Stats 101C HW8

Anna Piskun

12/10/2020

Problem 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.

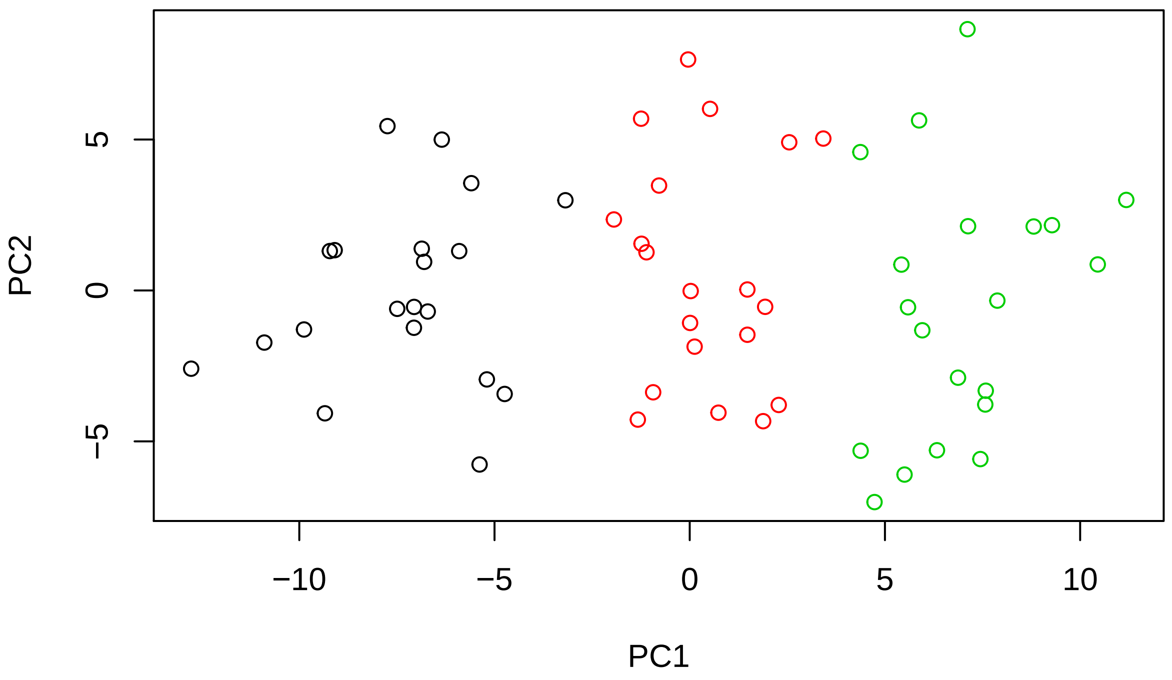
```r
set.seed(123)
x <- matrix(rnorm(20*50, mean = 0, sd = 2), nrow = 20)
y <- matrix(rnorm(20*50, mean = 1, sd = 2), nrow = 20)
z <- matrix(rnorm(20*50, mean = 2, sd = 2), nrow = 20)
data <- rbind(x,y,z)
```

(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.

```r
# We first conduct PCA to reduce the dimension of predictors.
data.pc.sc <- prcomp(data, cor = FALSE)
```

```
## Warning: In prcomp.default(data, cor = FALSE) :
##  extra argument 'cor' will be disregarded
```

```r
data.pc.sc <- data.pc.sc$x
class <- c(rep(1,20), rep(2,20), rep(3,20))
plot(data.pc.sc[,1:2], col= class)
```



(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.

```
# We fit the K-means Clustering.
# iter.max is the maximum number of iterations to try.
# nstart is the number of repetitions of the whole algorithm.
set.seed(123)
k.means.fit <- kmeans(data, centers = 3, nstart = 20,
                      iter.max = 10)
print(k.means.fit)
```

```
## K-means clustering with 3 clusters of sizes 19, 19, 22
##
## Cluster means:
##        [,1]       [,2]      [,3]       [,4]      [,5]       [,6]       [,7]
## 1 2.4219048  2.2910042 0.7527699  1.0224741 1.0960182  1.2491035  1.1629159
## 2 0.1693047 -0.1528008 0.1975143 -0.2007695 0.6850938 -0.6960277 -0.2327931
## 3 1.1716277  1.5591214 2.1710807  1.3183571 1.0987141  1.4644352  1.6749370
##         [,8]      [,9]      [,10]     [,11]      [,12]        [,13]     [,14]
## 1  0.4182156 1.0793802  0.6949150 0.5684394  1.1028825  1.621277879 1.5157407
## 2 -0.4478384 0.3740187 -0.3037477 0.1126754 -0.2865372 -0.009391583 0.3520651
## 3  1.9350751 0.9545456  2.3990103 2.2670748  1.5225506  0.876584827 1.8643893
##       [,15]      [,16]        [,17]      [,18]      [,19]      [,20]      [,21]
## 1 1.2848498  1.0005462  0.909982394  1.0195999  1.7604693  0.2866351  1.6709859
## 2 0.7084702 -0.2190971 -0.001390887 -0.2631721 -0.1942163 -0.2406180 -0.4980516
## 3 1.2135367  2.0034008  1.967828176  1.1046388  2.2662169  2.5762426  0.6377182
##       [,22]      [,23]     [,24]     [,25]      [,26]     [,27]     [,28]
## 1 0.5426070  1.5689756 0.5720755 1.6502270  0.8838032 1.7982759 1.4945078
## 2 0.4549682 -0.2272161 0.4262305 0.5791579 -0.5695415 0.4671659 0.1237334
## 3 1.9611966  2.1926771 1.8435170 2.6420244  2.6403283 1.6483413 1.3598896
##        [,29]      [,30]      [,31]      [,32]      [,33]      [,34]      [,35]
## 1  1.2194907  0.1523239  1.3552354  1.8981165 -0.2000468  2.0613357  0.6286079
## 2 -0.1868564 -0.2275952 -0.5282567 -0.3519146  0.2803192 -0.2860871 -0.3078460
## 3  2.2015165  2.2364729  1.6464098  2.0790260  2.2805274  1.1375724  0.9673041
##        [,36]       [,37]     [,38]     [,39]      [,40]     [,41]      [,42]
## 1 1.01958359  0.78148347 1.2071599 0.4540627  1.5124073 0.1161220  1.0915690
## 2 0.02513985 -0.01300186 0.6938293 0.4324791 -0.3465803 0.3034525 -0.6195495
## 3 1.63049543  2.22062164 1.7450522 2.3746942  1.7919539 2.4055741  2.0615332
##       [,43]     [,44]     [,45]     [,46]        [,47]      [,48]      [,49]
## 1 0.4735349 1.8564344 1.0977315 1.4558371 -0.006904823  1.9834207  1.4888433
## 2 0.5877309 0.9194712 0.1120535 0.2401107  0.242531084 -0.4372726 -0.6783598
## 3 2.6220856 1.5354112 1.7586586 1.6916595  1.579510001  2.5740087  2.0398927
##        [,50]
## 1 2.03689708
## 2 0.07796817
## 3 1.57897424
##
## Clustering vector:
##  [1] 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 1 1 1 3 3 1 1 1 3 1 1 1 3 1 1 1 1 3
## [39] 1 1 3 1 1 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3
##
## Within cluster sum of squares by cluster:
## [1] 3715.794 3499.218 4007.762
##  (between_SS / total_SS =  17.0 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
table(k.means.fit$cluster, class)
```

```
##    class
##      1  2  3
##   1  1 15  3
##   2 19  0  0
##   3  0  5 17
```

The clusters obtained using K-means clustering are all exactly clustered.

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

```
set.seed(123)
k.means.fit <- kmeans(data, centers = 2, nstart = 20,
                      iter.max = 10)
print(k.means.fit)
```

```
## K-means clustering with 2 clusters of sizes 32, 28
##
## Cluster means:
##        [,1]     [,2]      [,3]      [,4]      [,5]        [,6]       [,7]
## 1 1.8977620 2.096744 1.7953329 1.2924341 1.0561941  1.36599528 1.65848267
## 2 0.5100145 0.279669 0.2988758 0.1163697 0.8648081 -0.03520832 0.05176793
##          [,8]      [,9]      [,10]      [,11]    [,12]     [,13]     [,14]
## 1  1.39306450 1.1066881 1.81052900 1.94293322 1.378045 1.3462623 1.7911984
## 2 -0.09175877 0.4714487 0.08119563 0.02296298 0.175330 0.2439398 0.6852331
##       [,15]      [,16]     [,17]       [,18]      [,19]     [,20]     [,21]
## 1 1.1799047 1.80823893 1.4687678  1.25900541 2.45764120 1.5724310 0.9719382
## 2 0.9576406 0.03781083 0.4841032 -0.05764249 0.03468071 0.2583526 0.1861975
##        [,22]     [,23]     [,24]     [,25]      [,26]     [,27]     [,28]
## 1 1.88597082 1.9976400 1.5169016 2.4261598  2.1242893 1.7755168 1.4137772
## 2 0.06247098 0.3502802 0.3922977 0.8159192 -0.1399665 0.8032273 0.5508316
##         [,29]      [,30]        [,31]      [,32]     [,33]     [,34]
## 1 2.118076722  1.9120250  1.630588371 2.31899146 1.4638231 1.5960307
## 2 0.009819914 -0.4790196 -0.008757753 0.03245292 0.1733728 0.2744048
##         [,35]     [,36]     [,37]    [,38]     [,39]      [,40]     [,41]
## 1  0.92731277 1.0259336 1.7512691 1.597216 1.7339157 1.88035890 1.6933474
## 2 -0.08210156 0.8175275 0.2647935 0.835680 0.4857951 0.05007912 0.2395511
##        [,42]    [,43]    [,44]     [,45]     [,46]     [,47]    [,48]
## 1  2.0165348 1.523212 1.722147 1.4768422 1.5868653 1.1104394 2.563859
## 2 -0.3645361 1.039541 1.121877 0.5149091 0.6664224 0.1318593 0.141483
##        [,49]     [,50]
## 1  2.2262750 1.9328499
## 2 -0.3915705 0.4667385
##
## Clustering vector:
##  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 2 1 2 1 1 1 1 2 1 2 2 1 2 1
## [39] 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##
## Within cluster sum of squares by cluster:
## [1] 6352.580 5475.344
##  (between_SS / total_SS =  12.5 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```r
table(k.means.fit$cluster, class)
```

```
##    class
##      1  2  3
##   1  0 12 20
##   2 20  8  0
```

The outside classes remain classified correctly while the middle class is forced together. Thus, performing K-means clustering with K = 2 results in the loss of a class.

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

```r
set.seed(123)
k.means.fit <- kmeans(data, centers = 4, nstart = 20,
                      iter.max = 10)
print(k.means.fit)
```

```
## K-means clustering with 4 clusters of sizes 14, 19, 16, 11
##
## Cluster means:
```

```
##        [,1]       [,2]      [,3]       [,4]      [,5]       [,6]       [,7]
## 1 1.9463630  2.5643710 2.1120076  1.1203551 1.0384971  1.5573482  0.9190038
## 2 0.1693047 -0.1528008 0.1975143 -0.2007695 0.6850938 -0.6960277 -0.2327931
## 3 1.5197658  1.8523391 1.2912076  1.1727499 0.6594305  1.5964125  2.9124484
## 4 1.8387879  1.1173756 1.0762704  1.2710812 1.8096553  0.7822789 -0.0473827
##         [,8]      [,9]      [,10]     [,11]      [,12]        [,13]     [,14]
## 1  1.1749164 1.7584103  1.9995896 1.8398339  2.0230627  1.431774791 1.4075847
## 2 -0.4478384 0.3740187 -0.3037477 0.1126754 -0.2865372 -0.009391583 0.3520651
## 3  1.1289653 0.4478632  2.2121323 1.9245056  1.3392194  1.105790689 1.9283595
## 4  1.4550430 0.8840611  0.2357491 0.3751119  0.4273175  1.122877072 1.7505183
##       [,15]      [,16]        [,17]      [,18]      [,19]      [,20]      [,21]
## 1 1.7732017  2.6434643  2.779567981  1.5632626  2.9509240  2.5790303  0.2775035
## 2 0.7084702 -0.2190971 -0.001390887 -0.2631721 -0.1942163 -0.2406180 -0.4980516
## 3 0.4835673  0.7742776  1.352324380 -0.2749889  1.7707309  1.1273189  1.9004868
## 4 1.6861867  1.2443866  0.002794867  2.3807816  1.2419144  0.7254435  1.0441541
##       [,22]      [,23]     [,24]     [,25]      [,26]     [,27]      [,28]
## 1 1.4990442  1.9696141 2.2553383 2.5474662  2.4986543 2.5780305  2.9489482
## 2 0.4549682 -0.2272161 0.4262305 0.5791579 -0.5695415 0.4671659  0.1237334
## 3 1.4489615  1.3471348 0.8289335 1.9428516  1.7055089 0.9052001 -0.1536834
## 4 0.8441688  2.6291526 0.5990125 2.0662452  1.1463802 1.8050112  1.7715344
##        [,29]       [,30]      [,31]      [,32]      [,33]      [,34]
## 1  2.6773945  2.04982111  1.2857404  2.5877269  2.3177437  1.7687350
## 2 -0.1868564 -0.22759516 -0.5282567 -0.3519146  0.2803192 -0.2860871
## 3  1.2193908  1.49349952  2.1405276  1.7604045  1.6402929  1.2401887
## 4  1.3281736 -0.04517531  0.8837893  1.5825580 -1.1202168  1.7806056
##         [,35]      [,36]       [,37]     [,38]      [,39]      [,40]      [,41]
## 1  0.93162050 2.17092751  2.53057767 2.1349563  2.3301212  1.8389254  2.2763490
## 2 -0.30784603 0.02513985 -0.01300186 0.6938293  0.4324791 -0.3465803  0.3034525
## 3 -0.05930267 0.95163805  0.34292138 1.0911315  1.9588601  2.0579885  1.6698759
## 4  1.92094504 0.87489034  2.07154841 1.2708814 -0.2811814  0.8623593 -0.3143592
##        [,42]     [,43]     [,44]     [,45]     [,46]      [,47]      [,48]
## 1  1.4910762 2.4574612 2.5334607 2.1583859 1.8943848  1.0732865  2.6627658
## 2 -0.6195495 0.5877309 0.9194712 0.1120535 0.2401107  0.2425311 -0.4372726
## 3  2.3072658 1.7413359 0.7311855 1.6869371 0.4277343  1.5583906  1.8728460
## 4  0.7547476 0.4015650 1.9894438 0.2126356 2.8647525 -0.4856576  2.4608117
##        [,49]      [,50]
## 1  3.5070634 2.54933664
## 2 -0.6783598 0.07796817
## 3  0.3142056 1.86512160
## 4  1.7308624 0.71871086
##
## Clustering vector:
##  [1] 2 2 2 2 2 2 2 2 2 2 4 2 2 2 2 2 2 2 2 2 4 4 3 1 3 3 3 3 3 4 4 3 3 4 4 4 4 3
## [39] 3 4 3 1 3 1 3 3 1 1 1 3 1 4 1 3 1 1 1 1 1 1
##
## Within cluster sum of squares by cluster:
## [1] 2334.096 3499.218 3076.398 1833.296
##  (between_SS / total_SS =  20.6 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```r
table(k.means.fit$cluster, class)
```

```
##    class
##      1  2  3
##   1  0  1 13
##   2 19  0  0
##   3  0 10  6
##   4  1  9  1
```

Performing K-means clustering with K=4, we find that a class is actually added. Now, we have 4 classes. However, each cluster still consists of 20 observations.

(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.

```r
set.seed(123)
k.means.fit <- kmeans(data[,1:2], centers = 3, nstart = 20,
                      iter.max = 10)
print(k.means.fit)
```

```
## K-means clustering with 3 clusters of sizes 22, 17, 21
##
## Cluster means:
##          [,1]       [,2]
## 1 -0.09863337 -0.8053871
## 2  3.98322839  2.3090925
## 3  0.45065912  2.5424038
##
## Clustering vector:
##  [1] 1 1 1 1 1 1 3 1 1 3 2 1 3 3 3 2 3 1 1 1 1 1 3 3 1 2 3 2 2 3 2 2 1 1 3 1 3 3
## [39] 3 2 3 2 3 2 2 1 3 3 2 1 1 2 2 2 1 2 3 3 3 2
##
## Within cluster sum of squares by cluster:
## [1] 105.42882  85.23154  54.42815
##  (between_SS / total_SS =  57.2 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```r
table(k.means.fit$cluster, class)
```

```
##    class
##      1  2  3
##   1 12  6  4
##   2  2  6  9
##   3  6  8  7
```

This K-means clustering performs the same, telling us that the first two principal component score vectors hold enough significance to represent the entire data set.

(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.

```r
set.seed(123)
k.means.fit <- kmeans(scale(data), centers = 3, nstart = 20,
                      iter.max = 10)
print(k.means.fit)
```

```
## K-means clustering with 3 clusters of sizes 19, 22, 19
##
## Cluster means:
##          [,1]       [,2]       [,3]       [,4]        [,5]       [,6]
## 1 -0.46668355 -0.6726775 -0.5040389 -0.4793619 -0.15240232 -0.7132115
## 2 -0.03390266  0.1489485  0.6018924  0.2917433  0.07130115  0.3810548
## 3  0.50593926  0.5002108 -0.1928892  0.1415538  0.06984310  0.2719901
##         [,7]       [,8]        [,9]      [,10]      [,11]      [,12]
## 1 -0.5019966 -0.5567247 -0.23087335 -0.6296219 -0.4191492 -0.5283737
## 2  0.3369824  0.5988883  0.07637267  0.6721259  0.5473949  0.3379921
## 3  0.1118065 -0.1367249  0.14244183 -0.1486292 -0.2146764  0.1370145
##         [,13]      [,14]       [,15]        [,16]       [,17]      [,18]
## 1 -0.45017889 -0.5661490 -0.18446160 -0.544195366 -0.43731442 -0.4271936
## 2  0.02394199  0.3614630  0.06890398  0.462745385  0.41477992  0.2165137
## 3  0.42245658  0.1476129  0.10467805  0.008384921 -0.04295707  0.1764934
##        [,19]      [,20]       [,21]      [,22]      [,23]      [,24]      [,25]
## 1 -0.6460463 -0.5533353 -0.58128482 -0.2539692 -0.6675228 -0.3022667 -0.5143450
## 2  0.3989272  0.7457582  0.01710119  0.4055340  0.4418426  0.4548137  0.4541356
## 3  0.1841306 -0.3101742  0.56148344 -0.2155965  0.1559156 -0.2243598 -0.0114963
##         [,26]      [,27]      [,28]       [,29]      [,30]      [,31]
## 1 -0.79032308 -0.4797820 -0.4087492 -0.68399356 -0.4172793 -0.5572526
## 2  0.75919325  0.1833308  0.1606833  0.55259446  0.5870236  0.3121862
## 3 -0.08874279  0.2675043  0.2226949  0.04414734 -0.2624322  0.1957738
##        [,32]      [,33]       [,34]       [,35]       [,36]      [,37]
## 1 -0.6572862 -0.2805163 -0.59729970 -0.34541034 -0.42295541 -0.4977109
## 2  0.3389535  0.6847288  0.07472411  0.23102042  0.32852837  0.5406948
## 3  0.2648138 -0.5123276  0.51077705  0.07791301  0.04255413 -0.1283568
##         [,38]      [,39]      [,40]      [,41]       [,42]      [,43]
## 1 -0.24552672 -0.3398772 -0.7048833 -0.3670484 -0.59192699 -0.3105974
## 2  0.22546207  0.5782489  0.3931700  0.7174595  0.44878822  0.5796446
## 3 -0.01553463 -0.3296742  0.2496338 -0.4636942  0.07227747 -0.3605700
##         [,44]       [,45]      [,46]      [,47]      [,48]      [,49]
## 1 -0.22444211 -0.48775284 -0.3915160 -0.2057108 -0.8139696 -0.6549995
## 2  0.04011238  0.38914211  0.2280823  0.4630777  0.4962914  0.4029221
## 3  0.17799619  0.03716724  0.1274207 -0.3304845  0.2393163  0.1884582
##        [,50]
## 1 -0.4938866
## 2  0.1393491
## 3  0.3325349
##
## Clustering vector:
##  [1] 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 3 3 3 2 2 3 3 3 2 3 3 3 2 3 3 3 3 2
## [39] 3 3 2 3 3 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2
##
## Within cluster sum of squares by cluster:
```

```
## [1] 775.7771 860.5522 812.2476
##  (between_SS / total_SS =  17.0 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```r
table(k.means.fit$cluster, class)
```

```
##    class
##      1  2  3
##   1 19  0  0
##   2  0  5 17
##   3  1 15  3
```

These results are the same as in both (b) and (c).