# Short Introduction to Topolgical Data Analysis Using TDA Package in R

Caleb Torres

Spring 2019

## 1 Introduction

The tools of Topological Data Analysis (TDA) provide novel ways to analyze data sets for topological features. This short paper will introduce some of the topological concepts involved in this type of analysis, and briefly demonstrate how to analyze data sets using the R package "TDA."

In general, Topology studies the global structures of topological spaces, and is concerned with identifyig features such as connectivity, loops, and boundaries. Sometimes, these topological features are present in data. Thus, the goal of Topological Data Analysis is to provide a set of tools that inspect data for such features.

## 2 Topological Groundwork

For this short introduction to TDA, I will briefly review several topological concepts. These include an introduction to simplicial complexes, their homologies, as well as their persistant homologies.

### 2.1 Simplicial Complexes

The construction of simplicial complexes on data sets allows for an effective study of the topological features of data. This is because a simplicial complex essentially converts data into a topological space in order to reveal certain features wich are important to topology.

One of these important features is connectivity. Given a data set $\mathbb{X}$ reminiscent of a point cloud, the goal is to extract certain topological features using only the data points and a notion of distance.

More technically, a simplicial complex known as the Vietoris-Rips complex, $Rips_\alpha(\mathbb{X})$, is the set of simplices $[x_0, x_1, ..., x_k]$ such that $d_\mathbb{X}(x_i, x_j) \leq \alpha$ for all $(i, j)$. That is, for some parameter $\alpha$, if the value of the metric is less than the parameter, then we contruct a simplex between two points in the space $\mathbb{X}$. Generally, the space begins with one-simplex formations, but may eventually spawn 2-simplices and up to n-simplices, depending on the dimension of the space.

## 2.2 Homology

Another useful topological concept to consider is homology. In general, if $X$ is a topological space, then the homology of $X$ is a set of Abelian groups $H_k(X)_{k=0}^\infty$, called homology groups. However, for our purposes, we denote the kth homology group, $H_k(X)$, as being generated by elements representing k-dimensional features in $X$, such as holes and cycles.

Thus, computing the homology of a simplicial complex built on data will inform us about any interesting topological features that are present in the data at any given time.

## 2.3 Persistent Homology

To understand the nature of a given data set, it also important to know which homological features persist after changing the value of a given parameter. To handle this notion of persistance, we introduce the persistant homology of a topological space. Generally, the persistant homology of a topolgical structure will keep track of when certain homological features register and collapse over an interval, usually defined for a parameter $t$.

We often use a barcode diagram to display the persistance of certain homological features over the varying of a parameter $t$.

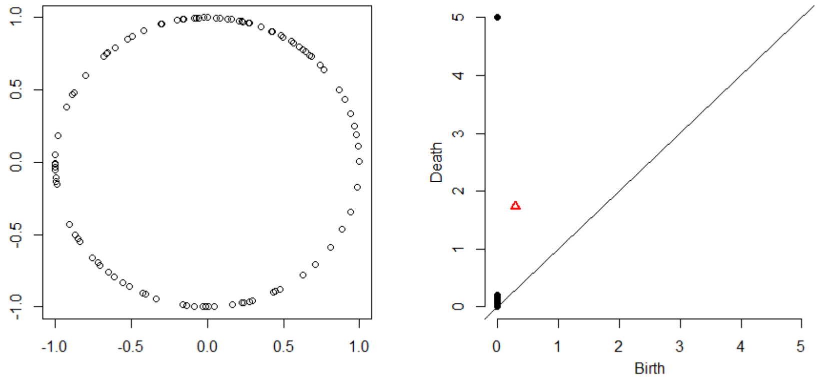
# 3 TDA Summary of Simple Data Set

We now summarize and compute the topological features of different data sets using the R package TDA. In particular, we will analyze the homological features of three different data sets. The first data set consists of a circle point cloud. The other two sets consist of two and three circle point clouds respectively.

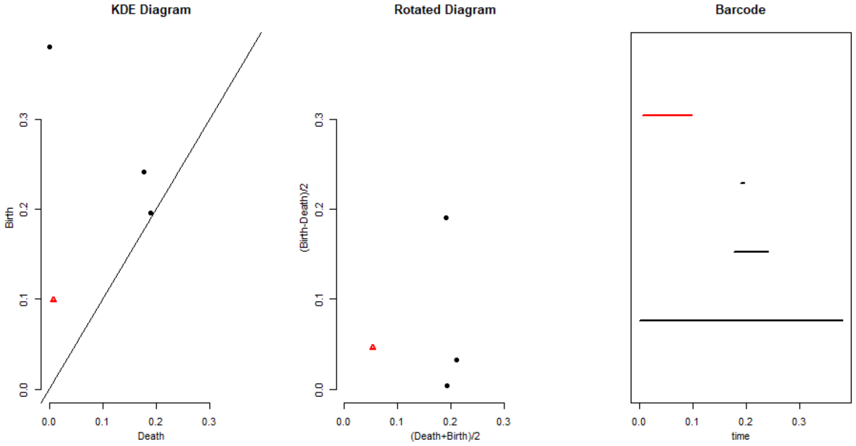
The TDA package generates sample plots corresponding to each data set, as well as their respective barcode plots and persistance diagrams. These plots are sufficient for our analysis.

## 3.1 Computations: Barcodes and Persistent Homology Plots

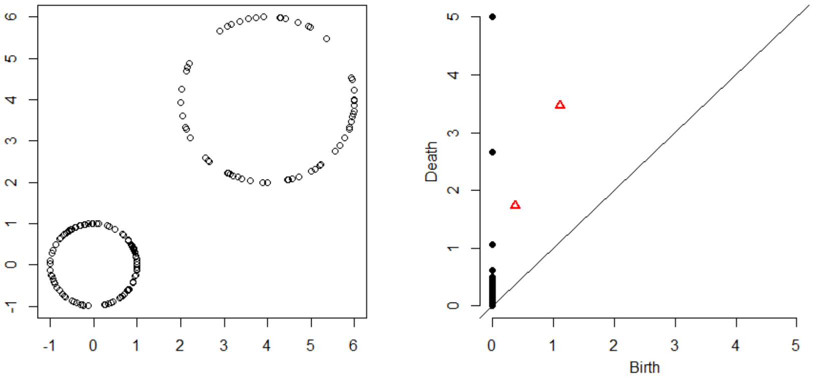
The plot below displays a sample set of 100 points that form a circle of radius one. Displayed next to this plot is the persistance diagram that shows the persistence of several cycles present in the data. Black points represent connected components, or paths between points, and red triangles one-cycles, or loops, present in the data. Note that our circle data shows the persistance of a loop or one-cycle.



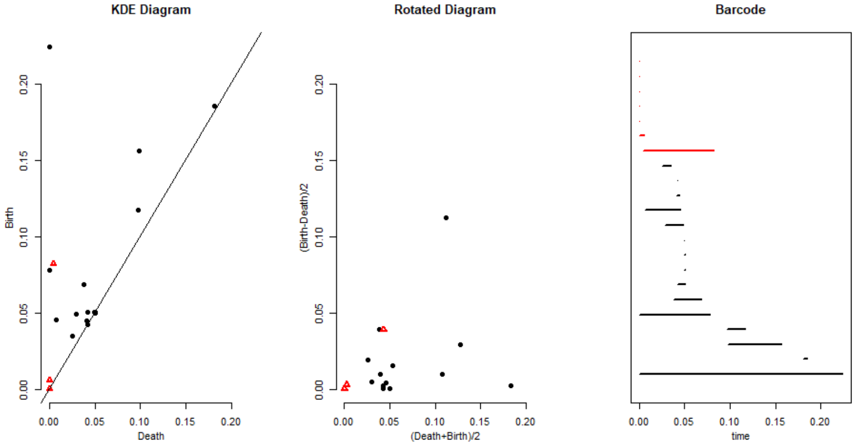
The following plots show several other diagrams, inlcuding the barcode diagram for our data. The barcode shows the persistance of a one-cycle in our data with a sprawled red line. The persistance of a cycle is measured against a time parameter.



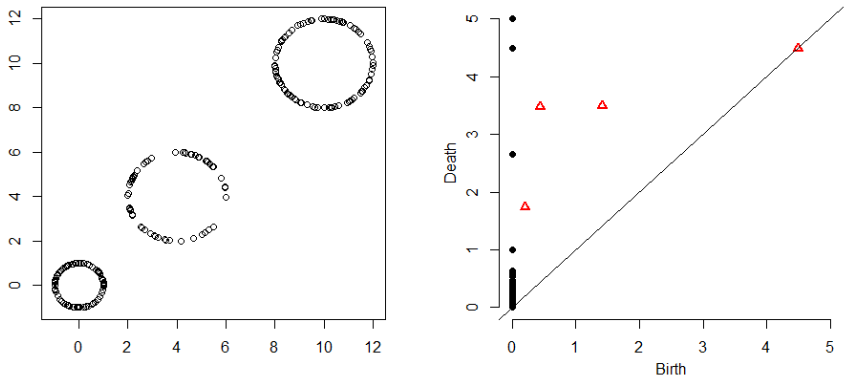
The following plot displays two cirlce sample sets. Note how the persistance diagram records the presence of two separate one-cycles in our data.



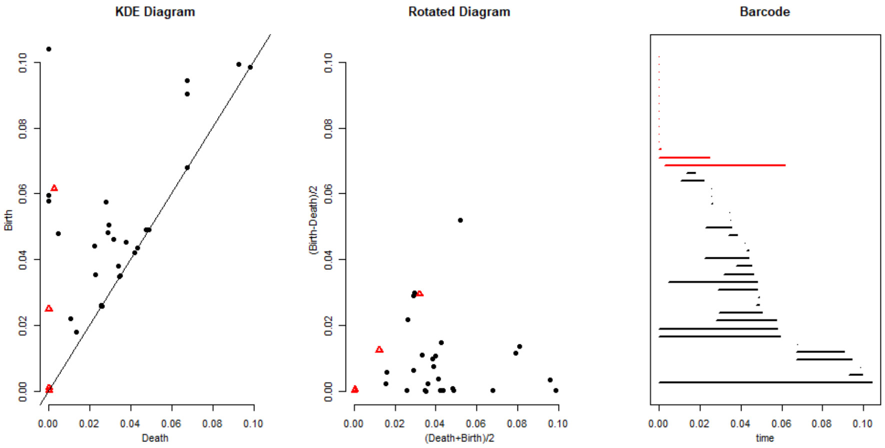
For our two circle data, note how the barcode also records the persistance of two one-cycles in the data. The persistance of the smaller circle's one-cycle is shorter, since the loop collapses earlier in time.



Lastly, we display the plot of three sample circle sets. Note how the persistance diagram records the presence of three separate one-cycles present in the data.



The barcode diagram now displays the persistance of three one-cycles in our data's simplicial complex. They again vary in persistance according to circle size.



# 4 Appendix: Source Code

```
 ###########################################################################
# loading R package TDA
###########################################################################
library(package = "TDA")
###########################################################################
# generating samples from two circles
###########################################################################
C1 <- circleUnif(n = 100)
C2 <- circleUnif(n = 60, r = 2) + 4
C3 <- circleUnif(n = 100, r = 2) + 10
TC <- rbind(C1, C2, C3)
```

```
plot(TC, xlab="", ylab="")
##########################################################################
# uniform sample on the circle
##########################################################################
circleSample <- TC
plot(circleSample)
#########################################################################
# uniform sample on the circle, and grid of points
##########################################################################
X <- TC
Xlim <- c(-20, 20)
Ylim <- c(-20, 20)
by <- 0.065
Xseq <- seq(from = Xlim[1], to = Xlim[2], by = by)
Yseq <- seq(from = Ylim[1], to = Ylim[2], by = by)
Grid <- expand.grid(Xseq, Yseq)
##########################################################################
# distance function
##########################################################################
distance <- distFct(X = X, Grid = Grid)
par(mfrow = c(1,2))
plot(X, xlab = "", ylab = "", main = "Sample X")
persp(x = Xseq, y = Yseq,
      z = matrix(distance, nrow = length(Xseq), ncol = length(Yseq)),
      xlab = "", ylab = "", zlab = "", theta = -20, phi = 35, scale = FALSE,
      expand = 3, col = "red", border = NA, ltheta = 50, shade = 0.5,
      main = "Distance Function")
##########################################################################
# uniform sample on the circle, and grid of points
##########################################################################
#X <- Circles
#Xlim <- c(-1.6, 1.6)
#Ylim <- c(-1.7, 1.7)
#by <- 0.065
#Xseq <- seq(from = Xlim[1], to = Xlim[2], by = by)
#Yseq <- seq(from = Ylim[1], to = Ylim[2], by = by)
#Grid <- expand.grid(Xseq, Yseq)
##########################################################################
# distance function
##########################################################################
distance <- distFct(X = X, Grid = Grid)
par(mfrow = c(1,2))
```

```
plot(X, xlab = "", ylab = "", main = "Sample X")
persp(x = Xseq, y = Yseq,
      z = matrix(distance, nrow = length(Xseq), ncol = length(Yseq)),
      xlab = "", ylab = "", zlab = "", theta = -20, phi = 35, scale = FALSE,
      expand = 3, col = "red", border = NA, ltheta = 50, shade = 0.5,
      main = "Distance Function")
##########################################################################
# distance to measure
##########################################################################
m0 <- 0.1
DTM <- dtm(X = X, Grid = Grid, m0 = m0)
par(mfrow = c(1,2))
plot(X, xlab = "", ylab = "", main = "Sample X")
persp(x = Xseq, y = Yseq,
      z = matrix(DTM, nrow = length(Xseq), ncol = length(Yseq)),
      xlab = "", ylab = "", zlab = "", theta = -20, phi = 35, scale = FALSE,
      expand = 3, col = "red", border = NA, ltheta = 50, shade = 0.5,
      main = "DTM")
##########################################################################
# k nearest neighbor density estimator
##########################################################################
k <- 60
kNN <- knnDE(X = X, Grid = Grid, k = k)
par(mfrow = c(1,2))
plot(X, xlab = "", ylab = "", main = "Sample X")
persp(x = Xseq, y = Yseq,
      z = matrix(kNN, nrow = length(Xseq), ncol = length(Yseq)),
      xlab = "", ylab = "", zlab = "", theta = -20, phi = 35, scale = FALSE,
      expand = 3, col = "red", border = NA, ltheta = 50, shade = 0.5,
      main = "kNN")
##########################################################################
# 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")
persp(x = Xseq, y = Yseq,
      z = matrix(kNN, nrow = length(Xseq), ncol = length(Yseq)),
      xlab = "", ylab = "", zlab = "", theta = -20, phi = 35, scale = FALSE,
      expand = 3, col = "red", border = NA, ltheta = 50, shade = 0.5,
      main = "KDE")
```

```
###########################################################################
# kernel distance
###########################################################################
h <- 0.3
Kdist <- kernelDist(X = X, Grid = Grid, h = h)
par(mfrow = c(1,2))
plot(X, xlab = "", ylab = "", main = "Sample X")
persp(x = Xseq, y = Yseq,
      z = matrix(Kdist, nrow = length(Xseq), ncol = length(Yseq)),
      xlab = "", ylab = "", zlab = "", theta = -20, phi = 35, scale = FALSE,
      expand = 3, col = "red", border = NA, ltheta = 50, shade = 0.5,
      main = "Kernel Distance")
###########################################################################
# persistent homology of a function over a grid
###########################################################################
Diag <- gridDiag(X = X, FUN = kde, lim = cbind(Xlim, Ylim), by = by,
                 sublevel = FALSE, library = "Dionysus", printProgress = FALSE, h = 0.3)
###########################################################################
# plotting persistence diagram
###########################################################################
par(mfrow = c(1,3))
plot(X, main = "Sample X")
persp(x = Xseq, y = Yseq, z = matrix(KDE, nrow = length(Xseq), ncol = length(Yseq)),
      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")
###########################################################################
# other options for plotting persistence diagram
###########################################################################
par(mfrow = c(1,3))
plot(Diag[["diagram"]], main = "KDE Diagram")
plot(Diag[["diagram"]], rotated = TRUE, main = "Rotated Diagram")
plot(Diag[["diagram"]], barcode = TRUE, main = "Barcode")
###########################################################################
# generating samples from two circles
###########################################################################
#Circle1 <- circleUnif(n = 60)
#Circle2 <- circleUnif(n = 60, r = 2) + 3
#Circles <- rbind(Circle1, Circle2)
#plot(Circles, xlab="", ylab="")
###########################################################################
```

```
# Rips persistence diagram
###########################################################################
Diag <- ripsDiag(X = X, maxdimension = 1, maxscale = 5,
                 library = "GUDHI", printProgress = FALSE)
par(mfrow=c(1,2))
plot(X, xlab="", ylab="")
plot(Diag[["diagram"]])
```

## References

[1] Bubenik, Peter, *Topology for Data Science 1: An Introduction to Topological Data Analysis.* `https://people.clas.ufl.edu/peterbubenik/files/abacus_1.pdf`

[2] Carlsson, Gunnar, *Topology and Data.* `https://web.stanford.edu/group/mmds/slides2008/carlss`

[3] Chazal, Frederic and Betrand Michel, *An Introduction to Topological Data Analysis fundamental and practical aspects for data scientists.* `https://arxiv.org/pdf/1710.04019.pdf`

[4] Kim, Jisu, *Tutorial on the R Package TDA.* `http://www.stat.cmu.edu/topstat/topstat_old/Talks`

[5] Sanchez, Giancarlo, *Topological Inference for Modern Data Analysis.* `http://faculty.fiu.edu/~yotovm/GiancarloMasters-project.pdf`

[6] Alfa, Zomorodian, *Topological Data Analysis.* `http://citeseerx.ist.psu.edu/viewdoc/download?d`