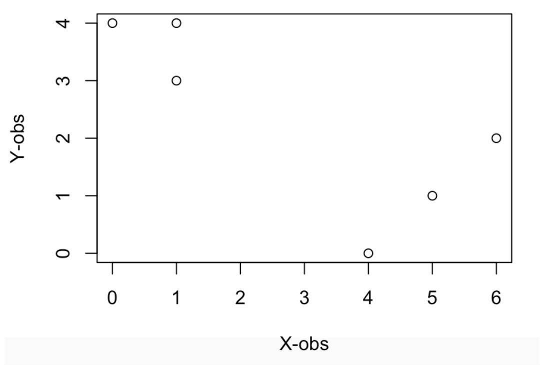
Rahul Rangwani
Problem Set 4
March 2, 2020
Intro to Machine Learning

Performing k-Means by Hand

1. To plot the data, I ran the following code:

```
x <- cbind(c(1, 1, 0, 5, 6, 4), c(4, 3, 4, 1, 2, 0))
plot(x, xlab="X-obs", ylab="Y-obs")
```

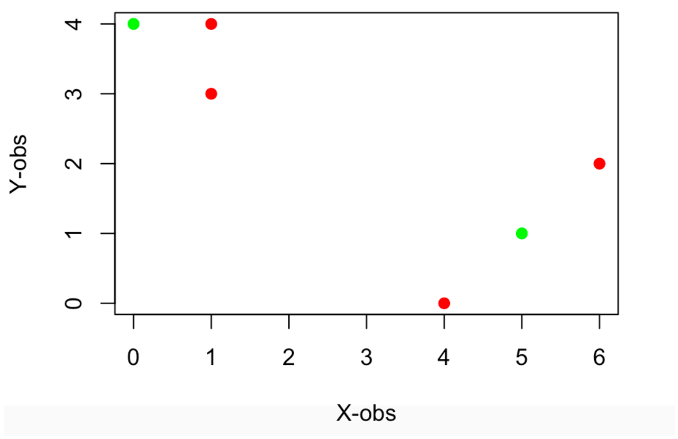
   And generated the following graph:



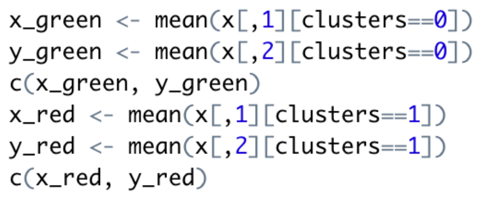
2. For this, I ran the following code:

```
set.seed(17)
clusters <- sample(seq(0,1), size=6, replace=TRUE)
plot(x[,1], x[,2], col=ifelse(clusters != 0,'red','green'), pch = 19, xlab="X-obs", ylab="Y-obs")
```

   And got the following plot:

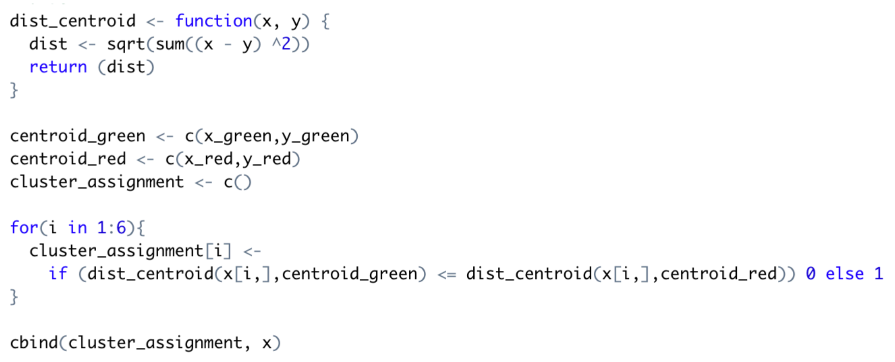


3. The centroid for the red cluster is (3.00,2.25) and for the green cluster is (2.5, 2.5). To get these values, I ran the following code:



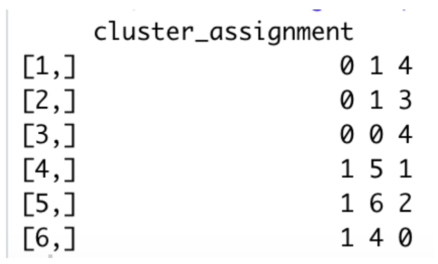
```
x_green <- mean(x[,1][clusters==0])
y_green <- mean(x[,2][clusters==0])
c(x_green, y_green)
x_red <- mean(x[,1][clusters==1])
y_red <- mean(x[,2][clusters==1])
c(x_red, y_red)
```

4. To determine the distances, I ran the following code:



```
dist_centroid <- function(x, y) {
  dist <- sqrt(sum((x - y) ^2))
  return (dist)
}

centroid_green <- c(x_green,y_green)
centroid_red <- c(x_red,y_red)
cluster_assignment <- c()

for(i in 1:6){
  cluster_assignment[i] <-
    if (dist_centroid(x[i,],centroid_green) <= dist_centroid(x[i,],centroid_red)) 0 else 1
}

cbind(cluster_assignment, x)
```

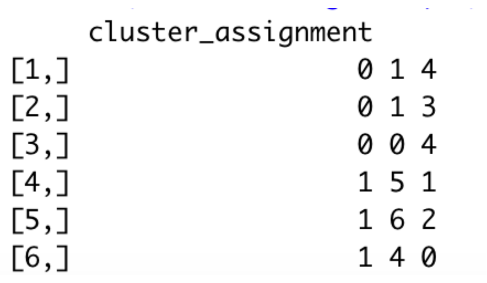
These are the assignments:



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

Two of the observations got flipped from red to green, and one from green to red.

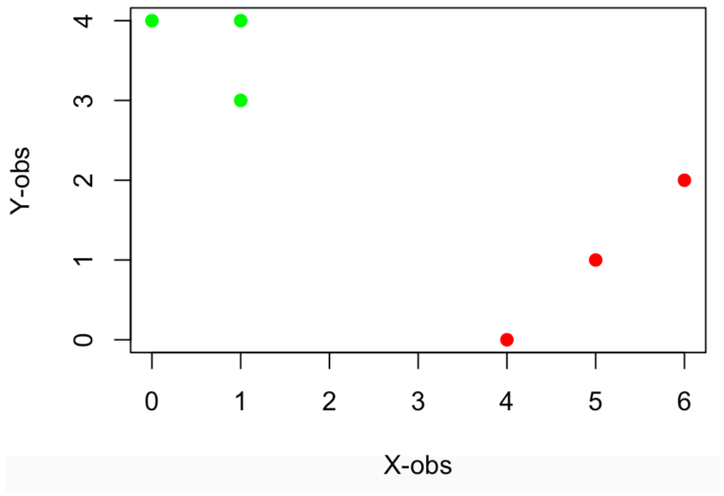
5. After running the code from 3 and 4 again, we see that the assignments didn't change. The results are the same as above:



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

6. I ran this code to get the plot:

```
plot(x[,1],
     x[,2],
     col=ifelse(cluster_assignment != 0,'red','green'),
     pch = 19,
     xlab="X-obs",
     ylab="Y-obs")
```



Clustering State Legislative Professionalism

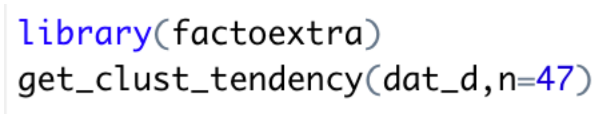
1. Loading the data:

```
dat <- load('legprof-components.v1.0.RData')
dat <- x
```

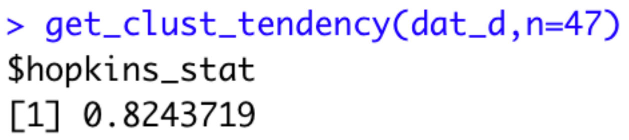
2. To munge the data, I did the following:

```
dat_a <- select(dat, "t_slength", "slength", "salary_real", "expend", "year", "state")

dat_b <- subset(dat_a, year==2009 | year==2010)

dat_c <- na.omit(dat_b)
state_names = dat_c$state

dat_d <- select(dat_c, "t_slength", "slength", "salary_real", "expend")
dat_d <- scale(dat_d)
```

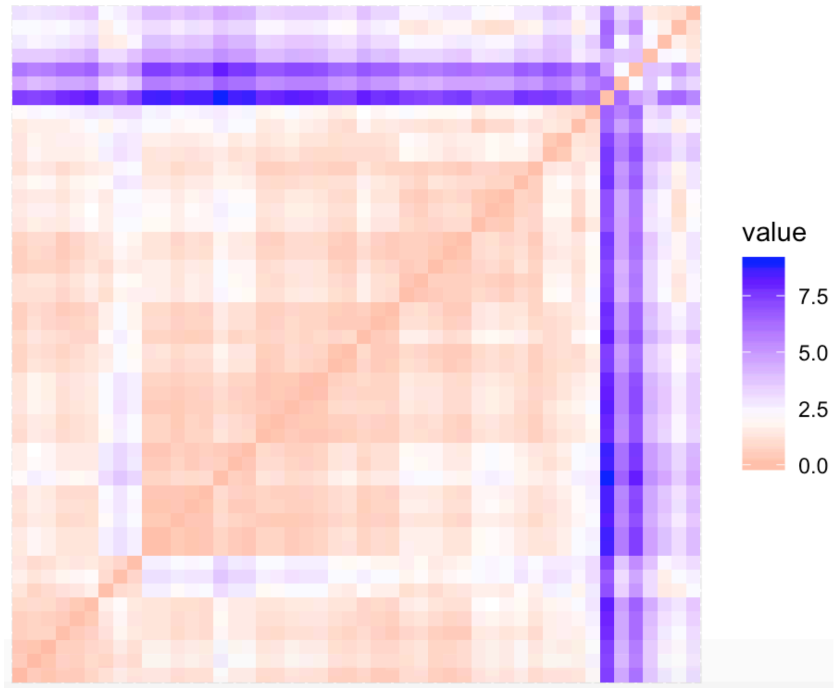
3. To assess clusterability, I used an ODI plot. We can see many adjacent red and white squares in the plot. This, along with our Hopkins stat being close to 1 (.824), we can see that there is some non-random structure and similarity among the data.



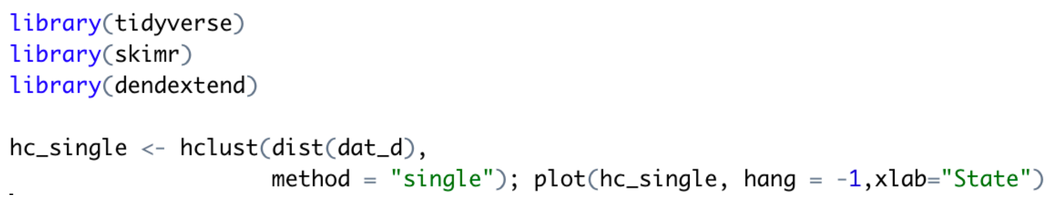
```
library(factoextra)
get_clust_tendency(dat_d,n=47)
```



```
> get_clust_tendency(dat_d,n=47)
$hopkins_stat
[1] 0.8243719
```

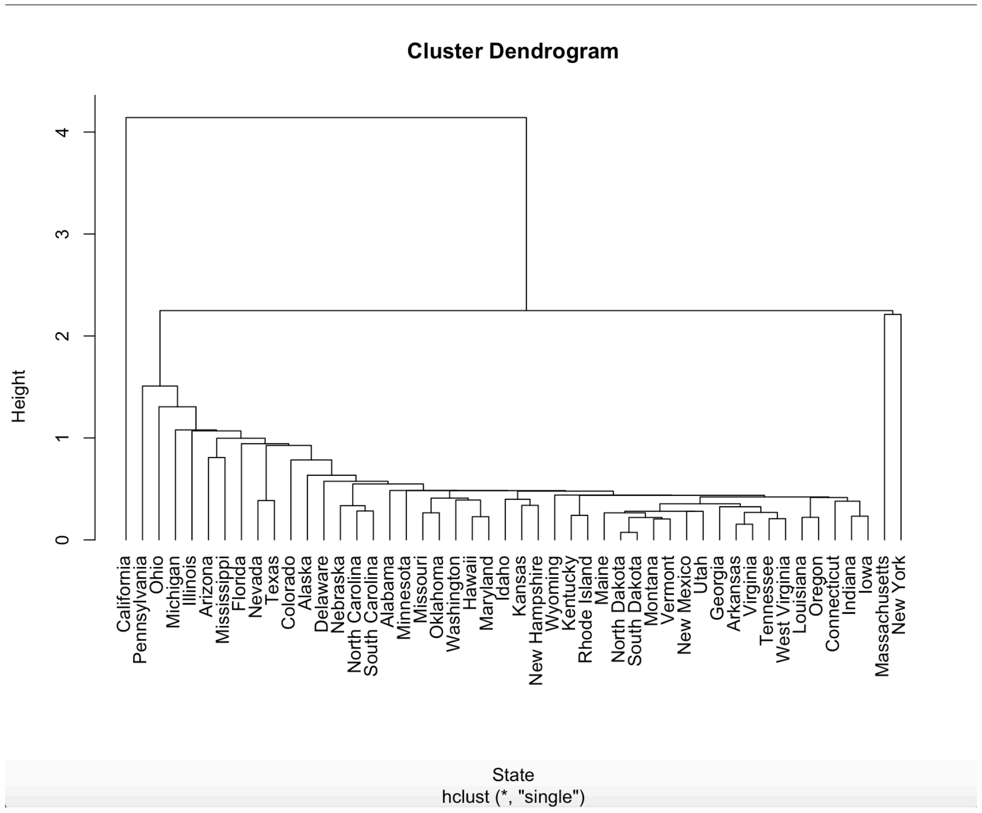


4. To run an agglomerative hierarchical clustering algorithm, I ran the following code:



```
library(tidyverse)
library(skimr)
library(dendextend)

hc_single <- hclust(dist(dat_d),
                    method = "single"); plot(hc_single, hang = -1,xlab="State")
```

My results were as follows:

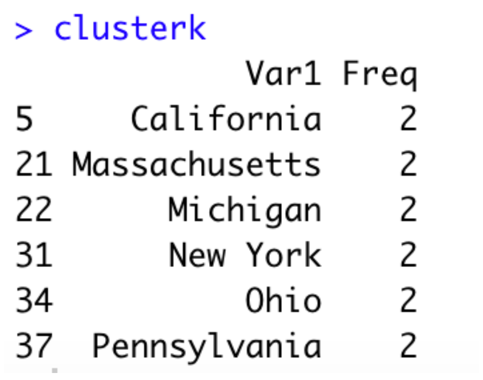


I used the single linkage method. Here, we can see that there are three states that are very dissimilar to the others: California, Massachusetts, and New York. We can see that, not surprisingly, North and South Carolina and North and South Dakota are very similar. Indiana and Iowa are another similar pair, which makes sense due to the geographic similarity and location. An unusual, or unexpected, similarity is between Hawaii and Maryland. The only patterns I see are with location. I can't say why Hawaii and Maryland are similar, but it makes sense that North and South Dakota are similar.

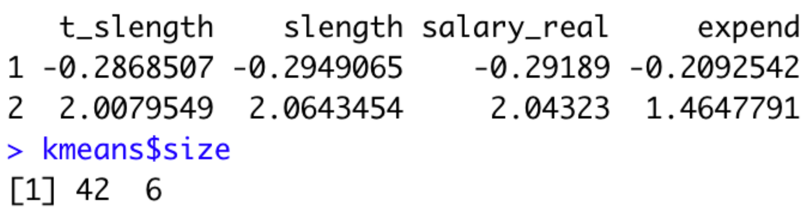
5. To run a k-means fit, I ran the following code:

```
kmeans <- kmeans(dat_d,
                 centers = 2,
                 nstart = 30)

t <- as.table(kmeans$cluster)
t <- data.frame(t)
clusterk <- t[t$Freq == "2",]
clusterk

str(kmeans)

kmeans$cluster
kmeans$centers
kmeans$size
```

The results were:



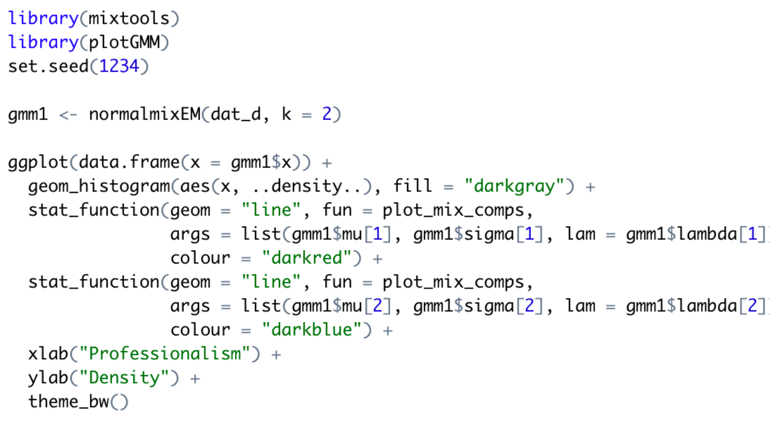
```
> clusterk
           Var1 Freq
5    California    2
21 Massachusetts   2
22     Michigan    2
31     New York    2
34         Ohio    2
37  Pennsylvania   2
```



```
   t_slength    slength salary_real     expend
1 -0.2868507 -0.2949065    -0.29189 -0.2092542
2  2.0079549  2.0643454     2.04323  1.4647791
> kmeans$size
[1] 42  6
```

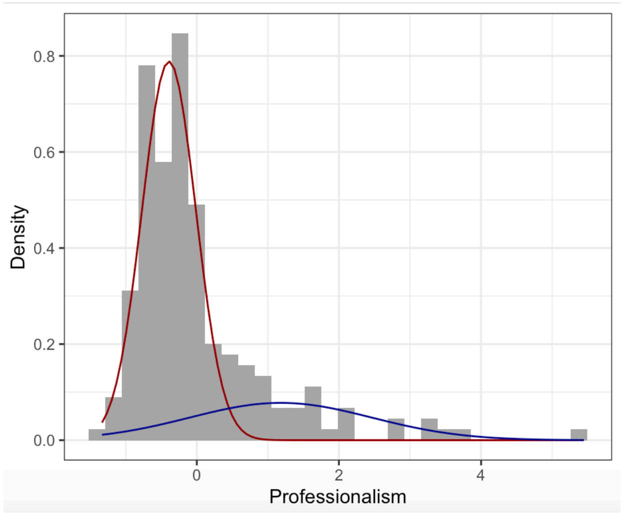
What we see here is that the two clusters are composed of 42 states, and 6 states. The 6 states in cluster 2 are shown above. Looking at the breakdown of the clusters, we can see that the states in cluster 2 have much higher values in each variable of interest.

6. To run the GMM, I ran the following code:



```
library(mixtools)
library(plotGMM)
set.seed(1234)

gmm1 <- normalmixEM(dat_d, k = 2)

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

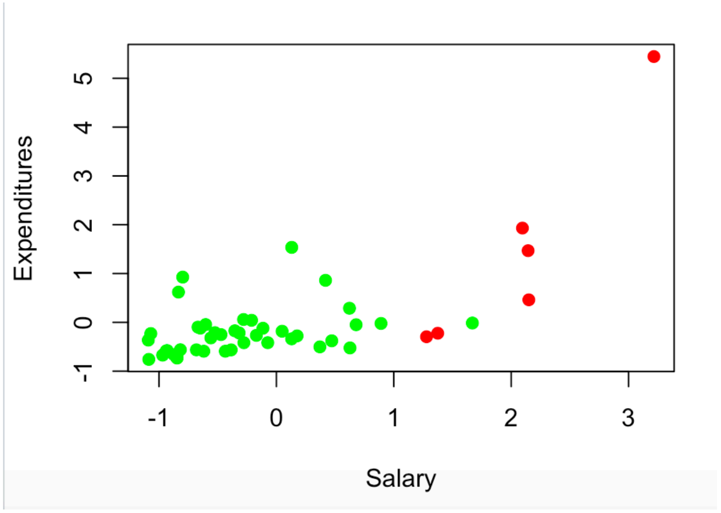
This was the resulting plot:



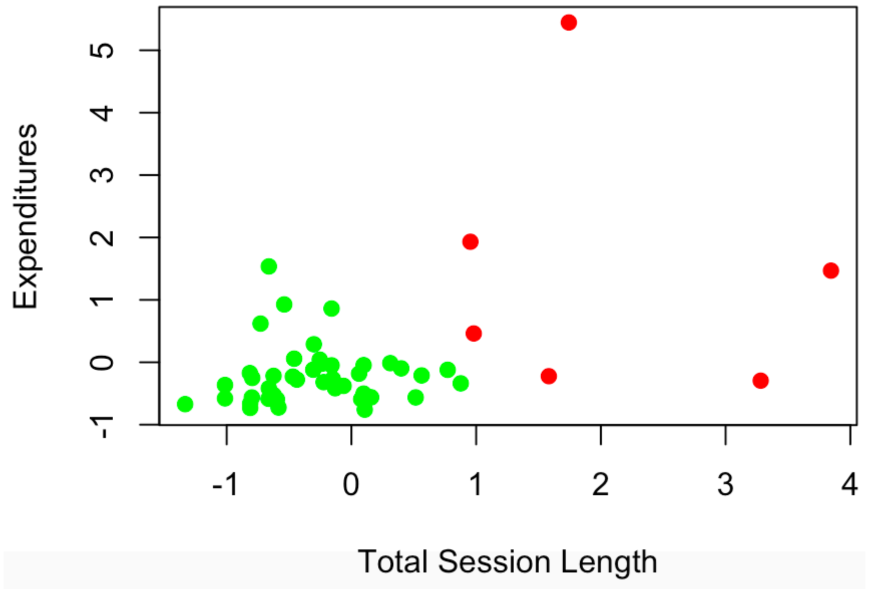
From this, we can see that there are again two distinct groups. One teds to more professionalism, but with much lower density (or frequency), than the other group which is much less professional with higher density. This reflects what we found in the k-means clustering.

7. First, we can see that on the dendrogram above, the 6 states that are most dissimilar are the same as we saw in the k-means clustering algorithm. To further explore this, I plotted the values of different variables and colored them based on assignment.

```
plot(dat_d[,3],
     dat_d[,4],
     col=ifelse(assignments != 0,'red','green'),
     pch = 19,
     xlab="Salary",
     ylab="Expenditures")
```



```r
plot(dat_d[,2],
     dat_d[,4],
     col=ifelse(assignments != 0,'red','green'),
     pch = 19,
     xlab="Total Session Length",
     ylab="Expenditures")
```



From these graphs, we see that there are two fairly distinct groups of states, and that the clustering of these groups makes sense. If we compare key values from the k-means and GMM models, we see this:

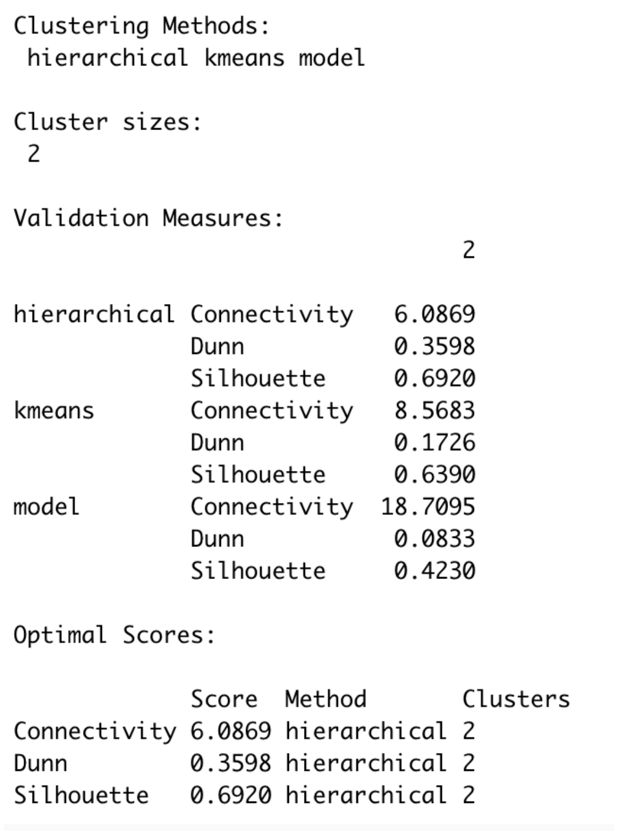
```
> kmeans$centers
   t_slength    slength salary_real     expend
1 -0.2868507 -0.2949065    -0.29189 -0.2092542
2  2.0079549  2.0643454     2.04323  1.4647791
> gmm1$mu
[1] -0.3919019  1.1807282
```

We see here that the centers for each cluster in both models is somewhat similar in magnitude and sign.

8. To run this validation check, I used the clValid library:

```r
library(clValid)
library(mclust)
validation_check <- clValid(dat_d,
                            c(2),
                            clMethods = c("hierarchical", "kmeans","model"),
                            validation = c("internal")); summary(validation_check)
```

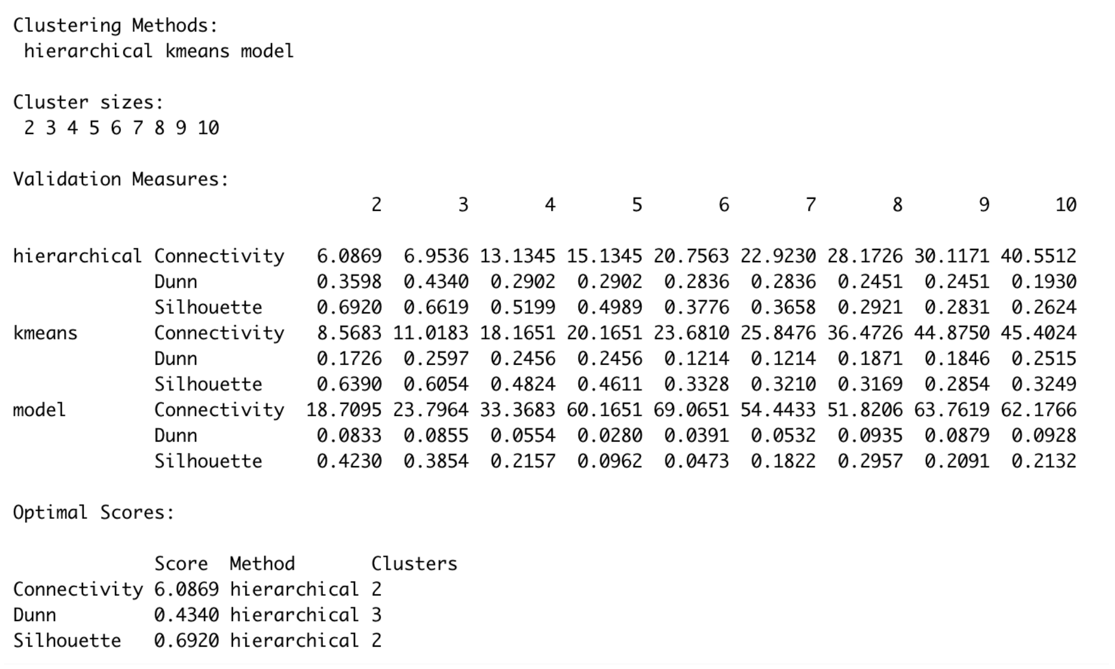
The outputs were:



```
Clustering Methods:
 hierarchical kmeans model

Cluster sizes:
 2

Validation Measures:
                                 2

hierarchical Connectivity   6.0869
             Dunn           0.3598
             Silhouette     0.6920
kmeans       Connectivity   8.5683
             Dunn           0.1726
             Silhouette     0.6390
model        Connectivity  18.7095
             Dunn           0.0833
             Silhouette     0.4230

Optimal Scores:

             Score  Method       Clusters
Connectivity 6.0869 hierarchical 2
Dunn         0.3598 hierarchical 2
Silhouette   0.6920 hierarchical 2
```

What we see here is that The silhouette scores for each model are fairly similar, with values of .69, .63, and .42. What this tells us is that first, the hierarchical model performed the best, with k-means coming in a close second, and then the GMM model following in last. If we run this with different cluster sizes, we get the following:

```
validation_check <- clValid(dat_d,
                            c(2:10),
                            clMethods = c("hierarchical", "kmeans","model"),
                            validation = c("internal")); summary(validation_check)
```



```
Clustering Methods:
 hierarchical kmeans model

Cluster sizes:
 2 3 4 5 6 7 8 9 10

Validation Measures:
                                 2       3       4       5       6       7       8       9      10

hierarchical Connectivity   6.0869  6.9536 13.1345 15.1345 20.7563 22.9230 28.1726 30.1171 40.5512
             Dunn           0.3598  0.4340  0.2902  0.2902  0.2836  0.2836  0.2451  0.2451  0.1930
             Silhouette     0.6920  0.6619  0.5199  0.4989  0.3776  0.3658  0.2921  0.2831  0.2624
kmeans       Connectivity   8.5683 11.0183 18.1651 20.1651 23.6810 25.8476 36.4726 44.8750 45.4024
             Dunn           0.1726  0.2597  0.2456  0.2456  0.1214  0.1214  0.1871  0.1846  0.2515
             Silhouette     0.6390  0.6054  0.4824  0.4611  0.3328  0.3210  0.3169  0.2854  0.3249
model        Connectivity  18.7095 23.7964 33.3683 60.1651 69.0651 54.4433 51.8206 63.7619 62.1766
             Dunn           0.0833  0.0855  0.0554  0.0280  0.0391  0.0532  0.0935  0.0879  0.0928
             Silhouette     0.4230  0.3854  0.2157  0.0962  0.0473  0.1822  0.2957  0.2091  0.2132

Optimal Scores:

             Score  Method       Clusters
Connectivity 6.0869 hierarchical 2
Dunn         0.4340 hierarchical 3
Silhouette   0.6920 hierarchical 2
```

As we can see, the silhouette scores decrease for each model after 2 clusters, which tells us that k=2 is the optimal number of clusters for this data.

9. From all of this analysis, what we see is that using the silhouette scores, all of our models perform best on k=2 clusters. However, if we look at the Dunn score, our hierarchical model performs best at k=3 clusters. This tells us that our model may not be perfect at k=2. However, with the silhouette score, the hierarchical model score is fairly close to 1 (.692), which indicates a better fit. It seems that our hierarchical model is best at a specification of k=2.

   We may select a “sub-optimal” clustering method due to the way a clustering algorithm works. If we select k=n(obs), then we’d have a perfectly fit model. However, the variance would be very high and the model wouldn’t actually tell us anything. If we can use a smaller amount of clusters, resulting in an imperfect model, it still may be much more informative than a “perfectly fit” model as described above. There is a tradeoff in choosing parameters, and often, we must choose a “sub-optimal” clustering method in order to have results that are informative.