Package ‘oasis’

February 21, 2018

Type Package

Title Multiple Sclerosis Lesion Segmentation using Magnetic Resonance Imaging (MRI)

Version 3.0.4

Date 2018-02-20

Description Trains and makes predictions from the OASIS method, described in detail in the paper ``OASIS is Automated Statistical Inference for Segmentation, with applications to multiple sclerosis lesion segmentation in MRI'' <doi:10.1016/j.nicl.2013.03.002>.
OASIS is a method for multiple sclerosis (MS)
lesion segmentation on structural magnetic resonance image (MRI) studies. OASIS creates probability maps of lesion presence using the FLAIR, T2, T1, and PD structural MRI volumes. This packages allows for training of the OASIS model and prediction of OASIS probability maps from a trained model with user supplied studies that have a gold standard lesion segmentation masks. The package will also create OASIS probability maps for MRI studies using the OASIS model from the OASIS paper if no gold standard lesion segmentation masks are available.

Depends R (>= 2.10)

Imports neurobase, fslr (>= 2.13), methods, stats, parallel, oro.nifti, mmand

Suggests httr, covr, ROCR

License GPL-2

LazyLoad yes

LazyData yes

Encoding UTF-8

RoxygenNote 6.0.1

NeedsCompilation no

Author Elizabeth M. Sweeney [aut, cre],
John Muschelli [aut],
R. Taki Shinohara [aut]

Maintainer Elizabeth M. Sweeney <elizabethmargaretsweeney@gmail.com>

Repository CRAN

Date/Publication 2018-02-21 18:22:15 UTC

R topics documented:

`correct_image_dim` *Image Dimension Correction*

Description

This function takes an image and drops dimensions until the volume is a user specified dimension.

Usage

```
correct_image_dim(image, dim = 3)
```

Arguments

image	volume of class `nifti`
dim	scalar value of desired image dimension

Value

Returns a volume of class `nifti` of desired dimension.

Examples

```
## Not run:
library(neurobase)
flair <- readnii('path/to/flair', reorient = FALSE)
flair <- correct_image_dim(flair, dim = 3)

## End(Not run)
```

`example_oasis_df` *Example* `data.frame` *of MS voxels*

Description

A `data.frame` of MS voxels for OASIS prediction

Usage

```
example_oasis_df
```

Format

A `data.frame` with 100 rows and 13 columns, corresponding to the predictors of the OASIS model.

`nopd_oasis_model` *Updated OASIS glm Object*

Description

A `glm` object containing the OASIS model from the updated, corrected data.

Usage

```
nopd_oasis_model
```

Format

A `glm` object

`oasis_erode` *OASIS Erode Mask*

Description

An alternative to using `fslerode` for mask erosion of a brain mask by a box kernel defined by millimeter

Usage

```
oasis_erode(mask, mm = c(5, 5, 5))
```

Arguments

mask	object of class nifti
mm	Number of erosion (in millimeters)

Value

Object of class nifti

Examples

```
library(neurobase)
library(fslr)
library(oasis)
niis = tempfile(fileext = ".nii.gz")
if (require(httr)) {
   url = paste0("https://s3.us-east-2.amazonaws.com/brainder/software/",
   "flair/templates/GG-853-FLAIR-2.0mm.nii.gz")
   req <- httr::GET(url,
   httr::write_disk(path = niis))
   httr::stop_for_status(req)

 flair <- fast_readnii(niis)
 res = oasis_erode(flair > 50, mm = c(2, 2, 2))
}
```

oasis_model *Updated OASIS glm Object*

Description

A glm object containing the OASIS model from the updated, corrected data. The original model from the OASIS paper is located at original_oasis_model.

Usage

```
oasis_model
```

Format

A glm object

oasis_predict *OASIS Prediction*

Description

This function creates the OASIS probability map from a single MRI study with FLAIR, T1, T2, and PD volumes.

Usage

```
oasis_predict(flair, t1, t2, pd = NULL, brain_mask = NULL, model = NULL,
  return_preproc = FALSE, binary = FALSE, threshold = 0.16,
  verbose = TRUE, oasis_dataframe = NULL, voxel_selection = NULL, ...)
```

Arguments

`flair`	flair volume of class `nifti`
`t1`	t1 volume of class `nifti`
`t2`	t2 volume of class `nifti`
`pd`	pd volume of class `nifti`
`brain_mask`	brain mask of class `nifti`, if `NULL` a brain mask will be created using `fslbet`. Note that provided brain masks should be in the same space as the T1 volume if `preproc = TRUE`, as all volumes will be registered to this space
`model`	an object of class `glm` used to make the OASIS predictions
`return_preproc`	is a logical value that indicates whether the preprocessed images should be returned, if `NULL` then the model from the OASIS paper will be used
`binary`	logical indicating whether a binary map should be returned by thresholding the probability map
`threshold`	numeric indicating the threshold value for the probability map, with default of 0.16 for the OASIS paper
`verbose`	print diagnostic messages
`oasis_dataframe`	if `oasis_train_dataframe` was already run, specify the `data.frame` and `voxel_selection` and `brain_mask` to make prediction
`voxel_selection`	image of selected voxels. If `oasis_train_dataframe` was already run, specify the `data.frame` and `voxel_selection` and `brain_mask` to make prediction
`...`	options passed to `oasis_train_dataframe`

Value

A list of volumes: the OASIS probability map, the preprocessed volumes (if `return_preproc = TRUE`), the brain mask for the subject, the voxel selection mask, and a thresholded, binary mask (if `binary = TRUE`) .

Examples

```
library(ROCR)
p = predict( oasis::oasis_model,
    newdata = example_oasis_df,
    type = 'response')
nopd_p = predict( oasis::nopd_oasis_model,
    newdata = example_oasis_df,
    type = 'response')
y =  example_oasis_df$GOLD_Lesions
pred = ROCR::prediction(p, y)
perf = ROCR::performance(pred, ”tpr”, ”fpr”)
plot(perf)


library(neurobase)
dl_file = function(url) {
   tfile = tempfile(fileext = ”.nii.gz”)
   req <- httr::GET(url,
   httr::write_disk(path = tfile))
   httr::stop_for_status(req)
   tfile
}
in_ci <- function() {
 nzchar(Sys.getenv(”CI”))
}
on_cran = function() {
 identical(Sys.getenv(”NOT_CRAN”), ”false”)
}
if (in_ci() || on_cran()) {
  if (fslr::have.fsl() && require(httr)) {
    mods = c(”FLAIR”, ”T1W”, ”T2W”, ”consensus_gt”, ”brainmask”)
    base_url = file.path(
      ”https://raw.githubusercontent.com/muschellij2/open_ms_data”,
      ”master/cross_sectional/coregistered/patient01/”)
    files = paste0(base_url, mods, ”.nii.gz”)
    files = sapply(files, dl_file)
    names(files) = mods

    flair <- readnii(files[”FLAIR”])
    t1 <- readnii(files[”T1W”])
    t2 <- readnii(files[”T2W”])
    brain_mask <- readnii(files[”brainmask”])
    gold_standard = readnii(files[”consensus_gt”])
    oasis_preprocessed_data <- oasis_predict(flair, t1, t2,
      brain_mask = brain_mask, preproc = TRUE)
  }
}
```

`oasis_preproc` *OASIS Image Preprocessing*

Description

This function does the required preprocessing for OASIS for the FLAIR, T2, T1, and PD volumes using FSL through `fslr`. The preprocessing steps are (1) inhomogeneity correct using `fsl_biascorrect` and (2) rigid registration using `flirt` to the T1 space.

Usage

```
oasis_preproc(flair, t1, t2, pd = NULL, brain_mask = NULL, verbose = TRUE,
  cores = 1)
```

Arguments

`flair`	FLAIR volume of class `nifti`
`t1`	T1 volume of class `nifti`
`t2`	T2 volume of class `nifti`
`pd`	PD volume of class `nifti`
`brain_mask`	binary mask volume of class `nifti`
`verbose`	a logical value for printing diagnostic output
`cores`	numeric indicating the number of cores to be used (no more than 4 is useful for this software implementation)

Value

Returns a list of objects of class `nifti`, namely the inhomogeneity corrected FLAIR, T1, T2, and PD registered to the space of the T1 volume.

Examples

```
library(neurobase)
dl_file = function(url) {
   tfile = tempfile(fileext = ".nii.gz")
   req <- httr::GET(url,
   httr::write_disk(path = tfile))
   httr::stop_for_status(req)
   tfile
}
in_ci <- function() {
 nzchar(Sys.getenv("CI"))
}
on_cran = function() {
 identical(Sys.getenv("NOT_CRAN"), "false")
}
if (in_ci() || on_cran()) {
  if (fslr::have.fsl() && require(httr)) {
    mods = c("FLAIR", "T1W", "T2W", "consensus_gt", "brainmask")
    base_url = file.path(
      "https://raw.githubusercontent.com/muschellij2/open_ms_data",
      "master/cross_sectional/coregistered/patient01/")
    files = paste0(base_url, mods, ".nii.gz")
    files = sapply(files, dl_file)
    names(files) = mods

    flair <- readnii(files["FLAIR"])
    t1 <- readnii(files["T1W"])
    t2 <- readnii(files["T2W"])
    brain_mask <- readnii(files["brainmask"])
    gold_standard = readnii(files["consensus_gt"])
    oasis_preprocessed_data <- oasis_preproc(flair, t1, t2,
      brain_mask = brain_mask)
  }
}
```

oasis_training *OASIS Training*

Description

This function trains the OASIS model from a `data.frame` produced by an element from the output of the function `oasis_train_dataframe`

Usage

```
oasis_training(..., formula = GoldStandard ~ FLAIR_10 * FLAIR + FLAIR_20 *
  FLAIR + PD_10 * PD + PD_20 * PD + T2_10 * T2 + T2_20 * T2 + T1_10 * T1 + T1_20
  * T1, remove_preproc = FALSE)
```

Arguments

`...` `data.frame`(s) produced by the `oasis_train_dataframe` function

`formula` formula to be fit by glm model

`remove_preproc` a logical stating if `oasis_dataframe` needs to be extracted from the list of objects. Will call `list$oasis_dataframe`

Value

Returns a `glm` object containing the trained OASIS coefficients to be used by the function `oasis_predict`.

Examples

```
df = oasis::example_oasis_df
df$GoldStandard = df$GOLD_Lesions
oasis_training(df)
```

oasis_train_dataframe *OASIS Training Data Frame*

Description

This function creates the training vectors from a single MRI study that has FLAIR, T1, T2, and PD volumes as well as binary masks of lesions. The function can create a brain mask for the data (or the user can supply a brain mask), can preprocess the data, and the user may supply already normalized data if they wish to use an alternative normalization method.

Usage

```
oasis_train_dataframe(flair, t1, t2, pd = NULL, gold_standard = NULL,
  brain_mask = NULL, preproc = FALSE, normalize = TRUE, slices = NULL,
  orientation = c("axial", "coronal", "sagittal"), return_preproc = FALSE,
  cores = 1, sigma = c(10, 20), verbose = TRUE, eroder = c("fsl",
  "oasis"))
```

Arguments

`flair` FLAIR volume of class `nifti`

`t1` T1 volume of class `nifti`

`t2` T2 volume of class `nifti`

`pd` PD volume of class `nifti`

`gold_standard` gold standard lesion segmentation mask of class `nifti`

`brain_mask` brain mask of class `nifti`, if NULL a brain mask will be created using `fslbet`

`preproc` is a logical value that determines whether to call the `oasis_preproc` function and performs the necessary preprocessing steps for OASIS

`normalize` is a logical value that determines whether to perform z-score normalization of the image over the brain mask, should be `TRUE` unless you train model using an alternative normalization

`slices` vector of desired slices to train on, if `NULL` then train over the entire brain mask

`orientation` string value telling which orientation the training slices are specified in, can take the values of "axial", "sagittal", or "coronal"

`return_preproc` is a logical value that indicates whether the preprocessed images should be returned

`cores` numeric indicating the number of cores to be used (no more than 4 is useful for this software implementation)

`sigma` Sigmas used to smooth the data, default is 10,20

`verbose` print diagnostic output

`eroder` Should `fslerode` or `oasis_erode` be used

Value

If `return_preproc = FALSE` the function returns a `data.frame` for use with the `oasis_training` function. Otherwise, the function returns a list containing: a `data.frame` for use with the `oasis_training` function, the FLAIR volume, the T1 volume, the T2 volume, the PD volume, the brain mask for the subject, and the voxel selection mask.

See Also

`oasis_training`

Examples

```
library(neurobase)
dl_file = function(url) {
   tfile = tempfile(fileext = ".nii.gz")
   req <- httr::GET(url,
   httr::write_disk(path = tfile))
   httr::stop_for_status(req)
   tfile
}
in_ci <- function() {
 nzchar(Sys.getenv("CI"))
}
on_cran = function() {
 identical(Sys.getenv("NOT_CRAN"), "false")
}
if (in_ci() || on_cran()) {
  if (fslr::have.fsl() && require(httr)) {
    mods = c("FLAIR", "T1W", "T2W", "consensus_gt", "brainmask")
    base_url = file.path(
      "https://raw.githubusercontent.com/muschellij2/open_ms_data",
      "master/cross_sectional/coregistered/patient01/")
    files = paste0(base_url, mods, ".nii.gz")
    files = sapply(files, dl_file)
    names(files) = mods

    flair <- readnii(files["FLAIR"])
    t1 <- readnii(files["T1W"])
    t2 <- readnii(files["T2W"])
    brain_mask <- readnii(files["brainmask"])
    gold_standard = readnii(files["consensus_gt"])
    oasis_preprocessed_data <- oasis_train_dataframe(flair, t1, t2,
      brain_mask = brain_mask, gold_standard = gold_standard)
  }
}
```

`original_oasis_model` *OASIS glm Object*

Description

A `glm` object containing the OASIS model from the OASIS paper.

Usage

```
original_oasis_model
```

Format

A `glm` object

`voxel_selection` *Voxel Selection Procedure*

Description

This function creates a binary mask for the voxel selection procedure for OASIS.

Usage

```
voxel_selection(flair, brain_mask, cutoff)

voxel_selection_with_erosion(flair, brain_mask, verbose = TRUE,
  eroder = c("fsl", "oasis"))
```

Arguments

`flair`	FLAIR volume of class `nifti`
`brain_mask`	brain mask of class `nifti`
`cutoff`	the percentile cutoff for the thresholding, passed to `quantile`
`verbose`	print diagnostic output
`eroder`	Should `fslerode` or `oasis_erode` be used

Value

Returns the voxel selection mask as an object of class `nifti`.

Examples

```
library(neurobase)
library(fslr)
library(oasis)
niis = tempfile(fileext = ".nii.gz")
if (require(httr)) {
   url = paste0("https://s3.us-east-2.amazonaws.com/brainder/software/",
   "flair/templates/GG-853-FLAIR-2.0mm.nii.gz")
   req <- httr::GET(url,
   httr::write_disk(path = niis))
   httr::stop_for_status(req)

 flair <- readnii(niis)
 if (have.fsl()) {
   brain_mask = fslbet(niis) > 0
 } else {
   ind = list(c(10L, 81L), c(12L, 101L), c(3L, 78L))
   all.ind = lapply(ind, function(x) seq(x[1], x[2]))
   brain_mask = niftiarr(flair, 0)
   eg = expand.grid(all.ind)
   eg = as.matrix(eg)
   brain_mask[eg] = 1
 }
 voxel_selection_mask <- voxel_selection(flair,
   brain_mask, cutoff = .85)
}
```

Index