

Jesse Anderson Project4 Documentation

Dependencies:

Tkinter

Numpy

Matplotlib

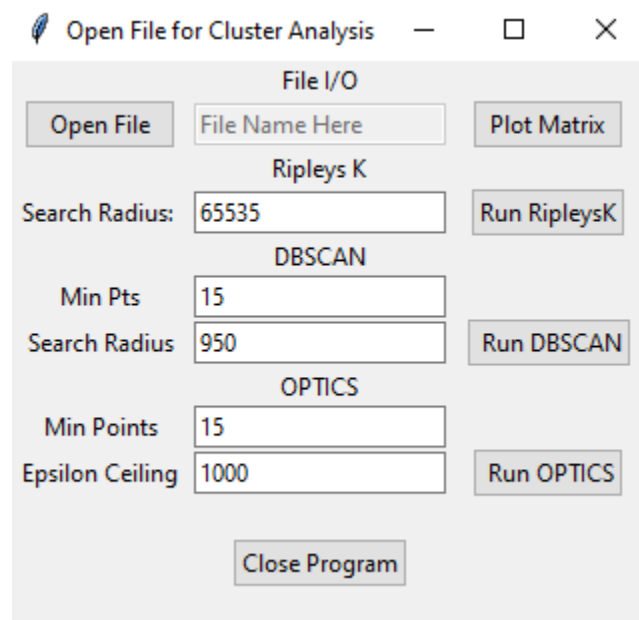
Astropy

sklearn.cluster/sklearn.neighbors

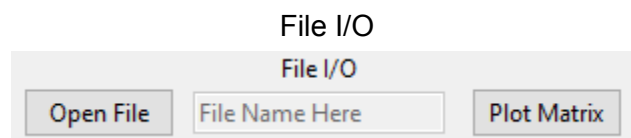
Clustering Data sets taken from: <http://cs.joensuu.fi/sipu/datasets/>

This program takes as input an XY matrix file and outputs either a simple plot of the original data, a Ripley's K/L graph with the domain size in title, a DBSCAN based clustering of the XY points, or an OPTICS based clustering of the XY points.

Below you will see an image of the GUI as of the first version:



An overview of each set of functions will be described below:



Open File: This button creates a file dialog box where a user can select a file to be opened. The file name and its path will be shown in the 'File Name Here' box to the immediate right. The default file type is a .txt file however one can click all files and all file types will be selectable(Not Recommended).

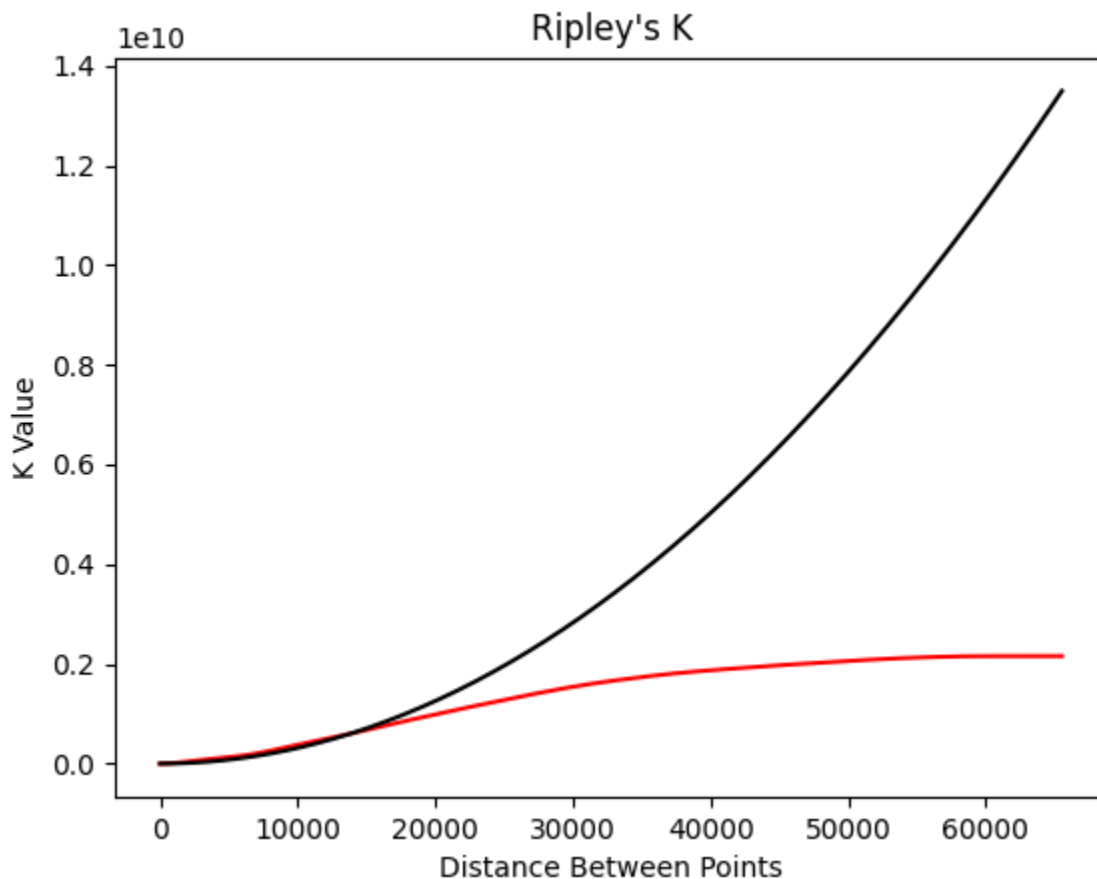
Plot Matrix: This button plots the XY coordinates using matplotlib and serves as a way to verify that the user selected the correct file. An error message will be thrown if the user selects a file which is not an XY file as defined by the number of columns the file has. Another error will be generated if the file exceeds 1 megabyte as the current implementation of Ripley's K in AstroPy is not optimized for large matrices. **If the file has 2 rows, but many columns then please transpose your matrix file.**

Ripley's K

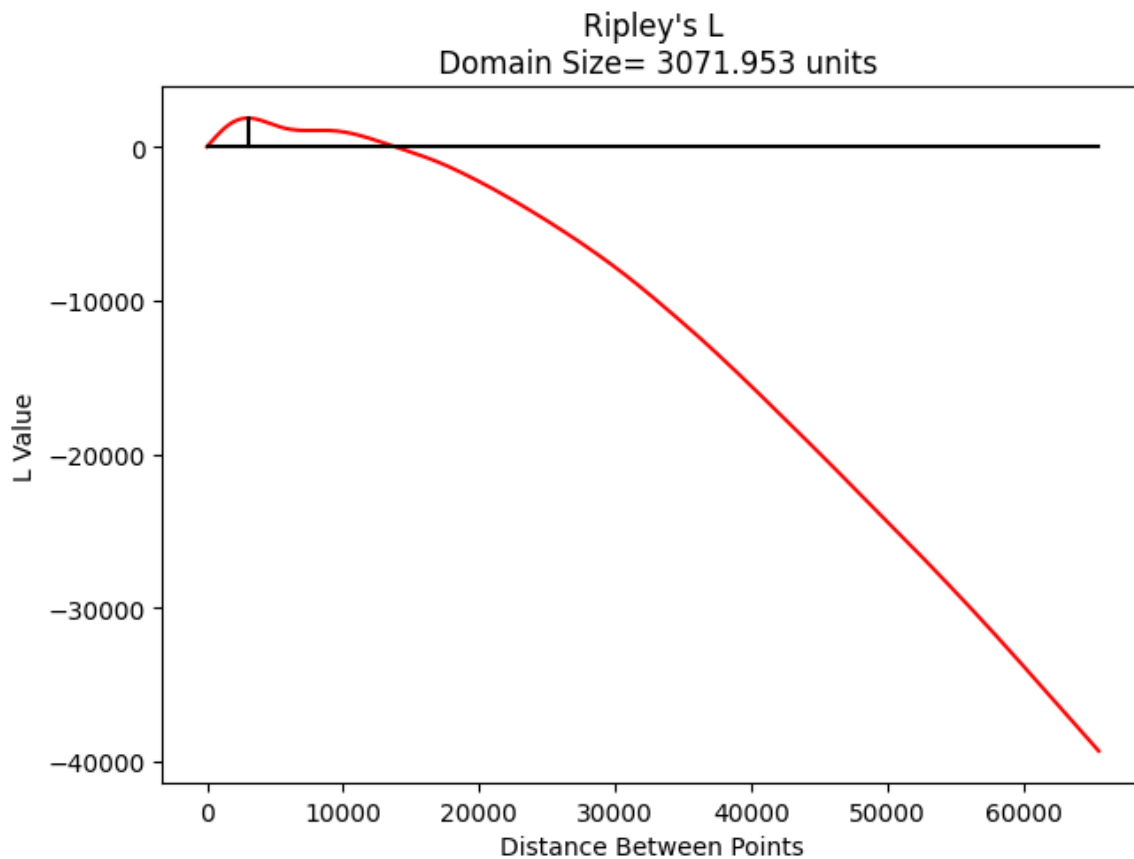
Ripleys K

Search Radius:

Ripley's K is a generalized overview of the degree of clustering present in a data set. It searches at 513 evenly spaced intervals from 0 to the inputted Search Radius and finds the degree of clustering present at each of those intervals. From here this degree of clustering is plotted against a Completely Spatially Random(CSR) distribution at those same evenly spaced intervals. If the points lay above the CSR then it is said to be clustered, below then it is not. That is the Ripley's K Graph pictured below:



For the second graph Ripley's L is calculated by taking Ripley's K values divided by pi and square rooting that value. After that calculation is performed each radius value is subtracted from each calculated value to give Ripley's L. The importance of Ripley's L is that the CSR distribution roughly goes to 0 so it gives a much better visual representation of clustering and the greatest degree of clustering which occurs. Additionally this program calculates that greatest degree of clustering and finds the 'Domain Size' which is another name for the greatest degree of clustering and is used in: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2726315/> This number is found as the maximum value of Ripley's L and a horizontal line is plotted from that number down to CSR to create a clear visual per the below graph:



Search Radius: Input Parameter that is the maximum distance to search in the Ripley's K algorithm. It is best left to the maximum value of the X/Y points then scaled back until it is roughly 10% above where Ripley's K crosses the CSR to create a better visual.

Run Ripley's K: Runs the Ripley's K program, checks to make sure an appropriate matrix file is opened and saves the Ripley's K data and graphs to Folders called '/Data' and '/Figures' respectively.

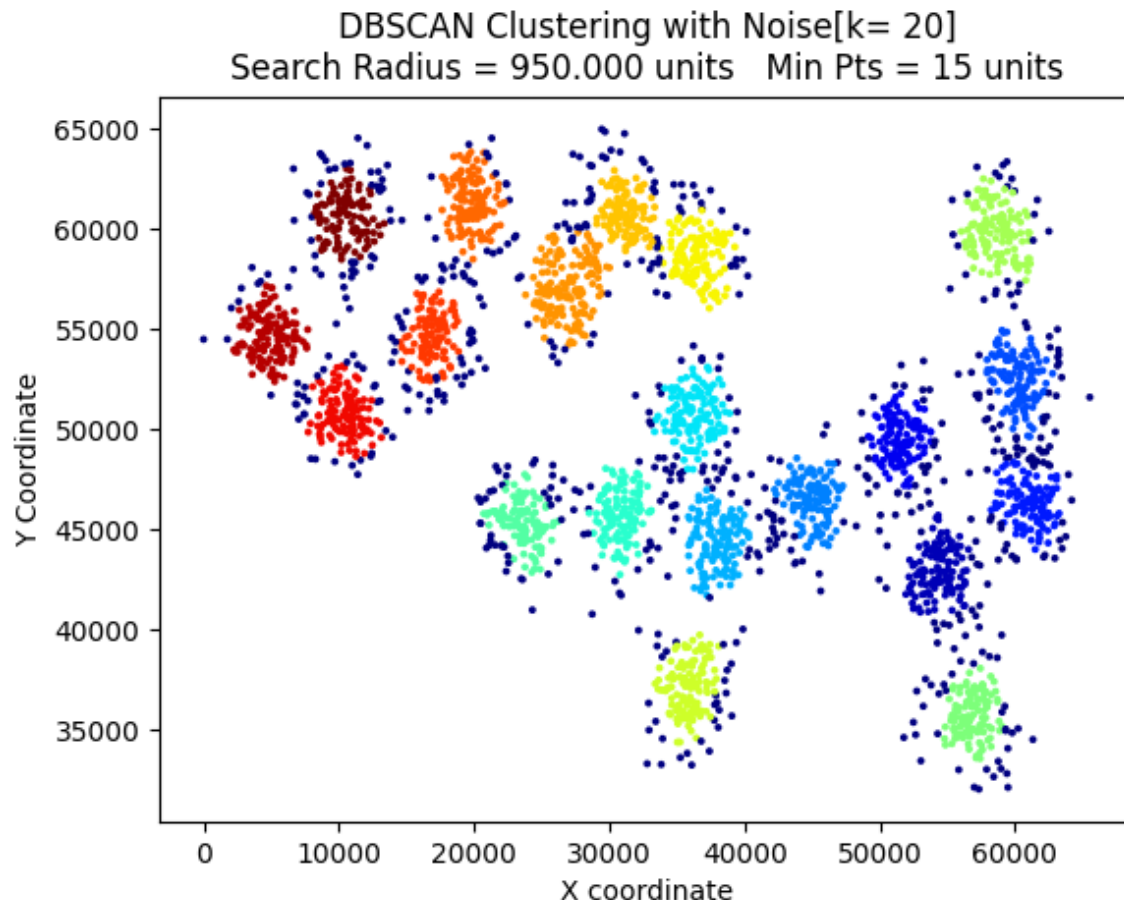
****Please note that for very very large matrices(200,000 points) R is better suited to run a Ripley's K analysis. The resulting array from Ripley's K in Python using the AstroPy library will be ~ 500.0 Gb+ in size. ****

DBSCAN

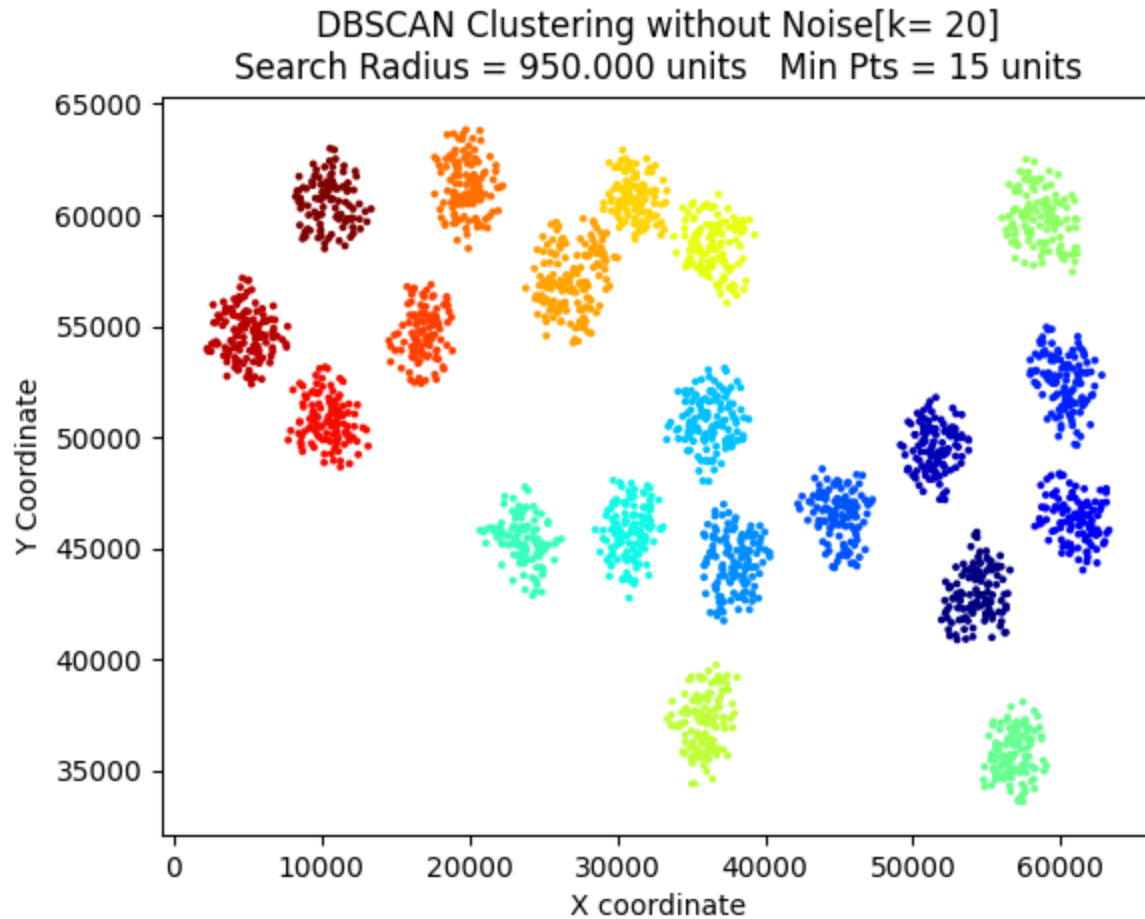
DBSCAN	
Min Pts:	<input type="text" value="15"/>
Search Radius:	<input type="text" value="950"/>
<input type="button" value="Run DBSCAN"/>	

DBSCAN or Density-Based Spatial Clustering of Applications with Noise, is a clustering routine that clusters XY points based on a search radius in which a circle is drawn around every point and if the number of points within that circle is at or equal to the value of a minimum number of points then it will belong to a cluster. A noise point is defined as a point that does not have the minimum number of points around it and thus does not belong to a cluster.

First the user selects the minimum number of points to define a cluster and the radius to search for the clusters. From here Run DBSCAN button is pressed and the clusters are found. Plots are created where the first is the DBSCAN with noise and all input parameters included in the graph are the number of clusters is given as $k = \text{'clusters'}$:



The second plot is the same exact data but with all noise filtered out:



Additionally another small window is created which lists some extra Data and can be customized by the user if they so choose by adding more data from the global variable **clusterMe** which contains all the DBSCAN data.

```
{Number of Clusters is: 20}
{Number of Noise points is: 609}
```

For an amazing interactive visualization of DBSCAN I recommend looking at this site by Naftali Harris: <https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/>

Min Pts: Specifies the minimum number of points to define a cluster.

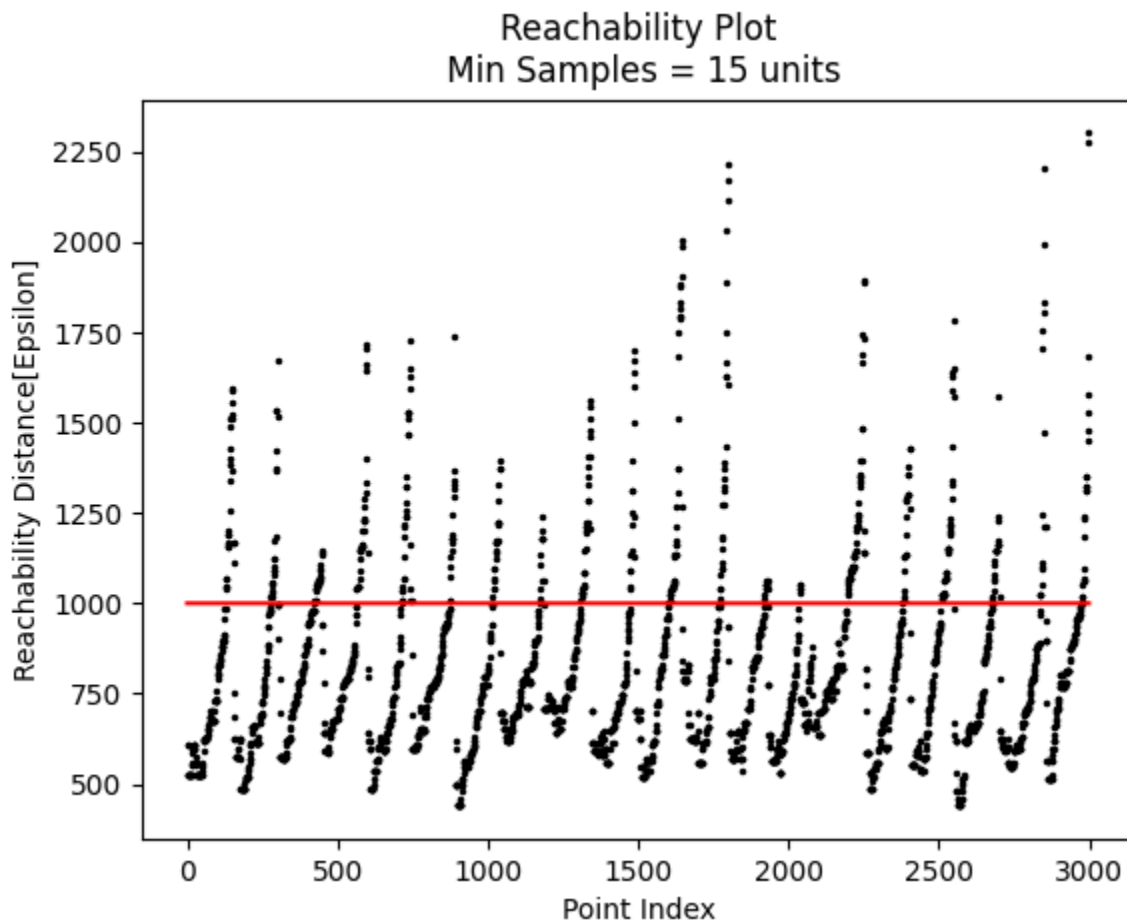
Search Radius: Specifies the search radius to scan at each individual point to look for the minimum number of points. If the minimum number of points is not met at a given point then that point will belong to 'Noise'.

Run DBSCAN: Runs DBSCAN, checks if an appropriate matrix has been opened, and runs the calculations and plots the Data. All Data is saved to '/Data' and all figures are saved to '/Figures',

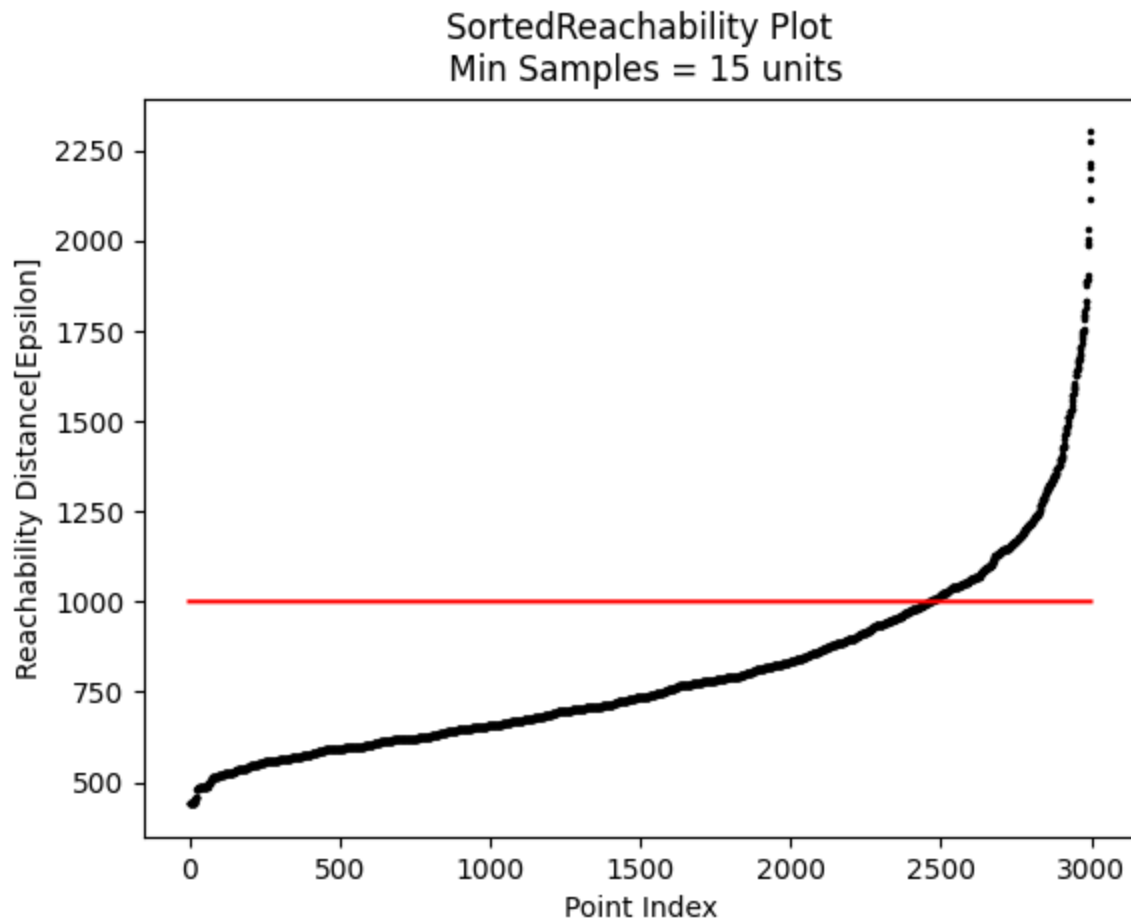
OPTICS

OPTICS	
Min Points:	<input type="text" value="15"/>
Epsilon Ceiling:	<input type="text" value="1000"/>
<input type="button" value="Run OPTICS"/>	

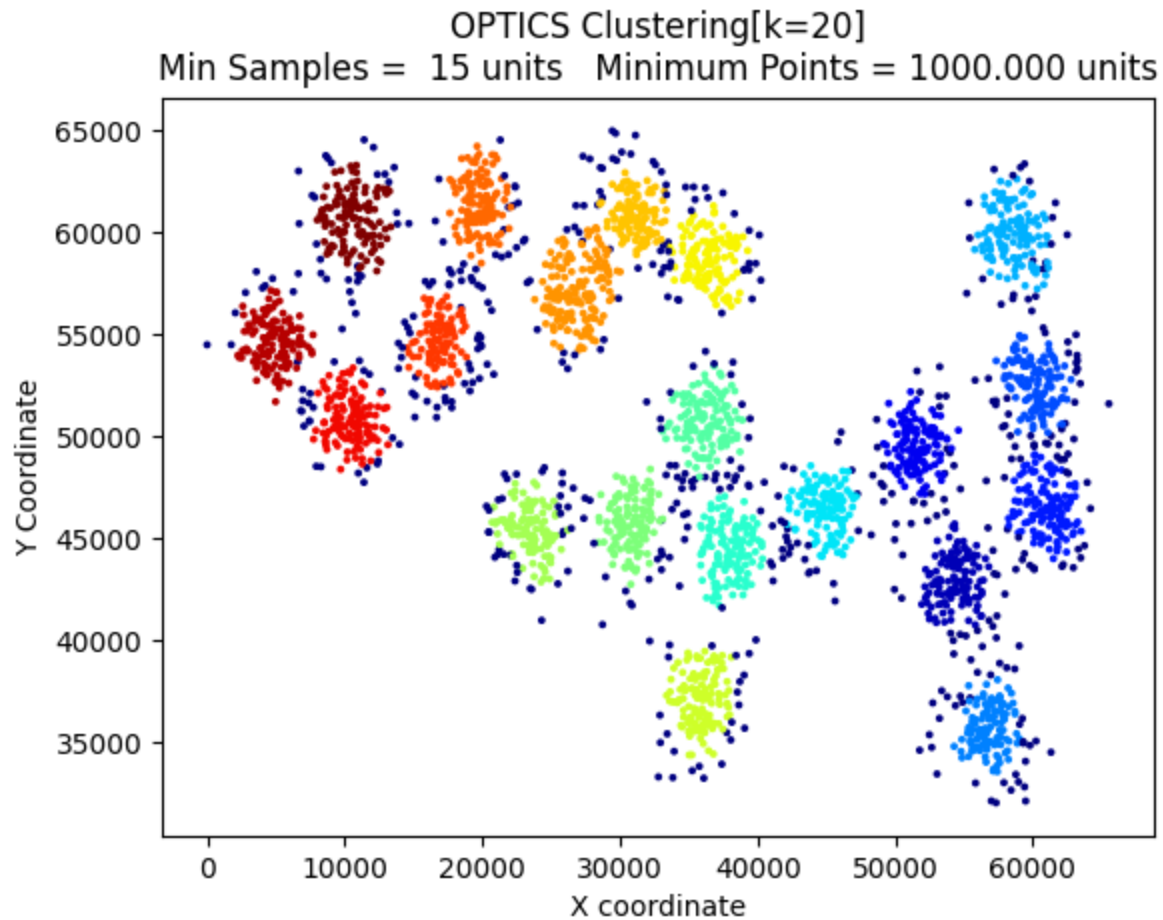
OPTICS or Ordering Points To Identify the Clustering Structure, is a clustering routine that clusters XY points based on their reachability distance. The program first looks at how far away each points are from each other at a given index. In the below example you see that the X values correspond to the index of the clusters, that is the first point, irrespective of its value, is assigned 1 all the way up to 3000. The Y values correspond to the distance the point is from all the other points, a higher value corresponds to a far lower chance of belonging to a cluster. The horizontal line drawn across the graph is the ceiling value for epsilon. Indices below this value correspond to clusters while indices above it correspond to noise. One adjusts this value based on the graph to better define clusters.



Additionally a sorted Reachability plot is generated which sorts the points from lowest to highest reachability distance. In general one can interpret this graph as choosing an epsilon such that it is roughly around the 'elbow' of the graph. In the case below the Data isn't very clearly clustered so there is a well defined elbow, but the best guess would be around ~1250, but that gives an erroneous clustering value so in the case of our '20clusters.txt' file this is not a useful graph.



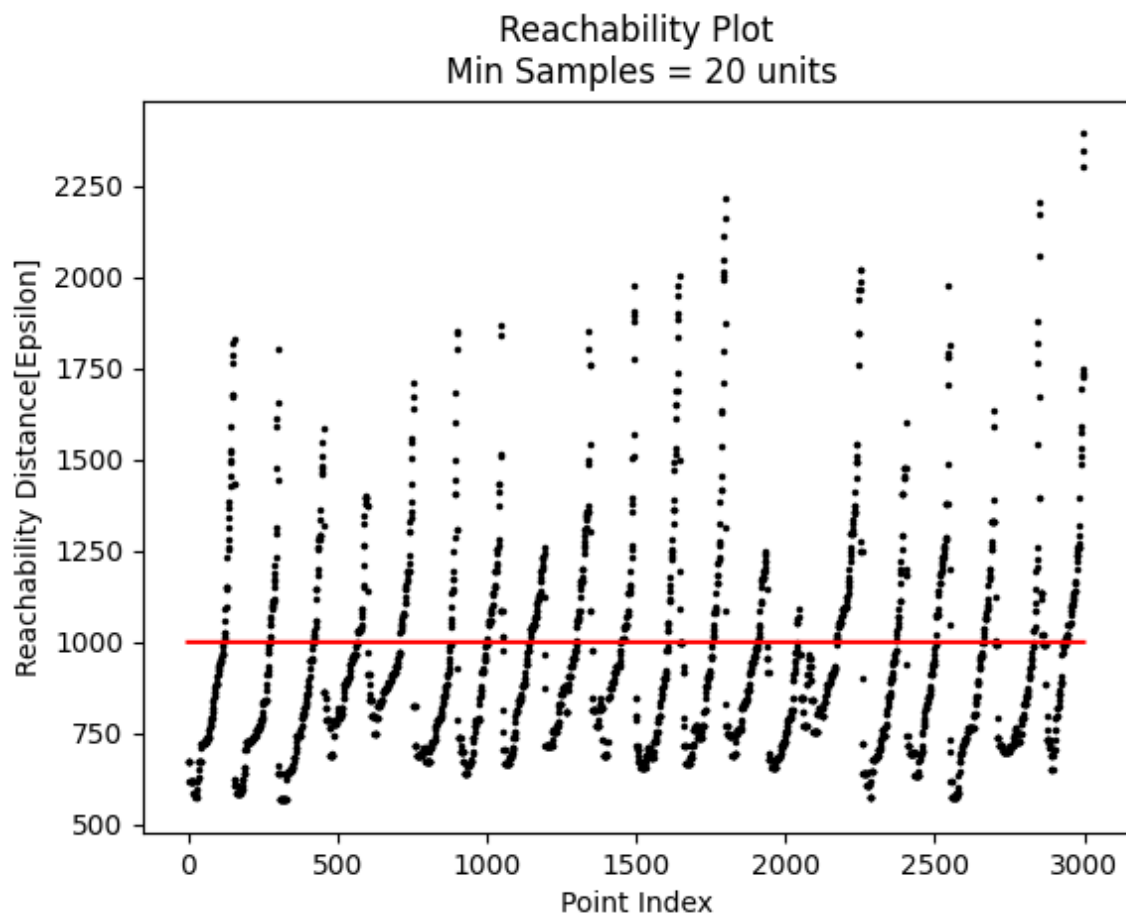
After one has chosen a suitable epsilon ceiling based on the shape of the reachability plot as defined by the minimum number of points a DBSCAN-like clustering scatterplot is produced as below:



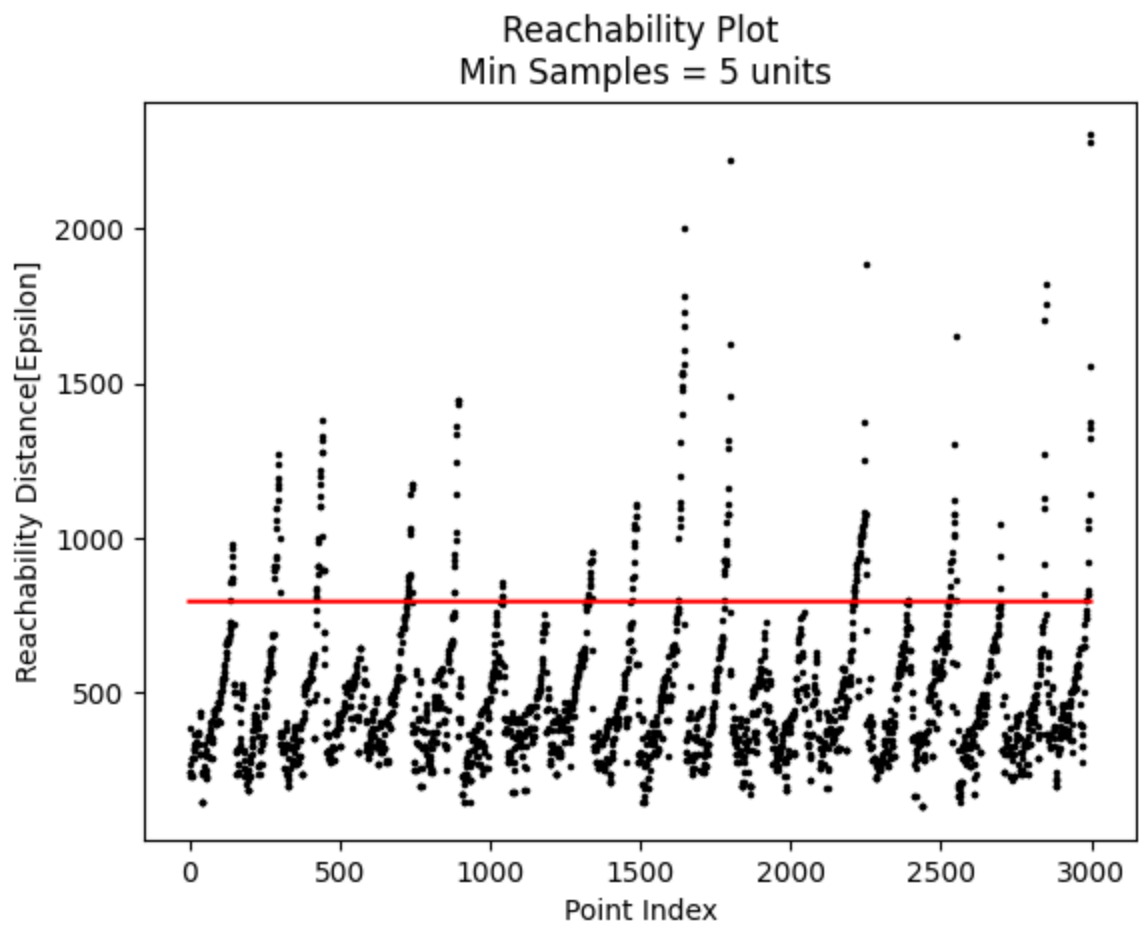
The noise points are just outside the clusters as expected and each cluster is color coded. The k value corresponds to the total number of clusters.

Min Points: The minimum number of points to define a cluster. This value is used to generate the reachability plot and different values will create different reachability plots. It is best to select several values for Min Pts until one finds a reachability plot with a high degree of peaks/valleys which are clearly defined.

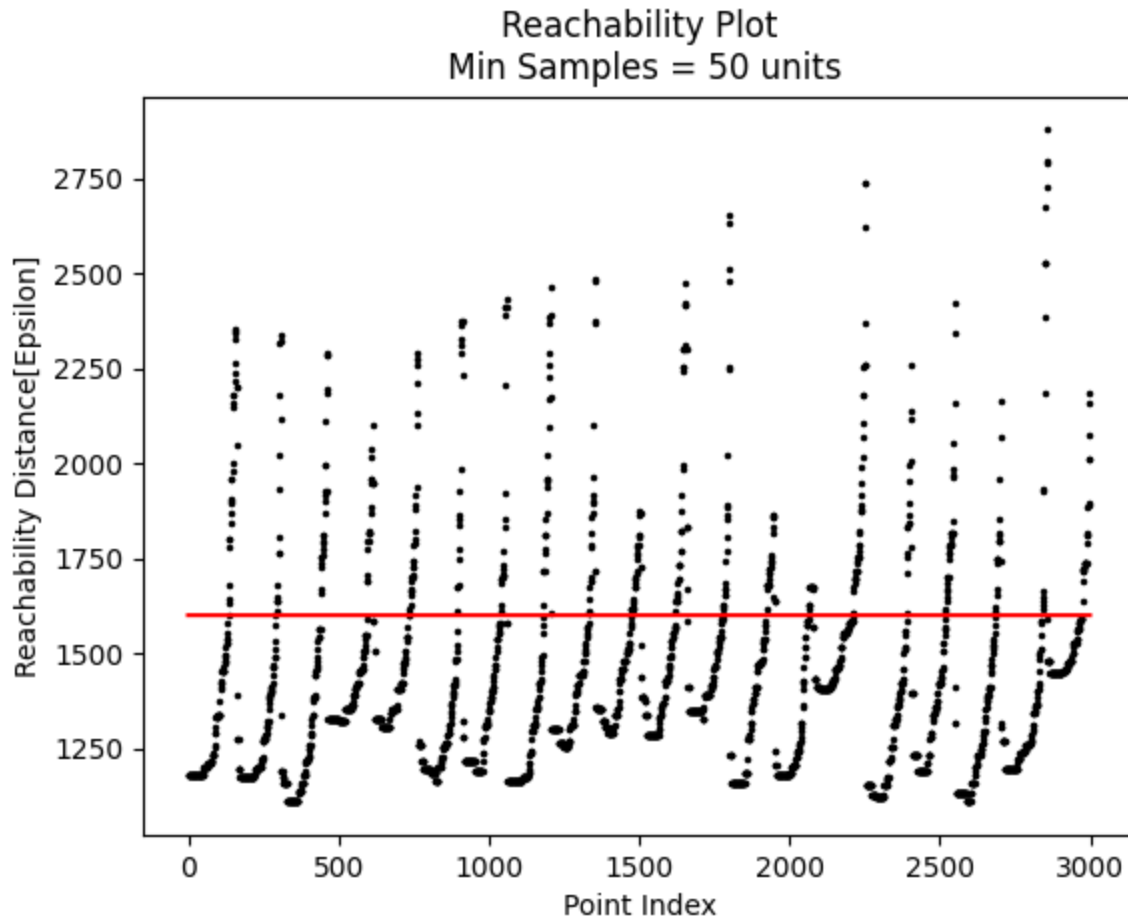
Epsilon Ceiling: The ceiling to search in OPTICS. In general select an epsilon ceiling that goes through every single peak. Every peak corresponds to a cluster and a good reachability plot generated from a good Minimum Points value is key to getting a good final clustering result plot. Below I will demonstrate some different Reachability Plots and the clusters found.



Above: 20 clusters found



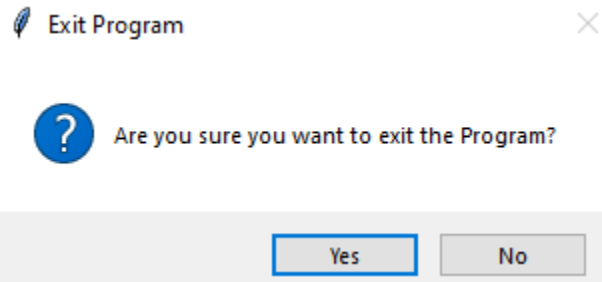
Above: 20 clusters found




Above: 20 clusters found.

Close

This button will prompt a dialog box asking the user if they really want to close the program.



If the user chooses yes then all tkinter windows will be closed along with all plots. If the user selects no, then a box will pop up stating that they will be returned back to the program:

 Return



Returning to the Program

OK