CS251 Proj7

CS251 Project 7 -Clustering Analysis

The main goal of this project was to implement a Clustering Analysis a data set and producing the result visually. The first part of the project required us to construct a K-Means Mathematical Classes within our analysis.py class. These were called *kmeans_numpy, kmeans_init, kmeans_classify, kmeans_algorithm,* and *kmeans.* Then, the bulk of this project was to then also incorporated all of these capabilities into our GUI, including clustering with color!

GUI Implementation:

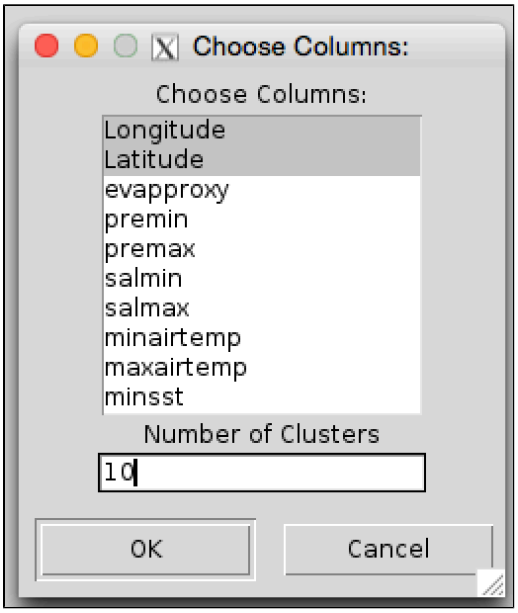
So, after I completed all the mathematical K-means methods in my analysis file, the first thing I focused on was constructing a Clustering Dialog Box for my Display. Next, I created a handleCluster() method in my display class that basically adds a "Cluster1" option to my dialog box when I choose to "Plot Points" that will display the result of completing a K-Means cluster of a given number on the given data. So, to run a clustering analysis in my GUI: Open a File, Choose the "Analysis" dropdown option and select "Clustering Analysis", and select your desired headers and give the list box a "Number of Clusters." Then, click the "Plot Points" on the right hand side of my program and then select X-axis, Y-axis, and Z-axis while looking for a new Header called "Clusters 1." Make sure that one is selected and then plot the points! The data should now be plotted and clustered in 3-D space with the Clusters sorted on the Z-axis!

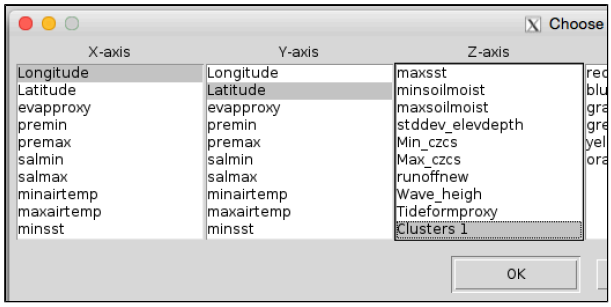
Next, the final part of the project was to help the user visualize these clusters by giving each cluster a random color. To do this, I simply made a list of a lot of random colors in my display *_init_*. Then, with some help from Bruce, I created a method called color fixed that simply assigned a random color from that list to each cluster set. Here is the code for colorFix():

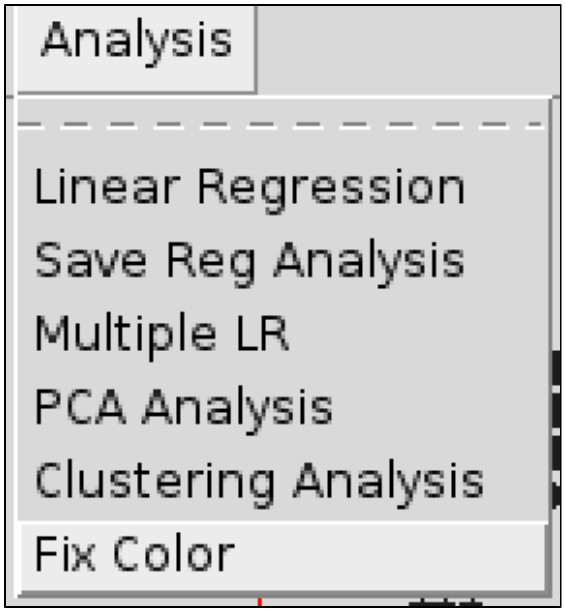
```
#HELP FROM BRUCE!!
def colorFix(self):
#figure out a way to specify a column to base the color of off
    colorColumn = self.data.get_data( ['Clusters 1'] )
    # if the data in colorColumn has less than 30 unique values
    unique = np.unique(np.asarray(colorColumn))
    unique = unique.tolist()
    if len(unique) <= 30:
        # for each visual object that has been plotted
        for i, point in enumerate(self.objects):
            # color index is unique.index( colorColumn[i,0] )
            colorIdx = unique.index(colorColumn[i,0])
            # set the color to the color string at that index
            self.canvas.itemconfigure(point, fill=str(self.colors[colorIdx]))
```

To call colorFix() on the data set after clustering, simply select the "Analysis" dropdown and then click "Fix Color"

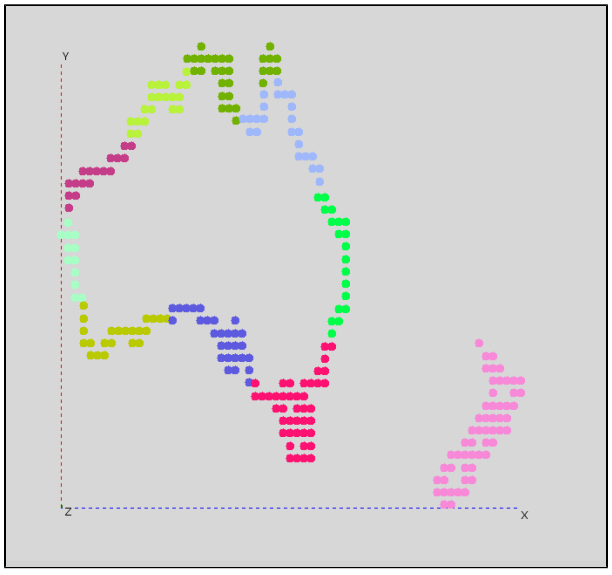
How to Cluster in A.N.A.L.Y.Z.:

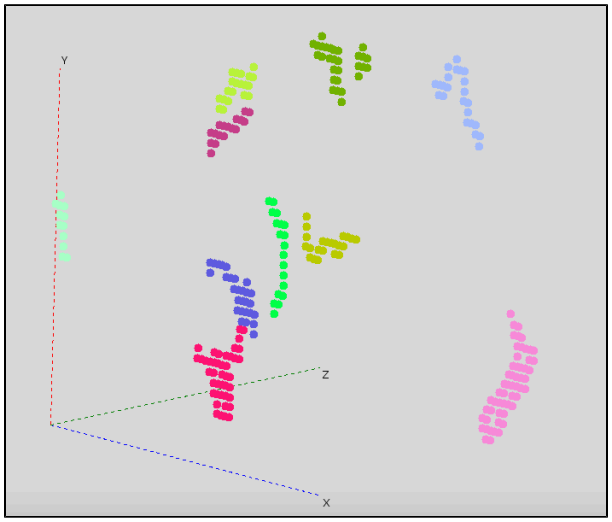






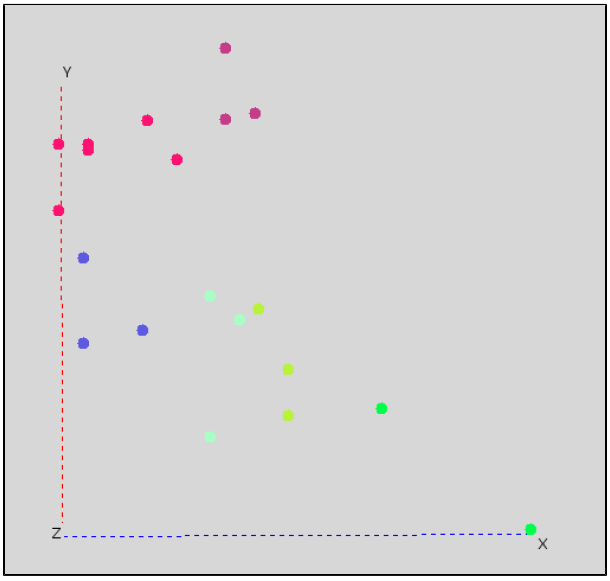
Australian Coast Screenshots:

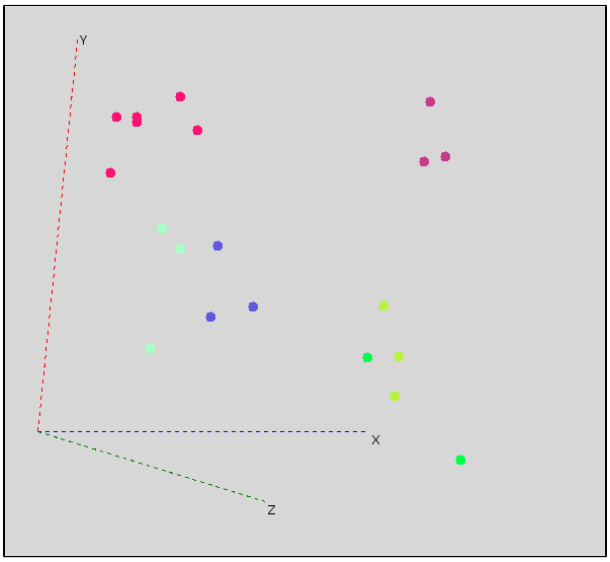




MLB Pitching Data Screenshots:

For my random data set cluster test, I used MLB Pitching data on Wins, Loses, ERAs(earned runs against), and other Baseball Pitching related statistics. Below, for my clustering test, I placed Wins on the X-axis, ERA on the Y-axis, and the Clusters on the Z-axis. You can clearly see the expected trend, the smaller/lower the ERA average, the more Wins. Basically, saying less runs against = more wins! You can see the data nicely clusters in the 6 clear clusters.





Note To Grader!:

Hello, I hope you're enjoying my project. I finished the coding work for Project 7 over a week ago, but badly dislocated my right shoulder last week. Due to this injury, I have had trouble typing and that is why this project is so late. I am very sorry for making you wait to grade this, but I had a medical emergency and talked to Stephanie and Bruce :) Now that I am able to somewhat type again, Project 8 is on the way! Thanks again, grader!

Who helped me:

For this project, I got help from CP Majgaard and met with Bruce a few times. In addition, I worked closely with Steve Parrott and Brendan Doyle.