README

A "How To"

Fred Kaesmann, Dani Rosenberg and Kun Woo Lee

April 4, 2020

How to Handle our Package

Welcome to the intro to

```
library(FredsVietorisRips)
```

We lovingly designed this package for use in Dr. Aaron Clark's MAT 499 Topological Data Analysis Independent Study. [1] This package was developed to replicate aspects of the Mapper Algorithm from *insert sing, carlsson et al*.

It is to be used in the following manner

Usage

Downloading

To download from Github, copy and paste the following into the console

```
if (!require(devtools)) install.packages("devtools")
library(devtools)
if (!require(FredsVietorisRips)) install_github("ftkjr/FredsVietorisRips")
library(FredsVietorisRips)
```

Simplices, 0 and 1

To create our 1-simplices,

Step 1: We generate a data frame of random x and y coordinates

Step 2: We create a matrix of pairwise distances

Step 3: Given a distance, ϵ, we develop an adjacency matrix which displays a 1 for each pair within a ball of diameter ϵ

Step 4: From the adjacency matrix, we pull out the points which are adjacent. This is a list of all 1-simplexes in the dataset

Step 5: Last, we visualize the simplices using the 'ggplot2' library

[1] TDAIS for short

```r
##### Step 1: Create Data Frame ####
frame_size <- 12
df <- data.frame(
  x = runif(frame_size),
  y = runif(frame_size),
  Point = paste0("P", c(1:frame_size))
)

##### Step 2: Pairwise Distance Matrix ####

pwdmat <- Pairwisedist(df$x, df$y)
pwdmat
```

```
##            P1        P2         P3         P4        P5        P6         P7
## P1  0.0000000 0.6122899 0.64499376 0.68815743 0.9033324 0.4543600 1.02151843
## P2  0.6122899 0.0000000 0.42845118 0.51204649 0.4360194 0.2889200 0.51751566
## P3  0.6449938 0.4284512 0.00000000 0.08375718 0.3224701 0.6216750 0.44375419
## P4  0.6881574 0.5120465 0.08375718 0.00000000 0.3557719 0.7009101 0.47019564
## P5  0.9033324 0.4360194 0.32247015 0.35577190 0.0000000 0.7169774 0.12336036
## P6  0.4543600 0.2889200 0.62167496 0.70091014 0.7169774 0.0000000 0.80535911
## P7  1.0215184 0.5175157 0.44375419 0.47019564 0.1233604 0.8053591 0.00000000
## P8  1.1139657 0.6052362 0.52553583 0.54367140 0.2123941 0.8938865 0.09301246
## P9  0.8718391 0.3305675 0.73806880 0.81900917 0.6270446 0.4328164 0.65079780
## P10 0.6258529 0.1081262 0.53417030 0.61791218 0.5322190 0.2303141 0.60283952
## P11 1.0259800 0.6478527 0.38348273 0.36400436 0.2278638 0.9156611 0.24171255
## P12 0.0650603 0.5985234 0.67460810 0.72354661 0.9168782 0.4139155 1.03247071
##             P8        P9       P10       P11       P12
## P1  1.11396573 0.8718391 0.6258529 1.0259800 0.0650603
## P2  0.60523616 0.3305675 0.1081262 0.6478527 0.5985234
## P3  0.52553583 0.7380688 0.5341703 0.3834827 0.6746081
## P4  0.54367140 0.8190092 0.6179122 0.3640044 0.7235466
## P5  0.21239406 0.6270446 0.5322190 0.2278638 0.9168782
## P6  0.89388652 0.4328164 0.2303141 0.9156611 0.4139155
## P7  0.09301246 0.6507978 0.6028395 0.2417126 1.0324707
## P8  0.00000000 0.7126826 0.6864336 0.2576665 1.1253645
## P9  0.71268263 0.0000000 0.2584989 0.8546307 0.8398645
## P10 0.68643364 0.2584989 0.0000000 0.7496889 0.6010943
## P11 0.25766653 0.8546307 0.7496889 0.0000000 1.0515215
## P12 1.12536453 0.8398645 0.6010943 1.0515215 0.0000000
```

```r
##### Step 3: Given epsilon, determine Adjacency ####
# Given epsilon
epsilon <- 0.25

# Determine Adjacency
adjacency_matrix <- AdjacencyMatrix(pwdmat, epsilon)
adjacency_matrix
```

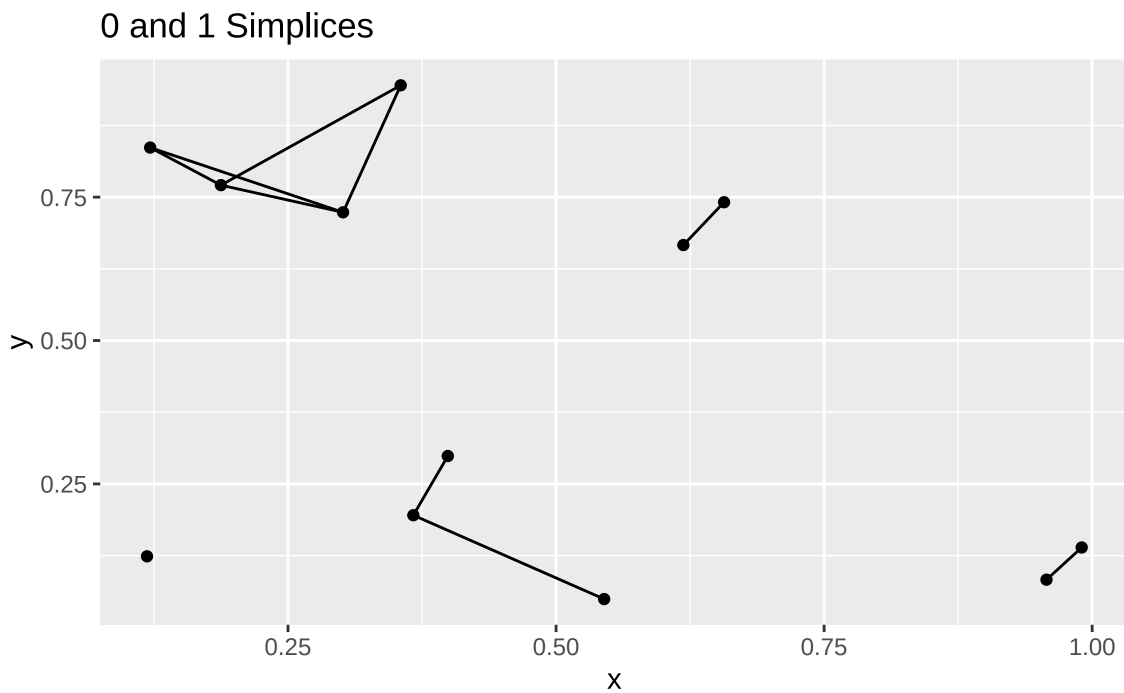
```
##     P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12
## P1   0  0  0  0  0  0  0  0  0   0   0   1
## P2   0  0  0  0  0  0  0  0  0   1   0   0
## P3   0  0  0  1  0  0  0  0  0   0   0   0
## P4   0  0  1  0  0  0  0  0  0   0   0   0
```

```
## P5   0  0  0  0  0  0  1  1  0   0   1   0
## P6   0  0  0  0  0  0  0  0  0   1   0   0
## P7   0  0  0  0  1  0  0  1  0   0   1   0
## P8   0  0  0  0  1  0  1  0  0   0   0   0
## P9   0  0  0  0  0  0  0  0  0   0   0   0
## P10  0  1  0  0  0  1  0  0  0   0   0   0
## P11  0  0  0  0  1  0  1  0  0   0   0   0
## P12  1  0  0  0  0  0  0  0  0   0   0   0
```

```
##### Step 4: Which Points are Adjacent? ####
paired_points <- AdjacentPairs(adjacency_matrix)
paired_points
```

```
##   Point_1 Point_2 Connection group
## 1      P3      P4          1     1
## 2      P5      P7          1     2
## 3      P5      P8          1     3
## 4      P7      P8          1     4
## 5      P2     P10          1     5
## 6      P6     P10          1     6
## 7      P5     P11          1     7
## 8      P7     P11          1     8
## 9      P1     P12          1     9
```

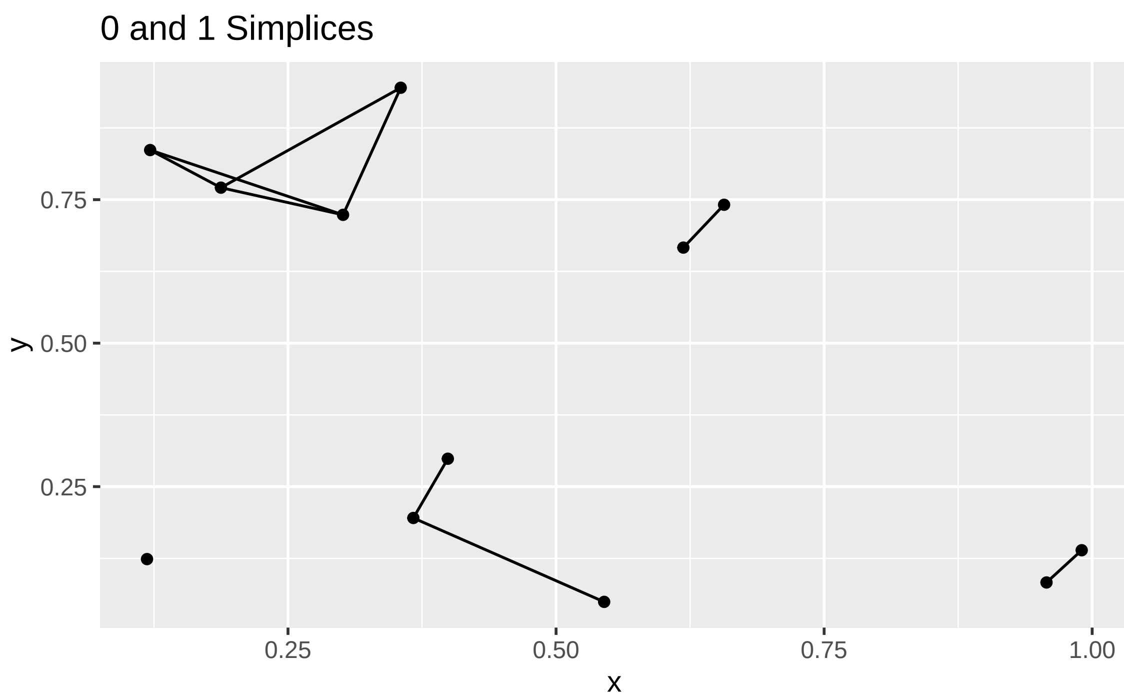
```
##### Step 5: Plot 0 and 1 Simplices ####
# Using the ggplot2 package
library(ggplot2)
paired_points %>%
  melt(measure.vars = c("Point_1", "Point_2")) %>%
  select(group, value) %>%
  rename(Point = value) %>%
  left_join(df, by = "Point") %>%
  ggplot() +
  geom_line(aes(x, y, group = group)) + # Plot the 1-simplexes
  geom_point(data = df, aes(x, y)) +    # Plot the 0-simplexes
  ggtitle("0 and 1 Simplices")
```



Do all

We have provided a "do all" function, if provided an x and y vector, with a distance ϵ it is generous enough to spit out a graph of the associated 1-simplices under the Vietoris-Rips arrangement.

```
Plot_Simplices(df$x, df$y, epsilon)
```



This may be most useful for iterating through various ϵ values to determine [2] which of various graphs best describes the underlying data.

[2] Subjectively