MACS 33002: Homework #4

Yujiao Song

Due Monday, Mar 2nd by 5pm

```
## -- Attaching packages ---------------------------- tidyverse 1.3.0 --

## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts ------------------------------- tidyverse_conflicts() --
## x dplyr::combine() masks gridExtra::combine()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()

## Loading required package: lattice

## Loading required package: ggformula

## Loading required package: ggstance

##
## Attaching package: 'ggstance'

## The following objects are masked from 'package:ggplot2':
##
##     geom_errorbarh, GeomErrorbarh

##
## New to ggformula?  Try the tutorials:
##  learnr::run_tutorial("introduction", package = "ggformula")
##  learnr::run_tutorial("refining", package = "ggformula")

## Loading required package: mosaicData

## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Registered S3 method overwritten by 'mosaic':
##   method                           from
##   fortify.SpatialPolygonsDataFrame ggplot2

##
## The 'mosaic' package masks several functions from core packages in order to add
## additional features.  The original behavior of these functions should not be affected by
##
## Note: If you use the Matrix package, be sure to load it BEFORE loading mosaic.

##
## Attaching package: 'mosaic'

## The following object is masked from 'package:Matrix':
##
##     mean

## The following objects are masked from 'package:dplyr':
##
##     count, do, tally

## The following object is masked from 'package:purrr':
##
##     cross

## The following object is masked from 'package:ggplot2':
##
##     stat

## The following objects are masked from 'package:stats':
##
##     binom.test, cor, cor.test, cov, fivenum, IQR, median, prop.test,
##     quantile, sd, t.test, var

## The following objects are masked from 'package:base':
##
##     max, mean, min, prod, range, sample, sum
```

```
##
## Attaching package: 'modelr'

## The following object is masked from 'package:broom':
##
##     bootstrap

## The following object is masked from 'package:mosaic':
##
##     resample

## The following object is masked from 'package:ggformula':
##
##     na.warn

## Loading required package: carData

##
## Attaching package: 'car'

## The following objects are masked from 'package:mosaic':
##
##     deltaMethod, logit

## The following object is masked from 'package:dplyr':
##
##     recode

## The following object is masked from 'package:purrr':
##
##     some

## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2

##
## Attaching package: 'GGally'

## The following object is masked from 'package:dplyr':
##
##     nasa

## dummies-1.5.6 provided by Decision Patterns
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select

##
## Attaching package: 'rcfss'

## The following object is masked from 'package:modelr':
##
##     mse

## For binary classification, the first factor level is assumed to be the event.
## Set the global option `yardstick.event_first` to `FALSE` to change this.

##
## Attaching package: 'yardstick'

## The following objects are masked from 'package:modelr':
##
##     mae, mape, rmse

## The following object is masked from 'package:readr':
##
##     spec

## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli

##
## Attaching package: 'skimr'

## The following object is masked from 'package:mosaic':
##
##     n_missing

##
## ----------------------
## 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
## Or contact: <tal.galili@gmail.com>
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ----------------------
```

```
##
## Attaching package: 'dendextend'
```

```
## The following object is masked from 'package:mosaic':
##
##     shuffle
```

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

```
## mixtools package, version 1.2.0, Released 2020-02-05
## This package is based upon work supported by the National Science Foundation under Grant
```

```
##
## Attaching package: 'mixtools'
```

```
## The following object is masked from 'package:car':
##
##     ellipse
```

Problem Set 4: Clustering

Fork the repository

Fork the repository for the problem set 4, `problem-set-4` (https://github.com/macss-ml20/problem-set-4). *Remember, all final submissions should be a single rendered PDF with code produced in-line.* Also, don't forget to **open the pull request** once you've committed your final submission to your forked repository. It needs to be merged back into the course master branch to be considered "submitted". See the syllabus for details.

This final problem set is **due Monday, 3/2, by 5 pm**.

The Data: State Legislative Professionalism

For the *second* "applied" part of the problem set, you will perform the three main clustering techniques we addressed in class (hierarchical, k-means, and Gaussian mixture models) on the Bowen and Greene state legislative professionalism data. In these data, there are four key features of interest, which record raw values for every state legislature in the U.S. from 1974-2010:

- Total session length
- Regular session length
- Legislators' salaries
- Expenditures per legislator

See the codebook for more details on these feature and the full set.

Performing k-Means By Hand

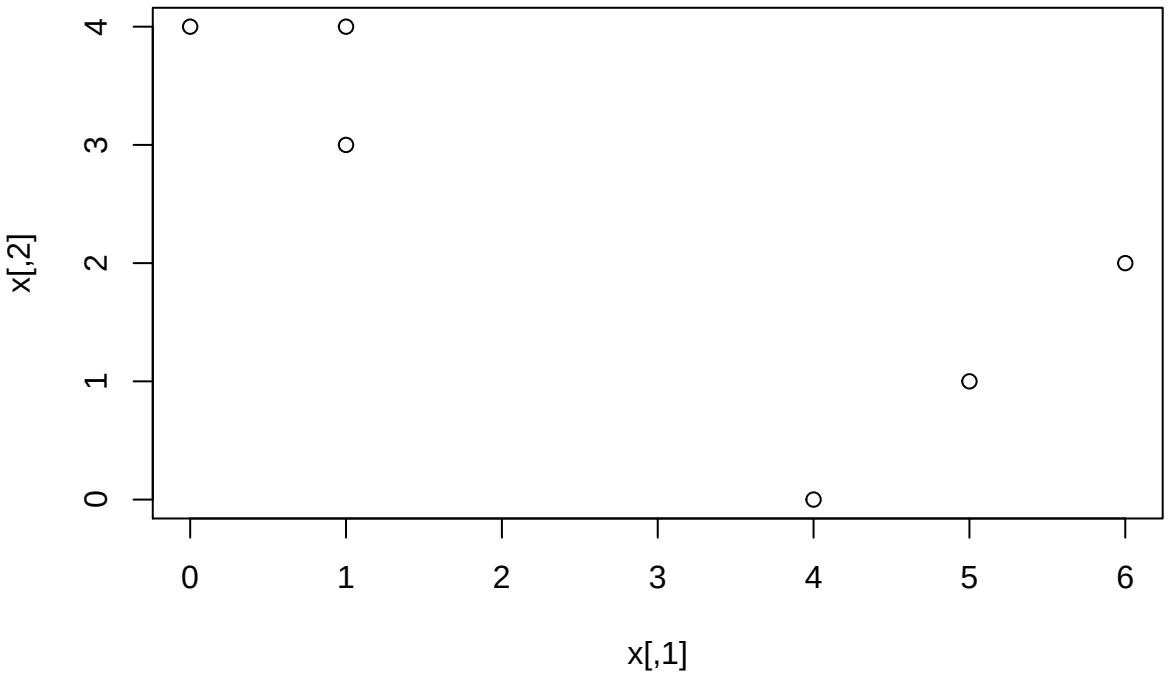
In this first part of the problem set, your goal is to gain a deeper conceptual understanding of the iterative process of k-means, which operates by initializing random cluster assignments, then updates cluster centroids, then cluster assignments, and so on, until convergence, which is defined by no furtehr changes to cluster configurations (i.e., optimal clusters definined by minimum sums of squares *within* clusters, and maximum sums of squares *between* clusters.

In short, then, by answering each of the following questions, you will be performing k-means clustering "by hand" to see and demonstrate this iterative process. Your simulated data includes `n = 6` observations and `p = 2` features, and you should set the number of clusters, `k`, equal to two (i.e., you are hunting for 2 clusters within these data). I will get you started with the observations in the set. Run the following line to create your simulated data:

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

1. (5 points) Plot the observations.

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

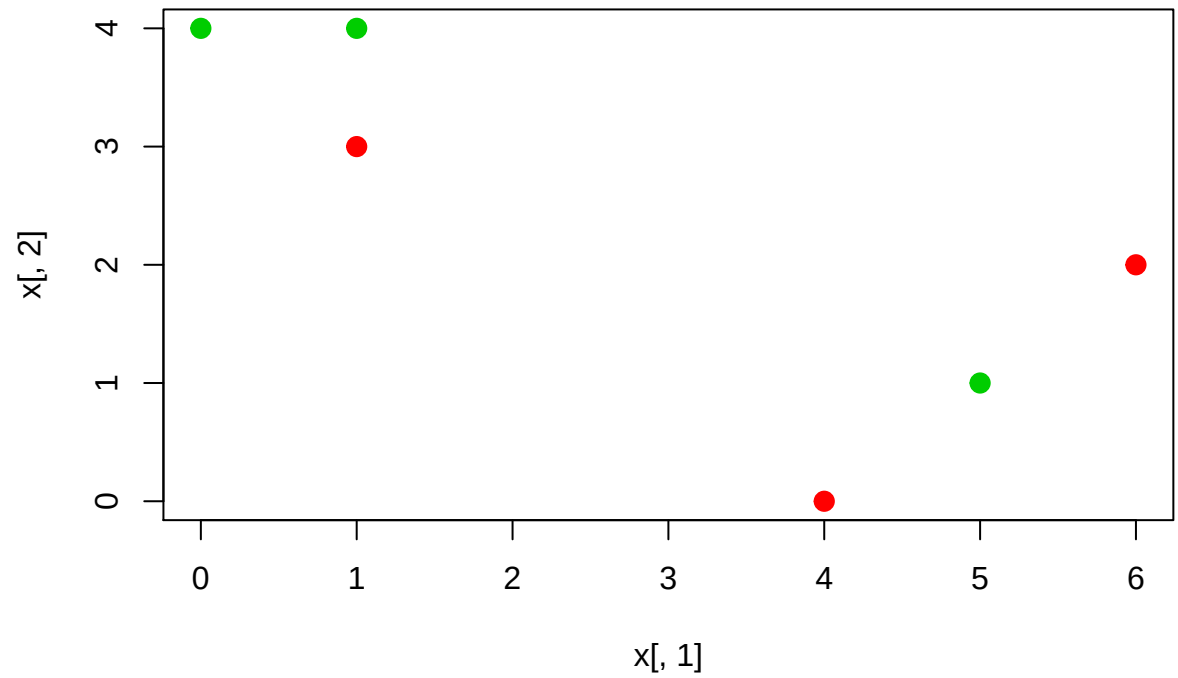


2. (5 points) Randomly assign a cluster label to each observation. Report the cluster labels for each observation *and* plot the results with a different color for each cluster (*remember to set your seed first*).

```
set.seed(100)
labels <- sample(2, nrow(x), replace = T)
labels
```

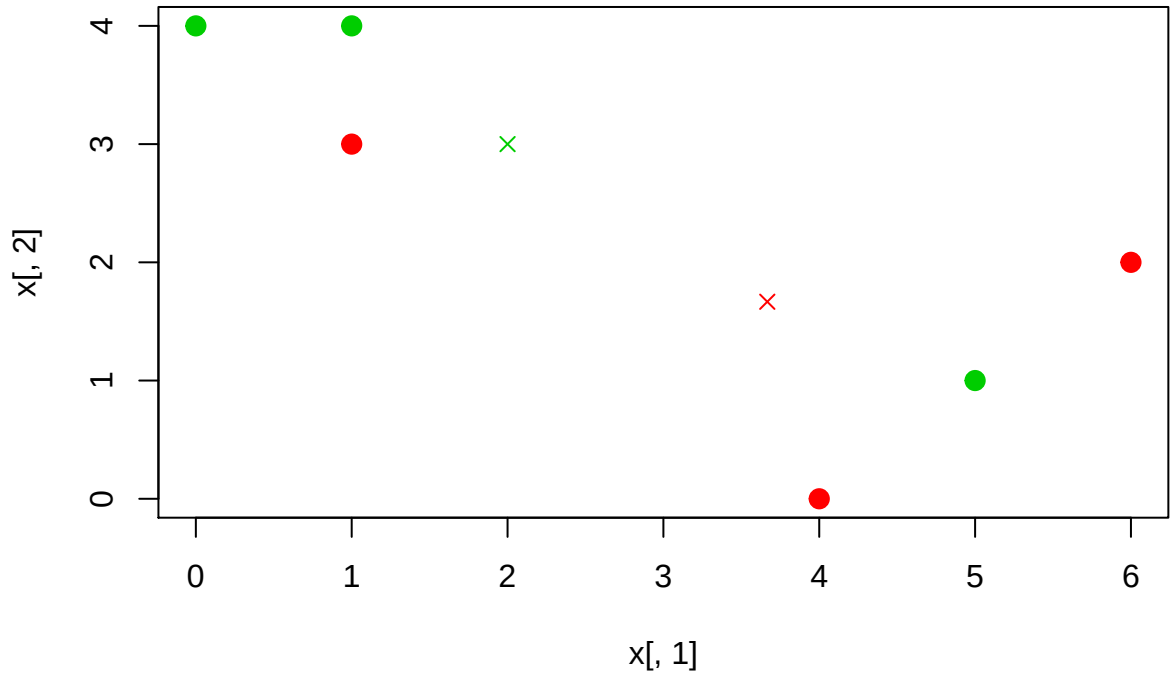
```
## [1] 2 1 2 2 1 1
```

```
plot(x[, 1], x[, 2], col = (labels + 1), pch = 20, cex = 2)
```



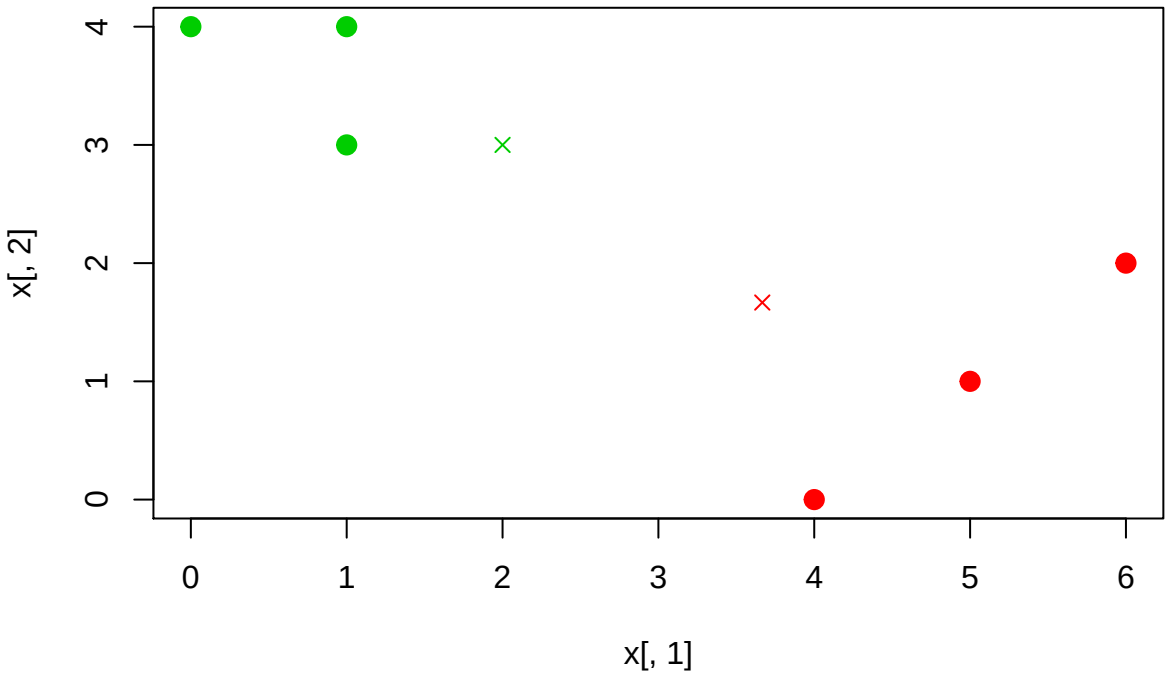
3. (10 points) Compute the centroid for each cluster.

```
centroid1 <- c(mean(x[labels == 1, 1]), mean(x[labels == 1, 2]))
centroid2 <- c(mean(x[labels == 2, 1]), mean(x[labels == 2, 2]))
plot(x[,1], x[,2], col=(labels + 1), pch = 20, cex = 2)
points(centroid1[1], centroid1[2], col = 2, pch = 4)
points(centroid2[1], centroid2[2], col = 3, pch = 4)
```



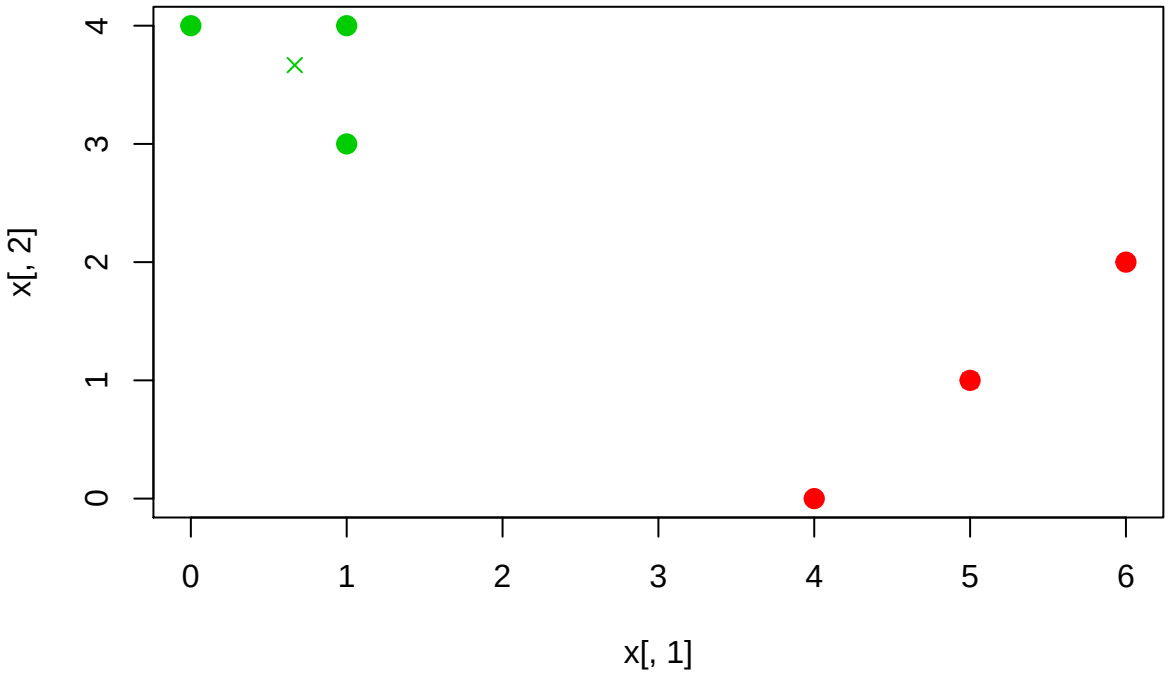
4. (10 points) Assign each observation to the centroid to which it is closest, in terms of Euclidean distance. Report the cluster labels for each observation.

```
labels <- c(2, 2, 2, 1, 1, 1)
plot(x[, 1], x[, 2], col = (labels + 1), pch = 20, cex = 2)
points(centroid1[1], centroid1[2], col = 2, pch = 4)
points(centroid2[1], centroid2[2], col = 3, pch = 4)
```



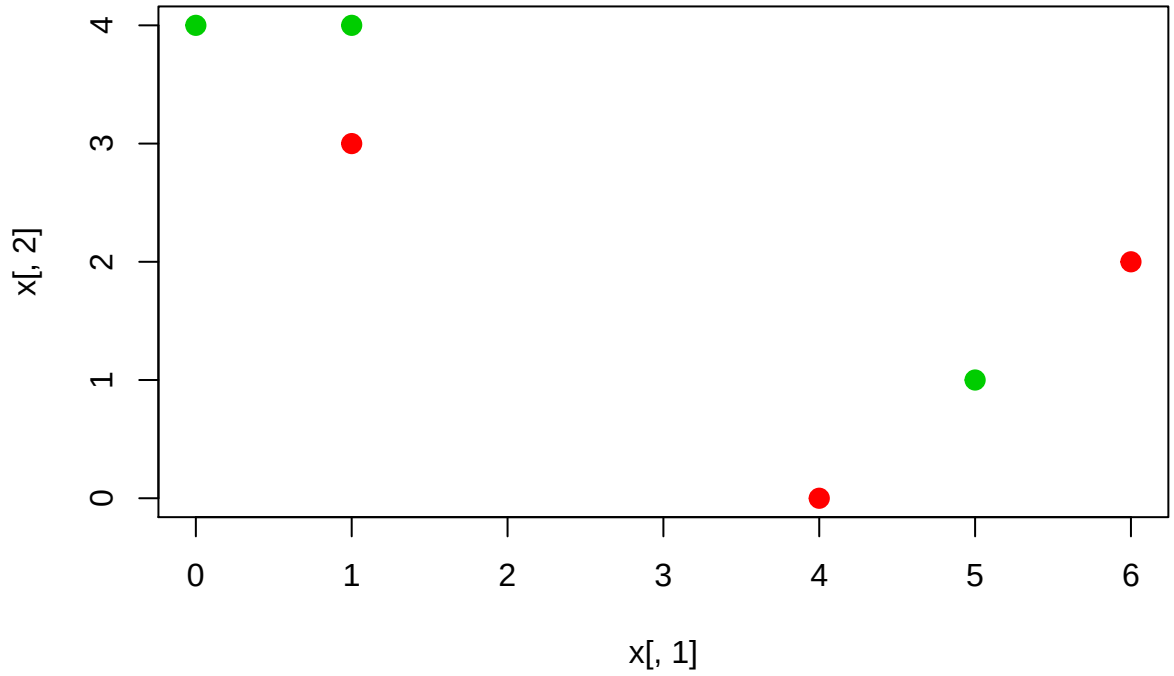
5. (5 points) Repeat (3) and (4) until the answers/clusters stop changing.

```
centroid1 <- c(mean(x[labels == 1, 1]), mean(x[labels == 1, 2]))
centroid2 <- c(mean(x[labels == 2, 1]), mean(x[labels == 2, 2]))
plot(x[,1], x[,2], col=(labels + 1), pch = 20, cex = 2)
points(centroid1[1], centroid1[2], col = 2, pch = 4)
points(centroid2[1], centroid2[2], col = 3, pch = 4)
```

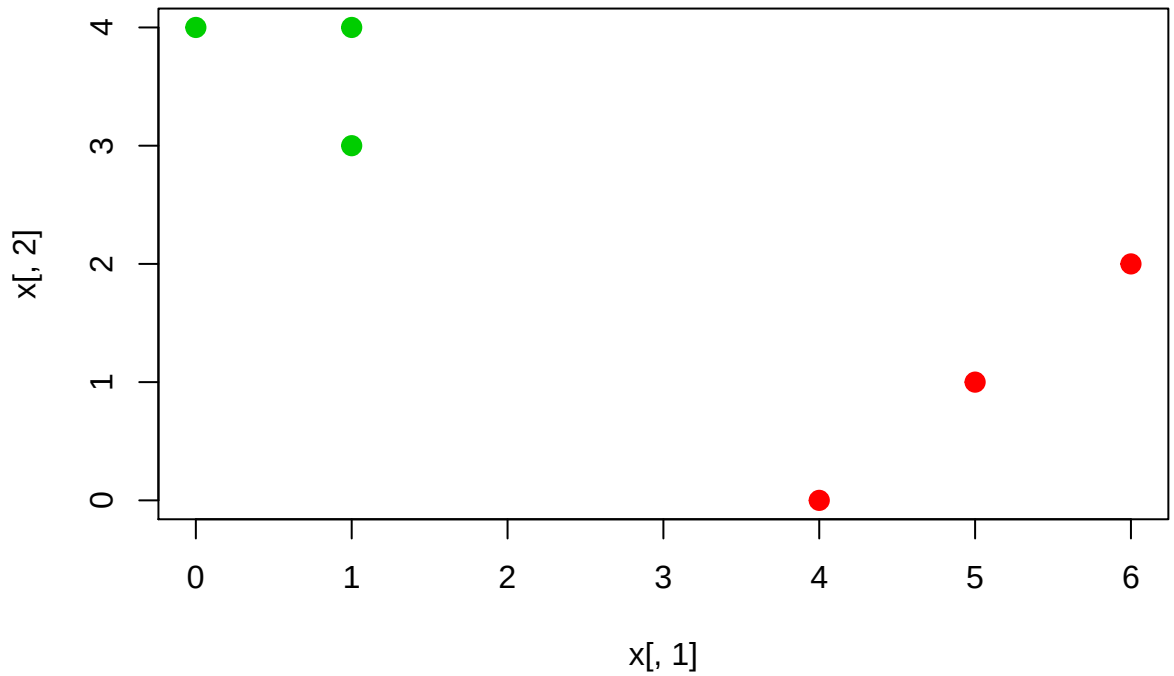


6. (10 points) Reproduce the original plot from (1), but this time color the observations *according to the clusters labels you obtained* by iterating the cluster centroid calculation and assignments.

```r
set.seed(100)
labels <- sample(2, nrow(x), replace = T)
plot(x[, 1], x[, 2], col=(labels + 1), pch = 20, cex = 2)
centroid1 <- c(mean(x[labels == 1, 1]), mean(x[labels == 1, 2]))
centroid2 <- c(mean(x[labels == 2, 1]), mean(x[labels == 2, 2]))
plot(x[, 1], x[, 2], col=(labels + 1), pch = 20, cex = 2)
```




```r
labels <- c(2, 2, 2, 1, 1, 1)
plot(x[, 1], x[, 2], col = (labels + 1), pch = 20, cex = 2)
```



```
centroid1 <- c(mean(x[labels == 1, 1]), mean(x[labels == 1, 2]))
centroid2 <- c(mean(x[labels == 2, 1]), mean(x[labels == 2, 2]))
plot(x[,1], x[,2], col=(labels + 1), pch = 20, cex = 2)
```

Clustering State Legislative Professionalism

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

```
load("/Users/sabrinasong/Desktop/problem-set-4-master/Data and Codebook/legprof-components.
prof <- x
```

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)

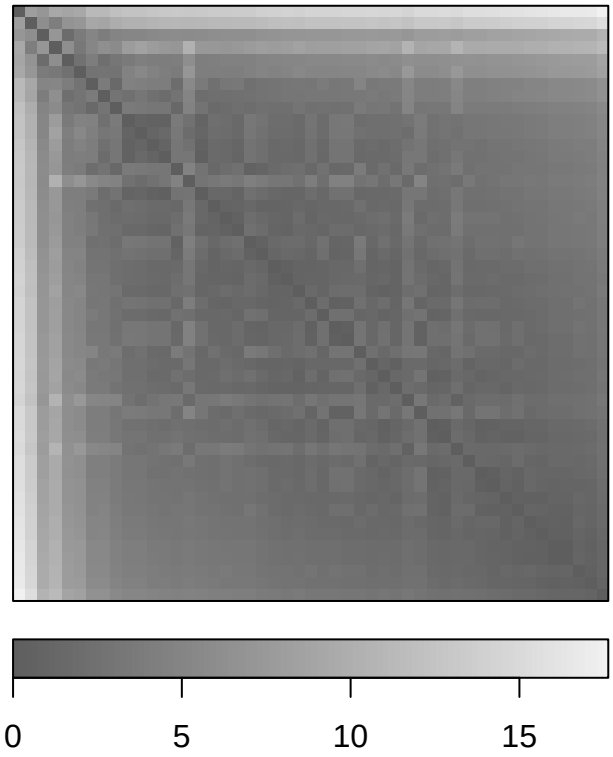
```
#Select: stateabv, t_slength, slength, salary_real + 2009/10 session
prof <- subset(prof, sessid == '2009/10', select = c("stateabv", "t_slength", "slength", "s

#Remove missing data
prof = prof[complete.cases(prof), ]

#Store names
prof_names <- subset(prof, select = "stateabv")

#Prior to scaling the data, let's store the summary stats so that we will be able to see
summary <- summary(prof)

#Drop names + scale the numeric columns
prof <- scale(subset(prof, select = c("t_slength", "slength", "salary_real", "expend")))
```

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.

```
library(seriation)
```

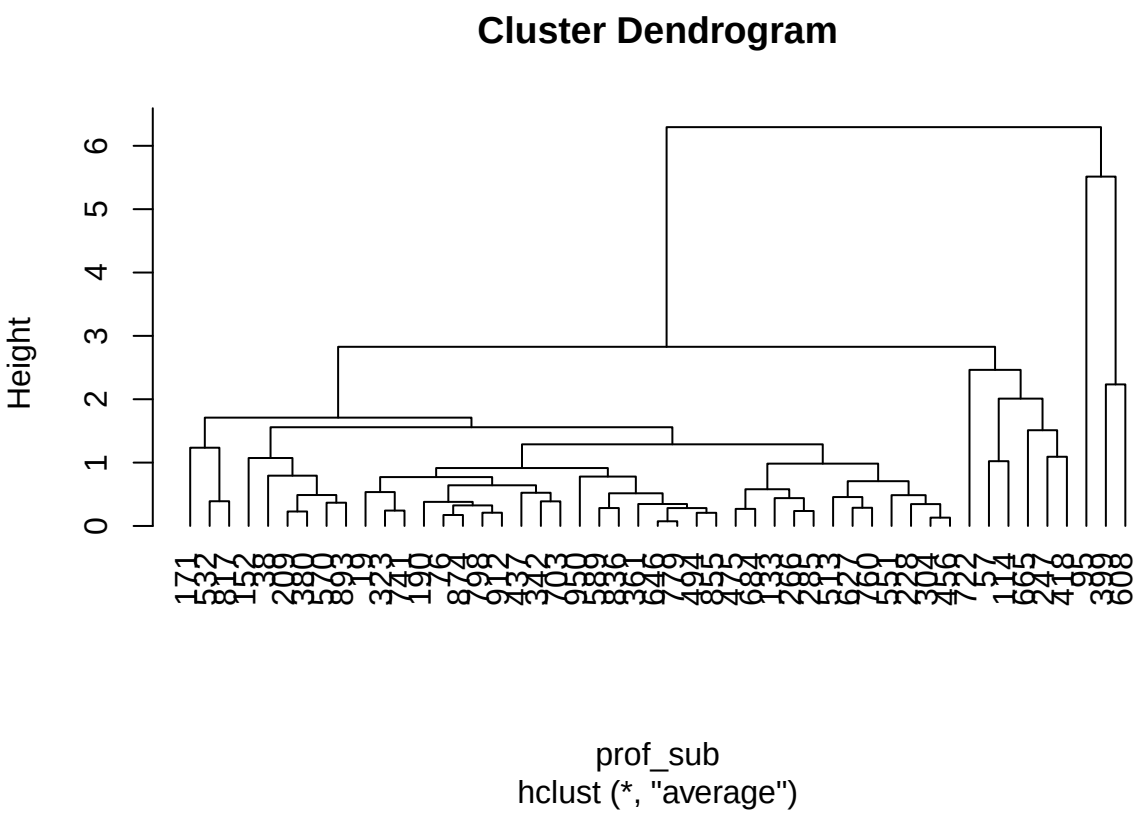
```
## Registered S3 method overwritten by 'seriation':
##   method         from
##   reorder.hclust gclus

##
## Attaching package: 'seriation'

## The following object is masked from 'package:modelr':
##
##     permute

## The following object is masked from 'package:lattice':
##
##     panel.lines
```

```
#Calculate distance using our scaled df
prof_dist <- dist(prof, method = "manhattan")
dissplot(prof_dist)
```



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.

```r
#Average Linkage
prof_sub <- prof %>% dist()
hc_average <- hclust(prof_sub, method = "average")
plot(hc_average, hang = -1)
```



#The numbers are state names. We notice that at the right end, three states clustered by themselves (95, 399, 608). These are CA, MA and NY. These are all mega states so they may be more similar to each other. Southern states tend to stay together.

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.

```r
kmeans <- kmeans(prof, centers = 2, nstart = 15)
str(kmeans)
```

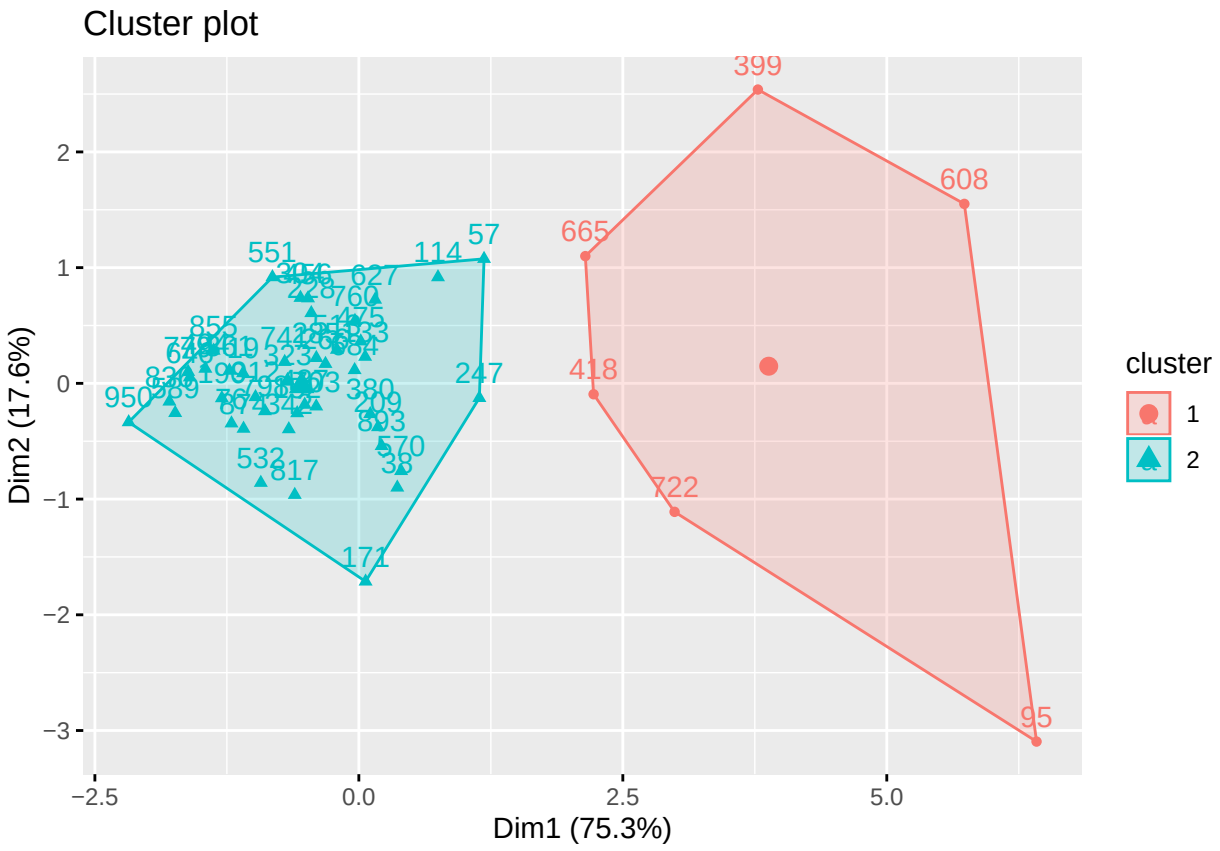
```
## List of 9
##  $ cluster     : Named int [1:49] 2 2 2 2 1 2 2 2 2 2 ...
##   ..- attr(*, "names")= chr [1:49] "19" "38" "57" "76" ...
##  $ centers     : num [1:2, 1:4] 2.1 -0.293 2.101 -0.293 2.031 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "1" "2"
##   .. ..$ : chr [1:4] "t_slength" "slength" "salary_real" "expend"
##  $ totss       : num 192
```

```
##  $ withinss    : num [1:2] 40.4 48.4
##  $ tot.withinss: num 88.7
##  $ betweenss   : num 103
##  $ size        : int [1:2] 6 43
##  $ iter        : int 1
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"
```

```
#let's save cluster assignments (add to the state names)
library(factoextra)
```

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

```
#Visualize the outcome of this clustering
fviz_cluster(kmeans, prof)
```




```
#let's save cluster assignments (add to the state names)
prof_names$k_cluster = as.factor(kmeans$cluster)
```

#From the above, we can see that 6 states were put into one cluster, and the remaining 43 were placed intothe second cluster. Specifically, we can check that the states in the blue cluster were CA, MA, MI, NY, OH and PA.

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.

```
set.seed(1234)
gmm1 <- mvnormalmixEM(prof, k = 2)
```

```
## number of iterations= 20
```

```
posterior <- data.frame(cbind(gmm1$x, gmm1$posterior))
posterior$component <- ifelse(posterior$comp.1 > 0.5, 1, 2)
head(posterior)
```

```
##     t_slength slength salary_real  expend     comp.1   comp.2
## 19    -0.3717 -0.4595     -1.0920 -0.2400  9.960e-01 0.003956
## 38    -0.2294 -0.1452      0.4011  0.8591  3.779e-09 1.000000
## 57     1.6453  0.7952     -0.1336 -0.1299  8.310e-68 1.000000
## 76    -0.8036 -0.7882     -0.4924 -0.2612  9.920e-01 0.007993
## 95     2.8807  1.7767      3.2070  5.4785 3.629e-127 1.000000
## 114    0.6827  0.9009      0.1114 -0.3486  9.621e-01 0.037866
##     component
## 19          1
## 38          2
## 57          2
## 76          1
## 95          2
## 114         1
```

```
table(posterior$component)
```

```
##
##  1  2
## 37 12
```

```
#add to the df where we are collating the cluster assignments for later
prof_names$gmm_cluster <- as.factor(posterior$component)
```

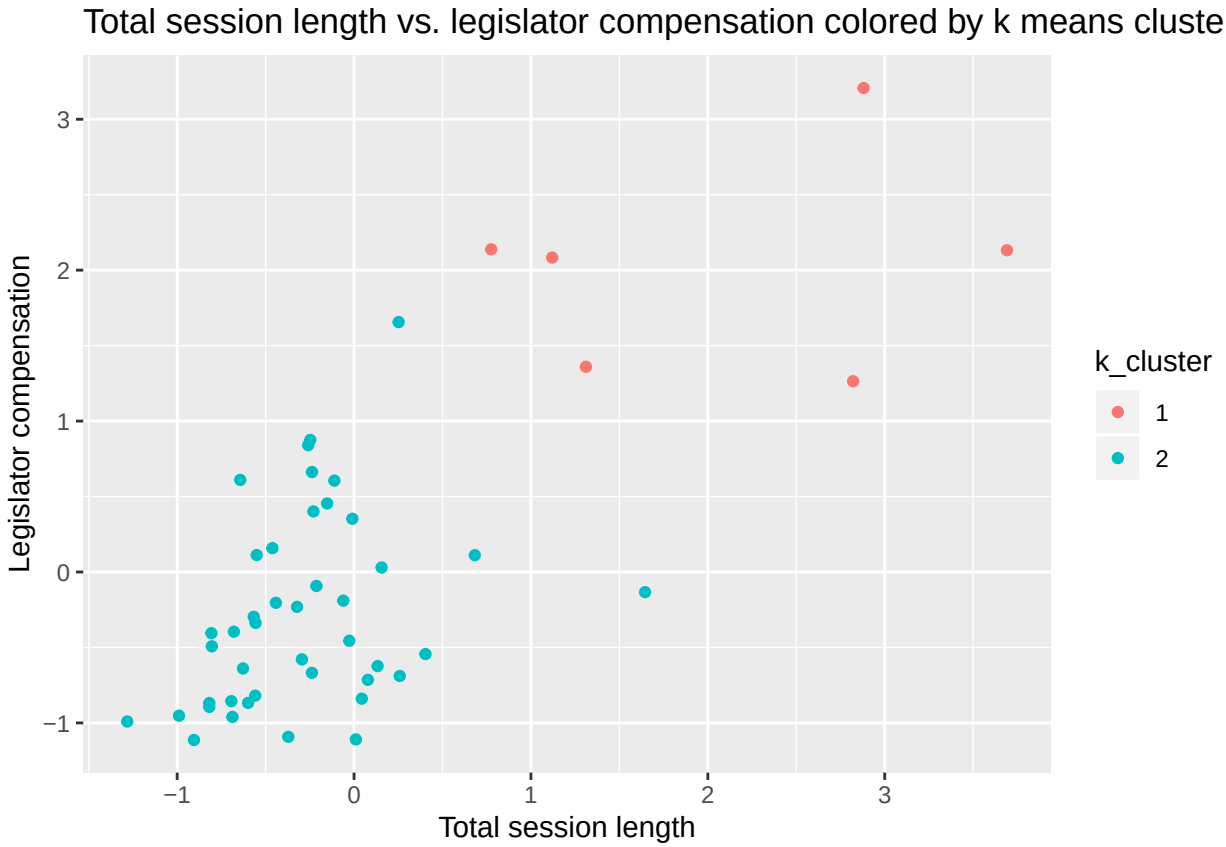
#We have one cluster that has 37 states and one which has 12 here, which is different than what we did kmeans. The probability scores of belonging to each component (comp.1 & comp.2) here are very high or low. While Gaussian models are a form of soft partitioning and thus it might be possible to belong to more than one cluster, here there seem to be very high or low probabilities for belonging to a cluster.

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.

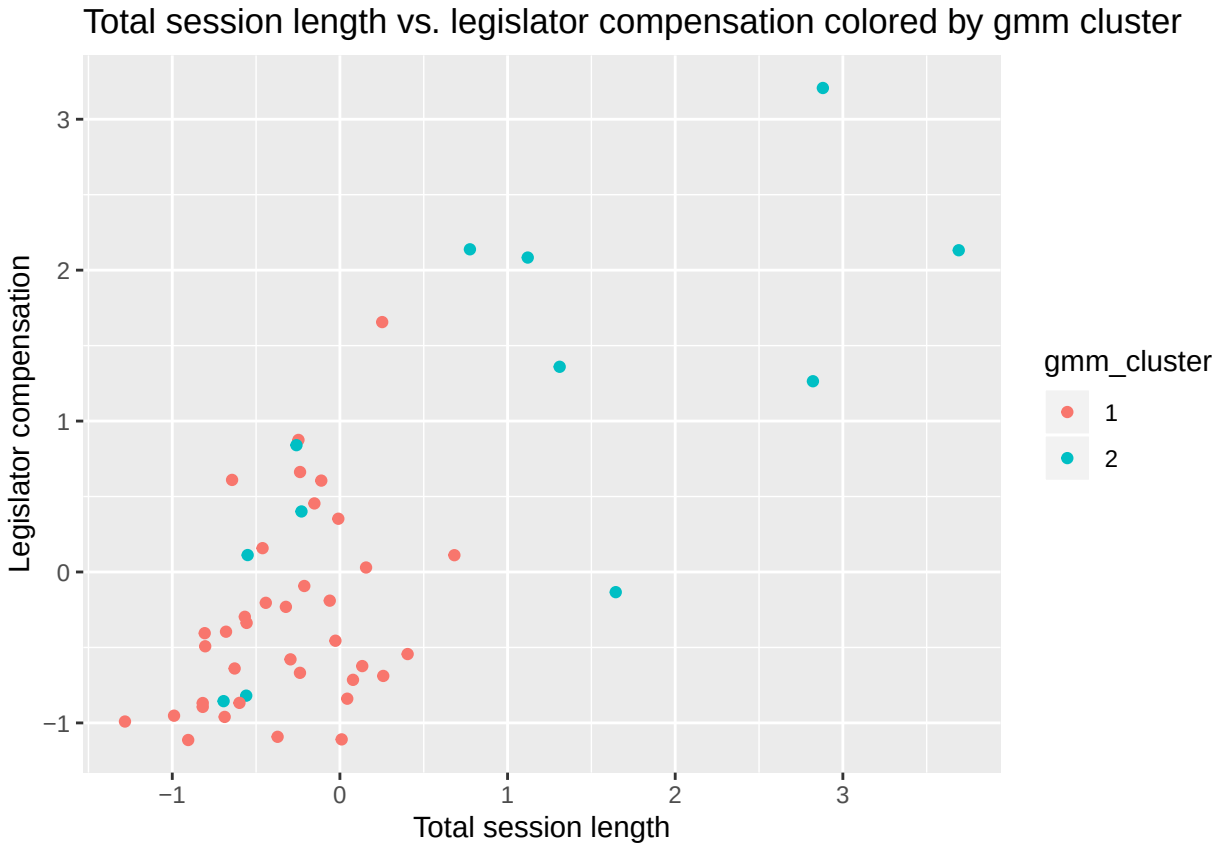
```
new_df <- cbind(prof_names, prof)
head(new_df)
```

```
##     stateabv k_cluster gmm_cluster t_slength slength salary_real
## 19        AL         2           1   -0.3717 -0.4595     -1.0920
## 38        AK         2           2   -0.2294 -0.1452      0.4011
## 57        AZ         2           2    1.6453  0.7952     -0.1336
## 76        AR         2           1   -0.8036 -0.7882     -0.4924
## 95        CA         1           2    2.8807  1.7767      3.2070
## 114       CO         2           1    0.6827  0.9009      0.1114
##      expend
## 19  -0.2400
## 38   0.8591
## 57  -0.1299
## 76  -0.2612
## 95   5.4785
## 114 -0.3486
```

```
new_df %>%
ggplot(aes(x = t_slength, y = salary_real, color = k_cluster)) +
geom_point() +
labs(x = "Total session length",
y = "Legislator compensation",
title = "Total session length vs. legislator compensation colored by k means cluster")
```




```
new_df %>%
ggplot(aes(x = t_slength, y = salary_real, color = gmm_cluster)) +
geom_point() +
labs(x = "Total session length",
y = "Legislator compensation",
title = "Total session length vs. legislator compensation colored by gmm cluster")
```



#Both graphs suggest that there are groupings emerging with longer session length and higher compensation vs. lower compensation and shorter total session length. The Gaussian model has a few additional points within the lower compensation / session length that appear to be grouped with the higher compensation / higher total session length. In sum, if I think that total session length and legislature compensation are valid measures of "professionalism". Kmeans cluster seem to do a better job of finding similar state legislatures.

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.

```
library(clValid)
```

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

```
library(cluster)
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
```

```
#Validate for hierarchical, kmeans and GMM
internal <- clValid(prof, 2:5, clMethods = c("hierarchical", "kmeans","model"), validation
summary(internal)
```

```
##
## Clustering Methods:
##  hierarchical kmeans model
##
## Cluster sizes:
##  2 3 4 5
##
## Validation Measures:
##                                  2      3      4      5
##
## hierarchical Connectivity   6.087  6.954 16.188 18.677
##              Dunn           0.364  0.437  0.256  0.284
##              Silhouette     0.699  0.671  0.493  0.444
## kmeans       Connectivity   8.446 10.896 16.188 28.744
##              Dunn           0.173  0.258  0.256  0.109
##              Silhouette     0.646  0.613  0.493  0.304
## model        Connectivity  10.739 28.612 39.069 67.840
##              Dunn           0.152  0.063  0.022  0.026
##              Silhouette     0.631  0.259  0.186  0.008
##
## Optimal Scores:
##
##              Score Method       Clusters
## Connectivity 6.087 hierarchical 2
## Dunn         0.437 hierarchical 3
## Silhouette   0.699 hierarchical 2
```

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

- What can you take away from the fit?

#We use internal validation to understand how well clustering algorithms perform relative to other algorithms. We compare 2-5 numbers of clusters and types of clustering to see which model is the best. we want high values for silhouette width and the Dunn index and low values on connectivity. Here, hierarchical is the strongest method of all three validation measures.

- Which approach is optimal? And optimal at what value of k?

#In this specific case, hierarchical was the strongest performer across all three measures. For the connectivity and silhouette width, 2 clusters would be the optimal number, for Dunn index 3 clusters would be best.

- What are reasons you could imagine selecting a technically "sub-optimal" clustering method, regardless of the validation statistics?

#It depends on the real world senario, namely, what we want to do with our clustering. For example, if we only wants two clusters to focus on, then we may choose a sub-optimal clustering method regardless of the Dunn index.