

Package ‘cellmigRation’

December 19, 2025

Type Package

Title Track Cells, Analyze Cell Trajectories and Compute Migration Statistics

Version 1.19.0

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 devel

git_last_commit 18cb9ed

git_last_commit_date 2025-10-29

Repository Bioconductor 3.23

Date/Publication 2025-12-19

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>

Contents

cellmigRation-package	5
AddDimension	5
aggregateFR	6
aggregateTrackedCells	7
bpass	8
CellMig-class	10
CellMigPCA	11
CellMigPCAclust	12
CellMigPCAclustALL	13
CellTracker	14
CellTrackerMainLoop	16
CentroidArray	17
CentroidValidation	18
circshift	19
cntrd	20
ComputeTracksStats	21
DetectRadii	22
DiAutoCor	23
DiRatio	24
DiRatioPlot	25
EstimateDiameterRange	26
FianlizeOptiParams	27
FilterTrackedCells	28
FinRes	29
fixDA	30
fixExpName	31
fixFM1	31
fixFM2	32
fixFM3	33
fixFM4	34
fixFM5	35
fixFM6	36
fixID1	37
fixID2	38
fixID3	38
fixID4	39
fixID5	40
fixID6	40
fixMSD	41

fixPER1	42
fixPER2	43
fixPER3	44
FMI	45
ForwardMigration	46
GenAllCombos	47
getAvailableAggrMetrics	48
getCellImages	49
getCellMigSlot	50
getCellsMeta	50
getCellsStats	51
getCellTrackMeta	52
getCellTracks	52
getCellTrackStats	53
getDACTable	54
getDiRatio	55
getFMitable	56
getForMigtable	57
getImageCentroids	58
getImageStacks	58
getMSDtable	59
getOptimizedParameters	60
getOptimizedParams	61
getPerAndSpeed	61
getPopulationStats	62
getProcessedImages	63
getProcessingStatus	63
getResults	64
getTracks	65
getVACTable	65
initializeTrackParams	66
innerBondRaster	67
internalPermutation	68
LinearConv2	68
LoadTiff	69
MakeHypercube	70
matfix	71
MigrationStats	72
MSD	73
NextOdd	74
NonParallel4OptimizeParams	75
NonParallelTrackLoop	76
nontrivialBondTracking	76
OptimizeParams	77
OptimizeParamsMainLoop	79
Parallel4OptimizeParams	80
ParallelTrackLoop	81
PerAndSpeed	81

pkfnd	83
plot3DAllTracks	84
plot3DTracks	85
plotAllTracks	86
plotSampleTracks	87
PlotTracksSeparately	88
PostProcessTracking	89
Prep4OptimizeParams	90
preProcCellMig	91
rmPreProcessing	92
runTrackingPermutation	93
setAnalyticParams	94
setCellMigSlot	94
setCellsMeta	95
setCellTracks	96
setExpName	97
setOptimizedParams	97
setProcessedImages	98
setProcessingStatus	99
setTrackedCellsMeta	99
setTrackedCentroids	100
setTrackedPositions	101
setTrackingStats	101
sinkAway	102
subNetworkTracking	103
ThreeConditions	103
track	104
TrackCellsDataset	105
trackedCells-class	106
trackHypercubeBuild	107
trackSlideProcessing	107
trackSlideWrapUp	108
TrajectoryDataset	109
trivialBondRaster	109
trivialBondTracking	110
ValidateTrackingArgs	111
VeAutoCor	112
visualizeCellTracks	113
VisualizeCntr	114
VisualizeImg	115
VisualizeStackCentroids	116
visualizeTrcks	117
warnMessage	118
WSADataset	119
wsaPreProcessing	119

cellmigRation-package *Track And Analyze Cell Migrations*

Description

Track Fluorescent Cells and Analyze their movements. Compute Migration Statistics and advanced metrics to understand motility and movement characteristics of a population of cells.

Author(s)

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

See Also

Useful links:

- <https://github.com/ocbe-uio/cellmigRation/>
- Report bugs at <https://github.com/ocbe-uio/cellmigRation/issues>

AddDimension *Add Dimension to a Molten Data Frame*

Description

Creates a new (molten) data matrix where all elements of y are added to each row of x. Each row in x is recycled for each element in y. Elements in y are added as the first column in the returned matrix.

Usage

```
AddDimension(x, y)
```

Arguments

x	a matrix or data.frame with at least 1 row and 1 column.
y	a vector with elements that will be added to x

Value

a matrix with an extra column as compared to x

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/

Examples

```
cellmigRation::AddDimension(
  x = cbind(seq(1,4,by=1), seq(4,1,by=-1)),
  y = c(9, 7))
```

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/

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

<code>x</code>	a trackedCells-class object where cells have already been tracked
<code>...</code>	one or more trackedCells-class object(s) where cells have already been tracked
<code>meta_id_field</code>	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/

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)
```

bpass

Perform a bandpass by convolving with an appropriate kernel

Description

Implements a real-space bandpass filter that suppresses pixel noise and long-wavelength image variations while retaining information of a characteristic size. First, a lowpassed image is produced by convolving the original with a gaussian. Next, a second lowpassed image is produced by convolving the original with a boxcar function. By subtracting the boxcar version from the gaussian version, we are using the boxcar version to perform a highpass. This code 'bpass.pro' is copyright 1997, John C. Crocker and David G. Grier. It should be considered 'freeware'- and may be distributed freely in its original form when properly attributed.

Usage

```
bpass(image_array, lnoise, lobject = NULL, threshold)
```

Arguments

image_array	Numeric matrix corresponding to the image to be filtered
lnoise	Characteristic lengthscale of noise in pixels.
lobject	Integer length in pixels somewhat larger than a typical object
threshold	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

Numeric matrix corresponding to the filtered image

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/ <https://www.mathworks.com/matlabcentral/fileexchange/60349-fasttracks>

Examples

```
x0 <- cbind(
  c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
  c(0, 0, 0, 0, 0, 1, 1, 4, 2, 0, 0, 0, 0, 0, 0),
  c(0, 0, 0, 0, 1, 2, 6, 5, 3, 0, 0, 0, 1, 0, 0),
  c(0, 0, 0, 0, 5, 5, 6, 8, 6, 1, 0, 0, 6, 2, 0),
  c(0, 0, 2, 5, 8, 7, 3, 5, 1, 0, 0, 0, 6, 2, 0),
  c(0, 0, 1, 5, 8, 7, 4, 5, 2, 0, 0, 0, 0, 0, 0),
  c(0, 0, 0, 5, 8, 7, 4, 5, 2, 0, 0, 0, 0, 0, 0),
  c(0, 0, 0, 1, 4, 5, 2, 4, 0, 0, 0, 0, 0, 0, 0),
  c(0, 0, 0, 0, 2, 3, 2, 1, 0, 0, 0, 0, 0, 0, 0),
  c(0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0),
  c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0),
  c(9, 9, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 2, 1),
  c(2, 9, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 2, 1),
  c(0, 2, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0),
  c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
  c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0))
y0 <- cellmigRation::bpass(x0, lnoise = 1, lobject = 5, threshold = 1)
par(mfrow = c(1, 2))
image(x0); title("original")
image(y0); title("after bpass")
```

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/

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)
PCAp1ot<-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/

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	<i>PCA Clusters of different conditions</i>
--------------------	---

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/

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 <code>trackedCells</code> object.
<code>import_optiParam_from</code>	a <code>trackedCells</code> object (optional) used to import optimized parameters; can be <code>NULL</code> .
<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 <code>NULL</code> if <code>OptimizeParams()</code> has already been run
<code>diameter</code>	numeric, diameter parameter; can be <code>NULL</code> if <code>OptimizeParams()</code> has already been run
<code>threshold</code>	numeric, threshold parameter; can be <code>NULL</code> if <code>OptimizeParams()</code> has already been run
<code>maxDisp</code>	numeric, maximum displacement of a cell per time interval. When many cells are detected in each frame, small <code>maxDisp</code> values should be used.
<code>memory_b</code>	numeric, <code>memory_b</code> parameter as used in the original <code>track.m</code> function. In the current R implementation, only the value <code>memory_b=0</code> is accepted
<code>goodenough</code>	numeric, <code>goodenough</code> parameter as used in the original <code>track.m</code> function. In the current R implementation, only the value <code>goodenough=0</code> is accepted
<code>threads</code>	integer, number of cores to use for parallelization
<code>show_plots</code>	logical, shall cells detected in each frame of the image stack be visualized
<code>verbose</code>	logical, shall info about the progress of the cell tracking job be printed
<code>dryrun</code>	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.mathworks.com/matlabcentral/fileexchange/60349-fasttracks>

Examples

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

CellTrackerMainLoop	<i>Cell Tracker Main Loop</i>
---------------------	-------------------------------

Description

Tool for Cell Tracker Main Loop

Usage

```
CellTrackerMainLoop(  
  FinalImage,  
  threads,  
  tc_obj,  
  min_frames_per_cell,  
  track_params  
)
```

Arguments

- FinalImage list of numeric matrices
- threads numeric, number of cores to use
- tc_obj trackingCell object
- min_frames_per_cell
 numeric, minimum number of frames per cell
- track_params a list of tracking parameters

Details

This is an internal function supporting the CellTracker function.

Value

list of processed data

Examples

```
cellmigRation:::CellTrackerMainLoop(list(1), 1, 1, list(1))
```

CentroidArray	<i>Build a Centroid Array</i>
---------------	-------------------------------

Description

Create an array containing centroid data for particles identified in each frame of the imported TIFF image stack

Usage

```
CentroidArray(stack, lobject, threshold, dryrun = FALSE)
```

Arguments

stack	3D matrix loaded to workspace from .tif stack
lobject	Integer length in pixels somewhat larger than a typical object
threshold	the minimum brightness of a pixel that might be local maxima
dryrun	logical, shall the execution be skipped

Value

data.frame of centroids, with 4 columns corresponding to x-position of centroid, y-postion of centroid, brightness, and square of the radius of gyration

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/ <https://www.mathworks.com/matlabcentral/fileexchange/60349-fasttracks>

Examples

```
# by default, the dryrun argument is set to FALSE
df <- get(data(TrackCellsDataset))
x0 <- getCellImages(df)
y0 <- cellmigRation::CentroidArray(x0, 16, 10, TRUE)
y0
```

CentroidValidation	<i>Validate Centroids</i>
--------------------	---------------------------

Description

Validate parameters used to identify cells in a image stack. A figure containing current image frame with identified particles labeled with circles and numerical tags is generated. This function is included for consistency and compatibility reasons with the original fastTracks software (Matlab). Also, consider using VisualizeStackCentroids() or visualizeCellTracks() instead.

Usage

```
CentroidValidation(
  stack,
  slice,
  lobject,
  threshold,
```

```

    pnt.cex = 1.2,
    txt.cex = 0.85,
    offset = 0.18
  )

```

Arguments

stack	stack of images to be evaluated
slice	index of the frame within the stack to be evaluated
lobject	integer, length in pixels somewhat larger than a typical object (cell)
threshold	the minimum brightness of a pixel that might be local maxima. NOTE: Make it big and the code runs faster but you might miss some particles. Make it small and you'll get everything and it'll be slow.
pnt.cex	cex of the circle drawn around each cell
txt.cex	cex of the text used for annotating cells
offset	offset used for annotating cells

Value

data.frame of centroid positions

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/ <https://www.mathworks.com/matlabcentral/fileexchange/60349-fasttracks>

Examples

```

x <- get(data(TrackCellsDataset))
x <- getCellImages(x)
x$images[[1]] <- x$images[[1]][seq(110,160,by=1), seq(100,160,by=1)]
cellmigRation::CentroidValidation(x, slice = 1, lobject =10, threshold = 5)

```

cirshift

Shift Array Circularly

Description

Circularly shift the elements in an array by a user-defined number of positions. This emulates the behavior of the corresponding Matlab Circhsift function.

Usage

```
circshift(x, n = 1)
```

Arguments

x a character, numeric, or logical vector with at least $n + 1$ elements
 n an integer corresponding to the number of positions for the shift

Value

a vector corresponding to x (same size, same class), whose elements have been shifted

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/

Examples

```
cellmigRation:::circshift(seq(1,10,by=1), -2)
```

cntrd	<i>Calculates Centroids</i>
-------	-----------------------------

Description

Calculates the centroid of bright spots to sub-pixel accuracy. Inspired by Grier & Crocker's feature for IDL, but greatly simplified and optimized for MATLAB, and then further ported to R. CREATED: Eric R. Dufresne, Yale University, Feb 4 2005.

Usage

```
cntrd(im, mx, sz, interactive = NULL)
```

Arguments

im numeric matrix corresponding to the image to process
 mx location of local maxima to pixel-levels accuracy
 sz diameter of the window over which to average to calculate the centroid. should be big enough.
 interactive numeric; if set to 1 (or any positive number), an image showing the computed centroids will be visualized

Value

a data.frame with 4 columns, containing, x, y, brightness, and the square of the radius of gyration for each cell.

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/ <https://www.mathworks.com/matlabcentral/fileexchange/60349-fasttracks>

Examples

```
x0 <- getCellImages(get(data(TrackCellsDataset)))
x0 <- x0$images[[1]][seq(80,150,by=1), seq(80,150,by=1)]
b <- cellmigRation::bpass(image_array = x0, lnoise = 2,
                          lobject = 15, threshold = 1)
pk <- cellmigRation::pkfnd(b, th = 2, sz = 5)
cnt <- cellmigRation::cntrd(im = b, mx = pk, sz = 5)
cnt
```

ComputeTracksStats	<i>Compute Tracks Stats</i>
--------------------	-----------------------------

Description

Wrapper for the MigrationStats() function. It computes statistics for a trackedCells object where cells have already been tracked.

Usage

```
ComputeTracksStats(tc_obj, time_between_frames, resolution_pixel_per_micron)
```

Arguments

tc_obj	a trackedCells object
time_between_frames	integer, time interval between two successive frames were taken
resolution_pixel_per_micron	integer, image resolution, i.e. number of pixels per micron

Value

a trackedCells object, including cell track statistics

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/ <https://www.mathworks.com/matlabcentral/fileexchange/60349-fasttracks>

Examples

```
x <- get(data(TrackCellsDataset))
x <- ComputeTracksStats(x, time_between_frames = 10,
                        resolution_pixel_per_micron = 20)
getCellsStats(x)
```

DetectRadii

Detect Linear Particle Diameters

Description

Estimates the diameters of particles in a numeric or logical vector

Usage

```
DetectRadii(x)
```

Arguments

x numeric or logical vector

Value

data.frame including two columns: MPOS indicates the centroid position of a particle, and LEN indicates the diameter size

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/

Examples

```
cellmigRation:::DetectRadii(c(0,0,1,1,0,1,1,1,1,0,0, 1,0,0,1,1))
```

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/

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

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

```
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/

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/

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

<code>x</code>	numeric matrix corresponding to a digital image
<code>px.margin</code>	integer, number of pixels used as margin while searching/filtering for neighboring particles
<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>quantile.val</code>	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
<code>plot</code>	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/

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)
```

FianlizeOptiParams	<i>Finalize Output in Parameter Optimization</i>
--------------------	--

Description

Finalize Output as part of the Optimization Parameter process

Usage

```
FianlizeOptiParams(all_results, all_params, target_cell_num, plot)
```

Arguments

all_results	list, including all intermediates
all_params	data.frame, including all parameter combinations to test
target_cell_num	numeric, number of expected cells
plot	logical, shall a series of plots be generated

Details

This is an internal function supporting the Optimization Parameter process

Value

a list including test results; an empty list is returned if an error is encountered.

Examples

```
cellmigRation:::FianlizeOptiParams(list(1), data.frame(1), 10, FALSE)
```

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/

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/

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)
```

fixDA

*Direction AutoCorrelation***Description**

This function is a part of the DiAutoCor function, which computes the angular persistence across several sequential time intervals.

Usage

```
fixDA(Object, Step, sLAG, sPLOT, aPLOT, color, export, ExpName, new.fld)
```

Arguments

Step	A numeric value of the number of trajectory steps.
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.
color	A vector of colors that will be used for the plots
export	if 'TRUE' (default), exports function output
ExpName	String, name of the experiment
new.fld	path to the folder where to save files
object	CellMig class object, which is a list of data frames resulted from the PreProcessing.

Value

A data frame named "DA.ResultsTable".

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/

Examples

```
cellmigRation:::fixDA(1, 1, 1, 1, 1, 1, 1, 1, 1)
```

fixExpName	<i>Handle non-NULL ExpName</i>
------------	--------------------------------

Description

The fixExpName helps adjusting the name of the experiment in case it is not NULL.

Usage

```
fixExpName(x)
```

Arguments

x string, name of the experiment.

Value

A string referring to the adjusted experiment name.

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/

Examples

```
cellmigRation:::fixExpName("Hello World")
```

fixFM1	<i>Forward Migration First Part</i>
--------	-------------------------------------

Description

This function is a part of the ForwardMigration function, which generates data and plots for forward persistence and speed.

Usage

```
fixFM1(Object, Step)
```

Arguments

Step	A numeric value of the number of trajectory steps.
object	CellMig class object, which is a list of data frames resulted from the PreProcessing.

Value

An CellMig class Object.

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/

Examples

```
cellmigRation:::fixFM1(1, 1)
```

fixFM2

Forward Migration Second Part

Description

This function is a part of the ForwardMigration function, which generates data and plots for forward persistence and speed.

Usage

```
fixFM2(Object, Step)
```

Arguments

Step	A numeric value of the number of trajectory steps.
object	CellMig class object, which is a list of data frames resulted from the PreProcessing.

Value

An CellMig class Object.

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/

Examples

```
cellmigRation:::fixFM2(1, 1)
```

fixFM3

Forward Migration Third Part

Description

This function is a part of the ForwardMigration function, which generates data and plots for forward persistence and speed.

Usage

```
fixFM3(Object, Step)
```

Arguments

Step	A numeric value of the number of trajectory steps.
object	CellMig class object, which is a list of data frames resulted from the PreProcessing.

Value

A data frame named "cosine.FP".

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/

Examples

```
cellmigRation:::fixFM3(1, 1)
```

`fixFM4`*Forward Migration Fourth Part*

Description

This function is a part of the ForwardMigration function, which generates data and plots for forward persistence and speed.

Usage

```
fixFM4(Object, TimeInterval, Step)
```

Arguments

TimeInterval	A numeric value of the time elapsed between successive frames in the time-lapse stack.
Step	A numeric value of the number of trajectory steps.
object	CellMig class object, which is a list of data frames resulted from the PreProcessing.

Value

An CellMig class Object.

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/

Examples

```
cellmigRation:::fixFM4(1, 1, 1)
```

fixFM5*Forward Migration Fifth Part*

Description

This function is a part of the ForwardMigration function, which generates data and plots for forward persistence and speed.

Usage

```
fixFM5(Object, TimeInterval, Step)
```

Arguments

TimeInterval	A numeric value of the time elapsed between successive frames in the time-lapse stack.
Step	A numeric value of the number of trajectory steps.
object	CellMig class object, which is a list of data frames resulted from the PreProcessing.

Value

A data frame named "FMResultsTable".

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/

Examples

```
cellmigRation:::fixFM5(1, 1, 1)
```

fixFM6

*Forward Migration Sixst Part***Description**

This function is a part of the ForwardMigration function, which generates data and plots for forward persistence and speed.

Usage

```
fixFM6(
  Object,
  FMResultsTable,
  Step,
  sfptPLOT,
  afptPLOT,
  export,
  color,
  TimeInterval,
  sfpPLOT,
  ExpName,
  new.fld
)
```

Arguments

FMResultsTable	A data frame resulted from the fixFM6().
Step	A numeric value of the number of trajectory steps.
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.
export	if 'TRUE' (default), exports function output to CSV file
color	A vector of colors that will be used for the plots
TimeInterval	A numeric value of the time elapsed between successive frames in the time-lapse stack.
sfpPLOT	A logical vector that allows generating individual plots of angular persistence vs speed per cell. Default is TRUE.
ExpName	String, name of the experiment
new.fld	path to the folder where to save files
object	CellMig class object, which is a list of data frames resulted from the PreProcessing.

Value

A data frame named "FMResultsTable".

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/

Examples

```
cellmigRation:::fixFM6(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
```

fixID1

Pre-processing First Part

Description

This function prepare the data in each data frame as a part of the pre-processing.

Usage

```
fixID1(ID_split, TimeInterval)
```

Arguments

ID_split A list of data frames.

TimeInterval A numeric value of the time elapsed between

Value

A list of data frames.

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/

Examples

```
cellmigRation:::fixID1(ID_split = list(data.frame()), TimeInterval = 1)
```

`fixID2`*Pre-processing Second Part*

Description

This function prepare the data in each data frame as a part of the pre-processing.

Usage

```
fixID2(ID_split)
```

Arguments

`ID_split` A list of data frames.
`TimeInterval` A numeric value of the time elapsed between

Value

A list of data frames.

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/

`fixID3`*Pre-processing Third Part*

Description

This function prepare the data in each data frame as a part of the pre-processing.

Usage

```
fixID3(ID_split)
```

Arguments

`ID_split` A list of data frames.
`TimeInterval` A numeric value of the time elapsed between

Value

A list of data frames.

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/

fixID4

Pre-processing Fourth Part

Description

This function prepare the data in each data frame as a part of the pre-processing.

Usage

```
fixID4(ID_split)
```

Arguments

ID_split A list of data frames.

TimeInterval A numeric value of the time elapsed between

Value

A list of data frames.

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/

`fixID5`*Pre-processing Fifth Part*

Description

This function prepare the data in each data frame as a part of the pre-processing.

Usage

```
fixID5(ID_split, TimeInterval)
```

Arguments

<code>ID_split</code>	A list of data frames.
<code>TimeInterval</code>	A numeric value of the time elapsed between

Value

A list of data frames.

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/

`fixID6`*Pre-processing Sixst Part*

Description

This function prepare the data in each data frame as a part of the pre-processing.

Usage

```
fixID6(ID_split, TimeInterval)
```

Arguments

<code>ID_split</code>	A list of data frames.
<code>TimeInterval</code>	A numeric value of the time elapsed between

Value

A list of data frames.

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/

fixMSD	<i>Mean Square Displacement</i>
--------	---------------------------------

Description

This function is a part of the MSD function, which computes the mean square displacements across several sequential time intervals.

Usage

```
fixMSD(
  Object,
  Step,
  SlopePlot,
  AllSlopesPlot,
  FurthPlot,
  AllFurthPlot,
  sLAG,
  ffLAG,
  color,
  export,
  ExpName,
  new.fld
)
```

Arguments

Step	A numeric value of the number of trajectory 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.
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.
color	A vector of colors that will be used for the plots
export	if 'TRUE' (default), exports function output
ExpName	String, name of the experiment
new.fld	path to the folder where to save files
object	CellMig class object, which is a list of data frames resulted from the PreProcessing.

Value

A data frame named "MSDResultsTable".

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/

Examples

```
cellmigRation:::fixMSD(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
```

fixPER1

Persistence and Speed First Part

Description

This function is a part of the PerAndSpeed(), which generates data and plots for persistence and speed.

Usage

```
fixPER1(Object, TimeInterval)
```

Arguments

TimeInterval	A numeric value of the time elapsed between
x	CellMig class object, which is a list of data

Value

A data frame named "PerResultsTable".

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/

Examples

```
cellmigRation:::fixPER1(1, 1)
```

fixPER2

Persistence and Speed Second Part

Description

This function is a part of the PerAndSpeed(), which generates data and plots for persistence and speed.

Usage

```
fixPER2(  
  Object,  
  PerResultsTable,  
  PtSplot,  
  AllPtSplot,  
  export,  
  color,  
  TimeInterval,  
  ExpName,  
  new.fld  
)
```

Arguments

TimeInterval	A numeric value of the time elapsed between
ExpName	String, name of the experiment
new.fld	path to the folder where to save files
x	CellMig class object, which is a list of data

Value

A data frame named "PerResultsTable".

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/

Examples

```
cellmigRation:::fixPER2(1, 1, 1, 1, 1, 1, 1, 1, 1)
```

fixPER3

Persistence and Speed Third Part

Description

This function is a part of the PerAndSpeed(), which generates data and plots for persistence and speed.

Usage

```
fixPER3(
  Object,
  PerResultsTable,
  ApSplot,
  AllApSplot,
  export,
  color,
  TimeInterval,
  ExpName,
  new.fld
)
```

Arguments

Object	CellMig class object, which is a list of data.
PerResultsTable	A data frame.
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.
color	A vector of colors that will be used for the plots
TimeInterval	A numeric value of the time elapsed between
ExpName	String, name of the experiment
new.fld	path to the folder where to save files

Value

A data frame named "PerResultsTable".

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/

Examples

```
cellmigRation:::fixPER3(1,1,1,1,1,1,1,1,1)
```

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/

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/

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)
```

GenAllCombos

Generate All Combinations

Description

Generate All Combinations as part of the Optimization Parameter process

Usage

```
GenAllCombos(...)
```

Arguments

... a series of arguments where each argument is a vector of values to be combined together

Details

This is an internal function supporting the Optimization Parameter steps

Value

a data frame of combined parameters to be tested

Examples

```
cellmigRation:::GenAllCombos(A=c(1,2,3), B = 10, C = c("x", "y", "z"))
```

`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/

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

*Method getCellImages***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.mathworks.com/matlabcentral/fileexchange/60349-fasttracks>

Examples

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

getCellsStats	<i>Get Cell migration stats</i>
---------------	---------------------------------

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.mathworks.com/matlabcentral/fileexchange/60349-fasttracks>

Examples

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

getCellTrackMeta	<i>Method getCellTrackMeta</i>
------------------	--------------------------------

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	<i>Method getCellTracks</i>
---------------	-----------------------------

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	<i>Method getCellTrackStats</i>
-------------------	---------------------------------

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

<code>object</code>	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/

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/

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/

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/

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	<i>Method getImageCentroids</i>
-------------------	---------------------------------

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	<i>Get Image Stacks</i>
----------------	-------------------------

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.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/

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.mathworks.com/matlabcentral/fileexchange/60349-fasttracks>

Examples

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

getOptimizedParams	<i>Method getOptimizedParams</i>
--------------------	----------------------------------

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	<i>Getting the table of Persistence and Speed.</i>
----------------	--

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/

Examples

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

getPopulationStats	<i>Get Cell population stats</i>
--------------------	----------------------------------

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.mathworks.com/matlabcentral/fileexchange/60349-fasttracks>

Examples

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

getProcessedImages	<i>Method getProcessedImages</i>
--------------------	----------------------------------

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	<i>Method getProcessingStatus</i>
---------------------	-----------------------------------

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/

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`*Get Track Data*

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.mathworks.com/matlabcentral/fileexchange/60349-fasttracks>

Examples

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

`getVACtable`*Getting the Velocity AutoCorrelation*

Description

The VeAutoCor function automatically compute the changes in both speed and direction across several sequential 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/

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))
```

initializeTrackParams *Initialize Tracking parameters*

Description

Initialize parameter variables used for the tracking

Usage

```
initializeTrackParams(dd = 0, params)
```

Arguments

dd numeric, value of the dd param

params a list containing a few tracking parameters that are needed for the analysis

Value

a list including parsed arguments

Author(s)

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

Examples

```
cellmigRation::initializeTrackParams(0, NULL)
```

innerBondRaster	<i>Inner Bond Raster</i>
-----------------	--------------------------

Description

Perform Inner Bond Raster as part of the Cell Tracking Processing

Usage

```
innerBondRaster(xyzs, maxdisp, j = 1, env)
```

Arguments

xyzs	data.frame, including input cell centroid positions
maxdisp	numeric, value of maximum cell dispersion in pixels
env	environment, including all objects used for the tracking
i	numeric, index of the iteration cycle

Details

a message is printed if an issue (typically arising by a non-suitable environment being passed as the env argument) is detected. See the example below.

Value

FALSE is returned; objects in env are updated

Examples

```
cellmigRation::innerBondRaster(data.frame(1), 1, 1, new.env())
```

internalPermutation	<i>Internal Permutation</i>
---------------------	-----------------------------

Description

Perform Internal Permutation as part of the Cell Tracking Processing

Usage

```
internalPermutation(xyzs, maxdisp, env)
```

Arguments

xyzs	data.frame, including input cell centroid positions
maxdisp	numeric, value of maximum cell dispersion in pixels
env	environment, including all objects used for the tracking

Details

a message is printed if an issue (typically arising by a non-suitable environment being passed as the env argument) is detected. See the example below.

Value

FALSE is returned while objects in env are updated

Examples

```
cellmigRation:::internalPermutation(data.frame(1), 1, new.env())
```

LinearConv2	<i>Linear Convolution of a Numeric Matrix</i>
-------------	---

Description

Performs a linear convolution of a Numeric Matrix, using a user-supplied linear kernel. The convolution can be executed in a column-wise fashion by setting the col.wise argument to TRUE. Alternatively, the convolution is performed in a row-wise fashion.

Usage

```
LinearConv2(x, krnl, col.wise = TRUE)
```

Arguments

<code>x</code>	numeric matrix that will be used as input for the convolution; this matrix typically corresponds to an image where signal (high values) indicates the presence of a cell or a cell-like particle
<code>krnl</code>	numeric vector corresponding to the kernel that will be used for the convolution. Briefly, the kernel includes the weights that will be used to compute a weighted sum at each position of the input numeric matrix
<code>col.wise</code>	logical; shall the linear convolution be performed in a column-wise or row-wise fashion?

Value

Linearly convoluted numeric matrix. The resulting matrix has the same dimensions of the input matrix

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/

Examples

```
graphics::par(mfrow = c(1, 2))
tmp <- vapply(
  seq_len(12),
  function(i) {(6 + abs(i - 6)) * c(seq(1,10,by=1), seq(10,1,by=-1))},
  FUN.VALUE = numeric(20)
)
cnv.tmp <- cellmigRation::LinearConv2(tmp, c(-3, 0, 3))
graphics::image(tmp); graphics::image(cnv.tmp)
```

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.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")
```

MakeHypercube

Make Hypercube

Description

Creates a Molten Hypercube with a user-defined number of dimensions. The values supplied by the user are used to fill each dimension. All possible combination of values are included in the resulting hyper cube.

Usage

```
MakeHypercube(vals, dims)
```

Arguments

vals	vector of values used to fill the hyper cube
dims	integer indicating the number of dimensions. The resulting molten data frame will have a number of columns equal to dims

Value

Matrix corresponding to a molten hyper cube. The number of columns is equal to dims; the number of rows is equal to $\text{length}(\text{vals})^{\wedge} \text{dims}$

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/

Examples

```
cellmigRation:::MakeHypercube(seq(1,3,by=1), 3)
```

matfix

Clean And Reformat a Numeric Matrix

Description

Convert any matrix-like object to a numeric Matrix, and coerces all the elements to integer. Row names and column names are removed.

Usage

```
matfix(x)
```

Arguments

x	matrix or data.frame including numeric data (or data that can be coerced to integer)
---	--

Value

numeric matrix with all its elements coerced to integer

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/

Examples

```
tmp <- data.frame(A = c(1,2,3,4), B=c(3.1, 2.8, 3.3, 9.1), C = FALSE)
cellmigRation:::matfix(tmp)
```

MigrationStats

Compute Cell Migration Statistics

Description

Calculate the statistics from X/Y positional data obtained from cell tracks

Usage

```
MigrationStats(tracks, interval_time, pixel_micron)
```

Arguments

tracks data.frame with cell tracks information
interval_time integer, time interval between two successive frames were taken
pixel_micron integer, image resolution, i.e. number of pixels per micron

Value

list of stats calculated for the cell tracks. Info include variables of speed, distance, euclidean displacement, persistence, angular displacement, yFMI, xFMI, y-displacement, x-displacement and frames

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/ <https://www.mathworks.com/matlabcentral/fileexchange/60349-fasttracks>

Examples

```
x0 <- cbind(c(30, 35, 1, 5, 6, 7, 50, 55, 56, 58),
           c(29, 37, 2, 7, 4, 9, 40, 50, 59, 49),
           c( 1,  2, 1, 2, 3, 4,  1,  2,  3,  4),
           c( 1,  1, 2, 2, 2, 2,  3,  3,  3,  3))
cellmigRation:::MigrationStats(x0, 10, 10)
```

MSD	<i>Mean Square Displacement</i>
-----	---------------------------------

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/

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)
```

NextOdd

Return the Next Odd Integer

Description

Returns the smallest odd number bigger than the number(s) provided as the argument

Usage

```
NextOdd(x)
```

Arguments

x a vector of class numeric

Value

a vector of class integer

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/

Examples

```
cellmigRation:::NextOdd(seq(2,5,by=1))
```

```
NonParallel4OptimizeParams
```

Non Paralle Parameter Optimization

Description

Non Parallel Optimization as part of the Optimization Parameter process

Usage

```
NonParallel4OptimizeParams(  
  tmp_img = tmp_img,  
  all_params = all_params,  
  verbose = verbose  
)
```

Arguments

tmp_img	numeric matrix, corresponding to the
all_params	data.frame, including all parameter combinations to test
verbose	logical, shall info be printed to console

Details

This is an internal function supporting the Optimization Parameter process

Value

a list including test results; an empty list is returned if an error is encountered.

Examples

```
cellmigRation:::NonParallel4OptimizeParams(matrix(1), data.frame(1), TRUE)
```

NonParallelTrackLoop *Single Core Tracking Loop*

Description

Tool for Single Core Tracking Loop

Usage

NonParallelTrackLoop(FinalImage, min_frames_per_cell, track_params)

Arguments

FinalImage a list of numeric matrices (images)
 min_frames_per_cell numeric, minimum number of frames
 track_params a list of tracking parameters

Details

This is an internal function supporting the CellTracker function.

Value

list of processed data

Examples

```
cellmigRation:::NonParallelTrackLoop(list(), 1, list())
```

nontrivialBondTracking *Non-Trivial Bond Tracking*

Description

Perform Non-Trivial Bond Tracking as part of the Cell Tracking Processing

Usage

nontrivialBondTracking(xyzs, maxdisp, i, env)

Arguments

xyzs	data.frame, including input cell centroid positions
maxdisp	numeric, value of maximum cell dispersion in pixels
i	integer, index of the current cycle
env	environment, including all objects used for the tracking

Details

a message is printed if an issue (typically arising by a non-suitable environment being passed as the env argument) is detected. See the example below.

Value

FALSE is returned; objects in env are updated

Examples

```
cellmigRation:::nontrivialBondTracking(data.frame(1), 1, 1, new.env())
```

OptimizeParams	<i>Optimize Detection Params</i>
----------------	----------------------------------

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

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

OptimizeParamsMainLoop

Main Loop to Parameter Optimization

Description

Main Loop as part of the Optimization Parameter process

Usage

```
OptimizeParamsMainLoop(
  tmp_img = tmp_img,
  all_params = all_params,
  threads = threads,
  verbose = verbose
)
```

Arguments

tmp_img	numeric matrix, corresponding to the
all_params	data.frame, including all parameter combinations to test
threads	numeric, number of cores to use for the parallelization
verbose	logical, shall info be printed to console

Details

This is an internal function supporting the Optimization Parameter process

Value

a list including test results; an empty list is returned if an error is encountered.

Examples

```
cellmigRation:::OptimizeParamsMainLoop(matrix(1), data.frame(1), 1, FALSE)
```

Parallel4OptimizeParams

Paralle Parameter Optimization

Description

Parallel Optimization as part of the Optimization Parameter process

Usage

```
Parallel4OptimizeParams(  
  tmp_img = tmp_img,  
  all_params = all_params,  
  use.cores = use.cores,  
  verbose = verbose  
)
```

Arguments

tmp_img	numeric matrix, corresponding to the
all_params	data.frame, including all parameter combinations to test
use.cores	numeric, number of cores to use for the parallelization
verbose	logical, shall info be printed to console

Details

This is an internal function supporting the Optimization Parameter process

Value

a list including test results; an empty list is returned if an error is encountered.

Examples

```
cellmigRation:::Parallel4OptimizeParams(matrix(1), data.frame(1), 1, TRUE)
```

ParallelTrackLoop	<i>Multi Core Tracking Loop</i>
-------------------	---------------------------------

Description

Tool for Multi Core Tracking Loop

Usage

ParallelTrackLoop(FinalImage, use.cores, min_frames_per_cell, track_params)

Arguments

- FinalImage a list of numeric matrices (images)
- use.cores numeric, number of cores to use
- min_frames_per_cell
 numeric, minimum number of frames
- track_params a list of tracking parameters

Details

This is an internal function supporting the CellTracker function.

Value

list of processed data

Examples

cellmigRation::ParallelTrackLoop(list(), 1, 1, list())

PerAndSpeed	<i>Persistence and Speed</i>
-------------	------------------------------

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/

Examples

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

pkfnd

*Find Signal Peaks***Description**

Finds local maxima in an image to pixel level accuracy. This provides a rough guess of particle centers to be used by `cntrd()`. Inspired by the `lmx` subroutine of Grier and Crocker's. CREATED: Eric R. Dufresne, Yale University, Feb 4 2005.

Usage

```
pkfnd(im, th, sz = NULL)
```

Arguments

<code>im</code>	image to process, particle should be bright spots on dark background with little noise often a bandpass filtered brightfield image
<code>th</code>	the minimum brightness of a pixel that might be local maxima. NOTE: Make it big and the code runs faster but you might miss some particles. Make it small and you'll get everything and it'll be slow.
<code>sz</code>	if your data is noisy, (e.g. a single particle has multiple local maxima), then set this optional keyword to a value slightly larger than the diameter of your blob. If multiple peaks are found within a radius of $sz/2$ then the code will keep only the brightest. Also gets rid of all peaks within sz of boundary

Value

a numeric data.frame with two columns, with the coordinates of local maxima

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/ <https://www.mathworks.com/matlabcentral/fileexchange/60349-fasttracks>

Examples

```
x0 <- getCellImages(get(data(TrackCellsDataset)))
x0 <- x0$images[[1]][seq(80,150,by=1), seq(80,150,by=1)]
b <- cellmigRation::bpass(image_array = x0, lnoise = 2,
                          lobject = 15, threshold = 1)
pk <- cellmigRation::pkfnd(b, th = 2, sz = 5)
pk
```

plot3DAllTracks	<i>A 3D rose-plot of all cells</i>
-----------------	------------------------------------

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/

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	<i>A 3D rose-plot</i>
--------------	-----------------------

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/

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	<i>A 2D rose-plot</i>
---------------	-----------------------

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/

Examples

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

plotSampleTracks	<i>A 2D rose-plot of sample cells</i>
------------------	---------------------------------------

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/

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/

Examples

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

PostProcessTracking	<i>Post Processing Tracking Routine</i>
---------------------	---

Description

Tool for Post Processing Tracking Routine

Usage

```
PostProcessTracking(tc_obj, all_centroids, track_params)
```

Arguments

tc_obj	a trackedCells object
all_centroids	list, frame by frame centroids
track_params	a list of tracking parameters

Details

This is an internal function supporting the CellTracker function.

Value

list of processed data

Examples

```
cellmigRation:::PostProcessTracking(list(1), list(2), list(A=3))
```

```
Prep4OptimizeParams    Prepare For Parameter Optimization
```

Description

Pre-processing as part of the Optimization Parameter process

Usage

```
Prep4OptimizeParams(  
  stack_img,  
  lnoise_range = NULL,  
  min.px.diam = 5,  
  diameter_range = NULL,  
  threshold_range = NULL,  
  target_cell_num = NULL,  
  quantile.val = NULL,  
  px.margin = NULL,  
  plot = FALSE,  
  verbose = FALSE  
)
```

Arguments

stack_img	input image
lnoise_range	numeric, lnoise values to test
min.px.diam	numeric, minimum number of pixels of a cell signal
diameter_range	numeric, numeric values for diameter
threshold_range	numeric, numeric values for threshold
target_cell_num	numeric, target number of cells, can be NULL
quantile.val	numeric, the quantile prob used to suppress noise
px.margin	numeric, the frame margin size
plot	logical, shall a plot be printed
verbose	logical, shall info be printed to console

Details

This is an internal function supporting the Optimization Parameter steps

Value

a data frame of combined parameters to be tested

Examples

```
x <- get(data(TrackCellsDataset))
x <- cellmigRation::getCellImages(x = x)
y <- cellmigRation::Prep4OptimizeParams(stack_img = x)
y$success
```

preProcCellMig	<i>Trajectories of 11 cells</i>
----------------	---------------------------------

Description

Intermediates and Results from Cell Tracking Analyses, used as a representative example of a S4 CellMig object

Usage

```
data(preProcCellMig)
```

Format

a list including 21 elements

Details

BT549 cell trajectories were computed by 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 NucLight 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(preProcCellMig)
```

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/

Examples

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

`runTrackingPermutation`*Run Tracking Permutation*

Description

Perform Internal Permutation as part of the Cell Tracking Processing

Usage

```
runTrackingPermutation(xyzs, maxdisp, nc, i, env)
```

Arguments

<code>xyzs</code>	data.frame, including input cell centroid positions
<code>maxdisp</code>	numeric, value of maximum cell dispersion in pixels
<code>nc</code>	numeric, value of the nc parameter
<code>i</code>	integer, index of the current cycle
<code>env</code>	environment, including all objects used for the tracking

Details

a message is printed if an issue (typically arising by a non-suitable environment being passed as the `env` argument) is detected. See the example below.

Value

FALSE is returned while objects in `env` are updated

Examples

```
cellmigRation:::runTrackingPermutation(data.frame(1), 1, 1, 1, new.env())
```

setAnalyticParams	<i>Method setAnalyticParams</i>
-------------------	---------------------------------

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.mathworks.com/matlabcentral/fileexchange/60349-fasttracks>

Examples

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

setCellTracks	<i>Method setCellTracks</i>
---------------	-----------------------------

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	<i>Method setOptimizedParams</i>
--------------------	----------------------------------

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	<i>Method setProcessedImages</i>
--------------------	----------------------------------

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	<i>Method setProcessingStatus</i>
---------------------	-----------------------------------

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	<i>Method setTrackedCellsMeta</i>
---------------------	-----------------------------------

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	<i>Method setTrackedCentroids</i>
---------------------	-----------------------------------

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	<i>Method setTrackedPositions</i>
---------------------	-----------------------------------

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	<i>Method setTrackingStats</i>
------------------	--------------------------------

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)
```

sinkAway

Sinking Output as part of Parameter Optimization

Description

Tool for Sinking Output as part of the Optimization Parameter process

Usage

```
sinkAway(ty = "M", fl = "/dev/null")
```

Arguments

ty	character, "M" for messages, anything else for output
fl	filename to sink the output to

Details

This is an internal function supporting the Optimization Parameter process

Value

value returned by the sink() function

Examples

```
print(1)
cellmigRation:::sinkAway("0")
print(2)
cellmigRation:::sinkAway("0", NULL)
print(3)
```

subNetworkTracking	<i>Subnetwork Tracking</i>
--------------------	----------------------------

Description

Perform Internal Subnetwork Tracking as part of the Cell Tracking Processing

Usage

```
subNetworkTracking(xyzs, maxdisp, env)
```

Arguments

xyzs	data.frame, including input cell centroid positions
maxdisp	numeric, value of maximum cell dispersion in pixels
env	environment, including all objects used for the tracking

Details

a message is printed if an issue (typically arising by a non-suitable environment being passed as the env argument) is detected. See the example below.

Value

FALSE is returned while objects in env are updated

Examples

```
cellmigRation:::subNetworkTracking(data.frame(1), 1, new.env())
```

ThreeConditions	<i>Intermediates and Results from Cell Tracking Analyses</i>
-----------------	--

Description

Intermediates and Results from Cell Tracking Analyses, used to build the package vignette.

Usage

```
data(ThreeConditions)
```

Format

a list including 3 elements

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 NucLight 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. Cells 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. Cells were treated with 100 uM Rac1 Inhibitor (1177865-17-6, Calbiochem) or left untreated (controls). TIFF images were imported and processed using the cellmigRation library.

Examples

```
data(ThreeConditions)
```

track	<i>Track cells</i>
-------	--------------------

Description

Constructs n-dimensional trajectories from a scrambled list of particle coordinates determined at discrete times (e.g. in consecutive image frames)

Usage

```
track(xyzs, maxdisp, params)
```

Arguments

xyzs	an array listing the xy coordinates and data of the different particles at different times
maxdisp	an estimate of the maximum distance that a particle would move in a single time interval
params	a list containing a few tracking parameters that are needed for the analysis

Value

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.mathworks.com/matlabcentral/fileexchange/60349-fasttracks>

Examples

```
x0 <- data.frame(row = c(1, 30, 50, 5, 35, 55, 6, 56, 7, 58),
                 col = c(1, 30, 50, 5, 35, 55, 6, 56, 7, 58),
                 tau = c(1, 1, 1, 2, 2, 2, 3, 3, 4, 4))
cellmigRation:::track(x0, maxdisp = 10, params = NULL)
```

TrackCellsDataset	<i>Sample Stack of Fluorescent Cells</i>
-------------------	--

Description

Sample Stack of Fluorescent Cells to be used for computing cell tracks and stats

Usage

```
data(TrackCellsDataset)
```

Format

a trackedCells object including 10 stacks

Details

This stack of Fluorescent Cell images was obtained by processing TIFF files via 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 NucLight 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. Cells 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. A small selection of cells and image stacks is included in this dataset.

Examples

```
data(TrackCellsDataset)
```

trackedCells-class	<i>The trackedCells Class.</i>
--------------------	--------------------------------

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>

trackHypercubeBuild	<i>Track Hypercube Build</i>
---------------------	------------------------------

Description

Build an hypercube used for the tracking step

Usage

```
trackHypercubeBuild(xyzs, env)
```

Arguments

xyzs	data.frame, including input cell centroid positions
env	environment, including all objects used for the tracking

Details

a message is printed if an issue (typically arising by a non-suitable environment being passed as the env argument) is detected. See the example below.

Value

NULL is returned while objects in env are updated

Examples

```
cellmigRation:::trackHypercubeBuild(data.frame(1), new.env())
```

trackSlideProcessing	<i>Tracking Slide Processing</i>
----------------------	----------------------------------

Description

Frame-by-frame Slide Processing as part of the Cell Tracking Processing

Usage

```
trackSlideProcessing(xyzs, maxdisp, i, env)
```

Arguments

xyzs	data.frame, including input cell centroid positions
maxdisp	numeric, value of maximum cell dispersion in pixels
env	environment, including all objects used for the tracking

Details

a message is printed if an issue (typically arising by a non-suitable environment being passed as the env argument) is detected. See the example below.

Value

FALSE is returned as long as cells are detected and tracked; TRUE is returned when no further cells to track are found; objects in env are updated

Examples

```
cellmigRation:::trackSlideProcessing(data.frame(1), 1, 1, new.env())
```

trackSlideWrapUp	<i>Tracking Slide Wrap Up</i>
------------------	-------------------------------

Description

Perform Tracking Slide Wrap Up as part of the Cell Tracking Processing

Usage

```
trackSlideWrapUp(xyzs, maxdisp, i, env)
```

Arguments

xyzs	data.frame, including input cell centroid positions
maxdisp	numeric, value of maximum cell dispersion in pixels
i	integer, index of the current cycle
env	environment, including all objects used for the tracking

Details

a message is printed if an issue (typically arising by a non-suitable environment being passed as the env argument) is detected. See the example below.

Value

FALSE is returned; objects in env are updated

Examples

```
cellmigRation:::trackSlideWrapUp(data.frame(1), 1, 1, new.env())
```

TrajectoryDataset	<i>Trajectories of 350 cells</i>
-------------------	----------------------------------

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 NucLight 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)
```

trivialBondRaster	<i>Trivial Bond Raster</i>
-------------------	----------------------------

Description

Perform Trivial Bond Raster as part of the Cell Tracking Processing

Usage

```
trivialBondRaster(xyzs, env)
```

Arguments

xyzs	data.frame, including input cell centroid positions
env	environment, including all objects used for the tracking

Details

a message is printed if an issue (typically arising by a non-suitable environment being passed as the env argument) is detected. See the example below.

Value

FALSE is returned; objects in env are updated

Examples

```
cellmigRation:::trivialBondRaster(data.frame(1), new.env())
```

trivialBondTracking	<i>Trivial Bond Tracking</i>
---------------------	------------------------------

Description

Perform Trivial Bond Tracking as part of the Cell Tracking Processing

Usage

```
trivialBondTracking(xyzs, env)
```

Arguments

xyzs	data.frame, including input cell centroid positions
env	environment, including all objects used for the tracking

Details

a message is printed if an issue (typically arising by a non-suitable environment being passed as the env argument) is detected. See the example below.

Value

FALSE is returned when particles are correctly tracked; TRUE is returned when no particles are found to be tracked; objects in env are updated

Examples

```
cellmigRation:::trivialBondTracking(data.frame(1), new.env())
```

ValidateTrackingArgs *Validate Tracking Arguments*

Description

Tool for Validate Tracking Arguments

Usage

```
ValidateTrackingArgs(  
  import_optiParam_from = NULL,  
  lnoise = NULL,  
  diameter = NULL,  
  threshold = NULL,  
  maxDisp = NULL,  
  memory_b = 0,  
  goodenough = 0,  
  show_plots = FALSE,  
  verbose = FALSE  
)
```

Arguments

import_optiParam_from	a trackedCells object, defaults to NULL
lnoise	numeric, lnoise value
diameter	numeric, lnoise value
threshold	numeric, lnoise value
maxDisp	numeric, lnoise value
memory_b	numeric, should be 0
goodenough	numeric, should be 0
show_plots	logical, shall plots be shown
verbose	logical, shall info be printed to console

Details

This is an internal function supporting the CellTracker function.

Value

list of processed data

Examples

```
cellmigRation:::ValidateTrackingArgs()
```

VeAutoCor	<i>Velocity AutoCorrelation</i>
-----------	---------------------------------

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/

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	<i>Visualize Cell Tracks originating at an Image Stack</i>
---------------------	--

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

tc_obj	a trackedCells object
stack	index of the stack
pnt.cex	cex of the point drawn around each cell
lwd	width of the lines visualizing cell tracks
col	color of the points and the tracks, e.g.: "red2"
col.untracked	color of the points that were not tracked further, e.g.: "gray45"
main	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.mathworks.com/matlabcentral/fileexchange/60349-fasttracks>

Examples

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

VisualizeCntr

Visualize Centroids

Description

Annotates centroids over an image

Usage

```
VisualizeCntr(
  centroids,
  width_px,
  height_px,
  pnt.cex = 1.2,
  txt.cex = 0.9,
  offset = 0.18,
  col = "red2"
)
```

Arguments

centroids	centroid data.frame
width_px	width of the image in pixels
height_px	height of the image in pixels
pnt.cex	cex of the point (circle) drawn around each cell
txt.cex	cex of the text used to annotate the image
offset	offset for the text annotations
col	color of the points, e.g. "red2"

Value

None

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/ <https://www.mathworks.com/matlabcentral/fileexchange/60349-fasttracks>

Examples

```
x1 <- data.frame(
  row = c(50, 80, 20, 65, 99),
  col = c(15, 25, 50, 65, 86))
plot(2, 2, xlim = c(0,1), ylim = c(0,1), xlab = "", ylab = "", las = 2)
cellmigRation::VisualizeCntr(x1, width_px = 100, height_px = 100)
```

VisualizeImg

Visualize a matrix image

Description

Shows an image representation of a numeric matrix. Typically, this is a non-negative numeric matrix, where signal (high values) corresponds to the presence of cells, or cell-like particles.

Usage

```
VisualizeImg(img_mtx, col = NULL, ...)
```

Arguments

img_mtx	numeric matrix corresponding to a image
col	character vector corresponding to a valid color palette
...	additional arguments will be passed to graphics::image()

Value

None

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/ <https://www.mathworks.com/matlabcentral/fileexchange/60349-fasttracks>

Examples

```
x <- vapply(
  seq_len(20),
  function(i) {runif(n = 20, min = 0, max = 10)},
  FUN.VALUE = numeric(20)
)
cellmigRation:::VisualizeImg(x)
```

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.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)
```

visualizeTrcks

Visualize Cell Tracks

Description

Annotates an image with cell centroids by adding cell ROIs and drawing cell tracks

Usage

```
visualizeTrcks(
  tracks,
  width_px,
  height_px,
  i.slice = 1,
  pnt.cex = 1.2,
  lwd = 1.2,
  col = "red"
)
```

Arguments

tracks	cell tracks
width_px	width in pixels
height_px	height in pixels
i.slice	index of the stack slice to use
pnt.cex	cex for the points (circles) drawn around the cells
lwd	lwd of cell tracks
col	color used for the cell tracks, .g. "red"

Value

None

Author(s)

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

References

https://www.data-pulse.com/dev_site/cellmigration/ <https://www.mathworks.com/matlabcentral/fileexchange/60349-fasttracks>

Examples

```
x1 <- data.frame(c(10, 30, 25, 55, 43, 39, 75, 72),
                 c(22, 28, 35, 24, 31, 39, 65, 73),
                 c( 1,  2,  3,  4,  5,  6,  7,  8),
                 c( 1,  1,  1,  1,  1,  1,  1,  1))
plot(2, 2, xlim = c(0,1), ylim = c(0,1), xlab = "", ylab = "", las = 2)
cellmigRation::visualizeTrcks(x1, width_px = 100, height_px = 100)
```

warnMessage

*Print Warning Messages***Description**

Print warning messages to the console when an issue is encountered

Usage

```
warnMessage(warn_log, quiet = FALSE)
```

Arguments

warn_log	list, including the warning logs
quiet	logical, shall the warning be printed

Value

log of warning messages is returned

Examples

```
cellmigRation::warnMessage(list("Hello world"), FALSE)
```

`WSADataset`*Trajectories of 147 cells*

Description

A dataset containing the coordinates and the ID of 147 cells from wound scratch migration experiment

Usage

```
data(WSADataset)
```

Format

A data frame with 11970 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 NucLight 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, a thin wound was introduced by scratching the cell monolayer. Next, cells 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(WSADataset)
```

`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/

Examples

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

Index

- * **No**
 - trackSlideProcessing, [107](#)
- * **Warning-**
 - trackSlideProcessing, [107](#)
- * **cellTracker**
 - VisualizeImg, [115](#)
- * **datasets**
 - TrajectoryDataset, [109](#)
- * **for**
 - trackSlideProcessing, [107](#)
- * **found**
 - trackSlideProcessing, [107](#)
- * **internal**
 - AddDimension, [5](#)
 - bpass, [8](#)
 - cellmigration-package, [5](#)
 - CellTrackerMainLoop, [16](#)
 - CentroidArray, [17](#)
 - CentroidValidation, [18](#)
 - circshift, [19](#)
 - cntrd, [20](#)
 - DetectRadii, [22](#)
 - FinalizeOptiParams, [27](#)
 - fixDA, [30](#)
 - fixExpName, [31](#)
 - fixFM1, [31](#)
 - fixFM2, [32](#)
 - fixFM3, [33](#)
 - fixFM4, [34](#)
 - fixFM5, [35](#)
 - fixFM6, [36](#)
 - fixID1, [37](#)
 - fixID2, [38](#)
 - fixID3, [38](#)
 - fixID4, [39](#)
 - fixID5, [40](#)
 - fixID6, [40](#)
 - fixMSD, [41](#)
 - fixPER1, [42](#)

- fixPER2, [43](#)
- fixPER3, [44](#)
- GenAllCombos, [47](#)
- initializeTrackParams, [66](#)
- innerBondRaster, [67](#)
- internalPermutation, [68](#)
- LinearConv2, [68](#)
- MakeHypercube, [70](#)
- matfix, [71](#)
- MigrationStats, [72](#)
- NextOdd, [74](#)
- NonParallel4OptimizeParams, [75](#)
- NonParallelTrackLoop, [76](#)
- nontrivialBondTracking, [76](#)
- OptimizeParamsMainLoop, [79](#)
- Parallel4OptimizeParams, [80](#)
- ParallelTrackLoop, [81](#)
- pkfnd, [83](#)
- PostProcessTracking, [89](#)
- Prep4OptimizeParams, [90](#)
- preProcCellMig, [91](#)
- runTrackingPermutation, [93](#)
- sinkAway, [102](#)
- subNetworkTracking, [103](#)
- ThreeConditions, [103](#)
- track, [104](#)
- TrackCellsDataset, [105](#)
- trackHypercubeBuild, [107](#)
- trackSlideProcessing, [107](#)
- trackSlideWrapUp, [108](#)
- trivialBondRaster, [109](#)
- trivialBondTracking, [110](#)
- ValidateTrackingArgs, [111](#)
- VisualizeCntr, [114](#)
- VisualizeImg, [115](#)
- visualizeTrcks, [117](#)
- warnMessage, [118](#)
- WSADataset, [119](#)

* **positions**

- trackSlideProcessing, [107](#)
- * **t=**
 - trackSlideProcessing, [107](#)
- AddDimension, [5](#)
- aggregateFR, [6](#)
- aggregateTrackedCells, [7](#)
- bpass, [8](#)
- CellMig (CellMig-class), [10](#)
- CellMig-class, [10](#)
- CellMigPCA, [11](#)
- CellMigPCAclust, [12](#)
- CellMigPCAclustALL, [13](#)
- cellmigRation (cellmigRation-package), [5](#)
- cellmigRation-package, [5](#)
- CellTracker, [14](#)
- CellTrackerMainLoop, [16](#)
- CentroidArray, [17](#)
- CentroidValidation, [18](#)
- circshift, [19](#)
- cntrd, [20](#)
- ComputeTracksStats, [21](#)
- DetectRadii, [22](#)
- DiAutoCor, [23](#)
- DiRatio, [24](#)
- DiRatioPlot, [25](#)
- EstimateDiameterRange, [26](#)
- FianlizeOptiParams, [27](#)
- FilterTrackedCells, [28](#)
- FinRes, [29](#)
- fixDA, [30](#)
- fixExpName, [31](#)
- fixFM1, [31](#)
- fixFM2, [32](#)
- fixFM3, [33](#)
- fixFM4, [34](#)
- fixFM5, [35](#)
- fixFM6, [36](#)
- fixID1, [37](#)
- fixID2, [38](#)
- fixID3, [38](#)
- fixID4, [39](#)
- fixID5, [40](#)
- fixID6, [40](#)
- fixMSD, [41](#)
- fixPER1, [42](#)
- fixPER2, [43](#)
- fixPER3, [44](#)
- FMI, [45](#)
- ForwardMigration, [46](#)
- GenAllCombos, [47](#)
- getAvailableAggrMetrics, [48](#)
- getCellImages, [49](#)
- getCellImages, trackedCells-method
(getCellImages), [49](#)
- getCellMigSlot, [50](#)
- getCellMigSlot, CellMig, character-method
(getCellMigSlot), [50](#)
- getCellsMeta, [50](#)
- getCellsStats, [51](#)
- getCellTrackMeta, [52](#)
- getCellTrackMeta, trackedCells-method
(getCellTrackMeta), [52](#)
- getCellTracks, [52](#)
- getCellTracks, trackedCells-method
(getCellTracks), [52](#)
- getCellTrackStats, [53](#)
- getCellTrackStats, trackedCells-method
(getCellTrackStats), [53](#)
- getDACtable, [54](#)
- getDiRatio, [55](#)
- getFMItable, [56](#)
- getForMigtable, [57](#)
- getImageCentroids, [58](#)
- getImageCentroids, trackedCells-method
(getImageCentroids), [58](#)
- getImageStacks, [58](#)
- getMSDtable, [59](#)
- getOptimizedParameters, [60](#)
- getOptimizedParams, [61](#)
- getOptimizedParams, trackedCells-method
(getOptimizedParams), [61](#)
- getPerAndSpeed, [61](#)
- getPopulationStats, [62](#)
- getProcessedImages, [63](#)
- getProcessedImages, trackedCells-method
(getProcessedImages), [63](#)
- getProcessingStatus, [63](#)
- getProcessingStatus, trackedCells-method
(getProcessingStatus), [63](#)
- getResults, [64](#)
- getTracks, [65](#)
- getVACtable, [65](#)

- initialize, CellMig-method
(CellMig-class), 10
- initialize, trackedCells-method
(trackedCells-class), 106
- initializeTrackParams, 66
- innerBondRaster, 67
- internalPermutation, 68
- LinearConv2, 68
- LoadTiff, 69
- MakeHypercube, 70
- matfix, 71
- MigrationStats, 72
- MSD, 73
- NextOdd, 74
- NonParallel4OptimizeParams, 75
- NonParallelTrackLoop, 76
- nontrivialBondTracking, 76
- OptimizeParams, 77
- OptimizeParamsMainLoop, 79
- Parallel4OptimizeParams, 80
- ParallelTrackLoop, 81
- PerAndSpeed, 81
- pkfnd, 83
- plot3DAllTracks, 84
- plot3DTracks, 85
- plotAllTracks, 86
- plotSampleTracks, 87
- PlotTracksSeparately, 88
- PostProcessTracking, 89
- Prep4OptimizeParams, 90
- preProcCellMig, 91
- rmPreProcessing, 92
- runTrackingPermutation, 93
- setAnalyticParams, 94
- setAnalyticParams, trackedCells, list-method
(setAnalyticParams), 94
- setCellMigSlot, 94
- setCellMigSlot, CellMig, character, ANY-method
(setCellMigSlot), 94
- setCellMigSlot, CellMig, character-method
(setCellMigSlot), 94
- setCellsMeta, 95
- setCellTracks, 96
- setCellTracks, trackedCells, matrix-method
(setCellTracks), 96
- setExpName, 97
- setExpName, CellMig, character-method
(setExpName), 97
- setOptimizedParams, 97
- setOptimizedParams, trackedCells, ANY, ANY-method
(setOptimizedParams), 97
- setOptimizedParams, trackedCells-method
(setOptimizedParams), 97
- setProcessedImages, 98
- setProcessedImages, trackedCells, list-method
(setProcessedImages), 98
- setProcessingStatus, 99
- setProcessingStatus, trackedCells, character, numeric-method
(setProcessingStatus), 99
- setTrackedCellsMeta, 99
- setTrackedCellsMeta, trackedCells, list-method
(setTrackedCellsMeta), 99
- setTrackedCentroids, 100
- setTrackedCentroids, trackedCells, list-method
(setTrackedCentroids), 100
- setTrackedPositions, 101
- setTrackedPositions, trackedCells, data.frame-method
(setTrackedPositions), 101
- setTrackingStats, 101
- setTrackingStats, trackedCells, ANY, ANY-method
(setTrackingStats), 101
- setTrackingStats, trackedCells-method
(setTrackingStats), 101
- sinkAway, 102
- subNetworkTracking, 103
- ThreeConditions, 103
- track, 104
- TrackCellsDataset, 105
- trackedCells (trackedCells-class), 106
- trackedCells-class, 106
- trackHypercubeBuild, 107
- trackSlideProcessing, 107
- trackSlideWrapUp, 108
- TrajectoryDataset, 109
- trivialBondRaster, 109
- trivialBondTracking, 110
- ValidateTrackingArgs, 111
- VeAutoCor, 112
- visualizeCellTracks, 113
- VisualizeCntr, 114

VisualizeImg, [115](#)
VisualizeStackCentroids, [116](#)
visualizeTrcks, [117](#)

warnMessage, [118](#)
WSADataset, [119](#)
wsaPreProcessing, [119](#)