Persistent Homology Testing

Mike Wu

September 19, 2015

The goal of this exercise is to test different types of topological analysis on a fake dataset created from sampling many uniform circles. Because a circle is made up of a single loop and void, it serves as a good baseline to differentiate the methods.

1 Setup

Each of the following diagrams are generated via N simulations of sampling n times from an uniform circle of radius r and center (c_0, c_1). Many of the functions used in the diagrams come from R's topological data analysis library.

```
circle <- function(num=n, rad=r, center=c(0, 0)) {
  x0 <- center[1]
  y0 <- center[2]
  u <- 2 * pi * runif(num)
  result <- rad * cbind(x = cos(u) + x0, y = sin(u) + y0)
  return(result)
}
```

2 RIPS Diagram

For each of the N simulations, it would be informative to plot the RIPS diagram for each N simulation overlayed on top of one another. By doing so should shed light on groups of 0-dimensional, and 1-dimensional objects like voids and loops.

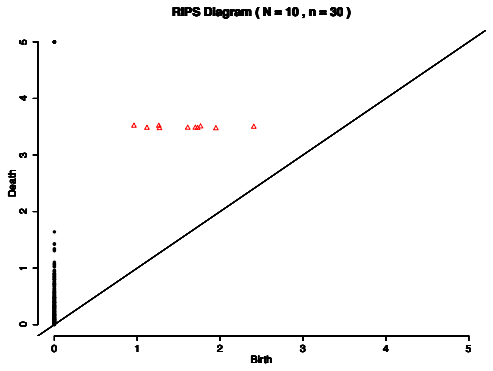
```
multiRIPS <- function(maxdimension, maxscale, N, K) {
  for (i in 1:N) {
    Circle <- circle(K, radius, center)
    Diag <- ripsDiag(Circle, maxdimension, maxscale,
                     library = "GUDHI", printProgress = FALSE)
    par(new = TRUE)
    plot(Diag$diagram, main = "RIPS Diagram")
  }
}
```

An example of usage of multiRIPS:

```
maxdimension <- 1          # components and loops
maxscale     <- 5          # limit of the filtration
```

```
radius      <- 2          # radius of circle
center      <- c(0, 0)    # center of circle
multiRIPS(maxdimension, maxscale, 10, 100)
```

For all RIPS plots, maxdimension is set to 1, and maxscale is set to 5. We are only looking for loops, and limit filtration to 5. Circles are centered at the origin, with a radius of 2, 100 samples are drawn, and 10 simulations are conducted.



More interesting is to see what happens when n, the number of points sampled for each circle, varies (i.e. $n = 10, 100, 1000, ...$). Intuitively, I would expect a smaller n to less perfectly represent a circle, possibly making it more difficult to find the correct loops and voids. As n grows large, the shape and characteristics of the circle should be more readily available.

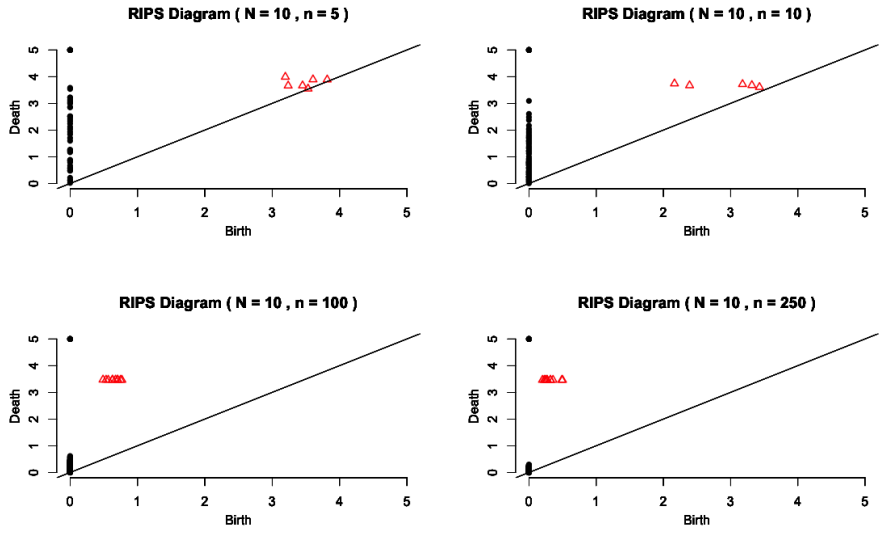


Figure 1: Plots of overlapping RIPS diagrams. The number of simulations(N) is held constant but the sample size is varied (n).

Something interesting is that growing from n = 5, to n = 10 to n = 100, the program gets much slower. I think ripsDiag as a function does not scale linearly. This makes sense since with a large point cloud, growing individual balls at each point and running linear algebraic operations throughout is an expensive procedure.

Two immediate recognizable patterns are that (1) the greater the sample size (bigger n), the earlier the detection (and formation) of higher dimensional simplicial complexes. For example, when $n = 250$, the loop is detected as early as 0.1 units after birth, while when $n = 5$, the loop is detection after 3.5 units..., and (2), as n grows larger, the amount of spread of deaths as time 0 is more concentrated. Compare the black dots when $n = 5$ to when $n = 250$. Intuitively, these observations make a lot of sense. Because there are more points, the circle is better interpolated, meaning that more points lie on the circle's perimeter. When the balls begin to grow, the amount of growth needed to form all the overlaps to find the circle is small. Thus, the loop is found immediately. When there are very few points, the balls need much longer to grow to overlap! An interesting conclusions from this is that given any 3 random points, there will be a loop found, even if there isn't actual structure to those three points. But most likely, that structure would not persist over time. The second observation is also explainable by the sample size. With more points, fewer initial 0-dimensional complexes appear (points), because points start out close enough to form longer segments. When n is small, the points are individual complexes, leading to increased deaths.

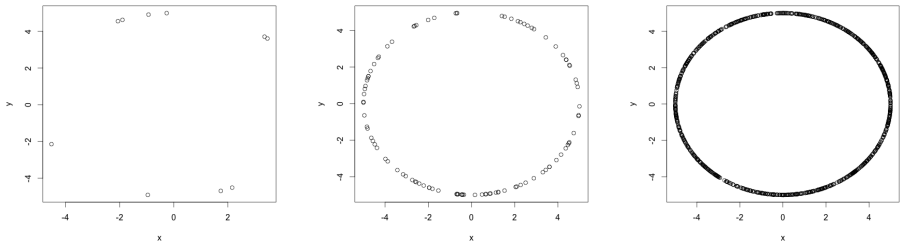


Figure 2: Plots of generated circles of different sample sizes.

3 KDE Diagram

A comparable series of operations can be done for Kernel Density Estimates in lieu of RIPS diagrams.

```
multiKDE <- function(N, Xlim, Ylim, by, h=0.3) {
  for (i in 1:N) {
    Circle <- circle(numparticle, radius, center)
    Diag <- gridDiag(X = Circle, FUN = kde, lim = cbind(Xlim, Ylim),
                     by = by, sublevel = FALSE, library = "Dionysus"
                     printProgress = FALSE, h = h)
    par(new = TRUE)
```

```
    plot(Diag$diagram, main = "KDE Diagram")
  }
}
```

An example of usage of multiKDE:

```
numparticle <- 10
radius      <- 2
center      <- c(0, 0)
X           <- circle(numparticle, radius, center)
Xlim        <- c(center[1] - radius - 1, center[2] + radius + 1)
Ylim        <- c(center[1] - radius - 1, center[2] + radius + 1)
by          <- 0.1
multiKDE(10, Xlim, Ylim, by, 0.3)
```

The same plots (n = 5, 10, 100, 1000) are generated for KDE: Again, circles are centered at the origin, with a radius of 2, and 10 simulations are conducted. Distance algorithms require a grid size. Our grid is defined by a box with a side length of radius + 1. The step size for the grid is set to 0.1. 0.3 is passed as the smoothing parameter for the gaussian kernel.

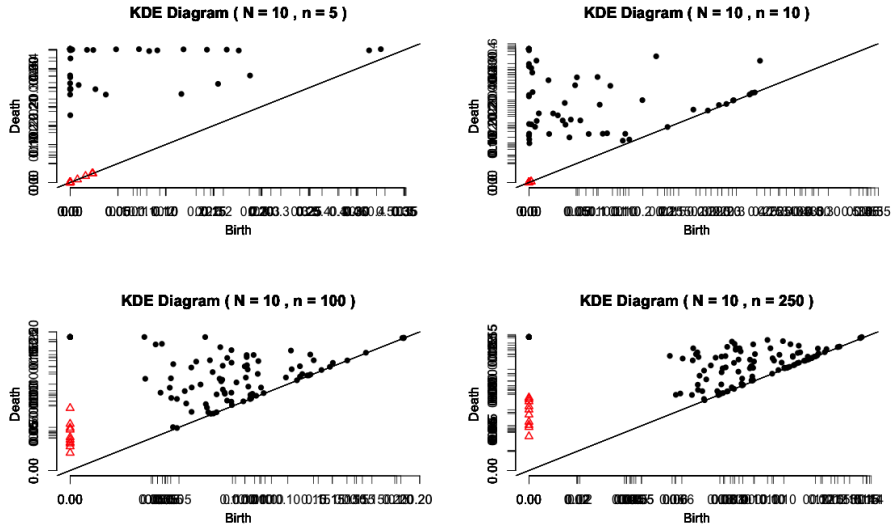


Figure 3: Plots of overlapping KDE diagrams. The number of simulations(N) is held constant but the sample size is varied (n).

Computationally, this method is less expensive than RIPS. The same setup of 10 simulations of 250 samples each was noticeably faster.

Kernel density estimates are usually used as a non-parametric way to find the probability density function (pdf) of some space. I'm assuming here that this function translates to a space that can be mapped onto a grid for which persistent homology can be applied. Qualitatively, as the number of samples grow, (1) the 1-dimensional complexes have longer lives, and (2), the 0-dimensional

complexes cluster together close to the linear birth-death line, and generally seem to be born later. **This doesn't use growing balls anymore correct? I have little intuition in how the algorithm works, so I'm sure why these patterns happen.** Why are the 0-dimensional complexes born later than the 1-dimensional ones?

4 DTM Diagram

Another possible algorithm to try is Distance to Measure. Let's repeat the process for this and see what happens. The settings are identical to the settings for KDE. Additionally, m_0 (the smoothing parameter for DTM aka for each point in the grid, when calculating distance, $m_0 * 100\%$ of all points are considered) is set to 0.1.

```
multiDTM <- function(N, Xlim, Ylim, by, m0=0.1) {
  for (i in 1:N) {
    Circle <- circle(numparticle, radius, center)
    Diag <- gridDiag(X = Circle, FUN = dtm, lim = cbind(Xlim, Ylim),
                     by = by, sublevel = FALSE, library = "Dionysus",
                     printProgress = FALSE, m0 = m0)
    par(new = TRUE)
    plot(Diag$diagram, main = "DTM Diagram")
  }
}
```

Usage of multiDTM is very comparable to multiKDE. The computational speed is about equivalent as well (being much faster than RIPS).

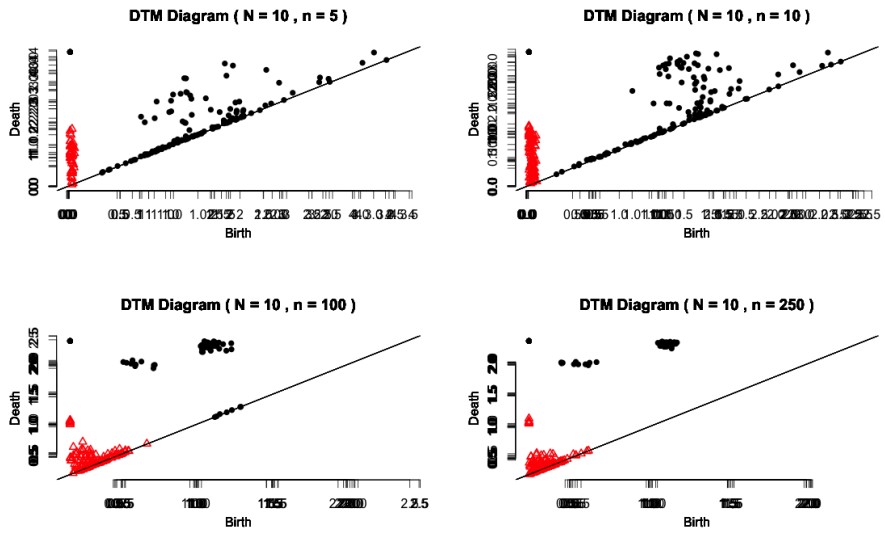


Figure 4: Plots of overlapping DTM diagrams. The number of simulations(N) is held constant but the sample size is varied (n).

Again, I lack intuition in the algorithm but qualitatively, as the number of samples increases per simulation, the 0-dimensional complexes form highly dense clusters with relatively late deaths, suggesting that there are stable point complexes. Additionally, it seems like with more samples, more 1-dimensional complexes are born and are killed early. Only a few red triangles survive (possibly representing the loop?). It is strange to me that again, the 1-dimensional complexes are found before the 0-dimensional ones (look at birth times).

5 Adding Noise

Real datasets are not this perfect. They are often muddled with noise and overlapping objects. In this small extension, let's pick a reasonably defined circle (n = 250), use RIPS, and perturb the matrix with differing levels of noise. To what extent will proper objects still be found? Persistent Homology seems to give all points in the point cloud equal weight. If that is true, will small amounts of additional points prevent us from finding the real object?

Adding Gaussian noise to the sample (x,y) points deteriorates the structure of the circle. Below are images of four different levels of noisy circles.

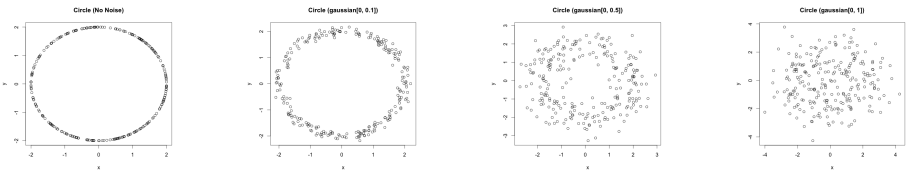


Figure 5: Plots of generated circles with added Gaussian ($\mu = 0$) noise.

We should expect that with additional noise, the persistent homologies should have a harder time finding true structure and may find more sub-voids than they should.
Notice that as more noise is added, fewer 1-dimensional complexes remain persistent and more 1-dimensional complexes are born and die almost immediately. Intuitively, with more noise, smaller voids appear immediately since the point cloud is dispersed more evenly within the circle. These small loops and voids attribute to the red triangles appearing on/nearby the line. As these balls grow, it is more likely that they combine to form one mass instead of creating a large loop. Therefore, with large noise (Gaussian(0, 1)), there are no 1-dimensional complexes that last longer than 2 units because the circle is filled up by small loops. However, in the original RIPS diagram (top left), there is no noise meaning that only m̈eaningfulöbjects appear.

A second manner of adding noise is permitting the original circle to maintain its perfect structure, but add additional random points. Will the original circle still be found at high levels of noise? Noise can come from a Gaussian distribution whose mean is the center of the circle and whose standard deviation is half of the radius.

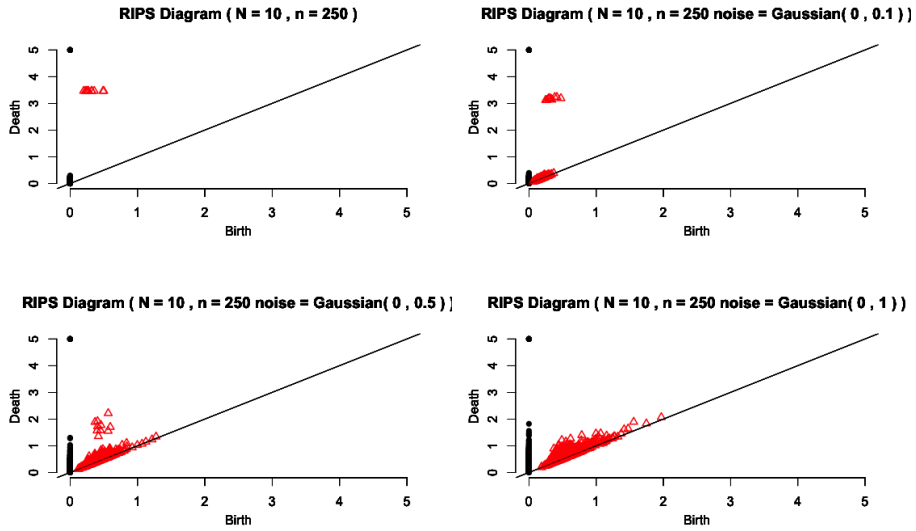


Figure 6: Plots of overlapping DTM diagrams. The number of simulations(N) and the sample size (n) is held constant but the noise added is varied (ϵ).

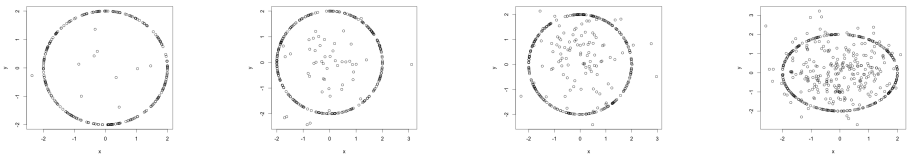


Figure 7: Plots of generated circles with added Gaussian points. From left to right, (10, 50, 100, 250) additional random points were added.

I expect this to perform better than the previous experiment for fewer points of noise because the true structure of the circle is still there. For high amounts of additional noise, I fear that the circle will be overwhelmed with small 1-dimensional complexes to a degree that the main loop will be ignored.

I think this has the same results as above. The same problem exists with the outer structure being largely ignored by smaller loops created from noisy data. **In general, persistent homology does not appear capable of handling sufficiently noisy data sets.** If we were to apply this to something as big and noisy as simulations of the universe, how can we mitigate the bad effects?

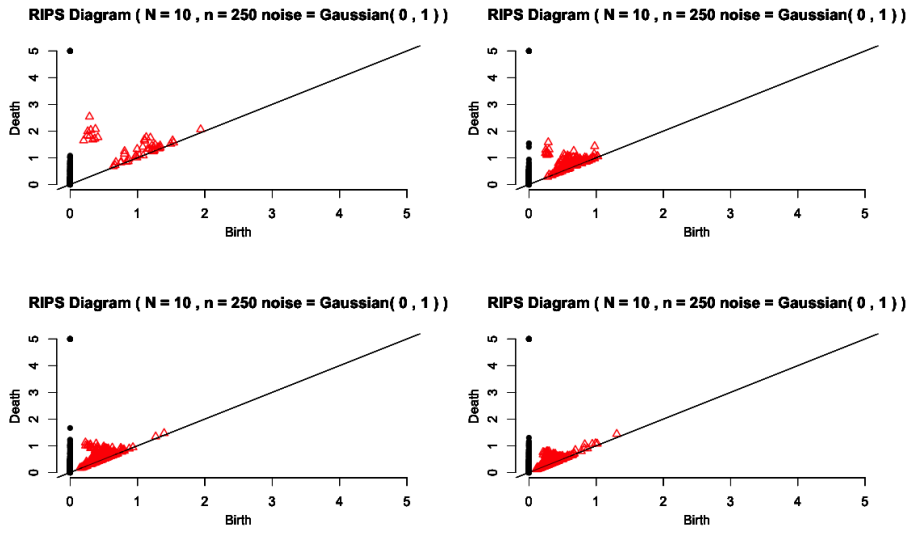


Figure 8: Plots of overlapping DTM diagrams. The number of simulations(N) and the sample size (n) is held constant but the additional noisy points added is varied. From left to right, (10, 50, 100, 250) additional points.