Stats 101C HW 8

Charles Liu (304804942)

12/8/2020

Loading Necessary Packages:

```r
library(ggplot2)
```

Problem 1 (Exercise 10.7.10)

In this problem, you will generate simulated data, and then perform PCA and K-means clustering on the data.

(a) Generate a simulated data set with 20 observations in each of three classes (i.e. 60 observations total), and 50 variables. Hint: There are a number of functions in R that you can use to generate data. One example is the rnorm() function; runif() is another option. Be sure to add a mean shift to the observations in each class so that there are three distinct classes.

```r
set.seed(1)
class_1 <- matrix(rnorm(20*50, mean = 0, sd = 0.1), ncol = 50)
class_2 <- matrix(rnorm(20*50, mean = 5, sd = 0.1), ncol = 50)
class_3 <- matrix(rnorm(20*50, mean = 10, sd = 0.1), ncol = 50)
sim_data <- rbind(class_1, class_2, class_3)
sim_data <- cbind(sim_data,
                  class = as.factor(c( rep(1,20), rep(2,20), rep(3,20) )))
sim_df <- data.frame(sim_data)
dim(sim_df) # 60 observations & 50 variables w/ class column
```

```
## [1] 60 51
```

```r
head(sim_df)
```

```
##            V1           V2          V3           V4          V5           V6
## 1 -0.06264538  0.091897737 -0.01645236  0.240161776 -0.05686687 -0.062036668
## 2  0.01836433  0.078213630 -0.02533617 -0.003924000 -0.01351786  0.004211587
## 3 -0.08356286  0.007456498  0.06969634  0.068973936  0.11780870 -0.091092165
## 4  0.15952808 -0.198935170  0.05566632  0.002800216 -0.15235668  0.015802877
## 5  0.03295078  0.061982575 -0.06887557 -0.074327321  0.05939462 -0.065458464
## 6 -0.08204684 -0.005612874 -0.07074952  0.018879230  0.03329504  0.176728727
##            V7           V8          V9          V10          V11          V12
```

```
## 1 -0.05059575 -0.19143594  0.04251004 -0.12313234  0.04094018 -0.173321841
## 2  0.13430388  0.11765833 -0.02386471  0.09838956  0.16888733  0.000213186
## 3 -0.02145794 -0.16649724  0.10584830  0.02199248  0.15865884 -0.063030033
## 4 -0.01795565 -0.04635304  0.08864227 -0.14672500 -0.03309078 -0.034096858
## 5 -0.01001907 -0.11159201 -0.06192430  0.05210227 -0.22852355 -0.115657236
## 6  0.07126663 -0.07508190  0.22061025 -0.01587546  0.24976616  0.180314191
##           V13         V14         V15         V16         V17         V18
## 1  0.07073107  0.09510128  0.03981302  0.08936737  0.13079015  0.07395892
## 2  0.10341077 -0.03892372 -0.04075286 -0.10472981  0.14970410 -0.10634574
## 3  0.02234804 -0.02843307  0.13242586  0.19713374  0.08147027  0.02462108
## 4 -0.08787076  0.08574098 -0.07012317 -0.03836321 -0.18697888 -0.02894994
## 5  0.11629646  0.17196273 -0.05806143  0.16541453  0.04820295 -0.22648894
## 6 -0.20001649  0.02700549 -0.10010722  0.15122127  0.04561356 -0.14088505
##           V19          V20         V21          V22          V23          V24
## 1 -0.25923277  0.076258651  0.10744410  0.143506957 -0.043383274 -0.045303708
## 2  0.13140022  0.111143108  0.18956548 -0.071037115  0.177261118  0.216536850
## 3 -0.06355430 -0.092320695 -0.06029973 -0.006506757 -0.001825971  0.124574667
## 4 -0.04299788  0.016434184 -0.03908678 -0.175946874  0.085281499  0.059549803
## 5 -0.01693183  0.115482519 -0.04162220  0.056972297  0.020516290  0.000488445
## 6  0.06122182 -0.005652142 -0.03756574  0.161234680 -0.300804860  0.027936078
##           V25          V26          V27         V28          V29         V30
## 1 -0.03572989  0.007730312 -0.073732753 -0.04184181 -0.121536404 -0.13765192
## 2 -0.11468141 -0.029686864  0.029066665  0.03551355 -0.002255863  0.01676799
## 3 -0.05174205 -0.118324224 -0.088484957  0.05134811  0.070123930  0.15846291
## 4 -0.03621238  0.001129269  0.020800648  0.00186074 -0.058748203  0.16778890
## 5  0.23505543  0.099160104 -0.004773017  0.13184490 -0.060672794  0.04882967
## 6  0.24465314  0.159396745 -0.168452065 -0.00658320  0.109664022  0.08786733
##           V31         V32          V33         V34         V35          V36
## 1 -0.03410670 -0.02555104 -0.243263975  0.03309763  0.10778503 -0.070756823
## 2  0.15024245 -0.17869381 -0.034048493  0.09763275 -0.11989744  0.197157201
## 3  0.05283077  0.17846628  0.071303319 -0.08433399  0.02166370 -0.008999868
## 4  0.05421914  0.17635863 -0.065903739 -0.09705799  0.01430870 -0.001401725
## 5 -0.01366734  0.06896002 -0.003640262 -0.17715313 -0.10657501 -0.112345694
## 6 -0.11367339 -0.11007406 -0.159328630 -0.03224703 -0.04286234 -0.134413012
##            V37         V38          V39         V40          V41          V42
## 1  0.048934096 -0.22451526 -0.001128123 -0.07598457 -0.108690882 -0.017405549
## 2 -0.077891030 -0.13353714  0.061967726  0.11489591 -0.182608301  0.096129056
## 3  0.174355935  0.12827752 -0.128123874 -0.08424763  0.099528181  0.029382666
## 4 -0.007838729  0.06907959 -0.012426133  0.03914133 -0.001186178  0.008099936
## 5 -0.097555379 -0.09670627  0.017574165  0.08913772 -0.059962839  0.018366184
## 6  0.007065982 -0.13457937  0.169277379 -0.13352587 -0.017794799  0.016625504
##           V43         V44          V45         V46         V47          V48
## 1  0.20057186  0.07960927 -0.094061130 -0.15414026  0.07372137 -0.004163922
## 2 -0.20705715  0.09864283  0.063470293  0.01943211  0.23213339  0.067611201
## 3  0.30557424 -0.07945317 -0.006248848  0.02644225  0.03489093  0.086643615
## 4 -0.02613506 -0.03088180  0.018283787 -0.11187352 -0.11339167  0.023517502
## 5 -0.04543933  0.03614448  0.110364102  0.06509530  0.04213353 -0.093397013
## 6  0.01575606  0.13987911  0.175203562 -0.10329002 -0.09245563  0.081325217
##            V49          V50 class
## 1 -0.079506319  0.046365865     1
## 2 -0.001995512  0.007477833     1
## 3 -0.251442512 -0.048683624     1
## 4  0.221095203  0.074891082     1
## 5 -0.148876223  0.046423458     1
```

```
## 6 -0.116075188  0.012942046     1
```

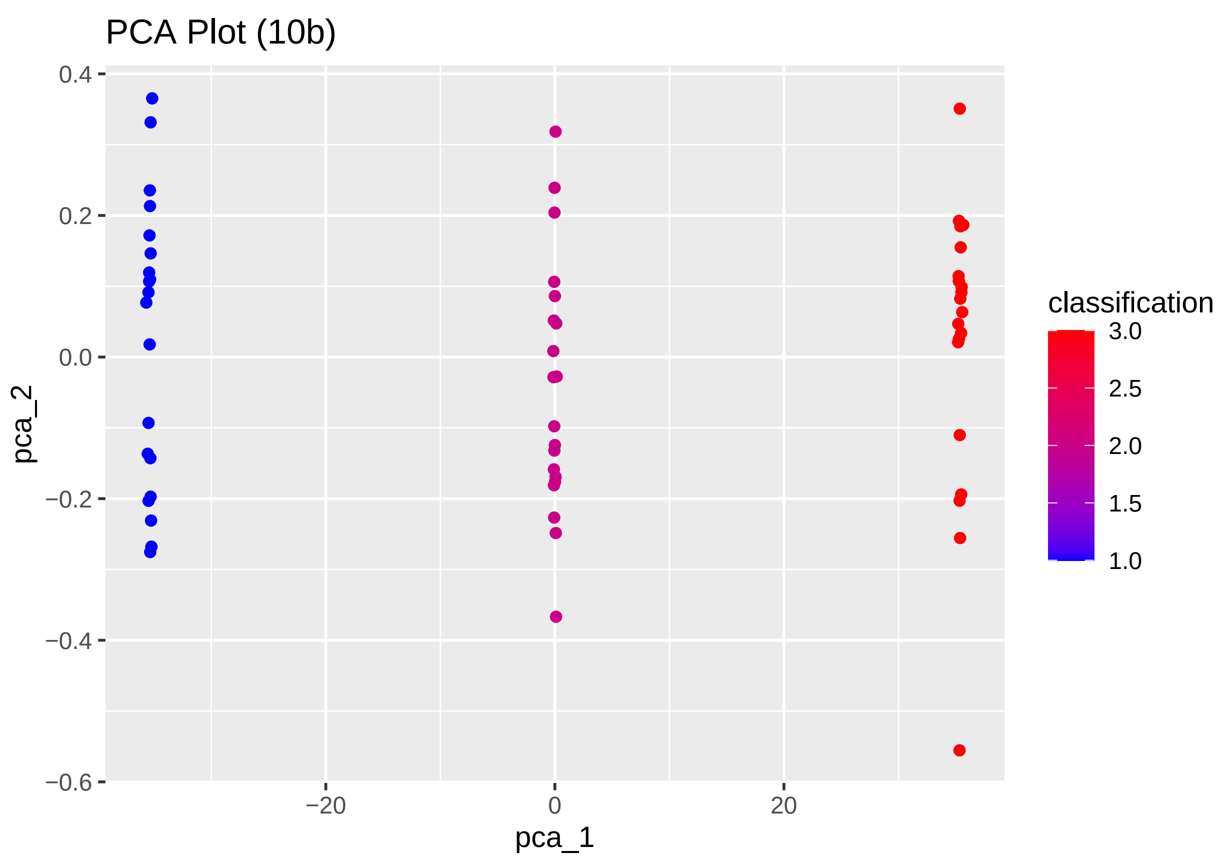
(b) Perform PCA on the 60 observations and plot the first two principal component score vectors. Use a different color to indicate the observations in each of the three classes. If the three classes appear separated in this plot, then continue on to part (c). If not, then return to part (a) and modify the simulation so that there is greater separation between the three classes. Do not continue to part (c) until the three classes show at least some separation in the first two principal component score vectors.

```
PCA_attempt <- prcomp(sim_df)
summary(PCA_attempt)
```

```
## Importance of components:
##                            PC1     PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     29.1309 0.19246 0.18219 0.18039 0.17140 0.16765 0.15956
## Proportion of Variance  0.9994 0.00004 0.00004 0.00004 0.00003 0.00003 0.00003
## Cumulative Proportion   0.9994 0.99943 0.99947 0.99951 0.99954 0.99958 0.99961
##                            PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     0.15313 0.14544 0.14393 0.14227 0.13793 0.13023 0.12795
## Proportion of Variance 0.00003 0.00002 0.00002 0.00002 0.00002 0.00002 0.00002
## Cumulative Proportion  0.99963 0.99966 0.99968 0.99971 0.99973 0.99975 0.99977
##                           PC15    PC16    PC17    PC18    PC19    PC20    PC21
## Standard deviation     0.12105 0.11708 0.11345 0.11203 0.10747 0.10412 0.10063
## Proportion of Variance 0.00002 0.00002 0.00002 0.00001 0.00001 0.00001 0.00001
## Cumulative Proportion  0.99979 0.99980 0.99982 0.99983 0.99985 0.99986 0.99987
##                           PC22    PC23    PC24    PC25    PC26    PC27    PC28
## Standard deviation     0.09908 0.09449 0.09211 0.08943 0.08447 0.08415 0.08371
## Proportion of Variance 0.00001 0.00001 0.00001 0.00001 0.00001 0.00001 0.00001
## Cumulative Proportion  0.99988 0.99989 0.99990 0.99991 0.99992 0.99993 0.99994
##                           PC29    PC30    PC31    PC32    PC33    PC34    PC35
## Standard deviation     0.08032 0.07445 0.07289 0.06808 0.06442 0.06273 0.06229
## Proportion of Variance 0.00001 0.00001 0.00001 0.00001 0.00000 0.00000 0.00000
## Cumulative Proportion  0.99994 0.99995 0.99996 0.99996 0.99997 0.99997 0.99998
##                          PC36    PC37    PC38    PC39    PC40    PC41    PC42
## Standard deviation     0.0546 0.05222 0.05059 0.04735 0.04369 0.04242 0.03693
## Proportion of Variance 0.0000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000
## Cumulative Proportion  1.0000 0.99998 0.99999 0.99999 0.99999 0.99999 1.00000
##                           PC43    PC44    PC45    PC46    PC47    PC48    PC49
## Standard deviation     0.03217 0.02907 0.02682 0.01972 0.01724 0.01586 0.01154
## Proportion of Variance 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000
## Cumulative Proportion  1.00000 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000
##                           PC50     PC51
## Standard deviation     0.01053 0.001167
## Proportion of Variance 0.00000 0.000000
## Cumulative Proportion  1.00000 1.000000
```

```
# Plot
PCA_ggplot <- ggplot(data.frame(
  pca_1 = PCA_attempt$x[,1],
  pca_2 = PCA_attempt$x[,2],
  classification = sim_df$class),
  aes(pca_1, pca_2, col = classification))
```

```r
PCA_ggplot +
  geom_point() +
  labs(title = "PCA Plot (10b)") +
  scale_color_gradient(low = "blue", high = "red")
```




```r
# There is clear separation between the classes -> move on to part (c)
```

(c) Perform K-means clustering of the observations with $K = 3$. How well do the clusters that you obtained in K-means clustering compare to the true class labels? Hint: You can use the table() function in R to compare the true class labels to the class labels obtained by clustering. Be careful how you interpret the results: K-means clustering will arbitrarily number the clusters, so you cannot simply check whether the true class labels and clustering labels are the same.

```r
K <- 3
set.seed(1)
k_means <- kmeans(sim_df, centers = K)
k_means_table <- table(k_means$cluster, c(rep(1,20), rep(2,20), rep(3,20)))
k_means_table # careful on this interpretation so best to look at the cluster
```

```
##
##       1  2  3
##   1   0  0 20
##   2  20  0  0
##   3   0 20  0
```

```r
k_means$cluster
```

```
##  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [39] 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

COMMENTS: We can see that the K-means clustering will arbitrarily number the clusters. Therefore, each cluster will be assigned to one class only. From the results, we can see that the K-Means Clustering performed exactly how it should, where each cluster is classified as one class only.

(d) Perform K-means clustering with K = 2. Describe your results.

```r
K <- 2
set.seed(1)
k_means <- kmeans(sim_df, centers = K)
k_means_table <- table(k_means$cluster, c(rep(1,20), rep(2,20), rep(3,20)))
k_means_table # careful on this interpretation so best to look at the cluster
```

```
##
##      1  2  3
##   1  0 20 20
##   2 20  0  0
```

```r
k_means$cluster
```

```
##  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [39] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

COMMENTS: We can see here that 2 of the classifications are assigned to 2 clusters. However, we can see that the 3rd classification has to be forced to go into one of the two clusters available because of the 2 classes for the K-Means (K = 2). For this case, the classification 3 went into class 1 K-Means cluster.

(e) Now perform K-means clustering with K = 4, and describe your results.

```r
K <- 4
set.seed(1)
k_means <- kmeans(sim_df, centers = K)
k_means_table <- table(k_means$cluster, c(rep(1,20), rep(2,20), rep(3,20)))
k_means_table # careful on this interpretation so best to look at the cluster
```

```
##
##      1  2  3
##   1  0  0 20
##   2 11  0  0
##   3  0 20  0
##   4  9  0  0
```

```r
k_means$cluster
```

```
##  [1] 4 2 4 2 4 4 2 4 2 4 2 4 2 2 4 4 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [39] 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

COMMENTS: For this case, the classification 2 gets split between cluster 2 and 4. For classification 1 and 3, they are assigned to their own clusters. We happened to split of the 3 classifications (specifically, classification 2) into one of the 4 clusters.

(f) Now perform K-means clustering with K = 3 on the first two principal component score vectors, rather than on the raw data. That is, perform K-means clustering on the 60 × 2 matrix of which the first column is the first principal component score vector, and the second column is the second principal component score vector. Comment on the results.

```
K <- 3
set.seed(1)
k_means_pca <- kmeans(PCA_attempt$x[, 1:2], centers = K)
k_means_pca_table <- table(k_means_pca$cluster,
                           c(rep(1,20), rep(2,20), rep(3,20)))
k_means_pca_table # careful on this interpretation so best to look at the cluster
```

```
##
##      1  2  3
##   1  0  0 20
##   2 20  0  0
##   3  0 20  0
```

```
k_means_pca$cluster
```

```
##  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [39] 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

COMMENTS: We see that it is a perfect match as part (c). All the classes are given each a single cluster only.

(g) Using the scale() function, perform K-means clustering with K = 3 on the data after scaling each variable to have standard deviation one. How do these results compare to those obtained in (b)? Explain.

```
K <- 3
set.seed(1)
k_means_scaled <- kmeans(scale(sim_df), centers = K)
k_means_scaled_table <- table(k_means_scaled$cluster,
                           c(rep(1,20), rep(2,20), rep(3,20)))
k_means_scaled_table # careful on this interpretation so best to look at the cluster
```

```
##
##      1  2  3
##   1  0  0 20
##   2 20  0  0
##   3  0 20  0
```

```
k_means_scaled$cluster
```

```
##  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [39] 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

COMMENTS: Just like parts (c) and (f), we can see that we get it exactly right again. The results are what we would expect from part (b). Each classification is assigned to a single cluster only.