ProblemSet4

Bohan Yin

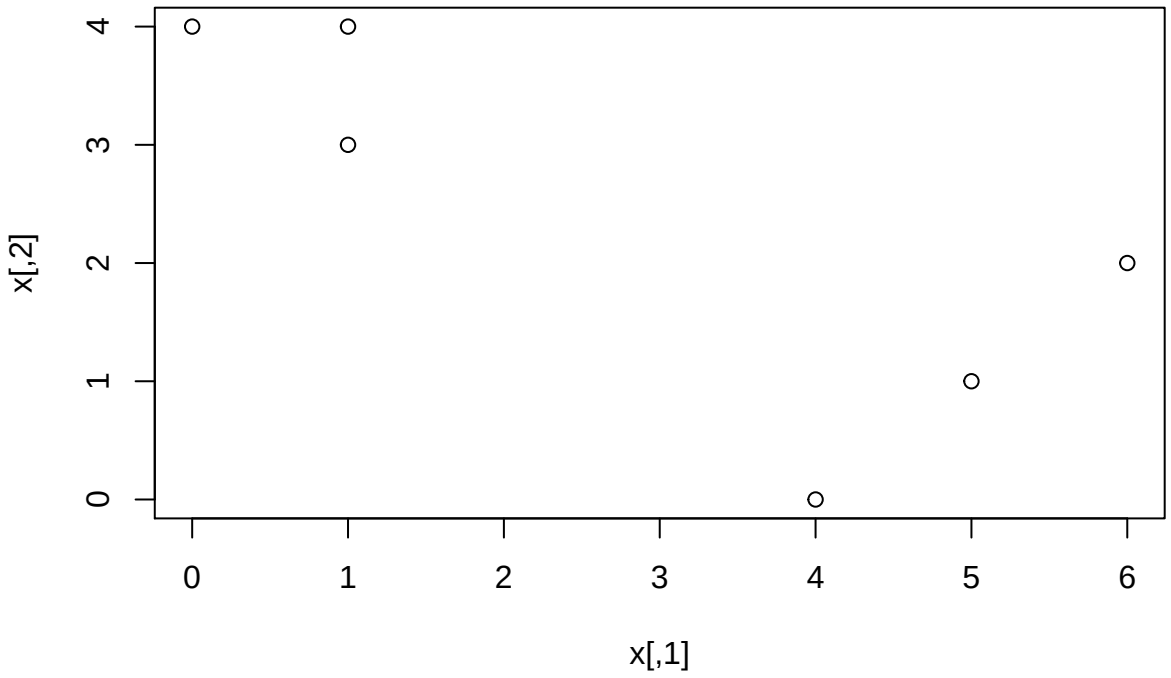
2/29/2020

Q1

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

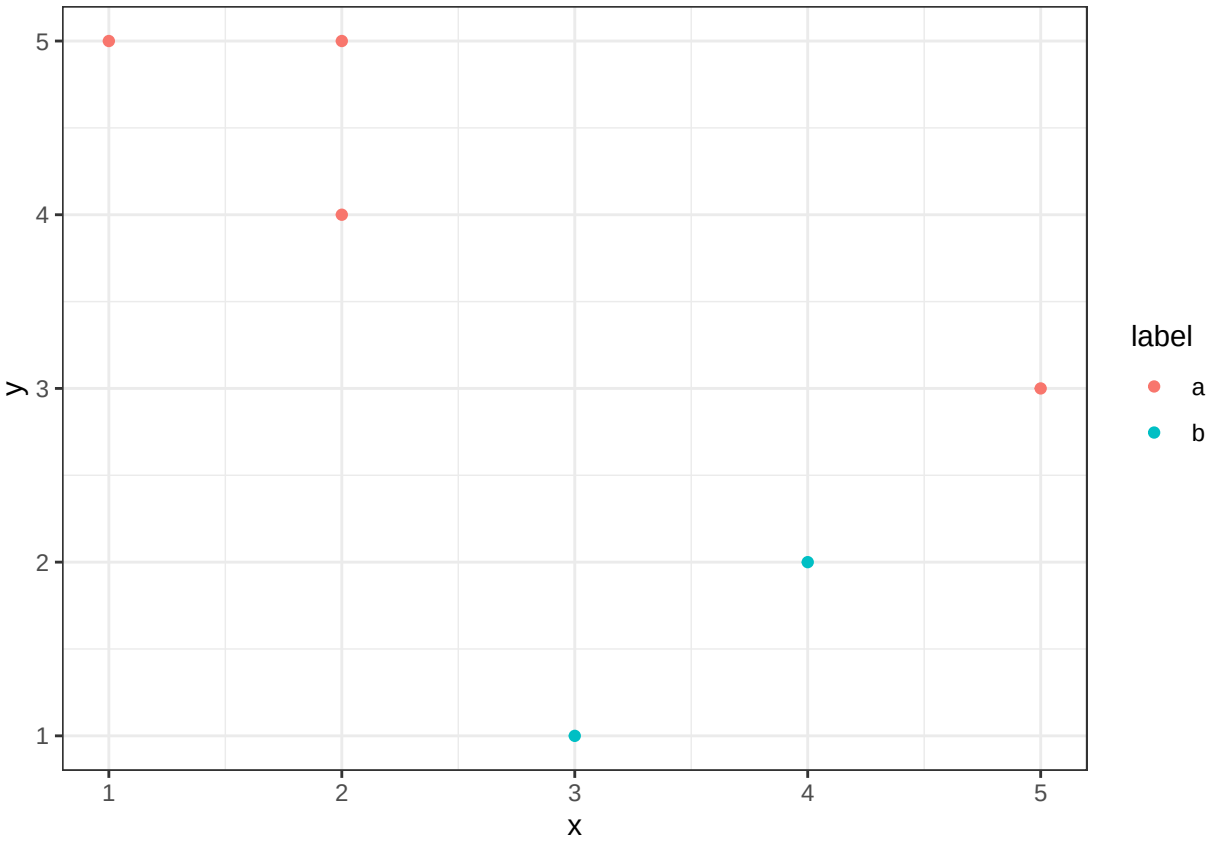
1. (5 points) Plot the observations.

```
plot(x)
```



2. (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*).

```
set.seed(123)
label <- sample(c('a','b'), size = nrow(x), replace = TRUE)
newData <- cbind(x, label) %>%
  data.frame() %>%
  rename(`x` = V1,
         `y` = V2)
cols.num <- c("x","y")
newData[cols.num] <- sapply(newData[cols.num], as.numeric)
newData %>%
  ggplot(aes(x, y, color = label)) +
  geom_point() +
  theme_bw()
```



3. (10 points) Compute the centroid for each cluster.

```
newData %>%
  group_by(label) %>%
  summarize_all(~mean(.)) %>%
  kableExtra::kable()
```

label	x	y
a	2.5	4.25
b	3.5	1.50

4. (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.

```
renew <- function(data){
  cent <- data %>%
  group_by(label) %>%
  summarize_all(~mean(.))
  data %<>% mutate(label = ifelse(
    ((data$x - cent$x[1])^2 + (data$y - cent$y[1])^2) < ((data$x - cent$x[2])^2 + (data$y -cent$y[2])^2
    'a', 'b'))
  return(data)
}

kableExtra:: kable(renew(newData))
```

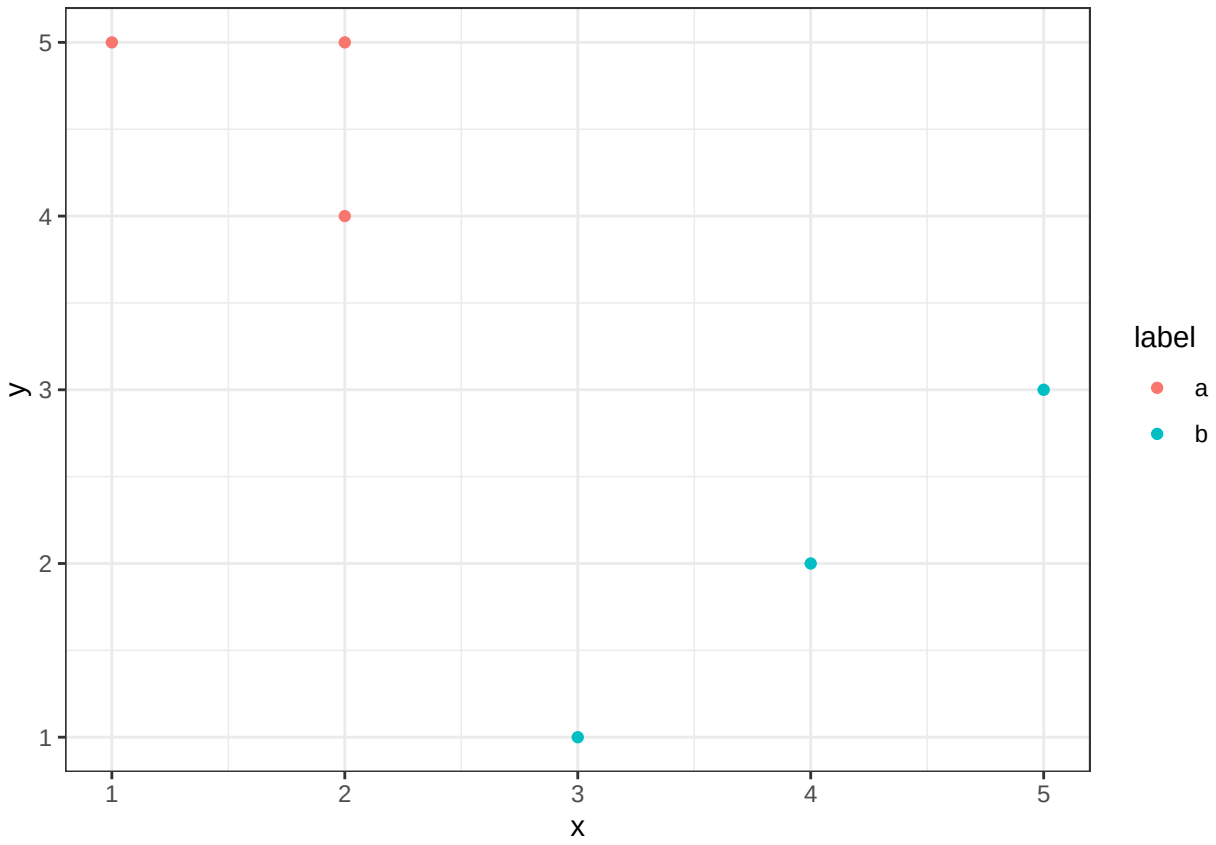
x	y	label
2	5	a
2	4	a
1	5	a
4	2	b
5	3	b
3	1	b

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

```
same = F
while(! same) {
  old_label <- newData$label
  newData <- renew(newData)
  same <- all(old_label == newData$label)
}
```

6. (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.

```
newData %>%
  ggplot(aes(x,y, color = label)) +
  geom_point() +
  theme_bw()
```



Q2

Clustering State Legislative Professionalism

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

```
load("./Data and Codebook/legprof-components.v1.0.RData")
```

2. (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)

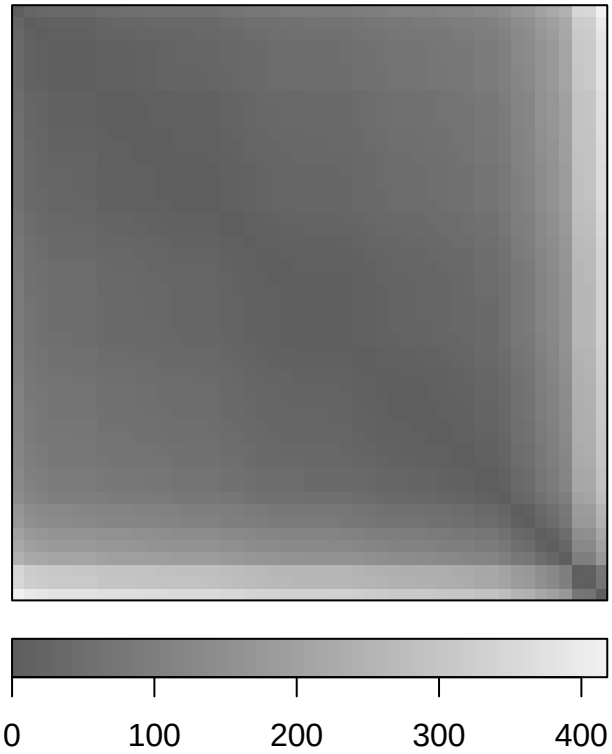
```
legit <- x %>%
  as.tibble() %>%
  filter(sessid == '2009/10') %>%
  select(stateabv, t_slength, slength, salary_real, expend) %>%
  na.omit() %>%
  remove_rownames() %>%
  column_to_rownames(., var = "stateabv") ## set state abvreviations as index
```

```
## Warning: `as.tibble()` is deprecated, use `as_tibble()` (but mind the new semantics).
## This warning is displayed once per session.
```

```
legit[,-1] <- scale(legit[,-1] )
```

3. (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.

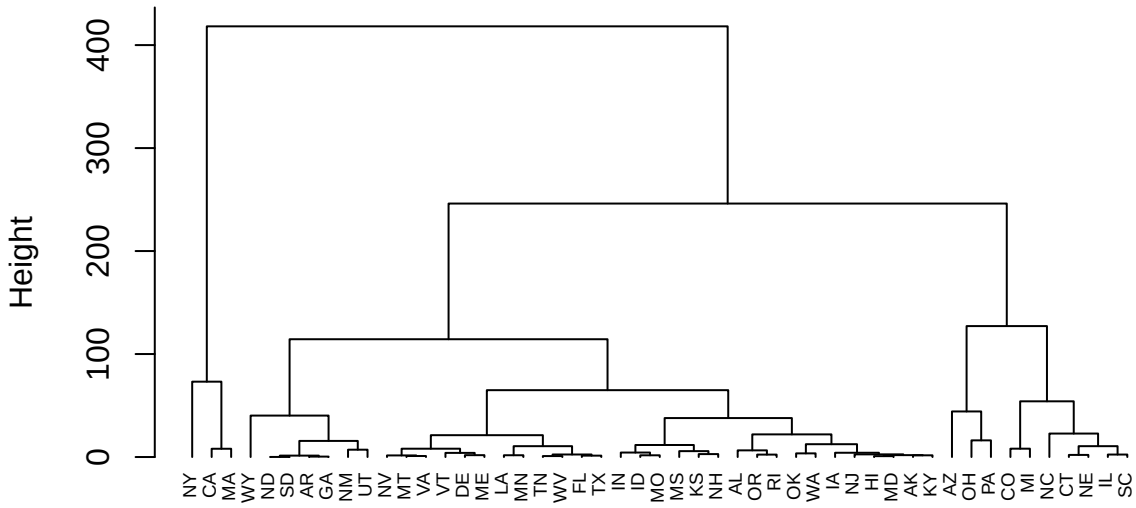
```
seriation::dissplot(legit %>% dist())
```



There are a few squares shown in the ODI and a diagonal that is visible. However, it is still difficult for clusterability and natural, non-random structure to exist.

4. (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.

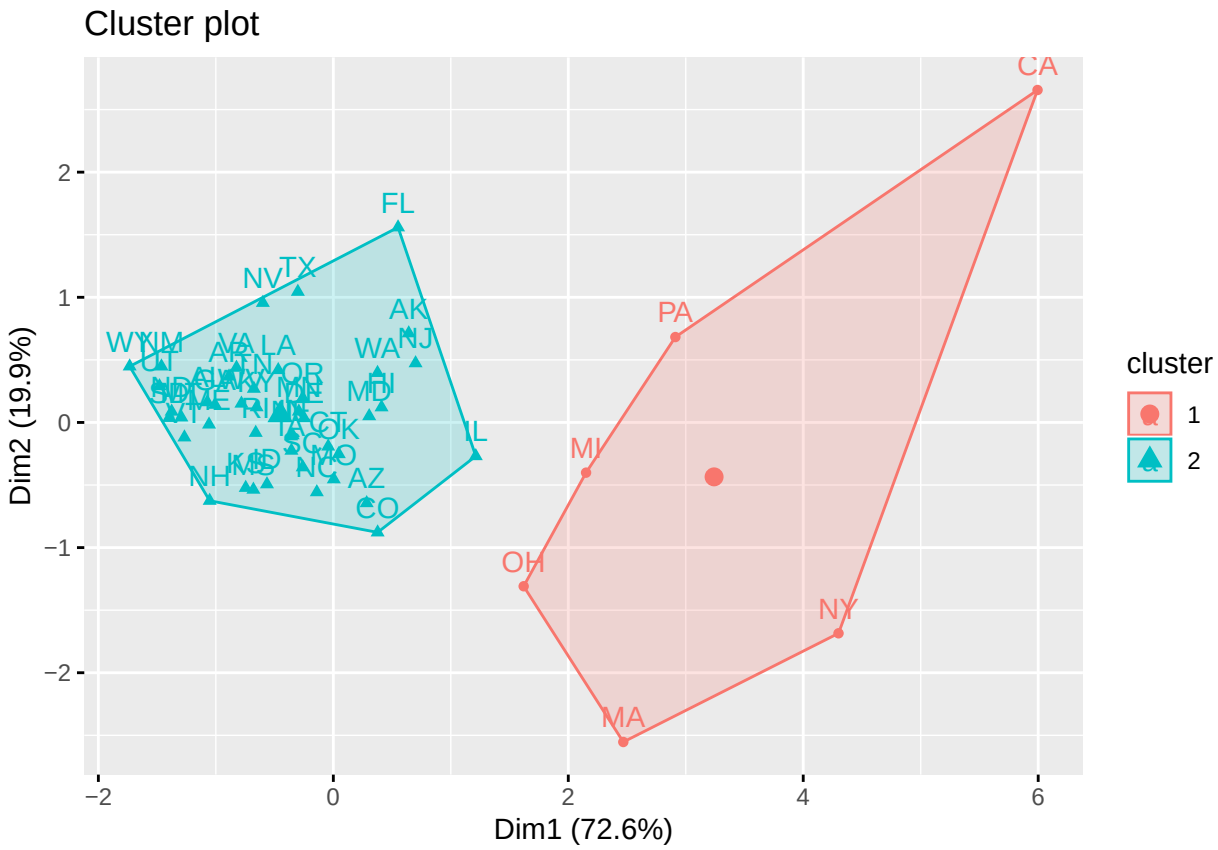
```
legit_tree <- legit %>%
  dist() %>%
  hclust(method = "complete")
plot(legit_tree, cex = 0.6, hang = -1)
```



The dendrogram shows a clustering result. If we split the tree in to two, we see that only three states(NY, CA, MA) are clustered under group1, and the rest states are all clustered under group 2.

5. (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.

```
k2 <- kmeans(legit[,-1], centers = 2, nstart = 25)
fviz <- fviz_cluster(k2, data = legit[,-1])
fviz
```



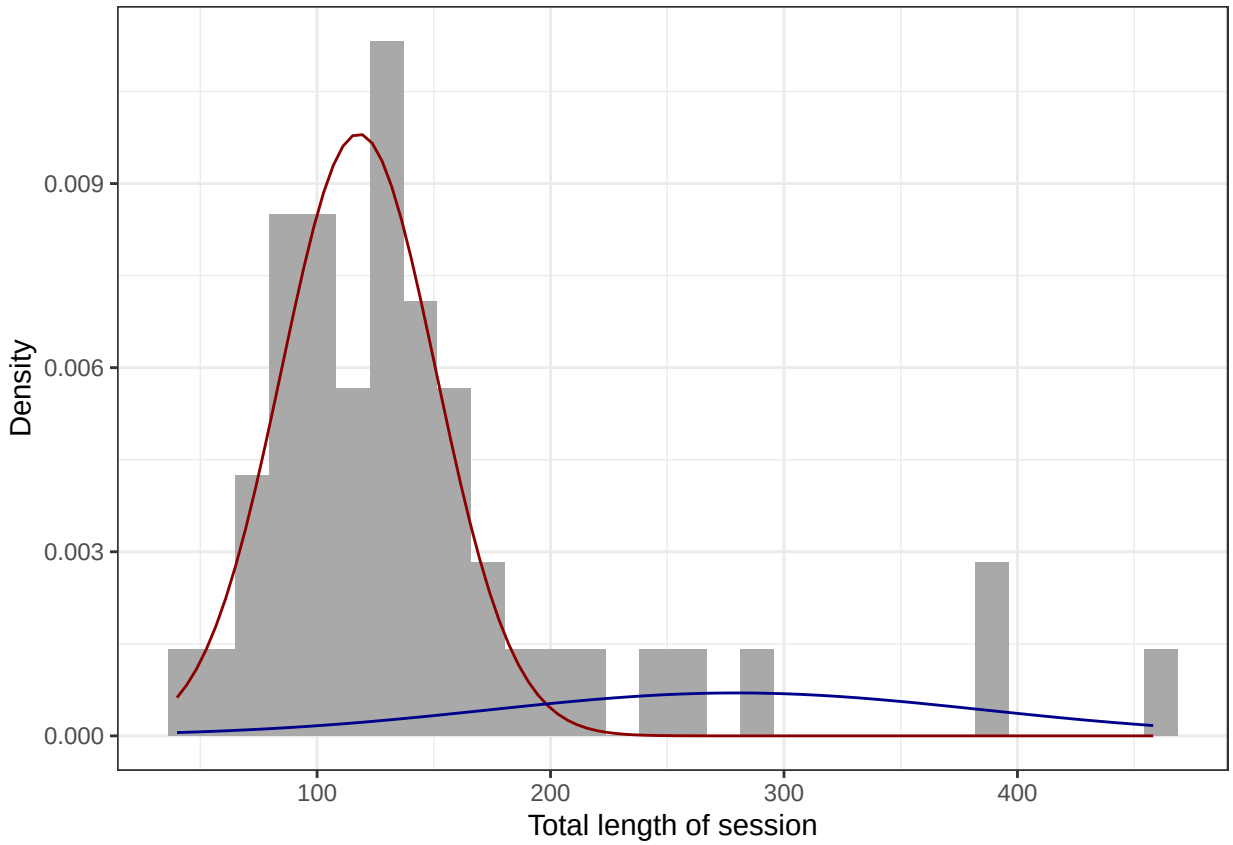
The result is a different visualization of clustering. Similar to the result from agglomerative hierarchical clustering algorithm, the result from kmean shows that only a few states (CA, PA, MI, OH, MA, NY) are clustered as one group, and the rest states are clustered as another group. However, it also differenciates from previous result that other than NY, CA, MA, three more states (MI, OH, PA) are also joining the red cluster.

6. (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.

```
gmm1 <- normalmixEM(legit$t_slength, k = 2) # fit the GMM using EM and 2 comps
```

```
## number of iterations= 37
```

```
p1 <- ggplot(data.frame(x = gmm1$x)) +
  geom_histogram(aes(x, ..density..), fill = "darkgray") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm1$mu[1], gmm1$sigma[1], lam = gmm1$lambda[1]),
                colour = "darkred") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm1$mu[2], gmm1$sigma[2], lam = gmm1$lambda[2]),
                colour = "darkblue") +
  xlab("Total length of session") +
  ylab("Density") +
  theme_bw()
p1
```



From the plot we can see that similar to the previous results, majority of states are clustered under red group, and a few states are clustered under blue group. The distribution of red group shows that the mean of total length of session of red group is around ~110, which is lower than the mean of total length of session from blue group.

7. (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.

```
set.seed(1234) # set seed for iterations to return to same place

gmm2 <- normalmixEM(legit$slength, k = 2)
```

```
## number of iterations= 55
```

```
p2 <- ggplot(data.frame(x = gmm2$x)) +
  geom_histogram(aes(x, ..density..), fill = "darkgray") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm2$mu[1], gmm2$sigma[1], lam = gmm2$lambda[1]),
                colour = "darkred") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm2$mu[2], gmm2$sigma[2], lam = gmm2$lambda[2]),
                colour = "darkblue") +
  xlab("Length of regular session") +
  ylab("Density") +
  ylim(0,1) +
```

```
  theme_bw()

gmm3 <- normalmixEM(legit$salary_real, k = 2)
```

```
## number of iterations= 59
```

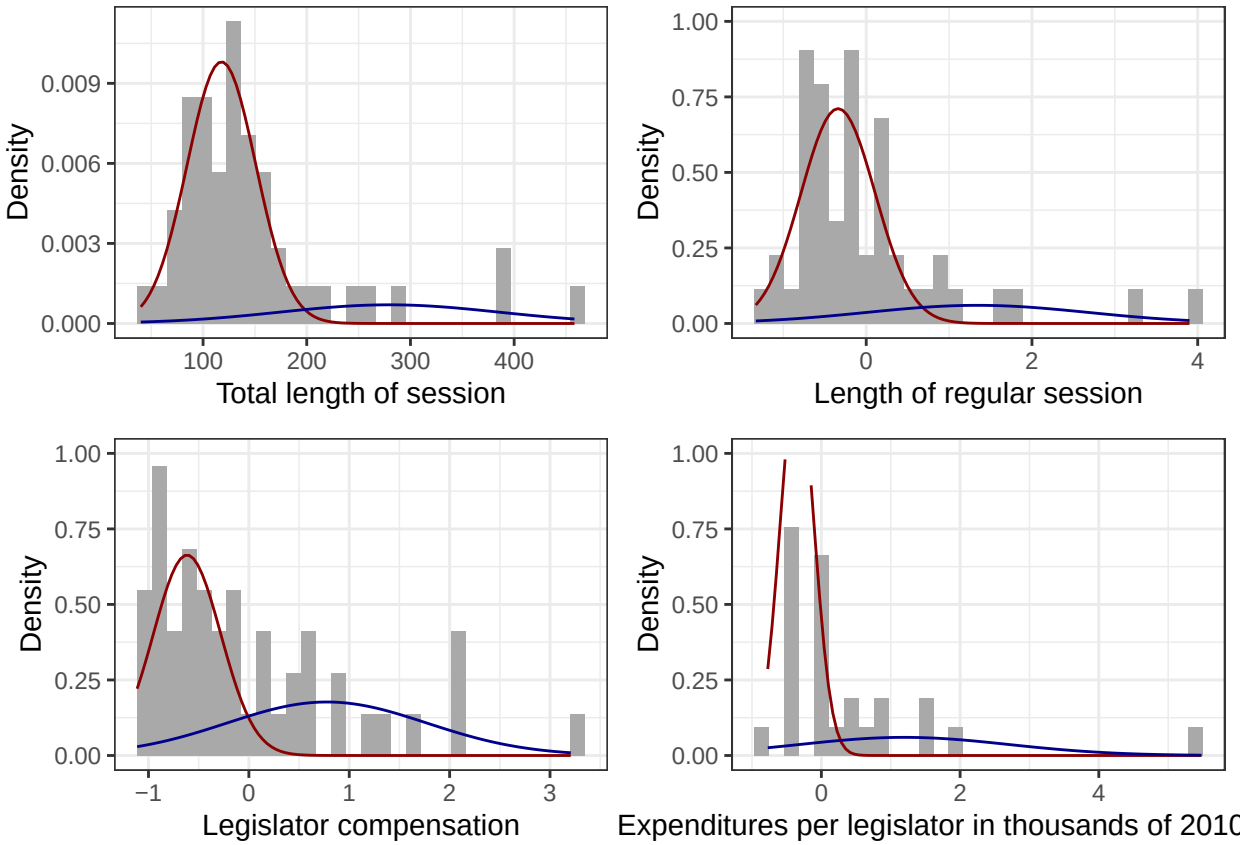
```
p3 <- ggplot(data.frame(x = gmm3$x)) +
  geom_histogram(aes(x, ..density..), fill = "darkgray") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm3$mu[1], gmm3$sigma[1], lam = gmm3$lambda[1]),
                colour = "darkred") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm3$mu[2], gmm3$sigma[2], lam = gmm3$lambda[2]),
                colour = "darkblue") +
  xlab("Legislator compensation") +
  ylab("Density") +
  ylim(0,1) +
  theme_bw()

gmm4 <- normalmixEM(legit$expend, k = 2)
```

```
## number of iterations= 19
```

```
p4 <- ggplot(data.frame(x = gmm4$x)) +
  geom_histogram(aes(x, ..density..), fill = "darkgray") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm4$mu[1], gmm4$sigma[1], lam = gmm4$lambda[1]),
                colour = "darkred") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm4$mu[2], gmm4$sigma[2], lam = gmm4$lambda[2]),
                colour = "darkblue") +
  xlab("Expenditures per legislator in thousands of 2010 dollars") +
  ylab("Density") +
  ylim(0,1) +
  theme_bw()

egg::ggarrange(p1,p2,p3,p4)
```



From the four plots above, we can see that there is no significant difference from different features under GMM. It shows that a big number of states are clustered under group 1, regardless of different features. Perhaps only under legislator compensation feature, there are slightly more states clustered under the blue group.

8. (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.

```
cl_validation <- clValid(legit, 2:5,
                         clMethods = c("hierarchical", "kmeans", "model"),
                         validation = "internal")
summary(cl_validation)
```

```
##
## Clustering Methods:
##  hierarchical kmeans model
##
## Cluster sizes:
##  2 3 4 5
##
## Validation Measures:
##                                  2        3        4        5
##
```

```
## hierarchical Connectivity   4.6536 10.0290 12.5290 15.2179
##              Dunn           0.4019  0.1654  0.1654  0.0739
##              Silhouette     0.7808  0.6656  0.6556  0.5939
## kmeans       Connectivity   7.0849 11.5317 14.0317 15.2179
##              Dunn           0.1343  0.0944  0.0944  0.0739
##              Silhouette     0.7472  0.6470  0.6354  0.5939
## model        Connectivity   9.6282 67.6294 69.1849 84.7135
##              Dunn           0.0742  0.0032  0.0027  0.0038
##              Silhouette     0.7385 -0.0120  0.0418 -0.1473
##
## Optimal Scores:
##
##              Score  Method       Clusters
## Connectivity 4.6536 hierarchical 2
## Dunn         0.4019 hierarchical 2
## Silhouette   0.7808 hierarchical 2
```

9. (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?

For the validation measure, I choose internal measure, which relies on information in the data only, that is the characteristics of the clusters themselves, such as compactness and separation.The validation result shows that agglomerative hierarchical clustering algorithm performs the best clusters where k = 2. For connectivity, hierarchical clustering algorithm has the lowest value, which means that it has the lowest connectedness of clusters; for Dunn, hierarchical clustering algorithm has the highest value, meaning that the smallest distance between observations not in the same cluster to the largest intra-cluster distance is the largest among all three algorithms.; for Silhouette, hierarchical clustering algorithm has the highest value that closes to 1.

However, it does not mean that hierarchical clustering algorithm could always perform the best under all situations. Sometimes it is more reasonable to select sub-optimal model. We have to be aware that each algorithm has strengths and weaknesses. When data size is large and with high dimensional, kmean could be a better option than hierarchical clustering algorithm. Unlike kmeans, GMM does not bias the cluster sizes to have specific structures.