Problem Set 4

Soowon Jo

2/29/2020

Data Preparation

```r
library(tidyverse)
```

```
##   Attaching packages                                         tidyverse 1.3.0

##   ggplot2 3.2.1      purrr   0.3.3
##   tibble  2.1.3      dplyr   0.8.3
##   tidyr   1.0.0      stringr 1.4.0
##   readr   1.3.1      forcats 0.4.0

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

```r
library(seriation)
```

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

```r
library(skimr)
library(dendextend) # for "cutree" function. to cut hierarchical tree
```

```
##
## ---------------------
## 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
```

```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-051
```

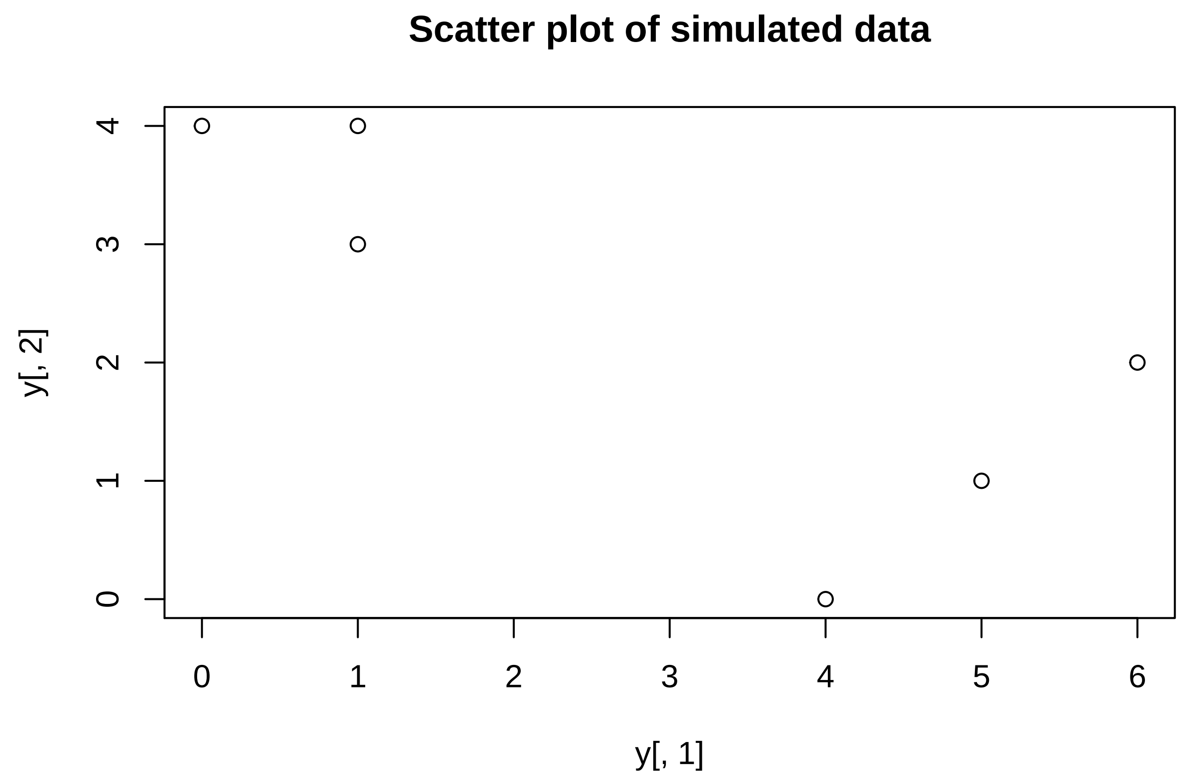
```r
library(plotGMM)
library(clValid)
```

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

```r
setwd("/Users/soowonjo/Desktop/MachineLearning/PB4/problem-set-4/Data")
load("legprof-components.v1.0.RData")
legprof <- x
```

1. (5 points) Plot the observations.

```r
y <- cbind(c(1, 1, 0, 5, 6, 4), c(4, 3, 4, 1, 2, 0))
plot(y[,1],y[,2],
    main = "Scatter plot of simulated data")
```

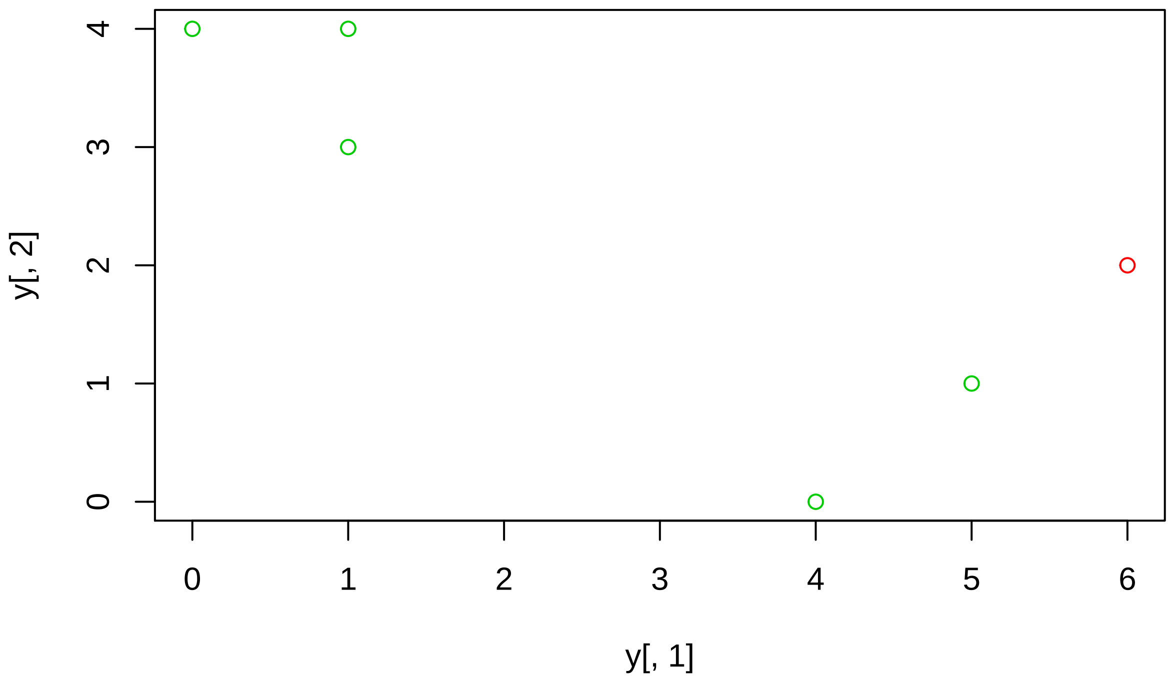


2. (5 points) Randomly assign a cluster label to each observation. Report the cluster labels for each observation and plot the results with a different color for each cluster (remember to set your seed first).

```r
set.seed(1234)
cluster.label <- sample(2, nrow(y), replace = T)
cluster.label
```

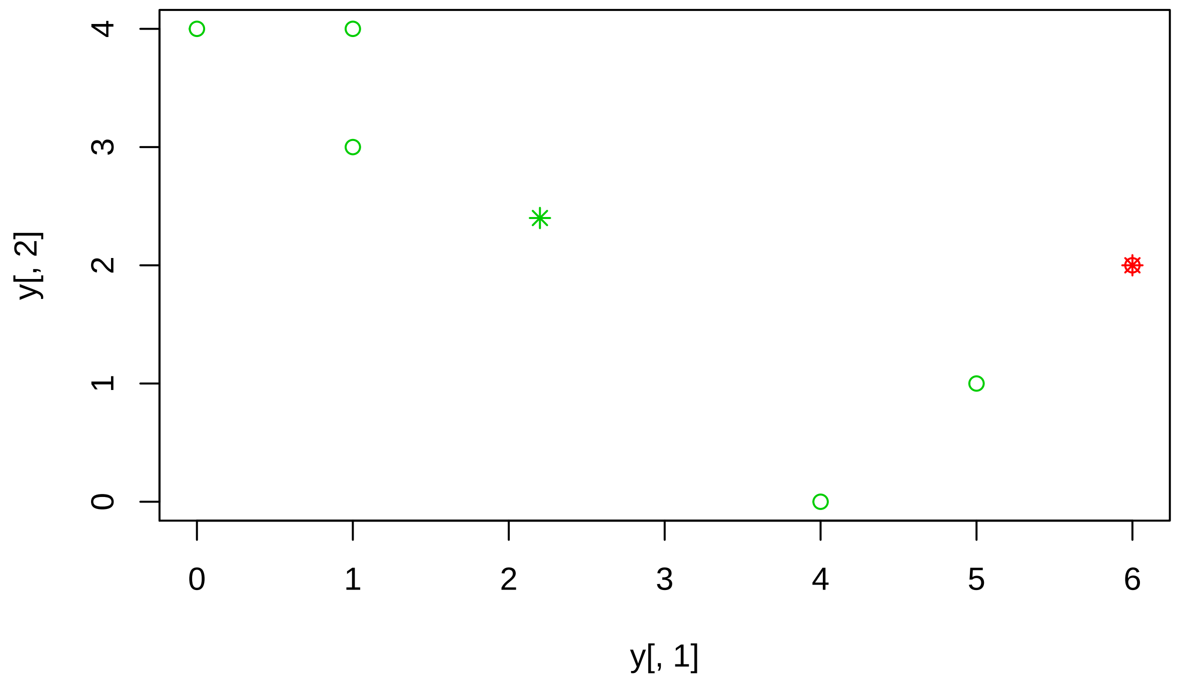
```
## [1] 2 2 2 2 1 2
```

```r
plot(y[,1], y[,2],
     col=(cluster.label + 1))
```



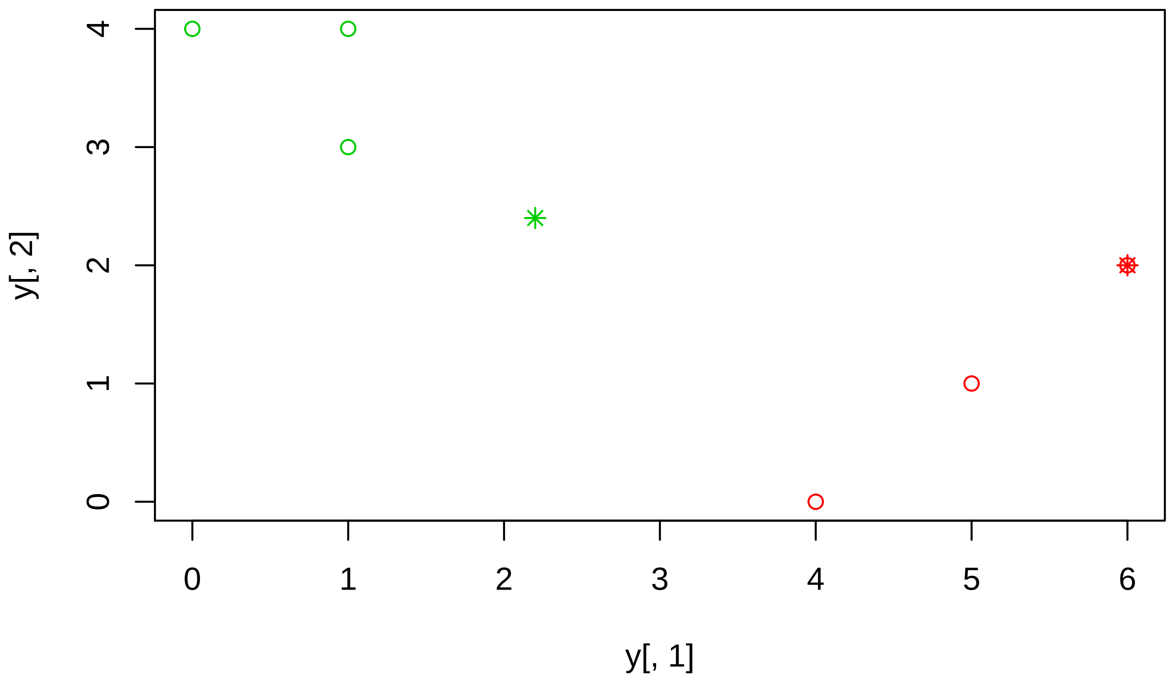
3. (10 points) Compute the centroid for each cluster.

```r
centroid1 <- c(mean(y[cluster.label == 1, 1]),
                mean(y[cluster.label == 1, 2]))
centroid2 <- c(mean(y[cluster.label == 2, 1]),
                mean(y[cluster.label == 2, 2]))

plot(y[,1], y[,2],
     col=(cluster.label + 1))

# centroid points
points(centroid1[1], centroid1[2],
       col = 2, pch = 8)
points(centroid2[1], centroid2[2],
       col = 3, pch = 8)
```



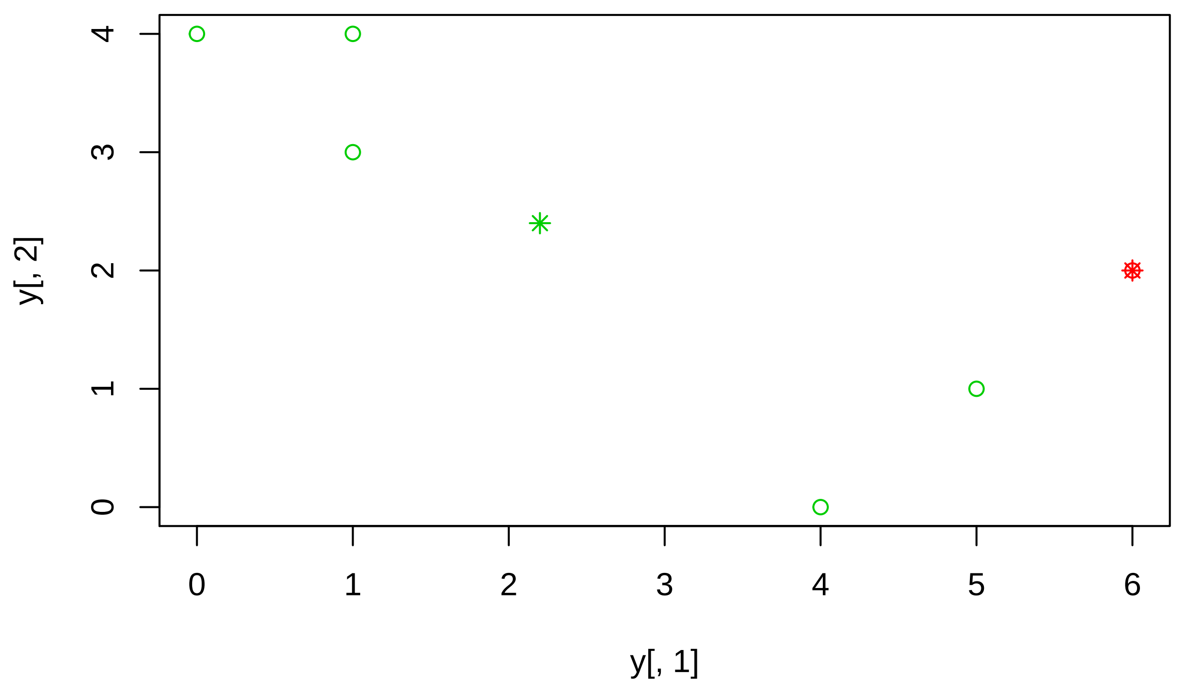
4. (10 points) Assign each observation to the centroid to which it is closest, in terms of Euclidean distance. Report the cluster labels for each observation.

```r
# Assign labels to observations
labels <- c(2, 2, 2, 1, 1, 1)

# plot relabeled observations
plot(y[, 1], y[, 2],
     col = (labels + 1))
points(centroid1[1], centroid1[2],
       col = 2, pch = 8)
points(centroid2[1], centroid2[2],
       col = 3, pch = 8)
```

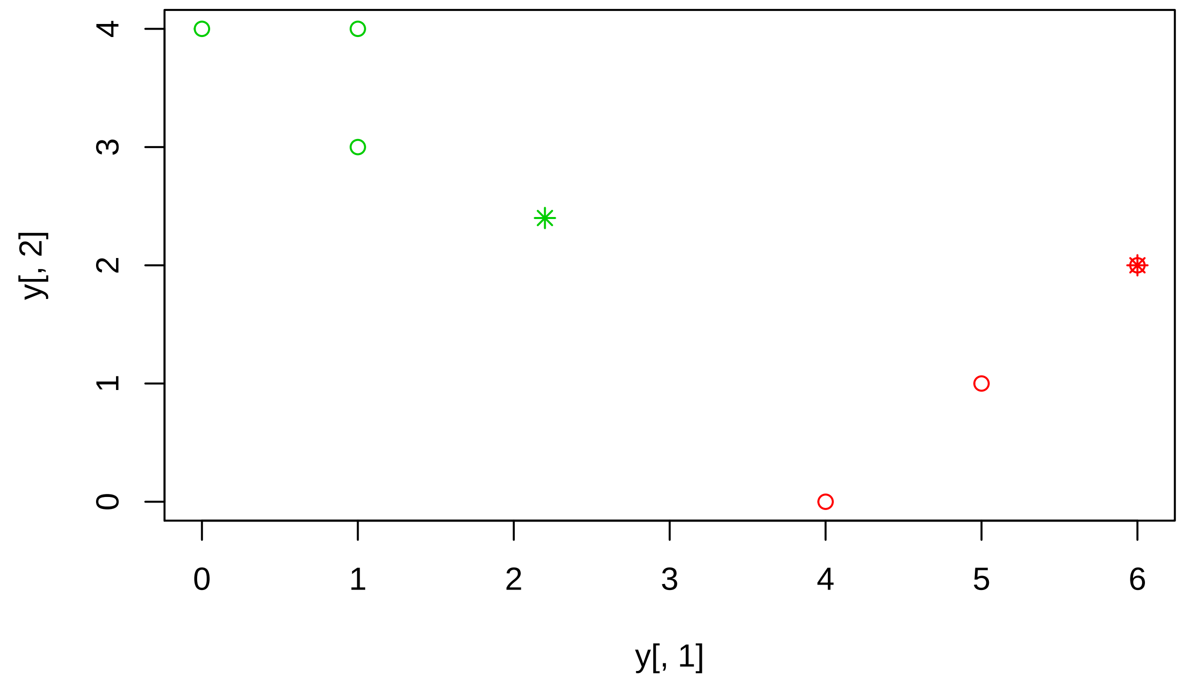


5. (5 points) Repeat (3) and (4) until the answers/clusters stop changing.

```r
centroid1 <- c(mean(y[cluster.label == 1, 1]),
                mean(y[cluster.label == 1, 2]))
centroid2 <- c(mean(y[cluster.label == 2, 1]),
                mean(y[cluster.label == 2, 2]))

plot(y[,1], y[,2],
     col=(cluster.label + 1))

#  centroid points
points(centroid1[1], centroid1[2],
       col = 2, pch = 8)
points(centroid2[1], centroid2[2],
       col = 3, pch = 8)
```

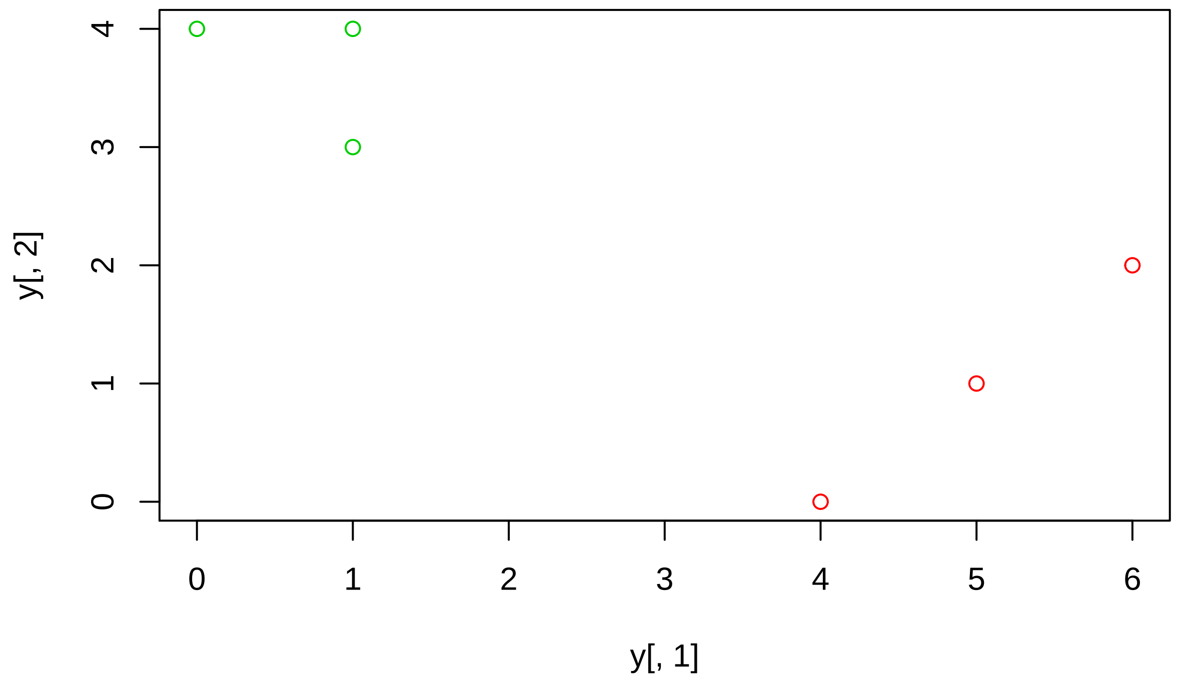


```
# Assign lables to observations
labels <- c(2, 2, 2, 1, 1, 1)

# plot relabeled observations
plot(y[, 1], y[, 2],
     col = (labels + 1))
points(centroid1[1], centroid1[2],
       col = 2, pch = 8)
points(centroid2[1], centroid2[2],
       col = 3, pch = 8)
```



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.

```
plot(y[, 1], y[, 2],
     col = (labels + 1))
```



Clustering State Legislative Professionalism

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

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)

```r
skim(legprof)
```

Table 1: Data summary

Name	legprof
Number of rows	950
Number of columns	11

Column type frequency:

Table 1: Data summary

AsIs	2
factor	1
numeric	8
Group variables	None

Variable type: AsIs

skim_variable	n_missing	complete_rate	n_unique	min_length	max_length
stateabv	0	1	50	1	1
state	0	1	50	1	1

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
sessid	0	1	TRUE	19	197: 50, 197: 50, 197: 50, 197: 50

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
fips	0	1.00	29.32	15.63	1.00	17.00	29.50	42.00	56.00	
t_slength	61	0.94	147.60	86.09	36.00	91.00	128.51	171.00	549.54	
slength	61	0.94	136.39	81.18	36.00	85.20	120.00	158.00	521.85	
salary_real	5	0.99	55.82	47.11	0.00	20.11	41.96	80.08	254.94	
expend	5	0.99	599.51	724.19	40.14	219.93	395.10	650.29	5523.10	
year	0	1.00	1992.09	10.96	1974.00	1982.00	1992.00	2002.00	2011.00	
mds1	61	0.94	0.00	1.48	-1.85	-0.93	-0.31	0.41	8.56	
mds2	61	0.94	0.00	0.72	-3.14	-0.34	0.09	0.30	3.35	

```r
legprof_subset <- legprof %>%
  filter(sessid == "2009/10") %>%
  select(state, t_slength, slength, salary_real, expend)

legprof_states <- scale(legprof_subset[, -c(1)]) %>%
  as.tibble() %>%
  na.omit()
```

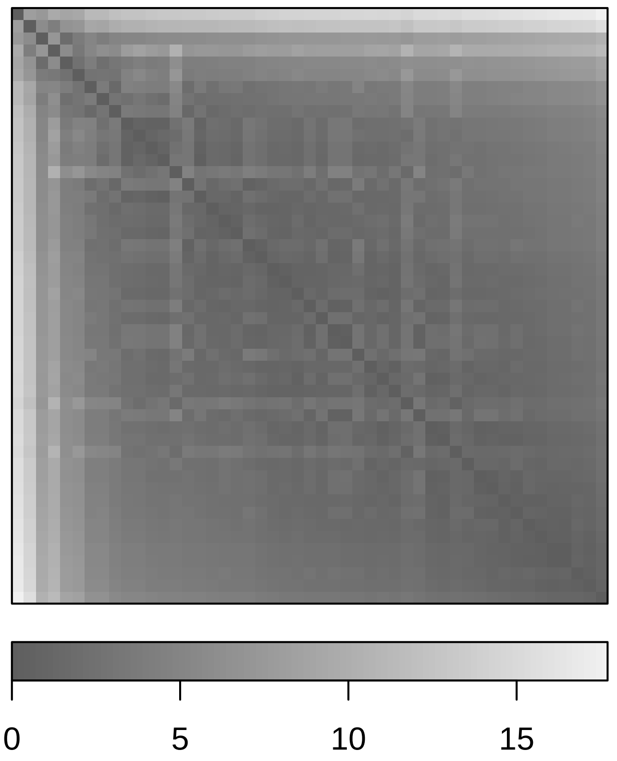
```
## Warning: `as.tibble()` is deprecated, use `as_tibble()` (but mind the new semantics).
## This warning is displayed once per session.
```

```r
# store state names
states <- legprof_subset %>%
    na.omit() %>%
    select(state)
states
```

```
##             state
## 1         Alabama
## 2          Alaska
## 3         Arizona
## 4        Arkansas
## 5      California
## 6        Colorado
## 7     Connecticut
## 8        Delaware
## 9         Florida
## 10        Georgia
## 11         Hawaii
## 12          Idaho
## 13       Illinois
## 14        Indiana
## 15           Iowa
## 16         Kansas
## 17       Kentucky
## 18      Louisiana
## 19          Maine
## 20       Maryland
## 21  Massachusetts
## 22       Michigan
## 23      Minnesota
## 24    Mississippi
## 25       Missouri
## 26        Montana
## 27       Nebraska
## 28         Nevada
## 29  New Hampshire
## 30     New Jersey
## 31     New Mexico
## 32       New York
## 33 North Carolina
## 34   North Dakota
## 35           Ohio
## 36       Oklahoma
## 37         Oregon
## 38   Pennsylvania
## 39   Rhode Island
## 40 South Carolina
## 41   South Dakota
## 42      Tennessee
## 43          Texas
## 44           Utah
## 45        Vermont
## 46       Virginia
## 47     Washington
## 48  West Virginia
## 50        Wyoming
```

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.

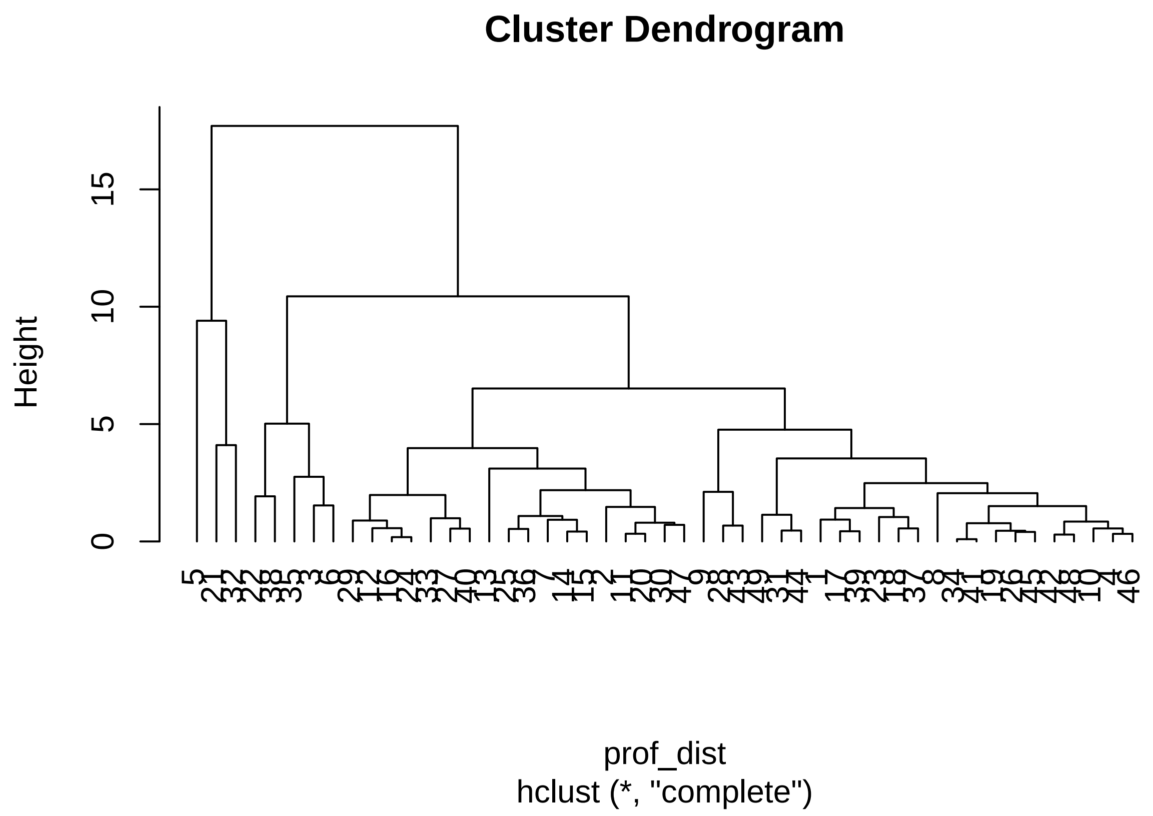
```r
prof_dist <- dist(legprof_states, method = "manhattan")
dissplot(prof_dist)
```



There may be few squares shown in the ODI, but the unclear delineation indicates very low likelihood 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.

```r
hc_complete <- hclust(prof_dist,
                      method = "complete"); plot(hc_complete, hang = -1)
```



The y-axis is a measure of closeness of either individual data points or clusters. The x-axis represents the objects and clusters. Our main interest is in similarity and clustering. Each joining of two clusters is represented on the graph by the splitting of a vertical line into two verticle lines. The vertical position of the split, shown by the short horizontal bar, give the distance (dissimiliariy) between the two clusters.

In this dendrogram, we see the three clusters as three branches that occur at about the same vertical distance. However, this interpretation can be justified only when the ultrametric tree in equality holds.

As an example, the dendogram suggests that Massachusetts(21) and New York(32) are much closer to each other than is New York(32) to Michigan(22).

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
set.seed(634)

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

```
## List of 9
##  $ cluster     : int [1:49] 1 1 1 1 2 1 1 1 1 1 ...
##  $ centers     : num [1:2, 1:4] -0.293 2.1 -0.293 2.101 -0.302 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "1" "2"
##   .. ..$ : chr [1:4] "t_slength" "slength" "salary_real" "expend"
##  $ totss       : num 193
##  $ withinss    : num [1:2] 48.6 40.9
##  $ tot.withinss: num 89.5
##  $ betweenss   : num 104
##  $ size        : int [1:2] 43 6
##  $ iter        : int 1
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"
```

```
variable_1 <- as.data.frame(kmeans$cluster)
variable_2 <- states

cbind(variable_1, variable_2)
```

```
##    kmeans$cluster          state
## 1               1        Alabama
## 2               1         Alaska
## 3               1        Arizona
## 4               1       Arkansas
## 5               2     California
## 6               1       Colorado
## 7               1    Connecticut
## 8               1       Delaware
## 9               1        Florida
## 10              1        Georgia
## 11              1         Hawaii
## 12              1          Idaho
## 13              1       Illinois
## 14              1        Indiana
## 15              1           Iowa
## 16              1         Kansas
## 17              1       Kentucky
## 18              1      Louisiana
## 19              1          Maine
## 20              1       Maryland
## 21              2  Massachusetts
## 22              2       Michigan
## 23              1      Minnesota
## 24              1    Mississippi
## 25              1       Missouri
## 26              1        Montana
## 27              1       Nebraska
## 28              1         Nevada
## 29              1  New Hampshire
## 30              1     New Jersey
## 31              1     New Mexico
## 32              2       New York
## 33              1 North Carolina
## 34              1   North Dakota
## 35              2           Ohio
## 36              1       Oklahoma
## 37              1         Oregon
## 38              2   Pennsylvania
## 39              1   Rhode Island
## 40              1 South Carolina
## 41              1   South Dakota
## 42              1      Tennessee
## 43              1          Texas
## 44              1           Utah
## 45              1        Vermont
## 46              1       Virginia
## 47              1     Washington
## 48              1  West Virginia
## 50              1        Wyoming
```

The result from kmeans shows that total 6 states (California, Massachusetts, Michigan, New York, Ohio, and Pennsylvania) form one cluster and the remaining 43 states were placed into another 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.

```r
set.seed(2000)
gmm <- mvnormalmixEM(legprof_states, k = 2)
```

```
## number of iterations= 16
```

```r
gmm$mu
```

```
## [[1]]
## [1] -0.2763465 -0.2450724 -0.2132464 -0.1960989
##
## [[2]]
## [1] 2.431846 2.156634 1.694880 1.705799
```

```r
gmm$sigma
```

```
## [[1]]
##            [,1]       [,2]      [,3]       [,4]
## [1,] 0.24528126 0.27954563 0.2100159 0.03137054
## [2,] 0.27954563 0.32491503 0.2379421 0.02504616
## [3,] 0.21001590 0.23794209 0.5825970 0.13823755
## [4,] 0.03137054 0.02504616 0.1382376 0.23965942
##
## [[2]]
##           [,1]       [,2]      [,3]       [,4]
## [1,] 0.8556104  1.0197243 0.3986101  0.3545036
## [2,] 1.0197243  1.5611386 0.3432493 -0.3997010
## [3,] 0.3986101  0.3432493 1.2353043  2.0069944
## [4,] 0.3545036 -0.3997010 2.0069944  4.4408608
```

```r
gmm$lambda
```

```
## [1] 0.897959 0.102041
```

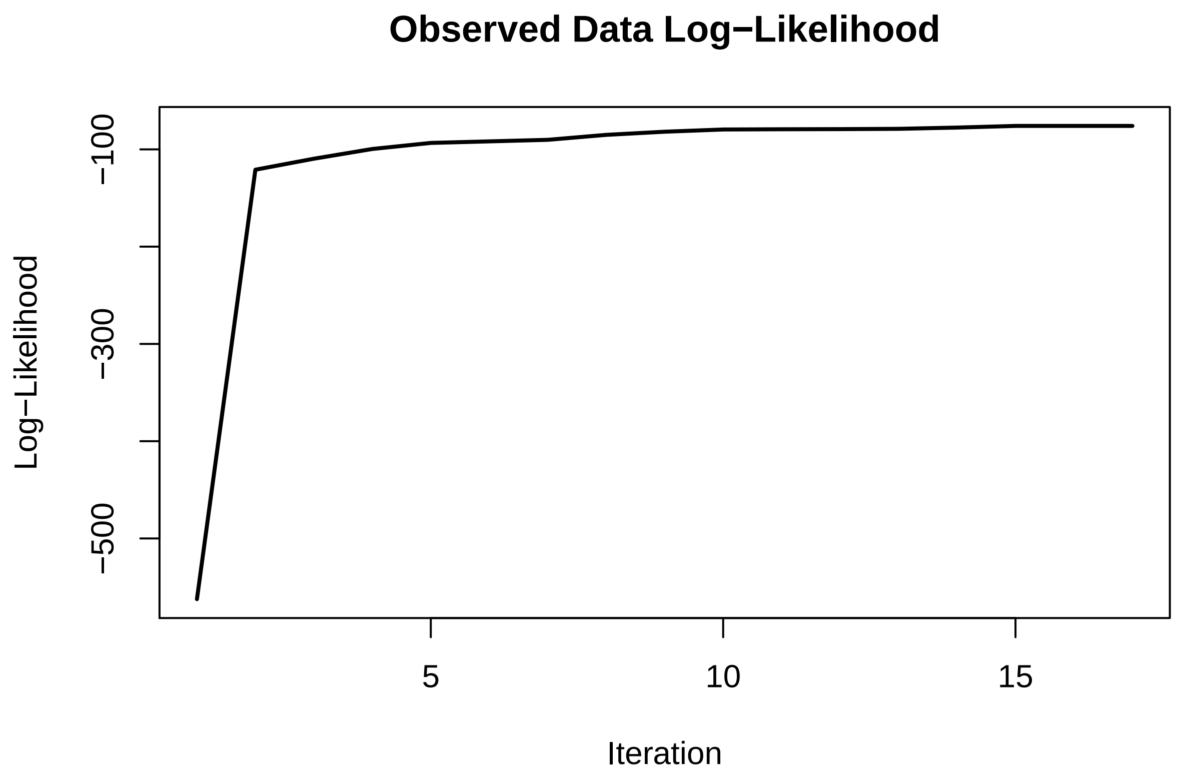
```r
posterior <- data.frame(cbind(gmm$x, gmm$posterior))
posterior$component <- ifelse(posterior$comp.1 > 0.5, 1, 2)
table(posterior$component)
```

```
##
##  1  2
## 44  5
```

Based on gmm, we can see that one cluster has 44 states whereas the other cluster has 5 states. These numbers are different from those found earlier using kmeans. In addition, with two lambda values (0.898 and 0.102), we could say that the larger lambda yields a higher curve, and substantively, that this cluster has more observations. Moreover, the probability scores of belonging to each cluster appear very high or low here. Usually Gaussian mixture model uses soft clustering technique, in which observations may belong to multiple clusters. Here, however, there seem to be very high or low probabilities for belonging to a cluster.

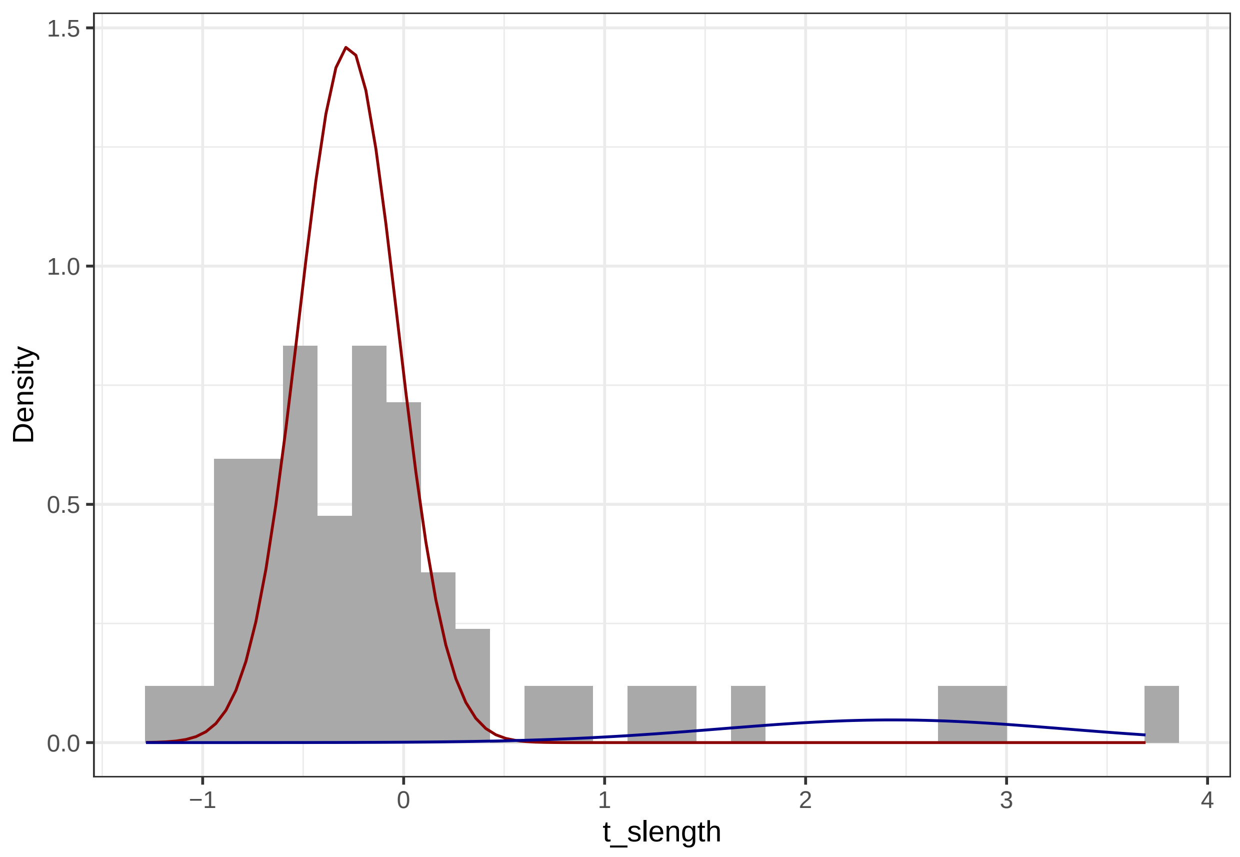
7. (15 points) Compare output of all in visually useful, simple ways (e.g., present the dendrogram, plot by state cluster assignment across two features like salary and expenditures, etc.). There should be several plots of comparison and output.

```r
plot.mixEM(gmm)
```



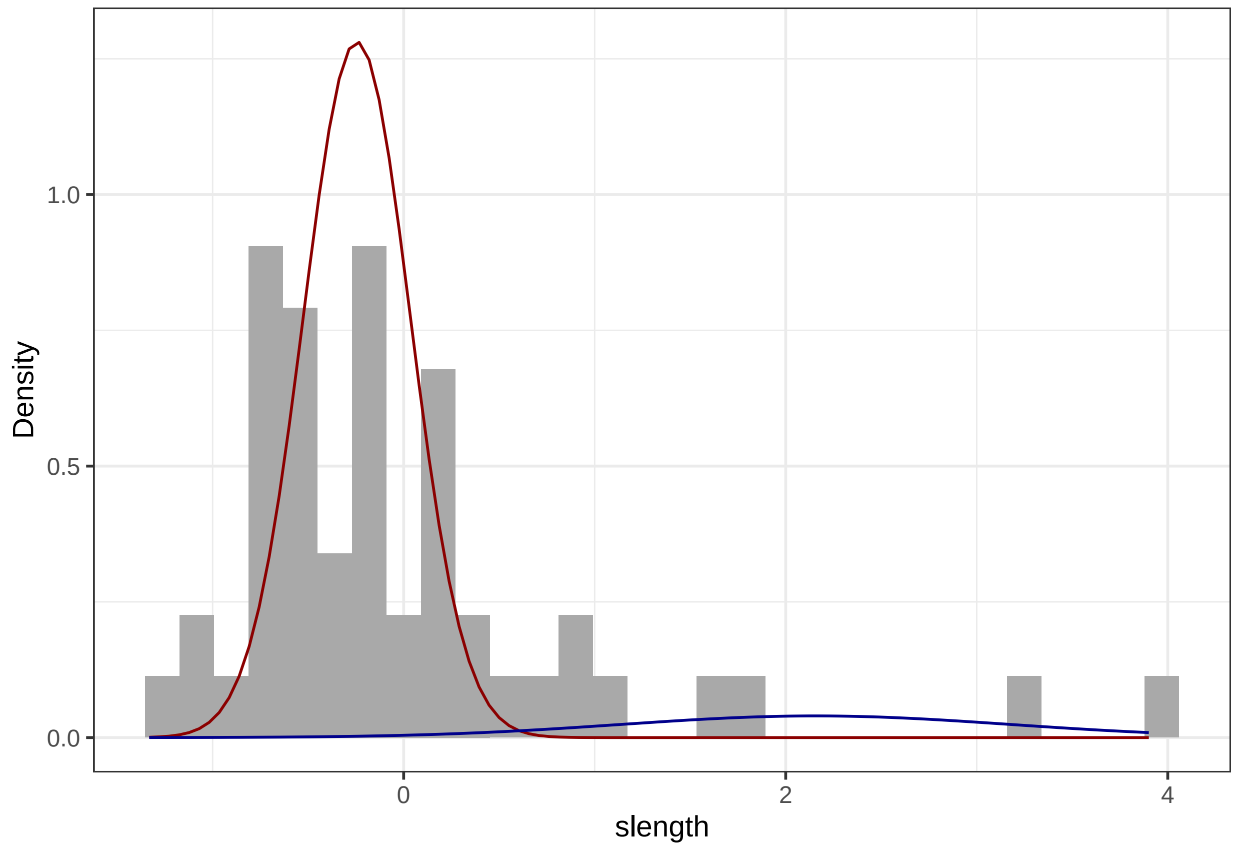
```r
ggplot(data.frame(x = gmm$x[,1])) +
  geom_histogram(aes(x, ..density..), fill = "darkgray") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm$mu[[1]][1], gmm$sigma[[1]][1], lam = gmm$lambda[1]),
                colour = "darkred") +
  stat_function(geom = "line",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

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



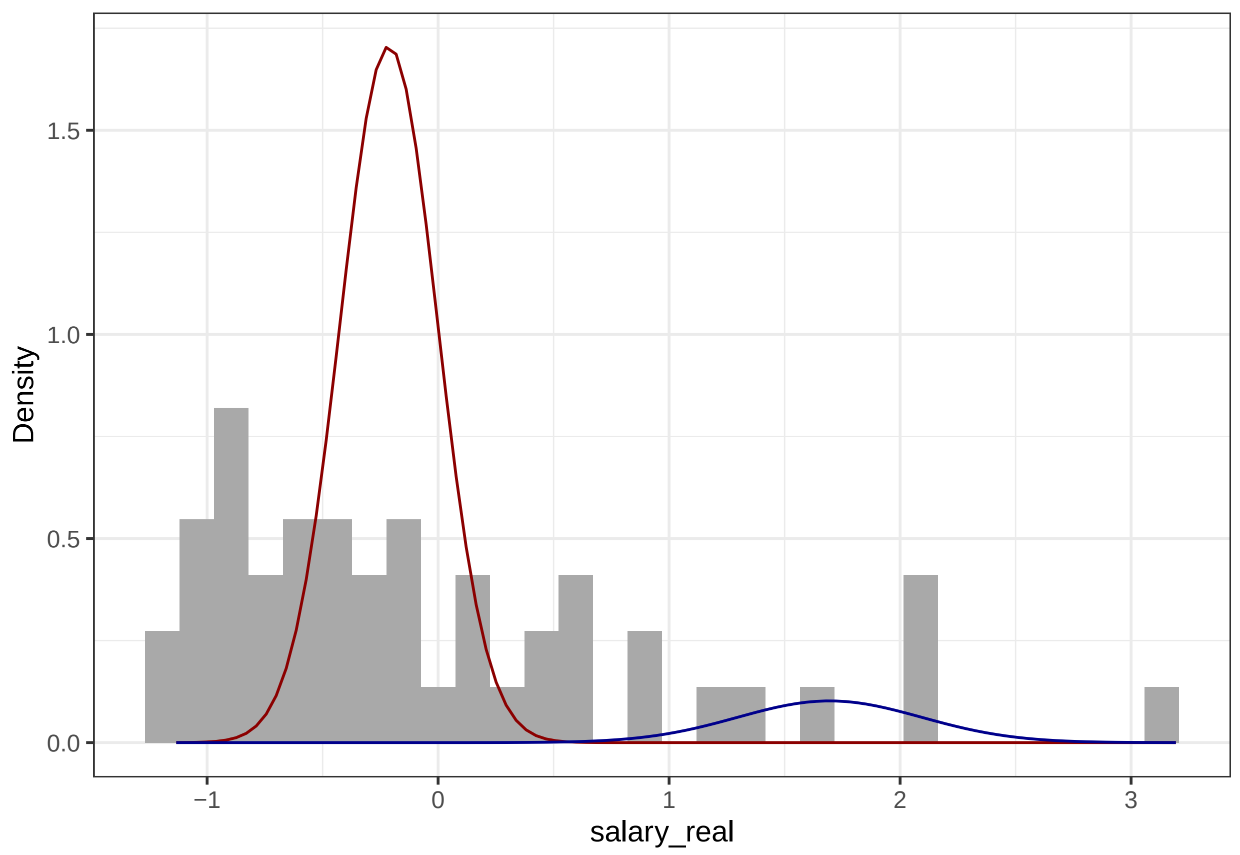
```
ggplot(data.frame(x = gmm$x[,2])) +
  geom_histogram(aes(x, ..density..), fill = "darkgray") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm$mu[[1]][2], gmm$sigma[[1]][2], lam = gmm$lambda[1]),
                colour = "darkred") +
  stat_function(geom = "line",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

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



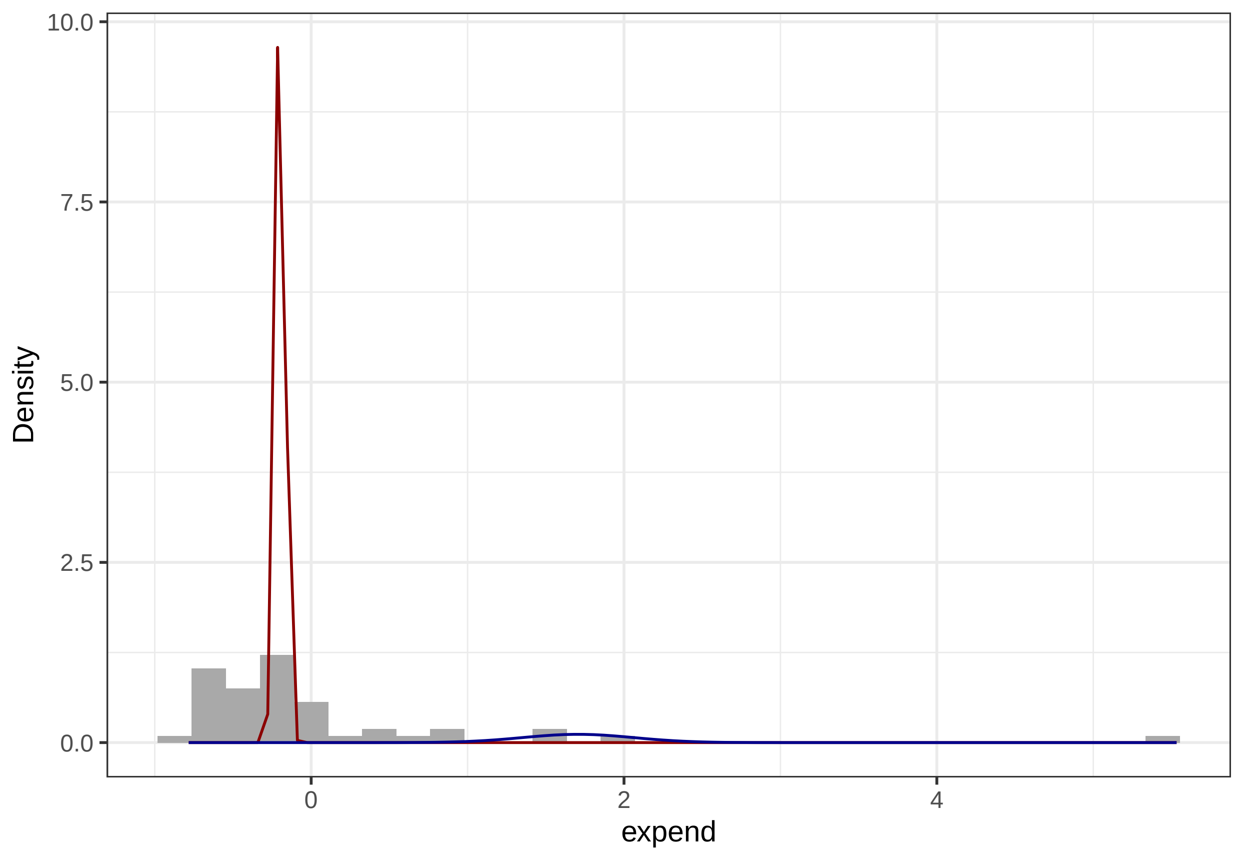
```r
ggplot(data.frame(x = gmm$x[,3])) +
  geom_histogram(aes(x, ..density..), fill = "darkgray") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm$mu[[1]][3], gmm$sigma[[1]][3], lam = gmm$lambda[1]),
                colour = "darkred") +
  stat_function(geom = "line",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

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



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

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



If the histogram shows a better fit of data, this suggests that certain variable is explaining the clustering well. Observations here are clustered based on 's_length' and 't_slength' along with some outliers in the data. If we look at the curve shown in the histogram that has 'salary_real' as the variable, it fits the data fairly well, meaning that it moderately accounts for the clustering. The curve in the histogram with a variable 'extend' fits the data poorly, thus suggesting that it has a weak influence on the clustering.

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.

```r
legprof_states_mat <- as.matrix(legprof_states)
clvalid <- clValid(legprof_states_mat, 2:5,
                          validation = "internal",
                          clMethods = c("model", "kmeans", "hierarchical"))
```

```
## Loading required package: 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

## Warning in clValid(legprof_states_mat, 2:5, validation = "internal", clMethods =
## c("model", : rownames for data not specified, using 1:nrow(data)
```

```r
summary(clvalid)
```

```
##
## Clustering Methods:
##  model kmeans hierarchical
##
## Cluster sizes:
##  2 3 4 5
##
## Validation Measures:
##                                  2       3       4       5
##
## model        Connectivity  10.8282 28.5869 39.0687 67.7833
##              Dunn           0.1512  0.0633  0.0224  0.0256
##              Silhouette     0.6313  0.2589  0.1860  0.0090
## kmeans       Connectivity   8.4571 10.9071 16.3996 28.9587
##              Dunn           0.1723  0.2585  0.2552  0.1091
##              Silhouette     0.6455  0.6127  0.4923  0.3032
## hierarchical Connectivity   6.0869  6.9536 16.3996 18.7774
##              Dunn           0.3605  0.4358  0.2552  0.2819
##              Silhouette     0.6992  0.6706  0.4923  0.4433
##
## Optimal Scores:
##
##              Score  Method       Clusters
## Connectivity 6.0869 hierarchical 2
## Dunn         0.4358 hierarchical 3
## Silhouette   0.6992 hierarchical 2
```

According to the result, the hiearchical method shows as the strongest performer on all three dimensions (Connectivity, Dunn, Silhouette). If the method has high values for silhouette width and the Dunn index and low values on connectivity, it is considered as the strongest performer.

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?"

The internal validation helped us to see how well clustering algorithms perform relative to other specifications even in a circumstance when we did not have a label that we could use to validate via an external measure.

To do so, we compared across different numbers of clusters and types of clustering to see which method is optimal. There is no consistent best performing method among Dunn, silhouette width, and connectivity, so we prioritized which metrics should be considered the most important.

In this case, the hiearchical clustering algorithm was the optimal method across all three measures of internal validation (as mentioned in Q8). To be more specific, 2 clusters would be the optimal for connectivity and silhouette width, while 3 clusters would be optimal for Dunn index.

It is nearly possible to come up with the strongest performer on all dimensions. Therefore, we may need to choose which measure should be prioritized and what number of clusters are commonly used in the field of interest. For instance, we would use the sub-optimal clustering method if particular method has more information in the dataset. GMM, for example, has more information such as probability of belonging to one of the clusters than kmeans so that we would select GMM over kmeans method even if it is suboptimal.