Package 'TDAkit'

January 20, 2025

Type Package

Title Toolkit for Topological Data Analysis

Version 0.1.2

Description Topological data analysis studies structure and shape of the data using topological features. We provide a variety of algorithms to learn with persistent homology of the data based on functional summaries for clustering, hypothesis testing, visualization, and others. We refer to Wasserman (2018) <doi:10.1146/annurev-statistics-031017-100045> for a statistical perspective on the topic.

License MIT + file LICENSE

Encoding UTF-8

Imports Rcpp, Rdpack, TDAstats, T4cluster, energy, ggplot2, maotai, stats, utils

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.1.1

RdMacros Rdpack

NeedsCompilation yes

Author Kisung You [aut, cre] (<https://orcid.org/0000-0002-8584-459X>), Byeongsu Yu [aut]

Maintainer Kisung You <kisungyou@outlook.com>

Repository CRAN

Date/Publication 2021-08-23 14:10:05 UTC

Contents

`diag2landscape` *Convert Persistence Diagram into Persistence Landscape*

Description

Persistence Landscape (PL) is a functional summary of persistent homology that is constructed given a `homology` object.

Usage

```
diag2landscape(homology, dimension = 1, k = 0, nseq = 1000)
```

Arguments

`homology`	an object of S3 class `"homology"` generated from `diagRips` or other homology-generating functions.
`dimension`	dimension of features to be considered (default: 1).
`k`	the number of top landscape functions to be used (default: 0). When `k=0` is set, it gives all relevant landscape functions that are non-zero.
`nseq`	grid size for which the landscape function is evaluated (default: 1000).

Value

a list object of `"landscape"` class containing

lambda an $(\texttt{nseq} \times k)$ landscape functions.

tseq a length-`nseq` vector of domain grid.

dimension dimension of features considered.

References

Peter Bubenik (2018). “The Persistence Landscape and Some of Its Properties.” *arXiv:1810.04963*.

Examples

```
# --------------------------------------------------------------------------
#              Persistence Landscape of 'iris' Dataset
#
# We will extract landscapes of dimensions 0, 1, and 2.
# For each feature, only the top 5 landscape functions are plotted.
# --------------------------------------------------------------------------
## Prepare 'iris' data
XX = as.matrix(iris[,1:4])

## Compute Persistence Diagram
pdrips = diagRips(XX, maxdim=2)

## Convert to Landscapes of Each Dimension
land0 <- diag2landscape(pdrips, dimension=0, k=5)
land1 <- diag2landscape(pdrips, dimension=1, k=5)
land2 <- diag2landscape(pdrips, dimension=2, k=5)

## Visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(2,2))
plot(pdrips$Birth, pdrips$Death, col=as.factor(pdrips$Dimension),
     pch=19, main="persistence diagram", xlab="Birth", ylab="Death")
matplot(land0$tseq, land0$lambda, type="l", lwd=3, main="dimension 0", xlab="t")
matplot(land1$tseq, land1$lambda, type="l", lwd=3, main="dimension 1", xlab="t")
matplot(land2$tseq, land2$lambda, type="l", lwd=3, main="dimension 2", xlab="t")
par(opar)
```

`diag2silhouette` *Convert Persistence Diagram into Persistent Silhouette*

Description

Persistence Silhouette (PS) is a functional summary of persistent homology that is constructed given a homology object. PS is a weighted average of landscape functions so that it becomes a uni-dimensional function.

Usage

```
diag2silhouette(homology, dimension = 1, p = 2, nseq = 100)
```

Arguments

`homology`	an object of S3 class `"homology"` generated from `diagRips` or other diagram-generating functions.
`dimension`	dimension of features to be considered (default: 1).
`p`	an exponent for the weight function of form $\lvert a-b\rvert^p$ (default: 2).
`nseq`	grid size for which the landscape function is evaluated.

Value

a list object of "`silhouette`" class containing

lambda an $(\texttt{nseq} \times k)$ landscape functions.

tseq a length-`nseq` vector of domain grid.

dimension dimension of features considered.

Examples

```
# ---------------------------------------------------------------------------
#              Persistence Silhouette of 'iris' Dataset
#
# We will extract silhouettes of dimensions 0, 1, and 2.
# ---------------------------------------------------------------------------
## Prepare 'iris' data
XX = as.matrix(iris[,1:4])

## Compute Persistence Diagram
pdrips = diagRips(XX, maxdim=2)

## Convert to Silhouettes of Each Dimension
sil0 <- diag2silhouette(pdrips, dimension=0)
sil1 <- diag2silhouette(pdrips, dimension=1)
sil2 <- diag2silhouette(pdrips, dimension=2)

## Visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(2,2))
plot(pdrips$Birth, pdrips$Death, col=as.factor(pdrips$Dimension),
     pch=19, main="persistence diagram", xlab="Birth", ylab="Death")
plot(sil0$tseq, sil0$lambda, type="l", lwd=3, main="dimension 0", xlab="t")
plot(sil1$tseq, sil1$lambda, type="l", lwd=3, main="dimension 1", xlab="t")
plot(sil2$tseq, sil2$lambda, type="l", lwd=3, main="dimension 2", xlab="t")
par(opar)
```

`diagRips` *Compute Vietoris-Rips Complex for Persistent Homology*

Description

`diagRips` computes the persistent diagram of the Vietoris-Rips filtration constructed on a point cloud represented as `matrix` or `dist` object. This function is a second-hand wrapper to **TDAstats**'s wrapping for `Ripser` library.

Usage

```
diagRips(data, maxdim = 1, threshold = Inf)
```

Arguments

`data`	a `'matrix'` or a S3 `'dist'` object.
`maxdim`	maximum dimension of the computed homological features (default: 1).
`threshold`	maximum value of the filtration (default: `Inf`).

Value

a dataframe object of S3 class `"homology"` with following columns

Dimension dimension corresponding to a feature.

Birth birth of a feature.

Death death of a feature.

References

Raoul R. Wadhwa, Drew F.K. Williamson, Andrew Dhawan, Jacob G. Scott (2018). "TDAstats: R Pipeline for Computing Persistent Homology in Topological Data Analysis." *Journal of Open Source Software*, **3**(28), 860. ISSN 2475-9066.

Ulrich Bauer (2019). "Ripser: Efficient Computation of Vietoris-Rips Persistence Barcodes." *arXiv:1908.02518*.

See Also

`calculate_homology`

Examples

```
# ---------------------------------------------------------------------------
# Check consistency of two types of inputs : 'matrix' and 'dist' objects
# ---------------------------------------------------------------------------
# Use 'iris' data and compute its distance matrix
XX = as.matrix(iris[,1:4])
DX = stats::dist(XX)

# Compute VR Diagram with two inputs
vr.mat = diagRips(XX)
vr.dis = diagRips(DX)

col1 = as.factor(vr.mat$Dimension)
col2 = as.factor(vr.dis$Dimension)

# Visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,2), pty="s")
plot(vr.mat$Birth, vr.mat$Death, pch=19, col=col1, main="from 'matrix'")
plot(vr.dis$Birth, vr.dis$Death, pch=19, col=col2, main="from 'dist'")
par(opar)
```

fsdist *Pairwise L_p Distance of Multiple Functional Summaries*

Description

Given multiple functional summaries $\Lambda_1(t), \Lambda_2(t), \ldots, \Lambda_N(t)$, compute L_p distance in a pairwise sense.

Usage

```
fsdist(fslist, p = 2, as.dist = TRUE)
```

Arguments

fslist	a length-N list of functional summaries of persistent diagrams.
p	an exponent in $[1, \infty)$ (default: 2).
as.dist	logical; if TRUE, it returns dist object, else it returns an $(N \times N)$ symmetric matrix.

Value

a S3 dist object or $(N \times N)$ symmetric matrix of pairwise distances according to as.dist parameter.

Examples

```
# ---------------------------------------------------------------------------
#      Compute L_2 Distance for 3 Types of Landscapes and Silhouettes
#
# We will compare dim=0,1 with top-5 landscape functions with
# - Class 1 : 'iris' dataset with noise
# - Class 2 : samples from 'gen2holes()'
# - Class 3 : samples from 'gen2circles()'
# ---------------------------------------------------------------------------
## Generate Data and Diagram from VR Filtration
ndata     = 10
list_rips = list()
for (i in 1:ndata){
  dat1 = as.matrix(iris[,1:4]) + matrix(rnorm(150*4), ncol=4)
  dat2 = gen2holes(n=100, sd=1)$data
  dat3 = gen2circles(n=100, sd=1)$data

  list_rips[[i]] = diagRips(dat1, maxdim=1)
  list_rips[[i+ndata]] = diagRips(dat2, maxdim=1)
  list_rips[[i+(2*ndata)]] = diagRips(dat3, maxdim=1)
}

## Compute Persistence Landscapes from Each Diagram with k=5 Functions
#  We try to get distance in dimensions 0 and 1.
```

```
list_land0 = list()
list_land1 = list()
for (i in 1:(3*ndata)){
  list_land0[[i]] = diag2landscape(list_rips[[i]], dimension=0, k=5)
  list_land1[[i]] = diag2landscape(list_rips[[i]], dimension=1, k=5)
}

## Compute Silhouettes
list_sil0 = list()
list_sil1 = list()
for (i in 1:(3*ndata)){
  list_sil0[[i]] = diag2silhouette(list_rips[[i]], dimension=0)
  list_sil1[[i]] = diag2silhouette(list_rips[[i]], dimension=1)
}

## Compute L2 Distance Matrices
ldmat0 = fsdist(list_land0, p=2, as.dist=FALSE)
ldmat1 = fsdist(list_land1, p=2, as.dist=FALSE)
sdmat0 = fsdist(list_sil0, p=2, as.dist=FALSE)
sdmat1 = fsdist(list_sil1, p=2, as.dist=FALSE)

## Visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(2,2), pty="s")
image(ldmat0[,(3*(ndata)):1], axes=FALSE, main="Landscape : dim=0")
image(ldmat1[,(3*(ndata)):1], axes=FALSE, main="Landscape : dim=1")
image(sdmat0[,(3*(ndata)):1], axes=FALSE, main="Silhouette : dim=0")
image(sdmat1[,(3*(ndata)):1], axes=FALSE, main="Silhouette : dim=1")
par(opar)
```

`fsdist2` *Pairwise L_p Distance for Two Sets of Functional Summaries*

Description

Given two sets of functional summaries $\Lambda_1(t), \ldots, \Lambda_M(t)$ and $\Omega_1(t), \ldots, \Omega_N(t)$, compute L_p distance across pairs.

Usage

```
fsdist2(fslist1, fslist2, p = 2)
```

Arguments

`fslist1`	a length-M list of functional summaries of persistent diagrams.
`fslist2`	a length-N list of functional summaries of persistent diagrams.
`p`	an exponent in $[1, \infty)$ (default: 2).

Value

an $(M \times N)$ distance matrix.

Examples

```
# --------------------------------------------------------------------------
#         Compute L1 and L2 Distance for Two Sets of Landscapes
#
# First  set consists of {Class 1, Class 2}, while
# Second set consists of {Class 1, Class 3} where
#
# - Class 1 : 'iris' dataset with noise
# - Class 2 : samples from 'gen2holes()'
# - Class 3 : samples from 'gen2circles()'
# --------------------------------------------------------------------------
## Generate Data and Diagram from VR Filtration
ndata      = 10
list_rips1 = list()
list_rips2 = list()
for (i in 1:ndata){
  dat1 = as.matrix(iris[,1:4]) + matrix(rnorm(150*4, sd=4), ncol=4)
  dat2 = gen2holes(n=100, sd=1)$data
  dat3 = as.matrix(iris[,1:4]) + matrix(rnorm(150*4, sd=4), ncol=4)
  dat4 = gen2circles(n=100, sd=1)$data

  list_rips1[[i]]       = diagRips(dat1, maxdim=1)
  list_rips1[[i+ndata]] = diagRips(dat2, maxdim=1)

  list_rips2[[i]]       = diagRips(dat3, maxdim=1)
  list_rips2[[i+ndata]] = diagRips(dat4, maxdim=1)
}

## Compute Persistence Landscapes from Each Diagram with k=10 Functions
#  We try to get distance in dimension 1 only for faster comparison.
list_pset1 = list()
list_pset2 = list()
for (i in 1:(2*ndata)){
  list_pset1[[i]] = diag2landscape(list_rips1[[i]], dimension=1, k=10)
  list_pset2[[i]] = diag2landscape(list_rips2[[i]], dimension=1, k=10)
}

## Compute L1 and L2 Distance Matrix
dmat1 = fsdist2(list_pset1, list_pset2, p=1)
dmat2 = fsdist2(list_pset1, list_pset2, p=2)

## Visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,2), pty="s")
image(dmat1[,(2*ndata):1], axes=FALSE, main="distance for p=1")
image(dmat2[,(2*ndata):1], axes=FALSE, main="distance for p=2")
par(opar)
```

fseqdist *Multi-sample Energy Test of Equal Distributions*

Description

Also known as k-sample problem, it tests whether multiple functional summaries are equally distributed or not via Energy statistics.

Usage

```
fseqdist(fslist, label, method = c("original", "disco"), mc.iter = 999)
```

Arguments

fslist	a length-N list of functional summaries of persistent diagrams.
label	a length-N vector of class labels.
method	(case-sensitive) name of methods; one of "original" or "disco".
mc.iter	number of bootstrap replicates.

Value

a (list) object of S3 class htest containing:

method name of the test.

statistic a test statistic.

p.value p-value under H_0 of equal distributions.

Examples

```
# ---------------------------------------------------------------------------
#         Test for Equality of Distributions via Energy Statistics
#
# We will compare dim=0's top-5 landscape functions with
# - Class 1 : 'iris' dataset with noise
# - Class 2 : samples from 'gen2holes()'
# - Class 3 : samples from 'gen2circles()'
# ---------------------------------------------------------------------------
## Generate Data and Diagram from VR Filtration
ndata     = 10
list_rips = list()
for (i in 1:ndata){
  dat1 = as.matrix(iris[,1:4]) + matrix(rnorm(150*4), ncol=4)
  dat2 = gen2holes(n=100, sd=1)$data
  dat3 = gen2circles(n=100, sd=1)$data

  list_rips[[i]] = diagRips(dat1, maxdim=1)
```

```
  list_rips[[i+ndata]] = diagRips(dat2, maxdim=1)
  list_rips[[i+(2*ndata)]] = diagRips(dat3, maxdim=1)
}

## Compute Persistence Landscapes from Each Diagram with k=5 Functions
list_land0 = list()
for (i in 1:(3*ndata)){
  list_land0[[i]] = diag2landscape(list_rips[[i]], dimension=0, k=5)
}

## Create Label and Run the Test with Different Options
list_lab = c(rep(1,ndata), rep(2,ndata), rep(3,ndata))
fseqdist(list_land0, list_lab, method="original")
fseqdist(list_land0, list_lab, method="disco")
```

`fshclust` *Hierarchical Agglomerative Clustering*

Description

Given multiple functional summaries $\Lambda_1(t), \Lambda_2(t), \ldots, \Lambda_N(t)$, perform hierarchical agglomerative clustering with L_2 distance.

Usage

```
fshclust(
  fslist,
  method = c("single", "complete", "average", "mcquitty", "ward.D", "ward.D2",
    "centroid", "median"),
  members = NULL
)
```

Arguments

`fslist`	a length-N list of functional summaries of persistent diagrams.
`method`	agglomeration method to be used. This must be one of `"single"`, `"complete"`, `"average"`, `"mcquitty"`, `"ward.D"`, `"ward.D2"`, `"centroid"` or `"median"`.
`members`	`NULL` or a vector whose length equals the number of observations. See `hclust` for details.

Value

an object of class `hclust`. See `hclust` for details.

Examples

```
# ---------------------------------------------------------------------------
#           K-Groups Clustering via Energy Distance
#
# We will cluster dim=0 under top-5 landscape functions with
# - Class 1 : 'iris' dataset with noise
# - Class 2 : samples from 'gen2holes()'
# - Class 3 : samples from 'gen2circles()'
# ---------------------------------------------------------------------------
## Generate Data and Diagram from VR Filtration
ndata     = 10
list_rips = list()
for (i in 1:ndata){
  dat1 = as.matrix(iris[,1:4]) + matrix(rnorm(150*4), ncol=4)
  dat2 = gen2holes(n=100, sd=1)$data
  dat3 = gen2circles(n=100, sd=1)$data

  list_rips[[i]] = diagRips(dat1, maxdim=1)
  list_rips[[i+ndata]] = diagRips(dat2, maxdim=1)
  list_rips[[i+(2*ndata)]] = diagRips(dat3, maxdim=1)
}
list_lab = c(rep(1,ndata), rep(2,ndata), rep(3,ndata))

## Compute Persistence Landscapes from Each Diagram with k=5 Functions
list_land0 = list()
for (i in 1:(3*ndata)){
  list_land0[[i]] = diag2landscape(list_rips[[i]], dimension=0, k=5)
}

## Run MDS for Visualization
embed = fsmds(list_land0, ndim=2)

## Clustering with 'single' and 'complete' linkage
hc.sing <- fshclust(list_land0, method="single")
hc.comp <- fshclust(list_land0, method="complete")

## Visualize
opar  = par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(embed, pch=19, col=list_lab, main="2-dim embedding")
plot(hc.sing, main="single linkage")
plot(hc.comp, main="complete linkage")
par(opar)
```

---

fskgroups — *k-Groups Clustering of Multiple Functional Summaries by Energy Distance*

Description

Given N functional summaries $\Lambda_1(t), \Lambda_2(t), \ldots, \Lambda_N(t)$, perform k-groups clustering by energy distance using L_2 metric.

Usage

```
fskgroups(fslist, k = 2, ...)
```

Arguments

`fslist`	a length-N list of functional summaries of persistent diagrams.
`k`	the number of clusters.
`...`	extra parameters including **maxiter** the number of iterations (default: 50). **nstart** the number of restarts (default: 2).

Value

a length-N vector of class labels (from $1 : k$).

Examples

```
# -------------------------------------------------------------------------
#           K-Groups Clustering via Energy Distance
#
# We will cluster dim=0 under top-5 landscape functions with
# - Class 1 : 'iris' dataset with noise
# - Class 2 : samples from 'gen2holes()'
# - Class 3 : samples from 'gen2circles()'
# -------------------------------------------------------------------------
## Generate Data and Diagram from VR Filtration
ndata     = 10
list_rips = list()
for (i in 1:ndata){
  dat1 = as.matrix(iris[,1:4]) + matrix(rnorm(150*4), ncol=4)
  dat2 = gen2holes(n=100, sd=1)$data
  dat3 = gen2circles(n=100, sd=1)$data

  list_rips[[i]] = diagRips(dat1, maxdim=1)
  list_rips[[i+ndata]] = diagRips(dat2, maxdim=1)
  list_rips[[i+(2*ndata)]] = diagRips(dat3, maxdim=1)
}

## Compute Persistence Landscapes from Each Diagram with k=5 Functions
list_land0 = list()
for (i in 1:(3*ndata)){
  list_land0[[i]] = diag2landscape(list_rips[[i]], dimension=0, k=5)
}

## Run K-Groups Clustering with different K's
```

```
label2  = fskgroups(list_land0, k=2)
label3  = fskgroups(list_land0, k=3)
label4  = fskgroups(list_land0, k=4)
truelab = rep(c(1,2,3), each=ndata)

## Run MDS & Visualization
embed = fsmds(list_land0, ndim=2)
opar  = par(no.readonly=TRUE)
par(mfrow=c(2,2), pty="s")
plot(embed, col=truelab, pch=19, main="true label")
plot(embed, col=label2,  pch=19, main="k=2 label")
plot(embed, col=label3,  pch=19, main="k=3 label")
plot(embed, col=label4,  pch=19, main="k=4 label")
par(opar)
```

---

fskmedoids *K-Medoids Clustering*

Description

Given N functional summaries $\Lambda_1(t), \Lambda_2(t), \ldots, \Lambda_N(t)$, perform k-medoids clustering using pairwise distances using L_2 metric.

Usage

```
fskmedoids(fslist, k = 2)
```

Arguments

fslist	a length-N list of functional summaries of persistent diagrams.
k	the number of clusters.

Value

a length-N vector of class labels (from $1:k$).

Examples

```
# ---------------------------------------------------------------------------
#            K-Groups Clustering via Energy Distance
#
# We will cluster dim=0 under top-5 landscape functions with
# - Class 1 : 'iris' dataset with noise
# - Class 2 : samples from 'gen2holes()'
# - Class 3 : samples from 'gen2circles()'
# ---------------------------------------------------------------------------
## Generate Data and Diagram from VR Filtration
```

```
ndata     = 10
list_rips = list()
for (i in 1:ndata){
  dat1 = as.matrix(iris[,1:4]) + matrix(rnorm(150*4), ncol=4)
  dat2 = gen2holes(n=100, sd=1)$data
  dat3 = gen2circles(n=100, sd=1)$data

  list_rips[[i]] = diagRips(dat1, maxdim=1)
  list_rips[[i+ndata]] = diagRips(dat2, maxdim=1)
  list_rips[[i+(2*ndata)]] = diagRips(dat3, maxdim=1)
}

## Compute Persistence Landscapes from Each Diagram with k=5 Functions
list_land0 = list()
for (i in 1:(3*ndata)){
  list_land0[[i]] = diag2landscape(list_rips[[i]], dimension=0, k=5)
}

## Run K-Medoids Clustering with different K's
label2  = fskmedoids(list_land0, k=2)
label3  = fskmedoids(list_land0, k=3)
label4  = fskmedoids(list_land0, k=4)
truelab = rep(c(1,2,3), each=ndata)

## Run MDS & Visualization
embed = fsmds(list_land0, ndim=2)
opar  = par(no.readonly=TRUE)
par(mfrow=c(2,2), pty="s")
plot(embed, col=truelab, pch=19, main="true label")
plot(embed, col=label2,  pch=19, main="k=2 label")
plot(embed, col=label3,  pch=19, main="k=3 label")
plot(embed, col=label4,  pch=19, main="k=4 label")
par(opar)
```

---

fsmds *Multidimensional Scaling*

Description

Given multiple functional summaries $\Lambda_1(t), \Lambda_2(t), \ldots, \Lambda_N(t)$, apply multidimensional scaling to get low-dimensional representation in Euclidean space. Usually, `ndim=2,3` is chosen for visualization.

Usage

```
fsmds(fslist, ndim = 2, method = c("classical", "metric"))
```

Arguments

`fslist`	a length-N list of functional summaries of persistent diagrams.
`ndim`	an integer-valued target dimension (default: 2).
`method`	name of an algorithm type (default: classical).

Value

an $(N \times ndim)$ matrix of embedding.

Examples

```
# --------------------------------------------------------------------------
#     Multidimensional Scaling for Multiple Landscapes and Silhouettes
#
# We will compare dim=0 with top-5 landscape and silhouette functions with
# - Class 1 : 'iris' dataset with noise
# - Class 2 : samples from 'gen2holes()'
# - Class 3 : samples from 'gen2circles()'
# --------------------------------------------------------------------------
## Generate Data and Diagram from VR Filtration
ndata     = 10
list_rips = list()
for (i in 1:ndata){
  dat1 = as.matrix(iris[,1:4]) + matrix(rnorm(150*4), ncol=4)
  dat2 = gen2holes(n=100, sd=1)$data
  dat3 = gen2circles(n=100, sd=1)$data

  list_rips[[i]] = diagRips(dat1, maxdim=1)
  list_rips[[i+ndata]] = diagRips(dat2, maxdim=1)
  list_rips[[i+(2*ndata)]] = diagRips(dat3, maxdim=1)
}

## Compute Landscape and Silhouettes of Dimension 0
list_land = list()
list_sils = list()
for (i in 1:(3*ndata)){
  list_land[[i]] = diag2landscape(list_rips[[i]], dimension=0)
  list_sils[[i]] = diag2silhouette(list_rips[[i]], dimension=0)
}
list_lab = rep(c(1,2,3), each=ndata)

## Run Classical/Metric Multidimensional Scaling
land_cmds = fsmds(list_land, method="classical")
land_mmds = fsmds(list_land, method="metric")
sils_cmds = fsmds(list_sils, method="classical")
sils_mmds = fsmds(list_sils, method="metric")

## Visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(2,2))
plot(land_cmds, pch=19, col=list_lab, main="Landscape+CMDS")
```

```
plot(land_mmds, pch=19, col=list_lab, main="Landscape+MMDS")
plot(sils_cmds, pch=19, col=list_lab, main="Silhouette+CMDS")
plot(sils_mmds, pch=19, col=list_lab, main="Silhouette+MMDS")
par(opar)
```

`fsmean` *Mean of Multiple Functional Summaries*

Description

Given multiple functional summaries $\Lambda_1(t), \Lambda_2(t), \ldots, \Lambda_N(t)$, compute the mean

$$\bar{\Lambda}(t) = \frac{1}{N} \sum_{n=1}^{N} \Lambda_n(t)$$

.

Usage

```
fsmean(fslist)
```

Arguments

`fslist`	a length-N list of functional summaries of persistent diagrams.

Value

a functional summary object.

Examples

```
# ---------------------------------------------------------------------------
#         Mean of 10 Persistence Landscapes from '2holes' data
# ---------------------------------------------------------------------------
## Generate 10 Diagrams with 'gen2holes()' function
list_rips = list()
for (i in 1:10){
  list_rips[[i]] = diagRips(gen2holes(n=100, sd=2)$data, maxdim=1)
}

## Compute Persistence Landscapes from Each Diagram with k=5 Functions
list_land = list()
for (i in 1:10){
  list_land[[i]] = diag2landscape(list_rips[[i]], dimension=0, k=5)
}

## Compute Weighted Sum of Landscapes
ldsum = fsmean(list_land)
```

```
## Visualize
sam5  <- sort(sample(1:10, 5, replace=FALSE))
opar  <- par(no.readonly=TRUE)
par(mfrow=c(2,3), pty="s")
for (i in 1:5){
  tgt = list_land[[sam5[i]]]
 matplot(tgt$tseq, tgt$lambda[,1:5], type="l", lwd=3, main=paste("landscape no.",sam5[i]))
}
matplot(ldsum$tseq, ldsum$lambda[,1:5], type="l", lwd=3, main="weighted sum")
par(opar)
```

---

fsnorm *L_p Norm of a Single Functional Summary*

Description

Given a functional summary $\Lambda(t)$, compute the p-norm.

Usage

```
fsnorm(fsobj, p = 2)
```

Arguments

fsobj	a functional summary object.
p	an exponent in $[1, \infty)$ (default: 2).

Value

an L_p-norm value.

Examples

```
## Generate Toy Data from 'gen2circles()'
dat = gen2circles(n=100)$data

## Compute PD, Landscapes, and Silhouettes
myPD  = diagRips(dat, maxdim=1)
myPL0 = diag2landscape(myPD, dimension=0)
myPL1 = diag2landscape(myPD, dimension=1)
myPS0 = diag2silhouette(myPD, dimension=0)
myPS1 = diag2silhouette(myPD, dimension=1)

## Compute 2-norm
fsnorm(myPL0, p=2)
fsnorm(myPL1, p=2)
fsnorm(myPS0, p=2)
```

```
fsnorm(myPS1, p=2)
```

---

fssc05Z *Spectral Clustering by Zelnik-Manor and Perona (2005)*

Description

Given N functional summaries $\Lambda_1(t), \Lambda_2(t), \ldots, \Lambda_N(t)$, perform spectral clustering proposed by Zelnik-Manor and Perona using a set of data-driven bandwidth parameters.

Usage

```
fssc05Z(fslist, k = 2, nnbd = 5)
```

Arguments

fslist	a length-N list of functional summaries of persistent diagrams.
k	the number of cluster (default: 2).
nnbd	neighborhood size to define data-driven bandwidth parameter (default: 5).

Value

a length-N vector of class labels (from $1 : k$).

References

Zelnik-manor L, Perona P (2005). "Self-Tuning Spectral Clustering." In Saul LK, Weiss Y, Bottou L (eds.), *Advances in Neural Information Processing Systems 17*, 1601–1608. MIT Press.

Examples

```
# ----------------------------------------------------------------------------
#           Spectral Clustering Clustering via Energy Distance
#
# We will cluster dim=0 under top-5 landscape functions with
# - Class 1 : 'iris' dataset with noise
# - Class 2 : samples from 'gen2holes()'
# - Class 3 : samples from 'gen2circles()'
# ----------------------------------------------------------------------------
## Generate Data and Diagram from VR Filtration
ndata     = 10
list_rips = list()
for (i in 1:ndata){
  dat1 = as.matrix(iris[,1:4]) + matrix(rnorm(150*4), ncol=4)
  dat2 = gen2holes(n=100, sd=1)$data
  dat3 = gen2circles(n=100, sd=1)$data
```

```
  list_rips[[i]] = diagRips(dat1, maxdim=1)
  list_rips[[i+ndata]] = diagRips(dat2, maxdim=1)
  list_rips[[i+(2*ndata)]] = diagRips(dat3, maxdim=1)
}

## Compute Persistence Landscapes from Each Diagram with k=5 Functions
list_land0 = list()
for (i in 1:(3*ndata)){
  list_land0[[i]] = diag2landscape(list_rips[[i]], dimension=0, k=5)
}

## Run Spectral Clustering using Different K's.
label2  = fssc05Z(list_land0, k=2)
label3  = fssc05Z(list_land0, k=3)
label4  = fssc05Z(list_land0, k=4)
truelab = rep(c(1,2,3), each=ndata)

## Run MDS & Visualization
embed = fsmds(list_land0, ndim=2)
opar  = par(no.readonly=TRUE)
par(mfrow=c(2,2), pty="s")
plot(embed, col=truelab, pch=19, main="true label")
plot(embed, col=label2,  pch=19, main="k=2 label")
plot(embed, col=label3,  pch=19, main="k=3 label")
plot(embed, col=label4,  pch=19, main="k=4 label")
par(opar)
```

`fssum` *Weighted Sum of Multiple Functional Summaries*

Description

Given multiple functional summaries $\Lambda_1(t), \Lambda_2(t), \ldots, \Lambda_N(t)$, compute the weighted sum

$$\bar{\Lambda}(t) = \sum_{n=1}^{N} w_n \Lambda_n(t)$$

with a specified vector of given weights $w_1, w_2, \ldots, w_N$.

Usage

```
fssum(fslist, weight = NULL)
```

Arguments

`fslist`	a length-N list of functional summaries of persistent diagrams.
`weight`	a weight vector of length N. If `NULL` (default), weights are automatically set as $w_1 = \cdots = w_N = 1/N$.

Value

a functional summary object.

Examples

```
# ---------------------------------------------------------------------------
#     Weighted Average of 10 Persistence Landscapes from '2holes' data
# ---------------------------------------------------------------------------
## Generate 10 Diagrams with 'gen2holes()' function
list_rips = list()
for (i in 1:10){
  list_rips[[i]] = diagRips(gen2holes(n=100, sd=2)$data, maxdim=1)
}

## Compute Persistence Landscapes from Each Diagram with k=5 Functions
list_land = list()
for (i in 1:10){
  list_land[[i]] = diag2landscape(list_rips[[i]], dimension=0, k=5)
}

## Some Random Weights
wrand = abs(stats::rnorm(10))
wrand = wrand/sum(wrand)

## Compute Weighted Sum of Landscapes
ldsum = fssum(list_land, weight=wrand)

## Visualize
sam5  <- sort(sample(1:10, 5, replace=FALSE))
opar  <- par(no.readonly=TRUE)
par(mfrow=c(2,3), pty="s")
for (i in 1:5){
  tgt = list_land[[sam5[i]]]
 matplot(tgt$tseq, tgt$lambda[,1:5], type="l", lwd=3, main=paste("landscape no.",sam5[i]))
}
matplot(ldsum$tseq, ldsum$lambda[,1:5], type="l", lwd=3, main="weighted sum")
par(opar)
```

`fstsne` *t-distributed Stochastic Neighbor Embedding*

Description

Given N functional summaries $\Lambda_1(t), \Lambda_2(t), \ldots, \Lambda_N(t)$, t-SNE mimicks the pattern of probability distributions over pairs of Banach-valued objects on low-dimensional target embedding space by minimizing Kullback-Leibler divergence.

Usage

```
fstsne(fslist, ndim = 2, ...)
```

Arguments

fslist	a length-N list of functional summaries of persistent diagrams.
ndim	an integer-valued target dimension.
...	extra parameters for Rtsne algorithm, such as perplexity, momentum, and others.

Value

a named list containing

embed an $(N \times ndim)$ matrix whose rows are embedded observations.

stress discrepancy between embedded and original distances as a measure of error.

See Also

Rtsne

Examples

```
# ---------------------------------------------------------------------------
#     Multidimensional Scaling for Multiple Landscapes and Silhouettes
#
# We will compare dim=0 with top-5 landscape and silhouette functions with
# - Class 1 : 'iris' dataset with noise
# - Class 2 : samples from 'gen2holes()'
# - Class 3 : samples from 'gen2circles()'
# ---------------------------------------------------------------------------
## Generate Data and Diagram from VR Filtration
ndata     = 10
list_rips = list()
for (i in 1:ndata){
  dat1 = as.matrix(iris[,1:4]) + matrix(rnorm(150*4), ncol=4)
  dat2 = gen2holes(n=100, sd=1)$data
  dat3 = gen2circles(n=100, sd=1)$data

  list_rips[[i]] = diagRips(dat1, maxdim=1)
  list_rips[[i+ndata]] = diagRips(dat2, maxdim=1)
  list_rips[[i+(2*ndata)]] = diagRips(dat3, maxdim=1)
}

## Compute Landscape and Silhouettes of Dimension 0
list_land = list()
list_sils = list()
for (i in 1:(3*ndata)){
  list_land[[i]] = diag2landscape(list_rips[[i]], dimension=0)
  list_sils[[i]] = diag2silhouette(list_rips[[i]], dimension=0)
}
```

```
list_lab = rep(c(1,2,3), each=ndata)

## Run t-SNE and Classical/Metric MDS
land_cmds = fsmds(list_land, method="classical")
land_mmds = fsmds(list_land, method="metric")
land_tsne = fstsne(list_land, perplexity=5)$embed
sils_cmds = fsmds(list_sils, method="classical")
sils_mmds = fsmds(list_sils, method="metric")
sils_tsne = fstsne(list_land, perplexity=5)$embed

## Visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(2,3))
plot(land_cmds, pch=19, col=list_lab, main="Landscape+CMDS")
plot(land_mmds, pch=19, col=list_lab, main="Landscape+MMDS")
plot(land_tsne, pch=19, col=list_lab, main="Landscape+tSNE")
plot(sils_cmds, pch=19, col=list_lab, main="Silhouette+CMDS")
plot(sils_mmds, pch=19, col=list_lab, main="Silhouette+MMDS")
plot(sils_tsne, pch=19, col=list_lab, main="Silhouette+tSNE")
par(opar)
```

---

gen2circles *Generate Two Intersecting Circles*

Description

It generates data from two intersecting circles.

Usage

```
gen2circles(n = 496, sd = 0)
```

Arguments

n	the total number of observations to be generated.
sd	level of additive white noise.

Value

a list containing

data an $(n \times 2)$ data matrix for row-stacked observations.

label a length-n vector for class label.

Examples

```
## Generate Data with Different Noise Levels
nn = 200
x1 = gen2circles(n=nn, sd=0)
x2 = gen2circles(n=nn, sd=0.1)
x3 = gen2circles(n=nn, sd=0.25)

## Visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3), pty="s")
plot(x1$data, pch=19, main="sd=0.00", col=x1$label)
plot(x2$data, pch=19, main="sd=0.10", col=x2$label)
plot(x3$data, pch=19, main="sd=0.25", col=x3$label)
par(opar)
```

---

gen2holes — *Generate Two Intertwined Holes*

Description

It generates data from two intertwine circles with empty interiors(holes).

Usage

```
gen2holes(n = 496, sd = 0)
```

Arguments

n	the total number of observations to be generated.
sd	level of additive white noise.

Value

a list containing

data an $(n \times 2)$ data matrix for row-stacked observations.

label a length-n vector for class label.

Examples

```
## Generate Data with Different Noise Levels
nn = 200
x1 = gen2holes(n=nn, sd=0)
x2 = gen2holes(n=nn, sd=0.1)
x3 = gen2holes(n=nn, sd=0.25)

## Visualize
opar <- par(no.readonly=TRUE)
```

```
par(mfrow=c(1,3), pty="s")
plot(x1$data, pch=19, main="sd=0.00", col=x1$label)
plot(x2$data, pch=19, main="sd=0.10", col=x2$label)
plot(x3$data, pch=19, main="sd=0.25", col=x3$label)
par(opar)
```

---

plkernel *Persistence Landscape Kernel*

Description

Given multiple persistence landscapes $\Lambda_1(t), \Lambda_2(t), \ldots, \Lambda_N(t)$, compute the persistence landscape kernel under the L_2 sense.

Usage

```
plkernel(landlist)
```

Arguments

`landlist`	a length-N list of `"landscape"` objects, which can be obtained from `diag2landscape` function.

Value

an $(N \times N)$ kernel matrix.

References

Jan Reininghaus, Stefan Huber, Ulrich Bauer, and Roland Kwitt (2015). "A stable multi-scale kernel for topological machine learning." *Proc. 2015 IEEE Conf. Comp. Vision & Pat. Rec. (CVPR '15).*

Examples

```
# ---------------------------------------------------------------------------
#      Persistence Landscape Kernel in Dimension 0 and 1
#
# We will compare dim=0,1 with top-20 landscape functions with
# - Class 1 : 'iris' dataset with noise
# - Class 2 : samples from 'gen2holes()'
# - Class 3 : samples from 'gen2circles()'
# ---------------------------------------------------------------------------
## Generate Data and Diagram from VR Filtration
ndata     = 10
list_rips = list()
for (i in 1:ndata){
  dat1 = as.matrix(iris[,1:4]) + matrix(rnorm(150*4), ncol=4)
  dat2 = gen2holes(n=100, sd=1)$data
```

```
  dat3 = gen2circles(n=100, sd=1)$data

  list_rips[[i]] = diagRips(dat1, maxdim=1)
  list_rips[[i+ndata]] = diagRips(dat2, maxdim=1)
  list_rips[[i+(2*ndata)]] = diagRips(dat3, maxdim=1)
}

## Compute Persistence Landscapes from Each Diagram with k=5 Functions
#  We try to get distance in dimensions 0 and 1.
list_land0 = list()
list_land1 = list()
for (i in 1:(3*ndata)){
  list_land0[[i]] = diag2landscape(list_rips[[i]], dimension=0, k=5)
  list_land1[[i]] = diag2landscape(list_rips[[i]], dimension=1, k=5)
}

## Compute Persistence Landscape Kernel Matrix
plk0 <- plkernel(list_land0)
plk1 <- plkernel(list_land1)

## Visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,2), pty="s")
image(plk0[,(3*(ndata)):1], axes=FALSE, main="Kernel : dim=0")
image(plk1[,(3*(ndata)):1], axes=FALSE, main="Kernel : dim=1")
par(opar)
```

`plot.homology` *Plot Persistent Homology via Barcode or Diagram*

Description

Given a persistent homology of the data represented by a reconstructed complex in S3 class `homology` object, visualize it as either a barcode or a persistence diagram using **ggplot2**.

Usage

```
## S3 method for class 'homology'
plot(x, ...)
```

Arguments

`x`	a `homology` object.
`...`	extra parameters including **method** type of visualization; either `"barcode"` or `"diagram"`.

Value

a **ggplot2** object.

Examples

```
# Use 'iris' data
XX = as.matrix(iris[,1:4])

# Compute VR Diagram
homology = diagRips(XX)

# Plot with 'barcode'
opar <- par(no.readonly=TRUE)
plot(homology, method="barcode")
par(opar)
```

---

plot.landscape *Plot Persistence Landscape*

Description

Given a persistence landscape object in S3 class `landscape`, visualize the landscapes using **ggplot2**.

Usage

```
## S3 method for class 'landscape'
plot(x, ...)
```

Arguments

`x`	a `landscape` object.
`...`	extra parameters including **top.k** the number of landscapes to be plotted (default: 5). **colored** a logical; `TRUE` to assign different colors for landscapes, or `FALSE` to use grey color for all landscapes.

Value

a **ggplot2** object.

Examples

```
# Use 'iris' data
XX = as.matrix(iris[,1:4])

# Compute Persistence diagram and landscape of order 0
homology  = diagRips(XX)
landscape = diag2landscape(homology, dimension=0)

# Plot with 'barcode'
opar <- par(no.readonly=TRUE)
plot(landscape)
par(opar)
```

Index