

# Package ‘sojourner’

June 30, 2022

**Title** Statistical analysis of single molecule trajectories

**Version** 1.10.0

**Date** 2015-10-15

**Description** Single molecule tracking has evolved as a novel new approach complementing genomic sequencing, it reports live biophysical properties of molecules being investigated besides properties relating their coding sequence; here we provided ‘sojourner’ package, to address statistical and bioinformatic needs related to the analysis and comprehension of high throughput single molecule tracking data.

**License** Artistic-2.0

**LazyData** FALSE

**Suggests** BiocStyle, knitr, rmarkdown, RUnit, BiocGenerics

**VignetteBuilder** knitr

**biocViews** Technology, WorkflowStep

**Imports**

ggplot2,dplyr,reshape2,gridExtra,EImage,MASS,R.matlab,Rcpp,fitdistrplus,mclust,minpack.lm,mixtools,mltools,nls2,pl

**NeedsCompilation** no

**RoxygenNote** 7.1.0

**Encoding** UTF-8

**URL** <https://github.com/sheng-liu/sojourner>

**BugReports** <https://github.com/sheng-liu/sojourner/issues>

**git\_url** <https://git.bioconductor.org/packages/sojourner>

**git\_branch** RELEASE\_3\_15

**git\_last\_commit** 077ecb3

**git\_last\_commit\_date** 2022-04-26

**Date/Publication** 2022-06-30

**Author** Sheng Liu [aut],  
Sun Jay Yoo [aut],  
Xiao Na Tang [aut],  
Young Soo Sung [aut],

Carl Wu [aut],  
 Anand Ranjan [ctb],  
 Vu Nguyen [ctb],  
 Sojourner Developer [cre]

**Maintainer** Sojourner Developer <sojourner.developer@outlook.com>

## R topics documented:

combineTrackll . . . . .	2
compareFolder . . . . .	3
compare_RT_CDF . . . . .	4
createTrackll . . . . .	5
Dcoef . . . . .	7
displacementCDF . . . . .	9
dispVariance . . . . .	10
dwelTime . . . . .	11
exportTrackll . . . . .	12
filterTrack . . . . .	13
fitNormDistr . . . . .	14
fitRT . . . . .	16
linkSkippedFrames . . . . .	17
maskTracks . . . . .	18
mergeTracks . . . . .	19
msd . . . . .	20
plotIndividual_Tracks . . . . .	22
plotLocalizations_Density . . . . .	23
plotTrack . . . . .	24
plotTrackOverlay_Dcoef . . . . .	26
sojourner . . . . .	28
sojournerGUI . . . . .	28
squareDisp . . . . .	29
<b>Index</b>	<b>30</b>

---

combineTrackll	<i>combineTrackll</i>
----------------	-----------------------

---

### Description

Combine multiple tracklls into one trackll.

### Usage

```
combineTrackll(trackll,name='combined trackll',merged=TRUE)
```

**Arguments**

trackll	The tracklls to be combined together.
name	a character string given to set the 'names' attribute for the combined trackll
merged	An Logical indicate if the tracklls to combine are merged or not.

**Details**

Combine multiple track lists (tracklls) from multiple folders into one trackll, i.e. combining track information from files in multiple folders (replicates) together as if they are in one folder. The tracklls can be either merged or un-merged.

The name argument sets the 'names' attribute for the combined trackll, which will be used in the same way as the folder names for the original tracklls, e.g., displayed as legend when plotting Dcoef or MSD for the combined trackll.

**Value**

- trackll: combined trackll.

**Examples**

```
# Generate trackll, and process,
# e.g. mask region of interest, merge tracks from multiple files.
folder1=system.file('extdata', 'HSF', package='sojourner')
trackll1=createTrackll(folder1, input=3, cores = 2)
trackll1=maskTracks(folder1, trackll1)
trackll1=mergeTracks(folder1, trackll1)

folder2=system.file('extdata', 'HSF_2', package='sojourner')
trackll2=createTrackll(folder2, input=2, cores = 2)
trackll2=maskTracks(folder2, trackll2)
trackll2=mergeTracks(folder2, trackll2)

# Combine the tracklls together, input trackll names when prompted,
trackll=combineTrackll(trackll=c(trackll1, trackll2), merged=TRUE)
```

---

compareFolder

*compareFolder*

---

**Description**

compare folders with Diatrack output files. merge track files in each folder into one item of a track list. This list can then be fed into other functions for comparison. It keeps folder information as names of the resulting list.

**Usage**

```
compareFolder(folders, input=1, ab.track=FALSE, cores=1)
```

**Arguments**

folders	a vector storing paths to the folders location.'...' indicates multiple (unlimited) folders can be added into the function.
input	Input file type (Diatrack .txt file = 1; Diatrack .mat session file = 2; ImageJ .csv file = 3; SlimFast .txt file = 4).
ab.track	a Logical indicating if absolute coordinates should be used.
cores	Number of cores used for parallel computation. This can be the cores on a workstation, or on a cluster.

**Value**

- trackll A list of tracks, each item of a track list correspond to a folder. This list can then be fed into other functions for comparison.

**Examples**

```
folder1=system.file('extdata','SWR1',package='sojourner')
folder2=system.file('extdata','HTZ1',package='sojourner')
trackll=compareFolder(folders=c(folder1, folder2), input=3)
str(trackll,max.level=1)
```

---

compare\_RT\_CDF

*Compare Residence time/Survival Curve (1-CDF)*

---

**Description**

Compare Residence time/Survival Curve of multiple trackll. Or simply plot the survival curve of one trackll.

**Usage**

```
compare_RT_CDF(trackll=NULL,x.max=30,
filter=c(min=3,max=Inf),t.interval=0.5,output=FALSE)
```

**Arguments**

trackll	trajectory list generated by createTrackll() and processing. if NULL, user will be prompted to enter the trackll name.
x.max	The maximum range of X axis, i.e. time, for the output plot. Default 30 sec.
filter	Filter the tracks by step/frame number (length). Only tracks pass through filter will be selected.
t.interval	time interval for image aquisition. Default 0.5 sec.
output	An Logical indicate if output should be generated. See Values for detail.

## Details

Compare Residence time/Survival Curve of multiple track list (trackll). Or simply plot the survival curve of one trackll. The survival curve/probability is calculated as 1-CDF of the length of tracks/trajectories.

If the acquisition time interval of the tracklls are different, set argument trackll=NULL, users will be prompted to input the number of the track list (trackll) to compare/plot. Then users will be prompted to input the name and acquisition time interval of each trackll. The trackll should be masked and merged. The maximum time range to be plotted can be set using x.max, this will not change the curve/probability, which is determined by all track information in the trackll.

## Value

- csv: 1-CDF of track lengths and time intervals output in .csv format, when output = TRUE.
- Plot: 1-CDF of track lengths of each input trackll will be plotted together in one plot.

## Examples

```
# Generate trackll, and process,
# e.g. mask region of interest, merge tracks from multiple files.
folder1=system.file('extdata','HSF',package='sojourner')
trackll1=createTrackll(folder1,input=3, cores = 2)
trackll1=maskTracks(folder1,trackll1)
trackll1=mergeTracks(folder1,trackll1)

folder2=system.file('extdata','HSF_2',package='sojourner')
trackll2=createTrackll(folder2,input=2, cores = 2)
trackll2=maskTracks(folder2,trackll2)
trackll2=mergeTracks(folder2,trackll2)

# Plot and output the survival curve,
compare_RT_CDF(trackll=c(trackll1,trackll2),x.max=30,
filter=c(min=3,max=Inf),t.interval=0.5,output=FALSE)
```

---

createTrackll

*createTrackll*

---

## Description

take in Diatrack (.txt or .mat), ImageJ Particle Tracker (.csv), SLIMfast (.txt), or u-track (.mat) input from a folder to output a list of track lists.

## Usage

```
createTrackll(folder, interact = FALSE, input = 1, ab.track = FALSE,
cores = 1, frameRecord = TRUE)
```

### Arguments

folder	Full path output file folder (ensure each folder has files of only one input type).
interact	Open interactive menu to choose the desired folder by selecting any file in it and select input type (script will process all files of that type in this folder).
input	Input file type (Diatrack .txt file = 1; Diatrack .mat session file = 2; ImageJ .csv file = 3; SLIMfast .txt file = 4; u-track .mat file = 5).
ab.track	Use absolute coordinates for tracks.
cores	Number of cores used for parallel computation. This can be the cores on a workstation, or on a cluster. Each core will be assigned to read one file when in parallel.
frameRecord	Add a fourth column to the track list after the xyz-coordinates for the frame that coordinate point was found (especially helpful when linking frames). Highly recommended to leave on.

### Details

(Note: When reading only Diatrack .mat session files (input = 2), intensities will be saved after the frame column)

It is highly advised that the frame record option be left on to preserve the most information, especially when linking frames and when using Utrack. If the frame record option is turned on for reading Diatrack .txt files (input = 1), take note that the frame record is artificially created as consecutive frames after the given start frame. Otherwise, all other data types naturally record the frames of every coordinate point.

The pre-censoring of single-frame tracks is dependent on the tracking software. For highest fidelity track data, use Diatrack (.mat) session files. If the initial creation of the trackll does not have a frame record, future exports and imports of the trackll will only preserve the start frames.

If the cores are set to the maximum number of cores available on the system, the script may return a error after processing all the files. This error is due to the requirement of some systems to have one core open for system functions. This error will not affect the trackll output, but to avoid it, one can input one less than the maximum number of cores available.

The naming scheme for the list of track list is as follows:

Track List: [full name of input file]

Track: [Last five characters of the file name].[Start frame].[Length]. [Track].[Index in overall list (will differ from Track # when merging)]

(Note: The last five characters of the file name, excluding the extension, cannot contain '.')

### Value

trackll

### Examples

```
# Designate a folder and then create trackll from .csv data
folder=system.file('extdata','SWR1',package='sojourner')
trackll = createTrackll(folder=folder, input=3)
```

```
# Alternatively, use interact to open file
# browser and select input data type
# trackll <- createTrackll(interact = TRUE)
```

---

Dcoef

*Dcoef*


---

## Description

Calculate diffusion coefficient (Dcoef) for trajectories.

## Usage

```
Dcoef( MSD=NULL, trackll=NULL, dt=6, filter=c(min=7,max=Inf), rsquare=0.8,
       resolution=0.107, binwidth=NULL, method=c("static", "percentage",
       "rolling.window"), plot=FALSE, output=FALSE, t.interval=0.01,
       profile=NULL)
```

## Arguments

MSD	Mean Square Displacement calculated using msd() function. Either MSD or trackll can be passed into Dcoef for calculation of diffusion coefficient.
trackll	Track list output from readDiatrack().
dt	Time intervals. Default 6.
filter	a vector specifies the minimum and max length of trajectories to be analyzed. Take only trajectories that has number of frames greater than ( $\geq$ ) min and less than ( $<$ ) max.
rsquare	rsquare filter on Dcoef results. Default to be 0.8. Set value to 0 if rsquare filter is not desired.
resolution	ratio of pixel to $\mu\text{m}$ .
binwidth	binwidth used for histogram. Default NULL, automatically assign binwidth.
method	"static", uses time lags 2~5 to calculate diffusion coefficient; "percentage", uses (tierd) percentage (default 0.25) of time lags (see Details). "rolling.window", time lags uses for Dcoef follows a rolling window with specified window size (default 4).
plot	A parameter for plotting. Default FALSE, no plot; If TRUE, automatically plots 'histogram' with count information, binwidth can be set through parameter binwidth; as well as 'density' with density/frequency.
output	An Logical indicate if output should be generated. See Values for detail.
t.interval	time interval between frames, default 0.010 s (10ms).
profile	Location of preference file. By default (NULL), it is stored at : system.file("extdata","PREF","profile.csv") User can provide preference file by specifying the location of the file, e.g. profile="/Users/shengliu/Desktop/profile.csv".

## Details

Generic parameters (parameter applied to all methods, such as resolution etc) are set in the function. Method dependent parameters (such as lag.start, lag.end for method = "static"), are stored in profile.csv in PREF folder under extdata. To change preference parameter, can either programably or manually go to folder system.file("extdata","PREF","profile.csv",package="sojourner"), and change the profile.csv.

lag.start: time lag used as start of dt for compute Dcoef. Default 2.

lag.end: Time lag used as end of dt for compute Dcoef. Default 2.

method for calculating Dcoef:

- **static** stabilize the number of time lags used for fitting using time lag 2~ 5 despite the total time lags measured.
- **percentage** "percentage", uses (tierd) percentage (default 0.25) of time lags.

	[,1]	TrackLength	Percentage	TimeLagsForFitting
	[,2]	31~	0.25	2~5-2~7
	[,3]	22~30	0.25	2~5-2~7
	[,4]	15~21	0.4	2~5-2~7
	[,5]	10~15	0.6	2~5-2~7
	[,6]	7~9	1	2~5-2~7

- **rolling.window** time lags uses for Dcoef follows a rolling window with specified window size (default 4).

## Value

- *Dcoef* A list of Dcoef for each file in trackll.
- *PDF* Log.Dcoef histogram fitted with density curve, when plot = TRUE.
- *csv* Dcoef output in csv format, when output = TRUE.

## Examples

```
# compare files
folder=system.file("extdata", 'SWR1', package="sojourner")
trackll = createTrackll(folder=folder, input=3)
MSD=msd(trackll=trackll)
Dcoef(MSD=MSD,method="static",plot=TRUE)

# compare folders
folder1=system.file("extdata", 'SWR1', package="sojourner")
folder2=system.file("extdata", 'HTZ1', package="sojourner")
trackll2=compareFolder(folders=c(folder1, folder2), input=3)
Dcoef(trackll=trackll2,method="percentage",plot=TRUE)
Dcoef(trackll=trackll2,method="rolling.window",plot=TRUE)
```



---

displacementCDF      *displacementCDF*

---

### Description

calculate cumulative distribution function of all displacement for individual trajectories.

### Usage

```
displacementCDF(track11,dt=1,resolution=0.107,plot=FALSE,output=FALSE,
bivar=FALSE)
```

### Arguments

dt	Time intervals.
resolution	ratio of pixel to uM.
bivar	bivar=FALSE, view displacement r as single variable; bivar=TRUE, view x,y as bivariate. Default value F.
track11	a list of track lists.
plot	An logical indicate if plot should be generated. See Values for detail.
output	An logical indicate if output should be generated. See Values for detail.

### Details

The cumulative radial distribution function,  $P(r, i*dt)$ , is the probability of finding the diffusing particle within a radius r from the origin at time lag  $i*dt$ :

$$P(r,i*dt) = 1 - e^{-(r^2/4*D*(i*dt))}$$

the CDF and UniqueDisplacement in the output file is corresponding to P and r in this formula. If intend to generate the CDF plot from the output file, the CDF and UniqueDisplacement is corresponding to the y and x values in the CDF output plot.

### Value

- list of "stepwise.displacement" and "CDF.displacement", A list of stepwise.displacement" and "CDF.displacement". the name of the list is the track folder name.
- Output file, Displacement of individual trajectoreis at specified dt. The output file is for user to plot in other applications. The column "UniqueDisplacement" is the x axis, and column "CDF" is the y axis for a CDF plot. The distribution of "UniqueDisplacement" is the density plot.
- CDF plot, CDF plot of displacement for individual files.

### Examples

```
folder1=system.file("extdata", "SWR1", package="sojourner")
folder2=system.file("extdata", "HTZ1", package="sojourner")
track11=compareFolder(folders=c(folder1, folder2), input=3)
displacementCDF(track11,dt=1,plot=TRUE)
```

---

dispVariance	<i>dispVariance</i>
--------------	---------------------

---

### Description

calculate square displacements for all tracks in a trackll datatype, and return the variances for the displacements of each trajectories.

### Usage

```
dispVariance(trackll, min=7, plot=FALSE, limits=c(), log=FALSE,
output=FALSE)
```

### Arguments

trackll	a list of track lists.
min	minimum points on trajectory, should be at least 3 to work.
plot	default: False, if true, show density plot for variances.
limits	vector of size2, variance cut-off range that one wants to plot. This will not affect the returned result.
log	default: False, if true, apply log10 to variance value for new spread. like limits, this will only affect the plot, not the returned value.
output	if True, generate a csv output for each tracklist files that are in trackll.

### Details

dispVariance applies the squareDisp function to each dataframe containing trajectories. the tracks somehow had to be converted into dataframes although they were expected to be in dataframes in the first place.

Since the tracks of shorter length are filtered out in the process, there is no guarantee that the length of tracklists equal that of the input.

The tracks should have length of at least 3, in order to have a valid displacement variance. If min argument is less than 3, the function will not be executed.

Generally, when plotting, you would want to use only one of limits or log. Although you may use both, using only one of the two would do the job.

### Value

- Variances calculated variance for all tracks in trackll

**Examples**

```

folder=system.file('extdata','SWR1',package='sojourner')
trackll=createTrackll(folder=folder, input=3)

# run dispVariance with default minimum tracklength (min=7)
dispVars = dispVariance(trackll)

# run dispVariance by setting the minimum tracklength to 3
dispVars = dispVariance(trackll, min=3)

# display plot only within certain range
dispVariance(trackll, plot=TRUE, limits = c(0,0.002))

# display plot with log-scale applied
dispVars = dispVariance(trackll, min=3, plot=TRUE, log=TRUE)

# display plot. Could get csv files if output = TRUE
dispVars = dispVariance(trackll, min=3, plot=TRUE, output=FALSE)

```

---

dwellTime

*dwellTime*


---

**Description**

Caclulate dwell time (/residence time) for trajecotries.

**Usage**

```

dwellTime(trackll,t.interval=10,x.scale=c(min=0,max=250),plot=TRUE,
output=FALSE)

```

**Arguments**

t.interval	t.interval time, default = 10ms.
trackll	Track list output from readDiatrack().
x.scale	x-scale min and max range.
plot	An Logical indicate if plot should be generated. If plot = TRUE, the plot data will also be output.
output	An Logical indicate if output should be generated. 1) dwell time of tracks in the track list output to csv file. Each item in the list will have an individual csv file. 2) Plot PDF and plot data will be saved.

**Value**

- dwell time list A list of dwell time for all trajectories, separated by file names of the trajectory file in Diatrack file folder. If combined dwell time is intended, use readDiatrack(folder, merge=TRUE) to generate a single length list, then apply this function.
- PDF dwell time frequency plot in PDF format, when plot = TRUE.
- csv dwell time output in csv format, when output = TRUE.

**Examples**

```
folder=system.file('extdata', 'SWR1', package='sojourner')
trackll=createTrackll(folder=folder, input=3)
dwellTime(trackll, plot=TRUE)
```

---

exportTrackll

*exportTrackll*

---

**Description**

take in a list of track lists (trackll) and export it into row-wise (ImageJ Particle Tracker style) .csv files in the working directory

**Usage**

```
exportTrackll(trackll, cores = 1)
```

**Arguments**

trackll	A list of track lists.
cores	Number of cores used for parallel computation. This can be the cores on a workstation, or on a cluster. Tip: each core will be assigned to read in a file when paralleled.

**Details**

The reason why ImageJ particle Tracker style .csv export was chosen is because it fully preserves track frame data, while maintaining short computation time and easy readability in Excel/etc.

In order to import this .csv export back into a trackll at any point (while preserving all information), select input = 3 in createTrackll.

If the track list does not have a fourth frame record column (not recommended), it will just output the start frame of each track instead and will take noticeably longer.

It is not recommended that exportTrackll be run on merged list of track lists (trackll). Also, ensure that the input trackll is a list of track lists and not just a track list.

The naming scheme for each export is as follows:

[Last five characters of the file name]\_[yy-MM-dd]\_[HH-mm-ss].csv

**Value**

.csv file output

**Examples**

```
folder=system.file('extdata', 'SWR1', package='sojourner')
trackll=createTrackll(folder=folder, input=3)

# Basic function call to exportTrackll into current directory
exportTrackll(trackll)

# Import export save back into a trackll
# Get current working directory
getwd()
trackll.2 <- createTrackll(folder = getwd(), input = 3)
```

---

filterTrack	<i>filterTrack</i>
-------------	--------------------

---

**Description**

methods for filter and trim tracks based on track length.

**Usage**

```
filterTrack(trackll, filter=c(min=7, max=Inf))
trimTrack(trackll, trimmer=c(min=1, max=32))
trackLength(trackll)
```

**Arguments**

trackll	a list of track lists.
filter	range of possible track lengths to keep
trimmer	range of track lengths allowed in output, otherwise trimmed.

**Details**

filterTrack() is used to filter out tracks that has length within a specified range (default 7~Inf). On the other hand, despite the lengths of tracks, trimTrack() is used to trim /cutoff all tracks to a specified range (default 1~32).

**Value**

- trackll filtered or trimmed tracks.
- len list of track lengths.

**Examples**

```

folder=system.file('extdata', 'SWR1', package='sojourner')
trackll=createTrackll(folder=folder, input=3)

trackll.filter=filterTrack(trackll, filter=c(7, Inf))
trackll.trim=trimTrack(trackll, trimmer=c(1, 20))

# see the min and max length of the trackll
# trackLength() is a helper function output track length of trackll
lapply(trackLength(trackll), min)
lapply(trackLength(trackll.filter), min)

lapply(trackLength(trackll), max)
lapply(trackLength(trackll.trim), max)

```

---

fitNormDistr

*fitNormDistr*


---

**Description**

fit normal distributions to diffusion coefficient calculated by Dcoef method and saves seed state as a attribute of the result

**Usage**

```

fitNormDistr(dcoef, components=NULL, log.transform=FALSE, binwidth=NULL,
combine.plot=FALSE, output=FALSE, proportion=NULL, means=NULL,
sd=NULL, constrain=FALSE)

```

**Arguments**

sd	numeric vector with estimates of standard deviation(sigma) values for each component.
constrain	logical indicate if mean and std deviation are set to the given value. This will not work for the unimodal distribution.
dcoef	diffusion coefficient calculated from Dcoef().
components	parameter specifying the number of components to fit. If NULL (default), a components analysis is done to determine the most likely components and this number is then used for subsequent analysis.
log.transform	logical indicate if log10 transformation is needed, default F.
binwidth	binwidth for the combined plot. If NULL (default), will automatic assign binwidth.
combine.plot	Logical indicate if all the plot should be combined into one, with same scale (/same axes breaks), same color theme, and same bin size for comparison.
output	logical indicate if output file should be generated.
proportion	numeric vector with estimates of each component's proportion of the whole data.
means	numeric vector with estimates of mean(mu) values for each component.

## Details

Components analysis uses the likelihood ratio test (LRT) to assess the number of mixture components. Bad Random seed generation may cause normalmixEM to crash. Using another seed will solve the issue.

Note: Ensure that a random number generator seed has been manually set! The seed is stored as an attribute of the returned object of fitNormDistr() and using the same seed makes results repeatable (see examples).

## Value

**proportions** The proportions of mixing components.

**mean** The Means of the components.

**sd** The Standard Deviations (SD) of components if not log transformed; if log transformed, it is then interpreted as Coefficient of Variation (CV).

**loglik** The log likelihood, useful for compare different fitting result, the bigger the better fit.

## Examples

```
# compare folders
folder1=system.file("extdata","SWR1",package="sojourner")
folder2=system.file("extdata","HTZ1",package="sojourner")
track11=compareFolder(folders=c(folder1,folder2), input=3)
MSD=msd(track11=track11)
dcoef=Dcoef(MSD,dt=6,plot=TRUE,output=FALSE)

# set unique seed (use any number)
set.seed(123)

# fit dcoef (function automatically saves seed state as an attribute of
# the result)
a=fitNormDistr(dcoef,components=NULL,log.transform=FALSE,
               combine.plot=FALSE, output=FALSE)

# to repeat results of 'a', load seed attribute of a into RNG state
.Random.seed=attr(a,"seed")
# or, reset the seed with same unique number
# set.seed(123)

b=fitNormDistr(dcoef,components=NULL,log.transform=FALSE,
               combine.plot=FALSE,output=FALSE)

# if 'a' and 'b' are the same
mapply(identical,a[[1]],b[[1]])

#try with log transformation
set.seed(234)
c=fitNormDistr(dcoef,components=2,log.transform=TRUE,combine.plot=FALSE,
               output=FALSE)

# trying with some parameters provided(this will be applied to all dcoef
```

```

# results).
# with constrain = FALSE, this will be used as the starting values for the
# EM-algorithm
# normally we should deal with only one dataset when working with
# constrains, since it will apply to all of them.
folder3=system.file("extdata","HSF", package="sojourner")
trackll=compareFolder(c(folder3),input=3)
MSD=msd(trackll=trackll)
dcoef=Dcoef(MSD,dt=6,plot=TRUE,output=FALSE)

# try with constrain=TRUE, the values will be forced to equal the provided
# ones.
set.seed(345)
e=fitNormDistr(dcoef,means=c(0.3,0.5), constrain=TRUE)

```

---

fitRT

*Fitting Residence time (1-CDF)*


---

### Description

Calculate average residence time for trajectories by fitting 1-CDF (survival distribution).

### Usage

```
fitRT(trackll=NULL,x.max=30,N.min=1.5,t.interval=0.5,maxiter.search=1e3,
maxiter.optim=1e3,k.ns=FALSE)
```

### Arguments

t.interval	time interval for image acquisition. Default 0.5 sec.
maxiter.optim	A positive integer specifying the maximum number of iterations allowed for nls.
trackll	trajectory list generated by createTrackll() and processing. if NULL, user will be prompted to enter the trackll name.
x.max	The maximum range of X axis, i.e. time, for the output plot. Default 30 sec.
N.min	The minimal duration of trajectory length in the unit of second, trajectories shorter than N.min will be filtered out. Default 1.5 sec.
k.ns	Logical indicate or numeric used to control fitting.
maxiter.search	A positive integer specifying the maximum number of iterations allowed for nls.

### Details

Calculating average residence time of particles by fitting 1-CDF (survival distribution) of its trajectories.

Input track list (trackll) should be processed, merged trackll, the fitting will start right away after running the function. The maximum time range to be plotted can be set using x.max, this will not change the fitting result. The result of two-component fit will be shown on the plot as well as in the console. Fitting result is determined by the input trackll, and N.min (filtering of the trackll).



**Value**

- On the Console output: Result of both one and two-component fit and parameters of goodness of the fit.
- Plot: fitting curves will be plotted over the raw data, with the number of tracks and result of two-component fitting shown in the plot area.

**Examples**

```
# Generate trackll, and process,
# e.g. mask region of interest, merge tracks from multiple files.
folder=system.file("extdata","HSF",package="sojourner")
trackll=createTrackll(folder=folder, input=3)
trackll=maskTracks(folder,trackll)
trackll=mergeTracks(folder,trackll)

# Fit the residence time of trackll
fitRT(trackll=trackll,x.max=30,N.min=1.5,t.interval=0.5)
```

---

linkSkippedFrames      *linkSkippedFrames*

---

**Description**

link trajectories skipped (or do not appear for) a number of frames

**Usage**

```
linkSkippedFrames(trackll, tolerance, maxSkip, cores = 1)
```

**Arguments**

tolerance	Distance tolerance level measured in pixels after the frame skip.
maxSkip	Maximum number of frames a trajectory can skip.
trackll	A list of track lists.
cores	Number of cores used for parallel computation. This can be the cores on a workstation, or on a cluster. Tip: each core will be assigned to read in a file when paralleled.

**Details**

Given user input for a tolerance level to limit how far the next point after the skip can deviate from the last point in pixel distance and a maximum number of frame skips possible, all trajectories falling within these parameters are automatically linked, renamed, and ordered accordingly. For a maxSkip example, if the maxSkip for a trajectory ending in frame 7 was 3, the next linked trajectory can start up to a maximum frame of 11.

Although not required, in order for the output to have a frame record column (recommended), the input must have one as well.

The naming scheme for each linked track is as follows:

[Last five characters of the file name].[Start frame #].[Length].[Track #]. [# of links]

Track List: [full name of input file]

Track: [Last five characters of the file name].[Start frame].[Length]. [Track].[# of links]. [Index in overall list (will differ from Track # when merging)]

(Note: The last five characters of the file name, excluding the extension, cannot contain '.')

## Value

linked trackll

## Examples

```
folder=system.file('extdata', 'SWR1', package='sojourner')
trackll=createTrackll(folder=folder, input=3)
#Basic function call of linkSkippedFrames
trackll.linked <- linkSkippedFrames(trackll, tolerance = 5, maxSkip = 10)
```

---

maskTracks

*maskTracks*

---

## Description

apply binary image masks to lists of track lists

## Usage

```
maskTracks(folder, trackll)
```

```
filterOnCell(trackll, numTracks = 0)
```

```
sampleTracks(trackll, num = 0)
```

## Arguments

folder	Full path to the output files.
trackll	A list of track lists.
numTracks	Minimum number of required tracks in the trackll
num	Number of tracks to randomly sample per trackl in trackll

**Details**

**IMPORTANT:** It will take an extremely long time to mask large datasets. Filter/trim first using `filterTrack()` and `trimTrack()`, then mask using `maskTracks(folder, trackll)`! Note the mask file should have the same name as the output files with a `'_MASK.tif'` ending. If there are more mask files than `trackll`, masking will fail. If there are less mask files, `tracklls` without masks will be deleted. Users can use `plotMask()` and `plotTrackOverlay()` to see the mask and its effect on screening tracks.

`filterOnCell()` eliminates all `trackl` in `trackll` that has less than `numTracks` tracks.

`sampleTracks()` randomly samples `num` number of tracks for each `trackl` in `trackll`.

**Value**

masked tracks in `trackll` format

**Examples**

```
#Basic masking with folder path with image masks
folder = system.file('extdata', 'ImageJ', package='sojourner')
trackll = createTrackll(folder=folder, input=3)
trackll.masked <- maskTracks(folder = folder, trackll = trackll)

#Compare the masking effect
plotTrackOverlay(trackll)
plotTrackOverlay(trackll.masked)

#Plot mask
mask.list=list.files(path=folder, pattern='_MASK.tif', full.names=TRUE)
plotMask(folder)

#If Nuclear image is available
plotNucTrackOverlay(folder=folder, trackll=trackll)
plotNucTrackOverlay(folder=folder, trackll=trackll.masked)

#Plot mask
plotMask(folder=folder)
```

---

mergeTracks

*mergeTracks*


---

**Description**

merge track lists in a list into one

**Usage**

```
mergeTracks(folder, trackll)
```

**Arguments**

folder	Full path to the output files.
trackll	A list of track lists.

**Details**

IMPORTANT: Once a trackll has been merged, it cannot be masked using maskTracks().

Merging creates the following data structure: (1) first level is the folder name, (2) second level is a list of data.frames/tracks from all output files merged into one.

If not merged, track lists takes the name of individual files in the folder. If merged, the single merged track list takes the folder name.

**Value**

modified trackll

**Examples**

```
#Basic masking with folder path with image masks
folder = system.file('extdata', 'SWR1', package = 'sojourner')
trackll=createTrackll(folder=folder, input=3)
trackll.merged <- mergeTracks(folder = folder, trackll = trackll)
```

---

msd

*msd*


---

**Description**

calculate mean square displacement for individual trajectory or summarize on trajectories.

**Usage**

```
msd(trackll,dt=6,resolution=0.107,summarize=FALSE,cores=1,
plot=FALSE,output=FALSE,filter=c(min=7,max=Inf))
  msd.track.vecdt(trackll,vecdt=NULL,resolution=0.107,output=FALSE)
  msd.perc(trackll,percentage=0.25,filter=c(min=7,max=Inf),
  trimmer=c(min=1,max=31),resolution=0.107,output=FALSE)
```

**Arguments**

dt	Time intervals. Default 6.
resolution	ratio of pixel to uM.
trackll	Track list output from createTrackll().
summarize	An logical indicate if MSD should be calculated on individual trajectories (Default) or summarized on all trajectories.

filter	a vector specifies the minimum and max length of trajectories to be analyzed. Take only trajectories that has number of frames greater than ( $\geq$ ) min and less than ( $<$ ) max.
cores	Number of cores used for parallel computation. This can be the cores on a workstation, or on a cluster. Tip: the computation on each file will be parallel assigned to each CPU core.
plot	An logical indicate if plot should be generated. See Values for detail.
output	An logical indicate if output should be generated. See Values for detail.
vecdt	A list containing varying dt values.
percentage	compute msd based on (tierd) percentage of its total length.
trimmer	vector used for trimming via trimTrack()

### Details

msd() calculate track (/trajectory)'s mean square displacement as a function of time (dt). For a track of N steps, at each dt, there are N-dt number of sub-trajectory/sub-tracks, mean of dt-wise sub-trajectories/ step-wise sub tracks average subtracks into one number at each dt.

the dt number of sub-tracks each contains N-dt steps. Because minimum step is 1 ( $N-dt \geq 1$ ), so the maximum dt is N-1 ( $dt \leq N-1$ ). As dt increase, the number of steps used to generate that mean decrease with the maximum dt ( $dt=N-1$ ) generated from one step.

if one wants to focus on a group of trajectory's evolution, he can simply filter on a number that is bigger than the dt he wanted to plot MSD.

by assigning cores, computation is paralleled on each each list of trackll (corresponding to one movie file).

### Value

- SummarizedMSD MSD summarized over all trajectories as a function of dt.
- IndividualMSD MSD of individual trajectories at specified dt. Row number corresponding to its dt. Notice only the trajectories that satisfies the specified dt is output, trajectories that does not satisfy (i.e. trajectories satisfies  $1:(dt-1)$ ) is not output here.
- StandardError Standard Error of the sample mean measures the variations of sample mean to underlying mean, it is estimated as  $SE=SD/\sqrt{N}$ .
- SampleSize The sample size (number of tracks/trajectories) used for calculating the msd and standard error.
- Trackll The msd function also returns the processed trackll. If passed to a variable, one can then export the trackll with this variable.

### Examples

```
# read in using createTrackll()
folder=system.file("extdata","SWR1",package="sojourner")
trackll=createTrackll(folder=folder, input=3)

# filter track based length
trackll.flt=filterTrack(trackll,filter=c(min=5,max=Inf))
```

```
msd=msd(track11.flt,dt=6,summarize=TRUE,plot=TRUE)
str(msd)

# focus on a group of trajectory by setting filter greater than dt
track11.flt2=filterTrack(track11,filter=c(min=7,max=Inf))
msd2=msd(track11.flt2,dt=6,summarize=FALSE,plot=TRUE) # individual
msd2=msd(track11.flt2,dt=6,summarize=TRUE,plot=TRUE) # summarized
```

---

plotIndividual\_Tracks *plotIndividual\_Tracks*

---

## Description

Plot individual tracks one by one, with grid layout. Tracks are sorted by their lengths.

## Usage

```
plotIndividual_Tracks(track11=track11,grid.size=c(1000,1000),
resolution=0.107,t.interval=0.5)
```

## Arguments

track11	trajectory list generated by createTrack11() and processing.
grid.size	The size of each plot grid in nanometers. Default 1000 nm for x and y axis, respectively.
resolution	ratio of pixel to uM.
t.interval	Time interval for image aquisition. Default 0.5 sec (500ms).

## Details

Plot individual track/trajectory one by one, and the tracks will be layout in 15X8 grids.

## Value

- PDF: One PDF file with 120 tracks on each page.

## Examples

```
# Generate track11, and process,
# e.g. mask region of interest, tracks from multiple files should not be
# merged.
folder=system.file('extdata','HSF',package='sojourner')
track11=createTrack11(folder=folder, input=3)
track11=filterTrack(track11,filter=c(7,Inf))
track11=maskTracks(folder, track11)
track11=mergeTracks(folder, track11)
```

```
# Plot individual tracks,  
plotIndividual_Tracks(trackll,grid.size=c(1000,1000),resolution=0.107,  
                      t.interval=0.5)
```

---

```
plotLocalizations_Density  
    plotLocalizations_Density
```

---

### Description

Plot localization map of molecules from a list of files in a folder with color coded by local density of each molecule.

### Usage

```
plotLocalizations_Density(trackll=trackll,scale=256,r=125,file.No=0,  
point.scale=0.15)
```

### Arguments

trackll	trajectory list generated by createTrackll() and processing. if NULL, user will be prompted to enter the trackll name.
scale	The pixel scale of image data.
r	Radius within each molecule to calculate density in nanometer (nm).
file.No	Select file(s) in the folder to plot. Default 0 for plotting all files in the folder.
point.scale	Size of the dots representing the molecules.

### Details

Plot localization map of molecules from a list of files in a folder with color coded by local density of each molecule. The localization of molecule is considered as the first position of its track.

Upon running of the function, users will be prompted to input the name of the track list (trackll). Input un-merged trackll and the plotting will start. The local density of each molecule is calculated by counting the number of molecules within a given radius around the position of the molecule. The higher the number, the higher the local density.

### Value

- PDF: One PDF file with one plot on each page.

**Examples**

```
# Generate trackll, and process,
# e.g. mask region of interest, tracks from multiple files should not be
# merged.
folder=system.file('extdata','HSF',package='sojourner')
trackll=createTrackll(folder=folder, input=3)
trackll=maskTracks(folder, trackll)

# Plot localization map,
plotLocalizations_Density(trackll=trackll, scale=128,
                          r=125, file.No=0, point.scale=0.3)
# Plot only file No. 2 and increase the point size,
plotLocalizations_Density(trackll=trackll, scale=128,
                          r=125, file.No=2, point.scale=1)
```

---

plotTrack

*plotTrack*


---

**Description**

Plot track/trajectory from track list. either randomly or specified.

**Usage**

```
plotTrack(ab.trackll, resolution=0.107, frame.min=8, frame.max=100,
          frame.start=1, frame.end=500)
```

```
plotTrackFromIndex(index.file, movie.folder,
                   resolution=0.107, frame.min=1, frame.max=100,
                   frame.start=1, frame.end=500, input=1)
```

```
plotTrackOverlay(trackll, max.pixel=128, nrow=2, ncol=2, width=16, height=16)
```

```
plotNucTrackOverlay(folder, trackll=NULL, cores=1,
                   max.pixel=128, nrow=2, ncol=2, width=16, height=16)
```

```
plotMask(folder, max.pixel=128, nrow=2, ncol=2, width=16, height=16)
```

```
trackOverlayData(trackl)
```

**Arguments**

resolution	ratio of pixel to uM.
frame.min	minimum frame number for plotting.
frame.max	max frame number for plotting.
frame.start	the first frame to plot. Default 1.



frame.end	last frame to plot. Default 500.
ab.trackll	absolute coordinates for plotting, generated from createTrackll(folder, input, ab.track=TRUE).
trackl	Track list
trackll	Track list output from createTrackll().
folder	folder containing desired input data.
index.file	a csv file that contains index of tracks in the first column. Leave a header line when preparing such a file.
movie.folder	the path to the folder which contains track files (presumably it is the same folder with movie files).
max.pixel	Number of pixels of imaging regime.
nrow	Number of rows in the final plot.
ncol	Number of columns in the final plot.
width	Width of the page for plotting.
height	Height of the page for plotting.
cores	Number of cores to be used.
input	Input file type (Diatrack .txt file = 1; Diatrack .mat session file = 2; ImageJ .csv file = 3; SlimFast .txt file = 4).

### Details

- plotTrackFromIndex: if user provide a csv file with first column listing the index of trajectories, this program will plot the tracks listed in the csv file. It is useful after manipulating with the output from Dcoef, to plot the tracks that of interest to the user (e.g. highest Dcoef). User need to provide the indexFile.csv, and specify the movie folder which contains the movies where specified trajectories are tracked.
- plotTrackOverlay: plot all tracks in trackll overlaid on one plot.
- plotNucTrackOverlay: plot tracks in a movie overlaid with nuclei image. The nuclei image file must end with "\_Nuclei.tif" to be recognized. If trackll is NULL (default), program will read in trackll from specified folder and return trackll, otherwise it will take the specified trackll directly.
- plotComponentTrackOverlay: plot tracks base on component fitting of diffusion coefficient. Combined with selComponentTracks() function, together it allows select and plot tracks based on component fitting of track diffusion coefficient. (This function is currently in sojourner.pro)
- plotMask: plot image mask. The mask file name must ended with \_MASK.tif to be recognized.

### Value

- PDF One PDF file with all the frames satisfy the criteria. If trackll has multiple items, it outputs multiple PDF files each corresponding to one item.
- csv Outputs csv file of the coordinates of the trajectory, which users can use other plotting software (e.g. Prism or Excel) to plot tracks in their favor.

**Examples**

```

folder=system.file("extdata","SWR1",package="sojourner")
trackll.ab=createTrackll(folder=folder, input=3, ab.track=TRUE)
plotTrack(trackll.ab)

## plot from index file
index.file=
  system.file("extdata","INDEX","indexFile.csv",package="sojourner")
movie.folder=system.file("extdata","SWR1",package="sojourner")
plotTrackFromIndex(
  index.file=index.file,movie.folder = movie.folder,input = 3)

## index file contain trajectories from multiple movie folders
folder1=system.file("extdata","SWR1",package="sojourner")
folder2=system.file("extdata","HTZ1",package="sojourner")
index.file2=system.file(
  "extdata","INDEX","indexFile2.csv",package="sojourner")
plotTrackFromIndex(
  index.file=index.file2, movie.folder = c(folder1,folder2),input = 3)

## masking with image mask
track.folder=system.file("extdata","SWR1_2",package="sojourner")
trackll=createTrackll(folder=track.folder, input=3)
trackll.masked <- maskTracks(folder=track.folder, trackll=trackll)
str(trackll,1)
str(trackll.masked,1)

## compare the masking effect
plotTrackOverlay(trackll,nrow=1,ncol=1,width=8,height=8)
plotTrackOverlay(trackll.masked,nrow=1,ncol=1,width=8,height=8)

## compare masking effect with nuclei image
plotNucTrackOverlay(folder=track.folder,trackll,
  nrow=1,ncol=1,width=8,height=8)
plotNucTrackOverlay(folder=track.folder,trackll.masked,
  nrow=1,ncol=1,width=8,height=8)

## plot mask
plotMask(track.folder,nrow=1,ncol=1,width=8,height=8)

```

---

plotTrackOverlay\_Dcoef

*plotTrackOverlay\_Dcoef*

---

**Description**

Plot track/trajectory overlays from a list of files in a folder with color coded by the Diffusion Coefficient (Dcoef) of each track/trajectory.

**Usage**

```
plotTrackOverlay_Dcoef(trackll=trackll,scale=128,dt=6,
  filter=c(min=7,max=Inf),
  resolution=0.107,rsquare=0.8,t.interval=0.01,Dcoef.range=c(-6,2),
  color=c('blue', 'white', 'red'),folder=NULL,file.No=0,line.width=0.1)
```

**Arguments**

trackll	trajectory list generated by createTrackll() and processing. if NULL, user will be prompted to enter the trackll name.
scale	The pixel scale of image data.
dt	Time intervals.
filter	Filter the tracks by step/frame number (length). Only tracks pass through filter will be selected.
resolution	ratio of pixel to uM.
rsquare	R square filter on Dcoef results. Default to be 0.8. Set value to 0 if rsquare filter is not desired.
t.interval	Time interval for image acquisition. Default 0.01 sec (10ms).
Dcoef.range	Select tracks whose Dcoef is within the range. Other tracks will not be plotted.
color	colors to interpolate to build the color gradient/ramp.
folder	Path to provide nuclear glow background. Images should end with '_Nuclei.tif'. Default is NULL and not adding background.
file.No	Select file(s) in the folder to plot. Default 0 for plotting all files in the folder.
line.width	Line width for plotted tracks, default is 0.1.

**Details**

Plot track/trajectory overlays from a list of files in a folder with color coded by the Diffusion Coefficient (Dcoef) of each track/trajectory. The range of Dcoef can be defined. Nuclei backgrounds can be added to the plots. Tracks from each file will be plot in one plot, i.e., one plot per file. Colors are flexible.

Upon running of the function, users will be prompted to input the name of the track list (trackll). Input un-merged trackll and the plotting will start. The Dcoef is calculated by 'static' method, which stabilizes the number of time lags used for fitting using time lag 2~ 5 despite the total time lags measured.

**Value**

- PDF: One PDF file with one plot on each page.

**Examples**

```
# Generate trackll, and process,
# e.g. mask region of interest, tracks from multiple files should not be
# merged.
```

```

folder=system.file('extdata', 'HSF', package='sojourner')
track11=createTrack11(folder=folder, input=3)
track11=maskTracks(folder, track11)

# Plot track overlays,
#plotTrackOverlay_Dcoef(track11=track11, scale=128, dt=6,
#filter=c(min=7, max=Inf),
#resolution=0.107, rsquare=0.8, t.interval=0.01, Dcoef.range=c(-6, 2),
#color=c('blue', 'white', 'red'), folder=NULL, file.No=0, line.width=0.1)
plotTrackOverlay_Dcoef(track11=track11, color=c('red', 'yellow'),
folder=folder, file.No=0)
plotTrackOverlay_Dcoef(track11=track11, color=c('red', 'yellow'),
folder=folder, file.No=c(1, 2))

```

---

sojourner

*sojourner - statistical analysis of single molecule trajectories*


---

## Description

Single molecule tracking reports live biophysical properties of molecules being investigated besides their coding sequence. It has evolved as a novel approach complementing genomic sequencing. Here we provided “sojourner” package, to address statistical and bioinformatic needs related to the analysis and comprehension of high throughput single molecule tracking data.

## Details

sojourner package provides statistical analysis of biophysical properties of single molecules. Current version primarily uses mean square displacement (MSD) based analysis, more functions using displacement based and hidden Markov model based method will be added in future release.

Current version includes: dwellTime (duration of the tracks), squareDisp (squared displacement), MSD (mean square displacement), Dcoef(diffusion coefficient), CDF(cumulative distribution function), etc., and various plotting functions for visualization. It also includes a graphical user interface (GUI) sojournerGUI(). The GUI version has the most frequently used functions but not all command line ones.

---

sojournerGUI

*sojournerGUI*


---

## Description

Graphical user interface (GUI) for sojourner.

## Usage

```
sojournerGUI()
```

**Details**

The GUI version has the most frequently used functions but not all command line ones. Current version includes: MSD (mean square displacement), Dcoef(diffusion coefficient), CDF(cumulative distribution function), and RT(residence time) etc.

**Value**

A graphical user interface for sojourner.

**Examples**

```
# not run
# library(sojourner)
# sojournerGUI()
```

---

squareDisp

*squareDisp*

---

**Description**

calculate square displacement of a track/trajectory as a function of time/step. data.frame has two column, x and y also calculate dx, dy bivariate squareDispCpp is the cpp version of squareDisp

**Usage**

```
squareDisp(track,dt=1,resolution=0.107)
squareDispCpp(track,dt=1,resolution=0.107)
```

**Arguments**

track	track dataframe with x and y coordinates.
dt	time step size(in frames).
resolution	resolution value, default is 0.107.

**Value**

- list of square displacements( $dx^2 + dy^2$ ) for varying dt values from 1 to dt

**Examples**

```
folder1=system.file('extdata','SWR1',package='sojourner')
folder2=system.file('extdata','HTZ1',package='sojourner')
track11=compareFolder(folders=c(folder1,folder2), input=3)
#use default filter with min=7
filtered.track11=filterTrack(track11)
track.dt=squareDisp(filtered.track11[[1]][[1]],dt=6)
```

# Index

`.dwellTime (dwellTime)`, 11  
`.exportRowWise (exportTrackll)`, 12  
`.linkSkippedFrames (linkSkippedFrames)`, 17  
`.one.comp.fit.rt (fitRT)`, 16  
`.plotLocalizations.Density (plotLocalizations_Density)`, 23  
`.plotTrack (plotTrack)`, 24  
`.plotTrackOverlay.Dcoef (plotTrackOverlay_Dcoef)`, 26  
`.singlecompFit (fitNormDistr)`, 14

`combineTrackll`, 2  
`compare_RT_CDF`, 4  
`compareFolder`, 3  
`createTrackll`, 5

`Dcoef`, 7  
`displacement.track (displacementCDF)`, 9  
`displacementCDF`, 9  
`dispVariance`, 10  
`dispVariance_track (dispVariance)`, 10  
`dwellTime`, 11

`exportTrackll`, 12

`filterOnCell (maskTracks)`, 18  
`filterTrack`, 13  
`fitNormDistr`, 14  
`fitRT`, 16

`linkSkippedFrames`, 17

`maskPoint (maskTracks)`, 18  
`maskTracks`, 18  
`mergeTracks`, 19  
`msd`, 20

`plotIndividual_Tracks`, 22  
`plotLocalizations_Density`, 23  
`plotMask (plotTrack)`, 24  
`plotNucTrackOverlay (plotTrack)`, 24  
`plotTrack`, 24  
`plotTrackFromIndex (plotTrack)`, 24  
`plotTrackOverlay (plotTrack)`, 24  
`plotTrackOverlay_Dcoef`, 26

`sampleTracks (maskTracks)`, 18  
`sojourner`, 28  
`sojournerGUI`, 28  
`squareDisp`, 29  
`squareDispCpp (squareDisp)`, 29

`trackLength (filterTrack)`, 13  
`trackOverlayData (plotTrack)`, 24  
`trimTrack (filterTrack)`, 13