borui sun ps4

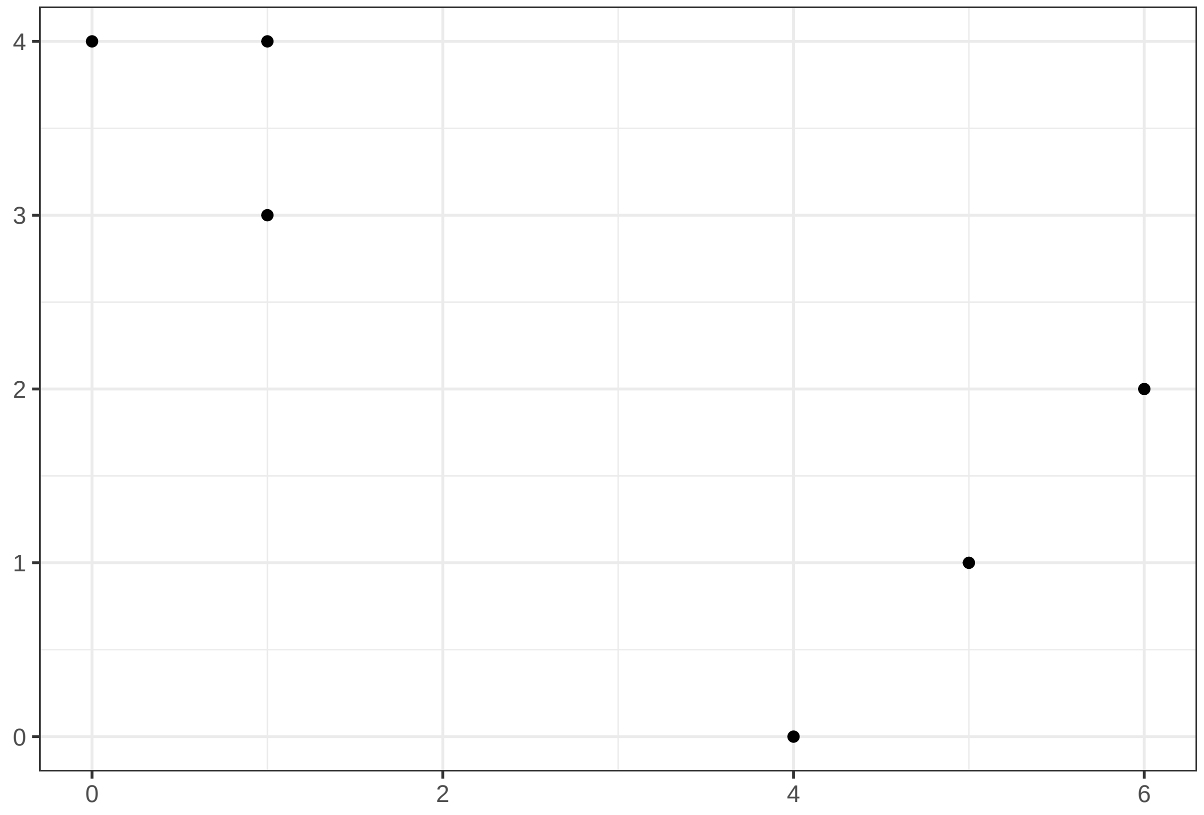
borui sun

3/1/2020

Performing k-Means By Hand

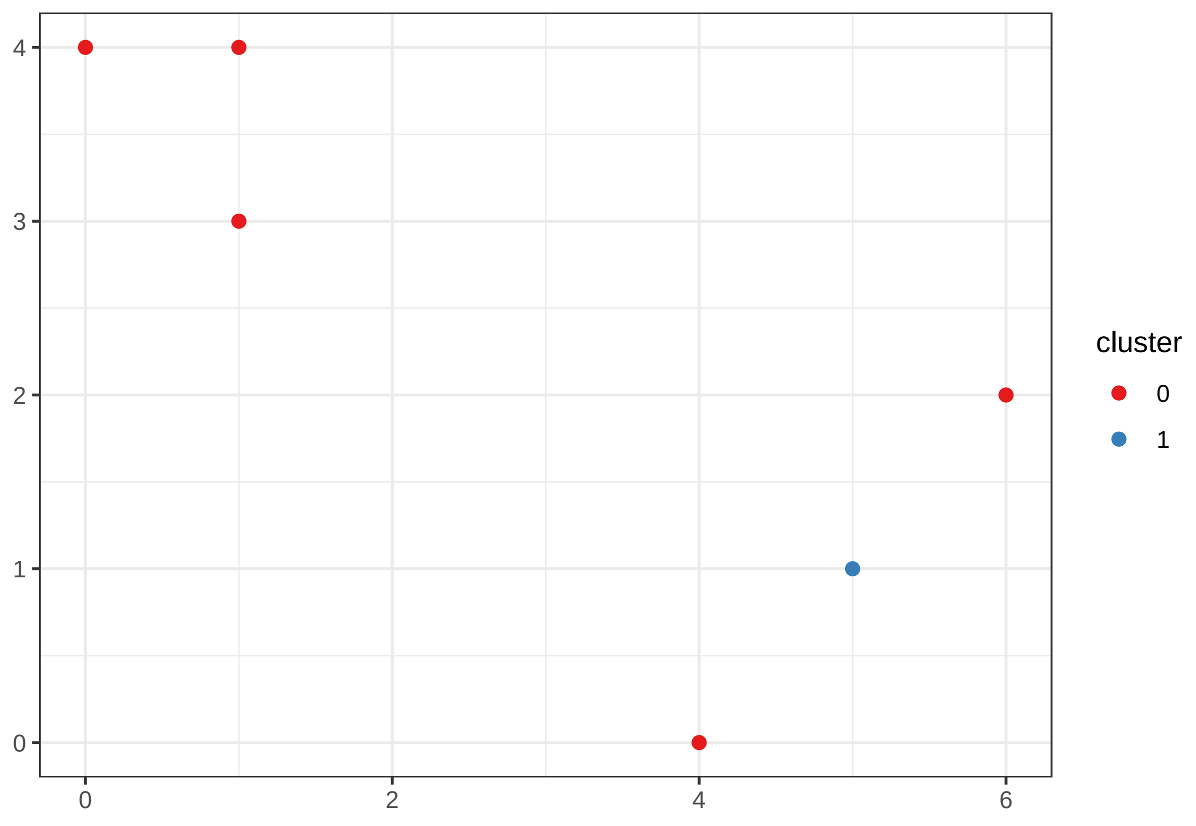
1. (5 points) Plot the observations.

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

ggplot(x, aes(x = V1, y = V2)) +
  geom_point() +
  labs(x = "", y = "")
```



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(906)
x$label <- sample(c(0,1), 6, replace = TRUE)

ggplot(x, aes(x = V1, y = V2, color = as.factor(label))) +
  geom_point(size = 2) +
  labs(x = "", y = "", color = "cluster") +
  scale_color_brewer(palette = "Set1")
```



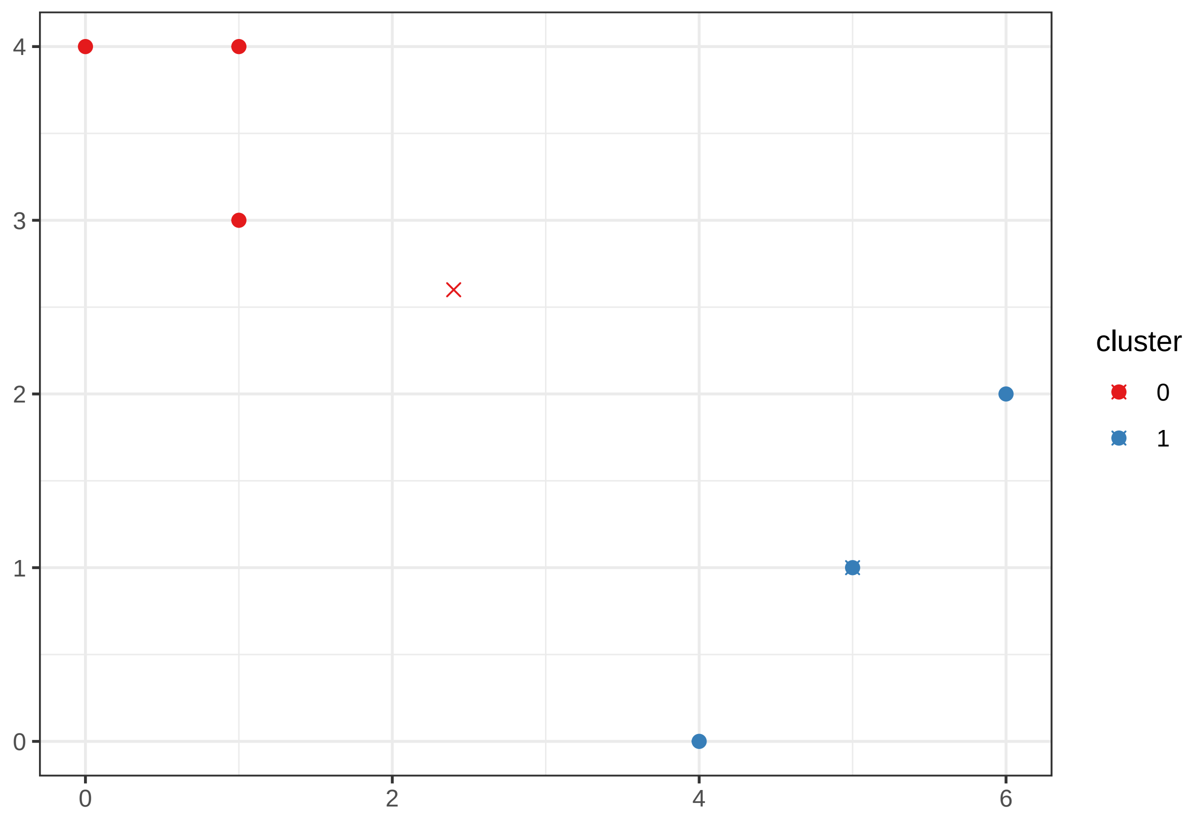
3. (10 points) Compute the centroid for each cluster.

```
centroids <- x %>%
  group_by(label) %>%
  summarise(V1.cen = mean(V1), V2.cen = mean(V2))
centroids %>% kable()
```

label	V1.cen	V2.cen
0	2.4	2.6
1	5.0	1.0

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.

```
for (i in 1:nrow(x)) {
  distance.c0 <- sqrt((x$V1[i]-centroids$V1.cen[1])^2 + (x$V2[i]-centroids$V2.cen[1])^2)
  distance.c1 <- sqrt((x$V1[i]-centroids$V1.cen[2])^2 + (x$V2[i]-centroids$V2.cen[2])^2)

  if(distance.c0 < distance.c1){x$label[i] <- 0}
  else{x$label[i] <- 1}
}

ggplot(x, aes(x = V1, y = V2, color = as.factor(label))) +
  geom_point(size = 2) +
  geom_point(data = centroids,
             aes(x = V1.cen, y = V2.cen, color = as.factor(label)),
             shape = 4, size = 2) +
  labs(x = "", y = "", color = "cluster") +
  scale_color_brewer(palette = "Set1")
```

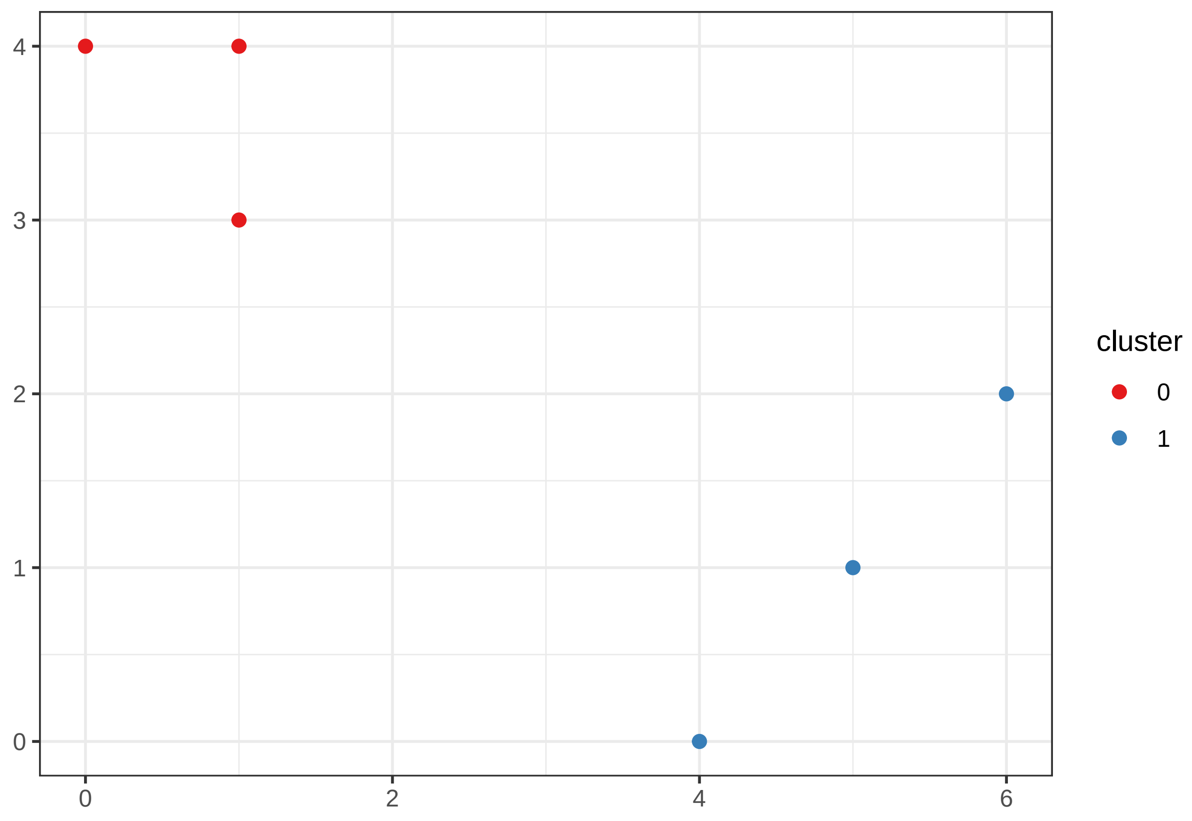


5. (5 points) Repeat (3) and (4) until the answers/clusters stop changing.

```
for (i in 1:10000000){

  label.old <- x$label

  cent <- x %>%
    group_by(label) %>%
    summarise(V1.cen = mean(V1), V2.cen = mean(V2))

  for (i in 1:nrow(x)) {
    distance.c0 <- sqrt((x$V1[i]-cent$V1.cen[1])^2 + (x$V2[i]-cent$V2.cen[1])^2)
    distance.c1 <- sqrt((x$V1[i]-cent$V1.cen[2])^2 + (x$V2[i]-cent$V2.cen[2])^2)

    if(distance.c0 < distance.c1){x$label[i] <- 0}
    else{x$label[i] <- 1}
  }
  label.new <- x$label

  if(sum(label.old - label.new) == 0) {break}
}
```

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.

```r
ggplot(x, aes(x = V1, y = V2, color = as.factor(label))) +
  geom_point(size = 2) +
  labs(x = "", y = "", color = "cluster") +
  scale_color_brewer(palette = "Set1")
```



Clustering State Legislative Professionalism

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

```r
load("./Data and Codebook/legprof-components.v1.0.RData")
```

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)

```
legprof <- x %>%
  filter(sessid == "2009/10") %>%
  select(stateabv, t_slength, slength, salary_real, expend) %>%
  drop_na() %>%
  remove_rownames() %>%
  column_to_rownames(var = "stateabv") %>%
  scale()
```

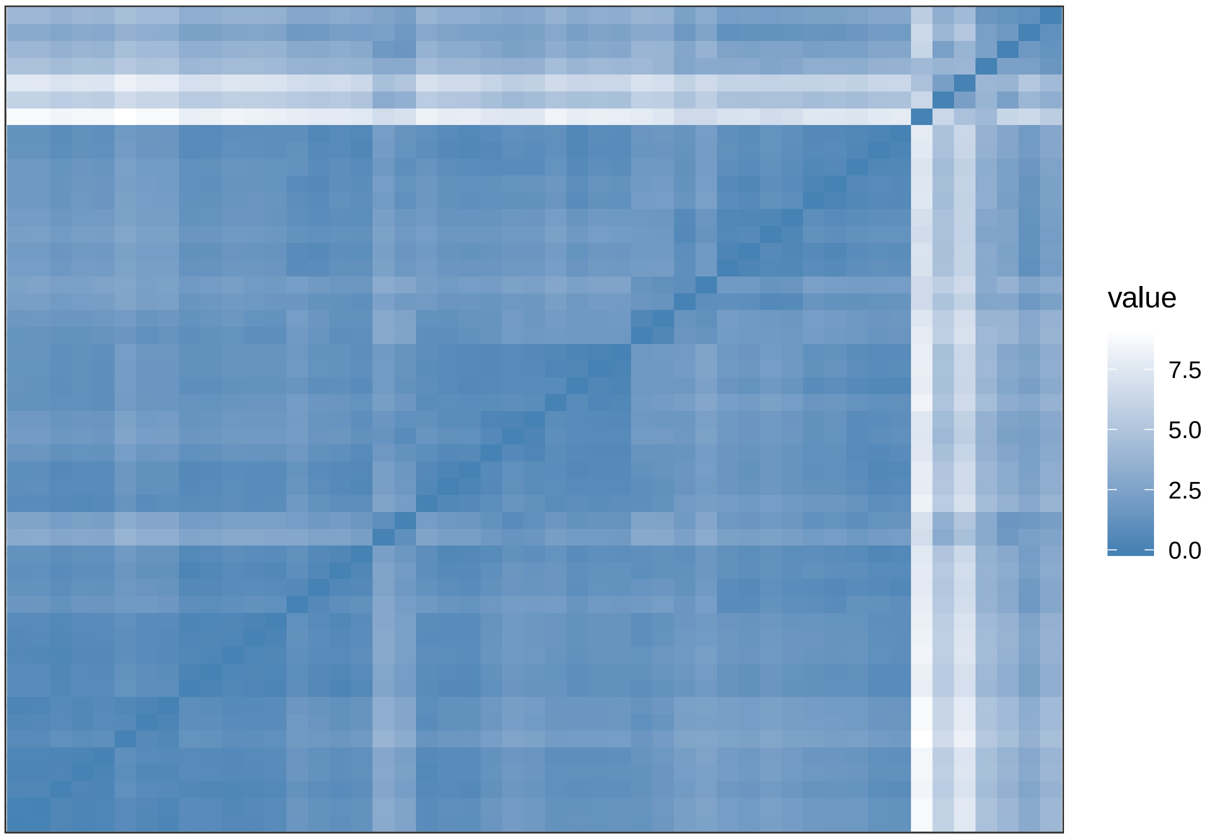
3. (5 points) Diagnose clusterability in any way you'd prefer (e.g., sparse sampling, ODI, etc.); display the results and discuss the likelihood that natural, non-random structure exist in these data. *Hint:* We didn't cover how to do this R in class, but consider `dissplot()` from the `seriation` package, the `factoextra` package, and others for calculating, presenting, and exploring the clusterability of some feature space.

I use the Hopkins Statistics along with a visual assessment of dissimilarity matrix to diagnose clusterability. We have a Hopkins Statistics of 0.84, which is very close to 1, indicating that our data is highly clustered. Similarly, in the tendency plot, we also observe a large amount of data (the blue squares) that has high level of similarity with each other.

```
clustend  <- get_clust_tendency(legprof, 40)

cat("Hopkins Statistics:", clustend$hopkins_stat)
```

```
## Hopkins Statistics: 0.8406165
```

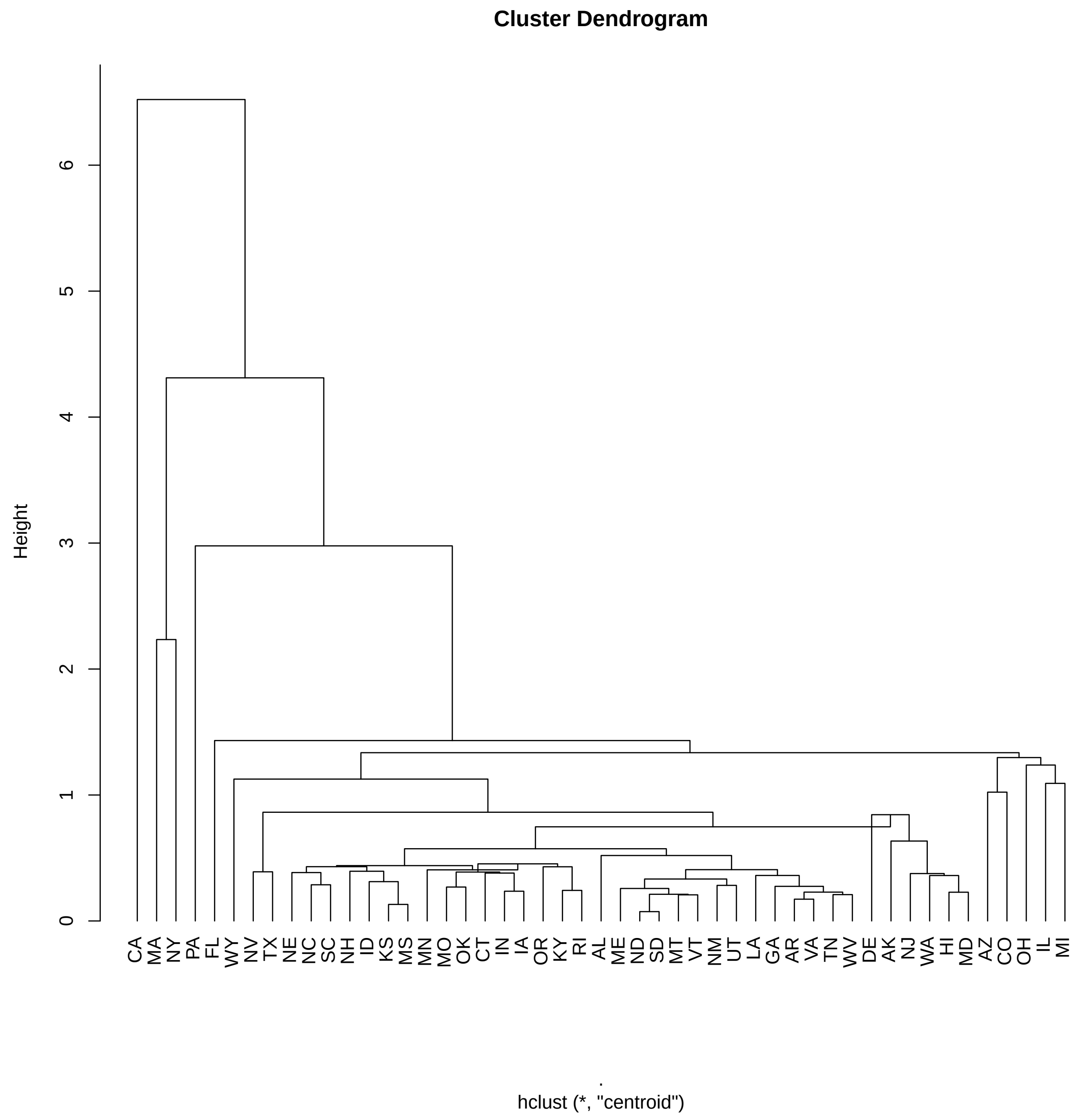
```
clustend$plot +
  scale_fill_gradient(low = "steelblue", high = "white")
```



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.

A centroid linkage method is used (I understand that average, complete and single linkage are more popular and centroid is often used in genomics but the result seems really interesting). According to the dendrogram plot beblow, we have one big cluster with over 40 states and a number of small clusters. Most notable feature from this plot is that California is in its own cluster with a very long branch, showing high level of dissimilarity with the rest. Massachusetts and New York are grouped together. Pennsylvania also has a very long branch. The four states all have large population and greater representations of the Democratic party, however, further research is needed in order to fully explain those clusters.

```r
hc <- legprof %>% dist() %>%
  hclust(method = "centroid"); plot(hc, hang = -1)
```



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.

By using the kmeans algorithm with a k of 2, we divide the 49 states into one big cluster and one small cluster with only 6 sates (shown in the table beblow). Similar to the agglomerative hierarchical clustering result, the kmeans also separates California, Massachusetts, New York, Pennsylvania from the huge, main cluster. In addition to the four, it also identifies Michigan and Ohio.

```
set.seed(906)
kmeans <- kmeans(legprof,
                 center = 2,
                 nstart = 15)
```

```r
kmeans$cluster %>% as.data.frame() %>%
  rownames_to_column() %>%
  `colnames<-`(c("state", "cluster")) %>%
  filter(cluster == 2) %>% kable()
```

state	cluster
CA	2
MA	2
MI	2
NY	2
OH	2
PA	2

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.

The Gaussian mixture model also has a big cluster and a small one, but contrary to the previous two methods, it identifies more states in the small cluster. There are 9 states in the smaller cluster. Consistently, California, Massachusetts, New York and Pennsylvania are grouped into the smaller cluster. In addition to the four states, we also have Alaska, Arizona, Florida, Nevada and Texas.

```r
set.seed(906)
gmm <- mvnormalmixEM(legprof, k = 2)
```

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

```r
gmm.cluster <- gmm$posterior %>% as.data.frame() %>%
  mutate(cluster = ifelse(comp.1 > comp.2, 1, 2)) %>%
  bind_cols(tibble(state = rownames(legprof))) %>%
  select(state, cluster)

gmm.cluster %>% filter(cluster == 1) %>% kable()
```

state	cluster
AK	1
AZ	1
CA	1
FL	1
MA	1
NV	1
NY	1
PA	1
TX	1

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.

The table beblow shows the 11 states that are grouped into the smaller cluster by at least one of the algorithems. We have: * Four states (California, Massachusetts, New York, Pennsylvania) that are consistently grouped into the smaller cluster by all methods; * Only one states (Florida) that is grouped into the smaller cluster by agglomerative hierarchical and gmm but not kmeans; * Six states that only one of the algorithems puts them in the smaller cluster.

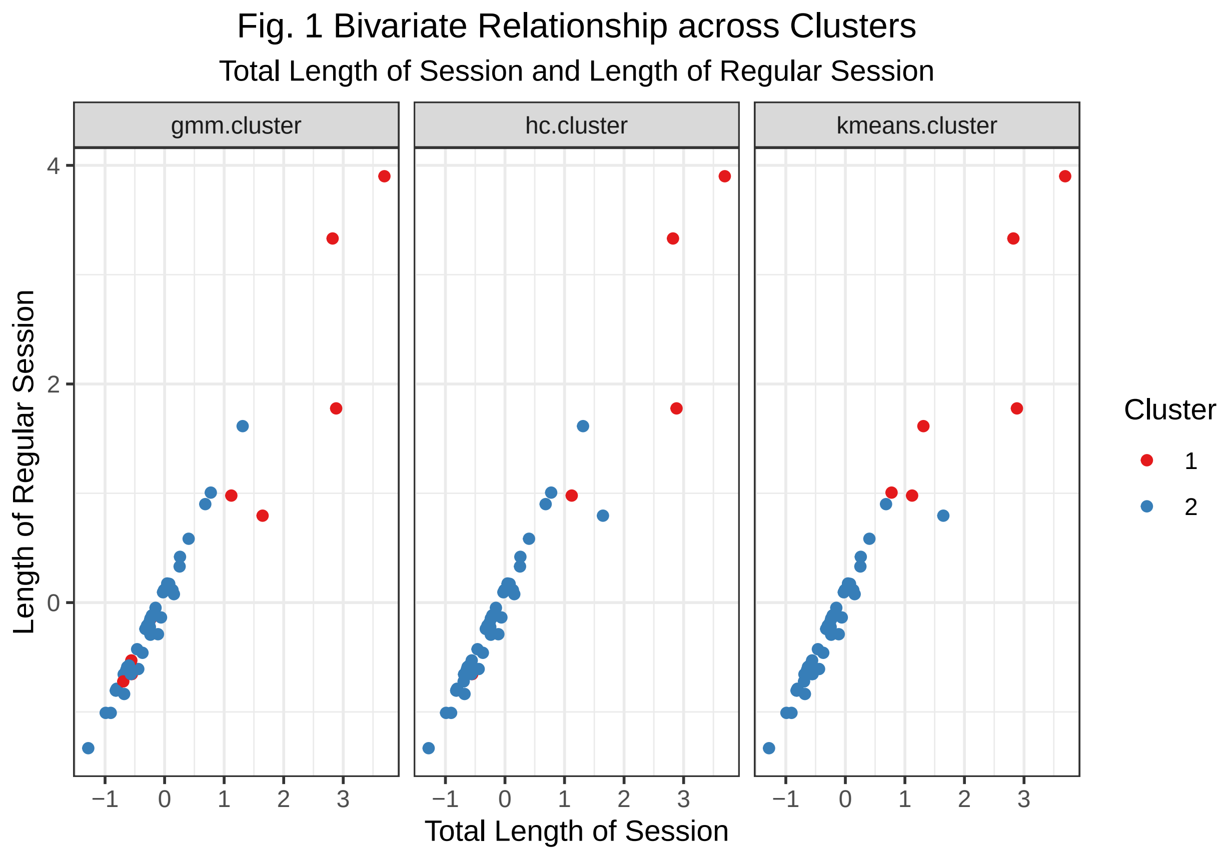
```
cluster_results <- legprof %>% as.data.frame() %>%
  rownames_to_column() %>%
  mutate(hc.cluster = cutree(hc, k = c(6))) %>% # because the hc method has 1 big cluster
                                                # and several very small clusters with 1 or 2 leaves,
                                                # I use a cut of 6 and treat the small clusters togethe
  mutate(kmeans.cluster = kmeans$cluster) %>%
  mutate(gmm.cluster = gmm.cluster$cluster) %>%
  mutate(hc.cluster = ifelse(hc.cluster == 1, 2, 1),
         kmeans.cluster = ifelse(kmeans.cluster == 1, 2, 1))

cluster_results %>% filter(gmm.cluster == 1 | kmeans.cluster == 1 | hc.cluster == 1) %>% kable()
```

rowname	t_slength	slength	salary_real	expend	hc.cluster	kmeans.cluster	gmm.cluster
AK	-0.2294089	-0.1452309	0.4011333	0.8591198	2	2	1
AZ	1.6453067	0.7951955	-0.1335656	-0.1299408	2	2	1
CA	2.8807257	1.7767099	3.2069914	5.4785453	1	1	1
FL	-0.5503063	-0.6538290	0.1121391	1.5395006	1	2	1
MA	2.8214940	3.3312922	1.2640239	-0.3053017	1	1	1
MI	0.7755062	1.0063116	2.1381147	0.4568995	2	1	2
NV	-0.6938656	-0.7210023	-0.8557336	0.6161906	2	2	1
NY	3.6912946	3.9007112	2.1319915	1.4714288	1	1	1
OH	1.3107315	1.6145208	1.3598244	-0.2330237	2	1	2
PA	1.1204292	0.9792801	2.0836049	1.9377038	1	1	1
TX	-0.5587509	-0.5290785	-0.8193559	0.9257258	2	2	1

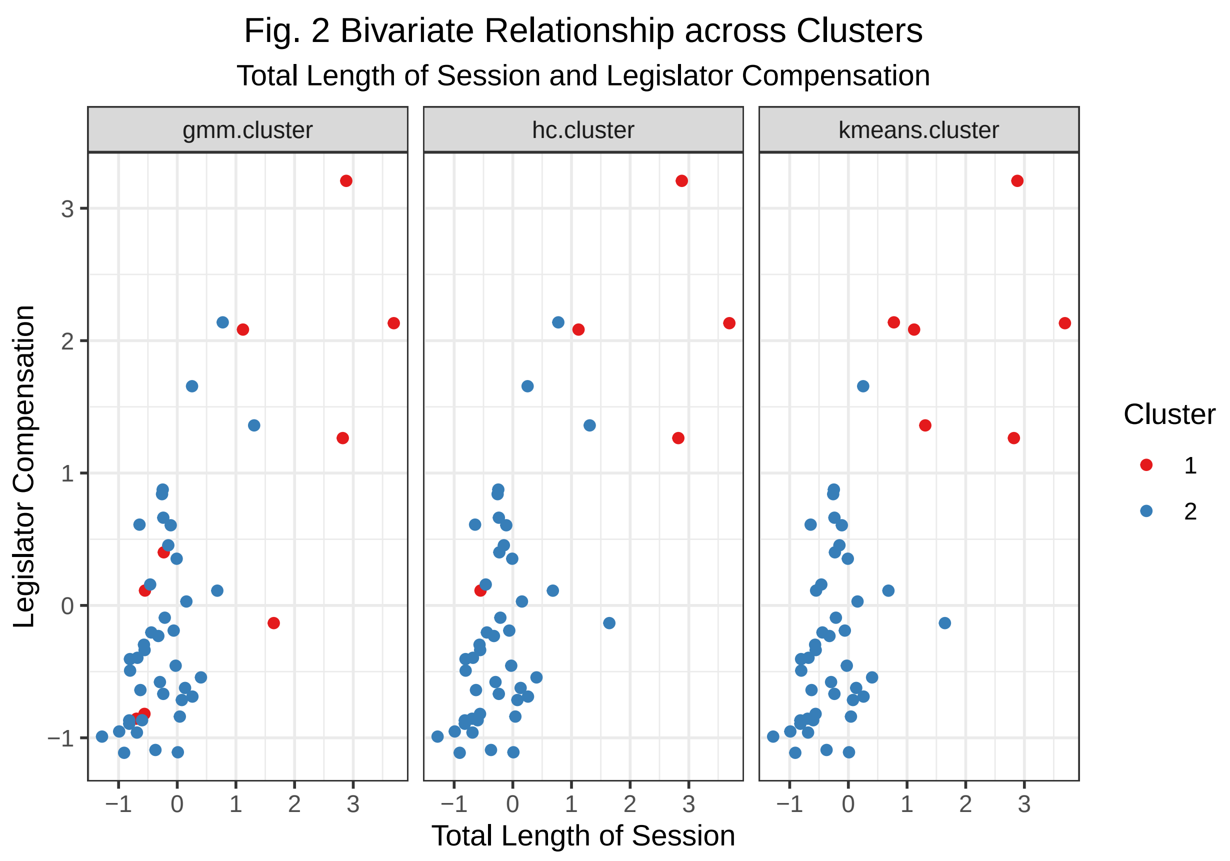
```
cluster_results <- cluster_results %>%
  pivot_longer(cols = ends_with("cluster"),
               names_to = "model",
               values_to = "cluster")
```

To further understand the intra-cluster dissimilarity and the inter-cluster similarity, I also create 6 plots of bivariate relationship between different features. While the results are very similar across the three algorithms, the Gaussian mixture seems to perform relatively worse than the other two. It has more "misclassification" (reds dots placed in obvious blue dots region or vice versa). This can be seen in Fig. 1, Fig. 2, Fig. 4 and Fig. 6. We also observe that certain algortithm has a better classification than the other with respects to certain features. For example, the kmeans has more "misclassification" than the other two method when only looking at the relationship between total sesssion length and expenditure (Fig.3). However, at Fig. 4 when looking at the relationship between regular session length and legislator compensation, kmeans is the only one that does not have "missclassification".

```
ggplot(data = cluster_results) +
  geom_point(aes(x = t_slength, y = slength, color = as.factor(cluster))) +
  facet_wrap(~model) +
  scale_color_brewer(palette = "Set1") +
  labs(x = "Total Length of Session",
       y = "Length of Regular Session",
       color = "Cluster",
       title = "Fig. 1 Bivariate Relationship across Clusters",
       subtitle = "Total Length of Session and Length of Regular Session")
```



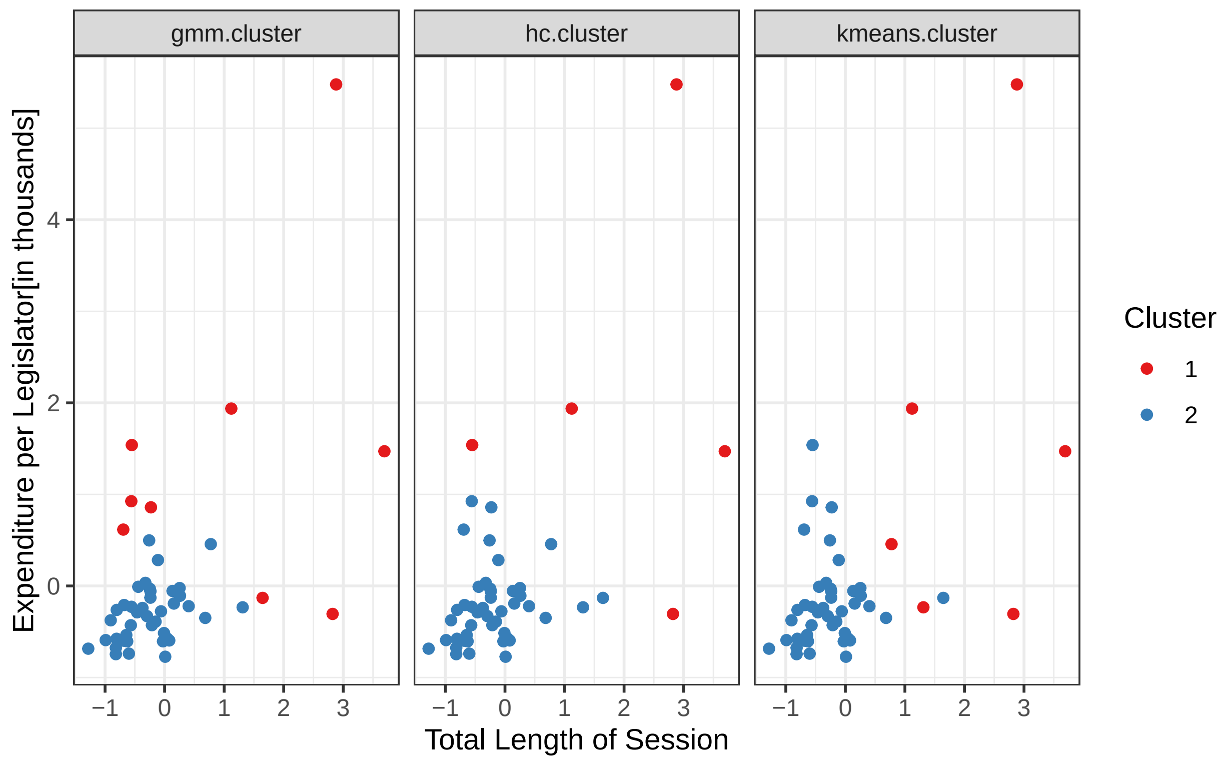
```
ggplot(data = cluster_results) +
  geom_point(aes(x = t_slength, y = salary_real, color = as.factor(cluster))) +
  facet_wrap(~model) +
  scale_color_brewer(palette = "Set1") +
  labs(x = "Total Length of Session",
       y = "Legislator Compensation",
       color = "Cluster",
       title = "Fig. 2 Bivariate Relationship across Clusters",
       subtitle = "Total Length of Session and Legislator Compensation")
```



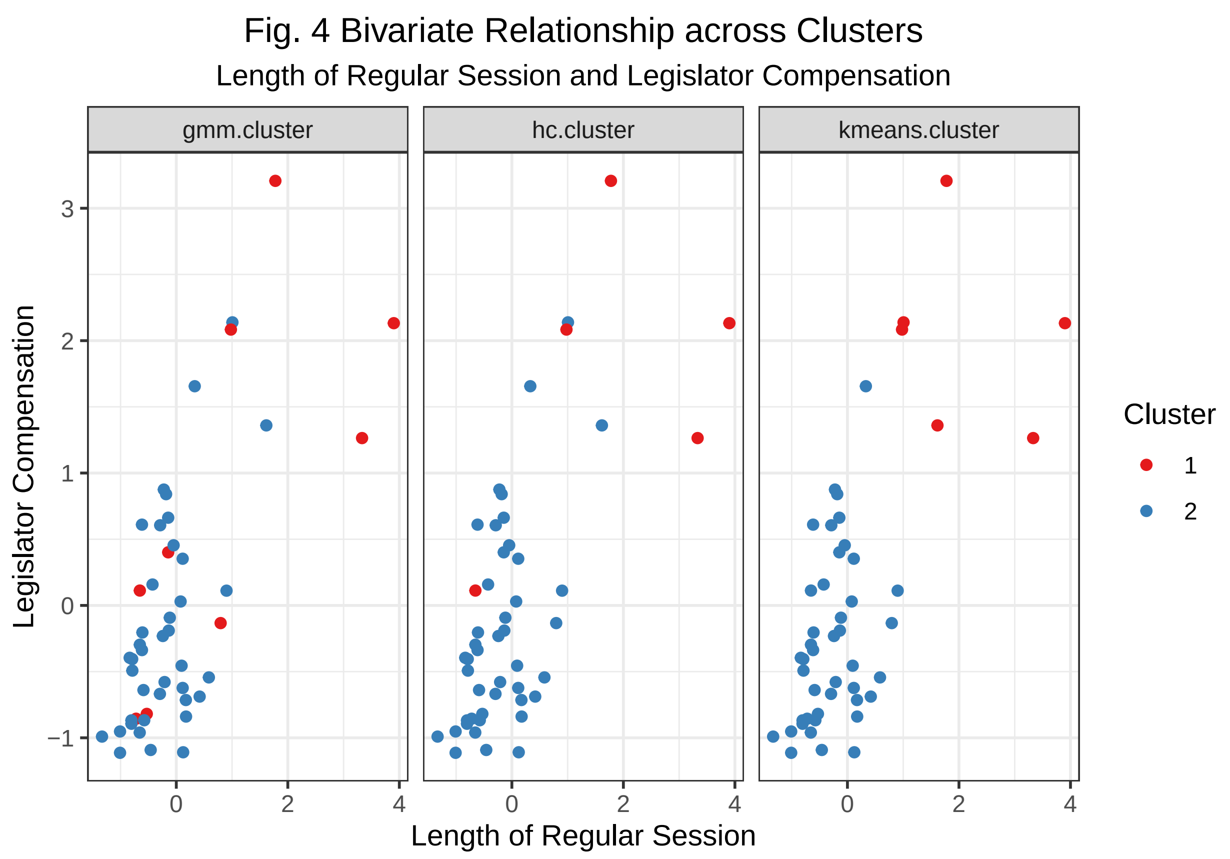
```r
ggplot(data = cluster_results) +
  geom_point(aes(x = t_slength, y = expend, color = as.factor(cluster))) +
  facet_wrap(~model) +
  scale_color_brewer(palette = "Set1") +
  labs(x = "Total Length of Session",
       y = "Expenditure per Legislator[in thousands]",
       color = "Cluster",
       title = "Fig. 3 Bivariate Relationship across Clusters",
       subtitle = "Total Length of Session and Expenditure")
```

Fig. 3 Bivariate Relationship across Clusters

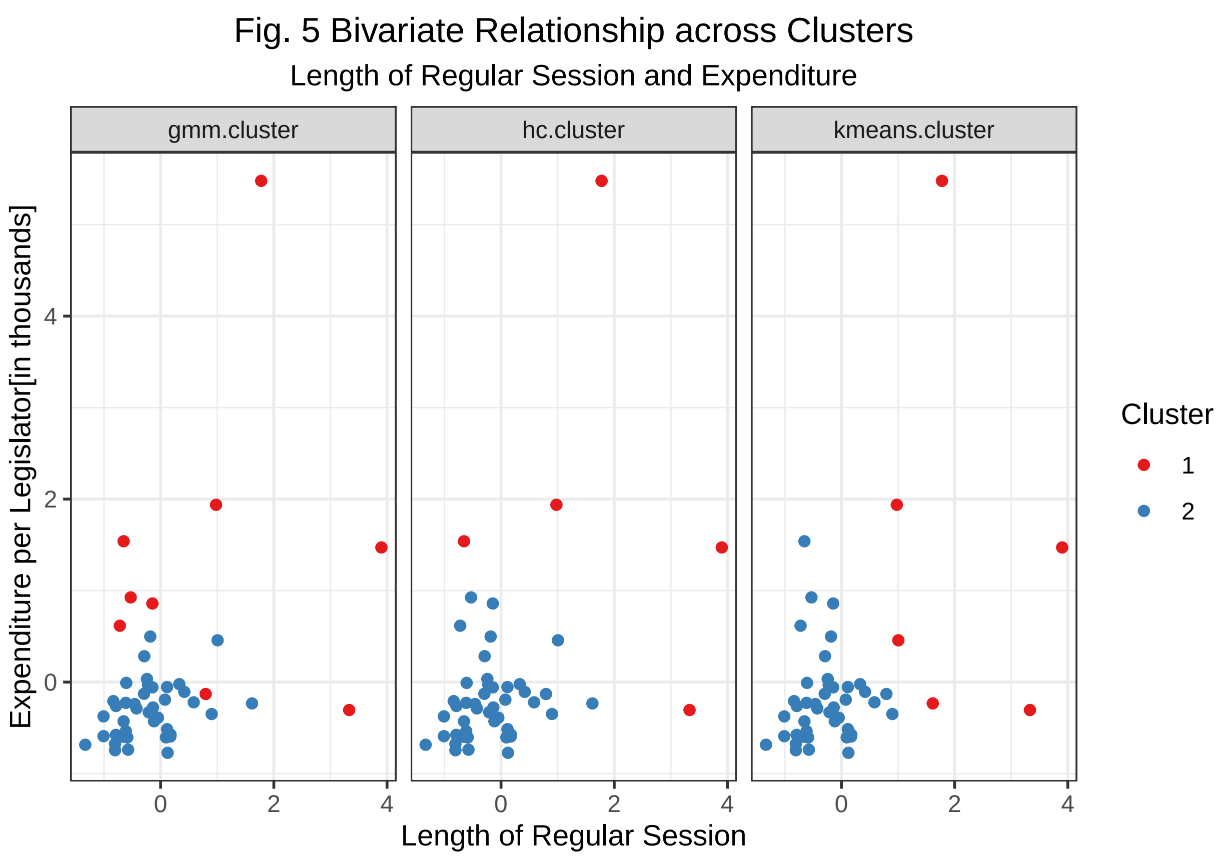
Total Length of Session and Expenditure



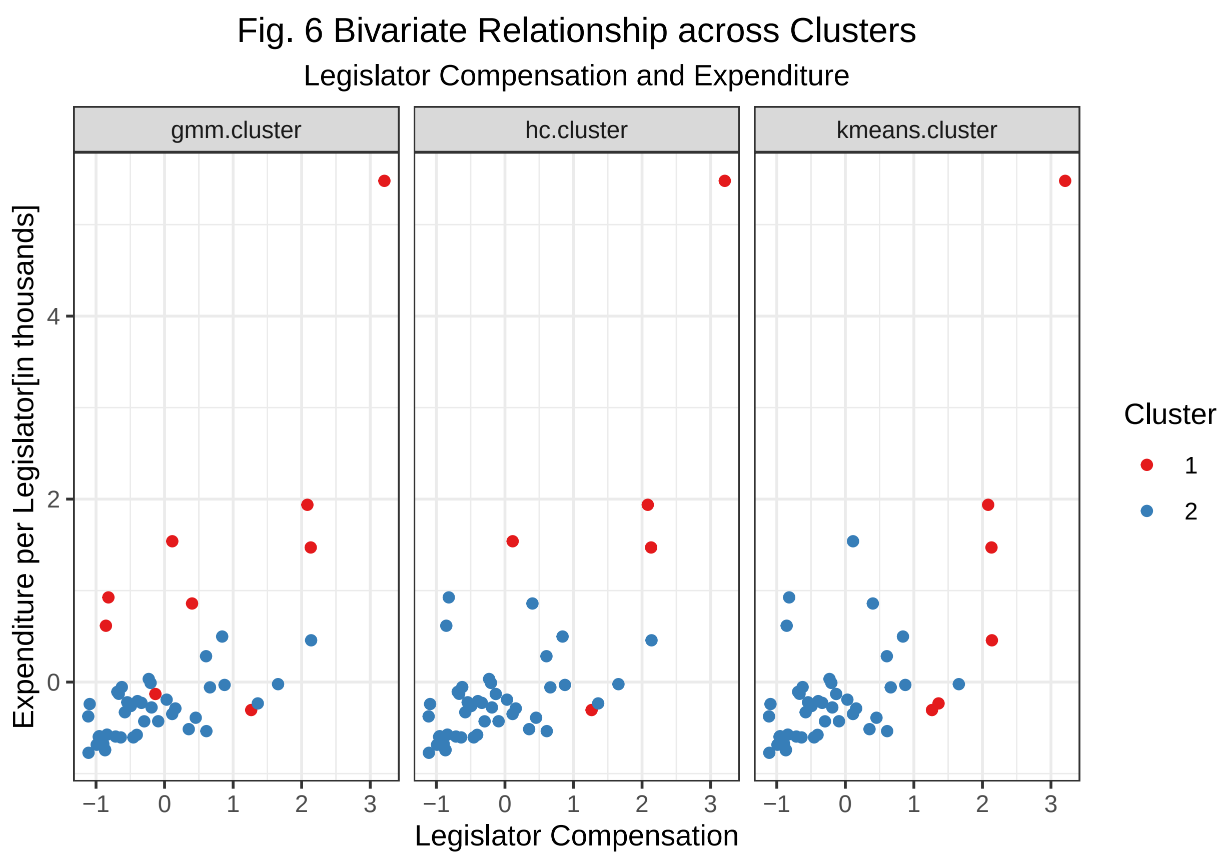
```r
ggplot(data = cluster_results) +
  geom_point(aes(x = slength, y = salary_real, color = as.factor(cluster))) +
  facet_wrap(~model) +
  scale_color_brewer(palette = "Set1") +
  labs(x = "Length of Regular Session",
       y = "Legislator Compensation",
       color = "Cluster",
       title = "Fig. 4 Bivariate Relationship across Clusters",
       subtitle = "Length of Regular Session and Legislator Compensation")
```



```r
ggplot(data = cluster_results) +
  geom_point(aes(x = slength, y = expend, color = as.factor(cluster))) +
  facet_wrap(~model) +
  scale_color_brewer(palette = "Set1") +
  labs(x = "Length of Regular Session",
       y = "Expenditure per Legislator[in thousands]",
       color = "Cluster",
       title = "Fig. 5 Bivariate Relationship across Clusters",
       subtitle = "Length of Regular Session and Expenditure")
```



```r
ggplot(data = cluster_results) +
  geom_point(aes(x = salary_real, y = expend, color = as.factor(cluster))) +
  facet_wrap(~model) +
  scale_color_brewer(palette = "Set1") +
  labs(x = "Legislator Compensation",
       y = "Expenditure per Legislator[in thousands]",
       color = "Cluster",
       title = "Fig. 6 Bivariate Relationship across Clusters",
       subtitle = "Legislator Compensation and Expenditure")
```



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.

The clValid function uses three validation strategy- Connectivity, Dunn and Silhouette, but we will stick to the Dunn Index. The Dunn index is the ratio of the smallest distance between observations not in the same cluster (minimal inter-cluster distance) to the largest intra-cluster distance. By comparing k from 2 to 15, we find that hierachical clustering with 4 clusters has the highest Dunn score among other (though 0.28 is still quite low). Looking at the dendrogram below, we can see that by using a cluster of 4, we have California in its own cluster, Massachusetts and New York in one cluster, Pennsylvania, Ohio, Illionis and Michigan in one cluster and the rest of states in another cluster.

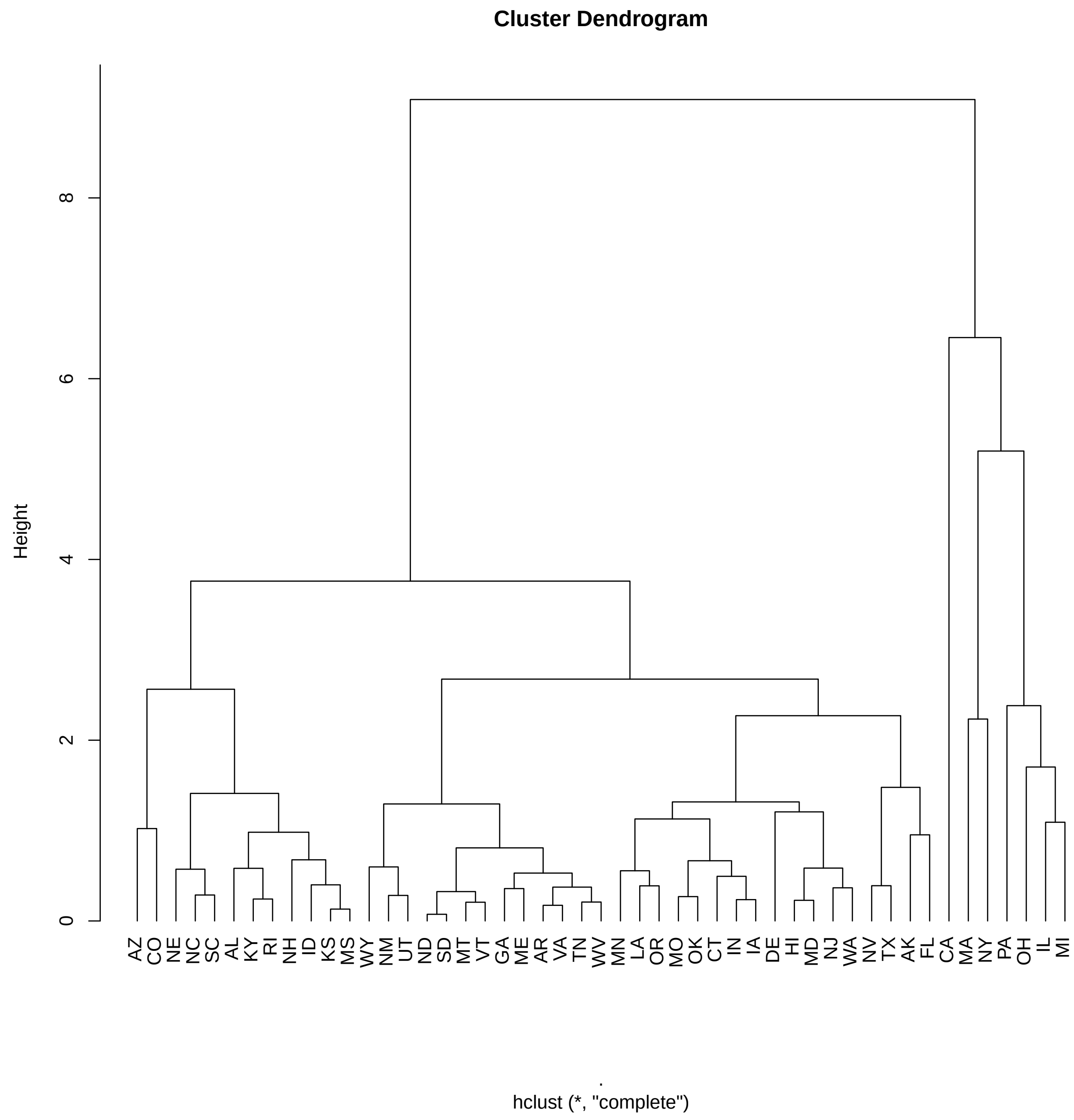
Note: I switched the hierarchical cluster method to complete here, so another dendrogram is provided

```r
intel.valid <- clValid(legprof, 2:15,
                       clMethods = c("hierarchical", "kmeans", "model"),
                       validation = "internal",
                       method = c("complete"))

summary(intel.valid)
```

```
##
## Clustering Methods:
```

```
##  hierarchical kmeans model
##
## Cluster sizes:
##  2 3 4 5 6 7 8 9 10 11 12 13 14 15
##
## Validation Measures:
##                                  2       3       4       5       6       7       8       9      10
##
## hierarchical Connectivity   7.9071 10.5238 12.9583 25.7397 31.7056 34.9440 37.2218 42.7048 44.7048 4
##              Dunn           0.1673  0.2077  0.2872  0.1731  0.1094  0.1177  0.1235  0.1255  0.1647
##              Silhouette     0.6204  0.5884  0.5236  0.2391  0.3125  0.3057  0.2996  0.3362  0.3213
## kmeans       Connectivity   8.4460 10.8960 16.1885 28.7437 35.1774 38.4913 39.3079 43.6623 45.6623 4
##              Dunn           0.1735  0.2581  0.2562  0.1090  0.1130  0.1181  0.1206  0.1057  0.1386
##              Silhouette     0.6458  0.6131  0.4932  0.3042  0.3388  0.3267  0.3216  0.3437  0.3288
## model        Connectivity  10.7393 28.6119 39.0687 67.8401 80.4806 69.9774 72.4377 46.7254 60.0976 6
##              Dunn           0.1522  0.0633  0.0225  0.0258  0.0283  0.0543  0.0710  0.1810  0.0977
##              Silhouette     0.6314  0.2588  0.1861  0.0085 -0.0562  0.0917  0.0752  0.2831  0.1905
##
## Optimal Scores:
##
##              Score  Method       Clusters
## Connectivity 7.9071 hierarchical 2
## Dunn         0.2872 hierarchical 4
## Silhouette   0.6458 kmeans       2
```

```r
legprof %>% dist() %>%
  hclust(method = "complete") %>% plot( hang = -1)
```



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?

From the results above, we can see that different internal validation method chooses different optimal algorithm. This happens because in our case, we have one huge cluster with high level of similaritiy but there also is relatively high level of dissimilarity in the smaller cluster. This can be seen in the two dendrograms (one with centroid linkage and one with complete linkage), where observations in the smaller cluster has high branches, California in particular. Therefore, it is really difficult to choose an optimal approch solely based on the technical clustering results. Literature analysis is necessary to assist the selection. Additionally, the optimal appraoch also depends on which validation method you use. While for the hierarchical clustering algorithm, the optimal k depends the which linkage method we use. However, comparing across the hierarchical clustering with two different linkage methods as well as the the other two algorithm, the optimal k would seem to be 2 with one big cluster and one small cluster.

It is very possible to select a sub-optimal clustering method, regardless of the validation statistics. For example, if the context of problem or literature reviews provide strong theoretical assumptions or empirical evidence to support the selection of a sub-optimal clustering method, it is not necessarily to solely rely on the validation statistics.