Unsupervised Learning Exercises Ch10

Darshan Patel

3/7/2019

The following set of problems are from the applied exercises section in ISLR Chapter 10: Unsupervised Learning.

```
rm(list = ls())
library(MASS)
library(ISLR)
library(tidyverse)
```

```
## Warning: package 'tibble' was built under R version 3.4.4
```

```
## Warning: package 'tidyr' was built under R version 3.4.4
```

```
## Warning: package 'purrr' was built under R version 3.4.4
```

```
## Warning: package 'dplyr' was built under R version 3.4.4
```

```
library(gridExtra)
library(ggdendro)
library(ggfortify)
```

```
## Warning: package 'ggfortify' was built under R version 3.4.4
```

Question 7: In the chapter, the use of correlation-based distance and Euclidean distance as dissimilarity measures for hierarchical clustering was mentioned. It turns out that these two measures are almost equivalent: if each observation has been centered to have mean zero and standard deviation one, and if r_{ij} denotes the correlation between the ith and jth observations, then the quantity $1 - r_{ij}$ is proportional to the squared Euclidean distance between the ith and the jth observations.

On the `USArrests` data, show that this proportionality holds.

Hint: The Euclidean distance can be calculated using the `dist()` function and correlations can be calculated using the `cor()` function.

```
df = USArrests
scaled_df = scale(df)
correlated_df = as.dist(1 - cor(t(scaled_df)))
euclid_df = (dist(scaled_df))^2

summary(correlated_df / euclid_df)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.000086 0.069135 0.133943 0.234193 0.262589 4.887686
```

When dividing the correlation values by the Euclidean distances, the mean proportionality constant is 0.23.

Question 8: A formula for calculating PVE, of the mth principal component is

$$\frac{\sum_{i=1}^{n}\left(\sum_{j=1}^{p}\phi_{jm}x_{ij}\right)^2}{\sum_{j=1}^{p}\sum_{i=1}^{n}x_{ij}^2}$$

The PVE can also be obtained using the `sdev` output of the `prcomp()` function.

On the `USArrests` dataset, calculate PVE in two ways:

(a) Using the `sdev` output of the `prcomp()` function

```
prcomp(df, scale. = TRUE)$sdev^2 / sum(prcomp(df, scale. = TRUE)$sdev^2)
```

```
## [1] 0.62006039 0.24744129 0.08914080 0.04335752
```

(b) By applying the above equation directly. That is, use the `prcomp()` function to compute the principal component loadings. Then, use those loadings in the above equation to obtain the PVE.

```
apply((as.matrix(scale(df)) %*%
         prcomp(df, scale = TRUE)$rotation)^2, 2, sum) /
  sum(scale(df)^2)
```

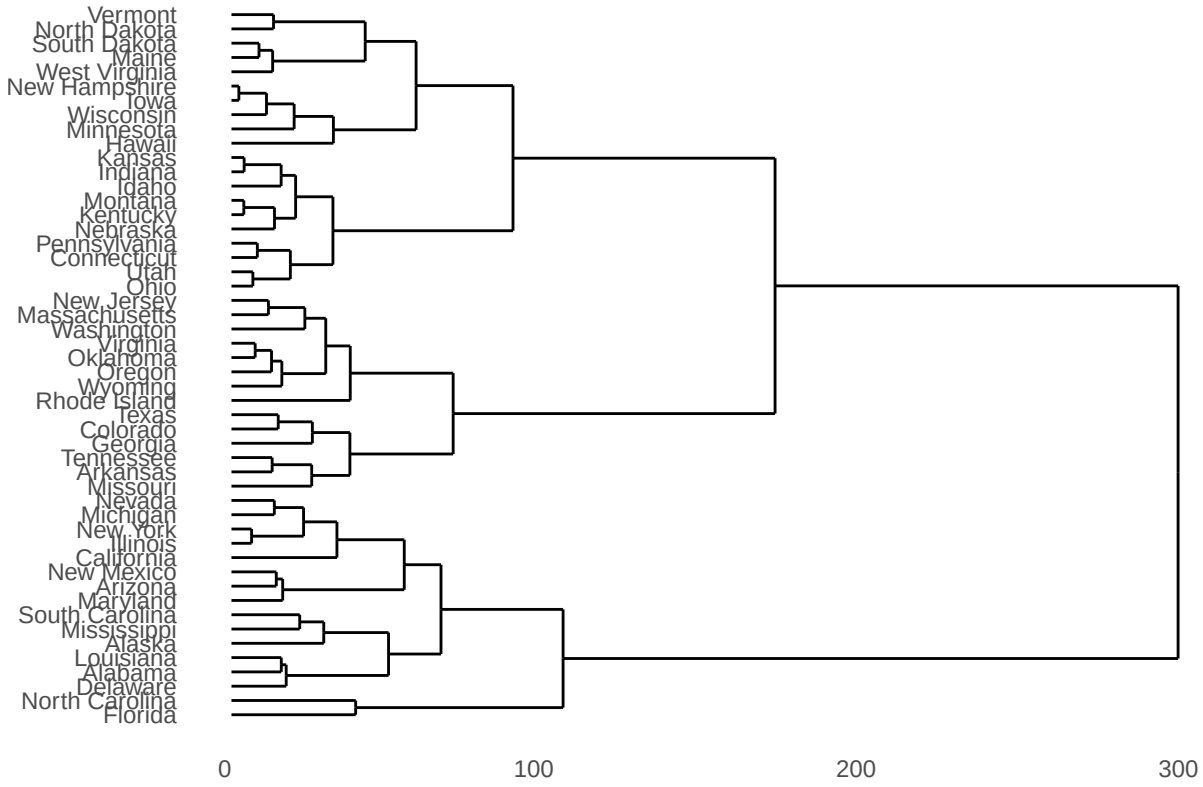
```
##         PC1         PC2         PC3         PC4
## 0.62006039 0.24744129 0.08914080 0.04335752
```

These two approaches give the same results.

Question 9: Consider the `USArrests` dataset. Now perform hierarchical clustering on the states.

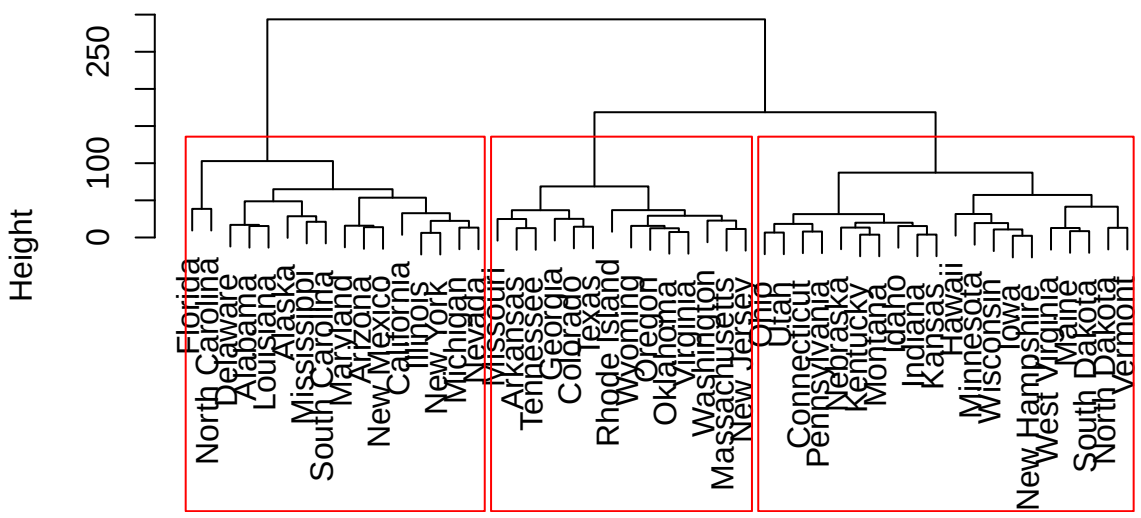
(a) Using hierarchical clustering with complete linkage and Euclidean distance, cluster the states.

```
df = USArrests
df_cluster = hclust(dist(df), method = "complete")
dend_data = dendro_data(df_cluster, type = "rectangle")
ggdendrogram(df_cluster, rotate = TRUE)
```

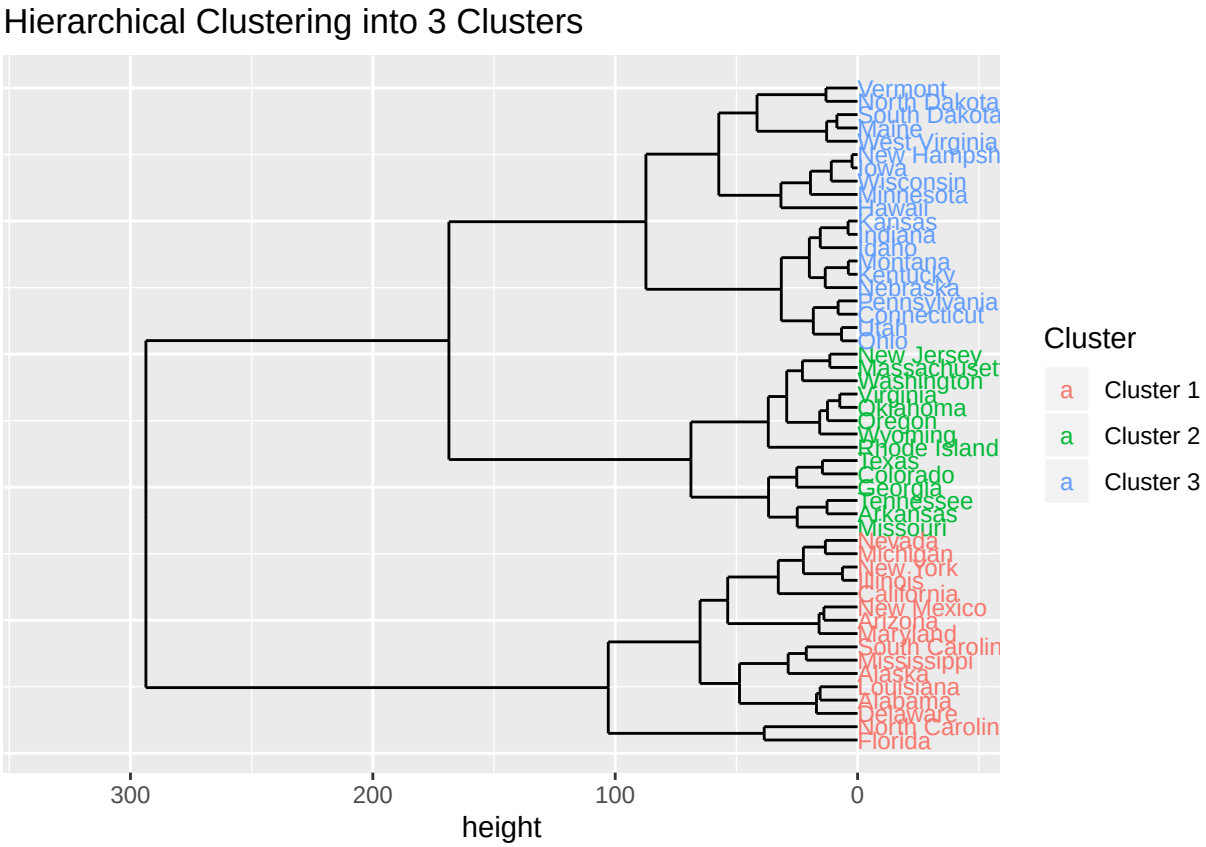


(b) Cut the dendrogram at a height that results in three distinct clusters. Which states belong to which clusters?

```
plot(df_cluster)
g1 = rect.hclust(df_cluster, k = 3)
```



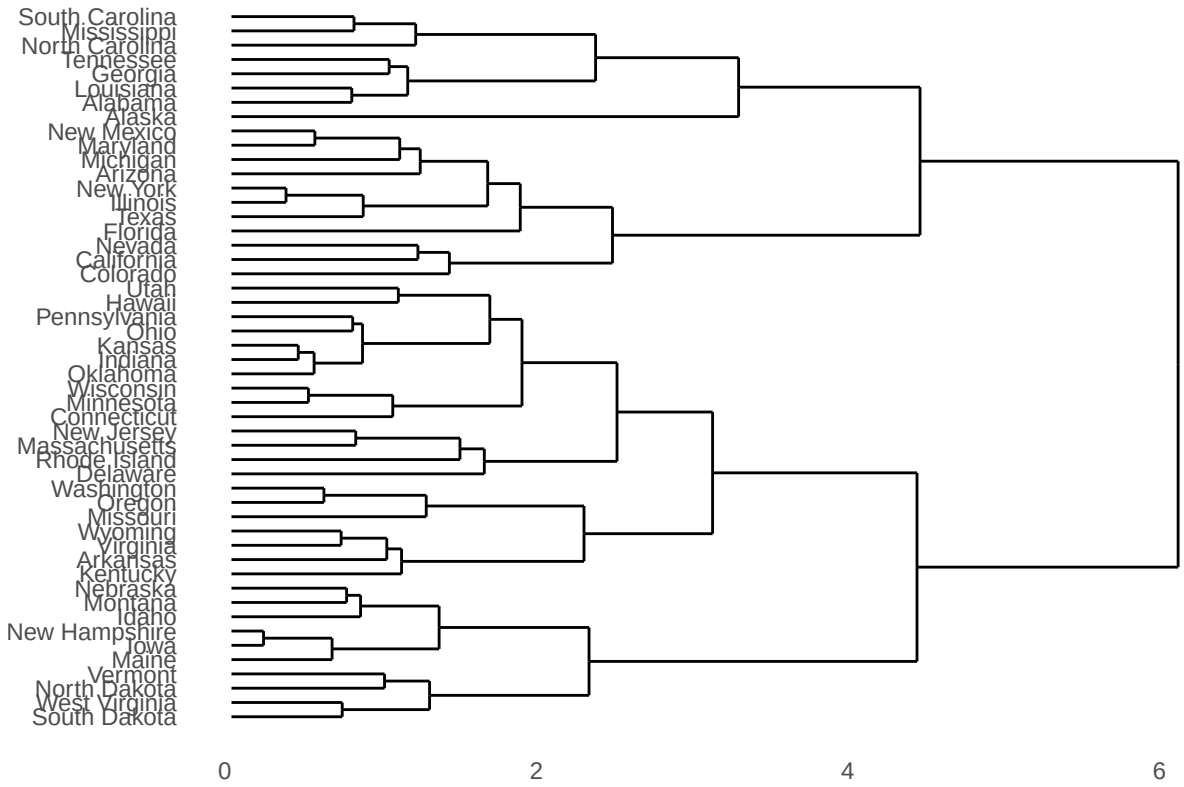

```
clustered_df = data.frame(num = unlist(g1),
                      clust=rep(c("Cluster 1","Cluster 2", "Cluster 3"),
                                times = sapply(g1,length)))
text_df = merge(label(dend_data), clustered_df,
                by.x = "label", by.y = "row.names")
ggplot() +
  geom_segment(data=segment(dend_data),
               aes(x = x, y = y, xend = xend, yend = yend)) +
  geom_text(data=text_df,
            aes(x = x, y = y, label = label, hjust = 0, color = clust),
            size = 3) +
  scale_color_discrete(name = "Cluster") +
  labs(y = "height", x = NULL,
       title = "Hierarchical Clustering into 3 Clusters") +
  coord_flip() + scale_y_reverse(expand = c(0.2, 0)) +
    theme(axis.title.y=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank())
```



The states are divided into three distinct clusters as shown above. Each cluster is represented by a different number from 1 to 3.

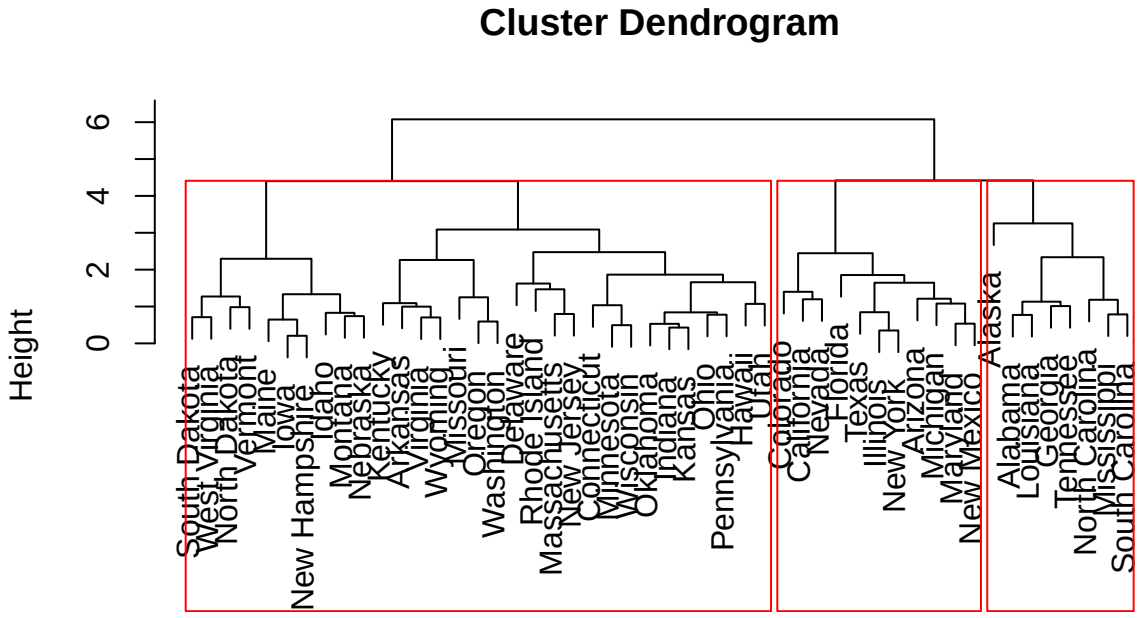
(c) Hierarchically cluster the states using complete linkage and Euclidean distance, *after scaling the variables to have standard deviation one.*

```
df_scaled_cluster = hclust(dist(scale(df)), method = "complete")
dend_data = dendro_data(df_scaled_cluster, type = "rectangle")
ggdendrogram(df_scaled_cluster, rotate = TRUE)
```



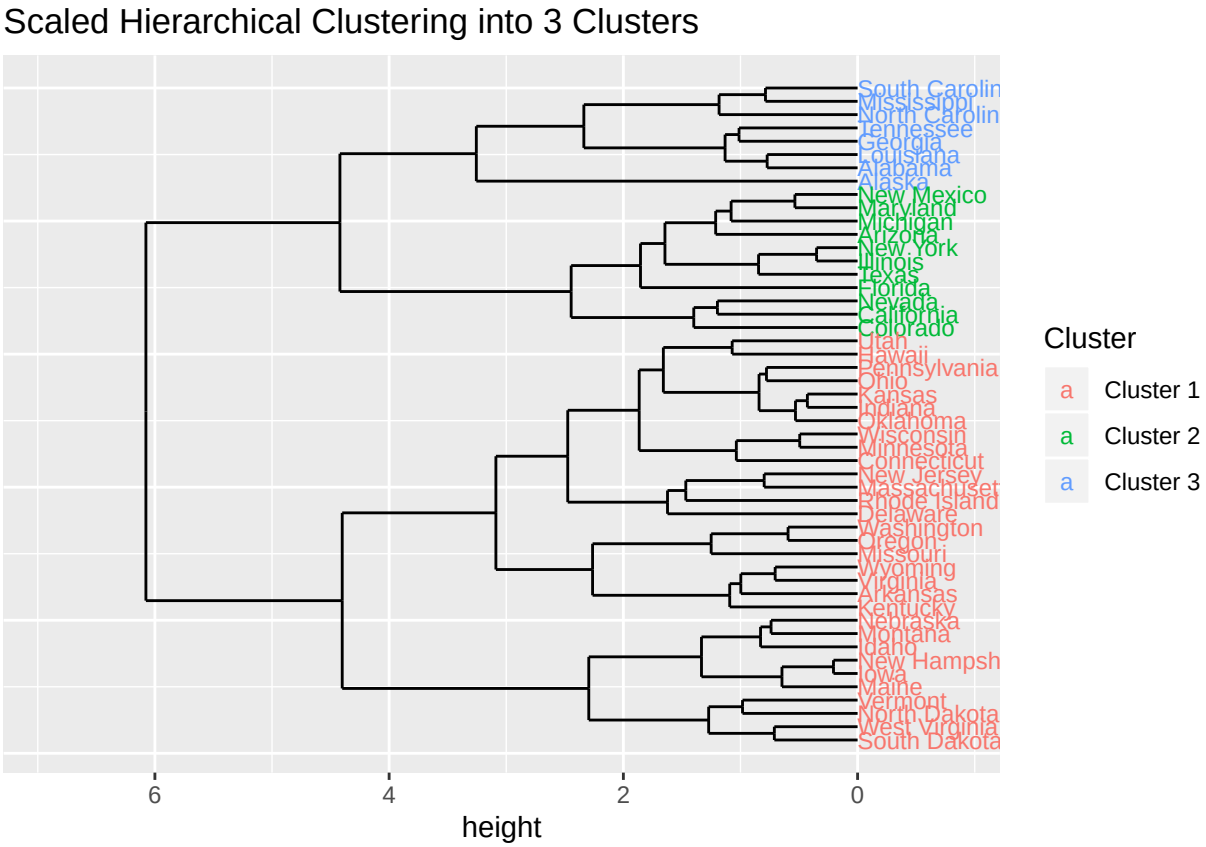
(d) What effect does scaling the variables have on the hierarchical clustering obtained? Should the variables be scaled before the inter-observations dissimilaries are computed? Provide a justification.

```
plot(df_scaled_cluster)
g2 = rect.hclust(df_scaled_cluster, k = 3)
```



dist(scale(df))
hclust (*, "complete")

```
clustered2_df = data.frame(num = unlist(g2),
                      clust=rep(c("Cluster 1","Cluster 2", "Cluster 3"),
                                times = sapply(g2,length)))
text_df = merge(label(dend_data), clustered2_df,
                by.x = "label", by.y = "row.names")
ggplot() +
  geom_segment(data=segment(dend_data),
               aes(x = x, y = y, xend = xend, yend = yend)) +
  geom_text(data=text_df,
            aes(x = x, y = y, label = label, hjust = 0, color = clust),
            size = 3) +
  scale_color_discrete(name = "Cluster") +
  labs(y = "height", x = NULL,
       title = "Scaled Hierarchical Clustering into 3 Clusters") +
  coord_flip() + scale_y_reverse(expand = c(0.2, 0)) +
    theme(axis.title.y=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank())
```



Scaling the observations affects the clusters. Now there appears to be more states in one of the clusters than the rest. The variables should be scaled between inter-observations dissimilarities are computed so that everything is equally scaled and so computations do not give outrageous results. When everything is on the same scale and units, results are more meaningful.

Question 10: In this problem, generate data and then perform PCA and K-means clustering on the dataset.

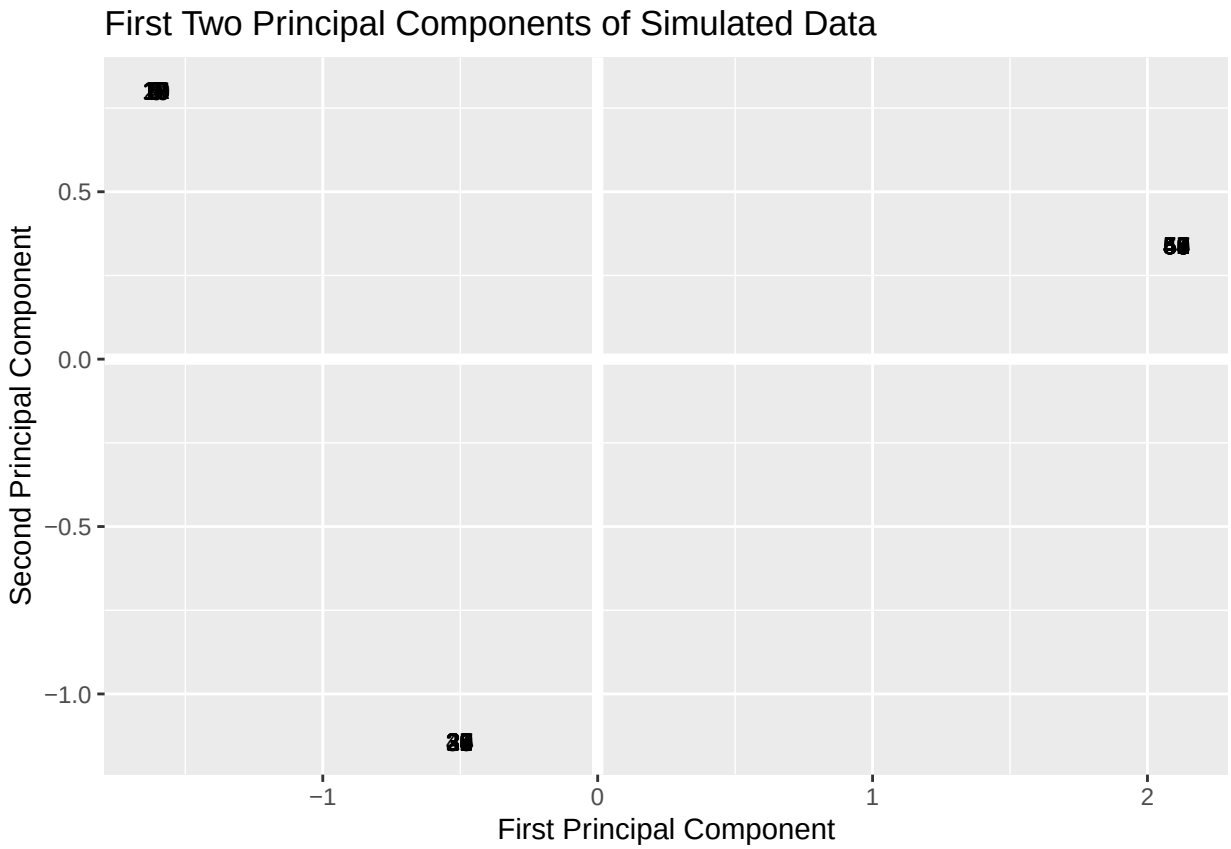
(a) Generate a simulated dataset with 20 observations in each of three classes (i.e. 60 observations total), and 50 variables.

```
set.seed(10)
X = matrix(rnorm(60 * 50, mean = 0, sd = 0.001), ncol = 50, nrow = 60)
X[1:20, 1] = 1
X[21:40, 2] = 2
X[41:60, 2] = 2
X[41:60, 3] = 3
Y = c(rep(1, 20), rep(2, 20), rep(3, 20))
```

(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 classses show at least some separation in the first two principal component score vectors.

```
ggplot(prcomp(X)$x[,1:2], aes(x = PC1, y = PC2)) +
  modelr::geom_ref_line(h = 0) +
```

```
  modelr::geom_ref_line(v = 0) +
  geom_text(aes(label = seq(1,60,1)), size = 3) +
  xlab("First Principal Component") +
  ylab("Second Principal Component") +
  ggtitle("First Two Principal Components of Simulated Data")
```



(c) Perform K-means clustering of the observations with $K = 3$. How well do the clusters obtained in K-means clustering compare to the true class labels?

```
set.seed(100)
k_cluster_3 <- kmeans(X, 3, nstart = 20)
table("true" = Y, "predicted" = k_cluster_3$cluster)
```

```
##     predicted
## true  1  2  3
##    1 20  0  0
##    2  0 20  0
##    3  0  0 20
```

The clusters are properly labeled.

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

```
set.seed(100)
k_cluster_2 <- kmeans(X, 2, nstart = 20)
table("true" = Y, "predicted" = k_cluster_2$cluster)
```

```
##     predicted
## true  1  2
```

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

The 2s and 3s got properly clustered. However the cluster containing the 2's also has all the 1s.

(e) Now perform K-means clustering with $K = 4$. Describe the results.

```
set.seed(100)
k_cluster_4 <- kmeans(X, 4, nstart = 20)
table("true" = Y, "predicted" = k_cluster_4$cluster)
```

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

The 1's got properly clustered. The 2s got broken into two smaller clusters. The 3s got properly clustered.

(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.

```
set.seed(100)
k_cluster_3two = kmeans(prcomp(X)$x[, 1:2], 3, nstart = 20)
table("true" = Y, "predicted" = k_cluster_3two$cluster)
```

```
##     predicted
## true  1  2  3
##    1 20  0  0
##    2  0 20  0
##    3  0  0 20
```

All the observations are properly clustered.

(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.

```
set.seed(100)
k_cluster_scaled = kmeans(scale(X), 3, nstart = 20)
table("true" = Y, "predicted" = k_cluster_scaled$cluster)
```

```
##     predicted
## true  1  2  3
##    1  0 20  0
##    2  3  2 15
##    3  9  0 11
```

The 1's got into its own cluster. The 2s got broken into 3 clusters whereas the 3s got broken into 2 clusters. This is different from the plot in (b) where there are clearly 3 distinctive clusters.

Question 11: On the book website, www.StatLearning.com, there is a gene expression dataset (`Ch10Ex11.csv`) that consists of 40 tissue samples with measurements on 1,000 genes. The first 20 samples are from healthy patients, while the second 20 are from a diseased group.

(a) Load in the data.

```
df = read_delim("Ch10Ex11.csv", delim = ',', col_names = FALSE)
```

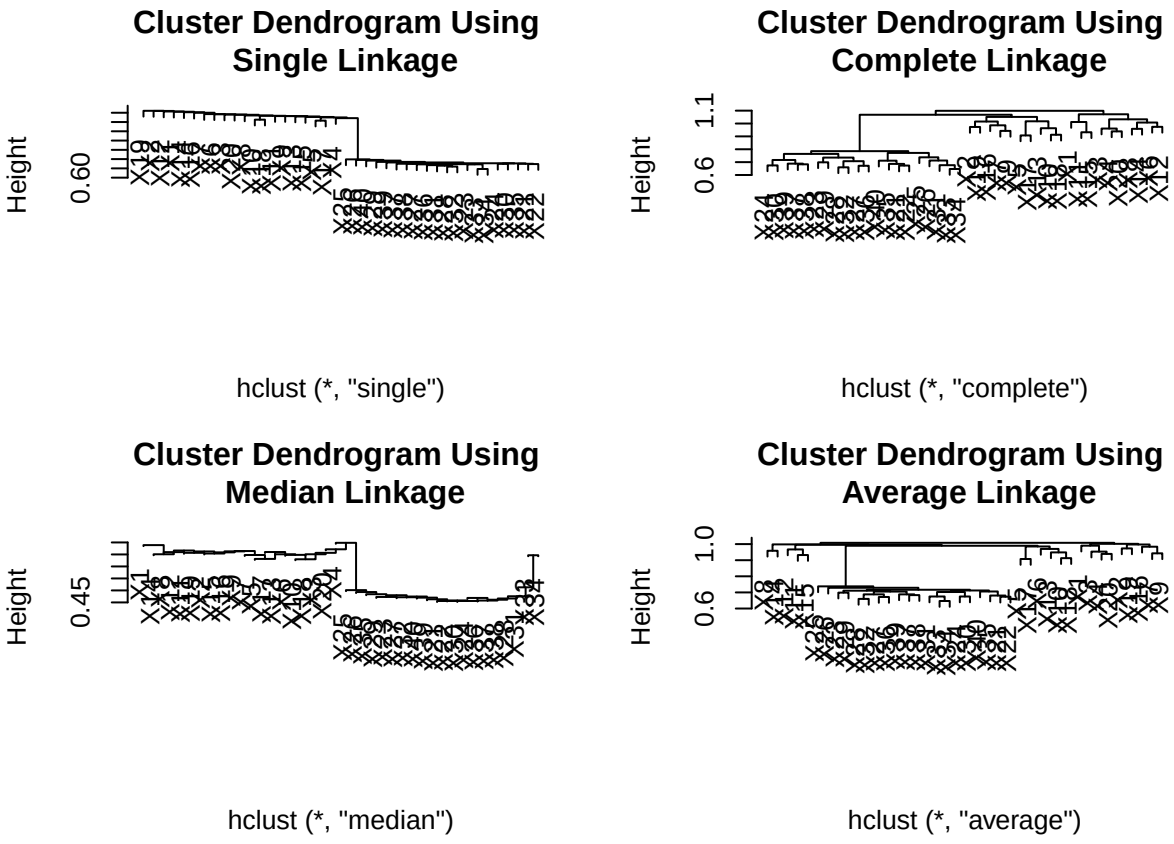
```
## Parsed with column specification:
## cols(
##   .default = col_double()
## )
```

```
## See spec(...) for full column specifications.
```

(b) Apply hierarchical clustering to the samples using correlation-based distance and plot the dendrogram. Do the genes separate the samples into the two groups? Do the results depend on the type of linkage used?

```
cluster_single = hclust(as.dist(1 - cor(df)), method = "single")
cluster_complete = hclust(as.dist(1 - cor(df)), method = "complete")
cluster_median = hclust(as.dist(1 - cor(df)), method = "median")
cluster_average = hclust(as.dist(1 - cor(df)), method = "average")

par(mfrow=c(2,2))
plot(cluster_single, xlab = "",
     main = "Cluster Dendrogram Using \n Single Linkage")
plot(cluster_complete, xlab = "",
     main = "Cluster Dendrogram Using \n Complete Linkage")
plot(cluster_median, xlab = "",
     main = "Cluster Dendrogram Using \n Median Linkage")
plot(cluster_average, xlab = "",
     main = "Cluster Dendrogram Using \n Average Linkage")
```



Depending on the linkage used, the observations were differently clustered. Using the single and complete linkage, two clusters were made but with median and average linkage, three clusters were made. The results does depend on the linkage used.

(c) Your collaborator wants to know which genes differ the most across the two groups. Suggest a way to answer this question and apply it here.

To answer this question, perform principal components analysis on the gene dataset. By summing up the loadings for each gene and ordering by it, the heaviest ones will be the ones that differ the most from the rest.

```
loadings = prcomp(t(df))$rotation
order(abs(rowSums(loadings)), decreasing = TRUE)[1:20]
```

```
##  [1] 865  68 911 428 624  11 524 803 980 822 529 765 801 771 570 654 451
## [18] 237 373 959
```

The above genes are the ones that differ the most across the two groups amongst all 1000 genes.

All of the practice applied exercises in this document are taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.