Package ‘ripserr’

April 6, 2025

Title Calculate Persistent Homology with Ripser-Based Engines

Version 0.3.0

Description Ports the Ripser <doi:10.48550/arXiv.1908.02518>
and Cubical Ripser <doi:10.48550/arXiv.2005.12692>
persistent homology calculation engines from C++.
Can be used as a rapid calculation tool in topological data analysis
pipelines.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

URL `https://github.com/tdaverse/ripserr/`

BugReports `https://github.com/tdaverse/ripserr/issues`

LinkingTo Rcpp

Depends R (>= 3.5.0)

Imports Rcpp (>= 1.0), stats (>= 3.0), utils (>= 3.0)

Suggests covr (>= 3.5), knitr (>= 1.29), rmarkdown (>= 2.3), testthat
(>= 2.3), lmtest

VignetteBuilder knitr

NeedsCompilation yes

Author Raoul Wadhwa [aut] (<`https://orcid.org/0000-0003-0503-9580`>),
Matt Piekenbrock [aut],
Jacob Scott [aut] (<`https://orcid.org/0000-0003-2971-7673`>),
Jason Cory Brunson [aut, cre] (<`https://orcid.org/0000-0003-3126-9494`>),
Emily Noble [ctb],
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.),
Xinyi Zhang [aut]

Maintainer Jason Cory Brunson `<cornelioid@gmail.com>`

Repository CRAN

Date/Publication 2025-04-06 18:20:02 UTC

Contents

`aegypti` Aedes aegypti *occurrences in Brazil in 2013*

Description

A geographic dataset of known occurrences of *Aedes aegypti* mosquitoes in Brazil, derived from peer-reviewed and unpublished literature and reverse-geocoded to states.

Usage

```
aegypti
```

Format

A tibble of 4411 observations and 13 variables:

vector species identification (*aegypti* versus *albopictus*)

occurrence_id unique occurrence identifier

source_type published versus unpublished, with reference identifier

location_type point or polygon location

polygon_admin admin level or polygon size; -999 for point locations

y latitudinal coordinate of point or polygon centroid

x longitudinal coordinate of point or polygon centroid

status established versus transient population

state_name name of reverse-geolocated state

state_code two-letter state code

Source

doi:10.5061/dryad.47v3c

Examples

```
# calculate persistence data for occurrences in Acre
acre_coord <- aegypti[aegypti$state_code == "AC", c("x", "y"), drop = FALSE]
acre_rips <- vietoris_rips(acre_coord)
plot.new()
plot.window(
  xlim = c(0, max(acre_rips$death)),
  ylim = c(0, max(acre_rips$death)),
  asp = 1
)
axis(1L)
axis(2L)
abline(a = 0, b = 1)
points(acre_rips[acre_rips$dim == 0L, c("birth", "death")], pch = 16L)
points(acre_rips[acre_rips$dim == 1L, c("birth", "death")], pch = 17L)
```

`as.PHom` *Convert to PHom Object*

Description

Converts valid objects to `PHom` instances.

Usage

```
as.PHom(x, dim_col = 1, birth_col = 2, death_col = 3)
```

Arguments

`x`	object being converted to `PHom` instance
`dim_col`	either `integer` representing column index for feature dimension data or `character` representing column name
`birth_col`	either `integer` representing column index for feature birth data or `character` representing column name
`death_col`	either `integer` representing column index for feature death data or `character` representing column name

Value

PHom instance

Examples

```
# construct data frame with valid persistence data
df <- data.frame(dimension = c(0, 0, 1, 1, 1, 2),
                 birth = rnorm(6),
                 death = rnorm(6, mean = 15))

# convert to `PHom` instance and print
df_phom <- as.PHom(df)
df_phom

# print feature details to confirm accuracy
print.data.frame(df_phom)
```

`case_predictors` *State-level predictors of mosquito-borne illness in Brazil*

Description

A data set of numbers of cases of Dengue in each state of Brazil in 2013 and three state-level variables used in a predictive model.

Usage

```
case_predictors
```

Format

A data frame of 27 observations and 4 variables:

POP	state population in 2013
TEMP	average temperature across state municipalities
PRECIP	average precipitation across state municipalities
CASE	number of state Dengue cases in 2013

Source

`https://web.archive.org/web/20210209122713/https://www.gov.br/saude/pt-br/assuntos/boletins-epidemiologicos-1/por-assunto`, `http://www.ipeadata.gov.br/Default.aspx`, `https://ftp.ibge.gov.br/Estimativas_de_Populacao/`, `https://www.ibge.Goiasv.br/geociencias/organizaca`

Data pre-processing: After acquiring data from above links, we converted any dataset embedded in PDF format to CSV. Using carried functionalities in the CSV file, we sorted all datasets alphabetically based on state names to make later iterations more convenient. Also, we calculated the annual average temperature and added to the original dataset where it was documented by quarter.

`cubical` *Calculating Persistent Homology via a Cubical Complex*

Description

This function is an R wrapper for the CubicalRipser C++ library to calculate persistent homology. For more information on the C++ library, see `https://github.com/CubicalRipser`. For more information on how objects of different classes are evaluated by `cubical`, read the Details section below.

Usage

```
cubical(dataset, ...)

## S3 method for class 'array'
cubical(dataset, threshold = 9999, method = "lj", ...)

## S3 method for class 'matrix'
cubical(dataset, ...)
```

Arguments

`dataset`	object on which to calculate persistent homology
`...`	other relevant parameters
`threshold`	maximum simplicial complex diameter to explore
`method`	either `"lj"` (for Link Join) or `"cp"` (for Compute Pairs); see Kaji et al. (2020) `https://arxiv.org/abs/2005.12692` for details

Details

`cubical.array` assumes `dataset` is a lattice, with each element containing the value of the lattice at the point represented by the indices of the element in the `array`.

`cubical.matrix` is redundant for versions of `R` at or after 4.0. For previous versions of `R`, in which objects with class `matrix` do not necessarily also have class `array`, `dataset` is converted to an `array` and persistent homology is then calculated using `cubical.array`.

Value

PHom object

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)
```

`head.PHom` *First Part of PHom Object*

Description

Returns the first part of a `PHom` instance.

Usage

```
## S3 method for class 'PHom'
head(x, ...)
```

Arguments

`x`	object of class `PHom`
`...`	other parameters

Examples

```
# create sample persistence data
df <- data.frame(dimension = c(0, 0, 1, 1, 1, 2),
                 birth = rnorm(6),
                 death = rnorm(6, mean = 15))
df_phom <- as.PHom(df)

# look at first 3 features
head(df_phom)

# look at last 3 features
tail(df_phom)
```

`is.PHom` *Check PHom Object*

Description

Tests if objects are valid `PHom` instances.

Usage

```
is.PHom(x)
```

Arguments

`x`	object whose `PHom`-ness is being tested

Value

`TRUE` if `x` is a valid `PHom` object; `FALSE` otherwise

Examples

```
# create sample persistence data
df <- data.frame(dimension = c(0, 0, 1, 1, 1, 2),
                 birth = rnorm(6),
                 death = rnorm(6, mean = 15))
df <- as.PHom(df)

# confirm that persistence data is valid
is.PHom(df)

# mess up df object (feature birth cannot be after death)
df$birth[1] <- rnorm(1, mean = 50)

# confirm that persistence data is NOT valid
is.PHom(df)
```

`PHom` *Persistence Data Container*

Description

PHom() creates instances of `PHom` objects, which are convenient containers for persistence data. Generally, data frame (or similar) objects are used to create `PHom` instances with users specifying which columns contain dimension, birth, and death details for each feature.

Usage

```
PHom(x, dim_col = 1, birth_col = 2, death_col = 3)
```

Arguments

`x`	object used to create `PHom` instance
`dim_col`	either `integer` representing column index for feature dimension data or `character` representing column name
`birth_col`	either `integer` representing column index for feature birth data or `character` representing column name
`death_col`	either `integer` representing column index for feature death data or `character` representing column name

Value

`PHom` instance

Examples

```
# construct data frame with valid persistence data
df <- data.frame(dimension = c(0, 0, 1, 1, 1, 2),
                 birth = rnorm(6),
                 death = rnorm(6, mean = 15))

# create `PHom` instance and print
df_phom <- PHom(df)
df_phom

# print feature details to confirm accuracy
print.data.frame(df_phom)
```

`print.PHom` *Printing Persistence Data*

Description

Print a PHom object.

Usage

```
## S3 method for class 'PHom'
print(x, ...)
```

Arguments

`x`	object of class `PHom`
`...`	other parameters; ignored

Examples

```
# create circle dataset
angles <- runif(25, 0, 2 * pi)
circle <- cbind(cos(angles), sin(angles))

# calculate persistent homology
circle_phom <- vietoris_rips(circle)

# print persistence data
print(circle_phom)
```

`ripserr` *Calculate Persistent Homology with Ripser-Based Engines*

Description

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

Author(s)

Maintainer: Jason Cory Brunson `<cornelioid@gmail.com>` (ORCID)

Authors:

- Raoul Wadhwa `<raoulwadhwa@gmail.com>` (ORCID)
- Matt Piekenbrock `<matt.piekenbrock@gmail.com>`
- Jacob Scott (ORCID)
- Xinyi Zhang `<ezhang0927@gmail.com>`

Other contributors:

- Emily Noble [contributor]
- Takeki Sudo (Takeki Sudo is a copyright holder for Cubical Ripser (GPL-3 license), which was refactored prior to inclusion in ripserr.) [copyright holder, contributor]
- Kazushi Ahara (Kazushi Ahara is a copyright holder for Cubical Ripser (GPL-3 license), which was refactored prior to inclusion in ripserr.) [copyright holder, contributor]
- Ulrich Bauer (Ulrich Bauer holds the copyright to Ripser (MIT license), which was refactored prior to inclusion in ripserr.) [copyright holder, contributor]

See Also

Useful links:

- `https://github.com/tdaverse/ripserr/`
- Report bugs at `https://github.com/tdaverse/ripserr/issues`

`tail.PHom` *Last Part of PHom Object*

Description

Returns the last part of a `PHom` instance.

Usage

```
## S3 method for class 'PHom'
tail(x, ...)
```

Arguments

`x`	object of class `PHom`
`...`	other parameters

Examples

```
# create sample persistence data
df <- data.frame(dimension = c(0, 0, 1, 1, 1, 2),
                 birth = rnorm(6),
                 death = rnorm(6, mean = 15))
df_phom <- as.PHom(df)

# look at first 3 features
head(df_phom)

# look at last 3 features
tail(df_phom)
```

`vietoris_rips` *Calculate Persistent Homology via a Vietoris-Rips Complex*

Description

This function is an R wrapper for the Ripser C++ library to calculate persistent homology. For more information on the C++ library, see `https://github.com/Ripser/ripser`. For more information on how objects of different classes are evaluated by `vietoris_rips`, read the Details section below.

Usage

```
vietoris_rips(dataset, ...)

## S3 method for class 'data.frame'
vietoris_rips(dataset, ...)

## S3 method for class 'matrix'
vietoris_rips(dataset, max_dim = 1L, threshold = -1, p = 2L, dim = NULL, ...)

## S3 method for class 'dist'
vietoris_rips(dataset, max_dim = 1L, threshold = -1, p = 2L, dim = NULL, ...)

## S3 method for class 'numeric'
vietoris_rips(
  dataset,
  data_dim = 2L,
  dim_lag = 1L,
  sample_lag = 1L,
  method = "qa",
  ...
)

## S3 method for class 'ts'
vietoris_rips(dataset, ...)

## Default S3 method:
vietoris_rips(dataset, ...)
```

Arguments

`dataset`	object on which to calculate persistent homology
`...`	other relevant parameters
`max_dim`	maximum dimension of persistent homology features to be calculated
`threshold`	maximum simplicial complex diameter to explore
`p`	prime field in which to calculate persistent homology
`dim`	deprecated; passed to `max_dim` or ignored if `max_dim` is specified
`data_dim`	desired end data dimension
`dim_lag`	time series lag factor between dimensions
`sample_lag`	time series lag factor between samples (rows)
`method`	currently only allows `"qa"` (quasi-attractor method)

Details

`vietoris_rips.data.frame` assumes `dataset` is a point cloud, with each row representing a point and each column representing a dimension.

`vietoris_rips.matrix` currently assumes `dataset` is a point cloud (similar to `vietoris_rips.data.frame`). Currently in the process of adding network representation to this method.

`vietoris_rips.dist` takes a `dist` object and calculates persistent homology based on pairwise distances. The `dist` object could have been calculated from a point cloud, network, or any object containing elements from a finite metric space.

`vietoris_rips.numeric` and `vietoris_rips.ts` both calculate persistent homology of a time series object. The time series object is converted to a matrix using the quasi-attractor method detailed in Umeda (2017) doi:10.1527/tjsai.D-G72. Persistent homology of the resulting matrix is then calculated.

Value

PHom object

Examples

```
# 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