STA 141A Homework 6

Li Yuan

5/30/2022

Contents

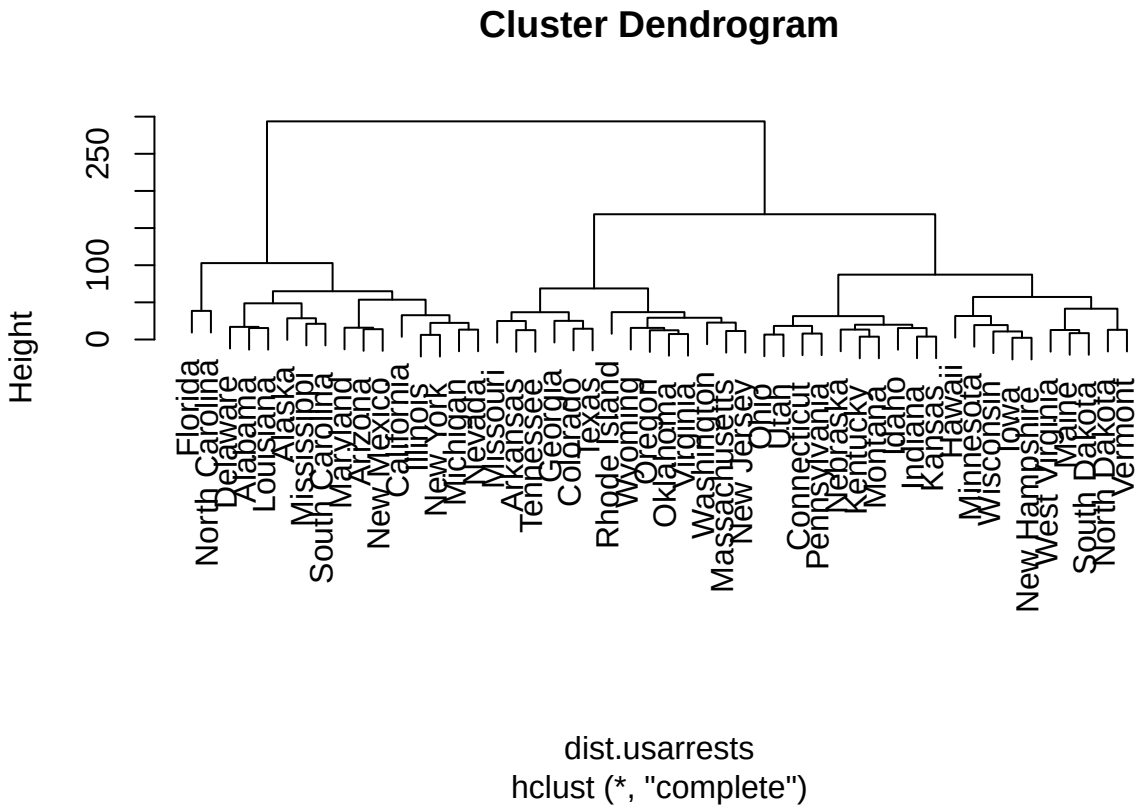
I. Clustering (James et al. 2013, Ch. 10, p. 413)

Consider the `USArrests` data contained in base R (type `data(USArrests)` and ?USArrests to read the help). You will now perform hierarchical clustering on the states.

Question 1

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

```
data(USArrests)
dist.usarrests = dist(USArrests)
hclust.complete = hclust(dist.usarrests, method="complete")
plot(hclust.complete)
```



Question 2

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

```
cutree(hclust.complete, k=3)
```

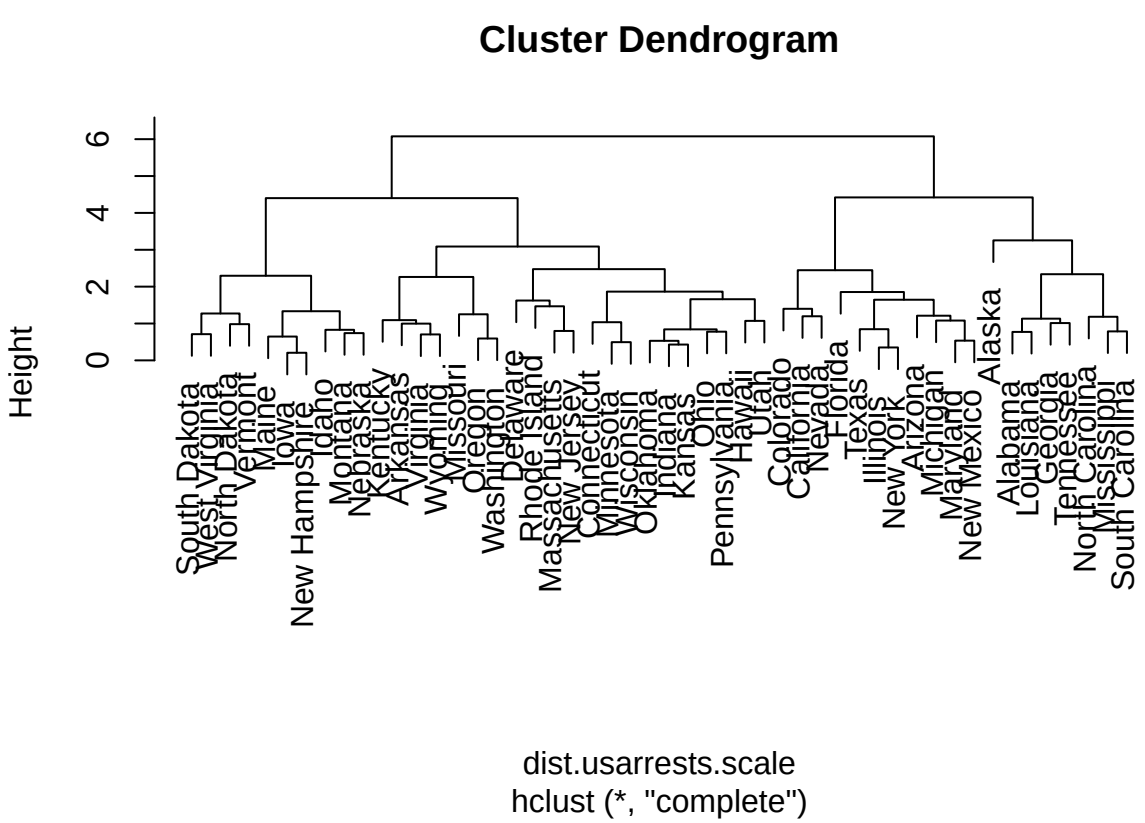
```
##        Alabama         Alaska        Arizona       Arkansas     California
##              1              1              1              2              1
```

```
##       Colorado    Connecticut       Delaware        Florida        Georgia
##              2              3              1              1              2
##         Hawaii          Idaho       Illinois        Indiana           Iowa
##              3              3              1              3              3
##         Kansas       Kentucky      Louisiana          Maine       Maryland
##              3              3              1              3              1
##  Massachusetts       Michigan      Minnesota    Mississippi       Missouri
##              2              1              3              1              2
##        Montana       Nebraska         Nevada  New Hampshire     New Jersey
##              3              3              1              3              2
##     New Mexico       New York North Carolina   North Dakota           Ohio
##              1              1              1              3              3
##       Oklahoma         Oregon   Pennsylvania   Rhode Island South Carolina
##              2              2              3              2              1
##   South Dakota      Tennessee          Texas           Utah        Vermont
##              3              2              2              3              3
##       Virginia     Washington  West Virginia      Wisconsin        Wyoming
##              2              2              3              3              2
```

Question 3

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

```
hclust.scale = scale(USArrests)
dist.usarrests.scale = dist(hclust.scale)
hclust.complete.scale = hclust(dist.usarrests.scale, method="complete")
plot(hclust.complete.scale)
```




```
cutree(hclust.complete.scale, k=3)
```

```
##        Alabama         Alaska        Arizona       Arkansas     California
##              1              1              2              3              2
##       Colorado    Connecticut       Delaware        Florida        Georgia
##              2              3              3              2              1
##         Hawaii          Idaho       Illinois        Indiana           Iowa
##              3              3              2              3              3
##         Kansas       Kentucky      Louisiana          Maine       Maryland
##              3              3              1              3              2
##  Massachusetts       Michigan      Minnesota    Mississippi       Missouri
##              3              2              3              1              3
##        Montana       Nebraska         Nevada New Hampshire     New Jersey
##              3              3              2              3              3
##     New Mexico       New York North Carolina   North Dakota           Ohio
##              2              2              1              3              3
##       Oklahoma         Oregon   Pennsylvania   Rhode Island South Carolina
##              3              3              3              3              1
##   South Dakota      Tennessee          Texas           Utah        Vermont
##              3              1              2              3              3
##       Virginia     Washington  West Virginia      Wisconsin        Wyoming
##              3              3              3              3              3
```

Question 4

(3 points) What effect does scaling the variables have on the hierarchical clustering obtained? In your opinion, should the variables be scaled before the inter-observation dissimilarities are computed? Provide a justification for your answer.

```
# Without scaling
table(cutree(hclust.complete, k=3))
```

```
##
##  1  2  3
## 16 14 20
```

```
# With scaling
table(cutree(hclust.complete.scale, k=3))
```

```
##
##  1  2  3
##  8 11 31
```

```
table(cutree(hclust.complete.scale, k=3),
      cutree(hclust.complete, k=3))
```

```
##
##      1  2  3
##   1  6  2  0
##   2  9  2  0
##   3  1 10 20
```

Scaling the variables impacts the clusters we obtain, the branch length, and the height of the tree. Without scaling, there are so many states that are clustered as group 1, but there are only 8 states that are clustered as group 1. The height of the tree without scaling is about 300, and the height of the tree with scaling is only 6. We also notice that the branch of Alaska is shorter in the scaled tree.

In this case, we should scale the variables before we compute the euclidean distances so that height variable will not affect our fingerwidth width.

II. Hierarchical clustering: centroid method step by step

(6 points) The goal of this problem is to implement a hierarchical clustering algorithm from scratch. In particular, we will use the centroid method to compute the distance between clusters. Following this method, the distance between two clusters is the distance between the two mean vectors of the clusters. At each step of the process we combine the two clusters that have the smallest centroid distance.

Create the algorithm so that it works on the following data set:

```
library(MASS)
set.seed(6)
df <- rbind(mvrnorm(5,c(0,0),diag(2)),mvrnorm(5,c(3,-3),diag(2)))
```

Question 1

As a first step, consider each data point as a single cluster. Calculate the Euclidean distance between all the 10 observations (=10 clusters) in the dataset.

```
dist(df)
```

```
##           1         2         3         4         5         6         7
## 2  1.9032507
## 3  0.7044199 2.5376229
## 4  1.4929818 2.7184237 1.1037985
## 5  1.4375265 0.7042462 1.9765047 2.0237281
## 6  3.6545534 1.7576955 4.2608733 4.2353003 2.2976267
## 7  4.7479773 3.5486545 5.4405482 6.0552877 4.2098600 3.3436663
## 8  5.1728660 3.2717876 5.8049099 5.8078173 3.8599895 1.5732169 3.3476860
## 9  5.1587008 3.3794280 5.8462035 6.0896124 4.0659986 2.1244352 2.1549942
## 10 4.0394832 3.0141723 4.7232977 5.3921801 3.6327602 3.1703941 0.7648718
##           8         9
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9  1.3008429
## 10 3.5314588 2.5073572
```

Question 2

Take the 2 closest data points and make them one cluster. Now the total clusters should be 9.

```
# Find the minimum distance
min(dist(df))
```

```
## [1] 0.7042462
```

The minimum distance is between 2 and 5. So we classify 2 and 5 as 1 cluster. I show the 9 clusters in the next question.

Question 3

Calculate the distance between the cluster created in step (2) and the remaining 8 clusters. In order to calculate the distance between two clusters, consider the centroid method.

```
# 9 clusters
# Centroid on 2 and 5
x2_5 = (df[2,1] + df[5,1]) / 2
y2_5 = (df[2,2] + df[5,2]) / 2
df2 = data.frame(df)
df2[2,1] = x2_5
```

```
df2[5,1] = x2_5
df2[2,2] = y2_5
df2[5,2] = y2_5
df2$cluster = c("1", "2", "3", "4", "2", "5", "6", "7", "8" ,"9")
dist(df2[,1:2])
```

```
##           1         2         3         4         5         6         7
## 2  1.6493733
## 3  0.7044199 2.2470112
## 4  1.4929818 2.3703714 1.1037985
## 5  1.6493733 0.0000000 2.2470112 2.3703714
## 6  3.6545534 2.0150187 4.2608733 4.2353003 2.0150187
## 7  4.7479773 3.8773631 5.4405482 6.0552877 3.8773631 3.3436663
## 8  5.1728660 3.5606272 5.8049099 5.8078173 3.5606272 1.5732169 3.3476860
## 9  5.1587008 3.7218877 5.8462035 6.0896124 3.7218877 2.1244352 2.1549942
## 10 4.0394832 3.3192017 4.7232977 5.3921801 3.3192017 3.1703941 0.7648718
##            8         9
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9  1.3008429
## 10 3.5314588 2.5073572
```

```
sort(dist(df2[,1:2]))[2]
```

```
## [1] 0.7044199
```

```
kable(df2, caption = "9 clusters")
```

Table 1: 9 clusters

X1	X2	cluster
-0.3680252	0.2696060	1
1.1788007	-0.3028989	2
-0.7386219	0.8686598	3
-0.0448730	1.7271955	4
1.1788007	-0.3028989	2
2.9453948	-1.2721489	5
1.2923226	-4.1785997	6
4.0943730	-2.3467933	7
3.2892818	-3.3685665	8
0.7925870	-3.5995546	9

Question 4

Repeat steps (2) and (3) until all the units are in one unique cluster.

```r
# 8 clusters
# Centroid on 1 and 3
x1_3 = (df2[1,1] + df2[3,1]) / 2
y1_3 = (df2[1,2] + df2[3,2]) / 2
df3 = data.frame(df2)
df3[1,1] = x1_3
df3[3,1] = x1_3
df3[1,2] = y1_3
df3[3,2] = y1_3
df3$cluster = c("1", "2", "1", "3", "2", "4", "5", "6", "7" ,"8")
dist(df3[,1:2])
```

```
##            1         2         3         4         5         6         7
## 2  1.9392509
## 3  0.0000000 1.9392509
## 4  1.2647652 2.3703714 1.2647652
## 5  1.9392509 0.0000000 1.9392509 2.3703714
## 6  3.9536501 2.0150187 3.9536501 4.2353003 2.0150187
## 7  5.0938566 3.8773631 5.0938566 6.0552877 3.8773631 3.3436663
## 8  5.4866847 3.5606272 5.4866847 5.8078173 3.5606272 1.5732169 3.3476860
## 9  5.5019172 3.7218877 5.5019172 6.0896124 3.7218877 2.1244352 2.1549942
## 10 4.3805743 3.3192017 4.3805743 5.3921801 3.3192017 3.1703941 0.7648718
##            8         9
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9  1.3008429
## 10 3.5314588 2.5073572
```

```r
sort(dist(df3[,1:2]))[3]
```

```
## [1] 0.7648718
```

```r
kable(df3, caption = "8 clusters")
```

Table 2: 8 clusters

X1	X2	cluster
-0.5533236	0.5691329	1
1.1788007	-0.3028989	2
-0.5533236	0.5691329	1
-0.0448730	1.7271955	3
1.1788007	-0.3028989	2
2.9453948	-1.2721489	4
1.2923226	-4.1785997	5
4.0943730	-2.3467933	6
3.2892818	-3.3685665	7

X1	X2	cluster
0.7925870	-3.5995546	8

```
# 7 clusters
# Centroid on 7 and 10
x7_10 = (df3[7,1] + df3[10,1]) / 2
y7_10 = (df3[7,2] + df3[10,2]) / 2
df4 = data.frame(df3)
df4[7,1] = x7_10
df4[10,1] = x7_10
df4[7,2] = y7_10
df4[10,2] = y7_10
df4$cluster = c("1", "2", "1", "3", "2", "4", "5", "6", "7" ,"5")
dist(df4[,1:2])
```

```
##           1        2        3        4        5        6        7        8
## 2  1.939251
## 3  0.000000 1.939251
## 4  1.264765 2.370371 1.264765
## 5  1.939251 0.000000 1.939251 2.370371
## 6  3.953650 2.015019 3.953650 4.235300 2.015019
## 7  4.735203 3.588769 4.735203 5.720559 3.588769 3.235660
## 8  5.486685 3.560627 5.486685 5.807817 3.560627 1.573217 3.419480
## 9  5.501917 3.721888 5.501917 6.089612 3.721888 2.124435 2.306331 1.300843
## 10 4.735203 3.588769 4.735203 5.720559 3.588769 3.235660 0.000000 3.419480
##           9
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9
## 10 2.306331
```

```
sort(dist(df4[,1:2]))[4]
```

```
## [1] 1.264765
```

```
kable(df4, caption = "7 clusters")
```

Table 3: 7 clusters

X1	X2	cluster
-0.5533236	0.5691329	1
1.1788007	-0.3028989	2
-0.5533236	0.5691329	1
-0.0448730	1.7271955	3
1.1788007	-0.3028989	2

X1	X2	cluster
2.9453948	-1.2721489	4
1.0424548	-3.8890772	5
4.0943730	-2.3467933	6
3.2892818	-3.3685665	7
1.0424548	-3.8890772	5

```
# 6 clusters
# Centroid on 1, 3, and 4
x1_3_4 = (df4[1,1] + df4[3,1] + df4[4, 1]) / 3
y1_3_4 = (df4[1,2] + df4[3,2] + df4[4, 2]) / 3
df5 = data.frame(df4)
df5[1,1] = x1_3_4
df5[3,1] = x1_3_4
df5[4,1] = x1_3_4
df5[1,2] = y1_3_4
df5[3,2] = y1_3_4
df5[4,2] = y1_3_4
df5$cluster = c("1", "2", "1", "1", "2", "3", "4", "5", "6" ,"4")
dist(df5[,1:2])
```

```
##           1        2        3        4        5        6        7        8
## 2  2.006126
## 3  0.000000 2.006126
## 4  0.000000 2.006126 0.000000
## 5  2.006126 0.000000 2.006126 2.006126
## 6  4.005581 2.015019 4.005581 4.005581 2.015019
## 7  5.049841 3.588769 5.049841 5.049841 3.588769 3.235660
## 8  5.563924 3.560627 5.563924 5.563924 3.560627 1.573217 3.419480
## 9  5.673304 3.721888 5.673304 5.673304 3.721888 2.124435 2.306331 1.300843
## 10 5.049841 3.588769 5.049841 5.049841 3.588769 3.235660 0.000000 3.419480
##           9
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9
## 10 2.306331
```

```
sort(dist(df5[,1:2]))[6]
```

```
## [1] 1.300843
```

```
kable(df5, caption = "6 clusters")
```

Table 4: 6 clusters

X1	X2	cluster
-0.383840	0.9551538	1
1.178801	-0.3028989	2
-0.383840	0.9551538	1
-0.383840	0.9551538	1
1.178801	-0.3028989	2
2.945395	-1.2721489	3
1.042455	-3.8890772	4
4.094373	-2.3467933	5
3.289282	-3.3685665	6
1.042455	-3.8890772	4

```
# 5 clusters
# Centroid on 8 and 9
x8_9 = (df4[8,1] + df4[9,1]) / 2
y8_9 = (df4[8,2] + df4[8,2]) / 2
df6 = data.frame(df5)
df6[8,1] = x8_9
df6[9,1] = x8_9
df6[8,2] = y8_9
df6[9,2] = y8_9
df6$cluster = c("1", "2", "1", "1", "2", "3", "4", "5", "5" ,"4")
dist(df6[,1:2])
```

```
##           1        2        3        4        5        6        7        8
## 2  2.006126
## 3  0.000000 2.006126
## 4  0.000000 2.006126 0.000000
## 5  2.006126 0.000000 2.006126 2.006126
## 6  4.005581 2.015019 4.005581 4.005581 2.015019
## 7  5.049841 3.588769 5.049841 5.049841 3.588769 3.235660
## 8  5.245371 3.239260 5.245371 5.245371 3.239260 1.308443 3.065586
## 9  5.245371 3.239260 5.245371 5.245371 3.239260 1.308443 3.065586 0.000000
## 10 5.049841 3.588769 5.049841 5.049841 3.588769 3.235660 0.000000 3.065586
##           9
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9
## 10 3.065586
```

```
sort(dist(df6[,1:2]))[7]
```

```
## [1] 1.308443
```

```
kable(df6, caption = "5 clusters")
```

Table 5: 5 clusters

X1	X2	cluster
-0.383840	0.9551538	1
1.178801	-0.3028989	2
-0.383840	0.9551538	1
-0.383840	0.9551538	1
1.178801	-0.3028989	2
2.945395	-1.2721489	3
1.042455	-3.8890772	4
3.691827	-2.3467933	5
3.691827	-2.3467933	5
1.042455	-3.8890772	4

```
# 4 clusters
# Centroid on 6, 8, and 9
x6_8_9 = (df6[6,1] + df6[8,1] + df6[9, 1]) / 3
y6_8_9 = (df6[6,2] + df6[8,2] + df6[9, 2]) / 3
df7 = data.frame(df6)
df7[6,1] = x6_8_9
df7[8,1] = x6_8_9
df7[9,1] = x6_8_9
df7[6,2] = y6_8_9
df7[8,2] = y6_8_9
df7[9,2] = y6_8_9
df7$cluster = c("1", "2", "1", "1", "2", "3", "4", "3", "3" ,"4")
dist(df7[,1:2])
```

```
##           1        2        3        4        5        6        7        8
## 2  2.006126
## 3  0.000000 2.006126
## 4  0.000000 2.006126 0.000000
## 5  2.006126 0.000000 2.006126 2.006126
## 6  4.828084 2.822798 4.828084 4.828084 2.822798
## 7  5.049841 3.588769 5.049841 5.049841 3.588769 3.061796
## 8  4.828084 2.822798 4.828084 4.828084 2.822798 0.000000 3.061796
## 9  4.828084 2.822798 4.828084 4.828084 2.822798 0.000000 3.061796 0.000000
## 10 5.049841 3.588769 5.049841 5.049841 3.588769 3.061796 0.000000 3.061796
##           9
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9
## 10 3.061796
```

```
sort(dist(df7[,1:2]))[9]
```

```
## [1] 2.006126
```

```
kable(df7, caption = "4 clusters")
```

Table 6: 4 clusters

X1	X2	cluster
-0.383840	0.9551538	1
1.178801	-0.3028989	2
-0.383840	0.9551538	1
-0.383840	0.9551538	1
1.178801	-0.3028989	2
3.443017	-1.9885785	3
1.042455	-3.8890772	4
3.443017	-1.9885785	3
3.443017	-1.9885785	3
1.042455	-3.8890772	4

```
# 3 clusters
# Centroid on 1, 2, 3, 4, and 5
x1_2_3_4_5 = (df7[1,1] + df7[2,1] + df7[3,1] + df7[4,1] + df7[5,1]) / 5
y1_2_3_4_5 = (df7[1,2] + df7[2,2] + df7[3,2] + df7[4,2] + df7[5,2]) / 5
df8 = data.frame(df7)
df8[1,1] = x1_2_3_4_5
df8[2,1] = x1_2_3_4_5
df8[3,1] = x1_2_3_4_5
df8[4,1] = x1_2_3_4_5
df8[5,1] = x1_2_3_4_5
df8[1,2] = y1_2_3_4_5
df8[2,2] = y1_2_3_4_5
df8[3,2] = y1_2_3_4_5
df8[4,2] = y1_2_3_4_5
df8[5,2] = y1_2_3_4_5
df8$cluster = c("1", "1", "1", "1", "1", "2", "3", "2", "2" ,"3")
dist(df8[,1:2])
```

```
##           1        2        3        4        5        6        7        8
## 2  0.000000
## 3  0.000000 0.000000
## 4  0.000000 0.000000 0.000000
## 5  0.000000 0.000000 0.000000 0.000000
## 6  4.025869 4.025869 4.025869 4.025869 4.025869
## 7  4.414335 4.414335 4.414335 4.414335 4.414335 3.061796
## 8  4.025869 4.025869 4.025869 4.025869 4.025869 0.000000 3.061796
## 9  4.025869 4.025869 4.025869 4.025869 4.025869 0.000000 3.061796 0.000000
## 10 4.414335 4.414335 4.414335 4.414335 4.414335 3.061796 0.000000 3.061796
##           9
## 2
## 3
```

```
## 4
## 5
## 6
## 7
## 8
## 9
## 10 3.061796
```

```
sort(dist(df8[,1:2]))[15]
```

```
## [1] 3.061796
```

```
kable(df8, caption = "3 clusters")
```

Table 7: 3 clusters

X1	X2	cluster
0.2412163	0.4519327	1
0.2412163	0.4519327	1
0.2412163	0.4519327	1
0.2412163	0.4519327	1
0.2412163	0.4519327	1
3.4430165	-1.9885785	2
1.0424548	-3.8890772	3
3.4430165	-1.9885785	2
3.4430165	-1.9885785	2
1.0424548	-3.8890772	3

```
# 2 clusters
# Centroid on 6, 7, 8, 9, and 10
x6_7_8_9_10 = (df8[6,1] + df8[7,1] + df8[8,1] + df8[9,1] + df8[10,1]) / 5
y6_7_8_9_10 = (df8[6,2] + df8[7,2] + df8[8,2] + df8[9,2] + df8[10,2]) / 5
df9 = data.frame(df8)
df9[6,1] = x6_7_8_9_10
df9[7,1] = x6_7_8_9_10
df9[8,1] = x6_7_8_9_10
df9[9,1] = x6_7_8_9_10
df9[10,1] = x6_7_8_9_10
df9[6,2] = y6_7_8_9_10
df9[7,2] = y6_7_8_9_10
df9[8,2] = y6_7_8_9_10
df9[9,2] = y6_7_8_9_10
df9[10,2] = y6_7_8_9_10
df9$cluster = c("1", "1", "1", "1", "1", "2", "2", "2", "2" ,"2")
dist(df9[,1:2])
```

```
##          1        2        3        4        5        6        7        8
## 2 0.000000
## 3 0.000000 0.000000
## 4 0.000000 0.000000 0.000000
## 5 0.000000 0.000000 0.000000 0.000000
```

```
## 6  3.907584 3.907584 3.907584 3.907584 3.907584
## 7  3.907584 3.907584 3.907584 3.907584 3.907584 0.000000
## 8  3.907584 3.907584 3.907584 3.907584 3.907584 0.000000 0.000000
## 9  3.907584 3.907584 3.907584 3.907584 3.907584 0.000000 0.000000 0.000000
## 10 3.907584 3.907584 3.907584 3.907584 3.907584 0.000000 0.000000 0.000000
##           9
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9
## 10 0.000000
```

```
sort(dist(df9[,1:2]))[21]
```

```
## [1] 3.907584
```

```
kable(df9, caption = "2 clusters")
```

Table 8: 2 clusters

X1	X2	cluster
0.2412163	0.4519327	1
0.2412163	0.4519327	1
0.2412163	0.4519327	1
0.2412163	0.4519327	1
0.2412163	0.4519327	1
2.4827918	-2.7487780	2
2.4827918	-2.7487780	2
2.4827918	-2.7487780	2
2.4827918	-2.7487780	2
2.4827918	-2.7487780	2

```
# 1 cluster
# Centroid on 1, 2, 3, 4, 5, 6, 7, 8, 9, and 10
x1_to_10 = (df9[1,1] + df9[2,1] + df9[3,1] + df9[4,1] + df9[5,1] +
            df9[6,1] + df9[7,1] + df9[8,1] + df9[9,1] + df9[10,1]) / 10
y1_to_10 = (df9[1,2] + df9[2,2] + df9[3,2] + df9[4,2] + df9[5,2] +
            df9[6,2] + df9[7,2] + df9[8,2] + df9[9,2] + df9[10,2]) / 10
df10 = data.frame(df9)
df10[1,1] = x1_to_10
df10[2,1] = x1_to_10
df10[3,1] = x1_to_10
df10[4,1] = x1_to_10
df10[5,1] = x1_to_10
df10[6,1] = x1_to_10
df10[7,1] = x1_to_10
df10[8,1] = x1_to_10
```

```
df10[9,1] = x1_to_10
df10[10,1] = x1_to_10
df10[1,2] = y1_to_10
df10[2,2] = y1_to_10
df10[3,2] = y1_to_10
df10[4,2] = y1_to_10
df10[5,2] = y1_to_10
df10[6,2] = y1_to_10
df10[7,2] = y1_to_10
df10[8,2] = y1_to_10
df10[9,2] = y1_to_10
df10[10,2] = y1_to_10
df10$cluster = c("1", "1", "1", "1", "1", "1", "1", "1", "1", "1")
dist(df10[,1:2])
```

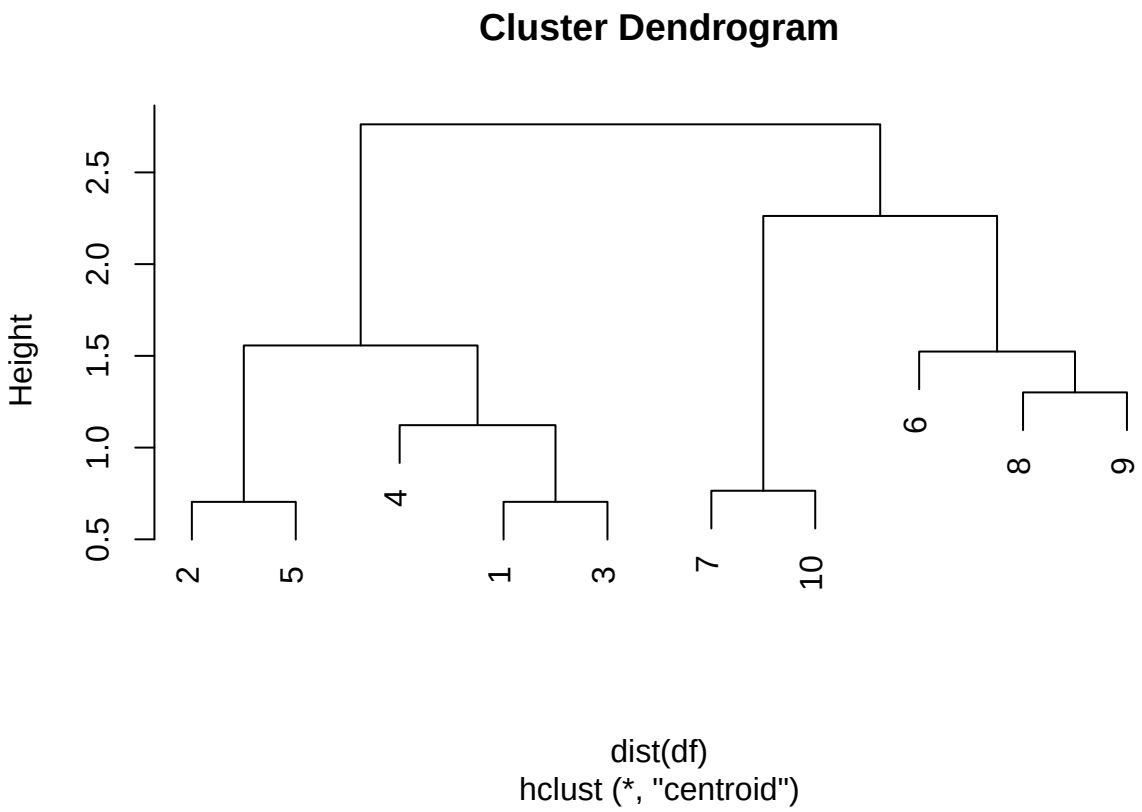
```
##    1 2 3 4 5 6 7 8 9
## 2  0
## 3  0 0
## 4  0 0 0
## 5  0 0 0 0
## 6  0 0 0 0 0
## 7  0 0 0 0 0 0
## 8  0 0 0 0 0 0 0
## 9  0 0 0 0 0 0 0 0
## 10 0 0 0 0 0 0 0 0 0
```

```
kable(df10, caption = "1 cluster")
```

Table 9: 1 cluster

X1	X2	cluster
1.362004	-1.148423	1
1.362004	-1.148423	1
1.362004	-1.148423	1
1.362004	-1.148423	1
1.362004	-1.148423	1
1.362004	-1.148423	1
1.362004	-1.148423	1
1.362004	-1.148423	1
1.362004	-1.148423	1
1.362004	-1.148423	1

```
# Check answer with the plot genrated by R
hc.centroid = hclust(dist(df),
                     method="centroid")
plot(hc.centroid)
```



III. Principal component analysis

The goal of this exercise is to explore the `environmental` data set using principal component analysis (PCA) and to understand how much information we lose by reducing the dimension of the data set.

The data set comes from the package `lattice`, but it can be loaded using the file `environmental.RData` which is available on Canvas. The data set contains daily measurements of ozone concentration, wind speed, temperature and solar radiation in New York City from May to September of 1973 (`111` observations of 4 features).

Question 1

(2 points) Calculate the correlation matrix of all four variables. Determine which two correlation coefficients are the highest in absolute value. Include the names of the variables that lead to the two highest correlation coefficients in absolute value.

```
load("environmental.RData")
# Get the correlation
correlation = cor(environmental)
diag(correlation) = 0
# Get the absolute correlation
abs.correlation = abs(correlation)
abs.correlation
```

```
##                 ozone radiation temperature      wind
## ozone       0.0000000 0.3483417   0.6985414 0.6129508
## radiation   0.3483417 0.0000000   0.2940876 0.1273656
## temperature 0.6985414 0.2940876   0.0000000 0.4971459
## wind        0.6129508 0.1273656   0.4971459 0.0000000
```

```
# Find the largest two coefficient
sort.correlation = unique(sort(abs.correlation, decreasing = TRUE))
sort.names1 = rownames(which(abs.correlation == sort.correlation[1],
                             arr.ind = TRUE))
sort.names2 = rownames(which(abs.correlation == sort.correlation[2],
                             arr.ind = TRUE))
# First highest correlation coefficients in absolute value
c(sort.correlation[1], sort.names1)
```

```
## [1] "0.698541409648639" "temperature"       "ozone"
```

```
# Second highest correlation coefficients in absolute value
c(sort.correlation[2], sort.names2)
```

```
## [1] "0.612950752214463" "wind"              "ozone"
```

Question 2

(2 points) Calculate the loadings of the principal components as well as the standard deviations of the scores. The variables should be shifted to be zero centered and scaled to have unit variance before the calculation takes place.

```
prcomp = prcomp(environmental, scale. = TRUE)
prcomp
```

```
## Standard deviations (1, .., p=4):
## [1] 1.5362936 0.9458106 0.6897738 0.5190919
##
## Rotation (n x k) = (4 x 4):
##                    PC1         PC2        PC3        PC4
## ozone        0.5890271 -0.06304115  0.1137638  0.7975780
## radiation    0.3168987  0.89855477 -0.2773707 -0.1234503
## temperature  0.5527125 -0.06128476  0.6585842 -0.5069713
## wind        -0.4971228  0.42996431  0.6902102  0.3026705
```

Standard deviation for PC1 is 1.5362936.

Standard deviation for PC2 is 0.9458106.

Standard deviation for PC3 is 0.6897738.

standard deviation for PC4 is 0.5190919.

Question 3

(3 points) Which are the two largest loadings in absolute value of the first principal component (include the names of the variables)? How does that correspond to the correlation analysis of the first question?

```
# Get the absolute value of the first principal component
abs.prcomp = abs(prcomp$rotation[,1])
abs.prcomp
```

```
##       ozone   radiation temperature        wind
##   0.5890271   0.3168987   0.5527125   0.4971228
```

```
# Find the first largest
sort(abs.prcomp,decreasing = TRUE)[1]
```

```
##     ozone
## 0.5890271
```

```
# Find the second largest
sort(abs.prcomp,decreasing = TRUE)[2]
```

```
## temperature
##   0.5527125
```

In the first question, the highest correlation is 0.6985414, between `ozone` and `temperature`, which also have the two highest absolute value of the first principal component.

Question 4

(4 points) What proportion of the total variance does the first principal component explain? Is it enough to use the first two principal components if we want to explain at least 90% of the total variance? Explain your answer.

```
summary(prcomp)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4
## Standard deviation     1.536 0.9458 0.6898 0.51909
## Proportion of Variance 0.590 0.2236 0.1190 0.06736
## Cumulative Proportion  0.590 0.8137 0.9326 1.00000
```

```
# First
summary(prcomp)$importance[2,1]
```

```
## [1] 0.59005
```

```
# First + second
summary(prcomp)$importance[2,1]+summary(prcomp)$importance[2,2]
```

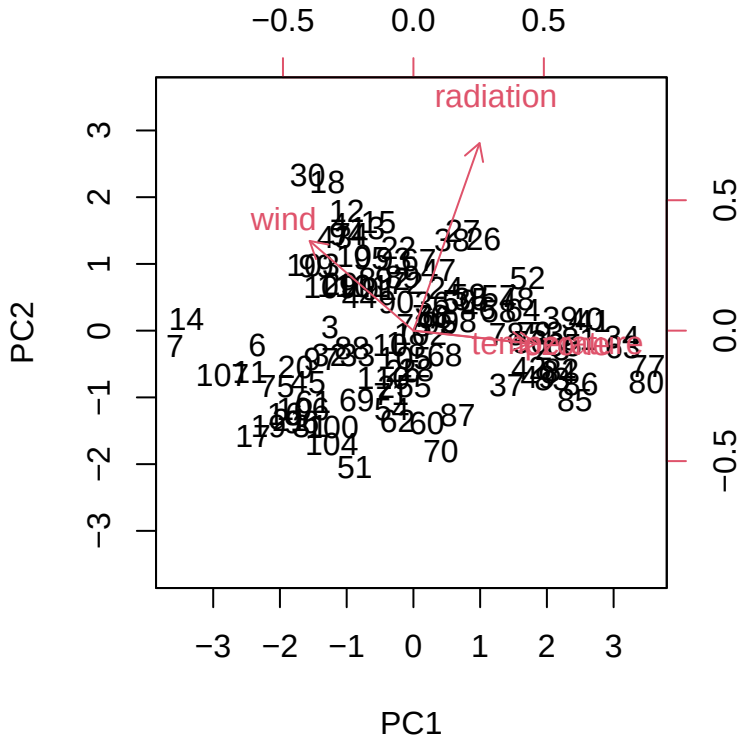
```
## [1] 0.81369
```

The first principle component explains 0.59005 total variance. Since the sum of proportion of variance is 0.81369, which is less than 0.9, so it is not enough to explain at least 90% of total variance.

Question 5

(3 points) Create a biplot and explain what information we see in the biplot.

```
biplot(prcomp, scale = 0)
```



We can interprete this plot in four parts. The x axis represents PC1, which should be `ozone`; the y axis represents PC2, which should be `radiation`.

In the bottom left side, these points have low `ozone` and low `radiation`.

In the bottom right side, these points have high `ozone` and low `radiation`.

In the top left side, these points have low `ozone` and high `radiation`.

In the top right side, these points have high `ozone` and high `radiation`.

Moreover, we can see that `temperature` and `ozone` are kind of overlapped. This is because their loadings in PC1 are very close to each other. However, the correlation between `temperature` and `ozone` is 0.6985414, which actually tells us that these two variables does not have the same effect. PC1 and PC2 can explain 81.369% of the total variance, which is better than using PC3 and PC4.

IV. Extra credit: kNN (6 points)

Starting from the `iris` dataset, the following code returns `P(Y='setosa'|X)` using the k-NN method and $k = 5$. In particular, `Sepal.Length` and `Sepal.Width` are used to compute the distance between units.

```
data(iris)
k <- 5
knn <- apply(as.matrix(dist(iris[,1:2])),1,order)[2:(k+1),]
pr <- function(i, y)
    {
        pr1 <- mean(y[i]=="setosa")
        return(pr1)
    }
apply(knn,2,pr,y=iris$Species)
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
## 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
## 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
## 1.0 0.2 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.2 0.0 0.2
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
## 0.2 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
## 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
## 0.0 0.0 0.0 0.0 0.0 0.0 0.2 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
## 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
## 141 142 143 144 145 146 147 148 149 150
## 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
```

Modify the above code to also return the probability of `virginica` and `versicolor`.

The code should thus return three probabilities per each unit (i.e. `P(Y=setosa'|X)`, `P(Y='virginica'|X)`, and `P(Y='versicolor'|X)`) instead of only one.

The three probabilities must appear in a matrix containing 3 rows and 150 columns. Each row must be named with the name of the species the probability refers to.

```
k <- 5
knn <- apply(as.matrix(dist(iris[,1:2])),1,order)[2:(k+1),]
pr <- function(i, y)
    {
        pr1 <- mean(y[i]=="setosa")
        pr2 <- mean(y[i]=="virginica")
        pr3 <- mean(y[i]=="versicolor")
        return(c(pr1, pr2, pr3))
    }
matrix.q3 = apply(knn,2,pr,y=iris$Species)
rownames(matrix.q3) = c("setosa", "virginica", "versicolor")
colnames(matrix.q3) = seq(1, 150)
matrix.q3
```

```
##            1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
## setosa     1 1 1 1 1 1 1 1 1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
## virginica  0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## versicolor 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

```
##            27 28 29 30 31 32 33 34 35 36 37 38 39 40 41  42 43 44 45 46 47 48
## setosa      1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 0.2  1  1  1  1  1  1
## virginica   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 0.2  0  0  0  0  0  0
## versicolor  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 0.6  0  0  0  0  0  0
##            49 50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65
## setosa      1  1 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.2 0.0 0.2 0.2 0.0 0.0 0.0 0.0
## virginica   0  0 0.8 0.8 0.8 0.2 0.6 0.4 0.8 0.2 0.6 0.0 0.2 0.6 0.4 0.2 0.2
## versicolor  0  0 0.2 0.2 0.2 0.8 0.4 0.6 0.2 0.6 0.4 0.8 0.6 0.4 0.6 0.8 0.8
##             66 67  68  69  70  71  72  73  74  75  76  77  78  79  80 81 82  83
## setosa     0.0  0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0  0  0 0.0
## virginica  0.6  0 0.6 0.4 0.2 0.4 0.2 0.8 0.2 0.8 0.6 0.4 0.6 0.2 0.4  0  0 0.6
## versicolor 0.4  1 0.4 0.6 0.8 0.6 0.8 0.2 0.8 0.2 0.4 0.6 0.4 0.8 0.6  1  1 0.4
##             84  85  86  87  88 89 90 91  92  93  94  95 96  97  98  99 100 101
## setosa     0.0 0.0 0.0 0.0 0.0  0  0  0 0.0 0.0 0.0 0.0  0 0.0 0.0 0.0 0.0 0.0
## virginica  0.2 0.2 0.6 0.6 0.4  0  0  0 0.4 0.4 0.2 0.2  0 0.2 0.6 0.2 0.4 0.8
## versicolor 0.8 0.8 0.4 0.4 0.6  1  1  1 0.6 0.6 0.8 0.8  1 0.8 0.4 0.8 0.6 0.2
##            102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118
## setosa     0.0 0.0 0.0 0.0   0 0.2   0 0.0 0.0 0.0 0.0 0.0   0 0.0 0.0 0.0   0
## virginica  0.6 0.6 0.6 0.4   1 0.0   1 0.4 0.8 0.8 0.8 0.4   0 0.2 0.8 0.4   1
## versicolor 0.4 0.4 0.4 0.6   0 0.8   0 0.6 0.2 0.2 0.2 0.6   1 0.8 0.2 0.6   0
##            119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135
## setosa       0 0.0 0.0   0   0   0 0.0 0.0 0.0 0.0 0.0 0.0   0   0 0.0   0 0.0
## virginica    1 0.4 0.6   0   1   1 0.6 0.6 0.4 0.4 0.6 0.8   1   1 0.6   1 0.2
## versicolor   0 0.6 0.4   1   0   0 0.4 0.4 0.6 0.6 0.4 0.2   0   0 0.4   0 0.8
##            136 137 138 139 140 141 142 143 144 145 146 147 148 149 150
## setosa       0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
## virginica    1 0.6 0.8 0.4 0.8 0.6 0.8 0.6 0.6 0.6 0.6 0.8 0.4 0.4 0.6
## versicolor   0 0.4 0.2 0.6 0.2 0.4 0.2 0.4 0.4 0.4 0.4 0.2 0.6 0.6 0.4
```

Appendix: R Script

```
knitr::opts_chunk$set(echo = TRUE)
rm(list=ls())
library(ggplot2)
library(knitr)
data(USArrests)
dist.usarrests = dist(USArrests)
hclust.complete = hclust(dist.usarrests, method="complete")
plot(hclust.complete)
cutree(hclust.complete, k=3)
hclust.scale = scale(USArrests)
dist.usarrests.scale = dist(hclust.scale)
hclust.complete.scale = hclust(dist.usarrests.scale, method="complete")
plot(hclust.complete.scale)
cutree(hclust.complete.scale, k=3)
# Without scaling
table(cutree(hclust.complete, k=3))
# With scaling
table(cutree(hclust.complete.scale, k=3))
table(cutree(hclust.complete.scale, k=3),
      cutree(hclust.complete, k=3))
library(MASS)
set.seed(6)
df <- rbind(mvrnorm(5,c(0,0),diag(2)),mvrnorm(5,c(3,-3),diag(2)))
dist(df)
# Find the minimum distance
min(dist(df))
# 9 clusters
# Centroid on 2 and 5
x2_5 = (df[2,1] + df[5,1]) / 2
y2_5 = (df[2,2] + df[5,2]) / 2
df2 = data.frame(df)
df2[2,1] = x2_5
df2[5,1] = x2_5
df2[2,2] = y2_5
df2[5,2] = y2_5
df2$cluster = c("1", "2", "3", "4", "2", "5", "6", "7", "8" ,"9")
dist(df2[,1:2])
sort(dist(df2[,1:2]))[2]
kable(df2, caption = "9 clusters")
# 8 clusters
# Centroid on 1 and 3
x1_3 = (df2[1,1] + df2[3,1]) / 2
y1_3 = (df2[1,2] + df2[3,2]) / 2
df3 = data.frame(df2)
df3[1,1] = x1_3
df3[3,1] = x1_3
df3[1,2] = y1_3
df3[3,2] = y1_3
df3$cluster = c("1", "2", "1", "3", "2", "4", "5", "6", "7" ,"8")
dist(df3[,1:2])
sort(dist(df3[,1:2]))[3]
```

```
kable(df3, caption = "8 clusters")

# 7 clusters
# Centroid on 7 and 10
x7_10 = (df3[7,1] + df3[10,1]) / 2
y7_10 = (df3[7,2] + df3[10,2]) / 2
df4 = data.frame(df3)
df4[7,1] = x7_10
df4[10,1] = x7_10
df4[7,2] = y7_10
df4[10,2] = y7_10
df4$cluster = c("1", "2", "1", "3", "2", "4", "5", "6", "7" ,"5")
dist(df4[,1:2])
sort(dist(df4[,1:2]))[4]
kable(df4, caption = "7 clusters")

# 6 clusters
# Centroid on 1, 3, and 4
x1_3_4 = (df4[1,1] + df4[3,1] + df4[4, 1]) / 3
y1_3_4 = (df4[1,2] + df4[3,2] + df4[4, 2]) / 3
df5 = data.frame(df4)
df5[1,1] = x1_3_4
df5[3,1] = x1_3_4
df5[4,1] = x1_3_4
df5[1,2] = y1_3_4
df5[3,2] = y1_3_4
df5[4,2] = y1_3_4
df5$cluster = c("1", "2", "1", "1", "2", "3", "4", "5", "6" ,"4")
dist(df5[,1:2])
sort(dist(df5[,1:2]))[6]
kable(df5, caption = "6 clusters")

# 5 clusters
# Centroid on 8 and 9
x8_9 = (df4[8,1] + df4[9,1]) / 2
y8_9 = (df4[8,2] + df4[8,2]) / 2
df6 = data.frame(df5)
df6[8,1] = x8_9
df6[9,1] = x8_9
df6[8,2] = y8_9
df6[9,2] = y8_9
df6$cluster = c("1", "2", "1", "1", "2", "3", "4", "5", "5" ,"4")
dist(df6[,1:2])
sort(dist(df6[,1:2]))[7]
kable(df6, caption = "5 clusters")

# 4 clusters
# Centroid on 6, 8, and 9
x6_8_9 = (df6[6,1] + df6[8,1] + df6[9, 1]) / 3
y6_8_9 = (df6[6,2] + df6[8,2] + df6[9, 2]) / 3
df7 = data.frame(df6)
df7[6,1] = x6_8_9
df7[8,1] = x6_8_9
```

```
df7[9,1] = x6_8_9
df7[6,2] = y6_8_9
df7[8,2] = y6_8_9
df7[9,2] = y6_8_9
df7$cluster = c("1", "2", "1", "1", "2", "3", "4", "3", "3" ,"4")
dist(df7[,1:2])
sort(dist(df7[,1:2]))[9]
kable(df7, caption = "4 clusters")

# 3 clusters
# Centroid on 1, 2, 3, 4, and 5
x1_2_3_4_5 = (df7[1,1] + df7[2,1] + df7[3,1] + df7[4,1] + df7[5,1]) / 5
y1_2_3_4_5 = (df7[1,2] + df7[2,2] + df7[3,2] + df7[4,2] + df7[5,2]) / 5
df8 = data.frame(df7)
df8[1,1] = x1_2_3_4_5
df8[2,1] = x1_2_3_4_5
df8[3,1] = x1_2_3_4_5
df8[4,1] = x1_2_3_4_5
df8[5,1] = x1_2_3_4_5
df8[1,2] = y1_2_3_4_5
df8[2,2] = y1_2_3_4_5
df8[3,2] = y1_2_3_4_5
df8[4,2] = y1_2_3_4_5
df8[5,2] = y1_2_3_4_5
df8$cluster = c("1", "1", "1", "1", "1", "2", "3", "2", "2" ,"3")
dist(df8[,1:2])
sort(dist(df8[,1:2]))[15]
kable(df8, caption = "3 clusters")

# 2 clusters
# Centroid on 6, 7, 8, 9, and 10
x6_7_8_9_10 = (df8[6,1] + df8[7,1] + df8[8,1] + df8[9,1] + df8[10,1]) / 5
y6_7_8_9_10 = (df8[6,2] + df8[7,2] + df8[8,2] + df8[9,2] + df8[10,2]) / 5
df9 = data.frame(df8)
df9[6,1] = x6_7_8_9_10
df9[7,1] = x6_7_8_9_10
df9[8,1] = x6_7_8_9_10
df9[9,1] = x6_7_8_9_10
df9[10,1] = x6_7_8_9_10
df9[6,2] = y6_7_8_9_10
df9[7,2] = y6_7_8_9_10
df9[8,2] = y6_7_8_9_10
df9[9,2] = y6_7_8_9_10
df9[10,2] = y6_7_8_9_10
df9$cluster = c("1", "1", "1", "1", "1", "2", "2", "2", "2" ,"2")
dist(df9[,1:2])
sort(dist(df9[,1:2]))[21]
kable(df9, caption = "2 clusters")

# 1 cluster
# Centroid on 1, 2, 3, 4, 5, 6, 7, 8, 9, and 10
x1_to_10 = (df9[1,1] + df9[2,1] + df9[3,1] + df9[4,1] + df9[5,1] +
            df9[6,1] + df9[7,1] + df9[8,1] + df9[9,1] + df9[10,1]) / 10
```

```
y1_to_10 = (df9[1,2] + df9[2,2] + df9[3,2] + df9[4,2] + df9[5,2] +
            df9[6,2] + df9[7,2] + df9[8,2] + df9[9,2] + df9[10,2]) / 10
df10 = data.frame(df9)
df10[1,1] = x1_to_10
df10[2,1] = x1_to_10
df10[3,1] = x1_to_10
df10[4,1] = x1_to_10
df10[5,1] = x1_to_10
df10[6,1] = x1_to_10
df10[7,1] = x1_to_10
df10[8,1] = x1_to_10
df10[9,1] = x1_to_10
df10[10,1] = x1_to_10
df10[1,2] = y1_to_10
df10[2,2] = y1_to_10
df10[3,2] = y1_to_10
df10[4,2] = y1_to_10
df10[5,2] = y1_to_10
df10[6,2] = y1_to_10
df10[7,2] = y1_to_10
df10[8,2] = y1_to_10
df10[9,2] = y1_to_10
df10[10,2] = y1_to_10
df10$cluster = c("1", "1", "1", "1", "1", "1", "1", "1", "1", "1")
dist(df10[,1:2])
kable(df10, caption = "1 cluster")

# Check answer with the plot genrated by R
hc.centroid = hclust(dist(df),
                     method="centroid")
plot(hc.centroid)
load("environmental.RData")
# Get the correlation
correlation = cor(environmental)
diag(correlation) = 0
# Get the absolute correlation
abs.correlation = abs(correlation)
abs.correlation
# Find the largest two coefficient
sort.correlation = unique(sort(abs.correlation, decreasing = TRUE))
sort.names1 = rownames(which(abs.correlation == sort.correlation[1],
                             arr.ind = TRUE))
sort.names2 = rownames(which(abs.correlation == sort.correlation[2],
                             arr.ind = TRUE))
# First highest correlation coefficients in absolute value
c(sort.correlation[1], sort.names1)
# Second highest correlation coefficients in absolute value
c(sort.correlation[2], sort.names2)
prcomp = prcomp(environmental, scale. = TRUE)
prcomp
# Get the absolute value of the first principal component
abs.prcomp = abs(prcomp$rotation[,1])
abs.prcomp
```

```r
# Find the first largest
sort(abs.prcomp,decreasing = TRUE)[1]
# Find the second largest
sort(abs.prcomp,decreasing = TRUE)[2]
summary(prcomp)
# First
summary(prcomp)$importance[2,1]
# First + second
summary(prcomp)$importance[2,1]+summary(prcomp)$importance[2,2]
biplot(prcomp, scale = 0)
data(iris)
k <- 5
knn <- apply(as.matrix(dist(iris[,1:2])),1,order)[2:(k+1),]
pr <- function(i, y)
    {
        pr1 <- mean(y[i]=="setosa")
        return(pr1)
    }
apply(knn,2,pr,y=iris$Species)
k <- 5
knn <- apply(as.matrix(dist(iris[,1:2])),1,order)[2:(k+1),]
pr <- function(i, y)
    {
        pr1 <- mean(y[i]=="setosa")
        pr2 <- mean(y[i]=="virginica")
        pr3 <- mean(y[i]=="versicolor")
        return(c(pr1, pr2, pr3))
    }
matrix.q3 = apply(knn,2,pr,y=iris$Species)
rownames(matrix.q3) = c("setosa", "virginica", "versicolor")
colnames(matrix.q3) = seq(1, 150)
matrix.q3
```