Package 'neuroSCC'

March 11, 2025

Type Package

Title Estimation of SCC's for PET Neuroimaging Data

Version 0.11.0

Author Juan A. Arias [aut, cre]

Maintainer Juan A. Arias <iguanamarina@protonmail.com>

Description This package provides functions for the estimation of Simultaneous Confidence Corridors (SCCs) on PET neuroimaging data. It is designed to help with the replication of methods implemented in the Ph.D. Thesis ``Development of statistical methods for neuroimage data analysis towards early diagnostic of neurodegenerative diseases" and it is best applied using the scripts available in the link below. Further information on the methodology for SCCs estimation can be found in: ``Wang Y, Wang G, Wang L, Ogden RT. Simultaneous confidence corridors for mean functions in functional data analysis of imaging data. Biometrics. 2020 Jun;76(2):427-437".

License use_mit_license()

URL https://github.com/iguanamarina/neuroSCC

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Suggests roxygen2 (>= 7.1.0),
knitr,
rmarkdown

Roxygen list(markdown = TRUE)

Depends R (>= 2.10)

Contents

`databaseCreator` *Create a database of processed PET image data with the appropriate format*

Description

This function automates the processing of PET images based on a specified file name pattern within a working directory. It reads each file matching the pattern, processes it using `neuroSCC::neuroCleaner`, and compiles the results into a comprehensive data frame. The function can handle both control and pathological data based on the `control` parameter.

Usage

```
databaseCreator(pattern, control = TRUE)
```

Arguments

`pattern`	`character`, a regular expression pattern that specifies which files to process.
`control`	`logical`, if `TRUE`, the function processes control group images and if `FALSE`, it processes pathological group images. Default is `TRUE`.

Details

The function first checks if there are files matching the pattern in the working directory. If no files are found, it throws an error. Otherwise, it processes each file individually, extracts necessary data, and appends it to a growing database. The function leverages `neuroSCC::neuroCleaner` for data processing. Each file's subject number is extracted from its name using regular expressions. Depending on the `control` parameter, the database will have either a `CN_number` column for control data or an `AD_number` column for pathological data.

Value

A `data.frame` that aggregates processed data from each image. Each row represents data from one image, including subject numbers and image data.

See Also

neuroCleaner for the underlying image processing.

Examples

```
# Set the working directory where your PET images are stored
setwd("~/GitHub/PhD-2023-Neuroimage-article-SCC-vs-SPM/PETimg_masked for simulations")

# Define the pattern for file names to process
pattern <- "^masked_swwwC\\d+_tripleNormEsp_w00_rrec_OSEM3D_32_it1.nii"

# Create the database for control group images
database_CN <- databaseCreator(pattern, control = TRUE)

# Create the database for pathological group images
database_AD <- databaseCreator(pattern, control = FALSE)
```

`getDimensions` *Get Dimensions from a DICOM File*

Description

This function loads a DICOM image file using the `oro.nifti` package and extracts its dimensions. It provides the X and Y dimensions as well as the total number of elements ('dim') in the image data.

Usage

```
getDimensions(filename = NULL)
```

Arguments

`filename`	`character`, optional; the name of the DICOM file to read. If not provided, the function will search for the first `.img` file in the current working directory.

Details

If no filename is provided, the function searches the current directory for the first file with a `.img` extension. It stops with an error message if no such files are found. It is important to ensure that the specified file or files in the directory are in the DICOM format.

Value

A list containing `xDim`, `yDim`, and `dim`, representing the dimensions of the image in the X and Y axes, and the total number of elements in the image, respectively.

See Also

readNIfTI for the function used to read the DICOM files.

Examples

```
# If 'filename' is not provided, it will get the first image in the current working directory:
dimensions <- getDimensions()

# Providing a specific filename:
dimensions <- getDimensions("003_S_1059.img")
```

getPoints — *Extract standalone significant SCC points from a SCC comparison object*

Description

This function processes an SCC comparison object to identify and extract coordinates of points where significant differences are detected by the SCC analysis. It returns the coordinates of positive and negative differences separately, which can be used for further analysis or visualization on neuroimages.

Usage

```
getPoints(aa)
```

Arguments

aa	A list containing SCC comparison results, expected to have elements like `Z.band`, `ind.inside.cover`, and `scc` matrices.

Value

A list with two elements: coordinates of points with positive differences and coordinates of points with negative differences. Each element is a matrix where rows are points and columns represent coordinates.

Examples

```
points <- getPoints(SCC_COMP_1)
```

matrixCreator — *Convert PET image data to a functional data matrix format*

Description

This function transforms PET image data, previously processed into a database format by `databaseCreator`, into a matrix format suitable for functional data analysis. Each row of the matrix represents a function, corresponding to data from one patient or control subject. The function recognizes whether the matrix is for Control or Alzheimer's group.

Usage

```
matrixCreator(database, pattern, param.z, xy)
```

Arguments

database	A data frame containing the PET image data with columns for 'CN_number', 'z', 'x', 'y', and 'pet' values, created by databaseCreator.
pattern	The regular expression pattern to match filenames in the database. This pattern should correspond to the naming conventions of the PET image files you want to process.
param.z	The specific z-coordinate slice to analyze. This parameter should be defined prior to running this function if not included in the script that calls it.
xy	The total number of data points per z-slice, calculated as the product of the dimensions x and y of the image slice. If not predefined, it must be calculated by the user based on the image dimensions.

Details

matrixCreator follows databaseCreator in the data processing workflow. It reads the data for each file specified by the matched pattern and extracts only the data for the specified z-slice. The function ensures that each row in the resulting matrix corresponds to one patient or control subject, transforming each subset of data into a format suitable for functional data analysis.

Value

A matrix where each row represents the PET data from one control subject, formatted as a continuous line of data points to simulate a function.

See Also

databaseCreator and neuroCleaner for initial data processing and image cleaning.

Examples

```
# Assuming 'database_CN', 'pattern', 'param.z', and 'xy' are already defined
SCC_CN <- matrixCreator(database_CN, pattern, param.z, xy)
```

---

meanNormalization *Mean Average Normalization for SCC Data*

Description

This function normalizes each row of the given SCC data by its mean value. It divides each element in a row by the mean of that row, ignoring NA values.

Usage

```
meanNormalization(SCC_data)
```

Arguments

SCC_data	matrix, a matrix where each row represents SCC data to be normalized.

Details

The function iterates over each row of the SCC data matrix, calculates the mean of the row, and then divides each element of the row by the calculated mean.

Value

A `matrix` where each row has been normalized by its mean value.

Examples

```
# Assume SCC_data is a matrix of SCC values
normalized_SCC_data <- neuroSCC::meanNormalization(SCC_data)
```

`neuroCleaner` *Clean and load data from NIFTI neuroimaging files*

Description

Loads a NIFTI-format neuroimaging file, converts it to a structured data frame, and organizes the data for further analysis. This function serves as the first step in the neuroimaging data processing pipeline, transforming raw PET data into a format suitable for functional data analysis. The neuroSCC package prepares and organizes neuroimaging data, while the actual Simultaneous Confidence Corridors (SCCs) are computed using functions from the ImageSCC package such as `ImageSCC::scc1g.image` or `ImageSCC::scc2g.image`.

Usage

```
neuroCleaner(name, demo = NULL, demoRow = 1)
```

Arguments

`name`	A character string specifying the path to the NIFTI file.
`demo`	An optional data frame containing demographic information for the participants. If provided, it should contain columns that match (case-insensitive): 'PPT', 'Group', 'Sex', and 'Age'. The 'PPT' column can contain participant IDs that match the `name` parameter for automatic matching, but if no match is found, the row specified by `demoRow` will be used. Default is `NULL`.
`demoRow`	An integer specifying which row of the demographic data to use when automatic matching fails or multiple matches are possible. Default is `1`.

Details

The function performs several key operations:

1. Reads the NIFTI file using `oro.nifti` package functions
2. Converts the 3D image data to a structured data frame
3. Extracts the dimensional information (`z`, `x`, `y` coordinates)
4. Organizes the PET intensity values
5. If demographic data is provided, merges it with the image data

The resulting data frame serves as input for subsequent analysis functions in the neuroSCC pipeline, such as `databaseCreator` and `matrixCreator`, which prepare the data for eventual analysis with the ImageSCC package functions.

Value

A data frame with the following columns:

- If demographic data provided: `PPT`, `Group`, `Sex`, `Age`, `z`, `x`, `y`, `pet`
- If no demographic data: `z`, `x`, `y`, `pet`

Each row represents a voxel (3D pixel) from the image, with `pet` containing the intensity value at that location.

See Also

`databaseCreator` for creating databases from multiple NIFTI files.

`oro.nifti::readNIfTI` for the underlying function used to read NIFTI files.

Examples

```
# Example 1: Using package sample data
## Not run:
# Access sample files included with the package
data_dir <- system.file("extdata", package = "neuroSCC")
nifti_file <- file.path(data_dir, "sampleFile1.nii")

# Basic usage with just a NIFTI file
pet_data <- neuroCleaner(nifti_file)

# Display the first few rows of the result
head(pet_data)
#>   z x  y      pet
#>   1 1  1  0.00000
#>   1 1  2  0.00000
#>   1 1  3  0.78526
#>   ...

# Example 2: With demographic data
# Load demographic data
demo_file <- file.path(data_dir, "Demographics.csv")
demo_data <- read.csv(demo_file)

# Display the demographic data
print(demo_data)
#>         PPT Group Sex Age
#> 1 041_S_1391    AD   M  85
#> 2 036_S_1001    AD   M  69
#> 3 037_S_0627    AD   F  59

# Process the first NIFTI file with demographic data
# This will attempt to match based on filename
pet_with_demo <- neuroCleaner(nifti_file, demo = demo_data)

# Display the first few rows with demographic information
head(pet_with_demo)
```

```
#>         PPT Group Sex Age z x  y      pet
#>  041_S_1391    AD   M  85 1 1  1  0.00000
#>  041_S_1391    AD   M  85 1 1  2  0.00000
#>  041_S_1391    AD   M  85 1 1  3  0.78526
#>  ...

# Example 3: Using a specific row from demographic data
# Process another NIFTI file and specify which demographic row to use
nifti_file2 <- file.path(data_dir, "sampleFile2.nii")
pet_with_specific_demo <- neuroCleaner(nifti_file2, demo = demo_data, demoRow = 3)

# Display the first few rows (now with demographic info from row 3)
head(pet_with_specific_demo)
#>         PPT Group Sex Age z x  y      pet
#>  037_S_0627    AD   F  59 1 1  1  0.00000
#>  037_S_0627    AD   F  59 1 1  2  0.00000
#>  037_S_0627    AD   F  59 1 1  3  0.78526
#>  ...

## End(Not run)

# Example 4: Creating synthetic data (fully reproducible)
# Create a simple 3D array as a synthetic NIFTI image
if (requireNamespace("oro.nifti", quietly = TRUE)) {
  # Temporary file path
  temp_nii <- tempfile(fileext = ".nii")

  # Create a small synthetic NIFTI file (3x3x3)
  img_data <- array(1:27, dim = c(3, 3, 3))
  nii_obj <- oro.nifti::nifti(img_data)
  oro.nifti::writeNIfTI(nii_obj, filename = temp_nii, verbose = FALSE)

  # Sample demographic data
  demo_data <- data.frame(
    PPT = c("sample001", "sample002", "sample003"),
    Group = c("Control", "Patient", "Patient"),
    Sex = c("M", "F", "M"),
    Age = c(65, 70, 75)
  )

  # Process the synthetic NIFTI with demographic data (use row 2)
  # This will not run if oro.nifti is not installed
  pet_data <- neuroCleaner(temp_nii, demo = demo_data, demoRow = 2)
  head(pet_data)
}
```

`neuroContour` *Obtain and save neuroimaging contours*

Description

This function extracts contours from neuroimaging data where values change according to specified levels. It uses the `contoureR::getContourLines` function to obtain the contours and stores the coordinates in a list.

Usage

```
neuroContour(data, levels = c(0))
```

Arguments

data	data.frame, a data frame containing the neuroimaging data to process.
levels	numeric, a vector of levels at which to draw the contours. Default is c(0).

Details

The function filters the contours by their GID and stores the coordinates of each contour in a list. It ensures that the contoureR package is loaded before attempting to use its functions.

Value

A list of data frames, where each data frame contains the x and y coordinates of a contour.

See Also

getContourLines for the underlying contour extraction.

Examples

```
# Example usage:
# Load sample data
data <- some_neuroimaging_data

# Get contours at level 0
contours <- neuroContour(data, levels = c(0))

# Plot the first contour
plot(contours[[1]])
if (length(contours) > 1) {
  for (j in 2:length(contours)) {
    points(contours[[j]]) # Holes or internal contours
  }
}
```

---

processROIs *Process ROIs and Save Data Tables*

Description

This function processes regions of interest (ROIs) from PET image data and saves the resulting data tables. It checks if the ROI tables already exist and, if not, processes the data using neuroSCC::neuroCleaner to clean and structure the data.

Usage

```
processROIs(base_dir, regions, numbers)
```

Arguments

base_dir	The base directory where the ROI files and tables are located.
regions	A vector of region names to be processed.
numbers	A vector of numbers corresponding to the patient or control subjects.

Details

processROIs iterates over each combination of region and number, checking if the corresponding ROI table file already exists. If the file does not exist, the function processes the data using neuroSCC::neuroCleaner, adds necessary metadata, and saves the cleaned data as an RDS file. The function provides informative messages about the progress of processing and file saving.

Value

This function does not return a value. It performs data processing and saves the resulting tables to the specified directory.

Examples

```
# Assuming 'base_dir', 'regions', and 'numbers' are already defined
processROIs(base_dir, regions, numbers)
```

Index