Chapter 10 Problem 10

Andira Putri

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

- Background on Principal Component Analysis (PCA)

When you have a large number of predictors that are closely related to each other, principal components regression aims to summarize this set with a smaller number of representative variables. These representative variables are special in that they are somehow able to capture the variability in the data set. Think about it this way. . . each of the n observations lives in a p-dimensional space, but not all dimensions are interesting. In our case, "interest" is measured by how much the data varies in that dimension. PCR finds the most interesting dimensions for us so we're not looking at all p-predictors; the dimension of the problem is reduced as a result.

Here is a link to a beautiful visualization of principal components analysis (PCA). I found this incredibly helpful: http://setosa.io/ev/principal-component-analysis/

- Background on K-means clustering

In unsupervised learning, clustering methods aim to find subgroups (or clusters) in the data in hopes of discovering some structure. The partitions in the data set are created by assigning similar observations to one cluster and different observations in other subgroups. What it means for observations to be "similar" or "different" is domain specific– you need to have a good understanding of the data's context. There are two main clustering methods; K-means and hierarchical clustering. K-means partitions a data set into K distinct, non-overlapping clusters.

Algorithm

1. Randomly assign a number, from 1 to K, to each of the observations. These serve as the initial cluster assignments for the observations.
2. Iterate until the cluster assignments stop changing.

- For each of the K clusters, compute the cluster centroid which is the vector of `p` feature means for the observations in the k-th cluster.
- Assign each observation to the cluster whose centroid is the closest (via Euclidean distance)

K-means finds a local optimum rather than global. With that being said, results do depend on initial cluster assignments. The algorithm should be performed multiple times from different initial configurations. Choose the result that minimizes the Euclidean distance between observations in a cluster (i.e. within-cluster variation).

a.) Generate a simulated data set with 20 observations in each of three classes (i.e 60 observations total), and 50 variables.

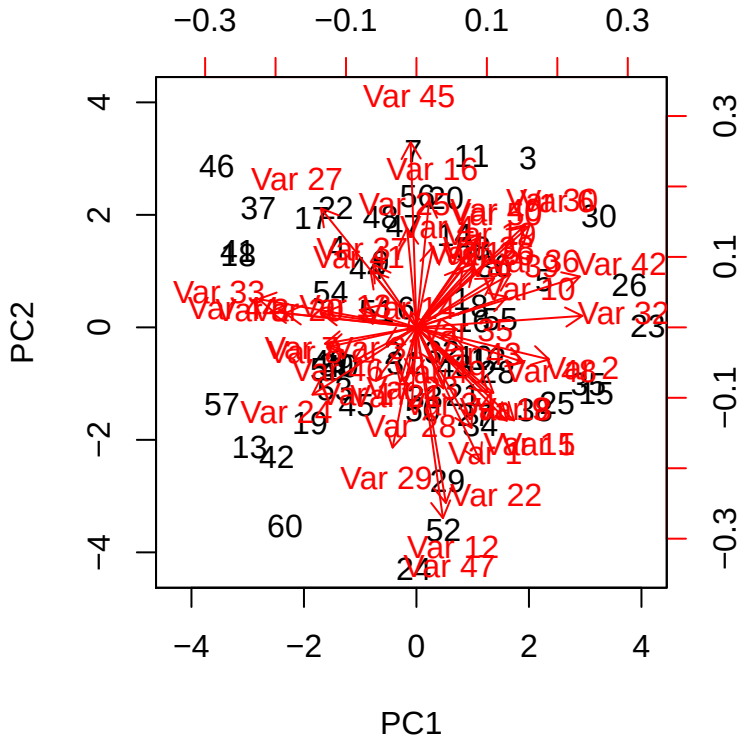
```
set.seed(1)
#initialize matrix
x=matrix(rnorm(20*3*50,mean=0,sd=0.1),ncol=50)
x[1:20, 2] = 1
x[21:40, 1] = 2
x[21:40, 2] = 2
x[41:60, 1] = 1
```

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 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 a greater separation between the three classes.

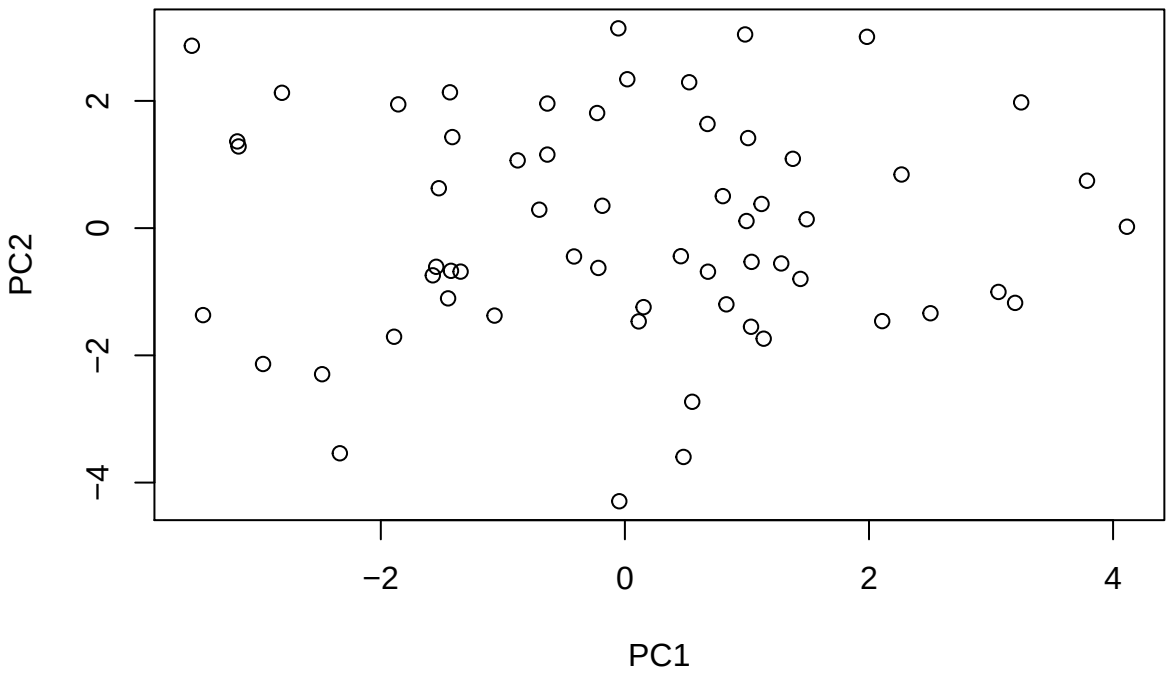
```
#examine variances, 2 denotes column-wise calculation
apply(x,2,var)
```

```
##  [1] 0.667817128 0.672413590 0.010480449 0.011071846 0.009490164
##  [6] 0.009830140 0.009537931 0.012410902 0.013266486 0.009839755
## [11] 0.012440290 0.012513770 0.011639998 0.012145133 0.011104442
## [16] 0.010293132 0.012131292 0.010668270 0.008734649 0.010594258
## [21] 0.011683827 0.010466901 0.007678369 0.009252186 0.010388146
## [26] 0.010483246 0.010881134 0.009395910 0.016213848 0.011332642
## [31] 0.010917383 0.010324702 0.011485788 0.012199831 0.010212300
## [36] 0.013599984 0.011747196 0.008588844 0.009482101 0.009549354
## [41] 0.011238585 0.010833602 0.006312143 0.009066529 0.009720875
## [46] 0.010173614 0.014285215 0.010741227 0.010730774 0.013750959
```

```
pca=prcomp(x,scale=TRUE)
#What are the principal components?
comps=pca$x[,1:2]
#Plot two principal components
biplot(pca,scale=0)
```




```
plot(comps)
```



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?

```
kmeans.3=kmeans(x,3,nstart=20)
#check cluster assignments with kmeans$cluster
#generate table for correct/incorrect observations
table(kmeans.3$cluster, c(rep(1,20), rep(2,20), rep(3,20)))
```

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

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

```
kmeans.2=kmeans(x,2,nstart=20)
kmeans.2$cluster
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [36] 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

e.) Perform K-means clustering with K=4. Describe your results.

```
kmeans.4=kmeans(x,4,nstart=20)
kmeans.4$cluster
```

```
##  [1] 4 1 4 1 4 4 1 4 1 4 1 4 1 1 4 4 1 1 1 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
## [36] 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
```

f.) Perform K-means clustering with K=3 on the first 2 principal component score vectors, rather than on the raw data. Comment on the results.

```
km.pca=kmeans(comps,3,nstart=20)
table(km.pca$cluster, c(rep(1,20), rep(2,20), rep(3,20)))
```

```
##
##      1  2  3
##   1  9  2  8
##   2  3  6  9
##   3  8 12  3
```

g.) Using the `scale()` function, perform K-means clustering wit K=3 on the data fter scaling each variable to have standard deviation 1. How do the results compare to those obtained in (b)? Explain.

```
km.sc=kmeans(scale(x),3,nstart=20)
table(km.sc$cluster, c(rep(1,20), rep(2,20), rep(3,20)))
```

```
##
##      1  2  3
##   1  6 10  1
##   2  9  4  9
##   3  5  6 10
```