Implementation

```
##### Packages ####
library(FredsVietorisRips)  # Our Very Own
library(magrittr)
library(dplyr)
library(ggplot2)
library(gridExtra)
library(reticulate)
```

FredsDBSCAN

The following code is the code behind `FredsDBSCAN`. This iterates through our data set, detecting connected components based on some distance, ϵ and some minimum number of connections.

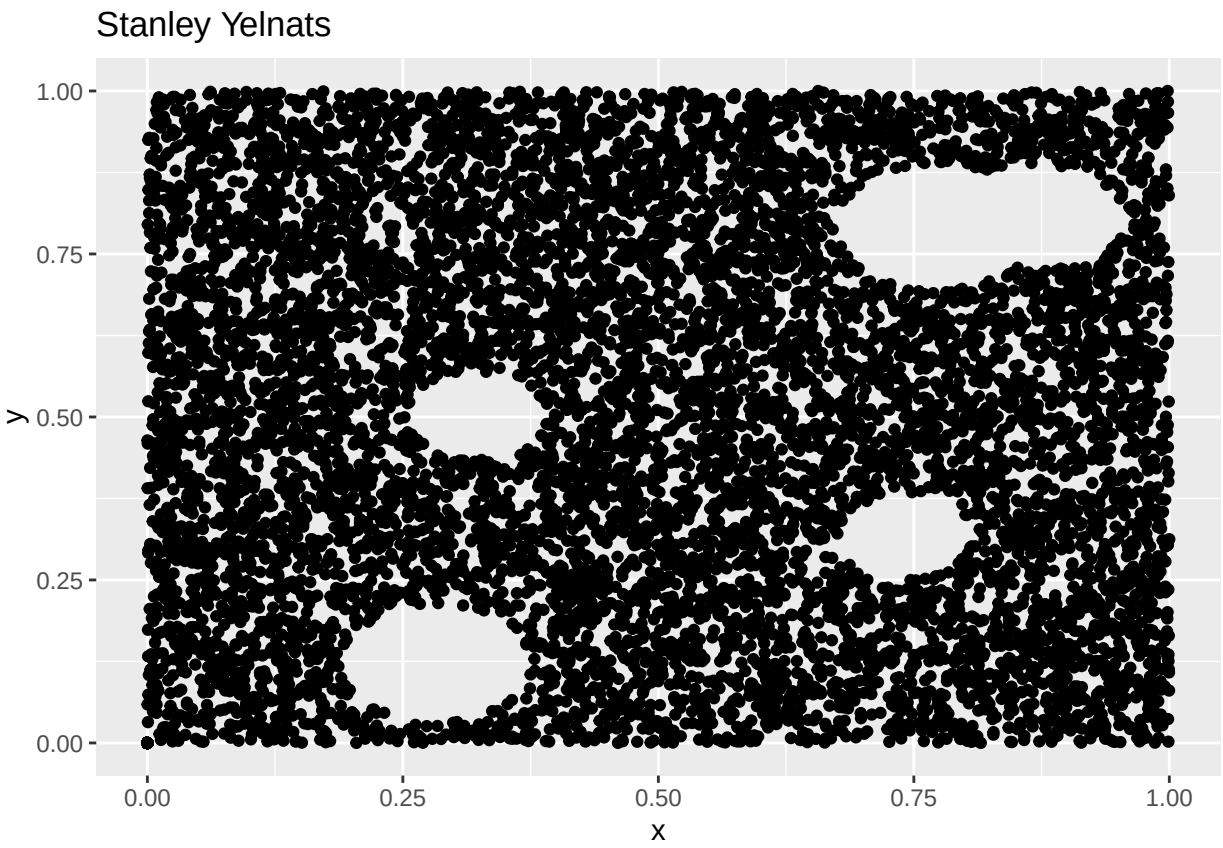
```
def find_starting_point(adjacency_matrix, untraversed_points, minimum_connections):
    for starting_point in untraversed_points:
        if sum(adjacency_matrix[starting_point]) >= minimum_connections:
            untraversed_points.remove(starting_point)

            return starting_point, untraversed_points

    else:

        return None, untraversed_points

def get_cluster(adjacency_matrix, minimum_connections, untraversed_points):
    ###############
    # Initialize #
    ###############
    cluster = []                                # List to store clustered points

    ########################
    # Find the first point #
    ########################
    # Start cluster with first point that meets cluster requirements
    starting_point, untraversed_points = find_starting_point(adjacency_matrix, untraversed_points, minim

    # If we can't find a starting point, return None
    if starting_point is None:
        return None, untraversed_points
    else:
        #   otherwise append the starting point and begin search
        cluster.append(starting_point)

    ################################################
    # Begin the search for points in this cluster #
```

```
    ################################################
    for point in cluster:                                                   # For points in our clust
        for other_point in untraversed_points:                              #   Search through the po
            if adjacency_matrix[point, other_point] == 1:                   #       if they are adjac
                untraversed_points.remove(other_point)                      #           remove them f
                if sum(adjacency_matrix[other_point]) >= minimum_connections:     #           if they m
                    cluster.append(other_point)                             #               add it to

    ##########
    # Return #
    ##########
    return cluster, untraversed_points
    # Fin



def FredsDBSCAN(adjacency_matrix, minimum_connections):

    ###############
    # Initialize #
    ###############
    cluster_list = []                                                       # List of Clusters
    cluster = []                                                            # Holds each Cluster as we
    untraversed_points = [p for p in range(len(adjacency_matrix))]          # List of points to hit

    ##################################
    # Traverse our Adjacency Matrix #
    ##################################

    while len(untraversed_points) >= minimum_connections:                   # While there's a possibil
        cluster, untraversed_points = get_cluster(adjacency_matrix,
                                                  minimum_connections,
                                                  untraversed_points)       #     find clusters and gi
        if cluster is None:                                                 #     if we didn't find an
            break                                                           #         STAHP
        else:                                                               #     assuming we found a
            cluster_list.append(cluster)                                    #         add it to our li

    return cluster_list
```

```
##### Source our Python Code for Later Use ####
# All the code
filelist <- list.files("../python/")
for (f in filelist) {
  source_python(paste0("../python/", f))
}
```

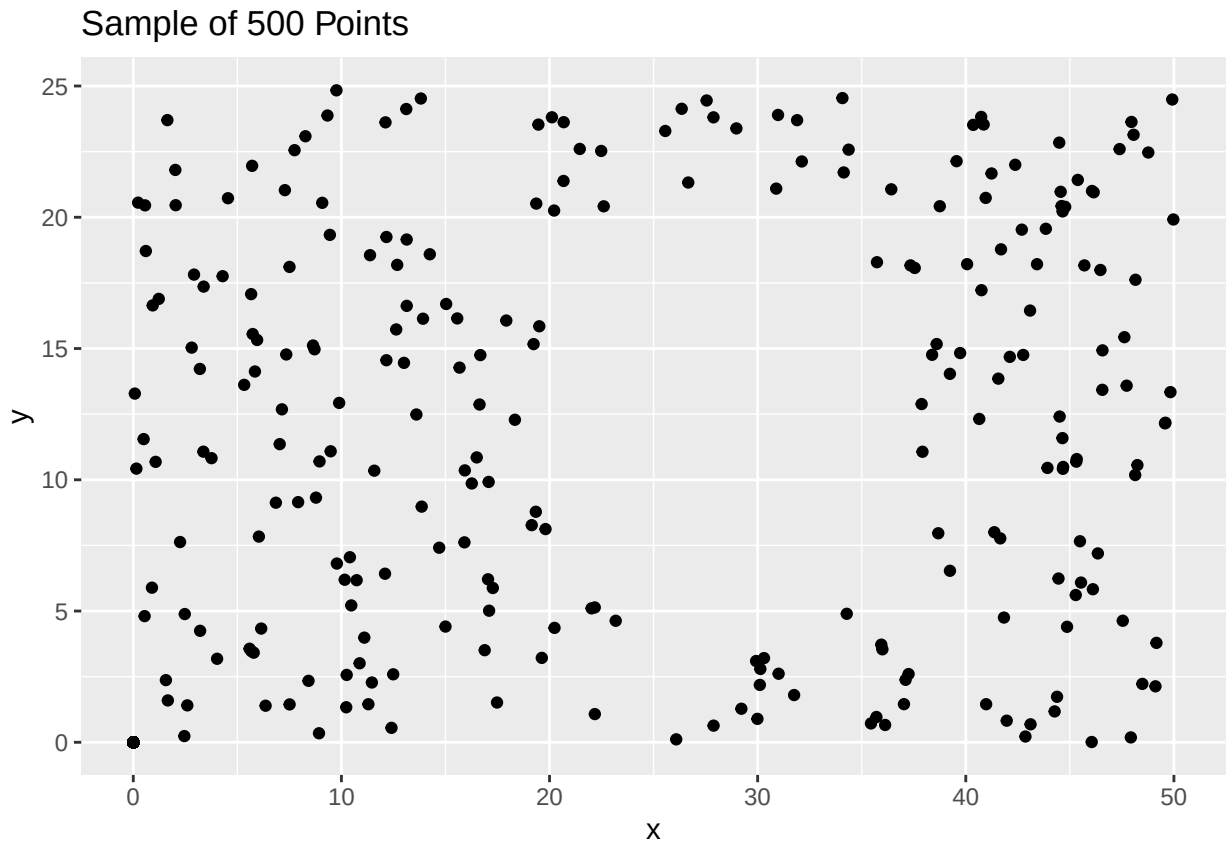
```
##### Import Data ####
df_original <- read.csv("../data/Clark_Sample_data.csv", col.names = c("x", "y"))

##### Visualize Data ####
df_original %>%
  ggplot(aes(x, y)) +
```

```
  geom_point() +
  ggtitle("Stanley Yelnats")
```



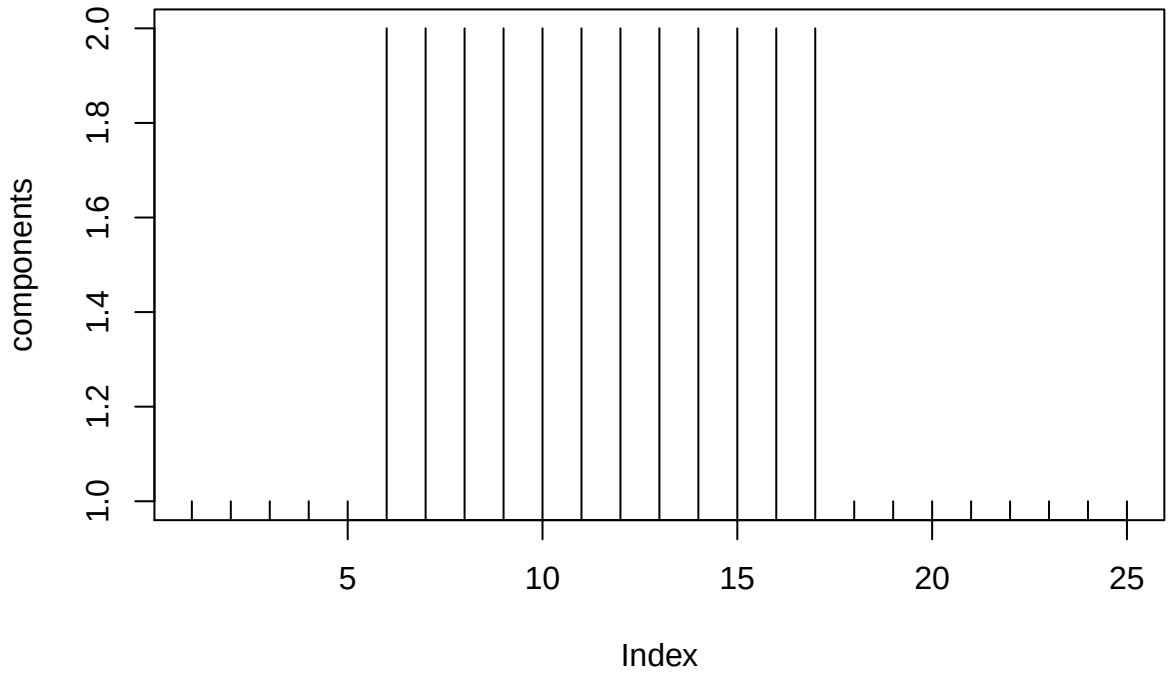
We take a section from our data set and pull a sample of 500 points.

```
##### Take a section of the original data ####
# blow it up
df <- df_original[df_original$x < 0.5 & df_original$y < 0.25, ] * 100
df <- sample_n(df, 500)

##### Let's seee what we're working with ####
df %>%
  ggplot(aes(x, y)) +
  geom_point() +
  ggtitle("Sample of 500 Points")
```

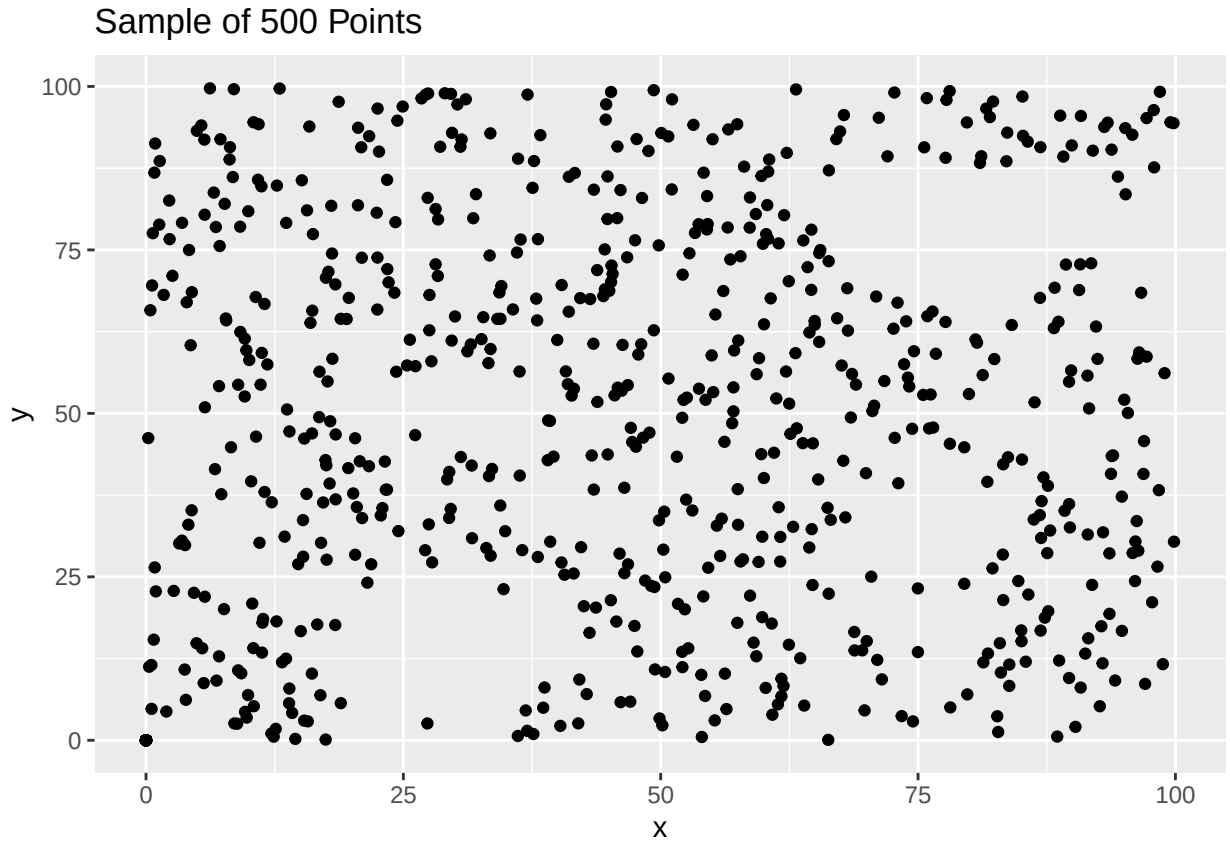



```
##### Filter ####
list_of_covers <- PullBackCovers(df$x, df$y, 4, 0.75)
```

```
##### Choose Epsilon and Minimum Connections ####
epsilon <- 15
min_connections <- 3

##### How many components do we have? #####
components <- vector(length = length(list_of_covers))
for (cover in c(1:length(list_of_covers))) {
  components[cover] <- Pairwisedist(list_of_covers[[cover]]$x, list_of_covers[[cover]]$y) %>%
    AdjacencyMatrix(epsilon) %>%
    FredsDBSCAN(min_connections) %>%
    length()
}

##### Simple Plot ####
plot(components, type = "h")
```




```
##### Sample Entire Set ####
df <- df_original * 100
df <- sample_n(df, 750)

##### What does it look like? ####
samplechart <- df %>%
  ggplot(aes(x, y)) +
  geom_point() +
  ggtitle("Sample of 500 Points")
print(samplechart)
```




```
##### List of Pullback Covers ####
list_of_covers <- PullBackCovers(df$x, df$y, 4, 0.75)

##### Choose Epsilon and Minimum Connections ####
epsilon <- 15
min_connections <- 3


components <- vector(length = length(list_of_covers))
for (cover in c(1:length(list_of_covers))) {
  components[cover] <- Pairwisedist(list_of_covers[[cover]]$x, list_of_covers[[cover]]$y) %>%
    AdjacencyMatrix(epsilon) %>%
    FredsDBSCAN(min_connections) %>%
    length()
}

compframe <- data.frame(
  index = c(1:length(list_of_covers)),
  components = components
)

compchart <- compframe %>%
  ggplot(aes(x = index,
             y = components)) +
  geom_col() +
  coord_flip() +
```

```
  ggtitle("Components by Pullback Cover")

grid.arrange(compchart, samplechart, nrow = 1)
```