Package 'neuroSCC'

March 18, 2025

Type Package

Title Bridging Simultaneous Confidence Corridors with Applications and Performance Evaluation in PET Neuroimaging

Version 0.13.0

Maintainer Juan A. Arias Lopez <juanantonio.arias.lopez@usc.es>

Description Provides functions for the structured processing of PET neuroimaging data in preparation for the estimation of Simultaneous Confidence Corridors (SCCs) for one-group, two-group, or single patient vs group comparisons. The package facilitates PET image loading, data restructuring, integration into a Functional Data Analysis framework, contour extraction, post-SCC identification of significant results, and performance evaluation. It bridges established neuroimaging packages (e.g., 'oro.nifti') with novel statistical methodologies (i.e., 'ImageSCC') and facilitates reproducible analysis pipelines, enabling performance comparison of SCC-based analyses with gold-standard methods such as Statistical Parametric Mapping (SPM).

License MIT + file LICENSE

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

https://github.com/iguanamarina/PhD-2023-SCC-vs-SPM-Group-vs-Group,

https://github.com/iguanamarina/PhD-2024-SCC-vs-SPM-SinglePatient-vs-Group

BugReports https://github.com/iguanamarina/neuroSCC/issues

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Depends R (>= 4.2.0)

Imports oro.nifti,
memisc,
contoureR

Suggests roxygen2 (>= 7.1.0),
knitr,
rmarkdown

VignetteBuilder knitr

Contents

---

calculateMetrics   *Evaluate SCC Detection Performance*

Description

Computes Sensitivity, Specificity, Positive Predictive Value (PPV), and Negative Predictive Value (NPV) for SCC-detected points compared to ground truth ROI points. This function is essential for evaluating the accuracy of SCC methods in detecting true regions of difference in neuroimaging analysis.

Usage

```
calculateMetrics(detectedPoints, truePoints, totalCoords, regionName)
```

Arguments

detectedPoints   A data frame of SCC-detected significant points with columns x and y.

truePoints   A data frame of true region-of-interest (ROI) points with columns x and y.

totalCoords   A data frame containing the full set of image coordinates (x, y).

regionName   A character string specifying the name of the region being evaluated.

Details

- **True Positives (TP):** SCC-detected points that match true ROI points. - **False Positives (FP):** SCC-detected points that are not in the true ROI. - **False Negatives (FN):** True ROI points that SCC failed to detect. - **True Negatives (TN):** Points in totalCoords that were neither detected nor part of the ROI. - Handles **division by zero** cases to prevent errors in metric calculations.

Value

A data frame with the following evaluation metrics:

- region: The analyzed region.
- sensitivity: True positive rate (TP / (TP + FN) * 100).
- specificity: True negative rate (TN / (TN + FP) * 100).
- PPV: Positive Predictive Value (TP / (TP + FP) * 100).
- NPV: Negative Predictive Value (TN / (TN + FN) * 100).

Examples

```
## Not run:
detected <- data.frame(x = c(1, 2, 3), y = c(2, 3, 4))  # SCC-detected points
trueROI <- data.frame(x = c(2, 3), y = c(3, 4))  # True positive regions
totalGrid <- expand.grid(x = 1:5, y = 1:5)  # Full coordinate grid

results <- calculateMetrics(detected, trueROI, totalGrid, "ExampleRegion")
print(results)

## End(Not run)
```

---

databaseCreator   *Create a database of processed PET image data*

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 [neuroCleaner](#), and compiles the results into a comprehensive data frame. The function serves as a key step in the neuroSCC workflow, bridging between individual image processing and preparation for functional data analysis with Simultaneous Confidence Corridors.

Usage

```
databaseCreator(
  pattern,
  control = TRUE,
  extractPattern = NULL,
  useSequentialNumbering = FALSE,
  demo = NULL,
  quiet = FALSE
)
```

Arguments

pattern   character, a regular expression pattern that specifies which files to process.
                 By default, the function is designed to work with filenames structured like:
                 "masked_swwwC1_tripleNormEsp_roiAD_0_1_rrec_OSEM3D_32_it1.nii", where
                 "C1" indicates the subject identifier.

control   logical, if TRUE, the function processes control group images and if FALSE, it
                   processes pathological group images. Default is TRUE.

extractPattern   character, optional custom regular expression to extract the subject number
                   from filenames. Should contain a capture group (\d+) to extract the numerical
                   ID. Default is NULL, which uses the built-in pattern "masked_swwwC(\d+)_.*".

useSequentialNumbering
                   logical, if TRUE, assigns sequential numbers (1,2,3,...) to files instead of at-
                   tempting to extract numbers from filenames. Default is FALSE.

demo   data.frame, optional demographic data. If provided, this demographic infor-
                   mation will be included in the output database for each file. Default is NULL.

quiet   logical, if TRUE, suppresses progress messages. Default is FALSE.

Details

The function performs several key operations:

1. Identifies files in the working directory that match the specified pattern 2. For each matching
file: a. Extracts the subject number from the filename or assigns a sequential number b. Processes
the file using [neuroCleaner](), including demographic data if provided c. Adds subject identifier
information to each row d. Appends the data to a growing database

The subject number extraction uses either:

- The custom pattern provided in extractPattern

- Sequential numbering (1,2,3,...) if useSequentialNumbering=TRUE

- The default pattern "masked_swwwC(\d+)_.*" which extracts numbers after "C" in filenames

This function is typically followed by [matrixCreator]() in the analysis pipeline, which transforms
the database into a format suitable for SCC computation.

Value

A data.frame that aggregates processed data from each image. Each row represents data from one
voxel (3D pixel), including:

- For control group (control=TRUE): CN_number, z, x, y, pet

- For pathological group (control=FALSE): AD_number, z, x, y, pet

- If demographic data is provided: Additional columns like PPT, Group, Sex, Age will be in-
  cluded

The CN_number or AD_number column contains the subject identifier extracted from the filename or
assigned sequentially. The pet column contains the intensity values.

See Also

[neuroCleaner]() for the underlying image processing function.

[matrixCreator]() for the next step in the workflow that converts the database to a matrix format for
SCC analysis.

Examples

```
# Example 1: Basic usage with default settings for control group
## Not run:
# Set the working directory where your PET images are stored
setwd("~/PET_Images")

# 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)

# Example output (first few rows):
# CN_number    z    x    y    pet
# 1           35    1    1    0.0
# 1           35    1    2    0.0
# 1           35    1    3    2.3
# ...

## End(Not run)

# Example 2: Using demographic data
## Not run:
# Load demographic data
demo_data <- read.csv("demographics.csv")

# Create database with demographic information
database_CN <- databaseCreator(pattern, control = TRUE, demo = demo_data)

# Example output with demographic data:
# PPT    Group   Sex  Age  CN_number    z    x    y    pet
# P001   Control  M   65   1           35    1    1    0.0
# P001   Control  M   65   1           35    1    2    0.0
# P001   Control  M   65   1           35    1    3    2.3
# ...

## End(Not run)

# Example 3: Processing pathological group images
## Not run:
# Define the pattern for pathological images
pattern <- "^masked_swwwC\\d+_tripleNormEsp_roiAD_0_8_rrec_OSEM3D_32_it1.nii"

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

# Example output (first few rows):
# AD_number    z    x    y    pet
# 1           35    1    1    0.0
# 1           35    1    2    0.0
# 1           35    1    3    1.8
# ...

## End(Not run)

# Example 4: Using a custom extraction pattern
```

```
## Not run:
# For files named like "subject_023_scan.nii"
pattern <- "^subject_\\d+_scan.nii"
extractPattern <- "subject_(\\d+)_scan"

database <- databaseCreator(pattern, extractPattern = extractPattern)

## End(Not run)

# Example 5: Using sequential numbering instead of extracting from filenames
## Not run:
# For files that don't contain subject numbers in their names
pattern <- "^pet_scan_.*\\.nii"

database <- databaseCreator(pattern, useSequentialNumbering = TRUE)

## End(Not run)

# Example 6: Reproducible example with synthetic data
if (requireNamespace("oro.nifti", quietly = TRUE)) {
  # Create temporary directory and files for demonstration
  temp_dir <- tempdir()
  old_dir <- getwd()
  setwd(temp_dir)

  # Create two simple synthetic NIFTI files
  for (i in 1:2) {
    # Create a small synthetic array (3x3x3)
    img_data <- array(1:27, dim = c(3, 3, 3))
    nii_obj <- oro.nifti::nifti(img_data)

    # Save with names that match our expected pattern
    filename <- paste0("masked_swwwC", i, "_tripleNormEsp_w00_rrec_OSEM3D_32_it1.nii")
 oro.nifti::writeNIfTI(nii_obj, filename = file.path(temp_dir, filename), verbose = FALSE)
  }

  # Process these files
  pattern <- "^masked_swwwC\\d+_tripleNormEsp_w00_rrec_OSEM3D_32_it1.nii"
  # This will not run if oro.nifti is not installed
  # database <- databaseCreator(pattern)

  # Clean up and restore working directory
  setwd(old_dir)
}
```

---

generatePoissonClones   *Generate Synthetic Poisson Clones for PET Data*

Description

This function generates synthetic clones of a PET data matrix by adding Poisson-distributed noise to each non-zero voxel. This is necessary to circumvent the limitations of functional data analysis (FDA) in **single patient vs. group (1 vs. Group) setups**, where a single subject does not provide enough variability to estimate Simultaneous Confidence Corridors (SCCs) reliably.

Usage

```
generatePoissonClones(originalMatrix, numClones, lambdaFactor)
```

Arguments

`originalMatrix`	A numeric matrix where each row represents a flattened PET image.
`numClones`	An integer specifying the number of synthetic clones to generate.
`lambdaFactor`	A positive numeric value controlling the magnitude of Poisson noise.

Details

- Values of '0' remain unchanged to preserve background regions. - 'NA' values are replaced with '0' before adding noise. - Poisson noise is applied only to positive values, scaled by 'lambdaFactor'. - This approach allows **SCC methods to operate in 1 vs. Group analyses**, ensuring statistical validity when a single observation is not sufficient.

Value

A numeric matrix with 'numClones' rows, each containing a modified version of 'originalMatrix' with added Poisson noise.

Examples

```
## Not run:
  # Simulated PET matrix (3 subjects, 4 pixels each)
  petMatrix <- matrix(c(5, 10, 15, 0,
                        3, 8, 12, 0,
                        7, 14, 21, 0), nrow = 3, byrow = TRUE)

  # Generate 5 synthetic clones
  clones <- generatePoissonClones(petMatrix, numClones = 5, lambdaFactor = 0.01)

  # Check the cloned dataset
  print(clones)

## End(Not run)
```

---

getDimensions	*Get Dimensions from a Neuroimaging File*

Description

Extracts dimensional information from NIFTI or similar neuroimaging files. This function is designed to work together with `neuroCleaner()` but can also be used independently for informative purposes.

Usage

```
getDimensions(file)
```

Arguments

file   A NIFTI file object or filename to extract dimensions from.

Details

The function can handle both NIFTI file paths and pre-loaded oro.nifti file objects. It provides a consistent way to extract dimensional information across the package.

Value

A list containing:

- `xDim`: Number of voxels in the X dimension
- `yDim`: Number of voxels in the Y dimension
- `zDim`: Number of slices in the Z dimension
- `dim`: Total number of voxels (xDim * yDim)

Examples

```
## Not run:
# Using a file path
dims <- neuroSCC::getDimensions("path/to/your/image.nii")

# If used within neuroCleaner
file <- oro.nifti::readNIfTI("path/to/your/image.nii")
dims <- neuroSCC::getDimensions(file)

## End(Not run)
```

---

getPoints   *Extract significant SCC points from an SCC comparison object*

Description

This function identifies and extracts coordinates where significant differences fall outside the simultaneous confidence corridors (SCCs). It processes the results from ImageSCC::scc.image, returning the extracted coordinates where the differences are statistically significant.

Usage

```
getPoints(sccResult)
```

Arguments

sccResult   A list containing SCC computation results from ImageSCC::scc.image. The list should include at least:

- `Z.band`: Matrix specifying grid positions.
- `ind.inside.cover`: Indices of grid points inside the confidence band.
- `scc`: 3D array containing computed SCC values.

Details

- Positive points indicate where the first group (e.g., Control) is significantly stronger. - Negative points indicate where the second group (e.g., Pathological) is significantly stronger. - This function must be used **after** running ImageSCC::scc.image.

Value

A named list:

- positivePoints: Data frame with coordinates where the first group shows significantly higher values than the second.
- negativePoints: Data frame with coordinates where the second group shows significantly higher values than the first.

See Also

ImageSCC::scc.image for SCC computation.

Examples

```
## Not run:
# Load SCC results (previously computed)
load("SCC_COMP.RData")

# Extract significant SCC points
significantPoints <- getPoints(SCC_COMP)

# Display extracted coordinates
head(significantPoints$positivePoints)  # External significant points
head(significantPoints$negativePoints)  # Internal significant points (if any)

## End(Not run)
```

---

matrixCreator	*Convert database from PET image data to a functional data matrix format*

Description

This function transforms a database created by [databaseCreator](#) into a matrix format suitable for functional data analysis. Each row of the matrix represents a subject's (patient or control) PET data, formatted as a continuous line of data points to simulate a functional representation.

Usage

```
matrixCreator(
  database,
  pattern = NULL,
  paramZ = 35,
  extractPattern = NULL,
  useSequentialNumbering = FALSE,
  quiet = FALSE
)
```

Arguments

database	A data frame created by [databaseCreator](#) containing PET image data with columns for group number, z, x, y, and pet values.
pattern	The regular expression pattern used to match filenames in the database. This should correspond to the naming conventions of the PET image files processed by databaseCreator.
paramZ	The specific z-coordinate slice to analyze. Default is 35.
extractPattern	Optional custom regular expression to extract subject numbers from filenames. Should contain a capture group for numerical ID.
useSequentialNumbering	
	If TRUE, assigns sequential numbers instead of extracting from filenames.
quiet	If TRUE, suppresses progress messages.

Details

This function is a critical step in the neuroSCC workflow for preparing data for Simultaneous Confidence Corridors (SCC) analysis. It performs several key operations:

1. Verifies the existence of the specified z-slice in the input database 2. Automatically calculates matrix dimensions based on the x and y coordinates of the specified z-slice 3. Extracts PET intensity values for each subject at the specified z-slice 4. Handles different group types (control or pathological) by recognizing appropriate identification columns 5. Provides flexible subject number extraction: - Uses a provided custom pattern - Falls back to sequential numbering - Supports default filename-based number extraction 6. Replaces any NaN values with zero to ensure matrix compatibility

The resulting matrix transforms multidimensional PET image data into a format suitable for functional data analysis techniques, particularly Simultaneous Confidence Corridors computation.

Typically follows [databaseCreator](#) and precedes [meanNormalization](#) in the neuroSCC analysis pipeline.

Value

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

See Also

[databaseCreator](#) for creating the input database [meanNormalization](#) for subsequent data normalization

Examples

```
# Assuming 'database_CN', 'pattern', and 'paramZ' are defined
SCC_CN <- matrixCreator(database_CN, pattern, paramZ = 35)
```

---

meanNormalization *Mean Average Normalization for Matrix Data*

Description

This function normalizes each row of the given matrix by its mean value. It divides each element in a row by the mean of that row, ignoring NA values. This normalization is a critical pre-processing step when comparing data from multiple sources to account for global intensity differences.

Usage

```
meanNormalization(
  matrixData,
  handleInvalidRows = c("warn", "error", "omit"),
  returnDetails = FALSE,
  quiet = FALSE
)
```

Arguments

matrixData   matrix, a matrix where each row represents data that needs to be normalized, typically the output from `matrixCreator`.

handleInvalidRows

character, specifies how to handle rows with invalid means (zero or NA). Options are: "warn" (default) warns and leaves row unnormalized, "error" stops with an error, or "omit" removes problematic rows from the result.

returnDetails   logical, if TRUE, returns a list containing the normalized matrix and additional diagnostic information. If FALSE (default), returns only the normalized matrix.

quiet   logical, if TRUE, suppresses console messages. Default is FALSE. -

Details

The function iterates over each row of the input matrix, calculates the mean of the row, and then divides each element of the row by the calculated mean. This process adjusts for global intensity differences between rows in the matrix.

In addition to normalization, the function performs statistical analysis on row means to identify potential outliers that might indicate data quality issues. Outlier detection uses the Interquartile Range (IQR) method with thresholds at 1.5xIQR and 3xIQR.

This function is typically used after `matrixCreator` in the neuroSCC workflow.

Value

If returnDetails=FALSE (default), returns a matrix where each row has been normalized by its mean value. If returnDetails=TRUE, returns a list containing:

- normalizedMatrix: The normalized matrix
- rowMeans: Vector of all calculated row means
- problemRows: Indices of rows with invalid means
- outlierRows: Indices of statistical outliers (based on 1.5*IQR)
- extremeOutlierRows: Indices of extreme outliers (based on 3*IQR)
- summary: Count of normal vs. problematic rows and thresholds used

See Also

matrixCreator for creating the data matrix to be normalized.

ImageSCC::scc2g.image for computing Simultaneous Confidence Corridors for the difference between groups using the normalized matrices of each group.

Examples

```
## Not run:
# In a typical workflow:
# Assuming data matrices already created with matrixCreator
matrixCN <- matrixCreator(database_CN, pattern, paramZ = 35)
matrixAD <- matrixCreator(database_AD, pattern, paramZ = 35)

# Basic usage - just normalize the matrices
matrixCN <- meanNormalization(matrixCN)
matrixAD <- meanNormalization(matrixAD)

# Advanced usage - get detailed diagnostics
results <- meanNormalization(matrixCN, returnDetails = TRUE)
normalizedMatrix <- results$normalizedMatrix

# Remove problematic rows automatically
cleanMatrix <- meanNormalization(matrixCN, handleInvalidRows = "omit")

## End(Not run)
```

---

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 from a NIFTI file*

Description

This function extracts contours from a neuroimaging NIFTI file where values change according to specified levels. It processes the NIFTI file with `neuroCleaner` to extract structured neuroimaging data, then extracts contours using `contoureR::getContourLines`. These contours serve as input for `Triangulation::TriMesh`, which is used in Simultaneous Confidence Corridors (SCCs) calculations.

While **not mandatory**, it is **highly recommended** that the input NIFTI file be pre-processed such that zero values represent the background and non-zero values represent regions of interest. The function's default behavior extracts contours at level `0`, which is ideal for well-masked data.

Usage

```
neuroContour(niftiFile, paramZ = 35, levels = c(0), plotResult = FALSE)
```

Arguments

`niftiFile`	`character`, the path to the NIFTI file containing neuroimaging data. Ideally, the file should be masked so that zero values represent the background.
`paramZ`	`integer`, the specific z-slice to extract contours from. Default is 35.
`levels`	`numeric`, a vector of levels at which to draw the contours. Default is `c(0)`.
`plotResult`	`logical`, if TRUE, plots the extracted contours. Default is `FALSE`.

Details

This function extracts contours from a **NIFTI** file, typically a **masked** image where background values are set to zero, and regions of interest contain non-zero values. While users can specify a different boundary level, the recommended approach is to use `levels = 0` for masked data.

The extracted contours are typically used as input to `Triangulation::TriMesh` to create a triangular mesh of the region, which is then used for Simultaneous Confidence Corridors calculations.

Value

A `list` of data frames, where each data frame contains the x and y coordinates of a contour. The first element typically represents the external boundary, while subsequent elements (if present) represent internal contours or holes. Each data frame has two columns:

- x: x-coordinates of the contour points
- y: y-coordinates of the contour points

See Also

[getContourLines](#) for the underlying contour extraction. `Triangulation::TriMesh` for the next step in the SCC calculation process.

Examples

```
## Not run:
# Load a sample NIFTI file from the neuroSCC package
library(neuroSCC)
maskPath <- system.file("extdata", "new_mask.nii", package = "neuroSCC")
```

```
# Extract contours from z-slice 35 and automatically show plot
contours <- neuroContour(maskPath, paramZ = 35, levels = c(0), plotResult = TRUE)

# Show the first few extracted points of the external contour
head(contours[[1]])

# If internal contours exist, show them as well
if (length(contours) > 1) {
  for (j in 2:length(contours)) {
    print(head(contours[[j]]))  # Show points of internal contours
  }
}

## End(Not run)
```

---

plotSCC   *Visualize SCC Results*

Description

This function visualizes Simultaneous Confidence Corridor (SCC) results as a heatmap, with the option to overlay significant points where detected differences exceed the confidence bands.

Usage

```
plotSCC(
  SCC_Result,
  title = "SCC Visualization",
  colorPalette = "viridis",
  showPoints = TRUE,
  alphaLevel = 0.05,
  xlab = NULL,
  ylab = NULL
)
```

Arguments

SCC_Result	A list containing SCC computation results from ImageSCC::scc.image().
title	A character string specifying the title of the plot. Default is '"SCC Visualization"'.
colorPalette	A character string specifying the color scheme. Options: '"viridis"', '"magma"', '"turbo"'. Default is '"viridis"'.
showPoints	A logical indicating whether to overlay significant SCC points. Default is TRUE.
alphaLevel	A numeric value specifying the confidence level threshold for SCC detection. Default is 0.05.
xlab	A character string specifying the x-axis label. Defaults to '"Longitudinal (1-X)"', where 'X' is extracted from SCC dimensions.
ylab	A character string specifying the y-axis label. Defaults to '"Transversal (1-Y)"', where 'Y' is extracted from SCC dimensions.

Details

- Extracts SCC matrix values and formats them for visualization. - Uses `getPoints()` to identify significant regions exceeding SCC thresholds. - Differentiates **positive** and **negative** SCC points using distinct markers.

Value

A heatmap plot displaying SCC values with optional overlays of significant points.

Examples

```
## Not run:
  # Load SCC results
  load("SCC_COMP.RData")

  # Basic SCC visualization
  plotSCC(SCC_COMP)

  # SCC plot with specific confidence threshold and custom color palette
 plotSCC(SCC_COMP, alphaLevel = 0.01, colorPalette = "magma", title = "SCC Analysis (alpha = 0.01)")

## End(Not run)
```

---

processROIs   *Process ROIs and Save Data Tables*

Description

This function processes **Regions of Interest (ROIs)** from binary NIfTI files containing manually defined hypoactive regions. It extracts voxel coordinates where 'pet = 1' (indicating hypoactivity) and formats them into structured '.RDS' tables.

The function is originally designed for the **Ph.D. thesis**: *"Development of statistical methods for the analysis of neuroimage data towards early diagnosis of neurodegenerative diseases"* conducted at the **University of Santiago de Compostela** by *Juan A. Arias Lopez, Prof. Pablo Aguiar-Fernández, Prof. Andrew H. Kemp, & Prof. Carmen Cadarso-Suárez*. However, options are provided for **other users and setups**.

Usage

```
processROIs(
 roiDir = "~/GitHub/PhD-2024-SCC-vs-SPM-SinglePatient-vs-Group/roisNormalizadas",
 outputDir =
  "~/GitHub/PhD-2024-SCC-vs-SPM-SinglePatient-vs-Group/roisNormalizadas/tables",
 regions,
 numbers,
 filePattern = "wwwx{region}_redim_crop_squ_flipLR_newDim_C{number}.nii",
 verbose = FALSE
)
```

Arguments

roiDir	character, the base directory where the ROI NIfTI files are stored. Default: '"~/GitHub/PhD-2024-SCC-vs-SPM-SinglePatient-vs-Group/roisNormalizadas"'.
outputDir	character, the directory where processed 'ROItable_*' files will be saved. Default: '"~/GitHub/PhD-2024-SCC-vs-SPM-SinglePatient-vs-Group/roisNormalizadas/tables"'.
regions	character vector, specifying the ROI region names (e.g., 'c("w32", "w79")').
numbers	integer vector, specifying the patient/control numbers (e.g., '1:16').
filePattern	character, a template for the expected NIfTI filenames. Should include {region} and {number} placeholders. Default: "wwwx\{region\}_redim_crop_squ_flipLR_newDim_C\{num
verbose	logical, if TRUE, prints progress messages. Default: FALSE.

Details

- This function **extracts voxel coordinates** ('x, y, z') where 'pet = 1' (indicating hypoactivity).
- The **default filename structure** is based on the research thesis setup but can be adjusted.
- Processed tables are saved as **'ROItable_*'** '.RDS' files for later use in **SCC vs. SPM comparisons**.

Value

This function does not return a value. It **processes and saves '.RDS' files** in 'outputDir'.

Examples

```
## Not run:
# Process ROIs using the thesis default setup
processROIs()

# Custom example for a different dataset
processROIs(
  roiDir = "/custom/path/rois",
  outputDir = "/custom/path/tables",
  regions = c("region1", "region2"),
  numbers = 1:10,
  filePattern = "custom_prefix_{region}_sub_{number}.nii",
  verbose = TRUE
)

## End(Not run)
```

Index