PS4

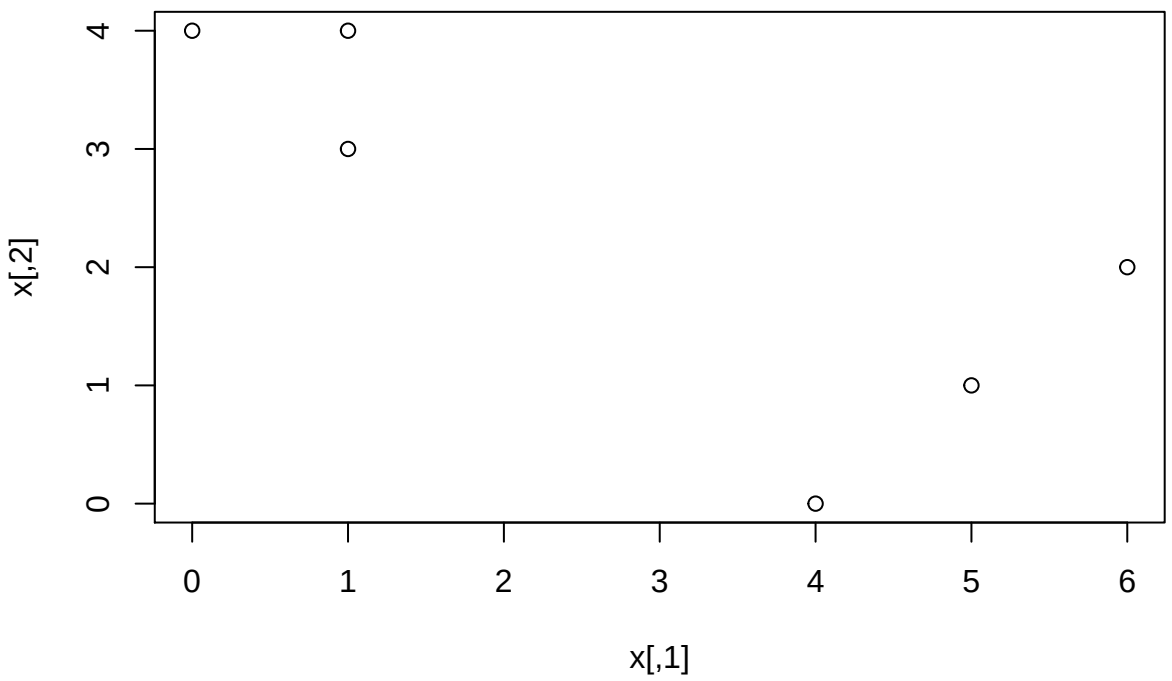
Artemis Yang

2/28/2020

Performing k-Means By Hand

1. (5 points) Plot the observations.

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

```
library(tibble)
set.seed(135)
label <- sample(c(1,2), size = nrow(x), replace = TRUE) #generate a random sequence of 0's and 1's
(cluster = cbind(x, label)) #bind the generated sequence, report label
```

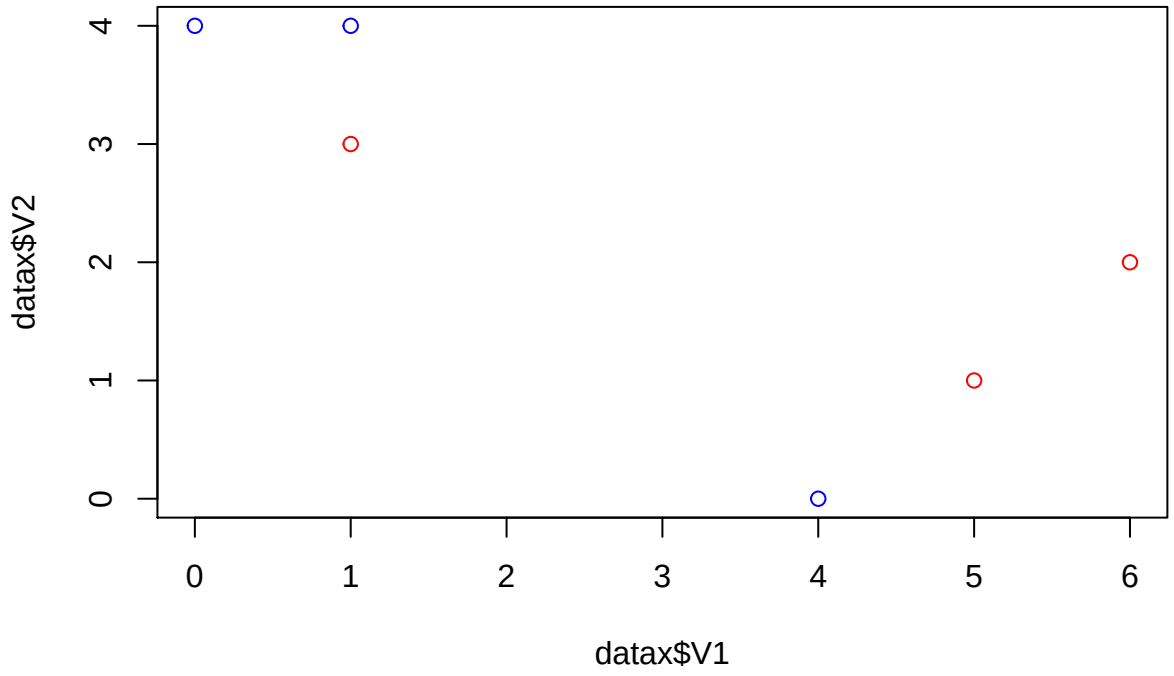
```
##          label
## [1,] 1 4     1
```

```
## [2,] 1 3     2
## [3,] 0 4     1
## [4,] 5 1     2
## [5,] 6 2     2
## [6,] 4 0     1
```

```
datax = as_tibble(cluster) #create a tibble, which is easier to work with
```

```
## Warning: `as_tibble.matrix()` requires a matrix with column names or a `.name_repair` argument. Using
## This warning is displayed once per session.
```

```
plot(datax$V1, datax$V2, col = ifelse(datax$label == 1, 'blue', ifelse(datax$label == 2, 'red','green'))
```



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

```
V1 = aggregate(datax$V1, by=list(datax$label), FUN=mean)[2] # mean of value 1
V2 = aggregate(datax$V2, by=list(datax$label), FUN=mean)[2] # mean of value 2
centroid = cbind(V1, V2, label=c(1,2))
colnames(centroid)[1] <- "V1"
colnames(centroid)[2] <- "V2"
centroid
```

```
##         V1       V2 label
## 1 1.666667 2.666667     1
## 2 4.000000 2.000000     2
```

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.

```
datax <- add_column(datax, D1=0) #a column to store distance between observation and first centroid
datax <- add_column(datax, D2=0) #a column to store distance between observation and second centroid
datax <- add_column(datax, new_label=0) #a column to store updated labels
for(i in 1:nrow(datax)){
  datax$D1[i] <- sqrt((datax$V1[i]-centroid$V1[1])^2 + (datax$V2[i]-centroid$V2[1])^2)
  datax$D2[i] <- sqrt((datax$V1[i]-centroid$V1[2])^2 + (datax$V2[i]-centroid$V2[2])^2)
  #assign new labels according the calculated distances
  if (datax$D1[i] < datax$D2[i]){
    datax$new_label[i] = 1
  }
  else {
    datax$new_label[i] = 2
  }
}
datax
```

```
## # A tibble: 6 x 6
##      V1    V2 label    D1    D2 new_label
##   <dbl> <dbl> <dbl> <dbl> <dbl>     <dbl>
## 1     1     4     1 1.49   3.61         1
## 2     1     3     2 0.745  3.16         1
## 3     0     4     1 2.13   4.47         1
## 4     5     1     2 3.73   1.41         2
## 5     6     2     2 4.38   2            2
## 6     4     0     1 3.54   2            2
```

Cluster labels are stored in "new_label".

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

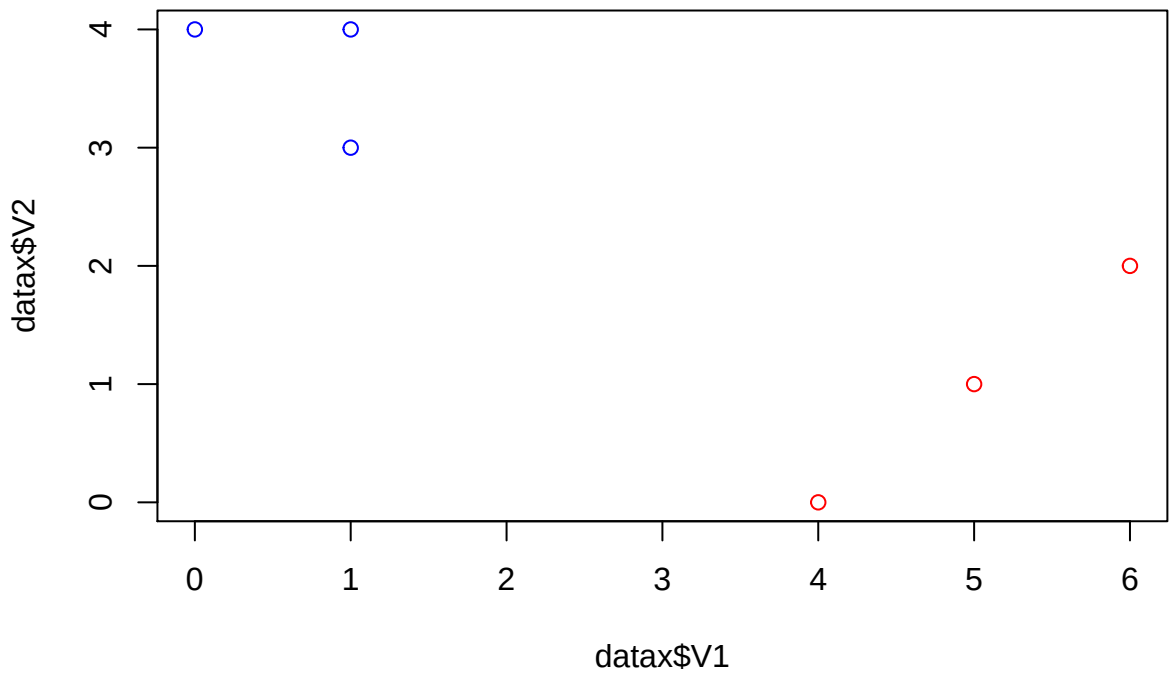
```
j <- 1
repeat {
  #update centroid using updated labels (repeat step 3)
  V1 = aggregate(datax$V1, by=list(datax$label), FUN=mean)[2] # mean of value 1
  V2 = aggregate(datax$V2, by=list(datax$label), FUN=mean)[2] # mean of value 2
  centroid = cbind(V1, V2, label=c(1,2))
  colnames(centroid)[1] <- "V1"
  colnames(centroid)[2] <- "V2"
  centroid
  #update the labels using updated centroid (repeat step 4)
  for(i in 1:nrow(datax)){
    datax$D1[i] <- sqrt((datax$V1[i]-centroid$V1[1])^2 + (datax$V2[i]-centroid$V2[1])^2)
    datax$D2[i] <- sqrt((datax$V1[i]-centroid$V1[2])^2 + (datax$V2[i]-centroid$V2[2])^2)
    #assign new labels according the calculated distances
    if (datax$D1[i] < datax$D2[i]){
      datax$new_label[i] = 1
    }
    else {
      datax$new_label[i] = 2
    }
  }
  #update labels if there's change, otherwise break the loop
```

```
  ifelse (datax$label != datax$new_label, datax$label<-datax$new_label, break)
  j=j+1
}
datax
```

```
## # A tibble: 6 x 6
##      V1    V2 label    D1    D2 new_label
##   <dbl> <dbl> <dbl> <dbl> <dbl>     <dbl>
## 1     1     4     1 1.49   3.61         1
## 2     1     3     1 0.745  3.16         1
## 3     0     4     1 2.13   4.47         1
## 4     5     1     2 3.73   1.41         2
## 5     6     2     2 4.38   2            2
## 6     4     0     2 3.54   2            2
```

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(datax$V1, datax$V2, col = ifelse(datax$label == 1, 'blue', ifelse(datax$label == 2, 'red','green'))
```



Clustering State Legislative Professionalism

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

```
load("/Users/artemisyang/Dropbox/MACS33002 Introduction to Machine Learning/problem-set-4/Data and Code
```

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)

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyverse)
```

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

## v ggplot2 3.2.1     v purrr   0.3.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

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

```
#select observations and features
data <- select(filter(x, sessid=='2009/10'), c(state, t_slength, slength, expend, salary_real)) %>%
        drop_na()
head(data)
```

```
##        state t_slength slength    expend salary_real
## 1    Alabama    116.55  104.55  535.1423    1.050421
## 2     Alaska    128.51  127.80 1493.8351   74.805863
## 3    Arizona    286.13  197.38  631.1329   48.393666
## 4   Arkansas     80.23   80.23  516.6376   30.669025
## 5 California    390.00  270.00 5523.1008  213.405133
## 6   Colorado    205.20  205.20  440.4498   60.492082
```

```
#scale data
rownames(data) <- data[,1] #store state names as rownames
scaled_data <- select(data, -state)  %>%
               scale()
head(scaled_data)
```

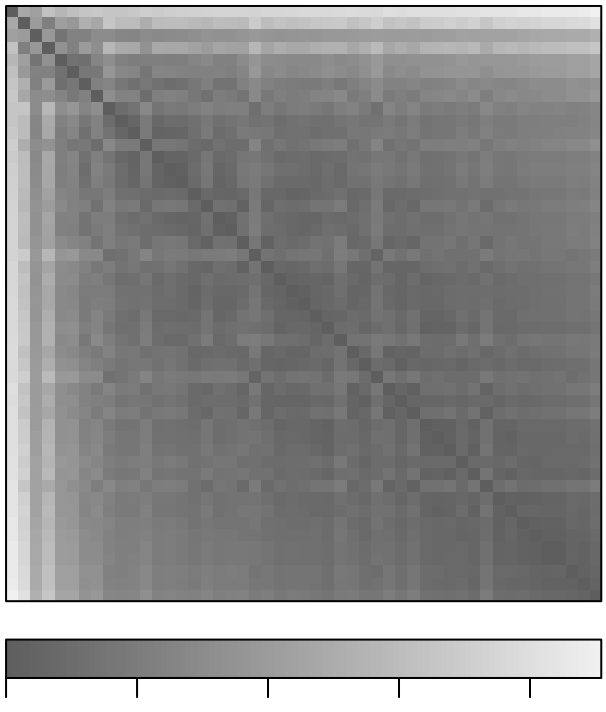
```
##             t_slength    slength     expend salary_real
## Alabama    -0.3716599 -0.4594723 -0.2399910  -1.0920009
## Alaska     -0.2294089 -0.1452309  0.8591198   0.4011333
## Arizona     1.6453067  0.7951955 -0.1299408  -0.1335656
## Arkansas   -0.8036462 -0.7881756 -0.2612061  -0.4923902
## California  2.8807257  1.7767099  5.4785453   3.2069914
## Colorado    0.6827338  0.9008887 -0.3485530   0.1113595
```

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.

```
# Use ODI to diagnose clusterability
library(seriation)
```

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

```
dist <- dist(scaled_data) #dissimilarity matrix
dissplot(dist)
```



From the ODI, we can see there're likely two clusters. One is smaller, represented by the black part on the high-left corner. The other one is bigger, represented by the black area on the low-right part of the image.

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.

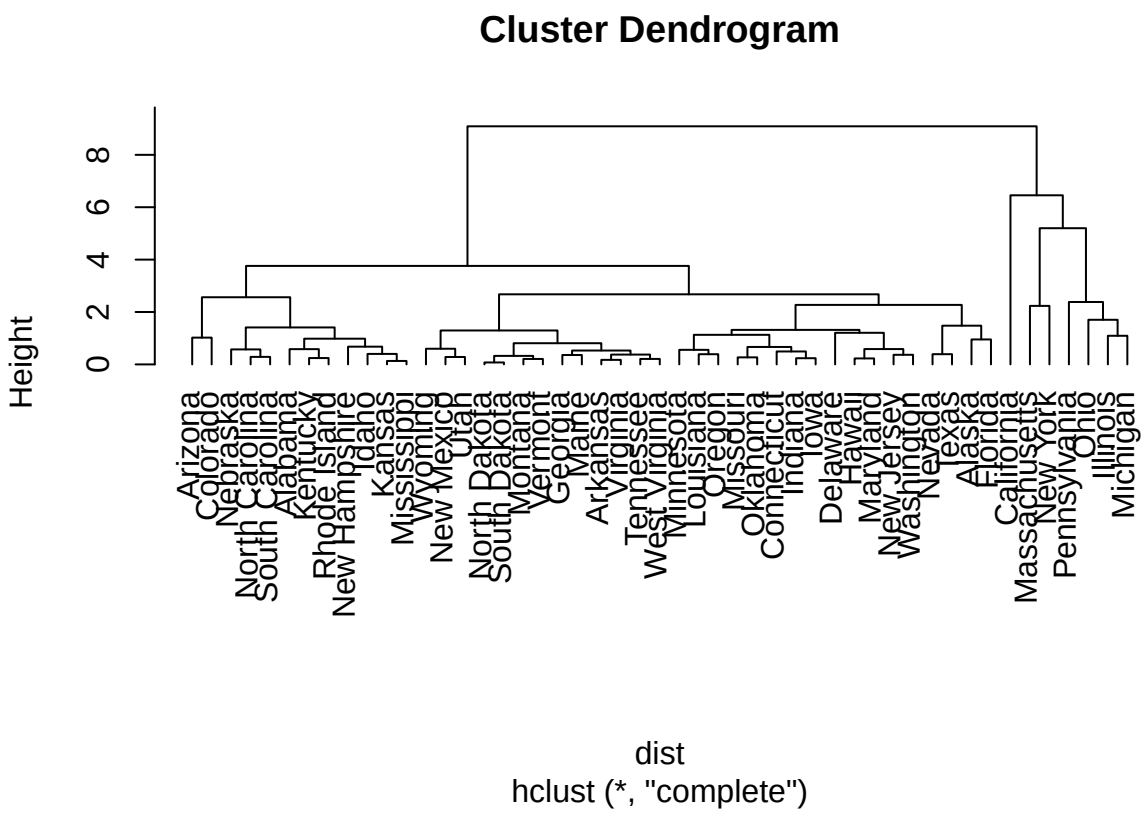
```r
library(dendextend)
```

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

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

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

```r
set.seed(1234)
hc_complete <- hclust(dist,
                      method = "complete")
#plot dendrogram
plot(hc_complete, hang = -1)
```



```
data <- add_column(data, hc_assignment=1)
#assign labels based on dendrogram
clist <- c("California", "Massachusetts", "New York", "Pennsylvania", "Ohio", "Illinois", "Michigan")
for (i in 1:nrow(data)) {
  for (s in clist) {
    if (data$state[i]==s) {
       data$hc_assignment[i]=2
    }
  }
}
```

The dendropgram shows that 7 states are clustered together and the other 42 are clustered together. This is consistent with the result from part 2.3, which indicates there's a big cluster and a small cluster.

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(scaled_data,
                 centers = 2,
                 nstart = 15)
str(kmeans)
```

```
## List of 9
##  $ cluster     : Named int [1:49] 1 1 1 1 2 1 1 1 1 1 ...
##   ..- attr(*, "names")= chr [1:49] "Alabama" "Alaska" "Arizona" "Arkansas" ...
##  $ centers     : num [1:2, 1:4] -0.293 2.1 -0.293 2.101 -0.205 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "1" "2"
##   .. ..$ : chr [1:4] "t_slength" "slength" "expend" "salary_real"
##  $ totss       : num 192
##  $ withinss    : num [1:2] 48.4 40.4
##  $ tot.withinss: num 88.7
##  $ betweenss   : num 103
##  $ size        : int [1:2] 43 6
##  $ iter        : int 1
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"
```

```
#Display assignments in table
t <- as.table(kmeans$cluster)
t <- data.frame(t)
colnames(t)[colnames(t)=="Freq"] <- "km_assignment"
colnames(t)[colnames(t)=="Var1"] <- "state"
t
```

```
##             state km_assignment
## 1         Alabama             1
## 2          Alaska             1
## 3         Arizona             1
## 4        Arkansas             1
```

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

The table shows that 6 states are clustered together and the other 43 are clustered together.

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
library(mixtools)
```

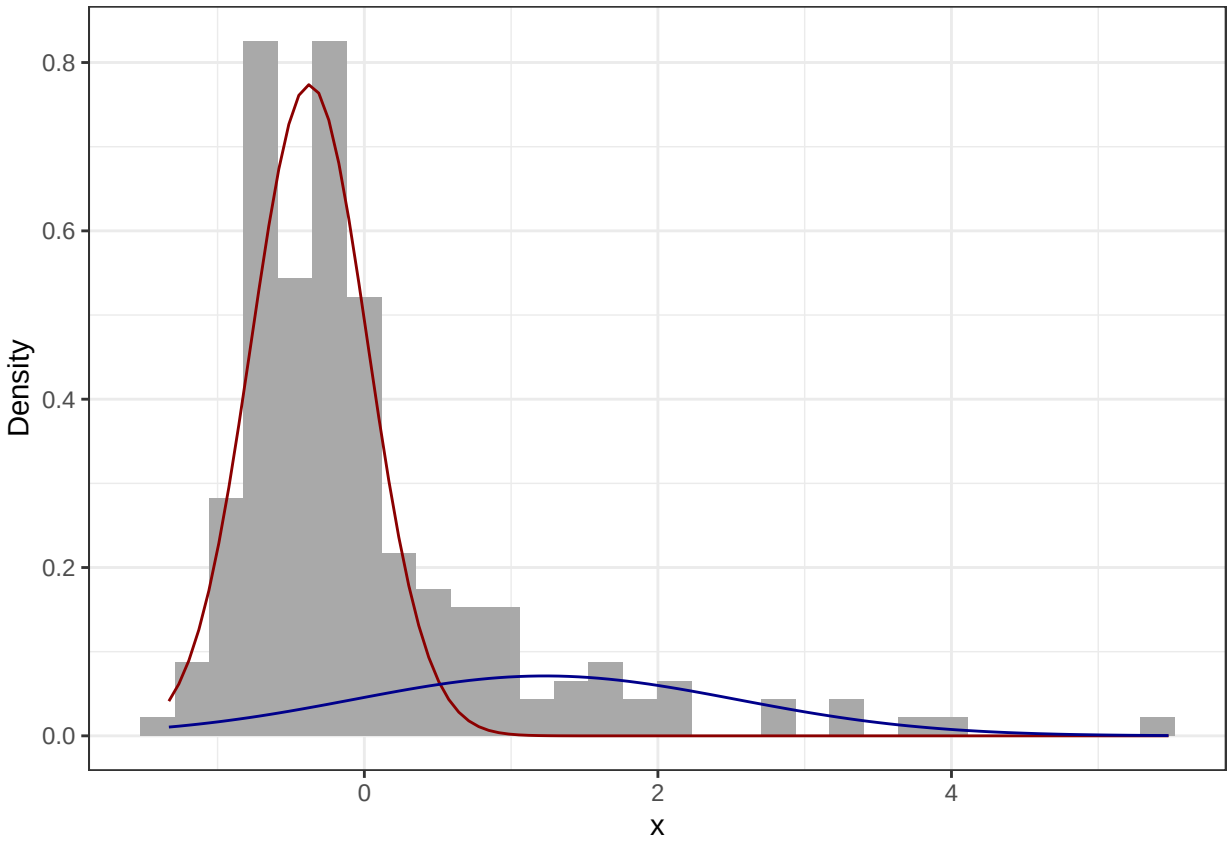
```
## mixtools package, version 1.2.0, Released 2020-02-05
## This package is based upon work supported by the National Science Foundation under Grant No. SES-0518
```

```r
library(plotGMM)
library(ggplot2)
set.seed(1234)
# fit the GMM using EM and 2 comps
gmm <- normalmixEM(scaled_data, k = 2)
```

```
## number of iterations= 41
```

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

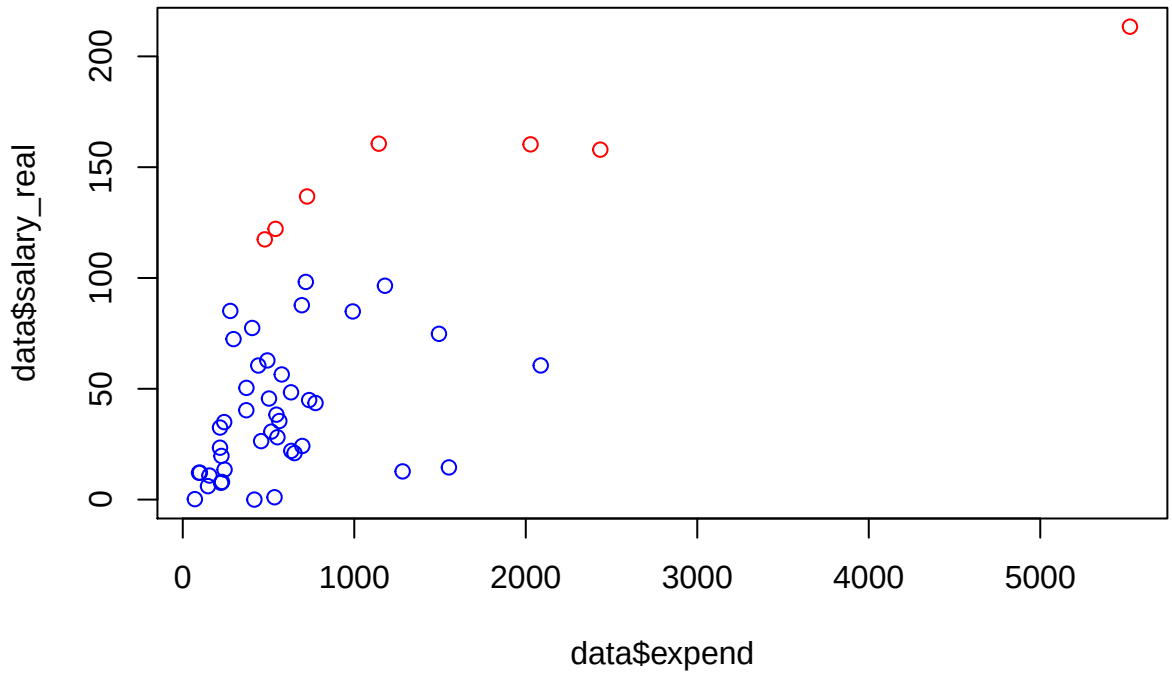
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



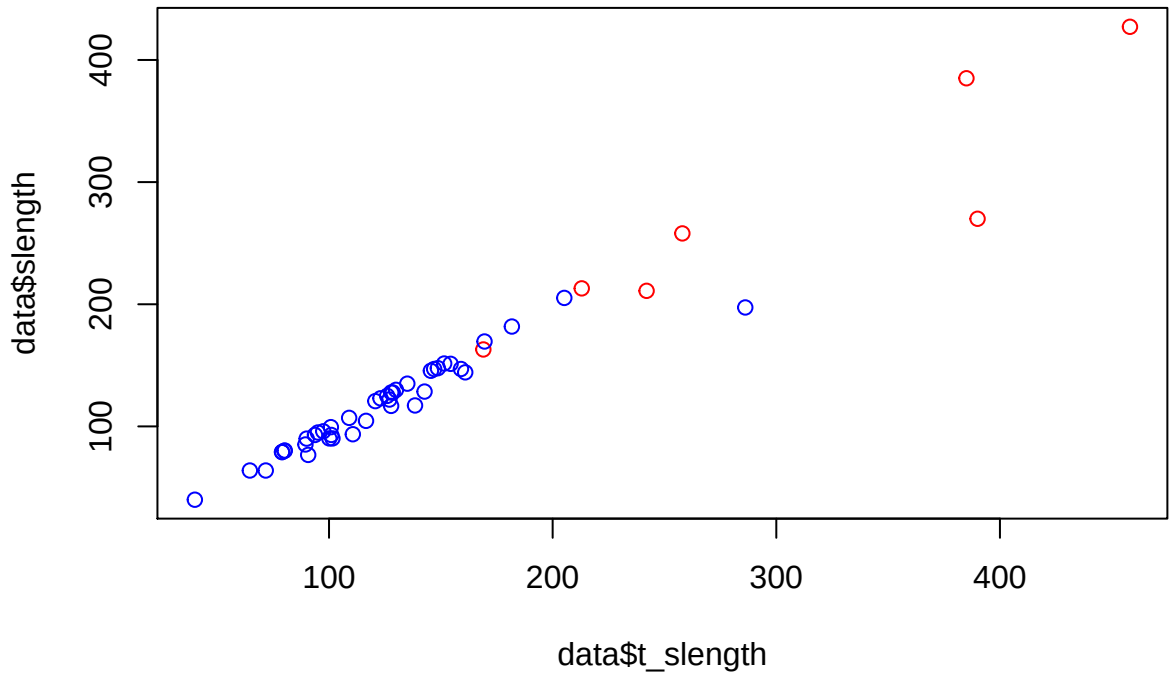
The GMM density plot shows that one component (red) has a high density while the other component has a low density (blue), which implies that one component has a lot more observations than the other one. There's also overlap between the two densities, which suggests that the current GMM doesn't fit with the observations very well, so more work is expected in order to make a better fit.

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.

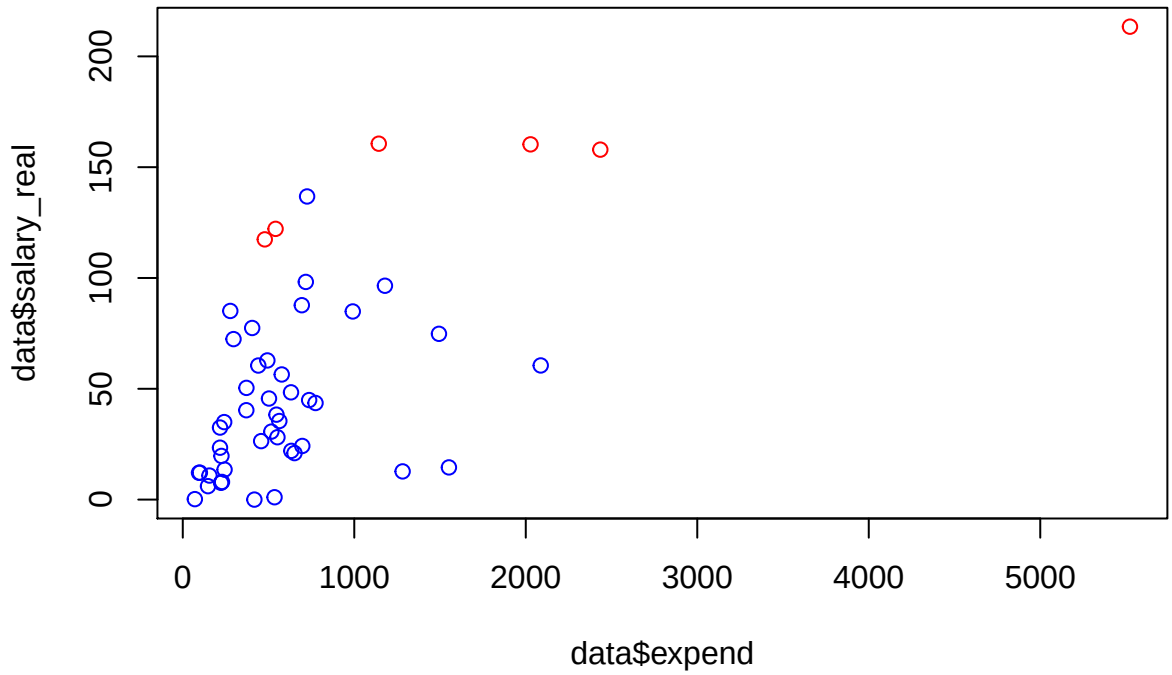
```
#combine data with k-means assignments for plotting
data <- merge(data,t,by="state") #combine data with k-means assignments
#plot across salary and expenditures for HAC assignments
plot(data$expend, data$salary_real,
    col = ifelse(data$hc_assignment == 1,
    'blue', ifelse(data$hc_assignment == 2, 'red','green')))
```



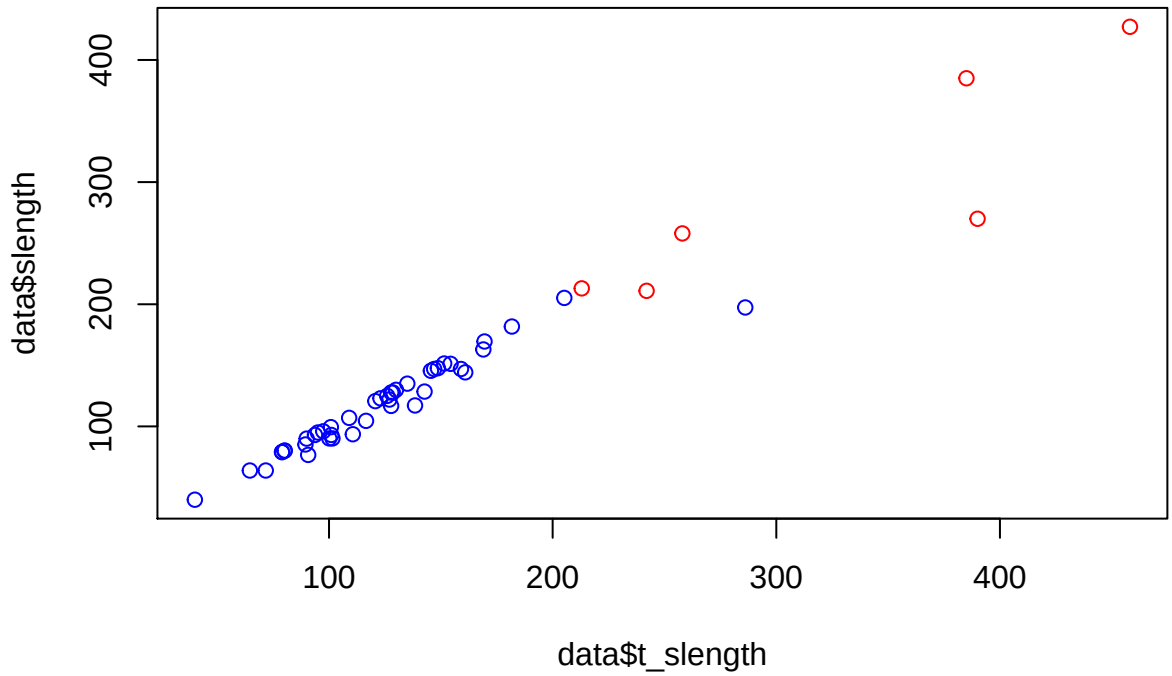

```
#plot across Length of regular session and Total length of session for HAC assignments
plot(data$t_slength, data$slength,
    col = ifelse(data$hc_assignment == 1,
    'blue', ifelse(data$hc_assignment == 2, 'red','green')))
```




```
#plot across salary and expenditures for k-means assignments
plot(data$expend, data$salary_real,
    col = ifelse(data$km_assignment == 1,
    'blue', ifelse(data$km_assignment == 2, 'red','green')))
```




```r
#plot across Length of regular session and Total length of session for k-means assignments
plot(data$t_slength, data$slength,
    col = ifelse(data$km_assignment == 1,
    'blue', ifelse(data$km_assignment == 2, 'red','green')))
```



We have shown the dendrogram of HAC and the density plot of GMM from previous questions. Both graphs show that there's one small cluster and one bigger cluster. In addition, from the two-way scatterplots above, we can see that k-means and HAC achieve basically the same results. The only difference is the cluster for Illinois. It looks like HAC fits Illinois better in terms of expenditure and real salary, but k-means fits Illinois better in terms of total session length and regular session length.

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
```

```
cl_validation <- clValid(scaled_data, 2:5, validation = "internal",
                         clMethods = c("hierarchical", "kmeans", "model"))
```

```
## 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
```

```
summary(cl_validation)
```

```
##
## Clustering Methods:
##  hierarchical kmeans model
##
## Cluster sizes:
##  2 3 4 5
##
## Validation Measures:
##                                  2       3       4       5
##
## hierarchical Connectivity   6.0869  6.9536 16.1885 18.6774
##              Dunn           0.3637  0.4371  0.2562  0.2836
##              Silhouette     0.6994  0.6711  0.4932  0.4440
## kmeans       Connectivity   8.4460 10.8960 16.1885 28.7437
##              Dunn           0.1735  0.2581  0.2562  0.1090
##              Silhouette     0.6458  0.6131  0.4932  0.3042
## model        Connectivity  10.7393 28.6119 39.0687 67.8401
##              Dunn           0.1522  0.0633  0.0225  0.0258
##              Silhouette     0.6314  0.2588  0.1861  0.0085
##
```

```
## Optimal Scores:
##
##              Score  Method       Clusters
## Connectivity 6.0869 hierarchical 2
## Dunn         0.4371 hierarchical 3
## Silhouette   0.6994 hierarchical 2
```

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

- What can you take away from the fit?

From part 2.7, it seems that hierarchical and k-means generate very similar results and it's hard to tell which one is better, but the validation score tells us that hierarchical method achieves the best fit, which offers a more accurate and formal evalution of the performance of different algorithms.

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

The validation process suggests that HAC performs better than the other two algorithms. The optimal value of k is 2 in terms of Connectivity and Silhouette width.

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

If the results are similar, a "sub-optimal" method could be selected because of its interpretability or other characteristics. For me, the k-means method's mechanism is very easy to understand, so I may prefer k-means when its performance is not too far off compared with other methods, even if the validation statistics suggest another method would be optimal.