Package 'LOMAR'

October 12, 2022

Type Package

Title Localization Microscopy Data Analysis

Version 0.2.1

Maintainer Jean-Karim Heriche <heriche@embl.de>

Description Read, register and compare point sets from single molecule localization microscopy.

URL https://git.embl.de/heriche/lomar

Depends R (>= 3.6.0)

biocViews

Imports Rcpp, FNN, stats, data.table, parallel, doParallel, foreach, proxy, reshape2, pracma, transport, RANN, ff, aws, dbscan, EBImage

LinkingTo BH (>= 1.78.0-0), Rcpp

Suggests testthat

License GPL-3

Encoding UTF-8

ByteCompile true

RoxygenNote 7.1.2

SystemRequirements C++14, gmp, fftw3

NeedsCompilation yes

Author Jean-Karim Heriche [cre, aut] (<https://orcid.org/0000-0001-6867-9425>)

Repository CRAN

Date/Publication 2022-03-16 11:10:04 UTC

R topics documented:

`apply_transformation` *apply_transformation*

Description

Apply rotation and translation to a point set

Usage

```
apply_transformation(X, R, t, s)
```

Arguments

X	a point set as an N x D matrix
R	D x D rotation matrix
t	1 x D translation vector
s	scaling factor

Value

transformed point set as a N x D matrix

---

ary2ps *ary2ps*

Description

Convert a 4d array to a list of 3d point sets. The points are formed by extracting the coordinates of array values strictly above the given cut-off (default 0).

Usage

```
ary2ps(ary, bkg = 0)
```

Arguments

ary	a 4d array with last dimension indexing instances.
bkg	Extract points for array values strictly above this (default = 0)

Value

a list of point sets.

---

circle_hough_transform *Circle Hough transform*

Description

Extract coordinates of the centres of circles from a 2D image using the Hough transform

Usage

```
circle_hough_transform(
  pixels,
  rmin,
  rmax,
  threshold,
  resolution = 360,
  ncpu = 1
)
```

Arguments

pixels	input data, either a matrix representing a 2D image or a data frame of signal coordinates with columns x, y. For images, background is expected to be 0 and signal to have positive values.
rmin	minimum search radius.
rmax	maximum search radius.
threshold	score threshold between 0 and 1.
resolution	number of steps in the circle transform (default: 360). This represents the maximum number of votes a point can get.
ncpu	number of threads to use to speed up computation (default: 1)

Value

a data frame with columns x, y, r and score

Examples

```
point.set <- data.frame(x = c(-9.8,-5.2,12.5,2.5,4.5,1.3,-0.2,0.4,9.3,-1.4,0.5,-1.1,-7.7),
                        y = c(-4.2,1.5,-0.5,12,-3,-7.2,10.9,6.7,-1.3,10,6.7,-6.2,2.9))
circles <- circle_hough_transform(pixels = point.set, rmin = 3, rmax = 6, resolution = 100,
                                  threshold = 0.1, ncpu = 1)
```

---

costWd *costWd*

Description

Objective function to minimize when using GMMs

Usage

```
costWd(Tr, X, Y, CX, CY, w1 = NULL, w2 = NULL, S = NULL)
```

Arguments

Tr	Transformation vector as translation vector + rotation (angle in 2d, quaternion in 3d))
X	matrix of means of first GMM (i.e. reference point set)
Y	matrix of means of second GMM (i.e. moving point set)
CX	array of covariance matrices of first GMM such that X[i,] has covariance matrix CX[,,i]
CY	array of covariance matrices of second GMM such that Y[i,] has covariance matrix CY[,,i]
w1	(optional) vector of mixture weights of first GMM.
w2	(optional) vector of mixture weights of second GMM.
S	(optional) array of pre-computed sqrtm(sqrtm(CX[,,i]) %*% CY[,,j] %*% sqrtm(CX[,,i]))

Value

cost value

---

cpd *cpd*

Description

Affine and rigid registration of two point sets using the coherent point drift algorithm. See: Myronenko A., Song X. (2010): "Point-Set Registration: Coherent Point Drift", IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 32, issue 12, pp. 2262-2275.

Usage

```
cpd(
  X,
  Y,
  w = 0,
  weights = NULL,
  scale = FALSE,
  maxIter = 100,
  subsample = NULL,
  tol = 1e-04
)
```

Arguments

X	reference point set, a N x D matrix
Y	point set to transform, a M x D matrix,
w	noise weight in the range [0, 1)
weights	a M x N matrix of point correspondence weights
scale	logical (default: FALSE), whether to use scaling
maxIter	maximum number of iterations to perform (default: 100)
subsample	if set, use this randomly selected fraction of the points
tol	tolerance for determining convergence

Value

a list of

- Y: transformed point set,
- R: rotation matrix,
- t: translation vector,
- s: scaling factor,
- P: matrix of correspondence probabilities between the two point sets,
- sigma: final variance,
- iter: number of iterations performed,
- converged: boolean, whether the algorithm has converged.

Examples

```
data.file1 <- system.file("test_data", "parasaurolophusA.txt", package = "LOMAR",
 mustWork = TRUE)
PS1 <- read.csv(data.file1, sep = '\t', header = FALSE)
data.file2 <- system.file("test_data", "parasaurolophusB.txt", package = "LOMAR",
 mustWork = TRUE)
PS2 <- read.csv(data.file2, sep = '\t', header = FALSE)
transformation <- cpd(PS1, PS2, maxIter = 10, tol = 1e-3)
## Not run:
# Visualize registration outcome
library(rgl)
plot3d(PS1, col = "blue")
points3d(PS2, col = "green")
points3d(transformation[['Y']], col = "magenta")

## End(Not run)
```

crop_point_set *crop_point_set*

Description

Retain points in the set that are within the given distance from the geometric median of the set. Using the geometric median is more robust than using the centre of mass (i.e. mean).

Usage

```
crop_point_set(point.set, size)
```

Arguments

point.set	a point set as a matrix with columns x,y,z.
size	vector of distances from the geometric median of the points along each axis. Points are discarded if they are outside the ellipsoid defined by size and centred on the geometric median of the points.

Value

point set as a matrix with columns x,y,z.

downsample *downsample*

Description

Weighted downsampling of a point set. If point weights are not provided, they are computed to be proportional to the local density around each point.

Usage

```
downsample(point.set, n = NULL, k = NULL, weights = NULL)
```

Arguments

point.set	a point set
n	integer, sample size.
k	integer, number of nearest neighbours to consider to estimate local density
weights	a vector of probability weights

Value

a point set

`find_elbow` *find_elbow*

Description

Find elbow in a 2D curve represented by a list of ordered values

Usage

```
find_elbow(values)
```

Arguments

`values`	vector of values in decreasing order

Details

This function finds the point with maximum distance from the line between the first and last points. Adapted from StackOverflow: http://stackoverflow.com/questions/2018178/finding-the-best-trade-off-point-on-a-curve

Value

index and value of the selected point

`Gaussian_Wd` *Gaussian_Wd*

Description

Compute 2-Wasserstein distance between two Gaussian distributions

Usage

```
Gaussian_Wd(m1, m2, S1, S2, S = NULL)
```

Arguments

`m1`	mean of first distribution
`m2`	mean of second distribution
`S1`	variance of first distribution
`S2`	variance of second distribution
`S`	(optional) matrix of pre-computed sqrtm(sqrtm(S1) %*% S2 %*% sqrtm(S1))

Value

distance value

get_kernel_matrix *get_kernel_matrix*

Description

Compute kernel/distance matrix between persistence diagrams.

Usage

```
get_kernel_matrix(
  Diag = NULL,
  method = c("sWd", "pssk"),
  dimensions = NULL,
  return.dist = FALSE,
  M = NULL,
  sigma = NULL,
  ncpu = 1
)
```

Arguments

Diag	list of persistence diagrams as n x 3 matrices
method	which kernel or distance to compute. One of sWd (for sliced Wasserstein kernel) or pssk (for the persistence scale-space kernel)
dimensions	vector of the dimensions of the topological features to consider, if NULL (default) use all available dimensions
return.dist	logical (default: FALSE) for method sWd, whether to return the sliced Wasserstein distance matrix instead of the kernel.
M	number of slices for the sliced Wasserstein kernel
sigma	kernel bandwidth
ncpu	number of parallel threads to use for computation

Value

a matrix

Examples

```
PS <- list(data.frame(x = c(2.4,-6.9,4.6,-0.7,-3.3,-4.9,-3.5,-3.5,4.2,-7),
                      y = c(5.7,1.9,4.8,3.4,-3,-2.1,7.2,1.8,6.1,-1.6),
                      z = c(2.7,-0.1,-0.7,-0.6,0.4,-1.5,-0.6,-0.9,2.2,0.7)),
           data.frame(x = c(0,0,3.1,-5.6,-5,-7.4,-0.7,-7.7,-6.7,4,4.2,0.2,5.8,3.9,3.9),
               y = c(6.3,-6.1,-3.5,4.6,-4.1,0.3,8.8,-2.3,2.9,3.7,-1.4,-3.9,5.5,-1.2,-6.7),
               z = c(-1.5,1.7,-0.4,-1.4,1.8,1.7,-0.9,-1.8,-0.5,1.7,1.3,0.5,-1.4,1.6,-0.1)),
           data.frame(x = c(-9.8,-5.2,12.5,2.5,4.5,1.3,-0.2,0.4,9.3,-1.4,0.5,-1.1,-7.7),
                      y = c(-4.2,1.5,-0.5,12,-3,-7.2,10.9,6.7,-1.3,10,6.7,-6.2,2.9),
```

```
                        z = c(3.4,-3.8,-1.4,1.8,3.5,2.5,2.6,-4.8,-3.8,3.9,4.1,-3.6,-4)))
Dgs <- get_persistence_diagrams(point.sets = PS, maxdimension = 1, maxscale = 5, ncpu = 1)
K <- get_kernel_matrix(Diag = Dgs, method = 'sWd', dimensions = c(0,1), M = 10, sigma = 5)
```

`get_persistence_diagrams` *get_persistence_diagrams*

Description

Compute persistence diagrams for a list of point sets. By default, compute persistent homology from the Vietoris-Rips filtration. If use.dtm is TRUE, compute instead the persistent homology of the sublevel set of the distance to measure evaluated over a grid.

Usage

```
get_persistence_diagrams(
  point.sets = NULL,
  maxdimension = NULL,
  maxscale = NULL,
  use.dtm = FALSE,
  m0 = NULL,
  grid.by = NULL,
  ncpu = 1
)
```

Arguments

`point.sets`	list of point sets, each as a data frame with columns x,y,z
`maxdimension`	maximum dimension of the homological features to be computed
`maxscale`	limit of the Vietoris-Rips filtration
`use.dtm`	logical (default: FALSE), whether to use the distance to measure function
`m0`	parameter for the dtm function
`grid.by`	vector of space between points of the grid for the dtm function along each dimension
`ncpu`	number of parallel threads to use for computation

Value

a list of persistence diagrams as n x 3 matrices. Each row is a topological feature and the columns are dimension, birth and death of the feature.

Examples

```
PS <- list(data.frame(x = c(2.4,-6.9,4.6,-0.7,-3.3,-4.9,-3.5,-3.5,4.2,-7),
                      y = c(5.7,1.9,4.8,3.4,-3,-2.1,7.2,1.8,6.1,-1.6),
                      z = c(2.7,-0.1,-0.7,-0.6,0.4,-1.5,-0.6,-0.9,2.2,0.7)),
           data.frame(x = c(0,0,3.1,-5.6,-5,-7.4,-0.7,-7.7,-6.7,4,4.2,0.2,5.8,3.9,3.9),
               y = c(6.3,-6.1,-3.5,4.6,-4.1,0.3,8.8,-2.3,2.9,3.7,-1.4,-3.9,5.5,-1.2,-6.7),
               z = c(-1.5,1.7,-0.4,-1.4,1.8,1.7,-0.9,-1.8,-0.5,1.7,1.3,0.5,-1.4,1.6,-0.1)))
Diags <- get_persistence_diagrams(point.sets = PS, maxdimension = 1, maxscale = 5, ncpu = 1)
```

GMM_Wd *GMM_Wd*

Description

Compute 2-Wasserstein distance between two Gaussian mixture models See: Delon J, Desolneux A. (2019) A Wasserstein-type distance in the space of Gaussian Mixture Models. hal-02178204v2

Usage

```
GMM_Wd(m1, m2, S1, S2, w1 = NULL, w2 = NULL, S = NULL)
```

Arguments

m1	matrix of means of first GMM
m2	matrix of means of second GMM
S1	array of covariance matrices of first GMM such that m1[i,] has covariance matrix S1[,,i]
S2	array of covariance matrices of second GMM such that m2[i,] has covariance matrix S2[,,i]
w1	(optional) vector of mixture weights of first GMM.
w2	(optional) vector of mixture weights of second GMM.
S	(optional) array of pre-computed sqrtm(sqrtm(S1[,,i]) %*% S2[,,j] %*% sqrtm(S1[,,i]))

Value

list of distance value d and optimal transport matrix ot

gradientWd *gradientWd*

Description

Gradient of the objective function with respect to rotation and translation parameters

Usage

```
gradientWd(Tr, X, Y, CX, CY, w1 = NULL, w2 = NULL, S = NULL)
```

Arguments

Tr	Transformation vector as translation vector + rotation (angle in 2d, quaternion in 3d))
X	matrix of means of first GMM (i.e. reference point set)
Y	matrix of means of second GMM (i.e. moving point set)
CX	array of covariance matrices of first GMM such that X[i,] has covariance matrix C1[,,i]
CY	array of covariance matrices of second GMM such that Y[i,] has covariance matrix C2[,,i]
w1	(optional) vector of mixture weights of first GMM.
w2	(optional) vector of mixture weights of second GMM.
S	(optional) array of pre-computed sqrtm(sqrtm(CX[,,i]) %*% CY[,,j] %*% sqrtm(CX[,,i]))

Value

gradient vector

icp *icp*

Description

Rigid registration of two point sets using the iterative closest point algorithm.

Usage

```
icp(
  X,
  Y,
  weights = NULL,
  iterations = 100,
  subsample = NULL,
  scale = FALSE,
  tol = 0.001
)
```

Arguments

X	reference point set, a N x D matrix
Y	point set to transform, a M x D matrix,
weights	vector of length nrow(Y) containing weights for each point in Y. Not implemented.
iterations	number of iterations to perform (default: 100)
subsample	if set, use this randomly selected fraction of the points
scale	logical (default: FALSE), whether to use scaling.
tol	tolerance for determining convergence

Value

a list of

- Y: transformed point set, a M x D matrix,
- R: rotation matrix,
- t: translation vector,
- s: scaling factor,
- iter: number of iterations performed,
- conv: boolean, whether the algorithm has converged.

Examples

```
data.file1 <- system.file("test_data", "parasaurolophusA.txt", package = "LOMAR",
 mustWork = TRUE)
PS1 <- read.csv(data.file1, sep = '\t', header = FALSE)
data.file2 <- system.file("test_data", "parasaurolophusB.txt", package = "LOMAR",
 mustWork = TRUE)
PS2 <- read.csv(data.file2, sep = '\t', header = FALSE)
transformation <- icp(PS1, PS2, iterations = 10, tol = 1e-3)
## Not run:
# Visualize registration outcome
library(rgl)
plot3d(PS1, col = "blue")
```

```
points3d(PS2, col = "green")
points3d(transformation[['Y']], col = "magenta")

## End(Not run)
```

`idx2rowcol` *idx2rowcol*

Description

Convert indices into a dist object to row, column coordinates of the corresponding distance matrix

Usage

```
idx2rowcol(idx, n)
```

Arguments

`idx`	vector of indices
`n`	size of the n x n distance matrix

Value

a matrix with two columns nr and nc

`img2ps` *img2ps*

Description

Read an image into a point set. The points are formed by extracting the coordinates of voxel values strictly above the given cut-off (default 0).

Usage

```
img2ps(img = NULL, bkg = 0, crop.size = NULL)
```

Arguments

`img`	either a 2d or 3d array or a path to a file containing a 2d or 3d image.
`bkg`	Extract points for values strictly above this (default = 0).
`crop.size`	vector (of length 2 or 3) containing the desired reduced size of the images along each dimension, e.g. c(30,30,30).

Value

a point set as matrix with columns x,y[,z]

Examples

```
img.file <- system.file("test_data/img", "alien1_3d.tif", package = "LOMAR",
 mustWork = TRUE)
point_set <- img2ps(img = img.file, bkg = 0)
```

---

jrmpc *jrmpc*

Description

Joint registration of multiple point sets See: G. D. Evangelidis, D. Kounades-Bastian, R. Horaud, andE. Z. Psarakis. A generative model for the joint registration of multiple point sets. In European Conference on Computer Vision, pages 109–122. Springer, 2014

Usage

```
jrmpc(
  V,
  C = NULL,
  K = NULL,
  g = NULL,
  initialPriors = NULL,
  updatePriors = TRUE,
  maxIter = 100,
  fixedVarIter = 0,
  tol = 0.01,
  initializeBy = NULL,
  model.selection = FALSE,
  model.selection.threshold = NULL,
  rotation.only = FALSE
)
```

Arguments

V	list of point sets as N x D matrices
C	(optional) list of arrays of covariance matrices with C[[j]][,,i] the covariance matrix associated with point i of set j.
K	(optional) number of components of the GMM, defaults to the average number of points in a set.
g	(optional) proportion of noisy points, defaults to 1/K. If set, priors will be initialized uniformly.

`initialPriors` (optional) vector of length K of prior probabilities. Defaults to uniform distribution using g. If set, will determine g so it is an error to specify g with initialPriors.

`updatePriors` logical, whether to update priors at each iteration (default: TRUE).

`maxIter` maximum number of iterations to perform (default: 100).

`fixedVarIter` number of iterations before starting variance updates

`tol` tolerance for determining convergence (default: 1e-2).

`initializeBy` (optional) how to initialize the GMM means. Defaults to distributing the means on the surface of the sphere enclosing all (centred) sets. Currently supported values are:

- 'sampling': sample from the data,
- a K x D matrix of points

`model.selection` whether to perform model selection (default: FALSE). If set to TRUE, GMM components with no support in the data are deleted.

`model.selection.threshold` value below which we consider a GMM component has no support, set to 1/K if not explicitly given

`rotation.only` if set to TRUE, no translation is performed (default: FALSE)

Value

a list of

- Y: list of transformed point sets as N x d matrices,
- R: list of d x d rotation matrices, one for each point set in V,
- t: list of translation vectors, one for each point set in V,
- M: centres of the GMM,
- S: variances of the GMM.
- a: list of posterior probabilities as N x K matrices
- iter: number of iterations
- conv: error value used to evaluate convergence relative to tol
- z: support scores of the GMM components

Examples

```
X <- read.csv(system.file("test_data", "parasaurolophusA.txt", package="LOMAR",
 mustWork = TRUE), sep = "\t")
Y <- read.csv(system.file("test_data", "parasaurolophusB.txt", package="LOMAR",
mustWork = TRUE), sep = "\t")
Z <- read.csv(system.file("test_data", "parasaurolophusC.txt", package="LOMAR",
mustWork = TRUE), sep = "\t")
PS <- list(X, Y, Z)
C <- list()
for(i in 1:3) {
```

```
 cv <- diag(0.1, ncol(PS[[i]])) + jitter(0.01, amount = 0.01)
 cv <- replicate(nrow(PS[[i]]), cv)
 C[[i]] <- cv
}
transformation <- jrmpc(PS, C = C, K = 100, maxIter = 20, tol = 0.01,
model.selection = TRUE)
## Not run:
# Visualize registration outcome
library(rgl)
colours <- c("blue", "green", "magenta")
Yt <- transformation[['Y']]
plot3d(Yt[[1]], col = colours[1])
for(i in 2:length(Yt)) {
 points3d(Yt[[i]], col = colours[i])
}
# Visualize GMM centres highlighting those with high variance
GMM <- as.data.frame(cbind(transformation[['M']], transformation[['S']]))
colnames(GMM) <- c("x", "y", "z", "S")
colours <- rep("blue", nrow(GMM))
# Find high variance components
threshold <- quantile(transformation[['S']], 0.75)
high.var.idx <- which(transformation[['S']]>threshold)
colours[high.var.idx] <- "red"
plot3d(GMM[, c("x", "y", "z")], col = colours, type = 's', size = 2, box = FALSE, xlab = '',
      ylab = '', zlab = '', xlim = c(-0.15,0.15), ylim = c(-0.15, 0.15),
      zlim = c(-0.15, 0.15))

## End(Not run)
```

`local_densities` *local_densities*

Description

Compute local point density at each point of a point set

Usage

```
local_densities(X, k = NULL)
```

Arguments

`X`	point set, a N x D matrix
`k`	(optional) number of nearest neighbors used (defaults to all points).

Details

Local density is computed as in Ning X, Li F, Tian G, Wang Y (2018) An efficient outlier removal method for scattered point cloud data. PLOS ONE 13(8):e0201280. https://doi.org/10.1371/journal.pone.0201280

Value

vector of density value for each point

`locprec2cov` *locprec2cov*

Description

Converts localization precision columns to a list of arrays of covariance matrices

Usage

```
locprec2cov(point.sets, scale = FALSE)
```

Arguments

`point.sets`	a list of n point sets with locprec columns (locprecz column required for 3D data)
`scale`	logical, whether to scale the localization precision by the variance of the coordinates

Value

a list of 2x2xn or 3x3xn arrays.

`locs2ps` *locs2ps*

Description

Cluster localizations into point sets using DBSCAN

Usage

```
locs2ps(
  points,
  eps,
  minPts,
  keep.locprec = TRUE,
  keep.channel = TRUE,
  cluster.2d = FALSE
)
```

Arguments

points	a point set as a data frame of coordinates with columns x,y,z.
eps	DBSCAN parameter, size of the epsilon neighbourhood
minPts	DBSCAN parameter, number of minimum points in the eps region
keep.locprec	logical (default: TRUE), whether to preserve the localization precision columns
keep.channel	logical (default: TRUE), whether to preserve channel information column
cluster.2d	logical (default: FALSE), whether to cluster only using x,y (and ignore z)

Value

a list of matrices with columns x,y,z and eventually locprec[z] and names set to the cluster indices.

---

locs_from_csv *locs_from_csv*

Description

Reads and filters single molecule localization events from a csv file as typically output by the SMAP software. The main columns of interest are the coordinates (x, y, z), point set membership (site) and localization precision (locprec and locprecz).

Usage

```
locs_from_csv(
  file = NULL,
  roi = NULL,
  channels = NULL,
  frame.filter = NULL,
  llrel.filter = NULL,
  locprec.filter = 0,
  locprecz.filter = 0
)
```

Arguments

file	a csv file with columns x[nm], y[nm], z[nm] and optionally site[numbers], channel, locprec[nm] and locprecz[nm], other columns are ignored.
roi	region of interest, keep points within the specified volume. Must be a data frame with columns x,y,z and rows min and max defining a bounding box.
channels	vector of integers indicating which channel(s) of a multicolour experiment to get data from.
frame.filter	vector of min and max values, filter out points from frames outside the specified range.
llrel.filter	vector of min and max values, filter out points on log-likelihood (for fitted data).

`locprec.filter` filter out points with locprec value greater than the specified number. Points with locprec == 0 are also removed.

`locprecz.filter` filter out points with locprecz value greater than the specified number. Points with locprecz == 0 are also removed.

Value

a data frame with columns x,y,z, optionally site, locprec and locprecz.

Examples

```
data.file <- system.file("test_data", "simulated_NUP107_data.csv", package = "LOMAR",
 mustWork = TRUE)
locs <- locs_from_csv(file = data.file, locprec.filter = 20)
```

`points2img` *points2img*

Description

Convert a data frame of point coordinates into an image. Expected photon count at each voxel is computed as in: F. Huang, S. L. Schwartz, J. M. Byars, and K. A. Lidke, "Simultaneous multiple-emitter fitting for single molecule super-resolution imaging," Biomed. Opt. Express 2(5), 1377–1393 (2011).

Usage

```
points2img(points, voxel.size, method, channels = NULL, ncpu = 1)
```

Arguments

`points` a point set as a data frame of coordinates with columns x,y,z.

`voxel.size` a numeric vector of length 3 indicating the size of the voxel along x,y and z in the same unit as the coordinates (e.g. nm)

`method` how to calculate voxel values. Available methods are:

- 'histogram': value is the number of points (i.e. emitters) in the voxel
- 'photon': value is the expected number of photons from the points in the voxel. Input data frame must have columns locprec, locprecz and phot[on].

`channels` vector of channels to consider, must be values present in the input data frame channel column

`ncpu` number of threads to use to speed up computation (default: 1)

Value

an array of dimensions x,y,z and channels if applicable

Examples

```
point.set <- data.frame(x = c(-9.8,-5.2,12.5,2.5,4.5,1.3,-0.2,0.4,9.3,-1.4,0.5,-1.1,-7.7),
                        y = c(-4.2,1.5,-0.5,12,-3,-7.2,10.9,6.7,-1.3,10,6.7,-6.2,2.9),
                        z = c(3.4,-3.8,-1.4,1.8,3.5,2.5,2.6,-4.8,-3.8,3.9,4.1,-3.6,-4))
img <- points2img(point.set, voxel.size = c(2,2,2), method = 'histogram')
```

---

points_from_roi *points_from_roi*

Description

Extract points within given bounding box. Points are translated so that (0,0,0) correspond to the bounding box corner defined by roi['min',c('x','y','z')]

Usage

```
points_from_roi(points, roi)
```

Arguments

points	a point set as a data frame of coordinates with columns x,y,z.
roi	a data frame with columns x,y,z and rows min and max defining a bounding box

Value

a data frame with same columns as input

---

point_sets_from_locs *point_sets_from_locs*

Description

Extracts list of point sets from a data frame of single molecule localization coordinates. By default, uses point set membership indicated in the site column.

Usage

```
point_sets_from_locs(
  locs = NULL,
  channels = NULL,
  min.cardinality = NULL,
  max.cardinality = NULL,
  crop.size = NULL,
  keep.locprec = TRUE,
  sample.size = NULL,
```

```
  ignore.site = FALSE,
  cluster.points = FALSE,
  eps = NULL,
  minPts = NULL
)
```

Arguments

locs,	a data frame with columns x[nm], y[nm], z[nm] and optionally site[numbers], locprec[nm] and locprecz[nm], other columns are ignored.
channels	vector of integers indicating which channel(s) of a multicolour experiment to extract point sets from.
min.cardinality	filter out point sets with less than the specified number of points.
max.cardinality	filter out point sets with more than the specified number of points.
crop.size	remove points from a set if they are further away than the specified distance from the center of the set.
keep.locprec	logical (default:TRUE). Whether to keep locprec information for each point.
sample.size	returns this number of randomly selected point sets. Selects the point sets after applying eventual filtering.
ignore.site	logical (default: FALSE), set to TRUE if point set membership is not present or needed.
cluster.points	logical (default: FALSE), whether to cluster the points using DBSCAN (only if ignore.site is also TRUE).
eps	DBSCAN parameter, size of the epsilon neighbourhood
minPts	DBSCAN parameter, number of minimum points in the eps region

Value

a list of matrices with columns x,y,z, optionally locprec and name set to the value of the site column (if applicable).

Examples

```
data.file <- system.file("test_data", "simulated_NUP107_data.csv", package = "LOMAR",
 mustWork = TRUE)
locs <- locs_from_csv(file = data.file, locprec.filter = 20)
point.sets <- point_sets_from_locs(locs, keep.locprec = TRUE, min.cardinality = 15)
```

point_sets_from_tiffs *point_sets_from_tiffs*

Description

Read in single molecule localization events from a series of 3D images in TIFF files where each image file represents a point set.

Usage

```
point_sets_from_tiffs(
  image_dir = NULL,
  pattern = NULL,
  image.size = NULL,
  sample.size = NULL,
  sample.first = FALSE,
  min.cardinality = NULL,
  max.cardinality = NULL,
  crop.size = NULL
)
```

Arguments

image_dir	path to a directory containing the TIFF files.
pattern	regular expression, select images whose file path matches the given pattern.
image.size	vector of length 3 containing the size of the images along each dimension, e.g. c(40,40,40).
sample.size	if set, selects this number of images at random. A sample size larger than the available number of samples produces a warning and is ignored.
sample.first	if TRUE, samples are selected before applying any eventual filtering. This is more efficient as it avoids reading all data files.
min.cardinality	if set, filter out all point sets with less than the specified number of points.
max.cardinality	if set, filter out all point sets with more than the specified number of points.
crop.size	vector of length 3 containing the desired reduced size of the images along each dimension, e.g. c(30,30,30).

Value

a list with two elements:

- point.sets: a list of point sets as matrices with columns x,y,z and
- file.names: a vector of paths to the TIFF files from which the point sets were extracted.

Examples

```
data.dir <- system.file("test_data/img", package = "LOMAR", mustWork = TRUE)
point_sets <- point_sets_from_tiffs(image_dir = data.dir, pattern = "\\.tiff?$",
 image.size = c(64, 64, 4), min.cardinality = 10)
```

---

ps2ary *ps2ary*

Description

Convert a list of 3d point sets to a 4d array. Also works for 2d point sets to 3d array conversion.

Usage

```
ps2ary(point.sets, dims)
```

Arguments

point.sets	a list of point sets.
dims	vector of dimensions of the axes (x,y in 2d, x,y,z in 3d).

Value

a 3d or 4d array.

---

pssk *pssk*

Description

Compute the persistence scale-space kernel on persistence diagrams. Reference: Jan Reininghaus, Stefan Huber, Ulrich Bauer, and Roland Kwitt. A stable multi-scale kernel for topological machine learning. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pages 4741–4748, 2015.

Usage

```
pssk(Dg1 = NULL, Dg2 = NULL, sigma = NULL, dimensions = NULL)
```

Arguments

Dg1	a persistence diagram as a n1 x 3 matrix where each row is a topological feature and the columns are dimension, birth and death of the feature.
Dg2	another persistence diagram as a n2 x 3 matrix
sigma	kernel bandwidth
dimensions	vector of the dimensions of the topological features to consider, if NULL (default) use all available dimensions

Value

kernel value

Examples

```
D1 <- matrix(c(0,0,0,1,1,0,0,0,1.5, 3.5,2,2.5,3, 4, 6), ncol = 3, byrow = FALSE)
D2 <- matrix(c(0,0,1,1,0, 0, 1.2, 2, 1.4, 3.2,4.6,6.5), ncol = 3, byrow = FALSE)
K <- pssk(Dg1 = D1, Dg2 = D2, sigma = 1)
```

---

q2dr *Get derivative of 3D rotation matrix from quaternion*

Description

Get derivative of 3D rotation matrix from quaternion

Usage

```
q2dr(q)
```

Arguments

q	quaternion

Value

derivative of rotation matrix

---

q2r *Convert quaternion to rotation matrix http://en.wikipedia.org/wiki/Quaternions_and_spatial_rotation*

Description

Convert quaternion to rotation matrix http://en.wikipedia.org/wiki/Quaternions_and_spatial_rotation

Usage

```
q2r(q)
```

Arguments

q	quaternion

Value

rotation matrix

restore_coordinates *restore_coordinates*

Description

Restore coordinates from mean 0 and standard deviation 1 to their original distribution

Usage

```
restore_coordinates(X, mu, sigma)
```

Arguments

X	standardized point set as N x D matrix
mu	1 x D vector of means
sigma	standard deviation

Value

N X D matrix of unstandardized coordinates

rotx *rotx*

Description

Create a rotation matrix representing a rotation of theta radians about the x-axis

Usage

```
rotx(theta)
```

Arguments

theta	angle in radians

Value

a 3x3 rotation matrix

roty *roty*

Description

Create a rotation matrix representing a rotation of theta radians about the y-axis

Usage

```
roty(theta)
```

Arguments

theta	angle in radians

Value

a 3x3 rotation matrix

rotz *rotz*

Description

Create a rotation matrix representing a rotation of theta radians about the z-axis

Usage

```
rotz(theta)
```

Arguments

theta	angle in radians

Value

a 3x3 rotation matrix

sliced_Wd *sliced_Wd*

Description

Compute sliced Wasserstein distance or kernel. Reference: Mathieu Carriere, Marco Cuturi, and Steve Oudot. Sliced Wasserstein kernel for persistence diagrams. In Proceedings of the 34th International Conference on Machine Learning, volume 70 of Proceedings of Machine Learning Research, pages 664–673, 2017.

Usage

```
sliced_Wd(Dg1, Dg2, M = 10, sigma = 1, dimensions = NULL, return.dist = FALSE)
```

Arguments

Dg1	a persistence diagram as a n1 x 3 matrix where each row is a topological feature and the columns are dimension, birth and death of the feature.
Dg2	another persistence diagram as a n2 x 3 matrix
M	number of slices (default: 10)
sigma	kernel bandwidth (default: 1)
dimensions	vector of the dimensions of the topological features to consider, if NULL (default) use all available dimensions
return.dist	logical (default: FALSE). Whether to return the kernel or distance value.

Value

kernel or distance value

Examples

```
D1 <- matrix(c(0,0,0,1,1,0,0,0,1.5, 3.5,2,2.5,3, 4, 6), ncol = 3, byrow = FALSE)
D2 <- matrix(c(0,0,1,1,0, 0, 1.2, 2, 1.4, 3.2,4.6,6.5), ncol = 3, byrow = FALSE)
K <- sliced_Wd(Dg1 = D1, Dg2 = D2, M = 10, sigma = 1, return.dist = TRUE)
```

standardize_coordinates

standardize_coordinates

Description

Transform coordinates to have mean 0 and standard deviation 1

Usage

```
standardize_coordinates(X)
```

Arguments

X	point set as N x D matrix

Value

a list of X: standardized matrix, mu: vector of means, sigma: standard deviation

---

tr *tr*

Description

Compute the trace of a matrix

Usage

```
tr(x)
```

Arguments

x	matrix

Value

trace of the matrix

---

wgmmreg *wgmmreg*

Description

Rigid registration of two point sets by minimizing the Wasserstein distance between GMMs

Usage

```
wgmmreg(
  X,
  Y,
  CX,
  CY,
  wx = NULL,
  wy = NULL,
  maxIter = 200,
  subsample = NULL,
  tol = 1e-08
)
```

Arguments

X	reference point set, a N x D matrix
Y	point set to transform, a M x D matrix,
CX	array of covariance matrices for each point in X
CY	array of covariance matrices for each point in Y
wx	(optional) vector of mixture weights for X.
wy	(optional) vector of mixture weights for Y.
maxIter	maximum number of iterations to perform (default: 200)
subsample	if set, use this randomly selected fraction of the points
tol	tolerance for determining convergence (default: 1e-8)

Value

a list of

- Y: transformed point set,
- R: rotation matrix,
- t: translation vector,
- c: final value of the cost function,
- converged: logical, whether the algorithm converged.

Examples

```
data.file1 <- system.file("test_data", "parasaurolophusA.txt", package = "LOMAR",
 mustWork = TRUE)
PS1 <- read.csv(data.file1, sep = '\t', header = FALSE)
data.file2 <- system.file("test_data", "parasaurolophusB.txt", package = "LOMAR",
 mustWork = TRUE)
C1 <- diag(0.1, ncol(PS1)) + jitter(0.01, amount = 0.01)
C1 <- replicate(nrow(PS1),C1)
PS2 <- read.csv(data.file2, sep = '\t', header = FALSE)
C2 <- diag(0.1, ncol(PS2)) + jitter(0.01, amount = 0.01)
```

```
C2 <- replicate(nrow(PS2),C2)
transformation <- wgmmreg(PS1, PS2, C1, C2, subsample = 0.1, maxIter = 30, tol = 1e-4)
## Not run:
# Visualize registration outcome
library(rgl)
plot3d(PS1, col = "blue")
points3d(PS2, col = "green")
points3d(transformation[['Y']], col = "magenta")

## End(Not run)
```

Index