

# Package ‘cellmigRation’

January 25, 2022

**Type** Package

**Title** Track Cells, Analyze Cell Trajectories and Compute Migration  
Statistics

**Version** 1.3.0

**Date** 2021-05-11

**Description**

Import TIFF images of fluorescently labeled cells, and track cell movements over time. Parallelization is supported for image processing and for fast computation of cell trajectories. In-depth analysis of cell trajectories is enabled by 15 trajectory analysis functions.

**biocViews** CellBiology, DataRepresentation, DataImport

**License** GPL-2

**Encoding** UTF-8

**LazyData** false

**Depends** R (>= 4.1), methods, foreach

**Imports** tiff, graphics, stats, utils, reshape2, parallel, doParallel,  
grDevices, matrixStats, FME, SpatialTools, sp, vioplot,  
FactoMineR, Hmisc

**Suggests** knitr, rmarkdown, dplyr, ggplot2, RUnit, BiocGenerics,  
BiocManager, kableExtra, rgl

**RoxygenNote** 7.1.1

**VignetteBuilder** knitr

**BugReports** <https://github.com/ocbe-uio/cellmigRation/issues>

**URL** <https://github.com/ocbe-uio/cellmigRation/>

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

**git\_branch** master

**git\_last\_commit** 71dc596

**git\_last\_commit\_date** 2021-10-26

**Date/Publication** 2022-01-25

**Author** Salim Ghannoum [aut, cph],  
 Damiano Fantini [aut, cph],  
 Waldir Leoncio [cre, aut],  
 Øystein Sørensen [aut]

**Maintainer** Waldir Leoncio <w.l.netto@medisin.uio.no>

## R topics documented:

aggregateFR . . . . .	3
aggregateTrackedCells . . . . .	5
CellMig-class . . . . .	6
CellMigPCA . . . . .	7
CellMigPCAclust . . . . .	8
CellMigPCAclustALL . . . . .	9
CellTracker . . . . .	11
ComputeTracksStats . . . . .	13
DiAutoCor . . . . .	14
DiRatio . . . . .	15
DiRatioPlot . . . . .	16
EstimateDiameterRange . . . . .	17
FilterTrackedCells . . . . .	18
FinRes . . . . .	19
FMI . . . . .	20
ForwardMigration . . . . .	21
getAvailableAggrMetrics . . . . .	22
getCellImages . . . . .	23
getCellMigSlot . . . . .	24
getCellsMeta . . . . .	24
getCellsStats . . . . .	25
getCellTrackMeta . . . . .	26
getCellTracks . . . . .	26
getCellTrackStats . . . . .	27
getDACTable . . . . .	28
getDiRatio . . . . .	29
getFMITable . . . . .	30
getForMigtable . . . . .	31
getImageCentroids . . . . .	32
getImageStacks . . . . .	32
getMSDtable . . . . .	33
getOptimizedParameters . . . . .	34
getOptimizedParams . . . . .	35
getPerAndSpeed . . . . .	35
getPopulationStats . . . . .	36
getProcessedImages . . . . .	37
getProcessingStatus . . . . .	37
getResults . . . . .	38
getTracks . . . . .	39

getVACtable . . . . .	39
LoadTiff . . . . .	40
MSD . . . . .	41
OptimizeParams . . . . .	43
PerAndSpeed . . . . .	44
plot3DAllTracks . . . . .	46
plot3DTracks . . . . .	47
plotAllTracks . . . . .	48
plotSampleTracks . . . . .	49
PlotTracksSeparately . . . . .	50
rmPreProcessing . . . . .	51
setAnalyticParams . . . . .	52
setCellMigSlot . . . . .	53
setCellsMeta . . . . .	53
setCellTracks . . . . .	54
setExpName . . . . .	55
setOptimizedParams . . . . .	56
setProcessedImages . . . . .	56
setProcessingStatus . . . . .	57
setTrackedCellsMeta . . . . .	58
setTrackedCentroids . . . . .	58
setTrackedPositions . . . . .	59
setTrackingStats . . . . .	60
trackedCells-class . . . . .	60
TrajectoryDataset . . . . .	61
VeAutoCor . . . . .	62
visualizeCellTracks . . . . .	63
VisualizeStackCentroids . . . . .	64
wsaPreProcessing . . . . .	65
<b>Index</b>	<b>67</b>

---

 aggregateFR

*Aggregating the outcome of several experiments or conditions.*


---

## Description

Aggregate two or more CellMig-class objects together. Input objects must carry information of trajectory analyses (otherwise an error will be raised). All trajectory results from the different experiments/conditions are returned in two data frames.

## Usage

```
aggregateFR(x, ..., export = FALSE)
```

**Arguments**

x	CellMig class object, which is a list of data frames resulted from the PreProcessing.
...	one or more CellMig-class object(s) where cells' trajectories have already been analyzed.
export	if 'TRUE' (default), exports function output to CSV file

**Details**

The visualization shows centered trajectories where the starting point of each track is located at the origin of the coordinate system (X=0,Y=0).

**Value**

two data frames: The first data frame shows the average of each parameter per experiment/condition. The second data frame shows the parameters of individual cells of all experiments/conditions.

**Author(s)**

Damiano Fantini and Salim Ghannoum <salim.ghannoum@medisin.uio.no> Damiano Fantini, <damiano.fantini@gmail.com>

**References**

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/)

**Examples**

```
data(WSADataset)
wasDF1 <- WSADataset[seq(1,300,by=1), ]
wsaTD1 <- CellMig(wasDF1)
wsaTD1 <- wsaPreProcessing(wsaTD1,FrameN=55)
wsaTD1 <-FMI(wsaTD1,TimeInterval=10)
wsaTD1 <-FinRes(wsaTD1,ParCor=FALSE, export=FALSE)
wasDF2 <- WSADataset[seq(500,700,by=1), ]
wsaTD2 <- CellMig(wasDF2)
wsaTD2 <- wsaPreProcessing(wsaTD2,FrameN=55)
wsaTD2 <-FMI(wsaTD2,TimeInterval=10)
wsaTD2 <-FinRes(wsaTD2,ParCor=FALSE, export=FALSE)
AGG<-aggregateFR(wsaTD1 ,wsaTD2 ,export=FALSE)
```

---

*aggregateTrackedCells Aggregate trackedCells Objects*

---

**Description**

Aggregate two or more trackedCells-class objects together. Input objects must carry information of cell tracks (otherwise an error will be raised). All tracks from the different experiments/images are returned in a large data.frame. A new unique ID is assigned to specifically identify each cell track from each image/experiment.

**Usage**

```
aggregateTrackedCells(  
  x,  
  ...,  
  meta_id_field = c("tiff_file", "experiment", "condition", "replicate")  
)
```

**Arguments**

x	a trackedCells-class object where cells have already been tracked
...	one or more trackedCells-class object(s) where cells have already been tracked
meta_id_field	string, can take one of the following values, c("tiff_file", "experiment", "condition", "replicate"). Indicates the meta-data column used as unique ID for the image/experiment. Can be abbreviated. Defaults to "tiff_file".

**Details**

each trackedCells-class object passed to this function requires a unique identifier (such as a unique tiff\_file name). Any of the metadata columns can be used as unique ID for an image/experiment. The function will raise an error if non-unique identifiers are found across the input objects.

**Value**

An aggregate data.frame including all cells that were tracked over two or more images/experiments. The data.frame includes the following columns: "new.ID", "frame.ID", "X", "Y", "cell.ID", "tiff\_name", "experiment", "condition", "replicate". The "new.ID" uniquely identifies a cell in a given image/experiment.

**Author(s)**

Damiano Fantini, <damiano.fantini@gmail.com>

**References**

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/)

**Examples**

```

# Please, see the package vignette
# for an example of how to use this function.
# A pseudo-code example is shown below
# Let x0, x1, x2, ... be trackedCells-class objects
# with a non-empty tracks slot.
x0 <- get(data(TrackCellsDataset))
x0 <- setCellsMeta(x0, experiment = "my_exp_01", condition = "CTRL")
x1 <- setCellsMeta(x0, experiment = "my_exp_01", condition = "DMSO")
x2 <- setCellsMeta(x0, experiment = "my_exp_01", condition = "DRUG")
y <- aggregateTrackedCells(x0, x1, x2, meta_id_field = "condition")
utils::head(y, 50)

```

---

CellMig-class

*The CellMig Class.*


---

**Description**

The CellMig class represents objects storing all information for both random migration (RM) and wound scratch assay (WSA). It comprises 14 slots.

**Usage**

```

CellMig(..., ExpName = NULL)

## S4 method for signature 'CellMig'
initialize(.Object, trajdata)

CellMig(..., ExpName = NULL)

```

**Arguments**

...	arguments to pass to the CellMig constructor
ExpName	string, experiment name (optional)
.Object	the CellMig object being built
trajdata	data frame including trajectory data

**Value**

An S4-class object  
a CellMig object

**Slots**

trajdata The raw trajectory data matrix organized into four columns: cell ID, X coordinates, Y coordinates and Track number, which is the track's path order.  
 adjDS A data frame of the trajectory data passed from the WSAprep function.  
 cellpos A binary vector showing on which side of the wound cells are located. "0" refers to a cell located above the wound whereas "1" refers to a cell located below the wound.  
 parE A numeric vector contains estimations for the imageH, woundH, upperE and lowerE.  
 preprocessedDS list object of data frames, each data frame shows the trajectories of a single cell.  
 DRtable A data frame of the results of running the DiRatio() function.  
 MSDtable A data frame of the results of running the MSD() function.  
 PerAanSpeedtable A data frame of the results of running the PerAndSpeed() function.  
 DACTable A data frame of the results of running the DiAutoCor() function.  
 VACTable A data frame of the results of running the VeAutoCor() function.  
 ForMigtable A data frame of the results of running the ForwardMigration() function.  
 FMItable A data frame of the results of running the FMI() function.  
 results A data frame of all the results.  
 parCor A data frame for Parameters Correlation.  
 meta A list including experiment name, meta data and other information.

**Author(s)**

Salim Ghannoum <salim.ghannoum@medisin.uio.no>

**Examples**

```
data("TrajectoryDataset")
CellMig(TrajectoryDataset)
```

---

CellMigPCA

PCA

---

**Description**

The CellMigPCA function automatically generates Principal Component Analysis.

**Usage**

```
CellMigPCA(object, parameters = c(1, 2, 3))
```

**Arguments**

object	CellMig class object, which is a list of data frames resulted from the PreProcessing.
parameters	A numeric vector contains the parameters to be included in the Principal Component Analysis. These numbers can be obtained from the outcome of the FinRes() function.

**Value**

PCA Graph of cells and PCA Graph of variables.

**Author(s)**

Salim Ghannoum <salim.ghannoum@medisin.uio.no>

**References**

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/)

**Examples**

```
data(WSADataset)
wasDF=WSADataset[seq(1,300,by=1),]
wsaTD <- CellMig(wasDF)
wsaTD <- wsaPreProcessing(wsaTD,FrameN=55)
wsaTD <-FMI(wsaTD,TimeInterval=10)
wsaTD <-ForwardMigration(wsaTD,TimeInterval=10)
wsaTD <-FinRes(wsaTD,ParCor=FALSE)
PCAplot<-CellMigPCA(wsaTD,parameters=c(1,4))
```

---

CellMigPCAclust

*PCA Clusters*

---

**Description**

The CellMigPCAclust function automatically generates clusters based on the Principal Component Analysis.

**Usage**

```
CellMigPCAclust(
  object,
  parameters = c(1, 2, 3),
  export = FALSE,
  interactive = TRUE
)
```



**Arguments**

object	CellMig class object, which is a list of data frames resulted from the PreProcessing.
parameters	A numeric vector contains the parameters to be included in the Principal Component Analysis. These numbers can be obtained from the outcome of the FinRes() function.
export	if 'TRUE' (default), exports function output to CSV file
interactive	logical, shall the PCA analysis be generated in a interactive fashion

**Value**

PCA Graph of cells and PCA Graph of variables.

**Author(s)**

Salim Ghannoum <salim.ghannoum@medisin.uio.no>

**References**

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/)

**Examples**

```
## The analysis only supports the interactive method!
## If interactive=FALSE, the function will return NULL
data(WSADataset)
wasDF <- WSADataset[seq(1, 300, by=1), ]
wsaTD <- CellMig(wasDF)
CellMigPCAclust(wsaTD, parameters=c(1,9), interactive=FALSE)
##
## A real world example is shown below (uncomment)
# data(WSADataset)
# wasDF <- WSADataset[seq(1,300,by=1),]
# wsaTD <- CellMig(wasDF)
# wsaTD <- wsaPreProcessing(wsaTD,FrameN=55)
# wsaTD <-FMI(wsaTD,TimeInterval=10)
# wsaTD <-ForwardMigration(wsaTD,TimeInterval=10)
# wsaTD <-FinRes(wsaTD,ParCor=FALSE)
# PCAclust <- CellMigPCAclust(wsaTD,parameters=c(1,9))
```

---

CellMigPCAclustALL      *PCA Clusters of different conditions*

---

**Description**

The CellMigPCAclust function automatically generates clusters based on the Principal Component Analysis.

**Usage**

```
CellMigPCAclustALL(
  object,
  ExpName = "PCA_Clusters",
  parameters = c(1, 2, 3),
  export = FALSE,
  interactive = TRUE
)
```

**Arguments**

object	CellMig class object, which is a list of data frames resulted from the PreProcessing.
ExpName	A character string. The ExpName will be appended to all exported tracks and statistics data.
parameters	A numeric vector contains the parameters to be included in the Principal Component Analysis. These numbers can be obtained from the outcome of the FinRes() function.
export	if 'TRUE' (default), exports function output to CSV file
interactive	logical, shall the PCA analysis be generated in a interactive fashion

**Value**

PCA Graph of cells and PCA Graph of variables.

**Author(s)**

Salim Ghannoum <salim.ghannoum@medisin.uio.no>

**References**

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/)

**Examples**

```
## The analysis only supports the interactive method!
## If interactive=FALSE, the function will return NULL
data(WSADataset)
wasDF1 <- WSADataset[seq(1,300,by=1), ]
wsaTD1 <- CellMig(wasDF1)
wsaTD1 <- wsaPreProcessing(wsaTD1,FrameN=55)
wsaTD1 <-FMI(wsaTD1,TimeInterval=10)
wsaTD1 <-FinRes(wsaTD1,ParCor=FALSE, export=FALSE)
wasDF2 <- WSADataset[seq(500,700,by=1), ]
wsaTD2 <- CellMig(wasDF2)
wsaTD2 <- wsaPreProcessing(wsaTD2,FrameN=55)
wsaTD2 <-FMI(wsaTD2, TimeInterval=10)
wsaTD2 <-FinRes(wsaTD2, ParCor=FALSE, export=FALSE)
AGG <- aggregateFR(wsaTD1, wsaTD2, export=FALSE)
```

```

CellMigPCAclustALL(AGG,ExpName="Aggregated_Conditions",
                  parameters=c(1,6), export=FALSE, interactive=FALSE)
# The previous line returns NULL
# In an interactive session, try running the following command (uncomment!)
# CellMigPCAclustALL(AGG,ExpName="Aggregated_Conditions",
#                   parameters=c(1,6), export=FALSE)

```

---

CellTracker

*Compute Cell Tracks*


---

### Description

Analyze Stacks, detect cells in each frame, and analyze cell tracks over time

### Usage

```

CellTracker(
  tc_obj,
  import_optiParam_from = NULL,
  min_frames_per_cell = 1,
  lnoise = NULL,
  diameter = NULL,
  threshold = NULL,
  maxDisp = NULL,
  memory_b = 0,
  goodenough = 0,
  threads = 1,
  show_plots = FALSE,
  verbose = FALSE,
  dryrun = FALSE
)

```

### Arguments

<code>tc_obj</code>	a trackedCells object.
<code>import_optiParam_from</code>	a trackedCells object (optional) used to import optimized parameters; can be NULL.
<code>min_frames_per_cell</code>	numeric, minimum number of consecutive frames in which a cell shall be found in order to retain that cell in the final cell tracks data.frame. Defaults to 1.
<code>lnoise</code>	numeric, lnoise parameter; can be NULL if OptimizeParams() has already been run
<code>diameter</code>	numeric, diameter parameter; can be NULL if OptimizeParams() has already been run

threshold	numeric, threshold parameter; can be NULL if OptimizeParams() has already been run
maxDisp	numeric, maximum displacement of a cell per time interval. When many cells are detected in each frame, small maxDisp values should be used.
memory_b	numeric, memory_b parameter as used in the original track.m function. In the current R implementation, only the value memory_b=0 is accepted
goodenough	numeric, goodenough parameter as used in the original track.m function. In the current R implementation, only the value goodenough=0 is accepted
threads	integer, number of cores to use for parallelization
show_plots	logical, shall cells detected in each frame of the image stack be visualized
verbose	logical, shall info about the progress of the cell tracking job be printed
dryrun	logical, shall a dryrun be performed

### Details

The `lnoise` param is used to guide a lowpass blurring operation, while the `lobject` param is used to guide a highpass background subtraction. The `threshold` param is used for a background correction following the initial image convolution

- **lnoise:** Characteristic lengthscale of noise in pixels. Additive noise averaged over this length should vanish. May assume any positive floating value. May be also set to 0, in which case only the highpass "background subtraction" operation is performed.
- **lobject** Integer length in pixels somewhat larger than a typical object. Can also be set to 0, in which case only the lowpass "blurring" operation defined by `lnoise` is done without the background subtraction defined by `lobject`
- **threshold** Numeric. By default, after the convolution, any negative pixels are reset to 0. Threshold changes the threshold for setting pixels to 0. Positive values may be useful for removing stray noise or small particles.

### Value

a `trackedCells` object

### Author(s)

Damiano Fantini, <damiano.fantini@gmail.com>

### References

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/) <https://www.mathworks.com/matlabcentral/fileexchange/60349-fasttracks>

### Examples

```
x <- get(data(TrackCellsDataset))
x <- CellTracker(x, dryrun=TRUE)
getTracks(x)[seq(1,12,by=1),]
```



DiAutoCor

*Direction AutoCorrelation***Description**

The DiAutoCor function automatically compute the angular persistence across several sequential time intervals.

**Usage**

```
DiAutoCor(
  object,
  TimeInterval = 10,
  sLAG = 0.25,
  sPLOT = TRUE,
  aPLOT = TRUE,
  export = FALSE,
  ExpName = NULL
)
```

**Arguments**

object	CellMig class object, which is a list of data frames resulted from the PreProcessing.
TimeInterval	A numeric value of the time elapsed between successive frames in the time-lapse stack.
sLAG	A numeric value to be used to get the number of lags for the slope fitting. Default is 0.25, which represents 25 percent of the steps.
sPLOT	A logical vector that allows generating individual plots showing the angular persistence across several sequential time intervals. Default is TRUE.
aPLOT	A logical vector that allows generating a plot showing the angular persistence across several sequential time intervals of all cells. Default is TRUE.
export	if 'TRUE' (default), exports function output to CSV file
ExpName,	string, name of the experiment. Can be NULL

**Value**

An CellMig class Object with a data frame and plots. The data frame, which contains six rows: "Cell Number", "Angular Persistence", "Intercept of DA quadratic model", "Mean Direction AutoCorrelation (all lags)", "Stable Direction AutoCorrelation through the track" and "Difference between Mean DA and Intercept DA".

**Author(s)**

Salim Ghannoum <salim.ghannoum@medisin.uio.no>

## References

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/)

## Examples

```
data(TrajectoryDataset)
rmDF=TrajectoryDataset[seq(1,220,by=1),]
rmTD <- CellMig(rmDF)
rmTD <- rmPreProcessing(rmTD,FrameN=55)
rmTD <- DiAutoCor(rmTD, TimeInterval=10, sLAG=0.25, sPLOT=FALSE,
                  aPLOT=FALSE, export=FALSE)
```

---

DiRatio	<i>Directionality Table</i>
---------	-----------------------------

---

## Description

Directionality Ratio is the displacement divided by the total length of the total path distance, where displacement is the straight line length between the start point and the endpoint of the migration trajectory,

## Usage

```
DiRatio(object, TimeInterval = 10, export = FALSE, ExpName = NULL)
```

## Arguments

object	CellMig class object, which is a list of data frames resulted from the PreProcessing.
TimeInterval	A numeric value of the time elapsed between successive frames in the time-lapse stack.
export	if 'TRUE' (default), exports function output to CSV file
ExpName	string

## Details

Directionality Ratio and Directional persistence

## Value

An CellMig class object with a data frame stored in the DRtable slot. It contains nine rows: "Cell Number", "Directionality Ratio", "Mean Cumulative Directionality Ratio", "Stable Directionality Ratio", "Number of returns", "Min CumDR", "Location of Min CumDR, Steps with less CumDR than DR", "Directional Persistence"

**Author(s)**

Salim Ghannoum <salim.ghannoum@medisin.uio.no>

**References**

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/)

**Examples**

```
rmTD <- get(data(preProcCellMig))
rmTD <- DiRatio(rmTD, export=FALSE)
```

---

DiRatioPlot

*Directionality Ratio plots*

---

**Description**

Directionality Ratio is the displacement divided by the total length of the total path distance, where displacement is the straightline length between the start point and the endpoint of the migration trajectory,

**Usage**

```
DiRatioPlot(object, TimeInterval = 10, export = FALSE, ExpName = NULL)
```

**Arguments**

object	CellMig class object, which is a list of data frames resulted from the PreProcessing.
TimeInterval	A numeric value of the time elapsed between successive frames in the time-lapse stack.
export	if 'TRUE' (default), exports plot to JPG file
ExpName	string, name of the experiment. Can be NULL

**Details**

Directionality Ratio

**Value**

Directionality Ratio plots

**Author(s)**

Salim Ghannoum <salim.ghannoum@medisin.uio.no>



## References

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/)

## Examples

```
rmTD <- get(data(preProcCellMig))
DiRatioPlot(object=rmTD, export=FALSE)
```

---

EstimateDiameterRange *Detect Particle Diameters in a Numeric matrix*

---

## Description

Estimates the diameters of particles in a numeric matrix

## Usage

```
EstimateDiameterRange(
  x,
  px.margin = 2,
  min.px.diam = 5,
  quantile.val = 0.99,
  plot = TRUE
)
```

## Arguments

x	numeric matrix corresponding to a digital image
px.margin	integer, number of pixels used as margin while searching/filtering for neighboring particles
min.px.diam	integer, minimum diameter of a particle (cell). Particles with a diameter smaller than min.px.diam are discarded
quantile.val	numeric, must be bigger than 0 and smaller than 1. Quantile for discriminating signal and background; only pixels with intensity higher than the corresponding quantile will count as signal while estimating particle diameters
plot	logical, shall a histogram of the distribution of diameters be shown

## Value

list including summary stats and data about the particles found in the image

## Author(s)

Damiano Fantini, <damiano.fantini@gmail.com>

**References**

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/)

**Examples**

```
a <- cbind(c(1, 1, 1, 0, 0, 0, 0, 0, 1, 1),
          c(1, 1, 0, 0, 0, 0, 0, 0, 1, 1),
          c(1, 0, 0, 0, 0, 0, 0, 0, 0, 0),
          c(0, 0, 0, 0, 1, 1, 0, 0, 0, 0),
          c(0, 0, 0, 1, 1, 1, 0, 0, 0, 0))
graphics::image(a)
b <- EstimateDiameterRange(a, min.px.diam = 2)
print(b$estim.cell.num)
print(b$raw)
```

---

FilterTrackedCells      *Filter an Aggregated Table of Cell Tracks*

---

**Description**

Filter an Aggregated Table (data.frame) of cell tracks (from multiple images/experiments) and retain cell tracks from images/experiments of interest

**Usage**

```
FilterTrackedCells(x, id_list, meta_id_field)
```

**Arguments**

x	data.frame, is an aggregated Table of Cell Tracks. Must include the following columns: "new.ID", "frame.ID", "X", "Y", "cell.ID", "tiff_name", "experiment", "condition", "replicate"
id_list	character vector, indicates the IDs (such as tiff_filenames) to be retained in the output data.frame
meta_id_field	string, can take one of the following values, c("tiff_file", "experiment", "condition", "replicate"). Indicates the meta-data column used as unique ID for the image/experiment. Can be abbreviated. Defaults to "tiff_file".

**Value**

data.frame, a filtered aggregated Table of Cell Tracks

**Author(s)**

Damiano Fantini, <damiano.fantini@gmail.com>

## References

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/)

## Examples

```
A <- data.frame(new.ID = seq(1,10,by=1), frame.ID = seq(10,1,by=(-1)),
               X = sample(seq(1,100,by=1), size = 10),
               Y = sample(seq(1,100,by=1), size = 10),
               cell.ID = c(rep(1, 5), rep(2, 5)),
               tiff_file= c(rep("ii", 3), rep("jj", 5), rep('kk', 2)))
FilterTrackedCells(A, id_list = c("jj", "kk"), "tiff_file")
```

---

FinRes

*Final Results*

---

## Description

The FinRes function automatically generates a data frame that contains all the results.

## Usage

```
FinRes(object, ParCor = TRUE, export = FALSE, ExpName = NULL)
```

## Arguments

object	CellMig class object, which is a list of data frames resulted from the PreProcessing.
ParCor	A logical vector that allows generating a correlation table. Default is TRUE.
export	if 'TRUE' (default), exports function output to CSV file
ExpName	string, name of the experiment. Can be NULL

## Value

A data frame that contains all the results.

## Author(s)

Salim Ghannoum <salim.ghannoum@medisin.uio.no>

## References

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/)

## Examples

```
data(WSADataset)
wasDF <- WSADataset[seq(1,300,by=1), ]
wsaTD <- CellMig(wasDF)
wsaTD <- wsaPreProcessing(wsaTD,FrameN=55)
wsaTD <-FMI(wsaTD,TimeInterval=10)
wsaTD <-ForwardMigration(wsaTD,TimeInterval=10,)
wsaTD <-FinRes(wsaTD,ParCor=FALSE, export=FALSE)
```

---

FMI

*Forward Migration Index*

---

## Description

The FMI function automatically generates data for the forward migration index

## Usage

```
FMI(object, TimeInterval = 10, export = FALSE, ExpName = NULL)
```

## Arguments

object	CellMig class object, which is a list of data frames resulted from the PreProcessing.
TimeInterval	A numeric value of the time elapsed between successive frames in the time-lapse stack.
export	if 'TRUE' (default), exports function output to CSV file
ExpName	string, name of the experiment. Can be NULL

## Value

An CellMig class Object with a data frame. The data frame is stored in the FMItable slot.

## Author(s)

Salim Ghannoum <salim.ghannoum@medisin.uio.no>

## References

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/)

**Examples**

```

data(WSADataset)
wasDF=WSADataset[seq(1,300,by=1),]
wsaTD <- CellMig(wasDF)
wsaTD <- wsaPreProcessing(wsaTD,FrameN=55)
wsaTD <-FMI(wsaTD,TimeInterval=10, export=FALSE)

```

---

ForwardMigration      *Forward Migration*

---

**Description**

The ForwardMigration function automatically generates data and plots for forward persistence and speed.

**Usage**

```

ForwardMigration(
  object,
  TimeInterval = 10,
  sfptPLOT = TRUE,
  afptPLOT = TRUE,
  sfpPLOT = TRUE,
  afpPLOT = TRUE,
  export = FALSE,
  ExpName = NULL
)

```

**Arguments**

object	CellMig class object, which is a list of data frames resulted from the PreProcessing.
TimeInterval	A numeric value of the time elapsed between successive frames in the time-lapse stack.
sfptPLOT	A logical vector that allows generating individual plots of persistence time vs speed per cell. Default is TRUE.
afptPLOT	A logical vector that allows generating a plot of persistence time vs speed for all cells. Default is TRUE.
sfpPLOT	A logical vector that allows generating individual plots of angular persistence vs speed per cell. Default is TRUE.
afpPLOT	A logical vector that allows generating a plot of angular persistence vs speed of all cells. Default is TRUE.
export	if 'TRUE' (default), exports function output to CSV file
ExpName	string, name of the experiment. Can be NULL

**Value**

An CellMig class Object with a data frame and plots. The data frame is stored in the ForMigtable slot.

**Author(s)**

Salim Ghannoum <salim.ghannoum@medisin.uio.no>

**References**

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/)

**Examples**

```
data(WSADataset)
wsaDF <- WSADataset[seq(1,500,by=1),]
wsaTD <- CellMig(wsaDF)
wsaTD <- wsaPreProcessing(wsaTD,FrameN=55)
wsaTD <- ForwardMigration(wsaTD, TimeInterval=10, sfptPLOT=FALSE,
                          afptPLOT= FALSE,sfpPLOT= FALSE,
                          afpPLOT= FALSE, export=FALSE)
```

---

getAvailableAggrMetrics

*Get Available Aggregate Cell Metrics*

---

**Description**

Retrieve a list of metrics computed for an aggregated result object. This getter function takes the output of aggregateFR() as input.

**Usage**

```
getAvailableAggrMetrics(object)
```

**Arguments**

object            list of length 2, returned by the aggregateFR() function

**Value**

character vector listing all available metrics

**Author(s)**

Damiano Fantini and Salim Ghannoum <salim.ghannoum@medisin.uio.no>

## References

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/)

## Examples

```
data(WSADataset)
wasDF1 <- WSADataset[seq(1,300,by=1), ]
wsaTD1 <- CellMig(wasDF1)
wsaTD1 <- wsaPreProcessing(wsaTD1,FrameN=65)
wsaTD1 <- FMI(wsaTD1,TimeInterval=10)
wsaTD1 <- FinRes(wsaTD1,ParCor=FALSE, export=FALSE)
wasDF2 <- WSADataset[seq(1001,1300,by=1), ]
wsaTD2 <- CellMig(wasDF2)
wsaTD2 <- wsaPreProcessing(wsaTD2,FrameN=65)
wsaTD2 <-FMI(wsaTD2,TimeInterval=10)
wsaTD2 <-FinRes(wsaTD2,ParCor=FALSE, export=FALSE)
AGG <- aggregateFR(wsaTD1 ,wsaTD2 ,export=FALSE)
getAvailableAggrMetrics(AGG)
```

---

getCellImages	<i>Method getCellImages</i>
---------------	-----------------------------

---

## Description

Retrieve Images from a trackedCells object.

## Usage

```
getCellImages(x)

## S4 method for signature 'trackedCells'
getCellImages(x)
```

## Arguments

x                    a trackedCells-class object

## Value

a list including all images

## Examples

```
data("TrackCellsDataset")
getCellImages(TrackCellsDataset)
```

---

getCellMigSlot	<i>Method getCellMigSlot</i>
----------------	------------------------------

---

**Description**

Get Data from a slot in a CellMig object.

**Usage**

```
getCellMigSlot(x, slot)

## S4 method for signature 'CellMig,character'
getCellMigSlot(x, slot)
```

**Arguments**

x	a CellMig-class object
slot	string pointing to the slot to be retrieved

**Value**

a slot from a CellMig object

**Examples**

```
data("TrajectoryDataset")
x <- CellMig(TrajectoryDataset)
getCellMigSlot(x, "trajdata")
```

---

getCellsMeta	<i>Get MetaData</i>
--------------	---------------------

---

**Description**

Extract MetaData from a trackedCells object

**Usage**

```
getCellsMeta(tc_obj)
```

**Arguments**

tc_obj	a trackedCells object
--------	-----------------------



**Value**

a list including four items: tiff filename, experiment name, condition label, and replicate ID.

**Author(s)**

Damiano Fantini, <damiano.fantini@gmail.com>

**References**

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/) <https://www.mathworks.com/matlabcentral/fileexchange/60349-fasttracks>

**Examples**

```
x0 <- get(data(TrackCellsDataset))
getCellsMeta(x0)
```

---

getCellsStats

*Get Cell migration stats*

---

**Description**

Extract cell migration statistics from a trackedCells object

**Usage**

```
getCellsStats(tc_obj)
```

**Arguments**

tc\_obj            a trackedCells object

**Value**

data.frame including cell migration stats

**Author(s)**

Damiano Fantini, <damiano.fantini@gmail.com>

**References**

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/) <https://www.mathworks.com/matlabcentral/fileexchange/60349-fasttracks>

**Examples**

```
x <- get(data(TrackCellsDataset))
getCellsStats(x)
```

---

getCellTrackMeta      *Method getCellTrackMeta*

---

**Description**

Retrieve Metadata from a trackedCells object.

**Usage**

```
getCellTrackMeta(x)

## S4 method for signature 'trackedCells'
getCellTrackMeta(x)
```

**Arguments**

x                    a trackedCells-class object

**Value**

a list including Meta Data

**Examples**

```
data("TrackCellsDataset")
getCellTrackMeta(TrackCellsDataset)
```

---

getCellTracks      *Method getCellTracks*

---

**Description**

Retrieve Cell Tracks from a trackedCells object.

**Usage**

```
getCellTracks(x)

## S4 method for signature 'trackedCells'
getCellTracks(x)
```

**Arguments**

x                    a trackedCells-class object

**Value**

a data.frame including Cell Tracks

**Examples**

```
data("TrackCellsDataset")
getCellTracks(TrackCellsDataset)
```

---

getCellTrackStats        *Method getCellTrackStats*

---

**Description**

Retrieve Stats from a trackedCells object.

**Usage**

```
getCellTrackStats(x)

## S4 method for signature 'trackedCells'
getCellTrackStats(x)
```

**Arguments**

x                    a trackedCells-class object

**Value**

a list including Track statistics

**Examples**

```
data("TrackCellsDataset")
getCellTrackStats(TrackCellsDataset)
```

---

`getDACtable`*Getting the Direction AutoCorrelation*

---

**Description**

The DiAutoCor function automatically compute the angular persistence across several sequential time intervals.

**Usage**

```
getDACtable(object)
```

**Arguments**

`object` CellMig class object, which is a list of data frames resulted from the PreProcessing.

**Value**

A data frame which contains six rows: "Cell Number", "Angular Persistence", "Intercept of DA quadratic model", "Mean Direction AutoCorrelation (all lags)", "Stable Direction AutoCorrelation through the track" and "Difference between Mean DA and Intercept DA".

**Author(s)**

Salim Ghannoum <salim.ghannoum@medisin.uio.no>

**References**

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/)

**Examples**

```
data(TrajectoryDataset)
rmDF=TrajectoryDataset[seq(1,300,by=1),]
rmTD <- CellMig(rmDF)
rmTD <- rmPreProcessing(rmTD,FrameN=55)
rmTD <- DiAutoCor(rmTD, TimeInterval=10, sLAG=0.25, sPLOT=FALSE,
                  aPLOT=FALSE, export=FALSE)
head(getDACtable(rmTD))
```

---

`getDiRatio`*Getting the Directionality Table*

---

**Description**

Directionality Ratio is the displacement divided by the total length of the total path distance, where displacement is the straight line length between the start point and the endpoint of the migration trajectory,

**Usage**

```
getDiRatio(object)
```

**Arguments**

`object` CellMig class object, which is a list of data frames resulted from the PreProcessing.

**Details**

Directionality Ratio and Directional persistence

**Value**

A data frame. It contains nine rows: "Cell Number", "Directionality Ratio", "Mean Cumulative Directionality Ratio", "Stable Directionality Ratio", "Number of returns", "Min CumDR", "Location of Min CumDR, Steps with less CumDR than DR", "Directional Persistence".

**Author(s)**

Salim Ghannoum <salim.ghannoum@medisin.uio.no>

**References**

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/)

**Examples**

```
rmTD <- get(data(preProcCellMig))
rmTD <- DiRatio(rmTD, export=FALSE)
head(getDiRatio(rmTD))
```

---

getFMitable	<i>Getting the Forward Migration Index</i>
-------------	--

---

### Description

The FMI function automatically generates data for the forward migration index

### Usage

```
getFMitable(object)
```

### Arguments

object	CellMig class object, which is a list of data frames resulted from the PreProcessing.
--------	---

### Value

A data frame for the FMI.

### Author(s)

Salim Ghannoum <salim.ghannoum@medisin.uio.no>

### References

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/)

### Examples

```
data(WSADataset)
wasDF=WSADataset[seq(1,300,by=1),]
wsaTD <- CellMig(wasDF)
wsaTD <- wsaPreProcessing(wsaTD,FrameN=55)
wsaTD <-FMI(wsaTD,TimeInterval=10, export=FALSE)
head(getFMitable(wsaTD))
```

---

getForMigtable	<i>Getting the Forward Migration</i>
----------------	--------------------------------------

---

### Description

The ForwardMigration function automatically generates data and plots for forward persistence and speed.

### Usage

```
getForMigtable(object)
```

### Arguments

object	CellMig class object, which is a list of data frames resulted from the PreProcessing.
--------	---

### Value

A data frame including values of the forward migration analysis.

### Author(s)

Salim Ghannoum <salim.ghannoum@medisin.uio.no>

### References

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/)

### Examples

```
data(WSADataset)
wsaDF <- WSADataset[seq(1,300,by=1),]
wsaTD <- CellMig(wsaDF)
wsaTD <- wsaPreProcessing(wsaTD,FrameN=55)
wsaTD <- ForwardMigration(wsaTD, TimeInterval=10, sfptPLOT=FALSE,
                          afptPLOT= FALSE, sfpPLOT= FALSE,
                          afpPLOT= FALSE, export=FALSE)
head(getForMigtable(wsaTD))
```

---

getImageCentroids      *Method getImageCentroids*

---

**Description**

Retrieve Image Centroids from a trackedCells object.

**Usage**

```
getImageCentroids(x)

## S4 method for signature 'trackedCells'
getImageCentroids(x)
```

**Arguments**

x                      a trackedCells-class object

**Value**

a list including all centroids

**Examples**

```
data("TrackCellsDataset")
getImageCentroids(TrackCellsDataset)
```

---

getImageStacks      *Get Image Stacks*

---

**Description**

Extract Images Stacks from a trackedCells object

**Usage**

```
getImageStacks(tc_obj)
```

**Arguments**

tc\_obj                a trackedCells object

**Value**

a list including stack images (formatted as numeric matrices)



**Author(s)**

Damiano Fantini, <damiano.fantini@gmail.com>

**References**

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/) <https://www.mathworks.com/matlabcentral/fileexchange/60349-fasttracks>

**Examples**

```
x0 <- get(data(TrackCellsDataset))
y0 <- getImageStacks(x0)
graphics::image(y0[[1]])
```

---

getMSDtable

*Getting the Mean Square Displacement*

---

**Description**

The MSD function automatically computes the mean square displacements across several sequential time intervals. MSD parameters are used to assess the area explored by cells over time.

**Usage**

```
getMSDtable(object)
```

**Arguments**

object	CellMig class object, which is a list of data frames resulted from the PreProcessing.
--------	---

**Value**

A data frame of MSD values.

**Author(s)**

Salim Ghannoum <salim.ghannoum@medisin.uio.no>

**References**

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/)

**Examples**

```
data(TrajectoryDataset)
rmDF <- TrajectoryDataset[seq(1,600,by=1), ]
rmTD <- CellMig(rmDF)
rmTD <- rmPreProcessing(rmTD,FrameN=100)
rmTD <- MSD(rmTD, sLAG=0.25, ffLAG=0.25, export=FALSE)
head(getMSDtable(rmTD))
```

---

getOptimizedParameters

*Get Auto Optimized Parameters*

---

**Description**

Extract Parameters that were automatically optimized

**Usage**

```
getOptimizedParameters(tc_obj)
```

**Arguments**

tc\_obj            a trackedCells object

**Value**

a list including optimized parameter values (Inoise, diameter, and threshold)

**Author(s)**

Damiano Fantini, <damiano.fantini@gmail.com>

**References**

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/) <https://www.mathworks.com/matlabcentral/fileexchange/60349-fasttracks>

**Examples**

```
x <- get(data(TrackCellsDataset))
getOptimizedParameters(x)
```

---

getOptimizedParams      *Method getOptimizedParams*

---

**Description**

Retrieve Optimized Params from a trackedCells object.

**Usage**

```
getOptimizedParams(x)

## S4 method for signature 'trackedCells'
getOptimizedParams(x)
```

**Arguments**

x                      a trackedCells-class object

**Value**

a list including Optimized Parameters

**Examples**

```
data("TrackCellsDataset")
getOptimizedParams(TrackCellsDataset)
```

---

getPerAndSpeed              *Getting the table of Persistence and Speed.*

---

**Description**

The PerAndSpeed() generates data and plots for persistence and speed.

**Usage**

```
getPerAndSpeed(object)
```

**Arguments**

object                      CellMig class object, which is a list of data frames resulted from the PreProcessing.

**Value**

A data frame of Persistence and Speed.

**Author(s)**

Salim Ghannoum <salim.ghannoum@medisin.uio.no>

**References**

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/)

**Examples**

```
rmTD <- get(data(preProcCellMig))
rmTD <- PerAndSpeed(rmTD,TimeInterval=10, export=FALSE)
head(getPerAndSpeed(rmTD))
```

---

getPopulationStats      *Get Cell population stats*

---

**Description**

Extract cell population statistics from a trackedCells object

**Usage**

```
getPopulationStats(tc_obj)
```

**Arguments**

tc\_obj                  a trackedCells object

**Value**

data.frame including cell population stats

**Author(s)**

Damiano Fantini, <damiano.fantini@gmail.com>

**References**

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/) <https://www.mathworks.com/matlabcentral/fileexchange/60349-fasttracks>

**Examples**

```
x <- get(data(TrackCellsDataset))
getPopulationStats(x)
```

---

getProcessedImages      *Method getProcessedImages*

---

**Description**

Retrieve Processed Images from a trackedCells object.

**Usage**

```
getProcessedImages(x)

## S4 method for signature 'trackedCells'
getProcessedImages(x)
```

**Arguments**

x                      a trackedCells-class object

**Value**

a list including all processed images

**Examples**

```
data("TrackCellsDataset")
getProcessedImages(TrackCellsDataset)
```

---

getProcessingStatus      *Method getProcessingStatus*

---

**Description**

Retrieve Processing Status from a trackedCells object.

**Usage**

```
getProcessingStatus(x)

## S4 method for signature 'trackedCells'
getProcessingStatus(x)
```

**Arguments**

x                      a trackedCells-class object

**Value**

a list including Processing Status

**Examples**

```
data("TrackCellsDataset")
getProcessingStatus(TrackCellsDataset)
```

---

getResults

*Final Results*

---

**Description**

The FinRes function automatically generates a data frame that contains all the results.

**Usage**

```
getResults(object)
```

**Arguments**

object            CellMig class object, which is a list of data frames resulted from the PreProcessing.

**Value**

A data frame that contains all the results.

**Author(s)**

Salim Ghannoum <salim.ghannoum@medisin.uio.no>

**References**

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/)

**Examples**

```
data(WSADataset)
wasDF <- WSADataset[seq(1,300,by=1), ]
wsaTD <- CellMig(wasDF)
wsaTD <- wsaPreProcessing(wsaTD,FrameN=55)
wsaTD <-FMI(wsaTD,TimeInterval=10)
wsaTD <-ForwardMigration(wsaTD,TimeInterval=10,)
wsaTD <-FinRes(wsaTD,ParCor=FALSE, export=FALSE)
head(getResults(wsaTD))
```

---

getTracks	<i>Get Track Data</i>
-----------	-----------------------

---

**Description**

Extract Track Data from a trackedCells object

**Usage**

```
getTracks(tc_obj, attach_meta = FALSE)
```

**Arguments**

tc_obj	a trackedCells object
attach_meta	logical, shall metaData be attached to tracks

**Value**

a data.frame including cell tracks data

**Author(s)**

Damiano Fantini, <damiano.fantini@gmail.com>

**References**

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/) <https://www.mathworks.com/matlabcentral/fileexchange/60349-fasttracks>

**Examples**

```
x <- get(data(TrackCellsDataset))
getTracks(x)[seq(1,10,by=1),]
```

---

getVACtable	<i>Getting the Velocity AutoCorrelation</i>
-------------	---

---

**Description**

The VeAutoCor function automatically compute the changes in both speed and direction across several sequantial time intervals.

**Usage**

```
getVACtable(object)
```

**Arguments**

object            CellMig class object, which is a list of data frames resulted from the PreProcessing.

**Value**

A data frame, which contains six rows: "Cell Number", "Velocity AutoCorrelation (lag=1)", "2nd normalized Velocity AutoCorrelation", "Intercept of VA quadratic model", "Mean Velocity AutoCorrelation (all lags)", "Mean |Acceleration|" and "Average Speed".

**Author(s)**

Salim Ghannoum <salim.ghannoum@medisin.uio.no>

**References**

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/)

**Examples**

```
data(TrajectoryDataset)
rmDF=TrajectoryDataset[seq(1,300,by=1),]
rmTD <- CellMig(rmDF)
rmTD <- rmPreProcessing(rmTD,FrameN=55)
rmTD <- VeAutoCor(rmTD, TimeInterval=10, sLAG=0.25, sPLOT=FALSE,
                  aPLOT=FALSE, export=FALSE)
head(getVACtable(rmTD))
```

---

LoadTiff

*Import Image from TIFF*

---

**Description**

Import a .tif stack containing fluorescently labeled point particles to be tracked

**Usage**

```
LoadTiff(tiff_file, experiment = NULL, condition = NULL, replicate = NULL)
```

**Arguments**

tiff\_file        path to a TIFF file to be read in  
 experiment     string, a label to describe the experiment (optional)  
 condition      string, a label to describe the experimental condition  
 replicate      string, a label to identify the replicate (optional)



**Value**

a trackedCells object

**Note**

'experiment', 'condition' and 'replicate' are optional arguments and can be NULL.

**Author(s)**

Damiano Fantini, <damiano.fantini@gmail.com>

**References**

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/) <https://www.mathworks.com/matlabcentral/fileexchange/60349-fasttracks>

**Examples**

```
# Let `path/to/tiff_file.tiff` be the path to tiff file we want to
# import. If an error is thrown, NULL is returned.
x <- LoadTiff(tiff_file = "path/to/tiff_file.tiff")
```

---

MSD

*Mean Square Displacement*

---

**Description**

The MSD function automatically computes the mean square displacements across several sequential time intervals. MSD parameters are used to assess the area explored by cells over time.

**Usage**

```
MSD(
  object,
  TimeInterval = 10,
  sLAG = 0.25,
  ffLAG = 0.25,
  SlopePlot = TRUE,
  AllSlopesPlot = TRUE,
  FurthPlot = TRUE,
  AllFurthPlot = TRUE,
  export = FALSE,
  ExpName = NULL
)
```

**Arguments**

object	CellMig class object, which is a list of data frames resulted from the PreProcessing.
TimeInterval	A numeric value of the time elapsed between successive frames in the time-lapse stack.
sLAG	A numeric value to be used to get the number of lags for the slope fitting. Default is 0.25, which represents 25 percent of the steps.
ffLAG	A numeric value to be used to get the number of lags for the Furth formula fitting. Default is 0.25, which represents 25 percent of the steps.
SlopePlot	A logical vector that allows generating individual plots showing the slope of the mean square displacement of the movement of individual cells. Default is TRUE.
AllSlopesPlot	A logical vector that allows generating a plot showing the slope of the mean square displacement of the movement of all cells. Default is TRUE.
FurthPlot	A logical vector that allows generating individual plots fitting the Furth formula using generalized regression by the Nelder–Mead method simplex method per cell. Default is TRUE.
AllFurthPlot	A logical vector that allows generating a plot fitting the Furth formula using generalized regression by the Nelder–Mead method simplex method for all cells. Default is TRUE.
export	if 'TRUE' (default), exports function output
ExpName	string, anem of the Experiment. Can be NULL

**Value**

An CellMig class object with a data frame and plots. The data frame is stored in the MSDtable slot.

**Author(s)**

Salim Ghannoum <salim.ghannoum@medisin.uio.no>

**References**

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/)

**Examples**

```
data(TrajectoryDataset)
rmDF <- TrajectoryDataset[seq(1,220,by=1), ]
rmTD <- CellMig(rmDF)
rmTD <- rmPreProcessing(rmTD,FrameN=100)
rmTD <- MSD(rmTD, sLAG=0.25, ffLAG=0.25, export=FALSE)
```

---

 OptimizeParams

*Optimize Detection Params*


---

**Description**

Optimize Detection Parameters for running a cell tracking job

**Usage**

```
OptimizeParams(
  tc_obj,
  lnoise_range = NULL,
  min.px.diam = 5,
  diameter_range = NULL,
  threshold_range = NULL,
  target_cell_num = NULL,
  threads = 1,
  quantile.val = NULL,
  px.margin = NULL,
  plot = FALSE,
  verbose = FALSE,
  dryrun = FALSE
)
```

**Arguments**

<code>tc_obj</code>	a trackedCells object
<code>lnoise_range</code>	numeric vector of lnoise values to be used in the optimization step. Can be NULL
<code>min.px.diam</code>	integer, minimum diameter of a particle (cell). Particles with a diameter smaller than <code>min.px.diam</code> are discarded
<code>diameter_range</code>	numeric vector of diameter values to be used in the optimization step. Can be NULL
<code>threshold_range</code>	numeric vector of threshold values to be used in the optimization step. Can be NULL
<code>target_cell_num</code>	integer, the expected (optimal) number of cells to be detected in each frame
<code>threads</code>	integer, number of cores to use for parallelization
<code>quantile.val</code>	numeric, argument passed to <code>EstimateDiameterRange()</code> . If NULL, it is defaulted to 0.99
<code>px.margin</code>	numeric, argument passed to <code>EstimateDiameterRange()</code> . If NULL, it is defaulted to 2
<code>plot</code>	if 'TRUE', plots results in the end
<code>verbose</code>	shall information about the progress of the operation be printed to screen/console
<code>dryrun</code>	shall a dryrun be performed

## Details

The `lnoise` param is used to guide a lowpass blurring operation, while the `lobject` param is used to guide a highpass background subtraction. The `lthreshold` param is used for a background correction following the initial image convolution

- **lnoise**: Characteristic lengthscale of noise in pixels. Additive noise averaged over this length should vanish. May assume any positive floating value. May be also set to 0, in which case only the highpass "background subtraction" operation is performed.
- **lobject**: Integer length in pixels somewhat larger than a typical object. Can also be set to 0, in which case only the lowpass "blurring" operation defined by `lnoise` is done without the background subtraction defined by `lobject`
- **lthreshold**: Numeric. By default, after the convolution, any negative pixels are reset to 0. Threshold changes the threshold for setting pixels to 0. Positive values may be useful for removing stray noise or small particles.

## Value

a `trackedCells` object

## Author(s)

Damiano Fantini, <damiano.fantini@gmail.com>

## References

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/) <https://www.mathworks.com/matlabcentral/fileexchange/60349-fasttracks>

## Examples

```
x <- get(data(TrackCellsDataset))
x <- OptimizeParams(tc_obj = x, lnoise_range = c(5,7,10),
                   diameter_range = c(12,14,18),
                   threshold_range = c(4,7), dryrun = TRUE)
getOptimizedParameters(x)
```

---

PerAndSpeed

*Persistence and Speed*

---

## Description

The `PerAndSpeed()` generates data and plots for persistence and speed.

**Usage**

```
PerAndSpeed(
  object,
  TimeInterval = 10,
  PtSplot = TRUE,
  AllPtSplot = TRUE,
  ApSplot = TRUE,
  AllApSplot = TRUE,
  export = FALSE,
  ExpName = NULL
)
```

**Arguments**

object	CellMig class object, which is a list of data frames resulted from the PreProcessing.
TimeInterval	A numeric value of the time elapsed between successive frames in the time-lapse stack.
PtSplot	A logical vector that allows generating individual plots of persistence time vs speed per cell. Default is TRUE.
AllPtSplot	A logical vector that allows generating a plot of persistence time vs speed for all cells. Default is TRUE.
ApSplot	A logical vector that allows generating individual plots of angular persistence vs speed per cell. Default is TRUE.
AllApSplot	A logical vector that allows generating a plot of angular persistence vs speed of all cells. Default is TRUE.
export	if 'TRUE' (default), exports function output
ExpName	string, indicates the name of the experiment. Can be NULL

**Value**

An CellMig class object with a data frame and plots. The data frame is stored in the PerAanSpeedtable slot.

**Author(s)**

Salim Ghannoum <salim.ghannoum@medisin.uio.no>

**References**

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/)

**Examples**

```
rmTD <- get(data(preProcCellMig))
rmTD <- PerAndSpeed(rmTD,TimeInterval=10, export=FALSE)
```

plot3DAllTracks      *A 3D rose-plot of all cells*

---

### Description

Plotting the trajectory data of all cells in 3D.

### Usage

```
plot3DAllTracks(object, VS = 3, size = 2, interactive = TRUE)
```

### Arguments

object	CellMig class object, which is a list of data frames resulted from the PreProcessing.
VS	A numeric value of the vertical separator between cells.
size	A numeric value of the point's size.
interactive	logical, shall the 3D plot be generated in a interactive fashion

### Details

The 3D visualization shows centered trajectories where the starting point of each track is located at the origin of the coordinate system (X=0,Y=0).

### Value

A 3D rose-plot showing the tracks of all cells.

### Note

This function requires the rgl package to be installed on your system.

### Author(s)

Salim Ghannoum <salim.ghannoum@medisin.uio.no>

### References

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/)

### Examples

```
if (Sys.info()[["sysname"]] != "Darwin") {  
  # interactive shall be set to TRUE (default)  
  rmTD <- get(data(preProcCellMig))  
  plot3DAllTracks(rmTD, VS=3, size=2, interactive = FALSE)  
}
```

---

plot3DTracks            *A 3D rose-plot*

---

**Description**

Plotting the trajectory data of particular cells in 3D.

**Usage**

```
plot3DTracks(object, VS = 3, size = 2, cells, interactive = TRUE)
```

**Arguments**

object	CellMig class object, which is a list of data frames resulted from the PreProcessing.
VS	A numeric value of the vertical separator between cells.
size	A numeric value of the point's size.
cells	A numeric vector containing the cell's numbers to be plotted.
interactive	logical, shall a 3D plot built in an interactive way.

**Details**

The 3D visualization shows centered trajectories where the starting point of each track is located at the origin of the coordinate system (X=0,Y=0).

**Value**

A 3D rose-plot showing the tracks of particular cells.

**Note**

This function requires the rgl package to be installed on your system.

**Author(s)**

Salim Ghannoum <salim.ghannoum@medisin.uio.no>

**References**

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/)

**Examples**

```
if (Sys.info()[["sysname"]] != "Darwin") {  
  # interactive shall be set to TRUE (default)  
  rmTD <- get(data(preProcCellMig))  
  plot3DTracks(rmTD, VS=3, size=2, cells=seq(1,5,by=1), interactive = FALSE)  
}
```

---

plotAllTracks                      *A 2D rose-plot*

---

### Description

Plotting the trajectory data of all cells.

### Usage

```
plotAllTracks(
  object,
  Type = "l",
  FixedField = TRUE,
  export = FALSE,
  ExpName = NULL
)
```

### Arguments

object	CellMig class object, which is a list of data frames resulted from the PreProcessing.
Type	has to be one of the following: c("p", "l", "b", "o") "p": Points; "l": Lines; "b": Both; "o": Both "overplotted".
FixedField	logical(1) Allows generating a plot with fixed field 800um x 800um. Default is TRUE.
export	if 'TRUE' (default), exports plot to JPG file
ExpName	string, name of the experiment. Can be NULL

### Details

The visualization shows centered trajectories where the starting point of each track is located at the origin of the coordinate system (X=0,Y=0).

### Value

A 2D rose-plot showing the tracks of all cells.

### Author(s)

Salim Ghannoum <salim.ghannoum@medisin.uio.no>

### References

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/)



**Examples**

```
rmTD <- get(data(preProcCellMig))
plotAllTracks(object=rmTD, Type="l", FixedField=TRUE,export=FALSE)
```

---

plotSampleTracks      *A 2D rose-plot of sample cells*

---

**Description**

Plotting the trajectory data of some cells.

**Usage**

```
plotSampleTracks(
  object,
  Type = "l",
  celNum = 35,
  FixedField = TRUE,
  export = FALSE,
  ExpName = NULL
)
```

**Arguments**

object	CellMig class object, which is a list of data frames resulted from the PreProcessing.
Type	has to be one of the following: c("p", "l", "b", "o")
celNum	A numeric value showing the desired number of cells to be plotted.
FixedField	logical(1) Allows generating a plot with fixed field 800um x 800um. Default is TRUE.
export	if 'TRUE' (default), exports plot to JPG file "p": Points; "l": Lines; "b": Both; "o": Both "overplotted".
ExpName	string, name of the experiment. Can be NULL

**Details**

The visualization shows centered trajectories where the starting point of each track is located at the origin of the coordinate system (X=0,Y=0).

**Value**

A 2D rose-plot showing the tracks of sample cells selected randomly based on the desired number of cells selected by the user.

**Author(s)**

Salim Ghannoum <salim.ghannoum@medisin.uio.no>

**References**

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/)

**Examples**

```
preProcCellMig <- get(data(preProcCellMig))
plotSampleTracks(preProcCellMig, Type="l", FixedField=TRUE,
                 celNum=5, export=FALSE, ExpName = NULL)
```

---

PlotTracksSeparately *A graphical display of the track of each cell.*

---

**Description**

Plotting the trajectory data of each cell.

**Usage**

```
PlotTracksSeparately(
  object,
  Type = "l",
  FixedField = TRUE,
  export = FALSE,
  ExpName = NULL
)
```

**Arguments**

object	CellMig class object, which is a list of data frames resulted from the PreProcessing.
Type	has to be one of the following: [p, l, b, o] "p": Points "l": Lines "b": Both "o": Both "overplotted"
FixedField	logical(1) Allows generating individual plots with fixed field. Default is TRUE.
export	if 'TRUE' (default), exports plot to JPG file
ExpName	string, name of the experiment. Can be NULL

**Details**

The visualization shows centered trajectories where the starting point of each track is located at the origin of the coordinate system (X=0,Y=0).

**Value**

2D rose-plots of the cells' track Separately.

**Author(s)**

Salim Ghannoum <salim.ghannoum@medisin.uio.no>

**References**

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/)

**Examples**

```
rmTD <- get(data(preProcCellMig))
PlotTracksSeparately(rmTD,Type="b", FixedField=FALSE, export = FALSE)
```

---

rmPreProcessing	<i>Data preprocessing for random migration (RM)</i>
-----------------	---

---

**Description**

This function allows preprocessing of the trajectory data from random migration (RM) experiments.

**Usage**

```
rmPreProcessing(  
  object,  
  PixelSize = 1.24,  
  TimeInterval = 10,  
  FrameN = NULL,  
  ExpName = NULL  
)
```

**Arguments**

object	CellMig class object.
PixelSize	A numeric value of the physical size of a pixel. Default is 1.24.
TimeInterval	A numeric value of the time elapsed between successive frames in the time-lapse stack. Default is 10 min.
FrameN	A numeric value of the number of frames. Default is NULL
ExpName	string, name of the experiment. Can be NULL

**Value**

An CellMig class object with preprocessed data.

**Author(s)**

Salim Ghannoum <salim.ghannoum@medisin.uio.no>

**References**

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/)

**Examples**

```
TrajectoryDataset <- get(data(TrajectoryDataset))
rmDF=TrajectoryDataset[seq(1,40,by=1),]
rmTD <- CellMig(rmDF)
rmTD <- rmPreProcessing(rmTD, FrameN=30)
```

---

setAnalyticParams      *Method setAnalyticParams*

---

**Description**

Set Analytic Params of a trackedCells object.

**Usage**

```
setAnalyticParams(x, params)

## S4 method for signature 'trackedCells,list'
setAnalyticParams(x, params)
```

**Arguments**

x                    a trackedCells-class object  
params                a list including all params

**Value**

a trackedCells object

**Examples**

```
data("TrackCellsDataset")
setAnalyticParams(TrackCellsDataset, list())
```

---

setCellMigSlot	<i>Method setCellMigSlot</i>
----------------	------------------------------

---

**Description**

Set Data of a slot in a CellMig object.

**Usage**

```
setCellMigSlot(x, slot, value)

## S4 method for signature 'CellMig,character'
setCellMigSlot(x, slot, value)
```

**Arguments**

x	a CellMig-class object
slot	string pointing to the slot to be updated
value	ANY value to be written

**Value**

a CellMig object

**Examples**

```
data("TrajectoryDataset")
x <- CellMig(TrajectoryDataset)
setCellMigSlot(x, "cellpos", c(1, 2, 3))
```

---

setCellsMeta	<i>Set MetaData</i>
--------------	---------------------

---

**Description**

Write/Replace MetaData of a trackedCells object

**Usage**

```
setCellsMeta(tc_obj, experiment = NULL, condition = NULL, replicate = NULL)
```

**Arguments**

tc_obj	a trackedCells object
experiment	string, a label to describe the experiment (optional). Can be NULL
condition	string, a label to describe the experimental condition (optional). Can be NULL
replicate	string, a label to identify the replicate (optional). Can be NULL

**Value**

a list including three items: experiment name, condition label, and replicate ID.

**Author(s)**

Damiano Fantini, <damiano.fantini@gmail.com>

**References**

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/) <https://www.mathworks.com/matlabcentral/fileexchange/60349-fasttracks>

**Examples**

```
x0 <- get(data(TrackCellsDataset))
x0 <- setCellsMeta(x0, experiment = "my_exp_01", condition = "DMSO")
getCellsMeta(x0)
```

---

setCellTracks

*Method setCellTracks*

---

**Description**

Set Tracks of a trackedCells object.

**Usage**

```
setCellTracks(x, tracks)
```

```
## S4 method for signature 'trackedCells,matrix'
setCellTracks(x, tracks)
```

**Arguments**

x	a trackedCells-class object
tracks	a matrix including all cell tracks

**Value**

a trackedCells object

**Examples**

```
data("TrackCellsDataset")
setCellTracks(TrackCellsDataset, matrix())
```

---

setExpName	<i>Method setExpName</i>
------------	--------------------------

---

**Description**

Set Experiment Name of a CellMig object.

**Usage**

```
setExpName(x, ExpName)

## S4 method for signature 'CellMig,character'
setExpName(x, ExpName)
```

**Arguments**

x	a CellMig-class object
ExpName	string corresponding to the ExpName

**Value**

a CellMig object

**Examples**

```
data("TrajectoryDataset")
x <- CellMig(TrajectoryDataset)
setExpName(x, "My Fav Experiment")
```

setOptimizedParams     *Method setOptimizedParams*

---

**Description**

Set Optimized Params of a trackedCells object.

**Usage**

```
setOptimizedParams(x, auto_params, results)
```

```
## S4 method for signature 'trackedCells'  
setOptimizedParams(x, auto_params, results)
```

**Arguments**

x	a trackedCells-class object
auto_params	automatically selected parameters
results	optimization analysis results

**Value**

a trackedCells object

**Examples**

```
data("TrackCellsDataset")  
setOptimizedParams(  
  TrackCellsDataset,  
  auto_params = list(lnoise = 6, diameter = 20, threshold = 10),  
  results = list())
```

---

setProcessedImages     *Method setProcessedImages*

---

**Description**

Set Processed Images of a trackedCells object.

**Usage**

```
setProcessedImages(x, procImages)
```

```
## S4 method for signature 'trackedCells,list'  
setProcessedImages(x, procImages)
```



**Arguments**

x                    a trackedCells-class object  
procImages        a list including all metadata

**Value**

a trackedCells object

**Examples**

```
data("TrackCellsDataset")  
prc.im <- getProcessedImages(TrackCellsDataset)  
setProcessedImages(TrackCellsDataset, prc.im)
```

---

setProcessingStatus    *Method setProcessingStatus*

---

**Description**

Set Operation Status of a trackedCells object.

**Usage**

```
setProcessingStatus(x, slot, value)  
  
## S4 method for signature 'trackedCells,character,numeric'  
setProcessingStatus(x, slot, value)
```

**Arguments**

x                    a trackedCells-class object  
slot                string pointing to the slot to be updated  
value                numeric value to be written

**Value**

a trackedCells object

**Examples**

```
data("TrackCellsDataset")  
setProcessingStatus(TrackCellsDataset, slot="optimized_params", value=0)
```

---

setTrackedCellsMeta    *Method setTrackedCellsMeta*

---

**Description**

Set Metadata of a trackedCells object.

**Usage**

```
setTrackedCellsMeta(x, meta)

## S4 method for signature 'trackedCells,list'
setTrackedCellsMeta(x, meta)
```

**Arguments**

x	a trackedCells-class object
meta	a list including all metadata

**Value**

a trackedCells object

**Examples**

```
data("TrackCellsDataset")
meta <- getCellTrackMeta(TrackCellsDataset)
meta[["condition"]] <- "DEMO N.2"
setTrackedCellsMeta(TrackCellsDataset, meta = meta)
```

---

setTrackedCentroids    *Method setTrackedCentroids*

---

**Description**

Set Centroids of a trackedCells object.

**Usage**

```
setTrackedCentroids(x, centroids)

## S4 method for signature 'trackedCells,list'
setTrackedCentroids(x, centroids)
```

**Arguments**

x                    a trackedCells-class object  
centroids           a list including all metadata

**Value**

a trackedCells object

**Examples**

```
data("TrackCellsDataset")  
setTrackedCentroids(TrackCellsDataset, list())
```

---

setTrackedPositions    *Method setTrackedPositions*

---

**Description**

Set positions of a trackedCells object.

**Usage**

```
setTrackedPositions(x, positions)  
  
## S4 method for signature 'trackedCells,data.frame'  
setTrackedPositions(x, positions)
```

**Arguments**

x                    a trackedCells-class object  
positions            a data.frame including all positions

**Value**

a trackedCells object

**Examples**

```
data("TrackCellsDataset")  
setTrackedPositions(TrackCellsDataset, data.frame())
```

setTrackingStats      *Method setTrackingStats*

---

**Description**

Set Tracking Statistics of a trackedCells object.

**Usage**

```
setTrackingStats(x, population, cells)

## S4 method for signature 'trackedCells'
setTrackingStats(x, population, cells)
```

**Arguments**

x	a trackedCells-class object
population	population-level statistics
cells	cell-level statistics

**Value**

a trackedCells object

**Examples**

```
data("TrackCellsDataset")
cel.sts <- getCellsStats(TrackCellsDataset)
pop.sts <- getPopulationStats(TrackCellsDataset)
setTrackingStats(TrackCellsDataset, pop.sts, cel.sts)
```

---

trackedCells-class      *The trackedCells Class.*

---

**Description**

An S4 class to represent a set of cells whose movements were tracked over time.

**Usage**

```
## S4 method for signature 'trackedCells'
initialize(.Object, x)
```

**Arguments**

.Object            the trackedCells object being built  
x                    imported TIFF image data

**Value**

An S4-class object  
a trackedCells object

**Slots**

images is a list of imported images  
proc\_images is a list of processed images  
ops is a list keeping track of the operations executed on the object  
optimized is a list including results of the params auto-optimization (optional)  
centroids is a list of detected centroids  
positions is a data.frame of cell positions across stacks  
tracks is a numeric matrix of cell tracks  
params is a list of parameters used for the analysis  
stats is a list of stats computed for the cell tracks  
metadata is a list including labels about the image, and the experiment

**Author(s)**

Damiano Fantini <damiano.fantini@gmail.com>

---

TrajectoryDataset            *Trajectories of 350 cells*

---

**Description**

A dataset containing the coordinates and the ID of 350 cells from a dense random migration experiment

**Usage**

```
data(TrajectoryDataset)
```

**Format**

A data frame with 50216 rows and 4 columns

## Details

BT549 cell trajectories were computed using cellmigRation. Imaging experiments were performed as described by Ghannoum S et al (paper in preparation). Briefly, triple negative breast cancer BT549 cells were cultured in RPMI supplemented with 10 and 1 NuLight green lentivirus (Essen BioScience), and then sorted by fluorescence-activated cell sorting (FACS). GFP-positive cells were seeded at a 1:3 ratio with untransduced BT549 cells in 96-well image-lock plates (EssenBio) at a density of 1000 total cells per well. Once cells reached the desired density, they were scanned at ten-minute intervals over 24h using an Incucyte S3 Live-Cell microscope (EssenBio) at 10x magnification and a Basler Ace 1920-155um camera with CMOS sensor. TIFF images were imported and processed using the cellmigRation library.

## Examples

```
data(TrajectoryDataset)
```

---

VeAutoCor

*Velocity AutoCorrelation*

---

## Description

The VeAutoCor function automatically compute the changes in both speed and direction across several sequential time intervals.

## Usage

```
VeAutoCor(
  object,
  TimeInterval = 10,
  sLAG = 0.25,
  sPLOT = TRUE,
  aPLOT = TRUE,
  export = FALSE,
  ExpName = NULL
)
```

## Arguments

object	CellMig class object, which is a list of data frames resulted from the PreProcessing.
TimeInterval	A numeric value of the time elapsed between successive frames in the time-lapse stack.
sLAG	A numeric value to be used to get the number of lags for the slope fitting. Default is 0.25, which represents 25 percent of the steps.
sPLOT	A logical vector that allows generating individual plots showing the velocity across several sequential time intervals. Default is TRUE.

aPLOT	A logical vector that allows generating a plot showing the velocity across several sequential time intervals of all cells. Default is TRUE.
export	if 'TRUE' (default), exports function output to CSV file
ExpName	string, name of the experiment. Can be NULL

**Value**

Plots and a data frame, which contains six rows: "Cell Number", "Velocity AutoCorrelation (lag=1)", "2nd normalized Velocity AutoCorrelation", "Intercept of VA quadratic model", "Mean Velocity AutoCorrelation (all lags)", "Mean |Acceleration|" and "Average Speed".

**Author(s)**

Salim Ghannoum <salim.ghannoum@medisin.uio.no>

**References**

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/)

**Examples**

```
data(TrajectoryDataset)
rmDF=TrajectoryDataset[1:300,]
rmTD <- CellMig(rmDF)
rmTD <- rmPreProcessing(rmTD,FrameN=55)
rmTD <- VeAutoCor(rmTD, TimeInterval=10, sLAG=0.25, sPLOT=FALSE,
                  aPLOT=FALSE, export=FALSE)
```

---

visualizeCellTracks     *Visualize Cell Tracks originating at an Image Stack*

---

**Description**

Visualize Cell Tracks that originated at an Image Stack of interest

**Usage**

```
visualizeCellTracks(
  tc_obj,
  stack = 1,
  pnt.cex = 1.2,
  lwd = 1.6,
  col = "red2",
  col.untracked = "gray45",
  main = NULL
)
```

**Arguments**

<code>tc_obj</code>	a trackedCells object
<code>stack</code>	index of the stack
<code>pnt.cex</code>	cex of the point drawn around each cell
<code>lwd</code>	width of the lines visualizing cell tracks
<code>col</code>	color of the points and the tracks, e.g.: "red2"
<code>col.untracked</code>	color of the points that were not tracked further, e.g.: "gray45"
<code>main</code>	string used as plot title, can be NULL

**Value**

None

**Author(s)**

Damiano Fantini, <damiano.fantini@gmail.com>

**References**

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/) <https://www.mathworks.com/matlabcentral/fileexchange/60349-fasttracks>

**Examples**

```
x <- get(data(TrackCellsDataset))
visualizeCellTracks(tc_obj = x, stack = 2)
```

---

VisualizeStackCentroids

*Visualize Cells in an Image Stack*

---

**Description**

Visualize objects that were identified as cells in a given image stack

**Usage**

```
VisualizeStackCentroids(  
  tc_obj,  
  stack = 1,  
  pnt.cex = 1.2,  
  txt.cex = 0.9,  
  offset = 0.18,  
  main = NULL  
)
```



**Arguments**

tc_obj	a trackedCells object
stack	index of the image stack to use
pnt.cex	cex of the points drawn around cells
txt.cex	cex of the text used to annotate cells
offset	offset value for the annotation
main	string used for the plot title, can be NULL= NULL

**Value**

None

**Author(s)**

Damiano Fantini, <damiano.fantini@gmail.com>

**References**

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/) <https://www.mathworks.com/matlabcentral/fileexchange/60349-fasttracks>

**Examples**

```
# Representative output
x <- get(data(TrackCellsDataset))
VisualizeStackCentroids(tc_obj = x, stack = 2, pnt.cex = 5, offset = 1.3)
```

---

wsaPreProcessing

*Data preprocessing for wound scratch assay (WSA).*

---

**Description**

This function allows filtering of cells and preprocessing of the trajectory data from wound scratch assay (WSA) experiments.

**Usage**

```
wsaPreProcessing(
  object,
  PixelSize = 1.24,
  TimeInterval = 10,
  FrameN = NULL,
  imageH = 1500,
  woundH = 600,
```

```

    upperE = 400,
    lowerE = 1000,
    mar = 75,
    clearW = TRUE,
    ExpName = NULL
  )

```

### Arguments

object	CellMig class object.
PixelSize	A numeric value of the physical size of a pixel.
TimeInterval	A numeric value of the time elapsed between successive frames in the time-lapse stack.
FrameN	A numeric value of the number of frames. Default is NULL
imageH	A numeric value of the image height.
woundH	A numeric value of the image height.
upperE	A numeric value of the upper edge of the wound.
lowerE	A numeric value of the lower edge of the wound.
mar	A numeric value of the margin to be used to narrow the clearing zone inside the zone.
clearW	A logical vector that allows removing the cells within the wound. Default is TRUE.
ExpName	string, name of the experiment. Can be NULL

### Value

An CellMig class object with filtered, annotated and preprocessed data.

### Author(s)

Salim Ghannoum <salim.ghannoum@medisin.uio.no>

### References

[https://www.data-pulse.com/dev\\_site/cellmigration/](https://www.data-pulse.com/dev_site/cellmigration/)

### Examples

```

WSADataset <- get(data(WSADataset))
wasDF=WSADataset[seq(1,30,by=1),]
wsaTD <- CellMig(wasDF)
wsaTD <- wsaPreProcessing(wsaTD,FrameN=20)

```

# Index

## \* datasets

- TrajectoryDataset, [61](#)
- aggregateFR, [3](#)
- aggregateTrackedCells, [5](#)
- CellMig (CellMig-class), [6](#)
- CellMig-class, [6](#)
- CellMigPCA, [7](#)
- CellMigPCAclust, [8](#)
- CellMigPCAclustALL, [9](#)
- CellTracker, [11](#)
- ComputeTracksStats, [13](#)
- DiAutoCor, [14](#)
- DiRatio, [15](#)
- DiRatioPlot, [16](#)
- EstimateDiameterRange, [17](#)
- FilterTrackedCells, [18](#)
- FinRes, [19](#)
- FMI, [20](#)
- ForwardMigration, [21](#)
- getAvailableAggrMetrics, [22](#)
- getCellImages, [23](#)
- getCellImages, trackedCells-method (getCellImages), [23](#)
- getCellMigSlot, [24](#)
- getCellMigSlot, CellMig, character-method (getCellMigSlot), [24](#)
- getCellsMeta, [24](#)
- getCellsStats, [25](#)
- getCellTrackMeta, [26](#)
- getCellTrackMeta, trackedCells-method (getCellTrackMeta), [26](#)
- getCellTracks, [26](#)
- getCellTracks, trackedCells-method (getCellTracks), [26](#)
- getCellTrackStats, [27](#)
- getCellTrackStats, trackedCells-method (getCellTrackStats), [27](#)
- getDACTable, [28](#)
- getDiRatio, [29](#)
- getFMITable, [30](#)
- getForMigtable, [31](#)
- getImageCentroids, [32](#)
- getImageCentroids, trackedCells-method (getImageCentroids), [32](#)
- getImageStacks, [32](#)
- getMSDtable, [33](#)
- getOptimizedParameters, [34](#)
- getOptimizedParams, [35](#)
- getOptimizedParams, trackedCells-method (getOptimizedParams), [35](#)
- getPerAndSpeed, [35](#)
- getPopulationStats, [36](#)
- getProcessedImages, [37](#)
- getProcessedImages, trackedCells-method (getProcessedImages), [37](#)
- getProcessingStatus, [37](#)
- getProcessingStatus, trackedCells-method (getProcessingStatus), [37](#)
- getResults, [38](#)
- getTracks, [39](#)
- getVACTable, [39](#)
- initialize, CellMig-method (CellMig-class), [6](#)
- initialize, trackedCells-method (trackedCells-class), [60](#)
- LoadTiff, [40](#)
- MSD, [41](#)
- OptimizeParams, [43](#)
- PerAndSpeed, [44](#)
- plot3DAllTracks, [46](#)
- plot3DTracks, [47](#)

plotAllTracks, 48  
plotSampleTracks, 49  
PlotTracksSeparately, 50  
rmPreProcessing, 51  
setAnalyticParams, 52  
setAnalyticParams, trackedCells, list-method  
    (setAnalyticParams), 52  
setCellMigSlot, 53  
setCellMigSlot, CellMig, character, ANY-method  
    (setCellMigSlot), 53  
setCellMigSlot, CellMig, character-method  
    (setCellMigSlot), 53  
setCellsMeta, 53  
setCellTracks, 54  
setCellTracks, trackedCells, matrix-method  
    (setCellTracks), 54  
setExpName, 55  
setExpName, CellMig, character-method  
    (setExpName), 55  
setOptimizedParams, 56  
setOptimizedParams, trackedCells, ANY, ANY-method  
    (setOptimizedParams), 56  
setOptimizedParams, trackedCells-method  
    (setOptimizedParams), 56  
setProcessedImages, 56  
setProcessedImages, trackedCells, list-method  
    (setProcessedImages), 56  
setProcessingStatus, 57  
setProcessingStatus, trackedCells, character, numeric-method  
    (setProcessingStatus), 57  
setTrackedCellsMeta, 58  
setTrackedCellsMeta, trackedCells, list-method  
    (setTrackedCellsMeta), 58  
setTrackedCentroids, 58  
setTrackedCentroids, trackedCells, list-method  
    (setTrackedCentroids), 58  
setTrackedPositions, 59  
setTrackedPositions, trackedCells, data.frame-method  
    (setTrackedPositions), 59  
setTrackingStats, 60  
setTrackingStats, trackedCells, ANY, ANY-method  
    (setTrackingStats), 60  
setTrackingStats, trackedCells-method  
    (setTrackingStats), 60  
trackedCells (trackedCells-class), 60  
trackedCells-class, 60  
TrajectoryDataset, 61  
VeAutoCor, 62  
visualizeCellTracks, 63  
VisualizeStackCentroids, 64  
wsaPreProcessing, 65