Package 'BioTrajectory'

May 6, 2026

Type Package

Title Image Processing Tools for Barnes Maze Experiments

Version 1.1.0

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 7.3.3

Repository CRAN

Date/Publication 2025-11-26 16:50:01 UTC

Contents

`as.trajectory` *Convert to Trajectory Object*

Description

Converts an object of type ‘data.frame‘ or similar into a ‘Trajectory‘ object. The ‘data.frame‘ should contain columns ‘x‘ and ‘y‘ representing the coordinates.

Usage

```
as.trajectory(x, delta_time = 1)
```

Arguments

`x`	An object of class ‘data.frame‘ with ‘x‘ and ‘y‘ columns.
`delta_time`	Control variable with a default value of 1.

Value

A ‘Trajectory‘ object containing the coordinates from the ‘data.frame‘.

centroid *Calculate the Centroid of Object*

Description

This function calculates the centroid of a set of points contained in an object of class ‘trajectory‘. The centroid is the average of the coordinates of all the points.

Usage

```
centroid(obj)
```

Arguments

obj	An object of class ‘trajectory‘ that must contain a component named ‘points‘.

Value

A numeric vector containing the coordinates of the centroid of the points stored in ‘obj$points‘.

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`	An object of class 'trajectory' containing a collection of points with coordinates.
`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.

`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(data)
```

Arguments

`data`	An object of class ‘trajectory‘ containing a collection of points with coordinates.

Value

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

`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.

Instead of directly returning the centroid coordinates, the function now creates an object of class ‘trayectoria‘ to store the calculated centroids. The coordinates of the object in each frame are added to the ‘trayectoria‘ object. If no object is detected in a frame, the corresponding coordinates are set to ‘NA‘. The resulting ‘trayectoria‘ object is returned, which allows users to further manipulate and analyze the trajectory of the object across frames.

The returned ‘trayectoria‘ object contains methods for extracting and analyzing the coordinates, making it easier to track the object over time.

Value

An object of class ‘trayectoria‘ containing the points (coordinates) of the object in each frame. The ‘trayectoria‘ object will include the ‘x‘ and ‘y‘ coordinates. If no object is detected in a frame, the corresponding coordinates will be set to ‘NA‘ in the ‘trayectoria‘ object.

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"),
    boardRadius = 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")), holeRadius = 13)

trajectory <- getTrajectory(images, B, 1, 1, 2)
print(trajectory)
```

---

heatmapFromTrajectory *Creates a heatmap from a trajectory.*

Description

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

Usage

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

Arguments

`data`	An object of class ‘trajectory‘ containing a collection of points with coordinates.
`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.

`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`	An object of class ‘trajectory‘ containing a collection of points with coordinates.
`n`	An integer indicating the number of intermediate points to generate between each pair of points. Default is 4.

Value

An object of class ‘trayectoria‘, which contains the ‘x‘ and ‘y‘ coordinates, including both the original points and any interpolated points. The ‘trayectoria‘ object allows users to store, manipulate, and analyze the coordinates of the object across frames. If no object is detected in a frame, the corresponding coordinates will be set to ‘NA‘.

is.trajectory *Check if Object is a Trajectory*

Description

Checks if an object is of class 'Trajectory'.

Usage

```
is.trajectory(x)
```

Arguments

x	An object to check.

Value

A logical value ('TRUE' or 'FALSE') indicating whether the object is of class 'Trajectory'.

isClosed *Check if the first and last points of a "trajectory" object are within a given tolerance*

Description

This function checks if the distance between the first and last points in an object of class 'trajectory' is smaller than a specified tolerance 'tol'. The distance is calculated using the Euclidean distance between the two points.

Usage

```
isClosed(obj, tol = 1e-06)
```

Arguments

obj	An object of class 'trajectory' that contains a component 'points'.
tol	A numeric value specifying the tolerance. If the distance between the first and last points is smaller than this value, the function returns 'TRUE'. Default is '1e-6'.

Value

A logical value ('TRUE' or 'FALSE'), indicating whether the distance between the first and last points is smaller than 'tol'.

length.trajectory *Length of the Trajectory*

Description

Calculates the total length of the trajectory, which is the sum of the distances between consecutive points in the ‘Trajectory‘ object.

Usage

```
## S3 method for class 'trajectory'
length(x)
```

Arguments

x	An object of class ‘Trajectory‘.

Value

A numeric value representing the total length of the trajectory.

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(data, targets, r)
```

Arguments

data	An object of class ‘trajectory‘ containing a collection of points with coordinates.
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.

`normalVectors` *Calculate the normals from tangents of a trajectory object*

Description

This function computes the normal vectors at each point of an object of class ‘trajectory‘ by rotating the tangent vectors 90 degrees.

Usage

```
normalVectors(obj)
```

Arguments

`obj`	An object of class ‘trajectory‘ that contains a component ‘points‘.

Value

A matrix of the same dimensions as ‘obj$points‘ with the normal vectors. Each row represents a normal vector at a given point, which is obtained by rotating the corresponding tangent vector by 90 degrees.

`or` *Combine Two Trajectories*

Description

Combines two ‘Trajectory‘ objects by concatenating their points. The result is a new trajectory that contains the coordinates of both input trajectories.

Usage

```
## S3 method for class 'trajectory'
tr1 | tr2
```

Arguments

tr1	A ‘Trajectory‘ object.
tr2	A ‘Trajectory‘ object.

Value

A new ‘Trajectory‘ object containing the concatenated points of the two input trajectories.

---

plot.trajectory *Plot the Trajectory*

Description

Plots the trajectory by displaying the ‘x‘ and ‘y‘ coordinates as points on a 2D plot. Optionally, you can add a line connecting the points.

Usage

```
## S3 method for class 'trajectory'
plot(x, ..., stepSize = 0)
```

Arguments

x	A ‘Trajectory‘ object to plot.
...	Other parameters to be passed.
stepSize	An integer specifying the interval for plotting segments of the trajectory. If set to 0, the entire trajectory is plotted. Default is 0.

Value

A plot displaying the trajectory.

---

print.summary.trajectory *Print Summary of the Trajectory*

Description

Prints a detailed summary of the trajectory, including statistics for both the ‘x‘ and ‘y‘ coordinates and the total number of frames.

Usage

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

Arguments

x	A ‘summary.trajectory‘ object, typically returned by the ‘summary.trajectory‘ method.
...	Other parameters to be passed.

Value

Prints the summary directly.

---

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)
```

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.

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

An object of class ‘trayectoria‘ containing the points (coordinates) of the object in each frame. The ‘trayectoria‘ object will include the ‘x‘ and ‘y‘ coordinates. If no object is detected in a frame, the corresponding coordinates will be set to ‘NA‘ in the ‘trayectoria‘ object.

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)
```

reverseTrajectory *Reverse the points of a trajectory object and create a new trajectory*

Description

This function reverses the order of the points in an object of class ‘trajectory‘ and then constructs a new trajectory using the reversed points. The function assumes the existence of a ‘trajectory‘ function that creates a trajectory from the given x and y coordinates, as well as a time step ‘delta_time‘.

Usage

```
reverseTrajectory(obj)
```

Arguments

obj	An object of class ‘trajectory‘ that contains a component ‘points‘, which is a matrix or data frame with the coordinates of the points. It is assumed that ‘points‘ has columns ‘x‘ and ‘y‘ representing the coordinates of the points in the trajectory.

Value

A new trajectory object created using the reversed points and the original ‘delta_time‘ from ‘obj‘.

---

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)
```

smoothTrajectory *Apply a moving average filter to the trajectory and create a smoothed trajectory*

Description

This function applies a moving average filter to the ‘x‘ and/or ‘y‘ coordinates of a "trajectory" object. The filter is applied using a window of a specified size. The function then creates a new trajectory object using the smoothed coordinates. The filter is applied symmetrically around each point.

Usage

```
smoothTrajectory(obj, method = c("x", "y", "both"), window = 3)
```

Arguments

obj	An object of class ‘trajectory‘ that contains a component ‘points‘, which is a data frame with the coordinates of the points. It is assumed that ‘points‘ has columns ‘x‘ and ‘y‘ representing the trajectory coordinates.
method	A character string specifying which coordinate(s) to smooth. Can be one of ‘"x"‘, ‘"y"‘, or ‘"both"‘. Default is ‘"both"‘, which smooths both the ‘x‘ and ‘y‘ coordinates.
window	An odd integer specifying the window size for the moving average filter. The window must be an odd number to ensure a symmetric filter. Default is 3.

Value

A new trajectory object created using the smoothed ‘x‘ and ‘y‘ coordinates.

subsampleTrajectory *Subsample the points of a trajectory object and create a new trajectory*

Description

This function subsamples the points of a ‘trajectory‘ object by selecting every ‘step‘-th point. It then creates a new trajectory using the selected points, adjusting the ‘delta_time‘ accordingly.

Usage

```
subsampleTrajectory(obj, step = 2)
```

Arguments

obj	An object of class ‘trajectory‘ that contains a component ‘points‘. The ‘points‘ component must have columns ‘x‘ and ‘y‘ representing the trajectory coordinates.
step	A positive integer that defines the step size for subsampling the points. Default is ‘2‘, meaning every second point will be selected.

Value

A new trajectory object created using the subsampled points and the adjusted ‘delta_time‘.

---

summary.trajectory *Summary of the Trajectory*

Description

Generates a summary of the trajectory, including the minimum, maximum, and mean of the ‘x‘ and ‘y‘ coordinates, as well as the total number of frames.

Usage

```
## S3 method for class 'trajectory'
summary(object, ...)
```

Arguments

object	An object of class ‘Trajectory‘.
...	Other parameters to be passed.

Value

A summary object containing statistics for both the ‘x‘ and ‘y‘ coordinates of the trajectory.

---

tangentVectors *Calculate the tangent cectors of a trajectory object*

Description

This function calculates the unit tangent vectors at each point of the trajectory represented by a ‘trajectory‘ object. The tangent vectors are computed by taking the difference between consecutive points and normalizing them to have unit length.

Usage

```
tangentVectors(obj)
```

Arguments

obj	An object of class ‘trajectory‘ that contains a component ‘points‘.

Value

A matrix where each row is a unit tangent vector corresponding to each pair of consecutive points in ‘obj$points‘.

---

trajectory *Trajectory Class*

Description

A class representing a trajectory of an object over time, including the ‘x‘ and ‘y‘ coordinates of the object's centroids across frames. It provides methods to manipulate and analyze the trajectory, such as calculating the length, summarizing the trajectory, printing the summary, and plotting the trajectory.

Usage

```
trajectory(x, y, delta_time = 1)
```

Arguments

x	Numeric vector with the 'x' coordinates of the trajectory.
y	Numeric vector with the 'y' coordinates of the trajectory.
delta_time	Control variable with a default value of 1.

Details

The ‘Trajectory‘ class stores the coordinates of an object across multiple frames. The class provides methods to compute various aspects of the trajectory, such as its length and summary statistics, as well as visualizations. It can also be used to manipulate the trajectory, such as combining it with another trajectory using the ‘|‘ operator.

trimTrajectory *Trim a trajectory object to a subset of points*

Description

This function trims the points of a ‘trajectory‘ object by selecting a subset of the points from the specified ‘from‘ to ‘to‘ indices. It then creates a new trajectory using the selected points.

Usage

```
trimTrajectory(obj, from = 1, to = nrow(obj$points))
```

Arguments

obj	An object of class ‘trajectory‘ that contains a component ‘points‘, which is a data frame with the coordinates of the points in the trajectory.
from	A positive integer indicating the starting index of the points to include. Default is 1.
to	A positive integer indicating the ending index of the points to include. Default is the total number of rows in ‘obj$points‘.

Value

A new trajectory object created using the subset of points from the ‘from‘ to ‘to‘ indices.

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

Description

This function is a wrapper that simplifies extracting frames from a video. It processes the video file and saves the frames as PNG images in the specified directory. By default, it extracts 15 frames per second.

Usage

```
videoToFrames(videoPath, outputDir, fps = 15)
```

Arguments

videoPath	Path to the video file.
outputDir	Path to the directory where the extracted images will be saved.
fps	The number of frames shown per second in the video. By default, it extracts 15 frames per second.

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, fps=15)
print(images)  # Displays the paths to the generated images
```

Index