Package ‘servosphereR’

May 14, 2019

Type Package

Title Analyze Data Generated from Syntech Servosphere Trials

Version 0.1.1

Description Functions that facilitate and speed up the analysis of data produced by a Syntech servosphere <http://www.ockenfels-syntech.com/products/locomotion-compensation/>,
which is equipment for studying the movement behavior of arthropods.
This package is designed to make working with data produced from a servosphere easy for someone new to or unfamiliar with R. The functions provided in this package fall into three broad-use categories: functions for cleaning raw data produced by the servosphere software, functions for deriving movement variables based on position data, and functions for summarizing movement variables for easier analysis. These functions are built with functions from the tidyverse package to work efficiently, as a single servosphere file may consist of hundreds of thousands of rows of data and a user may wish to analyze hundreds of files at a time. Many of the movement variables derivable through this package are described in the following papers:
Otálora-Luna, Fernando; Dickens, Joseph C. (2011) <doi:10.1371/journal.pone.0020990>
Party, Virginie; Hanot, Christophe; Busser, Daniela Schmidt; Rochat, Didier; Renou, Michel (2013) <doi:10.1371/journal.pone.0052897>
Bell, William J.; Kramer, Ernest (1980) <doi:10.1007/BF01402908>
Becher, Paul G; Guerin, Patrick M. (2009) <doi:10.1016/j.jinsphys.2009.01.006>.

License GPL-3

Encoding UTF-8

LazyData true

URL http://github.com/wittja01/servosphereR

BugReports http://github.com/wittja01/servosphereR/issues

RoxygenNote 6.1.1

Imports purrr (>= 0.2), data.table (>= 1.10), dplyr (>= 0.7), magrittr (>= 1.5), rlang (>= 0.3), stats (>= 3.4)

Depends R (>= 2.10)

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

NeedsCompilation no

Author Jacob T. Wittman [aut, cre],
Brian H. Aukema [ctb]

Maintainer Jacob T. Wittman <wittja01@gmail.com>

Repository CRAN

Date/Publication 2019-05-14 19:10:02 UTC

R topics documented:

`aggregateData` *Aggregate data frames*

Description

Aggregate the data frames in a list to reduce noise in movement recordings

Usage

```
aggregateData(list, n)
```

Arguments

`list` A list of data frame objects.

`n` The number of consecutive rows to aggregate over.

Details

This function requires that the change in time and change in position column names are "dT", "dx", and "dy", respectively.

Movement recording data generated by the servosphere software is quite noisy due to extraneous movements by the insect or side-to-side motion. Aggregating the data by aggregating rows reduces this noise. Aggregating also reduces the size of the data files, making future computing tasks faster.

To aggregate the data properly, we recommend watching recordings of the target organism moving on the servosphere and determine the length of time it takes the insect to move at least 50 percent of its body length. Use the time it takes the insect to move at least 50 percent of its body length as the minimum aggregation time. Longer periods may be necessary depending on the size and movement of the insect.

The resulting data frames will have an additional column named "length". The values in length should be approximately equal to the sum of dT from the `n` rows aggregated over, i.e. if each observation from the un-aggregated data is 10 ms and the user aggregates these observations to 1 second, the value of length should be approximately 1. Note that in aggregating the data frame, the stimulus column will also be aggregated The stimulus status value at the first row to be aggregated will become the value of the stimulus for the aggregated row.

Value

A list of aggregated data frames with an additional column, `length`, to check that the function worked.

Examples

```
# Aggregates every 5 rows in each data frame. Must be used after
# cleanNames()
 servosphere <- list(data.frame(id = rep(1, 200),
                               stimulus = rep(c(0, 1), each = 100),
                               dT = sample(8:12, 200, replace = TRUE),
                               dx = runif(200, 0, 5),
                               dy = runif(200, 0, 5),
                               treatment = rep("a", 200),
                               date = rep("2032018", 200)),
                    data.frame(id = rep(2, 200),
                               stimulus = rep(c(0, 1), each = 100),
                               dT = sample(8:12, 200, replace = TRUE),
                               dx = runif(200, 0, 5),
                               dy = runif(200, 0, 5),
                               treatment = rep("b", 200),
                               date = rep("2032018", 200)))
 servosphere <- aggregateData(servosphere, n = 5)
```

`atan3` *Calculate angle between a vector and y-axis*

Description

Calculate the angle between a vector and y-axis, not the x-axis as in atan2

Usage

```
atan3(x, y)
```

Arguments

`x` The x coordinate from the (x, y) locations.

`y` The y coordinate from the (x, y) location for an organism.

Details

This is a helper function. It is called in another function but is otherwise not "outward facing".

This function calculates the angle between a vector and the y-axis. The base function atan2 calculates the angle between a vector and the x-axis, which is not desirable in this case.

If the data will be aggregated, it is recommended to aggregate the data before running this function.

`calcBearing` *Calculate bearing*

Description

Calculates the bearing (direction of movement) for each time step

Usage

```
calcBearing(list)
```

Arguments

`list` A list of data frames with separate columns for x and y coordinate values.

Details

This function requires that the data has been previously processed with the `calcXY()` function, providing (x, y) coordinates. Calculate the direction moved by the organism relative to the y axis between each time step in your data frames.

If the data will be aggregated, it is recommended to aggregate the data before running this function.

Value

A list of data frames with a column for the bearing of the organism at each time step.

Examples

```
# Provide a list of data frames with two columns for the (x, y) coordinates

 servosphere <- list(data.frame(id = rep(1, 200),
                                stimulus = rep(c(0, 1), each = 100),
                                dT = sample(8:12, 200, replace = TRUE),
                                dx = runif(200, 0, 5),
                                dy = runif(200, 0, 5),
                                treatment = rep("a", 200),
                                date = rep("2032018", 200)),
                     data.frame(id = rep(2, 200),
                                stimulus = rep(c(0, 1), each = 100),
                                dT = sample(8:12, 200, replace = TRUE),
                                dx = runif(200, 0, 5),
                                dy = runif(200, 0, 5),
                                treatment = rep("b", 200),
                                date = rep("2032018", 200)))
servosphere <- calcXY(servosphere)

servosphere <- calcBearing(servosphere)
```

`calcDistance` *Calculate distance*

Description

Calculate distance moved between time steps

Usage

```
calcDistance(list)
```

Arguments

`list`	A list of data frames, each of which has a column for dx and dy.

Details

Use the dx and dy columns in the servosphere data frames to calculate the distance moved between each time recording. If the data will be aggregated, it is recommended to aggregate the data before running this function.

Value

A list of data frames, each of which has a variable for the distance moved between each data recording.

Examples

```
 servosphere <- list(data.frame(id = rep(1, 200),
                                stimulus = rep(c(0, 1), each = 100),
                                dT = sample(8:12, 200, replace = TRUE),
                                dx = runif(200, 0, 5),
                                dy = runif(200, 0, 5),
                                treatment = rep("a", 200),
                                date = rep("2032018", 200)),
                     data.frame(id = rep(2, 200),
                                stimulus = rep(c(0, 1), each = 100),
                                dT = sample(8:12, 200, replace = TRUE),
                                dx = runif(200, 0, 5),
                                dy = runif(200, 0, 5),
                                treatment = rep("b", 200),
                                date = rep("2032018", 200)))
servosphere <- calcDistance(servosphere)
```

`calcTurnAngle` *Calculate turn angle*

Description

Calculate the turn angle between two successive moves

Usage

```
calcTurnAngle(list)
```

Arguments

`list`	A list of data frames, where each data frame has a column for bearing.

Details

For this function to work, the data must have previously been processed with the `calcBearing` function.

This function calculates the turn angle between two successive movement vectors. If the organism has not moved for a period of time but begins moving again, the function calculates the turn angle between the last movement the organism made and its current move.

If the data will be aggregated, it is recommended to aggregate the data before running this function.

Value

A list of data frames that each contain a column for turn angle.

Examples

```
# Provide a data frame that includes a column with bearing data

 servosphere <- list(data.frame(id = rep(1, 200),
                                stimulus = rep(c(0, 1), each = 100),
                                dT = sample(8:12, 200, replace = TRUE),
                                dx = runif(200, 0, 5),
                                dy = runif(200, 0, 5),
                                treatment = rep("a", 200),
                                date = rep("2032018", 200)),
                     data.frame(id = rep(2, 200),
                                stimulus = rep(c(0, 1), each = 100),
                                dT = sample(8:12, 200, replace = TRUE),
                                dx = runif(200, 0, 5),
                                dy = runif(200, 0, 5),
                                treatment = rep("b", 200),
                                date = rep("2032018", 200)))
servosphere <- calcXY(servosphere)
servosphere <- calcBearing(servosphere)
servosphere <- calcTurnAngle(servosphere)
```

`calcTurnVelocity` *Calculate turn velocity*

Description

Calculate the turning velocity in degrees per second between two moves

Usage

```
calcTurnVelocity(list)
```

Arguments

`list`	A list of data frames, where each data frame has a column for turn angle.

Details

For this function to work, the data must have previously been processed with the `calcTurnAngle` function.

This function calculates the turning velocity between two consecutive moves. The units for turn velocity will be degrees per second.

If the data will be aggregated, it is recommended to aggregate the data before running this function.

Value

A list of data frames that each contain a column for turn velocity.

Examples

```
# Provide data previously processed by the calcTurnAngle function

 servosphere <- list(data.frame(id = rep(1, 200),
                                stimulus = rep(c(0, 1), each = 100),
                                dT = sample(8:12, 200, replace = TRUE),
                                dx = runif(200, 0, 5),
                                dy = runif(200, 0, 5),
                                treatment = rep("a", 200),
                                date = rep("2032018", 200)),
                     data.frame(id = rep(2, 200),
                                stimulus = rep(c(0, 1), each = 100),
                                dT = sample(8:12, 200, replace = TRUE),
                                dx = runif(200, 0, 5),
                                dy = runif(200, 0, 5),
                                treatment = rep("b", 200),
                                date = rep("2032018", 200)))
servosphere <- calcXY(servosphere)
servosphere <- calcBearing(servosphere)
servosphere <- calcTurnAngle(servosphere)
servosphere <- calcTurnVelocity(servosphere)
```

calcVelocity *Calculate velocity*

Description

Calculate the average velocity between two location recordings

Usage

```
calcVelocity(list)
```

Arguments

`list`	A list of data frames, where each data frame has a column for dT, dx, and dy.

Details

This function calculates the velocity of the organism on the servosphere between two location recordings. The units for velocity will be distance per second, where distance is the units of distance used by the software in recording the movement of the organism. For example, if the software recorded distance in centimeters, the units for velocity will be centimeters per second.

If the data will be aggregated, it is recommended to aggregate the data before running this function.

Value

A list of data frames that each contain a column for velocity.

Examples

```
 servosphere <- list(data.frame(id = rep(1, 200),
                                stimulus = rep(c(0, 1), each = 100),
                                dT = sample(8:12, 200, replace = TRUE),
                                dx = runif(200, 0, 5),
                                dy = runif(200, 0, 5),
                                treatment = rep("a", 200),
                                date = rep("2032018", 200)),
                     data.frame(id = rep(2, 200),
                                stimulus = rep(c(0, 1), each = 100),
                                dT = sample(8:12, 200, replace = TRUE),
                                dx = runif(200, 0, 5),
                                dy = runif(200, 0, 5),
                                treatment = rep("b", 200),
                                date = rep("2032018", 200)))

servosphere <- calcVelocity(servosphere)
```

calcXY *Calculate (x, y) coordinates*

Description

Calculates (x,y) coordinates from dx and dy values

Usage

```
calcXY(list)
```

Arguments

`list`	A list of data frame objects with columns dx and dy.

Details

Use the dx and dy columns in the servosphere data frames to calculate the (x, y) coordinate for each position recording. If the data will be aggregated, it is recommended to aggregate the data before running this function.

Value

Converts dx and dy values to (x, y) coordinates.

Examples

```
 servosphere <- list(data.frame(id = rep(1, 200),
                                stimulus = rep(c(0, 1), each = 100),
                                dT = sample(8:12, 200, replace = TRUE),
                                dx = runif(200, 0, 5),
                                dy = runif(200, 0, 5),
                                treatment = rep("a", 200),
                                date = rep("2032018", 200)),
                     data.frame(id = rep(2, 200),
                                stimulus = rep(c(0, 1), each = 100),
                                dT = sample(8:12, 200, replace = TRUE),
                                dx = runif(200, 0, 5),
                                dy = runif(200, 0, 5),
                                treatment = rep("b", 200),
                                date = rep("2032018", 200)))
servosphere <- calcXY(servosphere)
```

cleanNames *Set column names for list of data frames*

Description

Take a list of data frames and change the column names

Usage

```
cleanNames(list, colnames)
```

Arguments

`list`	A list of data frame objects.
`colnames`	A vector of strings holding the names for the columns.

Details

Once a list of data frames is generated with the getFiles function, use this function to set up the column names for all the data frames in the list.

Other functions in this package require that the change in time and position columns are named "dT", "dx", and "dy", respectively.

Value

Returns the list of data frames provided with the column names modified based on the provided vector `colnames`.

Examples

```
colnames <- c("stimulus", "dT", "dx", "dy")
servosphere <- cleanNames(servosphere, colnames)
```

getFiles — *Return a list of data frames generated from the servosphere*

Description

Import all the .csv files in a single directory and convert them to a list of data frames

Usage

```
getFiles(path, pattern, full.names = TRUE)
```

Arguments

`path`	A string for the file path.
`pattern`	A string with a unique pattern to look for in file names.
`full.names`	Return the full file path when TRUE or the file name when FALSE.

Details

When using the servosphere, each trial produces a .csv file that contains the data from that trial. This data should include a column for time (dT), position (dX, dY), and the stimulus. All the data files the user wishes to analyze should be in the same directory. They should also have a common naming convention to facilitate the identification of these files (i.e. "x_servosphere.csv", where x might be the date the trial was run or the unique trial ID).

Other functions in this package require that all the .csv files from the servosphere that the user wishes to analyze are contained in a list, where each item in the list is a data.table created from each .csv file.

This function makes use of the data.table package to read in the .csv files, as it is currently the fastest way to bring in such files. This function is noticeably slower when alternative csv reading functions are used and when the number of .csv files is large.

Value

A list where each item is a data.table

Examples

```
servosphere <- getFiles("./extdata/", pattern = "_servosphere.csv")
```

mergeTrialInfo *Merge trial id information*

Description

This function merges trial id information with the servosphere data.

Usage

```
mergeTrialInfo(list, trial.data, col.names, stimulus.keep,
  stimulus.split = FALSE)
```

Arguments

`list`	The list of servosphere output data.
`trial.data`	The data frame containing your trial id information. This must contain an identifier column titled 'id' and if the data are to be split by stimuli, an additional identifier column 'id_stim'. See the description for more details.
`col.names`	A string vector containing the names of the columns you want to transfer to your servosphere output data. The trial.data data frame may have columns not needed for the analysis, so the function asks the user to be explicit about which columns to retain.
`stimulus.keep`	An integer vector containing the stimuli numbers to retained in the data and split the data frames by. Omitted stimuli values will be discarded.
`stimulus.split`	A logical value indicating whether the data frames should be split by stimulus. Defaults to 'FALSE'. If 'TRUE', be sure to include a 'id_stim' column to give each trial/stimulus combination a unique ID.

Details

Users of the servosphere will need a separate data frame with trial id information in a column titled `id`. This should contain a unique identifier and any other relevant experimental information, such as treatments applied, date, time of day, etc. Make sure the rows in your trial id data frame are ordered in the same order as the list of data frames of your servosphere output.

Researchers may wish to compare data before and after some stimulus has been applied and this function allows the user to split their data into separate data frames based on different values of the stimulus variable to facilitate these comparison. If the data frames should be split by stimulus, the trial record data frame should contain a column `id_stim` that lists the id number of the trial, followed by an underscore, followed by each value of the stimulus variable retained. If the data should be split by stimulus, the rows of the trial id data frame should be ordered first by `id` in the same order as their data is stored within the list. Once ordered by `id` the trial data data frame should be further ordered within an `id` by `stimulus` (i.e. id_stim 1_1 should come before id_stim 1_2).

Data recorded during a particular stimuli may also be discarded if it is not required for analysis. For example, recordings may begin with a five minute adjustment period and the data associated with that period may not be used for analysis. The stimulus recorded by the software during that five minute adjustment period can be discarded by omitting that stimulus number from the `stimulus.keep` argument.

This function will also append an item to your list of data frames that contains the relevant column names to be retained in future manipulations of the data.

Value

Returns the list of data frames provided which have been merged with additional relevant trial information.

Examples

```
 servosphere <- list(data.frame(id = rep(1, 200),
                                stimulus = rep(c(0, 1), each = 100),
                                dT = sample(8:12, 200, replace = TRUE),
                                dx = runif(200, 0, 5),
                                dy = runif(200, 0, 5),
                                treatment = rep("a", 200),
                                date = rep("2032018", 200)),
                     data.frame(id = rep(2, 200),
                                stimulus = rep(c(0, 1), each = 100),
                                dT = sample(8:12, 200, replace = TRUE),
                                dx = runif(200, 0, 5),
                                dy = runif(200, 0, 5),
                                treatment = rep("b", 200),
                                date = rep("2032018", 200)))
 trial_record <- data.frame(id = c(1, 2),
                            treatment = c("a", "b"),
                            date = c("2032018", "2042018"),
                            time = c("13:30", "07:30"))
 trial_id_split <- data.frame(id = c(1, 2, 1, 2),
                              stimulus = c(1, 1, 2, 2),
                              treatment = c("a", "b", "a", "b"),
                              date = rep(c("2032018", "2042018"), times = 2),
                              time = rep(c("13:30", "07:30"), times = 2),
                              id_stim = c("1_1", "2_1", "1_2", "2_2"))
# Merge the columns id, treatment, and date from the trial_record data frame
# with all the data frames in our list servosphere.

 merged_servosphere <- mergeTrialInfo(servosphere,
   trial_record,
   col.names = c("id", "treatment"),
   stimulus.keep = c(0, 1))

# Repeat of the merger above without retaining the id column while
# also splitting the data provided into separate data frames based on the
# different stimuli recorded, keeping only data associated with stimuli 1 and
# 2. Splitting based on stimulus requires a column in the trial.data data
# frame titled id_stim.

 merged_servosphere <- mergeTrialInfo(servosphere,
     trial_id_split,
     col.names = c("id", "treatment"),
     stimulus.split = TRUE,
     stimulus.keep = c(0, 1))
```

servosphere *Example servosphere data*

Description

A list containing two data frames of example data output from the software used with a servosphere.This data has not been cleaned in anyway.

Usage

```
servosphere
```

Format

A list of two data frames:

X.cState stimulus designator

dT..ms. Change in time, in miliseconds

dx..cm. Distance moved in X direction during dT, in centimeters

dY..cm. Distance moved in Y direction during dT, in centimeters

servosphereR *servosphereR: A package to facilitate cleaning and analyzing data produced by a servosphere.*

Description

A servosphere, or locomotory compensator, is a device that can be used to study the movement of insects or other small critters. A servosphere consists of a ball that rests on top of motors. A camera pointed at the top of the sphere watches the insect as it moves and reports position changes to the motors. As the insect moves, the motors rotate to keep the insect positioned at the top of the sphere.

`summaryAvgBearing` *Calculate average bearing*

Description

Calculate the average bearing, or direction of movement, for a movement path.

Usage

```
summaryAvgBearing(list, summary.df = NA)
```

Arguments

`list`	A list of data frames, each of which has a column for bearing.
`summary.df`	The summary data frame containing a column recording the bearing of each recorded movement.

Details

The bearing is the direction of movement and and falls between 0 and 360. Outdoors degree measures of 0/360 will typically correspond to due north. For servosphere data, 0/360 will correspond to the movement in the direction of the positive y-axis.

Value

A list of two named vectors. The first named vector contains the average bearing calculated for each movement path. The second named vector contains the rho, a measure of concentration for the average bearing.

`summaryAvgVelocity` *Calculate average velocity*

Description

Calculate the average velocity for a movement path.

Usage

```
summaryAvgVelocity(list, summary.df = NA)
```

Arguments

`list`	A list of data frames, each of which has a column for velocity.
`summary.df`	The data frame object within which you are storing path summary variables. The default is NA if you do not currently have a summary data frame object started. When set to `NA` the function will create a new summary data frame. When an object is provided, the function will merge the summary data frame with the new data.

Details

Calculate the average velocity for a movement path. The units on velocity are equal to the distance units used to record the data per second.

Value

The inputed summary data frame or a new data frame if summary.df is NA

Examples

```
# If a summary data frame has not been started

 servosphere <- list(data.frame(id = rep(1, 200),
                                stimulus = rep(c(0, 1), each = 100),
                                dT = sample(8:12, 200, replace = TRUE),
                                dx = runif(200, 0, 5),
                                dy = runif(200, 0, 5),
                                treatment = rep("a", 200),
                                date = rep("2032018", 200)),
                     data.frame(id = rep(2, 200),
                                stimulus = rep(c(0, 1), each = 100),
                                dT = sample(8:12, 200, replace = TRUE),
                                dx = runif(200, 0, 5),
                                dy = runif(200, 0, 5),
                                treatment = rep("b", 200),
                                date = rep("2032018", 200)))
servosphere <- calcVelocity(servosphere)

summary_df <- summaryAvgVelocity(servosphere, summary.df = NA)
# If a summary data frame has been started

summary_df <- data.frame(id = c(1, 2),
                         treatment = c("a", "b"),
                         date = c("2032018", "2042018"),
                         stimulus = c(0, 0))

summary_df <- summaryAvgVelocity(servosphere, summary.df = summary_df)
```

summaryNetDisplacement *Calculate net displacement*

Description

Calculate the net displacement for a path taken by an organism

Usage

```
summaryNetDisplacement(list, summary.df = NA)
```

Arguments

`list` A list of data frames, each of which has a column representing dx and dy.

`summary.df` The data frame object within which you are storing path summary variables. The default is NA if you do not currently have a summary data frame object started. When set to NA the function will create a new summary data frame. When an object is provided, the function will merge the summary data frame with the new data.

Details

Net displacement is the distance between the start of a path and the end of a path. This function calculates the net displacement for a path recorded from the servosphere.

Value

A named vector of numbers where each number corresponds to the net displacement of a movement path. The numbers are ordered and named as they are in the data frames list.

`summaryStops` *Summarize the number and length of stops*

Description

Caulculate how many times an insect stopped walking during its recorded movement and the average length of those stops.

Usage

```
summaryStops(list, summary.df = NA, stop.threshold = 0)
```

Arguments

`list` A list of data frames, each of which must have a column recording the velocity variable.

`summary.df` The data frame object within which you are storing path summary variables. The default is NA if you do not currently have a summary data frame object started. When set to `NA` the function will create a new summary data frame. When an object is provided, the function will merge the summary data frame with the new data.

`stop.threshold` The velocity below which the insect is considered to be stopped. The default is 0, but should be adjusted based on observations of the insect and the recording equipment.

Details

This function evaluates a path for the total number of stops made during the recording, as well as the average length of those stops. The function requires the user to provide a movement threshold, below which the insect is considered to be stopped. The servosphere may record small movements of insect appendages or side to side motion as movement but typically this will be much slower than actual movement. Estimates of this threshold speed can be obtained by comparing recordings of the data to videos of the actual movement or in person observation.

`summaryTortuosity` *Calculate tortuosity*

Description

Calculate the tortuosity, or straightness, of a movement path

Usage

```
summaryTortuosity(summary.df, total.distance, net.displacement,
  inverse = FALSE)
```

Arguments

`summary.df` The summary data frame containing total distance and net displacement for all movement paths

`total.distance` The unquoted variable name in a data frame containing the total distance for all movement paths

`net.displacement` The unquoted variable name in a data frame containing the net displacement for all movement paths.

`inverse` Defaults to `FALSE`. When set to `FALSE`, this function calculates tortuosity as net displacement divided by total distance. Setting inverse to `TRUE` causes the function to calculate tortuosity as total distance divided by net displacement.

Details

To use this function, a summary data frame must already exist containing a column for total distance and net displacement (in other words, your data should have been processed by `summaryTotalDistance` and `summaryNetDisplacement`.

Tortuosity is a measure of how straight a path is. There are different methods for calculating path straightness. This function calculates tortuosity as the quotient of net displacement and total distance by default. The quotient can be reversed by setting inverse to `TRUE`.

Value

The inputed data frame of numbers where each number corresponds to the tortuosity of a movement path. The numbers are ordered and named as they are in the data frames list.

Examples

```
# Calculate tortuosity as the ratio of net displacement to total distance

summary_df <- data.frame(id = c(1, 2),
                         treatment = c("a", "b"),
                         date = c("2032018", "2042018"),
                         stimulus = c(0, 0),
                         total_distance = runif(2, 11, 20),
                         net_displacement = runif(2, 5, 10))

summary_df <- summaryTortuosity(summary.df = summary_df,
  total.distance = total_distance,
  net.displacement = net_displacement)

# Calculate tortuosity as the ratio of total distance to net displacement
# (the opposite of the previous example)

summary_df <- summaryTortuosity(summary.df = summary_df,
  total.distance = total_distance,
  net.displacement = net_displacement,
  inverse = TRUE)
```

`summaryTotalDistance` *Calculate total path distance*

Description

Calculate the total distance moved by an organism on the servosphere

Usage

```
summaryTotalDistance(list, summary.df = NA)
```

Arguments

`list` A list of data frames, each of which has a column recording the distance moved during each recording period.

`summary.df` The data frame object within which you are storing path summary variables. The default is NA if you do not currently have a summary data frame object started. When set to NA the function will create a new summary data frame. When an object is provided, the function will merge the summary data frame with the new data.

Details

Determine the total distance of a path taken by an organism on a servosphere.

Value

A named vector of numbers where each number corresponds to the total distance moved by an organism represented in the list of data frames. The numbers are ordered and named as they are in the data frames list.

Examples

```
# If a summary data frame has not been started

 servosphere <- list(data.frame(id = rep(1, 200),
                                stimulus = rep(c(0, 1), each = 100),
                                dT = sample(8:12, 200, replace = TRUE),
                                dx = runif(200, 0, 5),
                                dy = runif(200, 0, 5),
                                treatment = rep(”a”, 200),
                                date = rep(”2032018”, 200)),
                     data.frame(id = rep(2, 200),
                                stimulus = rep(c(0, 1), each = 100),
                                dT = sample(8:12, 200, replace = TRUE),
                                dx = runif(200, 0, 5),
                                dy = runif(200, 0, 5),
                                treatment = rep(”b”, 200),
                                date = rep(”2032018”, 200)))

servosphere <- calcDistance(servosphere)

summary_df <- summaryTotalDistance(servosphere, summary.df = NA)

# If a summary data frame has been started

summary_df <- data.frame(id = c(1, 2),
                         treatment = c(”a”, ”b”),
                         date = c(”2032018”, ”2042018”),
                         stimulus = c(0, 0))

summary_df <- summaryTotalDistance(servosphere, summary.df = summary_df)
```

Index