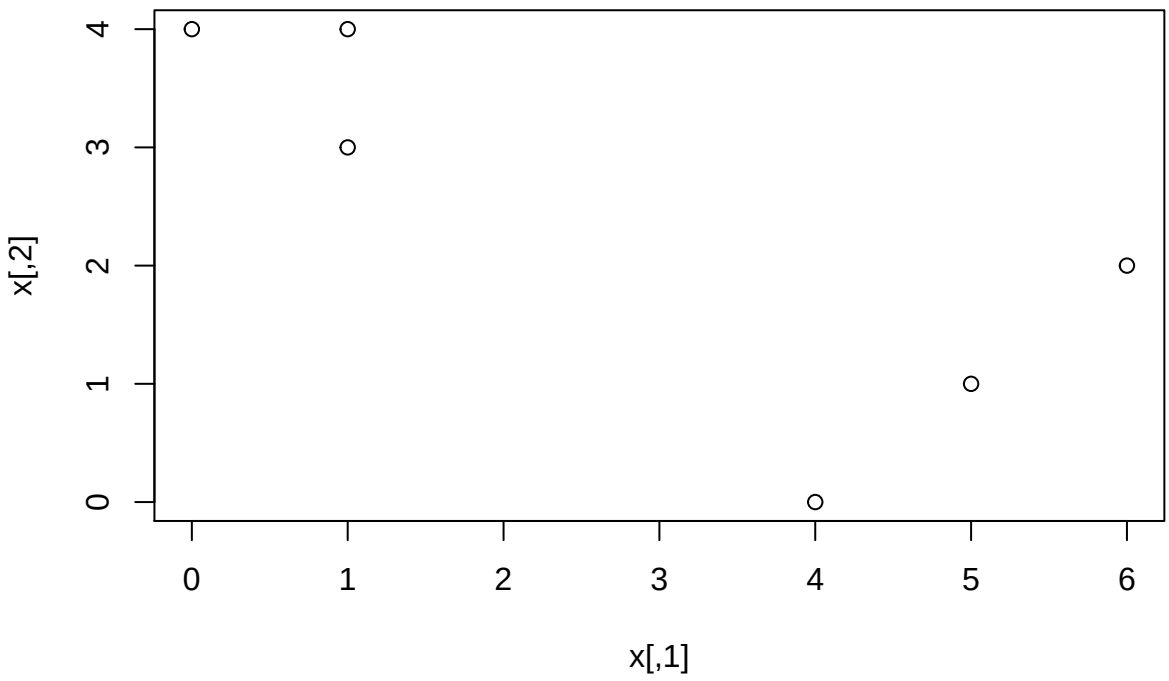
Problem Set 4: Clustering

John Kim

2020 2 24

1. Plot the observations.

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

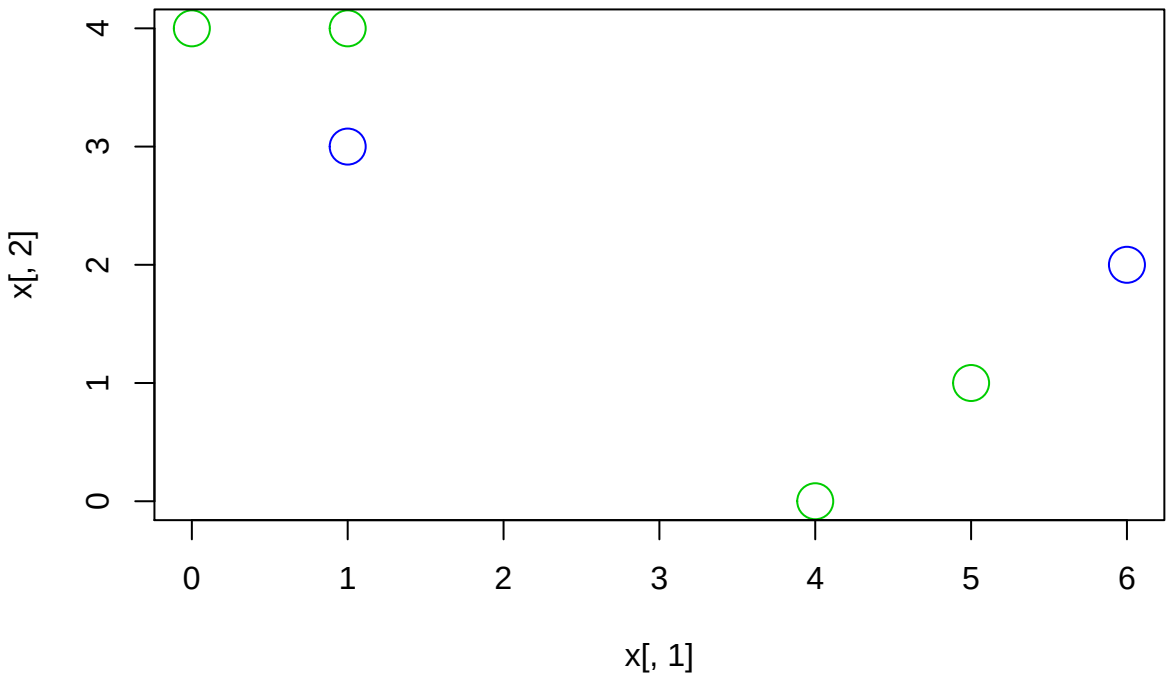


2. 1 2 1 1 2 1

```
set.seed(1)
lab <- sample(2, nrow(x), replace = TRUE)
lab
```

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

```
plot(x[,1], x[,2], col=(lab+2), cex = 2.5)
```



3. [1] 2.50 2.25 [1] 3.5 2.5

```
cent.1 <- c(mean(x[lab == 1, 1]), mean(x[lab ==1, 2]))
cent.2 <- c(mean(x[lab == 2, 1]), mean(x[lab ==2, 2]))
cent.1
```

```
## [1] 2.50 2.25
```

```
cent.2
```

```
## [1] 3.5 2.5
```

4. 1 1 1 2 2 2

```
euclid <- function(x, y) {
  return(sqrt((x[1] - y[1])^2 + (x[2]-y[2])^2))
}

assign <- function(x, cent.1, cent.2) {
  labels = rep(NA, nrow(x))
  for (i in 1:nrow(x)) {
    if (euclid(x[i,], cent.1) < euclid(x[i,], cent.2)) {
      labels[i] = 1
    } else {
      labels[i] = 2
    }
  }
  return(labels)
```

```
}

labels <- assign(x, cent.1, cent.2)
labels
```

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

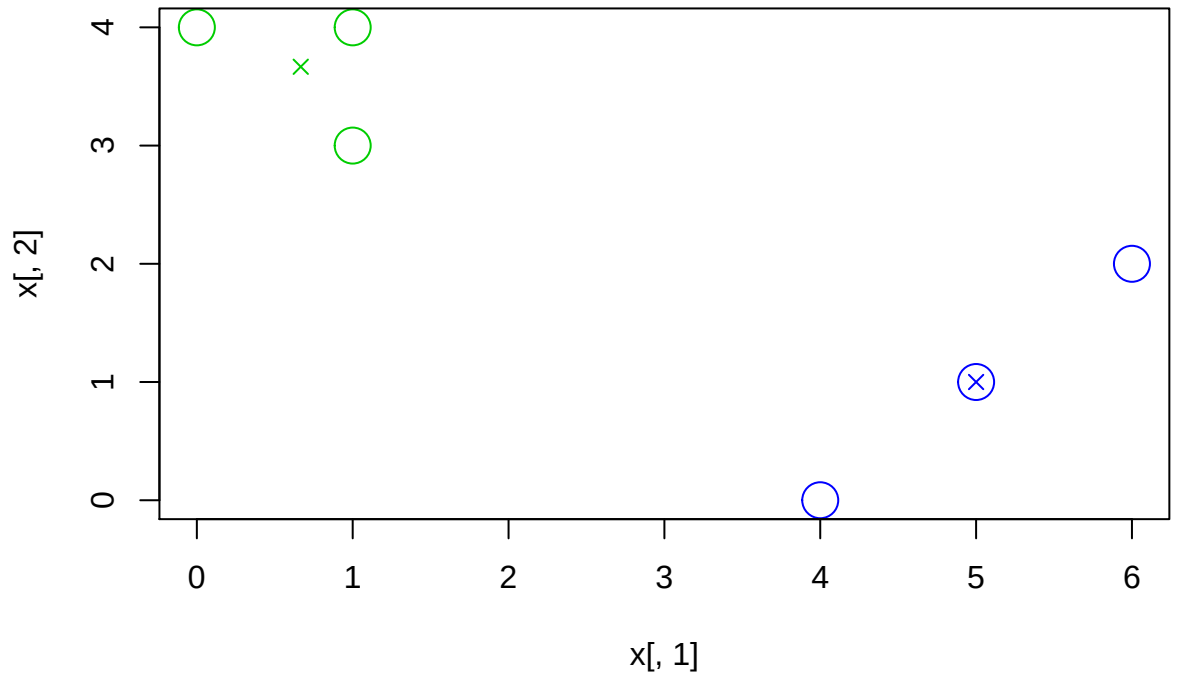
5. [1] 0.6666667 3.6666667 [1] 5 1

```
last_labels = rep(-1, 6)
while (!all(last_labels == labels)) {
  last_labels = labels
  cent.1 = c(mean(x[labels==1, 1]), mean(x[labels==1, 2]))
  cent.2 = c(mean(x[labels==2, 1]), mean(x[labels==2, 2]))
  print(cent.1)
  print(cent.2)
  labels = assign(x, cent.1, cent.2)
}
```

```
## [1] 0.6666667 3.6666667
## [1] 5 1
```

6. Plot below

```
plot(x[,1], x[,2], col=(labels+2), cex=2.5)
points(cent.1[1], cent.1[2], col=3, pch=4)
points(cent.2[1], cent.2[2], col=4, pch=4)
```



Clustering State Legislative Professionalism 1. Load the data

```
load("C:/Users/john_kim/Downloads/legprof-components.v1.0.RData")
data4 <- x
```

2. 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)

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.4
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------------------------------ ti
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
data4.1 <- subset(data4, sessid == "2009/10", select = c("stateabv", "t_slength", "slength"
                                                         , "salary_real", "expend", "year", "mds1", "mds
data4.1 <- data4.1[complete.cases(data4.1), ]
data4.1sn <- subset(data4.1, select = "stateabv")
data4.1 <- scale(subset(data4.1, select = c("t_slength", "slength", "salary_real", "expend")))
```

3. The plot below, we can observe a few rectangles covered in white lines. This indicates the clusterability in the data.

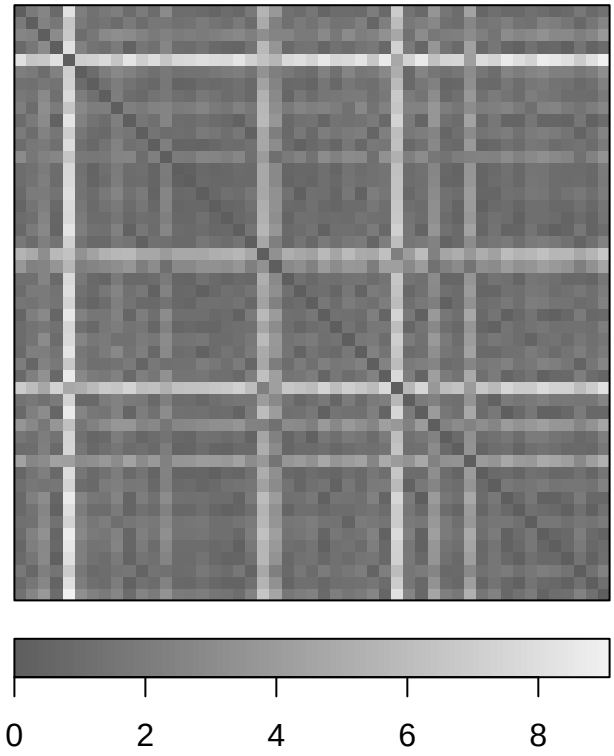
```
library(seriation)
```

```
## Registered S3 method overwritten by 'seriation':
##   method         from
##   reorder.hclust gclus
```

```
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
dist_data4 <- data4.1 %>%
  dist()
dissplot(dist_data4, method = NA)
```



4. I fit the agglomerative hierachical clustering algorithm with the linkage of "complete" and the result is shown as below. At height 8, the data can be divided into two clusters, whereas at height 6, it can be separated into three clusters. If we divide this into two clusters using hierarchical clustering, then, one cluster will have 42 and the other 7.

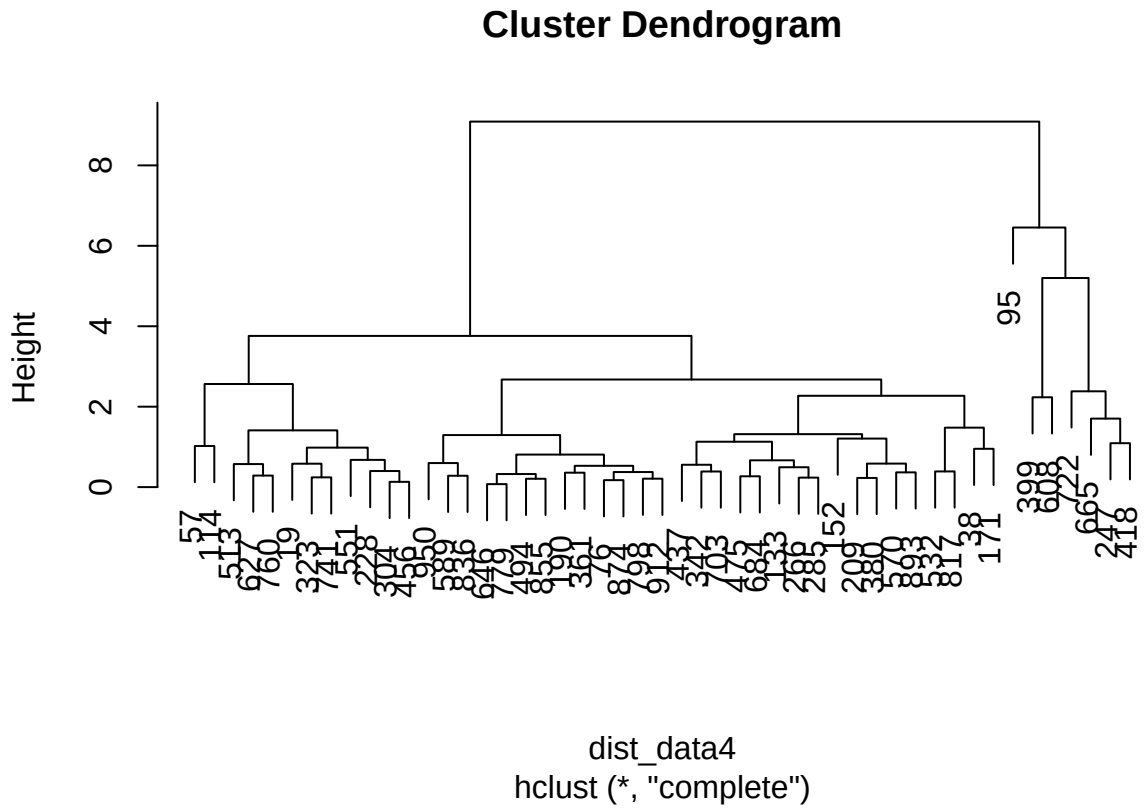
```
library(skimr)
library(dendextend)
```

```
##
## ---------------------
## Welcome to dendextend version 1.13.3
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues
## Or contact: <tal.galili@gmail.com>
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ---------------------

##
## Attaching package: 'dendextend'

## The following object is masked from 'package:stats':
##
##     cutree
```

```
clust <- hclust(dist_data4, method = "complete")
plot(clust)
```



5. From the below analysis, we can observe that one cluster contains 6 states and the other contains 43.

```
kmeans.4 <- kmeans(data4.1, centers = 2, nstart = 20)
str(kmeans.4)
```

```
## List of 9
##  $ cluster     : Named int [1:49] 1 1 1 1 2 1 1 1 1 1 ...
##   ..- attr(*, "names")= chr [1:49] "19" "38" "57" "76" ...
##  $ centers     : num [1:2, 1:4] -0.293 2.1 -0.293 2.101 -0.283 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "1" "2"
##   .. ..$ : chr [1:4] "t_slength" "slength" "salary_real" "expend"
##  $ totss       : num 192
##  $ withinss    : num [1:2] 48.4 40.4
##  $ tot.withinss: num 88.7
##  $ betweenss   : num 103
##  $ size        : int [1:2] 43 6
##  $ iter        : int 1
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"
```

```
data4.1sn$k_cluster <- as.factor(kmeans.4$cluster)
```

6. The gmm method again divides the 49 states into 6 and 43.

```r
library(mixtools)
```

```
## mixtools package, version 1.2.0, Released 2020-02-05
## This package is based upon work supported by the National Science Foundation under Grant No. SES-0518
```

```r
library(plotGMM)
set.seed(3)
gmm4 <- mvnormalmixEM(data4.1, k = 2)
```

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

```r
posterior <- data.frame(cbind(gmm4$x, gmm4$posterior))
posterior$component <- ifelse(posterior$comp.1 > 0.5, 1, 2)
gmm_cluster <- as.factor(posterior$component)
table(posterior$component)
```

```
##
##  1  2
##  6 43
```

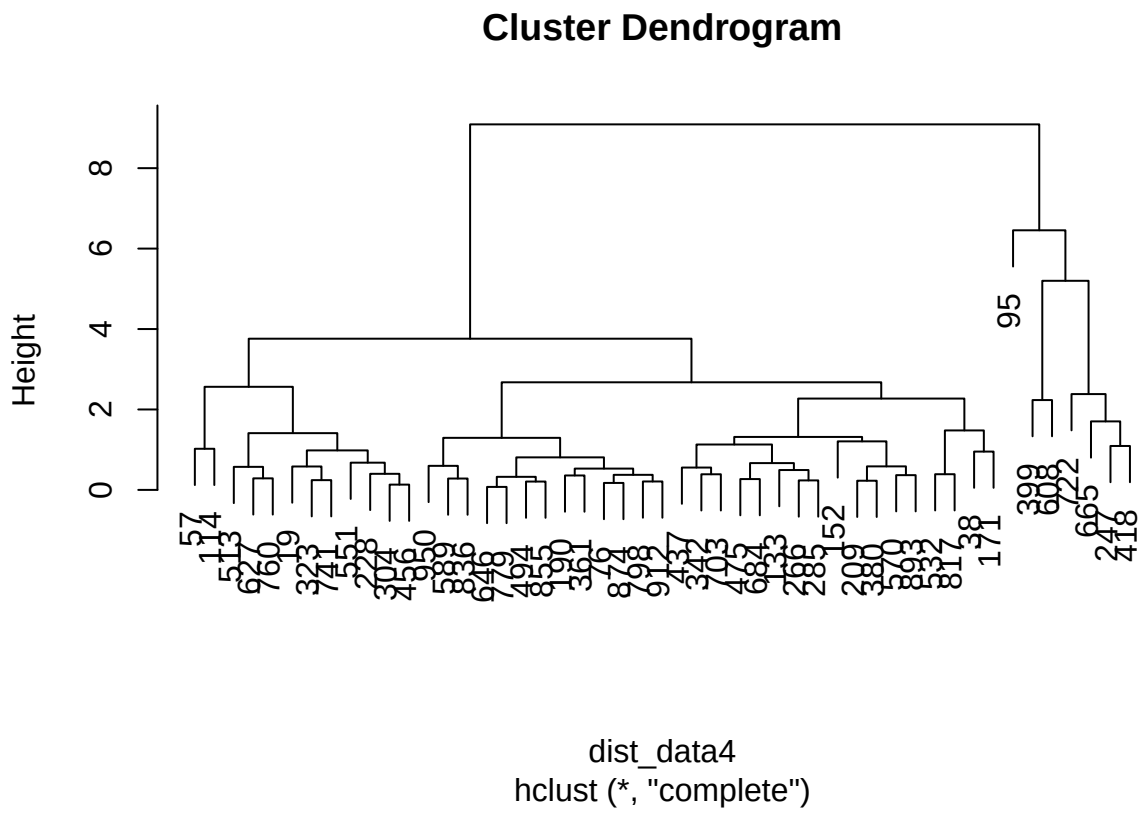
```r
cbind(data4.1sn, gmm_cluster)
```

```
##     stateabv k_cluster gmm_cluster
## 19        AL         1           2
## 38        AK         1           2
## 57        AZ         1           2
## 76        AR         1           2
## 95        CA         2           1
## 114       CO         1           2
## 133       CT         1           2
## 152       DE         1           2
## 171       FL         1           2
## 190       GA         1           2
## 209       HI         1           2
## 228       ID         1           2
## 247       IL         1           2
## 266       IN         1           2
## 285       IA         1           2
## 304       KS         1           2
## 323       KY         1           2
## 342       LA         1           2
## 361       ME         1           2
## 380       MD         1           2
## 399       MA         2           1
## 418       MI         2           1
## 437       MN         1           2
## 456       MS         1           2
## 475       MO         1           2
## 494       MT         1           2
## 513       NE         1           2
## 532       NV         1           2
## 551       NH         1           2
## 570       NJ         1           2
## 589       NM         1           2
## 608       NY         2           1
## 627       NC         1           2
```

```
## 646        ND          1           2
## 665        OH          2           1
## 684        OK          1           2
## 703        OR          1           2
## 722        PA          2           1
## 741        RI          1           2
## 760        SC          1           2
## 779        SD          1           2
## 798        TN          1           2
## 817        TX          1           2
## 836        UT          1           2
## 855        VT          1           2
## 874        VA          1           2
## 893        WA          1           2
## 912        WV          1           2
## 950        WY          1           2
```

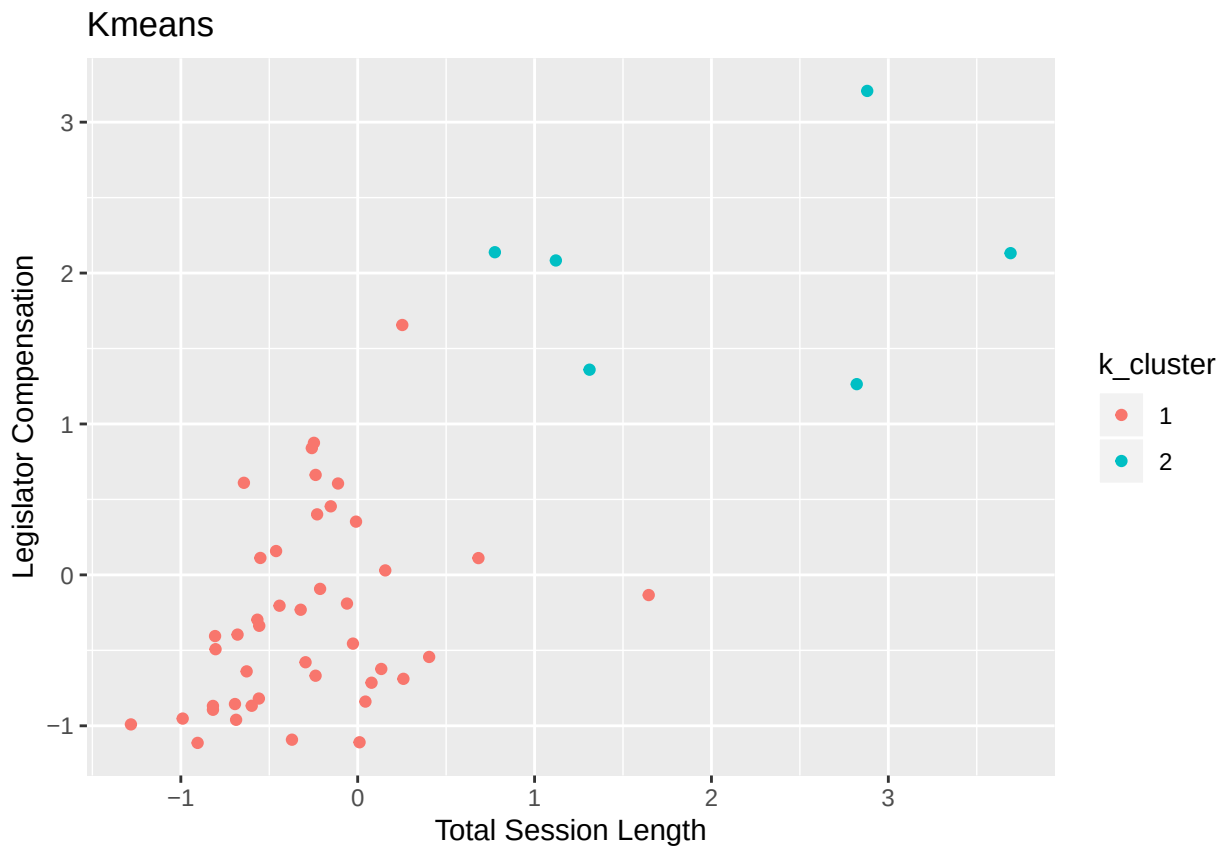
7. We can observe that the plots for kmeans and gmm clustering are identical. They both have six states in one cluster and 42 states in the other one. On the other hand, the hierarchical clustering has seven states in one cluster and 42 on the other side.

```
plot(clust)
```

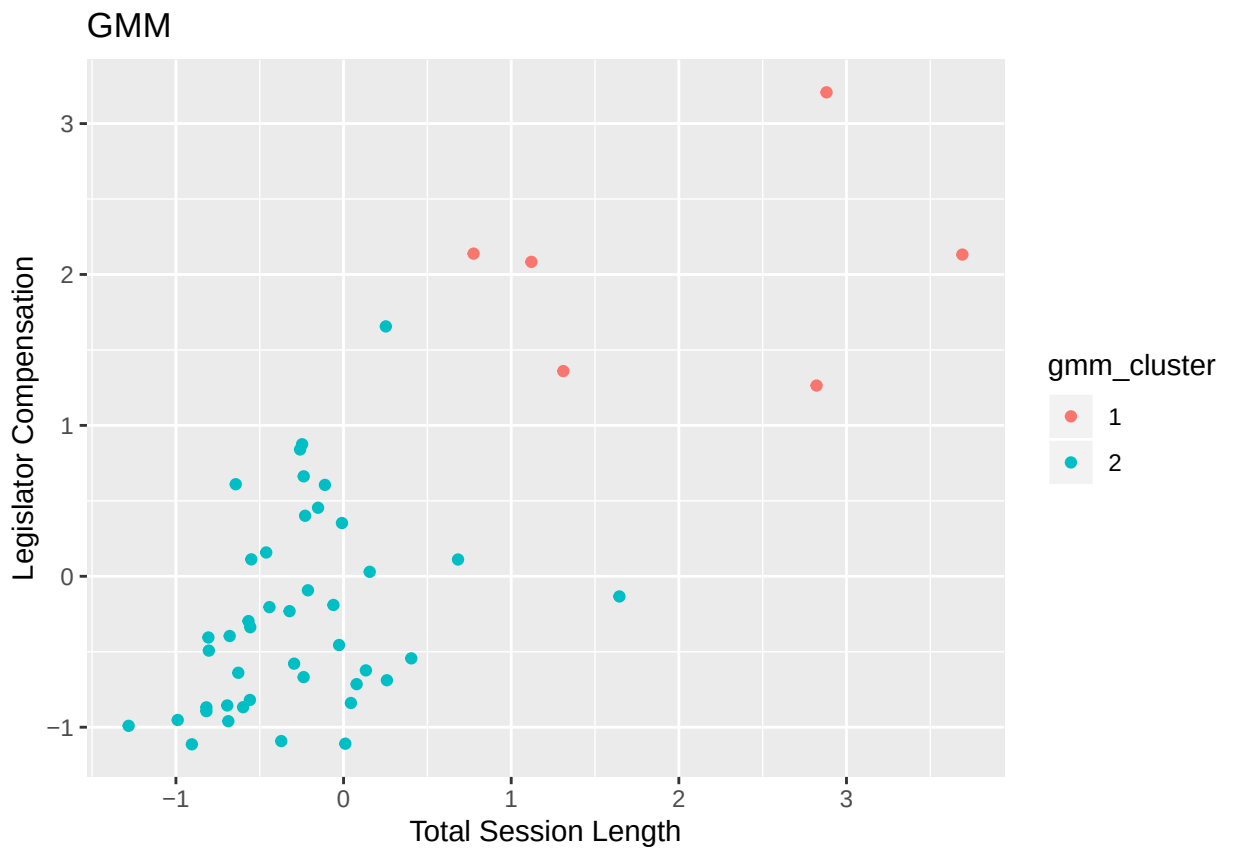



```
data4.2 <- cbind(data4.1sn, data4.1)
data4.2 %>%
  ggplot(aes(x = t_slength, y = salary_real, color = k_cluster)) +
  geom_point() +
```

```
  labs(x = "Total Session Length",
       y = "Legislator Compensation",
       title = "Kmeans")
```




```
data4.2 %>%
  ggplot(aes(x = t_slength, y = salary_real, color = gmm_cluster)) +
  geom_point() +
  labs(x = "Total Session Length",
       y = "Legislator Compensation",
       title = "GMM")
```



8.The Dunn index indicates the ratio of the shortest distance from within-cluster to the largest distance from inter-cluster. Therefore, the larger the Dunn index, the better the fit. Comparing the Dunn index across the methods, the hierachical clustering has that of 0.5872, and kmeans 0.1282 and gmm 0.0841. For this example, the hierarchical has the best internal valdity and it is followed by kmeans, and then gmm.

```
library(mclust)
```

```
## Package 'mclust' version 5.4.5
## Type 'citation("mclust")' for citing this R package in publications.
```

```
##
## Attaching package: 'mclust'
```

```
## The following object is masked from 'package:mixtools':
##
##     dmvnorm
```

```
## The following object is masked from 'package:purrr':
##
##     map
```

```
library(clValid)
```

```
## Loading required package: cluster
```

```
hcval <- clValid(data4.1, nClust = 2, clMethods = "hierarchical", validation = "internal", method = "co
kmval <- clValid(data4.1, nClust = 2, clMethods = "kmeans", validation = "internal")
gmmval <- clValid(data4.1, nClust = 2, clMethods = "model", validation = "internal")
summary(hcval)
```

```
##
## Clustering Methods:
##  hierarchical
##
## Cluster sizes:
##  2
##
## Validation Measures:
##                                 2
##
## hierarchical Connectivity  7.9071
##              Dunn          0.1673
##              Silhouette    0.6204
##
## Optimal Scores:
##
##              Score  Method       Clusters
## Connectivity 7.9071 hierarchical 2
## Dunn         0.1673 hierarchical 2
## Silhouette   0.6204 hierarchical 2
```

```
summary(kmval)
```

```
##
## Clustering Methods:
##  kmeans
##
## Cluster sizes:
##  2
##
## Validation Measures:
##                            2
##
## kmeans Connectivity  8.4460
##        Dunn          0.1735
##        Silhouette    0.6458
##
## Optimal Scores:
##
##              Score  Method Clusters
## Connectivity 8.4460 kmeans 2
## Dunn         0.1735 kmeans 2
## Silhouette   0.6458 kmeans 2
```

```
summary(gmmval)
```

```
##
## Clustering Methods:
##  model
##
## Cluster sizes:
##  2
##
## Validation Measures:
##                            2
```

```
##
## model Connectivity  10.7393
##       Dunn           0.1522
##       Silhouette     0.6314
##
## Optimal Scores:
##
##              Score   Method Clusters
## Connectivity 10.7393 model  2
## Dunn          0.1522 model  2
## Silhouette    0.6314 model  2
```

9. (10 points) Discuss the validation output, e.g.,

Without the label, we should compare internal validity using silhouette width, connectivity, and Dunn index to test different numbers and types of cluster. In this example, hierarchical clustering had the best fit among the three methods. The reason we could imagine selecting a technically "suboptimal" partitioning method, regardless of the validation statistics is that, however, we might not be able to find the strongest model for all the dimensions. Therefore, it will depend on the situation or our goal for this.