

Tutorial on the R package TDA

Jisu Kim

Brittany T. Fasy, Jisu Kim, Fabrizio Lecci, Clément Maria, Vincent Rouvreau

Abstract

This tutorial gives an introduction to the R package **TDA**, which provides some tools for Topological Data Analysis. The salient topological features of data can be quantified with persistent homology. The R package **TDA** provide an R interface for the efficient algorithms of the C++ libraries **GUDHI**, **Dionysus**, and **PHAT**. Specifically, The R package **TDA** includes functions for computing the persistent homology of the Rips complex, alpha complex, and alpha shape complex, and a function for the persistent homology of sublevel sets (or superlevel sets) of arbitrary functions evaluated over a grid of points. The R package **TDA** also provides a function for computing the confidence band that determines the significance of the features in the resulting persistence diagrams.

Keywords: Topological Data Analysis, Persistent Homology.

1. Introduction

R(<http://cran.r-project.org/>) is a programming language for statistical computing and graphics.

R has several good properties: R has many packages for statistical computing. Also, R is easy to make (interactive) plots. R is a script language, and it is easy to use. But, R is slow. C or C++ stands on the opposite end: C or C++ also has many packages(or libraries). But, C or C++ is difficult to make plots. C or C++ is a compiler language, and is difficult to use. But, C or C++ is fast. In short, R has short development time but long execution time, and C or C++ has long development time but short execution time.

Several libraries are developed for Topological Data Analysis: for example, **GUDHI**(<https://project.inria.fr/gudhi/software/>), **Dionysus**(<http://www.mrzv.org/software/dionysus/>), and **PHAT**(<https://code.google.com/p/phat/>). They are all written in C++, since Topological Data Analysis is computationally heavy and R is not fast enough.

R package **TDA**(<http://cran.r-project.org/web/packages/TDA/index.html>) bridges between C++ libraries(**GUDHI**, **Dionysus**, **PHAT**) and R. **TDA** package provides an R interface for the efficient algorithms of the C++ libraries **GUDHI**, **Dionysus** and **PHAT**. So by using **TDA** package, short development time and short execution time can be both achieved.

R package **TDA** provides tools for Topological Data Analysis. You can compute several different things with **TDA** package: you can compute common distance functions and density estimators, the persistent homology of the Rips filtration, the persistent homology of sublevel sets of a function over a grid, the confidence band for the persistence diagram, and the cluster density trees for density clustering.

2. Installation

First, you should download R. R of version at least 3.1.0 is required:

<http://cran.r-project.org/bin/windows/base/> (for Windows)

<http://cran.r-project.org/bin/macosx/> (for (Mac) OS X)

R is part of many Linux distributions, so you should check with your Linux package management system.

You can use whatever IDE that you would like to use (Rstudio, Eclipse, Emacs, Vim...). R itself also provides basic GUI or CUI. I personally use Rstudio:

<http://www.rstudio.com/products/rstudio/download/>

For Windows and Mac, you can install R package **TDA** as in the following code (or pushing 'Install R packages' button if you use Rstudio).

```
#####
# installing R package TDA
#####
if (!require(package = "TDA")) {
  install.packages(pkgs = "TDA")
}

## Loading required package: TDA
```

If you are using Linux, you should install R package **TDA** from the source. To do this, you need to install two libraries in advance: gmp (<https://gmplib.org/>) and mpfr (<http://www.mpfr.org/>). Installation of these packages may differ by your Linux distributions. Once those libraries are installed, you need to install four R packages: **parallel**, **FNN**, **igraph**, and **scales**. **parallel** is included when you install R, so you need to install **FNN**, **igraph**, and **scales** by yourself. You can install them by following code (or pushing 'Install R packages' button if you use Rstudio).

```
#####
# installing required packages
#####
if (!require(package = "FNN")) {
  install.packages(pkgs = "FNN")
}

## Loading required package: FNN

if (!require(package = "igraph")) {
  install.packages(pkgs = "igraph")
}

## Loading required package: igraph
##
## Attaching package: 'igraph'
## The following object is masked from 'package:FNN':
##
## knn
## The following objects are masked from 'package:stats':
##
## decompose, spectrum
## The following object is masked from 'package:base':
##
## union
```

```

if (!require(package = "scales")) {
  install.packages(pkgs = "scales")
}

## Loading required package: scales

```

Then you can install the R package **TDA** as in Windows or Mac:

```

#####
# installing R package TDA
#####
if (!require(package = "TDA")) {
  install.packages(pkgs = "TDA")
}

```

Once installation is done, R package **TDA** should be loaded as in the following code, before using the package functions.

```

#####
# loading R package TDA
#####
library(package = "TDA")

```

3. Sample on manifolds, Distance Functions, and Density Estimators

3.1. Uniform Sample on manifolds

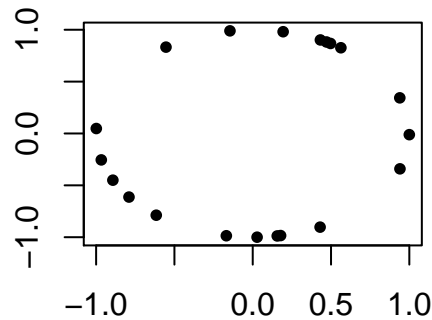
A set of n points $X = \{x_1, \dots, x_n\} \subset \mathbb{R}^d$ has been sampled from some distribution P .

- n sample from the uniform distribution on the circle in \mathbb{R}^2 with radius r .

```

#####
# uniform sample on the circle
#####
circleSample <- circleUnif(n = 20, r = 1)
plot(circleSample, xlab = "", ylab = "", pch = 20)

```



3.2. Distance Functions, and Density Estimators

We compute distance functions and density estimators over a grid of points. Suppose a set of points $X = \{x_1, \dots, x_n\} \subset \mathbb{R}^d$ has been sampled from some distribution P . The following code generates a sample of 400 points from the unit circle and constructs a grid of points over which we will evaluate the functions.

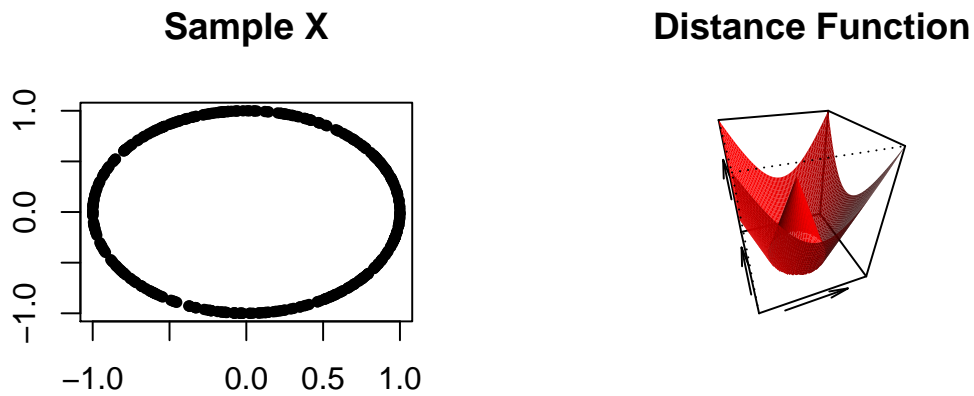
```
#####
# uniform sample on the circle, and grid of points
#####
X <- circleUnif(n = 400, r = 1)

lim <- c(-1.7, 1.7)
by <- 0.05
margin <- seq(from = lim[1], to = lim[2], by = by)
Grid <- expand.grid(margin, margin)
```

- The distance function is defined for each $y \in \mathbb{R}^d$ as $\Delta(y) = \inf_{x \in X} \|x - y\|_2$.

```
#####
# distance function
#####
distance <- distFct(X = X, Grid = Grid)

par(mfrow = c(1,2))
plot(X, xlab = "", ylab = "", main = "Sample X", pch = 20)
persp(x = margin, y = margin,
      z = matrix(distance, nrow = length(margin), ncol = length(margin)),
      xlab = "", ylab = "", zlab = "", theta = -20, phi = 35, scale = FALSE,
      expand = 3, col = "red", border = NA, ltheta = 50, shade = 0.5,
      main = "Distance Function")
```



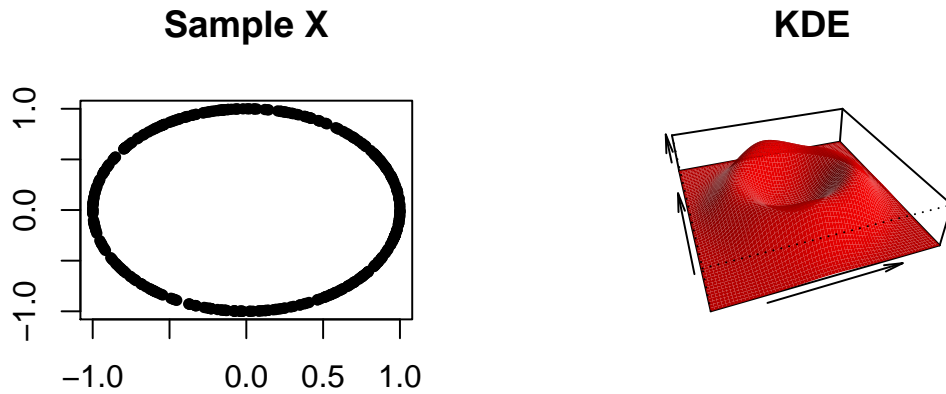
- The Gaussian Kernel Density Estimator (KDE), for each $y \in \mathbb{R}^d$, is defined as

$$\hat{p}_h(y) = \frac{1}{n(\sqrt{2\pi}h)^d} \sum_{i=1}^n \exp\left(-\frac{\|y - x_i\|_2^2}{2h^2}\right).$$

where h is a smoothing parameter.

```
#####
# kernel density estimator
#####
h <- 0.3
KDE <- kde(X = X, Grid = Grid, h = h)

par(mfrow = c(1,2))
plot(X, xlab = "", ylab = "", main = "Sample X", pch = 20)
persp(x = margin, y = margin,
      z = matrix(KDE, nrow = length(margin), ncol = length(margin)),
      xlab = "", ylab = "", zlab = "", theta = -20, phi = 35, scale = FALSE,
      expand = 3, col = "red", border = NA, ltheta = 50, shade = 0.5,
      main = "KDE")
```



4. Persistent Homology and Landscape

4.1. Persistent Homology Over a Grid

`gridDiag` function computes the persistent homology of sublevel (and superlevel) sets of the functions. The function `gridDiag` evaluates a given real valued function over a triangulated grid (in arbitrary dimension), constructs a filtration of simplices using the values of the function, and computes the persistent homology of the filtration. The user can choose to compute persistence diagrams using either the **Dionysus** library (`library = "Dionysus"`) or the **PHAT** library (`library = "PHAT"`).

The following code computes the persistent homology of the superlevel sets (`sublevel = FALSE`) of the kernel density estimator (`FUN = kde`, `h = 0.3`) using the point cloud stored in the matrix `X` from the previous example. The other inputs are the features of the grid over which the `kde` is evaluated (`lim` and `by`), and a logical variable that indicates whether a progress bar should be printed (`printProgress`).

```
#####
# persistent homology of a function over a grid
#####
Diag <- gridDiag(X = X, FUN = kde, lim = cbind(lim, lim), by = by,
  sublevel = FALSE, library = "Dionysus", printProgress = FALSE, h = 0.3)
```

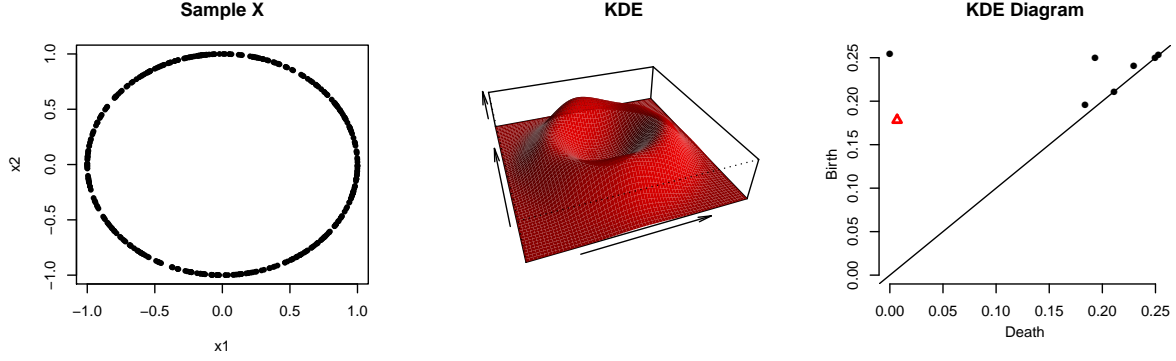
The function `plot` plots persistence diagram for objects of the class "diagram".

```
#####
# plotting persistence diagram
#####
par(mfrow = c(1,3))
plot(X, main = "Sample X", pch = 20)
persp(x = margin, y = margin,
  z = matrix(KDE, nrow = length(margin), ncol = length(margin)),
```

```

xlab = "", ylab = "", zlab = "", theta = -20, phi = 35, scale = FALSE,
expand = 3, col = "red", border = NA, ltheta = 50, shade = 0.9,
main = "KDE")
plot(x = Diag[["diagram"]], main = "KDE Diagram")

```



4.2. Landscape

The persistence landscape is a collection of continuous, piecewise linear functions $\lambda: \mathbb{Z}^+ \times \mathbb{R} \rightarrow \mathbb{R}$ that summarizes a persistence diagram. Consider the set of functions created by tenting each point $p = (x, y) = (\frac{b+d}{2}, \frac{d-b}{2})$ representing a birth-death pair (b, d) in the persistence diagram D as follows:

$$\Lambda_p(t) = \begin{cases} t - x + y & t \in [x - y, x] \\ x + y - t & t \in (x, x + y] \\ 0 & \text{otherwise} \end{cases} = \begin{cases} t - b & t \in [b, \frac{b+d}{2}] \\ d - t & t \in (\frac{b+d}{2}, d] \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

We obtain an arrangement of piecewise linear curves by overlaying the graphs of the functions $\{\Lambda_p\}_p$; see Figure 1 (left). The persistence landscape of D is the collection of functions

$$\lambda(k, t) = k\max_p \Lambda_p(t), \quad t \in [0, T], k \in \mathbb{N}, \quad (2)$$

where $k\max$ is the k th largest value in the set. see Figure 1 (middle).

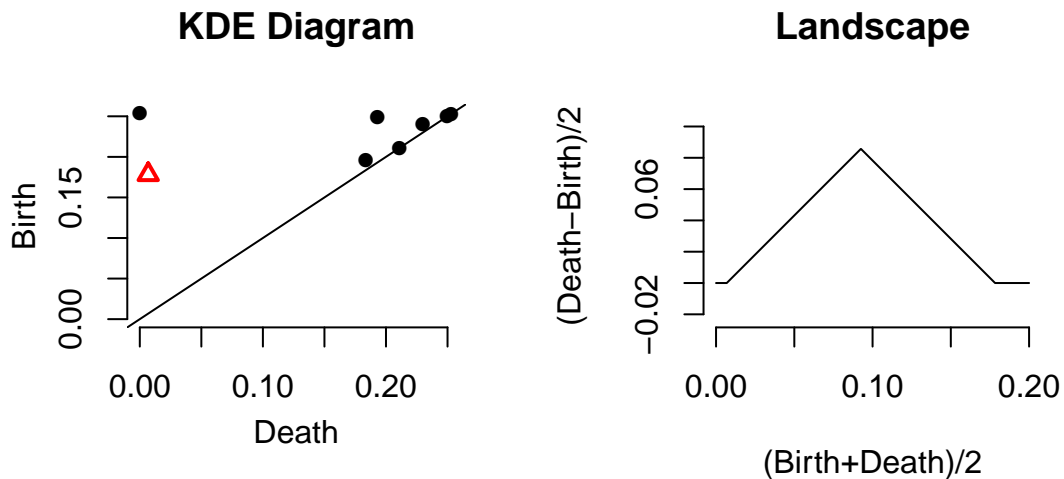


Figure 1: Left: we use the rotated axes to represent a persistence diagram D . A feature $(b, d) \in D$ is represented by the point $(\frac{b+d}{2}, \frac{d-b}{2})$ (pink). Right: the blue curve is the landscape $\lambda(1, \cdot)$.

landscape evaluates the landscape function over a one-dimensional grid of points `tseq`. In the following code, we use the `rips` persistence diagram in previous example to construct the

corresponding landscape for one-dimensional features (`dimension = 1`). The option `KK = 1` specifies that we are interested in the 1st landscape function. `landscape` return a real valued vector, which can be simply plotted with `plot(tseq, Land, type = "l")`.

```
#####
# computing landscape
#####
tseq <- seq(from = 0, to = 0.2, length = 1000) #domain
Land <- landscape(Diag = Diag[["diagram"]], dimension = 1, KK = 1, tseq = tseq)
par(mfrow=c(1,2))
plot(x = Diag[["diagram"]], main = "KDE Diagram")
plot(tseq, Land, type = "l", xlab = "(Birth+Death)/2",
      ylab = "(Death-Birth)/2", asp = 1, axes = FALSE, main = "Landscape")
axis(1); axis(2)
```



5. Statistical Inference on Persistent Homology and Landscape

$(1 - \alpha)$ confidence band can be computed for a function using the bootstrap algorithm, which we briefly describe using the kernel density estimator:

1. Given a sample $X = \{x_1, \dots, x_n\}$, compute the kernel density estimator \hat{p}_h ;
2. Draw $X^* = \{x_1^*, \dots, x_n^*\}$ from $X = \{x_1, \dots, x_n\}$ (with replacement), and compute $\theta^* = \sqrt{n} \|\hat{p}_h^*(x) - \hat{p}_h(x)\|_\infty$, where \hat{p}_h^* is the density estimator computed using X^* ;
3. Repeat the previous step B times to obtain $\theta_1^*, \dots, \theta_B^*$;
4. Compute $q_\alpha = \inf \left\{ q : \frac{1}{B} \sum_{j=1}^B I(\theta_j^* \geq q) \leq \alpha \right\}$;
5. The $(1 - \alpha)$ confidence band for $\mathbb{E}[\hat{p}_h]$ is $\left[\hat{p}_h - \frac{q_\alpha}{\sqrt{n}}, \hat{p}_h + \frac{q_\alpha}{\sqrt{n}} \right]$.

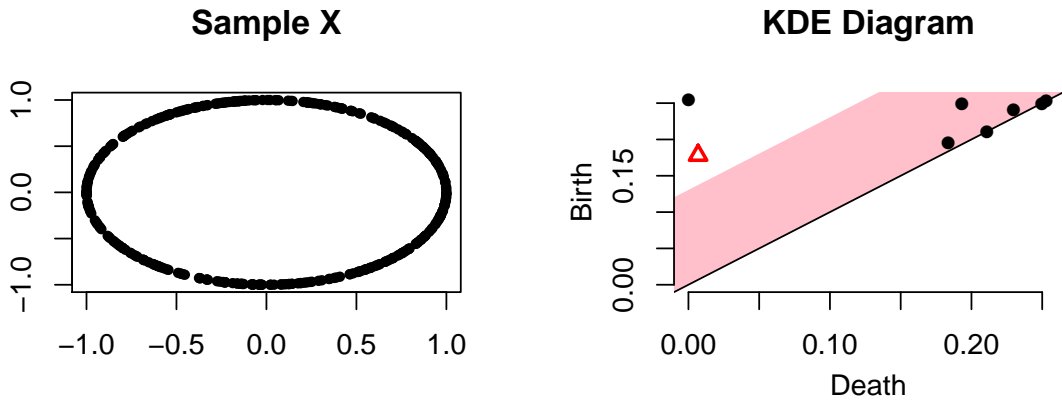
`bootstrapBand` computes $(1 - \alpha)$ bootstrap confidence band, with the option of parallelizing the algorithm (`parallel=TRUE`). The following code computes a 90% confidence band for $\mathbb{E}[\hat{p}_h]$.


```
#####
# bootstrap confidence band for kde function
#####
bandKDE <- bootstrapBand(X = X, FUN = kde, Grid = Grid, B = 100,
                        parallel = FALSE, alpha = 0.1, h = h)
print(bandKDE[["width"]])

##          90%
## 0.06526216
```

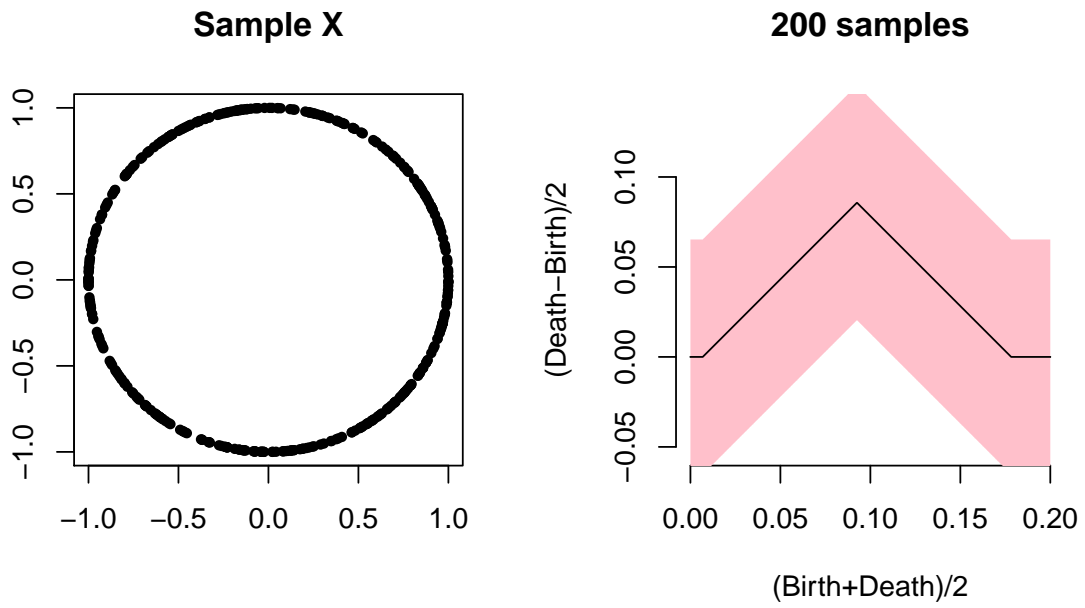
Then such confidence band for $\mathbb{E}[\hat{p}_h]$ can be used as the confidence band for the persistent homology.

```
#####
# bootstrap confidence band for persistent homology over a grid
#####
par(mfrow = c(1,2))
plot(X, xlab = "", ylab = "", main = "Sample X", pch = 20)
plot(x = Diag[["diagram"]], band = 2 * bandKDE[["width"]],
     main = "KDE Diagram")
```



Such confidence band for $\mathbb{E}[\hat{p}_h]$ can also be used as the confidence band for the landscape.

```
#####
# bootstrap confidence band for persistent homology over a grid
#####
par(mfrow = c(1,2))
plot(X, xlab = "", ylab = "", main = "Sample X", pch = 20)
plot(tseq, Land, type = "l", xlab = "(Birth+Death)/2",
     ylab = "(Death-Birth)/2", asp = 1, axes = FALSE, main = "200 samples")
axis(1); axis(2)
polygon(c(tseq, rev(tseq)), c(Land - bandKDE[["width"]],
                             rev(Land + bandKDE[["width"]])), col = "pink", lwd = 1.5,
       border = NA)
lines(tseq, Land)
```

**Affiliation:**

Firstname Lastname

Affiliation

Address, Country

E-mail: `name@address`URL: `http://link/to/webpage/`