Package 'TrackMateR'

August 9, 2022

Type Package

Title Working with TrackMate outputs in R

Version 0.1.0

Author Stephen Royle

Maintainer quantixed `<admin@quantixed.org>`

Description TrackMate, a plugin for ImageJ/Fiji, is a popular single-particle tracking solution.
Building on the trackR package by Julien Godet, the aim is to import Track-
Mate data into R for further analysis and visualization.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.0

Depends R (>= 2.10)

Imports doParallel,
dplyr,
foreach,
ggplot2,
patchwork,
reshape2,
XML,
zoo

R topics documented:

calculateAlpha	*Calculate alpha (relationship between MSD and normal diffusion)*

Description

Normal diffusion is alpha = 1. Subdiffusion is alpha < 1 and superdiffusion is alpha > 1. Input is a data matrix of msd curves. Output is mean of log2(alpha), one value for each trace.

Usage

```
calculateAlpha(alphaMat, tstep)
```

Arguments

alphaMat	matrix of msd curves, each col is a track, each row is time lag (will contain NAs)
tstep	variable. Time step in seconds

Value

numeric vector

calculateJD	*Calculate Jump Distance (JD)*

Description

Calculation of the JD of multiple tracks. Calculation is equivalent to a single time lag point on the ensemble MSD curve, typically represented as a histogram Input is a data frame of tracks imported using readTrackMateXML() The default time step is one frame - which is the equivalent to the plot generated to show displacement versus time.

Usage

```
calculateJD(dataList, deltaT = 1)
```

Arguments

dataList	list of data frame (must include at a minimum - trace (track ID), x, y and t (in real coords)) and calibration
deltaT	integer to represent the multiple of frames that are to be analysed

Value

a list of data frame of jump distances, NAs removed; and a numeric variable (jumptime)

Examples

```
xmlPath <- "~/Desktop/FakeTracks.xml"
tmObj <- readTrackMateXML(XMLpath = xmlPath)
tmObj <- correctTrackMateData(tmObj, xyscalar = 0.04)
jdObj <- calculateJD(dataList = tmObj, deltaT = 2)
```

---

calculateMSD   *Calculate Mean Squared Displacement (MSD)*

Description

Calculation of the MSD of multiple tracks. There are two methods for everaging MSD data from multiple tracks: ensemble = for each time lag average all squared displacements from all tracks time-averaged = find MSD for each track and then generate the average MSD from these curves The MSD curves will be identical if all tracks are the same length, and diverge if not. Standard deviation will be large for ensemble and smaller for time-averaged data. Input is a data frame of tracks imported using readTrackMateXML()

Usage

```
calculateMSD(df, method = "timeaveraged", N = 4, short = 0)
```

Arguments

df	data frame must include at a minimum - trace (track ID), x, y and t (in real coords)
method	string. Either "ensemble" or "timeaveraged" (default)
N	numeric variable for MSD. dt should be up to 1/N of number of data points (4 recommended)
short	numeric variable for the shortest number of points we will analyse. Note, this uses the number of frames from start, not number of points in track, i.e. a track with <short points and many gaps will remain

Value

list of a data frame and a vector

Examples

```
xmlPath <- "~/Desktop/FakeTracks.xml"
tmObj <- readTrackMateXML(XMLpath = xmlPath)
tmObj <- correctTrackMateData(tmObj, xyscalar = 0.04)
tmDF <-  tmObj[[1]]
msdObj <- calculateMSD(df = tmDF, method = "ensemble", N = 3, short = 8)
```

---

calculateTrackDensity   *Calculate density of tracks*

Description

Calculate for each track, using its starting frame, what is the relative density of tracks. We use a search radius to find how many tracks in the starting frame are neighbours of the track. The number of neighbours is normalised to the search circle, so that a track in the corner of the image with 2 neighbours has a density of 8. Code for calculating search area (intersection between search circle and the image border) is taken from https://petrelharp.github.io/circle_rectangle_intersection/circle_rectangle_intersectio

Usage

```
calculateTrackDensity(dataList, radius = 1)
```

Arguments

dataList	list of a data frame (must include at a minimum - trace (track ID), x, y and frame (in real coords)) and a calibration data frame
radius	numeric variable for search radius (in spatial units of the data)

Value

data frame

Examples

```
xmlPath <- "~/Desktop/FakeTracks.xml"
tmObj <- readTrackMateXML(XMLpath = xmlPath)
tmObj <- correctTrackMateData(dataList = tmObj, xyscalar = 0.04)
tdDF <- calculateTrackDensity(dataList = tmObj, radius = 2)
```

---

correctTrackMateData   *Correct distance and time of imported TrackMate data.*

Description

If the TrackMate data is in pixels and/or frames, the data frame can be converted with this function.

Usage

```
correctTrackMateData(
  dataList,
  xyscalar = 1,
  tscalar = 1,
  xyunit = NULL,
  tunit = NULL
)
```

Arguments

dataList	a list of a data frame (of track data) and a calibration data frame (from TrackMate XML)
xyscalar	numeric multiplier to correct pixel size of original movie. Assumes isotropic scaling, i.e. pixel height = pixel width
tscalar	numeric multiplier to correct frame interval of original movie. Frame interval of tracked data.
xyunit	string to describe spatial units
tunit	string to describe temporal unit

Value

list of two data frames

Examples

```
xmlPath <- "~/Desktop/FakeTracks.xml"
tmObj <- readTrackMateXML(XMLpath = xmlPath)
# in the case where pixel size is 0.03 um and original data is 1 pixel, xyscalar = 0.03
tmObj <- correctTrackMateData(dataList = tmObj, xyscalar = 0.03)
```

---

find_distances	*Find distance between xy coordinate and a series of other xy coordinates*

Description

Find distance between xy coordinate and a series of other xy coordinates

Usage

```
find_distances(xx, yy, df)
```

Arguments

xx	x coord of point for comparison
yy	y coord of point for comparison
df	data frame containing x and y columns for other points

Value

numeric vector of distances

find_td_A1   *Track Density - Find area A1*

Description

Track Density - Find area A1

Usage

```
find_td_A1(x, r)
```

Arguments

x	value
r	radius

Value

numeric variable

---

find_td_A2   *Track Density - Find area A2*

Description

Track Density - Find area A2

Usage

```
find_td_A2(x, y, r)
```

Arguments

x	value
y	value
r	radius

Value

numeric variable

---

find_td_area | *Track Density - Find search area*

Description

Find the area of the intersection of the circle centered at xy with radius r and the radius with vertical sides at a and horizontal sides at b. xy, a, and b must be vectors of length 2, and xy must lie within the rectangle.

Usage

```
find_td_area(r, xy, a, b)
```

Arguments

r	radius of search circle
xy	numeric vector (length 2)
a	numeric vector (length 2)
b	numeric vector (length 2)

Value

numeric variable

Examples

```
find_td_area(r=2, xy=c(4, 4), a=c(0, 8), b=c(0, 5))
```

---

fittingJD | *Fitting jump distance (JD) data*

Description

Jump Distances have been calculated for a given time lag. They can be described by fitting curves to the data, either using a histogram or cumulative probability density function. Firtting to a histogram is sensitive to binning parameters and ECDF performs better for general use. The idea behind this analysis is given in: - Weimann et al. (2013) A quantitative comparison of single-dye tracking analysis tools using Monte Carlo simulations. PloS One 8, e64287. - Menssen & Mani (2019) A Jump-Distance-Based Parameter Inference Scheme for Particulate Trajectories, Biophysical Journal, 117: 1, 143-156. The bulk of this code is taken from trackR by JuG

Usage

```
fittingJD(
  df,
  mode = "ECDF",
  nPop = 1,
  init,
  units = c("um", "s"),
  timeRes = 1,
  breaks = 100
)
```

Arguments

df	data frame with a column named jump of jump distances
mode	string indicated ECDF (default) or hist (histogram)
nPop	number of populations of diffusing species (1, 2 or 3)
init	initialisation parameters for the nls fit for example list(D2 = 200, D1 = 0.1) or list(D2 = 0.01, D1=0.1, D3=10, D4=100)
units	character vector to describe units (defaults are um, micrometres and s, seconds)
timeRes	time resolution per unit of jump. Frame interval is 0.5 s and jump interval is two steps, timeRes = 1.
breaks	number of bins for histogram. With ECDF breaks can be high e.g. 100, for mode = "hist" they should be low, perhaps 30.

Value

ggplot

Examples

```
xmlPath <- "~/Desktop/FakeTracks.xml"
tmObj <- readTrackMateXML(XMLpath = xmlPath)
tmObj <- correctTrackMateData(tmObj, xyscalar = 0.04)
jdObj <- calculateJD(dataList = tmObj, deltaT = 2)
jdDF <- jdObj[[1]]
fittingJD(df = jdDF, mode = "ECDF", nPop = 2, breaks = 100, timeRes = 0.06)
```

---

makeSummaryReport   *Make Summary Report*

Description

Generate several plots to visualise TrackMate data and generate a report. The use of um is because ggsave does not currently save unicode to PDF reliably. The code is switchable to accommodate making a "report" (one dataset) or a "summary" (several related datasets combined)

Usage

```
makeSummaryReport(
  tmList,
  msdList,
  jumpList,
  tddf,
  titleStr = "",
  subStr = "",
  auto = FALSE,
  summary = FALSE
)
```

Arguments

tmList	list of trackmate data and calibration
msdList	MSD summary and alpha list = output from calculateMSD()
jumpList	list of a data frame of jump data and a variable to be passed to timeRes
tddf	data frame of track density data
titleStr	string used as the title for the report
subStr	string used as the subtitle for the report
auto	boolean which selects for returning the patchwork report or a list of the patchwork report and a data frame of summary
summary	boolean which selects for summary

Value

patchwork ggplot or a list of patchwork ggplot and data frame of summary data

Examples

```
xmlPath <- "~/Desktop/FakeTracks.xml"
tmObj <- readTrackMateXML(XMLpath = xmlPath)
tmObj <- correctTrackMateData(tmObj, xyscalar = 0.04)
tmDF <- tmObj[[1]]
calibrationDF <- tmObj[[2]]
msdObj <- calculateMSD(df = tmDF, method = "ensemble", N = 3, short = 8)
jdObj <- calculateJD(dataList = tmObj, deltaT = 1)
tdDF <- calculateTrackDensity(dataList = tmObj, radius = 1.5)
fileName <- tools::file_path_sans_ext(basename(xmlPath))
reportObj <- makeSummaryReport(tmList = tmObj, msdList = msdObj, jumpList = jdObj, tddf = tdDF,
titleStr = "Report", subStr = fileName, auto = TRUE)
```

---

plotMSD	*Make a plot of MSD data*

Description

Generate a plot of MSD over a series of increasing time lags. Input is the output from CalculateMSD(), so the plot will display the ensemble or time-averaged MSD (whatever was requested) A fit to the first four points is displayed to evaluate alpha. Diffusion coefficient from this fit is displayed top-left.

Usage

```
plotMSD(
  df,
  units = c("um", "s"),
  bars = FALSE,
  xlog = FALSE,
  ylog = FALSE,
  auto = FALSE
)
```

Arguments

df   MSD summary = output from calculateMSD()

units   character vector to describe units (defaults are um, micrometres and s, seconds)

bars   boolean to request error bars (1 x SD)

xlog   boolean to request log10 x axis

ylog   boolean to request log10 y axis

auto   boolean to request plot only, TRUE gives plot and D as a list

Value

ggplot or ggplot and variable

Examples

```
xmlPath <- "~/Desktop/FakeTracks.xml"
datalist <- readTrackMateXML(XMLpath = xmlPath)
data <-  datalist[[1]]
# use the ensemble method and only look at tracks with more than 8 points
msdobj <- calculateMSD(df = data, method = "ensemble", N = 3, short = 8)
msddf <- msdobj[[1]]
plotMSD(msddf, bars = FALSE)
```

---

plotNMSD   *Plot several (n) MSD curves*

Description

Generate a plot of several MSD curves together with a summary curve.

Usage

```
plotNMSD(df)
```

Arguments

df   dataframe of MSD summary data from multiple datasets (labelled by dataid)

Value

ggplot

---

plot_tm_allTracks   *Make a plot of all tracks*

Description

Make a plot of all tracks

Usage

```
plot_tm_allTracks(
  input,
  summary = FALSE,
  xstr = "",
  ystr = "",
  alphaLevel = 0.5
)
```

Arguments

input	either a data frame of TrackMate data or list of TrackMate data frame and calibration data frame
summary	boolean to specify if plot is of one dataset or several related datasets
xstr	string to label x-axis
ystr	string to label y-axis
alphaLevel	numeric variable to set alpha for the plot

Value

ggplot

---

plot_tm_alpha   *Make a histogram of alpha values*

Description

Make a histogram of alpha values

Usage

```
plot_tm_alpha(
  df,
  median_alpha = NULL,
  xstr = "alpha (log2)",
  ystr = "Frequency"
)
```

Arguments

`df`	data frame of alpha values
`median_alpha`	variable for adding label to plot
`xstr`	string to label x-axis
`ystr`	string to label y-axis

Value

ggplot

`plot_tm_cumdistOverTime`

Make a plot of cumulative distance over time

Description

Make a plot of cumulative distance over time

Usage

```
plot_tm_cumdistOverTime(
  input,
  summary = FALSE,
  xstr = NULL,
  ystr = NULL,
  alphaLevel = 0.1
)
```

Arguments

`input`	either a data frame of TrackMate data or list of TrackMate data frame and calibration data frame
`summary`	boolean to specify if plot is of one dataset or several related datasets
`xstr`	string to label x-axis
`ystr`	string to label y-axis
`alphaLevel`	numeric variable to set alpha for the plot

Value

ggplot

plot_tm_displacementHist

Make a histogram of displacements

Description

Make a histogram of displacements

Usage

```
plot_tm_displacementHist(
  input,
  summary = FALSE,
  xstr = NULL,
  ystr = NULL,
  auto = FALSE
)
```

Arguments

input	either a data frame of TrackMate data or list of TrackMate data frame and calibration data frame
summary	boolean to specify if plot is of one dataset or several related datasets
xstr	string to label x-axis
ystr	string to label y-axis
auto	boolean to switch between returning a ggplot and a list of ggplot and variable

Value

ggplot or list of ggplot and variable

plot_tm_displacementOverTime

Make a plot of displacement over time

Description

Make a plot of displacement over time

Usage

```
plot_tm_displacementOverTime(input, summary = FALSE, xstr = NULL, ystr = NULL)
```

Arguments

input	either a data frame of TrackMate data or list of TrackMate data frame and calibration data frame
summary	boolean to specify if plot is of one dataset or several related datasets
xstr	string to label x-axis
ystr	string to label y-axis

Value

ggplot

---

plot_tm_neighbours	*Make a histogram of track density (number of neighbours)*

Description

Make a histogram of track density (number of neighbours)

Usage

```
plot_tm_neighbours(df, auto = FALSE)
```

Arguments

df	data frame of TrackMate data
auto	boolean to switch between returning a ggplot and a list of ggplot and variable

Value

ggplot or list of ggplot and variable

---

plot_tm_speed	*Make a histogram of average speed*

Description

Make a histogram of average speed

Usage

```
plot_tm_speed(input, summary = FALSE, xstr = NULL, ystr = NULL, auto = FALSE)
```

Arguments

input	either a data frame of TrackMate data or list of TrackMate data frame and calibration data frame
summary	boolean to specify if plot is of one dataset or several related datasets
xstr	string to label x-axis
ystr	string to label y-axis
auto	boolean to switch between returning a ggplot and a list of ggplot and variable

Value

ggplot or list of ggplot and variable

---

readTrackMateXML *Read TrackMate XML output files.*

Description

Produces a data frame of all spots from filtered tracks, ordered by track number. A warning is generated if the scaling is in pixels rather than real units.

Usage

```
readTrackMateXML(XMLpath)
```

Arguments

XMLpath   path to the xml file

Value

list of two data frames

Examples

```
xmlPath <- "~/Desktop/FakeTracks.xml"
tmObj <- readTrackMateXML(XMLpath = xmlPath)
# get the track data in a data frame
tmDF <-  tmObj[[1]]
# get the calibration data in a data frame
calibrationDF <- tmObj[[2]]
```

Index