Quinn McHugh
Dr. Justin Ziniel
CSE 5243

Assignment #4 Report

A. Program Description

The program which I wrote heavily borrows code from the last two programming assignments, with appropriate modifications being made to suit the needs of the k-means algorithm. There are three notable design decisions that I made while coding KMeans, which I outline below.

The first consideration to be made was how to deal with empty clusters which may form through the occasional bad luck of k-means. My method of dealing with empty clusters involved identifying the cluster with highest SSE, and then assigning a random point from that cluster as the centroid to the empty one. This technique worked well and was effective in improving the algorithm, as before empty clusters occurred in approximately 1 out of 5 runs.

The second design consideration made concerned when to halt the cluster formation process. Instead of devising a formal method of determining when the centroids had settled, through strictly empirical observation I found that choosing a high number of cluster formation iterations sufficed. This required less code to be written, and the number of 50 was settled upon for the amount of iterations to perform.

My last design decision pertains only to the TwoDimensional datasets provided. When implementing the randomPopulate() method inside my TwoDimRecord class, I opted to randomly create points bound in the range [0, 1). My reasoning behind this was that after surveying the data via plots, no wider range would be necessary. I believe that this limited range for random centroid generation led to a quicker convergence, and hence shorter runtime.

B. True Cluster Calculations

True Clusters (for reference)

TwoDimEasy		**TwoDimHard**	
True C1	(0.069, 0.817)	True C1	(0.319, 0.762)
True C2	(0.919, 0.251)	True C2	(0.592, 0.709)
		True C3	(0.441, 0.343)
		True C4	(0.764, 0.425)

True Cluster SSE

TwoDimEasy		**TwoDimHard**	
C1 SSE	16.999	C1 SSE	0.313
C2 SSE	2.359	C2 SSE	0.903

	C3 SSE	2.430
	C4 SSE	1.911

True Overall SSE

TwoDimEasy Overall SSE	19.358
TwoDimHard Overall SSE	5.557

True SSB

TwoDimEasy Overall SSB	35.642
TwoDimHard Overall SSB	15.314

C. Three K-Means Iterations

Initial Centroids (for reference)

TwoDimEasy			
	Trial 1	Trial 2	Trial 3
Cluster 1	(0.409, 0.319)	(0.184, 0.148)	(0.033, 0.554)
Cluster 2	(0.507, 0.109)	(0.787, 0.769)	(0.799, 0.333)
TwoDimHard			
	Trial 1	Trial 2	Trial 3
Cluster 1	(0.118, 0.157)	(0.605, 0.088)	(0.245, 0.489)
Cluster 2	(0.428, 0.071)	(0.407, 0.474)	(0.252, 0.212)
Cluster 3	(0.021, 0.743)	(0.037, 0.367)	(0.545, 0.745)
Cluster 4	(0.901, 0.427)	(0.613, 0.507)	(0.764, 0.280)

I) SSE and SSB

TwoDimEasy			
	Trial 1	Trial 2	Trial 3
Cluster 1 SSE	2.783	2.783	2.783

Cluster 2 SSE	16.295	16.295	16.295
Overall SSE	19.078	19.078	19.078
TwoDimHard			
	Trial 1	Trial 2	Trial 3
Cluster 1 SSE	0.918	1.471	0.562
Cluster 2 SSE	0.774	0.500	0.655
Cluster 3 SSE	4.257	1.845	4.437
Cluster 4 SSE	1.908	1.076	2.074
Overall SSE	7.857	4.892	7.782

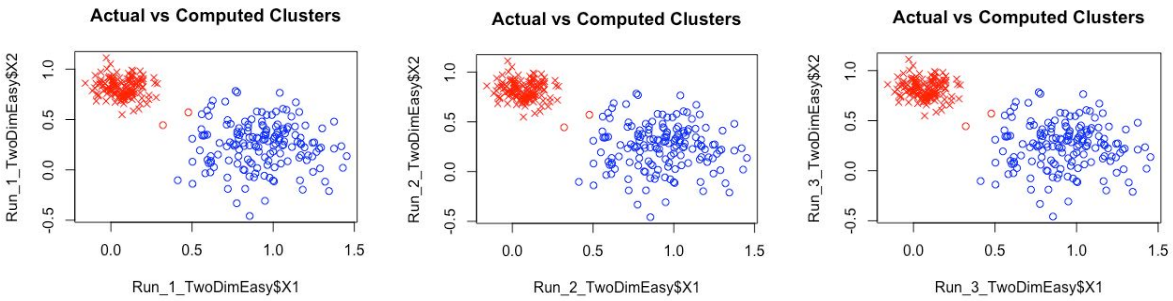
	Trial 1	Trial 2	Trial 3
TwoDimEasy SSB	78.045	78.045	78.045
TwoDimHard SSB	21.579	24.412	21.592

II) Silhouette Widths

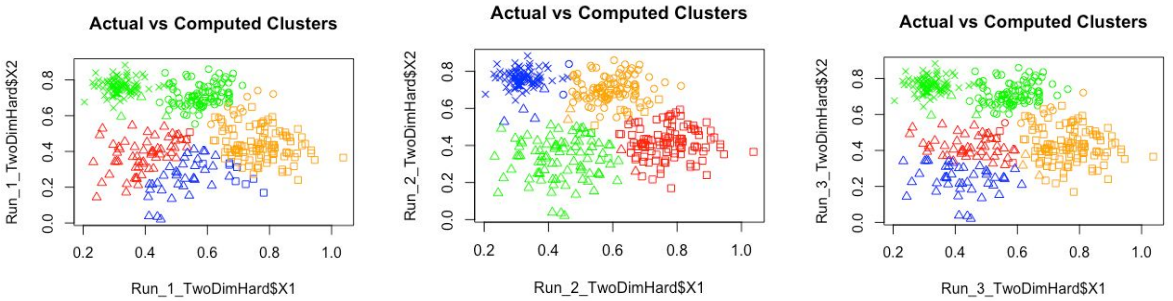
TwoDimEasy			
	Trial 1	Trial 2	Trial 3
Cluster 1 Width	0.830	0.830	0.830
Cluster 2 Width	0.598	0.598	0.598
TwoDimHard			
	Trial 1	Trial 2	Trial 3
Cluster 1 Width	0.430	0.516	0.430
Cluster 2 Width	0.711	0.711	0.711
Cluster 3 Width	0.516	0.430	0.521
Cluster 4 Width	0.521	0.521	0.516

III) True vs Estimated Scatterplots

TwoDimEasy Runs 1-3



TwoDimHard Runs 1-3



IV) Confusion Matrices

TwoDimEasy - Cluster 1 - Runs 1-3 (All same)			**TwoDimEasy - Cluster 2 - Runs 1-3 (All same)**		
	Estimated No	Estimated Yes		Estimated No	Estimated Yes
True No	160	2	True No	138	0
True Yes	0	138	True Yes	2	160

TwoDimHard - Cluster 1 - Run 1			**TwoDimHard - Cluster 1 - Run 2**		
	Estimated No	Estimated Yes		Estimated No	Estimated Yes
True No	251	60	True No	204	107
True Yes	89	0	True Yes	89	0
TwoDimHard - Cluster 1 - Run 3			**TwoDimHard - Cluster 2 - Run 1**		
	Estimated No	Estimated Yes		Estimated No	Estimated Yes
True No	261	50	True No	257	43
True Yes	89	0	True Yes	100	0
TwoDimHard - Cluster 2 - Run 2			**TwoDimHard - Cluster 2 - Run 3**		

	Estimated No	Estimated Yes		Estimated No	Estimated Yes
True No	207	93	True No	256	44
True Yes	98	2	True Yes	100	0
TwoDimHard - Cluster 3 - Run 1			**TwoDimHard - Cluster 3 - Run 2**		
	Estimated No	Estimated Yes		Estimated No	Estimated Yes
True No	122	181	True No	301	2
True Yes	94	3	True Yes	9	88
TwoDimHard - Cluster 3 - Run 3			**TwoDimHard - Cluster 4 - Run 1**		
	Estimated No	Estimated Yes		Estimated No	Estimated Yes
True No	120	183	True No	276	10
True Yes	94	3	True Yes	11	103
TwoDimHard - Cluster 4 - Run 2			**TwoDimHard - Cluster 4 - Run 3**		
	Estimated No	Estimated Yes		Estimated No	Estimated Yes
True No	186	100	True No	276	10
True Yes	106	8	True Yes	4	110

V) Discussion

In the case of the TwoDimEasy dataset, I saw that no matter the initial centroids that were selected by the k-means algorithm, the clusters eventually converged to the same points. This is representative of how well separated the easy clusters are how little ambiguity exists.

In the case of the TwoDimHard dataset, I saw that very different clusterings could occur based on differences in the initial centroids selected. This follows naturally as a result of the increased difficulty of the dataset. While examining the SSE produced in clustering the TwoDimHard dataset, I saw that the highest overall SSE experienced in a trial was 7.857, while the lowest overall was 4.892. The difference in error here is large. The SSB values observed in three trials of TwoDimHard also spanned a greater distance than the static SSB computed in TwoDimEasy.

The scatterplots (and the associated confusion matrices) are also highly variable under the TwoDimHard dataset and constant under the TwoDimEasy set. We can see by choice of initial centroids in the TwoDimHard scatterplots that well spaced initial centroids that are relatively close to actual cluster centroids fair well in producing agreeable metrics. This phenomenon of course occurs by chance and can be best seen in the 2nd run of k-means on TwoDimHard. In the other cases (1st and 3rd runs) the results were less favorable with suboptimal clusters being formed. This too can be traced back to the initial centroid positions, which I observed not to be near to the true centers.

D. Analysis on K=3

Initial Centroids (for reference)

TwoDimEasy		**TwoDimHard**	
	Centroid	Trial 2	Centroid
Cluster 1	(0.908, 0.743)	Cluster 1	(0.822, 0.364)
Cluster 2	(0.228, 0.420)	Cluster 2	(0.093, 0.010)
Cluster 3	(0.531, 0.431)	Cluster 3	(0.755, 0.725)

I) SSE and SSB

TwoDimEasy		**TwoDimHard**	
Cluster 1 SSE	5.261	Cluster 1 SSE	2.039
Cluster 2 SSE	2.783	Cluster 2 SSE	1.959
Cluster 3 SSE	5.711	Cluster 3 SSE	4.566
Overall SSE	13.755	Overall SSE	8.564

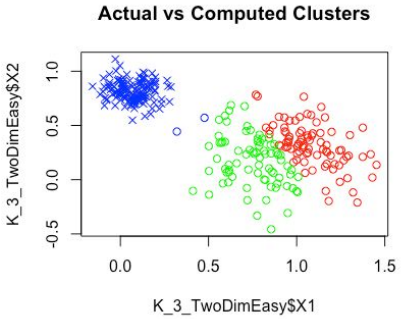
	SSB
TwoDimEasy	83.369
TwoDimHard	20.740

II) Silhouette Widths

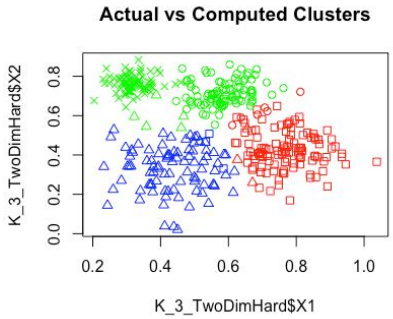
TwoDimEasy		**TwoDimHard**	
	Width		Width
Cluster 1	0.620	Cluster 1	0.624
Cluster 2	0.517	Cluster 2	0.515
Cluster 3	0.618	Cluster 3	0.622

III) True vs Estimated Scatterplots

TwoDimEasy



TwoDimHard



IV) Confusion Matrices

TwoDimEasy - Cluster 1			**TwoDimHard - Cluster 1**		
	Estimated No	Estimated Yes		Estimated No	Estimated Yes
True No	179	83	True No	192	119
True Yes	138	0	True Yes	89	0
TwoDimEasy - Cluster 2			**TwoDimHard - Cluster 2**		
	Estimated No	Estimated Yes		Estimated No	Estimated Yes
True No	100	138	True No	208	92
True Yes	160	2	True Yes	100	0
TwoDimEasy - Cluster 3			**TwoDimHard - Cluster 3**		
	Estimated No	Estimated Yes		Estimated No	Estimated Yes
True No	323	77	True No	119	184
True Yes	0	0	True Yes	92	5

V) Discussion

In this section I will compare the results of changing k to 3 on the two datasets compared to their original k-values, 2 and 4, respectively. With k=3 the SSE of TwoDimEasy is reduced significantly showing a promising improvement, but the SSE of TwoDimHard increased slightly with this k change. The SSB measures showed no real improvements with the change as TwoDimEasy's SSB got larger and TwoDimHard's merely stayed constant. The silhouette widths for TwoDimEasy did not improve as they got smaller, and no significant change in the average silhouette width was seen in TwoDimHard. Lastly, the most easily legible data (the scatterplots and confusion matrices) show different improvements depending on what dataset is being analyzed. In the case of TwoDimEasy, the k=3 clustering looks very unnatural and results in high error rates. In the case of TwoDimHard, the k=3 clustering seems less suspect when investigating the scatterplot than when analyzing TwoDimEasy. Of course, since it is

known that there is a true 4th cluster not being represented, a comparatively high error rate is also achieved on the TwoDimHard dataset.

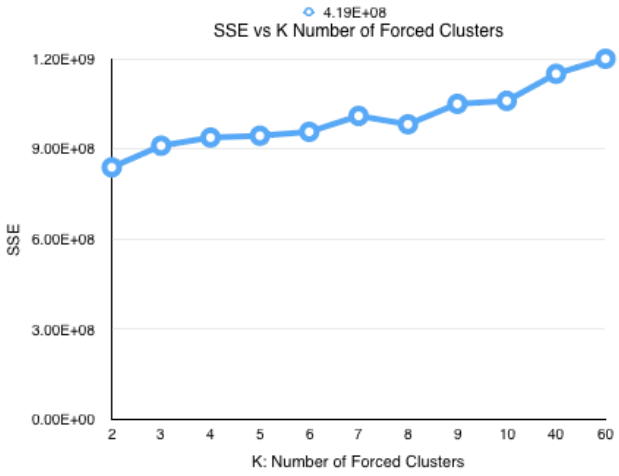
Finally to end this k=3 analysis, I conclude that while changing the number of clusters does indeed change the results, the change yields overall worse performance and should not be used.

E. K Cluster Experimentation

In my experimentation with the number of clusters, k, the principal method I used to evaluate the initial effectiveness of my clusterings was the SSE metric. The number of clusters, k, that I investigated were the integers in the range [2, 10] and two more significantly larger values, 40 and 60. These two larger values were included in the experiment to see if any long term trends might exist without necessarily including every value in between. The initial conclusion that I reached was that under my implementation of k-means, there was very low variability in the SSE metric, suggesting low differences in the quality of the various clusterings.

F. Detailed Analysis

Continuing from the previous section, this is the line plot produced from the average of my 13 test clusterings:



As can be seen, there appears to be a trend which states, “the higher the k value, the lower the clustering quality according to SSE”. This seems to run contrary to what we learned in class, which is that as the value k increases beyond a rational range, the clustering quality actually improves due to sub-clusters being formed around the true centroids.

Another way in which I analyzed the results of my algorithm’s application to the wine quality dataset involved silhouette widths. Below is a table of the widths of the computed clusters for k’s 2 through 10:

Number of Forced Clusters K vs Average Silhouette Width									
2	.618	**4**	.545	**6**	.505	**8**	.473	**10**	.446

3	.573	**5**	.522	**7**	.490	**9**	.459	

This table of silhouette widths shows a clear preference for clusterings with lower K value, as there exists a steady decreasing trend in width as K increases. This seems to agree with the finding of the K vs SSE line plot. Therefore from my analysis, my preferred clustering is k=2 as this is the optimal K value according to both metrics.

G. External Validation Comparison

According to the quality attribute in the provided dataset, there are six different integers which can be recorded ranging from 3 to 8. This would suggest that the true clustering for this dataset would have 6 distinct centroids. This finding is very different from what my algorithm produced, which seemed to prefer a k=2 clustering as stated in the previous section.

Part 2: A. Description of Clustering Used

The off-the-shelf clustering method which I chose to use was the standard pam() function available in R's 'cluster' library. Before using the pam function, however, the dataset was cleansed of unnecessary and pre-labeled fields including 'ID' and 'cluster'. The parameter settings used were while running the pam() function were 'stand' and 'metric'. The parameter 'metric' was set equal to 'euclidean' and 'stand' was set equal to FALSE for TwoDimEasy and wine_quality-red, and to TRUE for TwoDimHard. The 'stand' variable either enables or disables the normalization of column data, and resulted in improved silhouette width for the boolean just specified for each dataset. Lastly, k was set to 2 while clustering TwoDimEasy, set to 4 while clustering TwoDimHard, and set to 6 for wine_quality-red.

B. Results

Below is a table displaying the average silhouette widths for all clusters per dataset as outputted by R, and two tables displaying the computed medoids for each clustering:

Dataset	**Average Silhouette Width of All Clusters**
TwoDimEasy	.687
TwoDimHard	.549
wine_quality-red	.388

	Medoids			
TwoDimEasy	(.064, .80)	(.967, .284)		
TwoDimHard	(.319, .755)	(.594, .701)	(.435, .366)	(.768, .415)

wine_quality-red:

```
Medoids:
       ID fx_acidity vol_acidity citric_acid resid_sugar chlorides free_sulf_d tot_sulf_d density   pH sulph
[1,]  381        8.3       0.260        0.42         2.0     0.080          11         27 0.99740 3.21  0.80
[2,]  797        8.7       0.460        0.31         2.5     0.126          24         64 0.99746 3.10  0.74
[3,] 1026        8.6       0.830        0.00         2.8     0.095          17         43 0.99822 3.33  0.60
[4,]  648        8.3       0.845        0.01         2.2     0.070           5         14 0.99670 3.32  0.58
[5,]  780        7.1       0.520        0.03         2.6     0.076          21         92 0.99745 3.50  0.60
[6,] 1560        7.8       0.600        0.26         2.0     0.080          31        131 0.99622 3.21  0.52
```

C. Comparison of Results

Since the metric being quantified in the above results section is silhouette width, this then is the basis for comparison between R's pam() function and the k-means implementation from Part 1. The results show that for the TwoDimEasy dataset, R's pam() function performed better with a width averaging .687, besting Part 1's implementation which averaged .585. The opposite is true, however, for the TwoDimHard and wine_quality-red (assuming the true cluster count k=6) datasets where the widths of .587 and .505, respectively, are larger than pam()'s .549 and .388 widths.