Package: TDAvec (via r-universe)

October 28, 2026

Type Package

Title Vector Summaries of Persistence Diagrams

Version 0.1.41

Maintainer Aleksei Luchinsky <aluchi@bgsu.edu>

Description Provides tools for computing various vector summaries of persistence diagrams studied in Topological Data Analysis. For improved computational efficiency, all code for the vector summaries is written in 'C++' using the 'Rcpp' and 'RcppArmadillo' packages.

License GPL (>= 2)

Encoding UTF-8

Imports Rcpp (>= 1.0.5)

LinkingTo Rcpp, RcppArmadillo

Suggests knitr, TDAstats

VignetteBuilder knitr

URL https://github.com/uislambekov/TDAvec

BugReports https://github.com/uislambekov/TDAvec/issues

NeedsCompilation yes

Author Umar Islambekov [aut], Aleksei Luchinsky [aut, cre]

Repository https://cran.r-universe.dev

Date/Publication 2025-04-02 11:40:01 UTC

RemoteUrl https://github.com/cran/TDAvec

RemoteRef HEAD

RemoteSha 28c98905f9a8685195cb89c86e4a2a5d9e596ac6

Contents

computeAlgebraicFunctions 2
computeBettiCurve 3
computeComplexPolynomial 5
computeEulerCharacteristic 7
computeLimits 9
computeNormalizedLife 10
computePersistenceBlock 12
computePersistenceImage 14
computePersistenceLandscape 16
computePersistenceSilhouette 18
computePersistentEntropy 20
computeStats 22
computeTemplateFunction 23
computeTropicalCoordinates 25

Index **28**

`computeAlgebraicFunctions`

Compute Algebraic Functions from a Persistence Diagram

Description

For a given persistence diagram $D = \{(b_i, d_i)\}_{i=1}^{N}$ (corresponding to a specified homological dimension), `computeAlgebraicFunctions()` computes the following four algebraic functions based on the birth and death values:

- $f_1 = \sum_i b_i(d_i - b_i)$.
- $f_2 = \sum_i (d_{\max} - d_i)(d_i - b_i)$, where $d_{\max} = \max(d_i)$.
- $f_3 = \sum_i b_i^2(d_i - b_i)^4$.
- $f_4 = \sum_i (d_{\max} - d_i)^2(d_i - b_i)^4$.

Points in D with infinite death values are ignored.

Usage

```
computeAlgebraicFunctions(D, homDim)
```

Arguments

`D`	a persistence diagram: a matrix with three columns containing the homological dimension, birth and death values respectively.
`homDim`	the homological dimension (0 for H_0, 1 for H_1, etc.). Rows in `D` are filtered based on this value.

Details

The function extracts rows from `D` where the first column equals `homDim`, and computes the four algebraic functions based on the filtered data. If `D` does not contain any points corresponding to `homDim`, a vector of zeros is returned.

Value

A (named) numeric vector (f_1, f_2, f_3, f_4).

Author(s)

Umar Islambekov

References

1. Adcock, A., Carlsson, E. and Carlsson, G., 2013. The ring of algebraic functions on persistence bar codes. Homology, Homotopy Appl., 18:381–402, 2016.

2. Ali, D., Asaad, A., Jimenez, M.J., Nanda, V., Paluzo-Hidalgo, E. and Soriano-Trigueros, M., (2023). A survey of vectorization methods in topological data analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence.

Examples

```
N <- 100 # The number of points to sample

set.seed(123) # Set a random seed for reproducibility

# Sample N points uniformly from the unit circle and add Gaussian noise
theta <- runif(N, min = 0, max = 2 * pi)
X <- cbind(cos(theta), sin(theta)) + rnorm(2 * N, mean = 0, sd = 0.2)

# Compute the persistence diagram using the Rips filtration built on top of X
# The 'threshold' parameter specifies the maximum distance for building simplices
D <- TDAstats::calculate_homology(X, threshold = 2)

# Compute algebraic functions for homological dimension H_0
computeAlgebraicFunctions(D, homDim = 0)

# Compute algebraic functions for homological dimension H_1
computeAlgebraicFunctions(D, homDim = 1)
```

`computeBettiCurve` *A Vector Summary of the Betti Curve*

Description

For a given persistence diagram $D = (b_i, d_i)_{i=1}^N$ (corresponding to a specified homological dimension), `computeBettiCurve()` vectorizes the Betti curve

$$\beta(t) = \sum_{i=1}^{N} \mathbf{1}_{[b_i, d_i)}(t)$$

based on a scale sequence `scaleSeq`. The evaluation method depends on the argument `evaluate`.

Usage

```
computeBettiCurve(D, homDim, scaleSeq, evaluate = "intervals")
```

Arguments

`D`	a persistence diagram: a matrix with three columns containing the homological dimension, birth and death values respectively.
`homDim`	the homological dimension (0 for H_0, 1 for H_1, etc.). Rows in `D` are filtered based on this value.
`scaleSeq`	a numeric vector of increasing scale values used for vectorization.
`evaluate`	a character string indicating the evaluation method. Must be either `"intervals"` (default) or `"points"`.

Details

The function extracts rows from `D` where the first column equals `homDim`, and computes values based on the filtered data and `scaleSeq`. If `D` does not contain any points corresponding to `homDim`, a vector of zeros is returned.

Value

A numeric vector containing elements computed using `scaleSeq`=$\{t_1, t_2, \ldots, t_n\}$ according to the method specified by `evaluate`.

- `"intervals"`: Computes average values of the Betti curve over intervals defined by consecutive elements in `scaleSeq`:

$$\Big(\frac{1}{\Delta t_1}\int_{t_1}^{t_2}\beta(t)dt, \frac{1}{\Delta t_2}\int_{t_2}^{t_3}\beta(t)dt, \ldots, \frac{1}{\Delta t_{n-1}}\int_{t_{n-1}}^{t_n}\beta(t)dt\Big) \in \mathbb{R}^{n-1},$$

  where $\Delta t_k = t_{k+1} - t_k$.

- `"points"`: Computes values of the Betti curve at each point in `scaleSeq`:

$$(\beta(t_1), \beta(t_2), \ldots, \beta(t_n)) \in \mathbb{R}^n.$$

Author(s)

Umar Islambekov, Hasani Pathirana

References

1. Chazal, F., & Michel, B. (2021). An Introduction to Topological Data Analysis: Fundamental and Practical Aspects for Data Scientists. Frontiers in Artificial Intelligence, 108.

2. Chung, Y. M., & Lawson, A. (2022). Persistence curves: A canonical framework for summarizing persistence diagrams. Advances in Computational Mathematics, 48(1), 1-42.

3. Islambekov, U., & Pathirana. H. (2024). Vector Summaries of Persistence Diagrams for Permutation-based Hypothesis Testing. Foundations of Data Science 6 (1), 41-61.

Examples

```
N <- 100 # The number of points to sample

set.seed(123) # Set a random seed for reproducibility

# Sample N points uniformly from the unit circle and add Gaussian noise
theta <- runif(N, min = 0, max = 2 * pi)
X <- cbind(cos(theta), sin(theta)) + rnorm(2 * N, mean = 0, sd = 0.2)

# Compute the persistence diagram using the Rips filtration built on top of X
# The 'threshold' parameter specifies the maximum distance for building simplices
D <- TDAstats::calculate_homology(X, threshold = 2)

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

# Compute a vector summary of the Betti curve for homological dimension H_0
computeBettiCurve(D, homDim = 0, scaleSeq)

# Compute a vector summary of the Betti curve for homological dimension H_1
computeBettiCurve(D, homDim = 1, scaleSeq)
```

`computeComplexPolynomial` *Compute Complex Polynomial Coefficients from a Persistence Diagram*

Description

For a given persistence diagram $D = \{(b_i, d_i)\}_{i=1}^{N}$ (corresponding to a specified homological dimension), `computeComplexPolynomial()` computes the coefficients of a complex polynomial

$$C_X(z) = \prod_{i=1}^{N} [z - X(b_i, d_i)],$$

where $X : \mathbb{R}^2 \to \mathbb{C}$ is any one of the following three functions:

- $R(x, y) = x + iy$.

- $S(x,y) = \begin{cases} \frac{y-x}{\alpha\sqrt{2}}(x+iy) & \text{if } (x,y) \neq (0,0) \\ 0 & \text{otherwise.} \end{cases}$
- $T(x,y) = \frac{y-x}{2}[(\cos\alpha - \sin\alpha) + i(\cos\alpha + \sin\alpha)]$,

where $\alpha = \sqrt{x^2+y^2}$. Points in D with infinite death values are ignored.

Usage

```
computeComplexPolynomial(D, homDim, m = 1, polyType = "R")
```

Arguments

`D`	a persistence diagram: a matrix with three columns containing the homological dimension, birth and death values respectively.
`homDim`	the homological dimension (0 for H_0, 1 for H_1, etc.). Rows in `D` are filtered based on this value.
`m`	the number of coefficients to return (default is 1). Must be less than or equal to the number of points in the diagram.
`polyType`	a string specifying the polynomial type to use. Options are "R" (default), "S", or "T".

Details

The function extracts rows from `D` where the first column equals `homDim`, and computes values based on the filtered data and `polyType`. If `D` does not contain any points corresponding to `homDim`, a matrix of zeros is returned.

Value

A numeric matrix of dimension $m \times 2$. Each row corresponds to a coefficient of the polynomial $C_X(z)$:

- Column 1: Real part of the coefficient.
- Column 2: Imaginary part of the coefficient.

Author(s)

Umar Islambekov

References

1. Ferri, M. and Landi, C., (1999). Representing size functions by complex polynomials. Proc. Math. Met. in Pattern Recognition, 9, pp.16-19.

2. Di Fabio, B. and Ferri, M., (2015). Comparing persistence diagrams through complex vectors. In Image Analysis and Processing—ICIAP 2015: 18th International Conference, Genoa, Italy, September 7-11, 2015, Proceedings, Part I 18 (pp. 294-305). Springer International Publishing.

3. Ali, D., Asaad, A., Jimenez, M.J., Nanda, V., Paluzo-Hidalgo, E. and Soriano-Trigueros, M., (2023). A survey of vectorization methods in topological data analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence.

Examples

```
N <- 100 # The number of points to sample

set.seed(123) # Set a random seed for reproducibility

# Sample N points uniformly from the unit circle and add Gaussian noise
theta <- runif(N, min = 0, max = 2 * pi)
X <- cbind(cos(theta), sin(theta)) + rnorm(2 * N, mean = 0, sd = 0.2)

# Compute the persistence diagram using the Rips filtration built on top of X
# The 'threshold' parameter specifies the maximum distance for building simplices
D <- TDAstats::calculate_homology(X, threshold = 2)

# Compute first 5 coefficients of the complex polynomial of type "R" for dimension H_0
computeComplexPolynomial(D, homDim = 0, m = 5, polyType = "R")

# Compute the first 5 coefficients of the complex polynomial of type "S"
#  for homological dimension H_0
computeComplexPolynomial(D, homDim = 0, m = 5, polyType = "S")
```

`computeEulerCharacteristic`

A Vector Summary of the Euler Characteristic Curve

Description

Vectorizes the Euler characteristic curve

$$\chi(t) = \sum_{k=0}^{d} (-1)^k \beta_k(t),$$

where $\beta_0, \beta_1, \ldots, \beta_d$ are the Betti curves corresponding to persistence diagrams $D_0, D_1, \ldots, D_d$ of dimensions $0, 1, \ldots, d$ respectively, all computed from the same filtration. The evaluation method depends on the argument `evaluate`.

Usage

```
computeEulerCharacteristic(D, scaleSeq, maxhomDim = -1, evaluate = "intervals")
```

Arguments

`D`	a persistence diagram: a matrix with three columns containing the homological dimension, birth and death values respectively.
`scaleSeq`	a numeric vector of increasing scale values used for vectorization.
`maxhomDim`	the maximum homological dimension considered (0 for H_0, 1 for H_1, etc.). If set to -1 (default), it uses the maximum dimension found in `D`.
`evaluate`	a character string indicating the evaluation method. Must be either `"intervals"` (default) or `"points"`.

Value

A numeric vector containing elements computed using `scaleSeq`=$\{t_1, t_2, \ldots, t_n\}$ according to the method specified by `evaluate`. If `D` does not contain any points corresponding to `homDim`, a vector of zeros is returned.

- `"intervals"`: Computes average values of the Euler characteristic curve over intervals defined by consecutive elements in `scaleSeq`:

$$\Big(\frac{1}{\Delta t_1}\int_{t_1}^{t_2}\chi(t)dt, \frac{1}{\Delta t_2}\int_{t_2}^{t_3}\chi(t)dt, \ldots, \frac{1}{\Delta t_{n-1}}\int_{t_{n-1}}^{t_n}\chi(t)dt\Big) \in \mathbb{R}^{n-1},$$

  where $\Delta t_k = t_{k+1} - t_k$.

- `"points"`: Computes values of the Euler characteristic curve at each point in `scaleSeq`:

$$(\chi(t_1), \chi(t_2), \ldots, \chi(t_n)) \in \mathbb{R}^n.$$

Author(s)

Umar Islambekov

References

1. Richardson, E., & Werman, M. (2014). Efficient classification using the Euler characteristic. Pattern Recognition Letters, 49, 99-106.

Examples

```
N <- 100 # The number of points to sample

set.seed(123) # Set a random seed for reproducibility

# Sample N points uniformly from the unit circle and add Gaussian noise
theta <- runif(N, min = 0, max = 2 * pi)
X <- cbind(cos(theta), sin(theta)) + rnorm(2 * N, mean = 0, sd = 0.2)

# Compute the persistence diagram using the Rips filtration built on top of X
# The 'threshold' parameter specifies the maximum distance for building simplices
D <- TDAstats::calculate_homology(X, threshold = 2)

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

# Compute a vector summary of the Euler characteristic curve
computeEulerCharacteristic(D, scaleSeq)
```

`computeLimits` *Compute the Extreme Values of Birth, Death, and Persistence Across Multiple Persistence Diagrams*

Description

Given a list of persistence diagrams, `computeLimits()` computes the extreme values of birth, death, and persistence across all diagrams for a specified homological dimension. Points with infinite death values are ignored.

Usage

```
computeLimits(Dlist, homDim)
```

Arguments

`Dlist`	a list of persistence diagrams, where each diagram is a matrix containing three columns representing homological dimension, birth, and death values.
`homDim`	the homological dimension (0 for H_0, 1 for H_1, etc.). Rows of the diagrams are filtered based on this value.

Value

A (named) numeric vector containing:

- `minB`: the minimum birth value across all diagrams.
- `maxB`: the maximum birth value across all diagrams.
- `maxD`: the maximum death value across all diagrams.
- `minP`: the minimum persistence value across all diagrams.
- `maxP`: the maximum persistence value across all diagrams.

Author(s)

Umar Islambekov

Examples

```
set.seed(123)
N <- 100 # The size of point clouds
nD <- 50 # The number of persistence diagrams
Dlist <- list()
for (i in 1:nD){
  # sample N points uniformly from the unit circle and add Gaussian noise
  theta <- runif(N, min = 0, max = 2 * pi)
  X <- cbind(cos(theta), sin(theta)) + rnorm(2 * N, mean = 0, sd = 0.2)

  # Compute the persistence diagram using the Rips filtration built on top of X
```

```
  # The 'threshold' parameter specifies the maximum distance for building simplices
  Dlist[[i]] <- TDAstats::calculate_homology(X, threshold = 2)
}

# Compute the extreme values of birth, death, and persistence across
# all 50 diagrams for homological dimension H_0
computeLimits(Dlist, homDim = 0)

# Compute the extreme values of birth, death, and persistence across
# all 50 diagrams for homological dimension H_1
computeLimits(Dlist, homDim = 1)
```

`computeNormalizedLife` *A Vector Summary of the Normalized Life Curve*

Description

For a given persistence diagram $D = \{(b_i, d_i)\}_{i=1}^{N}$ (corresponding to a specified homological dimension), `computeNormalizedLife()` vectorizes the normalized life curve

$$sl(t) = \sum_{i=1}^{N} \frac{d_i - b_i}{L} \mathbf{1}_{[b_i, d_i)}(t),$$

where $L = \sum_{i=1}^{N}(d_i - b_i)$, based on a scale sequence `scaleSeq`. The evaluation method depends on the argument `evaluate`. Points in D with infinite death values are ignored.

Usage

```
computeNormalizedLife(D, homDim, scaleSeq, evaluate = "intervals")
```

Arguments

`D`	a persistence diagram: a matrix with three columns containing the homological dimension, birth and death values respectively.
`homDim`	the homological dimension (0 for H_0, 1 for H_1, etc.). Rows in `D` are filtered based on this value.
`scaleSeq`	a numeric vector of increasing scale values used for vectorization.
`evaluate`	a character string indicating the evaluation method. Must be either `"intervals"` (default) or `"points"`.

Details

The function extracts rows from `D` where the first column equals `homDim`, and computes values based on the filtered data and `scaleSeq`. If `D` does not contain any points corresponding to `homDim`, a vector of zeros is returned.

Value

A numeric vector containing elements computed using `scaleSeq`=$\{t_1, t_2, \ldots, t_n\}$ according to the method specified by `evaluate`.

- `"intervals"`: Computes average values of the normalized life curve over intervals defined by consecutive elements in `scaleSeq`:

$$\Big(\frac{1}{\Delta t_1}\int_{t_1}^{t_2} sl(t)dt, \frac{1}{\Delta t_2}\int_{t_2}^{t_3} sl(t)dt, \ldots, \frac{1}{\Delta t_{n-1}}\int_{t_{n-1}}^{t_n} sl(t)dt\Big) \in \mathbb{R}^{n-1},$$

  where $\Delta t_k = t_{k+1} - t_k$.

- `"points"`: Computes values of the normalized life curve at each point in `scaleSeq`:

$$(sl(t_1), sl(t_2), \ldots, sl(t_n)) \in \mathbb{R}^n.$$

Author(s)

Umar Islambekov

References

Chung, Y. M., & Lawson, A. (2022). Persistence curves: A canonical framework for summarizing persistence diagrams. Advances in Computational Mathematics, 48(1), 1-42.

Examples

```
N <- 100 # The number of points to sample

set.seed(123) # Set a random seed for reproducibility

# Sample N points uniformly from the unit circle and add Gaussian noise
theta <- runif(N, min = 0, max = 2 * pi)
X <- cbind(cos(theta), sin(theta)) + rnorm(2 * N, mean = 0, sd = 0.2)

# Compute the persistence diagram using the Rips filtration built on top of X
# The 'threshold' parameter specifies the maximum distance for building simplices
D <- TDAstats::calculate_homology(X, threshold = 2)

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

# Compute a vector summary of the normalized life curve for homological dimension H_0
computeNormalizedLife(D, homDim = 0, scaleSeq)

# Compute a vector summary of the normalized life curve for homological dimension H_1
computeNormalizedLife(D, homDim = 1, scaleSeq)
```

`computePersistenceBlock` *A Vector Summary of the Persistence Block*

Description

For a given persistence diagram $D = \{(b_i, p_i)\}_{i=1}^N$ (corresponding to a specified homological dimension), `computePersistenceBlock()` vectorizes the persistence block

$$f(x,y) = \sum_{i=1}^{N} \mathbf{1}_{E(b_i,p_i)}(x,y),$$

where $E(b_i,p_i) = [b_i - \frac{\lambda_i}{2}, b_i + \frac{\lambda_i}{2}] \times [p_i - \frac{\lambda_i}{2}, p_i + \frac{\lambda_i}{2}]$ and $\lambda_i = 2\tau p_i$ with $\tau \in (0,1]$. Points in D with infinite persistence values are ignored.

Usage

```
computePersistenceBlock(D, homDim, xSeq, ySeq, tau=0.3)
```

Arguments

`D`	a persistence diagram: a matrix with three columns containing the homological dimension, *birth* and *persistence* values respectively.
`homDim`	the homological dimension (0 for H_0, 1 for H_1, etc.). Rows in `D` are filtered based on this value.
`xSeq`	a numeric vector of increasing x (birth) values used for vectorization.
`ySeq`	a numeric vector of increasing y (persistence) values used for vectorization.
`tau`	a parameter (between 0 and 1) controlling block sizes. Default is `tau=0.3`.

Details

The function extracts rows from `D` where the first column equals `homDim`, and computes values based on the filtered data, `xSeq` and `ySeq`. If `D` does not contain any points corresponding to `homDim`, a vector of zeros is returned.

Value

A numeric vector whose elements are the weighted averages of the persistence block computed over each cell of the two-dimensional grid constructred from `xSeq`=$\{x_1, x_2, \ldots, x_n\}$ and `ySeq`=$\{y_1, y_2, \ldots, y_m\}$:

$$\Big(\frac{1}{\Delta x_1 \Delta y_1}\int_{[x_1,x_2]\times[y_1,y_2]} f(x,y)wdA, \ldots, \frac{1}{\Delta x_{n-1}\Delta y_{m-1}}\int_{[x_{n-1},x_n]\times[y_{m-1},y_m]} f(x,y)wdA\Big) \in \mathbb{R}^d,$$

where $d = (n-1)(m-1)$, $wdA = (x+y)dxdy$, $\Delta x_k = x_{k+1} - x_k$ and $\Delta y_j = y_{j+1} - y_j$.

If `homDim=0` and all the birth values are equal (e.g., zero), univariate persistence block functions are used instead for vectorization:

$$\Big(\frac{1}{\Delta y_1}\int_{y_1}^{y_2} f(y)ydy, \ldots, \frac{1}{\Delta y_{m-1}}\int_{y_{m-1}}^{y_m} f(y)ydy\Big) \in \mathbb{R}^{m-1},$$

where $f(y) = \sum_{i=1}^{N} \mathbf{1}_{[p_i - \frac{\lambda_i}{2}, p_i + \frac{\lambda_i}{2}]}(y)$ and $\Delta y_j = y_{j+1} - y_j$.

Author(s)

Umar Islambekov, Aleksei Luchinsky

References

1. Chan, K. C., Islambekov, U., Luchinsky, A., & Sanders, R. (2022). A computationally efficient framework for vector representation of persistence diagrams. Journal of Machine Learning Research 23, 1-33.

Examples

```
N <- 100 # The number of points to sample
set.seed(123) # Set a random seed for reproducibility

# Sample N points uniformly from the unit circle and add Gaussian noise
theta <- runif(N, min = 0, max = 2 * pi)
X <- cbind(cos(theta), sin(theta)) + rnorm(2 * N, mean = 0, sd = 0.2)

# Compute the persistence diagram using the Rips filtration built on top of X
# The 'threshold' parameter specifies the maximum distance for building simplices
D <- TDAstats::calculate_homology(X, threshold = 2)

# Switch from the birth-death to the birth-persistence coordinates
D[,3] <- D[,3] - D[,2]
colnames(D)[3] <- "Persistence"

# Construct one-dimensional grid of scale values
ySeqH0 <- unique(quantile(D[D[,1] == 0, 3], probs = seq(0, 1, by = 0.2)))
tau <- 0.3 # Parameter in [0,1] which controls the size of blocks around each point of the diagram

# Compute a vector summary of the persistence block for homological dimension H_0
computePersistenceBlock(D, homDim = 0, xSeq = NA, ySeq = ySeqH0, tau = tau)

xSeqH1 <- unique(quantile(D[D[,1] == 1, 2], probs = seq(0, 1, by = 0.2)))
ySeqH1 <- unique(quantile(D[D[,1] == 1, 3], probs = seq(0, 1, by = 0.2)))

# Compute a vector summary of the persistence block for homological dimension H_1
computePersistenceBlock(D, homDim = 1, xSeq = xSeqH1, ySeq = ySeqH1, tau = tau)
```

`computePersistenceImage` *A Vector Summary of the Persistence Surface*

Description

For a given persistence diagram $D = \{(b_i, p_i)\}_{i=1}^{N}$ (corresponding to a specified homological dimension), `computePersistenceImage()` computes the persistence image - a vector summary of the persistence surface:

$$\rho(x, y) = \sum_{i=1}^{N} f(b_i, p_i)\phi_{(b_i,p_i)}(x, y),$$

where $\phi_{(b_i,p_i)}(x, y)$ is the Gaussian distribution with mean (b_i, p_i) and covariance matrix $\sigma^2 I_{2\times 2}$ and

$$f(b, p) = w(p) = \begin{cases} 0 & p \leq 0 \\ p/p_{max} & 0 < p < p_{max} \\ 1 & p \geq p_{max} \end{cases}$$

is the weighting function with p_{max} being the maximum persistence value among all persistence diagrams considered in the experiment. Points in D with infinite persistence values are ignored.

Usage

```
computePersistenceImage(D, homDim, xSeq, ySeq, sigma)
```

Arguments

`D`	a persistence diagram: a matrix with three columns containing the homological dimension, *birth* and *persistence* values respectively.
`homDim`	the homological dimension (0 for H_0, 1 for H_1, etc.). Rows in `D` are filtered based on this value.
`xSeq`	a numeric vector of increasing x (birth) values used for vectorization.
`ySeq`	a numeric vector of increasing y (persistence) values used for vectorization.
`sigma`	a standard deviation (σ) of the Gaussian.

Details

The function extracts rows from `D` where the first column equals `homDim`, and computes values based on the filtered data, `xSeq` and `ySeq`. If `D` does not contain any points corresponding to `homDim`, a vector of zeros is returned.

Value

A numeric vector whose elements are the average values of the persistence surface computed over each cell of the two-dimensional grid constructred from `xSeq`=$\{x_1, x_2, \ldots, x_n\}$ and `ySeq`=$\{y_1, y_2, \ldots, y_m\}$:

$$\Big(\frac{1}{\Delta x_1 \Delta y_1}\int_{[x_1,x_2]\times[y_1,y_2]} \rho(x,y)dA, \ldots, \frac{1}{\Delta x_{n-1}\Delta y_{m-1}}\int_{[x_{n-1},x_n]\times[y_{m-1},y_m]} \rho(x,y)dA\Big) \in \mathbb{R}^d,$$

where $d = (n-1)(m-1)$, $dA = dxdy$, $\Delta x_k = x_{k+1} - x_k$ and $\Delta y_j = y_{j+1} - y_j$.

If `homDim`=`0` and all the birth values are equal (e.g., zero), univariate Gaussians are used instead for vectorization:

$$\Big(\frac{1}{\Delta y_1}\int_{y_1}^{y_2} \rho(y)dy, \ldots, \frac{1}{\Delta y_{m-1}}\int_{y_{m-1}}^{y_m} \rho(y)dy\Big) \in \mathbb{R}^{m-1},$$

where $\rho(y) = \sum_{i=1}^{N} f(p_i)\phi_{p_i}(y)$ and $\Delta y_j = y_{j+1} - y_j$.

Author(s)

Umar Islambekov

References

1. Adams, H., Emerson, T., Kirby, M., Neville, R., Peterson, C., Shipman, P., ... & Ziegelmeier, L. (2017). Persistence images: A stable vector representation of persistent homology. Journal of Machine Learning Research, 18.

Examples

```
N <- 100 # The number of points to sample
set.seed(123) # Set a random seed for reproducibility

# Sample N points uniformly from the unit circle and add Gaussian noise
theta <- runif(N, min = 0, max = 2 * pi)
X <- cbind(cos(theta), sin(theta)) + rnorm(2 * N, mean = 0, sd = 0.2)

# Compute the persistence diagram using the Rips filtration built on top of X
# The 'threshold' parameter specifies the maximum distance for building simplices
D <- TDAstats::calculate_homology(X, threshold = 2)

# Switch from the birth-death to the birth-persistence coordinates
D[,3] <- D[,3] - D[,2]
colnames(D)[3] <- "Persistence"

resB <- 5 # Resolution (or grid size) along the birth axis
resP <- 5 # Resolution (or grid size) along the persistence axis

# Compute persistence image for homological dimension H_0
minPH0 <- min(D[D[,1]==0,3]); maxPH0 <- max(D[D[,1]==0,3])
ySeqH0 <- seq(minPH0, maxPH0, length.out = resP + 1)
sigma <- 0.5 * (maxPH0 - minPH0) / resP
```

```
computePersistenceImage(D, homDim = 0, xSeq = NA, ySeq = ySeqH0, sigma = sigma)

# Compute persistence image for homological dimension H_1
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])
xSeqH1 <- seq(minBH1, maxBH1, length.out = resB + 1)
ySeqH1 <- seq(minPH1, maxPH1, length.out = resP + 1)
sigma <- 0.5 * (maxPH1 - minPH1) / resP
computePersistenceImage(D, homDim = 1, xSeq = xSeqH1, ySeq = ySeqH1, sigma = sigma)
```

`computePersistenceLandscape` *Vector Summaries of the Persistence Landscape Functions*

Description

For a given persistence diagram $D = \{(b_i, d_i)\}_{i=1}^N$ (corresponding to a specified homological dimension), `computePersistenceLandscape()` vectorizes the first k persistence landscape functions

$$\lambda_j(t) = j\text{max}_{1 \le i \le N} \Lambda_i(t), \quad j = 1, \ldots, k,$$

where $j$max returns the jth largest value and

$$\Lambda_i(t) = \begin{cases} t - b_i & t \in [b_i, \frac{b_i + d_i}{2}] \\ d_i - t & t \in (\frac{b_i + d_i}{2}, d_i] \\ 0 & \text{otherwise} \end{cases}$$

based on a scale sequence `scaleSeq`. For *generalized* persistence landscape functions, we instead take

$$\Lambda_i(t) = \begin{cases} \frac{d_i - b_i}{2K_h(0)} K_h(t - \frac{b_i + d_i}{2}) & \text{for } |\frac{t - \frac{b_i + d_i}{2}}{h}| \le 1 \\ 0 & \text{otherwise} \end{cases}$$

where $K_h(\cdot)$ is a kernel function with the bandwidth parameter h.

Usage

```
computePersistenceLandscape(D, homDim, scaleSeq, k = 1, generalized = FALSE,
  kernel = "triangle", h = NULL)
```

Arguments

`D`	a persistence diagram: a matrix with three columns containing the homological dimension, birth and death values respectively.
`homDim`	the homological dimension (0 for H_0, 1 for H_1, etc.). Rows in `D` are filtered based on this value.
`scaleSeq`	a numeric vector of increasing scale values used for vectorization.

k	an integer specifying the number of persistence landscape functions to consider (default is 1).
generalized	a logical value indicating whether to use a generalized persistence landscape or not. Default is FALSE.
kernel	a string specifying the kernel type to use if generalized = TRUE. Options are "triangle", "epanechnikov", or "tricubic". Default is "triangle".
h	a positive numeric value specifying the bandwidth for the kernel. Must be provided if generalized = TRUE.

Details

The function extracts rows from D where the first column equals homDim, and computes values based on the filtered data and scaleSeq. If D does not contain any points corresponding to homDim, a vector of zeros is returned. If generalized = TRUE, three different kernel functions are currently supported:

- Triangle kernel: $K_h(t) = \frac{1}{h}\max(0, 1 - \frac{|t|}{h})$
- Epanechnikov kernel: $K_h(t) = \frac{3}{4h}\max(0, 1 - \frac{t^2}{h^2})$
- Tricubic kernel: $K_h(t) = \frac{70}{81h}\max(0, (1 - \frac{|t|^3}{h^3})^3)$

Value

A matrix where the jth column contains the values of the jth order persistence landscape function evaluated at each point of scaleSeq=$\{t_1, t_2, \ldots, t_n\}$:

$$\begin{bmatrix} \lambda_1(t_1) & \lambda_2(t_1) & \cdots & \lambda_k(t_1) \\ \lambda_1(t_2) & \lambda_2(t_2) & \cdots & \lambda_k(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_1(t_n) & \lambda_2(t_n) & \cdots & \lambda_k(t_n) \end{bmatrix}$$

Author(s)

Umar Islambekov

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).

3. Berry, E., Chen, Y. C., Cisewski-Kehe, J., & Fasy, B. T. (2020). Functional summaries of persistence diagrams. Journal of Applied and Computational Topology, 4(2):211–262.

Examples

```
N <- 100 # The number of points to sample

set.seed(123) # Set a random seed for reproducibility

# Sample N points uniformly from the unit circle and add Gaussian noise
theta <- runif(N, min = 0, max = 2 * pi)
X <- cbind(cos(theta), sin(theta)) + rnorm(2 * N, mean = 0, sd = 0.2)

# Compute the persistence diagram using the Rips filtration built on top of X
# The 'threshold' parameter specifies the maximum distance for building simplices
D <- TDAstats::calculate_homology(X, threshold = 2)

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

# Compute vector summaries of the first three persistence landscape functions
# for homological dimension H_1

computePersistenceLandscape(D, homDim = 1, scaleSeq, k = 3)

# Compute vector summaries of the first three generalized persistence
# landscape functions (with triangle kernel and bandwidth h=0.2)
# for homological dimension H_1

computePersistenceLandscape(D, homDim = 1, scaleSeq, generalized = TRUE, k = 3, h = 0.2)
```

`computePersistenceSilhouette` *A Vector Summary of the Persistence Silhouette Function*

Description

For a given persistence diagram $D = \{(b_i, d_i)\}_{i=1}^N$ (corresponding to a specified homological dimension), `computePersistenceSilhouette()` vectorizes the pth power persistence silhouette function

$$\phi_p(t) = \frac{\sum_{i=1}^N |d_i - b_i|^p \Lambda_i(t)}{\sum_{i=1}^N |d_i - b_i|^p},$$

where

$$\Lambda_i(t) = \begin{cases} t - b_i & t \in [b_i, \frac{b_i + d_i}{2}] \\ d_i - t & t \in (\frac{b_i + d_i}{2}, d_i] \\ 0 & \text{otherwise} \end{cases}$$

based on a scale sequence `scaleSeq`. The evaluation method depends on the argument `evaluate`. Points in D with infinite death values are ignored.

Usage

```
computePersistenceSilhouette(D, homDim, scaleSeq, p = 1.0, evaluate = "intervals")
```

Arguments

D	a persistence diagram: a matrix with three columns containing the homological dimension, birth and death values respectively.
homDim	the homological dimension (0 for H_0, 1 for H_1, etc.). Rows in D are filtered based on this value.
scaleSeq	a numeric vector of increasing scale values used for vectorization.
p	power of the weights for the silhouette function. By default, p=1.
evaluate	a character string indicating the evaluation method. Must be either "intervals" (default) or "points".

Details

The function extracts rows from D where the first column equals homDim, and computes values based on the filtered data and scaleSeq. If D does not contain any points corresponding to homDim, a vector of zeros is returned.

Value

A numeric vector containing elements computed using scaleSeq=$\{t_1, t_2, \ldots, t_n\}$ according to the method specified by evaluate.

- "intervals": Computes average values of the persistence silhouette function over intervals defined by consecutive elements in scaleSeq:

$$\Big(\frac{1}{\Delta t_1}\int_{t_1}^{t_2}\phi_p(t)dt, \frac{1}{\Delta t_2}\int_{t_2}^{t_3}\phi_p(t)dt, \ldots, \frac{1}{\Delta t_{n-1}}\int_{t_{n-1}}^{t_n}\phi_p(t)dt\Big) \in \mathbb{R}^{n-1},$$

  where $\Delta t_k = t_{k+1} - t_k$.

- "points": Computes values of the persistence silhouette function at each point in scaleSeq:

$$(\phi_p(t_1), \phi_p(t_2), \ldots, \phi_p(t_n)) \in \mathbb{R}^n.$$

Author(s)

Umar Islambekov

References

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

Examples

```
N <- 100 # The number of points to sample

set.seed(123) # Set a random seed for reproducibility

# Sample N points uniformly from the unit circle and add Gaussian noise
theta <- runif(N, min = 0, max = 2 * pi)
X <- cbind(cos(theta), sin(theta)) + rnorm(2 * N, mean = 0, sd = 0.2)

# Compute the persistence diagram using the Rips filtration built on top of X
# The 'threshold' parameter specifies the maximum distance for building simplices
D <- TDAstats::calculate_homology(X, threshold = 2)

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

# Compute a vector summary of the persistence silhouette (with p=1) for homological dimension H_0
computePersistenceSilhouette(D, homDim = 0, scaleSeq)

# Compute a vector summary of the persistence silhouette (with p=1) for homological dimension H_1
computePersistenceSilhouette(D, homDim = 1, scaleSeq)
```

---

computePersistentEntropy

A Vector Summary of the Persistent Entropy Summary Function

Description

For a given persistence diagram $D = (b_i, d_i)_{i=1}^N$ (corresponding to a specified homological dimension), `computePersistentEntropy()` vectorizes the persistent entropy summary function

$$S(t) = -\sum_{i=1}^{N} \frac{l_i}{L} \log_2 (\frac{l_i}{L}) \mathbf{1}_{[b_i, d_i]}(t),$$

where $l_i = d_i - b_i$ and $L = \sum_{i=1}^N l_i$, based on a scale sequence `scaleSeq`. If $N = 1$, L is set to 1. The evaluation method depends on the argument `evaluate`. Points in D with infinite death values are ignored.

Usage

```
computePersistentEntropy(D, homDim, scaleSeq, evaluate = "intervals")
```

Arguments

`D`	a persistence diagram: a matrix with three columns containing the homological dimension, birth and death values respectively.
`homDim`	the homological dimension (0 for H_0, 1 for H_1, etc.). Rows in `D` are filtered based on this value.

scaleSeq	a numeric vector of increasing scale values used for vectorization.
evaluate	a character string indicating the evaluation method. Must be either "intervals" (default) or "points".

Details

The function extracts rows from D where the first column equals homDim, and computes values based on the filtered data and scaleSeq. If D does not contain any points corresponding to homDim, a vector of zeros is returned.

Value

A numeric vector containing elements computed using scaleSeq=$\{t_1, t_2, \ldots, t_n\}$ according to the method specified by evaluate.

- "intervals": Computes average values of the persistent entropy summary function over intervals defined by consecutive elements in scaleSeq:

$$\Big(\frac{1}{\Delta t_1}\int_{t_1}^{t_2} S(t)dt, \frac{1}{\Delta t_2}\int_{t_2}^{t_3} S(t)dt, \ldots, \frac{1}{\Delta t_{n-1}}\int_{t_{n-1}}^{t_n} S(t)dt\Big) \in \mathbb{R}^{n-1},$$

  where $\Delta t_k = t_{k+1} - t_k$.
- "points": Computes values of the persistent entropy summary function at each point in scaleSeq:

$$(S(t_1), S(t_2), \ldots, S(t_n)) \in \mathbb{R}^n.$$

Author(s)

Umar Islambekov

References

1. Atienza, N., Gonzalez-Díaz, R., & Soriano-Trigueros, M. (2020). On the stability of persistent entropy and new summary functions for topological data analysis. Pattern Recognition, 107, 107509.

Examples

```
N <- 100 # The number of points to sample

set.seed(123) # Set a random seed for reproducibility

# Sample N points uniformly from the unit circle and add Gaussian noise
theta <- runif(N, min = 0, max = 2 * pi)
X <- cbind(cos(theta), sin(theta)) + rnorm(2 * N, mean = 0, sd = 0.2)

# Compute the persistence diagram using the Rips filtration built on top of X
# The 'threshold' parameter specifies the maximum distance for building simplices
D <- TDAstats::calculate_homology(X, threshold = 2)
```

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

# Compute a vector summary of the persistent entropy summary function for homological dimension H_0
computePersistentEntropy(D, homDim = 0, scaleSeq)

# Compute a vector summary of the persistent entropy summary function for homological dimension H_1
computePersistentEntropy(D, homDim = 1, scaleSeq)
```

`computeStats` *Compute Descriptive Statistics for Births, Deaths, Midpoints, and Lifespans in a Persistence Diagram*

Description

For a given persistence diagram $D = \{(b_i, d_i)\}_{i=1}^{N}$ (corresponding to a specified homological dimension), `computeStats()` calculates descriptive statistics of the birth, death, midpoint (the average of birth and death), and lifespan (death minus birth) values. Additionally, it computes the total number of points and entropy of the lifespan values. Points in D with infinite death values are ignored.

Usage

```
computeStats(D, homDim)
```

Arguments

`D`	a persistence diagram: a matrix with three columns containing the homological dimension, birth and death values respectively.
`homDim`	the homological dimension (0 for H_0, 1 for H_1, etc.). Rows in `D` are filtered based on this value.

Details

The function extracts rows from `D` where the first column equals `homDim`, and computes the mean, standard deviation, median, IQR (interquartile range), range, 10th, 25th, 75th and 90th percentiles of the birth, death, midpoint, lifespan (or persistence) values; the total number of bars (or points in the diagram) and the entropy of the lifespan values (-$\sum_{i=1}^{N} \frac{l_i}{L} \log_2(\frac{l_i}{L})$, where $l_i = d_i - b_i$ (lifespan) and $L = \sum_{i=1}^{N} l_i$). If `D` does not contain any points corresponding to `homDim`, a vector of zeros is returned.

Value

A (named) 38-dimensional numeric vector containing:

- `mean_births`, `stddev_births`, `median_births`, `iqr_births`, `range_births`, `p10_births`, `p25_births`, `p75_births`, `p90_births`: Descriptive statistics for birth values.

- mean_deaths, stddev_deaths, median_deaths, iqr_deaths, range_deaths, p10_deaths, p25_deaths, p75_deaths, p90_deaths: Descriptive statistics for death values.
- mean_midpoints, stddev_midpoints, median_midpoints, iqr_midpoints, range_midpoints, p10_midpoints, p25_midpoints, p75_midpoints, p90_midpoints: Descriptive statistics for midpoint values (mean of birth and death values).
- mean_lifespans, stddev_lifespans, median_lifespans, iqr_lifespans, range_lifespans, p10_lifespans, p25_lifespans, p75_lifespans, p90_lifespans: Descriptive statistics for lifespan (or persistence) values (difference between death and birth values).
- total_bars: The total number of points in the specified homological dimension.
- entropy: The entropy of the lifespan values.

Author(s)

Umar Islambekov

References

1. Ali, D., Asaad, A., Jimenez, M.J., Nanda, V., Paluzo-Hidalgo, E. and Soriano-Trigueros, M., (2023). A survey of vectorization methods in topological data analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence.

Examples

```
N <- 100 # The number of points to sample

set.seed(123) # Set a random seed for reproducibility

# Sample N points uniformly from the unit circle and add Gaussian noise
theta <- runif(N, min = 0, max = 2 * pi)
X <- cbind(cos(theta), sin(theta)) + rnorm(2 * N, mean = 0, sd = 0.2)

# Compute the persistence diagram using the Rips filtration built on top of X
# The 'threshold' parameter specifies the maximum distance for building simplices
D <- TDAstats::calculate_homology(X, threshold = 2)

# Compute statistics for homological dimension H_0
computeStats(D, homDim = 0)

# Compute statistics for homological dimension H_1
computeStats(D, homDim = 1)
```

---

computeTemplateFunction

Compute a Vectorization of a Persistence Diagram based on Tent Template Functions

---

Description

For a given persistence diagram $D = \{(b_i, d_i)\}_{i=1}^N$ (corresponding to a specified homological dimension), `computeTemplateFunction()` computes a vectorization using a collection of tent template functions defined by

$$G_{(b,p),\delta}(D) = \sum_{i=1}^{N} \max\left\{0, 1 - \frac{1}{\delta}\max\left(|b_i - b|, |p_i - p|\right)\right\},$$

where $p_i = d_i - b_i$ (persistence), $b \geq 0$, $p > 0$ and $0 < \delta < p$. The point (b, p) is referred to as the center. Points in D with infinite death values are ignored.

Usage

```
computeTemplateFunction(D, homDim, delta, d, epsilon)
```

Arguments

`D`	a persistence diagram: a matrix with three columns containing the homological dimension, birth and death values respectively.
`homDim`	the homological dimension (0 for H_0, 1 for H_1, etc.). Rows in `D` are filtered based on this value.
`delta`	a positive scalar representing the increment size used in the computation of the template function.
`d`	a positive integer specifying the number of bins along each axis in the grid.
`epsilon`	a positive scalar indicating the vertical shift applied to the grid.

Details

The function extracts rows from `D` where the first column equals `homDim`, and computes the tent template function on a discretized grid determined by `delta`, `d`, and `epsilon`. The number of tent functions is controlled by `d`. The value of δ is chosen such that the box $[0, \delta d] \times [\epsilon, \delta d + \epsilon]$ contains all the points of the diagrams considered in the birth-persistence plane. ϵ should be smaller than the minimum persistence value across all the diagrams under consideration. If `D` does not contain any points corresponding to `homDim`, a vector of zeros is returned.

If `homDim=0` and all the birth values are equal (e.g., zero), one-dimensional tent template functions are used instead for vectorization:

$$G_{p,\delta}(D) = \sum_{i=1}^{N} \max(0, 1 - \frac{|p_i - p|}{\delta}).$$

Value

A numeric vector of dimension $(d+1)d$, containing the values of the tent template functions centered at the grid points $\{(\delta i, \delta j + \epsilon)\}_{i=0,j=1}^{d,d}$:

$$\{G_{(\delta i, \delta j + \epsilon),\delta}(D) \mid 0 \leq i \leq d,\ 1 \leq j \leq d\}.$$

When one-dimensional tent template functions are used, the returned vector has a dimension of d:

$$\{G_{\delta j+\epsilon,\delta}(D) \mid 1 \leq j \leq d\}.$$

Author(s)

Umar Islambekov

References

1. Perea, J.A., Munch, E. and Khasawneh, F.A., (2023). Approximating continuous functions on persistence diagrams using template functions. Foundations of Computational Mathematics, 23(4), pp.1215-1272.

Examples

```
N <- 100 # The number of points to sample

set.seed(123) # Set a random seed for reproducibility

# Sample N points uniformly from the unit circle and add Gaussian noise
theta <- runif(N, min = 0, max = 2 * pi)
X <- cbind(cos(theta), sin(theta)) + rnorm(2 * N, mean = 0, sd = 0.2)

# Compute the persistence diagram using the Rips filtration built on top of X
# The 'threshold' parameter specifies the maximum distance for building simplices
D <- TDAstats::calculate_homology(X, threshold = 2)

# Compute a vectorizaton based on tent template functions for homological dimension H_0
computeTemplateFunction(D, homDim = 0, delta = 0.1, d = 20, epsilon = 0.01)

# Compute a vectorizaton based on tent template functions for homological dimension H_1
computeTemplateFunction(D, homDim = 1, delta = 0.1, d = 9, epsilon = 0.01)
```

---

computeTropicalCoordinates

Compute Tropical Coordinates from a Persistence Diagram

Description

For a given persistence diagram $D = \{(b_i, d_i)\}_{i=1}^{N}$ (corresponding to a specified homological dimension), `computeTropicalCoordinates()` computes the following seven tropical coordinates based on the lifespans (or persistence) $\lambda_i = d_i - b_i$:

- $F_1 = \max_i \lambda_i$.
- $F_2 = \max_{i<j}(\lambda_i + \lambda_j)$.
- $F_3 = \max_{i<j<k}(\lambda_i + \lambda_j + \lambda_k)$.
- $F_4 = \max_{i<j<k<l}(\lambda_i + \lambda_j + \lambda_k + \lambda_l)$.

- $F_5 = \sum_i \lambda_i$.
- $F_6 = \sum_i \min(r\lambda_i, b_i)$, where r is a positive integer.
- $F_7 = \sum_j \big(\max_i(\min(r\lambda_i, b_i) + \lambda_i) - (\min(r\lambda_j, b_j) + \lambda_j)\big)$.

Points in D with infinite death values are ignored.

Usage

```
computeTropicalCoordinates(D, homDim, r = 1)
```

Arguments

D	a persistence diagram: a matrix with three columns containing the homological dimension, birth and death values respectively.
homDim	the homological dimension (0 for H_0, 1 for H_1, etc.). Rows in D are filtered based on this value.
r	a positive integer used to compute F_6 and F_7. Default is 1.

Details

The function extracts rows from D where the first column equals homDim, and computes the seven tropical coordinates based on the filtered data. If D does not contain any points corresponding to homDim, a vector of zeros is returned.

Value

A (named) numeric vector $(F_1, F_2, F_3, F_4, F_5, F_6, F_7)$.

Author(s)

Umar Islambekov

References

1. Kališnik, S., (2019). Tropical coordinates on the space of persistence barcodes. Foundations of Computational Mathematics, 19(1), pp.101-129.

2. Ali, D., Asaad, A., Jimenez, M.J., Nanda, V., Paluzo-Hidalgo, E. and Soriano-Trigueros, M., (2023). A survey of vectorization methods in topological data analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence.

Examples

```
N <- 100 # The number of points to sample

set.seed(123) # Set a random seed for reproducibility

# Sample N points uniformly from the unit circle and add Gaussian noise
theta <- runif(N, min = 0, max = 2 * pi)
X <- cbind(cos(theta), sin(theta)) + rnorm(2 * N, mean = 0, sd = 0.2)
```

```
# Compute the persistence diagram using the Rips filtration built on top of X
# The 'threshold' parameter specifies the maximum distance for building simplices
D <- TDAstats::calculate_homology(X, threshold = 2)

# Compute tropical coordinates for homological dimension H_0
computeTropicalCoordinates(D, homDim = 0)

# Compute tropical coordinates for homological dimension H_1
computeTropicalCoordinates(D, homDim = 1)
```

Index

computeAlgebraicFunctions, 2
computeBettiCurve, 3
computeComplexPolynomial, 5
computeEulerCharacteristic, 7
computeLimits, 9
computeNormalizedLife, 10
computePersistenceBlock, 12
computePersistenceImage, 14
computePersistenceLandscape, 16
computePersistenceSilhouette, 18
computePersistentEntropy, 20
computeStats, 22
computeTemplateFunction, 23
computeTropicalCoordinates, 25