Package 'LESYMAP'

March 15, 2019

Title Leions to Symptom Mapping in R

Version 0.0.0.9210

Date 2019-03-15

Description LESYMAP maps the specific brain areas responsible for cognitive deficits by taking a series of lesion maps and a vector of behavioral scores. Both univariate (t-test, Brunner-Munzel, regression) and multivariate (sparse canonical corelations) tests are available. LESYMAP is built to run both real and simulated lesion-to-symptom mapping analyses.

License Apache License 2.0 | file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 3.0),
ANTsR

Imports ANTsRCore,
graphics,
lmPerm,
Rcpp,
stats,
utils

Suggests nparcomp,
e1071

Remotes ANTsX/ANTsR,
ANTsX/ANTsRCore

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

RoxygenNote 6.1.0.9000

R topics documented:

`.createFolds` *createFolds*

Description

Used to create folds for k-fold validation

Usage

```
.createFolds(y, k = 10, list = TRUE, returnTrain = FALSE)
```

Arguments

`y`	split sample by balancing y
`k`	number of folds
`list`	logical whether to return folds in a list
`returnTrain`	logical whether to return training indices (T) or the test samples (F)

Author(s)

Caret Package

`BM` *Massive Brunner-Munzel tests*

Description

Takes a binary matrix of voxels and a vector of behavior and runs Brunner-Munzel tests on each voxel. This function is not compiled and is slow.

Usage

```
BM(lesmat, behavior)
```

Arguments

`lesmat`	matrix of voxels
`behavior`	vector of behavior

Value

Returned list with:

- `statistic` statistical values
- `dof` degrees of freedom

Author(s)

Dorian Pustina

Examples

```
set.seed(123)
lesmat = matrix(rbinom(200,1,0.5), ncol=2)
set.seed(123)
behavior = rnorm(100)
result = BM(lesmat, behavior)
```

BMfast *Fast Brunner-Munzel tests (v1)*

Description

Takes a binary matrix of voxels and a vector of behavior and runs Brunner-Munzel tests on each voxel. This is a fast function, but may produce infinite values for perfectly separated group. Use BMfast2 which avoids this problem.

Usage

```
BMfast(X, y)
```

Arguments

X	binary matrix ov voxlels (columns) for all subjects (rows)
y	vector of behavioral scores.

Value

List with two vectors: - statistic - BM values - dfbm - degrees of freedom

Author(s)

Dorian Pustina

/@export

BMfast2 *Fast Brunner-Munzel tests (v2)*

Description

Takes a binary matrix of voxels and a vector of behavior and runs Brunner-Munzel tests on each voxel. This is a fast function that corrects for infinite values with a similar approach as the nparcomp package.

Usage

```
BMfast2(X, y, computeDOF = TRUE)
```

Arguments

X	binary matrix ov voxlels (columns) for all subjects (rows)
y	vector of behavioral scores.
computeDOF	(true) chooses whether to compute degrees of freedom. Set to false to save time during permutations.

Value

List with two vectors:

- `statistic` - BM values
- `dfbm` - degrees of freedom

Author(s)

Dorian Pustina

Examples

```
set.seed(1234)
lesmat = matrix(rbinom(40,1,0.2), ncol=2)
set.seed(1234)
behavior = rnorm(20)
test = LESYMAP::BMfast2(lesmat, behavior)
test$statistic[,1] # -2.0571825 -0.8259754
test$dfbm[,1] # 16.927348  7.563432
```

`BMperm` *Fast Brunner-Munzel tests (v2) with permutations*

Description

Takes a binary matrix of voxels and a vector of behavior and runs Brunner-Munzel tests on each voxel. This is a fast function that corrects for infinite values with a similar approach as the nparcomp package. It calculates p-values by running permutations of each voxel and using the ratio of times the real BM score exceeds the permuted BM score.

Usage

```
BMperm(X, y, computeDOF = TRUE, npermBM = 20000L, alternative = 1L)
```

Arguments

`X`	binary matrix ov voxlels (columns) for all subjects (rows)
`y`	vector of behavioral scores.
`computeDOF`	(default true) chooses whether to compute degrees of freedom. Set to false to save time during permutations.
`npermBM`	(default 20000) number of permutations to run at each voxel
`alternative`	(default 1) integer to select the tail of pvalues. 1-greater, 2-less, 3-two.sided

Value

List with these objects:

- `statistic` - BM values
- `dfbm` - degrees of freedom
- `pvalue` - permutation-based probability value

Author(s)

Dorian Pustina

Examples

```
set.seed(1234)
lesmat = matrix(rbinom(40,1,0.2), ncol=2)
set.seed(1234)
behavior = rnorm(20)
test = LESYMAP::BMperm(lesmat, behavior, alternative=3)
test$statistic[,1] # -2.0571825 -0.8259754
test$dfbm[,1] # 16.927348  7.563432
test$pvalue[,1] # 0.1427929 0.4102795
```

`checkAntsInput` *checkAntsInput*

Description

Function to check a variable whether is composed of an antsImage, list of antsImages, or simply filenames. If none of the above, an error is returned.

Usage

```
checkAntsInput(input, checkHeaders = F)
```

Arguments

`input`	the variable to be checked
`checkHeaders`	make sure all images have the same headers

Value

Type of variable (antsImage, antsImageList, antsFiles) or error if variable cannot be established.

Author(s)

Dorian Pustina

Examples

```
## Not run:
        files = Sys.glob('/data/jag/nifti/*.nii.gz')
         myimagelist = imageFileNames2ImageList(files)
         checkAntsInput(myimagelist) # returns 'antsImageList'
         checkAntsInput(antsFiles) # returns 'antsFiles'
         checkAntsInput(myimagelist[[1]]) # returns 'antsImage'

## End(Not run)
```

`checkFilenameHeaders` *checkFilenameHeaders*

Description

Function to check that all filenames in a vector point to existing files with the same resolution, orientation, size, and origin.

Usage

```
checkFilenameHeaders(files, showError = T)
```

Arguments

`files`	character vector of filenames
`showError`	logical whether to show an error (True) or to return a boolean instead. Returned values are True=pass,False=Fail

Value

logical if the test was successful or not

Author(s)

Dorian Pustina

checkImageList *checkImageList*

Description

Function to check that all antsImages in a list have the same orientation, origin, and resolution. The function stops with an error if one of the images has unusual headers. This behavior can be overcome by setting showError=F, and using the returned status (True=pass, False=fail) to make decisions outside this function.

Usage

```
checkImageList(imgList, showError = T, binaryCheck = F)
```

Arguments

imgList	list of antsImages
showError	boolean indicating whether to show the exact error and interrupt the function (TRUE, default), or don't show the error and return the check status (FALSE). The returned values when showError=F are T=passed or F=Failed.
binaryCheck	boolean, check if images are binary (0/1 values). Useful when checking masks or lesions. This check slows the output of the function.

Value

True if list has images with same headers, otherwise False.

Author(s)

Dorian Pustina

Examples

```
## Not run:
files = Sys.glob('/data/jag/nifti/*.nii.gz')
myimagelist = imageFileNames2ImageList(files)
checkImageList(myimagelist) # no value returned
checkImageList(lesions, showError=F) # True returned
myimagelist[[4]] = cropIndices(myimagelist[[4]], c(1,1,1), c(20,20,20))
checkImageList(myimagelist) # error on image 4

## End(Not run)
```

checkMask *checkMask*

Description

Function to check if mask is in the same space as inputs

Usage

```
checkMask(lesions.list, mask)
```

Arguments

lesions.list	list of antsImages or character vector of filenames
mask	antsImage of mask to check

Value

Nothing is returned, function stops with error if mask is not in the same space as images in lesions.list

Author(s)

Dorian Pustina

getLesionLoad *getLesionLoad*

Description

Computes lesion loads from a series of images. A parcellation image (or simple mask) is required to define the regions from which to compute the lesion load.

Usage

```
getLesionLoad(lesions.list, parcellation, label = NA, mask = NA,
  binaryCheck = F, keepAllLabels = F, minSubjectPerLabel = "10%")
```

Arguments

lesions.list	list of antsImages or filenames. Must be binary (0 and 1 values).
parcellation	ansImage or filename of the parcellated volumes. A parcellation is an image brain regions showned as with integer values (i.e. ,1,2,3,...).
label	(default=NA) you can ask to get output for a specific label in the parcellation volume (i.e., label=122).
mask	(default=NA) if this mask is specified (antsImage or filename) lesioned voxels outside the mask are ignored. This is not a good choice, but in case you need it its there.
binaryCheck	(default=FALSE) check whether lesion maps are binary (0/1). Will output an error if lesion files are not binary.
keepAllLabels	(default=FALSE) by default labels are removed if affected in just few subjects. Setting this to TRUE will keep all labels.
minSubjectPerLabel	minimum number of subjects a parcel must be lesioned to keep and return it.

Value

- outputMatrix of lesion loads between 0 and 1. 1 means 100% lesioned. Each column is a single parcel and each row a single subject. Parcel numbers are placed as column names.

Author(s)

Dorian Pustina

Examples

```
lesydata = file.path(find.package('LESYMAP'),'extdata')
filenames = Sys.glob(file.path(lesydata, 'lesions', '*.nii.gz'))
lesions = imageFileNames2ImageList(filenames[1:10])
parcellation = antsImageRead(
file.path(lesydata,'template', 'Parcellation_403areas.nii.gz'))
lesload = getLesionLoad(lesions, parcellation)
```

---

getLesionSize *getLesionSize*

Description

Compute lesion sizes from a list of antsImages.

Usage

```
getLesionSize(lesions.list, showInfo = TRUE)
```

Arguments

lesions.list	List of antsImages or vector of filenames. It is assumed that images are binary (0/1).
showInfo	logical show or not informations/warnings

Value

vector of lesion sizes in mm3

Author(s)

Dorian Pustina

---

getUniqueLesionPatches

Unique Lesion Patches

Description

Compute uniqe patches of voxels with the same pattern of lesions in all subjects. Useful to understand the number of patterns that will be analyzed in a lesion dataset. A patch is a group of voxels, not necessarily close to each other, which have the same identical lesion pattern.

Usage

```
getUniqueLesionPatches(lesions.list, mask = NA, returnPatchMatrix = F,
  thresholdPercent = 0.1, binaryCheck = F, showInfo = T)
```

Arguments

lesions.list	list of antsImages (faster) or filenames (slower)
mask	(default=NA) a mask image to restrict the search for patches. Will be automatically calculated if not provided. Normally the mask restricts the search only to voxels lesioned in >10% of subejcts. To set this proportion use thresholdPercent.
returnPatchMatrix	(default=FALSE) logical, should the matrix of patches be returned. This is used in lesymap to run the analyses.
thresholdPercent	(default=0.1) voxels with lesions in less than this proportion of subjects will not be considered. I.e., 0.1 = 10%.
binaryCheck	(default=FALSE) set this to TRUE to verify that maps are binary.
showInfo	(default=TRUE) logical indicating whether to display information.

Value

List of objects named as follows:

- patchimg - antsImage with every voxel assigned a patch number
- patchimg.samples - antsImage mask of one representative voxel for each patch. Can be used to extract the patchmatrix.
- patchimg.size - antsImage with the patch size at every voxel
- patchimg.mask - antsImage of the mask used to extract patches. Can be used to put back results when combined with patchindx.
- patchindx - vector of patch membership for each voxel. Can be used to put back results in an image.
- npatches - number of unique patches in the image
- nvoxels - total number of lesioned voxels in patchimg.mask
- patchvoxels - vector of voxel count for each patch
- patchvolumes - vector of volume size for each patch
- patchmatrix - matrix of patches. This is used in lesymap to save time when running repetitive analyses.

Author(s)

Dorian Pustina

Examples

```
lesydata = file.path(find.package('LESYMAP'),'extdata')

filenames = Sys.glob(file.path(lesydata, 'lesions', '*.nii.gz'))
patchinfo = getUniqueLesionPatches(filenames[1:10]) # slower

lesions = imageFileNames2ImageList(filenames[1:10])
patchinfo = getUniqueLesionPatches(lesions) # faster
```

---

lesyload_mricron *lesyload_mricron*

Description

Function to load data from a previous analysis in MRIcron/npm in a ready format for use in lesymap

Usage

```
lesyload_mricron(valfile, imageFolder = NA, returnFilenames = F,
  checkHeaders = T, showInfo = T)
```

Arguments

`valfile`	mricron filename with extention *.val. The function will search for images in the same folder where valfile is located, unless you specify `imageFolder`. If any of the files listed in the .val file are not found in the folder, an error will be displayed.
`imageFolder`	(default=NA) folder to look for the image files
`returnFilenames`	(default=FALSE) By default the function will load the images in memory to speed up things in lesymap. This may require too much RAM memory in some cases, and you may want to use filenames instead, which requires less memory but is slower in lesymap.
`checkHeaders`	(default=TRUE) Headers will be checked to make sure all images have the same dimension/origin/resolution, etc.
`showInfo`	(default=TRUE) show information upon successful load

Value

List with the following information lesions - list of antsImages or vector of filenames behavior - vector of behavioral scores

Author(s)

Dorian Pustina

`lesymap` *Lesion to Symptom Mapping*

Description

Lesymap uses univariate and multivariate methods to map functional regions of the brain that, when lesioned, cause specific cognitive deficits. It requires is a set of binary lesion maps (nifti files) in template space and the vector of corresponding behavioral scores. Note, lesions must be already registered in template space, use the built-in function `registerLesionToTemplate` or other ANTs tools to register lesions. Lesymap will check that lesion maps are in the same space before running. For traditional mass-univariate analyses (i.e., BMfast, ttest, etc.), voxels with identical lesion patterns are grouped together in unique patches. Patch-based analysis decreases the number of multiple comparisons and speeds up the analyses. Multivariate analysis are performed using an optimized version of sparse canonical correlations (SCCAN, `method='sccan'`)or support vector regression (`method='svr'`).

Usage

```
lesymap(lesions.list, behavior, mask = NA, patchinfo = NA,
  method = "sccan", correctByLesSize = "none",
  multipleComparison = "fdr", pThreshold = 0.05, flipSign = F,
  minSubjectPerVoxel = "10%", nperm = 1000, saveDir = NA,
  binaryCheck = FALSE, noPatch = FALSE, showInfo = TRUE, ...)
```

Arguments

`lesions.list`	list of antsImages, or a vector of filenames, or a single antsImage with 4 dimensions.
`behavior`	vector of behavioral scores or filename pointing to a file with a single column of numbers.
`mask`	(default=NA) binary image to select the area where analysis will be performed. If not provided will be computed automatically by thresholding the average lesion map at `minSubjectPerVoxel`.
`patchinfo`	(default=NA) an object obtained with the `getUniqueLesionPatches` function. Useful to set if your run repetitive analysese and want toavoid the computation of patches each time. You can also pass the patchinfo object obtained from a previous analysis.
`method`	what analysis method to use to run, one of 'BM', 'BMfast', 'ttest', 'welch', 'regres', 'regresfast', 'regresPerm', 'sccan' (default) or 'svr',. `BM` - Brunner-Munzel non parametric test, also called the Generalized Wilcoxon Test. The BM test is the same test used in the npm/Mricron software (see Rorden (2007)). This method is slow, use 'BMfast" for a compiled faster analysis. `BMfast` - ultrafast Brunner-Munzel with compiled code. BMfast can be combined with `multipleComparison='FWERperm'` to perform permutation based thresholding in a short time. `ttest` - Regular single tailed t-test. Variances of groups are assumed to be equal at each voxel, which may not be true. This is the test used in the voxbo software. Relies on `t.test` function in R. It is assumed that 0 voxels are healthy, i.e., higher behavioral scores. See the `alternative` parameter for inverted cases. (see Bates (2003)). `welch` - t-test that does not assume equal variance between groups. Relies on `t.test` function in R. `regres` - linear model between voxel values and behavior. Uses the `lm` function in R. This is equivalent to a t-test, but is useful when voxel values are continuous. This method is R-based and slow, for faster analysis and to add covariates use the `"regresfast"` method compiled in LESYMAP. `regresfast` - fast linear regressions with compiled code. This method allows setting covariates. If covariates are specified the effect of each voxel will be estimated with the formula: `behavior ~ voxel + covar1 + covar2 + ...` The effect of covariates at each voxel is established with the Freedman-Lane method (see Winkler (2014)). This LESYMAP method allows multiple comparison correction with permutation based methods `"FWERperm"` and `"clusterPerm"`. `regresPerm` - linear model between voxel values and behavior. The p-value of each individual voxel is established by permuting voxel values. The lmPerm package is used for this purpose. Note, these permutations do not correct for multiple comparisons, they only establish voxel-wise p-values, a correction for multiple comparisons is still required. `chisq` - chi-square test between voxel values and behavior. The method is used when behavioral scores are binary (i.e. presence of absence of deficit). Relies on

the chisq.test R function. By default this method corrects individual voxel p-values with the Yates method (the same approach offered in the Voxbo software).

chisqPerm - chi-square tests. P-values are established through permutation tests instead of regular statistics. Relies on the chisq.test R function.

sccan - sparse canonical correlations. Multivariate method that considers all voxels at once by searching for voxel weights that collectively explain behavioral variance. For our purposes, this method can be considered a sparse regression technice. By default, lesymap will run a lengthy procedure to determine the optimal sparseness value (how extensive the results should be). You can set optimizeSparseness=FALSE if you want to skip this optimization. The search for optimal sparsness provides a cross-validated correlation measure that shows how well the sparseness value can predict new patients. If this predictive correlation is below significance (i.e., below pThreshold), the entire solution will be dropped and LESYMAP will return an empty statistical map. If the predictive correlation is significant, LESYMAP will return a statistical image with normalized weights between -1 and 1. The raw SCCAN weights image is returned in rawWeights.img as well. LESYMAP scales and centers both lesion and behavior data before running SCCAN (hardcoded in lsm_sccan). See more details in Pustina (2018)

svr - support vector regression. Multivariate method that considers all voxels at once by searching for voxel weights. To establish p-values, a number of permutations are needed as set with SVR.nperm. The current implementation of SVR in LESYMAP is not parallelized and takes many hours to finish all permutations. The SVR method is initially described in Zhang (2014), LESYMAP uses a contribution by the Tuebingen group.

`correctByLesSize`

whether to correct for lesion size in the analysis. Options are "none", "voxel", "behavior":

- "none": (default) no correction
- "voxel": divide voxel values by 1/sqrt(lesionsize). This is the method used in Mirman (2015) and Zhang (2014). This correction works only with 'regres' methods. Two sample comparisons (t-tests and Brunner-Munzel) use binary voxels and will ignore this correction.
- "behavior": residualize behavioral scores by removing the effect of lesion size. This works on all methods, but is more agressive on results.

`multipleComparison`

(default='fdr') method to adjust p-values. Standard methods include "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr". (see p.adjust)
Permutation methods include:
"FWERperm" (permutation based family-wise threshold) is enabled with methods 'BMfast' and 'regresfast'. In this case, many analysis are run with permuted behavioral scores, and the peak score is recorded each time (see Winkler 2014). The optimal threshold is established at 95th percentile of this distribution (or whatever pThreshold you choose). You can choose to use as reference another voxel lower in the ranks by specifying another 'v' value (i.e., lesymap(..., v=10) will record the 10th highest voxel).
"clusterPerm" (permutation based cluster correction) is enabled for 'regresfast'. It records the maximal cluster size from many random permutations of

	the behavior score and sets a cluster threshold based on that distribution. You must select pThreshold (voxel-wise, default=0.05) and clusterPermThreshold (cluster-wise, default 0.05) to achieve optimal thresholding with this approach.
`pThreshold`	(default=0.05) threshold statistics at this p-value (after corrections or permutations)
`flipSign`	logical (default=FALSE), invert the sign in the statistics image.
`minSubjectPerVoxel`	(default='10%') remove voxels/patches with lesions in less than X subjects. Value can be speficifed as percentage ('10%') or exact number of subjects (10).
`nperm`	(default=1000) number of permutations to run when necessary. This is used mostly for univariate analyses, while multivariate methods have their own permutation arguments. Check the documentation of each method to know more.
`saveDir`	(default=NA) save results in the specified folder.
`binaryCheck`	logical (default=FALSE), make sure the lesion matrix is 0/1. This will help if lesion maps are drawn in MRIcron or other software which label lesioned voxel with value 255.
`noPatch`	logical (default=FALSE), if True avoids using patch information and will analyze all voxels. It will take longer and results will be worse due to more multiple comparison corrections. This argument is ignored when performing multivariate analyses, SCCAN or SVR, for which all voxels are always used.
`showInfo`	logical (default=TRUE), display time-stamped info messages
`...`	arguments that will be passed down to other functions (i.e., sparsness=0.045)

Details

Several other parameters can be specified to lesymap() which will be passed to other called fuctions. Here are some examples:

`permuteNthreshold` - (default=9) for Brunner-Munzel tests in `method='BMfast'` or `method='BM'`. Voxels lesioned in less than this number of subjects will undergo permutation-based p-value estimation. Useful because the BM test is not valid when comparing groups with N < 9. Issue described in: Medina (2010)

`clusterPermThreshold` - threshold used to find the optimal cluster size when using `multipleComparison='clusterPerm'`.

`alternative` - (default='greater') for two sample tests (ttests and BM). By default LESYMAP computes single tailed p-values assuming that non-lesioned 0 voxels have higher behavioral scores. You can specify the opposite relationship with alternative='less' or compute two tailed p-values with alternative='two.sided'.

`covariates` - (default=NA) enabled for method = 'regresfast'. This will allow to model the effect of each voxel in the context of other covariates, i.e., formula "behavior ~ voxel + covar1 + covar2 + ...".
I,.e., lesymap(lesions,behavior, method='regresfast', covariates=cbind(lesionsize, age)).
If you choose permutation based thresholding with covariates, lesymap will use the Freedman-Lane method for extracting the unique effect of each voxel (see Winkler 2014, Freedman 1983)

`template` - antsImage or filename used for plotting the results if a saving directory is specified (see `saveDir`)

`v` - (default=1) which voxel to record for permutation based thresholding. Normally the peak voxel is used (1), but other voxels can be recorded. See Mirman (2017) for this approach.

Value

The following objects are typically found in the returned list:

- `stat.img` - statistical map
- `rawWeights.img` - (optional) raw SCCAN weights
- `pval.img` - (optional) p-values map
- `zmap.img` - (optional) zscore map
- `mask.img` - mask used for the analyses
- `average.img` - map of all lesions averaged, produced only if no mask is defined.
- `callinfo` - list of details of how you called lesymap
- `outputLog` - terminal output in a character variable
- `perm.vector` - (optional) the values obtained from each permutation
- `perm.clusterThreshold` - (optional) threshold computed for cluster thresholding
- `perm.FWERthresh` - (optional) threshold computed for FWERperm thresholding
- `patchinfo` - list of variables describing patch information:
  - `patchimg` - antsImage with the patch number each voxels belongs to
  - `patchimg.samples` - antsImage mask with a single voxel per patch
  - `patchimg.size` - antsImage with the patch size at each voxel
  - `patchimg.mask` - the mask within which the function will look for patches
  - `npatches` - number of unique patches in the image
  - `nvoxels` - total number of lesioned voxels in mask
  - `patchvoxels` - vector of voxel count for each patch
  - `patchvolumes` - vector of volume size for each patch
  - `patchmatrix` - the lesional matrix, ready for use in analyses. Matrix has size NxP (N=number of subjects, P=number of patches)

Author(s)

Dorian Pustina

Examples

```
lesydata = file.path(find.package('LESYMAP'),'extdata')
filenames = Sys.glob(file.path(lesydata, 'lesions', 'Subject*.nii.gz'))
behavior = Sys.glob(file.path(lesydata, 'behavior', 'behavior.txt'))
template = antsImageRead(
 Sys.glob(file.path(lesydata, 'template', 'ch2.nii.gz')))
lsm = lesymap(filenames, behavior, method = 'BMfast')
plot(template, lsm$stat.img, window.overlay = range(lsm$stat.img))

## Not run:
# Same analysis with SCCAN
lsm = lesymap(filenames, behavior, method = 'sccan',
sparseness=0.045, validateSparseness=FALSE)
plot(template, lsm$stat.img, window.overlay = range(lsm$stat.img))
```

```
save.lesymap(lsm, saveDir='/home/dp/Desktop/SCCANresults')

## End(Not run)
```

---

lsm_BM *lsm_BM*

Description

Lesion to symptom mapping performed on a prepared matrix. Brunner-Munzel tests are performed using each column of the matrix to split the behavioral scores in two groups.

Usage

```
lsm_BM(lesmat, behavior, permuteNthreshold = 9, nperm = 10000,
  alternative = "greater", showInfo = TRUE, ...)
```

Arguments

lesmat	binary matrix (0/1) of voxels (columns) and subjects (rows).
behavior	vector of behavioral scores.
permuteNthreshold	(default=9) Voxels lesioned in less than this number will undergo permutation based thresholding. See Medina et al 2010.
nperm	Number of permutations to perform when needed.
alternative	(default="greater") It is assumed that healthy voxels (0) have greater behavioral scores. If your data follow an inverted relationship choose "less" or "two.sided".
showInfo	display info messagges when running the function.
...	other arguments received from lesymap.

Value

List of objects returned:

- statistic - vector of statistical values
- pvalue - vector of pvalues
- zscore - vector of zscores

Author(s)

Dorian Pustina

Examples

```
{
set.seed(123)
lesmat = matrix(rbinom(200,1,0.5), ncol=2)
set.seed(123)
behavior = rnorm(100)
result = lsm_BM(lesmat, behavior)
}
```

---

lsm_BMfast *lsm_BMfast*

Description

Lesion to symptom mapping performed on a prepared matrix. Brunner-Munzel tests are performed using each column of the matrix to split the behavioral scores in two groups. This function relies on a compiled version for fast processing.

Usage

```
lsm_BMfast(lesmat, behavior, permuteNthreshold = 9,
  alternative = "greater", statOnly = FALSE, nperm = 1000,
  npermBM = 20000, FWERperm = FALSE, v = 1, pThreshold = 0.05,
  permuteAllVoxelsBM = FALSE, showInfo = FALSE, ...)
```

Arguments

lesmat	binary matrix (0/1) of voxels (columns) and subjects (rows).
behavior	vector of behavioral scores.
permuteNthreshold	(default=9) Voxels lesioned in fewer than 9 subjects may yield incorrect p-values with Brunner-Munzel tests, so they need to identify pvalues through individualized permutations. This parameter sets the threshold to find which voxels need permutations. See Medina et al (2010).
alternative	(default="greater") It is assumed that healthy voxels (0) have greater behavioral scores. If your data follow an inverted relationship choose "less" or "two.sided".
statOnly	logical (default=FALSE), skips some computations, mostly for internal use to speed up some things.
nperm	(default=1000) Number of permutations to perform on entire volumes when needed for multiple comparisons corrections (i.e., in FWERperm).
npermBM	(default=20000) Number of permutations to perform at every single voxel below permuteNthrehsold. Note, this argument is different from nperm, which controls volume-based permutations to perform multiple comparison corrections with FWERperm.

FWERperm	logical (default=FALSE) whether to perform permutation based FWER thresholding.
v	(default=1) which voxel to record at each permutation with FWERperm. All software use the peak voxel (v=1), but you can choose a voxel further down the list to relax the threshold (i.e., v=10 for 10 highest voxel) (see Mirman (2017)).
pThreshold	(default=0.05) what threshold to use for FWER
permuteAllVoxelsBM	(default=FALSE) whether to force the permutation-based p-value calulation for all voxels, instead of applying only to voxels below permuteNthrehsold. Setting this option to TRUE will force all voxels undergo permutation-based p-value calculation.
showInfo	display info messagges when running the function.
...	other arguments received from lesymap.

Value

List of objects returned:

- statistic - vector of statistical values
- pvalue - vector of pvalues
- zscore - vector of zscores
- perm.vector - (optional) vector of permuted statistics
- perm.FWERthresh - (optional) permutation threshold established from the distribution of perm.vector

Author(s)

Dorian Pustina

Note on zscores qnorm gives same values as MRIcron and relies on the normal distribution. however, we are computing t-scores, and should have relied on that distribution, which is the t-score itself.

Examples

```
{
set.seed(123)
lesmat = matrix(rbinom(200,1,0.5), ncol=2)
set.seed(123)
behavior = rnorm(100)
result = lsm_BMfast(lesmat, behavior)
}
```

lsm_chisq *lsm_chisq*

Description

Lesion to symptom mapping performed on a prepared matrix. The behavior must be a binary vector. Chi square tests are performed at each voxel. By default the Yates correction is performed, use `correct=FALSE` if you need to disable it. The behavior must be a binary vector. Exact p-values can be obtained with permutation based estimatins.

Usage

```
lsm_chisq(lesmat, behavior, YatesCorrect = TRUE, runPermutations = F,
  nperm = 2000, showInfo = TRUE, ...)
```

Arguments

`lesmat`	binary matrix (0/1) of voxels (columns) and subjects (rows).
`behavior`	vector of behavioral scores (must be binary.
`YatesCorrect`	(default=T) logical whether to use Yates correction.
`runPermutations`	logical (default=FALSE) whether to use permutation based p-value estimation.
`nperm`	(default=2000) The number of permutations to run.
`showInfo`	display info messagges when running the function.
`...`	other arguments received from `lesymap`.

Value

List of objects returned:

- `statistic` - vector of statistical values
- `pvalue` - vector of pvalues
- `zscore` - vector of zscores

Author(s)

Dorian Pustina

Examples

```
{
set.seed(123)
lesmat = matrix(rbinom(200,1,0.5), ncol=2)
set.seed(1234)
behavior = rbinom(100,1,0.5)
result = lsm_chisq(lesmat, behavior)
}
```

lsm_regres *lsm_regres*

Description

Lesion to symptom mapping performed on a prepared matrix. Regressions are performed between behavior and each column in the lesmat matrix.

Usage

```
lsm_regres(lesmat, behavior)
```

Arguments

lesmat	matrix of voxels (columns) and subjects (raws).
behavior	vector of behavioral scores.

Value

List of objects returned:

- statistic - vector of statistical values
- pvalue - vector of pvalues
- zscore - vector of zscores

Author(s)

Dorian Pustina

Examples

```
{
set.seed(123)
lesmat = matrix(rbinom(200,1,0.5), ncol=2)
set.seed(123)
behavior = rnorm(100)
result = lsm_regres(lesmat, behavior)
}
```

lsm_regresfast *lsm_regresfast*

Description

Lesion to symptom mapping performed on a prepared matrix. Regressions are performed between behavior and each column of the lesmat matrix. Fast function based on compiled code.

Usage

```
lsm_regresfast(lesmat, behavior, covariates = NA, FWERperm = F,
  nperm = 1000, v = 1, pThreshold = 0.05, clusterPerm = F,
  mask = NA, voxindx = NA, samplemask = NA,
  clusterPermThreshold = 0.05, showInfo = T, ...)
```

Arguments

lesmat	matrix of voxels (columns) and subjects (rows).
behavior	vector of behavioral scores.
covariates	(default=NA) vector of matrix of covariates.
FWERperm	logical (default=FALSE) whether to run permutation based FWER thresholding.
nperm	Number of permutations to perform when needed.
v	(default=1) what voxel to record for FWER thresholding.
pThreshold	(default=0.05) Voxel-wise threshold.
clusterPerm	logical (default=FALSE), whether to perform permutation based cluster thresholding.
mask	(default=NA) antsImage reference mask used for cluster computations.
voxindx	(default=NA) indices of voxels to put in mask
samplemask	(default=NA) antsImage used to extract voxels back in a matrix.
clusterPermThreshold	(default=0.05) threshold for cluster selection after obtaining cluster size distrubution.
showInfo	display info messagges when running the function.
...	other arguments received from lesymap.

Value

List of objects returned:

- statistic - vector of statistical values
- pvalue - vector of pvalues
- zscore - vector of zscores
- perm.vector - (optional) vector of permuted statistics

- perm.FWERthresh - (optional) permutation threshold established from the distribution of perm.vector
- perm.clusterThreshold - (optional) permutation threshold established from the distribution of perm.vector

Author(s)

Dorian Pustina

Examples

```
{
set.seed(123)
lesmat = matrix(rbinom(200,1,0.5), ncol=2)
set.seed(123)
behavior = rnorm(100)
result = lsm_regresfast(lesmat, behavior)
}
```

---

lsm_regresPerm *lsm_regresPerm*

Description

Lesion to symptom mapping performed on a prepared matrix. Regressions are performed between behavior and each column in the lesmat matrix. This function relies on the lmPerm package to run. The number of permutations required to reach stable p-values is estalbished automatically. For this reason, the user cannot specify a predefined number of permutations.

Usage

```
lsm_regresPerm(lesmat, behavior)
```

Arguments

lesmat	matrix of voxels (columns) and subjects (rows).
behavior	vector of behavioral scores.

Value

List with vectors of statistic, pvalue, and zscore.

Author(s)

Dorian Pustina

lsm_sccan *Sparse canonical correlations for symptom mapping.*

Description

Multivariate SCCAN adapted for lesion to symptom mapping purposes. By default an optimization routine is used to find the best `sparseness` value. If you specify sparseness manually, it will be validated to find the cross-validated correlation that can be obtained with that sparseness. You can skip the entire optimization/validation by choosing `optimizeSparseness=FALSE`. To understand SCCAN arguments, see `sparseDecom2`.

Usage

```
lsm_sccan(lesmat, behavior, mask, showInfo = TRUE,
  optimizeSparseness = TRUE, validateSparseness = FALSE,
  tstamp = "%H:%M:%S", pThreshold = 0.05, mycoption = 1,
  robust = 1, sparseness = 0.045, sparseness.behav = -0.99,
  nvecs = 1, cthresh = 150, its = 20, npermsSCCAN = 0,
  smooth = 0.4, maxBased = FALSE, directionalSCCAN = TRUE, ...)
```

Arguments

`lesmat`	matrix of voxels (columns) and subjects (rows).
`behavior`	vector of behavioral scores.
`mask`	antsImage binary mask to put back voxels in image.
`showInfo`	logical (default=TRUE) display messages
`optimizeSparseness`	logical (default=TRUE) whether to run the sparseness optimization routine. If FALSE, the default sparseness value will be used. If sparseness is manually defined this flag decides if cross validated correlations will be computed for the defined sparseness.
`validateSparseness`	logical (conditional default=TRUE) If sparseness is manually defined, this flag decides if cross validated correlations will be computed for the defined sparseness.
`tstamp`	timestamp format used in LESYMAP
`pThreshold`	(default=0.05) If cross validated correlations show significance below this value the results are considered null and an empty map is returned.
`mycoption`	(default=1) SCCAN parameter, see `sparseDecom2`
`robust`	(ddefault=1) SCCAN parameter, see `sparseDecom2`
`sparseness`	(default=1) SCCAN parameter. Decides the proportion of voxels that will receive a non-zero weight. A positive sparseness will force the solution of each component to be one sided, i.e., voxels cannot have both positive and negative weights. A negative sparseness allows dual sided solution, where some voxels

can have positive weights and other voxels can have negative weights. Setting sparseness manually without running the optimization routing is not recommended. For more, see sparseDecom2.

sparseness.behav
: SCCAN parameter, what sparsness to use for behavioral scores. Useful only if multiple behavioral scores are passed. This argument is not optimized, you should not change it if you are not familiar with SCCAN.

nvecs
: (default=1) SCCAN parameter. Normally only one eigenvector of weights is obtained in LESYMAP. Multiple maps/eigenvectors can be retrieved for mapping full deficit profiles in the future. For more, see sparseDecom2

cthresh
: (default=150) SCCAN parameter, see sparseDecom2

its
: (default=20) SCCAN parameter, see sparseDecom2

npermsSCCAN
: (default=0) SCCAN permutations. In theory can be used to determine if the cross-correlation between the two sides (behavior and lesions) is not random. However, LESYMAP uses k-fold validations, which are faster; this option has not been tested. For more, see sparseDecom2.

smooth
: (default=0.4) SCCAN parameter. Determines the amount of smoothing of weights in image space performed by sparseDecom2. The current default value is somewhat arbitrary, it was not determined through systematic simulations.

maxBased
: (default=FALSE) SCCAN parameter. Removes voxels with weights smaller than 10% of the peak weight during internal SCCAN iterations. Although similar to what is done in LESYMAP with standard SCCAN results, this strategy follows a different route, and produces different weights. The overall final result is, however, quite similar. This method is faster then the standard SCCAN call in LESYMAP, but has not been tested thoroughly. Note that the optimal sparseness obtained with maxBased=TRUE is not optimal when switching to maxBased=FALSE.

directionalSCCAN
: (default=TRUE) If TRUE, the upper and lower bounds of sparseness search will be negative. A negative sparseness permits positive and negative voxel weights, thus finding the direction of the relationship with behavior.

...
: other arguments received from lesymap.

Value

List of objects returned:

- statistic - vector of statistical values
- pvalue - vector of pvalues
- rawWeights.img - image with raw SCCAN voxel weights
- sccan.eig2 - SCCAN weight(s) for behavior column(s).
- sccan.ccasummary - SCCAN summary of projection correlations and permutation-derived pvalues
- optimalSparseness - (if optimizeSparseness=TRUE) optimal value found for sparseness
- CVcorrelation.stat - (if optimizeSparseness=TRUE) Correlation between true and predicted score with k-fold validation using the optimal sparseness value
- CVcorrelation.pval - (if optimizeSparseness=TRUE) p-value of the above correlation

Author(s)

Dorian Pustina

Examples

```
{
 ## Not run:
  lesydata = file.path(find.package('LESYMAP'),'extdata')
  filenames = Sys.glob(file.path(lesydata, 'lesions', '*.nii.gz'))
  behavior = Sys.glob(file.path(lesydata, 'behavior', 'behavior.txt'))
  behavior = read.table(behavior,header=FALSE)[,1]
  avg = antsAverageImages(filenames)
  mask = thresholdImage(avg, 0.1, Inf)
  lesmat = imagesToMatrix(filenames,mask)
  result = lsm_sccan(lesmat, behavior,
       optimizeSparseness=F, sparseness=0.8, mask = mask)

## End(Not run)
}
```

`lsm_svr` *Support Vector Regression for symptom mapping*

Description

Lesion to symptom mapping performed on a prepared matrix. The SVR method is used. The function relies on the `svm` function of the e1071 package. The analysis follows a similar logic found in the SVR-LSM code published by Zhang (2015). After a first run of SVM, p-values are established with a permutation procedures as the number of times weights are randomly exceeded in permutations. The returned p-values are not corrected for multiple comparisons.

Usage

```
lsm_svr(lesmat, behavior, SVR.nperm = 10000,
  SVR.type = "eps-regression", SVR.kernel = "radial", SVR.gamma = 5,
  SVR.cost = 30, SVR.epsilon = 0.1, showInfo = TRUE,
  tstamp = "%H:%M:%S", ...)
```

Arguments

`lesmat`	matrix of voxels (columns) and subjects (rows).
`behavior`	vector of behavioral scores.
`SVR.nperm`	(default=10,000) number of permutations to run to estimate p-values. Note, these p-values are uncorrected for multiple comparisons.
`SVR.type`	(default='eps-regression') type of SVM to run, see `svm`.
`SVR.kernel`	(default='radial') type of kernel to use, see `svm`

SVR.gamma	(default=5) gamma value, see svm.
SVR.cost	(default=30) cost value, see svm.
SVR.epsilon	(default=0.1) epsilon value, see svm.
showInfo	logical (default=TRUE) display messages
tstamp	(default="%H:%M:%S") timestamp format for messages
...	other arguments received from lesymap.

Value

List of objects returned:

- statistic - vector of statistical values
- pvalue - vector of pvalues

Author(s)

Daniel Wiesen, Dorian Pustina WHAT IS THE RATIONALE FOR SCALING BY 10? WHAT IS THE RATIONALE FOR SCALING BY 10?

---

lsm_ttest *lsm_ttest*

Description

Lesion to symptom mapping performed on a prepared matrix. T-tests are performed using each column of the matrix to split the behavioral scores in two groups. If var.equal=TRUE the Welch test is performed instead.

Usage

```
lsm_ttest(lesmat, behavior, var.equal = T, alternative = "greater",
  ...)
```

Arguments

lesmat	binary matrix (0/1) of voxels (columns) and subjects (rows).
behavior	vector of behavioral scores.
var.equal	logical (default=TRUE) should the variance between groups considered equal (t-test) or unequal (Welch test).
alternative	(default='greater') Sets the expected relationship between voxel value and behavior. By default voxels with zero are not lesioned, and behavior is expected to be higher, thus alternative='greater'. If the relationship in your data is inverted, use alternative='less', and if you don't have a relationship hypothesis data, use alternative='two.sided'.
...	other arguments received from lesymap.

Value

List of objects returned:

- `statistic` - vector of statistical values
- `pvalue` - vector of pvalues
- `zscore` - vector of zscores

Author(s)

Dorian Pustina

Examples

```
{
set.seed(123)
lesmat = matrix(rbinom(200,1,0.5), ncol=2)
set.seed(123)
behavior = rnorm(100)
result = lsm_ttest(lesmat, behavior)
}
```

`minSegDistance` *minSegDistance*

Description

This function computes the metric displacement between two binary masks

Usage

```
minSegDistance(manual, predict, get = "all", binarize = F, label = 1)
```

Arguments

`manual`	manual segmentation of class antsImage, used as reference
`predict`	other antsImage to compare to manual
`get`	(default='all') one of 'mean', 'max', 'min', or 'all'
`binarize`	logical (default=FALSE) whether to binarize the input images
`label`	(default=1) integer or vector of labels to binarize I.e., label=c(2,4) means label 2 from manual, and 4 from predict will be compared.

Value

Scalar (for 'mean', 'max', 'min') or list (for 'all'). Note, results are in milimeters

Note

max = Hausdorff distance

Author(s)

Dorian Pustina

`optimize_SCCANsparseness` *Optimization of SCCAN sparseness*

Description

Function used to optimize SCCAN sparseness for lesion to symptom mapping.

Usage

```
optimize_SCCANsparseness(lesmat, behavior, mask, nFolds = 4,
  sparsenessPenalty = 0.03, lowerSparseness = -0.9,
  upperSparseness = 0.9, tol = 0.03, justValidate = FALSE,
  cvRepetitions = ifelse(length(behavior) <= 30, 6,
  ifelse(length(behavior) <= 40, 5, ifelse(length(behavior) <= 50, 4, 3))),
  showInfo = TRUE, directionalSCCAN = TRUE, mycoption = 1,
  robust = 1, sparseness = NA, nvecs = 1, cthresh = 150,
  its = 30, npermsSCCAN = 0, smooth = 0.4,
  sparseness.behav = -0.99, maxBased = FALSE, ...)
```

Arguments

`lesmat`	lesion matrix
`behavior`	behavior vector
`mask`	antsImage mask
`nFolds`	how many folds to use
`sparsenessPenalty`	penalty term
`lowerSparseness`	minimum searched sparseness
`upperSparseness`	maximum searched sparseness
`tol`	tolerance value, see `optimize()` in R
`justValidate`	just check the CV of provided sparseness
`cvRepetitions`	number of cross-validations at each sparseness value. Dynamically set depending on sample size: <=30 to 6 reps, <=40 to 5 reps, <=50 to 4 reps, > 50 to 3 reps.

showInfo	logical (default=TRUE) display messages
directionalSCCAN	(default=TRUE) switching to FALSE will switch sparseness range in the positive side, 0.005 to 0.9
mycoption	standard SCCAN parameter
robust	standard SCCAN parameter
sparseness	standard SCCAN parameter
nvecs	standard SCCAN parameter
cthresh	standard SCCAN parameter
its	standard SCCAN parameter
npermsSCCAN	SCCAN permutations
smooth	standard SCCAN parameter
sparseness.behav	what sparsness to use for behavior
maxBased	standard SCCAN parameter
...	other arguments received from lesymap or lsm_sccan.

Value

List with:
minimum - best sparseness value
objective - minimum value of objective function
CVcorrelation - cross-validated correlation of optimal sparness

Author(s)

Dorian Pustina

the optimization function Will run SCCAN on each training fold, compute behavior prediction on the test fold, and finally return a cross validated correlation from entire sample

end of optimfun

`print.lesymap` *print.lesymap*

Description

Funciton to display some meaningful summary when a lesymap output is called in command line.

Usage

```
## S3 method for class 'lesymap'
print(x, ...)
```

Arguments

x	the output from a lesymap() call.
...	useless for compatibility with default print.

Author(s)

Dorian Pustina

`registerLesionToTemplate`

registerLesionToTemplate

Description

Brings lesion maps in template space by registering the subject\'s anatomical to the template and applying the same transform to the lesion. To improve the registration the anatomical image is bias corrected and denoised. In addition, you can choose to skull-strip the image and run a more careful registration brain-on-brain so that the skull does not impact the registration in any way. Note, for technical reasons the registration is performed counterintuitively by moving the template on the subject, and not the subject on the template. For this reason, to bring the subject in template space we use the inverse transformation. Also note, at the moment ANTsR does not produce an inverse affine transformation explicitly, both forward and inverse affine transforms are identical. You can use ANTs to compute the inverse, or tell ANTsR if you need to invert an affine matrix applying the transformations (see `whichtoinvert` in `antsApplyTransforms`).

Usage

```
registerLesionToTemplate(subImg, subLesion, templateImg = NA,
  templateBrainMask = NA, templateRegMask = NA, skullStrip = T,
  typeofTransform = "SyNCC", outprefix = "", tstamp = "%H:%M:%S",
  showInfo = T, ...)
```

Arguments

subImg	antsImage or character filename of the anatomical image of the subject. Typically this is a T1-weighted MRI image, on which you drew the lesion map.
subLesion	antsImage or character filename of the lesion map. Typically you draw this manually or obtain it from automated lesion segmentation software. You can try our LINDA toolbox for an automated alternative. Yet, manual drawing can be performed quickly and is preferred.
templateImg	antsImage or filename of the anatomical template image. This image should be with skull included.
templateBrainMask	antsImage or filename of the template brain mask. This mask is needed for skull-stripped registrations.

templateRegMask
: antsImage or filename of the template mask that includes the skull but no face. Useful for improving the skull stripping process.

skullStrip
: logical whether to remove the skull and perform brain-on-brain registration.

typeofTransform
: an antsRegistration parameter that controls the quality of registration. The default is SyNCC, which probably is the most robust and takes long (1-2 hours maybe). For faster registration you can try SyN.

outprefix
: character of the prefix where to save the output. If this is set, most of images and transformations will be saved at the specified path/prefix. The folder must exist or you will get an error. It is passed without modification to antsRegistration.

tstamp
: format of the timestamp when displaying info messages.

showInfo
: logical whether to show info messages or be completely quiet. If you want also verbose registration messages, please set verbose=TRUE.

...
: other arguments to pass to antsRegistration

Value

List of objects returned:

- subImg - subject\'s image in native space (after bias correction, denoising, skull stripping, etc.)
- subLesion - subject\'s lesion map in native space
- subImgTemplate - subject\'s image in template space
- subLesionTemplate - subject\'s lesion in template space
- subRegMask - registration mask in native space
- templateImg - the template used to register the subject
- templateBrainMask - the brain mask of the template image
- subLesionTemplate - the template mask with skull and no face
- registration$inverse_subject2template - transformation matrices subject to template
- registration$forward_template2subject - transformation matrices template to subject

Author(s)

Dorian Pustina

Examples

```
## Not run:
anatomical = '/mnt/c/User/dp/Desktop/Subject1_anat.nii.gz'
lesion = '/mnt/c/User/dp/Desktop/Subject1_les.nii.gz'
newles = registerLesionToTemplate(anatomical, lesion,
        outprefix = '/mnt/c/User/dp/Desktop/Subj1onTemplate_')

## End(Not run)
```

`regresfast` *Fast linear regressions*

Description

Takes a matrix of voxels and a vector of behavior and runs fast regressions for each voxel. Covariates can be defined (i.e. age) to find the effect of each voxel on behavior within the context of other predictive factors.

Usage

```
regresfast(X, y, covariates, hascovar = FALSE)
```

Arguments

`X`	matrix of voxlels (columns) for all subjects (rows).
`y`	vector of behavioral scores.
`covariates`	matrix with one or more columns. Must be of same length as behavior. This variable should always be set, and the next argument can tell if covariates should be used or not.
`hascovar`	logical to tell whether covariates should be used.

Value

List with:

- `statistic` - regression t-score
- `n` - number of subjects
- `kxfm` - degrees of freedom.

Author(s)

Dorian Pustina

Examples

```
set.seed(1234)
lesmat = matrix(rbinom(40,1,0.2), ncol=2)
set.seed(1234)
behavior = rnorm(20)
test = LESYMAP::regresfast(lesmat, behavior, as.matrix(behavior), hascovar=FALSE)
test$statistic[,1] # 0.6915683 1.1434760
test$kxmat # 2
```

`save.lesymap` *Save the output of lesymap.*

Description

Function to save the output of lesymap.

Usage

```
save.lesymap(lsm, saveDir, infoFile = "Info.txt", template = NA,
  saveTemplate = F, savePatchImages = T, plot.alpha = 0.8,
  plot.axis = 3, plot.quality = 8, outputLogFile = "outputLog.txt",
  ...)
```

Arguments

`lsm`	object obtained with lesymap()
`saveDir`	folder to save to, will be created if it doesn't exist.
`infoFile`	(default='Info.txt') what should be the filename of the file with information.
`template`	(default=NA) an antsImage to overlay the results to. If the template is provided, results will be plotted and saved as image.
`saveTemplate`	(default=FALSE) should the template image also be saved? Useful when passing the results to a colleague.
`savePatchImages`	(default=TRUE) should the patch images be saved
`plot.alpha`	see plot.antsImage
`plot.axis`	see plot.antsImage
`plot.quality`	see plot.antsImage
`outputLogFile`	(default='outputLog.txt') the filename to save the console output
`...`	other arguments to use for plot().

Value

Nothing is returned. Files saved include resulting maps and a descriptive file with a lot of information about the lesymap run.

Author(s)

Dorian Pustina

`simulateBehavior` *Simulation of behavior scores from lesion maps*

Description

Function simulate behavioral scores based on the lesion load of specific brain areas. Used to run simulation studies.

Usage

```
simulateBehavior(lesions.list, parcellation, label = NA, mask = NA,
  errorWeight = 0.5, binaryCheck = FALSE, exponent = 1)
```

Arguments

`lesions.list`	list of lesions (antsImages) or vector of filenames.
`parcellation`	mask or parcellation image. If a parcellation is passed, lesion load will be computed for each different label (value) in the image. Zero and non-affected labels are not returned by default. The parcellation input can be an antsImage or a character vector pointing to a file.
`label`	(default=NA) if a parcellation scheme id being used, you can select which labels to simulate behaviors for (i.e., c(101,43) to simulate behavior for labels with value 101 and 43 only). If not set a simulation will be returned from each parcel.
`mask`	mask to restrict the count of lesioned voxels. It is not recommended to use a mask, because lesions should affect behavior as they are, without the user restricting the lesions to masks defined in post-processing.
`errorWeight`	(default=0.5) the amount of error to be added, i.e., 0.5 means half of the simulation will be error, the other half signal
`binaryCheck`	(default=FALSE) check to make sure all lesions are binary
`exponent`	power exponent to elevate behavior in order to increase non-linearity relationship with lesion load. 1 is default, and 3 is what Wang (2013) reported as lesion load relationship with behavior.

Value

List of objectas returned:

- `behavload` - a matrix of simulated behavioral scores. Each column shows simulation for a single parcel. Column names indicate the label number in the parcellation file.
- `lesload` - same as behavload, but indicates lesions loads of the simulated regions.
- `lesbehavCorrelation` - vector of Pearson correlations between lesion load and simulated scores.
- `LesvolBehavCorrelation` - vector of Pearson correlations between lesion size and simulated scores.

Author(s)

Dorian Pustina

Examples

```
{
 ## Not run:
  lesydata = file.path(find.package('LESYMAP'),'extdata')
  parcellation = antsImageRead(
  Sys.glob(file.path(lesydata, 'template', 'Parcellation_403areas.nii.gz')))
  filenames = Sys.glob(file.path(lesydata, 'lesions', '*.nii.gz'))
  lesions = imageFileNames2ImageList(filenames)
  simBehavior = simulateBehavior(lesions.list = lesions, parcellation =
  parcellation, label = c(101,43))

## End(Not run)
}
```

Index