Computer Assignment 4 - PCA and Clustering

Machine Learning, Spring 2020

YOUR NAME

> Important note: We are noticing that there have been a lack of informative titles on the plots turned in for Computer Assignments. This is partly our fault, since not all plots given as examples have had titles. For the remainder of class we hope to improve on this, just as we hope to instill in each of you the habit of INFORMATIVE titles. Please try and make your plots as informative as possible moving forward. This is a valuable skill.

PCA for Authorship Identification

We will now apply above analysis to an interesting real life example.

In 1787, after the American Revolution, the colonies were debating whether to ratify the new Constitution. Three men - John Jay, James Madison, and Alexander Hamilton - wrote a series of essays in support of the Constitution, known as the Federalist Papers. In total, 85 papers were included.

Historical evidence shows that Jay authored 5 papers, Madison 14, and Hamilton 51. A further 3 were joint efforts between Madison and Hamilton. The remaining 12 are disputed. We will use PCA to try to guess at the authorship of these papers.

We will use a dataset that consists of word proportions of how many times a particular word was used divided by the total word count of an essay. This dataset is restricted to 70 function words - common words such as "a", "but", and "the".

Load and parse the data using the following commands:

```
# Load the data
fed = read.csv("Federalist.txt", header = TRUE)

# Have a look at the data
head(fed)

# Store the authorship vector separately
auths = fed[, 2]
known = auths != "DIS" & auths != "COL"

auths.known = fed[known, 2]

# Limit the data to known authors
fed.known = fed[known, -c(1,2)]
```

Questions

1. Treating the words as different variables, run PCA on the dataset using the *prcomp()* command. Here we use *scale = TRUE* to rescale the data. Are we centering the data here? (Hint: check the manual page!!) How many PCs are required to explain 80% of the variation in the data? Plot a screeplot of the PCs.

```
pc.fed = prcomp(fed.known, scale = TRUE)
YOUR CODE HERE
```

2. Make individual plots the first 3 PCs. Color the points by author by using the *col = auths.known* command in the *pairs()* function.

```
YOUR CODE HERE
pairs(pc.fed$x[,1:3], col = auths.known, main = "Pairs Plot of First Three PCs")
```

Does the data cluster by author? No need to explicitly cluster just yet; simply give your perspective on what you see in the biplots. How many dimensions would you project onto to separate essays by author?

YOUR ANSWER HERE

Now let's check to see if clustering confirms what we can see visually! Perform kmeans clustering on the first two principle components using the following code.

```
library(ggplot2)
set.seed(1)
km <- kmeans(pc.fed$x[,1:2], centers = 3)
ggplot(, aes(x = pc.fed$x[,1], y = pc.fed$x[,2])) +
  geom_point(aes(col = as.factor(auths.known), pch = as.factor(km$cluster))) + xlab("First PC") + ylab(
  scale_colour_discrete(name = "Author") + scale_shape_discrete(name = "Cluster") + geom_point(aes(x = I
```

Questions

1. Comment on the given plot. How well was kmeans able to distinguish the authorship? YOUR ANSWER HERE
2. What is the within-cluster sum of squares for this particular clustering?

```
YOUR CODE HERE
```

Prediction

We will now use the low-dimensional projection to help us guess about the disputed authorship. The function *predict()* will take information from a previous PCA and project new data onto the PC dimensions. Use the following code to find the projections for the disputed and collaborative essays.

```
fed.disp = fed[auths == "DIS", -c(1,2)]
fed.collab = fed[auths == "COL", -c(1,2)]
disp.pred = predict(pc.fed, fed.disp)
collab.pred = predict(pc.fed, fed.collab)
```

Questions

1. Plot the known data in first two PC dimensions, and color it by authorship (this will use *col = auths.known*). Then add the projection of the collaborative papers onto the first to PCs to the plot. We have provided most the the code to the latter plot because it was a little tricky. Make sure to add an informative title. If you would like a challenge, try to implement this last plot in ggplot (this is optional).

Make sure you label the axes and plot appropriately.

```
CODE FOR PLOTTING THE KNOWN DATA HERE

# To add the collaborative papers
## NOTE: Make sure to run the 'plot' and 'legend' commands simultaneously.  It will NOT WORK
```

```
##       if you do it line by line.
all.classifiers = as.factor(c(as.character(auths.known), rep("COL", times =nrow(collab.pred))))
plot(rbind(pc.fed$x[,1:2], collab.pred[,1:2]), col = all.classifiers, main = "Plot of First Two PCs; Co
legend(-10, 5,legend = c("AH", "COL", "JJ", "JM"), col = as.character(1:4), pch=rep(1, times = 4))
```

Based on this plot, who do you think did the primary work on the collaborations?

YOUR ANSWER HERE

2. Now add the projection of the disputed papers to the plot. Mimic what was done on the previous plot to do this. Based on this, who do you think authored the 12 papers?

```
YOUR CODE HERE
```

k-means

Now that we have seen a basic example of how to use kmeans with the Federalist papers, we are going to get further practice with a gene data example.

Load and parse the TCGA data from the course website using the following commands:

```
#Load the data
trial.sample = read.table("TCGA_sample.txt", header = TRUE)
#Store the subtypes of tissue and the gene expression data
Subtypes = trial.sample[,1]
Gene.Expression = as.matrix(trial.sample[,2:2001])
```

To run the k-means algorithm on the rows of a matrix X, the *kmeans(X, k, iter.max)* command can be used where k is the number of clusters to find and *iter.max* is the maximum number of iterations used to find the k partition. Once you run y = *kmeans(x,k,iter.max)*, you can type *y$cluster* to obtain the cluster labels of the data points. Also, you can type *y$tot.withinss* to obtain the total within cluster sum of squares (WCSS) for the identified partition.

Recall that k-means searches for the partition of the data that minimizes the WCSS for a fixed k. To get an idea of how k affects the WCSS, run k means for k from 1 to 20 and calculate the WCSS for each partition. Then, plot the WCSS across k by using the following code:

```
withinss = rep(NA, 20)

for(k in 1:20){
    z = kmeans(Gene.Expression,k,iter.max = 100)
    withinss[k] = z$tot.withinss
}

plot(withinss, xlab = "k", ylab = "WCSS", main = "k means on TCGA")
```

Questions:

1. Comment on WCSS as a function of k. Do your findings from the above plot make intuitive sense? Why or why not?
2. Comment about the WCSS at $k = 2$.
3. Re-run k means for $k = 2$ and compare the cluster labels for the results at $k = 2$ with the true subtype labels. What proportion of each cluster contains “Normal” and “Basal” subtypes?

k-means with Principal Component Analysis

It would be nice if we could visualize the clusters we just found, like we did with the Federalist Papers data. However, the TCGA data we are using has 2000 dimensions - and this is just a subset of the full data, which has about 20,000 genes! Luckily, we already have a way to reduce the dimensions of a dataset: PCA!

Perform PCA on the dataset *Gene.Expression* using the following code:

```
pc.tcga = prcomp(Gene.Expression)
```

Questions:

1. Plot the projections in the first two PC dimensions. Color by subtype using the *col = Subtypes* command in *plot( )*. Make sure to label your axes. Do the subtypes appear to cluster in this low-dimensional projection?

```
YOUR CODE HERE
```

2. Now plot the first two PC dimensions, coloring by the cluster you found in Part 1, Question 3. How well do these clusters appear to match the true subtypes?

```
YOUR CODE HERE
```

3. Perform k-means using **only** the first two PC dimensions. Plot the data one more time, coloring by the clusters found. What is the total within-cluster sum of squares for this clustering?

```
set.seed(1)
k.pca = kmeans(pc.tcga$x[,1:2], centers = 2)
YOUR CODE HERE
```

4. In your own words, explain why using PCA to reduce the number of dimensions from 2000 to 2 did not significantly change the results of k-means.

YOUR ANSWER HERE

Thinking further about inital cluster centers

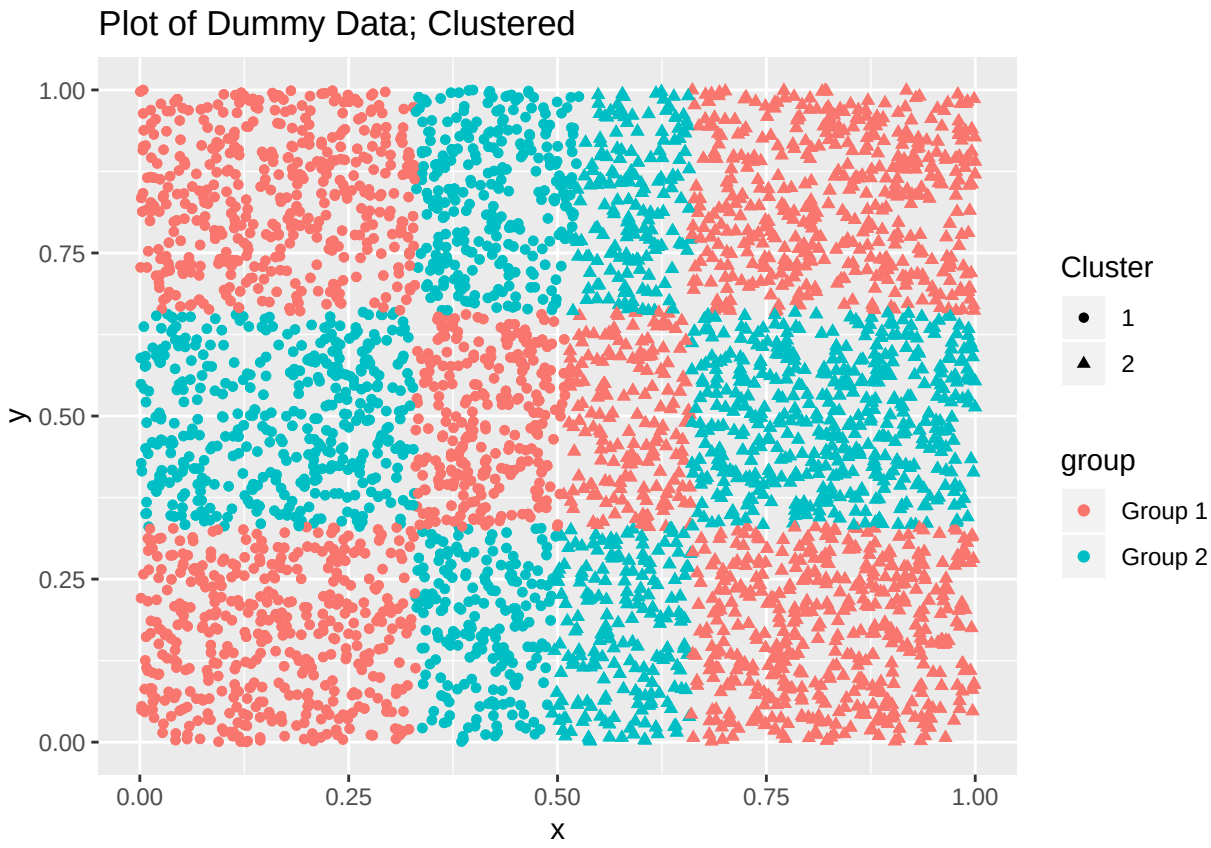
When passing a number x as the argument for `centers` to the kmeans algorithm, x cluster centers are chosen randomly, points are assigned to a cluster based on these centers, and then the cluster centers are iteratively updated in an attempt to find the centers that minimize total within-cluster sum of squares. The number of times `kmeans` updates is controlled by the argument `iter.max`. To examine how the choice of initial centers affects the algorithm, we will disable the updating steps via `iter.max` and choose some initial cluster centers of our own.

```
library(ggplot2)
# First, we create a dummy data set
set.seed(10)
random_data = data.frame(x = runif(4000),
                         y = runif(4000),
                         group = rep(NA, times = 8000))
random_data[which(random_data$x < 0.33 & random_data$y < 0.33 |
                  random_data$x > 0.66 & random_data$y < 0.33 |
                  random_data$x < 0.33 & random_data$y > 0.66 |
                  random_data$x > 0.66 & random_data$y > 0.66 |
                  (random_data$x > 0.33 & random_data$x < 0.66 &
                     random_data$y > 0.33 & random_data$y < 0.66)),]$group = "Group 1"
```

```
random_data[which(is.na(random_data$group)),]$group = "Group 2"

# This data set looks like:
ggplot(random_data, aes(x = x, y = y, col = group)) + geom_point() +
  ggtitle("Plot of Dummy Data")
```

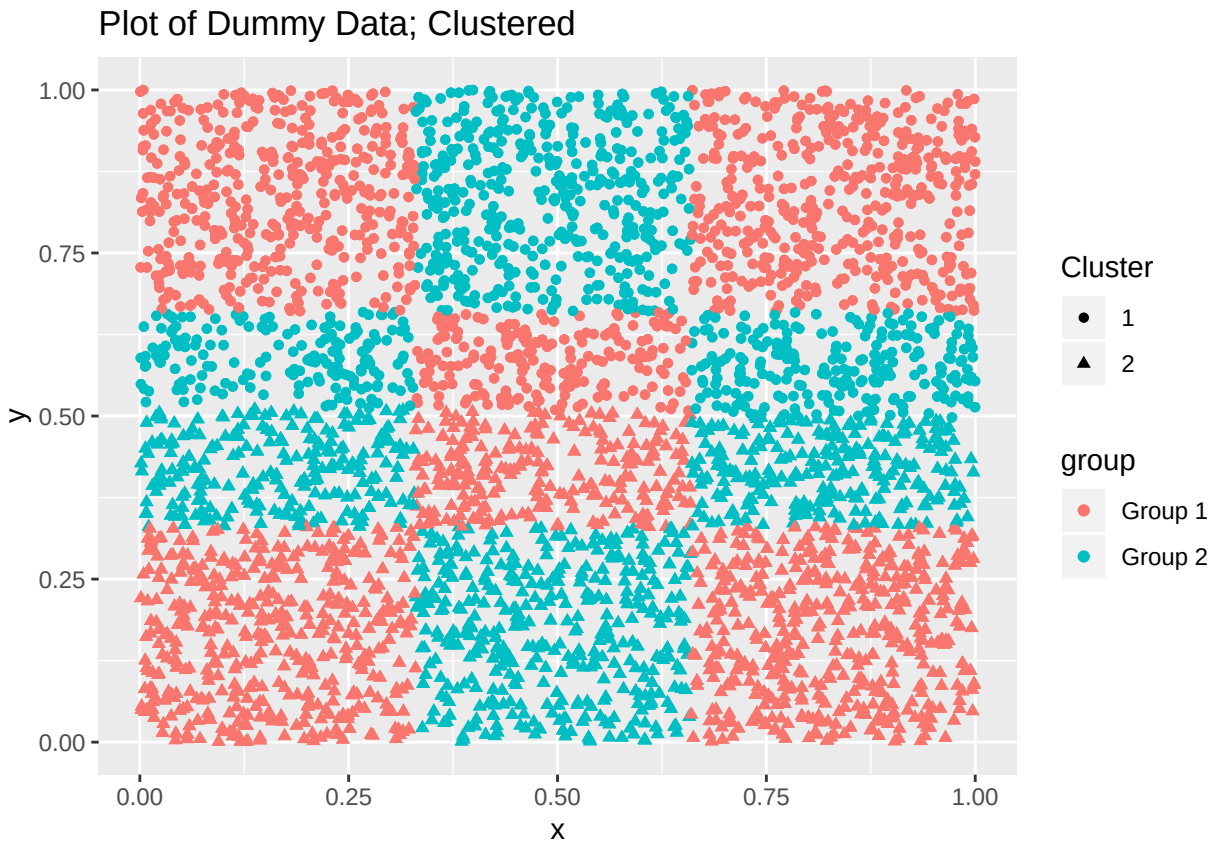

```
# We'll perform the kmeans with two random centers:
random.km = kmeans(random_data[,1:2], centers = 2, iter.max = 1)
ggplot(random_data, aes(x = x, y = y, col = group, pch = as.factor(random.km$cluster))) +
  geom_point() +
  scale_shape_discrete(name = "Cluster") +
  ggtitle("Plot of Dummy Data; Clustered")
```




```
# Which gives the following within-cluster sum of squares:
random.km$tot.withinss
```

```
## [1] 829.424
```

```
my_centers = random_data[1:2, 1:2]
random.km = kmeans(random_data[,1:2], centers = my_centers, iter.max = 1)
ggplot(random_data, aes(x = x, y = y, col = group, pch = as.factor(random.km$cluster))) +
  geom_point() +
  scale_shape_discrete(name = "Cluster") +
  ggtitle("Plot of Dummy Data; Clustered")
```




```
random.km$tot.withinss
```

```
## [1] 847.1953
```

Don't just look at the number!! Can you see how the clusters themselves changed? Now, choose your own cluster centers! These *do not* need to be actual observed points from our data. Try something weird, and report the total within-cluster sum of squares. Did it change? Was it better or worse than the random choice? Comment as to why you think that is.

```
YOUR CODE, AND ANALYSIS, HERE
```