PS4

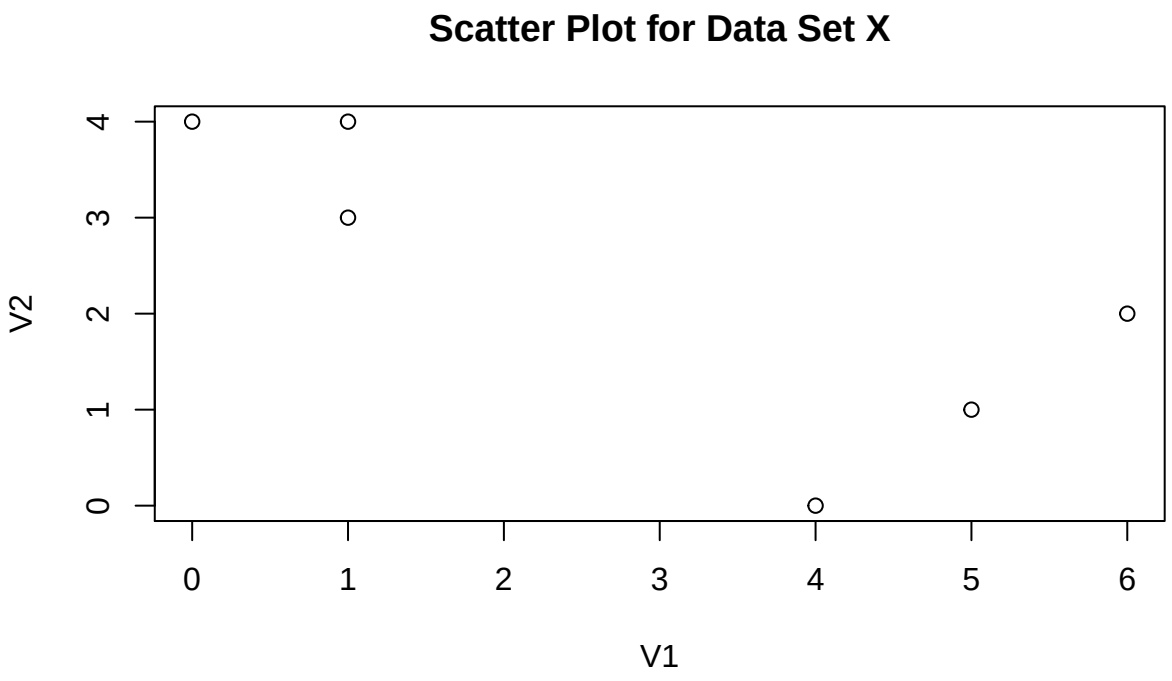
Joanna Zhang

3/02/2020

Performing k-Means By Hand

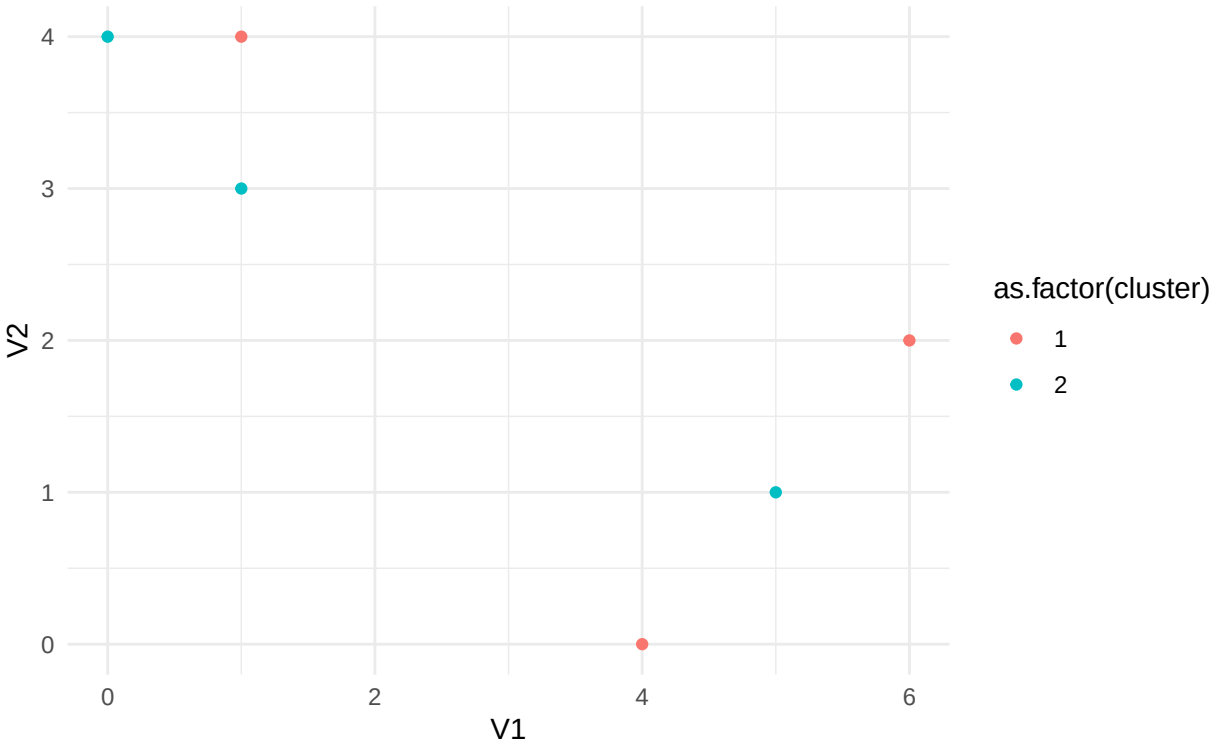
1. Plot the observations.

```
x <- cbind(c(1, 1, 0, 5, 6, 4), c(4, 3, 4, 1, 2, 0))
plot(x, main = "Scatter Plot for Data Set X", xlab = "V1", ylab = "V2")
```



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(1405)
df <- as.data.frame(x) %>%
  mutate(cluster = sample(2, nrow(x), replace =T))
df %>%
  ggplot()+
  geom_point(aes(V1,V2, color = as.factor(cluster)))+
  theme_minimal()
```



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

```
centroids <- df %>%
  group_by(cluster) %>%
  summarize(mean_v1 = mean(V1), mean_v2 = mean(V2))
knitr::kable(centroids)
```

cluster	mean_v1	mean_v2
1	3.666667	2.000000
2	2.000000	2.666667

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.

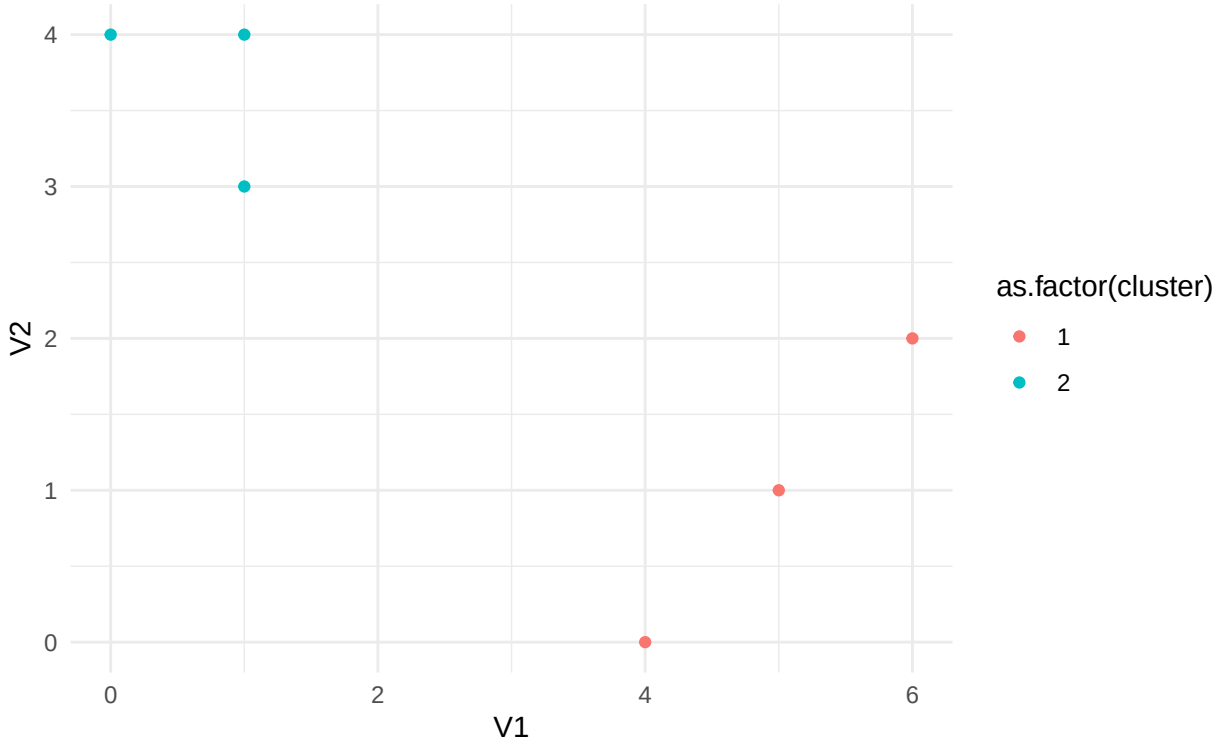
```
get_centroids <- function(data){
  curr_centroids <- data %>%
    group_by(cluster) %>%
    summarize(mean_v1 = mean(V1), mean_v2 = mean(V2))
  return(curr_centroids)
}

find_new_clstr <- function(data){
  curr_centroids <- get_centroids(data)
  newlabel <- vector()
  for (i in 1:nrow(data)){
  a <- data[[1]][i]
```

```
  b <- data[[1]][i]
  dist1 <- sqrt((a - curr_centroids[[2]][1])^2 + (b - curr_centroids[[3]][1])^2)
  dist2 <- sqrt((a - curr_centroids[[2]][2])^2 + (b - curr_centroids[[3]][2])^2)
  newlabel <- append(newlabel,ifelse(dist1 < dist2, 1, 2))
}
data <- data %>%
  mutate(cluster = newlabel)  %>%
  select(V1, V2, cluster)
return(data)
}
df<-find_new_clstr(df)
knitr::kable(df)
```

V1	V2	cluster
1	4	2
1	3	2
0	4	2
5	1	1
6	2	1
4	0	1

```
df %>%
  ggplot()+
  geom_point(aes(V1,V2, color = as.factor(cluster)))+
  theme_minimal()
```

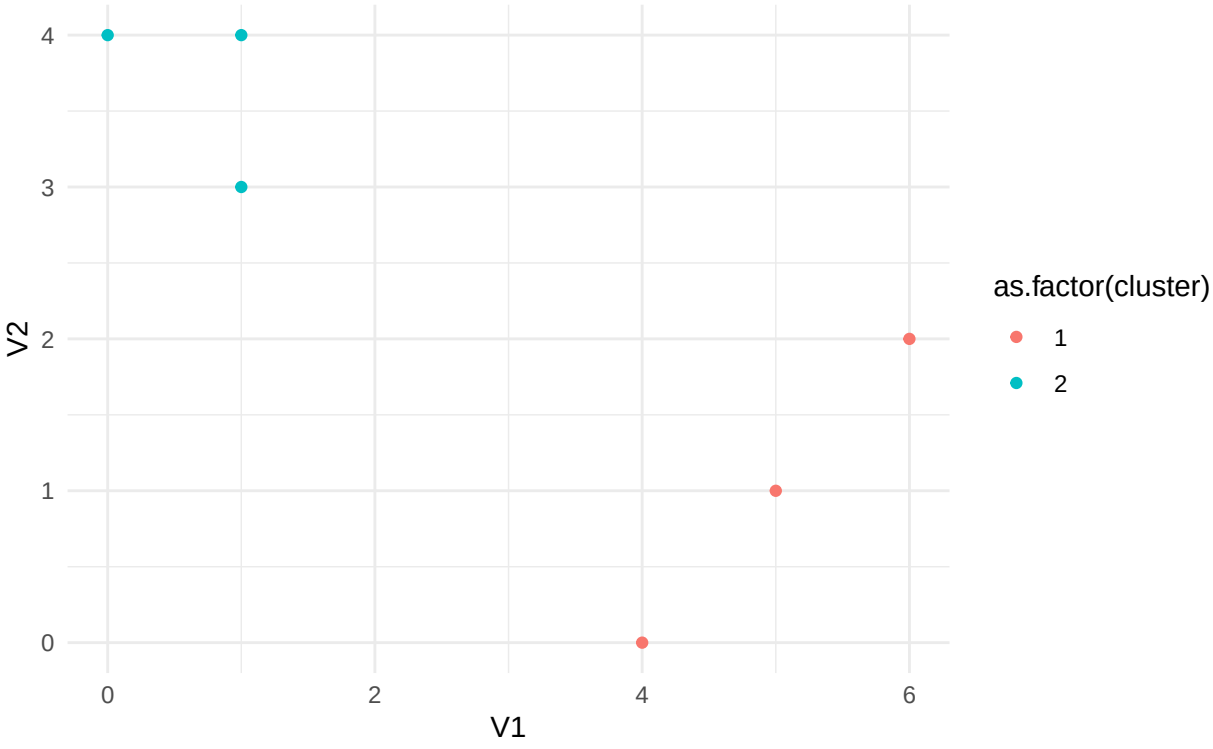


5. (5 points) Repeat (3) and (4) until the answers/clusters stop changing.

```
converge = F
while(!converge){
  current_clstr <- df$cluster
  df<-find_new_clstr(df)
  converge = all(current_clstr == df$cluster)
}
```

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.

```
df %>%
  ggplot()+
  geom_point(aes(V1,V2, color = as.factor(cluster)))+
  theme_minimal()
```



Clustering State Legislative Professionalism

1.

```
load("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)

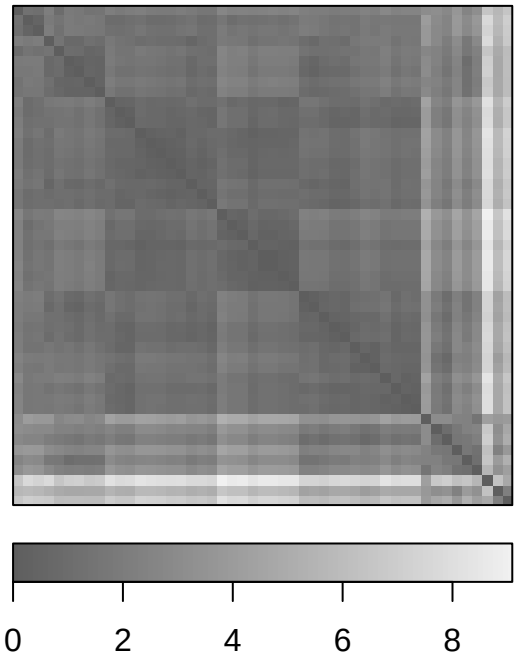
```
lp <- legprof %>%
  filter(sessid == "2009/10") %>%
  #set observation id using state abv
  mutate(rowname = stateabv) %>%
  column_to_rownames() %>%
  select(t_slength, slength, salary_real, expend) %>%
  drop_na() %>%
  scale()

states <- legprof %>%
  filter(sessid == "2009/10") %>%
  drop_na() %>%
  select(stateabv)
```

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.

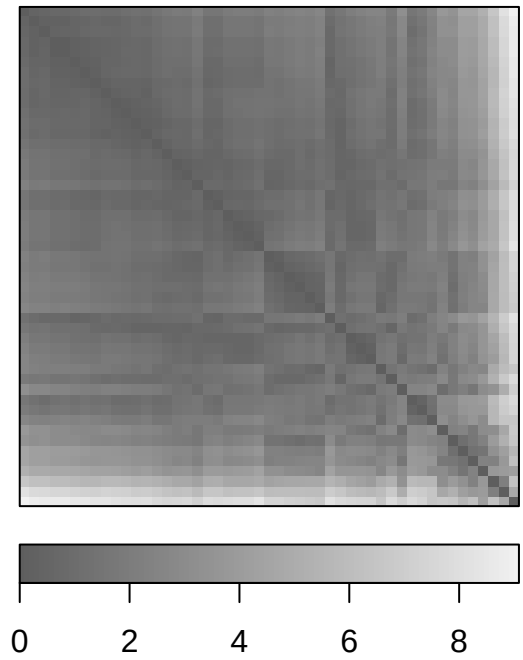
```
distance <- dist(lp, method = "euclidean")
dissplot(distance, method = "HC_average",
         options = list(main = "Dissimilarity Plot using hierarchical clustering"))
```

Dissimilarity Plot using hierarchical clustering




```
dissplot(distance, method = "VAT",
         options = list(main = "Dissimilarity Plot using ODI"))
```

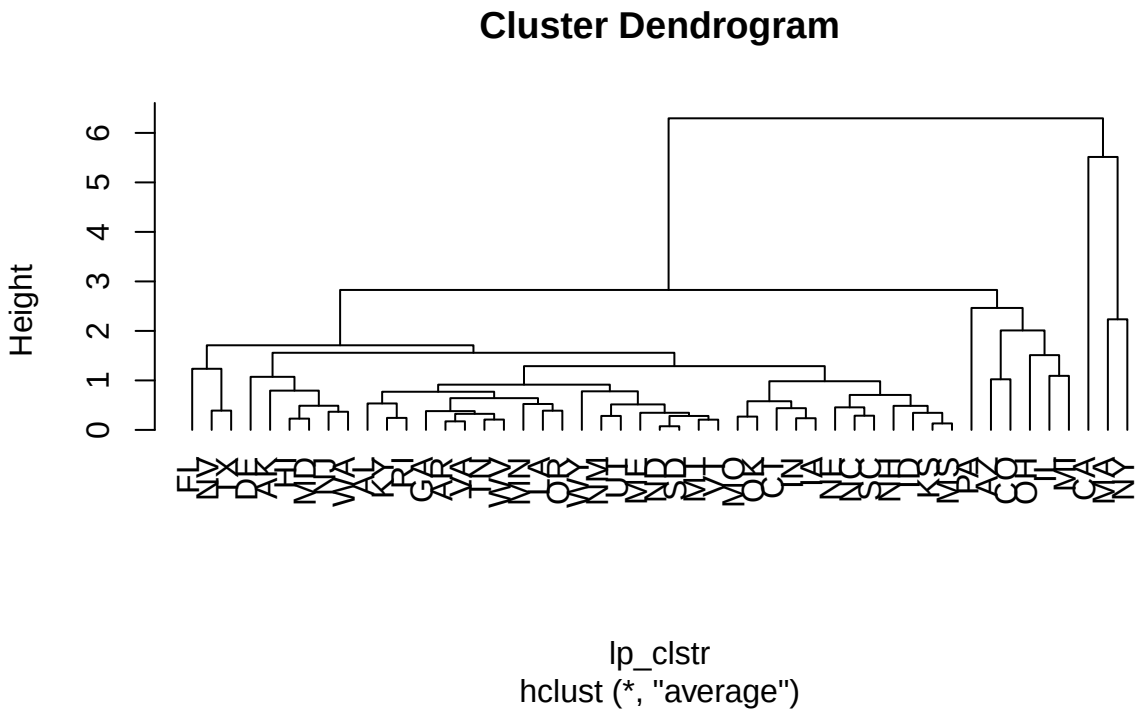
Dissimilarity Plot using ODI



The two pictures above show the dissimilarity plots generated using hierarchical clusting and ODI. From the pictures, there seems to be blocks showing the clustering and stratification of the observations. This implies that there could be a non-random structure in the observations.

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.

```
lp_clstr <- lp %>%
  scale() %>%
  dist()
hc_avg <- hclust(lp_clstr,
                   method = "average"); plot(hc_avg, hang = -1)
```




```
cuts <- cutree(hc_avg,  k = c(2,3))
table(`2 Clusters` = cuts[,1],
      `3 Clusters` = cuts[,2])
```

```
##            3 Clusters
## 2 Clusters  1  2  3
##          1 46  0  0
##          2  0  1  2
```

From the dengrogram we see that there are roughly two clusters. The cluster on the left is large, with over 40 states in it. Whereas the other one is pretty small, with only three states. Since we are not so sure whether 3 clusters can do better than two clusters, we cut and compare the trees. The matrix indicating the assignments shows that if we use 3 clusters in stead of 2 clusters, only two of the observations will move the the third cluster. For simplicity, 2 clusters are sufficient in capturing the general structure in the data.

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(123)
kmeans <- kmeans(lp, centers = 2)

kmeans_results <- as.table(kmeans$cluster)
(kmeans_results <- data.frame(kmeans_results))
```

```
##    Var1 Freq
## 1    AL    1
## 2    AK    1
## 3    AZ    1
## 4    AR    1
## 5    CA    2
## 6    CO    1
## 7    CT    1
## 8    DE    1
## 9    FL    1
## 10   GA    1
## 11   HI    1
## 12   ID    1
## 13   IL    1
## 14   IN    1
## 15   IA    1
## 16   KS    1
## 17   KY    1
## 18   LA    1
## 19   ME    1
## 20   MD    1
## 21   MA    2
## 22   MI    2
## 23   MN    1
## 24   MS    1
## 25   MO    1
## 26   MT    1
## 27   NE    1
## 28   NV    1
## 29   NH    1
## 30   NJ    1
## 31   NM    1
## 32   NY    2
## 33   NC    1
## 34   ND    1
## 35   OH    2
## 36   OK    1
## 37   OR    1
## 38   PA    2
## 39   RI    1
## 40   SC    1
## 41   SD    1
## 42   TN    1
## 43   TX    1
## 44   UT    1
## 45   VT    1
## 46   VA    1
## 47   WA    1
```

```
## 48   WV    1
## 49   WY    1
```

```
colnames(kmeans_results)[colnames(kmeans_results)=="Freq"] <- "Assignment"

kmeans_results %>%
  count(Assignment)
```

```
## # A tibble: 2 x 2
##   Assignment     n
##        <int> <int>
## 1          1    43
## 2          2     6
```

The k-means algorithm assigns 6 observations to the first group and 43 observations to the second group. Still, similar the groups generated by hierarchical clustering, one group is large with over 40 states, and the other one is very small.

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(12)
gmm <- mvnormalmixEM(lp, k = 2)
```

```
## number of iterations= 13
```

```
gmm$mu
```

```
## [[1]]
## [1] 0.9439668 0.9494717 1.0752029 1.2075319
##
## [[2]]
## [1] -0.2708856 -0.2724652 -0.3085457 -0.3465195
```

```
gmm$sigma
```

```
## [[1]]
##           [,1]      [,2]      [,3]      [,4]
## [1,] 2.2798979 2.2258528 1.4660136 0.7330673
## [2,] 2.2258528 2.3284537 1.2981383 0.1610122
## [3,] 1.4660136 1.2981383 1.5312247 0.9555286
## [4,] 0.7330673 0.1610122 0.9555286 2.2815467
##
## [[2]]
##            [,1]      [,2]       [,3]       [,4]
## [1,] 0.27736340 0.2542240 0.10740818 0.05184957
## [2,] 0.25422405 0.2595802 0.10950841 0.04024150
## [3,] 0.10740818 0.1095084 0.39434274 0.08754052
## [4,] 0.05184957 0.0402415 0.08754052 0.06746702
```

```
gmm$lambda
```

```
## [1] 0.2229782 0.7770218
```

```
posterior <- data.frame(gmm$posterior) %>%
  cbind(states)
```

```r
posterior$component <- ifelse(posterior$comp.1 > posterior$comp.2, 1, 2)
posterior <- posterior %>%
  select(component,stateabv)

posterior %>%
  count(component)
```

```
## # A tibble: 2 x 2
##   component     n
##       <dbl> <int>
## 1         1    11
## 2         2    38
```

The GMM puts 38 states in one group and 11 states in the other group.

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
#HC
lp %>%
  as.tibble() %>%
  mutate(assignment = cutree(hc_avg, 2)) %>%
  ggplot()+
  geom_point(aes(salary_real, expend, color = as.factor(assignment)))+
  geom_text(aes(salary_real, expend, label = rownames(lp)), hjust = 0.3, vjust = -0.2)+
  theme_minimal()+
  labs(x = "Salary", y = "Expenditures", title = "AHC")
```

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

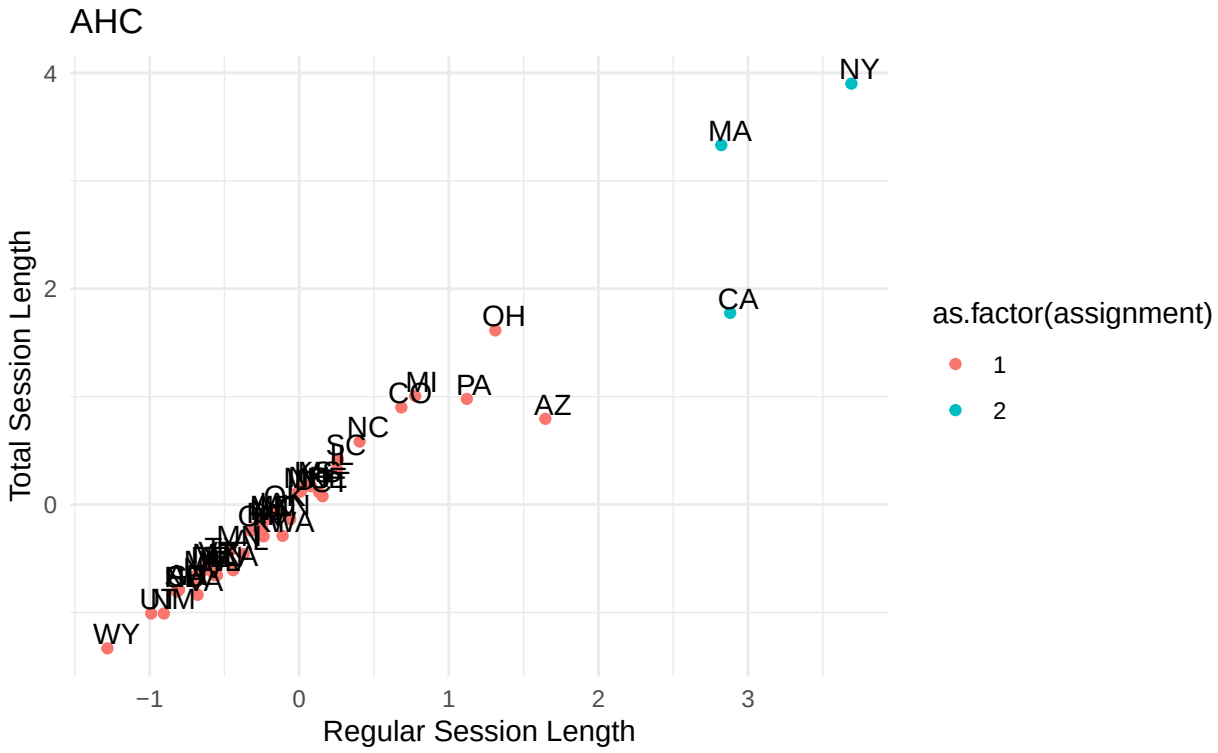

```
#kmeans
lp %>%
  as.tibble() %>%
  mutate(assignment = kmeans$cluster) %>%
  ggplot()+
  geom_point(aes(salary_real, expend, color = as.factor(assignment)))+
  geom_text(aes(salary_real, expend, label = rownames(lp)), hjust = 0.3, vjust = -0.2)+
  theme_minimal()+
  labs(x = "Salary", y = "Expenditures", title = "Kmeans")
```


```
# GMM
lp %>%
  cbind(posterior) %>%
  rename(assignment = component) %>%
  ggplot()+
  geom_point(aes(salary_real, expend, color = as.factor(assignment)))+
  geom_text(aes(salary_real, expend, label = stateabv), hjust = 0.3, vjust = -0.2)+
  theme_minimal()+
  labs(x = "Salary", y = "Expenditures", title = "GMM Classification")
```

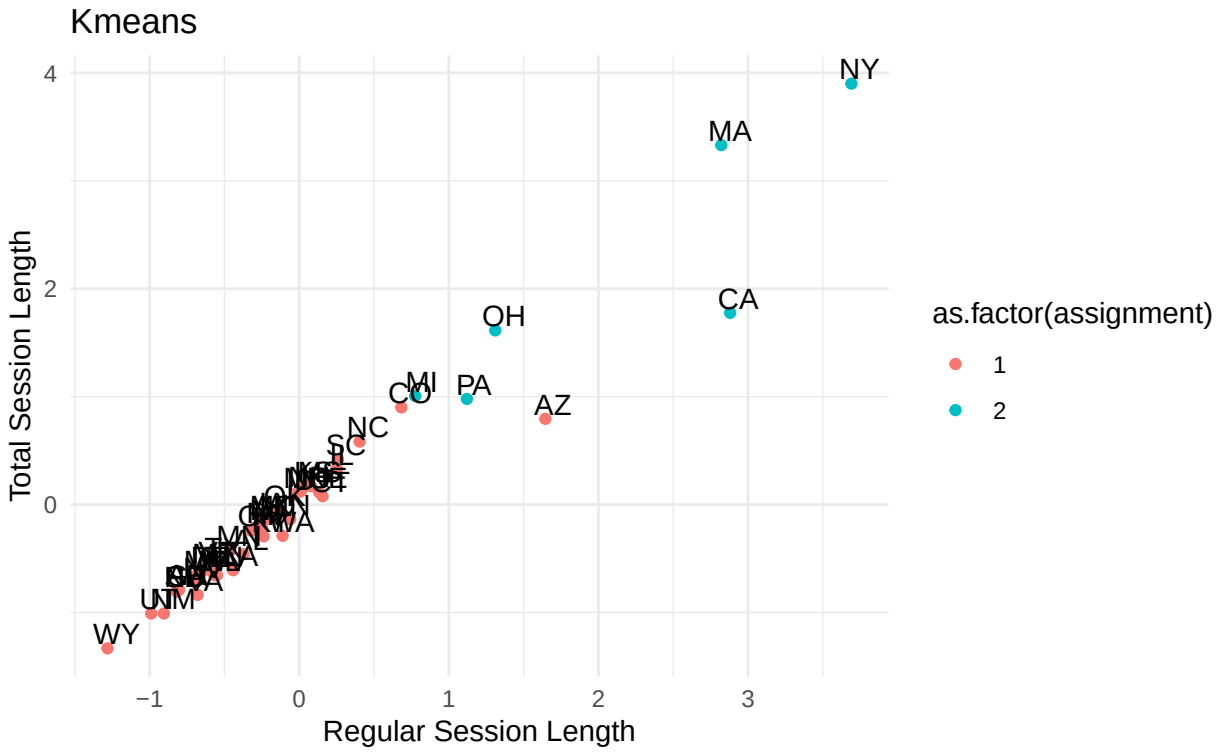
As we look at the pictures, it seems that the AHC algorithm and GMM are not producing a very good classification since the two groups have several states overlapping on each other (MA and NY for AHC, OH, IL, MA for GMM). But kmeans generates two relatively clean clusters, as there is only one state who belongs to the blue group (IL) in the middle of the red dots.

Now check total session length vs regular session length.

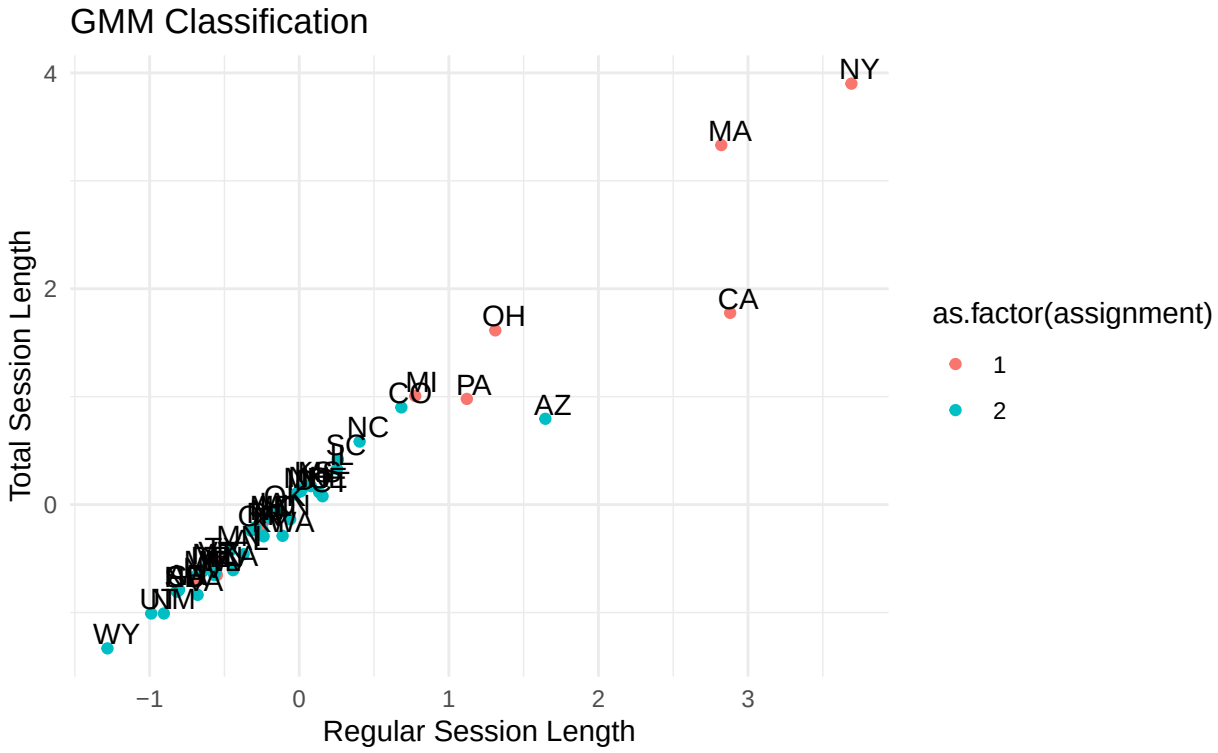
```
#HC
lp %>%
  as.tibble() %>%
  mutate(assignment = cutree(hc_avg, 2)) %>%
  ggplot()+
  geom_point(aes(t_slength, slength, color = as.factor(assignment)))+
  geom_text(aes(t_slength, slength, label = rownames(lp)), hjust = 0.3, vjust = -0.2)+
  theme_minimal()+
  labs(x = "Regular Session Length", y = "Total Session Length", title = "AHC")
```




```
#kmeans
lp %>%
  as.tibble() %>%
  mutate(assignment = kmeans$cluster) %>%
  ggplot()+
  geom_point(aes(t_slength, slength, color = as.factor(assignment)))+
  geom_text(aes(t_slength, slength, label = rownames(lp)), hjust = 0.3, vjust = -0.2)+
  theme_minimal()+
  labs(x = "Regular Session Length", y = "Total Session Length", title = "Kmeans")
```




```
# GMM
lp %>%
  cbind(posterior) %>%
  rename(assignment = component) %>%
  ggplot()+
  geom_point(aes(t_slength, slength, color = as.factor(assignment)))+
  geom_text(aes(t_slength, slength, label = stateabv), hjust = 0.3, vjust = -0.2)+
  theme_minimal()+
  labs(x = "Regular Session Length", y = "Total Session Length", title = "GMM Classification")
```



When looking at total session length and regular session length, it seems that hierarchical clustering divides the two groups with no overlapping at all. But just eyeballing the graphs is not a good way to decide which method does the best. Once we use another data set, the best model may change.

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).

I will compare the average silhouette widths of the three models.

```
library(clValid)
```

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

```
library(mclust)
```

```
## Package 'mclust' version 5.4.5
## Type 'citation("mclust")' for citing this R package in publications.
```

```
##
## Attaching package: 'mclust'
```

```
## The following object is masked from 'package:mixtools':
##
##     dmvnorm
```

```
## The following object is masked from 'package:purrr':
##
##     map
```

```
cl_validation <- clValid(lp, 2:5, validation = "internal",
                         clMethods = c("hierarchical", "kmeans", "model"))
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. Discuss the validation output

On average, the hierarchical clustering model produces the highest average silhouette width. In addition, we have the best model performance with respect to average silhouette width when we use hierarchical clustering with 2 clusters.

From the pictures in question 7, kmeans seems to generate the best classification. However, if we use average silhouette to measure the model performance, hierarchical clustering has the best performance. In addition, the fit of the models in general decreases as k increases, which means that we usually have the best model with just k=2 (two clusters).

Note that if we switch to another data set, the optimal model may change. We sometimes may choose a sub-optimal model for faster computation speed. For example, I noticed that when fitting the models, it takes longer for GMM to produce the results. If we have a very large data set, using a sub-optimal model instead of GMM could be more efficient. Also, depending on the objective of the classification, we may want to have an observation belonging to multiple clusters. In this case, although kmeans may produce the optimal model, but we may still use GMM for soft-partitioning.