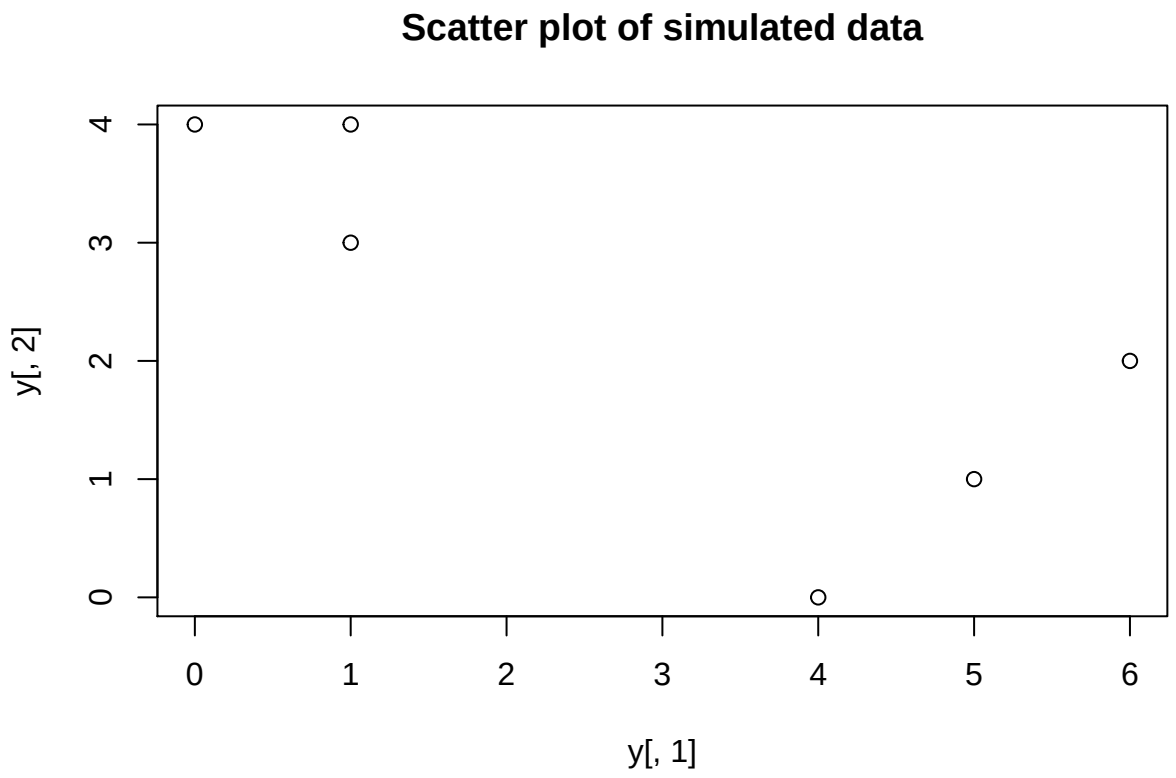
PS4

Grace Wright

2/24/2020

1. (5 points) Plot the observations.

```
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")
```

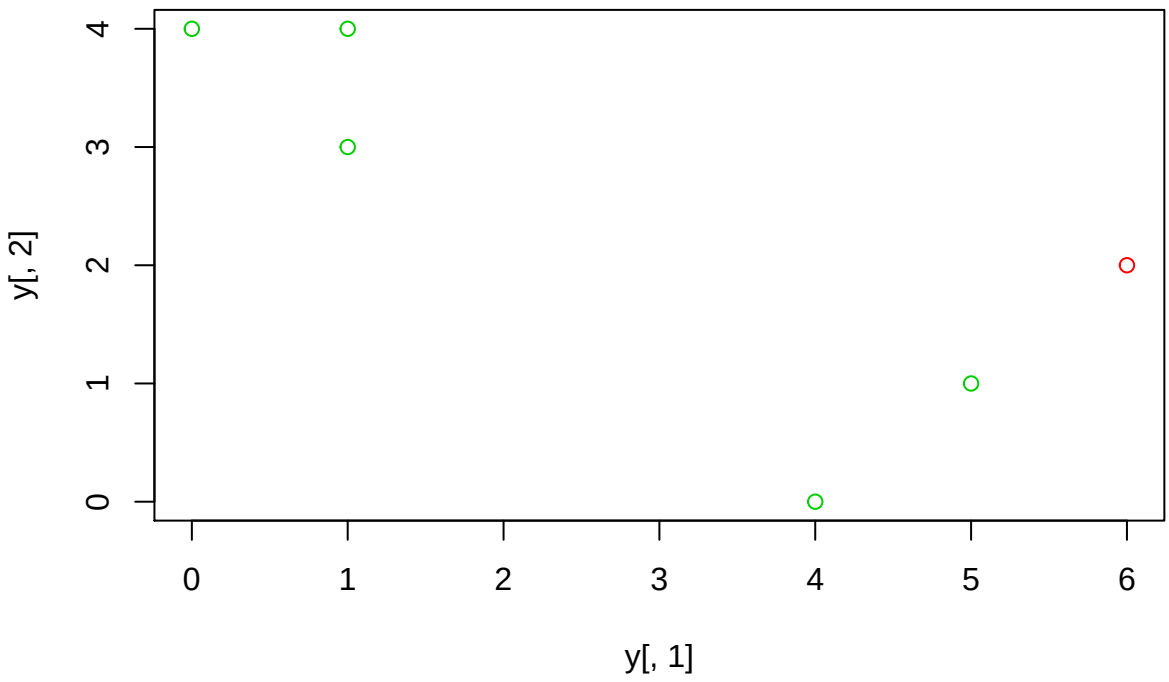


2. (5 points) Randomly assign a cluster label to each observation. Report the cluster labels for each observation and plot the results with a different color for each cluster (remember to set your seed first).

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

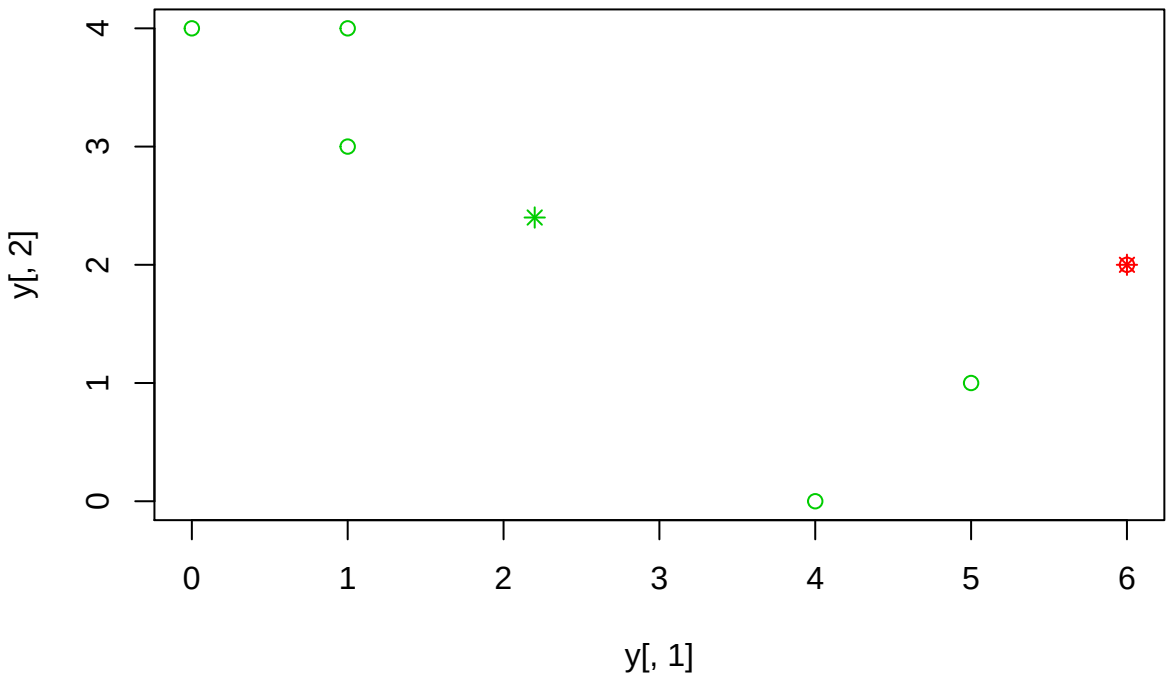
```
## [1] 2 2 2 2 1 2
```

```
#different colored plots
plot(y[,1], y[,2],
     col=(cluster_label + 1))
```



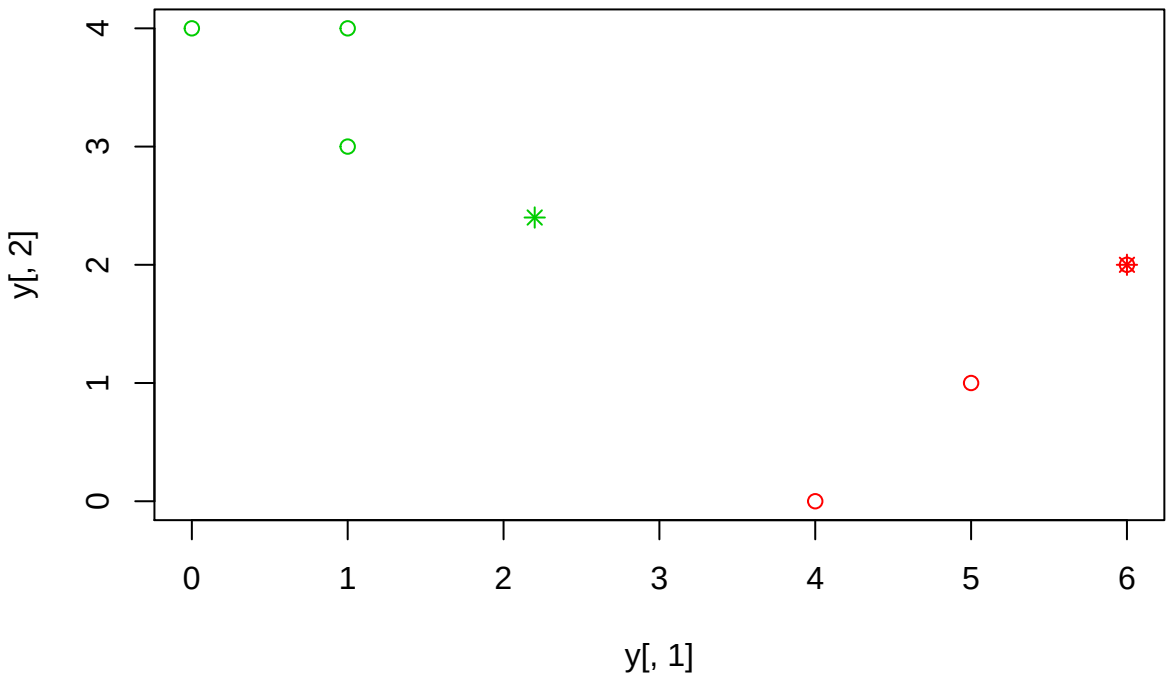
3. (10 points) Compute the centroid for each cluster.

```
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]))

# different colors for each feature
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)
```



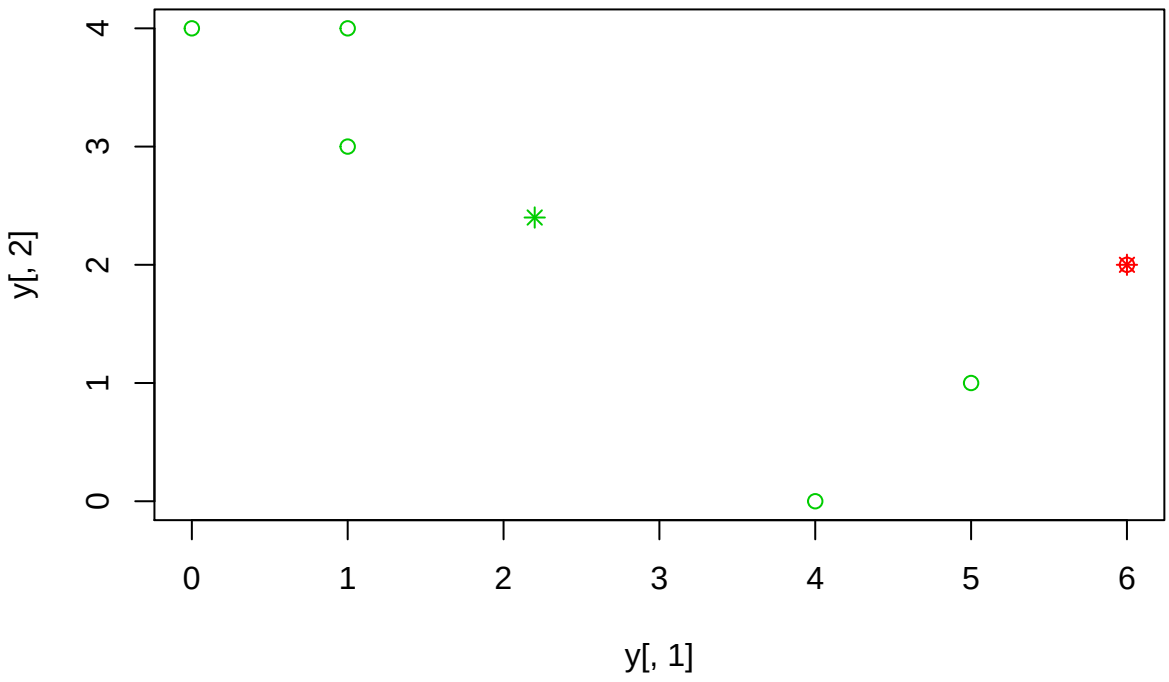
4. (10 points) Assign each observation to the centroid to which it is closest, in terms of Euclidean distance. Report the cluster labels for each observation.

```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)
```

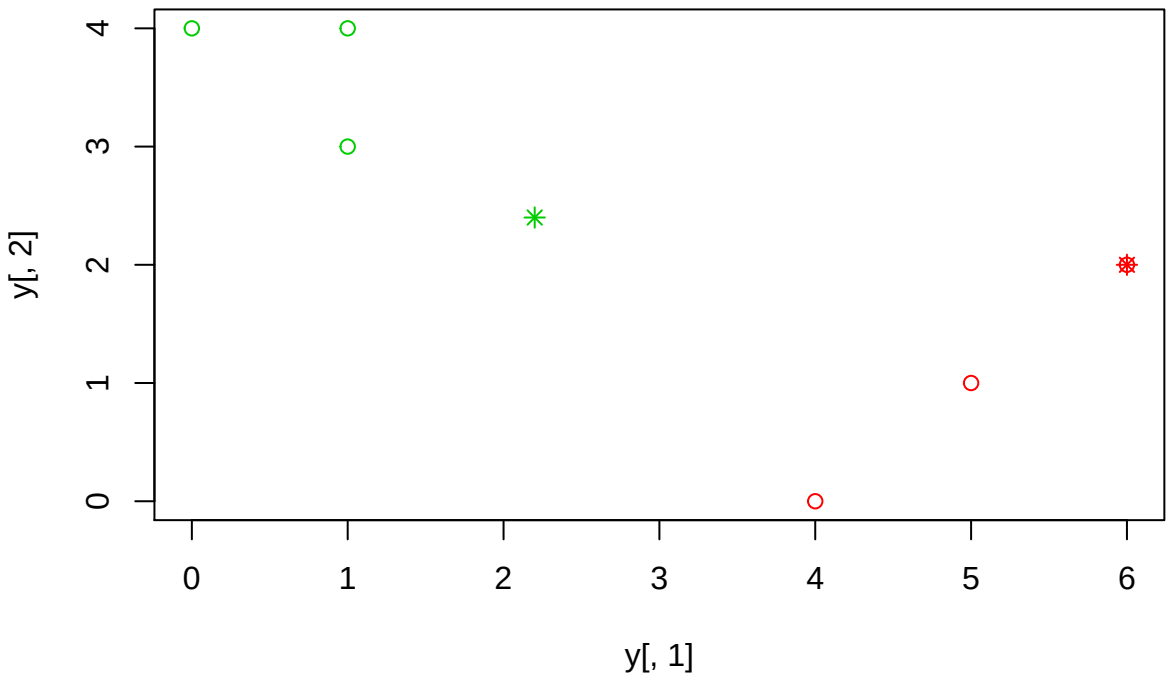


5. (5 points) Repeat (3) and (4) until the answers/clusters stop changing.

```
#Repeated 3 and 4 - answers stopped changing so stopped after first repeat
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]))

#  different colors for each feature
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)
```

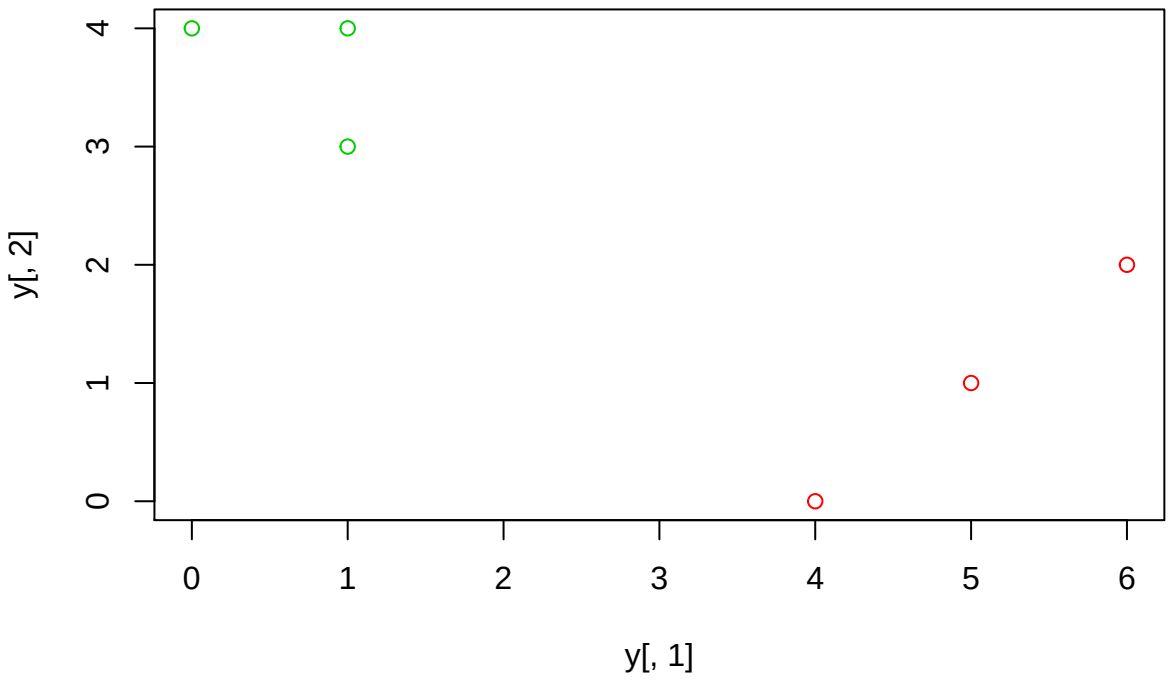



```
###

# 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.

```
# relabeled observations
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.

```
load("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)

```
# a, b, c, d, e

skim(legprof)
```

Table 1: Data summary

Name	legprof
Number of rows	950
Number of columns	11
Column type frequency:	
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	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
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	

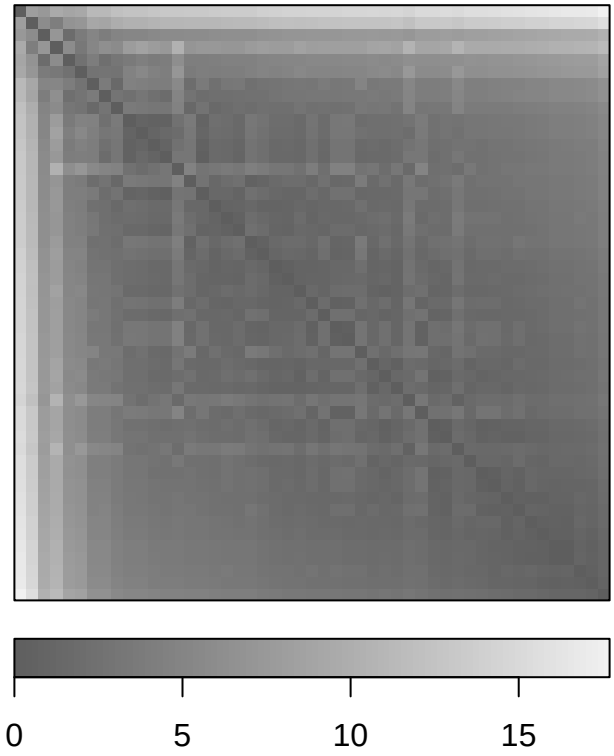
```
legprof_sub <- legprof %>%
  filter(sessid == "2009/10") %>%
 select(state, t_slength, slength, salary_real, expend)

legprof_states <- scale(legprof_sub[, -c(1)]) %>% #standardizing, making them unit-less
  as.tibble() %>%
  na.omit() #omit na

states <- legprof_sub %>%
  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.

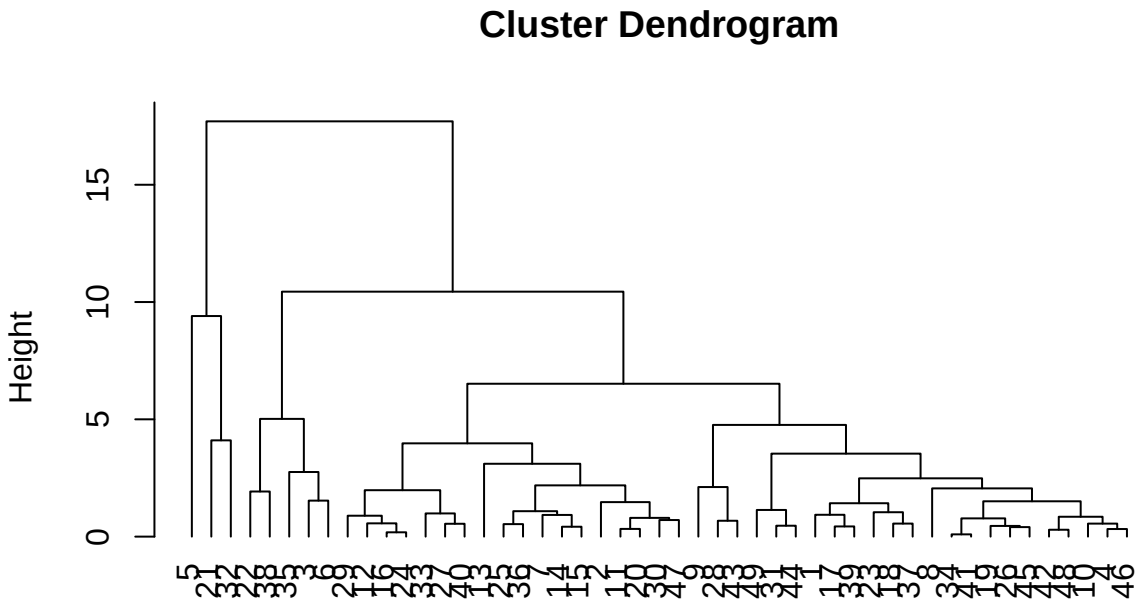
```
legprof_dist <- dist(legprof_states, method = "manhattan")
dissplot(legprof_dist)
```



There is very low likelihood for clusterability given that the boxes in the visual are not clearly defined. The entirety of the visual seems as one gray box, thus there is low likelihood for 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.

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



Given that each of the numbers on the x-axis represent a state, the states that join earlier, cluster earlier, are more similar to each other. For example, states 46 and 4 join very quickly, thus we can assume are more similar. Whereas the distance (or length) of the branch connecting states 5 and 21 is longer, and thus are more dissimilar.

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(1234)

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"
```

```
length(kmeans$cluster)
```

```
## [1] 49
```

```
variable1 <- as.data.frame(kmeans$cluster)
variable2 <- states


cbind(variable1, variable2)
```

```
##    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
```

Based on kmeans, the majority of observations were in one cluster rather than the other. It's hard to say why they're clustering, but it is notable that the smaller cluster has some highly populated, more wealthy, and potentially left-leaning states such as California, New York, Massachusetts.

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(5678) # set seed for iterations to return to same place

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

```
## number of iterations= 12
```

```
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] 0.102041 0.897959
##
## $mu
## $mu[[1]]
## [1] 2.431846 2.156634 1.694880 1.705799
##
## $mu[[2]]
## [1] -0.2763465 -0.2450724 -0.2132464 -0.1960989
##
##
## $sigma
## $sigma[[1]]
##           [,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
##
```

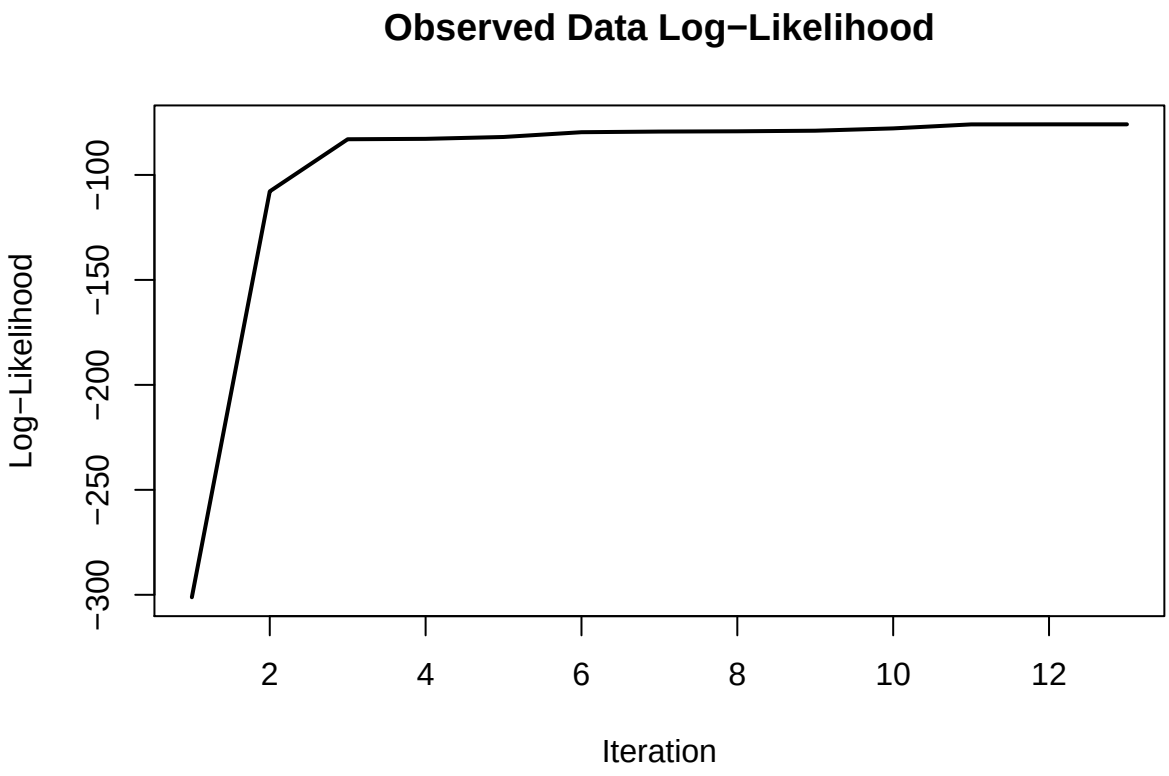
```
## $sigma[[2]]
##            [,1]       [,2]      [,3]       [,4]
## [1,] 0.24528126 0.27954563 0.2100159 0.03137054
## [2,] 0.27954563 0.32491503 0.2379421 0.02504616
## [3,] 0.21001590 0.23794209 0.5825970 0.13823755
## [4,] 0.03137054 0.02504616 0.1382376 0.23965942
##
##
## $loglik
## [1] -75.90016
##
## $posterior
##              comp.1        comp.2
##  [1,]  0.000000e+00  1.000000e+00
##  [2,]  0.000000e+00  1.000000e+00
##  [3,]  1.000000e+00  3.114059e-52
##  [4,]  0.000000e+00  1.000000e+00
##  [5,]  1.000000e+00 1.822861e-100
##  [6,] 3.231763e-261  1.000000e+00
##  [7,] 1.986916e-182  1.000000e+00
##  [8,]  7.562863e-06  9.999924e-01
##  [9,]  0.000000e+00  1.000000e+00
## [10,]  0.000000e+00  1.000000e+00
## [11,]  2.245279e-09  1.000000e+00
## [12,]  0.000000e+00  1.000000e+00
## [13,] 1.314565e-119  1.000000e+00
## [14,] 2.274529e-278  1.000000e+00
## [15,] 2.186647e-284  1.000000e+00
## [16,]  0.000000e+00  1.000000e+00
## [17,]  0.000000e+00  1.000000e+00
## [18,]  0.000000e+00  1.000000e+00
## [19,]  0.000000e+00  1.000000e+00
## [20,]  1.815905e-58  1.000000e+00
## [21,]  9.999999e-01  5.916944e-08
## [22,] 8.784074e-120  1.000000e+00
## [23,] 1.238858e-165  1.000000e+00
## [24,]  0.000000e+00  1.000000e+00
## [25,]  6.260875e-69  1.000000e+00
## [26,]  0.000000e+00  1.000000e+00
## [27,]  0.000000e+00  1.000000e+00
## [28,]  0.000000e+00  1.000000e+00
## [29,]  0.000000e+00  1.000000e+00
## [30,] 2.487705e-113  1.000000e+00
## [31,]  0.000000e+00  1.000000e+00
## [32,]  1.000000e+00  5.823076e-19
## [33,]  0.000000e+00  1.000000e+00
## [34,]  0.000000e+00  1.000000e+00
## [35,]  2.457689e-14  1.000000e+00
## [36,]  1.574447e-58  1.000000e+00
## [37,]  0.000000e+00  1.000000e+00
## [38,]  9.999993e-01  7.098316e-07
## [39,]  0.000000e+00  1.000000e+00
## [40,]  0.000000e+00  1.000000e+00
## [41,]  0.000000e+00  1.000000e+00
```

```
## [42,]  0.000000e+00  1.000000e+00
## [43,]  0.000000e+00  1.000000e+00
## [44,]  0.000000e+00  1.000000e+00
## [45,]  0.000000e+00  1.000000e+00
## [46,]  0.000000e+00  1.000000e+00
## [47,]  2.541446e-46  1.000000e+00
## [48,] 1.194603e-284  1.000000e+00
## [49,]  0.000000e+00  1.000000e+00
##
## $all.loglik
##  [1] -301.16918 -107.82417  -83.00093  -82.80410  -81.92240  -79.64104
##  [7]  -79.33612  -79.21341  -78.93187  -77.82309  -75.92188  -75.90016
## [13]  -75.90016
##
## $restarts
## [1] 0
##
## $ft
## [1] "mvnormalmixEM"
##
## attr(,"class")
## [1] "mixEM"
```

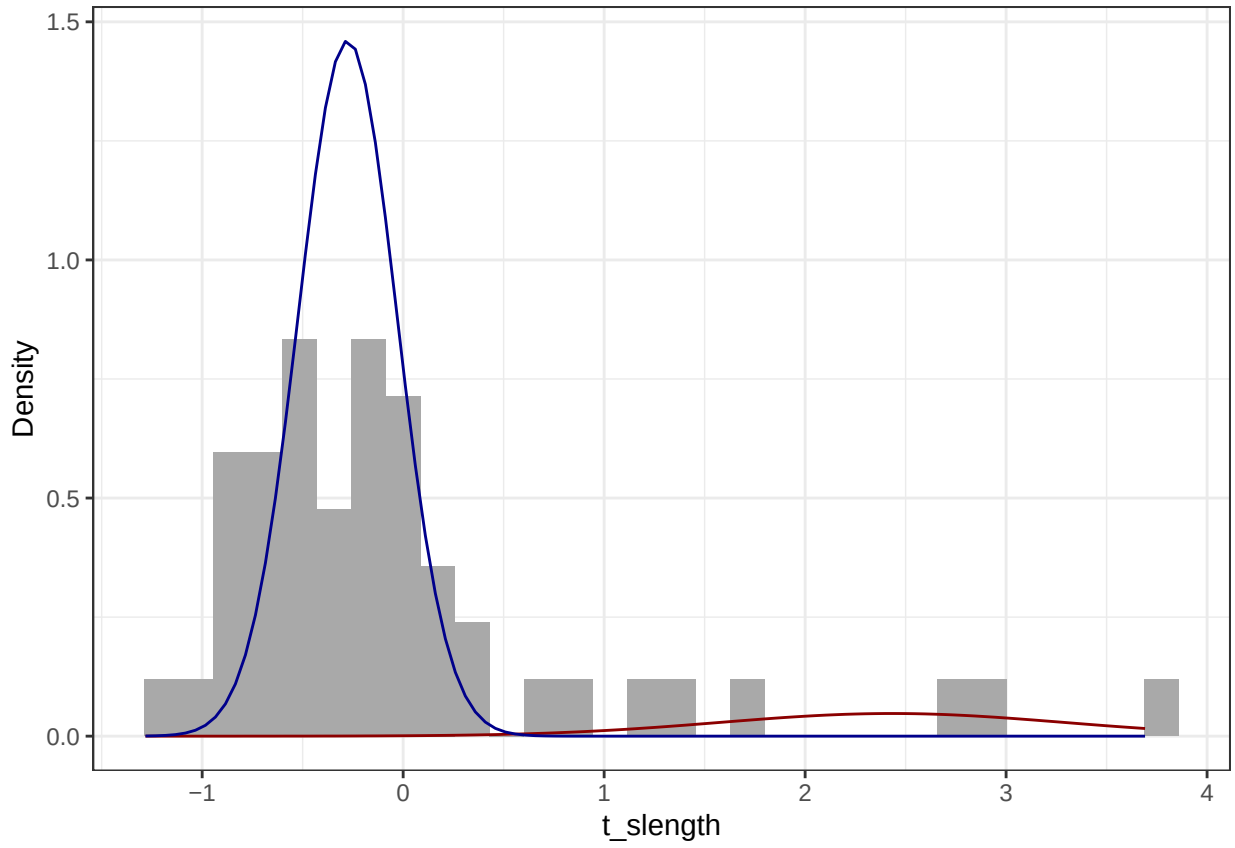
Based on gmm1, we can see that there are two lambda values (.102 and .898). We can say that the larger lambda will yield a higher curve, and substantively, that this cluster has more observations. Additionally, we have 2 mu's, or means, of length 4. One of the clusters has a lower mean for all features, while the other has a higher mean for all features. Finally, based on the sigma, or standard deviation, we can see that one of the clusters had a tighter standard deviation, while the other was more spread out.

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.

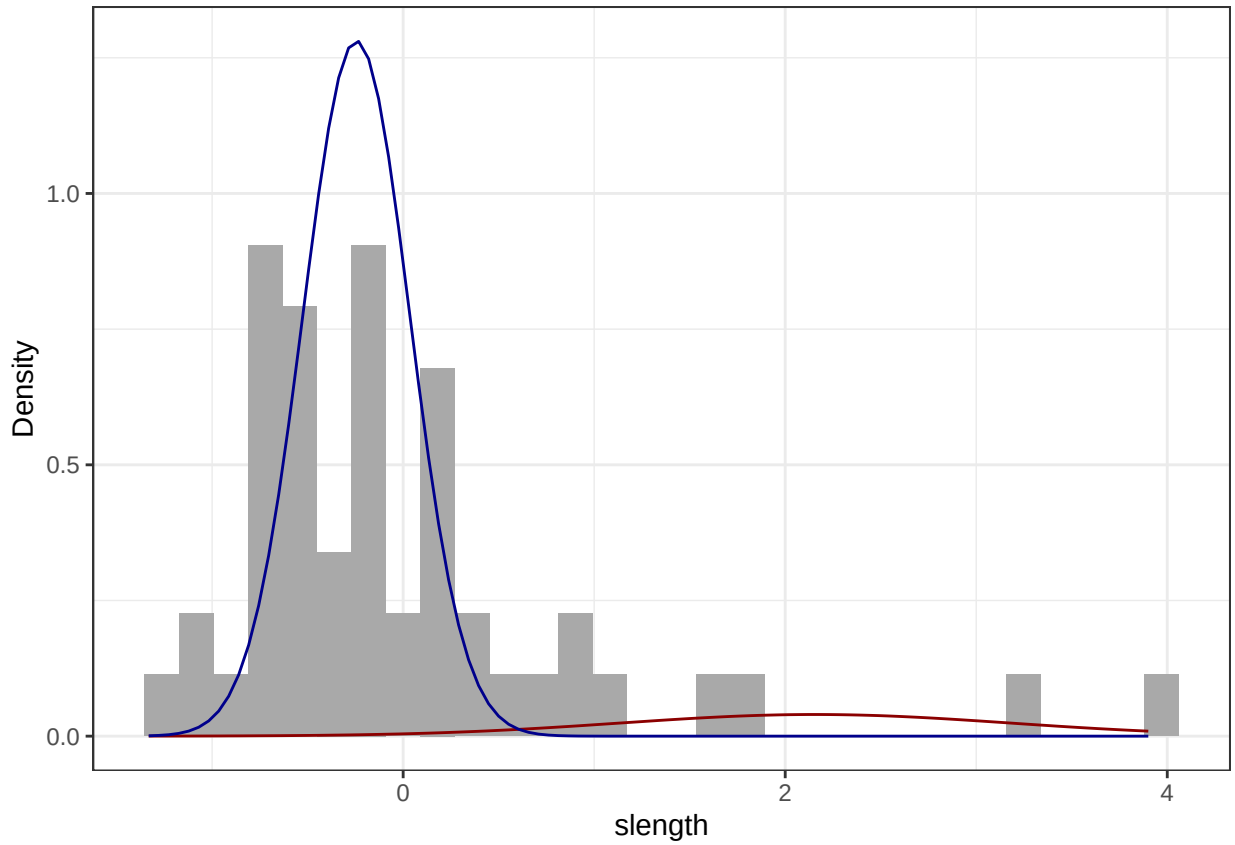
```
plot.mixEM(gmm1)
```



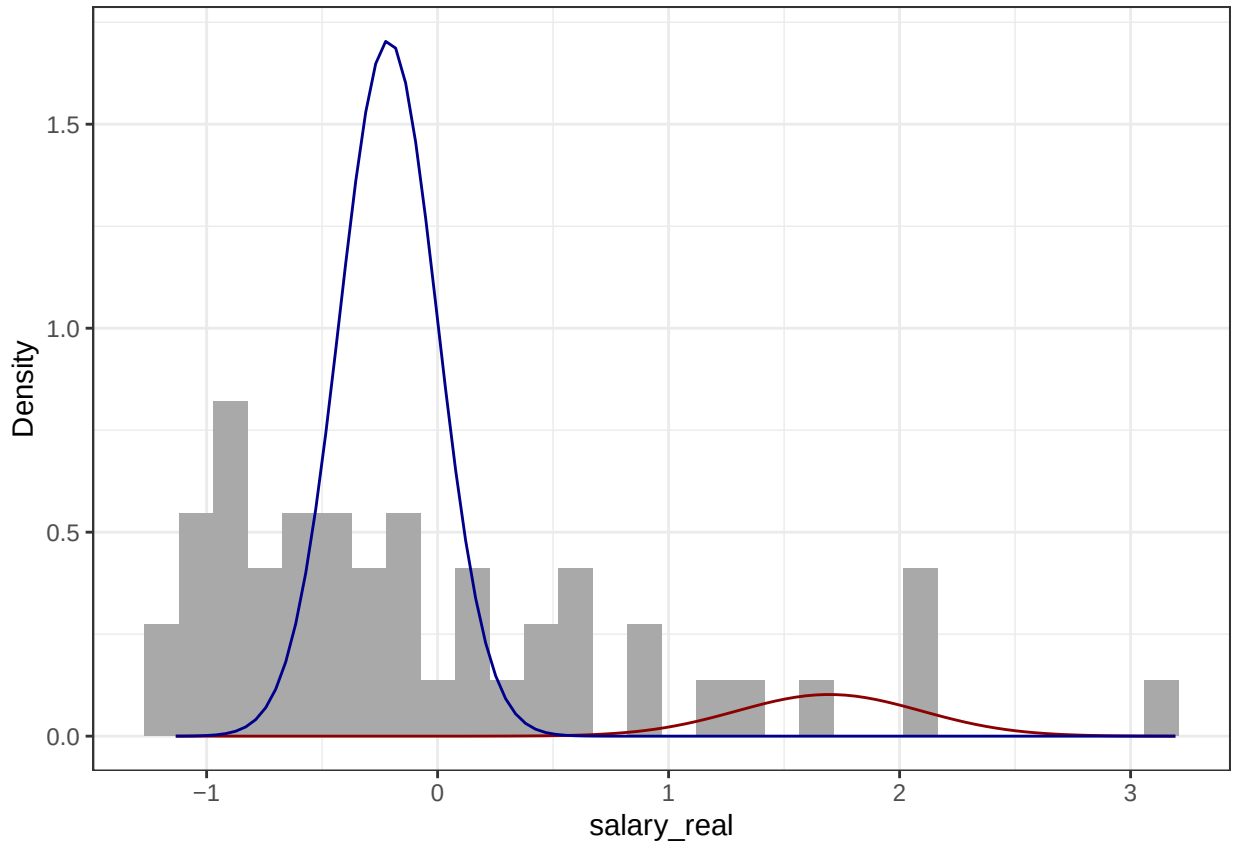

```
ggplot(data.frame(x = gmm1$x[,1])) +
  geom_histogram(aes(x, ..density..), fill = "darkgray") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm1$mu[[1]][1], gmm1$sigma[[1]][1], lam = gmm1$lambda[1]),
                colour = "darkred") +
  stat_function(geom =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



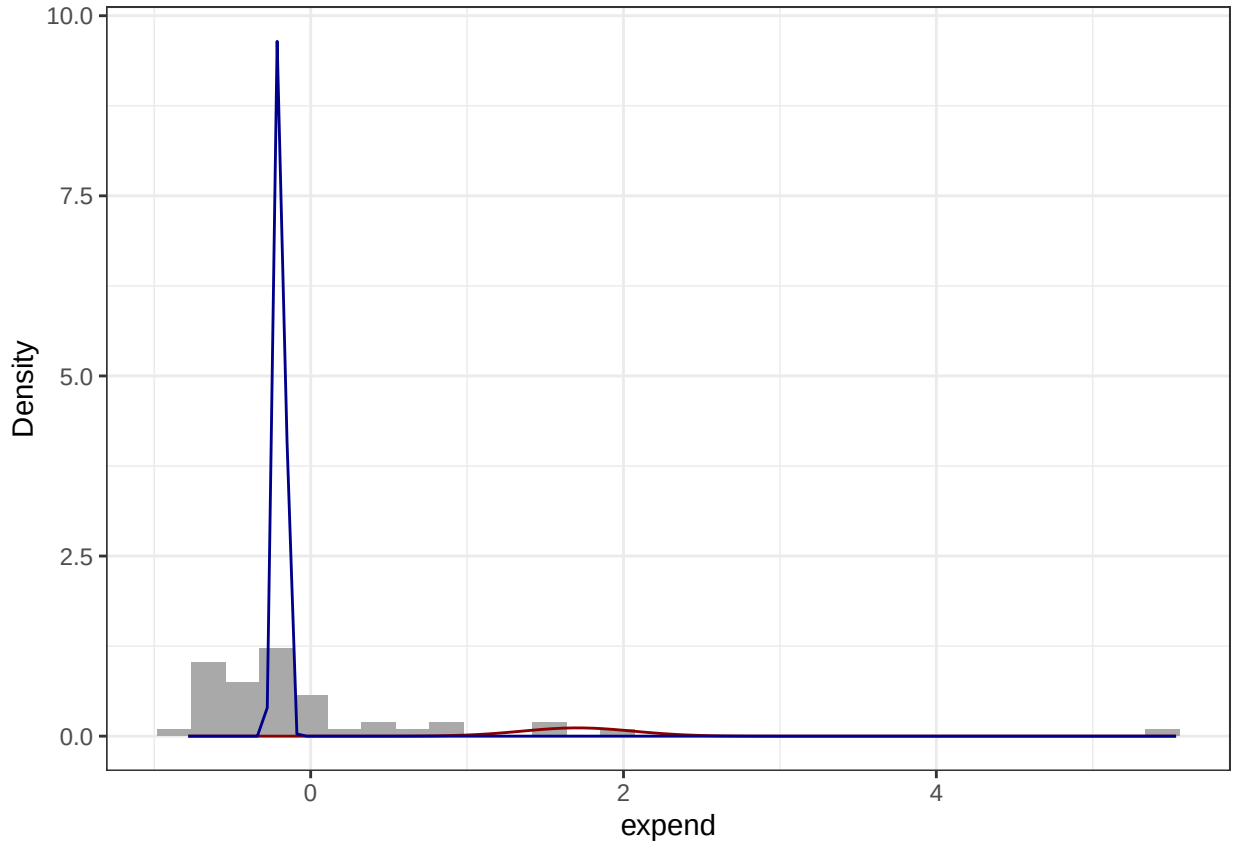

```
ggplot(data.frame(x = gmm1$x[,2])) +
  geom_histogram(aes(x, ..density..), fill = "darkgray") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm1$mu[[1]][2], gmm1$sigma[[1]][2], lam = gmm1$lambda[1]),
                colour = "darkred") +
  stat_function(geom =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




```
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 = "darkred") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm1$mu[[2]][3], gmm1$sigma[[2]][3], lam = gmm1$lambda[2]),
                colour = "darkblue") +
  xlab("salary_real") +
  ylab("Density") +
  theme_bw()
```




```
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()
```



The better the curve fits the data (the histogram) the more it suggests that particular variable is explaining the clustering. This multivariate model suggests that observations cluster based on s_length and t_slength, and that there are some outliers in the data as well. The curve for the variable salary_real fits the data moderately well, and thus it moderately accounts for the clustering. However, the curve does not fit the data for variable 'expend' very well, and thus we can infer it is weakly influencing 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.

```
library(clValid)
legprof_states_mat <- as.matrix(legprof_states)
clvalid <- clValid(legprof_states_mat, 2:20,
                           validation = "internal",
                           clMethods = c("model", "kmeans", "hierarchical"))
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 6
##              Dunn           0.1512  0.0633  0.0224  0.0256  0.0283  0.0543  0.0709  0.1809  0.0971
##              Silhouette     0.6313  0.2589  0.1860  0.0090 -0.0556  0.0919  0.0751  0.2825  0.1899
## kmeans       Connectivity   8.4571 10.9071 16.3996 28.9587 30.9587 37.7306 39.7679 41.1988 45.7417 4
##              Dunn           0.1723  0.2585  0.2552  0.1091  0.1091  0.1107  0.1251  0.1331  0.1392
##              Silhouette     0.6455  0.6127  0.4923  0.3032  0.2849  0.2741  0.3125  0.3308  0.3283
## hierarchical Connectivity   6.0869  6.9536 16.3996 18.7774 20.7774 21.8607 27.6476 35.6813 37.6147 4
##              Dunn           0.3605  0.4358  0.2552  0.2819  0.2819  0.2819  0.2956  0.1578  0.1578
##              Silhouette     0.6992  0.6706  0.4923  0.4433  0.4278  0.3514  0.2569  0.2665  0.2642
##
## Optimal Scores:
##
##              Score  Method       Clusters
## Connectivity 6.0869 hierarchical 2
## Dunn         0.4358 hierarchical 3
## Silhouette   0.6992 hierarchical 2
```

Based on the results, the hierarchical clustering algorithm is the optimal method. This method is most optimal when k = 2, or there are two clusters. However, it also performs well at k= 3.

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?"

These are the three ways to measure how well the clustering went: GMM, kmeans, and hierarchical (HAC). Based on the results, the hierarchical clustering algorithm is the optimal method. This method is most optimal when k = 2, or there are two clusters. However, it also performs well at k= 3. One could select a sub-optimal clustering method if that particular method is more their dataset. For example, GMM gives you more information such as likelihood, or probability, of belinging to one of the clusters than kmeans. Thus one might select GMM over kmeans, even if it is suboptimal. Additionally, though the hierarchical method performed best in this circumstance, one might not use it because it'd more computationally expensive.