Problem Set 4: Clustering

Jinglin Yang

Feb 2020

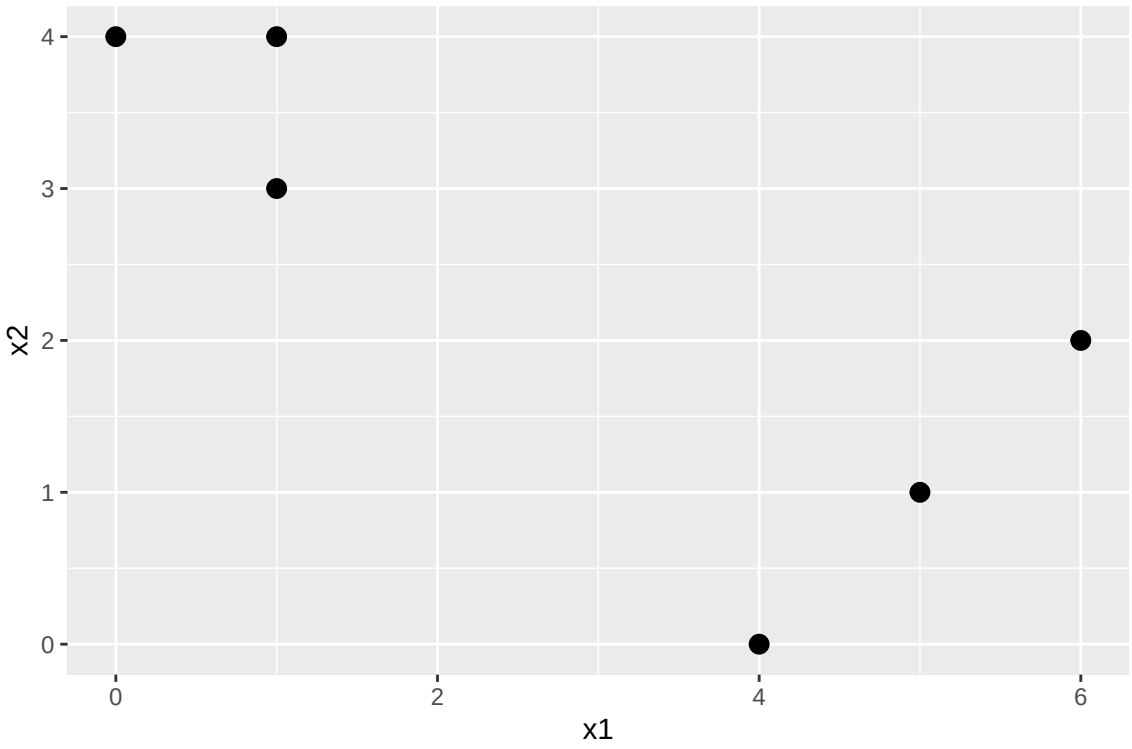
```
library(tidyverse)
library(skimr)
library(dendextend)
library(ggplot2)
library(plyr)
library(fields)
library(dplyr)
library(factoextra)
library(ggpubr)
library(mixtools)
library(plotGMM)
library(clValid)
library(mclust)
```

Performing k-Means By Hand

Question 1

Plot the observations.

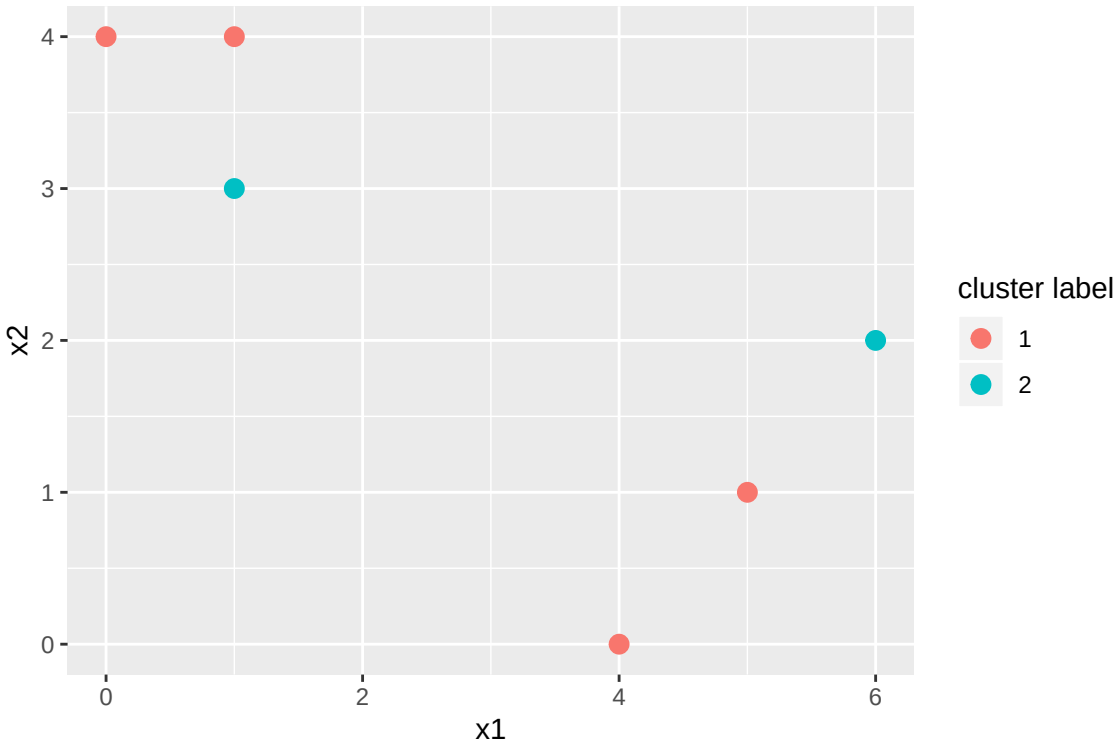
```
> # create the simulated data
> x <- cbind(c(1, 1, 0, 5, 6, 4), c(4, 3, 4, 1, 2, 0))
>
> # plot the observations
> ggplot(data.frame(x),aes(x=x[,1],y=x[,2])) +
+   geom_point(size=3) +
+   labs(x="x1",y="x2")
```



Question 2

Randomly assign a cluster label to each observation. Report the cluster labels for each observation and plot the results with a different color for each cluster (remember to set your seed first).

```
> # randomly assign a cluster label to each observation
> set.seed(1)
> clabel <- sample(c(1,2), size = nrow(x), replace = TRUE); clabel
[1] 1 2 1 1 2 1
>
> # plot
> data_x <- data.frame(cbind(x,clabel))
> data_x <- rename(data_x,c("V1"="x1","V2"="x2"))
> ggplot(data_x,aes(x=x1,y=x2,color=factor(clabel))) +
+   geom_point(size=3) +
+   scale_color_discrete(name="cluster label")
```



Question 3

Compute the centroid for each cluster.

```
> centroid <- ddply(data_x, .(clabel), summarize,  x1=mean(x1), x2=mean(x2)); centroid
  clabel  x1   x2
1      1 2.5 2.25
2      2 3.5 2.50
```

From the results, we know the centroid for the cluster 1 is (2.5, 2.25) and the centroid for the cluster 2 is (3.5, 2.50).

Question 4

Assign each observation to the centroid to which it is closest, in terms of Euclidean distance. Report the cluster labels for each observation.

```
> # calculate the euclidean distance between each observation and each centroid.
> dists <- rdist(data_x[,1:2],centroid[,2:3])
>
>
> # assign each obs. to the centroid to which it is closest.
> # If the distances between the obs. and the two centroids are the same,
> # then randomly assign one of the labels to it.
> set.seed(1)
> newlabel <- function(d){ifelse (length(which(d==min(d)))==1,
+                                 which(d==min(d)),sample(c(1,2),1))}
>
```

```
> data_x$new_label <- apply(dists,1,function(d){newlabel(d)})
> data_x$new_label
[1] 1 1 1 2 2 2
```

Question 5

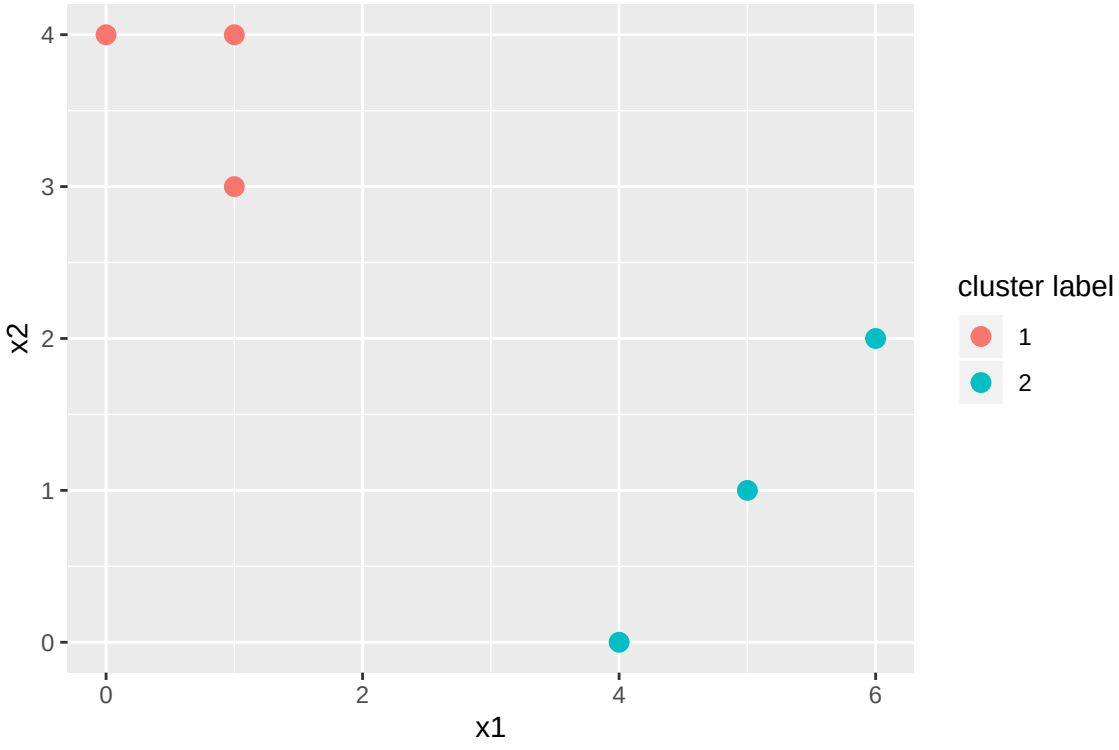
Repeat (3) and (4) until the answers/clusters stop changing.

```
> crit <- 1
> while (crit>0){
+   data_x$clabel <- data_x$new_label
+   centroid <- ddply(data_x, .(clabel), summarize,  x1=mean(x1), x2=mean(x2))
+   dists <- rdist(data_x[,1:2],centroid[,2:3])
+   data_x$new_label <- apply(dists,1,function(d){newlabel(d)}) # update the new cluster label
+
+   # crit will equal to zero only if the clabel for each obs. equals to its new_label.
+   crit <- max(abs(data_x$clabel-data_x$new_label))
+ }
```

Question 6

Reproduce the original plot from (1), but this time color the observations according to the clusters labels you obtained by iterating the cluster centroid calculation and assignments.

```
> ggplot(data_x,aes(x=x1,y=x2,color=factor(clabel))) +
+   geom_point(size=3) +
+   scale_color_discrete(name="cluster label")
```



Clustering State Legislative Professionalism

Question 1

Load the state legislative professionalism data.

```
> legprof <- get(load("/Users/jinglin0828/Downloads/legprof-components.v1.0.RData"))
```

Question 2

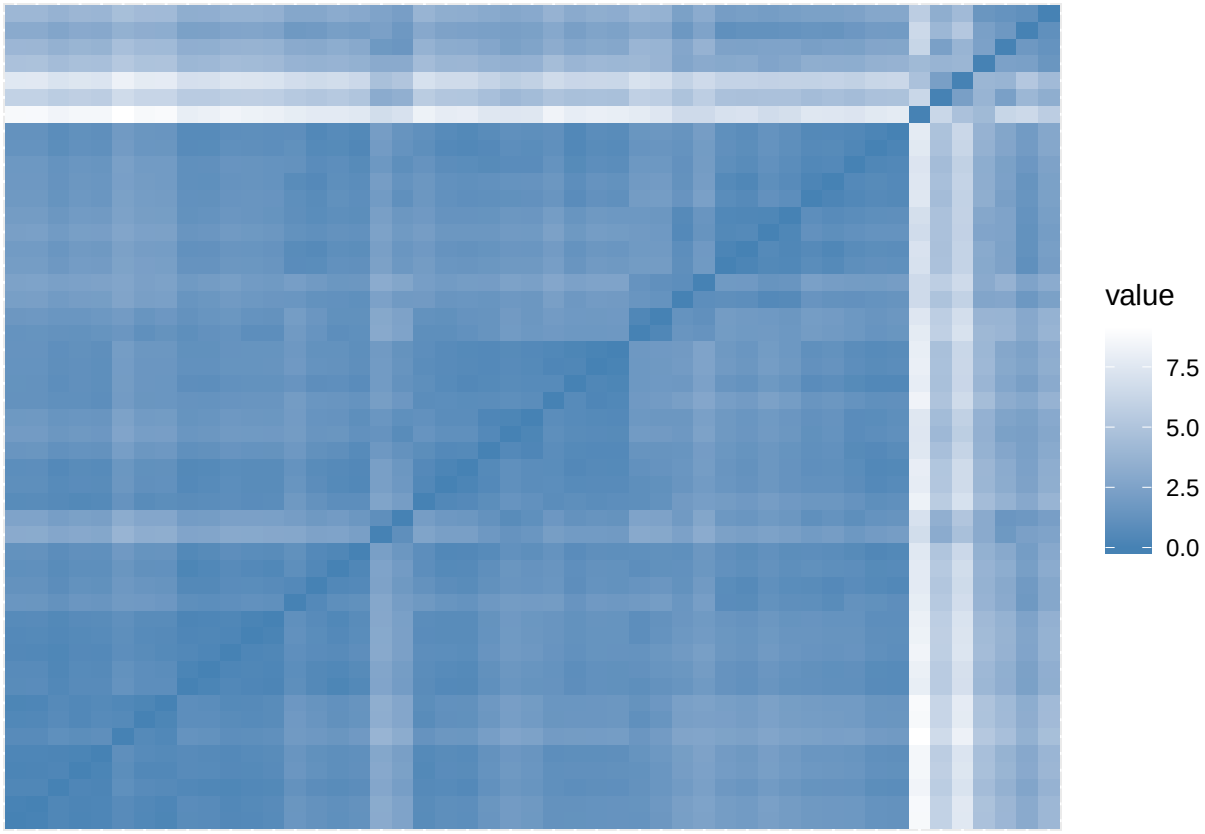
Munge the data.

```
> # (a) select only the continuous features that should capture
> # a state legislature's level of "professionalism"
> legprof <- legprof %>% select(c("stateabv","sessid","t_slength","slength","salary_real","expen
>
> # (b) restrict the data to only include the 2009/10 legislative session for consistency
> legprof <- filter(legprof, sessid == "2009/10")
>
> # (c) omit all missing values
> legprof <- na.omit(legprof)
>
> # (d) standardize the input features and store then as a separate object
> inputs <- scale(legprof[,3:6])
>
> # (e) store the state names as a separate object
> states <- legprof[,c("stateabv")]
>
> # change the row names to the state names
> row.names(legprof) <- states
> row.names(inputs) <- states
```

Quetion 3

Diagnose clusterability in any way you'd prefer (e.g., sparse sampling, ODI, etc.); display the results and discuss the likelihood that natural, non-random structure exist in these data.

```
> clust <- get_clust_tendency(inputs, n = nrow(inputs)-1,seed=1000,graph=TRUE)
>
> # diagnose clusterability in ODI
> clust$plot +  scale_fill_gradient(low = "steelblue", high = "white")
Scale for 'fill' is already present. Adding another scale for 'fill',
which will replace the existing scale.
```



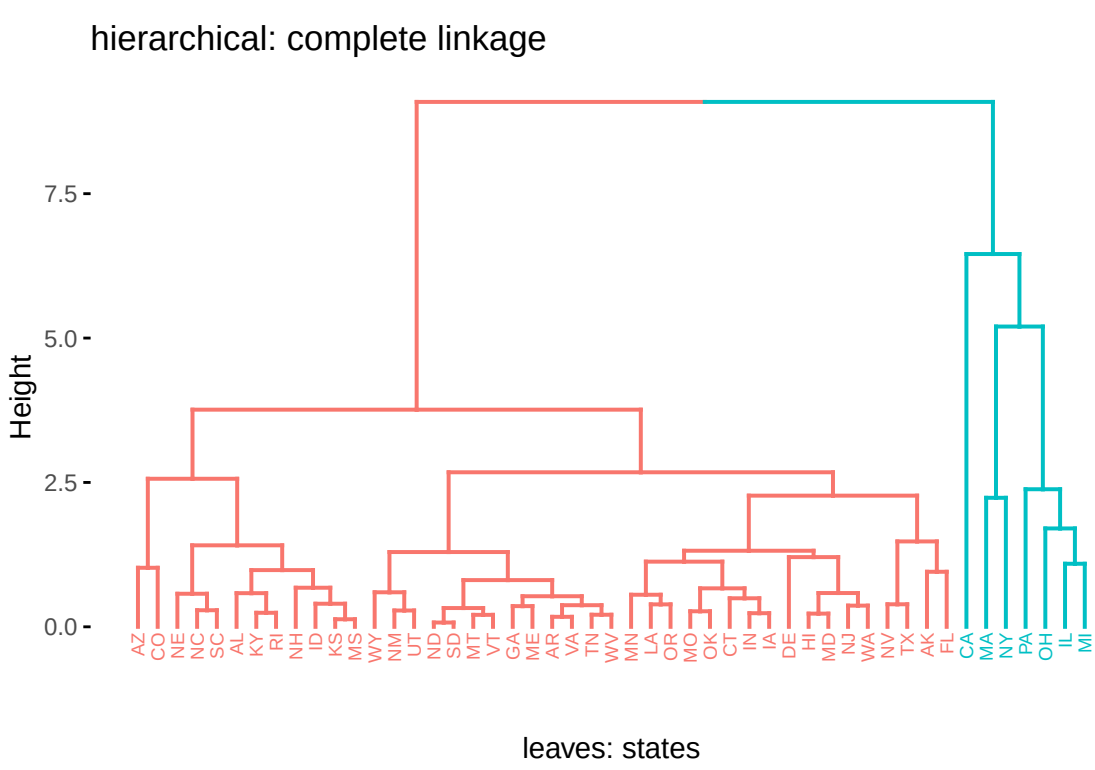

```
> # report the hopkins statistics
> clust$hopkins_stat
[1] 0.8402627
```

First, as we can see, the Hopkins statistics in this case is 0.8402627, which is much larger than 0.5, indicating that the data is highly clustered. Second, we can see that there is a quite large blue area in the ODI grpah, indicating a low dissimilarity (high similarity) between observations in the blue area. Both the Hopskins statistics and the ODI graph suggest clustering tendency is high in the dataset, which also means it is highly likely that there exists non-random structures in these data.

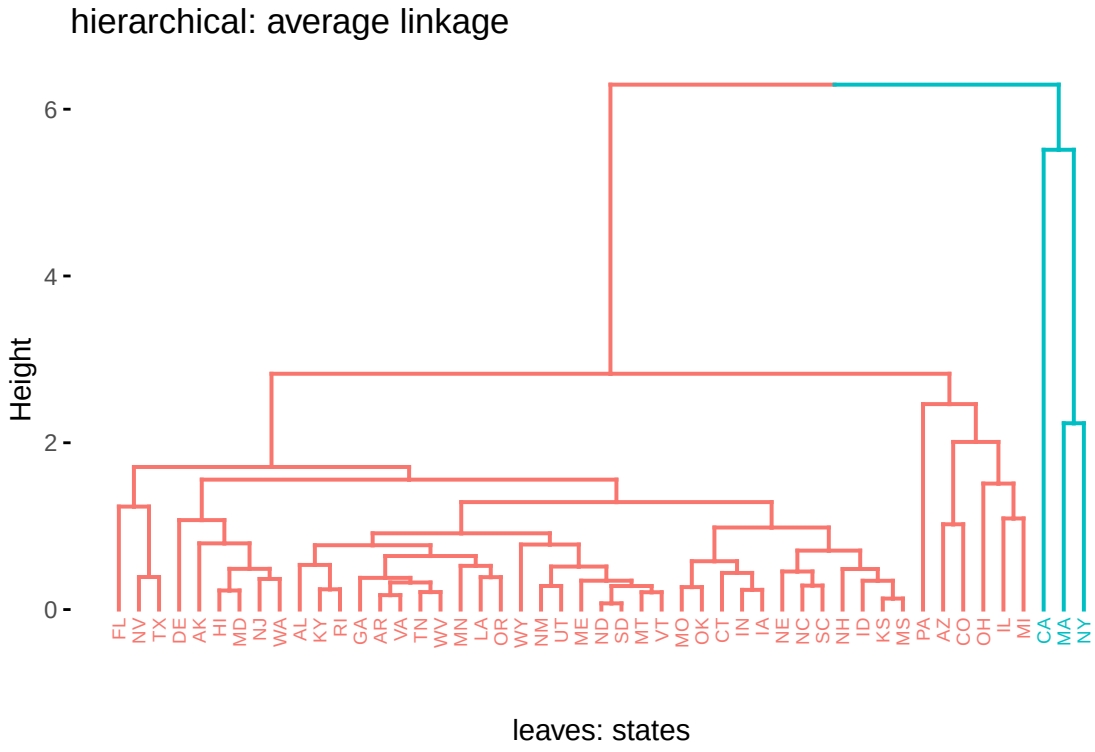
Question 4

Fit an agglomerative hierarchical clustering algorithm using any linkage method you prefer, to these data and present the results. Give a quick, high level summary of the output and general patterns.

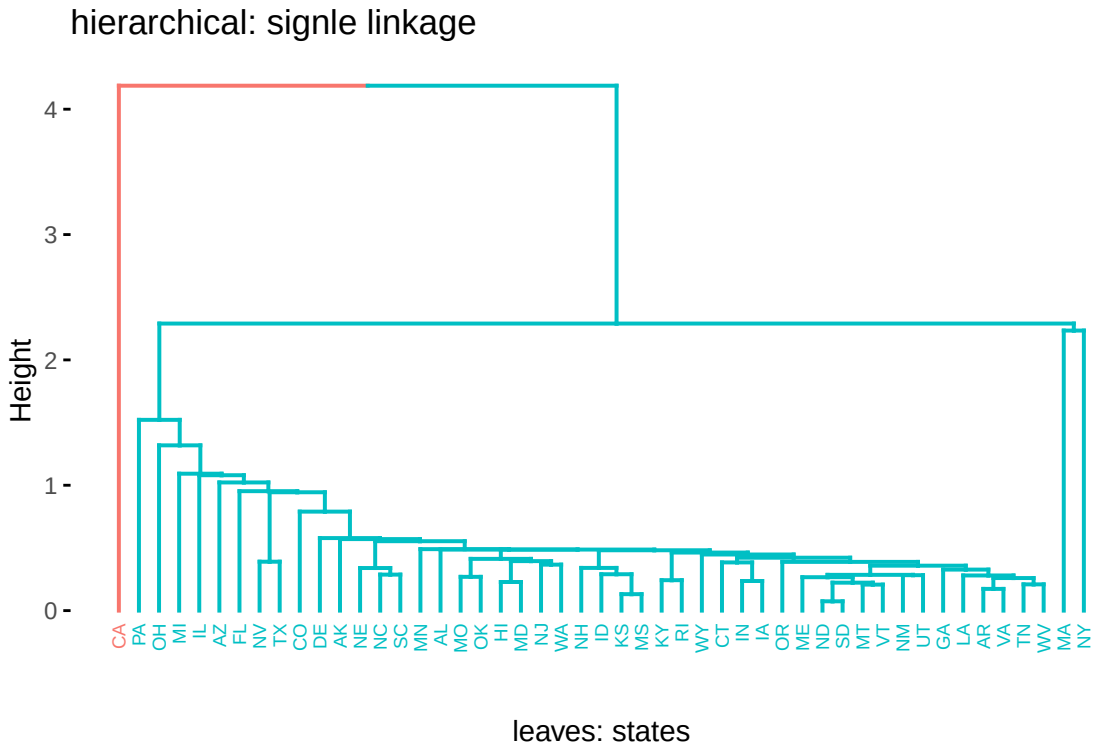
```
> leg_sub <- dist(inputs)
>
> hc_complete <- hclust(leg_sub, method = "complete")
> dendro_complete <- fviz_dend(hc_complete, as.ggplot = TRUE, k = 2, cex = 0.5, main = "hierarch
```




```
>
> hc_average <- hclust(leg_sub, method = "average")
> dendro_average <- fviz_dend(hc_average, as.ggplot = TRUE, k = 2, cex = 0.5, main = "hierarchic
```



```
>
> hc_single <- hclust(leg_sub, method = "single")
> dendro_single <- fviz_dend(hc_single, as.ggplot = TRUE, k = 2, cex = 0.5, main = "hierarchical
```



First, I use three linkage methods: complete, average, and single to fit the agglomerative hierarchical algorithm and draw an dendrogram for each one linkage method. From the three dendrograms, we can see that the tree height is the largest for the complete linkage, medium for the average linkage, and the lowest for the single linkage. This is also easy to know from their definitions. And I find that average and complete linkage tend to yield more balance clusters than single linkage.

We could also find that in the three dendrograms, a large number of states (leaves) fuse into a branch, impling high similarity among the states in this brancn (red branch in complete and average linkage methods, green branch in signle linkage method). The other group has much fewer states and generally they fuse at a larger height.

For simplicity, I will only use "**average linkage**" in the following questions.

Question 5

Fit a k-means algorithm to these data and present the results. Give a quick, high level summary of the output and general patterns. Initialize the algorithm at k = 2, and then check this assumption in the validation questions below.

```
> set.seed(1000)
> kmeans <- kmeans(inputs, 2)
> kmeans
K-means clustering with 2 clusters of sizes 6, 43
```

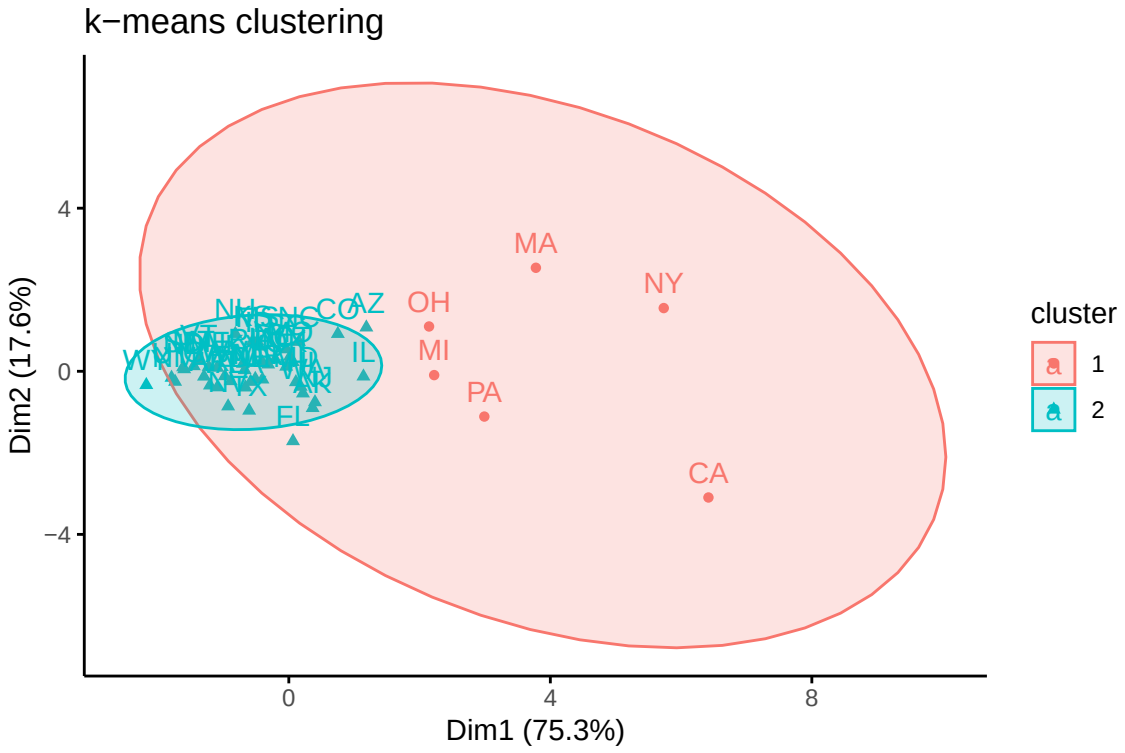
```
Cluster means:
   t_slength    slength salary_real     expend
1  2.1000302  2.1014710   2.0307585  1.4677087
2 -0.2930275 -0.2932285  -0.2833616 -0.2047966

Clustering vector:
AL AK AZ AR CA CO CT DE FL GA HI ID IL IN IA KS KY LA ME MD MA MI MN MS MO
 2  2  2  2  1  2  2  2  2  2  2  2  2  2  2  2  2  2  2  1  1  2  2  2
MT NE NV NH NJ NM NY NC ND OH OK OR PA RI SC SD TN TX UT VT VA WA WV WY
 2  2  2  2  2  2  1  2  2  1  2  2  1  2  2  2  2  2  2  2  2  2  2  2

Within cluster sum of squares by cluster:
[1] 40.36173 48.36594
 (between_SS / total_SS =  53.8 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"
[5] "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"
>
> km_fig <- fviz_cluster(list(data = inputs, cluster = kmeans$cluster),
+              ellipse.type = "norm", stand = FALSE,
+              main = "k-means clustering", show.clust.cent = FALSE,
+              ggtheme = theme_classic())
> km_fig
```



We can see from the graph above, most of states are in a cluster, which is also very centered, implying high similarity (small distance) among the observations. And the other cluster contains much fewer states, and the points in this group are sparsely distributed ("CA","PA","NY","MI","OH","MA" are in this group). The pattern in k-means clustering are very similar to the pattern in the hierarchical clustering.

Question 6

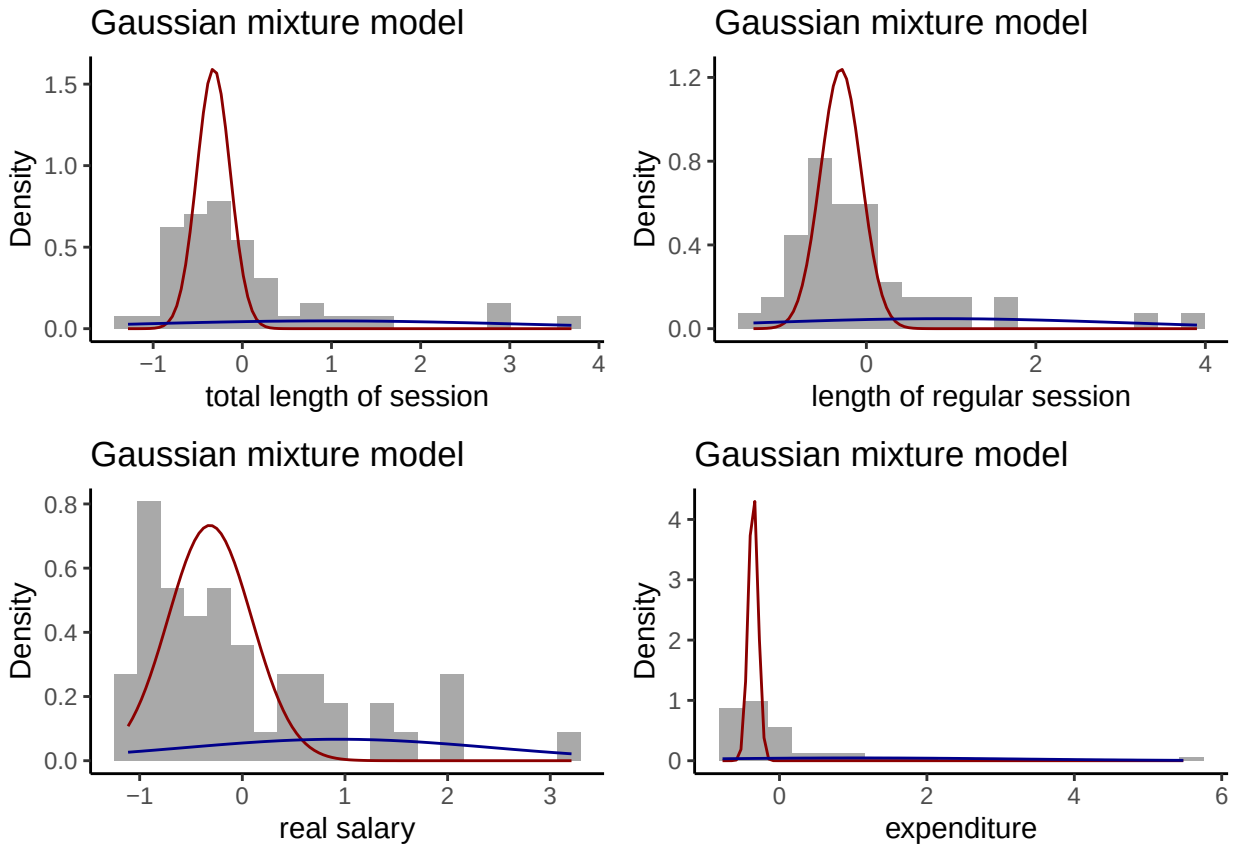
Fit a Gaussian mixture model via the EM algorithm to these data and present the results. Give a quick, high level summary of the output and general patterns. Initialize the algorithm at k = 2, and then check this assumption in the validation questions below.

```
>
> # fit a multivariate Gaussian mixture model via the EM algorithm and present the
> # mean, variance of the two distributions in the two component gmm
> set.seed(1234)
> gmm <- mvnormalmixEM(inputs, k = 2)
number of iterations= 20
> gmm$lambda; gmm$mu; gmm$sigma
[1] 0.7478602 0.2521398
[[1]]
[1] -0.3225760 -0.3008868 -0.3158336 -0.3566484

[[2]]
[1] 0.9567777 0.8924463 0.9367794 1.0578383
[[1]]
          [,1]       [,2]       [,3]       [,4]
[1,] 0.1870505 0.20906090 0.10530159 0.04208450
[2,] 0.2090609 0.24042904 0.11260289 0.03538731
[3,] 0.1053016 0.11260289 0.40683486 0.08685191
[4,] 0.0420845 0.03538731 0.08685191 0.06580480

[[2]]
          [,1]      [,2]     [,3]      [,4]
[1,] 2.1062552 2.0100844 1.271799 0.6272022
[2,] 2.0100844 2.1070027 1.195474 0.2039618
[3,] 1.2717987 1.1954741 1.504998 1.0097881
[4,] 0.6272022 0.2039618 1.009788 2.1936351
>
> # draw the density plots for the four features
> labels <- c("total length of session","length of regular session","real salary","expenditure")
>
> for (i in 1:4){
+ gmm_fig <- ggplot(data.frame(x = gmm$x[,i])) +
+   geom_histogram(aes(x, ..density..), fill = "darkgray", bins = 20) +
+   stat_function(geom = "line", fun = plot_mix_comps,
+                 args = list(gmm$mu[[1]][i], gmm$sigma[[1]][i,i], lam = gmm$lambda[1]),
+                 colour = "darkred") +
```

```
+   stat_function(geom = "line", fun = plot_mix_comps,
+                 args = list(gmm$mu[[2]][i], gmm$sigma[[2]][i,i], lam = gmm$lambda[2]),
+                 colour = "darkblue") +
+   labs(title = "Gaussian mixture model") +
+   xlab(labels[i]) +
+   ylab("Density") +
+   theme_classic()
+   assign(paste("gmm_fig",i,sep=""),gmm_fig)
+ }
>
> gmm_fig <- ggarrange(gmm_fig1,gmm_fig2,gmm_fig3,gmm_fig4, ncol = 2, nrow = 2)
> gmm_fig
```



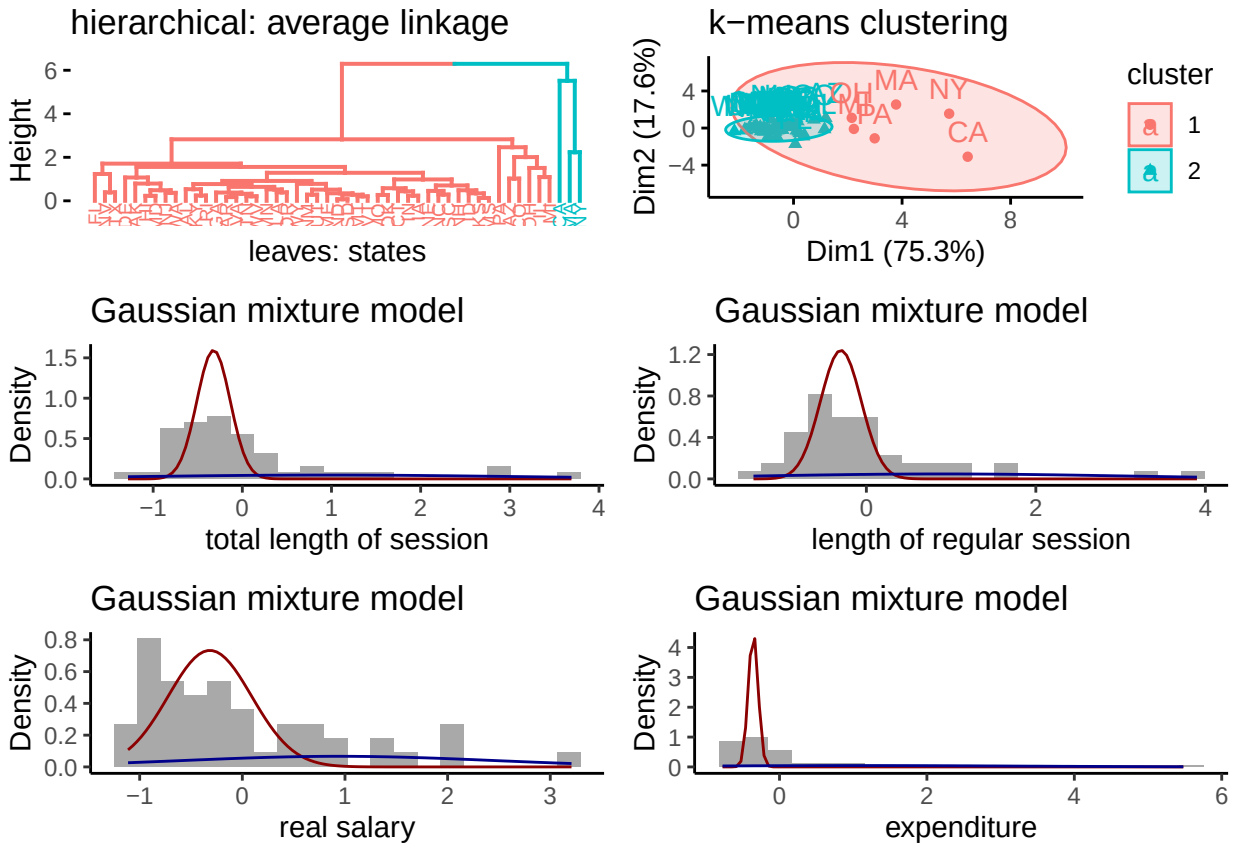
Actually, the GMM algorithm does not assign each observation into different clusters, but returns the probability that each observation belong to each of the two clusters. Therefore, we can interpret the λ in the results above as the overall probability that the observations are in each of the two cluster. In some sense, we could say that 74.8% states are in one cluster, and 25.2% states are in the other cluster. And the mean and the covariance matrix of the two gaussian distributions are reported above.

I also draw the marginal density functions for the four features seperatly. This is because the results are multi-variate gaussian distributions, we could only visualize its marginal densities. From the graphs above, we can also see that the two gaussian distributions are quite different, implying there exist two separate clusters in the dataset.

Question 7

Compare output of all in visually useful, simple ways (e.g., present the dendrogram, plot by state cluster assignment across two features like salary and expenditures, etc.). There should be several plots of comparison and output.

```
> figs <- ggarrange(dendro_average, km_fig, gmm_fig1, gmm_fig2, gmm_fig3, gmm_fig4, ncol = 2, nr
> figs
```



The plots for different clustering methods above have similar patterns. All suggest that most of states are in one cluster, which is centered. And the other cluster has much fewer observations, which are sparsely distributed.

Question 8

Select a single validation strategy (e.g., compactness via min(WSS), average silhouette width, etc.), and calculate for all three algorithms. Display and compare your results for all three algorithms you fit (hierarchical, k-means, GMM).

```
> row.names(inputs) <- c(1:49)
> hc_valid <- clValid(inputs, nClust = 2:10, clMethods = "hierarchical", method = "average", met

Clustering Methods:
 hierarchical

Cluster sizes:
```

```
 2 3 4 5 6 7 8 9 10

Validation Measures:
                                 2       3       4       5       6       7       8       9

hierarchical Connectivity   6.0869  6.9536 16.1885 18.6774 20.6774 21.7607 27.5476 35.5813 37.51
             Dunn           0.3637  0.4371  0.2562  0.2836  0.2836  0.2836  0.2960  0.1568  0.15
             Silhouette     0.6994  0.6711  0.4932  0.4440  0.4284  0.3525  0.2553  0.2652  0.26

Optimal Scores:

             Score  Method       Clusters
Connectivity 6.0869 hierarchical 2
Dunn         0.4371 hierarchical 3
Silhouette   0.6994 hierarchical 2
> km_valid <- clValid(inputs, nClust = 2:10, clMethods = "kmeans", metric = "euclidean", validat

Clustering Methods:
 kmeans

Cluster sizes:
 2 3 4 5 6 7 8 9 10

Validation Measures:
                           2       3       4       5       6       7       8       9      10

kmeans Connectivity   8.4460 10.8960 16.1885 28.7437 30.7437 37.5266 39.4552 40.8694 45.6623
       Dunn           0.1735  0.2581  0.2562  0.1090  0.1090  0.1108  0.1260  0.1324  0.1386
       Silhouette     0.6458  0.6131  0.4932  0.3042  0.2858  0.2750  0.3131  0.3307  0.3288

Optimal Scores:

             Score  Method Clusters
Connectivity 8.4460 kmeans 2
Dunn         0.2581 kmeans 3
Silhouette   0.6458 kmeans 2
> gmm_valid <- clValid(inputs, nClust = 2:10, clMethods = "model", metric = "euclidean", validat

Clustering Methods:
 model

Cluster sizes:
 2 3 4 5 6 7 8 9 10

Validation Measures:
                           2       3       4       5       6       7       8       9      10
```

```
model Connectivity  10.7393 28.6119 39.0687 67.8401 80.4806 69.9774 72.4377 46.7254 60.0976
      Dunn           0.1522  0.0633  0.0225  0.0258  0.0283  0.0543  0.0710  0.1810  0.0977
      Silhouette     0.6314  0.2588  0.1861  0.0085 -0.0562  0.0917  0.0752  0.2831  0.1905

Optimal Scores:

             Score   Method Clusters
Connectivity 10.7393 model  2
Dunn          0.1810 model  9
Silhouette    0.6314 model  2
```

In the results of the validation for the three algorithms, I choose to focus on the Silhouette width. The Silhouette width is an estimate of the average distance between clusters. Its value is comprised between 1 and -1 with a value of 1 indicating a very good cluster, and a value of -1 indicating a very poor cluster.

Based on the Silhouette width, we could see for all the three algorithms, $k = 2$ is best, which **justify our previous assumptions**. Also because hierarchical clustering has the highest Silhouette width, indicating that hierarchical clustering is the best algorithm in this case.

Question 9

Discuss the validation output

(1) From the three fits, we can see that it is likely that there exist two clusters in the dataset. Most of observations (states) are in one cluster, which is quite centered. And the other cluster has much few observations, which are sparsely distributed.

(2) As we discussed in question 8, based on the Silhouette width, hierarchical clustering is optimal because it has the highest Silhouette width. And $k = 2$ is optimal for all the three approaches.

(3) We should be very careful about the interpretation of the results for the clustering methods. Especially for hierarchical and k-means clustering method, because they both assign each observation into a cluster, there are some reasons why we may select technically "sub-optimal" clustering methods. For example, suppose the majority of observations are quite similar and belong to a group, and there exist a few outliers which are quite different from each other and the other observations (just like the case in this question). When we force every points into a cluster, then the clustering results may highly distorted by these outliers. Therefore, these results should not be taken as the ansolute truth about a dataset.