Zena's CS251 Project 7

This project was all about kmeans clustering! Which means we created a way for the user to analyze the data by grouping them into however many categories they desired. The algorithm keeps recalculating the means of the data points, and regrouping them into the closest mean's cluster. Then this repeats for a provided maximum number of iterations until the recalculated means are a certain distance from all of the points in their clusters. Starting with categories makes the results more accurate because we can initialize the cluster means with predetermined means instead of making the algorithm do it. My program lets the user do all of this by first opening a file, then choosing kmeans from the command menu. The user chooses which columns to include in the algorithm's inputted data and how many clusters to make. Then, this gets turned into a new csv file named originalfilename-#clusters.csv where the number of clusters is the "#". Lastly, the user can open that and plot it normally, except now a checkbox appears that lets the user choose whether to color code clusters smoothly or with distinct colors. I tested my program on my wages data from project 5.

Task 1 was to test our kmeans classifiers on the provided massive dataset about walking, running, jumping, etc.

What the provided program does: Does primary component analysis to create a pcadata, finds out how many dimensions are needed to represent most of the data, then does kmeans algorithm using the provided data. The confusion matrix shows how many points belong in each category, with the actual being the column and the result being the row of a given index.

How I modified it: I made it have more max_iterations and a smaller min_change. I thought that the more iterations, the more times the mean would be checked for its proximity to each point, which would make it more centered. And lowering the min_change would mean the mean would have to be closer to a point before ceasing to move toward it. But in the end, changing these did not do much.

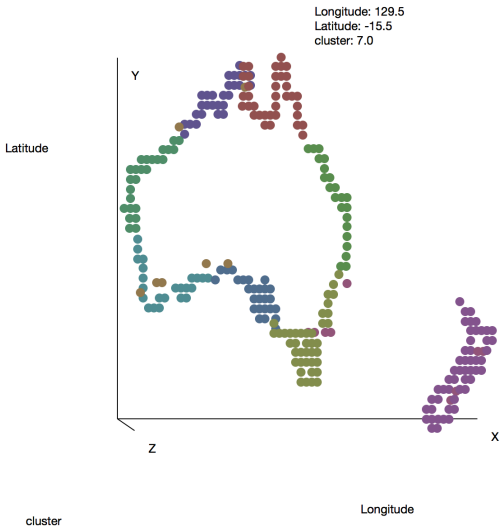
Step 2 was to enable the data class to write out to a file and also to add a column to the current data object. For writing a file, I first check if there are provided headers (there always are in this case, since the user selects them). If there are, I make a new matrix with those plus the cluster column (after I've added it in the add_column() method), and if not, I use the object's current matrix. Then, I write a line of the headers, separated by commas, a line of "numeric" for the types, also separated by commas, and finally, the actual data by row, separated by commas. To make sure the last item in a row doesn't get a comma, I also have an else statement that catches the last item based on the length of the list/matrix being looped over. I needed to convert row items to strings, as well.

For the add_column method, I simply horizontally stacked the column with the current matrix in the data object. Then, I added its name, which was passed in, manually to the dictionary, as well as appended another "numeric" to the raw_types, and appended the column numbers in header2matrix, raw_data, and header2raw.

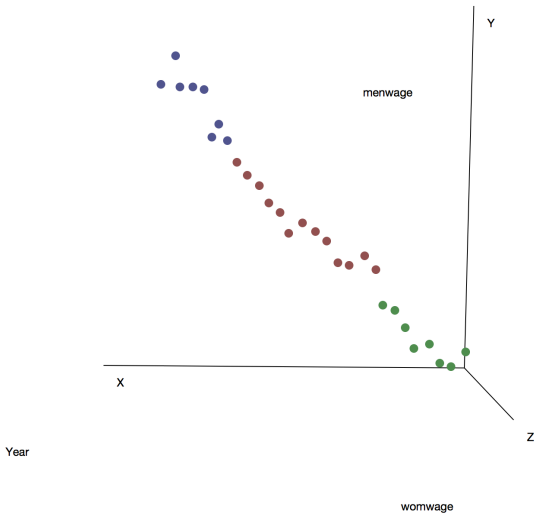
Step 3 was to add the ability to use the kmeans clustering in the interface. So I made a new option in the command menu for this and let the user choose what I mentioned before in my opening paragraph. I chose option 2, making a new file, because it saves the user the trouble of doing the analysis again every time they want to see the new data. Also, it is neater with just the columns the user used presented in the plotting menu.

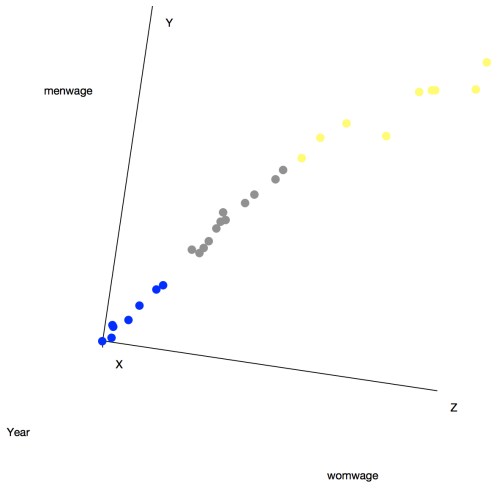
What gave me some trouble on step 4 was implementing color options. When plotting kmeans data, the assumption is that the user wants to use the default cluster ids to determine color, but if they specify a color column, that will override the default settings. Also, there is a checkbox for whether to have a gradient of yellow to blue or to have distinct colors. This checkbox only appears for kmeans data, which I confirm by checking if there is a column called "cluster". I ended up making a new color method, distinctColorFormula, which uses the colorsys python module to map the number inputted to an hsv tuple then converts it to an rgb tuple. I also multiply that tuple by 255 because it is just on a scale of 0 to 1 otherwise. An important part of creating distinct colors is to take into consideration how many clusters there are.

Task 5 was to plot the Australia data set cluster data (10 clusters). This is my result. The points are fairly well clustered but it seems that the light brown points are scattered around. Some lighter purple points are mixed into New Zealand, indicating that those were not all successfully grouped together, either. The more clusters we make, the less accurate the results seem to be.



For the last task, I clustered the data from project 5 about wages. I made 3 clusters; you can see them in blue, red, and green below. Since it was clustered about year, women's wages, and men's wages, it seems to be divided up into sections where the slope is consistent. All three dimensions were taken into account here. This shows that there is a sharper increase in men's wages compared to women's . As the second picture below shows, the men's wages tend to even out with the women's the higher they get, suggesting that the wages become more balanced the higher they get, but start off with a higher average for men. This is why they have been clustered along with the slight changes in slope.





This project was helpful for learning about how to code clustering algorithms. It was a challenge to incorporate colors, but I learned more about how to manipulate tuples through my new method. I also learned that my method of incorporating PCA is not very efficient, because I couldn't complete the extension doing kmeans on PCA data due to the way I checked for whether I had normal or PCA data so often. I ended up not doing it because I would have had to restructure a lot of things, which I don't have time for. It all leads back to the way I get the headers differently based on what kind of data is being used, it seems.

Thanks to: Stephanie, probably Melody