm3CH13	345
neg_violation	346
networkEiganat	346
NNHEmbed	349
NNHEmbed_update_residuals	351
nsa_flow	352
nsa_flow_autograd	354
nsa_flow_fa_diagram	355
nsa_flow_flowchart	356
nsa_flow_pca	357
nsa_flow_pca_fa	358
nsa_flow_pca_fa_2	359
nsa_flow_retract	361
nsa_flow_retract_auto	362
nsa_flow_torch	362
nsa_flow_torch_ag	365
oneHotToSegmentation	367
optimal_simlr_initializer	368
optimize_indicator_matrix	369
orth_residual	370
orthogonalize_feature_space	371
orthogonalizeAndQSparsify	372
orthogonalizeAndQSparsifyOld	373
padImageByFactor	374
padOrCropImageToSize	374
pairwise_matrix_similarity	375
pairwiseImageDistanceMatrix	376
partialVolumeCorrection	377
perfusionregression	378
petSUVR	379
pixeltype	380
plot.antsImage	382
plot_energy_decomposition	385
plot_features	386
plotBasicNetwork	387
plotPrettyGraph	390
posthoc_fa_summary	391
powers_areal_mni_itk	392
powers_mni_itk	393
ppmi_template_500Parcels_Yeo2011_17Networks_2023_homotopic	394
ppmi_yeo_labels	394
predict.fusedRidge	394
predict.fusedRidgeDirect	395
predict.fusedRidgeTorch	396
predictSimlr	397

preprocessfMRI	398
preprocessRestingBOLD	400
priorBasedSegmentation	402
project_to_orthonormal_nonnegative	403
project_to_partially_orthonormal_nonnegative	403
quantifyCBF	405
quantifySNPs	406
quantile.antsImage	407
randcca	408
randomlyTransformImageData	409
randomMask	411
randsvd	411
randwhiten	412
rankBasedMatrixSegmentation	413
rankIntensity	414
rapidlyInspectImageData	415
read_simlr	416
read_simlr_data_frames	416
readAntsrTransform	417
readNormalizedPopulationData	418
reconstructImageFromPatches	419
refbasis_brain	420
refbasis_head	420
reference_basis	420
reflectImage	420
reflectionMatrix	421
regressionMatchImage	422
regressProjections	423
regularizeSimlr	425
reho	426
renormalizeProbabilityImages	427
reorientImage	428
resampleImage	428
resampleImageToTarget	429
resels	431
reset_optimizer_state	432
rfSegmentation	432
rfSegmentationPredict	434
rftPval	435
rftResults	436
ri	439
ripmmarc	440
ripmmarcBasisImage	442
ripmmarcPop	443
robustMatrixTransform	444
rsfDenoise	445
rvcoef	447
rvcoef_components	448

rvcoef_gram_impl	448
rvcoef_trace_impl	449
safe_pca	449
save.ANTsR	450
scaleSpaceFeatureDetection	451
sd	452
segmentationRefinement.predict	453
segmentationRefinement.train	455
segmentationToOneHot	458
segmentShapeFromImage	459
select_joint_k	460
setANTsDeterministic	461
setAntsrTransformFixedParameters	462
setAntsrTransformParameters	462
setSeedBasedOnTime	463
shorten_pymm_names	463
simlr	464
simlr.parameters	467
simlr.perm	469
simlr.search	471
simlr_feature_orth	472
simlr_impute	473
simlr_path_models	474
simlr_sparseness	474
simlrU	475
simulateBiasField	477
simulateDisplacementField	478
skeletonize	479
smoothAppGradCCA	480
smoothImage	481
smoothMatrixPrediction	482
smoothMultiRegression	484
smoothRegression	485
softwareVersions	487
sparseDecom	487
sparseDecom2	490
sparseDecom2boot	493
sparseDecomboot	496
sparseDistanceMatrix	498
sparseDistanceMatrixXY	499
sparseRegression	501
sparsify_by_column_winner	503
spatialbayesianlm	504
splitChannels	505
splitData	506
splitMask	507
splitNDImageToList	508
statFieldThresh	508

step	510
subgradientL1Regression	511
subjectDataToGroupDataFrame	512
subset.antsImage	513
sum_preserving_matrix_partition	513
symm	514
T_template0_LobesBstem	515
take_abs_unsigned	515
taskFMRI	516
temporalwhiten	517
testEanat	518
thresholdImage	519
timeseries2matrix	520
timeseriesN3	520
timeserieswindow2matrix	521
tissues	523
tracts	523
transform_matrix	524
transform_prior_rows_to_uncorrelated	525
trapz	526
undo_transform_matrix	527
unique.antsImage	527
usePkg	528
var	528
vector_to_df	529
vectorToMultichannel	530
visualize_lowrank_relationships	531
visualize_permutation_test	531
vwnrfs	532
vwnrfs.predict	534
weingartenImageCurvature	535
whiten	536
whiten_matrix	537
wm_major_tracts	538
wmh_evidence	538
write_simlr	539
write_simlr_data_frames	539
writeAntsrTransform	540
writeNormalizedPopulationData	540

`.int_antsProcessArguments`*Processing arguments for command line parsing*

Description

Processing arguments for command line parsing

Usage

```
.int_antsProcessArguments(args)
```

Arguments

<code>args</code>	arguments to parse into those for ANTsR functions
-------------------	---

Value

A character vector `args = "hey"` `.int_antsProcessArguments(args)` `args = list(h = "hey")` `.int_antsProcessArguments(args)`
`args = list(h = list("hey"))` `.int_antsProcessArguments(args)`

`.motion_correction`*Simple motion_correction function.*

Description

motion corrects 4D time series imaging data

Usage

```
.motion_correction(  
  img,  
  fixed = NULL,  
  moreaccurate = 1,  
  txtype = "Affine",  
  verbose = FALSE  
)
```

Arguments

<code>img</code>	4D antsImage
<code>fixed</code>	target fixed image
<code>moreaccurate</code>	0, 1 or 2 with higher values being more accurate (use 2 for real applications, 0 for testing)
<code>txtype</code>	Transformation type
<code>verbose</code>	verbosity boolean

Value

list of outputs

Note

This function may give different results on multiple runs.

Author(s)

Avants BB

Examples

```
testing <- makeImage(c(10, 10, 10, 5), rnorm(5000))
testing <- iMath(testing, "PadImage", 5)
mocorr <- .motion_correction(testing)
aimg_to_array <- function(x) {
  if (is.antsImage(x)) {
    x <- as.array(x)
  }
  x
}
amocorr <- lapply(mocorr, aimg_to_array)
mocorr2 <- .motion_correction(testing)
amocorr2 <- lapply(mocorr2, aimg_to_array)

# This function may give different results on multiple runs
# without setting the seed
```

[<-,antsImage,NULL,ANY,ANY-method

Extract or Replace Parts of antsImage Object

Description

Operators acting on antsImage objects to extract or replace parts.

Usage

```
## 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,antsImage,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

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

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

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

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

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

```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```

Arguments

x antsImage

i	logical or i-th dimension
j	not used or j-th dimension
...	not used
value	value to assign to the subset in assignment operator
drop	method for missing data
k	not used or k-th dimension
l	not used or l-th dimension

Examples

```

fi <- antsImageRead(getANTsRData("r16"))
l <- prod(dim(fi))
sub <- sample(c(TRUE, FALSE), size = 1, replace = TRUE)
fi[sub] <- rnorm(sum(sub))
fi[, 1] <- array(10, dim = dim(fi[, 1]))

fi <- antsImageRead(getANTsRData("ch2"))
fi[2, 1, 2]
fi[2, 1, 2] <- 4
fi[, 1, 2]
fi[, 1, 2] <- array(10, dim = dim(fi[, 1, 2]))
fi[2, , 2]
fi[2, , 2] <- array(10, dim = dim(fi[2, , 2]))
fi[2, 1, ]
# fi[2, 1, ] = array(10, dim = dim(fi[2, 1, ]))
fi[2, , ]
fi[, 1, ]
fi[, , 2]
# fi[, , 2] = array(10, dim = dim(fi[, , 2]))

l <- prod(dim(fi))
sub <- sample(c(TRUE, FALSE), size = 1, replace = TRUE)
sub <- array(sub, dim = dim(fi))
fi[sub] <- rnorm(sum(sub))
fi <- antsImageRead(getANTsRData("r16"))
l <- prod(dim(fi))
sub <- sample(c(TRUE, FALSE), size = 1, replace = TRUE)
sub <- array(sub, dim = dim(fi))
fi[sub] <- rnorm(sum(sub))
fi <- antsImageRead(getANTsRData("r16"))
fi[fi > 200] <- 200
fi <- antsImageRead(getANTsRData("r16"))
i <- sample(seq(nrow(fi)), 5)
i <- as.numeric(i)
j <- i
fi[i, j] <- 26.2
fi <- antsImageRead(getANTsRData("r16"))
fi[1, ] <- array(10, dim = dim(fi[1, ]))
fi <- antsImageRead(getANTsRData("r16"))
fi[, 1] <- array(10, dim = dim(fi[, 1]))

```

```
fi <- antsImageRead(getANTsRData("r16"))
fi[, 1]
fi[, 2:5]
fi <- antsImageRead(getANTsRData("r16"))
fi[fi > 45]
fi <- antsImageRead(getANTsRData("r16"))
arr <- c(as.array(fi) > 45)
fi[arr]
fi[1:10, 2:14]
fi[1:10 * 1.0, 2:14 * 1.0]
fi <- antsImageRead(getANTsRData("ch2"))
fi[, , 2]
```

%>%	<i>Pipe an object forward</i>
-----	-------------------------------

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.

aal	<i>AAL Labels</i>
-----	-------------------

Description

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

Usage

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

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

Examples

```
data(aal)
```

abpBrainExtraction	<i>An ants-based brain extraction script.</i>
--------------------	---

Description

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

Usage

```
abpBrainExtraction(
  img,
  tem,
  temmask,
  temregmask = NULL,
  regtype = "SyN",
  tdir = NA,
  pad = 0,
  verbose = FALSE
)
```

Arguments

img	whole head image to which we map a brain mask
tem	Template image (the whole head) which has an associated label mask.
temmask	Template's antsImage brain mask.
temregmask	Template's registration mask including skull but not the face
regtype	registration type: 'SyN' (fast, default), 'SyNabp' (better, slower)
tdir	temporary directory (optional)
pad	argument passed to iMath for how much to zero-pad the image
verbose	print diagnostic messages

Value

outputs a brain image and brain mask.

Author(s)

Tustison N, Avants BB

Examples

```
n <- 64
fn <- getANTsRData("r16")
img <- antsImageRead(fn)
img <- resampleImage(img, c(n, n), 1, 0)
tf <- getANTsRData("r27")
tem <- antsImageRead(tf)
tem <- resampleImage(tem, c(n, n), 1, 0)
temmask <- antsImageClone(tem)
temmask[tem > 20] <- 1
temmask[tem <= 20] <- 0
## Not run:
bm <- abpBrainExtraction(img = img, tem = tem, temmask = temmask)

## End(Not run)
```

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,
  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
...	additional arguments to pass to n4BiasFieldCorrection

Value

outputs a bias corrected image. 1 indicates failure.

Author(s)

Tustison N, Avants BB

Examples

```
img <- antsImageRead(getANTsRData("r16"))
img2 <- abpN4(img)
img3 <- abpN4(img)
testthat::expect_equal(mean(img2), 82.6, tolerance = 1e-2)
testthat::expect_equal(img2, img3)
```

addNoiseToImage	<i>addNoiseToImage</i>
-----------------	------------------------

Description

Add noise to image using additive Gaussian, salt-and-pepper, shot, or speckle noise.

Usage

```
addNoiseToImage(  
  image,  
  noiseModel = c("additivegaussian", "saltandpepper", "shot", "speckle"),  
  noiseParameters  
)
```

Arguments

image	input image
noiseModel	either "additivegaussian", "saltandpepper", "shot", or "speckle".
noiseParameters	vector defining the noise models. additivegaussian: (mean, standardDeviation), saltandpepper: (probability, saltValue, pepperValue), shot: (scale), speckle: (standardDeviation),

Value

noise corrupted image.

Author(s)

NJ Tustison

Examples

```
image <- antsImageRead(getANTsRData("r16"))  
noiseImage <- addNoiseToImage(image, "additivegaussian", c(0, 1))  
noiseImage <- addNoiseToImage(image, "saltandpepper", c(0.1, 0, 100))  
noiseImage <- addNoiseToImage(image, "shot", c(1.0))  
noiseImage <- addNoiseToImage(image, "speckle", c(1.0))
```

adjusted_rvcoef	<i>Computes the Adjusted RV-Coefficient (Highly Optimized)</i>
-----------------	--

Description

This function avoids all redundant computations by getting all necessary components from a single call to an internal dispatcher.

Usage

```
adjusted_rvcoef(X, Y)
```

Arguments

X	A numeric matrix (n observations, p variables).
Y	A numeric matrix (n observations, q variables).

Value

A single scalar value for the adjusted RV-coefficient.

affineInitializer	<i>a multi-start optimizer for affine registration</i>
-------------------	--

Description

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

Usage

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

Arguments

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

Value

`transformationMatrix`

Note

See <https://github.com/ANTsX/ANTs/issues/444> for reproducibility discussion

Author(s)

Avants BB

Examples

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

mi2 <- resampleImage(mi, c(1.25, 1.25))
tx <- affineInitializer(fi, mi2)
tx2 <- affineInitializer(fi, mi2)
```

alffmap

alffmap

Description

Amplitude of Low Frequency Fluctuations (ALFF; Zang et al., 2007) and fractional Amplitude of Low Frequency Fluctuations (fALFF; Zou et al., 2008) are related measures that quantify the amplitude of low frequency oscillations (LFOs). This function outputs ALFF and fALFF for the input.

Usage

```
alffmap(  
  x,  
  flo = 0.01,  
  fhi = 0.1,  
  tr = 1,  
  detrend = TRUE,  
  takesqrt = FALSE,  
  kernel  
)
```

Arguments

x	input vector for the time series of interest
flo	low frequency, typically 0.01
fhi	high frequency, typically 0.1
tr	the period associated with the vector x (inverse of frequency)
detrend	detrend the input time series
takesqrt	take the sqrt of the computed values
kernel	smoothing kernel in estimate, see spec.pgram

Value

vector is output showing ALFF and fALFF values

Author(s)

Avants BB

Examples

```
set.seed(1)  
mat <- matrix(rnorm(3000), ncol = 50)  
fallf <- apply(mat, FUN = alffmap, MARGIN = 2)  
k <- kernel("daniell", rep(2, 3))  
fallf2 <- apply(mat, FUN = alffmap, MARGIN = 2, kernel = k)  
testthat::expect_equal(mean(fallf2), 3.00367017972746)
```

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 = c("linear", "nearestNeighbor", "multiLabel", "gaussian", "bSpline",
    "cosineWindowedSinc", "welchWindowedSinc", "hammingWindowedSinc",
    "lanczosWindowedSinc", "genericLabel"),
  imagetype = 0,
  whichtoinvert = NA,
  compose = NA,
  verbose = FALSE,
  ...
)
```

Arguments

fixed	fixed image defining domain into which the moving image is transformed.
moving	moving image to be mapped to fixed space.
transformlist	character vector of transforms generated by antsRegistration where each transform is a filename.
interpolator	Choice of interpolator. Supports partial matching. linear nearestNeighbor multiLabel for label images but genericlabel is preferred gaussian bSpline cosineWindowedSinc welchWindowedSinc hammingWindowedSinc lanczosWindowedSinc

	genericLabel use this for label images
imagetype	choose 0/1/2/3 mapping to scalar/vector/tensor/time-series
whichtoinvert	optional list of booleans, same length as transforms. whichtoinvert[i] is TRUE if transformlist[i] is a matrix, and the matrix should be inverted. If transformlist[i] is a warp field, whichtoinvert[i] must be FALSE. If the transform list is a matrix followed by a warp field, whichtoinvert defaults to c(TRUE,FALSE). Otherwise it defaults to rep(FALSE, length(transformlist)).
compose	if it is a character string pointing to a valid file location, this will force the function to return a composite transformation filename.
verbose	print command and run verbose application of transform.
...	extra parameters

Value

an antsImage or transformation filename 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 <- ri(1)
moving <- ri(2)
fixed <- resampleImage(fixed, c(64, 64), 1, 0)
moving <- resampleImage(moving, c(68, 68), 1, 0)
mytx <- antsRegistration(
  fixed = fixed, moving = moving,
  typeofTransform = "SyN", verbose = TRUE, printArgs = TRUE
)
mywarpedimage <- antsApplyTransforms(
  fixed = fixed, moving = moving,
  transformlist = mytx$fdwtransforms
)
testthat::expect_true(antsImagePhysicalSpaceConsistency(mywarpedimage, fixed))
invwarped_image <- antsApplyTransforms(
  fixed = moving, moving = fixed,
  transformlist = mytx$invtransforms
)
testthat::expect_true(antsImagePhysicalSpaceConsistency(invwarped_image, moving))
# full access via listing the inputs in standard ANTs format
```

```

res1 <- antsApplyTransforms(
  fixed = fixed, moving = moving,
  transformlist = mytx$fwdtransforms[2],
  whichtoinvert = 1, verbose = TRUE
)
cfile <- antsApplyTransforms(
  fixed = fixed, moving = moving,
  transformlist = mytx$fwdtransforms,
  compose = tempfile()
)
cimg <- antsImageRead(cfile)
cout <- antsApplyTransforms(
  fixed = fixed, moving = moving,
  transformlist = cimg
)
testthat::expect_error(
  antsApplyTransforms(
    fixed = fixed, moving = moving,
    transformlist = cimg, whichtoinvert = 1
  ), "nnot invert transform"
)
testthat::expect_error(
  antsApplyTransforms(
    fixed = fixed, moving = moving,
    transformlist = cimg, whichtoinvert = c(1, 2)
  ), "same length"
)
testthat::expect_error(antsApplyTransforms(
  fixed = fixed, moving = moving,
  transformlist = ""
))

```

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"), verbose = FALSE
)
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$fdwtransforms
)
wptsi <- antsApplyTransformsToPoints(
  dim = 2, points = wpts,
  transformlist = mytx$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	<i>Computes average of image list</i>
-------------------	---------------------------------------

Description

Calculate the mean of a list of antsImages

Usage

```
antsAverageImages(imageList, normalize = FALSE, weights, verbose = TRUE)
```

Arguments

imageList	list of antsImages or a character vector of filenames
normalize	boolean determines if image is divided by mean before averaging
weights	a vector of weights of length equal to imageList
verbose	print a progress bar

Author(s)

Avants BB, Pustina D

Examples

```

f1 <- getANTsRData("r16")
f2 <- getANTsRData("r64")
r16 <- antsImageRead(f1)
r64 <- antsImageRead(f2)
mylist <- list(r16, r64)
antsAverageImages(mylist)
bad <- resampleImage(r16, c(2, 2), useVoxels = FALSE)
antsAverageImages(mylist, normalize = TRUE)
antsAverageImages(c(f1, f2))
testthat::expect_error(antsAverageImages(list(f1, "")))
testthat::expect_error(antsAverageImages(list(r64, bad)))
diff_size <- as.antsImage(r64[1:10, 1:10], ref = r64)
testthat::expect_error(antsAverageImages(list(r64, diff_size)))

```

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 = NULL,
  mask = NULL,
  labels = NULL,
  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)
warpedParcellation <- maskImage(warpedParcellation, img.mask = mask)
old <- NA
labels <- warpedParcellation
gdens <- 0.2
threshLo <- 1
threshHi <- 90
freqLo <- 0.01
freqHi <- 0.1
winsortrim <- 0.02
result <- antsBOLDNetworkAnalysis(bold = bold, mask = mask, warpedParcellation)

## End(Not run)
```

antsCompileFlags	<i>return ants compile flags</i>
------------------	----------------------------------

Description

call this to get the compilation flags used for ants and dependent software

Usage

```
antsCompileFlags()
```

Author(s)

Avants BB

Examples

```
antsCompileFlags()
```

antsCopyImageInfo	<i>Copy header info</i>
-------------------	-------------------------

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)
testthat::expect_error(antsCopyImageInfo(img, 1))
```

antsCopyImageInfo2 *Copy header info with different input order*

Description

Copy origin, direction, and spacing from one antsImage to another. Opposite order of input relative to antsCopyImageInfo

Usage

```
antsCopyImageInfo2(target, reference)
```

Arguments

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

Value

Target image with reference header information.

Examples

```
img <- makeImage(c(10, 10), rnorm(100))
img2 <- makeImage(c(10, 10), rnorm(100))
img2 <- antsCopyImageInfo(img2, img)
testthat::expect_error(antsCopyImageInfo(img, 1))
```

antsDir *return ANTs binary directory information*

Description

call this to get the binary paths to ANTs executables.

Usage

```
antsDir()
```

Author(s)

Avants BB

Examples

```
antsDir()
```

antsImage-class	<i>An S4 class for an antsImage</i>
-----------------	-------------------------------------

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,
  isVector = FALSE,
  filename = ""
)
```

Arguments

object	input object to convert
.Object	input object to convert
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
isVector	logical indicator of the image is a vector
filename	character filename if the data was read in, otherwise ""

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
isVector	logical indicator of the image is a vector
filename	character filename if the data was read in, otherwise ""

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
 isVector boolean
 filename character filename if the data was read in, otherwise ""

antsImage-math *Math for antsImage Objects*

Description

Overloaded math for antsImage objects

Usage

```

## S4 method for signature 'antsImage'
Math(x)

## S4 method for signature 'antsImage'
abs(x)

## S4 method for signature 'antsImage'
sign(x)

## S4 method for signature 'antsImage'
sqrt(x)

## S4 method for signature 'antsImage'
ceiling(x)

## S4 method for signature 'antsImage'
round(x)

## S4 method for signature 'antsImage'
floor(x)

## S4 method for signature 'antsImage'
trunc(x)

## S4 method for signature 'antsImage'
log(x, base = exp(1))

## S4 method for signature 'antsImage'
log10(x)

```

```
## S4 method for signature 'antsImage'
log2(x)

## S4 method for signature 'antsImage'
acos(x)

## S4 method for signature 'antsImage'
asin(x)

## S4 method for signature 'antsImage'
atan(x)

## S4 method for signature 'antsImage'
acosh(x)

## S4 method for signature 'antsImage'
asinh(x)

## S4 method for signature 'antsImage'
atanh(x)

## S4 method for signature 'antsImage'
cos(x)

## S4 method for signature 'antsImage'
sin(x)

## S4 method for signature 'antsImage'
tan(x)

## S4 method for signature 'antsImage'
cosh(x)

## S4 method for signature 'antsImage'
sinh(x)

## S4 method for signature 'antsImage'
tanh(x)

## S4 method for signature 'antsImage'
cospi(x)

## S4 method for signature 'antsImage'
sinpi(x)

## S4 method for signature 'antsImage'
tanpi(x)
```

```
## S4 method for signature 'antsImage'
exp(x)

## S4 method for signature 'antsImage'
gamma(x)

## S4 method for signature 'antsImage'
lgamma(x)

## S4 method for signature 'antsImage'
!x
```

Arguments

x	is an object of class antsImage.
base	a positive or complex number: the base with respect to which logarithms are computed. Defaults to e=exp(1).

Examples

```
img01 <- as.antsImage(array(1:64, c(4, 4, 4, 1)))
testthat::expect_true(is.antsImage(abs(img01)))
testthat::expect_true(is.antsImage(sign(img01)))
testthat::expect_true(is.antsImage(sqrt(img01)))
testthat::expect_true(is.antsImage(ceiling(img01)))
testthat::expect_true(is.antsImage(floor(img01)))
testthat::expect_true(is.antsImage(trunc(img01)))
img01 <- as.antsImage(array(1:64, c(4, 4, 4, 1)))
testthat::expect_true(is.antsImage(log(img01)))
testthat::expect_true(is.antsImage(exp(img01)))

testthat::expect_true(is.antsImage(log(img01, base = exp(1))))
testthat::expect_true(is.antsImage(log(img01, base = 2)))
testthat::expect_true(is.antsImage(log(img01, base = 10)))
testthat::expect_true(is.antsImage(log10(img01)))
testthat::expect_true(is.antsImage(log2(img01)))

testthat::expect_true(is.antsImage(gamma(img01)))
testthat::expect_true(is.antsImage(lgamma(img01)))

testthat::expect_true(is.antsImage(cos(img01)))
testthat::expect_true(is.antsImage(cospi(img01)))
testthat::expect_true(is.antsImage(acos(img01)))
testthat::expect_true(is.antsImage(acosh(img01)))

testthat::expect_true(is.antsImage(sin(img01)))
testthat::expect_true(is.antsImage(sinpi(img01)))
testthat::expect_true(is.antsImage(asin(img01)))
testthat::expect_true(is.antsImage(asinh(img01)))

testthat::expect_true(is.antsImage(tan(img01)))
```

```
testthat::expect_true(is.antsImage(tanpi(img01)))  
testthat::expect_true(is.antsImage(atan(img01)))  
testthat::expect_true(is.antsImage(atanh(img01)))
```

antsImage-operators *Operations for antsImage Objects*

Description

Overloaded operators for antsImage objects

Usage

```
## S4 method for signature 'antsImage,antsImage'  
Arith(e1, e2)  
  
## S4 method for signature 'antsImage,antsImage'  
e1 + e2  
  
## S4 method for signature 'antsImage,antsImage'  
e1 - e2  
  
## S4 method for signature 'antsImage,antsImage'  
e1 * e2  
  
## S4 method for signature 'antsImage,antsImage'  
e1 / e2  
  
## S4 method for signature 'antsImage,antsImage'  
e1 ^ e2  
  
## S4 method for signature 'antsImage,antsImage'  
e1 %% e2  
  
## S4 method for signature 'antsImage,antsImage'  
e1 %/% e2  
  
## S4 method for signature 'antsImage,numeric'  
Arith(e1, e2)  
  
## S4 method for signature 'antsImage,numeric'  
e1 + e2  
  
## S4 method for signature 'antsImage,numeric'  
e1 - e2  
  
## S4 method for signature 'antsImage,numeric'
```

```
e1 * e2

## S4 method for signature 'antsImage,numeric'
e1 / e2

## S4 method for signature 'antsImage,numeric'
e1 ^ e2

## S4 method for signature 'antsImage,numeric'
e1 %% e2

## S4 method for signature 'antsImage,numeric'
e1 %/% e2

## S4 method for signature 'numeric,antsImage'
e1 + e2

## S4 method for signature 'numeric,antsImage'
e1 - e2

## S4 method for signature 'numeric,antsImage'
e1 * e2

## S4 method for signature 'numeric,antsImage'
e1 / e2

## S4 method for signature 'numeric,antsImage'
e1 ^ e2

## S4 method for signature 'numeric,antsImage'
e1 %% e2

## S4 method for signature 'numeric,antsImage'
e1 %/% e2

## S4 method for signature 'antsImage,missing'
Arith(e1, e2)

## S4 method for signature 'numeric,antsImage'
Arith(e1, e2)

## S4 method for signature 'antsImage,logical'
Arith(e1, e2)

## S4 method for signature 'logical,antsImage'
Arith(e1, e2)

## S4 method for signature 'antsImage,array'
```



```
Arith(e1, e2)

## S4 method for signature 'array,antsImage'
Arith(e1, e2)

## S4 method for signature 'list,antsImage'
Arith(e1, e2)

## S4 method for signature 'antsImage,list'
Arith(e1, e2)

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

## S4 method for signature 'antsImage,list'
e1 == e2

## S4 method for signature 'antsImage,array'
e1 == e2

## S4 method for signature 'array,antsImage'
e1 == e2

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

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

## S4 method for signature 'antsImage,antsImage'
e1 > e2

## S4 method for signature 'antsImage,list'
e1 > e2

## S4 method for signature 'antsImage,array'
e1 > e2

## S4 method for signature 'array,antsImage'
e1 > e2

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

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

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

```
e1 < e2

## S4 method for signature 'antsImage,list'
e1 < e2

## S4 method for signature 'antsImage,array'
e1 < e2

## S4 method for signature 'array,antsImage'
e1 < e2

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

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

## S4 method for signature 'antsImage,antsImage'
e1 != e2

## S4 method for signature 'antsImage,list'
e1 != e2

## S4 method for signature 'antsImage,array'
e1 != e2

## S4 method for signature 'array,antsImage'
e1 != e2

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

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

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

## S4 method for signature 'antsImage,list'
e1 <= e2

## S4 method for signature 'antsImage,array'
e1 <= e2

## S4 method for signature 'array,antsImage'
e1 <= e2

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

```
e1 <= e2

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

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

## S4 method for signature 'antsImage,list'
e1 >= e2

## S4 method for signature 'antsImage,array'
e1 >= e2

## S4 method for signature 'array,antsImage'
e1 >= e2

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

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

## S4 method for signature 'antsImage,antsImage'
e1 & e2

## S4 method for signature 'antsImage,antsImage'
e1 | e2

## S4 method for signature 'antsImage,array'
Logic(e1, e2)

## S4 method for signature 'array,antsImage'
Logic(e1, e2)

## S4 method for signature 'antsImage,logical'
e1 & e2

## S4 method for signature 'antsImage,logical'
e1 | e2

## S4 method for signature 'logical,antsImage'
Logic(e1, e2)

## S4 method for signature 'antsImage,numeric'
Logic(e1, e2)

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

```

Logic(e1, e2)

## S4 method for signature 'antsImage,ANY'
Logic(e1, e2)

## S4 method for signature 'ANY,antsImage'
Logic(e1, e2)

## S4 method for signature 'list,antsImage'
Logic(e1, e2)

## S4 method for signature 'antsImage,list'
Logic(e1, e2)

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

## S4 method for signature 'antsImage'
anyNA(x, recursive = FALSE)

## S4 method for signature 'antsImage'
is.nan(x)

## S4 method for signature 'antsImage'
is.infinite(x)

## S4 method for signature 'antsImage'
is.finite(x)

## S4 method for signature 'antsImage'
which(x, arr.ind = FALSE, useNames = TRUE)

```

Arguments

e1	is an object of class antsImage.
e2	is an object of class antsImage.
x	a logical vector or array.
recursive	not used
arr.ind	logical; should array indices be returned when x is an array?
useNames	logical indicating if the value of arrayInd() should have (non-null) dimnames at all.

Details

Overloading binary operators for antsImage Objects

Author(s)

John Muschelli <muschellij2@gmail.com>

Examples

```
img01 <- as.antsImage(array(1:64, c(4, 4, 4, 1)))
img02 <- as.antsImage(array(64:1, c(4, 4, 4, 1)))
is.antsImage(img01 + img02)
is.antsImage(sqrt(2) * img01)
is.antsImage(img02 / pi)
img01 <- as.antsImage(array(1:64, c(4, 4, 4, 1)))
et <- function(x) {
  testthat::expect_true(x)
}
x <- is.antsImage(img01 == img01)
et(x)
x <- is.antsImage(img01 == as.array(img01))
et(x)
x <- is.antsImage(img01 == 1)
et(x)
x <- is.antsImage(1 == img01)
et(x)
x <- is.antsImage(as.array(img01) == img01)
et(x)

x <- is.antsImage(img01 >= img01)
et(x)
x <- is.antsImage(img01 >= as.array(img01))
et(x)
x <- is.antsImage(img01 >= 1)
et(x)
x <- is.antsImage(1 >= img01)
et(x)
x <- is.antsImage(as.array(img01) >= img01)
et(x)

x <- is.antsImage(img01 > img01)
et(x)
x <- is.antsImage(img01 > as.array(img01))
et(x)
x <- is.antsImage(img01 > 1)
et(x)
x <- is.antsImage(1 > img01)
et(x)
x <- is.antsImage(as.array(img01) > img01)
et(x)

x <- is.antsImage(img01 < img01)
et(x)
x <- is.antsImage(img01 < as.array(img01))
et(x)
x <- is.antsImage(img01 < 1)
```

```

et(x)
x <- is.antsImage(1 < img01)
et(x)
x <- is.antsImage(as.array(img01) < img01)
et(x)

x <- is.antsImage(img01 <= img01)
et(x)
x <- is.antsImage(img01 <= as.array(img01))
et(x)
x <- is.antsImage(1 <= img01)
et(x)
x <- is.antsImage(img01 <= 1)
et(x)
x <- is.antsImage(as.array(img01) <= img01)
et(x)

x <- is.antsImage(img01 != img01)
et(x)
x <- is.antsImage(img01 != as.array(img01))
et(x)
x <- is.antsImage(1 != img01)
et(x)
x <- is.antsImage(img01 != 1)
et(x)
x <- is.antsImage(as.array(img01) != img01)
et(x)

#####
# Multi image operations
#####
fname <- system.file("extdata",
  "multi_component_image.nii.gz",
  package = "ANTsRCore"
)
img <- antsImageRead(fname)

testthat::expect_is(img > 0, "antsImage")
testthat::expect_is(0 < img, "antsImage")
testthat::expect_is(img > c(0, 0), "antsImage")

testthat::expect_is(img >= 0, "antsImage")
testthat::expect_is(0 <= img, "antsImage")
testthat::expect_is(img >= c(0, 0), "antsImage")

testthat::expect_is(img < 0, "antsImage")
testthat::expect_is(0 > img, "antsImage")
testthat::expect_is(img < c(0, 0), "antsImage")

testthat::expect_is(img <= 0, "antsImage")
testthat::expect_is(0 >= img, "antsImage")
testthat::expect_is(img <= c(0, 0), "antsImage")

```

```

testthat::expect_is(img == 0, "antsImage")
testthat::expect_is(0 == img, "antsImage")
testthat::expect_is(img == c(0, 0), "antsImage")

testthat::expect_is(img != 0, "antsImage")
testthat::expect_is(0 != img, "antsImage")
testthat::expect_is(img != c(0, 0), "antsImage")

img01 <- as.antsImage(array(1:64, c(4, 4, 4, 1)))
testthat::expect_error(img01 == list())
img01 <- as.antsImage(array(1:64, c(4, 4, 4, 1)))
testthat::expect_error(img01 > list())
img01 <- as.antsImage(array(1:64, c(4, 4, 4, 1)))
testthat::expect_error(img01 < list())
img01 <- as.antsImage(array(1:64, c(4, 4, 4, 1)))
testthat::expect_error(img01 != list())
img01 <- as.antsImage(array(1:64, c(4, 4, 4, 1)))
testthat::expect_error(img01 <= list())
img01 <- as.antsImage(array(1:64, c(4, 4, 4, 1)))
testthat::expect_error(img01 >= list())
img01 <- as.antsImage(array(1:64, c(4, 4, 4, 1)))
mask1 <- img01 == 1
mask2 <- img01 == 2
testthat::expect_true(is.antsImage(mask1))
testthat::expect_true(is.antsImage(mask2))
testthat::expect_true(is.antsImage(mask1 & mask2))
testthat::expect_true(is.antsImage(mask1 | mask2))
testthat::expect_true(is.antsImage(as.array(mask2) | mask1))
testthat::expect_true(is.antsImage(as.array(mask2) & mask1))
testthat::expect_true(is.antsImage(mask2 | as.array(mask1)))
testthat::expect_true(is.antsImage(mask2 & as.array(mask1)))

testthat::expect_true(is.antsImage(mask2 | 1))
testthat::expect_true(is.antsImage(mask2 | TRUE))
testthat::expect_true(is.antsImage(mask2 & 1))
testthat::expect_true(is.antsImage(mask2 & TRUE))
img <- makeImage(c(10, 10), rnorm(100))
mask <- img > 0
which(mask)
which(mask, arr.ind = TRUE)

```

Description

Overloaded Summary for antsImage objects

Usage

```
## S4 method for signature 'antsImage'
Summary(x, ..., mask = NULL, na.rm = FALSE)

max(x, ..., na.rm = FALSE)

min(x, ..., na.rm = FALSE)

## S4 method for signature 'antsImage'
min(x, ..., mask = NULL, na.rm = FALSE)

## S4 method for signature 'antsImage'
max(x, ..., mask = NULL, na.rm = FALSE)

range(x, ..., na.rm = FALSE)

prod(x, ..., na.rm = FALSE)

sum(x, ..., na.rm = FALSE)

any(x, ..., na.rm = FALSE)

all(x, ..., na.rm = FALSE)
```

Arguments

<code>x</code>	is an object of class <code>antsImage</code> .
<code>...</code>	further arguments passed to summary methods
<code>mask</code>	binary mask of values to subset
<code>na.rm</code>	logical: should missing values be removed?

Examples

```
vec <- 1:64
img01 <- as.antsImage(array(vec, c(4, 4, 4, 1)))
testthat::expect_equal(max(img01), max(vec))
max(img01)
testthat::expect_equal(min(img01), min(vec))
min(img01)
testthat::expect_equal(sum(img01), sum(vec))
range(img01)
testthat::expect_equal(range(img01), range(vec))
prod(img01 / 25)
testthat::expect_equal(prod(img01 / 25), prod(vec / 25), tolerance = 1e-5)
```

antsImageClone	<i>Clone an antsImage.</i>
----------------	----------------------------

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

<code>in_image</code>	image object of S4 class antsImage to be cloned.
<code>out_pixeltype</code>	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")
multi <- antsImageRead(getANTsRData("multi_component_image"))
img <- antsImageRead(getANTsRData("r16"), 2)
m2 <- antsImageClone(multi)
```

antsImageGetSet	<i>antsImageGetSet</i>
-----------------	------------------------

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))
antsGetSpacing(img)
antsSetSpacing(img, c(2.0, 2.0))
antsGetOrigin(img)
antsSetOrigin(img, c(0.5, 0.5))
img <- makeImage(c(5, 5), rnorm(25))
antsGetDirection(img)
testthat::expect_error(antsGetDirection(as.array(img)))
outimg <- makeImage(c(5, 5), rnorm(25))
antsGetDirection(outimg)
direct <- antsGetDirection(outimg)
antsSetDirection(outimg, direct)
testthat::expect_error(antsSetDirection(as.array(outimg), direct))
testthat::expect_error(antsSetDirection(outimg, as.numeric(direct)))
testthat::expect_error(antsSetDirection(outimg, diag(length(dim(outimg)) + 1)))
```

antsImageHeaderInfo	<i>Read file info from image header</i>
---------------------	---

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"))  
  
img <- antsImageRead(getANTsRData("r16"))  
antsImageHeaderInfo(img)  
antsImageHeaderInfo(getANTsRData("r16"))  
testthat::expect_error(antsImageHeaderInfo(""))
```

antsImageIterator	<i>antsImageIterator</i>
-------------------	--------------------------

Description

Get image iterator

Usage

```
antsImageIterator(x)
```

Arguments

x	antsImage
---	-----------

Value

antsImageIterator

Examples

```
img <- makeImage(c(5, 5), rnorm(25))
it <- antsImageIterator(img)
```

antsImageIterator-class	<i>An S4 class for an antsImageIterator</i>
-------------------------	---

Description

C++ type used to represent an ITK image iterator. the actual iterator is of C++ type 'itk::ImageRegionIteratorWithIndex< itk::image< pixeltype, dimension > >'

Usage

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

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

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

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, currently only 1 is supported
pointer	to the memory location of the itk object

antsImageIteratorGet *antsImageIteratorGet*

Description

Get value at current position

Usage

```
antsImageIteratorGet(x)
```

Arguments

x	antsImageIterator
---	-------------------

Value

pixel value

Examples

```
img <- makeImage(c(5, 5), rnorm(25))
it <- antsImageIterator(img)
value <- antsImageIteratorGet(it)
```

antsImageIteratorGetIndex
antsImageIteratorGetIndex

Description

Get index at current position

Usage

```
antsImageIteratorGetIndex(x)
```

Arguments

x antsImageIterator

Value

image index

Examples

```
img <- makeImage(c(5, 5), rnorm(25))  
it <- antsImageIterator(img)  
index <- antsImageIteratorGetIndex(it)
```

antsImageIteratorGoToBegin
antsImageIteratorGoToBegin

Description

move iterator to begining

Usage

```
antsImageIteratorGoToBegin(x)
```

Arguments

x antsImageIterator

Value

antsImageIterator

Examples

```
img <- makeImage(c(5, 5), rnorm(25))
it <- antsImageIterator(img)
it <- antsImageIteratorNext(it)
it <- antsImageIteratorGoToBegin(it)
```

```
antsImageIteratorGoToReverseBegin
      antsImageIteratorGoToReverseBegin
```

Description

move iterator to beginning of reverse data

Usage

```
antsImageIteratorGoToReverseBegin(x)
```

Arguments

x antsImageIterator

Value

antsImageIterator

Examples

```
img <- makeImage(c(5, 5), rnorm(25))
it <- antsImageIterator(img)
it <- antsImageIteratorGoToReverseBegin(it)
```

```
antsImageIteratorIsAtEnd
      antsImageIteratorIsAtEnd
```

Description

test if iterator is at end of data

Usage

```
antsImageIteratorIsAtEnd(x)
```

Arguments

x antsImageIterator

Value

boolean indicating position

Examples

```
img <- makeImage(c(5, 5), rnorm(25))
it <- antsImageIterator(img)
it <- antsImageIteratorNext(it)
flag <- antsImageIteratorIsAtEnd(it)
```

antsImageIteratorIsAtReverseEnd

antsImageIteratorIsAtReverseEnd

Description

test if iterator is at end of reverse data

Usage

```
antsImageIteratorIsAtReverseEnd(x)
```

Arguments

x antsImageIterator

Value

boolean indicating position

Examples

```
img <- makeImage(c(5, 5), rnorm(25))
it <- antsImageIterator(img)
it <- antsImageIteratorGoToReverseBegin(it)
it <- antsImageIteratorPrevious(it)
flag <- antsImageIteratorIsAtReverseEnd(it)
```

`antsImageIteratorNext` *antsImageIteratorNext*

Description

advance iterator forward

Usage

```
antsImageIteratorNext(x)
```

Arguments

x antsImageIterator

Value

antsImageIterator

Examples

```
img <- makeImage(c(5, 5), rnorm(25))
it <- antsImageIterator(img)
it <- antsImageIteratorNext(it)
```

`antsImageIteratorPrevious`
antsImageIteratorPrevious

Description

advance iterator backward

Usage

```
antsImageIteratorPrevious(x)
```

Arguments

x antsImageIterator

Value

antsImageIterator

Examples

```
img <- makeImage(c(5, 5), rnorm(25))
it <- antsImageIterator(img)
it <- antsImageIteratorPrevious(it)
```

```
antsImageIteratorRemaining
      antsImageIteratorRemaining
```

Description

test if iterator is at end of data

Usage

```
antsImageIteratorRemaining(x)
```

Arguments

x antsImageIterator

Value

boolean indicating if data remains

Examples

```
img <- makeImage(c(5, 5), rnorm(25))
it <- antsImageIterator(img)
it <- antsImageIteratorNext(it)
flag <- antsImageIteratorRemaining(it)
```

```
antsImageIteratorSet    antsImageIteratorSet
```

Description

Set value at current position

Usage

```
antsImageIteratorSet(x, value)
```

Arguments

x antsImageIterator
value pixel value to set

Value

TRUE'

Examples

```
img <- makeImage(c(5, 5), rnorm(25))
it <- antsImageIterator(img)
flag <- antsImageIteratorSet(it, 2.0)
```

antsImageIteratorSetIndex
antsImageIteratorSetIndex

Description

move iterator to a given index

Usage

```
antsImageIteratorSetIndex(x, index)
```

Arguments

x	antsImageIterator
index	index to move to

Value

TRUE'

Examples

```
img <- makeImage(c(5, 5), rnorm(25))
it <- antsImageIterator(img)
flag <- antsImageIteratorSetIndex(it, c(2, 3))
```

antsImageMutualInformation

mutual information between two images

Description

compute mutual information between two images

Usage

```
antsImageMutualInformation(in_image1, in_image2, sampling.percentage, nBins)
```

Arguments

in_image1	antsImage
in_image2	antsImage
sampling.percentage	value between zero and one determining ratio of points in image to use for metric estimation
nBins	number of bins to use in joint histogram estimate, often set between 8 and 32

Value

mutual information value

Author(s)

Brian B. Avants

Examples

```
fi <- antsImageRead(getANTsRData("r16"), 2)
mi <- antsImageRead(getANTsRData("r64"), 2)
testthat::expect_error(antsImageMutualInformation(fi, "hey"))
testthat::expect_error(antsImageMutualInformation("hey", fi))
fi3 <- resampleImage(fi, c(2, 2))
fi2 <- antsImageClone(fi)
pixeltype(fi2) <- "unsigned int"
testthat::expect_error(antsImageMutualInformation(fi2, mi))
testthat::expect_error(antsImageMutualInformation(fi3, mi))
```

`antsImagePhysicalSpaceConsistency`*Check for physical space consistency*

Description

Check if two `antsImage` objects occupy the same physical space

Usage

```
antsImagePhysicalSpaceConsistency(  
  img1,  
  img2,  
  coordinate.tolerance = 0.01,  
  direction.tolerance = 0.01,  
  data.type = FALSE  
)
```

Arguments

<code>img1</code>	Image object of S4 class <code>antsImage</code> .
<code>img2</code>	Image object of S4 class <code>antsImage</code> .
<code>coordinate.tolerance</code>	floating point error tolerance in origin
<code>direction.tolerance</code>	floating point error tolerance in direction matrix
<code>data.type</code>	boolean, if TRUE check pixeltype and components, default is FALSE

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)  
check <- antsImagePhysicalSpaceConsistency(img1, img2, data.type = TRUE)  
  
testthat::expect_error(  
  antsImagePhysicalSpaceConsistency(img1, as.array(img2))  
)  
  
i2 <- img2  
pixeltype(i2) <- "unsigned int"  
testthat::expect_false(  
  antsImagePhysicalSpaceConsistency(img1, i2, data.type = TRUE)
```

```

)

i2 <- mergeChannels(list(img2, img2))
testthat::expect_false(
  antsImagePhysicalSpaceConsistency(img1, i2, data.type = TRUE)
)

i2 <- as.antsImage(img2[1:5, 1:5], reference = img2)
testthat::expect_false(
  antsImagePhysicalSpaceConsistency(img1, i2)
)
i2 <- img2
antsSetSpacing(i2, c(0.5, 0.5))
testthat::expect_false(
  antsImagePhysicalSpaceConsistency(img1, i2)
)
i2 <- img2
antsSetOrigin(i2, c(1, 2))
testthat::expect_false(
  antsImagePhysicalSpaceConsistency(img1, i2)
)

i2 <- img2
antsSetDirection(i2, -1 * antsGetDirection(i2))
testthat::expect_false(
  antsImagePhysicalSpaceConsistency(img1, i2)
)

```

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")
```

antsImageTypeCast	<i>Cast images to either a passed pixeltype or the most precise of the inputs</i>
-------------------	---

Description

Cast images to either a passed pixeltype or the most precise of the inputs

Usage

```
antsImageTypeCast(imageList, pixeltype = NA)
```

Arguments

imageList	list of images
pixeltype	to cast to

Value

list of antsImages

Author(s)

J Duda

Examples

```
img <- antsImageRead(getANTsRData("rand")) %>% resampleImage(c(32, 32))
img2 <- antsImageRead(getANTsRData("rand"), pixeltype = "double") %>%
  resampleImage(c(32, 32))
allDoubles <- antsImageTypeCast(list(img, img2))
floats <- antsImageTypeCast(list(img))
pixeltype(img) <- "unsigned int"
ints <- antsImageTypeCast(list(img))
```

antsImageWrite

Image Write

Description

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

Usage

```
antsImageWrite(image, filename, as.tensor = FALSE)
```

Arguments

image	Image object of S4 class antsImage to be written.
filename	Name of the file to write the image to.
as.tensor	flag indicating to write as symmetric tensor if image has 6 components

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 char"),
  tempfile(fileext = ".jpg")
)
antsImageWrite(
  antsImageClone(fi, "float"),
  tempfile(fileext = ".tif")
)
antsImageWrite(fi, tempfile(fileext = ".mrc"))
antsImageWrite(fi, tempfile(fileext = ".hd5"))
components(fi) <- 0L
antsImageWrite(fi, tempfile(fileext = ".nii.gz"))

```

antsIncludes	<i>return ANTs installation information</i>
--------------	---

Description

call this to get the include path for ANTs headers and libraries

Usage

```
antsIncludes()
```

Author(s)

Avants BB

Examples

```
antsIncludes()
```

antsLibs	<i>return ants library location</i>
----------	-------------------------------------

Description

call this to get the path to ants static libraries to which you will link

Usage

```
antsLibs()
```

Author(s)

Avants BB

Examples

```
antsLibs()
```

antsMatrix-class	<i>antsMatrix Class</i>
------------------	-------------------------

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)
```

Arguments

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

Slots

elementtype	string of the type of storage of the matrix e.g. "float"
pointer	the memory location

Examples

```
mat <- as.antsMatrix(matrix(rnorm(10), nrow = 2))
as.data.frame(mat)
as.matrix(mat)
```

antsMotionCalculation *antsMotionCalculation*

Description

Correct 4D time-series data for motion.

Usage

```
antsMotionCalculation(
  img,
  mask = NULL,
  fixed = NULL,
  moreaccurate = 1,
  txtype = "Affine",
  framewise = 1,
  verbose = FALSE,
  ...
)
```

Arguments

<code>img</code>	antsImage, usually 4D.
<code>mask</code>	mask for image (3D). If not provided, estimated from data.
<code>fixed</code>	Fixed image to register all timepoints to. If not provided, mean image is used.
<code>moreaccurate</code>	strategy desired for motion correction. One of 0 (test) 1 (high-res only), 2 (multi-level inter-subject), 3 (FIXME), or a special method for intraSubjectBOLD.
<code>txtype</code>	Type of transform. One of "Affine", "Rigid", or "SyN".
<code>framewise</code>	Calculate framewise displacement?
<code>verbose</code>	enables verbose output.
<code>...</code>	additional argument to .motion_correction

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.

fd Time-series mean and max displacements.

dvars DVARS, derivative of frame-wise intensity changes.

Author(s)

Benjamin M. Kandel

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(paramList)

Arguments

paramList 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

Note

This function may give different results on multiple runs.

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))
aimg2 <- new("antsImage", "float", 3)
```

```
antsMotionCorr(list(d = 3, a = bold, o = aimg2))
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)
```

antspymm_nuisance_names

return nuisance variable strings

Description

these strings can be used with `getNamesFromDataFrame` or `multigrep` to either include or exclude nuisance variables.

Usage

```
antspymm_nuisance_names()
```

antspymm_predictors *Generate Predictors from ANTsPyMM Imaging Data*

Description

This function generates a list of variable names to be used as predictors in a model, based antspymm tabular version of imaging data. It filters and processes the data to create meaningful predictors. LRAVG

Usage

```
antspymm_predictors(demog, doasym = FALSE, return_colnames = FALSE)
```

Arguments

demog	A dataframe containing demographic and imaging data.
doasym	boolean
return_colnames	boolean

Value

A dataframe with processed demographic and imaging data.

Examples

```
# predictors <- antspymm_predictors(demographic_data)
```

antspymm_qc_names	<i>Get Quality Control (QC) Metric Names</i>
-------------------	--

Description

This function returns a vector of quality control (QC) metric names used in the antspymm package. These metrics include volume measures, reflection errors, PSNR (Peak Signal-to-Noise Ratio), CNR (Contrast-to-Noise Ratio), and various metrics related to motion correction in rsfMRI and DTI.

Usage

```
antspymm_qc_names()
```

Value

A character vector containing the names of QC metrics.

Examples

```
qc_names <- antspymm_qc_names()
print(qc_names)
```

antspymm_vartype	<i>Determine the Variable Type Based on Its Name</i>
------------------	--

Description

This function inspects the input character vector for specific patterns indicating the type of variable (e.g., "T1", "rsfMRI", "DTI", "NM2") and returns a corresponding string identifier for the first matched type. If no known pattern is found, it returns 'NA'.

Usage

```
antspymm_vartype(x)
```

Arguments

x	A character vector containing names or identifiers to be checked against known patterns for variable types. It must be a character vector.
---	--

Value

A character string indicating the type of variable matched based on the predefined patterns ("T1", "rsfMRI", "DTI", "NM2DMT"). Returns 'NA' if no pattern matches.

Note

This function only checks for the first occurrence of a pattern and does not account for multiple different patterns within the same input. The order of pattern checking is fixed and may affect the result if multiple patterns are present.

Examples

```
antspymm_vartype("This is a T1 weighted image") # Returns "T1"
antspymm_vartype("Subject underwent rsfMRI")    # Returns "rsfMRI"
antspymm_vartype("DTI sequence")                 # Returns "DTI"
antspymm_vartype("Analysis of NM2")              # Returns "NM2DMT"
antspymm_vartype("Unknown type")                 # Returns NA
```

antsr_mcca_features	<i>Generate Multi-view CCA-based Feature Projections for a List of Matrices</i>
---------------------	---

Description

Applies multi-view Canonical Correlation Analysis (MCCA) to a list of subject-by-feature matrices. Each view (modality) is projected into a shared latent space that maximizes inter-view correlation.

Usage

```
antsr_mcca_features(
  data_matrices,
  k = 10,
  regularization = 0.001,
  center = TRUE,
  scale = FALSE
)
```

Arguments

data_matrices	A named list of numeric matrices, each subjects x features. All matrices must have the same number of subjects (rows).
k	Integer. Number of canonical components to extract.
regularization	Scalar ridge penalty to improve numerical stability (default = 1e-3).
center	Logical. Whether to center each matrix before analysis (default = TRUE).
scale	Logical. Whether to scale each matrix to unit variance (default = FALSE).

Details

This provides a classical linear baseline for benchmarking against SiMLR or multimodal deep models.

Value

A list containing:

loadings	List of projection matrices (features x k) for each view.
scores	List of canonical variates (subjects x k) for each view.
shared_embedding	Average canonical embedding across views (subjects x k).
correlations	Vector of canonical correlations.

Examples

```
## Not run:
set.seed(1500)
nsub <- 25
npix <- c(100, 200, 133)
nk <- 5
outcome <- matrix(rnorm(nsub * nk), ncol = nk)
outcome1 <- matrix(rnorm(nsub * nk), ncol = nk)
outcome2 <- matrix(rnorm(nsub * nk), ncol = nk)
outcome3 <- matrix(rnorm(nsub * nk), ncol = nk)
view1tx <- matrix(rnorm(npix[1] * nk), nrow = nk)
view2tx <- matrix(rnorm(npix[2] * nk), nrow = nk)
view3tx <- matrix(rnorm(npix[3] * nk), nrow = nk)
mat1 <- (outcome %*% t(outcome1) %*% (outcome1)) %*% view1tx
mat2 <- (outcome %*% t(outcome2) %*% (outcome2)) %*% view2tx
mat3 <- (outcome %*% t(outcome3) %*% (outcome3)) %*% view3tx
matlist <- list(vox = mat1, vox2 = mat2, vox3 = mat3)
mcca <- antsr_mcca_features(matlist, k = 5)
str(mcca)

## End(Not run)
```

antsr_pca_features	<i>Generate PCA-based feature projections for a list of voxel matrices</i>
--------------------	--

Description

This function applies principal component analysis (PCA) to each matrix in a list of voxel matrices and returns a list of projection matrices (principal axes).

Usage

```
antsr_pca_features(data_matrices, k)
```


Arguments

- `data_matrices` A list of numeric matrices. Each matrix should have dimensions (subjects by voxels).
- `k` Integer. Number of principal components to retain.

Value

A named list of projection matrices (voxels by k), one per input matrix.

`antsr_random_features` *Generate reproducible random feature projections for a list of voxel matrices*

Description

This function applies random projection to each matrix in a list of voxel matrices. It uses a fixed seed to ensure reproducibility across runs.

Usage

```
antsr_random_features(data_matrices, k, seed = 42)
```

Arguments

- `data_matrices` A list of numeric matrices. Each matrix should have dimensions (subjects by voxels).
- `k` Integer. Number of projection dimensions (features) to generate.
- `seed` Integer. Random seed for reproducibility. Default is 42.

Value

A list of projection matrices, each with dimensions (voxels by k).

`antsr_rgcca_baseline` *Multi-view Regularized CCA baseline using RGCCA*

Description

Multi-view Regularized CCA baseline using RGCCA

Usage

```
antsr_rgcca_baseline(data_matrices, tau = 0.5, scheme = "factorial", ncomp = 3)
```

Arguments

data_matrices	A named list of subject x feature matrices
tau	Regularization parameter (0 = no regularization, 1 = PCA limit)
scheme	RGCCA scheme ('horst', 'factorial', 'centroid')
ncomp	Number of components

antsr_rgcca_sparse	<i>Sparse Multi-view CCA Baseline using RGCCA (version-safe)</i>
--------------------	--

Description

Implements a sparse multi-view canonical correlation analysis using RGCCA, enforcing sparsity on each modality to obtain interpretable feature loadings. Compatible with RGCCA ≥ 3.0 (using 'blocks' and 'connection' arguments).

Usage

```
antsr_rgcca_sparse(
  data_matrices,
  ncomp = 3,
  tau = 0.5,
  sparsity = 0.3,
  scheme = "factorial",
  scale = TRUE,
  verbose = FALSE
)
```

Arguments

data_matrices	Named list of subject x feature matrices. Each must have the same number of rows.
ncomp	Number of canonical components per view (default = 3)
tau	Numeric in [0,1] or vector. Regularization per view (0 = pure CCA, 1 = PCA limit)
sparsity	Numeric in (0,1]. L1 sparsity per view; lower = sparser (default = 0.3)
scheme	RGCCA optimization scheme: 'factorial', 'centroid', or 'horst' (default = 'factorial')
scale	Logical. Whether to z-score features (default = TRUE)
verbose	Logical. Print progress messages (default = FALSE)

Value

A list with:

loadings	List of sparse feature loadings (features x components)
scores	List of subject x component canonical variates
shared_embedding	Average canonical embedding across modalities
correlations	Canonical correlations across shared space

antsr_sPCA_features	<i>Generate sparse PCA-based feature projections for a list of voxel matrices</i>
---------------------	---

Description

Applies sparse principal component analysis using the selected backend ("elasticnet", "PMA", or "sparsepca") to each matrix in a list of subject by voxel data.

Usage

```
antsr_sPCA_features(
  data_matrices,
  k,
  method = c("default", "elasticnet", "PMA", "sparsepca"),
  para = NULL
)
```

Arguments

data_matrices	A list of numeric matrices. Each matrix should be subjects by voxels.
k	Integer. Number of components to retain.
method	Character. Sparse PCA backend to use. One of "default", "elasticnet", "PMA", or "sparsepca".
para	Sparsity control parameter(s). Interpretation depends on backend: - For "elasticnet": number of nonzero loadings per component (length-k vector). - For "PMA": L1 bound on loading vector (scalar or length-k). - For "sparsepca": ignored (uses built-in defaults). - For "default": length-k vector of sparsity parameters.

Value

A named list of sparse projection matrices (voxels by k).

antsRegion-class	<i>An S4 class to hold a region of an antsImage</i>
------------------	---

Description

An S4 class to hold a region of an antsImage

Slots

index indices of the region
size size of the region

antsRegistration	<i>Perform registration between two images.</i>
------------------	---

Description

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

Usage

```
antsRegistration(
  fixed = NULL,
  moving = NULL,
  typeofTransform = "SyN",
  initialTransform = NA,
  outprefix = NULL,
  mask = NULL,
  movingMask = NULL,
  maskAllStages = FALSE,
  gradStep = 0.2,
  flowSigma = 3,
  totalSigma = 0,
  affMetric = "mattes",
  affSampling = 32,
  synMetric = "mattes",
  synSampling = 32,
  affIterations,
  regIterations = c(40, 20, 0),
  multivariateExtras,
  restrictTransformation,
  writeCompositeTransform = FALSE,
  samplingPercentage = 0.2,
  verbose = FALSE,
```

```

    printArgs = FALSE,
    ...
)

```

Arguments

<code>fixed</code>	fixed image to which we register the moving image.
<code>moving</code>	moving image to be mapped to fixed space.
<code>typeofTransform</code>	A linear or non-linear registration type. Mutual information metric by default. See Details.
<code>initialTransform</code>	either a single transform file name, a vector of transform file names, or a single <code>antsTransform</code> object. If NA, an initial translation aligns the center of mass of the moving image to that of the fixed image, unless the transform is "deformable only", in which case the initial transform is 'Identity'.
<code>outprefix</code>	output will be named with this prefix.
<code>mask</code>	Registration metric mask in the fixed image space.
<code>movingMask</code>	Registration metric mask in the moving image space.
<code>maskAllStages</code>	If true, apply metric mask(s) to all registration stages, instead of just the final stage.
<code>gradStep</code>	gradient step size (not for all tx)
<code>flowSigma</code>	smoothing for update field
<code>totalSigma</code>	smoothing for total field
<code>affMetric</code>	the metric for the affine part (GC, mattes, meansquares)
<code>affSampling</code>	the sampling parameter for the affine metric
<code>synMetric</code>	the metric for the syn part (CC, mattes, meansquares, demons)
<code>synSampling</code>	the nbins or radius parameter for the syn metric
<code>affIterations</code>	vector of iterations for low-dimensional registration. we will set the smoothing and multi-resolution parameters based on the length of this vector.
<code>regIterations</code>	vector of iterations for syn. we will set the smoothing and multi-resolution parameters based on the length of this vector.
<code>multivariateExtras</code>	list of additional images and metrics which will trigger the use of multiple metrics in the registration process in the deformable stage. Multivariate metrics needs 5 entries: name of metric, fixed, moving, weight, samplingParam. the list should be of the form <code>list(list("nameOfMetric2", img, img, weight, metricParam))</code> . Another example would be <code>list(list("MeanSquares", f2, m2, 0.5, 0), list("CC", f2, m2, 0.5, 2))</code> . This is only compatible with the <code>SynOnly</code> , <code>ElasticOnly</code> , or <code>antsRegistrationSyn*</code> transformations.
<code>restrictTransformation</code>	This option allows the user to restrict the optimization of the displacement field, translation, rigid or affine transform on a per-component basis. For example, if one wants to limit the deformation or rotation of 3-D volume to the first two

dimensions, this is possible by specifying a weight vector of $c(1, 1, 0)$ for a 3D deformation field or $c(1, 1, 0, 1, 1, 0)$ for a rigid transformation. Restriction currently only works if there are no preceding transformations.

`writeCompositeTransform` if TRUE, will write transformations to h5 format. Defaults to FALSE.

`samplingPercentage` value between zero and one that allows the percentage of points sampled to be controlled in low-dimensional metric estimation.

`verbose` request verbose output (useful for debugging)

`printArgs` print raw command line (useful for debugging)

... additional options see `antsRegistration` in ANTs

Details

`typeofTransform` can be one of:

"Translation": Translation transformation.

"Rigid": Rigid transformation: Only rotation and translation.

"Similarity": Similarity transformation: scaling, rotation and translation.

"QuickRigid": Rigid transformation: Only rotation and translation. May be useful for quick visualization fixes.'

"DenseRigid": Rigid transformation: Only rotation and translation. Employs dense sampling during metric estimation.'

"BOLDRigid": Rigid transformation: Parameters typical for BOLD to BOLD intrasubject registration'.'

"Affine": Affine transformation: Rigid + scaling.

"AffineFast": Fast version of Affine.

"BOLDAffine": Affine transformation: Parameters typical for BOLD to BOLD intrasubject registration'.'

"TRSAA": translation, rigid, similarity, affine (twice). please set `regIterations` if using this option. this would be used in cases where you want a really high quality affine mapping (perhaps with mask).

"ElasticSyN": Symmetric normalization: Affine + deformable transformation. Uses `synMetric` as optimization metric and elastic regularization.

"SyN": Symmetric normalization: Affine + deformable transformation. Uses `synMetric` as optimization metric.

"SyNRA": Symmetric normalization: Rigid + Affine + deformable transformation. Uses `synMetric` as optimization metric.

"SyNOnly": Symmetric normalization: no initial transformation. Uses `synMetric` as optimization metric. Assumes images are aligned by an initial transformation. Can be useful if you want to run an unmasked affine followed by masked deformable registration.

"ElasticOnly": Elastic normalization: no initial transformation.

"SyNCC": SyN, but with cross-correlation as the metric. Note, the default or chosen parameters will be replaced with `synMetric="CC"`, `synSampling=4`, `synits="2100x1200x1200x20"`, `smoothingsigmas="3x2x1x0"`, `shrinkfactors="4x3x2x1"`.

"SyNabp": SyN optimized for `abpBrainExtraction`, forces mutual information as optimization metric.

"SyNBold": SyN, but optimized for registrations between BOLD and T1 images.

"SyNBoldAff": SyN, but optimized for registrations between BOLD and T1 images, with additional affine step.

"SyNAggro": SyN, but with more aggressive registration (fine-scale matching and more deformation). Takes more time than SyN.

"TV[n ":] time-varying diffeomorphism with where 'n' indicates number of time points in velocity field discretization. The initial transform should be computed, if needed, in a separate call to `ants.registration`.

"TVMSQ": time-varying diffeomorphism with mean square metric

"TVMSQC": time-varying diffeomorphism with mean square metric for very large deformation

"Elastic": simple elastic deformation. one might want to run an affine transformation before this. may not produce diffeomorphic transformations. user may need to explore gradient and sigma parameters. this will not produce a valid inverse deformation. `totalSigma` should be greater than zero.

"antsRegistrationSyN[x ":] recreation of the `antsRegistrationSyN.sh` script in ANTs where 'x' is one of the transforms available (e.g., 't', 'b', 's')

"antsRegistrationSyNQuick[x ":] recreation of the `antsRegistrationSyNQuick.sh` script in ANTs where 'x' is one of the transforms available (e.g., 't', 'b', 's')

"antsRegistrationSyNRepro[x ":] reproducible registration. x options as above.

"antsRegistrationSyNQuickRepro[x ":] quick reproducible registration. x options as above.

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.

Output of 1 indicates failure

Author(s)

Shrinidhi KL, Tustison NJ, Avants BB

Examples

```

# print help
antsRegistration()
fi <- antsImageRead(getANTsRData("r16"))
mi <- antsImageRead(getANTsRData("r64"))
mytx2 <- antsRegistration(fixed = fi, typeofTransform = "")
rig <- antsRegistration(
  fixed = fi, moving = mi,
  typeofTransform = "Rigid", verbose = TRUE
)
trans <- readAntsrTransform(rig$fdwtransforms, 2)
postrig <- antsRegistration(
  fixed = fi, moving = mi,
  typeofTransform = "Affine", initialTransform = trans
)
for (itype in c("AffineFast", "BOLDAffine")) {
  print(itype)
  mytx2 <- antsRegistration(
    fixed = fi, moving = mi,
    typeofTransform = itype
  )
}
mytx2 <- antsRegistration(
  fixed = fi, moving = mi,
  typeofTransform = "SyNOnly",
  multivariateExtras = list(list("MeanSquares", fi, mi, 0.5, 0))
)
testthat::expect_error(
  antsRegistration(fixed = fi, moving = mi, typeofTransform = "sdf")
)
bad <- antsRegistration(fixed = fi, moving = mi, regIterations = 40)
affIterations <- c(3, 2, 1, 0)
mytx2 <- antsRegistration(
  fixed = fi, moving = mi,
  affIterations = affIterations
)
fi <- antsImageRead(getANTsRData("r16"))
mi <- antsImageRead(getANTsRData("r64"))
fi <- resampleImage(fi, c(60, 60), 1, 0)
mi <- resampleImage(mi, c(50, 50), 1, 0) # speed up
mytx <- antsRegistration(fixed = fi, moving = mi, typeofTransform = c("SyN"))
mywarpedimage <- antsApplyTransforms(
  fixed = fi, moving = mi,
  transformlist = mytx$fdwtransforms
)
mytx2 <- antsRegistration(fixed = fi, moving = mi, typeofTransform = c("SyN"))
mywarpedimage2 <- antsApplyTransforms(
  fixed = fi, moving = mi,
  transformlist = mytx2$fdwtransforms
)
# testthat::expect_equal(as.array(mywarpedimage), as.array(mywarpedimage2))

```



```
## Not run:
# quick visualization fix for images with odd orientation
mni <- antsImageRead(getANTsRData("mni"))
strokt1 <- antsImageRead("strokt1.nii.gz")
strokt1reg <- antsRegistration(
  fixed = mni,
  moving = strokt1,
  typeofTransform = "QuickRigid", verbose = TRUE
)
plot(strokt1reg$warpedmovout, axis = 3, nslices = 20)
# now - how to use a mask
fi <- antsImageRead(getANTsRData("r16"))
fiseg <- kmeansSegmentation(fi, 3)
mi <- antsImageRead(getANTsRData("r64"))
msk <- thresholdImage(fiseg$segmentation, 0, 0)
mytx <- antsRegistration(
  fixed = fi, moving = mi, typeofTransform = c("SyNCC"),
  mask = msk, verbose = F
)
jac <- createJacobianDeterminantImage(fi, mytx$fwdtransforms[1])

## End(Not run)
```

antsrGetPointerName	<i>Return the antsImage pointer string</i>
---------------------	--

Description

This is a low-level function that may be useful for debugging. It will return the pointer name itself from an `antsImage` object.

Usage

```
antsrGetPointerName(img)
```

Arguments

<code>img</code>	image whose pointer we want
------------------	-----------------------------

Value

string

Author(s)

Avants BB

Examples

```
img <- antsImageRead(getANTsRData("r16"))
antsrGetPointerName(img)
antsrGetPointerName(antsImageClone(img))
```

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)
mean.impute <- antsrImpute(mydat$B)
median.impute <- antsrImpute(mydat, median)
```

antsrMetric-class	<i>An S4 class for image metrics between two antsImages</i>
-------------------	---

Description

C++ type used to represent an ITK image metric

Usage

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

## S4 method for signature 'antsrMetric'
initialize(.Object, dimension = 3, type = "MeanSquares")
```

Arguments

- object input object to convert
- .Object input object to convert
- dimension dimensionality of the transform (2,3,or 4)
- type type of transform' etc noting that short is not supported

Slots

- dimension usually 2 or 3 but can be 4
- precision math precision is float or double'
- type The type of transform: usually one of AffineTransform, CenteredAffineTransform, CenteredEuler3DTransform, CenteredRigid2DTransform, CenteredSimilarity2DTransform, Euler2DTransform, Euler3DTransform, QuaternionRigidTransform, Rigid2DTransform, Similarity2DTransform, Similarity3DTransform
- pointer to the memory location of the itk object

antsrMetricCreate	<i>antsrMetricCreate</i>
-------------------	--------------------------

Description

create object that measures similarity between two images

Usage

```
antsrMetricCreate(
  fixed,
  moving,
  type = c("MeanSquares", "MattesMutualInformation", "ANTSNeighborhoodCorrelation",
    "Correlation", "Demons", "JointHistogramMutualInformation"),
  fixed.mask = NULL,
  moving.mask = NULL,
  sampling.strategy = "none",
  sampling.percentage = 1,
  nBins = 32,
  radius = 3
)
```

Arguments

fixed	the fixed antsImage
moving	the moving antsImage
type	image metric to calculate MeanSquares Correlation ANTSNeighborhoodCorrelation MattesMutualInformation JointHistogramMutualInformation Demons
fixed.mask	mask for the fixed image
moving.mask	mask for the moving image
sampling.strategy	sampling strategy, default if full sampling none Full sampling random regular
sampling.percentage	percentage of data to sample when calculating metric
nBins	number of bins for histogram metrics (default = 32)
radius	radius for neighborhoo metrics (default = 3)

Value

value of image to image metric

Examples

```
x <- antsImageRead(getANTsRData("r16"))
y <- antsImageRead(getANTsRData("r30"))
metric <- antsrMetricCreate(x, y, type = "MeanSquares")
```

antsrMetricGetDerivative
antsrMetricGetDerivative

Description

get derivative of image metric

Usage

```
antsrMetricGetDerivative(metric)
```

Arguments

metric an 'antsrMetric'

Value

image similarity value

Examples

```
x <- antsImageRead(getANTsRData("r16"))
y <- antsImageRead(getANTsRData("r30"))
metric <- antsrMetricCreate(x, y, type = "MeanSquares")
tx <- createAntsrTransform(precision = "double", type = "AffineTransform", dimension = 2)
setAntsrTransformParameters(tx, c(0, -1, 1, 0, 0, 0))
setAntsrTransformFixedParameters(tx, c(128, 128))
antsrMetricSetMovingTransform(metric, tx)
metricValue <- antsrMetricGetDerivative(metric)
```

antsrMetricGetValue *antsrMetricGetValue*

Description

get current value of metric

Usage

```
antsrMetricGetValue(metric)
```

Arguments

metric an 'antsrMetric'

Value

image similarity value

Examples

```
x <- antsImageRead(getANTsRData("r16"))
y <- antsImageRead(getANTsRData("r30"))
metric <- antsrMetricCreate(x, y, type = "MeanSquares")
metricValue <- antsrMetricGetValue(metric)
```

antsrMetricInitialize *antsrMetricInitialize*

Description

prepare to return values

Usage

```
antsrMetricInitialize(metric)
```

Arguments

metric an 'antsrMetric'

Note

must call this after setting up object, before getting values back

Examples

```
x <- antsImageRead(getANTsRData("r16"))
y <- antsImageRead(getANTsRData("r30"))
z <- getMask(y)
metric <- antsrMetricCreate(x, y, type = "MeanSquares")
antsrMetricSetMovingImageMask(metric, z)
antsrMetricInitialize(metric)
```

```
antsrMetricSetFixedImage  
    antsrMetricSetFixedImage
```

Description

set fixed image for image to image metric

Usage

```
antsrMetricSetFixedImage(metric, image)
```

Arguments

metric	an 'antsrMetric'
image	the fixed 'antsImage'

Note

After calling this, must call antsrMetricInitialize(metric)

Examples

```
x <- antsImageRead(getANTsRData("r16"))  
y <- antsImageRead(getANTsRData("r30"))  
metric <- antsrMetricCreate(x, y, type = "MeanSquares")  
z <- x * 2  
antsrMetricSetFixedImage(metric, z)
```

```
antsrMetricSetFixedImageMask  
    antsrMetricSetFixedImageMask
```

Description

set fixed image for image to image metric

Usage

```
antsrMetricSetFixedImageMask(metric, image)
```

Arguments

metric	an 'antsrMetric'
image	the fixed 'antsImage'

Note

After calling this, must call `antsrMetricInitialize(metric)`

Examples

```
x <- antsImageRead(getANTsRData("r16"))
y <- antsImageRead(getANTsRData("r30"))
metric <- antsrMetricCreate(x, y, type = "MeanSquares")
z <- getMask(x)
antsrMetricSetFixedImageMask(metric, z)
```

`antsrMetricSetFixedTransform`

antsrMetricSetFixedTransform

Description

set transform for fixed image

Usage

```
antsrMetricSetFixedTransform(metric, transform)
```

Arguments

<code>metric</code>	an 'antsrMetric'
<code>transform</code>	an 'antsrTransform'

Examples

```
x <- antsImageRead(getANTsRData("r16"))
y <- antsImageRead(getANTsRData("r30"))
metric <- antsrMetricCreate(x, y, type = "ANTSNeighborhoodCorrelation")
tx <- createAntsrTransform(precision = "double", type = "AffineTransform", dimension = 2)
setAntsrTransformParameters(tx, c(0, -1, 1, 0, 0, 0))
setAntsrTransformFixedParameters(tx, c(128, 128))
antsrMetricSetFixedTransform(metric, tx)
antsrMetricGetValue(metric)
```

```
antsrMetricSetMovingImage  
    antsrMetricSetMovingImage
```

Description

set moving image for image to image metric

Usage

```
antsrMetricSetMovingImage(metric, image)
```

Arguments

metric	an 'antsrMetric'
image	the moving 'antsImage'

Note

After calling this, must call antsrMetricInitialize(metric)

Examples

```
x <- antsImageRead(getANTsRData("r16"))  
y <- antsImageRead(getANTsRData("r30"))  
metric <- antsrMetricCreate(x, y, type = "MeanSquares")  
z <- y * 2  
antsrMetricSetMovingImage(metric, z)
```

```
antsrMetricSetMovingImageMask  
    antsrMetricSetMovingImageMask
```

Description

set moving image for image to image metric

Usage

```
antsrMetricSetMovingImageMask(metric, image)
```

Arguments

metric	an 'antsrMetric'
image	the moving 'antsImage'

Note

After calling this, must call `antsrMetricInitialize(metric)`

Examples

```
x <- antsImageRead(getANTsRData("r16"))
y <- antsImageRead(getANTsRData("r30"))
metric <- antsrMetricCreate(x, y, type = "MeanSquares")
z <- getMask(y)
antsrMetricSetMovingImageMask(metric, z)
```

```
antsrMetricSetMovingTransform
      antsrMetricSetMovingTransform
```

Description

set transform for moving image

Usage

```
antsrMetricSetMovingTransform(metric, transform)
```

Arguments

<code>metric</code>	an 'antsrMetric'
<code>transform</code>	an 'antsrTransform'

Examples

```
x <- antsImageRead(getANTsRData("r16"))
y <- antsImageRead(getANTsRData("r30"))
metric <- antsrMetricCreate(x, y, type = "ANTSNeighborhoodCorrelation")
tx <- createAntsrTransform(precision = "double", type = "AffineTransform", dimension = 2)
setAntsrTransformParameters(tx, c(0, -1, 1, 0, 0, 0))
setAntsrTransformFixedParameters(tx, c(128, 128))
antsrMetricSetMovingTransform(metric, tx)
antsrMetricGetValue(metric)
```

```
antsrMetricSetNumberOfHistogramBins
      antsrMetricSetNumberOfHistogramBins
```

Description

set histogram bins image to image metric

Usage

```
antsrMetricSetNumberOfHistogramBins(metric, nBins)
```

Arguments

metric	an 'antsrMetric' of type 'MattesMutualInformation' or 'JointHistogramMutualInformation'
nBins	number of bins (minimum is 5 even for binary data)

Examples

```
x <- antsImageRead(getANTsRData("r16"))
y <- antsImageRead(getANTsRData("r30"))
metric <- antsrMetricCreate(x, y, type = "MattesMutualInformation")
antsrMetricSetNumberOfHistogramBins(metric, 12)
```

```
antsrMetricSetRadius      antsrMetricSetRadius
```

Description

set neighborhood radius

Usage

```
antsrMetricSetRadius(metric, radius)
```

Arguments

metric	an 'antsrMetric' of type 'ANTSNeighborhoodCorrelation'
radius	radius of neighborhood

Examples

```
x <- antsImageRead(getANTsRData("r16"))
y <- antsImageRead(getANTsRData("r30"))
metric <- antsrMetricCreate(x, y, type = "ANTSNeighborhoodCorrelation")
antsrMetricSetRadius(metric, 5)
```

antsrMetricSetSampling

antsrMetricSetSampling

Description

set image sampling strategy and rate

Usage

```
antsrMetricSetSampling(metric, sampling.strategy, sampling.percentage)
```

Arguments

metric an 'antsrMetric'

sampling.strategy
 sampling strategy, default if full sampling
 none Full sampling
 random
 regular

sampling.percentage
 percentage of data to sample when calculating metric

Note

After calling this, must call antsrMetricInitialize(metric)

Examples

```
x <- antsImageRead(getANTSRData("r16"))
y <- antsImageRead(getANTSRData("r30"))
metric <- antsrMetricCreate(x, y, type = "MeanSquares")
antsrMetricSetSampling(metric, "random", 0.4)
```

antsrMotionCalculation

antsrMotionCalculation

Description

Correct time-series data for motion.

Usage

```
antsrMotionCalculation(
  img,
  fixed,
  mask,
  typeofTransform = c("Rigid", "QuickRigid", "BOLDRigid", "Affine", "AffineFast",
    "BOLDAffine", "SyN", "SyNOnly"),
  getMotionDescriptors = TRUE,
  verbose = FALSE,
  ...
)
```

Arguments

<code>img</code>	antsImage, usually ND where D=4.
<code>fixed</code>	Fixed image to register all timepoints to. If not provided, mean image is used.
<code>mask</code>	mask for image (ND-1). If not provided, estimated from data.
<code>typeofTransform</code>	One of "Affine", "Rigid", "BOLDAffine", "BOLDRigid", "QuickRigid".
<code>getMotionDescriptors</code>	computes dvars and framewise displacement. May take additional memory.
<code>verbose</code>	enables verbose output.
<code>...</code>	extra parameters passed to antsRegistration

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.
- fd** Time-series mean and max displacements.
- dvars** DVARS, derivative of frame-wise intensity changes.

Author(s)

BB Avants, Benjamin M. Kandel, JT Duda, Jeffrey S. Phillips

antsrSurf	<i>Render a function onto a surface.</i>
-----------	--

Description

Use a system call to antsSurf in order to render a function onto a surface. This requires antsSurf to be in the environment as well as ConvertScalarImageToRGB, both available in Advanced Normalization Tools software on github.

Usage

```
antsrSurf(
  x,
  y,
  z,
  quantlimits = c(0.1, 0.9),
  colormap = "jet",
  alpha = NA,
  inflationFactor = 25,
  smoothingSigma = 0,
  rotationParams = c(270, 0, 270),
  overlayLimits = NA,
  backgroundColor,
  filename = NA,
  antspath = NA,
  verbose = FALSE
)
```

Arguments

x	input antsImage defining the surface on which to render
y	input antsImage list defining the function to render on the surface. these image(s) should be in the same space as x.
z	input antsImage list mask for each y function to render on the surface. these image(s) should be in the same space as y.
quantlimits	lower and upper quantile limits for overlay
colormap	character, one of: grey, red, green, blue, copper, jet, hsv, spring, summer, autumn, winter, hot, cool, overunder, custom
alpha	transparency vector for underlay and each overlay, default zero
inflationFactor	number of inflation iterations to run
smoothingSigma	gaussian smooth the overlay by this sigma
rotationParams	3 Rotation angles expressed in degrees or a matrix of rotation parameters that will be applied in sequence.

overlayLimits absolute lower and upper limits for functional overlay. this parameter will override quantlimits. Currently, this will set levels above overlayLimits[2] to overlayLimits[2]. Can be a list of length of y.

backgroundColor a three entry vector where each entry is in the range of 0 to 255 and corresponds to red green and blue color channels. Alternatively, pass a string white or black.

filename prefix filename for output pngs

antspath pass the ANTSPATH here otherwise we try to detect it from environment

verbose prints the command used to call antsrSurf

Value

no output

Author(s)

Avants BB, Tustison NJ

Examples

Not run:

```
myantspath <- Sys.getenv("ANTSPATH")
surf_ex <- file.path(myantspath, "antsSurf")
if (file.exists(surf_ex)) {
  ch2i <- antsImageRead(getANTsRData("ch2"))
  ch2seg <- thresholdImage(ch2i, "Otsu", 3)
  wm <- thresholdImage(ch2seg, 3, 3)
  wm2 <- smoothImage(wm, 1) %>% thresholdImage(0.5, Inf)
  king <- weingartenImageCurvature(ch2i, 1.5) %>% smoothImage(1)
  rp1 <- matrix(c(90, 180, 90), ncol = 3)
  rp2 <- matrix(c(90, 180, 270), ncol = 3)
  rp3 <- matrix(c(90, 180, 180), ncol = 3)
  rp <- rbind(rp1, rp3, rp2)
  antsrSurf(
    x = wm2, y = list(king), z = list(wm2 %>% iMath("MD", 3)),
    inflationFactor = 255, overlayLimits = c(-0.3, 0.3), verbose = TRUE,
    rotationParams = rp, filename = tempfile()
  )

  fn <- getANTsRData("surf")
  img <- antsImageRead(fn) # see antsSurf on github for data
  wm <- thresholdImage(img, 3, 4)
  wm <- thresholdImage(img, 3, 4) %>% iMath("FillHoles")
  wm <- wm %>% thresholdImage(1, 2)
  wms <- smoothImage(wm, 1.0)
  wmk <- weingartenImageCurvature(wms, 1.5, 0)
  # will display to screen
  antsrSurf(
    x = wm, y = list(wmk %>% smoothImage(1)), z = list(wm %>% iMath("MD", 1)),
    rotationParams = c(270, 0, 90), overlayLimits = c(-0.4, 0.4)
  )
}
```

```

)
# surface and two overlays
blob <- antsImageRead(getANTsRData("blob"))
blob[1:266, 1:266, 1:100] <- 0
z <- list(wm %>% iMath("MD", 1), blob %>% smoothImage(1))
antsrSurf(
  x = wm, y = list(wmk %>% smoothImage(1), blob), z = z,
  colormap = c("jet", "blue"), alpha = c(1, 0.5, 1),
  rotationParams = c(270, 0, 90),
  overlayLimits = list(c(-0.4, 0.4), c(0.9, 1.001))
)
# separate pos and neg curvature
y <- list(
  thresholdImage(wmk, 0.00, Inf),
  thresholdImage(wmk, -100, -0.00)
)
z <- list(
  y[[1]] %>% iMath("MD", 1),
  y[[2]] %>% iMath("MD", 1)
)
antsrSurf(
  x = wm, y = y, z = z, smoothingSigma = 0.5, alpha = c(1, 1, 1),
  colormap = c("red", "blue"),
  inflationFactor = 155, overlayLimits = list(c(0.5, 1.0001), c(0.5, 1.0001)),
  verbose = TRUE, rotationParams = rp[1, ]
)
}

## End(Not run)

```

antsrTransform-class *An S4 class for an antsrTransform*

Description

C++ type used to represent an ITK image transform.

Usage

```

## S4 method for signature 'antsrTransform'
show(object)

## S4 method for signature 'antsrTransform'
initialize(
  .Object,
  precision = "float",
  dimension = 3,
  type = "AffineTransform",
  parameters = NA
)

```


Arguments

object	input object to convert
.Object	input object to convert
precision	string e.g. "float" or "double"
dimension	dimensionality of the transform (2,3,or 4)
type	type of transform' etc noting that short is not supported
parameters	transformation parameters to send to setAntsrTransformParameters

Slots

dimension	usually 2 or 3 but can be 4
precision	math precision is float or double'
type	The type of transform: usually one of AffineTransform, CenteredAffineTransform, CenteredEuler3DTransform, CenteredRigid2DTransform, CenteredSimilarity2DTransform, Euler2DTransform, Euler3DTransform, QuaternionRigidTransform, Rigid2DTransform, Similarity2DTransform, Similarity3DTransform
pointer	to the memory location of the itk object

antsrTransformFromDisplacementField
antsrTransformFromDisplacementField

Description

Convert deformation field (multiChannel image) to antsrTransform

Usage

```
antsrTransformFromDisplacementField(field)
```

Arguments

field	deformation field (multiChannel image)
-------	--

Value

antsrTransform'

antsrVersions

antsrVersions

Description

A data frame defining the git tag for the current ANTsR version. One can also see antsVersions from ANTsRCore to get the git tag for those dependencies.

Format

A data frame listing the following variables.

Dependency Name of software dependency.

GitTag The git tag. This can also be used to trace other dependencies, e.g. the ITK version used by the current ANTs version.

References

<https://github.com/ANTsX/ANTs>

Examples

```
data(antsrVersions)
```

antsrVol

Render an image volume with or without overlay.

Description

Use a system call to antsVol in order to render an image. This requires antsVol to be in the environment as well as ConvertScalarImageToRGB, both available in Advanced Normalization Tools software on github.

Usage

```
antsrVol(
  x,
  y,
  quantlimits = c(0.1, 0.9),
  colormap = "jet",
  rotationParams = c(270, 0, 270),
  overlayLimits = NA,
  magnificationFactor = 1,
  intensityTruncation = c(0, 1),
  filename = NA,
  antspath = NA,
  verbose = FALSE
)
```

Arguments

<code>x</code>	input antsImage defining the image to render
<code>y</code>	input antsImage list defining the function to render on the image. these image(s) should be in the same space.
<code>quantlimits</code>	lower and upper quantile limits for overlay
<code>colormap</code>	character, one of: grey, red, green, blue, copper, jet, hsv, spring, summer, autumn, winter, hot, cool, overunder, custom
<code>rotationParams</code>	3 Rotation angles expressed in degrees or a matrix of rotation parameters that will be applied in sequence.
<code>overlayLimits</code>	absolute lower and upper limits for functional overlay. this parameter will override quantlimits. Currently, this will set levels above overlayLimits[2] to overlayLimits[2].
<code>magnificationFactor</code>	zooms in on image during rendering
<code>intensityTruncation</code>	lower and upper quantiles at which to truncate intensity
<code>filename</code>	prefix filename for output pngs
<code>antspath</code>	pass the ANTSPATH here otherwise we try to detect it from environment
<code>verbose</code>	prints the command used to call antsVol

Value

no output

Author(s)

Avants BB, Tustison NJ

Examples

```
## Not run:
ch2i <- antsImageRead(getANTsRData("mni"))
ch2seg <- thresholdImage(ch2i, "Otsu", 3)
wm <- thresholdImage(ch2seg, 3, 3)
king <- weingartenImageCurvature(ch2i, 1.5) %>% smoothImage(1)
ap <- path.expand("~/code/ants-src/bin/")
rp0 <- matrix(c(90, 180, 90), ncol = 3)
rp2 <- matrix(c(0, 0, 0), ncol = 3)
rp3 <- matrix(c(270, 90, 90), ncol = 3)
rp <- rbind(rp0, rp2, rp3) # pass these as rotationParams
antsrVol(wm, list(king), quantlimits = c(0.01, 0.99))

## End(Not run)
```

antsSetPixels	<i>Set a pixel value at an index</i>
---------------	--------------------------------------

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	<i>Perform spatial ICA on fMRI bold data.</i>
--------------------	---

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 = NULL,
  numberOfICComponents = 20,
  normalizeComponentImages = TRUE,
  verbose = FALSE
)
```

Arguments

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

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(2017)
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)
maskImage <- getAverageOfTimeSeries(boldImages[[1]]) * 0 + 1
icaResults <- antsSpatialICAfMRI(boldImages, 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))
arr <- as.array(img)
testthat::expect_error(
  antsTransformPhysicalPointToIndex(arr, c(2, 2)), "antsImage"
)
testthat::expect_error(
  antsTransformPhysicalPointToIndex(img, c("2", 2)), "point must be"
)
testthat::expect_error(
  antsTransformPhysicalPointToIndex(img, c(2, 2, 2)), "matrix must be of"
)
```

antsVersions	<i>antsVersions</i>
--------------	---------------------

Description

A data frame defining the git tag for the current ANTsRCore version of ANTs.

Format

A data frame listing the following variables.

Dependency Name of software dependency.

GitTag The git tag. This can also be used to trace other dependencies, e.g. the ITK version used by the current ANTs version.

References

<https://github.com/ANTsX/ANTs>

`apply`*Extension of apply to allow input of antsImage*

Description

Returns an antsImage after applying function.

Usage

```
apply(X, MARGIN, FUN, ...)  
  
## Default S3 method:  
apply(X, MARGIN, FUN, ...)  
  
## S3 method for class 'antsImage'  
apply(X, MARGIN, FUN, ...)
```

Arguments

X	an antsImage
MARGIN	a vector giving the subscripts which the function will be applied over.
FUN	the function to be applied.
...	optional arguments to FUN

Value

output is antsImage of lower dimension than input

Author(s)

Duda JT

Examples

```
img <- makeImage(c(4, 4, 4), rnorm(4 * 4 * 4))  
img2 <- apply(img, c(1, 2), mean)  
is.antsImage(img2)
```

apply_simlr_matrices *Apply SIMLR projection matrices with r01.1-style versioning*

Description

Automatically handles name collisions by appending versioned suffixes:

- No collision -> petPC1, petPC2, ...
- Collision -> petPCr01.1, petPCr01.2, ...
- Next round -> petPCr02.1, petPCr02.2, ...

Usage

```
apply_simlr_matrices(
  existing_df,
  simlr_v,
  n_limit = NULL,
  version_prefix = "r",
  verbose = FALSE
)
```

Arguments

existing_df	Data frame (samples in rows, features in columns)
simlr_v	Named list of weight matrices (rownames = original features)
n_limit	Optional: keep only first n_limit components per block
version_prefix	Prefix before version number (default: "r")
verbose	Print messages when versioning occurs

Value

List with extended_df and new_colnames

Examples

```
set.seed(123)
df <- data.frame(petPC1 = rnorm(5), petPC2 = rnorm(5), age = 1:5)
W <- matrix(rnorm(6), 2, 3,
            dimnames = list(c("petPC1", "petPC2"), c("PC1", "PC2", "PC3")))
weights <- list(pet = W)

# First application
res1 <- apply_simlr_matrices(df, weights, verbose = TRUE)
res1$new_colnames
# [1] "petPCr01.1" "petPCr01.2" "petPCr01.3"

# Second application
```

```

res2 <- apply_simlr_matrices(res1$extended_df, weights)
res2$new_colnames
# [1] "petPCr02.1" "petPCr02.2" "petPCr02.3"

# No collision example
clean <- df[, "age", drop = FALSE]
apply_simlr_matrices(clean, weights)$new_colnames
# [1] "petPC1" "petPC2" "petPC3"

```

```
apply_simlr_matrices_dtfix
```

Apply simlr matrices to an existing data frame and combine the results with DTI naming fix.

Description

This function takes a list of matrices, applies each matrix via matrix multiplication to an existing data frame, and combines the resulting projections with the original data frame. Handles an as yet uncharacterized issue with column names in prior versions of data processing.

Usage

```

apply_simlr_matrices_dtfix(
  existing_df,
  matrices_list,
  n_limit = NULL,
  robust = FALSE,
  center = FALSE,
  scale = FALSE,
  verbose = FALSE
)

```

Arguments

<code>existing_df</code>	An existing data frame to which the matrices will be applied.
<code>matrices_list</code>	A list of matrices read from CSV files.
<code>n_limit</code>	NULL or integer that can limit the number of projections
<code>robust</code>	boolean
<code>center</code>	boolean center the data before applying
<code>scale</code>	boolean scale the data before applying
<code>verbose</code>	boolean

Value

A list including (entry one) data frame with the original data frame combined with the projections (entry two) the new column names

apply_transform_matrix	<i>Apply a learned transform to a new matrix</i>
------------------------	--

Description

Applies the same centering, scaling, and normalization learned from a training matrix.

Usage

```
apply_transform_matrix(Xnew, params)
```

Arguments

Xnew	New data matrix to transform.
params	Parameters returned by transform_matrix().

Value

Transformed matrix using the learned parameters.

applyAntsrTransform	<i>applyAntsrTransform</i>
---------------------	----------------------------

Description

Apply transform to point, vector or antsImage data

Usage

```
applyAntsrTransform(transform, data, dataType = "point", reference = NULL, ...)
```

Arguments

transform	antsrTransform
data	data to transform
dataType	data type for non-antsImage data. Either "point" or "vector"
reference	target space for transforming an antsImage
...	options passed to ‘applyAntsrTransformToImage’

Value

transformed data

Examples

```
tx <- createAntsrTransform(dimension = 2, precision = "float", type = "AffineTransform")
params <- getAntsrTransformParameters(tx)
setAntsrTransformParameters(tx, params * 2)
pt2 <- applyAntsrTransform(tx, c(1, 2, 3))
```

```
applyAntsrTransformToImage
      applyAntsrTransformToImage
```

Description

Apply transform to spatial point

Usage

```
applyAntsrTransformToImage(
  transform,
  image,
  reference,
  interpolation = "linear"
)
```

Arguments

transform	antsrTransform
image	antsImage to transform
reference	antImage giving the reference output space
interpolation	type of interpolator to use

Value

antsImage

Examples

```
img <- antsImageRead(getANTSRData("r16"))
tx <- new("antsrTransform", precision = "float", type = "AffineTransform", dimension = 2)
setAntsrTransformParameters(tx, c(0.9, 0, 0, 1.1, 10, 11))
img2 <- applyAntsrTransformToImage(tx, img, img)
# plot(img, img2)
```

```

applyAntsrTransformToPoint
      applyAntsrTransformToPoint

```

Description

Apply transform to spatial point

Usage

```
applyAntsrTransformToPoint(transform, points)
```

Arguments

transform	antsrTransform
points	a matrix which each row is a spatial point

Value

array of coordinates

Examples

```

tx <- new("antsrTransform")
params <- getAntsrTransformParameters(tx)
setAntsrTransformParameters(tx, params * 2)
pt2 <- applyAntsrTransformToPoint(tx, c(1, 2, 3))

```

```

applyAntsrTransformToVector
      applyAntsrTransformToVector

```

Description

Apply transform to spatial vector

Usage

```
applyAntsrTransformToVector(transform, vectors)
```

Arguments

transform	antsrTransform
vectors	a matrix where each row is a vector to transform

Value

array of coordinates

Examples

```
transform <- new("antsrTransform",
  precision = "float",
  type = "AffineTransform", dimension = 2
)
vec2 <- applyAntsrTransformToVector(transform, c(1, 2, 3))
```

as.antsImage	<i>Convert Objects into antsImage objects</i>
--------------	---

Description

Convert types to antsImage

Usage

```
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)))),
  direction = diag(length(dim(object))),
  components = FALSE,
  reference = NULL,
  ...
)

## 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)))),
  direction = diag(length(dim(object))),
  components = FALSE,
  reference = NULL
)

## 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)))),
    direction = diag(length(dim(object))),
    components = FALSE,
    reference = NULL
  )

```

Arguments

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)
direction	numeric matrix matching image dimensionality eg diag(2)
components	number of components per voxel, e.g. 1
reference	optional reference antsImage providing all header info
...	Extra named arguments passed to FUN

Examples

```

arr <- array(rnorm(10^3), dim = rep(10, 3))
img <- as.antsImage(arr)
i2 <- as.antsImage(arr, reference = img)

```

as.antsMatrix	<i>Coerce Object to as.antsMatrix</i>
---------------	---------------------------------------

Description

convert types to an antsMatrix

Usage

```

as.antsMatrix(object, elementtype = "float", ...)

## 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
elementtype	e.g. "float" or "double"
...	other parameters

Examples

```
as.antsMatrix(matrix(rnorm(10), nrow = 2))
mat <- matrix(rnorm(10), nrow = 2)
df <- as.data.frame(mat)
as.antsMatrix(df)
as.antsMatrix(as.list(df))
```

as.array

*Coerce antsImage objects to array***Description**

Converts antsImage, antsImage object to different data types

Usage

```
## S4 method for signature 'antsImage'
dim(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())
)

## S3 method for class '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())
)

## S3 method for class 'antsImage'
as.array(
  x,
  ...,
  mask = logical(),
  region = new("antsRegion", index = integer(), size = integer())
)

## S3 method for class 'antsImage'
as.character(
  x,
  ...,
  mask = logical(),
  region = new("antsRegion", index = integer(), size = integer())
)

## S3 method for class 'antsImage'
as.factor(
  x,
  ...,
  mask = logical(),
  region = new("antsRegion", index = integer(), size = integer())
)

## S3 method for class 'antsImage'
factor(
  x,
  ...,
  mask = logical(),
  region = new("antsRegion", index = integer(), size = integer())
)

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

## S3 method for class 'antsMatrix'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S4 method for signature 'antsMatrix'
```

```
as.matrix(x, ...)

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

Arguments

x	object of class <code>antsImage</code> or <code>antsMatrix</code>
mask	a logical vector/array or binary <code>antsImage</code> object
region	a <code>antsRegion</code> object
...	additional arguments passed to functions
row.names	NULL or a character vector giving the row names for the data frame.
optional	passed to <code>as.data.frame</code>

Examples

```
outimg <- makeImage(c(2, 10), rnorm(20))
as.numeric(outimg)
as.numeric(outimg, mask = outimg > 1)
testthat::expect_error(as.numeric(outimg, mask = outimg))

outimg <- makeImage(c(2, 10), rnorm(20))
as.matrix(outimg)
as.matrix(outimg, mask = outimg > 1)
testthat::expect_error(as.matrix(outimg, mask = outimg))
outimg <- makeImage(c(2, 10, 2), rnorm(40))
testthat::expect_error(as.matrix(outimg))
outimg <- makeImage(c(2, 10), rnorm(20))
as.matrix(outimg)
as.matrix(outimg, mask = outimg > 1)
outimg <- makeImage(c(2, 10, 2), rnorm(40))
testthat::expect_error(as.matrix(outimg))
img <- antsImageRead(getANTsRData("r16"))
img[img > 5] <- 0
sort(unique(as.character(img)))
factor(img)
table(img, img)
```

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 = NULL,
  tc = NA,
  nuisance = NA,
  method = "regression",
  ...
)
```

Arguments

<code>asl</code>	input asl image
<code>mask</code>	in which to calculate perfusion
<code>tc</code>	vector indicating which images are tagged and which are controls. Strongly recommended if using regression or bayesian methods.
<code>nuisance</code>	nuisance covariates to include in regression
<code>method</code>	method to use for computing average. One of <code>sincSubtract</code> , <code>simpleSubtract</code> , <code>cubicSubtract</code> , <code>surroundSubtract</code> , <code>regression</code> , or <code>bayesian</code> . See Details.
<code>...</code>	additional parameters to pass to ASL averaging functions. See Details.

Details

Two major types of methods are available for ASL signal averaging:

Subtraction : All the subtraction methods are based on subtracting the tag from control images and averaging the result. `simple` subtracts adjacent tag and control images. The other methods use interpolation to obtain a subtracted time-series. `Sinc` subtraction may be marginally more accurate than cubic interpolation, but takes much longer. `Surround` subtraction uses linear interpolation and is fast.

Regression : Regression regresses the time-series on a vector of alternating tag-control dummy variables. Regression can incorporate nuisance covariates. Bayesian regression incorporates regularization in the regression to encourage all voxels of the same tissue type to have similar perfusion values.

For `bayesian`, two more arguments are required:

segmentation : a segmentation image

tissuelist : a list of tissue probability images

These would be as output from `atropos`; see Examples for a sample usage.

Author(s)

Kandel BM, Avants BB

Examples

```

set.seed(1)
nvox <- 8 * 8 * 8 * 10
dims <- c(8, 8, 8, 10)
voxvals <- array(rnorm(nvox) + 500, dim = dims)
asl <- makeImage(dims, voxvals)
tc <- rep(c(-0.5, 0.5), dims[4] / 2)
avg <- aslAveraging(asl, tc = tc)
testthat::expect_equal(mean(avg), 0.00739, tolerance = .1)
slice <- extractSlice(asl, 4, 4)
mask <- getMask(slice)
seg <- atropos(d = 3, a = slice, x = mask, i = "kmeans[6]", m = "[0.0,1x1x1]")
perfSurr <- aslAveraging(asl, mask = NULL, method = "surroundSubtract")
bayesAvg <- aslAveraging(asl,
  tc = tc, method = "bayesian",
  segmentation = seg$segmentation, tissuelist = seg$probabilityimages
)

```

aslCensoring

*Censor bad volumes from ASL data.***Description**

Censor bad volumes from ASL data.

Usage

```

aslCensoring(
  asl,
  mask = NULL,
  nuis = NA,
  method = "outlier",
  reject.pairs = FALSE,
  ...
)

```

Arguments

asl	input asl image
mask	mask for calculating perfusion
nuis	fixed nuisance parameters
method	one of 'outlier', 'robust', or 'scor'. See Details.
reject.pairs	whether to reject only tag-control pairs of images, as opposed to single images. Rejecting pairs of images is necessary for non-regression-based ASL averaging methods.
...	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:

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.

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).

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

```
set.seed(1)
nvox <- 5 * 5 * 5 * 30
dims <- c(5, 5, 5, 30)
voxvals <- array(rnorm(nvox) + 500, dim = dims)
voxvals[, , , 5] <- voxvals[, , , 5] + 600
asl <- makeImage(dims, voxvals)
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,
  covariates = NA,
  selectionthresh = 0.1,
  maxnoise_preds = 2:12,
  polydegree = "loess",
  crossvalidationgroups = 4,
  scalemat = F,
  noise_poolfun = max,
  usecompcor = F,
  verbose = F
)
```

Arguments

<code>boldmatrix</code>	input bold matrix
<code>targety</code>	target to predict
<code>covariates</code>	motion or other parameters / nuisance variables
<code>selectionthresh</code>	e.g. 0.1 take 10 percent worst variables for noise estimation
<code>maxnoise_preds</code>	integer search range e.g 1:10
<code>polydegree</code>	eg 4 for polynomial nuisance variables or 'loess'
<code>crossvalidationgroups</code>	prior defined or integer valued
<code>scalemat</code>	boolean
<code>noise_poolfun</code>	function to help select noise pool e.g. max
<code>usecompcor</code>	boolean
<code>verbose</code>	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,
  covariates = dv, selectionthresh = 0.1,
  maxnoisepreds = c(1:2), polydegree = 2, crossvalidationgroups = 2
)
testthat::expect_equal(dnz$R2atBestN, 7, tolerance = 0.5)

## 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)

# default mode network example

if (!exists("bold")) {
  bold <- antsImageRead(getANTsRData("rsbold"))
  meanbold <- getAverageOfTimeSeries(bold)
  boldmask <- getMask(meanbold)
  # map to mni
  mni <- antsImageRead(getANTsRData("mni"))
  mniaal <- antsImageRead(getANTsRData("mnia"))
  mymap <- antsRegistration(meanbold * boldmask, mni,
    typeofTransform = "SyNBold",
    verbose = 1
  )
  aaling <- antsApplyTransforms(meanbold, mniaal, mymap$fwdtransforms,
    interpolator = "NearestNeighbor"
  )
}

```

```

data("aal", package = "ANTsR")
timeselect <- 10:dim(bold)[4]
if (!exists("moco")) {
  moco <- antsMotionCalculation(bold, boldmask)
}
sbold <- smoothImage(moco$moco_img, 3.0)
antsImageWrite(boldmask, "boldmask.nii.gz")
antsImageWrite(meanbold, "boldmean.nii.gz")
antsImageWrite(aalimg, "boldaal.nii.gz")
boldmask <- boldmask * thresholdImage(aalimg, 1, Inf)
}
postcing <- aal$label_num[grep("Cingulum_Post", aal$label_name)]
postCingMask <- maskImage(boldmask, aalimg,
  level = as.numeric(postcing), binarize = T
)
mpostCingMask <- antsImageClone(postCingMask) * 0
mpostCingMask[postCingMask == 0] <- 1
boldmat <- timeseries2matrix(sbold, boldmask * mpostCingMask)
boldmat <- boldmat[timeselect, ]
boldmat <- frequencyFilterfMRI(boldmat, tr = antsGetSpacing(bold)[4], opt = "trig")
dmnvec <- (timeseries2matrix(sbold, postCingMask)[timeselect, ])
dmnvec <- rowMeans(
  frequencyFilterfMRI(dmnvec, tr = antsGetSpacing(bold)[4], opt = "trig")
)
dmnmat <- matrix(dmnvec, ncol = 1)
mocpar <- moco$moco_params[timeselect, 3:14]
dnz <- aslDenoiseR(boldmat, dmnvec,
  covariates = mocpar, selectionthresh = 0.2,
  maxnoisepreds = c(2:10), polydegree = "loess",
  crossvalidationgroups = 8
)
boldmat <- timeseries2matrix(sbold, boldmask)
boldmat <- boldmat[timeselect, ]
boldmat <- frequencyFilterfMRI(boldmat, tr = antsGetSpacing(bold)[4], opt = "trig")
mdl <- bigLMStats(lm(boldmat ~ dmnvec + dnz$covariates + dnz$noiseu), 0.001)
betas <- mdl$beta.t[1, ]
betaImg <- makeImage(boldmask, betas)
antsImageWrite(betaImg, "dmnBetas.nii.gz")
# this should give default mode network around beta = 12

## 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 = NULL,
  checkmeansignal = 100,
  moco_results = NULL,
  regweights = NULL,
  useDenoiser = NULL,
  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 * 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)
aslmats <- timeseries2matrix(asl, aslmask)
for (i in 1:10) aslmats[, i * 2] <- aslmats[, i * 2] * 2
asl <- matrix2timeseries(asl, aslmats, aslmats)
# NOT WORKING
## Not run:
pcasl.processing <- aslPerfusion(asl, moreaccurate = 1, dorobust = 0)
testthat::expect_equal(mean(pcasl.processing$m1), 62.2115522470984)
pcasl.processing <- aslPerfusion(asl, moreaccurate = 1, ncompcor = 2)
# allow some rejection
pcasl.processing <- aslPerfusion(asl, moreaccurate = 1, dorobust = 0.925)

## End(Not run)
```

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,
verbose = FALSE,
use_random_seed = FALSE,
...
)

```

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.
verbose	boolean
use_random_seed	Initialize internal random number generator with a random seed. Otherwise, initialize with a constant seed number. If this is TRUE, the results are not fully reproducible.
...	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(
  a = img, m = "[0.2,1x1]",
  c = "[2,0]", i = "kmeans[3]", x = mask
)

# Use probabilities from k-means seg as priors

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

```

ba_svd

Robust SVD with Automatic rsvd Fallback

Description

This function attempts a standard SVD using base R's `svd()`. If that fails (typically due to ill-conditioning or non-convergence), it automatically falls back to a randomized SVD via the `rsvd` package.

Usage

```

ba_svd(
  x,
  nu = min(nrow(x), ncol(x)),
  nv = min(nrow(x), ncol(x)),
  dividebymax = FALSE,
  NA2Noise = TRUE
)

```

Arguments

x	A numeric matrix.
nu	Number of left singular vectors to compute. Default: <code>min(nrow(x), ncol(x))</code> .
nv	Number of right singular vectors to compute. Default: <code>min(nrow(x), ncol(x))</code> .
dividebymax	Logical. If TRUE, scale x by its max absolute value before SVD. Default: FALSE.
NA2Noise	boolean replaces NA values with small noise values

Details

Optionally, the matrix can be scaled by its maximum absolute value to improve numerical stability before decomposition.

Value

A list with components `u`, `d`, and `v`, matching the structure of base R's `svd()` output.

Examples

```
mat <- matrix(rnorm(100 * 50), 100, 50)
sv <- ba_svd(mat, nu = 10, nv = 0)
```

backup_simlr	<i>backup simlr implementation</i>
--------------	------------------------------------

Description

backup simlr implementation

Usage

```
backup_simlr(
  data_matrices,
  smoothingMatrices,
  iterations = 500,
  sparsenessQuantiles = NULL,
  positivities = NULL,
  initialUMatrix = NULL,
  mixAlg = c("svd", "ica", "avg", "rrpca-l", "rrpca-s", "pca", "stochastic",
    "newton-schulz", "ica-newton"),
  repeatedMeasures = NA,
  lineSearchRange = c(-500, 500),
  lineSearchTolerance = 1e-12,
  randomSeed = 0,
  constraint = c("nsaflowx0.5x10", "orthox0x1", "Grassmannx0", "Stiefelx0", "none"),
  energyType = c("cca", "regression", "normalized", "ucca", "lowRank",
    "lowRankRegression", "normalized_correlation", "acc", "nc", "dat", "lrr", "reconorm",
    "logcosh", "exp", "kurtosis", "gauss"),
  vmats = NULL,
  connectors = NULL,
  optimizationStyle = c("bidirectional_lookahead", "armijo_gradient", "lookahead",
    "bidirectional_armijo_gradient"),
  scale = c("center", "eigenvalue"),
  expBeta = 0.9,
  jointInitialization = TRUE,
  sparsenessAlg = "soft",
```

```

    orthogonalizeU = FALSE,
    domainMatrices = NULL,
    domainLambdas = NULL,
    sparse_gradient = TRUE,
    verbose = FALSE
)

```

Arguments

<code>data_matrices</code>	A list that contains the named matrices. Note: the optimization will likely perform much more smoothly if the input matrices are each scaled to zero mean unit variance e.g. by the <code>scale</code> function.
<code>smoothingMatrices</code>	list of (sparse) matrices that allow parameter smoothing/regularization. These should be square and same order and size of input matrices.
<code>iterations</code>	number of gradient descent iterations
<code>sparsenessQuantiles</code>	vector of quantiles to control sparseness - higher is sparser
<code>positivities</code>	vector that sets for each matrix if we restrict to positive or negative solution (beta) weights. choices are positive, negative or either as expressed as a string.
<code>initialUMatrix</code>	list of initialization matrix size n by k for each modality. Otherwise, pass a single scalar to control the number of basis functions in which case random initialization occurs. One may also pass a single initialization matrix to be used for all matrices. If this is set to a scalar, or is missing, a random matrix will be used.
<code>mixAlg</code>	'svd', 'ica', 'rrpca-l', 'rrpca-s', 'stochastic', 'pca', 'newton-schulz', 'ica-newton' or 'avg' denotes the algorithm employed when estimating the mixed modality bases
<code>repeatedMeasures</code>	list of repeated measurement identifiers. this will allow estimates of per identifier intercept.
<code>lineSearchRange</code>	lower and upper limit used in <code>optimize</code>
<code>lineSearchTolerance</code>	tolerance used in <code>optimize</code> , will be multiplied by each matrix norm such that it scales appropriately with input data
<code>randomSeed</code>	controls repeatability of ica-based decomposition
<code>constraint</code>	one of none, Grassmann, GrassmannInv or Stiefel
<code>energyType</code>	one of regression, normalized, lowRank, cca or ucca
<code>vmats</code>	optional initial v matrix list
<code>connectors</code>	a list (length of projections or number of modalities) that indicates which modalities should be paired with current modality
<code>optimizationStyle</code>	the simlr optimizer

scale	options to standardize each matrix. e.g. divide by the square root of its number of variables (Westerhuis, Kourti, and MacGregor 1998), divide by the number of variables or center or center and scale or ... (see code). can be a vector which will apply each strategy in order.
expBeta	if greater than zero, use exponential moving average on gradient and mixing (default 0.9).
jointInitialization	boolean for initialization options, default TRUE
sparsenessAlg	NA is default otherwise basic, spmp or orthorank
orthogonalizeU	boolean controlling whether we orthogonalize the U matrices
domainMatrices	matrices containing domain knowledge length of data_matrices with number of columns also equal to each corresponding data matrix
domainLambdas	weights for domain knowledge term length of data_matrices
sparse_gradient	boolean default TRUE
verbose	boolean to control verbosity of output - set to level 2 in order to see more output, specifically the gradient descent parameters.

basal_forebrain	<i>basal_forebrain</i>
-----------------	------------------------

Description

A data frame label numbers and names for basal forebrain regions.

Usage

```
basal_forebrain
```

Format

A data frame with columns: Label, Description, Side.

basicInPaint	<i>Inpaints missing imaging data from boundary data</i>
--------------	---

Description

Smooths data along the boundary into the missing region.

Usage

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

Arguments

<code>img</code>	antsImage to be approximated / painted
<code>paintMask</code>	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.
<code>speedimage</code>	- larger means faster/better
<code>its</code>	- iterations of graddescent
<code>gparam</code>	- 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)
## Not run:
painted <- basicInPaint(fi, mask)

## 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 = FALSE,
  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	<i>Simple bayesian regression function.</i>
------------	---

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)
# testthat::expect_equal(bmdl$beta, c(0.157536274628774, -0.224079937323326))
bmdlNoPrior <- bayesianlm(X2, wt2)
print(priorcoeffs)
print(bmdl$beta)
print(bmdlNoPrior$beta)
## Not run:
fn <- "PEDS012_20131101_pcasl_1.nii.gz"
fn <- getANTsRData("pcasl")
# image available at http://files.figshare.com/1701182/PEDS012\_20131101.zip
asl <- antsImageRead(fn, 4)
tr <- antsGetSpacing(asl)[4]
aslmean <- getAverageOfTimeSeries(asl)
aslmask <- getMask(aslmean, lowThresh = mean(aslmean), cleanup = TRUE)
aslmat <- timeseries2matrix(asl, aslmask)
tc <- as.factor(rep(c("C", "T"), nrow(aslmat) / 2))
dv <- computeDVARs(aslmat)
# do some comparison with a single y, no priors
y <- rowMeans(aslmat)
perfmodel <- lm(y ~ tc + dv) # standard model
t1m <- bigLMStats(perfmodel)
X <- model.matrix(perfmodel)
blm <- bayesianlm(X, y)
print(t1m$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
nmatings <- 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)
  nmatings[[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$beta.p)
for (i in 1:ncol(aslmat))
{
  parammat <- nmatings[[1]][, i]
  for (k in 2:length(nmatings)) {
    parammat <- cbind(parammat, nmatings[[k]][, i])
  }
  locinvcov <- solve(cov(parammat))
  localprior <- (smoothcoeffmat[, i])
  blm <- bayesianlm(X, aslmat[, i], localprior, locinvcov * 1.e4)
  bayesianperfusionloc[i] <- blm$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 $\text{lm}(y \sim x)$ calculates the coefficients, but statistical inference is not provided. This function provides basic statistical inference efficiently.

Usage

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

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)
```

bloodPerfusionSVD	<i>Calculate blood perfusion using deconvolution.</i>
-------------------	---

Description

Implementation of the deconvolution technique of Ostergaard et al. (<https://pubmed.ncbi.nlm.nih.gov/8916023>) for calculating cerebral (or pulmonary) blood flow. Other relevant references include <https://pubmed.ncbi.nlm.nih.gov/162615>, <https://pubmed.ncbi.nlm.nih.gov/8916023>, and <https://pubmed.ncbi.nlm.nih.gov/15332240>.

Usage

```
bloodPerfusionSVD(
  perfusionImage,
  voiMaskImage,
  aifMaskImage,
  thresholdSVD = 0.2,
  deltaTime = 1
)
```

Arguments

perfusionImage	time series (n-D + time) perfusion acquisition.
voiMaskImage	n-D mask image indicating where the cerebral blood flow parameter images are calculated.
aifMaskImage	n-D mask image indicating where the arterial input function is calculated.
thresholdSVD	is used to threshold the smaller elements of the diagonal matrix during the SVD regularization. 0.2 is a common choice (cf. page 571, end of column 2 in https://pubmed.ncbi.nlm.nih.gov/16971140).
deltaTime	time between volumetric acquisitions. We assume a uniform time sampling.

Value

list with the cerebral blood flow image (cbfImage), cerebral blood volume image (cbvImage), mean transit time (mttImage), and arterial input function signal from the image (aifSignal) and the calculated arterial input function concentration (aifConcentration).

Author(s)

Tustison NJ

Examples

```
## Not run:

perfusionFileName <- ""
if (file.exists(perfusionFileName)) {
  perfusionImage <- antsImageRead(
    filename = perfusionFileName,
    dimension = 4, pixeltype = "float"
  )
  voiMaskImage <- antsImageRead(
    filename = voiMaskFileName,
    dimension = 3, pixeltype = "unsigned int"
  )
  aifMaskImage <- antsImageRead(
    filename = aifMaskFileName,
    dimension = 3, pixeltype = "unsigned int"
  )

  deltaTime <- 3.4

  results <- bloodPerfusionSVD(perfusionImage,
    voiMaskImage, aifMaskImage,
    thresholdSVD = 0.2, deltaTime = deltaTime
  )

  antsImageWrite(results$cbfImage, paste0("cbf.nii.gz"))
  antsImageWrite(results$cbvImage, paste0("cpbv.nii.gz"))
  antsImageWrite(results$mttImage, paste0("mtt.nii.gz"))
}

## End(Not run)
```

bold_correlation_matrix
BOLD Correlation Matrix

Description

A data frame with an example square BOLD correlation matrix.

Usage

```
bold_correlation_matrix
```

Format

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

References

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

Examples

```
data(bold_correlation_matrix)
```

boxplot.antsImage	<i>Boxplot of Values in an Image</i>
-------------------	--------------------------------------

Description

Computes the boxplot of values of an image with the option for a mask.

Usage

```
## S3 method for class 'antsImage'  
boxplot(x, ..., mask)
```

Arguments

x	Object of class antsImage
...	Arguments passed to boxplot.default
mask	object to subset the image. If missing, then all values of the image are plotted.

Value

Output of [boxplot](#)

Examples

```
img <- makeImage(c(10, 10), rnorm(100))  
mask <- img > 0  
boxplot(img, mask = mask)
```

buildTemplate	<i>ANTs template building.</i>
---------------	--------------------------------

Description

Iteratively estimate a population shape and intensity average image. This can be computationally intensive and currently is not parallelized. Perhaps better to use official `antsMultivariateTemplateConstruction*` in ANTs. However, this code can be useful for smaller problems/populations.

Usage

```
buildTemplate(
    initialTemplate,
    imgList,
    typeofTransform,
    iterations = 3,
    gradientStep = 0.25,
    blendingWeight = 0.5,
    segList,
    weights,
    verbose = TRUE,
    ...
)
```

Arguments

<code>initialTemplate</code>	fixed image to which we register the population.
<code>imgList</code>	moving image list from which template will be built.
<code>typeofTransform</code>	A linear or non-linear registration type. Mutual information metric by default. See <code>antsRegistration</code> .
<code>iterations</code>	should be greater than 1 less than 10.
<code>gradientStep</code>	speed of template shape update step, less than 1.
<code>blendingWeight</code>	parameter between zero and one, default 0.5; higher values lead to sharper intensity but may also cause artifacts.
<code>segList</code>	segmentations for each target image, this will trigger a joint label fusion call for each iteration and use masks during registration.
<code>weights</code>	numeric vector, length of number of images providing weights
<code>verbose</code>	print diagnostic messages, passed to <code>antsRegistration</code>
<code>...</code>	Additional options to pass to <code>antsRegistration</code>

Value

template `antsImage`

Author(s)

Avants BB

Examples

```
pop <- getANTsRData("population")
avg <- antsAverageImages(pop) # this is in ANTsR
template <- buildTemplate(avg, pop, "SyN", iterations = 1)
```

calculate_simlr_energy

Calculate SiMLR energy for optimization

Description

Computes the energy for SiMLR based on the specified energy type, with partial matching for energy_type. Returns negative values for maximization objectives to align with minimization goals.

Usage

```
calculate_simlr_energy(
  V,
  X,
  U,
  energy_type,
  lambda = 1,
  prior_matrix = NULL,
  verbose = 0
)
```

Arguments

V	Matrix V in the SiMLR decomposition.
X	Data matrix for the modality.
U	Matrix U in the SiMLR decomposition.
energy_type	Character string specifying the energy type. Supports partial matching (e.g., "reg" for "regression", "cca" for "cca" or "acc"). Valid options: regression, reconorm, lowRankRegression, lrr, cca, acc, logcosh, kurtosis, exp, gauss, normalized_correlation, dat.
lambda	Numeric, weight for domain energy (default = 1.0).
prior_matrix	Optional prior matrix for domain energy (used with "dat").
verbose	Integer, verbosity level: 0 (none), 1 (log matched energy_type) (default = 0).

Value

Numeric energy value (negative for maximization objectives).

 calculate_simlr_gradient

Calculate SiMLR Similarity Gradient

Description

This dispatcher computes the gradient for the similarity part of the SiMLR objective, ensuring it is a descent direction for the energy function. Supports partial matching for energy_type.

Usage

```
calculate_simlr_gradient(
  V,
  X,
  U,
  energy_type,
  clipping_threshold = NULL,
  lambda = 1,
  prior_matrix = NULL,
  verbose = 0
)
```

Arguments

V	The current loading matrix [p x k].
X	The data matrix for the modality [n x p].
U	The shared basis matrix [n x k].
energy_type	Character string specifying the similarity objective. Supports partial matching (e.g., "reg" for "regression", "cca" for "cca" or "acc"). Valid options: regression, reconorm, lowRankRegression, lrr, cca, acc, normalized_correlation, nc, logcosh, kurtosis, exp, gauss, dat.
clipping_threshold	Optional numeric value for gradient clipping.
lambda	Numeric, weight for domain energy (default = 1.0).
prior_matrix	Optional matrix of prior weights, same shape as V (used with "dat").
verbose	Integer, verbosity level: 0 (none), 1 (log matched energy_type) (default = 0).

Value

A matrix [p x k] representing the descent direction.

cerebellum	<i>cerebellum</i>
------------	-------------------

Description

cerebellum

cerebellum_labels	<i>cerebellum_labels</i>
-------------------	--------------------------

Description

A data frame label numbers and names for cerebellum regions.

Usage

cerebellum_labels

Format

A data frame with columns: Label, Description.

check_ants-methods	<i>Check if antsImage or read in</i>
--------------------	--------------------------------------

Description

Simple check to see if input is character, list, nifti, or class antsImage

Usage

```
check_ants(x, ...)  
  
## S4 method for signature 'antsImage'  
check_ants(x, ...)  
  
## S4 method for signature 'ANY'  
check_ants(x, ...)  
  
## S4 method for signature 'character'  
check_ants(x, ...)  
  
## S4 method for signature 'list'  
check_ants(x, ...)
```

Arguments

x character path of image or an object of class `antsImage`
 ... arguments passed to other methods, namely [antsImageRead](#)

Value

`antsImage` object

Author(s)

John Muschelli <muschellij2@gmail.com>

See Also

[antsImageRead](#)

check_zero_variance	<i>Check a matrix for zero-variance rows and columns</i>
---------------------	--

Description

This function checks a numeric matrix for rows or columns with zero variance. It reports how many and which ones are affected, using either their names (if present) or indices.

Usage

```
check_zero_variance(mat)
```

Arguments

mat A numeric matrix.

Details

Zero variance means all values in a row or column are identical (after removing NAs).

Value

Boolean checking if any zero-variance rows or columns found.

Examples

```
m <- matrix(c(1, 1, 1,
              2, 3, 4,
              5, 6, 7),
            nrow = 3, byrow = TRUE)
rownames(m) <- c("rowA", "rowB", "rowC")
colnames(m) <- c("colX", "colY", "colZ")
check_zero_variance(m)
```

<code>cit168_brainstem</code>	<i>cit168_brainstem</i>
-------------------------------	-------------------------

Description

A data frame label numbers and names for the CIT168 brainstem atlas.

Usage

`cit168_brainstem`

Format

A data frame with columns: Label, Description, Side.

<code>cit168_reinf_learn</code>	<i>cit168_reinf_learn</i>
---------------------------------	---------------------------

Description

A data frame label numbers and names for the CIT168 reinforcement learning atlas.

Usage

`cit168_reinf_learn`

Format

A data frame with columns: Label, Description.

<code>CIT168_Reinf_Learn_v1_label_descriptions_pad</code>	<i>CIT168 reinforcement learning atlas label descriptions</i>
---	---

Description

CIT168 reinforcement learning atlas label descriptions

CIT168_T1w_700um_pad_adni_brainstem
<i>CIT168 T1w 700um pad adni brainstem</i>

Description

CIT168 T1w 700um pad adni brainstem

clip_gradient_by_quantile
<i>Clip a Gradient Matrix by a Quantile of its Values</i>

Description

This function provides an adaptive method for controlling gradient magnitudes. It calculates a threshold based on a specified quantile of the absolute values within the gradient matrix itself. Any value exceeding this dynamic threshold is "clipped" or shrunk back to the threshold, preserving its original sign.

Usage

```
clip_gradient_by_quantile(gradient, quantile = 0.8)
```

Arguments

gradient	The gradient matrix to be clipped.
quantile	The quantile to use for determining the clipping threshold. For example, a value of ‘0.98’ means that any gradient value larger in magnitude than the 98th percentile of all absolute gradient values will be clipped. Must be between 0 and 1.

Details

This is more robust than a fixed threshold as it automatically adapts to the scale of the gradients at each optimization step.

Value

A new gradient matrix with extreme values clipped.

Examples

```
# Create a gradient with some large outlier values
set.seed(123)
grad_matrix <- matrix(rnorm(100, mean = 0, sd = 1), 10, 10)
grad_matrix[1, 1] <- 10 # Large positive outlier
grad_matrix[5, 5] <- -12 # Large negative outlier

# Clip at the 80th percentile. This will tame the outliers.
clipped_grad <- clip_gradient_by_quantile(grad_matrix, quantile = 0.80)

cat("Original Gradient Range:\n")
print(range(grad_matrix))

cat("\n80th Percentile Threshold:\n")
# The threshold will be the 80th percentile of the absolute values
print(quantile(abs(grad_matrix), probs = 0.80))

cat("\nClipped Gradient Range:\n")
# The new range will be capped at the threshold
print(range(clipped_grad))
```

clip_gradient_norm	<i>Clip a Gradient Matrix by its Frobenius Norm</i>
--------------------	---

Description

This function prevents exploding gradients by rescaling any gradient whose Frobenius norm exceeds a specified threshold. The direction of the gradient is preserved.

This function prevents exploding gradients by rescaling any gradient whose Frobenius norm exceeds a specified threshold. The direction of the gradient is preserved.

Usage

```
clip_gradient_norm(gradient, threshold = 1)
```

```
clip_gradient_norm(gradient, threshold = 1)
```

Arguments

gradient	The gradient matrix to be clipped.
threshold	The maximum allowed Frobenius norm for the gradient.

Value

A gradient matrix whose Frobenius norm is guaranteed to be less than or equal to the threshold.

A gradient matrix whose Frobenius norm is guaranteed to be less than or equal to the threshold.

clusterTimeSeries	<i>Split time series image into k distinct images</i>
-------------------	---

Description

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

Usage

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

Arguments

mat	input time series matrix
krange	k cluster range to explore
nsyddims	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)
```

coerce_mask

Simple Mask Maker

Description

Creates a logical array from an antsImage object

Usage

```
coerce_mask(x, error = TRUE)
```

Arguments

x	antsImage object
error	should the function error if the mask as not binary?

Value

Array, logical if all values within 0, 1, NA, and NaN

Examples

```
img <- makeImage(c(10, 10), rnorm(100))
mask <- img > 0
coerce_mask(mask, error = TRUE)
coerce_mask(mask + 1, error = FALSE)
testthat::expect_error(coerce_mask(mask + 1, error = TRUE))
coerce_mask(as.array(mask) > 0, error = FALSE)
```

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 = FALSE
)
```

Arguments

<code>inmat</code>	Input predictor matrix.
<code>target</code>	Target outcome matrix.
<code>globalpredictors</code>	Global predictors of size <code>nrow(inmat)</code> by <code>n</code> , where <code>n</code> is the number of global predictors.
<code>maxpreds</code>	Maximum number of predictors to output.
<code>localpredictors</code>	Local predictor array of size <code>nrow(inmat)</code> by <code>ncol(inmat)</code> by <code>m</code> , where <code>m</code> is the number of local predictors.
<code>method</code>	Method of selecting noisy voxels. One of 'svd' or 'cv'. See Details.
<code>k</code>	Number of cross-validation folds.
<code>covariates</code>	Covariates to be considered when assessing prediction of <code>target</code> .
<code>ordered</code>	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

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 = NULL,
  randomSamples = 1,
  returnv = FALSE,
  returnhighvarmat = FALSE,
  returnhighvarmatinds = FALSE,
  highvarmatinds = NA,
  scale = TRUE
)
```

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
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
scale	scale the matrix of high variance voxels, default FALSE. note that you may get slightly different results by scaling the input matrix before passing into this function.

Value

dataframe of nuisance predictors is output

Author(s)

Avants BB

Examples

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

composeAntsrTransforms

composeAntsrTransforms

Description

compose multiple transforms

Usage

```
composeAntsrTransforms(transformList)
```

Arguments

`transformList` a list of antsrTransforms in the reverse order they should be applied

Value

antsrTransform of type "CompositeTransform"

Examples

```
tx <- new("antsrTransform", precision = "float", type = "AffineTransform", dimension = 2)
setAntsrTransformParameters(tx, c(0, -1, 1, 0, 0, 0))
tx2 <- new("antsrTransform", precision = "float", type = "AffineTransform", dimension = 2)
setAntsrTransformParameters(tx2, c(0, -1, 1, 0, 0, 0))
tx3 <- composeAntsrTransforms(list(tx, tx2))
```

composeDisplacementFields	
	<i>composeDisplacementFields</i>

Description

Compose displacement fields.

Usage

```
composeDisplacementFields(displacementField, warpingField)
```

Arguments

displacementField	
	displacement field.
warpingField	warping field.

Value

composite displacement field

Author(s)

NJ Tustison

composeTransformsToField	
	<i>Compose arbitrary transform list into a deformation field.</i>

Description

A deformation field can capture any domain to domain correspondence that we can represent in ANTsR. This function will map a list of transforms into a single deformation field object.

Usage

```
composeTransformsToField(image, transforms)
```

Arguments

image	input image defines the transformation domain
transforms	list of transform filenames or antrTransforms

Value

field deformation object is output

Author(s)

Avants BB, Duda JT

Examples

```
fi <- antsImageRead(getANTsRData("r16"))
mi <- antsImageRead(getANTsRData("r64"))
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"))
compfield <- composeTransformsToField(fi, mytx$fwd)
atx <- antsrTransformFromDisplacementField(compfield)
wrped <- applyAntsrTransformToImage(atx, mi, fi)
```

compute_communalities *Compute Communalities from Factor Scores*

Description

This function calculates the communality for each variable in the dataset, defined as the proportion of variance explained by the factor scores (R^2 from multiple linear regression of each variable on the scores). It provides an empirical estimate of how much each variable's variance is accounted for by the common factors. The function assumes the data is centered or standardized as appropriate for factor analysis.

Usage

```
compute_communalities(data, scores)
```

Arguments

data	Numeric matrix or data.frame with n rows (observations) and p columns (variables).
scores	Numeric matrix of factor scores with n rows and k columns (factors).

Details

For each variable, a linear model is fit: `lm(data[, i] ~ scores)`, and the R^2 is extracted. This aligns with model-based communalities in factor analysis for large samples but may slightly overestimate due to in-sample fitting. For efficiency with large p, consider matrix-based alternatives, but this loop-based approach is simple and leverages base R.

Value

A named numeric vector of length p containing the communalities for each variable, named by the column names of data (if available).

Examples

```
## Not run:
# Synthetic data example
set.seed(42)
n <- 100; p <- 5; k <- 2
scores <- matrix(rnorm(n * k), n, k)
loadings <- matrix(runif(p * k, 0.5, 0.8), p, k)
data <- scores %*% t(loadings) + matrix(rnorm(n * p, sd = 0.5), n, p)
colnames(data) <- paste0("Var", 1:p)
comm <- compute_communalities(data, scores)
print(comm)

## End(Not run)
```

compute_variance_explained_per_factor

Compute Variance Explained per Factor

Description

This function calculates the variance explained by each factor in a factor analysis model.

Usage

```
compute_variance_explained_per_factor(data, scores, method = "PAF")
```

Arguments

data	A numeric matrix or data frame of observed variables ($n \times p$).
scores	A numeric matrix of factor scores ($n \times k$).
method	A string indicating the factor analysis method (e.g., "PAF").

Value

A data frame with the variance explained by each factor.

Examples

```
# data(mtcars)
# fa_result <- factanal(mtcars, factors = 2)
# scores <- fa_result$scores
# compute_variance_explained_per_factor(mtcars, scores)
```

computeDVARSSpatialMap	<i>computeDVARSSpatialMap</i>
------------------------	-------------------------------

Description

compute the DVARSS quality control metric

Usage

computeDVARSS(boldMatrix)

Arguments

boldMatrix matrix of bold signal

Value

DVARSS vector.

Author(s)

Tustison NJ, Avants BB

Examples

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

computeDVARSSpatialMap	<i>computeDVARSSpatialMap</i>
------------------------	-------------------------------

Description

compute the DVARSS quality control metric at every voxel

Usage

computeDVARSSpatialMap(boldMatrix)

Arguments

boldMatrix matrix of bold signal

Value

DVARS spatial vector.

Author(s)

Tustison NJ, Avants BB

Examples

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

construct_sim_names	<i>Construct SiMLR Feature Names</i>
---------------------	--------------------------------------

Description

A helper function to generate a character vector of SiMLR feature names based on a set of modality prefixes and numeric indices.

Usage

```
construct_sim_names(base_names, indices)
```

Arguments

base_names	A character vector of modality prefixes (e.g., c("t1", "dt", "rsf")).
indices	An integer vector of component indices (e.g., 1:10).

Value

A character vector of combined names (e.g., "t1PC1", "dtPC1", ... "rsfPC10").

Examples

```
modalities <- c("t1", "dt", "rsf")
components <- 1:3
construct_sim_names(modalities, components)
```

convolveImage	<i>convolve one image with another</i>
---------------	--

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	<i>produces a correlation matrix via weighted correlation.</i>
------	--

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(rnorm(nrow(mat))))
```

create_optimizer	<i>Create and Initialize a SIMLR Optimizer</i>
------------------	--

Description

This factory function creates an optimizer object that holds the state and update logic for a chosen optimization strategy.

Usage

```
create_optimizer(optimizer_type = "hybrid_adam", vmats, ...)
```

Arguments

optimizer_type	A string identifying the optimizer. One of: "hybrid_adam" (Adam direction with Armijo line search - robust), "adam" (Adam with a learning rate schedule - fast), "nadam" (Nesterov-accelerated Adam - fast and robust), "sgd_momentum" (SGD with momentum and learning rate schedule - classic), "armijo_gradient" (steepest descent with robust Armijo backtracking line search - maximally robust).
vmats	The list of initial V matrices, used to set up the dimensions of the optimizer's internal state variables.
...	Additional hyperparameters for the optimizer (e.g., 'beta1', 'beta2').

Value

An optimizer object (a list with a class attribute).

createAntsrTransform	<i>createAntsrTransform</i>
----------------------	-----------------------------

Description

Create and initialize an antsrTransform

Usage

```
createAntsrTransform(  
  type = "AffineTransform",  
  precision = "float",  
  dimension = 3,  
  matrix = NA,  
  offset = NA,  
  center = NA,  
  translation = NA,  
  parameters = NA,  
  fixed.parameters = NA,  
  displacement.field = NULL,  
  supported.types = FALSE  
)
```

Arguments

type	type of transform AffineTransform CenteredAffineTransform Euler2DTransform Euler3DTransform Rigid2DTransform QuaternionRigidTransform Similarity2DTransform CenteredSimilarity2DTransform Similarity3DTransform CenteredRigid2DTransform CenteredEuler3DTransform DisplacementFieldTransform
precision	numerical precision
dimension	spatial dimension of transform
matrix	matrix for linear transforms
offset	offset for linear transforms
center	center for linear transforms

translation translation for linear transforms
 parameters array of parameters
 fixed.parameters
 array of fixed parameters
 displacement.field
 multichannel antsImage for non-linear transform
 supported.types
 flag that returns array of possible transforms types

Value

antsrTransform or list of transform types

Examples

```
trans <- c(3, 4, 5)
tx <- createAntsrTransform(type = "Euler3DTransform", translation = trans)
```

```
createJacobianDeterminantImage
      createJacobianDeterminantImage
```

Description

Compute the jacobian determinant from a transformation file

Usage

```
createJacobianDeterminantImage(domainImg, tx, doLog = FALSE, geom = FALSE)
```

Arguments

domainImg image that defines transformation domain
 tx deformation transformation file name
 doLog return the log jacobian
 geom use the geometric jacobian calculation (boolean)

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)

```

createWarpedGrid	<i>Create a warped grid for visualization.</i>
------------------	--

Description

Deforming a grid is a helpful way to visualize a deformation field. This function enables a user to define the grid parameters and apply a deformable map to that grid.

Usage

```

createWarpedGrid(
  img,
  gridStep = 10,
  gridWidth = 2,
  gridDirections = c(TRUE, TRUE),
  fixedReferenceImage = NULL,
  transform = NA,
  foreground = 1,
  background = 0
)

```

Arguments

img	input image
gridStep	width of grid blocks
gridWidth	width of grid lines
gridDirections	directions in which to draw grid lines, boolean vector
fixedReferenceImage	reference image space
transform	vector of transforms
foreground	intensity value for grid blocks
background	intensity value for grid lines

Value

image is output

Author(s)

Avants BB

Examples

```

fi <- antsImageRead(getANTsRData("r16"))
mi <- antsImageRead(getANTsRData("r64"))
mygr <- createWarpedGrid(mi)
## Not run:
mytx <- antsRegistration(fixed = fi, moving = mi, typeofTransform = c("SyN"))
mywarpedgrid <- createWarpedGrid(mi,
  gridDirections = c(F, T),
  transform = mytx$fwdtransforms, fixedReferenceImage = fi
)
plot(fi, mywarpedgrid, alpha = 0.75, color.overlay = "blue")

## End(Not run)

```

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, 100)

```

cropImageCenter	<i>Crop the center of an image.</i>
-----------------	-------------------------------------

Description

Crop the center of an image.

Usage

```
cropImageCenter(image, cropSize)
```

Arguments

image	Input ANTs image
cropSize	width, height, depth (if 3-D), and time (if 4-D) of the cropped image.

Value

a cropped image

Author(s)

Tustison NJ

Examples

```
library( ANTsR )

image <- antsImageRead( getANTsRData( "r16" ) )
croppedImage <- cropImageCenter( image, c( 64, 64 ) )
```

cropImageFromCenterPoint	<i>Crop a patch from an image around a center point</i>
--------------------------	---

Description

Crop a patch from an image around a center point

Usage

```
cropImageFromCenterPoint(image, centerPoint, patchSize)
```

Arguments

image	Input ANTs image
centerPoint	Physical space coordinates of center point (vector).
patchSize	Vector defining patch size.

Value

A cropped resampled patch image

Author(s)

Tustison NJ

Examples

```
library( ANTsR )
image <- antsImageRead( getANTsRData( "r16" ) )
center <- c( 128, 128 )
patch <- cropImageFromCenterPoint( image, center, c( 64, 64 ) )
```

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(10, 10), c(100, 100))
cropped <- smoothImage(cropped, 5)
decropped <- decropImage(cropped, fi)
```

crossvalidatedR2	<i>Cross-Validated R² value</i>
------------------	--

Description

Computes an R² 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 <code>lm.fit</code> instead of <code>lm</code> . 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)
```

cut.antsImage	<i>Perform Cut on an image</i>
---------------	--------------------------------

Description

Cuts a numeric image into an integer factor, with the option of a mask.

Usage

```
## S3 method for class 'antsImage'
cut(x, breaks, ..., mask)
```

Arguments

x	Object of class antsImage
breaks	either a numeric vector of two or more unique cut points or a single number (greater than or equal to 2) giving the number of intervals into which x is to be cut. Passed to cut)
...	Arguments passed to cut
mask	object to subset the image. If missing, then all values of the image are used

Value

Object of class antsImage with an attribute of levels

Examples

```
img <- makeImage(c(10, 10), rnorm(100))
mask <- img > 0
cut(img, mask = mask, breaks = 4)
cut(img, breaks = 4)
```

cvEigenanatomy	<i>Cross-validation method for eigenanatomy decompositions.</i>
----------------	---

Description

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

Usage

```
cvEigenanatomy(
  demog,
  images,
  outcome,
  ratio = 10,
  mask = NULL,
  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)

```

dataAugmentation

Randomly transform image data.

Description

Given an input image list (possibly multi-modal) and an optional corresponding segmentation image list, this function will perform data augmentation with the following augmentation possibilities: spatial transformations, added image noise, simulated bias field, and histogram warping.

Usage

```

dataAugmentation(
  inputImageList,
  segmentationImageList = NULL,
  pointsetList = NULL,
  numberOfSimulations = 10,
  referenceImage = NULL,
  transformType = "affineAndDeformation",
  noiseModel = "additivegaussian",
  noiseParameters = c(0, 0.05),
  sdSimulatedBiasField = 1,
  sdHistogramWarping = 0.05,
  sdAffine = 0.05,
  sdDeformation = 0.2,
  outputNumpyFilePrefix = NULL,
  verbose = FALSE
)

```

Arguments

<code>inputImageList</code>	list of lists of input images to warp. The internal list sets contains one or more images (per subject) which are assumed to be mutually aligned. The outer list contains multiple subject lists which are randomly sampled to produce output image list.
<code>segmentationImageList</code>	list of segmentation images corresponding to the input image list (optional).
<code>pointsetList</code>	list of pointsets (matrices) corresponding to the input image list (optional). If using this option, the <code>transformType</code> must be invertible.
<code>numberOfSimulations</code>	number of output images. Default = 10.
<code>referenceImage</code>	defines the spatial domain for all output images. If the input images do not match the spatial domain of the reference image, we internally resample the target to the reference image. This could have unexpected consequences. Resampling to the reference domain is performed by testing using <code>antsImagePhysicalSpaceConsistency</code> then calling <code>resampleImageToTarget</code> upon failure.
<code>transformType</code>	one of the following options <code>c("translation", "rigid", "scaleShear", "affine", "deformation" , "affineAndDeformation")</code> .
<code>noiseModel</code>	one of the following options <code>c("additivegaussian", "saltandpepper", "shot", "speckle", or "random")</code> . Alternatively, one can specify an array or list of one or more of the options and one is selected at random with reasonable, randomized parameters. Note that the "speckle" model takes much longer than the others.
<code>noiseParameters</code>	parameters for the specified noise model.
<code>sdSimulatedBiasField</code>	Characterize the standard deviation of the amplitude.
<code>sdHistogramWarping</code>	Determines the strength of the histogram transformation.
<code>sdAffine</code>	Determines the amount of affine transformation.
<code>sdDeformation</code>	Determines the amount of deformable transformation.
<code>outputNumpyFilePrefix</code>	Filename of output numpy array containing all the simulated images and segmentations.
<code>verbose</code>	print progress to the screen.

Value

list of lists of transformed images and/or outputs to a numpy array.

Author(s)

Tustison NJ

Examples

```
## Not run:
library( ANTsR )
image1 <- antsImageRead( getANTsRData( "r16" ) )
image2 <- antsImageRead( getANTsRData( "r64" ) )
segmentation1 <- thresholdImage( image1, "Otsu", 3 )
segmentation2 <- thresholdImage( image2, "Otsu", 3 )
points1 = getCentroids( segmentation1 )[,1:2]
points2 = getCentroids( segmentation2 )[,1:2]
data <- dataAugmentation(
  list( list( image1 ), list( image2 ) ),
  list( segmentation1, segmentation2 ),
  list( points1, points2 ), transformType = 'scaleShear' )
rm(segmentation1); gc()
rm(segmentation2); gc()
rm(image1); gc()
rm(image2); gc()

## End(Not run)
```

decode_pymm_names

Decode ANTsPyMM technical shorthand to legible anatomical names

Description

This function uses consolidated atlas data (AAL, DKT, CIT168, JHU, etc.) to translate technical shorthand strings into their original, legible anatomical descriptions. It performs normalization to handle variations in dots, underscores, and case.

Usage

```
decode_pymm_names(x)
```

Arguments

x A character vector of shorthand names to decode.

Value

A character vector of legible anatomical names.

Examples

```
decode_pymm_names("cit168.bn.str.pu.left")
decode_pymm_names("dti.fa.thalamus.left.jhu.icbm.labels.1mm")
```

deconvolutionSVD	<i>Regularize SVD (deconvolution) solution of cerebral blood flow.</i>
------------------	--

Description

Ostergaard's regularization approach for a model independent solution of cerebral blood flow using a blood pool contrast agent (<https://pubmed.ncbi.nlm.nih.gov/8916023>).

Usage

```
deconvolutionSVD(arterialInputFunction, thresholdSVD = 0.2)
```

Arguments

arterialInputFunction
vector specifying the arterial input function over time.

thresholdSVD
is used to threshold the smaller elements of the diagonal matrix during the SVD regularization. 0.2 is a common choice (cf. page 571, end of column 2 in <https://pubmed.ncbi.nlm.nih.gov/16971140>).

Value

regularized residue function, i.e. the right-hand side of $CBF * dT * R = (V * L^{-1} * U^T) * C_{voi}$.

Author(s)

Tustison NJ

Examples

```
S0aif <- 17.62
Saif <- c(
  16.25, 16.37, 20.22, 78.96, 230.5, 249.79, 198.58, 147.76,
  110.39, 111.64, 129.43
)
Caif <- -log(Saif / S0aif)

dSVD <- deconvolutionSVD(Caif, 0.2)
```

decropImage	<i>decrop a sub-image back into the full image</i>
-------------	--

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)
```

denoiseImage	<i>Denoise an image</i>
--------------	-------------------------

Description

Denoise an image using a spatially adaptive filter originally described in J. V. Manjon, P. Coupe, Luis Marti-Bonmati, D. L. Collins, and M. Robles. Adaptive Non-Local Means Denoising of MR Images With Spatially Varying Noise Levels, Journal of Magnetic Resonance Imaging, 31:192-203, June 2010.

Usage

```
denoiseImage(
  img,
  mask,
  shrinkFactor = 1,
  p = 1,
  r = 3,
  noiseModel = c("Rician", "Gaussian"),
  verbose = FALSE
)
```

Arguments

img	scalar image to denoise.
mask	optional to limit the denoise region.
shrinkFactor	downsampling level performed within the algorithm.
p	patch radius for local sample. can be a vector.
r	search radius from which to choose extra local samples. can be a vector.
noiseModel	either Rician or Gaussian.
verbose	boolean

Value

antsImage denoised version of image

Author(s)

N Tustison, B Avants

Examples

```
img <- antsImageRead(getANTSRData("rand")) %>% resampleImage(c(32, 32))
dimg <- denoiseImage(img, img * 0 + 1)
dimg2 <- denoiseImage(img)
```

density.antsImage	<i>Density of Values in an Image</i>
-------------------	--------------------------------------

Description

Computes the density of values of an image with the option for a mask.

Usage

```
## S3 method for class 'antsImage'
density(x, ..., mask)
```

Arguments

x	Object of class <code>antsImage</code>
...	Arguments passed to <code>density.default</code>
mask	object to subset the image. If missing, then all values of the image are plotted.

Value

Output of `density`

Examples

```
img <- makeImage(c(10, 10), rnorm(100))
mask <- img > 0
density(img, mask = mask)
```

DesikanKillianyTourville

DesikanKillianyTourville

Description

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

Usage

```
DesikanKillianyTourville
```

Format

A data frame listing the following variables.

label_num Numerical label value.

label_name Shorthand anatomical value.

tissue Shorthand tissue value.

contralateral contralateral anatomical value.

lobe which lobe: temporal, parietal, occipital, frontal.

hemisphere left or right hemisphere.

References

<https://pubmed.ncbi.nlm.nih.gov/23227001>

Examples

```
data(DesikanKillianyTourville)
```

dice_overlap_soft_matrix

Compute pairwise Dice overlaps for matrix rows or columns

Description

Calculates Dice similarity coefficients between corresponding rows or columns of two matrices, treating all non-zero values as 1 (soft binary overlap).

Usage

```
dice_overlap_soft_matrix(A, B, margin = 1)
```

Arguments

A	Numeric matrix.
B	Numeric matrix of the same dimensions as 'A'.
margin	Integer: 1 for row-wise comparison, 2 for column-wise comparison.

Value

A numeric vector of Dice coefficients (length = nrow(A) or ncol(A)).

Examples

```
mat1 <- matrix(c(0,1,0,2, 1,0,0,0), nrow=2, byrow=TRUE)
mat2 <- matrix(c(1,1,0,0, 0,0,1,0), nrow=2, byrow=TRUE)
dice_overlap_soft_matrix(mat1, mat2, margin = 1)
dice_overlap_soft_matrix(t(mat1), t(mat2), margin = 2)
```

dice_overlap_soft_vector

Compute Dice overlap for sparse vectors with optional quantile thresholding

Description

This function computes the Dice similarity coefficient for two vectors, either using a soft binary threshold (non-zero = 1), or by thresholding values by quantile.

Usage

```
dice_overlap_soft_vector(x, y, quantile = 0.5)
```

Arguments

x	Numeric vector.
y	Numeric vector of the same length as 'x'.
quantile	Numeric between 0 and 1. If > 0, values below the given quantile (based on absolute value) are set to 0 before computing Dice.

Value

Dice similarity coefficient (between 0 and 1).

Examples

```

dice_overlap_soft_vector(c(0,1,0,2), c(3,1,0,0)) # returns 0.5
dice_overlap_soft_vector(c(0,0,0,0), c(0,0,0,0)) # returns 1
dice_overlap_soft_vector(c(1,2,3,4), c(1,0,0,4), quantile=0.5) # uses threshold

```

```

displacementFieldFromAntsrTransform
      displacementFieldFromAntsrTransform

```

Description

Conver antsrTransform to displacement field

Usage

```
displacementFieldFromAntsrTransform(tx, reference = NA)
```

Arguments

tx	antsrTransform
reference	reference antsImage if converting linear transform

Value

antsImage

<code>divide_by_column_sum</code>	<i>Divide each column by the sum of column absolute values</i>
-----------------------------------	--

Description

Divide each column by the sum of column absolute values

Usage

```
divide_by_column_sum(m)
```

Arguments

`m` A numeric matrix

Value

A matrix with each column divided by its sum

<code>dkt</code>	<i>dkt</i>
------------------	------------

Description

`dkt`

<code>dkt_cit_labels</code>	<i>dkt_cit_labels</i>
-----------------------------	-----------------------

Description

A data frame label numbers and names for consolidated DKT and CIT labels.

Usage

```
dkt_cit_labels
```

Format

A data frame with columns: Label, Description.

dkt_cortex_cit_deep_brain
<i>dkt cortex cit deep brain</i>

Description

dkt cortex cit deep brain

dkt_labels	<i>dkt_labels</i>
------------	-------------------

Description

A data frame label numbers and names for DKT cortical labels.

Usage

dkt_labels

Format

A data frame with columns: Label, Description, type.

dlbs	<i>dlbs</i>
------	-------------

Description

dlbs

eanatDef

Eigenanatomy with deflation

Description

Simplified, low-parameter eigenanatomy implemented with deflation. The algorithm is able to automatically select hidden sparseness parameters, given the key parameter `nvecs`. The user should select the `cthresh` and `smoother` regularization parameters for a given application and also based on observing algorithm behavior when `verbose=TRUE`.

Usage

```
eanatDef(
  inmat,
  nvecs = 0,
  mask = NULL,
  smoother = 0,
  cthresh = 0,
  its = 5,
  eps = 0.1,
  positivity = FALSE,
  priors = NA,
  priorWeight = 0,
  sparEpsilon = 1e-04,
  whiten = FALSE,
  verbose = FALSE
)
```

Arguments

<code>inmat</code>	input matrix
<code>nvecs</code>	number of eigenanatomy vectors to compute. see <code>eanatSelect</code> for a method to compute an optimal <code>nvecs</code> value.
<code>mask</code>	input mask, must match matrix
<code>smoother</code>	regularization parameter, typically 0 or 0.5, in voxels
<code>cthresh</code>	remove isolated voxel islands of size below this value
<code>its</code>	number of iterations
<code>eps</code>	gradient descent parameter
<code>positivity</code>	return unsigned eigenanatomy vectors
<code>priors</code>	external initialization matrix.
<code>priorWeight</code>	weight on priors in range 0 to 1.
<code>sparEpsilon</code>	threshold that controls initial sparseness estimate
<code>whiten</code>	use ICA style whitening.
<code>verbose</code>	controls whether computation is silent or not.

Value

matrix is output, analogous to `svd(mat, nu=0, nv=nvecs)`

Author(s)

Avants BB, Tustison NJ

References

Kandel, B. M.; Wang, D. J. J.; Gee, J. C. & Avants, B. B. Eigenanatomy: sparse dimensionality reduction for multi-modal medical image analysis. *Methods*, 2015, 73, 43-53. PS Dhillon, DA Wolk, SR Das, LH Ungar, JC Gee, BB Avants Subject-specific functional parcellation via Prior Based Eigenanatomy *NeuroImage*, 2014, 99, 14-27.

See Also

`eanatSelect` <https://github.com/stnava/blindSourceSeparationInANTsR>

Examples

```
## Not run:
mat <- matrix(rnorm(2000), ncol = 50)
nv <- eanatSelect(mat, selectorScale = 1.2)
esol <- eanatDef(mat, nvecs = nv)
es2 <- sparseDecom(mat, nvecs = nv)
print(paste("selected", nrow(esol), "pseudo-eigenvectors"))
print(mean(abs(cor(mat %*% t(esol))))) # what we use to select nvecs
networkPriors <- getANTsRData("fmrinetworks")
ilist <- networkPriors$images
mni <- antsImageRead(getANTsRData("mni"))
mnireg <- antsRegistration(meanbold * mask, mni, typeofTransform = "Affine")
for (i in 1:length(ilist)) {
  ilist[[i]] <- antsApplyTransforms(meanbold, ilist[[i]], mnireg$fdwtransform)
}
pr <- imageListToMatrix(ilist, cortMask)
esol <- eanatDef(boldMat,
  nvecs = length(ilist), cortMask, verbose = FALSE,
  cthresh = 25, smoother = 0, positivity = TRUE, its = 10, priors = pr,
  priorWeight = 0.15, eps = 0.1
)

## End(Not run)
```

eanatSelect

Selection of n eigenvectors and sparsity for eigenanatomy

Description

The algorithm automatically selects the key `nvecs` and hidden sparseness parameters. The user should select the `cthresh` regularization parameters for his or her application. The principle used here is that we want few but sparse pseudo-eigenvectors that are minimally correlated in row-space. true left and right eigenvectors are uncorrelated in both row and column (left and right eigenvector) spaces, but this is not the case when we impose sparsity.

Usage

```
eanatSelect(
  inmat,
  mask = NULL,
  cthresh = 0,
  smooth = 0,
  maxNEvec = 0,
  selectorScale = 1.1,
  whiten = FALSE,
  verbose = FALSE
)
```

Arguments

<code>inmat</code>	input matrix
<code>mask</code>	input mask, must match matrix
<code>cthresh</code>	remove isolated voxel islands of size below this value
<code>smooth</code>	smooth the input data first by this value
<code>maxNEvec</code>	integer that, if set greater than zero, indicates that we use a low-rank approximation to the input matrix before proceeding to <code>eanat</code> . this value should be greater than <code>nvecs</code>
<code>selectorScale</code>	influences automatic selection of <code>nvecs</code> and tries to find the knee in the correlation plot. This parameter produces fewer, less sparse eigenanatomy pseudo-eigenvectors as its value increases. Its minimum value is 1 and a reasonable range is between 1 and 2. The user should look at the plot produced when verbosity is turned on.
<code>whiten</code>	use ICA style whitening.
<code>verbose</code>	controls whether computation is silent or not.

Value

`nvecs` is output, analogous to `nvecs` in `svd(mat, nu=0, nv=nvecs)`

Author(s)

Avants BB, Tustison NJ

Examples

```
## Not run:
mat <- matrix(rnorm(2000), ncol = 50)
nvecsSel <- eanatSelect(mat, selectorScale = 1.2, maxNEvec = 4)
esol <- sparseDecom(mat, nvecs = nvecsSel)
print(paste("selected", nvecsSel, "pseudo-eigenvectors"))

## End(Not run)
```

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 = NULL,
  imgList = NA,
  applySegmentationToImages = FALSE,
  cthresh = 0,
  smooth = 1
)
```

Arguments

mask	D-dimensional mask > 0 defining segmentation region.
imgList	list containing antsImages or a matrix matching the mask.
applySegmentationToImages	boolean determines if original image list is modified by the segmentation.
cthresh	throw away isolated clusters smaller than this value
smooth	smooth the input data first by this value

Value

segmentation image.

Author(s)

Avants BB

Examples

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

ensembled_sparsity	<i>Create a Smoothed Sparse Matrix via Permutation Ensemble Averaging</i>
--------------------	---

Description

This function removes the bias of treating a single column as "special" by running the ‘spar-sify_by_column_winner’ function multiple times. In each run, the columns of the input matrix are permuted so that a different column is in the "first" position. The results are un-permuted and then averaged to produce a stable, unbiased sparse representation.

Usage

```
ensembled_sparsity(X, default_constraint = "positive")
```

Arguments

X A numeric matrix [p_features x k_components].

default_constraint The constraint to apply to all other columns.

Value

A single, averaged sparse matrix with the same dimensions as X.

Examples

```
set.seed(42)
mat <- matrix(rnorm(15), nrow=5, ncol=3)
mat[2,1] <- 10; mat[4,2] <- -12; mat[1,3] <- 14
print("Original Matrix:")
print(round(mat, 2))

# Keep one column dense, sparsify others by magnitude ('either')
```

```
# sparse_ensembled <- ensembled_sparsity(mat,
# default_constraint = "either"
# )
```

error_not_antsImage *Stop if not an antsImage object*

Description

Stop if not an antsImage object

Usage

```
error_not_antsImage(x, argname = "")
```

Arguments

x	object to test
argname	Name of argument to say was not an antsImage

Value

Invisible NULL

Examples

```
img <- antsImageRead(getANTsRData("r16"), 2)
error_not_antsImage(img)
testthat::expect_error(error_not_antsImage(as.array(img)))
```

estimate_joint_rank *Estimate the Joint Rank of a Multi-View Dataset via Parallel Analysis*

Description

This function provides a robust estimate of the number of significant, shared components in a list of data matrices. It uses a streamlined Parallel Analysis approach, comparing the singular value spectrum of the real data against the spectrum of permuted (noise) data.

Usage

```
estimate_joint_rank(
  mat_list,
  n_permutations = 20,
  pre_scaling = c("frobenius", "none"),
  plot_screes = TRUE
)
```

Arguments

<code>mat_list</code>	A list of numeric matrices [subjects x features].
<code>n_permutations</code>	The number of permutations to perform to create a stable null distribution. A higher number is more stable but slower. Defaults to 20.
<code>pre_scaling</code>	Method to scale matrices before combining them. "frobenius" (default) scales each matrix to have a total variance of 1, ensuring fair contribution. "none" performs no scaling.
<code>plot_scee</code>	Logical. If TRUE, generates a scree plot comparing the real and permuted singular values, which is essential for visual diagnosis.

Details

The function first combines all modalities into a single matrix. It then computes the SVD and its eigenvalues (squared singular values) for this real data. It repeats this process 'n_permutations' times on shuffled versions of the data to generate a stable "null" or "chance" eigenvalue distribution. The optimal k is determined as the last component where the real eigenvalue exceeds the mean of the permuted eigenvalues.

Value

A list containing:

<code>optimal_k</code>	The estimated number of significant components.
<code>results</code>	A tibble showing the real vs. permuted eigenvalues for each component.
<code>plot</code>	A ggplot object of the scree plot.

estimate_learning_rate_nsa

Estimate Robust Learning Rate for NSA-Flow

Description

General-purpose estimator for an initial learning rate in `nsa_flow()`, supporting multiple automatic strategies including the original Brent line search.

Usage

```
estimate_learning_rate_nsa(
  Y0,
  X0,
  w = 0.5,
  retraction = "soft_polar",
  nsa_energy = NULL,
  apply_nonneg = TRUE,
  method = c("brent", "grid", "armijo", "golden", "adaptive", "default"),
  search_range = c(-8, 2),
```



```

    verbose = TRUE,
    plot = TRUE
  )

```

Arguments

<code>Y0</code>	Numeric matrix, initial point.
<code>X0</code>	Numeric matrix, target matrix.
<code>w</code>	Weighting parameter between fidelity and orthogonality.
<code>retraction</code>	Retraction method used by <code>nsa_flow</code> .
<code>nsa_energy</code>	Optional function(<code>Y</code>) to compute NSA energy; if <code>NULL</code> , uses default.
<code>apply_nonneg</code>	Logical; if <code>TRUE</code> , applies nonnegativity.
<code>method</code>	Character; learning rate estimation method: one of <code>c("brent", "grid", "armijo", "golden", "adaptive")</code> . "default" returns fixed values (0.001 if <code>nonneg</code> , else 1.0).
<code>search_range</code>	Numeric vector of length 2; $\log_{10}(\alpha)$ search range
<code>verbose</code>	Logical; if <code>TRUE</code> , prints diagnostics.
<code>plot</code>	Logical; if <code>TRUE</code> , plots energy vs learning rate (if applicable).

Value

A list with:

- `estimated_learning_rate`: numeric, chosen value.
- `search_df`: data frame with $\log_{10}(\alpha)$, α , and energy (if available).
- `plot`: ggplot object (if `plot=TRUE`), otherwise `NULL`.

```
estimate_rank_by_permutation_rv
```

Estimate Joint Rank via Cross-View RV Analysis

Description

Estimate Joint Rank via Cross-View RV Analysis

Usage

```

estimate_rank_by_permutation_rv(
  mat_list,
  n_permutations = 20,
  var_explained_threshold = 0.9,
  handle_missing = TRUE,
  return_max = FALSE
)

```

Arguments

mat_list	A list of numeric matrices [subjects x features].
n_permutations	Number of permutations (0 = Elbow method).
var_explained_threshold	Variance threshold to determine individual k_max.
handle_missing	Boolean; if TRUE, replaces NA/Constant columns with row means.
return_max	Boolean; if TRUE, returns only the k_max and skips curves.

Value

A list containing optimal_k, results tibble, and a diagnostic plot.

estSmooth	<i>Estimate image smoothness</i>
-----------	----------------------------------

Description

Estimates smoothness of a single image or image matrix

Usage

```
estSmooth(x, mask, rdf, scaleResid = TRUE, sample = NULL, verbose = TRUE)
```

Arguments

x	may be an image of class "antsImage" or an image matrix
mask	input mask, must match matrix
rdf	residual degrees of freedom
scaleResid	logical. if TRUE residuals are scaled
sample	number of images to use for estimating smoothing (default uses all images)
verbose	enables verbose output

Details

The partial derivatives of an image in x, y, and z directions are used to create a covariance matrix which in turn is used to calculate the full-widths at half maxima (FWHM). The FWHM is equivalent to the estimated image smoothness.

The resels per voxel image (RPVImg) represents the estimated resel at each individual voxel throughout the search region. This may be used in place of volumetric measurements (or sum voxel measurements) when estimating the p-value of a cluster using `rftPval`. The intent behind using the RPV image to estimate cluster level statistics is to offset the natural probability of obtaining significant clusters solely by chance in very smooth regions at low thresholds.

It is possible to use a single statistical field image to estimate the FWHM. However, it's recommended that FWHM estimates are obtained from the scaled residuals of statistical models (Stefan

J.K et al., 1999). Therefore, this function is optimized to estimate the pooled smoothness of the residual images from a fitted model. By default residuals are scaled (scaleResid = TRUE).

A scaling factor is used to correct for differences when using the `sample` option. Scaling isn't effective when the number of images is very low and typically results in an overestimation of the the FWHM. If only one image or numeric vector is entered then the scaling factor is not used. If a numeric vector is entered the `imageMake` function is used to prepare it for smoothness estimation (see Worsley et al., 1999).

Any NA values in object will be set to zero.

Value

Outputs the estimated FWHM and RPV image

Author(s)

Zachary P. Christensen

References

Hayasaka (2004) Nonstationary cluster-size inference with random field and permutation methods.

Worsley K.J. (1992) A Three-Dimensional Statistical Analysis for CBF Activation Studies in Human Brain.

Worsley K.J. (1996) A Unified Statistical Approach for Determining Significant Signals in Images of Cerebral Activation.

Worsley K.J. (1999) Detecting Changes in Nonisotropic Images

Stefan J.K. (1999) Robust Smoothness Estimation in Statistical Parametric Maps Using Standardized Residual from the General Linear Model

See Also

`resels`

Examples

```
# estimate individual image
mnit1 <- antsImageRead(getANTsRData("r16"))
mask <- getMask(mnit1)
fwhm1 <- estSmooth(mnit1, mask)
```

euler

*Euler Character***Description**

Mostly an for internal use in conjunction with resels. Calculates the euler characteristic at a threshold level.

Usage

```
euler(u, df, fieldType)
```

Arguments

u	statistical value (typically the maxima of a cluster or statistical field)
df	degrees of freedom expressed as $df = c(\text{degrees of interest}, \text{degrees of error})$
fieldType	T: T-field F: F-field X: Chi-square field' Z: Gaussian field

Value

A vector of estimated euler characteristics for dimensions 0:D

Author(s)

Zachary P. Christensen

References

Worsley K.J., (1996) A Unified Statistical Approach for Determining Significant Signals in Images of Cerebral Activation.

See Also

rftPval, resels

Examples

```
mask <- getMask(antsImageRead(getANTsRData("r16")))
myresels <- resels(mask, c(1, 1))
# pretend these results are from a t-statistical field fitted with 10 df to
# find a peak value.
df <- c(1, 10) # made up degrees of freedom
ec <- euler(4, df, fieldType = "T") # peak value is 4
ez <- euler(3, df, fieldType = "T") # we thresholded at 3
# uncorrected peak-level p-value
```

```
pvox <- sum(ec * myresels) / sum(ez * myresels)
```

exemplarInpainting	<i>Uses example images to inpaint or approximate an existing image.</i>
--------------------	---

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]]))
```

exploratory_visualization

Exploratory Clustering and Visualization

Description

This function performs automated clustering using PAM and silhouette method, and visualizes the results using ggplot2 and gg dendro.

Usage

```
exploratory_visualization(data, dotsne = FALSE, verbose = FALSE)
```

Arguments

data	A data frame with numeric columns.
dotsne	boolean
verbose	boolean

Value

A list containing the combined plot and the optimal k value.

```
extend_simlr_embedding_with_new_modalities
```

Extend an existing SIMLR embedding with new modality blocks

Description

Rebuilds the projected SIMLR feature space from an existing SIMLR result, adds one or more new modality blocks from the projected data, constructs block-specific adjacency structures, chooses a joint initialization rank, and runs `simlr.perm()` on the combined blocks.

Usage

```
extend_simlr_embedding_with_new_modalities(
  pymm,
  simlr_result,
  new_modalities,
  mode = c("concatenate", "split", "auto"),
  split_prefixes = c("t1", "dt", "rsf"),
  adjacency_type = c("feature_correlation"),
  cor_threshold = 0.8,
  use_abs_correlation = TRUE,
  k_new = NULL,
  k_method = c("cumulative", "elbow"),
  cumvar_threshold = 0.9,
  min_k = 2L,
  joint_k_policy = c("max", "median", "min"),
  preprocess = c("drop", "zero"),
  verbose = FALSE,
  ...
)
```

Arguments

<code>pymm</code>	Input multimodal object passed through to <code>apply_simlr_matrices_dtfix()</code> .
<code>simlr_result</code>	Existing SIMLR result object with a <code>\$v</code> element.
<code>new_modalities</code>	Named list mapping modality names to projected-data column names, e.g. <code>list(pet = c("petPC1", "petPC2"))</code> .
<code>mode</code>	How to construct existing blocks from the current SIMLR projection. One of "concatenate", "split", or "auto".
<code>split_prefixes</code>	Character vector of prefixes used when <code>mode = "split"</code> .
<code>adjacency_type</code>	Type of within-block adjacency to build. Currently "feature_correlation" is supported.
<code>cor_threshold</code>	Nonnegative threshold applied to absolute correlations when building adjacency matrices.

<code>use_abs_correlation</code>	Logical; if TRUE, threshold on absolute correlation magnitude.
<code>k_new</code>	Optional rank specification for new modalities. May be NULL, a single positive integer, or a named numeric vector keyed by new modality names.
<code>k_method</code>	Method for automatic rank selection when <code>k_new</code> is not supplied for a modality. One of "cumulative" or "elbow".
<code>cumvar_threshold</code>	Cumulative variance threshold used when <code>k_method</code> = "cumulative".
<code>min_k</code>	Minimum admissible latent dimension.
<code>joint_k_policy</code>	Policy used to combine per-modality recommended ranks into a single initialization rank. One of "max", "median", or "min".
<code>preprocess</code>	Policy for invalid columns. One of "drop" or "zero".
<code>verbose</code>	Logical; emit progress messages.
<code>...</code>	Additional arguments forwarded to <code>simlr.perm()</code> .

Details

This implementation is intentionally strict: - input validation is explicit - preprocessing is deterministic - constant / all-NA columns are dropped by default rather than jittered - the input `simlr_result` is not mutated - diagnostics are returned for auditability

Value

A list with:

- updated_simlr_result** Copy of `simlr_result` with any returned modality-specific \$v entries merged in, without mutating input.
- simlr_permutations** Result from `simlr.perm()`.
- blocks** Named list of processed block matrices used for fitting.
- adjacency** Named list of adjacency matrices.
- projected_data** Projected data returned by `apply_simlr_matrices_dtfix()`.
- k_new_used** Named integer vector of per-new-modality ranks used.
- joint_k** Joint initialization rank passed to `initializeSimlr()`.
- diagnostics** List of validation and preprocessing diagnostics.

extractImagePatchCoordinates

Extract 2-D or 3-D image patch coordinates.

Description

Extract 2-D or 3-D image patch coordinates.

Usage

```
extractImagePatchCoordinates(
    image,
    patchSize,
    maxNumberOfPatches = "all",
    strideLength = 1,
    maskImage = NULL,
    physicalCoordinates = TRUE,
    cornerCoordinates = TRUE,
    randomSeed,
    randomize = TRUE
)
```

Arguments

image	Input ANTs image with one or more components.
patchSize	Width, height, and depth (if 3-D) of patches.
maxNumberOfPatches	Maximum number of patches returned. If "all" is specified, then all patches in sequence (defined by the strideLength are extracted.
strideLength	Defines the sequential patch overlap for maxNumberOfPatches = "all". Can be a image-dimensional vector or a scalar.
maskImage	optional image specifying the sampling region for the patches when maximumNumberOfPatches does not equal "all". The way we constrain patch selection using a mask is by forcing each returned patch to have a masked voxel at its center.
physicalCoordinates	boolean to determine whether indices or spatial coordinates are returned.
cornerCoordinates	boolean to determine whether indices or spatial coordinates of the corner or the center are returned.
randomSeed	integer seed that allows reproducible patch extraction across runs.
randomize	boolean controlling whether we randomize indices when masking.

Value

a matrix of image patch spatial or index coordinates.

Author(s)

Tustison NJ, Avants B

Examples

```
## Not run:
library( ANTsR )

image <- antsImageRead( getANTsRData( "r16" ) )
maskImage <- getMask( image, 1, 1000 )
patchCoordsP <- extractImagePatchCoordinates( image, c( 32, 32 ), 10, c( 32, 32 ),
  maskImage, randomSeed = 0, physicalCoordinates = TRUE )
patchCoordsI <- extractImagePatchCoordinates( image, c( 32, 32 ), 10, c( 32, 32 ),
  maskImage, randomSeed = 0, physicalCoordinates = FALSE )

## End(Not run)
```

extractImagePatches *Extract 2-D or 3-D image patches.*

Description

Extract 2-D or 3-D image patches.

Usage

```
extractImagePatches(
  image,
  patchSize,
  maxNumberOfPatches = "all",
  strideLength = 1,
  maskImage = NULL,
  randomSeed,
  returnAsArray = FALSE,
  randomize = TRUE
)
```

Arguments

image	Input ANTs image with one or more components.
patchSize	Width, height, and depth (if 3-D) of patches.
maxNumberOfPatches	Maximum number of patches returned. If "all" is specified, then all patches in sequence (defined by the strideLength are extracted.
strideLength	Defines the sequential patch overlap for maxNumberOfPatches = "all". Can be a image-dimensional vector or a scalar.

maskImage	optional image specifying the sampling region for the patches when maximumNumberOfPatches does not equal "all". The way we constrain patch selection using a mask is by forcing each returned patch to have a masked voxel at its center.
randomSeed	integer seed that allows reproducible patch extraction across runs.
returnAsArray	specifies the return type of the function. If FALSE (default) the return type is a list where each element is a single patch. Otherwise the return type is an array of size dim(numberOfPatches, patchSize).
randomize	boolean controlling whether we randomize indices when masking.

Value

a list (or array) of image patches.

Author(s)

Tustison NJ

Examples

```
library( ANTsR )

image <- antsImageRead( getANTsRData( "r16" ) )
maskImage <- getMask( image, 1, 1000 )
patchSet1 <- extractImagePatches( image, c( 32, 32 ), 10, c( 32, 32 ), randomSeed = 0 )
patchSet2 <- extractImagePatches( image, c( 32, 32 ), 10, c( 32, 32 ), randomSeed = 1 )
patchSet3 <- extractImagePatches( image, c( 32, 32 ), 10, c( 32, 32 ), maskImage, randomSeed = 0 )
```

extractSlice	<i>extract a slice from an image</i>
--------------	--------------------------------------

Description

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

Usage

```
extractSlice(image, slice, direction, collapseStrategy = 0)
```

Arguments

image	antsImage to crop
slice	which slice, integer
direction	which axis, integer
collapseStrategy	collapse sub-matrix. 0: Collapse sub-matrix if positive definite. Otherwise, thrown exception. Default. 1: Set sub-matrix to identity. 2: Collapse if positive definite. Otherwise, set to identity.

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)
```

FA_JHU_labels_edited	<i>FA JHU labels edited</i>
----------------------	-----------------------------

Description

FA JHU labels edited

fast_whiten	<i>Fast Matrix Whitening</i>
-------------	------------------------------

Description

Perform whitening (sphering) of a data matrix X such that the covariance matrix of the output is approximately the identity. The implementation automatically chooses the most efficient algorithm depending on whether the number of features p is much larger than the number of samples n ($p \gg n$) or vice versa ($n \gg p$).

Usage

```
fast_whiten(X, epsilon = 1e-08)
```

Arguments

- | | |
|---------|--|
| X | A numeric matrix of size $n \times p$, where n is the number of samples (rows) and p is the number of features (columns). |
| epsilon | A small non-negative numeric value used to stabilize the inversion of singular values (default = 1e-8). |

Details

Whitening is performed via an SVD-based approach with Tikhonov-style regularization to handle small singular values.

- If $p > n$, whitening is computed in the sample space (avoids forming a $p \times p$ covariance). - If $n \geq p$, whitening is computed in the feature space (avoids forming an $n \times n$ covariance).

Computational complexity:

- $O(n^2p)$ when $p \gg n$
- $O(np^2)$ when $n \gg p$

Value

A numeric matrix of the same dimensions as X , with whitened columns. The covariance of the whitened data is approximately the identity in the relevant space.

Examples

```
set.seed(42)

# Case 1: p >> n
X1 <- matrix(rnorm(50 * 2000), 50, 2000)
Xw1 <- fast_whiten(X1)
round(cov(t(Xw1))[1:5, 1:5], 3)

# Case 2: n >> p
X2 <- matrix(rnorm(2000 * 50), 2000, 50)
Xw2 <- fast_whiten(X2)
round(cov(Xw2)[1:5, 1:5], 3)
```

fastMarchingExtension *Fast Marching Extension filter will extend an intensity or label image from a known region into the unknown region along a geodesic path*

Description

This is an implementation of ITK's FastMarchingExtensionImageFilter - From http://www.itk.org/Doxygen/html/classitk_1_1FastMarchingExtensionImageFilter.html

- Fast marching can be used to extend auxiliary variables smoothly from the zero level set. Starting from an initial position on the front, this class simultaneously calculate the signed distance and extend a set of auxiliary values. Implementation of this class is based on Chapter 11 of "Level Set Methods and Fast Marching Methods", J.A. Sethian, Cambridge Press, Second edition, 1999.

Usage

```
fastMarchingExtension(speedImage, labelImage, valueImage)
```

Arguments

speedImage	defines the cost or distance function
labelImage	defines the known (value 1) and unknown (value 2) regions
valueImage	these values are extended into the unknown regions

Value

antsImage

Author(s)

Duda, JT

Examples

```
img <- makeImage(c(7, 7), 0)
img[3:5, 3:5] <- 1:9
mask <- getMask(img)
mask[mask == 0] <- 2
speed <- smoothImage(mask, 1)
extendedImg <- fastMarchingExtension(speed, mask, img)
```

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 = NULL,
```

```

    labels = NULL,
    graphdensity = 0.5,
    seg = NULL,
    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)

```

```
fitBsplineDisplacementField
    fitBsplineDisplacementField
```

Description

Fit a b-spline object to a dense displacement field image and/or a set of points with associated displacements and smooths them using B-splines. The inverse can also be estimated.. This is basically a wrapper for the ITK filter http://docs.itk.org/projects/doxygen/en/stable/classitk_1_1DisplacementFieldToBsplineImageFilter.html which, in turn is a wrapper for the ITK filter used for the function ANTsR::fitBsplineObjectToScatteredData

Usage

```
fitBsplineDisplacementField(
  displacementField = NULL,
  displacementWeightImage = NULL,
  displacementOrigins = NULL,
  displacements = NULL,
  displacementWeights = NULL,
  origin = NULL,
  spacing = NULL,
  size = NULL,
  direction = NULL,
  numberOfFittingLevels = 4,
  meshSize = 1,
  splineOrder = 3,
  enforceStationaryBoundary = TRUE,
  estimateInverse = FALSE,
  rasterizePoints = FALSE
)
```

Arguments

<code>displacementField</code>	input displacement field. Either this and/or the points must be specified.
<code>displacementWeightImage</code>	input image defining weighting of the voxelwise displacements in the displacementField. if NULL, defaults to identity weighting for each displacement. Default = NULL.
<code>displacementOrigins</code>	matrix (numberOfPoints x dimension) defining the origins of the input displacement points. Default = NULL.
<code>displacements</code>	matrix (numberOfPoints x dimension) defining the displacements of the input displacement points. Default = NULL.
<code>displacementWeights</code>	vector defining the individual weighting of the corresponding scattered data value. Default = NULL meaning all values are weighted the same.

origin	vector defining the physical origin of the B-spline displacement field. Must be specified if displacementField is not specified. Otherwise, the displacementField is used to define the domain parameters.
spacing	vector defining the physical spacing of the B-spline object. Defines the sampling rate in the parametric domain. Must be specified if displacementField is not specified. Otherwise, the displacementField is used to define the domain parameters.
size	vector defining the size (length) of the B-spline object. Note that the length of the B-spline object in dimension d is defined as parametricDomainSpacing[d] * (parametricDomainSize[d]-1). Must be specified if displacementField is not specified. Otherwise, the displacementField is used to define the domain parameters.
direction	matrix defining the orientation of the sampled B-spline object. Must be specified if displacementField is not specified. Otherwise, the displacementField is used to define the domain parameters.
numberOfFittingLevels	integer specifying the number of fitting levels.
meshSize	vector defining the mesh size at the initial fitting level.
splineOrder	spline order of the B-spline object. Default = 3.
enforceStationaryBoundary	ensure no displacements on the image boundary. Default = TRUE.
estimateInverse	estimate the inverse displacement field. Default = FALSE.
rasterizePoints	Use nearest neighbor rasterization of points for estimating the field (potential speed-up).

Value

ANTsR image.

Author(s)

NJ Tustison

Examples

```
# Perform 2-D fitting

points <- matrix(data = c(-50, -50), nrow = 1, byrow = TRUE)
deltas <- matrix(data = c(10, 10), nrow = 1, byrow = TRUE)

bsplineField <- fitBsplineDisplacementField(
  displacementOrigins = points, displacements = deltas,
  origin = c(0.0, 0.0), spacing = c(1.0, 1.0), size = c(100, 100),
  direction = matrix(data = c(-1, 0, 0, -1), nrow = 2, byrow = TRUE),
  numberOfFittingLevels = 4, meshSize = c(1, 1)
)
```

```
fitBsplineObjectToScatteredData
    fitBsplineObjectToScatteredData
```

Description

Fit a b-spline object to scattered data. This is basically a wrapper for the ITK filter http://docs.itk.org/projects/doxygen/en/stable/classitk_1_1BSplineScatteredDataPointSetToImageFilter.html. This filter is flexible in the possible objects that can be approximated. Possibilities include:

curve: 1/2/3/4-D

surface: 2-D surface in 3-D space (not available/templated)

scalar: 2/3/4-D scalar field

displacement: 2/3-D displacement field

displacement: 2/3-D time-varying velocity field

In order to understand the input parameters, it is important to understand the difference between the parametric and data dimensions. A curve has one parametric dimension but the data dimension can be 1-D, 2-D, 3-D, or 4-D. In contrast, a 3-D displacement field has a parametric and data dimension of 3. The scattered data is what's approximated by the B-spline object and the parametric point is the location of scattered data within the domain of the B-spline object.

Usage

```
fitBsplineObjectToScatteredData(
    scatteredData,
    parametricData,
    parametricDomainOrigin,
    parametricDomainSpacing,
    parametricDomainSize,
    isParametricDimensionClosed = NULL,
    dataWeights = NULL,
    numberOfFittingLevels = 4,
    meshSize = 1,
    splineOrder = 3
)
```

Arguments

<code>scatteredData</code>	matrix defining the scattered data input to be approximated. Data is organized by row → data v, column → data dimension.
<code>parametricData</code>	matrix defining the parametric location of the scattered data. Data is organized by row → parametric point, column → parametric dimension. Note that each row corresponds to the same row in the <code>scatteredData</code> .
<code>parametricDomainOrigin</code>	vector defining the parametric origin of the B-spline object.

parametricDomainSpacing
vector defining the parametric spacing of the B-spline object. Defines the sampling rate in the parametric domain.

parametricDomainSize
vector defining the size (length) of the B-spline object. Note that the length of the B-spline object in dimension *d* is defined as `parametricDomainSpacing[d] * (parametricDomainSize[d]-1)`.

isParametricDimensionClosed
vector of bools defining whether or not the corresponding parametric dimension is closed (e.g., closed loop). Default = FALSE.

dataWeights
vector defining the individual weighting of the corresponding scattered data value. Default = NULL meaning all values are weighted the same.

numberOfFittingLevels
integer specifying the number of fitting levels.

meshSize
vector defining the mesh size at the initial fitting level.

splineOrder
spline order of the B-spline object. Default = 3.

Value

Matrix for B-spline curve. Otherwise, returns ANTsR image.

Author(s)

NJ Tustison

Examples

```
# Perform 2-D curve example

x <- seq(from = -4, to = 4, by = 0.1)
y <- exp(-(x * x)) + runif(length(x), min = -0.1, max = 0.1)
u <- seq(from = 0.0, to = 1.0, length.out = length(x))
scatteredData <- cbind(x, y)
parametricData <- as.matrix(u, ncol = 1)
numberOfSamplePoints <- 100
spacing <- 1 / (numberOfSamplePoints - 1) * 1.0
bsplineCurve <- fitBsplineObjectToScatteredData(scatteredData, parametricData,
  parametricDomainOrigin = c(0.0), parametricDomainSpacing = c(spacing),
  parametricDomainSize = c(numberOfSamplePoints), isParametricDimensionClosed = c(FALSE),
  numberOfFittingLevels = 5, meshSize = 1
)

plot(x, y, "p", col = "red")
points(scatteredData[, 1], scatteredData[, 2], col = "green")
lines(bsplineCurve[, 1], bsplineCurve[, 2], col = "blue")

# Perform 2-D scalar field (i.e., image) example

numberOfRandomPoints <- 10000
```

```

img <- antsImageRead(getANTsRData("r16"))
imgArray <- as.array(img)
rowIndices <- sample(2:(dim(imgArray)[1] - 1), numberOfRandomPoints, replace = TRUE)
colIndices <- sample(2:(dim(imgArray)[2] - 1), numberOfRandomPoints, replace = TRUE)

scatteredData <- as.matrix(array(data = 0, dim = c(numberOfRandomPoints, 1)))
parametricData <- as.matrix(array(data = 0, dim = c(numberOfRandomPoints, 2)))
for (i in seq_len(numberOfRandomPoints))
{
  scatteredData[i, 1] <- imgArray[rowIndices[i], colIndices[i]]
  parametricData[i, 1] <- rowIndices[i]
  parametricData[i, 2] <- colIndices[i]
}

bsplineImage <- fitBsplineObjectToScatteredData(scatteredData, parametricData,
  parametricDomainOrigin = c(0.0, 0.0), parametricDomainSpacing = c(1.0, 1.0),
  parametricDomainSize = dim(img),
  numberOfFittingLevels = 7, meshSize = c(1, 1)
)
plot(bsplineImage)

```

fitThinPlateSplineDisplacementField

fitThinPlateSplineDisplacementField

Description

Fit a thin-plate spline object to a set of source points with associated displacements. This is basically a wrapper for the ITK class http://docs.itk.org/projects/doxygen/en/stable/itkThinPlateSplineKernelTransform_8h.html. Returns a displacement field.

Usage

```

fitThinPlateSplineDisplacementField(
  displacementOrigins = NULL,
  displacements = NULL,
  origin = NULL,
  spacing = NULL,
  size = NULL,
  direction = NULL
)

```

Arguments

displacementOrigins
matrix (numberOfPoints x dimension) defining the origins of the input displacement points. Default = NULL.

displacements	matrix (numberOfPoints x dimension) defining the displacements of the input displacement points. Default = NULL.
origin	vector defining the physical origin of the B-spline displacement field. Must be specified if displacementField is not specified. Otherwise, the displacementField is used to define the domain parameters.
spacing	vector defining the physical spacing of the B-spline object. Defines the sampling rate in the parametric domain. Must be specified if displacementField is not specified. Otherwise, the displacementField is used to define the domain parameters.
size	vector defining the size (length) of the B-spline object. Note that the length of the B-spline object in dimension d is defined as parametricDomainSpacing[d] * (parametricDomainSize[d]-1). Must be specified if displacementField is not specified. Otherwise, the displacementField is used to define the domain parameters.
direction	matrix defining the orientation of the sampled B-spline object. Must be specified if displacementField is not specified. Otherwise, the displacementField is used to define the domain parameters.

Value

ANTsR image.

Author(s)

NJ Tustison

Examples

```
# Perform 2-D fitting

points <- matrix(data = c(-50, -50), nrow = 1, byrow = TRUE)
deltas <- matrix(data = c(10, 10), nrow = 1, byrow = TRUE)

tpsField <- fitThinPlateSplineDisplacementField(
  displacementOrigins = points, displacements = deltas,
  origin = c(0.0, 0.0), spacing = c(1.0, 1.0), size = c(100, 100),
  direction = matrix(data = c(-1, 0, 0, -1), nrow = 2, byrow = TRUE)
)
```

fitTimeVaryingTransformToPointSets

fitTimeVaryingTransformToPointSets

Description

Estimate a time-varying transform from corresponding point sets (> 2).

Usage

```

fitTimeVaryingTransformToPointSets(
    pointSets,
    timePoints = NULL,
    initialVelocityField = NULL,
    numberOfTimeSteps = NULL,
    domainImage = NULL,
    numberOfFittingLevels = 4,
    meshSize = 1,
    splineOrder = 3,
    displacementWeights = NULL,
    numberOfCompositions = 10,
    compositionStepSize = 0.5,
    numberOfIntegrationSteps = 100,
    sigma = 0,
    convergenceThreshold = 1e-06,
    rasterizePoints = FALSE,
    verbose = FALSE
)

```

Arguments

<code>pointSets</code>	Corresponding points across sets specified in physical space as a $n \times d$ matrix where n is the number of points and d is the dimensionality.
<code>timePoints</code>	Set of scalar values, one for each point-set designating its time position in the velocity flow. If not set, it defaults to equal spacing between 0 and 1.
<code>initialVelocityField</code>	Optional velocity field for initializing optimization. Overrides the number of integration points.
<code>numberOfTimeSteps</code>	Time-varying velocity field parameter. Needs to be equal to or greater than the number of point sets. If not specified, it defaults to the number of point sets.
<code>domainImage</code>	Defines physical domain of the B-spline transform. Must be defined for nonlinear transforms.
<code>numberOfFittingLevels</code>	Integer specifying the number of fitting levels for the B-spline interpolation of the displacement field.
<code>meshSize</code>	Defines the mesh size at the initial fitting level for the B-spline interpolation of the displacement field.
<code>splineOrder</code>	Spline order of the B-spline displacement field.
<code>displacementWeights</code>	vector defining the individual weighting of the corresponding scattered data value. Default = NULL meaning all displacements are weighted the same.
<code>numberOfCompositions</code>	Total number of compositions for the diffeomorphic transform.

compositionStepSize	Scalar multiplication factor of the weighting of the update field for the diffeomorphic transforms.
numberOfIntegrationSteps	Number of steps used for integrating the velocity field.
sigma	Gaussian smoothing standard deviation of the update field (in mm).
convergenceThreshold	Composition-based convergence parameter for the diffeomorphic transforms using a window size of 10 values.
rasterizePoints	Use nearest neighbor rasterization of points for estimating update field (potential speed-up).
verbose	Print progress to the screen.

Value

object containing ANTsR transform, error, and scale (or displacement field)

Author(s)

B Avants, N Tustison

fitTransformToPairedPoints
fitTransformToPairedPoints

Description

Estimate a transform using paired point sets. An ANTsR transform is returned.

Usage

```
fitTransformToPairedPoints(
  movingPoints,
  fixedPoints,
  transformType = "affine",
  regularization = 1e-06,
  domainImage = NULL,
  numberOfFittingLevels = 4,
  meshSize = 1,
  splineOrder = 3,
  enforceStationaryBoundary = TRUE,
  displacementWeights = NULL,
  numberOfCompositions = 10,
  compositionStepSize = 0.5,
  sigma = 0,
```

```

    convergenceThreshold = 1e-06,
    numberOfTimeSteps = 2,
    numberOfIntegrationSteps = 100,
    rasterizePoints = FALSE,
    verbose = FALSE
)

```

Arguments

<code>movingPoints</code>	Moving points specified in physical space as a $n \times d$ matrix where n is the number of points and d is the dimensionality.
<code>fixedPoints</code>	Fixed points specified in physical space as a $n \times d$ matrix where n is the number of points and d is the dimensionality.
<code>transformType</code>	'rigid', 'similarity', 'affine', 'bspline', 'tps', 'diffeo', 'syn', 'tv', or 'time-varying'.
<code>regularization</code>	Ridge penalty in $[0,1]$ for linear transforms.
<code>domainImage</code>	Defines physical domain of the B-spline transform. Must be defined for nonlinear transforms.
<code>numberOfFittingLevels</code>	Integer specifying the number of fitting levels for the B-spline interpolation of the displacement field.
<code>meshSize</code>	Defines the mesh size at the initial fitting level for the B-spline interpolation of the displacement field.
<code>splineOrder</code>	Spline order of the B-spline displacement field.
<code>enforceStationaryBoundary</code>	Ensure no displacements on the image boundary (B-spline only).
<code>displacementWeights</code>	Defines the individual weighting of the corresponding scattered data value. Default = None meaning all displacements are weighted the same.
<code>numberOfCompositions</code>	Total number of compositions for the diffeomorphic transform.
<code>compositionStepSize</code>	Scalar multiplication factor of the weighting of the update field for the diffeomorphic transforms.
<code>sigma</code>	Gaussian smoothing standard deviation of the update field (in mm).
<code>convergenceThreshold</code>	Composition-based convergence parameter for the diffeomorphic transforms using a window size of 10 values.
<code>numberOfTimeSteps</code>	Time-varying velocity field parameter.
<code>numberOfIntegrationSteps</code>	Number of steps used for integrating the velocity field.
<code>rasterizePoints</code>	Use nearest neighbor rasterization of points for estimating update field (potential speed-up).
<code>verbose</code>	Print progress to the screen.

Value

object containing ANTsR transform, error, and scale (or displacement field)

Author(s)

B Avants, N Tustison

Examples

```
fixed <- matrix(c(50, 50, 200, 50, 50, 200), ncol = 2, byrow = TRUE)
moving <- matrix(c(50, 50, 50, 200, 200, 200), ncol = 2, byrow = TRUE)

# Affine transform
xfrm <- fitTransformToPairedPoints(moving, fixed, transformType = "Affine",
regularization = 0)
params <- getAntsrTransformParameters(xfrm)

# Rigid transform
xfrm <- fitTransformToPairedPoints(moving, fixed, transformType = "Rigid",
regularization = 0)
params <- getAntsrTransformParameters(xfrm)

# Similarity transform
xfrm <- fitTransformToPairedPoints(moving, fixed,
transformType = "Similarity", regularization = 0)
params <- getAntsrTransformParameters(xfrm)

# B-spline transform
domainImage <- antsImageRead(getANTsRData("r16"))
xfrm <- fitTransformToPairedPoints(moving, fixed,
transformType = "Bspline", domainImage = domainImage,
numberOfFittingLevels = 5)
```

Description

This function leverages structural image processing based on ANTs cortical thickness to implement standard functional image processing recommendations. The function will crop out the first k time frames, do motion correction and produce a variety of nuisance regressors. It will also do spatial and temporal filtering as well as interpolation between time frames that exceed a given framewise displacement. Finally, we return maps to the common coordinate system. Output may be trimmed in the future but currently provides access at different stages: merging versus filtering and normalized, fused BOLD images in both subject and template space.

Usage

```
fMRINormalization(
  img,
  steadyT = 20,
  fdthresh = Inf,
  repeatMotionEst = 2,
  freqLimits = c(0.01, 0.1),
  nCompCor = 0,
  polydegree = NA,
  structuralImage = NULL,
  structuralSeg = NULL,
  structuralNodes = NULL,
  templateMap = NULL,
  templateImage = NULL,
  smoothingSigmas = NA,
  extraRuns = list(),
  verbose = FALSE
)
```

Arguments

<code>img</code>	input time series <code>antsImage</code> .
<code>steadyT</code>	number of seconds for steady state (used to remove initial volumes)
<code>fdthresh</code>	threshold for framewise displacement. determines what time frames should be interpolated. Set typically between 0.1 and 0.5 or <code>Inf</code> .
<code>repeatMotionEst</code>	number of times to repeat motion estimation. We recommend the value 2, in general. The first run improves the template estimate such that the 2nd run gives a more accurate correction.
<code>freqLimits</code>	pair defining bandwidth of interest from low to high.
<code>nCompCor</code>	number of compcor components to use in CSF plus WM mask.
<code>polydegree</code>	eg 4 for polynomial nuisance variables.
<code>structuralImage</code>	the structural <code>antsImage</code> of the brain.
<code>structuralSeg</code>	a 3 or greater class tissue segmentation of the structural image.
<code>structuralNodes</code>	regions of interest for network analysis, in the structural image space.
<code>templateMap</code>	<code>antsRegistration</code> output mapping template space (as moving) to <code>struturalImage</code> (fixed).
<code>templateImage</code>	template reference space to which we map the BOLD image.
<code>smoothingSigmas</code>	4-vector defining amount of smoothing in FWHM units.
<code>extraRuns</code>	a list containing additional BOLD images (runs) to be merged with the first image.
<code>verbose</code>	enables visualization as well as commentary.

Value

outputs a list containing:

fusedImg: runs fused into one image and corrected if polydegree set

fusedImgFilt: runs fused into one image and filtered

seg2bold: structural segmentation in BOLD space.

nodes2bold: structural nodes in BOLD space.

boldToTemplate: BOLD fusedImg mapped to template space.

mapsToTemplate: invertible maps from BOLD to template space.

runID: Identifies which run over time series.

boldMat: Matrix of filtered BOLD data.

boldMask: BOLD mask.

motionCorr: Motion corrected data.

polyNuis: Polynomial nuisance variables.

timevals: Temporal variables.

nuisance: Nuisance variables.

FD: mean framewise displacement.

badtimes: time frames that are above FD threshold.

dmnAtBOLDres: Default mode network at BOLD resolution in MNI space.

seg2template: Segmentation in BOLD resolution MNI space.

networkPriors2Bold: WIP: standard network priors in BOLD space.

powersLabels: Powers nodes in BOLD space i.e. fusedImg.

powersPoints: Powers points in BOLD space i.e. fusedImg.

connMatNodes: User provided nodal system connectivity matrix.

connMatNodesPartialCorr: User provided nodal system partial correlation matrix.

Author(s)

Avants BB, Duda JT

Examples

```
# this example is long-running ( perhaps 10 minutes on an OSX laptop 2016 )
## Not run:
exrun <- fMRINormalization(verbose = TRUE) # will download ex data
myid <- "BBAvants" # some MRI data
pre <- paste("~/rsfTest/", sep = "")
fn <- list.files(pre,
  full.names = TRUE, recursive = TRUE,
  pattern = glob2rx(paste(myid, "*rsfMRI0.nii.gz", sep = ""))
)
img <- antsImageRead(fn)
sfn <- list.files(pre,
```

```

    full.names = TRUE, recursive = TRUE,
    pattern = glob2rx(paste(myid, "*BrainSegmentation.nii.gz", sep = ""))
  )
  seg <- antsImageRead(sfn)
  t1fn <- list.files(pre,
    full.names = TRUE, recursive = TRUE,
    pattern = glob2rx(paste(myid, "*BrainSegmentation0N4.nii.gz", sep = ""))
  )
  t1 <- antsImageRead(t1fn)
  tt <- fMRINormalization(img,
    repeatMotionEst = 1,
    structuralImage = t1, structuralSeg = seg, verbose = TRUE
  )
  # bold to template
  antsApplyTransforms(mni, getAverageOfTimeSeries(img),
    transformlist = tt$mapsToTemplate$toTemplate,
    whichtoinvert = tt$mapsToTemplate$toTemplateInversion
  )
  # template to bold
  antsApplyTransforms(getAverageOfTimeSeries(img), mni,
    transformlist = tt$mapsToTemplate$toBold,
    whichtoinvert = tt$mapsToTemplate$toBoldInversion
  )

## 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
tr <- 4
for (ftype in c("butt", "stl", "trig")) {
  myres <- frequencyFilterfMRI(fmat, tr = tr, freqLo = 0.01, freqHi = 0.05, opt = ftype)
  comparemat <- cbind(fmat[, k], myres[, k])
  plot(ts(comparemat), main = ftype)
  Sys.sleep(0.3)
  # uncomment below to visualize effect of frequency filtering
  # temp = spectrum( ts(fmat[,k], frequency=1/tr ) )
  # plot( temp$freq, temp$spec, type='l' )
  # temp = spectrum( ts(myres[,k], frequency=1/tr ) )
  # plot( temp$freq, temp$spec, type='l' )
}
```

frob

*Frobenius Norm of a Matrix***Description**

Computes the Frobenius norm of a matrix, defined as the square root of the sum of squared elements.

Usage

```
frob(A)
```

Arguments

A Numeric matrix.

Details

The Frobenius norm is $\sqrt{\sum_{i,j} A_{ij}^2}$, equivalent to the Euclidean norm of the vectorized matrix. Used for scaling and computing residuals in NSA-Flow.

Value

Numeric scalar, the Frobenius norm $\|A\|_F$.

Examples

```
A <- matrix(1:4, 2, 2)
frob(A)
```

fsl2antsrTransform	<i>fsl2antsrTransform</i>
--------------------	---------------------------

Description

Convert an FSL linear transform to an antsrTransform

Usage

```
fsl2antsrTransform(matrix, reference, moving)
```

Arguments

matrix	4x4 matrix of parameters
reference	target image read in as antsImage
moving	image, read in as antsImage

Author(s)

JT Duda

Examples

```
fslMat <- matrix(0, 4, 4)
diag(fslMat) <- rep(1, 4)
fslMat[1:3, 4] <- c(2, 3, 4)
img <- antsImageRead(getANTsRData("ch2"))
tx <- fsl2antsrTransform(fslMat, img, img)
```

functionalLungSegmentation

Ventilation-based segmentation of hyperpolarized gas lung MRI.

Description

Lung segmentation into classes based on ventilation as described in this paper:

Usage

```
functionalLungSegmentation(  
  image,  
  mask,  
  numberOfIterations = 2,  
  numberOfAtroposIterations = 5,  
  mrfParameters = "[0.7,2x2x2]",  
  numberOfClusters = 6,  
  clusterCenters = NA,  
  biasCorrection = "n4",  
  verbose = TRUE  
)
```

Arguments

image	input proton-weighted MRI.
mask	mask image designating the region to segment. 0/1 = background/foreground.
numberOfIterations	number of Atropos <=> bias correction iterations (outer loop).
numberOfAtroposIterations	number of Atropos iterations (inner loop). If numberOfAtroposIterations = 0, this is equivalent to K-means with no MRF priors.
mrfParameters	parameters for MRF in Atropos.
numberOfClusters	number of tissue classes (default = 4)
clusterCenters	initialization centers for k-means
biasCorrection	apply n3, n4, or no bias correction (default = "n4").
verbose	print progress to the screen.

Details

<https://pubmed.ncbi.nlm.nih.gov/21837781/>

Value

segmentation image, probability images, and processed input image.

Author(s)

Tustison NJ

Examples

```
## Not run:  
  
library(ANTSr)
```

```

image <- antsImageRead("lung.nii.gz")
mask <- antsImageRead("mask.nii.gz")
output <- functionalLungSegmentation(image, mask)
antsImageWrite(output$segmentationImage, "outputSegmentation.nii.gz")

## End(Not run)

```

fusedRidge

*Manifold-Regularized Fused Ridge Regression***Description**

Fits a joint Fused Ridge regression model across multiple binarized thresholds, smoothing the coefficient trajectories using a 1D Graph Laplacian.

Usage

```

fusedRidge(
  X_pcs,
  y_raw,
  thresholds,
  covariates = NULL,
  lambda1 = 0.5,
  lambda2 = 0.5,
  family = "binomial",
  standardize = TRUE,
  foldid = NULL,
  topK = NULL,
  thresh = 1e-04,
  nlambdas = 100,
  nfolds = 10,
  cv = TRUE,
  lambda = NULL,
  ...
)

```

Arguments

<code>X_pcs</code>	Matrix of predictor variables. For high-dimensional data (e.g., voxels), it is recommended to use dimensionality reduction (e.g., PCA) first, as the internal Kronecker product can be memory-intensive.
<code>y_raw</code>	Vector of continuous exposure or response variable.
<code>thresholds</code>	Vector of thresholds at which to binarize <code>y_raw</code> .
<code>covariates</code>	Optional matrix or data frame of covariates to adjust for. Covariates are unpenalized. An intercept is automatically added if not present.
<code>lambda1</code>	Ridge penalty weight (defaults to 0.5).

lambda2	Fusion penalty weight (defaults to 0.5).
family	Model family, passed to glmnet. Defaults to "binomial".
standardize	Boolean indicating whether to scale the predictor matrix. Defaults to TRUE.
foldid	Optional fold IDs for group-structured cross-validation.
topK	Optional integer. If provided, fits a first pass model, selects the union of the topK largest absolute feature weights across all thresholds, and refits the model using only those features (enables sparsity control via two-pass refitting).
thresh	Convergence threshold for glmnet.
nlambda	Number of lambda values to use in glmnet (defaults to 100).
nfolds	Number of folds for cross-validation (defaults to 10).
cv	Boolean indicating whether to select the Ridge lambda by <code>glmnet::cv.glmnet</code> . If FALSE and lambda is NULL, a lambda of 1 is used. Defaults to TRUE.
lambda	Optional numeric value. If provided, CV is skipped and the model is fit directly using the specified lambda value.
...	Additional arguments passed to <code>glmnet::cv.glmnet</code> and <code>glmnet::glmnet</code> .

Details

Fits a joint Fused Ridge regression model across multiple binarized thresholds of a continuous response variable, smoothing the coefficient trajectories using a 1D Graph Laplacian.

The model is formulated by transforming the predictor matrix using a transformation matrix $H = (\lambda_1 I + \lambda_2 L)^{-1/2}$, where L is a 1D Graph Laplacian matrix. The design matrix is stacked block-diagonally for unpenalized covariates and using Kronecker products for penalized features: $Z = H \otimes X$. This transformation maps the fused ridge problem back into standard ridge regression, allowing the use of highly optimized solvers like glmnet.

Value

A list containing the fitted glmnet model, cross-validation results, optimal lambda, reconstructed full coefficient matrix, and other helper parameters.

References

- Lettink, A., Chinapaw, M. J. M., & van Wieringen, W. N. (2023). Two-dimensional fused targeted ridge regression for health indicator prediction from accelerometer data. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 72(4), 1064-1078. doi:10.1093/jrsssc/qlad038
- Li, H., & Li, Y. (2008). Network-constrained regularization and variable selection for analysis of genomic data. *Bioinformatics*, 24(9), 1175-1182. doi:10.1093/bioinformatics/btn081

fusedRidgeDirect

*Direct Manifold-Regularized Fused Ridge Regression (General)***Description**

Fits a joint Fused Ridge regression model across multiple binarized thresholds of a continuous response variable using direct optimization (L-BFGS). This avoids the memory-intensive Kronecker product design matrix expansion used in fusedRidge, making it highly scalable for high-dimensional predictors.

Usage

```
fusedRidgeDirect(
  X_pcs,
  y_raw,
  thresholds,
  covariates = NULL,
  lambda1 = 0.5,
  lambda2 = 0.5,
  family = "binomial",
  standardize = TRUE,
  foldid = NULL,
  topK = NULL,
  thresh = 1e-04,
  nlambdas = 20,
  nfolds = 10,
  cv = TRUE,
  ncores = 1,
  optim_control = list(maxit = 200),
  lambda = NULL,
  ...
)
```

Arguments

X_pcs	Matrix of predictor variables.
y_raw	Vector of continuous exposure or response variable.
thresholds	Vector of thresholds at which to binarize y_raw.
covariates	Optional matrix or data frame of covariates to adjust for. Covariates are unpenalized. An intercept is automatically added if not present.
lambda1	Ridge penalty weight (defaults to 0.5).
lambda2	Fusion penalty weight (defaults to 0.5).
family	Model family. Supported values are "binomial", "gaussian", and "poisson". Defaults to "binomial".
standardize	Boolean indicating whether to scale the predictor matrix. Defaults to TRUE.

<code>foldid</code>	Optional fold IDs. If <code>cv = TRUE</code> , used for group-structured cross-validation.
<code>topK</code>	Optional integer. If provided, fits a first pass model, selects the union of the topK largest absolute feature weights across all thresholds, and refits the model using only those features.
<code>thresh</code>	Convergence threshold. Ignored (provided for compatibility with <code>fusedRidge</code>).
<code>nlambda</code>	Number of lambda values to use in the cross-validation grid. Defaults to 20.
<code>nfolds</code>	Number of folds for cross-validation (defaults to 10).
<code>cv</code>	Boolean indicating whether to perform pathwise group-structured cross-validation to select the optimal overall regularization scale. Defaults to <code>TRUE</code> .
<code>ncores</code>	Number of CPU cores to use for cross-validation. Defaults to 1 (sequential processing). Ignored on Windows.
<code>optim_control</code>	Optional list of control parameters passed to <code>optim</code> . Defaults to <code>list(maxit = 200)</code> .
<code>lambda</code>	Optional numeric value. If provided, CV is skipped and the model is fit directly using the specified lambda value.
<code>...</code>	Additional arguments passed to <code>optim</code> .

Details

Fits a joint Fused Ridge regression model across multiple binarized thresholds of a continuous response variable, smoothing the coefficient trajectories using a 1D Graph Laplacian. Unlike `fusedRidge`, which transforms the design matrix using a Kronecker product expansion, `fusedRidgeDirect` optimizes the loss function directly using R's built-in `optim` function with the "L-BFGS-B" method and analytical gradients.

If `cv = TRUE`, it performs pathwise group-structured cross-validation (with warm starts) to select the optimal overall scale multiplier for the penalties, matching the cross-validation behavior of `fusedRidge`.

Value

A list of class `c("fusedRidgeDirect", "fusedRidge")` containing:

<code>fit</code>	The final fitted <code>optim</code> object (or list of options).
<code>cv</code>	Cross-validation details including grid values, mean validation errors, and standard deviations.
<code>optimal_lambda</code>	Optimal lambda multiplier selected by cross-validation (or NA if <code>cv = FALSE</code>).
<code>a0</code>	0 (intercepts are included in <code>coefs_covs</code>).
<code>coefs_covs</code>	Reconstructed covariate coefficients matrix.
<code>coefs_full</code>	Reconstructed full coefficient matrix.
<code>theta_matrix</code>	Coefficients in the original space.
<code>H</code>	Identity matrix placeholder.
<code>thresholds</code>	Input thresholds vector.
<code>family</code>	Model family.

y_matrix	Binarized target matrix.
mean_X	Scaling center parameters.
sd_X	Scaling scale parameters.
covs_names	Names of covariates.

fusedRidgeTorch

PyTorch-Based Manifold-Regularized Fused Ridge Regression

Description

Fits a joint Fused Ridge regression model across multiple binarized thresholds of a continuous response variable using PyTorch L-BFGS and automatic differentiation. This avoids the memory-intensive Kronecker product design matrix expansion, and leverages PyTorch's highly optimized solvers and potential hardware acceleration (MPS/CUDA).

Usage

```
fusedRidgeTorch(
  X_pcs,
  y_raw,
  thresholds,
  covariates = NULL,
  lambda1 = 0.5,
  lambda2 = 0.5,
  family = "binomial",
  standardize = TRUE,
  foldid = NULL,
  topK = NULL,
  thresh = 1e-04,
  nlambda = 20,
  nfolds = 10,
  cv = TRUE,
  device = "cpu",
  optim_control = list(maxit = 200),
  alpha = 0,
  sparsity_thresh = 1e-04,
  lambda = NULL,
  ...
)
```

Arguments

X_pcs	Matrix of predictor variables.
y_raw	Vector of continuous exposure or response variable.
thresholds	Vector of thresholds at which to binarize y_raw.

covariates	Optional matrix or data frame of covariates to adjust for. Covariates are unpenalized. An intercept is automatically added if not present.
lambda1	Ridge penalty weight (defaults to 0.5).
lambda2	Fusion penalty weight (defaults to 0.5).
family	Model family. Supported values are "binomial", "gaussian", and "poisson". Defaults to "binomial".
standardize	Boolean indicating whether to scale the predictor matrix. Defaults to TRUE.
foldid	Optional fold IDs. If cv = TRUE, used for group-structured cross-validation.
topK	Optional integer. If provided, fits a first pass model, selects the union of the topK largest absolute feature weights across all thresholds, and refits the model using only those features.
thresh	Convergence threshold.
nlambda	Number of lambda values to use in the cross-validation grid. Defaults to 20.
nfolds	Number of folds for cross-validation (defaults to 10).
cv	Boolean indicating whether to perform pathwise group-structured cross-validation to select the optimal overall regularization scale. Defaults to TRUE.
device	PyTorch device. Can be "cpu", "cuda", "mps", or "auto" (which detects CUDA or MPS availability). Defaults to "cpu".
optim_control	Optional list of control parameters passed to the PyTorch optimizer. Defaults to <code>list(maxit = 200)</code> .
alpha	Elastic net mixing parameter between 0 and 1. $\alpha = 0$ is pure ridge, $\alpha = 1$ is pure lasso. Defaults to 0.
sparsity_thresh	Hard threshold for sparsity post-optimization. Any coefficient smaller in absolute value is zeroed out. Defaults to $1e-4$.
lambda	Optional numeric value. If provided, CV is skipped and the model is fit directly using the specified lambda value.
...	Additional arguments (currently ignored).

Value

A list of class `c("fusedRidgeTorch", "fusedRidgeDirect", "fusedRidge")` containing:

fit	Details from the final PyTorch fit.
cv	Cross-validation details including grid values, mean validation errors, and standard deviations.
optimal_lambda	Optimal lambda multiplier selected by cross-validation (or NA if cv = FALSE).
a0	0 (intercepts are included in <code>coefs_covs</code>).
coefs_covs	Reconstructed covariate coefficients matrix.
coefs_full	Reconstructed full coefficient matrix.
theta_matrix	Coefficients in the original space.
H	Identity matrix placeholder.

thresholds	Input thresholds vector.
family	Model family.
y_matrix	Binarized target matrix.
mean_X	Scaling center parameters.
sd_X	Scaling scale parameters.
covs_names	Names of covariates.

fuzzySpatialCMeansSegmentation
fuzzySpatialCMeansSegmentation

Description

Fuzzy spatial c-means for image segmentation.

Usage

```
fuzzySpatialCMeansSegmentation(
  image,
  mask = NULL,
  numberOfClusters = 4,
  m = 2,
  p = 1,
  q = 1,
  radius = 2,
  maxNumberOfIterations = 20,
  convergenceThreshold = 0.02,
  verbose = FALSE
)
```

Arguments

image	image to be segmented.
mask	optional mask image. Otherwise, the entire image is used.
numberOfClusters	number of segmentation clusters
m	fuzziness parameter (default = 2).
p	membership importance parameter (default = 1).
q	spatial constraint importance parameter (default = 1). $q = 0$ is equivalent to conventional fuzzy c-means.
radius	neighborhood radius (scalar or array) for spatial constraint.
maxNumberOfIterations	iteration limit (default = 20).

convergenceThreshold
Convergence between iterations is measured using the Dice coefficient (default = 0.02).

verbose
print progress.

Details

Image segmentation using fuzzy spatial c-means as described in Chuang et al., Fuzzy c-means clustering with spatial information for image segmentation. CMIG: 30:9-15, 2006.

Value

list containing segmentation and probability images

Author(s)

NJ Tustison

Examples

```
image <- antsImageRead(getANTsRData("r16"))
mask <- getMask(image)
fuzzy <- fuzzySpatialCMeansSegmentation(image, mask)
```

generate_structured_multiview_data

Generate a Sophisticated Simulated Multi-View Dataset

Description

This function creates simulated multi-view data with a complex and realistic ground truth, containing both shared and modality-specific latent signals.

Usage

```
generate_structured_multiview_data(
  n_subjects,
  n_features,
  k_shared,
  k_specific,
  noise_sd = 0.1,
  sparsity_level = 0.5,
  seed = NULL
)
```

Arguments

n_subjects	The number of subjects (rows).
n_features	A vector of integers specifying the number of features for each view.
k_shared	The number of latent components common to ALL views.
k_specific	A vector of integers specifying the number of unique latent components for EACH view. If a single number is given, it's recycled.
noise_sd	The standard deviation of the Gaussian noise added to the data.
sparsity_level	The proportion of elements in the shared loading matrix (V_f) to set to zero. A value between 0 and 1. Defaults to 0.5.
seed	An optional random seed for reproducibility.

Value

A list containing:

data_list	A named list of the final [n x p] data matrices.
ground_truth	A list containing the true latent structures: <ul style="list-style-type: none"> • 'U_shared': The ground truth shared basis [n x k_shared]. This is what 'simlr' should aim to recover. • 'V_shared': A list of the ground truth shared loading matrices [p x k_shared]. This is what the 'simlr' 'v' matrices should be compared against. • 'U_specific': A list of the ground truth modality-specific bases. • 'U_combined': The full latent basis [n x (k_shared + sum(k_specific))] that was used to generate the data.

generateAifMaskImage	<i>Automatically generate an arterial input function mask.</i>
----------------------	--

Description

Automated approach for generating a mask to be used for modeling the arterial input function using the method described in <https://pubmed.ncbi.nlm.nih.gov/15956519>.

Usage

```
generateAifMaskImage(perfusionImage, voiMaskImage, index = NA)
```

Arguments

perfusionImage	time series (n-D + time) perfusion acquisition.
voiMaskImage	n-D mask image indicating where the cerebral blood flow parameter images are calculated.
index	If the user prefers a specific voxel if looking at the plots, they can simply specify the index (gleaned from the file names) and the corresponding AIF mask image is created.

Value

list(mask image, fitting results)

Author(s)

Tustison NJ

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 = NULL,
  beta = 0.9
)
```

Arguments

img	input image or list of images (multiple features) where 1st image would typically be the primary contrast
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,
  verbose = FALSE,
  method = ifelse(Sys.info()["sysname"] == "Linux", "wget", "auto"),
  quiet = 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
verbose	optional boolean
method	Method to be used for downloading files, passed to download.file
quiet	If TRUE, suppress status messages (if any), and the progress bar.

Value

filename string

Author(s)

Avants BB

Examples

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

```
getAntsrTransformFixedParameters  
  getAntsrTransformFixedParameters
```

Description

Get fixed parameters of transform

Usage

```
getAntsrTransformFixedParameters(transform)
```

Arguments

transform antsrTransform

Value

array of fixed parameters

Examples

```
tx <- new("antsrTransform")  
params <- getAntsrTransformFixedParameters(tx)
```

```
getAntsrTransformParameters
      getAntsrTransformParameters
```

Description

Get parameters of transform

Usage

```
getAntsrTransformParameters(transform)
```

Arguments

transform antsrTransform

Value

array of parameters'

Examples

```
tx <- new("antsrTransform")
params <- getAntsrTransformParameters(tx)
```

```
getASLNoisePredictors    Get nuisance predictors from ASL images
```

Description

Get nuisance predictors from ASL images

Usage

```
getASLNoisePredictors(
  aslmat,
  tc,
  noisefrac = 0.1,
  polydegree = "loess",
  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, with a value of 0 indicating no detrending, or 'loess' for LOESS-based estimation of global time-series trends.
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)
mask <- getMask(getAverageOfTimeSeries(img))
aslmat <- timeseries2matrix(img, mask)
tc <- rep(c(0.5, -0.5), length.out = nrow(aslmat))
noise <- getASLNoisePredictors(aslmat, tc, k = 2, npreds = 2, noisefrac = 0.5)
cm <- colMeans(noise)
rounding_type <- RNGkind()[3]
if (getRversion() < "3.6.0" || rounding_type == "Rounding") {
  testthat::expect_equal(cm, c(-0.223292128499263, 0.00434481670243642), tolerance = .01)
} else {
  testthat::expect_equal(cm, c(-0.223377249912075, 0.0012754214030999), tolerance = .01)
}
```

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)
```

getCenterOfMass	<i>center of mass</i>
-----------------	-----------------------

Description

compute an image center of mass in physical space which is defined as the mean of the intensity weighted voxel coordinate system.

Usage

```
getCenterOfMass(img)
```

Arguments

img antsImage

Value

vector center of mass

Author(s)

Brian B. Avants

Examples

```
fi <- antsImageRead(getANTsRData("r16"))
com1 <- getCenterOfMass(fi)
fi <- antsImageRead(getANTsRData("r64"))
com2 <- getCenterOfMass(fi)
```

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 = 0)
```

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

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)
```

getfMRIInuisanceVariables

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

```
getfMRIInuisanceVariables(fmri, maskThresh = 500, moreaccurate = 1, mask = NULL)
```

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 input 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 <- getfMRIInuisanceVariables(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 <- ri(1)
mask <- getMask(img)
testthat::expect_error(getMask(img, lowThresh = "hey"))
pixeltype(img) <- "unsigned int"
mask2 <- getMask(img)
```

getMultiResFeatureMatrix

build multiple resolution predictor matrix from feature images

Description

Represents feature images as a neighborhood across scales. each subject gets a label image and feature list. these labels/features should be the same type for all subjects. e.g each subject has a k-label image where the labels cover the same anatomy and the feature images are the same. for each label in a mask, produce a multi-resolution neighborhood sampling from the data within the label, for a given feature. do this for each feature. so, 2 labels will double the width of the predictor matrix. an additional scale parameter or feature will increase the width. more samples will increase the number of rows in the predictor matrix. the labels allow one to collect predictors side-by-side from different parts of an image-based feature set. future work will allow other covariates.

Usage

```
getMultiResFeatureMatrix(
  x,
  labelmask,
  rad = NA,
  multiResSchedule = c(0),
  nsamples = 10
)
```

Arguments

x	a list of feature images
labelmask	the mask defines the image space for the associated feature and increases the number of predictors. more labels means more predictors. a different mask may be used for each feature, if desired, in which case this should be a list of masks where the length of the list is equal to the number of features.
rad	vector of dimensionality d define nhooood radius
multiResSchedule	a vector of smoothing values
nsamples	defines the number of samples to take for each label.

Value

mat a matrix of predictors with n samples rows

Author(s)

Avants BB, Tustison NJ

Examples

```
img <- antsImageRead(getANTsRData("r16"))
seg <- kmeansSegmentation(img, 3)$segmentation
flist <- list(img, img %>% iMath("Grad"))
featMat <- getMultiResFeatureMatrix(flist, seg,
  rad = c(1, 1),
  multiResSchedule = c(2, 1), nsamples = 10
)
```

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 = NULL,
  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)
```

getNamesFromDataframe *Extract column names with concatenated search parameters*

Description

Extract column names with concatenated search parameters

Usage

```
getNamesFromDataframe(x, demogIn, exclusions)
```

Arguments

x	vector of strings
demogIn	the dataframe with column names to search.
exclusions	the strings to exclude

Value

vector of string column names

Author(s)

Avants BB

Examples

```
# nms = getNamesFromDataframe( c("sp","ed"), cars )
```

```
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

values numeric vector of values

indices matrix 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)
randlist <- getNeighborhoodAtVoxel(img, center, kernel,
  physical.coordinates = TRUE
)
arr <- as.array(img)
testthat::expect_error(getNeighborhoodAtVoxel(arr, center, kernel), "class")
testthat::expect_error(
  getNeighborhoodAtVoxel(img, as.character(center), kernel),
  "center must be"
)
testthat::expect_error(
  getNeighborhoodAtVoxel(img, center, c(radius, 3)),
  "kernel must have same"
)
```

getNeighborhoodInMask *Get neighborhoods for voxels within mask*

Description

this converts a scalar image to a matrix with rows that contain neighbors around a center voxel

Usage

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

Arguments

image	image object of S4 class <code>antsImage</code> to get values from.
mask	image object of S4 class <code>antsImage</code> 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)

<code>physical.coordinates</code>	logical indicating if voxel indices and offsets should be in voxel or physical coordinates
<code>boundary.condition</code>	string indicating how to handle voxels in a neighborhood, but not in the mask. See Details.
<code>spatial.info</code>	a boolean indicating if voxel locations and neighborhood offsets should be returned along with pixel values.
<code>get.gradient</code>	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 mean 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)
mat <- getNeighborhoodInMask(r16, mask, radius,
  boundary.condition = "image"
)
mat <- getNeighborhoodInMask(r16, mask, radius,
  boundary.condition = "mean"
)
randlist <- getNeighborhoodInMask(r16, mask, radius,
  physical.coordinates = TRUE
```

```

)
arr <- as.array(r16)
testthat::expect_error(getNeighborhoodInMask(arr, mask, radius), "antsImage")
testthat::expect_error(
  getNeighborhoodInMask(r16, as.numeric(mask), radius),
  "mask must be"
)
testthat::expect_error(
  getNeighborhoodInMask(r16, mask, as.character(radius)),
  "radius must be"
)
# testthat::expect_error(getNeighborhoodInMask(r16,mask,c(radius, 3)),
# "Radius must")

```

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 = NULL,  
  outprefix = NA,  
  convertToTal = FALSE  
)
```

Arguments

<code>imagePairToBeLabeled</code>	e.g. the template image and the activation map
<code>templatePairWithLabels</code>	e.g. the mni image and brodmann label set
<code>labelnames</code>	a list of names for the labels
<code>outprefix</code>	if output to a file, provide file prefix
<code>convertToTal</code>	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 rorden's mricron
# but you can do something with any other image
# e.g. a statistical image
#
tem <- antsImageRead(getANTsRData("ch2"), 3)
clust <- antsImageClone(tem)
atem <- as.array(tem)
clust[atem < 80] <- 0
clust[atem > 90] <- 0
clust[atem > 80 & atem < 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)
```

Description

Impute missing data using GLM models

Usage

```
glm_impute(
  dataframe,
  columns_to_impute,
  predictor_columns,
  family = "gaussian"
)
```

Arguments

`dataframe` A data frame containing the data to impute.

`columns_to_impute` A vector of column names to impute.

`predictor_columns` A vector of column names to use as predictors.

`family` A string specifying the GLM family (default is 'gaussian').

Value

A data frame with imputed values.

Examples

```
set.seed(123)
df <- data.frame(
  age = c(25, 30, 35, NA, 45, 50, NA, 40, 35, NA),
  income = c(50000, 60000, 70000, 80000, 90000, 100000, 110000, NA, 120000, 130000),
  education = c(12, 16, 14, 12, NA, 18, 20, 16, 14, 12)
)
columns_to_impute <- c("age")
predictor_columns <- c("income", "education")
imputed_data <- glm_impute(df, columns_to_impute, predictor_columns, family = 'gaussian')
print(imputed_data)
```

gradient_invariant_orthogonality_defect

Compute the Gradient of the Orthogonality Defect

Description

This function computes the gradient of the orthogonality defect for a matrix A. The orthogonality defect is defined as the sum of the squared off-diagonal elements of $t(A) \%*\% A$.

Usage

```
gradient_invariant_orthogonality_defect(A)

gradient_invariant_orthogonality_defect(A)
```

Arguments

A A numeric matrix.

Value

A matrix representing the gradient of the orthogonality defect with respect to A_p .

gradient_invariant_orthogonality_salad

Gradient of the Invariant Orthogonality Defect Measure

Description

This function computes the gradient of the orthogonality defect measure with respect to the input matrix 'A'. The gradient is useful for optimization techniques that require gradient information. The gradient will be zero for matrices where ' $A^t A$ ' equals the diagonal matrix 'D'.

Usage

```
gradient_invariant_orthogonality_salad(A)
```

Arguments

A A numeric matrix.

Value

A numeric matrix representing the gradient of the orthogonality defect measure.

Examples

```
# library(salad)
# A <- matrix(runif(20), nrow = 10, ncol = 2)
# gradient_invariant_orthogonality_salad(A)
```

hausdorffDistance	<i>hausdorffDistance</i>
-------------------	--------------------------

Description

Wrapper for the ITK class HausdorffDistanceImageFilter. More documentaiton available here:

Usage

```
hausdorffDistance(inputImage1, inputImage2)
```

Arguments

- inputImage1 image 1.
- inputImage2 image 2.

Details

http://docs.itk.org/projects/doxygen/en/stable/classitk_1_1HausdorffDistanceImageFilter.html

Value

data frame with overlap measures

Author(s)

Tustison NJ

Examples

```
sourceImage <- antsImageRead(getANTsRData("r16"), 2)
sourceSegmentation <- kmeansSegmentation(sourceImage, 3)$segmentation
referenceImage <- antsImageRead(getANTsRData("r16"), 2)
referenceSegmentation <- kmeansSegmentation(referenceImage, 3)$segmentation
hausdorff <- hausdorffDistance(sourceImage, referenceImage)
```

hemisphere	<i>hemisphere</i>
------------	-------------------

Description

hemisphere

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 subtracted 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)

```

hessianObjectness	<i>hessianObjectness</i>
-------------------	--------------------------

Description

Interface to ITK filter. Based on the paper by Westin et al., "Geometrical Diffusion Measures for MRI from Tensor Basis Analysis" and Luca Antiga's Insight Journal paper <http://hdl.handle.net/1926/576>.

Usage

```
hessianObjectness(
    image,
    objectDimension = 1,
    isBrightObject = TRUE,
    sigmaMin = 0.1,
    sigmaMax = 10,
    numberOfSigmaSteps = 10,
    useSigmaLogarithmicSpacing = TRUE,
    alpha = 0.5,
    beta = 0.5,
    gamma = 5,
    setScaleObjectnessMeasure = TRUE
)
```

Arguments

image	input image
objectDimension	0: 'sphere', 1: 'line', or 2: 'plane'.
isBrightObject	Set 'true' for enhancing bright objects and 'false' for dark objects.
sigmaMin	Define scale domain for feature extraction.
sigmaMax	Define scale domain for feature extraction.
numberOfSigmaSteps	Define number of samples for scale space.
useSigmaLogarithmicSpacing	Define sample spacing the for scale space.
alpha	Hessian filter parameter.
beta	Hessian filter parameter.
gamma	Hessian filter parameter.
setScaleObjectnessMeasure	...

Value

hessian objectness image.

Author(s)

NJ Tustison

Examples

```
image <- antsImageRead(getANTsRData("r16"))
hessianObjectnessImage <- hessianObjectness(image)
```

hist.antsImage	<i>Histogram of Values in an Image</i>
----------------	--

Description

Computes and displays a histogram of the values of an image with the option for a mask.

Usage

```
## S3 method for class 'antsImage'
hist(x, ..., mask)
```

Arguments

x	Object of class antsImage
...	Arguments passed to hist.default
mask	object to subset the image. If missing, then all values of the image are plotted.

Value

Output of [hist](#)

Examples

```
img <- makeImage(c(10, 10), rnorm(100))
mask <- img > 0
hist(img)
hist(img, mask = mask)
```

```
histogramEqualizeImage
```

histogramEqualizeImage

Description

Histogram equalize image

Usage

```
histogramEqualizeImage(image, numberOfHistogramBins = 256)
```

Arguments

`image` image to undergo intensity transformation.
`numberOfHistogramBins` number of histogram levels.

Value

equalized image.

Author(s)

NJ Tustison

Examples

```
image <- antsImageRead(getANTsRData("r16"), 2)
eqImage <- histogramEqualizeImage(image)
```

```
histogramMatchImage      histogramMatchImage
```

Description

Match intensity profile with a reference image.

Usage

```
histogramMatchImage(
  sourceImage,
  referenceImage,
  numberOfHistogramBins = 255,
  numberOfMatchPoints = 64,
  useThresholdAtMeanIntensity = FALSE
)
```

Arguments

sourceImage image to undergo intensity transformation.
referenceImage image providing reference intensity profile.
numberOfHistogramBins
 number of histogram levels.
numberOfMatchPoints
 number of histogram match points.
useThresholdAtMeanIntensity
 use a simple background exclusion criterion.

Value

source image intensity matched to reference image.

Author(s)

NJ Tustison

Examples

```
sourceImage <- antsImageRead(getANTsRData("r16"), 2)
referenceImage <- antsImageRead(getANTsRData("r64"), 2)
matchedImage <- histogramMatchImage(sourceImage, referenceImage)
```

histogramMatchImage2 *Transform image intensities based on histogram mapping.*

Description

Apply B-spline 1-D maps to an input image for intensity warping.

Usage

```
histogramMatchImage2(  
  sourceImage,  
  referenceImage,  
  sourceMask = NULL,  
  referenceMask = NULL,  
  matchPoints = 64,  
  transformDomainSize = 255  
)
```

Arguments

sourceImage source image.
referenceImage reference image.
sourceMask source mask.
referenceMask reference mask.
matchPoints parametric points at which the intensity transform displacements are specified between [0, 1]. Alternatively, a single number can be given and the sequence is linearly spaced in [0, 1].
transformDomainSize
 Defines the sampling resolution of the B-spline warping.

Value

warped intensity image

Author(s)

Tustison NJ

Examples

```
library( ANTsR )
sourceImage <- antsImageRead(getANTsRData("r16"), 2)
referenceImage <- antsImageRead(getANTsRData("r64"), 2)
matchedImage <- histogramMatchImage2(sourceImage, referenceImage)
```

histogramWarpImageIntensities

Transform image intensities based on histogram mapping.

Description

Apply B-spline 1-D maps to an input image for intensity warping.

Usage

```
histogramWarpImageIntensities(
  image,
  breakPoints = c(0.25, 0.5, 0.75),
  displacements = NULL,
  clampEndPoints = c(FALSE, FALSE),
  sdDisplacements = 0.05,
  transformDomainSize = 20
)
```

Arguments

image	input image.
breakPoints	parametric points at which the intensity transform displacements are specified between [0, 1]. Alternatively, a single number can be given and the sequence is linearly spaced in [0, 1].
displacements	displacements to define intensity warping. Length must be equal to the breakPoints. Alternatively, if NULL random displacements are chosen (random normal: mean = 0, sd = sdDisplacements).
clampEndPoints	specify non-zero intensity change at the ends of the histogram.
sdDisplacements	characterize the randomness of the intensity displacement.
transformDomainSize	Defines the sampling resolution of the B-spline warping.

Value

warped intensity image

Author(s)

Tustison NJ

Examples

```
library( ANTsR )
image <- antsImageRead( getANTsRData( "r16" ) )
transformedImage <- histogramWarpImageIntensities( image, transformDomainSize = 10 )
rm(image); gc()
rm(transformedImage); gc()
```

humanize_anatomical_name

Humanize anatomical names by expanding abbreviations and reordering

Description

Humanize anatomical names by expanding abbreviations and reordering

Usage

```
humanize_anatomical_name(x)
```

Arguments

x	character string
---	------------------

Value

humanized character string

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))
multismoo <- multismoo %>% iBind(smoothImage(fi, 4))
```

icawhiten

*Simple icawhitening function.***Description**

Icawhiten 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)
```

ilr

Efficiently compute an image-based linear regression model (ilr)

Description

This function simplifies calculating p-values from linear models in which there is a similar formula that is applied many times with a change in image-based predictors. Image-based variables are stored in the input matrix list. They should be named consistently in the input formula and in the image list. If they are not, an error will be thrown. All input matrices should have the same number of rows and columns.

Usage

```
ilr(dataFrame, voxmats, myFormula)
```

Arguments

dataFrame	This data frame contains all relevant predictors except for the matrices associated with the image variables.
voxmats	The named list of matrices that contains the changing predictors.
myFormula	This is a character string that defines a valid regression formula.

Value

A list of different matrices that contain names derived from the formula and the coefficients of the regression model.

Author(s)

BB Avants.

Examples

```
set.seed(1500)
nsub <- 100
outcome <- rnorm(nsub)
covar <- rnorm(nsub)
mat <- replicate(nsub, rnorm(nsub))
mat2 <- replicate(nsub, rnorm(nsub))
myform <- " outcome ~ covar + vox "
df <- data.frame(outcome = outcome, covar = covar)
result <- ilr(df, list(vox = mat), myform)
print(names(result))
print(rownames(result$pValue))
myform <- " vox2 ~ covar + vox "
df <- data.frame(outcome = outcome, covar = covar)
result <- ilr(df, list(vox = mat, vox2 = mat2), myform)
print(names(result))
print(rownames(result$pValue))
```

ilr.predict

Predict from ilr output

Description

This function computes a prediction given ilr output.

Usage

```
ilr.predict(ilrResult, dataFrame, voxmats, myFormula)
```

Arguments

ilrResult	This output from ilr
dataFrame	This data frame contains all relevant predictors except for the matrices associated with the image variables.
voxmats	The named list of matrices that contains the changing predictors.
myFormula	This is a character string that defines a valid regression formula.

Value

the predicted matrix.

Author(s)

BB Avants.

See Also

[ilr](#)

Examples

```
set.seed(1500)
nsub <- 100
trte <- sample(1:nsub)[1:70]
covarIn <- rnorm(nsub)
mat <- replicate(nsub, rnorm(nsub))
mat2 <- replicate(nsub, rnorm(nsub))
outcomeAtVox <- mat[, 2]
# also try myform = " vox ~ covar + vox2 "
myform <- "outcome ~ covar + vox + vox2"
df <- data.frame(outcome = outcomeAtVox, covar = covarIn)
result <- ilr(df[trte, ], list(vox = mat[trte, ], vox2 = mat2[trte, ]), myform)
df2 <- data.frame(covar = covarIn[-trte])
pred <- ilr.predict(result, df2, list(vox = mat[-trte, ], vox2 = mat2[-trte, ]), myform)
max(cor(pred, outcomeAtVox[-trte]))
```

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)
```

`imageDomainToSpatialMatrix`*Image domain to spatial position*

Description

Transform a d-dimensional image into a d by p matrix that encodes the spatial coordinates of the image domain.

Usage

```
imageDomainToSpatialMatrix(image, mask)
```

Arguments

image	input image
mask	optional input mask

Value

matrix is output

Author(s)

Avants BB

Examples

```
img <- antsImageRead(getANTsRData("r16"))
spmat <- imageDomainToSpatialMatrix(img)
```

```
imageFileNames2ImageList
```

Simple imageFileNames2ImageListing function.

Description

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

Usage

```
imageFileNames2ImageList(x)
```

Arguments

x input file name list

Value

a list containing the images this : mylist<-list(img1, img2 , etcetera) is the

Author(s)

Avants BB, Kandel BM

Examples

```
dir <- paste(tempdir(), .Platform$file.sep, sep = "")
img <- makeImage(c(4, 4))
for (ii in 1:6) {
  antsImageWrite(img, paste(dir, "image", ii, ".nii.gz", sep = ""))
}
imagenames <- list.files(dir, glob2rx("*.nii.gz"), full.names = TRUE)
images <- imageFileNames2ImageList(imagenames)
```

```
imageListToMatrix
```

Read Images into a Matrix

Description

Read images into rows of a matrix.

Usage

```
imageListToMatrix(
  imageList,
  mask,
  sigma = NA,
  epsilon = 0,
  asymmetryTx = NULL,
  asymmetryMask = NULL
)
```

Arguments

imageList	A character vector containing a list of image files to read, in order - these are image objects, not file names. these are assumed to be in a normalized space i.e. they are all in registration.
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. If the mask is a different size than the image, the images will be downsampled and smoothed to the size of the mask.
sigma	smoothing operation in physical space. See smoothImage.
epsilon	threshold value determining what is included in the mask
asymmetryTx	a reflection transform
asymmetryMask	a mask defining left and right side of the image. this should be defined in the space of the images within the imageList.

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 <- ri(1) %>% resampleImage(c(32, 32))
imglist <- list()
nsubj <- 3
for (ii in 1:nsubj) {
  imglist[[ii]] <- img
}
mask <- getMask(img)
imgmat <- imageListToMatrix(imglist, mask)
```

imageSimilarity	<i>imageSimilarity</i>
-----------------	------------------------

Description

measure similarity between two images

Usage

```
imageSimilarity(
  fixed,
  moving,
  type = c("MeanSquares", "MattesMutualInformation", "ANTSNeighborhoodCorrelation",
    "Correlation", "Demons", "JointHistogramMutualInformation"),
  fixed.mask = NULL,
  moving.mask = NULL,
  sampling.strategy = "none",
  sampling.percentage = 1,
  nBins = 32,
  radius = 3
)
```

Arguments

fixed	the fixed antsImage
moving	the moving antsImage
type	image metric to calculate
	MeanSquares
	Correlation
	ANTSNeighborhoodCorrelation
	MattesMutualInformation
	JointHistogramMutualInformation
	Demons
fixed.mask	mask for the fixed image
moving.mask	mask for the moving image
sampling.strategy	sampling strategy, default if full sampling
	none Full sampling
	random
	regular
sampling.percentage	percentage of data to sample when calculating metric
nBins	the number of bins for histogram metrics
radius	neighborhood radius of ANTSNeighborhoodCorrelation

Value

value of image to image metric

Examples

```
x <- antsImageRead(getANTsRData("r16"))
y <- antsImageRead(getANTsRData("r30"))
metric <- imageSimilarity(x, y, type = "MeanSquares")
```

imagesToMatrix	<i>Read Images into a Matrix</i>
----------------	----------------------------------

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/s. Additional parameters should be specific for each operation. See the the full iMath in ANTs, on which this function is based.

Usage

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

Arguments

<code>img</code>	input object, usually <code>antsImage</code>
<code>operation</code>	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.
<code>param</code>	... additional parameters
<code>...</code>	further parameter options

Author(s)

JT Duda

Examples

```

fi <- antsImageRead(getANTsRData("r16"), 2)
mask <- getMask(fi)
op1 <- iMath(fi, "GD", 1) # gray matter dilation by 1 voxel
op2 <- iMath(mask, "D") # distance transform
op3 <- iMath(0, "GetOperations") # list all ops

if (usePkg("magrittr")) { # string ops together
  lapgd <- fi %>%
    iMath("Laplacian", 1) %>%
    iMath("GD", 3)
}
# Canny is useful e.g.
# canny = ( iMath(myreg$warpedmovout,"Normalize")*255 ) %>% iMath("Canny",1,5,12)
tf <- getANTsRData("r27")
tem <- antsImageRead(tf)
mask <- tem > 20
fh <- iMath(mask, "FillHoles") # list all ops
stopifnot(range(fh) == c(0, 1))
filled <- fh > 0

```

iMathOps

*Operations for iMath Operator***Description**

Examples, description and categorization of iMath operations.

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

```

data(iMathOps)
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")
for (j in c(1:nrow(iMathOps)))
{
  op <- as.character(iMathOps$Operation[j])
}
```

indicator_opt_both_ways

Helper Function to Optimize Indicator Matrix with Best Sum

Description

This function runs the optimization on both the input matrix ‘m’ and its negative counterpart ‘m * (-1)’, returning the result that maximizes the sum ‘sum(m * I)’.

Usage

```
indicator_opt_both_ways(m, verbose = FALSE)
```

Arguments

m	A numeric matrix for which the optimization will be performed.
verbose	Logical. If TRUE, reports the objective value and convergence progress at each iteration.

Value

m*I

Examples

```
set.seed(123)
m <- matrix(rnorm(500), nrow = 5)
indicator_opt_both_ways(m)
```

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. Can be used to estimate sparseness parameters per eigenvector. The user then only chooses nvecs and optional regularization parameters.

Usage

```
initializeEigenanatomy(initmat, mask = NULL, nreps = 1, smoothing = 0)
```

Arguments

initmat	input matrix where rows provide initial vector values. alternatively, this can be an antsImage which contains labeled regions.
mask	mask if available
nreps	nrepetitions to use
smoothing	if using an initial label image, optionally smooth each roi

Value

list is output

Author(s)

Avants BB

Examples

```
mat <- t(replicate(3, rnorm(100)))
initdf <- initializeEigenanatomy(mat) # produces a mask
dmat <- replicate(100, rnorm(20)) # data matrix
svdv <- t(svd(mat, nu = 0, nv = 10)$v)
ilist <- matrixToImages(svdv, initdf$mask)
eseg <- eigSeg(initdf$mask, ilist, TRUE)
eanat <- sparseDecom(dmat,
  inmask = initdf$mask,
  sparseness = 0, smooth = 0,
  initializationList = ilist, cthresh = 0,
  nvecs = length(ilist)
)
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$initlist)
)
# now a regression
eanatMatrix <- eanat$eigenanatomyimages
# 'averages' loosely speaking anyway
myEigenanatomyRegionAverages <- dmat %*% t(eanatMatrix)
dependentvariable <- rnorm(nrow(dmat))
res <- 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(
  inmatrix = list(dmat, dmat2),
  inmask = list(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
)

```

initializeSimlr	<i>Initialize simlr</i>
-----------------	-------------------------

Description

Four initialization approaches for simlr. Returns either a single matrix derived from dimensionality reduction on all matrices bound together (`jointReduction=TRUE`) or a list of reduced dimensionality matrices, one for each input. Primarily, a helper function for SiMLR.

Usage

```

initializeSimlr(
  voxmats,
  k,
  jointReduction = FALSE,
  zeroUpper = FALSE,
  uAlgorithm = "ba_svd",
  addNoise = 0
)

```

Arguments

<code>voxmats</code>	list that contains the named matrices.
<code>k</code>	rank of U matrix
<code>jointReduction</code>	boolean determining whether one or length of list bases are returned.
<code>zeroUpper</code>	boolean determining whether upper triangular part of initialization is zeroed out
<code>uAlgorithm</code>	either "svd", "random", "randomProjection", eigenvalue, "pca" (default), "ica" or "cca"
<code>addNoise</code>	scalar value that adds zero mean unit variance noise, multiplied by the value of <code>addNoise</code>

Value

A single matrix or list of matrices

Author(s)

BB Avants.

Examples

```
set.seed(1500)
nsub <- 3
npix <- c(10, 6, 13)
nk <- 2
outcome <- initializeSimlr(
  list(
    matrix(rnorm(nsub * npix[1]), ncol = npix[1]),
    matrix(rnorm(nsub * npix[2]), ncol = npix[2]),
    matrix(rnorm(nsub * npix[3]), ncol = npix[3])
  ),
  k = 2, uAlgorithm = "pca"
)
```

integrateVectorField *Integrate vector field*

Description

Utility function to integrate a vector field and create a deformation field.

Usage

```
integrateVectorField(
  referenceImage,
  vectorFieldFileName,
  deformationFieldFileName,
  lowerTime = 0,
  upperTime = 1,
  deltaTime = 0.01
)
```

Arguments

referenceImage	defines the image domain
vectorFieldFileName	the vector field exists on disk.
deformationFieldFileName	the deformation field output file name.
lowerTime	the starting time, usually zero for forward transformation and one for the inverse transformation.
upperTime	the ending time, usually one for forward transformation and zero for the inverse transformation.
deltaTime	the integration time step

Author(s)

Avants BB

Examples

```
## Not run:
set.seed(1234)
fi <- (ri(1))
mi <- (ri(2))
mytx2 <- antsRegistration(fixed = fi, mi, typeofTransform = "SyN")
integrateVectorField(fi, mytx2$velocityfield, "/tmp/def.nii.gz")

## End(Not run)
```

integrateVelocityField

integrateVelocityField

Description

Integrate velocity field

Usage

```
integrateVelocityField(
  velocityField,
  lowerIntegrationBound = 0,
  upperIntegrationBound = 1,
  numberOfIntegrationSteps = 10
)
```

Arguments

velocityField time-varying displacement field

lowerIntegrationBound
 Lower time bound for integration in [0, 1]

upperIntegrationBound
 Upper time bound for integration in [0, 1]

numberOfIntegrationSteps
 Number of integration steps used in the Runge-Kutta solution

Value

integrated field

Author(s)

NJ Tustison

Examples

```
fi <- antsImageRead(getANTsRData("r16"))
mi <- antsImageRead(getANTsRData("r27"))
reg <- antsRegistration(fi, mi, "TV[2]")
velocityField <- antsImageRead(reg$velocityfield)
field <- integrateVelocityField(velocityField, 0.0, 1.0, 10)
```

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)
```

interpret_simlr_vector

Interpret SiMLR Vector

Description

This function interprets a vector from SiMLR (similarity-driven multivariate linear reconstruction) specifically focusing on a given variable (e.g., a specific principal component or cluster). It extracts and normalizes the vector associated with the specified SiMLR variable, sorts it to identify the top elements, and optionally filters out non-significant values. This function is useful for understanding the contribution of different features in the context of the SiMLR analysis.

Usage

```
interpret_simlr_vector(
  simlrResult,
  simlrMats,
  simlrVariable,
  n2show = 5,
  shortnames = TRUE,
  return_dataframe = FALSE
)
```

Arguments

simlrResult	A list containing SiMLR analysis results, which should include a matrix ‘v’ representing vectors of interest (e.g., principal components).
simlrMats	A list of matrices associated with SiMLR analysis, where each matrix corresponds to a different modality or data type analyzed by SiMLR.
simlrVariable	A string specifying the variable within ‘simlrResult’ to interpret. The variable name should include both an identifier (e.g., "PC" for principal component) and a numeric index.
n2show	An integer specifying the number of top elements to show from the sorted, normalized vector. Defaults to 5. If ‘NULL’ or greater than the length of the vector, all elements are shown.
shortnames	boolean
return_dataframe	boolean

Value

A named vector of the top ‘n2show’ elements (or all if ‘n2show’ is ‘NULL’ or too large), sorted in decreasing order of their absolute values. Elements are named according to their identifiers in ‘simlrMats’ and filtered to exclude non-significant values (absolute value > 0).

Examples

```
# This example assumes you have SiMLR result `simlrResult`, matrices `simlrMats`, and you want to
# interpret the first principal component "PC1".
# simlrResult <- list(v = list(PC = matrix(runif(20), ncol = 2)))
# simlrMats <- list(PC = matrix(runif(100), ncol = 10))
# simlrVariable <- "PC1"
# interpretedVector <- interpret_simlr_vector(simlrResult, simlrMats, simlrVariable)
# print(interpretedVector)
```

```
interpret_simlr_vector2
```

Interpret SiMLR Vector

Description

This function interprets a vector from SiMLR (similarity-driven multivariate linear reconstruction) specifically focusing on a given variable (e.g., a specific principal component or cluster). It extracts and normalizes the vector associated with the specified SiMLR variable, sorts it to identify the top elements, and optionally filters out non-significant values. This function is useful for understanding the contribution of different features in the context of the SiMLR analysis. Assumes this input is generated by NNHEmbed.

Usage

```
interpret_simlr_vector2(
  simlrResult,
  simlrVariable,
  n2show = 5,
  shortnames = TRUE,
  return_dataframe = FALSE
)
```

Arguments

<code>simlrResult</code>	a specific v matrix out of SiMLR
<code>simlrVariable</code>	A string specifying the variable within 'simlrResult' to interpret. The variable name should include both an identifier (e.g., "PC" for principal component) and a numeric index.
<code>n2show</code>	An integer specifying the number of top elements to show from the sorted, normalized vector. Defaults to 5. If 'NULL' or greater than the length of the vector, all elements are shown.
<code>shortnames</code>	boolean
<code>return_dataframe</code>	boolean

Value

A named vector of the top ‘n2show’ elements (or all if ‘n2show’ is ‘NULL’ or too large), sorted in decreasing order of their absolute values. Elements are named according to their identifiers in ‘simlrMats’ and filtered to exclude non-significant values (absolute value > 0).

Examples

```
# This example assumes you have SiMLR result `simlrResult`, matrices `simlrMats`, and you want to
# interpret the first principal component "PC1".
# simlrResult <- list(v = list(PC = matrix(runif(20), ncol = 2)))
# simlrMats <- list(PC = matrix(runif(100), ncol = 10))
# simlrVariable <- "PC1"
# interpretedVector <- interpret_simlr_vector2(simlrResult$v[[1]])
# print(interpretedVector)
```

inv_sqrt_sym

Inverse Square Root of a Symmetric Matrix

Description

Computes the inverse square root of a symmetric positive semi-definite matrix using eigen-decomposition.

Usage

```
inv_sqrt_sym(M, epsilon = 1e-10)
```

Arguments

M	Numeric symmetric matrix (k x k).
epsilon	Small numeric value for numerical stability (default 1e-10).

Details

For a symmetric matrix M, computes $M^{-1/2}$ via eigen-decomposition, clipping eigenvalues below 1e-12 to ensure numerical stability. Used in polar and soft retractions for NSA-Flow.

Value

Numeric matrix, the inverse square root of M.

Examples

```
M <- matrix(c(4, 1, 1, 2), 2, 2)
inv_sqrt_sym(M)
```

inv_sqrt_sym_newton	<i>Inverse square root of a symmetric matrix via Newton-Schulz</i>
---------------------	--

Description

Inverse square root of a symmetric matrix via Newton-Schulz

Usage

```
inv_sqrt_sym_newton(  
    A,  
    epsilon = 1e-10,  
    max_iter = 10L,  
    tol = 1e-06,  
    verbose = FALSE  
)
```

Arguments

A	symmetric matrix
epsilon	small value for stability
max_iter	maximum number of iterations
tol	tolerance for convergence
verbose	boolean for verbosity

invariant_orthogonality_defect	<i>Calculate the invariant orthogonality defect</i>
--------------------------------	---

Description

Calculate the invariant orthogonality defect

Usage

```
invariant_orthogonality_defect(A)
```

Arguments

A	input matrix
---	--------------

```
invertAntsrTransform  invertAntsrTransform
```

Description

invert a linear antsTransform

Usage

```
invertAntsrTransform(transform)
```

Arguments

transform transform to invert

Value

antsTransform

Examples

```
img <- antsImageRead(getANTSRData("r16"))
tx <- new("antsrTransform", precision = "float", type = "AffineTransform", dimension = 2)
setAntsrTransformParameters(tx, c(0, -1, 1, 0, dim(img)[1], 0))
txinv <- invertAntsrTransform(tx)
```

```
invertDisplacementField
      invertDisplacementField
```

Description

Invert displacement field.

Usage

```
invertDisplacementField(
  displacementField,
  inverseFieldInitialEstimate,
  maximumNumberOfIterations = 20,
  meanErrorToleranceThreshold = 0.001,
  maxErrorToleranceThreshold = 0.1,
  enforceBoundaryCondition = TRUE
)
```

Arguments

displacementField
displacement field.
inverseFieldInitialEstimate
initial guess.
maximumNumberOfIterations
number of iterations.
meanErrorToleranceThreshold
mean error tolerance threshold.
maxErrorToleranceThreshold
max error tolerance threshold.
enforceBoundaryCondition
enforce stationary boundary condition.

Value

inverse displacement field

Author(s)

NJ Tustison

is.antsImage	<i>is.antsImage</i>
--------------	---------------------

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))
```

jhu_fa_labels	<i>jhu_fa_labels</i>
---------------	----------------------

Description

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

Usage

jhu_fa_labels

Format

A data frame with columns: Label, Description.

jhu_md_labels	<i>jhu_md_labels</i>
---------------	----------------------

Description

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

Usage

jhu_md_labels

Format

A data frame with columns: Label, Description.

jlfProp	<i>Propagate Joint Label Fusion (JLF) solutions through a time series</i>
---------	---

Description

The JLF initialization is reused throughout a time series with an additional lag argument.

Usage

```
jlfProp(
  targetI,
  targetIMask,
  atlasList,
  rad = 2,
  labelList = NULL,
  rSearch = 3,
  lagValue = 3,
  verbose = FALSE,
  ...
)
```

Arguments

targetI	antsImage list to be approximated
targetIMask	mask with value 1
atlasList	list containing antsImages with intensity images
rad	neighborhood radius, default to 2
labelList	optional list containing antsImages with segmentation labels
rSearch	radius of search, default is 3
lagValue	number of prior images to use to fwd propagate JLF solution
verbose	boolean
...	arguments to pass to jointLabelFusion

Value

segmentation of time series

Author(s)

Brian B. Avants

Examples

```
## Not run:
set.seed(123)
ref <- ri(1) %>%
  resampleImage(4) %>%
  iMath("Normalize")
mi1 <- ri(2)
mi2 <- ri(3)
mi3 <- ri(4)
mi4 <- ri(5)
mi5 <- ri(6)
refmask <- getMask(ref)
refmask <- iMath(refmask, "MD", 10) # just to speed things up
```

```

ilist <- list(mi1, mi2, mi3, mi4, mi5)
seglis <- list()
for (i in 1:length(ilist))
{
  ilist[[i]] <- iMath(ilist[[i]], "Normalize")
  mytx <- antsRegistration(
    fixed = ref, moving = ilist[[i]],
    typeofTransform = c("Affine"), verbose = TRUE
  )
  mywarpedimage <- antsApplyTransforms(
    fixed = ref,
    moving = ilist[[i]],
    transformlist = mytx$fdtransforms
  )
  ilist[[i]] <- mywarpedimage
  seg <- thresholdImage(ilist[[i]], "Otsu", 3)
  seglist[[i]] <- seg + 1
}
tarlist <- list(
  iMath(ref, "GD", 3),
  iMath(ref, "GD", 2),
  iMath(ref, "GD", 1),
  iMath(ref, "GD", 0),
  iMath(ref, "GE", 1),
  iMath(ref, "GE", 2),
  iMath(ref, "GE", 3)
)
pp <- jlfProp(tarlist, refmask, ilist,
  rSearch = 2,
  labellist = seglist, rad = rep(2, length(dim(ref)))
)

## End(Not run)

```

joinEigenanatomy

Simple joinEigenanatomy function.

Description

joinEigenanatomy joins the input matrix using a community membership approach.

Usage

```

joinEigenanatomy(
  datamatrix,
  mask = NULL,
  listEanatImages,
  graphdensity = 0.65,
  joinMethod = "walktrap",
  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

```
## Not run:
# if you dont have images
mat <- replicate(100, rnorm(20))
mydecom <- sparseDecom(mat)
kk <- joinEigenanatomy(mat, mask = NULL, mydecom$eigenanatomyimages, 0.1)
# or select optimal parameter from a list
kk <- joinEigenanatomy(mat, mask = NULL, mydecom$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)
eanatimages <- matrixToImages(mydecom$eigenanatomyimages, mask)
kki <- joinEigenanatomy(mat, mask = mask, eanatimages, 0.1)
if (usePkg("igraph")) {
  mydecomf <- sparseDecom(mat,
    inmask = mask, initializationList = kki$fusedlist,
    sparseness = 0, nvecs = length(kki$fusedlist)
  )
}

## End(Not run)
```

Description

A multiple atlas voting scheme to customize labels for a new subject. This function will also perform intensity fusion. It almost directly calls the C++ in the ANTs executable so is much faster than other variants in ANTsR. One may want to normalize image intensities for each input image before passing to this function. If no labels are passed, we do intensity fusion. Note on computation time: the underlying C++ is multithreaded. You can control the number of threads by setting the environment variable `ITK_GLOBAL_DEFAULT_NUMBER_OF_THREADS` e.g. to use all or some of your CPUs. This will improve performance substantially. For instance, on a macbook pro from 2015, 8 cores improves speed by about 4x.

Usage

```
jointLabelFusion(
  targetI,
  targetIMask,
  atlasList,
  beta = 4,
  rad = 2,
  labelList = NULL,
  rho = 0.01,
  usecor = FALSE,
  rSearch = 3,
  nonnegative = FALSE,
  maxLabelPlusOne = FALSE,
  noZeroes = FALSE,
  verbose = FALSE
)
```

Arguments

<code>targetI</code>	antsImage to be approximated
<code>targetIMask</code>	mask with value 1
<code>atlasList</code>	list containing antsImages with intensity images
<code>beta</code>	weight sharpness, default to 2
<code>rad</code>	neighborhood radius, default to 2
<code>labelList</code>	optional list containing antsImages with segmentation labels
<code>rho</code>	ridge penalty increases robustness to outliers but also makes image converge to average
<code>usecor</code>	employ correlation as local similarity
<code>rSearch</code>	radius of search, default is 3
<code>nonnegative</code>	constrain weights to be non-negative
<code>maxLabelPlusOne</code>	boolean this will add max label plus one to the non-zero parts of each label where the target mask is greater than one. NOTE: this will have a side effect of adding to the original label images that are passed to the program. It also

	guarantees that every position in the labels have some label, rather than none. Ie it guarantees to explicitly parcellate the input data.
noZeroes	boolean will zero out target mask regions that have any zero label. this prevents JLF from computing a solution in regions not covered by the initial library.
verbose	boolean

Value

approximated image, segmentation and probabilities

Author(s)

Brian B. Avants, Hongzhi Wang, Paul Yushkevich, Nicholas J. Tustison

Examples

```

set.seed(123)
ref <- ri(1)
ref <- resampleImage(ref, c(50, 50), 1, 0)
ref <- iMath(ref, "Normalize")
mi <- ri(2)
mi2 <- ri(3)
mi3 <- ri(4)
mi4 <- ri(5)
mi5 <- ri(6)
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"), verbose = TRUE
  )
  mywarpedimage <- antsApplyTransforms(
    fixed = ref,
    moving = ilist[[i]],
    transformlist = mytx$fwdtransforms
  )
  ilist[[i]] <- mywarpedimage
  seg <- thresholdImage(ilist[[i]], "Otsu", 3)
  seglist[[i]] <- seg
}
r <- 2
pp <- jointLabelFusion(ref, refmask, ilist,
  rSearch = 2,
  labellist = seglist, rad = rep(r, length(dim(ref)))
)
pp2 <- jointLabelFusion(ref, refmask, ilist,
  rSearch = 2,

```

```

    labellist = seglist, rad = rep(r, length(dim(ref)))
  )
  testthat::expect_equal(pp2$segmentation, pp$segmentation)
  pp <- jointLabelFusion(ref, refmask, ilist,
    rSearch = 2,
    rad = rep(r, length(dim(ref)))
  )

  ## Not run:
  ref <- antsImageRead(getANTsRData("ch2"))
  n <- 50
  ref <- resampleImage(ref, c(n, n, n), 1, 0)
  ref <- iMath(ref, "Normalize")
  refmask <- getMask(ref)
  ilist <- list()
  seglist <- list()
  for (k in 1:5) {
    mi <- antsImageClone(ref) + rnorm(n^3, 0, 0.1)
    mykseg <- kmeansSegmentation(mi, 3, refmask)$segmentation
    ilist[[k]] <- mi
    seglist[[k]] <- mykseg
  }
  pp <- jointLabelFusion(ref, refmask, ilist,
    rSearch = 2,
    labellist = seglist, rad = rep(2, length(dim(ref))), verbose = TRUE
  )
  plot(ref, pp$segmentation)
  plot(pp$intensity)

  ## End(Not run)

```

jointSmoothMatrixReconstruction
joint smooth matrix prediction

Description

Joints reconstruct a n by p , q , etc matrices or predictors. The reconstruction can be regularized. $\#$
 $\text{norm}(x - u_x v_x^t) + \text{norm}(y - u_y v_y^t) + \dots$ etc

Usage

```

jointSmoothMatrixReconstruction(
  x,
  nvecs,
  parameters,
  iterations = 10,
  subIterations = 5,

```

```

    gamma = 1e-06,
    sparsenessQuantile = 0.5,
    positivity = FALSE,
    smoothingMatrix = NA,
    rowWeights = NA,
    repeatedMeasures = NA,
    doOrth = FALSE,
    verbose = FALSE
  )

```

Arguments

<code>x</code>	input list of matrices to be jointly predicted.
<code>nvecs</code>	number of basis vectors to compute
<code>parameters</code>	should be a ncomparisons by 3 matrix where the first two columns define the pair to be matched and the last column defines the weight in the objective function.
<code>iterations</code>	number of gradient descent iterations
<code>subIterations</code>	number of gradient descent iterations in sub-algorithm
<code>gamma</code>	step size for gradient descent
<code>sparsenessQuantile</code>	quantile to control sparseness - higher is sparser
<code>positivity</code>	restrict to positive or negative solution (beta) weights. choices are positive, negative or either as expressed as a string.
<code>smoothingMatrix</code>	a list containing smoothing matrices of the same length as x.
<code>rowWeights</code>	vectors of weights with size n (assumes diagonal covariance)
<code>repeatedMeasures</code>	list of repeated measurement identifiers. this will allow estimates of per identifier intercept.
<code>doOrth</code>	boolean enforce gram-schmidt orthonormality
<code>verbose</code>	boolean option

Value

matrix list each of size p by k is output

Author(s)

Avants BB

Examples

```

## Not run:
mat <- replicate(100, rnorm(20))
mat2 <- replicate(100, rnorm(20))
mat <- scale(mat)

```

```

mat2 <- scale(mat2)
wt <- 0.666
mat3 <- mat * wt + mat2 * (1 - wt)
params <- matrix(nrow = 2, ncol = 3)
params[1, ] <- c(1, 2, 1)
params[2, ] <- c(2, 1, 1)
x <- list((mat), (mat3))
jj <- jointSmoothMatrixReconstruction(x, 2, params,
  gamma = 1e-4, sparsenessQuantile = 0.5, iterations = 10,
  smoothingMatrix = list(NA, NA), verbose = TRUE
)

# latent feature
mask <- getMask(antsImageRead(getANTsRData("r16")))
spatmat <- t(imageDomainToSpatialMatrix(mask, mask))
smoomat <- knnSmoothingMatrix(spatmat, k = 27, sigma = 120.0)
lfeats <- t(replicate(100, rnorm(3)))
# map these - via matrix - to observed features
n <- sum(mask)
ofeats1 <- (lfeats + rnorm(length(lfeats), 0.0, 1.0)) %%% rbind(rnorm(n), rnorm(n), rnorm(n))
ofeats2 <- (lfeats + rnorm(length(lfeats), 0.0, 1.0)) %%% rbind(rnorm(n), rnorm(n), rnorm(n))
# only half of the matrix contains relevant data
ofeats1[, 1:round(n / 2)] <- matrix(rnorm(round(n / 2) * 100), nrow = 100)
ofeats2[, 1:round(n / 2)] <- matrix(rnorm(round(n / 2) * 100), nrow = 100)
x <- list((ofeats1), (ofeats2))
jj <- jointSmoothMatrixReconstruction(x, 2, params,
  gamma = 0.0001, sparsenessQuantile = 0.75, iterations = 19,
  subIterations = 11,
  smoothingMatrix = list(smoomat, smoomat), verbose = TRUE
)
p1 <- ofeats2 %%% jj$v[[2]]
p2 <- ofeats1 %%% jj$v[[1]]
cor(p1, lfeats)
cor(p2, lfeats)
print(cor(rowMeans(ofeats1), lfeats))
print(cor(rowMeans(ofeats2), lfeats))

# a 2nd example with 3 modalities
imageIDs <- c("r16", "r27", "r30", "r62", "r64", "r85")
images <- list()
feature1Images <- list()
feature2Images <- list()
feature3Images <- list()
ref <- antsImageRead(getANTsRData("r16"))
for (i in 1:length(imageIDs))
{
  cat("Processing image", imageIDs[i], "\n")
  tar <- antsRegistration(ref, antsImageRead(getANTsRData(imageIDs[i])),
    typeofTransform = "Affine"
  )$warpedmov
  images[[i]] <- tar
  feature1Images[[i]] <- iMath(images[[i]], "Grad", 1.0)
  feature2Images[[i]] <- iMath(images[[i]], "Laplacian", 1.0)
}

```

```

    feature3Images[[i]] <- reflectImage(tar, axis = 0, tx = "Affine")$warpedmovout
  }
  i <- 1
  mask <- getMask(antsImageRead(getANTsRData(imageIDs[i])))
  mask2 <- iMath(mask, "ME", 2)
  spatmat <- t(imageDomainToSpatialMatrix(mask, mask))
  smoomat <- knnSmoothingMatrix(spatmat, k = 125, sigma = 100.0)
  spatmat2 <- t(imageDomainToSpatialMatrix(mask2, mask2))
  smoomat2 <- knnSmoothingMatrix(spatmat2, k = 125, sigma = 100.0)
  params <- matrix(nrow = 6, ncol = 3)
  params[1, ] <- c(1, 2, 1)
  params[2, ] <- c(2, 1, 1)
  params[3, ] <- c(1, 3, 1)
  params[4, ] <- c(3, 1, 1)
  params[5, ] <- c(2, 3, 1)
  params[6, ] <- c(3, 2, 1)
  mat <- imageListToMatrix(feature1Images, mask)
  mat2 <- imageListToMatrix(feature2Images, mask2)
  mat3 <- imageListToMatrix(feature3Images, mask)
  scl <- F
  x <- list(scale(mat, scale = scl), scale(mat2, scale = scl), scale(mat3, scale = scl))
  slist <- list(smoomat2, smoomat, smoomat, smoomat, smoomat, smoomat2)

  jj <- jointSmoothMatrixReconstruction(x, 4, params,
    positivity = T,
    gamma = 1e-6, sparsenessQuantile = 0.9, iterations = 10,
    smoothingMatrix = slist, verbose = TRUE
  )
  mm <- makeImage(mask, abs(jj$v[[2]][, 1])) %>% iMath("Normalize")
  plot(ref, mm, doCropping = FALSE, window.overlay = c(0.1, 1))

  p1 <- mat2 %*% jj$v[[1]]
  p2 <- mat %*% jj$v[[2]]
  diag(cor(p1, p2))

  p1 <- mat3 %*% jj$v[[5]]
  p2 <- mat2 %*% jj$v[[6]]
  diag(cor(p1, p2))

  ## End(Not run)

```

Description

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

Usage

```

kellyKapowski(
  s,
  g,
  w,
  its = 45,
  r = 0.025,
  m = 1.5,
  x = FALSE,
  e = FALSE,
  q = NULL,
  timeSigma = 1,
  verbose = FALSE,
  ...
)

```

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
x	matrix-based smoothing
e	restrict deformation boolean
q	time spacing, a vector equal to the number of time dimensions
timeSigma	a scalar sigma value for distances between time points
verbose	boolean
...	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	<i>k means image segmentation.</i>
--------------------	------------------------------------

Description

k means image segmentation that is a wrapper around atropos

Usage

```
kmeansSegmentation(img, k, kmask = NULL, mrf = 0.1, verbose = FALSE, ...)
```

Arguments

img	input image
k	integer number of classes
kmask	segment inside this mask
mrf	smoothness, higher is smoother
verbose	boolean
...	additional arguments to pass to atropos

Value

segmentation and probability images

Note

This function will likely give different results on multiple runs.

Author(s)

Brian B. Avants

Examples

```

fi <- antsImageRead(getANTsRData("r16"), 2)
orig <- antsImageClone(fi)
fi <- n3BiasFieldCorrection(fi, 4)
seg <- kmeansSegmentation(fi, 3)
seg2 <- kmeansSegmentation(fi, 3)
arr1 <- as.array(seg$segmentation)
arr2 <- as.array(seg2$segmentation)
testthat::expect_equal(arr1, arr2)

```

```

set.seed(2)
# set below for slower but numerically repeatable results
# these should be set in .Renviron not by sys calls
# Sys.setenv(ITK_GLOBAL_DEFAULT_NUMBER_OF_THREADS = 1 )
orig <- antsImageRead(getANTsRData("r16"), 2)
seg <- kmeansSegmentation(orig, 3, use_random_seed = TRUE)
seg2 <- kmeansSegmentation(orig, 3, use_random_seed = TRUE)
arr1 <- as.array(seg$segmentation)
arr2 <- as.array(seg2$segmentation)
tab <- table(arr1, arr2)
tab
identical(arr1, arr2)
seg <- kmeansSegmentation(orig, 3, use_random_seed = FALSE)
seg2 <- kmeansSegmentation(orig, 3, use_random_seed = FALSE)
arr1 <- as.array(seg$segmentation)
arr2 <- as.array(seg2$segmentation)
tab <- table(arr1, arr2)
tab

```

knnSmoothImage

k-nearest neighbors constrained image smoothing

Description

Compute a smoothing matrix based on an input matrix of point coordinates as well as neighborhood intensity patterns. this performs a form of edge preserving smoothing.

Usage

```

knnSmoothImage(
  img,
  mask,
  radius,
  intensityWeight = 0.1,
  spatialSigma = 20,
  iterations = 1,
  returnMatrix = FALSE
)

```

Arguments

img	input image to smooth
mask	input image mask
radius	number of neighbors, higher causes more smoothing
intensityWeight	weight for intensity component, value 1 will weight local voxel intensity roughly equally to spatial component

spatialSigma	for gaussian defining spatial distances
iterations	number of iterations over which to apply smoothing kernel
returnMatrix	boolean, will return smoothing matrix instead of image.

Value

antsImage is output

Author(s)

Avants BB

Examples

```
## Not run:
img <- antsImageRead(getANTsRData("r16"))
mask <- getMask(img)
simg <- knnSmoothImage(
  img = img, mask = mask, radius = 2, intensityWeight = 1,
  spatialSigma = 1.5, iterations = 1
)

## End(Not run)
```

knnSmoothingMatrix	<i>k-nearest neighbors constrained smoothing</i>
--------------------	--

Description

Compute a smoothing matrix based on an input matrix of point coordinates

Usage

```
knnSmoothingMatrix(x, k, sigma, segmentation, ...)
```

Arguments

x	input matrix of point coordinates of dimensions n-spatial spatial dimensions by p points
k	number of neighbors, higher causes more smoothing
sigma	sigma for the gaussian function
segmentation	optional boolean to restrict specific rows to have minimal respons
...	arguments passed to sparseDistanceMatrix

Value

sparse matrix is output

Author(s)

Avants BB

Examples

```
## Not run:
mask <- getMask(antsImageRead(getANTsRData("r16")))
spatmat <- t(imageDomainToSpatialMatrix(mask, mask))
smoothingMatrix <- knnSmoothingMatrix(spatmat, k = 25, sigma = 3.0)
rvec <- rnorm(nrow(smoothingMatrix))
srvec <- smoothingMatrix %*% rvec
rvi <- makeImage(mask, rvec)
srv <- makeImage(mask, as.numeric(srvec))

## End(Not run)
```

`l1_normalize_features` *L1 Normalize Columns of a Matrix*

Description

This function scales the columns of a numeric matrix such that the sum of the absolute values of each column (the L1 norm) is equal to 1. This is a common preprocessing step for feature matrices.

Usage

```
l1_normalize_features(features)
```

Arguments

<code>features</code>	A numeric matrix or a data frame that can be coerced into one. The normalization is applied column-wise.
-----------------------	--

Details

The function is designed to be highly efficient by using vectorized operations. It also robustly handles columns that sum to zero to prevent division-by-zero errors.

Value

A matrix with the same dimensions as the input, where each column has been L1-normalized.

Examples

```
# Create a sample feature matrix
set.seed(123)
my_features <- matrix(rnorm(12, mean = 5), nrow = 4, ncol = 3)
print(my_features)
#>      [,1]      [,2]      [,3]
#> [1,] 4.4395244 5.487429 5.704929
#> [2,] 5.2325834 6.738325 5.558708
#> [3,] 3.4291247 5.575781 4.129288
#> [4,] 5.5060559 4.694612 4.261173

# Check the column sums of absolute values before normalization
print(colSums(abs(my_features)))
#> [1] 18.60729 22.49615 19.65410

# Apply L1 normalization
normalized_features <- l1_normalize_features(my_features)
print(normalized_features)
#>      [,1]      [,2]      [,3]
#> [1,] 0.23859089 0.2439247 0.2800889
#> [2,] 0.28121151 0.2995321 0.2891398
#> [3,] 0.18429107 0.2478546 0.2151676
#> [4,] 0.29590653 0.2086886 0.2221037

# Confirm that the new column sums of absolute values are all 1
print(colSums(abs(normalized_features)))
#> [1] 1 1 1
```

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,
  fullyConnected = FALSE
)
```

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
fullyConnected boolean sets neighborhood connectivity pattern

Value

labeled cluster image is output

Author(s)

Avants BB

Examples

```
img <- antsImageRead(getANTsRData("r16"))  
timgFully <- labelClusters(img, 10, 128, 150, TRUE)  
timgFace <- labelClusters(img, 10, 128, 150, FALSE)  
## Not run:  
plot(img, timgFace)  
plot(img, timgFully)  
  
## End(Not run)
```

labelGeometryMeasures *labelGeometryMeasures*

Description

Wrapper for the ANTs funtion labelGeometryMeasures

Usage

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

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	<i>labelImageCentroids</i>
---------------------	----------------------------

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)
```

labelImageRegistration

Perform label image registration.

Description

Perform pairwise registration using fixed and moving sets of label images (and, optionally, sets of corresponding intensity images).

Usage

```
labelImageRegistration(
    fixedLabelImages,
    movingLabelImages,
    fixedIntensityImages = NULL,
    movingIntensityImages = NULL,
    fixedMask = NULL,
    movingMask = NULL,
    initialTransforms = "affine",
    typeOfDeformableTransform = "antsRegistrationSyNQuick[so]",
    labelImageWeighting = 1,
    outputPrefix = "",
    randomSeed = NULL,
    verbose = FALSE
)
```

Arguments

fixedLabelImages	A single (or set of) fixed label image(s).
movingLabelImages	A single (or set of) moving label image(s).
fixedIntensityImages	Optional—a single (or set of) fixed intensity image(s).
movingIntensityImages	Optional—a single (or set of) moving intensity image(s).
fixedMask	Defines region for similarity metric calculation in the space of the fixed image.
movingMask	Defines region for similarity metric calculation in the space of the moving image.
initialTransforms	If specified, there are two options: 1) Use label images with the centers of mass to calculate linear transform of type 'identity', 'rigid', 'similarity', 'affine'. 2) Specify a list of transform files, e.g., the output of antsRegistration.

typeOfDeformableTransform

Only works with deformable-only transforms, specifically the family of `antsRegistrationSyN*[so]` or `antsRegistrationSyN*[bo]` transforms. See `'typeOfTransform'` in `antsRegistration`. Additionally, one can use a list to pass a more tailored deformable-only transform optimization using SyN or BSplineSyN transforms. The order of parameters in the list would be 1) transform specification, i.e. "SyN" or "BSplineSyN", 2) gradient (real), 3) intensity metric (string), 4) intensity metric parameter (real), 5) convergence iterations per level (vector) 6) smoothing factors per level (tuple), 7) shrink factors per level (vector). An example would `typeOfDeformable-Transform = list("SyN", 0.2, "CC", 4, c(100,50,10), c(2,1,0), c(4,2,1))`.

labelImageWeighting

Float or vector of floats giving the relative weighting for the label images.

outputPrefix

String definining the output prefix for the filenames of the output transform files.

randomSeed

Definition for deformable registration.

verbose

Print progress to the screen.

Value

outputs a list containing:

fwdtransforms: Transforms to move from moving to fixed image.

invtransforms: Transforms to move from fixed to moving image.

Output of 1 indicates failure.

Author(s)

Tustison NJ

Examples

[illegible]

labelOverlapMeasures *labelOverlapMeasures*

Description

Wrapper for the ANTs function `LabelOverlapMeasures`. More documentation available here:

Usage

```
labelOverlapMeasures(sourceLabelImage, targetLabelImage)
```

Arguments

`sourceLabelImage`
label image for source image.

`targetLabelImage`
label image for target/reference image.

Details

<https://www.insight-journal.org/browse/publication/707>

Value

data frame with overlap measures

Author(s)

Avants BB, Tustison NJ

Examples

```
sourceImage <- antsImageRead(getANTsRData("r16"), 2)
sourceSegmentation <- kmeansSegmentation(sourceImage, 3)$segmentation
referenceImage <- antsImageRead(getANTsRData("r16"), 2)
referenceSegmentation <- kmeansSegmentation(referenceImage, 3)$segmentation
overlap <- labelOverlapMeasures(sourceSegmentation, referenceSegmentation)
```

labels2matrix	<i>Convert label image to a matrix</i>
---------------	--

Description

Convert a labeled image to an $n \times m$ binary matrix where n = number of voxels and m = number of labels. Only includes values inside the provided mask while including background (`img == 0`) for consistency with `timeseries2matrix` and other image to matrix operations.

Usage

```
labels2matrix(img, mask, targetLabels = NULL, missingVal = NA)
```

Arguments

<code>img</code>	input label image
<code>mask</code>	defines domain of interest
<code>targetLabels</code>	defines target regions to be returned. if the target label does not exist in the input label image, then the matrix will contain a constant value of <code>missingVal</code> (default <code>NA</code>) in that row.
<code>missingVal</code>	for missing label values.

Value

matrix is output

Author(s)

Avants BB

Examples

```
fi <- antsImageRead(getANTsRData("r16"), 2) %>% resampleImage(c(60, 60), 1, 0)
mask <- getMask(fi)
labs <- kmeansSegmentation(fi, 3)$segmentation
labmat <- labels2matrix(labs, mask)
```

labelStats	<i>labelStats</i>
------------	-------------------

Description

Get label statistics from image.

Usage

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

Arguments

image	Image to calculate statistics from.
labelImage	Label image.

Value

Data frame with one row per label (including zero) and 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 <- kmeansSegmentation(img, 3)
labelStats(img, segs1$segmentation)
testthat::expect_error(labelStats(img, as.array(segs1$segmentation)))
testthat::expect_error(labelStats(img, as.array(segs1$segmentation)))
sub_seg <- segs1$segmentation
sub_seg <- as.antsImage(sub_seg[1:62, 1:62], reference = sub_seg)
testthat::expect_error(labelStats(img, sub_seg))
```

lappend	<i>Simple list append tool</i>
---------	--------------------------------

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

```
mat <- replicate(100, rnorm(20))
mat2 <- replicate(100, rnorm(20))
mylist <- list(mat)
mylist <- lappend(mylist, mat2)
mylist <- lappend(mat2, mylist)
mylist <- lappend(mylist, mylist)
lappend(1, 1)
```

list_simlr_optimizers *List Available SIMLR Optimizers*

Description

Dynamically returns the names of all optimizer types with registered ‘step.*’ methods.

Usage

```
list_simlr_optimizers(torch = FALSE)
```

Arguments

torch	Logical; if TRUE, returns a predefined list of optimizers compatible with torch.
-------	--

Value

A character vector of optimizer names (e.g., "adam", "nadam", etc.)

load.ANTsR	<i>Load R sessions with ANTsR objects</i>
------------	---

Description

Loads a previously saved ANTsR session or object. Simply point the function to the folder where you previously saved the data with save.ANTsR.

Usage

```
load.ANTsR(filename=file.path('.', '.ANTsRsession'), env=as.environment(1))
```

Arguments

- filename path of the saved ANTsR session. If not specified, the default behavior is to search for the folder \".ANTsRsession\" in the current working directory.
- env the environment level. Don't change unless you know what are you doing.

Author(s)

Dorian Pustina

lobes	<i>lobes</i>
-------	--------------

Description

A data frame label numbers and names for brain lobes.

Usage

```
lobes
```

Format

A data frame with columns: Label, Description, Side, SideNumber.

lobes_brainstem	<i>lobes_brainstem</i>
-----------------	------------------------

Description

A data frame label numbers and names for lobes and brainstem regions.

Usage

lobes_brainstem

Format

A data frame with columns: Label, Description, Side.

localGyrificationIndex	<i>Local gyrification index.</i>
------------------------	----------------------------------

Description

Estimate the gyrification index at a specific scale. This will produce images of local sulcification and gyrification as as well as the difference of these two measurements. The most gyral values are near 1 and most sulcal negative 1. The function is not limited to brain data and should work on any 3D shape. See the paper entitled the shape operator for differential image analysis.

Usage

localGyrificationIndex(segmentation, sigma = 3, k = 25, ksigma = 3)

Arguments

- segmentation - potentially from antsCorticalThickness
- sigma - scale parameter
- k - size of neighborhood
- ksigma - sigma for k-neighborhood

Value

lGI image

Author(s)

Avants BB

Examples

```
## Not run:
seg <- antsImageRead("segmentation.nii.gz")
wm <- thresholdImage(seg, 3, 3) %>% morphology("dilate", 1)
lgi <- localGyrificationIndex(wm)

## End(Not run)
```

localJointLabelFusion *local joint label and intensity fusion*

Description

A local version of joint label fusion that focuses on one or more specific labels. This is primarily different from standard JLF because it performs registration on a per label basis and focuses JLF on the label(s) alone. It requires an initial segmentation of the target region which can be provided either by a manual or automated initialization. Registration by SyN is a good choice for the latter approach.

Usage

```
localJointLabelFusion(
  targetI,
  whichLabels,
  targetMask,
  initialLabel,
  atlasList,
  labelList,
  submaskDilation = 10,
  typeofTransform = "SyN",
  affMetric = "meansquares",
  synMetric = "mattes",
  synSampling = 32,
  regIterations = c(40, 20, 0),
  affIterations,
  localMaskTransform,
  maxLabelPlusOne = FALSE,
  noZeroes = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

targetI	antsImage to be labeled
whichLabels	label number(s) from the library on which to focus

targetMask	a mask for the target image (optional), passed to joint fusion
initialLabel	the initial approximate label(s) for the target region.
atlasList	list containing antsImages with intensity images
labellList	list containing antsImages with segmentation labels
submaskDilation	amount to dilate initial mask to define region on which we perform focused registration
typeofTransform	passed to antsRegistration.
affMetric	the metric for the affine part (GC, mattes, meansquares)
synMetric	the metric for the syn part (CC, mattes, meansquares, demons)
synSampling	the nbins or radius parameter for the syn metric
regIterations	vector of iterations for syn. we will set the smoothing and multi-resolution parameters based on the length of this vector. passed to antsRegistration.
affIterations	vector of iterations for low-dimensional transforms.
localMaskTransform	type of transform for local mask initialization; would usually set to Rigid, Similarity or Affine
maxLabelPlusOne	boolean this will add max label plus one to the non-zero parts of each label where the target mask is greater than one. NOTE: this will have a side effect of adding to the original label images that are passed to the program. It also guarantees that every position in the labels have some label, rather than none. Ie it guarantees to explicitly parcellate the input data.
noZeroes	boolean will zero out target mask regions that have any zero label. this prevents JLF from computing a solution in regions not covered by the initial library.
verbose	boolean
...	extra parameters passed to JLF

Value

label probabilities and segmentations

Author(s)

Brian B. Avants

lowrankRowMatrix	<i>Produces a low rank version of the input matrix</i>
------------------	--

Description

Produces a low rank version of the input matrix

Usage

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

Arguments

A	input matrix
k	rank to use
faster	boolean

Value

matrix is output

Author(s)

Avants BB

Examples

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

make3ViewPNG	<i>Rotate an existing 3d window into different views.</i>
--------------	---

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(
  surfing = list(bm),
  alphasurf = 0.1, smoothsval = 1.5
)
make3ViewPNG(diag(4), diag(4), diag(4), tempfile(fileext = ".png"))

## End(Not run)
```

makeGraph	<i>Simple function to create and measure a graph from a square input matrix.</i>
-----------	--

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,
  inverseValuesAsWeights = FALSE
)
```

Arguments

`mat` input matrix
`graphdensity` fraction of edges to keep
`communityMethod` see igraph's community detection
`getEfficiency` boolean, this is slow to compute
`inverseValuesAsWeights`
 if TRUE, high correlations produce small edge weights. This detail is important when using igraph algorithms as different methods use weights in different ways. See igraph for its `is_weighted` documentation which notes that weights are used as distances in shortest path calculations and as strength (similar to degree) for community methods.

Value

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

Author(s)

Avants BB

Examples

```
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)
```

makeImage

Simple makeImage function.

Description

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

Usage

```
makeImage(
  imagesize,
  voxval = 0,
  spacing = c(NA),
  origin = c(NA),
  direction = c(NA),
  components = FALSE,
  pixeltype = "float"
)
```

Arguments

imagesize	input image size or mask
voxval	input image value or vector, size of mask
spacing	image spatial resolution
origin	image spatial origin
direction	direction matrix to convert from index to physical space
components	whether there are components per pixel or not
pixeltype	data type of image values

Value

antsImage is output

Author(s)

Avants BB

Examples

```
outimg <- makeImage(c(2, 10), 1)
outimg <- makeImage(outimg, c(2, 10))
testthat::expect_error(makeImage(outimg, c(2, 10), components = 2))
```

makePointsImage	<i>Create label image from physical space points</i>
-----------------	--

Description

Creates spherical points in the coordinate space of the target image based on the n-dimensional matrix of points that the user supplies. The image defines the dimensionality of the data so if the input image is 3D then the input points should be 2D or 3D.

Usage

```
makePointsImage(pts, mask, radius = 5)
```

Arguments

pts	input powers points
mask	antsImage mask defining target space
radius	for the points

Value

antsImage is output

Author(s)

Avants BB, Duda JT

Examples

```
## Not run:
mni <- antsImageRead(getANTSRData("mni")) %>% getMask()
data("powers_areal_mni_itk", package = "ANTsR", envir = environment())
powersLabels <- makePointsImage(powers_areal_mni_itk[, 1:3], mni, radius = 3)

## End(Not run)
```

makePowersPointsImage *Label image from Powers points*

Description

Creates spherical points in the coordinate space of the target image.

Usage

```
makePowersPointsImage(pts, mask, radius = 5)
```

Arguments

pts	input powers points
mask	antsImage mask defining target space
radius	for the points

Value

antsImage is output

Author(s)

Avants BB, Duda JT

Examples

```
## Not run:
mni <- antsImageRead(getANTSRData("mni")) %>% getMask()
data("powers_areal_mni_itk", package = "ANTsR", envir = environment())
powersLabels <- makePowersPointsImage(powers_areal_mni_itk, mni)
# if you have a different target space
# pts = antsApplyTransformsToPoints( 3, powers_areal_mni_itk,
#   transformlist = concatenatedMaps$toTemplate,
#   whichtoinvert = concatenatedMaps$toTemplateInversion )
```

```
## End(Not run)
```

mapAsymVar	<i>Convert left/right variables to a measure of asymmetry</i>
------------	---

Description

Convert left/right variables to a measure of asymmetry

Usage

```
mapAsymVar(  
  mydataframe,  
  leftvar,  
  leftname = "left",  
  rightname = "right",  
  replacer = "Asym"  
)
```

Arguments

- | | |
|-------------|---|
| mydataframe | dataframe containing relevant variables |
| leftvar | left side variable names ie the full names of the variables to asym |
| leftname | the variable substring indicating left side |
| rightname | the variable substring indicating right side |
| replacer | string to replace left with in column names of output |

Value

fixed x

Author(s)

Avants BB

mapLRAverageVar	<i>Convert left/right variables to an average measurement</i>
-----------------	---

Description

Convert left/right variables to an average measurement

Usage

```
mapLRAverageVar(  
  mydataframe,  
  leftvar,  
  leftname = "left",  
  rightname = "right",  
  replacer = "LRAVG"  
)
```

Arguments

- mydataframe dataframe containing relevant variables
- leftvar left side variable names ie the full names of the variables to average
- leftname the variable substring indicating left side
- rightname the variable substring indicating right side
- replacer string to replace left with in column names of output

Value

fixed x

Author(s)

Avants BB

mask_values	<i>Simple Value extractor for antsImage Values</i>
-------------	--

Description

Takes in a mask and an image and then returns the values

Usage

```
mask_values(x, mask)
```


Arguments

x	Object of class antsImage
mask	object to subset the image. If missing, then all values of the image are plotted.

Value

Vector of values

Examples

```
img <- makeImage(c(10, 10), rnorm(100))
mask <- img > 0
mask_values(img, mask)
mask_values(img)
```

maskImage	<i>Mask input image by mask image.</i>
-----------	--

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)
myimg.mask <- maskImage(myimg, mask)
seg <- kmeansSegmentation(myimg, 3)
myimg.mask <- maskImage(myimg, seg$segmentation, c(1, 3))
```

matrix2timeseries	<i>Simple matrix2timeseries function.</i>
-------------------	---

Description

matrix2timeseries 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	<i>Convert rows of a matrix to images</i>
----------------	---

Description

Unmasks rows of a matrix and writes as images.

Usage

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

Arguments

dataMatrix	A matrix such as that created by <code>imagesToMatrix</code> .
mask	An <code>antsImage</code> 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)
```

MD_JHU_labels_edited	<i>MD JHU labels edited</i>
----------------------	-----------------------------

Description

MD JHU labels edited

mean.antsImage	<i>Mean for antsImage Objects</i>
----------------	-----------------------------------

Description

Overloaded Mean for antsImage objects

Usage

```
## S3 method for class 'antsImage'
mean(x, ..., mask = NULL, na.rm = FALSE)
```

Arguments

x	is an object of class antsImage.
...	additional arguments to pass to mean
mask	binary mask of values to subset
na.rm	Remove missing values

Examples

```
img <- makeImage(c(10, 10, 10), rnorm(1000))
mean(img)
mean(img, na.rm = TRUE)
mean(img > 0)
mean(img > 0, na.rm = TRUE)
mean(img, mask = img > 0)
mean(img, mask = img > 0, na.rm = TRUE)
arr <- as.array(img)
arr[1, 1, 1] <- NA
img <- as.antsImage(arr, reference = img)
m <- mean(img)
stopifnot(is.nan(m))
mean(img, na.rm = TRUE)
m <- mean(0 < img)
stopifnot(is.nan(m))
m <- mean(img > 0)
# stopifnot(is.nan(m))
mean(img > 0, na.rm = TRUE)
m <- mean(img, mask = img > 0)
stopifnot(is.nan(m))
mean(img, mask = img > 0, na.rm = TRUE)
```

median.antsImage	<i>Median for antsImage Objects</i>
------------------	-------------------------------------

Description

Overloaded Median for antsImage objects

Usage

```
## S3 method for class 'antsImage'  
median(x, na.rm = FALSE, ..., mask = NULL)
```

Arguments

x	is an object of class antsImage.
na.rm	a logical value indicating whether NA should be removed
...	additional arguments to send to median
mask	is an object of class antsImage

mergeChannels	<i>merge images into a multiChannel antsImage</i>
---------------	---

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

```

dims <- c(30, 30)
n <- prod(dims)
r <- floor(seq(n) / (n) * 255)
dim(r) <- dims
arr <- r
r <- as.antsImage(r)
g <- r * 0
b <- r * 0
rgbImage <- mergeChannels(list(r, g, b))
testthat::expect_error(mergeChannels(list(arr, arr)))

```

mergeListToNDImage	<i>mergeListToNDImage</i>
--------------------	---------------------------

Description

This function will copy a list of d-1 images into a d-dimension target image. This function reverses the operation performed by `splitNDImageToList`.

Usage

```
mergeListToNDImage(img, imgList)
```

Arguments

<code>img</code>	input image of class <code>antsImage</code> , last dimension will be split
<code>imgList</code>	input list to be merged back into dimension d space

Value

output list filled with d-1 volumes of class `antsImage`

Author(s)

Pustina D, Avants B

Examples

```

my4Dimage <- makeImage(c(5, 5, 5, 4), rnorm(5 * 5 * 5 * 4))
my3Dlist <- splitNDImageToList(my4Dimage)
my4DimageR <- mergeListToNDImage(my4Dimage, my3Dlist)

```

mild	<i>Efficiently compute a multivariate, image-based (mixed) linear decomposition (mild)</i>
------	--

Description

This function simplifies calculating image-wide multivariate PCA maps. The model will minimize a matrix energy similar to $\text{norm}(X - UV^T - U\lambda V^T)$ where the U are standard design and random effect (intercept) design matrices. The random intercept matrix is only included if repeated measures are indicated.

Usage

```
mild(
  dataFrame,
  voxmats,
  basisK,
  myFormulaK,
  smoothingMatrix,
  iterations = 10,
  gamma = 1e-06,
  sparsenessQuantile = 0.5,
  positivity = c("positive", "negative", "either"),
  initializationStrategy = 0,
  repeatedMeasures = NA,
  orthogonalize = FALSE,
  verbose = FALSE
)
```

Arguments

dataFrame	This data frame contains all relevant predictors except for the matrices associated with the image variables.
voxmats	The named list of matrices that contains the changing predictors.
basisK	an integer determining the size of the basis.
myFormulaK	This is a character string that defines a valid regression which in this case should include predictors named as <code>paste0("mildBasis", 1:basisK)</code>
smoothingMatrix	allows parameter smoothing, should be square and same size as input matrix
iterations	number of gradient descent iterations
gamma	step size for gradient descent
sparsenessQuantile	quantile to control sparseness - higher is sparser
positivity	restrict to positive or negative solution (beta) weights. choices are positive, negative or either as expressed as a string.

`initializationStrategy` optional initialization matrix or seed. seed should be a single number; matrix should be a n by k matrix.

`repeatedMeasures` list of repeated measurement identifiers. this will allow estimates of per identifier intercept.

`orthogonalize` boolean to control whether we orthogonalize the v

`verbose` boolean to control verbosity of output

Value

A list of different matrices that contain names derived from the formula and the coefficients of the regression model.

Author(s)

BB Avants.

See Also

[milr](#)

Examples

```
set.seed(1500)
nsub <- 12
npix <- 100
outcome <- rnorm(nsub)
covar <- rnorm(nsub)
mat <- replicate(npix, rnorm(nsub))
mat2 <- replicate(npix, rnorm(nsub))
nk <- 3
myform <- paste(
  " vox2 ~ covar + vox + ",
  paste0("mildBasis", 1:nk, collapse = "+")
) # optional covariates
df <- data.frame(outcome = outcome, covar = covar)
result <- mild(df, list(vox = mat, vox2 = mat2),
  basisK = 3, myform,
  initializationStrategy = 10
)
result <- mild(df, list(vox = mat, vox2 = mat2),
  basisK = 3, myform,
  initializationStrategy = 4
)
myumat <- svd(mat2, nv = 0, nu = 3)$u
result <- mild(df, list(vox = mat, vox2 = mat2),
  basisK = 3, myform,
  initializationStrategy = 0
)
```

milr	<i>Efficiently compute a multivariate, penalized image-based linear regression model (milr)</i>
------	---

Description

This function simplifies calculating image-wide multivariate beta maps from linear models. Image-based variables are stored in the input matrix list. They should be named consistently in the input formula and in the image list. If they are not, an error will be thrown. All input matrices should have the same number of rows and columns. The model will minimize a matrix energy similar to $\text{norm}(X - UV^T - U_{\text{ran}}V_{\text{ran}}^T)$ where the U are standard design and random effect (intercept) design matrices. The random intercept matrix is only included if repeated measures are indicated.

Usage

```
milr(
  dataFrame,
  voxmats,
  myFormula,
  smoothingMatrix,
  iterations = 10,
  gamma = 1e-06,
  sparsenessQuantile,
  positivity = c("positive", "negative", "either"),
  repeatedMeasures = NA,
  orthogonalize = FALSE,
  verbose = FALSE
)
```

Arguments

dataFrame	This data frame contains all relevant predictors except for the matrices associated with the image variables.
voxmats	The named list of matrices that contains the changing predictors.
myFormula	This is a character string that defines a valid regression formula.
smoothingMatrix	allows parameter smoothing, should be square and same size as input matrix
iterations	number of gradient descent iterations
gamma	step size for gradient descent
sparsenessQuantile	quantile to control sparseness - higher is sparser
positivity	restrict to positive or negative solution (beta) weights. choices are positive, negative or either as expressed as a string.
repeatedMeasures	list of repeated measurement identifiers. this will allow estimates of per identifier intercept.

orthogonalize boolean to control whether we orthogonalize the v
 verbose boolean to control verbosity of output

Value

A list of different matrices that contain names derived from the formula and the coefficients of the regression model.

Author(s)

BB Avants.

Examples

```
set.seed(1500)
nsub <- 12
npix <- 100
outcome <- rnorm(nsub)
covar <- rnorm(nsub)
mat <- replicate(npix, rnorm(nsub))
mat2 <- replicate(npix, rnorm(nsub))
myform <- " vox2 ~ covar + vox "
df <- data.frame(outcome = outcome, covar = covar)
result <- milr(df, list(vox = mat, vox2 = mat2), myform)

## Not run:
# a 2nd example with 3 modalities
imageIDs <- c("r16", "r27", "r30", "r62", "r64", "r85")
images <- list()
feature1Images <- list()
feature2Images <- list()
feature3Images <- list()
ref <- antsImageRead(getANTsRData("r16"))
mask <- ref * 0
for (i in 1:length(imageIDs)) {
  cat("Processing image", imageIDs[i], "\n")
  tar <- antsImageRead(getANTsRData(imageIDs[i]))
  images[[i]] <- tar
  mask <- mask + tar
  feature1Images[[i]] <- iMath(images[[i]], "Grad", 1.0)
  feature2Images[[i]] <- iMath(images[[i]], "Laplacian", 1.0)
  feature3Images[[i]] <- reflectImage(tar, axis = 0, tx = "Affine")$warpedmovout
}
i <- 1
mask <- getMask(mask)
spatmat <- t(imageDomainToSpatialMatrix(mask, mask))
smoomat <- knnSmoothingMatrix(spatmat, k = 23, sigma = 100.0)
mat <- imageListToMatrix(images, mask)
mat2 <- imageListToMatrix(feature1Images, mask)
mat3 <- imageListToMatrix(feature3Images, mask)
myscale <- function(x) {
  return(scale(x, center = TRUE, scale = T))
}
```

```

}
x <- list(x1 = myscale(mat[-5, ]), x2 = myscale(mat2[-5, ]), x3 = myscale(mat3[-5, ]))
df <- data.frame(m1 = rowMeans(x$x1), m2 = rowMeans(x$x2), m3 = rowMeans(x$x3))
myform <- " x1 ~ x2 + x3 + m2 + m3 "
result <- milr(df, x, myform,
  iterations = 32, smoothingMatrix = smoomat,
  gamma = 1e-1, verbose = T
)
k <- 1
mm <- makeImage(mask, (result$prediction[k, ])) %>% iMath("Normalize")
tar <- makeImage(mask, x$x1[k, ])
plot(mm, doCropping = FALSE)
plot(tar, doCropping = FALSE)
cor.test(result$prediction[k, ], x$x1[k, ])
myol <- makeImage(mask, abs(result$v[, "x2"])) %>% iMath("Normalize")
plot(tar, myol, doCropping = FALSE)
myol <- makeImage(mask, abs(result$v[, "m3"])) %>% iMath("Normalize")
plot(tar, myol, doCropping = FALSE)
result <- milr(df, x, myform,
  iterations = 11, smoothingMatrix = smoomat,
  sparsenessQuantile = 0.5, positivity = "positive",
  gamma = 1e-2, verbose = T
)
myol <- makeImage(mask, abs(result$v[, "x2"])) %>% iMath("Normalize")
plot(tar, myol, doCropping = FALSE, window.overlay = c(0.1, 1))
mm <- makeImage(mask, (result$prediction[k, ])) %>% iMath("Normalize")
tar <- makeImage(mask, x$x1[k, ])
plot(mm, doCropping = FALSE)
plot(tar, doCropping = FALSE)
cor.test(result$prediction[k, ], x$x1[k, ])
# univariate outcome
myform <- " m1 ~ x2 + x3 + m2 + m3 "
result <- milr(df, x, myform,
  iterations = 11, smoothingMatrix = smoomat,
  gamma = 1e-2, verbose = T
)

## End(Not run)

```

milr.predict

Predict from a milr output

Description

This function computes a prediction or low-dimensional embedding, given milr output. It will return a predictive model if the outcome variable is a scalar. Otherwise, it will return a low-dimensional embedding without a specific predictive model.

Usage

```

milr.predict(
  milrResult,
  dataFrameTrain,
  voxmatsTrain,
  dataFrameTest,
  voxmatsTest,
  myFormula
)

```

Arguments

<code>milrResult</code>	This output form milr
<code>dataFrameTrain</code>	This data frame contains all relevant predictors in the training data except for the matrices associated with the image variables.
<code>voxmatsTrain</code>	The named list of matrices that contains the changing predictors.
<code>dataFrameTest</code>	This data frame contains all relevant predictors in the training data except for the matrices associated with the image variables in test data.
<code>voxmatsTest</code>	The named list of matrices that contains the changing predictors in test data.
<code>myFormula</code>	This is a character string that defines a valid regression formula.

Value

the predicted matrix.

Author(s)

BB Avants.

See Also

[milr](#)

Examples

```

nsub <- 24
npix <- 100
outcome <- rnorm(nsub)
covar <- rnorm(nsub)
mat <- replicate(npix, rnorm(nsub))
mat2 <- replicate(npix, rnorm(nsub))
mat3 <- replicate(npix, rnorm(nsub))
myform <- " vox2 ~ covar + vox + vox3 "
istr <- c(rep(TRUE, round(nsub * 2 / 3)), rep(FALSE, nsub - round(nsub * 2 / 3)))
df <- data.frame(outcome = outcome, covar = covar)
ltr <- list(vox = mat[istr, ], vox2 = mat2[istr, ], vox3 = mat3[istr, ])
lte <- list(vox = mat[!istr, ], vox2 = mat2[!istr, ], vox3 = mat3[!istr, ])
result <- milr(df[istr, ], ltr, myform)
pred <- milr.predict(result, df[istr, ], ltr, df[!istr, ], lte, myform)

```

`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://imaging.mrc-cbu.cam.ac.uk/imaging/MniTalairach>

Examples

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

morphology	<i>Morphological image operations</i>
------------	---------------------------------------

Description

Apply image morphological operations

Usage

```
morphology(  
  input,  
  operation,  
  radius,  
  type = "binary",  
  value = 1,  
  shape = "ball",  
  radiusIsParametric = FALSE,  
  thickness = 1,  
  lines = 3,  
  includeCenter = FALSE  
)
```

Arguments

input	input image
operation	operation to apply "close" Morpholgical closing "dilate" Morpholgical dilation "erode" Morpholgical erosion "open" Morpholgical opening
radius	radius of structuring element
type	type of morphology "binary" Binary operation on a single value "grayscale" Grayscale operations
value	value to operation on (type='binary' only)
shape	shape of the structuring element (type='binary' only) "ball" spherical structuring element "box" box shaped structuring element "cross" cross shaped structuring element "annulus" annulus shaped structuring element "polygon" polygon structuring element
radiusIsParametric	used parametric radius boolean (shape='ball' and shape='annulus' only)

thickness thickness (shape='annulus' only)
 lines number of lines in polygon (shape='polygon' only)
 includeCenter include center of annulus boolean (shape='annulus' only)

Value

antsImage is output

Author(s)

Avants BB

Examples

```
fi <- antsImageRead(getANTsRData("r16"), 2)
mask <- getMask(fi)
dilatedBall <- morphology(mask,
  operation = "dilate", radius = 3,
  type = "binary", shape = "ball"
)
erodedBox <- morphology(mask,
  operation = "erode", radius = 3,
  type = "binary", shape = "box"
)
openedAnnulus <- morphology(mask,
  operation = "open", radius = 5,
  type = "binary", shape = "annulus", thickness = 2
)
out <- morphology(mask, operation = "open", radius = 5, type = "binary", shape = "polygon")
out <- morphology(mask, operation = "open", radius = 5, type = "binary", shape = "cross")
out <- morphology(mask, operation = "close", radius = 5, type = "binary", shape = "polygon")
out <- morphology(mask, operation = "erode", radius = 5, type = "binary", shape = "polygon")
out <- morphology(mask, operation = "dilate", radius = 5, type = "binary", shape = "polygon")
testthat::expect_error(morphology(mask, operation = "open", radius = 5, shape = "hey"))
```

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,
  labelmasks,
  rad = NA,
  nsamples = 1,
  ntrees = 500,
  multiResSchedule = c(4, 2, 1),
  asFactors = TRUE,
  voxchunk = 50000,
  ...
)
```

Arguments

<code>y</code>	list of training labels. either an image or numeric value
<code>x</code>	a list of lists where each list contains feature images
<code>labelmasks</code>	a mask (or list of masks) used to define where the samples will come from. Note, two labels (e.g., GM and WM) will double the number of samples from each feature images. If the mask is binary, samples are selected randomly within 1 values.
<code>rad</code>	vector of dimensionality d define nhooood radius
<code>nsamples</code>	(per subject to enter training)
<code>ntrees</code>	(for the random forest model)
<code>multiResSchedule</code>	an integer vector defining multi-res levels
<code>asFactors</code>	boolean - treat the y entries as factors
<code>voxchunk</code>	value of maximal voxels to predict at once. This value is used to split the prediction into smaller chunks such that memory requirements do not become too big
<code>...</code>	arguments to pass to vwnrfs

Value

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

Author(s)

Avants BB, Tustison NJ, Pustina D

Examples

```
mask <- makeImage(c(10, 10), 0)
mask[3:6, 3:6] <- 1
mask[5, 5:6] <- 2
```



```

ilist <- list()
lablist <- list()
inds <- 1:5
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 <- mrwnrfs(lablist, ilist, mask,
    rad = rad, multiResSchedule = mr,
    asFactors = (predtype == "label")
  )
  rfmresult <- mrwnrfs.predict(rfm$rflist,
    ilist, mask,
    rad = rad, asFactors = (predtype == "label"),
    multiResSchedule = mr
  )
  if (predtype == "scalar") {
    print(cor(unlist(lablist), unlist(rfmresult$seg)))
  }
} # end predtype loop

```

mrwnrfs.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

```

mrvmrfs.predict(
  rflist,
  x,
  labelmasks,
  rad = NA,
  multiResSchedule = c(4, 2, 1),
  asFactors = TRUE,
  voxchunk = 60000
)

```

Arguments

rflist	a list of random forest models from mrvmrfs
x	a list of lists where each list contains feature images
labelmasks	a mask (or list of masks) used to define the area to predict. This is used to save time by constrain the prediction in within the brain.
rad	vector of dimensionality d define nhoud radius
multiResSchedule	an integer vector defining multi-res levels
asFactors	boolean - treat the y entries as factors
voxchunk	value of maximal voxels to predict at once. This value is used to split the prediction into smaller chunks such that memory requirements do not become too big

Value

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

Author(s)

Avants BB, Tustison NJ, Pustina D

mtl_description	<i>mtl_description</i>
-----------------	------------------------

Description

mtl description

mtl_labels	<i>mtl_labels</i>
------------	-------------------

Description

A data frame label numbers and names for medial temporal lobe regions.

Usage

```
mtl_labels
```

Format

A data frame with columns: Label, Description.

multichannelPCA	<i>Shape analysis of multichannel images based on PCA</i>
-----------------	---

Description

Converts a set of multichannel images (e.g. deformation fields) to a matrix to enable various forms of PCA. Returns the components of shape variability and variance explained. May employ different decomposition methods (WIP).

Usage

```
multichannelPCA(  
  x,  
  mask,  
  k = NA,  
  pcaOption = "PCA",  
  auxiliaryModality,  
  center = TRUE,  
  sigma = NA,  
  verbose = FALSE  
)
```

Arguments

- | | |
|-----------|--|
| x | list containing multichannel images all of the same size |
| mask | mask to apply to the multichannel images |
| k | rank to use |
| pcaOption | currently only PCA and randPCA, latter being much faster. We also allow fastICA. |

auxiliaryModality	if you pass this matrix, then will do CCA. This will only work with one option.
center	subtract the mean column vector.
sigma	parameter for kernel PCA.
verbose	produces more explanatory output.

Value

list of the pca output and conversion to multichannel images

Author(s)

Avants BB

Examples

```
img1 <- antsImageRead(getANTsRData("r16")) %>%
  resampleImage(c(4, 4))
img2 <- antsImageRead(getANTsRData("r64")) %>%
  resampleImage(c(4, 4))
img3 <- antsImageRead(getANTsRData("r27")) %>%
  resampleImage(c(4, 4))
img4 <- antsImageRead(getANTsRData("r30")) %>%
  resampleImage(c(4, 4))
reg1 <- antsRegistration(img1, img2, "SyN")
reg2 <- antsRegistration(img1, img3, "SyN")
reg3 <- antsRegistration(img1, img4, "SyN")
w1 <- antsImageRead(reg1$fdttransforms[1])
w2 <- antsImageRead(reg2$fdttransforms[1])
w3 <- antsImageRead(reg3$fdttransforms[1])
mask <- getMask(img1)
x <- list(w1, w2, w3)
dpca <- multichannelPCA(x, mask)
warpTx <- antsRTransformFromDisplacementField(dpca$pcaWarps[[1]])
warped <- applyAntsRTransform(warpTx, data = img1, reference = img1)
```

multichannelToVector *Convert multichannel antsImage to a vector.*

Description

Employs a mask to flatten each channel of a multi-channel image into a contiguous piece of a vector.

Usage

```
multichannelToVector(multichannelimage, mask)
```

Arguments

- multichannelimage input multichannel image
- mask mask of same dimensionality as multichannelimage

Value

vector is output

Author(s)

Avants BB

Examples

see vectorToMultichannel

multigrep	<i>Grep entries with a vector search parameters</i>
-----------	---

Description

Grep entries with a vector search parameters

Usage

multigrep(x, desc, intersect = FALSE)

Arguments

- x a vector of search terms
- desc target vector of items to be searched
- intersect boolean whether to use intersection or union otherwise

Value

result of grep (indices of desc that match x)

Author(s)

Avants BB

multiLabelMorphology *Morphology on multi label images.*

Description

Wraps calls to iMath binary morphology. Additionally, dilation and closing operations preserve pre-existing labels. The choices of operation are:

Usage

```
multiLabelMorphology(
  labelsIn,
  operation,
  radius,
  dilationMask = NULL,
  labelList = NA
)
```

Arguments

labelsIn	Input image should contain only 0 for background and positive integers for labels.
operation	One of 'MD', 'ME', 'MC', 'MO', passed to iMath.
radius	radius of the morphological operation.
dilationMask	Optional binary mask to constrain dilation only (eg dilate cortical label into WM).
labelList	Optional list of labels, to perform operation upon. Defaults to all unique intensities in labels.in.

Details

Dilation: dilates all labels sequentially, but does not overwrite original labels. This reduces dependence on the intensity ordering of adjoining labels. Ordering dependence can still arise if two or more labels dilate into the same space - in this case, the label with the lowest intensity is retained. With a mask, dilated labels are multiplied by the mask and then added to the original label, thus restricting dilation to the mask region.

Erosion: Erodes labels independently, equivalent to calling iMath iteratively.

Closing: Close holes in each label sequentially, but does not overwrite original labels. Opening: Opens each label independently, equivalent to calling iMath iteratively.

Value

processed labels, of type antsImage.

Author(s)

Cook PA

Examples

```
img <- antsImageRead(getANTsRData("r16"), 2)
labels <- getMask(img, 1, 150) + getMask(img, 151, 225) * 2
labelsDilated <- multiLabelMorphology(labels, "MD", 2)

# should see original label regions preserved in dilated version
# label N should have mean N and 0 variance
labelStats(labelsDilated, labels)
```

multiResRandomForestRegression

multiple resolution neighborhood random forest regression

Description

Represents feature images as a neighborhood across scales to build a random forest prediction from an image population. A use case for this function is to predict cognition from multiple image features, e.g. from the voxelwise FA of the corpus callosum and, in parallel, voxelwise measurements of the volume of the inferior frontal gyrus.

Usage

```
multiResRandomForestRegression(
  y,
  x,
  labelmasks,
  rad = NA,
  nsamples = 10,
  multiResSchedule = c(0),
  ntrees = 500
)
```

Arguments

y	vector of scalar values or labels. if a factor, do classification, otherwise regression.
x	a list of lists where each list contains feature images
labelmasks	a list of masks where each mask defines the image space for the given list and the number of parallel predictors. more labels means more predictors. alternatively, separate masks may be used for each feature in which case this should be a list of lists. see examples.
rad	vector of dimensionality d define nhood radius

nsamples (per subject to enter training)
multiResSchedule
 an integer vector defining multi-res levels
ntrees (for the random forest model)

Value

list with a random forest model, a vector identifying which rows correspond to which subjects and a prediction vector.

Author(s)

Avants BB, Tustison NJ

References

Pustina, D, et al. Automated segmentation of chronic stroke lesions using LINDA: Lesion Identification with Neighborhood Data Analysis, Human Brain Mapping, 2016. (related work, not identical)

See Also

[getMultiResFeatureMatrix mrvnrfs](#)

Examples

```
mask <- makeImage(c(100, 100), 0)
mask[30:60, 30:60] <- 1
mask[35:45, 50:60] <- 2
ilist <- list()
masklist <- list()
inds <- 1:8
yvec <- rep(0, length(inds))
scl <- 0.33 # a noise parameter
for (i in inds) {
  img <- antsImageClone(mask)
  imgb <- antsImageClone(mask)
  limg <- antsImageClone(mask)
  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))
  ilist[[i]] <- list(img, imgb) # two features
  yvec[i] <- i^2.0 # a real outcome
  masklist[[i]] <- antsImageClone(mask)
}
r <- c(1, 1)
mr <- c(2, 0)
featMat <- getMultiResFeatureMatrix(ilist[[1]], masklist[[1]],
  rad = r, , multiResSchedule = mr
)
rfm <- multiResRandomForestRegression(
  yvec, ilist, masklist,
```



```

    rad = r, multiResSchedule = mr
  )
  preds <- predict(rfm, newdata = featMat)
  ## Not run:
  # data: https://github.com/ANTsX/ANTsTutorial/tree/master/phantomData
  fns <- Sys.glob("phantom*wmgm.jpg")
  ildist <- imageFileNames2ImageList(fns)
  masklist <- list()
  flist <- list()
  for (i in 1:length(fns))
  {
    # 2 labels means 2 sets of side by side predictors and features at each scale
    locseg <- kmeansSegmentation(ildist[[i]], 2)$segmentation
    masklist[[i]] <- list(locseg, locseg %>% thresholdImage(2, 2), locseg)
    flist[[i]] <- list(
      ildist[[i]], ildist[[i]] %>% iMath("Laplacian", 1),
      ildist[[i]] %>% iMath("Grad", 1)
    )
  }
  yvec <- factor(rep(c(1, 2), each = 4)) # classification
  r <- c(1, 1)
  mr <- c(2, 1, 0)
  ns <- 50
  trn <- c(1:3, 6:8)
  ytrain <- yvec[trn]
  ftrain <- flist[trn]
  mtrain <- masklist[trn]
  mrrfr <- multiResRandomForestRegression(ytrain, ftrain, mtrain,
    rad = c(1, 1),
    nsamples = ns, multiResSchedule = mr
  )
  mypreds <- rep(NA, length(fns))
  mymode <- function(x) {
    ux <- unique(x)
    ux[which.max(tabulate(match(x, ux)))]
  }
  for (i in 4:5) # test set
  {
    fmat <- getMultiResFeatureMatrix(flist[[i]], masklist[[i]],
      rad = r, multiResSchedule = mr, nsamples = ns
    )
    myp <- predict(mrrfr, newdata = fmat)
    mypreds[i] <- mymode(myp) # get the most frequent observation
    # use median or mean for continuous predictions
  }
  print("predicted")
  print(mypreds[-trn])
  print("ground truth")
  print(yvec[-trn])

  ## End(Not run)

```

multiscaleSVD

*Multi-scale svd***Description**

Maggioni's multi-scale SVD algorithm explores the dimensionality of a dataset by investigating the change in eigenvalues with respect to a scale parameter. The scale parameter is defined by the radius of a ball that sits at each point in the data. The ball, at each scale, is moved across the dataset and SVD is computed within the intersection of the ball and the data at each point. The shape in this collection of eigenvalues, with respect to scale, enables us to estimate both signal and noise dimensionality and scale. The estimate can be computed efficiently on large datasets if the sampling is chosen appropriately. The challenge, in this algorithm, is classifying the dimensions of noise, curvature and data. This classification currently uses variations on heuristics suggested in work by Maggioni et al.

Usage

```
multiscaleSVD(x, r, locn, nev, knn = 0, verbose = FALSE, plot = 0)
```

Arguments

x	input matrix, should be n (samples) by p (measurements)
r	radii to explore
locn	number of local samples to take at each scale
nev	maximum number of eigenvalues to compute
knn	randomly sample neighbors to assist with large datasets. set k with this value.
verbose	boolean to control verbosity of output
plot	boolean to control whether we plot results. its value determines which eigen-vector off which to base the scale of the y-axis.'

Value

list with a vector of tangent, curvature, noise dimensionality and a dataframe containing eigenvalues across scale, in correspondence with r:

dim: The tangent, curvature and noise dimensionality vector. The data dimensionality is the first entry, the curvature dimensionality exists from the second to the first entry of the noise vector.

noiseCutoffs: Dimensionalities where the noise may begin. These are candidate cutoffs but may contain some curvature information.'

evalsVsScale: eigenvalues across scale

evalClustering: data-driven clustering of the eigenvalues

Author(s)

Avants BB

References

Maggioni, M., et al. "Multiscale data geometry."

Examples

```
sphereDim <- 9
embeddDim <- 100
n <- 1000
if (usePkg("pracma")) {
  set.seed(20190919)
  sphereData <- pracma::rands(n, sphereDim, 1.)
  mysig <- 0.1
  spherEmbed <- matrix(rnorm(n * embeddDim, 0, mysig), nrow = n, ncol = embeddDim)
  spherEmbed[, 1:ncol(sphereData)] <- spherEmbed[, 1:ncol(sphereData)] + sphereData
  myr <- seq(1.0, 2.2, 0.05) # scales at which to sample
  mymssvd <- multiscaleSVD(spherEmbed, myr, locn = 5, nev = 20, plot = 1)
  if (getRversion() < "3.6.0") {
    testthat::expect_equal(mymssvd$noiseCutoffs, c(10, 11))
    cm <- unname(colMeans(mymssvd$evalsVsScale[11:25, ]))
    testthat::expect_equal(
      cm,
      c(
        0.133651668406975, 0.0985695151401464, 0.0914110478052329,
        0.086272017653314, 0.081188302173622, 0.0766100356616153, 0.0719736252996842,
        0.067588745051721, 0.0622331185687704, 0.0415236318358749, 0.0192976885668337,
        0.0183063537558787, 0.0174990088862745, 0.0170012938275551, 0.0163859378707545,
        0.0158265354487181, 0.0153357773252783, 0.0147933538908736, 0.0143510807701235,
        0.0140473978346935
      )
    )
  } else {
    testthat::expect_equal(mymssvd$noiseCutoffs, c(11, 15))
    cm <- unname(colMeans(mymssvd$evalsVsScale[13:25, ]))
    testthat::expect_equal(
      cm,
      c(
        0.138511257441516, 0.106071822485487, 0.0989441114152412, 0.092910922851038,
        0.0877970523897918, 0.0832570763653118, 0.0782599820599334, 0.0734433988152632,
        0.0678992413676906, 0.0432283615430504, 0.0202481578919003, 0.0191747572787057,
        0.0185718929604774, 0.0178301092823977, 0.0172423799670431, 0.0166981650233669,
        0.0162072551503541, 0.015784555784915, 0.0153600119986575, 0.0149084240854556
      )
    )
  }
}
```

Description

Perform Multiview PCA on multiple datasets with an option for sparse PCA.

Usage

```
multiview_pca(
  views,
  n_components,
  sparse = 0.5,
  max_iter = 100,
  sparsenessAlg = "basic",
  verbose = FALSE
)
```

Arguments

<code>views</code>	A list of data matrices for each view.
<code>n_components</code>	Number of principal components to compute.
<code>sparse</code>	vector of length views with values between zero and one
<code>max_iter</code>	Maximum number of iterations for the optimization.
<code>sparsenessAlg</code>	NA is default otherwise basic, spmp or orthorank
<code>verbose</code>	Logical, whether to print information about energy and sparsity.

Value

A list containing the common representation Z and transformation matrices W .

Examples

```
set.seed(123)
n_samples <- 100
n_features_1 <- 50
n_features_2 <- 60
n_features_3 <- 70
n_components <- 5
view1 <- matrix(rnorm(n_samples * n_features_1), nrow = n_samples, ncol = n_features_1)
view2 <- matrix(rnorm(n_samples * n_features_2), nrow = n_samples, ncol = n_features_2)
view3 <- matrix(rnorm(n_samples * n_features_3), nrow = n_samples, ncol = n_features_3)
result <- multiview_pca(list(view1, view2, view3), n_components, sparse = rep(0.5,3),
  verbose = TRUE)
print(result)
```

`n3BiasFieldCorrection` *N3 bias field correction*

Description

Perform N3 bias field correction on the given image

Usage

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

Arguments

<code>img</code>	antsImage to correct
<code>downsampleFactor</code>	integer e.g. 4 downsample by a factor of 4
<code>...</code>	not used

Value

antsImage

Author(s)

Shrinidhi KL

Examples

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

`n3BiasFieldCorrection2`
N3 Bias field correction

Description

Perform N3 bias field correction on the given image

Usage

```
n3BiasFieldCorrection2(
  img,
  mask,
  rescaleIntensities = FALSE,
  shrinkFactor = 4,
  convergence = list(iters = 50, tol = 1e-07),
  splineParam = 200,
  numberOfFittingLevels = 4,
  weightMask = NULL,
  returnBiasField = FALSE,
  verbose = FALSE
)
```

Arguments

<code>img</code>	input antsImage
<code>mask</code>	input mask, if one is not passed one will be made
<code>rescaleIntensities</code>	At each iteration, a new intensity mapping is calculated and applied but there is nothing which constrains the new intensity range to be within certain values. The result is that the range can "drift" from the original at each iteration. This option rescales to the [min,max] range of the original image intensities within the user-specified mask. A mask is required to perform rescaling. Default is FALSE in ANTsR/ANTsPy but TRUE in ANTs.
<code>shrinkFactor</code>	Shrink factor for multi-resolution correction, typically integer less than 4
<code>convergence</code>	List of: iters, maximum number of iterations and tol, the convergence tolerance. Default tolerance is 1e-7 in ANTsR/ANTsPy but 0.0 in ANTs.
<code>splineParam</code>	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. Default is a mesh size of 1 per dimension.
<code>numberOfFittingLevels</code>	Parameter controlling number of fitting levels.
<code>weightMask</code>	antsImage of weight mask
<code>returnBiasField</code>	bool, return the field instead of the corrected image.
<code>verbose</code>	enables verbose output.

Value

bias corrected image or bias field

Author(s)

Avants BB, Tustison NJ

Examples

```

dims <- c(50, 50)
img <- makeImage(imagesize = dims, rnorm(prod(dims)))
n3img <- n3BiasFieldCorrection2(img)
n3img <- n3BiasFieldCorrection2(img, mask = img > 0)
testthat::expect_error(n3BiasFieldCorrection2(img, weightMask = "somepath"))
testthat::expect_error(n3BiasFieldCorrection2(img, splineParam = rep(200, 3)))
# n3img<-n3BiasFieldCorrection2(img, splineParam = c(200, 20)) # long running

rm(img)
gc()
rm(n3img)
gc()
fname <- getANTsRData("r16")
in_img <- antsImageRead(fname)
n3 <- n3BiasFieldCorrection2(in_img)
rm(n3)
gc()
mask <- in_img > 0
mask2 <- antsImageClone(mask, out_pixeltype = "float")
# fails
mask
sum(mask)
## Not run:
n3 <- n3BiasFieldCorrection2(in_img, mask = mask, verbose = TRUE)
# fails
n3 <- n3BiasFieldCorrection2(in_img, mask = mask2)

## End(Not run)

```

n4BiasFieldCorrection *N4 bias field correction*

Description

Perform N4 bias field correction on the given image

Usage

```

n4BiasFieldCorrection(
  img,
  mask,
  rescaleIntensities = FALSE,
  shrinkFactor = 4,
  convergence = list(iters = c(50, 50, 50, 50), tol = 1e-07),
  splineParam = 200,
  returnBiasField = FALSE,
  verbose = FALSE,
  weightMask = NULL
)

```

Arguments

<code>img</code>	input antsImage
<code>mask</code>	input mask, if one is not passed one will be made
<code>rescaleIntensities</code>	At each iteration, a new intensity mapping is calculated and applied but there is nothing which constrains the new intensity range to be within certain values. The result is that the range can "drift" from the original at each iteration. This option rescales to the [min,max] range of the original image intensities within the user-specified mask. A mask is required to perform rescaling. Default is FALSE in ANTsR/ANTsPy but TRUE in ANTs.
<code>shrinkFactor</code>	Shrink factor for multi-resolution correction, typically integer less than 4.
<code>convergence</code>	List of: <code>iters</code> , vector of maximum number of iterations for each shrinkage factor, and <code>tol</code> , the convergence tolerance. Default tolerance is 1e-7 in ANTsR/ANTsPy but 0.0 in ANTs.
<code>splineParam</code>	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. Default in ANTsR/ANTsPy is 200 mm per mesh element in each dimension. The ANTs default is a mesh size of 1 per dimension.
<code>returnBiasField</code>	bool, return the field instead of the corrected image.
<code>verbose</code>	enables verbose output.
<code>weightMask</code>	antsImage of weight mask

Value

bias corrected image or bias field

Author(s)

BB Avants

Examples

```

dims <- c(50, 50)
img <- makeImage(imagesize = dims, rnorm(prod(dims)))
n4img <- n4BiasFieldCorrection(img)
n4img <- n4BiasFieldCorrection(img, mask = img > 0)
testthat::expect_error(n4BiasFieldCorrection(img, weightMask = "somepath"))
testthat::expect_error(n4BiasFieldCorrection(img, splineParam = rep(200, 3)))
n4img <- n4BiasFieldCorrection(img, splineParam = c(200, 20))

rm(img)
gc()
rm(n4img)
gc()
fname <- getANTsRData("r16")

```



```
in_img <- antsImageRead(fname)
n4 <- n4BiasFieldCorrection(in_img)
rm(n4)
gc()
mask <- in_img > 0
mask2 <- antsImageClone(mask, out_pixeltype = "float")
# fails
mask
sum(mask)
## Not run:
n4 <- n4BiasFieldCorrection(in_img, mask = mask, verbose = TRUE)
# fails
n4 <- n4BiasFieldCorrection(in_img, mask = mask2)

## End(Not run)
```

nbm3_labels	<i>nbm3_labels</i>
-------------	--------------------

Description

A data frame label numbers and names for NBM3 regions.

Usage

```
nbm3_labels
```

Format

A data frame with columns: Label, Description, Side.

nbm3CH13	<i>nbm3CH13</i>
----------	-----------------

Description

```
nbm3CH13
```

neg_violation	<i>Nonnegativity Violation</i>
---------------	--------------------------------

Description

Measures the extent to which a matrix violates nonnegativity constraints.

Usage

```
neg_violation(Y)
```

Arguments

Y Numeric matrix (p x k) to evaluate.

Details

Computes $||\min(Y, 0)||_F$, the Frobenius norm of the negative part of Y, which quantifies the magnitude of negative entries. Used to monitor nonnegativity enforcement in NSA-Flow.

Value

Numeric scalar, the Frobenius norm of negative entries.

Examples

```
Y <- matrix(c(-1, 2, -3, 4), 2, 2)
neg_violation(Y)
```

networkEiganat	<i>Convenience wrapper for eigenanatomy decomposition.</i>
----------------	--

Description

Decomposes a matrix into sparse eigenvectors 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 solution
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$eig)
  snpse <- as.matrix(snps[train, ]) %%% projmat
  traindf <- data.frame(bmi = numericalpheno[train, phind], snpse = snpse)
  snpse <- as.matrix(snps[!train, ]) %%% projmat
  testdf <- data.frame(bmi = numericalpheno[!train, phind], snpse = snpse)
  myrf <- glm(bmi ~ ., data = traindf)
  preddf <- predict(myrf, newdata = testdf)
  cor.test(predf, testdf$bmi)
} # ch1 and ch2
#####

## End(Not run)
```

NNHEmbed

Perform SiMLR Analysis on Multimodal ANTsPyMM Data

Description

This function processes multimodal data using SiMLR. It is designed to be flexible, allowing for various preprocessing steps and analysis options. The analysis can be adjusted through multiple parameters, offering control over the inclusion of certain data types, permutation testing, and more.

Usage

```

NNHEmbed(
  blaster,
  select_training_boolean,
  connect_cog,
  energy,
  nsimlr,
  constraint,
  covariates = "1",
  myseed = 3,
  doAsym = TRUE,
  returnidps = 0,
  restrictDFN = FALSE,
  resnetGradeThresh = 1.02,
  doperm = FALSE,
  exclusions = NULL,
  inclusions = NULL,
  sparseness = NULL,
  mixAlg = NULL,
  iterations = NULL,
  path_modeling = NULL,
  sparsenessAlg = "soft",
  optimizationStyle = NULL,
  domainMatrices = NULL,
  domainLambdas = NULL,
  verbose = FALSE
)

```

Arguments

<code>blaster</code>	A dataframe containing multimodal data for analysis.
<code>select_training_boolean</code>	boolean vector to define which entries are in training data
<code>connect_cog</code>	Vector of column names to be treated as a special target matrix; often used for cognitive data and in a supervised variant of simlr. Exclude this argument if this is unclear.

energy	The type of energy model to use for similarity analysis. Usually 'reg' or 'cca'. Other options include 'lrr', 'regression', 'nc', 'base.pca', 'base.spca', 'base.rand.1', and 'base.rand.0'.
nsimlr	Number of components.
constraint	orthogonality constraint of the form constraintxFloatWeightEnergyxFloatWeight-Grad where constraints is ortho or nsafow or none
covariates	any covariates to adjust training matrices. if covariates is set to 'mean' then the rowwise mean will be factored out of each matrix. this can be a vector e.g. c('center', 'scale', 'rank'). pass the name opt to NNHEmbed_update_residuals to have the function print the options.
myseed	Seed for random number generation to ensure reproducibility. Defaults to 3.
doAsym	integer 0 for FALSE, 1 for TRUE and 2 for separate matrices for asymm variables.
returnidps	Integer indicating whether to return the idp names (1) or matrices (2) or neither (0). Defaults to 0.
restrictDFN	Logical indicating whether to restrict analysis to default network features. Defaults to FALSE.
resnetGradeThresh	image quality threshold (higher better).
doperm	Logical indicating whether to perform permutation tests. Defaults to FALSE. Will randomize image features in the training data and thus leads to "randomized" but still regularized projections.
exclusions	vector of strings to exclude from predictors
inclusions	vector of strings to include in predictors
sparseness	vector or scalar value to set sparseness.
mixAlg	string 'svd', 'ica', 'rrpca-l', 'rrpca-s', 'stochastic', 'pca', 'newton-schulz' or 'avg' denotes the algorithm employed when estimating the mixed modality bases
iterations	int value to set max iterations
path_modeling	the result of a call to simlr_path_models(n)
sparsenessAlg	NA is default otherwise basic, spmp or orthorank
optimizationStyle	see list_simlr_optimizers
domainMatrices	matrices containing domain knowledge length of data_matrices with number of columns also equal to each corresponding data matrix
domainLambdas	weights for domain knowledge term length of data_matrices
verbose	boolean

Value

A list containing the results of the similarity analysis and related data.

Examples

```
# Example usage:
# result <- NNHEmbed(dataframe)
```

`NNHEmbed_update_residuals`*Update Residuals for SiMLR*

Description

This function updates residuals for SiMLR by applying different covariate transformations.

Usage

```
NNHEmbed_update_residuals(  
  mats,  
  x,  
  covariate,  
  blaster2,  
  allnna,  
  n.comp,  
  opt = NULL  
)
```

Arguments

<code>mats</code>	A list of matrices.
<code>x</code>	An index to select a matrix from the list.
<code>covariate</code>	A character string specifying the covariate transformation to apply.
<code>blaster2</code>	A data frame containing additional covariate data.
<code>allnna</code>	A logical vector indicating non-missing data points.
<code>n.comp</code>	An integer specifying the number of components.
<code>opt</code>	A character string. If set to "opt", the function will print available covariate options.

Value

The residual matrix after applying the specified covariate transformation. If 'opt' is set to "opt", the function will print available covariate options and return NULL.

Examples

```
## Not run:  
NNHEmbed_update_residuals(mats, 1, "whiten", blaster2, allnna, 5)  
NNHEmbed_update_residuals(opt = "opt")  
  
## End(Not run)
```

nsa_flow

*NSA-Flow Optimization***Description**

Performs optimization to balance fidelity to a target matrix and orthogonality of the solution matrix using a weighted objective function. The function supports multiple retraction methods and includes robust convergence checks. These constraints provide global control over column-wise sparseness by projecting the matrix onto the approximate Stiefel manifold.

Usage

```
nsa_flow(
  Y0,
  X0 = NULL,
  w = 0.5,
  retraction = c("soft_polar", "polar", "none"),
  max_iter = 500,
  tol = 1e-05,
  verbose = FALSE,
  seed = 42,
  apply_nonneg = TRUE,
  optimizer = "fast",
  initial_learning_rate = "brent",
  record_every = 1,
  window_size = 5,
  c1_armijo = 1e-06,
  simplified = FALSE,
  plot = FALSE
)
```

Arguments

<code>Y0</code>	Numeric matrix of size $p \times k$, the initial guess for the solution.
<code>X0</code>	Numeric matrix of size $p \times k$, the target matrix for fidelity. If <code>NULL</code> , initialized as $\text{pmax}(Y0, 0)$ with a small perturbation added to <code>Y0</code> .
<code>w</code>	Numeric scalar in $[0, 1]$, weighting the trade-off between fidelity ($1 - w$) and orthogonality (w). Default is 0.5.
<code>retraction</code>	Character string specifying the retraction method to enforce orthogonality constraints.
<code>max_iter</code>	Integer, maximum number of iterations. Default is 100.
<code>tol</code>	Numeric, tolerance for convergence based on relative gradient norm and energy stability. Default is $1e-6$.
<code>verbose</code>	Logical, if <code>TRUE</code> , prints iteration details. Default is <code>FALSE</code> .
<code>seed</code>	Integer, random seed for reproducibility. If <code>NULL</code> , no seed is set. Default is 42.

apply_nonneg	Logical, if TRUE, enforces non-negativity on the solution after retraction. Default is TRUE.
optimizer	Character string, optimization algorithm to use. The "fast" option will select the best option based on whether simplified = TRUE or FALSE. otherwise, pass the names of optimizers supported by create_optimizer() as seen in list_simlr_optimizers(). Default is "fast".
initial_learning_rate	Numeric, initial learning rate for the optimizer. Default is 1e-3 for non-neg and 1 for unconstrained. Otherwise, you can use estimate_learning_rate_nsa() to find a robust value. . pass a string one of c("brent", "grid", "armijo", "golden", "adaptive") to engage this method.
record_every	Integer, frequency of recording iteration metrics. Default is 1 (record every iteration).
window_size	Integer, size of the window for energy stability convergence check. Default is 5.
c1_armijo	Numeric, Armijo condition constant for line search.
simplified	Logical, if TRUE, uses the simplified objective

$$\min_U (1-w) \frac{1}{2} \|U - Z\|_F^2 + w \frac{1}{2} \|U^\top U - I_k\|_F^2$$

. If FALSE, uses the invariant defect objective. Default is FALSE.

plot	Logical, if TRUE, generates a ggplot of fidelity and orthogonality traces with dual axes. Default is FALSE.
------	---

Details

The function minimizes a weighted objective combining fidelity to X_0 and orthogonality of Y , defined as:

$$E(Y) = (1-w) * \|Y - X_0\|_F^2 / (2 * p * k) + w * defect(Y)$$

where $defect(Y)$ measures orthogonality deviation.

The optimization uses a Riemannian gradient descent approach with optional retraction to enforce orthogonality constraints. Convergence is checked via relative gradient norm and energy stability over a window of iterations.

Value

A list containing:

- Y : Numeric matrix, the best solution found (lowest total energy).
- `traces`: Data frame with columns `iter`, `time`, `fidelity`, `orthogonality`, and `total_energy` for recorded iterations.
- `final_iter`: Integer, number of iterations performed.
- `plot`: ggplot object of the optimization trace (if `plot = TRUE`), otherwise `NULL`.
- `best_total_energy`: Numeric, the lowest total energy achieved.

Examples

```
set.seed(123)
Y0 <- matrix(runif(20), 5, 4)
X0 <- matrix(runif(20), 5, 4)
# The original function relies on helper functions not shown here, such as:
# create_optimizer, step, inv_sqrt_sym, symm, and invariant_orthogonality_defect.
# The following example is conceptual:
# result <- nsa_flow(Y0, X0, w = 0.0, max_iter = 10, verbose = TRUE, plot = TRUE)
# print(result$plot)
# print(result$traces)
```

nsa_flow_autograd

NSA-Flow Optimization via PyTorch AutoGrad (R wrapper)

Description

Mirror of Python ‘nsa_flow_autograd()’ but callable from R via reticulate. Uses the Python implementation (nsa_flow.nsa_flow_autograd) and returns R-friendly results (matrix, data.frame, ggplot).

Usage

```
nsa_flow_autograd(
  Y0,
  X0 = NULL,
  w = 0.5,
  retraction = c("soft_polar", "polar", "none"),
  max_iter = 500L,
  tol = 1e-06,
  verbose = FALSE,
  seed = 42L,
  apply_nonneg = TRUE,
  optimizer = "asgd",
  initial_learning_rate = NULL,
  lr_strategy = "bayes",
  aggression = 0.5,
  fidelity_type = "scale_invariant",
  orth_type = "scale_invariant",
  record_every = 1L,
  window_size = 10L,
  warmup_iters = 0L,
  plot = FALSE,
  precision = "float64"
)
```

Arguments

Y0	numeric matrix p x k initial guess
X0	numeric matrix p x k target (or NULL to initialize from Y0)
w	numeric in [0,1] weighting fidelity vs orthogonality
retraction	character retraction method
max_iter	integer max iterations
tol	numeric convergence tolerance
verbose	logical
seed	integer random seed
apply_nonneg	logical or 'softplus'/'none' etc.
optimizer	character optimizer name (e.g. 'Adam','lars','sgdp')
initial_learning_rate	NULL (auto), numeric, or character strategy string
lr_strategy	character passed to Python if initial_learning_rate is NULL/'auto'
aggression	numeric controls aggressiveness of learning rate adaptation
fidelity_type	character ('basic','scale_invariant','symmetric','normalized')
orth_type	character ('basic','scale_invariant')
record_every	integer frequency of recording traces
window_size	integer window for energy stability
warmup_iters	integer number of warmup iterations
plot	logical produce ggplot (default FALSE)
precision	'float32' or 'float64'

Value

list: Y (matrix), traces (data.frame), final_iter, best_total_energy, best_Y_iteration, plot (ggplot or NULL), settings

nsa_flow_fa_diagram	<i>Generate NSA-Flow FA Flowchart Diagram</i>
---------------------	---

Description

This function creates a flowchart diagram representing the NSA-Flow FA process using the DiagrammeR package. The diagram illustrates the steps involved in the algorithm, including data input, regularization, iteration, convergence checks, and output.

Usage

nsa_flow_fa_diagram()

Details

The diagram is generated using Graphviz syntax via `DiagrammeR::grViz`. It visualizes the flow from input data through various processing steps to the final output. Ensure the `DiagrammeR` package is installed to use this function.

Value

A `DiagrammeR` graph object representing the flowchart.

Examples

```
## Not run:
library(DiagrammeR)
nsa_flow_fa_diagram()

## End(Not run)
```

<code>nsa_flow_flowchart</code>	<i>NSA-Flow Algorithm Flowchart</i>
---------------------------------	-------------------------------------

Description

Generates a `DiagrammeR` flowchart summarizing the full NSA-Flow optimization pipeline, including initialization, gradient computation, Riemannian projection, descent, retraction, non-negativity, and convergence checks.

Usage

```
nsa_flow_flowchart(
  node_fill_color = "lightblue",
  node_font_color = "black",
  edge_color = "black",
  font_name = "Helvetica",
  node_shape = "rectangle",
  graph_rankdir = "TB",
  fontsize = 10
)
```

Arguments

<code>node_fill_color</code>	Color for node fill (default: "lightblue").
<code>node_font_color</code>	Color for node text (default: "black").
<code>edge_color</code>	Color for edges (default: "black").
<code>font_name</code>	Font for node labels (default: "Helvetica").

node_shape	Node shape (default: "rectangle").
graph_rankdir	Graph layout direction (default: "TB" = top to bottom).
fontsize	Font size (default: 10).

Value

A DiagrammeR graph object.

nsa_flow_pca	<i>NSA-Flow Sparse PCA (stable + efficient)</i>
--------------	---

Description

Performs sparse PCA using a proximal-gradient scheme with optional NSA-flow proximal regularization. This implementation preserves the stable behavior of the original ‘sparse_pca_imp()’ while improving efficiency by: - precomputing $X^T X$, using it for gradient/energy, - in-place trial updates to reduce allocations, - retaining Armijo backtracking and adaptive LR scheduling.

Usage

```
nsa_flow_pca(
  X,
  k,
  lambda = 0.1,
  alpha = 1e-04,
  max_iter = 100,
  proximal_type = c("basic", "nsa_flow"),
  w_pca = 1,
  nsa_w = 0.5,
  apply_soft_thresh_in_nns = FALSE,
  tol = 1e-06,
  retraction = def_ret,
  grad_tol = 1e-04,
  nsa_flow_fn = nsa_flow_autograd,
  verbose = FALSE
)
```

Arguments

X	numeric matrix (n x p), rows = observations, cols = variables
k	integer > 0 and <= min(n,p), number of components
lambda	non-negative numeric, L1 proximal weight
alpha	positive numeric, initial step size
max_iter	integer, maximum iterations
proximal_type	character, "basic" or "nsa_flow"

w_pca	positive numeric, weight for PCA fidelity
nsa_w	numeric in [0,1], weight parameter passed to nsa_flow
apply_soft_thresh_in_nns	logical (passed through if used)
tol	numeric, tolerance for relative parameter change convergence
retraction	retraction function or identifier (passed to nsa_flow)
grad_tol	numeric, gradient-norm tolerance for convergence
nsa_flow_fn	optional, nsa_flow function to use (default: nsa_flow)
verbose	logical, print iteration diagnostics

Value

list with components:	
Y	best p x k loading matrix found
energy_trace	vector of per-iteration energy/explained-variance records
final_iter	last iteration index
best_energy	best energy attained
converged	logical
no_improve_count	internal counter at exit
lr_reductions	number of LR reductions performed
expl_var_ratio	explained variance ratio for best_Y

nsa_flow_pca_fa	<i>NSA-Flow Sparse PCA or FA</i>
-----------------	----------------------------------

Description

Optimize either a PCA objective (default) or a Factor Analysis (FA) objective in a single call. Armijo backtracking, optional NSA-flow proximal, adaptive LR scheduling, and optional rotation every iteration.

Usage

```
nsa_flow_pca_fa(  
  X,  
  k,  
  lambda = 0.1,  
  alpha = 0.01,  
  max_iter = 100,  
  proximal_type = c("basic", "nsa_flow"),  
  w_pca = 1,  
  ...  
)
```

```

    tol = 1e-06,
    grad_tol = 1e-04,
    nsa_flow_fn = NULL,
    verbose = FALSE,
    objective = c("pca", "fa"),
    rotate = c("none", "varimax", "promax", "oblimin"),
    nsa_flow_args = list()
)

```

Arguments

X	numeric matrix (n x p), rows = observations, cols = variables
k	integer > 0 and <= min(n,p), number of components
lambda	non-negative numeric, L1 proximal weight
alpha	positive numeric, initial step size
max_iter	integer, maximum iterations
proximal_type	character, "basic" or "nsa_flow"
w_pca	positive numeric, weight for PCA fidelity
tol	numeric, tolerance for relative parameter change convergence
grad_tol	numeric, gradient-norm tolerance for convergence
nsa_flow_fn	optional, nsa_flow function to use (default: nsa_flow)
verbose	logical, print iteration diagnostics
objective	character, either "pca" (default) or "fa". If "fa", the function optimizes a FA-like loss: reconstructing the sample covariance R with $Y Y^T + \text{diag}(\psi)$ where $\psi \geq 0$ is set to the diagonal of $R - Y Y^T$ (clipped to a small positive eps).
rotate	character; one of "none", "varimax", "promax", "oblimin". If not "none", the loadings are rotated every iteration (keeps shapes).
nsa_flow_args	list, optional additional args forwarded to nsa_flow_fn()

Value

list with components: Y, energy_trace, final_iter, best_energy, converged, no_improve_count, lr_reductions, expl_var_ratio

nsa_flow_pca_fa_2	<i>Perform Proximal Gradient Optimization for PCA or Factor Analysis</i>
-------------------	--

Description

This function implements a proximal gradient descent algorithm for either Principal Component Analysis (PCA) or Factor Analysis (FA), with support for sparsity via L1 regularization or custom proximal operators (e.g., NSA flow). For PCA, it maximizes explained variance; for FA, it minimizes the least-squares reconstruction error of the covariance matrix using an alternating optimization over loadings and uniquenesses. Initialization uses SVD, and optional rotation is applied at the end.

Usage

```
nsa_flow_pca_fa_2(
  X,
  k,
  lambda = 0.1,
  alpha = 0.01,
  max_iter = 100,
  proximal_type = c("basic", "nsa_flow"),
  w_pca = 1,
  tol = 1e-06,
  grad_tol = 1e-04,
  nsa_flow_fn = NULL,
  verbose = FALSE,
  objective = c("pca", "fa"),
  rotate = c("none", "varimax", "promax", "oblimin"),
  nsa_flow_args = list()
)
```

Arguments

X	Numeric matrix of data (n rows observations, p columns variables).
k	Integer number of components/factors ($1 \leq k \leq \min(n, p)$).
lambda	Non-negative regularization parameter for L1 sparsity (default 0.1).
alpha	Positive initial step size for gradient descent (default 0.01).
max_iter	Maximum number of iterations (default 100).
proximal_type	Type of proximal operator: "basic" for soft-thresholding or "nsa_flow" for custom (default "basic").
w_pca	Positive weight for PCA objective (default 1.0).
tol	Tolerance for convergence based on relative change (default 1e-6).
grad_tol	Tolerance for gradient norm (default 1e-4).
nsa_flow_fn	Function for NSA proximal operator (required if proximal_type = "nsa_flow").
verbose	Logical; if TRUE, print iteration progress (default FALSE).
objective	Objective: "pca" or "fa" (default "pca").
rotate	Rotation method: "none", "varimax", "promax", or "oblimin" (default "none").
nsa_flow_args	List of additional arguments for nsa_flow_fn (default list()).

Value

A list containing:

- Y: Optimized loadings matrix (unrotated).
- loadings: Final loadings (possibly rotated).
- communalities: Vector of communalities for each variable.
- energy_trace: Vector of energy values per iteration.

- `final_iter`: Number of iterations performed.
- `best_energy`: Best (lowest) energy achieved.
- `converged`: Logical indicating convergence.
- `no_improve_count`: Count of iterations without improvement.
- `lr_reductions`: Number of learning rate reductions.
- `expl_var_ratio`: Explained variance ratio (PCA-style).

Examples

```
## Not run:
data(iris)
X <- as.matrix(iris[, 1:4])
res <- nsa_flow_pca_fa_2(X, k = 2, objective = "fa", verbose = TRUE)
print(res$loadings)

## End(Not run)
```

<code>nsa_flow_retract</code>	<i>Retraction onto the Stiefel Manifold</i>
-------------------------------	---

Description

Retraction onto the Stiefel Manifold

Usage

```
nsa_flow_retract(Y_cand, w_retract, retraction_type)
```

Arguments

<code>Y_cand</code>	Candidate matrix.
<code>w_retract</code>	Weight for soft retraction.
<code>retraction_type</code>	Type of retraction ('polar', 'soft_polar', or 'none').

nsa_flow_retract_auto	<i>Retraction onto the Stiefel Manifold (soft-polar multiplicative, with adaptive inv-sqrt)</i>
-----------------------	---

Description

Retraction onto the Stiefel Manifold (soft-polar multiplicative, with adaptive inv-sqrt)

Usage

```
nsa_flow_retract_auto(  
    Y_cand,  
    w_retract = 1,  
    retraction_type = c("polar", "soft_polar", "none"),  
    eps_rf = 1e-08,  
    inv_method = "diag",  
    ns_iter = 1L,  
    eig_thresh = 128L,  
    diag_thresh = 8192L,  
    verbose = FALSE  
)
```

Arguments

Y_cand	Candidate matrix.
w_retract	Weight for soft retraction.
retraction_type	Type of retraction ('polar', 'soft_polar', or 'none').
eps_rf	Small numeric value for stability.
inv_method	Method for inverse square root.
ns_iter	Number of iterations for Newton-Schulz.
eig_thresh	Threshold for eigen-decomposition.
diag_thresh	Threshold for diagonal check.
verbose	Verbose output.

nsa_flow_torch	<i>NSA-Flow Optimization via PyTorch</i>
----------------	--

Description

Performs optimization to balance fidelity to a target matrix and orthogonality of the solution matrix using a weighted objective function. The function supports multiple retraction methods and includes robust convergence checks. These constraints provide global control over column-wise sparseness by projecting the matrix onto the approximate Stiefel manifold.

Usage

```

nsa_flow_torch(
  Y0,
  X0 = NULL,
  w = 0.5,
  retraction = c("soft_polar", "polar", "none"),
  max_iter = 500,
  tol = 1e-05,
  verbose = FALSE,
  seed = 42,
  apply_nonneg = TRUE,
  optimizer = "fast",
  initial_learning_rate = "default",
  record_every = 1,
  window_size = 5,
  cl_armijo = 1e-06,
  simplified = FALSE,
  project_full_gradient = FALSE,
  plot = FALSE,
  precision = "float64"
)

```

Arguments

<code>Y0</code>	Numeric matrix of size $p \times k$, the initial guess for the solution.
<code>X0</code>	Numeric matrix of size $p \times k$, the target matrix for fidelity. If NULL, initialized as $\text{pmax}(Y0, 0)$ with a small perturbation added to <code>Y0</code> .
<code>w</code>	Numeric scalar in $[0, 1]$, weighting the trade-off between fidelity ($1 - w$) and orthogonality (w). Default is 0.5.
<code>retraction</code>	Character string specifying the retraction method to enforce orthogonality constraints.
<code>max_iter</code>	Integer, maximum number of iterations. Default is 100.
<code>tol</code>	Numeric, tolerance for convergence based on relative gradient norm and energy stability. Default is $1e-6$.
<code>verbose</code>	Logical, if TRUE, prints iteration details. Default is FALSE.
<code>seed</code>	Integer, random seed for reproducibility. If NULL, no seed is set. Default is 42.
<code>apply_nonneg</code>	Logical, if TRUE, enforces non-negativity on the solution after retraction. Default is TRUE.
<code>optimizer</code>	Character string, optimization algorithm to use. The "fast" option will select the best option based on whether <code>simplified = TRUE</code> or FALSE. otherwise, pass the names of optimizers supported by <code>create_optimizer()</code> as seen in <code>list_simlr_optimizers()</code> . Default is "fast".
<code>initial_learning_rate</code>	Numeric, initial learning rate for the optimizer. Default is $1e-3$ for non-neg and 1 for unconstrained. Otherwise, you can use <code>estimate_learning_rate_nsa()</code>

	to find a robust value. . pass a string one of c("brent", "grid", "armijo", "golden", "adaptive") to engage this method.
record_every	Integer, frequency of recording iteration metrics. Default is 1 (record every iteration).
window_size	Integer, size of the window for energy stability convergence check. Default is 5.
c1_armijo	Numeric, Armijo condition constant for line search.
simplified	Logical, if TRUE, uses the simplified objective

$$\min_U (1-w) \frac{1}{2} \|U - Z\|_F^2 + w \frac{1}{2} \|U^\top U - I_k\|_F^2$$

. If FALSE, uses the invariant defect objective. Default is FALSE.

project_full_gradient	Logical, if TRUE, projects the full gradient instead of just the orthogonal component. Default is FALSE.
plot	Logical, if TRUE, generates a ggplot of fidelity and orthogonality traces with dual axes. Default is FALSE.
precision	Character string, either 'float32' or 'float64' to set the precision

Details

The function minimizes a weighted objective combining fidelity to X_0 and orthogonality of Y , defined as:

$$E(Y) = (1-w) * \|Y - X_0\|_F^2 / (2 * p * k) + w * defect(Y)$$

where $defect(Y)$ measures orthogonality deviation.

The optimization uses a Riemannian gradient descent approach with optional retraction to enforce orthogonality constraints. Convergence is checked via relative gradient norm and energy stability over a window of iterations.

Value

A list containing:

- Y : Numeric matrix, the best solution found (lowest total energy).
- `traces`: Data frame with columns `iter`, `time`, `fidelity`, `orthogonality`, and `total_energy` for recorded iterations.
- `final_iter`: Integer, number of iterations performed.
- `plot`: ggplot object of the optimization trace (if `plot = TRUE`), otherwise `NULL`.
- `best_total_energy`: Numeric, the lowest total energy achieved.

Examples

```
set.seed(123)
Y0 <- matrix(runif(20), 5, 4)
X0 <- matrix(runif(20), 5, 4)
# The original function relies on helper functions not shown here, such as:
# create_optimizer, step, inv_sqrt_sym, symm, and invariant_orthogonality_defect.
```

```
# The following example is conceptual:
# result <- nsa_flow_torch(Y0, X0, w = 0.0, max_iter = 10, verbose = TRUE, plot = TRUE)
# print(result$plot)
# print(result$traces)
```

nsa_flow_torch_ag

NSA-Flow Optimization via PyTorch AutoGrad

Description

Performs optimization to balance fidelity to a target matrix and orthogonality of the solution matrix using a weighted objective function. The function supports multiple retraction methods and includes robust convergence checks. These constraints provide global control over column-wise sparseness by projecting the matrix onto the approximate Stiefel manifold.

Usage

```
nsa_flow_torch_ag(
  Y0,
  X0 = NULL,
  w = 0.5,
  retraction = c("soft_polar", "polar", "none"),
  max_iter = 500,
  tol = 1e-05,
  verbose = FALSE,
  seed = 42,
  apply_nonneg = TRUE,
  optimizer = "fast",
  initial_learning_rate = "default",
  record_every = 1,
  window_size = 5,
  c1_armijo = 1e-06,
  simplified = FALSE,
  project_full_gradient = FALSE,
  plot = FALSE,
  precision = "float64"
)
```

Arguments

<code>Y0</code>	Numeric matrix of size $p \times k$, the initial guess for the solution.
<code>X0</code>	Numeric matrix of size $p \times k$, the target matrix for fidelity. If <code>NULL</code> , initialized as $\text{pmax}(Y0, 0)$ with a small perturbation added to <code>Y0</code> .
<code>w</code>	Numeric scalar in $[0, 1]$, weighting the trade-off between fidelity ($1 - w$) and orthogonality (w). Default is 0.5.

retraction	Character string specifying the retraction method to enforce orthogonality constraints.
max_iter	Integer, maximum number of iterations. Default is 100.
tol	Numeric, tolerance for convergence based on relative gradient norm and energy stability. Default is 1e-6.
verbose	Logical, if TRUE, prints iteration details. Default is FALSE.
seed	Integer, random seed for reproducibility. If NULL, no seed is set. Default is 42.
apply_nonneg	Logical, if TRUE, enforces non-negativity on the solution after retraction. Default is TRUE.
optimizer	Character string, optimization algorithm to use. The "fast" option will select the best option based on whether simplified = TRUE or FALSE. otherwise, pass the names of optimizers supported by create_optimizer() as seen in list_simlr_optimizers(). Default is "fast".
initial_learning_rate	Numeric, initial learning rate for the optimizer. Default is 1e-3 for non-neg and 1 for unconstrained. Otherwise, you can use estimate_learning_rate_nsa() to find a robust value. . pass a string one of c("brent", "grid", "armijo", "golden", "adaptive") to engage this method.
record_every	Integer, frequency of recording iteration metrics. Default is 1 (record every iteration).
window_size	Integer, size of the window for energy stability convergence check. Default is 5.
c1_armijo	Numeric, Armijo condition constant for line search.
simplified	Logical, if TRUE, uses the simplified objective

$$\min_U (1-w) \frac{1}{2} \|U - Z\|_F^2 + w \frac{1}{2} \|U^\top U - I_k\|_F^2$$

. If FALSE, uses the invariant defect objective. Default is FALSE.

project_full_gradient	Logical, if TRUE, projects the full gradient instead of just the orthogonal component. Default is FALSE.
plot	Logical, if TRUE, generates a ggplot of fidelity and orthogonality traces with dual axes. Default is FALSE.
precision	Character string, either 'float32' or 'float64' to set the precision

Details

The function minimizes a weighted objective combining fidelity to X_0 and orthogonality of Y , defined as:

$$E(Y) = (1-w) * \|Y - X_0\|_F^2 / (2 * p * k) + w * defect(Y)$$

where $defect(Y)$ measures orthogonality deviation.

The optimization uses a Riemannian gradient descent approach with optional retraction to enforce orthogonality constraints. Convergence is checked via relative gradient norm and energy stability over a window of iterations.

Value

A list containing:

- `Y`: Numeric matrix, the best solution found (lowest total energy).
- `traces`: Data frame with columns `iter`, `time`, `fidelity`, `orthogonality`, and `total_energy` for recorded iterations.
- `final_iter`: Integer, number of iterations performed.
- `plot`: ggplot object of the optimization trace (if `plot = TRUE`), otherwise `NULL`.
- `best_total_energy`: Numeric, the lowest total energy achieved.

Examples

```
set.seed(123)
Y0 <- matrix(runif(20), 5, 4)
X0 <- matrix(runif(20), 5, 4)
# The original function relies on helper functions not shown here, such as:
# create_optimizer, step, inv_sqrt_sym, symm, and invariant_orthogonality_defect.
# The following example is conceptual:
# result <- nsa_flow_torch(Y0, X0, w = 0.0, max_iter = 10, verbose = TRUE, plot = TRUE)
# print(result$plot)
# print(result$traces)
```

oneHotToSegmentation *Inverse one-hot transformation*

Description

Inverse of basic one-hot transformation of segmentations array

Usage

```
oneHotToSegmentation(oneHotArray, domainImage, channelFirstOrdering = FALSE)
```

Arguments

`oneHotArray` an array where the channel dimension contains the one-hot encoding.

`domainImage` image defining the geometry of the returned probability images.

`channelFirstOrdering`
 Specifies the ordering of the dimensions.

Value

a list of probability images.

Author(s)

Tustison NJ

Examples

```

r16 <- antsImageRead( getANTsRData( "r16" ) )
seg <- kmeansSegmentation( r16, 3 )$segmentation
oneHot <- segmentationToOneHot( as.array( seg ) )
oneHotInv <- oneHotToSegmentation( oneHot, seg )

r16 <- antsImageRead( getANTsRData( "r16" ) )
seg <- kmeansSegmentation( r16, 3 )$segmentation
oneHot <- segmentationToOneHot( as.array( seg ),
                                channelFirstOrdering = TRUE )
oneHotInv <- oneHotToSegmentation( oneHot, seg,
                                   channelFirstOrdering = TRUE )

```

optimal_simlr_initializer

Calculate SiMLR energy for optimization Find Optimal SiMLR Initializer

Description

This function generates multiple random candidate U matrices, evaluates their associated energy (without gradient descent), and returns the best-scoring initialization. A seed parameter ensures reproducibility across runs.

This function generates multiple random candidate U matrices, evaluates their associated energy (without gradient descent), and returns the best-scoring initialization. A seed parameter ensures reproducibility across runs.

Usage

```

optimal_simlr_initializer(
  data_matrices,
  n_init = 10,
  basisK,
  energyType = "acc",
  domainMatrices = NULL,
  domainLambdas = NULL,
  verbose = TRUE,
  seed = NULL
)

optimal_simlr_initializer(
  data_matrices,

```



```

    n_init = 10,
    basisK,
    energyType = "acc",
    domainMatrices = NULL,
    domainLambdas = NULL,
    verbose = TRUE,
    seed = NULL
)

```

Arguments

<code>data_matrices</code>	A list of modality-specific data matrices.
<code>n_init</code>	Number of random initializations to try (default = 10).
<code>basisK</code>	Number of basis components (columns in U).
<code>energyType</code>	Energy function to evaluate.
<code>domainMatrices</code>	Optional list of domain priors (same length as <code>data_matrices</code>).
<code>domainLambdas</code>	Optional vector of domain weights.
<code>verbose</code>	Logical, whether to print progress.
<code>seed</code>	Optional numeric seed for reproducibility (default = NULL, no seed set).

Value

A list with elements:

bestU List of U matrices (one per modality) for the best initialization.

bestV List of corresponding V matrices.

bestEnergy Mean energy for the selected initialization.

A list with elements:

bestU List of U matrices (one per modality) for the best initialization.

bestV List of corresponding V matrices.

bestEnergy Mean energy for the selected initialization.

optimize_indicator_matrix

Optimize Binary Indicator Matrix with Row Uniformity

Description

This function optimizes the sum of the matrix $'m * I'$, where $'I'$ is a binary indicator matrix with the constraint that each column may have only one non-zero entry. It ensures that the distribution of 1's is uniform across rows, softens the constraint to avoid infinite loops, and includes an optional verbose output to report the progress of the optimization.

Usage

```
optimize_indicator_matrix(  
  m,  
  max_iter = 1000,  
  tol = 1e-06,  
  preprocess = TRUE,  
  verbose = FALSE  
)
```

Arguments

m	A numeric matrix to optimize.
max_iter	The maximum number of iterations to avoid infinite loops. Default is 1000.
tol	A numeric value representing the tolerance for convergence. If the change in the objective function is less than this value, the loop stops. Default is 1e-6.
preprocess	Logical. If TRUE, flips the sign of each row where the entries are predominantly negative.
verbose	Logical. If TRUE, reports the objective value and convergence progress at each iteration.

Value

‘m * I’

Examples

```
set.seed(123)  
m <- matrix(rnorm(500), nrow = 5)  
result <- optimize_indicator_matrix(m, max_iter = 1000, tol = 1e-6, verbose = TRUE)  
print(result)
```

orth_residual	<i>Orthogonality Residual</i>
---------------	-------------------------------

Description

Measures the scale-invariant orthogonality defect of a matrix, assessing deviation from orthonormality.

Usage

```
orth_residual(Y)
```

Arguments

Y	Numeric matrix (p x k) to evaluate.
---	-------------------------------------

Details

The residual is computed as $\sqrt{\sum_{i \neq j} ((Y^T Y / \|Y\|_F^2)_{ij})^2}$, where $Y^T Y / \|Y\|_F^2$ is the normalized Gram matrix. This measures how far $Y^T Y$ is from a scaled identity matrix, ignoring diagonal terms.

Value

Numeric scalar, the orthogonality residual.

Examples

```
Y <- matrix(runif(10*5), 10, 5)
orth_residual(Y)
```

orthogonalize_feature_space

Orthogonalize a List of Matrices via Gradient Descent with Convergence Check

Description

This function takes a list of matrices (with the same number of columns) and iteratively updates them to make the columns within each matrix more orthogonal. It stops when the change in the total orthogonality error is below a given tolerance or when the maximum number of iterations is reached.

Usage

```
orthogonalize_feature_space(
  matrix_list,
  max_iterations = 100,
  learning_rate = 0.05,
  tolerance = 1e-04,
  verbose = TRUE
)
```

Arguments

<code>matrix_list</code>	A list of numeric matrices. All matrices must have the same number of columns (e.g., k components).
<code>max_iterations</code>	The maximum number of gradient descent iterations to perform.
<code>learning_rate</code>	The step size for each update.
<code>tolerance</code>	The convergence threshold. The algorithm stops if the absolute change in the total orthogonality defect between iterations is less than this value.
<code>verbose</code>	If TRUE, prints the orthogonality error at each iteration.

Value

A list of orthogonalized matrices with the same dimensions as the input.

orthogonalizeAndQSparsify

Sparsify and optionally orthogonalize a matrix

Description

This function implements a quantile-based sparsification operation.

Usage

```
orthogonalizeAndQSparsify(
  v,
  sparsenessQuantile = 0.5,
  positivity = "either",
  orthogonalize = TRUE,
  softThresholding = FALSE,
  unitNorm = FALSE,
  sparsenessAlg = NA
)
```

Arguments

<code>v</code>	Input matrix
<code>sparsenessQuantile</code>	Quantile to control sparseness - higher is sparser
<code>positivity</code>	Restrict to positive or negative solution (beta) weights. Choices are "positive", "negative", or "either".
<code>orthogonalize</code>	Run Gram-Schmidt if TRUE.
<code>softThresholding</code>	Use soft thresholding if TRUE.
<code>unitNorm</code>	Normalize each vector to unit norm if TRUE.
<code>sparsenessAlg</code>	If specified, use a matrix partition algorithm ("orthorank", "smp", "sum_preserving_matrix_partition" or "basic").

Value

A sparsified and optionally orthogonalized matrix.

Examples

```
mat <- replicate(100, rnorm(20))
mat <- orthogonalizeAndQSparsify(mat)
```

```
orthogonalizeAndQSparsifyOld  
  sparsify a matrix
```

Description

This implements a quantile based sparsification operation

Usage

```
orthogonalizeAndQSparsifyOld(  
  v,  
  sparsenessQuantile = 0.5,  
  positivity = "either",  
  orthogonalize = TRUE,  
  softThresholding = FALSE,  
  unitNorm = FALSE,  
  sparsenessAlg = NA  
)
```

Arguments

v	input matrix
sparsenessQuantile	quantile to control sparseness - higher is sparser
positivity	restrict to positive or negative solution (beta) weights. choices are positive, negative or either as expressed as a string.
orthogonalize	run gram-schmidt if TRUE.
softThresholding	use soft thresholding
unitNorm	set each vector to unit norm
sparsenessAlg	NA is default otherwise basic, spmp or orthorank

Value

matrix

Author(s)

Avants BB

Examples

```
mat <- replicate(100, rnorm(20))  
mat <- orthogonalizeAndQSparsify(mat)
```

padImageByFactor	<i>Pad an image based on a factor.</i>
------------------	--

Description

Pad image of size (x, y, z) to (x', y', z') where (x', y', z') is a divisible by a user-specified factor.

Usage

```
padImageByFactor(image, factor)
```

Arguments

image	Input ANTs image
factor	padding factor. Can be an integer or vector of size equal to the image dimensionality.

Value

a padded image

Author(s)

Tustison NJ, Avants BB

Examples

```
library( ANTsR )
image <- antsImageRead( getANTSRData( "r16" ) )
image <- cropImage( image )
paddedImage <- padImageByFactor( image, 4 )
```

padOrCropImageToSize	<i>Pad or crop image to a specified size</i>
----------------------	--

Description

Pad or crop image to a specified size

Usage

```
padOrCropImageToSize(image, size)
```

Arguments

image	Input ANTs image
size	size of the output image.

Value

a padded/cropped image

Author(s)

Tustison NJ

Examples

```
library( ANTsR )

image <- antsImageRead( getANTSRData( "r16" ) )
paddedImage <- padOrCropImageToSize( image, c( 333, 333 ) )
```

pairwise_matrix_similarity

pairwise application of matrix similarity to matrix List projected onto a feature list

Description

Computes a measure such as the RV coefficient between low-rank projections of all pairs of matrices in a list.

Usage

```
pairwise_matrix_similarity(mat_list, feat_list, FUN = adjusted_rvcoef)
```

Arguments

mat_list	List of matrices
feat_list	List of feature lists corresponding to each matrix
FUN	function to apply

Value

Matrix of RV coefficients, where each entry [i, j] represents the similarity between the low-rank projections of matrices i and j

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

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

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

Shrinidhi KL Avants BB

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

petSUVR *Compute SUVr from PET time series images*

The standardized uptake value (SUV) ratio (SUVR) is a semi-quantative transformation that is performed on a PET image before interpretation. This function will run rigid motion correction on the PET time series before computing SUVR. The implementation also requires an anatomical image and a segmentation that defines the reference region. This function will produce a mask, a motion corrected image and a PET suvr image mapped to the reference anatomical space.

```
petSUVR(  
  petTime,  
  anatomicalImage,  
  anatomicalSegmentation,  
  smoothingParameter = 2.5,  
  labelValue = 0,  
  subtractBackground = FALSE,  
  mapToPet = TRUE,  
  noMotionCorr = FALSE  
)
```

petTime	pet time series antsImage or 3D image
---------	---------------------------------------

anatomicalImage	antsImage that indicates the target anatomical image. one should choose this image carefully, as it will impact results. one might compare results with, for example, either whole head or whole brain targets.
anatomicalSegmentation	antsImage in the same space as anatomicalImage. This image should contain a whole brain mask in addition to a labeled reference region for computing SUVR.
smoothingParameter	physical space smoothing parameter see smoothImage
labelValue	the integer value of the reference region as it appears in anatomicalSegmentation. If not set, we assume a binary segmentation.
subtractBackground	if TRUE, the method will attempt to estimate the background (non-anatomical) activation levels before computing SUVR. In this case, SUVR will be computed after subtracting this background value from the mean activation image.
mapToPet	boolean option causing pet to be used as fixed image in the rigid registration between anatomical and pet
noMotionCorr	boolean option activating simple and fast approach that may be better for dynamic data with insufficient per-frame information

Value

suvr antsImage

Author(s)

Avants BB

Examples

```
## Not run:
suvr <- petSUVR(petTime, seg)

## End(Not run)
```

pixeltype

Extract Image Attributes pixeltype from antsImages

Description

Methods that act on the slots of the antsImage object

Usage

```
pixeltype(object)

## S4 method for signature 'antsImage'
pixeltype(object)

pixeltype(object) <- value

## S4 replacement method for signature 'antsImage'
pixeltype(object) <- value

components(object)

## S4 method for signature 'antsImage'
components(object)

components(object) <- value

## S4 replacement method for signature 'antsImage'
components(object) <- value

spacing(object)

## S4 method for signature 'antsImage'
spacing(object)

spacing(object) <- value

## S4 replacement method for signature 'antsImage'
spacing(object) <- value

direction(object)

## S4 method for signature 'antsImage'
direction(object)

direction(object) <- value

## S4 replacement method for signature 'antsImage'
direction(object) <- value

origin(object)

## S4 method for signature 'antsImage'
origin(object)

origin(object) <- value
```

```
## S4 replacement method for signature 'antsImage'
origin(object) <- value
```

Arguments

object	is an object of class antsImage.
value	is the value to assign to the slot.

Examples

```
img01 <- as.antsImage(array(1:64, c(4, 4, 4, 1)))
stopifnot(is.character(pixeltype(img01)))
pixeltype(img01) <- "float"
testthat::expect_error({
  pixeltype(img01) <- 1.5
})
img01 <- as.antsImage(array(1:64, c(4, 4, 4, 1)))
stopifnot(components(img01) == 1)
components(img01) <- 1L
testthat::expect_error({
  components(img01) <- 1.5
})
img01 <- as.antsImage(array(1:64, c(4, 4, 4, 1)))
stopifnot(all(spacing(img01) == rep(1, 4)))
spacing(img01) <- rep(1, 4)
img01 <- as.antsImage(array(1:64, c(4, 4, 4, 1)))
stopifnot(is.matrix(direction(img01)))
direction(img01) <- diag(4)
img01 <- as.antsImage(array(1:64, c(4, 4, 4, 1)))
stopifnot(all(origin(img01) == rep(0, 4)))
origin(img01) <- rep(0, 4)
```

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 = NULL,
  color.img = "white",
  color.overlay = c("jet", "red", "blue", "green", "yellow", "viridis", "magma",
    "plasma", "inferno"),
```

```

axis = 2,
slices,
colorbar = FALSE,
title.colorbar,
title.img,
title.line = NA,
color.colorbar,
window.img,
window.overlay,
quality = 2,
outname = NA,
alpha = 1,
direction = 1,
begin = 0,
end = 1,
newwindow = FALSE,
nslices = 10,
domainImageMap = NULL,
ncolumns = 4,
useAbsoluteScale = FALSE,
doCropping = TRUE,
text,
...
)

```

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 (not displayed when file is saved with outname)
title.line	title vertical displacement (irrelevant when a file is saved with outname)
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. if you set window.overlay = c(0, 100) and useAbsoluteScale=T then the image overlay will map into that range. otherwise, the min and max of the image will be used.
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
direction	Sets the order of colors in the scale. If 1, the default, colors are ordered from darkest to lightest. If -1, the order of colors is reversed.
begin	The (corrected) hue in [0,1] at which the viridis colormap begins.
end	The (corrected) hue in [0,1] at which the viridis colormap ends
newwindow	boolean controlling if we open a new device for this plot
nslices	number of slices to view
domainImageMap	this input antsImage or list contains a reference image (domainImage) and optional reference mapping named domainMap. these will be used to map the input image(s) to this antsImage space before plotting. this is useful for non-standard image orientations.
ncolumns	number of columns in plot
useAbsoluteScale	boolean determines whether dynamic range is maximized when visualizing overlays
doCropping	apply cropping, defaults to TRUE
text	vector containing x, y, label, cex and color values passed to text command, e.g. txt=list(x=0,y=0,label='my text',cex=2,col='red')
...	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))
txt <- list(x = 2.5, y = 1.5, label = "my text", cex = 4, col = "red")
plot(img, list(mask), text = txt)
testthat::expect_error(plot(multi_component_image))

## Not run:
```



```

img <- antsImageRead(getANTSRData("r16"))
betaVals <- rnorm(prod(dim(img)), 0, 20)
betaImg <- makeImage(dim(img), betaVals) %>% smoothImage(3.5)
betaImg[abs(betaImg) < 1.5] <- 0
plot(img, betaImg, window.img = range(img), window.overlay = range(betaImg))

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,
      color.overlay = c("red", "blue"), outname = ofn
)

## End(Not run)

```

plot_energy_decomposition

Plot Energy Decomposition During SIMLR Optimization

Description

Creates line plots showing the evolution of similarity, domain, and total energy across optimization iterations, for each modality. Energies are rescaled to [0,1] within each type for comparability.

Usage

```
plot_energy_decomposition(simlr_result, modality, show_weights = FALSE)
```

```
plot_energy_decomposition(simlr_result, modality, show_weights = FALSE)
```

Arguments

simlr_result	A list returned by simlr(), containing energyPath, sim_energy, domain_energy, and total_energy.
modality	Character string giving the name of the modality to plot (must match names in energyPath\$modality).
show_weights	Logical, if TRUE also show weighted energies.

Value

A ggplot2 object.

A ggplot2 object.

Examples

```
## Not run:
res <- simlr(...)
plot_energy_decomposition(res, modality = "RNA")

## End(Not run)
## Not run:
res <- simlr(...)
plot_energy_decomposition(res, modality = "RNA")

## End(Not run)
```

plot_features

Plot Features

Description

Create bar plots for each column in each data frame, showing only non-zero values. Normalize each feature s.t. max is one.

Usage

```
plot_features(data_list, take_abs = TRUE, n_limit = 12)
```

Arguments

data_list	A list of data frames.
take_abs	boolean
n_limit	Integer, limit features to top n_limit with highest value

Value

A list of ggplot objects.

Examples

```
## Not run:
# Simulate data
set.seed(123)
feature_names <- paste("Feature", 1:10)
data_list <- list(
  df1 = data.frame(matrix(ifelse(runif(100) < 0.5, 0, runif(100)), nrow = 10)),
  df2 = data.frame(matrix(ifelse(runif(100) < 0.5, 0, runif(100)), nrow = 10))
)
```

```

)

# Set rownames
rownames(data_list$df1) <- feature_names
rownames(data_list$df2) <- feature_names

# Use the function
plots <- plot_features(data_list)

# Display the plots
for (i in 1:length(plots)) {
  print(plots[[i]])
}

## 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 = -1,
  backgroundColor = "white",
  nodecolors = "blue",
  nodetype = "s",
  edgeContrast = c(0, 1),
  quantileTransformWeights = FALSE,
  lwd = 2,
  minRadius = 0,
  maxRadius = 3,
  radius = NA,
  showOnlyConnectedNodes = TRUE
)

```

Arguments

centroids	input matrix of size N 3D points (in rows) by 3 (in columns), for N nodes.
brain	input rendering object which is output of renderSurfaceFunction. or a function derived from renderSurfaceFunction.

weights	edge weights, a symmetric matrix of size N. Weights should be non-negative.
edgecolors	a color(map) for edges. If a color map function, weights will be transformed to the range [0,1], which is compatible with functions returned by colorRamp.
backgroundColor	background color.
nodecolors	a color or color vector for nodes.
nodetype	sphere or other node type supported by RGL.
edgeContrast	a vector of length 2, specifying the contrast range for edge colors. Weights are normalized to the range [0,1]. The normalized weights can be rescaled with this parameter, eg <code>c(0.05, 0.95)</code> would stretch the contrast over the middle 90% of normalized weight values.
quantileTransformWeights	quantile transform the weights.
lwd	line width for drawing edges.
minRadius	minimum node radius. Ignored if the radius is specified explicitly with radius.
maxRadius	maximum node radius. The node radius between minRadius and maxRadius is determined from the sum of the edge weights connecting the node. Ignored if the radius is specified explicitly with radius.
radius	a constant radius or vector of length <code>nrow(centroids)</code> . If not specified, node radius is determined by the sum of edge weights connected to the node.
showOnlyConnectedNodes	boolean, if TRUE, only nodes with non-zero edge weights are plotted.

Details

If edgecolors is not specified, a heat-like color palette is used. Weights can be quantile transformed or clipped at a given quantile in order to improve contrast.

If weights are not specified, only the nodes are plotted.

Value

None

Author(s)

Avants BB, Duda JT, Cook PA

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")
```

```

mnit <- thresholdImage(mnit, 1, 2)
cnt <- getCentroids(mnia, clustparam = 0)
aalcnt <- cnt[1:90, 1:3]
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, edgecolors = "red")
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)
# another example
mni <- getANTsRData("mni")
mni <- antsImageRead(mni)
mnit <- thresholdImage(mni, 1, max(mni))
mnit <- iMath(mnit, "FillHoles")
mniseg <- kmeansSegmentation(mni, 3)$segmentation
wmbkgd <- thresholdImage(mniseg, 3, 3) %>%
  iMath("GetLargestComponent") %>%
  iMath("FillHoles")
wmbkgd <- smoothImage(iMath(wmbkgd, "MD", 1), 2.0)
brain <- renderSurfaceFunction(
  surfimg = list(wmbkgd),
  alphasurf = 0.8, smoothsval = 1.0
)
data(powers_areal_mni_itk)
coords <- powers_areal_mni_itk[, 1:3]
id <- rgl::par3d("userMatrix")
rid <- rotate3d(id, -pi / 2, 1, 0, 0)
rid2 <- rotate3d(id, pi / 2, 0, 0, 1)
rid3 <- rotate3d(id, -pi / 2, 0, 0, 1)
rgl::par3d(userMatrix = id)
handMat2 <- t(matrix(c(
  -0.9998656511, 0.01626961, 0.00198165,
  0, -0.0163816363, -0.99584705, -0.08955579, 0, 0.0005163439,
  -0.08957647, 0.99597979, 0, 0.0000000000, 0.00000000,
  0.00000000, 1
), ncol = 4))
loccolor <- as.character(powers_areal_mni_itk$Color)
loccolor[loccolor == "Peach"] <- "sienna1"
loccolor[loccolor == "Cyan"] <- "cyan"
loccolor[loccolor == "Orange"] <- "orange"
loccolor[loccolor == "Purple"] <- "darkorchid1"
loccolor[loccolor == "Pink"] <- "deeppink"
loccolor[loccolor == "Red"] <- "red"

```

```

loccolor[loccolor == "Gray"] <- "gray74"
loccolor[loccolor == "Teal"] <- "turquoise4"
loccolor[loccolor == "Blue"] <- "blue"
loccolor[loccolor == "Yellow"] <- "yellow"
loccolor[loccolor == "Black"] <- "black"
loccolor[loccolor == "Brown"] <- "brown"
loccolor[loccolor == "Pale blue"] <- "steelblue1"
loccolor[loccolor == "Green"] <- "green"
tt <- plotBasicNetwork(
  centroids = coords, brain,
  nodecolors = loccolor, radius = 3
)
dd <- make3ViewPNG(handMat2, id, rid2, tempfile(fileext = ".png"))
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 = tempfile(fileext = ".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)
}
```

posthoc_fa_summary	<i>Post-hoc Factor Analysis Summary: Scores, Communalities, and Variance Explained</i>
--------------------	--

Description

Given data and factor loadings, this function computes factor scores, communalities, uniquenesses, and per-factor variance explained using linear modeling. It provides a convenient post-hoc summary of a factor analysis solution.

Usage

```
posthoc_fa_summary(data, loadings, scores = NULL, method = "PAF")
```

Arguments

data	Numeric matrix or data.frame of observed variables (n x p).
loadings	Numeric matrix of factor loadings (p x k).
scores	Optional numeric matrix of factor scores (n x k). If NULL scores are computed by least squares regression.
method	Character string describing the extraction method (e.g., "PAF", "NSA-Flow", "ML").

Value

A list with components:

scores Matrix of factor scores (n x k).

communalities Named numeric vector of communalities for each variable.

uniquenesses Named numeric vector of uniquenesses (1 - communality).

variance_per_factor Data frame with variance explained per factor.

total_variance_explained Total proportion of variance explained across factors.

reconstructed Reconstructed data from scores and loadings.

residuals Matrix of residuals: data - reconstructed.

Examples

```
set.seed(42)
n <- 100; p <- 6; k <- 2
scores_true <- matrix(rnorm(n * k), n, k)
loadings <- matrix(runif(p * k, 0.5, 0.8), p, k)
data <- scores_true %*% t(loadings) + matrix(rnorm(n * p, sd = 0.5), n, p)
colnames(data) <- paste0("Var", 1:p)

results <- posthoc_fa_summary(data, loadings, method = "PAF")
str(results)
```

powers_areal_mni_itk *powers_areal_mni_itk*

Description

A data frame providing coordinates for the nodes identified by Powers et al. Coordinates have been modified from nifti space to ITK space

Usage

```
powers_areal_mni_itk
```

Format

A data frame listing the following variables.

x x coordinate

y y coordinate

z z coordinate

ROI unique integer for each node

SystemLabel unique integer for each system

SystemName unique name for each system
 Color name of color for system
 r red value for system
 g green value for system
 b blue value for system
 Anatomy anatomical location as determined by OASIS labeling
 Lobe lobe location of node
 Brodmann Brodmann area number
 AAL AAL region label

References

Power, J. D., et al. (2011). "Functional network organization of the human brain." *Neuron* 72(4): 665-678.

Examples

```

data(powers_areal_mni_itk)
## Not run:
fixed <- antsImageRead(getANTsRData("ch2"))
moving <- antsImageRead(getANTsRData("mni"))
mytx <- antsRegistration(
  fixed = fixed, moving = moving,
  typeofTransform = c("SyN")
)
data("powers_areal_mni_itk", package = "ANTsR", envir = environment())
coords <- powers_areal_mni_itk[, 1:3]
ch2reg <- antsRegistration(fixed, moving, typeofTransform = "SyN")
coordsw <- antsApplyTransformsToPoints(
  dim = 3, points = coords,
  transformlist = ch2reg$fdwtransforms,
  whichtoinvert = c(FALSE, FALSE)
)
ptrd <- 3
powersLabels <- makePointsImage(coordsw, moving, radius = pard)
plot(moving, powersLabels, axis = 3, nslices = 30)

## End(Not run)

```

powers_mni_itk

powers mni itk

Description

powers mni itk

ppmi_template_500Parcels_Yeo2011_17Networks_2023_homotopic
<i>ppmi template 500Parcels Yeo2011 17Networks 2023 homotopic</i>

Description

ppmi template 500Parcels Yeo2011 17Networks 2023 homotopic

ppmi_yeo_labels	<i>ppmi_yeo_labels</i>
-----------------	------------------------

Description

A data frame providing coordinates and names for the PPMI 500 parcels Yeo atlas.

Usage

ppmi_yeo_labels

Format

A data frame with columns: x, y, z, ROI, SystemName, AAL.

predict.fusedRidge	<i>Predict Method for Fused Ridge Regression</i>
--------------------	--

Description

Obtains predictions from a fitted fusedRidge object on new data.

Usage

```
## S3 method for class 'fusedRidge'
predict(
  object,
  newx,
  newcovs = NULL,
  type = c("link", "response"),
  topK = NULL,
  ...
)
```

Arguments

object	A fitted fusedRidge object.
newx	Matrix of new predictor variables.
newcovs	Optional matrix or data frame of new covariates. Must be provided if the model was trained with covariates.
type	Type of prediction. "link" returns the linear predictor, and "response" returns the fitted probabilities (only relevant for family = "binomial").
topK	Optional integer. If provided, keeps only the topK absolute largest feature weights for prediction, zeroing out all others (enables sparsity control).
...	Additional arguments (not used).

Value

A matrix of predictions with dimensions nrow(newx) x length(thresholds).

predict.fusedRidgeDirect

Predict Method for Direct Fused Ridge Regression

Description

Obtains predictions from a fitted fusedRidgeDirect object on new data.

Usage

```
## S3 method for class 'fusedRidgeDirect'
predict(
  object,
  newx,
  newcovs = NULL,
  type = c("link", "response"),
  topK = NULL,
  ...
)
```

Arguments

object	A fitted fusedRidgeDirect object.
newx	Matrix of new predictor variables.
newcovs	Optional matrix or data frame of new covariates. Must be provided if the model was trained with covariates.
type	Type of prediction. "link" returns the linear predictor, and "response" returns the fitted probabilities (only relevant for family = "binomial" or "poisson").
topK	Optional integer. If provided, keeps only the topK absolute largest feature weights for prediction, zeroing out all others (enables sparsity control).
...	Additional arguments (passed to predict.fusedRidge).

Value

A matrix of predictions with dimensions `nrow(newx) x length(thresholds)`.

<code>predict.fusedRidgeTorch</code>
<i>Predict Method for PyTorch Fused Ridge Regression</i>

Description

Obtains predictions from a fitted `fusedRidgeTorch` object on new data.

Usage

```
## S3 method for class 'fusedRidgeTorch'
predict(
  object,
  newx,
  newcovs = NULL,
  type = c("link", "response"),
  topK = NULL,
  ...
)
```

Arguments

<code>object</code>	A fitted <code>fusedRidgeTorch</code> object.
<code>newx</code>	Matrix of new predictor variables.
<code>newcovs</code>	Optional matrix or data frame of new covariates. Must be provided if the model was trained with covariates.
<code>type</code>	Type of prediction. "link" returns the linear predictor, and "response" returns the fitted probabilities.
<code>topK</code>	Optional integer. If provided, keeps only the topK absolute largest feature weights for prediction, zeroing out all others (enables sparsity control).
<code>...</code>	Additional arguments (passed to <code>predict.fusedRidge</code>).

Value

A matrix of predictions with dimensions `nrow(newx) x length(thresholds)`.

predictSimlr	<i>predict output matrices from simlr solution</i>
--------------	--

Description

predict output matrices from simlr solution

Usage

```
predictSimlr(x, simsol, targetMatrix, sourceMatrices, projectv = TRUE)
```

Arguments

x	A list that contains the named matrices.
simsol	the simlr solution
targetMatrix	an optional index that sets a fixed target (outcome) matrix. If not set, each basis set will be used to predict its corresponding matrix.
sourceMatrices	an optional index vector that sets predictor matrices. If not set, each basis set will be used to predict its corresponding matrix.
projectv	boolean to determine whether raw u or $x * v$ is used; default to TRUE which uses the $x * v$ approach

Value

A list of variance explained, predicted matrices and error metrics:

varx: Mean variance explained for u_i predicting x_i .

predictions: Predicted x_i matrix.

initialErrors: Initial matrix norm.

finalErrors: Matrix norm of the error term.

aggregateTstats: Summary t-stats for each matrix and each u_i .

uOrder: Estimated order of the u_i predictors aggregated over all terms.

Author(s)

BB Avants.

Examples

```
# see simlr examples
```

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

Arguments

<code>boldImage</code>	4-D ANTs image fMRI data.
<code>maskImage</code>	3-D ANTs image defining the region of interest.
<code>maskingMeanRatioThreshold</code>	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.
<code>initialNuisanceVariables</code>	Optional initial nuisance variables.
<code>numberOfCompCorComponents</code>	Numer of CompCor nuisance components.
<code>doMotionCorrection</code>	Boolean indicating whether motion correction should be performed and used in nuisance regression.
<code>useMotionCorrectedImage</code>	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.

<code>motionCorrectionAccuracyLevel</code>	Accuracy for the motion correcting registration: 0 = fast/debug parameters, 1 = intrasession parameters, or 2 = intersession/intersubject parameters.
<code>meanBoldFixedImageForMotionCorrection</code>	Optional target fixed image for motion correction.
<code>frequencyLowThreshold</code>	Lower threshold for bandpass filtering.
<code>frequencyHighThreshold</code>	Upper threshold for bandpass filtering.
<code>spatialSmoothingType</code>	Either none, gaussian (isotropic) or perona-malik (anisotropic) smoothing.
<code>spatialSmoothingParameters</code>	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. <code>c(0.25, 5)</code> .
<code>residualizeMatrix</code>	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.

Preprocess resting BOLD fMRI image data.

Description

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

Usage

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

Arguments

<code>boldImage</code>	4-D ANTs image fMRI data.
<code>maskImage</code>	3-D ANTs image defining the region of interest.
<code>maskingMeanRatioThreshold</code>	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.
<code>initialNuisanceVariables</code>	Optional initial nuisance variables.
<code>denseFramewise</code>	boolean for dense framewise stats
<code>numberOfCompCorComponents</code>	Numer of CompCor nuisance components.
<code>doMotionCorrection</code>	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.
motionCorrectionIterations	number of iteration for motion correction
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. <code>c(0.25, 5)</code> .
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.

priorBasedSegmentation

spatial prior-based image segmentation.

Description

markov random field regularized, prior-based image segmentation that is a wrapper around atropos (see ANTs and related publications).

Usage

```
priorBasedSegmentation(
  img,
  priors,
  mask,
  priorWeight = 0.25,
  mrf = 0.1,
  iterations = 25,
  verbose = FALSE
)
```

Arguments

img	input image or image list for multivariate segmentation
priors	list of priors that cover the number of classes
mask	segment inside this mask
priorWeight	usually 0 (priors used for initialization only), 0.25 or 0.5.
mrf	regularization, higher is smoother, a numerical value in range 0.0 to 0.2
iterations	maximum number of iterations. could be a large value eg 25.
verbose	maximum number of iterations. could be a large value eg 25.

Value

segmentation and probability images

Author(s)

Brian B. Avants

Examples

```
fi <- antsImageRead(getANTsRData("r16"))
seg <- kmeansSegmentation(fi, 3)
msk <- thresholdImage(seg$segmentation, 1, Inf)
pseg <- priorBasedSegmentation(fi, seg$probabilityimages, msk, 0.25, 0.1, 3)
```

project_to_orthonormal_nonnegative

Project Non-Negative Matrix to Orthonormal Columns Matrix

Description

Iteratively projects a non-negative matrix onto the intersection of non-negative matrices and matrices with orthonormal columns.

Usage

```
project_to_orthonormal_nonnegative(
    X,
    max_iter = 100,
    tol = 1e-04,
    constraint = "positive"
)
```

Arguments

X	A square numeric matrix with non-negative entries.
max_iter	Maximum number of iterations for the alternating projection.
tol	Tolerance for convergence. The projection stops if the change in the matrix norm is below this tolerance, or if max_iter is reached.
constraint	either or positive

Value

A matrix Y that is non-negative and whose columns are orthonormal, closest to X in a sense defined by the alternating projections.

project_to_partially_orthonormal_nonnegative

Project Non-Negative Matrix with Controllable Orthonormality

Description

Iteratively projects a non-negative matrix towards the intersection of non-negative matrices and matrices with orthonormal columns. Allows control over the degree of orthonormality enforced.

Usage

```
project_to_partially_orthonormal_nonnegative(
  X,
  max_iter = 100,
  tol = 1e-06,
  constraint = "positive",
  ortho_strength = 1,
  epsilon = 1e-04
)
```

Arguments

<code>X</code>	A square or rectangular numeric matrix with non-negative entries.
<code>max_iter</code>	Maximum number of iterations for the alternating projection.
<code>tol</code>	Tolerance for convergence (Frobenius norm of change).
<code>constraint</code>	'positive' or 'either' (influences initial data preprocessing).
<code>ortho_strength</code>	A value between 0 and 1, controlling the degree of orthonormality enforced. 0 means no orthonormality enforcement (only non-negativity), 1 means full orthonormality.
<code>epsilon</code>	very small value for stability.

Value

A matrix `Y` that is non-negative and has columns that are controlled in their orthonormality by `ortho_strength`.

Examples

```
# Example matrix (non-negative)
X_sample <- matrix(c(
  1, 2, 0.1,
  0.5, 1, 0.8,
  0.2, 0.5, 1
), nrow=3, byrow=TRUE)

# Full orthonormality (ortho_strength = 1)
Y_full_ortho = project_to_partially_orthonormal_nonnegative(
  X_sample, ortho_strength = 1, max_iter = 5)

# No orthonormality enforcement (ortho_strength = 0) -
# should just be pmax(X, 0)
Y_no_ortho <- project_to_partially_orthonormal_nonnegative(
  X_sample, ortho_strength = 0, max_iter = 5) # Low iter for speed
```

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 minimum)
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 = NULL, 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 = TRUE,
  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)
```

quantile.antsImage	<i>Sample Quantiles</i>
--------------------	-------------------------

Description

Computes sample quantiles for an image, with the option of a mask.

Usage

```
## S3 method for class 'antsImage'
quantile(x, ..., mask)
```

Arguments

x	Object of class antsImage
...	Arguments passed to quantile
mask	object to subset the image. If missing, then all values of the image are used

Value

Output of [quantile](#)

Examples

```
img <- makeImage(c(10, 10), rnorm(100))
mask <- img > 0
quantile(img, mask = mask)
```

randcca	<i>randomized cca function.</i>
---------	---------------------------------

Description

uses random matrix to estimate cca results

Usage

```
randcca(x, y, k, seed = NA)
```

Arguments

x	input matrix
y	input matrix
k	rank to use
seed	for testing

Value

outputs a list containing:

whitened: low-rank whitened joint matrix

svd: low-rank svd of joint matrix

Author(s)

Avants BB

Examples

```
set.seed(13)
x <- matrix(rnorm(3000), nrow = 50)
y <- x %% matrix(rnorm(60 * 100), nrow = 60)
k <- 10
dr <- randcca(x, y, k, 1)
```

randomlyTransformImageData

Randomly transform image data (optional: with corresponding segmentations).

Description

Apply rigid, affine and/or deformable maps to an input set of training images. The reference image domain defines the space in which this happens.

Usage

```
randomlyTransformImageData(
  referenceImage,
  inputImageList,
  segmentationImageList = NULL,
  numberOfSimulations = 10,
  transformType = "affine",
  sdAffine = 0.02,
  deformationTransformType = c("bspline", "exponential"),
  numberOfRandomPoints = 1000,
  sdNoise = 10,
  numberOfFittingLevels = 4,
  meshSize = 1,
  sdSmoothing = 4,
  inputImageInterpolator = "linear",
  segmentationImageInterpolator = "nearestNeighbor"
)
```

Arguments

- `referenceImage` defines the spatial domain for all output images. If the input images do not match the spatial domain of the reference image, we internally resample the target to the reference image. This could have unexpected consequences. Resampling to the reference domain is performed by testing using `antsImagePhysicalSpaceConsistency` then calling `resampleImageToTarget` upon failure.
- `inputImageList` list of lists of input images to warp. The internal list sets contains one or more images (per subject) which are assumed to be mutually aligned. The outer list contains multiple subject lists which are randomly sampled to produce output image list.
- `segmentationImageList` list of segmentation images corresponding to the input image list (optional).
- `numberOfSimulations` number of output images. Default = 10.
- `transformType` one of the following options `c("translation", "rotation", "rigid", "scaleShear", "affine", "deformation", "affineAndDeformation")`. Default = `"affine"`.

<code>sdAffine</code>	parameter dictating deviation amount from identity for random linear transformations. Default = 0.02.
<code>deformationTransformType</code>	one of the following options <code>c("bspline", "exponential")</code> if deformation is specified in the transformType. Default = <code>"bspline"</code> .
<code>numberOfRandomPoints</code>	number of displacement points for the deformation field. Default = 1000.
<code>sdNoise</code>	standard deviation of the displacement field noise (in mm). Default = 10.0.
<code>numberOfFittingLevels</code>	(bspline deformation only) number of fitting levels. Default = 4.
<code>meshSize</code>	(bspline deformation only) scalar or n-D vector determining fitting resolution. Default = 1.
<code>sdSmoothing</code>	(exponential deformation only) standard deviation of the Gaussian smoothing in mm. Default = 4.0.
<code>inputImageInterpolator</code>	one of the following options <code>c("linear", "gaussian", "bspline")</code> . Default = <code>"linear"</code> .
<code>segmentationImageInterpolator</code>	one of the following options <code>c("nearestNeighbor", "genericLabel")</code> . Default = <code>"nearestNeighbor"</code> .

Value

list (if no directory set) or boolean for success, failure

Author(s)

Avants BB, Tustison NJ

Examples

```
library( ANTsR )
image1 <- antsImageRead( getANTsRData( "r16" ) )
image2 <- antsImageRead( getANTsRData( "r64" ) )
segmentation1 <- thresholdImage( image1, "Otsu", 3 )
segmentation2 <- thresholdImage( image2, "Otsu", 3 )
data <- randomlyTransformImageData( image1,
  list( list( image1 ), list( image2 ) ),
  list( segmentation1, segmentation2 ) )
rm(segmentation1); gc()
rm(segmentation2); gc()
rm(image1); gc()
rm(image2); gc()
```

randomMask	<i>Get a random mask</i>
------------	--------------------------

Description

Get a mask with a specific number of voxels randomly distributed on the input image/mask

Usage

```
randomMask(img, nsamples, perLabel = FALSE, seed)
```

Arguments

- | | |
|----------|---|
| img | an antsImage, either continuous or mask-like. This will define the space where to sample voxels from. If this image is continuous (i.e., a T1), voxels will be sampled from all nonzero voxels. |
| nsamples | the number of random voxels to select from |
| perLabel | logical, if true the input image must be multi-label and the output will have nsamples from each label (i.e., 200 samples from each label) |
| seed | random seed |

Value

binary antsImage with random voxels set to 1

Author(s)

Pustina D

Examples

```
img <- antsImageRead(getANTSRData("r16"))
randmask <- randomMask(img, 200)
```

randsvd	<i>randomized svd function.</i>
---------	---------------------------------

Description

uses random matrix to estimate svd results

Usage

```
randsvd(A, k, seed = NA)
```

Arguments

A	input matrix
k	rank to use
seed	for testing

Value

randomized svd is output

Author(s)

Avants BB

References

N. Halko, P.G. Martinsson, J. Tropp "Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions" arXiv 0909.4061

Examples

```
A <- matrix(rnorm(3000), ncol = 50)
k <- 10
dr <- randsvd(A, k)$d[1:k]
dt <- svd(A)$d[1:k]
cor(dr, dt)
```

randwhiten

randomized whitening function.

Description

uses random matrix to whiten a matrix

Usage

```
randwhiten(A, k, seed = NA)
```

Arguments

A	input matrix
k	rank to use
seed	for testing

Value

whitened matrix is output

Author(s)

Avants BB

References

N. Halko, P.G. Martinsson, J. Tropp "Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions" arXiv 0909.4061

Examples

```
A <- matrix(rnorm(3000), ncol = 50)
k <- 10
Aw <- randwhiten(A, k)
# (Aw) %*% t(Aw)
```

rankBasedMatrixSegmentation

rank-based segmentation of a matrix

Description

rank-based segmentation of a matrix

Usage

```
rankBasedMatrixSegmentation(
  v,
  sparsenessQuantile,
  basic = FALSE,
  positivity = "positive",
  transpose = FALSE
)
```

Arguments

v	input matrix
sparsenessQuantile	a value in zero to one
basic	simplest keep top k-entries in each row
positivity	one of positive, negative, either
transpose	work on the transpose

Value

matrix

Author(s)

Avants BB

Examples

```
mat <- matrix(1:200, nrow = 10)
matr <- rankBasedMatrixSegmentation(mat, 0.9, basic = FALSE, positivity = "positive")
```

rankIntensity	<i>rank intensity transformation for an image</i>
---------------	---

Description

Rank transform the intensity of an image to reduce the impact of outliers; useful when using correlation-based metrics for registration and possibly for segmentation tasks. best used with a mask.

Usage

```
rankIntensity(x, mask, getMask = TRUE, method = "max")
```

Arguments

x	input image
mask	optional mask
getMask	boolean only used if mask is not present
method	optional method passed to ties.method in rank

Value

image is output

Author(s)

Avants BB

Examples

```
fi <- makeImage(c(10, 10, 10), rnorm(1000))
rfi <- rankIntensity(fi)
```

`rapidlyInspectImageData`*Simple rapidlyInspectImageData function.*

Description

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

Usage

```
rapidlyInspectImageData(myfiles)
```

Arguments

<code>myfiles</code>	input list of filenames
----------------------	-------------------------

Value

matrix is output

Author(s)

Avants BB

Examples

```
## Not run:
fn1 <- c(
  getANTsRData("r16"),
  getANTsRData("r27"),
  getANTsRData("r62"),
  getANTsRData("r64"),
  getANTsRData("r85")
)
mm <- rapidlyInspectImageData(fn1)

## End(Not run)
```

read_simlr	<i>Read all SiMLR outputs from disk</i>
------------	---

Description

Reconstructs SiMLR outputs written by `write_simlr` using the manifest with relative paths for portability. Restores row names for matrices and data frames from the CSV files.

Usage

```
read_simlr(dir)
```

Arguments

<code>dir</code>	A character string specifying the directory with SiMLR outputs.
------------------	---

Value

A named list reconstructing the original SiMLR object.

read_simlr_data_frames	<i>Read a list of data frames from disk with SiMLR-specific naming convention</i>
------------------------	---

Description

This function reads a list of data frames from disk into a list, assuming the files are named with a common prefix and the names of the data frames. It converts the column named 'X' to the row names of the read data frame.

Usage

```
read_simlr_data_frames(file_prefix, data_names, verbose = FALSE)
```

Arguments

<code>file_prefix</code>	A character string used as the prefix for the filenames.
<code>data_names</code>	A character vector of names for the data frames.
<code>verbose</code>	boolean

Value

A list of data frames read from disk with the column named 'X' set as row names.

Examples

```
# data_names <- c("data1", "data2")
# data_list <- read_simlr_data_frames(file_prefix = "output", data_names = data_names)
# dim(data_list[[1]])
# dim(data_list[[2]])
```

readAntsrTransform	<i>readAntsrTransform</i>
--------------------	---------------------------

Description

read a transform from file

Usage

```
readAntsrTransform(filename, dimension = NA, precision = "float")
```

Arguments

filename	filename of transform
dimension	spatial dimension of transform
precision	numerical precision of transform

Value

antsrTransform

Examples

```
trans <- c(3, 4, 5)
tx <- createAntsrTransform(type = "Euler3DTransform", translation = trans)
txfile <- tempfile(fileext = ".mat")
writeAntsrTransform(tx, txfile)
tx2 <- readAntsrTransform(txfile)
testthat::expect_error(readAntsrTransform(txfile, 2), "space dim")
```

```
readNormalizedPopulationData
```

Read normalized population from h5 file or directory

Description

This function reads a file created by writeNormalizedPopulationData.

Usage

```
readNormalizedPopulationData(filename)
```

Arguments

filename input filename for hdf5 file.

Value

list containing demographics, imageMat, imageMask and imageBoolean

Author(s)

Avants BB

Examples

```
tfn <- system.file("extdata", "normpop.h5", package = "ANTsR")
if (file.exists(tfn)) {
  dlist <- readNormalizedPopulationData(tfn)
}
## Not run:
ilist <- getANTsRData("population")
mask <- getMask(ilist[[1]])
imat <- imageListToMatrix(ilist, mask)
demog <- data.frame(age = c(11, 7, 18, 22), sex = c("M", "M", "F", "F"))
ibool <- c(TRUE, TRUE, TRUE, FALSE)
tfn <- tempfile(fileext = ".h5")
if (usePkg("hdf5r")) writeNormalizedPopulationData(demog, imat, mask, ibool, tfn)
if (usePkg("hdf5r")) {
  dlist <- readNormalizedPopulationData(tfn)
  all.equal(dlist$demographics, demog)
  all.equal(dlist$imageMat, imat)
  all.equal(as.array(dlist$imageMask), as.array(mask))
  all.equal(dlist$imageBool, ibool)
}

## End(Not run)
```

reconstructImageFromPatches

Reconstruct image from a list of patches.

Description

Reconstruct image from a list of patches.

Usage

```
reconstructImageFromPatches(
  patches,
  domainImage,
  strideLength = 1,
  domainImageIsMask = FALSE
)
```

Arguments

patches	List or array of patches defining an image. Patches are assumed to have the same format as returned by extractImagePatches.
domainImage	Image or mask to define the geometric information of the reconstructed image. If this is a mask image, the reconstruction will only use patches in the mask.
strideLength	Defines the sequential patch overlap for maxNumberOfPatches = all. Can be a image-dimensional vector or a scalar.
domainImageIsMask	boolean specifying whether the domain image is a mask used to limit the region of reconstruction from the patches.

Value

an ANTs image.

Author(s)

Tustison NJ

Examples

```
library( ANTsR )

image <- antsImageRead( getANTsRData( "r16" ) )
patchSet <- extractImagePatches( image, c( 64, 64 ), "all", c( 8, 8 ) )
imageReconstructed <-
  reconstructImageFromPatches( patchSet, image, c( 8, 8 ) )
testthat::expect_equal(as.array(image), as.array(imageReconstructed))
rm(image); gc()
```

```
rm(patchSet); gc()
rm(imageReconstructed); gc()
```

refbasis_brain	<i>refbasis brain</i>
----------------	-----------------------

Description

refbasis brain

refbasis_head	<i>refbasis head</i>
---------------	----------------------

Description

refbasis head

reference_basis	<i>reference basis</i>
-----------------	------------------------

Description

reference basis

reflectImage	<i>reflectImage</i>
--------------	---------------------

Description

reflects an image along its axis

Usage

```
reflectImage(img1, axis = NA, tx = NA, metric = "mattes", verbose = TRUE, ...)
```

Arguments

- | | |
|---------|---|
| img1 | input object, an antsImage |
| axis | which dimension to reflect across, numbered from 0 to imageDimension-1 |
| tx | transformation type to estimate after reflection |
| metric | similarity metric for image registration. see antsRegistration. |
| verbose | print diagnostic messages, passed to antsRegistration and antsApplyTransforms |
| ... | Additional options to pass to antsRegistration |

Author(s)

BB Avants

See Also[antsRegistration](#)**Examples**

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

reflectionMatrix	<i>reflectionMatrix</i>
------------------	-------------------------

Description

return a filename pointing to a reflection matrix

Usage

```
reflectionMatrix(img1, axis = NA, reflectionMatrixFilename)
```

Arguments

img1	input object, an antsImage
axis	which dimension to reflect across, numbered from 0 to imageDimension-1
reflectionMatrixFilename	a filename ending in .mat

Author(s)

BB Avants

regressionMatchImage *Image intensity normalization using linear regression.*

Description

Image intensity normalization by regressing the image intensities of the reference image with the source image.

Usage

```
regressionMatchImage(  
  sourceImage,  
  referenceImage,  
  mask = NULL,  
  polyOrder = 1,  
  truncate = TRUE  
)
```

Arguments

sourceImage	image whose intensities we will match to the referenceImage intensities.
referenceImage	defines the reference intensity function.
mask	Defines voxels for regression modeling.
polyOrder	of polynomial fit. Default is 1 (linear fit).
truncate	boolean which turns on/off the clipping of intensities.

Value

the sourceImage matched to the referenceImage.

Author(s)

Avants BB

Examples

```
library(ANTSRCore)  
sourceImage <- antsImageRead( getANTSRData( "r16" ) )  
referenceImage <- antsImageRead( getANTSRData( "r64" ) )  
matchedImage <- regressionMatchImage( sourceImage, referenceImage )  
bad_source = sourceImage[1:200, 1:200]  
testthat::expect_error(regressionMatchImage( bad_source, referenceImage ))
```

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

<code>input.train</code>	Masked imaging data from training set of type <code>antsImage</code> or <code>matrix</code> . This will typically be read from a <code>train.mha</code> file generated by, e.g., <code>sccan --imageset-to-matrix</code> .
<code>input.test</code>	Masked imaging data from testing set of type <code>antsImage</code> or <code>matrix</code> . This will typically be read from a <code>test.mha</code> file generated by, e.g., <code>sccan --imageset-to-matrix</code> .
<code>demog.train</code>	Data frame of demographics information for training images.
<code>demog.test</code>	Data frame of demographics information for testing images.
<code>eigenvectors</code>	List of eigenvector images for dimensionality reduction.
<code>mask</code>	Mask image of type <code>antsImage</code> .
<code>outcome</code>	Name of outcome variable to be predicted. Must be present in <code>demog.train</code> and <code>demog.test</code> .
<code>covariates</code>	List of names of covariates to be used for the prediction. All names must be present in <code>demog.train</code> and <code>demog.test</code> .
<code>model.function</code>	Modeling function for predicting outcome from input data. Can be any function that has formula and data arguments and a <code>predict</code> method. Defaults to <code>glm</code> , but <code>svm</code> , <code>randomForest</code> , 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)
eanatimages <- matrixToImages(mydecom$eig, mask)
result <- regressProjections(
  input[data.split$rows.in, ],
  input[data.split$rows.out, ],
  data.split$data.in, data.split$data.out,
  eanatimages,
  mask, "outcome"
)
```

regularizeSimlr

*Automatically produce regularization matrices for simlr***Description**

Automatically produce regularization matrices for simlr

Usage

```
regularizeSimlr(x, knn, fraction = 0.1, sigma, kPackage = "FNN")
```

Arguments

x	A list that contains the named matrices. Note: the optimization will likely perform much more smoothly if the input matrices are each scaled to zero mean unit variance e.g. by the scale function. Note: x may also contain a mixture of raw matrix data and masks which are binary antsImages. If a mask is passed, this function will assume the user wants spatial regularization for that entry.
knn	A vector of knn values (integers, same length as matrices)
fraction	optional single scalar value to determine knn
sigma	optional sigma vector for regularization (same length as matrices)
kPackage	name of package to use for knn. FNN is reproducible but RcppHNSW is much faster (with nthreads controlled by environment variable ITK_GLOBAL_DEFAULT_NUMBER_OF_THREADS) for larger problems. For large problems, compute the regularization once and save to disk; load for repeatability.

Value

A list of regularization matrices.

Author(s)

BB Avants.

Examples

```
# see simlr examples
```

reho

REHO

Description

regional homogeneity for a time series image

Usage

```
reho(rsfmri, mask, shifter = 1, method = "kendall", maxDistance = 1.1)
```

Arguments

rsfmri	antsImage usually 4D
mask	antsImage binary mask
shifter	size of neighborhood
method	one of kendall, spearman or pearson
maxDistance	maximum distance from center voxel

Value

antsImage is output

Author(s)

Tustison, N.

References

Zang, Y., Jiang, T., Lu, Y., He, Y., Tian, L., 2004. Regional homogeneity approach to fMRI data analysis. *Neuroimage* 22, 394-400.

Examples

```
## Not run:  
rehoimg <- reho(rsfmri)  
  
## End(Not run)
```

`renormalizeProbabilityImages`*Normalize (modified) probability images*

Description

Probability image lists should sum to one at every voxel. Smoothing or other common operations may break this constraint. This function allows one to easily ensure a list of probability images sums to one everywhere. It also allows one to prioritize specific images in the probability set such that its probability is maintained while others are normalized with respect to the prioritized value. Will also set values less than zero, to zero.

Usage

```
renormalizeProbabilityImages(probs, mask, k = NA)
```

Arguments

<code>probs</code>	list of images
<code>mask</code>	image
<code>k</code>	which image, if any, to prioritize

Value

list is output

Author(s)

Avants BB

Examples

```
mat <- t(matrix(rnorm(4000), ncol = 10))
mask <- makeImage(c(20, 20), 1)
plist <- matrixToImages(mat, mask)
rplist <- renormalizeProbabilityImages(plist, mask)
rplist2 <- renormalizeProbabilityImages(plist, mask, 2)
```

reorientImage	<i>reorient image by its principal axis</i>
---------------	---

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

resampleImage	<i>resampleImage</i>
---------------	----------------------

Description

Resample image by spacing or number of voxels with various interpolators. Works with multi-channel images.

Usage

```
resampleImage(
  image,
  resampleParams,
  useVoxels = FALSE,
  interpType = "nearestneighbor"
)
```

Arguments

image	input antsImage
resampleParams	vector of size dimension with numeric values
useVoxels	true means interpret resample params as voxel counts
interpType	one of 0 (linear), 1 (nearestNeighbor), 2 (gaussian), 3 (windowedSinc), 4 (bspline). The user can pass either a numeric or character argument here.

Value

output antsImage

Author(s)

Avants BB

Examples

```
fi <- antsImageRead(getANTsRData("r16"))
r <- range(fi)
finn <- resampleImage(fi, c(50, 60), TRUE, 0)
range(finn)
stopifnot(diff(range(finn)) > 0)
filin <- resampleImage(fi, c(1.5, 1.5), FALSE, 1)
range(filin)
stopifnot(diff(range(filin)) > 0)
pixeltype(fi) <- "float"
arr <- as.array(fi)
arr <- floor(arr)
class(arr) <- "integer"
fi <- as.antsImage(arr, pixeltype = "unsigned int")
filin <- resampleImage(fi, c(1.5, 1.5), FALSE, 1)
fi <- as.antsImage(arr > 14, pixeltype = "unsigned char")
filin <- resampleImage(fi, c(1.5, 1.5), FALSE, 1)

fi <- antsImageRead(getANTsRData("multi_component_image"))
filin <- resampleImage(fi, c(2, 2), FALSE, 1)
```

resampleImageToTarget *resampleImageToTarget*

Description

Resample image by using another image as target reference. This function uses `antsApplyTransform` with an identity matrix to achieve proper resampling.

Usage

```
resampleImageToTarget(
  image,
  target,
  interpType = "linear",
  imagetype = 0,
  verbose = FALSE,
  ...
)
```

Arguments

image	image to resample
target	image of reference, the output will be in this space
interpType	Choice of interpolator. Supports partial matching. linear nearestNeighbor multiLabel for label images but genericlabel is preferred gaussian bSpline cosineWindowedSinc welchWindowedSinc hammingWindowedSinc lanczosWindowedSinc genericLabel use this for label images
imagetype	choose 0/1/2/3 mapping to scalar/vector/tensor/time-series
verbose	print command and run verbose application of transform.
...	additional arugment passed to antsApplyTransforms C code

Value

output antsImage resampled with target's resolution/origin/orientation/direction

Author(s)

Pustina D

Examples

```
fname <- getANTsRData("r16")
fi <- antsImageRead(fname)
fi2mm <- resampleImage(fi, c(2, 2), useVoxels = 0, interpType = "linear")
resampled <- resampleImageToTarget(fi2mm, fi)
testthat::expect_error(resampleImageToTarget(fi2mm))
```

resels	<i>Estimates image resels</i>
--------	-------------------------------

Description

Utilize the estimated FWHM to find the resels per voxel

Usage

```
resels(mask, fwhm)
```

Arguments

mask	statistical value (typically the maxima of a cluster or statistical field)
fwhm	the full width at half maxima measurement

Details

Interprets a given antsImage mask (binarized so as to only contain 0s and 1s) into resolutions elements (as opposed to voxels). Doing so emphasizes the interdependent nature of voxels when undergoing RFT based statistical analyses. Optimized for three dimensions (2D has not yet been tested yet).

Value

A vector of resels for dimensions 0:D

Author(s)

Zachary P. Christensen

References

Worsley K.J., (1996) A Unified Statistical Approach for Determining Significant Signals in Images of Cerebral Activation.

See Also

rftPval, euler, rftResults

Examples

```
mask <- getMask(antsImageRead(getANTsRData("r16")))
myresels <- resels(mask, c(1, 1))
```

reset_optimizer_state	<i>Reset Optimizer State</i>
-----------------------	------------------------------

Description

Resets the internal state (momentum, second moments, etc.) of an optimizer. Useful when the optimization needs to restart from a previous best state.

Usage

```
reset_optimizer_state(optimizer, vmats)
```

Arguments

optimizer	The optimizer object.
vmats	The current V matrices to get the correct dimensions.

Value

The optimizer object with reset state.

rfSegmentation	<i>A rfSegmentation function.</i>
----------------	-----------------------------------

Description

Unsupervised image segmentation via random forests. An example.

Usage

```
rfSegmentation(  
  featureMatrix,  
  mask,  
  labeling = NULL,  
  ntrees = 100,  
  verbose = FALSE  
)
```

Arguments

featureMatrix	input matrix of features matched to mask size n predictors in rf
mask	input antsImage mask
labelimg	input antsImage labelimage, optional for supervised seg
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

See Also

[mrvmrfs](#)

Examples

```
## Not run:
if (usePkg("randomForest")) {
  img <- antsImageRead(getANTsRData("r16")) %>% iMath("Normalize")
  mask <- getMask(img)
  segs <- kmeansSegmentation(img, k = 3, kmask = mask)
  fmat0 <- t(antsrimate(getNeighborhoodInMask(img, mask, c(2, 2))))
  fmat1 <- t(antsrimate(getNeighborhoodInMask(
    segs$probabilityimages[[1]], mask, c(2, 2)
  )))
  fmat2 <- t(antsrimate(getNeighborhoodInMask(
    segs$probabilityimages[[2]], mask, c(2, 2)
  )))
  fmat3 <- t(antsrimate(getNeighborhoodInMask(
    segs$probabilityimages[[3]], mask, c(2, 2)
  )))
  fmat <- cbind(fmat0, fmat1, fmat2, fmat3)
  # produces proximity between all voxel pairs
  rfsegs <- rfSegmentation(fmat, verbose = FALSE)
  lrr <- lowrankRowMatrix(rfsegs, 10, faster = TRUE)
  nv <- eanatSelect(
    inmat = lrr, mask = mask, selectorScale = 1.2, cthresh = 50,
    verbose = T, smooth = 1
  )
  ee <- eanatDef(lrr, mask,
    nvecs = nv, smooth = 0., cthresh = 50,
    its = 2, verbose = TRUE
  )
  eseg <- eigSeg(mask, ee)
  plot(img, eseg)
}

## End(Not run)
```

rfSegmentationPredict *A rfSegmentationPredict function.*

Description

Predict image segmentation via random forests.

Usage

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

Arguments

rfSegmentationModel	input rf model
featureMatrix	input feature matrix
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)
  fmat <- t(antsrImpute(getNeighborhoodInMask(img, mask, c(2, 2))))
  rfsegs <- rfSegmentation(fmat, mask, segs$segmentation, ntrees = 100)
  fmat2 <- t(antsrImpute(getNeighborhoodInMask(img, mask2, c(2, 2))))
  rfseg2 <- rfSegmentationPredict(rfsegs$rfModel, fmat2, mask2)
}
```

rftPval

*RFT p-values***Description**

Calculates p-values of a statistical field using random field theory

Usage

```
rftPval(D, c, k, u, n, resels, df, fieldType)
```

Arguments

D	image dimensions
c	Threshold
k	spatial extent in resels (minimum cluster size in resels)
u	Number of clusters
n	number of statistical field in conjunction
resels	resel measurements of the search region
df	degrees of freedom expressed as $df = c(\text{degrees of interest, degrees of error})$
fieldType	T: T-field F: F-field X: Chi-square field' Z: Gaussian field

Details

This function calculates p-values of a thresholded statistical field at various levels:

set-level `rft.pval(D, c, k, u, n, resels, df, fieldType)`

cluster-level `rft.pval(D, 1, k, u, n, resels, df, fieldType)`

peak-level `rft.pval(D, 1, 0, u, n, resels, df, fieldType)`

Where set-level refers to obtaining the set of clusters, cluster-level refers to a specific cluster, and peak-level refers to the maximum (or peak) of a single cluster.

Value

The probability of obtaining the specified cluster

Pcor: corrected p-value

Pu: uncorrected p-value

Ec: expected number of clusters

ek: expected number of resels per cluster

Author(s)

Zachary P. Christensen

References

Friston K.J., (1994) Assessing the Significance of Focal Activations Using Their Spatial Extent.
 Friston K.J., (1996) Detecting Activations in PET and fMRI: Levels of Inference and Power.
 Worsley K.J., (1996) A Unified Statistical Approach for Determining Significant Signals in Images of Cerebral Activation.

See Also

rftResults, resels

Examples

```
## Not run:
# using rftPval for hypothetical 3D t-statistical image
# assume resels have been calculated and df = c(dfi, dfe)

# peak RFT p-value (peak = the maximum of a specific cluster)
peakP <- rftPval(3, 1, 0, peak, 1, resels, df, fieldType = "T")$Pcor

# cluster RFT p-value (u = the value the statistical field was threshold at
# and k = the size of the cluster in resels)
clusterP <- rftPval(3, 1, k, u, 1, resels, df, fieldType = "T")$Pcor

# set RFT p-value
setP <- rftPval(3, numberOfClusters, minimumClusterSize, u, 1, resels, df,
  fieldType = "T"
)$Pcor

## End(Not run)
```

rftResults

RFT Statistical Results

Description

Returns RFT based statistical results for a single statistical image

Usage

```
rftResults(
  x,
  resels,
  fwhm,
  df,
```

```

    fieldType,
    RPVImg = NULL,
    k = 1,
    threshType = "pRFT",
    pval = 0.05,
    pp = 0.001,
    n = 1,
    statdir = NULL,
    verbose = FALSE
)

```

Arguments

<code>x</code>	statistical field image of class <code>antsImage</code>
<code>resels</code>	resel values for the mask
<code>fwhm</code>	full width at half maxima
<code>df</code>	degrees of freedom expressed as $df = c(\text{degrees of interest, degrees of error})$
<code>fieldType</code>	T: T-field F: F-field X: Chi-square field' Z: Gaussian field
<code>RPVImg</code>	resels per voxel image
<code>k</code>	minimum desired cluster size (default = 1)
<code>threshType</code>	a numeric value to <code>threshTypeold</code> the statistical field or a character of the following methods: cRFT: computes a <code>threshTypeold</code> per expected cluster level probability pRFT: uses the mask and <code>pval</code> calculates the minimum statistical <code>threshTypeold</code> cFDR: uses an uncorrected <code>threshTypeold</code> at the alpha level and then computes and FDR <code>threshTypeold</code> based on cluster maxima pFDR: computes the <code>fdr threshTypeold</code> for the entire field of voxels
<code>pval</code>	the p-value for estimating the <code>threshTypeold</code> (default = .05)
<code>pp</code>	the primary (initial) p-value for <code>threshTypeolding</code> (only used for FDR methods; default = .001)
<code>n</code>	number of images in conjunction
<code>statdir</code>	directory where output is saved (if not specified images are not saved)
<code>verbose</code>	enables verbose output

Details

`rftPval` is used to compute all family-wise error (FWE) corrected statistics while `p.adjust` is used to compute all false-discovery rate based statistics. All statistics herein involve implementation of random field theory (RFT) to some extent.

Both cluster-level and peak-level statistics are described by the uncorrected p-value along with the FDR and FWE corrected p-values for each cluster. Peak-level statistics are described by the maximum statistical value in each cluster and the comparable Z statistic. The ClusterStats table also contains coordinates for each cluster and the number of voxels therein. By default threshType = "pRFT" and pval=.05. Alternatively, the user may use a specific numeric value for threshTypeolding the statistical field. statFieldThresh more fully describes using appropriate threshTypeolds for statistical fields and how pp plays a role in FDR threshTypeolding.

Value

Outputs a statistical value to be used for threshTypeold a statistical field image

SetStats: set-level statistics and number of clusters

ClusterStats: cluster-level statistics and descriptors

PeakStats: peak-level statistics and descriptor"

LabeledClusters: image of labeled clusters

threshTypeold: the threshTypeold used

Author(s)

Zachary P. Christensen

References

Chumbley J., (2010) Topological FDR for neuroimaging

Friston K.J., (1996) Detecting Activations in PET and fMRI: Levels of Inference and Power

Worsley K.J., (1992) A Three-Dimensional Statistical Analysis for CBF Activation Studies in Human Brain.

Examples

```
## Not run:
mnit1 <- antsImageRead(getANTsRData("mni"))
mask <- getMask(mnit1)
ilist <- list()
for (i in 1:10) {
  ilist <- lappend(ilist, antsImageClone(mnit1) * rnorm(1))
}
response <- rnorm(10)
imat <- imageListToMatrix(ilist, mask)
residuals <- matrix(nrow = nrow(imat), ncol = ncol(imat))
tvals <- matrix(nrow = nrow(imat), ncol = ncol(imat))
for (i in 1:ncol(imat)) {
  fit <- lm(response ~ imat[, i])
  tvals <- coefficients(fit)[2]
  residuals[, i] <- residuals(fit)
}
myfwhm <- estSmooth(residuals, mask, fit$df.residual)
res <- resels(mask, myfwhm$fwhm)
```

```

ting <- makeImage(mask, tvals)

# threshold to create peak values with p-value of .05 (default)
results1 <- rftResults(ting, res, myfwhm$fwhm, df,
  fieldType = "T",
  threshType = "pRFT", pval = .05
)

# threshold to create clusters with p-value of .05
results2 <- rftResults(ting, res, myfwhm$fwhm, df,
  fieldType = "T",
  threshType = "cRFT", pval = .05
)

# initial threshold at p-value of .001 followed by peak FDR threshTypeold at
# p-value of .05
results3 <- rftResults(ting, res, myfwhm$fwhm, df,
  fieldType = "T",
  threshType = "pFDR", pval = .05, pp = .01
)

# initial threshold at p-value of .001 followed by cluster FDR threshold at
# p-value of .05
results4 <- rftResults(ting, res, myfwhm$fwhm, df,
  fieldType = "T",
  threshType = "cFDR", pval = .05, pp = .01
)

# correcting for non-isotropic
results5 <- rftResults(ting, res, myfwhm$fwhm, df,
  fieldType = "T",
  fwhm$RPVImg
)

## End(Not run)

```

ri

Minimal function for creating r images

Description

r16, r27, r30, r62, r64, r85 are t1-weighted MRI slices of the human brain that have been around for a long time. We use them for quick registration, segmentation and other processing tests. This function will return these images to the user.

Usage

```
ri(x)
```

Arguments

`x` one of 16, 27, 30, 62, 64, 85 or 'all'. values 1 through 6 are also acceptable.

Value

image or image list is output

Author(s)

Avants BB

Examples

```
r16 <- ri(16)
```

ripmmarc

Rotation Invariant Patch-based Multi-Modality Analysis aRChitecture

Description

Patch-based and rotation invariant image decomposition. This is similar to patch-based dictionary learning in N-dimensions. Warning: there may be an overflow/underflow error in the C++ underlying this function that appears to occur when insufficient patchSamples are selected.

Usage

```
ripmmarc(
  img,
  mask,
  patchRadius = 3,
  patchSamples = 1000,
  patchVarEx = 0.95,
  meanCenter = TRUE,
  canonicalFrame = NULL,
  evecBasis = NULL,
  rotationInvariant = TRUE,
  regressProjections = TRUE,
  verbose = FALSE
)
```

Arguments

`img` Image to decompose

`mask` Binary mask defining regions in which to decompose.

`patchRadius` Scalar radius defining the patch size.

patchSamples	Scalar defining the number of random patches to sample.
patchVarEx	Scalar defining the target variance explained. If this is greater than one, then it defines the number of eigenvectors. Otherwise, it defines the target variance explained.
meanCenter	boolean whether we mean center the patches.
canonicalFrame	pass in an existing canonicalFrame.
evecBasis	pass in an existing eigenvector basis.
rotationInvariant	boolean sets whether patches are rotationInvariant.
regressProjections	boolean return reconstruction parameters.
verbose	boolean sets verbosity.

Value

list including the canonical frame, the matrix basis, the patches for the full image, the projection coefficients for the full image, the variance explained and a reconstructed image.

Author(s)

Kandel BM, Avants BB

Examples

```
img <- antsImageRead(getANTsRData("r16"))
msk <- getMask(img) %>% iMath("ME", 1)
lap <- iMath(img, "Laplacian", 2)
mskTestTrain <- antsImageClone(msk)
mskTestTrain[1:128, 1:256] <- 2
mskTestTrain <- mskTestTrain * msk
pr <- 2
nv <- 15
ripped <- ripmmarc(img, thresholdImage(mskTestTrain, 2, 2),
  patchRadius = pr,
  patchSamples = 5000, patchVarEx = nv, rotationInvariant = FALSE
)
ipatches <- ripped$imagePatchMat
ibasis <- scale(ripped$basisMat)
k <- 5
kk <- lm(ipatches[k, ] ~ t(ibasis[1:10, ]))
ring <- ripmmarcBasisImage(ripped$canonicalFrame, ipatches[k, ])
# plot( ring, doCropping=FALSE )
bimg <- ripmmarcBasisImage(ripped$canonicalFrame, ibasis[5, ])
# plot( bimg, doCropping=FALSE )

rippedTest <- ripmmarc(img, thresholdImage(mskTestTrain, 1, 1),
  patchRadius = pr,
  evecBasis = ripped$basisMat, canonicalFrame = ripped$canonicalFrame,
  patchSamples = 500, patchVarEx = nv, rotationInvariant = FALSE
```

```

)
lapTrainVox <- lap[mskTestTrain == 2]
lapTestVox <- lap[mskTestTrain == 1]
mydftr <- data.frame(lap = lapTrainVox, t1feats = ripped$evecCoeffs)
mydfte <- data.frame(lap = lapTestVox, t1feats = rippedTest$evecCoeffs)
mdl <- lm(lap ~ ., data = mydftr)
preds <- predict(mdl, newdata = mydfte)
# cor.test( preds, mydfte$lap )

```

ripmmarcBasisImage	<i>Generate an antsImage from ripmmarc basis data</i>
--------------------	---

Description

This function converts a vectorized image patch back into an `antsImage` such that it may be displayed in its extrinsic dimensionality.

Usage

```
ripmmarcBasisImage(canonicalFrame, patchBasisVector, eps = 1e-12)
```

Arguments

<code>canonicalFrame</code>	canonical frame image generated by <code>ripmmarc</code>
<code>patchBasisVector</code>	the basis vector to convert to image space
<code>eps</code>	small value threshold to help generate mask from <code>canonicalFrame</code>

Value

`antsImage`

Author(s)

Kandel BM, Avants BB

Examples

```

img <- antsImageRead(getANTsRData("rand"))
img <- resampleImage(img, c(32, 32))
msk <- thresholdImage(img, quantile(img[img > 0], 0.5)[1], max(img))
ripped <- ripmmarc(img, msk, patchRadius = 1, patchSamples = 20, patchVarEx = 2)
bimg <- ripmmarcBasisImage(ripped$canonicalFrame, ripped$basisMat[1, ])

```

ripmmarcPop	<i>RIPMMARC population basis</i>
-------------	----------------------------------

Description

Patch-based and rotation invariant image decomposition. This is similar to patch-based dictionary learning in N-dimensions. This implementation is more efficient for building a basis from image populations.

Usage

```
ripmmarcPop(  
  ilist,  
  mask,  
  patchRadius = 3,  
  patchSamples = 1000,  
  patchVarEx = 0.95,  
  meanCenter = TRUE,  
  seed  
)
```

Arguments

ilist	list of antsImages from which to learn basis
mask	Binary mask defining regions in which to decompose or a list of masks corresponding to ilist.
patchRadius	Scalar radius defining the patch size.
patchSamples	Scalar defining the number of random patches to sample.
patchVarEx	Scalar defining the target variance explained. If this is greater than one, then it defines the number of eigenvectors. Otherwise, it defines the target variance explained.
meanCenter	boolean whether we mean center the patches.
seed	seed to pass to randomMask . This behavior is different than setting the seed globally and running as randomMask is called multiple times and that runs set.seed

Value

list including the canonical frame, the matrix basis and its eigenvalues

Author(s)

Kandel BM, Avants BB

See Also

[ripmmarc](#)

Examples

```

pop <- getANTsRData("population") # list of example images
popmasks <- list()
for (i in 1:length(pop)) {
  popmasks[[i]] <- getMask(pop[[i]])
}
set.seed(1234)
rp <- ripmmarcPop(pop, popmasks,
  patchRadius = 3,
  meanCenter = TRUE, patchSamples = 1000
)
set.seed(1234)
rp2 <- ripmmarcPop(pop, popmasks,
  patchRadius = 3,
  meanCenter = TRUE, patchSamples = 1000
)
testthat::expect_equal(rp, rp2)
rp3 <- ripmmarcPop(pop, popmasks,
  patchRadius = 3,
  meanCenter = TRUE, patchSamples = 1000, seed = 1234
)
testthat::expect_failure(testthat::expect_equal(rp, rp3))
## Not run:
nv <- 15
rippedTest <- ripmmarc(pop[[3]], popmasks[[3]],
  patchRadius = 3,
  evectBasis = rp$basisMat[1:nv, ], patchVarEx = nv, meanCenter = TRUE,
  canonicalFrame = rp$canonicalFrame, regressProjections = TRUE
)
mm <- makeImage(popmasks[[3]], rippedTest$evectCoeffs[, 1])

## End(Not run)

```

robustMatrixTransform *Simple robustMatrixTransform function.*

Description

Transform a matrix such that each column has rank-ordered entries where rank #' is determined by the column values.

Usage

```
robustMatrixTransform(mat)
```

Arguments

mat	input matrix
-----	--------------

Value

matrix is output

Author(s)

Avants BB

Examples

```
mat <- replicate(100, rnorm(20))
rmat <- robustMatrixTransform(mat)
```

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("pcas1")
# 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, maxnoise_preds=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)

```

rvcoef

*Computes the RV-Coefficient (Optimized Public API)***Description**

This function automatically selects the fastest algorithm based on data dimensions.

Usage

```
rvcoef(X, Y)
```

Arguments

X	A numeric matrix (n observations, p variables).
Y	A numeric matrix (n observations, q variables).

Value

A single scalar value for the RV-coefficient.

rvcoef_components	<i>Computes the RV-Coefficient (Internal)</i>
-------------------	---

Description

This function automatically selects the fastest algorithm based on data dimensions. Internal Dispatcher: Returns all components from the fastest method.

Usage

rvcoef_components(X, Y)

Arguments

- X A numeric matrix (n observations, p variables).
- Y A numeric matrix (n observations, q variables).

Value

values.

rvcoef_gram_impl	<i>Computes the RV-Coefficient (Internal) Gram Matrix Method for Tall data</i>
------------------	--

Description

Computes the RV-Coefficient (Internal) Gram Matrix Method for Tall data

Usage

rvcoef_gram_impl(X_centered, Y_centered)

Arguments

- X_centered A numeric matrix (n observations, p variables).
- Y_centered A numeric matrix (n observations, q variables).

Value

values.

rvcoef_trace_impl	<i>Computes the RV-Coefficient (Internal) Trace Method for Wide data</i>
-------------------	--

Description

Computes the RV-Coefficient (Internal) Trace Method for Wide data

Usage

```
rvcoef_trace_impl(X_centered, Y_centered)
```

Arguments

- X_centered A numeric matrix (n observations, p variables).
- Y_centered A numeric matrix (n observations, q variables).

Value

values.

safe_pca	<i>Safe PCA with Deterministic Bad-Column Repair</i>
----------	--

Description

Performs PCA after automatically detecting and fixing problematic columns (e.g., zero variance, NA-only, partial NAs) in a deterministic way. Ensures that the output dimensions match those from a clean dataset.

Usage

```
safe_pca(X, nc = min(dim(X)), center = TRUE, scale = TRUE)
```

Arguments

- X A numeric matrix or data frame.
- nc Number of components to return.
- center Logical, whether to center columns.
- scale Logical, whether to scale columns to unit variance.

Value

A list with the same structure as [prcomp](#), with rotation, sdev, x, etc.

Examples

```
set.seed(123)
mat <- matrix(rnorm(50), nrow = 10)
mat[, 3] <- 1 # zero variance col
mat[, 5] <- NA # all NA col
# result <- safe_pca(mat, nc = 3)
# result$rotation
```

save.ANTsR

*Save R sessions including ANTsR objects***Description**

Use this function to save an object or a full R session for later use. Any object can be saved (list, antsImage, data frame, etc.). The function will save all antsImage objects as nifti files with random filenames (to avoid using (sub)variable names, which can be identical). These files is what load.ANTsR uses later to fill back the original antsImage variables. For variables that contain filenames, you can choose to copy those files in your saved folder so you can completely recreate the environment even if you change computer or the temporary folder is deleted.

Usage

```
save.ANTsR(
  filename = file.path(".", ".ANTsRsession"),
  objects = NA,
  env = as.environment(1),
  overwrite = FALSE,
  clonediskfiles = TRUE,
  ...
)
```

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.
overwrite	logical to select whether overwriting existing data is allowed.
clonediskfiles	logical enables the copying of disk files that are not loaded in workspace but where character variables point. Set to TRUE if moving sessions from one computer to another or if you save antsRegistration() outputs. Set to FALSE if you save temporary sessions to continue later on the same computer.
...	Additional arguments to pass to save.

Author(s)

Dorian Pustina

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)
```

scaleSpaceFeatureDetection

Scale space feature detector

Description

Deploy a multiscale laplacian blob detector on an image. The function will return the blob descriptors as an image and data frame. NOTE: you will get different features running on the raw image versus its negation.

Usage

```
scaleSpaceFeatureDetection(
  image,
  numberOfBlobsToExtract,
  minScale,
  maxScale,
  stepsPerOctave = 10,
  negate = FALSE
)
```

Arguments

image	the input image
numberOfBlobsToExtract	the estimated blob count
minScale	the minimum amount of smoothing in scale space
maxScale	the maximum amount of smoothing in scale space
stepsPerOctave	the number of steps to take over scale space octaves
negate	boolean to compute features from both image and its negation

Value

list of antsImage and dataframe for blob descriptors

Author(s)

Avants BB

Examples

```
# WIP: should explore effect of parameters further
blob1 <- scaleSpaceFeatureDetection(ri(1), 50)
blob2 <- scaleSpaceFeatureDetection(max(ri(1)) - ri(1), 50)
```

sd	<i>SD generic</i>
----	-------------------

Description

Calculates the SD of an image
Overloaded SD for antsImage objects

Usage

```
sd(x, na.rm = FALSE, ...)
```

Default S3 method:

```
sd(x, na.rm = FALSE, ...)
```

S3 method for class 'antsImage'

```
sd(x, na.rm = FALSE, ..., mask = NULL)
```

Arguments

- x an object for which we want to compute the SD
- na.rm a logical value indicating whether NA should be removed
- ... Any additional arguments to be passed to [sd](#)
- mask is an object of class antsImage

Examples

```
img <- antsImageRead(getANTSRData("r16"))
sd(img)
sd(img, mask = img > 0)
```

segmentationRefinement.predict

Segmentation refinement using corrective learning (prediction)

Description

A random forest implementation of the corrective learning wrapper introduced in Wang, et al., Neuroimage 2011 (<https://pubmed.ncbi.nlm.nih.gov/21237273>). The prediction process involves using the label-specific training models to refine an initial segmentation.

Usage

```
segmentationRefinement.predict(
    segmentationImage,
    labelSet,
    labelModels,
    featureImages,
    featureImageNames,
    dilationRadius = 2,
    neighborhoodRadius = 0,
    normalizeSamplesPerLabel = TRUE,
    useEntireLabeledRegion = TRUE
)
```

Arguments

segmentationImage	image to refine via corrective learning.
labelSet	a vector specifying the labels of interest. Must be specified.
labelModels	a list of models. Each element of the labelSet requires a model.
featureImages	a list of feature images.
featureImageNames	is a vector of character strings naming the set of features. Must be specified.
dilationRadius	specifies the dilation radius for determining the ROI for each label using binary morphology. Alternatively, the user can specify a float distance value, e.g., "dilationRadius = '2.75mm'", to employ an isotropic dilation based on physical distance. For the latter, the distance value followed by the character string 'mm' (for millimeters) is necessary.
neighborhoodRadius	specifies which voxel neighbors should be included in prediction. The user can specify a scalar or vector but it must match with what was used for training.
normalizeSamplesPerLabel	if TRUE, the samples from each ROI are normalized by the mean of the voxels in that ROI. Can be a vector (one element per feature).

useEntireLabeledRegion

if TRUE, estimation is performed on the full dilated ROI for each label. If FALSE, estimation is performed on the combined inner and outer boundary region determined by the neighborhoodRadius parameter. Can also specify as a vector to determine per label.

Value

a list consisting of the refined segmentation estimate (RefinedSegmentationImage) and a list of the foreground probability images (ForegroundProbabilityImages).

Author(s)

Tustison NJ

Examples

```
## Not run:

library(ANTsR)
library(ggplot2)

imageIDs <- c("r16", "r27", "r30", "r62", "r64", "r85")

# Perform simple 3-tissue segmentation. For convenience we are
# going to use atropos segmentation to define the "ground-truth"
# segmentations and the kmeans to define the segmentation we
# want to "correct". We collect feature images for each image.
# The gradient and laplacian images chosen below as feature
# images are simply selected for convenience.

segmentationLabels <- c(1, 2, 3)

featureImageNames <- c("T1", "Gradient", "Laplacian")

images <- list()
kmeansSegs <- list()
atroposSegs <- list()
featureImages <- list()

for (i in 1:length(imageIDs))
{
  cat("Processing image", imageIDs[i], "\n")
  images[[i]] <- antsImageRead(getANTsRData(imageIDs[i]))
  mask <- getMask(images[[i]])
  kmeansSegs[[i]] <- kmeansSegmentation(images[[i]],
    length(segmentationLabels), mask,
    mrf = 0.0
  )$segmentation
  atroposSegs[[i]] <- atropos(images[[i]], mask,
    i = "KMeans[3]",
    m = "[0.25,1x1]", c = "[5,0]"
  )
}
```

```

)$segmentation

featureImageSetPerImage <- list()
featureImageSetPerImage[[1]] <- images[[i]]
featureImageSetPerImage[[2]] <- iMath(images[[i]], "Grad", 1.0)
featureImageSetPerImage[[3]] <- iMath(images[[i]], "Laplacian", 1.0)
featureImages[[i]] <- featureImageSetPerImage
}

# Perform training. We train on images "r27", "r30",
# "r62", "r64", "r85" and
# test/predict on image "r16".

cat("\nTraining\n\n")

segLearning <- segmentationRefinement.train(
  featureImages = featureImages[2:6],
  truthLabelImages = atroposSegs[2:6], segmentationImages = kmeansSegs[2:6],
  featureImageNames = featureImageNames, labelSet = segmentationLabels,
  maximumNumberOfSamplesOrProportionPerClass = 100, dilationRadius = 1,
  neighborhoodRadius = c(1, 1), normalizeSamplesPerLabel = TRUE,
  useEntireLabeledRegion = FALSE
)

cat("\nPrediction\n\n")

refinement <- segmentationRefinement.predict(
  segmentationImage = kmeansSegs[[1]], labelSet = segmentationLabels,
  segLearning$LabelModels, featureImages[[1]], featureImageNames,
  dilationRadius = 1, neighborhoodRadius = c(1, 1),
  normalizeSamplesPerLabel = TRUE
)

# Compare "ground truth" = atroposSegs[[1]] with
# refinement$RefinedSegmentationImage

antsImageWrite(
  refinement$RefinedSegmentationImage,
  "r16RefinedSegmentation.nii.gz"
)

## End(Not run)

```

segmentationRefinement.train

Segmentation refinement using corrective learning (training)

Description

A random forest implementation of the corrective learning wrapper introduced in Wang, et al., Neuroimage 2011 (<https://pubmed.ncbi.nlm.nih.gov/21237273>). The training process involves building

two sets of models from training data for each label in the initial segmentation data.

Usage

```
segmentationRefinement.train(
  featureImages,
  truthLabelImages,
  segmentationImages,
  featureImageNames = c(),
  labelSet = c(),
  maximumNumberOfSamplesOrProportionPerClass = 1,
  dilationRadius = 2,
  neighborhoodRadius = 0,
  normalizeSamplesPerLabel = TRUE,
  useEntireLabeledRegion = TRUE
)
```

Arguments

- featureImages** a list of lists of feature images. Each list of feature images corresponds to a single subject. Possibilities are outlined in the above-cited paper.
- truthLabelImages** a list of "ground-truth" segmentation images, one for each set of feature images.
- segmentationImages** a list of estimated segmentation images, one for each set of feature images.
- featureImageNames** a vector of character strings naming the set of features. This parameter is optional but does help in investigating the relative importance of specific features.
- labelSet** a vector specifying the labels of interest. If not specified, the full set is determined from the truthLabelImages.
- maximumNumberOfSamplesOrProportionPerClass** specified the maximum number of samples used to build the model for each element of the labelSet. If ≤ 1 , we use it as as a proportion of the total number of voxels.
- dilationRadius** specifies the dilation radius for determining the ROI for each label using binary morphology. Alternatively, the user can specify a float distance value, e.g., "dilationRadius = '2.75mm'", to employ an isotropic dilation based on physical distance. For the latter, the distance value followed by the character string 'mm' (for millimeters) is necessary.
- neighborhoodRadius** specifies which voxel neighbors should be included in building the model. The user can specify a scalar or vector.
- normalizeSamplesPerLabel** if TRUE, the samples from each ROI are normalized by the mean of the voxels in that ROI. Can also specify as a vector to normalize per feature image.

useEntireLabeledRegion

if TRUE, samples are taken from the full dilated ROI for each label. If FALSE, samples are taken only from the combined inner and outer boundary region determined by the neighborhoodRadius parameter. Can also specify as a vector to determine per label.

Value

list with the models per label (LabelModels), the label set (LabelSet), and the feature image names (FeatureImageNames).

Author(s)

Tustison NJ

Examples

```
## Not run:

library(ANTsR)
library(ggplot2)

imageIDs <- c("r16", "r27", "r30", "r62", "r64", "r85")

# Perform simple 3-tissue segmentation. For convenience we are
# going to use atropos segmentation to define the "ground-truth"
# segmentations and the kmeans to define the segmentation we
# want to "correct". We collect feature images for each image.
# The gradient and laplacian images chosen below as feature
# images are simply selected for convenience.

segmentationLabels <- c(1, 2, 3)

featureImageNames <- c("T1", "Gradient", "Laplacian")

images <- list()
kmeansSegs <- list()
atroposSegs <- list()
featureImages <- list()

for (i in 1:length(imageIDs))
{
  cat("Processing image", imageIDs[i], "\n")
  images[[i]] <- antsImageRead(getANTsRData(imageIDs[i]))
  mask <- getMask(images[[i]])
  kmeansSegs[[i]] <- kmeansSegmentation(images[[i]],
    length(segmentationLabels), mask,
    mrf = 0.0
  )$segmentation
  atroposSegs[[i]] <- atropos(images[[i]], mask,
    i = "KMeans[3]",
    m = "[0.25,1x1]", c = "[5,0]"
  )
}
```

```

)$segmentation

featureImageSetPerImage <- list()
featureImageSetPerImage[[1]] <- images[[i]]
featureImageSetPerImage[[2]] <- iMath(images[[i]], "Grad", 1.0)
featureImageSetPerImage[[3]] <- iMath(images[[i]], "Laplacian", 1.0)
featureImages[[i]] <- featureImageSetPerImage
}

# Perform training. We train on images "r27", "r30", "r62", "r64",
# "r85" and test/predict
# on image "r16".

cat("\nTraining\n\n")

segLearning <- segmentationRefinement.train(
  featureImages = featureImages[2:6],
  truthLabelImages = atroposSegs[2:6],
  segmentationImages = kmeansSegs[2:6],
  featureImageNames = featureImageNames, labelSet = segmentationLabels,
  maximumNumberOfSamplesOrProportionPerClass = 100, dilationRadius = 1,
  normalizeSamplesPerLabel = TRUE, useEntireLabeledRegion = FALSE
)

## End(Not run)

```

segmentationToOneHot *One-hot encoding function*

Description

Basic one-hot transformation of segmentations array.

Usage

```

segmentationToOneHot(
  segmentationsArray,
  segmentationLabels = NULL,
  channelFirstOrdering = FALSE
)

```

Arguments

segmentationsArray
multi-label numpy array

segmentationLabels
vector of segmentation labels. Note that a background label (typically 0) needs to be included.

channelFirstOrdering
Specifies the ordering of the dimensions.

Value

an n-D array of shape where the segmentation labels are expanded in a one-hot fashion in the channels dimension.

Author(s)

Tustison NJ

Examples

```
r16 <- antsImageRead( getANTsRData( "r16" ) )
seg <- kmeansSegmentation( r16, 3 )$segmentation
oneHot <- segmentationToOneHot( as.array( seg ) )

r16 <- antsImageRead( getANTsRData( "r16" ) )
seg <- kmeansSegmentation( r16, 3 )$segmentation
oneHot <- segmentationToOneHot( as.array( seg ),
                                channelFirstOrdering = TRUE )
```

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 = NULL,
  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)
```

select_joint_k	<i>Select an Optimal Joint Number of Components for Multi-View Data</i>
----------------	---

Description

This function determines a single optimal number of components (k) to represent a multi-view dataset by analyzing the joint variance explained curve. It offers multiple methods for selecting k and includes a built-in self-test to verify its own correctness.

Usage

```
select_joint_k(
  mat_list = NULL,
  method = c("pca", "spca"),
  k_max = NULL,
  sparsity = 0.5,
  selection_method = c("elbow", "threshold"),
  variance_threshold = 0.9,
  self_test = FALSE
)
```

Arguments

mat_list	A list of numeric matrices [subjects x features]. Required unless 'self_test = TRUE'.
method	The decomposition method. One of "pca" (fast SVD-based) or "spca".
k_max	The maximum number of components to consider.
sparsity	A single sparsity parameter (0-1) for SPCA. Ignored for PCA.

selection_method	The method for choosing k. One of "elbow" (point of maximum deviation from a straight line of improvement) or "threshold".
variance_threshold	The proportion of variance (0-1) to be explained. Only used when 'selection_method = "threshold"'.
self_test	Logical. If TRUE, the function will ignore all other inputs, run a built-in suite of tests on simulated data, and print the results. This is for verifying the function's integrity. Defaults to FALSE.

Value

If 'self_test = FALSE', a list containing:

- 'optimal_k': The selected optimal number of components.
- 'joint_variance_curve': A tibble with 'k' and the cumulative proportion of variance explained at each 'k'.
- 'plot': A ggplot object visualizing the results.

If 'self_test = TRUE', it prints test results and returns invisibly.

setANTsDeterministic *Set ANTs deterministic behavior*

Description

Set ANTs deterministic behavior

Usage

```
setANTsDeterministic(on = TRUE, seedValue = 123)
```

Arguments

on	Set deterministic behavior.
seedValue	Assign random seed value.

Value

No return value, called for side effects.

Examples

```
setANTsDeterministic(TRUE)
```

setAntsrTransformFixedParameters

setAntsrTransformFixedParameters

Description

Set parameters of transform

Usage

```
setAntsrTransformFixedParameters(transform, parameters)
```

Arguments

transform	antsrTransform
parameters	array of parameters'

Value

TRUE

Examples

```
tx <- new("antsrTransform")
params <- getAntsrTransformFixedParameters(tx)
setAntsrTransformFixedParameters(tx, params * 2)
```

setAntsrTransformParameters

setAntsrTransformParameters

Description

Set parameters of transform

Usage

```
setAntsrTransformParameters(transform, parameters)
```

Arguments

transform	antsrTransform
parameters	array of parameters

Value

TRUE

Examples

```
tx <- new("antsrTransform")
params <- getAntsrTransformParameters(tx)
setAntsrTransformParameters(tx, params * 2)
```

setSeedBasedOnTime	<i>Set Seed Based on Current Time</i>
--------------------	---------------------------------------

Description

This function sets the random number generator seed based on the current time, with a fine resolution of seconds. This ensures a different seed is used each time the function is called, provided calls are at least one second apart.

Usage

```
setSeedBasedOnTime()
```

Value

The numeric value used as the seed, derived from the current time in seconds.

Examples

```
seedValue <- setSeedBasedOnTime()
print(seedValue)
```

shorten_pymm_names	<i>shorter antspymm names</i>
--------------------	-------------------------------

Description

shorter antspymm names

Usage

```
shorten_pymm_names(x)
```

Arguments

x	string or strings
---	-------------------

Value

string

Author(s)

Avants BB

simlr

*Similarity-driven multiview linear reconstruction model (simlr) for N modalities***Description**

simlr minimizes reconstruction error across related modalities. That is, simlr will reconstruct each modality matrix from a basis set derived from the other modalities. The basis set can be derived from SVD, ICA or a simple sum of basis representations. This function produces dataset-wide multivariate beta maps for each of the related matrices. The multivariate beta maps are regularized by user input matrices that encode relationships between variables. The idea is overall similar to canonical correlation analysis but generalizes the basis construction and to arbitrary numbers of modalities.

Usage

```
simlr(
  data_matrices,
  smoothingMatrices,
  iterations = 500,
  sparsenessQuantiles = NULL,
  positivities = NULL,
  initialUMatrix = NULL,
  mixAlg = c("svd", "ica", "avg", "rrpca-l", "rrpca-s", "pca", "stochastic",
    "newton-schulz", "ica-newton"),
  repeatedMeasures = NA,
  lineSearchRange = c(-500, 500),
  lineSearchTolerance = 1e-12,
  randomSeed = 0,
  constraint = c("nsaflowx0.5x10", "orthox0x1", "Grassmannx0", "Stiefelx0", "none"),
  energyType = c("cca", "regression", "normalized", "ucca", "lowRank",
    "lowRankRegression", "normalized_correlation", "acc", "nc", "dat", "lrr", "reconorm",
    "logcosh", "exp", "kurtosis", "gauss"),
  vmats = NULL,
  connectors = NULL,
  optimizationStyle = c("bidirectional_lookahead", "armijo_gradient", "lookahead",
    "bidirectional_armijo_gradient"),
  scale = c("center", "eigenvalue"),
  expBeta = 0.9,
  jointInitialization = TRUE,
```



```

    sparsenessAlg = "soft",
    orthogonalizeU = FALSE,
    domainMatrices = NULL,
    domainLambdas = NULL,
    sparse_gradient = TRUE,
    verbose = FALSE
)

```

Arguments

<code>data_matrices</code>	A list that contains the named matrices. Note: the optimization will likely perform much more smoothly if the input matrices are each scaled to zero mean unit variance e.g. by the <code>scale</code> function.
<code>smoothingMatrices</code>	list of (sparse) matrices that allow parameter smoothing/regularization. These should be square and same order and size of input matrices.
<code>iterations</code>	number of gradient descent iterations
<code>sparsenessQuantiles</code>	vector of quantiles to control sparseness - higher is sparser
<code>positivities</code>	vector that sets for each matrix if we restrict to positive or negative solution (beta) weights. choices are positive, negative or either as expressed as a string.
<code>initialUMatrix</code>	list of initialization matrix size n by k for each modality. Otherwise, pass a single scalar to control the number of basis functions in which case random initialization occurs. One may also pass a single initialization matrix to be used for all matrices. If this is set to a scalar, or is missing, a random matrix will be used.
<code>mixAlg</code>	'svd', 'ica', 'rrpca-l', 'rrpca-s', 'stochastic', 'pca', 'newton-schulz', 'ica-newton' or 'avg' denotes the algorithm employed when estimating the mixed modality bases
<code>repeatedMeasures</code>	list of repeated measurement identifiers. this will allow estimates of per identifier intercept.
<code>lineSearchRange</code>	lower and upper limit used in optimize
<code>lineSearchTolerance</code>	tolerance used in optimize, will be multiplied by each matrix norm such that it scales appropriately with input data
<code>randomSeed</code>	controls repeatability of ica-based decomposition
<code>constraint</code>	one of none, Grassmann, GrassmannInv or Stiefel
<code>energyType</code>	one of regression, normalized, lowRank, cca or ucca
<code>vmats</code>	optional initial v matrix list
<code>connectors</code>	a list (length of projections or number of modalities) that indicates which modalities should be paired with current modality
<code>optimizationStyle</code>	the simlr optimizer

scale	options to standardize each matrix. e.g. divide by the square root of its number of variables (Westerhuis, Kourti, and MacGregor 1998), divide by the number of variables or center or center and scale or ... (see code). can be a vector which will apply each strategy in order.
expBeta	if greater than zero, use exponential moving average on gradient and mixing (default 0.9).
jointInitialization	boolean for initialization options, default TRUE
sparsenessAlg	NA is default otherwise basic, smp or orthorank
orthogonalizeU	boolean controlling whether we orthogonalize the U matrices
domainMatrices	matrices containing domain knowledge length of data_matrices with number of columns also equal to each corresponding data matrix
domainLambdas	weights for domain knowledge term length of data_matrices
sparse_gradient	boolean default TRUE
verbose	boolean to control verbosity of output - set to level 2 in order to see more output, specifically the gradient descent parameters.

Value

A list of u, x, y, z etc related matrices.

Author(s)

BB Avants.

See Also

[milr](#) [mild](#) [simlrU](#)

Examples

```
## Not run:
set.seed(1500)
nsub <- 25
npix <- c(100, 200, 133)
nk <- 5
outcome <- matrix(rnorm(nsub * nk), ncol = nk)
outcome1 <- matrix(rnorm(nsub * nk), ncol = nk)
outcome2 <- matrix(rnorm(nsub * nk), ncol = nk)
outcome3 <- matrix(rnorm(nsub * nk), ncol = nk)
view1tx <- matrix(rnorm(npix[1] * nk), nrow = nk)
view2tx <- matrix(rnorm(npix[2] * nk), nrow = nk)
view3tx <- matrix(rnorm(npix[3] * nk), nrow = nk)
mat1 <- (outcome %*% t(outcome1) %*% (outcome1)) %*% view1tx
mat2 <- (outcome %*% t(outcome2) %*% (outcome2)) %*% view2tx
mat3 <- (outcome %*% t(outcome3) %*% (outcome3)) %*% view3tx
matlist <- list(vox = mat1, vox2 = mat2, vox3 = mat3)
```

```

result <- simlr(matlist)
p1 <- mat1 %*% (result$v[[1]])
p2 <- mat2 %*% (result$v[[2]])
p3 <- mat3 %*% (result$v[[3]])
regs <- regularizeSimlr(matlist)
result2 <- simlr(matlist)
pred1 <- predictSimlr(matlist, result)
pred2 <- predictSimlr(matlist, result2)

# compare to permuted data
s1 <- sample(1:nsub)
s2 <- sample(1:nsub)
resultp <- simlr(list(vox = mat1, vox2 = mat2[s1, ], vox3 = mat3[s2, ]))
p1p <- mat1 %*% (resultp$v[[1]])
p2p <- mat2[s1, ] %*% (resultp$v[[2]])
p3p <- mat3[s2, ] %*% (resultp$v[[3]])

# compare to SVD
svd1 <- svd(mat1, nu = nk, nv = 0)$u
svd2 <- svd(mat2, nu = nk, nv = 0)$u
svd3 <- svd(mat3, nu = nk, nv = 0)$u

# real
range(cor(p1, p2))
range(cor(p1, p3))
range(cor(p3, p2))

# permuted
range(cor(p1p, p2p))
range(cor(p1p, p3p))
range(cor(p3p, p2p))

# svd
print(range(cor(svd1, svd2)))

resultp <- simlr(list(vox = mat1, vox2 = mat2[s1, ], vox3 = mat3[s2, ]),
  initialUMatrix = nk, verbose = TRUE, iterations = 5,
  energyType = "normalized"
)

## End(Not run)

```

simlr.parameters

Generate Parameter Grid for SIMLR Search

Description

This function creates and optionally subsets all possible combinations of input parameters for use in a SIMLR grid search.

Usage

```
simlr.parameters(
  nsimlr_options,
  prescaling_options,
  objectiver_options,
  mixer_options,
  sparval_options,
  expBeta_options,
  positivites_options,
  optimus_options,
  constraint_options = list("none"),
  sparsenessAlg = list("soft"),
  num_samples = 10,
  search_type = c("random", "deterministic", "full")
)
```

Arguments

`nsimlr_options` A list of options for the ‘nsimlr’ parameter.

`prescaling_options` A list of options for the ‘prescaling’ parameter.

`objectiver_options` A list of options for the ‘objectiver’ parameter.

`mixer_options` A list of options for the ‘mixer’ parameter.

`sparval_options` A list of options for the ‘sparval’ parameter.

`expBeta_options` A list of options for the ‘ebber’ parameter.

`positivities_options` A list of options for the ‘pizzer’ parameter.

`optimus_options` A list of options for the ‘optimus’ parameter.

`constraint_options` A list of options for the ‘constraint’ parameter, default is ‘list("none")’.

`sparsenessAlg` A list of options for the ‘sparsenessAlg’ parameter, default is ‘list(NA)’.

`num_samples` if not full, then the size of the subset space

`search_type` string vector either full random or deterministic

Value

A list containing all (or a subset of) combinations of the provided parameters. Each row in the data frame represents a unique combination of the parameters.

The columns of the returned data frame include:

- `nsimlr`: Values corresponding to the ‘nsimlr’ parameter.
- `prescaling`: Values corresponding to the ‘prescaling’ parameter.

- `objectiver`: Values corresponding to the ‘objectiver’ parameter.
- `mixer`: Values corresponding to the ‘mixer’ parameter.
- `constraint`: Values corresponding to the ‘constraint’ parameter.
- `sparval`: Values corresponding to the ‘sparval’ parameter.
- `ebber`: Values corresponding to the ‘ebber’ parameter.
- `pizzer`: Values corresponding to the ‘pizzer’ parameter.
- `optimus`: Values corresponding to the ‘optimus’ parameter.

Each row represents a specific set of parameters that can be passed to the ‘simlr.search’ function for evaluation. The returned data frame is intended for use in parameter tuning and optimization.

simlr.perm

Assess Significance of SiMLR Call

Description

This function performs permutation tests to assess the significance of a SiMLR analysis. For more detail on input parameters, see the original function.

Usage

```
simlr.perm(
  data_matrices,
  smoothingMatrices = NULL,
  iterations = 10,
  sparsenessQuantiles = NULL,
  positivities = NULL,
  initialUMatrix = NULL,
  mixAlg = c("svd", "ica", "avg", "rrpca-l", "rrpca-s", "pca", "stochastic"),
  orthogonalizeU = FALSE,
  repeatedMeasures = NA,
  lineSearchRange = c(-500, 500),
  lineSearchTolerance = 1e-12,
  randomSeed = 0,
  constraint = c("nsaflowx0.5x10", "orthox0.001x1", "Grassmannx0", "Stiefelx0", "none"),
  energyType = c("cca", "regression", "normalized", "acc", "dat", "lowRank",
    "lowRankRegression", "normalized_correlation", "logcosh", "exp", "kurtosis", "gauss"),
  vmats = NULL,
  connectors = NULL,
  optimizationStyle = "adam",
  scale = c("centerAndScale", "eigenvalue"),
  expBeta = 0,
  jointInitialization = TRUE,
  sparsenessAlg = NA,
  verbose = FALSE,
```

```

    nperms = 50,
    FUN = adjusted_rvcoef,
    cores = 1
)

```

Arguments

<code>data_matrices</code>	A list of voxel matrices.
<code>smoothingMatrices</code>	A list of smoothing matrices.
<code>iterations</code>	Number of iterations. Default is 10.
<code>sparsenessQuantiles</code>	A vector of sparseness quantiles.
<code>positivities</code>	A vector of positivity constraints.
<code>initialUMatrix</code>	Initial U matrix for the algorithm.
<code>mixAlg</code>	The mixing algorithm to use. Default is 'svd'.
<code>orthogonalizeU</code>	Logical indicating whether to orthogonalize U. Default is FALSE.
<code>repeatedMeasures</code>	Repeated measures data. Default is NA.
<code>lineSearchRange</code>	Range for line search. Default is c(-1e+10, 1e+10).
<code>lineSearchTolerance</code>	Tolerance for line search. Default is 1e-08.
<code>randomSeed</code>	Seed for random number generation.
<code>constraint</code>	The constraint type. Default is 'none'.
<code>energyType</code>	The energy type. Default is 'cca'.
<code>vmats</code>	List of V matrices - optional initialization matrices
<code>connectors</code>	List of connectors. Default is NULL.
<code>optimizationStyle</code>	The simlr optimizer.
<code>scale</code>	Scaling method. Default is 'centerAndScale'.
<code>expBeta</code>	Exponential beta value. Default is 0.9.
<code>jointInitialization</code>	Logical indicating joint initialization. Default is TRUE.
<code>sparsenessAlg</code>	Sparseness algorithm. Default is NA.
<code>verbose</code>	Logical indicating whether to print verbose output. Default is FALSE. values > 1 lead to more verbosity
<code>nperms</code>	Number of permutations for significance testing. Default is 50.
<code>FUN</code>	function for summarizing variance explained
<code>cores</code>	number of cores to use

Value

A data frame containing p-values for each permutation.

simlr.search

Perform Parallel SIMLR Grid Search (with Progress Bar)

Description

This version runs parameter sets in parallel using all available CPU cores (minus one by default), and shows a progress bar with ETA.

Usage

```
simlr.search(
  mats,
  regs,
  options_df,
  maxits = 100,
  connectors = NULL,
  nperms = 1,
  verbose = 0,
  FUN = rvcoef,
  cores = 1
)
```

Arguments

mats	List of input matrices for SIMLR.
regs	List of regularization parameters for SIMLR.
options_df	Data frame of parameter combinations.
maxits	Maximum iterations for SIMLR.
connectors	List of connectors for modalities.
nperms	Number of permutations for significance test.
verbose	Verbosity level.
FUN	Evaluation function.
cores	Number of CPU cores to use (default: detectCores() - 1). parallel::detectCores() - 1

Value

List with best SIMLR result, significance, best parameters, and search table.

simlr_feature_orth	<i>Compute the Average Invariant Orthogonality Defect for Multiple Features</i>
--------------------	---

Description

This function calculates the average invariant orthogonality defect for a list of features, using the ‘invariant_orthogonality_defect’ function. The orthogonality defect measures the deviation of feature vectors from orthogonality.

Usage

```
simlr_feature_orth(p)
```

Arguments

p	A list of matrices (or data frames), each representing a set of feature vectors.
---	--

Details

The function iterates over each matrix in the list ‘p’ and applies the ‘invariant_orthogonality_defect’ function to calculate the orthogonality defect for each matrix. The final result is the average orthogonality defect across all matrices.

Value

A numeric value representing the average invariant orthogonality defect across all the provided feature matrices.

Examples

```
# Example data: Two sets of feature matrices
feature1 <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2)
feature2 <- matrix(c(7, 8, 9, 10, 11, 12), nrow = 2)
feature_list <- list(feature1, feature2)

# Calculate the average orthogonality defect
avg_orthogonality_defect <- simlr_feature_orth(feature_list)
print(avg_orthogonality_defect)
```

simlr_impute	<i>Impute missing SiMLR data in a specified column based on other columns</i>
--------------	---

Description

Impute missing SiMLR data in a specified column based on other columns

Usage

```
simlr_impute(
  dataframe,
  nms,
  vecnum,
  toimpute,
  separator = "PC",
  family = "gaussian"
)
```

Arguments

dataframe	A data frame containing the data to impute.
nms	A vector of base column names.
vecnum	A numeric value to append to the column names.
toimpute	The base name of the target column to be imputed.
separator	A string specifying the separator between the column name and feature.
family	A string specifying the GLM family (default is 'gaussian').

Value

A data frame with imputed values.

Examples

```
set.seed(123)
n=50
df <- data.frame(
  t1PC1 = rnorm(n),
  t1aPC1 = rnorm(n),
  dtPC1 = rnorm(n),
  dtaPC1 = rnorm(n),
  rsfPC1 = rnorm(n),
  perfPC1 = rnorm(n)
)
df[ sample(1:nrow(df),20),6 ]=NA
nms <- c("t1", "t1a", "dt", "dta", "rsf", "perf")
vecnum <- 1
```

```
toimpute <- "perf"
df = simlr_impute(df, nms, vecnum, toimpute, family = 'gaussian')
```

simlr_path_models	<i>SiMLR Path Models helper</i>
-------------------	---------------------------------

Description

Creates the list that should be passed to the connectors argument to simlr. the different options are default (0) which connects each to all others but not itself, a single integer > 0 which forces modeling to focus on a given target and then a pca like option.

Usage

```
simlr_path_models(n, type = 0)
```

Arguments

n	The length of the output list.
type	The type of modeling. Can be one of the following: - 0: "Exclude self" - each entry contains all integers except the one corresponding to its position. - Integer greater than 0: "focus on a target" - each entry contains the target integer, except for the entry at the target position, which contains all integers except the target. - 'pca': "Identity" - each entry contains its own position number.

Value

A list of length n with contents based on the input type.

simlr_sparseness	<i>Apply SiMLR Sparseness and Normalize Features</i>
------------------	--

Description

Applies smoothing and sparsity constraints to a matrix or vector as used in SiMLR or manifold learning, followed by optional L1 feature normalization depending on the energy type.

Usage

```
simlr_sparseness(
  v,
  constraint_type = c("none", "ortho", "nsaflow"),
  smoothing_matrix = NULL,
  positivity = "positive",
  sparseness_quantile = 0.8,
  constraint_weight = NA,
  constraint_iterations = 1,
  sparseness_alg = "soft",
  energy_type = "acc"
)
```

Arguments

v A numeric matrix or vector to be transformed.

constraint_type Character. Type of manifold constraint; one of "nsaflow" or "None".

smoothing_matrix Optional numeric matrix. If provided, v is left-multiplied by this matrix.

positivity Character positive, negative or either.

sparseness_quantile Numeric between 0 and 1. Fraction of elements to sparsify using quantile thresholding.

constraint_weight Numeric. Weight for the constraint, used in orthogonalization.

constraint_iterations Numeric. Number of iterations for the orthogonalization optimization.

sparseness_alg Character. Sparsity algorithm to use (relevant for Stiefel and Grassmann). . ensemble or nnorth are test options.

energy_type Character. If set to one of "acc", "cca", "nc", "normalized_correlation", "lowRankRegression", or "lrr", then the returned matrix is normalized using l1_normalize_features.

Value

A numeric matrix of the same dimensions as v, with applied smoothing, sparsity, and optional normalization.

Description

simlr minimizes reconstruction error across related modalities. One crucial component of the reconstruction is the low-dimensional cross-modality basis. This function computes that basis, given a mixing algorithm.

Usage

```
simlrU(
  projections,
  mixingAlgorithm,
  initialW,
  orthogonalize = FALSE,
  connectors = NULL,
  expBeta = 0
)
```

Arguments

projections	A list that contains the low-dimensional projections.
mixingAlgorithm	the elected mixing algorithm. see <code>simlr</code> . can be 'svd', 'ica', 'rrpca-l', 'rrpca-s', 'pca', 'stochastic' or 'avg'.
initialW	initialization matrix size n by k for fastICA.
orthogonalize	boolean
connectors	a list (length of projections or number of modalities) that indicates which modalities should be paired with current modality
expBeta	exponential moving average beta

Value

u matrix for modality i

Author(s)

BB Avants.

See Also

[simlr](#)

Examples

```
set.seed(1500)
nsub <- 25
npix <- c(100, 200, 133)
nk <- 5
outcome <- matrix(rnorm(nsub * nk), ncol = nk)
outcome1 <- matrix(rnorm(nsub * nk), ncol = nk)
```

```
outcome2 <- matrix(rnorm(nsub * nk), ncol = nk)
u <- simlrU(list(outcome, outcome1, outcome2), 2, "avg")
```

simulateBiasField	<i>Simulate random bias field</i>
-------------------	-----------------------------------

Description

Low frequency, spatial varying simulated random bias field using random points and B-spline fitting.

Usage

```
simulateBiasField(
  domainImage,
  numberOfPoints = 10,
  sdBiasField = 1,
  numberOfFittingLevels = 4,
  meshSize = 1
)
```

Arguments

domainImage	image to define the spatial domain of the bias field.
numberOfPoints	number of randomly defined points to define the bias field (default = 10).
sdBiasField	characterize the standard deviation of the amplitude (default = 1).
numberOfFittingLevels	B-spline fitting parameter.
meshSize	B-spline fitting parameter (scalar or vector of size image dimension).

Value

simulated bias field

Author(s)

Tustison NJ

Examples

```
library( ANTsR )
image <- antsImageRead( getANTsRData( "r16" ) )
logField <- simulateBiasField(image, numberOfPoints = 10, sdBiasField = 1.0,
  numberOfFittingLevels = 2, meshSize = 10 ) %>% iMath( "Normalize" )
logField <- ( exp( logField ) )^4
image <- image * logField
rm(image); gc()
rm(logField); gc()
```

```
simulateDisplacementField
    simulateDisplacementField
```

Description

Simulate a displacement field using random points with B-spline smoothing

Usage

```
simulateDisplacementField(
  domainImage,
  fieldType = c("bspline", "exponential"),
  numberOfRandomPoints = 1000,
  sdNoise = 10,
  enforceStationaryBoundary = TRUE,
  numberOfFittingLevels = 4,
  meshSize = 1,
  sdSmoothing = 4
)
```

Arguments

domainImage	image to define the domain of the field.
fieldType	either "bspline" or "exponential".
numberOfRandomPoints	number of displacement points. Default = 1000.
sdNoise	standard deviation of the displacement field noise (in mm). Default = 10.0.
enforceStationaryBoundary	boolean determining fixed boundary conditions. Default = TRUE.
numberOfFittingLevels	(bspline only) number of fitting levels. Default = 4.
meshSize	(bspline only) scalar or n-D vector determining fitting resolution. Default = 1.
sdSmoothing	(exponential only) standard deviation of the Gaussian smoothing in mm. Default = 4.0.

Value

ANTsR displacement field.

Author(s)

NJ Tustison

Examples

```
domainImage <- antsImageRead(getANTsRData("r16"), 2)
expField <- simulateDisplacementField(domainImage, fieldType = "exponential")
bsplineField <- simulateDisplacementField(domainImage, fieldType = "bspline")
warp <- antsrTransformFromDisplacementField(bsplineField * 3)
wimg <- applyAntsrTransformToImage(warp, domainImage, domainImage)
```

skeletonize*Skeletonize a binary segmentation.*

Description

Employ a topologically constrained label propagation tool in order to thin an existing segmentation into a skeleton. The function will perform better if the initial segmentation is well-composed.

Usage

```
skeletonize(x, laplacianThreshold = 0.25, propagationOption = 1)
```

Arguments

x	input binary image
laplacianThreshold	between zero and one, controls the speed image
propagationOption	either 0 (none), 1 (well-composed) or 2 (topological) constraint

Value

output binary image

Author(s)

Avants BB

Examples

```
img <- antsImageRead(getANTsRData("r16")) %>% thresholdImage("Otsu", 3)
skel <- skeletonize(thresholdImage(img, 3, 3))
```

smoothAppGradCCA	<i>cca via sparse smooth matrix prediction</i>
------------------	--

Description

This implements a sparse and graph-regularized version of CCA based on the AppGrad style of implementation by Ma, Lu and Foster, 2015.

Usage

```
smoothAppGradCCA(
  x,
  y,
  smoox = NA,
  smooy = NA,
  sparsenessQuantile = 0.5,
  positivity = "either",
  k = 2,
  iterations = 10,
  stochastic = NA,
  initialization = "randxy",
  verbose = FALSE
)
```

Arguments

x	input view 1 matrix
y	input view 2 matrix
smoox	smoothingMatrix for x
smooy	smoothingMatrix for y
sparsenessQuantile	quantile to control sparseness - higher is sparser
positivity	restrict to positive or negative solution (beta) weights. choices are positive, negative or either as expressed as a string.
k	number of basis vectors to compute
iterations	number of gradient descent iterations
stochastic	size of subset to use for stochastic gradient descent
initialization	type of initialization, currently only supports a character randxy
verbose	boolean option

Value

list with matrices each of size p or q by k

Author(s)

Avants BB

Examples

```
mat <- replicate(100, rnorm(20))
mat2 <- replicate(100, rnorm(20))
mat <- scale(mat)
mat2 <- scale(mat2)
wt <- 0.666
mat3 <- mat * wt + mat2 * (1 - wt)
jj <- smoothAppGradCCA(mat, mat3)
```

smoothImage	<i>Smooth image</i>
-------------	---------------------

Description

Smooth image or multi-channel image

Usage

```
smoothImage(
  inimg,
  sigma,
  sigmaInPhysicalCoordinates = TRUE,
  FWHM = FALSE,
  max_kernel_width = 70
)
```

Arguments

inimg	Image to smooth
sigma	Smoothing factor. Can be scalar, in which case the same sigma is applied to each dimension, or a vector of length dim(inimg) to specify a unique smoothness for each dimension.
sigmaInPhysicalCoordinates	If true, the smoothing factor is in millimeters; if false, it is in pixels.
FWHM	If true, sigma is interpreted as the full-width-half-max (FWHM) of the filter, not the sigma of a Gaussian kernel.
max_kernel_width	Maximum kernel width

Value

antsImage smoothed image

Author(s)

Kandel BM, Avants BB

Examples

```

set.seed(1234)
img <- makeImage(c(5, 5), rnorm(25))
simg <- smoothImage(img, c(1.2, 1.5))
testthat::expect_equal(sum(simg), -5.59352651424706)
testthat::expect_error(smoothImage(img, c(1.2, 1.5, 3)))
simg2 <- smoothImage(img, c(1.2, 1.5), FWHM = TRUE)

multi <- antsImageRead(getANTsRData("multi_component_image"))
smulti <- smoothImage(multi, c(1.2, 1.5))
means <- sapply(splitChannels(smulti), mean)
testthat::expect_equal(means, c(86.0794883092244, 81.913664855957, 85.519760093689))

```

smoothMatrixPrediction

smooth matrix prediction

Description

Reconstruct a n by p matrix given n by k basis functions or predictors. The reconstruction can be regularized. $\# \text{norm}(x - uv^t) \# d/dv \dots$ leads to $(-u, x - uv^t) \# u^t u v^t - u^t x$

Usage

```

smoothMatrixPrediction(
  x,
  basisDf,
  modelFormula = as.formula(" x ~ ."),
  iterations = 10,
  gamma = 1e-06,
  sparsenessQuantile = 0.5,
  positivity = c("positive", "negative", "either"),
  smoothingMatrix = NA,
  repeatedMeasures = NA,
  rowWeights = NA,
  LRR = NA,
  doOrth = FALSE,
  verbose = FALSE
)

```

Arguments

x	input matrix to be predicted.
basisDf	data frame for basis predictors
modelFormula	a formula object which has, on the left, the variable x and the prediction formula on the right.
iterations	number of gradient descent iterations
gamma	step size for gradient descent
sparsenessQuantile	quantile to control sparseness - higher is sparser.
positivity	restrict to positive or negative solution (beta) weights. choices are positive, negative or either as expressed as a string.
smoothingMatrix	allows parameter smoothing, should be square and same size as input matrix
repeatedMeasures	list of repeated measurement identifiers. this will allow estimates of per identifier intercept.
rowWeights	vectors of weights with size n (assumes diagonal covariance)
LRR	integer value sets rank for fast version exploiting matrix approximation
doOrth	boolean enforce gram-schmidt orthonormality
verbose	boolean option

Value

matrix of size p by k is output

Author(s)

Avants BB

Examples

```
## Not run:
mask <- getMask(antsImageRead(getANTsRData("r16")))
spatmat <- t(imageDomainToSpatialMatrix(mask, mask))
smoomat <- knnSmoothingMatrix(spatmat, k = 5, sigma = 1.0)
mat <- matrix(rnorm(sum(mask) * 50), ncol = sum(mask), nrow = 50)
mat[1:25, 100:10000] <- mat[1:25, 100:10000] + 1
age <- rnorm(1:nrow(mat))
for (i in c(5000:6000, 10000:11000, 16000:17000)) {
  mat[, i] <- age * 0.1 + mat[, i]
}
gen <- c(rep("M", 25), rep("F", 12), rep("T", 13))
repmeas <- rep(c("A", "B", "C", "D", "E", "F", "G"), nrow(mat))[1:nrow(mat)]
mydf <- data.frame(age = scale(age), gen = gen)
fit <- smoothMatrixPrediction(
  x = mat, basisDf = mydf, iterations = 10,
```

```

gamma = 1.e-6, sparsenessQuantile = 0.5,
smoothingMatrix = smoomat, repeatedMeasures = repmeas,
verbose = T
)
tt <- mat %*% fit$v
print(cor.test(mydf$age, tt[, 1]))
print(cor.test(fit$u[, "genM"], tt[, 2]))
vimg <- makeImage(mask, (fit$v[, 1]))
print(range(vimg) * 10)
plot(mask, vimg, window.overlay = range(abs(vimg)))
vimg <- makeImage(mask, (fit$v[, 2]))
print(range(vimg) * 10)
plot(mask, vimg, window.overlay = range(abs(vimg)))

## End(Not run)

```

smoothMultiRegression *Reconstruct a n by k vector given n by p matrix of predictors.*

Description

Reconstruct a n by k vector given n by p matrix of predictors.

Usage

```

smoothMultiRegression(
  x,
  y,
  iterations = 10,
  sparsenessQuantile = 0.5,
  positivity = FALSE,
  smoothingMatrixX = NA,
  smoothingMatrixY = NA,
  nv = 2,
  extraPredictors,
  verbose = FALSE
)

```

Arguments

x	input matrix on which prediction is based
y	target matrix
iterations	number of gradient descent iterations
sparsenessQuantile	quantile to control sparseness - higher is sparser.
positivity	restrict to positive or negative solution (beta) weights. choices are positive, negative or either as expressed as a string.

smoothingMatrixX	allows parameter smoothing, should be square and same size as input matrix
smoothingMatrixY	allows parameter smoothing, should be square and same size as input matrix
nv	number of predictor spatial vectors
extraPredictors	additional column predictors
verbose	boolean option

Value

vector of size p is output

Author(s)

Avants BB

Examples

```
## Not run:
mask <- getMask(antsImageRead(getANTsRData("r16")))
spatmat <- t(imageDomainToSpatialMatrix(mask, mask))
smoomat <- knnSmoothingMatrix(spatmat, k = 200, sigma = 1.0)
mat <- matrix(rnorm(sum(mask) * 50), ncol = sum(mask), nrow = 50)
mat[1:25, 100:10000] <- mat[1:25, 100:10000] + 1
age <- matrix(rnorm(nrow(mat) * 2), ncol = 2)
for (i in c(5000:6000, 10000:11000, 16000:17000)) {
  mat[, i] <- age[, 1] * 0.1 + mat[, i]
}
sel <- 1:25
fit <- smoothMultiRegression(
  x = mat[sel, ], y = age[sel, ], iterations = 10,
  sparsenessQuantile = 0.5,
  smoothingMatrixX = smoomat, smoothingMatrixY = NA, verbose = T
)
tt <- mat %*% fit$v
print(cor.test(age[-sel, 1], tt[-sel, 1]))
vimg <- makeImage(mask, (fit$v[, 1]))
print(range(vimg) * 10)
plot(mask, vimg, window.overlay = range(abs(vimg)))

## End(Not run)
```

smoothRegression

smooth matrix-based regression

Description

Reconstruct a n by 1 vector given n by p matrix of predictors.

Usage

```
smoothRegression(
  x,
  y,
  iterations = 10,
  sparsenessQuantile = 0.5,
  positivity = FALSE,
  smoothingMatrix = NA,
  nv = 2,
  extraPredictors,
  verbose = FALSE
)
```

Arguments

x	input matrix on which prediction is based
y	target vector
iterations	number of gradient descent iterations
sparsenessQuantile	quantile to control sparseness - higher is sparser.
positivity	restrict to positive or negative solution (beta) weights. choices are positive, negative or either as expressed as a string.
smoothingMatrix	allows parameter smoothing, should be square and same size as input matrix
nv	number of predictor spatial vectors
extraPredictors	additional column predictors
verbose	boolean option

Value

vector of size p is output

Author(s)

Avants BB

Examples

```
## Not run:
mask <- getMask(antsImageRead(getANTsRData("r16")))
spatmat <- t(imageDomainToSpatialMatrix(mask, mask))
smoomat <- knnSmoothingMatrix(spatmat, k = 200, sigma = 1.0)
mat <- matrix(rnorm(sum(mask) * 50), ncol = sum(mask), nrow = 50)
mat[1:25, 100:10000] <- mat[1:25, 100:10000] + 1
age <- rnorm(1:nrow(mat))
for (i in c(5000:6000, 10000:11000, 16000:17000)) {
```

```
mat[, i] <- age * 0.1 + mat[, i]
}
sel <- 1:25
fit <- smoothRegression(
  x = mat[sel, ], y = age[sel], iterations = 10,
  sparsenessQuantile = 0.5,
  smoothingMatrix = smoomat, verbose = T
)
tt <- mat %*% fit$v
print(cor.test(age[-sel], tt[-sel, 1]))
vimg <- makeImage(mask, (fit$v[, 1]))
print(range(vimg) * 10)
plot(mask, vimg, window.overlay = range(abs(vimg)))

## End(Not run)
```

softwareVersions	<i>softwareVersions</i>
------------------	-------------------------

Description

softwareVersions

sparseDecom	<i>Convenience wrapper for eigenanatomy decomposition.</i>
-------------	--

Description

Decomposes a matrix into sparse eigenvectors to maximize explained variance. Note: we do not scale the matrices internally. We leave scaling choices to the user.

Usage

```
sparseDecom(
  inmatrix = NA,
  inmask = NULL,
  sparseness = 0.1,
  nvecs = 10,
  its = 5,
  cthresh = 50,
  statdir = NA,
  z = 0,
  smooth = 0,
  initializationList = list(),
  mycoption = 0,
  robust = 0,
```

```

    ell1 = 1,
    getSmall = 0,
    verbose = FALSE,
    powerit = 0,
    priorWeight = 0,
    maxBased = FALSE
  )

```

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)
verbose	activates verbose output
powerit	alternative power iteration implementation, faster
priorWeight	scalar weight typically in range zero to two
maxBased	boolean that chooses max-based thresholding

Value

outputs a decomposition of a population or time series matrix

Author(s)

Avants BB

Examples

```

mat <- replicate(100, rnorm(20))
mydecom <- sparseDecom(mat)
mat <- scale(mat)
mydecom2 <- sparseDecom(mat)
# params that lead to algorithm similar to NMF
mydecom3 <- sparseDecom(mat, z = 1, sparseness = 1)

## 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$eigenanatomyimages) > 0)
  ))
# compare to http://arxiv.org/pdf/0707.0701v2.pdf
# now SNPs
data(mice)
snps <- quantifySNPs(mice.X, shiftit = TRUE)
numericalpheno <- as.matrix(mice.pheno[, c(4, 5, 13, 15)])
nfolds <- 6
train <- sample(rep(c(1:nfolds), 1800 / nfolds))
train <- (train < 4)
lrmat <- lowrankRowMatrix(as.matrix(snps[train, ]), 50)
lrmat <- scale(lrmat)
snpd <- sparseDecom(lrmat - min(lrmat), nvecs = 20, sparseness = (0.001), z = -1)
projmat <- as.matrix(snpd$eig)
snpsse <- as.matrix(snps[train, ]) %%% projmat
traindf <- data.frame(bmi = numericalpheno[train, 3], snpsse = snpsse)
snpsse <- as.matrix(snps[!train, ]) %%% projmat
testdf <- data.frame(bmi = numericalpheno[!train, 3], snpsse = snpsse)
myrf <- randomForest(bmi ~ ., data = traindf)
predddf <- predict(myrf, newdata = testdf)
cor.test(predddf, testdf$bmi)

```

```

    plot(preddf, testdf$bmi)
} # check for packages
# prior-based example
set.seed(123)
ref <- antsImageRead(getANTsRData("r16"))
ref <- iMath(ref, "Normalize")
mi <- antsImageRead(getANTsRData("r27"))
mi2 <- antsImageRead(getANTsRData("r30"))
mi3 <- antsImageRead(getANTsRData("r62"))
mi4 <- antsImageRead(getANTsRData("r64"))
mi5 <- antsImageRead(getANTsRData("r85"))
refmask <- getMask(ref)
refmask <- iMath(refmask, "ME", 2) # just to speed things up
ilist <- list(mi, mi2, mi3, mi4, mi5)
for (i in 1:length(ilist))
{
  ilist[[i]] <- iMath(ilist[[i]], "Normalize")
  mytx <- antsRegistration(
    fixed = ref, moving = ilist[[i]],
    typeOfTransform = c("Affine")
  )
  mywarpedimage <- antsApplyTransforms(
    fixed = ref, moving = ilist[[i]],
    transformlist = mytx$fdwtransforms
  )
  ilist[[i]] <- mywarpedimage
}
mat <- imageListToMatrix(ilist, refmask)
kmseg <- kmeansSegmentation(ref, 3, refmask)
initlist <- list()
for (k in 1:3) {
  initlist[[k]] <-
    thresholdImage(kmseg$probabilityimages[[k]], 0.1, Inf) *
    kmseg$probabilityimages[[k]]
}
eanat <- sparseDecom(mat,
  inmask = refmask, ell1 = 0.1,
  sparseness = 0.0, smooth = 0.5, verbose = 1,
  initializationList = initlist, cthresh = 25,
  nvecs = 3, priorWeight = 0.5
)
ee <- matrixToImages(eanat$eigenanatomyimages, refmask)
eseg <- eigSeg(refmask, ee)
priormat <- imageListToMatrix(initlist, refmask)
cor(t(eanat$eigenanatomyimages), t(priormat))
plot(ref, eseg)

## End(Not run)

```

Description

Decomposes two matrices into paired sparse eigenvectors to maximize canonical correlation. Note: we do not scale the matrices internally. We leave scaling choices to the user.

Usage

```
sparseDecom2(
  inmatrix,
  inmask = list(NULL, NULL),
  sparseness = c(0.01, 0.01),
  nvecs = 3,
  its = 20,
  cthresh = c(0, 0),
  statdir = NA,
  perms = 0,
  uselong = 0,
  z = 0,
  smooth = 0,
  robust = 0,
  mycoption = 0,
  initializationList = list(),
  initializationList2 = list(),
  ell1 = 10,
  priorWeight = 0,
  verbose = FALSE,
  rejector = 0,
  maxBased = FALSE
)
```

Arguments

<code>inmatrix</code>	input as <code>inmatrix=list(mat1,mat2)</code> . n by p input matrix and n by q input matrix , spatial variable lies along columns.
<code>inmask</code>	optional pair of <code>antsImage</code> masks
<code>sparseness</code>	a <code>c(.,.)</code> pair of values e.g <code>c(0.01,0.1)</code> enforces an unsigned 99 percent and 90 percent sparse solution for each respective view
<code>nvecs</code>	number of eigenvector pairs
<code>its</code>	number of iterations, 10 or 20 usually sufficient
<code>cthresh</code>	cluster threshold pair
<code>statdir</code>	temporary directory if you want to look at full output
<code>perms</code>	number of permutations. settings permutations greater than 0 will estimate significance per vector empirically. For small datasets, these may be conservative. p-values depend on how one scales the input matrices.
<code>uselong</code>	enforce solutions of both views to be the same - requires matrices to be the same size
<code>z</code>	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
verbose	activates verbose output to screen
rejector	rejects small correlation solutions
maxBased	boolean that chooses max-based thresholding

Value

outputs a decomposition of a pair of matrices

Author(s)

Avants BB

Examples

```
mat <- replicate(100, rnorm(20))
mat2 <- replicate(100, rnorm(20))
mat <- scale(mat)
mat2 <- scale(mat2)
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"))
mask <- thresholdImage(im, 250, Inf)
dd <- sum(mask == 1)
mat1 <- matrix(rnorm(dd * 10), nrow = 10)
mat2 <- matrix(rnorm(dd * 10), nrow = 10)
initlist <- list()
for (nvecs in 1:2) {
```

```

    init1 <- antsImageClone(mask)
    init1[mask == 1] <- rnorm(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)

```

Description

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

Usage

```
sparseDecom2boot(
  inmatrix,
  inmask = c(NULL, NULL),
  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,
  priorWeight = 0,
  verbose = FALSE,
  estimateSparseness = 0.2
)
```

Arguments

<code>inmatrix</code>	input as <code>inmatrix=list(mat1,mat2)</code> . n by p input matrix and n by q input matrix , spatial variable lies along columns.
<code>inmask</code>	optional pair of <code>antsImage</code> masks
<code>sparseness</code>	a <code>c(...)</code> pair of values e.g <code>c(0.01,0.1)</code> enforces an unsigned 99 percent and 90 percent sparse solution for each respective view
<code>nvecs</code>	number of eigenvector pairs
<code>its</code>	number of iterations, 10 or 20 usually sufficient
<code>cthresh</code>	cluster threshold pair
<code>statdir</code>	temporary directory if you want to look at full output
<code>perms</code>	number of permutations
<code>uselong</code>	enforce solutions of both views to be the same - requires matrices to be the same size
<code>z</code>	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
priorWeight	Scalar value weight on prior between 0 (prior is weak) and 1 (prior is strong). Only engaged if initialization is used
verbose	activates verbose output to screen
estimateSparseness	effect size to estimate sparseness per vector

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 eigenvectors to maximize explained variance.

Usage

```
sparseDecomboot(
  inmatrix,
  inmask = NULL,
  sparseness = 0.01,
  nvecs = 50,
  its = 5,
  cthresh = 250,
  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
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	bootstrap integer e.g. 10 equals 10 bootstraps
nsamp	value less than or equal to 1, e.g. 0.9 means 90 percent of data will be used in each bootstrap 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$cca$outAuto)
  trairdf <- data.frame(
    lclass = as.factor(lymphoma$y[training]),
    eig = lymphoma$x[training, ] %*% t(outmat)
  )
  testdf <- data.frame(
    lclass = as.factor(lymphoma$y[!training]),
    eig = lymphoma$x[!training, ] %*% t(outmat)
  )
  myrf <- randomForest(lclass ~ ., data = trairdf)
  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 dontrun

## End(Not run)

```

`sparseDistanceMatrix` *Create sparse distance, covariance or correlation matrix*

Description

Exploit k-nearest neighbor algorithms to estimate a sparse similarity matrix. Critical to the validity of this function is the basic mathematical relationships between euclidean distance and correlation and between correlation and covariance. For applications of such matrices, one may see relevant publications by Mauro Maggioni and other authors.

Usage

```
sparseDistanceMatrix(
  x,
  k = 3,
  r = Inf,
  sigma = NA,
  kmetric = c("euclidean", "correlation", "covariance", "gaussian"),
  eps = 1e-06,
  ncores = NA,
  sinkhorn = FALSE,
  kPackage = "RcppHNSW",
  verbose = FALSE
)
```

Arguments

<code>x</code>	input matrix, should be n (samples) by p (measurements)
<code>k</code>	number of neighbors
<code>r</code>	radius of epsilon-ball
<code>sigma</code>	parameter for kernel PCA.
<code>kmetric</code>	similarity or distance metric determining k nearest neighbors
<code>eps</code>	epsilon error for rapid knn
<code>ncores</code>	number of cores to use
<code>sinkhorn</code>	boolean
<code>kPackage</code>	name of package to use for knn. FNN is reproducible but RcppHNSW is much faster (with nthreads controlled by environment variable <code>ITK_GLOBAL_DEFAULT_NUMBER_OF_THREADS</code>) for larger problems. For large problems, compute the regularization once and save to disk; load for repeatability.
<code>verbose</code>	verbose output

Value

matrix sparse p by p matrix is output with p by k nonzero entries

Author(s)

Avants BB

References

Maggioni, M., et al. "Multiscale data geometry."

Examples

```
## Not run:
set.seed(120)
mat <- matrix(rnorm(60), ncol = 10)
smat <- sparseDistanceMatrix(mat, 2)
r16 <- antsImageRead(getANTsRData("r16"))
mask <- getMask(r16)
mat <- getNeighborhoodInMask(
  image = r16, mask = mask, radius = c(0, 0),
  physical.coordinates = TRUE, spatial.info = TRUE
)
smat <- sparseDistanceMatrix(t(mat$indices), 10) # close points
testthat::expect_is(smat, "Matrix")
testthat::expect_is(smat, "dgCMatrix")
testthat::expect_equal(sum(smat), 18017)

## End(Not run)
```

sparseDistanceMatrixXY

Create sparse distance, covariance or correlation matrix from x, y

Description

Exploit k-nearest neighbor algorithms to estimate a sparse matrix measuring the distance, correlation or covariance between two matched datasets. Critical to the validity of this function is the basic mathematical relationships between euclidean distance and correlation and between correlation and covariance. For applications of such matrices, one may see relevant publications by Mauro Maggioni and other authors.

Usage

```
sparseDistanceMatrixXY(
  x,
  y,
  k = 3,
  r = Inf,
  sigma = NA,
  kmetric = c("euclidean", "correlation", "covariance", "gaussian"),
  eps = 1e-06,
```

```

    kPackage = "RcppHNSW",
    ncores = NA,
    verbose = FALSE
  )

```

Arguments

x	input matrix, should be n (samples) by p (measurements)
y	input matrix second view, should be n (samples) by q (measurements)
k	number of neighbors
r	radius of epsilon-ball
sigma	parameter for kernel PCA.
kmetric	similarity or distance metric determining k nearest neighbors
eps	epsilon error for rapid knn
kPackage	name of package to use for knn. FNN is reproducible but RcppHNSW is much faster (with nthreads controlled by environment variable ITK_GLOBAL_DEFAULT_NUMBER_OF_THREADS) for larger problems. For large problems, compute the regularization once and save to disk; load for repeatability.
ncores	number of cores to use
verbose	verbose output

Value

matrix sparse p by q matrix is output with p by k nonzero entries

Author(s)

Avants BB

References

Maggioni, M., et al. "Multiscale data geometry."

Examples

```

## Not run:
set.seed(120)
mat <- matrix(rnorm(60), nrow = 6)
mat2 <- matrix(rnorm(120), nrow = 6)
smat <- sparseDistanceMatrixXY(mat, mat2, 3)
smat2 <- sparseDistanceMatrixXY(mat2, mat, 3)
testthat::expect_is(smat, "Matrix")
testthat::expect_is(smat, "dgCMatrix")
testthat::expect_is(smat2, "Matrix")
testthat::expect_is(smat2, "dgCMatrix")
testthat::expect_equal(sum(smat), 154.628961265087)
testthat::expect_equal(sum(smat2), 63.7344262003899)

## End(Not run)

```

sparseRegression	<i>Sparse regression on input images.</i>
------------------	---

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 = NULL,
  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

```

nsubj <- 50
prop.train <- 1 / 2
subj.train <- sample(1:nsubj, prop.train * nsubj, replace = FALSE)
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)

```

sparsify_by_column_winner

Sparsify a Matrix via Column-wise Winner-Take-All

Description

This function applies a "winner-take-all" sparsity model. For each column, it finds the single most significant entry based on specified sign constraints and sets all other entries in that column to zero.

Usage

```
sparsify_by_column_winner(
  X,
  first_column_constraint = c("none", "either", "positive", "negative"),
  default_constraint = c("either", "positive", "negative"),
  ensure_row_membership = TRUE
)
```

Arguments

X A numeric matrix [p_features x k_components].

first_column_constraint How to treat the first column. One of:

- "none": (Default) The first column is left dense (not sparsified).
- "either": The entry with the largest absolute value is kept, regardless of sign.
- "positive": The largest entry with a positive sign is kept.
- "negative": The entry with the largest magnitude (most negative) with a negative sign is kept.

default_constraint How to treat all other columns (from the second onwards). Takes the same options as 'first_column_constraint'. Defaults to "either".

ensure_row_membership Logical. If TRUE, ensures every feature is represented in at least one component by "reviving" the single most significant entry for any row that becomes all-zero after sparsity.

Details

It includes special handling for the first column, which often represents a main effect that may not require the same constraints as subsequent components.

Value

A sparsified matrix with the same dimensions as X.

Examples

```

set.seed(123)
mat <- matrix(c(
  -5, 0.1, 0.2, 0.3, 0.4,    # Col 1: Large negative value
  1, 2.0, 0.1, 0.2, 0.3,    # Col 2: Large positive value
  -1, -0.2, -3, -0.4, -0.5, # Col 3: Only negative values
  1, -2, 3, -4, 5           # Col 4: Mixed signs
), nrow = 5, ncol = 4)

res1 <- sparsify_by_column_winner(mat, 'positive', 'positive')

```

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 = NULL, 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
testthat::expect_error(splitChannels(r))
rgbImage <- mergeChannels(list(r, g, b))
imgList <- splitChannels(rgbImage)
testthat::expect_length(imgList, 3)
sapply(imgList, function(x) {
  testthat::expect_s4_class(x, class = "antsImage")
})
testthat::expect_error(splitChannels("hey"))

```

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 $1/\text{ratio}$ 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)
```

splitMask*split a mask into n labeled sub-masks*

Description

split a mask into n labeled sub-masks

Usage

```
splitMask(mask, n = NA, voxchunk = NA)
```

Arguments

mask	antsImage mask
n	number of mask chunks (if voxchunk is not set)
voxchunk	number of voxels per chunk (if n is not set)

Value

relabeledMask

Author(s)

Avants BB, Tustison NJ, Pustina D

Examples

```
mask <- getMask(antsImageRead(getANTsRData("r16")))
smask <- splitMask(mask, 10)
```

splitNDImageToList	<i>splitNDImageToList</i>
--------------------	---------------------------

Description

This function splits the dimensionality *d* input image into a list of images with dimensionality *d*-1 by slicing the input image along its last dimension.

Usage

```
splitNDImageToList(img)
```

Arguments

img input image of class *antsImage*, last dimension will be split

Value

output list filled with *d*-1 volumes of class *antsImage*

Author(s)

Pustina D, Avants B

Examples

```
my4Dimage <- makeImage(c(5, 5, 5, 4))
my3Dlist <- splitNDImageToList(my4Dimage)
```

statFieldThresh	<i>Produces a threshold value based on cluster or voxel level statistics</i>
-----------------	--

Description

Produces a threshold value based on cluster or voxel level statistics

Usage

```
statFieldThresh(
  x,
  pval,
  nvox,
  n,
  fwhm,
  resels,
```

```

    df,
    fieldType,
    threshType,
    pp = 0.001,
    verbose = FALSE
)

```

Arguments

x	statistical map of class antsImage
pval	p-value for determining threshold
nvox	minimum desired cluster size (in voxels)
n	number of images in conjunction
fwhm	full width at half maxima
resels	resolution elements in search region
df	degrees of freedom expressed as $df = c(\text{degrees of interest}, \text{degrees of error})$
fieldType	T: T-field F: F-field X: Chi-square field' Z: Gaussian field
threshType	cRFT: computes a threshold per expected cluster level probability pRFT: uses the mask and pval calculates the minimum statistical threshold cFDR: uses an uncorrected threshold at the alpha level and then computes and FDR threshold based on cluster maxima pFDR: computes the fdr threshold for the entire field of voxels
pp	primary (initial) p-value threshold used in FDR methods
verbose	enables verbose output

Details

A statistical threshold level is determined using the estimated p-value pval given the provided parameters. "cRFT" and "pRFT" specify the method of estimation should use RFT cluster and peak statistic estimates respectively.

In addition to RFT based thresholds the user may choose to utilize false-discovery rate (FDR) based thresholds. These use a primary p-value threshold (default $pp = .001$) to create suprathreshold clusters which in turn are used to determine a final threshold (utilizing pval at this point). If the estimated peak-FDR statistic is used ("pFDR") then all suprathreshold voxels are FDR corrected to determine the threshold. If the estimated cluster-FDR statistic is used then the cluster maxima are FDR corrected in order to determine the threshold.

It is important that the user recognize that when statistical analyses are computed using RFT that the threshold level plays a role in predicting the p-value. This is because the probability of obtaining results is weighted against the probability of any occurrence given the threshold. For example, if two clusters where exactly the same but had one was obtained using a lower threshold it would have a lower p-value. This has been shown with a power analysis using similar parameters to those

presented herein with the additional variable of signal characteristics to demonstrate the effect of image modality on analysis (Friston et al., 1996). Therefore, parameters should be chosen according to the type of analysis being performed (fMRI, PET, or VBM) and the hypothesis being tested. Due to widely varying opinions on appropriate thresholding procedures no specific recommendations are made here. This function simply facilitates the use of several approaches that users may utilize after consulting available literature.

Value

Outputs a statistical value to be used for threshold a SPM

Author(s)

Zachary P. Christensen

References

- Chumbley J., (2010) Topological FDR for neuroimaging
 Friston K.J., (1994) Assessing the Significance of Focal Activations Using Their Spatial Extent
 Friston K.J., (1996) Detecting Activations in PET and fMRI: Levels of Inference and Power

Examples

```
## Not run:
# threshold according to estimated peak height
thresh_image <- statFieldThresh(timg, .05, 10, 1, mysmooth$fwhm,
  c(1, fit$df.residual),
  fieldType = "T",
  threshType = pRFT
)

## End(Not run)
```

step

Take One Optimization Step

Description

This is a generic S3 function that dispatches to the correct update rule based on the optimizer's class.

Usage

```
step(optimizer, i, V_current, descent_gradient, ...)
```

Arguments

optimizer	The optimizer object from 'create_optimizer()'.
i	The index of the modality currently being updated.
V_current	The current V matrix for modality 'i'.
descent_gradient	The Riemannian DESCENT gradient (-grad(E)) for V_current.
...	Additional arguments required by the specific method (e.g., 'full_energy_function' for line searches, 'learning_rate' for others).

Value

A list containing 'updated_V' and the updated 'optimizer' object.

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 row 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)
```

subset.antsImage	<i>Subsetting antsImage Objects</i>
------------------	-------------------------------------

Description

Returns subsets of antsImage objects as vectors

Usage

```
## S3 method for class 'antsImage'
subset(x, subset, ...)
```

Arguments

- x antsImage object
- subset logical expression of subset
- ... further arguments to be passed to or from other methods.

sum_preserving_matrix_partition	<i>Sum preserving matrix partition</i>
---------------------------------	--

Description

This function takes a matrix as input and partitions each column such that only one entry in each row is non-zero, and the sums of each row are roughly equivalent.

Usage

```
sum_preserving_matrix_partition(X, option = 2, tol = 0.001)
```

Arguments

<code>X</code>	A matrix of size $p \times k$
<code>option</code>	An integer indicating whether to allow +/- values (option 1) or only + values (option 2). Default is 1.
<code>tol</code>	A numeric value indicating the tolerance for the row sums. Default is 0.1.

Details

The function uses a greedy algorithm to select the entry with the maximum absolute value (or maximum positive value) in each column. The rows are then normalized to ensure that the sums are roughly equivalent.

Value

A matrix with segmented rows

Examples

```
X <- matrix(rnorm(3000), nrow = 1000)
Y <- sum_preserving_matrix_partition(X, option = 1)
Y <- sum_preserving_matrix_partition(X, option = 2)
```

symm

Symmetrize a Matrix

Description

Computes the symmetric part of a matrix by averaging it with its transpose.

Usage

```
symm(A)
```

Arguments

<code>A</code>	Numeric matrix to symmetrize.
----------------	-------------------------------

Details

The symmetric part is defined as $\text{sym}(A) = (A + A^T)/2$, ensuring the result is symmetric. This is used in computing the Riemannian gradient for the Stiefel manifold.

Value

Numeric matrix, the symmetric part $(A + A^T)/2$.

Examples

```
A <- matrix(c(1, 2, 3, 4), 2, 2)
symm(A)
```

T_template0_LobesBstem	<i>T template0 LobesBstem</i>
------------------------	-------------------------------

Description

T template0 LobesBstem

take_abs_unsigned	<i>Take Absolute Value of Unsigned Columns</i>
-------------------	--

Description

This function iterates through each column of a numeric matrix. If a column contains only zero and positive values, or only zero and negative values, the absolute value of that column is taken. Otherwise, the column is returned as is.

Usage

```
take_abs_unsigned(m)
```

Arguments

m A numeric matrix.

Value

A matrix with absolute values taken for unsigned columns.

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 getfMRIin nuisanceVariables
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 <- getfMRIInuisanceVariables(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)
```

testEanat	<i>Test eigenanatomy in order</i>
-----------	-----------------------------------

Description

This tests each eigenanatomy region in order invisibly to the user and stops when testing stops conserving power. The algorithm returns NA if there is no good testing procedure, given the statistical target. The basic idea is to test the variables that explain the most variance first and continue testing as long as (a) there is no significant relationship yet found or (b) the existing significant relationship remains significant, given the correction for multiple comparisons.

Usage

```
testEanat(mymdl, myvar, sigthresh = 0.05, method = "BH")
```

Arguments

mymdl	input model from lm with eigenanatomy on left
myvar	name of variable in model to test
sigthresh	significance threshold, for example 0.05
method	a method for p.adjust

Value

nvecs is output, analogous to nvecs in `svd(mat, nu=0, nv=nvecs)`

Author(s)

Avants BB, Tustison NJ

References

Avants BB, Hackman D, LM Betancourt, GM Lawson, H Hurt, MJ Farah #' Relation of Childhood Home Environment to Cortical Thickness in Late Adolescence: Specificity of Experience and Timing, PloS one, 2015

Examples

```
mat <- matrix(rnorm(300), ncol = 3)
n <- nrow(mat)
g <- factor(c(rep(1, n / 2), rep(0, n / 2)))
mydf <- data.frame(gen = g, a = rnorm(n))
mymdl <- lm(mat ~ a + g, data = mydf)
nv <- testEanat(mymdl, myvar = "g1")
mat[1:(n / 2), 3] <- mat[1:(n / 2), 3] + 2
mymdl <- lm(mat ~ a + g, data = mydf)
nv <- testEanat(mymdl, myvar = "g1")
```

thresholdImage	<i>Threshold Image</i>
----------------	------------------------

Description

converts a scalar image into a binary image by thresholding operations

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

```
set.seed(1234)
img <- makeImage(c(5, 5), rnorm(25) + 0.5)
imgt <- thresholdImage(img, 0.5, Inf)
testthat::expect_equal(sum(imgt), 9)
imgt <- thresholdImage(img > 0.5, 0.1, Inf)
testthat::expect_equal(sum(imgt), 9)
```

timeseries2matrix	<i>Time-series image to matrix</i>
-------------------	------------------------------------

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	<i>Run N3 on slices of timeseries.</i>
--------------	--

Description

Repeatedly run N3 on slices of time series and return image

Usage

```
timeseriesN3(boldimg, mask, ncorrections = c(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

<code>timeseriesmatrix</code>	Input timeseriesmatrix from timeseries2matrix
<code>mask</code>	Input mask of type 'antsImage' ... the mask will be replicated into a 4D image of length timewindow + zeropadvalue.
<code>eventlist</code>	time indices for the events to extract
<code>timewindow</code>	n-timepoints around each event, forward in time.
<code>zeropadvalue</code>	pads the mask by this amount fwd and bwd in time.
<code>spacing</code>	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, sparseness = 0.1, z = 0.5, nvecs = 2,
  its = 5, cthresh = 0, mycoption = 1)
enomask <- eanat2$eigenanatomyimages[1, ]
emask <- eanat$eigenanatomyimages[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, sparseness = -0.9, z = 0.5, nvecs = 2,
its = 5, cthresh = 0, mycoption = 1)
enomask <- eanat2$eigenanatomyimages[, 1]
# back to timematrix
tmat <- matrix(enomask, nrow = 6)
# back to image
eimg <- antsImageClone(msk)
eimg[msk == 1] <- tmat[1, ]
# convert image space to evec space
emat <- eanat2$eigenanatomyimages
# convert emat to events FIXME this does not currently work
# eavent<-timeserieswindow2matrix( data.matrix(emat) , msk, 1 , 6, 0 )
# emask<-eavent$eventmatrix[1,]
#####
```

tissues	<i>tissues</i>
---------	----------------

Description

A data frame label numbers and names for tissue types.

Usage

tissues

Format

A data frame with columns: Label, Description, ProbabilityNumber.

tracts	<i>tracts</i>
--------	---------------

Description

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

Usage

tracts

Format

A data frame listing the following variables.

label_num Numerical label value.

label_name Shorthand anatomical value.

References

<https://pubmed.ncbi.nlm.nih.gov/18407524>

Examples

```
data(tracts)
```

transform_matrix	<i>Transform a matrix by standard preprocessing</i>
------------------	---

Description

Applies centering, scaling, and/or normalization transforms to prepare data for optimization or learning-rate estimation.

Usage

```
transform_matrix(  
  X,  
  method = c("none", "center", "zscore", "minmax", "l2", "frob", "frob_zscore")  
)
```

Arguments

X	Numeric matrix.
method	Character string specifying the transformation: <ul style="list-style-type: none">• "none": no transformation• "center": subtract column means• "zscore": center and scale columns by their standard deviations• "minmax": rescale columns to [0, 1]• "l2": normalize each column to unit L2 norm, then multiply by sqrt(p*k)• "frob": normalize entire matrix to unit Frobenius norm, then multiply by sqrt(p*k)• "frob_zscore": z-score columns, then normalize Frobenius norm, then multiply by sqrt(p*k)

Value

A list with:

- Xs: transformed matrix
- params: list of learned transform parameters

transform_prior_rows_to_uncorrelated

Transform prior rows to (approximately) uncorrelated representative rows

Description

Given a prior matrix P ($k \times p$) whose rows are correlated and potentially over-represent some latent signals, this function produces a reduced set of row-priors that are uncorrelated and representative.

Usage

```
transform_prior_rows_to_uncorrelated(
  P,
  j = NULL,
  method = c("svd", "cluster_then_svd"),
  row_center = TRUE,
  row_scale = FALSE,
  varimax = FALSE,
  cluster_k = NULL,
  var_exp_threshold = 0.9,
  verbose = TRUE
)
```

Arguments

<code>P</code>	numeric matrix ($k \times p$) — rows are the priors to compress/orthogonalize.
<code>j</code>	integer or NULL. Number of components to keep. If NULL, chosen automatically.
<code>method</code>	character, one of "svd" or "cluster_then_svd".
<code>row_center</code>	logical. Subtract row means before analysis (recommended).
<code>row_scale</code>	logical. Scale rows to unit sd (optional).
<code>varimax</code>	logical. Apply varimax rotation to loadings (improves interpretability).
<code>cluster_k</code>	integer or NULL. If method = "cluster_then_svd", number of clusters. If NULL, chosen with silhouette / dynamic.
<code>var_exp_threshold</code>	numeric in (0,1). If j is NULL, choose smallest j with cumulative variance $\geq$ this threshold.
<code>verbose</code>	logical.

Details

Methods: - "svd" : row-PCA / SVD on P (default). Returns top j orthogonal components. - "cluster_then_svd": cluster correlated rows, compute cluster centroids, then run SVD on centroids to produce orthogonal representatives.

The function will: 1. optionally normalize rows 2. compute SVD of P (or of centroids) 3. select j (user provided or automatic by variance explained) 4. return j orthogonal (uncorrelated) row-vectors in R^p

Value

A list with elements: - P_orig: original input - P_trans: transformed row-priors ($j \times p$) (rows uncorrelated) - U: left singular vectors ($k \times j$) - S: singular values (j) - Vt: right singular vectors transposed ($j \times p$) - transform_fn: function(newP) to map new data with same transform - diagnostics: list with eigenvalues, variance explained, chosen j, clustering info

trapz

Calculate the area under a sampled curve (or set of curves).

Description

Given a vector (or a matrix) representing a curve (or set of curves, columnwise), the area (or set of areas) is calculated using the trapezoidal rule.

Usage

```
trapz(x, y)
```

Arguments

x	vector of samples for the dependent variable.
y	vector or matrix of samples for the independent variable. In the case of the latter, curves are organized column-wise.

Value

area (areas) under the sampled curve (curves).

Author(s)

Tustison NJ

Examples

```
x <- seq(0, 1, by = 0.0001)
y <- exp(x)

# Compare with true area of exp( 1 ) - 1 = 1.718282...
areaEstimate <- trapz(x, y)
```

`undo_transform_matrix` *Undo a previously applied matrix transform*

Description

Undo a previously applied matrix transform

Usage

```
undo_transform_matrix(Xs, params)
```

Arguments

<code>Xs</code>	Transformed matrix.
<code>params</code>	List of parameters returned by <code>transform_matrix()</code> .

Value

The untransformed (original scale) matrix.

`unique.antsImage` *Unique for antsImage Objects*

Description

Overloaded uniqueness operation for `antsImage` objects

Usage

```
## S3 method for class 'antsImage'
unique(x, incomparables = FALSE, mask = NULL, ...)
```

Arguments

<code>x</code>	is an object of class <code>antsImage</code> .
<code>incomparables</code>	a vector of values that cannot be compared.
<code>mask</code>	binary mask of values to subset
<code>...</code>	additional arguments passed to <code>unique</code>

Examples

```
img <- antsImageRead(getANTsRData("r16"))
img[img > 5] <- 0
sort(unique(img))
```

usePkg	<i>Use any package. If package is not installed, this will install from CRAN.</i>
--------	---

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 <i>*string*</i> .
allowInstall	let the package be installed from CRAN

Value

TRUE if package successfully loaded, FALSE otherwise.

Author(s)

Benjamin M. Kandel, BB Avants

Examples

```
usePkg("randomForest")
usePkg("stats", allowInstall = TRUE)
```

var	<i>Variance generic</i>
-----	-------------------------

Description

Calculates the variance of an image

Usage

```
var(x, ...)

## Default S3 method:
var(x, ...)

## S3 method for class 'antsImage'
var(x, ..., na.rm = FALSE, mask = NULL)
```


Arguments

- x an object for which we want to compute the variance
- ... Any additional arguments to be passed to `var`
- na.rm a logical value indicating whether NA should be removed
- mask is an object of class `antsImage`

Examples

```
img <- antsImageRead(getANTsRData("r16"))
var(img)
```

vector_to_df	<i>Convert a Vector to a Data Frame</i>
--------------	---

Description

Convert a Vector to a Data Frame

Usage

```
vector_to_df(vector, column_name)
```

Arguments

- vector The input vector
- column_name The base name for the columns

Value

A data frame with one row and columns determined by the length of the vector

Examples

```
vector_to_df(1:10, "nm")
```

vectorToMultichannel *Convert vector to multichannel antsImage.*

Description

Converts a vector of size n-entries in mask times number of channels to a multi-channel image with the same dimensionality as the mask.

Usage

```
vectorToMultichannel(v, mask)
```

Arguments

v	input multichannel vector
mask	mask of same dimensionality as multichannelimage

Value

multichannelimage is output

Author(s)

Avants BB

Examples

```
fi <- antsImageRead(getANTSRData("r16")) %>%
  resampleImage(c(60, 60), 1, 0)
mi <- antsImageRead(getANTSRData("r64")) %>%
  resampleImage(c(60, 60), 1, 0)
mytx <- antsRegistration(fixed = fi, moving = mi, typeofTransform = c("SyN"))
mcimg <- antsImageRead(mytx$fwd[1])
msk <- getMask(fi)
vv <- multichannelToVector(mcimg, msk)
mcimg2 <- vectorToMultichannel(vv, msk)
vv2 <- multichannelToVector(mcimg2, msk)
cor.test(vv2, vv)
stopifnot(all(mcimg2[30, 30] == mcimg[30, 30]))
```

visualize_lowrank_relationships
<i>Visualize Low-Rank Relationships</i>

Description

Visualize Low-Rank Relationships

Usage

```
visualize_lowrank_relationships(  
  X1,  
  X2,  
  V1,  
  V2,  
  plot_title = NULL,  
  nm1 = "X1",  
  nm2 = "X2",  
  max_pairs = 5  
)
```

Arguments

X1	First matrix (Subjects x Features)
X2	Second matrix (Subjects x Features)
V1	First feature loading matrix (Features x Components)
V2	Second feature loading matrix (Features x Components)
plot_title	Title of the plot
nm1	Label for view 1
nm2	Label for view 2
max_pairs	Maximum components to include in the pairs plot (Safety cap)

visualize_permutation_test
<i>Visualize Permutation Test Results</i>

Description

This function visualizes the results of a permutation test by plotting a histogram of the permutation test statistics. A red dotted line indicates the location of the original unpermuted test statistic.

Usage

```
visualize_permutation_test(
  permutation_results,
  original_stat,
  stat_name,
  plot_title,
  bin_width = 0.1
)
```

Arguments

<code>permutation_results</code>	A numeric vector of permutation test statistics.
<code>original_stat</code>	A numeric value representing the original unpermuted test statistic.
<code>stat_name</code>	A character string representing the name of the test statistic.
<code>plot_title</code>	string for plot title
<code>bin_width</code>	optional bin width for the histogram

Value

A ggplot object showing the histogram of permutation test statistics with the original test statistic marked.

Examples

```
## Not run:
set.seed(123)
n_perms <- 1000
permutation_results <- rnorm(n_perms, mean = 0, sd = 1)
original_stat <- 2
visualize_permutation_test(permutation_results, original_stat, "Simulated Statistic")

## End(Not run)
```

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,
  labelmasks,
  rad = NA,
  nsamples = 8,
  ntrees = 500,
  asFactors = TRUE,
  reduceFactor = 1,
  ...
)
```

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
labelmasks	a list of masks where each mask defines the image space for the given list. that is, the nth mask indexes the nth feature set. multi-label masks will try to balance sampling for each label.
rad	vector of dimensionality d define nhooood radius
nsamples	(per subject to enter training)
ntrees	(for the random forest model)
asFactors	boolean - treat the y entries as factors
reduceFactor	integer factor by which to reduce (imaging) data resolution
...	arguments to pass to randomForest

Value

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

Author(s)

Avants BB, Tustison NJ, Pustina D

Examples

```
mask <- makeImage(c(10, 10), 0)
mask[3:6, 3:6] <- 1
mask[5, 5:6] <- 2
ilist <- list()
lablist <- list()
masklist <- list()
inds <- 1:5
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
    masklist[[i]] <- mask
  }
  rfm <- vwnrfs(lablist, ilist, masklist[[1]], rad = c(2, 2)) # use single mask
  rfm <- vwnrfs(lablist, ilist, masklist, 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

```

vwnrfs.predict

voxelwise neighborhood random forest prediction

Description

Takes a model created with vwnrfs and builds a prediction based on similar features used to train vwnrfs

Usage

```

vwnrfs.predict(
  rfm,
  x,
  labelmasks,
  rad = NA,
  asFactors = TRUE,
  voxchunk = 30000,
  reduceFactor = 1
)

```

Arguments

rfm	random forest model trained with vwnrfs with certain number of features.
x	a list of lists. Each list contains the list of feature images required to predict a response or an image. The features must be the same used during training. I.e., if you train on T1 and T2 images, those should be the same features used for prediction, in the same exact order for each subject.
labelmasks	a list of masks where each mask defines the space to predict from. These can be individual masks for each subject (i.e., custom brain masks) or a single antsImage that will be used for all subjects.
rad	vector of dimensionality d define the neighborhood radius. Must be the same radius with which the model was trained, i.e., c(1,1,1)
asFactors	boolean - treat the y entries as factors. If this is true, the prediction will be a classification, and the output will produce images. If this is false, the prediction will be a regression, and the output will produce a single response value.
voxchunk	value of maximal voxels to predict at once. This value is used to split the prediction into smaller chunks such that memory requirements do not become too big.
reduceFactor	value of resolution reduction (i.e., for 1mm voxels and reduceFactor=3) the model will be trained on ~3mm images.

Value

list a 2-list with the rf model, training vector, feature matrix and the random mask

Author(s)

Pustina D

Examples

```
## Do not run
## vwnrfs.predict(rfm, x=x, labelmasks=labelmasks,
## rad=rad, asFactors=TRUE, voxchunk=voxchunk,
## reduceFactor = mr)mask<-makeImage( c(10,10), 0 )
## End do not run
```

weingartenImageCurvature

image curvature for 2D or 3D

Description

uses the weingarten map to estimate image mean or gaussian curvature

Usage

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

Arguments

image	antsImage
sigma	smoothing parameter
opt	mean by default, otherwise use string gaussian or characterize
labeled	boolean if TRUE then assume image contains labels which denote the surface on which curvature is to be calculated. in this case, the surface should hold integer value 1; the background should be zero; the interior of the surface should be greater than 1 (e.g. 2).

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)
## 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 = NULL, 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)
```

whiten_matrix

Whitening Matrix

Description

This function performs matrix whitening on the input matrix 'X' using Singular Value Decomposition (SVD). Whitening transforms the input matrix into one where the covariance matrix is the identity matrix.

Usage

```
whiten_matrix(X)
```

Arguments

X	A numeric matrix to be whitened. The matrix should have observations as rows and features as columns.
---	---

Value

A list containing:

whitened_matrix

The whitened matrix where the covariance matrix is the identity matrix.

whitening_matrix

The whitening transformation matrix used to whiten the input matrix.

Examples

```
set.seed(123)
X <- matrix(rnorm(1000), nrow = 20, ncol = 50) # Example with p = 50 and n = 20
result <- whiten_matrix(X)
X_whitened <- result$whitened_matrix
whitening_matrix <- result$whitening_matrix
# Verify that the covariance matrix of the whitened matrix is close to identity
cov_X_whitened <- cov(X_whitened)
print(round(cov_X_whitened, 2))
```

<code>wm_major_tracts</code>	<i>wm_major_tracts</i>
------------------------------	------------------------

Description

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

Usage

```
wm_major_tracts
```

Format

A data frame with columns: Label, Description, side.

<code>wmh_evidence</code>	<i>wmh_evidence</i>
---------------------------	---------------------

Description

A data frame describing WMH evidence features.

Usage

```
wmh_evidence
```

Format

A data frame with columns: Value, Description.

write_simlr	<i>Write all SiMLR outputs to disk</i>
-------------	--

Description

Saves SiMLR outputs to a structured directory with relative paths for portability. Matrices and data frames are written as CSV files with row names included and stored in the manifest. Other R objects are saved as '.rds'. NULL components are recorded without files.

Usage

```
write_simlr(simlr_object, file_prefix, clear_dir = FALSE)
```

Arguments

simlr_object	A named list containing SiMLR outputs (e.g., 'mysim\$simlrX').
file_prefix	A character string used as the prefix for the output directory.
clear_dir	Logical: if TRUE, clear the output directory before writing (default: FALSE).

Value

No return value, called for side effects.

write_simlr_data_frames	<i>Write a list of data frames to disk with SiMLR-specific naming convention</i>
-------------------------	--

Description

This function writes each data frame in a list to a separate CSV file on disk, using the names of each data frame to create unique filenames.

Usage

```
write_simlr_data_frames(data_list, file_prefix)
```

Arguments

data_list	A list of data frames to write to disk.
file_prefix	A character string to use as the prefix for the filenames.

Value

No return value, called for side effects.

Examples

```
mysim <- list(simlrX = list(v = list(
  data1 = data.frame(matrix(rnorm(147 * 171), nrow = 147, ncol = 171)),
  data2 = data.frame(matrix(rnorm(156 * 171), nrow = 156, ncol = 171))
)))
write_simlr_data_frames(mysim$simlrX$v, "output")
```

writeAntsrTransform	<i>writeAntsrTransform</i>
---------------------	----------------------------

Description

write antsrTransform to disk

Usage

```
writeAntsrTransform(transform, filename)
```

Arguments

transform	antsrTransform
filename	filename of transform (file extension is ".mat" for affine transforms)

Value

TRUE

Examples

```
trans <- c(3, 4, 5)
tx <- createAntsrTransform(type = "Euler3DTransform", translation = trans)
txfile <- tempfile(fileext = ".mat")
writeAntsrTransform(tx, txfile)
```

writeNormalizedPopulationData	<i>Save normalized population data to a directory or h5 file</i>
-------------------------------	--

Description

This function simplifies the steps of saving a population (image) analysis into an efficient hdf5 file, using the h5 library or to a directory. The population image data is stored as a matrix. Along with the matrix will be a data frame with population demographics. The number of matrix columns will be defined by a mask that the user should also supply. Finally, a boolean vector should be passed in that matches the images to the demographics. This function cannot check if the matching between demographics and images is correct and, as such, the user should take care in creating the inputs. **WARNING:** when using h5, the demographics file will currently be cast to a data.matrix so one should not reuse this file as the reference demographics file.

Usage

```
writeNormalizedPopulationData(  
  demographics,  
  imageMat,  
  imageMask,  
  imageBoolean,  
  filename  
)
```

Arguments

demographics	data frame that identifies the variables of interest.
imageMat	a matrix that stores the normalized image data.
imageMask	mask with number of non-zero entries defining the matrix columns.
imageBoolean	a vector of booleans with length equal to the number of rows in the demographics data frame and number of true values equal to the number of rows in the image matrix.
filename	output filename for hdf5 file (if .h5) or directory name.

Value

successOrFailure boolean

Author(s)

Avants BB

Examples

```
# read below will test out writing as well  
## Not run:  
ilist <- getANTsRData("population")  
mask <- getMask(ilist[[1]])  
imat <- imageListToMatrix(ilist, mask)  
demog <- data.frame(  
  id = c("A", "B", "C", NA),  
  age = c(11, 7, 18, 22), sex = c("M", "M", "F", "F")  
)  
ibool <- c(TRUE, TRUE, TRUE, FALSE)  
tfn <- tempfile(fileext = ".h5")  
if (usePkg("hdf5r")) {  
  demographics <- demog  
  imageMat <- imat  
  imageMask <- mask  
  imageBoolean <- ibool  
  filename <- tfn  
  writeNormalizedPopulationData(  
    demog, imat, mask, ibool,  
    tfn  
  )  
}
```

542

writeNormalizedPopulationData

```
)  
}
```

```
## End(Not run)
```

Index

- !, antsImage-method (antsImage-math), [36](#)
- !=, ANY, antsImage-method
 - (antsImage-operators), [39](#)
- !=, antsImage, ANY-method
 - (antsImage-operators), [39](#)
- !=, antsImage, antsImage-method
 - (antsImage-operators), [39](#)
- !=, antsImage, array-method
 - (antsImage-operators), [39](#)
- !=, antsImage, list-method
 - (antsImage-operators), [39](#)
- !=, array, antsImage-method
 - (antsImage-operators), [39](#)
- * **Talairach**
 - getMultivariateTemplateCoordinates, [235](#)
 - getTemplateCoordinates, [241](#)
 - mniztal, [325](#)
- * **Template**
 - getMultivariateTemplateCoordinates, [235](#)
 - getTemplateCoordinates, [241](#)
- * **asl**
 - bayesianCBF, [128](#)
- * **convolve**
 - convolveImage, [156](#)
- * **crop**
 - cropImage, [161](#)
 - cropIndices, [163](#)
- * **curvature**
 - weingartenImageCurvature, [535](#)
- * **datasets**
 - aal, [18](#)
 - antsrVersions, [98](#)
 - basal_forebrain, [127](#)
 - bold_correlation_matrix, [136](#)
 - cerebellum_labels, [141](#)
 - cit168_brainstem, [143](#)
 - cit168_reinf_learn, [143](#)
 - DesikanKillianyTourville, [173](#)
 - dkt_cit_labels, [176](#)
 - dkt_labels, [177](#)
 - iMathOps, [265](#)
 - jhu_fa_labels, [278](#)
 - jhu_md_labels, [278](#)
 - lobes, [302](#)
 - lobes_brainstem, [303](#)
 - mtl_labels, [331](#)
 - nbm3_labels, [345](#)
 - powers_areal_mni_itk, [392](#)
 - ppmi_yeo_labels, [394](#)
 - tissues, [523](#)
 - tracts, [523](#)
 - wm_major_tracts, [538](#)
 - wmh_evidence, [538](#)
- * **decrop**
 - decropImage, [171](#)
- * **design**
 - blockStimulus, [134](#)
 - hemodynamicRF, [246](#)
- * **extract**
 - extractSlice, [195](#)
- * **flow**
 - bayesianCBF, [128](#)
- * **fusion**
 - jlfProp, [278](#)
 - jointLabelFusion, [281](#)
 - localJointLabelFusion, [304](#)
- * **geometry**
 - getCenterOfMass, [230](#)
 - reorientImage, [428](#)
- * **image**
 - getCenterOfMass, [230](#)
 - reorientImage, [428](#)
- * **inpainting**
 - basicInPaint, [127](#)
 - exemplarInpainting, [189](#)
- * **mask**

- getMask, 233
- * **regression**
 - blockStimulus, 134
 - hemodynamicRF, 246
- * **shape**
 - segmentShapeFromImage, 459
- * **svd**
 - randsvd, 411
- * **template**
 - basicInPaint, 127
 - exemplarInpainting, 189
 - jlfProp, 278
 - jointLabelFusion, 281
 - localJointLabelFusion, 304
 - segmentShapeFromImage, 459
- * **whiten**
 - randwhiten, 412
- *, antsImage, antsImage-method
(antsImage-operators), 39
- *, antsImage, numeric-method
(antsImage-operators), 39
- *, numeric, antsImage-method
(antsImage-operators), 39
- +, antsImage, antsImage-method
(antsImage-operators), 39
- +, antsImage, numeric-method
(antsImage-operators), 39
- +, numeric, antsImage-method
(antsImage-operators), 39
- , antsImage, antsImage-method
(antsImage-operators), 39
- , antsImage, numeric-method
(antsImage-operators), 39
- , numeric, antsImage-method
(antsImage-operators), 39
- .int_antsProcessArguments, 13
- .motion_correction, 13, 67
- /, antsImage, antsImage-method
(antsImage-operators), 39
- /, antsImage, numeric-method
(antsImage-operators), 39
- /, numeric, antsImage-method
(antsImage-operators), 39
- <, ANY, antsImage-method
(antsImage-operators), 39
- <, antsImage, ANY-method
(antsImage-operators), 39
- <, antsImage, antsImage-method
(antsImage-operators), 39
- <, antsImage, array-method
(antsImage-operators), 39
- <, antsImage, list-method
(antsImage-operators), 39
- <=, ANY, antsImage-method
(antsImage-operators), 39
- <=, antsImage, ANY-method
(antsImage-operators), 39
- <=, antsImage, antsImage-method
(antsImage-operators), 39
- <=, antsImage, array-method
(antsImage-operators), 39
- <=, antsImage, list-method
(antsImage-operators), 39
- <=, array, antsImage-method
(antsImage-operators), 39
- ==, ANY, antsImage-method
(antsImage-operators), 39
- ==, antsImage, ANY-method
(antsImage-operators), 39
- ==, antsImage, antsImage-method
(antsImage-operators), 39
- ==, antsImage, array-method
(antsImage-operators), 39
- ==, antsImage, list-method
(antsImage-operators), 39
- ==, array, antsImage-method
(antsImage-operators), 39
- >, ANY, antsImage-method
(antsImage-operators), 39
- >, antsImage, ANY-method
(antsImage-operators), 39
- >, antsImage, antsImage-method
(antsImage-operators), 39
- >, antsImage, array-method
(antsImage-operators), 39
- >, antsImage, list-method
(antsImage-operators), 39
- >, array, antsImage-method
(antsImage-operators), 39
- >=, ANY, antsImage-method
(antsImage-operators), 39
- >=, antsImage, ANY-method
(antsImage-operators), 39
- >=, antsImage, antsImage-method
(antsImage-operators), 39

- (antsImage-operators), 39
- >=, antsImage, array-method
 - (antsImage-operators), 39
- >=, antsImage, list-method
 - (antsImage-operators), 39
- >=, array, antsImage-method
 - (antsImage-operators), 39
- [, ANY, antsImage, ANY, ANY-method
 - ([, antsImage, NULL, ANY, ANY-method), 14
- [, antsImage, ANY, ANY, ANY-method
 - ([, antsImage, NULL, ANY, ANY-method), 14
- [, antsImage, ANY, ANY-method
 - ([, antsImage, NULL, ANY, ANY-method), 14
- [, antsImage, NULL, ANY, ANY-method
 - ([, antsImage, NULL, ANY, ANY-method), 14
- [, antsImage, NULL, NULL, ANY-method
 - ([, antsImage, NULL, ANY, ANY-method), 14
- [, antsImage, NULL, NULL-method
 - ([, antsImage, NULL, ANY, ANY-method), 14
- [, antsImage, NULL, numeric, ANY-method
 - ([, antsImage, NULL, ANY, ANY-method), 14
- [, antsImage, NULL, numeric-method
 - ([, antsImage, NULL, ANY, ANY-method), 14
- [, antsImage, antsImage, ANY, ANY-method
 - ([, antsImage, NULL, ANY, ANY-method), 14
- [, antsImage, logical, ANY, ANY-method
 - ([, antsImage, NULL, ANY, ANY-method), 14
- [, antsImage, logical, ANY-method
 - ([, antsImage, NULL, ANY, ANY-method), 14
- [, antsImage, missing, missing, ANY-method
 - ([, antsImage, NULL, ANY, ANY-method), 14
- [, antsImage, missing, missing-method
 - ([, antsImage, NULL, ANY, ANY-method), 14
- [, antsImage, missing, numeric, ANY-method
 - ([, antsImage, NULL, ANY, ANY-method), 14
- [, antsImage, missing, numeric-method
 - ([, antsImage, NULL, ANY, ANY-method), 14
- [, antsImage, numeric, NULL, ANY, ANY-method
 - ([, antsImage, NULL, ANY, ANY-method), 14
- [, antsImage, numeric, NULL-method
 - ([, antsImage, NULL, ANY, ANY-method), 14
- [, antsImage, numeric, missing, ANY-method
 - ([, antsImage, NULL, ANY, ANY-method), 14
- [, antsImage, numeric, missing-method
 - ([, antsImage, NULL, ANY, ANY-method), 14
- [, antsImage, numeric, numeric, ANY-method
 - ([, antsImage, NULL, ANY, ANY-method), 14
- [, antsImage, numeric, numeric-method
 - ([, antsImage, NULL, ANY, ANY-method), 14
- [<-, antsImage, NULL, ANY, ANY-method, 14
- [<-, antsImage, NULL, NULL, numeric-method
 - ([, antsImage, NULL, ANY, ANY-method), 14
- [<-, antsImage, NULL, antsRegion, ANY-method
 - ([, antsImage, NULL, ANY, ANY-method), 14
- [<-, antsImage, NULL, antsRegion-method
 - ([, antsImage, NULL, ANY, ANY-method), 14
- [<-, antsImage, NULL, numeric, numeric-method
 - ([, antsImage, NULL, ANY, ANY-method), 14
- [<-, antsImage, NULL-method
 - ([, antsImage, NULL, ANY, ANY-method), 14
- [<-, antsImage, antsImage, ANY, ANY-method
 - ([, antsImage, NULL, ANY, ANY-method), 14
- [<-, antsImage, antsImage-method
 - ([, antsImage, NULL, ANY, ANY-method), 14
- [<-, antsImage, array, ANY, ANY-method
 - ([, antsImage, NULL, ANY, ANY-method), 14
- [<-, antsImage, array, antsRegion, ANY-method

- ([<-, antsImage, NULL, ANY, ANY-method),
14
- [<-, antsImage, array, antsRegion-method
([<-, antsImage, NULL, ANY, ANY-method),
14
- [<-, antsImage, array-method
([<-, antsImage, NULL, ANY, ANY-method),
14
- [<-, antsImage, list, ANY, ANY-method
([<-, antsImage, NULL, ANY, ANY-method),
14
- [<-, antsImage, list-method
([<-, antsImage, NULL, ANY, ANY-method),
14
- [<-, antsImage, logical, ANY, ANY-method
([<-, antsImage, NULL, ANY, ANY-method),
14
- [<-, antsImage, logical, antsRegion, ANY-method
([<-, antsImage, NULL, ANY, ANY-method),
14
- [<-, antsImage, logical, antsRegion-method
([<-, antsImage, NULL, ANY, ANY-method),
14
- [<-, antsImage, logical-method
([<-, antsImage, NULL, ANY, ANY-method),
14
- [<-, antsImage, matrix, ANY, ANY-method
([<-, antsImage, NULL, ANY, ANY-method),
14
- [<-, antsImage, matrix, antsRegion, ANY-method
([<-, antsImage, NULL, ANY, ANY-method),
14
- [<-, antsImage, matrix, antsRegion-method
([<-, antsImage, NULL, ANY, ANY-method),
14
- [<-, antsImage, matrix-method
([<-, antsImage, NULL, ANY, ANY-method),
14
- [<-, antsImage, missing, numeric, array-method
([<-, antsImage, NULL, ANY, ANY-method),
14
- [<-, antsImage, numeric, NULL, numeric-method
([<-, antsImage, NULL, ANY, ANY-method),
14
- [<-, antsImage, numeric, missing, array-method
([<-, antsImage, NULL, ANY, ANY-method),
14
- [<-, antsImage, numeric, numeric, numeric-method
- ([<-, antsImage, NULL, ANY, ANY-method),
14
- %%, antsImage, antsImage-method
(antsImage-operators), 39
- %%, antsImage, numeric-method
(antsImage-operators), 39
- %%, numeric, antsImage-method
(antsImage-operators), 39
- %%, antsImage, antsImage-method
(antsImage-operators), 39
- %%, antsImage, numeric-method
(antsImage-operators), 39
- %%, numeric, antsImage-method
(antsImage-operators), 39
- &, antsImage, antsImage-method
(antsImage-operators), 39
- &, antsImage, logical-method
(antsImage-operators), 39
- %>%, 18
- ^, antsImage, antsImage-method
(antsImage-operators), 39
- ^, antsImage, numeric-method
(antsImage-operators), 39
- ^, numeric, antsImage-method
(antsImage-operators), 39
- aal, 18
- abpBrainExtraction, 19
- abpN4, 21
- abs, antsImage-method (antsImage-math),
36
- acos, antsImage-method (antsImage-math),
36
- acosh, antsImage-method
(antsImage-math), 36
- addNoiseToImage, 22
- adjusted_rvcoef, 23
- affineInitializer, 23
- alffmap, 24
- all (antsImage-summary), 47
- antsApplyTransforms, 26, 420
- antsApplyTransformsToPoints, 28
- antsAverageImages, 30
- antsBOLDNetworkAnalysis, 31
- antsCompileFlags, 33
- antsCopyImageInfo, 33
- antsCopyImageInfo2, 34
- antsDir, 34
- antsGetDirection (antsImageGetSet), 49

antsGetOrigin (antsImageGetSet), 49
 antsGetSpacing (antsImageGetSet), 49
 antsImage-class, 35
 antsImage-math, 36
 antsImage-operators, 39
 antsImage-summary, 47
 antsImageClone, 49
 antsImageGetSet, 49
 antsImageHeaderInfo, 51
 antsImageIterator, 52
 antsImageIterator-class, 52
 antsImageIteratorGet, 53
 antsImageIteratorGetIndex, 54
 antsImageIteratorGoToBegin, 54
 antsImageIteratorGoToReverseBegin, 55
 antsImageIteratorIsAtEnd, 55
 antsImageIteratorIsAtReverseEnd, 56
 antsImageIteratorNext, 57
 antsImageIteratorPrevious, 57
 antsImageIteratorRemaining, 58
 antsImageIteratorSet, 58
 antsImageIteratorSetIndex, 59
 antsImageMutualInformation, 60
 antsImagePhysicalSpaceConsistency, 61
 antsImageRead, 62, 64, 142
 antsImageTypeCast, 63
 antsImageWrite, 63, 64
 antsIncludes, 65
 antsLibs, 66
 antsMatrix-class, 66
 antsMotionCalculation, 67
 antsMotionCorr, 68
 antspyymm_nuisance_names, 69
 antspyymm_predictors, 69
 antspyymm_qc_names, 70
 antspyymm_vartype, 70
 antsr_mcca_features, 71
 antsr_pca_features, 72
 antsr_random_features, 73
 antsr_rgcca_baseline, 73
 antsr_rgcca_sparse, 74
 antsr_scca_features, 75
 antsRegion-class, 76
 antsRegistration, 27, 29, 76, 138, 420, 421
 antsrGetPointerName, 81
 antsrimpute, 82
 antsrMetric-class, 83
 antsrMetricCreate, 83
 antsrMetricGetDerivative, 85
 antsrMetricGetValue, 85
 antsrMetricInitialize, 86
 antsrMetricSetFixedImage, 87
 antsrMetricSetFixedImageMask, 87
 antsrMetricSetFixedTransform, 88
 antsrMetricSetMovingImage, 89
 antsrMetricSetMovingImageMask, 89
 antsrMetricSetMovingTransform, 90
 antsrMetricSetNumberOfHistogramBins, 91
 antsrMetricSetRadius, 91
 antsrMetricSetSampling, 92
 antsrMotionCalculation, 92
 antsrSurf, 94
 antsrTransform-class, 96
 antsrTransformFromDisplacementField, 97
 antsrVersions, 98
 antsrVol, 98
 antsSetDirection (antsImageGetSet), 49
 antsSetOrigin (antsImageGetSet), 49
 antsSetPixels, 100
 antsSetSpacing (antsImageGetSet), 49
 antsSpatialICAfMRI, 100
 antsTransformIndexToPhysicalPoint, 102
 antsTransformPhysicalPointToIndex, 102
 antsVersions, 103
 any (antsImage-summary), 47
 anyNA, antsImage-method
 (antsImage-operators), 39
 apply, 104
 apply_simlr_matrices, 105
 apply_simlr_matrices_dtfix, 106
 apply_transform_matrix, 107
 applyAntsrTransform, 107
 applyAntsrTransformToImage, 108
 applyAntsrTransformToPoint, 109
 applyAntsrTransformToVector, 109
 Arith, antsImage, antsImage-method
 (antsImage-operators), 39
 Arith, antsImage, array-method
 (antsImage-operators), 39
 Arith, antsImage, list-method
 (antsImage-operators), 39
 Arith, antsImage, logical-method
 (antsImage-operators), 39
 Arith, antsImage, missing-method

- (antsImage-operators), 39
- Arith, antsImage, numeric-method (antsImage-operators), 39
- Arith, array, antsImage-method (antsImage-operators), 39
- Arith, list, antsImage-method (antsImage-operators), 39
- Arith, logical, antsImage-method (antsImage-operators), 39
- Arith, numeric, antsImage-method (antsImage-operators), 39
- as.antsImage, 110
- as.antsImage, array-method (as.antsImage), 110
- as.antsImage, matrix-method (as.antsImage), 110
- as.antsMatrix, 111
- as.antsMatrix, data.frame-method (as.antsMatrix), 111
- as.antsMatrix, list-method (as.antsMatrix), 111
- as.antsMatrix, matrix-method (as.antsMatrix), 111
- as.array, 112
- as.array, antsImage-method (as.array), 112
- as.array.antsImage (as.array), 112
- as.character.antsImage (as.array), 112
- as.data.frame, 114
- as.data.frame, antsMatrix-method (as.array), 112
- as.data.frame.antsMatrix (as.array), 112
- as.factor.antsImage (as.array), 112
- as.list, antsMatrix-method (as.array), 112
- as.matrix, antsImage-method (as.array), 112
- as.matrix, antsMatrix-method (as.array), 112
- as.matrix.antsImage (as.array), 112
- as.numeric, antsImage-method (as.array), 112
- asin, antsImage-method (antsImage-math), 36
- asinh, antsImage-method (antsImage-math), 36
- aslAveraging, 114
- aslCensoring, 116
- aslDenoiseR, 118
- aslPerfusion, 120
- atan, antsImage-method (antsImage-math), 36
- atanh, antsImage-method (antsImage-math), 36
- atropos, 122, 289
- ba_svd, 124
- backup_simlr, 125
- basal_forebrain, 127
- basicInPaint, 127
- bayesianCBF, 128
- bayesianlm, 130
- bigLMStats, 132
- blockStimulus, 134
- bloodPerfusionSVD, 135
- bold_correlation_matrix, 136
- boxplot, 137
- boxplot.antsImage, 137
- boxplot.default, 137
- buildTemplate, 138
- calculate_simlr_energy, 139
- calculate_simlr_gradient, 140
- ceiling, antsImage-method (antsImage-math), 36
- cerebellum, 141
- cerebellum_labels, 141
- check_ants (check_ants-methods), 141
- check_ants, antsImage-method (check_ants-methods), 141
- check_ants, ANY-method (check_ants-methods), 141
- check_ants, character-method (check_ants-methods), 141
- check_ants, list-method (check_ants-methods), 141
- check_ants-methods, 141
- check_zero_variance, 142
- cit168_brainstem, 143
- cit168_reinf_learn, 143
- CIT168_Reinf_Learn_v1_label_descriptions_pad, 143
- CIT168_T1w_700um_pad_adni_brainstem, 144
- clip_gradient_by_quantile, 144
- clip_gradient_norm, 145
- clusterTimeSeries, 146

- coerce_mask, 147
- combineNuisancePredictors, 147
- compcor, 149
- components (pixeltype), 380
- components, antsImage-method (pixeltype), 380
- components<- (pixeltype), 380
- components<-, antsImage-method (pixeltype), 380
- composeAntsrTransforms, 150
- composeDisplacementFields, 151
- composeTransformsToField, 151
- compute_communalities, 152
- compute_variance_explained_per_factor, 153
- computeDVARs, 154
- computeDVARsSpatialMap, 154
- construct_sim_names, 155
- convolveImage, 156
- corw, 156
- cos, antsImage-method (antsImage-math), 36
- cosh, antsImage-method (antsImage-math), 36
- cospi, antsImage-method (antsImage-math), 36
- create_optimizer, 157
- createAntsrTransform, 158
- createJacobianDeterminantImage, 159
- createWarpedGrid, 160
- cropImage, 161
- cropImageCenter, 162
- cropImageFromCenterPoint, 162
- cropIndices, 163
- crossvalidatedR2, 164
- cut, 165
- cut.antsImage, 165
- cvEigenanatomy, 165
- dataAugmentation, 167
- decode_pymm_names, 169
- deconvolutionSVD, 170
- decropImage, 171
- denoiseImage, 171
- density, 173
- density.antsImage, 172
- density.default, 173
- DesikanKillianyTourville, 173
- dice_overlap_soft_matrix, 174
- dice_overlap_soft_vector, 174
- dim, antsImage-method (as.array), 112
- direction (pixeltype), 380
- direction, antsImage-method (pixeltype), 380
- direction<- (pixeltype), 380
- direction<-, antsImage-method (pixeltype), 380
- displacementFieldFromAntsrTransform, 175
- divide_by_column_sum, 176
- dkt, 176
- dkt_cit_labels, 176
- dkt_cortex_cit_deep_brain, 177
- dkt_labels, 177
- dlbs, 177
- download.file, 226
- eanatDef, 178
- eanatSelect, 179, 180
- eigSeg, 181
- ensembled_sparsity, 182
- error_not_antsImage, 183
- estimate_joint_rank, 183
- estimate_learning_rate_nsa, 184
- estimate_rank_by_permutation_rv, 185
- estSmooth, 186
- euler, 188
- exemplarInpainting, 189
- exp, antsImage-method (antsImage-math), 36
- exploratory_visualization, 190
- extend_simlr_embedding_with_new_modalities, 191
- extractImagePatchCoordinates, 193
- extractImagePatches, 194
- extractSlice, 195
- FA_JHU_labels_edited, 196
- factor.antsImage (as.array), 112
- fast_whiten, 196
- fastMarchingExtension, 197
- filterfMRIforNetworkAnalysis, 198
- fitBsplineDisplacementField, 200
- fitBsplineObjectToScatteredData, 202
- fitThinPlateSplineDisplacementField, 204
- fitTimeVaryingTransformToPointSets, 205

- fitTransformToPairedPoints, 207
- floor, antsImage-method
 - (antsImage-math), 36
- fMRINormalization, 209
- frequencyFilterfMRI, 212
- frob, 213
- fsl2antsrTransform, 214
- functionalLungSegmentation, 214
- fusedRidge, 216
- fusedRidgeDirect, 218
- fusedRidgeTorch, 220
- fuzzySpatialCMeansSegmentation, 222
- gamma, antsImage-method
 - (antsImage-math), 36
- generate_structured_multiview_data, 223
- generateAifMaskImage, 224
- geoSeg, 225
- getANTsRData, 226
- getAntsrTransformFixedParameters, 227
- getAntsrTransformParameters, 228
- getASLNoisePredictors, 228
- getAverageOfTimeSeries, 229
- getCenterOfMass, 230
- getCentroids, 231
- getfMRIInuisanceVariables, 232
- getMask, 233, 261, 263, 315
- getMultiResFeatureMatrix, 234, 336
- getMultivariateTemplateCoordinates, 235
- getNameFromDataframe, 236
- getNeighborhoodAtVoxel, 237
- getNeighborhoodInMask, 238
- getPixels, 240
- getTemplateCoordinates, 241
- glm_impute, 242
- gradient_invariant_orthogonality_defect, 243
- gradient_invariant_orthogonality_salad, 244
- hausdorffDistance, 245
- hemisphere, 245
- hemodynamicRF, 246
- hessianObjectness, 249
- hist, 250
- hist.antsImage, 250
- hist.default, 250
- histogramEqualizeImage, 251
- histogramMatchImage, 251
- histogramMatchImage2, 252
- histogramWarpImageIntensities, 253
- humanize_anatomical_name, 254
- iBind, 255
- icawhiten, 255
- ilr, 256, 258
- ilr.predict, 257
- image2ClusterImages, 258
- imageDomainToSpatialMatrix, 259
- imageFileNames2ImageList, 260
- imageListToMatrix, 260
- imageSimilarity, 262
- imagesToMatrix, 263, 315
- iMath, 20, 233, 264
- iMathOps, 265
- indicator_opt_both_ways, 266
- initialize, antsImage-method
 - (antsImage-class), 35
- initialize, antsImageIterator-method
 - (antsImageIterator-class), 52
- initialize, antsMatrix-method
 - (antsMatrix-class), 66
- initialize, antsrMetric-method
 - (antsrMetric-class), 83
- initialize, antsrTransform-method
 - (antsrTransform-class), 96
- initializeEigenanatomy, 266
- initializeSimlr, 268
- integrateVectorField, 269
- integrateVelocityField, 270
- interleaveMatrixWithItself, 271
- interpret_simlr_vector, 272
- interpret_simlr_vector2, 273
- inv_sqrt_sym, 274
- inv_sqrt_sym_newton, 275
- invariant_orthogonality_defect, 275
- invertAntsrTransform, 276
- invertDisplacementField, 276
- is.antsImage, 277
- is.finite, antsImage-method
 - (antsImage-operators), 39
- is.infinite, antsImage-method
 - (antsImage-operators), 39
- is.na, antsImage-method
 - (antsImage-operators), 39

- is.nan, antsImage-method
(antsImage-operators), 39
- jhu_fa_labels, 278
- jhu_md_labels, 278
- jlfProp, 278
- joinEigenanatomy, 280
- jointLabelFusion, 279, 281
- jointSmoothMatrixReconstruction, 284
- kellyKapowski, 287
- kmeansSegmentation, 289
- knnSmoothImage, 290
- knnSmoothingMatrix, 291
- l1_normalize_features, 292
- labelClusters, 293
- labelGeometryMeasures, 294
- labelImageCentroids, 295
- labelImageRegistration, 296
- labelOverlapMeasures, 298
- labels2matrix, 299
- labelStats, 300
- lappend, 300
- lgamma, antsImage-method
(antsImage-math), 36
- list_simlr_optimizers, 301
- load.ANTsR, 302
- lobes, 302
- lobes_brainstem, 303
- localGyrificationIndex, 303
- localJointLabelFusion, 304
- log, antsImage-method (antsImage-math),
36
- log10, antsImage-method
(antsImage-math), 36
- log2, antsImage-method (antsImage-math),
36
- Logic, antsImage, antsImage-method
(antsImage-operators), 39
- Logic, antsImage, ANY-method
(antsImage-operators), 39
- Logic, antsImage, array-method
(antsImage-operators), 39
- Logic, antsImage, list-method
(antsImage-operators), 39
- Logic, antsImage, logical-method
(antsImage-operators), 39
- Logic, antsImage, numeric-method
(antsImage-operators), 39
- Logic, ANY, antsImage-method
(antsImage-operators), 39
- Logic, array, antsImage-method
(antsImage-operators), 39
- Logic, list, antsImage-method
(antsImage-operators), 39
- Logic, logical, antsImage-method
(antsImage-operators), 39
- Logic, numeric, antsImage-method
(antsImage-operators), 39
- lowrankRowMatrix, 306
- make3ViewPNG, 306
- makeGraph, 307
- makeImage, 308
- makePointsImage, 309
- makePowersPointsImage, 310
- mapAsymVar, 311
- mapLRAverageVar, 312
- mask_values, 312
- maskImage, 313
- Math, antsImage-method (antsImage-math),
36
- matrix2timeseries, 314
- matrixToImages, 261, 263, 315
- max (antsImage-summary), 47
- max, antsImage-method
(antsImage-summary), 47
- MD_JHU_labels_edited, 315
- mean, 316
- mean.antsImage, 316
- median.antsImage, 317
- mergeChannels, 317
- mergeListToNDImage, 318
- mild, 319, 466
- milr, 320, 321, 324, 466
- milr.predict, 323
- min (antsImage-summary), 47
- min, antsImage-method
(antsImage-summary), 47
- mni2tal, 325
- morphology, 326
- mrvmrfs, 327, 336, 433
- mrvmrfs.predict, 329
- mtl_description, 330
- mtl_labels, 331
- multichannelPCA, 331

- multichannelToVector, 332
- multigrep, 333
- multiLabelMorphology, 334
- multiResRandomForestRegression, 335
- multiscaleSVD, 338
- multiview_pca, 339

- n3BiasFieldCorrection, 341
- n3BiasFieldCorrection2, 341
- n4BiasFieldCorrection, 21, 343
- nbm3_labels, 345
- nbm3CH13, 345
- neg_violation, 346
- networkEiganat, 346
- NNHEmbed, 349
- NNHEmbed_update_residuals, 351
- nsa_flow, 352
- nsa_flow_autograd, 354
- nsa_flow_fa_diagram, 355
- nsa_flow_flowchart, 356
- nsa_flow_pca, 357
- nsa_flow_pca_fa, 358
- nsa_flow_pca_fa_2, 359
- nsa_flow_retract, 361
- nsa_flow_retract_auto, 362
- nsa_flow_torch, 362
- nsa_flow_torch_ag, 365

- oneHotToSegmentation, 367
- optimal_simlr_initializer, 368
- optimize_indicator_matrix, 369
- origin(pixeltype), 380
- origin,antsImage-method(pixeltype), 380
- origin<- (pixeltype), 380
- origin<- ,antsImage-method(pixeltype), 380
- orth_residual, 370
- orthogonalize_feature_space, 371
- orthogonalizeAndQSparsify, 372
- orthogonalizeAndQSparsifyOld, 373

- padImageByFactor, 374
- padOrCropImageToSize, 374
- pairwise_matrix_similarity, 375
- pairwiseImageDistanceMatrix, 376
- partialVolumeCorrection, 377
- perfusionregression, 378
- petSUVR, 379
- pixeltype, 380
- pixeltype,antsImage-method(pixeltype), 380
- pixeltype<- (pixeltype), 380
- pixeltype<- ,antsImage-method(pixeltype), 380
- plot.antsImage, 382
- plot_energy_decomposition, 385
- plot_features, 386
- plotBasicNetwork, 387
- plotPrettyGraph, 390
- posthoc_fa_summary, 391
- powers_areal_mni_itk, 392
- powers_mni_itk, 393
- ppmi_template_500Parcels_Yeo2011_17Networks_2023_homotopic, 394
- ppmi_yeo_labels, 394
- prcomp, 449
- predict.fusedRidge, 394
- predict.fusedRidgeDirect, 395
- predict.fusedRidgeTorch, 396
- predictSimlr, 397
- preprocessfMRI, 398
- preprocessRestingBOLD, 400
- priorBasedSegmentation, 402
- prod(antsImage-summary), 47
- project_to_orthonormal_nonnegative, 403
- project_to_partially_orthonormal_nonnegative, 403

- quantifyCBF, 405
- quantifySNPs, 406
- quantile, 407
- quantile.antsImage, 407

- randcca, 408
- randomlyTransformImageData, 409
- randomMask, 411, 443
- randsvd, 411
- randwhiten, 412
- range(antsImage-summary), 47
- rankBasedMatrixSegmentation, 413
- rankIntensity, 414
- rapidlyInspectImageData, 415
- read_simlr, 416
- read_simlr_data_frames, 416
- readAntsrTransform, 417
- readNormalizedPopulationData, 418
- reconstructImageFromPatches, 419

- refbasis_brain, [420](#)
- refbasis_head, [420](#)
- reference_basis, [420](#)
- reflectImage, [420](#)
- reflectionMatrix, [421](#)
- regressionMatchImage, [422](#)
- regressProjections, [423](#)
- regularizeSimlr, [425](#)
- reho, [426](#)
- renormalizeProbabilityImages, [427](#)
- reorientImage, [428](#)
- resampleImage, [428](#)
- resampleImageToTarget, [429](#)
- resels, [431](#)
- reset_optimizer_state, [432](#)
- rfSegmentation, [432](#)
- rfSegmentationPredict, [434](#)
- rftPval, [435](#)
- rftResults, [436](#)
- ri, [439](#)
- ripmmarc, [440](#), [443](#)
- ripmmarcBasisImage, [442](#)
- ripmmarcPop, [443](#)
- robustMatrixTransform, [444](#)
- round, antsImage-method
(antsImage-math), [36](#)
- rsfDenoise, [445](#)
- rvcoef, [447](#)
- rvcoef_components, [448](#)
- rvcoef_gram_impl, [448](#)
- rvcoef_trace_impl, [449](#)

- safe_pca, [449](#)
- save.ANTsR, [450](#)
- scaleSpaceFeatureDetection, [451](#)
- sd, [452](#), [452](#)
- segmentationRefinement.predict, [453](#)
- segmentationRefinement.train, [455](#)
- segmentationToOneHot, [458](#)
- segmentShapeFromImage, [459](#)
- select_joint_k, [460](#)
- set.seed, [443](#)
- setANTsDeterministic, [461](#)
- setAntsrTransformFixedParameters, [462](#)
- setAntsrTransformParameters, [97](#), [462](#)
- setSeedBasedOnTime, [463](#)
- shorten_pymm_names, [463](#)
- show, antsImage-method
(antsImage-class), [35](#)
- show, antsImageIterator-method
(antsImageIterator-class), [52](#)
- show, antsrMetric-method
(antsrMetric-class), [83](#)
- show, antsrTransform-method
(antsrTransform-class), [96](#)
- sign, antsImage-method (antsImage-math),
[36](#)
- simlr, [464](#), [476](#)
- simlr.parameters, [467](#)
- simlr.perm, [469](#)
- simlr.search, [471](#)
- simlr_feature_orth, [472](#)
- simlr_impute, [473](#)
- simlr_path_models, [474](#)
- simlr_sparseness, [474](#)
- simlrU, [466](#), [475](#)
- simulateBiasField, [477](#)
- simulateDisplacementField, [478](#)
- sin, antsImage-method (antsImage-math),
[36](#)
- sinh, antsImage-method (antsImage-math),
[36](#)
- sinpi, antsImage-method
(antsImage-math), [36](#)
- skeletonize, [479](#)
- smoothAppGradCCA, [480](#)
- smoothImage, [481](#)
- smoothMatrixPrediction, [482](#)
- smoothMultiRegression, [484](#)
- smoothRegression, [485](#)
- softwareVersions, [487](#)
- spacing (pixeltype), [380](#)
- spacing, antsImage-method (pixeltype),
[380](#)
- spacing<- (pixeltype), [380](#)
- spacing<- , antsImage-method (pixeltype),
[380](#)
- sparseDecom, [487](#)
- sparseDecom2, [424](#), [490](#)
- sparseDecom2boot, [493](#)
- sparseDecomboot, [496](#)
- sparseDistanceMatrix, [498](#)
- sparseDistanceMatrixXY, [499](#)
- sparseRegression, [501](#)
- sparsify_by_column_winner, [503](#)
- spatialbayesianlm, [504](#)
- splitChannels, [505](#)

splitData, 506
 splitMask, 507
 splitNDImageToList, 508
 sqrt, antsImage-method (antsImage-math), 36
 statFieldThresh, 508
 step, 510
 subgradientL1Regression, 511
 subjectDataToGroupDataFrame, 512
 subset.antsImage, 513
 sum (antsImage-summary), 47
 sum_preserving_matrix_partition, 513
 Summary, antsImage-method (antsImage-summary), 47
 symm, 514

 T_template0_LobesBstem, 515
 take_abs_unsigned, 515
 tan, antsImage-method (antsImage-math), 36
 tanh, antsImage-method (antsImage-math), 36
 tanpi, antsImage-method (antsImage-math), 36
 taskFMRI, 516
 temporalwhiten, 517
 testEanat, 518
 thresholdImage, 519
 timeseries2matrix, 520
 timeseriesN3, 520
 timeserieswindow2matrix, 521
 tissues, 523
 tracts, 523
 transform_matrix, 524
 transform_prior_rows_to_uncorrelated, 525
 trapz, 526
 trunc, antsImage-method (antsImage-math), 36

 undo_transform_matrix, 527
 unique, 527
 unique.antsImage, 527
 usePkg, 528

 var, 528, 529
 vector_to_df, 529
 vectorToMultichannel, 530
 visualize_lowrank_relationships, 531
 visualize_permutation_test, 531
 vwnrfs, 328, 532
 vwnrfs.predict, 534

 weingartenImageCurvature, 535
 which, antsImage-method (antsImage-operators), 39
 whiten, 536
 whiten_matrix, 537
 wm_major_tracts, 538
 wmh_evidence, 538
 write_simlr, 416, 539
 write_simlr_data_frames, 539
 writeAntsrTransform, 540
 writeNormalizedPopulationData, 540