Package 'TDAvec'

August 5, 2022

Type Package

Title Vector Summaries of Persistence Diagrams

Version 0.1.0

Author Umar Islambekov, Alexey Luchinsky

Maintainer Alexey Luchinsky <aluchi@bgsu.edu>

Description
Tools for computing various vector summaries of persistence diagrams studied in Topological Data Analysis.

License GPL (>= 2)

Encoding UTF-8

LazyData true

Imports Rcpp (>= 1.0.9), TDA

LinkingTo Rcpp

Suggests knitr

VignetteBuilder knitr

NeedsCompilation yes

RoxygenNote 7.2.0

R topics documented:

computeECC *computeECC*

Description

computeECC

Usage

```
computeECC(D, maxhomDim, scaleSeq)
```

Examples

```
N <- 100
set.seed(123)
X <- TDA::circleUnif(N) + rnorm(2*N,mean = 0,sd = 0.2)
D <- TDA::ripsDiag(X,maxdimension = 1,maxscale = 2)$diagram
scaleSeq = seq(0,2,length.out=11)
computeECC(D,maxhomDim = 1,scaleSeq)
```

computePES *computePES*

Description

computePES

Usage

```
computePES(D, homDim, scaleSeq)
```

Examples

```
N <- 100
set.seed(123)
X <- TDA::circleUnif(N) + rnorm(2*N,mean = 0,sd = 0.2)
D <- TDA::ripsDiag(X,maxdimension = 1,maxscale = 2)$diagram # compute PD using Rips filtration
scaleSeq = seq(0,2,length.out=11) # sequence of scale values
computePES(D,homDim = 0,scaleSeq) # compute PES for homological dimension H0
computePES(D,homDim = 1,scaleSeq) # compute PES for homological dimension H1
```

computePI *Calculates the Persistence Image*

Description

Calculates the Persistence Image

Usage

```
computePI(D, homDim, res, sigma, minB, maxB, minP, maxP)
```

Arguments

D	N by 3 matrix (columns contain dimension, birth and persistence values respectively)
homDim	homological dimension (0 for H0, 1 for H1, etc.)
res	resolution parameter
sigma	sigma parameter
minB	minimal birth value
maxB	maximal birth value
minP	minimal persistance value
maxP	maxzimal persistance value

Examples

```
N <- 100
set.seed(123)
X <- TDA::circleUnif(N) + rnorm(2*N,mean = 0,sd = 0.2)
D <- TDA::ripsDiag(X,maxdimension = 1,maxscale = 2)$diagram
D[,3] <- D[,3] - D[,2]
colnames(D)[3] <- "Persistence"
res <- 5 # resolution or grid size

minPH0 <- min(D[D[,1]==0,3]); maxPH0 <- max(D[D[,1]==0,3])
sigma <- 0.5*(maxPH0-minPH0)/res
computePI(D,homDim=0,res,sigma,minB=NA,maxB=NA,minPH0,maxPH0)

minBH1 <- min(D[D[,1]==1,2]); maxBH1 <- max(D[D[,1]==1,2])
minPH1 <- min(D[D[,1]==1,3]); maxPH1 <- max(D[D[,1]==1,3])
sigma <- 0.5*(maxPH1-minPH1)/res # default way of selecting sigma - can be overridden
computePI(D,homDim=1,res,sigma,minBH1,maxBH1,minPH1,maxPH1)
```

computePL *A Vector Summary of the Persistence Landscape Function*

Description

Vectorizes the persistence landscape function computed from a given persistence diagram.

Usage

```
computePL(D, homDim, k, scaleSeq)
```

Arguments

D	N by 3 matrix (columns contain dimension, birth and death values respectively)
homDim	homological dimension (0 for H_0, 1 for H_1, etc.)
k	order of landscape function
scaleSeq	sequence of increasing scale values for vectorization

Value

A numeric vector where i-th element is the value of k-th order landscape function evaluated at `scaleSeq[i]`.

References

1. Bubenik, P. (2015). Statistical topological data analysis using persistence landscapes. Journal of Machine Learning Research, 16(1), 77-102.

2. Chazal, F., Fasy, B. T., Lecci, F., Rinaldo, A., & Wasserman, L. (2014, June). Stochastic convergence of persistence landscapes and silhouettes. In Proceedings of the thirtieth annual symposium on Computational geometry (pp. 474-483).

Examples

```
N <- 100
set.seed(123)
# sample N points uniformly from unit circle and add Gaussian noise
X <- TDA::circleUnif(N,r=1) + rnorm(2*N,mean = 0,sd = 0.2)

# compute a persistence diagram using the Rips filtration built on top of X
D <- TDA::ripsDiag(X,maxdimension = 1,maxscale = 2)$diagram

scaleSeq = seq(0,2,length.out=11) # sequence of scale values

# compute persistence landscape (PL) for homological dimension H0 with order of landscape k=1
computePL(D,homDim=0,k=1,scaleSeq)

# compute persistence landscape (PL) for homological dimension H1 with order of landscape k=1
computePL(D,homDim=1,k=1,scaleSeq)
aesetredfgsd sd
```

computePS *Calculates the Persistence Silhouettes*

Description

Calculates the Persistence Silhouettes

Usage

```
computePS(D, homDim, p, scaleSeq)
```

Arguments

D	N by 3 matrix (columns contain dimension, birth and persistence values respectively)
homDim	homological dimension (0 for H0, 1 for H1, etc.)
p	power of the weights for the silhouette function
scaleSeq	sequence of scale values for vectorization

Examples

```
N <- 100
set.seed(123)
X <- TDA::circleUnif(N) + rnorm(2*N,mean = 0,sd = 0.2)
D <- TDA::ripsDiag(X,maxdimension = 1,maxscale = 2)$diagram
scaleSeq = seq(0,2,length.out=11) # sequence of scale values
computePS(D,homDim=0,p=1,scaleSeq)
```

computeVAB *Vector Persistense Blocks*

Description

Vector Persistense Blocks

Usage

```
computeVAB(D, homDim, scaleSeq)
```

Arguments

D	N by 3 matrix (columns contain dimension, birth and persistence values respectively)
homDim	homological dimension (0 for H0, 1 for H1, etc.)
scaleSeq	sequence of scale values for vectorization

Examples

```
N <- 100
set.seed(123)
X <- TDA::circleUnif(N) + rnorm(2*N,mean = 0,sd = 0.2)
D <- TDA::ripsDiag(X,maxdimension = 1,maxscale = 2)$diagram
scaleSeq = seq(0,2,length.out=11)
computeVAB(D,homDim = 0,scaleSeq)
computeVAB(D,homDim = 1,scaleSeq)
```

---

computeVPB *Calculates the Vector Persistence Block*

Description

Calculates the Vector Persistence Block

Usage

```
computeVPB(D, homDim, xSeq, ySeq, tau)
```

Arguments

D	N by 3 matrix (columns contain dimension, birth and persistence values respectively)
homDim	homological dimension (0 for H0, 1 for H1, etc.)
xSeq	sequence of x (birth) values of the grid vertices
ySeq	sequence of y (persistence) values of the grid vertices
tau	parameter (between 0 and 1) controlling block width. weight function w(x,y) = x+y

Examples

```
X <- TDAstats::circle2d + rnorm(200,mean = 0,sd = 0.1)
D <- TDAstats::calculate_homology(X,dim = 1,threshold = 2)
D[,3] <- D[,3] - D[,2]
xSeq <- unique(quantile(D[D[,1]==1,2],probs = seq(0,1,by=0.1)))
ySeq <- unique(quantile(D[D[,1]==1,3],probs = seq(0,1,by=0.1)))
computeVPB(D,homDim = 0, xSeq=xSeq, ySeq,tau = 0.5)
```

Index