Problem Set 4

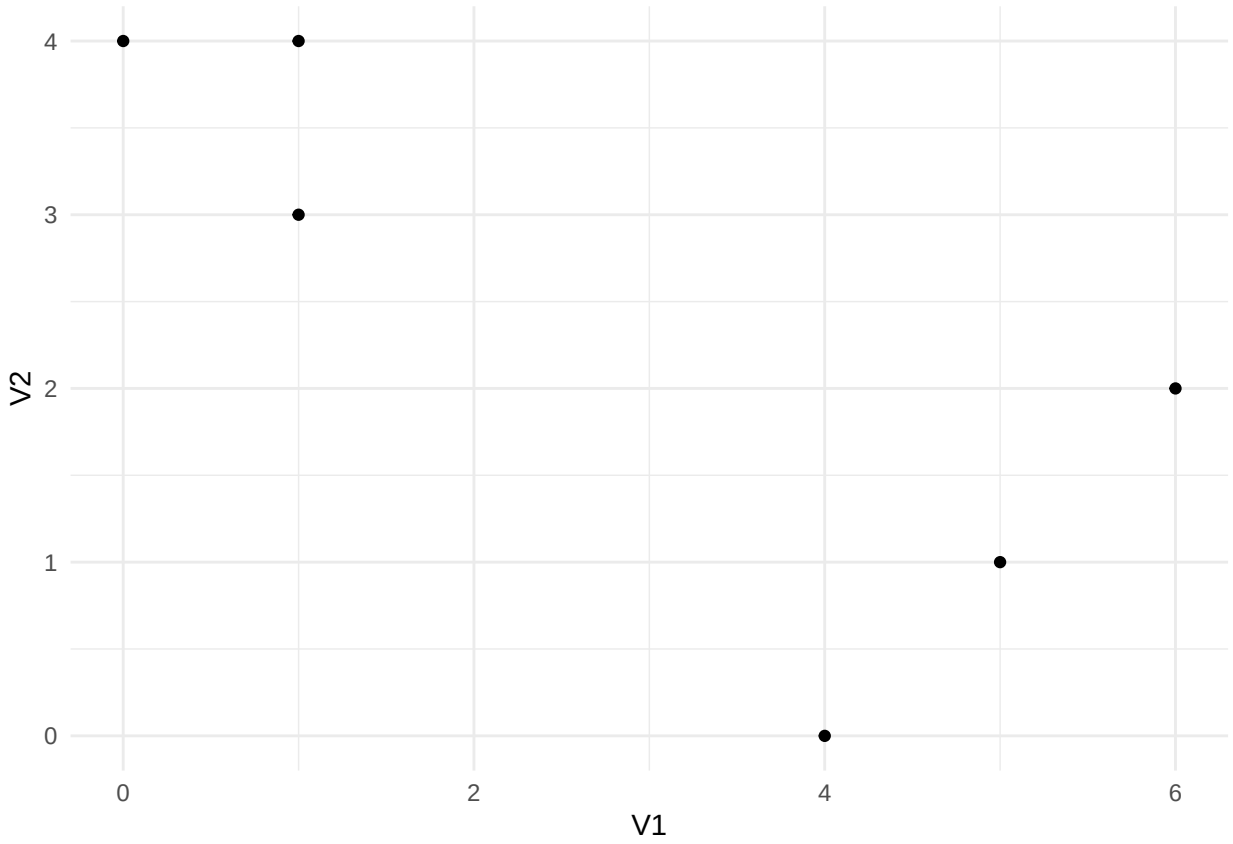
Seth Harrison

3/2/2020

```
library(tidyverse)
library(seriation)
library(knitr)
library(mixtools)
library(clValid)
library(mclust)
library(plotGMM)
```

1. (5 points) Plot the observations.

```
data <- cbind(c(1, 1, 0, 5, 6, 4), c(4, 3, 4, 1, 2, 0)) %>%
      as.data.frame()

data%>%
  ggplot(aes(x = V1, y = V2)) +
  geom_point() +
  theme_minimal()
```

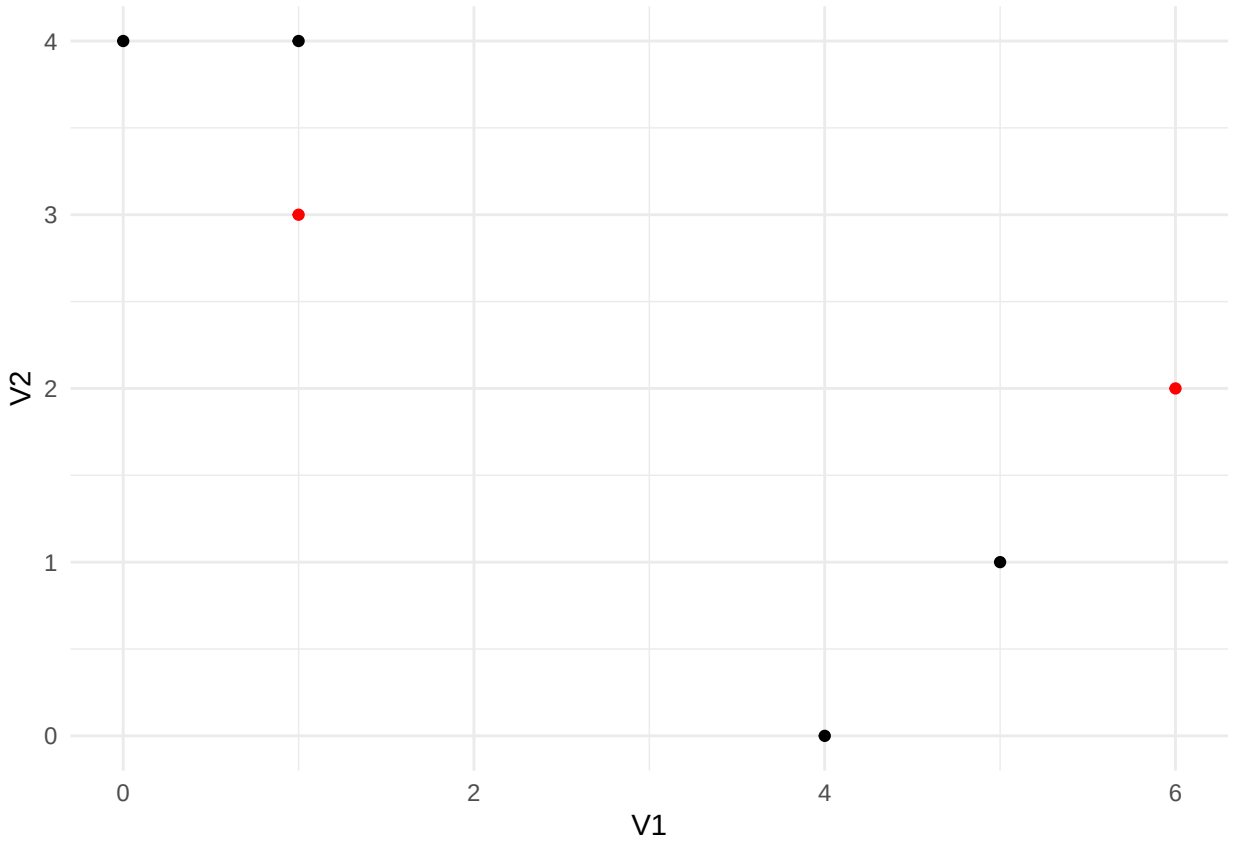


2. (5 points) Randomly assign a cluster label to each observation. Report the cluster labels for each observation and plot the results with a different color for each cluster (remember to set your seed first).

```
set.seed(1)

random <- sample(2, nrow(data), replace = TRUE)
random
```

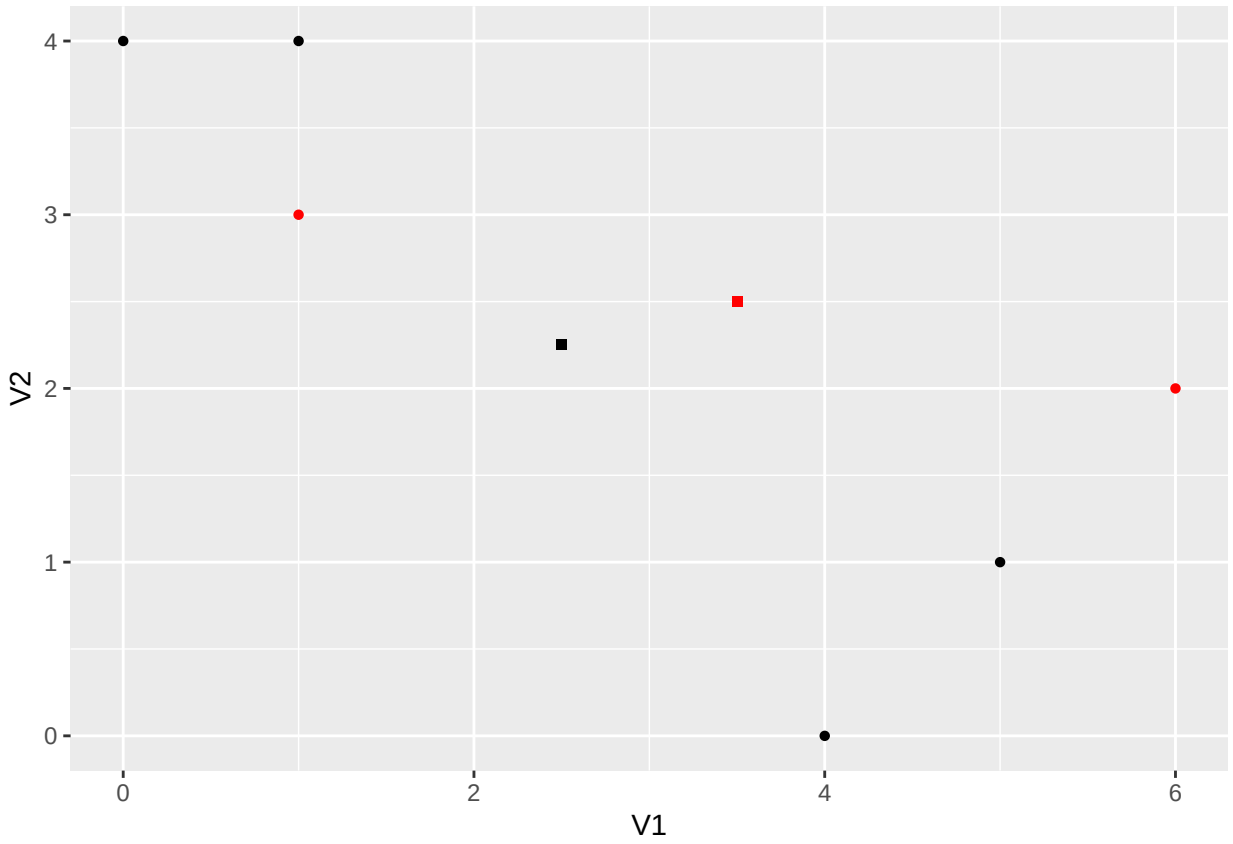
```
## [1] 1 2 1 1 2 1
```

```
data %>%
  ggplot(aes(x = V1, y = V2)) +
  geom_point(color = random) +
  theme_minimal()
```



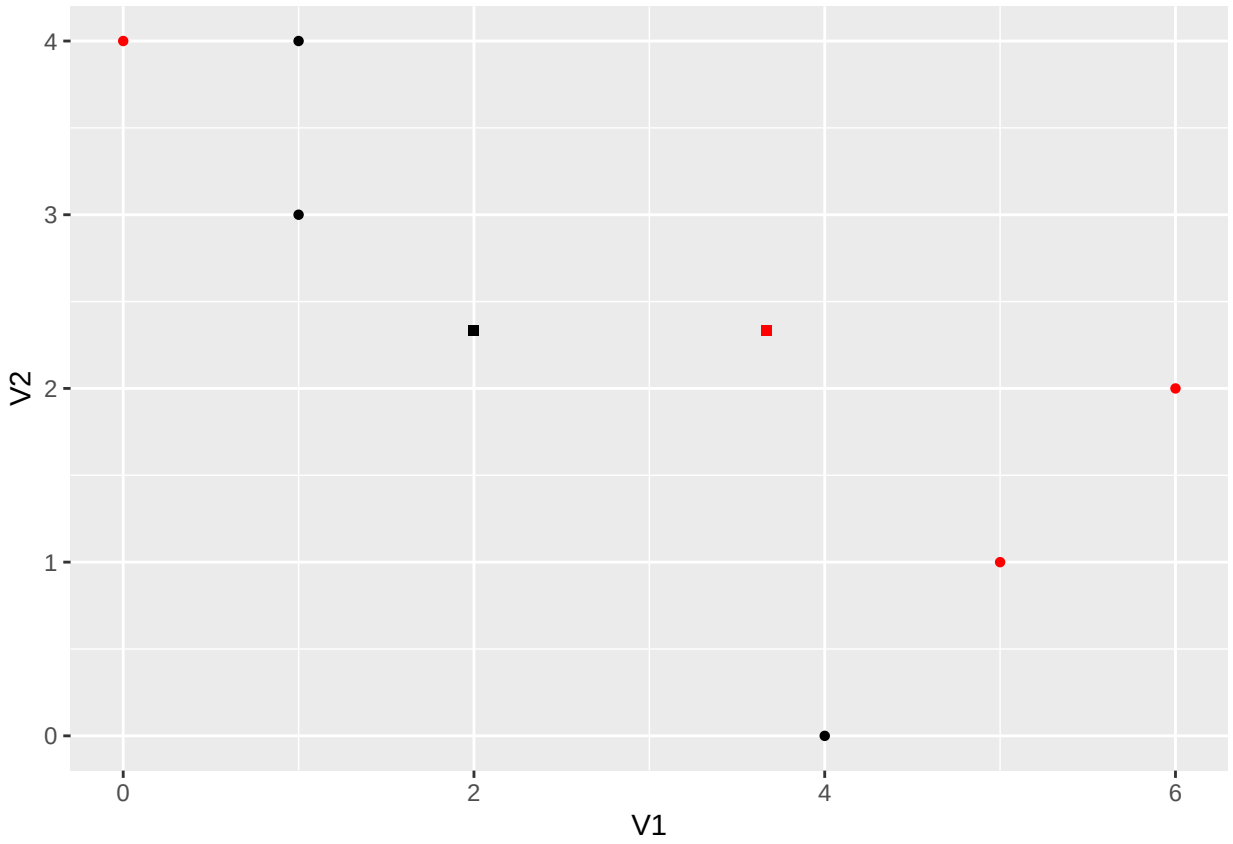
3. (10 points) Compute the centroid for each cluster.

```
centroid <- c(mean(data[random == 1, 1]),
              mean(data[random == 1, 2]))
centroid2 <- c(mean(data[random == 2, 1]),
               mean(data[random == 2, 2]))

datai1 <- data %>%
        rbind(centroid, centroid2) %>%
        mutate(centroid_01 = if_else(V1 == 2.5 | V1 == 3.5, 15, 16)) %>%
        mutate(class = c(1,2,1,1,2,1,1,2))

datai1 %>%
  ggplot(aes(x = V1, y = V2)) +
  geom_point(shape = datai1$centroid_01,
             color = datai1$class)
```



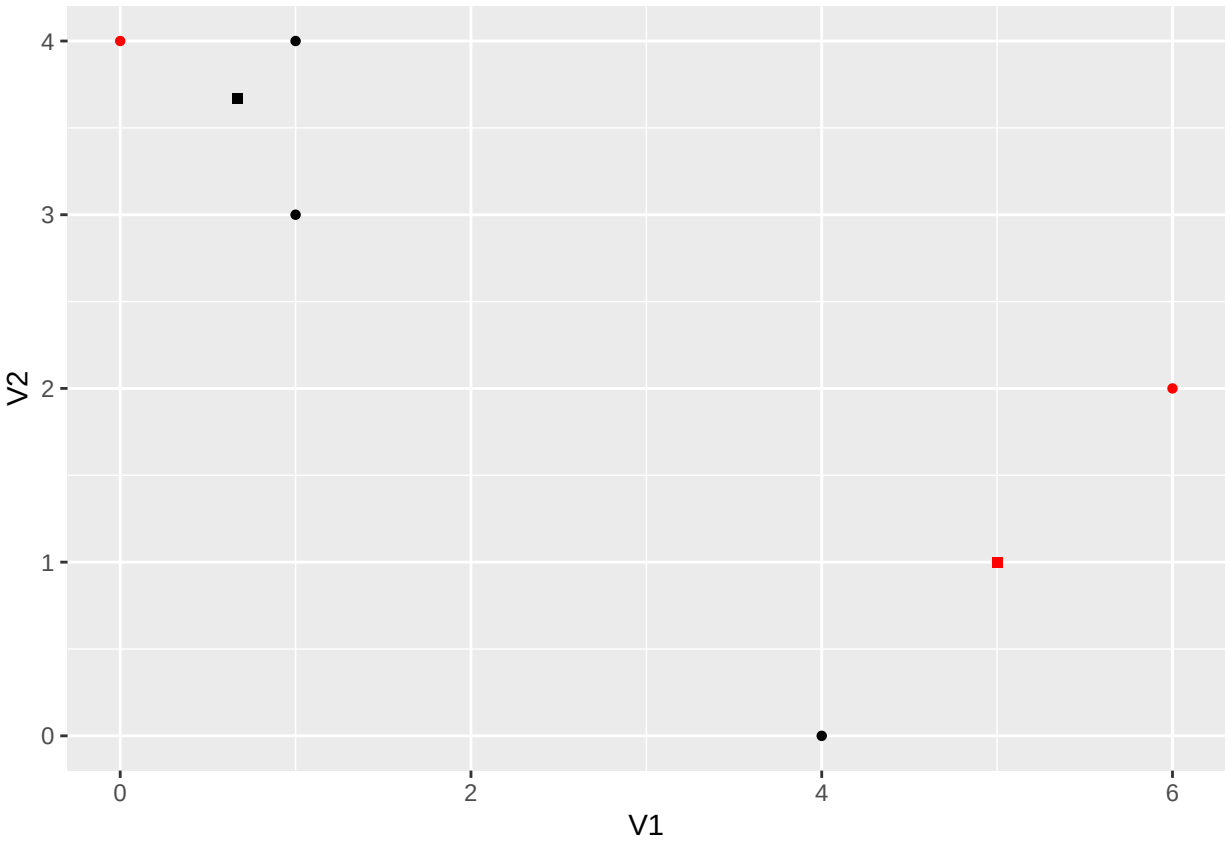
4. (10 points) Assign each observation to the centroid to which it is closest, in terms of Euclidean distance. Report the cluster labels for each observation.

```
notrandom <- c(1,1,2,2,2,1)

centroid3 <- c(mean(data[notrandom == 1, 1]),
               mean(data[notrandom == 1, 2]))
centroid4 <- c(mean(data[notrandom == 2, 1]),
               mean(data[notrandom == 2, 2]))

datai2 <- data %>%
        rbind(centroid3, centroid4) %>%
        mutate(centroid_01 = if_else(V1 == 2 | V1 == 11/3, 15, 16)) %>%
        mutate(class = c(1,1,2,2,2,1,1,2))

datai2 %>%
  ggplot(aes(x = V1, y = V2)) +
  geom_point(shape = datai2$centroid_01,
             color = datai2$class)
```



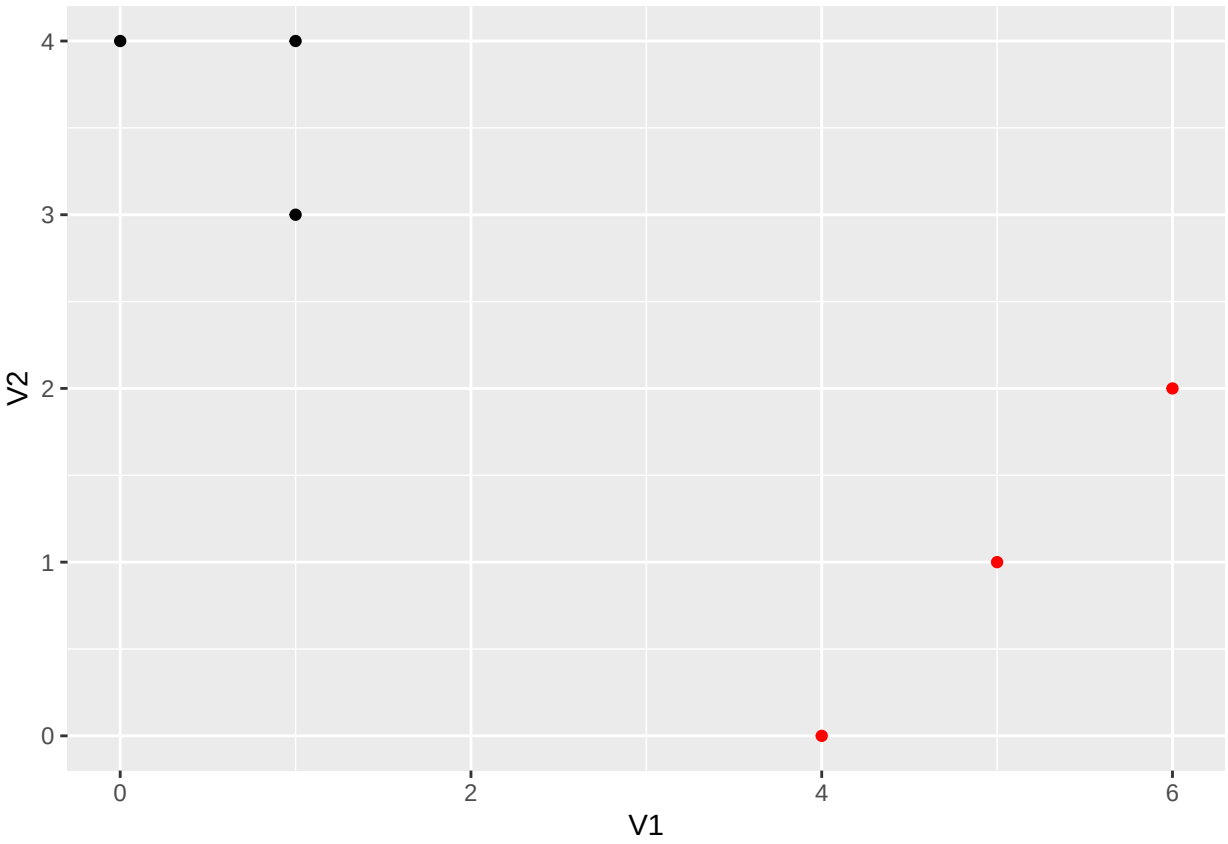
5. (5 points) Repeat (3) and (4) until the answers/clusters stop changing.

```
stable <- c(1,1,1,2,2,2)

centroid5 <- c(mean(data[stable == 1, 1]),
               mean(data[stable == 1, 2]))
centroid6 <- c(mean(data[stable == 2, 1]),
               mean(data[stable == 2, 2]))

datai3 <- data %>%
        rbind(centroid5, centroid6) %>%
        mutate(centroid_01 = if_else(V1 == 2/3 | V1 == 5, 15, 16)) %>%
        mutate(class = c(1,1,1,2,2,2,1,2))

datai3 %>%
  ggplot(aes(x = V1, y = V2)) +
  geom_point(shape = datai2$centroid_01,
             color = datai2$class)
```



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.

```
data4 <- cbind(data,stable)

data4 %>%
  ggplot(aes(x = V1, y = V2)) +
  geom_point(color = stable)
```



Clustering State Legislative Professionalism

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

```
legprof <- load("~/myrepo/Machine Learning/Problem-Set-4/Data and Codebook/legprof-components.v1.0.RData

legprof <- 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;

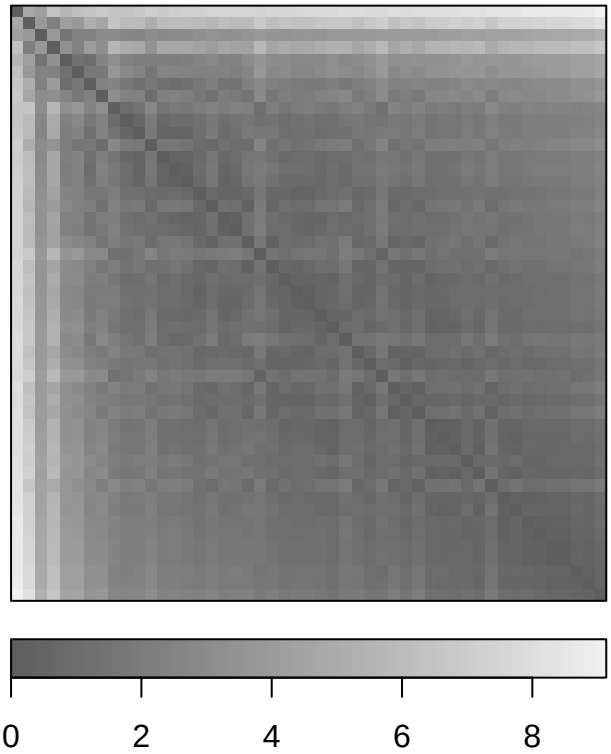
e. and anything else you think necessary to get this subset of data into workable form (hint: consider storing the state names as a separate object to be used in plotting later)

```
subset <- legprof %>%
        filter(sessid =="2009/10") %>%
        select(state,
               t_slength,
```

```
                slength,
                salary_real,
                expend)

subsetscale <- scale(subset[, -c(1)]) %>%
                   as_tibble() %>%
                   na.omit()

states <- subset %>%
          select(state) %>%
          filter(state!= "Wisconsin")
```

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.

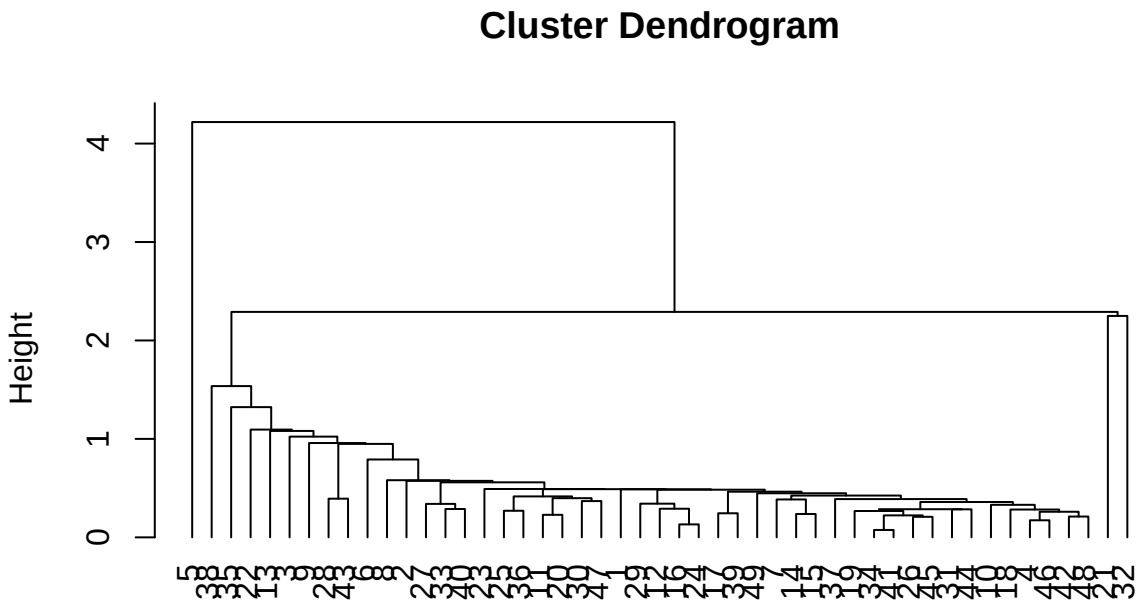
```
ODI <- dist(subsetscale, method = "euclidean")

dissplot(ODI)
```



The data do not look particularly clusterable. I would like to see darker blocks along the diagonal of the matrix.

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.

```
AHC <- hclust(ODI, method = "single")

plot(AHC, hang = -1)
```



At the bottom, dendrogram shows several pairs of states with similarily professionalized legislatures. For example, Tennessee (42) is similar to West Virginia (48) and Arkansas (4) is similar to Virginia (46). At the top, the dendrogram shows that California (5) is unique compared to the other 48 (one is omitted) states in terms of professionalized legislatures.

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.

```
set.seed(1)

kmeans2 <- kmeans(subsetscale,
                  centers = 2,
                  nstart = 15)

t <- as.table(kmeans2$cluster)

t <- data.frame(t)
```

```
t$Var1 <- NULL

cbind(states,t) %>%
  kable()
```

state	Freq
Alabama	1
Alaska	1
Arizona	1
Arkansas	1
California	2
Colorado	1
Connecticut	1
Delaware	1
Florida	1
Georgia	1
Hawaii	1
Idaho	1
Illinois	1
Indiana	1
Iowa	1
Kansas	1
Kentucky	1
Louisiana	1
Maine	1
Maryland	1
Massachusetts	2
Michigan	2
Minnesota	1
Mississippi	1
Missouri	1
Montana	1
Nebraska	1
Nevada	1
New Hampshire	1
New Jersey	1
New Mexico	1
New York	2
North Carolina	1
North Dakota	1
Ohio	2
Oklahoma	1
Oregon	1
Pennsylvania	2
Rhode Island	1
South Carolina	1
South Dakota	1
Tennessee	1
Texas	1
Utah	1
Vermont	1
Virginia	1
Washington	1

state	Freq
West Virginia	1
Wyoming	1

Forty-three states were placed in the same cluster. Roughly, it seems that the six states placed in the second cluster tend to be more wealthy and populous.

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(3)

gmm1 <- mvnormalmixEM(subsetscale, k = 2)
```

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

```
gmm1
```

```
## $x
##          t_slength     slength salary_real      expend
##  [1,] -0.371659901 -0.45947229 -1.11233507 -0.24448046
##  [2,] -0.229408926 -0.14523091  0.38325317  0.86590633
##  [3,]  1.645306675  0.79519547 -0.15232455 -0.13330124
##  [4,] -0.803646214 -0.78817557 -0.51173889 -0.26591316
##  [5,]  2.880725686  1.77670986  3.19372289  5.53272310
##  [6,]  0.682733830  0.90088869  0.09300309 -0.35415622
##  [7,]  0.155953294  0.07723846  0.01122723 -0.19589855
##  [8,] -0.643316513 -0.61557933  0.59262728 -0.54337729
##  [9,] -0.550306302 -0.65382896  0.09378402  1.55326725
## [10,] -0.806381805 -0.79128419 -0.42455645 -0.58447303
## [11,] -0.247368734 -0.22362235  0.85793472 -0.03331348
## [12,] -0.026736988  0.09467375 -0.47468508 -0.61261906
## [13,]  0.252174862  0.33052374  1.64003924 -0.02468402
## [14,] -0.060515652 -0.13563473 -0.20889464 -0.28192756
## [15,] -0.212519663 -0.11644243 -0.11143661 -0.43443875
## [16,]  0.043793814  0.17482220 -0.85945920 -0.58199301
## [17,] -0.238210429 -0.29390430 -0.68785022 -0.13112235
## [18,] -0.441952865 -0.60746990 -0.22312061 -0.01084832
## [19,] -0.627973378 -0.58854782 -0.65895804 -0.61278369
## [20,] -0.237853581 -0.14523091  0.64499032 -0.06020858
## [21,]  2.821493958  3.33129222  1.24756200 -0.31046122
## [22,]  0.775506248  1.00631164  2.12308936  0.45955958
## [23,] -0.461458846 -0.42635872  0.13965218 -0.29289554
## [24,]  0.078048154  0.17144321 -0.73428667 -0.60301895
## [25,] -0.009490831  0.11427161  0.33485529 -0.52157294
## [26,] -0.687442853 -0.65612661 -0.98034416 -0.60575115
## [27,]  0.133235911  0.11427161 -0.64297985 -0.05707593
## [28,] -0.693865590 -0.72100229 -0.87567943  0.62048488
## [29,]  0.010371952  0.12332714 -1.12954636 -0.78272496
## [30,] -0.259262629 -0.18307507  0.82317239  0.50109836
## [31,] -0.904982206 -1.00888793 -1.13363515 -0.38076899
```

```
## [32,]  3.691294567  3.90071117  2.11695616  1.48449705
## [33,]  0.403940989  0.58407939 -0.56320747 -0.22491071
## [34,] -0.818275700 -0.80479995 -0.91382656 -0.68417895
## [35,]  1.310731529  1.61452076  1.34351995 -0.23744167
## [36,] -0.152217572 -0.04791749  0.43646185 -0.39552742
## [37,] -0.322300309 -0.24119288 -0.24996495  0.03231297
## [38,]  1.120429207  0.97928013  2.06849000  1.95555567
## [39,] -0.294944314 -0.21010659 -0.59845289 -0.33491369
## [40,]  0.258478612  0.41878162 -0.70840055 -0.10985277
## [41,] -0.818275700 -0.80479995 -0.88830750 -0.75450729
## [42,] -0.556610007 -0.61557933 -0.35639625 -0.23158722
## [43,] -0.558750912 -0.52907846 -0.83924197  0.93319558
## [44,] -0.989428844 -1.00888788 -0.97222941 -0.59932522
## [45,] -0.599190173 -0.57503206 -0.88725099 -0.74861360
## [46,] -0.679355001 -0.83615650 -0.41480789 -0.21212016
## [47,] -0.111183625 -0.28917375  0.58799257  0.28381220
## [48,] -0.567195612 -0.65382896 -0.31587632 -0.43514108
## [49,] -1.282137610 -1.33191452 -1.01097132 -0.69382507
##
## $lambda
## [1] 0.1227436 0.8772564
##
## $mu
## $mu[[1]]
## [1] 2.094522 2.095865 2.014059 1.476489
##
## $mu[[2]]
## [1] -0.2930604 -0.2932483 -0.3029357 -0.2088978
##
##
## $sigma
## $sigma[[1]]
##           [,1]       [,2]       [,3]       [,4]
## [1,] 1.1751098  1.0805707  0.1464243  0.6942696
## [2,] 1.0805707  1.2666233 -0.1625594 -0.3383160
## [3,] 0.1464243 -0.1625594  0.4084750  1.2025634
## [4,] 0.6942696 -0.3383160  1.2025634  3.9719814
##
## $sigma[[2]]
##            [,1]        [,2]      [,3]        [,4]
## [1,] 0.25252957 0.232775959 0.1015790 0.020271766
## [2,] 0.23277596 0.238827668 0.1054384 0.009346694
## [3,] 0.10157901 0.105438413 0.4042302 0.106424641
## [4,] 0.02027177 0.009346694 0.1064246 0.235276639
##
##
## $loglik
## [1] -107.4313
##
## $posterior
##             comp.1       comp.2
##  [1,] 2.973398e-46 1.000000e+00
##  [2,] 1.753989e-28 1.000000e+00
##  [3,] 1.351662e-62 1.000000e+00
```

```
##  [4,] 7.799968e-32 1.000000e+00
##  [5,] 1.000000e+00 3.697415e-39
##  [6,] 8.040426e-13 1.000000e+00
##  [7,] 7.571697e-16 1.000000e+00
##  [8,] 2.269085e-07 9.999998e-01
##  [9,] 2.245653e-47 1.000000e+00
## [10,] 1.868805e-24 1.000000e+00
## [11,] 1.642648e-05 9.999836e-01
## [12,] 3.096569e-24 1.000000e+00
## [13,] 2.480935e-02 9.751906e-01
## [14,] 8.011349e-20 1.000000e+00
## [15,] 2.312493e-18 1.000000e+00
## [16,] 1.678480e-35 1.000000e+00
## [17,] 1.332583e-34 1.000000e+00
## [18,] 1.636255e-22 1.000000e+00
## [19,] 3.950427e-30 1.000000e+00
## [20,] 3.151292e-08 1.000000e+00
## [21,] 1.000000e+00 1.028207e-12
## [22,] 9.915146e-01 8.485423e-03
## [23,] 1.097640e-14 1.000000e+00
## [24,] 3.133665e-30 1.000000e+00
## [25,] 2.211343e-09 1.000000e+00
## [26,] 3.493773e-40 1.000000e+00
## [27,] 1.903138e-33 1.000000e+00
## [28,] 8.443574e-66 1.000000e+00
## [29,] 5.001174e-40 1.000000e+00
## [30,] 9.552899e-11 1.000000e+00
## [31,] 1.299371e-46 1.000000e+00
## [32,] 1.000000e+00 1.951204e-17
## [33,] 1.617476e-32 1.000000e+00
## [34,] 1.108369e-36 1.000000e+00
## [35,] 9.981527e-01 1.847263e-03
## [36,] 6.515127e-09 1.000000e+00
## [37,] 1.569046e-29 1.000000e+00
## [38,] 9.999434e-01 5.658730e-05
## [39,] 8.410688e-33 1.000000e+00
## [40,] 2.169725e-40 1.000000e+00
## [41,] 1.015713e-34 1.000000e+00
## [42,] 3.615226e-25 1.000000e+00
## [43,] 4.907940e-79 1.000000e+00
## [44,] 6.574955e-40 1.000000e+00
## [45,] 3.774656e-34 1.000000e+00
## [46,] 1.033356e-25 1.000000e+00
## [47,] 1.375722e-09 1.000000e+00
## [48,] 7.119409e-22 1.000000e+00
## [49,] 1.046126e-39 1.000000e+00
##
## $all.loglik
##  [1] -426.5945 -123.2988 -117.0074 -114.1647 -113.2423 -112.9772 -111.0691
##  [8] -107.4597 -107.4314 -107.4313 -107.4313 -107.4313 -107.4313 -107.4313
## [15] -107.4313
##
## $restarts
## [1] 0
```

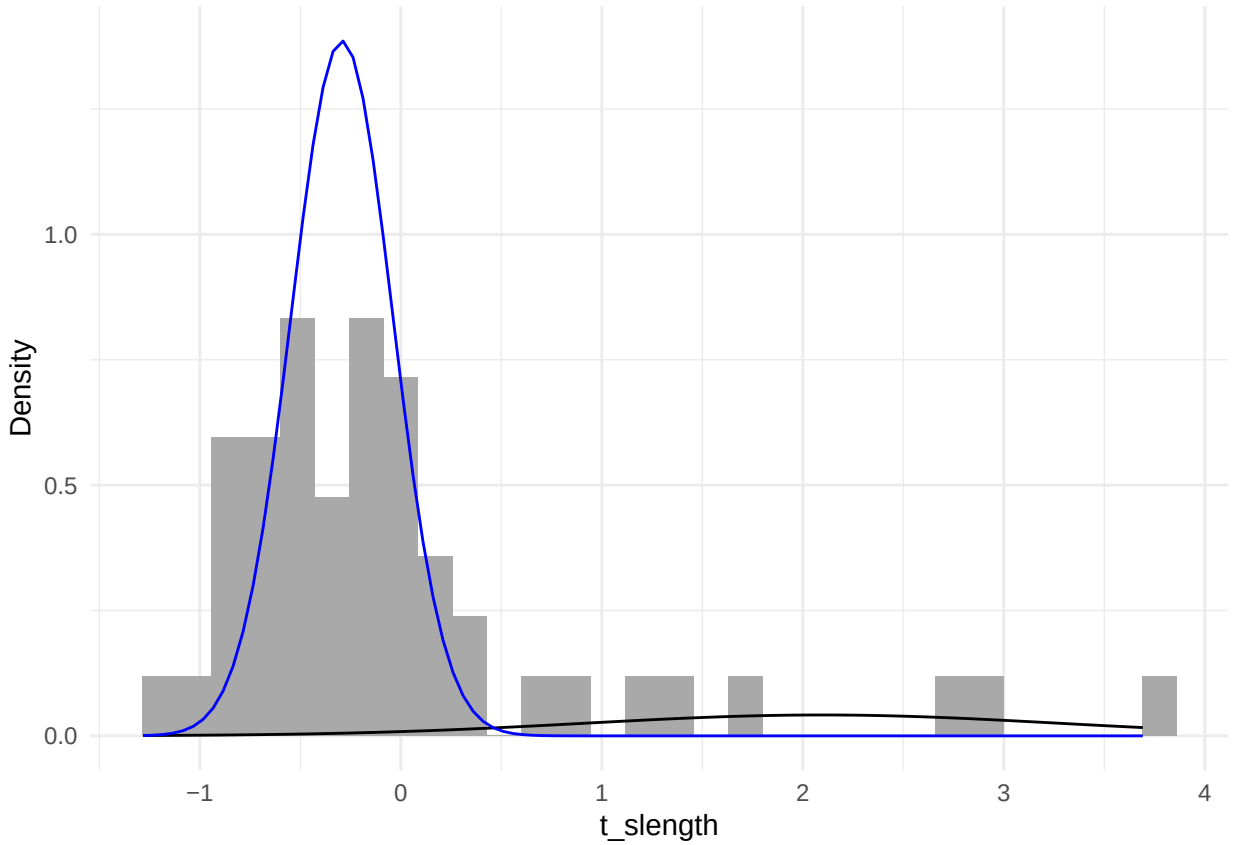
```
##
## $ft
## [1] "mvnormalmixEM"
##
## attr(,"class")
## [1] "mixEM"
```

The first distribution with has more observations (lamda), a lower mean (mu), and is more tightly spread (sigma).

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.

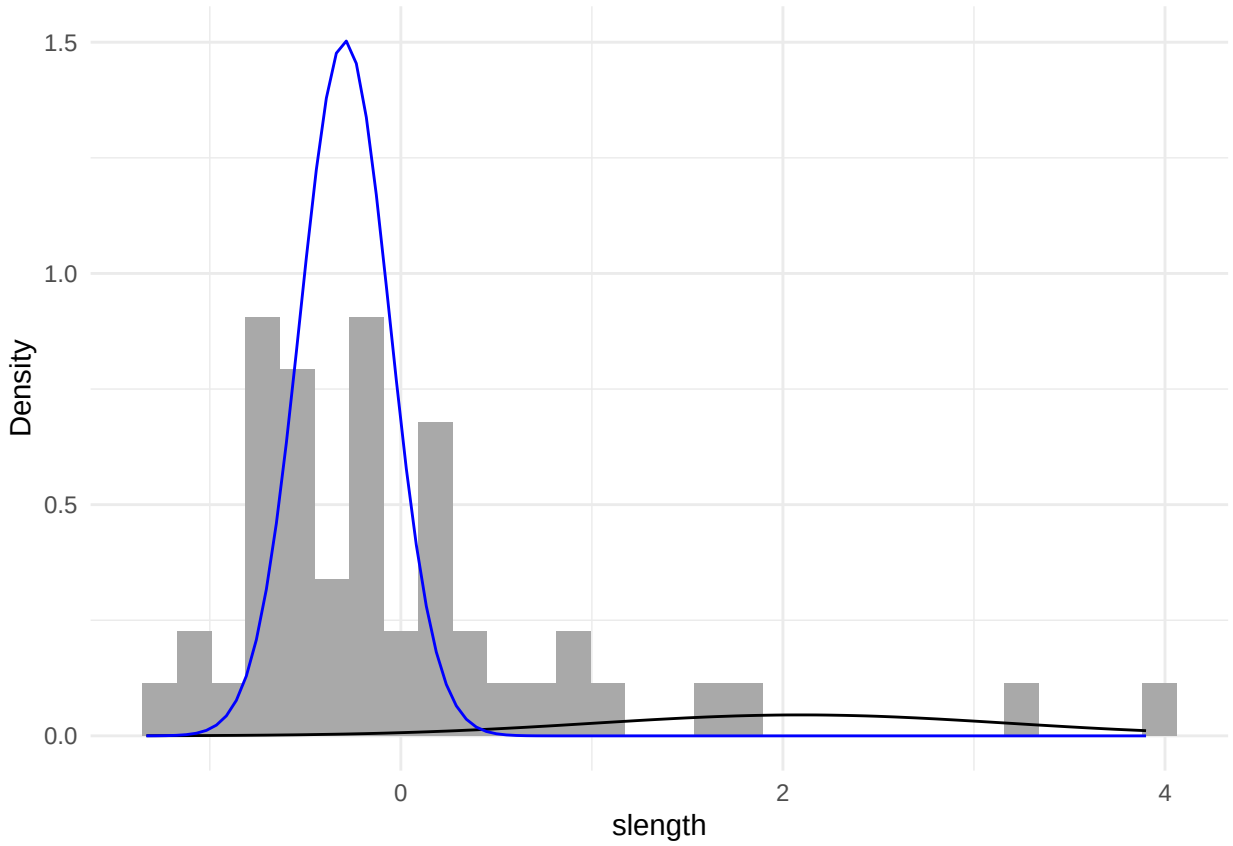
```
ggplot(data.frame(x = gmm1$x[,1])) +
  geom_histogram(aes(x, ..density..), fill = "darkgray") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm1$mu[[1]][1],
                            gmm1$sigma[[1]][1],
                            lam = gmm1$lambda[1]),
   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

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




```
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 = "black") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm1$mu[[2]][2], gmm1$sigma[[2]][2], lam = gmm1$lambda[2]),
                colour = "blue") +
  xlab("slength") +
  ylab("Density") +
  theme_minimal()
```

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

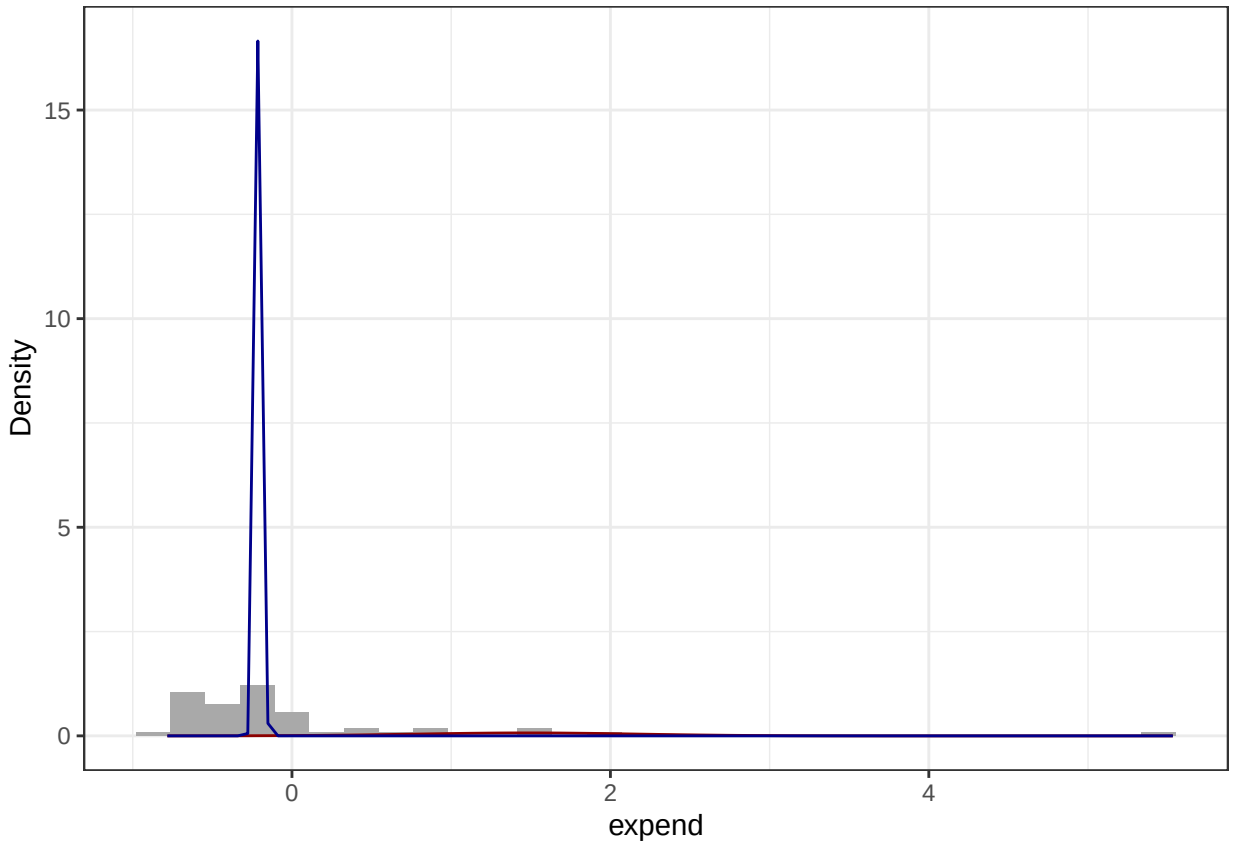



```
ggplot(data.frame(x = gmm1$x[,3])) +
  geom_histogram(aes(x, ..density..), fill = "darkgray") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm1$mu[[1]][3],
                            gmm1$sigma[[1]][3],
                            lam = gmm1$lambda[1]),
   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

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


```
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()
```

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



Salary_real seems to fit the distributions the best, indicating it has the most influence in the clustering. Expenditure does not, indicating it does not have very much influence over 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.

```
legprof_matrix <- as.matrix(subsetscale)
clvalid <- clValid(legprof_matrix, 2:20,
                           validation = "internal",
                           clMethods = c("model", "kmeans", "hierarchical"))
```

```
## Warning in clValid(legprof_matrix, 2:20, validation = "internal", clMethods =
## c("model", : rownames for data not specified, using 1:nrow(data)
```

```
summary(clvalid)
```

```
##
## 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       9      10
##
## model        Connectivity  10.8282 28.5869 39.0687 67.7833 80.2960 69.9317 72.4504 47.1464 60.5187 67
##              Dunn           0.1512  0.0633  0.0224  0.0256  0.0283  0.0543  0.0709  0.1809  0.0971  0
##              Silhouette     0.6313  0.2589  0.1860  0.0090 -0.0556  0.0919  0.0751  0.2825  0.1899  0
## kmeans       Connectivity   8.4571 10.9071 16.3996 28.9587 30.9587 37.7306 39.7679 41.1988 45.7417 45
##              Dunn           0.1723  0.2585  0.2552  0.1091  0.1091  0.1107  0.1251  0.1331  0.1392  0
##              Silhouette     0.6455  0.6127  0.4923  0.3032  0.2849  0.2741  0.3125  0.3308  0.3283  0
## hierarchical Connectivity   6.0869  6.9536 16.3996 18.7774 20.7774 21.8607 27.6476 35.6813 37.6147 47
##              Dunn           0.3605  0.4358  0.2552  0.2819  0.2819  0.2819  0.2956  0.1578  0.1578  0
##              Silhouette     0.6992  0.6706  0.4923  0.4433  0.4278  0.3514  0.2569  0.2665  0.2642  0
##
## Optimal Scores:
##
##              Score  Method       Clusters
## Connectivity 6.0869 hierarchical 2
## Dunn         0.4358 hierarchical 3
## Silhouette   0.6992 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?"

Internal validation shows that AHC with 2 clusters is optimial. GMM allows for probablistic assesments because, as a soft partioning method, it allows clusters to overlap. AHC is also computationally expensive and requries that k and n be small.