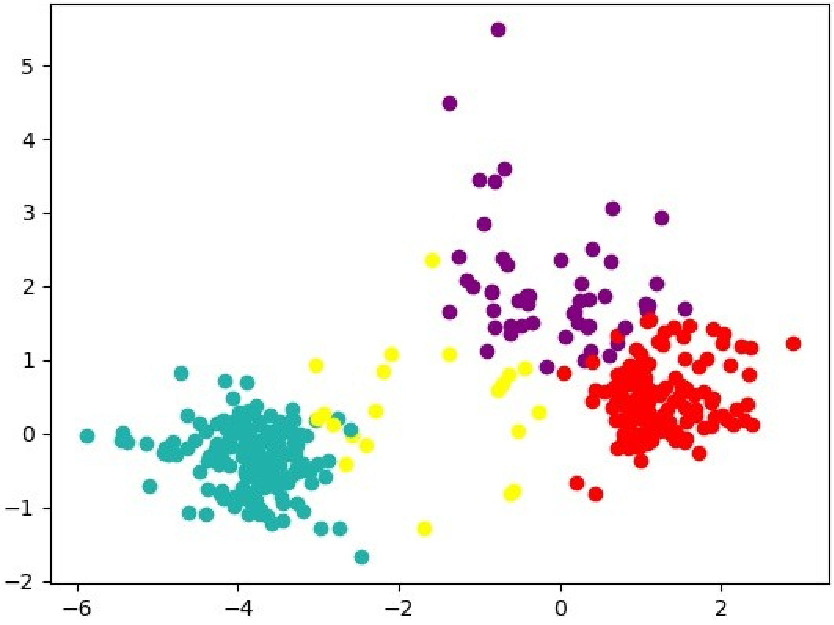
Nate Dauterman
ECE 49595
April 6th, 2022

Yeast Gene Centroids

```
[[-2.41271429e-01 -1.28750000e-01  6.22500000e-02  1.73350000e-01
   2.17914286e-01  1.65169286e+00  1.90532143e+00]

 [-9.53509804e-01 -1.47164706e+00  7.75294118e-02 -1.79490196e-01
  -1.00482353e+00  1.15121569e+00  9.68803922e-01]

 [ 1.65103659e-01  9.16524390e-02 -1.03884146e-01 -5.52585366e-01
  -6.30085366e-01 -1.72318293e+00 -1.75481098e+00]

 [ 2.32857143e-02  2.50809524e-01 -2.76952381e-01 -3.64000000e-01
  -7.35428571e-01 -8.94619048e-01  7.00142857e-01]

 [-1.55555556e-03  1.56506173e-01  3.56253086e-01  7.01580247e-01
   1.00971605e+00  1.84231481e+00  1.64341975e+00]

 [-3.93289474e-02  1.53947368e-01  4.36078947e-01  1.10581579e+00
   1.44871053e+00  3.01634211e+00  2.82938158e+00]]
```

Yest Gene PCA Plot



Utilities Merges

```
0-th merging: 12, 21, 23
1-th merging: 10, 13, 24
2-th merging: 4, 24, 25
3-th merging: 7, 23, 26
4-th merging: 25, 20, 27
5-th merging: 14, 19, 28
6-th merging: 1, 18, 29
7-th merging: 15, 26, 30
8-th merging: 29, 28, 31
9-th merging: 2, 27, 32
10-th merging: 8, 16, 33
11-th merging: 32, 30, 34
12-th merging: 34, 22, 35
13-th merging: 9, 31, 36
14-th merging: 35, 36, 37
15-th merging: 6, 37, 38
16-th merging: 3, 38, 39
17-th merging: 39, 33, 40
18-th merging: 17, 40, 41
19-th merging: 11, 41, 42
20-th merging: 5, 42, 43
```

K-Means Code

```
def assignCluster(dataSet, k, centroids):
    '''For each data point, assign it to the closest centroid
    Inputs:
        dataSet: each row represents an observation and
                 each column represents an attribute
        k:  number of clusters
        centroids: initial centroids or centroids of last iteration
    Output:
        clusterAssment: list
            assigned cluster id for each data point
    '''
    clusterAssment = []

    dataSetCopy = copy.deepcopy(dataSet)

    for data in dataSetCopy:
        minDist = -1
        minIndex = -1
        for cenInd, center in enumerate(centroids):
            totalDist = 0
            #print(data, center)
            for dat, cen in zip(np.asarray(data).flatten(), np.asarray(center).flatten()):
                #print(dat, cen)
                totalDist += (dat - cen) ** 2
            totalDist = sqrt(totalDist)
            if minDist == -1 or totalDist < minDist:
                minDist = totalDist
                minIndex = cenInd
        clusterAssment.append(minIndex)

    return clusterAssment


def getCentroid(dataSet, k, clusterAssment):
    '''recalculate centroids
    Input:
        dataSet: each row represents an observation and
            each column represents an attribute
        k:  number of clusters
        clusterAssment: list
            assigned cluster id for each data point
    Output:
        centroids: cluster centroids
    '''

    dataSetCopy = copy.deepcopy(dataSet)

    centroids = []
    #print(centroids)

    lastIndex = max(clusterAssment)
    for cenInd in range(lastIndex + 1):
        #print(cenInd)
        indexes = np.where(np.array(clusterAssment) == cenInd)[0]
        #print(indexes)
        totalPoints = 0
        for ind in indexes:
            if isinstance(totalPoints, int):
                totalPoints = copy.deepcopy(dataSet[ind])
            else:
                totalPoints += dataSet[ind]
        #print(totalPoints)
        totalPoints /= len(indexes)
        #print(totalPoints.tolist()[0])
        centroids.append(totalPoints.tolist()[0])
    centroids = np.matrix(centroids)

    return centroids
```

Hierarchical Cluster Code

```
def merge_cluster(distance_matrix, cluster_candidate, T):
    ''' Merge two closest clusters according to min distances
    1. Find the smallest entry in the distance matrix—suppose the entry
        is i-th row and j-th column
    2. Merge the clusters that correspond to the i-th row and j-th column
        of the distance matrix as a new cluster with index T

    Parameters:
    ------------
    distance_matrix : 2-D array
        distance matrix
    cluster_candidate : dictionary
        key is the cluster id, value is point ids in the cluster
    T: int
        current cluster index

    Returns:
    ------------
    cluster_candidate: dictionary
        upadted cluster dictionary after merging two clusters
        key is the cluster id, value is point ids in the cluster
    merge_list : list of tuples
        records the two old clusters' id and points that have just been merged.
        [(cluster_one_id, point_ids_in_cluster_one),
         (cluster_two_id, point_ids_in_cluster_two)]
    '''
    merge_list = []

    minValue = np.amin(distance_matrix)
    #print(minValue)
    index = np.where(distance_matrix.flatten() == minValue)
    coords = np.unravel_index(index, (len(distance_matrix), len(distance_matrix[0])))
    for key, value in cluster_candidate.items():
        if coords[0].flatten()[0] in value:
            id1 = key
        if coords[0].flatten()[1] in value:
            id2 = key

    merge_list = [(id1, copy.deepcopy(cluster_candidate[id1])), (id2,
copy.deepcopy(cluster_candidate[id2]))]

    points1 = copy.deepcopy(cluster_candidate[id1])
    points2 = copy.deepcopy(cluster_candidate[id2])
    points1.extend(points2)
    cluster_candidate[T] = points1

    cluster_candidate.pop(id1)
    cluster_candidate.pop(id2)

    return cluster_candidate, merge_list


def update_distance(distance_matrix, cluster_candidate, merge_list):
    ''' Update the distantce matrix

    Parameters:
    ------------
    distance_matrix : 2-D array
        distance matrix
    cluster_candidate : dictionary
        key is the updated cluster id, value is a list of point ids in the cluster
    merge_list : list of tuples
        records the two old clusters' id and points that have just been merged.
        [(cluster_one_id, point_ids_in_cluster_one),
         (cluster_two_id, point_ids_in_cluster_two)]

    Returns:
    ------------
    distance_matrix: 2-D array
        updated distance matrix
    '''

    coordList = [[a, b] for a in merge_list[0][1] for b in merge_list[1][1] if a != b]
    for coord in coordList:
        distance_matrix[coord[0]][coord[1]] = 100000
        distance_matrix[coord[1]][coord[0]] = 100000

    return distance_matrix
```