Package ‘ripserr’

October 20, 2020

Title Calculate Persistent Homology with Ripser-Based Engines
Version 0.1.1
License GPL-3
Encoding UTF-8
LazyData true
RoxygenNote 7.1.1
URL https://rrrlw.github.io/ripserr/
BugReports https://github.com/rrrlw/ripserr/issues
LinkingTo Rcpp
Depends R (>= 3.5.0)
Imports methods (>= 3.0), Rcpp (>= 1.0), stats (>= 3.0)
SystemRequirements C++11
Suggests testthat (>= 2.3), covr (>= 3.5), knitr (>= 1.29), rmarkdown (>= 2.3)
NeedsCompilation yes
Author Raoul Wadhwa [aut, cre] (<https://orcid.org/0000-0003-0503-9580>),
Matt Piekenbrock [aut],
Jacob Scott [aut] (<https://orcid.org/0000-0003-2971-7673>),
Takeki Sudo [cph, ctb] (Takeki Sudo is a copyright holder for Cubical Ripser (GPL-3 license), which was refactored prior to inclusion in ripserr.),
Kazushi Ahara [cph, ctb] (Kazushi Ahara is a copyright holder for Cubical Ripser (GPL-3 license), which was refactored prior to inclusion in ripserr.),
Ulrich Bauer [cph, ctb] (Ulrich Bauer holds the copyright to Ripser (MIT license), which was refactored prior to inclusion in ripserr.)
cubical

Description

Calculates the persistent homology of a 2- to 4-dimensional numeric array using a Cubical complex. This function is an R wrapper for Takeki Sudo and Kazushi Ahara’s Cubical Ripser C++ library. For more information on the C++ library, see https://github.com/CubicalRipser.

Usage

cubical(
  dataset,  # numeric array containing pixel/voxel data
  threshold = 9999,  # maximum diameter for computation of Cubical complex
  method = 0,  # defaults to 0 for link join; alternatively, can be 1 for compute pairs. See original Cubical Ripser code at GitHub user CubicalRipser for details.
  standardize = FALSE,  # boolean determining whether point cloud size should be standardized
  return_format = "df"  # defaults to "df", returning a data frame; if mat, returns a numeric matrix
)

Arguments

dataset  numeric array containing pixel/voxel data
threshold  maximum diameter for computation of Cubical complex
method  defaults to 0 for link join; alternatively, can be 1 for compute pairs. See original Cubical Ripser code at GitHub user CubicalRipser for details.
standardize  boolean determining whether point cloud size should be standardized
return_format  defaults to "df", returning a data frame; if mat, returns a numeric matrix

Value

3-column matrix with each row representing a TDA feature
Examples

# 2-dim example
dataset <- rnorm(10 ^ 2)
dim(dataset) <- rep(10, 2)
cubical_hom2 <- cubical(dataset)

# 3-dim example
dataset <- rnorm(8 ^ 3)
dim(dataset) <- rep(8, 3)
cubical_hom3 <- cubical(dataset)

# 4-dim example
dataset <- rnorm(5 ^ 4)
dim(dataset) <- rep(5, 4)
cubical_hom4 <- cubical(dataset)

ripserr

Calculate Persistent Homology with Ripser-Based Engines

Description

Ports Ripser-based persistent homology calculation engines from C++ to R using the Rcpp package.

vietoris_rips

Calculate Persistent Homology of a Point Cloud

Description

Calculates the persistent homology of a point cloud, as represented by a Vietoris-Rips complex. This function is an R wrapper for Ulrich Bauer’s Ripser C++ library for calculating persistent homology. For more information on the C++ library, see https://github.com/Ripser/ripser.

Usage

vietoris_rips(
    dataset,
    dim = 1,
    threshold = -1,
    p = 2L,
    format = "cloud",
    standardize = FALSE,
    return_format = "df"
)
Arguments

- **dataset**: numeric matrix containing point cloud or distance matrix
- **dim**: maximum dimension of features to calculate
- **threshold**: maximum diameter for computation of Vietoris-Rips complexes
- **p**: number of the prime field \( \mathbb{Z}/p\mathbb{Z} \) to compute the homology over
- **format**: format of mat, either "cloud" for point cloud or "distmat" for distance matrix
- **standardize**: boolean determining whether point cloud size should be standardized
- **return_format**: defaults to "df", returning a data frame; if mat, returns a numeric matrix

Details

The mat parameter should be a numeric matrix with each row corresponding to a single point, and each column corresponding to a single dimension. Thus, if mat has 50 rows and 5 columns, it represents a point cloud with 50 points in 5 dimensions. The dim parameter should be a positive integer. Alternatively, the mat parameter could be a distance matrix (upper triangular half is ignored); note: format should be specified as "distmat".

Value

3-column matrix or data frame, with each row representing a TDA feature

Examples

```r
# create a 2-d point cloud of a circle (100 points)
num.pts <- 100
rand.angle <- runif(num.pts, 0, 2*pi)
pt.cloud <- cbind(cos(rand.angle), sin(rand.angle))

# calculate persistent homology (num.pts by 3 numeric matrix)
pers.hom <- vietoris_rips(pt.cloud)
```
Index

cubical, 2
ripserr, 3
vietoris_rips, 3