Problem Set 4

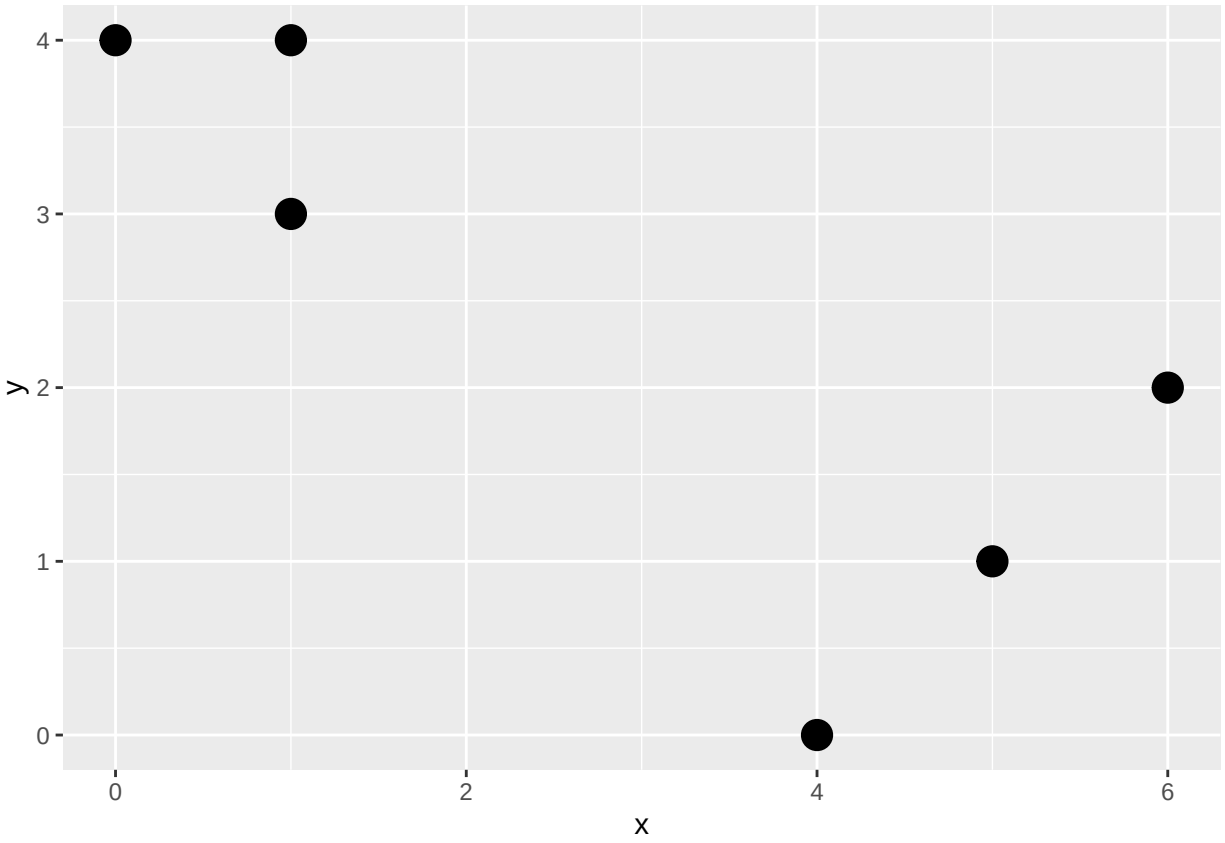
Siva

2/24/2020

Performing k-Means By Hand

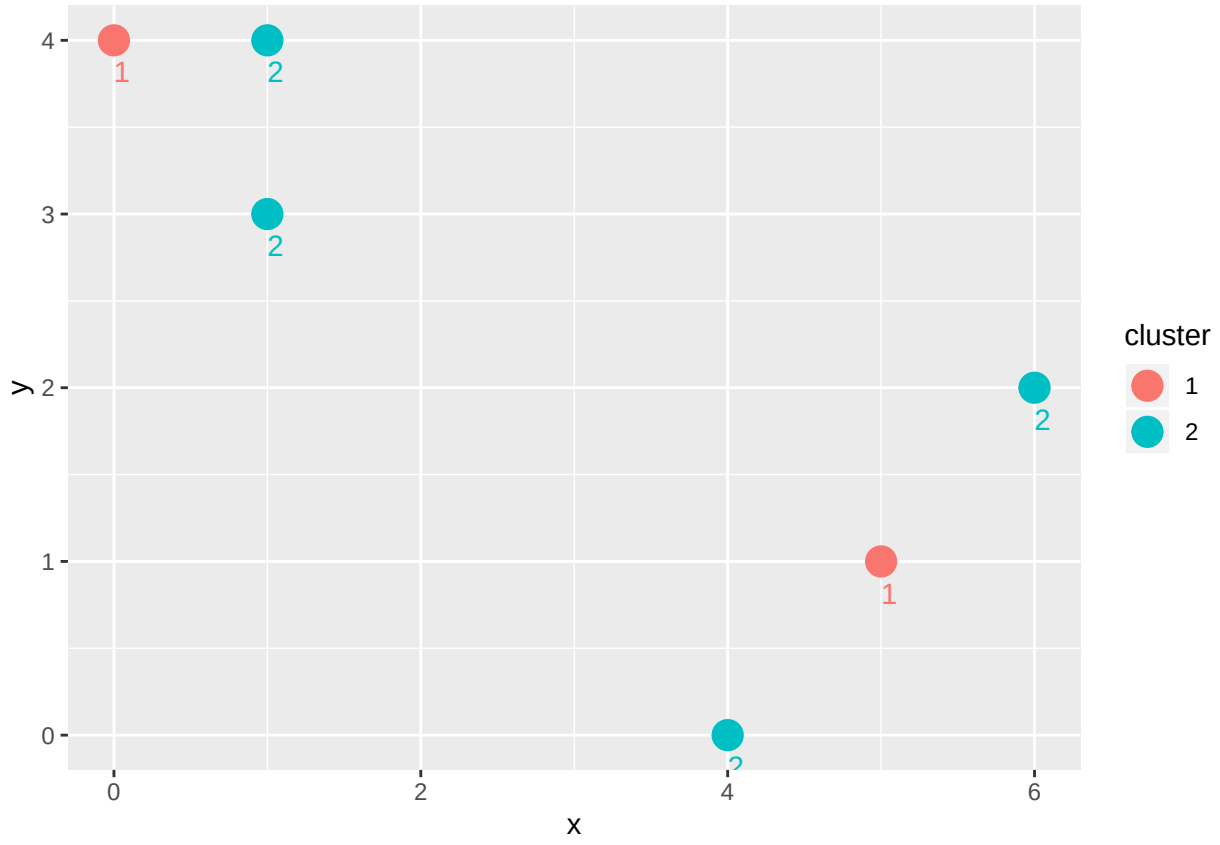
1. (5 points) Plot the observations.

```
x <- cbind(c(1, 1, 0, 5, 6, 4), c(4, 3, 4, 1, 2, 0))
x = as.data.frame(x)
colnames(x) = cbind ("x", "y")
ggplot(x, aes(x = x, y =y)) + geom_point(size = 5)
```



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*).

```r
set.seed(122)
random = sample(1:2, 6, replace=TRUE)
x$cluster = random
x$cluster = as.factor(x$cluster)
ggplot(x, aes(x = x, y =y, color=cluster, label = cluster)) + geom_point(size=5) + geom_text(aes(label=
```



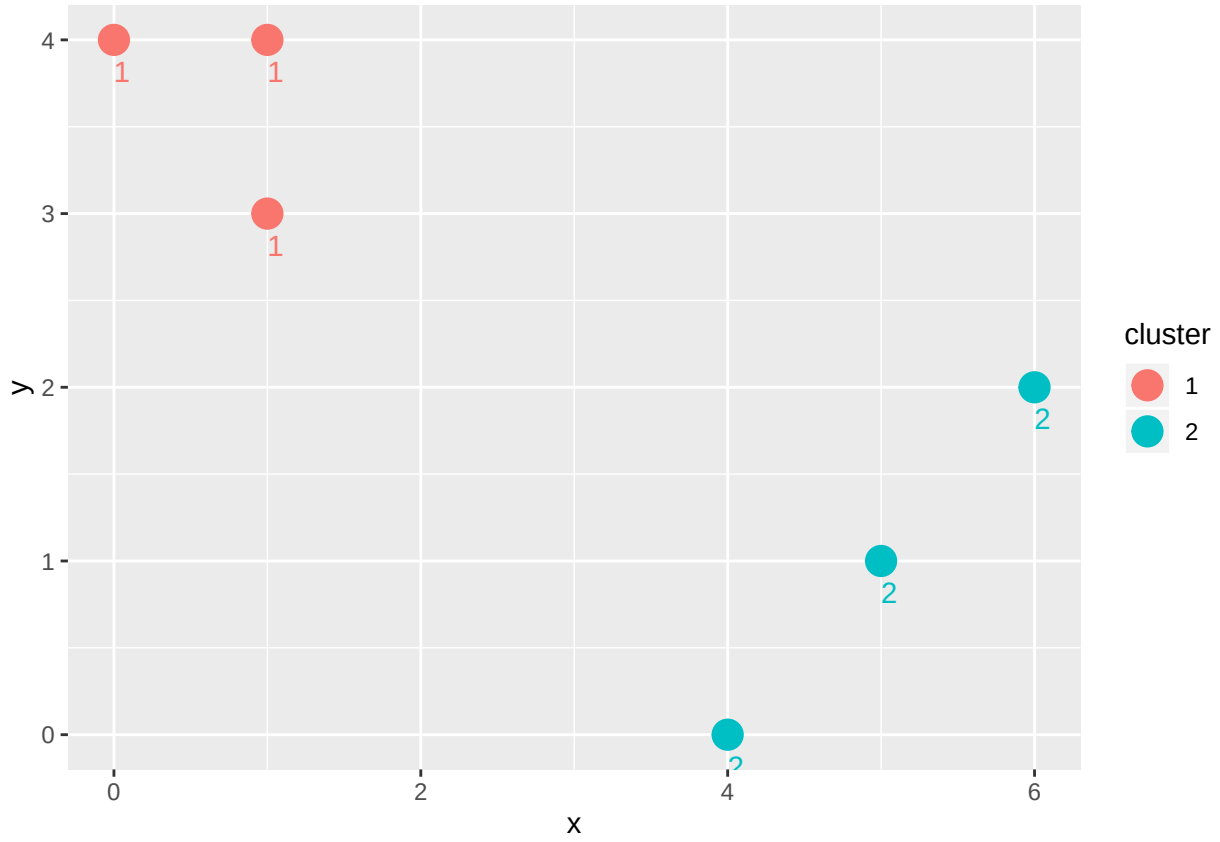
3. (10 points) Compute the centroid for each cluster.

```r
centroid_x = c()
centroid_x[1] = mean(x$x[which(x$cluster==1)])
centroid_x[2] = mean(x$x[which(x$cluster==2)])

centroid_y = c()
centroid_y[1] = mean(x$y[which(x$cluster==1)])
centroid_y[2] = mean(x$y[which(x$cluster==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.

```r
x$cluster1_distance = ((x$x - centroid_x[1])^2) + ((x$y - centroid_y[1])^2)
x$cluster2_distance = ((x$x - centroid_x[2])^2) + ((x$y - centroid_y[2])^2)

for (i in 1:6) {
x$cluster[i] = ifelse(x$cluster1_distance[i]< x$cluster2_distance[i],"1","2")
```

```
}

ggplot(x, aes(x = x, y =y, color=cluster, label = cluster)) + geom_point(size=5) + geom_text(aes(label=
```

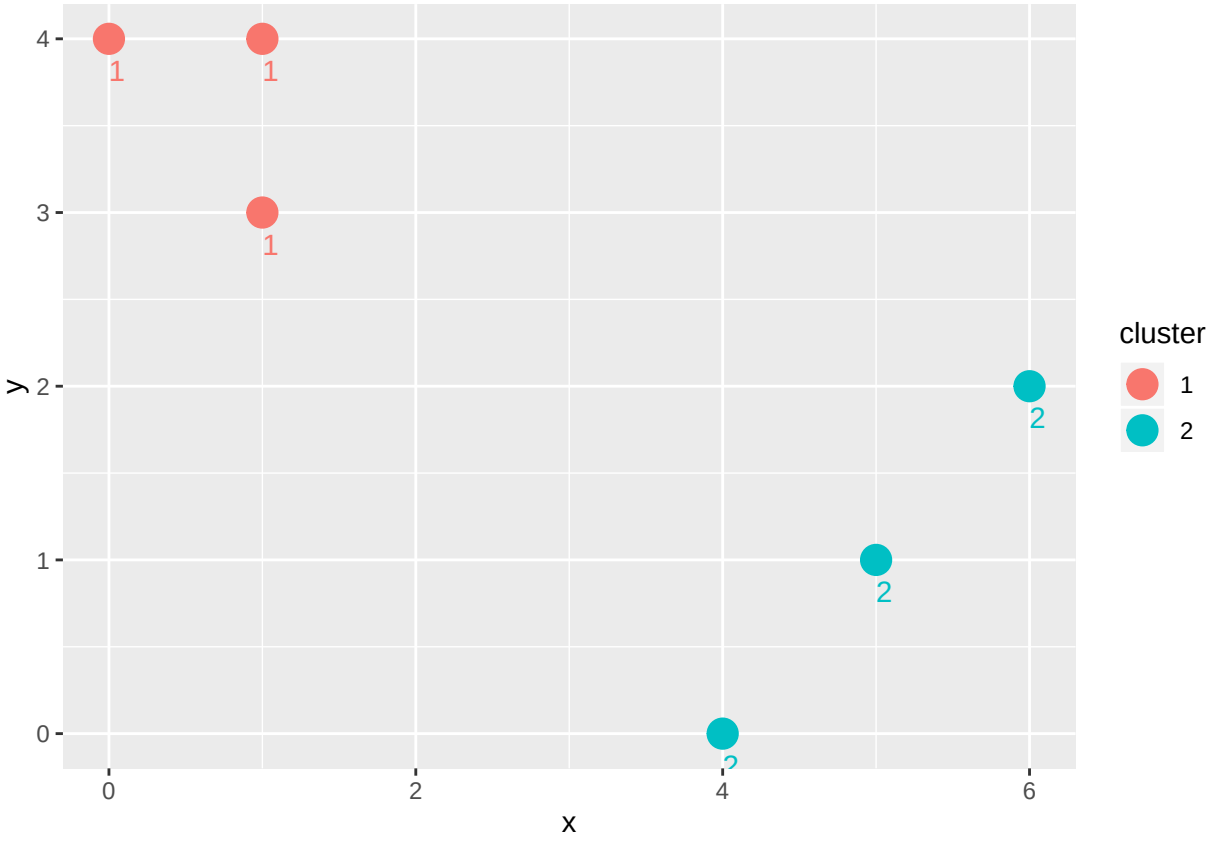


5. (5 points) Repeat (3) and (4) until the answers/clusters stop changing.

```
centroid_x = c()
centroid_x[1] = mean(x$x[which(x$cluster==1)])
centroid_x[2] = mean(x$x[which(x$cluster==2)])

centroid_y = c()
centroid_y[1] = mean(x$y[which(x$cluster==1)])
centroid_y[2] = mean(x$y[which(x$cluster==2)])

x$cluster1_distance = ((x$x - centroid_x[1])^2) + ((x$y - centroid_y[1])^2)
x$cluster2_distance = ((x$x - centroid_x[2])^2) + ((x$y - centroid_y[2])^2)

for (i in 1:6) {
x$cluster[i] = ifelse(x$cluster1_distance[i]< x$cluster2_distance[i],"1","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.

```
ggplot(x, aes(x = x, y =y, color=cluster, label = cluster)) + geom_point(size=5) + geom_text(aes(label=
```

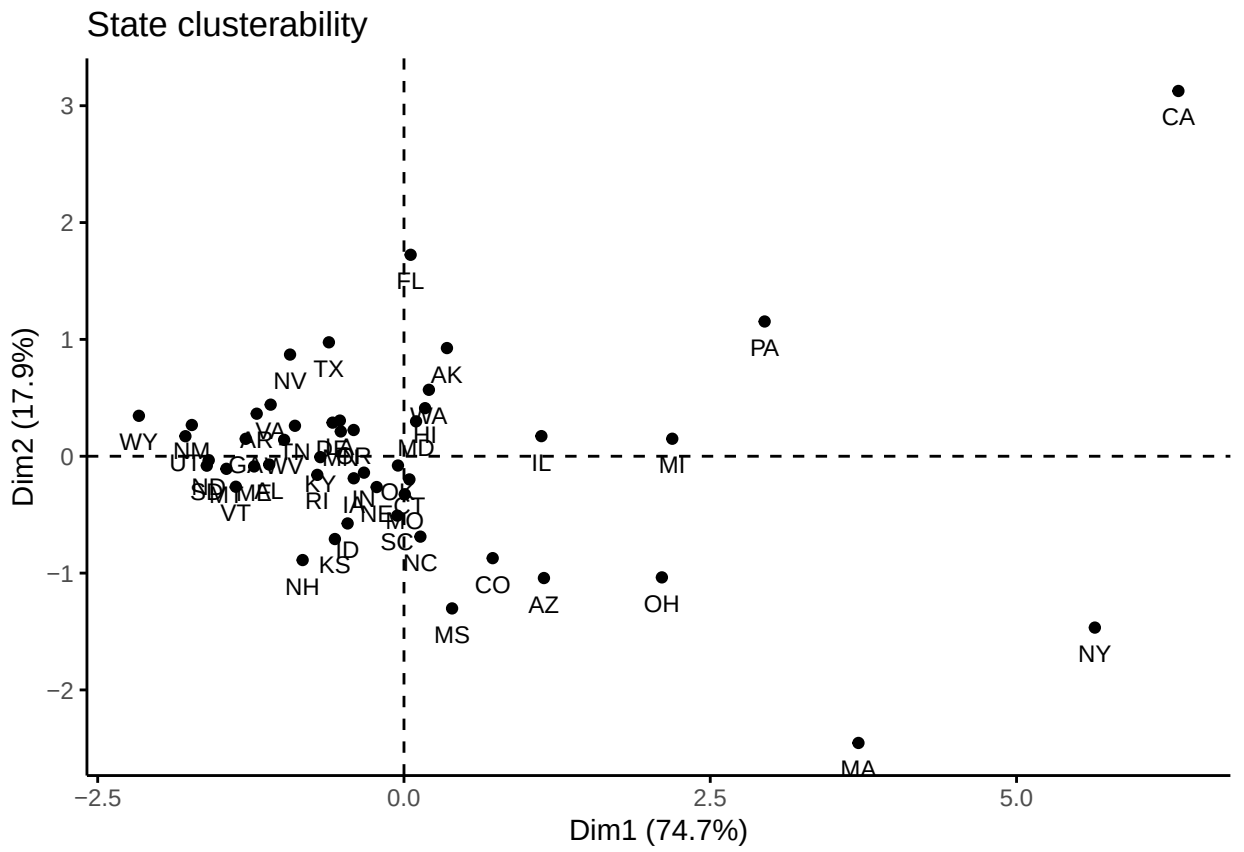


Clustering State Legislative Professionalism

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

```
# Plot faithful data set
fviz_pca_ind(prcomp(x), title = "State clusterability",
             palette = "jco",
```

```
            geom = "point", ggtheme = theme_classic(),
            legend = "bottom") + geom_text(label = state$stateabv, size=3, vjust=2)
```




```
get_clust_tendency(x, n =2, graph=FALSE)
```

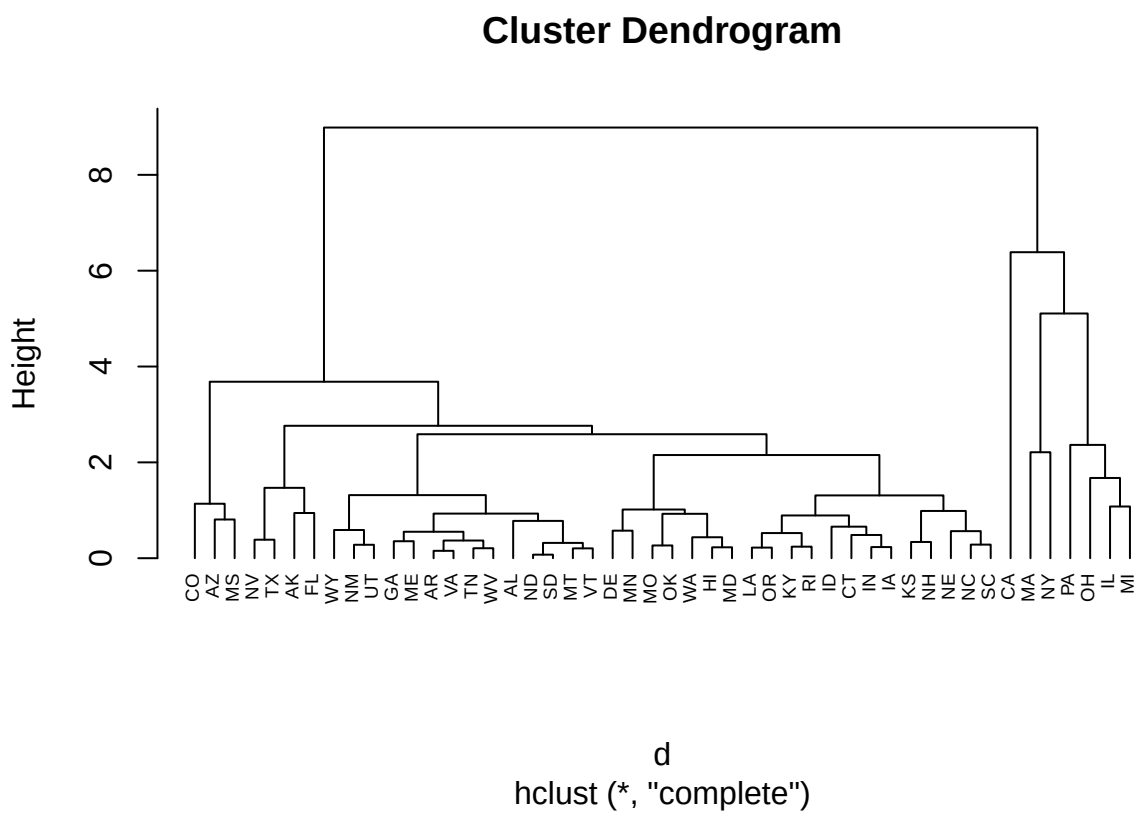
```
## $hopkins_stat
## [1] 0.8053219
##
## $plot
## NULL
```

The Hopkins clusterability tendency is 0.81, so it is very likely that there are clusters within this dataset. To calculate the Hopkin's statistic, the distances between datapoints in our dataset is compared to the distance between datapoints in a random dataset with the same variation as our original dataset. If distances between the points in the random dataset are much further away than distances between points in our dataset that suggests that there is clusterability. The PCA suggests that there visually appears to be some clustering.

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.

```
d <- dist(x, method = "euclidean")
```

```r
# Hierarchical clustering using Complete Linkage
hc1 <- hclust(d, method = "complete" )

states = state$stateabv

# Plot the obtained dendrogram
plot(hc1, cex=0.6, hang=-1, labels= states)
```




```r
hc_cluster <- cutree(hc1,2)
```

In agglomorative hierarchical clustering, the number of clusters can be decided by visual inspection of the dendrogram and deciding the height cut off. There appears to be 2 main clusters. California seems to be very separate from the other states.

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
k = kmeans(x, 2, iter.max = 10, nstart = 2,
      trace=FALSE)

plotx$state = state$stateabv
plotx$cluster = k$cluster
plotx$cluster=as.factor(plotx$cluster)
kplot = ggplot(plotx, aes(x = plotx$salary_real, y=plotx$expend, color=plotx$cluster)) + geom_point(siz
```

The two clusters were very uneven, with 6 states in one cluster, and 42 in the other cluster. California is an outlier, and seems to pull the centroid towards it and may be separating states from their true cluster membership.

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.

```
k = kmeans(x, 2, iter.max = 10, nstart = 2,
      trace=FALSE)

mixmdl <- mvnormalmixEM(x, k = 2)
```

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

```
mixmdl$mu
```

```
## [[1]]
## [1] 1.760823 1.823237 1.991799 1.261337
##
## [[2]]
## [1] -0.2984658 -0.3090452 -0.3376171 -0.2138011
```

```
mixmdl$sigma
```

```
## [[1]]
##           [,1]        [,2]        [,3]       [,4]
## [1,] 1.3336213  1.26278604  0.19973688 0.88085713
## [2,] 1.2627860  1.42726289 -0.06306026 0.01552892
## [3,] 0.1997369 -0.06306026  0.36601909 1.08027453
## [4,] 0.8808571  0.01552892  1.08027453 3.56746762
##
## [[2]]
##            [,1]        [,2]       [,3]        [,4]
## [1,] 0.30445820 0.254437008 0.06709800 0.007139260
## [2,] 0.25443701 0.244241231 0.07441893 0.002079211
## [3,] 0.06709800 0.074418934 0.29664647 0.079452049
## [4,] 0.00713926 0.002079211 0.07945205 0.225054344
```

```
mixmdl$lambda
```

```
## [1] 0.1449363 0.8550637
```

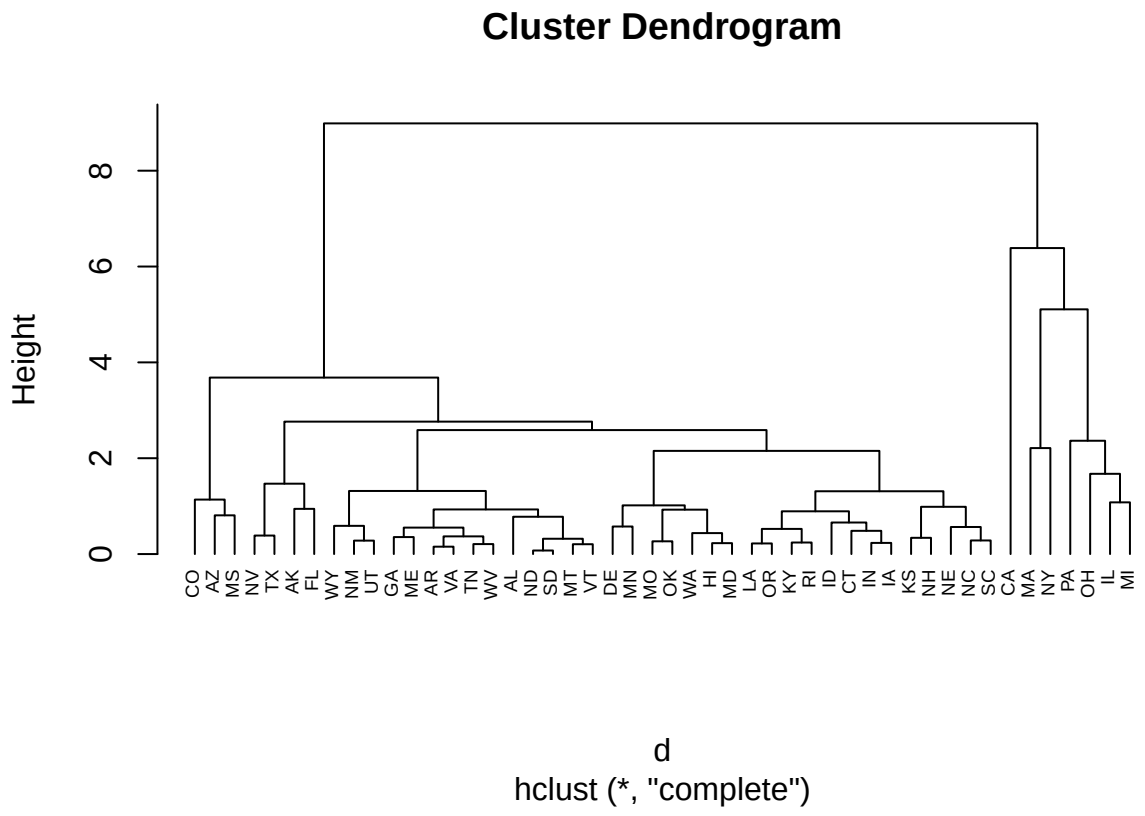
```
plotx$c1 = mixmdl$posterior[,1]
plotx$c2 = mixmdl$posterior[,2]
plotx$cluster = ifelse(plotx$c1>plotx$c2, 1, 2)
plotx$cluster=as.factor(plotx$cluster)
gmm_cluster = as.numeric(plotx$cluster)

gmmplot1 = ggplot(plotx, aes(x=plotx$t_slength, color=plotx$cluster)) +
```

```
  geom_density() + labs(title="GMM model")

gmmplot2 = ggplot(plotx, aes(x=plotx$salary_real, color=plotx$cluster)) +
  geom_density()+ labs(title="GMM model")
```

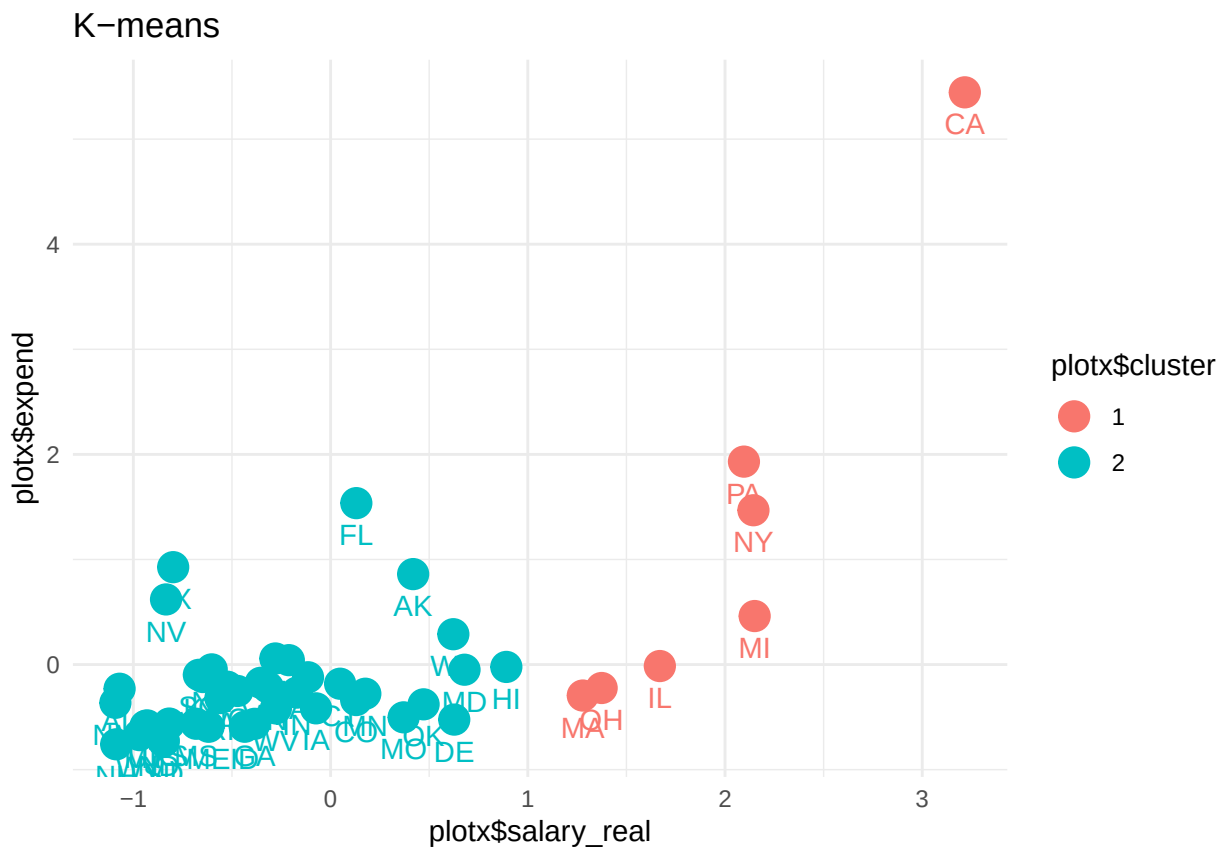
There are two clusters with weights of 0.125 and 0.875. Similar to K-means, one cluster has the majority of observations. The means of the four components are different between the 2 clusters (mu1 and mu2). Sigma is the standard deviation of the components between the 2 clusters.

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.

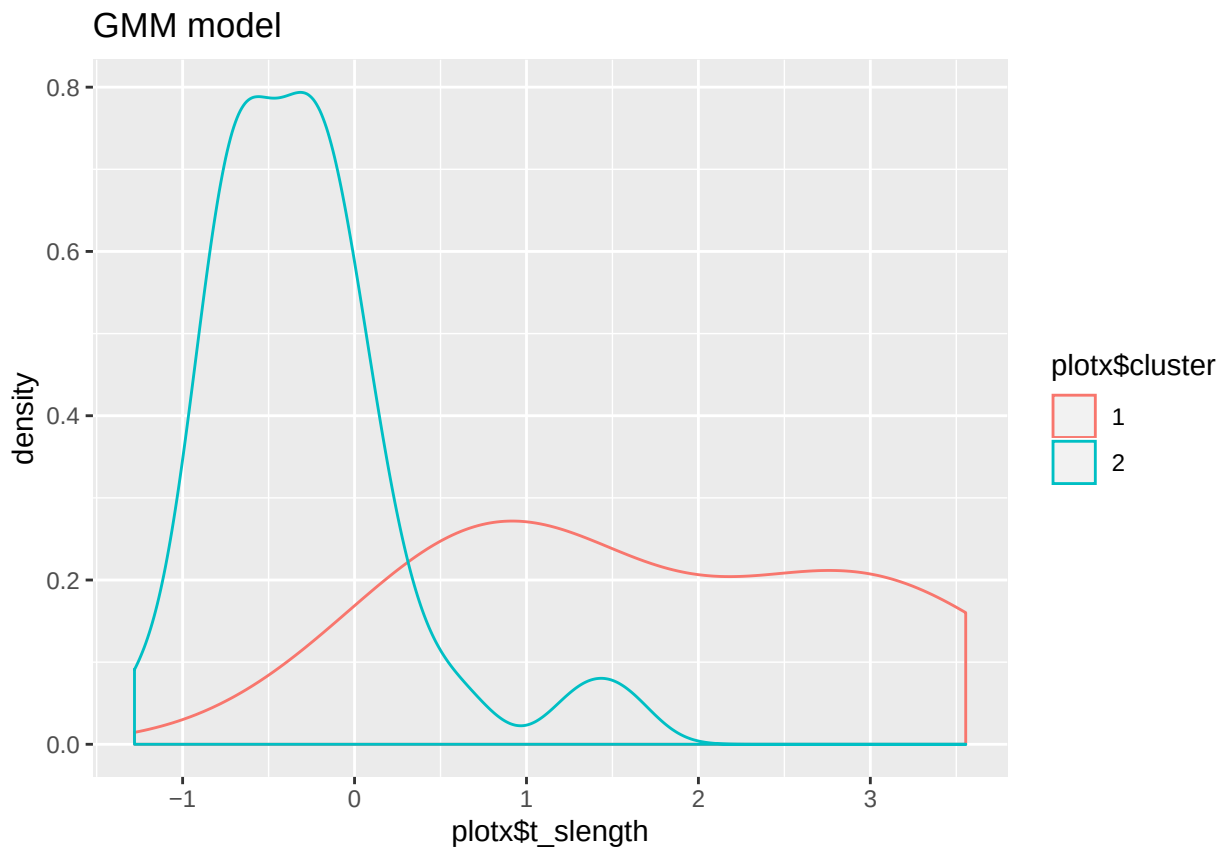
```
plot(hc1, cex=0.6, hang=-1, labels= states)
```



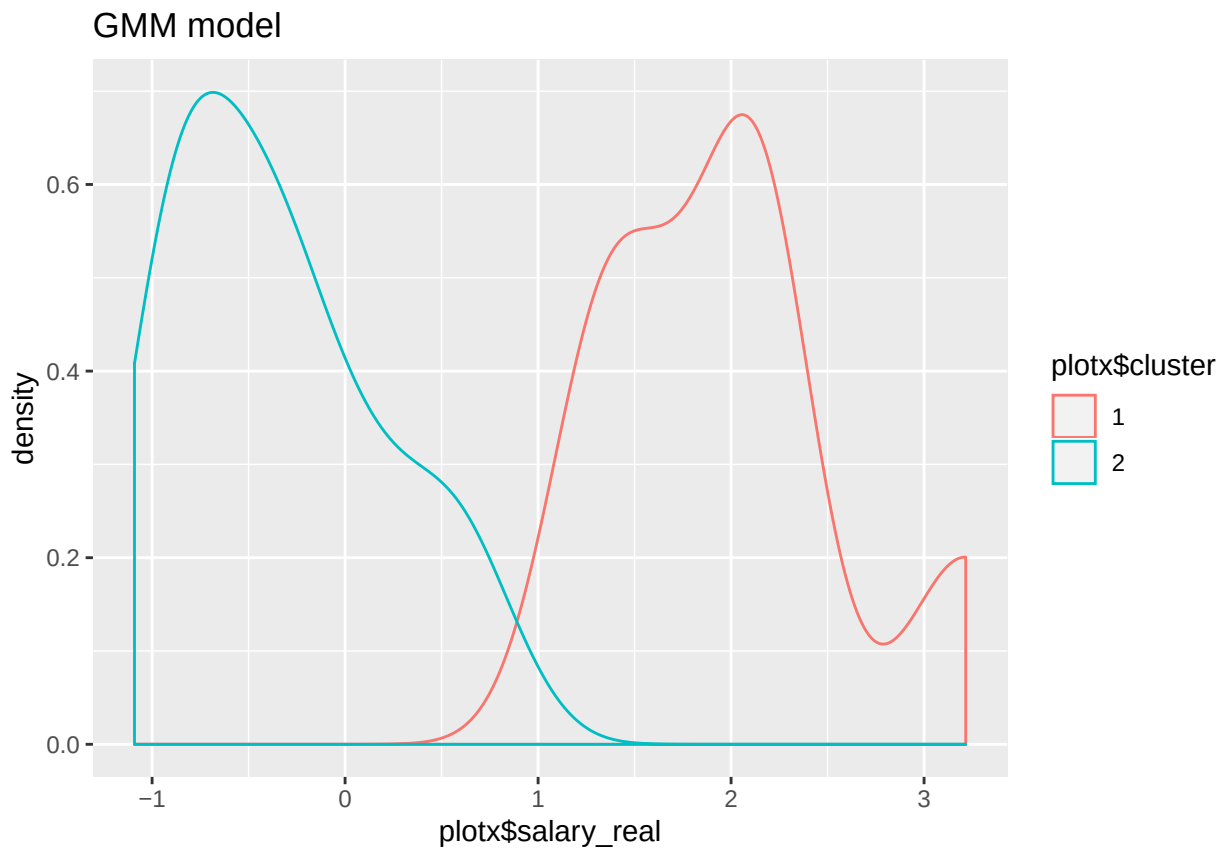

```
kplot
```


K−means
plotx$expend
plotx$salary_real
plotx$cluster
1
2
CA
PA
NY
MI
IL
MA
OH
FL
AK
NV
HI
MD
DE
MO
4
2
0
−1
0
1
2
3

```
gmmplot1
```


GMM model
0.8
0.6
0.4
0.2
0.0
density
−1
0
1
2
3
plotx$t_slength
plotx$cluster
1
2

```
gmmplot2
```



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.

```
dist_dunn <- dist(x, method = "euclidean")

dunn_value_k <- dunn(distance = dist_dunn, clusters = k$cluster)
dunn_value_hc <- dunn(distance = dist_dunn, clusters = hc_cluster)
dunn_value_gmm <- dunn(distance = dist_dunn, clusters = gmm_cluster)

dunn_value_k
```

```
## [1] 0.1725627
```

```
dunn_value_hc
```

```
## [1] 0.1673953
```

```
dunn_value_gmm
```

```
## [1] 0.1673953
```

The Dunn index is a metric for assessing the quality of clustering, with higher indices corresponding to better clusters. In general terms, it evaluates for maximization of inter-cluster distances and minimization of intra-cluster distances. The Dunn index for the k-means is highest at 0.17, followed by heirarchical clustering at 0.167, and GMM at 0.167.

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?

```
k_3 = kmeans(x, 3, iter.max = 10, nstart = 2,
      trace=FALSE)
k_4 = kmeans(x, 4, iter.max = 10, nstart = 2,
      trace=FALSE)
k_5 = kmeans(x, 5, iter.max = 10, nstart = 2,
      trace=FALSE)
k_6 = kmeans(x, 6, iter.max = 10, nstart = 2,
      trace=FALSE)

dist_dunn <- dist(x, method = "euclidean")
dunn_k3 <- dunn(distance = dist_dunn, clusters = k_3$cluster)
dunn_k4 <- dunn(distance = dist_dunn, clusters = k_4$cluster)
dunn_k5 <- dunn(distance = dist_dunn, clusters = k_5$cluster)
dunn_k6 <- dunn(distance = dist_dunn, clusters = k_6$cluster)
```

The k-means approach had the lowest Dunn index. The Dunn index continues to decrease with more clusters. The context of the problem would likely determine the clustering method selection more than the Dunn index. For instance, if the grouping using larger number of clusters revealed a geographic variation in states that is scientifically interesting and revealing, that may call for using that number of clusters rather than an arbitrary and assumption-less metric of intra- and inter-cluster distances.