Package ‘BioTrajectory’

May 7, 2025

Type Package

Title Image Processing Tools for Barnes Maze Experiments

Version 1.0.0

Description Tools to process the information obtained from experiments conducted in the Barnes Maze. These tools enable the detection of trajectories generated by subjects during trials, as well as the acquisition of precise coordinates and relevant statistical data regarding the results. Through this approach, it aims to facilitate the analysis and interpretation of observed behaviors, thereby contributing to a deeper understanding of learning and memory processes in such experiments.

License LGPL-3

Imports av, imager, png, tiff, jpeg, MASS, graphics, utils, grDevices, dplyr, grid, rpanel, tools, stats

Encoding UTF-8

Author Antonio Guerrero [cre],
Vanessa Ramirez [aut],
Jorge Macias [ctb]

Maintainer Antonio Guerrero <jaguerrero@correo.uaa.mx>

NeedsCompilation no

RoxygenNote 7.3.2.9000

Repository CRAN

Date/Publication 2025-05-07 12:00:06 UTC

Contents

`detectBarnes` *Detect Circles in a Maze Image*

Description

The function first applies the Canny edge detection algorithm to the input image. It then detects circles corresponding to the board and holes using the `detectCircles` function. The function filters the detected hole circles based on their distance from the board circle, ensuring they fall within acceptable ranges relative to the specified radii. Finally, it calculates the median radius and angle differences of the remaining circles and constructs a circular representation for the holes based on these parameters.

Usage

```
detectBarnes(im, boardRadius, holeRadius, sigma = 25, plot = TRUE)
```

Arguments

`im`	A matrix representing the image where the circles will be detected.
`boardRadius`	The expected radius of the board circles.
`holeRadius`	The expected radius of the hole circles.
`sigma`	An optional parameter that controls the standard deviation for the Gaussian filter used in edge detection (default value is 25).
`plot`	Opcion of plot

Details

Detects circles representing boards and holes in a given image of a maze. It utilizes edge detection to identify potential circle patterns based on specified radius parameters. The function returns the detected circles' coordinates, along with additional information about the board and hole radii.

Value

A list containing:

`c1`	Coordinates of the detected board circle.
`boardRadius`	Radius of the board circle.
`c2`	A list of coordinates representing the detected hole circles.
`holeRadius`	Radius of the hole circles.

Examples

```
path <- system.file('extdata/data.tiff', package='BioTrajectory')
im <- tiff::readTIFF(path)
im <- imager::as.cimg(t(im[,,1]))
Barnes <- detectBarnes(im, boardRadius=207, holeRadius=13, sigma=25)
```

---

distanceToTarget *Calculates distances to a target point*

Description

Given a dataset of points, it computes the Euclidean distances from each point to a specified target point. The function also identifies contiguous segments of points that fall within a specified radius around the target.

Usage

```
distanceToTarget(data, target, targetRadious = 0)
```

Arguments

`data`	A matrix or data frame where each row represents a point in multi-dimensional space.
`target`	A numeric vector representing the coordinates of the target point.
`targetRadious`	A numeric value indicating the radius around the target point. Points within this radius are considered to be close to the target. Default is 0.

Value

A list containing:

`distance`	A numeric vector of distances from each point in ‘data‘ to the ‘target‘.
`r`	A list with two components: ‘start‘, indicating the starting indices of contiguous segments of points within the target radius, and ‘length‘, indicating the lengths of these segments.
`target_radious`	The radius around the target point.

Examples

```
# Create an example data matrix
data <- matrix(c(1, 1, 2, 2, 3, 3), ncol = 2, byrow = TRUE)
target <- c(2, 2)
result <- distanceToTarget(data, target, targetRadious = 0)
print(result)
```

`distanceTraveled` *Calculates the total distance traveled through a series of points.*

Description

Computes the cumulative distance between consecutive points in a given set of coordinates. It sums the Euclidean distances between each pair of adjacent points, providing the total distance traveled along the path defined by the points.

Usage

```
distanceTraveled(points)
```

Arguments

`points`	A matrix or data frame where each row represents a point with its coordinates (e.g., x and y).

Value

A numeric value representing the total distance traveled through the points.

Examples

```
# Define a set of points
points <- matrix(c(0, 0, 1, 1, 2, 0), ncol = 2, byrow = TRUE)
# Calculate the distance traveled
total_distance <- distanceTraveled(points)
# Print the total distance
print(total_distance)
```

`getRadius` *Estimate the Radius of a Circle Fitting Four Points*

Description

This function estimates the center and radius of a circle that best fits four points provided by the user. The user interacts with the plot to select four points, and the function optimizes the parameters (center and radius) using a least squares approach.

Usage

```
getRadius(frame)
```

Arguments

`frame`	The plot or frame to display the points and interact with the user.

Details

The function plots the provided ‘frame‘ and then uses the ‘locator()‘ function to allow the user to click on four points. The function then performs optimization using the ‘optim()‘ function to minimize the difference between the selected points and the circle’s equation.

The objective function (‘ftemp‘) calculates the sum of squared differences between the points and the expected distance from the circle’s center.

Value

A vector of length 3, where: - The first element is the x-coordinate of the circle center (cx). - The second element is the y-coordinate of the circle center (cy). - The third element is the radius of the circle (r).

Examples

```
if(interactive()){
path <- system.file('extdata/data.tiff', package='BioTrajectory')
im <- tiff::readTIFF(path)
im <- imager::as.cimg(t(im[,,1]))
frame <- plot(im)
circle_params <- getRadius(frame)
print(circle_params)
}
```

`getTrajectory` *Get Object Trajectory from Image Sequence*

Description

This function calculates the trajectory of an object in a sequence of images. It compares each frame to a background image to detect movement. The function identifies the largest object in each frame and calculates its centroid coordinates across the sequence.

Usage

```
getTrajectory(listImages, Barnes, iBackground, iBegin, iEnd)
```

Arguments

`listImages`	A vector of file paths to the images in the sequence.
`Barnes`	A parameter used by the 'removeBackground()' function to remove background noise.
`iBackground`	An index indicating which image from the sequence is used as the background.
`iBegin`	An index specifying the first image in the sequence to start tracking.
`iEnd`	An index specifying the last image in the sequence to track.

Details

The function reads the images from 'listImages' and compares each frame to the background image specified by 'iBackground'. Background subtraction is performed using the 'removeBackground()' function, followed by thresholding to identify significant changes between the background and the current frame. The largest connected component in the thresholded image is assumed to be the object of interest, and its centroid is calculated. The trajectory is tracked across frames, and the centroid coordinates are returned for each frame.

Value

A data frame with two columns: the x and y coordinates of the object centroid in each frame. If no object is detected in a frame, the coordinates are set to 'NA'.

Examples

```
# Not run:
path <- system.file('extdata/frames', package='BioTrajectory')
images <- list.files(path, full.names = TRUE)
B <- list(c1 = structure(list(x = 342L, y = 263L), row.names = 1L, class = "data.frame"),
   r1 = 207, c2 = structure(list(x = c(157L, 172L, 202L, 245L,
   297L, 352L, 408L, 455L, 494L, 517L, 522L, 507L, 476L, 430L,
   375L, 318L, 262L, 215L, 180L, 160L), y = c(242L, 188L, 141L,
   105L, 85L, 80L, 93L, 124L, 166L, 219L, 277L, 334L, 383L,
   420L, 440L, 442L, 426L, 394L, 350L, 298L), class = "data.frame"), r2 = 13))
trajectory <- getTrajectory(images, B, 1, 1, 1)
print(trajectory)
```

`heatmapFromPoints` *Creates a heatmap from a set of points.*

Description

Generates a heatmap based on the density of points in a given dataset. It utilizes kernel density estimation to visualize the concentration of points, allowing for adjustments in color palette and plotting limits.

Usage

```
heatmapFromPoints(data, plim = NULL, plot.pal = TRUE, ...)
```

Arguments

`data`	A data frame containing the coordinates of the points with columns named 'x' and 'y'.
`plim`	Optional vector of length two specifying the plotting limits for the density values. If provided, values outside this range will be adjusted.
`plot.pal`	A logical value indicating whether to plot the color palette alongside the heatmap (default is TRUE).
`...`	Additional graphical parameters to customize the image.

Value

A matrix of density values corresponding to the heatmap.

Examples

```
# Generate example data
set.seed(123)
data <- data.frame(x = rnorm(1000), y = rnorm(1000))

# Create a heatmap from the points
heatmap_result <- heatmapFromPoints(data)
```

`interpolateTrajectory` *Interpolates a trajectory*

Description

Given a dataset of points, it interpolates to generate intermediate points between them.

Usage

```
interpolateTrajectory(data, n = 4)
```

Arguments

`data`	A data frame with at least two columns: 'x' and 'y', representing the coordinates of the points.
`n`	An integer indicating the number of intermediate points to generate between each pair of points. Default is 4.

Value

A data frame containing the ‘x‘ and ‘y‘ coordinates, which includes both the original points and the interpolated points.

Examples

```
# Create an example data frame
data <- data.frame(x = c(1, 2, 3), y = c(1, 4, 9))
# Interpolate the trajectory
interpolated <- interpolateTrajectory(data, n = 4)
print(interpolated)
```

`nearestTarget` *Finds the nearest targets to a set of points within a specified radius.*

Description

Calculates the distances between a set of points and target locations. It identifies the nearest target for each point and checks if the distance is within a specified radius. If a target is found within the radius, its index and distance are returned; otherwise, -1 is returned for both.

Usage

```
nearestTarget(points, targets, r)
```

Arguments

`points`	A matrix or data frame containing the coordinates of the points with rows representing points.
`targets`	A matrix or data frame containing the coordinates of the target locations with rows representing targets.
`r`	A numeric value specifying the radius within which to consider targets.

Value

A data frame with two columns:

`nt`	Index of the nearest target for each point. If no target is found within the radius, this will be -1.
`d`	Distance to the nearest target. If no target is found within the radius, this will be -1.

Examples

```
# Define a set of points and targets
points <- matrix(c(1, 2, 3, 4), ncol = 2)
targets <- matrix(c(2, 3, 5, 6), ncol = 2)
radius <- 2

# Find the nearest targets
nearest_results <- nearestTarget(points, targets, radius)

# Print the results
print(nearest_results)
```

---

readImage *Read and Resize an Image*

Description

This function reads an image from a file path and resizes it to a specified number of rows. It supports several image formats, including JPG, JPEG, PNG, TIFF, and TIF. The function also converts the image into a suitable format for further processing.

Usage

```
readImage(path, resizeRows = 640)
```

Arguments

`path`	The file path of the image to be read.
`resizeRows`	The desired number of rows (height) to resize the image. The aspect ratio of the image will be maintained during resizing. Default is 640 rows.

Details

The function detects the image format based on the file extension. It currently supports the following formats: JPG, JPEG, PNG, TIFF, and TIF. If the image has more than 3 dimensions (such as an RGBA image with an alpha channel), the alpha channel is discarded. The image is resized only if its height exceeds the specified ‘resizeRows‘.

Value

A cimg object containing the resized image.

Examples

```
# Example usage
img_path <- system.file('extdata/data.tiff', package='BioTrajectory')
img <- readImage(img_path, resizeRows = 500)
plot(img)  # Visualizes the resized image
```

readtrackData *Reads tracking data from a specified file.*

Description

Reads a text file containing tracking data, where each line represents coordinates. If a line contains "null", it adds NA values for that entry. Optionally, it can remove rows with NA values.

Usage

```
readtrackData(file, na.rm = FALSE)
```

Arguments

`file`	A character string specifying the path to the file containing the tracking data.
`na.rm`	A logical value indicating whether to remove rows with NA values (default is FALSE).

Value

A data frame with two columns (‘x‘ and ‘y‘) containing the coordinates read from the file. If ‘na.rm‘ is TRUE, rows with NA values are omitted.

Examples

```
# Read tracking data from a file
path <- system.file('extdata/track.txt', package='BioTrajectory')
tracking_data <- readtrackData(path, na.rm = TRUE)
# Print the resulting data frame
print(tracking_data)
```

selFrame *Image Frame Selector and Viewer*

Description

This function provides an interactive image frame viewer where the user can navigate through a sequence of images and select a specific one. The viewer uses buttons for navigation (previous and next) and an option to select an image. The function uses the ‘rp.control‘ to create a window for the interface, and ‘grid‘ for displaying images.

Usage

```
selFrame(image_files)
```

Arguments

`image_files`	A vector of file paths to the images that will be displayed in the viewer.

Details

The function displays a sequence of images from the provided file paths, allowing the user to navigate between them using "Previous" and "Next" buttons. The images are displayed using ‘grid.raster‘ from the ‘grid‘ package. A "Select" button allows the user to select the current image, and the function returns the index of the selected image.

Value

The index of the selected image.

Examples

```
# Not run:
path <- system.file('extdata/frames/', package='BioTrajectory')
image_files <- list.files(path, pattern = "\\.png$", full.names = TRUE)
selFrame(image_files)
```

`showDistanceToTarget` *Visualizes the distance to the target in a plot.*

Description

This function generates a plot of the distance to a target over time or some other associated parameter in the object. It also draws a horizontal line to indicate the target radius and a red segment that shows the start and length of a parameter associated with the ‘obj‘.

Usage

```
showDistanceToTarget(obj, ...)
```

Arguments

`obj`	An object that must contain at least three elements: distance:A numeric vector representing the distance to the target targetRadious: A numeric value indicating the target radius. r:An object containing at least two elements: - start: A numeric value representing the start of the range. - length: A numeric value representing the length of the range.
`...`	Additional parameters of object.

Details

This function generates a line plot of the distance to a target (stored in ‘obj$distance‘). It also draws a red horizontal line at the value of ‘obj$targetRadious‘ to indicate the target’s radius. Additionally, a red vertical segment is drawn to show the start of the range defined by ‘obj$r$start‘ and the length defined by ‘obj$r$length‘.

Value

A plot showing the relationship between the object’s position and the target’s position.

Examples

```
# Create a fictional object with example data
obj <- list(
  distance = rep(c(5, 10, 15, 20, 15, 10, 5),5),
  targetRadious = 12,
  r = list(start = 2, length = 10)
)
# Visualize the distance to the target using the function
showDistanceToTarget(obj)
```

`showTrajectory` *Displays a trajectory plot from a set of coordinates.*

Description

Visualizes the trajectory represented by a series of points. The function can display the full trajectory or a subset based on a specified step size. It supports customizable color palettes for enhanced visualization.

Usage

```
showTrajectory(data, stepSize = 0, pal)
```

Arguments

`data`	A data frame containing the trajectory coordinates with columns ‘x‘ and ‘y‘.
`stepSize`	An integer specifying the interval for plotting segments of the trajectory. If set to 0, the entire trajectory is plotted. Default is 0.
`pal`	A color palette to be used for plotting the trajectory segments.

Value

A plot displaying the trajectory based on the provided coordinates and settings.

Examples

```
# Generate example trajectory data
path <- system.file('extdata/track.txt', package='BioTrajectory')
data <- BioTrajectory::readtrackData(path)
data <- na.omit(data)
palette <- grDevices::colorRampPalette(c("purple","blue","cyan","yellow","orange","red"))
# Show the full trajectory
showTrajectory(data, stepSize = 0, pal = palette)
# Show the trajectory with a step size of 36
showTrajectory(data, stepSize = 36, pal = palette)
```

`videoToFrames` *Extract Frames from a Video and Save as Images*

Description

This function takes a video file, processes it, and saves the frames as individual images in the specified directory. The images are saved in PNG format by default, and at a rate of 1 frame per second.

Usage

```
videoToFrames(videoPath, outputDir)
```

Arguments

`videoPath`	Path to the video file.
`outputDir`	Path to the directory where the extracted images will be saved.

Value

A list of file paths to the generated images.

Examples

```
# Not run:
videoPath <- system.file('extdata/video.mp4', package='BioTrajectory')
outputDir <- system.file('extdata/frames/', package='BioTrajectory')
images <- videoToFrames(videoPath, outputDir)
print(images)  # Displays the paths to the generated images
```

Index