# Problem Set 4

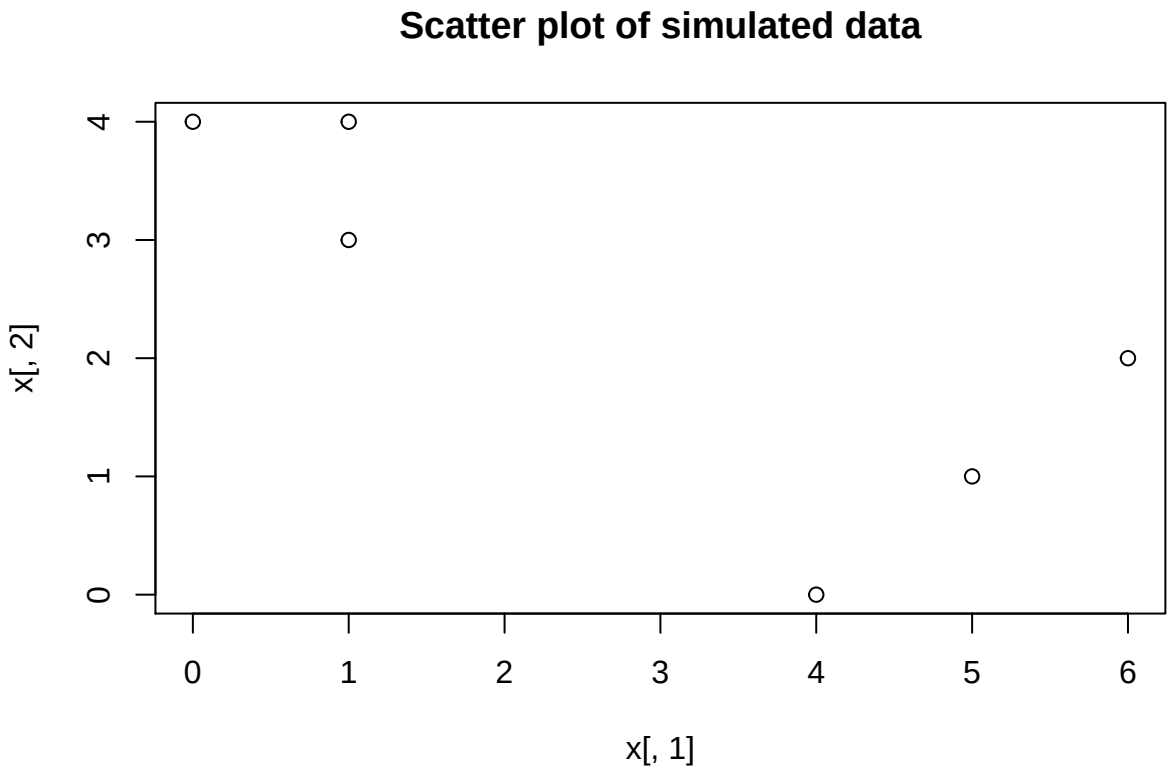
*Minyoung Do*

*2/26/2020*

## Part 1

**(5 points) Plot the observations.**

```
x <- cbind(c(1, 1, 0, 5, 6, 4), c(4, 3, 4, 1, 2, 0))

plot(x[,1], x[,2],
     main="Scatter plot of simulated data")
```



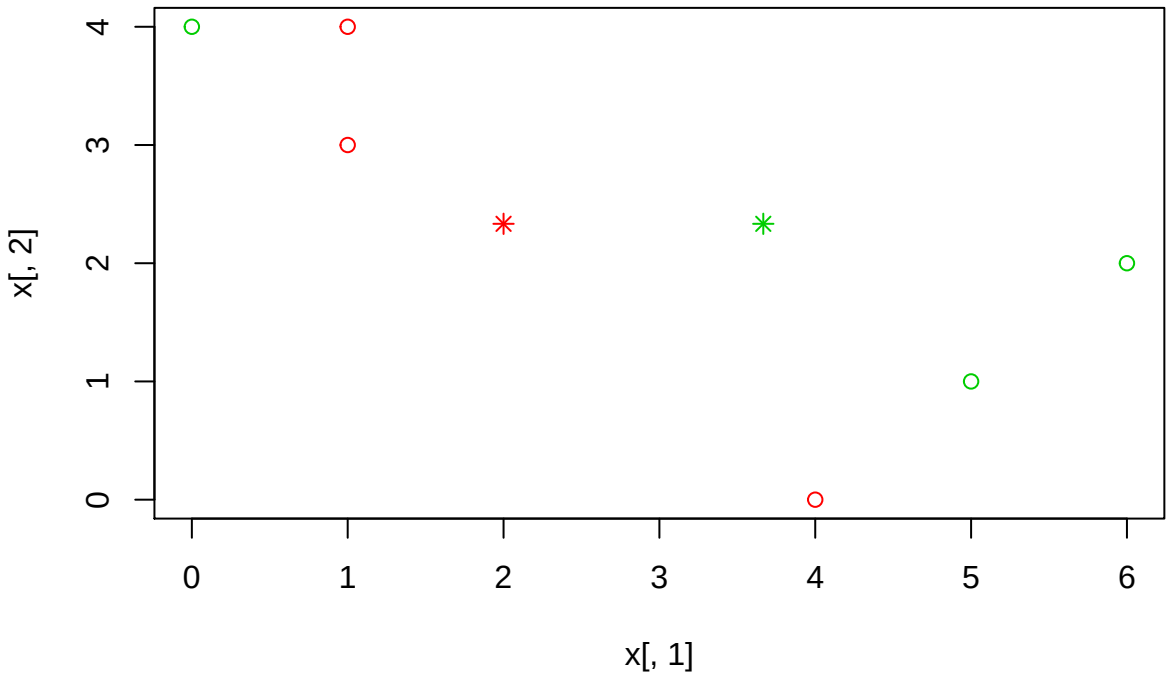

**(5 points) Randomly assign a cluster label to each observation. Report the cluster labels for each observation and plot the results with a different color for each cluster (remember to set your seed first).**

```r
set.seed(420)
labels <- sample(2, nrow(x), replace = T)
labels
```

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

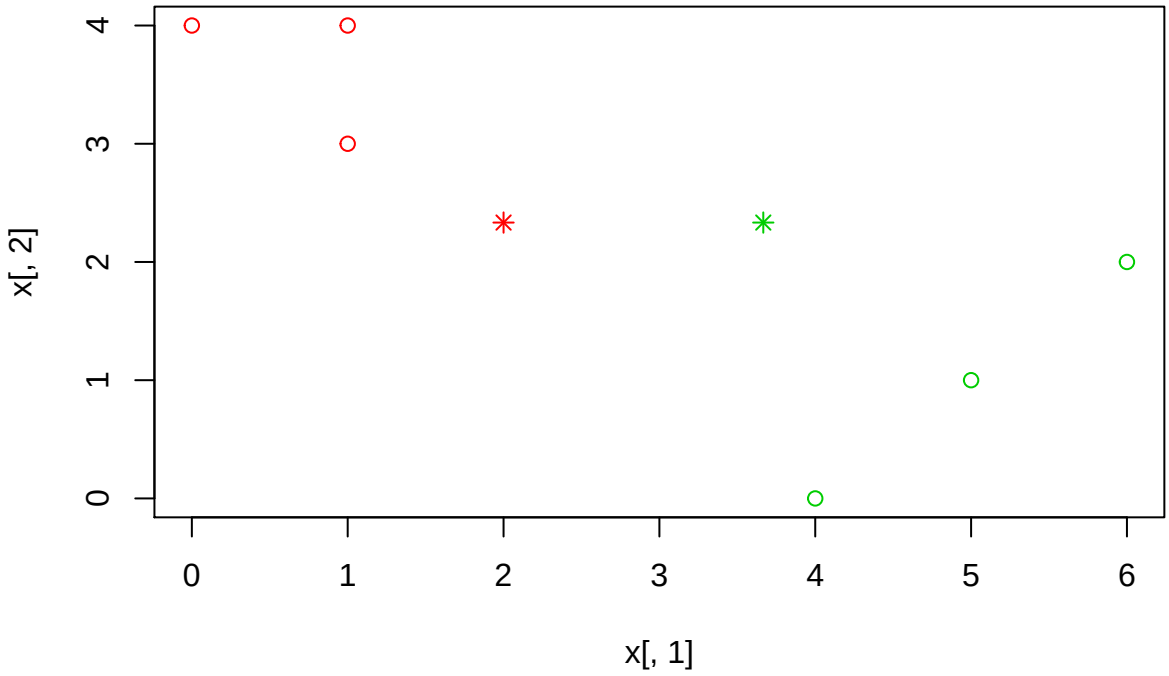
**(10 points) Compute the centroid for each cluster.**

```r
centroid_1 <- c(mean(x[labels == 1, 1]),
                mean(x[labels == 1, 2]))
centroid_2 <- c(mean(x[labels == 2, 1]),
                mean(x[labels == 2, 2]))

# creating a plot with different colors for each feature
plot(x[,1], x[,2],
     col=(labels + 1))

# plotting centroid points
points(centroid_1[1], centroid_1[2],
       col = 2, pch = 8)
points(centroid_2[1], centroid_2[2],
       col = 3, pch = 8)
```



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

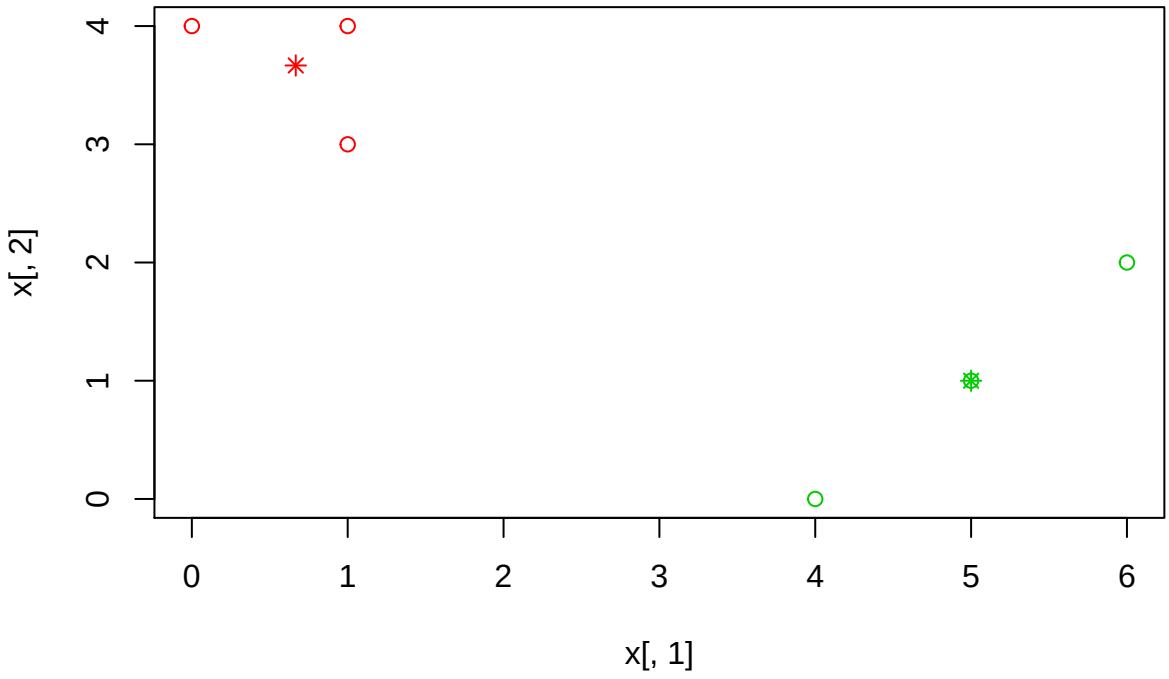
```
# Assigning 1 to 3 observations on the left side, and 2 to the ones on the right side
labels <- c(1, 1, 1, 2, 2, 2)

# plotting relabeled observations
plot(x[, 1], x[, 2],
     col = (labels + 1))
points(centroid_1[1], centroid_1[2],
       col = 2, pch = 8)
points(centroid_2[1], centroid_2[2],
       col = 3, pch = 8)
```



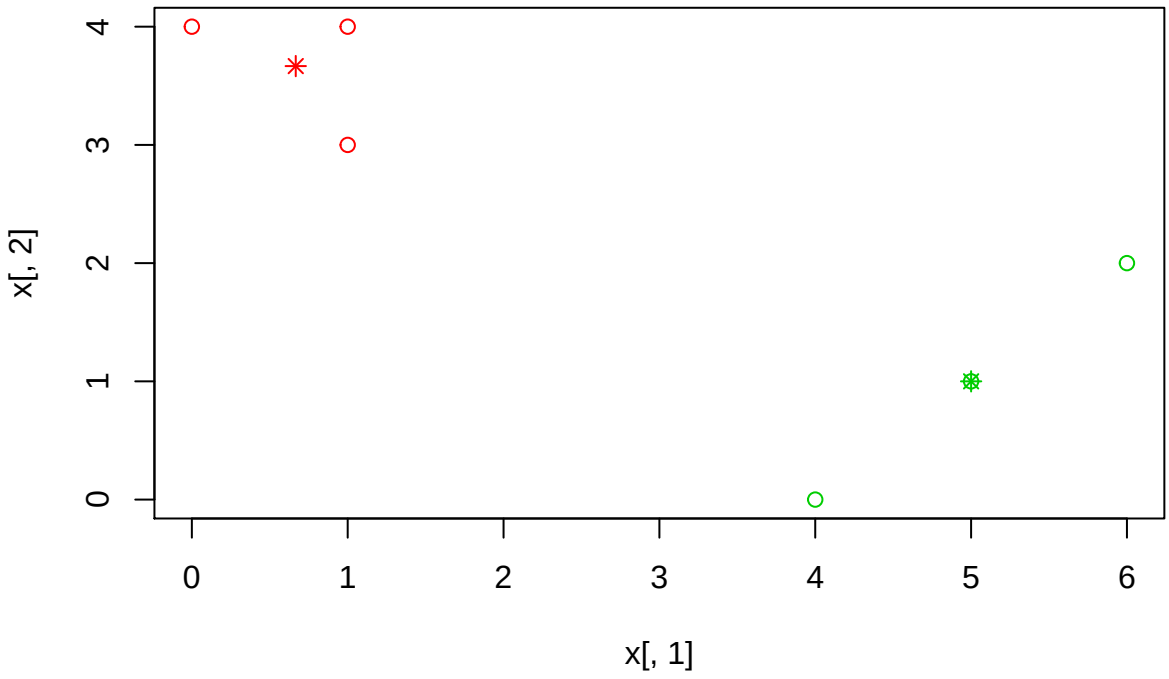

**(5 points) Repeat (3) and (4) until the answers/clusters stop changing.**

```
centroid_1 <- c(mean(x[labels == 1, 1]),
                mean(x[labels == 1, 2]))
centroid_2 <- c(mean(x[labels == 2, 1]),
                mean(x[labels == 2, 2]))

# creating a plot with different colors for each feature
plot(x[,1], x[,2],
```

```
     col=(labels + 1))

# plotting centroid points
points(centroid_1[1], centroid_1[2],
       col = 2, pch = 8)
points(centroid_2[1], centroid_2[2],
       col = 3, pch = 8)
```

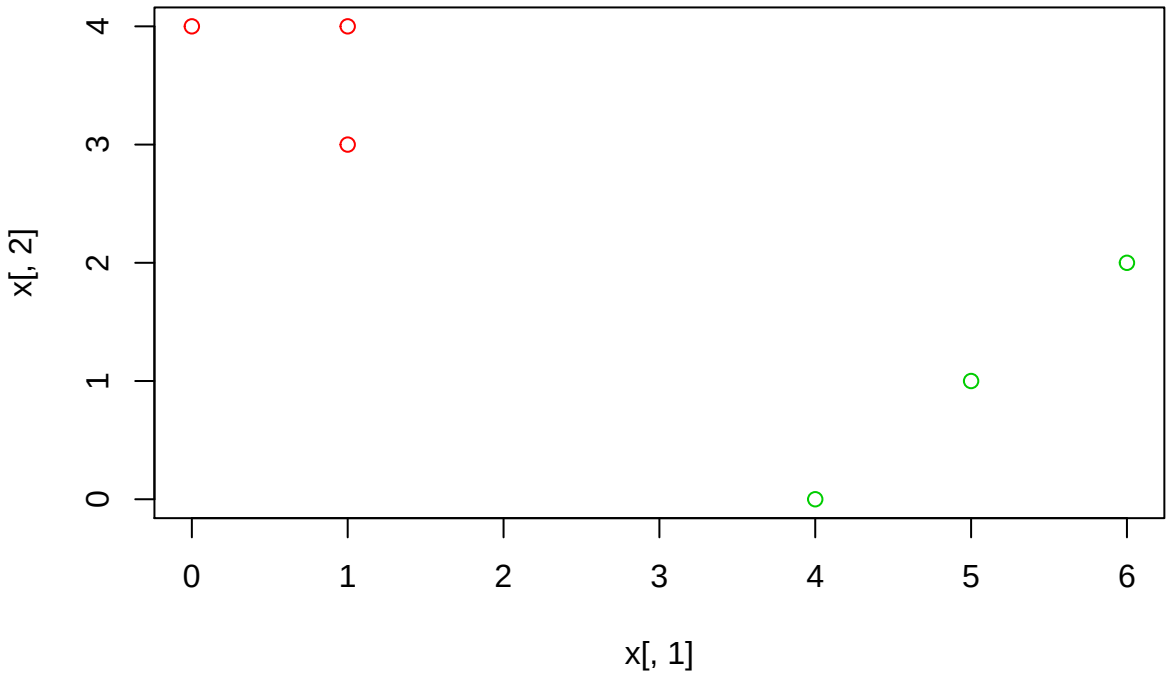



```
# Assigning 1 to 3 observations on the left side, and 2 to the ones on the right side
labels <- c(1, 1, 1, 2, 2, 2)

# plotting relabeled observations
plot(x[, 1], x[, 2],
     col = (labels + 1))
points(centroid_1[1], centroid_1[2],
       col = 2, pch = 8)
points(centroid_2[1], centroid_2[2],
       col = 3, pch = 8)
```




**(10 points) 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.**

```
# plotting relabeled observations
plot(x[, 1], x[, 2],
     col = (labels + 1))
```



# Clustering State Legislative Professionalism

Load the state legislative professionalism data. See the codebook (or above) for further reference.

```
# importing data
load("legprof-components.v1.0.RData")
# giving the data frame a better name
legprof <- x
```

**(5 points) Munge the data:**

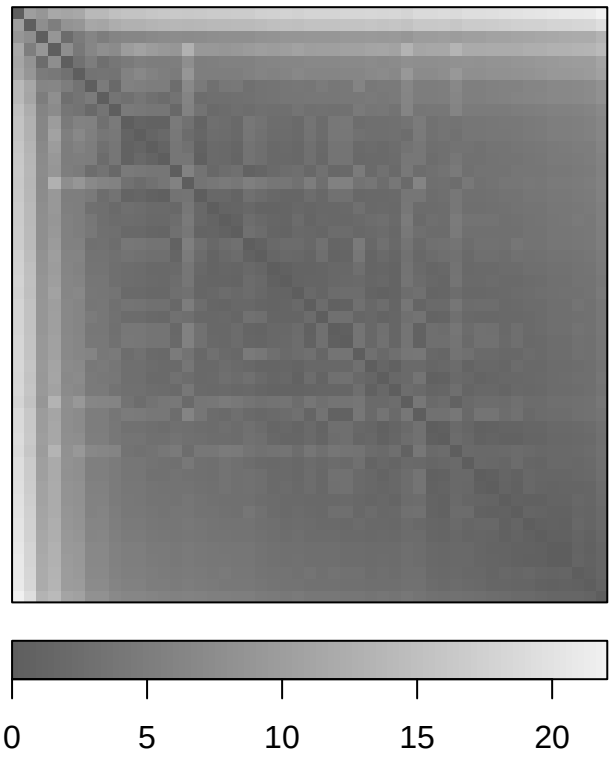
- a. select only the continuous features that should capture a state legislature's level of "professionalism" (session length (total and regular), salary, and expenditures);
- b. restrict the data to only include the 2009/10 legislative session for consistency;
- c. omit all missing values;
- d. standardize the input features;
- e. and anything else you think necessary to get this subset of data into workable form (hint: consider storing the state names as a separate object to be used in plotting later)

```
tidy_legprof <- legprof %>%
  filter(sessid == "2009/10") %>%
  select(stateabv, t_slength, slength, salary_real, expend) %>%
  na.omit()

std <- scale(tidy_legprof[ , -1])

st_legprof <- cbind(tidy_legprof[ , 1], std)
```

**(5 points) 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.**

Hint: We didn't cover how to do this R in class, but consider dissplot() from the seriation package, the factoextra package, and others for calculating, presenting, and exploring the clusterability of some feature space.

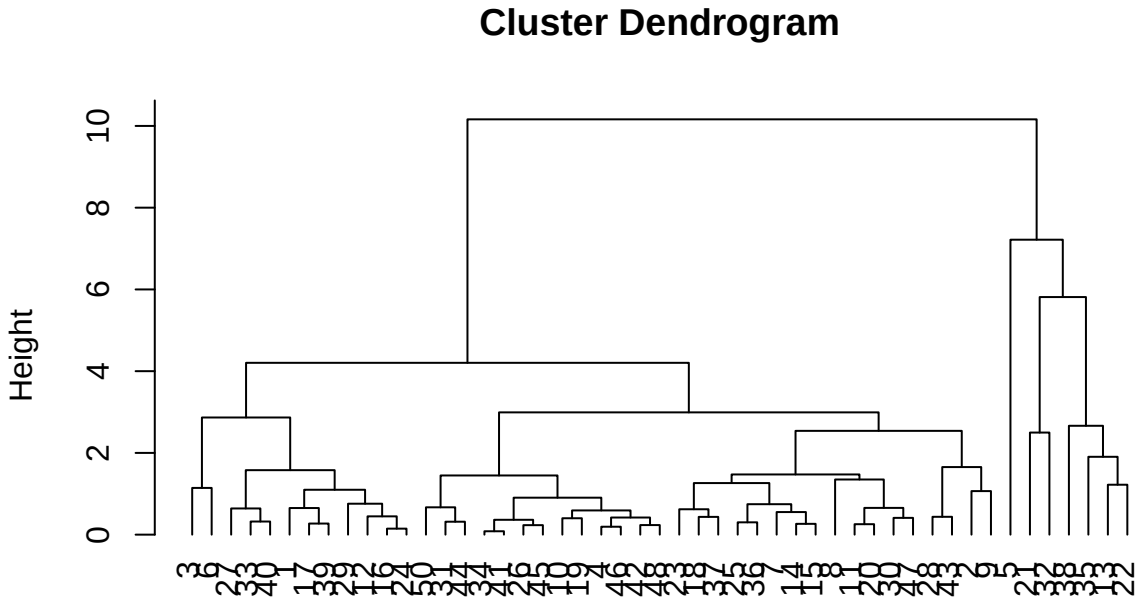
```
legprof_dist <- dist(st_legprof, method = "manhattan")
dissplot(legprof_dist)
```




The seriation algorithm of this function places large similaritites closer to the diagonal line and apply darker shades to low dissimilarity on the diagonal. As seen in the plot, there are a slight differences of shades, especially around the top and left side. However, we do not observe any clear structure that would be visualized as a rectangular/triangular shape in the plot. Therefore, this dissimilarity plot suggests no concerete evidence for strong/conclusive clustering structure of our data, which leads to a low likelihood for the existence of natural, non-random structure in the data.

**(5 points) 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.**

```
dist_legprof <- dist(st_legprof)

hc_complete <- hclust(dist_legprof,
       # maximum distance: complete
       method = "complete"); plot(hc_complete, hang = -1)
```




Hierarchical clustering algorithm treats each observation as a cluster, and then identifies and merge two clusters that are most similar, and the process repeats until all the clusters are merged together. This dendrogram is the result of this process; each number represents a state (except for Wisconsin that was removed due to missing values) on the x-axis. States are considered more similar when they were merged earlier in the process, while it means states share less of similarities if it takes longer for them to merge together. Therefore, this dendrogram shows us similarities and dissimilarities among the states.

**(5 points) 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(77)

kmeans <- kmeans(st_legprof[ , -1],
                 # centers = k (number of clusters)
                 centers = 2,
                 nstart = 50)

State <- tidy_legprof$stateabv
Cluster <- as.tibble(kmeans$cluster)
```

```r
state_cluster_1 <- cbind(State, Cluster) %>%
  as.tibble() %>%
  filter(value == 1)

state_cluster_2 <- cbind(State, Cluster) %>%
  as.tibble() %>%
  filter(value == 2)

kableExtra::kable(state_cluster_1)
```

| State | value |
|---|---|
| CA | 1 |
| MA | 1 |
| MI | 1 |
| NY | 1 |
| OH | 1 |
| PA | 1 |

```r
kableExtra::kable(state_cluster_2)
```

| State | value |
|---|---|
| AL | 2 |
| AK | 2 |
| AZ | 2 |
| AR | 2 |
| CO | 2 |
| CT | 2 |
| DE | 2 |
| FL | 2 |
| GA | 2 |
| HI | 2 |
| ID | 2 |
| IL | 2 |
| IN | 2 |
| IA | 2 |
| KS | 2 |
| KY | 2 |
| LA | 2 |
| ME | 2 |
| MD | 2 |
| MN | 2 |
| MS | 2 |
| MO | 2 |
| MT | 2 |
| NE | 2 |
| NV | 2 |
| NH | 2 |
| NJ | 2 |
| NM | 2 |
| NC | 2 |
| ND | 2 |
| OK | 2 |
| OR | 2 |
| RI | 2 |
| SC | 2 |
| SD | 2 |
| TN | 2 |
| TX | 2 |
| UT | 2 |
| VT | 2 |
| VA | 2 |
| WA | 2 |
| WV | 2 |
| WY | 2 |

This k-means analysis shows that the 88% of observations (43 states) fall under the cluster 2, while only 12% of them (6 states) are clustered as 1. These 6 states (*California*, *Massachusetts*, *Michigan*, *New York*, *Ohio*, *Pennsylvania*), are very sparsely populated in the cluster 1, while cluster 2 has most of the observations. Although it is rather unclear why these states are in the same cluster, this result illustrates what states are similar to each other. It might be a good idea to increase $k$ in order to check for potential clusters we might be missing.

**(5 points) 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.**

```
set.seed(514)
gmm1 <- mvnormalmixEM(std, k = 2)
```

```
## number of iterations= 14
```

```
gmm1$lambda
```

```
## [1] 0.7478598 0.2521402
```

```
gmm1$mu
```

```
## [[1]]
## [1] -0.3225765 -0.3008874 -0.3158343 -0.3566485
##
## [[2]]
## [1] 0.9567776 0.8924466 0.9367799 1.0578367
```

```
gmm1$sigma
```

```
## [[1]]
##             [,1]       [,2]       [,3]       [,4]
## [1,] 0.18704971 0.20906003 0.10530066 0.04208438
## [2,] 0.20906003 0.24042801 0.11260179 0.03538716
## [3,] 0.10530066 0.11260179 0.40683365 0.08685168
## [4,] 0.04208438 0.03538716 0.08685168 0.06580475
##
## [[2]]
```

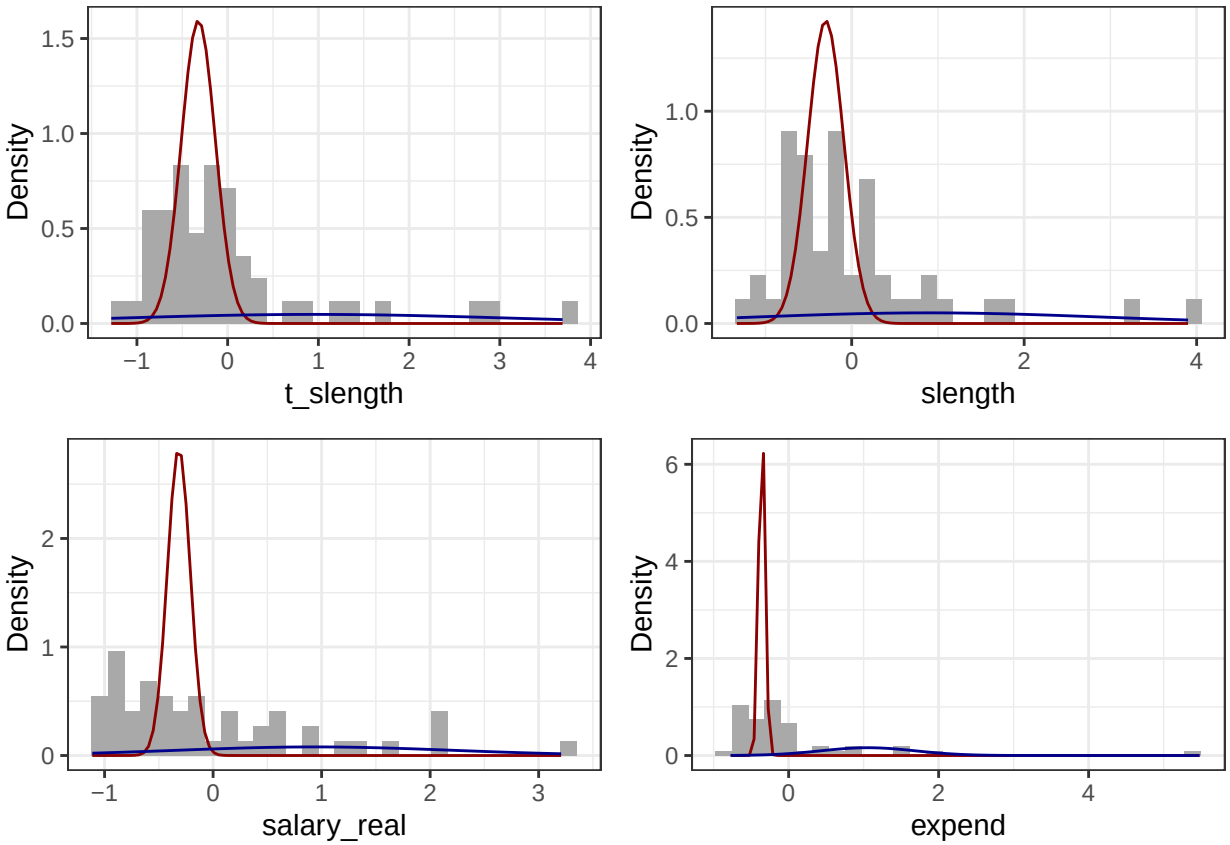
```
##           [,1]      [,2]     [,3]      [,4]
## [1,] 2.1062526 2.0100820 1.271797 0.6272014
## [2,] 2.0100820 2.1070004 1.195473 0.2039611
## [3,] 1.2717971 1.1954728 1.504996 1.0097863
## [4,] 0.6272014 0.2039611 1.009786 2.1936339
```

The first thing that pops up is two lambda ($\lambda$) values, 0.12 and 0.88, that are very similar to the proportion of observations in each cluster as presented above in the previous question. Since the algorithm takes the initial value of mixing proportions as lambda, it seems like an obvious result. Also, we have 4 mu ($\mu$) values and 4 x 4 matrix of sigma ($\sigma$) values for each clusters reported above. This EM algorithm output for mixtures of multivariate normal distributions might be useful to examine the probabilities of each observation belonging to each cluster in the future analysis.

**(15 points) 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.**

```
a <- ggplot(data.frame(x = gmm1$x[,1])) +
  geom_histogram(aes(x, ..density..), fill = "darkgray") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm1$mu[[1]][1], gmm1$sigma[[1]][1], lam = gmm1$lambda[1]),
                colour = "darkred") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm1$mu[[2]][1], gmm1$sigma[[2]][1], lam = gmm1$lambda[2]),
                colour = "darkblue") +
  xlab("t_slength") +
  ylab("Density") +
  theme_bw()

b <- ggplot(data.frame(x = gmm1$x[,2])) +
  geom_histogram(aes(x, ..density..), fill = "darkgray") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm1$mu[[1]][2], gmm1$sigma[[1]][2], lam = gmm1$lambda[1]),
                colour = "darkred") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm1$mu[[2]][2], gmm1$sigma[[2]][2], lam = gmm1$lambda[2]),
                colour = "darkblue") +
  xlab("slength") +
  ylab("Density") +
  theme_bw()

c <- ggplot(data.frame(x = gmm1$x[,3])) +
```

```
  geom_histogram(aes(x, ..density..), fill = "darkgray") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm1$mu[[1]][3], gmm1$sigma[[1]][3], lam = gmm1$lambda[1]),
                colour = "darkred") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm1$mu[[2]][3], gmm1$sigma[[2]][3], lam = gmm1$lambda[2]),
                colour = "darkblue") +
  xlab("salary_real") +
  ylab("Density") +
  theme_bw()

d <- ggplot(data.frame(x = gmm1$x[,4])) +
  geom_histogram(aes(x, ..density..), fill = "darkgray") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm1$mu[[1]][4], gmm1$sigma[[1]][4], lam = gmm1$lambda[1]),
                colour = "darkred") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm1$mu[[2]][4], gmm1$sigma[[2]][4], lam = gmm1$lambda[2]),
                colour = "darkblue") +
  xlab("expend") +
  ylab("Density") +
  theme_bw()

ggarrange(a, b, c, d)
```




These plots show how much a variable explains the clustering patterns by fitting curves. As discussed in class, it seems that the blue curves are stretched out due to outtliers. It seems that curves in the `t_slength` and `slength` plots represent the data better than the other two variables; that is, `t_slength` and `slength` are closely related to the clustering process, whereas the other two variables have a weaker relation to the clustering.

**(5 points) 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). Hint: Here again, we didn't cover this in R in class, but think about using the clValid package, though there are many other packages and ways to validate cluster patterns across iterations.**

```
clvalid_output <- clValid(std, 2:20,
        validation = "internal",
        clMethods = c("model", "kmeans", "hierarchical"))
summary(clvalid_output)
```

```
##
## Clustering Methods:
```

```
##  model kmeans hierarchical
##
## Cluster sizes:
##  2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
##
## Validation Measures:
##                                  2       3       4       5       6       7       8
##
## model        Connectivity  10.7393 28.6119 39.0687 67.8401 80.4806 69.9774 72.4377 46
##              Dunn           0.1522  0.0633  0.0225  0.0258  0.0283  0.0543  0.0710  0
##              Silhouette     0.6314  0.2588  0.1861  0.0085 -0.0562  0.0917  0.0752  0
## kmeans       Connectivity   8.4460 10.8960 16.1885 28.7437 30.7437 37.5266 39.4552 40
##              Dunn           0.1735  0.2581  0.2562  0.1090  0.1090  0.1108  0.1260  0
##              Silhouette     0.6458  0.6131  0.4932  0.3042  0.2858  0.2750  0.3131  0
## hierarchical Connectivity   6.0869  6.9536 16.1885 18.6774 20.6774 21.7607 27.5476 35
##              Dunn           0.3637  0.4371  0.2562  0.2836  0.2836  0.2836  0.2960  0
##              Silhouette     0.6994  0.6711  0.4932  0.4440  0.4284  0.3525  0.2553  0
##
## Optimal Scores:
##
##              Score  Method       Clusters
## Connectivity 6.0869 hierarchical 2
## Dunn         0.4371 hierarchical 3
## Silhouette   0.6994 hierarchical 2
```

**(10 points) Discuss the validation output, e.g.,**

- What can you take away from the fit?
- Which approach is optimal? And optimal at what value of k?
- What are reasons you could imagine selecting a technically "sub-optimal" clustering method, regardless of the validation statistics?

The validation output shows how well each method did in clustering. Of all three different clustering measures, this result suggests that the hierarchical clustering algorithm seems to be the optimal method for our data when the number of clusters are set as 2, while it performs well when $k = 3$ as well. Therefore, further analyses should be performed in order to select the optimal clustering method.

Since each method has its own strength and weakness, sometimes it might make more sense to choose *sub-optimal* clustering method depending on the purpose. For example, considering hierarchical clustering algorithm can't handle big data as well as K Means clustering can, one might prefer K-means over hierarchical clustering when they have a big size of data to analyze. Likewise, as k-means clustering assumes the clusters tot be spherical, one might choose GMM that works better with complex geometrical shaped data.