Implementation Summary

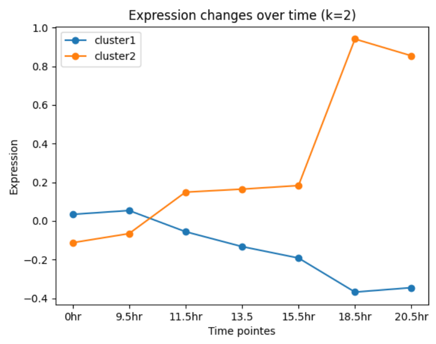
The K-mean clustering implementation takes in a matrix, which is implemented as a list of lists that can be accessed with matrix[i][j], and a k value that specifies how many clusters should be formed. The implementation starts with calling *initializeCenters* to choose k random centers from all the data points. Then, each data point is assigned to a cluster based on the minimum Euclidean distance between the point and the cluster centers. If a point has equal distance to more than one cluster centers, the tie is broken randomly. Then, *updateCenter* is called to modify the values of all existing centers based on new centers of gravities. Finally, *convergence* function takes in both the list of old centers and updated centers, to test for center convergence, which in this case the threshold is set at 10^{-6}. The implementation returns two items: a list of converged centers (list of lists), and a list of lists of data points (each list corresponds to all data points assigned to one cluster). Since a variety of plots was created throughout the project, only one instance was included in the code as an example.

The sample data (yeast gene expression data over 7 timepoints) were read in by the pandas library and data points with an entire row of NA values were removed, leaving 6392 rows in the matrix. For the rest of the missing values, column/timepoint mean was used as the method of imputation. After data imputation was done, yeast data was converted from a pandas data frame to a list of lists and inputted into the K-mean clustering implementation.

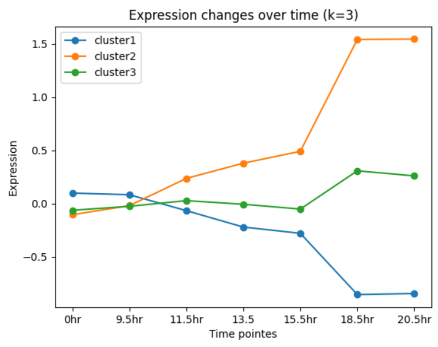
Analysis of Clustering Results and Outliers

By observing cluster centers with different K values:

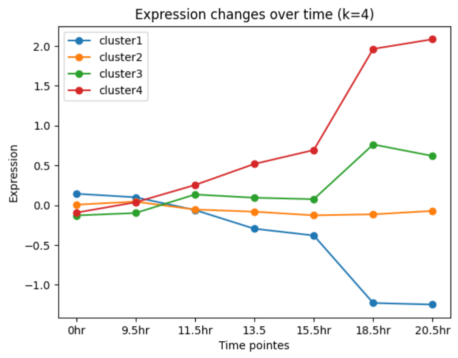
K=2



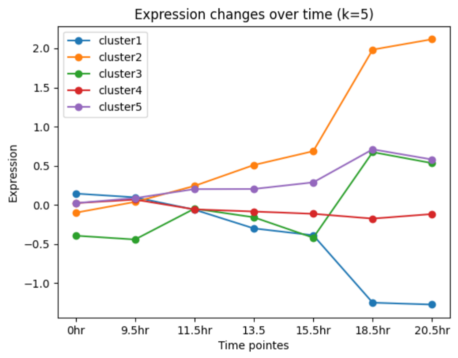
K=3



K=4

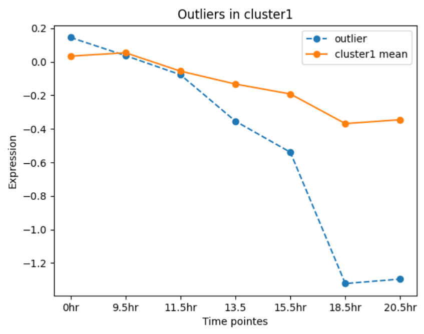


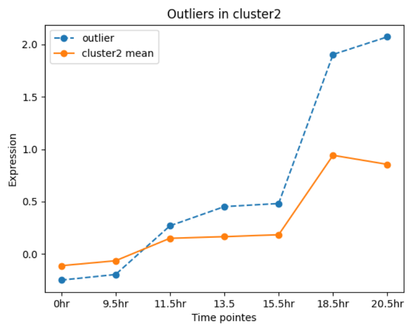
K=5



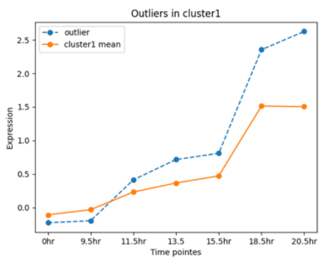
When the data is split into only 2 clusters, the two cluster means are one with increasing trend and one with decreasing trend, with the greatest difference at the latest time point. Time=20.5hr is the timepoint allowing the greatest variation of expression. When K value gets larger, most of the clusters are distinguished by their expression levels at 20.5hr (besides k=5). From the plots, it seems that K=3 or K=4 would be the best Ks to pick for clustering this yeast data set, as their cluster centers have clear distinctions and in general are non-overlapping. K=3 separates the data into 3 clusters of increasing, decreasing, and near-constant trends over time. K=4 separates the data into 4 clusters, with the third and the fourth clusters further differentiating the near-constant cluster in K=3. At K=5, the cluster centers are still distinct, but for two of the clusters at timepoint 20.5hr there is only a small difference in expression levels.

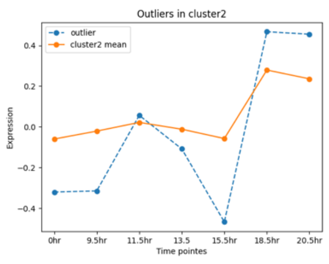
K=2 outliers in each cluster:

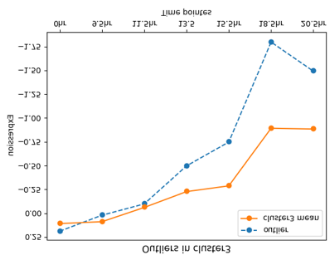




K=3 outliers in each cluster:



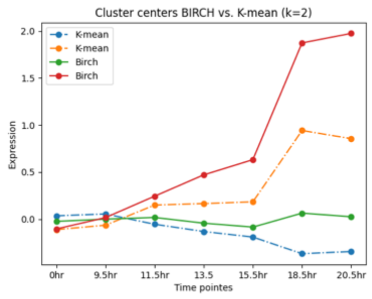


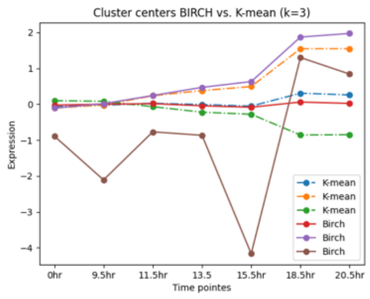


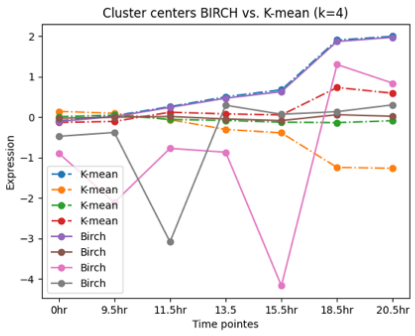
Outliers are found within each cluster using the distance from a point to its cluster center. Lower and upper boundaries for outliers are calculated using interquartile ranges. There are multiple outliers in each cluster, and the above plots show the mean of each cluster and the cluster's outlier mean together. When K=2, the outliers differ greatly in both clusters. It is likely because that 2 clusters are not enough to capture all varying patterns of the entire dataset. When K=3, the outliers persist, but the difference between those outliers to each cluster mean is slightly smaller. As K gets larger, it is expected that the difference between outliers and cluster mean in each cluster should get smaller (K=4 outliers plot in supplement).

Comparison of Results with the BIRCH Clustering Algorithm

The BIRCH algorithm was used for additional clustering of the yeast data and produced the following comparison:







From the comparison, K-means seems to give a better clustering result than BIRCH. Upon research, it was found that BIRCH utilizes clustering features (CF) and CF tree to cluster the data. It is mainly used to summarize large, multidimensional dataset into smaller, denser regions called clustering feature entries. Unlike the K-mean clustering algorithm which uses a partition approach, BIRCH clusters data hierarchically, which could be the reason here making the difference. There has been attempts of first applying BIRCH to generate a tree using hierarchical clustering, which would give a large number of clusters. Then, the K-mean method would be applied to reduce the number of clusters and with more accuracy and less sum of square error.

Future Improvements

Potential improvements of the K-mean clustering implementation could be to use a different data structure of storing the matrix. The major data structure used in the project was list of lists. A better and more efficient alternative would be to use numpy arrays, since numpy arrays provide more straight-forward methods of extracting row sums, column manipulations, and vectorized computations. Currently using a list of lists, entries had to be appended individually first into a row vector, then each individual row vector would need to be appended. By switching to numpy arrays, a lot of iteration steps, such as nested for loops, can be prevented to save time, hence making the code more readable. Another improvement would be to draw an elbow plot detailing the within-sum-of-square values on the K-mean clustering implementation. This could provide statistically grounded evidence to the optimal K value.