Homework 4 – K-Mean Clustering

Anushka Nair, Bam Nakapakorn, and Madeline Luong

Preliminary treatment:

The previous data has been standardized to prevent bias and all non- numerical and missing values have been discarded. The filtering is 0.4 or 0.7 strength and 0 or 1 for italic. We have used the datasets listed below.
Before Filtering:
BODONI – 3964 instances and 403 features
REFERENCE – 4652 instances and 403 features
ROMAN – 4776 instances and 403 features
After Filtering:
BODONI – 991 instances and 403 features (CL1)
REFERENCE –1163 instances and 403 features (CL2)
ROMAN –1194 instances and 403 features (CL3)
We define the three classes (CL1, CL2. And CL3) to each dataset and unionize them into a full dataset. Next, we standardize the data and call it SFONT. SFONT then contains 3348 observations / instances and 403 features.

1.1 Applying the K- Means function on standardized Font data where K = 1-10.

K- Means is a form of unsupervised machine learning algorithm that aims to classify a given dataset through k predefined clusters, where k is specified by the user. The objective of k-means is to group similar observations and discover underlying patterns. First, the algorithm identifies k random observations to be the initial centroids, and then assigns each observation to the closest cluster centroid. Next, it recalculates the centroid value based on cluster membership. This process is repeated until the cluster center is stabilized. Applying multiply K values is a trial-and-error process to identify the best number of clusters. Here we apply the kmeans() function with multiple values to our data set using a for loop with an nstart of 50. The nstart is used to produce consistent results. The default number of iterations for kmeans() is 10, which in our case is not enough for the algorithm to converge and reach its stopping criterion, so we set the number of iterations to 50 to overcome this issue. This function provides us the cluster means, cluster vector, and the cluster reduction of variance. We use the k means () function on the standardized features SFONT Y (1) … Y(400). The values are placed in an excel document labeled k-means values.

1.2 Computing each K and its respective Reduction of Variance performance Perf(K).

The goal of the k-means algorithm is to minimize the dispersion, also referred to as the within-cluster sum of squares (SWS), which is a measure of variability of observations within each cluster. A lower within-cluster sum of squares indicates that the cluster is more compact. To calculate the within-cluster variation for the kth cluster, the sum of all the pairwise squared Euclidian distances between the observations in the kth cluster are calculated and then divided by the total number of observations in the kth cluster. In order to calculate the clustering performance, we first need the clustering quality - Q(k). Q(k) is calculated by dividing SWS by SWS(1), where SWS(1) is the dispersion of the whole dataset around the center. In R, we extract SWS by calling the $tot.withinss function, and SWS(1) by calling the $totss function. Finally, the reduction of variance performance is calculated using the formula Perf(k) = 1-Q(k). The values are placed in the table below, where K is the K value, and perf(k) is the respective % reduction of variance.

K	SWS(K)	Q(K): SWS(K)/SWS(1)	Perf(K): 1-Q(K)
1	1338800	1.000	0.000
2	1195653	0.893	0.107
3	1113653	0.832	0.168
4	1059145	0.791	0.209
5	1021218	0.763	0.237
6	994182	0.743	0.257
7	970991.7	0.725	0.275
8	950497.9	0.710	0.290
9	930547	0.695	0.305
10	916743.2	0.685	0.315

1.3 Visualizing Perf(k) vs K and finding best K Using the Elbow Rule.

To select the optimal number of clusters, we must visualize the change in variance of the models against the respective values of K. To understand how well the model is performing, we plot Q(k), i.e SWS(K)/SWS(1) squares by the clusters (k). We then use the elbow rule to identify the best value for K, which is done by picking a K value where the within sum of squared distance begins to flatten and form an elbow.
Below are the plots for K versus the clustering quality and Perf(K) respectively.

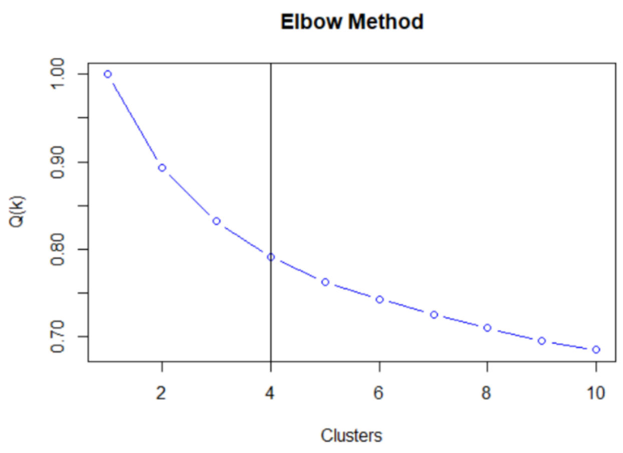


Figure – 1 Q(k) vs Clusters (k) Plot
The x-axis is the number of clusters k and the y-axis is the clustering quality. The vertical line represents the "ideal" k value.

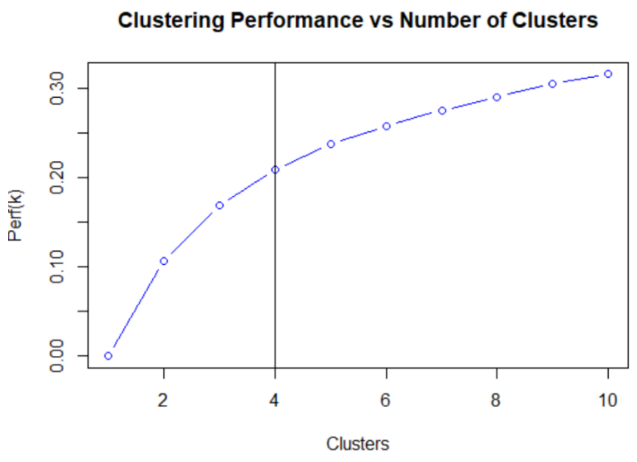


Figure – 2 Perf(k) vs Clusters (k) Plot
The x-axis is the number of clusters k and the y-axis is the reduction of variance performance. The vertical line represents the "ideal" k value.

The elbow rule seeks to find a k value k* such that Q(k) decreases fast for k<k* and decreases slowly for k>k*. Looking at the Q(k) plot above, we can see that when k<4, the slope of Q(k) drops faster and once it passes the k value of k=4, the slope seems to level off. Another option is using the silhouette analysis to determine the degree of separation between each cluster.

2.1 Computing the matrix with each cluster center with best K and the cluster plot

After identifying the "ideal" K (k =4) cluster value, the K – means algorithm will be conducted using the ideal k. Next, using the $centers call, we extract the 4x400 (kxp) matrix which computes the mean of each predictor by cluster number. Each row is the coordinates for each center. The full values will be displayed in the excel document. Below are the first 6 columns of the of the matrix. We call the cluster centers CENT1, CENT2, CENT3 and CENT4.

	Y1	**Y2**	**Y3**	**Y4**	**Y5**	**Y6**
1	-0.1072543	-0.2188557	-0.2703826	-0.2629172	-0.2892679	-0.3719344
2	-0.0170443	0.08798093	0.134552	0.1459339	0.15921503	0.1495674
3	0.49877531	0.48115	0.50272684	0.48434259	0.56931798	0.76221315
4	-0.0002677	0.03027237	0.03851512	0.02573986	0.02027922	0.06539181

K-means partitioning clusters is a clustering method to classify observations into groups based on similarity. Plotting each coordinate/ point can aid in understanding the relationship of each cluster partition. To visualize each cluster, we use kmbest (k=4) and plot the data using the fviz_cluster() function. The plot shows the first two dimensions.

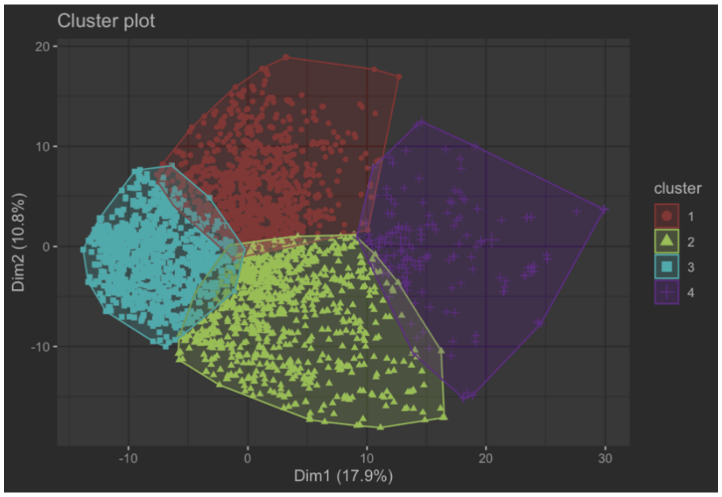
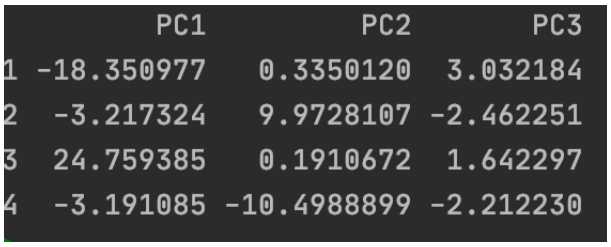


Figure 2.1 – 4 Clusters plot
The X- axis represents the dimension 1 from PCA and the Y – axis represents dimension 2 from PCA. The legend shows the corresponding color cluster pair.

We can see that a majority of the overlap is between cluster three and two. This can lead to a lower accuracy in these two clusters. We can assume that each cluster is disjoint from each other.

2.2 Conducting a PCA to compute the three-dimensional vectors CENT1, CENT2, CENT3 and CENT4

We conduct a PCA on the four cluster centers and obtain a 4 x 4 matrix, where the columns are the four principal components. This will give us the orthogonal projection of respective CENTERS on the three-dimensional space which is generated by first eigenvectors of the correlation matrix. We do this in R by applying the prcomp() function to the k x p cluster center matrix we computed in section 2.1. Next, we extract the first three columns, representing the first three principal components. Looking at the R output, we find that the first three principal components explain 100% of the variation in the cluster centers. The 3-dimensional vectors c1, c2, c3 and c4 are listed below:



```
         PC1        PC2       PC3
1 -18.350977  0.3350120  3.032184
2  -3.217324  9.9728107 -2.462251
3  24.759385  0.1910672  1.642297
4  -3.191085 -10.4988899 -2.212230
```

2.3 Displaying the 4 x 3 matrix on a 3D graph.

We want to visualize the cluster centers to see the relation between the 4 cluster mean distances.
Ideally, we would want the cluster means to be far apart. The 4x3 principal component matrix obtained above are the coordinates of the 4-centroid means. We extract and plot these coordinates using the 3D scatter function to visualize the centroid distances.

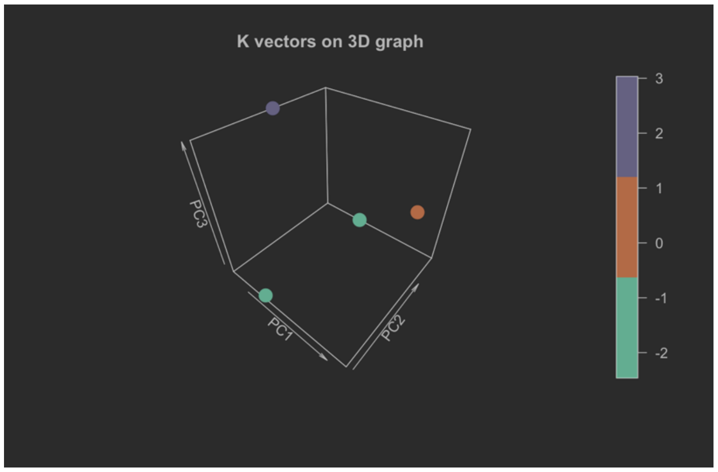


Figure – 3 Plot displaying K Vectors
The X – Axis is PC1 and the Y – Axis is PC2. The Z – Axis is PC3. Each point represents the centroid mean location. Green is PC1 centroids location. The orange color is PC2's centroid location and PC3 is Purple.

We can see that the centroids are far from each other. If they are close together then the clusters are too similar.
The points are displayed on a 3D plot, where the three dimensions are the three principal components.
As the plot indicates, two of the centroids, namely cluster 2 and 3 are mainly captured by PC1. The purple centroid of cluster 1 is captured by PC3. The orange centroid of cluster 4 is mainly explained by PC2.
From the 3D plot, we can infer that cluster 1 will perform the best, since it is the farthest away from all the other cluster centroids. This is confirmed later when we calculate the accuracy.

2.4 Identifying the largest Cluster and computing the PCA and extracting only the first 3 PC's.

We append the cluster assignment column to SFONT using the $cluster call. CLU1 has 1104 cases, CLU2 has 908 cases, CLU3 has 269 cases and CLU4 has 1067 cases. Cluster 1 is the biggest cluster with 1104 cases.
Since the K-means algorithm is distance based, a bigger cluster will tend to do better than a smaller cluster.
We perform a PCA on bigCLU using the prcomp() function, then we extract the first three principal components leaving us with a 1104x3 matrix. These are the three-dimensional vectors v1…v1104.

2.5 Displaying the Vectors on a 3D plot.

We display the three-dimensional vectors v1…v1104 using the libraries rgl, scatterplot3d, and plot3D in R. To understand the composition of bigCLU1 better, we assign three different colors to the three font classes.

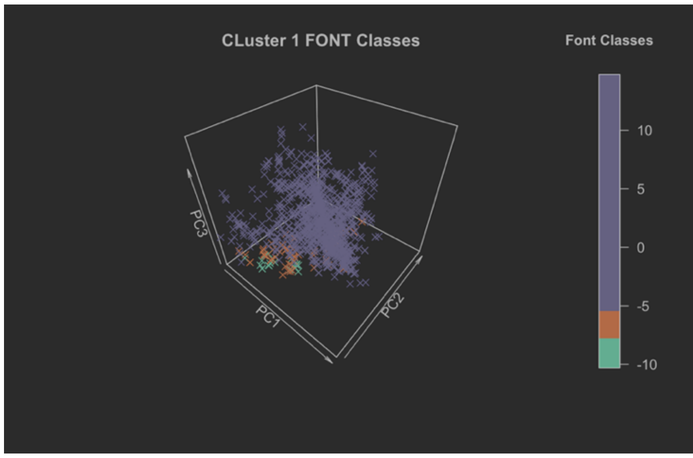


Figure 4 – Cluster 1 Font Class 3D Plot
The X – Axis is PC1 and the Y – Axis is PC2. The Z – Axis is PC3. The green points are class 1. The orange points are class 2. The purple points are class 3. This plot is the top profile.

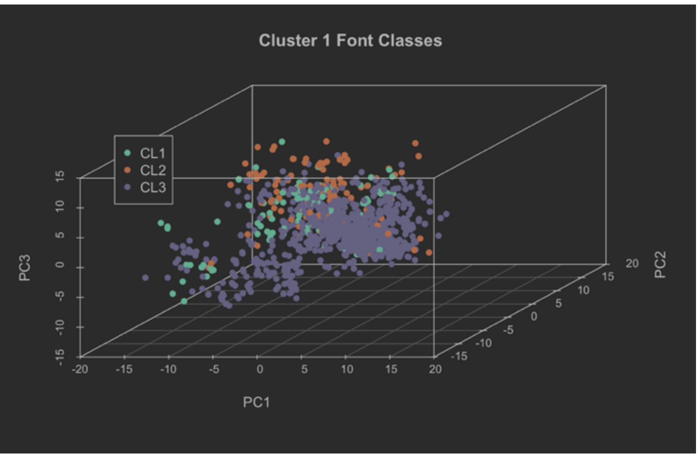


Figure 5 – Cluster 1 Font Class 3D Plot
The X – Axis is PC1 and the Y – Axis is PC2. The Z – Axis is PC3. The corresponding point colors are on the legend. This plot is the side profile

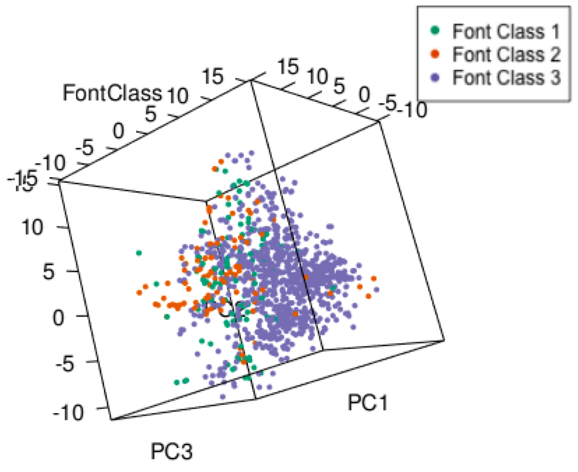


Figure 6 – Cluster 1 Font Class 3D Plot
The X – Axis is PC1 and the Y – Axis is PC2. The Z – Axis is PC3. The corresponding point colors are on the legend.

Looking at the plots, we can see that majority of the points classify as font class 3. This implies that bigCLU (cluster 1) is the "best" cluster to classify font class 3.

3.1 Compute the Gini index for each cluster and the impurity of best kmeans clustering

The Gini index is used to evaluate the purity of classification. To find the Gini Index for each cluster, we use the formula *F1(j)*(1-F1(j)) + ... + Fk(j)*(1-Fk(j))* where Fmj represents the % cases of class(m) within cluster j. The closer the gini index is to 0, the higher the purity of classification. If all the elements are linked with a single class, then it would have a gini index of 0. Below are the Gini index values for each cluster.

Gini(Cluster 1) = 0.3299842
Gini(Cluster 2) = 0.5610021
Gini(Cluster 3) = 0.4484183
Gini(Cluster 4) = 0.5477717
Imp(k*) = 1.887

The values indicate that cluster 1 is the purest cluster, since its value is the closest to 0. The next "purest" cluster would be cluster 3. These two clusters will have the highest two accuracies in classified their respective class fonts. The least "pure" is cluster 2, this is explained because cluster 2 has some general overlapping from figure 2.1.

The impurity i.e. Imp(k*) of the clustering CLU1, CLU2, CLU3 and CLU4 is calculated by using the formula: Imp(k*) = Gini(Cluster 1)+ Gini(Cluster 2)+ Gini(Cluster 3)+ Gini(Cluster 4). Adding the calculated values, we get Imp(k*) = 1.887. A way to improve imp(k) is to apply different k values.

3.2 Compute A(m,j) and Fm(j):

To find each font class and cluster intersection, which is denoted by A(m,j), we create a subset for each individual cluster. Then we use the table function to obtain the sizes of each font class/ cluster pair. To calculate the frequency of the class and cluster intersection we use the formula: Fmj = A(m,j)/Sj where m represents the class number, j represents the cluster number and Sj= size(CLUj). Below, we have displayed the values of each class and cluster intersection along with its respective frequency.

CLUSTER 1	
A (Class1, Cluster1) = 112	Fm(class1, cluster1) = 10.14%
A (Class2, Cluster1) =101	Fm(class2, cluster1) = 9.14%
A (Class3, Cluster1) = 891	Fm(class3, cluster1) = 80.70%
CLUSTER 2	
A (Class1, Cluster2) = 476	Fm(class1, cluster2) = 52%
A (Class2, Cluster2) = 361	Fm(class2, cluster2) = 39.75%
A (Class3, Cluster2) = 71	Fm(class3, cluster2) = 7.81%
CLUSTER 3	
A (Class1, Cluster3) = 192	Fm(class1, cluster3) = 71.37%
A (Class2, Cluster3) = 45	Fm(class2, cluster3) = 16.72%
A (Class3, Cluster3) = 32	Fm(class3, cluster3) = 11.89%
CLUSTER 4	
A (Class1, Cluster4) = 211	Fm(class1, cluster4) = 19.77%
A (Class2, Cluster4) = 656	Fm(class2, cluster4) = 61.48%
A (Class3, Cluster4) = 200	Fm(class3, cluster4) = 18.74%

3.3 Display the 3x4 frequency matrix

The confusion matrix is created by using the table function in R. Below displays the relationships between the Clusters and Classes.

		CLUSTERS			
CLASSES		**CLU1**	**CLU2**	**CLU3**	**CLU4**
	CL1	10.14%	52%	71.37%	19.77%
	CL2	9.14%	39.75%	16.72%	61.48%
	CL3	80.70%	7.81%	11.89%	18.74%

From this confusion matrix we can see that a majority of cluster 1 is CL3. Further, we can see that cluster 2 mostly contains CL1 and CL2. Cluster 3 has a majority of CL1, and Cluster 4 has a majority of CL2. This confusion matrix shows that every class is a majority in one of the clusters. Cluster 1 and Cluster 3 seem to have the most homogeneity in their cases.

3.4 Compute A(TOP(j),j)

For each j=(1,2,3,4) , we want to calculate the majority class index TOP(j). To find this value we use the top frequency (class) value of each cluster. The following formula is used: A(TOP(j),j)=max(A(1,j), A(2,j), A(3,j), A(4,j)). Below are the values:

A(TOP(1),1) = class 3 which is 80.70652%
A(TOP(2),2) = class 1 which is 52.42291%
A(TOP(3),3) = class 1 which is 71.37546%
A(TOP(4),4) = class 2 which is 61.48079%

We can see that cluster 1 will classify CL3, cluster 2 and 3 will classify CL1 and cluster 4 will classify CL2. These are not accuracies, they are frequencies. To visualize how accurate the K – means algorithm is, we must calculate the confusion matrix comparing the predicted and true class.

4.1 Compute the confusion matrix of the predictor

The output $cluster of the kmeans function lists the cluster assignment for each case n. We use these cluster assignments to define the predictor Pred(n), which looks at the cluster assignment. Next, it classifies each case(n) into a font class using the A(TOP(j),j) values that were calculated in section 3.4. If a specific case(n) is assigned to cluster 1, then Pred(n) will classify the case as Class 3. Similarly, if case(n) is assigned to cluster 2 or 3, the predictor will classify the case as Class 1. Lastly, if a case is assigned to cluster 4, then the classification based on the predictor will be Class 2. We use the table function to calculate the confusion matrix:

		TRUE		
		CL1	**CL2**	**CL3**
PRED	**CL1**	56.75%	34.49%	8.75%
	CL2	19.78%	61.48%	18.74%
	CL3	10.14%	9.15%	80.71%

Total accuracy: 66.31%

The overall accuracy of the k means model is not very high. The confusion matrix indicates the most accurately classified class is CL3 at 80.71%. However, there is a high number of false positive (19%) and false negative (26%). CL2 has 61.48% true accuracies with 43% false negatives and 37% false positives. CL1 has true accuracy of 56.75% but with 29% false positives and 42% false negatives.

Class 1 has the lowest accuracy, and that could be because both Cluster 2 and 3 were used to classify font Class 1(Bodoni). Size(cluster2) +Size (Cluster3) => 908+269 = 1177 cases are automatically labeled as Class 1 by the predictor, which is more than the number of cases of Class 1: Bodoni (991 classified instances). This leads to high percentage (37%) of false positives.

Class 3 has the highest accuracy, which could be because Cluster 1 was used to classify it. The centroid for cluster 1 was the farthest away from all the other cluster centroids, indicating a distinct grouping. Class 3: Roman:1194 features. Cluster 1:1104 classified instances.

Comparing CONF vs FREQ matrices.

The accuracy of Predicted class 2 is the same as frequency distribution of Cluster 4. The accuracy of Predicted class 3 is the same as frequency distribution of Cluster 1. This is because only Cluster 4 was used to predict

Class 2, and only cluster 1was used to predict Class 3. The accuracies for predicted Class 1 is the proportion of Cluster 2 and 3, since they both were used to predict it. If the number of clusters were equal to the number of classes, the frequency and confusion matrix would be identical. However, in our case we had three classes and four clusters.

Conclusions

Since the k means algorithm is distance based, it is extremely sensitive to outliers. However, k means is a good way to visualize the distances between groups in a dataset and relatively simple to implement. An alternative to k-means clustering is hierarchical clustering, which is a method that does not require you to specify a k value. This might improve results, since the k means() accuracy is entirely dependent on choosing the right k value.

Script :

```
################################# Data Cleaning #################################
library(dplyr) # Package for subseting data

attach(BODONI)
bodoni_clean <- select(BODONI,-c(2,3,6,7,8,9,10,11,12)) # Discard the 9 columns
# View(bodoni_clean)
# 3964x403

attach(REFERENCE)
ref_clean <- select(REFERENCE,-c(2,3,6,7,8,9,10,11,12))
# View(ref_clean)
# 4652x403

attach(ROMAN)
roman_clean <- select(ROMAN,-c(2,3,6,7,8,9,10,11,12))
# View(roman_clean)
# 4776x403

# Discarding row containing missing numerical data
BODONI_clean <- na.omit(bodoni_clean)
REF_clean <- na.omit(ref_clean)
ROMAN_clean <- na.omit(roman_clean)

# Defining three classes of images of normal characters
# cl1 = all rows of BODONI_clean.csv file for which (strength = 0.4 and italic =0)
# cl2 = all rows of REF_clean.csv file for which (strength = 0.4 and italic =0)
# cl3 = all rows of ROMAN_clean.csv file for which (strength = 0.4 and italic =0)

BODONI_clean <- data.frame(BODONI_clean)#creating a data frame to add conditional
statements to filter out non needed i features
BODONI_clean$CL = ifelse((BODONI_clean$strength == 0.4 & BODONI_clean$italic ==
0),"CL1","NA")
BODONI_CLEAN = BODONI_clean[which(BODONI_clean$CL =="CL1"),] #labeling the new filter
data as cl1
# 404 columns
# 991 rows

REF_clean <- data.frame(REF_clean)
REF_clean$CL = ifelse((REF_clean$strength == 0.4 & REF_clean$italic == 0),"CL2","NA")
REF_CLEAN = REF_clean[which(REF_clean$CL =="CL2"),]
# 404 columns
# 1163 rows

ROMAN_clean <- data.frame(ROMAN_clean)
ROMAN_clean$CL = ifelse((ROMAN_clean$strength == 0.4 & ROMAN_clean$italic ==
0),"CL3","NA")
ROMAN_CLEAN = ROMAN_clean[which(ROMAN_clean$CL =="CL3"),]
# 404 columns
# 1194 rows

# Combine CL1, CL2, CL3 into DATA
DATA <- rbind(BODONI_CLEAN,REF_CLEAN,ROMAN_CLEAN)
# View(DATA)

# Binded all 3 data sets to a full data set (DATA) which is the union of 3 classes (CL1,
CL2, CL3)
# where N = 716
# 404 columns
# Create standardize data set without font, strength, and italics (include CL)
library(standardize)
SFONT <- DATA %>% mutate_if(is.numeric, function (x) as.vector(scale(x))) # scaling by
(xj-mj)/sd
SFONT = SFONT[,-c(1,2,3)] # taking out numerical functions of font, strength, and italics
# View(SFONT)
# 401 columns
# 716 rows

############################## K- MEANS LIBARBY ####################################
library(factoextra)
library(plot3D)
library(rgl)
library(scatterplot3d)

############################# QUESTION 1 ####################################
SFONT_p <- SFONT[,-c(401)]
#applying K- MEANS
k_max<-10

for (k in 1:k_max){
  kmfont<- kmeans(SFONT_p,k,nstart = 50,iter.max = 50)
}

#finding the the reduction of variance for each k (perf(k))
Varfont<-kmfont$withinss
Varfont
totalvalue=kmfont$tot.withinss # sum of all withins

#plot the curve Perf(k) vs K and identifying using the elbow rule
Qm <- numeric(10)
Perf.m <- numeric(10)
for (k in seq(1,10)){
  kmfont <- kmeans(SFONT_p, k, nstart=50, iter.max=50) # SDATA
  Q <- kmfont$tot.withinss/kmfont$totss
  Perf <- 1-(kmfont$tot.withinss/kmfont$totss)
  Qm[k] = Q
  Perf.m[k] = Perf
}
par(mfrow=c(2,1))
plot(seq(1,10),Qm, xlab = 'Clusters', ylab = 'Q(m)',
     main = 'Elbow Method for Clustering Quality',col='red', type='b')
plot(seq(1,10),Perf.m,xlab = 'Clusters', ylab = 'Perf(m)',
     main = 'Clustering Performance vs Number of Clusters',col='red', type='b')
par(mfrow=c(1,1))

#best k = is at 4

########################### QUESTION 2 ############################

#running K means on best k = 4
kmbest<- kmeans(SFONT_p,4,nstart = 50)

#disjoint 2D plot
library(plotly)
fviz_cluster(kmbest, SFONT_p, stand = FALSE, frame = FALSE, geom = "point")


#computing the centers
kmcenters<- kmbest$centers #matrix showing 3x400
write.table(kmcenters, file = "K Means Centriod values.csv", sep = ",", row.names = TRUE)
```

```r
#running a PCA on the centers
pcakm<-prcomp(kmcenters,scale=TRUE)

#finding the vectors of pca centers
pcakm
#computing kx3 matrix listing the k vectors; extracting only first three pcs
xvector<-data.frame(pcakm$x[,1:3])
xvector # k centriod mean vectors value points
#displaying 3D graph of each vector
x <- xvector$PC1
y <- xvector$PC2
z <- xvector$PC3
scatter3D(x,y,z,main= "K vectors on 3D graph", xlab="PC1",ylab="PC2",zlab="PC3",col =
c("#1B9E77", "#D95F02", "#7570B3")
,pch=19,cex=2)

#counting clusters
table(kmbest$cluster) # cluster1: 1652, cluster2: 494 cluster3:1202
SFONT$cluster<-as.factor(kmbest$cluster) #binding each cluster to SFONT

#calling cluster with the largest size
bigCLU<-subset(SFONT,SFONT$cluster==1) #size is 1652 x 401 Taking out only cluster 1

#compute 3D vectors using the 3 Principal components
bigCLU1<- bigCLU[,c(-401,-402)] #taking out non-numerical values to apply pca on cluster

#applying PCA on cluster 1
bigCLU.pca<-prcomp(bigCLU1,scale=TRUE)
PC<-bigCLU.pca$x # The PCA values that has been tranpose and multiplied
PC3<-data.frame(PC[,c(1,2,3)]) # extracting only the first three Principal compoments
PC3$CL<-bigCLU[,401] # adding the class column back in

PC3$CL <- factor(PC3$CL, levels = c("CL1","CL2","CL3")) # creating levels for colors to
append

PCA_Clus1<-data.frame(PC3) #dataframe with the CL levels

colors <- c("#1B9E77", "#D95F02", "#7570B3") # creating color function
colors <- colors[as.numeric(PCA_Clus1$CL)] # adding the colors numbers to each CL

### creating plots for cluster 1 where each class reppressents a color###

# top plot
scat1<-scatter3D(PCA_Clus1$PC1,PCA_Clus1$PC2,PCA_Clus1$PC3, pch = 4,
col=colors,main="CLuster 1 FONT Classes",
                 xlab= "PC1",ylab ="PC2",zlab="PC3",clab = "Font Classes")

#side plot
scat2<-scatterplot3d(PCA_Clus1[,1:3], pch = 16, color=colors, main = "Cluster 1 Font
Classes")
legend(scat2$xyz.convert(-25,1,15),legend = levels(PCA_Clus1$CL),col =c("#1B9E77",
"#D95F02", "#7570B3"),pch=16 )

#3D movement plot
plot3d(PCA_Clus1, col=colors, main="FontClass")

# add legend
legend3d("topright", legend = paste('Font Class', c('1', '2', '3')), pch = 16, col=
c("#1B9E77", "#D95F02", "#7570B3"),
         cex=1, inset=c(0.02))
```

```r
################################# QUESTION 3
#################################################
#calculating gini index for each cluster
#cluster 1
clu1<-subset(SFONT,SFONT$cluster == "1")
table(clu1$CL) # cl1 : 112, cl2: 101, cl3:891 # this cluster defines class 3
#frequency
(clu1f1<-112/1104)*100 #10.14493%
(clu1f2<-101/1104)*100 #9.148551%
(clu1f3<-891/1104)*100 #80.70652%
#top j is class 3

#gini
(ginicl1<-clu1f1*(1-clu1f1)+clu1f2*(1-clu1f2)+clu1f3*(1-clu1f3))*100 # 0.3299842



#cluster 2
clu2<-subset(SFONT,SFONT$cluster == "2")
table(clu2$CL)
#frequency
(clu2f1<-476/908)*100 #52.42291%
(clu2f2<-361/908)*100 #39.75771%
(clu2f3<-71/908)*100 #7.819383%
#top j is class 1

#gini
(ginicl2<-clu2f1*(1-clu2f1)+clu2f2*(1-clu2f2)+clu2f3*(1-clu2f3)) # 0.5610021

#cluster 3
clu3<-subset(SFONT,SFONT$cluster == "3")
table(clu3$CL)
#frequency
(clu3f1<-192/269)*100 #71.37546%
(clu3f2<-45/269)*100 #16.72862%
(clu3f3<-32/269)*100 #11.89591%
#top j is class 1

#gini
(ginicl3<-clu3f1*(1-clu3f1)+clu3f2*(1-clu3f2)+clu3f3*(1-clu3f3)) # 0.4484183

#cluster 4
clu4<-subset(SFONT,SFONT$cluster == "4")
table(clu4$CL)
#frequency
(clu4f1<-211/1067)*100 #19.77507%
(clu4f2<-656/1067)*100 #61.48079%
(clu4f3<-200/1067)*100 # 18.74414%
#top j is class 2

#gini
(ginicl4<-clu4f1*(1-clu4f1)+clu4f2*(1-clu4f2)+clu4f3*(1-clu4f3)) # 0.5477717

#compute the IMP(K*)
(imp_k <- ginicl1+ginicl2+ginicl3+ginicl4) # 1.887176

#display frequency in 3x4 of frequencies FREQ= fm(j)

###### WRITE DOWN AMJ#########
```

```
#computing maxiumum A(TOP(j),j)= max A(1,j),A(2,j),A(3,j)
#A(TOP(1),1) = class 3 which is 80.70652%
#A(TOP(2),2) = class 1 which is 52.42291%
#A(TOP(3),3) = class 1 which is 71.37546%
#A(TOP(3),4) = class 2 which is 61.48079%

################################## QUESTION 4
###############################################
SFONT$predclass = NA
SFONT$predclass<-ifelse(SFONT$cluster =="1","CL_3",SFONT$predclass)
SFONT$predclass<-ifelse(SFONT$cluster =="2","CL_1",SFONT$predclass)
SFONT$predclass<-ifelse(SFONT$cluster =="3","CL_1",SFONT$predclass)
SFONT$predclass<-ifelse(SFONT$cluster =="4","CL_2",SFONT$predclass)

table(SFONT$predclass,SFONT$CL) # rows are predicted , columns are true values

mean(SFONT$predclass == SFONT$CL)
```